ФГБОУ ВПО
«Воронежский государственный
технический университет»

М.Ю. Сергеев Т.И. Сергеева

WEB-ДИЗАЙН:
СОЗДАНИЕ WEB-САЙТОВ
С ПОМОЩЬЮ HTML И CSS

Утверждено Редакционно-издательским советом
университета в качестве учебного пособия

Воронеж 2012

УДК 681.3

Сергеев М.Ю. Web-дизайн: создание Web-сайтов с помощью HTML и CSS: учеб. пособие / М.Ю. Сергеев, Т.И. Сергеева. Воронеж: ФГБОУ ВПО «Воронежский государственный технический университет», 2012. 219 с.

В учебном пособии рассматриваются вопросы проектирования, разработки и оформления Web-сайтов с помощью HTML и CSS.

Издание соответствует требованиям Федерального государственного образовательного стандарта высшего профессионального образования по направлению подготовки бакалавров 230100 «Информатика и вычислительная техника», профилю «Вычислительные машины, комплексы, системы и сети», дисциплине «Web-дизайн».

Табл. 1. Ил. 38. Библиогр.: 9 назв.

Научный редактор д-р техн. наук, проф. С.Л. Подвальный

Рецензенты: кафедра информатики и вычислительной техники Международного института компьютерных технологий (канд. техн. наук, доц. И.Н. Крючкова); д-р техн. наук, проф. В.Л. Бурковский

© Сергеев М.Ю., Сергеева Т.И., 2012

© Оформление. ФГБОУ ВПО «Воронежский государственный технический университет», 2012

ВВЕДЕНИЕ

Одной из наиболее динамично развивающихся областей использования информационных технологий является разработка веб-страниц и веб-сайтов различной тематики, направленности и степени сложности.

Основной целью данного учебного пособия является изложение теоретического материала по тегам HTMLи каскадным таблицам стилей CSS для создания статических сайтов в рамках дисциплины «Web-дизайн».

В первой главах рассматриваются основы создания элементов веб-страниц с помощью стандартных тегов HTML версии 4.01. Рассмотрены теги заголовка документа, теги для форматирования текста, изображений и создания таблиц. Также приведены описания тегов для создания веб-форм.

Далее рассматривается разработка веб-страниц с использованием такого мощного инструмента, как каскадные таблицы стилей CSS. Описаны способы управления селекторами стилей, свойства стилей, отвечающие за оформление различных элементов web-страницы. Также приведены методы и способы создания web-страниц на основе блочной разметки.

Заключительные главы пособия посвящены использованию новых технологий web-дизайна, привнесенные новыми стандартами HTML 5 и CSS 3.0.

Полученные теоретические и практические навыки позволят разрабатывать статическое содержимое веб-страниц и веб-сайтов с использованием современных методов и способов web-дизайна.

Пособие соответствует типовой программе по дисциплине «Web-дизайн» и предназначено для студентов очной полной и сокращенной форм обучения.

1. СОЗДАНИЕ ВЕБ-СТРАНИЦ С ПОМОЩЬЮ ТЕГОВ HTML

1.1. История развития HTML

В 1989 году британский ученый Тимоти Бернерс-Ли предложил руководству Европейского совета по ядерным исследованиям в Женеве (CERN) проект распределенной гипертекстовой системы, которую он назвал World Wide Web (WWW) – Всемирная паутина. Первоначально идея системы состояла в том, чтобы при помощи гипертекстовой навигационной системы объединить все множество информационных ресурсов CERN в единую систему. Технология оказалась настолько удачной, что дала толчок к развитию одной из самых популярных в мире глобальных информационных систем – Интернет.

Важным компонентом технологии создания распределенной гипертекстовой системы World Wide Web стал язык гипертекстовой разметки HTML (HyperText Markup Language), разработанный Тимоти Бернерсом-Ли в 1989 – 1991 годах на основе стандарта языка разметки печатных документов – SGML (Standart Generalised Markup Language, стандартный обобщенный язык разметки).

Язык HTML позволяет размечать электронный документ, который отображается на экране с полиграфическим уровнем оформления; результирующий документ может содержать самые разнообразные метки, иллюстрации, аудио- и видеофрагменты и т.д. В состав языка вошли развитые средства для создания различных уровней заголовков, шрифтовых выделений, различные списки, таблицы и многое другое.

Первый официальным стандартом HTML стала версия HTML 2.0, принятая таковой в сентябре 1995 года. Версии HTML 1.0 официально не существует. До 1995 года существовало множество неофициальных стандартов HTML. Чтобы стандартная версия отличалась от них, ей сразу присвоили второй номер.

В 1997 году W3C (World Wide Web Concortium - Консорциум Всемирной паутины) – организация, разрабатывающая и внедряющая технологические стандарты для WWW – предложила новый стандарт HTML 3.2, за которым вскоре последовал версии 4.0. Версия 4.01, являющаяся в данный момент последней официально принятой, был одобрена как стандарт в декабре 1999 года.

С 1998 года W3C вел работу над новым языком разметки, основанном на HTML 4, но соответствующим синтаксису XML. Впоследствии новый язык получил название XHTML, первая версия которого была опубликована в 2000 году. Работа над второй версией HTML велась до 2009 года, когда было принято решение отказаться от разделения на XHTML и HTML и сосредоточить все усилия на создании новой версии HTML – версии 5.0.

В данный момент существует черновой стандарт версии HTML 5.0, который планируется доработать и принять как основной в 2014 году. Поэтому в первой части пособия будет рассмотрен язык HTML версии 4.01, а новые особенности HTML будет освещены в отдельной главе последнего раздела.

1.2. Структура HTML-документа

HTML – документ – это просто текстовый файл с расширением .html или .htm.

Язык HTML является описательным языком разметки документов, в нем используются указатели разметки (теги или метки). Теговая модель описывает документ как совокупность контейнеров, каждый из которых начинается и заканчивается (или не заканчивается) тегами. Например контейнер может начинаться с тега <head> и заканчиваться тегом </head>.

Чаще всего HTML-теги состоят из начального и конечного тега, между которыми размещаются текст и другие элементы документа. Имя конечного тега идентично имени начального, но перед именем конечного тега ставится косая черта (/). На-

пример, для тега стиля шрифта – курсив начальный тег , а конечный тег .

Теги HTML-документов в большинстве своем просты и понятны, поскольку они образованы с помощью общеупотребительных слов английского языка. По своему значению теги близки к понятию скобок «begin/end» в универсальных языках программирования. Теги определяют область действия правил интерпретации текстовых документов.

Тег может иметь атрибуты с конкретными значениями, которые устанавливаются для изменения функции тега. Атрибуты тега следуют за начальным именем тега и отделяются друг от друга одним или несколькими знаками табуляции, пробелами или символами возврата к началу строки. Порядок записи атрибутов в теге не имеет значения. Значения атрибута, если таковое имеется, следует за знаком равенства, стоящим после имени атрибута. Если значение атрибута – одно слово или число, то его можно просто указать после знака равенства, не выделяя дополнительно. Все остальные значения необходимо заключать в одинарные или двойные кавычки, особенно если они содержат несколько слов, разделенных пробелами. Длина значения атрибута ограничена 1024 символами. Регистр символов в именах тегов и атрибутов не учитывается, чего нельзя сказать о значениях атрибутов.

Например, при описании таблицы открывающий тег с атрибутами может выглядеть так:

<table width=”570” align=”center” cellpadding=”10”
cellspacing=”2” border=”2”>

Эта запись означает следующее: таблица шириной 570 пикселей, выровнена по центру, поле между рамкой и содержимым ячеек 10 пикселей, поле рамки 2 пикселя, ширина рамки 2 пикселя.

При использовании вложенных тегов в документе следует соблюдать аккуратность. Вложенные теги нужно закрывать, начиная с последнего тега.

Некоторые HTML-теги не имеют конечного компонента, поскольку являются автономными элементами. Например, тег

изображения <img>, который служит для вставки в документ графического изображения, конечного компонента не требует. К автономным тегам также относятся разрыв строки (<br>), горизонтальная линейка (<hr>) и теги, содержащие такую информацию о документе, которая не влияет на его отображаемое содержимое, например тег <meta>.

Общая схема построения контейнера в формате HTML может быть записана в следующем виде:

<имя тега список атрибутов>
содержание контейнера
</имя тега>

Кроме тегов, элементами HTML являются CER (Character Entity Reference), они предназначены для представления специальных символов в документе HTML, которые могут быть неверно обработаны браузером.

Например, чтобы представить символ «<» в документе HTML, нужно заменить его на <, а символ «>» - >. Если указать в тексте HTML строку <body>, она будет выглядеть на экране как текст <body>.

В отличие от тегов HTML, наименования CER чувствительны к регистру символов.

Пример: задать авторский знак в тексте.

© Сайт создан студентом группы ВМ-041 Ивановым И.Ю.

Записать в тексте программы:

© Сайт создан студентом группы ВМ-041 Ивановым И.Ю.

Некоторые часто использующиеся на веб-страницах CER приведены в прил. 1.

Группы тегов HTML

Все теги HTML по их назначению и области действия можно разделить на следующие основные группы:

– определяющие структуру документа;

– оформление блоков гипертекста (параграфы, списки, таблицы, картинки);

– гипертекстовые ссылки и закладки;

– формы для организации диалога;

– вызов программ.

Вот самый простой HTML-документ:

```
<html>
  <head>
  <title>
          Пример 1
  </title>
   </head>
  <body>
          <h1>
                  Привет!
          </h1>
          <P>
          Это простейший пример HTML-документа.
          </P>
  </body>
</html>
```

HTML-документ – это один большой контейнер, который начинается с тега <html> и заканчивается тегом </html>.

Контейнер HTML или гипертекстовый документ состоит из двух других вложенных контейнеров: заголовка документа (head) и тела документа (body).

1.3. Элементы разметки заголовка документа

Элемент разметки <head>. Элемент разметки <head> содержит заголовок HTML-документа. Данный элемент разметки не является обязательным. Атрибутов у тега начала контейнера нет. Контейнер заголовка служит для размещения информации, относящейся ко всему документу в целом. В контейнер <head>

вкладываются все остальные элементы разметки заголовка документа. Следует заметить, что содержимое тега <head> не отображается напрямую на веб-странице, за исключением тега <title> устанавливающего заголовок окна веб-страницы.

Элемент разметки <title>. Элемент разметки <title> служит для именования документа в сети Интернет. Иными словами назначение элемента – именование окна браузера, в котором просматривается документ. Состоит контейнер из тега начала, содержания и тега конца. Наличие тега конца обязательно. Тег не имеет специфических атрибутов.

При выборе текста для содержания контейнера <title> следует учитывать, что отображается он системным шрифтом, так как является заголовком окна браузера. В нелокализованных версиях операционных систем и графических оболочек русский текст содержания элемента <title> будет отображаться абракадаброй.

Синтаксис контейнера <title> в общем виде выглядит следующим образом:

<title> название документа </title>

Пример: задать заглавие документа в окне браузера – labour exchange.

```
<html>
<head>
  <title> labour exchange </title>
</head>
…
</html>
```

Результат работы данного тега можно увидеть на рис. 1.

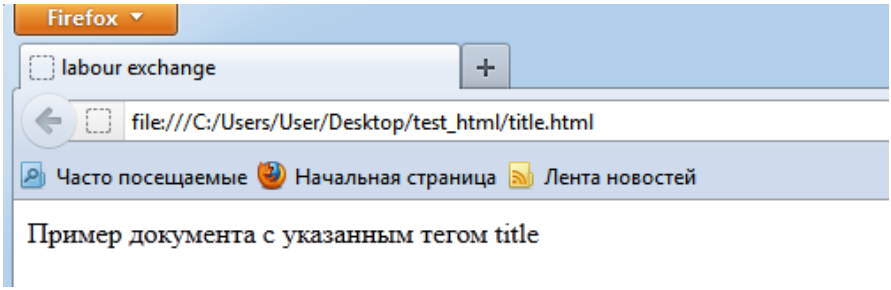

Рис. 1. Пример окна браузера Firefox с заголовком документа из тега <title>

Тег <title> выполняет множество задач, как прямо, так и косвенно. По тексту заголовка пользователь получает дополнительную информацию, что это за сайт, на котором он находится и как называется текущая страница. Большинство браузеров поддерживают возможность сохранения веб-страницы на локальный компьютер. В этом случае имя сохраненного файла совпадает с названием заголовка документа. Если в тексте заголовка содержатся символы недопустимые в имени файла (\ / : * ? " < > |), они будут проигнорированы или автоматически заменены другими дозволенными символами.

В результатах поиска по ключевым словам, поисковые системы используют заголовок страницы для указания ссылки на данный документ. Интересно написанный заголовок, содержащий ключевые слова, увеличит шансы на то, что сайт посетит больше людей.

Элемент разметки <meta>. <meta> содержит управляющую информацию, которую браузер использует для правильного отображения и обработки содержания тела документа. Механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных. Разрешается использовать более чем один метатег, все они размещаются в контейнере <head>.

В теге <meta> есть атрибуты практически совпадающие по назначению http-equiv и name, а также content и charset. Для па-

раметра name зарезервированы три наиболее используемых значения – author, description, keywords. После указания значения параметра name с помощью параметра content указывают данные для выбранной позиции. Например, для name =”author” в поле content указывают имя автора. description ответственен за описание страницы, за краткое содержание документа (многие поисковые машины проводят сортировку именно по этому тегу). keywords дает представление о темах, затрагиваемых в документе. Рекомендуется использовать и русские, и английские ключевые слова. Вот некоторые наиболее типичные meta-инструкции:

<meta name=”description” content=”Это руководство – учебник по Web-дизайну”>

<meta name=”keywords” content=”Интернет, HTML, WWW, руководство, публикация, гипертекст”>

Последняя инструкция определяет переменную keywords, содержащую набор ключевых слов, описывающих содержание документа.

Другая группа meta-инструкций определяет тип кодировки документа (ключевое слово – charset).

Следующая meta-инструкция дает браузеру указание интерпретировать загружаемый документ как содержащий HTML-текст в кодировке Windows-1251 (Названия кодировок приведены в прил. 2).

<meta http-equiv=”Content-type” content=”text/html;
charset=windows-1251”>

Для кодировки документа также предназначен отдельный атрибут charset, в котором можно указать тип кодировки. Например:

<meta charset=”utf-8”>

Практика показывает, что при индексировании можно указывать и атрибут name, и атрибут http-equiv с одинаковыми значениями (но лучше в разных тегах meta). Это связано с тем, что одни роботы индексирования анализируют содержание meta-элемента по атрибуту name, а другие – по атрибуту http-equiv. Можно использовать эти атрибуты одновременно, но для разных целей. Например:

```
<head>
  <title> Основы Web-технологий</title>
  <meta http-equiv="Content-Type" content="text/html; char-set=utf-8">
  <meta name="keywords" content="HTML, meta, метатег, тег, поисковая система">
</head>
```

Элемент разметки <style>. Данный тег предназначен для размещения описателей стилей.

Основной атрибут для этого тега – type, который задает тип описателя стиля. Это может быть либо text/css, либо text/javascript. В общем виде запись тега style выглядит так:

```
<style type=”тип_описания_стилей”>
описание стиля/стилей
</style>
```

Данный тег будет подробно рассмотрен позднее, когда речь пойдет о каскадных таблицах стилей CSS.

Элемент разметки <script>. Данный тег служит для размещения кода JavaScript или VBScript.

В общем виде запись контейнера выглядит следующим образом:

```
<script [type=тип_языка_программирования][src=URL]>
JavaScript/VBScript-код
```

</script>

Атрибут type задает тип языка программирования (основные значения – text/javascript и text/vbscript). Атрибут src задает ссылку на файл скрипта, если он подключается к документу извне.

Существует несколько скриптовых языков: JavaScript, VBScript, Jscript. По умолчанию подразумевается JavaScript.

1.4. Определитель типа документа

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, поскольку 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 языка, на которого ориентированы. В таблице приведены основные типы документов с их описанием.

Виды элемента <!DOCTYPE>

<!DOCTYPE>	Описание
<!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.

Продолжение таблицы

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">	В HTML-документе применяются фреймы.
<!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.
<!DOCTYPE html>	Документ HTML 5

Синтаксис

<!DOCTYPE [Элемент верхнего уровня] [Публичность] "[Регистрация]//[Организация]//[Тип] [Имя]//[Язык]" "[URL]">

Параметры

Элемент верхнего уровня – указывает элемент верхнего уровня в документе, для HTML это тег <HTML>.

Публичность – объект является публичным (значение PUBLIC) или системным ресурсом (значение SYSTEM), например, таким как локальный файл. Для HTML/XHTML указывается значение PUBLIC.

Регистрация – сообщает, что разработчик DTD зарегистрирован в международной организации по стандартизации (International Organization for Standardization, ISO). Принимает одно из двух значений: плюс (+) – разработчик зарегистрирован в ISO и - (минус) – разработчик не зарегистрирован. Для W3C значение ставится «-».

Организация – уникальное название организации, разработавшей DTD. Официально HTML/XHTML публикует W3C, это название и пишется в <!DOCTYPE>.

Тип – тип описываемого документа. Для HTML/XHTML значение указывается DTD.

Имя – уникальное имя документа для описания DTD.

Язык – язык, на котором написан текст для описания объекта. Содержит две буквы, пишется в верхнем регистре. Для документа HTML/XHTML указывается английский язык (EN).

URL – адрес документа с DTD.

Закрывающий тег не требуется.

1.5. Тег <body> – контейнер тела документа

Теги тела документа предназначены для управления отображением информации в браузере пользователя. Тело документа состоит из:

– иерархических контейнеров и заставок;

– заголовков различного уровня;

– блоков (параграфы, списки, формы, таблицы, изображения и т.д.);

– горизонтальных линий и адресов;

– текста, разбитого на области действия стилей (подчеркивание, выделение, курсив);

– графики, гипертекстовых ссылок и т.д.

Тело документа – контейнер <body>. Описание тегов тела документа следует начать с тега <body>, который имеет ряд атрибутов.

Атрибут background определяет фон, на котором отображается текст документа. Так, если источником для фона HTML-документа является графический файл image.jpg, то начальный тег будет выглядеть так:

<body background=”image.jpg”>

В качестве значения данного атрибута используется адрес в сокращенной форме URL. В данном случае это адрес локального файла.

В отличие от обычных изображений, для фона не устанавливаются ширина и высота, и он всегда отображается в натуральную величину с масштабом 100%. Если рисунок по размеру меньше окна браузера, то картинка повторяется по горизонтали вправо и вниз, выстраиваясь, как мозаика.

Существуют также следующие основные атрибуты тега <body>:

– bgcolor – определяет цвет фона документа;

– text – определяет цвет текста;

– link – определяет цвет непосещенных ссылок;

– vlink – определяет цвет посещенных ссылок;

– alink – определяет цвет активных ссылок.

В качестве значения данных атрибутов может использоваться английское обозначение цвета (red, white, darkblue) или константа вида #XXXXXX, определяющая цвет в терминах RGB (Red, Green , Blue) в шестнадцатеричной нотации.

Пример: установка цвета символов и фона для всего документа

```
<html>
  <body bgcolor=”#C0C0C0” text=”#0000FF”>
        …
  </body>
</html>
```

Пример: установка цвета символов и фона для всего документа, а также цвета ссылок и посещенных ссылок

```
<html>
  <body bgcolor=”#C0C0C0” text=”#0000FF”
  vlink=”#FF0000” link=”#FF0000”>
        …
  </body>
</html>
```

Замечание: данные цветовые константы представляют собой цвета Silver, Blue, Red и Lime соответственно. Некоторые варианты цветов приведены в приложении 3.

