

Самоучитель HTML

Авторские права

Самоучитель HTML (далее произведение) распространяется на условиях следующей лицензии Creative Commons: «Указание авторства — Некоммерческое использование — Без производных» (Attribution-NonCommercial-NoDerivs 2.5 Generic (CC BY-NC-ND 2.5)).

Вы можете свободно



— копировать, распространять и передавать данное произведение.

На следующих условиях



— обязательно указывать автора произведения (Влад Мержевич).



— нельзя использовать произведение для заработка.



— запрещается изменять произведение или создавать другие с его помощью.

Обозначения

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

`` — тег.

`align` — атрибут тега, ключевое слово или выделение.

`right` — значение.

`File > Open` — пункт меню указанной программы.

`Tab` — клавиша на клавиатуре.



Этот символ призван привлечь внимание к некоторому замечанию по тексту.

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.

Введение в HTML

Быстрый старт

Чтобы сразу же ввести в курс дела нетерпеливых читателей, предложим им возможность создания веб-страницы без последовательного изучения правил HTML. По крайней мере, вы сумеете убедиться, что создание веб-страниц достаточно простая штука.

В примере 1.1 приведен несложный пример такого кода.

Пример 1.1. Первая веб-страница

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>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Моя первая веб-страница</title>
</head>
<body>

  <h1>Заголовок страницы</h1>
  <p>Основной текст.</p>

</body>
</html>
```

Чтобы посмотреть результат примера в действии, проделайте следующие шаги.

1. В Windows откройте программу Блокнот (Пуск > Выполнить > набрать «notepad» или Пуск > Программы > Стандартные > Блокнот).
2. Наберите или скопируйте код в Блокноте (рис. 1.1).

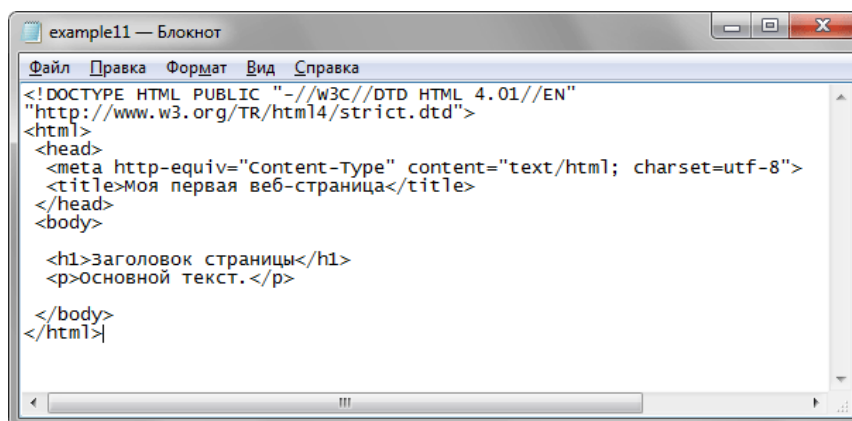


Рис. 1.1. Вид HTML-кода в программе Блокнот

3. Сохраните готовый документ (Файл > Сохранить как...) под именем `c:\www\example11.html`, при этом обязательно поставьте в диалоговом окне сохранения тип файла: Все файлы и кодировку UTF-8 (рис. 1.2). Обратите внимание, что расширение у файла должно быть именно html.

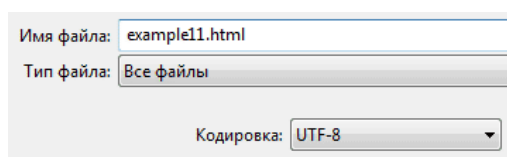


Рис. 1.2. Параметры сохранения файла в Блокноте

4. Запустите браузер Internet Explorer (Пуск > Выполнить > набрать «iexplore» или Пуск > Программы > Internet Explorer).
5. В браузере выберите пункт меню Файл > Открыть и укажите путь к вашему файлу.
6. Если все сделано правильно, то в браузере вы увидите результат, как показано на рис. 1.3.

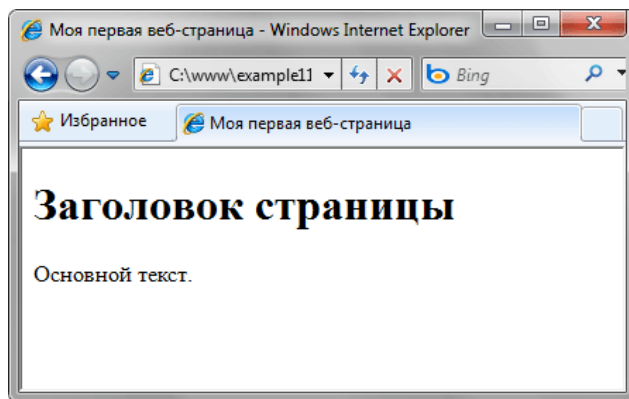


Рис. 1.3. Вид веб-страницы в окне браузера

В случае возникновения каких-либо ошибок проверьте правильность набора кода согласно примеру 1.1, расширение файла (должно быть html) и путь к документу.

Инструментарий

Для эффективной работы не обойтись без необходимых и привычных инструментов, в том числе и при написании кода HTML. Поэтому для начальной разработки веб-страниц или даже небольшого сайта — так называется набор страниц, связанных между собой ссылками и единым оформлением, нам понадобятся следующие программы.

- Текстовый редактор.
- Браузер для просмотра результатов.
- Валидатор — программа для проверки синтаксиса HTML и выявления ошибок в коде.
- Графический редактор.
- Справочник по тегам HTML.

Далее рассмотрим эти инструменты подробнее.

Текстовый редактор

HTML-документ можно создавать в любом текстовом редакторе, хоть Блокноте, тем не менее, для этой цели подойдет не всякая программа. Нужна такая, чтобы поддерживала следующие возможности:

- подсветка синтаксиса — выделение тегов, текста, ключевых слов и параметров разными цветами. Это облегчает поиск нужного элемента, ускоряет работу разработчика и снижает возникновение ошибок;
- работа с вкладками. Сайт представляет собой набор файлов, которые приходится править по отдельности, для чего нужен редактор, умеющий одновременно работать сразу с несколькими документами. При этом файлы удобно открывать в отдельных вкладках, чтобы быстро переходить к нужному документу;
- проверка текущего документа на ошибки.

Ссылки на некоторые подобные редакторы приведены ниже.

PSPad

<http://www.pspad.com/ru/download.php>

HtmlReader

<http://manticora.ru/download.htm>

Notepad++

<http://notepad-plus.sourceforge.net/ru/site.htm>

EditPlus

<http://www.editplus.com>

Браузер

Браузер это программа, предназначенная для просмотра веб-страниц. На первых порах подойдет любой браузер, но с повышением опыта и знаний потребуется завести целый «зверинец», чтобы проверять правильность отображения сайта в разных браузерах. Дело в том, что каждый браузер имеет свои уникальные особенности, поэтому для проверки универсальности кода требуется просматривать и корректировать код с их учетом. На сегодняшний день наибольшей популярностью пользуются три браузера: Firefox, Internet Explorer и Opera.

Mozilla Firefox

Перспективный и развивающийся браузер, получивший признание во всем мире. Его особенность — простота и расширяемость, которая получается за счет специальных расширений, как они называются. Изначально Firefox имеет набор только самых необходимых функций, но, устанавливая желаемые расширения, в итоге можно нарастить браузер до системы, выполняющей все необходимые для вашей работы действия. Браузер Firefox является открытой системой, разрабатываемый группой Mozilla.

Где скачать

<http://www.mozilla.ru/products/firefox/>

Microsoft Internet Explorer (IE)

Один из старейших браузеров, который бесплатно поставляется вместе с операционной системой Windows. Это и определило его популярность. Версия IE 7 по удобству приблизилась к своим давним конкурентам, в частности, появились вкладки. К сожалению, этот браузер хуже всех поддерживает спецификацию HTML, поэтому для корректного отображения в IE приходится порой отдельно отлаживать код специально под него.

Где скачать

<http://www.microsoft.com/rus/windows/ie/default.msp>

Opera

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

Где скачать

<http://ru.opera.com/download/>

Safari

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

Где скачать

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

Google Chrome

Самый свежий браузер, появившийся на рынке в конце 2008 году. Разработан компанией Google.

Где скачать

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

Валидатор

Валидация HTML-документа предназначена для выявления ошибок в синтаксисе веб-страницы и расхождений со спецификацией HTML. Соответственно, программа или система для такой проверки называется валидатором.

Как проверить HTML-файл на валидность

Если есть доступ в Интернет, то следует зайти по адресу <http://validator.w3.org> и ввести путь к проверяемому документу или сайту в специальной форме. После проверки будут показаны возможные ошибки или появится надпись, что документ прошел валидацию успешно.

Tidy

Для проверки локального HTML-файла или при отсутствии подключения к Интернету, предназначена программа Tidy. Некоторые редакторы, например, PSPad, уже содержат встроенный Tidy и валидацию документа можно провести без дополнительных средств.

Где скачать

<http://tidy.sourceforge.net>

Графический редактор

Графический редактор необходим для обработки изображений и их подготовки для публикации на веб-странице. Самой популярной программой такого рода является Photoshop, ставший стандартом для обработки фотографий и создания графических изображений для сайтов. Но в большинстве случаев мощь Photoshop-а избыточна, и лучше воспользоваться чем-нибудь более простым и проворным. В частности, программа Paint.Net позволяет сделать все необходимые манипуляции с изображениями, вдобавок бесплатна для использования.

Скачать Paint.Net

<http://www.getpaint.net/download.html>

Справочник по тегам HTML

Запоминать все теги и их параметры наизусть на первых порах сложно, поэтому требуется периодически заглядывать в руководство, чтобы уточнить тот или иной вопрос. Вообще, хороший справочник нужен всем, независимо от уровня подготовки.

Справочники в Интернете

Описание тегов HTML (на английском языке)

<http://www.w3.org/TR/html4/index/elements.html>

На этом сайте вы также найдете один из лучших справочников по тегам в Рунете.

Теги

Чтобы браузер при отображении документа понимал, что имеет дело не с простым текстом, а с элементом форматирования и применяются теги. Общий синтаксис написания тегов следующий.

```
<тег атрибут1="значение" атрибут2="значение">  
<тег атрибут1="значение" атрибут2="значение">...</тег>
```

Как видно из данного примера, теги бывают двух типов — одиночные и парные (контейнеры). Одиночный тег используется самостоятельно, а парный может включать внутри себя другие теги или текст. У тегов допустимы различные атрибуты, которые разделяются между собой пробелом. Впрочем, есть теги без всяких дополнительных атрибутов. Условно атрибуты можно подразделить на обязательные, они непременно должны присутствовать, и необязательные, их добавление зависит от цели применения тега.

В примере 3.1 показан типичный HTML-документ с тегами и текстом.

Пример 3.1. Теги в документе

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>  
<meta http-equiv="content-type" content="text/html; charset=utf-8">  
<title>Lorem ipsum</title>  
</head>  
<body>  
<p>Lorem ipsum dolor sit amet consectetur cursus pede pellentesque vitae  
pretium. Tristique mus at elit lobortis libero Sed vestibulum ut eleifend habitasse.  
Quis Nam Mauris adipiscing Integer ligula dictum sed at enim urna. Et scelerisque  
id et nibh dui tincidunt Curabitur faucibus elit massa. Tincidunt et gravida  
Phasellus eget parturient faucibus tellus at justo sollicitudin. Mi nulla ut  
adipiscing.</p>  
</body>  
</html>
```

В данном примере используется одиночный тег `<meta>`, а парных тегов сразу несколько: `<html>`, `<head>`, `<title>`, `<body>` и `<p>`.

Парные теги

Парные теги, называемые по-другому контейнеры, состоят из двух частей — открывающий и закрывающий тег. Открывающий тег обозначается как и одиночный — `<тег>`, а в закрывающем используется слэш — `</тег>`. Допускается вкладывать в контейнер другие теги, однако следует соблюдать их порядок. Так, на рис. 3.1 демонстрируется, как можно и нельзя добавлять один контейнер внутрь другого.

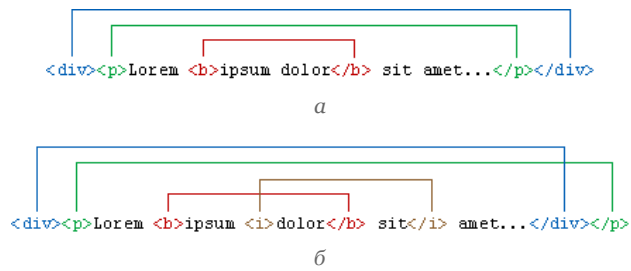


Рис. 3.1. Вложение тегов, а — правильное, б — неверное

Если связать открывающий и закрывающий тег между собой скобкой, как показано на рис. 3.1, то несколько скобок обозначающих разные контейнеры, не должны пересекаться между собой (рис. 3.1а). Любое пересечение условных скобок (рис. 3.1б) говорит о том, что правильная последовательность тегов нарушена.



Не все контейнеры требуют обязательно закрывающего тега, иногда его можно и опустить. Тем не менее, закрывайте все требуемые теги, так вы приучитесь сводить к нулю возможные ошибки.

Правила применения тегов

Для тегов любого типа действуют определенные правила их использования. Причем, некоторые правила обязательны для выполнения, а другие являются рекомендациями, т.е. их можно выполнять, а можно и нет.

Атрибуты тегов и кавычки

Согласно спецификации HTML все значения атрибутов тегов следует указывать в двойных ("пример") или одинарных кавычках ('пример'). Отсутствие кавычек не приведет к ошибкам, браузеры во многих случаях достаточно корректно обрабатывают код и без кавычек, за исключением текста, содержащего пробелы (пример 3.2).

Пример 3.2. Использование кавычек в атрибутах тегов

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>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>Кавычки в атрибуте alt</title>
</head>
<body>
<p></p>
<p><img src='images/arena.png' alt=Вид заголовка width="400" height="101"></p>
</body>
</html>
```

В данном примере строка 8 написана правильно, со всеми кавычками, а в строке 9 у атрибута `alt` кавычки отсутствуют. Из-за этого браузер в качестве значения `alt` возьмет только первое слово («Вид»), а слово «заголовка» будет воспринято как ошибочное значение. Поэтому всегда приучайтесь указывать атрибуты тегов в кавычках.

Теги можно писать как прописными, так и строчными символами

Любые теги, а также их атрибуты нечувствительны к регистру, поэтому форму записи вы вольны выбирать сами, как писать — `
`, `
` или `
`. В любом случае рекомендуется придерживаться выбранной формы записи на протяжении всех страниц сайта. Заметим также, что текст, полностью набранный прописными символами, читается хуже, чем текст со строчными символами или смешанный.

Переносы строк

Внутри тега между его атрибутами допустимо ставить перенос строк. В примере 3.3 показана одна и та же строка, но оформленная разными способами.

Пример 3.3. Переносы строк в коде тега

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>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>Кавычки в атрибуте alt</title>
</head>
<body>
<p></p>
<p></p>
</body>
</html>
```

В данном примере первый тег `` набран в одну строку, включая все его атрибуты, а второй тег `` разбит на несколько строк.



Хотя ошибки при переносе текста в подобном случае и не возникнет, рекомендуем писать теги в одну строку, иначе ухудшается восприятие кода и его становится сложнее править.

Неизвестные теги и атрибуты

Если какой-либо тег или его атрибут был написан неверно, то браузер проигнорирует подобный тег и будет отображать текст так, словно тега и не было. Опять же, следует избегать неизвестных тегов, поскольку код HTML не пройдет валидацию.

Порядок тегов

Существует определенная иерархия вложенности тегов. Например, тег `<title>` должен находиться внутри контейнера `<head>` и нигде иначе. Чтобы не возникло ошибки, следите за тем, чтобы теги располагались в коде правильно.

Если теги между собой равноценны в иерархии связи, то их последовательность не имеет значения. Так, можно поменять местами теги `<title>` и `<meta>`, на конечном результате это никак не скажется.

Закрывают все теги

Существует три состояния закрывающего тега: обязательен, не требуется или не обязательен. Обязательный закрывающий тег должен присутствовать всегда, иначе это приведет к ошибке при отображении документа. Для некоторых тегов вроде `
` закрывающего тега нет в принципе. Необязательный закрывающий тег говорит о том, что разработчик может его как добавить, так и опустить, к ошибке это не приведет. Однако рекомендуем закрывать все подобные теги, включая необязательные, это дисциплинирует, создает более стройный и строгий код, который легко модифицировать.

Атрибуты тегов

Чтобы расширить возможности отдельных тегов и более гибко управлять содержимым контейнеров и применяются атрибуты тегов.

Для атрибутов тегов используются значения по умолчанию

Когда для тега не добавлен какой-либо допустимый атрибут, это означает, что браузер в этом случае будет подставлять значение, заданное по умолчанию. Если вы ожидали получить иной результат на веб-странице, проверьте, возможно, следует явно указать значения некоторых атрибутов.

Атрибуты без значений

Допустимо использовать некоторые атрибуты у тегов, не присваивая им никакого значения, как показано в примере 3.4.

Пример 3.4. Атрибуты без значений

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>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Добавление формы</title>
</head>
<body>
  <form action="self.php">
    <p><input type="text"></p>
    <p><input type="submit" disabled></p>
  </form>
</body>
</html>
```

В данном примере используется атрибут `disabled`, у которого явно не задано значение. Подобная запись называется «сокращенный атрибут тега».

Порядок атрибутов в тегах

Порядок атрибутов в любом теге не имеет значения и на результат отображения элемента не влияет. Поэтому теги вида `` и `` по своему действию равны.

Формат атрибутов

Каждый атрибут тега относится к определенному типу (например: текст, число, путь к файлу и др.), который обязательно должен учитываться при написании атрибута. Так, в примере 3.3 упоминается тег ``, он добавляет на веб-страницу рисунок, а его атрибут `width` задает ширину изображения в пикселах. Если поставить не число, а нечто другое, то значение будет проигнорировано и возникнет ошибка при валидации документа.

Структура HTML-кода

Если открыть любую веб-страницу, то она будет содержать в себе типичные элементы, которые не меняются от вида и направленности сайта. В примере 4.1 показан код простого документа, содержащего основные теги.

Пример 4.1. Исходный код веб-страницы

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>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Пример веб-страницы</title>
</head>
<body>
  <h1>Заголовок</h1>
  <!-- Комментарий -->
  <p>Первый абзац.</p>
  <p>Второй абзац.</p>
</body>
</html>
```

Скопируйте содержимое данного примера и сохраните его в папке c:\www\ под именем example41.html. После этого запустите браузер и откройте файл через пункт меню Файл > Открыть файл (Ctrl+O). В диалоговом окне выбора документа укажите файл example41.html. В браузере откроется веб-страница, показанная на рис. 4.1.

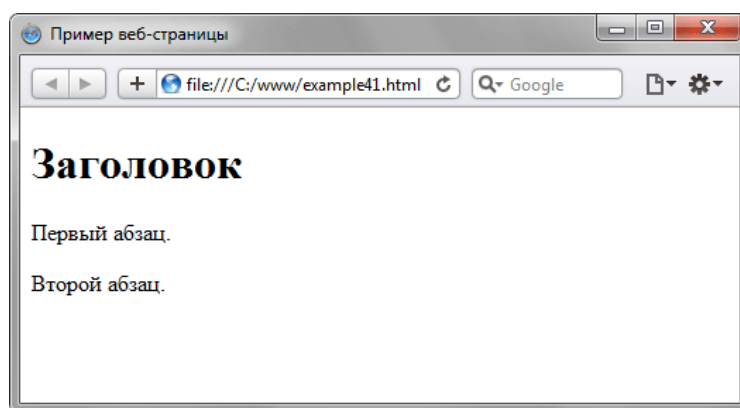


Рис. 4.1. Результат выполнения примера

Далее разберем отдельные строки нашего кода.

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

Элемент **<!DOCTYPE>** предназначен для указания типа текущего документа — DTD (document type definition, описание типа документа). Это необходимо, чтобы браузер понимал, как следует интерпретировать текущую веб-страницу, ведь HTML существует в нескольких версиях, кроме того, имеется XHTML (EXtensible HyperText Markup Language, расширенный язык разметки гипертекста), похожий на HTML, но различающийся с ним по синтаксису. Чтобы браузер «не путался» и понимал, согласно какому стандарту отображать веб-страницу и необходимо в первой строке кода задавать **<!DOCTYPE>**.

Существует несколько видов **<!DOCTYPE>**, они различаются в зависимости от версии HTML, на которую ориентированы. В табл. 4.1. приведены основные типы документов с их описанием.

Табл. 4.1. Допустимые DTD


DOCTYPE	Описание
HTML 4.01	
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">	Строгий синтаксис HTML.
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">	Переходный синтаксис HTML.
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">	В HTML-документе применяются фреймы.
HTML 5	
<!DOCTYPE html>	В этой версии HTML только один доктайп.
XHTML 1.0	
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">	Строгий синтаксис XHTML.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">	Переходный синтаксис XHTML.
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">	Документ написан на XHTML и содержит фреймы.
XHTML 1.1	
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">	Разработчики XHTML 1.1 предполагают, что он постепенно вытеснит HTML. Как видите, никакого деления на виды это определение не имеет, поскольку синтаксис один и подчиняется четким правилам.

Разница между строгим и переходным описанием документа состоит в различном подходе к написанию кода документа. Строгий HTML требует жесткого соблюдения спецификации HTML и не прощает ошибок. Переходный HTML более «спокойно» относится к некоторым огрехам кода, поэтому этот тип в определенных случаях использовать предпочтительнее.

Например, в строгом HTML и XHTML непременно требуется наличие тега `<title>`, а в переходном HTML его можно опустить и не указывать. При этом помним, что браузер в любом случае покажет документ, независимо от того, соответствует он синтаксису или нет. Подобная проверка осуществляется при помощи валидатора и предназначена в первую очередь для разработчиков, чтобы отслеживать ошибки в документе.

В дальнейшем будем применять преимущественно строгий `<!DOCTYPE>`, кроме случаев, когда это оговаривается особо. Это позволит нам избегать типичных ошибок и приучит к написанию синтаксически правильного кода.

 Часто можно встретить код HTML вообще без использования `<!DOCTYPE>`, веб-страница в подобном случае все равно будет показана. Тем не менее, может получиться, что один и тот же документ отображается в браузере по-разному при использовании `<!DOCTYPE>` и без него. Кроме того, браузеры могут по-своему показывать такие документы, в итоге страница «рассыплется», т.е. будет отображаться совсем не так, как это требуется разработчику. Чтобы не произошло подобных ситуаций, всегда добавляйте `<!DOCTYPE>` в начало документа.

```
<html>
```

Тег `<html>` определяет начало HTML-файла, внутри него хранится заголовок (`<head>`) и тело документа (`<body>`).

```
<head>
```

Заголовок документа, как еще называют блок `<head>`, может содержать текст и теги, но содержимое этого раздела не показывается напрямую на странице, за исключением контейнера `<title>`.

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

Тег `<meta>` является универсальным и добавляет целый класс возможностей, в частности, с помощью метатегов, как обобщенно называют этот тег, можно изменять кодировку страницы, добавлять ключевые слова, описание документа и многое другое. Чтобы браузер понимал, что имеет дело с кодировкой UTF-8 (Unicode transformation format, формат преобразования Юникод) и добавляется данная строка.

```
<title>Пример веб-страницы</title>
```

Тег `<title>` определяет заголовок веб-страницы, это один из важных элементов предназначенный для решения множества задач. В операционной системе Windows текст заголовка отображается в левом верхнем углу окна браузера (рис. 4.2).

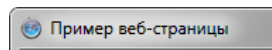



Рис. 4.2. Вид заголовка в браузере

 Тег `<title>` является обязательным и должен непременно присутствовать в коде документа.

```
</head>
```

Обязательно следует добавлять закрывающий тег `</head>`, чтобы показать, что блок заголовка документа завершен.

```
<body>
```

Тело документа `<body>` предназначено для размещения тегов и содержательной части веб-страницы.

```
<h1>Заголовок</h1>
```

HTML предлагает шесть текстовых заголовков разного уровня, которые показывают относительную важность секции,

расположенной после заголовка. Так, тег `<h1>` представляет собой наиболее важный заголовок первого уровня, а тег `<h6>` служит для обозначения заголовка шестого уровня и является наименее значительным. По умолчанию, заголовок первого уровня отображается самым крупным шрифтом жирного начертания, заголовки последующего уровня по размеру меньше. Теги `<h1>` . . . `<h6>` относятся к блочным элементам, они всегда начинаются с новой строки, а после них другие элементы отображаются на следующей строке. Кроме того, перед заголовком и после него добавляется пустое пространство.

```
<!-- Комментарий -->
```

Некоторый текст можно спрятать от показа в браузере, сделав его комментарием. Хотя такой текст пользователь не увидит, он все равно будет передаваться в документе, так что, посмотрев исходный код, можно обнаружить скрытые заметки.

