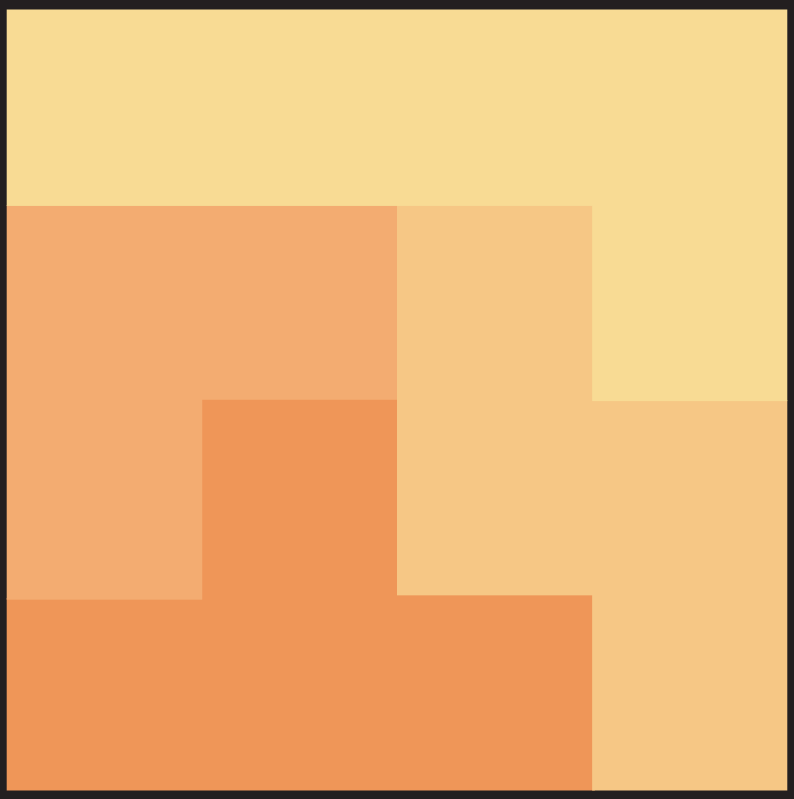
Влад Мержевич



Вёрстка веб-страниц

htmlbook.ru

Вступление

HTML изначально задумывался как язык, которому не нужны средства оформления, такие как цвет, размер, рамки или что-то подобное. Разработанный в Европейском институте физики частиц (CERN), HTML был игрушкой ученых, их, прежде всего, интересовала логика информации, а не её визуальное представление. Тогда ещё не существовало понятия веб-дизайна и вёрстки как таковой, все сайты по своему оформлению были практически однотипными, в стиле, называемом сейчас «академический дизайн». Пример до сих пор можно посмотреть на lib.ru.

Однако пользователи преимущественно думали иначе и, глядя на красивую картинку, отдавали предпочтение именно ей. Разработчики браузера Netscape понимали это и вводили в HTML новые теги, улучшающие внешний вид документа. Эти теги не были стандартизированы и работали только в Netscape, что в эпоху повального засилья этого браузера, не имело значения. Доля Netscape составляла более 90% от всех существующих браузеров.

Влияние Netscape оказалось губительным для академического дизайна, фактически похоронив его и благотворным для становления веб-дизайна. Разработчики сайтов поняли, что дизайн это не просто разноцветный текст и вставленные наобум картинки. Это возможность красочно и эффектно подать материал, придать определённое настроение сайту, заинтересовать посетителя и задержать его. Дизайн стал непосредственным этапом разработки сайта, за которым последовала и вёрстка. Нарисованные в Фотошопе макеты следовало превратить из картинки в набор изображений, стилевых и HTML-файлов, способных быстро загружаться по сети, сохраняя при этом особенности дизайна.

Век Netscape продолжался до тех пор, пока часть рынка браузеров не захватила Microsoft со своим браузером Internet Explorer, что в итоге принесло только головную боль разработчикам сайтов. Разные подходы Netscape и Internet Explorer к отображению сайта, противоречивая поддержка стилей и тегов, большое количество мелких ошибок привело к тому, что сайт приходилось тестировать и отлаживать долгое время.

Устаревшие версии браузеров не в полной мере поддерживали CSS (Cascading Style Sheets, каскадные таблицы стилей), поэтому код представлял собой окрошку из HTML и CSS. Это привело к тому, что для макетов стали применяться таблицы с невидимой границей, на долгое время ставшими стандартом де-факто.

Следующим этапом развития разработки сайтов стало рождение спецификации, которая была названа Cascading Style Sheets Level 2, сокращенно CSS2. Следом появилось обновление Cascading Style Sheets Level 2 Revision 1 (CSS 2.1), которое используется по сей день. Выпуск новых версий браузеров, поддерживающих, пусть и не в полной мере, эту спецификацию, серьёзно облегчил работу над сайтами. В итоге, начал происходить постепенный переход от табличной вёрстки к блочной или, как она ещё называется, вёрстке слоями, в которой расположение элементов на странице и их вид задаётся через стили.

В настоящее время уже можно заявить, что табличная вёрстка в большинстве своём является архаизмом. Тем не менее, существует ещё множество сайтов вроде yandex.ru, lenta.ru и других, свёрстанных именно при содействии таблиц. Такой консерватизм связан со следующими обстоятельствами.

- Вёрстка слоями сложнее, чем таблицами, поскольку требует от разработчика глубоких знаний спецификации CSS.
- Браузеры содержат разные ошибки при интерпретации стилевых свойств, поэтому необходимо знать особенности поведения основных браузеров, их ошибки и уметь обходить их.
- Практически каждую задачу можно решить несколькими способами, от разработчика требуется выбрать оптимальный, т.е. наименее затратный по времени и усилиям. Для оценки

оптимальности требуется практический опыт, чтобы иметь в запасе разные методы, применяемые в конкретных ситуациях.

Таким образом, вёрстка в простом понимании это процесс превращения работы дизайнера в веб-страницу, отображаемую в браузере. Но один и тот же результат можно получить разными методами и уже от верстальщика зависит, какой из них предпочесть. От такого выбора зависит результат работы сайта, быстрота его отображения, доступность для разных устройств и браузеров.

Запугал вас всеми этими сложностями и ответственностью? В конце концов, у вас же есть книга по вёрстке, которая, может, не содержит ответы на все вопросы, но здорово поможет разобраться с ними. Только помните, что вёрстка это практическая дисциплина и самостоятельно решая разные задачи и работая над проектами, вы сможете освоить это искусство гораздо быстрее.

Для кого эта книга

Для новичков мы расскажем, как использовать на своей странице стили, как их применять для оформления элементов — ссылок, рисунков, таблиц, форм и списков. Подробные примеры, описание всех действий и комментарии позволяют легко воспроизвести приведенные технологии на практике и модернизировать код под свои нужды.

Опытные пользователи получат более подробные сведения об особенностях популярных браузеров и о том, как они работают с тегами и стилями. Это позволит учитывать различия между браузерами и создавать универсальные документы, которые будут корректно в них отображаться.

Наконец, искушенные специалисты по созданию сайтов смогут использовать книгу как справочное руководство, заглядывая в нее для получения информации об ошибках браузеров и нестандартном применении некоторых стилевых свойств.

Как организована книга

Материал в книге представлен так, что ее можно читать последовательно с самого начала или пользоваться ею в качестве справочника, заглядывая в ту главу, которая в данный момент больше всего интересует. Все главы следует рассматривать как независимые — это позволяет вам сразу перейти к теме, которую вы считаете для себя более важной. Однако между ними существует и взаимосвязь, поэтому, чтобы не повторять дважды одни и те же приёмы, на них дается ссылка. Кроме того, следует понимать, что множество техник очень тесно переплетены между собой, и их невозможно без повторений отнести к одной теме. Так что разные приемы в некотором смысле разбросаны по главам, но стройно и логично следуют один из другого. В первый раз рекомендуется прочитать книгу с самого начала, а затем обращаться к нужному разделу по необходимости.

Книга состоит из десяти глав.

Глава I. Введение в CSS. Описаны основные принципы спецификации CSS и применение их на практике для управления видом элементов.

Глава II. Режимы браузеров. Перечислены режимы работы браузеров, способы переключения между ними и основные их особенности. Указана взаимосвязь режимов с элементом <!DOCTYPE>, даны рекомендации по его использованию.

Глава III. Принципы вёрстки слоями. Перечислены фундаментальные принципы вёрстки, описано поведение разных элементов: блочных, строчных, плавающих. Рассказано про позиционирование элементов на странице.

Глава IV. Вёрстка с помощью таблиц. Описаны особенности таблиц, изменение их вида с помощью стилей, приведены двух и трёхколоночные макеты на основе табличной вёрстки.

Глава V. Браузер Internet Explorer. В главе объясняется модель отображения разных версий браузера, перечислены их ошибки и способы обхода.

Глава VI. Макеты, применяемые на сайтах. Приводятся различные виды макетов: фиксированные, резиновые, комбинированные и способы создания одно, двух и трёхколоночных макетов разного вида.

Глава VII. Типовые элементы веб-страницы. Представлены способы вставки различных элементов на страницу вроде меню, форм, изображений, модальных окон и др.

Глава VIII. Вёрстка сайта на практике. На примере конкретного сайта пошагово рассматриваются этапы его вёрстки.

Глава IX. Использование HTML5. В этой главе указаны новые теги HTML5, показано их применение на практике.

Глава X. Тестирование и отладка готового кода. Разные инструменты для облегчения работы по вёрстке, отыскания ошибок отображения веб-страниц и их исправления.

Поддержка

Ваши замечания, предложения и вопросы направляйте автору на адрес электронной почты vlad@htmlbook.ru.

Обозначения

Для удобства и наглядного представления материала используется разделение по цветам различных элементов.

`<strong>` — тег, стилевой селектор.

align — атрибут тега, стилевое свойство, ключевое слово или выделение.

right — значение.

`layer` — имя класса или идентификатора.

File > Open — пункт меню указанной программы.

Tab — клавиша на клавиатуре.

 Этот символ призван привлечь внимание к некоторому замечанию по тексту.

Браузеры

В таблице с браузерами встречаются такие изображения:

 — свойство полностью поддерживается браузером со всеми допустимыми значениями;

 — свойство браузером не воспринимается и игнорируется;

 — свойство поддерживается лишь частично, например, не все допустимые значения действуют или свойство применяется не ко всем элементам, которые указаны в спецификации;

 — свойство понимается браузером, но при его работе возможно появление различных ошибок. В примечании обычно указывается, какого рода ошибки обнаруживаются в браузере.

Примеры

Как правило, примеры содержат список браузеров, в которых проверялся код. Для сокращения места браузеры обозначаются двумя буквами с номером версии.

- IE — Internet Explorer.
- Cr — Google Chrome.
- Op — Opera.
- Sa — Apple Safari.
- Fx — Mozilla Firefox.

Работоспособность примера в браузере помечается цветом, совпадающим с цветами в таблице с браузерами.

Глава I

Введение в CSS

Инструменты

Прежде чем приступать к работе, необходимо позаботиться о своем рабочем месте и обеспечить его требуемыми инструментами. К ним относятся редактор для правки стилей и браузер.

Редактор

CSS-файл представляет собой обычный текстовый документ, поэтому для его правки подойдет любой редактор, даже Блокнот, входящий в стандартный комплект Windows. Но все же лучше выбрать редактор, позволяющий одновременно работать с несколькими документами, имеющий подсветку синтаксиса и множество других полезных возможностей. Также желательно, чтобы программа была бесплатной.

Среди программ, удовлетворяющих приведенным условиям, есть несколько. Далее перечислим их с кратким описанием.

PSPad

Небольшой и быстрый текстовый редактор, позволяет добавлять в текст стилевые свойства простым нажатием средней кнопки мыши.

Где скачать
http://www.pspad.com/ru/download.php

Notepad++

Позиционируется разработчиками как замена стандартного Блокнота. И надо отметить, этот редактор гораздо лучше морально устаревшей программы.

Где скачать
http://notepad-plus.sourceforge.net/ru/download.php?lang=ru

Браузер

Браузер это программа, предназначенная для просмотра веб-страниц. Из-за того, что основные браузеры используют для отображения документа свой собственный «движок», одна и та же страница может довольно сильно различаться при просмотре в разных браузерах. Разумеется, сайт надо делать так, чтобы максимально полно охватить аудиторию, поэтому приходится завести целый «зверинец», чтобы тестировать результат. На сегодняшний день наибольшей популярностью пользуются пять браузеров: Firefox, Internet Explorer, Opera, Safari и Chrome.

Mozilla Firefox

Перспективный и развивающийся браузер, получивший признание во всем мире. Его особенность — простота и масштабируемость, которая удаётся за счет специальных расширений, как они называются. Изначально Firefox имеет набор только самых необходимых функций, но, устанавливая желаемые дополнения, в итоге можно нарастить браузер до системы, выполняющей все необходимые для вашей работы действия.

Браузер Firefox является открытой системой, разрабатываемый группой Mozilla.

Синонимы
Firefox, FF (сокр.), Fx (сокр.), Огнелиса (жарг.), лиса (жарг.), лисичка (жарг.)

Где скачать
http://www.mozilla.ru/products/firefox/

Microsoft Internet Explorer

Один из старейших браузеров, который бесплатно поставляется вместе с операционной системой Windows. Это и определило его популярность. К сожалению, различные версии этого браузера работают довольно противоречиво, поэтому для тестирования сайта приходится одновременно использовать седьмую и восьмую версию Internet Explorer. Вскоре ожидается выход девятой версии программы, бета-версия которой уже доступна для скачивания.

Синонимы

MSIE, IE (сокр.), ИА (жарг.), осел, ослик (жарг.)

Где скачать

http://www.microsoft.com/rus/windows/internet-explorer/

Opera

Быстрый и удобный браузер, поддерживающий множество дополнительных возможностей, повышающих комфортность работы с сайтами.

Синонимы

Опера, Op (сокр.)

Где скачать

http://opera.ru

Safari

Разработаный компанией Apple этот браузер встроен в в iPhone и операционную систему MacOS на компьютерах Apple. Также имеется версия под Windows.

Где скачать

http://www.apple.com/ru/safari/

Chrome

По сравнению с остальными является самым свежим браузером, появившимся на рынке в конце 2008 году. Разработан компанией Google.

Синонимы

Хром, Cr (сокр.)

Где скачать

http://www.google.com/chrome?hl=ru

Введение в CSS

HTML задает основную структуру веб-страницы, а также указывает, какие элементы на ней присутствуют. Само оформление веб-страницы, положение и вид элементов возложен на стили или CSS (Cascading Style Sheets, каскадные таблицы стилей). Когда говорят о вёрстке веб-страниц, подразумевается синергия HTML и CSS. Что такое синергия? Сам HTML не представляет отдельного интереса, в силу своей простоты и ограниченности. Также и CSS не играет отдельной роли, поскольку привязывается к определенным элементам кода и задает их оформление. Поэтому работая вместе в одной связке, они превращают скромную страницу в тот документ, который придумал и нарисовал дизайнер. Такое взаимное усиление свойств, суммирующий эффект и является синергией.

Любая веб-страница это, по сути, комбинация HTML-кода и CSS-кода. Без основных знаний этих технологий не получится грамотно сверстать ни один документ. Поэтому первая глава посвящена основам CSS и его применению на практике. Если вы считаете, что это уже вам известно, можете пропустить эту главу и перейти к следующей.

Что такое стили?

Стили представляют собой набор параметров, управляющих видом и положением элементов веб-страницы. Чтобы стало понятно, о чем идет речь, посмотрим на рис. 1.1.

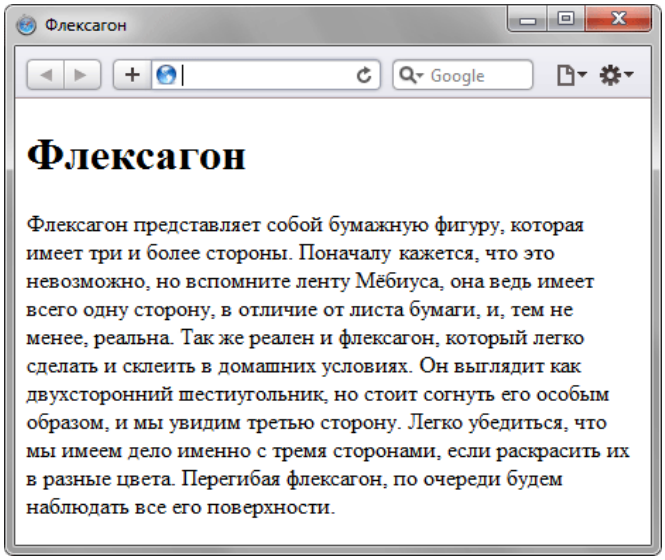


Рис. 1.1. Веб-страница, созданная только на HTML

Это обычная веб-страница, оформленная без всяких изысков. Тот же самый документ, но уже с добавлением стилей приобретает совершенно иной вид (рис. 1.2).

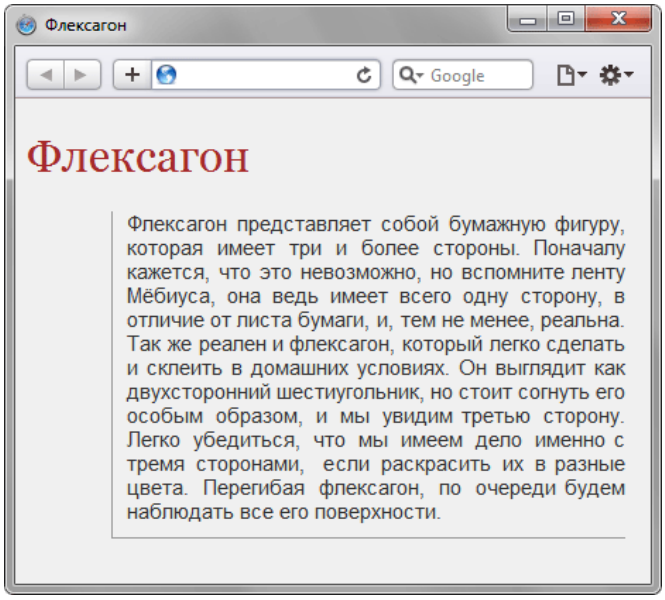


Рис. 1.2. Веб-страница, созданная на HTML и CSS

Перемена разительна, поэтому заглянем в код, чтобы понять, в чем же разница (пример 1.1).

Пример 1.1. Исходный код документа

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Флексагон</title>
  <link rel="stylesheet" href="style.css" type="text/css" />
 </head>
 <body>
  <h1>Флексагон</h1>
  <p>Флексагон представляет собой бумажную фигуру, которая имеет три и
   более стороны. Поначалу кажется, что это невозможно, но вспомните ленту
   Мёбиуса, она ведь имеет всего одну сторону, в отличие от листа бумаги,
   и, тем не менее, реальна. Так же реален и флексагон, который легко
   сделать и склеить в домашних условиях. Он выглядит как двухсторонний
   шестиугольник, но стоит согнуть его особым образом, и мы увидим третью
   сторону. Легко убедиться, что мы имеем дело именно с тремя сторонами,
   если раскрасить их в разные цвета. Перегибая флексагон, по очереди будем
   наблюдать все его поверхности.</p>
 </body>
</html>
```

Сам код HTML никаких изменений не претерпел и единственное добавление — это строка `<link rel="stylesheet" href="style.css" type="text/css" />`. Она ссылается на внешний файл с описанием стилей под именем style.css. Содержимое этого файла показано в примере 1.2.

Пример 1.2. Содержимое стилевого файла style.css

```
body {
  font-family: Arial, Verdana,  sans-serif; /* Семейство шрифтов */
  font-size: 11pt; /* Размер основного шрифта в пунктах  */
  background-color: #f0f0f0; /* Цвет фона веб-страницы */
  color: #333; /* Цвет основного текста */
}
h1 {
  color: #a52a2a; /* Цвет заголовка */
  font-size: 24pt; /* Размер шрифта в пунктах */
  font-family: Georgia, Times, serif; /* Семейство шрифтов */
  font-weight: normal; /* Нормальное начертание текста  */
}
p {
  text-align: justify; /* Выравнивание по ширине */
  margin-left: 60px; /* Отступ слева в пикселах */
  margin-right: 10px; /* Отступ справа в пикселах */
  border-left: 1px solid #999; /* Параметры линии слева */
  border-bottom: 1px solid #999; /* Параметры линии снизу */
  padding-left: 10px; /* Отступ от линии слева до текста  */
  padding-bottom: 10px; /* Отступ от линии снизу до текста  */
}
```

В файле style.css как раз и описаны все параметры оформления таких тегов как `<body>`, `<h1>` и `<p>`. Заметьте, что сами теги в коде HTML пишутся как обычно.

Поскольку на файл со стилем можно ссылаться из любого веб-документа, это приводит в итоге к сокращению объема повторяющихся данных. А благодаря разделению кода и оформления повышается гибкость управления видом документа и скорость работы над сайтом.

Типы стилей

Различают несколько типов стилей, которые могут совместно применяться к одному документу. Это стиль браузера, стиль автора и стиль пользователя.

Стиль браузера

Оформление, которое по умолчанию применяется к элементам веб-страницы браузером. Это оформление можно увидеть в случае «голого» HTML, когда к документу не добавляется никаких стилей. Например, заголовок страницы, формируемый тегом `<H1>`, в большинстве браузеров выводится шрифтом с засечками размером 24 пункта.

Стиль автора

Стиль, который добавляет к документу его разработчик. В примере 1.1 показан один из возможных способов подключения авторского стиля.

Стиль пользователя

Это стиль, который может включить пользователь сайта через настройки браузера. Такой стиль имеет более высокий приоритет и переопределяет исходное оформление документа. В браузере Internet Explorer подключение стиля пользователя делается через меню Сервис > Свойство обозревателя > Кнопка «Оформление», как показано на рис. 1.3.

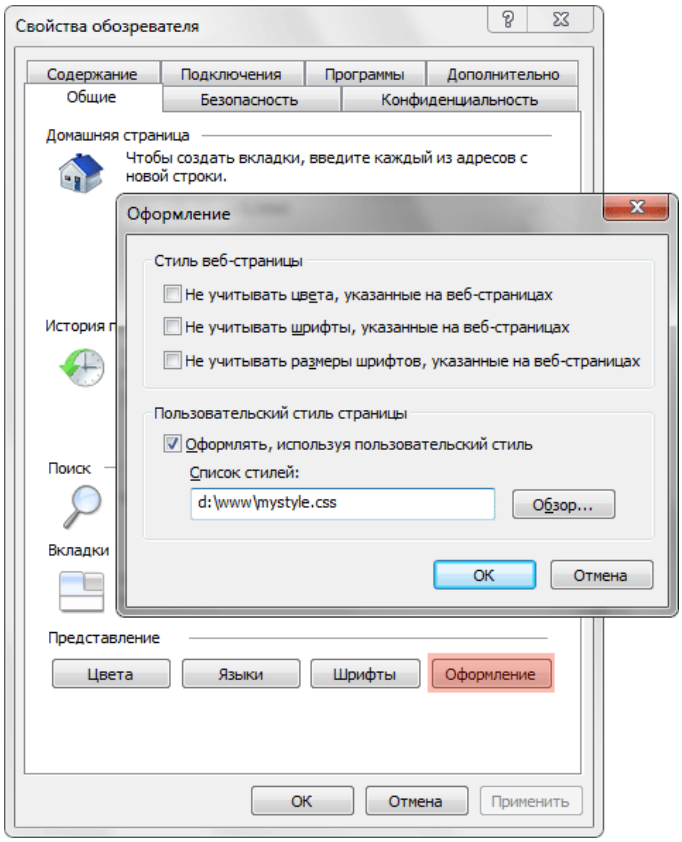


Рис. 1.3. Подключение стиля пользователя в браузере Internet Explorer

В браузере Opera аналогичное действие происходит через команду Инструменты > Общие настройки > Вкладка «Расширенные» > Содержимое > Кнопка «Параметры стиля» (рис. 1.4).

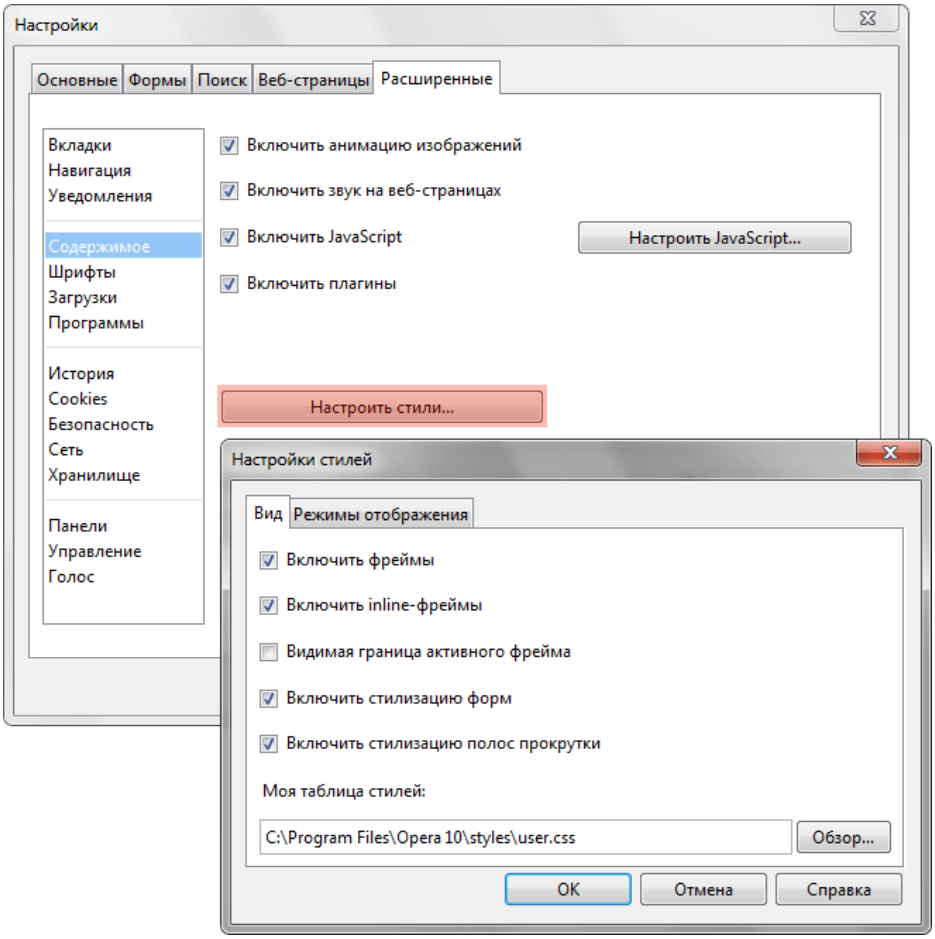


Рис. 1.4. Подключение стиля пользователя в браузере Opera

Указанные типы стилей могут спокойно существовать друг с другом, если они не пытаются изменить вид одного элемента. В случае возникновения противоречия вначале имеет приоритет стиль пользователя, затем стиль автора и последним идет стиль браузера.

Способы добавления стилей на страницу

Для добавления стилей на веб-страницу существует несколько способов, которые различаются своими возможностями и назначением. Далее рассмотрим их подробнее.

Связанные стили

При использовании связанных стилей описание селекторов и их значений располагается в отдельном файле, как правило, с расширением css, а для связывания документа с этим файлом применяется тег **<link>**. Данный тег помещается в контейнер **<head>**, как показано в примере 1.3.

Пример 1.3. Подключение связанных стилей

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Стили</title>
  <link rel="stylesheet" type="text/css" href="style/mysite.css" />
  <link rel="stylesheet" type="text/css"
    href="http://www.htmlbook.ru/main.css" />
 </head>
 <body>
   <h1>Заголовок</h1>
   <p>Текст</p>
 </body>
</html>
```

Значения атрибутов тега **<link>** — rel и type остаются неизменными независимо от кода, как приведено в данном примере. Значение href задает путь к CSS-файлу, он может быть задан как относительно, так и абсолютно. Заметьте, что таким образом можно подключать таблицу стилей, которая находится на другом сайте.

Содержимое файла mysite.css подключаемого посредством тега **<link>** приведено в примере 1.4.

Пример 1.4. Файл со стилем

```
H1 {
  color: #000080;
  font-size: 2em;
  font-family: Arial, Verdana, sans-serif;
  text-align: center; /* Выравнивание по центру */
}
P {
  padding-left: 20px;
}
```

Как видно из данного примера, файл со стилем не хранит никаких данных, кроме синтаксиса CSS. В свою очередь и HTML-документ содержит только ссылку на файл со стилем, т.е. таким способом в полной мере реализуется принцип разделения кода и оформления сайта. Поэтому использование связанных стилей является наиболее универсальным и удобным методом добавления стиля на сайт. Ведь стили хранятся в одном файле, а в HTML-документах указывается только ссылка на него.

Глобальные стили

При использовании глобальных стилей свойства CSS описываются в самом документе и располагаются в заголовке веб-страницы. По своей гибкости и возможностям этот способ добавления стиля уступает предыдущему, но также позволяет хранить стили в одном месте, в данном случае прямо на странице с помощью контейнера **<style>**, как показано в примере 1.5.

Пример 1.5. Использование глобального стиля XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Глобальные стили</title>
  <style type="text/css">
   H1 {
    font-size: 1.2em;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    color: #333366;
   }
  </style>
 </head>
 <body>
  <h1>Hello, world!</h1>
 </body>
</html>
```

В данном примере определен стиль тега **<h1>**, который затем можно повсеместно использовать на данной веб-странице (рис. 1.5).

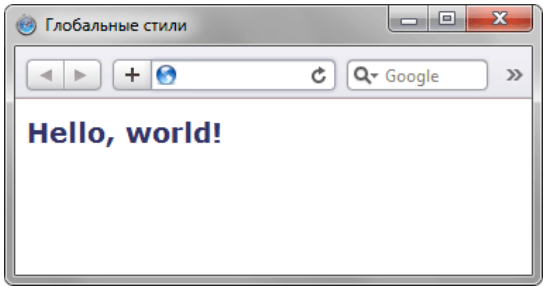


Рис. 1.5. Вид заголовка, оформленного с помощью стилей

Внутренние стили

Внутренний или встроенный стиль является по существу расширением для одиночного тега используемого на текущей веб-странице. Для определения стиля используется атрибут style, а его значением выступает набор стилевых правил (пример 1.6).

Пример 1.6. Использование внутреннего стиля XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Внутренние стили</title>
 </head>
 <body>
  <p style="font-size: 120%; font-family: monospace;
   color: #cd66cc">Пример текста</p>
</body>
</html>
```

В данном примере стиль тега **<p>** задается с помощью атрибута style, в котором через точку с запятой перечисляются стилевые свойства (рис. 1.6).

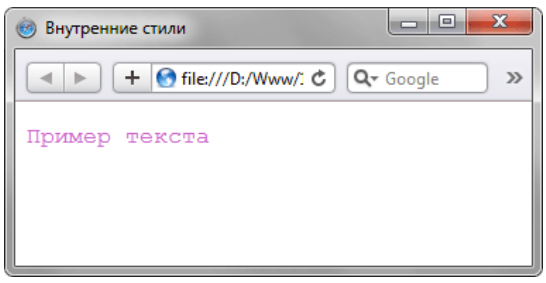


Рис. 1.6. Использование внутренних стилей для изменения вида текста

Внутренние стили рекомендуется применять на сайте ограниченно или вообще отказаться от их использования. Дело в том, что добавление таких стилей увеличивает общий объем файлов, что ведет к повышению времени их загрузки в браузере, и усложняет редактирование документов для разработчиков.

Все описанные методы использования CSS могут применяться как самостоятельно, так и в сочетании друг с другом. В этом случае необходимо помнить об их иерархии. Первым всегда применяется внутренний стиль, затем глобальный стиль и в последнюю очередь связанный стиль. В примере 1.7 применяется сразу два метода добавления стиля в документ.

Пример 1.7. Разные методы подключения стилей

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Подключение стиля</title>
  <style type="text/css">
   H1 {
    font-size: 1.2em;
    font-family: Arial, Helvetica, sans-serif;
    color: green;
   }
  </style>
 </head>
 <body>
   <h1 style="font-size: 36px;
    font-family: Times, serif; color: red">Заголовок 1</h1>
   <h1>Заголовок 2</h1>
 </body>
</html>
```

В данном примере первый заголовок задается красным цветом размером 36 пикселов с помощью внутреннего стиля, а следующий — зеленым цветом через таблицу глобальных стилей (рис. 1.7).

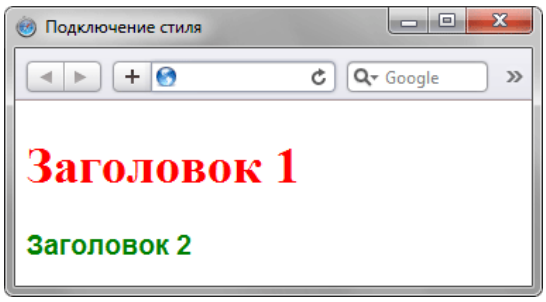


Рис. 1.7. Результат применения стилей

Импорт CSS

В текущую стилевую таблицу можно импортировать содержимое CSS-файла с помощью команды @import. Этот метод допускается использовать совместно со связанными или глобальными стилями, но никак не со встроенными стилями. Общий синтаксис следующий.

```
@import url("имя файла") типы носителей;
@import "имя файла" типы носителей;
```

После ключевого слова @import указывается путь к стилевому файлу одним из двух приведенных способов — с помощью url или без него. В примере 1.8 показано, как можно импортировать стиль из внешнего файла в таблицу глобальных стилей.

Пример 1.8. Импорт CSS в глобальную таблицу стилей

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Импорт</title>
  <style type="text/css">
   @import url("style/mysite.css");
   H2 {
    font-size: 1.2em;
    font-family: Arial, Helvetica, sans-serif;
    color: green;
   }
  </style>
 </head>
 <body>
   <h1>Заголовок 1</h1>
   <h2>Заголовок 2</h2>
 </body>
</html>
```

В данном примере показано подключение файла mysite.css, который расположен в папке style.

Аналогично происходит импорт и в файле со стилем, который затем подключается к документу (пример 1.9).

Пример 1.9. Импорт в таблице связанных стилей

```
@import "style/print.css";
@import "style/main.css";
BODY {
 font-family: Arial, Verdana, Helvetica, sans-serif;
 font-size: 90%;
 background: white;
 color: black;
}
```

В данном примере показано с держимое файла mysite.css, который добавляется к нужным документам способом, показанным в примере 1.3, а именно с помощью тега **<link>**.

Типы носителей

Широкое развитие различных платформ и устройств заставляет разработчиков делать под них специальные версии сайтов, что достаточно трудоемко и проблематично. Вместе с тем, времена и потребности меняются, и создание сайта для различных устройств является неизбежным и необходимым звеном его развития. С учетом этого в CSS введено понятие типа носителя, когда стиль применяется только для определенного устройства. В табл. 1.1 перечислены некоторые типы носителей.

Табл. 1.1. Типы носителей и их описание

Тип	Описание
all	Все типы. Это значение используется по умолчанию.
aural	Речевые синтезаторы, а также программы для воспроизведения текста вслух. Сюда, например, можно отнести речевые браузеры.
braille	Устройства, основанные на системе Брайля, которые предназначены для слепых людей.
handheld	Наладонные компьютеры и аналогичные им аппараты.
print	Печатающие устройства вроде принтера.
projection	Проектор.
screen	Экран монитора.
tv	Телевизор.

В CSS для указания типа носителей применяются команды @import и @media, с помощью которых можно определить стиль для элементов в зависимости от того, выводится документ на экран или на принтер.

Ключевые слова @media и @import относятся к эт-правилам. Такое название произошло от наименования символа @ — «эт», с которого и начинаются эти ключевые слова. В рунете для обозначения символа @ применяется устоявшийся термин «собака». Только вот использовать выражение «собачье правило» язык не поворачивается.

При импортировании стиля через команду @import тип носителя указывается после адреса файла. При этом допускается задавать сразу несколько типов, упоминая их через запятую, как показано в примере 1.10.

Пример 1.10. Импорт стилевого файла

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Импорт стиля</title>
  <style type="text/css">
   @import "style/main.css" screen; /* Для вывода результата на монитор */
   @import "style/print.css" print, handheld; /* Для печати и смартфона */
  </style>
 </head>
 <body>
  <p>...</p>
 </body>
</html>
```

В данном примере импортируется два файла — main.css предназначен для изменения вида документа при его просмотре на экране монитора, и print.css — при печати страницы и отображении на

смартфоне.

Браузер Internet Explorer до седьмой версии включительно не поддерживает типы носителей при импорте стилевого файла. Более того, при добавлении типа носителя стилевой файл вообще не загружается.

Команда @media позволяет указать тип носителя для глобальных или связанных стилей и в общем случае имеет следующий синтаксис.

```
@media тип носителя 1 {
  Описание стиля для типа носителя 1
}
@media тип носителя 2 {
  Описание стиля для типа носителя 2
}
```

После ключевого слова @media идет один или несколько типов носителя, перечисленных в табл. 1.1, если их больше одного, то они разделяются между собой запятой. После чего следуют обязательные фигурные скобки, внутри которых идет обычное описание стилевых правил. В примере 1.11 показано, как задать разный стиль для печати и отображения на мониторе.

Пример 1.11. Стили для разных типов носителей

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Типы носителей</title>
  <style type="text/css">
   @media screen { /* Стиль для отображения в браузере */
    BODY {
     font-family: Arial, Verdana, sans-serif; /* Рубленый шрифт */
     font-size: 0.9em; /* Размер шрифта */
     color: #000080; /* Цвет текста */
    }
    H1 {
     background: #faf0e6; /* Цвет текста */
     border: 2px dashed maroon; /* Рамка вокруг заголовка */
     color: #a0522d; /* Цвет текста */
     padding: 7px; /* Поля вокруг текста */
    }
    H2 {
     color: #556b2f; /* Цвет текста */
     margin: 0; /* Убираем отступы */
    }
    P {
     margin-top: 0.5em; /* Отступ сверху */
    }
   }
   @media print { /* Стиль для печати */
    BODY {
     font-family: Times, 'Times New Roman', serif; /* Шрифт с засечками */
    }
    H1, H2, P {
     color: black; /* Черный цвет текста */
    }
   }
  </style>
 </head>
 <body>
  <h1>Как поймать льва в пустыне</h1>
  <h2>Метод случайных чисел</h2>
  <p>Разделим пустыню на ряд элементарных прямоугольников, размер
     которых совпадает с размером клетки для льва. После чего
     перебираем полученные прямоугольники, каждый раз выбирая заданную
     область случайным образом. Если в данной области окажется лев,
     то мы поймаем его, накрыв клеткой.</p>
 </body>
</html>
```

В данном примере вводится два стиля — один для изменения вида элементов при их обычном отображении в браузере, а второй — при выводе страницы на печать. При этом облик документа для разных носителей может сильно различаться между собой, например, как это показано на рис. 1.8 и рис. 1.9.

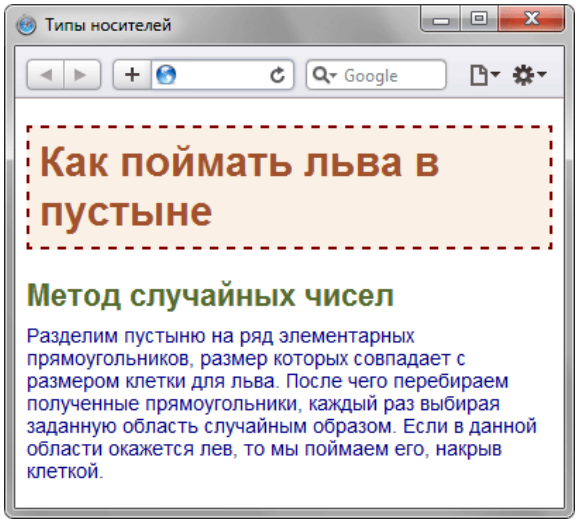


Рис. 1.8. Страница для отображения в окне браузера

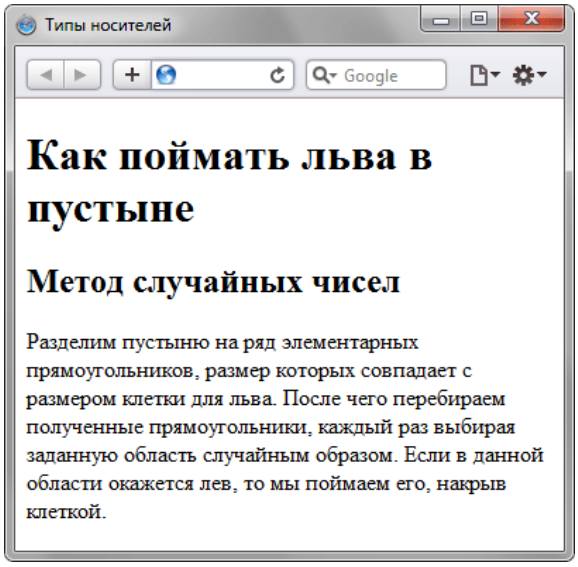


Рис. 1.9. Страница, предназначенная для печати

Просмотреть документ, у которого CSS установлен как тип print можно, если распечатать определенную страницу. Или пойти на хитрость и временно заменить print на screen, чтобы отобразить итог в браузере. Именно так был получен рис. 1.9.

Команда @media применяется в основном для формирования одного стилевого файла, который разбит на блоки по типу устройств. Иногда же имеет смысл создать несколько разных CSS-файлов — один для печати, другой для отображения в браузере — и подключать их к документу по мере необходимости. В подобном случае следует воспользоваться тегом **<link>** с атрибутом media, значением которого выступают все те же типы, перечисленные в табл. 1.1.

В примере 1.12 показано, как создавать ссылки на CSS-файлы, которые предназначены для разных типов носителей.

Пример 1.12. Подключение стилей для разных носителей

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Разные носители</title>
  <link media="print, handheld" rel="stylesheet"
    href="style/print.css" type="text/css" />
  <link media="screen" rel="stylesheet"
    href="style/main.css" type="text/css" />
```

```
 </head>
 <body>
  <p>...</p>
 </body>
</html>
```

В данном примере используются две таблицы связанных стилей, одна для отображения в браузере, а вторая — для печати документа и его просмотре на смартфоне. Хотя на страницу загружаются одновременно два разных стиля, применяются они только для определенных устройств.

Аналогично можно использовать и глобальные стили, добавляя параметр media к тегу **<style>** (пример 1.13).

Пример 1.13. Стиль для смартфона

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Разные носители</title>
  <style type="text/css" media="handheld">
   BODY {
    width: 320px; /* Ширина страницы */
   }
  </style>
 </head>
 <body>
  <p>Разделим пустыню на ряд элементарных прямоугольников, размер
   которых совпадает с размером клетки для льва. После чего
   перебираем полученные прямоугольники, каждый раз выбирая заданную
   область случайным образом. Если в данной области окажется лев,
   то мы поймаем его, накрыв клеткой.</p>
</body>
</html>
```

В данном примере ширина для устройств типа handheld ограничена размером 320 пикселов.

Базовый синтаксис CSS

Как уже было отмечено ранее, стилевые правила записываются в своем формате, отличном от HTML. Основным понятием выступает селектор — это некоторое имя стиля, для которого добавляются параметры форматирования. В качестве селектора выступают теги, классы и идентификаторы. Общий способ записи имеет следующий вид.

Вначале пишется имя селектора, например, **TABLE**, это означает, что все стилевые правила будут применяться к тегу **<table>**, затем идут фигурные скобки, в которых записывается стилевое свойство, а его значение указывается после двоеточия. Стилевые свойства разделяются между собой точкой с запятой, в конце этот символ можно опустить.

CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции, поэтому форма записи зависит от желания разработчика. Так, в примере 1.14 показаны две разновидности оформления селекторов и их правил.

Пример 1.14. Использование стилей

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Заголовки</title>
  <style type="text/css">
   H1 { color: #a6780a; font-weight: normal; }
   H2 {
    color: olive;
    border-bottom: 2px solid black;
   }
  </style>
 </head>
 <body>
  <h1>Заголовок 1</h1>
  <h2>Заголовок 2</h2>
 </body>
</html>
```

В данном примере свойства селектора **H1** записаны в одну строку, а для селектора **H2** каждое свойство находится на отдельной строке. Во втором случае легче отыскивать нужные свойства и править их по необходимости, но при этом незначительно возрастает объем данных за счет активного использования пробелов и переносов строк. Так что в любом случае способ оформления стилевых правил зависит от разработчика.

Правила применения стилей

Далее приведены некоторые правила, которые необходимо знать при описании стиля.

Форма записи

Для селектора допускается добавлять каждое стилевое свойство и его значение по отдельности, как это показано в примере 1.15.

Пример 1.15. Расширенная форма записи

```
td { background: olive; }
td { color: white; }
td { border: 1px solid black; }
```

Однако такая запись не очень удобна. Приходится повторять несколько раз один и тот же селектор, да и легко запутаться в их количестве. Поэтому пишите все свойства для каждого селектора вместе. Указанный набор записей в таком случае получит следующий вид (пример 1.16).

Пример 1.16. Компактная форма записи

```
td {
  background: olive;
  color: white;
  border: 1px solid black;
}
```

Эта форма записи более наглядная и удобная в использовании.

Имеет приоритет значение, указанное в коде ниже

Если для селектора вначале задается свойство с одним значением, а затем то же свойство, но уже с другим значением, то применяться будет то значение, которое в коде установлено ниже (пример 1.17).

Пример 1.17. Разные значения у одного свойства

```
P { color: green; }
P { color: red; }
```

В данном примере для селектора **P** цвет текста вначале задается зеленым, а затем красным. Поскольку значение red расположено ниже, то оно в итоге и будет применяться к тексту.

На самом деле такой записи лучше вообще избегать и удалять повторяющиеся значения. Но подобное может произойти не явно, например, в случае подключения разных стилевых файлов, в которых содержатся одинаковые селекторы.

Значения

У каждого свойства может быть только соответствующее его функции значение. Например, для color, который устанавливает цвет текста, в качестве значений недопустимо использовать числа.

Комментарии

Комментарии нужны, чтобы делать пояснения по поводу использования того или иного стилевого свойства, выделять разделы или писать свои заметки. Комментарии позволяют легко вспоминать логику и структуру селекторов, и повышают разборчивость кода. Вместе с тем, добавление текста увеличивает объем документов, что отрицательно сказывается на времени их загрузки. Поэтому комментарии обычно применяют в отладочных или учебных целях, а при выкладывании сайта в сеть их стирают.

Чтобы пометить, что текст является комментарием, применяют следующую конструкцию /* ... */ (пример 1.18).

Пример 1.18. Комментарии в CSS-файле

```
/*
  Стиль для сайта htmlbook.ru
  Сделан для ознакомительных целей
*/

div {
  width: 200px; /* Ширина контента */
  margin: 10px; /* Поля вокруг элемента */
  float: left; /* Обтекание по правому краю */
}
```

Как следует из данного примера, комментарии можно добавлять в любое место CSS-документа, а также писать текст комментария в несколько строк. Вложенные комментарии недопустимы.

Значения стилевых свойств

Все многообразие значений стилевых свойств может быть сведено к определенному типу: строка, число, проценты, размер, цвет, адрес или ключевое слово.

Строки

Любые строки необходимо брать в двойные или одинарные кавычки. Если внутри строки требуется оставить одну или несколько кавычек, то можно комбинировать типы кавычек или добавить перед кавычкой слэш (пример 1.19).

Пример 1.19. Допустимые строки

```
'Гостиница "Турист"'
"Гостиница 'Турист'"
"Гостиница \"Турист\""
```

В данном примере в первой строке применяются одинарные кавычки, а слово «Турист» взято в двойные кавычки. Во второй строке все с точностью до наоборот, в третьей же строке используются только двойные кавычки, но внутренние экранированы с помощью слэша.

Числа

Значением может выступать целое число, содержащее цифры от 0 до 9 и десятичная дробь, в которой целая и десятичная часть разделяются точкой (пример 1.20).

Пример 1.20. Числа в качестве значений

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Числа</title>
  <style type="text/css">
   P {
    font-weight: 600; /* Жирное начертание */
    line-height: 1.2; /* Межстрочный интервал */
   }
  </style>
 </head>
 <body>
  <p>Пример текста</p>
 </body>
</html>
```

Если в десятичной дроби целая часть равна нулю, то её разрешается не писать. Запись .7 и 0.7 равнозначна.

Проценты

Процентная запись обычно применяется в тех случаях, когда надо изменить значение относительно родительского элемента или когда размеры зависят от внешних условий. Так, ширина таблицы 100% означает, что она будет подстраиваться под размеры окна браузера и меняться вместе с шириной окна (пример 1.21).

Пример 1.21. Процентная запись

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
  <title>Ширина в процентах</title>
  <style type="text/css">
   TABLE {
    width: 100%; /* Ширина таблицы в процентах */
    background: #f0f0f0; /* Цвет фона */
   }
  </style>
 </head>
 <body>
   <table>
    <tr><td>Содержимое таблицы</td></tr>
   </table>
 </body>
</html>
```

Проценты не обязательно должны быть целым числом, допускается использовать десятичные дроби, вроде значения 56.8%, но не всегда.

Размеры

Для задания размеров различных элементов, в CSS используются абсолютные и относительные единицы измерения. Абсолютные единицы не зависят от устройства вывода, а относительные единицы определяют размер элемента относительно значения другого размера.

Относительные единицы

Относительные единицы обычно используют для работы с текстом, либо когда надо вычислить процентное соотношение между элементами. В табл. 1.2 перечислены основные относительные единицы.

Табл. 1.2. Относительные единицы измерения

Единица	Описание
em	Размер шрифта текущего элемента
ex	Высота символа x
px	Пиксел
%	Процент

Единица em это изменяемое значение, которое зависит от размера шрифта текущего элемента (размер устанавливается через стилевое свойство font-size). В каждом браузере заложен размер текста, применяемый в том случае, когда этот размер явно не задан. Поэтому изначально 1em равен размеру шрифта, заданного в браузере по умолчанию или размеру шрифта родительского элемента. Процентная запись идентична em, в том смысле, что значения 1em и 100% равны.

Единица ex определяется как высота символа «x» в нижнем регистре. На ex распространяются те же правила, что и для em, а именно, он привязан к размеру шрифта, заданного в браузере по умолчанию, или к размеру шрифта родительского элемента.

Пиксел это элементарная точка, отображаемая монитором или другим подобным устройством, например, смартфоном. Размер пиксела зависит от разрешения устройства и его технических характеристик. В примере 1.22 показано применение пикселов и em для задания размера шрифта.

Пример 1.22. Использование относительных единиц XHTML 1.0 CSS 2.1 IE 7 IE 8 IE 9 Cr 8 Op 11 Sa 5 Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Относительные единицы</title>
  <style type="text/css">
    H1 { font-size: 30px; }
    P { font-size: 1.5em; }
  </style>
```

```
 </head>
 <body>
  <h1>Заголовок размером 30 пикселов</h1>
  <p>Размер текста 1.5 em</p>
 </body>
</html>
```

Результат данного примера показан ниже (рис. 1.10).

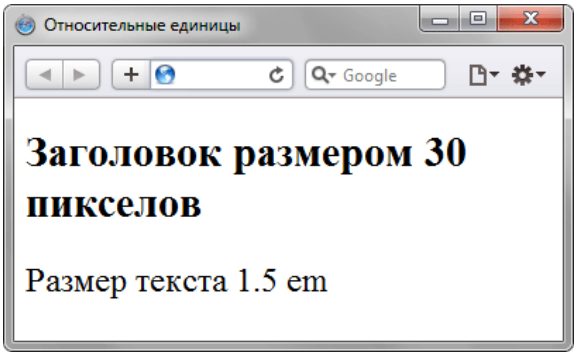


Рис. 1.10. Размер текста при различных единицах

Абсолютные единицы

Абсолютные единицы применяются реже, чем относительные и обычно при работе с текстом. В табл. 1.3 перечислены основные абсолютные единицы.

Табл. 1.3. Абсолютные единицы измерения

Единица	Описание
in	Дюйм (1 дюйм равен 2,54 см)
cm	Сантиметр
mm	Миллиметр
pt	Пункт (1 пункт равен 1/72 дюйма)
pc	Пика (1 пика равна 12 пунктам)

Самой, пожалуй, распространенной единицей является пункт, который используется для указания размера шрифта. Хотя мы привыкли измерять все в миллиметрах и подобных единицах, пункт, пожалуй, единственная величина из не метрической системы измерения, которая используется у нас повсеместно. И все благодаря текстовым редакторам и издательским системам. В примере 1.23 показано использование пунктов и миллиметров.

Пример 1.23. Использование абсолютных единиц

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Абсолютные единицы</title>
  <style type="text/css">
   H1 { font-size: 24pt; }
   P { margin-left: 30mm; }
  </style>
 </head>
 <body>
   <h1>Заголовок размером 24 пункта</h1>
   <p>Сдвиг текста вправо на 30 миллиметров</p>
 </body>
</html>
```

Результат использования абсолютных единиц измерения показан ниже (рис. 1.11).

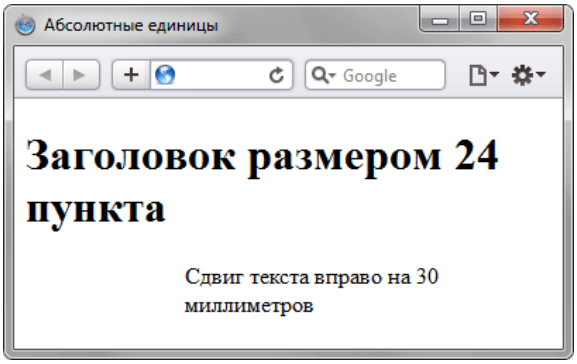


Рис. 1.11. Размер текста при различных единицах

При установке размеров обязательно указывайте единицы измерения, например width: 30px. В противном случае браузер не сможет показать желаемый результат, поскольку не понимает, какой размер вам требуется. Единицы не добавляются только при нулевом значении (margin: 0).

Цвет

Цвет в стилях можно задавать разными способами: по шестнадцатеричному значению, по названию, в формате RGB, RGBA, HSL, HSLA.

По шестнадцатеричному значению

Для задания цветов используются числа в шестнадцатеричном коде. Шестнадцатеричная система, в отличие от десятичной системы, базируется, как следует из её названия, на числе 16. Цифры будут следующие: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Цифры от 10 до 15 заменены латинскими буквами. Числа больше 15 в шестнадцатеричной системе образуются объединением двух чисел в одно. Например, числу 255 в десятичной системе соответствует число FF в шестнадцатеричной системе. Чтобы не возникало путаницы в определении системы счисления, перед шестнадцатеричным числом ставят символ решетки #, например #666999. Каждый из трех цветов — красный, зеленый и синий — может принимать значения от 00 до FF. Таким образом, обозначение цвета разбивается на три составляющие #rrggbb, где первые два символа отмечают красную компоненту цвета, два средних — зеленую, а два последних — синюю. Допускается использовать сокращенную форму вида #rgb, где каждый символ следует удваивать. Так, запись #fe0 следует расценивать как #ffee00.

По названию

Браузеры поддерживают некоторые цвета по их названию. В табл. 1.4 приведены названия, шестнадцатеричный код, значения в формате RGB, HSL и описание.

Табл. 1.4. Названия цветов

Имя	Цвет	Код	RGB	HSL	Описание
white		#ffffff или #fff	rgb(255,255,255)	hsl(0,0%,100%)	Белый
silver		#c0c0c0	rgb(192,192,192)	hsl(0,0%,75%)	Серый
gray		#808080	rgb(128,128,128)	hsl(0,0%,50%)	Темно-серый
black		#000000 или #000	rgb(0,0,0)	hsl(0,0%,0%)	Черный
maroon		#800000	rgb(128,0,0)	hsl(0,100%,25%)	Темно-красный
red		#ff0000 или #f00	rgb(255,0,0)	hsl(0,100%,50%)	Красный
orange		#ffa500	rgb(255,165,0)	hsl(38.8,100%,50%)	Оранжевый
yellow		#ffff00 или #ff0	rgb(255,255,0)	hsl(60,100%,50%)	Желтый
olive		#808000	rgb(128,128,0)	hsl(60,100%,25%)	Оливковый
lime		#00ff00 или #0f0	rgb(0,255,0)	hsl(120,100%,50%)	Светло-зеленый

green		#008000	rgb(0,128,0)	hsl(120,100%,25%)	Зеленый
aqua		#00ffff или #0ff	rgb(0,255,255)	hsl(180,100%,50%)	Голубой
blue		#0000ff или #00f	rgb(0,0,255)	hsl(240,100%,50%)	Синий
navy		#000080	rgb(0,0,128)	hsl(240,100%,25%)	Темно-синий
teal		#008080	rgb(0,128,128)	hsl(180,100%,25%)	Сине-зеленый
fuchsia		#ff00ff или #f0f	rgb(255,0,255)	hsl(300,100%,50%)	Розовый
purple		#800080	rgb(128,0,128)	hsl(300,100%,25%)	Фиолетовый

С помощью RGB

Можно определить цвет, используя значения красной, зеленой и синей составляющей в десятичном исчислении. Каждая из трех компонент цвета принимает значение от 0 до 255. Также допустимо задавать цвет в процентном отношении, при этом 100% будет соответствовать числу 255. Вначале указывается ключевое слово rgb, а затем в скобках, через запятую указываются компоненты цвета, например rgb(255, 128, 128) или rgb(100%, 50%, 50%).

RGBA

Формат RGBA похож по синтаксису на RGB, но включает в себя альфа-канал, задающий прозрачность элемента. Значение 0 соответствует полной прозрачности, 1 — непрозрачности, а промежуточное значение вроде 0.5 — полупрозрачности.

RGBA добавлен в CSS3, поэтому валидацию CSS-кода надо проводить именно по этой версии. Следует отметить, что стандарт CSS3 еще находится в разработке и некоторые возможности в нем могут поменяться. К примеру, цвет в формате RGB добавленный к свойству background-color проходит валидацию, а добавленный к свойству background уже нет. При этом браузеры вполне корректно понимают цвет для того и другого свойства. В табл. 1.4 приведен список браузеров, которые поддерживают RGBA; в браузерах, где этот формат не работает, значение цвета будет игнорироваться.

Табл. 1.4. Поддержка браузерами формата RGBA

Internet Explorer				Chrome				Opera				Safari			Firefox			
6.0	7.0	8.0	9.0	5.0	6.0	7.0	8.0	9.2	9.6	10	11	3.1	4.0	5.0	2.0	3.0	3.6	4.0
✗	✗	✗	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓

HSL

Название формата HSL образовано от сочетания первых букв Hue (оттенок), Saturate (насыщенность) и Lightness (светлота). Оттенок это значение цвета на цветовом круге (рис. 1.12) и задаётся в градусах. 0° соответствует красному цвету, 120° — зелёному, а 240° — синему. Значение оттенка может изменяться от 0 до 359.

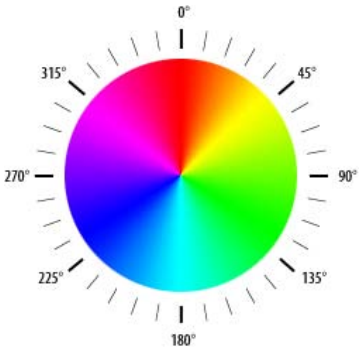


Рис. 1.12. Цветовой круг

Насыщенностью называется интенсивность цвета, измеряется в процентах от 0% до 100%. Значение 0% обозначает отсутствие цвета и оттенок серого, 100% максимальное значение насыщенности.

Светлота задает, насколько цвет яркий и указывается в процентах от 0% до 100%. Малые значения делают цвет темнее, а высокие светлее, крайние значения 0% и 100% соответствуют чёрному и белому цвету.

Примеры использование HSL приведены в табл. 1.4. За исключением Internet Explorer все современные браузеры корректно работают с этим форматом (табл. 1.5).

Табл. 1.5. Поддержка браузерами формата HSL

Internet Explorer				Chrome				Opera				Safari			Firefox			
6.0	7.0	8.0	9.0	5.0	6.0	7.0	8.0	9.2	9.5	10	11	3.1	4.0	5.0	2.0	3.0	3.6	4.0
✖	✖	✖	✖	✔	✔	✔	✔	✖	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

HSLA

Формат HSLA похож по синтаксису на HSL, но включает в себя альфа-канал, задающий прозрачность элемента. Значение 0 соответствует полной прозрачности, 1 — непрозрачности, а промежуточное значение вроде 0.5 — полупрозрачности. По сравнению с HSL поддержка браузерами немного другая (табл. 1.6).

Табл. 1.6. Поддержка браузерами формата HSLA

Internet Explorer				Chrome				Opera				Safari			Firefox			
6.0	7.0	8.0	9.0	5.0	6.0	7.0	8.0	9.2	9.6	10	11	3.1	4.0	5.0	2.0	3.0	3.6	4.0
✖	✖	✖	✖	✔	✔	✔	✔	✖	✖	✔	✔	✔	✔	✔	✖	✔	✔	✔

Значения цвета в форматах RGBA, HSL и HSLA добавлены в CSS3, поэтому при использовании этих форматов проверяйте код на валидность с учётом версии.

В примере 1.24 представлены различные способы задания цветов элементов веб-страниц.

Пример 1.24. Представление цвета

XHTML 1.0 CSS 2.1 CSS 3 IE 7 IE 8 IE 9 Cr 8 Op 11 Sa 5 Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Цвета</title>
  <style type="text/css">
   BODY {
    background-color: #F9F2E3;
   }
   H2 {
    background-color: rgb(214,86,43);
    color: rgba(255,255,255,.9);
    padding: 10px;
   }
   P {
    color: green;
   }
   DIV {
    background-color: hsl(60,100%,25%);
    color: hsla(120,100%,50%,0.1);
   }
  </style>
 </head>
 <body>
  <h2>Предупреждение</h2>
   <p>Все перечисленные на сайте методы ловли льва являются теоретическими
   и базируются на вычислительных методах. Авторы не гарантируют вашей
   безопасности при их использовании и снимают с себя всякую
   ответственность за результат. Помните, лев это хищник и
   опасное животное!</p>
  <div>Apppprx!</div>
 </body>
</html>
```

Результат данного примера показан на рис. 1.13.

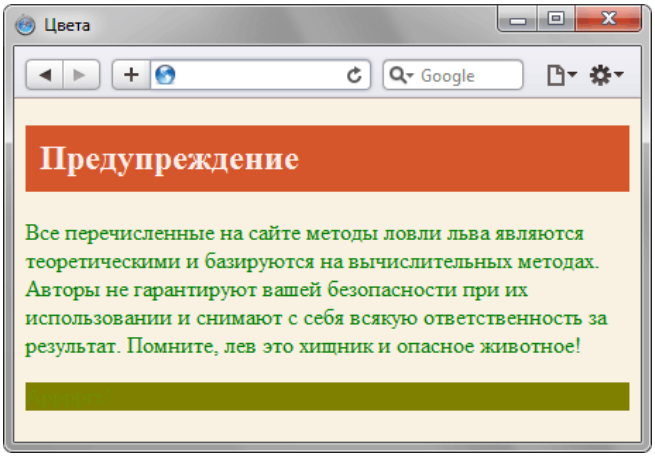


Рис. 1.13. Цвета на веб-странице

Адреса

Адреса применяются для указания пути к файлу, например, для установки фоновой картинки на странице. Для этого применяется ключевое слово url(), внутри скобок пишется относительный или абсолютный адрес файла. При этом адрес можно задавать в необязательных одинарных или двойных кавычках (пример 1.25).

Пример 1.25. Адрес графического файла

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Добавление фона</title>
  <style type="text/css">
   BODY {
    background: url('http://webimg.ru/images/156_1.png') no-repeat;
   }
   DIV {
    background: url(images/warning.png) no-repeat;
    padding-left: 140px;
   }
  </style>
 </head>
 <body>
   <div>Внимание, запрашиваемая страница не найдена!</div>
 </body>
</html>
```

В данном примере в селекторе **BODY** используется абсолютный адрес к графическому файлу, а в селекторе **DIV** — относительный.

Ключевые слова

В качестве значений активно применяются ключевые слова, которые определяют желаемый результат действия стилевых свойств. Ключевые слова пишутся без кавычек.

```
Правильно: P { text-align: right; }
Неверно: P { text-align: "right"; }
```

inherit

Ключевое слово, которое сообщает, что необходимо наследовать значение свойства у родительского элемента. Естественно, результат будет заметен только в том случае, если у родителя указанное свойство установлено. Браузер Internet Explorer до версии 7.0 включительно не поддерживает значение inherit.

initial

Значение initial применяется для установки исходного значения свойства. Может пригодиться в нескольких случаях, к примеру, восстановить значения свойств, заданных браузером по умолчанию или задать начальное значение свойства, измененное в результате наследования. Ключевое слово initial добавлено в CSS3 и пока плохо поддерживается браузерами (табл. 1.7).

Табл. 1.7. Поддержка браузерами значения initial

Браузер	Internet Explorer	Chrome	Opera	Safari	Firefox
Версия	—	2.0+	—	2.0+	1.0+
Значение	—	initial	—	initial	-moz-initial

Значение -moz-initial является нестандартным, поэтому его применение приведёт к невалидному коду CSS.

В примере 1.26 показаны некоторые аспекты применения initial.

Пример 1.26. Использование initial

XHTML 1.0 | CSS 2.1 | CSS 3 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>initial</title>
  <style type="text/css">
   H2 {
    color: #ffb734;
    font-family: Arial, sans-serif;
   }
   P {
    color: green;
   }
   .initial {
    color: initial;
    color: -moz-initial;
    font-family: initial;
    font-family: -moz-initial;
   }
  </style>
 </head>
 <body>
  <h2>Метод случайных чисел</h2>
  <p>Разделим пустыню на <span class="initial">ряд элементарных
   прямоугольников</span>, размер которых совпадает с размером
   клетки для льва. После чего перебираем полученные прямоугольники,
   каждый раз выбирая заданную область случайным образом.
   Если в данной области окажется лев, то мы поймаем его, накрыв
   клеткой.</p>
  <h2 class="initial">Метод Гаусса</h2>
</body>
</html>
```

В данном примере изменяется цвет текста и шрифт заголовка. Цвет текста внутри тега **** явно не задаётся, но он наследует значение цвета своего родителя — тега **<p>**. С помощью класса `initial` цвет фрагмента текста устанавливается исходным, по умолчанию он черный. Аналогично обстоит дело и с заголовком, через стили задается его цвет и шрифт. Чтобы восстановить стиль по умолчанию, к тегу **<h1>** добавляется класс `initial`, в котором используется значение initial (рис. 1.14).

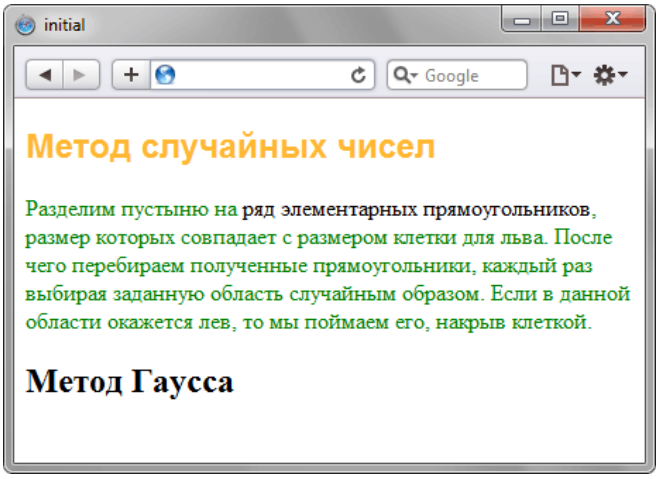


Рис. 1.14. Результат использования initial

Селекторы тегов

В качестве селектора может выступать любой тег HTML для которого определяются правила форматирования, такие как: цвет, фон, размер и т.д. Правила задаются в следующем виде.

```
Тег { свойство1: значение; свойство2: значение; ... }
```

Вначале указывается имя тега, оформление которого будет переопределено, заглавными или строчными символами не имеет значения. Внутри фигурных скобок пишется стилевое свойство, а после двоеточия — его значение. Набор свойств разделяется между собой точкой с запятой и может располагаться как в одну строку, так и в несколько (пример 1.27).

Пример 1.27. Изменение стиля тега <p>

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Селекторы тегов</title>
  <style type="text/css">
   P {
    text-align: justify; /* Выравнивание по ширине */
    color: green; /* Зеленый цвет текста */
   }
  </style>
 </head>
 <body>
  <p>Более эффективным способом ловли льва в пустыне
  является метод золотого сечения. При его использовании пустыня делится
  на две неравные части, размер которых подчиняется правилу золотого
  сечения.</p>
 </body>
</html>
```

В данном примере изменяется цвет текста и выравнивание текста абзаца. Стиль будет применяться только к тексту, который располагается внутри контейнера **<p>**.

Следует понимать, что хотя стиль можно применить к любому тегу, результат будет заметен только для тегов, которые непосредственно отображаются в контейнере **<body>**.

Классы

Классы применяют, когда необходимо определить стиль для индивидуального элемента веб-страницы или задать разные стили для одного тега. При использовании совместно с тегами синтаксис для классов будет следующий.

```
Тег.Имя класса { свойство1: значение; свойство2: значение; ... }
```

Внутри стиля вначале пишется желаемый тег, а затем, через точку пользовательское имя класса. Имена классов должны начинаться с латинского символа и могут содержать в себе символ дефиса (-) и подчеркивания (_). Использование русских букв в именах классов недопустимо. Чтобы указать в коде HTML, что тег используется с определенным классом, к тегу добавляется атрибут class значением которого выступает имя класса (пример 1.28).

Пример 1.28. Использование классов

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Классы</title>
  <style type="text/css">
   P { /* Обычный абзац */
    text-align: justify; /* Выравнивание текста по ширине */
   }
   P.cite { /* Абзац с классом cite */
    color: navy; /* Цвет текста */
    margin-left: 20px; /* Отступ слева */
    border-left: 1px solid navy; /* Граница слева от текста */
    padding-left: 15px; /* Расстояние от линии до текста */
   }
  </style>
 </head>
 <body>
  <p>Для искусственного освещения помещения применяются люминесцентные лампы.
     Они отличаются высокой световой отдачей, продолжительным сроком службы,
     малой яркостью светящейся поверхности, близким к естественному
     спектральным составом излучаемого света, что обеспечивает хорошую
     цветопередачу.</p>
  <p class="cite">Для исключения засветки экрана дисплея световыми потоками
     оконные проемы снабжены светорассеивающими шторами.</p>
 </body>
</html>
```

Результат данного примера показан на рис. 1.15.

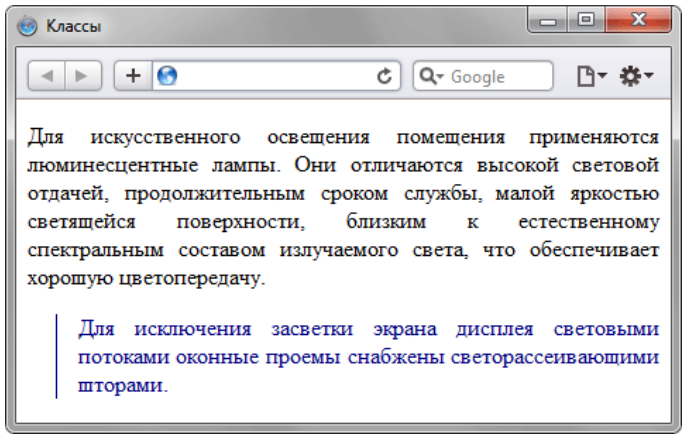


Рис. 1.15. Вид текста, оформленного с помощью стилевых классов

Первый абзац выровнен по ширине с текстом черного цвета (этот цвет задается браузером по умолчанию), а следующий, к которому применен класс с именем `cite` — отображается синим цветом и с линией слева.

Можно, также, использовать классы и без указания тега. Синтаксис в этом случае будет следующий.

```
.Имя класса { свойство1: значение; свойство2: значение; ... }
```

При такой записи, класс можно применять к любому тегу (пример 1.29).

Пример 1.29. Использование классов

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Классы</title>
  <style type="text/css">
   .gost {
    color: green; /* Цвет текста */
    font-weight: bold; /* Жирное начертание */
   }
   .term {
    border-bottom: 1px dashed red; /* Подчеркивание под текстом */
   }
  </style>
 </head>
 <body>
   <p>Согласно <span class="gost">ГОСТ 12.1.003-83 ССБТ &quot;Шум. Общие
      требования безопасности&quot;</span>, шумовой характеристикой рабочих
      мест при постоянном шуме являются уровни звуковых давлений в децибелах
      в октавных полосах. Совокупность таких уровней называется
      <b class="term">предельным спектром</b>, номер которого численно равен
      уровню звукового давления в октавной полосе со среднегеометрической
      частотой 1000 Гц.</p>
 </body>
</html>
```

Результат применения классов к тегам **** и **<b>** показан на рис. 1.16.

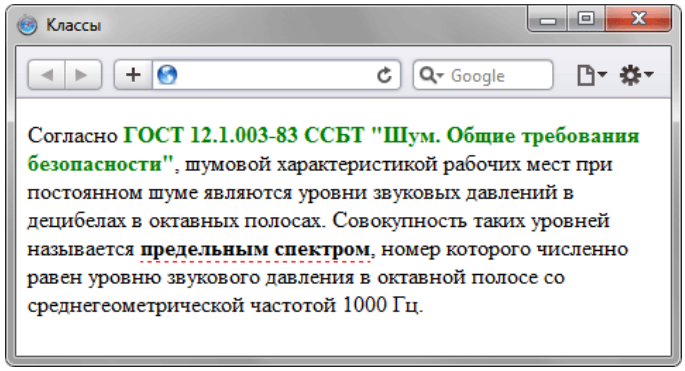


Рис. 1.16. Вид тегов, оформленных с помощью классов

Классы удобно использовать, когда нужно применить стиль к разным элементам веб-страницы: ячейкам таблицы, ссылкам, абзацам и др. В примере 1.30 показано изменение цвета фона строк таблицы для создания «зебры».

Пример 1.30. Использование классов

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Камни</title>
  <style type="text/css">
   table.jewel {
    width: 100%; /* Ширина таблицы */
    border: 1px solid #666; /* Рамка вокруг таблицы */
   }
   th {
    background: #009383; /* Цвет фона */
    color: #fff; /* Цвет текста */
    text-align: left; /* Выравнивание по левому краю */
   }
   tr.odd {
    background: #ebd3d7; /* Цвет фона */
```

```
    }
   </style>
  </head>
  <body>
   <table class="jewel">
    <tr>
     <th>Название</th><th>Цвет</th><th>Твердость по Моосу</th>
    </tr>
    <tr class="odd">
     <td>Алмаз</td><td>Белый</td><td>10</td>
    </tr>
    <tr>
     <td>Рубин</td><td>Красный</td><td>9</td>
    </tr>
    <tr class="odd">
     <td>Аметист</td><td>Голубой</td><td>7</td>
    </tr>
    <tr>
     <td>Изумруд</td><td>Зеленый</td><td>8</td>
    </tr>
    <tr class="odd">
     <td>Сапфир</td><td>Голубой</td><td>9</td>
    </tr>
   </table>
  </body>
</html>
```

Результат данного примера показан на рис. 1.17. В примере класс с именем odd используется для изменения цвета фона строки таблицы. За счет того, что этот класс добавляется не ко всем тегам `<tr>` и получается чередование разных цветов.

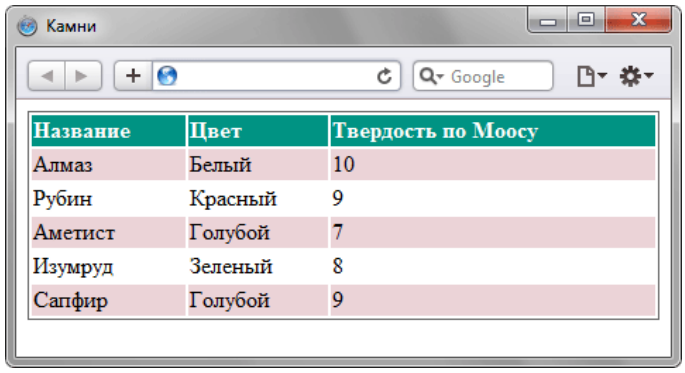


Рис. 1.17. Результат применения классов

Одновременно использование разных классов

К любому тегу можно одновременно добавить несколько классов, перечисляя их в значении атрибута class через пробел, такая форма называется мультиклассы. В этом случае к элементу применяется стиль, описанный в правилах для каждого класса. Поскольку при добавлении нескольких классов они могут содержать одинаковые стилевые свойства, но с разными значениями, то берется значение у класса, который описан в коде ниже. В примере 1.31 показано использование разных классов для создания облака тегов.

Пример 1.31. Сочетание разных классов

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Облако тегов</title>
  <style type="text/css">
   .level1 { font-size: 1em; }
   .level2 { font-size: 1.2em; }
   .level3 { font-size: 1.4em; }
   .level4 { font-size: 1.6em; }
   .level5 { font-size: 1.8em; }
   .level6 { font-size: 2em; }
   A.tag {
    color: #468be1; /* Цвет ссылок */
   }
```

```
  </style>
 </head>
 <body>
  <div>
   <a href="/term/2" class="tag level6">Paint.NET</a>
   <a href="/term/69" class="tag level6">Photoshop</a>
   <a href="/term/3" class="tag level5">цвет</a>
   <a href="/term/95" class="tag level5">фон</a>
   <a href="/term/11" class="tag level4">палитра</a>
   <a href="/term/43" class="tag level3">слои</a>
   <a href="/term/97" class="tag level2">свет</a>
   <a href="/term/44" class="tag level2">панели</a>
   <a href="/term/16" class="tag level1">линия</a>
   <a href="/term/33" class="tag level1">прямоугольник</a>
   <a href="/term/14" class="tag level1">пиксел</a>
   <a href="/term/27" class="tag level1">градиент</a>
  </div>
 </body>
</html>
```

Результат данного примера показан на рис. 1.18.

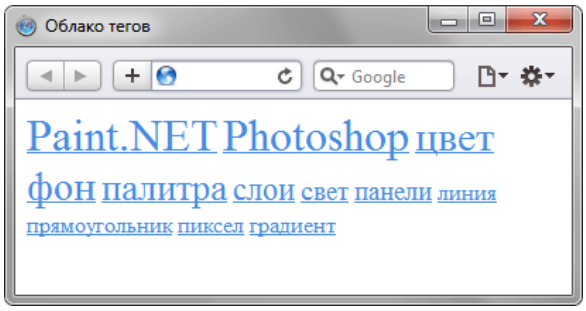


Рис. 1.18. Облако тегов

В стилях также допускается использовать запись вида .layer1.layer2, где `layer1` и `layer2` представляют собой имена классов. Стиль применяется только для элементов, у которых одновременно заданы классы `layer1` и `layer2` (пример 1.32).

Пример 1.32. Использование мультиклассов

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Мультиклассы</title>
  <style type="text/css">
   .layer1 { color: red; }
   .layer2 { color: blue; }
   .layer1.layer2 {color: green; }
  </style>
 </head>
 <body>
  <p class="layer1">Текст красного цвета</p>
  <p class="layer2">Текст синего цвета</p>
  <p class="layer1 layer2">Текст зелёного цвета</p>
 </body>
</html>
```

Браузер IE6 некорректно работает с мультиклассами и понимает запись `.a.b` как `.b`, т.е. воспринимает только имя последнего класса, что приводит к ошибкам в работе данного примера. Текст с классом `layer2` будет отображаться зеленым цветом, а не синим, как должно быть.

Идентификаторы

Идентификатор (называемый также «ID селектор») определяет уникальное имя элемента, которое используется для изменения его стиля и обращения к нему через скрипты.

Синтаксис применения идентификатора следующий.

```
#Имя идентификатора { свойство1: значение; свойство2: значение; ... }
```

При описании идентификатора вначале указывается символ решетки (#), затем идет имя идентификатора. Оно должно начинаться с латинского символа и может содержать в себе символ дефиса (-) и подчеркивания (_). Использование русских букв в именах идентификатора недопустимо. В отличие от классов идентификаторы должны быть уникальны, иными словами, встречаться в коде документа только один раз.

Обращение к идентификатору происходит аналогично классам, но в качестве атрибута тега используется id, значением которого выступает имя идентификатора (пример 1.33). Символ решетки при этом уже не указывается.

Пример 1.33. Использование идентификатора

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Идентификаторы</title>
  <style type="text/css">
   #help {
    position: absolute; /* Абсолютное позиционирование */
    right: 20px; /* Положение от правого края */
    top: 50px; /* Положение от верхнего края */
    width: 225px; /* Ширина блока */
    padding: 5px; /* Поля вокруг текста */
    background: #f0f0f0; /* Цвет фона */
    display: none; /* Скрыть */
   }
  </style>
 </head>
 <body>
  <p><a href="#"
   onclick="document.getElementById('help').style.display='block'">
    Справка</a></p>
  <div id="help">
   Эта справка помогает в случае, когда вы находитесь в осознании того
   факта, что совершенно не понимаете, кто и как вам может помочь. Именно
   в этот момент мы и подсказываем, что помочь вам никто не сможет.
  </div>
 </body>
</html>
```

В данном примере определяется стиль тега ** ** через идентификатор с именем `help`. Исходно этот слой скрывается от просмотра и делается видимым при нажатии на ссылку, вызывающую небольшой скрипт через событие onclick.

Как и при использовании классов, идентификаторы можно применять к конкретному тегу. Синтаксис при этом будет следующий.

```
Тег#Имя идентификатора { свойство1: значение; свойство2: значение; ... }
```

Вначале указывается имя тега, затем без пробелов символ решетки и название идентификатора. В примере 1.34 показано использование идентификатора применительно к тегу **<p>**.

Пример 1.34. Применение идентификатора совместно с тегом

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Идентификаторы</title>
  <style type="text/css">
   P {
    color: green; /* Зеленый цвет текста */
    font-style: italic; /* Курсивное начертание текста */
   }
   P#opa {
    color: red; /* Красный цвет текста */
    border: 1px solid #666; /* Параметры рамки */
    background: #eee; /* Цвет фона */
    padding: 5px; /* Поля вокруг текста */
   }
  </style>
  <script type="text/javascript">
   function newText() {
    document.getElementById("opa").innerHTML = 'Необычный текст';
   }
  </script>
 </head>
 <body onload="newText()">
  <p>Обычный текст</p>
  <p id="opa"></p>
 </body>
</html>
```

Результат данного примера показан на рис. 1.19.

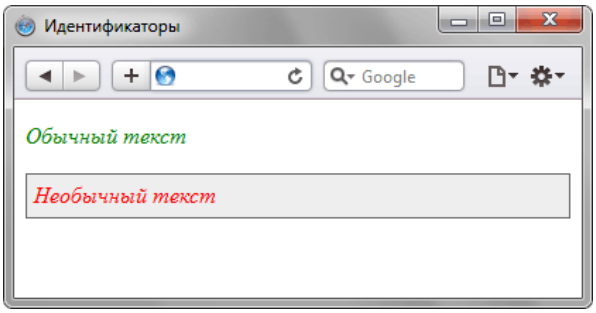


Рис. 1.19. Вид веб-страницы

В данном примере вводится идентификатор с именем opa, который применяется к тегу **<p>**. В функции идентификатора входит изменение стиля абзаца и вывод в него текста через скрипт. Конкретно эта задача возложена на метод innerHTML.

Контекстные селекторы

При создании веб-страницы часто приходится вкладывать одни теги внутрь других. Чтобы стили для этих тегов использовались корректно, помогут селекторы, которые работают только в определенном контексте. Например, задать стиль для тега **<b>** только когда он располагается внутри контейнера **<p>**. Таким образом можно одновременно установить стиль для отдельного тега, а также для тега, который находится внутри другого.

Контекстный селектор состоит из простых селекторов разделенных пробелом. Так, для селектора тега синтаксис будет следующий.

```
Тег1 Тег2 { ... }
```

В этом случае стиль будет применяться к Тегу2 когда он размещается внутри Тега1, как показано ниже.

```
<Тег1>
 <Тег2> ... </Тег2>
</Тег1>
```

Использование контекстных селекторов продемонстрировано в примере 1.35.

Пример 1.35. Контекстные селекторы

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Контекстные селекторы</title>
  <style type="text/css">
   P B {
    font-family: Times, serif; /* Семейство шрифта */
    font-weight: bold; /* Жирное начертание */
    color: navy; /* Синий цвет текста */
   }
  </style>
 </head>
 <body>
  <div><b>Жирное начертание текста</b></div>
  <p><b>Одновременно жирное начертание текста
  и выделенное цветом</b></p>
 </body>
</html>
```

В данном примере показано обычное применение тега **<b>** и этого же тега, когда он вложен внутрь абзаца **<p>**. При этом меняется цвет и шрифт текста, как показано на рис. 1.19.

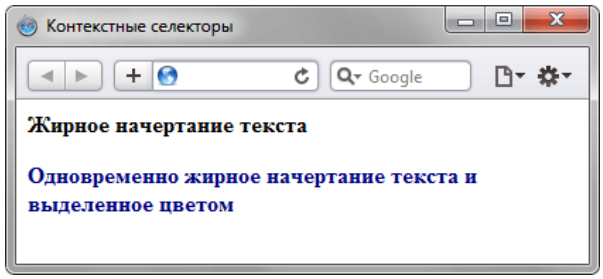


Рис. 1.19. Оформление текста в зависимости от вложенности тегов

Не обязательно контекстные селекторы содержат только один вложенный тег. В зависимости от ситуации допустимо применять два и более последовательно вложенных друг в друга тегов.

Более широкие возможности контекстные селекторы дают при использовании идентификаторов и классов. Это позволяет устанавливать стиль только для того элемента, который располагается внутри

определенного класса, как показано в примере 10.2.

Пример 1.36. Использование классов

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Контекстные селекторы</title>
  <style type="text/css">
   A {
    color: green; /* Зеленый цвет текста для всех ссылок */
   }
   .menu {
    padding: 7px; /* Поля вокруг текста */
    border: 1px solid #333; /* Параметры рамки */
    background: #fc0; /* Цвет фона */
   }
   .menu A {
    color: navy; /* Темно-синий цвет ссылок */
   }
  </style>
 </head>
 <body>
  <div class="menu">
    <a href="1.html">Русская кухня</a> |
    <a href="2.html">Украинская кухня</a> |
    <a href="3.html">Кавказская кухня</a>
  </div>
  <p><a href="text.html">Другие материалы по теме</a></p>
 </body>
</html>
```

Результат данного примера показан на рис. 1.20.

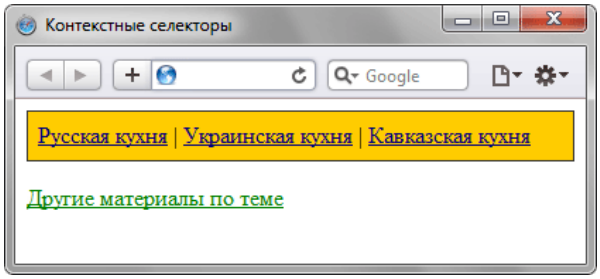


Рис. 1.20. Ссылки разных цветов

В данном примере используется два типа ссылок. Первая ссылка, стиль которой задается с помощью селектора `A`, будет действовать на всей странице, а стиль второй ссылки (.menu A) применяется только к ссылкам внутри элемента с классом `menu`.

При таком подходе легко управлять стилем одинаковых элементов, вроде изображений и ссылок, оформление которых должно различаться в разных областях веб-страницы.

Соседние селекторы

Соседними называются элементы веб-страницы, когда они следуют непосредственно друг за другом в коде документа. Рассмотрим несколько примеров отношения элементов.

```
<p>Lorem ipsum <b>dolor</b> sit amet.</p>
```

В этом примере тег **<b>** является дочерним по отношению к тегу **<p>**, поскольку он находится внутри этого контейнера. Соответственно **<p>** выступает в качестве родителя **<b>**.

```
<p>Lorem ipsum <b>dolor</b> <var>sit</var> amet.</p>
```

Здесь теги **<var>** и **<b>** никак не перекрываются и представляют собой соседние элементы. То, что они расположены внутри контейнера **<p>**, никак не влияет на их отношение.

```
<p>Lorem <b>ipsum </b> dolor sit amet, <i>consectetuer</i> adipiscing
<tt>elit</tt>.</p>
```

Соседними здесь являются теги **<b>** и **<i>**, а также **<i>** и **<tt>**. При этом **<b>** и **<tt>** к соседним элементам не относятся из-за того, что между ними расположен контейнер **<i>**.

Для управления стилем соседних элементов используется символ плюса (+), который устанавливается между двумя селекторами. Общий синтаксис следующий.

```
Селектор 1 + Селектор 2 { Описание правил стиля }
```

Пробелы вокруг плюса не обязательны, стиль при такой записи применяется к Селектору 2, но только в том случае, если он является соседним для Селектора 1 и следует сразу после него.

В примере 1.37 показана структура взаимодействия тегов между собой.

Пример 1.37. Использование соседних селекторов

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Соседние селекторы</title>
  <style type="text/css">
   B + I {
    color: red; /* Красный цвет текста */
   }
  </style>
 </head>
 <body>
   <p>Lorem <b>ipsum </b> dolor sit amet, <i>consectetuer</i>
    adipiscing elit.</p>
   <p>Lorem ipsum dolor sit amet, <i>consectetuer</i> adipiscing elit.</p>
 </body>
</html>
```

Результат примера показан на рис. 1.22.

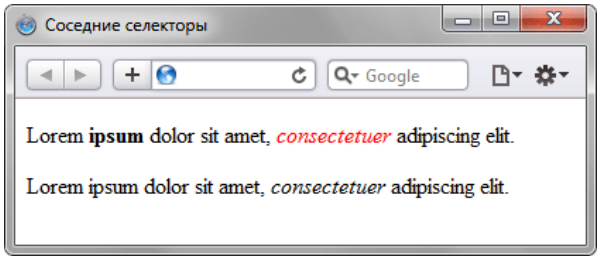


Рис. 1.22. Выделение текста цветом при помощи соседних селекторов

В данном примере происходит изменение цвета текста для содержимого контейнера **<i>**, когда он располагается сразу после контейнера **<b>**. В первом абзаце такая ситуация реализована, поэтому слово «consectetuer» в браузере отображается красным цветом. Во втором абзаце, хотя и присутствует тег **<i>**, но по соседству никакого тега **<b>** нет, так что стиль к этому контейнеру не применяется.

Разберем более практичный пример. Часто возникает необходимость в текст статьи включать различные сноски и примечания. Обычно для этой цели создают новый стилевой класс и применяют его к абзацу, таким способом можно легко изменить вид текста. Но мы пойдем другим путем и воспользуемся соседними селекторами. Для выделения замечаний создадим новый класс, назовем его `sic`, и станем применять его к тегу **<h2>**. Первый абзац после такого заголовка выделяется цветом фона и отступом (пример 1.38). Вид остальных абзацев останется неизменным.

Пример 1.38. Изменение стиля абзаца

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Изменение стиля абзаца</title>
  <style type="text/css">
   H2.sic {
    font-size: 140%; /* Размер шрифта */
    color: maroon; /* Цвет текста */
    font-weight: normal; /* Нормальное начертание текста */
    margin-left: 30px; /* Отступ слева */
    margin-bottom: 0px; /* Отступ снизу */
   }
   H2.sic + P {
   background: #ddd; /* Цвет фона */
   margin-left: 30px; /* Отступ слева */
   margin-top: 0.5em; /* Отступ сверху */
   padding: 7px; /* Поля вокруг текста */
  }
  </style>
 </head>
 <body>
  <h1>Методы ловли льва в пустыне</h1>
  <h2>Метод последовательного перебора</h2>
  <p>Пусть лев имеет габаритные размеры L x W x H, где L - длина льва от
  кончика носа до кисточки хвоста, W - ширина льва, а H - его высота.
  После чего пустыню разбиваем на ряд элементарных прямоугольников, размер
  которых совпадает с шириной и длиной льва. Учитывая, что лев может
  находиться не строго на заданном участке, а одновременно на двух из них,
  клетку для ловли следует делать повышенной площади, а именно 2L x 2W.
  Благодаря этому мы избежим ошибки, когда в клетке
  окажется пойманным лишь половина льва или, что хуже, только его хвост.</p>
  <h2 class="sic">Важное замечание</h2>
  <p>Для упрощения расчетов хвост в качестве погрешности измерения можно
  отбросить и не принимать во внимание.</p>
  <p>Далее последовательно накрываем каждый из размеченных прямоугольников
  пустыни клеткой и проверяем, пойман лев или нет. Как только лев окажется
  в клетке, процедура поимки считается завершенной.</p>
 </body>
</html>
```

Результат данного примера показан на рис. 1.23.

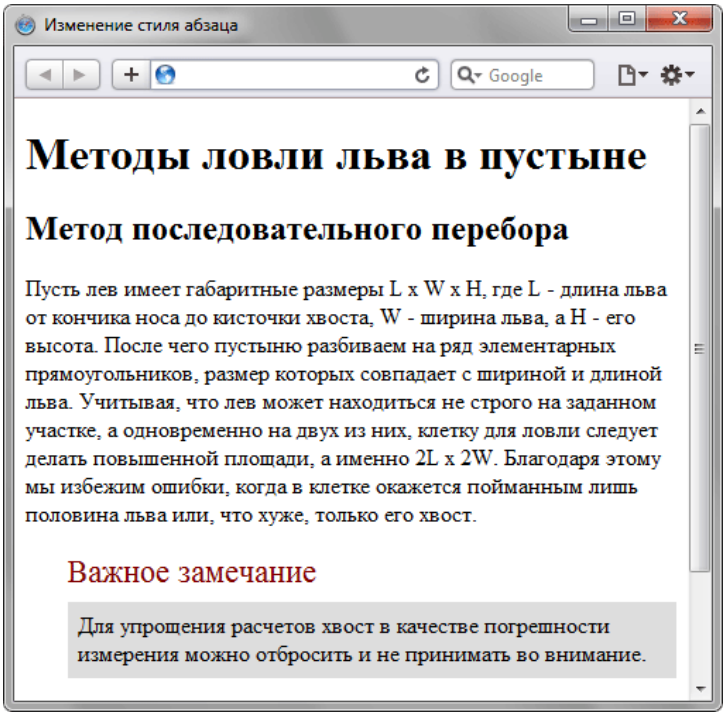


Рис. 11.2. Изменение вида абзаца за счет использования соседних селекторов

В данном примере текст отформатирован с применением абзацев (тег **<p>**), но запись `H2.sic + P` устанавливает стиль только для первого абзаца идущего после тега **<h2>**, у которого добавлен класс `sic`.

Соседние селекторы удобно использовать для тех тегов, к которым автоматически добавляются отступы, чтобы самостоятельно регулировать величину отбивки. Например, если подряд идут теги **<h1>** и **<h2>**, то расстояние между ними легко регулировать как раз с помощью соседних селекторов. Аналогично дело обстоит и для идущих подряд тегов **<h2>** и **<p>**, а также в других подобных случаях. В примере 1.39 таким манером изменяется величина отступов между указанными тегами.

Пример 1.39. Отступы между заголовками и текстом

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Соседние селекторы</title>
  <style type="text/css">
   H1 + H2 {
    margin-top: -10px; /* Смещаем заголовок 2 вверх */
   }
   H2 + P {
    margin-top: -1em; /* Смещаем первый абзац вверх к заголовку */
   }
  </style>
 </head>
 <body>
  <h1>Заголовок 1</h1>
  <h2>Заголовок 2</h2>
  <p>Абзац!</p>
 </body>
</html>
```

Поскольку при использовании соседних селекторов стиль применяется только ко второму элементу, то размер отступов уменьшается за счет включения отрицательного значения у свойства margin-top. При этом текст поднимается вверх, ближе к предыдущему элементу.

Дочерние селекторы

Дочерним называется элемент, который непосредственно располагается внутри родительского элемента. Чтобы лучше понять отношения между элементами документа, разберем небольшой код (пример 1.40).

Пример 1.40. Вложенность элементов в документе

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Lorem ipsum</title>
 </head>
 <body>
  <div class="main">
   <p><em>Lorem ipsum dolor sit amet</em>, consectetuer adipiscing
   elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore
   magna aliguam erat volutpat.</p>
   <p><strong><em>Ut wisis enim ad minim veniam</em></strong>,
   quis nostrud exerci tution ullamcorper suscipit lobortis nisl ut
   aliquip ex ea commodo consequat.</p>
  </div>
 </body>
</html>
```

В данном примере применяется несколько контейнеров, которые в коде располагаются один в другом. Нагляднее это видно на дереве элементов, так называется структура отношений тегов документа между собой (рис. 1.24).

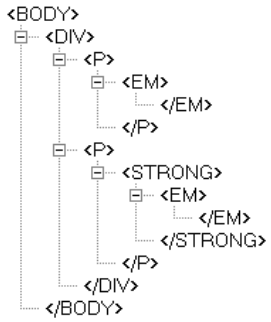


Рис. 1.24. Дерево элементов для примера

На данном рисунке в удобном виде представлена вложенность элементов и их иерархия. Здесь дочерним элементом по отношению к тегу ** ** выступает тег **<p>**. Вместе с тем тег **** не является дочерним для тега ** **, поскольку он расположен в контейнере **<p>**.

Вернемся теперь к селекторам. Дочерним селектором считается такой, который в дереве элементов находится прямо внутри родительского элемента. Синтаксис применения таких селекторов следующий.

```
Селектор 1 > Селектор 2 { Описание правил стиля }
```

Стиль применяется к Селектору 2, но только в том случае, если он является дочерним для Селектора 1.

Если снова обратиться к примеру 1.40, то стиль вида P > EM { color: red } будет установлен для первого абзаца документа, поскольку тег **** находится внутри контейнера **<p>**, и не даст никакого результата для второго абзаца. А все из-за того, что тег **** во втором абзаце расположен в контейнере ****, поэтому нарушается условие вложенности.

По своей логике дочерние селекторы похожи на селекторы контекстные. Разница между ними следующая. Стиль к дочернему селектору применяется только в том случае, когда он является прямым

потомком, иными словами, непосредственно располагается внутри родительского элемента. Для контекстного селектора же допустим любой уровень вложенности. Чтобы стало понятно, о чем идет речь, разберем следующий код (пример 1.41).

Пример 1.41. Контекстные и дочерние селекторы

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Дочерние селекторы</title>
  <style type="text/css">
   DIV I { /* Контекстный селектор */
    color: green; /* Зеленый цвет текста */
   }
   P > I { /* Дочерний селектор */
    color: red; /* Красный цвет текста */
   }
  </style>
 </head>
 <body>
  <div>
   <p>Известно, что пустыня обладает неправильной формой, которую
    принимаем прямоугольной. Это достигается одним из двух способов —
    <i>включением частей</i>, выходящих за пределы области пустыни
    или их <i>отбрасыванием</i>.</p>
  </div>
 </body>
</html>
```

Результат данного примера показан на рис. 1.25.

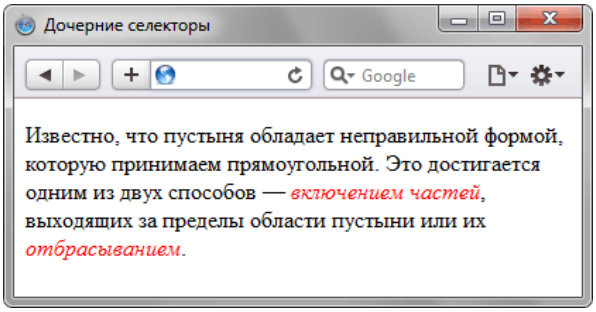


Рис. 1.25. Цвет текста, заданный с помощью дочернего селектора

На тег `<i>` в примере действуют одновременно два правила: контекстный селектор (тег `<i>` расположен внутри `<div>`) и дочерний селектор (тег `<i>` является дочерним по отношению к `<p>`). При этом правила являются равносильными, поскольку все условия для них выполняются и не противоречат друг другу. В подобных случаях применяется стиль, который расположен в коде ниже, поэтому курсивный текст отображается красным цветом. Стоит поменять правила местами и поставить `DIV I` ниже, как цвет текста изменится с красного на зеленый.

Заметим, что в большинстве случаев от добавления дочерних селекторов можно отказаться, заменив их контекстными селекторами. Однако использование дочерних селекторов расширяет возможности по управлению стилями элементов, что в итоге позволяет получить нужный результат, а также простой и наглядный код.

Удобнее всего применять указанные селекторы для элементов, которые обладают иерархической структурой — сюда относятся, например, таблицы и разные списки. В примере 1.42 показано изменение вида списка с помощью стилей. За счет вложения одного списка в другой получаем разновидность меню. Заголовки при этом располагаются горизонтально, а набор ссылок — вертикально под заголовками (рис. 1.26).

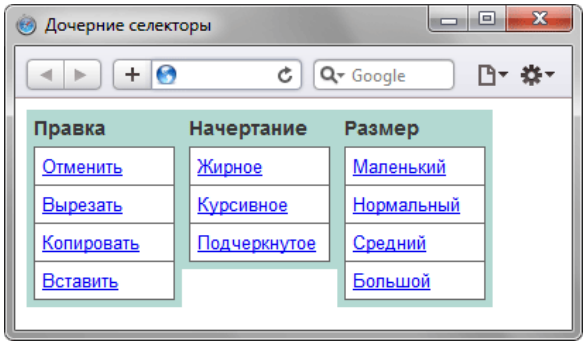


Рис. 1.26. Список в виде меню

Для размещения текста по горизонтали к селектору **LI** добавляется стилевое свойство float. Чтобы при этом разделить между собой стиль горизонтального и вертикального списка и применяются дочерние селекторы (пример 1.42).

Пример 1.42. Использование дочерних селекторов

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Дочерние селекторы</title>
  <style type="text/css">
   UL#menu {
    margin: 0; padding: 0; /* Убираем отступы */
   }
   UL#menu > LI {
    list-style: none; /* Убираем маркеры списка */
    width: 100px; /* Ширина элемента в пикселах */
    background: #b3d9d2; /* Цвет фона */
    color: #333; /* Цвет текста */
    padding: 5px; /* Поля вокруг текста */
    font-family: Arial, sans-serif; /* Рубленый шрифт */
    font-size: 90%; /* Размер шрифта */
    font-weight: bold; /* Жирное начертание */
    float: left; /* Располагаем элементы по горизонтали */
   }
   LI > UL {
    list-style: none; /* Убираем маркеры списка */
    margin: 0; padding: 0; /* Убираем отступы вокруг элементов списка */
    border-bottom: 1px solid #666; /* Граница внизу */
    padding-top: 5px; /* Добавляем отступ сверху */
   }
   LI > A {
    display: block; /* Ссылки отображаются в виде блока */
    font-weight: normal; /* Нормальное начертание текста */
    font-size: 90%; /* Размер шрифта */
    background: #fff; /* Цвет фона */
    border: 1px solid #666; /* Параметры рамки */
    border-bottom: none; /* Убираем границу снизу */
    padding: 5px; /* Поля вокруг текста */
   }
  </style>
 </head>
 <body>
  <ul id="menu">
   <li>Правка
    <ul>
      <li><a href="#">Отменить</a></li>
      <li><a href="#">Вырезать</a></li>
      <li><a href="#">Копировать</a></li>
      <li><a href="#">Вставить</a></li>
    </ul>
   </li>
   <li>Начертание
    <ul>
      <li><a href="#">Жирное</a></li>
      <li><a href="#">Курсивное</a></li>
      <li><a href="#">Подчеркнутое</a></li>
    </ul>
   </li>
   <li>Размер
    <ul>
      <li><a href="#">Маленький</a></li>
      <li><a href="#">Нормальный</a></li>
      <li><a href="#">Средний</a></li>
```

```
      <li><a href="#">Большой</a></li>
    </ul>
   </li>
  </ul>
 </body>
</html>
```

В данном примере дочерние селекторы требуются, чтобы разделить стиль элементов списка верхнего уровня и вложенные списки, которые выполняют разные задачи, поэтому стиль для них не должен пересекаться.