1.6. Теги для оформления текста документа

Заголовки

Заголовок обозначает начало раздела документа. В стандарте определено 6 уровней заголовков: от h1 до h6, 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.

Примеры заголовков приведены на рис. 2.

Пример написания:

<h1> Общие сведения</h1>

или

<h6> Заголовок 6-го уровня </h6>\

Заголовки очень «любят» поисковые системы, они повышают ценность текста на веб-странице, который располагается внутри тегов <h1>...<h6>. По этой причине лучше всегда использовать эти теги, несмотря на то, что с помощью стилей любой текст можно установить большого размера и, тем самым, сделать его заголовком.

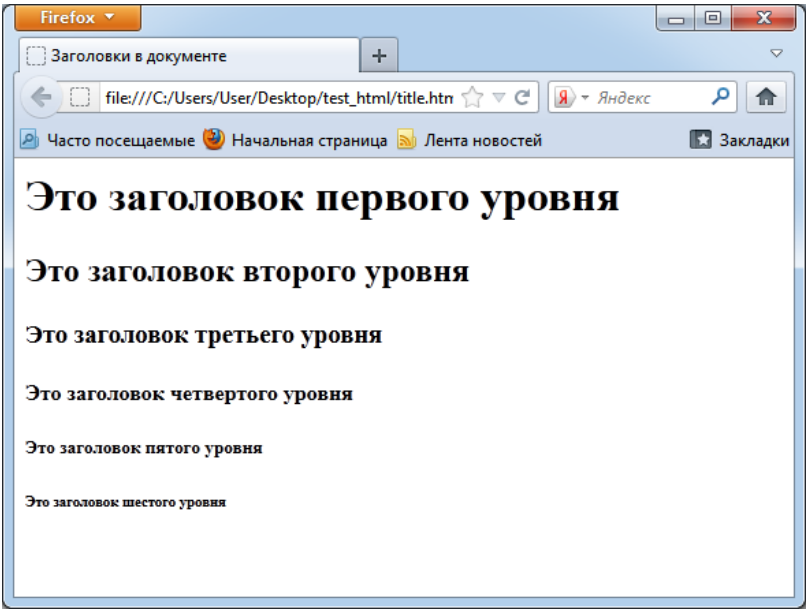

Рис. 2. Пример веб-страницы с заголовками различного уровня

Параграфы

Тег <p> применяют для разделения текста на параграфы. Данный тег является блочным элементом, всегда начинается с новой строки, абзацы текста, идущие друг за другом, разделяются между собой пустым промежутком.

Теги заголовков и параграфов имеют одинаковые атрибуты.

Атрибут align позволяет выровнять текст по левому или правому краю, по центру или ширине. По умолчанию текст выравнивается по левому краю.

Возможные значения атрибута align:

– justify – выравнивание текста по ширине.

– left – выравнивание по левому краю. Задается автоматически, при этом правый край не выравнивается.

– right – выравнивание по правому краю. Левый край не выравнивается.

– center – выравнивание по центру.

Пример: задать заголовок текста как заголовок первого уровня, выровненный по центру.

```
<html>
  …
  <body>
        …
        <h1 align="center">
        ОБУЧЕНИЕ И ПЕРЕПОДГОТОВКА КАДРОВ
        </h1
        …
  </body>
</html>
```

Пример: создать абзац, выровненный по ширине.

```
<p align="justify">
Зарегистрироваться в отделе занятости может любой
безработный от 18 лет и до пенсионного возраста.
</p>
```

**Тег разрыва строки
**. Принудительный перевод строки используется для того, чтобы нарушить стандартный порядок отображения текста. При обычном режиме интерпретации браузер пользователя отображает текст в рабочем окне, автоматически разбивая его на строки.

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

Тег запрета переноса <nobr> . Тег <nobr> (no break, без обрыва) дает браузеру команду отображать весь текст в одной строке, не обрывая ее. Если текст, заключенный в теги <nobr> </nobr>, не поместится на экране, браузер добавит в нижней части окна документа горизонтальную полосу прокрутки.

Теги управления отображением символов

Курсив, усиление, подчеркивание, верхний индекс, нижний индекс, шрифт большой, маленький, красный, синий и т.д. – все это различные варианты формы отображения.

Тег задает шрифт текста в данной области.

Атрибут size тега позволяет задать размер текста в данной области. Некоторые браузеры тег не поддерживают, поэтому желательно употреблять его только внутри текстовой области. Например, следующий тег позволит отобразить текст между тегами на один уровень крупнее обычного:

 сообщение

В языке HTML для тега <font> были определены семь основных размеров шрифта, измеряемых не в пунктах или пикселях, а в некоторых условных единицах – от 1 до 7. Как правило, обычный шрифт имеет размер «3».

Необходимо различать записи <font size=”+1”> и <font size=”1”>. В первом случае указывается относительный размер шрифта, а во втором – абсолютный. В качестве цифр можно использовать 2, 3 и т.д.

Атрибут face тега <font> позволяет задать название шрифта или список названий шрифтов, разделенный запятыми. Например:

<font face=”Verdana”>
<font face=”Arial, Times New Roman”>

Можно указывать названия семейств:

Универсальные семейства шрифтов:

serif – шрифты с засечками, типа Times;

sans-serif – рубленные шрифты (шрифты без засечек или гротески), типичный представитель – Arial;

cursive – курсивные шрифты;

fantasy – декоративные шрифты;

monospace – моноширинные шрифты, ширина каждого символа в таком семействе одинакова.

Атрибут color позволяет задать цвет символов. Например: тег <font color=”#0000FF”> сделает помещенный между тегами текст синим.

```
<body>
<meta http-equiv="Content-type" content="text/html;
charset=windows-1251">
<h1>
Всем <font color="#0000FF">привет!</font>
</h1>
</body>
```

Пример: задать заголовок текста как заголовок первого уровня, выровненный по центру и задать для заголовка свой цвет, отличный от цвета остального текста:

```
<h1 align=”center”> <font color=”#FF0000”>
Муниципальные отделы занятости </font> </h1>
```

Теги <big> и <small> – управляют изменением размеров шрифтов. Текст, расположенный между тегами <big> </big> или <small> </small>, будут соответственно, больше или меньше стандартного на единицу (как и в случае тега <font> используются семь основных размеров шрифта). Допускается применение вложенных тегов <big> (<small>), при этом размер шрифта будет больше (меньше) с каждым уровнем.

С помощью тегов <sup> и <sub> можно задавать верхние и нижние индексы любого размера. Например:

```
<p> Встреча состоится в 19 <sup> 00 </sup> </p>
```

Существуют следующие теги для задания стиля отображения шрифта:

… – делает шрифт курсивным;

… – делает шрифт жирным;

<s>…</s> – перечеркивает текст. Также используется полная форма написания тега <strike>…</strike>;

<u>…</u> – подчеркивает текст.

Пример: задать для текста полужирный шрифт и шрифт с подчеркиванием.

 <u> Регистрация. </u>

Создание списков

Списки являются важным средством структурирования текста. В HTML имеются следующие основные виды списков:

– ненумерованный список (ul);

– нумерованный список (ol).

Маркированный список записывается в виде последовательности тегов:

```
<ul>
  <li>первый элемент списка</li>
  <li>второй элемент списка</li>
  <li>третий элемент списка</li>
</ul>
```

Теги <ul> и - это теги начала и конца ненумерованного списка, тег <li> (list item) задает тег элемента списка.

Пример: создать абзац, имеющий маркированный список. Абзац имеет персональный цвет, отличный от цвета остального текста. Абзац выровнен по ширине. Например, следующий:

<p align=”justify”>

Пособие по безработице начисляется в соответствии с законом "О занятости". При этом учитывается:

<ul type=”disc”>

сколько времени гражданин находится без работы;
по каким причинам гражданин уволился;
в какой раз гражданин становится на учет.

</p>

Атрибут type задает вид маркеров для данного списка:

<ul type=”disc”> тег создает маркеры в виде закрашенных окружностей;.

<ul type=”circle”> тег создает маркеры в виде не закрашенных окружностей.

<ul type=”square”> тег создает сплошные квадратные маркеры.

Нумерованные списки. Тег <ol> вместе с атрибутом type позволяет создавать нумерованные списки, используя в качестве номеров не только обычные числа, но и строчные и прописные буквы, а также строчные и прописные римские цифры.

Используются следующие значения атрибута type:

<ol type=”1”> тег создает список с нумерацией в формате 1., 2., и т.д.

<ol type=”A”> создает список с нумерацией в формате A., B., C. и т.д.

<ol type=”a”> тег создает список с нумерацией в формате a., b., c. и т.д.

<ol type=”I”> тег создает список с нумерацией в формате I., II., III. и т.д.

Для создания нумерации, начинающейся не с единицы, используют атрибут start. Например, запись <ol start=”10”> вызовет нумерацию элементов списка, начиная с числа 10. Далее пойдут элементы с номерами 11, 12 и т.д. При этом не имеет значения, какой тип списка установлен с помощью type, атрибут start одинаково работает и с римскими и с арабскими числами.

Списки определений

Список определений несколько отличается от других видов списков. Вместо меток <li> в списках определений используются метки <dt> (от английского definition term – определяемый термин) и <dd> (от английского definition definition – определение определения). Весь список определений заключается в теги <dl> и </dl>.

Например, необходимо создать следующий фрагмент текста:

Список определений:
Провайдер
Организация, предоставляющая услуги Интернет
Хост
Узел глобальной Сети
Зеркало
Сервер, являющейся копией другого популярного сервера

Код веб-документа, создающий вышеуказанный текст, имеет следующий вид:

```
<html>
<body>
Список определений:
<dl>
  <dt> Провайдер
  <dd> Организация, предоставляющая услуги Интернет
  <dt> Хост
          <dd> Узел глобальной Сети
  <dt> Зеркало
  <dd> Сервер, являющейся копией другого популярного сервера
</dl>
</body>
</html>
```

Теги блоков текста

Кроме тегов параграфа, заголовков, шрифтов, существуют еще два тега позволяющие выделить в документе блок или фрагмент текста.

Тег используется для выделения блока текста, который следует рассматривать как отдельную логическую единицу (или отдельный абзац). Данный тег имеет атрибут align (как параграф), а также атрибуты для привязки сценариев и стилей к блоку текста.

Тег используется для выделения слов или строк текста. Он весьма похож на тег , но текст, выделяемый тегом не образует отдельный параграф или абзац.

Основное достоинство данных тегов заключается в том, что с их помощью можно легко и удобно применять стили к различным блокам и фрагментам содержимого. В дальнейшем, при изучении стилей, данные теги будут одними из самых часто встречаемых на веб-страницах.

Тег выравнивания по центру

Для выравнивания текста или изображения по центру страницы или ячейки таблицы можно использовать тег . Например:

текст, выровненный по центру

Тег бегущей строки

Тег <marquee> предназначен для создания текстовой строки, которая может перемещаться в выделенной для нее области (например справа налево по горизонтали).

Основные атрибуты тега:

– behavior – управляет движением бегущей строки по области вывода. Имеет следующие значения:

– alternate – текст бегущей строки безостановочно движется то в одном, то в противоположном направлении, «отражаясь» от краев области вывода и меняя направление движения.

– scroll (значение по умолчанию) – текст постоянно движется в одну сторону до края области вывода, где исчезает, а потом появляется с другой стороны.

– slide – текст бегущей строки пересекает область вывода и останавливается, когда первый символ достигает противоположной стороны области вывода.

direction – задает направление бегущей строки. По умолчанию равен left. Также может иметь значения right, up и down.

– bgcolor – цвет фона бегущей строки.

– height и width – высота и ширина полосы вывода бегущей строки.

– scrolldelay – устанавливает величину задержки в миллисекундах между последовательными выводами текста в бегущей строке. Большинство браузеров не делают задержку вывода менее 60 миллисекунд. Для того, чтобы это ограничение снять, следует добавить атрибут truespeed.

Тег горизонтальной линии

Тег <hr> Рисует горизонтальную линию, которая по своему виду зависит от используемых параметров, а также браузера. Линия всегда начинается с новой строки, а после нее все элементы отображаются на следующей строке. Для настройки вида линии используются следующие атрибуты:

– align – определяет выравнивание линии (имеет три значения: left, center, right);

– color – определяет цвет линии;

– noshade – рисует линию без трехмерных эффектов (значение у атрибута отсутствует);

– size – толщина линии в пикселях;

– width – ширина линии в пикселях или процентах (при значении 100% в качестве ширины берется ширина родительского элемента.).

1.7. Теги для логического выделения фрагментов текста

Теги, описываемые в данном разделе, в аспекте отображения содержимого в окне браузера практически эквивалентны некоторым тегом, описанным ранее. Например, теги и выделяют текст жирным и курсивным шрифтов соответственно, что аналогично тегам <b> и <i>.

Но теги <b> и , также как <i> и , несмотря на сходный результат, не совсем эквивалентны и заменяемы. Первый тег <b> – относится к тегам физической разметки и устанавливает жирный текст, а тег – является тегом логической разметки и определяет важность помеченного текста. Такое разделение тегов на логическое и физическое форматирование изначально предназначалось, чтобы сделать HTML универсальным, в том числе не зависящим от устройства вывода информации. Однако получилось так, что в популярных браузерах результат использования этих тегов равнозначен.

**Тег ** предназначен для акцентирования текста, придания ему наибольшей важности. Браузеры отображают такой текст жирным начертанием.

**Тег ** предназначен для акцентирования текста, придания ему относительно средней важности. Браузеры отображают такой текст курсивным начертанием.

Тег <cite> помечает текст как цитату или сноску на другой материал. Такое выделение удобно для изменения стиля текста через CSS, а также применяется для разделения кода HTML на структурные элементы. Браузеры обычно устанавливают текст внутри контейнера <cite> курсивом.

Тег <abbr> указывает, что последовательность символов является аббревиатурой. По умолчанию, текст заключенный в контейнере <abbr> подчеркивается пунктирной линией. Данный тег полезно использовать вместе с атрибутом title, с помощью которого дается расшифровка сокращения, что позволяет понимать аббревиатуру тем людям, которые с ней не знакомы. Кроме того, поисковые системы индексируют полнотекстовый ва-

риант сокращения, что может использоваться для повышения рейтинга документа. Пример:

<abbr title="Common Gateway Interface, общий шлюзовый интерфейс">CGI</abbr>

обозначается протокол, с помощью которого любые внешние программы взаимодействуют с веб-сервером. С помощью CGI на сервере можно выполнять программы на любом языке программирования и результат их действия выводить в виде веб-страницы.

Результат использования данного примера приведен на рис. 3

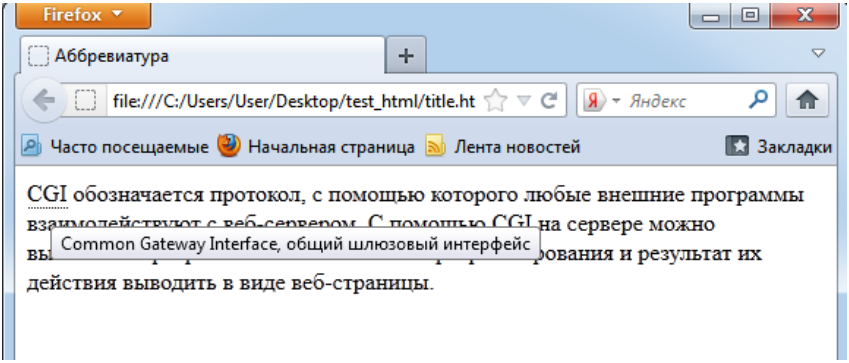

Рис. 3. Пример веб-страницы с тегом <abbr>

Тег <acronym> указывает на то, что текст является акронимом. В отличие от аббревиатуры, акроним – это устоявшееся сокращение, которое применяется как самостоятельное слово (произносимое как единое целое, а не побуквенно). К акронимам, например, можно отнести следующие слова: БАМ, НАТО, вуз, DOS и др.

По умолчанию, текст заключенный в контейнере <acronym> подчеркивается пунктирной линией. Внешний вид выделенного фрагмента такой же, как и у тега <abbr>.

Тег <address> предназначен для хранения информации об авторе и может включать в себя любые элементы HTML вроде ссылок, текста, выделений и т.д. Подразумевается, что поисковые системы будут анализировать содержимое этого тега для сбора информации об авторах сайтов.

По умолчанию текст внутри контейнера <address> отображается курсивным начертанием.

Тег <bdo> устанавливает направление вывода текста и преимущественно предназначен для использования с языками, где чтение происходит справа налево. Например, к ним относится арабский язык. Для указания направления используется атрибут dir имеющий два значения: rtl (right-to-left – справа налево) и ltr (left-to-right – слева направо). Пример (результат на рис. 4):

```
<p><bdo dir="ltf">Нормальный текст</bdo></p>
<p><bdo dir="rtl">Перевернутый текст</bdo></p>
```

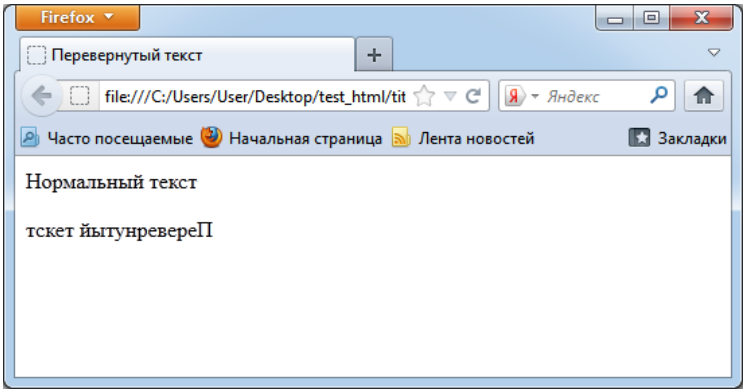

Рис. 4. Пример использования тега <bdo>

Тег <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 промежутком сверху и снизу.

Тег <code> предназначен для отображения одной или нескольких строк текста, который представляет собой программный код. Сюда относятся имена переменных, ключевые слова, операторы и т.д. Браузеры обычно отображают содержимое контейнера <code> как моноширинный текст уменьшенного размера.

Следует заметить, что дополнительные пробелы внутри контейнера <code> не учитываются, так же, как и переносы текста. Поэтому используйте тег
 или <p> для создания переносов.

Тег <del> 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 <del> как перечеркнутый. Аналогичный эффект дает тег <s> или <strike>.

Тег <dfn> применяется для выделения новых (предположительно незнакомых читателю) терминов при их первом появлении в тексте. Браузеры отображают содержимое контейнера <dfn> с помощью курсивного начертания.

Тег <ins> предназначен для выделения текста, который был добавлен в новую версию документа. Подобное форматирование позволяет отследить, какие изменения в тексте документа были сделаны. Браузеры обычно помечают текст в контейнере <ins> как подчеркнутый.

Тег <kbd> используется для обозначения текста, который набирается на клавиатуре или для названия клавиш. Браузеры обычно помечают текст в контейнере <kbd> моноширинным шрифтом.