Комментарии нужны для внесения в код своих записей, не влияющих на вид страницы. Начинаются они тегом `<!--` и заканчиваются тегом `-->`. Все, что находится между этими тегами отображаться на веб-странице не будет.

```
<p>Первый абзац.</p>
```

Тег `<p>` определяет абзац (параграф) текста. Если закрывающего тега нет, считается, что конец абзаца совпадает с началом следующего блочного элемента.

```
<p>Второй абзац.</p>
```

Тег `<p>` является блочным элементом, поэтому текст всегда начинается с новой строки, абзацы идущие друг за другом разделяются между собой отбивкой (так называется пустое пространство между ними). Это хорошо видно на рис. 4.1.

```
</body>
```

Следует добавить закрывающий тег `</body>`, чтобы показать, что тело документа завершено.

```
</html>
```

Последним элементом в коде всегда идет закрывающий тег `</html>`.

Типы тегов

Каждый тег HTML принадлежит к определенной группе тегов, например, табличные теги направлены на формирование таблиц и не могут применяться для других целей.

Условно теги делятся на следующие типы:

- теги верхнего уровня;
- теги заголовка документа;
- блочные элементы;
- встроенные элементы;
- универсальные элементы;
- списки;
- таблицы.

Следует учитывать, что один и тот же тег может одновременно принадлежать разным группам, например, теги `` и `` относятся к категории списков, но также являются и блочными элементами.

Далее рассмотрим только те теги, которые потребуются нам в дальнейшей работе.

Теги верхнего уровня

Эти теги предназначены для формирования структуры веб-страницы и определяют раздел заголовка и тела документа.

<html>

Тег **<html>** является контейнером, который заключает в себе всё содержимое веб-страницы, включая теги **<head>** и **<body>**. Открывающий и закрывающий теги **<html>** в документе необязательны, но хороший стиль диктует неизменное их использование.

<head>

Тег **<head>** предназначен для хранения других элементов, цель которых — помочь браузеру в работе с данными. Также внутри контейнера **<head>** находятся метатеги, которые используются для хранения информации, предназначенной для браузеров и поисковых систем. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных.

<body>

Тег **<body>** предназначен для хранения содержания веб-страницы, отображаемого в окне браузера. Информацию, которую следует выводить в документе, следует располагать именно внутри контейнера **<body>**. К такой информации относятся текст, изображения, таблицы, списки и др.

Набор тегов верхнего уровня и порядок их вложения показан в примере 5.1.

Пример 5.1. Теги верхнего уровня

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

В данном примере показано, что контейнер **<html>** определяет «каркас» всей веб-страницы, внутри него вначале задается тег **<head>**, затем идет контейнер **<body>**, в нем хранится содержательная часть документа, которая и отображается в браузере. Теги **<html>** и **<body>** хотя и не относятся к обязательным тегам (т. е. их можно не размещать в коде), все же стоит добавлять всегда. Это позволяет получить четкую и понятную структуру документа.

Заметьте, что в примере не упоминается **<!DOCTYPE>**, поскольку этот обязательный элемент кода веб-страницы не является тегом, а предназначен для браузеров, чтобы сообщить им, как интерпретировать текущий документ.

Теги заголовка документа

К этим тегам относятся элементы, которые располагаются в контейнере `<head>`. Все эти теги напрямую не отображаются в окне браузера, за исключением тега `<title>`, который определяет название веб-страницы.

`<title>`

Используется для отображения строки текста в левом верхнем углу окна браузера. Такая строка сообщает пользователю название сайта и другую информацию, которую добавляет разработчик.

`<meta>`

Метатеги используются для хранения информации, предназначенной для браузеров и поисковых систем. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных. Хотя тег `<meta>` всего один, он имеет несколько атрибутов, поэтому к нему и применяется множественное число.

Так, для краткого описания содержимого веб-страницы используется значение `description` атрибута `name`, как показано в примере 5.2.

Пример 5.2. Использование `description`

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>HTML</title>
<meta name="description" content="Сайт об HTML и создании сайтов">
<meta http-equiv="content-type" content="text/html; charset=utf-8">
</head>
<body>
<p>...</p>
</body>
</html>
```

Описание сайта, заданное с помощью тега `<meta>` и значения `description`, обычно отображается в поисковых системах или каталогах при выводе результатов поиска. Значение `keywords` также предназначено в первую очередь для повышения рейтинга сайта в поисковых системах, в нем перечисляются ключевые слова, встречаемые на веб-странице (пример 5.3).

Пример 5.3. Использование `keywords`

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>HTML</title>
<meta name="keywords" content="HTML, META, метатег, тег, поисковая система">
<meta http-equiv="content-type" content="text/html; charset=utf-8">
</head>
<body>
<p>...</p>
</body>
</html>
```

Ключевые слова можно перечислять через пробел или запятую. Поисковые системы сами приведут запись к виду, который они используют.

Блочные элементы

Блочные элементы характеризуются тем, что занимают всю доступную ширину, высота элемента определяется его содержимым, и он всегда начинается с новой строки.

<blockquote>

Предназначен для выделения длинных цитат внутри документа. Текст, обозначенный этим тегом, традиционно отображается как выровненный блок с отступами слева и справа (примерно по 40 пикселей), а также с пустым пространством сверху и снизу.

<div>

Тег **<div>** относится к универсальным блочным контейнерам и применяется в тех случаях, где нужны блочные элементы без дополнительных свойств. Также с помощью тега **<div>** можно выравнивать текст внутри этого контейнера с помощью атрибута **align**.

<h1>, ..., <h6>

Эта группа тегов определяет текстовые заголовки разного уровня, которые показывают относительную важность секции, расположенной после заголовка.

<hr>

Рисует горизонтальную линию, которая по своему виду зависит от используемых атрибутов. Линия всегда начинается с новой строки, а после нее все элементы отображаются на следующей строке.

<p>

Определяет параграф (абзац) текста.

<pre>

Задаёт блок предварительно форматированного текста. Такой текст отображается обычно моноширинным шрифтом и со всеми пробелами между словами. В HTML любое количество пробелов идущих в коде подряд на веб-странице показывается как один. Тег **<pre>** позволяет обойти эту особенность и отображать текст как требуется разработчику.

Следующие теги не должны размещаться внутри контейнера **<pre>**: **<big>**, ****, **<small>**, **<sub>** и **<sup>**.

Строчные элементы

Строчными называются такие элементы веб-страницы, которые являются непосредственной частью другого элемента, например, текстового абзаца. В основном они используются для изменения вида текста или его логического выделения.

<a>

Тег **<a>** является одним из важных элементов HTML и предназначен для создания ссылок. В зависимости от присутствия атрибутов **name** или **href** тег **<a>** устанавливает ссылку или якорь.

Определяет жирное начертание шрифта.

<big>

Тег **<big>** увеличивает размер шрифта на единицу по сравнению с обычным текстом. В HTML размер шрифта измеряется в условных единицах от 1 до 7, средний размер текста, используемый по умолчанию, принят 3. Таким образом, добавление тега **<big>** увеличивает текст на одну условную единицу.

**
**

Тег **
** устанавливает перевод строки в том месте, где этот тег находится. В отличие от тега параграфа **<p>**, использование тега **
** не добавляет пустой отступ перед строкой.

Тег **** предназначен для акцентирования текста. Браузеры отображают такой текст курсивным начертанием.

<i>

Устанавливает курсивное начертание шрифта.

Тег **** предназначен для отображения на веб-странице изображений в графическом формате GIF, JPEG или PNG. Если необходимо, то рисунок можно сделать ссылкой на другой файл, поместив тег **** в контейнер **<a>**. При этом вокруг изображения отображается рамка, которую можно убрать, добавив атрибут **border="0"** в тег ****.

<small>

Тег **<small>** уменьшает размер шрифта на единицу по сравнению с обычным текстом. По своему действию похож на тег **<big>**, но действует с точностью до наоборот.

Универсальный тег, предназначенный для определения строчного элемента внутри документа.

Тег **** предназначен для акцентирования текста. Браузеры отображают такой текст жирным начертанием.

<sub>

Отображает шрифт в виде нижнего индекса. Текст при этом располагается ниже базовой линии остальных символов строки и уменьшенного размера — H₂O.

<sup>

Отображает шрифт в виде верхнего индекса. По своему действию похож на **<sub>**, но текст отображается выше базовой линии текста — м².

Разница между блочными и строчными элементами следующая.

- Строчные элементы могут содержать только данные или другие строчные элементы, а в блочные допустимо вкладывать другие блочные элементы, строчные элементы, а также данные. Иными словами, строчные элементы никак не могут хранить блочные элементы.
- Блочные элементы всегда начинаются с новой строки, а строчные таким способом не акцентируются.
- Блочные элементы занимают всю доступную ширину, например, окна браузера, а ширина строчных элементов равна их содержимому плюс значения отступов, полей и границ.

Универсальные элементы

Особенность этих тегов состоит в том, что они в зависимости от контекста могут использоваться как блочные или встроенные элементы.

Тег **** используется для выделения текста, который был удален в новой версии документа. Подобное форматирование позволяет отследить, какие изменения в тексте документа были сделаны. Браузеры обычно помечают текст в контейнере **** как перечеркнутый.

<ins>

Тег **<ins>** предназначен для акцентирования вновь добавленного текста и обычно применяется наряду с тегом ****. Браузеры помечают содержимое контейнера **<ins>** подчеркиванием текста.

Теги для списков

Списком называется взаимосвязанный набор отдельных фраз или предложений, которые начинаются с маркера или цифры. Списки предоставляют возможность упорядочить и систематизировать разные данные и представить их в наглядном и удобном для пользователя виде.

Тег **** устанавливает нумерованный список, т.е. каждый элемент списка начинается с числа или буквы и увеличивается по нарастающей.

Устанавливает маркированный список, каждый элемент которого начинается с небольшого символа — маркера.

Тег **** определяет отдельный элемент списка. Внешний тег **** или **** устанавливает тип списка — маркированный или нумерованный.

<dd>, <dt>, <dl>

Тройка элементов предназначена для создания списка определений. Каждый такой список начинается с контейнера **<dl>**, куда входит тег **<dt>** создающий термин и тег **<dd>** задающий определение этого термина. Закрывающий тег **</dd>** не обязателен, поскольку следующий тег сообщает о завершении предыдущего элемента. Тем не менее, хорошим стилем является закрывать все теги.

Теги для таблиц

Таблица состоит из строк и столбцов ячеек, которые могут содержать текст и рисунки. Обычно таблицы используются для упорядочения и представления табличных данных.

<table>

Служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются с помощью тегов `<tr>` и `<td>`.

<td>

Предназначен для создания одной ячейки таблицы. Тег `<td>` должен размещаться внутри контейнера `<tr>`, который в свою очередь располагается внутри тега `<table>`.

<th>

Тег `<th>` предназначен для создания одной ячейки таблицы, которая обозначается как заголовочная. Текст в такой ячейке отображается браузером обычно жирным шрифтом и выравнивается по центру.

<tr>

Тег `<tr>` служит контейнером для создания строки таблицы.

Теги для фреймов

Фреймы разделяют окно браузера на отдельные области, расположенные вплотную друг к другу. В каждую из таких областей загружается самостоятельная веб-страница определяемая с помощью тега `<frame>`. С помощью фреймов веб-страница делится на два или более документа, которые обычно содержат навигацию по сайту и его контент. Механизм фреймов позволяет открывать документ в одном фрейме, по ссылке, нажатой в совершенно другом фрейме. Допустимо также использовать вложенную структуру элементов, это позволяет дробить фреймы на мелкие области.

`<frame>`

Тег `<frame>` определяет свойства отдельного фрейма, на которые делится окно браузера.

`<frameset>`

Тег `<frameset>` заменяет собой элемент `<body>` на веб-странице и формирует структуру фреймов.

`<iframe>`

Тег `<iframe>` создает плавающий фрейм, который находится внутри обычного документа, он позволяет загружать в область заданных размеров любые другие независимые документы.

Значения атрибутов тегов

Атрибуты тегов расширяют возможности самих тегов и позволяют гибко управлять различными настройками отображения элементов веб-страницы. Общее количество атрибутов достаточно велико, но их значения, как правило, можно сгруппировать по разным типам, например, задающих цвет, размер, адрес и др. Далее рассмотрим основные типы значений.

Цвет

В HTML цвет задается одним из двух путей: с помощью шестнадцатеричного кода и по названию некоторых цветов. Преимущественно используется способ, основанный на шестнадцатеричной системе исчисления, как наиболее универсальный.

Шестнадцатеричные цвета

Для задания цветов в HTML используются числа в шестнадцатеричном коде. Шестнадцатеричная система, в отличие от десятичной системы, базируется, как следует из ее названия, на числе 16. Цифры будут следующие: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Числа от 10 до 15 заменены латинскими буквами. В табл. 6.1 приведено соответствие десятичных и шестнадцатеричных чисел.

Табл. 6.1. Десятичные и шестнадцатеричные числа меньше 16

Десятичные	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Шестнадцатеричные	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Числа больше 15 в шестнадцатеричной системе образуются объединением двух чисел в одно (табл. 6.2). Например, числу 255 в десятичной системе соответствует число FF в шестнадцатеричной.

Табл. 6.2. Десятичные и шестнадцатеричные числа больше 16

Десятичные	16	17	18	19	20	21	22	23	24	25	26	27	28
Шестнадцатеричные	10	11	12	13	14	15	16	17	18	19	1A	1B	1C

Чтобы не возникало путаницы в определении системы счисления, перед шестнадцатеричным числом ставится символ решетки #, например #aa69cc. При этом регистр значения не имеет, поэтому допустимо писать #F0F0F0 или #f0f0f0.

Типичный цвет, используемый в HTML, выглядит следующим образом.

```
<body bgcolor="#fa8e47">
```

Здесь цвет фона веб-страницы задан как #FA8E47. Символ решетки # перед числом означает, что оно шестнадцатеричное. Первые две цифры (FA) определяют красную составляющую цвета, цифры с третьей по четвертую (8E) — зеленую, а последние две цифры (47) — синюю. В итоге получится такой цвет.

$$\text{FA} + \text{8E} + \text{47} = \text{FA8E47}$$

Каждый из трех цветов — красный, зеленый и синий — может принимать значения от 00 до FF, что в итоге образует 256 оттенков. Таким образом, общее количество цветов может быть $256 \times 256 \times 256 = 16.777.216$ комбинаций. Цветовая модель, основанная на красной, зеленой и синей составляющей получила название RGB (red, green, blue; красный, зеленый, синий). Эта модель аддитивная (от add — складывать), при которой сложение всех трех компонент образует белый цвет.

Чтобы легче ориентироваться в шестнадцатеричных цветах, примите во внимание некоторые правила.

- Если значения компонент цвета одинаковы (например: #D6D6D6), то получится серый оттенок. Чем больше число, тем светлее цвет, значения при этом меняются от #000000 (черный) до #FFFFFF (белый).
- Ярко-красный цвет образуется, если красный компонент сделать максимальным (FF), а остальные компоненты обнулить. Цвет со значением #FF0000 самый красный из возможных красных оттенков. Аналогично обстоит с зеленым цветом (#00FF00) и синим (#0000FF).
- Желтый цвет (FFFF00) получается смешением красного с зеленым. Это хорошо видно на цветовом круге (рис. 6.1), где представлены основные цвета (красный, зеленый, синий) и комплементарные или дополнительные. К ним относятся желтый, голубой и фиолетовый (еще называемый пурпурным). Вообще, любой цвет можно получить смешением близлежащих к нему цветов. Так, голубой (#00FFFF) получается за счет объединения синего и зеленого цвета.

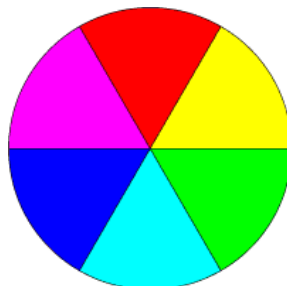


Рис. 6.1. Цветовой круг

Цвета по шестнадцатеричным значениям не обязательно подбирать эмпирическим путем. Для этой цели подойдет графический редактор, умеющий работать с разными цветовыми моделями, например, Adobe Photoshop. На рис. 6.2 показано окно для выбора цвета в этой программе, линией обведено полученное шестнадцатеричное значение текущего цвета. Его можно скопировать и вставить к себе в код.

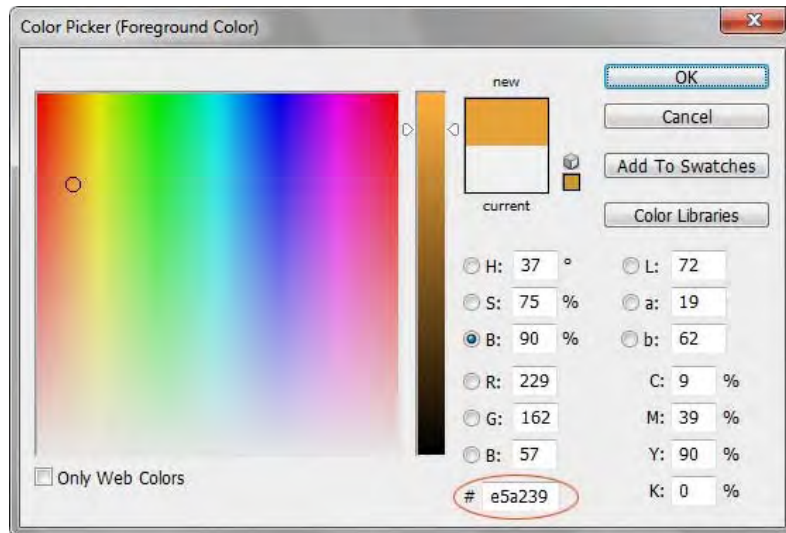


Рис. 6.2. Окно для выбора цвета в программе Photoshop

Веб-цвета

Если установить качество цветопередачи монитора в 8 бит (256 цветов), то один и тот же цвет может показываться в разных браузерах по-своему. Это связано со способом отображения графики, когда браузер работает со своей собственной палитрой и не может показать цвет, который у него в палитре отсутствует. В этом случае цвет заменяется сочетанием пикселей других, близких к нему, цветов, имитирующих заданный. Чтобы цвет оставался неизменным в разных браузерах, ввели палитру так называемых веб-цветов. Веб-цветами называются такие цвета, для каждой составляющей — красной, зеленой и синей, устанавливается одно из шести значений — 0 (00), 51 (33), 102 (66), 153 (99), 204 (CC), 255 (FF). В скобках указано шестнадцатеричное значение данной компоненты. Общее количество цветов из всех возможных сочетаний дает 6x6x6 — 216 цветов. Пример веб-цвета — #33FF66.

Основная особенность веб-цвета заключается в том, что он показывается одинаково во всех браузерах. В данный момент актуальность веб-цветов весьма мала из-за повышения качества мониторов и расширения их возможностей.

Цвета по названию

Чтобы не запоминать совокупность цифр, вместо них можно использовать имена широко используемых цветов. В табл. 6.3 приведены имена популярных названий цветов.

Табл. 6.3. Названия некоторых цветов

Имя цвета	Цвет	Описание	Шестнадцатеричное значение
black		Черный	#000000
blue		Синий	#0000FF
fuchsia		Светло-фиолетовый	#FF00FF
gray		Темно-серый	#808080
green		Зеленый	#008000
lime		Светло-зеленый	#00FF00
maroon		Темно-красный	#800000
navy		Темно-синий	#000080
olive		Оливковый	#808000
purple		Темно-фиолетовый	#800080
red		Красный	#FF0000
silver		Светло-серый	#C0C0C0
teal		Сине-зеленый	#008080
white		Белый	#FFFFFF
yellow		Желтый	#FFFF00

Не имеет значения, каким способом вы задаете цвет — по его имени или с помощью шестнадцатеричных чисел. По своему действию эти способы равны. В примере 6.1 показано, как установить цвет фона и текста веб-страницы.

Пример 6.1. Цвет фона и текста

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>
<title>Цвета</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
```

```
<body bgcolor="teal" text="#ffffff">
<p>Пример текста</p>
</body>
</html>
```

В данном примере цвет фона задается с помощью атрибута `bgcolor` тега `<body>`, а цвет текста через атрибут `text`. Для разнообразия значение у атрибута `text` установлено в виде шестнадцатеричного числа, а у `bgcolor` с помощью зарезервированного ключевого слова `teal`.

Размер

В HTML размеры элементов или расстояния между ними задаются в пикселах или процентах. Пиксел — это элементарная точка на экране монитора, является относительной единицей измерения, ее величина зависит от установленного экранного разрешения и размера монитора. Возьмем, к примеру, популярное разрешение монитора 1024x768 пикселей. Картинка с такими же размерами будет занимать всю область экрана. Увеличив разрешение монитора до 1280x1024, мы, тем самым, уменьшим размеры изображения на экране.

При использовании пикселей в качестве значений пишется только число без указания единиц, например: `width="380"`. В примере 6.2 приведено добавление изображения с заданными размерами.

Пример 6.2. Размеры изображения в пикселах

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>Изображение</title>
</head>
<body>
<p></p>
</body>
</html>
```

В данном примере рисунок имеет ширину 100 пикселей (`width="100"`), высоту 111 пикселей (`height="111"`), горизонтальный и вертикальный отступ по 4 пиксела (`hspace` и `vspace`) и толщину границы вокруг картинки 2 пиксела (`border="2"`).

Процентная запись удачно дополняет пиксели, поскольку позволяет привязаться к размерам определенного элемента, к примеру, окна браузера. Так, если задать у картинки ширину 100%, то рисунок будет заполнять все свободное пространство окна по ширине. Браузер понимает, что речь идет о процентах, если после числа добавляется символ %, например: `width="40%"`.



Размеры допустимо задавать только в целых числах. Это правило относится как к пикселям, так и процентам.

Учтите, что размер в процентах вычисляется от размеров родительского элемента, иными словами, контейнера, внутри которого располагается элемент. Если родитель явно не задан, тогда за отсчет принимается окно браузера. В примере 6.3 приведен код веб-страницы, в котором ширина элементов задается в процентах.

Пример 6.3. Размеры изображения в процентах

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>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>Изображение</title>
</head>
<body>
<p></p>
</body>
</html>
```

В данном примере ширина картинки задана как 100%, при этом высота изображения явно не задается, поскольку она вычисляется автоматически. Вид страницы при таких размерах картинки показан на рис. 6.3.

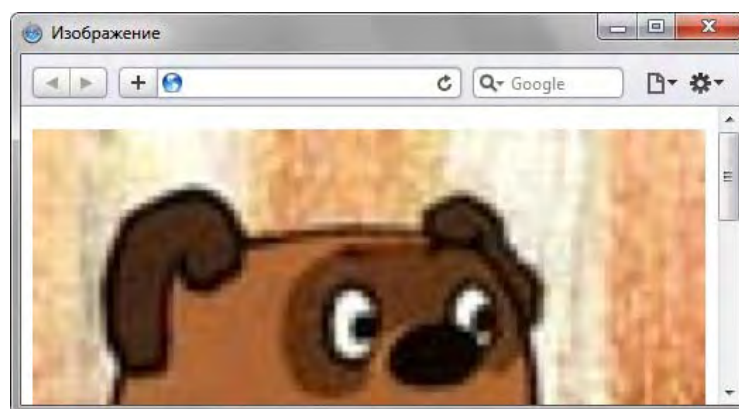


Рис. 6.3. Изображение с шириной 100%

Обратите внимание, что в изображении появляются заметные искажения, это связано с увеличением картинки вопреки ее исходным размерам.

Как вы понимаете, ширина окна принимается за 100%, но ее легко превысить, причем ненароком. В частности, стоит только добавить в примере 6.3 к тегу `` отступы по горизонтали (`hspace="10"`) и ширина изображения станет `100%+20`. Это в свою очередь приведет к появлению горизонтальной полосы прокрутки. Учитывайте этот нюанс при установке размеров

ЭЛЕМЕНТОВ.

Адрес

Адресом называется путь к документу, например, к графическому файлу. Адрес необходим в тех случаях, когда делается ссылка на веб-страницу или загружается определенный файл. Например, в теге `` адрес используется в качестве значения атрибута `src`, он задает путь к файлу с изображением.

Синонимом адреса выступает URL (Universal Resource Locator, универсальный указатель ресурсов), различают абсолютные и относительные адреса.

Абсолютные адреса

Подобные адреса работают везде и всюду независимо от имени сайта или веб-страницы, где задан URL и начинаются всегда с указания протокола передачи данных. Для веб-страниц это обычно HTTP (HyperText Transfer Protocol, протокол передачи гипертекста), соответственно, абсолютные адреса начинаются с ключевого слова `http://`. В примере 6.4 приведена ссылка, в которой применяется абсолютный адрес.

Пример 6.4. Использование абсолютного адреса в ссылке

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>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>Ссылка</title>
</head>
<body>
<p><a href="http://htmlbook.ru/html/body">Описание тега BODY</a></p>
</body>
</html>
```

В данном примере текстовая ссылка ведет на сайт `htmlbook.ru` и указывает на веб-страницу с именем `body.html`, которая располагается в каталоге `html`.