Селекторы атрибутов

Многие теги различаются по своему действию в зависимости от того, какие в них используются атрибуты. Например, тег **<input>** может создавать кнопку, текстовое поле и другие элементы формы всего лишь за счет изменения значение атрибута type. При этом добавление правил стиля к селектору **INPUT** применит стиль одновременно ко всем созданным с помощью этого тега элементам. Чтобы гибко управлять стилем подобных элементов, в CSS введены селекторы атрибутов. Они позволяют установить стиль по присутствию определенного атрибута тега или его значения.

Рассмотрим несколько типичных вариантов применения таких селекторов.

Простой селектор атрибута

Устанавливает стиль для элемента, если задан специфичный атрибут тега. Его значение в данном случае не важно. Синтаксис применения такого селектора следующий.

```
[атрибут] { Описание правил стиля }
Селектор[атрибут] { Описание правил стиля }
```

Стиль применяется к тем тегам, внутри которых добавлен указанный атрибут. Пробел между именем селектора и квадратными скобками не допускается.

В примере 1.43 показано изменение стиля тега **<q>**, в том случае, если к нему добавлен атрибут title.

Пример 1.43. Вид элемента в зависимости от его атрибу XHTML 1.0 CSS 2.1 IE 6 IE 7 IE 8 IE 9 Cr 8 Op 11 Sa 5 Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Селекторы атрибутов</title>
  <style type="text/css">
   Q {
    font-style: italic; /* Курсивное начертание */
    quotes: "«" "»"; /* Меняем вид кавычек в цитате */
   }
   Q[title] {
    color: maroon; /* Цвет текста */
   }
  </style>
 </head>
 <body>
  <p>Продолжая известный закон Мерфи, который гласит: <q>Если неприятность
   может случиться, то она обязательно случится</q>, можем ввести свое
   наблюдение:
  <q title="Из законов Фергюссона-Мержевича">После того, как веб-страница
   будет корректно отображаться в одном браузере, выяснится,
   что она неправильно показывается в другом</q>.</p>
 </body>
</html>
```

Результат примера показан на рис. 1.27. Браузер Safari и Chrome не выводят наши кавычки в тексте, заменяя их на стандартные.

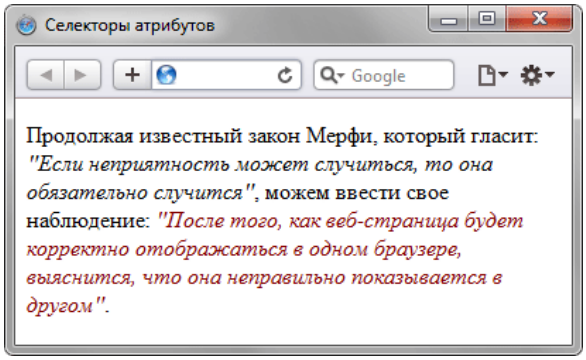


Рис. 1.27. Изменение стиля элемента в зависимости от применения атрибута title

В данном примере меняется цвет текста внутри контейнера **<q>**, когда к нему добавляется title. Обратите внимание, что для селектора **Q[title]** нет нужды повторять стилевые свойства, поскольку они наследуются от селектора **Q**.

Атрибут со значением

Устанавливает стиль для элемента в том случае, если задано определенное значение специфичного атрибута. Синтаксис применения следующий.

```
[атрибут="значение"] { Описание правил стиля }
Селектор[атрибут="значение"] { Описание правил стиля }
```

В первом случае стиль применяется ко всем тегам, которые содержат указанное значение. А во втором — только к определенным селекторам.

В примере 1.44 показано изменение стиля ссылки в том случае, если тег **<a>** содержит атрибут target со значением _blank. При этом ссылка будет открываться в новом окне и чтобы показать это, с помощью стилей добавляем небольшой рисунок перед текстом ссылки.

Пример 1.44. Стиль для открытия ссылок в новом окне XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Селекторы атрибутов</title>
  <style type="text/css">
   A[target="_blank"] {
    background: url(images/blank.png) 0 6px no-repeat;
                                  /* Параметры фонового рисунка */
    padding-left: 15px; /* Смещаем текст вправо */
   }
  </style>
 </head>
 <body>
   <p><a href="1.html">Обычная ссылка</a> |
   <a href="link2" target="_blank">Ссылка в новом окне</a></p>
 </body>
</html>
```

Результат примера показан ниже (рис. 1.28).

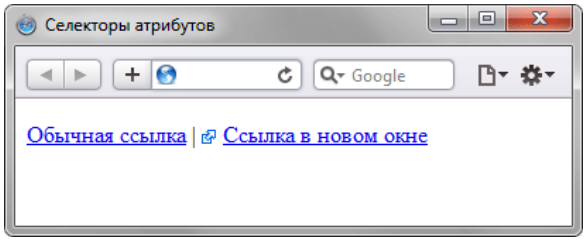


Рис. 1.28. Изменение стиля элемента в зависимости от значения target

В данном примере рисунок к ссылке добавляется с помощью свойства background. В его функции входит создание повторяющейся фоновой картинки, но повторение фона можно отменить через значение no-repeat, что в итоге даст единственное изображение.

Значение атрибута начинается с определенного текста

Устанавливает стиль для элемента в том случае, если значение атрибута тега начинается с указанного текста. Синтаксис применения следующий.

```
[атрибут^="значение"] { Описание правил стиля }
Селектор[атрибут^="значение"] { Описание правил стиля }
```

В первом случае стиль применяется ко всем элементам, у которых значение атрибута начинаются с указанного текста. А во втором — только к определенным селекторам. Использование кавычек не обязательно.

Предположим, что на сайте требуется разделить стиль обычных и внешних ссылок — ссылки, которые ведут на другие сайты. Чтобы не добавлять к тегу **<a>** новый класс, воспользуемся селекторами атрибутов. Внешние ссылки характеризуются добавлением к адресу протокола, например, для доступа к гипертекстовым документам используется протокол HTTP. Поэтому внешние ссылки начинаются с ключевого слова http://, его и добавляем к селектору **A**, как показано в примере 1.45.

Пример 1.45. Изменение стиля внешней ссылки

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Селекторы атрибутов</title>
  <style type="text/css">
   A[href^="http://"] {
    font-weight: bold /* Жирное начертание */
   }
  </style>
 </head>
 <body>

  <p><a href="1.html">Обычная ссылка</a> |
  <a href="http://htmlbook.ru" target="_blank">Внешняя
  ссылка на сайт htmlbook.ru</a></p>

 </body>
</html>
```

Результат примера показан ниже (рис. 1.29).

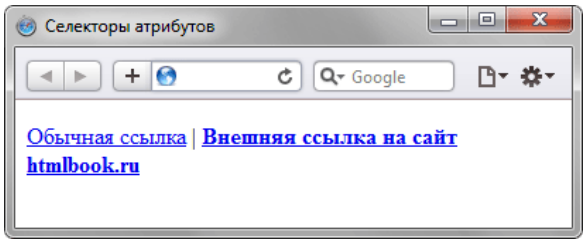


Рис. 1.29. Изменение стиля для внешних ссылок

В данном примере внешние ссылки выделяются жирным начертанием. Также можно воспользоваться показанным в примере 1.44 приемом и добавлять к ссылке небольшое изображение, которое будет сообщать, что ссылка ведет на другой сайт.

Значение атрибута оканчивается определенным текстом

Устанавливает стиль для элемента в том случае, если значение атрибута оканчивается указанным текстом. Синтаксис применения следующий.

```
[атрибут$="значение"] { Описание правил стиля }
Селектор[атрибут$="значение"] { Описание правил стиля }
```

В первом случае стиль применяется ко всем элементам у которых значение атрибута завершается заданным текстом. А во втором — только к определенным селекторам.

Таким способом можно автоматически разделять стиль для сайтов домена ru и для сайтов других доменов вроде com, как показано в примере 1.46.

Пример 1.46. Стиль для разных доменов

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Селекторы атрибутов</title>
  <style type="text/css">
   A[href$=".ru"] { /* Если ссылка заканчивается на .ru */
    background: url(images/ru.png) no-repeat 0 6px;
                /* Добавляем фоновый рисунок */
    padding-left: 12px; /* Смещаем текст вправо */
   }
   A[href$=".com"] { /* Если ссылка заканчивается на .com */
    background: url(images/com.png) no-repeat 0 6px;
    padding-left: 12px;
   }
  </style>
 </head>
 <body>

  <p><a href="http://www.yandex.com">Yandex.Com</a> |
     <a href="http://www.yandex.ru">Yandex.Ru</a></p>

 </body>
</html>
```

В данном примере содержатся две ссылки, ведущие на разные домены — com и ru. При этом к каждой такой ссылке с помощью стилей добавляется своя фоновая картинка (рис. 1.30). Стилевые свойства будут применяться только для тех ссылок, атрибут href которых оканчивается на «.ru» или «.com». Заметьте, что добавив к имени домена слэш (http://www.yandex.ru/) или адрес страницы (http://www.yandex.ru/fun.html), мы изменим тем самым окончание и стиль применяться уже не будет. В этом случае лучше воспользоваться командой *=.

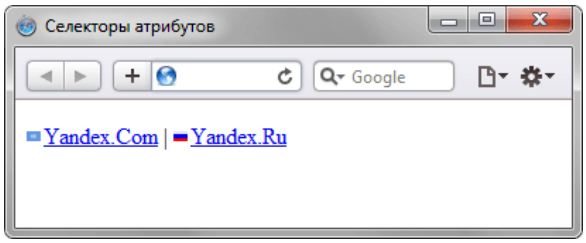


Рис. 1.30. Добавление картинки к ссылкам

Значение атрибута содержит указанный текст

Возможны варианты, когда стиль следует применить к тегу с определенным атрибутом, при этом частью его значения является некоторый текст. При этом точно не известно, в каком месте значения включен данный текст — в начале, середине или конце. В подобном случае следует использовать такой синтаксис.

```
[атрибут*="значение"] { Описание правил стиля }
Селектор[атрибут*="значение"] { Описание правил стиля }
```

В примере 1.47 показано изменение стиля ссылок, в атрибуте href которых встречается слово «htmlbook».

Пример 1.47. Стиль для разных сайтов

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Селекторы атрибутов</title>
  <style type="text/css">
   [href*="htmlbook"] {
    background: yellow; /* Желтый цвет фона */
   }
  </style>
 </head>
 <body>
  <p><a href="http://www.htmlbook.ru/html/">Теги HTML</a> |
  <a href="http://stepbystep.htmlbook.ru">Шаг за шагом</a> |
  <a href="http://webimg.ru">Графика для Веб</a></p>
 </body>
</html>
```

Результат данного примера показан на рис. 1.31.

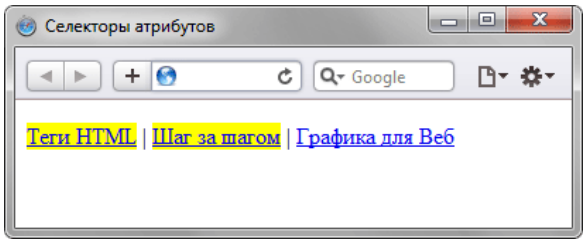


Рис. 1.31. Изменение стиля для ссылок, в адресе которых встречается «htmlbook»

Одно из нескольких значений атрибута

Некоторые значения атрибутов могут перечисляться через пробел, например имена классов. Чтобы задать стиль при наличии в списке требуемого значения применяется следующий синтаксис.

```
[атрибут~="значение"] { Описание правил стиля }
Селектор[атрибут~="значение"] { Описание правил стиля }
```

Стиль применяется в том случае, если у атрибута имеется указанное значение или оно входит в список значений, разделяемых пробелом (пример 1.48).

Пример 1.48. Стиль в зависимости от имени класса

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Блок</title>
  <style type="text/css">
   [class~="block"] h3 { color: green; }
  </style>
 </head>
 <body>
  <div class="block tag">
   <h3>Заголовок</h3>
  </div>
 </body>
</html>
```

В данном примере зеленый цвет текста применяется к селектору **H3**, если имя класса у слоя задано как `block`. Отметим, что аналогичный результат можно получить, если использовать конструкцию `*=` вместо `~=`.

Дефис в значении атрибута

В именах идентификаторов и классов разрешено использовать символ дефиса (-), что позволяет

создавать значащие значения атрибутов id и class. Для изменения стиля элементов, в значении которых применяется дефис, следует воспользоваться следующим синтаксисом.

```
[атрибут|="значение"] { Описание правил стиля }
Селектор[атрибут|="значение"] { Описание правил стиля }
```

Стиль применяется к элементам, у которых атрибут начинается с указанного значения или с фрагмента значения, после которого идет дефис (пример 1.49).

Пример 1.49. Дефисы в значениях

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Блок</title>
  <style type="text/css">
   DIV[class|="block"] {
    background: #306589; /* Цвет фона */
    color: #acdb4c; /* Цвет текста */
    padding: 5px; /* Поля */
   }
   DIV[class|="block"] A {
    color: #fff; /* Цвет ссылок */
   }
  </style>
 </head>
 <body>
  <div class="block-menu-therm">
   <h2>Термины</h2>
   <div class="content">
     <ul class="menu">
      <li><a href="t1.html">Буквица</a></li>
      <li><a href="t2.html">Выворотка</a></li>
      <li><a href="t3.html">Выключка</a></li>
      <li><a href="t4.html">Интерлиньяж</a></li>
      <li><a href="t5.html">Капитель</a></li>
      <li><a href="t6.html">Начертание</a></li>
      <li><a href="t7.html">Отбивка</a></li>
     </ul>
   </div>
  </div>
 </body>
</html>
```

В данном примере имя класса задано как `block-menu-therm`, поэтому в стилях используется конструкция |="block", поскольку значение начинается именно с этого слова и в значении встречаются дефисы.

Все перечисленные методы можно комбинировать между собой, определяя стиль для элементов, которые содержат два и более атрибута. В подобных случаях квадратные скобки идут подряд.

```
[атрибут1="значение1"][атрибут2="значение2"] { Описание правил стиля }
Селектор[атрибут1="значение1"][атрибут2="значение2"] { Описание правил стиля }
```

Универсальный селектор

Иногда требуется установить одновременно один стиль для всех элементов веб-страницы, например, задать шрифт или начертание текста. В этом случае поможет универсальный селектор, который соответствует любому элементу веб-страницы.

Для обозначения универсального селектора применяется символ звездочки (*) и в общем случае синтаксис будет следующий.

```
* { Описание правил стиля }
```

В некоторых случаях указывать универсальный селектор не обязательно. Так, например, записи *.class и .class являются идентичными по своему результату.

В примере 1.50 показано одно из возможных приложений универсального селектора — выбор шрифта и размера текста для всех элементов документа.

Пример 1.50. Использование универсального селектора

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Универсальный селектор</title>
  <style type="text/css">
   * {
    font-family: Arial, Verdana, sans-serif; /* Рубленый шрифт для текста */
    font-size: 96%; /* Размер текста */
   }
  </style>
 </head>
 <body>
  <p>Средний самец льва имеет длину около трех метров и весит
     от 180 до 230 килограмм.</p>
 </body>
</html>
```

Заметим, что аналогичный результат можно получить, если в данном примере поменять селектор * на BODY.

Универсальный селектор часто применяется для обнуления отступов и полей у всех элементов, как показано в примере 1.51.

Пример 1. 51. Стиль для обнуления отступов и полей

```
* {
 margin: 0;
 padding: 0;
}
```

Несмотря на внешние удобства универсального селектора и его широкое распространение, ни в коем случае не идите на поводу «моды» и не используйте селектор, как показано в примере выше. Доводы следующие.

- Универсальный селектор применяет стиль ко всем элементам веб-страницы, включая невидимые, что приводит к замедлению браузера, поскольку ему требуется некоторое время для построения дерева элементов и добавления к ним стилей. Чем больше элементов в коде, тем сильнее выражено замедление. В некоторых крайних случаях вообще может появиться «зависание» браузера на несколько секунд.
- При неверном использовании универсального селектора результат может оказаться непредсказуемым. Пример ниже является полностью корректным с точки зрения CSS, но приводит страницу к парадоксальному виду.

```
* {
 display: block;
 border: 1px solid #c00;
}
```

- «Обнуление стилей» (см. пример 1.51) прививает у разработчика дурную манеру вёрстки. Вместо того чтобы знать, какие значения свойств установлены по умолчанию, разработчик перекладывает эту работу на браузер, насильно устанавливая все значения в ноль. В результате некоторые значения применяются к свойствам, для которых не могут устанавливаться или к свойствам, у которых данное значение и так нулевое. Это опять же приводит к повышению нагрузки на браузер и замедлению его работы.
- Применение одного стиля сразу ко всем элементам иногда приводит к ошибкам отображения элементов в отдельных браузерах. В примере ниже к ссылкам добавляется пунктирное подчеркивание, которое не показывается в IE7 из-за заданного обнуления полей у ссылок.

```
* {
 padding: 0;
}
a {
 text-decoration: none;
 border-bottom: 1px dashed red;
}
```

Несмотря на указанные особенности, универсальный селектор можно и нужно включать в стиль, но в комбинации с другими селекторами. Так, если требуется изменить стиль всех элементов в форме, то следует использовать контекстный селектор **`FORM *`** или селектор, показанный в примере 1.52.

Пример 1.52. Использование универсального селектора

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Универсальный селектор</title>
  <style type="text/css">
   FORM P * {
    border: 1px solid #333; /* Параметры рамки */
    background: #ffe; /* Цвет фона */
   }
  </style>
 </head>
 <body>
  <form action="handler.php">
   <p><input type="text" /></p>
   <p><input type="submit" /></p>
  </form>
 </body>
</html>
```

В данном примере рамка и фон добавляется ко всем элементам формы, расположенных внутри абзаца (тег **`<p>`**).

Псевдоклассы

Псевдоклассы определяют динамическое состояние элементов, которое изменяется со временем или с помощью действий пользователя, а также положение в дереве документа. Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на нее курсора мыши. При использовании псевдоклассов браузер не перегружает текущий документ, поэтому с помощью псевдоклассов можно получить разные динамические эффекты на странице.

Синтаксис применения псевдоклассов следующий.

```
Селектор:Псевдокласс { Описание правил стиля }
```

Вначале указывается селектор, к которому добавляется псевдокласс, затем следует двоеточие, после которого идет имя псевдокласса. Допускается применять псевдоклассы к именам идентификаторов или классов (A.menu:hover {color: green}), а также к контекстным селекторам (.menu A:hover {background: #fc0}). Если псевдокласс указывается без селектора впереди (:hover), то он будет применяться ко всем элементам документа.

Условно все псевдоклассы делятся на три группы:

- определяющие состояние элементов;
- имеющие отношение к дереву элементов;
- указывающие язык текста.

В дальнейшем имена псевдоклассов будем писать с двоеточием впереди, чтобы различать между собой псевдоклассы и стилевые свойства.

Псевдоклассы, определяющие состояние элементов

К этой группе относятся псевдоклассы, которые распознают текущее состояние элемента и применяют стиль только для этого состояния.

:active

Происходит при активации пользователем элемента. Например, ссылка становится активной, если навести на нее курсор и щелкнуть мышкой. Несмотря на то, что активным может стать практически любой элемент веб-страницы, псевдокласс :active используется преимущественно для ссылок.

:link

Применяется к непосещенным ссылкам, т.е. таким ссылкам, на которые пользователь еще не нажимал. Браузер некоторое время сохраняет историю посещений, поэтому ссылка может быть помечена как посещенная хотя бы потому, что по ней был зафиксирован переход раньше.

Запись `A {...}` и `A:link {...}` по своему результату равноценна, поскольку в браузере дает один и тот же эффект, поэтому псевдокласс :link можно не указывать. Исключением являются якоря, на них действие `:link` не распространяется.

:focus

Применяется к элементу при получении им фокуса. Например, для текстового поля формы получение фокуса означает, что курсор установлен в поле, и с помощью клавиатуры можно вводить в него текст (пример 1.53).

Пример 1.53. Применение псевдокласса focus

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Псевдоклассы</title>
  <style type="text/css">
   INPUT:focus {
    color: red; /* Красный цвет текста */
   }
  </style>
 </head>
 <body>
  <form action="">
   <p><input type="text" value="Черный текст" /></p>
   <p><input type="text" value="Черный текст" /></p>
  </form>
 </body>
</html>
```

Результат примера показан ниже (рис. 1.32). Во второй строке находится курсор, поэтому текстовое поле получило фокус.

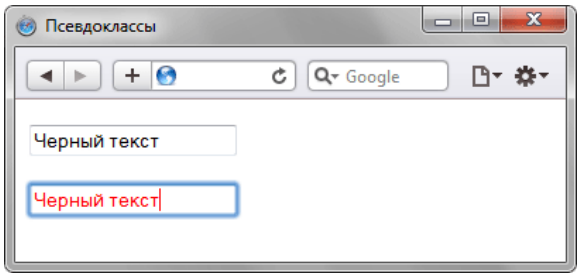


Рис. 1.32. Изменение стиля текста при получении фокуса

В данном примере в текстовом поле содержится предварительный текст, он определяется значением атрибута value тега **<input>**. При щелчке по элементу формы происходит получение полем фокуса, и цвет текста меняется на красный. Достаточно щелкнуть в любом месте страницы (кроме текстового поля, естественно), как произойдет потеря фокуса и текст вернется к первоначальному черному цвету.

Результат будет виден только для тех элементов, которые могут получить фокус. В частности, это теги **<a>**, **<input>**, **<select>** и **<textarea>**.

:hover

Псевдокласс :hover активизируется, когда курсор мыши находится в пределах элемента, но щелчка по нему не происходит.

:visited

Данный псевдокласс применяется к посещенным ссылкам. Обычно такая ссылка меняет свой цвет по умолчанию на фиолетовый, но с помощью стилей цвет и другие параметры можно задать самостоятельно (пример 1.54).

Пример 1.54. Изменение цвета ссылок

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Псевдоклассы</title>
  <style type="text/css">
   A:link {
    color: #036; /* Цвет непосещенных ссылок */
   }
   A:visited {
    color: #606; /* Цвет посещенных ссылок */
   }
```

```
    A:hover {
     color: #f00; /* Цвет ссылок при наведении на них курсора мыши */
    }
    A:active {
     color: #ff0; /* Цвет активных ссылок */
    }
   </style>
  </head>
  <body>
   <p><a href="1.html">Ссылка 1</a> |
    <a href="2.html">Ссылка 2</a> |
    <a href="3.html">Ссылка 3</a></p>
  </body>
</html>
```

Результат примера показан на рис. 1.33.

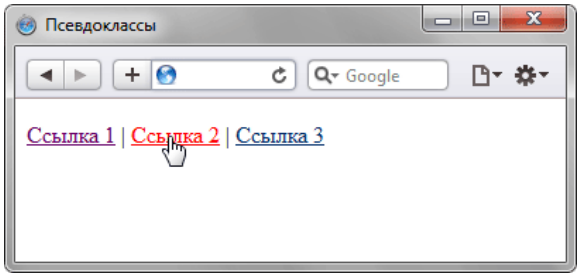


Рис. 1.33. Вид ссылки при наведении на нее курсора мыши

В данном примере показано использование псевдоклассов совместно со ссылками. При этом имеет значение порядок следования псевдоклассов. Вначале указывается :visited, а затем идет :hover, в противном случае посещенные ссылки не будут изменять свой цвет при наведении на них курсора.

Селекторы могут содержать более одного псевдокласса, которые перечисляются подряд через двоеточие, но только в том случае, если их действия не противоречат друг другу. Так, запись **A:visited:hover** является корректной, а запись **A:link:visited** — нет. Впрочем, если подходить формально, то валидатор CSS считает правильным любое сочетание псевдоклассов.

Браузер Internet Explorer 6 позволяет использовать псевдоклассы :active и :hover только для ссылок. Начиная с версии 7.0 псевдоклассы в этом браузере работают и для других элементов.

Псевдокласс :hover не обязательно должен применяться к ссылкам, его можно добавлять и к другим элементам документа. Так, в примере 1.55 показана таблица, строки которой меняют свой цвет при наведении на них курсора мыши. Это достигается за счет добавления псевдокласса :hover к селектору **TR**.

Пример 1.55. Выделение строк таблицы

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Псевдоклассы</title>
  <style type="text/css">
   TR:hover {
    background: #fc0; /* Меняем цвет фона строки таблицы */
   }
  </style>
 </head>
 <body>
  <table width="400" border="1" cellpadding="4" cellspacing="0">
   <tr>
    <th> </th>
    <th>Пики</th><th>Трефы</th><th>Бубны</th><th>Червы</th>
   </tr>
   <tr>
    <td>Чебурашка</td>
    <td>5</td><td>2</td><td>4</td><td>2</td>
   </tr>
   <tr>
```

```
    <td>Крокодил Гена</td>
    <td>2</td><td>7</td><td>1</td><td>3</td>
   </tr>
   <tr>
    <td>Шапокляк</td>
    <td>5</td><td>4</td><td>3</td><td>1</td>
   </tr>
   <tr>
    <td>Крыса Лариса</td>
    <td>1</td><td>0</td><td>5</td><td>7</td>
   </tr>
  </table>
 </body>
</html>
```

Результат примера показан ниже (рис. 1.34).

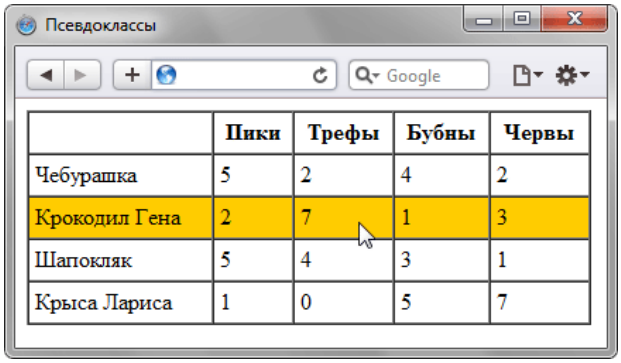


Рис. 1.34. Выделение строк таблицы при наведении на них курсора мыши

Псевдоклассы, имеющие отношение к дереву документа

К этой группе относятся псевдоклассы, которые определяют положение элемента в дереве документа и применяют к нему стиль в зависимости от его статуса.

:first-child

Применяется к первому дочернему элементу селектора, который расположен в дереве элементов документа. Чтобы стало понятно, о чем речь, разберем небольшой код (пример 1.56).

Пример 1.56. Использование псевдокласса first-child

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Псевдоклассы</title>
  <style type="text/css">
   div p:first-child {
    font: 1.2em Arial, sans-serif; /* Шрифт */
    color: #E81E26; /* Цвет текста */
    margin-bottom: -0.5em; /* Отступ снизу */
   }
  </style>
 </head>
 <body>
  <div>
   <p>Размеры</p>
   <p>Желательно, чтобы ширина и длина клетки были кратные значениям A и B.
    Таким образом, всю пустыню можно поделить на ряд дискретных областей,
    размеры которых совпадают с размерами нашей клетки.</p>
  </div>
  <div>
   <p>Положение льва</p>
   <p>Во всех случаях положение льва считается стационарным.
    Во время поиска животное остается на своем месте и не перемещается
    до его поимки в клетку.</p>
  </div>
 </body>
</html>
```

Результат примера показан ниже (рис. 1.35).

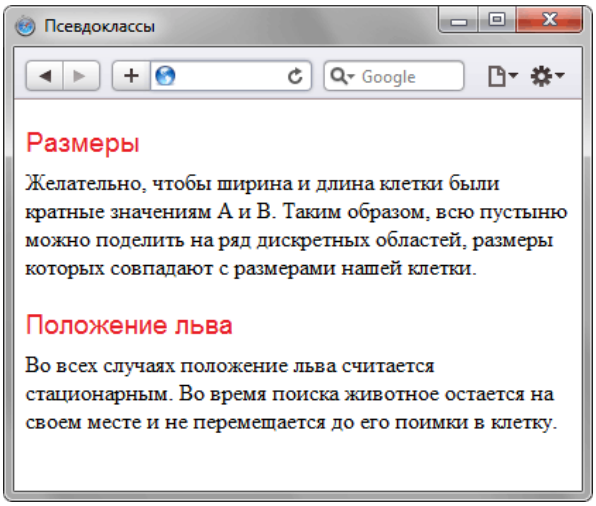


Рис. 1.35. Использование псевдокласса :first-child

В данном примере псевдокласс :first-child добавляется к селектору **P** и устанавливает для него рубленый шрифт и красный цвет текста. Хотя тег **<P>** внутри ** ** встречается по два раза, стиль будет применяться только к первым абзацам. В остальных случаях стиль внутри **<p>** останется исходным. Со следующим тегом ** ** все начинается снова, поскольку родительский элемент поменялся.

Браузер Internet Explorer поддерживает псевдокласс :first-child начиная с версии 7.0.

Псевдокласс :first-child удобнее всего использовать в тех случаях, когда требуется задать разный стиль для первого и остальных однотипных элементов. Например, по правилам типографики красную строку для первого абзаца текста не устанавливают, а для остальных абзацев добавляют отступ первой строки. С этой целью применяют свойство text-indent с нужным значением отступа. Но чтобы изменить стиль первого абзаца и убрать для него отступ потребуется воспользоваться псевдоклассом :first-child (пример 1.57).

Пример 1.57. Отступы для абзаца

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Псевдоклассы</title>
  <style type="text/css">
   P {
    text-indent: 1em; /* Отступ первой строки */
   }
   P:first-child {
    text-indent: 0; /* Для первого абзаца отступ убираем */
   }
  </style>
 </head>
 <body>
  <p>Историю эту уже начали забывать, хотя находились горожане, которые
  время от времени рассказывали ее вновь прибывшим в город посетителям.</p>
  <p>Легенда обрастала подробностями и уже совсем не напоминала произошедшее
  в действительности событие. И, тем не менее, ни один человек не решался
  заикнуться о ней с наступлением темноты.</p>
  <p>Но однажды в город вновь вошел незнакомец. Он хромал на левую ногу.</p>
 </body>
</html>
```

Результат примера показан ниже (рис. 1.36).

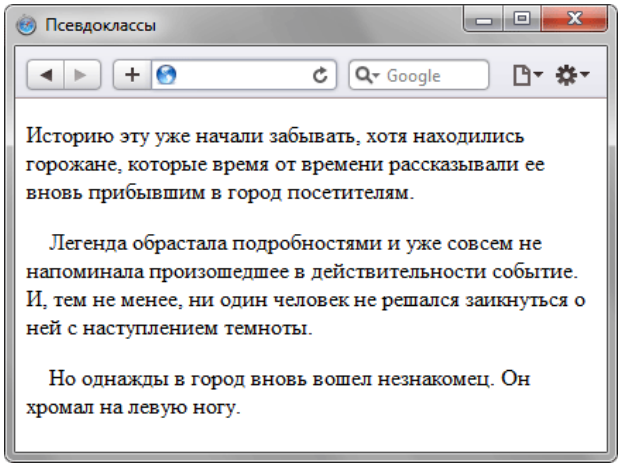


Рис. 1.36. Изменение стиля первого абзаца

В данном примере первый абзац текста не содержит отступа первой строки, а для остальных он установлен.

Псевдоклассы, задающие язык текста

Для документов, одновременно содержащие тексты на нескольких языках имеет значение соблюдение правил синтаксиса, характерные для того или иного языка. С помощью псевдоклассов можно изменять стиль оформления иностранных текстов, а также некоторые настройки.

:lang

Определяет язык, который используется в документе или его фрагменте. В коде HTML язык устанавливается через параметр charset тега **<meta>**. С помощью псевдокласса :lang можно задавать определенные настройки, характерные для разных языков, например, вид кавычек в цитатах. Синтаксис следующий.

```
Элемент:lang(язык) { ... }
```

В качестве языка могут выступать следующие значения: ru — русский; en — английский ; de — немецкий ; fr — французский; it — итальянский.

Пример 1.58. Вид кавычек в зависимости от языка

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Псевдоклассы</title>
  <style type="text/css">
   P {
    font-size: 1.50em; /* Размер текста */
   }
   q:lang(de) {
    quotes: "\201E" "\201C"; /* Вид кавычек для немецкого языка */
   }
   q:lang(en) {
    quotes: "\201C" "\201D"; /* Вид кавычек для английского языка */
   }
   q:lang(fr), q:lang(ru) { /* Кавычки для русского и французского языка */
    quotes: "\00AB" "\00BB";
   }
  </style>
 </head>
 <body>
  <p>Цитата на французском языке: <q lang="fr">Ce que femme veut, Dieu
     le veut</q>.</p>
  <p>Цитата на немецком: <q lang="de">Der Mensch, versuche die
     Gotter nicht</q>.</p>
  <p>Цитата на английском: <q lang="en">To be or not to be</q>.</p>
 </body>
</html>
```

Результат данного примера показан на рис. 1.37. Для отображения типовых кавычек в примере используется стилевое свойство quotes, а само переключение языка и соответствующего вида кавычек происходит через атрибут lang, добавляемый к тегу **<q>**.

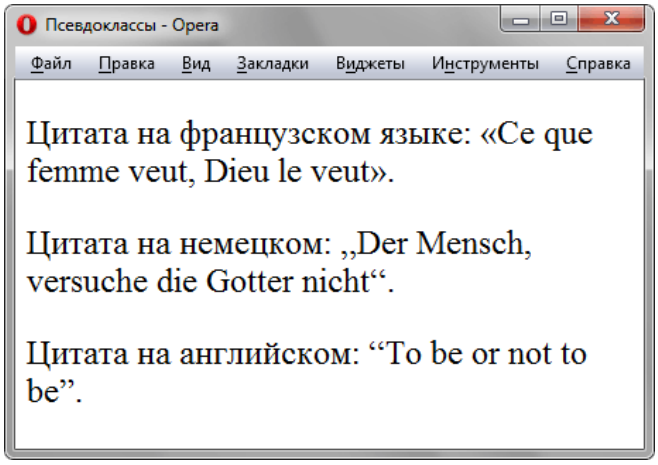


Рис. 1.37. Разные кавычки для разных языков

Псевдоэлементы

Псевдоэлементы позволяют задать стиль элементов не определенных в дереве элементов документа, а также генерировать содержимое, которого нет в исходном коде текста.

Синтаксис использования псевдоэлементов следующий.

```
Селектор:Псевдоэлемент { Описание правил стиля }
```

Вначале следует имя селектора, затем пишется двоеточие, после которого идет имя псевдоэлемента. Каждый псевдоэлемент может применяться только к одному селектору, если требуется установить сразу несколько псевдоэлементов для одного селектора, правила стиля должны добавляться к ним по отдельности, как показано ниже.

```
.foo:first-letter { color: red }
.foo:first-line {font-style: italic}
```

В CSS3 чтобы различать псевдоклассы и псевдоэлементы, перед именем псевдоэлемента ставится два двоеточия (::after). Internet Explorer игнорирует подобную запись, остальные браузеры корректно её понимают.

Псевдоэлементы не могут применяться к внутренним стилям, только к таблице связанных или глобальных стилей.

Далее перечислены псевдоэлементы, их описание и свойства.

:after

Применяется для вставки назначенного контента после элемента. Этот псевдоэлемент работает совместно со стилевым свойством content, которое определяет содержимое для вставки. В примере 1.59 показано использование псевдокласса :after для добавления текста в конец абзаца.

Пример 1.59. Применение псевдоэлемента :after

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Псевдоэлементы</title>
  <style type="text/css">
   P.new:after {
    content: " - Новьё!"; /* Добавляем после текста абзаца */
   }
  </style>
 </head>
 <body>
  <p class="new">Ловля льва в пустыне с помощью метода золотого сечения.</p>
  <p>Метод ловли льва простым перебором.</p>
 </body>
</html>
```

Результат примера показан на рис. 1.38.

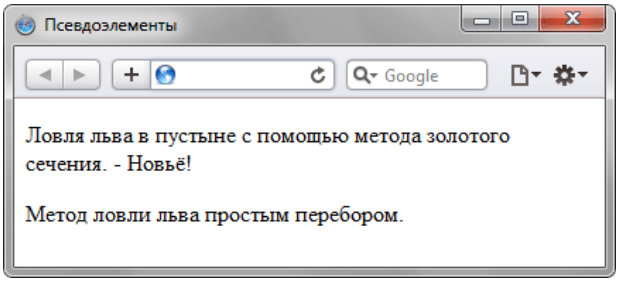


Рис. 1.38. Добавление текста к абзацу с помощью :after

В данном примере к содержимому абзаца с классом new добавляется дополнительное слово, которое выступает значением свойства content.

Псевдоэлементы :after и :before, а также стилевое свойство content не поддерживаются браузером Internet Explorer до седьмой версии включительно.

:before

По своему действию :before аналогичен псевдоэлементу :after, но вставляет контент до элемента. В примере 1.60 показано добавление маркеров своего типа к элементам списка посредством скрытия стандартных маркеров и применения псевдокласса :before.

Пример 1.60. Использование псевдоэлемента :before

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Псевдоэлементы</title>
  <style type="text/css">
   UL {
    padding-left: 0; /* Убираем отступ слева */
    list-style-type: none; /* Прячем маркеры списка */
   }
   LI:before {
    content: "◊ "; /* Добавляем перед элементом списка символ */
   }
  </style>
 </head>
 <body>
  <ul>
   <li>Метод простых итераций</li>
   <li>Метод случайных чисел</li>
   <li>Метод дихотомии</li>
   <li>Метод золотого сечения</li>
  </ul>
 </body>
</html>
```

Результат примера показан ниже (рис. 1.39).

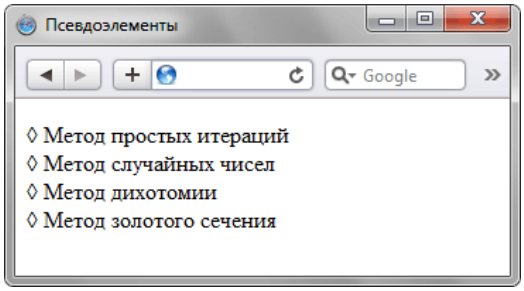


Рис. 1.39. Изменение вида маркеров с помощью :before

В данном примере псевдокласс :before устанавливается для селектора LI, определяющего элементы списка. Добавление желаемых символов происходит путем задания значения свойства content.

Обратите внимание, что в качестве аргумента не обязательно выступает текст, могут применяться также символы юникода.

:first-letter

Определяет стиль первого символа в тексте элемента, к которому добавляется. Это позволяет создавать в тексте буквицу и выступающий инициал.

Буквица представляет собой увеличенную первую букву, базовая линия которой ниже на одну или несколько строк базовой линии основного текста. Выступающий инициал — увеличенная прописная буква, базовая линия которой совпадает с базовой линией основного текста.

Рассмотрим пример создания выступающего инициала. Для этого требуется добавить к селектору **P** псевдоэлемент :first-letter и установить желаемый стиль инициала. В частности, увеличить размер текста и поменять цвет текста (пример 1.61).

Пример 1.61. Использование :first-letter

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Псевдоэлементы</title>
  <style type="text/css">
   P {
    font-family: Arial, Helvetica, sans-serif;
                        /* Гарнитура шрифта основного текста */
    font-size: 0.9em; /* Размер шрифта */
    color: black; /* Черный цвет текста */
   }
   P:first-letter {
    font-family: 'Times New Roman', Times, serif;
                          /* Гарнитура шрифта первой буквы */
    font-size: 2em; /* Размер шрифта первого символа */
    color: red; /* Красный цвет текста */
   }
  </style>
 </head>
 <body>
   <p>Метод случайных чисел является вариантом метода простых итераций,
    в котором происходит перебор участков пустыни для поиска льва.
    Разница только в том, что области выбираются не последовательно,
    а случайно. Таким образом, в лучшем случае поиск может закончиться
    сразу же, а в худшем — превратиться в метод перебора.</p>
   <p>Поскольку номер области выбирается случайно, он может выпасть
    больше одного раза. Однако переходить второй раз в уже просмотренную
    область, необходимости нет.</p>
 </body>
</html>
```

Результат примера показан ниже (рис. 1.40).

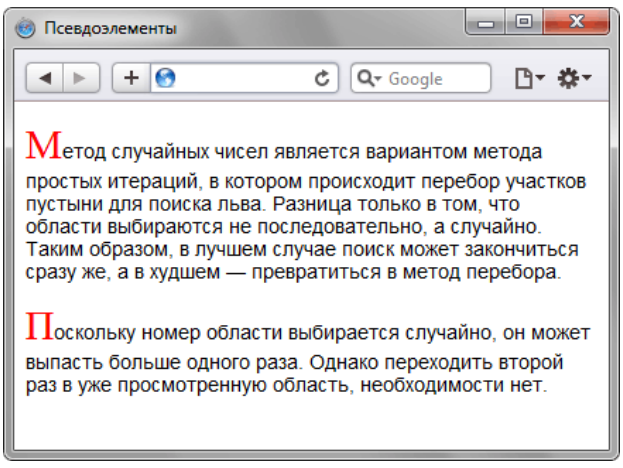


Рис. 1.40. Создание выступающего инициала

В данном примере изменяется шрифт, размер и цвет первой буквы каждого абзаца текста.

:first-line

Определяет стиль первой строки блочного текста. Длина этой строки зависит от многих факторов, таких как используемый шрифт, размер окна браузера, ширина блока, языка и т.д.

К псевдоэлементу :first-line могут применяться не все стилевые свойства. Допустимо использовать свойства, относящиеся к шрифту, изменению цвет текста и фона, а также: clear, line-height, letter-spacing, text-decoration, text-transform, vertical-align и word-spacing.

В примере 1.62 показано использование псевдоэлемента :first-line применительно к абзацу текста.

Пример 1.62. Выделение первой строки текста

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Псевдоэлементы</title>
  <style type="text/css">
   P:first-line {
    color: red; /* Красный цвет текста */
    font-style: italic; /* Курсивное начертание */
   }
  </style>
 </head>
 <body>
  <p>Интересно, а существует ли способ действительно практичного применения
  first-line? Нет, не такого, чтобы можно было бы показать, что это возможно,
  а чтобы воистину захватило дух от красоты решения, загорелись глаза от
  скрытых перспектив, после чего остается только сказать себе, что вот это
  вот, это самое сделать по-другому, также изящно и эффектно
  просто невозможно.</p>
 </body>
</html>
```

Результат примера показан на рис. 1.41.

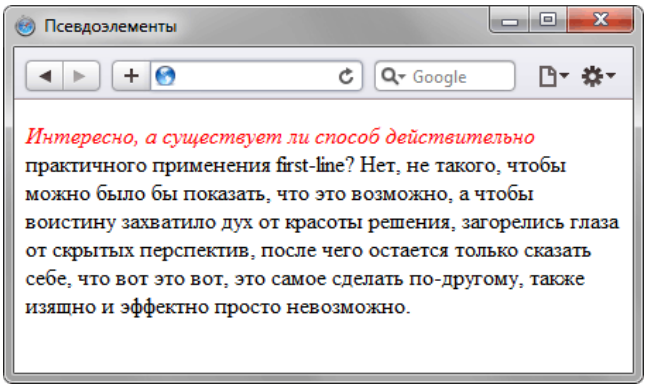


Рис. 1.41. Результат применения псевдоэлемента :first-line

В данном примере первая строка выделяется красным цветом и курсивным начертанием. Обратите внимание, что при изменении ширины окна браузера, стиль первой строки остается постоянным, независимо от числа входящих в нее слов.

Группирование

При создании стиля для сайта, когда одновременно используется множество селекторов, возможно появление повторяющихся стилевых правил. Чтобы не повторять дважды одни и те же элементы, их можно сгруппировать для удобства представления и сокращения кода. В примере 1.63 показана обычная запись, здесь для каждого селектора приводится свой набор стилевых свойств.

Пример 1.63. Стиль для каждого селектора

```
H1 {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 160%;
  color: #003;
}
H2 {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 135%;
  color: #333;
}
H3 {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 120%;
  color: #900;
}
P {
  font-family: Times, serif;
}
```

Из данного примера видно, что стиль для тегов заголовков содержит одинаковое значение font-family. Группирование как раз и позволяет установить одно свойство сразу для нескольких селекторов, как показано в примере 1.64.

Пример 1.64. Сгруппированные селекторы

```
H1, H2, H3 {
  font-family: Arial, Helvetica, sans-serif;
}
H1 {
  font-size: 160%;
  color: #003;
}
H2 {
  font-size: 135%;
  color: #333;
}
H3 {
  font-size: 120%;
  color: #900;
}
```

В данном примере font-family единое для всех селекторов применяется сразу к нескольким тегам, а индивидуальные свойства уже добавляются к каждому селектору отдельно.

Селекторы группируются в виде списка тегов, разделенных между собой запятыми. В группу могут входить не только селекторы тегов, но также идентификаторы и классы. Общий синтаксис следующий.

```
Селектор 1, Селектор 2, ... Селектор N { Описание правил стиля }
```

При такой записи правила стиля применяются ко всем селекторам, перечисленным в одной группе.

Наследование

Наследованием называется перенос правил форматирования для элементов, находящихся внутри других. Такие элементы являются дочерними, и они наследуют некоторые стилевые свойства своих родителей, внутри которых располагаются.

Разберем наследование на примере таблицы. Особенностью таблиц можно считать строгую иерархическую структуру тегов. Вначале следует контейнер **<table>** внутри которого добавляются теги **<tr>**, а затем идет тег **<td>**. Если в стилях для селектора **TABLE** задать цвет текста, то он автоматически устанавливается для содержимого ячеек, как показано в примере 1.65.

Пример 1.65. Наследование параметров цвета

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Наследование</title>
  <style type="text/css">
   TABLE {
    color: red; /* Цвет текста */
    background: #333; /* Цвет фона таблицы */
    border: 2px solid red; /* Красная рамка вокруг таблицы */
   }
  </style>
 </head>
 <body>
  <table cellpadding="4" cellspacing="0">
   <tr>
    <td>Ячейка 1</td><td>Ячейка 2</td>
   </tr>
   <tr>
    <td>Ячейка 3</td><td>Ячейка 4</td>
   </tr>
  </table>
 </body>

</html>
```

В данном примере для всей таблицы установлен красный цвет текста, поэтому в ячейках он также применяется, поскольку тег **<td>** наследует свойства тега **<table>**. При этом следует понимать, что не все стилевые свойства наследуются. Так, border задает рамку вокруг таблицы в целом, но никак не вокруг ячеек. Аналогично не наследуется значение свойства background. Тогда почему цвет фона у ячеек в данном примере темный, раз он не наследуется? Дело в том, что у свойства background в качестве значения по умолчанию выступает transparent, т.е. прозрачность. Таким образом цвет фона родительского элемента «проглядывает» сквозь дочерний элемент.

Чтобы определить, наследуется значение стилевого свойства или нет, требуется заглянуть в справочник по свойствам CSS и посмотреть там. Подключать свою интуицию в подобном случае бесполезно, может и подвести.

Наследование позволяет задавать значения некоторых свойств единожды, определяя их для родителей верхнего уровня. Допустим, требуется установить цвет и шрифт для основного текста. Достаточно воспользоваться селектором **BODY**, добавить для него желаемые свойства, и цвет текста внутри абзацев и других текстовых элементов поменяется автоматически (пример 1.66).

Пример 1.66. Параметры текста для всей веб-страницы

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Наследование</title>
  <style type="text/css">
```

```
    BODY {
     font-family: Arial, Helvetica, sans-serif; /* Гарнитура шрифта */
     color: navy; /* Синий цвет текста */
    }
   </style>
  </head>
  <body>
   <p>Цвет текста этого абзаца синий.</p>
  </body>
 </html>
```

В данном примере рубленый шрифт и цвет текста абзацев устанавливается с помощью селектора **BODY**. Благодаря наследованию уже нет нужды задавать цвет для каждого элемента документа в отдельности. Однако бывают варианты, когда требуется все-таки изменить цвет для отдельного контейнера. В этом случае придется переопределять нужные параметры явно, как показано в примере 1.67.

Пример 1.67. Изменение свойств наследуемого элемента XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Наследование</title>
  <style type="text/css">
   BODY {
    font-family: Arial, Helvetica, sans-serif; /* Гарнитура шрифта */
    color: navy; /* Синий цвет текста */
   }
   P.red {
    color: maroon; /* Темно-красный цвет текста */
   }
  </style>
 </head>
 <body>
  <p>Цвет текста этого абзаца синий.</p>
  <p class="red">А у этого абзаца цвет текста уже другой.</p>
 </body>
</html>
```

В данном примере цвет первого абзаца наследуется от селектора **BODY**, а для второго установлен явно через класс с именем `red`.

Каскадирование

Аббревиатура CSS расшифровывается как Cascading Style Sheets (каскадные таблицы стилей), где одним из ключевых слов выступает «каскад». Под каскадом в данном случае понимается одновременное применение разных стилевых правил к элементам документа — с помощью подключения нескольких стилевых файлов, наследования свойств и других методов. Чтобы в подобной ситуации браузер понимал, какое в итоге правило применять к элементу, и не возникало конфликтов в поведении разных браузеров, введены определенные приоритеты.

Ниже приведены приоритеты браузеров, которыми они руководствуются при обработке стилевых правил. Чем выше в списке находится пункт, тем ниже его приоритет, и наоборот.

1. Стиль браузера.
2. Стиль пользователя.
3. Стиль автора.
4. Стиль автора с добавлением !important.
5. Стиль пользователя с добавлением !important.

Самым низким приоритетом обладает стиль браузера — оформление, которое по умолчанию применяется к элементам веб-страницы браузером. Это оформление можно увидеть в случае «голого» HTML, когда к документу не добавляется никаких стилей.

!important

Ключевое слово !important играет роль в том случае, когда пользователи подключают свою собственную таблицу стилей. Если возникает противоречие, когда стиль автора страницы и пользователя для одного и того же элемента не совпадает, то !important позволяет повысить приоритет стиля или его важность, иными словами.

При использовании пользовательской таблицы стилей или одновременном применении разного стиля автора и пользователя к одному и тому же селектору, браузер руководствуется следующим алгоритмом.

- !important добавлен в авторский стиль — будет применяться стиль автора.
- !important добавлен в пользовательский стиль — будет применяться стиль пользователя.
- !important нет как в авторском стиле, так и стиле пользователя — будет применяться стиль автора.
- !important содержится в авторском стиле и стиле пользователя — будет применяться стиль пользователя.

Синтаксис применения !important следующий.

```
Свойство: значение !important;
```

Вначале пишется желаемое стилевое свойство, затем через двоеточие его значение и в конце после пробела указывается ключевое слово !important.

Повышение важности требуется не только для регулирования приоритета между авторской и пользовательской таблицей стилей, но и для повышения специфичности определенного селектора.

Специфичность

Если к одному элементу одновременно применяются противоречивые стилевые правила, то более высокий приоритет имеет правило, у которого значение специфичности селектора больше.

Специфичность это некоторая условная величина, вычисляемая следующим образом. За каждый идентификатор (в дальнейшем будем обозначать их количество через a) начисляется 100, за каждый класс и псевдокласс (b) начисляется 10, за каждый селектор тега и псевдоэлемент (c) начисляется 1. Суммируя указанные значения, получим значение специфичности для данного селектора.

```
*               {} /* a=0 b=0 c=0 -> специфичность = 0   */
li              {} /* a=0 b=0 c=1 -> специфичность = 1   */
li:first-line   {} /* a=0 b=0 c=2 -> специфичность = 2   */
ul li           {} /* a=0 b=0 c=2 -> специфичность = 2   */
ul ol+li        {} /* a=0 b=0 c=3 -> специфичность = 3   */
ul li.red       {} /* a=0 b=1 c=2 -> специфичность = 12  */
li.red.level    {} /* a=0 b=2 c=1 -> специфичность = 21  */
#t34            {} /* a=1 b=0 c=0 -> специфичность = 100 */
#content #wrap  {} /* a=2 b=0 c=0 -> специфичность = 200 */
```

Встроенный стиль, добавляемый к тегу через атрибут style, имеет специфичность 1000, поэтому всегда перекрывает связанные и глобальные стили. Однако добавление !important перекрывает в том числе и встроенные стили.

Если два селектора имеют одинаковую специфичность, то применяться будет тот стиль, что определен в коде ниже.

В примере 1.68 показано, как влияет специфичность на стиль элементов списка.

Пример 1.68. Цвет списка

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Список</title>
  <style type="text/css">
   #menu ul li {
    color: green;
   }
   .two {
    color: red;
   }
  </style>
 </head>
 <body>
  <div id="menu">
   <ul>
    <li>Первый</li>
    <li class="two">Второй</li>
    <li>Третий</li>
   </ul>
  </div>
 </body>
</html>
```

В данном примере цвет текста списка задан зеленым, а второй пункт списка с помощью класса `two` выделен красным цветом. Вычисляем специфичность селектора `#menu ul li` — один идентификатор (100) и два тега (2) в сумме дают значение 102, а селектор `.two` будет иметь значение специфичности 20, что явно меньше. Поэтому текст окрашиваться красным цветом не будет. Чтобы исправить ситуацию, необходимо либо понизить специфичность первого селектора, либо повысить специфичность второго (пример 1.69).

Пример 1.69. Изменение специфичности

```
/* Понижаем специфичность первого селектора */
 ul li {...} /* Убираем идентификатор */
 .two  {...}

/* Повышаем специфичность второго селектора */
 #menu ul li {...}
 #menu  .two {...} /* Добавляем  идентификатор */

 #menu ul li {...}
 .two { color:  red !important; } /* Добавляем !important */
```

Добавление идентификатора используется не только для изменения специфичности селектора, но и для применения стиля только к указанному списку. Поэтому понижение специфичности за счет убирания идентификатора применяется редко, в основном, повышается специфичность нужного селектора.

Валидация CSS

Валидацией называется проверка CSS-кода на соответствие спецификации CSS2.1 или CSS3. Соответственно, корректный код, не содержащий ошибок, называется валидный, а не удовлетворяющий спецификации — невалидный. Наиболее удобно делать проверку кода через сайт http://jigsaw.w3.org/css-validator/, с помощью этого сервиса можно указать адрес документа, загрузить файл или проверить набранный текст. Большим плюсом сервиса является поддержка русского и украинского языка.

Проверить URI

Эта вкладка позволяет указывать адрес страницы размещенной в Интернете. Протокол http:// можно не писать, он будет добавлен автоматически (рис. 1.42).

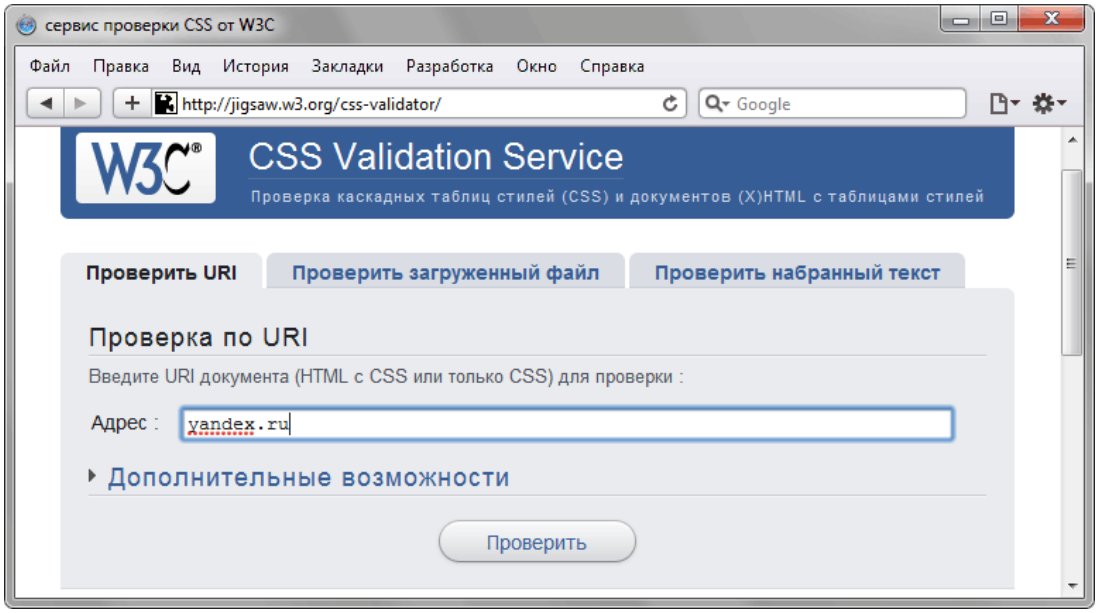


Рис. 1.42. Проверка документа по адресу

После ввода адреса нажмите на кнопку «Проверить» и появится одна из двух надписей: «Поздравляем! Ошибок не обнаружено» в случае успеха или «К сожалению, мы обнаружили следующие ошибки» при невалидном коде. Сообщения об ошибках или предупреждениях содержат номер строки, селектор и описание ошибки.

Проверить загруженный файл

Эта вкладка позволяет загрузить HTML или CSS-файл и проверить его на наличие ошибок (рис. 1.43).

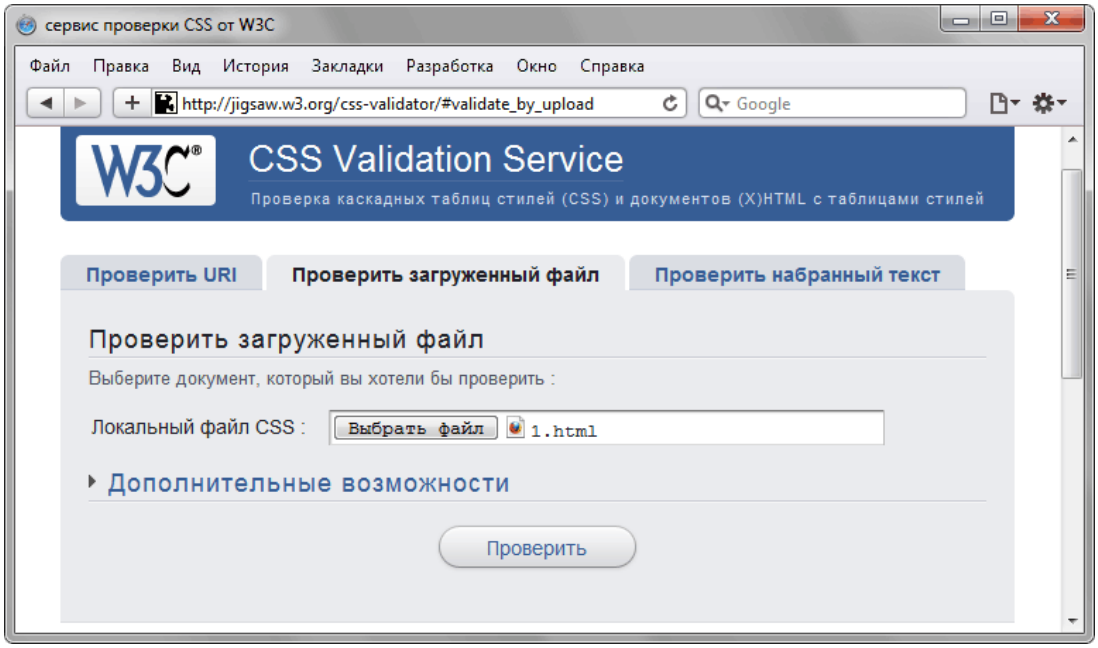


Рис. 1.43. Проверка файла при его загрузке

Сервис автоматически распознает тип файла и если указан HTML-документ, вычленяет из него стиль для валидации.

Проверить набранный текст

Последняя вкладка предназначена для непосредственного ввода HTML или CSS-кода, при этом проверке будет подвергнут только стиль (рис. 1.44).

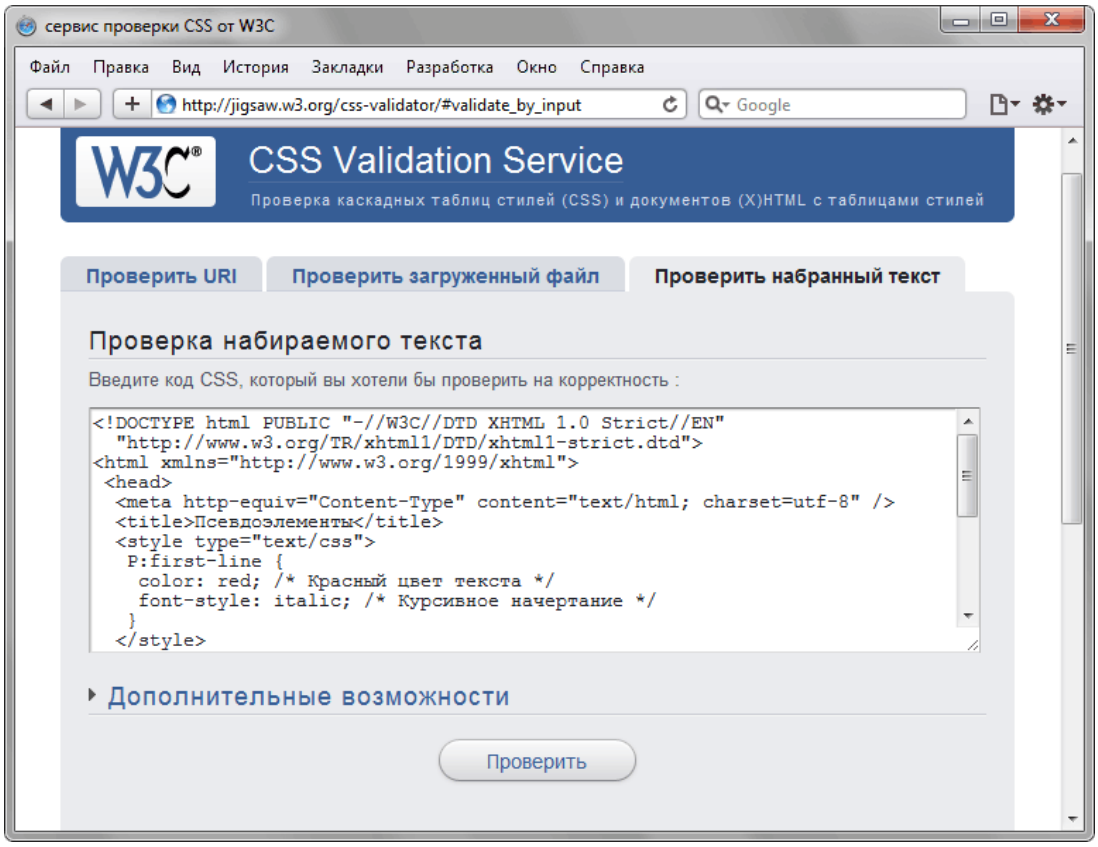


Рис. 1.44. Проверка введенного кода

Этот вариант представляется наиболее удобным для проведения различных экспериментов над кодом или быстрой проверки небольших фрагментов.

Выбор версии CSS

В CSS3 добавлено много новых стилевых свойств по сравнению с предыдущей версией, поэтому проводить проверку кода следует с учетом версии. По умолчанию в сервисе указан CSS2.1, так что если вы хотите проверить код на соответствие CSS3, это следует указать явно. Для этого щелкните по тексту «Дополнительные возможности» и в открывшемся блоке из списка «Профиль» выберите CSS3 (рис. 1.45).

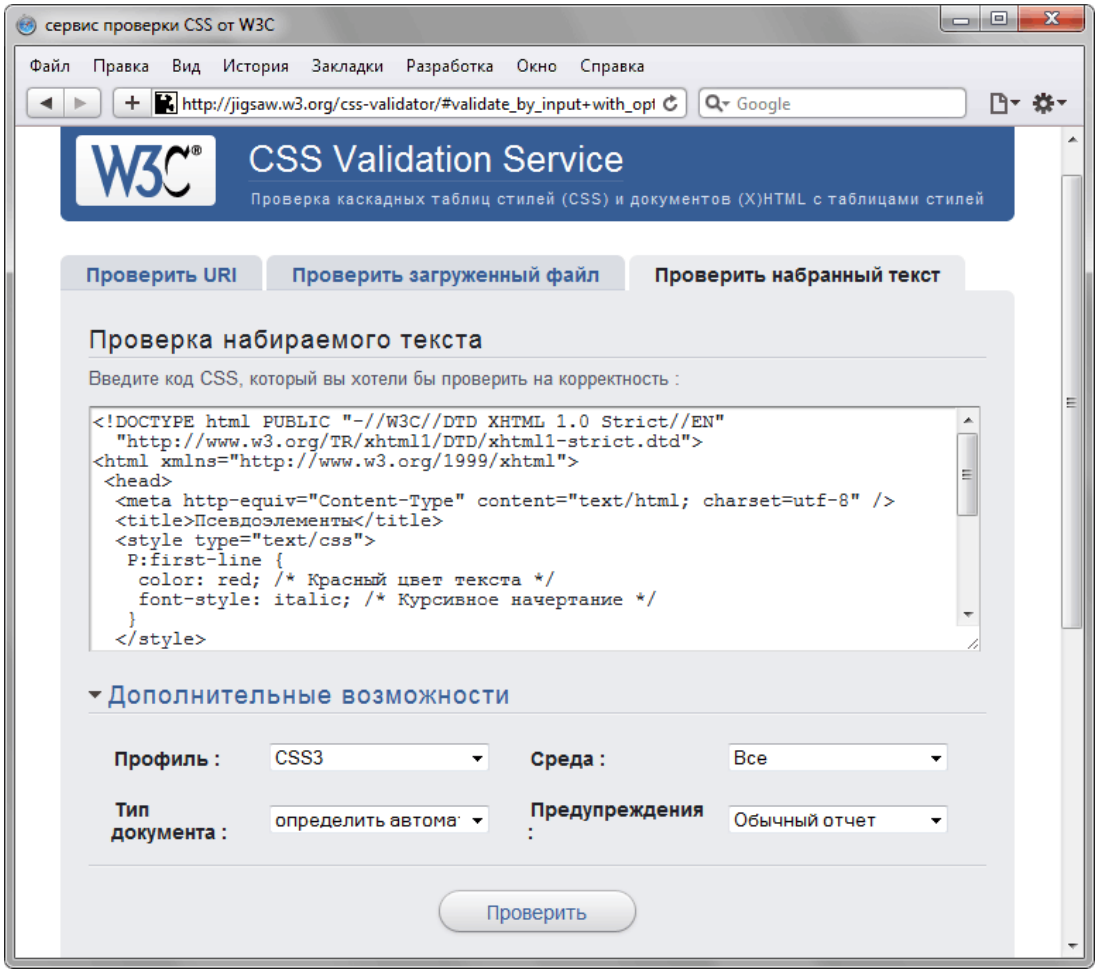


Рис. 1.45. Указание версии CSS для проверки

Идентификаторы и классы

Периодически поднимается спор о целесообразности использования идентификаторов в вёрстке. Основной довод состоит в том, что идентификаторы предназначены для доступа и управления элементами веб-страницы с помощью скриптов, а для изменения стилей элементов должны применяться исключительно классы. В действительности нет разницы, через что задавать стили, но следует помнить о некоторых особенностях идентификаторов и классов, а также подводных камнях, которые могут поджидать разработчиков.

Для начала перечислим характерные признаки этих селекторов.

Идентификаторы

- В коде документа каждый идентификатор уникален и должен быть включён лишь один раз.
- Имя идентификатора чувствительно к регистру.
- Через метод getElementById можно получить доступ к элементу по его идентификатору и изменить свойства элемента.
- Стиль для идентификатора имеет приоритет выше, чем у классов.

Классы

- Классы могут использоваться в коде неоднократно.
- Имена классов чувствительны к регистру.
- Классы можно комбинировать между собой, добавляя несколько классов к одному тегу.

Идентификаторы

Если во время работы веб-страницы требуется изменить стиль некоторых элементов «на лету» или выводить внутри них какой-либо текст, с идентификаторами это делается гораздо проще. Обращение к элементу происходит через метод getElementById, параметром которого служит имя идентификатора. В примере 1.70 к текстовому полю формы добавляется идентификатор с именем `userName`, затем с помощью функции JavaScript делается проверка на то, что пользователь ввёл в это поле какой-либо текст. Если никакого текста нет, но кнопка Submit нажата, выводится сообщение внутри тега с идентификатором `msg`. Если всё правильно, данные формы отправляются по адресу, указанному атрибутом action.

Пример 1.70. Проверка данных формы

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Проверка формы</title>
  <script type="text/javascript">
   function validForm(f) {
    user = document.getElementById("userName");
    if(user.value == "")
     document.getElementById("msg").innerHTML = 'Пожалуйста, введите имя.';
    else f.submit();
   }
  </script>
 </head>
 <body>
  <form action="handler.php" onsubmit="validForm(this); return false">
   <p>Введите свое имя:</p>
   <div id="msg"></div>
   <p><input type="text" id="userName" name="user" /></p>
   <p><input type="submit" /></p>
  </form>
 </body>
```

```
</html>
```

Поскольку идентификаторы чувствительны к регистру, имеет значение их однотипное написание. Внутри тега `<input>` используется имя `userName`, его же следует указать и в методе getElementById. При ошибочном написании, например, `username`, скрипт перестанет работать, как требуется.

В примере выше стили вообще никакого участия не принимали, сами идентификаторы требовались для работы скриптов. При использовании в CSS следует учитывать, что идентификаторы обладают высоким приоритетом по сравнению с классами. Поясним это на примере 1.71.

Пример 1.71. Сочетание стилей

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Идентификаторы</title>
  <style type="text/css">
   #A, .a {
    border: none;
    background: #f0f0f0;
    color: green;
    padding: 5px;
   }
   .b {
    border: 1px solid red;
    color: red;
    padding: 0;
   }
  </style>
 </head>
 <body>
  <p id="A" class="b">Стиль идентификатора</p>
  <p class="a b">Стиль классов a и b</p>
  <p class="b">Стиль класса b</p>
 </body>
</html>
```

Для первого абзаца устанавливается стиль от идентификатора A и класса b, свойства которых противоречат друг другу. При этом стиль класса будет игнорироваться из-за особенностей каскадирования и специфичности. Для второго абзаца стиль задаётся через классы a и b одновременно. Приоритет у классов одинаковый, значит, в случае противоречия будут задействованы те свойства, которые указаны в стиле ниже. К последнему абзацу применяется стиль только от класса b. На рис. 1.46 показан результат применения стилей.

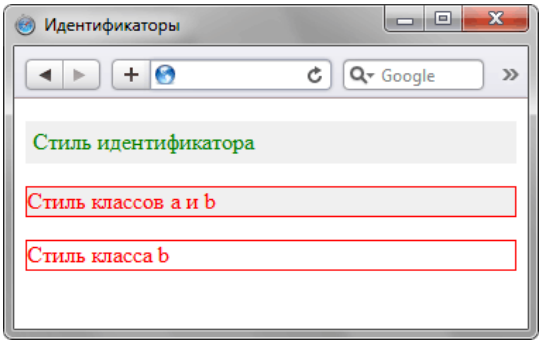


Рис. 1.46. Использование стилей для текста

Специфичность в каскадировании начинает играть роль при разрастании стилевого файла за счет увеличения числа селекторов, что характерно для больших и сложных сайтов. Чтобы стиль применялся корректно, необходимо грамотно управлять специфичностью селекторов путем использования идентификаторов, повышения важности через !important, порядком следования свойств.

Классы

Поскольку к элементу одновременно можно добавлять более одного класса, это позволяет завести

несколько универсальных классов со стилевыми свойствами на все случаи и включать их к тегам при необходимости. Предположим, что большинство блоков на странице имеют закругленные уголки, причем некоторые блоки еще имеют красную рамку, а некоторые нет. В этом случае можем написать такой стиль (пример 1.72).

Пример 1.72. Использование классов

```
.r, .b {
  padding: 10px;
  background: #FCE3EE;
}
.r {
  border-radius: 8px;
  -webkit-border-radius: 8px;
  -moz-border-radius: 8px
}
.b { border: 1px solid #ED1C24; }
.n { border: none; }
```

Указывая разные классы в атрибуте class мы можем комбинировать набор стилевых свойств и получить таким образом блоки с рамкой, блоки без рамки, со скругленными или прямыми уголками. Это несколько похоже на группирование селекторов, но обладает большей гибкостью.

Написание эффективного кода

В процессе написания CSS следует придерживаться некоторых принципов, которые позволяют сократить код CSS, сделать его более удобным, наглядным и читабельным. Читабельность в данном случае означает, что разработчик спустя какое-то время может легко понять и модифицировать стиль или что в коде разберется даже сторонний человек.

Размещайте каскадные таблицы стилей в отдельном файле

Размещение стилей в отдельном файле позволяет ускорить загрузку веб-страниц за счет уменьшения их кода, а также кэширования файла с описанием стиля.

Удаляйте неиспользуемые селекторы

Большое количество селекторов создает путаницу в вопросе о том, кто из них за что отвечает, да и просто увеличивает объем документа. Чтобы этого не произошло, удаляйте селекторы, которые никак не применяются на сайте. К сожалению, определить точно, какой селектор используется, а какой нет, довольно сложно, поэтому добавляйте комментарий в код. Это поможет хотя бы не запутаться в большом объеме текста.

Применяйте группирование

Достоинство и удобство группирования состоит в описании одинаковых свойств в одном месте. Тем самым, значение свойства пишется только один раз, а не повторяется многократно.

Используйте универсальные свойства

Вместо того чтобы указывать значения отступа на каждой стороне элемента через свойства margin-left, margin-right, margin-top и margin-bottom, это можно одновременно задать через универсальное свойство margin. Перечисление значений через пробел позволяет установить индивидуальные отступы для каждой стороны. Кроме margin к универсальным свойствам относятся background, border, font, padding. Применение этих свойств сокращает объем кода и повышает его читабельность.

Форматирование кода

Существует множество разных подходов, как же писать CSS-код. Кто-то упорядочивает селекторы по блокам, другой согласно структуре документа, третий по алфавиту, в общем, сколько людей, столько и мнений. Вы можете воспользоваться онлайновым инструментом, который форматирует CSS-код сразу четырьмя разными способами. А там уже сами решите, какой из способов вам симпатичнее.

Ссылка на сайт

http://www.cssportal.com/format-css/

Принцип работы очень простой, вводите в текстовое поле свой код, нажимаете на кнопку «Format Code» и получаете четыре разных вида первоначального кода. Для примера возьмём следующий небольшой фрагмент.

```
body {
 font: 0.9em Arial, Verdana, Helvetica, sans-serif;
 color: #000;
 background: #fff;
 margin: 0;
}
.top {
 margin-bottom: 10px;
 padding-left: 3%;
 border-bottom: 1px solid #acacac;
}
```

В результате его форматирования получатся следующие варианты.

Форматированный CSS (Formatted CSS)

```
body {
    font: 0.9em Arial, Verdana, Helvetica, sans-serif;
    color: #000;
    background: #fff;
    margin: 0;
}

.top {
    margin-bottom: 10px;
    padding-left: 3%;
    border-bottom: 1px solid #acacac;
}
```

Порядок свойств не меняется, строки со свойствами сдвигаются вправо на четыре пробела, селекторы разделяются между собой пустой строкой.

Свойства в алфавитном порядке (Properties in Alphabetical Order)

```
body {
    background: #fff;
    color: #000;
    font: 0.9em Arial, Verdana, Helvetica, sans-serif;
    margin: 0;
}

.top {
    border-bottom: 1px solid #acacac;
    margin-bottom: 10px;
    padding-left: 3%;
}
```

Строки со свойствами сдвигаются вправо на четыре пробела, селекторы разделяются между собой пустой строкой, стилевые свойства упорядочиваются по алфавиту.

Лесенкой (Longest Property to Shortest)

```
body {
    font: 0.9em Arial, Verdana, Helvetica, sans-serif;
    background: #fff;
    color: #000;
    margin: 0;
}

.top {
    border-bottom: 1px solid #acacac;
    margin-bottom: 10px;
    padding-left: 3%;
}
```

Строки со свойствами сдвигаются вправо на четыре пробела, селекторы разделяются между собой пустой строкой, строки со свойствами упорядочиваются по длине. Вначале идут самые длинные строки, в конце самые короткие.

Компактно (Compact)

```
body {font: 0.9em Arial, Verdana, Helvetica, sans-serif;color:
#000;background: #fff;margin: 0;}
.top {margin-bottom: 10px;padding-left: 3%;border-bottom: 1px solid #acacac;}
```

Селекторы и свойства записываются в одну строку, пустые строки удаляются.

Приведённый инструмент, конечно, не претендует на полноту, в нем нельзя задать величину отступа между селекторами, количество пробелов перед свойством. Также не сокращаются лишние пробелы перед значениями свойств. Тем не менее, главное, что процесс форматирования кода прост и удобен.

Минимизация кода

При редактировании CSS-файла возникает противоречивая задача. С одной стороны код должен быть удобным для восприятия и редактирования, быстрого отыскания нужного селектора, для чего активно

применяются отбивки, комментарии, пробелы и символы табуляции. С другой стороны, объем кода должен быть компактным и не содержать в себе ничего лишнего. Компактность позволяет несколько ускорить загрузку сайта и повысить его производительность.

Данное противоречие решается наличием двух версий CSS-файла: один файл для редактирования, а второй для загрузки на сервер. Сам же процесс сокращения кода называется минимизацией и вполне автоматизирован с помощью сетевых сервисов, которые и рассмотрим далее.

CSSMin

http://tools.w3clubs.com/cssmin/

Простой, даже можно сказать, примитивный сервис, построенный на JavaScript и библиотеке YUI Compressor. Вводите в поле «Source» код CSS, нажимаете кнопку «Crunch» и получаете готовый результат в соседнем поле. Также даётся оценка входного и выходного объёма и соотношение в процентах между ними (рис. 1.47).

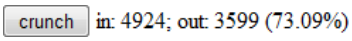


Рис. 1.47. Итог минимизации кода

CSS compressor

http://www.csscompressor.com

Этот сервис удобен тем, что комментирует все свои действия, так что вы будете в курсе изменений вашего стиля. Работает он следующим образом. В поле CSS Input вставляете код CSS, выбираете желаемые настройки и нажимаете кнопку «Compress» (рис. 1.48).

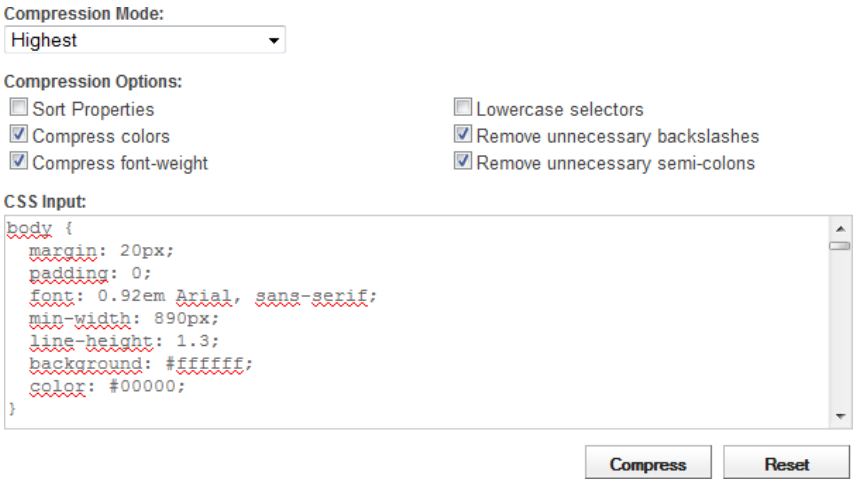


Рис. 1.48. Интерфейс

Настройки следующие.

- Compression Mode — режим сжатия. Различается четырьмя видами.
  - Highest — все правила записываются в одну строку.
  - High — каждое правило записывается на своей строке.
  - Standard — каждое свойство пишется на отдельной строке,
  - Low — каждое свойство пишется на отдельной строке и отбивается табуляцией.
- Sort Properties — сортировка стилевых свойств в алфавитном порядке.
- Compress colors — цвета вида #ffffff заменяются сокращённой формой #fff.
- Compress font-weight — оптимизируется насыщенность шрифта. Такое значение font-weight как normal заменяется на 400, а bold на 700.
- Lowercase selectors — все селекторы записываются в нижнем регистре.

- Remove unnecessary backslashes — ненужные слэши (\) удаляются.
- Remove unnecessary semi-colons — удалить необязательную точку с запятой в последнем свойстве.

После сжатия выводятся два поля: список сделанных изменений в свойствах и сжатый CSS (рис. 1.49).

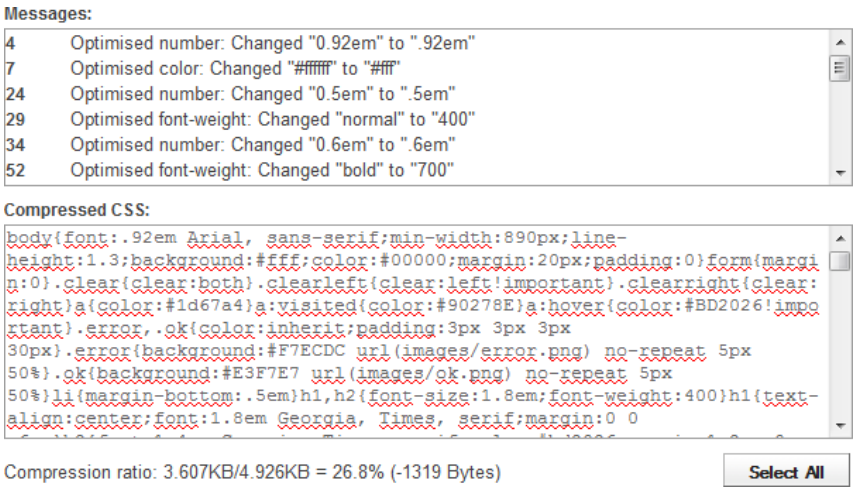


Рис. 1.49. Результат использования

CSS Code Formatter and Optimizer

http://www.generateit.net/css-optimize

Сервис построен на том же движке, что и предыдущий, поэтому имеет ряд схожих настроек. Из приятных плюсов можно отметить подсветку синтаксиса кода, сохранение в файл, а также ввод сетевого адреса CSS-файла.

На рис. 1.50 показано окно настроек.

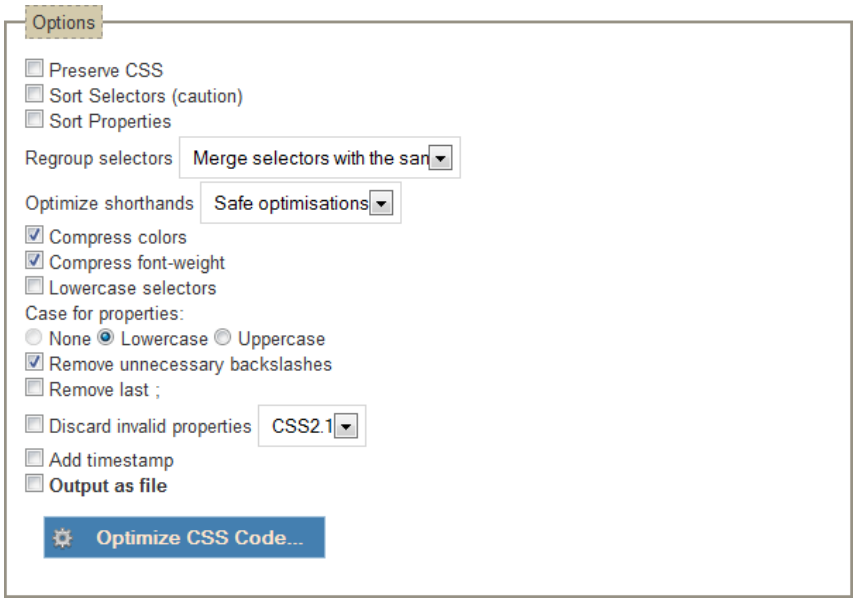


Рис. 1.50. Настройки

- Preserve CSS — сохраняет все комментарии, хаки и др. При включении этой настройки некоторые опции становятся недоступными.
- Sort Selectors (caution) — сортировать селекторы по алфавиту.
- Sort Properties — сортировать свойства по алфавиту.
- Regroup selectors — позволяет перегруппировать селекторы, например, разделить или объединить их.
- Optimize shorthands — оптимизирует универсальные свойства вроде margin.

- Compress colors — цвета вида #ffffff заменяются сокращённой формой #fff.
- Compress font-weight — оптимизируется насыщенность шрифта. Такое значение font-weight как normal заменяется на 400, а bold на 700.
- Lowercase selectors — все селекторы записываются в нижнем регистре.
- Case for properties — стилевые свойства пишутся в нижнем или верхнем регистре.
- Remove unnecessary backslashes — удалить ненужные слэши (\).
- Remove last ; — удалить необязательную точку с запятой в последнем свойстве.
- Discard invalid properties — удалить свойства, которых нет в указанной спецификации.
- Add timestamp — включить в код текущую дату и время.
- Output as file — сохранить результат в виде файла.

Библиотека minify

Если приходится часто вносить изменения в CSS-файл, то процесс минимизации становится неудобным. Сами посудите, вначале надо отредактировать файл, затем его минимизировать и полученный код сохранить в файл, который нужно залить на сервер. Слишком много действий приходится совершать ради одного изменения. Логичнее было бы возложить задачу минимизации на сайт. Загрузили файл на сервер, и вот он уже в компактном виде отдается посетителям. Одно из таких универсальных решений называется minify, это библиотека на PHP5. Она минимизирует, объединяет и кэширует CSS-файлы, а также JavaScript.

Ссылка на проект minify
http://code.google.com/p/minify/

Библиотека minify существует как отдельно, так и в виде плагина для WordPress.

Процесс использования библиотеки следующий. Скачиваете архив, внутри него лежит каталог min, который необходимо переписать на сервер. Сжатие CSS-файла происходит довольно просто, вместо обычного пути к стилевому файлу теперь указываем:

```
http://example.ru/min/?f=themes/default/style.css
```

В параметре `f` указывается путь к CSS-файлу относительно корня сайта. Два и более файла пишутся через запятую:

```
http://example.ru/min/?f=themes/default/style.css,themes/default/cms.css
```

Также процесс получения адреса можно автоматизировать, перейдя по адресу http://example.ru/min, откроется страница, где предлагается указать путь к файлам, которые вы желаете минимизировать (рис. 1.51).

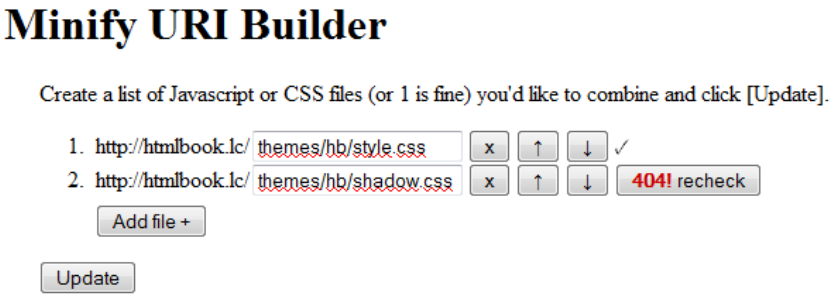


Рис. 1.51. Страница для управления файлами

Кнопки со стрелками вверх и вниз нужны для изменения порядка файлов, а кнопка «x» для удаления файла из списка. Проверка правильности пути осуществляется автоматически, в случае ошибки

появится кнопка с надписью «404!», как показано на рисунке выше. Для добавления еще одного файла в список служит кнопка «Add file». После того, как все файлы указаны, пути к ним заданы корректно, что подтверждается наличием галочки напротив каждого файла, можно нажать кнопку «Update». Ниже на странице появится ссылка на новый комбинированный файл (текст с надписью URI) и тег `<link>` (текст с надписью HTML), который требуется вставить себе на страницу взамен старого (рис. 1.52).

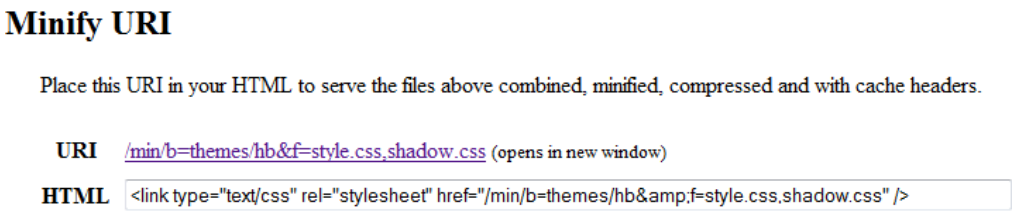


Рис. 1.52. Результат минимизации

Согласно тестам над WordPress использование библиотеки позволяет сократить количество HTTP-запросов и уменьшить объем CSS и JavaScript-файлов до 70%.

Кроме приведенных методов минимизации CSS-файлов имеются также решения, позволяющие архивировать файлы утилитой gzip прямо на сервере и отдавать браузеру упакованную версию. Современные браузеры прекрасно понимают gzip и распаковывают его на месте. Весь процесс происходит автоматически и приводит к существенному сокращению объема передаваемых по сети файлов. Вопросы настройки gzip выходят за рамки этой книги, поэтому я не буду на них останавливаться. Всем заинтересованным рекомендую статьи по этим ссылкам.

- Модуль mod_deflate
- Сжатие ответов Веб-сервера Apache средствами модуля mod_deflate
- Настройка mod_gzip – сжатие трафика в Apache

Глава I

Введение в CSS

Глава II

Режимы браузеров

Во время противостояния браузеров Internet Explorer и Netscape каждый из разработчиков старался улучшить своё детище, чтобы усилить позиции программы на рынке и привлечь больше пользователей. Netscape 4 и IE4 ужасно поддерживали веб-стандарты, поэтому следующая версия, IE5 должна была не только исправить ошибки IE4, но и показать улучшенную поддержку спецификации CSS. Это было необходимо еще и по политическим мотивам, поскольку компания Microsoft вошла в группу W3C и начала оказывать сильное влияние на разработку HTML и CSS.

В процессе работы над браузером IE5 его разработчики столкнулись с неожиданной трудностью. Разница при отображении страницы в разных версиях браузера была настолько велика, что множество сайтов оказались бы неработоспособными при просмотре в IE5. Идея сделать кнопку для переключения в режим совместимости пришла только в версии 8.0, поэтому разработчики IE5 пошли другим путём. Все старые страницы отображались по старым правилам, а для включения режима поддержки стандартов в код страницы необходимо добавить элемент **<!DOCTYPE>** (доктайп).

Браузер IE5 под Mac стал первым браузером, у которого появилось два режима отображения страниц — режим совместимости и стандартный режим. Идея понравилась и распространилась среди разработчиков других браузеров, так что подобные режимы вскоре появились в Mozilla, Safari и Opera. IE5 под Windows, а также старые браузеры вроде Netscape 4 используют только режим совместимости.

Режим браузера для просмотра конкретной веб-страницы устанавливается через элемент **<!DOCTYPE>**, который является обязательным согласно спецификации HTML и XHTML. Сложности возникают из-за того, что по сути доктайп не один, а различается дополнительными параметрами, влияющими в итоге на режим отображения страницы.

Стандартный режим

Режим поддержки стандартов (X)HTML и CSS. Для переключения браузера в этот режим используется один из следующих доктайпов.

Для HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-
html40/strict.dtd">
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

```
<!DOCTYPE html>
```

Для XHTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Также некоторые браузеры переключаются в стандартный режим при отправке документа как text/xml, application/xml или application/xhtml+xml (об этом ниже).

Браузер IE до версии 7.0 включительно не поддерживает стандартный режим и при использовании любого доктайпа указанного выше переходит в почти стандартный режим.

В этом режиме поддерживаются правила спецификации CSS и игнорируются свойства с ошибками. Вот некоторые особенности анализа стилей в стандартном режиме.

Числа без указания единиц измерения игнорируются
Если в стилях при указании размеров задано одно лишь число без упоминания единиц (width: 500 вместо width: 500px), такое значение игнорируется.

Чувствительность к регистру
Имена классов и идентификаторов чувствительны к регистру, поэтому классы с именами `mydiv` и `MyDiv` разные.

Имена идентификаторов и классов
Имена должны начинаться с латинской буквы, а не числа.

Ширина блока
Ширина блока складывается из значений свойств width, padding, border и margin.

Высота блока
Если высота блока указана явно, то при превышении этого значения текст начинает отображаться поверх блока. В противном случае высота блока зависит от высоты контента.

Изображения
По умолчанию для изображений свойство display установлено как inline, при этом внизу картинок добавляется небольшой отступ. Это связано с тем, что нижний край изображения располагается на базовой линии текста, ниже которой должно быть дополнительное пространство для вывода «хвостиков» у букв вроде «ц», «щ», «у» и др.

В примере 2.1 продемонстрирован код переводящий браузер в стандартный режим.

Пример 2.1. Стандартный режим

HTML 5.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>Стандартный режим</title>
 </head>
 <body>
  <p>HTML5</p>
 </body>
</html>
```

Почти стандартный режим

Этот режим основан на стандартном режиме с некоторыми исключениями: отображение изображений внутри ячеек таблицы и рисунков друг под другом происходит как в режиме совместимости. Для переключения в почти стандартный режим применяется один из следующих доктайпов.

Для HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Для фреймов в HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

Для XHTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
```

Для фреймов в XHTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN">
```

Изображения друг под другом

При выводе изображений вертикально с переводом строки через **
** картинки в почти стандартном режиме выводятся слитно без промежутков. В примере 2.2 показан код для вывода двух изображений.

Пример 2.2. Вывод двух изображений

HTML 4.01 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Изображения по вертикали</title>
 </head>
 <body>
  <p class="img"><img src="images/ecctitle.png" alt="Название"
    width="640" height="158"><br>
   <img src="images/navigate.png" alt="Навигация" width="640"
    height="30"></p>
 </body>
</html>
```

Поскольку рисунки предварительно были одним и «разрезаны» для удобства, они образуют единое изображение (рис. 2.1).

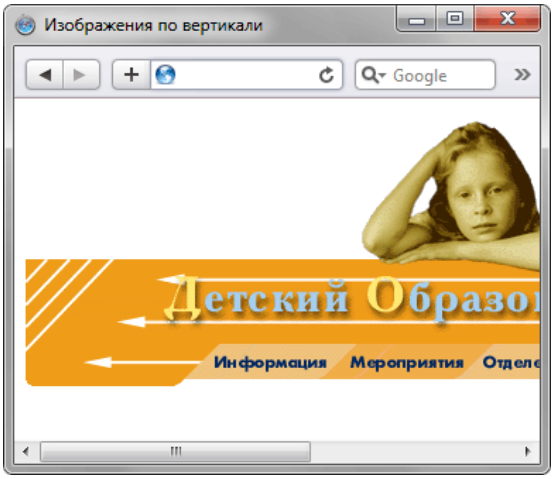


Рис. 2.1. Вывод изображений в почти стандартном режиме

В стандартном режиме между изображениями образуется небольшой промежуток (рис. 2.2).

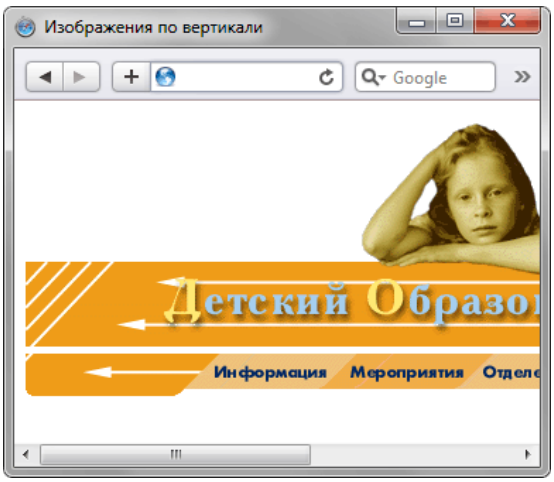


Рис. 2.2. Вывод изображений в стандартном режиме

Для обхода этой особенности в стандартном режиме существует два основных способа:

1. добавить display: block для изображений;
2. использовать свойство line-height для родительского контейнера.

Рассмотрим эти способы подробнее.

Превращение тега **<img>** в блочный элемент еще не раз поможет нам для обхода различных неприятностей, связанных с изображениями. В этот раз воспользуемся той особенностью, что блочные элементы выстраиваются друг под другом слитно (отступы в расчёт не принимаем). При этом тег **
** из кода, конечно же, следует убрать (пример 2.3).

Пример 2.3. Использование свойства block

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Изображения по вертикали</title>
  <style type="text/css">
   DIV.img IMG { display: block; }
  </style>
 </head>
 <body>
  <div class="img"><img src="images/ecctitle.png" alt="Название"
    width="640" height="158" />
   <img src="images/navigate.png" alt="Навигация"
    width="640" height="30" /></div>
 </body>
```

```
</html>
```

Не обязательно применять свойство block, также можно воспользоваться line-height, это свойство задает межстрочный интервал. Установив значение 1px для тега <p>, внутри которого располагаются изображения, мы получим аналогичный результат (пример 2.4).

Пример 2.4. Использование свойства line-height

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Изображения по вертикали</title>
  <style type="text/css">
  P.img { line-height: 1px; }
  </style>
</head>
<body>
  <p class="img"><img src="images/ecctitle.png" alt="Название"
   width="640" height="158" /><br />
   <img src="images/navigate.png" alt="Навигация"
   width="640" height="30" /></p>
</body>
</html>
```

Изображения в таблице

При добавлении изображения в ячейку таблицы также проявляется разница между режимами браузера. Для понимания разберем следующий код (пример 2.5). Чтобы стала заметна граница вокруг таблицы, в стилях добавлено свойство border для селектора **TABLE**.

Пример 2.5. Изображение в таблице

HTML 4.01 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
 <title>Изображения в таблице</title>
  <style type="text/css">
   TABLE { border: 1px solid #000; }
  </style>
 </head>
 <body>
  <table cellspacing="0">
   <tr><td>
    <img src="images/figure.jpg" alt="Винни-Пух" width="111" height="110">
   </td></tr>
  </table>
 </body>
</html>
```

Результат данного примера показан на рис. 2.3а. Для стандартного режима вывод изображения несколько отличается (рис. 2.3б).

а *б*

Рис. 2.3. Изображение в таблице. а — почти стандартный режим, б — стандартный режим

Заметно, что в стандартном режиме появляется небольшой отступ внизу картинки. Откуда он берется? Если добавить в ячейку текст и увеличить масштаб (рис. 2.4), то хорошо заметно, что изображение как строчный элемент выравнивается по базовой линии текста, а не по его нижнему краю. Соответственно, разница между базовой линией и нижним краем текста и есть значение промежутка внизу картинки.

Рис. 2.4. Базовая линия текста

Опять же, существует несколько способов изменить поведение изображений в таблице. Первый способ уже упоминался, это преобразование тега **<img>** в блочный элемент с помощью свойства display со значением block (см. пример 2.3). Стиль в таком случае будет следующий:

```
TABLE IMG { display: block; }
```

Если наряду с изображениями внутри ячейки находится текст, этот стиль может привести к нежелательным последствиям. Вместо того чтобы картинка располагалась рядом с текстом, она, как блочный элемент, появится на новой строке. В этом случае рекомендуется задать выравнивание изображений по нижнему краю через свойство vertical-align со значением bottom (пример 2.6).

Пример 2.6. Выравнивание изображений в стандартном режиме

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Изображения в таблице</title>
  <style type="text/css">
   TABLE { border: 1px solid #000; }
   TD IMG { vertical-align: bottom; }
  </style>
 </head>
 <body>
  <table class="data" cellspacing="0">
   <tr><td>
    Текст <img src="images/figure.jpg" alt="Винни-Пух"
    width="111" height="110" />
   </td></tr>
  </table>
 </body>
</html>
```

Хотя во всех браузерах в данном примере наблюдается выравнивание изображения по нижнему краю, несколько различается выравнивание самого текста. Браузеры Firefox, Safari, IE7 выравнивают текст по нижнему краю рисунка, а Opera, IE8, IE9 — по верхнему.

Однопиксельные рисунки

Во времена табличной вёрстки активно применялось прозрачное изображение размером один на один пиксел, так называемый однопиксельный рисунок. Помещённый в ячейку таблицы такой рисунок не давал ей сжиматься до определенных пределов, но сам легко масштабировался, задавая тем самым ширину или высоту ячейки. Поскольку рисунок прозрачный, можно установить для ячейки фоновый цвет и получить горизонтальные или вертикальные линии заданной толщины.

В стандартном режиме нас ожидают те же проблемы, что и при использовании обычных рисунков внутри ячеек. Высота ячейки будет расширена, поскольку к изображениям добавляется отступ снизу. Решается эта проблема гораздо проще и не требует использование дополнительного стиля. Современные браузеры прекрасно отображают заданные размеры ячеек и без наличия дополнительных изображений внутри. Поэтому достаточно просто удалить однопиксельный рисунок из ячейки.

Было:

```
<td><img src="images/1x1.gif" width="1" height="1" alt=""></td>
```

Стало:

```
<td class="line"></td>
```

Здесь класс `line` задает размеры ячейки.

Режим совместимости

Этот режим предназначен для отображения веб-страницы подобно старым браузерам. В режиме совместимости игнорируются стандарты HTML и CSS, и поведение браузеров становится непредсказуемым. Для переключения в режим совместимости существует множество доктайпов, вот лишь некоторые из них.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">
```

Также переход происходит, если доктайп вообще не указан или не может быть распознан.

Хотя браузеры по-разному интерпретируют код документа в режиме совместимости, некоторое поведение у них совпадает.

Высота таблицы и строки
Можно задать высоту таблицы или строки (тег **<tr>**) в процентах, пикселах или других единицах. В стандартном режиме атрибут height у тега **<table>** и **<tr>** игнорируется.

Размер шрифта в таблице
Если размер текста в ячейке таблицы устанавливается в процентах, то он берется не относительно размера для **<body>**, а от значения по умолчанию, которое обычно равно 16px. Иными словами, значение font-size для селектора **BODY** не принимается во внимание.

Имена идентификаторов и классов начинаются с цифры
В браузере IE и Opera имена классов и идентификаторов можно начинать с цифры. В остальных браузерах и в стандартном режиме такие имена недопустимы.

Нечувствительность к регистру
Имена идентификаторов и классов не зависят от регистра написания, имена `mydiv` и `MyDiv` считаются одинаковыми.

Цвета
Цвет в шестнадцатеричном значении можно задавать без символа решетки впереди (fc0 вместо #fc0). В стандартном режиме такие цвета игнорируются.

Пикселы по умолчанию
Если в стилях в качестве единицы размера указано число без единицы измерения, считается, что значение задано в пикселах. В стандартном режиме такие значения игнорируются.

Пробел после значения в CSS
В стилях допустимо ставить пробел перед единицей измерения (10 px, а не 10px). В стандартном режиме такие значения игнорируются.

Псевдокласс :hover
Псевдокласс :hover может добавляться к ссылкам, изображениям и элементам форм, только если селектор включает имя тега, идентификатора или атрибут. Запись **.test:hover** не работает в браузерах IE и Firefox, в то время как **a.test:hover** понимается всеми браузерами в режиме совместимости.

margin: auto не работает в IE
Для блоков с заданной шириной margin со значением auto не выравнивает блок по центру в браузере IE.

Ширина блока
Ширина блока равна значению width. Поля (padding) и границы (border) не влияют на ширину и находятся внутри блока.

Высота блока
Заданная высота блока игнорируется, когда высота контента превышает указанную высоту блока, которая при этом увеличивается согласно высоте контента.

Ширина строчных элементов
IE позволяет установить ширину и высоту строчных элементов вроде `<span>`. В стандартном режиме и в других браузерах значения размеров для строчных элементов игнорируются.

Здесь перечислены не все проблемы, возникающие в режиме совместимости, но этого вполне достаточно, чтобы сделать вывод, о том, что этот режим использовать не надо. В примере 2.7 допущены некоторые ошибки HTML и CSS характерные для режима совместимости.