Тег <pre> определяет блок предварительно форматированного текста. Такой текст отображается обычно моноширинным шрифтом и со всеми пробелами между словами. По умолчанию, 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. Внутри контейнера <pre> допустимо применять любые

теги кроме следующих: <big>, <img>, <object>, <small>, <sub> и <sup>. Пример (результат на рис. 5):

```
<pre>
Раздел 1
        Глава 1.1
        Глава 1.2
        Глава 1.3
Раздел 2
        Глава 2.1
        Глава 2.2
</pre>
```

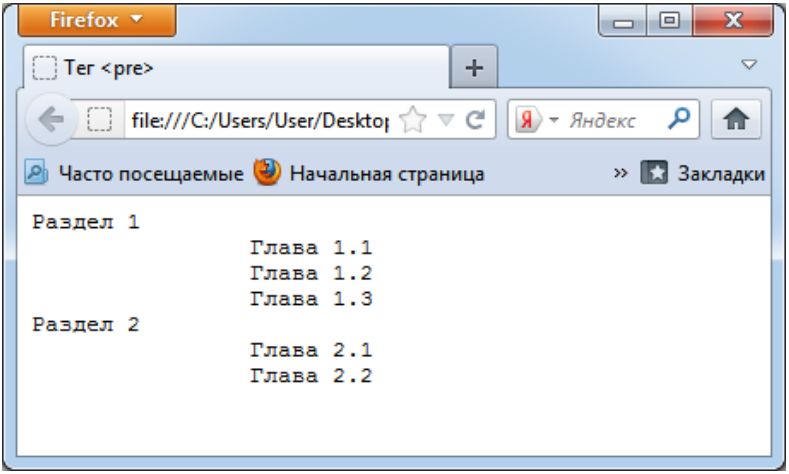

Рис. 5. Пример использования тега <pre>

Тег <q> используется для выделения в тексте цитат. Содержимое контейнера автоматически отображается в браузере в кавычках.

Тег <samp> используется для отображения текста, который является результатом вывода компьютерной программы

или скрипта. Браузеры обычно отображают текст в контейнере <samp> с помощью моноширинного шрифта.

Тег <var> используется для выделения переменных компьютерных программ. Браузеры обычно помечают текст в контейнере <var> курсивным начертанием.

Тег <xmp> отображает содержимое контейнера «как есть» и шрифтом фиксированной ширины. Пока тег <xmp> не закрыт, все теги внутри него отображаются как обычный текст.

Комментарии

Комментарии начинаются с символа <!-- и заканчиваются символом -->. Комментарии могут состоять из нескольких строк текста. Вложенные комментарии (когда один комментарий расположен внутри другого) не допустимы.

1.8. Теги для оформления таблиц

Создание таблицы. Для описания таблиц используется тег <table>. Тег <table>, как и многие другие, автоматически переводит строку до и после таблицы.

Тег <tr> (table row, строка таблицы) создает строку таблицы. Весь текст, другие теги и атрибуты, которые требуется поместить в одну строку, должны размещаться между тегами <tr> и </tr>.

Внутри строки таблицы обычно размещаются ячейки с данными. Каждая ячейка, содержащая текст или изображение, должна быть окружена тегами <td> и </td>. Число тегов <td> </td> в строке определяет число ячеек.

Пример создания простейшей таблицы.

```
<html>
<head>
<title> Создание таблицы</title>
</head>
<body>
<h1>Простейшая таблица</h1>
<table border=”2”>      <!--Это начало таблицы-->
```

```
<caption>                    <!--Это заголовок таблицы-->
ВГТУ
</caption>
<tr>                <!--Это начало первой строки-->
  <td>    <!--Это начало первой ячейки-->
          Факультет
  </td>   <!--Это конец первой ячейки-->
  <td>    <!--Это начало второй ячейки-->
          Специальности
  </td>   <!--Это конец второй ячейки-->
</tr>               <!--Это конец первой строки-->

<tr>                <!--Это начало второй строки-->
  <td>    <!--Это начало первой ячейки-->
          ФАЭМ
  </td>   <!--Это конец первой ячейки-->
  <td>    <!--Это начало второй ячейки-->
          ВМ
  </td>   <!--Это конец второй ячейки-->
</tr>               <!--Это конец второй строки-->
</table>            <!--Это конец таблицы-->
</body>
</html>
```

Данный HTML-код создаст следующую таблицу (рис. 6).

Простейшая таблица

ВГТУ	
Факультет	Специальности
ФАЭМ	ВМ

Рис. 6. Пример простейшей таблицы

Заголовки столбцов таблицы. Заголовки для столбцов и строк таблицы задаются с помощью тега заголовка <th> </th>

(Table Header, заголовок таблицы). Эти теги подобны тегам <td> </td>. Отличие состоит в том, что текст, заключенный между тегами <th> </th>, автоматически записывается жирным шрифтом и по умолчанию располагается посередине ячейки. Если воспользоваться <td></td> с тегом <b> и атрибутом <align=”center”>, текст тоже будет выглядеть как заголовок. Однако следует иметь в виду, что не все браузеры поддерживают в таблицах жирный шрифт, поэтому лучше задавать заголовки таблиц с помощью <th>.

Пример. Создать таблицу, в которой задать заголовок первого столбца и общий заголовок для второго, третьего и четвертого столбцов.

```
<table border=”4”>
<tr>
  <th> Шифр <br>группы</th>
  <th>Фамилия</th>
  <th>Имя</th>
  <th>Отчество</th>
</tr>
<tr>
  <td>ВМ-051  </td>
  <td> Антонов</td>
  <td> Юрий </td>
  <td> Михайлович</td>
</tr>
</table>
```

В результате написания данного HTML-кода будет создана следующая таблица (рис. 7):

Шифр группы	Фамилия	Имя	Отчество
ВМ-051	Антонов	Юрий	Михайлович

Рис. 7. Пример таблицы с заголовками столбцов таблицы

Заголовок таблицы. Таблица может иметь заголовок (<caption>…</caption>), хотя заголовок не является обязательным. Тег <caption> может включать атрибут align. Допустимые значения: top (заголовок помещается над таблицей) и bottom (заголовок помещается под таблицей). Заголовок может состоять из любого текста и изображений. Текст будет разбит на строки, соответствующие ширине таблицы. Иногда тег <caption> используется для подписи под рисунком. Для этого достаточно описать таблицу без границ.

Пример. Создать таблицу, имеющую заголовок над таблицей.

```
<table border=”6”>
<caption align=”top”> Факультеты </caption>
<tr>
  <td> ФАЭМ</td>
  <td> РТФ</td>
  <td> ИЭФ</td>
</tr>
</table>
```

Данный HTML-код создаст следующую таблицу (рис. 8):

Факультеты

ФАЭМ	РТФ	ИЭФ

Рис. 8. Пример таблицы с заголовком

Атрибуты тегов таблиц. Теги, работающие с таблицами, имеют следующие атрибуты.

Атрибут nowrap. Обычно любой текст, не помещающийся в одну строку ячейки таблицы, переходит на следующую строку. Однако при использовании атрибута nowrap с тегами <th> или <td> длина ячейки расширяется настолько, чтобы за-

ключенный в ней текст поместился в одну строку. Примеры записи атрибута:

<th nowrap> </th>
<td nowrap> </td>

Атрибут colspan. Теги <td> и <th> модифицируются с помощью атрибута colspan (column span, соединение столбцов). Устанавливает число ячеек, которые должны быть объединены по горизонтали. Этот атрибут имеет смысл для таблиц, состоящих из нескольких столбцов. Например, colspan=”3” означает, что ячейка простирается на три колонки.

Атрибут rowspan. Данный атрибут используется в тегах <td> и <th> и устанавливает число ячеек, которые должны быть объединены по вертикали. Этот атрибут имеет смысл для таблиц, состоящих из нескольких строк.

Атрибут width. Устанавливает ширину таблицы, группы ячеек или ячейки. Ширину таблицы можно задать в пикселях (например, <table width=”400”>) или в процентах от ширины страницы (например, <table width=”80%”>) или в процентах от ширины таблицы, если задается ширина ячейки (например, <td width=”50%”>). В первом случае, задают таблицу шириной 400 пикселей независимо от размера страницы на мониторе. Во втором случае таблица будет занимать 80 % ширины страницы при любом размере изображения на экране. В третьем случае задают ширину ячейки, которая будет занимать 50 % ширины таблицы.

Атрибут cellpadding. Определяет ширину пустого пространства между содержимым ячейки и ее границами, то есть задает поля внутри ячейки. Например: <table cellpadding=”3”>.

Атрибут cellspacing. Определяет ширину промежутков между ячейками в пикселях. Если этот атрибут не указан, по умолчанию задается величина, равная двум пикселям. Например: <table cellspacing=”6”>.

Атрибуты align и valign. Теги <tr>, <td>, <th> имеют атрибуты align и valign. Атрибут align устанавливает выравнива-

ние текста и графики по горизонтали. Допустимые значения: left (выравнивание по левому краю), center (выравнивание по центру), right (выравнивание по правому краю). Атрибут valign устанавливает вертикальное выравнивание текста в ячейках строки. Допустимые значения: top (выравнивание по верхнему краю), middle (выравнивание по центру), bottom (выравнивание по нижнему краю).

Атрибут border. Устанавливает ширину внешней рамки таблицы и ячеек в пикселях (например, <table border= ”4”>). Если атрибут не установлен, таблица показывается без рамки <table border=”0”>.

Если ячейка таблицы пуста, вокруг нее не рисуется рамка. Если ячейка пуста, а рамка нужна, в ячейку можно ввести символьный объект   (non-breaking space – неразрывный пробел). Ячейка по-прежнему будет пустой, а рамка вокруг нее появится.

Атрибут bgcolor. Позволяет установить цвет фона. В зависимости от того, с каким тегом (table, tr, td) он применяется, цвет фона может быть установлен для всей таблицы (<table bgcolor=”green”>), для строки (<tr bgcolor=”green”>) или для отдельной ячейки (<td bgcolor=”green”>). Значением данного атрибута является RGB-код или стандартное название цвета.

Рамка таблицы не обязательно может быть серой. Для этого служит **атрибут bordercolor**, которому можно дать в качестве значения любой доступный цвет. Например, bordercolor=”green”.

Атрибут background задает фоновое изображение для таблиц. Применим к тегам table и td. Его значением является имя файла с фоновым изображением.

Любая ячейка таблицы может содержать в себе другую таблицу.

Пример. Создать таблицу, имеющую объединенные ячейки.

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

```
<table width="50%" border="3" cellspacing="6" cellpadding="20" align="center" bordercolorlight="lime" bgcolor="white">
<tr>
  <td width="33%"> это первая ячейка</td>
  <td width="33%"> это вторая ячейка</td>
  <td rowspan="3"> это три ячейки третьего столбца объединились в одну большую ячейку </td>
</tr>
<tr>
  <td colspan="2"> это единственная ячейка второй строки, объединяющая оба столбца</td>
</tr>
<tr>
  <td> это первая ячейка третьей строки </td>
  <td> это вторая ячейка третьей строки</td>
</tr>
</table>
</body>
</html>
```

Данный HTML-код создаст следующую таблицу (рис. 9).

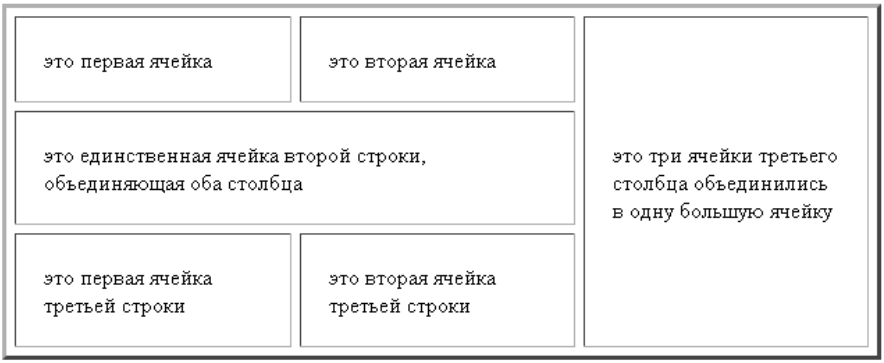

Рис. 9. Пример таблицы с объединением ячеек по горизонтали и вертикали

Таблицы могут иметь такие элементы оформления, как заголовок и заключение. Можно все основное содержание таблицы обозначить тегом <tbody>. Тогда заголовок таблицы обозначается тегом <thead>, а заключение – тегом <tfoot>. Все три тега употребляются с закрывающими тегами и могут иметь атрибуты bgcolor, align, valign.

Добавим в предыдущий пример заголовок и заключение.

```
<html>
<body>
<table width="50%" border="3" cellspacing="6" cellpadding="20" align="center" bgcolor="white">
<thead bgcolor="Aqua">
<tr>
  <th colspan="3"> Учебная таблица</th>
</tr>
</thead>
<tbody>
<tr>
  <td width="33%">
        это первая ячейка
  </td>
  <td width="33%">
        это вторая ячейка
  </td>
  <td rowspan="3">
это три ячейки третьего столбца объединились в одну большую ячейку
  </td>
</tr>
<tr>
  <td colspan="2">
это единственная ячейка второй строки, объединяющая оба столбца
  </td>
</tr>
```

```
<tr>
  <td>
        это первая ячейка третьей строки
  </td>
  <td>
        это вторая ячейка третьей строки
  </td>
</tr>
</tbody>
<tfoot bgcolor="yellow">
<tr>
  <td colspan="3" align="center">
        Итоговая строка
  </td>
</tr>
</tfoot>
</table>
</body>
</html>
```

Данный HTML-код создаст следующую таблицу (рис. 10):

Рис. 10. Пример таблицы с добавленным заголовком и заключением

1.9. Создание и оформление гиперссылок

В HTML переход от одного фрагмента текста к другому задается с помощью метки вида:

 фрагмент текста или какой-либо другой объект

В качестве параметра [адрес перехода] может использоваться несколько типов аргументов. Самое простое – это задать имя другого HTML-документа, к которому нужно перейти. Например, из текущего файла необходимо перейти к файлу с именем p1.html:

 На первую

Такой фрагмент HTML-текста приведет к появлению в документе выделенного фрагмента «На первую», при нажатии на который в текущее окно будет загружен документ p1.html.

Если в адресе перехода не указан каталог, переход будет выполнен внутри текущего каталога. Если в адресе перехода не указан сервер, переход будет выполнен на текущем сервере.

Из этого следует одно очень важное практическое соображение. Если группа HTML-документов ссылается друг на друга только по имени файла и находится в одном каталоге, то можно переместить ее в другой каталог и на другой компьютер, где она будет также работать.

Гиперссылка может ссылаться на документ, хранимый в Интернете. Например:

Справка

Содержание контейнера гипертекстовой ссылки, заключенное между тегом начала и тегом конца, выделяется в тексте цветом, определенным для контекстных гипертекстовых ссылок в атрибутах тега <body>.

Другой формой использования тега <a> является определение точек внутри текста (анкеров, якорей), на которые можно ссылаться.

Для создания якоря задают оператор:

Текст, который помечается точкой

Например:

ВГТУ

Слова «ВГТУ» при этом никак не будут выделены в тексте документа.

Для задания перехода на точку (якорь) в месте перехода создают гиперссылку следующим образом:

гиперссылка

Например, переход на точку с именем ААА:

На начало

Пример фрагмента программы, в которой слово «На начало» ссылается на слово «ВГТУ».

```
<html>
<!-- Простейшая программа-->
<head>
  <meta http-equiv="Content-type" content="text/html;
          charset=windows-1251">
<title>
Пример ссылок на фрагменты текста
</title>
</head>
<body>
```

```
<h1>
<a name="aaa">ВГТУ</a>
</h1>
<p> Факультеты ВГТУ</p>
<ol>
  <li>ФАЭМ</li>
  <li>ЕГФ</li>
  <li>РТФ</li>
</ol>
<a href="#aaa">На начало</a>
</body>
</html>
```

Если необходимо перейти к определенному слову в другом файле, то гиперссылку оформляют следующим образом:

```
<a href=”2.html#AAA”>Переход на точку</a>
```

Во избежание недоразумений рекомендуется задавать имена якорей латинскими буквами. Большинство браузеров отличают прописные и строчные буквы. Если имя якоря определено как “AAA”, ссылка на якорь “aaa” или “AaA” не переведет на нужный анкер.

Возможны ссылки не только на HTML-документы, но и на другие виды ресурсов:

```
<a href=”ftp://server/directory/file.ext”> Загрузка файла </a>
```

Такая ссылка, если ей воспользоваться , запустит протокол передачи файлов и начнет загрузку файла file.ext, находящегося в каталоге directory на сервере server, на локальный диск пользователя.

```
<a href=”mailto:user@mail.box”> Послать письмо </a>
```

Если пользователь совершит переход по такой ссылке, у него на экране откроется окно ввода исходящего сообщения его почтовой программы. В строке To: (“Куда») окна почтовой программы будет указано user@mail.box.

Чтобы не загружать ссылочную страницу в том же окне браузера, в которое была загружена текущая страница, а открыть ее в новом окне, нужно в теге <a> установить атрибут target= со значением «_blank». Значение «_self» - загружает в то же окно, в котором был сделан щелчок на гиперссылке. Значение “_self” определено по умолчанию. Например:

```
<a href="1.htm"  target="_blank">  На первую </a>
```

1.10. Добавление и оформление изображений

Для того, чтобы вставить в web-страницу изображение, необходимо иметь изображение в формате gif (файл с расширением *.gif) или jpeg (файл с расширением *.jpeg или *.jpg). Может также использоваться изображение формата PNG (*.png).

Вставка изображения осуществляется с помощью тега <img>, который не имеет закрывающего тега. Пример включения в документ изображения из файла pic.gif, находящегося в одном каталоге с HTML-документом:

```
<img src=”pic.gif”>
```

Атрибут src определяет имя файла. Кроме того, может быть атрибут alt=”текст”, содержащий альтернативный текст (Еще можно добавить атрибут title для появления всплывающей подсказки). Например:

```
<img src=”pic.gif” alt=”Картинка“>
<img src="people3.jpg" alt="Совещание">
```

Изображения на Web-странице могут использоваться как гиперссылки. Для обозначения изображения как гиперссылки используют тег <a>, где между <a> и вставляется тег изображения <img>:

```
<a href=”адрес файла или изображения”>
  <img src= ”image.gif”>
</a>
```

При этом изображение, используемое в качестве гипертекстовой ссылки, обводится дополнительной рамкой. Например:

```
<a href="1.htm">
  <img src="people4.jpg" alt="На первую">
</a>
```

Пример: задать заголовок текста как заголовок первого уровня и рядом с ним картинку, являющуюся ссылкой на первую страницу

```
<h1 align=center> УЧЕБНЫЙ ЦЕНТР "КАДРЫ"
  <a href="str1.html">
          <img src="people3.jpg" alt="На первую">
  </a>
</h1>
```

Тег изображения имеет один обязательный атрибут src и необязательные атрибуты.

Атрибут src указывает файл изображения и путь к нему; изображение должно быть загружено в браузер и размещено в том месте документа, где расположен тег изображения.

Атрибут alt позволяет указать текст, который будет выводиться вместо изображения браузерами, в настройках которых отключен показ изображений. В некоторых случаях при недостаточной пропускной способности линий связи пользователи

отключают отображение графики. Наличие названий вместо картинок облегчает восприятие web-страниц в таком режиме.