Абсолютные адреса применяются в первую очередь для указания на другой сетевой ресурс и достаточно редко используются в рамках одного сайта.

Относительные адреса

Относительные адреса указываются от корня сайта или текущего документа. Например, код `` означает загрузить графический файл с именем `pic.gif`, который располагается в той же папке, что и сама веб-страница. Далее рассмотрим несколько примеров таких адресов.

/

Адрес указывает обычно на файл `index.html`, который находится в корне сайта. Если файл `index.html` отсутствует, браузер, как правило, показывает список файлов, находящихся в данном каталоге. Имя файла не обязательно должно быть `index.html`, этот параметр меняется через настройки веб-сервера — так называется программа, которая анализирует приходящие от браузера запросы и передает ему документы, показываемые пользователю.

`/images/pic.gif`

Слэш (символ `/`) перед адресом говорит о том, что адресация начинается от корня сайта. Ссылка ведет на рисунок `pic.gif`, который находится в папке `images`. А та в свою очередь размещена в корне сайта.

`../help/me.html`

Две точки перед именем указывает браузеру перейти на уровень выше в списке каталогов сайта и там «поискать» файл `me.html`.

`manual/info.html`

Если перед именем папки нет никаких дополнительных символов, вроде точек или слэша, то папка размещена внутри текущего каталога, а уже в ней располагается файл `info.html`.



Адреса относительно корня сайта вроде `/demo/` работают только под управлением веб-сервера и на локальном компьютере не применимы.

В примере 6.5 приведены ссылки, в которых используются относительные адреса.

Пример 6.5. Относительные адреса в ссылках

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>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>Ссылки</title>
</head>
<body>
<p><a href="images/xxx.jpg">Посмотрите на мою фотографию!</a></p>
<p><a href="tip.html">Как сделать такое же фото?</a></p>
</body>
</html>
```

Иногда можно встретить в адресе ссылки путь в виде `.file/doc.html`. Точка со слэшем означает, что отсчет ведется от текущей папки. Подобная запись избыточна и ее можно сократить до `file/doc.html`.

Текст

Для изменения вида текста существует достаточно большое количество различных тегов. Это и немудрено, ведь текст самый популярный вид информации.

Особенности текста в HTML

Прежде чем редактировать код веб-страницы, следует принять во внимание некоторые особенности, которые присущи HTML при работе с текстом.

Любое количество пробелов идущих подряд, в браузере отображается как один

Сколько бы вы не поставили пробелов между словами, это никак не повлияет на конечный вид текста. Это же правило относится к символам табуляции и переносу текста. Поэтому не ставьте лишних пробелов, поскольку это лишь увеличит общий объем файла, но никак не изменит вид документа в браузере. Приведенные ниже строки будут отображаться на веб-странице одинаково, несмотря на их разное написание.

```
<p>Измеряй микрометром. Отмечай мелом. Отрубай топором.</p>
<p>Измеряй микрометром.  Отмечай мелом.  Отрубай топором.</p>
<p>Измеряй микрометром.
  Отмечай мелом.
  Отрубай топором.</p>
```

Исключением из этого правила является тег `<pre>`, внутри которого любое число пробелов отображается именно так, как оно указано в коде.

Нет расстановки переносов в тексте

HTML не поддерживает расстановку переносов в словах как это делают текстовые редакторы, иначе говоря, все слова пишутся целиком без их разбиения. Это условие несущественно, пока не используется выравнивание текста по ширине. В этом случае блок текста выравнивается по левому и правому краю. Короткие строки при этом растягиваются за счет автоматического добавления пробелов между словами. Иногда пустые блоки между словами настолько велики, что портят внешний вид страницы и ухудшают читабельность текста.

Представьте, что у вас в середине предложения есть какое-нибудь длинное слово, вот например «Дегидроэпиандростерон». В текстовом редакторе это слово будет перенесено по слогам так, чтобы текст занял указанную ширину, а на веб-странице подобное слово будет отображаться целиком, без переносов.

Текст занимает ширину окна браузера

Если вы просто напишите одну длинную строку в коде HTML, то в браузере она будет отформатирована, чтобы текст поместился по ширине в окно. Переносы текста будут добавлены автоматически в местах пробела или дефиса. Что произойдет, если в тексте нет ни того, ни другого символа? Браузер не сможет создать переносы и отобразит текст одной строкой. Если она шире окна браузера, то неминуемо появится горизонтальная полоса прокрутки.

Абзацы

Как правило, блоки текста разделяют между собой абзацами (параграфами). По умолчанию между параграфами существует небольшой вертикальный отступ, называемый отбивкой. Синтаксис создания абзацев следующий.

```
<p>Абзац 1</p>
<p>Абзац 2</p>
```

Каждый абзац начинается с тега `<p>` и должен иметь необязательный закрывающий тег `</p>`.



В любой книге для выделения следующего абзаца используется отступ первой строки, еще называемый «красная строка». Это позволяет читателю легко отыскивать взглядом новую строку и повышает, таким образом, читабельность текста. На веб-странице этот прием обычно не используется, а для разделения абзацев применяется отбивка.

В примере 7.1 показано применение абзацев для создания отступов между строками.

Пример 7.1. Использование абзацев

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>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Применение абзацев</title>
</head>
<body>
<p>В одних садах цветёт миндаль, в других метёт метель.</p>
<p>В одних краях ещё февраль, в других - уже апрель.</p>
<p>Проходит время, вечный счёт: год за год, век за век...</p>
<p>Во всём - его неспешный ход, его кромешный бег.</p>
<p>В году на радость и печаль по двадцать пять недель.</p>
<p>Мне двадцать пять недель февраль, и двадцать пять - апрель.</p>
<p>По двадцать пять недель в туман уходит счёт векам.</p>
<p>Летит мой звонкий балаган куда-то к облакам.</p>
<p><i>М. Щербаков</i></p>
</body>
</html>
```

Результат данного примера показан на рис. 7.1.

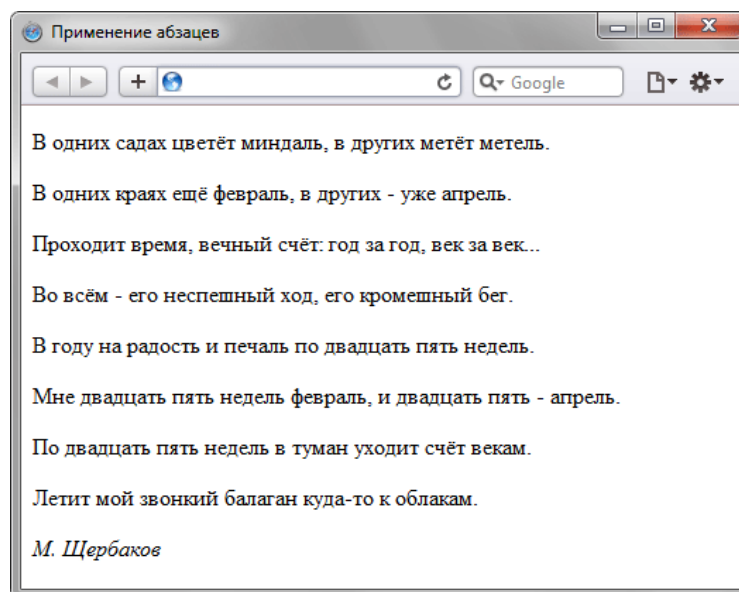


Рис. 7.1. Отступы на веб-странице при использовании абзацев

Как видно из рисунка, при использовании тега `<p>` между абзацами возникают слишком большие отступы. От них можно избавиться, если в местах переноса строк добавлять тег `
`. В отличие от абзаца, тег переноса строки `
` не создает дополнительных вертикальных отступов между строками и может применяться практически в любом тексте.

Так, текст примера 7.1 с учетом переноса строк будет преобразован следующим образом (пример 7.2).

Пример 7.2. Тег `
`

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>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Переносы в тексте</title>
</head>
<body>
<p>В одних садах цветёт миндаль, в других метёт метель.<br>
В одних краях ещё февраль, в других - уже апрель.<br>
```



```
Проходит время, вечный счёт: год за год, век за век...<br>
Во всём - его неспешный ход, его крошечный бег.<br>
В году на радость и печаль по двадцать пять недель.<br>
Мне двадцать пять недель февраль, и двадцать пять - апрель.<br>
По двадцать пять недель в туман уходит счёт векам.<br>
Летит мой звонкий балаган куда-то к облакам.</p>
<p><i>М. Щербаков</i></p>
</body>
</html>
```

Результат примера продемонстрирован на рис. 7.2. Видно, что расстояние между строками текста уменьшилось и он приобрел более компактный вид.

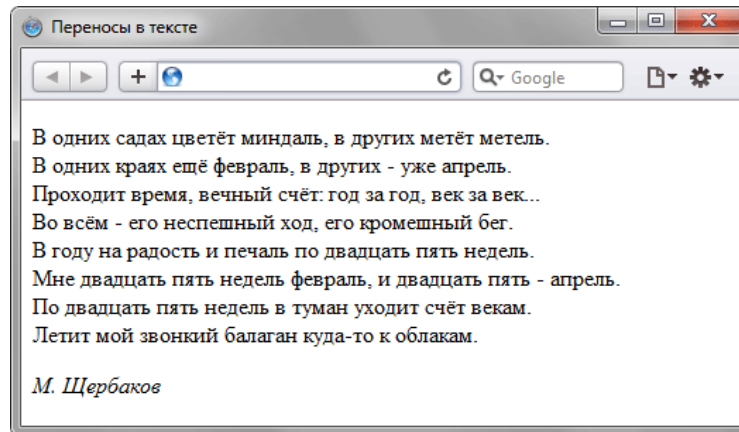


Рис. 7.2. Вид текста с учетом переносов

Заголовки

Заголовки выполняют важную функцию на веб-странице. Во-первых, они показывают важность раздела, к которому относятся. Чем больше заголовок и его «вес», тем более он значимый. Вспомните, что в газетах и журналах передовицы набраны крупным шрифтом, тем самым, привлекая к ним внимание и говоря: «вот это надо читать обязательно». Во-вторых, с помощью различных заголовков легко регулировать размер текста. Чем выше уровень заголовка, тем больше размер шрифта. Самым верхним уровнем является уровень 1 (`<h1>`), а самым нижним — уровень 6 (`<h6>`). И, в-третьих, поисковики добавляют рейтинг тексту, если он находится внутри тега заголовка. Это важно для раскрутки сайта и для его занятия первых строк выдачи результата в поисковой системе по ключевым словам.

Синтаксис создания заголовков показан в примере 7.3.

Пример 7.3. Добавление заголовков

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>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Заголовки в тексте</title>
</head>
<body>
  <h1>Заголовок первого уровня</h1>
  <h2>Заголовок второго уровня</h2>
  <h3>Заголовок третьего уровня</h3>
  <h4>Заголовок четвертого уровня</h4>
  <h5>Заголовок пятого уровня</h5>
  <h6>Заголовок шестого уровня</h6>
</body>
</html>
```

Результат данного примера показан на рис. 7.3. Содержимое тега `<h1>` отображается самым крупным шрифтом жирного начертания, а `<h6>` — самым мелким.

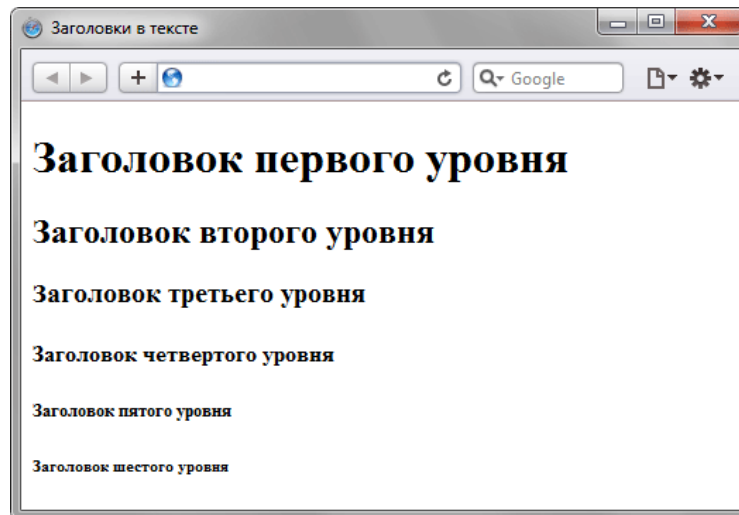


Рис. 7.3. Вид заголовков на веб-странице

Как правило, на веб-странице применяют заголовки с первого по третий уровень, их вполне достаточно. Редко когда приходится использовать заголовки более низкого уровня.

Выравнивание текста

Выравнивание текста определяет его внешний вид и ориентацию краев абзаца и может выполняться по левому краю, правому краю, по центру или по ширине. Наиболее распространенный вариант — выравнивание по левому краю, когда слева текст сдвигается до края, а правый остается неровным. Выравнивание по правому краю и по центру в основном используется в заголовках и подзаголовках. Следует иметь в виду, что при использовании выравнивания по ширине, в тексте между словами могут появиться большие интервалы, что не очень красиво.

Для установки выравнивания текста обычно используется тег параграфа `<p>` с атрибутом `align`, который определяет способ выравнивания. Также блок текста допустимо выравнивать с помощью тега `<div>` с аналогичным атрибутом `align`. Он может принимать следующие значения:

- `left` — выравнивание по левому краю, задается по умолчанию;
- `right` — выравнивание по правому краю;
- `center` — выравнивание по центру;
- `justify` — выравнивание по ширине (одновременно по правому и левому краю). Это значение работает только для текста, длина которого более, чем одна строка.

Атрибут `align` можно применять как для текста, так и для заголовков (пример 7.4).

Пример 7.4. Способы выравнивания текста

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>Выравнивание текста</title>
</head>
<body>
<h1 align="center">Как поймать льва?</h1>
<h2 align="right">Метод перебора</h2>
<p align="justify">Делим пустыню на ряд элементарных участков, размер
которых совпадает с габаритными размерами льва, но при этом меньше размера
клетки. Далее простым перебором определяем участок, в котором находится лев,
что автоматически приводит к его поимке.</p>
<h2 align="right">Метод дихотомии</h2>
<p align="justify">Делим пустыню на две половины. В одной части - лев, в
другой его нет. Берем ту половину, в которой находится лев, и снова делим
ее пополам. Так повторяем до тех пор, пока лев не окажется пойман.</p>
</body>
</html>
```

Результат данного примера показан на рис. 7.4.

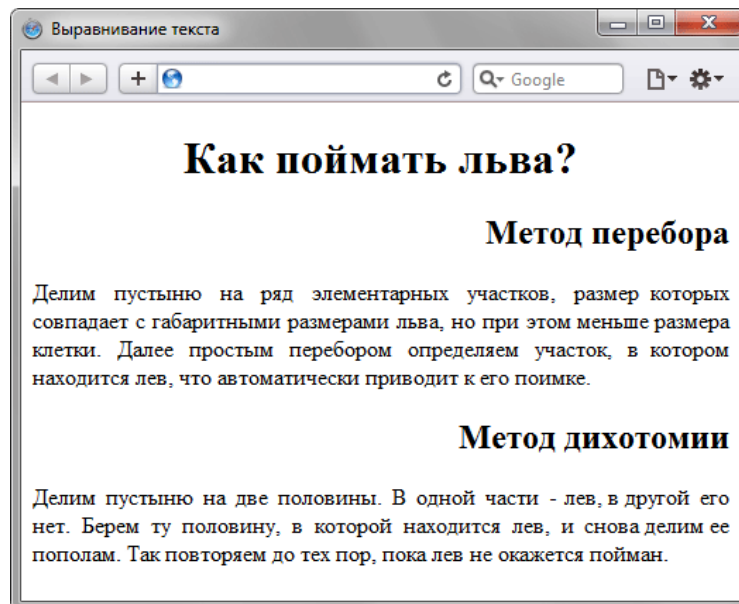


Рис. 7.4. Вид текста при его выравнивании

Начертание

Жирное начертание

Насыщенностью называют увеличение толщины линий шрифта и соответственно контраста. Обычно различают четыре вида насыщенности: светлое начертание, нормальное, полужирное и жирное. Однако с помощью HTML можно установить только нормальное и жирное начертание. Для установки текста жирного начертания применяется два тега: `` и ``.

```
<b>Жирное начертание шрифта</b>  
<strong>Сильное выделение текста</strong>
```

Курсивное начертание

Курсивный шрифт представляет собой не просто наклон отдельных символов, для некоторых шрифтов это полная переделка под новый стиль, имитирующий рукописный. Курсив для текста определяют два тега: `<i>` и ``.

```
<i>Курсивное начертание шрифта</i>  
<em>Выделение текста</em>
```

Следует отметить, что теги `` и ``, также как `<i>` и `` хотя и похожи по своему действию, являются не совсем эквивалентными и заменяемыми. Первый тег `` — является тегом физической разметки и устанавливает жирное начертание текста, а тег `` — тегом логической разметки и выделяет помеченный текст. Такое разделение тегов на логическое и физическое форматирование изначально предназначалось, чтобы сделать HTML универсальным, в том числе не зависящим от устройства вывода информации. Теоретически, если воспользоваться, например, речевым браузером, то текст, оформленный с помощью тегов `` и ``, будет отмечен по-разному. Однако получилось так, что в популярных браузерах результат использования этих тегов равнозначен.

В примере 7.5 показано использование тегов `` и `` для оформления текстов.

Пример 7.5. Теги `` и ``

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>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">  
<title>Оформление текста</title>  
</head>  
<body>  
<p><strong>А где же печенье и самоговаренье?!</strong> —  
<em>воскликнул Мальчиш-плохиш</em>.</p>  
</body>  
</html>
```

Результат данного примера показан на рис. 7.5.

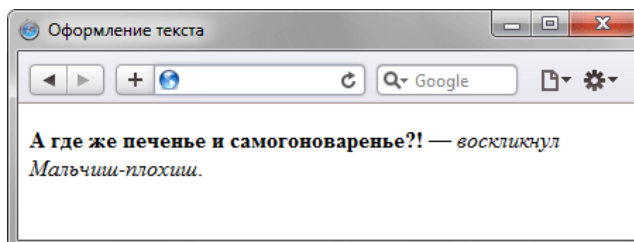


Рис. 7.5. Жирное и курсивное начертание текста

Верхний и нижний индексы

Индексом по отношению к тексту называется смещение символов относительно базовой линии вверх или вниз. В зависимости от положительного или отрицательного значения, индекс называется, соответственно, верхним или нижним. Они активно применяются в математике, физике, химии и для обозначения единиц измерения. HTML предлагает два тега для создания индекса: `<sup>` — верхний индекс и `<sub>` — индекс нижний. Текст, помещенный в один из этих контейнеров, обозначается меньшим размером, чем базовый текст и смещается относительно горизонтали.

В примере 7.6 показано, где применяется подобный текст

Пример 7.6. Использование нижнего индекса

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>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Нижний индекс</title>
</head>
<body>
<p><b>Формула серной кислоты:</b> <i>H<sub>2</sub>SO<sub>4</sub></i></p>
</body>
</html>
```

Результат данного примера показан на рис. 7.6.

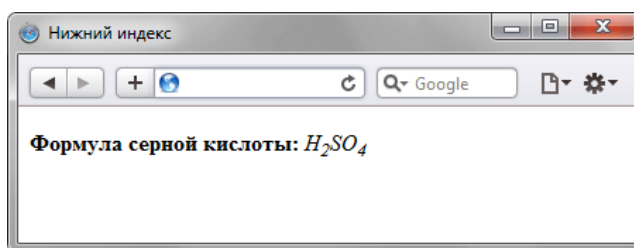


Рис. 7.6. Отображение текста в виде нижнего регистра

Спецсимволы

Для отображения символов, которых нет на клавиатуре применяются специальные знаки, начинающиеся с амперсанда (&) и заканчивающиеся точкой с запятой (;). В табл. 7.1 приведены некоторые популярные спецсимволы.

Табл. 7.1. Спецсимволы

Имя	Код	Вид	Описание
 	 		неразрывный пробел
£	£	£	фунт стерлингов
€	€	€	знак евро
¶	¶	¶	символ параграфа
§	§	§	параграф
©	©	©	знак copyright
®	®	®	знак зарегистрированной торговой марки
™	™	™	знак торговой марки
°	°	°	градус
±	±	±	плюс-минус
¼	¼	¼	дробь - одна четверть
½	½	½	дробь - одна вторая
¾	¾	¾	дробь - три четверти
×	×	×	знак умножения
÷	÷	÷	знак деления
ƒ	ƒ	ƒ	знак функции
Греческие буквы			
Α	Α	Α	греческая заглавная буква альфа
Β	Β	Β	греческая заглавная буква бета
Γ	Γ	Γ	греческая заглавная буква гамма
Δ	Δ	Δ	греческая заглавная буква дельта
Ε	Ε	Ε	греческая заглавная буква эпсилон
Ζ	Ζ	Ζ	греческая заглавная буква дзета
Η	Η	Η	греческая заглавная буква эта
Θ	Θ	Θ	греческая заглавная буква тета
Ι	Ι	Ι	греческая заглавная буква иота
Κ	Κ	Κ	греческая заглавная буква каппа
Λ	Λ	Λ	греческая заглавная буква лямбда
Μ	Μ	Μ	греческая заглавная буква мю
Ν	Ν	Ν	греческая заглавная буква ню
Ξ	Ξ	Ξ	греческая заглавная буква кси
Ο	Ο	Ο	греческая заглавная буква омикрон
Π	Π	Π	греческая заглавная буква пи
Ρ	Ρ	Ρ	греческая заглавная буква ро
Σ	Σ	Σ	греческая заглавная буква сигма
Τ	Τ	Τ	греческая заглавная буква тау
Υ	Υ	Υ	греческая заглавная буква ипсилон
Φ	Φ	Φ	греческая заглавная буква фи
Χ	Χ	Χ	греческая заглавная буква хи
Ψ	Ψ	Ψ	греческая заглавная буква пси
Ω	Ω	Ω	греческая заглавная буква омега
α	α	α	греческая строчная буква альфа
β	β	β	греческая строчная буква бета
γ	γ	γ	греческая строчная буква гамма
δ	δ	δ	греческая строчная буква дельта

ε	ε	ε	греческая строчная буква эпсилон
ζ	ζ	ζ	греческая строчная буква дзета
η	η	η	греческая строчная буква эта
θ	θ	θ	греческая строчная буква тета
ι	ι	ι	греческая строчная буква иота
κ	κ	κ	греческая строчная буква каппа
λ	λ	λ	греческая строчная буква лямбда
μ	μ	μ	греческая строчная буква мю
ν	ν	ν	греческая строчная буква ню
ξ	ξ	ξ	греческая строчная буква кси
ο	ο	ο	греческая строчная буква омикрон
π	π	π	греческая строчная буква пи
ρ	ρ	ρ	греческая строчная буква ро
ς	ς	ς	греческая строчная буква сигма
σ	σ	σ	греческая строчная буква сигма
τ	τ	τ	греческая строчная буква тау
υ	υ	υ	греческая строчная буква ипсилон
φ	φ	φ	греческая строчная буква фи
χ	χ	χ	греческая строчная буква хи
ψ	ψ	ψ	греческая строчная буква пси
ω	ω	ω	греческая строчная буква омега
Стрелки			
←	←	←	стрелка влево
↑	↑	↑	стрелка вверх
→	→	→	стрелка вправо
↓	↓	↓	стрелка вниз
↔	↔	↔	стрелка влево-вправо
Прочие символы			
♠	♠	♠	знак масти "пики"
♣	♣	♣	знак масти "трефы"
♥	♥	♥	знак масти "червы"
♦	♦	♦	знак масти "бубны"
"	"	"	двойная кавычка
&	&	&	амперсанд
<	<	<	знак "меньше"
>	>	>	знак "больше"
Знаки пунктуации			
…	…	...	многоточие ...
′	′	'	одиночный штрих - минуты и футы
″	″	"	двойной штрих - секунды и дюймы
Общая пунктуация			
–	–	—	тире
—	—	—	длинное тире
‘	‘	'	левая одиночная кавычка
’	’	'	правая одиночная кавычка
‚	‚	,	нижняя одиночная кавычка
“	“	“	левая двойная кавычка
”	”	”	правая двойная кавычка
„	„	„	нижняя двойная кавычка
«	«	«	левая двойная угловая скобка
»	»	»	правая двойная угловая скобка

ССЫЛКИ

Ссылки являются основой гипертекстовых документов и позволяют переходить с одной веб-страницы на другую.

Особенность их состоит в том, что сама ссылка может вести не только на HTML-файлы, но и на файл любого типа, причем этот файл может размещаться совсем на другом сайте. Главное, чтобы к документу, на который делается ссылка, был доступ. Иными словами, если путь к файлу можно указать в адресной строке браузера, и файл при этом будет открыт, то на него можно сделать ссылку.

Для создания ссылки необходимо сообщить браузеру, что является ссылкой, а также указать адрес документа, на который следует сделать ссылку. Оба действия выполняются с помощью тега `<a>`. Общий синтаксис создания ссылок следующий.