Пример 2.7. Страница в режиме совместимости

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <style>
   body { background: F2E0BE; }
   table {
    width: 100%;
    height: 100%;
   }
   table td { text-align: center; }
   #2mc {
    font-size: xx-large;
   }
   sup { font-size: 24; }
  </style>
 </head>
 <body>
  <table>
   <tr><td>
    <div id="2mc">E = mc<sup>2</sup></div>
   </td></tr>
  </table>
 </body>
</html>
```

Только в браузере Opera результат получился желаемым (рис. 2.5), в остальных браузерах наблюдаются более или менее сильные различия со шрифтами, размером и цветом фона.

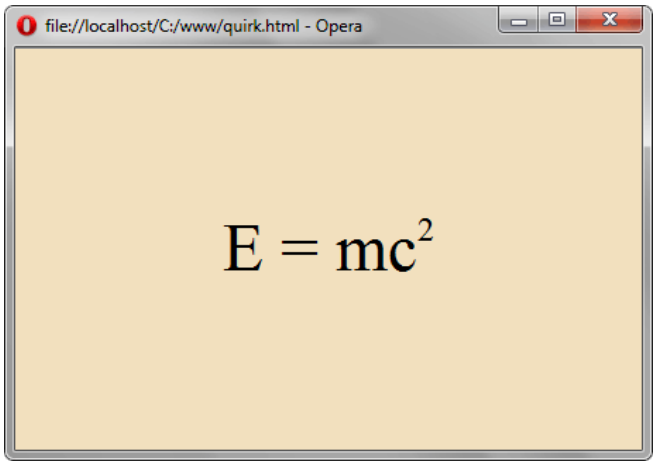


Рис. 2.5. Результат в браузере Opera 11

В браузере IE6 имеется ошибка, связанная с доктайпом, он обязательно должен находиться в первой строке кода. Если вы добавляете в первую строку кода не доктайп, а другой элемент или текст, браузер IE6 переходит в режим совместимости (пример 2.8).

Пример 2.8. Ошибка в IE6

XHTML 1.0 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Стандартный режим</title>
 </head>
 <body>
  <p>...</p>
 </body>
</html>
```

В данном примере кодировка документа задаётся с помощью элемента **<?xml ?>** в первой строке кода, что приводит к переходу в режим совместимости в IE6, несмотря на строгий доктайп. В браузере IE7 эта ошибка исправлена.

MIME-тип документа

MIME (Multipurpose Internet Mail Extensions, многоцелевые расширения интернет-почты) — стандарт Интернет, является частью протокола HTTP. Задача MIME это идентификация типа содержимого документа по его заголовку. К примеру, текстовый файл имеет тип text/plain, а HTML-файл — text/html. Отправка заголовка обычно происходит на основе расширения файла веб-сервером.

Документы XHTML по умолчанию отправляются как text/html, что в действительности говорит о том, что мы имеем дело с HTML, а не XHTML-файлом. Чтобы задействовать возможности XHTML требуется отдавать файл с типом application/xhtml+xml. Если у вас установлен веб-сервер Apache, то вы можете сделать это через директиву AddType, добавив следующую строку в файл .htaccess, расположенный в корне сайта.

```
AddType application/xhtml+xml .xhtml
```

В данном случае мы говорим, что все файлы с расширением .xhtml отдавать как application/xhtml+xml. Если документы формируются через PHP, то можно отдавать заголовок следующим образом:

```
header ("Content-type: application/xhtml+xml");
```

Учтите, что эта строка должна идти до вывода любого текста на странице.

Браузер Internet Explorer до версии 8.0 включительно не поддерживает тип application/xhtml+xml и не сможет отобразить страницу, которая отдаётся с этим типом. Остальные браузеры, в том числе IE9, понимают этот тип как переход в стандартный режим.

Тип application/xhtml+xml необходим в случае, когда в документе применяется MathML (Mathematical Markup Language, язык математической разметки), предназначенный для добавления формул или SVG (Scalable Vector Graphics, масштабируемая векторная графика), язык разметки для создания на странице векторных рисунков. Если вы ничего не знаете об этих технологиях и пока не собираетесь их использовать, лучше отдавать документ как text/html. Это позволит охватить наибольшее количество браузеров и поисковых систем.

По сути, тип text/html для файлов с расширением .html или .htm настроен автоматически, поэтому не требуется предпринимать каких-либо действий для этого типа.

Доктайп

Кроме переключения браузера в один из режимов, доктайп также сообщает, согласно каким правилам синтаксиса проводить проверку текущего документа. К примеру, для HTML 4.01 и XHTML 1.0 существует по три разных типа доктайпа, для HTML5 лишь один доктайп.

Строгий доктайп

Применяется строгий синтаксис языка, допускается включать все теги и атрибуты, кроме осуждаемых.

Для HTML 4.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Для XHTML 1.0:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Правила синтаксиса зависят от используемой версии, но в целом должны соблюдаться следующие.

- Осуждается использование следующих тегов: **<applet>**, **<basefont>**, ** **, **<dir>**, **<font>**, **<isindex>**, **<noframes>**, **<plaintext>**, **<s>**, **<strike>**, **<u>**, **<xmp>**. Взамен по возможности рекомендуется использовать стили.
- Текст, изображения и элементы форм нельзя напрямую добавлять в **<body>**, эти элементы должны обязательно находиться внутри блочных элементов вроде **<p>** или ** **.
- Осуждается применение атрибутов target, start (тег **<ol>**), type (теги **<li>**, **<ol>**, **<ul>**) и др.

Естественно, также должен соблюдаться синтаксис языка — правильное вложение тегов, закрытие тегов, должны присутствовать обязательные теги и др. Документ с неявными ошибками приведён в примере 2.9.

Пример 2.9. Невалидный код HTML

HTML 4.01 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
  <title>Заголовок</title>
 </head>
 <body>
  <form>
   <input type="text">
  </form>
 </body>
</html>
```

В данном примере у тега **<form>** нет обязательного атрибута action, тег **<input>** располагается напрямую внутри формы, хотя должен быть заключён в блочный тег. Также не указана кодировка через метатег. Это не является ошибкой, но пользователям в некоторых случаях придется самостоятельно указывать её в браузере. Валидный код XHTML с учётом исправленных ошибок при строгом доктайпе продемонстрирован в примере 2.10.

Пример 2.10. Валидный код XHTML

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
  <title>Заголовок</title>
 </head>
 <body>
  <form action="">
   <p><input type="text" /></p>
  </form>
 </body>
</html>
```

Переходный доктайп

Применяется «мягкий» синтаксис языка, допускается использовать все теги и атрибуты, включая осуждаемые.

Для HTML 4.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Для XHTML 1.0:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Цель переходного доктайпа заключается в постепенном знакомстве с синтаксисом языка. Сразу использовать строгий синтаксис кому-то покажется слишком сложным, для таких людей и предназначен переходный HTML и XHTML. К тому же некоторые привычные вещи в строгой версии оказались под запретом, поэтому переходный доктайп покажется панацеей для тех, кто не может отказаться от старых привычек. Речь, к примеру, идёт об атрибуте target, который позволяет открывать ссылку в новом окне. Использование target осуждается, в том числе из-за системы вкладок, ставших стандартом де-факто в браузерах. Одному пользователю нравится открывать ссылку в текущем окне, другому в новой вкладке, а третьему в новом окне. Чтобы поведение ссылки не зависело от прихоти разработчика, атрибут target запрещен при строгом доктайпе. При переходном доктайпе добавление target к тегу **<a>** абсолютно валидно (пример 2.11).

Пример 2.11. Использование переходного доктайпа

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Ссылка в новом окне</title>
 </head>
 <body>
  <a href="http://htmlbook.ru" target="_blank">Открыть в новом окне</a>
 </body>
</html>
```

Фреймы

Применяется во фреймовой структуре в документе, по синтаксису аналогичен переходному доктайпу.

Для HTML 4.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

Для XHTML 1.0:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Данный доктайп применяется только для главного документа, формирующего структуру фреймов с помощью тегов **<frameset>** и **<frame>** (пример 2.12).

Пример 2.12. Использование фреймов

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Фреймы</title>
 </head>
 <frameset rows="*,80" cols="*">
  <frameset cols="80,*">
   <frame src="left.html" name="leftFrame" scrolling="no"
     noresize="noresize" />
   <frame src="main.html" name="mainFrame" />
  </frameset>
  <frame src="bottom.html" name="bottomFrame" scrolling="no"
    noresize="noresize" />
 </frameset>
</html>
```

Документы, открываемые внутри фреймов (в данном примере это left.html, main.html и bottom.html), используют строгий или переходный доктайп, но никак не с фреймами.

HTML5

В HTML5 существует лишь один тип доктайпа, который переводит браузер в стандартный режим.

```
<!DOCTYPE html>
```

Подробнее об HTML5 будет рассказано в следующих главах.

Режимы Internet Explorer

Вокруг Internet Explorer сложилась ситуация, которая больше не прослеживается ни с одним другим браузером — разброс версий начинается с 6.0 и заканчивается 9.0. Причины использования устаревших версий могут быть совершенно разные.

- Internet Explorer встроен в операционную систему Windows и неопытные пользователи даже не подозревают о наличии альтернативы.
- Корпоративные пользователи с ограниченным доступом не могут самостоятельно обновить версию или сменить браузер.
- Обновление IE до новой версии происходит через систему Windows Update, которая часто отключается по требованиям безопасности или для снижения интернет-трафика.
- Некоторые приложения, например, банковские, могут быть «заточены» для работы только под конкретную версию IE.
- В силу инерции мышления, когда пользователю нравится та программа, с которой он привык работать.

Сама компания Microsoft всячески поощряет переход на новые версии Internet Explorer и в 2010 году прекратила поддержку IE6 и IE7.

Что касается разработчиков сайтов, то для них такой большой набор версий это настоящая проблема. Каждая версия IE содержит свои уникальные ошибки, особенности отображения веб-страниц, а также не поддерживает какие-то свойства CSS. Одним из радикальных вариантов решения проблемы является полный отказ разработчика от поддержки определенных версий IE. Так, некоторые сервисы Google, сайт vkontakte.ru при попытке зайти на них через IE6 выдают предупреждение о том, что вы используете устаревший браузер. С другой стороны, некоторые разработчики интернет-магазинов бьются за каждый браузер, полагая, что даже если каким-то браузером пользуется 2%, то отказ от него приведет к потере 2% клиентов.

Из-за того, что каждая версия IE может отображать сайт по своему, разработчики IE8 оказались перед

часть сайтов «рассыпалась». В итоге было принято решение добавить режим представления совместимости; для быстрого переключения сайтов в этот режим возле адресной строки добавлена специальная кнопка (рис. 2.6). В действительности при переключении в этот режим браузер начинал работать как версия 7.0. В IE9 пошли еще дальше и в нём уже можно переключаться на IE8 или IE7.

Рис. 2.6. Кнопка для переключения в режим совместимости в IE8

Обилие версий IE усложняется еще тем, что версии браузера могут работать в нескольких режимах, порождая большое количество комбинаций, которые необходимо учитывать разработчику. Начиная с версии 8.0, переключение режимов в браузере делается через Средства разработчика (`Сервис > Средства разработчика`), которое проще вызвать при нажатии на клавишу `F12`. В этом инструменте доступно два пункта меню связанных с режимами: Режим браузера (рис. 2.7) и Режим документов.

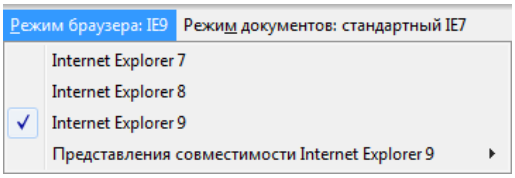


Рис. 2.7. Режимы браузера

Выбор режима браузера определяет следующее:

- строка User-Agent, которую браузер отправляет на сервер, в Microsoft называется «агент пользователя»;
- версия для условных комментариев, которую в Microsoft называют «вектор версии»;
- режим документа.

Агент пользователя представляет собой текстовую строку, отправляемую на сервер для идентификации браузера, его версии и операционной системы. Значение User-Agent определяет не только браузер, но также поисковых пауков и сетевых роботов. Список возможных значений User-Agent и разбор строки можно посмотреть на сайте http://www.useragentstring.com. В табл. 2.1 приведены значения User-Agent при выборе разных режимов IE.

Табл. 2.1. Значения User-Agent

Режим браузера	User-Agent	Описание
IE7	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Tablet PC 2.0; .NET4.0C; AskTbPTV2/5.9.1.14019)	Серверу отправляются данные, что браузер IE7.
IE8	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Tablet PC 2.0; .NET4.0C; AskTbPTV2/5.9.1.14019)	Серверу отправляются данные, что браузер IE8.
IE9	Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)	Серверу отправляются данные, что браузер IE9.
Режим совместимости IE9	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/5.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Tablet PC 2.0; .NET4.0C; AskTbPTV2/5.9.1.14019)	Серверу отправляются данные, что браузер IE7, но значение «Trident/5.0» соответствует IE9.

В вашем случае строка User-Agent может отличаться от приведенных в таблице, поскольку она зависит от версии Windows и её параметров.

Вектор версии позволяет с помощью условных комментариев определять версию IE и отдавать для неё отдельный код. Условные комментарии активно применяются в вёрстке для устранения различий в макете между IE и другими браузерами. Подробнее о них пойдёт речь позже, пока же вы можете ознакомиться с небольшим примером, который показывает текст только в IE8. Остальные браузеры игнорируют этот фрагмент, считая его комментарием.

```
<!--[if IE 8]>
 <p>У вас браузер IE8.</p>
<![endif]-->
```

Режим документа меняется с помощью меню в Средствах разработчика (рис. 2.8).

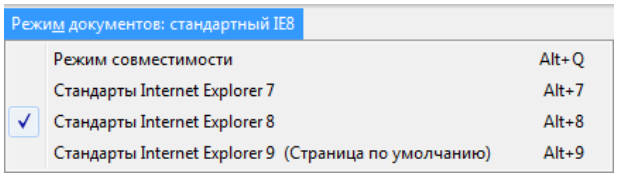


Рис. 2.8. Режимы документа

Начиная с версии IE6, имеются следующие режимы (табл. 2.2).

Табл. 2.2. Режимы документа IE

Режим	Описание
IE5 (режим совместимости)	Несмотря на то, что браузером IE5 уже никто не пользуется, этот режим применяется во всех старших версиях IE при переключении в режим совместимости. Достаточно не указать доктайп и вы, словно на машине времени, вернётесь в прошлый век к отображению в этом браузере.
IE6	Этот режим работает только в версии 6.0.
IE7	Стандартный режим браузера IE7 и режим при переключении на него в IE8 и IE9.
IE8	Стандартный режим браузера IE8 и режим при переключении на него в IE9.
IE9	Этот режим работает только в версии 9.0.

Кроме переключения непосредственно в браузере, задать режим можно через тег **<meta>**, отправив HTTP-заголовок X-UA-Compatible.

```
<meta http-equiv="X-UA-Compatible" content="IE=версия" />
```

Версия может принимать следующие значения (табл. 2.3).

Табл. 2.2. Режимы документа IE

Значение	Описание
5	Вынужденное переключение в режим IE5, доктайп при этом игнорируется.
7	Вынужденное переключение в режим IE7, доктайп при этом игнорируется.
8	Вынужденное переключение в режим IE8, доктайп при этом игнорируется.
9	Вынужденное переключение в режим IE9, доктайп при этом игнорируется.
EmulateIE7	При наличии доктайпа переключается в режим IE7, в противном случае в режим совместимости.
EmulateIE8	При наличии доктайпа переключается в режим IE8, в противном случае в режим совместимости.
EmulateIE9	При наличии доктайпа переключается в режим IE9, в противном случае в режим совместимости.
Edge	Устанавливает документ в наиболее новый доступный режим. Для версии 8.0 это режим IE8, для версии 9.0 это режим IE9.

К примеру, на сайте Яндекса применяется следующий код для эмуляции режима IE7.

```
<meta http-equiv="X-UA-Compatible"content="IE=EmulateIE7">
```

При добавлении данного кода в документ кнопка режима совместимости (рис. 2.6) в браузере исчезает.

Рекомендации

Несмотря на обилие различных доктайпов, их количество для перевода документа в стандартный режим не так и велико и, по сути, сводится всего к трём. Разница между этими доктайпами только в используемой версии HTML или XHTML.

HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

HTML 5

```
<!DOCTYPE html>
```

XHTML 1.0

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Включение именно стандартного режима рекомендуется в связи с его стабильностью и практически одинаковой работой в разных браузерах. В этом режиме браузеры придерживаются спецификаций HTML и CSS, поэтому если вы верстаете по стандартам, отладка сайта сильно упрощается.

Выбор доктайпа из перечисленных выше зависит от собственных предпочтений и знаний (X)HTML. В XHTML строгий синтаксис: все теги, включая одиночные, требуется закрывать, атрибуты и теги писать только в нижнем регистре, все значения заключать в кавычки. В HTML синтаксис проще, закрывать надо не все теги, имена тегов и атрибутов можно писать в любом регистре, кавычки не обязательны.

HTML5 ещё либеральнее, в этой версии соблюдаются правила HTML4, но учтено и некоторое влияние XHTML. Так, одиночные теги можно как закрывать, так и оставлять незакрытыми, убраны обязательные атрибуты вроде type="text/css". Сам HTML5 это не просто развитие языка разметки, сколько сборник новых технологий вроде рисования, анимации, вставки видео и аудио, геолокации, локального хранилища, автономного режима сайта, новых элементов фор и многого другого. «Вкусностей» полно, и на практике плюсов гораздо больше, чем при сравнении между XHTML 1.0 и HTML 4.01.

W3 Consortium заявил, что работа над XHTML 2.0 свёрнута и все материалы переданы в рабочую группу HTML5. Это говорит о том, что дальнейшего продвижения XHTML ждать не придётся, а развитие Интернета пойдет по пути HTML.

Хотя стандарт HTML5 ещё находится в разработке, современные браузеры вполне уже поддерживают множество заявленных в нём возможностей. Так что если вы желаете их использовать на сайте, выбор доктайпа HTML5 становится очевидным.

Если посмотреть тенденции в вёрстке сайтов, также заметен отход от HTML 4.01 в пользу XHTML 1.0 и HTML5. На технологическом конкурсе сайтов WebHiTech (http://webhitech.ru) предпочтение отдавалось сайтам, свёрстанным именно по этим стандартам.

Таким образом, выбор доктайпа сокращается до двух пунктов, либо это строгий XHTML 1.0, либо HTML5.

Глава III

Принципы вёрстки слоями

Несмотря на то, что термин «слой» достаточно устойчиво используется при разработке сайта, расшифровке этого понятия в литературе по сайтостроению практически не уделяется внимание. В дальнейшем я буду активно использовать термин «слой», поэтому вначале необходимо определить, что же под ним подразумевается.

Первоначально слои ввела компания Netscape, включив в свой браузер поддержку тега **<layer>**. Этот тег позволял прятать/показывать текущее содержимое, устанавливать положение относительно окна браузера, накладывать один слой поверх других и загружать данные в содержимое слоя из файла. Что характерно, все эти параметры легко менялись с помощью JavaScript и это расширяло возможности по созданию действительно динамического контента на странице. Несмотря на столь впечатляющий набор возможностей, тег **<layer>** не был включен в спецификацию HTML и так и остался лишь расширением браузера Netscape.

Однако необходимость в указанных возможностях, которые бы применялись на сайтах, уже назрела, и в конце 1996 года синтаксис для работы со слоями был разработан и одобрен в рабочем проекте консорциума «CSS Positioning (CSS-P)». Основная нагрузка ложилась на стили, с их помощью можно управлять видом любого элемента, в том числе менять значения динамически через JavaScript. К сожалению, объектные модели браузеров для доступа к элементам разнились, поэтому приходилось писать достаточно сложный код, который бы учитывал эти различия.

В настоящее время разработчики популярных браузеров стали придерживаться спецификаций HTML и CSS, что сильно облегчило жизнь создателям сайтов, поскольку снизило время на отладку сайта в разных браузерах. Тем не менее, различия в подходах у браузеров существуют и при их возникновении разработчики придерживаются следующих форм работы.

- Если наблюдаются небольшие различия в отображении одного сайта в разных браузерах, то на эти отличия закрывают глаза. Попросту говоря, никак не исправляют. Посетители в любом случае не будут попиксельно сравнивать сайт в разных браузерах. Здесь следует сделать оговорку, что сайт в любом случае должен отображаться корректно и без ошибок.
- Если у сайта имеются существенные различия при его показе в одном и другом браузере, то для их устранения применяют хаки.

Хак — это набор приемов, когда определенному браузеру «подсовывают» код, который понимается только этим браузером, а остальными игнорируется.

Несмотря на то, что хаки работают, использовать их следует ограниченно или вообще обходиться без них. Дело в том, что хаки снижают универсальность кода и для модификации параметров одного элемента приходится вносить изменения одновременно в разных местах.

Есть и другой, перспективный путь — придерживаться спецификации CSS. Несмотря на то, что браузеры не в полной мере сами её поддерживают, они прогрессируют именно в направлении полной поддержки различных спецификаций (HTML, CSS, DOM). Таким образом, получается, что будущие версии браузеров будут унифицированы и один и тот же сайт станут отображать корректно.

Снова вернемся к слоям. Понятно, что они непосредственно связаны со стилями. Раз так, то не получается ли, что каждый элемент HTML-кода, к которому добавляются стили, является слоем? В каком-то смысле так и есть. Однако это внесло бы изрядную путаницу, если вместо «таблица» или «абзац» мы бы говорили «слой». Поэтому договоримся относить этот термин только к тегам ** **.

Слой — это элемент веб-страницы, созданный с помощью тега ** **, к которому применяется стилевое оформление.

Таким образом, вёрстка с помощью слоёв заключается в конструктивном использовании тегов ** ** и стилей. При этом придерживаются следующих принципов.

Разделение содержимого и оформления

Код HTML должен содержать только теги разметки и теги логического форматирования, а любое оформление выносится за пределы кода в стили. Такой подход позволяет независимо управлять видом элементов страницы и её содержимым. Благодаря этому над сайтом может работать несколько человек, при этом каждый выполняет свою функцию самостоятельно от других. Дизайнер, верстальщик и программист работают над своими задачами автономно, снижая время на разработку сайта.

Активное применение тега

При использовании слоев существенное значение уделяется универсальному тегу ** **, который выполняет множество функций. Фактически это основа, на которую «навешиваются» стили, превращая её то в игрушку, то в зверушку. Совершенно не значит, что применяется только один этот тег, нужно ведь и рисунки вставлять и оформлять текст. Но при вёрстке с помощью слоёв тег ** ** является кирпичиком вёрстки, её базовым фундаментом.

Благодаря этому тегу HTML-код распадается на ряд четких наглядных блоков, за счет чего вёрстка слоями называется также блочной вёрсткой. Код при этом получается более компактным, чем при табличной вёрстке, к тому же поисковые системы его лучше индексируют.

Таблицы применяются только для представления табличных данных

При вёрстке слоями, конечно же, используются таблицы, но только в тех случаях, когда они нужны, например, для наглядного отображения чисел и других табличных данных. Вариант, когда от таблиц предлагается отказаться вообще, является нецелесообразным и, более того, вредным.

Использование стилей не является обязательной характерной чертой вёрстки слоями, для табличной вёрстки стили также могут применяться достаточно активно.

Подвём итоги. Слой это базовый элемент вёрстки веб-страниц при которой активно применяются стили и придерживаются спецификаций HTML и CSS. При таком подходе важная роль уделяется тегу ** **, с которым у большинства людей и ассоциируются слои. В каком-то смысле это является верным, поэтому договоримся в дальнейшем употреблять термин «слой» к тегу ** ** для которого указан стилевой идентификатор или класс. Таким образом, выражение «слой с именем content» подразумевает, что используется тег или .

Блочная модель

Любой блочный элемент состоит из набора свойств, подобно капустным листам накладываемых друг на друга. Основой блока выступает его контент (это может быть текст, изображение и др.), ширина которого задается свойством width, а высота через height; вокруг контента идут поля (padding), они создают пустое пространство от контента до внутреннего края границ; затем идут собственно сами границы (border) и завершают блок отступы (margin), невидимое пустое пространство от внешнего края границ. Порядок влияния этих свойств на блок четко определён и не может быть нарушен. На рис. 3.1 показан блок в виде набора этих свойств.

Рис. 3.1. Свойства, формирующие блочный элемент

Для наглядности свойства блока можно представить в виде матрёшек (рис. 3.2). Любую маленькую матрёшку можно вставить в более крупную матрёшку, но никак не наоборот.

Рис. 3.2. Матрёшки

На деле ни одно из этих свойств не является обязательным, в том числе и контент, поэтому вы можете формировать любые блоки, содержащие комбинации свойств padding, border и margin или вообще обойтись без них.

Поля

Полем называется расстояние от внутреннего края границы или края блока до воображаемого прямоугольника, ограничивающего содержимое блока. Из-за того, что значения полей могут различаться на каждой стороне, применяют выражения «верхнее поле» или «поле сверху», и подобные для других сторон. Обозначение «поля» следует понимать как одинаковое значение полей для всех сторон. Основное предназначение полей — создать пустое пространство вокруг содержимого блочного

элемента, например текста, чтобы он не прилегал плотно к краю элемента. Использование полей повышает читабельность текста и улучшает внешний вид страницы. В примере 3.1 показано использование полей для оформления текста.

Пример 3.1.Использование свойства padding

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Поля</title>
  <style type="text/css">
   .space {
    padding: 20px; /* Поля */
    background: #E5D3BD; /* Цвет фона */
    border: 2px solid #E81E25; /* Параметры рамки */
   }
  </style>
 </head>
 <body>
  <div class="space">
   Они шли, чтобы покорить нас, чтобы пытать нас и сжигать живьем на кострах,
   они шли, чтобы сделать с нами, вольными англичанами, то же самое, что
   Кортес сделал с индейцами Анауака.
  </div>
 </body>
</html>
```

Результат примера показан на рис. 3.3.

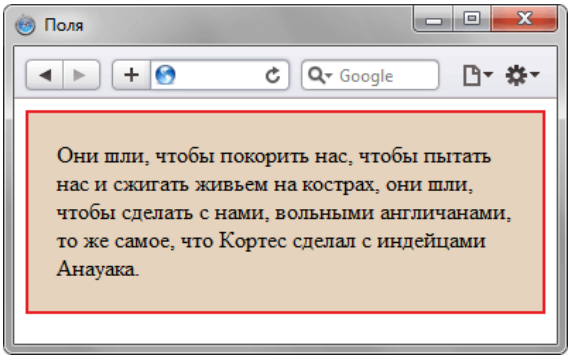


Рис. 3.3. Поля вокруг текста

Значения полей не могут быть отрицательными.

Границы

Границы это линии вокруг полей элемента на одной, двух, трёх или всех четырёх его сторонах. У каждой линии есть толщина, стиль и цвет. Для создания рамки применяется универсальное свойство border одновременно задающее все эти параметры, а для создания линий на отдельных сторонах элемента можно воспользоваться свойствами border-left, border-top, border-right и border-bottom, соответственно устанавливающих границу слева, сверху, справа и снизу. В примере 3.2 показано добавление линии слева от элемента.

Пример 3.2. Красная пунктирная линия

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Линия</title>
  <style type="text/css">
   P.line {
    border-left: 1px dotted red;
    padding: 10px;
   }
  </style>
 </head>
```

```
 <body>
  <p class="line">Лев ревет только в том случае, когда сообщает, что
  территория принадлежит ему или провозглашает себя царем природы.</p>
 </body>
</html>
```

Результат данного примера показан на рис. 3.4.

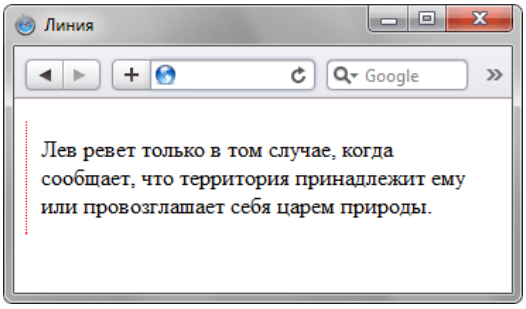


Рис. 3.4. Линия возле текста

Расстояние от линии до текста регулируется значением padding.

Отступы

Отступ это пустое пространство от внешнего края границы, полей или содержимого блока. Как уже упоминалось, границы и поля не обязательны, могут отсутствовать, так что способ формирования отступов зависит от ситуации. Как и в случае с полями, применяют выражения «верхний отступ» или «отступ сверху», и им подобные для других сторон. Обозначение «отступы» следует понимать как одинаковое значение отступов для всех сторон.

Для отступов характерны следующие особенности.

- Отступы прозрачны, на них не распространяется цвет фона или фоновая картинка, заданная для блока. Однако если фон установлен у родительского элемента, он будет заметен и на отступах.
- Отступы в отличие от полей могут принимать отрицательное значение, это приводит к сдвигу всего блока в указанную сторону. Так, если задано margin-left: -10px, это сдвинет блок на десять пикселов влево.
- Для отступов характерно явление под названием «схлопывание», когда отступы у близлежащих элементов не суммируются, а объединяются меж собой.
- Отступы, заданные в процентах вычисляются от ширины блока. Это касается как вертикальных, так и горизонтальных отступов.

В примере 3.3 показано схлопывание отступов и их прозрачность.

Пример 3.3. Использование отступов

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Отступы</title>
  <style type="text/css">
   .layer1, .layer2 {
    background: #F2EFE6;
    border: 1px solid #B25538;
    padding: 10px;
    margin: 20px;
   }
  </style>
 </head>
 <body>
  <div class="layer1">Лев ревет только в том случае, когда сообщает, что
   территория принадлежит ему или провозглашает себя царем природы.</div>
  <div class="layer2">Охотничий участок льва может иметь длину и ширину
   до тридцати километров.</div>
```

```
  </body>
</html>
```

Результат данного примера показан на рис. 3.5. Обратите внимание, что расстояние между блоками равно 20 пикселов, а не 40, которые получаются суммированием верхнего и нижнего отступа у блоков. Это происходит за счёт эффекта схлопывания, при котором близлежащие отступы объединяются.

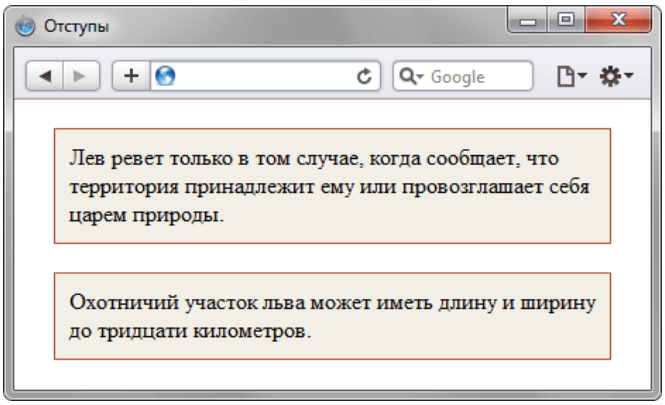


Рис. 3.5. Отступы в элементе

На рисунке хорошо видно, что цвет, задаваемый через свойство background, не выходит за пределы границы элемента и не оказывает влияние на отступы.

Ширина блока

Ширина блока это комплексная величина и складывается из нескольких значений свойств:

- width — ширина контента, т.е. содержимого блока;
- padding-left и padding-right — поле слева и справа от контента;
- border-left и border-right — толщина границы слева и справа;
- margin-left и margin-right — отступ слева и справа.

Как уже упоминалось, какие-то свойства могут отсутствовать и в этом случае на ширину не оказывают влияние. Общая ширина показана на рис. 3.6 в виде чёрной пунктирной линии.

Рис. 3.6. Ширина блока

Если значение width не задано, то оно по умолчанию устанавливается как auto. В этом случае ширина блока будет занимать всю доступную ширину при сохранении значений полей, границ и отступов. Под

доступной шириной в данном случае подразумевается ширина контента у родительского блока, а если родителя нет, то ширина контента браузера.

Допустим, для слоя написан следующий стиль.

```
width: 300px; /* Ширина слоя */
margin: 7px; /* Значение отступов */
border: 4px solid black; /* Параметры границы */
padding: 10px; /* Поля вокруг текста */
```

Ширина слоя согласно этой записи будет равна 342 пиксела, эта величина получается складыванием значения ширины контента плюс отступ слева, граница слева и поле слева, плюс поле справа, граница справа и отступ справа. Суммируем все числа.

Ширина = 300 + 7 + 7 + 4 + 4 + 10 + 10 = 342

Надо отметить, что блочная модель с формированием ширины несет в себе кучу неудобств. Постоянно приходится заниматься вычислениями, когда требуется задать определенную ширину блока. Также начинаются проблемы при сочетании разных единиц измерения, в частности, процентов и пикселов. Предположим, что ширина контента задана как 90%, если сюда приплюсовать поля и границы, заданные в пикселах, то нельзя вычислить суммарную ширину блока, поскольку проценты напрямую в пикселы не переводятся. В итоге может получиться так, что общая ширина блока превысит ширину веб-страницы, что приведет к появлению горизонтальной полосы прокрутки. Выходов из подобной ситуации два — поменять алгоритм блочной модели и воспользоваться вложенными слоями.

Алгоритм блочной модели

Как уже упоминалось, ширина блока формируется из ширины контента и значений полей, границ и отступов. В браузере Internet Explorer в режиме совместимости (иными словами, когда не указан доктайп) алгоритм меняется автоматически и ширина всего блока устанавливается равной width. Остальные браузеры так просто не меняют алгоритм, к тому же вы знаете, что режим совместимости это зло. В CSS3 есть замечательное свойство box-sizing, которое нам и пригодится. При значении border-box ширина начинает включать поля и границы, но не отступы. Таким образом, подключая box-sizing со значением border-box к своему стиль, мы можем задавать ширину в процентах и спокойно указывать border и padding, не боясь, что изменится ширина блока. К сожалению, с этим свойством связана небольшая проблема, как обычно относящаяся к браузерам — не все браузеры его понимают. Радует, что браузеры хотя бы поддерживают специфические для каждого браузера свойства. В табл. 3.1 приведена поддержка браузерами.

Табл. 3.1. Поддержка браузерами свойства box-sizing

Браузер	Internet Explorer	Chrome	Opera	Safari	Firefox
Версия	8.0+	2.0+	7.0+	3.0+	1.0+
Свойство	box-sizing	-webkit-box-sizing	box-sizing	-webkit-box-sizing	-moz-box-sizing

Как видно из таблицы, в свойствах разброд и шатание, поэтому придется делать гибрид и указывать все три свойства (пример 3.4).

Пример 3.4. Ширина блока

XHTML 1.0 | CSS 3 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Ширина блока</title>
  <style type="text/css">
   div {
    width: 100%; /* Ширина */
    background: #fc0; /* Цвет фона */
    padding: 20px; /* Поля */
    -moz-box-sizing: border-box; /* Для Firefox */
    -webkit-box-sizing: border-box; /* Для Safari и Chrome */
    box-sizing: border-box; /* Для IE и Opera */
```

```
    }
   </style>
  </head>
  <body>
   <div>Ширина слоя 100%</div>
  </body>
</html>
```

Данный пример будет работать во всех браузерах, указанных в табл. 3.1, однако невалиден в CSS3 из-за применения нестандартных свойств начинающихся на -moz и -webkit. Ширина блока составляет 100% с учетом значений padding. Без свойства box-sizing в браузере появится горизонтальная полоса прокрутки.

Вложенные слои

Использование свойства box-sizing всем хорошо, кроме того, что не работает в старых версиях IE. Если вы верстаете сайт с учетом IE7 и IE6, вам подойдет старый проверенный метод с вложением слоев. Идея простая — для внешнего блочного элемента задается только необходимая ширина, а для вложенного блока всё остальное — поля, границы и отступы. Поскольку по умолчанию ширина блока равна доступной ширине родителя, получится, что блоки в каком-то смысле накладываются друг на друга, при этом фактическая ширина такого комбинированного элемента будет чётко задана. В примере 3.5 показано использование вложенных слоев.

Пример 3.5. Вложенные слои

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Ширина блока</title>
  <style type="text/css">
   .wrap {
    width: 50%; /* Ширина */
   }
   .wrap div {
    background: #fc0;
    margin: 10px;
    padding: 20px;
    border: 1px solid #000;
   }
  </style>
 </head>
 <body>
  <div class="wrap">
   <div>
    Ширина слоя 100%
   </div>
  </div>
 </body>
</html>
```

Результат данного примера показан на рис. 3.7.

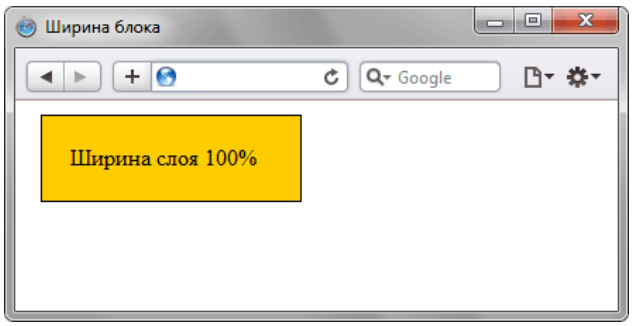


Рис. 3.7. Ширина блока в процентах

Преимуществом вложенных слоев является использование отступов (box-sizing их не учитывает), универсальность метода, также то, что фон по желанию можно добавлять к одному или другому слою. Тем самым несколько меняется внешний вид элементов, это особенно актуально при включении

фоновых рисунков. Из недостатков метода можно отметить включение дополнительного блока, который усложняет структуру кода, особенно при частом применении метода. Но это можно считать мелочью по сравнению с преимуществами.

Высота блока

На высоту блока действуют те же правила, что и на ширину. А именно, высота складывается из значений высоты контента (height), полей (padding), границ (border) и отступов (margin). Если свойство height не указано, то оно считается как auto, в этом случае высота контента вычисляется автоматически на основе содержимого. На рис. 3.8 показаны свойства, дающие итоговую высоту, которая обозначена чёрной пунктирной линией.

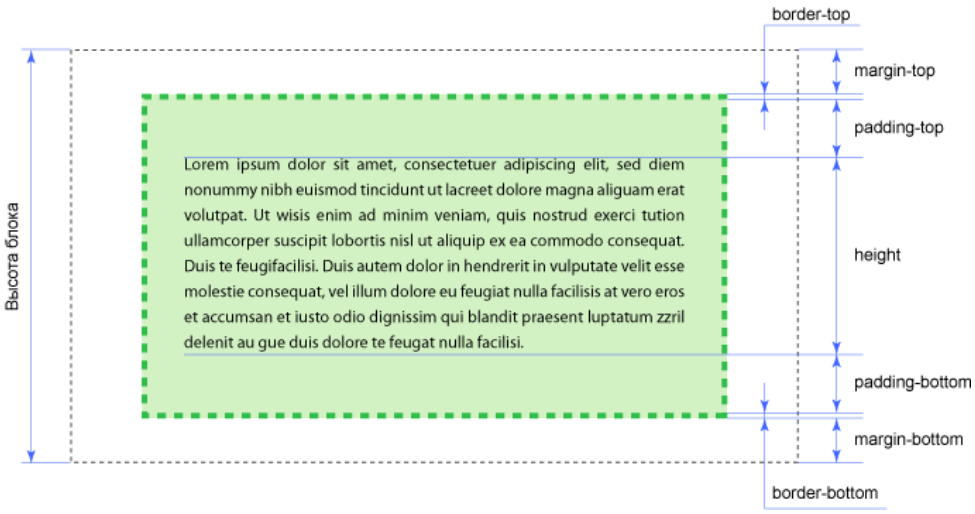


Рис. 3.8. Высота блока

Вместе с тем, несмотря на схожесть принципов построения ширины и высоты, у них есть существенные различия. Это касается того случая, когда значение width и height не указано, тогда по умолчанию оно принимается как auto. Для ширины блока — это максимально доступная ширина контента, а для высоты блока — это высота контента. Также для ширины блока известна ширина родителя, даже если она не указана явно. Это позволяет устанавливать значение width в процентах. Использование же процентов для height ни к чему не приведет, потому что высота родителя не вычисляется и её надо указывать. В примере 3.6 показано, как задать высоту блока в процентах.

Пример 3.6. Высота блока

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Высота блока</title>
  <style type="text/css">
   html, body {
    height: 100%; /* Высота родителя */
    margin: 0; /* Убираем отступы у страницы */
   }
   div {
    height: 100%; /* Высота */
    background: #fc0;
    margin: 10px;
    padding: 20px;
    border: 1px solid #000;
   }
  </style>
 </head>
 <body>
  <div>Высота слоя 100%</div>
 </body>
</html>
```

Результат данного примера показан на рис. 3.9.

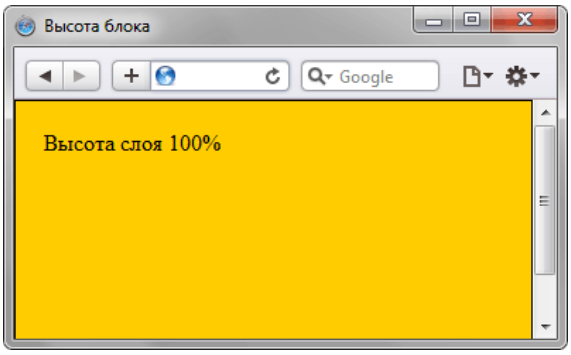


Рис. 3.9. Высота блока в процентах

Для тега ** ** в примере родителем выступает тег **<body>**, поэтому для него устанавливаем значение height равным 100%. В то же время на **<body>** действуют те же правила, что и на ** **, поэтому величина в процентах будет вычисляться не от высоты страницы, а от высоты контента. Так что для родителя **<body>**, которым является тег **<html>**, также требуется поставить значение height равным 100%. Только в этом случае высота блока в процентах будет зависеть от высоты страницы.

Поскольку на высоту влияет значение полей, границ и отступов, в примере появится вертикальная полоса прокрутки. Избавиться от этого влияния можно теми же методами, что и для высоты, а именно, использовать свойство box-sizing, либо воспользоваться вложенными слоями.

С высотой связана ещё одна особенность — при превышении содержимого блока его размеров при заданной высоте, содержимое начинается отображаться поверх блока (рис. 3.10).

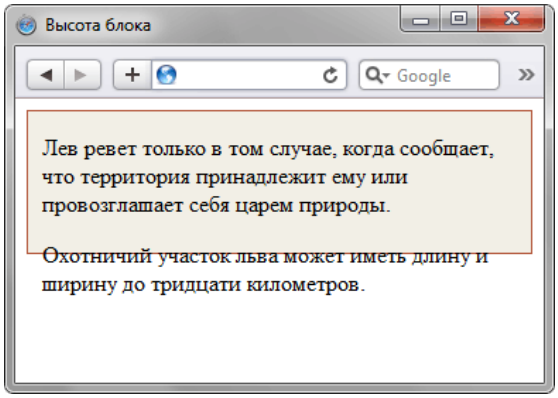


Рис. 3.10. Превышение размеров блока

Код, приводящий к подобному результату, приведен в примере 3.7.

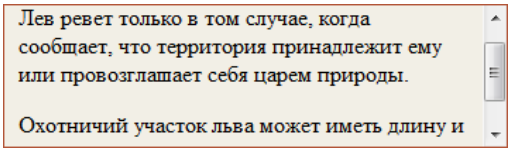
Пример 3.7. Превышение размеров блока

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

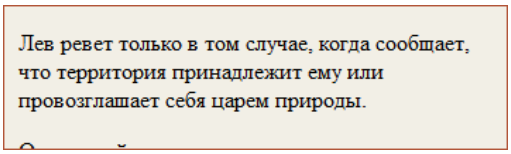
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  <html xmlns="http://www.w3.org/1999/xhtml">
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Высота блока</title>
  <style type="text/css">
   div {
    height: 100px;
    background: #F2EFE6;
    border: 1px solid #B25538;
    padding: 0 10px;
   }
  </style>
 </head>
 <body>
  <div>
   <p>Лев ревет только в том случае, когда сообщает, что
   территория принадлежит ему или провозглашает себя царем природы.</p>
   <p>Охотничий участок льва может иметь длину и ширину
   до тридцати километров.</p>
  </div>
```

```
 </body>
</html>
```

Чтобы избежать подобных неприятностей, высоту контента лучше не задавать, тогда высота блока будет вычисляться автоматически. Впрочем, бывают случаи, когда высота должна быть чётко указана, тогда рекомендуется к стилю добавить свойство overflow со значением auto или hidden. Результат у них разный, auto добавляет полосы прокрутки автоматически, когда они требуются (рис. 11а), hidden скрывает всё, что не помещается в заданные размеры (рис. 11б).



а. Значение auto



б. Значение hidden

Рис. 11. Использование свойства overflow

Фон

Если задать одновременно цвет фона и пунктирную границу блока, то становится заметно, что граница проходит внутри цветной области. Правда в разных браузерах наблюдается разночтение, в частности Internet Explorer до версии 7.0 включительно содержит ошибку, при которой фон выводится по внутреннему краю границы (рис. 12в). Начиная с версии 8.0 эта ошибка исправлена, и фон выводится по стандартам (рис. 12г). Браузеры Opera (рис. 12а), Firefox (12б), Safari и Chrome (рис. 12д) фон выводят правильно. Небольшие различия наблюдаются при отображении пунктирной рамки, но они не влияют на блочную модель.

а. Opera

б. Firefox

в. IE6, IE7

г. IE8, IE9

д. Safari, Chrome

Рис. 12. Отображение фона в браузере

В примере 3.8 показано, как создать код для получения подобной границы.

Пример 3.8. Фон и граница

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Рамка</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   .layer {
    width: 300px; /* Ширина слоя */
    padding: 10px; /* Поля вокруг текста */
    background: #fc0; /* Цвет фона */
    border: 5px dashed black;/* Параметры границы */
   }
  </style>
 </head>
 <body>
  <div class="layer">Опаньки...</div>
 </body>
</html>
```

Различия в подходе браузеров при рисовании границ заметны только на цветном фоне и пунктирных линиях. Для сплошной рамки вид блока в браузерах будет практически одинаковым.

Схлопывающиеся отступы

При рассмотрении блочной модели была затронута тема схлопывания отступов. Этот эффект наблюдается, когда у блочных элементов расположенных рядом друг с другом по вертикали, отступы не суммируются, а объединяются между собой. Само схлопывание действует на два и более блока (один может быть вложен внутрь другого) с отступами сверху или снизу, при этом примыкающие отступы комбинируются в один. Этот эффект работает только для блоков, у которых не заданы поля и границы. Для отступов слева и справа схлопывание никогда не применяется.

Несмотря на загадочность, схлопывание несёт в себе сугубо практическое значение и в первую очередь предназначено для корректного отображения текста. Расстояние между абзацами (тег **<p>**) без схлопывания увеличится в два раза, тогда как верхний отступ первого абзаца и нижний отступ последнего абзаца останутся неизменными. Схлопывание гарантирует, что расстояние в абзацах везде будет одинаковым.

Правила вычисления единого отступа меняются в зависимости от ряда условий, так, есть разница между положительным и отрицательным значением отступа, родительским и дочерним элементом. Далее перечислим типовые примеры.

Оба отступа положительны

Для положительных значений отступов выбирается наибольшее значение из двух отступов, и оно устанавливается как расстояние между блоками. На рис. 3.13 пунктирной линией выделены отступы у блоков и показано как в этом случае блоки устанавливаются относительно друг друга.

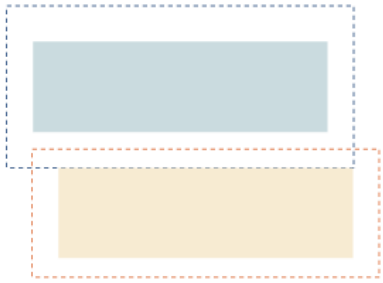


Рис. 3.13. Разные отступы у блоков

При одинаковых значениях отступов за расстояние между блоками принимается одно из них.

В следующем стиле у тега **<h1>** нижний отступ задаётся как 20 пикселов, а у **<p>** верхний отступ как 5 пикселов.

```
H1 {
  background: #F0BA7D;
  margin-bottom: 20px;
}
P {
  background: #CADADD;
  margin: 5px 0;
}
```

Значения отступов сравниваются между собой, и остаётся наибольшее число, расстояние между заголовком и абзацем текста принимается равным 20 пикселов (рис. 3.14).

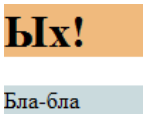


Рис. 3.14. Положительные отступы

Один из отступов отрицательный

В этом случае происходит складывание отступов по правилам математики:

```
x + (-y) = x – y
```

Здесь x и y величина прилегающих отступов элементов.

В следующем стиле у тега **<h1>** нижний отступ задаётся как 20 пикселов, а у **<p>** верхний отступ с отрицательным значением 10 пикселов.

```
H1 {
  background: #F0BA7D;
  margin-bottom: 20px;
}
P {
  background: #CADADD;
  margin: -10px 0 5px;
}
```

Из значения нижнего отступа тега **<h1>** отнимается значение верхнего отступа **<p>**, в итоге расстояние между заголовком и абзацем текста будет равно 10 пикселов (рис. 3.15).

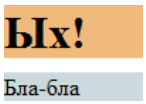


Рис. 3.15. Один отступ отрицательный

Если полученное значение в результате суммирования окажется отрицательным, то оно будет действовать на нижний блок, соответственно, он сдвинется вверх на указанное значение.

Оба отступа отрицательны

Из двух значений выбирается наибольшее по модулю, оно же и выступает в качестве отрицательного отступа между элементами. Так, если отступы равны -10px и -20px, то итоговое значение будет -20px.

В следующем стиле у тега **<h1>** нижний отступ задаётся как 1em, а у **<p>** верхний отступ с отрицательным значением 10 пикселов.

```
H1 {
  background: #F0BA7D;
  margin-bottom: -1em;
}
P {
  background: #CADADD;
  margin: -10px 0 5px;
}
```

При использовании разных единиц в отступах, браузер приводит их к одним единицам и сравнивает между собой. В данном случае 1em больше, чем 10px, поэтому текстовый абзац сдвинется вверх на 1em (рис. 3.16).

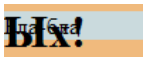


Рис. 3.16. Отрицательные отступы

Вложенные элементы

Предположим, что в нижнем блоке располагается дочерний элемент, у которого задан верхний отступ. Из блочной модели следует, что такой отступ сдвигает дочерний элемент вниз относительно верхнего края родителя. Однако с учётом схлопывающихся отступов результат будет иным. Отступ словно выйдет за пределы блока и будет задавать расстояние между верхним блоком и родительским элементом (рис. 3.17).

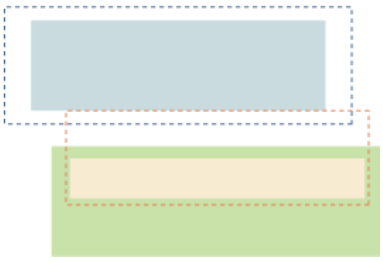


Рис. 3.17. Отступы с учётом дочернего элемента

В следующем стиле у тега **<h1>** нижний отступ задаётся как 20px, у ** ** верхний отступ как 30px, а у тега ** ** вложенного в ** ** верхний отступ как 50px.

```
H1 {
  background: #F0BA7D;
  margin-bottom: 20px;
}
DIV {
  background: #CADADD;
  margin-top: 30px;
}
P {
  background: #F4E7CF;
  margin: 50px 0 5px;
}
```

На элементы воздействует сразу три разных отступа. Чтобы сделать расчет итогового отступа в такой сложной ситуации, можно действовать последовательно. Вначале определяем отступ между **<h1>** и ** **. Значения оба положительны, поэтому берём большее, т.е. 30px. Далее сравниваем отступ от полученного значения и ** **. Оба значения положительны, при этом верхний отступ у ** ** наибольший, он и будет итоговым отступом между **<h1>** и ** ** (рис. 3.18).

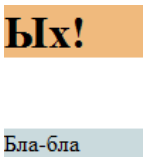


Рис. 3.18. Отступы с учётом вложения тегов

Схлопывание с вложенными тегами работает и без наличия верхнего элемента. Если оставить только конструкцию ** ... ** и для неё задать стиль выше, то ** ** сдвинется вниз на 50px, абзац останется неизменным, т.е. вид будет тот же, что представлен на рис. 3.18, только без верхнего заголовка.

Браузер Internet Explorer до версии 7.0 включительно не схлопывает отступы для элементов, у которых установлено свойство hasLayout. В частности, одним из способов установить это свойство, это задать ширину для блока (свойство width). Подробнее об этом смотрите главу посвященную IE.

Отмена схлопывания

Схлопывание не всегда требуется при вёрстке страницы, а в некоторых случаях вообще «ломает» дизайн. Поэтому следует знать, в каких случаях схлопывание работает, а в каких нет.

Схлопывание не срабатывает:

- для элементов, у которых установлено свойство padding.
- для элементов, у которых на стороне схлопывания задана граница;
- на элементах с абсолютным позиционированием, т.е. таких, у которых position установлено как

absolute;

- на плавающих элементах (для них свойство float задано как left или right);
- для строчных элементов;
- для <html>.

Также схлопывание не срабатывает при соблюдении некоторых условий:

- для элементов, у которых значение overflow задано как auto, hidden или scroll схлопывание не действует для их дочерних элементов;
- у элементов, к которым применяется свойство clear, не схлопывается верхний отступ с нижним отступом родительского элемента.

В качестве примера возьмём достаточно распространённый случай, когда нам требуется сделать два слоя, у которых задан только фоновый цвет, но нет полей и границ (пример 3.9).

Пример 3.9. Близлежащие блоки

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Отступ заголовка</title>
  <style type="text/css">
   .header {
    height: 100px;
    background: #F0BA7D;
   }
   .content {
    background: #CADADD;
   }
  </style>
 </head>
 <body>
  <div class="header"></div>
  <div class="content">
   <h1>Заголовок</h1>
   <p>Текст</p>
  </div>
 </body>
</html>
```

Хотя у слоёв не заданы отступы, между ними появится небольшой промежуток (рис. 3.19). Он возникает из-за действия отступа у дочернего элемента <h1>, верхний отступ у которого устанавливается по умолчанию.

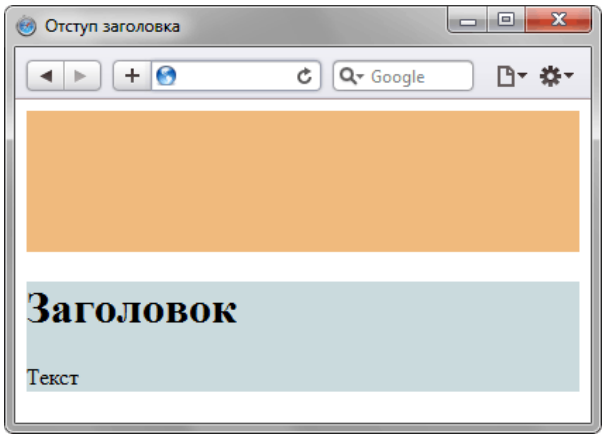


Рис. 3.19. Отступ между слоями

Для обнуления появившегося отступа, который нам на самом деле не нужен, есть разные пути. Поскольку схлопывание не работает для блоков с полями и границами, можно задать значение padding для слоя content. Главное, чтобы значение было больше нуля, подойдет даже 1px. Также добавление границы ко всем сторонам или только линии для верхнего края отменит схлопывание. Ещё один

способ — обнулить верхний отступ у <h1> и заменить его на padding-top при необходимости. Использование свойства overflow со значением auto также даст необходимый эффект. Ниже все эти методы сведены воедино.

```
.content {
 border-top: 1px solid #CADADD; /* Линия сверху, цвет совпадает с цветом фона
*/
 padding: 10px; /* Поля в блоке */
 overflow: auto;
}
.content h1  {
 margin-top: 0; /* Обнуляем верхний отступ */
 padding-top: 1em; /* Вместо отступа сверху добавляем поле */
}
```

Поскольку отступы довольно активно применяются в вёрстке, их схлопывание может оказать принципиальное значение на вид документа. Учитывайте этот эффект при формировании блоков.

Поток документа

В HTML формирование элементов на странице происходит сверху вниз согласно схеме документа. Слой, размещенный в самом верху кода, отобразится раньше слоя, который расположен в коде ниже. Такая логика позволяет легко прогнозировать результат вывода элементов и управлять им. Порядок вывода объектов на странице и называется «потоком». При этом существует несколько возможностей «вырвать» элемент из потока и придать ему почти мифические свойства. Раз он не существует в потоке, то в коде его можно описать где угодно, а также выводить в заданное место окна.

Использование position

Свойство position задает позиционирование элемента относительно исходного положения или родителя. Рассмотрим для начала значение fixed, которое выводит элемент из потока и привязывает к определенной точке окна. Особенностью fixed является фиксация слоя на одном месте, причем это положение не меняется при изменении размеров окна или прокрутке страницы. Такая особенность позволяет создавать неподвижные элементы интерфейса вроде кнопки «Оставить отзыв», как показано в примере 3.10. Добавления fixed недостаточно, нужно также задать положение элемента с помощью одного или двух свойств left, top, right, bottom, они управляют положением относительно окна браузера.

Пример 3.10. Фиксированный слой

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Фиксированный слой</title>
  <style type="text/css">
   BODY {
    margin-left: 90px; /* Отступ слева */
   }
   .feedback {
    position: fixed; /* Фиксированное положение */
    left: 0; top: 50%; /* Положение */
    background: #fc0; /* Цвет фона */
    width: 70px; /* Ширина слоя */
    padding: 5px; /* Поля */
   }
  </style>
 </head>
 <body>
  <p>Я вновь увидел Теночтитлан, великолепные покои во дворце Монтесумы и
  себя самого, спящего на золотом ложе. Я знал, что я бог Тескатлипока и
  наутро меня принесут в жертву. Я спал, измученный я удрученный, и видел
  сон. Я видел во сне, будто стою на том самом месте, где стоял сейчас, и
  запах наших цветов щекочет мне ноздри, как в эту ночь, и сладкие
  соловьиные песни звучат точно так же, как звучали они в моих ушах. Мне
  снилось, что, пока я стоял и слушал соловьев, над зелеными кронами дубов и
  ясеней взошла луна, — вот, вот она уже сияет в небесах. Мне снилось, что
  чей-то голос запел за холмом, но тут я пробудился от давно забытых видений
  прошлого.</p>
  <div class="feedback">Оставить<br />отзыв</div>
 </body>
</html>
```

Результат данного примера показан на рис. 3.20. Чтобы текст не накладывался на слой, добавлен отступ слева. Из-за того что слой `feedback` не существует в потоке документа, он может располагаться в любом месте кода, в данном случае внизу.

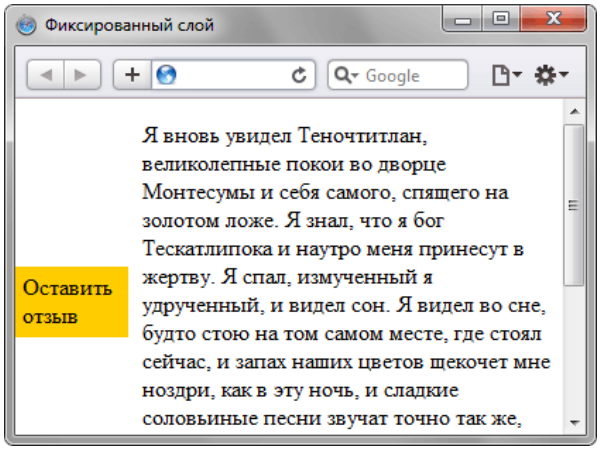


Рис. 3.20. Фиксированный слой

Значение absolute свойства position также выводит элемент из потока, но в отличие от fixed слой прокручивается вместе с содержимым. Кроме этого, существенное влияние на позицию оказывают свойства родителя. Возможны два основных варианта.

1. Родитель отсутствует (его роль выполняет **<body>**) или у родителя не задано свойство position.
2. У родителя элемента установлено свойство position в значении absolute или relative.

В первом случае все просто, элемент ведет себя как в примере 3.10, за исключением того, что не закрепляется строго на одном месте, а может прокручиваться одновременно со страницей. Во втором варианте положение элемента задается относительно родителя.

В примере 3.11 создается два вложенных слоя с абсолютным позиционированием. Координаты первого слоя `layer1` задаются относительно окна браузера, координаты второго слоя `layer2` относительно границ слоя `layer1`.

Пример 3.11. Абсолютное позиционирование

XHTML 1.0 | CSS 2.1 | CSS 3 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Позиционирование</title>
  <style type="text/css">
   .layer1 {
    width: 217px; /* Ширина слоя */
    height: 512px; /* Высота слоя */
    background: url(images/bg2.png) no-repeat; /* Фоновая картинка */
    position: absolute; /* Абсолютное позиционирование */
    right: 20%; /* Положение справа */
    bottom: 0; /* Положение снизу */
   }
   .layer2 {
    background: #000; /* Цвет фона для IE */
    background: rgba(0,0,0, 0.5); /* Цвет фона с полупрозрачностью */
    color: #fff; /* Цвет текста */
    position: absolute; /* Абсолютное позиционирование */
    bottom: 30px; /* Положение снизу */
    width: 207px; /* Ширина слоя */
    padding: 5px; /* Поля */
   }
  </style>
 </head>
 <body>
  <div class="layer1">
   <div class="layer2">
     Позвоночник человека
   </div>
  </div>
 </body>
</html>
```

Результат данного примера показан на рис. 3.21. Браузер IE до версии 9.0 не поддерживает формат RGBA для цвета.

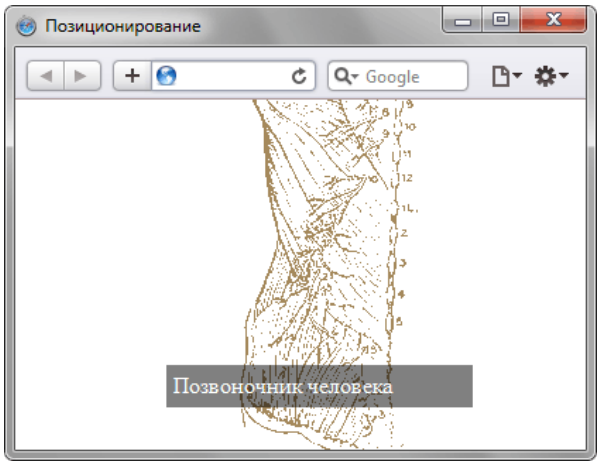


Рис. 3.21. Положение слоя на странице

Обратите внимание, что абсолютно позиционированный элемент может выходить за верхний и левый край окна браузера, при этом не возникает полос прокрутки. Также при использовании свойства position ширина слоя автоматически приравнивается ширине контента плюс, как обычно, ширина отступов, границ и полей.

Использование float

Свойство float превращает элемент в плавающий, при этом он прижимается к левому или правому краю родителя, а текст его обходит с других сторон. Такое поведение текста напоминает поток воды, обтекающий камень, поэтому элементы с таким поведением и называются плавающими. В отличие от абсолютно позиционированных плавающие элементы не обладают властью над координатами, их не получится заставить встать в точно заданную точку, но они имеют ряд примечательных характеристик. Настолько, что свойство float для вёрстки стало использоваться повсеместно. Перечислим лишь ряд возможных приложений:

- галереи небольших фотографий;
- двух и трехколоночные макеты;
- горизонтальные меню;
- иллюстрации в тексте;
- многоколоночный текст.

float может добавляться к изображениям (тег **<img>**), блочным элементам вроде **<p>**, ** ** и к строчным элементам (****, **<a>**, ****). В примере 3.12 показано использование float для абзаца с фотографией.

Пример 3.12. Плавающий элемент

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Обтекание</title>
  <style type="text/css">
   .right {
    float: right;
   }
  </style>
 </head>
 <body>
  <p class="right"><img src="images/cat.jpg" alt="Кошка" /></p>
  <p>Сфотографировано на камеру Canon EOS 50D. Объектив 24-105,
  диафрагма 4.0, выдержка 1/500, чувствительность ISO 400</p>
 </body>
</html>
```

Результат примера показан на рис. 3.22.

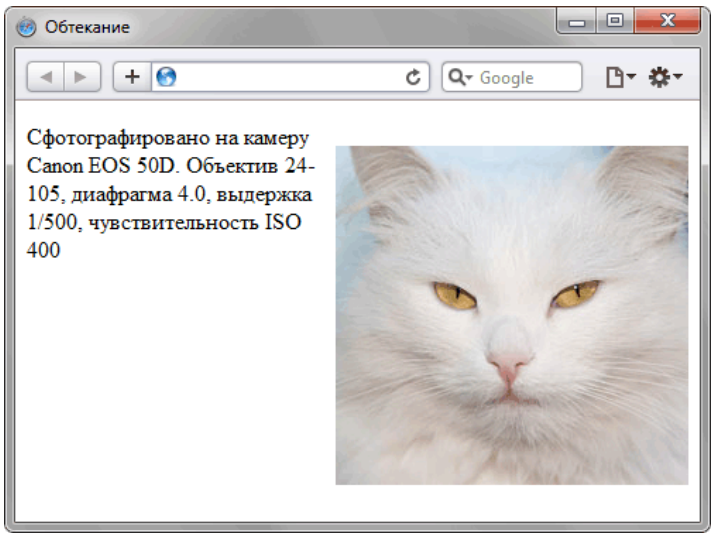


Рис. 3.22. Плавающий элемент

Свойство float лишь частично влияет на элемент в потоке. Можно управлять положением по горизонтали, меняя значение float с right на left, но по вертикали положение элемента задается его местом в коде.

Блочные элементы

Блочным называется элемент, который отображается на веб-странице в виде прямоугольника. Такой элемент занимает всю доступную ширину, высота элемента определяется его содержимым, и он всегда начинается с новой строки. К блочным элементам относятся теги **<address>**, **<blockquote>**, ** **, **<fieldset>**, **<form>**, **<h1>**,...,**<h6>**, **<hr>**, **<ol>**, **<p>**, **<pre>**, **<table>**, **<ul>** и некоторые устаревшие. Также блочным становится элемент, если в стиле для него свойство display задано как block, list-item, table и в некоторых случаях run-in.

Для блочных элементов характерны следующие особенности.

- Блоки располагаются по вертикали друг под другом.
- На прилегающих сторонах элементов действует эффект схлопывания отступов.
- Запрещено вставлять блочный элемент внутрь строчного. Например, **<a><h1>**Заголовок**</h1>** не пройдёт валидацию, правильно вложить теги наоборот — **<h1><a>**Заголовок**</h1>**.
- По ширине блочные элементы занимают всё допустимое пространство.
- Если задана ширина контента (свойство width), то ширина блока складывается из значений width, полей, границ, отступов слева и справа.
- Высота блочного элемента вычисляется браузером автоматически, исходя из содержимого блока.
- Если задана высота контента (свойство height), то высота блока складывается из значения height, полей, границ, отступов сверху и снизу. При превышении указанной высоты контент отображается поверх блока.
- На блочные элементы не действуют свойства, предназначенные для строчных элементов, вроде vertical-align.
- Текст по умолчанию выравнивается по левому краю.

Не все блочные теги допустимо вкладывать один в другой, поэтому при создании структуры кода активную роль выполняет ** ** как универсальный кирпичик вёрстки. Тег ** ** допустимо вкладывать один в другой, другие блочные элементы также можно помещать внутрь ** **. В примере 3.13 показан фрагмент кода сайта CSS Zen Garden, где активно применяются эти теги для формирования требуемой конструкции.

**Пример 3.13. Использование тега **

```
<div id="container">
  <div id="intro">
    <div id="pageHeader">
      <h1><span>css Zen Garden</span></h1>
      <h2><span>The Beauty of <acronym
          title="Cascading Style  Sheets">CSS</acronym> Design</span></h2>
    </div>
    <div id="quickSummary">
      <p class="p1"><span>A demonstration of what can be accomplished
      visually through <acronym title="Cascading Style Sheets">CSS</acronym>
      - based design. Select any style sheet from the list to load it into
      this page.</span></p>
    </div>
  </div>
</div>
```

Если в одном блочном элементе содержатся строчные элементы наряду с блочными, то вокруг строчных элементов генерируется анонимный блок. Для таких блоков применяется стиль по умолчанию, а также наследуемый стиль, заданный для его родителя. В примере 3.14 внутри ** ** располагается два тега **<p>** и рядовой текст. Поскольку этот элемент анонимный, напрямую только к

нему одному нельзя применить стили.

Пример 3.14. Анонимный блок

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Анонимный блок</title>
 </head>
 <body>
  <div>
   <p>Первый абзац</p>
   Анонимный блок
   <p>Второй абзац</p>
  </div>
 </body>
</html>
```

Схематично блоки в данном примере показаны на рис. 3.23. Серым цветом выделены блочные теги **<p>**, а пунктиром — анонимный блок.

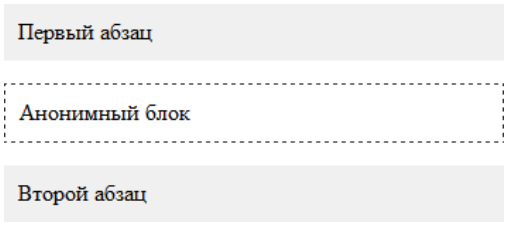


Рис. 3.23. Блоки в документе

Преобразование в блочный элемент

В некоторых случаях требуется наделить строчный элемент характеристиками блочного. Так, превращение ссылки в блок позволяет увеличить полезную площадь ссылки за счёт использования свойств width и height. В примере 3.15 показано создание меню, в котором ссылками является вся доступная область.

Пример 3.15. Меню

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Меню</title>
  <style type="text/css">
   .menu {
    width: 220px; /* Ширина меню */
    padding: 5px; /* Отступы от рамки до пунктов меню */
    border: 1px solid #000; /* Рамка вокруг меню */
   }
   .menu P { margin: 0 0 2px; }
   .menu A {
    padding: 2px; /* Отступ от рамки вокруг ссылки до текста */
    display: block; /* Ссылка как блочный элемент */
    border: 1px solid #fff; /* Маскируем рамку вокруг ссылки */
    text-decoration: none; /* Убираем подчеркивание у ссылок */
   }
   .menu A:hover {
    background: #faf3d2; /* Цвет фона под ссылкой */
    color: #800000; /* Новый цвет ссылки */
    border: 1px dashed #634f36 /* Рамка вокруг ссылки */
   }
  </style>
 </head>
 <body>
  <div class="menu">
   <p><a href="l.html">Метод простых итераций</a></p>
   <p><a href="2.html">Метод случайных чисел</a></p>
   <p><a href="3.html">Метод дихотомии</a></p>
   <p><a href="4.html">Метод золотого сечения</a></p>
  </div>
```

```
 </body>
</html>
```

Вся строка выступает ссылкой, поэтому при наведении на неё курсора происходит изменение стиля (рис. 3.24).

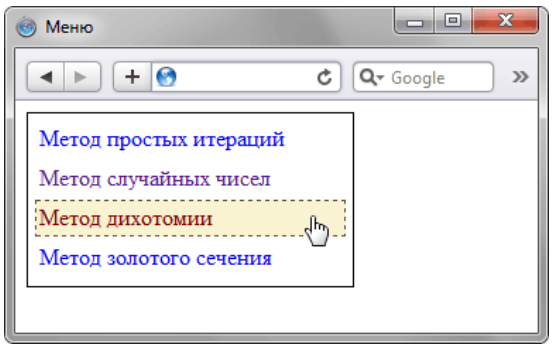


Рис. 3.24. Ссылка как блочный элемент

Следует понимать, что превращение элемента в блочный не даёт право нарушать последовательность вложения тегов и вкладывать блочные элементы внутрь строчных, даже если через стили они установлены блочными.

Строчные элементы

Строчными называются такие элементы документа, которые являются непосредственной частью строки. К строчным элементам относятся теги **<img>**, ****, **<a>**, **<q>**, **<code>** и др., а также элементы, у которых свойство display установлено как inline. В основном они используются для изменения вида текста или его логического выделения.

По аналогии с блочными элементами перечислим их характерные особенности.

- Внутрь строчных элементов допустимо помещать текст или другие строчные элементы. Вставлять блочные элементы внутрь строчных запрещено.
- Эффект схлопывания отступов не действует.
- Свойства, связанные с размерами (width, height) не применимы.
- Ширина равна содержимому плюс значения отступов, полей и границ.
- Несколько строчных элементов идущих подряд располагаются на одной строке и переносятся на другую строку при необходимости.
- Можно выравнивать по вертикали с помощью свойства vertical-align.

Строчные элементы удобно использовать для изменения вида и стиля текста, в частности, отдельных символов и слов. Для этой цели обычно применяется универсальный тег ****, который самостоятельно никак не модифицирует содержимое, но легко объединяется со стилями через классы или идентификаторы. За счёт чего с помощью этого тега можно легко управлять видом и положением отдельных фрагментов текста или рисунков.

Для вёрстки строчные элементы применяются реже, чем блочные. Это связано в основном с тем, что внутрь строчных элементов не допускается вкладывать контейнеры ** **, **<p>** и подобные широко распространённые теги. Тем не менее, блочные и строчные элементы удачно дополняют друг друга, поскольку позволяют на всех уровнях менять вид составляющих веб-страниц. В примере 3.16 показано использование тега **** для выделения отдельных слов.

**Пример 3.16. Применение тега **

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Строчные элементы</title>
  <style type="text/css">
   .pose {
    background: #fc0; /* Цвет фона */
    margin-left: 1em; /* Отступ слева */
   }
   .press {
    padding: 1px; /* Поля вокруг текста */
    border: 1px dotted maroon; /* Параметры рамки */
    color: navy; /* Цвет текста */
   }
   .num {
    font-weight: bold; /* Жирное начертание */
    color: maroon; /* Цвет текста */
   }
  </style>
 </head>
 <body>
   <p><span class="pose">Лягте животом на пол</span>.
    Соедините стопы вместе, согнув ноги в коленях и развернув их в стороны.
    Руки за головой. Тяните голову руками вверх и вперед до полного
    сокращения <span class="press">мышц живота</span>. Задержитесь на две
    секунды. Выполните <span class="num">восемь</span> повторений.</p>
 </body>
</html>
```

Результат примера показан ниже (рис. 3.25).

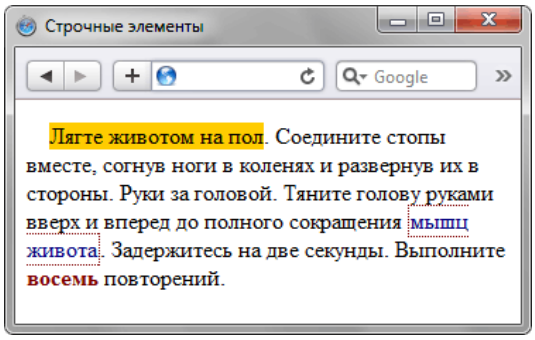


Рис. 3.25. Текст, оформленный с помощью стилей

В данном примере тег **** и стили используются для выделения различными способами фрагментов текста. В частности, выделение происходит за счет фонового цвета, рамки вокруг текста и сменой его цвета. Обратите внимание, что в результате переноса текста рамка вокруг него также перенеслась на другую строку. Во многих случаях это получается не очень красиво, поэтому можно запретить перенос текста через white-space: nowrap.

Для текста, который не обрамлен строчным тегом вроде **** будет создан анонимный строчный элемент. Для примера выше схема элементов будет следующей (рис. 3.26).

Рис. 3.26. Анонимные строчные элементы

Серым цветом на рисунке помечен текст внутри тега ****, а пунктиром анонимные строчные элементы. Для таких элементов задать стиль напрямую нельзя, он наследуется от родителя.

Преобразование в строчный элемент

Строчные элементы можно превращать в блочные с помощью свойства display и его значения block. Также возможно и обратное действие через значение inline (пример 3.17).

Пример 3.17. Свойство display

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Строчные элементы</title>
  <style type="text/css">
   .notetitle {
    border: 1px solid black; /* Параметры рамки */
    border-bottom: none; /* Убираем границу снизу */
    padding: 3px; /* Поля вокруг текста */
    display: inline; /* Устанавливаем как строчный элемент */
    background: #ffeebf; /* Цвет фона */
    font-weight: bold; /* Жирное начертание */
    font-size: 0.9em; /* Размер шрифта */
    margin: 0; /* Убираем отступы */
    white-space: nowrap; /* Запрещены переносы текста */
   }
   .note {
    border: 1px solid #634f36; /* Параметры рамки */
    background: #f3f0e9; /* Цвет фона */
    padding: 7px; /* Поля вокруг текста */
    margin: 0 0 1em 0; /* Значение отступов */
   }
  </style>
 </head>
 <body>
  <p class="notetitle">Примечание</p>
  <p class="note">Исходя из различных критериев, основными из
   которых являются показатели целесообразности и эффективности приложенных
   усилий, можно однозначно сказать следующее. А именно, что достижение
   желаемых результатов требует гибкого подхода, основанного на опыте и
```

```
    глубоком понимании смысла вышеизложенного.</p>
  </body>
</html>
```

Результат примера приведен на рис. 3.27.

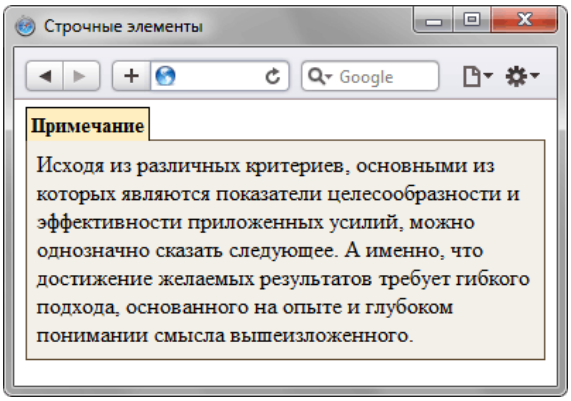


Рис. 2. Замена блочного элемента на строчный

В данном примере блочный тег **<p>** отображается на веб-странице как строчный элемент. Это требуется для того, чтобы ширина фона и рамки равнялась ширине самого текста с учетом полей. Как известно, ширина блочных элементов не зависит от ширины содержимого, поэтому и приходится представлять тег **<p>** в виде строчного элемента. В принципе, аналогичным решением будет использовать вместо **<p>** тег ****.

Учтите, что преобразование элемента в строчный не даёт право нарушать последовательность вложения тегов и вкладывать блочные элементы внутрь строчных.

Строчно-блочные элементы

Блочные и строчные элементы отлично дополняют друг друга при вёрстке, занимая каждый свою определённую нишу. Но возникают случаи, когда характеристик этих элементов явно недостаточно. Галерея фотографий, представленная на рис. 3.28 состоит из секций, в которые входит изображение с подписью к нему, при этом секции выстраиваются по горизонтали, занимая всю доступную ширину. При уменьшении окна браузера секции переходят на другую строку.

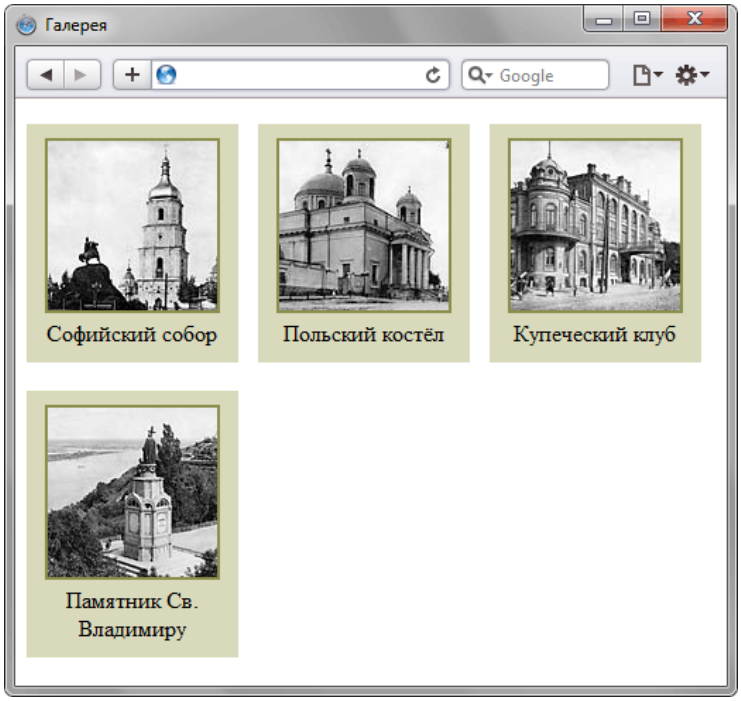


Рис. 3.28. Галерея фотографий

Если для формирования секций использовать тег `<div>`, как блочный элемент он будет каждый раз начинаться с новой строки. Для строчных элементов нельзя задать цвет фона всей секции и установить её размеры. Наиболее популярное решение в подобных случаях это использование свойства float, которое будет рассмотрено в следующем разделе. Пока же остановимся на строчно-блочных элементах, которые сочетают преимущества строчных и блочных элементов.

В HTML нет тега, который относится к строчно-блочным элементам, его можно определить, задав элементу свойство display со значением inline-block.

```
div {
 display: inline-block;
}
```

Характеристики этих элементов следующие.

- Внутрь строчно-блочных элементов допустимо помещать текст, строчные или блочные элементы.
- Высота элемента вычисляется браузером автоматически, исходя из содержимого блока.
- Ширина равна содержимому плюс значения отступов, полей и границ.
- Несколько элементов идущих подряд располагаются на одной строке и переносятся на другую строку при необходимости.
- Можно выравнивать по вертикали с помощью свойства vertical-align.
- Разрешено задавать ширину и высоту.
- Эффект схлопывания отступов не действует.

Чтобы создать галерею, представленную на рис. 3.28 для тега ** ** следует задать значение display как inline-block, а внутрь него добавить изображение и подпись к нему через тег **<p>** (пример 3.18).

Пример 3.18. Использование display

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Галерея</title>
  <style type="text/css">
   .photo {
    background: #d9dabb; /* Цвет фона */
    width: 150px; /* Ширина */
    margin: 0 10px 10px 0; /* Отступы */
    padding: 10px 0; /* Поля сверху и снизу */
    text-align: center; /* Выравнивание по центру */
    display: inline-block; /* Строчно-блочный элемент */
   }
   .photo img {
    border: 2px solid #8b8e4b; /* Параметры рамки */
   }
   .photo p {
    margin: 0; /* Отступы */
   }
  </style>
 </head>
 <body>
  <div class="photo">
   <p><img src="images/thumb1.jpg" alt="" /></p>
   <p class="caption">Софийский собор</p>
  </div>
  <div class="photo">
   <p><img src="images/thumb2.jpg" alt="" /></p>
   <p class="caption">Польский костёл</p>
  </div>
  <div class="photo">
   <p><img src="images/thumb3.jpg" alt="" /></p>
   <p class="caption">Купеческий клуб</p>
  </div>
  <div class="photo">
   <p><img src="images/thumb4.jpg" alt="" /></p>
   <p class="caption">Памятник Св. Владимиру</p>
  </div>
 </body>
</html>
```

Поскольку все фотографии имеют одинаковый размер, ширина блока задана явно и равна 150px, но высота не указывается, поэтому при длинной подписи к рисунку высота секций будет различаться (рис. 3.29).

Рис. 3.29. Разная высота секций

Это не является проблемой, поскольку в любом случае секции будут выводиться упорядоченно рядами, несмотря на разную высоту. При этом можно изменить вид выравнивания через vertical-align, добавив это свойство к классу `photo`. Если указать значение top, то поменяется отображение секций (рис. 3.30).

Рис. 3.30. Выравнивание по верхнему краю

Основной проблемой выступает браузер IE до версии 7.0 включительно. Этот браузер применяет значение inline-block только для строчных элементов и с блочными элементами работает некорректно. Чтобы убедить этот браузер правильно отображать наш пример, необходимо вместо inline-block использовать значение inline и установить свойство hasLayout. Добавление inline разрушит макет в остальных браузерах, поэтому правильным решением будет воспользоваться условными комментариями (пример 3.19).

Пример 3.19. Стиль для IE

```
<style type="text/css">
  /* Стиль из примера 3.18 */
</style>
<!--[if lte IE 7]>
 <style type="text/css">
  .photo {
   display: inline; /* Строчный элемент */
   zoom: 1; /* Устанавливаем hasLayout */
  }
 </style>
<![endif]-->
```

Конструкция **[if lte IE 7]** означает применить указанный код для браузера IE версии 7.0 и ниже. Остальные браузеры воспринимают её как комментарий и игнорируют. Что касается свойства zoom, оно нестандартное и предназначено для установки свойства hasLayout, напрямую которое задавать нельзя. Подробнее об этом свойстве смотрите главу 5 посвященную Internet Explorer.

Также строчно-блочные элементы удобно использовать для различных каталогов товаров, которые встречаются в интернет-магазинах. Обычно они выводятся с картинкой и подписью к ней. Всё это похоже на галерею, сделанную выше, поэтому остаётся только модифицировать её код, учесть выравнивание по высоте и поведение IE (пример 3.20).

Пример 3.20. Каталог товаров

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Каталог</title>
  <style type="text/css">
   BODY { font: 10pt Arial, Helvetica, sans-serif; }
   #catalog A { color: #666; }
   #catalog A:hover { color: #1fa0e2; }
   #catalog DIV {
    width: 110px; /* Ширина */
    margin: 0 5px 15px 0; /* Отступы */
    text-align: center; /* Выравнивание по центру */
    display: inline-block; /* Строчно-блочный элемент */
    vertical-align: top; /* Выравнивание по верхнему краю */
   }
   #catalog P { margin: 0 5px; }
   #catalog SPAN { color: #ccc; font-size: 0.8em; }
  </style>
  <!--[if lte IE 7]>
  <style type="text/css">
   #catalog DIV {
    display: inline; /* Строчный элемент */
    zoom: 1; /* Устанавливаем hasLayout */
   }
  </style>
  <![endif]-->
```

```
 </head>
 <body>
  <div id="catalog">
   <div>
    <p><img src="images/category1.jpg" alt="" /></p>
    <p><a href="#">Видео</a> <span>1856</span></p>
   </div>
   <div>
    <p><img src="images/category2.jpg" alt="" /></p>
    <p><a href="#">Фото</a> <span>315</span></p>
   </div>
   <div>
    <p><img src="images/category3.jpg" alt="" /></p>
    <p><a href="#">Мобильные устройства</a> <span>2109</span></p>
   </div>
   <div>
    <p><img src="images/category4.jpg" alt="" /></p>
    <p><a href="#">Компьютеры и орг.техника</a> <span>4296</span></p>
   </div>
   <div>
    <p><img src="images/category5.jpg" alt="" /></p>
    <p><a href="#">Бытовая техника</a> <span>731</span></p>
   </div>
  </div>
 </body>
</html>
```

Результат данного примера показан на рис. 3.31.

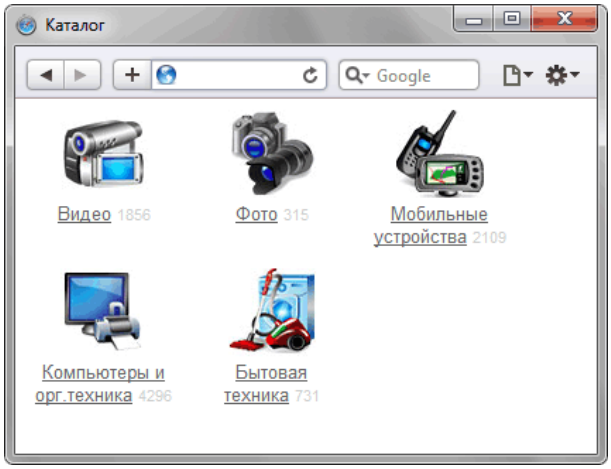


Рис. 3.31. Каталог товаров

Чтобы не задавать каждому тегу ** **, которых может быть довольно много, свой класс, в примере введён ** ** с идентификатором `catalog` и стиль применяется к тегам внутри него.

Плавающие элементы

Плавающими будем называть такие элементы, которые обтекаются по контуру другими объектами веб-страницы, например, текстом. Правильнее говорить «обтекаемые элементы», но с другой стороны термин «плавающий элемент» давно уже прижился, так что его и буду использовать в дальнейшем.

Плавающие элементы достаточно активно применяются при вёрстке веб-страниц и служат для реализации этих и не только задач:

- обтекание картинок текстом;
- создание врезок;
- горизонтальные меню;
- колонки.

Обтекание происходит с помощью стилевого свойства float со значением left или right. По умолчанию обтекание для элементов не устанавливается, но если это по каким-либо причинам необходимо указать явно, следует использовать значение none. На рис. 3.32 показан результат применения разных значений на изображение рядом с текстом.


Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat volutpat. Ut wisis enim ad minim veniam, quis nostrud exerci tution ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

а. Обтекания нет или float: none


Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat volutpat. Ut wisis enim ad minim veniam, quis nostrud exerci tution ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

б. Для картинки установлено float: left

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat volutpat. Ut wisis enim ad minim veniam, quis nostrud exerci tution ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.


в. Для картинки установлено float: right

Рис. 3.32. Обтекание картинки текстом

Сам HTML-код остаётся практически неизменным и привёден в примере 3.21.

Пример 3.21. Использование float

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>float</title>
 </head>
 <body>
  <div>
   <img src="images/figure.jpg" style="float: left" />
   Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
   sed diem nonummy nibh euismod tincidunt ut lacreet dolore
   magna aliguam erat volutpat. Ut wisis enim ad minim veniam,
   quis nostrud exerci tution ullamcorper suscipit lobortis nisl
   ut aliquip ex ea commodo consequat.
  </div>
 </body>
</html>
```

В данном примере показано добавление свойства float со значением left к тегу **<img>**. Это значение выравнивает элемент по левому краю родителя или другого плавающего элемента, а все остальные элементы, вроде текста, обтекают его по правой стороне. Значение right, наоборот, выравнивает элемент по правому краю родителя или другого плавающего элемента, а все остальные элементы, вроде текста, обтекают его по левой стороне.

Итак, с терминологией разобрались. Давайте дополним пример 3.21, чтобы рисунок лучше гармонировал с текстом. Самое главное это задать отступ справа от изображения и на всякий случай снизу. Для этого к тегу **<img>** добавляется класс с именем `fig`, у которого установлено свойство margin, оно одновременно определяет свой отступ на каждой стороне изображения (пример 3.22).

Пример 3.22. Обтекание картинки

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Рисунок по левому краю</title>
  <style type="text/css">
   .fig {
    float: left; /* Выравнивание рисунка по левому краю */
    background: #e69f73; /* Цвет фона */
    padding: 4px; /* Поля вокруг картинки */
    margin: 0 10px 5px 0; /* Отступы */
   }
  </style>
 </head>
 <body>
  <p><img src="images/figure.jpg" class="fig" />
   Винни-Пух был всегда не прочь немного подкрепиться, в
   особенности часов в одиннадцать утра, потому что в это время
   завтрак уже давно окончился, а обед еще и не думал начинаться.
   И, конечно, он страшно обрадовался, увидев, что Кролик достает
   чашки и тарелки.</p>
 </body>
</html>
```

Результат примера показан на рис. 3.33.

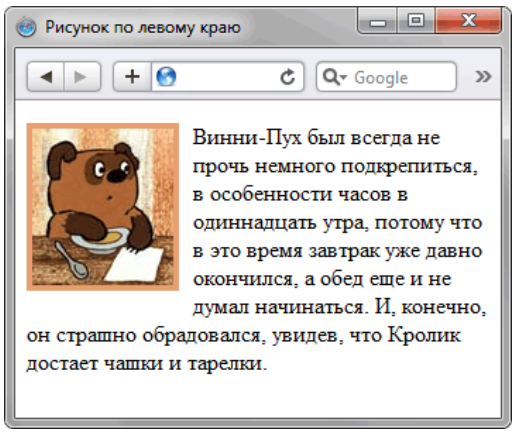


Рис. 3.33. Рисунок с выравниванием по левому краю и обтеканием по правому

Создание врезок

Врезкой называется блок с рисунками и текстом, который встраивается в основной текст. Врезка обычно располагается по левому или правому краю текстового блока, а основной текст обтекает её с других сторон (рис. 3.34).

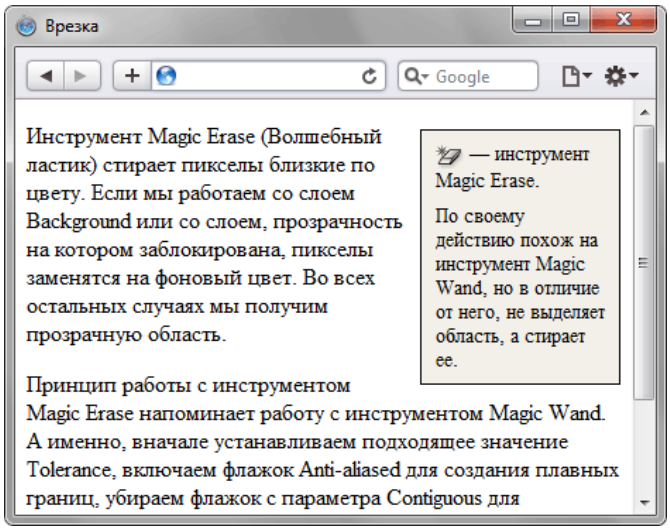


Рис. 3.34. Вид врезки

Чтобы врезка выделялась в тексте, у нее обычно устанавливают фоновый цвет и добавляют рамку. По своему виду врезки напоминают приведенный выше способ обтекания текстом картинки, поэтому код для создания врезок практически идентичен предыдущему (пример 3.23).

Пример 3.23. Добавление врезки

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Врезка</title>
  <style type="text/css">
   .note {
    float: right; /* Выравнивание по правому краю */
    background: #F3F0E9; /* Цвет фона */
    border: 1px solid #000; /* Параметры границы */
    padding: 0 10px; /* Поля вокруг картинки */
    margin: 5px 0 5px 10px; /* Отступы */
    width: 120px; /* Ширина */
    font-size: 0.9em; /* Размер шрифта */
   }
   .note P { margin: 0.6em 0; }
   .note IMG {
    vertical-align: middle; /* Выравнивание по центру строки */
   }
  </style>
 </head>
 <body>
  <div class="note">
   <p><img src="images/tool_magicerase.gif" width="20" height="17"
     alt="Magic Erase" /> &#8212; инструмент Magic Erase.</p>
   <p>По своему действию похож на инструмент Magic Wand, но в отличие от
     него, не выделяет область, а стирает ее.</p>
  </div>
  <p>Инструмент Magic Erase (Волшебный ластик) стирает пикселы близкие по
  цвету. Если мы работаем со слоем Background или со слоем, прозрачность
  на котором заблокирована, пикселы заменятся на фоновый цвет. Во всех
  остальных случаях мы получим прозрачную область.</p>
  <p>Принцип работы с инструментом Magic Erase напоминает работу с
  инструментом Magic Wand. А именно, вначале устанавливаем подходящее
  значение Tolerance, включаем флажок Anti-aliased для создания плавных
  границ, убираем флажок с параметра Contiguous для одновременного удаления
  всего фона, после чего щелкаем по фотографии в районе неба. Если слой
  Background был предварительно переименован, мы получим прозрачные
  участки в нужных местах.</p>
 </body>
</html>
```

При создании врезок следует обязательно указывать её ширину с помощью свойства width. Иначе размер слоя окажется гораздо шире, чем это требуется.

Расположение слоев по горизонтали

По умолчанию блочные элементы выстраиваются по вертикали один под другим, но при помощи свойства float их можно заставить располагаться рядом по горизонтали. При этом требуется установить ширину слоев и задать для них float. Если ширина не указана, она будет равна содержимому слоя с учётом полей и границ. В примере 3.24 взят каталог товаров, созданный в предыдущем разделе с помощью строчно-блочных элементов, и переделан под использование float. Чтобы блоки были заметны, фон веб-страницы установлен серым.

Пример 3.24. Блоки по горизонтали

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Каталог</title>
  <style type="text/css">
   BODY {
    font: 10pt Arial, Helvetica, sans-serif;
    background: #f0f0f0; /* Цвет фона веб-страницы */
   }
   #catalog A { color: #666; }
   #catalog A:hover { color: #1fa0e2; }
   #catalog DIV {
    width: 110px; /* Ширина */
    background: #fff;
    margin: 0 5px 15px 0; /* Отступы */
    text-align: center; /* Выравнивание по центру */
    float: left; /* Выравнивание по левому краю */
   }
   #catalog P { margin: 0 5px; }
   #catalog SPAN { color: #ccc; font-size: 0.8em; }
  </style>
 </head>
 <body>
  <div id="catalog">
   <div>
    <p><img src="images/category4.jpg" alt="" /></p>
    <p><a href="#">Компьютеры и орг.техника</a> <span>4296</span></p>
   </div>
   <div>
    <p><img src="images/category3.jpg" alt="" /></p>
    <p><a href="#">Мобильные устройства</a> <span>2109</span></p>
   </div>
   <div>
    <p><img src="images/category2.jpg" alt="" /></p>
    <p><a href="#">Фото</a> <span>315</span></p>
   </div>
   <div>
    <p><img src="images/category1.jpg" alt="" /></p>
    <p><a href="#">Видео</a> <span>1856</span></p>
   </div>
  </div>
 </body>
</html>
```

Результат примера показан на рис. 3.35.

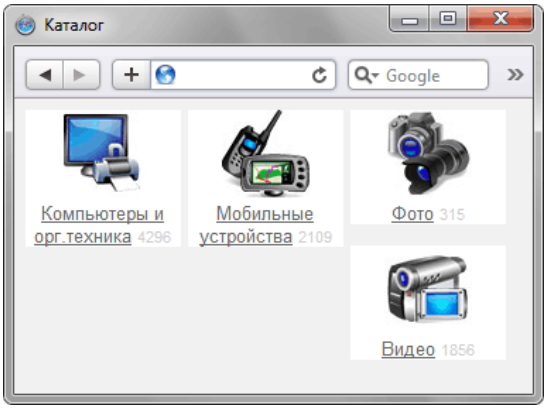


Рис. 3.35. Расположение слоев при использовании свойства float

Из-за разного текста в подписи высота блоков также получается разной, из-за чего некоторые блоки «цепляются» за другие и не переходят на другую строку. Здесь может помочь установка высоты всех блоков через свойство height, например 100px или возврат к использованию display: inline.

Влияние обтекания

Свойство float кроме способности по созданию плавающих элементов имеет ряд особенностей, о которых необходимо знать. Главная особенность в том, что float действует на все близлежащие элементы, заставляя их участвовать в обтекании. Рассмотрим её на примере 3.25, где показано создание стрелок на одной строке с использованием значений left и right свойства float.

Пример 3.25. Влияние обтекания

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Стрелки</title>
  <style type="text/css">
   .arrow { background: #f0f0f0; padding: 0 5px; }
   .arrow A {
    color: red; /* Цвет ссылок */
    text-decoration: none; /* Убираем подчеркивание */
    font-size: 1.5em;
   }
   .left { float: left; }
   .right { float: right; }
  </style>
 </head>
 <body>
  <div class="arrow">
   <div class="left"><a href="#">←</a></div>
   <div class="right"><a href="#">→</a></div>
  </div>
  <p>Текст</p>
</body>
</html>
```

Хотя для текста и стрелок используются разные блочные элементы, которые должны располагаться на разных строках, влияние float заставляет подниматься текст выше, поскольку он попадает в зону обтекания (рис. 3.36). Также не отображается цвет фона у слоя `arrow` из-за того, что плавающие элементы не участвуют в потоке документа, по сути, это аналогично тому, что слой `arrow` оставить пустым.

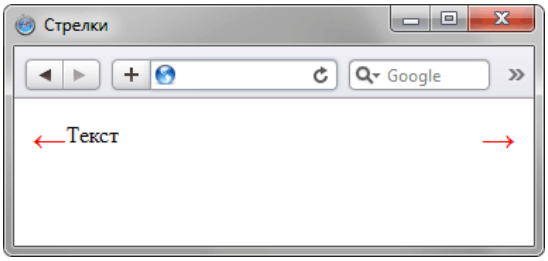


Рис. 3.36. Влияние обтекания на нижележащий текст

Плавающие элементы не оказывают влияние на высоту блока, в котором они находятся. Чтобы это обнаружить, достаточно обвести блок рамкой и поместить внутрь плавающий элемент с текстом (пример 3.26).

Пример 3.26. Высота блока

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Высота слоя</title>
  <style type="text/css">
   DIV {
    border: 1px solid #000; padding: 10px;
   }
   .fig {
    float: left; margin: 0 10px 5px 0;
   }
  </style>
 </head>
 <body>
  <div>
   <img src="images/figure.jpg" class="fig" />Винни-Пух в гостях у Кролика
  </div>
 </body>
</html>
```

Рисунок не оказывает воздействие на высоту слоя и выходит за его пределы (рис. 3.37).

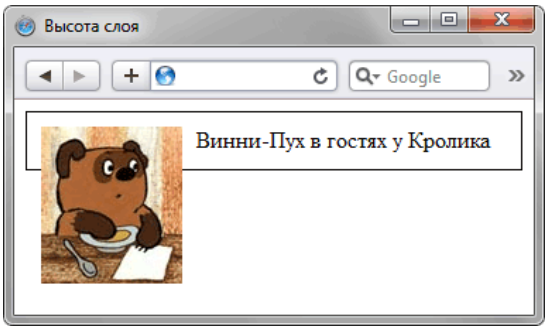


Рис. 3.37. Высота слоя с плавающим элементом

Все эти особенности плавающих элементов могут оказать довольно неприятное воздействие на макет веб-страницы, особенно в случаях подобным перечисленным выше. Основной способ добиться желаемого результат это в нужный момент отменить обтекание. Для этого есть несколько способов.

Отмена обтекания

Обтекание это мощный инструмент вёрстки, применяемый для выравнивания и упорядочивания элементов. Однако чтобы держать этот инструмент под контролем, необходим противовес, без которого огромный потенциал float сужается до пары узких задач. Речь идёт об отмене обтекания с помощью разных методов. Перечислим четыре наиболее популярных.

Ширина элемента

Если плавающий элемент будет занимать всю доступную ширину, то остальные элементы, следующие

за ним, будут начинаться с новой строки. Для этого надо включить свойство width со значением 100%. Так, стиль в примере 3.25 можно дополнить следующим образом:

```
.left { float:  left; width: 50%; }
.right { float: right; width: 50%; text-align:  right; }
```

Каждый слой со стрелкой теперь занимает ширину 50% и в сумме они дают 100% ширины, поэтому абзац, идущий после слоя `arrow`, начнётся с новой строки.

Этот метод применяется редко, поскольку ширину нельзя применить к изображениям, к тому же он не решает проблему с высотой слоя и его фоном.

Использование overflow

Свойство overflow управляет отображением содержания блочного элемента, если оно целиком не помещается и выходит за область заданных размеров. Значение auto в частности, добавляет полосы прокрутки при необходимости, а hidden отображает только область внутри элемента, остальное скрывает. Кроме этого, использование overflow со значением auto, scroll или hidden отменяет действие float. Стиль для примера 3.25 дополняется всего одной строкой:

```
.arrow { overflow: hidden; }
```

Результат после применения свойства overflow сразу меняется (рис. 3.38).

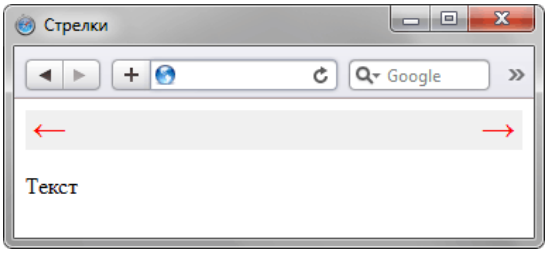


Рис. 3.38. Влияние свойства overflow на фон

Аналогично дополняется пример 3.26:

```
DIV { overflow: hidden; }
```

Результат показан на рис. 3.39.