Атрибут align определяет положение изображения относительно окружающего его текста. Основные значения атрибута:

left – определяет огибаемое текстом изображение. Изображение располагается вдоль левой границы документа, а последующие строки текста огибают его справа;

right – определяет огибаемое текстом изображение. Изображение располагается вдоль правой границы документа, а последующие строки текста огибают его слева;

top – выравнивает верх изображения по верхнему краю самого высокого элемента в строке окружающего текста.

middle – выравнивает центр изображения по базовой линии строки окружающего текста.

bottom – выравнивает нижний край изображения по базовой линии строки окружающего текста.

Кроме основных значений атрибута существует еще ряд аргументов, которые расширяют возможности взаимного размещения графики и текста.

Атрибут border определяет толщину рамки вокруг изображения в пикселях. Если значение равно нулю, рамка отсутствует. Чтобы не вводить пользователей в заблуждение, не стоит задействовать атрибут border в изображениях, которые представляют собой часть ссылки, поскольку рисунки, применяемые в качестве гиперссылок, обычно выделяются цветной рамкой.

Атрибут hspace задает горизонтальное расстояние в пикселях между вертикальной границей страницы и изображением, а также между изображением и огибающим его текстом.

Атрибут vspace задает вертикальное расстояние между строками текста и изображением в пикселях.

Атрибут width и height задают значения размеров изображения в пикселях (или процентах от величины родительского элемента) по горизонтали и по вертикали соответственно.

Пример применения аргументов.

```
<img src=”image.gif” alt=”изображение” width=”100”
  height=”200” hspace=”10” vspace=”10” border=”2”
  align=”left”>
```

Создание карты изображения. Карта изображения – это изображение с так называемыми активными областями, которые ссылаются на URL других страниц или узлов.

Порядок создания активного изображения следующий.

1. Вставить изображение на страницу с атрибутом usemap, который должен содержать имя карты графических ссылок с предварительным знаком #. Когда карта еще не создана, ей можно дать любое имя. Например:

```
<img src="people3.jpg" width="77" height="109" border="0"
  alt="совещание" usemap="#pip">
```

2. Создать карту графических ссылок, которая начинается с тега <map> и завершается закрывающим тегом </map>. В теге <map> следует установить единственный возможный атрибут name. Его значением должно быть имя карты. Например:

```
<map name="map1">
```

3. Между тегами <map> и </map> должна находится основная часть карты. Она состоит из нескольких тегов <area>, каждый из которых определяет какую-то активную область рисунка. При щелчке мышью на «активной области» происходит переход по гиперссылке.

4. «Активные области» могут быть прямоугольной формы, либо круглой, либо многоугольной. Это определяется установкой значения атрибута shape в теге <area> (он используется без закрывающего тега). Если значение этого атрибута – ”rect” , область будет прямоугольной, если “circle” – круглой, а если “poly” – многоугольной.

5. Для каждой создаваемой области нужно определить координаты области и записать их как значения атрибута coords. Если «активная область» прямоугольная, то следует указать ко-

ординаты левого верхнего ее угла и правого нижнего. Местоположение любой точки на рисунке может быть определено с помощью двух чисел – ее горизонтальной и вертикальной координат. Самая левая верхняя точка на рисунке имеет координаты «0,0» . Чем больше горизонтальная координата, тем правее расположена точка, и, соответственно, чем больше вертикальная координата, тем точка ниже. Например, если рисунок имеет размер 400x300 точек и нужно определить прямоугольную активную область, занимающую его левый верхний угол и имеющую половину ширины и высоты рисунка, то необходимо написать следующее:

```
<area shape=”rect” cords = ”0,0,200,150”>
```

Если «активная область» круглая, то нужно задать координаты центра окружности, а также указать ее радиус, например:

```
<area shape=”circle” cords = ”200,150,40”>
```

Такая запись определяет круглую область, расположенную в центре рисунка и имеющую окружность радиусом 40.

Если «активная область» - многоугольная, то необходимо указать координаты всех углов многоугольника.

6. В теге <area> необходимо установить атрибут href, определяющий гиперссылку.

7. Полезно определить также атрибут alt для появления всплывающего текста, поясняющего, куда будет происходить переход.

Пример создания двух «активных областей» на изображении размером 77 на 109.

```
<html>
<body>
<h1>
Создание карты-изображения
</h1>
<map name="mymap">
```

```
<area alt="файл p1" shape="rect" coords="0,0,35,50" href="p1.html">
<area alt="файл p2" shape="rect" coords=35,0,77,109" href="p2.html">
</map>
<img src="people3.jpg" width="77" height="109" border="0" alt="совещание" usemap="#mymap">
</body>
</html>
```

1.11. Добавление на страницу различных объектов

Тег <object>

Элемент <object> сообщает браузеру, как загружать и отображать объекты, которые исходно браузер не понимает. Как правило, такие объекты требуют подключения к браузеру специального модуля, который называется плагин, или запуска вспомогательной программы.

Спецификация HTML 4 разрешает вкладывать несколько тегов <object> с разным содержанием друг в друга. Это позволяет отображать тот контент, который понимает браузер, при отсутствии нужного плагина. Например, внешний тег <object> загружает видеофайл, а для случая, когда соответствующий кодек (программа для сжатия и восстановления видеоданных) не установлен, внутренний тег <object> показывает графическое изображение. Дополнительно внутрь контейнера <object> можно поместить тег <param>, который передает дополнительные параметры для отображения объекта.

Среди атрибутов тега object можно отметить следующие.

Атрибут archive позволяет задать произвольное число файлов необходимых для корректной работы объекта или для его отображения. Файлы перечисляются через пробел вместе с полным или локальным адресом.

Атрибут classid указывает браузеру, какую программу, плагин или апплет нужно загрузить. Можно также использовать

идентификатор зарегистрированного ActiveX приложения, предваряя его ключевым словом clsid.

Атрибут data определяет файл, который следует отобразить в окне браузера. Для популярных форматов данных достаточно указать путь к файлу и его тип (атрибут type) для загрузки и демонстрации результата. Для специфичных плагинов желательно еще включить атрибут classid.

Атрибут height устанавливает высоту объекта, а **атрибут width** – его ширину. В заданные размеры входит не только само изображение, например в случае воспроизведения видеофайла, но и панель управления им, включая кнопки проигрывания, паузы, остановки и т.д. По этой причине на размер отображаемого объекта влияет тип файла и применяемый плагин.

Атрибут type сообщает браузеру о типе объекта, который указан в атрибуте data. Браузер может использовать эту информацию, чтобы подготовить необходимые ресурсы для воспроизведения файла. Значением атрибута выступает Имя MIME-типа в любом регистре (MIME (Multipurpose Internet Mail Extension, Многоцелевые расширения почты Интернета) – спецификация для передачи по сети файлов различного типа: изображений, музыки, текстов, видео, архивов и др. Указание MIME-типа используется в HTML обычно при передаче данных форм и вставки на страницу различных объектов. Примеры см. в прил. 4).

Атрибуты hspace и vspace задают горизонтальные и вертикальные отступы от объекта до окружающего контента в пикселях.

Пример. Вставка flash-ролика на страницу.

```
<p>
<object type="application/x-shockwave-flash"
data="images/mouse.swf" width="400" height="300">
<param name="quality" value="high">
<param name="wmode" value="opaque">
</object>
</p>
```

Следует заметить, что в данном примере используется тег <param>, предназначенный для передачи значений параметров объектам веб-страницы. Количество одновременно используемых тегов <param> может быть больше одного и для каждого из них задается пара «имя/значение» через атрибуты name и value.

Тег <embed>

Элемент <embed>, как и тег <object>, используется для загрузки и отображения объектов (например, видеофайлов, флэш-роликов, некоторых звуковых файлов и т.д).

Спецификация HTML 4.0 рекомендует использовать тег <object> для загрузки внешних данных вместо тега <embed>. Однако некоторые браузеры не отображают таким образом нужную информацию, поэтому наилучшим вариантом будет поместить <embed> внутрь контейнера <object>.

Данный тег имеет атрибуты width, height, hspace, vspace и type, которые аналогичны одноименным атрибутам тега <object>. Но у тега <embed> есть свои специфические атрибуты.

Атрибут hidden представляет собой выключатель, который определяет, отображать объект в окне браузера или нет. Это особенно удобно для скрытия панели управления при воспроизведении фоновой музыки. Если этот атрибут указан, значение атрибутов width и height игнорируется.

Атрибут src указывает путь к файлу, который необходимо загрузить в окно браузера. Браузер анализирует расширение файла и решает по нему, какой плагин или внешняя программа требуется для отображения файла.

Пример с загрузкой flash-ролика на страницу в данном случае будет выглядеть следующим образом:

```
<embed src="images/mouse.swf" width="400" height="300"
type="application/x-shockwave-flash"
pluginspage="http://www.macromedia.com/go/getflashplayer"
>
</embed>
```

Если браузер не поддерживает работу с плагинами, то с помощью тега <noembed> можно вывести на экран соответствующее сообщение. Если может отобразить объект, указанный в теге <embed>, то содержимое тега <noembed> будет проигнорировано.

Таким образом, предыдущий пример с добавлением тега <noembed> примет следующий вид:

```
<embed src="mouse.swf" width="400" height="300"
  type="application/x-shockwave-flash"
  pluginspage=
  "http://www.macromedia.com/go/getflashplayer">
</embed>
<noembed>
  Браузер не поддерживает работу с плагинами
</noembed>
```

1.12. Создание элементов форм

Формы создают и используют для получения отклика пользователя на предоставленную информацию и для обеспечения диалога с пользователем.

Форма – это инструмент, с помощью которого HTML-документ может послать некоторую информацию в некоторую заранее определенную точку внешнего мира, где информация будет некоторым образом обработана.

Формы передают информацию программам-обработчикам в виде пар:

[имя переменной]=[значение переменной].

Имена переменных следует задавать латинскими буквами. Значения переменных воспринимаются обработчиками как строки, даже если они содержат только цифры.

Тег <form>

Задание формы осуществляется с помощью тега <form>. Форма определяется последовательностью тегов input, размещенных внутри пары <form> и </form>. Например:

```
<form>
  Введите фамилию <input type=«text» name=NameF>
  Введите имя <input type=«text» name=NameI>
</form>
```

HTML-документ может содержать в себе несколько форм, однако формы не должны находиться одна внутри другой.

Тег <form> может содержать следующие атрибуты.

accept-charset – устанавливает кодировку, в которой сервер может принимать и обрабатывать данные формы.

action – указывает обработчик, к которому обращаются данные формы при их отправке на сервер. В качестве обработчика может выступать CGI-программа или HTML-документ, который включает в себя серверные сценарии (например, Parser). После выполнения обработчиком действий по работе с данными формы он возвращает новый HTML-документ. Если атрибут action отсутствует, текущая страница перезагружается, возвращая все элементы формы к их значениям по умолчанию.

Пример:

```
<form action=”pro1.html”>
```

autocomplete – управляет автозаполнением полей форм. Значение может быть перекрыто атрибутом autocomplete у конкретных элементов формы. Автозаполнение производит браузер, который запоминает написанные при первом вводе значения, а затем подставляет их при повторном наборе в поля формы. При этом автозаполнение может быть отключено в настройках браузера и не может быть в таком случае изменено при помощи атрибута autocomplete.

method – атрибут определяет, каким образом (иначе говоря, с помощью какого метода протокола передачи гипертекстов) данные из формы будут переданы обработчику. Допустимые значения: POST и GET. Если значение атрибута не установлено, по умолчанию предполагается method=«GET». В данном случае пары «имя=значение» присоединяются в этом случае к адресу после вопросительного знака и разделяются между собой амперсандом (символ &). Удобство использования метода GET заключается в том, что адрес со всеми параметрами можно использовать неоднократно, сохранив его, например, в закладки браузера, а также менять значения параметров прямо в адресной строке. Метод POST посылает на сервер данные в запросе браузера. Это позволяет отправлять большее количество данных, чем доступно методу GET, поскольку у него установлено ограничение в 4 Кб. Большие объемы данных используются в форумах, почтовых службах, заполнении базы данных, при пересылке файлов и др.

enctype – атрибут определяет, каким образом данные из формы будут закодированы для передачи обработчику. Если значение атрибута не установлено, по умолчанию предполагается enctype=application/x-www-form-unrecorded. При использовании данной кодировки пробелы в данных заменяются на символы «+», а все другие непечатаемые символы – на комбинацию символа «%» и шестнадцатеричного кода, соответствующего данному символу в ASCII-таблице. Если используется поле для отправки файла (input type="file"), следует определить атрибут enctype как multipart/form-data.

name – определяет уникальное имя формы. Как правило, имя формы используется для доступа к ее элементам через скрипты.

Тег <input> – поле ввода

Тег <input> определяет область внутри формы, куда вводят данные. Каждый тег <input> должен включать атрибут name =[имя], определяющий имя переменной, через которую будет

передаваться обработчику. Имя должно задаваться только латинскими буквами.

Большинство тегов <input> должно включать атрибут value=«значение», определяющий значение, которое будет передано обработчику под этим именем. Для тегов <input type=text> и <input type=password>, этот атрибут не обязателен, поскольку значение соответствующей переменной может вводиться пользователем с клавиатуры.

Тег <input> формирует поле для ввода информации пользователем. Это может быть текстовое поле; поле-флажок, группа переключателей, изображение или кнопка. Вид поля ввода определяется значением атрибута type.

Атрибут type=”text”. Когда необходимо ввести небольшое количество текста (одну или несколько строк) используется тег input с атрибутом type, имеющим значение text. Это значение принято по умолчанию и указывать его необязательно.

Также указывается атрибут name для определения наименования переменной поля. Например:

```
<p>
  Введите название <input  type=«text» name=«nazv»>
</p>
```

Существуют дополнительные атрибуты:

– атрибут maxlength ограничивает число символов, вводимых пользователем в текущее поле. По умолчанию данное число не ограничено;

– атрибут size определяет размер видимой на экране области, занимаемой текущим полем. Размер определяется числом символов моноширинного шрифта. Если размер шрифта изменяется с помощью стилей, ширина также соответственно меняется. Если значение maxlength больше, чем size, браузер будет прокручивать данные в окне;

– атрибут value определяет начальное значение поля ввода.

Например, формирование двух полей для ввода символьных данных осуществляется следующим образом:

```
<form>
Ваша фамилия <input name=«namef» type=«text» size=«35»>
<br>Ваше имя <input name=«namen» type=«text» size=«20»>
</form>
```

Атрибут type=”password”. Разновидностью текстового поля можно считать поле для ввода пароля, которое создается при значении атрибута type= «password». При наборе текст в данном поле скрыт символами «*».

Атрибут type=”checkbox” позволяет создать поле-флажок, позволяющее получить ответ на вопрос типа «Да/Нет». Форма может содержать несколько полей-флажков для ввода данных. Когда форма использует тег <input> с атрибутом checkbox, в ней должны присутствовать и атрибуты name и value. Атрибут name указывает на наименование данного поля (флага) ввода. В атрибуте value будет содержаться значение поля. Пример использования:

```
Россия <input name="strana1" type=”checkbox”
value="Россия">
Страны СНГ <input name=”strana2” type=”checkbox”
value=”СНГ”>
```

В некоторых случаях необходимо инициализировать данный флаг, как уже отмеченный. В таких случаях тег <input> должен содержать атрибут checked. Пример использования:

```
Россия<input name=”strana” type=”checkbox”
value="Россия" checked>
```

Атрибут type=”radio” позволяет организовать выбор одного из нескольких возможных значений. Для создания формы ввода при выборе пользователем одного значения из нескольких возможных необходимо использовать тег <input> с атрибутом type=”radio”. Когда в форме применяется данный атрибут, в теге <input> должны быть указаны атрибуты name и value. Атрибут name указывает наименование соответствующего поля (кнопки). Атрибут value содержит значение поля. Пример использования:

```
Пол мужской
<input name=”pol” type=”radio” value="мужской">
Пол женский
<input name=”pol” type=”radio” value="женский">
```

Атрибут type=”image” позволяет создать в форме изображение (например, пользователь может щелкнуть мышью на изображении, чтобы завершить работу с формой). Когда пользователь щелкает мышью по изображению, браузер сохраняет координаты соответствующей точки экрана. Далее он «обрабатывает» введенную в форму информацию. Когда форма использует атрибут image, тег <input> должен содержать также атрибуты name и src. name указывает наименование поля ввода формы. Атрибут src содержит имя файла – источник изображения. Пример использования:

```
<br>Для выхода нажмите на кнопку<input type=”image” name=”point” src=”ris1.bmp”>
```

Атрибут type=”button” позволяет создать на форме простую кнопку. Надпись на кнопку добавляется с помощью атрибута value.

Атрибут type=”file” позволяет создать поле для ввода имени файла, который пересылается на сервер. Рядом с полем автоматически добавляется кнопка с именем «Обзор…», которая позволяет выбрать имя файла через диалог.

Атрибут type=”reset” используют для возможности начать ввод информации сначала и очистить от данных уже введенные поля. Если в форме используется атрибут reset, тег <input> может дополнительно содержать атрибут value. Данный атрибут определяет надпись на изображении кнопки, нажатие на которую будет очищать содержимое полей формы. Пример использования:

```
<input type=”reset” value="Очистить форму">
```

Атрибут type=”submit” обеспечивает процесс передачи данных из формы обработчику. Браузер выводит данный элемент как кнопку, по которой пользователь может щелкнуть, чтобы все имеющиеся в форме данные будут переданы обработчику, определенному в теге <form>. Когда в форме используется тег <input> с атрибутом submit, данный элемент может содержать два дополнительных атрибута: name и value. Атрибут value указывает наименование кнопки Submit. Пример использования:

```
<br><input type=”submit” value="Закончить ввод">
```

Надпись, нанесенную на кнопку, можно при необходимости передать обработчику путем введения в определение кнопки атрибута name=[имя]. Например:

```
<input type=”submit” name=”my_button” value=”Ввод “>
```

При нажатии на такую кнопку обработчик вместе со всеми остальными данными получит и переменную my_button со значением Ввод.

В форме может быть несколько кнопок типа submit с различными именами и/или значениями. Обработчик, таким образом, может действовать по-разному в зависимости от того, какую кнопку submit нажал пользователь.

Для того, чтобы создать простую кнопку следует в атрибуте type указать значение button.

Тег <select> – поле со списком

Данный тег используется для создания поля со списком. Для этого применяют тег <select> с атрибутом type=”select”. Для определения списка используют тег <option>. Тег <select> поддерживает три необязательных атрибута: multiple, name, size.

Атрибут multiple позволяет выбрать более чем одно наименование. Атрибут name определяет наименование объекта. Атрибут size определяет число видимых пользователю пунктов списка. Если в форме установлено значение атрибута size=”1”, то браузер выводит на экране список в виде выпадающего меню. В случае size > 1 браузер представляет на экране обычный список.

В форме может использоваться тег <option> только внутри тега <select>. Эти теги поддерживают два дополнительных атрибута: selected и value.

Атрибут selected используется для первоначального выбора значения элемента по умолчанию. Атрибут value указывает на значение, возвращаемое формой после выбора пользователем данного пункта. По умолчанию значение поля равно значению тега <option>. Пример использования:

```
<br>
ВУЗ<select name=”vuz”>
  <option value=”ВГУ”>ВГУ</option>
  <option value=”ВГТУ” selected>ВГТУ</option>
  <option value=”ВГАУ”>ВГАУ</option>
  <option value=”ВГТА”>ВГТА</option>
  <option value=”ВГПУ”>ВГПУ</option>
</select>
```

В примере формируется поле со списком, которое имеет подпись «ВУЗ», результат выбора сохраняется в переменной vuz. По умолчанию переменная имеет значение «ВГТУ».