```
<a href="URL">текст ссылки</a>
```

Атрибут `href` определяет URL (Universal Resource Locator, универсальный указатель ресурса), иными словами, адрес документа, на который следует перейти, а содержимое контейнера `<a>` является ссылкой. Текст, расположенный между тегами `<a>` и ``, по умолчанию становится синего цвета и подчеркивается. В примере 8.1 показано создание нескольких ссылок на разные веб-страницы.

Пример 8.1. Добавление ссылок

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>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>Ссылки на странице</title>
</head>
<body>
<p><a href="dog.html">Собаки</a></p>
<p><a href="cat.html">Кошки</a></p>
</body>
</html>
```

В данном примере создаются две ссылки с разными текстами. При щелчке по тексту «Собаки» в окне браузера откроется документ `dog.html`, а при щелчке на «Кошки» — файл `cat.html`.

Результат примера показан на рис. 8.1. Обратите внимание, что при наведении курсора мыши на ссылку, в строке состояния браузера отображается полный путь к ссылке файлу.

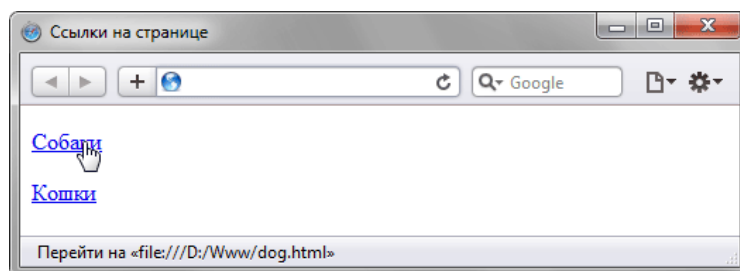


Рис. 8.1. Вид ссылок на странице

Если указана ссылка на файл, которого не существует, например, его имя в атрибуте `href` набрано с ошибкой, то такая ссылка называется битая. Битых ссылок следует категорически избегать, поскольку они вводят посетителей сайта в заблуждение. Так, при щелчке по ссылке из примера 8.1 в браузере Safari откроется не сам документ, а окно с предупреждением (рис. 8.2).

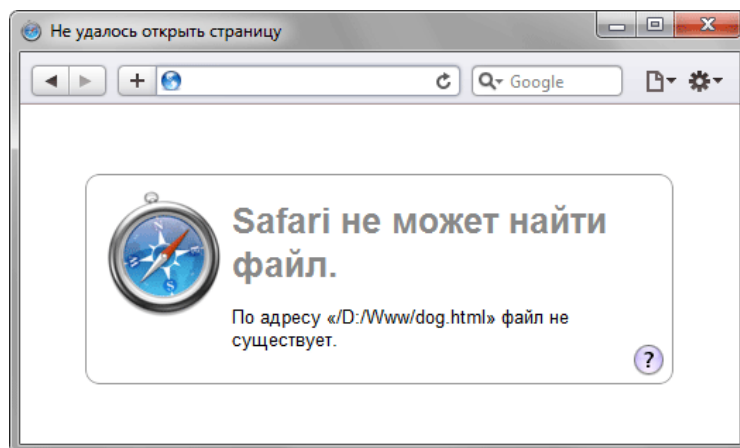


Рис. 8.2. Результат при открытии битой ссылки

Естественно, подобное сообщение будет различаться в браузерах, но смысл остается один — документ, на который ведет ссылка, не может быть открыт. Чтобы не возникало подобных ошибок, тестируйте все ссылки на их работоспособность и сразу же устраняйте имеющиеся погрешности.

Файл по ссылке открывается в окне браузера только в тех случаях, когда браузер знает тип документа. Но поскольку ссылку можно сделать на файл любого типа, то браузер не всегда может отобразить документ. При этом выводится сообщение, как следует обработать файл — открыть его или сохранить в указанную папку. Например, в браузере Firefox выводится следующее окно (рис. 8.3).

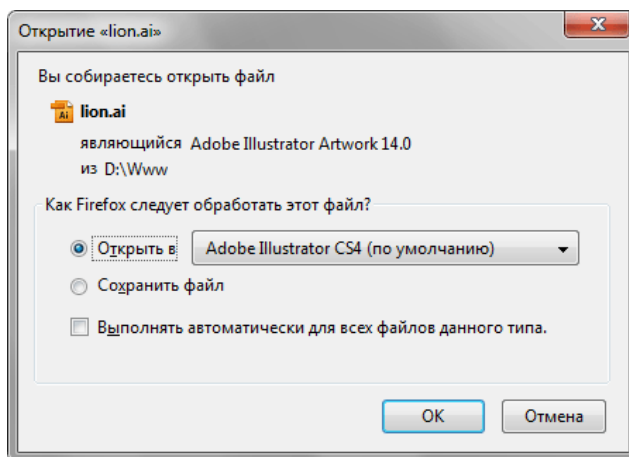


Рис. 8.3. Окно для выбора действия с файлом в Firefox

Абсолютные и относительные ссылки

Адрес ссылки может быть как абсолютным, так и относительным. Абсолютные адреса должны начинаться с указания протокола (обычно http://) и содержать имя сайта. Относительные ссылки ведут отсчет от корня сайта или текущего документа.

В примере 8.2 показано создание абсолютной ссылки на другой сайт.

Пример 8.2. Использование абсолютных ссылок

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>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>Абсолютный адрес</title>
</head>
<body>
<p><a href="http://htmlbook.ru">Изучение HTML</a></p>
</body>
</html>
```

В данном примере ссылка вида `Изучение HTML` является абсолютной и ведет на главную страницу сайта htmlbook.ru.



При указании в качестве ссылки каталога сайта (например, `http://htmlbook.ru/css/`), отображается индексный файл. Это файл, который загружается по умолчанию при обращении к каталогу без явного указания имени файла. Обычно в качестве индексного файла выступает документ с именем `index.html`.

Абсолютные ссылки обычно применяются для указания документа на другом сетевом ресурсе, впрочем, допустимо делать абсолютные ссылки и внутри текущего сайта. Однако подобное практикуется нечасто, поскольку такие ссылки достаточно длинные и громоздкие. Поэтому внутри сайта преимущественно используются относительные ссылки.

Ссылки относительно текущего документа

При создании относительных ссылок надо понимать, какое значение для атрибута `href` следует указывать, поскольку оно зависит от исходного расположения файлов. Рассмотрим несколько типичных вариантов.

1. Файлы располагаются в одной папке (рис. 8.4).

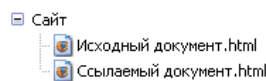


Рис. 8.4

Необходимо сделать ссылку из исходного документа на ссылаемый. В таком случае код будет следующий.

```
<a href="Ссылаемый документ.html">Ссылка</a>
```

Подобное имя файла взято только для образца, на сайте в именах файлов не следует использовать русские символы с пробелами, да еще и в разном регистре.

2. Файлы размещаются в разных папках (рис. 8.5).

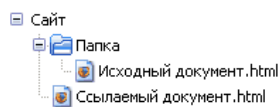


Рис. 8.5

Когда исходный документ хранится в одной папке, а ссылаемый в корне сайта, то перед именем файла в адресе ссылки следует поставить две точки и слэш (`/`), как показано ниже.

```
<a href=" ../Ссылаемый документ.html">Ссылка</a>
```

Две точки в данном случае означают выйти из текущей папки на уровень выше.

3. Файлы размещаются в разных папках (рис. 8.6).

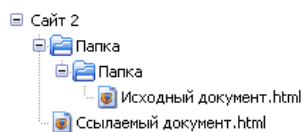


Рис. 8.6

Теперь исходный файл находится в двух вложенных папках, и чтобы сослаться на документ в корне сайта, требуется повторить написание предыдущего примера два раза.

```
<a href="../../Ссылаемый документ.html">Ссылка</a>
```

Аналогично обстоит дело с любым числом вложенных папок.

4. Файлы размещаются в разных папках (рис. 8.7).

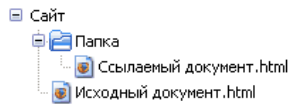


Рис. 8.7

Теперь ситуация меняется, исходный файл располагается в корне сайта, а файл, на который необходимо сделать ссылку - в папке. В этом случае путь к файлу будет следующий.

```
<a href="Папка/Ссылаемый документ.html">Ссылка</a>
```

Заметьте, что никаких дополнительных точек и слэшей перед именем папки нет. Если файл находится внутри не одной, а двух папок, то путь к нему записывается так.

```
<a href="Папка 1/Папка 2/Ссылаемый документ.html">Сылка</a>
```

Ссылки относительно корня сайта

Иногда можно встретить путь к файлу относительно корня сайта, он выглядит как `"/Папка/Имя файла"` со слэшем в начале. Так, запись `Курсы` означает, что ссылка ведет в папку с именем course, которая располагается в корне сайта, а в ней необходимо загрузить индексный файл.

Учтите, что такая форма записи не работает на локальном компьютере, а только под управлением веб-сервера.

Виды ссылок

Любая ссылка на веб-странице может находиться в одном из следующих состояний.

Непосещенная ссылка. Такое состояние характеризуется для ссылок, которые еще не открывали. По умолчанию непосещенные текстовые ссылки изображаются синего цвета и с подчеркиванием.

Активная ссылка. Ссылка помечается как активная в момент ее открытия. Поскольку время между нажатием на ссылку и началом загрузки нового документа достаточно мало, подобное состояние ссылки весьма кратковременно. Активной ссылка становится также, при ее выделении с помощью клавиатуры. Цвет такой ссылки по умолчанию красный.

Посещенная ссылка. Как только пользователь открывает документ, на который указывает ссылка, она помечается как посещенная и меняет свой цвет на фиолетовый, установленный по умолчанию.

Правила вложений для тега <a>

Любая ссылка является встроенным элементом, поэтому для нее действуют те же правила, что и для встроенных элементов. А именно, нельзя размещать внутри тега <a> блочные элементы, но допустимо делать наоборот, и вкладывать ссылку в блочный контейнер. В примере 8.3 показано ошибочное и правильное использование тегов.

Пример 8.3. Вложение тегов

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>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <title>Ошибки при использовании ссылок</title>
</head>
<body>
  <a href="lion.html"><h1>Охота на льва</h1></a>
  <h1><a href="lion.html">Как поймать льва в пустыне</a></h1>
</body>
</html>
```

В строке 8 данного примера содержится типичная ошибка — тег <h1> располагается внутри контейнера <a>. Поскольку <h1> это блочный элемент, то его недопустимо вкладывать внутрь ссылки. В строке 9 этого же примера показан корректный вариант.

Атрибуты ссылок

Основной атрибут `href` тега `<a>` мы уже освоили, рассмотрим еще несколько полезных, но необязательных атрибутов этого тега.

target

По умолчанию, при переходе по ссылке документ открывается в текущем окне или фрейме. При необходимости, это условие может быть изменено атрибутом `target` тега `<a>`. Синтаксис следующий.

```
<a target="имя окна">...</a>
```

В качестве значения используется имя окна или фрейма, заданное атрибутом `name`. Если установлено несуществующее имя, то будет открыто новое окно. В качестве зарезервированных имен применяются следующие.

- `_blank` — загружает страницу в новое окно браузера.
- `_self` — загружает страницу в текущее окно (это значение задается по умолчанию).
- `_parent` — загружает страницу во фрейм-родитель, если фреймов нет, то это значение работает как `_self`.
- `_top` — отменяет все фреймы и загружает страницу в полном окне браузера, если фреймов нет, то это значение работает как `_self`.

В примере 8.4 показано, как сделать, чтобы ссылка открывалась в новом окне.

Пример 8.4. Открытие ссылки в новом окне

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>Ссылка в новом окне</title>
</head>
<body>
<p><a href="new.html" target="_blank">Открыть
в новом окне</a></p>
</body>
</html>
```



Атрибут `target` корректно использовать только при переходном `<!DOCTYPE>`, при строгом `<!DOCTYPE>` будет сообщение об ошибке, поскольку в этой версии HTML `target` уже не поддерживается.

Учтите также, что пользователи не любят, когда ссылки открываются в новых окнах, поэтому используйте подобную возможность осмотрительно и при крайней необходимости.

title

Добавляет поясняющий текст к ссылке в виде всплывающей подсказки. Такая подсказка отображается, когда курсор мыши задерживается на ссылке, после чего подсказка через некоторое время пропадает. Синтаксис следующий.

```
<a title="текст">...</a>
```

В качестве значения указывается любая текстовая строка. Строка должна заключаться в двойные или одинарные кавычки. В примере 8.5 показано, как использовать атрибут `title` для ссылок.

Пример 8.5. Создание всплывающей подсказки

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>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>Подсказка к ссылке</title>
</head>
<body>
<p><a href="zoo.html" title="Рисунки различных животных и не только...">Рисунки</a></p>
</body>
</html>
```

Результат данного примера показан на рис. 8.8.

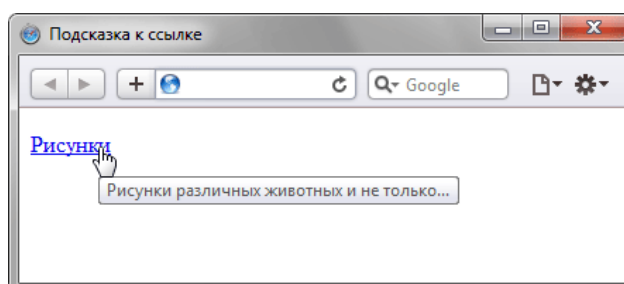


Рис. 8.8. Вид всплывающей подсказки в браузере

Цвета и оформления всплывающей подсказки зависят от настроек операционной системы и браузера, и меняться разработчиком не могут.

Ссылка на адрес электронной почты

Создание ссылки на адрес электронной почты делается почти также как и ссылка на веб-страницу. Только вместо URL указывается `mailto:адрес электронной почты` (пример 8.6).

Пример 8.6. Ссылка на адрес электронной почты

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>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <title>Адрес почты</title>
</head>
<body>
<p><a href="mailto:vlad@htmlbook.ru">Задавайте вопросы по электронной почте</a></p>
</body>
</html>
```

В атрибуте `href` тега `<a>` вначале пишется ключевое слово `mailto`, затем через двоеточие желаемый почтовый адрес. Подобная ссылка ничем не отличается от ссылки на веб-страницу, но при нажатии на нее запускается почтовая программа, установленная по умолчанию. Поэтому в названии ссылки следует указывать, что она имеет отношение к электронной почте, чтобы читатели понимали, к чему приведет нажатие на нее.

Можно также автоматически добавить тему сообщения, присоединив к адресу электронной почты через символ вопроса (?) параметр `subject="тема сообщения"`, как показано в примере 8.7.

Пример 8.7. Задание темы сообщения

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>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <title>Тема письма</title>
</head>
<body>
<p><a href="mailto:vlad@htmlbook.ru?subject=Вопрос по HTML">Задавайте
  вопросы по электронной почте</a></p>
</body>
</html>
```

При запуске почтовой программы поле Тема (Subject) будет заполнено автоматически.

Якоря

Якорем называется закладка с уникальным именем на определенном месте веб-страницы, предназначенная для создания перехода к ней по ссылке. Якоря удобно применять в документах большого объема, чтобы можно было быстро переходить к нужному разделу.

Для создания якоря следует вначале сделать закладку в соответствующем месте и дать ей имя при помощи атрибута **name** тега `<a>` (пример 9.1). В качестве значения **href** для перехода к этому якорю используется имя закладки с символом решетки (#) впереди.

Пример 9.1. Создание якоря

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>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Быстрый переход внутри документа</title>
</head>
<body>
  <p><a name="top"></a></p>
  <p>...</p>
  <p><a href="#top">Наверх</a></p>
</body>
</html>
```

Между тегам `` и `` текст не обязателен, так как требуется лишь указать местоположение перехода по ссылке, находящейся внизу страницы. Имя ссылки на якорь начинается с символа #, после чего идет имя якоря, оно выбирается любое, соответствующее тематике. Главное, чтобы значения атрибутов **name** и **href** совпадали (символ решетки не в счет).



С якорями связана одна особенность работы браузера. После перехода к указанному якорю нажатие на кнопку «Назад» возвращает не на предыдущую просмотренную страницу, а к ссылке, с которой был сделан переход к якорю. Получается, что для перехода к предыдущему документу надо нажать кнопку «Назад» два раза.

Ссылку можно также сделать на закладку, находящуюся в другой веб-странице и даже другом сайте. Для этого в атрибуте **href** тега `<a>` надо указать адрес документа и в конце добавить символ решетки # и имя закладки (пример 9.2).

Пример 9.2. Ссылка на закладку из другой веб-страницы

```
<!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>Якорь в другом документе</title>
</head>
<body>
  <p><a href="text.html#bottom">Перейти к нижней части текста</a></p>
</body>
</html>
```

В данном примере показано создание ссылки на файл `text.html`, при открытии этого файла происходит переход на закладку с именем `bottom`.



Если на веб-странице содержится ссылка на якорь, а самого якоря нет, то никакой ошибки не возникнет.

Изображения

Добавление изображения происходит в два этапа: вначале готовится графический файл желаемого размера, затем он добавляется на страницу через тег ``. Сам HTML предназначен только для того, чтобы отобразить требуемую картинку, при этом никак ее не меняя.

При подготовке изображений следует учесть несколько моментов.

1. Поскольку веб-страница загружается по сети, существенным фактором выступает объем графического файла, встроенного в документ. Чем он меньше, тем быстрее отобразится изображение.
2. Размер картинки необходимо ограничить по ширине, например, установить не более 800 пикселей. Иначе изображение целиком не поместится в окне браузера, и появятся полосы прокрутки.

Форматы файлов

Широкое распространение для веб-графики получили два формата — GIF и JPEG. Их многофункциональность, универсальность, небольшой объем исходных файлов при достаточном для сайта качестве, сослужили им положительную службу, фактически определив их как стандарт веб-изображений. Есть еще формат PNG, который также поддерживается браузерами при добавлении изображений и существует в двух ипостасях — PNG-8 и PNG-24. Однако популярность PNG сильно уступает признанию форматов GIF и JPEG.

Формат GIF

GIF (Graphics Interchange Format) — формат графических файлов, широко применяемый при создании сайтов. GIF использует 8-битовый цвет и эффективно сжимает сплошные цветные области, при этом сохраняя детали изображения.

Особенности

- Количество цветов в изображении может быть от 2 до 256, но это могут быть любые цвета из 24-битной палитры.
- Файл в формате GIF может содержать прозрачные участки. Если используется отличный от белого цвета фон, он будет проглядывать сквозь «дыры» в изображении.
- Поддерживает покадровую смену изображений, что делает формат популярным для создания баннеров и простой анимации.
- Использует свободный от потерь метод сжатия

Область применения

Текст, логотипы, иллюстрации с четкими краями, анимированные рисунки, изображения с прозрачными участками, баннеры.

Формат JPEG

JPEG (Joint Photographic Experts Group) — популярный формат графических файлов, широко применяемый при создании сайтов и хранения изображений. JPEG поддерживает 24-битовый цвет и сохраняет яркость и оттенки цветов в фотографиях неизменными. Данный формат называют сжатием с потерями, поскольку алгоритм JPEG выборочно отвергает данные. Метод сжатия может внести искажения в рисунок, особенно содержащий текст, мелкие детали или четкие края. Формат JPEG не поддерживает прозрачность. Когда вы сохраняете фотографию в этом формате, прозрачные пиксели заполняются определенным цветом.

Особенности

- Количество цветов в изображении — около 16 миллионов, что вполне достаточно для сохранения фотографического качества изображения.
- Основная характеристика формата — качество, позволяющее управлять конечным размером файла.
- Поддерживает технологию, так называемый прогрессивный JPEG, в котором версия рисунка с низким разрешением появляется в окне просмотра до полной загрузки самого изображения.

Область применения

Используется преимущественно для фотографий. Не очень подходит для рисунков содержащих прозрачные участки, мелкие детали или текст.

Формат PNG-8

PNG-8 (Portable Network Graphics) — формат по своему действию аналогичен GIF. По заверению разработчиков использует улучшенный формат сжатия данных, но как показывает практика, это не всегда так.

Особенности

- Использует 8-битную палитру (256 цветов) в изображении, за что и получил в своем названии цифру восемь. При этом можно выбирать, сколько цветов будет сохраняться в файле — от 2 до 256.
- В отличие от GIF, не отображает анимацию ни в каком виде.

Область применения

Текст, логотипы, иллюстрации с четкими краями, изображения с градиентной прозрачностью.

Формат PNG-24

PNG-24 — формат, аналогичный PNG-8, но использующий 24-битную палитру цвета. Подобно формату JPEG, сохраняет яркость и оттенки цветов в фотографиях. Подобно GIF и формату PNG-8, сохраняет детали изображения, как, например, в линейных рисунках, логотипах, или иллюстрациях

Особенности

- Использует примерно 16,7 млн. цветов в файле, из-за чего этот формат применяется для полноцветных изображений.
- Поддерживает многоуровневую прозрачность, это позволяет создавать плавный переход от прозрачной области изображения к цветной, так называемый градиент.
- Из-за того, что используемый алгоритм сжатия сохраняет все цвета и пиксели в изображении неизменными, если сравнить с другими форматами, то у PNG-24 конечный объем графического файла получается наибольшим.

Область применения

Фотографии, рисунки, содержащие прозрачные участки, рисунки с большим количеством цветов и четкими краями

изображений.

Добавление рисунка

Для добавления изображения на веб-страницу используется тег ``, атрибут `src` которого определяет адрес графического файла. Общий синтаксис добавления изображения будет следующий.

```

```

URL (Universal Resource Locator, универсальный указатель ресурсов) представляет собой путь к графическому файлу. Для его указания можно использовать как абсолютный, так и относительный адрес. Далее рассмотрим несколько разных путей к графическому файлу для размещения его на веб-странице. Для примера возьмем файл с рисунком, который называется `sample.gif` и хранится в папке `images` корня сайта.

- Если в начале адреса стоит слэш (символ `/`), это значит, что отсчет идет от корня сайта. Например, адрес сайта — `http://baklan.narod.ru`, значит, написав путь к изображению как `/images/bird.jpg`, мы, тем самым говорим серверу, что показать следует файл `http://baklan.narod.ru/images/bird.jpg`. Учтите, что подобные ссылки со слэшем впереди работают только на веб-сервере, на локальном компьютере они действовать не будут.
- Если перед адресом добавляется упоминание протокола `http` (`http://`), то речь идет об абсолютной ссылке. Изображение всегда будет загружаться с указанного адреса в Интернете, даже при сохранении веб-страницы на локальный компьютер.
- Двоеточие со слэшем (`../`) в начале адреса говорит о том, что и рисунок и веб-страница находятся в разных папках, которые размещены на одном уровне. На рис. 10.1 показан файл `index.html`, в который требуется поместить рисунок `pic.gif`. Тогда относительный путь к изображению из `index.html` будет `../images/pic.gif`. Возможны случаи хранения файлов, что обращение из одного файла к другому превращается в набор двоеточий, например: `../../images/pic.gif`.



Рис. 10.1. Пример размещения файлов

- Имя папки в начале пути, без всяких слэшей и двоеточий, сообщает, что и текущий файл и папка с изображением находятся на одном уровне. Как показано на рис. 10.2, относительный путь к рисунку `pic.gif` из файла `index.html` будет `images/pic.gif`.



Рис. 10.2. Пример размещения файлов

В примере 10.1 показано несколько способов добавления рисунка на веб-страницу.

Пример 10.1. Вставка изображения в документ

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>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Добавление рисунков</title>
</head>
<body>
  <p></p>
  <p></p>
  <p></p>
</body>
</html>
```

Как правило, в качестве формата графического файла выступает GIF и JPEG.