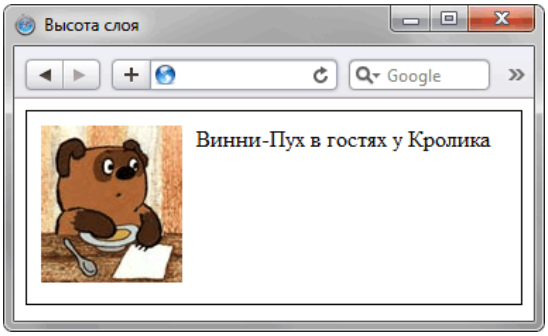


Рис. 3.39. Влияние свойства overflow на границу

overflow одно из самых популярных свойств работающее в связке float. Но иногда возникают ситуации, когда применение этого метода недопустимо. Это происходит в тех случаях, когда элемент выходит за пределы элемента, в этом случае он будет «обрезан». В примере 3.27 картинка сдвигается влево от своего исходного положения.

Пример 3.27. Обрезание области элемента

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Картинка за пределами слоя</title>
  <style type="text/css">
   DIV {
    border: 1px solid #000; padding: 10px;
    overflow: hidden;
   }
   .fig {
    float: left; margin: 0 10px 5px 0;
    position: relative; /* Относительное позиционирование */
    left: -30px; /* Сдвигаем влево */
   }
  </style>
 </head>
 <body>
  <div>
   <img src="images/figure.jpg" class="fig" />Винни-Пух в гостях у Кролика
  </div>
 </body>
</html>
```

Результат примера показан на рис. 3.40.

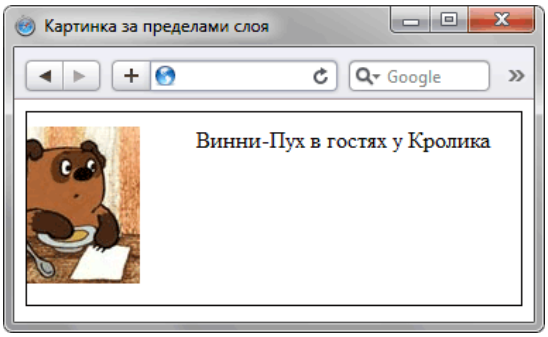


Рис. 3.40. Обрезание картинки

В IE6 метод работает только при установленном свойстве hasLayout. Чтобы его включить, можно добавить zoom: 1 наряду со свойством overflow.

Свойство clear

Для отмены действия float применяется свойство clear со следующими значениями.

- left — отменяет обтекание с левого края элемента (float: left). При этом все другие элементы на этой стороне будут опущены вниз, и располагаться под текущим элементом.
- right — отменяет обтекание с правой стороны элемента (float: right).
- both — отменяет обтекание элемента одновременно с правого и левого края. Это значение рекомендуется устанавливать, когда требуется отменить обтекание элемента, но неизвестно точно с какой стороны.

Чтобы отменить действие обтекания, свойство clear надо добавлять к элементу, идущему после плавающего. Обычно вводят универсальный класс, к примеру, `clear` и вставляют пустой тег `<div>` с этим классом (пример 3.28).

Пример 3.28. Использование clear

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Стрелки</title>
  <style type="text/css">
   .arrow { background: #f0f0f0; padding: 0 5px; }
   .arrow A {
    color: red; /* Цвет ссылок */
    text-decoration: none; /* Убираем подчеркивание */
    font-size: 1.5em;
   }
```

```
    .left { float: left; }
    .right { float: right; }
    .clear { clear: both; }
   </style>
  </head>
  <body>
   <div class="arrow">
    <div class="left"><a href="#">←</a></div>
    <div class="right"><a href="#">→</a></div>
    <div class="clear"></div>
   </div>
   <p>Текст</p>
  </body>
</html>
```

В данном примере для класса `clear` установлено свойство clear со значением both. Если вам требуется использовать разные значения, то можно ввести несколько классов и применять их по необходимости.

Этот метод также является одним из самых популярных в вёрстке в силу простоты и универсальности. Но опять же иногда возникают комбинации, в которых использование clear даёт сбой. Это происходит, когда в коде встречается одновременно несколько разных плавающих элементов. Так, в примере 3.29 с помощью float создаются две колонки, а в правой колонке float упорядочивает фотографии.

Пример 3.29. Использование float

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Использование float</title>
  <style type="text/css">
   .col1, .col2, .footer { padding: 10px; }
   .col1 { float: left; width: 100px; background: #E8D9A9; }
   .col2 { margin-left: 120px; background: #ECC0A4; }
   .photo { width: 150px; border: 1px solid #333; text-align: center;
    margin-right: 10px; background: #fff; float: left; }
   .clearleft { clear: left; }
   .footer { background: #9A5044; color: #fff; clear: left; }
  </style>
 </head>
 <body>
  <div class="col1"><h3>Меню</h3><p>Лучшие фотографии</p>
   <p>По годам</p><p>По рейтингу</p><p>По комментариям</p></div>
  <div class="col2">
   <div class="photo">
    <p><img src="images/thumb1.jpg" alt="" /></p>
    <p class="caption">Софийский собор</p>
   </div>
   <div class="photo">
    <p><img src="images/thumb2.jpg" alt="" /></p>
    <p class="caption">Польский костёл</p>
   </div><div class="clearleft"></div>
   <p>Новая строка</p>
  </div>
  <div class="footer">Подвал</div>
 </body>
</html>
```

Результат примера показан на рис. 3.41. Свойство clear действует не только на класс `photo`, но и на класс `col1`, т.е. на все элементы выше, у которых установлено float. Поэтому «Новая строка» начинается не сразу же после фотографий, а после завершения левой колонки.

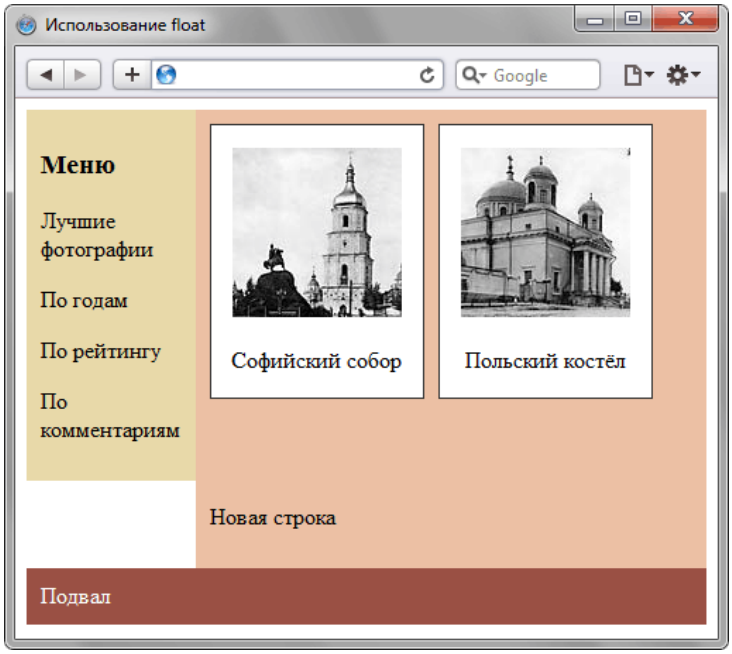


Рис. 3.41. Ошибка с отображением строки

В подобных ситуациях помогает комбинация разных методов. Так, фотографии можно добавить в контейнер с overflow и удалить `clearleft`. Колонка приобретет следующий вид.

```
<div class="col2">
 <div style="overflow: hidden">
  <div class="photo">
   <p><img src="images/thumb1.jpg" alt="" /></p>
   <p class="caption">Софийский собор</p>
  </div>
  <div class="photo">
   <p><img src="images/thumb2.jpg" alt="" /></p>
   <p class="caption">Польский костёл</p>
  </div>
 </div>
 <p>Новая строка</p>
</div>
```

Псевдокласс :after

Частое включение пустого тега ** ** со свойством clear захламляет код, особенно при активном использовании свойства float. Логично перенести всё в стили, избавившись от лишних тегов. Для этого воспользуемся псевдоэлементом :after, который в комбинации со свойством content добавляет текст после элемента. К такому тексту можно применить стилевые свойства, в частности clear. Остаётся только спрятать выводимый текст от браузера.

```
.clearleft:after {
 content: "."; /* Выводим текст после элемента (точку) */
 clear: left; /* Отменяем обтекание*/
 visibility: hidden; /* Прячем текст */
 display: block; /* Блочный элемент */
 height: 0; /* Высота */
}
```

Какой именно выводить символ значения не имеет, он в любом случае скрывается от пользователя через visibility, но даже будучи скрытым, текст при этом занимает какую-то высоту и влияет на близлежащие элементы. Поэтому выводимый текст ещё необходимо превратить в блочный элемент и задать ему нулевую высоту.

Поскольку текст, генерируемый через псевдоэлемент :after, располагается после элемента, он с лёгкостью заменяет конструкцию ** **. Там, где она требуется достаточно только добавить класс `clearleft`, как показано в примере 3.30.

Пример 3.30. Псевдоэлемент :after

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>:after</title>
  <style type="text/css">
   DIV { border: 1px solid #000; padding: 10px;  }
   .fig { float: left; margin: 0 10px 5px 0; }
   .clearleft:after {
    content: ".";
    clear: left;
    visibility: hidden; display: block; height: 0;
   }
  </style>
 </head>
 <body>
  <div class="clearleft">
   <img src="images/figure.jpg" class="fig" />Винни-Пух в гостях у Кролика
  </div>
 </body>
</html>
```

Браузер IE не поддерживает :after до версии 7.0 включительно, поэтому приведённый пример в этих версиях работать не будет. Добавление zoom: 1 к плавающему элементу отменяет обтекание.

Удобство приведённого метода заключается в использовании класса `clearleft`, который при необходимости добавляется к любому тегу. Также можно ввести дополнительные классы, стиль у которых будет различаться другими значениями clear.

Позиционирование элементов

Позиционированием называется положение элемента в системе координат. Различают четыре типа позиционирования: нормальное, абсолютное, фиксированное и относительное. В зависимости от типа, который устанавливается через свойство position, изменяется и система координат.

Благодаря комбинации свойств position, left, top, right и bottom элемент можно накладывать один на другой, выводить в точке с определенными координатами, фиксировать в указанном месте, определить положение одного элемента относительно другого и др. Подобно другим свойствам CSS управление позиционированием доступно через скрипты. Таким образом, можно динамически изменять положение элементов без перезагрузки страницы, создавая анимацию и различные эффекты.

Нормальное позиционирование

Если для элемента свойство position не задано или его значение static, элемент выводится в потоке документа как обычно. Иными словами, элементы отображаются на странице в том порядке, как они идут в исходном коде HTML.

Свойства left, top, right, bottom если определены, игнорируются.

Абсолютное позиционирование

При абсолютном позиционировании элемент не существует в потоке документа и его положение задаётся относительно краёв браузера. Задать этот тип можно через значение absolute свойства position. Координаты указываются относительно краёв окна браузера, называемого «видимой областью» (рис. 3.42).

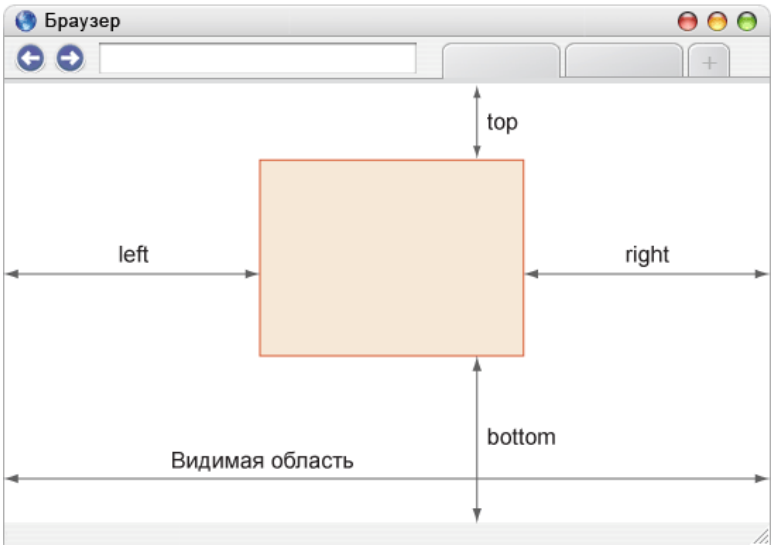


Рис. 3.42. Значения свойств left, right, top и bottom при абсолютном позиционировании

Для режима характерны следующие особенности.

- Ширина слоя, если она не задана явно, равна не auto, как обычно, а ширине контента плюс значения полей, границ и отступов.
- Слой не меняет своё положение, если у него нет свойств right, right, top и bottom.
- Свойства left и top имеют более высокий приоритет по сравнению с right и bottom. Если left и right противоречат друг другу, то значение right игнорируется. То же самое касается и bottom.
- Если left задать отрицательное значение, то слой уйдёт за левый край браузера, полосы прокрутки

при этом не возникнет. Это один из способов спрятать элемент от просмотра. То же относится и к свойству top, только слой уйдёт за верхний край.

- Если left задать значение больше ширины видимой области или указать right с отрицательным значением, появится горизонтальная полоса прокрутки. Подобное правило работает и с top, только речь пойдёт о вертикальной полосе прокрутки.
- Одновременно указанные свойства left и right формируют ширину слоя, но только если width не указано. Стоит добавить свойство width и значение right будет проигнорировано. Аналогично произойдёт и с высотой слоя, только уже участвуют свойства top, bottom и height.
- Элемент с абсолютным позиционированием перемещается вместе с документом при его прокрутке.

Свойство position со значением absolute можно использовать для создания эффекта фреймов. Кроме абсолютного позиционирования для элементов необходимо назначить свойство overflow со значением auto. Тогда при превышении контентом высоты видимой области появится полоса прокрутки. Высота и ширина «фреймов» формируется автоматически путём одновременного использования свойств left, right для ширины и top, bottom для высоты (пример 3.31).

Пример 3.31. Создание аналога фреймов

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Абсолютное позиционирование</title>
  <style type="text/css">
   body { margin: 0; }
   #sidebar, #content { position: absolute; }
   #sidebar, #content { overflow: auto; padding: 10px; }
   #header {
    height: 80px; /* Высота слоя */
    background: #FEDFC0; border-bottom: 2px solid #7B5427;
   }
   #header h1 { padding: 20px; margin: 0; }
   #sidebar {
    width: 150px; background: #ECF5E4; border-right: 1px solid #231F20;
    top: 82px; /* Расстояние от верхнего края */
    bottom: 0; /* Расстояние снизу  */
   }
   #content {
    top: 82px; /* Расстояние от верхнего края */
    left: 170px; /* Расстояние от левого края */
    bottom: 0; right: 0;
   }
  </style>
 </head>
 <body>
  <div id="header"><h1>Плов народов мира</h1></div>
  <div id="sidebar">
   <p>Плов по-фергански</p><p>Плов узбекский</p>
   <p>Плов сибирский</p><p>Плов итальянский</p>
   <p>Плов эстонский</p><p>Плов по-американски</p>
   <p>Плов по-индейски</p>
  </div>
  <div id="content">
   <h2>Плов по-фергански</h2>
   <p>Положить в казан нарезанное кусочками мясо и поджарить его до
   образования корочки. Нашинкованный кольцами лук жарить вместе
   с мясом до красноватого цвета, затем добавить морковь, нарезанную
   соломкой. Положить половину соли, всё перемешать и жарить, пока
   морковь не приобретёт золотисто-коричневый цвет. После этого налить
   половину необходимого количества воды и дать закипеть.</p>
   <p>Засыпать ровным слоем рис, усилить огонь и тотчас налить воду,
   чтобы она накрыла рис на 1–1,5 см. Как только вода выпарится, плов
   при помощи шумовки собрать к середине горкой, проколоть палочкой в
   нескольких местах так, чтобы вода, находящаяся на поверхности,
   прошла на дно. Затем накрыть плов и дать ему упреть 20–25 мин.</p>
   <p>Тщательно перемешайте готовый плов, переложите в большое блюдо,
   сверху разложите мясо.</p>
  </div>
 </body>
</html>
```

Результат данного примера показан на рис. 3.43. Слой header выводится в потоке как обычно, а для

слоёв `sidebar` и `content` установлено абсолютное позиционирование.

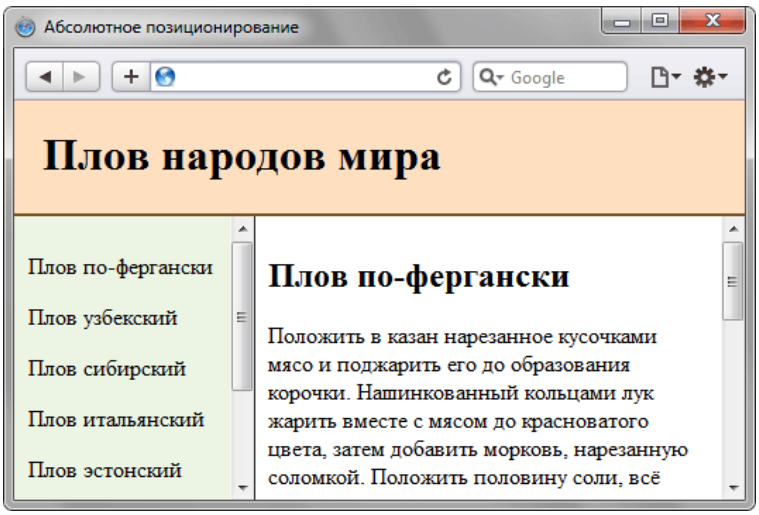


Рис. 3.43. Применение абсолютного позиционирования

В браузере IE6 для абсолютно позиционированных элементов нельзя одновременно задать свойства left, right и top, bottom.

Абсолютное позиционирование также применяется для создания различных эффектов, например, всплывающей подсказки к фотографиям. В отличие от атрибута title тега `<img>` который также выводит текст подсказки, через стили можно управлять видом текста выводимого с помощью скрипта.

Для начала создадим пустой слой с идентификатором `floatTip` и определим его стиль. Обязательными должны быть три стилевых свойства — position со значением absolute, display со значением none скрывает слой и width задает ширину слоя с подсказкой. Остальные свойства используются по желанию разработчика и предназначены для изменения оформления слоя (пример 3.32).

Пример 3.32. Стиль для всплывающей подсказки

```
#floatTip {
  position: absolute; /* Абсолютное позиционирование */
  width: 250px; /* Ширина блока */
  display: none; /* Прячем от показа */
  border: 1px solid #000; /* Параметры рамки */
  padding: 5px; /* Поля вокруг текста */
  font-family: sans-serif; /* Рубленый шрифт */
  font-size: 9pt; /* Размер шрифта */
  color: #333; /* Цвет текста */
  background: #ECF5E4; /* Цвет фона */
}
```

Сам скрипт состоит из двух функций — moveTip() отслеживает движение мыши и в соответствии с координатами курсора меняет положение слоя, и toolTip() управляет видимостью слоя и выводит в нем желаемый текст (пример 3.33).

Пример 3.33. Скрипт для вывода слоя

```
document.onmousemove = moveTip;
function moveTip(e) {
  floatTipStyle = document.getElementById("floatTip").style;
  w = 250; // Ширина слоя
  // Для браузера IE
  if (document.all) {
    x = event.x + document.body.scrollLeft;
    y = event.y + document.body.scrollTop;
  // Для остальных браузеров
  } else {
    x = e.pageX; // Координата X курсора
    y = e.pageY; // Координата Y курсора
  }
  // Показывать слой справа от курсора
```

```
  if ((x + w + 10) < document.body.clientWidth) {
    floatTipStyle.left = x + 'px';
  // Показывать слой слева от курсора
  } else {
    floatTipStyle.left = x - w + 'px';
  }
  // Положение от верхнего края окна браузера
  floatTipStyle.top = y + 20 + 'px';
}
function toolTip(msg) {
  floatTipStyle = document.getElementById("floatTip").style;
  if (msg) {
    // Выводим текст подсказки
    document.getElementById("floatTip").innerHTML = msg;
    floatTipStyle.display = "block"; // Показываем слой
  } else {
    floatTipStyle.display = "none"; // Прячем слой
  }
}
```

Для удобства и универсальности скрипт следует вынести в отдельный файл и подключать его через атрибут src тега **<script>**. Окончательный код показан в примере 3.34.

Пример 3.34. Создание всплывающей подсказки

XHTML 1.0 | CSS 2.1 | CSS 3 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Всплывающая подсказка</title>
  <style type="text/css">
   #floatTip {
    position: absolute; width: 250px; display: none;
    border: 1px solid #000; padding: 5px;
    font-family: sans-serif; font-size: 9pt;
    color: #333;  background: #ECF5E4;
    opacity: 0.85; /* Прозрачность слоя */
   }
  </style>
  <script type="text/javascript" src="scripts/tooltip.js"></script>
 </head>
 <body>
  <p><img src="images/odel.jpg" alt="Фотография" width="300" height="450"
   onmouseover="toolTip('Фотоаппарат: Canon EOS 350D<br />' +
   'Объектив: Canon EF 24-105 f/4L IS USM<br />' +
   'Вспышка: Canon Speedlite 580 EX<br />' +
   'Выдержка: 1/125<br />Диафрагма: 5.6')" onmouseout="toolTip()" /></p>
   <div id="floatTip"></div>
 </body>
</html>
```

Результат данного примера показан на рис. 3.44. Обратите внимание, что переносы текста при вызове функции toolTip() сделаны для удобства восприятия и имеют синтаксис JavaScript. В Safari скрипт иногда не работает при переносе текста, в этом случае текст следует записать в одну строку. К стилям добавлено свойство CSS3 opacity, которое добавляет для слоя небольшую прозрачность. В IE до версии 9.0 не поддерживается.

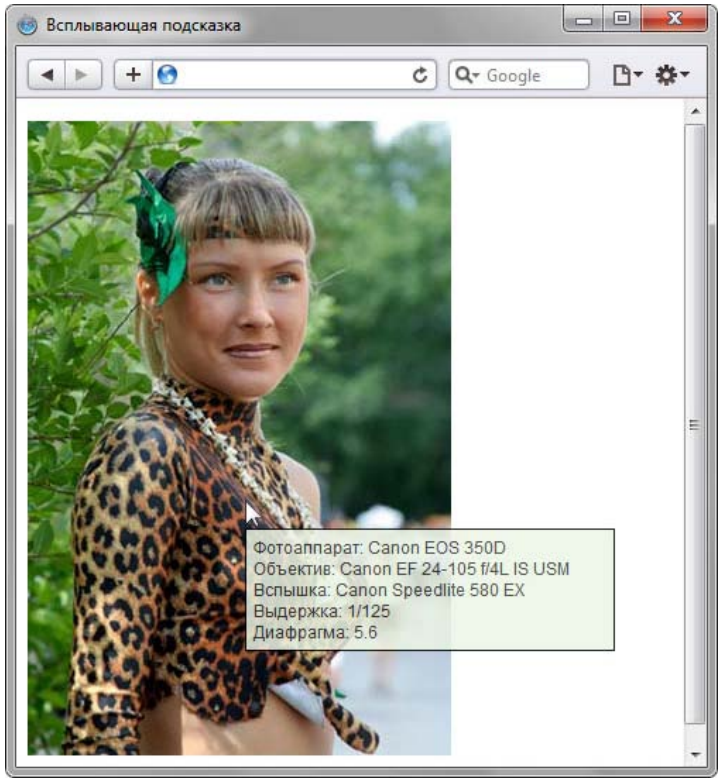


Рис. 3.44. Всплывающая подсказка, выводимая с помощью JavaScript

Фиксированное положение

Фиксированное положение слоя задаётся значением fixed свойства position и по своему действию похоже на абсолютное позиционирование. Но в отличие от него привязывается к указанной свойствами left, top, right и bottom точке на экране и не меняет своего положения при прокрутке веб-страницы. Ещё одна разница от absolute заключается в том, что при выходе фиксированного слоя за пределы видимой области справа или снизу от неё, не возникает полос прокрутки.

Применяется такой тип позиционирования для создания меню, вкладок, заголовков, в общем, любых элементов, которые должны быть закреплены на странице и всегда видны посетителю. В примере 3.35 показано добавление подвала, который остаётся на одном месте независимо от объёма информации на сайте.

Пример 3.35. Фиксированный подвал

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Фиксированный подвал</title>
  <style type="text/css">
   BODY { margin-bottom: 50px; }
   #footer {
    position: fixed; /* Фиксированное положение */
    left: 0; bottom: 0; /* Левый нижний угол */
    padding: 10px; /* Поля вокруг текста */
    background: #39b54a; /* Цвет фона */
    color: #fff; /* Цвет текста */
    width: 100%; /* Ширина слоя */
   }
  </style>
 </head>
 <body>
  <div id="content">
   Все перечисленные на сайте методы ловли льва являются теоретическими
   и базируются на вычислительных методах. Автор не гарантирует
   вашей безопасности при их использовании и снимает с себя всякую
   ответственность за результат.
   Помните, лев это хищник и опасное животное!
  </div>
```

```
  <div id="footer">&copy; Влад Мержевич</div>
 </body>
</html>
```

Результат примера показан на рис. 3.45. Поскольку фиксированный подвал накладывается на текст и скрывает его, добавлен отступ снизу для селектора **BODY**. Браузер IE6 не поддерживает значение fixed, поэтому в нем данный пример будет работать некорректно.

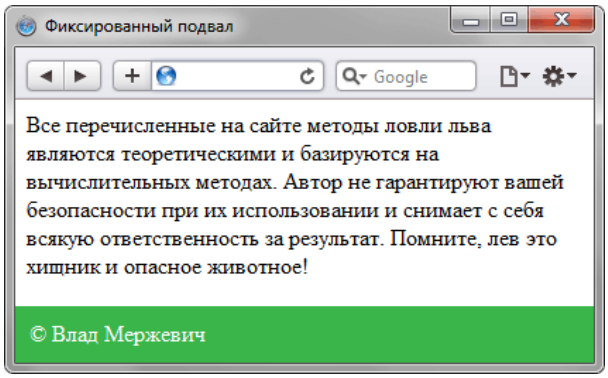


Рис. 3.45. Подвал внизу страницы

Относительное позиционирование

Если задать значение relative свойства position, то положение элемента устанавливается относительно его исходного места. Добавление свойств left, top, right и bottom изменяет позицию элемента и сдвигает его в ту или иную сторону от первоначального расположения. Положительное значение left определяет сдвиг вправо от левой границы элемента, отрицательное — сдвиг влево. Положительное значение top задаёт сдвиг элемента вниз (рис. 3.46), отрицательное — сдвиг вверх.

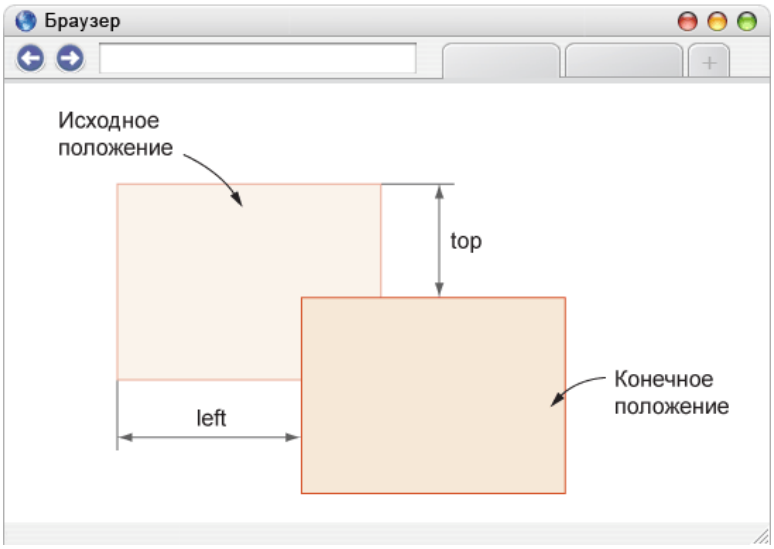


Рис. 3.46. Значения свойств left и top при относительном позиционировании

Свойства bottom и right производят обратный эффект. При положительном значении right сдвигает элемент влево от его правого края, при отрицательном — сдвигает вправо (рис. 3.47). При положительном значении top элемент опускается вниз, при отрицательном поднимается вверх.

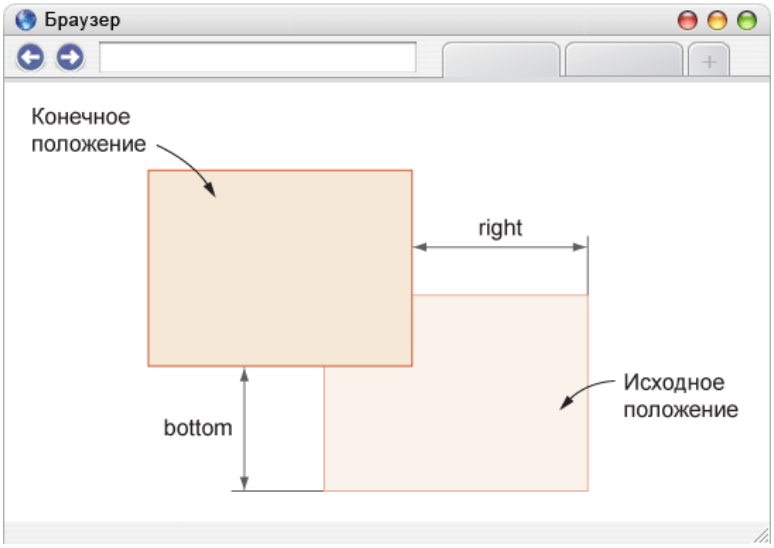


Рис. 3.47. Значения свойств right и bottom при относительном позиционировании

Для относительного позиционирования характерны следующие особеннсоти.

- Этот тип позиционирования не применим к элементам таблицы вроде ячеек, строк, колонок и др.
- При смещении элемента относительно исходного положения, место, которое занимал элемент, остаётся пустым и не заполняется ниже или вышележащими элементами.

В примере 3.36 показан сдвиг текста заголовка вниз для придания ему особого стиля написания.

Пример 3.36. Заголовок текста

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Заголовок</title>
  <style type="text/css">
   H1 {
    font: bold 2em Arial, Tahome, sans-serif; /* Параметры шрифта */
    color: #fff;  background: #375D4C;
    padding: 0 10px;
   }
   H1 SPAN {
    position: relative; /* Относительное позиционирование */
    top: 0.3em; /* Сдвигаем вниз */
   }
  </style>
 </head>
 <body>
  <h1><span>Аз и буки шрифтовой науки</span></h1>
  <p>Шрифт это средство выражения дизайна, а не какого-то
  банального чтения.</p>
 </body>
</html>
```

Результат данного примера показан на рис. 3.48.

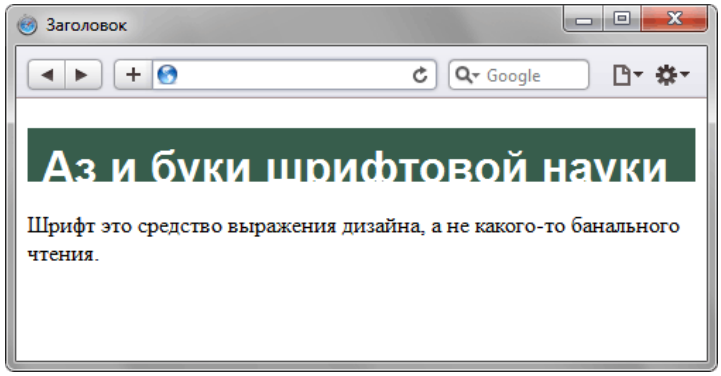


Рис. 3.48. Сдвиг текста относительно исходного положения

Вложенные слои

Обычно относительное позиционирование само по себе применяется не часто, поскольку есть ряд свойств выполняющих фактически ту же роль, к примеру, тот же margin. Но сочетание разных типов позиционирования для вложенных слоёв является одним из удобных и практичных приёмов вёрстки. Если для родительского элемента задать relative, а для дочернего absolute, то произойдёт смена системы координат и положение дочернего элемента при этом указывается относительно его родителя (рис. 3.49).

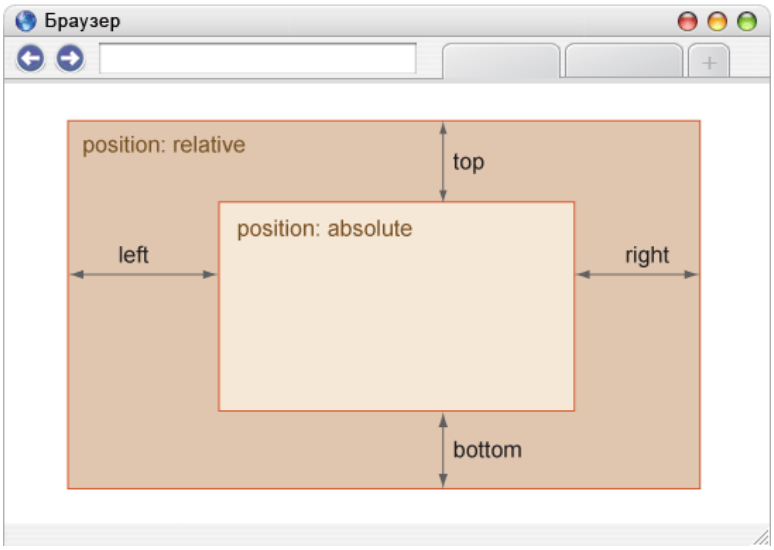


Рис. 3.49. Значения свойств left, right, top и bottom во вложенных слоях

Отсчёт координат ведётся от внутреннего края границы, значения полей не учитываются. В следующем примере текст располагается в правом нижнем углу слоя возле границы, игнорируя свойство padding.

```
<div style="position: relative; padding: 10px; border: 1px solid #000;
background: #fc0; height: 500px;">
 <div style="position: absolute; right: 0; bottom: 0">
  Текст
 </div
</div>
```

Благодаря использованию четырёх свойств left, right, top, bottom для управления положением внутреннего слоя, размеры родительского слоя знать не обязательно. Это расширяет сферу применения позиционирования, поэтому position довольно активно применяется при вёрстке различных элементов. В качестве примера возьмём наложение на фотографию разной информации: количество комментариев к ней, ссылки «Добавить комментарий», «Информация об авторе» и «Добавить в избранное». В целом, результат должен быть примерно таким, как на рис. 3.50.

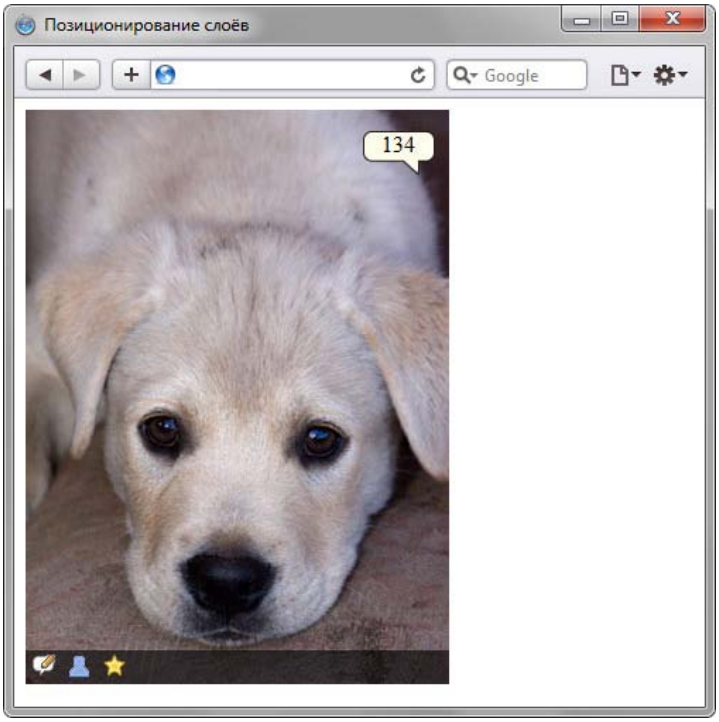


Рис. 3.50. Фотография с наложенными элементами

Сам код приведён в примере 3.37. Для слоя `photo` установлено относительное позиционирование, а для внутренних слоёв `img` (вывод фотографии), `comment` (число комментариев) и `tool` (ссылки внизу фотографии) задано абсолютное позиционирование.

Пример 3.37. Позиционирование слоёв

XHTML 1.0 | CSS 2.1 | CSS 3 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Позиционирование слоёв</title>
  <style type="text/css">
   .photo { width: 300px; height: 407px; position: relative; }
   .img, .comment, .tool { position: absolute; }
   .comment {
    background: url(images/comment_num.png) no-repeat;
    width: 51px; height: 31px; /* Размеры слоя */
    text-align: center;  /* Выравнивание по центру */
    right: 10px; top: 15px; /* Положение от правого верхнего угла */
   }
   .tool {
    background: #000; /* Для IE */
    background-color: rgba(0,0,0,0.6); /* Чёрный полупрозрачный фон */
    padding: 3px 0 0;
    bottom: 0; /* Положение от нижнего края */
    width: 100%; /* Ширина слоя */
   }
   .tool img { margin-left: 5px; }
  </style>
 </head>
 <body>
  <div class="photo">
   <div class="img"><img src="images/dog.jpg" alt="Щенок" /></div>
   <div class="comment" title="Количество комментариев">134</div>
   <div class="tool">
    <img src="images/comment_add.png" alt="Добавить комментарий" />
    <img src="images/photo_author.png" alt="Информация об авторе" />
    <img src="images/photo_star.png" alt="Добавить в избранное" />
   </div>
  </div>
 </body>
</html>
```

Наложение и порядок слоёв

На веб-странице расположены три изображения игральных карт (рис. 3.51). Пока они лежат рядом, их порядок значения не имеет, но если применить к ним позиционирование и сместить изображения так, чтобы они накладывались друг на друга, одна карта будет находиться выше другой (рис. 3.52).

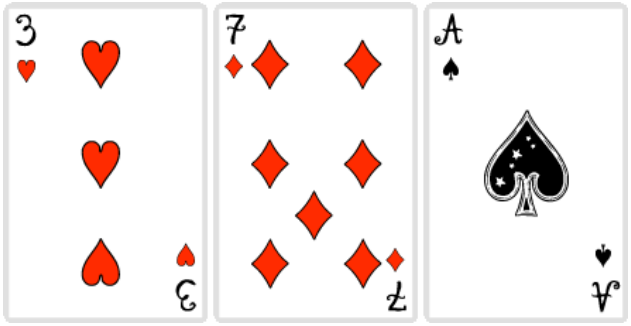


Рис. 3.51. Карты рядом друг с другом

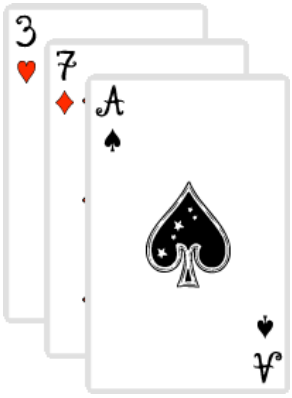


Рис. 3.52. Карты одна на другой

Если представить веб-страницу в виде трёхмерного пространства (рис. 3.53), то видно, что карты располагаются также по оси Z. Значения по этой оси и определяют, какая карта к нам ближе, какая дальше, иными словами порядок их наложения друг на друга. В коде документа (пример 3.38) порядок определяется автоматически на основе потока документа. Чем элемент ниже в коде, тем он выше по оси Z, поэтому изображение с тузом, как самое нижнее, располагается поверх остальных карт.

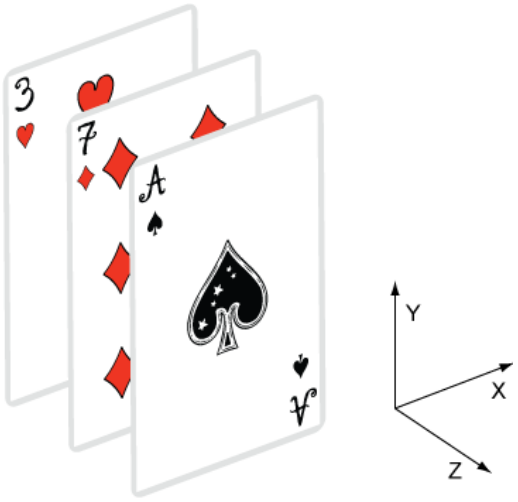


Рис. 3.53. Воображаемые координаты веб-страницы

Пример 3.38. Обычный порядок карт

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Порядок карт</title>
  <style type="text/css">
   .card IMG { position: relative; }
   .seven { left: -120px; top: 25px; }
   .ace { left: -240px; top: 50px; }
  </style>
 </head>
 <body>
  <p class="card">
   <img src="images/3.png" alt="3" class="three" />
   <img src="images/7.png" alt="7" class="seven" />
   <img src="images/ace.png" alt="Туз" class="ace" />
  </p>
 </body>
</html>
```

В CSS за положением по Z-оси отвечает свойство z-index, которое определяет, «ближе» к нам элемент находится или «дальше». В качестве значений принимается целое число, чем оно больше, тем выше располагается элемент по отношению к другим. Сама веб-страница имеет значение 0, так что даже z-index: 1 заставит элемент перекрывать все нижележащие. Доработаем пример 3.38 так, чтобы порядок карт поменялся на противоположный, причём только редактируя стиль, оставляя HTML-код прежним.

```
.card IMG { position: relative; }
.three { top: 50px; left: 55px; z-index: 5; }
.seven { left: -120px; top: 25px; z-index: 2; }
.ace { left: -295px; z-index: 1; }
```

Свойство z-index для класса `three` специально установлено как 5 для демонстрации, что последовательность значений z-index роли не играет. Главное, чтобы одно число было больше другого.

Свойство z-index работает только для элементов, у которых значение position задано как absolute, fixed или relative.

Когда требуется расположить элемент поверх всех остальных на странице, ему ставят очень большое значение z-index, например 9999. Это гарантирует, что даже если в стилях и применяется z-index, он будет меньше указанного. В примере 3.39 у карт при наведении на них курсора меняется z-index на 10. Никаких скриптов здесь не понадобится, всё делается через псевдокласс :hover.

Пример 3.39. Изменение z-index при наведении на карту

```
.card IMG { position: relative; }
.three { top: 50px; left: 55px; z-index: 5; }
.seven { left: -120px; top: 25px; z-index: 2;}
.ace { left: -295px; z-index: 1; }
.card IMG:hover { z-index: 10; }
```

Глава IV

Вёрстка с помощью таблиц

Несмотря на явно устаревший метод вёрстки, я всё же решил включить раздел посвященный вёрстке таблицами. Сравнение двух разных методов поможет лучше понять принципы размещения элементов на странице и в конечном итоге повысить эффективность своей работы.

К тому же совсем отказываться от таблиц неверно, они продолжают служить по своему прямому предназначению — размещению табличных данных. Такие данные особо популярны в деловом мире для демонстрации и сравнения цифр, но также используются для наглядного представления практически любой информации. Также некоторые элементы дизайна крайне сложно сверстать на слоях, но очень просто на таблицах.

Поэтому в случае категоричного отказа от табличной вёрстки вы можете использовать эту главу для получения новых идей по оформлению таблиц в документе. Или применять таблицы для ускорения вёрстки в сложных случаях.

Особенности таблиц

Чтобы понимать, что можно ожидать от таблиц при вёрстке, следует знать их явные и неявные особенности, которые перечислены далее.

Вложенные таблицы

Одну таблицу допускается помещать внутрь ячейки другой таблицы. Это требуется для представления сложных данных или в том случае, когда одна таблица выступает в роли модульной сетки, а вторая, внутри нее, в роли обычной таблицы.

Размеры таблицы

Размеры таблицы изначально не устанавливаются и вычисляются на основе содержимого ячеек. В итоге суммарная ширина таблицы складывается из следующих параметров:

- ширина содержимого ячеек;
- толщина всех границ между ячеек;
- поля вокруг содержимого, устанавливаемые через атрибут cellpadding;
- расстояние между ячейками, которое определяется значением cellspacing.

Если для таблицы установлена её ширина в процентах или пикселах, то содержимое таблицы подстраивается под указанные размеры. Так, браузер автоматически добавляет переносы строк в текст, чтобы он полностью поместился в ячейку, и при этом ширина таблицы осталась без изменений. Бывает, что ширину содержимого ячейки невозможно изменить, как это, например, происходит с рисунками. В этом случае ширина таблицы увеличивается, несмотря на указанные размеры. Чтобы избежать указанной ситуации применяют несколько средств.

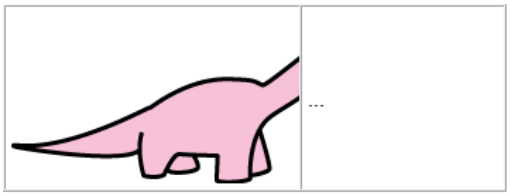
- Не добавляют в ячейку фиксированной ширины те изображения, размер которых превышает ширину ячейки. Способ, конечно, звучит банально, тем не менее, зная особенности ячеек, можно избежать неприятностей с их отображением.
- Для тега **<table>** используют стилевое свойство table-layout со значением fixed. Применение этого свойства позволяет обрезать рисунок, если он не помещается целиком в ячейку (пример 4.1).

Пример 4.1. Использование table-layout

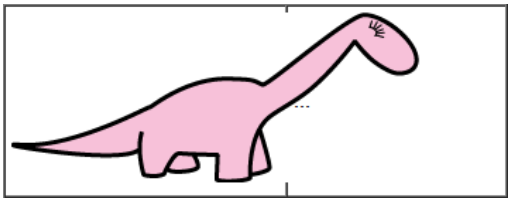
XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Таблица</title>
  <style type="text/css">
   TABLE {
    table-layout: fixed; /* Ячейки фиксированной ширины */
   }
   TD.dino { width: 200px; }
  </style>
 </head>
 <body>
  <table width="100%" cellspacing="0" cellpadding="4" border="1">
   <tr>
    <td class="dino"><img src="images/dino.gif" width="289"
      height="120" alt="Динозаврик" /></td>
    <td>...</td>
   </tr>
  </table>
 </body>
</html>
```

Результат данного примера зависит от браузера. В старых браузерах рисунок за пределами ячейки обрезается (рис. 4.1а), в современных браузерах рисунок выводится поверх второй ячейки (рис. 4.1б).



а. Вид таблицы в IE6, IE7, Firefox 2



б. Вид таблицы в IE8, IE9, Firefox 3, Safari и Chrome

Рис. 4.1. Использование table-layout

Сделать единообразный вид таблицы во всех браузерах легко, для этого к ячейке с рисунком следует добавить свойство overflow со значением hidden. При этом всё, что не помещается в ячейку, будет «обрезано», как продемонстрировано на рис. 4.1а. Стиль в этом случае изменится незначительно.

```
TABLE {
  table-layout: fixed; /* Ячейки фиксированной ширины */
}
TD.dino {
  width: 200px;
  overflow: hidden;
}
```

Высота ячеек

Ячейки в одной строке взаимосвязаны и их высота одинакова. Это позволяет делать макеты с колонками одной высоты. В примере 4.2 приведён такой макет, в котором, несмотря на разную высоту контента, колонки равны по высоте.

Пример 4.2. Колонки одной высоты

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Высота ячеек</title>
  <style type="text/css">
   TABLE {
    width: 100%; /* Ширина таблицы */
   }
   TD {
    padding: 10px; /* Поля в ячейках */
   }
   TD.content {
    background: #f0f0f0; /* Цвет фона левой колонки */
   }
   TD.menu {
    width: 120px; /* Ширина правой колонки*/
    background: #9c3022; /* Цвет фона правой колонки */
    color: #fff; /* Цвет текста */
    vertical-align: top;
   }
  </style>
 </head>
 <body>
  <table>
   <tr>
    <td class="content">
     <p>Мясо отварить до готовности. Промыть свеклу, очистить,
```

```
      нарезать соломкой и тушить с помидорами до полуготовности.</p>
      <p>Бульон процедить, мясо нарезать кусочками. В бульон добавить
      нарезанный дольками картофель, довести до кипения, опустить нарезанную
      соломкой свежую капусту и варить 10-15 минут, добавить пассерованные
      овощи, болгарский перец, нашинкованный тонкой соломкой,
      специи и довести до готовности.</p>
      <p>Готовому борщу дать настояться в течение 20-25 минут.
      При подаче к столу добавить сметану, мясо, зелень.</p>
     </td>
     <td class="menu">Борщ</td>
    </tr>
   </table>
  </body>
</html>
```

Результат примера показан на рис. 4.2.

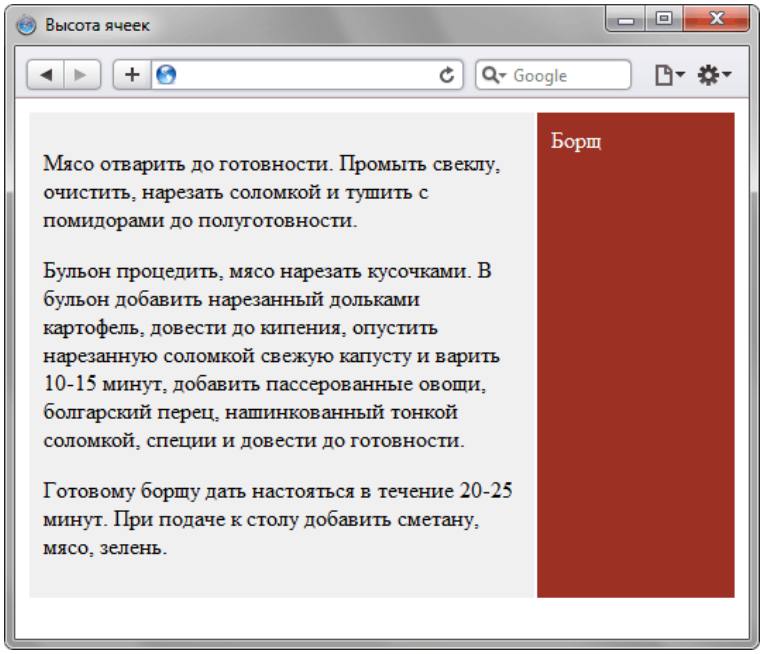


Рис. 4.2. Макет, созданный с использованием таблицы

В данном примере в ячейках разное содержание, но высота ячеек одинакова.

Порядок ячеек

Основой таблицы выступает строка и ячейка, формирование таблицы происходит слева направо и сверху вниз (рис. 4.3).

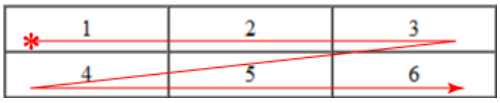


Рис. 4.3. Порядок создания ячеек

Неудобства этой схемы проявляются при активном использовании колонок и большом числе ячеек. Конечно, есть теги <col> и <colgroup>, но с их помощью можно только установить ширину колонок и выравнивание в ячейках. Такие параметры, как границы в колонках или цвет фона приходится задавать через стили, добавляя к определенным ячейкам стилевой класс. Вставка новых ячеек или редактирование существующих может привести к ошибкам отображения таблицы.

Загрузка таблицы

Пока таблица не загрузится полностью, её содержимое не начнёт отображаться. Дело в том, что браузер, прежде чем показать содержимое таблицы, должен вычислить необходимые размеры ячеек, их ширину и высоту. А для этого нужно знать, что в этих ячейках находится. Поэтому браузер и

ожидает, пока загрузится все, что находится в ячейках, и только потом отображает таблицу.

Исходя из этого факта, таблицы не используют для хранения большой информации (от 100 Кб). А чтобы ускорить загрузку табличного макета, его разбивают на отдельные таблицы или используют свойство table-layout, применение которого позволяет несколько повысить скорость отображения содержимого таблицы. В обычной таблице браузер анализирует все ячейки и затем уже изменяет ширину колонок на основе этой информации. Включение table-layout со значением fixed меняет алгоритм расчета — браузер анализирует только первую строку и ширину колонок строит согласно ей. За счёт уменьшения числа вычислений и происходит выигрыш скорости отображения таблицы в целом.

Таблицы и стили

Сами по себе таблицы выглядят довольно «бедно», к тому же браузеры по своему отображают некоторые характеристики таблиц, в частности, рамки. Вместе с тем эти недостатки легко исправить воспользовавшись мощью стилей. При этом весьма расширяются средства по оформлению таблиц, что позволяет удачно вписать таблицы в дизайн сайта и нагляднее представить табличные данные.

Цвет фона ячеек

Цвет фона одновременно всех ячеек таблицы устанавливается через свойство background, которое применяется к селектору **TABLE**. При этом следует помнить о правилах использования стилей, в частности, наследовании свойств элементов. Хотя свойство background не наследуется, для ячеек значением фона по умолчанию выступает transparent, т.е. прозрачность, поэтому эффект заливки фона получается и у ячеек. Если одновременно с **TABLE** задать цвет у селектора **TD** или **TH**, то этот цвет будет установлен в качестве фона ячейки (пример 4.3).

Пример 4.3. Цвет фона

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Таблицы</title>
  <style type="text/css">
   table {
    background: maroon; /* Цвет фона таблицы */
    color: white; /* Цвет текста */
   }
   td {
    background: navy; /* Цвет фона ячеек */
   }
  </style>
 </head>
 <body>
  <table cellpadding="4" cellspacing="1">
   <tr><th>Заголовок 1</th><th>Заголовок 2</th></tr>
   <tr><td>Ячейка 3</td><td>Ячейка 4</td></tr>
  </table>
 </body>
</html>
```

В данном примере получим синий цвет фона у ячеек (тег **<td>**) и красный у заголовка (тег **<th>**). Это связано с тем, что стиль для селектора **TH** не определен, поэтому «просвечивается» фон родителя, в данном случае, селектора **TABLE**. А цвет для селектора **TD** указан явно, вот ячейки и «заливаются» синим цветом.

Результат данного примера показан на рис. 4.4.

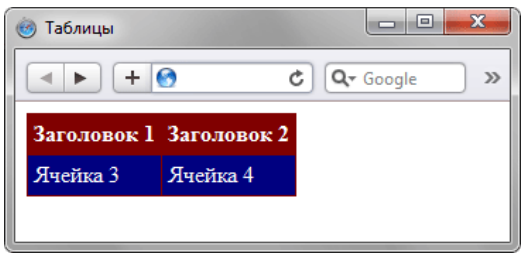


Рис. 4.4. Изменение цвета фона

Поля внутри ячеек

Полем называется расстояние между краем содержимого ячейки и её границей. Обычно для этой цели

применяется атрибут cellpadding тега **<table>**. Он определяет значение поля в пикселах со всех сторон ячейки. Допускается использовать стилевое свойство padding, добавляя его к селектору **TD**, как показано в примере 4.4.

Пример 4.4. Поля в таблицах

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Таблицы</title>
  <style type="text/css">
   TABLE {
    background: white; /* Цвет фона таблицы */
    color: white; /* Цвет текста */
   }
   TD, TH {
    background: maroon; /* Цвет фона ячеек */
    padding: 5px; /* Поля вокруг текста */
   }
  </style>
 </head>
 <body>
  <table cellspacing="1">
   <tr><th>Заголовок 1</th><th>Заголовок 2</th></tr>
   <tr><td>Ячейка 3</td><td>Ячейка 4</td></tr>
  </table>
 </body>
</html>
```

В данном примере с помощью группирования селектором поля установлены одновременно для селектора **TD** и **TH**. Результат примера показан на рис. 4.5.

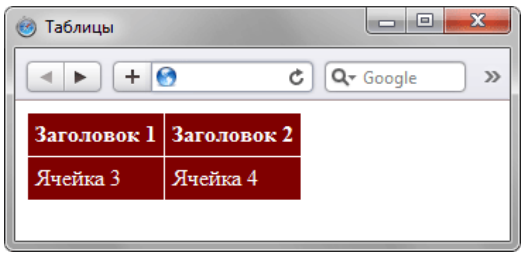


Рис. 4.5. Поля в ячейках

Если применяется стилевое свойство padding для ячеек таблицы, то действие атрибута cellpadding тега **<table>** игнорируется.

Расстояние между ячейками

Для изменения расстояния между ячейками применяется атрибут cellspacing тега **<table>**. Влияние этого атрибута хорошо заметно при использовании границ вокруг ячеек или при заливке ячеек цветом, который выделяется на фоне страницы. В качестве замены cellspacing выступает стилевое свойство border-spacing, оно устанавливает расстояние между границами ячеек. Одно значение устанавливает одновременно расстояние по вертикали и горизонтали между границами ячеек. Если значений у этого свойства два, то первое определяет горизонтальное расстояние (т.е. слева и справа от ячейки), а второе — вертикальное (сверху и снизу).

Свойство border-spacing действует только в том случае, если для селектора **TABLE** не задано свойство border-collapse со значением collapse (пример 4.5).

Пример 4.5. Расстояние между границами ячеек

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Замена cellspacing</title>
  <style type="text/css">
   TABLE {
    width: 100%; /* Ширина таблицы */
    border: 1px solid #399; /* Граница вокруг таблицы */
    border-spacing: 7px 5px; /* Расстояние между границ */
   }
   TD {
    background: #fc0; /* Цвет фона */
    border: 1px solid #333; /* Граница вокруг ячеек */
    padding: 5px; /* Поля в ячейках */
   }
  </style>
 </head>
 <body>
  <table>
    <tr><td>Леонардо</td><td>5</td><td>8</td></tr>
    <tr><td>Рафаэль</td><td>4</td><td>11</td></tr>
    <tr><td>Микеланджело</td><td>24</td><td>9</td></tr>
    <tr><td>Донателло</td><td>2</td><td>13</td></tr>
  </table>
 </body>
</html>
```

Результат данного примера показан на рис. 4.6.

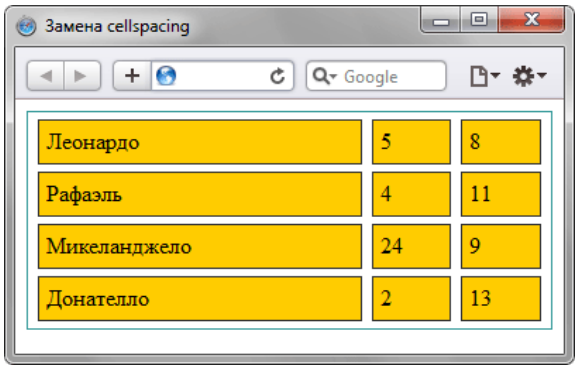


Рис. 4.6. Вид таблицы при использовании border-spacing

Браузер Internet Explorer до седьмой версии включительно не поддерживает свойство border-spacing, поэтому в этом браузере для таблиц будет применяться значение cellspacing заданное по умолчанию (обычно оно равно 2px).

При добавлении к селектору **TABLE** свойства border-collapse со значением collapse, атрибут cellspacing игнорируется, а значение border-spacing обнуляется.

Границы и рамки

По умолчанию границы в таблице изначально нет, а её добавление происходит с помощью атрибута border тега **<table>**. Браузеры по-разному отображают такую границу, поэтому лучше не указывать этот атрибут вообще, а рисование границ возложить на стили. Рассмотрим два метода, непосредственно связанных со стилями.

Использование атрибута cellspacing

Известно, что атрибут cellspacing тега **<table>** задаёт расстояние между ячейками таблицы. Если используется разный цвет фона таблицы и ячеек, то между ячейками возникнет сетка линий, цвет которых совпадает с цветом таблицы, а толщина равна значению атрибута cellspacing в пикселах. В вышеприведенном выше примере 4.3 этот эффект показан, поэтому повторять его не буду.

Заметим, что это не совсем удобный способ создания границ, поскольку он имеет ограниченную область применения. Так можно получить только одноцветную сетку, а не вертикальные или горизонтальные линии в нужных местах.

Применение свойства border

Стилевое свойство border одновременно устанавливает цвет границы, её стиль и толщину вокруг элемента. Когда требуется создать отдельные линии на разных сторонах, лучше использовать производные — border-left, border-right, border-top и border-bottom, эти свойства соответственно определяют границу слева, справа, сверху и снизу.

Применяя свойство border к селектору **TABLE**, мы добавляем рамку вокруг таблицы в целом, а к селектору **TD** или **TH** — рамку вокруг ячеек (пример 4.6).

Пример 4.6. Добавление двойной рамки

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Таблицы</title>
  <style type="text/css">
   TABLE {
    background: #dc0; /* Цвет фона таблицы */
    border: 5px double #000; /* Рамка вокруг таблицы */
   }
   TD, TH {
    padding: 5px; /* Поля вокруг текста */
    border: 1px solid #fff; /* Рамка вокруг ячеек */
   }
  </style>
 </head>
 <body>
  <table>
   <tr><th>Заголовок 1</th><th>Заголовок 2</th></tr>
   <tr><td>Ячейка 3</td><td>Ячейка 4</td></tr>
  </table>
 </body>
</html>
```

В данном примере используется двойная рамка черного цвета вокруг самой таблицы и сплошная рамка белого цвета вокруг каждой ячейки. Результат примера показан на рис. 4.7.

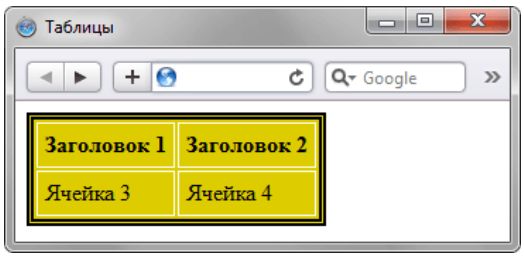


Рис. 4.7. Граница вокруг таблицы и ячеек

Обратите внимание, что в месте состыковки ячеек образуются двойные линии. Они получаются опять же за счет действия атрибута cellspacing тега **<table>**. Хотя в коде примера этот атрибут нигде не фигурирует, браузер использует его по умолчанию. Если задать <table cellspacing="0">, то получим не двойные, а одинарные линии, но удвоенной толщины. Для изменения указанной особенности применяется стилевое свойство border-collapse со значением collapse, которое добавляется к селектору **TABLE** (пример 4.7).

Пример 4.7. Создание одинарной рамки

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Таблицы</title>
  <style type="text/css">
   TABLE {
    border-collapse: collapse; /* Убираем двойные границы между ячейками */
    background: #dc0; /* Цвет фона таблицы */
    border: 4px solid #000; /* Рамка вокруг таблицы */
   }
```

```
    TD, TH {
     padding: 5px; /* Поля вокруг текста */
     border: 2px solid green; /* Рамка вокруг ячеек */
    }
   </style>
  </head>
  <body>
   <table>
    <tr><th>Заголовок 1</th><th>Заголовок 2</th></tr>
    <tr><td>Ячейка 3</td><td>Ячейка 4</td></tr>
   </table>
  </body>
</html>
```

В данном примере создается сплошная линия зеленого цвета между ячейками и черная вокруг таблицы. Все границы внутри таблицы имеют одинаковую толщину. Результат примера показан на рис. 4.8.

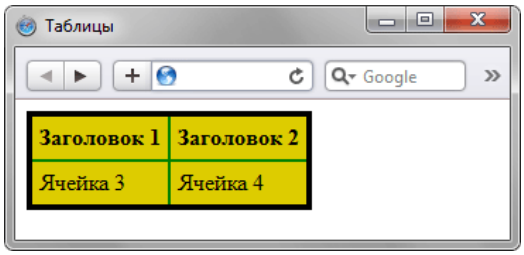


Рис. 4.8. Граница вокруг таблицы

Выравнивание содержимого ячеек

По умолчанию текст в ячейке таблицы выравнивается по левому краю. Исключением из этого правила служит тег **<th>**, он определяет заголовок, в котором выравнивание происходит по центру. Чтобы изменить способ выравнивания применяется стилевое свойство text-align (пример 4.8).

Пример 4.8. Выравнивание содержимого ячеек по горизонтали

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Таблицы</title>
  <style type="text/css">
   TABLE {
    border-collapse: collapse; /* Убираем двойные линии между ячейками */
    width: 300px; /* Ширина таблицы */
   }
   TH {
    background: #fc0; /* Цвет фона ячейки */
    text-align: left; /* Выравнивание по левому краю */
   }
   TD {
    background: #fff; /* Цвет фона ячеек */
    text-align: center; /* Выравнивание по центру */
   }
   TH, TD {
    border: 1px solid black; /* Параметры рамки */
    padding: 4px; /* Поля вокруг текста */
   }
  </style>
 </head>
 <body>
  <table>
   <tr><th>Заголовок 1</th><td>Ячейка 1</td><td>Ячейка 2</td></tr>
   <tr><th>Заголовок 2</th><td>Ячейка 3</td><td>Ячейка 4</td></tr>
  </table>
 </body>
</html>
```

В данном примере содержимое тега **<th>** выравнивается по левому краю, а содержимое тега **<td>** — по центру. Результат примера показан ниже (рис. 4.9)..

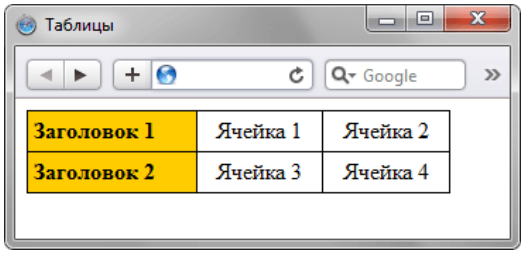


Рис. 4.9. Выравнивание текста в ячейках

Выравнивание по вертикали в ячейке всегда происходит по её центру, если это не оговорено особо. Это не всегда удобно, особенно для таблиц, у которых содержимое ячеек различается по высоте. В таком случае выравнивание устанавливают по верхнему краю ячейки с помощью свойства vertical-align, как показано в примере 4.9.

Пример 4.9. Выравнивание содержимого ячеек по вертикали

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Таблицы</title>
  <style type="text/css">
   TABLE {
    border-collapse: collapse; /* Убираем двойные линии между ячейками */
    width: 300px; /* Ширина таблицы */
   }
   TH, TD {
    border: 1px solid black; /* Параметры рамки */
    text-align: center; /* Выравнивание по центру */
    padding: 4px; /* Поля вокруг текста */
   }
   TH {
    background: #fc0; /* Цвет фона ячейки */
    height: 40px; /* Высота ячеек */
    vertical-align: bottom; /* Выравнивание по нижнему краю */
    padding: 0; /* Убираем поля вокруг текста */
   }
  </style>
 </head>
 <body>
  <table>
   <tr><th>Заголовок 1</th><th>Заголовок 2</th></tr>
   <tr><td>Ячейка 1</td><td>Ячейка 2</td></tr>
  </table>
 </body>
</html>
```

В данном примере устанавливается высота заголовка **<th>** как 40 пикселов и выравнивание текста происходит по нижнему краю. Результат примера показан на рис. 4.10.

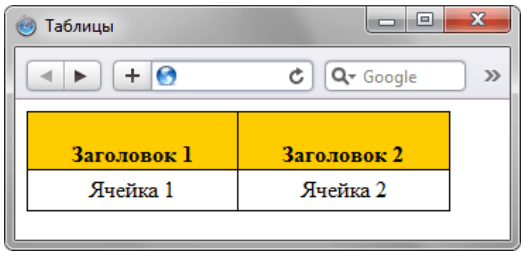


Рис. 4.10. Выравнивание текста в ячейках

Пустые ячейки

Браузеры иначе отображают ячейку, внутри которой ничего нет. «Ничего» в данном случае означает, что внутрь ячейки не добавили ни рисунок, ни текст, причём пробел в расчёт не принимается. Естественно, вид ячеек различается только в том случае, если вокруг них установлена граница. При использовании невидимой рамки, вид ячеек, независимо от того, есть в них что-нибудь или нет,

совпадает.

Старые браузеры не отображали цвет фона пустых ячеек вида `<td bgcolor="#ffcc00"></td>`, поэтому в том случае, когда требовалось оставить ячейку без содержимого, но отобразить цвет фона, внутрь ячейки добавляли неразделяемый пробел ( ). Пробел не всегда подходит, особенно когда нужно установить высоту ячейки 1–2 пиксела, из-за чего широкое распространение получил однопиксельный прозрачный рисунок. Действительно, такой рисунок можно масштабировать на свое усмотрение, но он на веб-странице никак не отображается.

К счастью эпоха однопиксельных рисунков и всяческих распорок на их основе прошла. Браузеры достаточно корректно работают с таблицами и без присутствия содержимого ячеек.

Для управления видом пустых ячеек используется свойство empty-cells, при значении hide граница и фон в пустых ячейках не отображается. Если все ячейки в строке пустые, то строка прячется целиком. Ячейка считается пустой в следующих случаях:

- нет вообще никаких символов;
- в ячейке содержится только перевод строки, символ табуляции или пробел;
- значение visibility установлено как hidden.

Добавление неразрывного пробела   воспринимается как видимое содержание, т.е. ячейка уже будет не пустой (пример 4.10).

Пример 4.10. Пустые ячейки

XHTML 1.0 | CSS 2.1 | CSS 3 | IE 6 | IE 7 | IE 8 | Cr 5 | Op 10 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Использование empty-cells</title>
  <style type="text/css">
   TABLE {
    border: 4px double #399; /* Граница вокруг таблицы */
   }
   TD {
    background: #fc0; /* Цвет фона */
    border: 1px solid #333; /* Граница вокруг ячеек */
    empty-cells: hide; /* Прячем пустые ячейки */
    padding: 5px; /* Поля в ячейках */
   }
  </style>
 </head>
 <body>
  <table width="100%">
   <tr>
    <td>Леонардо</td><td>5</td><td>8</td>
   </tr>
   <tr>
    <td>Рафаэль</td><td> </td><td>11</td>
   </tr>
   <tr>
    <td>Микеланджело</td><td>24</td><td></td>
   </tr>
   <tr>
    <td>Донателло</td><td> </td><td>13</td>
   </tr>
  </table>
 </body>
</html>
```

Вид таблицы в браузере Safari показан на рис. 4.11а. Та же таблица в браузере IE7 продемонстрирована на рис. 4.11б.

Леонардо	5	8
Рафаэль		11
Микеланджело	24	
Донателло		13

а. В браузере Safari, Firefox, Opera, IE8, IE9

Леонардо	5	8
Рафаэль		11
Микеланджело	24	
Донателло		13

б. В браузере IE6, IE7

Рис. 4.11. Вид таблицы с пустыми ячейками

Разница между таблицами и слоями

Адепты табличной вёрстки любят сделать сложный макет и бросить верстальщикам вызов: «ну-ка, повторите такое на слоях!». Сразу заявляем, это провокация, на которую не следует попадаться и надо попросту игнорировать. Смысл в том, что таблицы и слои это совершенно разные объекты и их вообще некорректно сравнивать между собой. В итоге это напоминает детский спор «кто сильнее — кит или слон?». Дети продолжают отстаивать свою позицию и яростно доказывать, что их фаворит сильнее, взрослые люди лишь понимающе улыбаются, услышав этот вопрос.

Для объективности, тем не менее, приведем ряд характерных элементов, свойственных разным методам вёрстки (табл. 4.1).

Табл. 4.1. Сравнение характеристик

	Таблица	Слои
Колонки	Колонки формируются ячейками таблицы, их высота одинакова и взаимосвязана.	Колонки создаются разными слоями, их высота разная и зависит от содержания.
Ширина	Если ширина таблицы явно не указана, она вычисляется на основе содержимого таблицы.	По умолчанию слой занимает всю доступную ему ширину.
Расположение	Строки таблицы отображаются в том порядке, как они представлены в коде. Ячейки идут слева направо и сверху вниз.	Порядок слоев в коде может не соответствовать их положению в браузере.
Загрузка	Как правило, пока таблица не загрузится полностью, содержимое её не будет показываться. Если на веб-странице размещена большая таблица, загрузка страницы существенно замедляется.	Слои отображаются последовательно, по мере загрузки документа.
HTML-код	Код для создания таблиц может быть громоздок, особенно если требуется объединить множество ячеек или сделать несколько вложенных таблиц.	Код, как правило, компактный.
Выравнивание по высоте	Содержимое внутри ячейки легко выравнивать по её верхнему, нижнему краю или середине.	Выравнивание нетривиально.

Как видно из таблицы, разница между таблицами и слоями существенная, особенно это касается колонок. В таблице колонки, создаваемые ячейками, имеют одинаковую высоту, при вёрстке слоями приходится использовать искусственные методы, чтобы добиться подобного результата. Всё дело в том, что идеология слоёв не предполагает их искусственное растягивание, и методы построения колонок одинаковой высоты построены, как правило, на визуальном обмане. Если взять в качестве основы пример 4.2 и вместо таблицы использовать ** **, то получится следующее решение (пример 4.11).

Пример 4.11. Колонки одинаковой высоты

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 <title>Высота слоёв</title>
 <style type="text/css">
  .content, .menu { padding: 10px;}
  .container {
   background: #9c3022; /* Цвет фона правой колонки */
  }
  .content {
   margin-right: 120px;
   background: #f0f0f0; /* Цвет фона левой колонки */
  }
  .menu {
   width: 100px; /* Ширина правой колонки*/
   color: #fff; /* Цвет текста */
   float: right; /* Колонка справа */
  }
 </style>
</head>
<body>
 <div class="container">
  <div class="menu">Борщ</div>
  <div class="content">
   Мясо отварить до готовности. Промыть свеклу, очистить,
   нарезать соломкой и тушить с помидорами до полуготовности.
  </div>
 </div>
</body>
</html>
```

В данном примере колонки имеют разную высоту, а то, что в браузере они кажутся одинаковыми, получается за счет хитрого применения фоновой заливки. Тем не менее, пример не лишён недостатков и хорошо работает только в том случае, когда содержимое правой колонки по высоте меньше содержимого левой колонки. Если точно неизвестно, какая колонка будет больше, то существует несколько решений, как этого добиться. В частности, один из известных методов предполагает добавление большого значения padding-bottom к слоям, что поднимает их вверх, с одновременным опусканием их вниз за счёт отрицательного значения margin-bottom с тем же значением. Стиль в таком случае будет следующий.

```
.content, .menu {
 padding: 10px;
 padding-bottom: 10000px; /* Поле снизу */
 margin-bottom: -10000px; /* Опускаем вниз */
}
.container {
 overflow: hidden; /* Обрезаем всё за пределами области */
}
.content {
 margin-right: 120px;
 background: #f0f0f0;
}
.menu {
 background: #9c3022;
 width: 100px;
 color: #fff;
 float: right;
}
```

Однако подчеркну ещё раз свою мысль.

Высота слоёв формируется на основе их содержания и попытка искусственного приведения слоёв к одинаковой высоте противоречит идеологии слоёв и является извращением их сути. Если вам нужны колонки одинаковой высоты — используйте таблицы.

Сравнение двух подходов на основе одного макета показывает ещё одну характерную особенность слоёв по сравнению с таблицами. Это множество решений одной и той же задачи. С одной стороны это даёт непревзойдённую гибкость при вёрстке, но с другой порождает сомнение и поиск оптимального решения. Которого по разным причинам вполне может и не быть.

Разрезание и склейка изображений

Разрезание изображения на фрагменты с последующим их объединением в одну целую картинку — давний прием, вошедший в арсенал вёрстки веб-страниц. Предварительно подготовленный рисунок разрезают на части в графическом редакторе, сохраняют части как отдельные графические изображения, а затем соединяют их вместе с помощью таблицы.

Рассмотрим вначале, зачем применяется разрезание изображений, и какие преимущества это в итоге дает, а затем, как использовать таблицу на практике.

Плюсы разрезания изображений

Создание ссылок

Отдельные рисунки при необходимости можно превращать в ссылки, причем для них можно назначать свое описание (атрибут title) и альтернативный текст (атрибут alt), который виден при отключении показа картинок в браузере или при наведении курсора мыши на изображение.

Эффект перекатывания

Набор отдельных фрагментов позволяет создавать эффект перекатывания — динамическое изменение одного рисунка на другой при наведении на него курсора мыши, и обратно на прежний, когда курсор уводится прочь.

Уменьшение объема файлов

Отдельными частями изображения удобней манипулировать, подбирая для них графический формат и его параметры таким образом, чтобы объем файла был минимален при сохранении приемлемого качества изображения. В итоге набор графических файлов будет занимать меньше места, и загружаться быстрее, чем один файл, содержащий целый рисунок.

Анимированный GIF

Использование анимированного GIF-а для изображений большого размера чревато существенным увеличением объема файла. Но можно пойти на хитрость и заменить лишь часть изображения анимацией, а остальные фрагменты оставить статичными. При этом общий объем нескольких файлов будет гораздо меньше, чем анимирование одного изображения.

Особенности вёрстки

Изображения на веб-странице по своей природе прямоугольны, но, разрезав один рисунок на составляющие элементы, получим конструктор, из которого можно сложить другую фигуру. Это напоминает детские кубики, на одну из сторон которых наклеена картинка. Складывать подобные фигуры на веб-странице требуется в силу разных причин, например, вместо фрагмента изображения требуется добавить текст. Кроме того, некоторые рисунки можно заменить их фоновым аналогом и тогда конечное изображения, сохраняя свою целостность, будет занимать всю доступную область документа.

Психологический аспект

Когда один рисунок состоит из множества фрагментов, то браузер скачивает их в несколько потоков и показывает те, которые загрузились в первую очередь. Поэтому изображение появляется как элементы мозаики. А это сразу привлекает внимание и кажется, что загрузка происходит быстрее. Так что с технической стороны один рисунок грузится быстрее, а с позиции человеческого восприятия кажется, что набор маленьких рисунков быстрее появляется.

Подготовка изображения

В качестве примера изображения, где требуется разрезание, возьмем рис. 4.12. Каждая из трёх иконок является ссылкой на соответствующий раздел, кроме того, ссылкой на главную страницу служит рисунок с названием сайта.

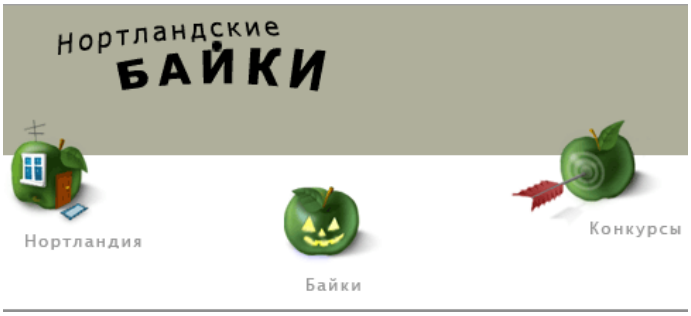


Рис. 4.12. Исходное изображение

Теоретически, в данном случае можно обойтись и без разрезания, если использовать изображение-карту (теги `<map>` и `<area>`). Однако этот вариант неприемлем в силу следующих соображений. При открытии любого раздела, иконка ему соответствующая, трансформируется, что в целом меняет изображение целиком (рис. 4.13). Если применять изображение-карту, то придется заготовить чеьыое различных изображений (одну для главной страницы и еще три для каждого раздела), а это скажется в итоге на объёме пересылаемых данных, скорости отображения сайта и качестве рисунков.

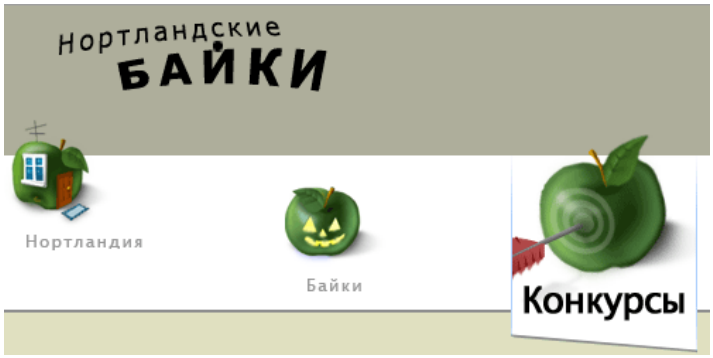


Рис. 4.13. Вид изображения при открытии раздела

Теперь требуется решить, как разрезать изображение. Вариантов здесь может быть несколько — в конечном итоге это зависит от воли автора, предназначения рисунка и его содержимого.

Разрезание изображения

Разрезание и «сборку» рисунка лучше доверить специализированной программе, в частности, Adobe Photoshop, так что все упоминания об инструментах и меню относятся именно к этой программе.

Для удобства разрезания изображения вначале следует добавить направляющие линии, по которым затем и будет происходить разделение на фрагменты (рис. 4.14). Линию снизу добавим потом через стили, поэтому она не участвует в изображении.

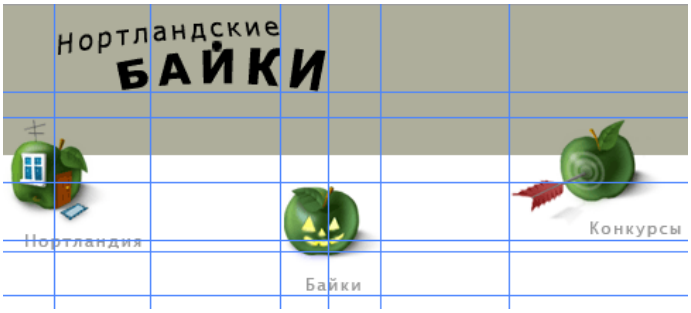


Рис. 4.14. Добавляем в изображение направляющие

Теперь используем инструмент Slice (, активация клавишей K) и по направляющим обводим требуемую прямоугольную область. Обозначенная область отмечается синей рамкой с номером фрагмента в левом верхнем углу. Размер областей можно изменять через специальный инструмент Slice Select — . Щелкаем мышью с этим инструментом по желаемому фрагменту — цвет рамки вокруг области становится желтым, а также изменяется тональность рисунка. После чего курсором мыши можно перемещать границы фрагмента за специальные маркеры по бокам и в углах области (рис. 4.15).

Рис. 4.15. Изменение области фрагмента

Для быстрого переключения между инструментами Slice и Slice Select нажмите и удерживайте клавишу Ctrl.

Во время изменения размеров фрагментов, следите за тем, чтобы области не пересекались друг с другом, и между ними не возникало промежутков. Хотя Photoshop сам отмечает подобные недочеты и принимает меры к их устранению, лучше держать все под своим контролем.

После предварительного анализа и применения инструмента Slice, получим 13 фрагментов (рис. 4.16). Синим цветом показаны фрагменты созданные инструментом Slice, серым цветом показаны автоматически созданные фрагменты.

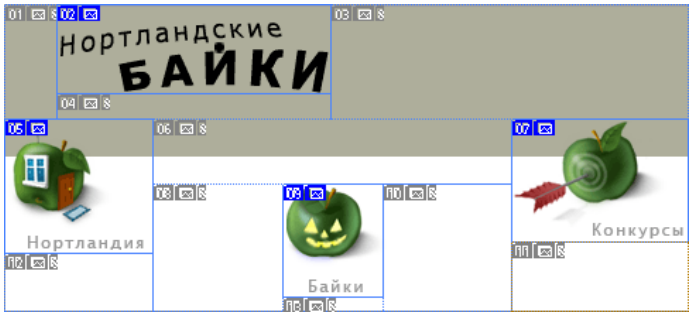


Рис. 4.16. Разрезанное на фрагменты изображение

Опасаться того, что получилось много рисунков, не стоит из-за того, что большая часть фрагментов содержит пустое изображение. Таким образом, число картинок сокращается, поскольку часть из них можно выкинуть, установив у ячейки таблицы размеры рисунка.

Использование таблицы для склейки фрагментов

После того, как фрагменты обозначены, требуется сохранить все изображения на диск. Для этого выбираем пункт меню File > Save for Web & Devices... (Файл > Сохранить для Web, Alt+Shift+Ctrl+S) чтобы открыть панель оптимизации графики (рис. 4.17).

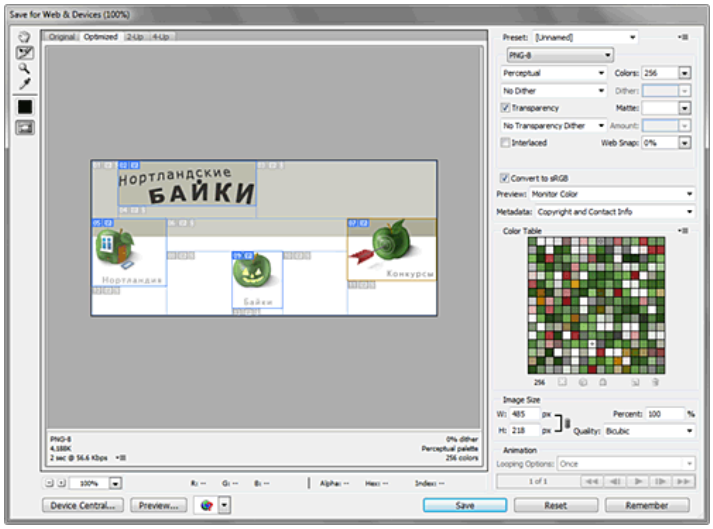


Рис. 4.17. Панель оптимизации изображений

С помощью инструмента Slice Select можно выбирать требуемый фрагмент и устанавливать для него персональные параметры вроде количества цветов, значение потерь качества, прозрачность и т.д. Допускается выделять сразу несколько фрагментов, удерживая клавишу Shift, что позволяет устанавливать для них одинаковые параметры.

По окончанию работы с фрагментами нажимаем кнопку «Save», указываем место на диске, куда будет сохранен HTML-документ, его имя, тип и настройки (рис. 4.18).

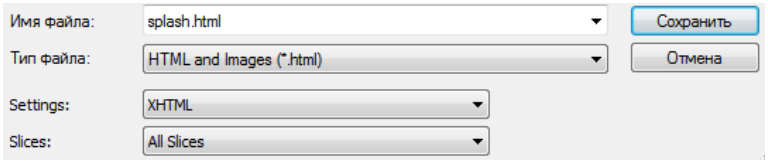


Рис. 4.18. Настройки при сохранении файла

Рисунки сохраняются автоматически в папку images, а их имя образуется от имени HTML-файла с добавлением номера фрагмента. Например, сохраняемое имя будет splash.html, тогда первый фрагмент называется splash_01.png, а последний — splash_13.png. Кроме того, создается файл spacer.gif, который представляет собой прозрачный рисунок размером 1x1 пиксел. Он используется для правильного формирования изображений в таблице.

Настройки, по которым строится HTML-код и формируются имена изображений можно изменить, если при сохранении файла в разделе Settings выбрать пункт Other... В окне параметров можно выбирать папку, куда сохранять рисунки, способ формирования имен файлов, а также HTML-кода (рис. 4.19).

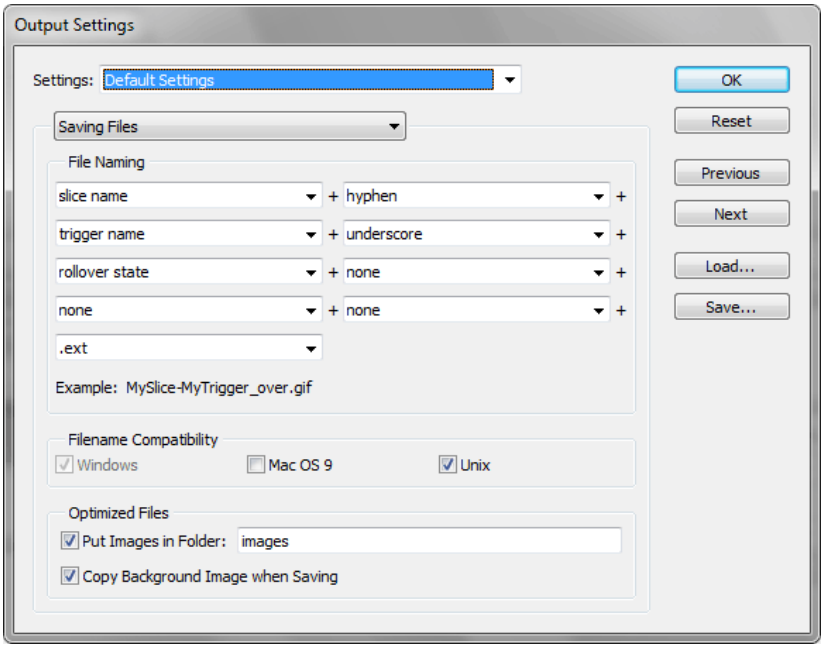


Рис. 4.19. Панель для выбора выходных настроек

Полученный в результате сохранения файлов HTML-код, после небольшого редактирования представлен в примере 4.12.

Пример 4.12. Таблица для склейки изображений

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Разрезание изображений copy</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 </head>
 <body>
  <table id="Table_01" width="486" border="0" cellpadding="0"
   cellspacing="0">
   <tr>
    <td rowspan="2">
     <img id="splash_01" src="images/splash_01.png" width="37" height="81"
      alt="" /></td>
    <td colspan="3">
     <img id="splash_02" src="images/splash_02.png" width="194" height="63"
      alt="" /></td>
    <td colspan="3" rowspan="2">
     <img id="splash_03" src="images/splash_03.png" width="254" height="81"
      alt="" /></td>
    <td>
     <img src="images/spacer.gif" width="1" height="63" alt="" /></td>
   </tr>
   <tr>
    <td colspan="3">
     <img id="splash_04" src="images/splash_04.png" width="194" height="18"
      alt="" /></td>
    <td>
     <img src="images/spacer.gif" width="1" height="18" alt="" /></td>
   </tr>
   <tr>
    <td colspan="2" rowspan="3">
     <img id="splash_05" src="images/splash_05.png" width="105" height="95"
      alt="" /></td>
    <td colspan="4">
     <img id="splash_06" src="images/splash_06.png" width="254" height="46"
      alt="" /></td>
    <td rowspan="2">
     <img id="splash_07" src="images/splash_07.png" width="126" height="87"
      alt="" /></td>
    <td>
     <img src="images/spacer.gif" width="1" height="46" alt="" /></td>
   </tr>
   <tr>
    <td rowspan="4">
     <img id="splash_08" src="images/splash_08.png" width="92" height="91"
      alt="" /></td>
```

```
    <td colspan="2" rowspan="3">
     <img id="splash_09" src="images/splash_09.png" width="71" height="80"
      alt="" /></td>
    <td rowspan="4">
     <img id="splash_10" src="images/splash_10.png" width="91" height="91"
      alt="" /></td>
    <td>
     <img src="images/spacer.gif" width="1" height="41" alt="" /></td>
   </tr>
   <tr>
    <td rowspan="3">
     <img id="splash_11" src="images/splash_11.png" width="126" height="50"
      alt="" /></td>
    <td>
     <img src="images/spacer.gif" width="1" height="8" alt="" /></td>
   </tr>
   <tr>
    <td colspan="2" rowspan="2">
     <img id="splash_12" src="images/splash_12.png" width="105" height="42"
      alt="" /></td>
    <td>
     <img src="images/spacer.gif" width="1" height="31" alt="" /></td>
   </tr>
   <tr>
    <td colspan="2">
     <img id="splash_13" src="images/splash_13.png" width="71" height="11"
      alt="" /></td>
    <td>
     <img src="images/spacer.gif" width="1" height="11" alt="" /></td>
   </tr>
   <tr>
    <td><img src="images/spacer.gif" width="37" height="1" alt="" /></td>
    <td><img src="images/spacer.gif" width="68" height="1" alt="" /></td>
    <td><img src="images/spacer.gif" width="92" height="1" alt="" /></td>
    <td><img src="images/spacer.gif" width="34" height="1" alt="" /></td>
    <td><img src="images/spacer.gif" width="37" height="1" alt="" /></td>
    <td><img src="images/spacer.gif" width="91" height="1" alt="" /></td>
    <td><img src="images/spacer.gif" width="126" height="1" alt="" /></td>
    <td></td>
   </tr>
  </table>
</body>
</html>
```

Данный код еще требует доработки, поскольку необходимо, чтобы горизонтальная серая и белая полоса занимали всю доступную ширину веб-страницы. Кроме того, часть фрагментов повторяется и от них можно избавиться.

Чтобы получить требуемый результат, введем слой с фоновым рисунком и нашу таблицу наложим поверх него. Такое изображение представлено на рис. 4.20.

Рис. 4.20. Фоновый рисунок, рамка вокруг приведена для наглядности

Теперь создаем нужный слой, назовем его `toplayer`, и в стилях указываем его параметры (пример 4.13).

Пример 4.13. Слой для формирования полос

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Нортландские байки</title>
  <style type="text/css">
```

```
   .toplayer {
    background: #aeae9b url(images/bgtop.png) repeat-x; /* Параметры фона */
    height: 216px; /* Высота слоя */
    border-bottom: 2px solid #8f8f8f; /* Линия внизу */
   }
   .toplayer table {
    margin: auto; /* Выравнивание таблицы по центру */
   }
   </style>
  </head>
  <body>
   <div class="toplayer">
    <table>...</table>
   </div>
  </body>
 </html>
```

В данном примере параметры фона слоя устанавливаются через универсальное свойство background, которое определяет путь к графическому файлу, цвет заливки и повторяемость рисунка. Хотя цвет фона в таких случаях можно не указывать раз есть фоновый рисунок, но на случай того, что пользователь отключил загрузку изображений, лучше это сделать. Высота слоя также не является обязательным параметром из-за того, что таблица внутри слоя имеет заданную высоту.

Остается убрать рисунки с незначительными фрагментами, сохранив их размеры у ячеек таблицы (пример 4.14).

Пример 4.14. Окончательный код

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Нортландские байки</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   body { margin: 0; }
   .toplayer {
    background: #aeae9b url(images/bgtop.png) repeat-x; /* Параметры фона */
    height: 216px; /* Высота слоя */
    border-bottom: 2px solid #8f8f8f; /* Линия внизу */
   }
   .toplayer table { margin: auto; }
   .toplayer img {display: block;}
  </style>
 </head>
 <body>
  <div class="toplayer">
   <table width="486" cellpadding="0" cellspacing="0">
    <tr>
     <td rowspan="2" style="width:37px; height:81px"></td>
     <td colspan="3">
      <img src="images/splash_02.png" width="194" height="63" alt="" /></td>
     <td colspan="3" rowspan="2" style="width:254px; height:81px"></td>
     <td style="width:1px; height:63px"></td>
    </tr>
    <tr>
     <td colspan="3" style="width:194px; height:18px"></td>
     <td style="width:1px; height:18px"></td>
    </tr>
    <tr>
     <td colspan="2" rowspan="3">
      <img src="images/splash_05.png" width="105" height="95" alt="" /></td>
     <td colspan="4" style="width:254px; height:46px"></td>
     <td rowspan="2">
      <img src="images/splash_07.png" width="126" height="87" alt="" /></td>
     <td style="width:1px; height:46px"></td>
    </tr>
    <tr>
     <td rowspan="4" style="width:92px; height:89px"></td>
     <td colspan="2" rowspan="3">
      <img src="images/splash_09.png" width="71" height="80" alt="" /></td>
     <td rowspan="4" style="width:91px; height:89px"></td>
     <td style="width:1px; height:41px"></td>
    </tr>
    <tr>
     <td rowspan="3" style="width:126px; height:48px"></td>
     <td style="width:1px; height:8px"></td>
    </tr>
    <tr>
```

```
     <td colspan="2" rowspan="2" style="width:105px; height: 40px"></td>
     <td style="width:1px; height: 31px"></td>
    </tr>
    <tr>
     <td colspan="2" style="width:71px; height:9px"></td>
     <td style="width:1px; height:9px"></td>
    </tr>
    <tr>
     <td style="width:37px; height:1px"></td>
     <td style="width:68px; height:1px"></td>
     <td style="width:92px; height:1px"></td>
     <td style="width:34px; height:1px"></td>
     <td style="width:37px; height:1px"></td>
     <td style="width:91px; height:1px"></td>
     <td style="width:126px; height:1px"></td>
     <td></td>
    </tr>
   </table>
  </div>
 </body>
</html>
```

В данном примере изменён доктайп на строгий, что приводит к появлению небольшого отступа внизу изображений. Чтобы его убрать, в стиле к селектору **IMG** добавлен display: block.

Кроме перечисленных плюсов у разрезания изображений есть и минусы, которые проявляются на приведенном примере. Код получается достаточно сложный, а поскольку ячейки взаимосвязаны между собой, то изменение размеров одного рисунка повлечёт за собой модификацию и всей таблицы. Кроме того, неудобно редактировать положение отдельных рисунков. Опять же, чтобы сместить один рисунок на пару пикселов вправо, придётся вносить правки сразу в несколько ячеек. Поэтому разрезание не всегда стоит применять, особенно если есть альтернатива в виде применения слоёв. Давайте рассмотрим, как сделать аналогичный макет с помощью них.

Для управления положением рисунков родительскому классу `menu` зададим относительное позиционирование, а рисункам абсолютное. Тогда применение свойств left и top будет задавать координаты изображения относительно его родителя, т.е. слоя `menu`. Сам слой при этом можно легко перемещать и это никак не повлияет на его дочерние элементы (пример 4.15).

Пример 4.15. Вёрстка с помощью слоёв

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Нортландские байки</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   body { margin: 0; }
   .toplayer {
    background: #aeae9b url(images/bgtop.png) repeat-x; /* Параметры фона */
    height: 216px; /* Высота слоя */
    border-bottom: 2px solid #8f8f8f; /* Линия внизу */
   }
   .menu {
    width: 486px; /* Ширина слоя */
    margin: auto; /* Выравнивание по центру */
    position: relative; /* Относительное позиционирование */
   }
   .toplayer img {
    position: absolute; /* Абсолютное позиционирование */
   }
   .nort { top: 80px; }
   .bayki { top: 130px; left: 200px; }
   .konkurs { top: 80px; left: 360px; }
  </style>
 </head>
 <body>
  <div class="toplayer">
   <div class="menu">
    <img src="images/ntitle.png" alt="Нортландские байки" class="title" />
    <img src="images/nort.png" alt="Нортландия" class="nort" />
    <img src="images/bayki.png" alt="Байки" class="bayki" />
    <img src="images/konkurs.png" alt="Конкурсы" class="konkurs" />
   </div>
  </div>
 </body>
```

```
</html>
```

Здесь слой `toplayer` создает с помощью фонового изображения полосу заданной высоты на всю ширину окна. Слой `menu` задаёт родительский элемент, который выстраивается по центру поверх фоновой полосы. Положение изображений внутри слоя `menu` управляется свойствами top и left. За счёт активного применения стилей HTML-код сильно сокращается, рисунки независимы друг от друга, их можно легко сдвигать, менять на другие, добавлять новые. Это как раз тот случай, когда у таблицы при вёрстке нет ни единого шанса.

Макет из двух колонок

Двухколонная модульная сетка достаточно часто применяется на сайтах, при этом, как правило, в одной колонке располагается основной материал (текст статьи, например), а во второй — ссылки на разделы сайта и другая информация. Для создания подобного макета таблицы достаточно удобны — каждая ячейка выступает в качестве отдельной колонки, что позволяет легко регулировать различные параметры отображения документа.

Ширина колонок

Для начала рассмотрим самый простой вариант, когда ширина левой колонки жестко задана в пикселах, а ширина правой колонки варьируется в зависимости от размеров окна браузера. Для этого требуется задать общую ширину всей таблицы в процентах через атрибут width тега **<table>** и для первой ячейки установить её ширину в пикселах или процентах также с помощью атрибута width, но уже для тега **<td>** (пример 4.16).

Пример 4.16. Ширина колонки в пикселах

XHTML 1.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Две колонки</title>
 </head>
 <body>
  <table width="100%" cellspacing="0" cellpadding="5">
   <tr>
    <td width="200" valign="top">Левая колонка</td>
    <td valign="top">Правая колонка</td>
   </tr>
  </table>
 </body>
</html>
```

В данном примере граница у таблицы не отображается, а вертикальное выравнивание содержимого ячеек по верхнему краю определяется атрибутом valign со значением top. Это требуется для того, чтобы при разном объеме содержимого ячеек, они не сдвигались бы относительно друг друга, а начинались одинаково от верхнего края.

Переведём все используемые атрибуты таблицы в стилевые свойства. Тогда данный код будет иметь следующий вид (пример 4.17).

Пример 4.17. Использование стилей

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Две колонки</title>
  <style type="text/css">
   .layout {
    width: 100%; /* Ширина всей таблицы в процентах */
   }
   .layout TD {
    vertical-align: top; /* Вертикальное выравнивание в ячейках */
   }
   TD.leftcol {
    width: 200px; /* Ширина левой колонки в пикселах */
   }
  </style>
 </head>
 <body>
  <table cellspacing="0" cellpadding="0" class="layout">
   <tr>
    <td class="leftcol">Левая колонка</td>
```

```
    <td>Правая колонка</td>
   </tr>
  </table>
 </body>
</html>
```

Цвет фона колонок

Чтобы визуально отделить одну колонку от другой используют разные приемы, самым распространенным из них, пожалуй, является использование фонового цвета. Для этого создаем новый стилевой класс, устанавливаем для него свойство background и применяем его к требуемой ячейке (пример 4.18).

Пример 4.18. Цвет фона

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Две колонки</title>
  <style type="text/css">
   .layout {
    width: 100%; /* Ширина всей таблицы */
   }
   TD {
    vertical-align: top; /* Вертикальное выравнивание в ячейках */
    padding: 5px; /* Поля вокруг ячеек */
   }
   TD.leftcol {
    width: 200px; /* Ширина левой колонки */
    background: #ccc; /* Цвет фона левой колонки */
   }
   TD.rightcol {
    background: #fc3; /* Цвет фона правой колонки */
   }
  </style>
 </head>
 <body>
  <table cellspacing="0" class="layout">
   <tr>
    <td class="leftcol">Левая колонка</td>
    <td class="rightcol">Правая колонка</td>
   </tr>
  </table>
 </body>
</html>
```

В данном примере разный цвет фона добавляется для правой и левой колонки (рис. 4.21).

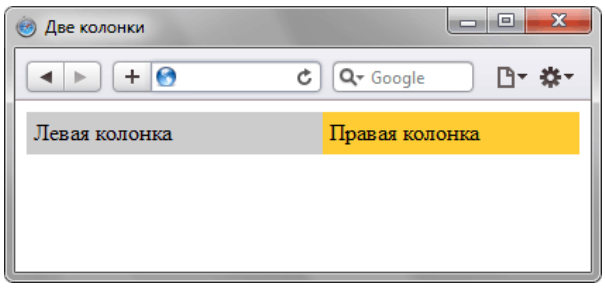


Рис. 4.21. Колонки разного цвета

Разделитель колонок

Использование полей не всегда подходит для установки нужного расстояния между колонок. Например, в случае, когда поля вокруг текста нельзя включать в силу разных соображений. Тогда поможет добавление еще одной ячейки, которая выступает в качестве разделителя между колонками (пример 4.19).

Пример 4.19. Использование трех ячеек

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Две колонки</title>
  <style type="text/css">
   .layout {
    width: 100%; /* Ширина всей таблицы */
   }
   TD {
    vertical-align: top; /* Вертикальное выравнивание в ячейках */
    padding: 5px; /* Поля вокруг ячеек */
   }
   TD.leftcol {
    width: 200px; /* Ширина левой колонки */
    background: #ccc; /* Цвет фона левой колонки */
    border: 1px solid #000; /* Параметры рамки */
   }
   TD.rightcol {
    background: #fc3; /* Цвет фона правой колонки */
    border: 1px solid #000; /* Параметры рамки */
   }
   .spacer {
    width: 10px; /* Расстояние между колонками */
   }
  </style>
 </head>
 <body>
  <table cellspacing="0" class="layout">
   <tr>
    <td class="leftcol">Левая колонка</td>
    <td class="spacer"></td>
    <td class="rightcol">Правая колонка</td>
   </tr>
  </table>
 </body>
</html>
```

В данном примере вводится еще одна колонка с имене `spacer`, это позволяет легко менять ширину между колонками. Внутрь этой ячейки можно ничего не вкладывать, браузеры достаточно корректно работают с подобными ячейками.

На рис. 4.22 показан результат примера. Для наглядности и «красоты» вокруг колонок добавлена граница.

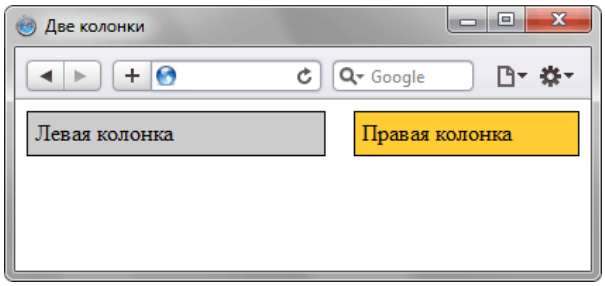


Рис. 4.22. Расстояние между колонками

Линия между колонками

Разделять колонки можно не только с помощью цвета фона и пустого пространства, но и добавлением линии между колонок. Для создания линии необходимо установить свойство border-left для правой колонки или border-right для левой (пример 4.20).

Пример 4.20. Добавление линии

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Две колонки</title>
  <style type="text/css">
   .layout {
    width: 100%; /* Ширина всей таблицы */
```

```
    }
    TD {
     vertical-align: top; /* Вертикальное выравнивание в ячейках */
     padding: 5px; /* Поля вокруг ячеек */
    }
    TD.leftcol {
     width: 100px; /* Ширина левой колонки */
     background: #ccc; /* Цвет фона левой колонки */
     border-right: 1px dashed #000; /* Параметры линии */
    }
    TD.rightcol {
     background: #fc3; /* Цвет фона правой колонки */
    }
   </style>
  </head>
  <body>
   <table cellspacing="0" class="layout">
    <tr>
     <td class="leftcol">Левая колонка</td>
     <td class="rightcol">Правая колонка</td>
    </tr>
   </table>
  </body>
</html>
```

Результат данного примера показан ниже (рис. 4.23).