У тега <option> есть следующие атрибуты.

value – значение пункта списка, которое будет отправлено на сервер или прочитано с помощью скриптов.

selected – заранее устанавливает определенный пункт списка выделенным.

disabled – блокирует для доступа соответствующий элемент списка.

label – в качестве значения указывается метки пункта списка, сокращённой по сравнению с текстом внутри <option>. Если атрибут label присутствует, то текст внутри тега <option> игнорируется и в списке выводится значение label.

Пример текста программы для формирования формы для ввода данных:

```
<h1> Форма для ввода данных</h1>
<form>
  Ваша фамилия <input name=”namef” type=”text”
          size=”35”>
  <br>Ваше имя <input name=”namen” type=”text”
          size=”20”>
  <br>Россия <input name=”strana” type=”checkbox”
          value="Россия" checked>
        Страны СНГ<input name=”strana” type=”checkbox”
          value="СНГ">
  <br> Пол мужской<input name=”pol” type=”radio”
          value="мужской">
        Пол женский<input name=”pol” type=”radio”
          value="женский">
  <br>
  ВУЗ
  <select name=”vuz”>
          <option value=”ВГУ”>ВГУ</option>
```

```
            <option value=”ВГТУ” selected>ВГТУ</option>
            <option value=”ВГАУ”>ВГАУ</option>
            <option value=”ВГТА”>ВГТА</option>
            <option value=”ВГПУ”>ВГПУ</option>
    </select>
    <br>Для выхода нажмите на кнопку
    <input type=”image” name=”point” src=”ris1.bmp”>
    <p>
            <input type=”reset” value=”Очистить форму”>
    </p>
    <br><input type=”submit” value="Закончить ввод
            или редактирование">
</form>
```

В результате выполнения данной программы формируется форма, представленная на рис. 11.

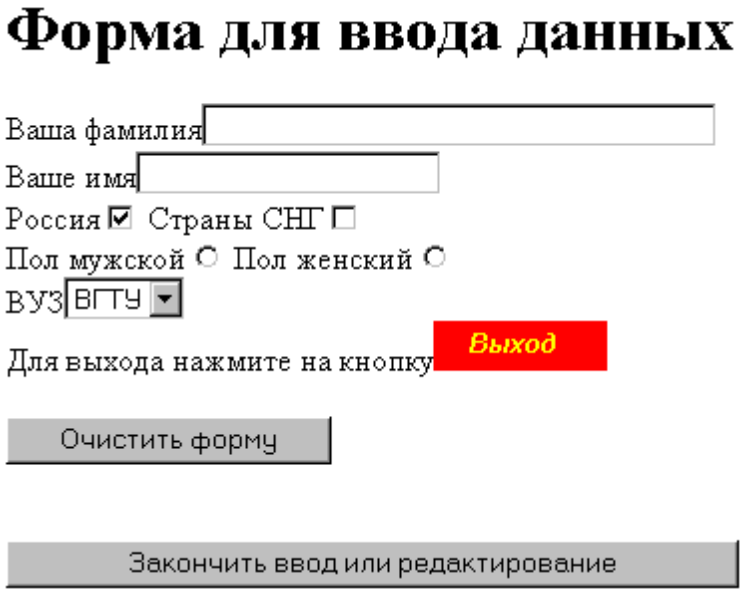

Рис. 11. Пример формы для ввода данных

Другие теги формы

Тег <textarea>. Представляет собой элемент формы для создания области, в которую можно вводить несколько строк текста. В отличие от тега <input> в текстовом поле допустимо делать переносы строк, они сохраняются при отправке данных на сервер.

Данный тег имеет следующие важные атрибуты.

cols – ширина поля в символах.

rows – высота поля в строках.

maxlength – максимальное число введенных символов.

wrap – параметры переноса строк. Значение soft указывает, что длинный текст, который самостоятельно не помещается в поле по ширине, будет автоматически перенесен на новую строку, однако передаваться на сервер будет как одна строка. Нажатие клавиши Enter устанавливает перенос текста, который сохраняется при отправке формы. Значение hard указывает, что слова в поле переносятся механически, чтобы они поместились в размер области, и при отправке на сервер места автоматического переноса сохраняются. При этом значении обязательно должен присутствовать атрибут cols. По умолчанию подразумевается значение soft.

Пример:

```
<p>
<textarea rows="10" cols="20" wrap="hard"> </textarea>
</p>
<p>
<input type="submit" value="Отправить">
</p>
```

Тег <button>. Тег <button> создает на веб-странице кнопки и по своему действию напоминает результат, получаемый с помощью тега <input> с атрибутом type со значением button. Но, в отличие от этого тега, <button> предлагает расширенные возможности по созданию кнопок. Например, на подобной кнопке можно размещать любые элементы HTML, в том числе

изображения. Используя стили можно определить вид кнопки путем изменения шрифта, цвета фона, размеров и других параметров. Важным атрибутом данного тега является **type**, определяющий тип кнопки: button (обычная кнопка), reset (очистка формы) или submit (отправка данных формы на сервер).

Пример:

```
<button>
<img src="images/button1.gif" alt="Кнопка1">
Кнопка с рисунком
</button>
<button type="reset">
Очистить форму
</button>
<button type="submit">
Отправить форму
</button>
```

Тег <fieldset>. Элемент <fieldset> предназначен для группирования элементов формы. Такая группировка облегчает работу с формами, содержащих большое число данных. Например, один блок может быть предназначен для ввода текстовой информации, а другой – для флажков.

Браузеры для повышения наглядности отображают результат использования тега <fieldset> в виде рамки. **Тег <legend>** применяется для создания заголовка данной рамки.

Например, приведенный ниже код будет иметь вид, показанный на рис. 12.

```
<fieldset>
  <legend>Укажите образование</legend>
    <input type="checkbox"> Среднее<br>
    <input type="checkbox"> Среднее специальное<br>
    <input type="checkbox"> Высшее
     <p><input type="submit"></p>
 </fieldset>
```

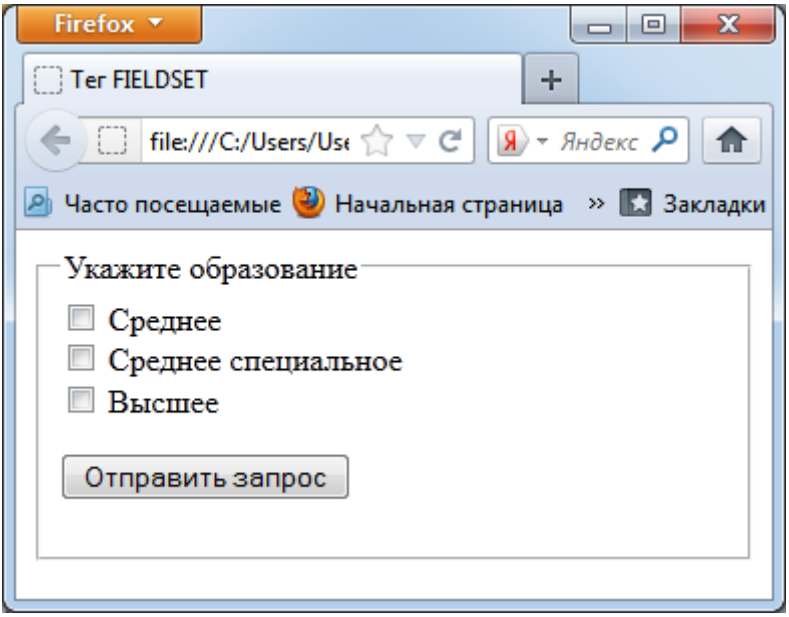

Рис. 12. Пример тега <fieldset>

Тег <label>. Элемент <label> устанавливает связь между определенной меткой, в качестве которой обычно выступает текст, и элементом формы (<input>, <select>, <textarea>). С точки зрения пользователя, тег <label> выступает в роли подписи элемента формы.

Существует два способа связывания объекта и метки. Первый заключается в использовании идентификатора id внутри элемента формы и указании его имени в качестве атрибута for тега <label>. При втором способе элемент формы помещается внутрь контейнера <label>.

Пример:

```
<label> Фамилия
<input type="text" name="fam">
</label><br>
<label> Имя
```

```
<input type="text" name="imya">
</label><br>
<label> Отчество
<input type="text" name="otch">
</label></br>
```

1.13. Универсальные атрибуты

Универсальные атрибуты применяются практически ко всем тегам, поэтому есть смысл рассмотреть их в отдельном разделе. Далее приведены самые основные универсальные атрибуты.

Атрибут accesskey – позволяет получить доступ к элементу с помощью сочетания клавиш с заданной в атрибуте буквой или цифрой. Браузеры при этом используют различные комбинации клавиш. Например, для accesskey="s" работают следующие сочетания:

Internet Explorer: Alt + S
Chrome: Alt + S
Opera: Shift + Esc + S
Safari: Alt + S
Firefox: Shift + Alt + S

Обычно элемент при использовании комбинации клавиш получает фокус, но конкретное действие зависит от применяемого тега. К примеру, для <a> произойдет переход по ссылке, для текстовых полей можно вводить текст, для флажков (<input type="checkbox">) будет поставлена галочка и т.д.

В качестве значения атрибута могут выступать цифры от 0 до 9 или латинские буквы от a до z.

Атрибут class – позволяет задать стилевой класс, который позволяет связать определенный тег со стилевым оформлением. В значении допускается указывать сразу несколько классов, разделяя их между собой пробелом. Имена классов могут содержать в себе латинские буквы (A–Z, a–z), цифры (0–9), символ дефиса (-) и подчеркивания (_). Использование русских букв в классах недопустимо. Смысл данного (и последующего)

атрибута раскроется более полно при изучении каскадных таблиц стилей CSS.

Атрибут id – задает стилевой идентификатор – уникальное имя элемента, которое используется для изменения его стиля и обращения к нему через скрипты. Идентификатор в коде документа должен быть в единственном экземпляре, иными словами, встречаться только один раз. Идентификатор должен обязательно начинаться с латинского символа и может содержать в себе латинские буквы (A–Z, a–z), цифры (0–9), символ дефиса (-) и подчеркивания (_). Использование русских букв в именах идентификатора недопустимо.

Атрибут dir – задает направление и отображение текста – слева направо или справа налево. Браузеры обычно самостоятельно различают направление текста, если он задан в кодировке Юникод, но с помощью атрибута dir можно указать, в каком направлении отображать текст. Для арабских и еврейских символов приоритетным является направление, заложенное в Юникод, поэтому на них атрибут dir действовать не будет.

Атрибут lang. Текст документа может быть набран как на одном языке, так и содержать вставки на других языках, которые могут различаться по своим правилам оформления текста. Чтобы указать язык, на котором написан текст внутри текущего элемента и применяется атрибут lang. Браузер использует его значение для правильного отображения некоторых символов. В качестве значения указывается код языка. Например, ru - русский язык, fr – французский, ja – японский и т.д.

Атрибут style – применяется для определения стиля элемента с помощью правил CSS. В качестве значений указываются стилевые правила: вначале следует имя стилевого свойства, затем через двоеточие его значение. Стилевые свойства разделяются между собой точкой с запятой. Более подробно данный атрибут будет рассмотрен в следующей главе.

Атрибут tabindex – устанавливает порядок получения фокуса при переходе между элементами с помощью клавиши Tab. Переход происходит от меньшего значения к большему, например от 1 к 2, затем к 3 и так далее. При этом строгая по-

следовательность не важна, допускается пропускать какие-то числа и начинать с любой цифры. Если значения tabindex у элементов совпадают, тогда учитывается их порядок появления в коде. Заблокированные элементы (у которых установлен атрибут disabled) не участвуют в переходе и фокус не получают.

Переход к элементам, у которых не задан атрибут tabindex или его значение равно 0, происходит после всех «нумерованных» элементов в том порядке, как они указаны в коде.

В качестве значения выступает любое целое положительное число.

Атрибут title – описывает содержимое элемента в виде всплывающей подсказки, которая появляется при наведении курсора на элемент. Вид такой подсказки зависит от браузера, настроек операционной системы и не может быть изменен с помощью HTML-кода или стилей. В качестве значения выступает любая текстовая строка, заключеннная в двойные или одинарные кавычки.

Пример всплывающей подсказки (рис. 13):

```
<p title="А вот и подсказка!">Пример всплывающей подсказки</p>
```

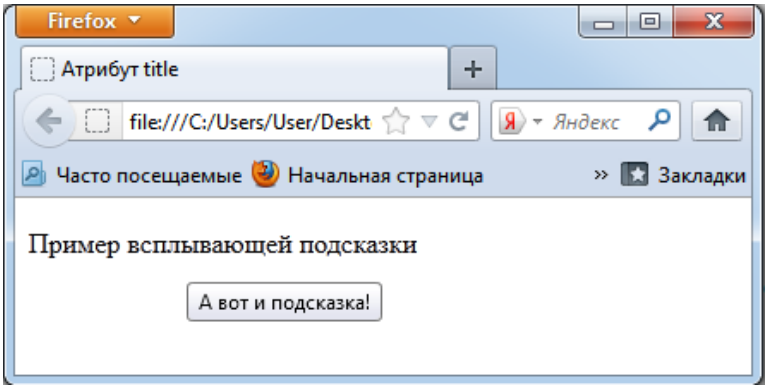

Рис. 13. Пример всплывающей подсказки, созданной с помощью атрибута title

1.14. Особенности XHTML

XHTML (англ. eXtensible Hypertext Markup Language – Расширяемый язык разметки гипертекста) – язык разметки веб-страниц, по возможностям сопоставимый с HTML, созданный на базе XML (eXtensible Markup Language). Версия XHTML 1.1 одобрен в качестве Рекомендации Консорциума Всемирной паутины (W3C) в 2001 году.

Стандарт XHTML построен не как отдельное описание языка, а как перечень различий между HTML 4.01 и XHTML.

Основные отличия XHTML от HTML

Все элементы (теги) должны быть закрыты. Теги, которые не имеют закрывающего тега (например, <img> или <br>) должны иметь на конце / (например, <br />).

Булевы атрибуты записываются в развёрнутой форме. Например, следует писать <option selected="selected"> или <td nowrap="nowrap">.

Имена тегов и атрибутов должны быть записаны строчными буквами (например, вместо <IMG ALT="" />);

Значения атрибутов пишутся только в кавычках – двойных или одинарных;

Специальные символы везде должны быть заменены их кодами. Например, < и & везде, даже в URL, должны замещаться < и & соответственно;

Кодировкой по умолчанию является UTF-8 (в отличие от HTML, где кодировкой по умолчанию является ISO 8859-1).

Версии XHTML

XHTML 1.0 Переходный (Transitional): предназначен для лёгкой миграции из HTML 3.2 и выше.

XHTML 1.0 Строгий (Strict): полностью отделяет содержание документа от оформления (которое теперь задаётся только через CSS), многие атрибуты (такие как, например, bgcolor и

align) более не поддерживаются, их поведение можно задавать только через таблицу стилей.

XHTML 1.0 Фрэймовый (Frameset): используется, если необходимо разделить окно браузера на несколько фрэймов.

XHTML 1.1 Модульный (Module-based): авторы могут импортировать дополнительные свойства в их разметку.

XHTML Основной (Basic): специальная облегчённая версия XHTML для устройств, которые не могут использовать полный набор элементов XHTML – в основном используется в миниатюрных устройствах, таких как мобильные телефоны.

Валидным (т. е. отвечающим всем правилам) XHTML-документом считается документ, удовлетворяющий технической спецификации. В идеале, все браузеры должны следовать веб-стандартам и, в соответствии с ними, валидные документы должны отображаться во всех браузерах на всех платформах. Валидация XHTML-документа рекомендована даже несмотря на то, что она не гарантирует кросс-браузерности. Документ может быть проверен на соответствие спецификации с помощью онлайновой Службы валидации разметки W3C. Валидация обнаружит и разъяснит ошибки в XHTML-разметке.

Валидный документ должен содержать определение типа документа (DTD – Document Type Definition). DTD должен быть расположен до всех других элементов документа. Вот наиболее распространённые типы DTD для XHTML:

XHTML 1.0 Strict

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

XHTML 1.0 Transitional

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

XHTML 1.0 Frameset

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

XHTML 1.1

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

Помимо определения типа документа следует добавить в корневой тег <html> обозначение пространства тегов xhtml.