Альтернативный текст

Альтернативный текст позволяет получить текстовую информацию о рисунке при отключенном в браузере показе картинок или во время их загрузки. Такой текст появляется раньше самого изображения и дает представление об его содержании (рис. 10.3). Затем зарезервированное пустое поле заменяется картинкой (рис. 10.4).

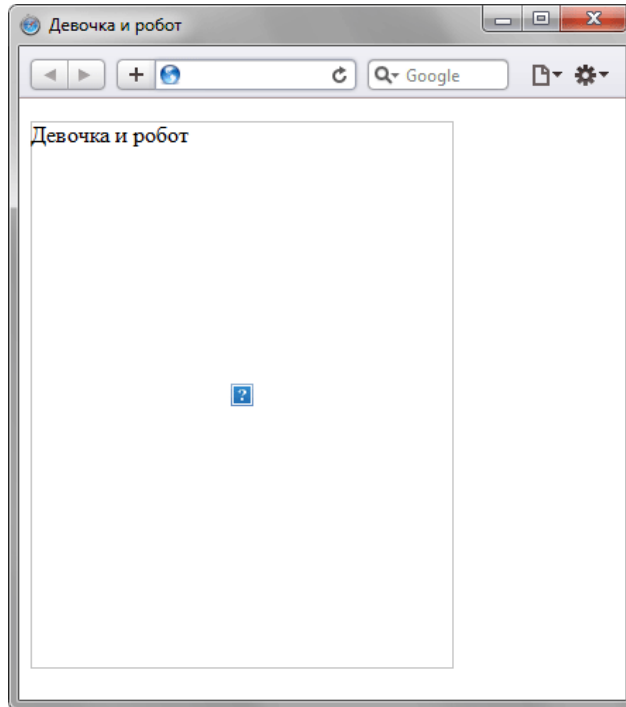


Рис. 10.3. Альтернативный текст до загрузки изображения

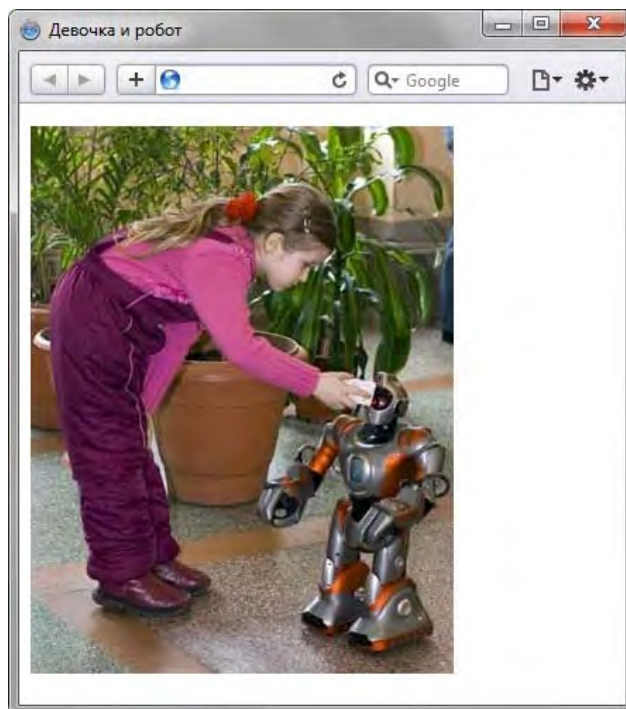


Рис. 10.4. Веб-страница после загрузки изображения



Вид всплывающей подсказки, а именно, ее цвет, фон, шрифт и др. параметры задаются с помощью настроек операционной системы и не могут быть изменены через HTML-файл.

Для создания альтернативного текста используется атрибут `alt` тега ``, как показано в примере 10.2.

Пример 10.2. Добавление альтернативного текста

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>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

```
<title>Альтернативный текст</title>
</head>
<body>
<p></p>
</body>
</html>
```

Учтите, что текст в атрибуте **alt** обязательно должен быть взят в кавычки, как в данном примере.

Не все браузеры отображают альтернативный текст в виде всплывающей подсказки. Поэтому для ее создания используйте атрибут **title** (пример 10.3).

Пример 10.3. Всплывающая подсказка

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>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Атрибут title </title>
</head>
<body>
<p><a href="index.html"></a></p>
</body>
</html>
```

Как показано в данном примере, значения у атрибутов **alt** и **title** может различаться, что позволяет установить определенный текст для разных случаев. Только учтите, что длинный текст будет «обрезаться» и отображается не весь. Но поисковые системы, для которых иной раз и применяют атрибут **title** и **alt**, вполне его читают.

Изменение размеров рисунка

Для изменения размеров рисунка средствами HTML у тега `` предусмотрены атрибуты `width` (ширина) и `height` (высота). В качестве значения используются пиксели, при этом аргументы должны совпадать с физическими размерами картинки. Например, на рис. 10.6 показано изображение, которое имеет размеры 100x111 пикселей.



Рис. 10.6. Картинка исходного размера

Соответственно, HTML-код для размещения данного рисунка, приведен в примере 10.4.

Пример 10.4. Размеры рисунка

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>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Размеры изображения</title>
</head>
<body>
<p></p>
</body>
</html>
```

Если размеры изображения указаны явно, то браузер использует их для того, чтобы отображать соответствующую картинке пустую область в процессе загрузки документа (рис. 10.7). В противном случае браузер ждет, когда рисунок загрузится полностью, после чего меняет ширину и высоту картинки (рис. 10.8). При этом может произойти переформатирование текста, поскольку первоначально размер картинки не известен и автоматически он устанавливается небольшим.

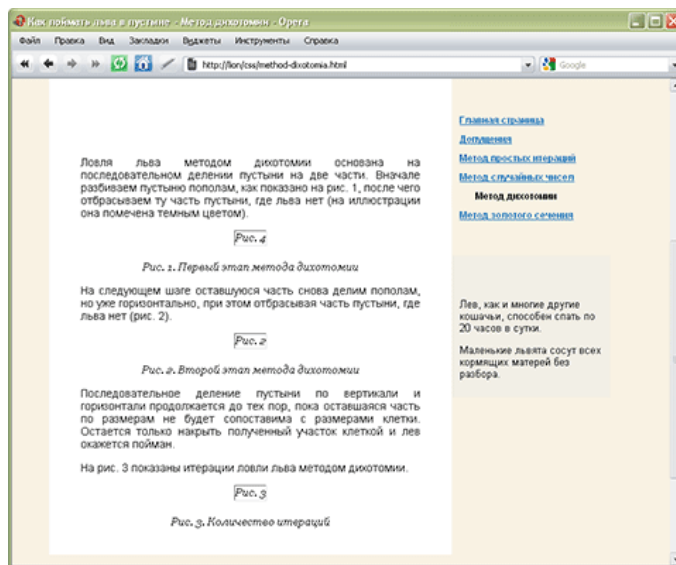


Рис. 10.7. Размеры картинки не указаны и она еще не загрузилась



Рис. 10.8. Картинка загружена, текст переформатирован

Ширину и высоту изображения можно менять как в меньшую, так и большую сторону. Однако на скорость загрузки рисунка это никак не влияет, поскольку размер файла остается неизменным. Поэтому с осторожностью уменьшайте изображение, т.к. это может вызвать недоумение у читателей, отчего такой маленький рисунок так долго грузится. А вот увеличение размеров приводит к обратному эффекту — размер изображения велик, но файл относительно изображения аналогичного размера загружается быстрее.

На рис. 10.9 приведено то же изображение, что показано на рис. 10.6, но с увеличенной в два раза шириной и высотой.



Рис. 10.9. Вид картинки, увеличенной в браузере

Код для такого рисунка останется практически неизменным и показан в примере 10.5.

Пример 10.5. Изменение размера рисунка

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>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Увеличение размеров изображения</title>
</head>
<body>
<p></p>
</body>
</html>
```

Такое изменение размеров называется ресемплированием, при этом алгоритм браузера по своим возможностям уступает графическим редакторам. Поэтому увеличивать таким способом изображения нужно только в особых случаях, а то слишком ухудшается качество картинки. Лучше воспользоваться какой-нибудь графической программой. Исключением являются рисунки, содержащие прямоугольные области. На рис. 10.10 приведен файл узора, который занимает 54 байта и имеет исходный размер 8 на 8 пикселей, увеличенных до 150 пикселей.

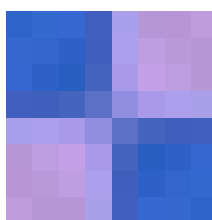


Рис. 10.10. Увеличенное изображение



Браузеры используют два алгоритма для ресемплирования — бикубический (дает сглаженные границы и плавный

тоновый диапазон цветов) и по ближайшим точкам (сохраняет первоначальный набор цветов и резкость краев). Последние версии браузеров применяют бикубический алгоритм, а старые браузеры, наоборот, алгоритм по ближайшим точкам.

Списки

Списком называется взаимосвязанный набор отдельных фраз или предложений, которые начинаются с маркера или цифры. Списки предоставляют возможность упорядочить и систематизировать разные данные и представить их в наглядном и удобном для пользователя виде.

Любой список представляет собой контейнер ``, который устанавливает маркированный список, или ``, определяющий список нумерованный. Каждый элемент списка должен начинаться с тега ``.

Маркированный список

Маркированный список определяется тем, что перед каждым элементом списка добавляется небольшой маркер, обычно в виде закрашенного кружка. Сам список формируется с помощью контейнера ``, а каждый пункт списка начинается с тега ``, как показано ниже.

```
<ul>
<li>Первый пункт</li>
<li>Второй пункт</li>
<li>Третий пункт</li>
</ul>
```

В списке непременно должен присутствовать закрывающий тег ``, иначе возникнет ошибка. Закрывающий тег `` хотя и не обязателен, но советуем всегда его добавлять, чтобы четко разделять элементы списка.

В примере 11.1 приведен код HTML для добавления маркированного списка на веб-странице.

Пример 11.1. Создание маркированного списка

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>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Маркированный список</title>
</head>
<body>
<hr>
<ul>
<li>Чебурашка</li>
<li>Крокодил Гена</li>
<li>Шапокляк</li>
<li>Крыса Лариса</li>
</ul>
<hr>
</body>
</html>
```

Результат данного примера показан на рис. 11.1.

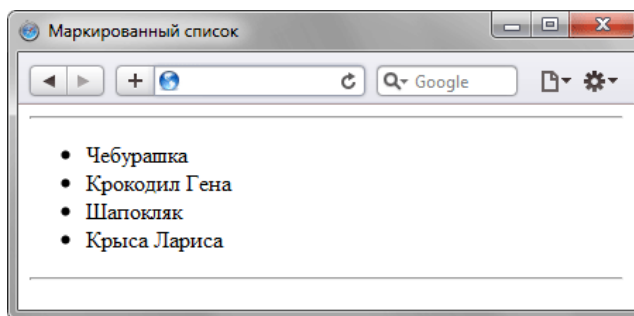


Рис. 11.1. Вид маркированного списка

Обратите внимание на отступы сверху, снизу и слева от списка. Такие отступы добавляются автоматически.

Маркеры могут принимать один из трех видов: круг (по умолчанию), окружность и квадрат. Для выбора стиля маркера используется атрибут `type` тега ``. Допустимые значения приведены в табл. 11.1

Табл. 11.1. Стили маркеров списка

Тип списка	Код HTML	Пример
Список с маркерами в виде круга	<code><ul type="disc"></code> <code>...</code> <code></code>	<ul style="list-style-type: none">• Первый• Второй• Третий
Список с маркерами в виде окружности	<code><ul type="circle"></code> <code>...</code> <code></code>	<ul style="list-style-type: none">◦ Первый◦ Второй◦ Третий
Список с квадратными маркерами	<code><ul type="square"></code> <code>...</code> <code></code>	<ul style="list-style-type: none">▪ Первый▪ Второй▪ Третий



Вид маркеров может незначительно различаться в разных браузерах, а также при смене шрифта и размера текста.



Создание списка с квадратными маркерами показано в примере 11.2.

Пример 11.2. Вид маркеров

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>Маркированный список</title>
</head>
<body>
<p><strong>Изменение убеждений</strong></p>
<ul type="square">
<li>изменение религиозной веры (на выбор: буддизм, конфуцианство, индуизм).
Специальное предложение - иудаизм и мусульманство вместе;</li>
<li>изменение веры в непогрешимость любимой партии;</li>
<li>убеждение в том, что инопланетяне существуют;</li>
<li>предпочтение политического строя, как самого лучшего в своем роде
(на выбор: феодализм, социализм, коммунизм, капитализм).</li>
</ul>
</body>
</html>
```

Результат данного примера показан на рис. 11.2.

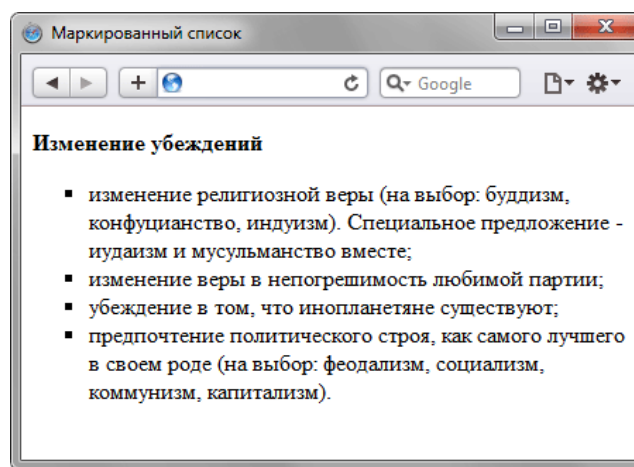


Рис. 11.2. Вид списка с квадратными маркерами

Нумерованный список

Нумерованные списки представляют собой набор элементов с их порядковыми номерами. Вид и тип нумерации зависит от атрибутов тега ``, который и применяется для создания списка. Каждый пункт нумерованного списка обозначается тегом ``, как показано ниже.

```
<ol>
<li>Первый пункт</li>
<li>Второй пункт</li>
<li>Третий пункт</li>
</ol>
```

Если не указывать никаких дополнительных атрибутов и просто написать тег ``, то по умолчанию применяется список с арабскими числами (1, 2, 3,...), как показано в примере 11.3.

Пример 11.3. Создание нумерованного списка

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>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Нумерованный список</title>
</head>
<body>
<p><strong>Работа со временем</strong></p>
<ol>
<li>создание пунктуальности (никогда не будете никуда опаздывать);</li>
<li>излечение от пунктуальности (никогда никуда не будете торопиться);</li>
<li>изменение восприятия времени и часов.</li>
</ol>
</body>
</html>
```

Результат данного примера показан на рис. 11.3.

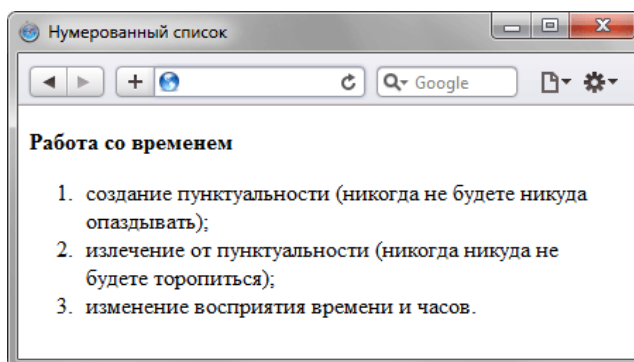


Рис. 11.3. Вид нумерованного списка

Заметьте, что в нумерованном списке также добавляются автоматические отступы сверху, снизу и слева от текста.

В качестве нумерующих элементов могут выступать следующие значения:

- арабские числа (1, 2, 3, ...);
- прописные латинские буквы (A, B, C, ...);
- строчные латинские буквы (a, b, c, ...);
- прописные римские числа (I, II, III, ...);
- строчные римские числа (i, ii, iii, ...).

Для указания типа нумерованного списка применяется атрибут `type` тега ``. Его возможные значения приведены в табл. 11.2.

Табл. 11.2. Типы нумерованного списка

Тип списка	Код HTML	Пример
Арабские числа	<code><ol type="1"></code> <code>...</code> <code></code>	1. Чебурашка 2. Крокодил Гена 3. Шапокляк
Прописные буквы латинского алфавита	<code><ol type="A"></code> <code>...</code> <code></code>	A. Чебурашка B. Крокодил Гена C. Шапокляк
Строчные буквы латинского алфавита	<code><ol type="a"></code> <code>...</code> <code></code>	a. Чебурашка b. Крокодил Гена c. Шапокляк
	<code><ol type="I"></code>	I. Чебурашка

Римские числа в верхнем регистре	... 	II. Крокодил Гена III. Шапокляк
Римские числа в нижнем регистре	<ol type="i"> ... 	i. Чебурашка ii. Крокодил Гена iii. Шапокляк

Чтобы начать список с определенного значения, используется атрибут **start** тега ``. При этом не имеет значения, какой тип списка установлен с помощью **type**, атрибут **start** одинаково работает и с римскими и с арабскими числами. В примере 11.4 показано создание списка с использованием римских цифр в верхнем регистре, начинающихся с восьми.

Пример 11.4. Нумерация списка

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>Римские числа</title>
</head>
<body>
<ol type="I" start="8">
<li>Король Магнум XLIV</li>
<li>Король Зигфрид XVI</li>
<li>Король Сигизмунд XXI</li>
<li>Король Хусбрандт I</li>
</ol>
</body>
</html>
```

Результат данного примера показан на рис. 11.4.

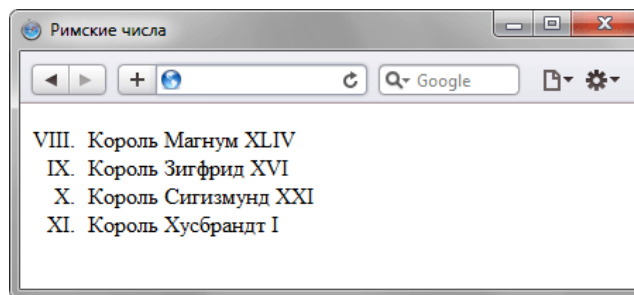


Рис. 11.4. Нумерованный список с римскими числами

Список определений

Список определений состоит из двух элементов — термина и его определения. Сам список задается с помощью контейнера `<dl>`, термин — тегом `<dt>`, а его определение — с помощью тега `<dd>`. Вложение тегов для создания списка определений продемонстрировано в примере 11.5.

Пример 11.5. Общая структура списка определений

```
<dl>
  <dt>Термин 1</dt>
  <dd>Определение 1</dd>
  <dt>Термин 2</dt>
  <dd>Определение 2</dd>
</dl>
```

Список определений хорошо подходит для расшифровки терминов, создания глоссария, словаря, справочника и т.д. В примере 11.6 показано одно из возможных использований этого вида списка.

Пример 11.6. Создание списка определений

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>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Список определений</title>
</head>
<body>
  <dl>
    <dt>Тег</dt>
    <dd>Тег — это специальный символ разметки, который применяется для
      вставки различных элементов на веб-страницу таких как: рисунки,
      таблицы, ссылки и др., и для изменения их вида.</dd>
    <dt>HTML-документ</dt>
    <dd>Обычный текстовый файл, который может содержать в себе текст,
      теги и стили. Изображения и другие объекты хранятся отдельно.
      Содержимое такого файла обычно называется HTML-код.</dd>
    <dt>Сайт</dt>
    <dd>Сайт — это набор отдельных веб-страниц, которые связаны между собой
      ссылками и единым оформлением.</dd>
  </dl>
</body>
</html>
```

Результат примера показан на рис. 11.5.

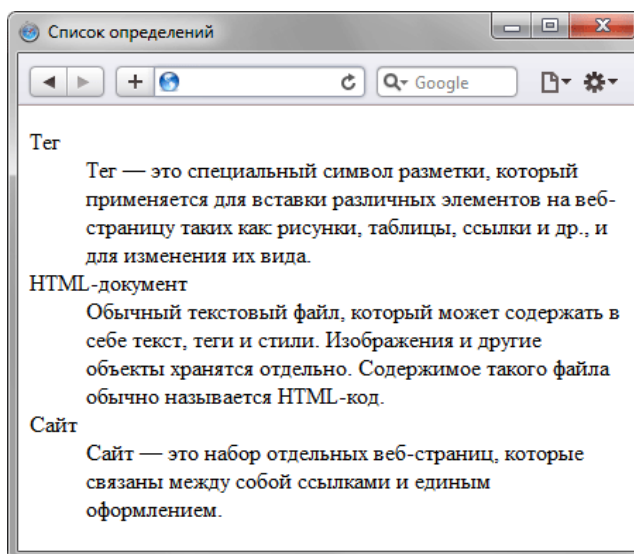


Рис. 11.5. Вид списка определений

Как видно на картинке, текст термина прижимается к левому краю окна браузера, а его определение сдвигается вправо.

Таблицы

Благодаря универсальности таблиц, большому числу параметров, управляющих их видом, таблицы надолго стали определенным стандартом для верстки веб-страниц. Таблица с невидимой границей представляет собой словно модульную сетку, в блоках которой удобно размещать элементы веб-страницы. Тем не менее, это не совсем правильный подход, ведь каждый объект HTML определен для своих собственных целей и если он используется не по назначению, причем повсеместно, это значит, что альтернатив нет. Так оно и было долгое время, пока на смену таблицам при верстке сайтов не пришли слои. Это не значит, что слои теперь используются сплошь и рядом, но тенденция уже наметилась четко — таблицы применяются для размещения табличных данных, а слои — для верстки и оформления.

Создание таблицы

Таблица состоит из строк и столбцов ячеек, которые могут содержать текст и рисунки. Обычно таблицы используются для упорядочения и представления данных, однако возможности таблиц этим не ограничиваются. С помощью таблиц удобно верстать макеты страниц, расположив нужным образом фрагменты текста и изображений.

Для добавления таблицы на веб-страницу используется тег `<table>`. Этот элемент служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются соответственно с помощью тегов `<tr>` и `<td>`. Таблица должна содержать хотя бы одну ячейку (пример 12.1). Допускается вместо тега `<td>` использовать тег `<th>`. Текст в ячейке, оформленной с помощью тега `<th>`, отображается браузером шрифтом жирного начертания и выравнивается по центру ячейки. В остальном, разницы между ячейками, созданными через теги `<td>` и `<th>` нет.