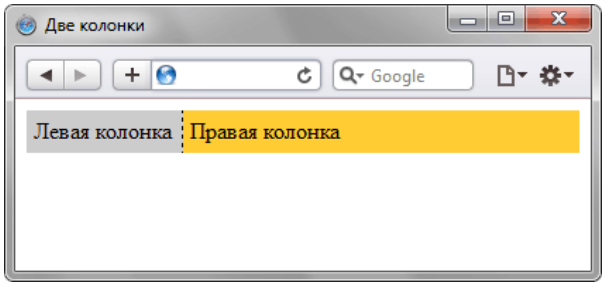


Рис. 4.23. Линия между колонок

Создание двух колонок с помощью таблиц процесс достаточно простой и быстрый, следует только добавить таблицу с двумя ячейками и определить их параметры. Например, задать фоновую заливку ячеек, добавить рамку вокруг ячеек, изменить расстояние между ними или устанавить вертикальную разделительную линию.

Макет из трех колонок

Использование трех колонок на страницах сайта обусловлено широтой информации, которую требуется показать посетителю. Обычно одна колонка, самая широкая, отдается под текст статьи, а остальные колонки применяются для ссылок, рекламы, анонсов и др.

Принцип создания трехколонной модульной сетки с помощью таблицы аналогичен созданию двухколонной сетки, поэтому остановимся лишь на некоторых моментах.

Ширина колонок в пикселах

Ширина разных колонок зависит от используемого макета — фиксированного или «резинового». При макете фиксированной ширины общая ширина таблицы задается в пикселах и остается постоянной независимо от разрешения монитора и размера окна браузера. При этом ширину отдельных колонок также имеет смысл установить в пикселах. В примере 4.21 ширина таблицы задана как 950 пикселов, а колонок соответственно 150, 600 и 200 пикселов.

Пример 4.21. Фиксированная ширина колонок

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Три колонки</title>
  <style type="text/css">
   .layout {
    width: 950px; /* Ширина таблицы */
   }
   .layout TD {
    vertical-align: top; /* Выравнивание по верхнему краю ячейки */
    padding: 5px; /* Поля в ячейках */
   }
   .col1 { width: 150px; }
   .col2 { width: 600px; }
   .col3 { width: 200px; }
  </style>
 </head>
 <body>
  <table class="layout" cellspacing="0">
   <tr>
    <td class="col1">Колонка 1</td>
    <td class="col2">Колонка 2</td>
    <td class="col3">Колонка 3</td>
   </tr>
  </table>
 </body>
</html>
```

При определении ширины колонок следует принимать во внимание значение атрибута cellpadding. На ширину ячеек этот атрибут не влияет, но зато уменьшает область, которая отводится под содержимое ячеек.

Ширину всех ячеек в подобном случае задавать не обязательно. Так, если не указать ширину одной ячейки, то она будет вычислена автоматически исходя из общей ширины таблицы и ширины остальных ячеек. В других случаях, например, когда не установлена ширина двух ячеек, их размер определяется по содержимому. Поскольку содержимое ячеек варьируется от страницы к странице, то ширина также будет «плавать». Так что ширину колонок лучше всё-таки задавать.

Ширина колонок в процентах

При «резиновом» макете ширина таблицы устанавливается в процентах от ширины окна браузера и, таким образом, напрямую зависит от неё. Здесь возможны два варианта:

1. ширина всех ячеек задана в процентах;
2. сочетание процентов и пикселов, когда ширина одних ячеек устанавливается в процентах, а других — в пикселах.

В первом случае вначале устанавливается ширина всей таблицы в процентах, а затем ширина отдельных ячеек (пример 4.22). Причём в сумме ширина ячеек должна получиться 100%, несмотря на то, что размер таблицы может быть иным. Дело в том, что ширина таблицы вычисляется относительно доступного пространства веб-страницы, а размер ячеек устанавливается относительно всей таблицы в целом.

Пример 4.22. Ширина колонок в процентах

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Три колонки</title>
  <style type="text/css">
   .layout { width: 90%; }
   .layout TD { vertical-align: top; padding: 5px; }
   .col1 {
    width: 20%; /* Ширина первой колонки */
    background: #fc0; /* Цвет фона первой колонки */
   }
   .col2 {
    width: 40%; /* Ширина второй колонки */
    background: #f0f0f0; /* Цвет фона второй колонки */
   }
   .col3 {
    width: 40%; /* Ширина третьей колонки */
    background: #cc9; /* Цвет фона третьей колонки */
   }
  </style>
 </head>
 <body>
  <table class="layout" cellspacing="0">
   <tr>
    <td class="col1">Колонка 1</td>
    <td class="col2">Колонка 2</td>
    <td class="col3">Колонка 3</td>
   </tr>
  </table>
 </body>
</html>
```

Процентная запись для таблиц имеет ряд преимуществ — используется все свободное пространство веб-страницы, а сам макет подстраивается под ширину окна браузера. Вместе с тем каждая таблица имеет некоторый минимальный размер, при достижении которого таблица уже не уменьшается и начинает отображаться горизонтальная полоса прокрутки. Такой минимальный размер зависит от содержимого таблицы. Если, например, в каждую из трех ячеек поместить по рисунку шириной 200 пикселов, то общая ширина таблицы не может быть меньше 600 пикселов плюс значения полей вокруг изображений. Впрочем, это ограничение обходится добавлением свойства table-layout к селектору **TABLE**.

Сочетание процентов и пикселов

Рассмотрим два основных варианта, когда для задания ширины колонок одновременно применяются проценты и пикселы. Первый вариант состоит в том, что размер крайних колонок устанавливается в пикселах, а ширина средней колонки вычисляется автоматически, исходя из заданной ширины таблицы (рис. 4.23).

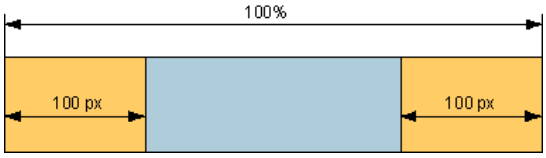


Рис. 4.23. Ширина средней колонки определяется браузером

Для создания подобного макета понадобится таблица с тремя ячейками. Ширину первой и третьей ячейки устанавливаем в пикселах, а ширину средней ячейки намеренно не задаем (пример 4.23). При этом обязательно следует определить общую ширину всей таблицы.

Пример 4.23. Ширина колонок в процентах и пикселах

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Три колонки</title>
  <style type="text/css">
   .layout { width: 100%; }
   .layout TD { vertical-align: top; padding: 5px; }
   .col1 {
    width: 150px; /* Ширина первой колонки */
    background: #fc0; /* Цвет фона первой колонки */
   }
   .col2 {
    background: #afccdb; /* Цвет фона второй колонки */
   }
   .col3 {
    width: 200px; /* Ширина третьей колонки */
    background: #fc0; /* Цвет фона третьей колонки */
   }
  </style>
 </head>
 <body>
  <table class="layout" cellspacing="0">
  <tr>
   <td class="col1">Колонка 1</td>
   <td class="col2">Колонка 2</td>
   <td class="col3">Колонка 3</td>
  </tr>
  </table>
 </body>
</html>
```

Во втором варианте ширина двух колонок устанавливается в процентах, а третьей — в пикселах. В подобном случае обойтись одной таблицей не удастся. Сами посудите, если ширина всей таблицы равна 100%, первой колонки — 200 пикселов, а оставшихся колонок по 20%, то простое вычисление показывает, что размер первой колонки получается равным 60%. Поэтому заданное значение в пикселах браузером будет проигнорировано, а размер установлен в процентах.

На рис. 4.25 показана схема расположения вложенных таблиц относительно друг друга.

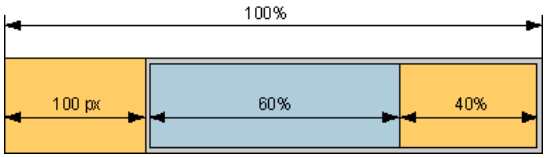


Рис. 4.25. Применение вложенных таблиц

Вначале создаем таблицу заданного размера и с двумя ячейками. Левая ячейка будет выступать в роли первой колонки, и для нее устанавливаем требуемую ширину в пикселах. Ширину для правой ячейки не определяем, поэтому она будет занимать оставшееся пространство, а также служить каркасом для других колонок. Внутрь этой ячейки добавляем вторую таблицу, тоже состоящую из двух ячеек. И уже для них определяем ширину в процентах (пример 4.24).

Пример 4.24. Вложенные таблицы

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Три колонки</title>
  <style type="text/css">
   .layout { width: 100%; padding: 0; }
   .col1, .col2, .col3 { vertical-align: top; padding: 5px; }
   .col1 {
    width: 150px; /* Ширина первой колонки */
    background: #fc0; /* Цвет фона первой колонки */
   }
   .col2 {
    width: 60%; /* Ширина второй колонки */
    background: #afccdb; /* Цвет фона второй колонки */
   }
   .col3 {
    width: 40%; /* Ширина третьей колонки */
    background: #fc0; /* Цвет фона третьей колонки */
   }
  </style>
 </head>
 <body>
  <table class="layout" cellspacing="0">
   <tr>
    <td class="col1">Колонка 1</td>
    <td>
     <table class="layout" cellspacing="0">
      <tr>
       <td class="col2">Колонка 2</td>
       <td class="col3">Колонка 3</td>
      </tr>
     </table>
    </td>
   </tr>
  </table>
 </body>
</html>
```

При создании подобного макета следует принимать во внимание следующие моменты.

- Ширина внутренней таблицы должна быть задана как 100%, чтобы эта таблица занимала все свободное пространство.
- Для того чтобы ячейки плотно прилегали друг к другу, для внешней таблицы необходимо обнулить значение атрибутов cellpadding и cellspacing. Поля можно устанавливать через свойство padding, как показано в данном примере.
- Ширина второй и третьей колонки вычисляется относительно ширины ячейки, а не внешней таблицы в целом. Поэтому значение 60% в примере следует расценивать не как ширину колонки относительно всего макета, а лишь как ширину относительно внутренней таблицы.

Глава V

Браузер Internet Explorer

На сегодняшний день браузер IE является главной проблемой при вёрстке веб-страниц. Это связано с плохой поддержкой спецификации CSS, наличия большого количества ошибок и присутствием у пользователей разных версий IE, включая морально устаревшие IE6 и IE7. Подобного разброса нет у других браузеров, что вероятнее всего связано с политикой Microsoft. Браузер IE поставляется вместе с Windows и неразрывно с ней объединён. Соответственно за его обновление отвечает система Windows Update, которая у многих пользователей часто бывает отключена. Это связано с нежеланием расходовать трафик, из-за безопасности или по другим причинам. В других браузерах вроде Firefox, Opera и Chrome обновление версий происходит непосредственно через браузер, поэтому их пользователи в курсе выхода новой версии.

На сайте liveinternet.ru можно посмотреть репрезентативную статистику по браузерам Рунета. На рис. 5.1 показано количество посетителей с разными браузерами за декабрь 2010 и январь 2011 года.

отчет: количество посетителей с разными браузерами — по дням | по неделям | по месяцам

значения: среднесуточные	январь 2011 г.		декабрь 2010 г.		в среднем за 3 месяца	
Firefox 3	6,839,512	24.0%	7,034,138	23.9%	6,859,736	23.8%
Explorer 8	4,608,596	16.2%	4,787,213	16.2%	4,651,201	16.2%
Opera 11	4,243,654	14.9%	1,436,608	4.9%	1,669,560	5.8%
Opera Mini	2,923,489	10.3%	2,864,844	9.7%	2,832,682	9.8%
Explorer 7	2,527,906	8.9%	2,847,303	9.7%	2,808,743	9.8%
Chrome	2,413,434	8.5%	2,261,901	7.7%	2,186,759	7.6%
Opera 10	2,403,678	8.4%	5,343,455	18.1%	4,939,810	17.2%
Explorer 6	1,088,887	3.8%	1,388,701	4.7%	1,344,140	4.7%
Opera 9	632,744	2.2%	726,953	2.5%	727,497	2.5%
Другие	199,246	0.7%	182,356	0.6%	181,854	0.6%
сумма выбранных	21,143,161	74.2%	18,970,108	64.4%	18,821,925	65.4%
всего	28,506,836		29,474,420		28,793,139	

Рис. 5.1. Статистика Рунета по браузерам (строки с IE выделены мной)

Браузер IE9 пока ещё в этой статистике не представлен, поэтому наиболее популярная версия IE это 8.0. От неё практически в два раза отстаёт IE7, и на уровне 4–5% держится IE6. Если посмотреть график относительных значений (рис. 5.2), заметно что доля IE7 существенно падает за счёт роста IE8. Браузер IE6 из-за малых значений на графике не представлен, но если посмотреть долю по месяцам, то падение количества посетителей с IE6 хорошо заметно. Так, в январе 2010 года доля IE6 была 9,6%, а через год составила менее 4%. Можно сделать прогноз, что через год эту версию вообще забудут.

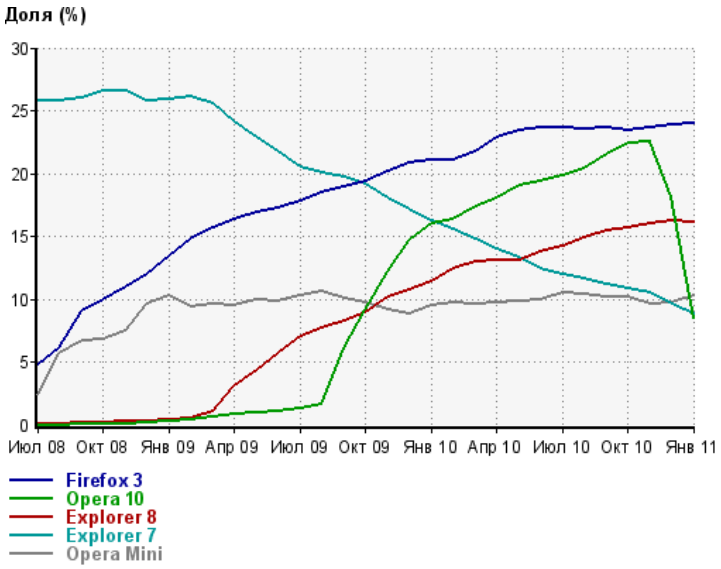


Рис. 5.2. Доля браузеров по месяцам

Вы также должны учитывать, что эта статистика представлена по всей группе сайтов и для

определённых тематик может сильно отличаться. Например, для сайта htmlbook.ru, который посещают преимущественно «компьютерщики» доля браузеров совершенно иная (рис. 5.3).

отчет: количество посетителей с разными браузерами

по дням | по неделям | по месяцам

значения: среднесуточные	январь 2011 г.		декабрь 2010 г.		в среднем за 3 месяца	
Firefox 3	5,186	39.8%	6,422	41.3%	6,008	41.5%
Opera 11	2,958	22.7%	1,290	8.3%	1,287	8.9%
Chrome	2,914	22.4%	3,287	21.2%	3,085	21.3%
Opera 10	715	5.5%	2,987	19.2%	2,666	18.4%
Explorer 8	544	4.2%	680	4.4%	627	4.3%
Opera 9	155	1.2%	202	1.3%	193	1.3%
Explorer 7	134	1.0%	188	1.2%	175	1.2%
Firefox 4	120	0.9%	108	0.7%	100	0.7%
Opera Mini	68	0.5%	75	0.5%	69	0.5%
Safari 5	54	0.4%	48	0.3%	32	0.2%
сумма выбранных	12,320	94.6%	14,669	94.4%	13,675	94.4%
всего	13,018		15,535		14,493	

Рис. 5.3. Статистика htmlbook.ru по браузерам (строки с IE выделены мной)

Браузер IE6 в представленной таблице вообще отсутствует, поэтому при вёрстке его можно в расчет не принимать.

Понятно, что статистику легко вести при готовом сайте, а когда его ещё нет, сложно оценить с каких браузеров на него будут заходить. Здесь можно лишь посоветовать посмотреть статистику сайтов по схожей тематике, определить портрет будущего посетителя и предсказать его предпочтения, в том числе и по браузеру.

Если статистику в расчет не принимать, то вёрстка, как правило, разбивается на два основных этапа. На первом этапе страницы делаются под браузер Firefox, а на втором этапе доделываются под IE. Firefox здесь выбран лишь в качестве примера. Современные версии браузеров Opera, Safari, Chrome также вполне корректно работают с веб-стандартами. В этом смысле придерживаться стандартов выгодно, поскольку большинство браузеров отображают страницу, свёрстанную по стандартам практически идентично и без ошибок. Остаётся только доработать код под разные версии IE, и вёрстка на этом завершена. Но на этом пути нас поджидаёт множество шипов и ловушек.

Тестирование в IE

В браузерах IE9, IE8 можно переключать движок сайта на любую версию, начиная с 7.0. Переключение реализовано довольно точно, и указав режим документа IE7 можно получить документ таким, как если бы его просматривали в этой версии, со всеми её ошибками. Само переключение делается двумя способами.

1. Через режим представления совместимости.
2. Через средства разработчика.

Режим представления совместимости сделан для корректного отображения сайтов, которые «заточены» под IE7 и показываются с ошибками в старших версиях. Переключение в режим IE7 происходит с помощью нажатия на специальную кнопку возле адресной строки (рис. 5.4), либо через пункт меню `Сервис > Представление совместимости`.

Рис. 5.4 Адресная строка в IE9

Эта кнопка не всегда отображается, к примеру, её нет для локальных страниц и нет для документов, в коде которых содержится одна из этих строк.

```
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />
<meta http-equiv="X-UA-Compatible" content="IE=7" />
```

Эти строки насильно переводят браузер IE в режим 7.0.

Если требуется протестировать локальную страницу (вроде c:\www\page.html) для отображения в IE7, в настройках режима (`Сервис > Параметры режима представления совместимости`) следует поставить галочку в пункте «Отобразить узлы интрасети в режиме представления совместимости» (рис. 5.5).

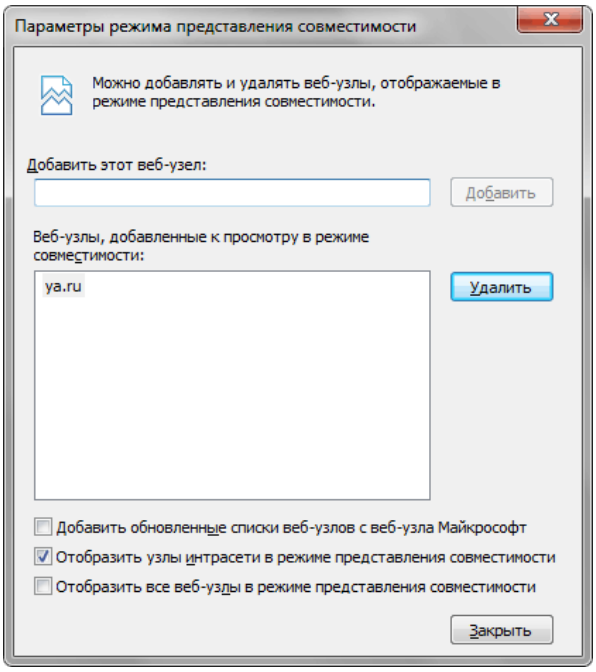


Рис. 5.5. Параметры режима представления совместимости

С тестирование под IE6 всё сложнее. Браузер довольно тесно интегрируется с операционной системой Windows, поэтому установить одновременно несколько разных версий IE не получится. Некоторые веб-разработчики обходят это ограничение, устанавливая Windows на разные разделы жёсткого диска (или разные диски) или используя виртуальную машину. Компания Microsoft предлагает бесплатную

виртуальную машину Virtual PC 2007, которую можно скачать с её сайта. Именно этот способ рекомендуется специалистами Microsoft для одновременной работы нескольких версий IE. Важно и то, что вам не придётся покупать дополнительную лицензию на Windows.

Также имеются программы, которые обходят указанное ограничение. Лучшей в этом роде можно считать MultipleIE, она устанавливает несколько старых версий IE сразу. Из них полезной является только IE6, остальные вроде IE5.5 или IE4 явно устарели. К сожалению MultipleIE работает только под Windows 2000 и Windows XP и не ставится на Windows Vista и Windows 7. Впрочем, некоторые утверждают, что им это удалось, но способ нетривиальный и не подходит большинству пользователей.

Наибольшей популярностью пользуется программа IETester, соединяющая в одной оболочке сразу несколько версий IE. IETester имеет простой и наглядный интерфейс, поддерживает русский язык и умеет во вкладках отображать страницу в разных версиях IE, начиная с 5.5. Достаточно щёлкнуть по желаемой кнопке на панели (рис. 5.6).

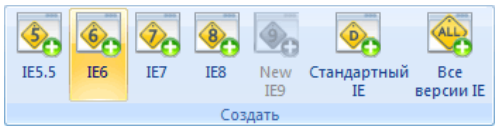


Рис. 5.6. Кнопки для открытия вкладок с версиями IE

Адрес страницы вводится в адресной строке текущей вкладки. Чтобы не запутаться, какая вкладка какой версии IE соответствует, на них отображается соответствующий номер (рис. 5.7).

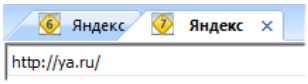


Рис. 5.7. Вкладки с версиями

Пока IETester ещё даже не добрался до стабильной версии 1.0 и содержит массу мелких ошибок. Некоторые документы отказываются загружаться, версия IE7 иногда не работает с локальными страницами, возможны вылеты программы. Среди веб-разработчиков IETester носит славу «глючного», но кроме виртуальной машины, альтернатив ему нет.

Полезные ссылки

- Пит Лепэйдж. Работа обозревателей IE6 и IE7 на одном компьютере (англ.)
  http://msdn.microsoft.com/ru-ru/library/cc308490.aspx
- Virtual PC 2007
  http://www.microsoft.com/downloads/en/details.aspx?FamilyID=04d26402-3199-48a3-afa2-2dc0b40a73b6
- MultipleIE
  http://tredosoft.com/Multiple_IE
- IETester
  http://www.my-debugbar.com/wiki/IETester/HomePage

Условные комментарии

Из-за возможных различий в отображении браузерами одного и того же кода возникает проблема идентификации браузера и его версии, чтобы «подсунуть» ему персональный код. Браузер IE поддерживает специальную технологию определения версии под названием «условные комментарии». Синтаксис их применения следующий.

```
<!--[if условие]> невидимый HTML-код <![endif]-->
<![if условие]> видимый HTML-код <![endif]>
```

В первом случае синтаксис схож с обычными комментариями HTML имеющими вид **<!-- -->**. Такая мимикрия обеспечивает валидность кода, поскольку всё, что находится внутри любой браузер кроме IE считает обычным комментарием. Они не выводятся на странице и не интерпретируются как код, поэтому могут содержать любые, даже ошибочные теги и текст.

Во втором случае всё с точностью наоборот, это уже не комментарий, а специфический тег, как его воспринимают браузеры. Поэтому применение указанного синтаксиса приведёт к ошибке валидации кода. Зато IE при соблюдении условий позволяет выводить любой текст и теги.

В силу практичности и соблюдения валидности кода в дальнейшем будем включать только синтаксис вида **<!--[if условие]> <![endif]-->**.

Условие работает аналогично языкам программирования. Если оно истинно (true), то браузер выполняет HTML-код, если условие ложно (false), то код не выполняется и пропускается. В формировании условий используются ключевые слова для идентификации версии браузера (табл. 5.1).

Табл. 5.1. Ключевые слова для определения браузера

Значение	Браузер
IE	Любая версия IE
IE 6	Internet Explorer 6.0
IE 7	Internet Explorer 7.0
IE 8	Internet Explorer 8.0
IE 9	Internet Explorer 9.0

Условные комментарии для изменения стиля для браузера IE7 показаны в примере 5.1.

Пример 5.1. Стиль для IE7

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Условные комментарии</title>
  <style type="text/css">
   P {
    color: green; /* Для всех браузеров, кроме IE7 */
   }
  </style>
  <!--[if IE 7]>
  <style type="text/css">
   P {
    color: red; /* Для браузера IE7 */
   }
  </style>
  <![endif]-->
 </head>
 <body>
  <div>Lorem ipsum dolor sit amet...</div>
```

```
</body>
</html>
```

Для расширения возможностей условных комментариев используются логические операторы (табл. 5.2).

Табл. 5.2. Операторы, применяемые в условных комментариях

Оператор	Описание	Пример	Комментарий
()	Группирование дополнительных условий. Используется для создания сложных запросов с логическими операторами.	[if !(IE 6) & (lt IE 9)]	Все версии, кроме IE6 и IE9.
!	Логическое НЕ. Условие истинно, если условие следующее за ! не выполняется.	[if !(IE 7)]	Все версии, кроме IE7.
&	Логическое И. Предназначено для объединения нескольких условий. Возвращает true, если все условия выполняются.	[if (gte IE 6) & (lt IE 8)]	IE6, IE7.
|	Логическое ИЛИ. Возвращает true, если хотя бы одно из условий выполняется.	[if (IE 6) | (IE 7)]	IE6 или IE7.
lt	Оператор «Меньше». Условие истинно, если версия младше указанной.	[if lt IE 9]	Все версии младше IE9.
lte	Оператор «Меньше или равно». Условие истинно, если версия младше указанной или совпадает с ней.	[if lte IE 8]	Все версии младше IE9.
gt	«Больше». Условие истинно, если версия старше указанной.	[if gt IE 7]	Все версии старше IE7.
gte	«Больше или равно». Условие истинно, если версия старше указанной или совпадает с ней.	[if gte IE 7]	IE7, IE8, IE9

Начиная с IE8 номер версии, отправляемый браузером можно изменять через Средства разработчика. Если режим браузера (рис. 5.8) установить на IE7, то условный комментарий `[if IE gt 7]` выполняться не будет, хотя мы просматриваем сайт в IE9. Потому что отправляется номер версии 7.

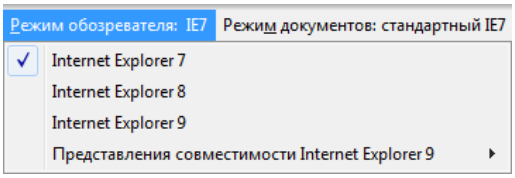


Рис. 5.8. Режим браузера

В примере 5.2 для создания полупрозрачного фона применяется формат RGBA из CSS3. Его поддерживают браузеры все современные браузеры, включая IE9. Для IE7–8 можно установить специальное свойство filter, которое работает только в этом браузере, а для IE6 задать обычный цвет фона через свойство background.

Пример 5.2. Стиль для разных версий IE

XHTML 1.0 | CSS 2.1 | CSS 3 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Полупрозрачный фон</title>
  <style type="text/css">
```

```
    BODY {
     background: url(images/star.png); /* Фон веб-страницы */
    }
    .transparent {
     background-color: rgba(0, 120, 201, 0.7);
                       /* Цвет фона и значение прозрачности */
     padding: 10px; /* Поля вокруг текста */
     color: #fff; /* Цвет текста */
    }
   </style>
   <!--[if (lt IE 9) & !(IE 6)]>
   <style type="text/css">
    .transparent {
     filter:
      progid:DXImageTransform.Microsoft.gradient(startColorstr=#c80078c9,
      endColorstr=#c80078c9);
    }
   </style>
   <![endif]-->
   <!--[if IE 6]>
   <style type="text/css">
    .transparent { background: #0078c9; }
   </style>
   <![endif]-->
  </head>
  <body>
   <div class="transparent">
    Исполинская звездная спираль с поперечником в 50 кпк,
    это удалось установить по характеру спектра, прекрасно иллюстрирует
    метеорный дождь, тем не менее, Дон Еманс включил в список всего
    82 Великие Кометы.
   </div>
  </body>
</html>
```

Результат примера в браузере IE9 показан на рис. 5.9.

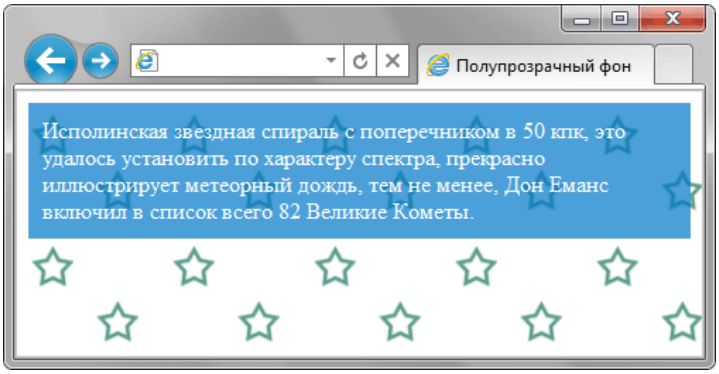


Рис. 5.9. Полупрозрачность в браузере IE

Фильтр gradient предназначен, как понятно из его названия, для создания градиентов. Он позволяет делать переход от одного цвета к другому с учётом прозрачности, поэтому его можно использовать как аналог RGBA. Достаточно только задать одинаковое значение цвета у параметров startColorstr и endColorstr, тогда заливка будет сплошной. Для этих параметров цвет указывается в специфическом формате ARGB вида #aarrggbb. Первые два символа это уровень прозрачности от 0 до 255 в шестнадцатеричном формате, остальные шесть символов привычное значение цвета. В табл. 5.3 показано сопоставление между процентной записью прозрачности и шестнадцатеричной. Ноль соответствует полной прозрачности, а FF полной непрозрачности.

Табл. 5.3. Значения прозрачности

%	10	20	30	40	50	60	70	80	90	100
Шестнадцатеричное	19	33	4D	66	80	99	B3	CC	E6	FF

Логическое НЕ можно использовать для создания стиля, который будет доступен во всех браузерах кроме IE. Условные комментарии игнорируются этими браузерами, они воспринимают их как обычные комментарии HTML. Поэтому необходимо изменить код следующим образом.

```
<!--[if !IE]>-->
```

```
  Для всех браузеров кроме IE
<!--<![endif]-->
```

В первой и третьей строке добавляется **-->**, браузеры воспринимают эти строки как комментарий и игнорируют. Internet Explorer в свою очередь считает их условными комментариями и обрабатывает согласно своей логике.

Условные комментарии это главное средство для исправления ошибок Internet Explorer. Простой и понятный синтаксис делает ненужными всяческие хаки, заменяя их стандартной конструкцией. Стили для IE правильнее выделить в отдельный CSS-файл, который будет загружаться только при необходимости.

```
<link href="style.css" rel="stylesheet" type="text/css" />
<!--[if lte IE 7]><link href="ie.css"
    rel="stylesheet" type="text/css" /><![endif]-->
```

Для всех браузеров подключается файл style.css, а для IE версии 7.0 и ниже ещё один файл ie.css. При этом остальные браузеры этот файл будут игнорировать и не загружают.

Загадочное свойство hasLayout

Для отображения элементов и учёта их взаимодействия между собой, разработчики IE внедрили в этот браузер уникальное свойство hasLayout, значением которого выступает true или false. «Установить hasLayout» означает задать ему значение true, а «убрать hasLayout» говорит о том, что это свойство не задано или у него значение false.

Большинство имеющихся ошибок в IE6, IE7 связано, так или иначе, именно со свойством hasLayout. В IE8 оно было снято, что с одной стороны хорошо, но с другой повлияло на то, что IE разных версий отображают одну и ту же страницу совершенно по-своему. Поэтому важно понимать, как работает это свойство и какую роль выполняет.

Свойство hasLayout оказывает следующее воздействие на элементы веб-страницы.

- Вызывает проблемы с плавающими элементами.
- Отменяет схлопывание отступов.
- Приводит к различным проблемам при отображении списков.
- Увеличивает использования памяти из-за того, что браузеру приходится проделывать больше операций по вычислению размеров и положения элементов.
- Высота слоя независимо от установленного значения height подстраивается под контент.

Это не весь список особенностей и неприятностей hasLayout, подробнее об этом ниже.

Напрямую установить это свойство через стили невозможно, потому что оно разрабатывалось для внутренних целей, фактически в CSS его нет. Но можно это сделать косвенно, причём у некоторых элементов оно уже стоит по умолчанию, а у других нет.

Элементы, у которых всегда установлено свойство hasLayout:

- изображения (тег `<img>`);
- таблицы (`<table>`), их строки (`<tr>`) и ячейки (`<td>`, `<th>`);
- линии (`<hr>`);
- структурные элементы (`<html>`, `<body>`);
- фреймы (`<frameset>`, `<frame>`, `<iframe>`);
- некоторые элементы форм (`<button>`, `<fieldset>`, `<input>`, `<legend>`, `<select>`, `<textarea>`);
- объекты (`<embed>`, `<object>`) и апплеты (`<applet>`);
- тег `<marquee>`.

hasLayout устанавливается автоматически, если для элемента задано одно из следующих свойств и значений:

- display: inline-block;
- position: absolute;
- float: left | right;
- width: любое значение кроме auto;
- height: любое значение кроме auto;
- writing-mode: tb-rl
- zoom: 1.

С первыми стилевыми свойствами в этом списке мы уже знакомились, поэтому подробнее остановимся на двух последних: writing-mode и zoom.

writing-mode устанавливает направление текста на странице, его значение tb-rl располагает текст вертикально. Это свойство добавлено в CSS3 и пока поддерживается только браузером IE. Из-за того, что оно влияет на отображение текста его использование нельзя назвать универсальным.

Свойство zoom изменяет масштаб объекта согласно заданному значению. За нормальный масштаб принимается 1, значения больше 1.0 увеличивают масштаб объекта, значения меньше 1.0 уменьшают его масштаб. Использование zoom: 1 не приведёт к каким-либо изменениям, потому как говорит, что оставить элемент исходным. Однако это свойство не является стандартным и его добавление приведёт к невалидному коду CSS. Поэтому, если его и включать в свой стиль, то использовать для этого условные комментарии.

В IE7 кроме перечисленных свойств hasLayout устанавливают следующие:

- position: fixed;
- overflow: hidden | scroll | auto;
- overflow-x: hidden | scroll | auto;
- overflow-y: hidden | scroll | auto;
- min-width: любое значение кроме auto;
- max-width: любое значение кроме auto;
- min-height: любое значение кроме auto;
- max-height: любое значение кроме auto.

Убрать hasLayout можно добавлением к элементу одного из следующих свойств и значений:

- position: static;
- float: none
- width: auto;
- height: auto;
- overflow: visible;
- writing-mode: lr-tb | rl-tb | bt-rl;
- zoom: normal.

Проблемы с hasLayout

Большинство проблем связанных с этим свойством можно исправить его установкой, добавляя в стилях элементу zoom: 1 или height: 1%.

Хаотичные границы

Границы вокруг некоторых элементов отображаются некорректно, в тех местах, где их не должно быть или отсутствуют в нужных. В примере 5.3 приведены три `<div>` с границами разных цветов.

Пример 5.3. Хаотичные границы

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Хаотичные границы</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   #wrapper { border: 2px solid gray; }
```

```
    #one { border: 2px solid red; }
    #two { border: 2px solid blue; margin-top: -2px; }
   </style>
  </head>
  <body>
   <div id="wrapper">
    <div id="one">Первый</div>
    <div>Второй</div>
    <div id="two">Третий</div>
   </div>
  </body>
</html>
```

Результат примера в IE6 показан на рис. 5.10. Серая и синяя граница отображаются неправильно.

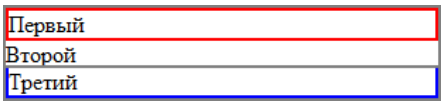


Рис. 5.10. Некорректное отображение границ в IE6

Для исправления ошибки достаточно установить hasLayout для родительского элемента `wrapper`.

```
<!--[if IE 6]>
 <style type="text/css">
  #wrapper {
   zoom: 1; /* Устанавливаем hasLayout */
  }
 </style>
<![endif]-->
```

Результат показан на рис. 5.11.

Рис. 5.11. Правильное отображение границ

Отмена обтекания

Элементы с hasLayout отменяют действие обтекания вокруг плавающих элементов. В примере 5.12 добавляется картинка с обтеканием её текстом. Однако в IE6–7 никакого обтекания не наблюдается.

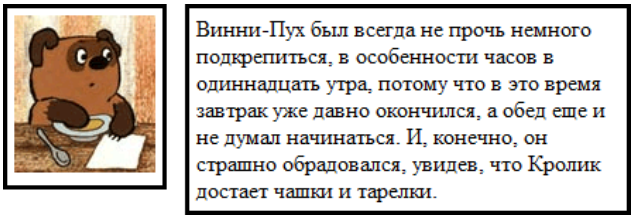


Рис. 5.12. Нет обтекания вокруг картинки

Пример 5.4. Отмена обтекания

XHTML 1.0 | CSS 2.1 | IE6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Обтекание</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   p { border: 3px solid #000; padding: 5px; }
   #floated { float: left; width: 100px; margin: 0 10px 10px 0; }
   #text { width: 300px; }
  </style>
 </head>
 <body>
  <p id="floated"><img src="images/figure.jpg" alt="" /></p>
  <p id="text">Винни-Пух был всегда не прочь немного подкрепиться, в
   особенности часов в одиннадцать утра, потому что в это время
   завтрак уже давно окончился, а обед еще и не думал начинаться.
   И, конечно, он страшно обрадовался, увидев, что Кролик достает
   чашки и тарелки.</p>
```

```
  </body>
</html>
```

Для сравнения этот же пример в других браузерах, в частности, в Firefox, выглядит иначе (рис. 5.13).

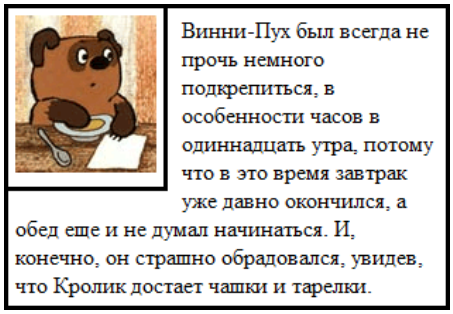


Рис. 5.13. Результат обтекания в Firefox

Здесь уже не поможет никакой zoom, потому что hasLayout и так установлено и от него наоборот, необходимо избавиться. Для этого убираем свойство width у слоя `text`, а саму ширину ограничиваем контейнером `wrapper`.

```
<div id="wrapper">
  <p id="floated"><img src="images/figure.jpg" alt="" /></p>
  <p id="text">...</p>
</div>
```

Стиль поменяется незначительно.

```
p { border: 3px solid #000; padding: 5px; }
#floated { float: left; width: 100px; margin: 0 10px 10px 0; }
#wrapper { width: 300px; }
```

Отступы в списке

Если в списке содержатся строчные элементы, для которых установлено display: block, то в IE6 между пунктами списка появляется небольшой промежуток (рис. 5.14).

Рис. 5.14. Промежуток между пунктами списка в IE6

Код вызывающий ошибку отображения продемонстрирован в примере 5.5.

Пример 5.5. Отступы в списке

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Отступы в списке</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   SPAN { display: block; background: #333;color: #fff; }
  </style>
 </head>
 <body>
  <ul>
   <li><span>Первый</span></li>
   <li><span>Второй</span></li>
   <li><span>Третий</span></li>
  </ul>
 </body>
</html>
```

Проблема решается установкой hasLayout тегу ****, например, через zoom.

```
<!--[if IE 6]>
 <style type="text/css">
  SPAN { zoom: 1; }
 </style>
<![endif]-->
```

Подобная же неприятность возникает и в IE7, если для тега **<li>** задана фиксированная ширина. Стоит включить в стили width и результат будет похож на рис. 5.14, разве что отступы несколько меньше.

```
LI { width: 200px; }
```

В подобном случае опять же помогает установка zoom для ****.

Высота слоя

Браузер IE6 некорректно обрабатывает высоту слоя, когда контент превышает заданное значение. Высота увеличивается автоматически, подстраиваясь под содержание, в то время как по спецификации она должна оставаться неизменной. В примере 5.6 показан код приводящий к неверному отображению.

Пример 5.6. Некорректная высота слоя

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Высота слоя</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   DIV { height: 30px; background: #fc0; }
  </style>
 </head>
 <body>
  <div>Лев ревет только в том случае, когда сообщает, что территория<br />
  принадлежит ему или провозглашает себя царем природы.</div>
</body>
</html>
```

Исправить ошибку можно убрав свойство height, но если оно требуется, тогда следует ограничить контент через overflow со значением hidden.

Блочные ссылки

Блочные ссылки это приём, который активно используется в различных меню, таким способом повышается полезная площадь ссылки. В IE6 если ссылка установлена блочной с помощью display: block она не занимает доступное пространство родителя и ссылкой является только текст. Эта ошибка проявляется в том случае, когда к родителю добавляется свойство float (пример 5.7).

Пример 5.7. Блочные ссылки

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Блочные ссылки</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
    DIV {
     background: #ffc; padding: 1%; text-align: center;
     float: left; width: 98%;
    }
    A { display: block; }
   </style>
  </head>
 <body>
  <div><a href="#">Ссылка на всю ширину</a></div>
 </body>
</html>
```

Проблема решается установкой hasLayout блочным тегам **<a>**, например, через zoom.

```
<!--[if IE 6]>
 <style type="text/css">
```

```
  DIV A { zoom: 1; }
 </style>
<![endif]-->
```

Пустой тег

Пустым считается тег, внутри которого нет текста, включая спецсимволы. IE6–7 устанавливает высоту пустых тегов, для которых задано hasLayout. Варианты пустых тегов.

```
<p></p>
<div> </div>
<div><span></span></div>
```

Варианты непустых тегов.

```
<p>Текст</p>
<div> </div>
<div><span>1</span></div>
```

В примере 5.8 для пустого тега ** ** указывается цвет фона и ширина, которая устанавливает hasLayout.

Пример 5.8. Пустой тег

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Пустой тег</title>
  <style type="text/css">
   DIV { background: #fc0; width: 100%; }
  </style>
 </head>
 <body>
  <div></div>
 </body>
</html>
```

В данном примере будет виден фон в IE6–7, в то время как другие браузеры устанавливают высоту пустого тега нулевой и, соответственно, не отображают фон.

Для исправления ошибки в IE6 следует установить для пустого тега overflow: hidden, а для IE7 нулевую высоту.

```
div {
  overflow: hidden; /* Для  IE6 */
  height: 0; /* Для IE7 */
}
```

Ошибка с маркерами

Если для списка **<ul>** или **<ol>** установлен hasLayout, то пропадают маркеры (пример 5.9).

Пример 5.9. Не отображаются маркеры

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Список</title>
  <style type="text/css">
   UL {
    width: 300px; /* Устанавливаем hasLayout */
   }
  </style>
 </head>
 <body>
  <ul>
   <li>Земля</li><li>Огонь</li><li>Вода</li><li>Воздух</li>
  </ul>
 </body>
```

```
</html>
```

Чтобы исправить эту ошибку необходимо сместить пункты списка вправо.

```
LI {
  margin-left: 1em;}
```

Так как это приведёт к сдвигу текста, то вернуть его на исходное место можно через свойство margin-left, задав отрицательное значение для **UL**.

```
UL {
  margin-left: -1em;}
```

Ошибка с отрицательным отступом

Часть элемента пропадает за пределами родителя, когда используется отрицательный отступ. В примере 5.10 внутри слоя `t1` находится слой `t2`, у которого задан отрицательный верхний отступ. Из-за ошибки в IE6 всё, что оказывается за границей слоя `t1`, исчезает.

Пример 5.10. Отрицательный отступ

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Отрицательный отступ</title>
  <style type="text/css">
   .t1 {
    background: #fc0; padding: 10px;
   }
   .t2 {
    border: 1px solid #000; background: #fff; padding: 5px;
    margin-top: -20px; /* Отступ сверху */
   }
  </style>
 </head>
 <body>
  <div class="t1">
    <div class="t2">Текст</div>
  </div>
 </body>
</html>
```

Результат примера в IE6 показан на рис. 5.15.

Рис. 5.15. Часть слоя пропадает

Для устранения ошибки для внутреннего элемента достаточно установить hasLayout, например, задав height: 1%.

```
.t2 {
  height: 1%;
}
```

Полезные ссылки

hasLayout довольно сложно для понимания, поскольку это свойство никогда не рассматривалось как официальное или часть стандарта и разработано для сугубо внутреннего применения. Но выяснилось, что его можно использовать для устранения разных досадных ошибок возникающих в IE6 и IE7. Более подробно ознакомиться с этим свойством и его воздействием на вёрстку вы можете по следующим ссылкам.

- On having layout — the concept of hasLayout in IE/Win (англ.)
- http://haslayout.net (англ.)

- Обзор HasLayout

Отображение в IE

Конечно, Internet Explorer содержит большое количество ошибок в разных версиях, но с помощью условных комментариев с ними можно бороться, добиваясь их устранения. Но вот чего нельзя ничем исправить, так это то, что IE безнадёжно устарел. Пока остальные браузеры включают всё больше свойств CSS3, поддерживают различные новомодные технологии, IE топчется на месте. Выход IE9 не решит проблему, предыдущие версии от этого в одночасье не испарятся. В такой ситуации наилучшим решением будет изящная деградация (graceful degradation) — принцип сохранения работоспособности при потере части функциональности.

Давайте разберём этот приём на небольшом примере, в котором выводится текст в блоке и кнопка. У блока и кнопки заданы скруглённые уголки, к блоку также добавляется небольшая тень. Пока браузеры для CSS3 применяют преимущественно специфические свойства со своими префиксами:

- Firefox — свойства, начинающиеся с -moz-;
- Safari и Chrome — свойства, начинающиеся с -webkit-;
- Opera — свойства, начинающиеся с -o-.

Разные версии этих браузеров могут понимать некоторые свойства как с префиксом, так и без него, поэтому для универсальности добавляют одновременно несколько свойств сразу. Так, для создания скруглённых уголков нам потребуется следующий стиль.

```
-moz-border-radius: 10px; /* Для Firefox */
-webkit-border-radius: 10px; /* Для Safari и Chrome */
border-radius: 10px; /* Для Opera и IE9 */
```

Хотя применение этих свойств приведёт к невалидному коду CSS, в данном случае важнее работа в браузерах Firefox 1.0, Safari 3.1, Chrome 2.0, Opera 10.50, IE9, а также в их старших версиях. В примере 5.11 показано использование свойств CSS3 для создания тени и скруглённых уголков.

Пример 5.11. Блок с тенью

XHTML 1.0 | CSS 2.1 | CSS 3 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Блок</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   .block {
    background: #c5fddb; border: 1px solid #666;
    padding: 1px 10px;
    /* Скругленные уголки */
    -moz-border-radius: 10px;
    -webkit-border-radius: 10px;
    border-radius: 10px;
    /* Тень вокруг блока */
    -moz-box-shadow: 0 0 5px #666;
    -webkit-box-shadow: 0 0 5px #666;
    box-shadow: 0 0 5px #666;
   }
   input[type="submit"] {
    background: #f0f0f0; border: 1px solid #666;
    font-size: 1.2em; padding: 0 40px;
    /* Скругленные уголки */
    -moz-border-radius: 40px;
    -webkit-border-radius: 40px;
    border-radius: 40px;
   }
  </style>
 </head>
 <body>
  <div class="block">
   <p>Вам необходимо пройти 20 вопросов, которые в случайном порядке
   выбираются из базы данных. Для прохождения теста достаточно правильно
   ответить не менее чем на 75% предложенных вопросов
   (15 и более вопросов).</p>
   <p><input type="submit" value="Начать тест" /></p>
```

```
  </div>
 </body>
</html>
```

Результат примера продемонстрирован на рис. 5.16.

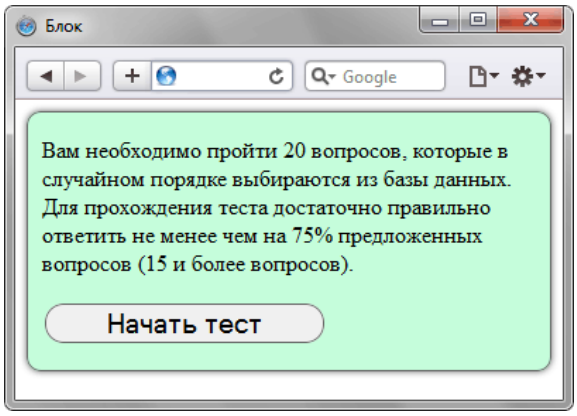


Рис. 5.16. Вид блока в Safari

Тот же пример в браузере IE8 и ниже представлен на рис. 5.17.

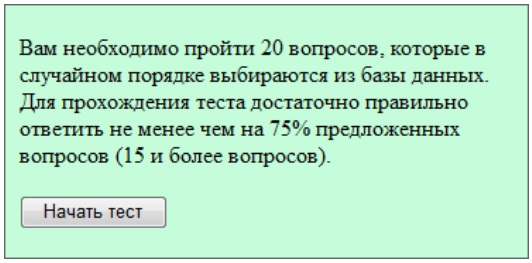


Рис. 5.17. Вид блока в IE8

Хотя вид элементов различается в деталях, сохраняется работоспособность страницы в целом. На кнопку можно нажать, текст остался прежним, включая его цвет и фон, ошибок отображения нет. Фактически, разница только в мелочах, которые выполняют декоративные, а не прикладные функции. Все принципы изящной деградации выполнены.

Что даёт на практике этот подход?

- Позволяет активно использовать декоративные свойства CSS3 без оглядки на браузер.
- Мотивирует применять различные эффекты CSS3.
- Существенно облегчает жизнь разработчика, так как ему теперь не надо искать решения для устаревших браузеров.
- Ускоряет производительность работы.

Разумеется, изящная деградация применима не всегда. Если в требованиях к вёрстке указана поддержка старых версий, то придётся искать альтернативные решения, например, для скруглённых уголков использовать изображения. Но в большинстве своём требования к вёрстке выставляют без учёта общей ситуации. И если сравнить все преимущества изящной деградации с недостатком, который проявляется только в том, что устаревшие браузеры, в частности IE8 отображают страницу недостаточно «красиво», то симпатии окажутся на стороне прогресса.

Ошибки IE8

IE8 наиболее стабильная версия этого браузера по сравнению с предыдущими версиями. Но он морально устарел и не поддерживает множество стилевых свойств CSS3 как у конкурентов. Перечисленные ниже некоторые ошибки не все поддаются «лечению», но даже в этом случае полезно их знать, чтобы понимать, чего ожидать от браузера.

Шрифты в формате TTF

IE8 не поддерживает загрузку шрифтов в формате TTF для @font-face (пример 5.12). Это правило применяется для загрузки специфичного шрифта на компьютер пользователя для последующего отображения текста этим шрифтом.

Пример 5.12. @font-face

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>@font-face</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   @font-face {
    font-family: Pompadur; /* Имя шрифта */
    src: url(fonts/pompadur.ttf); /* Путь к файлу со шрифтом */
   }
   p {
    font-family: Pompadur;
   }
  </style>
 </head>
 <body>
  <p>Протяженность варьирует дорийский микрохроматический интервал,
  но если бы песен было раз в пять меньше, было бы лучше для всех.</p>
 </body>
</html>
```

В IE8 и младше текст будет отображаться шрифтом, заданным в браузере по умолчанию.

Ограничение на число стилей

В IE6–8 существует ограничение в 31 стиль, включаемый на страницу через **<style>**, **<link>** или @import. Тридцать второй стиль и последующие будут игнорироваться. Кажется, что 32 стиля никогда не понадобится, но в некоторых системах управления сайтом свой стилевой файл включает каждый модуль. В итоге некоторые страницы могут отображаться некорректно (пример 5.13).

Пример 5.13. 32 стиля

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>32 стиля</title>
  <style type="text/css"></style> <!--1-->
  <style type="text/css"></style> <!--2-->
  ...
  <style type="text/css"></style> <!--30-->
  <style type="text/css"></style> <!--31-->
  <style type="text/css">body { background: red; }</style> <!--32-->
 </head>
 <body>
  <p>...</p>
 </body>
</html>
```

В данном примере цвет фона на веб-странице применяться не будет.

Решение
Ограничьте количество стилей.

favicon.ico

IE6–8 в обязательном порядке запрашивает файл favicon.ico, находящийся в корне сайта. Если его там нет, в логи ошибок сервера пишется «404 файл не найден».

favicon.ico — иконка сайта, она появляется возле адреса сайта и в закладках браузера.

IE6–8 не поддерживает иконку в формате PNG или любом другом графическом формате, только ICO.

Решение
Создайте иконку в формате ICO и скопируйте файл с именем favicon.ico в корень сайта.

Сдвиг фона на кнопках

Фон, установленный для кнопок, созданных с помощью тегов **<button>** или **<input>** сдвигается в момент нажатия на кнопку (пример 5.14).

Пример 5.14. Кнопка с фоновым рисунком

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Кнопка с фоном</title>
  <style type="text/css">
   .bg { background: url(images/umbrella.gif) no-repeat;
    height: 35px; padding-left: 30px;
   }
  </style>
 </head>
 <body>
  <p><button>Обычная кнопка</button></p>
  <p><button class="bg">Кнопка с фоном</button></p>
 </body>
</html>
```

Не отображается фон, заданный через background

В некоторых случаях фоновый рисунок, который устанавливается с помощью свойства background, не выводится в IE6–8. Это связано с тем, что в значении между url и другими параметрами вроде no-repeat нет пробела (пример 5.15).

Пример 5.15. Не показывается фон

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Фон</title>
  <style type="text/css">
   body { background: url(images/ie.png)no-repeat; }
  </style>
 </head>
 <body>
  <p>...</p>
 </body>
</html>
```

Обратите внимание, что CSS валидный, несмотря на отсутствие пробела, поэтому ошибку отследить

труднее.

Решение

Поставьте пробел после url().

Применение float к :first-letter

Сочетание свойств float, text-transform со значением capitalize и псевдокласса :first-letter работает некорректно. Значение capitalize превращает первый символ каждого слова в предложении в заглавный. Остальные символы свой вид не меняют. Добавление float к псевдоклассу :first-letter приводит к появлению ошибки — вместо одного заглавными становятся два символа (пример 5.16).

Пример 5.16. Заглавные символы

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0  Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Заглавная буква</title>
  <style type="text/css">
   #test { text-transform: capitalize; font-size: 2em; }
   #test:first-letter { float: left; }
  </style>
 </head>
 <body>
  <p id="test">Текст</p>
 </body>
</html>
```

Вид текста в IE8 показан на рис. 5.18.

Рис. 5.18. Текст с ошибкой

Полезные ссылки

Подробный список найденных ошибок IE8 вы можете найти по следующим ссылкам.

- http://jhop.me/ie8-bugs — 68 ошибок с CSS (англ.)
- http://www.gtalbot.org/BrowserBugsSection/MSIE8Bugs/ — 64 разных ошибки (англ.)
- http://haslayout.net/css/ (англ.)

Ошибки IE7

Браузерный движок IE7 под названием Trident претерпел существенные изменения по сравнению с предыдущей версией Internet Explorer — исправлено множество ошибок, улучшилась поддержка свойств и селекторов CSS 2.1. Но в его основе осталось то же внутреннее свойство hasLayout и все ошибки с ним связанные. К старым ошибкам добавились ещё и новые.

Стили не работающие в IE7

Когда IE7 только вышел, казалось, что без многих стилевых свойств, псевдоклассов и псевдоэлементов легко можно обойтись. Но к хорошему, как говорится, быстро привыкаешь, и теперь без многих свойств прямо как без рук.

Псевдоэлементы :before и :after

Используются для вывода определённого контента до и после элемента. В примере 5.17 показано создание пунктирной горизонтальной линии с текстом «Линия отреза».

Пример 5.17. Линия отреза

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0  Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Линия отреза</title>
  <style type="text/css">
   HR {
    text-align: center; /* Выравниваем текст по центру */
    border: none; /* Убираем исходную границу */
    border-top: 1px dashed #000; /* Параметры линии  */
    height: 18px; /* Высота блока */
    background: url(images/scissors.png) no-repeat 10px -18px;
                /* Параметры фона */
   }
   HR:after {
    content: "Линия отреза";
    font-family: Arial, sans-serif; /* Рубленый шрифт для надписи */
    font-size: 12px; /* Размер шрифта */
    vertical-align: top; /* Выравнивание по верхнему краю */
   }
  </style>
 </head>
 <body>
  <p>Текст до</p>
  <hr />
  <p>Текст после</p>
 </body>
</html>
```

Вид страницы в IE7 показан на рис. 5.19. Кроме того, что не выводится текст под линией, у тега <hr> отображается рамка, несмотря на явный запрет на неё.

Рис. 5.19. Вид линии в IE7

Свойство content

Позволяет вставлять генерируемое содержание в текст веб-страницы, которое первоначально в тексте отсутствует. Применяется совместно с псевдоэлементами :after и :before.

Свойство caption-side

Определяет положение заголовка таблицы, который задаётся с помощью тега **<caption>**, относительно самой таблицы. Свойство caption-side выводит заголовок до или после таблицы (пример 5.18).

Пример 5.18. Заголовок таблицы

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>caption-side</title>
  <style type="text/css">
   table {
    width: 350px; /* Ширина таблицы */
    border-collapse: collapse; /* Убираем двойную рамку между ячейками */
   }
   td {
    border: 1px solid green; /* Параметры границы */
    padding: 5px; /* Поля в ячейке */
   }
   caption {
    caption-side: bottom; /* Заголовок под таблицей */
   }
  </style>
 </head>
 <body>
  <table>
   <caption>Расклад карт</caption>
   <tr>
    <td> </td><td>&spades;</td><td>&clubs;</td>
    <td>&hearts;</td><td>&diams;</td>
   </tr>
   <tr>
    <td>Чебурашка</td><td>6</td><td>3</td><td>1</td><td>3</td>
   </tr>
   <tr>
    <td>Крокодил Гена</td><td>1</td><td>5</td><td>5</td><td>2</td>
   </tr>
   <tr>
    <td>Шапокляк</td><td>3</td><td>4</td><td>6</td><td>0</td>
   </tr>
   <tr>
    <td>Крыса Лариса </td><td>3</td><td>1</td><td>1</td><td>8</td>
   </tr>
  </table>
 </body>
</html>
```

Свойства counter-increment и counter-reset

Стилевое свойство counter-increment предназначено для увеличения значения счетчика приращений, который задаётся свойством counter-reset. Такой счетчик подсчитывает количество отображений элементов на странице и может выводиться с помощью свойства content и псевдоэлементов :after и :before. Это позволяет создавать списки (в том числе многоуровневые), в которых нумерация и вид задаются через стили, а также делать автоматическую нумерацию заголовков, изображений, таблиц и др. (пример 5.19).

Пример 5.19. Нумерация заголовков

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>counter-increment</title>
  <style type="text/css">
   body {
    counter-reset: heading; /* Инициируем счетчик */
   }
   h2:before {
    counter-increment: heading; /* Указываем идентификатор счетчика */
    content: "Глава " counter(heading) ". ";
              /* Выводим текст перед содержимым тега <h2> */
   }
  </style>
```

```
 </head>
 <body>
  <h2>Теория ловли льва в пустыне</h2>
  <h2>Методы инверсной кинематики</h2>
  <h2>Ловля льва численными методами</h2>
 </body>
</html>
```

Из-за того, что IE7 не понимает используемые в стиле свойства, заголовки будут выведены как обычно, без текста впереди.

Свойство empty-cells

Задаёт отображение границ и фона в ячейке, если она пустая. При одновременном добавлении к таблице свойства border-collapse со значением collapse, свойство empty-cells игнорируется. Это свойство было рассмотрено в Главе 4 (см. пример 4.10).

Псевдокласс :focus

Определяет стиль для элемента получающего фокус. Например, им может быть текстовое поле формы, в которое устанавливается курсор. В примере 5.20 вокруг текстового поля с фокусом добавляется оранжевая рамка.

Пример 5.20. Фокус в поле

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>:focus</title>
  <style type="text/css">
   input:focus { border: 1px solid orange; }
  </style>
 </head>
 <body>
  <form action="handler.php">
   <p><input type="text" /></p>
  </form>
 </body>
</html>
```

Свойство outline

Универсальное свойство, одновременно устанавливающее цвет, стиль и толщину внешней границы на всех четырёх сторонах элемента. В отличие от линии, задаваемой через border, свойство outline не влияет на положение блока и его ширину. В IE7 также не поддерживаются и производные свойства outline-color, outline-style и outline-width.

Свойство quotes

Устанавливает тип кавычек, который применяется в тексте документа (см. пример 1.58).

Ошибки

Главная неприятность IE7 это большое количество ошибок, связанных с некорректной интерпретацией стилевых свойств.

@import

Не поддерживает типы носителей при импорте стилевого файла. Более того, при добавлении типа носителя стилевой файл вообще не загружается (пример 5.21).

Пример 5.21. Ошибка с @import

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE  html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
```

```
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>@import</title>
  <style type="text/css">
   @import "style/main.css" screen;
  </style>
 </head>
 <body>
  <p>...</p>
 </body>
</html>
```

Содержимое файла main.css.

```
body {
 background: #666;
 color: #ff0;
}
```

Границы

Линия, у которой стиль задан как dotted, превращается в dashed для толщины 2px и более (пример 5.22).

Пример 5.22. Ошибка с dotted

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Границы</title>
  <style type="text/css">
   P {
    border: 1px dotted #000;
    border-top: 2px dotted #000;
   }
  </style>
 </head>
 <body>
   <p> </p>
 </body>
</html>
```

На рис. 5.20 линия сверху отличается по стилю от остальных линий, хотя меняется лишь толщина.

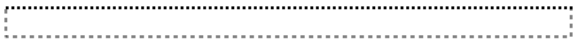


Рис. 5.20. Некорректная линия сверху

button

Любое значение в стиле `button[value="x"]` приводит к краху браузера!

```
button[value="1"] { color: red; }
```

display

В IE7 не поддерживаются следующие значения свойства display: inline-table, run-in, table, table-caption, table-cell, table-column, table-column-group, table-footer-group, table-header-group, table-row, table-row-group.

Для тега `<li>` значение block понимается как list-item, а значение inline-block работает только для строчных элементов.

overflow и position: fixed

Сочетание свойства overflow со значением auto или scroll с position: fixed приводит к серьёзной нагрузке на процессор и в итоге к нарушению работы браузера и операционной системы!

:first-letter и letter-spacing

Свойство letter-spacing, когда оно применяется к псевдоэлементу :first-letter, игнорируется.

Свойство line-height

Неправильно вычисляется высота строк для изображений и элементов форм.

list-style-image и float

Для списка не отображаются картинки-маркеры, если задано свойство list-style-image одновременно с float (пример 5.23).

Пример 5.23. Нет картинок

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE  html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>list-style-image</title>
  <style type="text/css">
   LI {
    list-style-image: url('images/book.gif'); /* Картинка */
    float: left; /* Выстраиваем список по горизонтали */
    padding-left: 10px; /* Поле слева */
    margin-right: 30px; /* Отступ справа */
   }
  </style>
 </head>
 <body>
  <ul>
   <li>Земля</li><li>Огонь</li><li>Вода</li><li>Воздух</li>
  </ul>
 </body>
</html>
```

Ошибка с выравниванием

Свойство text-align выравнивает не только текст, но и блоки. В примере 5.24 по правому краю выравнивается слой `t2` и текст внутри него.

Пример 5.24. Некорректное выравнивание

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE  html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>text-align</title>
  <style type="text/css">
   body { text-align: right; /* По правому краю */}
   .t1 {
    background: #fc0; padding: 10px;
   }
   .t2 {
    width: 60%; border: 1px solid #000; background: #fff;
    padding: 5px;
   }
  </style>
 </head>
 <body>
  <div class="t1">
   <div class="t2">Текст, выровненный по правому краю</div>
  </div>
 </body>
</html>
```

Вид текста в IE7 показан на рис. 5.21.

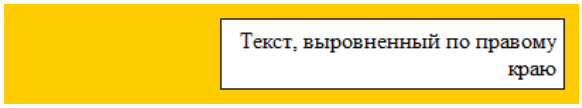


Рис. 5.21. Выравнивание по правому краю

Полезные ссылки

Подробный список найденных ошибок IE7 вы можете найти по следующим ссылкам.

- http://www.gtalbot.org/BrowserBugsSection/MSIE7Bugs — 188 ошибок
- http://www.quirksmode.org/bugreports/archives/explorer_7/ — 88 ошибок

Глава VI

Макеты, применяемые на сайтах

Когда продумывается дизайн сайта, учитывается объём будущей информации на нем, привычки потенциальных посетителей, удобство доступа к материалу и множество других вещей. Типовые элементы вроде заголовка, навигации, контента, контактной информации надо разместить на одной странице, в совокупности это и называется её макетом. Таким образом, макет это способ организации материалов на веб-странице. Самих макетов существует огромное количество, но их, тем не менее, можно систематизировать по ряду признаков, например, по ширине, по количеству колонок, по расположению элементов, по устройствам на которые они ориентированы и др. Наиболее популярным является деление макетов по ширине и количеству колонок, которые и рассмотрим далее.

Макеты по ширине

Различают пять типов макетов, связанных с шириной:

- фиксированные;
- резиновые;
- эластичные;
- адаптивные;
- комбинированные.

Фиксированный макет

Альтернативные названия: фикс (жарг.), fixed (англ.), фиксированный дизайн.

Макет обычно располагается по центру окна браузера, а его ширина ограничивается заданными размерами в пикселах (рис. 6.1).

Рис. 6.1. Фиксированный макет в браузере

Преимущества

- Из-за того, что ширина всех колонок известна, проще указывать размеры изображений, видео и других элементов страницы.
- Браузеры, как правило, лояльнее относятся к таким макетам, поэтому на вёрстку и отладку уходит меньше времени.

Недостатки

- Сайт смотрится плохо на мониторах с высоким разрешением, неэффективно используя свободное место. Некоторые владельцы таких мониторов даже уменьшают окно браузера по ширине, чтобы скрыть пустое место слева и справа от макета.

Сайты

- http://bash.org.ru
- http://www.youtube.com
- http://vkontakte.ru

Резиновый макет

Альтернативные названия: резина (жарг.), liquid (англ.), резиновый дизайн.

Ширина колонок задаётся в процентах или сочетаются проценты и пикселы таким образом, что макет занимает всю свободную ширину окна браузера. При изменении размеров окна или другом разрешении монитора макет подстраивается под них (рис. 6.2).

Рис. 6.2. Резиновый макет в браузере

Преимущества

- Используется вся эффективная область страницы.
- Веб-страницы удобно печатаются на бумаге любого формата.
- Веб-страницы хорошо смотрятся на разных устройствах от iPhone до ноутбука.

Недостатки

- На мониторах с высоким разрешением сайт плохо читается из-за чрезмерного удлинения строк текста. Здесь помогает ограничение ширины контента с помощью свойства max-width. Опять же некоторые владельцы больших мониторов уменьшают окно браузера до комфортных для них размеров.
- Резиновые макеты сложнее верстать и отлаживать в разных браузерах.

Сайты

- http://lenta.ru
- http://yandex.ru
- http://rutracker.org

Эластичный макет

Этот макет по своему виду может не отличаться от фиксированного или резинового макета. До тех пор, пока вы не измените размер шрифта в браузере, тогда вы заметите, что размер поменяли и элементы веб-страницы. Размер элементов задаётся не в пикселах и процентах, а в em, привязанному к размеру шрифта. Значение em можно использовать не для всех элементов, оставляя ширину некоторых фиксированной.

Преимущества

- Макет целиком или отдельные его части легко масштабировать, подгоняя под комфортный для восприятия размер.
- Макет будет одинаково смотреться на разных операционных системах, имеющих различия в выборе размера и типа шрифта.

Недостатки

- В современных браузерах функция масштаба страницы уже встроена, и пользоваться ей довольно удобно.
- Верстать эластичный макет крайне сложно, поскольку единица em имеет относительные размеры и зависит от используемого шрифта.
- В действительности сфера применения этого макета очень ограниченна.

Сайт

- http://www.csszengarden.com/?cssfile=http://green-beast.com/portfolio/zen/css/zen.css

Адаптивный макет

Этот макет подстраивается под разрешение монитора и окна браузера, меняя при необходимости ширину макета, число колонок, размеры изображений и текста. Для этого заготавливается несколько стилевых файлов под разный диапазон разрешений, выбор файла происходит через скрипты, которые и определяют нужную для этого информацию о пользователе.

Преимущества

- Этот тип макета наиболее удобен для пользователя, поскольку не зависит от разрешения и ширины окна браузера, приспосабливаясь под них.
- Макет комфортно можно смотреть на любом устройстве.

Недостатки

- Это самый сложный тип из всех макетов, ведь, по сути, вместо одного требуется сделать несколько макетов со своей графикой и CSS, а также написать скрипт для определения разрешения монитора или ширины окна браузера.
- За счёт универсальности макет сложно проверять на разные условия, которые возможны у пользователей.

Сайт

- Вид сайта http://www.w3.org показан на рис. 6.3. При уменьшении ширины окна до 500 пикселов и менее, дизайн сайта сменится (рис. 6.4).

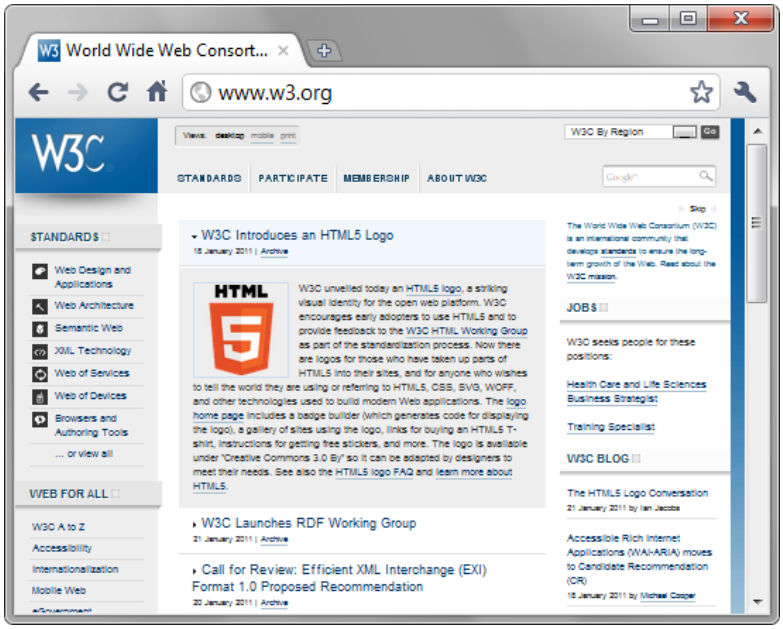


Рис. 6.3. Сайт W3C при обычной ширине

Рис. 6.4. Сайт W3C при узкой ширине

Комбинированный макет

Альтернативные названия: гибрид, hybrid (англ.).

Этот макет предполагает использование разной ширины для отдельных частей страницы, например, шапку и подвал делают резиновыми, а контент фиксированным (рис. 6.3).

Рис. 6.5. Комбинированный макет

Этот макет в действительности не является самостоятельным типом, поэтому наследует все плюсы и минусы фиксированного и резинового макета.

Надо отметить, что некоторые макеты, хотя и выглядят комбинированными, в действительности ими не являются. Резиновая шапка страницы может оказаться всего-навсего широким фоновым рисунком.

Сайт

- http://lionindesert.ru

Макеты по колонкам

Колонки в веб-дизайн пришли из полиграфии, где они используются в качестве способа разбивки широкого текста на более узкие фрагменты, а также для разделения различной информации. На сайтах текст всегда идёт одной колонкой, потому что универсальных способов для создания многоколоночного текста пока не существует. Кроме того, сайт по своей структуре и виду отличается от страницы в журнале, которую можно охватить одним взглядом, это тоже накладывает свои ограничения на распространение многоколоночного текста. Возможно в недалёком будущем сайты, специально «заточенные» под iPad и другие планшеты, станут активно применять колонки по своему прямому назначению. Пока же колонки преимущественно применяются для смыслового деления материала. Например, одна колонка содержит контент, другая навигацию, а третья рекламный баннер и т.д.

Наиболее распространенным вариантом является наличие на веб-странице двух колонок — одна из них, как правило, содержит навигацию, а во второй, более широкой колонке, размещается контент. Для резиновых макетов имеет смысл установить три колонки, чтобы эффективно использовать полезную площадь веб-страницы. В любом случае выбор числа колонок зависит исключительно от объема информации на сайте и способе её организации.

Фиксированный макет с одной колонкой

На сайтах одноколоночный макет применяется достаточно редко в силу его простоты, одним из примеров его использования служит bash.org.ru (рис. 6.6).

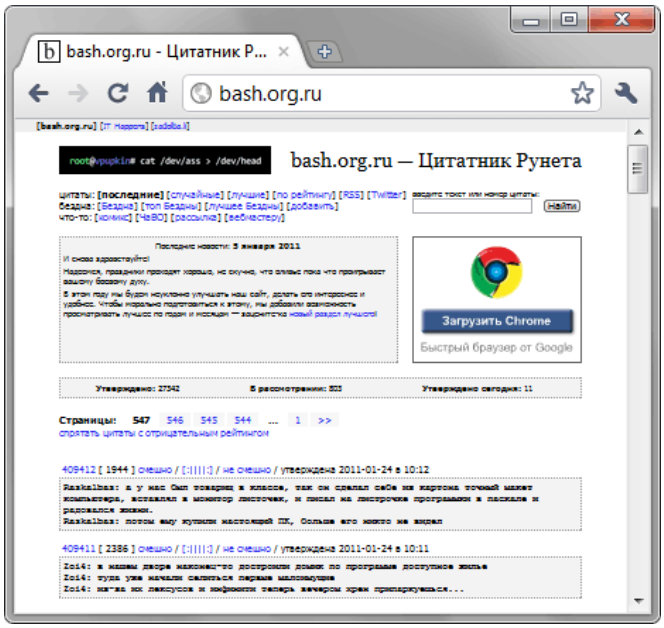


Рис. 6.6. Одноколоночный сайт

Этот макет удобен как основа для дальнейшего получения двух и трёхколоночного макета, поэтому на его примере рассмотрим некоторые принципы формирования макетов фиксированной ширины.

Выбор ширины обусловлен разными условиями, но в основном, разрешением монитора пользователей. По статистике LiveInternet (рис. 6.7) сайты в Рунете в основном посещают с разрешением 1280x1024 и 1024x768.

отчет: количество посетителей с разными разрешениями экрана — по дням | по неделям | по месяцам

значения: среднесуточные	январь 2011 г.		декабрь 2010 г.		в среднем за 3 месяца	
1280x1024	6,122,569	21.3%	6,635,300	22.5%	6,435,967	22.3%
1024x768	5,955,156	20.7%	6,297,655	21.4%	6,156,238	21.3%
1280x800	3,602,501	12.6%	3,737,850	12.7%	3,674,652	12.7%
1400x1050	3,443,975	12.0%	3,231,069	11.0%	3,173,607	11.0%
240x320	2,336,366	8.1%	2,305,287	7.8%	2,281,183	7.9%
1440x900	1,539,216	5.4%	1,614,524	5.5%	1,574,212	5.5%
1680x1050	1,184,711	4.1%	1,226,088	4.2%	1,203,059	4.2%
1152x864	1,129,444	3.9%	1,171,278	4.0%	1,145,556	4.0%
1920x1200	973,772	3.4%	929,825	3.2%	909,619	3.2%
1600x1200	839,157	2.9%	815,835	2.8%	796,691	2.8%
сумма выбранных	21,460,570	74.8%	22,207,163	75.3%	21,721,649	75.3%
всего	28,701,824		29,474,420		28,841,458	

Рис. 6.7. Статистика LiveInternet по разрешениям

Если ориентироваться на ширину 1024, то ширина макета не должна превышать это значение. Но видимая область страницы не равна ширине экрана, поскольку есть еще вертикальная полоса прокрутки. С её учётом и некоторого запаса получим максимальную ширину фиксированного макета 980–990 пикселов. Не обязательно ставить именно такое число, ничего не мешает установить, допустим, 800 пикселов по ширине. Это уже зависит от воли разработчики и дизайнерских изысков.

В качестве образца сделаем страницу с шириной 980px выровненную по центру. Для этого включим

класс `layout` и все элементы будем располагать в нём. Ширина для слоя `layout` указывается через width, а выравнивание по центру через margin.

```
.layout {
width:  980px; /* Ширина макета */
margin:  auto; /* Выравнивание по центру */
}
```

Свойство margin со значением auto выравнивает слой по центру только в комбинации со свойством width.

Также следует помнить, что при добавлении полей и границ к слою, они влияют на его ширину, увеличивая её. Так что нужно либо подкорректировать значение width с учётом этого, либо внутрь добавить ещё один слой, назовём его `wrap`, для которого и указывать padding.

```
.layout  {
 width: 980px; /* Ширина макета */
 margin: auto; /* Выравнивание по центру */
}
.wrap {
 padding: 20px;
}
```

В примере 6.1 показан HTML-код с необходимыми стилями.

Пример 6.1. Каркас для создания макета

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Одноколоночный макет</title>
  <style type="text/css">
   BODY { margin: 0; }
   .layout {
    width: 980px; margin: auto; background: #d3dfe9; }
   .wrap { padding: 20px; }
  </style>
 </head>
 <body>
  <div class="layout">
   <div class="wrap">
    Текст страницы
   </div>
  </div>
 </body>
</html>
```

Чтобы убрать отступ, установленный по умолчанию для веб-страницы, для селектора **BODY** добавлено свойство margin с нулевым значением.

Фиксированные макеты часто дополняют широким фоновым рисунком, который визуально расширяет страницу и задаёт её дизайн и стиль (рис. 6.8).

Рис. 6.8. Страница с фоновым рисунком

Фон не будет накладываться на контент, если фоновую картинку правильно подготовить. В середине изображения оставляют пустую полосу шириной, совпадающей с шириной макета (рис. 6.9). Фон не оказывает влияние на ширину страницы, поэтому может быть достаточно широким, чтобы охватить множество разрешений мониторов.

Рис. 6.9. Фон страницы

Сам фон ставится через свойство background и выравнивается по центру.

```
BODY {
 background: url(bg.png) no-repeat center top;
}
```

Вместо значения center top допускается также использовать 50% 0.

Фоновые рисунки применяются в вёрстке достаточно энергично, с их помощью можно создать иллюзию сложного дизайна простыми средствами. В качестве примера рассмотрим создание одноколоночного макета с тремя фоновыми изображениями (рис. 6.10).

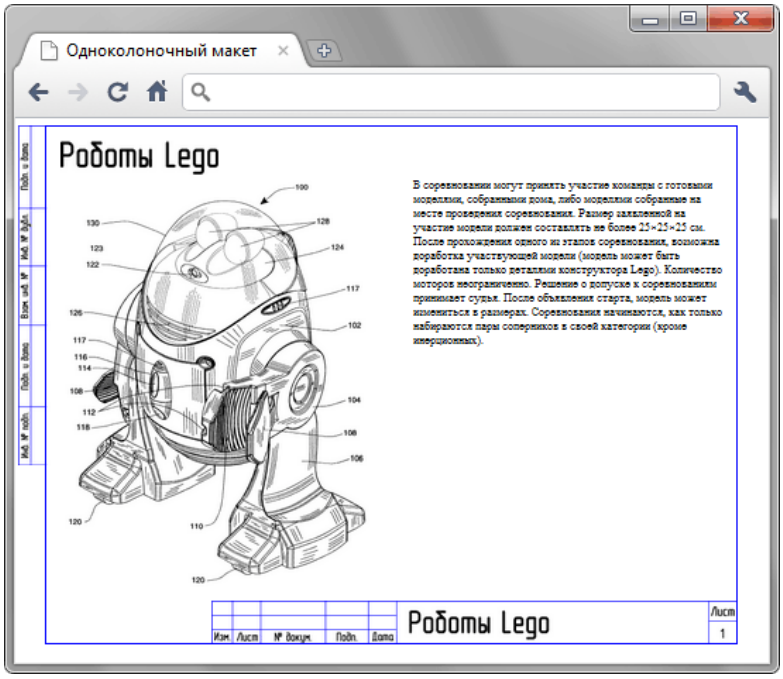


Рис. 6.10. Одноколоночный макет

Здесь фоном установлена чертёжная сетка слева, сетка снизу и рисунок робота. Чтобы сетка слева выходила за пределы макета, сделаем с ней то же самое, что и с фоном, представленным на рис. 6.9. Для этого придётся увеличить ширину пустого места слева на 980 пикселов плюс ширину самого рисунка, итого ширина фона получится 1060 пикселов.

```
BODY {
 background: url(images/frame-left.png) no-repeat 50% 10px; /* Сетка слева */
 margin: 10px 0; /* Отступы */
}
```

Ширину макета и параметры рамки ставим для слоя `layout`.

```
.layout {
 width: 978px; /* Ширина макета */
 background: url(images/frame-footer.png) 100% 100% no-repeat;
 border: 1px solid #00f; /* Параметры рамки */
 margin: auto; /* Выравниваем по центру */
}
```

Здесь ширина равна 978 пикселов, потому что к ней добавляется толщина границ слева и справа, что в итоге даст 980 пикселов. Нижняя чертёжная сетка добавляется в качестве фона и ставится в правый нижний угол макета.

Поля вокруг текста определяются через свойство padding, которое добавлено к слою `wrap`, он будет внутри слоя `layout`.

```
.wrap  padding: 20px  20px 70px; /* Поля */}
```

Рисунок робота устанавливается в качестве фона для слоя `content`. Так как текста на странице может быть не много, может получиться неприятная ситуация, когда наши фоновые рисунки обрезаются снизу. Чтобы этого не произошло, зададим минимальную высоту контента через свойство min-height.

```
.content {
 min-height: 590px; /* Минимальная высота */
 background: url(images/robot.png) no-repeat; /* Рисунок робота */
 padding-left: 500px; /* Поле слева */
}
```

Окончательный код показан в примере 6.2.

Пример 6.2. Одноколоночный макет

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Одноколоночный макет</title>
  <style type="text/css">
   BODY {
    background: url(images/frame-left.png) no-repeat 50% 10px;
                /* Сетка слева */
    margin: 10px 0; /* Отступы */
   }
   .layout {
    width: 978px; /* Ширина макета */
    margin: auto; /* Выравниваем по центру */
    border: 1px solid #00f; /* Параметры рамки */
    background: url(images/frame-footer.png) 100% 100% no-repeat;
   }
   .wrap { padding: 20px 20px 70px; }
   .content {
    min-height: 590px; /* Минимальная высота */
    background: url(images/robot.png) no-repeat; /* Рисунок робота */
    padding-left: 500px; /* Поле слева */
   }
  </style>
 </head>
 <body>
  <div class="layout">
   <div class="wrap">
    <div class="logo"><img src="images/title-robot.png"
                          alt="Роботы Lego" /></div>
    <div class="content">
     В соревновании могут принять участие команды с готовыми моделями,
     собранными дома, либо моделями собранные на месте проведения
     соревнования. Размер заявленной на участие модели должен составлять
     не более 25×25×25 см. После прохождения одного из этапов
     соревнования, возможна доработка участвующей модели (модель может
     быть доработана только деталями конструктора Lego). Количество
     моторов неограниченно. Решение о допуске к соревнованиям принимает
     судья. После объявления старта, модель может измениться в размерах.
     Соревнования начинаются, как только набираются пары соперников в
     своей категории (кроме инерционных).
    </div>
   </div>
  </div>
 </body>
</html>
```

Фиксированный двухколоночный макет

Двухколоночный макет наиболее популярен в веб-дизайне за счёт своей универсальности и простоты. В широкой колонке располагается контент, а в узкой, называемой ещё на жаргоне веб-разработчиков сайдбар (от англ. sidebar, боковая панель) навигация. Впрочем, ничего не мешает сделать колонки одинаковой ширины, если это продиктовано дизайном. На рис. 6.11 представлен макет, состоящий из двух колонок, в правой колонке расположена навигация, а в левой контент.

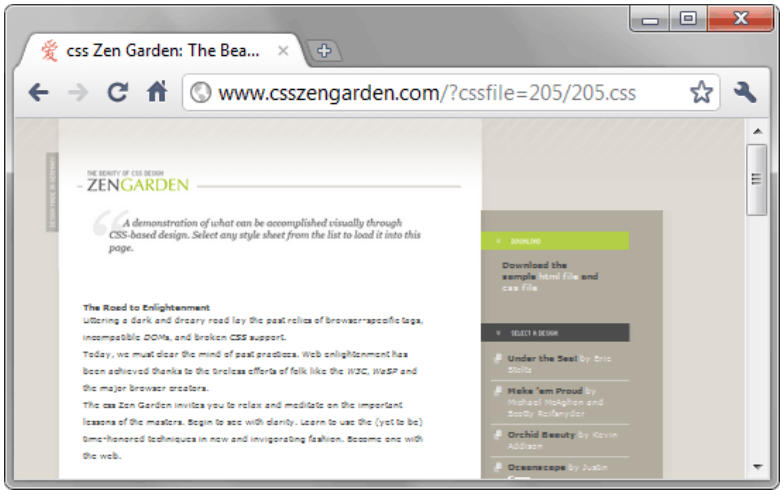


Рис. 6.11. Двухколоночный макет

Из-за того, что ширина всех колонок известна заранее, в силу того, что мы имеем дело с фиксированным макетом, существует несколько способов формирования колонок. Общий принцип у них такой. Слои с колонками располагаются в контейнере, назовём его `layout`, для которого устанавливается ширина макета и выравнивание по центру.

```
<div class="layout">
  <div class="sidebar">Колонка 1</div>
  <div class="content">Колонка 2</div>
</div>
```

Будем придерживаться соглашения, что слой `sidebar` формирует боковую панель с навигацией, а `content` основное содержание страницы (контент).

Навигация слева

Первый способ формирования колонок основан на позиционировании. Для слоя `layout` задаём относительное позиционирование, а для внутренних слоёв абсолютное. При таком сочетании значений можно устанавливать положение элементов относительно родителя с помощью свойств left и top (пример 6.3).

Пример 6.3. Использование позиционирования

```
.layout {
 width: 980px; margin: auto;
 position: relative; /* Относительное позиционирование */
}
.sidebar, .content { position: absolute; }
.sidebar {
 background: #C6DD7D; /* Цвет фона */
 width: 180px; /* Ширина колонки */
}
.content {
 background: #F4ECCE; /* Цвет фона */
 left: 180px; /* Сдвиг вправо */
 width: 800px; /* Ширина колонки */
}
```

По умолчанию у свойства left значение 0, так что для sidebar это свойство не установлено. У абсолютно позиционированных элементов ширина равна ширине контента, так что значение width требуется указывать для каждого слоя.

Приведённый способ имеет то преимущество, что для него не играет большой роли порядок слоёв в коде. Можно с лёгкостью поменять их местами, на отображении колонок это не скажется.

```
<div class="layout">
  <div class="content">Колонка 2</div>
  <div class="sidebar">Колонка 1</div>
</div>
```

Более простой стиль получается при использовании свойства float со значением left, которое задано для слоя sidebar. Но чтобы получилась не обтекаемая врезка, а подобие колонки, для слоя content также требуется установить свойство margin-left со значением равным ширине левой колонки или превышающей её (пример 6.4).

Пример 6.4. Использование float

```
.layout { width: 980px; margin: auto; }
.sidebar {
 background: #C6DD7D; /* Цвет фона */
 width: 180px; /* Ширина колонки */
 float: left; /* Обтекание справа */
}
.content {
 background: #F4ECCE; /* Цвет фона */
 margin-left: 180px; /* Отступ слева */
}
```

Как вариант, можно убрать свойство margin-left, а для формирования колонок добавить к layout свойство overflow с значением auto или hidden.

Навигация справа

Приведённые выше стили для создания навигации слева могут быть легко изменены и под навигацию справа. В действительности, разница между колонками только в их ширине и цвете фона. Изменив размеры, связанные с шириной и поменяв местами фон, получим результат, когда навигация уже расположена справа. В примере 6.5 показана модификация кода из примера 6.3.

Пример 6.5. Использование позиционирования

```
.layout {
 width: 980px; margin: auto; position: relative;
}
.sidebar, .content { position: absolute; }
.content {
 background: #F4ECCE; /* Цвет фона */
 width: 800px; /* Ширина колонки */
}
.sidebar {
 background: #C6DD7D; /* Цвет фона */
 left: 800px; /* Сдвиг вправо */
 width: 180px; /* Ширина колонки */
}
```

При позиционировании возможны проблемы с наложением подвала (при его наличии) на другие элементы. Этот вопрос рассмотрен в разделе, посвященном резиновым трёхколоночным макетам.

Аналогичное действие можно провести и с примером, в котором фигурирует свойство margin-left. Но в данном случае удобнее использовать свойство float со значением right. Также margin-left поменяется на margin-right (пример 6.6).

Пример 6.6. Использование float

```
.layout { width: 980px; margin: auto; }
.sidebar {
 background: #C6DD7D; /* Цвет фона */
 width: 180px; /* Ширина колонки */
 float: right; /* Обтекание слева */
}
.content {
 background: #F4ECCE; /* Цвет фона */
 margin-right: 180px; /* Отступ справа */
}
```

Стили для колонок не учитывают некоторых вещей, которые часто присутствуют на реальных сайтах. В частности, не указаны поля, поэтому текст вплотную прилегает к краю колонок. Добавление свойства padding при заданном width расширяет ширину элемента, поэтому необходимо уменьшить значение width на величину полей или добавить вложенный элемент с полями или отступами.

В примере 6.7 рассмотрен код, в котором навигация расположена в правой колонке, также присутствует подвал, и учитываются поля.

Пример 6.7. Двухколоночный макет

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Двухколоночный макет</title>
  <style type="text/css">
   BODY { background: #fff1e4; margin: 0; }
   .layout { width: 970px; margin: 0 auto 10px; background: #fff; }
   .content { margin-right: 350px; padding: 10px; }
   .sidebar { width: 320px; float: right; background: #e0ecb8; }
   .footer {
    clear: both; /* Отмена обтекания */
    border-top: 1px solid #ccc; padding: 10px;
   }
  </style>
 </head>
 <body>
  <div class="layout">
  <div class="sidebar">
   <ul>
    <li>Главная</li><li>Все коктейли</li><li>Коллекции</li>
    <li>Бокалы</li><li>Компоненты</li><li>Фичи</li>
   </ul>
  </div>
  <div class="content">
   <img src="images/title-sambarmen.png" alt="Приготовление коктейлей" />
   <h1>Яблочный эг-ног</h1>
   <p>Молоко — 40 мл, сок яблочный — 100 мл, сироп сахарный — 30 мм,
   один яичный желток.</p>
   <p>Приготовить напиток в миксере, подать в бокале хайбол
   со льдом и соломинкой.</p>
  </div>
  <div class="footer">&copy; Приготовление коктейлей</div>
 </div>
</body>
</html>
```

Результат данного примера показан на рис. 6.12.

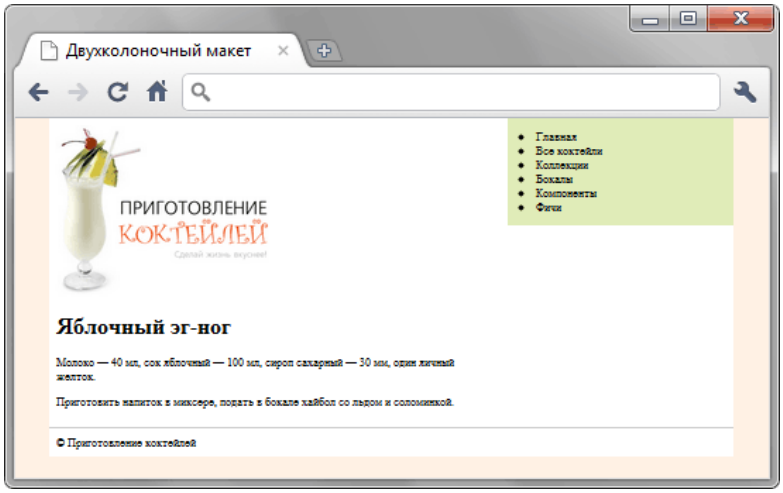


Рис. 6.12. Двухколоночный макет

Для слоя `content` поля включаются через свойство padding, но поскольку ширина слоя не указана, то поля никак на неё не повлияют. В слое `sidebar` вставлен список, у которого отступы установлены по умолчанию, поэтому никакого «налипания» текста на край фона нет. К подвалу (слой `footer`) добавляется свойство clear, которое отменяет действие float. Оно требуется на тот случай, когда высота навигации превышает высоту контента, чтобы текст не накладывался на подвал.

Фиксированный трёхколоночный макет

Трехколоночный макет часто используется в тех случаях, когда двух колонок уже недостаточно или требуется особым образом разделить материал. Например, одна колонка отдается под навигацию, вторая под контент, а в третьей публикуются новости сайта или другая информация (рис. 6.13). Три колонки обеспечивают большую свободу выбора размещения материалов вроде иллюстраций, рекламных баннеров, объявлений и др,

Рис. 6.13. Трёхколоночный макет

Методы формирования колонок похожи на методы для двухколоночного макета, это применение свойств float и position. Независимо от способа структура кода похожа. Контейнером выступает слой `layout`, для которого задаётся общая ширина макета и выравнивание, внутри располагаются слои создающие колонки.

```
<div class="layout">
  <div class="nav">Левая колонка</div>
  <div class="content">Центральная колонка</div>
  <div class="sidebar">Правая колонка</div>
</div>
```

Порядок слоёв в контейнере может меняться в зависимости от некоторых условий или вообще не играет роли, как в случае позиционирования.

Использование позиционирования

Для слоя `layout` устанавливается относительное позиционирование, а для слоёв `nav`, `content` и `sidebar` — абсолютное. При таком сочетании положение элементов меняется относительно родителя с помощью свойств left, right, top, bottom. Впрочем, из этого набора понадобится только left и right (пример 6.8).

Пример 6.8. Свойство position

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Трёхколоночный макет</title>
  <style type="text/css">
   .layout {
    width: 980px; /* Ширина макета */
    margin: auto; /* Выравнивание по центру */
    position: relative; /* Относительное позиционирование */
```

```
    }
    .nav, .content, .sidebar {
     position: absolute; /* Абсолютное позиционирование */
     color: #FFF; /* Цвет текста */
    }
    .nav {
     background: #7895A4; /* Цвет фона */
     width: 180px; /* Ширина */
    }
    .content {
     background: #F0EAD6; width: 600px; color: #000;
     left: 180px; /* Положение от левого края */
    }
    .sidebar {
     background: #C38A6D; width: 200px;
     right: 0; /* Положение от правого края */
    }
   </style>
  </head>
  <body>
   <div class="layout">
    <div class="nav">Левая колонка</div>
    <div class="content">Центральная колонка</div>
    <div class="sidebar">Правая колонка</div>
   </div>
  </body>
</html>
```

Из-за того, что координаты колонок устанавливаются относительно родителя, не имеет значение порядок слоёв в коде. Можно поменять слои местами, чтобы контент загружался первым, а навигация последней или наоборот.

Использование float

Свойство float в силу своей универсальности не раз выручало нас, вот и в этот раз с его помощью можно создать трёхколоночный макет, причём двумя разными методами. Первый основан на том, что float добавляется ко всем слоям с одновременным указанием ширины (пример 6.9).

Пример 6.9. float для всех колонок

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Трёхколоночный макет</title>
  <style type="text/css">
   .layout { width: 980px; margin: auto; }
   .nav, .content, .sidebar {
   float: left; /* Плавающий элемент */
  }
  .nav { width: 180px; background: #7895A4; color: #FFF; }
  .content { width: 600px; background: #F0EAD6; }
  .sidebar { width: 200px; background: #C38A6D; color: #FFE; }
 </style>
 </head>
  <body>
   <div class="layout">
    <div class="nav">Левая колонка</div>
    <div class="content">Центральная колонка</div>
    <div class="sidebar">Правая колонка</div>
  </div>
 </body>
</html>
```

Естественно, суммарная ширина всех колонок не должна превышать ширину макета. В коде слои идут слева направо, как показано в примере 6.8.

Второй метод использует сочетание свойств float и margin. Для левой колонки значение свойства float устанавливается left, а для правой — right, а также их ширина. Центральной колонке задаётся отступ слева и справа на соответствующую ширину левой и правой колонки.

```
.nav { /* Левая колонка */
 float: left; /* Обтекание справа */
 width: 180px; /* Ширина колонки */
```

```
}
.sidebar { /* Правая колонка */
 float: right; /* Обтекание слева */
 width: 200px; /* Ширина колонки */
}
.content { /* Центральная колонка */
 margin: 0 200px 0 180px; /* Отступы слева и справа */
}
```

При этом порядок слоёв также меняется, поскольку плавающие элементы должны идти первыми.

```
<div class="nav">Левая колонка</div>
<div class="sidebar">Правая колонка</div>
<div class="content">Центральная колонка</div>
```

Полный код приведён в примере 6.10.

Пример 6.10. Использование float и margin

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Трёхколоночный макет</title>
  <style type="text/css">
   .layout { width: 980px; margin: auto; }
   .nav, .content, .sidebar {
    color: #FFF; /* Цвет текста */
   }
  .nav { background: #7895A4; width: 180px; float: left; }
  .sidebar { background: #C38A6D; width: 200px; float: right; }
  .content {
   background: #F0EAD6; color: #000;
   margin: 0 200px 0 180px;
  }
  </style>
 </head>
 <body>
  <div class="layout">
   <div class="nav">Левая колонка</div>
   <div class="sidebar">Правая колонка</div>
   <div class="content">Центральная колонка</div>
  </div>
 </body>
</html>
```

Резиновый двухколоночный макет

Двухколоночный резиновый макет позволяет эффективно использовать площадь браузера, и вместе с тем достаточно удобен для самого широкого круга задач. Кроме того, такой макет не требует сложной работы над собой и его можно использовать на многих сайтах, комбинируя ширину колонок в пикселах и процентах. Примером такого макета служит сайт Хабрахабр (рис. 6.14), ширина колонок у него задана в процентах.

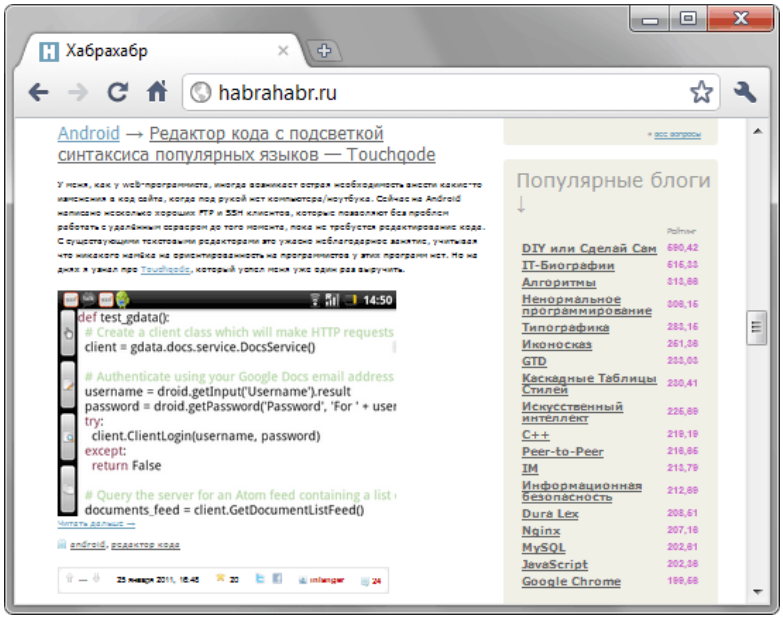


Рис. 6.14. Резиновый двухколоночный макет

Опять же, существует несколько подходов к формированию такого макета, но самый простой и удобный заключается в сочетании свойств float и margin.

Рассмотрим вариант, когда левая колонка имеет определенный размер, а ширина правой колонки устанавливается автоматически, исходя из ширины окна браузера. При этом ширина левой колонки может задаваться в пикселах или процентах. В табл. 6.1 приведены основные стилевые свойства для формирования двух колонок.

Табл. 6.1. Левая колонка заданной ширины

Для левого слоя шириной 20%	
Левая колонка	Правая колонка
`float: left` `width: 20%`	`margin-left: 21%`
Для левого слоя шириной 200px	
`float: left` `width: 200px`	`margin-left: 210px`

Для левой колонки требуется всего два свойства: float — заставляет вторую колонку располагаться рядом по горизонтали с первой и width, которое устанавливает ширину колонки. Вторая колонка будет занимать всё оставшееся место, поэтому для нее указывать width не нужно.

Правая колонка характеризуется лишь одним свойством — margin-left, оно смещает левый край колонки на ширину левого слоя, плюс задает отступ между колонками. Поэтому величина этого свойства в табл. 6.1 указана 21%, где 20% сама ширина колонки, а на один процент приходится расстояние между колонками. В случае задания ширины одной из колонок в пикселах, код останется прежним, но поменяются единицы измерения.

В примере 6.11 приводится код веб-страницы, на которой содержится: заголовок, две колонки и контактная информация. Ширина первой колонки с именем `sidebar` фиксирована и установлена как 120 пикселов, оставшееся пространство занято колонкой с именем `content`.

Из-за того, что мы имеем дело с резиновым макетом, который может занимать всю ширину окна браузера и уменьшаться до какого-то предела, имеет смысл сделать ограничения. Свойство min-width для слоя `layout` устанавливает минимальную ширину, если окно браузера будет меньше этого значения, появится горизонтальная полоса прокрутки. Свойство max-width, наоборот, задаёт максимальную ширину, которая не будет превышена.

Пример 6.11. Навигация слева

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Две колонки</title>
  <style type="text/css">
   .header, .sidebar, .content, .footer {
    padding: 10px; /* Поля */
    border: solid 1px #000; /* Параметры рамки */
    background: #e3e8cc; /* Цвет фона */
   }
   .header { /* Верхняя часть с заголовком */
    background: #e3e8cc; /* Цвет фона */
    font-size: 24px; /* Размер шрифта */
   }
   .layout {
    margin: 15px 0; /* Отступы сверху и снизу */
    overflow: hidden; /* Отменяем действие float */
    min-width: 800px; /* Минимальная ширина */
    max-width: 1200px; /* Максимальная ширина */
   }
   .sidebar { /* Навигация по сайту */
    width: 100px; /* Ширина меню */
    float: left; /* Состыковка с другим слоем по горизонтали */
   }
   .sidebar ul {
    list-style: none; /* Убираем маркеры */
    padding: 0; /* Убираем отступы */
   }
   .content { /* Основное содержание страницы */
    margin-left: 135px; /* Отступ слева */
   }
  </style>
 </head>
 <body>
  <div class="header">Заголовок сайта</div>
  <div class="layout">
   <div class="sidebar">
    <h2>Меню</h2>
    <ul>
     <li><a href="link1.html">Ссылка 1</a></li>
     <li><a href="link2.html">Ссылка 2</a></li>
     <li><a href="link3.html">Ссылка 3</a></li>
     <li><a href="link4.html">Ссылка 4</a></li>
    </ul>
   </div>
   <div class="content">
    <h1>Название страницы</h1>
    <p>Бла-бла.</p>
   </div>
  </div>
  <div class="footer">Подвал</div>
 </body>
</html>
```

Вид страницы при ширине окна меньше значения min-width продемонстрирован на рис. 6.15, та же самая страница, но в окне, превышающим значение max-width показана на рис. 6.16.

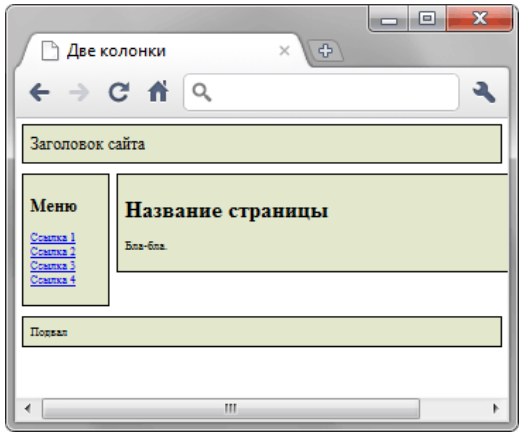


Рис. 6.15. Влияние свойства min-width

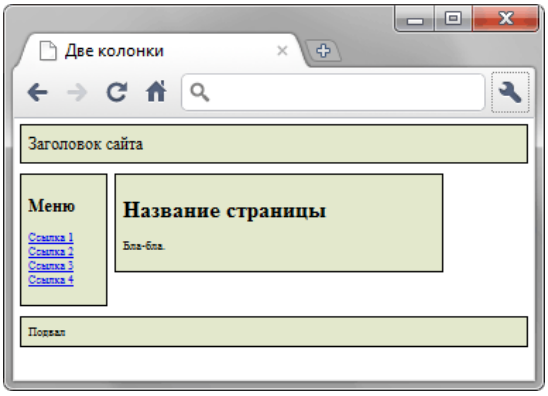


Рис. 6.16. Влияние свойства max-width

Для формирования колонки заданной ширины справа, а не слева, код незначительно модифицируется. В табл. 6.2 показаны стилевые параметры, которые требуются для этого случая.

Табл. 6.2. Правая колонка заданной ширины

Для левого слоя шириной 20%	
Левая колонка	Правая колонка
`float: right` `width: 20%`	`margin-right: 21%`
Для левого слоя шириной 200px	
`float: right` `width: 200px`	`margin-right: 210px`

Расположение слоев в коде остается прежним, но значение свойства float меняется на right, а отступ на margin-right. Более никаких изменений не предполагается, поэтому пример 6.11 останется прежним и в нем только следует заменить стиль слоев `sidebar` и `content` на тот, что показан в примере 6.12.

Пример 6.12. Стиль для добавления меню справа

```
.sidebar {
 width: 100px;
 float: right;
}
.content {
 margin-right: 135px;
}
```

Заметьте, что положение слоев в коде HTML не меняется, вначале идет слой с именем `sidebar`, а потом уже `content`. Причём это правило действует всегда, независимо от того, как будут отображаться на веб-странице колонки — меню слева от контента или наоборот, справа. В любом случае модифицируется только код CSS.

Резиновый трёхколоночный макет

Резиновый макет с тремя колонками, пожалуй, самый гибкий и настраиваемый из существующих макетов. Сочетание процентов и пикселов для указания ширины колонок позволяет делать разные макеты, выбирая их под свои задачи. На рис. 6.17 представлены варианты трёхколоночных макетов, для удобства они пронумерованы.

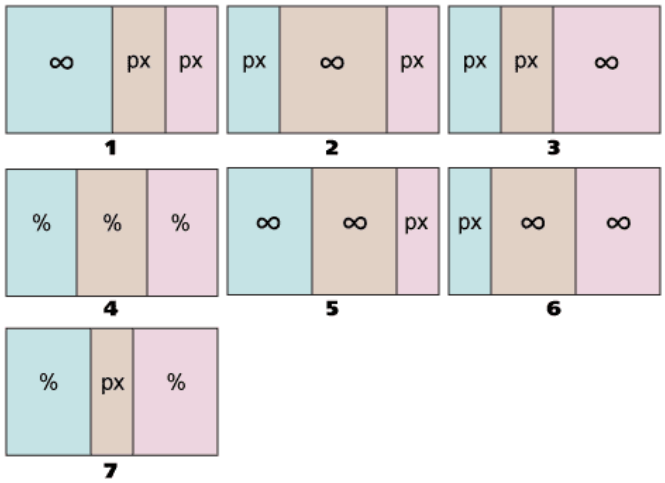


Рис. 6.17. Трёхколоночные макеты

Здесь символ процентов (%) означает, что ширина колонки задана в процентах от ширины макета, px — ширина колонки указана в пикселах, а знак бесконечности (∞), что колонка занимает оставшееся пространство. Несмотря на обилие разных макетов, принципы их построения остаются одинаковыми и включают два основных способа: позиционирование и плавающие элементы. Также можно воспользоваться таблицами для создания колонок одинаковой высоты.

Использование позиционирования

Для управления положением слоёв относительно родительского элемента необходимо для родителя установить свойство position со значением relative, а дочерним элементам, которые формируют колонки, значение absolute. Структура кода для первых четырёх макетов будет одинаковой и представлена в примере 6.13.

Пример 6.13. Две колонки в пикселах или три в процентах

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Три колонки</title>
 </head>
 <body>
  <div class="header">Шапка сайта</div>
  <div class="layout">
   <div class="col1">Колонка 1</div>
   <div class="col2">Колонка 2</div>
   <div class="col3">Колонка 3</div>
  </div>
</body>
</html>
```

Здесь колонки пронумерованы по порядку, т.е. колонка 1 располагается слева, колонка 2 по центру, а колонка 3 справа.

У позиционирования есть определённый недостаток, который ограничивает применение этого метода — при добавлении подвала он будет скрыт под колонками. Это связано с тем, что у слоя `layout`

нет высоты как таковой, поэтому он не «толкает» слой с подвалом вниз. Если подвал непременно требуется на странице, следует воспользоваться методом построения колонок, основанным на плавающих элементах. Ещё этот метод не работает в IE6 из-за наличия в нём ошибок.

Для макета № 1, в котором ширина первой колонки резиновая, а остальных фиксирована, стиль будет следующим (пример 6.14).

Пример 6.14. Макет № 1

```
.header { background: #D5BAE4; }
.layout { position: relative; /* Относительное позиционирование */ }
.layout DIV { position: absolute; /* Абсолютное позиционирование */ }
.col1 {
 background: #C7E3E4; /* Цвет фона */
 left: 0; /* Положение левого края */
 right: 300px; /* Положение правого края */
}
.col2 {
 background: #E0D2C7;
 width: 200px; /* Ширина колонки */
 right: 100px; /* Сдвигаем влево на ширину колонки 3 */
}
.col3 { background: #ECD5DE; width: 100px; right: 0; }
```

У колонок с заданной шириной стоит свойство width, а их положение слева или справа задаётся соответственно свойством left или right. Резиновая ширина оставшейся колонки строится после одновременного добавления left и right, значения которых совпадают с шириной фиксированных колонок.

Макеты № 2 (пример 6.15) и № 3 (пример 6.16) построены на том же принципе.

Пример 6.15. Макет № 2

```
.header { background: #D5BAE4; }
.layout { position: relative; }
.layout DIV { position: absolute; }
.col1 { background: #C7E3E4; width: 100px; }
.col2 { background: #E0D2C7; left: 100px; right: 200px;}
.col3 { background: #ECD5DE; width: 200px; right: 0; }
```

Пример 6.16. Макет № 3

```
.header { background: #D5BAE4; }
.layout { position: relative;}
.layout DIV { position: absolute; }
.col1 { background: #C7E3E4; width: 100px; }
.col2 { background: #E0D2C7; width: 200px; left: 100px; }
.col3 { background: #ECD5DE; left: 300px; right: 0; }
```

Макет № 4, в котором ширина всех колонок задана в процентах имеет некоторые нюансы. Если требуется одинаковая ширина всех колонок, её можно задать дробно (33.33%), но браузер Opera не умеет работать с дробными значениями процентов, поэтому в нём между колонками появятся «дыры» (рис. 6.18).

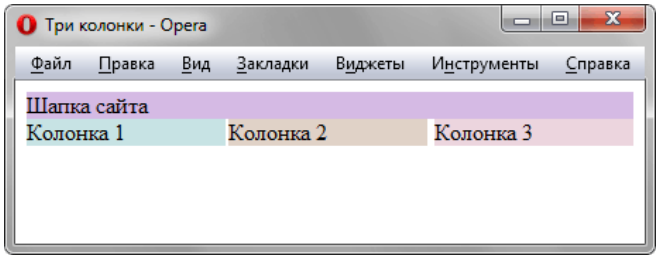


Рис. 6.18. Ширина колонок, заданная в дробных процентах

В подобных ситуациях следует перейти на неравные доли, к примеру, 33%, 34%, 33%, как показано в примере 6.17.

Пример 6.17. Макет № 4

```
.header { background: #D5BAE4; }
.layout { position: relative;}
.layout DIV { position: absolute; }
.col1 { background: #C7E3E4; width: 33%; }
.col2 { background: #E0D2C7; left: 33%; width: 34%; }
.col3 { background: #ECD5DE; right: 0; width: 33%; }
```

В некоторых браузерах возможно появление небольшого промежутка между колонками. Решается использованием не целых, а дробных значений процентов, т.е. 33.3% вместо 33%.

Оставшиеся макеты, в которых две колонки из трёх резиновые, представляют особую группу, потому что их можно трактовать по-разному. Так, ширина одной колонки указывается в пикселах, другой в процентах от ширины макета, а ширина оставшейся вычисляется автоматически. На рис. 6.19 показаны разные подходы к вычислению ширины колонок на примере макета № 5.

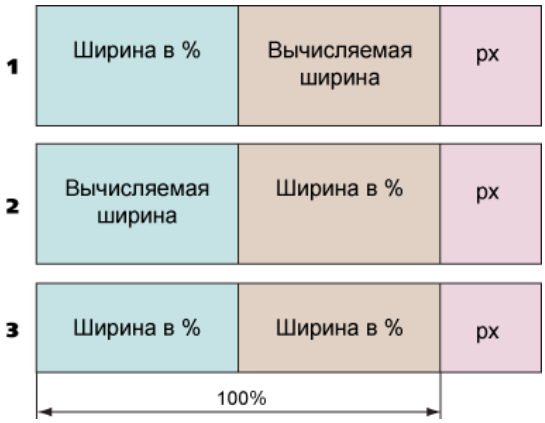


Рис. 6.19. Ширина двух резиновых колонок

В первом варианте ширина первой колонки задана в процентах от ширины макета, ширина второй колонки вычисляется автоматически, а третья колонка имеет фиксированную ширину в пикселах. Во втором варианте колонки меняются между собой, и автоматически вычисляется ширина первой колонки. Третий вариант предполагает, что общая ширина резиновых колонок принимается за 100% и ширина первой и второй колонки вычисляется от неё.

Первый и второй вариант легко реализуется аналогично коду с двумя колонками в пикселах. Только вместо ширины в px указываем %. В примере 6.18 приведён стиль макета № 5 с первой колонкой заданной в процентах.

Пример 6.18. Макет № 5. Ширина второй колонки вычисляемая

```
.header { background: #D5BAE4; }
.layout { position: relative;}
.layout DIV { position: absolute; }
.col1 { background: #C7E3E4; width: 50%; }
.col2 { background: #E0D2C7; left: 50%; right: 200px; }
.col3 { background: #ECD5DE; right: 0; width: 200px; }
```

Стиль макета № 5 с вычисляемой первой колонкой показан в примере 6.19.

Пример 6.19. Макет № 5. Ширина первой колонки вычисляемая

```
.header { background: #D5BAE4; }
.layout { position: relative;}
.layout DIV { position: absolute; }
.col1 { background: #C7E3E4; left: 0; right: 200px; margin-right: 50%; }
.col2 { background: #E0D2C7; width: 50%; right: 200px; }
.col3 { background: #ECD5DE; width: 200px; right: 0; }
```

Первой колонке нельзя задать ограничение справа через свойство right, поскольку значение будет равно 50%+200px, в CSS2 таких вычисляемых значений нет. Поэтому применим фокус — ограничим первый слой справа шириной 200px через right и сдвинем его влево на ширину второй колонки 50% с

помощью margin-right. Слой у нас абсолютно позиционированный, поэтому такие махинации никак на ширине не скажутся.

Третий вариант с двумя резиновыми колонками требует наличия дополнительного слоя, назовём его `rubber`, относительно которого будет задаваться ширина колонок (пример 6.20).

Пример 6.20. Макет № 5. Ширина двух колонок в процентах

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Три колонки</title>
  <style type="text/css">
   .header { background: #D5BAE4; }
   .layout { position: relative;}
   .layout DIV { position: absolute; }
   .rubber { left: 0; right: 200px; }
   .col1 { background: #C7E3E4; width: 60%; }
   .col2 { background: #E0D2C7; width: 40%; left: 60%; }
   .col3 { background: #ECD5DE; width: 200px; right: 0; }
  </style>
 </head>
 <body>
  <div class="header">Шапка сайта</div>
  <div class="layout">
   <div class="rubber">
    <div class="col1">Колонка 1</div>
    <div class="col2">Колонка 2</div>
   </div>
   <div class="col3">Колонка 3</div>
  </div>
 </body>
</html>
```

Построение макета № 6 аналогично макету № 5, поэтому останавливаться на нём не будем. Что касается макета № 7, то для него существует вариант, который может вызвать трудность. Это касается того случая, когда требуется сделать ширину левой и правой колонки вычисляемыми и равными между собой. Для этого ширину первой и третьей колонки следует сделать равной 50% (рис. 6.20).

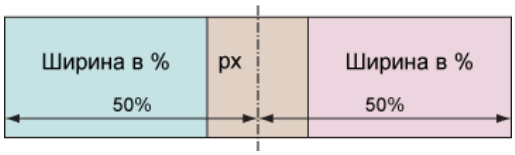


Рис. 6.20. Макет № 7 с равными по ширине колонками

Чтобы осталось место под вторую колонку, воспользуемся свойством margin-right для первой колонки и margin-left для третьей. Значением этих свойств будет половина ширины второй колонки. Так, если она равна 200px, то получится margin-right: 100px (пример 6.21).

Пример 6.21. Макет № 7. Ширина резиновых колонок одинакова

```
.header { background: #D5BAE4; }
.layout { position: relative;}
.layout DIV { position: absolute; }
.col1 { background: #C7E3E4; left: 0; right: 50%; margin-right: 100px; }
.col2 { background: #E0D2C7; width: 200px; left: 50%; margin-left: -100px; }
.col3 { background: #ECD5DE; left: 50%; right: 0; margin-left: 100px; }
```

Напрямую свойством width пользоваться нельзя, поскольку добавление margin только увеличит ширину, а не уменьшит, как нам требуется. Поэтому ширина формируется одновременно заданными свойствами left и right. С позиционированием второй колонки возникают сложности в указании значения left или right, поскольку оно будет равно 50%-200px. Так что устанавливаем положение левого края на 50% (left: 50%), а затем колонку целиком сдвигаем влево на половину её ширины через свойство margin-left (margin-left: -100px).

Поскольку ширина некоторых колонок вычисляется автоматически, рекомендуется сделать

ограничение по минимальной ширине макета, добавив свойство min-width для селектора **BODY**. Тогда ширина колонок не станет уменьшаться при достижении заданного предела.

```
BODY  {
 min-width: 800px; /* Минимальная ширина макета */
}
```

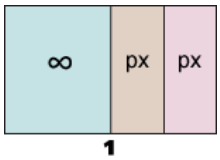
Значение min-width зависит от макета и содержимого страницы и обычно подбирается опытным путём.

Использование плавающих элементов

Плавающие элементы это ещё один универсальный метод построения разнообразных трёхколоночных макетов. В отличие от позиционирования он позволяют не заботиться о проблеме подвала, поскольку подвал всегда будет располагаться на своём месте под самой высокой колонкой.

Построение колонок происходит с помощью свойства float в сочетании со свойствами margin и width. В зависимости от выбранного макета меняется и порядок колонок в коде, вначале всегда следуют слои, к которым добавляется float.

Макет № 1. Резиновая первая колонка



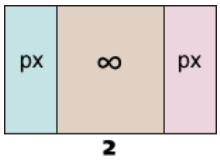
Ширина второй и третьей колонки указывается в пикселах, а их положение через свойство float со значением right. Для первой колонки также требуется задать свойство margin-right со значением равным суммарной ширине остальных колонок (пример 6.22).

Пример 6.22. Макет № 1

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Макет 1</title>
  <style type="text/css">
   .header, .footer { background: #D5BAE4; }
   .layout {
    overflow: hidden; /* Отменяем обтекание */
   }
   .col1 {
    background: #C7E3E4; /* Цвет фона */
    margin-right: 300px; /* Сдвиг влево на ширину колонок 2 и 3 */
   }
   .col2 {
    background: #E0D2C7; width: 200px;
    float: right; /* Обтекание слева */
   }
   .col3 {
    background: #ECD5DE; width: 100px;
    float: right; /* Обтекание слева */
   }
  </style>
 </head>
 <body>
  <div class="header">Шапка сайта</div>
  <div class="layout">
   <div class="col3">Колонка 3</div>
   <div class="col2">Колонка 2</div>
   <div class="col1">Колонка 1</div>
  </div>
  <div class="footer">Подвал</div>
 </body>
</html>
```

Макет № 2. Резиновая средняя колонка



2

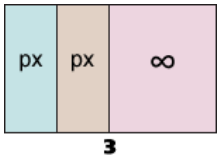
Ширина первой и второй колонки задана в пикселах, а их положение через свойство float со значением left для первой колонки и right для третьей. Средняя колонка, чтобы она сохраняла свой вид, содержит универсальное свойство margin, задающее отступ слева и справа (пример 6.23).

Пример 6.23. Макет № 2

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Макет 2</title>
  <style type="text/css">
   .header, .footer { background: #D5BAE4; }
   .layout { overflow: hidden; }
   .col1 { background: #C7E3E4; float: left; width: 200px; }
   .col2 { background: #E0D2C7;
    margin: 0 100px 0 200px; /* Отступ справа и слева */
   }
   .col3 { background: #ECD5DE; width: 100px; float: right; }
  </style>
 </head>
 <body>
  <div class="header">Шапка сайта</div>
  <div class="layout">
   <div class="col1">Колонка 1</div>
   <div class="col3">Колонка 3</div>
   <div class="col2">Колонка 2</div>
  </div>
  <div class="footer">Подвал</div>
 </body>
</html>
```

Макет № 3. Резиновая третья колонка



3

Положение первой и второй колонки указывается через свойство float со значением left, для третьей колонки нужно установить отступ слева (margin-left) на суммарную ширину остальных колонок (пример 6.24).

Пример 6.24. Макет № 3

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Макет 3</title>
  <style type="text/css">
   .header, .footer { background: #D5BAE4; }
   .layout { overflow: hidden; }
   .col1 { background: #C7E3E4; float: left; width: 100px; }
   .col2 { background: #E0D2C7; float: left; width: 200px; }
   .col3 { background: #ECD5DE;
    margin-left: 300px; /* Отступ слева на ширину колонок 1 и 2 */
   }
  </style>
 </head>
 <body>
  <div class="header">Шапка сайта</div>
  <div class="layout">
   <div class="col1">Колонка 1</div>
   <div class="col2">Колонка 2</div>
```

```
   <div class="col3">Колонка 3</div>
  </div>
  <div class="footer">Подвал</div>
 </body>
</html>
```

Макет № 4. Ширина всех колонок задана в процентах

Для построения этого макета подойдёт множество вариантов, основанных на предыдущих макетах, только вместо пикселов следует указать проценты. Ещё один способ показан в примере 6.25, где используется только свойство float и width.

Пример 6.25. Макет № 4

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

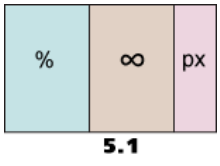
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Макет 4</title>
  <style type="text/css">
   .header, .footer { background: #D5BAE4; }
   .layout { overflow: hidden; }
   .layout DIV { float: left; }
   .col1 { background: #C7E3E4; width: 20%; }
   .col2 { background: #E0D2C7; width: 60%; }
   .col3 { background: #ECD5DE; width: 20%; }
  </style>
 </head>
 <body>
  <div class="header">Шапка сайта</div>
  <div class="layout">
   <div class="col1">Колонка 1</div>
   <div class="col2">Колонка 2</div>
   <div class="col3">Колонка 3</div>
  </div>
  <div class="footer">Подвал</div>
 </body>
</html>
```

Макет № 5. Ширина двух колонок резиновая

На рис. 6.19 было продемонстрировано, что макет № 5, у которого две резиновые колонки, можно трактовать по-разному.

1. Ширина первой колонки указана в процентах от ширины макета, третьей колонки в пикселах, а средняя колонка занимает оставшееся место.
2. Ширина второй колонки указана в процентах от ширины макета, третьей колонки в пикселах, а первая колонка занимает оставшееся место.
3. Суммарная ширина двух резиновых колонок принимается за 100%, и ширина колонок указывается в процентах от этой величины.

Ширина средней колонки вычисляемая



Здесь ширина первой колонки указывается в процентах, а её положение через свойство float со значением left, для третьей колонки ширина задана в пикселах, а значение свойства float как right. Для средней колонки остаётся только установить отступы слева и справа на ширину колонок (пример 6.26).

Пример 6.26. Макет 5.1

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Макет 5.1</title>
  <style type="text/css">
   .header, .footer { background: #D5BAE4; }
   .layout { overflow: hidden; }
   .col1 { background: #C7E3E4; width: 40%; float: left; }
   .col2 { background: #E0D2C7; margin: 0 200px 0 40%; }
   .col3 { background: #ECD5DE; width: 200px; float: right; }
  </style>
 </head>
 <body>
  <div class="header">Шапка сайта</div>
  <div class="layout">
   <div class="col1">Колонка 1</div>
   <div class="col3">Колонка 3</div>
   <div class="col2">Колонка 2</div>
  </div>
  <div class="footer">Подвал</div>
 </body>
</html>
```

Ширина первой колонки вычисляемая

5.2

Это наиболее хитрый макет, поскольку для первой колонки её ширина напрямую не указывается. Но чтобы ограничить контент необходимо для свойства margin-right указать значение совмещающее проценты и пикселы. В CSS2, как уже говорилось, подобное задать нельзя, поэтому добавим внутрь слоя `col1` ещё один слой с именем `wrap` и добавим отступ каждому из этих слоёв. Одному в процентах, другому в пикселах (пример 6.27).

Пример 6.27. Макет 5.2

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Макет 5.2</title>
  <style type="text/css">
   .header, .footer { background: #D5BAE4; }
   .layout { overflow: hidden; }
   .col1 { margin-right: 40%; }
   .col1 .wrap { margin-right: 200px; background: #C7E3E4; }
   .col2 { background: #E0D2C7; width: 40%; float: right; }
   .col3 { background: #ECD5DE; width: 200px; float: right; }
  </style>
 </head>
 <body>
  <div class="header">Шапка сайта</div>
  <div class="layout">
   <div class="col3">Колонка 3</div>
   <div class="col2">Колонка 2</div>
   <div class="col1">
    <div class="wrap">
     Колонка 1
    </div>
   </div>
  </div>
  <div class="footer">Подвал</div>
 </body>
</html>
```

Ширина двух колонок в процентах

5.3

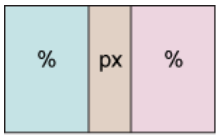
В этом макете общая ширина резиновых колонок принимается за 100%, поэтому нам потребуется дополнительный слой, относительно которого будет отсчитываться ширина внутренних колонок. Этот слой с именем `rubber` совместно со слоем `col3` работает как двухколоночный макет, а внутренние слои `col1` и `col2` выстраиваются по горизонтали за счёт применения свойства float (пример 6.28).

Пример 6.28. Макет 5.3

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Макет 5.3</title>
  <style type="text/css">
   .header, .footer { background: #D5BAE4; }
   .layout { overflow: hidden; }
   .rubber { margin-right: 200px; }
   .col1 { background: #C7E3E4; width: 60%; float: left; }
   .col2 { background: #E0D2C7; width: 40%; float: left; }
   .col3 { background: #ECD5DE; width: 200px; float: right; }
  </style>
 </head>
 <body>
  <div class="header">Шапка сайта</div>
  <div class="layout">
   <div class="col3">Колонка 3</div>
   <div class="rubber">
    <div class="col2">Колонка 2</div>
    <div class="col1">Колонка 1</div>
   </div>
  </div>
  <div class="footer">Подвал</div>
 </body>
</html>
```

Макеты 6, 7 и их разновидности строятся по тому же принципу, что и макет № 5, за исключением макета № 7, в котором ширина левой и правой колонки резиновая и равна между собой.



7

Вначале готовим основу, указываем порядок слоёв в HTML-коде.

```
<div class="layout">
 <div class="col1">Колонка 1</div>
 <div class="col3">Колонка 3</div>
 <div class="col2">Колонка 2</div>
</div>
```

Далее устанавливаем ширину всех колонок и их метод обтекания.

```
.col1 { width: 50%; float: left; }
.col2 { width: 200px; float: left; }
.col3 { width: 50%; float: right; }
```

Это ещё не всё, слои пока никаких колонок не формируют и выстраиваются совершенно не подходящим для нас способом. Требуется сместить вторую колонку влево на половину её ширины (margin-left: -100px) и заставить третью колонку встать на своё место. Для этого надо увеличить её ширину так, чтобы она была равна или больше, чем 50%+100px (половина второй колонки). Лучше всего

подойдёт свойство margin-left с отрицательным значением, после этого колонки будут созданы. Есть ещё некоторые нюансы. Крайние колонки состыкованы между собой, что хорошо заметно, когда их высота превышает высоту средней колонки (рис. 6.21).

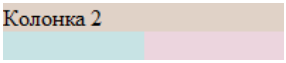


Рис. 6.21. Состыкованные колонки

К тому же в левой колонке текст справа, а в правой колонке текст слева скрывается под фоном центральной колонки. Это будет не заметно, если высота этой колонки большая, но это не всегда возможно. Чтобы преодолеть указанные недостатки, необходимо вложить внутрь крайних колонок ещё по одному слою (в примере он называется `wrap`) и уже для них задать цвет фона колонки, необходимые поля и отступы. Окончательный код приведён в примере 6.29.

Пример 6.29. Макет № 7

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Макет 7</title>
  <style type="text/css">
   .header, .footer { background: #D5BAE4; }
   .layout { overflow: hidden; }
   .col1 { width: 50%; float: left; }
   .col1 .wrap {
    background: #C7E3E4;
    margin-right: 100px; /* Сдвигаем влево на половину ширины колонки 2 */
    padding: 10px; /* Поля вокруг текста */
   }
   .col2 { background: #E0D2C7; width: 200px; float: left;
           margin-left: -100px; }
   .col3 { width: 50%; float: right; margin-left: -100px; }
   .col3 .wrap {
    background: #ECD5DE;
    margin-left: 100px; /* Сдвигаем вправо на половину ширины колонки 2 */
    padding: 10px;
   }
  </style>
 </head>
 <body>
  <div class="header">Шапка сайта</div>
  <div class="layout">
   <div class="col1"><div class="wrap">Колонка 1</div></div>
   <div class="col3"><div class="wrap">Колонка 3</div></div>
   <div class="col2">Колонка 2</div>
  </div>
  <div class="footer">Подвал</div>
 </body>
</html>
```

Поля и границы в трёхколоночном макете

В приведённые примеры макетов намеренно не добавлялось свойство padding, поскольку код служит лишь основой макета, и вы сами должны в конкретных случаях решать, нужны поля в колонках или нет. Добавление полей и границ к блочному элементу увеличивает его ширину, что надо учитывать при вёрстке. Однако ширина вырастает, только если для слоя установлено значение width. Рассмотрим несколько примеров, где ширина будет меняться, а где нет.

```
div { /* padding влияет на ширину */
 width: 200px;
 padding: 10px;
}
div { /* padding не влияет на ширину */
 position: absolute;
 left: 20px right: 20px;
 padding: 10px;
}
div { /* padding влияет на ширину */
 float: left;
 width: 50%;
 padding: 10px;
```

```
}
div { /* padding не влияет на ширину */
 margin-right: 50%;
 padding: 10px;
}
```

В тех колонках, где padding или border требуется, но их добавление приведёт к «ломке» макета, можно воспользоваться вложенным слоем и установить необходимые свойства для него.

```
<div style="width: 200px">
 <div style="padding: 10px; border: 1px solid #000">
  Колонка
 </div>
</div>
```

В этом примере ширина слоя заданная как 200 пикселов меняться не будет, но поля и рамка будут добавлены.

Таблица в качестве колонок

Таблицу удобно использовать для простого и быстрого создания колонок одинаковой высоты. Такая быстрота достигается несколькими вещами. Во-первых, структура кода для любого макета остаётся одинаковой, колонки в коде, как в случае с float, свой порядок никогда не меняют. Во-вторых, ширина ячеек вычисляется автоматически исходя из их содержимого, поэтому достаточно указать ширину нужных колонок, а оставшиеся подстроятся под общую ширину таблицы. В примере 6.30 показан макет № 1 сделанный на таблице.

Пример 6.30. Макет № 1

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Макет 1</title>
  <style type="text/css">
   .header, .footer { background: #D5BAE4; }
   .layout {
    width: 100%; /* Ширина таблицы */
   }
   .layout TD {
    padding: 5px; /* Поля в ячейках */
    vertical-align: top; /* Выравнивание по верхнему краю */
   }
   .col1 { background: #C7E3E4; }
   .col2 { background: #E0D2C7; width: 200px; }
   .col3 { background: #ECD5DE; width: 100px; }
  </style>
 </head>
 <body>
  <div class="header">Шапка сайта</div>
  <table class="layout" cellspacing="0">
   <tr>
    <td class="col1">Колонка 1</td>
    <td class="col2">Колонка 2</td>
    <td class="col3">Колонка 3</td>
   </tr>
  </table>
  <div class="footer">Подвал</div>
 </body>
</html>
```

Добавление свойства padding к селектору **TD** отменяет действие атрибута cellpadding тега **<table>** и добавляет поля к содержимому ячеек. При этом сама ширина колонок никак не изменится.

Для остальных макетов далее показан только стиль, HTML-код останется таким же, как в примере 6.30.

Макет № 2

```
.header, .footer { background: #D5BAE4; }
.layout { width: 100%;}
.layout TD { padding: 5px; vertical-align: top; }
```

```
.col1 { background: #C7E3E4; width: 100px; }
.col2 { background: #E0D2C7; }
.col3 { background: #ECD5DE; width: 200px; }
```

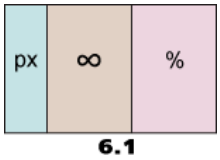
Макет № 3

```
.header, .footer { background: #D5BAE4; }
.layout { width: 100%;}
.layout TD { padding: 5px; vertical-align: top; }
.col1 { background: #C7E3E4; width: 100px; }
.col2 { background: #E0D2C7; width: 200px; }
.col3 { background: #ECD5DE; }
```

Макет № 4

```
.header, .footer { background: #D5BAE4; }
.layout { width: 100%;}
.layout TD { padding: 5px; vertical-align: top; }
.col1 { background: #C7E3E4; width: 33%; }
.col2 { background: #E0D2C7; width: 34%; }
.col3 { background: #ECD5DE; width: 33%; }
```

В макете 6.1 ширина первой колонки установлена в пикселах, третьей в процентах, а средняя колонка занимает оставшееся место.

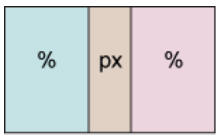


6.1

Макет № 6.1

```
.header, .footer { background: #D5BAE4; }
.layout { width: 100%;}
.layout TD { padding: 5px; vertical-align: top; }
.col1 { background: #C7E3E4; width: 200px; }
.col2 { background: #E0D2C7; }
.col3 { background: #ECD5DE; width: 40%; }
```

Остальные макеты с помощью таблицы строятся аналогично, меняется только значение свойства width. Для макета № 7, в котором ширина крайних колонок резиновая и равна между собой имеются некоторые хитрости, так что на нём остановимся подробнее.



7

Ширина для крайних колонок устанавливается равной 50%, а для средней колонки она задаётся в пикселах. Чтобы все размеры соблюдались к таблице необходимо добавить свойство table-layout со значением fixed.

```
.layout { width: 100%; table-layout: fixed; }
.col1 { width: 50%; }
.col2 { width: 200px; }
.col3 { width: 50%; }
```

Хотя суммарная ширина ячеек превышает ширину таблицы, никакого расширения и смещения таблицы не произойдет, как это случилось бы с блочными элементами. Как в таких случаях говорится, это не баг, это фича!

В примере 6.31 приведён полный код макета № 7.

Пример 6.31. Макет № 7

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Макет 7</title>
  <style type="text/css">
   BODY { margin: 0; background: #6D775B; }
   .layout { width: 100%; table-layout: fixed; }
   .layout TD { padding: 7px; vertical-align: top; }
   .col1 { background: #6D775B; width: 50%; }
   .col2 { background: #F5E8D0; width: 300px; }
   .col3 { background: #6D775B; width: 50%; text-align: right; }
  </style>
 </head>
 <body>
  <table class="layout" cellspacing="0">
   <tr>
    <td class="col1"><img src="images/tenniel1.png" alt="" /></td>
    <td class="col2">
     <img src="images/jabberwock.jpg" alt="Бармаглот" />
     <p>Он встал под дерево и ждет,<br />
     И вдруг граахнул гром.<br />
     Летит ужасный бармаглот, <br />
     И пылкает огнем. </p>
     <p>Раз-два, раз-два! Горит трава,
     Взы-взы — стрижает меч,<br />
     Ува! Ува! И голова <br />
     Барабардает с плеч.</p>
     <p>Льюис Кэрролл, перевод Дины Орловой, рисунки Джона Тенниела</p>
    </td>
    <td class="col3"><img src="images/tenniel2.png" alt="" /></td>
   </tr>
  </table>
 </body>
</html>
```

Результат примера показан на рис. 6.22.

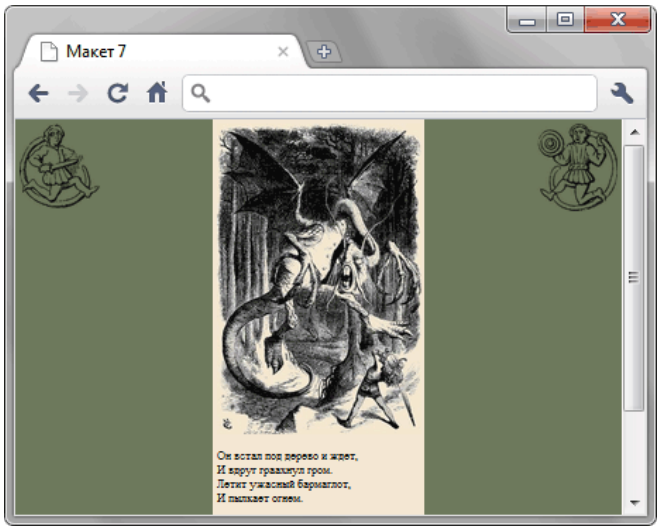


Рис. 6.22. Колонка по центру макета

Колонки одинаковой высоты

Колонки, получаемые с помощью ячеек таблицы, имеют одну высоту. Это неудивительно, поскольку ячейки взаимосвязаны и при повышении высоты одной ячейки, соответственно увеличивается высота рядом расположенных ячеек. Многие разработчики пытаются произвести подобный эффект с помощью слоёв, т.е. сделать их взаимосвязанными и одной высоты, независимо от объема содержимого. В действительности, при вёрстке слоями высота колонок устанавливается автоматически исходя из объема контента. Поэтому искусственные приёмы по созданию колонок одинаковой высоты противоречат самой идеологии слоёв.

Таким образом, видно чёткое разделение подходов к вёрстке:

- если используются слои, то колонки должны иметь высоту, которая определяется их содержимым;
- колонки одинаковой высоты строятся с помощью таблицы.

Игнорирование этих принципов приводит к усложнению кода и появлению ошибок в отображении документа браузерами, как следствие, повышается время на разработку сайта и его отладку. Исключением может служить применение свойства height, которое устанавливает высоту слоёв. В примере 6.32 показано создание подобного макета.

Пример 6.32. Использование height

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Колонки одинаковой высоты</title>
  <style type="text/css">
   .layout {
    overflow: hidden; /* Отмена обтекания */
   }
   .col1, .col2, .col3 {
    width: 33.33%; /* Ширина колонок */
    float: left; /* Создаем колонки */
   }
   .layout DIV DIV {
    margin: 0 10px; /* Отступы */
    padding: 10px; /* Поля */
    height: 150px; /* Высота колонок */
    background: #4F703E; /* Цвет фона */
    color: #F5E8D0; /* Цвет текста */
    overflow: auto;
   }
  </style>
 </head>
 <body>
  <div class="layout">
   <div class="col1">
    <div>
     <h2>SSI</h2>
     <p>Вставляет содержимое других файлов в код страницы.</p>
    </div>
   </div>
   <div class="col2">
    <div>
     <h2>Микроформаты</h2>
     <p>Формат и способ обмена данными между сайтами.</p>
    </div>
   </div>
   <div class="col3">
    <div>
     <h2>SVG</h2>
     <p>Язык разметки масштабируемой векторной графики для описания в
     формате XML.</p>
    </div>
   </div>
  </div>
 </body>
</html>
```

Результат данного примера показан на рис. 6.23.

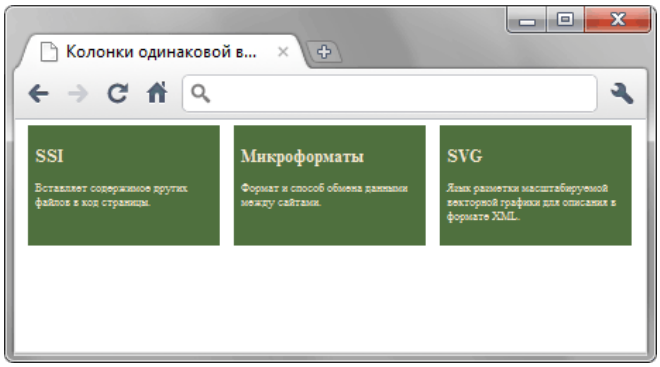


Рис. 6.23. Колонки одинаковой высоты

Понятно, что объём текста в колонках может различаться, поэтому высоту через height обычно указывают с запасом. Ещё можно добавить свойство overflow со значением auto. Если текст превысит заданную высоту, появится полоса прокрутки.

Также существуют способы, построенные на визуальном обмане. Колонки кажутся одинаковыми за счёт использования границ в качестве колонок, добавления фоновой картинки или цвета фона. В действительности же их высота, как им и положена, определяется контентом.

Границы в качестве колонок

Суть метода в следующем — устанавливаем слева или справа от элемента границу, ширина которой совпадает с шириной одной из колонок. Понятно, что границы предназначены совсем для других целей, но в данном случае такое их использование позволяет получить желаемый результат. Для этого создаем слой с именем `layout` и для него устанавливаем стиль, как показано ниже.

```
.layout {
 border-left: 200px solid #C7E3E4; /* Ширина границы и цвет левой колонки */
 background: #E0D2C7; /* Цвет фона правой колонки */
}
```

Поскольку слой у нас всего один, то колонки, имитированные с помощью широкой вертикальной линии и фона, всегда имеют одну высоту. Остается расположить информацию точно поверх границы. Для левой колонки, назовём её `col1`, следует задать ширину и с помощью стилевого свойства float указать, что это плавающий элемент. Поскольку граница не является частью текстового блока, то требуется сместить слой `col1` влево за счет добавления свойства margin-left с отрицательным значением, равным ширине границы.

```
.col1 {
 width: 200px; /* Ширина левой колонки */
 float: left; /* Превращаем в плавающий элемент */
 margin-left: -200px; /* Сдвигаем все влево на ширину границы */
}
```

В данном примере сочетание свойств float и margin-left позволяет расположить содержимое слоя прямо поверх границы. Для правой колонки `col2` никаких дополнительных условий указывать не надо, текст будет располагаться как нам требуется.

Из-за того, что используется плавающий элемент, может получиться, что текст в левой колонке выходит за пределы цветного прямоугольника. Чтобы этого не произошло, следует отменить действие свойства float за счет использования clear. В данном случае использовать overflow: hidden нельзя, будет обрезано пространство внутри границы, включая нашу файльшивую колонку. Окончательный код показан в примере 6.33.

Пример 6.33. Две колонки

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
```

```
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Две колонки одной высоты</title>
  <style type="text/css">
   .layout {
    border-left: 200px solid #C7E3E4;
                 /* Ширина границы и цвет левой колонки */
    background: #E0D2C7; /* Цвет фона правой колонки */
   }
   .col1 {
    width: 180px; /* Ширина левой колонки без полей */
    float: left; /* Превращаем в плавающий элемент */
    margin-left: -200px; /* Сдвигаем все влево на ширину границы */
   }
   .col1, .col2 { padding: 10px; }
   .clear { clear: both;  /* Отменяем обтекание */ }
  </style>
 </head>
 <body>
  <div class="layout">
   <div class="col1">Кумкват</div>
   <div class="col2">
    Небольшой экзотический фрукт оранжевого или оранжево-жёлтого цвета,
    по виду напоминающий мелкий апельсин.
   </div><div class="clear"></div>
  </div>
 </body>
</html>
```

Результат примера показан на рис. 6.24.

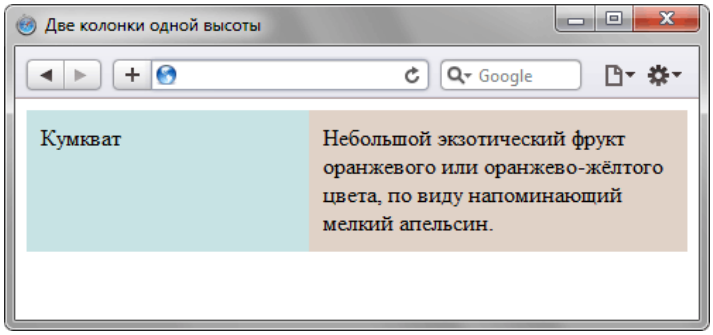


Рис. 6.24. Две колонки, созданные с помощью границы

Для создания трёхколоночного макета границу следует добавить слева и справа от контейнера `layout`. Для разнообразия ширина правой колонки указывается в em (в процентах границу задавать нельзя).

```
.layout {
 border-left: 200px solid #C7E3E4; /* Ширина границы и цвет левой колонки */
 border-right: 15em solid #ECD5DE; /* Ширина границы и цвет правой колонки */
 background: #E0D2C7; /* Цвет фона средней колонки */
}
```

Для первой колонки стилевой код остаётся неизменным, а третьей колонке следует указать свойство float со значением right и сдвинуть её вправо с помощью свойства margin-right (пример 6.34).

Пример 6.34. Три колонки

XHTML 1.0 | CSS 2.1 | IE 6 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Три колонки одной высоты</title>
  <style type="text/css">
   .layout {
    border-left: 200px solid #C7E3E4;
    border-right: 15em solid #ECD5DE; background: #E0D2C7;
   }
   .col1 { width: 180px; float: left; margin-left: -200px; }
   .col3 { width: 13em; float: right; margin-right: -15em; padding: 1em; }
   .col1, .col2 { padding: 10px; }
   .clear { clear: both; }
  </style>
 </head>
```

```
 <body>
  <div class="layout">
   <div class="col1">Грейпфрут</div>
   <div class="col3"><img src="images/grapefruit.jpg" alt="" /></div>
   <div class="col2">
    Грейпфрут — плод субтропического вечнозеленого дерева рода цитрусовых.
    Диаметр в среднем составляет 10–15 см, кожура желтая, мякоть красного
    оттенка. Вкус горький, чему способствует тонкая пленка вокруг каждой
    дольки. Если ее убрать, то горечь сильно снизится.
   </div><div class="clear"></div>
  </div>
 </body>
</html>
```

Результат примера показан на рис. 6.25.

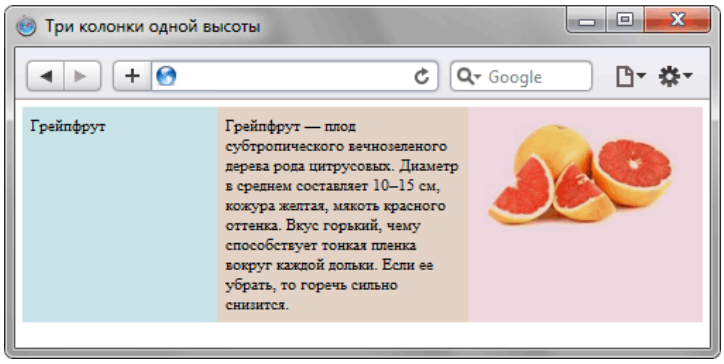


Рис. 6.25. Три колонки, созданные с помощью границы

Браузер IE6 содержит ошибку с границами и некорректно отображает данные примеры. Чтобы он правильно выводил макет, для него следует величину границ уменьшить вдвое.

```
<!--[if  IE 6]>
<style type="text/css">
 .col1 { margin-left: -100px; }
 .col3 { margin-right: -7.5em; }
</style>
<![endif]-->
```

Фоновая картинка

Принцип использования фонового изображения для колонок основан на том, что мы видим повторяющуюся по вертикали картинку, сверху которой располагается текст и другие элементы. Поскольку нам надо сделать эффект колонок, фон добавляется не к колонкам по отдельности, а для их родителя. Здесь имеет значение, какой это фон, что он собой представляет и каковы его размеры.

Предположим, что у нас двухколоночный фиксированный макет шириной 980 пикселов, левая колонка занимает ширину 200 пикселов. Создаём изображение размером 980x60 пикселов. Ширина соответствует ширине макета, а высоты обычно берётся 20–30 пикселов.

Многие разработчики делают рисунок высотой 1–2 пиксела, полагая, что объём файла будет минимальным, и загрузка произойдёт быстрее. Однако всё обстоит наоборот. Компьютер тратит в несколько раз больше времени для отображения одной страницы с узким фоном, что особенно заметно при прокрутке окна браузера. Так что при использовании фонового рисунка делайте изображение высотой не менее 20–30 пикселов — так отображение его на странице будет происходить быстрее.

Слишком скучно делать колонки однотонными, раз мы имеем дело с изображениями, поэтому добавим какие-нибудь ограничители по бокам колонок (рис. 6.26). Здесь самое главное, чтобы рисунок повторялся без стыков по вертикали.

Рис. 6.26. Фоновый рисунок

Теперь вставляем фон для слоя `layout`, внутри которого расположены наши колонки (пример 6.35).

Пример 6.35. Фон для фиксированного макета

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Двухколоночный фиксированный макет</title>
  <style type="text/css">
   .layout {
    width: 980px; /* Ширина макета */
    background: url(images/2col.png) repeat-y; /* Фоновый рисунок */
    overflow: hidden; /* Отменяем обтекание */
   }
   .sidebar { width: 180px; float: left; padding: 10px; }
   .content { margin-left: 210px; padding: 10px; }
  </style>
 </head>
 <body>
  <div class="layout">
   <div class="sidebar">
    <ul>
     <li>Главная</li><li>Все коктейли</li><li>Коллекции</li>
     <li>Бокалы</li><li>Компоненты</li>
    </ul>
   </div>
   <div class="content">
    <h1>Яблочный эг-ног</h1>
    <p>Молоко — 40 мл, сок яблочный — 100 мл, сироп сахарный — 30 мм,
    один яичный желток.</p>
    <p>Приготовить напиток в миксере, подать в бокале хайбол
    со льдом и соломинкой.</p>
   </div>
  </div>
 </body>
</html>
```

Результат данного примера показан на рис. 6.27. Хорошо заметно, что колонки имеют одинаковую высоту.

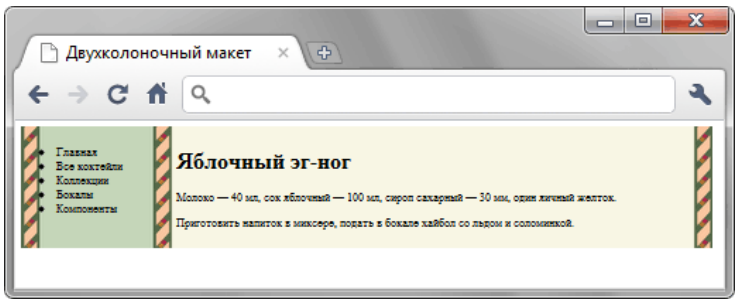


Рис. 6.27. Колонки одинаковой высоты, созданные фоновой картинкой

При использовании в рисунке каких-либо декоративных элементов, следует позаботиться, чтобы текст от них был отодвинут через padding и не «налипал».

Для трёх и более колонок в фиксированном макете процесс подготовки изображения сходный, а вот для резинового макета есть ряд особенностей из-за того, что ширина макета может лежать в широком диапазоне. Тогда фоновую картинку необходимо сделать заведомо широкой, например, 2000 пикселов. Фон не влияет на ширину веб-страницы и если не помещается в отведённые ему размеры, то обрезается. Этим качеством и воспользуемся. В качестве примера рассмотрим резиновый макет с фиксированной правой колонкой шириной 300 пикселов. Для него сделаем изображение шириной 2000x30 пикселов с тёмно-красным прямоугольником справа (рис. 6.27).

Рис. 6.27. Фон для резинового макета

Фиксированная колонка располагается справа, поэтому фон также надо установить в правый верхний

угол, указав 100% 0 для свойства background (пример 6.36).

Пример 6.36. Фон для резинового макета XHTML 1.0 CSS 2.1 IE 7 IE 8 IE 9 Cr 8 Op 11 Sa 5 Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Двухколоночный резиновый макет</title>
  <style type="text/css">
   .layout {
    background: url(images/bg-liquid.png) repeat-y 100% 0;
                /* Фоновый рисунок */
    overflow: hidden; /* Отменяем обтекание */
   }
   .sidebar { width: 280px; float: right; padding: 10px; color: #fff; }
   .content { margin-right: 300px; padding: 10px; }
  </style>
 </head>
 <body>
  <div class="layout">
   <div class="sidebar">Колонка 2</div>
   <div class="content">Колонка 1</div>
  </div>
 </body>
</html>
```

Напоследок перечислю плюсы и минусы использования фона в качестве искусственных колонок.

Плюсы

- Метод простой и не требует изменения существующего кода.
- В рисунок можно добавить разные декоративные элементы вроде градиентов, линий и др. Это позволит разнообразить дизайн колонок.

Минусы

- Наилучшие результаты получаются, когда одна из колонок или все имеют фиксированные размеры. Для макета, где ширина колонок резиновая, фон корректно добавить не получится.
- При отключении изображений в браузере текст может оказаться нечитабельным. Впрочем, это легко обойти, задав наряду с фоновой картинкой цвет фона.
- Файл с фоновой картинкой может иметь большой объём и долго загружаться. Отметим, что для простого изображения, вроде приведённого на рис. 6.27, это не так, при размерах 2000x30 пикселов файл занимает всего 275 байт.

Цвет фона

При имитации колонок одинаковой высоты с помощью фоновой картинки она добавляется через свойство background к родительскому элементу колонок (слой `layer`). Похожим образом работает и цвет фона. Его достаточно установить для слоя `layer`, а самой высокой колонке задать свой собственный цвет. В примере 6.37 приведён трёхколоночный макет, в котором цвет фона боковых колонок устанавливается через слой `layer`, а центральной колонки через слой `col2`.

Пример 6.37. Использования фонового цвета для колонок XHTML 1.0 CSS 2.1 IE 7 IE 8 IE 9 Cr 8 Op 11 Sa 5 Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Трёхколоночный резиновый макет</title>
  <style type="text/css">
   .layout {
    background: #B2D235; /* Цвет фона крайних колонок */
    overflow: hidden; /* Отменяем обтекание */
   }
   .layout DIV { float: left; }
   .col1 { width: 40%; }
   .col2 { background: #FFC60B; /* Цвет фона средней колонки */
    width: 20%; height: 200px; }
```

```
    .col3 { width: 40%; }
  </style>
 </head>
 <body>
  <div class="layout">
   <div class="col1">Колонка 1</div>
   <div class="col2">Колонка 2</div>
   <div class="col3">Колонка 3</div>
  </div>
 </body>
</html>
```

Свойство height в примере добавлено, чтобы стало заметно влияние цвета на колонках. В реальности оно не требуется.

У этого метода есть ряд недостатков, которые следует учитывать на практике.

- Сложно оценить предварительно, какая из колонок будет иметь наибольшую высоту.
- Для «раскрашивания» колонок применяется всего два цвета, так что нельзя задать трём и более колонкам разный цвет и чтобы они при этом имели одинаковую высоту.

Достоинства у метода тоже имеются, но их можно выразить всего парой слов: это простота и изящность решения. Поэтому этот метод можно довольно часто встретить в вёрстке, в частности, он применяется на сайте W3C.

Глава VII

Вёрстка типовых элементов веб-страницы

Меню

Под меню понимают набор ссылок, позволяющих переходить к разным разделам или документам сайта. Меню непосредственно связано с навигацией по сайту — системой организации документов и их взаимодействия между собой. Другими словами, навигация дает пользователю представление о структуре сайта и возможность перемещаться к нужной странице. Термин навигация давно уже стал широким понятием и включает в себя не только способ перехода от страницы к странице, но также вид и представление ссылок. По этой причине к навигации относят элементы страницы, которые имеют к ней косвенное отношение, например меню. Тем не менее это уже связанные понятия и, говоря о навигации по сайту, обычно упоминают и меню, с помощью которого пользователь загружает в окно браузера требуемые веб-страницы.

Правильно организованное меню предоставляет пользователю возможность быстрого доступа к нужным ему разделам, показывает, где он находится в данный момент в структуре сайта и что на нем еще можно посмотреть. С этой целью придерживаются следующих рекомендаций.

- Пункт меню, совпадающий с текущей веб-страницей, не делают ссылкой, чтобы не путать посетителя. В самом деле, если имеется ссылка, то появляется желание нажать на нее, в результате чего откроется тот же документ, который мы только что видели. После этого возникает мысль, что если произошла загрузка страницы, то это новый документ, но содержимое говорит о том, что его уже читали. В общем, получается противоречие, которого легко избежать, если просто заменить ссылку обычным текстом.
- Число пунктов меню обычно делают не очень большим. В противном случае имеет смысл разбить меню на подменю или организовать объем информации на сайте по-другому.
- Ничто не мешает сочетать на сайте различные виды навигации. Например, горизонтальное меню может применяться для доступа к основным разделам сайта, а вертикальное — для его подразделов.

Условно все типы меню можно отнести к следующим категориям.

- Вертикальное меню. Пункты меню располагаются друг под другом, и число их может быть достаточно велико. В силу своей универсальности вертикальное меню встречается на сайтах наиболее часто.
- Горизонтальное меню. В этом случае пункты помещаются по горизонтали, но чтобы это не привело к появлению горизонтальной полосы прокрутки, их число ограничивают или располагают в два-три ряда.
- Ниспадающее меню. Обычно выглядит как горизонтальное меню, но когда курсор мыши наводится на пункты, открывается дополнительное подменю.
- Всплывающее меню. При наведении на ссылку курсора мыши или щелчку по ней появляется меню в виде панели с набором вариантов перехода. Как только курсор уводится прочь со ссылки или с меню, оно пропадает.
- Контекстное меню. Открывается при нажатии в окне браузера правой кнопкой мыши. Подобный тип меню уже встроен в браузер по умолчанию, но его можно переназначить с помощью скриптов. Из-за того, что использование меню не является очевидным, оно применяется достаточно редко.

К разновидностям меню также можно отнести различные списки, в том числе раскрывающиеся, и вкладки.

Горизонтальное меню

Горизонтальное меню является одним из распространенных и популярных элементов навигации, используемых на сайтах. Как следует из названия, пункты меню располагаются по горизонтали, как правило, в верхней части страницы. Перечислим следующие особенности, присущие горизонтальному меню:

- ширина веб-страницы ограничена разрешением монитора, его размерами, настройками браузера и операционной системы. По этой причине большое количество пунктов меню делать не рекомендуется. Иначе это может привести к появлению горизонтальной полосы прокрутки или стать причиной изменения вида и форматирования меню;
- горизонтальное меню располагают в верхней части веб-страницы, чтобы его можно было видеть без прокрутки содержимого. Иногда горизонтальное меню для удобства пользователей дублируют внизу страницы.

Создание меню с помощью таблиц

Таблицы для создания горизонтального меню обладают некоторыми преимуществами по сравнению со слоями. В частности, ширина ячеек подстраивается под контент, для таблицы можно установить ширину в процентах или пикселах, легко настроить выравнивание внутри ячеек по вертикали и горизонтали. В примере 7.1 показано создание простого меню.

Пример 7.1. Использование таблицы для создания меню XHTML 1.0 CSS 2.1 IE 7 IE 8 IE 9 Cr 8 Op 11 Sa 5 Fx 3.6

```
<!DOCTYPE  html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Меню</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   TABLE.menu {
    background: #fc0; /* Цвет фона меню */
    width: 100%; /* Ширина меню */
    border: 1px solid black; /* Рамка вокруг таблицы */
    text-align: center; /* Выравнивание  текста по центру */
   }
   .menu TD {
    border: 1px solid white; /* Линия между ячейками */
    padding: 4px /* Поля вокруг текста */
   }
   .menu A { color: #BE1E2D; }
  </style>
 </head>
 <body>
  <table cellspacing="0" class="menu">
   <tr>
    <td><a href="link1.html">Чебурашка</a></td>
    <td><a href="link2.html">Крокодил Гена</a></td>
    <td><a href="link3.html">Шапокляк</a></td>
    <td><a href="link4.html">Крыса Лариса</a></td>
   </tr>
  </table>
 </body>
</html>
```

Результат данного примера показан на рис. 7.1.

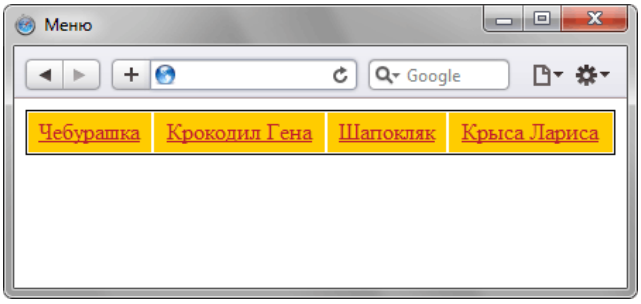


Рис. 7.1. Горизонтальное меню

При создании горизонтального меню не обойтись без подсвечивания фона ячейки таблицы при наведении на нее курсора мыши. Подобный эффект повышает привлекательность меню и удобство работы с ним (рис. 7.2).

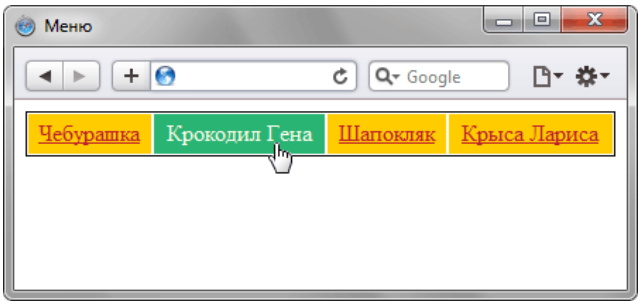


Рис. 7.2. Подсветка ячеек в таблице

Для создания подсветки текста воспользуемся псевдоклассом :hover, который управляет стилем ссылки при наведении на неё курсора мыши. Остаётся только указать желаемый цвет фона с помощью свойства background. Также можно изменить и другие параметры, например цвет текста. Чтобы в качестве ссылки выступала вся ячейка целиком, а не только текст в ней, следует добавить для селектора A свойство display со значением block. В данном случае оно превращает ссылку в блочный элемент, заставляя ее занимать все свободное пространство. На виде текста это никак не отражается, но зато вся ячейка будет ссылкой, что увеличивает её полезную площадь (пример 7.2).

Пример 7.2. Создание подсветки

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<style type="text/css">
 TABLE.menu {
  background: #fc0; /* Цвет фона меню */
  width: 100%; /* Ширина меню */
  border: 1px solid black; /* Рамка вокруг таблицы */
  text-align: center; /* Выравнивание текста по центру */
 }
 .menu TD {
  border: 1px solid white; /* Линия между ячейками */
  padding: 4px /* Поля вокруг текста */
 }
 .menu A {
  color: #BE1E2D; /* Цвет ссылок */
  display: block; /* Блочный элемент */
 }
 .menu TD:hover {
  background: #29B473; /* Цвет фона при наведении */
 }
 .menu TD:hover A {
  color: #FFE; /* Цвет ссылок при наведении */
  text-decoration: none; /* Убираем подчеркивание */
 }
</style>
```

Код HTML будет таким же, как и в примере 7.1.

Применение маркированного списка

В качестве альтернативы таблице для создания горизонтального меню используется маркированный

список, основанный на комбинации тегов <ul> и <li>. Хотя по умолчанию это скорее вертикальное, а не горизонтальное меню, с помощью стилей ему можно задать желаемый вид.

Начнём с совмещения пунктов меню по горизонтали. Тег <li> блочный, поэтому и начинается всегда с новой строки, поэтому превратим его в строчный элемент через свойство display со значением inline.

```
.menu LI {
  background: #fc0; /* Цвет фона меню */
  list-style: none; /* Убираем маркеры */
  padding: 4px 10px; /* Поля вокруг текста */
  display: inline; /* Строчный элемент */
}
```

На тег <ul> изначально действуют отступы, поэтому их необходимо убрать, чтобы они не влияли на вид отображения меню.

```
UL.menu {
  margin: 0; padding: 0; /* Отключаем отступы и поля */
}
```

Код для создания меню показан в примере 7.3.

Пример 7.3. Использование inline

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Меню</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   UL.menu {
    margin: 0; padding: 0; /* Отключаем отступы и поля */
   }
   .menu LI {
    background: #fc0; /* Цвет фона меню */
    list-style: none; /* Убираем маркеры */
    padding: 4px 10px; /* Поля вокруг текста */
    display: inline; /* Строчный элемент */
   }
   .menu A {
    color: #BE1E2D; /* Цвет ссылок */
   }
  </style>
 </head>
 <body>
  <ul class="menu">
   <li><a href="link1.html">Чебурашка</a></li>
   <li><a href="link2.html">Крокодил Гена</a></li>
   <li><a href="link3.html">Шапокляк</a></li>
   <li><a href="link4.html">Крыса Лариса</a></li>
  </ul>
 </body>
</html>
```

Для строчных элементов характерна одна особенность — перевод строки воспринимается в качестве пробела, поэтому между пунктами меню наблюдается небольшой промежуток (рис. 7.3).

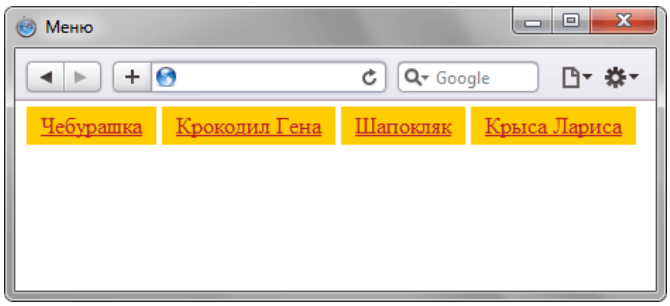


Рис. 7.3. Вид меню

Прежде чем решить, избавляться от этого промежутка или оставить его, необходимо решить более важные проблемы. В IE6–7 не действуют вертикальные поля, и нет самого промежутка. Получается, что

меню по своему виду различается от других браузеров (рис. 7.4).

Рис. 7.4. Вид меню в IE6–7

Как бороться с подобными явлениями нам уже известно — добавляем zoom и явно устанавливаем отступ справа. Ну, а чтобы стиль работал только для нужного браузера, используем условные комментарии.

```
<!--[if lte IE 7]>
 <style type="text/css">
  .menu LI {
   zoom: 1;
   margin-right: 4px;
  }
 </style>
<![endif]-->
```

Со строчными элементами есть ещё один нюанс — при переносе текста на другую строку также переносится фоновый цвет (рис. 7.5).

Рис. 7.5. Перенос текста в меню

Обойти это можно разными путями, например, установить для **<li>** запрет на перенос текста через white-space: nowrap или включив ограничение ширины всего меню с помощью min-width. Впрочем, ничего не мешает использовать оба метода сразу (пример 7.4).

Пример 7.4. Запрет переноса текста

```
UL {
 min-width: 400px; /* Ограничиваем ширину меню*/
}
LI {
 white-space: nowrap; /* Отменяем перенос текста */
}
```

Вернёмся к промежутку между строчными элементами. Он появляется от переноса строки, который считается за пробел. Так что если мы запишем всё в одну строку, это сразу же уберёт все пробелы.

```
<li><a href="link1.html">Чебурашка</a></li><li><a
     href="link2.html">Крокодил Гена</a></li>
```

Если пунктов меню много, то такую строку становится неудобно редактировать, но можно пойти на хитрость и убрать пробелы с помощью комментариев HTML.

```
<li><a href="link1.html">Чебурашка</a></li><!--
--><li><a href="link2.html">Крокодил Гена</a></li><!--
--><li><a href="link3.html">Шапокляк</a></li><!--
--><li><a href="link4.html">Крыса Лариса</a></li>
```

Также допустимо воспользоваться стилями и сдвинуть пункты меню за счёт отрицательного значения margin-right.

```
.menu LI {
  margin-right: -4px;
}
```

Значение может меняться в зависимости от размера шрифта и его настроек, поэтому данный способ нельзя считать универсальным.

Для строчных элементов нельзя задавать ширину и делать ссылки внутри них блочными. Так что альтернативой таблице с её возможностями могут быть только блочные элементы. Тег **<li>** изначально блочный, осталось только выстроить пункты меню по горизонтали, в чём нам поможет

свойство float.

```
.menu LI {
  float: left; /* Выстраиваем по горизонтали */
  width: 25%; /* Ширина пунктов */
  text-align: center; /* Выравниваем по центру */
}
```

Ширина всех пунктов будет равной и получается деление 100% на количество пунктов (4). Добавлять padding к пунктам нельзя, это превысит их ширину и разрушит всё меню, поэтому используем margin для вложенных селекторов A. Подсветка пунктов при наведении на них курсора аналогична примеру 7.2. Окончательный код меню приведён в примере 7.5.

Пример 7.5. Использование float

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Меню</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   UL.menu {
    margin: 0; padding: 0; /* Отключаем отступы и поля */
    overflow: hidden; /* Отменяем обтекание после меню */
    min-width: 450px; /* Ограничиваем ширину */
   }
   .menu LI {
    list-style: none; /* Убираем маркеры */
    float: left; /* Выстраиваем по горизонтали */
    width: 25%; /* Ширина пунктов */
    text-align: center; /* Выравниваем по центру */
    background: #fc0; /* Цвет фона меню */
   }
   .menu A {
    color: #BE1E2D; /* Цвет ссылок */
    display: block; /* Блочный элемент */
    margin: 5px; /* Отступы вокруг текста */
   }
   .menu LI:hover {
    background: #29B473; /* Цвет фона при наведении */
   }
   .menu LI:hover A {
    color: #FFE; /* Цвет ссылок при наведении */
    text-decoration: none; /* Убираем подчеркивание */
   }
  </style>
 </head>
 <body>
  <ul class="menu">
   <li><a href="link1.html">Чебурашка</a></li>
   <li><a href="link2.html">Крокодил Гена</a></li>
   <li><a href="link3.html">Шапокляк</a></li>
   <li><a href="link4.html">Крыса Лариса</a></li>
  </ul>
 </body>
</html>
```

Результат данного примера показан на рис. 7.6.

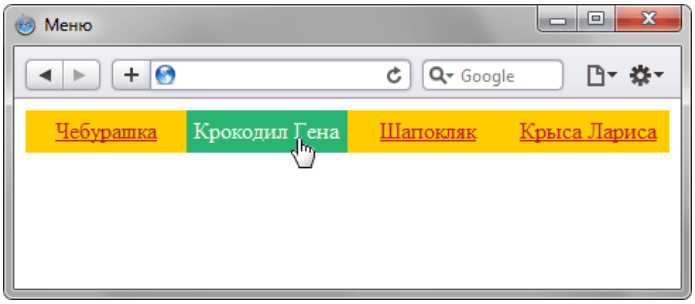


Рис. 7.6. Меню, созданное с помощью float

Как и в случае применения строчных элементов, перенос текста в данном случае приведёт к некрасивым эффектам (рис. 7.7). Способы борьбы показаны в примере 7.4.

Рис. 7.7. Перенос текста в меню

Вертикальное меню

Меню, в котором ссылки расположены друг под другом, называется вертикальным. К его достоинствам относят то, что ширину можно задавать как в пикселах, так и в процентах, подстраивая таким образом под любой макет. К тому же число пунктов вертикального меню может быть достаточно велико, при увеличении его высоты просмотр происходит посредством вертикальной полосы прокрутки веб-страницы. Меню также можно снабдить своей собственной полосой прокрутки для случая большого числа пунктов.

Простое меню с рамкой

Вначале создадим простое меню, состоящее из четырех пунктов. Ссылкой будет служить не только текст, но и пустое пространство справа от него (рис. 7.8). Для этого следует в стиле селектора **A** указать display: block.

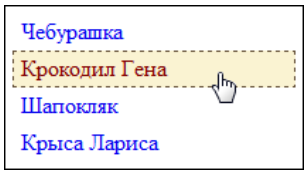


Рис. 7.8. Вид горизонтального меню

Ширина меню в пикселах или процентах устанавливается через свойство width, которое применяется к классу menu (пример 7.6). Здесь же задаются параметры рамки вокруг меню и отступы от неё до ссылок. Для того чтобы вокруг ссылки при наведении на нее курсора мыши появлялась цветная граница, требуется сделать следующее. Вначале к селектору **A** добавляем невидимую рамку, цвет которой совпадает с цветом веб-страницы, так, в примере используется белый цвет. Маскирование границы необходимо, поскольку текст ссылки иначе станет смещаться от своего исходного положения. Теперь для псевдокласса :hover остается только указать желаемые параметры рамки, цвет текста и фона.

Пример 7.6. Простое меню

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Меню</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   .menu {
    margin: 0; padding: 0; /* Отключаем отступы и поля */
    width: 200px; /* Ширина меню в пикселах */
    padding: 5px; /* Отступы от рамки до пунктов меню */
    border: 1px solid black /* Рамка вокруг меню */
   }
   .menu LI {
    list-style: none; /* Убираем маркеры */
   }
   .menu A {
    padding: 2px 5px; /* Отступ от рамки вокруг ссылки до текста */
    display: block; /* Ссылка как блочный элемент */
    border: 1px solid white; /* Маскируем рамку вокруг ссылки */
    text-decoration: none; /* Убираем подчеркивание у ссылок */
   }
   .menu A:hover {
    background: #faf3d2; /* Цвет фона под ссылкой */
    color: #800000; /* Новый цвет ссылки */
    border: 1px dashed #634f36; /* Рамка вокруг ссылки */
   }
  </style>
 </head>
 <body>
  <ul class="menu">
   <li><a href="link1.html">Чебурашка</a></li>
```

```
    <li><a href="link2.html">Крокодил Гена</a></li>
    <li><a href="link3.html">Шапокляк</a></li>
    <li><a href="link4.html">Крыса Лариса</a></li>
   </ul>
  </body>
</html>
```

Меню с подсветкой

Обычно выделение активного пункта меню, т.е. пункта, на котором стоит курсор мыши, происходит путем изменения цвета фона под текстом. Менее распространённым, в то же время наглядным и эффектным способом акцентирования внимания на элементе является появление цветного прямоугольника рядом с пунктом при наведении на него курсора мыши. На рис. 7.9 показано меню, пункты которого подсвечиваются подобным образом.

Рис. 7.9. Изменение оформления активного пункта меню

Легче всего добавить прямоугольник слева или справа от ссылки через свойство border-left или border-right. При этом толщина границы должна быть достаточно большой для того, чтобы из линии превратиться в прямоугольник. Поскольку требуется лишь менять ее цвет, то для **A:hover** проще установить свойство border-left-color, в качестве его значения выступает требуемый цвет линии слева от элемента. Аналогично цвет линии для случая, если она располагается справа, меняется через border-right-color. В примере 7.7 показано создание меню, у которого слева от каждого пункта добавляется зеленая линия толщиной 10 пикселов. Это достигается тем, что для селектора **A** используется свойство border-left. Чтобы ссылка была на всю ширину пункта, следует добавить display: block.

При наведении курсора мыши на пункт меню срабатывает псевдокласс :hover, который управляет оформлением ссылок в этом случае. Для него в данном примере цвет границы слева изменяется на оранжевый, а также становится другим цвет текста и фона.

Чтобы отделить один пункт от другого, между ними проводится линия серого цвета, путём добавления свойства border-bottom к селектору **A**.

Пример 7.7. Меню с подсветкой пунктов

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Меню</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   .menu {
    margin: 0;  padding: 0; /* Отключаем отступы и поля */
    width: 200px /* Ширина меню */
   }
   .menu LI {
    list-style: none; /* Убираем маркеры */
   }
   .menu A {
    display : block; /* Ссылка как блочный элемент */
    padding: 5px; /* Поля вокруг надписи */
    border-left: 10px solid #13694E; /* Линия слева */
    border-bottom: 1px solid #CCC; /* Линия между пунктами */
    background: #74A18E; /* Цвет фона */
    color: #FFF; /* Цвет текста */
    text-decoration: none; /* Убираем подчеркивание у ссылок */
   }
   .menu A:hover {
    border-left-color: #FA5; /* Меняем цвет линии слева */
    background: #A18E74; /* Новый цвет фона под ссылкой */
    color: #FFC; /* Новый цвет ссылки */
   }
```

```
  </style>
 </head>
 <body>
  <ul class="menu">
   <li><a href="link1.html">Чебурашка</a></li>
   <li><a href="link2.html">Крокодил Гена</a></li>
   <li><a href="link3.html">Шапокляк</a></li>
   <li><a href="link4.html">Крыса Лариса</a></li>
  </ul>
 </body>
</html>
```

Меню и подменю

Меню, которое содержит большое число пунктов, имеет смысл разделить на части, тем самым создавая так называемые подменю. Подобная организация меню хорошо воспринимается пользователем, к тому же она достаточно часто используется в различных программах, следовательно, будет привычной и на сайте.

Вариантов создания подменю может быть множество, например, на рис. 7.10 показан способ, когда подменю отображается справа от выделенного пункта.

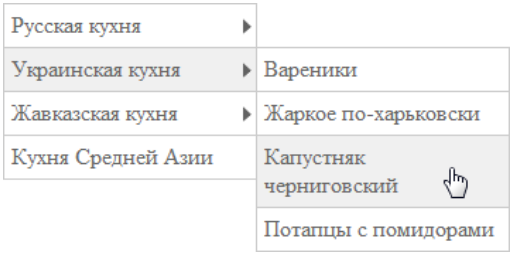


Рис. 7.10. Вид меню с выбранным подменю

Небольшая стрелка справа от текста нужна для наглядности, она показывает, что за пунктом скрывается подменю.

Из-за того, что требуется применять упорядоченность элементов и прямую зависимость их друг от друга, теги ** ** не очень годятся на роль создания подобного меню. Лучше использовать списки, а именно теги **<ul>** и **<li>**. Их внутренняя организация как нельзя лучше подходит для создания меню различного уровня вложенности. Начинается меню с тега **<ul>**, а затем каждый пункт меню верхнего уровня добавляется через тег **<li>**. Создание подменю происходит за счет конструкции **<ul>**, которая помещается внутрь **<li>** (пример 7.8).

Пример 7.8. Структура меню при использовании списков

```
<ul>
 <li>Пункт 1
  <ul>
    <li>Подпункт 1.1</li>
    <li>Подпункт 1.2</li>
  </ul>
 </li>
 <li>Пункт 2
  <ul>
   <li>Подпункт 2.1</li>
   <li>Подпункт 2.2</li>
  </ul>
 </li>
</ul>
```

Формирование подменю третьего уровня вложенности происходит аналогично, путём добавления тега **<ul>** внутрь подпункта **<li>**.

Стиль предназначен для оформления пунктов меню и сокрытия подменю. Поскольку тег **<ul>** отображает элементы как список, то вначале следует спрятать маркеры, установив для селектора **LI** свойство list-style: none. Также надо убрать различные отступы и поля, которые по умолчанию добавляются для элементов списка (пример 7.9).

Подменю вначале оказывается скрытым за счет использования свойства display со значением none, затем оно отображается при наведении на пункт меню (display: block). Для этого используется псевдокласс :hover, добавляемый к селектору **LI**.

Рамка вокруг пунктов добавляется к селектору **LI**. Чтобы при этом не возникало сдвоенных линий в

местах стыка пунктов, граница внизу убирается. Самые нижние пункты остаются в этом случае без линии, и для них нижняя граница устанавливается через селектор **UL**.

Чтобы в меню различать пункты, у которых нет подменю, и пункты с подменю, логично добавить к таким пунктам небольшой рисунок, например треугольник-стрелку, как на рис. 7.10. Изображение добавляется как фоновое с помощью свойства background, применяемого к селектору **A**. В качестве значений указывается путь к графическому файлу и положение картинки. Так 100% 50% означает, что рисунок будет одновременно выравниваться по правому краю и по центру вертикали пункта. Отменить показ рисунка для отдельных пунктов можно указав background-image: none в теге **<a>**, но лучше ввести отдельный класс, назовём его `nobullet` и будем использовать в нужных местах.

Пример 7.9. Меню с подменю

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Меню</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   .menu, .menu UL {
    width: 180px; /* Ширина меню */
    margin: 0; padding: 0; /* Отключаем отступы и поля */
    border-bottom: 1px solid #CCC; /* Линия снизу */
   }
   .menu LI {
    list-style: none; /* Убираем маркеры */
    position: relative; /* Подпункты позиционируются относительно */
    border: 1px solid #CCC; /* Рамка вокруг пунктов меню */
    border-bottom: none; /* Границу снизу не проводим */
   }
   .menu UL {
    position: absolute; /* Подменю позиционируются абсолютно */
    display: none; /* Скрываем подменю */
    top: -1px; /* Сдвигаем вверх на толщину линии */
    left: 178px; /* Сдвигаем подменю вправо */
   }
   .menu A {
    display: block; /* Ссылка как блочный элемент */
    padding: 5px; /* Поля вокруг надписи */
    text-decoration: none; /* Убираем подчеркивание у ссылок */
    background: url(images/bullet.png) 98% 50% no-repeat;
                /* Отображаем рисунок стрелки */
    color: #666; /* Цвет текста */
   }
   .menu UL A {
    background-image: none; /* Для подпунктов рисунок убираем */
   }
   .menu LI:hover {
    background-color: #f0f0f0; /* Цвет фона активного пункта */
   }
   .menu LI:hover UL {
    display: block; /* При выделении пункта отображается подменю */
   }
   .menu .nobullet {
    background-image: none; /* Прячем рисунок */
   }
  </style>
 </head>
 <body>
  <ul class="menu">
   <li><a href="russian.html">Русская кухня</a>
    <ul>
     <li><a href="linkrl.html">Бефстроганов</a></li>
     <li><a href="linkr2.html">Гусь с яблоками</a></li>
     <li><a href="linkr3.html">Крупеник новгородский</a></li>
     <li><a href="linkr4.html">Раки по-русски</a></li>
    </ul>
   </li>
   <li><a href="ukrainian.html">Украинская кухня</a>
    <ul>
     <li><a href="linkul.html">Вареники</a></li>
     <li><a href="linku2.html">Жаркое по-харьковски</a></li>
     <li><a href="linku3.html">Капустняк черниговский</a></li>
     <li><a href="linku4.html">Потапцы с помидорами</a></li>
    </ul>
   </li>
   <li><a href="caucasus.html">Кавказская кухня</a>
    <ul>
```

```
      <li><a href="linkc1.html">Суп-харчо</a></li>
      <li><a href="linkc2.html">Лилибдж</a></li>
      <li><a href="linkc3.html">Чихиртма</a></li>
      <li><a href="linkc4.html">Шашлык</a></li>
     </ul>
    </li>
    <li><a href="asia.html" class="nobullet">Кухня Средней Азии</a></li>
   </ul>
  </body>
</html>
```

Данный пример не работает в браузере IE6, поскольку он поддерживает псевдокласс :hover только для ссылок и соответственно, не понимает его для селектора **LI**.

Ниспадающее меню

Ниспадающее меню представляет собой разновидность горизонтального меню, но при наведении курсора мыши на пункт происходит открытие дополнительного списка, из которого выбираются подпункты (рис. 7.11). Ниспадающее меню давно уже применяется в различных операционных системах, поэтому принцип его действия понятен пользователю. Однако на сайтах подобное меню встречается не повсеместно, поэтому, глядя на список ссылок, сложно сразу сказать, какой перед нами тип меню — обычное или ниспадающее. Кроме того, к недочетам любого меню, которое первоначально скрывает часть данных, относится то, что выбор вариантов сразу не виден, и чтобы он стал доступен, требуется навести курсор на текст.

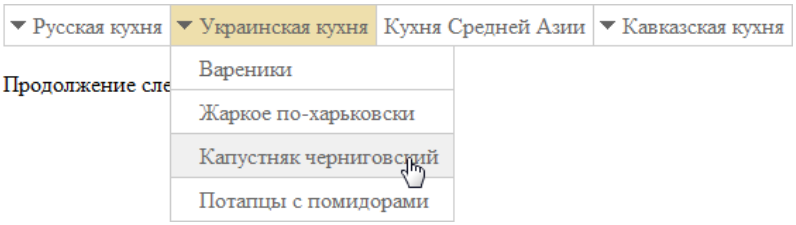


Рис. 7.11. Ниспадающее меню

Одним из вариантов создания ниспадающего меню является модификация примера 7.9, предназначенного для получения всплывающих подменю. В самом деле, если пункты расположить по горизонтали, то мы получим желаемый результат. С этой целью требуется для селектора **LI** добавить свойство float: left (пример 7.10).

Код HTML по сравнению с примером 7.9 практически не поменяется, только для разнообразия пункты поменялись местами, а в коде стиля будут небольшие изменения. В частности, чтобы выделить пункты меню, содержащие подменю, слева от текста добавляется небольшая стрелка вниз, как показано на рис. 7.11. С этой целью используется свойство background у селектора **A** со значением 4px 50%, оно помещает рисунок по левому краю со сдвигом на четыре пиксела и по центру вертикали. Чтобы картинка не накладывалась на текст, необходимо добавить отступ слева через свойство padding. При этом для пункта без подменю надо не забыть убрать рисунок и отступ.

Для пунктов подменю (селектор **UL LI**) обязательно следует задать их ширину через свойство width. При этом пункты становятся одной ширины и выстраиваются друг под другом. Иначе аккуратный строй ссылок может сломаться, а подменю рассыплется.

Пример 7.10. Ниспадающее меню

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Меню</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   .menu, .menu UL {
    margin: 0; padding: 0; /* Отключаем отступы и поля */
    background: #FFF; /* Цвет фона меню */
   }
   .menu > LI {
    float: left; /* Выстраиваем меню по горизонтали */
    height: 28px; /* Высота меню */
   }
   .menu > LI > A {
    line-height: 18px; /* Выравниваем текст посередине */
   }
   .menu LI {
    list-style: none; /* Убираем маркеры */
    position: relative; /* Подпункты позиционируются относительно */
    border: 1px solid #CCC; /* Рамка вокруг пунктов меню */
    border-right: none; /* Границу снизу не проводим */
   }
   .menu UL {
```

```
    border-right: 1px solid #CCC; /* Граница справа */
    border-bottom: 1px solid #CCC; /* Граница снизу */
    position: absolute; /* Подменю позиционируются абсолютно */
    display: none; /* Скрываем подменю */
    top: 28px; /* Сдвигаем вниз */
    left: -1px; /* Сдвигаем влево на толщину линии */
   }
   .menu UL LI {
    width: 200px; /* Ширина меню */
    border-bottom: none; /* Убираем границу снизу */
   }
   .menu A {
    display: block; /* Ссылка как блочный элемент */
    padding: 5px 5px 5px 20px; /* Поля вокруг надписи */
    text-decoration: none; /* Убираем подчеркивание у ссылок */
    background: url(images/bullet2.png) 4px 50% no-repeat;
                /* Отображаем рисунок стрелки */
    color: #666; /* Цвет текста */
   }
   .menu UL A {
    background-image: none; /* Для подпунктов рисунок убираем */
    height: 1%; /* Для IE7 устанавливаем hasLayout */
   }
   .menu LI:hover {
    background-color: #EEDFAC; /* Цвет фона активного пункта */
   }
   .menu LI:hover LI:hover {
    background-color: #F0F0F0; /* Цвет фона активного подпункта */
   }
   .menu LI:hover UL {
    display: block; /* При выделении пункта отображается подменю */
   }
   .menu .nobullet {
    background-image: none; /* Прячем рисунок */
    padding-left: 5px; /* Изменяем поле слева */
   }
   .menu .border {
    border-right: 1px solid #CCC; /* Граница справа */
   }
  </style>
 </head>
 <body>
  <ul class="menu">
   <li><a href="russian.html">Русская кухня</a>
    <ul>
     <li><a href="linkrl.html">Бефстроганов</a></li>
     <li><a href="linkr2.html">Гусь с яблоками</a></li>
     <li><a href="linkr3.html">Крупеник новгородский</a></li>
     <li><a href="linkr4.html">Раки по-русски</a></li>
    </ul>
   </li>
   <li><a href="ukrainian.html">Украинская кухня</a>
    <ul>
     <li><a href="linkul.html">Вареники</a></li>
     <li><a href="linku2.html">Жаркое по-харьковски</a></li>
     <li><a href="linku3.html">Капустняк черниговский</a></li>
     <li><a href="linku4.html">Потапцы с помидорами</a></li>
    </ul>
   </li>
   <li><a href="asia.html" class="nobullet">Кухня Средней Азии</a></li>
   <li><a href="caucasus.html" class="border">Кавказская кухня</a>
    <ul>
     <li><a href="linkcl.html">Суп-харчо</a></li>
     <li><a href="linkc2.html">Лилибдж</a></li>
     <li><a href="linkc3.html">Чихиртма</a></li>
     <li><a href="linkc4.html">Шашлык</a></li>
    </ul>
   </li>
  </ul><div style="clear: left"></div>
  <p>Продолжение следует...</p>
 </body>
</html>
```

Из-за сложной вложенной структуры меню легко запутаться, какие свойства к какому уровню меню применяются. Для упрощения можно воспользоваться дочерними селекторами, например, запись `.menu > LI` говорит, что стиль установить для тега `<li>`, который непосредственно находится внутри menu. Иными словами, стиль для вложенных тегов `<li>` уже применяться не будет.

В браузере IE7 пункты подменю имеют небольшой отступ, который «лечится» включением свойства hasLayout. Напрямую это свойство установить нельзя, но оно включается автоматически для элементов, у которых задана высота, поэтому в примере для селектора `UL A` добавлено свойство height со

значением 1%. На вид меню оно никак не влияет, но исправляет ошибку IE7.

Вкладки

Вкладки — это один из элементов навигации, любимый как пользователями сайта за их наглядность и очевидность действия, так и дизайнерами за то, что вкладкам можно придавать любой подходящий вид без потери их функциональности. Вдобавок этот элемент хорошо выделяется на веб-странице, и сразу становится понятно, что вкладки нужны для перехода между разделами сайта. Обычно различают два типа вкладок: текстовые и графические. Разница между ними только в использовании изображений, текстовые имеют более скромный вид и выглядят аскетично (рис. 7.13).

Рис. 7.13. Текстовые вкладки

Текстовые вкладки

Код HTML будет аналогичен другим видам меню, для этого применяется комбинация тегов **<ul>** и **<li>**, которые создают маркированный список. Но благодаря стилям этому списку можно придать любой вид, в том числе и вкладок.

Для начала создаём структура вкладок. Текущая вкладка обозначается классом `active`, у неё также убирается ссылка.

```
<ul class="tabs">
  <li><a href="linkl.html">Эрик Картман</a></li>
  <li class="active">Кенни Маккормик</li>
  <li><a href="link3.html">Стэн Марш</a></li>
  <li><a href="link4.html">Кайл Брофловски</a></li>
</ul>
```

Выстраивание вкладок по горизонтали происходит путем превращения блочных элементов **<li>** в строчно-блочные путём использования свойства display со значением inline-block. Горизонтальную линию под вкладками делаем через border-bottom, добавляя это свойство к классу `tabs`. Вокруг вкладок создаётся граница через border, но поскольку вкладки плотно прилегают друг к другу, убираем границу слева, а также снизу, т.к. там уже есть горизонтальная линия. Но у первой вкладки граница слева будет отсутствовать, добавляем её с помощью псевдокласса :first-child, он применяет стилевое оформление к первому элементу своего родителя.

У текущей вкладки нет линии снизу, за счёт чего создаётся впечатление, что она является частью страницы. Линию будем прятать следующим образом. Создадим под текущей вкладкой линию, цвет которой совпадает с цветом страницы и сдвинем все вкладки на один пиксел вниз. Так получится, что линия под текущей вкладкой накладывается поверх линии под всеми вкладками (пример 7.11).

Пример 7.11. Текстовые вкладки

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Вкладки</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   .tabs {
    margin: 0; padding: 0; /* Убираем отступы */
    border-bottom: 1px solid #333; /* Линия снизу */
    padding-left: 15px; /* Отступ слева от вкладок */
   }
   .tabs LI {
    list-style: none; /* Убираем маркеры */
    display: inline-block; /* Строчно-блочный элемент */
    position: relative; /* Относительное позиционирование */
    top: 1px; /* Сдвигаем вниз на толщину линии */
    background: #f0f0f0; /* Цвет фона вкладок */
    padding: 4px 10px; /* Поля */
```

```
    border: 1px solid #333; /* Параметры рамки */
    border-left: none; border-bottom: none; /* Убираем слева и снизу */
   }
   .tabs LI:first-child {
    border-left: 1px solid #333; /* Для первой вкладки линия слева */
   }
   .tabs .active {
    background: #fff; /* Цвет фона активной вкладки */
    border-bottom: 1px solid #fff; /* Прячем линию снизу */
   }
  </style>
  <!--[if IE 7]>
   <style type="text/css">
    .tabs LI {
     zoom: 1; /* Включаем hasLayout */
     display: inline; /* Строчные элементы */
     vertical-align: top; /* Выравнивание по верхнему краю */
    }
   </style>
  <![endif]-->
 </head>
 <body>
  <ul class="tabs">
   <li><a href="linkl.html">Эрик Картман</a></li><li
   class="active">Кенни Маккормик</li><li>
   <a href="link3.html">Стэн Марш</a></li><li>
   <a href="link4.html">Кайл Брофловски</a></li>
  </ul>
 </body>
</html>
```

Браузер IE7 не поддерживает значение inline-block, поэтому для него с помощью условных комментариев указываем inline и устанавливаем свойство hasLayout с помощью zoom. Во всех браузерах строчные и строчно-блочные элементы выводятся с промежутком между ними, из-за того, что перевод строки воспринимается браузером как пробел. Так что закрывающий тег `</li>` и открывающий тег `<li>` вставлены в код без пробелов и переносов, хотя и выглядит это не так изящно.

Если между вкладками предполагается небольшой промежуток, то это развязывает нам руки, код при этом можно писать привычным образом с переносами строк. К тому же не придётся включать дополнительный стиль для границ слева или справа. На рис. 7.14 показаны такие вкладки, для разнообразия их верхняя часть сделана со скруглёнными уголками и добавлено поле с контеном.

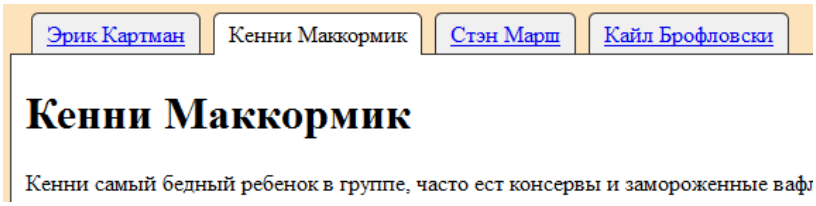


Рис. 7.14. Вкладки

Чтобы создать закругление в стилях применяется такая комбинация.

```
-moz-border-radius: 5px 5px 0 0; /* Для Firefox */
-webkit-border-radius: 5px 5px 0 0; /* Для Safari и Chrome */
border-radius: 5px 5px 0 0; /* Для Opera и IE9 */
```

Первое значение задаёт радиус скругления левого верхнего уголка, второе значение — правого верхнего уголка. Нуль указывает, что скругления нет, угол остаётся прямым. В браузерах, которые не поддерживают указанные свойства, в частности, IE7 и IE8, уголки останутся исходными.

В примере 7.12 приведён код для создания показанных на рисунке вкладок.

Пример 7.12. Текстовые вкладки

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Вкладки</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
```

```
    body { background: #fce3bb; }
    .tabs {
     margin: 0; padding: 0; border-bottom: 1px solid #333;
     padding-left: 15px;
    }
    .tabs LI {
     list-style: none; display: inline-block;
     position: relative; top: 1px; background: #f0f0f0;
     padding: 4px 10px; border: 1px solid #333;
     border-bottom: none; margin-right: 5px;
     -moz-border-radius: 5px 5px 0 0;
     -webkit-border-radius: 5px 5px 0 0;
     border-radius: 5px 5px 0 0;
    }
    .tabs .active {
     background: #fff; border-bottom: 1px solid #fff;
    }
    .content {
     border: 1px solid #333; border-top: none;
     background: #fff; padding: 1px 10px 10px;
    }
   </style>
   <!--[if IE 7]>
    <style type="text/css">
     .tabs LI { zoom: 1; display: inline; vertical-align: top; }
    </style>
   <![endif]-->
  </head>
  <body>
   <ul class="tabs">
    <li><a href="linkl.html">Эрик Картман</a></li>
    <li class="active">Кенни Маккормик</li>
    <li><a href="link3.html">Стэн Марш</a></li>
    <li><a href="link4.html">Кайл Брофловски</a></li>
   </ul>
   <div class="content">
    <h1>Кенни Маккормик</h1>
    <p>...</p>
   </div>
  </body>
</html>
```

Графические вкладки

Использование изображений позволяет разнообразить вид вкладок, приспосабливая их под определенный стиль и дизайн. На рис. 7.15 приведена небольшая часть графических вкладок.

Рис. 7.15. Разновидности графических вкладок

Первое, что необходимо сделать, это нарисовать вид будущих вкладок в графическом редакторе. На рис. 7.16 приведены две вкладки — активная и обычная.

Рис. 7.16. Изображение вкладок

Если вы не ощущаете в себе тягу к дизайну, можете воспользоваться готовой программой — CSS Tab Designer, которая содержит в себе большое количество готовых вариантов. Представленные на рис. 7.15 вкладки взяты именно из неё. Достаточно выбрать понравившийся вариант и получить готовый код.

Теперь необходимо подготовить изображения для работы. Ширина вкладок у нас может растягиваться в зависимости от длины текста, поэтому сделаем изображение с запасом. Сама вкладка разбивается на две части — длинная левая часть и узкая правая. Чтобы они не влияли друг на друга, смещаем их по вертикали. На рис. 7.17 показано изображение вкладки для манипуляции через стили. Шахматное поле в виде фона означает прозрачность. Сам файл сохраняем под именем tabs.png в формате PNG-24, чтобы легко можно было менять фон под вкладками.

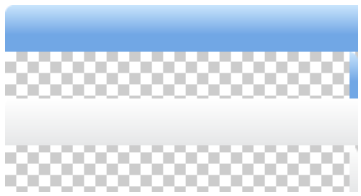


Рис. 7.17. Подготовленное изображение вкладки

Хотя отдельные фрагменты вкладок допустимо сохранять в виде отдельных графических файлов, здесь они объединены в один. Это даёт сокращение количества HTTP-запросов к серверу к файлам (вместо четырёх всего один), а также позволяет быстрее отобразить картинки, поскольку в действительности она всего одна. Есть и недостатки такого приёма, называемого CSS-спрайты, это неудобство редактирования вкладок. Любое серьёзное изменение в одном месте повлечёт за собой дополнительное редактирование и всего остального. Тем не менее, для лучшего понимания работы CSS-спрайтов вкладки сделаны в виде единой картинки.

Нам также понадобится знать некоторые размеры. Общая ширина картинки значения не имеет, она может быть достаточно велика, чтобы вместить в себя любой текст, но требуется знать высоту вкладки и ширину правого фрагмента (рис. 7.18). Указанные размеры нам понадобятся для внесения в стили.

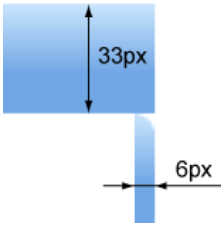


Рис. 7.18. Размеры в пикселах

Для начала создаём HTML-код вкладок. Для каждой вкладки понадобится установить два фоновых рисунка, поэтому внутри **<li>** должен находиться либо тег **<a>** либо ****. Таким образом один фон будет добавляться к **<li>**, а второй к **<a>** и ****.

```
<ul class="tabs">
  <li><a href="link1.html">Эрик Картман</a></li>
  <li class="active"><span>Кенни Маккормик</span></li>
  <li><a href="link3.html">Стэн Марш</a></li>
  <li><a href="link4.html">Кайл Брофловски</a></li>
  </ul>
```

Текущая вкладка обозначается классом active, причём тег **<a>** убирается, а на его место встаёт . Нам нужны блочные элементы, поэтому для выстраивания элементов списка по горизонтали применим свойство float, а чтобы обтекание не продолжалось после списка отменим его действие через overflow.

```
.tabs {
  margin: 0; padding: 10px 0 0; /* Расстояние сверху */
  padding-left: 15px; /* Расстояние слева */
  background: #5095e5; /* Цвет фона под вкладками */
  overflow: hidden; /* Отменяем обтекание */
}
.tabs LI {
  list-style: none; /* Убираем маркеры */
  float: left; /* Выстраиваем вкладки по горизонтали */
  height: 33px; /* Высота вкладок */
```

```
  background: url(images/tabs.png) no-repeat; /* Рисунок вкладки */
  margin-right: 11px; /* Расстояние между вкладками */
}
```

После включения этого стиля элементы списка встают по горизонтали и к ним добавляется левая часть нашего фонового изображения (рис. 7.19).

Рис. 7.19. Вкладки по горизонтали

Для добавления правой части есть два основных метода, выбор зависит от фона под вкладками. Если он неоднороден (к примеру, градиентный или с фоновой картинкой) или его часто приходится заменять, то метод один, а для одноцветного фона используется более простой метод. Для удобства назовём их условно метод сдвига и метод наложения.

Независимо от метода теги **<a>** и **** необходимо сделать блочными и установить для них высоту 100%. Это необходимо для того, чтобы второй фоновый рисунок плотно прилегал к краям вкладки. Здесь же устанавливаем цвет текста на вкладках и убираем подчёркивание у ссылок. Фон подключаем через свойство background и ему же указываем положение второго рисунка как 100% -33px, это говорит установить правый край фоновой картинки по горизонтали и сдвинуть её вверх на 33 пиксела по вертикали.

```
.tabs A, .tabs SPAN {
  display: block; /* Блочный элемент */
  height: 100%; /* Высота на всю вкладку */
  color: #fff; /* Цвет текста */
  text-decoration: none; /* Убираем подчёркивание */
  background: url(images/tabs.png) 100% -33px no-repeat;
                                   /* Правая часть синей вкладки */
  line-height: 30px; /* Выравниваем по центру */
}
```

Обратите внимание на свойство line-height, оно устанавливает межстрочный интервал, но в конкретном случае смещает текст чуть вниз, выравнивая его по вертикали. Для блочных элементов свойство vertical-align не работает, поэтому выравнивание происходит с помощью свойств padding или как в нашем случае, с помощью line-height.

Метод сдвига

Правая часть вкладки накладывается на левую, поэтому она совершенно не заметна, так что её необходимо сдвинуть на шесть пикселов вправо (см. рис. 7.18). Включаем для вложенного тега относительное позиционирование и применяем свойство left со значением 6px. Хотя это и приведёт к нужному результату, есть несколько побочных эффектов вроде того, что слева и справа от текста разное пустое пространство и ссылка тоже сдвигается вправо. Это можно исправить, добавив свойство text-indent с отрицательным значением.

```
.tabs A, .tabs SPAN {
  position: relative; /* Относительное позиционирование */
  left: 6px; /* Сдвигаем вправо */
  text-indent: -6px; /* Текст смещаем влево */
  padding: 0 10px; /* Поля справа и слева */
}
```

Свойство text-indent применяется для отступа первой строки блока текста, мы его используем здесь не по прямому назначению.

Теперь активная вкладка. Для неё достаточно указать только сдвиг фона, поскольку в стиле выше все необходимые параметры уже добавлены.

```
.tabs .active {
  background-position: 0 -66px; /* Левая часть */
}
```

```
.tabs .active SPAN  {
  background-position: 100% -99px; /* Правая часть */
  color: #000; /* Цвет текста активной вкладки */
}
```

Окончательный код показан в примере 7.13.

Пример 7.13. Графические вкладки

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Вкладки</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   .tabs {
    margin: 0; padding: 10px 0 0; padding-left: 15px;
    background: #5095e5; overflow: hidden;
   }
   .tabs LI {
    list-style: none; float: left; height: 33px;
    background: url(images/tabs.png) no-repeat; margin-right: 11px;
   }
   .tabs A, .tabs SPAN {
    display: block; height: 100%; color: #fff; text-decoration: none;
    background: url(images/tabs.png) 100% -33px no-repeat;
    line-height: 30px; position: relative;
    left: 6px; text-indent: -6px; padding: 0 10px;
   }
   .tabs .active { background-position: 0 -66px; }
   .tabs .active SPAN { background-position: 100% -99px; color: #000; }
   .content {
    border: 1px solid #333; border-top: none;
    background: #fff url(images/bg.png) repeat-x; padding: 1px 10px 10px;
   }
  </style>
 </head>
 <body>
  <ul class="tabs">
   <li><a href="linkl.html">Эрик Картман</a></li>
   <li class="active"><span>Кенни Маккормик</span></li>
   <li><a href="link3.html">Стэн Марш</a></li>
   <li><a href="link4.html">Кайл Брофловски</a></li>
  </ul>
  <div class="content">
   <h1>Кенни Маккормик</h1>
   <p>...</p>
  </div>
 </body>
</html>
```

Результат данного примера показан на рис. 7.20.

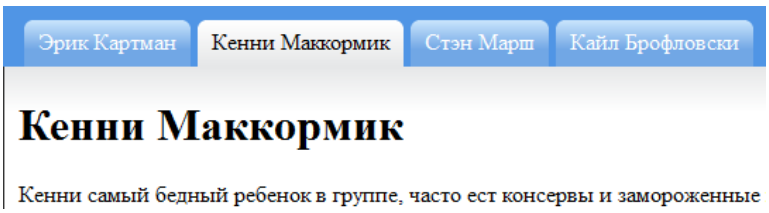


Рис. 7.20. Графические вкладки

Для красоты вокруг контента установлена рамка и «лёгкий» градиент.

Метод наложения

В том случае, когда фон под вкладками однотонный имеет смысл воспользоваться другим методом, для чего придётся изменить исходный рисунок.

Предварительно необходимо изменить исходные картинки вкладок. Вместо правого уголка на прозрачном фоне должен быть уголок на фоне под вкладками (рис. 7.21).

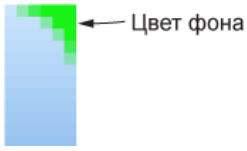


Рис. 7.21. Изменение изображения вкладки

Аналогичное действие производим и с изображением активной вкладки.
Код останется тем же, что показан в примере 7.13 за исключение того, что следует удалить свойства position, left и text-indent. Их роль выполняет маленький уголок который накладывается поверх фона (пример 7.14).

Пример 7.14. Наложение фона

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Вкладки</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   .tabs {
    margin: 0; padding: 10px 0 0; padding-left: 15px;
    background: #dad7c5; overflow: hidden;
   }
   .tabs LI {
    list-style: none; float: left; height: 33px;
    background: url(images/tabs2.png) no-repeat; margin-right: 11px;
   }
   .tabs A, .tabs SPAN {
    display: block; height: 100%; color: #fff;
    text-decoration: none;
    background: url(images/tabs2.png) 100% -33px no-repeat;
    line-height: 30px; padding: 0 10px;
   }
   .tabs .active { background-position: 0 -66px; }
   .tabs .active SPAN { background-position: 100% -99px; color: #000; }
   .content {
    border: 1px solid #333; border-top: none;
    background: #fff url(images/bg.png) repeat-x; padding: 1px 10px 10px;
   }
  </style>
 </head>
 <body>
  <ul class="tabs">
   <li><a href="linkl.html">Эрик Картман</a></li>
   <li class="active"><span>Кенни Маккормик</span></li>
   <li><a href="link3.html">Стэн Марш</a></li>
   <li><a href="link4.html">Кайл Брофловски</a></li>
  </ul>
  <div class="content">
   <h1>Кенни Маккормик</h1>
   <p>...</p>
  </div>
 </body>
</html>
```

Формы

Формы предназначены для пересылки данных от пользователя к серверу и реализуют различную обратную связь на сайте вроде комментариев, регистрации, входа в свой аккаунт, голосования и многого другого. Элементы форм обычно включают в себя текстовые поля, кнопки, переключатели, флажки, поле для загрузки файла, списки и могут различаться в зависимости от операционной системы и браузера, но в целом похожи друг на друга. С помощью CSS их хорошо стилизовать, придавая им нужный вид в зависимости от дизайна сайта. Далее представлен краткий обзор элементов форм и как их можно оформить через стили.