Пример 12.1. Создание таблицы

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>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Тер TABLE</title>
</head>
<body>
  <table border="1" width="100%" cellpadding="5">
    <tr>
      <th>Ячейка 1</th>
      <th>Ячейка 2</th>
    </tr>
    <tr>
      <td>Ячейка 3</td>
      <td>Ячейка 4</td>
    </tr>
  </table>
</body>
</html>
```

Порядок расположения ячеек и их вид показан на рис. 12.1.

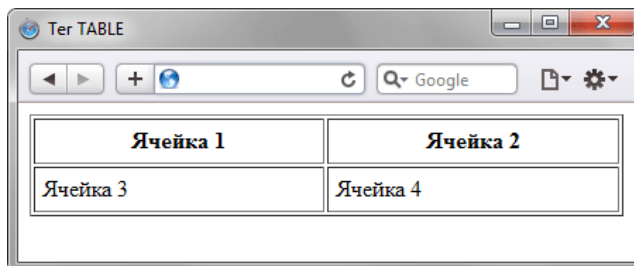


Рис. 12.1. Результат создания таблицы с четырьмя ячейками

Атрибуты тега <table>

Тот факт, что таблицы применяются достаточно часто и не только для отображения табличных данных обязан не только их гибкости и универсальности, но и обилию атрибутов тегов <table>, <tr> и <td>. Далее перечислены некоторые атрибуты тега <table>, которые применяются наиболее часто.

align

Задаёт выравнивание таблицы по краю окна браузера. Допустимые значения: **left** — выравнивание таблицы по левому краю, **center** — по центру и **right** — по правому краю. Когда используются значения **left** и **right**, текст начинает обтекать таблицу сбоку и снизу.

bgcolor

Устанавливает цвет фона таблицы.

border

Устанавливает толщину границ в пикселах. Граница отображается вокруг таблицы и между ячейками.

cellpadding

Определяет расстояние между границей ячейки и ее содержимым. Этот атрибут добавляет пустое пространство к ячейке, увеличивая тем самым ее размеры. Без **cellpadding** текст в таблице «налипает» на рамку, ухудшая тем самым его восприятие. Добавление же **cellpadding** позволяет улучшить читабельность текста. При отсутствии границ особого значения этот атрибут не имеет, но может помочь, когда требуется установить пустой промежуток между ячейками.

cellspacing

Задаёт расстояние между внешними границами ячеек. Если установлен атрибут **border**, толщина границы принимается в расчёт и входит в общее значение.

cols

Атрибут **cols** указывает количество столбцов в таблице, помогая браузеру в подготовке к ее отображению. Без этого атрибута таблица будет показана только после того, как все ее содержимое будет загружено в браузер и проанализировано. Использование атрибута **cols** позволяет несколько ускорить отображение содержимого таблицы.

rules

Сообщает браузеру, где отображать границы между ячейками. По умолчанию рамка рисуется вокруг каждой ячейки, образуя тем самым сетку. В дополнение можно указать отображать линии между колонками (значение **cols**), строками (**rows**) или группами (**groups**), которые определяются наличием тегов <thead>, <tfoot>, <tbody>, <colgroup> или <col>. Толщина границы указывается с помощью атрибута **border**.

width

Задаёт ширину таблицы. Если общая ширина содержимого превышает указанную ширину таблицы, то браузер будет пытаться «втиснуться» в заданные размеры за счёт форматирования текста. В случае, когда это невозможно, например, в таблице находятся изображения, атрибут **width** будет проигнорирован, и новая ширина таблицы будет вычислена на основе ее содержимого. Как и в случае с высотой, если ширина явно не указана, то она будет вычисляться на основе содержимого таблицы.

Атрибуты тега <td>

Каждая ячейка таблицы, задаваемая через тег `<td>`, в свою очередь тоже имеет свои атрибуты, часть из которых совпадает с атрибутами тега `<table>`.

align

Задаёт выравнивание содержимого ячейки по горизонтали. Возможные значения: `left` — выравнивание по левому краю, `center` — по центру и `right` — по правому краю ячейки.

bgcolor

Устанавливает цвет фона ячейки. Используя этот атрибут совместно с атрибутом `bgcolor` тега `<table>` можно получить разнообразные цветовые эффекты в таблице.

colspan

Устанавливает число ячеек, которые должны быть объединены по горизонтали. Этот атрибут имеет смысл для таблиц, состоящих из нескольких строк. Например, как для таблицы, показанной на рис. 12.2.

ячейка 1	
ячейка 2	ячейка 3

Рис. 12.2. Пример таблицы, где используется горизонтальное объединение ячеек

В приведенной на рис. 12.2 таблице содержатся две строки и две колонки, причем верхние горизонтальные ячейки объединены с помощью атрибута `colspan`.

height

Браузер сам устанавливает высоту таблицы и ее ячеек исходя из их содержимого. Однако при использовании атрибута `height` высота ячеек будет изменена. Здесь возможны два варианта. Если значение `height` меньше, чем содержимое ячейки, то этот атрибут будет проигнорирован. В случае, когда установлена высота ячейки, превышающая ее содержимое, добавляется пустое пространство по вертикали.

rowspan

Устанавливает число ячеек, которые должны быть объединены по вертикали. Этот атрибут имеет смысл для таблиц, состоящих из нескольких строк. Например, как для таблицы, показанной на рис. 12.3.

ячейка 1	ячейка 2
	ячейка 3

Рис. 12.3. Пример таблицы, где применяется вертикальное объединение ячеек

В приведенной на рис. 12.3 таблице содержатся две строки и две колонки, левые вертикальные ячейки объединены с помощью атрибута `rowspan`.

valign

Устанавливает вертикальное выравнивание содержимого ячейки. По умолчанию содержимое ячейки располагается по ее вертикали в центре. Возможные значения: `top` — выравнивание по верхнему краю строки, `middle` — выравнивание по середине, `bottom` — выравнивание по нижнему краю, `baseline` — выравнивание по базовой линии, при этом происходит привязка содержимого ячейки к одной линии.

width

Задаёт ширину ячейки. Суммарное значение ширины всех ячеек может превышать общую ширину таблицы только в том случае, если содержимое ячейки превышает указанную ширину.

Особенности таблиц

У каждого параметра таблицы есть свое значение установленное по умолчанию. Это означает, что если какой-то атрибут пропущен, то неявно он все равно присутствует, причем с некоторым значением. Из-за чего вид таблицы может оказаться совсем другим, нежели предполагал разработчик. Чтобы понимать, что можно ожидать от таблиц, следует знать их явные и неявные особенности, которые перечислены далее.

- 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, уже как обычная таблица.
- Размеры таблицы изначально не установлены и вычисляются на основе содержимого ячеек. Например, общая ширина определяется автоматически исходя из суммарной ширины содержимого ячеек плюс ширина границ между ячейками, поля вокруг содержимого, устанавливаемые через атрибут `cellpadding` и расстояние между ячейками, которые определяются значением `cellspacing`.
- Если для таблицы задана ее 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.
- Пока таблица не загрузится полностью, ее содержимое не начнет отображаться. Дело в том, что браузер, прежде чем показать содержимое таблицы, должен вычислить необходимые размеры ячеек, их ширину и высоту. А для этого необходимо знать, что в этих ячейках находится. Поэтому браузер и ожидает, пока загрузится все, что находится в ячейках, и только потом отображает таблицу.

Выравнивание таблиц

Для задания выравнивания таблицы по центру веб-страницы или по одному из ее краев предназначен атрибут `align` тега `<table>`. Результат будет заметен только в том случае, если ширина таблицы не занимает всю доступную область, другими словами, меньше, чем 100%. На самом деле `align` не только устанавливает выравнивание, но и заставляет текст обтекать таблицу с других сторон аналогично поведению тега ``. В примере 12.2 показано выравнивание таблицы по правому краю и ее обтекание текстом.

Пример 12.2. Выравнивание таблицы по правому краю

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>Выравнивание таблицы</title>
</head>
<body>
<table width="200" bgcolor="#c0c0c0" cellspacing="0" cellpadding="5" border="1" align="right">
<tr><td>Содержимое таблицы</td></tr>
</table>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh
eiusmod tincidunt ut lacreet dolore magna aliquam erat volutpat.</p>
</body>
</html>
```

В данном примере создается таблица с фоном серого цвета и выравниванием по правому краю. Результат примера показан на рис. 12.4.

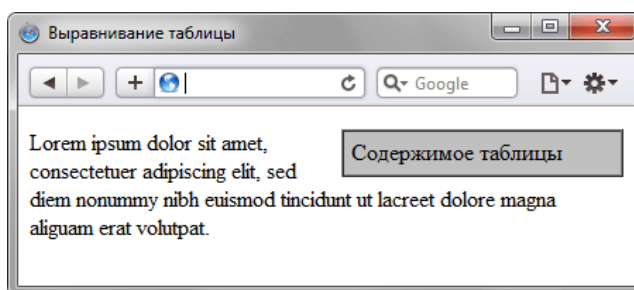


Рис. 12.4. Таблица, выровненная по правому краю окна браузера

По умолчанию таблица формируется в виде сетки, при этом в каждой строке таблицы содержится одинаковое количество ячеек. Такой вариант вполне подходит для формирования простых таблиц, но совершенно не годится для тех случаев, когда предстоит сделать сложную таблицу. В подобных ситуациях применяются два основных метода: объединение ячеек и вложенные таблицы.

Объединение ячеек

Для объединения двух и более ячеек в одну используются атрибуты `colspan` и `rowspan` тега `<td>`. Атрибут `colspan` устанавливает число ячеек объединяемых по горизонтали. Аналогично работает и атрибут `rowspan`, с тем лишь отличием, что объединяет ячейки по вертикали. Перед добавлением атрибутов проверьте число ячеек в каждой строке, чтобы не возникло ошибок. Так, `<td colspan="3">` заменяет три ячейки, поэтому в следующей строке должно быть три тега `<td>` или конструкция вида `<td colspan="2">...</td><td>...</td>`. Если число ячеек в каждой строке не будет совпадать, появятся пустые фантомные ячейки. В примере 12.3 приведен хотя и валидный, но неверный код, в котором как раз проявляется подобная ошибка.

Пример 12.3. Неверное объединение ячеек

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>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Неправильное использование colspan</title>
</head>
<body>
<table border="1" cellpadding="5" width="100%">
<tr>
<td colspan="2">Ячейка 1</td>
<td>Ячейка 2</td>
</tr>
<tr>
<td>Ячейка 3</td>
<td>Ячейка 4</td>
</tr>
</table>
</body>
</html>
```

Результат данного примера показан на рис. 12.5.

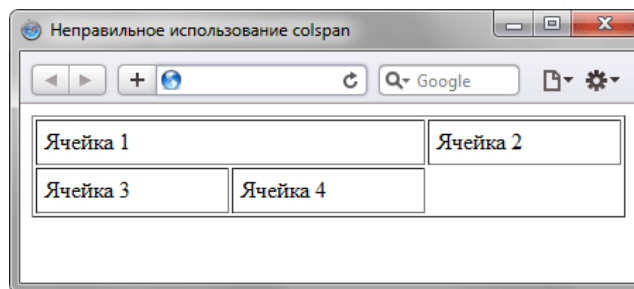


Рис. 12.5. Появление дополнительной ячейки в таблице

В первой строке примера задано три ячейки, две из них объединены с помощью атрибута `colspan`, а во второй строке добавлено только две ячейки. Из-за этого возникает дополнительная ячейка, которая отображается в браузере. Ее хорошо видно на рис. 12.5.

Правильное использование атрибутов `colspan` и `rowspan` продемонстрировано в примере 12.4.

Пример 12.4. Объединение ячеек по вертикали и горизонтали

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>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Объединение ячеек</title>
</head>
<body>
<table border="1" cellpadding="4" cellspacing="0">
<tr>
<td rowspan="2">Браузер</td>
<th colspan="2">Internet Explorer</th>
<th colspan="3">Opera</th>
<th colspan="2">Firefox</th>
</tr>
<tr>
<th>6.0</th><th>7.0</th><th>7.0</th><th>8.0</th><th>9.0</th><th>1.0</th><th>2.0</th>
</tr>
<tr align="center">
<td>Поддерживается</td>
<td>Нет</td><td>Да</td><td>Нет</td><td>Да</td><td>Да</td><td>Да</td>
</tr>
</table>
</body>
</html>
```

Результат данного примера показан на рис. 12.6.

Браузер	Internet Explorer		Opera			Firefox	
	6.0	7.0	7.0	8.0	9.0	1.0	2.0
Поддерживается	Нет	Да	Нет	Да	Да	Да	Да

Рис. 12.6. Таблица с объединенными ячейками

В данной таблице установлено восемь колонок и три строки. Часть ячеек с надписями «Internet Explorer» , «Opera» и «Firefox» объединены где по две, а где и по три ячейки. В ячейке с надписью «Браузер» применено объединение по вертикали.

Вложенные таблицы

Объединение ячеек имеет некоторые недостатки, поэтому этот метод создания таблиц нельзя использовать повсеместно. Для примера рассмотрим пример 12.5, где задается высота ячейки с помощью атрибута `height`.

Пример 12.5. Явно заданная высота ячейки

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>Объединение ячеек</title>
</head>
<body>
<table width="100%" border="1" cellpadding="4" cellspacing="0">
<tr>
<td width="100" valign="top">Duis te feugifacilisi. Duis autem dolor in hendrerit
in vulputate velit esse molestie consequat.</td>
<td rowspan="2" valign="top">Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet
dolore magna aliquam erat volutpat. Ut wisis enim ad minim veniam, quis
nostrud exerci tution ullamcorper suscipit lobortis nisl ut aliquip ex ea
commodo consequat. Duis te feugifacilisi. Ut wisi enim ad minim veniam, quis
nostrud exerci taion ullamcorper suscipit lobortis nisl ut aliquip ex
en commodo consequat.</td>
</tr>
<tr>
<td height="40">Lorem ipsum</td>
</tr>
</table>
</body>
</html>
```

Результат данного примера показан на рис. 12.7.

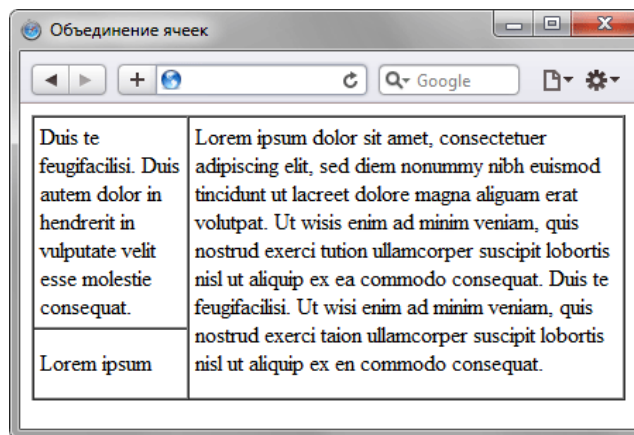


Рис. 12.7. Высота ячеек в браузере Opera 9

Левая нижняя ячейка согласно коду HTML имеет высоту 40 пикселей, но поскольку высота содержимого правой колонки больше, чем содержимое левой колонки, то высота ячейки меняется. Получается, что атрибут `height` в данном случае игнорируется. Заметим, что данная особенность проявляется только в браузере Opera, но и другие браузеры могут отображать сложные таблицы с ошибками. Это часто выражается в тех таблицах, где явно устанавливается высота ячеек и их объединение по вертикали. Для упрощения верстки применяется прием с вложенными таблицами.

Суть идеи проста — в ячейку вкладывается еще одна таблица со своими параметрами. Поскольку эти таблицы в каком-то смысле независимы, то можно создавать довольно причудливые конструкции. Чтобы вложенная таблица занимала всю ширину ячейки, таблице надо задать ширину 100%.

В примере 12.6 показан пример использования вложенных таблиц для создания двух колонок и навигации.

Пример 12.6. Вложенные таблицы

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>Вложенные таблицы</title>
</head>
<body>
<table width="100%" border="0" cellpadding="5" cellspacing="0">
<tr>
<td width="150" valign="top" bgcolor="#f0f0f0">
<table width="100%" cellpadding="2" cellspacing="1">
<tr><td bgcolor="#ffffff">Lorem</td></tr>
<tr><td bgcolor="#ffffff">Ipsum</td></tr>
<tr><td bgcolor="#ffffff">Dolor</td></tr>
<tr><td bgcolor="#ffffff">Sit</td></tr>
<tr><td bgcolor="#ffffff">Amet</td></tr>
</table>

```

```

</td>
<td valign="top" bgcolor="#ffffee">Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet
dolore magna aliquam erat volutpat. Ut wisis enim ad minim veniam, quis
nostrud exerci tution ullamcorper suscipit lobortis nisl ut aliquip ex ea
commodo consequat.</td>
</tr>
</table>
</body>
</html>

```

Результат данного примера показан на рис. 12.8.

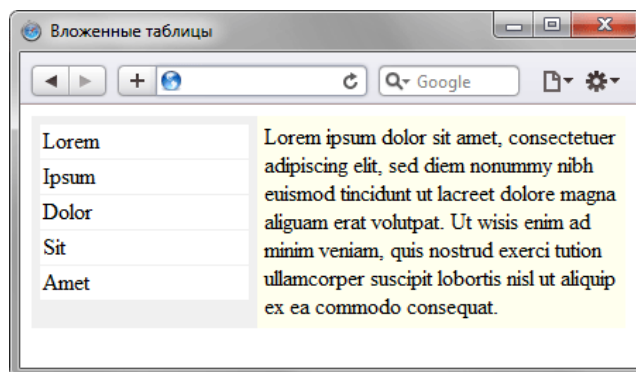


Рис. 12.8. Вид вложенных таблиц

В данном макете с помощью таблицы создается две колонки, причем левая колонка имеет фиксированную ширину 150 пикселей. Чтобы создать подобие навигации, внутрь ячейки добавлена еще одна таблица с шириной 100%.

Как видно из рис. 12.8, если не задавать границы, то определить наличие таблиц по виду веб-страницы довольно сложно. По этой причине таблицы до сих пор активно применяются для верстки сложных макетов.

Заголовок таблицы

При большом количестве таблиц на странице каждой из них удобно задать заголовок, содержащий название таблицы и ее описание. Для этой цели в HTML существует специальный тег `<caption>`, который устанавливает текст и его положение относительно таблицы. Проще всего размещать текст по центру таблицы сверху или снизу от нее, в остальных случаях браузеры по разному интерпретируют атрибуты тега `<caption>`, из-за чего результат получается неодинаковый. По умолчанию заголовок помещается сверху таблицы по центру, его ширина не превышает ширины таблицы и в случае длинного текста он автоматически переносится на новую строку. Для изменения положения заголовка у тега `<caption>` существует атрибут `align`, который может принимать следующие значения.

- `left` — выравнивает заголовок по левому краю таблицы. Браузер Firefox располагает текст сбоку от таблицы, Internet Explorer и Opera располагают заголовок сверху, выравнивая его по левому краю.
- `right` — в браузере Internet Explorer и Opera располагает заголовок сверху таблицы и выравнивает его по правому краю таблицы. Firefox отображает заголовок справа от таблицы.
- `center` — заголовок располагается сверху таблицы по ее центру. Такое расположение задано в браузерах по умолчанию.
- `top` — результат аналогичен действию атрибута `center`, но в отличие от него входит в спецификацию HTML 4 и понимается всеми браузерами.
- `bottom` — заголовок размещается внизу таблицы по ее центру.

В примере 12.7 показано, как установить заголовок сверху таблицы. Обратите внимание, что тег `<caption>` находится внутри контейнера `<table>`, это его стандартное местоположение.

Пример 12.7. Создание заголовка таблицы

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>Заголовок таблицы</title>
</head>
<body>
<table width="100%" border="1" cellpadding="4" cellspacing="0">
<caption>Изменение добычи ресурсов по годам</caption>
<tr>
<th>&nbsp;</th><th>2003</th><th>2004</th><th>2005</th>
</tr>
<tr>
<td>Нефть</td><td>43</td><td>51</td><td>79</td>
</tr>
<tr>
<td>Золото</td><td>29</td><td>34</td><td>48</td>
</tr>
<tr>
<td>Дерево</td><td>38</td><td>57</td><td>36</td>
</tr>
</table>
</body>
</html>
```

Ниже показан результат данного примера (рис. 12.9).



	2003	2004	2005
Нефть	43	51	79
Золото	29	34	48
Дерево	38	57	36

Рис. 12.9. Вид заголовка таблицы в браузере Safari

Фреймы

Фреймы разделяют окно браузера на отдельные области, расположенные рядом друг с другом. В каждую из таких областей загружается самостоятельная веб-страница. Поскольку вокруг фреймов существует много разговоров об их необходимости, далее приведем достоинства и недостатки фреймов, чтобы можно было самостоятельно решить стоит ли их использовать на своем сайте.

Достоинства фреймов

Простота

С помощью фреймов веб-страница разграничивается на две области, которые содержат навигацию по сайту и его контент. Механизм фреймов позволяет открывать документ в одном фрейме, по ссылке, нажатой в совершенно другом фрейме. Такое разделение веб-страницы на составляющие интуитивно понятно и логически обусловлено.

Быстрота

Для верстки без фреймов характерно размещение на одной странице и навигации и содержания. Это увеличивает объем каждой страницы и в сумме может существенно повлиять на объем загружаемой с сайта информации. А так как фреймы используют разделение информации на части, страницы с ними будут загружаться быстрее.

Размещение

Фреймы предоставляют уникальную возможность — размещение информации точно в нужном месте окна браузера. Так, можно поместить фрейм внизу браузера и независимо от прокручивания содержимого, эта область не изменит своего положения.

Изменение размеров областей

Можно изменять размеры фреймов «на лету», чего не позволяет сделать традиционная верстка HTML.

Загрузка

Загрузка веб-страницы происходит только в указанное окно, остальные остаются неизменными. С помощью языка JavaScript можно осуществить одновременную загрузку двух и более страниц во фреймы.

Недостатки фреймов

Навигация

Пользователь зачастую оказывается на сайте, совершенно не представляя, куда он попал, потому что всего лишь нажал на ссылку, полученную в поисковой системе. Чтобы посетителю сайта было проще разобраться, где он находится, на каждую страницу помещают название сайта, заголовок страницы и навигацию. Фреймы, как правило, нарушают данный принцип, отделяя заголовок сайта от содержания, а навигацию от контента. Представьте, что вы нашли подходящую ссылку в поисковой системе, нажимаете на нее, а в итоге открывается документ без названия и навигации. Чтобы понять, где мы находимся или посмотреть другие материалы, придется редактировать путь в адресной строке, что в любом случае доставляет неудобство.

Плохая индексация поисковыми системами

Поисковые системы плохо работают с фреймовой структурой, поскольку на страницах, которые содержат контент, нет ссылок на другие документы.

Внутренние страницы нельзя добавить в «Закладки»

Фреймы скрывают адрес страницы, на которой находится посетитель, и всегда показывают только адрес сайта. По этой причине понравившуюся страницу сложно поместить в закладки браузера.

Несовместимость с разными браузерами

Параметры фреймов обладают свойством совершенно по-разному отображаться в различных браузерах. Причем противоречие между ними настолько явное, что одни и те же параметры интерпретируются браузерами совершенно по-своему.

Непрестижность

Весьма странный недостаток, который не имеет никакого отношения к техническим особенностям создания сайта, а носит скорее идеологический характер. Сайты с фреймами считаются несолидными, а их авторы сразу выпадают из разряда профессионалов, которые никогда не используют фреймы в своих работах. Исключение составляют чаты, где без фреймов обойтись хотя можно, но достаточно хитрыми методами, а с помощью фреймов создавать чаты достаточно просто.

Надо отметить, что некоторые приведенные недостатки вполне обходятся. Так, с помощью скриптов можно сделать, что открытый в браузере отдельный документ формируется со всех фреймовой структурой. Поисковые системы также уже лучше индексируют фреймовые документы, чем это было несколько лет назад.

Создание фреймов

Несмотря на то, что сайты с фреймами встречаются все реже, изучение HTML было бы неполным без рассмотрения темы о фреймах. К тому же фреймы в каком-то смысле заняли свою нишу и применяются для систем администрирования и справки. Там, где недостатки фреймов не имеют особого значения, а преимущества наоборот, активно востребованы.

Для создания фрейма используется тег `<frameset>`, который заменяет тег `<body>` в документе и применяется для разделения экрана на области. Внутри данного тега находятся теги `<frame>`, которые указывают на HTML-документ, предназначенный для загрузки в область (рис. 13.1).

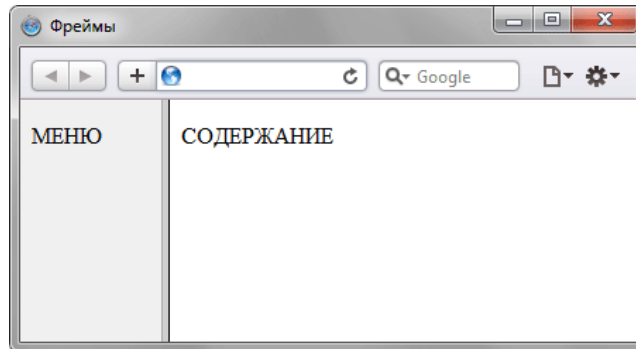


Рис. 13.1. Пример разделения окна браузера на два фрейма

При использовании фреймов необходимо как минимум три HTML-файла: первый определяет фреймовую структуру и делит окно браузера на две части, а оставшиеся два документа загружаются в заданные окна. Количество фреймов не обязательно равно двум, может быть и больше, но никак не меньше двух, иначе вообще теряется смысл применения фреймов.

Рассмотрим этапы создания фреймов на основе страницы, продемонстрированной на рис. 13.1. Нам понадобится три файла: `index.html` — определяет структуру документа, `menu.html` — загружается в левый фрейм и `content.html` — загружается в правый фрейм. Из них только `index.html` отличается по структуре своего кода от других файлов (пример 13.1).

Пример 13.1. Файл `index.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 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Фреймы</title>
</head>
<frameset cols="100,*">
<frame src="menu.html" name="MENU">
<frame src="content.html" name="CONTENT">
</frameset>
</html>
```

В случае использования фреймов в первой строке кода пишется следующий тип документа.