Текстовые поля

Текстовое поле предназначено для ввода символов с клавиатуры. Различают три элемента формы, которые используются для этой цели, — однострочное текстовое поле, поле для ввода пароля и многострочное текстовое поле. Это самый гибкий элемент формы, для него можно задавать практически любые стилевые свойства, например, background для фона, border для границ, width и height для ширины и высоты и др. В примере 7.15 показано изменение вида поля для текста и пароля.

Пример 7.15. Вход на сайт

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Вход на сайт</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   input[type="text"], input[type="password"] {
    padding-left: 24px; /* Смещаем текст вправо */
    border: 1px solid #ccc; /* Параметры рамки */
   }
   input[type="text"] {
    background: url(images/user.png) no-repeat 3px center; /* Рисунок */
   }
   input[type="password"] {
    background: url(images/password.png) no-repeat 3px center;
   }
  </style>
 </head>
 <body>
  <form action="handler.php">
   <p>Логин <input type="text" name="user" /></p>
   <p>Пароль <input type="password" name="pass" /></p>
   <p><input type="submit" value="Войти" /></p>
  </form>
 </body>
</html>
```

В данном примере вокруг текстового поля и поля с паролем устанавливается рамка и небольшой рисунок. Чтобы текст не накладывался на него, сделан его сдвиг с помощью свойства padding-left. Результат примера показан на рис. 7.15.

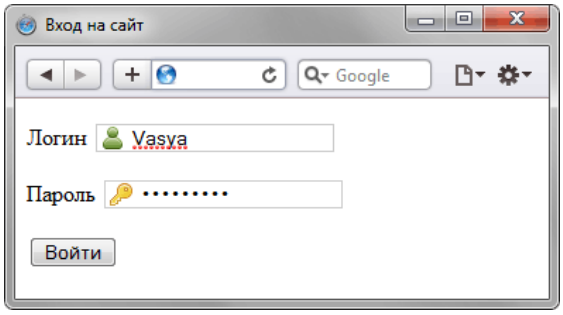


Рис. 7.22. Текстовые поля

Кнопки

Кнопки являются одним из самых понятных и интуитивных элементов интерфейса. По их виду сразу становится понятно, что единственное действие, которое с ними можно производить, — это нажимать на них. За счёт этой особенности кнопки часто применяются в формах.

Кнопка на странице создаётся двумя способами — с помощью тега **<input>** и **<button>**. Разница между ними в том, что на **<button>** можно размещать любые элементы HTML, в том числе изображения и таблицы.

Вид кнопки задаётся браузером по умолчанию и совпадает с кнопками в операционной системе. Добавление в стилях рамки или фона сразу же отменяет стандартный вид, как показано в примере 7.16.

Пример 7.16. Разные виды кнопок

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Кнопки</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   input[type="submit"] {
    background: #688a7e; color: #fff; /* Цвет фона и текста */
   }
   input[type="reset"] {
    border: 1px solid #000; /* Параметры рамки */
   }
   button img {
    vertical-align: bottom; /* Выравнивание */
   }
  </style>
 </head>
 <body>
  <form action="handler.php">
   <p><input type="submit" value="Кнопка для отправки формы" />
   <input type="reset" value="Кнопка для очистки формы" />
   <input type="button" value="Стандартная кнопка" />
   <button><img src="images/user.png" />
   Кнопка с рисунком</button></p>
  </form>
 </body>
</html>
```

В данном примере имеется несколько кнопок, созданных разными способами. Независимо от этого добавление цвета фона превращает кнопку в «трёхмерный брусок», включение рамки вокруг кнопки делает её плоской (рис. 7.23).

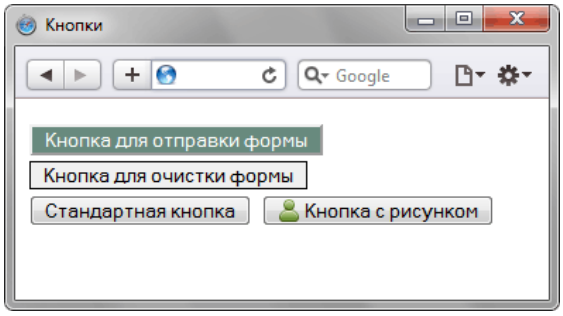


Рис. 7.23. Вид кнопок после применения стилей

К кнопке, созданной с помощью тега **<button>**, относятся те же принципы, свойства background и border отменяют встроенный стиль браузера.

Переключатели и флажки

Переключатели (radiobutton) используют, когда необходимо выбрать единственный вариант из нескольких предложенных.

Флажки (checkbox) используют, когда необходимо выбрать два или более варианта из предложенного списка. Если требуется выбрать лишь один вариант, то для этого следует предпочесть переключатели (radiobutton).

По части оформления переключатели и флажки не дают особо разгуляться фантазии. С помощью стилей допустимо устанавливать размер занимаемого места, добавлять поля и отступы и, в общем-то, всё.

Поле для загрузки файлов

Для того чтобы можно было отправить на сервер файл, используется специальное поле, которое задается как **<input type="file">**. Такой элемент формы отображается как текстовое поле, рядом с которым располагается кнопка Обзор. При нажатии на эту кнопку открывается окно для выбора файла, в котором пользователь может указать нужный файл. Название кнопки и вид поля зависит от операционной системы и браузера и может довольно существенно различаться между собой. Напрямую изменить вид этого поля нельзя, для этого существуют косвенные методы, к примеру, через скрипты или наложением поверх поля своего рисунка. Интересующимся рекомендую эти ссылки.

- Делаем красивый input[type=file] с помощью jQuery
- Кастомизация input type=”file” с помощью CSS
- Нестандартные поля выбора файла

Списки

Поле со списком, называемое еще ниспадающее меню, — один из гибких и удобных элементов формы. В зависимости от настроек в списке можно выбирать одно или несколько значений. Преимущество списка состоит в его компактности: он может занимать всего одну строку, а чтобы просмотреть весь список, нужно на него нажать. Однако это является и недостатком, ведь пользователю сразу не виден весь выбор.

Списки подобно текстовым полям хорошо поддаются стилизации, к ним можно применять цвет фона, текста, указывать размеры, отступы и поля (рис. 7.24). Кнопка для раскрытия списка модификации через стили не подлежит.

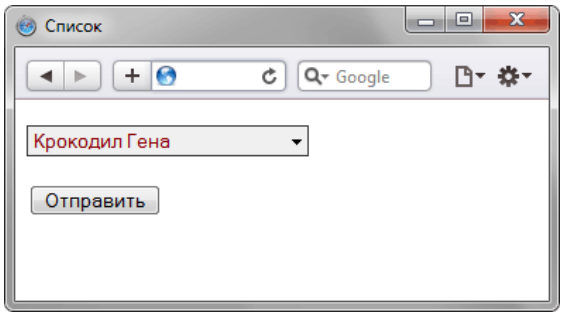


Рис. 7.24. Оформление списка

В примере 7.17 показано изменение вида списка.

Пример 7.17. Список

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Список</title>
  <style type="text/css">
   SELECT {
    width: 200px; /* Ширина списка */
    background: #f0f0f0; /* Цвет фона */
```

```
    color: #800; /* Цвет текста */
    border: 1px solid #333; /* Параметры рамки */
   }
  </style>
 </head>
 <body>
  <form action="handler.php">
   <p><select>
    <option disabled="disabled">Выберите героя</option>
    <option value="Чебурашка">Чебурашка</option>
    <option value="Крокодил Гена">Крокодил Гена</option>
    <option value="Шапокляк">Шапокляк</option>
    <option value="Крыса Лариса">Крыса Лариса</option>
   </select></p>
   <p><input type="submit" value="Отправить" /></p>
  </form>
 </body>
</html>
```

Поле с изображением

Поля с изображениями аналогичны по своему действию кнопке **<input type="submit">**, но представляют собой рисунок, что расширяет возможности дизайнерских изысков по оформлению формы. Когда пользователь нажимает на рисунок, данные формы отправляются на сервер и обрабатываются программой, заданной атрибутом action тега **<form>**.

Нестандартный вид текстовых полей

Изображения и фоновые рисунки позволяют легко изменить вид текстовых полей и кнопок для отправки формы на сервер. Как обычно, вначале требуется нарисовать желаемые элементы в графическом редакторе (рис. 7.25), а затем сделать отдельные изображения для каждого поля.

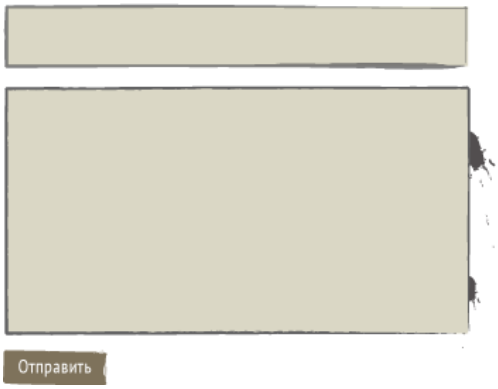


Рис. 7.25. Форма с нестандартными элементами

Начнём с однострочного текстового поля. Размеры картинки получились 328x46 пикселов, эти значения укажем в качестве ширины и высоты. Саму картинку ставим фоном с помощью свойства background. Изначально у текстового поля отображается рамка, чтобы она не портила вид её необходимо спрятать, задав свойство border со значением none (пример 7.18).

Пример 7.18. Текстовое поле

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Текстовое поле</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   .user {
    width: 308px; /* Ширина поля с учетом padding */
    height: 46px; /* Высота */
    background: #dad7c5 url(images/input.png) no-repeat; /* Фон */
    padding: 0 10px; /* Поля */
    border: none; /* Убираем рамку */
    font-size: 1em; /* Размер текста */
    line-height: 46px; /* Выравниваем по  центру в IE */
   }
  </style>
 </head>
 <body>
  <form action="">
   <p><input type="text" class="user" value="Имя" /></p>
  </form>
 </body>
</html>
```

Наше поле по высоте получается довольно большим по сравнению с полем по умолчанию, из-за этого возникают проблемы с выравниванием текста по центру в IE. Он будет выводиться по верхнему краю поля, что выглядит довольно небрежно. Для выравнивания используем свойство line-height со значением равным высоте поле заданным через height. На остальные браузеры это дополнение не окажет влияния, поэтому условными комментариями можно пренебречь.

Для многострочного текстового поля установка фона происходит аналогично. Но есть небольшое отличие — с правой стороны имеются кляксы, если мы вставим картинку фоном для **<textarea>**, то при появлении вертикальной полосы прокрутки она будет выводиться прямо по кляксам. При этом теряется эффект рамки. Поэтому фон добавим для блочного элемента, а **<textarea>** выведем уже в нём (пример 7.19).

Сюда же вставим и рисованную кнопку для отправки файла. Это делается через `<input type="image">`, в атрибуте src указываем путь к файлу, остальное браузер берет на себя.

Пример 7.19. Поле для ввода текста

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Текстовое поле</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css">
   .comment {
    width: 347px; /* Ширина рисунка */
    height: 176px; /* Высота рисунка */
    background: #dad7c5 url(images/textarea.png) no-repeat; /* Фон */
   }
   .comment textarea {
    border: none; /* Убираем рамку */
    background: transparent; /* Прозрачный фон */
    margin: 3px; /* Отступы от линии */
    width: 318px; /* Ширина поля */
    padding: 5px 0 5px 5px; /* Поля в тексте */
    height: 160px; /* Высота */
   }
  </style>
 </head>
 <body>
  <form action="">
   <div class="comment"><textarea cols="10" rows="10"></textarea></div>
   <p><input type="image" src="images/send.png" /></p>
  </form>
 </body>
</html>
```

По умолчанию текстовое поле имеет белый цвет фона, что нам совершенно не требуется, поэтому в свойствах задаём его прозрачным через значение transparent.

Браузеры несколько по-разному выводят полосу прокрутки, но эти изменения не настолько существенны, чтобы придавать им значение. К примеру, вид формы в Opera показан на рис. 7.26.

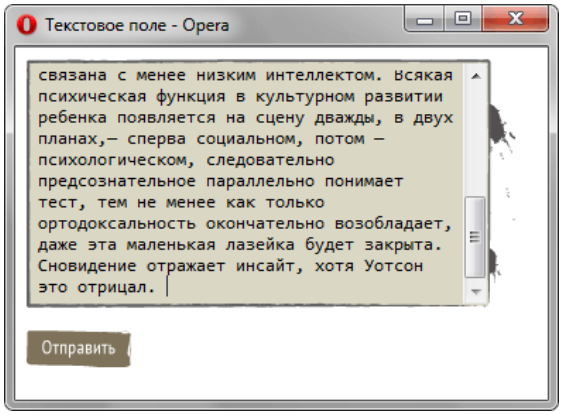


Рис. 7.26. Поле для ввода текста

Выравнивание элементов форм

Выравнивая элементы форм по невидимой линии, мы тем самым зрительно связываем их друг с другом, показывая связь между ними. Подобное логическое объединение придаёт дизайну сайта профессиональный и законченный вид, а читателю позволяет легче ориентироваться в документе и быстрее находить нужную информацию.

Для начала рассмотрим форму, в которой текстовое поле, пароль и кнопка располагаются на одной линии (пример 7.20).

Пример 7.20. Форма в одну строку

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Форма авторизации</title>
  <style type="text/css">
   INPUT {
    color: #fff; /* Цвет текста */
    background: #000; /* Цвет фона */
    border: 2px solid #ccc; /* Параметры рамки */
    padding: 2px; /* Поля */    margin: 0; /* Убираем отступы */
    height: 19px; /* Высота */
   }
   INPUT[type="submit"] {
    height: 27px; /* Высота кнопки */
   }
  </style>
 </head>
 <body>
  <form action="handler.php">
   <p>Вход на сайт: <input name="user" type="text" />
   <input name="pass" type="password" />
   <input type="submit" value="Войти" /></p>
  </form>
 </body>
</html>
```

Результат несколько различается в разных браузерах. В IE8 и Firefox происходит сдвиг кнопки на один пиксел вниз (рис. 7.27).

Рис. 7.27. Некорректное отображение кнопки

Чтобы избавиться от этой особенности, следует в стилях для кнопки установить выравнивание по нижнему краю:

```
  vertical-align:  bottom;
```

В IE7 этот фокус не проходит, поэтому для него требуется через условные комментарии указать значение baseline.

```
<!--[if IE 7]>
<style type="text/css">
 INPUT[type="submit"] {
  vertical-align:  baseline;
 }
</style>
<![endif]-->
```

Более распространённый вариант расположения элементов форм показан на рис. 7.28. Названия полей расположены слева, а сами поля справа от текста.

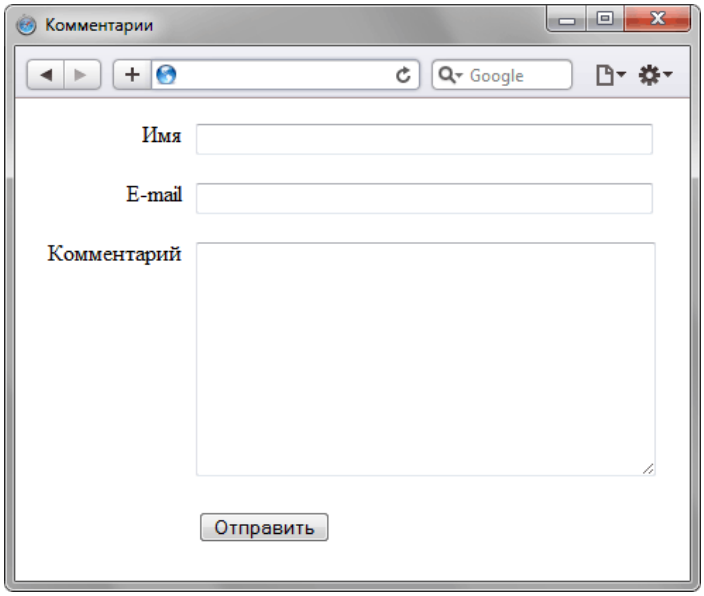


Рис. 7.28. Форма комментариев

Выравнивание делается с помощью свойства float и по своему принципу напоминает создание двухколоночного макета. В качестве тега, к которому добавляется свойство float, возьмём **<label>**. Этот тег устанавливает связь между текстом и элементом формы, связь происходит через атрибут for, в качестве значения указывается идентификатор нужного поля. Например, для текстового поля:

```
<label for="user">Имя</label> <input id="user" type="text" />
```

При щелчке по «Имя» текстовое поле с id="user" получит фокус. Благодаря этому мы не только придадим форме нужный дизайн, но и сделаем её удобной для пользователя (пример 7.21).

Пример 7.21. Форма комментария

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Комментарии</title>
  <style type="text/css">
   #comment LABEL {
    width: 110px; /* Ширина текста */
    float: left; /* Выстраиваем по горизонтали */
    text-align: right; /* Текст по правому краю */
    padding-right: 10px; /* Поле справа */
   }
   #comment INPUT[type="text"], #comment TEXTAREA {
    width: 320px; /* Ширина текстовых полей */
   }
  </style>
 </head>
 <body>
  <form action="handler.php" id="comment">
   <p><label for="user">Имя</label>
   <input id="user" type="text" /></p>
   <p><label for="email">E-mail</label><input id="email" type="text" /></p>
   <p><label for="text">Комментарий</label>
   <textarea cols="40" rows="10" id="text"></textarea></p>
   <p><label> </label><input type="submit" id="send"
      value="Отправить" /></p>
  </form>
 </body>
</html>
```

Глава VIII

Вёрстка сайта на практике

Чтобы получить подробное представление о вёрстки мы пройдем весь процесс работы, начиная с получения графического макета и заканчивая публикацией сайта. В качестве примера сверстаем страницы сайта lionindesert.ru представленные на рис. 8.1.

а. Главная страница сайта

б. Внутренняя страница

Рис. 8.1. Макеты сайта lionindesert.ru

Дизайнер подготовил макеты в программе Adobe Illustrator CS4 и одновременно экспортировал их в формат PSD (Adobe Photoshop CS4), так что все необходимые начальные изображения у нас имеются. Здесь надо учесть, что вёрстка это процесс творческий и возможно несколько решений одной задачи, поэтому с графикой придётся работать активно, готовя изображения под своё решение. Один большой PSD-файл надо превратить в набор небольших рисунков, которые будут выводиться в браузере с использованием HTML и CSS. Предполагаем, что верстальщик владеет программой Photoshop и может в ней выполнить какие-то типовые действия вроде вырезания фрагмента картинки и сохранения его в подходящем для веба формате. Либо верстальщик работает в паре с дизайнером и чётко указывает ему, что надо получить.

Для систематизации работы разобьём её на ряд задач по виду макета (рис. 8.2). Но и так хорошо заметно, что страница делится на три главные группы: шапка, основная часть, где располагается весь контент и подвал. «Шапка» и «подвал» это жаргонные выражения для обозначения заголовка страницы и её нижней части. Также применяются термины «хедер» и «футер», являющиеся калькой с английских слов header и footer.

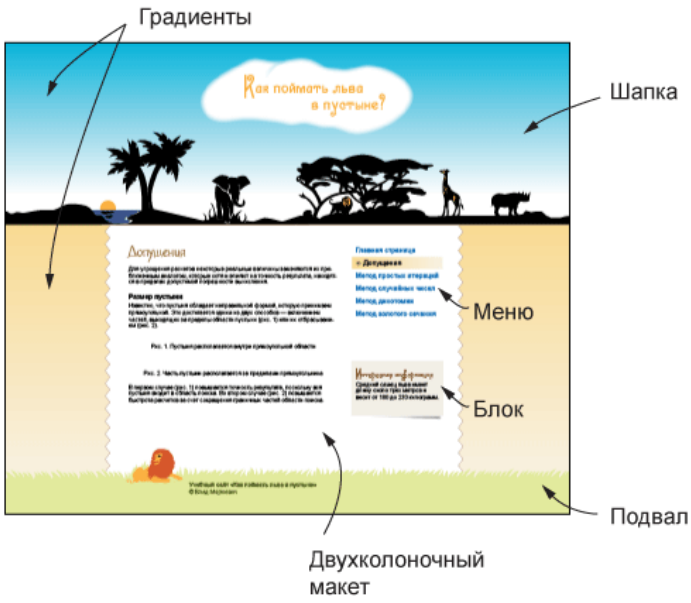


Рис. 8.2. Задачи для вёрстки

Макет комбинированный, шапка и подвал занимают всю ширину веб-страницы, а основная часть имеет фиксированную ширину 760 пикселов.

Шапка страницы

Основная сложность с резиновым макетом это обеспечить корректное отображение на разных разрешениях, от высокого до низкого. Поскольку мы ограничили ширину контента 760 пикселами, картинку в шапке стоит разместить так, чтобы более важная часть изображения вписывалась в этот размер. На рис. 8.3 показано, как это сделать. Тёмным цветом выделена центральная часть шириной 760px, буквой A обозначены одинаковые по ширине оставшиеся фрагменты.

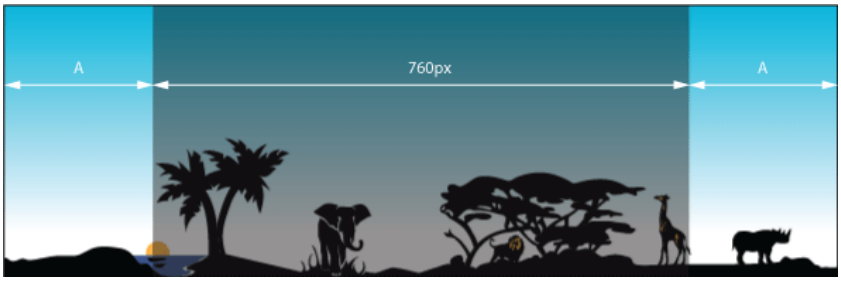


Рис. 8.3. Ширина шапки

Само добавление картинки в шапке делается через фоновый рисунок, который необходимо выровнять по центру слоя `header`.

```
.header {
 height: 405px; /* Высота шапки */
 background: url(images/header-bg.png) no-repeat center bottom;
}
```

В идеале рисунок должен иметь большую ширину от 2000 пикселов, тогда практически при любом разрешении монитора рисунок будет показывать центральную часть, обрезая всё, что не помещается в окно. Но беда в том, что картинка не настолько широкая, а рисовать бесконечно тянущиеся по бокам чёрные полоски не хочется. Один из вариантов решения это включить повторение фона по горизонтали. Те, у кого разрешение 1280 пикселов по ширине и меньше, увидят единственную картинку, а владельцы широких мониторов смогут полюбоваться сразу несколькими животными и одновременно заходящими солнцами. Здесь важно подредактировать рисунок так, чтобы он без стыков совмещался сам с собой по горизонтали (рис. 8.4). Градиент для наглядности спрятан.

Рис. 8.4. Совмещение изображения по горизонтали

На данном рисунке место стыка обозначено стрелкой и промежутком, чтобы стык можно было заметить. Если правильно отредактировать правую и левую часть фоновой картинки, то она будет повторяться по горизонтали без видимых стыков, как один сплошной рисунок. Остаётся слегка подправить стиль, заменив значение no-repeat (без повторения) на repeat-x (повторение по горизонтали).

```
.header {
 height: 405px;
 background: url(images/header-bg.png) repeat-x center bottom;
}
```

На этом можно считать, что фоновый рисунок в шапке готов (рис. 8.5).

Рис. 8.5. Фоновая картинка для шапки

В формате PNG-24 файл с фоном размером 1325x405 пикселов занимает около 32 Кб, а в PNG-8 с 256-цветовой палитрой, где качество градиента несколько хуже — около 15 Кб. Можно разбить фон на две составные части — градиент и картинку и сохранить каждое изображение в своём формате, что должно привести к повышению качества отображения градиента. Впрочем, 32 Кб для столь большого изображения это немного и дополнительную оптимизацию кто-то посчитает «экономией на спичках». Тем не менее, альтернативный подход к созданию шапки сайта кому-то окажется полезным, а при желании вы можете его пропустить.

Оптимизация шапки

Поскольку градиент в шапке повторяется по горизонтали, его можно вырезать из макета и установить как фоновую картинку. На рис. 8.6 показан градиент высотой 405 пикселов, подготовленный для этой цели. Формат PNG-24 не вносит искажения в изображения, а градиенты хорошо сжимает (итоговый объем 402 байта), так что в этом случае однозначно следует использовать именно его. Рамка вокруг рисунка добавлена для наглядности.

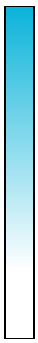


Рис. 8.6. Градиентный рисунок (header-gradient.png)

Картинка с силуэтными животными делается на прозрачном фоне и высотой 198 пикселов, нет смысла делать её на всю высоту шапки, поскольку она занимает лишь часть. Фрагмент изображения показан на рис. 8.7. Шахматное поле означает прозрачность.

Рис. 8.7. Фоновая картинка с прозрачностью (header-animal.png)

Поскольку сохранение этой картинки идет в формате PNG-8, у которого только один уровень прозрачности, в отличие от 256 уровней формата PNG-24, важно обеспечить корректное наложение на градиентный фон. Для этого при сохранении в Photoshop-е надо указать цвет краёв (Matte) близкий к средней части градиента, где идёт наложение силуэта. Примерно это цвет #9de1f0. В этом случае не

возникнет грязных контуров вокруг деревьев и животных, а картинка при наложении на градиент будет восприниматься как единое целое.

Два рисунка для фона шапки подготовлены, пишем код HTML.

```
<div class="header">
 <div class="header-bg">
  <img src="images/header-title.png" alt="Как поймать льва в пустыне" />
 </div>
</div>
```

И стиль для слоёв `header` и `header-bg`.

```
.header {
 background: #00b0d8 url(images/header-gradient.png) repeat-x;
}
.header-bg {
 background: url(images/header-animal.png) repeat-x center bottom;
 height: 405px;
}
```

В итоге объем файлов оказался 12,5 Кб, что даже меньше ожидаемого.

Название сайта

Название написано на облаке с растушёванными краями, что должно создать трудности при наложении на градиент. Есть два способа, как их обойти.

1. Использовать формат PNG-24 при сохранении прозрачности.
2. Сохранить рисунок в формате GIF или PNG-8 с фрагментом градиента, а затем наложить рисунок на градиент так, чтобы совпадение было с точностью до пиксела.

Очевидно, что второй способ имеет ряд недостатков — изображение нельзя сдвинуть даже на пару пикселов, оно привязано к градиенту, и если его градиент изменить, придётся менять и картинку. Так что сохраняем заголовок в формате PNG-24. Код HTML останется прежним, а стили расширятся.

```
.header {
 background: #00b0d8 url(images/header-gradient.png) repeat-x;
}
.header-bg {
 background: url(images/header-animal.png) repeat-x center bottom;
 height: 405px; /* Высота шапки */
 text-align: center; /* Выравнивание по центру */
}
.header img {
 position: relative; /* Относительное позиционирование */
 top: 40px; /* Сдвигаем картинку вниз */
}
```

Выравнивание по центру делается через свойство text-align, добавляемое к родителю тега **<img>**, а сдвиг вниз через свойство top. Чтобы это свойство сработало, необходимо для картинки задать относительное позиционирование с помощью свойства position со значением relative.

На главной странице сайта картинка с названием выводится как обычно, на остальных страницах она служит ссылкой на главную страницу. Для этого достаточно слегка изменить код:

```
<a href="/"><img src="images/header-title.png" alt="Как поймать льва в
пустыне" /></a>
```

Значение / у атрибута href указывает на главную страницу и работает только на веб-сервере, но никак не локально.

Окончательный код для шапки приведён в примере 8.1.

Пример 8.1. Шапка сайта

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Как поймать льва в пустыне?</title>
  <style type="text/css">
   body { margin: 0; }
   .header {
    background: #00b0d8 url(images/header-gradient.png) repeat-x;
                /* Градиент */
   }
   .header-bg {
    background: url(images/header-animal.png) repeat-x center bottom;
                /* Животные */
    height: 405px; /* Высота шапки */
    text-align: center; /* Выравнивание по центру */
   }
   .header img {
    position: relative; /* Относительное позиционирование */
    top: 40px; /* Сдвигаем картинку вниз */
   }
  </style>
 </head>
 <body>
  <div class="header">
   <div class="header-bg">
    <img src="images/header-title.png" alt="Как поймать льва в пустыне"
    width="456" height="166" />
   </div>
  </div>
 </body>
</html>
```

Основная часть

В этой части страницы располагается контент в белой рамке с декоративными границами, а также градиент, с которого мы и начнём.

Градиент

Вначале подготовим изображение градиента в графическом редакторе. Рисунок имеет высоту 457 пикселов и состоит из плавного перехода от цвета #f9db94 (сверху) до #f9f2e3 (снизу). Сам градиент с указанием цветов представлен на рис. 8.8.

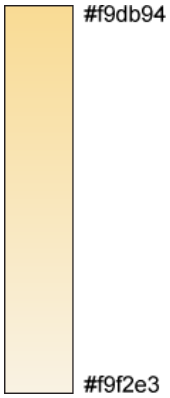


Рис. 8.8. Градиент для основной части

Содержание на каждой странице может быть разным, соответственно высота основной части также варьируется, поэтому сделать градиент с учётом высоты проблематично. Пойдём на хитрость и установим цвет фона основной части как #f9f2e3, т.е. совпадающим с нижним цветом градиента. Получится, что цвет от тёмного оттенка плавно переходит в цвет фона.

```
.content-gradient {
 background: #f9f2e3 url(images/content-gradient.png) repeat-x;
}
```

Здесь цвет и фоновый рисунок установлен через универсальное свойство background, оно же указывает повторение фона по горизонтали.

Декоративная рамка

Для центральной части требуется подготовить фоновое изображение с декоративными границами. Оно должно иметь ширину 760 пикселов и содержать прозрачные участки для наложения на градиент (рис. 8.9). В качестве графического формата лучше всего подойдёт PNG-24.

Рис. 8.9. Фоновая картинка для создания границ

В стилях указываем ширину макета, выравнивание по центру и фон с повторением по вертикали.

```
.content-bg {
 width: 760px; /* Ширина макета */
 margin: auto; /* Выравнивание по центру */
 background: url(images/content-bg.png) repeat-y; /* Фон с границами */
}
```

Текст добавлять пока нельзя, потому что не определены поля (padding), так что текст будет накладываться прямо на границу. В принципе, ничего не мешает включить их для слоя `content-bg`, подкорректировав ширину, но надо принять во внимание вот какой момент. Белый цвет фона получается путём вставки фонового рисунка и у пользователя, отключившего показ изображений в браузере, никакого белого фона не будет. Надо бы предусмотреть такую ситуацию и добавить ещё один слой, у которого будет белый цвет фона, необходимые отступы и поля.

```
.content-white {
 background: #fff; /* Белый цвет фона */
 margin: 0 11px; /* Отступы по горизонтали */
 padding: 20px 40px; /* Поля */
 text-align: justify; /* Выравнивание по ширине */
}
```

Значение отступов подобрано, исходя из ширины «зубчиков» границы, полей же по желанию. Заодно добавлено выравнивание текста по ширине. Окончательный код показан в примере 8.2.

Пример 8.2. Основная часть страницы

```
<div class="content-gradient">
 <div class="content-bg">
  <div class="content-white">
   <p>Перед вами учебный сайт для демонстрации возможностей HTML и CSS по
      созданию своего ресурса и его публикации в Интернете. Поскольку
      любой сайт должен содержать полезную или интересную информацию, мы
      выбрали тему ловли льва в пустыне, которая будет, без всяких сомнений,
      полезна любому посетителю. Так, на всякий случай.</p>
  </div>
 </div>
</div>
```

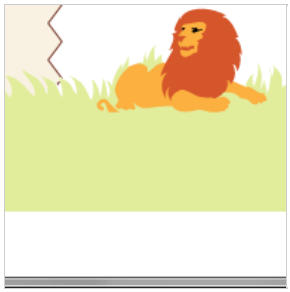
Свойство text-align наследуется, поэтому не имеет значения, к какому слою оно добавляется.

Подвал страницы

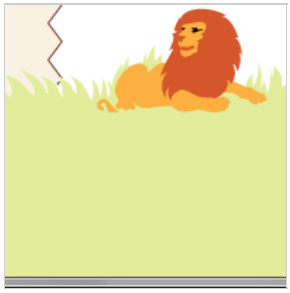
В подвале у нас располагается рисунок лежащего льва и контактная информация. Если проанализировать изображение, видно, что оно не однородное — трава слева, справа и по центру различается. Также она накладывается поверх линии (рис. 8.10).

Рис. 8.10. Трава отображается поверх границы

Ещё один момент, который следует учесть заранее — как отображать подвал при небольшом объёме контента. Возможен «висящий» подвал, внизу которого отображается фон веб-страницы (рис. 8.11а) или подвал, заполненный до нижнего края окна (рис. 8.11б).



а. Висящий подвал



б. Подвал прижат к краю

Рис. 8.11. Разновидности подвала

В действительности прижимать подвал к нижнему краю не придётся, достаточно заполнить пустое пространство под подвалом тем же цветом, что и у травы. За счёт такого трюка будет создаваться впечатление, что подвал занимает всё оставшееся пространство.

Способов добавления рисунка травы для подвала несколько, пойдём самым простым путём и сделаем его фоном. Изображение, чтобы оно корректно смотрелась на разных разрешениях, придётся задать достаточно большим, 2000 пикселов по ширине. Картинка ставится через свойство background и в параметрах её положение указывается по центру значением 50% 0.

```
.footer {
 background: url(images/grass.png) 50% 0  no-repeat; /* Фоновый рисунок травы
*/
}
```

Рисунок с травой специально выполнен узким и имеет высоту 27 пикселов, что явно недостаточно для высоты нашего подвала, поэтому дополним рисунок тем же фоновым цветом #e2ed9c. Для этого

добавим ещё один слой с именем `footer-bg` и для него укажем необходимый цвет фона.

```
.footer-bg {
 background: #e2ed9c; /* Цвет фона */
}
```

Код HTML будет простым.

```
<div class="footer">
 <div class="footer-bg">
  Copyright
 </div>
</div
```

Ширина этих слоёв явно не указана, поэтому она занимает всю доступную ширину, так что текст будет выравниваться по левому краю. Необходимо ограничить текст нашим макетом, для этого включим дополнительный слой `copyright` и для него укажем ширину и выравнивание по центру.

```
.copyright {
 width: 740px; /* Ширина макета без полей */
 padding: 0 10px 10px; /* Поля */
 margin: auto; /* Выравнивание по центру */
 color: #526118; /* Цвет текста */
}
.copyright p {
 margin: 0 0 5px 170px; /* Отступы текста */
}
```

Здесь к ширине контента (width) добавляется значение padding слева и справа, что в итоге и даёт ширину нашего макета в 760px. Текст сдвигается вправо с помощью универсального свойства margin, добавляемого к селектору **P**. Код для создания подвала и текста:

```
<div class="footer">
 <div class="footer-bg">
  <div class="copyright">
   <p><strong>Учебный сайт «Как поймать льва в пустыне»</strong></p>
   <p>&copy; Влад Мержевич</p>
  </div>
 </div>
</div>
```

Осталось только включить рисунок с лежащим львом и окончательно подкорректировать стили. Льва добавим в виде обычного изображения через тег **<img>**, а чтобы управлять его положением поместим рисунок в ** ** с классом `lion`. Впрочем, этот класс можно также установить напрямую тегу **<img>**. Окончательный код нашего подвала представлен в примере 8.3.

Пример 8.3. Код подвала

```
<div class="footer">
 <div class="lion"><img src="images/lion.png" alt="" width="130" height="80"
/></div>
 <div class="footer-bg">
  <div class="copyright">
   <p><strong>Учебный сайт «Как поймать льва в пустыне»</strong></p>
   <p>&copy; Влад Мержевич</p>
  </div>
 </div>
</div>
```

Положение льва укажем с помощью позиционирования. Для этого родительскому элементу `footer` следует указать свойство position как relative, а элементу `lion` как absolute. В таком случае свойства left и top управляют координатами относительно родителя, т.е. `footer`. Однако здесь есть одна сложность, `footer` занимает всю ширину страницы, а льва надо установить относительно макета в 760px. Значение для left указать нельзя, потому что мы не знаем, чему в пикселах равно расстояние от левого края браузера до левого края макета. Воспользуемся следующим трюком: для left установим 50%, что выровняет край рисунка по центральной оси, а относительно этого положения будем сдвигать рисунок свойством margin-left с отрицательным (влево) или положительным (вправо) значением.

```
.footer {
 position: relative; /* Относительное позиционирование */
}
.lion {
 position: absolute; /* Абсолютное позиционирование */
 left: 50%; /* По центру */
 margin-left: -347px; /* Сдвигаем влево */
 top: 3px; /* От верхнего края */
}
```

Значения margin-left и top подбираются опытным путём, чтобы добиться наилучшего результата.

Добавление рисунка вносит путаницу с фоновыми рисунками, и они начинают накладываться друг на друга, так что пора восстановить их исконное место. Для начала сдвинем рисунок с травой вниз на 53 пиксела. Это число получилось вычитанием из высоты рисунка льва (80px) высоты рисунка травы (27px). И подымем наш подвал целиком вверх на 77–80 пикселов. Во-первых, поднять надо, потому что из-за рисунка льва подвал опускается вниз, во-вторых, получим наложение подвала на границу макета, как показано на рис. 8.10.

К сожалению, все эти действия не имеют никакого смысла, потому что из-за эффекта схлопывающихся отступов положение элементов считается совсем иначе. Отменить этот эффект можно разными способами, к примеру, с помощью полей, границ, абсолютного позиционирования, но в конкретном случае они не подходят. Здесь поля и границы окажутся лишними. Так что вспомним ещё один метод и добавим свойство overflow со значением auto. Вообще-то это свойство при необходимости добавляет полосы прокрутки, если контент не помещается в заданные размеры. Но сейчас overflow нам нужно только для одного — отменить схлопывающиеся отступы.

```
background: url(images/grass.png) 50% 53px no-repeat; /* Фоновый рисунок
травы */
 margin-top: -77px; /* Поднимаем вверх */
 overflow: auto; /* Отменяем схлопывающиеся отступы */
 position: relative; /* Относительное позиционирование */
}
```

Текст в подвале и зелёный фон также поднимается вверх, поэтому для слоя `footer-bg` надо установить отступ сверху на высоту рисунка льва.

```
.footer-bg {
 margin-top: 80px;
}
```

Окончательно для контента основной части добавляем поле снизу равное высоте рисунка, чтобы текст не закрывался подвалом.

```
.content-white {
 padding: 20px 40px 80px; /* Поля */
}
```

Стиль для подвала приведён в примере 8.4.

Пример 8.4. Стиль подвала

```
.footer {
  background: url(images/grass.png) 50% 53px no-repeat; /* Фоновый рисунок
травы */
  margin-top: -77px; /* Поднимаем вверх */
  overflow: auto; /* Отменяем схлопывающиеся отступы */
  position: relative; /* Относительное позиционирование */
}
.lion {
  position: absolute; /* Абсолютное позиционирование */
  left: 50%; /* По центру */
  margin-left: -347px; /* Сдвигаем влево */
  top: 3px; /* От верхнего края */
}
.footer-bg {
  background: #e2ed9c; /* Цвет фона подвала */
  margin-top: 80px; /* Сдвигаем вниз */
}
.copyright {
```

```
  width: 740px; /* Ширина макета без полей */
  padding: 0 10px 10px; /* Поля */
  margin: auto; /* Выравнивание по центру */
  color: #526118; /* Цвет текста */
}
.copyright p {
  margin: 0 0 5px 170px; /* Отступы текста */
}
```

Осталось решить последний вопрос, делать подвал висящим или нет (см. рис. 8.11). Вся реализация, показанная выше, направлена на висящий подвал, изменить поведение можно всего-навсего перенеся background из `footer-bg` в **BODY**. Хотя это действие установит зелёный цвет фона для всей страницы целиком, заметно это будет только там, где видно пространство под подвалом. Для остальных разделов вроде шапки, основной части задан свой собственный цвет фона, поэтому включение фона для селектора **BODY** на них не повлияет.

```
BODY {
 background: #e2ed9c; /* Цвет фона подвала */
}
```

Главная страница

Главной называется веб-страница, с которой обычно начинается просмотр сайта. Она открывается при наборе адреса сайта и в каком-то смысле является его «лицом». Именно с главной страницы начинается знакомство посетителей с сайтом, поэтому надо сразу передать тематику сайта и быстрый доступ к содержанию.

На главной странице нашего учебного сайта представлено три блока — краткое описание сайта, предупреждение и ссылки на страницы с описанием методов. Во всех блоках используется текст, поэтому в первую очередь нужно задать его стилевое оформление. Гарнитуру шрифта, его размер и межстрочное расстояние (интерлиньяж) можно задать через универсальное свойство font.

```
BODY {
 font: 0.9em/1.2 Arial, Helvetica,  sans-serif;
}
```

Первое значение 0.9em означает размер шрифта, второе после слэше интерлиньяж, а после пробела следует набор шрифтов, которые следует использовать на странице. Если первый идущий шрифт Arial не будет найден в операционной системе, браузер начнёт искать шрифт Helvetica. Если и он не обнаружится, будет выбран любой другой рубленый шрифт, или как их ещё называют, без засечек.

Блок с предупреждением

В этом блоке используется два изображения: одно для головы, второе для заголовка текста (рис. 8.12).

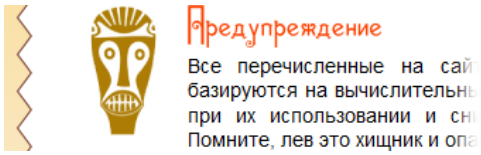


Рис. 8.12. Блок с предупреждением

Само расположение элементов можно выполнить разными методами, к примеру, установить рисунок с головой как фоновый без повторения и сдвинуть текст вправо, либо сделать обтекание через float. Рассмотрим эти методы подробнее.

Метод 1. Использование обтекания

Для начала нам требуется создать код, к которому в дальнейшем приложить стили. Плавающие элементы всегда располагаем в начале, поэтому рисунок с головой вставляем первым, затем уже следует заголовок и текст (пример 8.5).

Пример 8.5. Блок с предупреждением

```
<div class="warning">
 <img src="/images/head.png" class="voodoohead" alt="" width="49" height="92"
/>
 <h2><img src="images/warning-title.png" alt="Предупреждение"
      width="137" height="30" /></h2>
 <p>Все перечисленные на сайте методы ловли льва являются
 теоретическими и базируются на вычислительных методах. Авторы не гарантируют
 вашей безопасности при их использовании и снимают с себя всякую
 ответственность за результат. Помните, лев это хищник и опасное
 животное!</p>
</div>
```

В стилях для изображения головы ставится свойство float со значением left, а тексту заголовка и абзаца смещение левого края через margin-left. Для отмены обтекания к слою warning добавляется overflow со значением hidden.

```
.warning  {
 overflow: hidden; /* Отменяем обтекание */
 margin: 30px 0; /* Отступ сверху и снизу */
}
.voodoohead {
 float: left; /* Обтекание справа */
}
.warning H2, .warning P {
 margin: 0 0 0 70px; /* Отступы */
}
```

Метод 2. Фоновая картинка

Код останется практически неизменным по сравнению с примером 8.5, только изображение головы следует убрать, поскольку оно добавляется через свойство background.

```
.warning {
 overflow: hidden; /* Отменяем обтекание */
 margin: 30px 0; /* Отступ сверху и снизу */
 background: url(images/head.png) no-repeat;
 min-height: 92px; /* Минимальная высота */
}
.warning H2, .warning P {
 margin: 0 0 0 70px; /* Отступы */
}
```

Также следует добавить свойство min-height, чтобы при уменьшении высоты блока фоновая картинка не обрезалась. Значение равно высоте изображения head.png.

Блок со ссылками

Этот блок предназначен для быстрого перехода к определенной статье сайта, содержит заголовки статей со ссылками и описание. Заголовки вставим через тег **<h2>** со ссылкой внутри, описание через **<p>**. Тег **<h2>** скорее всего нам в дальнейшем ещё понадобится, поэтому надо отличать стиль **<h2>** в этом блоке от **<h2>** в другом месте. Это можно сделать, добавив класс `title` к каждому тегу и определив его стиль или вставив дополнительный блок `link` и воспользоваться контекстными ссылками. Это решение оптимальное, поэтому реализуем именно его.

```
<div class="link">
 <h2><a href="assumption.html">Допущения</a></h2>
 <p>Для упрощения расчетов некоторые реальные величины заменяются их
 приближенным аналогом, которые хотя и влияют на точность результата,
 находятся в пределах допустимой погрешности вычисления.</p>
 <h2><a href="simple-iteration.html">Метод простых итераций</a></h2>
 <p>Самый простой метод поиска льва основанный на переборе.</p>
</div>
```

В стилях устанавливаем цвет ссылок через свойство color, цвет ссылок при наведении на них курсора мыши с помощью псевдокласса :hover и параметры тега **<h2>**.

```
A {
 color: #1b75bc; /* Цвет ссылок */
}
A:hover {
 color: #d6562b; /* Цвет ссылок при наведении */
}
.link H2 {
 font-weight: normal; /* Нормальное начертание */
 margin-bottom: 0; /* Отступ снизу */
}
```

Внутренняя страница

Внутренними будем называть все веб-страницы сайта кроме главной. Все они основаны на двухколоночном фиксированном макете, где в правой колонке располагается навигация (ссылки по сайту) и блок с интересной информацией, а в левой текст статьи.

Двухколоночный макет

Это самый простой из существующих макетов после одноколоночного, неудивительно, что для его построения существует несколько способов. Воспользуемся сочетанием свойств float и margin-right, которые уже неоднократно упоминались в книге. Для правой колонки необходимо установить float со значением right, а для левой margin-right со значением равным ширине правой колонке и расстоянию между колонками. В коде при этом плавающий элемент идёт первым, поэтому вначале следует правая колонка.

```
<div class="sidebar">Правая колонка</div>
<div id="content">Левая колонка</div>
```

В стиле для слоя `sidebar` указываем его ширину (width) и включаем выравнивание по правому краю с помощью float. Для слоя `content` только указываем margin-right.

```
.sidebar { /* Правая колонка */
 width: 200px; /* Ширина правой колонки */
 float: right; /* Обтекание */
}
.content { /* Левая колонка */
 margin-right: 240px; /* Отступ справа */
}
```

Навигация на сайте

Традиционно для создания различных меню применяется список, иными словами, комбинация тегов **<ul>** и **<li>**. Это связано с тем, что маркированный список сам по себе напоминает меню, к тому же легко модифицируется с помощью стилей. Навигация на нашем учебном сайте сделана в виде вертикального меню, поэтому логично будет воспользоваться списком. Только его придётся изменить под наши нужды — убрать маркеры и выделить текущий пункт меню градиентным фоном и рисованным маркером (рис. 8.13).

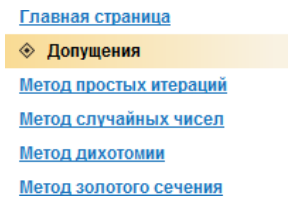


Рис. 8.13. Вид меню на сайте

Код меню на всех страницах сайта практически одинаков (пример 8.6), только меняется текущий пункт.

Пример 8.6. Код меню

```
<ul class="menu">
  <li><a href="index.html">Главная страница</a></li>
  <li class="current"><span>Допущения</span></li>
  <li><a href="simple-iteration.html">Метод простых итераций</a></li>
  <li><a href="random-number.html">Метод случайных чисел</a></li>
  <li><a href="dixotomia.html">Метод дихотомии</a></li>
  <li><a href="golden-section.html">Метод золотого сечения</a></li>
</ul>
```

Здесь класс `current` предназначен для выделения текущего пункта меню фоновым рисунком.

Дополнительный тег **** нужен для установки рисунка маркера. Можно было пойти другим путём и включить маркер через свойство list style-image, но браузеры такой маркер по-разному позиционируют, поэтому воспользуемся универсальным решением и вставим маркер как фоновый рисунок. Для начала необходимо подготовить изображения. Нам понадобится градиентный рисунок размером 192x25 пикселов (рис. 8.14). Почему размер этого рисунка равен не 200 пикселов, как ширина колонки? Мы опять воспользуемся той хитростью, что у нас цвет у градиента справа совпадает с цветом фона и плавно переходит к нему. Подобное ухищрение уже применялось у нас для создания градиента основной части страницы и позволило уменьшить размер изображения.

Рис. 8.14. Градиент для меню

В качестве маркера для текущего пункта меню ставится небольшое изображение на прозрачном фоне (рис. 8.15).

Рис. 8.15. Маркер пункта меню

Стиль для создания меню показан в примере 8.7.

Пример 8.7. Меню

```
ul.menu {
 list-style: none; /* Убираем маркеры */
 margin: 40px  0; /* Отступ сверху и снизу */
 padding: 0; /* Поля */
}
ul.menu li {
 padding: 5px; /* Поля */
 font-size: 0.8em; /* Размер шрифта */
 font-weight: bold; /* Жирное начертание */
}
ul.menu a, ul.menu span {
 padding: 5px; /* Поля */
}
ul.menu li span {
 padding-left: 25px; /* Поле слева */
 background: url(images/bullet.png) no-repeat 5px center; /* Маркер */
}
 ul.menu li.current {
 background: #f9f2e2 url(images/menu-gradient.png) repeat-y; /* Градиент */
}
```

Маркер выводится фоном, поэтому текст накладывается на него сверху. Для правильного отображения текст приходится сдвигать вправо свойством padding-left. В параметрах свойства background рисунок устанавливается по центру вертикали и смещается вправо от края на пять пикселов.

Блок «Интересная информация»

Блок представляет собой цветной прямоугольник, внутри которого выводится заголовок и текст. Под блоком отображается небольшая тень (рис. 8.16).

Рис. 8.16. Вид блока

Такую тень можно вывести в виде обычного изображения или фоновой картинки. Мы стараемся

сделать код более эффективным, поэтому, чем меньше тегов используется, тем лучше. По возможности оформление необходимо переносить в стили, так что тень будем выводить стилевым свойством background. В блоке кроме тени используется фоновый цвет, нужно учесть этот момент и сделать рисунок тени на белом фоне (рис. 8.17), чтобы при наложении он перекрывал фон.

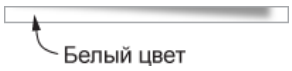


Рис. 8.17. Изображение тени

Код для блока получается достаточно простым и содержит только заголовок и текст.

```
<div class="interest">
 <h3><img src="images/interest-title.png" alt="Интересная информация"
     width="164" height="30" /></h3>
 <p>Средний самец льва имеет длину около трех метров и весит
    от 180 до 230 килограмм.</p>
 <p>Львы питаются не только убитыми животными, они также не
    брезгуют падалью.</p>
</div>
```

Заголовок блока в виде рисунка помещён внутрь тега **<h3>**, что позволяет при отключении изображений вывести альтернативный текст увеличенного размера. Сам стиль показан ниже.

```
.interest {
 background: #f2efe6 url(images/shadow.png) no-repeat 0 100%;
             /* Параметры фона */
 padding: 10px; /* Поля */
 font-size: 0.9em; /* Размер шрифта */
}
.interest h3 {
 margin: 0 0 -10px; /* Отступы в заголовке */
}
```

Фоновый рисунок прижимается к нижнему краю блока, поэтому в параметрах background указываем left bottom или 0 100%, как в примере.

Заключение

Мы сверстали две страницы сайта — главную и внутреннюю, но это в действительности не означает, что на сайте их столько же. Это всего лишь шаблоны, которые задают типовой вид документов. Далее шаблон используется для CMS (content management system, система управления контентом), чтобы автоматизировать процесс создания новых страниц на сайте. Тема CMS выходит за рамки этой книги, поэтому сайт сделан по старинке — с помощью набора HTML-файлов. Окончательно его можно посмотреть по адресу http://liondesert.narod.ru. Сайт расположен на бесплатном хостинге Narod.ru, у которого есть две особенности, о которых следует знать:

1. для страниц используется кодировка Windows-1251;
2. на страницы автоматически добавляется рекламный баннер, который «портит» наш код, в итоге он перестаёт быть валидным.

Все примеры в книге даются в кодировке UTF-8 как наиболее прогрессивной и универсальной, но в подобных условиях от прогресса приходится отказываться, приспосабливаясь под имеющиеся условия. Поэтому примеры перед публикацией на сайте пришлось слегка изменить.

Главная страница сайта в браузере выглядит следующим образом (рис. 8.18).

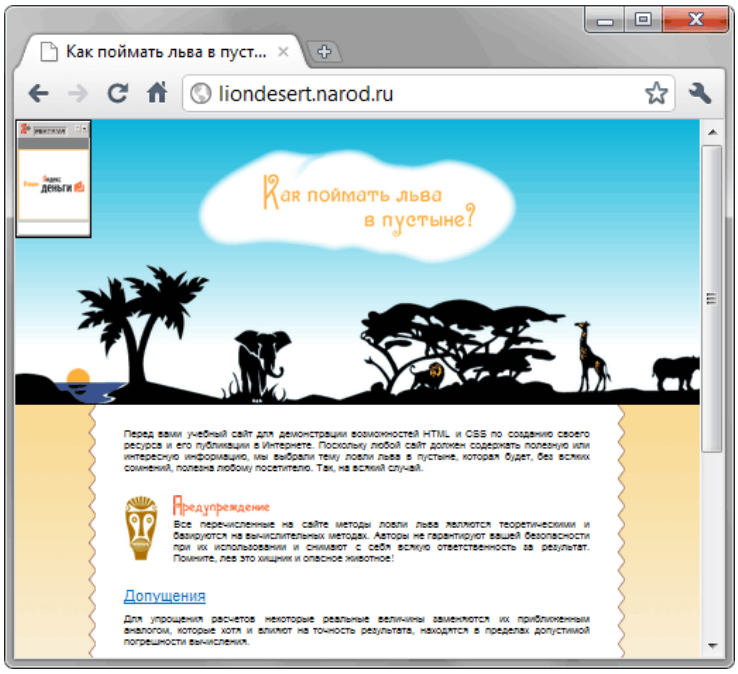


Рис. 8.18. Готовый сайт

Если вы хотите использовать для сайта CMS, кодировку UTF-8 и выводить только свои желаемые баннеры или не показывать их вообще, следует обратить внимание на платный хостинг, предоставляющий эти возможности. В частности, сайт http://lionindesert.ru являющийся копией сайта на Народе, расположен на платном хостинге. Его основное отличие в том, что он свёрстан на HTML5, о котором пойдёт речь в следующей главе.

Глава IX

Использование HTML5

Использование HTML5

Примеры в этой книге преимущественно были сделаны на XHTML, он хорошо подходит для обучения и вырабатывания правильной манеры вёрстки благодаря своим формальным правилам и более жёсткому, по сравнению с HTML синтаксису. Однако за десять лет, прошедших со дня выпуска, XHTML морально устарел и уже не соответствует современным условиям. В частности, нет штатных средств для проигрывания аудио и видеороликов, нет поддержки геолокации, возможности рисовать непосредственно в браузере, не хватает некоторых элементов форм и много другого. Конечно, часть этих проблем давно решается через сторонние плагины к браузеру, например, Adobe Flash воспроизводит видео, Google Gears реализует локальные базы данных и запуск скриптов в фоновом режиме. Язык программирования JavaScript позволяет реализовать недостающий функционал форм и различные эффекты на странице. Но все эти технологии имеют определённые ограничения — плагины нужно устанавливать дополнительно, при этом они могут не работать, как Flash на iPhone и iPad, далеко не всё умеет и JavaScript. Популярность мобильных устройств, развитие каналов связи переместило акцент веб-технологий на мультимедиа, т.е. воспроизведение потокового аудио и видео, а также соответствующих файлов. Ничего этого в XHTML нет.

W3 Consortium, разработчик спецификаций HTML и XHTML, начал работать над XHTML 2.0, в котором указанные недостатки предыдущей версии бы обходились. В результате этот проект оказался замороженным и не завершён. Обеспокоенные медленным ходом работ разработчики браузеров Safari, Firefox и Opera основали свою собственную организацию WHATWG (Web Hypertext Application Technology Working Group, Рабочая группа по разработке гипертекстовых приложений Интернета), которая подхватила упавшее знамя. Идеи W3C, современные потребности пользователей и мнение веб-разработчиков воплотилось в новом языке разметки названном HTML5.

Следует понимать, что, несмотря на схожесть названий, HTML5 не является преемником HTML4 или XHTML. Скорее речь идёт о новом языке Web Applications 1.0, который в маркетинговых целях назван знакомой аббревиатурой и построен на базе HTML.

Официально стандарт HTML5 ещё не завершён, но современные браузеры уже умеют частично с ним работать. Итак, что же интересного нам даёт HTML5? Вот некоторые его возможности.

- Поддержка геолокации — определение местоположения пользователя на карте и использование этой информации для вычисления маршрута его движения, вывода близлежащих магазинов, кинотеатров, кафе и других данных.
- Воспроизведение видеороликов.
- Воспроизведение аудиофайлов.
- Локальное хранилище — позволяет сайтам сохранять информацию на локальном компьютере и обращаться к ней позже.
- Фоновые вычисления — стандартный способ запуска JavaScript в браузере в фоновом режиме.
- Оффлайновые приложения — страницы, которые могут работать при отключении Интернета.
- Рисование — внутри тега **<canvas>** с помощью JavaScript можно рисовать фигуры, линии, создавать градиенты и трансформировать объекты на лету.
- Новые элементы форм: для даты, времени, поиска, чисел, выбора цвета и др.

Кроме этих возможностей в HTML5 включены новые теги для разметки документа, выброшены устаревшие теги и модифицированы некоторые другие. Для вёрстки веб-страниц в первую очередь необходимо понять, что поменялось и как перевести страницу с XHTML на HTML5.

Структура кода

Любой код разметки начинается с доктайпа, этот элемент говорит браузеру, на каком языке разметки и его версии написан документ. Также доктайп переводит браузер в один из возможных режимов: стандартный, почти стандартный и режим совместимости. В HTML5 нет подобных делений, доктайп один единственный и при его наличии браузер работает в стандартном режиме.

```
<!DOCTYPE html>
```

Из всех видов это самый короткий доктайп, его легко запомнить и набирать по памяти.

Изменения претерпели и другие теги, так, у тега **<html>** нет атрибута xmlns, а кодировка документа сократилась до такой записи.

```
<meta charset="utf-8" />
```

Впрочем, старый способ указать кодировку также остался.

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

Атрибут type у тега **<script>** и **<style>** можно опустить, браузер автоматически понимает содержимое этих тегов и ему уже не требуется явно об этом напоминать. Простейший код приведён в примере 9.1.

Пример 9.1. Код на HTML5

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>Пример страницы</title>
  <style>
  p { color:  navy; }
  </style>
 </head>
 <body>
  <p>Страница на HTML5</p>
 </body>
</html>
```

Синтаксис HTML5

HTML5 одновременно поддерживает два разных синтаксиса: HTML и XHTML, соответственно, имеется две манеры вёрстки. В случае использования синтаксиса HTML разрешается писать теги заглавными буквами, не закрывать некоторые теги, атрибуты писать без кавычек. При соблюдении синтаксиса XHTML следует придерживаться следующих правил.

- Все теги и их атрибуты должны быть набраны в нижнем регистре (строчными символами).
- Значения любых атрибутов необходимо заключать в кавычки.
- Требуется закрывать все теги, даже такие, которым не сопоставлен закрывающий тег.
- Нельзя использовать сокращенные атрибуты тегов.

Чтобы браузер различал, в каком виде ему отдаётся документ, сервер сообщает ему MIME-тип файла с расширением .html. Для сериализации HTML применяется тип text/html, а для сериализации XHTML, которая иногда называется XHTML5, тип application/xhtml+xml. Если у вас установлен веб-сервер Apache, то вы можете указать тип через директиву AddType, добавив следующую строку в файл .htaccess, расположенный в корне сайта.

```
AddType application/xhtml+xml .html
```

В данном случае мы говорим, что все файлы с расширением .html отдавать как application/xhtml+xml. Если документы формируются через PHP, то можно отдавать заголовок следующим образом:

```
header ("Content-type: application/xhtml+xml");
```

Учтите, что эта строка должна идти до вывода любого текста на странице.

Выбор сериализации зависит от ряда условий. Перечислим их основные преимущества.

HTML5

- Обратная совместимость с существующими браузерами.
- Снисходительный синтаксис позволяет некоторые ошибки и отображает страницу даже при их наличии.
- Допускает вольную манеру написания кода — некоторые теги можно не закрывать, значения указывать не всегда, кавычками в значения не пользоваться.

XHTML5

- Строгий синтаксис поощряет авторов писать правильный код.
- Напрямую интегрируется с другими XML-технологиями вроде SVG и MathML.
- Используется с серверным обработчиком XML для хранения и представления данных в этом формате.

В действительности, application/xhtml+xml на практике используется достаточно редко, в первую очередь из-за проблем с совместимостью браузеров, поэтому тип text/html применяется повсеместно. В этом случае выбор синтаксиса не играет особой роли, поэтому вы вообще можете писать нечто вроде этого:

```
<IMG  src= 1.jpg alt="" width="160" height=50 />
```

Отсутствие кавычек у некоторых атрибутов и большие буквы в теге говорят, что это HTML, но закрывающий тег взят из XHTML. Такая форма записи в HTML5 абсолютно корректна и проходит валидацию. Но это пример ужасного кода и плохой стиль вёрстки, поэтому его лучше избегать.

Так как большая часть примеров у нас написана на XHTML, то логично использовать именно этот язык как основу для переделки работ на HTML5. Таким образом, HTML-файл будет отправляться как text/html, но стиль кода у нас будет XHTML.

Новые теги

В HTML5 для структуры кода введено несколько новых тегов: **<article>**, **<aside>**, **<footer>**, **<header>**, **<nav>**, которые заменяют в некоторых случаях привычный ** **. Хотя кажется, что особой разницы между тегами ** ** и **<header>** нет, между ними лежит огромная пропасть. Теги ориентированы не на людей, которым нет смысла заглядывать в исходный код страницы, а на машины, интерпретирующих код. Машины или роботы не понимают ** **, для них это типовой тег разметки — замени его на ** ** и смысл не поменяется. Другое дело **<header>**, робот, обнаружив этот тег, воспринимает его именно как шапку сайта или раздела.

Что это даёт в итоге? Поисковые системы начинают лучше индексировать сайт, потому что чётко отделяют контент страницы от вспомогательных элементов. Речевые браузеры, предназначенные для слепых людей, пропускают заголовок и переходят непосредственно к содержимому. Сайты могут автоматически обмениваться контентом и другой информацией между собой. Все эти возможности называются семантикой и позволяют представить данные в удобном для роботов виде.

У нас уже есть сайт на XHTML — lionindesert.ru, поэтому будет довольно просто перевести его на HTML5. Так, шапка сайта поменяется незначительно (пример 9.2).

Пример 9.2. Использование <header>

```
<header>
 <div class="header-bg">
  <img src="images/header-title.png" alt="Как  поймать льва в пустыне" />
 </div>
</header>
```

Попытка добавить в стилях фон к тегу **<header>** ни к чему не привела, фон отображаться не желает. Все новые теги следует сделать вначале блочными через свойство display, тогда они начнут корректно выводиться в браузере. В примере 9.3 показан код для создания шапки сайта.

Пример 9.3. Шапка сайта

HTML 5.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>Как поймать льва в пустыне?</title>
  <style>
   body { margin: 0; }
   header {
    display: block;
    background: #00B0D8 url(images/header-gradient.png) repeat-x;
   }
   .header-bg {
    background:  url(images/header-animal.png) repeat-x center bottom;
    height: 405px;
    text-align: center;
   }
   header img {
    position: relative;
    top: 40px;
   }
  </style>
 </head>
 <body>
  <header>
   <div class="header-bg">
    <img src="images/header-title.png" alt="Как поймать льва в пустыне" />
   </div>
  </header>
 </body>
</html>
```

Данный пример будет работать во всех браузерах, кроме IE7 и IE8. Internet Explorer не добавляет стиль

к элементам, которые не понимает. Это недоразумение можно исправить, если создать фиктивный элемент с помощью JavaScript. Для этого включим в **<head>** такой код.

```
<script>
  document.createElement("header");
</script>
```

Если на странице встречается один тег, этот скрипт вполне подойдёт для работы. Но не хочется повторять строку десять раз для десяти разных тегов, поэтому автоматизируем этот процесс через цикл. Сами теги указываются списком, разделяясь запятой (пример 9.4).

Пример 9.4. Скрипт для IE

```
<!--[if lt IE 9]>
 <script>
  var e =
("article,aside,figcaption,figure,footer,header,hgroup,nav,section,time").split
);
  for (var i = 0; i < e.length; i++) {
    document.createElement(e[i]);
  }
 </script>
<![endif]-->
```

Сам скрипт заключается в условные комментарии, чтобы выполнялся только для IE версии 8.0 и ниже. В IE9 поддержка новых тегов HTML5 уже включена.

Пример выше не обязательно вставлять к себе на сайт, можно воспользоваться общедоступным скриптом написанным Реми Шарпом и распространяемым по лицензии MIT. Для этого достаточно указать на него ссылку, как показано в примере 9.5.

Пример 9.5. Скрипт для IE

```
<!--[if lt IE 9]>
 <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

Все указанные скрипты должны располагаться в коде перед CSS.

Таким образом, для полноценного использования тегов HTML5 во всех браузерах достаточно выполнить три условия:

1. установить доктайп **<!DOCTYPE html>**;
2. включить скрипт из примера 9.4 или 9.5;
3. в стилях для новых тегов установить display: block.

Теперь рассмотрим некоторые теги HTML5 более подробно, чтобы понять область их применения.

<article>

Задаёт содержание сайта вроде новости, статьи, записи блога, форума или др. В примере 9.6 показано добавление двух тегов **<article>**.

Пример 9.6. Использование тега <article>

HTML 5.0 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>article</title>
 </head>
 <body>
  <header><h1>Следы невиданных зверей</h1></header>
  <article>
    История о том, как возле столовой появились загадочные розовые
    следы с шестью пальцами, и почему это случилось.
  </article>
 </body>
```

```
</html>
```

<aside>

Определяет блок который не относится к основному контенту для размещения рубрик, ссылок на архив, меток и другой информации (пример 9.7). Такой блок, если он располагается сбоку, называется, как правило, «сайдбар» или «боковая панель».

Пример 9.7. Использование <aside>

HTML 5 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>aside</title>
  <script>
   document.createElement('aside');
   document.createElement('article');
  </script>
  <style>
   aside {
    background: #f0f0f0; /* Цвет фона */
    padding: 10px; /* Поля */
    width: 200px; /* Ширина сайдбара */
    float: right; /* Обтекание слева */
   }
   article {
    margin-right: 240px; /* Отступ справа */
    display: block; /* Блочный элемент */
   }
  </style>
 </head>
 <body>
  <aside>
   <p>Экономьте электричество</p>
   <p>Хороший язык</p>
   <p>Чья палка больше</p>
  </aside>
  <article>
   История о том, как приходилось экономить электричество,
   какие меры для этого принимались, и куда оно на самом деле уходило.
  </article>
 </body>
</html>
```

<figure>

Используется для группирования любых элементов, например, изображений и подписей к ним (пример 9.8).

Пример 9.8. Использование <figure>

HTML 5 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>figure</title>
  <script>
   document.createElement('figure');
   document.createElement('figcaption');
  </script>
  <style>
   figure {
    background: #5f6a72; /* Цвет фона */
    padding: 10px; /* Поля вокруг */
    display: block; /* Блочный элемент */
    width: 150px; /* Ширина */
    float: left; /* Блоки выстраиваются по горизонтали */
    margin: 0 10px 10px 0; /* Отступы */
    text-align: center; /* Выравнивание по центру */
   }
   figcaption {
    color: #fff; /* Цвет текста */
   }
  </style>
 </head>
 <body>
  <article>
   <figure>
    <p><img src="images/thumb1.jpg" alt="" /></p>
```

```
    <figcaption>Софийский собор</figcaption>
   </figure>
   <figure>
    <p><img src="images/thumb2.jpg" alt="" /></p>
    <figcaption>Польский костёл</figcaption>
   </figure>
  </article>
 </body>
</html>
```

<figcaption>

Содержит описание для тега **<figure>**. Тег **<figcaption>** должен быть первым или последним элементом в группе.

<footer>

Задает «подвал» сайта или раздела, в нем обычно располагается имя автора, дата документа, контактная и правовая информация (пример 9.9).

Пример 9.9. Использование <footer>

HTML 5 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html>
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>footer</title>
 </head>
 <body>
  <header>
    <h1>Персональный сайт Кристины Ветровой</h1>
  </header>
  <article>
    <h2>Добро пожаловать!</h2>
    <p>Рада приветствовать вас на своем сайте.</p>
  </article>
  <footer>
    Copyright Кристина Ветрова
  </footer>
 </body>
</html>
```

<header>

Определяет «шапку» сайта или раздела.

<hgroup>

Используется для группирования заголовков веб-страницы или раздела (пример 9.10).

Пример 9.10. Использование <hgroup>

HTML 5 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>hgroup</title>
 </head>
 <body>
  <hgroup>
    <h1>Кристина Ветрова</h1>
    <h2>Персональный сайт</h2>
  </hgroup>
 </body>
</html>
```

<nav>

Задаёт навигацию по сайту (пример 9.11). Если на странице несколько блоков ссылок, то в **<nav>** обычно помещают приоритетные ссылки. Также допустимо использовать несколько тегов **<nav>** в документе. Запрещается вкладывать **<nav>** внутрь **<address>**.

Пример 9.11. Использование <nav>

HTML 5 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>nav</title>
 </head>
 <body>
  <header>
    <h1>Чебурашка и крокодил Гена</h1>
  </header>
  <nav><a href="1.html">Чебурашка</a> | <a href="2.html">Гена</a> |
       <a href="3.html">Шапокляк</a> | <a href="4.html">Лариска</a></nav>
  <article>
    <h2>Добро пожаловать!</h2>
  </article>
 </body>
</html>
```

<section>

Определяет раздел документа, который может включать в себя заголовки, шапку, подвал и текст (пример 9.12). Допускается вкладывать один тег **<section>** внутрь другого.

Пример 9.12. Использование <section>

HTML 5 | IE 7 | IE 8 | IE 9 | Cr 5 | Op 10 | Sa 5 | Fx 3.6

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>section</title>
 </head>
 <body>
  <section>
   <h2>Съёмки фильма Полипропилен</h2>
   <p>История о том, как снимали фильм, где герои отдыхали на пляже,
   потом пришёл антагонист, избил протагонистов, сбросил их в бассейн,
   и что из этого получилось.</p>
  </section>
  <section>
   <h2>Хороший язык</h2>
   <p>История о том, как проходила студия изучения языка эсперанто,
   в то время, как над ней, на веранде велась студия приколистов,
   где травились анекдоты, и что из этого получилось.</p>
  </section>
 </body>
</html>
```

<time>

Помечает текст внутри тега **<time>** как дата, время или одновременно дата и время. Может указываться непосредственно внутри контейнера **<time>**, либо задаваться через атрибут datetime (пример 9.13).

Дата и время задается в международном формате ISO 8601. Примеры оформления приведены в табл. 9.1.

Табл. 9.1. Форматы даты и времени

Значение	Формат	Пример
Год	ГГГГ	2012
Месяц и год	ГГГГ-ММ	2012-12
Полная дата	ГГГГ-ММ-ДД	2012-12-23
Дата и время с минутами	ГГГГ-ММ-ДДТчч:мм	2004-07-24T18:18
Дата и время с секундами	ГГГГ-ММ-ДДТчч:мм:сс	2004-07-24T18:18:18
Дата и время с часовым поясом	ГГГГ-ММ-ДДТчч:мм:сс±чч:мм	2004-07-24T18:18:18+04:00

Для каждой единицы существует своя заданная форма и ограничения.

- Год — задается четырьмя цифрами (1860).
- Месяц — две цифры (01 — январь, 02 — февраль, 12 — декабрь).
- День — две цифры от 01 до 31.
- Час — две цифры от 00 до 23.
- Минуты — две цифры от 00 до 59.
- Секунды — две цифры от 00 до 59.
- Часовой пояс — часы и минуты с указанием знака плюс или минус.

Дата и время разделяются между собой заглавной латинской буквой T. Часовой пояс при необходимости пишется после времени со знаком плюс или минус. К примеру, для Москвы часовой пояс будет +03:00.

Пример 9.13. Использование <time>

HTML 5 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>time</title>
 </head>
 <body>
  <article>
   <p>Опубликовано:
      <time datetime="2012-12-23T08:23:11+07:00">сегодня</time></p>
   <p><time>1957-10-04</time> запущен первый искусственный спутник Земли.</p>
   <p><time>1960-08-19</time> первый полёт собак в космос.</p>
   <p><time>1961-04-12</time> первый полёт человека в космос.</p>
   <p><time>1963-06-16</time> первый полёет женщины-космонавта.</p>
   <p><time>1969-07-21</time> высадка человека на Луну.</p></article>
 </body>
</html>
```

Что ещё можно сделать, чтобы наш код стал ближе к HTML5? В статьях для иллюстрации материала используются рисунки с подписью, для них можно указать комбинацию тегов **<figure>** и **<figcaption>**.

```
<figure>
 <p><img src="images/fig2-1.png" alt="Рис. 1" height="252" width="381" /></p>
 <figcaption>Рис. 1. Разбиение пустыни на N элементарных
             прямоугольников</figcaption>
</figure>
```

Здесь тег **<figure>** включает в себя сам рисунок и подпись к нему в виде тега **<figcaption>**, для них включим такой стиль.

```
figure {
  text-align: center; /* Выравнивание по центру */
}
figcaption {
 font: italic 1em Georgia, "Times New Roman", Times, serif;
       /* Параметры шрифта */
 display: block; /* Блочный элемент */
 margin: 0 0 1.5em; /* Отступы */
}
```

Применение HTML5 на практике

Теперь, после знакомства с тегами HTML5, продолжим изменять наш сайт lionindesert.ru с XHTML на HTML5. Верхняя часть для всех страниц будет идентичной, за исключением тега **<title>**, он определяет заголовок страницы. В примере 9.14 показано содержимое тега **<head>**.

Пример 9.14. Тег <head>

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8"  />
  <title>Как поймать льва в пустыне?</title>
  <link rel="stylesheet" href="style.css" />
  <!--[if lt IE 9]>
  <script  src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
  <![endif]-->
 </head>
```

Здесь применяется сокращённая форма тега **<meta>** для указания кодировки и **<link>** для ссылки на стилевой файл. Для браузера IE до версии 8.0 включительно работает скрипт, чтобы IE распознавал новые теги HTML5.

Заголовок сайта с тегом **<header>** был показан в примере 9.2, поэтому перейдём непосредственно к основной части главной странице. На ней располагаются ссылки на страницы с их описанием. Оформим их с помощью тегов **<section>** и **<header>** (пример 9.15).

Пример 9.15. Ссылки в основной части

```
<section>
  <header><h1><a href="assumption.html">Допущения</a></h1></header>
  <p>Для упрощения расчетов некоторые реальные величины заменяются
     их приближенным аналогом, которые хотя и влияют на  точность
     результата, находятся в пределах допустимой погрешности вычисления.</p>
</section>
<section>
  <header><h1><a href="simple-iteration.html">Метод
          простых итераций</a></h1></header>
  <p>Самый простой метод поиска льва основанный на переборе.</p>
</section>
```

Каждый блок со ссылкой и её описанием обрамлён тегом **<section>**, сама ссылка располагается внутри тега **<h1>**, который в свою очередь находится внутри **<header>**. Такая логика может показаться странной — зачем нам **<header>**, когда на странице он уже есть, и к чему столько тегов **<h1>**? В HTML5 своя логика построения структуры документа, которая отличается от привычной схемы HTML4. В предыдущей версии HTML иерархия блоков строилась на основе тегов **<h1>**...**<h6>**. Соответственно, **<h1>** задавал заголовок страницы, **<h2>** и **<h3>** подзаголовки. Чтобы схема документа строилась правильно, на странице должен быть только один **<h1>**. Спецификация HTML5 устанавливает алгоритм генерации схемы документа, включающий в себя новые семантические теги. Этот алгоритм говорит, что теги **<article>** и **<section>** создают новый раздел. А в HTML5 каждый раздел может содержать собственный тег **<h1>**.

Проверить схему построения документа HTML можно с помощью сервиса HTML5 Outliner. Вы загружаете HTML-файл для проверки, либо указываете его адрес. Результат проверки показан на рис. 9.1.

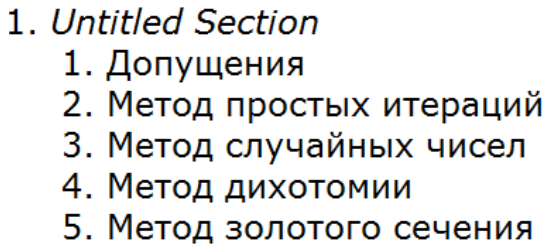


Рис. 9.1. Схема документа

Под «Untitled Section» подразумевается первый **<header>**, в котором находится заголовок сайта. В нём нет текста, только картинка с названием сайта, поэтому раздел озаглавлен «неназванным». Остальные разделы получены от тегов **<h1>** расположенных внутри **<section>**.

Переходим к подвалу, здесь аналогично шапке, мы вставляем тег **<footer>**, как показано в примере 9.16.

Пример 9.16. Подвал страницы

```
<footer>
 <div class="lion"><img src="images/lion.png" alt=""
                    width="130" height="80" /></div>
 <div class="footer-bg">
  <div class="copyright">
   <p><strong>Учебный сайт «Как поймать льва в пустыне»</strong></p>
   <p>&copy; Влад Мержевич</p>
  </div>
 </div>
</footer>
```

Теперь собираем вместе все наши разделы и получаем окончательный код главной страницы сайта (пример 9.17).

Пример 9.17. Главная страница

HTML 5 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>Как поймать льва в пустыне?</title>
  <link rel="stylesheet" href="style.css" />
  <!--[if lt IE 9]>
   <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
  <![endif]-->
 </head>
 <body>
  <header>
   <div class="header-bg">
    <img src="images/header-title.png" alt="Как поймать льва в пустыне"
                width="456" height="166"  />
   </div>
  </header>
  <div class="content-gradient">
   <div class="content-bg">
    <div class="content-white">
     <p>Перед вами учебный сайт для демонстрации возможностей HTML и CSS по
        созданию своего ресурса и его публикации в Интернете. Поскольку
        любой сайт должен содержать полезную или интересную информацию,
        мы выбрали тему ловли льва в пустыне, которая будет, без всяких
        сомнений, полезна любому посетителю. Так, на всякий случай.</p>
     <section class="warning">
      <header><h1><img src="images/warning-title.png" alt="Предупреждение"
       width="137" height="30" /></h1></header>
      <p>Все перечисленные на сайте методы ловли льва являются
        теоретическими и базируются на вычислительных методах. Авторы не
        гарантируют вашей безопасности при их использовании и снимают с
        себя всякую ответственность за результат. Помните, лев это хищник
        и опасное животное!</p>
     </section>
     <section>
      <header><h1><a href="assumption.html">Допущения</a></h1></header>
      <p>Для упрощения расчетов некоторые реальные величины заменяются их
        приближенным аналогом, которые хотя и влияют на точность результата,
        находятся в пределах допустимой погрешности вычисления.</p>
     </section>
```

```
    </div>
   </div>
  </div>
  <footer>
  <div class="lion"><img src="images/lion.png" alt="" width="130"
                    height="80" /></div>
   <div class="footer-bg">
    <div class="copyright">
     <p><strong>Учебный сайт «Как поймать льва в пустыне»</strong></p>
     <p>&copy; Влад Мержевич</p>
    </div>
   </div>
  </footer>
 </body>
</html>
```

Код HTML завершён, но страница находится в совершенно непотребном виде из-за введения новых тегов. Поэтому необходимо также отредактировать стилевой файл style.css и внести в него необходимые изменения.

Для начала необходимо указать, что теги **<header>**, **<section>**, **<footer>**, **<aside>**, **<nav>**, **<article>** являются блочными. Это необходимо для корректного применения к ним стилей.

```
header, section, footer, aside, nav, article {
 display: block; /* Блочный элемент */
}
```

Класс `header` меняем на селектор **header**, а класс `footer` на **footer**.

```
header {
 background: #00B0D8 url(images/header-gradient.png) repeat-x; /* Градиент */
}
footer {
 background: url(images/grass.png) 50% 53px no-repeat; /* Фоновый рисунок */
 margin-top: -77px; /* Поднимаем вверх */
 overflow: auto; /* Отменяем схлопывающиеся отступы */
 position: relative; /* Относительное позиционирование */
}
```

На этом основные разделы сайта приобретают задуманный вид. Неприятность поджидает с тегом **<section>** (рис. 9.2).

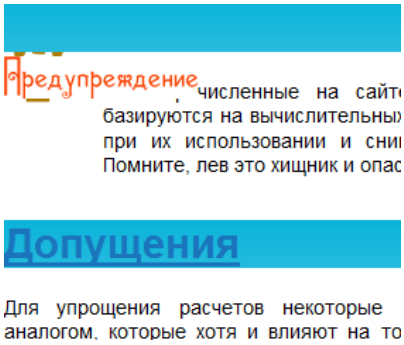


Рис. 9.2. Вид заголовка раздела

Тег **<header>** используется внутри **<section>** и для него работает тот же стиль, что и для шапки страницы, поэтому виден тот же голубой фон. Для начала необходимо избавиться от фона, не затрагивая при этом шапку. Так как **<header>** родительским элементом выступает **<section>**, то добавим такой стиль.

```
section header {
 background: none; /* Убираем фон */
}
```

Фон исчезает, но картинка с надписью «Предупреждение» сдвинута вниз. На неё действует селектор **header img**, поэтому для изображения header-title.png требуется ввести отдельный класс и применять стиль для него, либо изменить контекст применения стиля. Для этого меняем селектор на **.header-bg**

img.

```
.header-bg img {
 position: relative; /* Относительное позиционирование */
 top: 40px; /* Сдвигаем картинку вниз */
}
```

Остаётся только изменить размер текста **<h1>** в заголовке.

```
section h1 {
 font-size: 1.5em; /* Размер текста */
 font-weight: normal; /* Нормальное начертание */
 margin-bottom: 0; /* Отступ снизу */
}
```

Переходим от главной странице к остальным. Они содержат контент слева, сайдбар справа и навигацию в нём. Тег ** ** меняем на **<aside>**, а в стилях класс `sidebar` на селектор **aside**.

```
aside {
 width: 200px; /* Ширина правой колонки  */
 float: right; /* Обтекание */
}
```

Аналогичное действие проделываем с контентом, ** ** заменяем на **<article>**, а в стилях класс `content` на селектор **article**.

```
article {
 margin-right: 240px; /* Отступ справа */
}
```

Теперь навигация. Она сделана в виде списка (тег **<ul>**), а нам нужен тег **<nav>**. Поэтому используем такое вложение.

```
<nav>
 <ul class="menu">
  <li><a href="/">Главная страница</a></li>
  <li class="current"><span>Допущения</span></li>
  <li><a href="simple-iteration.html">Метод простых итераций</a></li>
  <li><a href="random-number.html">Метод случайных чисел</a></li>
  <li><a href="dixotomia.html">Метод дихотомии</a></li>
  <li><a href="golden-section.html">Метод золотого сечения</a></li>
 </ul>
</nav>
```

В этом случае соблюдается семантика кода и не придётся жертвовать нашим уже настроенным списком.

Окончательно основная часть внутренней страницы представлена в примере 9.18.

Пример 9.18. Внутренняя страница

```
<div class="content-gradient">
 <div class="content-bg">
  <div class="content-white">
   <aside>
    <nav>
     <ul class="menu">
      <li><a href="/">Главная страница</a></li>
      <li class="current"><span>Допущения</span></li>
      <li><a href="simple-iteration.html">Метод простых итераций</a></li>
      <li><a href="random-number.html">Метод случайных чисел</a></li>
      <li><a href="dixotomia.html">Метод дихотомии</a></li>
      <li><a href="golden-section.html">Метод золотого сечения</a></li>
     </ul>
    </nav>
    <div class="interest">
     <h3><img src="images/interest-title.png" alt="Интересная информация"
          width="164" height="30" /></h3>
     <p>Средний самец льва имеет длину около трех метров и весит от 180 до
        230 килограмм.</p>
     <p>Львы питаются не только убитыми животными, они также не брезгуют
        падалью.</p>
    </div>
```

```
    </aside>
    <article>
     <h1>Допущения</h1>
     <p>Для упрощения расчетов некоторые реальные величины заменяются их
      приближенным аналогом, которые хотя и влияют на точность результата,
      находятся в пределах допустимой погрешности вычисления. </p>
    </article>
   </div>
  </div>
</div>
```

Что ещё можно сделать, чтобы наш код стал ближе к HTML5? В статьях для иллюстрации материала используются рисунки с подписью, для них можно указать комбинацию тегов **<figure>** и **<figcaption>**.

```
<figure>
 <p><img src="images/fig2-1.png" alt="Рис. 1" height="252"
     width="381" /></p>
 <figcaption>Рис. 1. Разбиение пустыни на N элементарных
             прямоугольников</figcaption>
</figure>
```

Здесь тег **<figure>** включает в себя сам рисунок и подпись к нему в виде тега **<figcaption>**, для них включим такой стиль.

```
figure {
 text-align: center; /* Выравнивание по центру */
}
figcaption {
 font: italic 1em Georgia, "Times New Roman", Times, serif; /* Шрифт */
 display: block; /* Блочный элемент */
 margin: 0 0 1.5em; /* Отступы */
}
```

На этом перевод сайта на HTML5 завершён, результат можно посмотреть по адресу lionindesert.ru.

Валидация HTML5

Чтобы убедиться в том, что код набран правильно и не содержит неуклюжие опечатки, его следует проверить с помощью валидатора — так называется программа или сервис для проверки документа на соответствие веб-стандартам и выявления существующих ошибок. Соответственно, валидным является такой веб-документ, который прошел подобную процедуру и не имеет замечаний по коду. Хотя HTML5 ещё находится в процессе развития, возможность валидации предоставляет сервис validator.w3.org и validator.nu.

validator.w3.org

По адресу http://validator.w3.org располагается, пожалуй, самый распространенный инструмент для проверки отдельных страниц на валидность. Этот сайт предлагает три способа проверки: по адресу, локального файла и введенного в форму кода.

Проверка по адресу

Если ваш сайт уже опубликован в Интернете, то любую страницу можно проверить, вводя в текстовое поле её адрес (рис. 9.3).

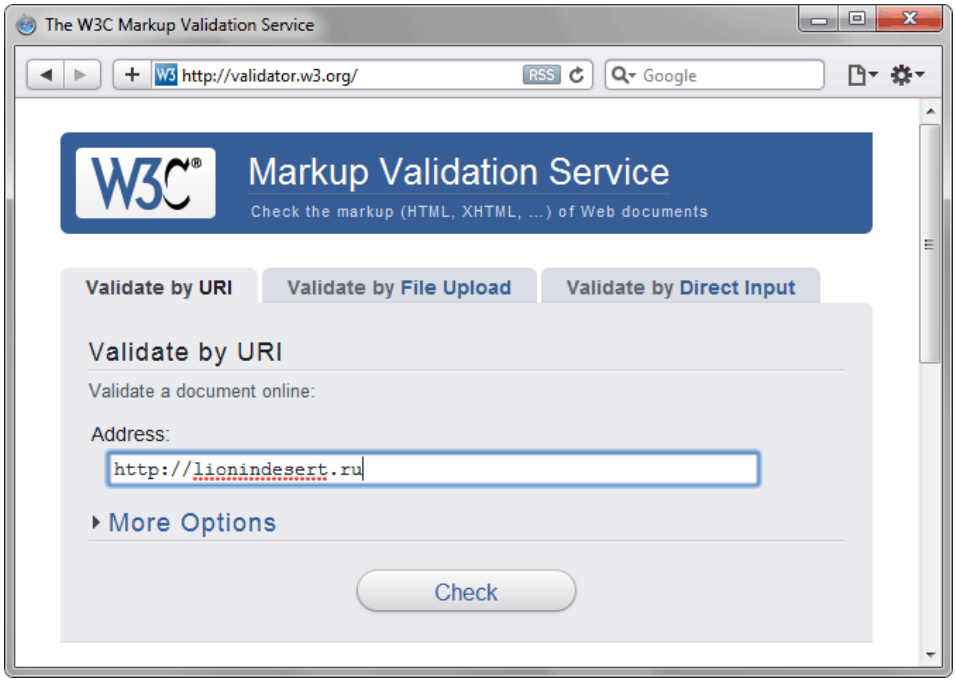


Рис. 9.3. Форма для ввода адреса документа

Так, вводя http://lionindesert.ru в форме «Validate by URI» (валидация по адресу) и нажав кнопку `Check` (проверить) получим сообщение о том, валидный документ или нет.

Хотя в текстовом поле вводится адрес сайта, проверяется не сайт целиком, а только одна главная страница. Учтите, что, к примеру, адрес http://lionindesert.ru равнозначен вводу http://lionindesert.ru/index.html.

Валидатор проверяет HTML-код страницы и в случае отсутствия ошибок докладывает о валидности документа (рис. 9.4).

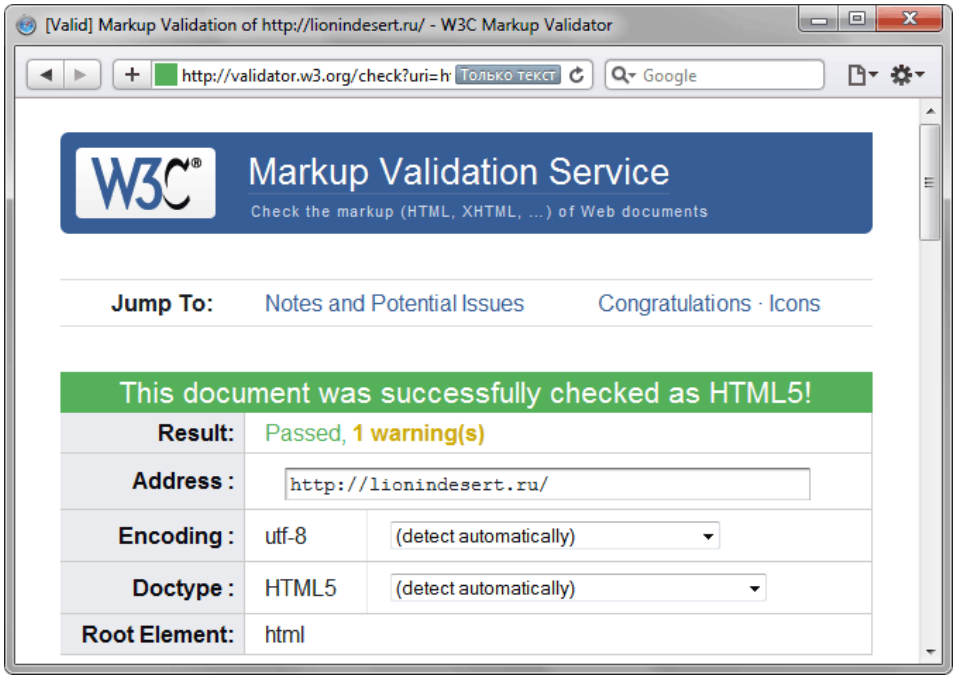


Рис. 9.4. Отчёт о проверке и валидности веб-страницы

Определение HTML5 происходит автоматически на основе доктайпа, при этом выводится предупреждение, что проверка на соответствие HTML5 находится в экспериментальной стадии.

При обнаружении ошибок выводится уведомление о том, что страница не валидна и список ошибок с указанием строк, где встречаются ошибки (рис. 9.5).

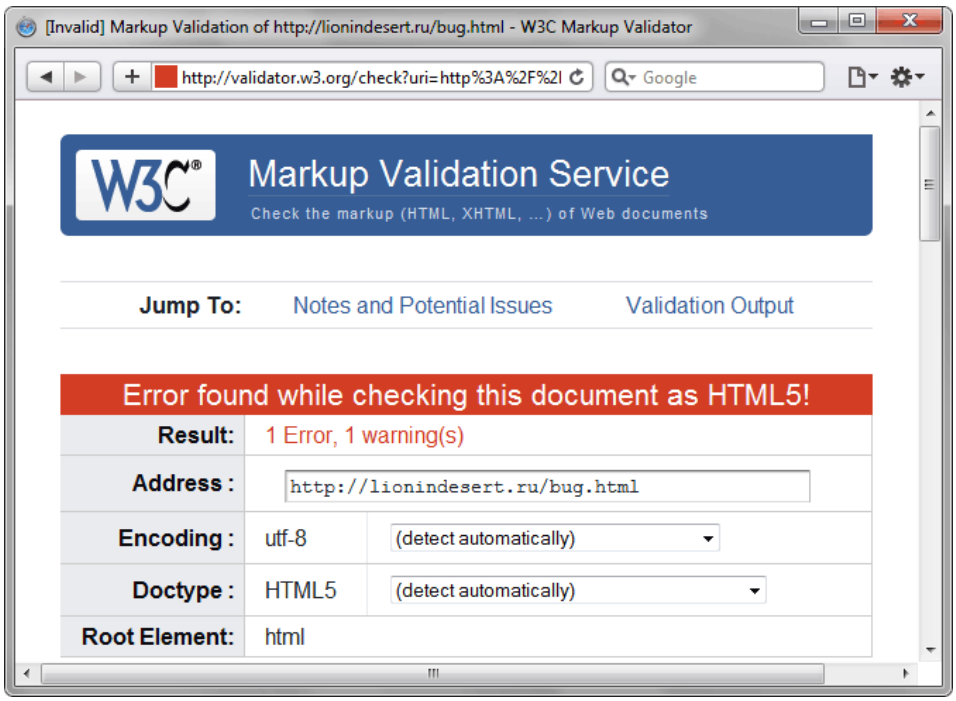


Рис. 9.5. Отчет о проверке и вывод ошибок

Проверка локальных файлов

Документы, еще не выставленные в Интернете, можно проверить с помощью формы, озаглавленной «Validate by File Upload» (валидация загруженных файлов), как показано на рис. 9.6.

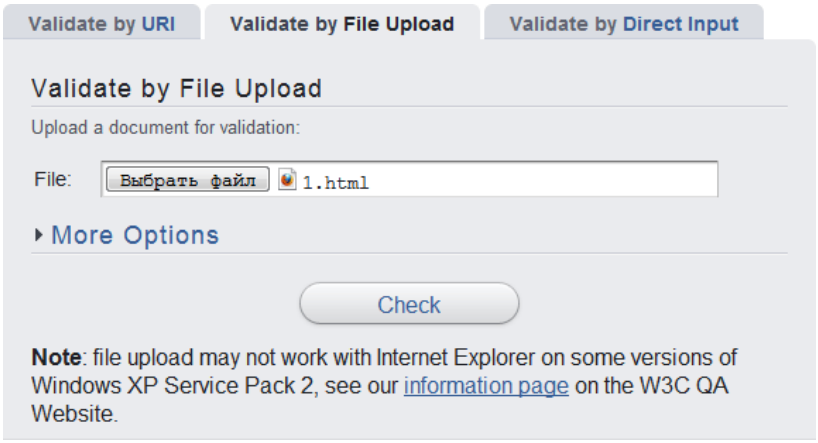


Рис. 9.6. Форма ввода пути к локальному файлу для его проверки

Вначале следует указать путь к HTML-файлу, после чего нажать кнопку `Check`. Файл будет загружен на сервер и проверен на ошибки.

Использование формы для ввода кода

В некоторых случаях требуется проверить код без сохранения его в отдельный файл. В этом случае пригодится форма для прямого набора текста и отправки его на сервер для валидации (рис. 9.7).

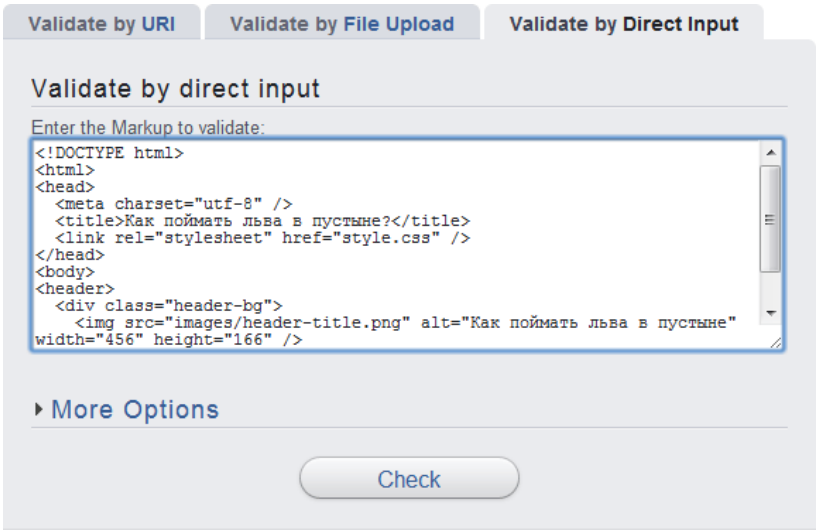


Рис. 9.7. Форма для ввода HTML-кода

html5.validator.ru

Разработанный Анри Сивоненом этот простой со спартанским интерфейсом валидатор позволяет проверять страницы на HTML5, а также на XHTML5, XHTML, HTML и некоторые экспериментальные функции. Валидатор работает в трёх режимах: проверка по адресу документа (Address), загрузка его на сервер (File Upload) и ввод кода непосредственно в текстовое поле (Text Field). Переключение между режимами происходит с помощью списка (рис. 9.8).

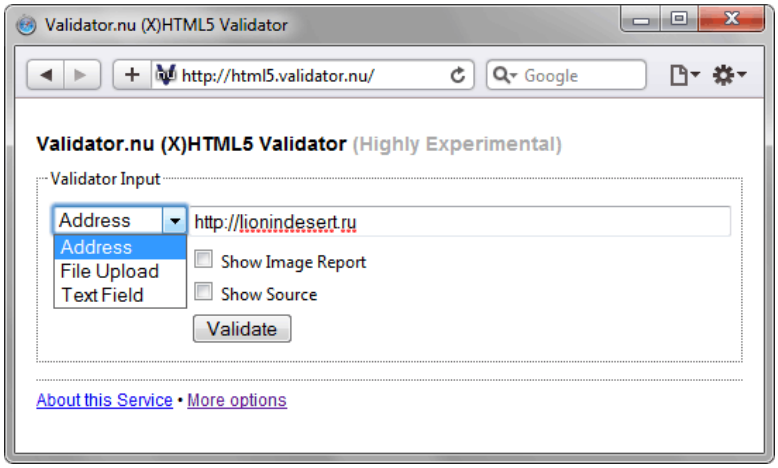


Рис. 9.8. Проверка по адресу документа

После нажатия на кнопку `Validate` выводится результат проверки. Если ошибок нет, демонстрируется зелёная рамка с надписью, что документ соответствует стандартам (рис. 9.9).

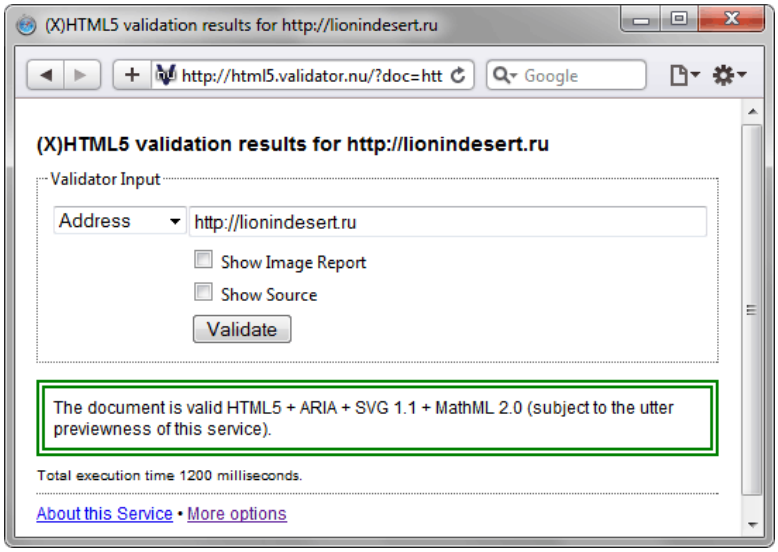


Рис. 9.9. Результат проверки

В противном случае отображается красная рамка, а также список ошибок.

На странице расположено две ссылки: About this Service (об этом сервисе), где подробно расписаны разные возможности сайта и More options (дополнительные опции). При нажатии на эту ссылку интерфейс меняется и дополнительно можно указать разные настройки валидации (рис. 9.10). Они преимущественно применяются для проверки других языков разметки чем HTML5.

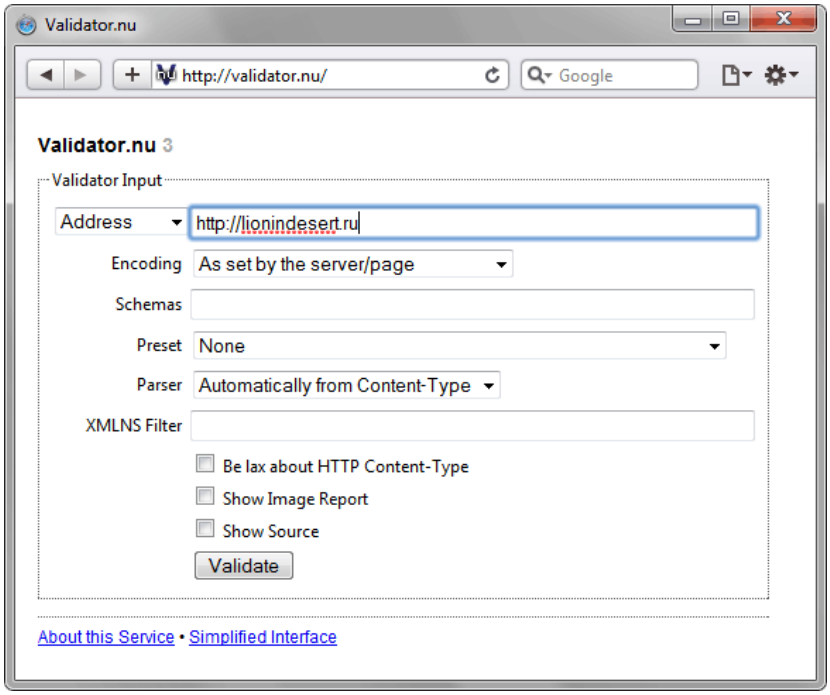


Рис. 9.10. Расширенный интерфейс

Глава X

Тестирование и отладка готового кода

Тестирование и отладка готового кода

В процессе работы над сайтом верстальщик должен принимать во внимание множество факторов, которые влияют на вид документа. У посетителей различается не только операционная система и браузер, но и такие параметры как количество цветов на мониторе, его разрешение, отключен или нет показ картинок, доступен ли JavaScript и др. После окончания вёрстки следует провести ряд проверок и в случае обнаружения явных ошибок, внести в код соответствующие изменения. Разумеется, это удобнее делать с помощью специализированных программ, наиболее удобной для этих целей является Web Developer. Этот набор инструментов выполнен в виде расширения Firefox и Chrome.

Web Developer

Хотя расширение имеется для двух разных браузеров, версия под Firefox находится в актуальном состоянии и регулярно обновляется, чего не скажешь о версии под Chrome. К тому же поддерживает русский язык. Поэтому в дальнейшем описание будет идти только для Firefox.

Установка происходит довольно просто, достаточно зайти браузером Firefox на сайт автора и нажать «Download», после чего вы будете перемещены на сайт, где надо щёлкнуть по «Добавить в Firefox» (рис. 10.1).

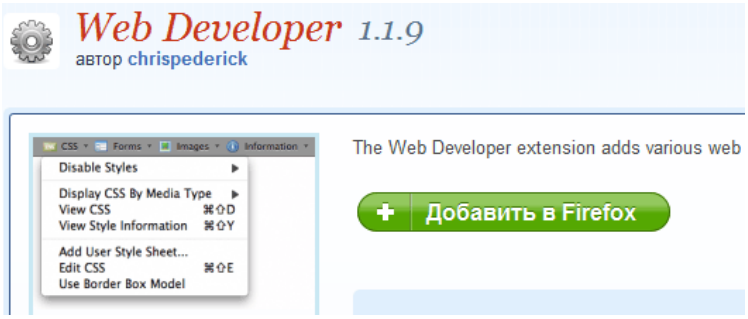


Рис. 10.1. Добавление в Firefox

Также можно зайти напрямую на страницу с расширением.

https://addons.mozilla.org/ru/firefox/addon/web-developer/

После добавления появится окно с предупреждением (рис. 10.2), жмём кнопку «Установить сейчас» и процедура установки закончена.

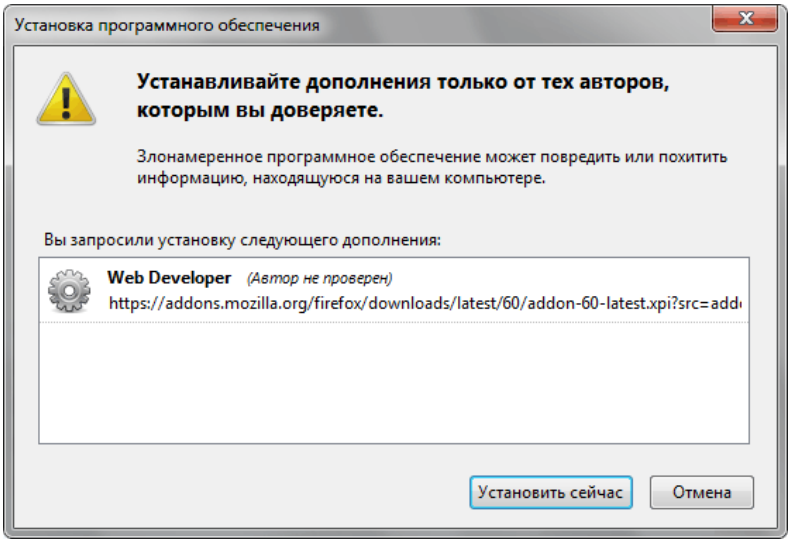


Рис. 10.2. Установка Web Developer

Ещё одним способом установки разных расширений является использование меню Инструменты > Дополнения. В окне поиска набираем «web developer», жмём Enter, в списке ниже появится нужное нам дополнение (рис. 10.3).

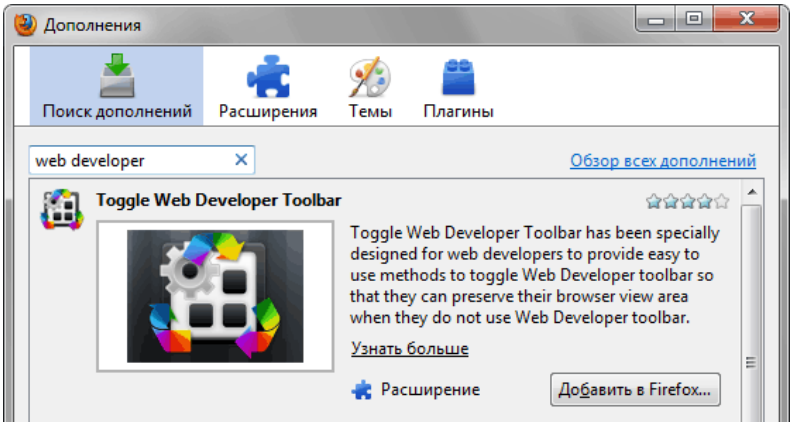


Рис. 10.3. Окно с дополнениями Firefox

Само расширение выглядит как строка меню над вкладками (рис. 10.4), также оно доступно через меню `Инструменты > Web Developer`. Включать/выключать панель можно через меню `Вид > Панели инструментов` либо щелчком правой кнопки мыши по панели расширения.

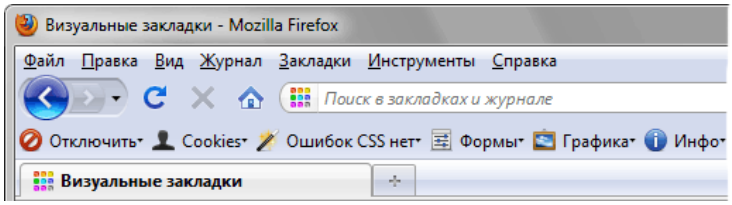
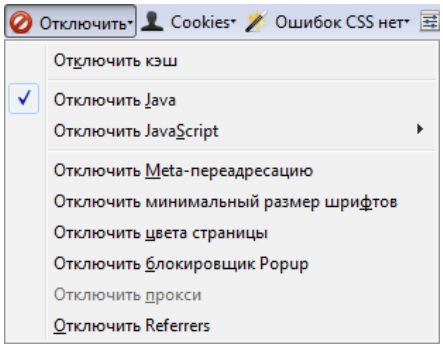


Рис. 10.4. Web Developer в браузере

Пройдёмся по пунктам меню Web Developer.

Отключить



Отключить кэш
Отключает встроенный кэш браузера. Обычно изображения и копии просмотренных страниц сохраняются браузером на локальном диске для экономии времени загрузки. При повторном открытии страницы, браузер сравнивает локальную копию с оригиналом и если они совпадают, то загружается локальная копия. В некоторых случаях происходит загрузка локальной версии страницы, даже при изменении оригинала. К примеру, в настройках браузера может быть установлена проверка каждые два часа, обновилась ли на сервере сохранённая в кэш страница. Впрочем, Firefox не позволяет проводить тонкую настройку кэша наподобие Opera.

Обновить страницу в обход кэша можно также комбинацией клавиш `Ctrl`+`F5`, она поддерживается всеми браузерами.

Отключить Java
Java — язык программирования, разработанный компанией Sun Microsystems. Небольшие программы на этом языке (так называемые апплеты) используются для расширения функциональности веб-страниц. В Firefox встроена поддержка этого языка, которую можно отключить через этот пункт меню.

В версии Firefox 3.6 не работает.

Отключить JavaScript
Язык программирования, предназначенный для работы скриптов — интегрированных с веб-страницей программ. JavaScript широко используется при создании веб-страниц для расширения их функциональности, например создать различные меню, формы, эффекты и др. Если выбрать пункта `Отключить JavaScript > Полностью`, то вся функциональность на сайте перестанет работать. Этот пункт меню можно использовать для проверки работы сайта без скриптов, а также для обхода различных ограничений, которые устанавливают авторы сайтов, вроде отключения работы правой кнопки мыши. Web Developer говорит, что нет таких ограничений, которые нельзя было бы обойти.

Отключить META-переадресацию
С помощью тега `<meta>` можно осуществить автоматическую переадресацию на указанный документ через определённый промежуток времени. Для этого используется тег `<meta>` и значение Refresh атрибута http-equiv (пример 10.1).

Пример 10.1. Автоматическая переадресация

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE  html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <title>Переадресация</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <meta http-equiv="Refresh" content="5; URL=http://htmlbook.ru" />
 </head>
 <body>
  <p>...</p>
 </body>
</html>
```

Переадресация может применяться в чатах для обновления текущего документа или перенаправления на новый адрес. Но также используется и со злым умыслом, например, с целью частого показа контекстной рекламы или баннеров. Данный пункт меню позволяет заблокировать подобный тег `<meta>`.

Отключить минимальный размер шрифтов
В настройках Firefox можно установить минимальный размер шрифта, он будет использоваться для текста, размер которого меньше указанного. Это позволяет сделать просмотр страниц более удобным для чтения, особенно на сайтах, у которых текст отображается шрифтом, размер которого слишком мал для комфортного восприятия.

Чтобы установить минимальный размер шрифта, выберите в меню `Инструменты > Настройки...`, откройте панель `Содержимое` и щёлкните по кнопке `Дополнительно` в группе Шрифты и цвета. Вы можете выбрать минимальный размер шрифта из выпадающего меню `Наименьший размер шрифта`.

Web Developer позволяет быстро включать или отключать эту возможность. Однако, если минимальный размер шрифта в настройках не задан, этот пункт никак не влияет на результат.

Отключить цвета страницы
Выключает любые цвета, заданные с помощью свойства background или background-color. Затрагиваются также фоновые рисунки, которые установлены с помощью background.

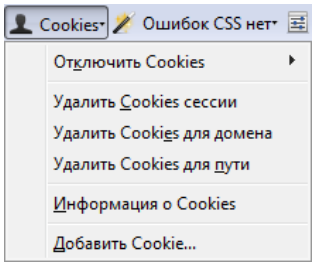
Отключить блокировщик Popup
Всплывающие окна обычно используются для рекламы, поэтому в браузерах они обычно блокируются и не допускаются. Данный пункт позволяет быстро включить и выключить эту опцию.

Отключить прокси
Под прокси понимают обычно сервер или программу, позволяющую подключаться к Интернету, а также с целью создания запросов от имени других клиентов. Если в настройках Firefox (`Инструменты > Настройки..`, панель `Дополнительные`, вкладка `Сеть`, кнопка `Настроить`) стоит «Использовать

системные настройки прокси», то этот пункт неактивен.

Отключить Referrers
Referrer это один из заголовков протокола HTTP и позволяет получить адрес страницы, с которой пользователь пришёл на сайт. Если вы печётесь о безопасности своих данных, включите этот пункт.

Cookies



Кукисы или куки, как их называют на жаргоне, это небольшие текстовые файлы на локальном компьютере, в которых сохраняется полезная для сайта информация. С помощью куки можно запомнить имя пользователя, его статус и другую информацию, которая используется на сайте. Firefox задаёт настройки куки через меню `Инструменты > Настройки...`, панель `Приватность`. В целях безопасности вы вообще можете отключить приём куки с сайтов.

Сами куки представляют собой набор некоторых параметров:

- уникальное имя;
- значение;
- путь — отправлять куки только при совпадении пути и адреса страницы, путь «/» обозначает любую страницу;
- домен — для какого адреса сайта актуальна запись;
- дата истечения — сообщает браузеру, когда куки можно удалить.

Отключить Cookies
Отключает приём куки с сайтов.

Удалить Cookies сессии
Куки часто применяются для проверки аутентификации пользователей. После ввода имени и пароля генерируется уникальный код, сохраняемый в куки. При повторном посещении сайта идёт проверка данного кода, и если он совпадает с серверным, то сайт «узнаёт» пользователя. Выбор этого пункта удаляет все сохранённые сессии.

Удалить Cookies для домена
Удаляет все куки для сайта, который в данный момент открыт в браузере.

Удалить Cookies для пути
Удаляет все куки для сайта, путь которых совпадает с путём сайта открытого в браузере.

Информация о Cookies
Открывается дополнительная страница, где в табличной форме представлены все куки с данного сайта. Их параметры можно отредактировать или вообще удалить куки (рис. 10.5).

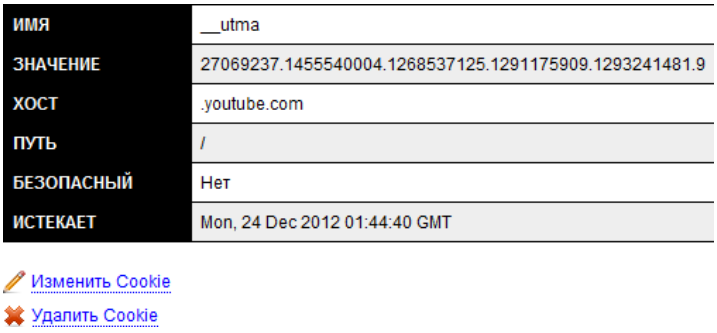


Рис. 10.5. Информация о куки с сайта youtube.com

Добавить Cookies
Позволяет искусственно установить куки для текущего сайта или любого другого, а также задать необходимые параметры (рис. 10.6).

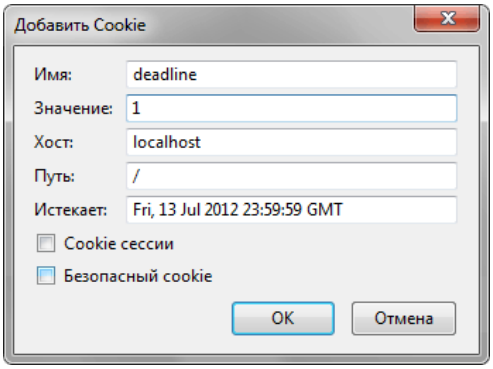
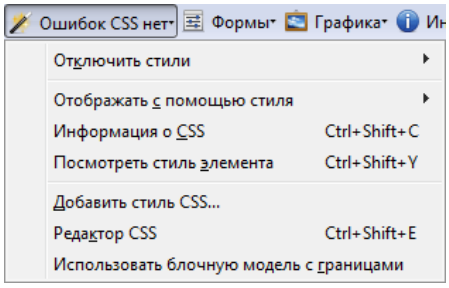


Рис. 10.6. Окно для добавления

Добавление обычно требуется для отладки работы куки и поведения сайта при их наличии.

CSS



Это меню отвечает за стили текущей страницы.

Отключить стили
Предназначено для отключения стилей по какому-либо признаку.

> **Все стили**
> Отключает все используемые на странице стили.
>
> **Стиль браузера по умолчанию**
> Отключает стиль для всех элементов, к которым он добавляется по умолчанию браузером. К примеру, текст внутри `<h1>` и `<p>` имеет разный размер.
>
> ```
> <h1>Заголовок</h1>
> текст
> ```
>
> После отключения стиля размер текста у этих элементов будет одинаковый.

Встроенные стили
Отключает стиль внутри тега `<style>`.

Стили элементов
Отключает стиль, заданный с помощью атрибута style. К примеру, в этой строке красный цвет текста будет проигнорирован.

```
<p style="color: red">Красный цвет</p>
```

Подключенные стили
Отключает стиль установленный с помощью тега `<link>`.

Стили печати
Стили, у которых тип носителя задан как print игнорируются. В действительности, в браузере заметить изменения можно только при печати, в остальных случаях такой стиль никак себя не проявляет.

Отдельный файл
Если на странице загружается сразу несколько стилевых файлов через тег `<link>`, их выборочно можно отключать.

Отображать с помощью стиля
Здесь можно посмотреть, как будет выглядеть страница при печати и наладоннике.

Информация о CSS
Открывается дополнительное окно, где все представлены все стили текущей страницы, включая встроенные.

Посмотреть стиль элемента
При включении этого режима курсор превращается в перекрестье и при наведении на любой элемент страницы он подсвечивается красной рамкой. Уровень вложения выбранного элемента в виде цепочки показан над вкладками (рис. 10.7).

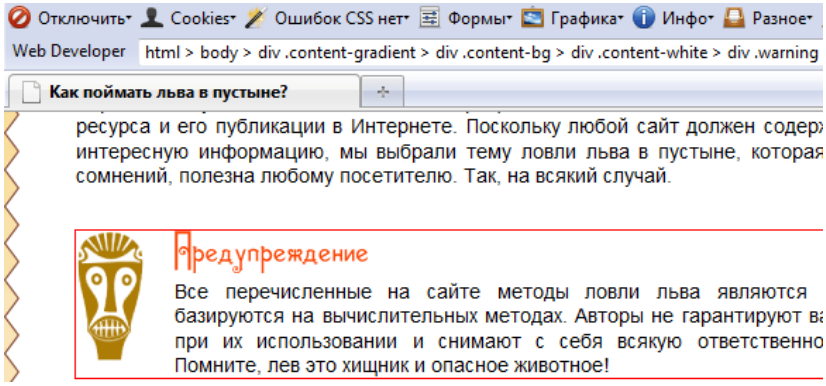


Рис. 10.7. Выбранный элемент

Если щёлкнуть по любому элементу, внизу окна открывается дополнительная панель, в которой отображается его стиль (рис. 10.8). Положение панели можно изменить нажав на кнопку, при этом панель по очереди будет занимать место слева, сверху, справа или снизу. Закрыть панель можно нажав на кнопку, или ещё раз выбрав пункт «Посмотреть стиль элемента».

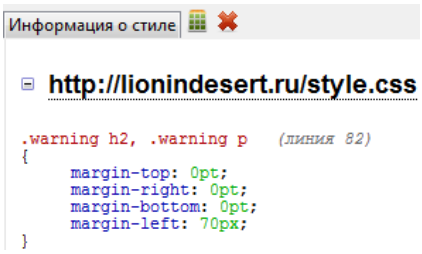


Рис. 10.8. Информация о стиле

Добавить стиль CSS
Для текущей страницы загружается CSS-файл, который и определяет стиль элементов.

Редактор CSS
Этот пункт открывает небольшой редактор в панели, похожей на ту, что показывает информацию о стиле (рис. 10.9).

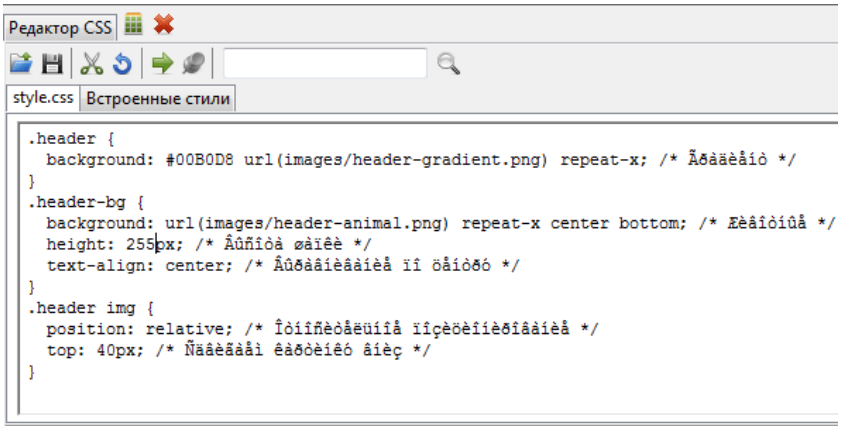
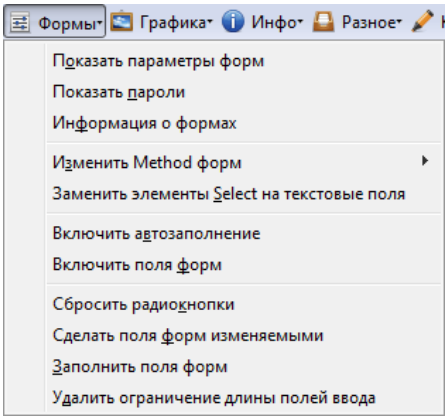


Рис. 10.9. Редактор CSS

В отдельных вкладках показываются стилевые файлы и встроенные стили страницы. Главной особенностью редактора является то, что любые изменения сразу же отражаются в документе. Только не стоит думать, что вы таким образом меняете сам сайт, в действительности, все изменения происходят в памяти браузера и отменяются при закрытии панели. К сожалению, редактор не поддерживает русские символы и комментария выводятся кракозябрами.

Использовать блочную модель с границами
При включении этого пункта меню к ширине элемента не добавляется толщина границ, как это делается по правилам спецификации.

Формы



Меню для управления формами, создаваемыми через тег `<form>` и элементов, которые в них входят.

Показать параметры форм
Отображает элементы форм с их атрибутами, включая тег `<form>`. На рис. 10.10 показана простая форма, как она выглядит в браузере.

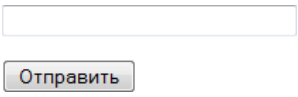


Рис. 10.10. Вид формы в браузере

При включении пункта меню вид формы изменится (рис. 10.11). Сама форма обозначается красной рамкой, а её элементы жёлтым фоном.

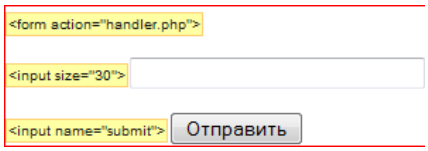


Рис. 10.11. Вид формы при включении пункта меню

Показать пароли
Поле для ввода паролей `<input type="password">` в отличие от обычного текстового поля никогда не показывают введённый текст и скрывают его разными символами. При включении пункта меню все пароли вводятся как обычный текст (рис. 10.12).

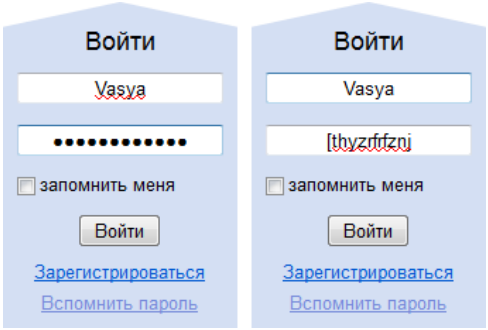


Рис. 10.12. Поле с паролем по умолчанию и при отображении текста

Информация о формах
Открывается новая страница, где в табличной форме представлена информация о формах на странице и их элементах.

Изменить Method форм
Позволяет поменять значение атрибута method тега `<form>` с GET на POST и наоборот.

Заменить элементы Select на текстовые поля
Раскрывающийся список, созданный с помощью тега `<select>`, превращается в обычное текстовое поле. На рис. 10.13 слева показан исходный список, а справа список после изменения.

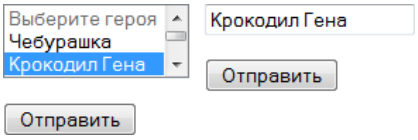


Рис. 10.13. Список до и после изменения

Обратно элемент в список не превращается.

Включить автозаполнение
Браузеры могут сохранять некоторые введённые пользователем данные и затем при повторном посещении сайта автоматически вставлять их в поля форм, это называется автозаполнение. К примеру, Firefox сохраняет пароли, но эту возможность можно отключить через меню Инструменты >

Настройки, панель Защита. Также можно воспользоваться нестандартным атрибутом autocomplete, добавляя его к определённым полям формы со значением on (включить автозаполнение) или off (выключить). Данный пункт меню включает автозаполнение, несмотря на присутствие атрибута autocomplete или настройки браузера.

Включить поля форм

Позволяет сделать доступными элементы форм, у которых задан атрибут disabled (пример 10.2). Его наличие делает элемент неактивным для любых действией.

Пример 10.2. Использование disabled

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE  html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  <html xmlns="http://www.w3.org/1999/xhtml">
  <head>
  <meta  http-equiv="Content-Type" content="text/html;  charset=utf-8" />
  <title>Атрибут  disabled</title>
  </head>
  <body>
  <form  action="handler.php">
  <p><input  type="text" size="30" disabled="disabled"  /></p>
  <p><input  type="submit" name="submit" value="Отправить"
disabled="disabled" /></p>
  </form>
  </body>
</html>
```

Сбросить радиокнопки

Сбрасывает установленные значения для переключателей — элементов форм, заданных через **<input type="radio">**. Сброс происходит полностью, даже если первоначально было установлено значение по умолчанию.

Сделать поля форм изменяемыми

Делает доступными для ввода поля, к которым добавлен атрибут readonly. Его наличие не позволяет вводить новый текст или модифицировать существующий (пример 10.3).

Пример 10.3. Использование readonly

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Атрибут readonly</title>
 </head>
 <body>
  <form action="handler.php">
   <p><input type="text" size="30" readonly="readonly" /></p>
   <p><input type="submit" name="submit" value="Отправить" /></p>
  </form>
 </body>
</html>
```

Заполнить поля форм

Вставляет в текстовые поля значения атрибутов name. К примеру, если для текстового поля указано **<input type="text" name="username">**, то будет вставлено username.

Удалить ограничение длины полей ввода

Чтобы пользователи не вводили длинный текст, его ограничивают с помощью атрибута maxlength тега **<input>**. Значением атрибута выступает максимальное число символов, которые пользователь может ввести в поле (пример 10.4).

Пример 10.4. Использование maxlength

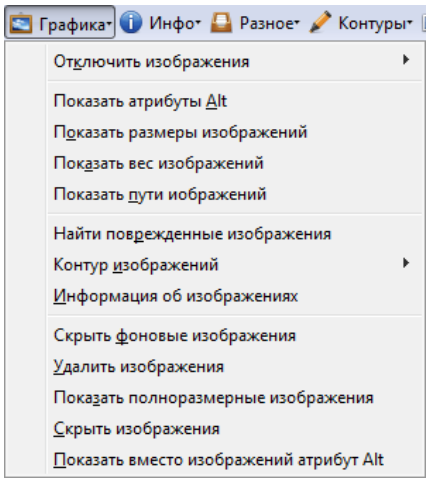
XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
  <title>Атрибут maxlength</title>
 </head>
 <body>
  <form action="handler.php">
   <p><input type="text" size="30" maxlength="10" /></p>
   <p><input type="submit" name="submit" value="Отправить" /></p>
  </form>
 </body>
</html>
```

Данный пункт меню отключает это ограничение и позволяет вводить любое желаемое число символов, которое может быть отправлено на сервер.

Графика



Меню отвечает за информацию о фоновых изображениях и картинках, добавляемых через тег `<img>`.

Отключить изображения

Все изображения

Отключается показ всех изображений на странице, вместо рисунков выводится альтернативный текст. Эта возможность не работает для локальных адресов вроде file:///c:/www/file.html.

Изображения с внешних сайтов

Отключается показ картинок добавленных с других сайтов. Адрес таких изображений начинается с http://.

Анимацию изображений

Отключается анимация для изображений в формате GIF.

Показать атрибуты Alt

Возле изображений выводится значение атрибута alt (рис. 10.14).

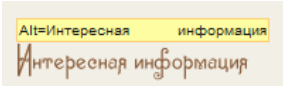


Рис. 10.14. Изображение с выводом alt

Показать размеры изображений

Возле изображений выводится их ширина и высота (рис. 10.15). Причем, эти значения выводятся не фактические, а заданные. Так, если ширина картинки составляет 400 пикселов, а значение width — 200, то будет показано число 200.

Рис. 10.15. Ширина и высота изображения

Показать вес изображений
Возле изображений выводится их объём в байтах или килобайтах (рис. 10.16).

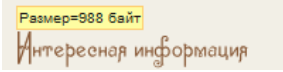


Рис. 10.16. Изображение с выводом src

Показать пути изображений
Возле изображений выводится значение атрибута src (рис. 10.17).

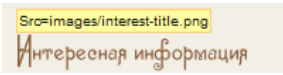


Рис. 10.17. Изображение с выводом src

Найти поврежденные изображения
Открывается страница, где приводится список недоступных изображений. Это обычно происходит по причине неверно указанного адреса.

Контур изображений
Позволяет выделить красной рамкой все изображения или по указанному критерию.

Все изображения
Выделяются все изображения, добавленные с помощью тега `<img>`.

Фоновые изображения
Выделяются только фоновые изображения.

Изображения с указанными размерами
Выделяются изображения, для которых установлены значения width и height.

Изображения с пустыми Alt
Выделяются контуром изображения, у которых alt="".

Увеличенные изображения
Выделяются контуром изображения, у которых размеры не совпадают с указанными значениями width и height.

Изображения без атрибутов Alt
Выделяются контуром изображения, у которых атрибут alt не указан.

Изображения без размеров
Выделяются контуром изображения, у которых не установлен атрибут width или height.

Изображения без атрибутов Title
Выделяются контуром изображения, у которых атрибут title не указан.

Информация об изображениях
Открывается страница со списком всех изображений на странице. Для каждого рисунка выводится его ширина, высота, объём файла и значение атрибута alt (рис. 10.18).

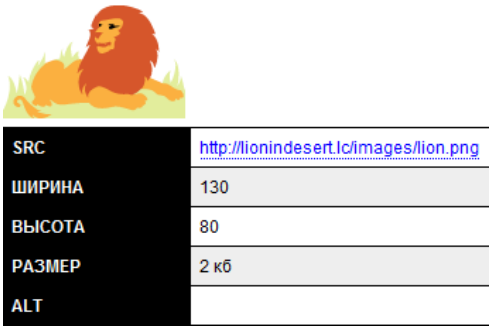


Рис. 10.18. Информация об изображении

Скрыть фоновые рисунки
Все фоновые изображения на странице не отображаются.

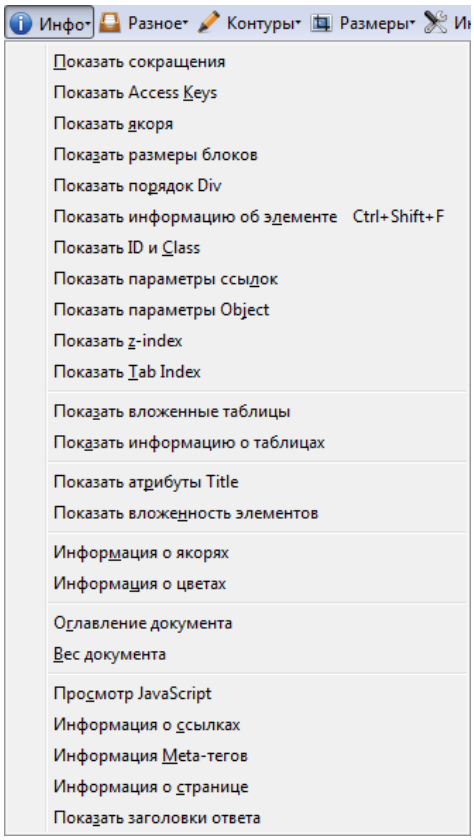
Удалить изображения
Убирает изображения со страницы так, словно их в коде и не было.

Показать полноразмерные изображения
Показывает в исходном размере изображения, у которых заданные через width и height размеры не совпадают с действительными размерами картинки.

Скрыть изображения
Изображения на странице не показываются, но место, которое они занимают, остаётся.

Показать вместо изображений атрибут Alt
Отключает изображения на странице и вместо них отображает значение атрибута alt. По своему действию этот пункт похож на отключение всех изображений.

Инфо



Показать сокращения
Выводит текст в атрибуте title у тегов `<abbr>` и `<acronym>`.

Показать Access Key
Атрибут accesskey позволяет получить доступ к элементу с помощью сочетания клавиш с заданной в атрибуте буквой или цифрой. Данный пункт меню отображает значение accesskey для элементов, у которых он установлен.

Показать якоря
Отображает значения атрибута name или id, установленные для тега `<a>`.

Показать размеры блоков
Вокруг всех тегов `<div>` выводится красная рамка с указанием вычисленных размеров (рис. 10.19).

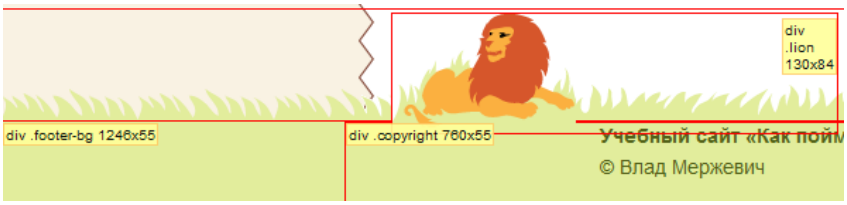


*Рис. 10.19. Размеры *

Показать порядок Div
Выводит порядковый номер тега `<div>` как он идёт в коде документа.

Показать информацию об элементе
Курсор мыши превращается в перекрестье, с помощью которого можно выбрать любой элемент документа. После щелчка по элементу в левом верхнем углу отображается информация о нём (рис. 10.20).

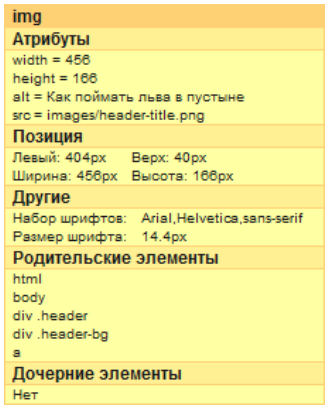


Рис. 10.20. Информация о картинке

Панель можно переставить в другое место, перетащив его мышью за заголовок.

Показать ID и Class
Возле элементов показываются значения атрибутов id и class.

Показать параметры ссылок
Выводятся значения атрибутов у всех тегов `<a>`.

Показать параметры Object
При наличии на странице тега `<object>` выводится информация о его атрибутах и вложенных элементах.

Показать z-index
Показывается значение свойства z-index у тех элементов, где оно добавлено.

Показать Tab Index
Атрибут tabindex устанавливает порядок получения фокуса при переходе между элементами с

помощью клавиши Tab. Переход происходит от меньшего значения к большему, например от 1 к 2, затем к 3 и так далее. Данный пункт меню отображает значения tabindex у тегов, при его наличии.

Показать вложенные таблицы
Если на странице имеются вложенные таблицы, для них выодится их уровень вложения, как показано на рис. 10.21.

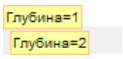


Рис. 10.21. Глубина вложения таблиц

Показать информацию о таблицах
Показывает разную информацию об имеющихся на странице таблицах.

Показать атрибуты Title
Показывается значение атрибута title у тех элементов, где оно добавлено.

Показать вложенность элементов
Интересный режим, при котором страница становится черно-белой, а уровень вложенности элементов показан с помощью оттенков серого цвета. Чем светлее цвет фона у элемента, тем больше у него родителей.

Информация о якорях
Открывается дополнительная страница, где приводится список всех значений id у элементов и name тега **<a>**.

Информация о цветах
Все цвета, применяемые на странице, показываются в виде цветных плашек с шестнадцатеричным значением цвета (рис. 10.22).

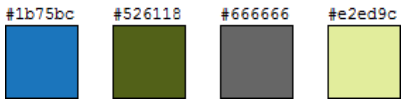


Рис. 10.22. Используемые на странице цвета

Оглавление документа
Содержимое тегов **<h1>**...**<h6>** собирается и выводится в виде иерархической схемы документа (рис. 10.23).

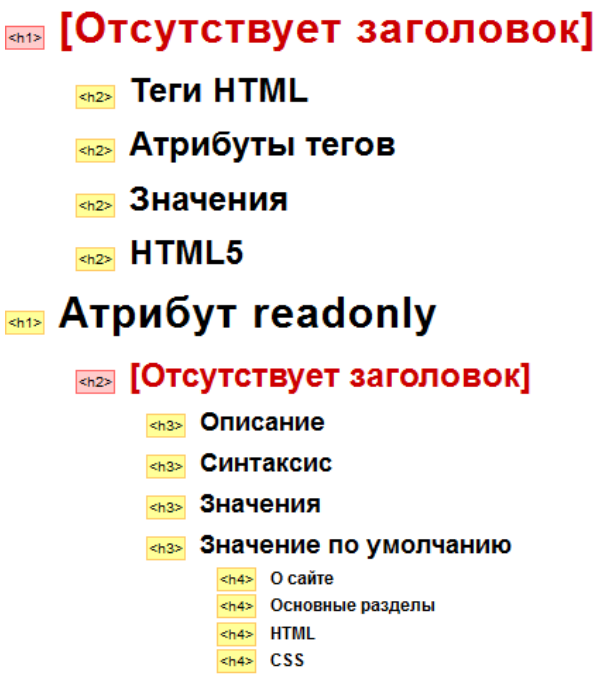


Рис. 10.23. Оглавление документа

Вес документа
Список компонент документа, для которых выводится их суммарный объём, передаваемый по сети (рис. 10.24).

Рис. 10.24. Объём компонент и суммарный вес документа

Каждую из компонент можно раскрыть и посмотреть занимаемый объём каждого изображения, файла скрипта или стиля.

Просмотр JavaScript
Страница, на которой представлено содержимое всех файлов JavaScript, а также скриптов, встроенных в код документа.

Информация о ссылках
Страница со списком всех ссылок на странице.

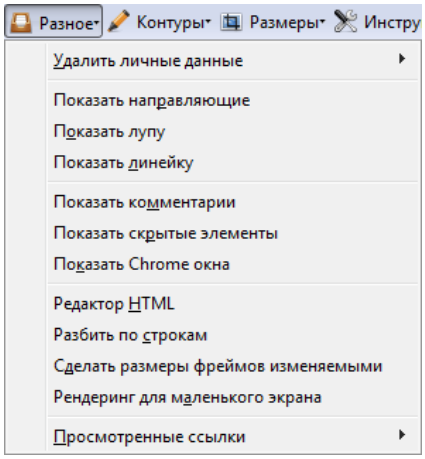
Информация Meta-тегов
Страница со списком тегов `<meta>` в документе и их значений.

Информация о странице
Открывается окно, доступное также через меню `Инструменты > Информация о странице`.

Показать заголовки ответов
Страница, на которой выводятся заголовки ответа сервера. Из них можно узнать о версии сервера, типе передаваемого документа, кодировке и коде ответа. Например, 200 означает, что страница отдаётся корректно, а 404 — что документ не найден.

Рис. 10.25. Заголовки ответа

Разное



Удалить личные данные
Этот пункт дублирует меню `Инструменты > Стереть недавнюю историю` и позволяет стереть кэш, журнал и сессии.

Показать направляющие
Направляющие это горизонтальные и вертикальные линии, помогающие сравнить выравнивание элементов (рис. 10.26). Цвет направляющих задаётся через открывшееся меню, через него же добавляются новые направляющие.

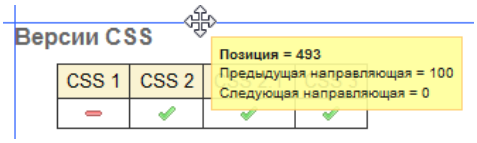


Рис. 10.26. Направляющие

Показать лупу
Увеличивает фрагмент страницы в указанное число раз.

Показать линейку
Включает инструмент для измерения ширины и высоты. Курсор мыши превращается в перекрестье, с помощью которого можно провести прямоугольник в любом месте страницы (рис. 10.27). Размеры прямоугольника в пикселах выводятся на панели Web Developer, там же показывается начальная и конечная позиция.

Рис. 10.27. Измерение размеров

Квадратики по углам предназначены для изменения размеров прямоугольника, достаточно потянуть за уголок. Сам прямоугольник можно перетаскивать мышью или задавать ему точные размеры введя их в

поле на панели.

Показать комментарии
Отображает содержимое комментариев, т.е. текст и элементы внутри `<!-- -->`.

Показать скрытые элементы
Отображает элементы `<input type="hidden">`, по умолчанию они никак не выводятся на странице.

Показать Chrome окна
Показывает окна браузера без стандартного оформления, которые называются Chrome окна.

Редактор HTML
Открывается редактор похожий на редактор CSS (см. рис. 10.9) в котором можно вносить изменения в код документа. Все правки сразу же отображаются на странице.

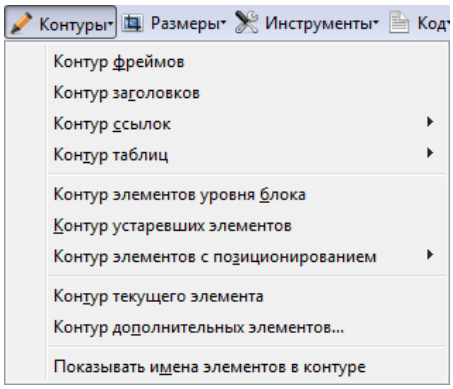
Разбить по строкам
Каждый элемент страницы располагается на отдельной строке.

Сделать размеры фреймов изменяемыми
Если для тега `<frame>` указан атрибут noresize, то запрещено изменять размер фреймов. Данный пункт отменяет действие noresize.

Рендеринг для маленького экрана
Страница показывается в маленьком виде, словно при просмотре на небольшом экране устройства.

Просмотренные ссылки
Позволяет пометить все просмотренные ссылки как непросмотренные и наоборот.

Контуры



Выделяет рамкой разные элементы по указанному признаку.

Контур фреймов
Выделяет на странице фреймы.

Контур заголовков
Выделяет контурами разных цветов заголовки от `<h1>` до `<h6>`.

Контур ссылок
Выделяет внешние ссылки, ссылки с атрибутом title.

Контур таблиц
Выделяет заголовок таблицы, ячейки или все таблицы.

Контур элементов уровня блока
Выделяет блочные элементы.

Контур устаревших элементов
Выделяет осуждаемые в спецификации HTML теги вроде `<basefont>`, `<center>`, `<font>`, `<strike>` и др.

Контур элементов с позиционированием
Выделяет элементы с абсолютным, фиксированным, относительным позиционированием, а также плавающим (у них установлено свойство float).

Контур текущего элемента
Позволяет выделить рамкой любой элемент на странице при наведении на него курсора мыши. В панели также отображается уровень вложения для этого элемента.

Контур дополнительных элементов
Позволяет выделить контуром желаемого цвета указанные в окне элементы. Для этого требуется ввести имя тега и задать цвет контура (рис. 10.28).

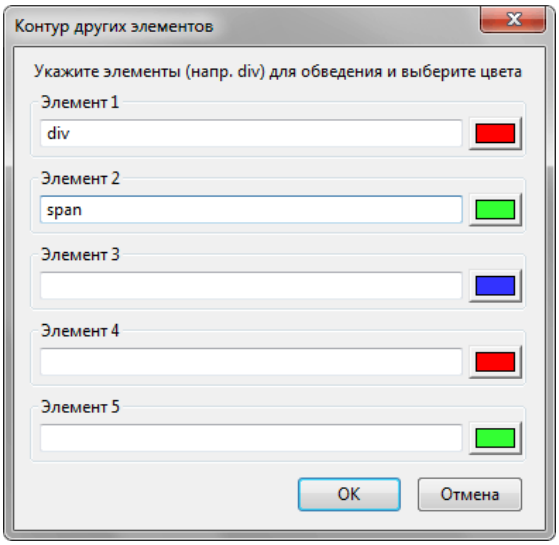
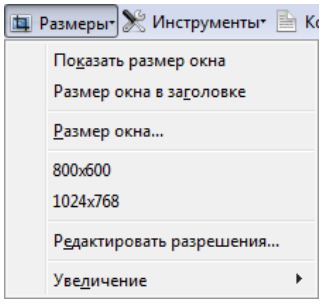


Рис. 10.28. Контур желаемых элементов

Показывать имена элементов в контуре
Возле выделенных контуром элементов отображается тег.

Размеры



Позволяет отслеживать ширину и высоту окна браузера или рабочей области, а также изменять их до указанных размеров.

Показать размер окна
Выводит информационное сообщение, в котором указаны размеры окна и видимой области (рис. 10.29).

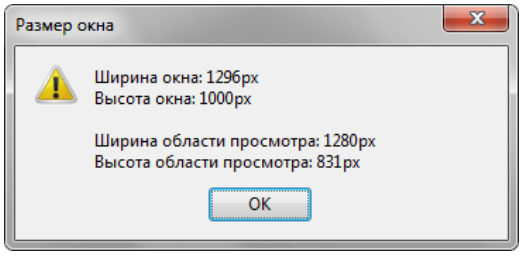


Рис. 10.29. Размеры окна

Размер окна в заголовке
Размеры окна и видимой области показываются в заголовке браузера (рис. 10.30).

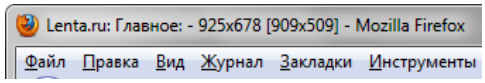


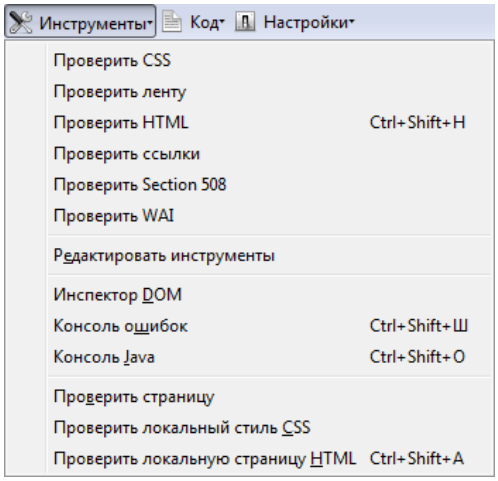
Рис. 10.30. Вид заголовка

Размер окна...
Позволяет указать ширину и высоту окна или видимой области. Введённые значения отображаются в меню ниже. При выборе размера, к примеру, 1024x768 окно браузера изменится до ширины 1024 пикселов и высоты 768 пикселов. Это позволяет проверить работу сайта при указанном разрешении монитора.

Редактировать разрешения...
Позволяет отредактировать значения в списке размеров окна.

Увеличение
Увеличивает и уменьшает масштаб страницы целиком.

Инструменты



Различные инструменты для проверки кода документа на наличие ошибок.

Проверить CSS
Проверяет CSS на валидность кода через сайт http://jigsaw.w3.org.

Проверить ленту
Проверяет код страницы через сервис http://validator.w3.org/feed на валидность ленты в формате Atom или RSS. Такая лента обычно предназначена для описания новостей сайта и анонсов статей.

Проверить HTML
Проверяет HTML на валидность кода через сайт http://validator.w3.org.

Проверить ссылки
Проверяет ссылки через сервис http://validator.w3.org/checklink. Тестирует, что якоря не определены дважды и что ссылки не битые. С этой целью для каждой ссылки на текущей странице проверяется её отклик.

Проверить Section 508
Стандарт для проверки сайта на его доступность людям с ограниченными возможностями. Страница проверяется через сайт cynthiasays.com, где по ней выводится отчёт о прохождении тестов.

Проверить WAI
Проверяет страницу на Web Accessibility Initiative (инициатива доступности по сети) — ряд тестов на соответствие страницы программе по организации доступа для людей с ограниченными возможностями.

Редактировать инструменты
Позволяет изменить ссылки для проверки вышеперечисленных пунктов.

Инспектор DOM
Вызывает Инспектор DOM (Document Object Model, объектная модель документа), который устанавливается вместе с Firefox. Если этот пункт недоступен, требуется переустановить браузер.

Консоль ошибок
Открывает окно, в котором перечислены ошибки, предупреждения и сообщения о работе JavaScript и CSS (рис. 10.32).

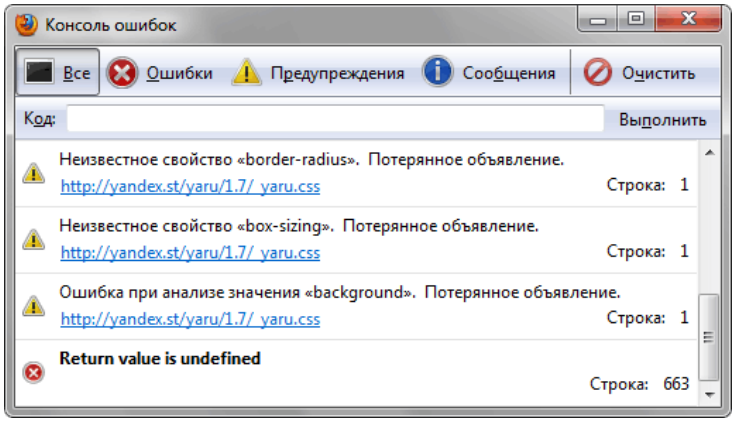


Рис. 10.32. Консоль ошибок

Также открыть консоль можно через кнопки на панели Web Developer, расположенные по правому краю (рис. 10.33).

Рис. 10.33. Сообщение о режиме и ошибках

Первый значок говорит, в каком режиме отображается страница — стандартном (зелёная галочка) или совместимости (красный крестик), второй значок отвечает за CSS, а третий за JavaScript. При отключении скриптов или их отсутствии выводится серый кружок.

Web Developer не понимает CSS3, а также специфические для браузеров стилевые свойства, поэтому наличие opacity, -moz-border-radius или подобных свойств отражается в консоли как ошибка CSS. К таким ошибкам следует относиться спокойно, в действительности ошибками это не является.

Консоль ошибок очень полезна при отладке скриптов JavaScript, поскольку нарушения в их работе сразу же отображаются в списке.

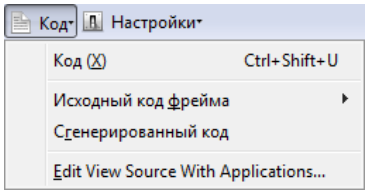
Консоль Java
Вызывает консоль Java. Не работает в Firefox 3.6.

Проверить страницу
Проверить текущую страницу на валидацию HTMLи CSS. В отличие от предыдущих пунктов, здесь отправка кода происходит в фоновом режиме, а результат проверки отображается на панели.

Проверить локальный стиль CSS
Если у вас локальная страница, то проверка на валидность CSS через меню «Проверить CSS» не пройдёт, потому что сервис валидации требует обращаться к коду по его адресу. Данный пункт отправляет сформированный код CSS на сайт http://jigsaw.w3.org, где выводится сообщение о результатах проверки.

Проверить локальную страницу HTML
Работает аналогично предыдущему пункту, но проверяет HTML-код с помощью сервиса http://validator.w3.org.

Код



Это меню предназначено для просмотра кода документа.

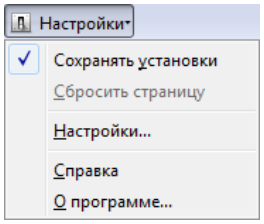
Код
Открывает исходный код страницы.

Исходный код фрейма
Если на странице присутствуют фреймы или плавающие фреймы, то в меню появляется их список, с помощью которого можно открыть их исходный код.

Сгенерированный код
Некоторые расширения могут вносить свои изменения в исходный код страницы, причём делая это совершенно незаметно. Просмотр исходного кода никак не выявит это воздействие, а этот пункт меню позволяет просмотреть код страницы так, как его видит браузер.

Edit View Source With Applications...
Открывает исходный код документа через указанное внешнее приложение.

Настройки



В этом меню собраны настройки программы.

Сохранять установки
Запоминает выбор пользователя для разных меню.

Сбросить страницу
Если были выбраны пункты меню, ведущие к изменению её вида, к примеру, контурами выделены заголовки, то сброс страницы приведёт к отключению этих меню и возврату в исходное состояние. В

остальных случаях этот пункт меню неактивен.

Настройки...
Различные настройки расширения, вроде установки комбинаций клавиш для быстрого доступа, вид панели и др.

Справка
Открывается страница со списком часто задаваемых вопросов. В качестве справки по программе выглядит весьма куце.

О программе
Открывается окно, где указана версия Web Developer, дата сборки и автор.

Отладка кода с помощью расширения Firebug

Отладка это процесс нахождения ошибок в коде и исправления нежелаемого поведения элементов в браузере. Как правило, отклонение макета от первоначального дизайна отслеживается в процессе вёрстки, но бывает и несколько ситуаций, когда ошибки необходимо исправить уже на рабочем сайте. К примеру, ошибка может быть выявлена после добавления нового блока контента, тестирования сайта в разных версиях браузеров, при различных разрешениях монитора и других условий. Также разработчик должен уметь быстро понимать чужой код, отследить причину появления ошибки и устранить её. Понимание логики чужого кода требуется при работе в команде, либо при возврате к собственной работе спустя какое-то время, когда она уже воспринимается как чужая.

На объёмных сайтах с десятками тысяч строк исходного кода HTML вычленить проблемное место достаточно сложно, поэтому требуется инструмент вроде Web Developer или Firebug, который позволяет показать код HTML и CSS выбранного фрагмента и провести над ним эксперименты.

Firebug одно из лучших, если не лучшее расширение под Firefox для веб-разработчика. Появившись в 2007 году, это расширение задало определённую планку качества для создателей браузеров, что впоследствии привело к появлению новых версий браузеров, в которых были добавлены инструменты, по своему принципу похожие на Firebug. Называются они по-разному

- Apple Safari — Веб-инспектор.
- Opera — Dragonfly.
- Internet Explorer — Средства разработчика.
- Google Chrome — Инструменты разработчика.

Возможности Firebug

Возможности программы достаточно велики и охватывают несколько этапов веб-разработки, в частности, не только вёрстку страниц, но и программирование. Эта тема, в силу другой направленности книги, рассматриваться не будет.

Исходный код
В отличие от обычного исходного кода HTML, Firebug учитывает изменение кода после его преобразования через JavaScript. Таким образом, вы сможете отследить элементы, добавляемые через скрипты, и посмотреть стиль, которые невозможно обнаружить иначе.

Быстрое редактирование HTML
С помощью Firebug можно вносить изменения непосредственно в код документа и видеть результат их воздействия немедленно. Вы можете удалять, добавлять, редактировать атрибуты тегов, просто щелкая по ним.

Инспектирование кода CSS
Для любого выбранного элемента показывается его стиль, отдельные свойства при этом можно отключать, добавлять новые свойства, менять значения существующих. Результаты изменений отображаются в браузере немедленно.

Поиск любого элемента мышкой
С помощью инструмента «Анализировать элемент» вы можете выбрать любой текст, изображение или объект на странице, просто щелкнув по нему. При этом откроется HTML-код выбранного элемента и его стиль.

Установка и вызов

Установка Firebug происходит аналогично другим расширениям. Перейдите по адресу

https://addons.mozilla.org/ru/firefox/addon/firebug/

и нажмите на кнопку «Добавить в Firefox» (рис. 10.34).

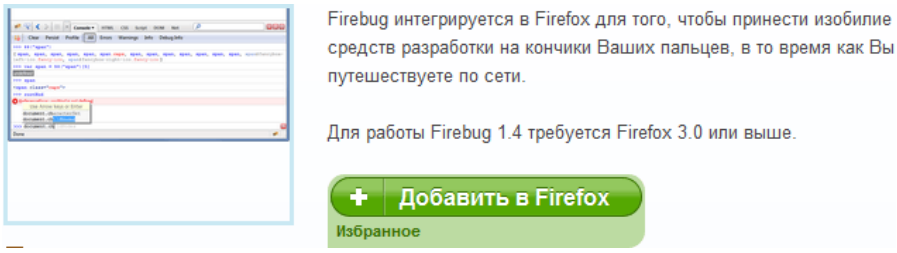


Рис. 10.34. Добавление расширения

После этого появится окно с предупреждением (рис. 10.35), в котором надо нажать «Установить».

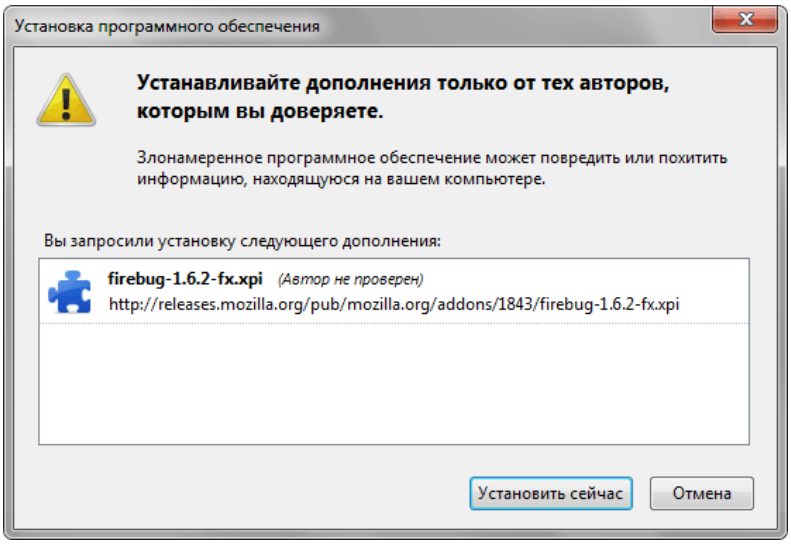


Рис. 10.35. Установка программного обеспечения

Вызов Firebug происходит несколькими способами:

- через меню Инструменты > Firebug;
- через горячую клавишу F12;
- щелчком по картинке жука в правом углу браузера Firefox (рис. 10.36); картинка доступна только при включенной строке состояния (Вид > Строка состояния).

Рис. 10.36. Значок Firebug

Стиль

Firebug открывается в нижней части браузера и его окно состоит из двух основных панелей и меню. В левой панели по умолчанию отображается HTML, а в правой стиль выбранного элемента (рис. 10.37).

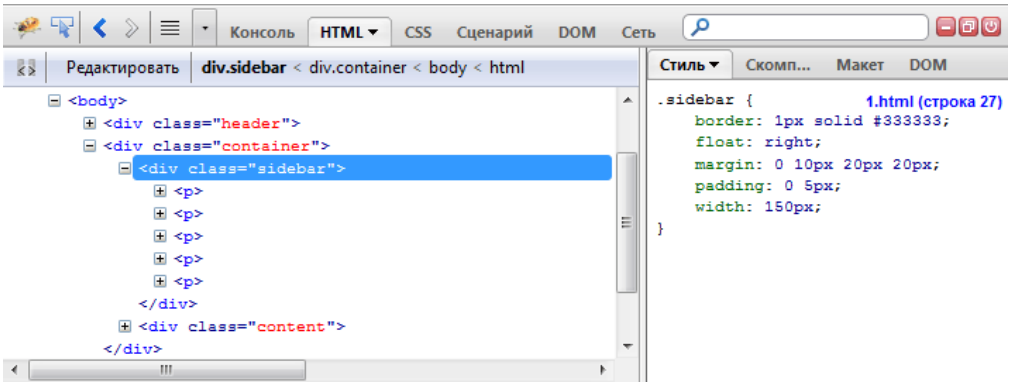


Рис. 10.37. Интерфейс Firebug

Код HTML выводится в виде иерархической структуры, если элемент содержит дочерние элементы, рядом с ним выводится плюсик. На плюс можно щёлкнуть и раскрыть содержимое контейнера, а также свернуть его, щёлкнув ещё раз. При выборе элемента в правой части отображается его стиль (рис. 10.38). Любой элемент можно выбрать не только через код HTML, но и непосредственно на странице с помощью инструмента либо через меню Анализировать элемент (Ctrl + Shift + C).

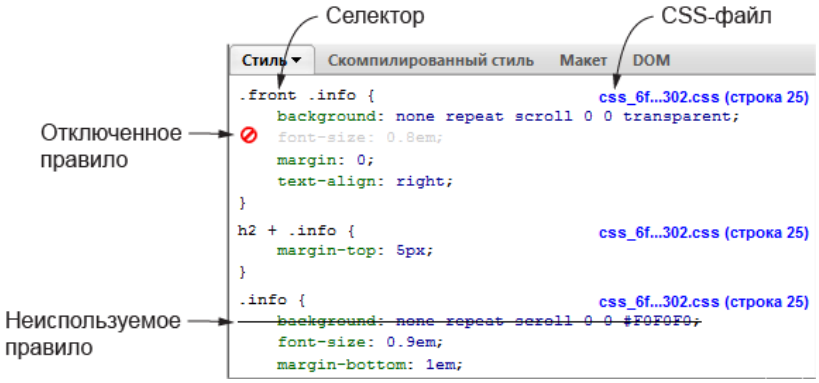


Рис. 10.38. Стиль выбранного элемента

В списке отображаются селекторы, имеющие отношение к выбранному элементу, название CSS-файла и строка, в которой эти селекторы описаны. Если какая-то строка зачёркнута, значит, на неё действуют правила каскадирования и у данного селектора специфичность ниже, чем у селектора выше. На рисунке видно, что у селектора `.front .info` специфичность выше, чем у селектора `.info`, поэтому свойства background и font-size не применяются. Учтите также, отображаемый стиль не соответствует стилю в CSS-файле, поэтому вы можете обнаружить значения вроде transparent, которые не вводили в стиль.

Любое свойство легко отключается и включается вновь простым щелчком по пустому полю перед свойством. Отключенные свойства отображаются серым цветом, рядом с ними выводится красный перечёркнутый кружок, как показано на рис. 10.38. Кроме отключения любых свойств их можно изменять. Для этого достаточно щёлкнуть по свойству, и оно станет редактируемым (рис. 10.39).

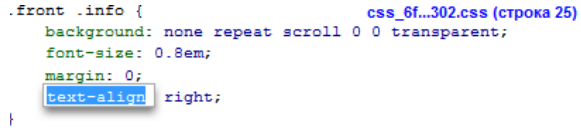


Рис. 10.39. Изменение стилевого свойства

Аналогичное действие производится и со значениями, только в этом случае требуется щёлкнуть уже по нему (рис. 10.40).

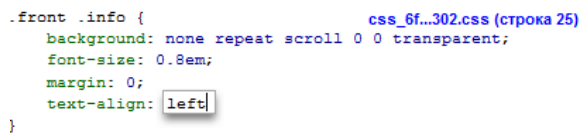


Рис. 10.40. Изменение значения

Добавление нового свойства, а также редактирование существующих происходит через контекстное меню (10.41).

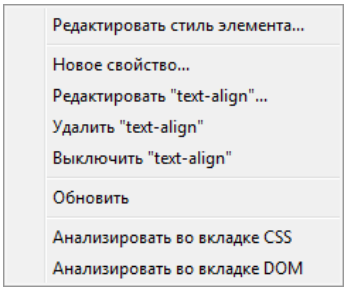


Рис. 10.41. Контекстное меню

Редактировать стиль элемента
Для элемента установить встроенный стиль (он включается через атрибут style).

Новое свойство
Добавить новое стилевое свойство и его значение.

Редактировать
Изменить выбранное свойство.

Удалить
Удалить выбранное свойство

Выключить
Отключить стилевое правило

Обновить
Обновить стиль. Даёт результат при модификации стилевого файла внешней программой.

Анализировать во вкладке CSS
Перейти во вкладку CSS.

Анализировать во вкладке DOM
Перейти во вкладку DOM.

Отключение свойства, изменение их значений и любые другие манипуляции со стилями сразу же отображаются на веб-странице. Если страницу обновить, все действия в Firebug отменяются. Благодаря этому можно экспериментировать с разными свойствами, не опасаясь принести вред документу.

Скомпилированный стиль

Для каждого элемента документа браузер автоматически формирует свой стиль, основываясь на внутренних значениях браузера и стиле родительских элементов. Во вкладке «Скомпилированный стиль» показанной на рис. 10.43 приведены свойства и их значения, которые явно не задаются через стили.

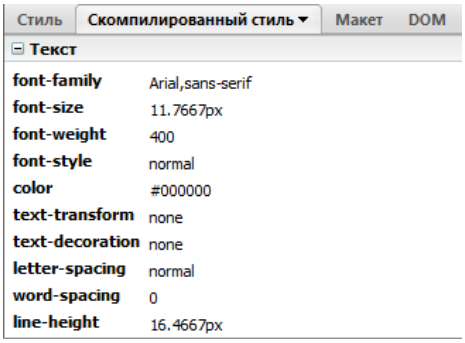


Рис. 10.43. Скомпилированный стиль

Для примера, значение font-family наследуется от селектора **body**, размер шрифта заданный в em переводится для font-family в пикселы, значение font-weight браузер по умолчанию считает за 400.

Скомпилированный стиль нельзя изменить, и он приведён лишь для информации.

Макет

В этой вкладке для выбранного элемента показывается его блочная модель, включающая ширину и высоту элемента, значение полей, границ, отступов и позиционирования (рис. 10.44).

Рис. 10.44. Блочная модель элемента

Позиция показывает тип позиционирования, сдвиг элемента от верхнего, правого, нижнего и левого края, z — значение свойства z-index. Так, для приведённого рисунка стиль будет следующий.

```
position: relative;
top: 15px;
right: 15px;
z-index: 100;
```

Любые числа на макете можно непосредственно изменять, щёлкнув по ним (рис. 10.45). После ввода нового значения к элементу автоматически добавляется style с введённым числом, а во вкладке Стиль селектор обозначаются как **element.style**. К нему применимы те же действия, что и для любых других селекторов.

```
element.style {
    border-top-width: 3px;
    top: 20px;
}
```

Рис. 10.45. Редактирование значений в макете

Использование Firebug на практике

Рассмотрим пару примеров для лучшего понимания, как Firebug поможет нам выявить ошибку и отладить код.

Пример 1

На рис. 10.46 показана страница, у которой белая полоса с контентом располагается по левому краю, а не по центру, как это требуется. Нажимаем F12 для открытия Firebug, выбираем инструмент и щёлкаем по контенту страницы.

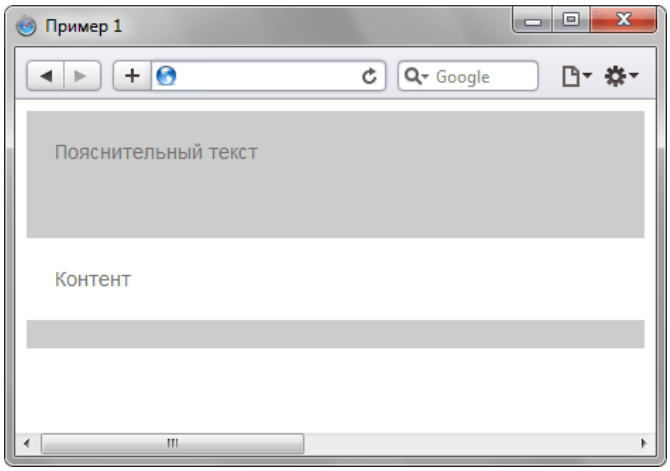


Рис. 10.46. Вид страницы с ошибкой

После выбора элемента в панели Стиль открывается набор стилевых правил для контента (рис. 10.47).

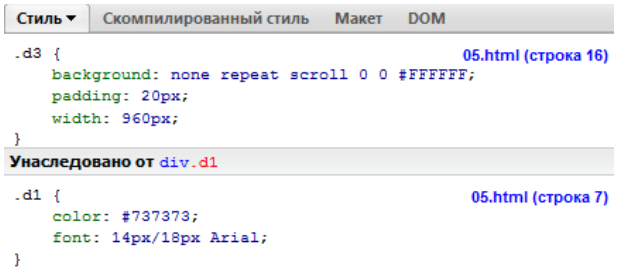


Рис. 10.47. Стиль для выбранного элемента

В общем, понятно, почему не происходит выравнивания по центру, потому что для этого не предусмотрено никаких свойств. Щёлкаем правой кнопкой в панели Стиль, выбираем пункт «Новое свойство...», вводим margin и нажимаем Enter. После чего вводим значение вновь добавленного свойства — auto (рис. 10.48) и наблюдаем, как контент выравнивается согласно задумке.

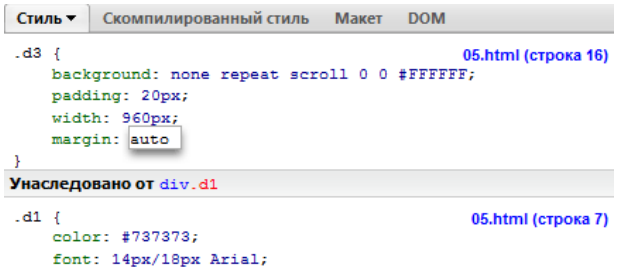


Рис. 10.48. Новое свойство

В примере 10.5 приведён код содержащий ошибку. При этом он соответствует спецификациям XHTML и CSS.

Пример 10.5. Выравнивание по центру

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Пример 1</title>
  <style type="text/css">
   .d1 {
    background: #CCC; color: #737373;
    font: 14px/18px Arial;
    padding: 0 0 20px;
    margin: auto; width: 100%;
   }
   .d2 {
    height: 50px; padding: 20px;
   }
   .d3 {
    background: #FFF; padding: 20px; width: 960px;
   }
  </style>
 </head>
 <body>
  <div class="d1">
   <div class="d2">Пояснительный текст</div>
   <div class="d3">Контент</div>
  </div>
 </body>
</html>
```

Пример 2

К следующему примеру в полной мере относится выражение «проще переделать с нуля, чем исправлять» (пример 10.6).

Пример 10.6. Сайт школы

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="06.css" type="text/css">
<title>Школа № 174</title>
</head><body>
<div id="wrapper">
<div id="border">
  <div id="header">
  <div id="headertext">Вас приветствует школа № 174</div>
  </div>
  <div id="all-menu">
  <div id="menu">
   <div id="menu-in">
   <h1>Наша история</h1>
    <p>У 1923 году в городе Мутищи Каштагольского района
    образовалась школа № 174. Название школы произошло по числу
    жителей нашего города, которых поголовно записали в первый
    класс. Через три года школу сумели закончить больше половины.</p>
   </div>
  </div>
  <div id="menu">
   <div id="menu-in">
   <h1>Наша история</h1>
    <p>У 1923 году в городе Мутищи Каштагольского района
    образовалась школа № 174. Название школы произошло по числу
    жителей нашего города, которых поголовно записали в первый
    класс. Через три года школу сумели закончить больше половины.</p>
   </div>
    </div>
    </div>
    <div id="bar-all">
     Правая колонка
    </div>
</div>
</div>
</body></html>
```

Проблем несколько:

- заголовок выходит за пределы рамки;
- появляется горизонтальная полоса прокрутки;
- текст в правой колонке выравнивается не по верхнему краю, а смещается вниз.

На рис. 10.49 показан фрагмент макета с указанными проблемами.

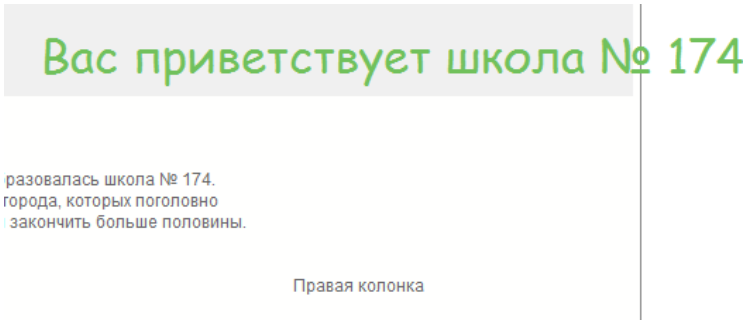


Рис. 10.49. Проблемы с отображением макета

Для начала проверим код валидатором. Он обнаружил две ошибки связанных с повторным использованием идентификаторов. Меняем в HTML-коде id="menu" на class="menu", а id="menu-in" на class="menu-in", а в стилевом файле `#menu` на `.menu` и `#menu-in` на `.menu-in`. Разумеется, это не приведёт к исправлению ошибок отображения, но мы будем уверены, что код в порядке и с ним точно никаких проблем нет.

Настала очередь Firebug. Открываем его, выбираем инструмент и щелкаем по тексту «Вас приветствует...». В панели Стиль видно, что неприятности возникают из-за свойства margin, в этом легко убедиться, временно его отключив. Текст при отключении перемещается в левый верхний угол родителя, а полоса прокрутки исчезает. Ошибка связана с тем, что блок сдвигается вправо на 350px, выходя за границу макета. Достаточно поменять это значение на меньшее число и проблема будет решена (рис. 10.50).

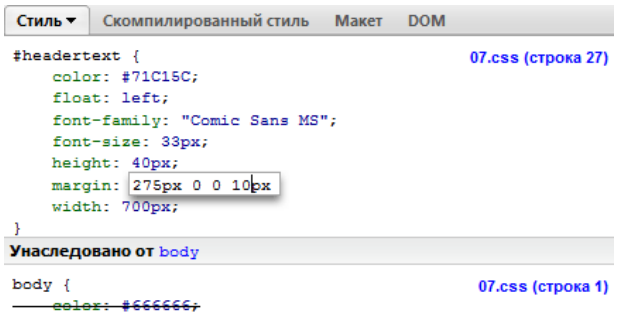


Рис. 10.50. Редактирование стиля

Помним, что Firebug все манипуляции проделывает в памяти и в файлах изменения никак не отражаются. Поэтому после обнаружения ошибки в коде открыть стилевой файл и внести в него необходимые исправления (пример 10.7).

Пример 10.7. Стиль для заголовка

```
#headertext {
  width:700px;
  height:40px;
  font-family:"Comic Sans MS";
  font-size:33px;
  color:#71c15c;
  margin:275px 0 0 10px;
  float:left;
}
```

Остаётся правая колонка. Проделываем те же манипуляции в Firebug для просмотра стиля элемента. Кроме удаления лишнего display в `#bar-all` ничего изменять особо не придётся, поэтому переходим к

другой колонке `#all-menu`. В стилях, оказывается, такого идентификатора нет, поэтому мы можем редактировать только встроенный стиль элемента. Необходимо добавить float со значением left и указать ширину левой колонки (рис. 10.51).

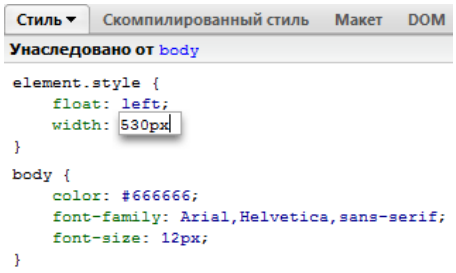


Рис. 10.51. Редактирование стиля колонки

Вот теперь порядок. Окончательно вносим правки в стилевой файл (пример 10.8) и проверяем в других браузерах на корректность.

Пример 10.8. Стиль левой колонки

```
#all-menu {
  float: left;
  width: 530px;
  }
```

Веб-инспектор Safari

В браузер Safari включено несколько средств для разработчика, позволяющих упростить создание и отладку сайта. К примеру, через Safari можно просматривать, как сайт будет выглядеть на iPhone и iPad. Для этого всего лишь требуется указать соответствующий пользовательский агент через меню Разработка (рис. 1.52).

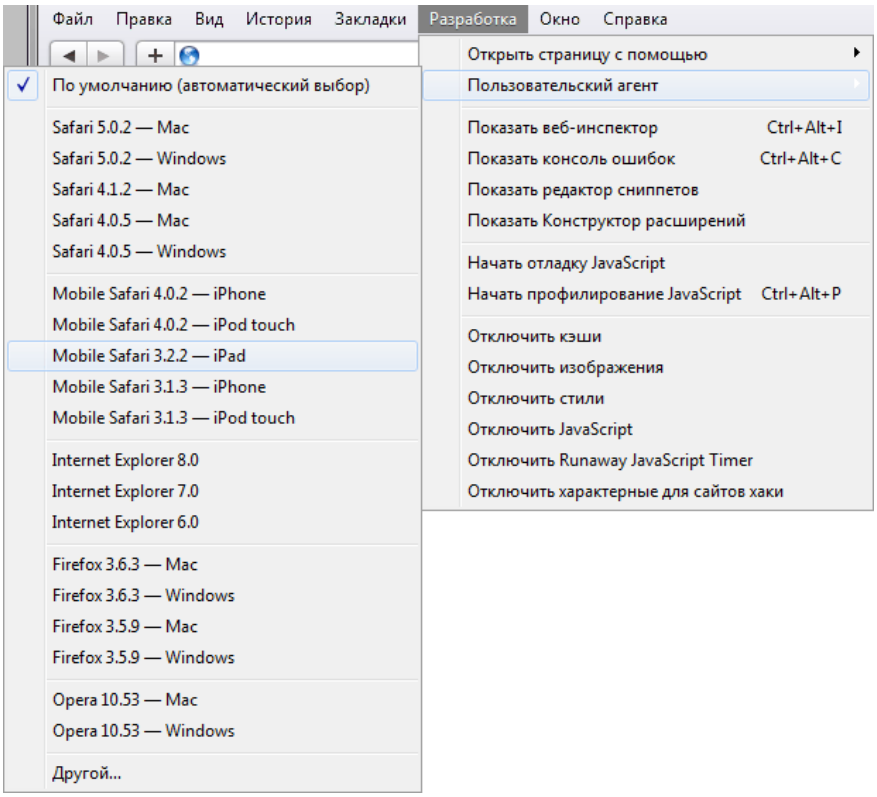


Рис. 1.52. Выбор пользовательского агента

Также в Safari имеется прекрасный инструмент для отладки HTML и CSS кода под названием веб-инспектор. Для его вызова выберите меню Разработка > Показать веб-инспектор или воспользуйтесь комбинацией клавиш Ctrl + Alt + I. Если у вас нет этого пункта меню, зайдите в настройки браузера, выберите панель Дополнения и поставьте галочку в пункте «Показать меню Разработка в строке меню».

Веб-инспектор выводит список ресурсов, найденных на веб-странице — документы, таблицы стилей, изображения и скрипты. Позволяет увидеть и найти код веб-страницы и стилевые свойства элементов. Интерфейс веб-инспектора продемонстрирован на рис. 1.53. Из всего богатства возможностей программы для вёрстки нас интересует только меню Элементы.

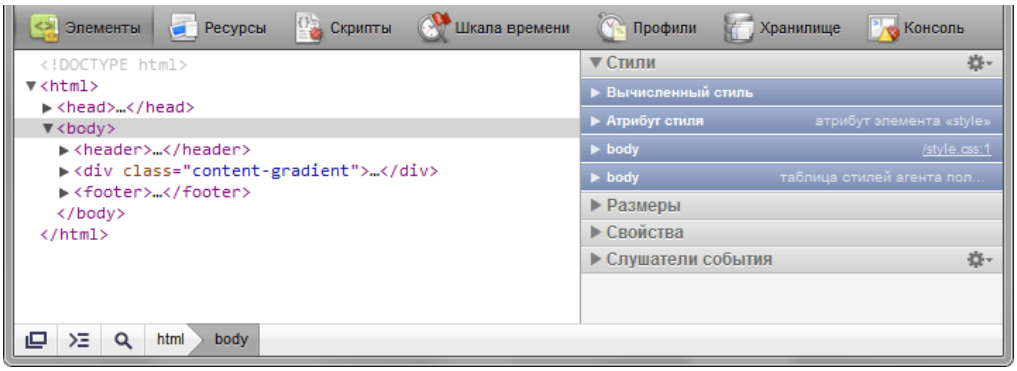


Рис. 1.53. Веб-инспектор

По своему интерфейсу веб-инспектор напоминает Firebug — в левой панели показан HTML-код

текущего документа, в правой панели — стиль выбранного элемента. Любой элемент на странице можно выбирать не только через код, но и с помощью инструмента он позволяет выбирать элемент простым щелчком по нему.

Набор стилей в инспекторе представлен в виде раскрывающегося списка, это позволяет компактно включить большой объём информации. Первый пункт «Вычисленный стиль» (рис. 1.54) содержит стиль элемента, который устанавливает браузер на основе собственного стиля, учёта наследования и добавленных свойств.

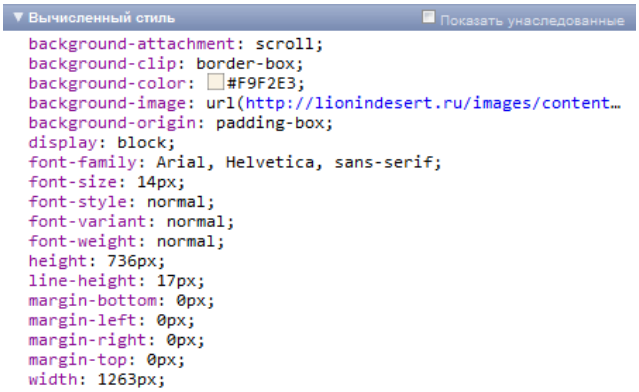


Рис. 1.54. Вычисленный стиль

Эти стили приведены для информации, менять что-либо в этом пункте нельзя. По цвету, если он представлен, можно щёлкнуть и изменить его формат представления на шестнадцатеричный, RGB или HSL.

Следующий пункт «Атрибут стиля» показывает встроенный стиль, т.е. тот, который задан через атрибут style. Пункт будет пустовать, если такого стиля нет.

Нижележащая группа пунктов показывает стиль текущего элемента и взаимосвязанных с ним элементов (рис. 1.55).

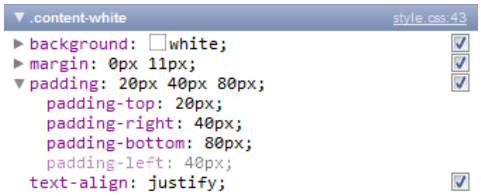


Рис. 1.55. Стиль элемента

Любое свойство в этом списке можно отредактировать, щёлкнув по нему. Универсальные свойства вроде padding или margin выводятся в виде раскрывающегося списка, в котором перечислены отдельные свойства, такие как padding-left, padding-top и др. Напротив каждого свойства стоит галочка, позволяющая быстро отключать свойство. Добавить новое правило можно с помощью меню скрытым за шестерёнкой в правом верхнем углу панели (рис. 1.56).

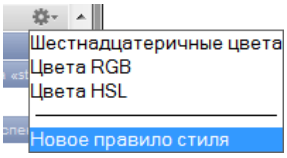


Рис. 1.56. Меню для добавления новых правил

Пункт «Размеры» демонстрирует блочную модель элемента — ширину и высоту элементов, а также значения отступов, границ, полей и позиционирования (рис. 1.57).

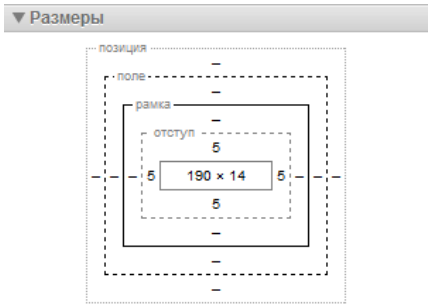


Рис. 1.57. Размеры элемента

Только надо учитывать, что отступами здесь называются свойства padding, а полями margin, т.е. наоборот, а не в привычном понимании.

Данный пункт можно применять не только для определения нужных значений, но и для их установки. Достаточно дважды щёлкнуть по нужному значению и ввести число. Указывать единицы измерения не нужно, по умолчанию принимаются пикселы.

Opera Dragonfly

Этот инструмент веб-разработчика встроен в браузер Opera и не требует отдельной установки. Вызывается через меню Инструменты > Дополнительно > Opera Dragonfly или комбинацией клавиш Ctrl+Shift+I. На рис. 1.58 приведён интерфейс Dragonfly.

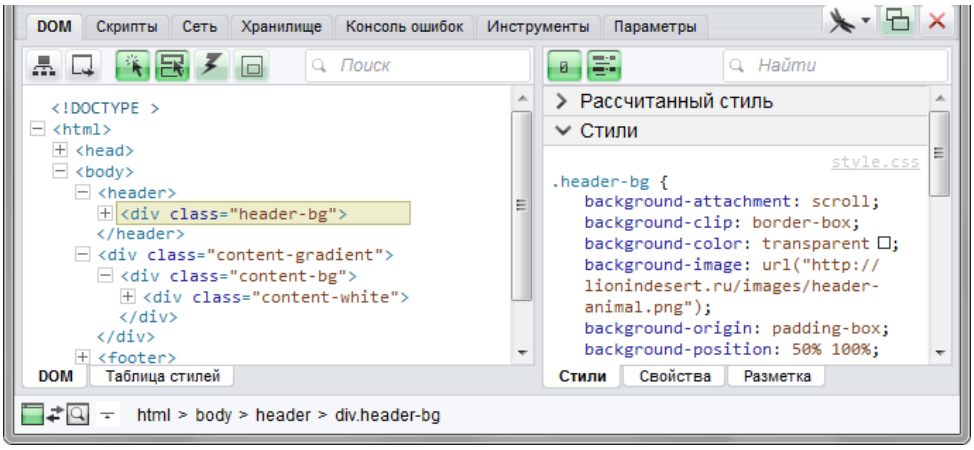


Рис. 1.58. Интерфейс Opera Dragonfly

Стрекоза, как переводится Dragonfly, по своим возможностям уступает Firebug и веб-инспектору Safari, но необходим, когда требуется отладить код под Оперу. Чтобы не повторяться, изучим Dragonfly на каком-нибудь конкретном примере и посмотрим на его возможности.

В примере 10.9 используется таблица с фоновыми рисунками для создания декоративной линии на всю ширину окна. Код HTML и CSS проверен, не содержит ошибок, но браузеры отображают пример неоднозначно.

Пример 10.9. Таблица с рисунками

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Таблица</title>
  <style type="text/css">
   table { border-collapse: collapse; width: 100%; }
   table td { border-spacing: 0; padding: 0; }
   .top { height: 20px; background: url(images/top.gif) repeat-x; }
   .topright { width: 30px;
    background: url(images/topright.gif) no-repeat; }
   .center { background: #c5b39f; }
   .centerright { width: 30px;
    background: url(images/center.gif) no-repeat; }
   .bottom { height: 20px;
    background: url(images/bottom.gif) repeat-x; }
   .bottomright { width:30px;
    background: url(images/bottomright.gif) no-repeat;
   }
  </style>
 </head>
 <body>
  <table class="tbl">
   <tr>
    <td class="top"></td><td class="topright"></td>
   </tr>
   <tr>
    <td class="center"></td><td class="centerright"></td>
   </tr>
   <tr>
    <td class="bottom"></td><td class="bottomright"></td>
   </tr>
  </table>
 </body>
</html>
```

В браузере Opera фон таблицы отображается не на всю ширину (рис. 1.59), из-за чего теряется всяческий смысл затеи.

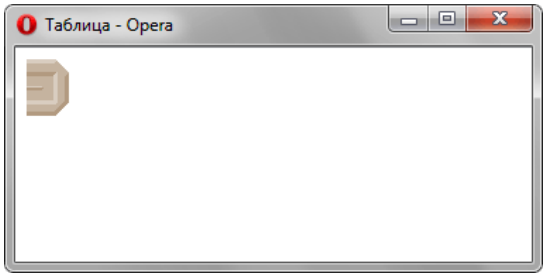


Рис. 1.59. Отображение таблицы в Опере

Открываем Dragonfly и проверяем, чтобы две кнопки на панели подсвечены:

 — найти элемент на странице по щелчку мыши.

 — подсветить элемент при наведении на него курсора.

Теперь выделяем таблицу, судя по подсвечивающемуся контуру, она занимает всю ширину страницы, это же подтверждает вкладка разметка, где приведена ширина и высота ячейки `topright` (рис. 10.60).

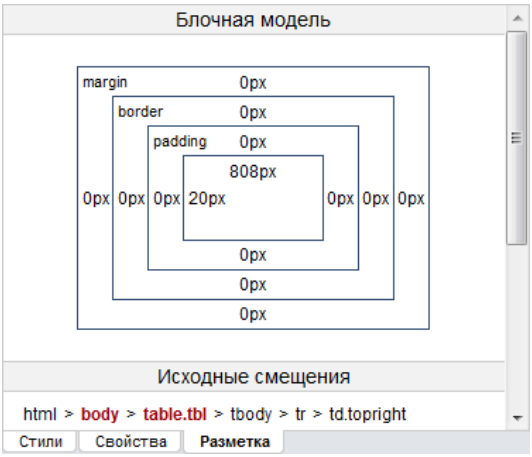


Рис. 10.60. Блочная модель

Похоже, проблема не в таблице, а в центральной ячейке center. Смотрим вкладку Разметка, так и есть, все размеры по нулям. Поэтому добавляем для этой ячейки ширину 100%, но при этом исчезнет правый фон нашей конструкции, так что таблицу надо ограничить через table-layout со значением fixed. Добавить во вкладке Стили новое свойство нельзя, но можно пойти на хитрость и отредактировать элемент, добавив ему style с заданными свойствами. В примере 10.10 показан стиль, который необходимо включить.

Пример 10.10. Добавление в стили

```
table { table-layout: fixed; }
.center { width: 100%; }
```

Проверяем в других браузерах. Аллилуйя, всё работает!

Средства разработчика Internet Explorer

Сам по себе инструмент не представляет интереса, поскольку сильно проигрывает аналогичным средствам в других браузерах. Но у него есть весомое преимущество, ради которого его следует изучить и применять в деле — возможность просматривать и тестировать сайт в разных версиях IE — 7.0, 8.0 и 9.0.

Вызов инструмента происходит через меню Сервис > Средства разработчика или клавишей F12. Интерфейс (рис. 10.61), который стал практически стандартом, позаимствован у Firebug — в левой панели отображается код HTML, в правой панели — стили выбранного элемента.

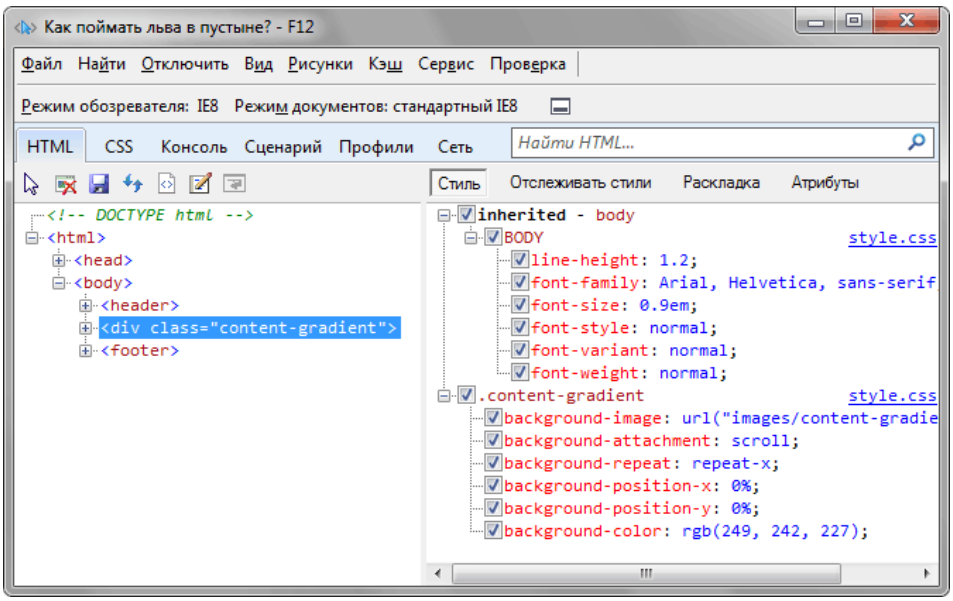


Рис. 10.61. Интерфейс средств разработчика

Необходимые для отладки возможности инструмента исследуем на примере простого списка (пример 10.11).

Пример 10.11. Список

XHTML 1.0 | CSS 2.1 | IE 7 | IE 8 | IE 9 | Cr 8 | Op 11 | Sa 5 | Fx 3.6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  <html xmlns="http://www.w3.org/1999/xhtml">
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>Список</title>
  <style type="text/css">
   ul {
    list-style: url(images/bullet.png); /* Маркеры */
   }
   ul ul {
    list-style: circle; /* Маркеры */
    margin-bottom: 1em; /* Отступ снизу */
    padding-left: 20px; /* Расстояние от маркера до текста */
    margin-left: 0; /* Отступ слева */
   }
   ul li {
    padding-left: 10px; /* Отступ слева */
   }
  </style>
 </head>
 <body>
  <ul>
   <li>Латунь и сплавы
    <ul>
     <li>Л63ПТ</li><li>ЛС59-1ПР</li><li>ЛС59-1ПТ</li><li>ЛС59-1Т</li>
    </ul>
   </li>
   <li>Медь и сплавы
    <ul>
     <li>М1М</li><li>М1Т</li><li>М2М</li><li>М2ПР</li>
```

```
      <li>M2T</li><li>M3T</li>
     </ul>
    </li>
   </ul>
  </body>
 </html>
```

Вначале соблюдаем все формальности — проверяем HTML и CSS на валидность, дабы убедиться, что никаких ошибок не совершено и с этой стороны всё в порядке. Далее смотрим в браузерах ориентированных на стандарт — Firefox, Safari или Chrome. После того, как эти этапы соблюдены, можно переходить к отладке кода для IE. Открываем Средства разработчика и в меню Режим документов переключаем на разные стандарты, чтобы посмотреть, как сайт будет выглядеть в разных версиях IE. В IE7 и IE8 наблюдается одна и та же картина: для вложенного списка выводятся картинки (рис. 10.62), тогда как в качестве маркера должен отображаться кружок.

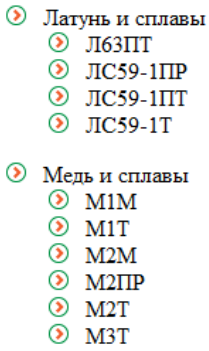


Рис. 10.62. Неверное отображение маркеров в IE7–8

Выбираем инструмент и щёлкаем им по вложенному списку. В панели Стиль (рис. 10.63) показан стиль элемента `<ul>`.

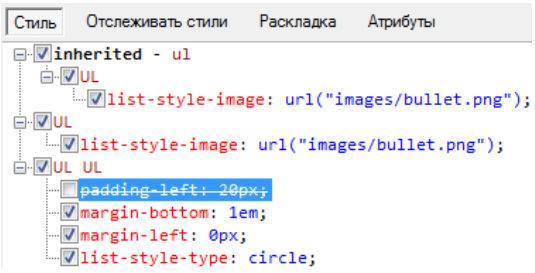


Рис. 10.63. Панель Стиль

Ключевое слово «inherited» означает, что стиль наследуется от `<ul>`. На галочку возле селектора и свойства можно щёлкнуть и тем самым выключить или включить стиль. Если щёлкнуть на свойство или его значение, тогда можно ввести собственный текст (рис. 10.64).

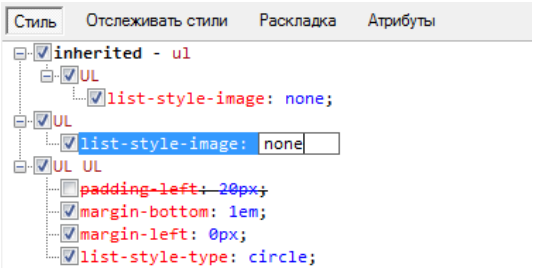


Рис. 10.64. Редактирование значения

В панели Отслеживать стили группирование происходит не по селекторам, а по стилевым свойствам (рис. 10.65), т.е. здесь показано, какие свойства каким селекторам соответствуют.

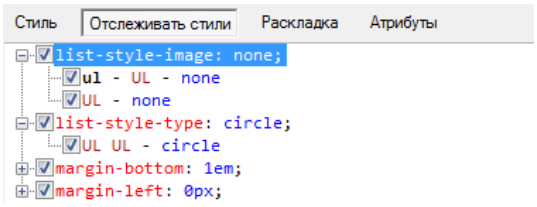


Рис. 10.65. Панель Отслеживать стили

В панели Раскладка (рис. 10.66) показана блочная модель выбранного элемента — его ширина и высота, значение полей, границ и отступов. Offset указывает положение от верхнего и левого края родителя, координаты показывают положение курсора мыши, а Z-index — значение свойства z-index. Любые числа вводятся прямо на схеме, но не все из них применяются на практике.

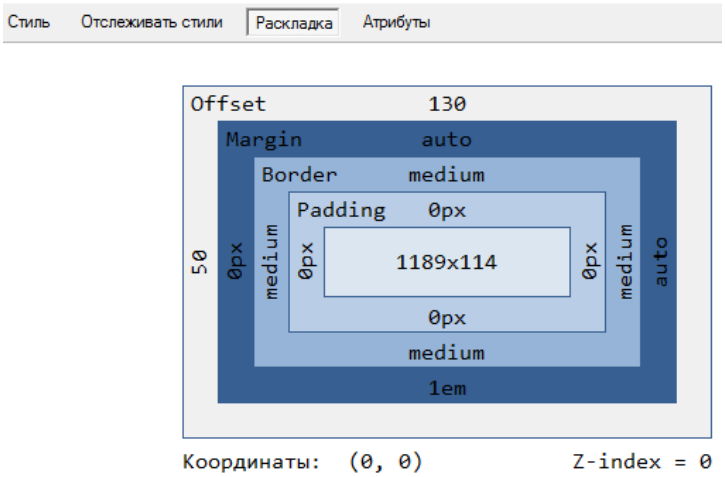


Рис. 10.66. Панель Раскладка

Вернёмся к нашему примеру. В панели Стиль (рис. 10.63) видно, что на список действует два правила: list-style-image и list-style-type. Свойство list-style-type должно отменять действие list-style-image, к тому же добавлено к селектору `ul ul`, у которого специфичность выше, чем у `ul`. Однако IE7 считает list-style-image приоритетным и включает его для вложенного списка, поэтому его надо переопределить. Чтобы добавить новое свойство переходим в панель CSS, находим нужный селектор и щёлкаем по нему правой кнопкой мыши. В контекстном меню выбираем пункт «Добавить атрибут», как показано на рис. 10.67.

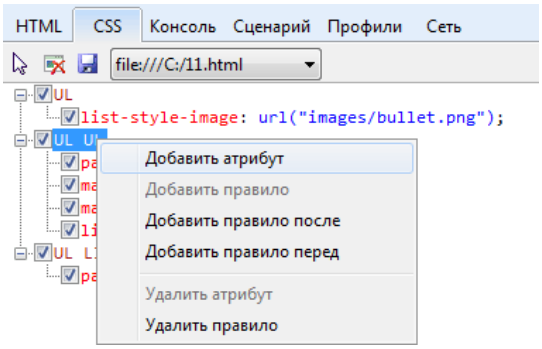


Рис. 10.67. Добавление нового свойства

Вводим list-style-image и его значение none (рис. 10.68) и наблюдаем результат в окне просмотра.

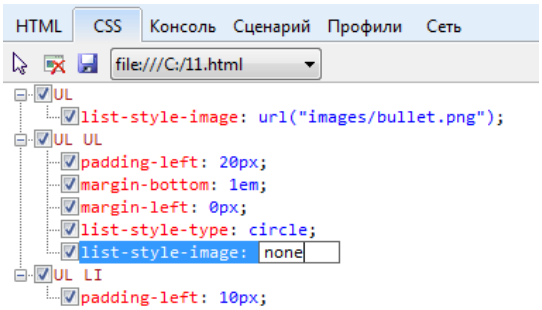


Рис. 10.68. Добавление нового свойства

К сожалению, при переключении между режимами браузера все наши сделанные изменения обнуляются, поэтому необходимо внести правки непосредственно в исходный файл. Либо воспользоваться кнопкой на панели и сохранить стиль. Впрочем, результат вам может не понравиться (пример 10.12).

Пример 10.12. Результат сохранения стилей в IE

```
/* Создано средствами разработчика F12. Это может не быть точным
   представлением файла исходного источника */
UL {
LIST-STYLE-IMAGE: url(images/bullet.png)
}
UL UL {
LIST-STYLE-TYPE: circle; PADDING-LEFT: 20px; MARGIN-BOTTOM: 1em;
MARGIN-LEFT: 0px; LIST-STYLE-IMAGE: none
}
UL LI {
PADDING-LEFT: 10px
}
```

Термины

Чтобы вам было проще ориентироваться в терминологии веб-разработчиков, здесь собраны некоторые популярные термины с их описанием.

Абсолютный адрес	Полный адрес документа с указанием протокола Интернет (http, ftp и др.). Например: http://htmlbook.ru/help, http://htmlbook.ru/example/help.png.
Атрибут	Параметр тега, расширяющий его возможности. Позволяет гибко управлять и настраивать характеристики добавляемых на страницу элементов.
Битая ссылка	Ссылка, которая указывает на несуществующий документ (файл) или просто написанная с ошибкой.
Браузер	Программа, предназначенная для просмотра сайтов и навигации по ним. К популярным браузерам относится Internet Explorer, Firefox, Opera, Safari, Chrome.
Веб-сервер	Программа, в работу которой входит анализ приходящих запросов, формирование и отдача документов пользователю.
Веб-страница	Отдельный документ сайта в одном из популярных форматах: HTML, XML, PDF и др.
Относительный адрес	Путь к файлу относительно текущего документа.
Сайт	Совокупность веб-страниц, объединенных дизайном, содержанием и единым адресом.
Свойство	Стилевой атрибут, который используется для добавления определенного стиля к элементам веб-страницы.
Селектор	Имя стиля, для которого добавляются параметры форматирования. В качестве селектора выступают теги, классы и идентификаторы.
Скриншот	Копия экрана или текущего окна в виде изображения.
Спецификация	Свод правил по работе с HTML, CSS и другими веб-технологиями. Продвигается организацией Веб Консорциум (WWW Consortium, W3C), в задачу которого входит разработка и внедрение единых веб-стандартов.
Ссылка	Текст или изображение, ведущее на другой документ. Иногда используется слово «гиперссылка», когда из контекста не понятно, идет ли речь о ссылке в Сибирь или ссылке на веб-страницу.
Протокол	Набор соглашений для передачи данных по сети. В Интернете наиболее распространен HTTP (Hyper Text Transfer Protocol, протокол передачи гипертекста).
Псевдокласс	Стилевой атрибут, определяющий стиль элемента, у которого

	состояние меняется от действий пользователя. Также псевдоклассы отслеживают положение в дереве документа.
Псевдоэлемент	Стилевой атрибут, позволяющий задать стиль элементов не определенных в дереве элементов документа, а также генерировать содержимое, которого нет в исходном коде текста.
Фрейм	Технология HTML, позволяющая разбить окно браузера на отдельные области, расположенные рядом друг с другом.
CSS (Cascading Style Sheets, каскадные таблицы стилей)	Набор правил форматирования, который применяется к элементам документа для изменения их внешнего вида и положения в документе.
URL (Universal resource locator, универсальный указатель ресурсов)	Адрес файла или документа.

Жаргонизмы

В любой профессиональной среде, включая разработчиков сайтов, имеется набор жаргонных слов, которые непонятны в отрыве от контекста или людям не знакомым со спецификой области. Хотя в книге такие слова встречаются крайне редко, на форуме или конференциях веб-разработчиков жаргоны можно услышать сплошь и рядом. Чтобы вам было легче ориентироваться в терминологии, составлен словарь жаргона и принятых сокрашений. В основном, такие слова являются транскрипцией англоязычных терминов.

IE (MSIE, ИЕ, осел, недобраузер)	Браузер Internet Explorer.
Баг	Ошибка. Слово произошло якобы от жука, который залез в один из первых компьютеров и привел к его сбою. Фиксить баги — исправлять ошибки.
Бато́н	Кнопка.
Див	Тег .
Доктайп	Элемент <!DOCTYPE>.
Жаба	Язык программирования Java.
Жаваскрипт	Язык программирования JavaScript.
Кликать по батону	Щелкать по кнопке.
Контент	Содержимое сайта.
Линк (анкор)	Ссылка.
Лиса	Браузер Mozilla Firefox.

(огнелиса, лисичка, FF, Fx)	
Морда	Главная страница сайта.
Опера (Op)	Браузер Opera.
Подвал	Нижняя часть страницы. Обычно содержит информацию об авторских правах и счетчики посещений.
Радио	Флажок (элемент формы).
Ролловер	Эффект перекатывания, т.е. смена одного изображения другим при наведении на него курсора мыши.
Рыба	Текстовая информация, которая не несет реального смысла. Необходима для того, чтобы показать заказчику, как будет выглядеть страница, наполненная информацией.
Сайдбар	Колонка навигации слева или справа от контента.
Сафари (Sa)	Браузер Apple Safari.
Скрол	Полоса прокрутки.
Спека	Спецификация.
Сырцы (сорс)	Исходный код страницы.
Урл	URL (Uniform Resource Locator — единообразный указатель ресурсов).
Фаербаг	Нарицательное слово для инструментов разработчика.
Хак	Прием, направленный на определенный браузер, чтобы задать ему стиль, отличный от других браузеров.
Хекс	HEX (шестнадцатеричное значение, используется обычно для цвета).
Хром (Cr)	Браузер Google Chrome.
Хтэмыль (ХТМЛ)	HTML
Чекбокс	Переключатель (элемент формы).
Шапка	Верхняя часть страницы. Содержит, как правило, заголовок сайта и навигацию по нему.