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

Данный `<!DOCTYPE>` указывают браузеру, что он имеет дело с фреймами, эта строка кода является обязательной. Контейнер `<head>` содержит типовую информацию вроде кодировки страницы и заголовка документа. Вот только учтите, что заголовок остается неизменным, пока HTML-файлы открываются внутри фреймов.

В данном примере окно браузера разбивается на две колонки с помощью атрибута `cols`, левая колонка занимает 100 пикселей, а правая — оставшееся пространство, заданное символом звездочки. Ширину или высоту фреймов можно также задавать в процентном отношении, наподобие таблиц.

В теге `<frame>` задается имя HTML-файла, загружаемого в указанную область с помощью атрибута `src`. В левое окно будет загружен файл, названный `menu.html` (пример 13.2), а в правое — `content.html` (пример 13.3). Каждому фрейму желательно задать его уникальное имя, чтобы документы можно было загружать в указанное окно с помощью атрибута `name`.

Пример 13.2. Файл `menu.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>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Навигация по сайту</title>
</head>
<body style="background: #f0f0f0">
<p>МЕНЮ</p>
</body>
</html>
```

В данном примере серый фон на странице задается с помощью стилей, о которых речь пойдет далее.

Пример 13.3. Файл content.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>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Содержание сайта</title>
</head>
<body>
  <p>СОДЕРЖАНИЕ</p>
</body>
</html>
```

Рассмотрим более сложный пример уже с тремя фреймами (рис. 13.2).

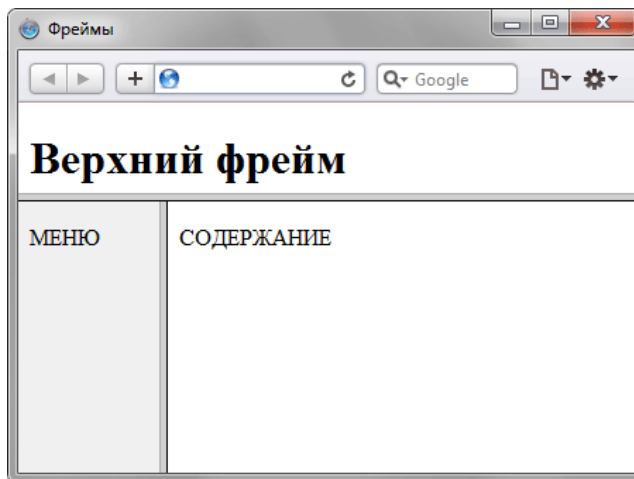


Рис. 13.2. Разделение страницы на три фрейма

В данном случае опять используется тег `<frameset>`, но два раза, причем один тег вкладывается в другой. Горизонтальное разбиение создается через атрибут `rows`, где для разнообразия применяется процентная запись (пример 13.4).

Пример 13.4. Три фрейма

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 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Фреймы</title>
</head>
<frameset rows="25%,75%">
  <frame src="top.html" name="TOP" scrolling="no" noresize>
  <frameset cols="100,*">
    <frame src="menu.html" name="MENU">
    <frame src="content.html" name="CONTENT">
  </frameset>
</frameset>
</html>
```

Как видно из данного примера, контейнер `<frameset>` с атрибутом `rows` вначале создает два горизонтальных фрейма, но вместо второго фрейма подставляется еще один `<frameset>`, который повторяет уже известную вам структуру из примера 13.1. Чтобы не появилась вертикальная полоса прокрутки, и пользователь не мог самостоятельно изменить размер верхнего фрейма, добавлены атрибуты `scrolling="no"` и `noresize`.

Ссылки во фреймах

В обычном HTML-документе при переходе по ссылке, в окне браузера текущий документ заменяется новым. При использовании фреймов схема загрузки документов отличается от стандартной. Основное отличие — возможность загружать документ в выбранный фрейм из другого. Для этой цели используется атрибут `target` тега `<a>`. В качестве значения используется имя фрейма, в который будет загружаться документ, указанный атрибутом `name` (пример 13.5).

Пример 13.5. Ссылка на другой фрейм

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 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Фреймы</title>
</head>
<frameset cols="100,*">
<frame src="menu2.html" name="MENU">
<frame src="content.html" name="CONTENT">
</frameset>
</html>
```

В приведенном примере фрейму присваивается имя `CONTENT`. Чтобы документ загружался в указанный фрейм, используется конструкция `target="CONTENT"`, как показано в примере 13.6.

Пример 13.6. Содержимое файла `menu2.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 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Навигация по сайту</title>
</head>
<body style="background: #f0f0f0">
<p>МЕНЮ</p>
<p><a href="text.html" target="CONTENT">Текст</a></p>
</body>
</html>
```

Имя фрейма должно начинаться на цифру или латинскую букву. В качестве зарезервированных имен используются следующие:

- `_blank` — загружает документ в новое окно;
- `_self` — загружает документ в текущий фрейм;
- `_parent` — загружает документ во фрейм, занимаемый родителем, если фрейма-родителя нет значение действует также, как `_top`;
- `_top` — отменяет все фреймы и загружает документ в полное окно браузера.

Границы между фреймами

Граница между фреймами отображается по умолчанию и, как правило, в виде трехмерной линии. Чтобы ее скрыть используется атрибут `frameborder` тега `<frameset>` со значением `0`. Однако в браузере Opera граница хоть и становится в этом случае бледной, все же остается. Для этого браузера требуется добавить `framespacing="0"`. Таким образом, комбинируя разные атрибуты тега `<frameset>`, получим универсальный код, который работает во всех браузерах. Линия при этом показываться никак не будет (пример 13.6).

Пример 13.6. Убираем границу между фреймами

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 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Фреймы</title>
</head>
<frameset cols="100,*" frameborder="0" framespacing="0">
<frame src="menu.html" name="MENU">
<frame src="content.html" name="CONTENT">
</frameset>
</html>
```

Учтите, что атрибуты `frameborder` и `framespacing` не являются валидными и не соответствуют спецификации HTML.

Если граница между фреймами все же нужна, в браузере она рисуется по умолчанию, без задания каких-либо атрибутов. Можно, также, задать цвет рамки с помощью атрибута `bordercolor`, который может применяться в тегах `<frameset>` и `<frame>`. Цвет указывается по его названию или шестнадцатеричному значению (пример 13.7), а толщина линии управляется атрибутом `border`. Браузер Opera игнорирует этот атрибут и обычно отображает линию черного цвета.

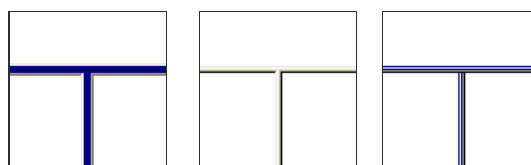
Пример 13.7. Изменение цвета границы

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 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Фреймы</title>
</head>
<frameset cols="100,*" bordercolor="#000080" border="5">
<frame src="menu.html" name="MENU">
<frame src="content.html" name="CONTENT">
</frameset>
</html>
```

Атрибуты `bordercolor` и `border` тега `<frameset>` также не являются валидными и не признаются спецификацией HTML.

В данном примере линия между фреймами задается синего цвета толщиной пять пикселей. Линии различаются по своему виду в разных браузерах, несмотря на одинаковые параметры (рис. 13.3).



Internet Explorer

Opera

Firefox

Рис. 13.3. Вид границы между фреймами в разных браузерах

Браузер Opera никак не изменяет цвет границы между фреймами, Internet Explorer устанавливает широкую границу практически сплошного цвета, а Firefox границу отображает в виде набора линий.

Полосы прокрутки

Если содержимое фрейма не помещается в отведенное окно, автоматически появляются полосы прокрутки для просмотра информации. В некоторых случаях полосы прокрутки нарушают дизайн веб-страницы, поэтому от них можно отказаться. Для управления отображением полос прокрутки используется атрибут `scrolling` тега `<frame>`. Он может принимать два основных значения: `yes` — всегда вызывает появление полос прокрутки, независимо от объема информации и `no` — запрещает их появление (пример 13.9).

Пример 13.9. Запрет на изменение размера фреймов

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 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Фреймы</title>
</head>
<frameset cols="100,*">
<frame src="menu.html" name="MENU" noresize scrolling="no">
<frame src="content.html" name="CONTENT">
</frameset>
</html>
```

При выключенных полосах прокрутки, если информация не помещается в окно фрейма, просмотреть ее будет сложно. Поэтому `scrolling="no"` следует использовать осторожно.

Если атрибут `scrolling` не указан, то полосы прокрутки добавляются браузером только по необходимости, в том случае, когда содержимое фрейма превышает его видимую часть.

Плавающие фреймы

Разговор о фреймах будет неполным без упоминания плавающих фреймов. Так называется фрейм, который можно добавлять в любое место веб-страницы. Еще одно его название — встроенный фрейм, он называется так из-за своей особенности встраиваться прямо в тело веб-страницы. На рис. 13.4 приведен пример такого фрейма.

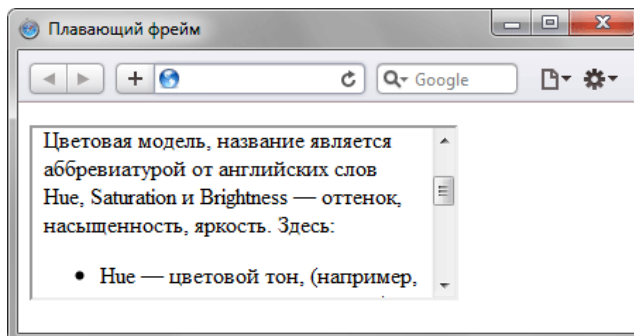


Рис. 13.4. Плавающий фрейм на веб-странице

Во фрейм можно загружать HTML-документ и прокручивать его содержимое независимо от остального материала на веб-странице. Размеры фрейма устанавливаются самостоятельно согласно дизайну сайта или собственных предпочтений.

Создание плавающего фрейма происходит с помощью тега `<iframe>`, он имеет обязательный атрибут `src`, указывающий на загружаемый во фрейм документ (пример 13.10).

Пример 13.10. Использование тега `<iframe>`

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>Плавающий фрейм</title>
</head>
<body>
<p><iframe src="hsb.html" width="300" height="120"></iframe></p>
</body>
</html>
```

В данном примере ширина и высота фрейма устанавливается через атрибуты `width` и `height`. Сам загружаемый во фрейм файл называется `hsb.html`. Заметьте, что если содержимое не помещается целиком в отведенную область, появляются полосы прокрутки.

Еще одно удобство плавающих фреймов состоит в том, что в него можно загружать документы по ссылке. Для этого требуется задать имя фрейма через атрибут `name`, а в теге `<a>` указать это же имя в атрибуте `target` (пример 13.11).

Пример 13.11. Загрузка документа во фрейм

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>Плавающий фрейм</title>
</head>
<body>
<p><a href="rgb.html" target="color">RGB</a> |
<a href="cmyk.html" target="color">CMYK</a> |
<a href="hsb.html" target="color">HSB</a></p>
<p><iframe src="model.html" name="color" width="100%" height="200"></iframe></p>
</body>
</html>
```

В данном примере добавлено несколько ссылок, они открываются во фрейме с именем `color`.



Тег `<iframe>` проходит валидацию только при использовании переходного `<!DOCTYPE>`.

Валидация документов

Валидацией будем называть проверку документа на соответствие веб-стандартам и выявление существующих ошибок. Соответственно, валидным является такой веб-документ, который прошел подобную процедуру и не имеет замечаний по коду. Код веб-страницы должен подчиняться определенным правилам, которые называются спецификацией, ее разрабатывает W3 Консорциум (www.w3c.org) при поддержке разработчиков браузеров.

На первый взгляд, кажется, что валидация необходима, ведь речь идет о сокращении количества ляпов разработчиков и написании «правильного» кода. На деле все обстоит гораздо сложнее и вокруг валидации до сих пор ведутся горячие споры об ее актуальности. Чтобы объективно раскрыть этот вопрос далее рассмотрим плюсы и минусы такой проверки.

Плюсы валидации

Хотя HTML-код имеет достаточно простую иерархическую структуру, при разрастании объема документа в коде легко запутаться, следовательно, просто и совершить ошибку. Браузеры, несмотря на явно неверный код, в любом случае постараются отобразить веб-страницу. Но поскольку не существует единого регламента о том, как же должен быть показан «кривой» документ, каждый браузер пытается сделать это по-своему. А это в свою очередь приводит к тому, что один и тот же документ может выглядеть по-разному в популярных браузерах. Исправление явных промахов и систематизация кода приводит, как правило, к стабильному результату.

Тенденции

Времена, когда производители браузеров добавляли уникальные возможности в свой продукт вопреки всем стандартам, начинают уходить в прошлое. Каждая новая версия браузера все больше поддерживает спецификации и отображает документы с минимальными ошибками или вообще без них. Разработчики сайтов, также придерживающихся канонов веб-стандартов, таким образом соответствуют современным тенденциям развития веб-технологий.

Не стоит забывать и об XML (eXtensible Markup Language, расширяемый язык разметки). Этот язык становится стандартом де-факто для хранения данных и обмена информацией между разными приложениями. Синтаксис XML более жесткий, чем HTML и не прощает малейших ошибок. В каком-то смысле XML похож на языки программирования, в которых программа не будет скомпилирована, пока код не отлажен. HTML является первой ступенькой к изучению XML, поэтому приучая себя писать код по всем правилам, будет легче перейти к следующему этапу развития HTML.

Мода на валидацию

Как это не удивительно, но среди веб-разработчиков тоже существует своя мода. Текущая мода — создавать валидные документы и вывешивать специальный значок в виде картинки, что сайт соответствует спецификации HTML. Подобная тенденция затронула даже заказчиков сайтов и при написании технического задания на разработку сайта некоторые из них специально оговаривают, чтобы сайт был выполнен по веб-стандартам.

Косвенные преимущества

Следование стандартам во многом дает множество выгод, которые проявляются в мелочах и становятся заметными при достижении определенной критической массы. В частности, объем кода становится меньше, компактнее и читабельнее. Соответственно, для пользователей повышается скорость загрузки сайта в целом.

Минусы валидации

Сайты, конечно же, делают для того, чтобы их посещали люди. Именно посетители выступают мерилем работы сайта, а их интересует информация и способ ее получения. Пользователь желает, чтобы сайт корректно отображался в его любимом браузере, быстро загружался и содержал те материалы, которые ему нужны. Заметьте, в этом списке нет ничего про код документа и его валидность, посетителей это просто не интересует. Поэтому совершенно невалидный сайт, но выполненный с душой, наполненный интересными материалами привлечет к себе больше посетителей, чем пустой ресурс, но сделанный по всем «правилам».

Браузеры

Разработчики браузеров не всегда следуют спецификации и в некоторых случаях трактуют код не по заданным правилам, а по-своему. В конечном итоге это приводит к тому, что веб-страница, которая правильно (т.е. так, как и задумывали разработчики) отображается в одном браузере, выводится с ошибками в другом. Следование спецификации в подобных случаях, скорее всего, отпугнет пользователей некоторых браузеров. К примеру, Internet Explorer (IE) в настоящее время занимает лидирующее положение среди браузеров, но при этом поддерживает спецификацию HTML и CSS хуже, чем Firefox и Opera. Очевидно, что пользователи IE при посещении сайта выполненного по всем стандартам, но не учитывающего специфику этого браузера, увидят неприглядную картину.

Заказчиком сайта, а также их разработчикам подобная ситуация не по нраву, поэтому стоя перед выбором: стандарты или браузер, они в большинстве своем выбирают браузер.

Получается неутешительная картина — тратить время на отладку кода для соответствия спецификации нет особой нужды. Это время лучше посвятить тому, чтобы документ без проблем работал в разных браузерах — так в основном размышляют веб-разработчики.

Резюме

Так стоит ли проводить валидацию документов и заниматься этим этапом при написании веб-страниц? Безусловно, да. Кому-то она поможет выявить существующие недочеты, другому поможет писать корректный код. Исправлять же ошибки, добиваясь полного соответствия стандартам, или оставить их ради совместимости с разными браузерами — здесь уже каждый решает сам, какие цели он преследует и что для него важнее.

Проверка данных на валидность

Для проверки веб-страниц на наличие ошибок и замечаний существует множество путей и способов. Условно они делятся на онлайнные и локальные. Онлайнные предназначены для проверки страниц с помощью браузера через Интернет, а локальные используются для проверки документов на текущем компьютере. Далее рассмотрим популярные методы валидации документов.

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 ее адрес (рис. 14.1).

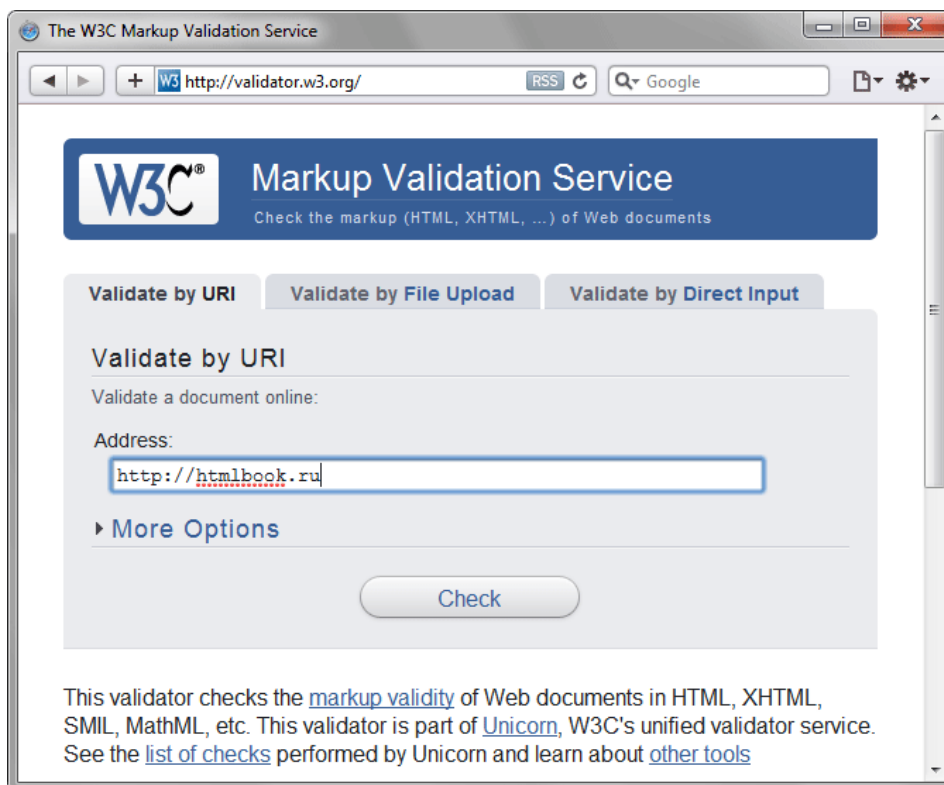


Рис. 14.1. Форма для ввода адреса документа

Так, вводя `http://htmlbook.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://htmlbook.ru` равнозначен вводу `http://htmlbook.ru/index.php`.

Валидатор проверяет HTML-код страницы и в случае отсутствия ошибок докладывает о валидности документа (рис. 14.2).

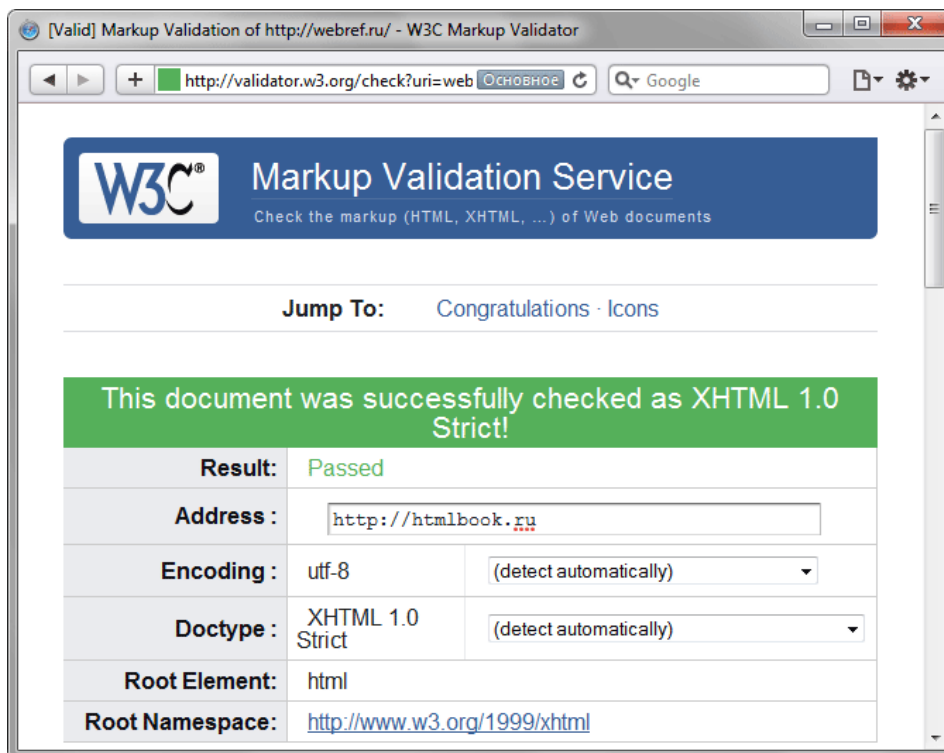


Рис. 14.2. Отчет о проверке и валидности веб-страницы

При обнаружении ошибок выводится уведомление о том, что страница не валидна и список ошибок с указанием строк, где встречаются ошибки (рис. 14.3).

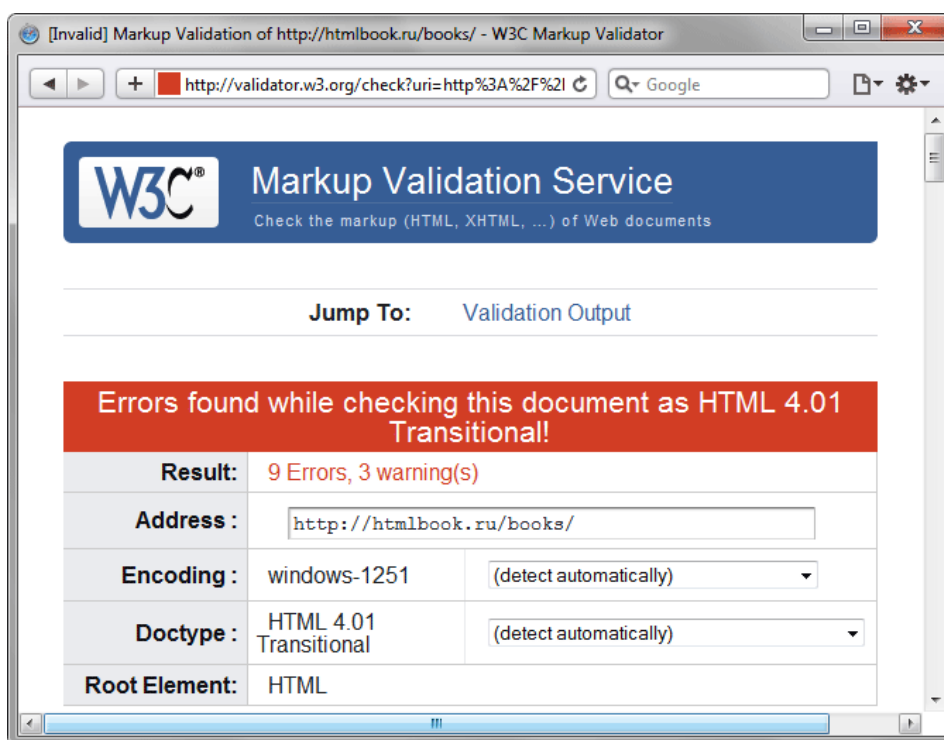


Рис. 14.3. Отчет о проверке и вывод ошибок

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. 14.4.

Validate by URI Validate by File Upload Validate by Direct Input

Validate by File Upload

Upload a document for validation:

File: 1.html

▸ More Options

Note: file upload may not work with Internet Explorer on some versions of Windows XP Service Pack 2, see our [information page](#) on the W3C QA Website.

Рис. 14.4. 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. 14.5).

Validate by URI Validate by File Upload Validate by Direct Input

Validate by direct input

Enter the Markup to validate:

```
<!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>Выворотка</title>
<style type="text/css">
.invert {
background: #000; /* Черный цвет фона */
color: #fff; /* Белый цвет текста */
padding: 2px; /* Поля вокруг текста */
}

```

▸ More Options

Рис. 14.5. Форма для ввода HTML-кода

Расширение HTML Validator для браузера Firefox

Популярность браузера Firefox обусловлена наличием для него большого количества разнообразных расширений — программ, которые добавляют новые возможности в браузер. Расширения построены по открытой технологии и написать их может любой разработчик. Не оставлены без внимания и веб-разработчики — для их удобства создано множество расширений, в том числе и для валидации документа прямо в браузере. В данном случае нас интересует HTML Validator. Эта программа построена по той же технологии, что и валидатор W3C, но не требует подключения к Интернету и работает прямо «на лету».

Где скачать

<http://users.skynet.be/mgqueury/mozilla/>

Установка расширения

После скачивания файла установить расширение можно несколькими способами.

1. Через менеджер расширений

Запустите Firefox и откройте меню **Инструменты > Расширения**. Перетащите мышью загруженный файл (он имеет расширение xpi) в открывшееся окно. Далее расширение будет установлено автоматически.

2. С помощью открытия файла

Выберите в меню Firefox пункт **Файл > Открыть файл...** и укажите путь к файлу с расширением, дальнейшие действия браузер выполнит сам.

3. Копирование файла в папку extension

Откройте папку на диске, где установлен Firefox (к примеру c:\Program Files\Mozilla Firefox) и найдите в ней подпапку extension, в которую скопируйте расширение. После запуска браузера дальнейшая установка пройдет самостоятельно.

Все приведенные методы установки требуют перезагрузки браузера после установки расширения. Работа HTML Validator начинается сразу же после повторного запуска Firefox.

Если указанные способы по каким-либо причинам не помогли, вы можете обратиться на сайт поддержки браузера Mozilla Firefox и прочитать обо всех возможных методах установки расширений по адресу http://forum.mozilla-russia.org/doku.php?id=general:extensions_installing

Использование HTML Validator

При открытии веб-страницы HTML Validator начинает сразу же свою работу, и результат проверки отображается в строке состояния, в ее правом нижнем углу в виде небольшой картинки. Изображение зависит от статуса проверки и показано на рис. 14.6.

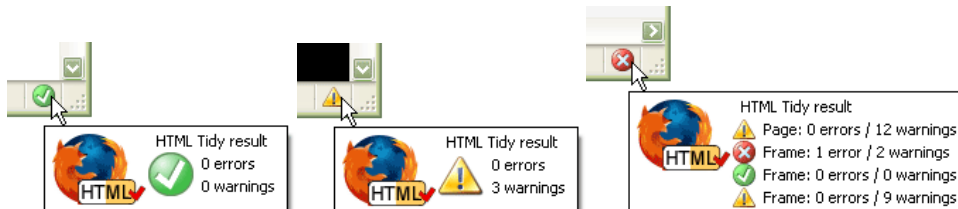


Рис. 14.6. Виды картинок, отображаемых при проверке документа

Кружок с галочкой (рис. 14.6а) показывает, что документ валидный, желтый треугольник с восклицательным знаком (рис. 14.6б) — по коду имеются замечания, которые могут быть исправлены автоматически. А красный кружок с крестиком (рис. 14.6в) предупреждает, что есть серьезные ошибки.

Просмотреть все ошибки можно двояко. Во-первых, заглянуть в HTML-код документа через меню Вид > Исходный код страницы или щелкнуть правой кнопкой и в контекстном меню выбрать Просмотр исходного кода страницы (рис. 14.7).

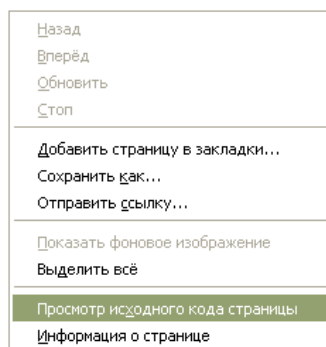


Рис. 14.7. Контекстное меню с пунктом выбора исходного кода

Окно исходного кода веб-страницы разделено на три части (рис. 14.8), где верхний блок содержит собственно HTML-код. В левом нижнем блоке отображается список ошибок и замечаний или информационные сообщения в случае валидного документа. Правый нижний блок предназначен для подробных подсказок о текущих замечаниях.

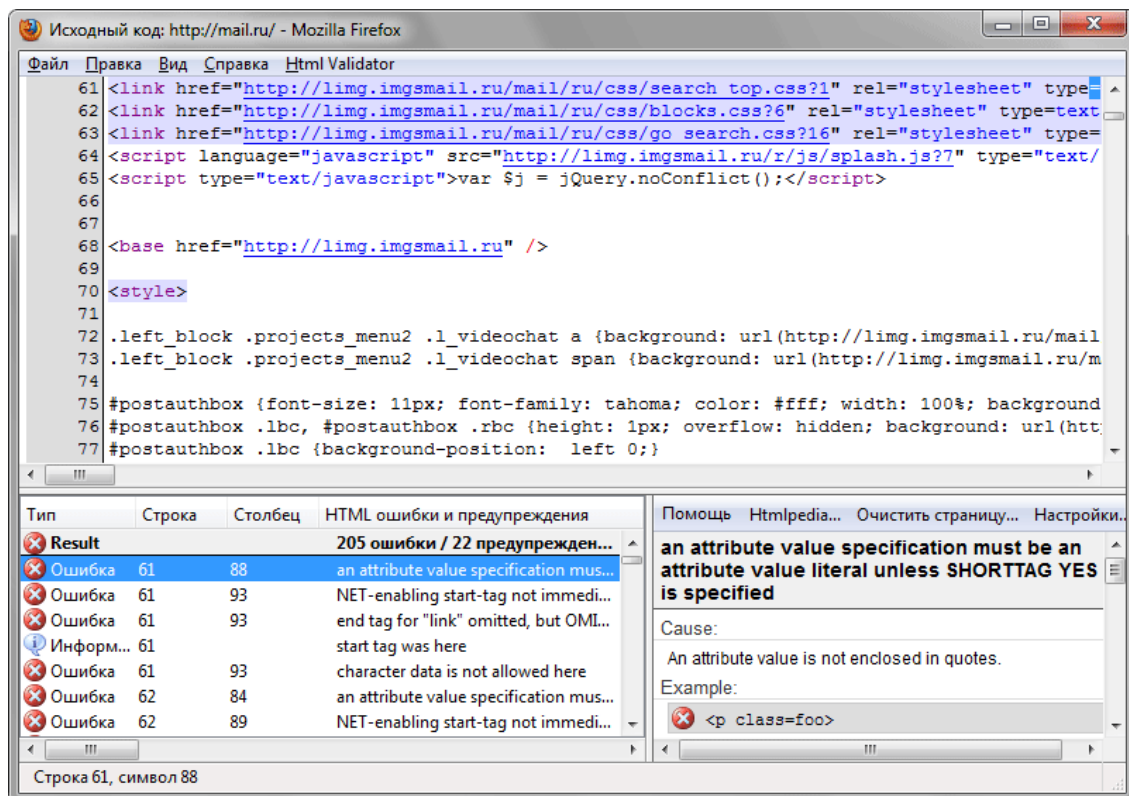


Рис. 14.8. Результат работы расширения HTML Validator

Написание корректного кода

Валидация документов предназначена не только для того чтобы удостовериться, что код соответствует спецификации HTML, но и с целью устранения имеющихся ошибок и замечаний в документе. Между тем, формирование определенной культуры написания кода позволяет существенно снизить или даже вообще избавиться от возможных ошибок. Такая культура складывается из знания спецификаций и типовых ляпов разработчиков, которых надо избегать.

По адресу <http://www.w3.org/TR/html401/> ознакомиться с правилами HTML версии 4.01 может каждый, здесь же мы рассмотрим рядовые ошибки и научимся, как же их обходить.

Ошибки в коде обычно возникают по следующим причинам:

- на странице не задан `<!DOCTYPE>`;
- опечатка (неверно написан тег или его атрибут);
- не указан обязательный атрибут тега;
- используется тег или его атрибут, который не входит в спецификацию;
- неверное вложение тегов.

Далее разберем эти ошибки подробнее.

Не указан `<!DOCTYPE>`

Элемент `<!DOCTYPE>` располагается в первой строке кода документа и сообщает браузеру, как интерпретировать код и отображать данную веб-страницу. Разница между страницей с `<!DOCTYPE>` и без него может быть очень существенной, к тому же валидатор в первую очередь проверяет наличие этого элемента в коде.

Опечатки

Очевидно, что самая простая для исправления ошибка возникает из-за опечатки, когда допущено неверное написание требуемого тега. После валидации выдается тип ошибки и номер строки в коде, где она имеется, так что остается только поменять значение на корректное.

Не указан обязательный атрибут тега

У некоторых тегов имеются атрибуты, которые обязательно должны присутствовать. Например, нельзя просто указать тег `<style>`, необходимо писать `<style type="text/css">`.

Атрибут или значение не входит в спецификацию

В порыве завоевать рынок пользователей разработчики браузеров добавляли в них специальные теги, не входящие в спецификацию HTML, но расширяющие возможности веб-страниц. Со временем часть таких тегов была включена в спецификацию, но многие так и остались «за бортом». При этом поддержка браузером осталась, так что результат работы тега наблюдать можно, но валидацию документ не пройдет. Типичным примером подобного тега является `<marquee>` придуманный компанией Microsoft и понимаемый всеми современными браузерами. Вот только в спецификацию этот тег не включен.

Неверное вложение тегов

Ошибка с вложением одного контейнера внутри другого может быть вызвана следующими причинами:

- блочный элемент располагается внутри встроенного, когда должно быть наоборот — встроенные элементы допустимо помещать внутри блочных;
- пересечение тегов, например, как это показано в следующем примере: `текст`. Здесь закрывающий тег `` помещается в контейнер ``, тогда как он должен следовать только после тега ``;
- не соблюдается порядок вложения тегов. В определенных элементах вроде списка и таблицы принципиальное значение имеет порядок следования тегов. Перестановка тегов местами может привести к неверному отображению объекта и появлению ошибок при валидации документа.

Напоследок отметим еще раз простые правила написания кода, соблюдение которых поможет существенно сократить количество ошибок или обойтись без них.

Закрывайте все теги

Хотя HTML и не требует присутствия некоторых закрывающих тегов, их наличие поможет сохранить строгость кода и четко определить порядок следования тегов.

Указывайте значения атрибутов тегов в кавычках

Валидатор во многих случаях пропустит значения атрибутов указанных без всяких кавычек, тем не менее, кавычки лучше писать всегда. Во-первых, подобный навык поможет для устранения возможных ошибок связанных с атрибутами тегов. А во-вторых, поможет легче перейти на XHTML (Extensible Hypertext Markup Language, расширяемый язык разметки гипертекста), синтаксически более строгую версию HTML. В XHTML кавычки выступают обязательным элементом синтаксиса.

Коллекционируйте заготовки

Большинство элементов веб-страницы достаточно шаблонно, поэтому имея в своем запасе набор проверенных заготовок на разные случаи, можно сократить затраты времени и быть уверенным, что код корректный.

Используйте блочные элементы

Нельзя так просто вставить текст в код документа, он должен располагаться внутри абзаца (тег `<p>`) или другого блочного

элемента. В тех случаях, когда вы не знаете, какой блочный тег использовать, добавляйте универсальный элемент `<div>`.

Переключайте `<!DOCTYPE>`

В HTML-коде обычно применяется строгий `<!DOCTYPE>`, который наиболее полно соответствует спецификации. Однако он же и требует соблюдения всех, самых жестких правил написания кода. В тех случаях, когда это сложно или затратно по времени, переключайтесь на переходный `<!DOCTYPE>`.

Исправление ошибок

Большинство ошибок, возникающих при валидации кода можно свести к набору типовых вариантов, зная которые легко понять, на что «намекает» валидатор. В качестве образца возьмем расширение HTML Validator для браузера Firefox, предназначенное для проверки кода и рассмотрим список ошибок и замечаний по коду.

Посмотреть все возможные сообщения валидатора можно по адресу http://www.htmlpedita.org/wiki/HTML_Tidy, далее приведены основные ошибки с их описанием и решением. Зеленым цветом выделен корректный вариант, другой цвет используется для обозначения ошибки.

Notice: entity "... " doesn't end in ";"

Это замечание возникает при использовании спецсимволов вроде `<` при отсутствии на конце точки с запятой.

✔ ` `;

✘ ` `

Решение

Добавьте в конце спецсимвола точку с запятой.

Notice: numeric character reference "... " doesn't end in ';'

Возникает при использовании числовых спецсимволов вроде `—` когда в конце забыли добавить точку с запятой.

✔ `™`;

✘ `™`

Решение

Добавьте в конце спецсимвола точку с запятой.

unescaped & or unknown entity "&..."

Символ амперсанда (&) часто применяется в адресах ссылок (атрибут `href` тега `<a>`), поскольку он разделяет несколько параметров. Однако амперсанд зарезервирован для спецсимволов вроде ` `; поэтому в ссылках необходимо указывать `&` вместо `&`.

✔ `Ссылка`

✘ `Ссылка`

Решение

Заменить `&` на `&`.

missing </...>

Отсутствует обязательный закрывающий тег.

✔ `<head><title>Заголовок</title></head>`

✘ `<head><title>Заголовок</head>`

Решение

Добавьте закрывающий тег.

missing </aaa> before <bbb>

Ошибка возникает при нарушении порядка тегов, когда блочный тег располагается внутри встроенного. В данном случае блочный тег `<bbb>` находится внутри встроенного тега `<aaa>`.

✔ `<p>Текст</p>`

✘ `<p>Текст</p>`

Решение

Поменяйте расположение тегов — перенесите встроенный тег внутрь блочного.

discarding unexpected <...>

Обнаружен открывающий или закрывающий тег, у которого нет пары. Подобная ошибка возникает в двух случаях: есть открывающий тег, но нет закрывающего; имеется закрывающий тег, которому не соответствует открывающий.

✔ `<div><div>Текст</div></div>`

✘ `<div>Текст</div></div>`

✘ `<div><div>Текст</div>`

Решение

В зависимости от ситуации добавьте или удалите открывающий или закрывающий тег.

Notice: nested emphasis ...

Контейнер содержит аналогичный тег физического форматирования, который не должен повторяться.

✔ `<p>Текст</p>`

✘ `<p>Текст</p>`

Решение

Удалите один из тегов.

replacing unexpected ... by </...>

Закрывающий тег не соответствует открывающему тегу.

✔ `<p>Текст</p>`

✘ `<p>Текст</p>`

Решение

Замените открывающий или закрывающий тег на парный.

... isn't allowed in <...> elements

Обнаружены теги, которые запрещено размещать внутри указанных элементов.

✔ `<head><title>Заголовок</title></head>`

✘ `<head><body>Текст</body></head>`

Решение

Переместите HTML-элемент в правильный раздел.

missing <...>

Нет обязательного тега в структуре элементов. Ошибка, к примеру, может возникнуть при формировании таблицы, когда пропущен тег `<tr>` и сразу же после `<table>` следует `<td>`.

✔ `Список`

✘ `Список`

Решение

Проверить правильность вложения тегов в текущем элементе и наличие обязательных элементов.

Notice: inserting implicit <...>

Сообщение возникает из-за предыдущей ошибки на странице.

Решение

Исправьте предыдущие ошибки.

Insert missing <title> element

В коде не вставлен тег `<title>`.

✔ `<head><title>Заголовок</title></head>`

✘ `<head></head>`

Решение

Добавьте контейнер `<title>`.

Multiple <frameset> elements

Тег `<frameset>` используется в документе более одного раза без вложения. Допускается вставлять несколько элементов `<frameset>`, но вложенных один в другой.

✔ `<frameset ...><frame ...>
</frameset ...><frame ...></frameset>
</frameset>`

✘ `<frameset ...><frame ...></frameset>`
✘ `<frameset ...><frame ...></frameset>`

Решение

Используйте вложенные теги `<frameset>`.

<...> is not approved by W3C

Указанный тег не входит в спецификацию HTML.

✔ `текст без переносов`

✘ `<nobr>текст без переносов</nobr>`

Решение

Удалите тег или замените его подходящим эквивалентом.

Error: <...> is not recognized!

Тег не распознан и не входит в спецификацию HTML.

✔ Правильно: `<p>Текст</p>`

✘ Неверно: `<p><adres>Текст</adres></p>`

Решение

Удалите неизвестный тег.

Trimming Empty Tag

Контейнер пустой или содержит только пробел.

✔ `<p>Текст</p>`

✔ `<p> </p>`

✘ `<p></p>`

Решение

Удалите тег или добавьте внутрь контейнера текст.

<a> is probably intended as

В закрывающем теге `<a>` отсутствует слэш.

✔ `Ссылка на сайт`

✘ `Ссылка на сайт<a>`

Решение

Добавьте слэш к закрывающему тегу.

... shouldn't be nested

Некоторые теги вроде `<form>` не могут содержать сами себя. Это сообщение также возникает из-за предыдущей ошибки.

✔ `<form action="gb.php" name="guestbook"></form>`
✔ `<form action="gb2.php" name="guestbook2"></form>`

✘ `<form action="gb.php" name="guestbook">`
✘ `<form action="gb2.php" name="guestbook2"></form>`
✘ `</form>`

Решение

Удалите вложенные теги или исправьте предыдущую ошибку.

Text found after closing </body>-tag

Теги или текст добавляется после закрывающего тега `</body>`.

✔ `<html>`
✔ `<head><title>Заголовок</title></head>`
✔ `<body><p>Основной текст</p></body>`
✔ `</html>`

<html>

```
✘ <head><title>Заголовок</title></head>
  <body><p>Основной текст</p></body>
  <b>Привет!</b>
</html>
```

Решение

Удалите текст после тега `</body>` или перенесите этот тег в конец текста.

Adjacent hyphens within comment

Комментарии в коде HTML определяются конструкцией вида `<!-- комментарий -->`. Если в тексте комментария подряд идет два и более дефиса, возникает ошибка.

```
✔ <!-- Комментарий - заголовок -->
```

```
✘ <!--- комментарий --->
```

```
✘ <!-- Комментарий -- тело документа -->
```

Решение

Удалите лишние дефисы.

SYSTEM, PUBLIC, W3C, DTD, EN must be upper case

Элемент `<!DOCTYPE>` указан неверно, в частности следующие атрибуты необходимо писать в верхнем регистре: `SYSTEM`, `PUBLIC`, `W3C`, `DTD`, `EN`.

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

```
✘ <!doctype html public "-//w3c//dtd html 4.01 Transitional//en" "http://www.w3.org/TR/html4/loose.dtd">
```

Решение

Пишите `<!DOCTYPE>` корректно.

Warning: missing <!DOCTYPE> declaration

Не указан элемент `<!DOCTYPE>`.

```
✔ <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
  <html>
  <head>
  <title>Заголовок</title>
  </head>
  <body>
  <p>Основной текст</p>
  </body>
  </html>
```

```
✘ <html>
  <head>
  <title>Untitled Document</title>
  </head>
  <body>
  </body>
  </html>
```

Решение

Поместите элемент `<!DOCTYPE>` в самую первую строку кода документа.

Too much <...>-elements

Повторяется тег, который в коде должен быть только один. К таким тегам относится `<html>`, `<head>`, `<title>` и `<body>`.

```
✔ <head>
  <title>Заголовок</title>
</head>
```

```
✘ <head>
  <title>Заголовок</title>
  <title>Название статьи</title>
</head>
```

Решение

Удалите повторяющийся тег.

<...> inserting "..." attribute

Не указан обязательный атрибут для данного тега.

✔ `<style type="text/css">`

✘ `<style>`

Решение

Проверьте тег и добавьте недостающие атрибуты.

... attribute ... lacks value

Атрибут тега не содержит обязательное значение или оно написано с синтаксической ошибкой.

✔ `Ссылка`

✘ `<a href>Ссылка`

Решение

Проверьте атрибуты тега и добавьте недостающие значения.

... attribute "..." has invalid value "..."

Атрибут содержит некорректное значение. Ошибка проявляется в тех случаях, когда в значении вместо текста пишется число и наоборот. Так, атрибуты `id` и `name` должны начинаться с символа ([A-Za-z]) и могут содержать цифры ([0-9]), дефис (-), подчеркивание (_), двоеточие (:) и точку (.). Значение ширины и высоты в атрибутах тегов не должно содержать ничего, кроме цифр ([0-9]).

✔ `<div id="layer1">Слой 1</div>`

✔ ``

✘ `<div id="2layer">Слой 2</div>`

✘ ``

Решение

Проверьте атрибут тега и измените его значение.

<...> missing > for end of tag

Ошибка может возникать в двух случаях: некорректно написан тег, что происходит, когда забыли добавить закрывающую скобку и применение `>` вместо использования спецсимвола.

✔ `<r>Пример текста</r>`

✔ `<r>Для случая 0<r рассмотрим следующий пример.</r>`

✘ `<r Пример текста</r>`

✘ `<r>Для случая 0<r рассмотрим следующий пример.</r>`

Решение

Вставьте отсутствующую закрывающую скобку.

Замените `<` на `<`.

<...> proprietary attribute "..."

Тег содержит атрибут, специфичный только для браузера Internet Explorer или другого и не входящий в спецификацию. Примером является атрибут `height` тега `<table>`.

Список всех атрибутов, входящих в спецификацию HTML приведен по адресу <http://www.w3.org/TR/html4/index/attributes.html>

✔ `<table style="height: 100%">`

✘ `<table height="100%">`

Решение

Список наиболее характерных атрибутов тегов приведен в табл. 14.1.

Табл. 14.1. Замена нестандартных атрибутов тегов

Тег	Устаревший атрибут	Стандартный атрибут
<code><body></code>	<code>marginwidth=0, marginheight=0, leftmargin=0, topmargin=0</code>	<code>style="margin: 0"</code>
<code><table></code>	<code>height=100%</code>	<code>style="height: 100%"</code>

<table>	nowrap	style="white-space: nowrap" или <td nowrap>
<td>	background="abc.gif"	style="background-image:url(abc.gif)"

... proprietary attribute value "..."

Значение атрибута не входит в спецификацию HTML и является специфичным для браузера Internet Explorer или другого. Например, значение `align="absmiddle"` тега `` недопустимо.

✔ `<p></p>`

✔ `<p></p>`

✘ `<p></p>`

Решение

Используйте стандартные значения атрибутов тегов или используйте стилевой эквивалент.

... dropping value "..." for repeated attribute "..."

Атрибут применяется в теге больше одного раза.

✔ ``

✘ ``

Решение

Удалите повторяющийся атрибут.

... unexpected or duplicate quote mark

Отсутствует открывающая или закрывающая кавычка в атрибуте тега.

✔ ``

✘ ``

Решение

Добавьте парную кавычку к значению атрибута.

... attribute with missing trailing quote mark

Тег содержит атрибут, в котором задано неверное количество кавычек.

✔ `<p id="my_id">`

✘ `<p id="my_id"'>`

Решение

Добавьте или удалите одну из кавычек.

... id and name attribute value mismatch

Ошибка возникает, когда значения атрибутов `id` и `name` не совпадают между собой, что приводит к конфликту при обращении к свойствам элемента через скрипты.

✔ ``

✔ ``

✘ ``

Решение

Удалите один из атрибутов или сделайте значения атрибутов `name` и `id` одинаковыми.

Notice: replacing <...> by <...>

Ошибка возникает в следующих случаях:

- неверный порядок тегов;
- добавлен лишний закрывающий тег;
- имеется открывающий тег без наличия обязательного закрывающего.

✔ `<p>Текст</p>
`

✘ `<p>Текст</p></p>`

✘ `<p>abc
<table>...</table></p>`

Решение

Измените порядок тегов или удалите один из открывающих или закрывающих тегов.

... anchor "..." already defined

Значения атрибута `name` у различных тегов совпадает между собой. Значение `name` должно быть уникальным.

✔ `<form name="my_form1" action="test1.php"></form>`
`<form name="my_form2" action="test2.php"></form>`

✘ `<form name="my_form" action="test1.php"></form>`
`<form name="my_form" action="test2.php"></form>`

Решение

Выберите другое имя или измените предыдущие имена таким образом, чтобы они не совпадали.

<...> is probably intended as </...>

Тег повторяется дважды в коде HTML, тогда как подобный тег не должен содержать сам себя.

✔ `Привет, мир!`

✘ `Привет, мир!`

Решение

Удалите один из тегов.

<...> lacks "..." attribute

Требуется обязательный атрибут тега, который, тем не менее, отсутствует.

✔ `<form action="my_action.php">`

✘ `<form>`

Решение

Добавьте недостающий атрибут к тегу.