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xml:lang="ru" lang="ru">
```

Последние значения являются определенным атрибутом XML, который объявляет информацию о языке и региональных параметрах элемента объекта.

Также следует внимательно проставлять тип содержимого документа MIME-тип документа.

В том случае, если MIME-тип – text/html (а это чаще всего и есть значение по умолчанию), все современные браузеры поддерживают XHTML. Он также совместим и со старыми браузерами, т.к. в основе XHTML лежит HTML. Например:

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

Однако если автор страницы задал MIME-тип как application / xhtml+xml, браузер Internet Explorer 6 не сможет обрабатывать страницу, поскольку у него нет XML-парсера (синтаксического анализатора текста).

2. ИСПОЛЬЗОВАНИЕ КАСКАДНЫХ ТАБЛИЦ СТИЛЕЙ CSS

2.1. Способы применения стилей

Каскадные таблицы стилей CSS (Cascading Style Sheets) позволяют разработать и применить стиль отображения страницы или сайта в целом.

CSS позволяют остаться в рамках декларативного характера разметки страницы и полностью контролировать форму представления элементов HTML-разметки.

Таблицы стилей также позволяют определить цвет и начертание текстового фрагмента, изменять эти параметры внутри текстового блока, выполнять выравнивание текстового блока относительно других блоков и компонентов страницы.

Наличие подобных возможностей позволяет говорить о CSS как о средстве разделения логической структуры документа и формы его представления. Логическая структура документа определяется элементами HTML-разметки, в то время как форма представления каждого из этих элементов задается CSS-описателем элемента.

Первая версия CSS была принята стандартом в декабре 1996 года. С того времени было разработано несколько новых версии, в т.ч. CSS 2.1, которая была принята как рекомендация W3C в июне 2011 года. В текущий момент разрабатывается версия CSS 3.0, некоторые возможности которой будут рассмотрены в третьей главе.

Различают четыре способа применения стилей:

– переопределение стиля в элементе разметки с помощью атрибута style;

– размещение описания стилей в заголовке документа в теге <style> (формирование внутренней таблицы стилей);

– размещение ссылки на внешнее описание через элемент Link (формирование внешней таблицы стилей);

– импорт описания стиля в документ.

Переопределение стиля с помощью атрибута style

Самый простой способ использования CSS – это ввести в код HTML атрибут Style с соответствующим значением. Его можно добавить практически ко всем тегам (кроме таких, как, например, <head> и <html>).

Например:

<h1 style=”color: red; font-size: 24px;”>

Но данное объявление стиля неудобно и загромождает код программы. Поэтому лучше создавать таблицы стилей, размещаемые в теге style документа или в отдельном файле.

Внутренние таблицы стилей

Внутренние таблицы стилей оформляются с помощью тега <style>, который помечает начало и конец таблицы стилей (в конце ставится закрывающий тег </style>). Обычно таблица стилей размещается в заголовке документа, обрамленном тегом head.

В таблице стилей может быть один стиль или множество разных стилей. Каждый стиль содержит несколько элементов. Структуру стиля можно представить следующим образом:

Селектор { свойство1: значение1; свойство2: значение2; …; свойствоN: значениеN}

Назначение элементов стиля:

– селектор – сообщает браузеру, к какому элементу или элементам web-страницы применяется стиль;

– блок объявлений стиля – содержит все форматирующие команды, которые можно применить к селектору. Блок обрамляется фигурными скобками;

– объявление свойства – в блок объявлений можно добавить одно или несколько определений или форматирующих команд. Каждое объявление имеет две части – свойство и значе-

ние свойства, разделяемые воеточиием, и заканчивается точкой с запятой;

– свойство – представляет собой слово или несколько записанных через дефис слов, определяющих конкретный стиль или стилевой эффект. Обычно эти слова доступны для понимания (font-size, background-color, margin-top);

– значение – различные CSS-свойства требуют определенных типов значений. Это может быть цвет (red или #FF000), длина (18px или 2in), URL(images/background/gif) или ключевое слово (top, center, bottom).

Описание стиля может размещаться не в одной, а в нескольких строках для облегчения восприятия, например:

```
p {
  color: red;
  font-size: 1em;
}
```

Внутренняя таблица стилей может выглядеть следующим образом:

```
<style type=”text/css”>
h1 {
  color: #FF7643;
font-size: 20pt;
}
p {
  color: red;
  font-size: 1em;
}
</style>
```

Внутренние таблицы стилей – довольно удачный способ добавления стилей в код web-страницы, но имеет определенные недостатки. Во-первых, объемные таблицы усложняют код web-страницы, что приводит к увеличению времени ее загрузки. Во-

вторых, при наличии большого количества web-страниц одинаковой структуры и оформления, требуется добавлять такую таблицы на каждую web-страницу, что не очень удобно. Поэтому существует возможность описания таблиц стилей в отдельном файле.

Внешние таблицы стилей

Внешняя таблица стилей – не что иное, как текстовый файл, содержащий набор стилей CSS. Он не должен содержать HTML-кода, поэтому тег style в файле не используется. Такой файл всегда должен иметь расширение .css (и относительно наглядное название).

После создания таблицы стилей ее необходимо подключить к веб-странице, которую нужно отформатировать. Это можно сделать двумя способами:

1. Связывание таблицы стилей с HTML-кодом.

Для это надо использовать тег link.

Код HTML:

```
<link rel=”stylesheet” type=”text/css” href=”css/global.css”>
```

Код XHTML:

```
<link rel=”stylesheet” type=”text/css” href=”css/global.css” />
```

Атрибут rel указывает тип ссылки – в данном случае это ссылка на таблицу стилей; атрибут type сообщает браузеру, данные какого типа ему следует ожидать (в данном случае – текстовый файл, содержащий CSS-код); href – указывает местонахождение внешнего CSS-файла.

2. Импорт таблиц стилей в HTML-документ.

CSS имеет встроенный способ привязки внешних таблиц стилей к коду HTML – правило @import. Его нужно добавить в содержимое тега style. Например:

```
<style type=”text/css”>
```

```
  @import url(css/global.css);
</style>
```

В скобках после url следует указать ссылку на файл CSS. Кавычки использовать не обязательно.

Можно использовать несколько правил @import внутри одного тега style. После правила @import можно добавлять внутренние описания стилей.

2.2. Управление селекторами стилей

Селекторы типов

Селекторы типов представляют весьма эффективное средство проектирования дизайна веб-страниц, поскольку определяют стиль всех экземпляров конкретного HTML-элемента.

В качестве селекторов можно использовать название форматируемых элементов, например h1, h2, p, table и т.д.

Например, такой тег установить гарнитуру шрифта и его цвет для всех тегов h1:

```
h1 {
  font-family: “Arial”;
  color: #000000;
}
```

Но данный метод имеет свои недостатки – невозможно элементам одного типа присвоить разные стили. Поэтому в многих случаях следует использовать более точное указание с помощью селекторов классов.

Селекторы классов

Селекторы классов позволяют указать конкретный элемент веб-страницы, независимо от тегов. Названия классов должны удовлетворять следующим правилам:

– все названия селекторов классов должны начинаться с точки;

– при именовании стилевых классов разрешается использовать только буквы алфавита, цифры, дефисы, знаки подчеркивания;

– название после точки всегда должно начинаться с символа – буквы алфавита;

– имена стилевых классов чувствительны к регистру.

Пример объявления класса:

```
.special {
  color: #F0F122;
  font-family: Verdana;
}
```

Чтобы применить этот стиль к конкретному абзацу, следует добавить к тегу p данного абзаца атрибут class с указанием имени класса.

```
<p class=”special”>
```

Название класса в значении атрибута пишется без точки.

ID-селекторы

В качестве альтернативного варианта точного управления селекторами можно воспользоваться идентификаторами элементов страницы – создавать ID-селекторы.

У практически каждого тега может присутствовать атрибут id с некоторым уникальным значением, которое позволяет однозначно идентифицировать данный тег. Например:

```
<h1 id=”header”>
```

Данный тег можно связать с ID-селектором. ID-селектор состоит из значения атрибута id с добавленным в начало символом #.

```
#header {
  font-size: 36px;
  text-align: center;
}
```

Чтобы точно выбрать, что именно использовать – ID-селекторы или селекторы классов, можно воспользоваться следующими правилами:

– для стиля используемого неоднократно, применяют классы;

– для определения однократно используемого стиля лучше применять ID-селекторы;

– при конфликте селекторов классов и ID-селекторов более высокий приоритет имеют последние.

Селекторы потомков, групповые селекторы и псевдоклассы

Использование селекторов потомков позволяет отформатировать теги, находящиеся внутри тегов более высокого уровня. Например, если необходимо изменить цвет текста выделенного жирным шрифтом внутри абзаца, можно создать стиль следующего вида:

```
p strong {
  color: red;
}
```

В данном случае, красным цветом будет выделен только тот текст, помеченный тегом , который находится внутри тега <p>.

В данной конструкции тег более высокого уровня пишется слева, более низкого – справа. Еще один пример:

```
li a {
  font-family: Arial;
}
```

В данном случае стиль будет применен к гиперссылкам, расположенным в элементах списка.

Допускается несколько уровней иерархии в селекторе потомков. Также можно указывать не только имена тегов элементов, но идентификаторы или имена классов. Например:

```
p.intro a {
  color: yellow;
}
```

Данный стиль будет применен к гиперссылкам, находящимся внутри абзацев p, относящихся к классу .intro. Между p и .intro пробел не ставится, иначе селектор будет интерпретироваться по-другому.

Можно создавать групповые селекторы применяя один и тот же стиль к разным типам элементов. Например:

```
h1, h2, h3, h4, h5, h6 {
  color: #F1CD33;
}
```

Друг от друга селекторы отделяются запятыми. Также можно использовать разные виды селекторов:

```
h1, p, .copyright, #header {
  color: #F1CD33;
}
```

Существует понятие универсального селектора, который можно применить ко всем элементам страницы. В данном селекторе в качестве названия используется символ *.

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

```
}
```

Такой селектор обнулит значения отступов и полей у всех элементов страницы.

Для более гибкого управления содержимым тегов страницы используются так называемые псевдоклассы элементов.

Четыре псевдокласса предназначены для управления отображение гиперссылки:

a:link – обозначает обычную непосещенную ссылку;

a:visited – обозначает посещенную ссылку;

a:hover – обозначает ссылку, на которую наведен указатель мыши;

a:active – обозначает ссылку в момент нажатия на нее указателем мыши.

Например:

```
a:hover {
  color: yellow;
}
```

Замечание: если разные псевдоклассы для работы со ссылками используются одновременно, то они должны быть описаны в таблице стиля в определенном порядке: **link, visited, hover, active**. Иначе возможны сбои в работе данных стилей. Также следует заметить, что псевдокласс hover может использоваться для стилизации других элементов веб-страницы.

Существует псевдокласс **:focus**, который используется для управления отображением элементов, получивших фокус ввода. Обычно он используется вместе с полями для ввода информации пользователем (тег input). Например:

```
input:focus {
  background-color: #FFFFAA;
}
```

С тегом абзаца P могут быть использованы также псевдоклассы **:first-letter** и **:first-line**, которые могут быть использованы для форматирования первой буквы и первой строки абзаца соответственно.

Также можно использовать селекторы атрибутов, применяемы к тегам, у которых либо есть определенные атрибуты или атрибуты имеют определенные значения. Например:

```
p[align] {
  color: red;
}

a[href=”http://vorstu.ru”] {
  color: red;
  font-weight: bold;
}
```

Каскадность стилей

Каскадность – набор правил, определяющих, какие именно стилевые свойства необходимы элементам веб-страницы, то есть задающих последовательность применения многократно определенных стилей. Каскадность определяет, каким образом браузер должен обработать сложную структуру, относящуюся к одному и тому же тегу, если возникает конфликт свойств.

Если стили оказываются вложенным друг в друга, то стили более низкого уровня унаследуют свойства от стилей более высоких уровней. Например:

```
body {font-family: Verdana;}
p {color: #999944;}
strong {font-size: 24px;}
```

Если тег strong будет указан внутри тега P, то он унаследует цвет текста от стиля тега P и гарнитуру шрифта от стиля тега body.

Если между стилями возникает конфликт свойств, то преимущество будет у стиля тега более близкого элемента-предка. Например:

```
body {font-family: Verdana;}
p {color: #999944; font-family: Arial}
strong {font-size: 24px;}
```

Если тег strong будет указан внутри тега p, то он унаследует гарнитуру шрифта от стиля тега p, а не от стиля тега body. Однако, если у тега strong указать значение гарнитуры шрифта, то именно оно будет применено к его содержимому. Таким образом, непосредственно примененный стиль имеет более высокий приоритет, по сравнению со стилем предком.

В зависимости от типа селекторов стили имеют разные приоритеты.

По возрастанию приоритета стили можно выстроить следующим образом:

селектор тегов;
селектор классов;
id-селектор;
встроенный стиль.

Если конфликтуют стили, имеющие одинаковый приоритет, то побеждает стиль, описанный последним. Например, если сначала в раздел стилей добавлена внешняя таблица, а потом описаны внутренние стили, то последние будут иметь преимущество.

Можно разрешить конфликт стилей, путем указания свойства **!important**. Например, в случае пересечения стилей для тега документа приоритет будет иметь первый вариант:

```
body {background-color: red !important;}
body {background-color: black}
```

2.3. Стилизация текста и списков

Большинство сайтов в Интернете построены на текстовом содержимом. С помощью CSS можно придать текстовому содержанию привлекательный дизайн.

CSS предлагает обширный набор команд форматирования, которые позволяют назначать шрифты, цвет, размеры, межстрочный интервал и другие свойства и атрибуты, которые оказывают влияние на визуальное восприятие как отдельных элементов веб-страницы (заголовков, маркированных списков, обычных абзацев текста), так и всей веб-страницы.

Существуют следующие свойства CSS, предназначенные для форматирования текста.

font-family

Используется для выбора типа шрифта. Например:

```
p {
  font-family: Arial;
}
```

В данном случае ко всем абзацам будет применен шрифт Arial.

Рекомендуется выбирать шрифты, которые делают текст привлекательным для глаза (если это заголовок) и читабельным (если это основной текст). Существуют два основных вида шрифтов – с засечками и без засечек (гротесковые). Первые используются для больших текстовых блоков, поскольку засечки – небольшие черточки на концах основных штрихов литер – смягчают переходы между символами, делая текст более читабельным. Примеры – Times, Times New Roman, Georgia. Гротесковые шрифты часто используются для прорисовки заголовков благодаря их четким и простым начертаниям. Примеры: Arial, Helvetica, Verdana.

К сожалению, нельзя пользоваться какими угодно шрифтами, так как текст web-страницы не будет отображен адекватно

в браузере пользователя, если у него не установлен аналогичный шрифт (будет использован шрифт, установленный в браузере по умолчанию).

Одно из решений данной проблемы – определение в стиле необходимого шрифта, а также нескольких резервных. Например:

```
p {
  font-family: Arial, Helvetica, Verdana;
}
```

В данном случае браузер будет использовать шрифт Arial, если он установлен, в противном случае будет использоваться шрифт Helvetica и т.д.

Примеры часто используемых сочетаний шрифтов:

Arial, Helvetica, sans-serif;

“Times New Roman”, Times, serif;

“Courier New”, Courier, monospace;

Georgia, “Times New Roman”, Times, serif;

Verdana, Arial, Helvetica, sans-serif;

Geneva, Arial, Helvetica, sans-serif;

Как видно из примеров, если название состоит из двух или более слов, разделенных пробелами, название шрифта пишется в кавычках.

color

Данное свойство задает цвет шрифта. Наиболее распространенный вариант – задание цвета с помощью шестнадцатиричной константы, задающей уровень интенсивности RGB. Например:

```
h1 {
  color: #3E8988;
}
```

Можно также использовать метод кодирования RGB (Red, Blue, Green). Значение цвета кодируется тремя числами, представленными в диапазоне 0-255 или процентыми соотношениями 0 – 100%, показывающими интенсивности цветов.

Например, для белого цвета следует установить

color: rgb(100%, 100%, 100%);
или
color: rgb(255, 255, 255);

Как вариант можно использовать названия цветов. Например:

color: red;
или
color: yellow;

font-size

Данное свойство отвечает за установку размера шрифта. Существует ряд единиц измерения, в которых может выставляться данная величина:

– px – пиксели – наиболее простая и понятная единица измерения (но обладает одним недостатком - браузер Internet Explorer 6 не увеличивает размеры элементов, заданные в пикселях).

– em – стандартная единица измерения, пришедшая из типографской системы. Она равна ширине буквы M базового шрифта браузера (обычно его размер 16 пикселей).

– % – проценты от базового шрифта. Могут быть как больше, так и меньше 100%. Например, 50% - это половина размера базового шрифта, а 200% - двойной размер базового шрифта. (Но при наследовании стилей эти свойства надо использовать осторожно – так как при наложении свойств размеров, выраженных в процентах, могут получиться непредсказуемые результаты).

– ключевые слова – CSS предлагает семь ключевых слов, определяющих размер шрифта относительно базового: xx-small, x-small, small, medium, large, x-large, xx-large. Среднее значение medium – базовый размер шрифта браузера. Остальные значения уменьшают или увеличивают размер шрифта с коэффициентом 1,2.

– pt – пункты – еще один типографский вариант размера шрифта. Он часто используется в текстовых редакторах (например, в MS Word).

Примеры:

font-size: 12px;
font-size: 1.5em;
font-size: large;
font-size: 14pt;

Как видно из примеров, единица измерения указывается после величины шрифта без пробела, а в дробных значениях в качестве разделителя используется точка.

font-style

Данное свойство управляет стилем шрифта. Имеет три значения:

italic – делает шрифт курсивным;

normal – устанавливает обычный шрифт;

oblique – наклонный шрифт (в отличие от курсива, имитирующий рукописный наклонный шрифт, наклонный же образуется путем наклона обычных знаков вправо).

font-weight

Свойство устанавливает жирность шрифта. В качестве значения могут использоваться числа в диапазоне 100, 200…900. Также можно использовать два символьных значения:

bold – устанавливается жирный шрифт;

normal – устанавливается обычный шрифт.

text-transform

Свойство управляет регистром отображения символов. Имеет следующие значения:

uppercase – преобразовывает символы текста в верхний регистр;

lowercase – преобразовывает символы текста в нижний регистр;

capitalize – превращает первые буквы слов в прописные;

none – отменяет изменения регистра текста (удобно при управлении наследуемыми свойствами).

font-variant

Свойство, позволяющее преобразовывать текст в малые прописные буквы. Имеет значения:

small-caps – преобразует буквы текста в малые прописные;

normal – преобразует буквы текст в обычный вид.

text-decoration

Свойство используется для улучшения внешнего вида текста путем добавления дополнительных элементов. Имеет значения:

underline – подчеркнутый снизу текст;

overline – подчеркнутый сверху текст;

line-through – перечеркнутый текст;

blink – мерцающий («мигающий») текст;

none – отменяет декорирование текста (можно, например, удалить подчеркивание гиперссылки).

В данном свойстве можно комбинировать значения, например:

text-decoration: underline overline;

letter-spacing

Задает межсимвольный интервал. Выставляется в почти любых единицах измерения, применимых к тексту: пикселях, em, процентах. Для разреженного интервала следует использовать положительные значения, для сжатия – отрицательные, например:

letter-spacing: -1px;
letter-spacing: 10px;

word-spacing

Задает межсловный интервал. Принцип применения аналогичен предыдущему тегу.

line-height

Задает межстрочный интервал – промежуток между базовыми линиями двух соседних строк текстов. Стандартный интервал равен 120% - следовательно, меньшее значение сжимает данный интервал, а большее растягивает.

Можно использовать в качестве единиц измерения пиксели, проценты или em (последние два более желательны, так как изменяются пропорционально свойству font-size).

Чтобы определить размер межстрочного интервала браузер вычитает высоту шрифта из высоты строки. В результате получается размер межстрочного интервала между двумя соседними строками текста абзаца. Допустим, размер шрифта составляет 12 пикселей. Межстрочный интервал, установленный в размере 150%, в итоге равняется 18 пикселям. В итоге, браузер добавляет пустой промежуток размером 6 пикселям между двумя строками текста.

Межстрочный интервал можно также указать в виде числового значения. Например:

line-height: 1.5;

После этого значения не нужно указывать единицу измерения. Чтобы определить межстрочный интервал или высоту строки, браузер умножит указанное число на размер шрифта.

text-align

Данное свойство предназначено для выравнивания текста в блоке. Имеет следующие значения:

left – выравнивание по левому краю;

right – выравнивание по правому краю;

justify – выравнивание по ширине;

center – выравнивание по центру.

text-indent

Данное свойство предназначено для указания отступа красной строки абзаца. Для указания величины отступа лучше всего использовать величины em или пиксели. Например:

```
p {
  text-indent: 5em;
}
```

Управлять отображение первой строки абзаца также можно с помощью псевдоклассов first-line и first-letter, которые описаны в предыдущей лекции.

font

Универсальное свойство, которое позволяет одновременно задать несколько характеристик шрифта и текста. Оно может объединять сразу следующие свойства: font-style, font-variant, font-weight, font-size, line-height и font-family. Например:

```
p {
  font: italic small-caps bold 12px/14px Arial;
}
```

Следует обратить внимание на три вещи:

– если указываются размер шрифта и межстрочный интервал, то они должны быть указаны через косую черту в виде font-size/line-height;

– обязательными являются свойства font-size и font-family;

– все свойства могут быть указаны в произвольном порядке (кроме пары font-size/line-height).

Стилизация списков

Теги <ul> и <ol> создают маркированные и нумерованные списки соответственно. Списки состоят из последовательности элементов помеченных тегом <li>. Для управления их отображением можно использовать ряд свойств.

list-style-type

Используется для задания типа маркеров (тег <ul>) или типа нумерации (тег <ol>).

Для задания маркеров можно использовать следующие значения свойства (примеры маркеров приведены на рис. 14).

square – маркер в виде квадрата, залитого цветом;

disc – марке в виде круга, залитого цветом;

circle – маркер в виде полой окружности.

Для задания типа нумерации можно использовать следующие значения свойства:

decimal – список вида 1, 2, 3 и т.д.;

decimal-leading-zero – список вида 01, 02, 03 и т.д. (некоторые браузеры воспринимают это свойство как decimal);

upper-alpha – список вида A, B, C и т.д.;

lower-alpha – список вида a, b, c и т.д.;

upper-roman – список вида I, II, III и т.д.;

lower-roman – список вида i, ii, iii и т.д.

Можно совсем убрать маркеры, указав в свойстве значение none.

Например:

```
ul {
  list-style-type: none;
}
```

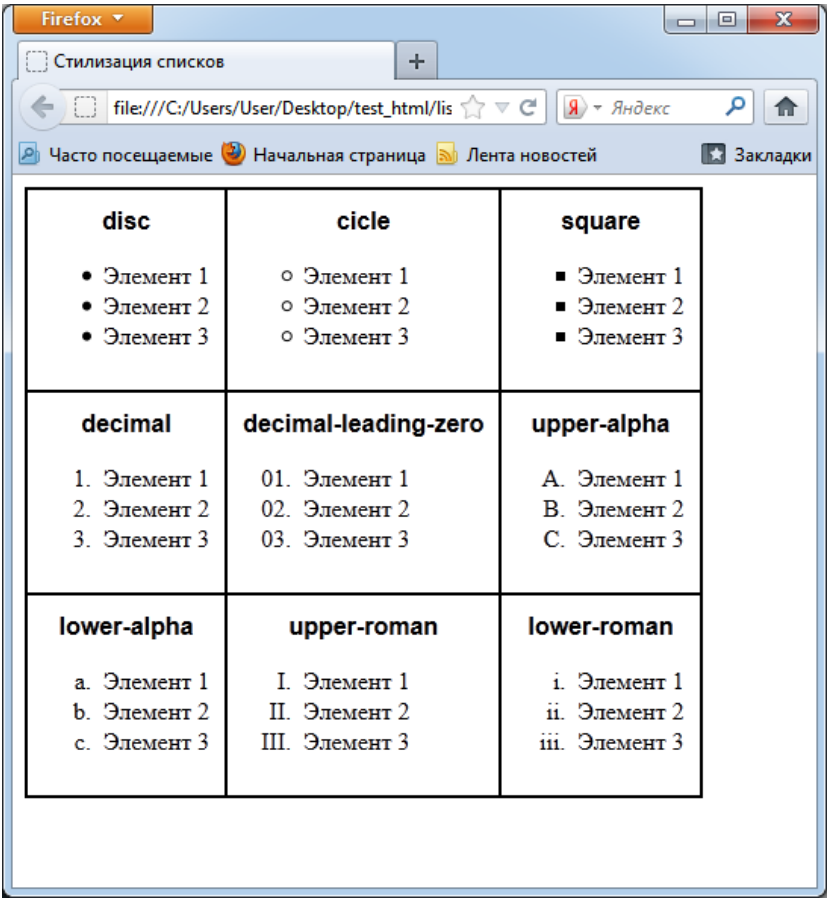

Рис. 14. Примеры различных типов маркеров списков

Данное свойство можно применять не только к тегам <ul> и <ol>, но и к тегам <li>, смешивая в одном списке разные маркеры. Например, можно создать такой класс стиля:

```
.circle {
  list-style-type: circle;
}
```

и применить его следующим образом:

```
<ul>
  <li>Item 1 </li>
  <li class=”circle”>Item 2 </li>
  <li>Item 3 </li>
  <li class=”circle”>Item 4 </li>
</ul>
```

В данном списке элементы 2 и 4 будут иметь маркер в виде полой окружности, а остальные два – в виде залитого круга (он устанавливается по умолчанию).

list-style-position

Определяет, каким образом будут размещаться маркеры относительно текста. Имеет два значения:

outside – маркеры размещаются обособленно от элементов списка слева (стандартный вариант);

inside – маркеры размещаются внутри текстовых блоков элемента списка слева.

list-style-image

Позволяет указать путь к графическому файлу, который используется в качестве маркера. Синтаксис свойства следующий:

list-style-image: url(images/bullet.gif);

Термин url и круглые скобки обязательны. Часть, заключенная в скобки, указывает путь к графическому файлу, который будет использован в качестве маркера.

2.4. Управление полями, отступами и границами

Понятие блочной модели

Браузер обрабатывает все теги как небольшие блоки. Для него любой тег является контейнером с содержимым: текстом,

изображением и т.д. У каждого блока может быть набор следующих свойств:

padding – *отступ, «заполнение»* - промежуток между содержимым и его границей;

border – *граница, рамка* – линия, черта или контур с любой стороны элемента. Граница может быть со всех четырех сторон или с одной стороны в любой их комбинации;

background-color – *цвет фона* – заполняет пространство внутри границы, включая область отступа padding;

margin – *поле* – пустое пространство, отделяющее один тег от другого. Полем может являться промежуток, отделяющий один абзац от другого.

Блочная структура в наглядном виде приведена на рис. 15.

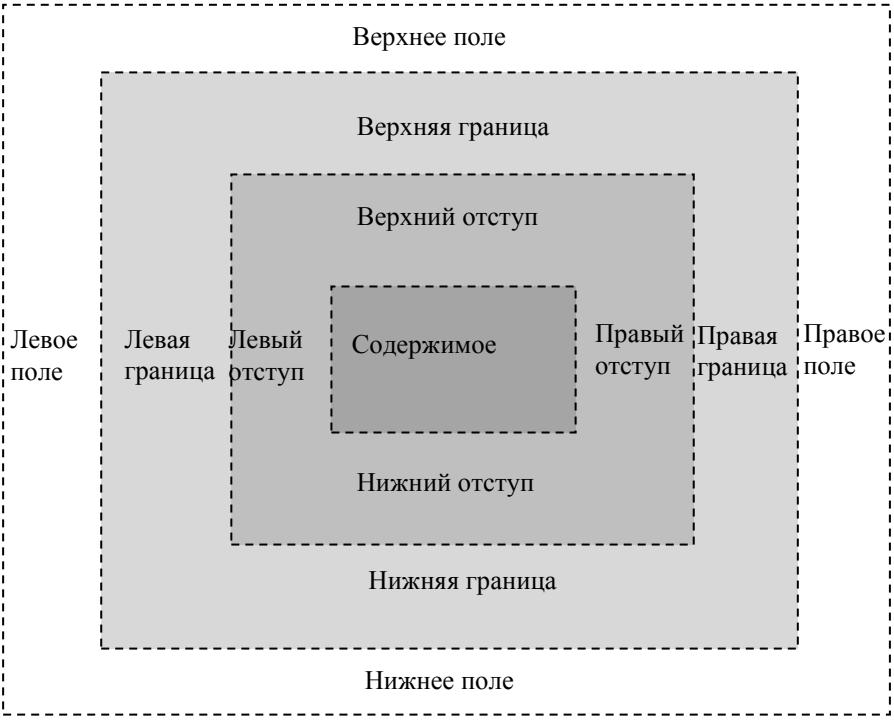

Рис. 15. Блочная структура элемента

Управление размерами полей и отступов

Для управления размерами полей используются четыре свойства:

margin-top – поле сверху;
margin-bottom – поле снизу;
margin-left – поле слева;
margin-right – поле справа.

Для управления размерами отступов используются аналогичные свойства:

padding-top – отступ сверху;
padding-bottom – отступ снизу;
padding-left – отступ слева;
padding-right – отступ справа.

В качестве единицы измерения можно использовать пиксели, em, проценты. Например:

```
margin-left: 10px;
padding-bottom: 2em;
margin-top: 10%;
```

Величина, указанная в процентах, измеряется относительно ширины элемента-контейнера. Например, если ширина элемента-контейнера 500 пикселей, то отступ слева, равный 10%, будет шириной 50 пикселей. Если ширина контейнера изменится, размер отступа тоже будет пересчитан.

Можно использовать сокращенное обозначение групп свойств margin и padding. Например:

```
margin: 5px 0 10px 20px;
padding: 10px 10px 5px 0;
```

Порядок значений важен – они должны идти в следующей последовательности:

top
right
bottom

left

Если значение свойства равно 0, то единицу измерения можно не указывать.

Если есть необходимость использовать одинаковое значение для всех отступов, то можно указать единственное значение:

padding: 10px;

Замечание: при использовании полей может произойти конфликт между соседними элементами. Например, если первый абзац имеет поле снизу в 20 пикселей, а второй – поле сверху 30 пикселей, то большинство браузеров не будут складывать эти значения, а выберут ***большее*** из них. Таким образом, получится промежуток между абзацами, равный не 50, а 30 пикселям.

Добавление границ

Граница представляет собой линию, которая очерчивает стилизуемый элемент. Она располагается между отступом и полем элемента. Можно добавить границу ко всем сторонам блока, а можно – только к некоторым из них.

У любой границы есть три различных свойства:

color – цвет;

width – ширина;

style – стиль.

Цвет выставляется стандартными способами, описанными ранее.

Ширина – толщина линии границы. Можно использовать любые единицы измерения CSS кроме процентов (лучше всего – пиксели) или ключевые слова thin (тонкая линия), medium (средняя) или thick (толстая).

Свойство style отвечает за стиль прорисовки границы и имеет следующие значения: solid, dotted, dashed, double, groove, ridge, inset, outset, none и hidden. Например, solid рисует

сплошную линию, dashed – пунктирную, dotted – линию из точек, none –убирает прорисовку линии и т.д.

С помощью свойства border можно указать параметры сразу всех границ. Например:

```
border: 4px solid #FF00AC;
```

Здесь указаны сразу все три свойства границы (толщина, стиль и цвет). Порядок указания не имеет значения.

Можно добавить только одну линию, например снизу:

```
border-bottom: 2px dashed red;
```

Можно добавить две линии справа и слева:

```
border-left: 2px solid red;
border-right: 2px sold green;
```

Можно указать свойство для всех границ, а потом переопределить для одной из границ:

```
border: 2px solid red;
border-bottom: 2px dashed black;
```

Подобным образом можно убрать одну из границ:

```
border: 2px inset #BBAA09;
border-top: none;
```

Можно более тонко настраивать разные свойства границ. Например:

```
border-width: 4px;
border-left-color: black;
```

Можно также переопределять конкретное свойство конкретной границы:

```
border: 4px solid red;
border-bottom-style: dashed;
```

Во всех случаях переопределения порядок имеет значение. Сначала идет общее свойство для всех границ, а потом – свойство, которое нужно переопределить.

2.5. Стилизация фона элементов

Цвет фона

Цвет фона элемента устанавливается с помощью свойства **background-color**. В качестве значения указывается цветовая константа в одном из ранее указанных форматов. Данный свойтсво может быть применено ко всему документу (как стиль тега body), так и к отдельному блоку (как стиль тегов h1, p, div и т.д.). Например:

```
body {
        background-color: aqua;
}
h1 {
        background-color: #AADEFB;
}
```

Работа с фоновыми изображениями

Для работы с фоновыми изображениями используется свойство **background-image**. Данное свойство добавляет рисунок в качестве фона элемента. Чтобы поместить его на заднем плане всей web-страницы, можно создать стиль для тега body:

```
body {
        background-image: url(images/bg.jpg);
}
```

В качестве значения указывается слово url, за которым следует путь к файлу рисунка, заключенный в круглые скобки. Можно использовать абсолютный URL, например:

```
url(http://www.mysite.ru/images/bg.jpg)
```

Можно использовать относительный путь от документа или корневого каталога сайта:

```
url(../images/bg.jpg)
url(/images/bg.jpg)
```

Если файл с таблицами стилей лежит в том же каталоге, что и документ (или таблица стилей находится в самом документе), то лучше использовать второй вариант. Если файл страницы лежит в корневом каталоге, файл со стилями – в подкаталоге styles, а файл с изображением – в подкаталоге images, то лучше воспользоваться первым вариантом, т.е.

```
body {
        background-image: url(../images/bg.jpg);
}
```

Этот путь будет интерпретироваться следующим образом:

../ – переход на уровень выше (к каталогу, содержащему папку style);

images/ – переход из корневого каталога к папке images;

bg.jpg – название файла изображения.

Кавычки в названии могут использоваться, но они не обязательны.

С помощью свойства **background-repeat** можно управлять повторением изображения. Данное свойство имеет четыре значения:

– repeat – параметр по умолчанию, обеспечивает повторение изображения по всей площади страницы;

– no-repeat – отображает фоновый рисунок один раз, без повторения и прекрытия;

– repeat-x – повторяет фоновое изображение горизонтально вдоль оси X, т.е. по всей ширине web-страницы (можно использовать для добавления баннера или заголовка);

– repeat-y – повторяет фоновое изображение вертикально вдоль оси Y, т.е. по всей длине web-страницы (можно использовать при добавлении вертикальных панелей на страницу).

Позиционирование фоновых изображений можно осуществлять с помощью свойства **background-position**, в качестве значения которого можно указать пару ключевых слов. Первое слово отвечает за горизонтальное позиционирование, второе – за вертикальное.

Горизонтальное позиционирование может быть трех видов:

left

center

right

Вертикальное позиционирование может быть трех видов:

top

center

bottom

Например, чтобы переместить фоновое изображение в правый верхний угол рисунка, следует указать свойство:

background-position: right top;

Для помещения изображения по центру страницы:

background-position: center center;

Следует заметить, что в Firefox не всегда правильно отображается фоновое изображение со значением вертикальной позиции bottom. Если высота содержимого страницы меньше высоты окна браузера, то нижний край последнего элемента считается нижним краем страницы (и изображение выравнивается по нему). Данная ситуация решается добавлением к странице стиля

```
html {
        height: 100%;
}
```

Можно использовать в качестве значений данного свойства точные значения: одно будет определять расстояние между левой стороной изображения и левым краем элемента-контейнера, а второе – расстояние между верхней стороной изображения и верхним краем элемента контейнера. например:

```
background-position: 10px 20px;
```

Можно также указывать отрицательные значения отступов, чтобы «подрезать» часть изображения относительно краев элемента.

Можно также управлять фиксацией изображения на странице, т.е. при прокрутке страницы перемещать ее содержимое, при этом не меняя позицию фонового изображения (например, чтобы на фоне страницы все время находился логотип сайта или что-нибудь еще).

Для этого используется свойство **background-attachment**, которое может принимать два значения:

scroll – фоновое изображение прокручивается вместе с текстом и другим содержимым;

fixed – фоновое изображение остается на месте, а содержимое прокручивается на фоне изображения.

2.6. Управление размерами элементов. Плавающие элементы

Для установки размеров объектов в CSS используются два свойства:

width (ширина);

height (высота).

Данные свойства можно применять для разработки базового дизайна: назначения ширины таблиц, изображений, создания заголовков, бокового меню или подножий страниц.

Для обозначения величины ширины или высоты объекта можно пользоваться стандартными единицами измерения:

– пиксели – наиболее простой и понятный вариант, обеспечивающий точное значение высоты или ширины;

– em – примерно то же самое, что и размер шрифта но в условных единицах (величина имеет относительных характер и зависит от базового шрифта браузера);

– проценты – в данном случае величина объекта рассчитывается на основании ширины элемента-контейнера. Например, если ширина заголовка установлена в 75% и заголовок не вложен ни в какие другие элементы веб-страницы с явно определенной шириной, то ширина текста заголовка составит ¾ от ширины окна браузера. Если данный заголовок размещен в теге шириной 200 пикселей, то ширина заголовка будет равна 150 пикселям. Если речь идет о высоте (height), то расчет базируется на высоте элемента-контейнера.

Есть определенные нюансы при комбинировании свойств width и height со свойствами, задающими поля и отступы.

Например:

```
div {
        margin :10px;
        border-width: 5px;
        padding: 15px;
        width: 100px;
}
```

В данном случае фактическая ширина тега div будет равна:

100 + 2*10 + 2*5 + 2*15 = 160 пикселей.

Это справедливо для браузеров Firefox, IE 6 и выше.

Для браузера IE 5 вместо 160 будет 120 пикселей (т.к. там учитываются только свойства margin и width).

Свойство overflow

Если содержимое тега, в стиле которого заданы точные размеры, вылезает за рамки элемента, то может произойти следующее:

– при отображении в IE (по крайней мере, в версиях до 6.0 включительно) размеры элемента будут растянуты по величине его содержимого;

– при отображении в других браузерах содержимое вылезет за пределы элемента.

Для решения данной проблемы можно использовать свойство **overflow**.

Оно имеет четыре значения, с помощью которых можно указать, как должно отображаться содержимое, выходящее за пределы блочного элемента (пример использования свойства **overflow** с разными значениями приведен на рис. 16).

– **visible** – это значение, принимаемое браузером по умолчанию. Имеет тот же эффект, что и отсутствие установки свойства overflow;

– **scroll** – позволяет добавить в элемент полосы прокрутки. Параметр создает своего рода окно мини-браузера, подобное HTML-рамкам (фреймам). Недостаток – полосы прокрутки отображаются всегда, даже если содержимое полностью помещается внутри блока;

– **auto** – аналогично предыдущему свойству, но добавляет полосы прокрутки только при необходимости;

– **hidden** – скрывает любое содержимое, выходящее за пределы блочного элемента (т.е. не вмещающаяся часть содержимого будет просто не видна).

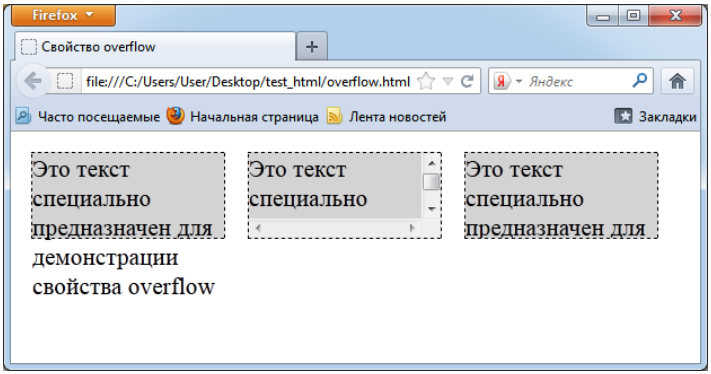

Рис. 16. Пример свойства overflow со значениями visible, scroll и hidden (слева направо)

Управление плавающими элементами

Элементы HTML по умолчанию отображаются в обычном порядке начиная от верхнего края браузера к нижнему один под другим. Есть возможность изменить данный порядок компоновки элементов, воспользовавшись свойством **float**.

Свойство **float** перемещает любой элемент в нужное место, выравнивая его по левому или правому краю страницы. В процессе перемещения содержимое, находящееся ниже позиционируемого плавающего элемента, смещается вверх и плавно обтекает сам плавающий элемент. Таки образом, можно управлять размещением изображений в тексте или создавать боковые панели меню на страницах.

В качестве значения может указываться одно из трех ключевых слов:

– left – перемещает стилизуемый элемент влево, при этом содержимое, находящееся ниже его обтекает правый край элемента;

– right – перемещает элемент вправо;

– none – отменяет обтекание и возвращает элемент в его обычную позицию.

Данное свойство можно применять не только к тегу <img>, но и к тегу , содержащему другие HTML – теги.

Если добавить к данному тегу свойство width, то можно создать боковое меню, примечание или отдельный блок на web-странице.

Пример использования свойства float со значением left приведен на рис. 17. В данном случае оно выставлено для изображения.

Для отмены правил обтекания можно воспользоваться свойством **clear**, которое позволяет запретить обтекание элемента с одной или с двух сторон.

Данное свойство указывает браузеру, что стилизуемый элемент не должен обтекать плавающий элемент. Это приводит к тому, что он смещается вниз по тексту web-страницы, располагаясь ниже плавающего элемента.

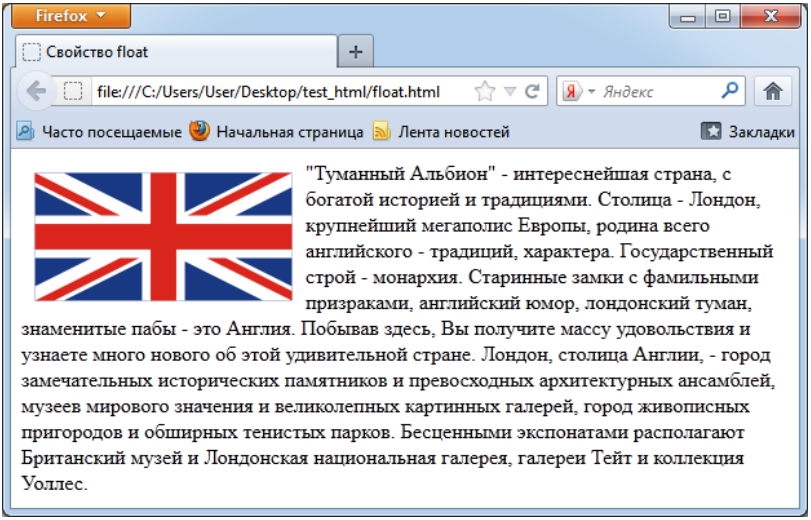

Рис. 17. Пример использования свойства float для обтекания изображения текстом

Данное свойство допускает следующие значения:

– left – стилизуемый элемент смещается вниз относительно плавающего с установленным левым обтеканием, но правое обтекание остается в силе;

– right – стилизуемый элемент смещается вниз относительно плавающего с установленным правым обтеканием, параметр левого обтекания остается в силе;

– both – вызывает перемещение стилизуемого элемента относительно плавающего с установленным правым или левым обтеканием.

Например, если необходимо поместить блок информации об авторском праве, который должен всегда располагаться внизу страницы и не должен обтекать другое содержимое, то для него следует создать следующий стиль:

```
#copyright {
        clear: both;
}
```

Встроенные и блочные элементы

В CSS все элементы делятся на два вида: встроенные и блочные.

В блочных элементах (block) создается разрыв строки перед тегом и после него. Например это теги , <p>, <table> и т.д.

Встроенные элементы (inline) не создают разрывов строк, а размещаются на одной строке с содержимым соседних тегов. Например , , <cite> и т.д.

В большинстве случаев CSS работает с данными элементами одинаково. Можно устанавливать шрифты, цвет, фоновые параметры, границы к обоим типам элементов. Но к встроенным элементам нельзя применять свойства, устанавливающие отступы и поля сверху или снизу (можно обойти данное ограничение путем установки межстрочного интервала).

Если необходимо, чтобы блочный элемент вел себя как встроенный (или наоборот) можно воспользоваться свойством **display**. Это свойство может принимать следующие значения:

– inline – заставляет браузер интерпретировать стилизуемый элемент как встроенный;

– block – заставляет браузер интерпретировать стилизуемый элемент как блочный;

– none – заставляет браузер вообще не отображать данный элемент.

2.7. Разметка страницы на основе плавающих элементов

Изначально для разметки страницы (разбиения ее на области типа баннера, навигации, основного содержимого и сведений об авторах) использовался тег table. Это было простое, но неэффективное решение – код страницы становился громоздким и неудобочитаемым.

CSS дал возможность осуществлять разметку страницы с помощью плавающих элементов. Каждая часть веб-страницы оформлялась как тег , к которому был применен стиль, позиционирующий элемент в нужном месте с помощью свойств float, clear, width и height.

Осуществлять разметку веб-страницы можно несколькими способами:

– с фиксированной шириной элементов;

– с непостоянной шириной элементов;

– с использованием гибкого дизайна.

Каждый вариант имеет свои достоинства и недостатки. Первый вариант достаточно прост в реализации и неплохо смотрится, но на мониторах с разным разрешением будет выглядеть весьма по-разному.

Второй вариант может соответствовать любой ширине браузера, но на больших мониторах может смотреться несколько размашисто.

Третий вариант основывается на указании постоянного размера, но не в пикселях, а в em. Таким образом, размер элементов будет зависеть от базового шрифта браузера (т.е. при его изменении будет и меняться размер элементов, что удобно для людей с плохим зрением).

Разметка, основанная на плавающих элементах, использует возможности свойства float, для позиционирования элементов на странице относительно друг друга.

Рассмотрим наиболее простой вариант разметки страницы (рис. 18).

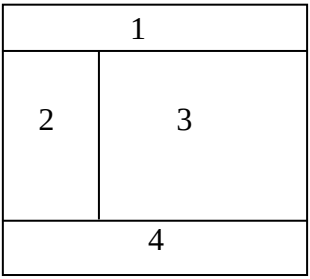

Рис. 18. Структура типовой веб-страницы

Здесь 1 – баннер, 2 – навигационная панель, 3 – основное содержание, 4 – подножие страницы.

Элементы страницы следует описывать в порядке их нумерации.

Последовательность действий будет примерно следующей.

1. Заключить каждый элемент страницы в тег . Каждому элементу присвоить свой уникальный идентификатор (ID). Например, баннеру – “banner”; навигации – “sidebar”; основному содержимому – “main”; подножию – “footer”.

2. Создать стиль для баннера (настроить цвет фона, фоновое изображение, поля и отступы и т.д.).

3. Создать стиль для бокового меню. Обязательно указать свойство float со значением left. Установить ширину для бокового меню в пикселях или в em. В итоге должно получиться, например:

```
#sidebar {
        width: 140px;
        float: left;
}
```

4. Создать стиль для основного содержания. Обязательно установить свойство поля слева (margin-left) в значение, равное (или несколько большее) ширине бокового меню. Например:

```
#main {
        margin-left: 145px;
}
```

5. Создать стиль для подножия. Чтобы панель со сведениями об авторских правах всегда находилась под основным содержанием и главным меню следует добавить в стиль свойство clear со значением both. Например:

```
#footer {
        clear: both;
}
```

6. При необходимости – создать «обертку» вокруг элементов страницы, для задания фиксированной ширины. Для этого следует разместить все элементы страницы внутри тега , помеченного идентификатором (например,), и создать новый стиль. Например:

```
#wrapper {
        margin-left: auto;
        margin-right: auto;
        width: 1000px;
}
```

В данном случае вся страница помещается в «обертку» шириной 1000px и выравнивается по середине экрана (с помощью указания значения auto в свойствах margin-left и margin-right).

Управление перемещением столбцов

С помощью CSS можно разбивать содержимое страницы не только на два, но и на большее количество столбцов.

Например, требуется реализовать следующую схему (рис. 19).

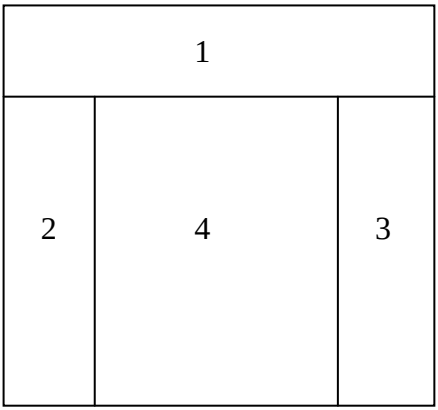

Рис. 19. Структура веб-страницы с двумя боковыми колонками

Цифрами обозначены:

1 – баннер;

2 – левое боковое меню;

3 – правое боковое меню;

4 – основное содержимое.

Элементы должны описываться именно в том порядке, в котором указаны.

Для добавления правого бокового меню, надо добавить тег div, его описывающий, и присвоить ему отдельный идентификатор или класс. В стиле данного меню следует задать ему фиксированную ширину и сместить его вправо. Например:

```
#rightmenu {
        float: right;
        width: 100px;
}
```

Для стиля, описывающего основное содержимое, следует добавить соответствующий отступы справа и слева, чтобы он поместился между левым и правым меню. Например:

```
#main {
        margin-left: 105px;
        margin-right: 105px;
}
```

Иногда требуется отображать элементы страницы не в том порядке, который приведен выше. Например, для более удобной и корректной обработки сайта поисковыми машинами, следует размещать основное содержимое как можно выше (часто для поисковых машин, текст размещенный в HTML выше имеет более высокий приоритет, чем остальной текст). Есть два варианта решения данной проблемы:

Первый вариант заключается в размещении элементов страницы в следующем порядке:

1 – баннер;

2 – левое боковое меню;

3 – основное содержимое;

4 – правое боковое меню.

Для правильного отображения основного содержимого, следует добавить в стиль тега основного содержимого свойство **float: left** и задать ему фиксированную ширину (так, чтобы поместились все столбцы). Например:

```
#main  {
        margin-left:5px;
        margin-right: 5px;
        width: 590px;
        background-color: blue;
        float: left;
}
```

Также можно указать небольшие отступы, чтобы было небольшое свободное пространство между блоками.

Второй вариант заключается в размещении элементов в следующем порядке:

1 – баннер;
2 – основное содержимое;
3 – левое боковое меню;
4 – правое боковое меню.

Для данного размещения следует использовать методику перемещения элемента внутри другого плавающего элемента. Для этого следует добавить еще один тег div, «обертывающий» основное содержимое и левое боковое меню. В его стиле должно быть свойство **float** со значением **left**.

Внутри данного тега надо сместить левое боковое меню влево, основное содержимое – вправо. Стили для данных компонентов могут иметь следующий вид.

```
#main  {
        margin-left:5px;
        margin-right: 5px;
        width: 590px;
        background-color: blue;
        float: right;
}

#leftmenu {
        width: 200px;
        float: left;
        background-color: yellow;
}
```

Содержимое тела документа может быть представлено следующим образом:

```
<div id="wrapper">
  <div id="banner">
```

```
            BANNER
    </div>
    <div id="wrapper1">

             <div id="main">
                    MAIN
            </div>

            <div id="leftmenu">
                    NAV1
            </div>

    </div>

    <div id="rightmenu">
            NAV2
    </div>

  </div>
```

Работа с отрицательными полями

Как известно, значения полей и отступов могут быть как положительными, так и отрицательными. Используя отрицательные поля, можно поместить теги div в любом порядке в HTML-коде, а затем разместить их в другом порядке на экране.

В данном методе есть особенность – необходимо тщательно высчитывать отступы, поля и размеры элементов, чтобы достичь корректного результата.

Рассмотрим порядок действий при разметке страницы с использованием отрицательных полей.

1. Добавить тег , который определяет все содержимое страницы («обертка» - wrapper). Этот шаг предусматривает создание контейнера для установки фиксированной ширины всего содержимого страницы и предоставляет легкий спо-

соб сделать баннер, столбцы и нижний колонтитул одинаковой ширины.

2. Установить ширину для созданного тега .

3. Описать каждый столбец в теге с атрибутом ID или class. Эта часть процесса – практически такая же, как при создании любой блочной разметки.

Порядок описания – обертка, баннер, основное содержимое, левое боковое меню, правое боковое меню.

4. Переместить разделы div для каждого столбца с помощью свойства float. Наиболее распространенный вариант – левое боковое меню и основное содержимое влево, а правое боковое меню – вправо.

5. Установить ширину всех столбцов. Определять ширину плавающего элемента надо практически всегда, но при этом следует учитывать, что полная ширина элемента в окне браузера – это сумма значений CSS-свойства width, левых и правых полей, отступов и границ для каждого столбца. Если неправильно рассчитать ширину элементов может возникнуть перепад в размещении элементов.

6. Добавить левое поле к главному столбцу. Именно здесь методика отрицательных полей отличается от ранее указных методов разметки. Левое поле должно занимать то же пространство, что и левое боковое меню. Если ширина бокового левого поля – 200 пикселей, то следует установить левое поле основного содержимого (margin-left: 200px;). Если у бокового меню есть поля и отступы, то их следует учесть.

7. Применить отрицательное поле к тегу левого бокового меню. На данный момент левое боковое меню перемещено влево, но поскольку оно идет после главного столбца оно появляется справа от основного содержимого.

Для перемещения левого меню в правильную позицию, следует добавить к нему отрицательный отступ слева, который рассчитывается по следующей формуле:

Отступ = (-1)*(ширина основного столбца + левые и правые отступы и поля + левые и правые границы).

Например, если ширина основного столбца равна 598 пикселей, левое поле 200 пикселей, а правая и левая граница имеют ширину по 1 пикселю, то отступ будет равен

(–1) * (598 + 200 + 1 + 1)= –800 пикселей

Таким образом, следует добавить в стиль левого меню свойство

left-margin: -800px;

8. Устранить ошибку двойного поля старых версий Internet Explorer. При использовании отрицательного поля, в IE версий 6 и ниже возникает ошибка двойного поля, приводящая к удвоению левого поля основного столбца. Устранение данной проблемы состоит в добавлении к стилю основного столбца свойства display: inline.

При правильном расчете данный метод разметки является самым гибким. Например, если необходимо поменять местами боковые меню, то можно просто поменять местами содержимое их стилей.

Текст файла со стилями может быть примерно следующим:

```
#banner {
        background-color: aqua;
}

#main  {
        margin-left:200px;
        width: 598px;
        border-left: 1px solid black;
        border-right: 1px solid black;
        background-color: blue;
        float: left;
}
```

```
#leftmenu {
        width: 200px;
        float: left;
        background-color: yellow;
        margin-left:-800px;
}
#rightmenu {
        width: 200px;
        float: right;
        background-color: green;
}
#wrapper {
        width: 1000px;
        margin-left: auto;
        margin-right: auto;
}
```

Текст файла основного содержимого будет примерно следующим:

```
<html>
  <head>
        <style>
                @import url("styles.css");
        </style>
  </head>
  <body>
        <div id="wrapper">
                <div id="banner">
                        BANNER
                </div>

                <div id="main">
                        MAIN
                </div>
```

```
                <div id="leftmenu">
                        NAV1
                </div>

                 <div id="rightmenu">
                        NAV2
                </div>
        </div>
  </body>
</html>
```

Создание столбцов на всю высоту

Табличная разметка по сравнению с блочной является сложной и громоздкой, но имеет одно преимущество – возможность создавать столбцы равной высоты. Их применение позволяет добавлять фоновый цвет или графику к отдельной ячейке и делать так, чтобы таблица или строка заполняла всю высоту страницы. В CSS выравнивание столбцов про высоте следует воспользоваться обходными путями.

Например, существует страница следующей структуры (рис. 20).

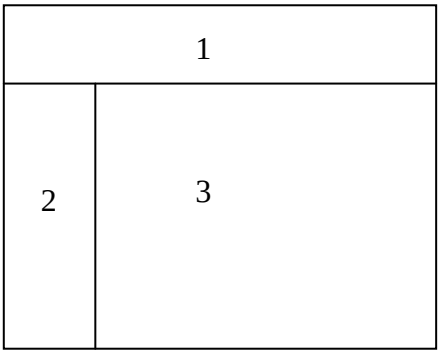

Рис. 20. Страница с боковой колонкой

Необходимо сделать так, чтобы левое меню (2) и основное содержимое (3) были одинаковы по высоте. Для этого можно сделать следующее:

– «обернуть» блоки 2 и 3 тегом div, присвоив ему особый идентификатор (например, «wrapper») – в итоге он будет иметь высоту, такую же как самый длинный элемент;

– подобрать изображение, цвет и ширина которого совпадают с цветом и шириной левого меню;

– сделать данное изображение фоном для тега обертки прописав ему следующие свойства:

```
#wrapper {
        background: url (images/back.jpg) repeat-y left top;
}
```

Браузер разместит данное изображение прямо под боковым меню, создавая иллюзию того, что у меню есть фоновый цвет.

Если необходимо применить данный метод для правого столбца, то следует прописать:

```
#wrapper {
        background: url (images/back.jpg) repeat-y right top;
}
```

Если необходимо использовать данный метод для двух столбцов, то можно сделать две обертки вокруг боковых меню и основного содержимого и прописать для них два стиля:

```
#wrapper1 {
        background: url (images/back.jpg) repeat-y left top;
}
#wrapper2 {
        background: url (images/back.jpg) repeat-y right top;
}
```

2.8. Стилизация ссылок

Большинство браузеров распознает четыре основных состояния ссылок:

– непосещенная ссылка;

– посещенная (visited) ссылка;

– ненажатая (rollover) ссылка, над которой находится указатель мыши;

– нажатая (active) ссылка, по которой еще не осуществлен переход.

Как рассказывалось ранее, CSS предоставляет четыре псевдокласса селекторов, соответствующих этим состояниям:

:link
:visited
:hover
:active

Используя псевдоклассы, можно применить различное форматирование для каждого состояния ссылки, обеспечивая соответствующие подсказки и однозначно давая понять посетителям сайта, какие ссылки он уже посещал, а какие – нет.

Предположим, необходимо изменить цвет текста непосещенной ссылки с синего цвета на оранжевый. Для этого следует добавить следующий стиль:

```
a:link {
        color: orange;
}
```

При щелчке кнопкой мыши на этой ссылке, ее состояние изменяется на посещенное, а цвет в большинстве браузеров становится фиолетовым. Чтобы изменить его на красный, след:

```
a:visited {
        color: red;
}
```

С помощью псевдокласса :hover можно полностью изменить вид гиперрсыылки при наведении на нее указателя мыши. Например, можно поменять ее цвет и стиль шрифта на курсив:

```
a:hover {
        color: aqua;
        font-style: italic;
}
```

Для полной стилизации гиперссылки можно использовать псевдокласс :active, который меняет вид гиперссылки на те доли секунды, когда пользователь щелкает кнопкой мыши на ссылке (впрочем, он может полюбоваться эффектом подольше, нажав и удержав кнопку мыши в тот момент, когда она находится на ссылке).

Например, можно выделить ссылку зеленым цветом:

```
a: active {
        color: green;
}
```

Если для стилизации ссылок используются все псевдоклассы, то они должны располагаться в определенной последовательности:

:link
:visited
:hover
:active

Для запоминания данного порядка можно использовать аббревиатуру по буквам соответствующих стилям английских слов **L**o**V**e/**HA**te (в переводе «любовь/ненависть»).

Выборка отдельных ссылок

Стили, которые создавались выше, представляют собой простейшие стили ссылки <a>. Они применяются к ссылкам в

определенном состоянии, но при этом стилизуются абсолютно все ссылки, независимо от их расположения на веб-странице. Что же делать, если нужно отформатировать одни ссылки одним способом, а другие – другим? Существует простое решение – применить стилевые классы к нужным тегам.

Допустим, на веб-странице имеется набор ссылок, одни из которых указывают на другие страницы этого же сайта, другие – на внешние источники. Последние можно отформатировать особым образом, чтобы посетители заранее знали, что при щелчке кнопкой мыши на ссылке они перейдут с текущего сайта на какой-либо другой. В данном случае рекомендуется применить стилевой класс:

```
<a href=”http://google.ru” class=”external”>Google</a>
```

Для стилизации данной ссылки можно создать ряд стилей:

```
a.external:link {
        font-family: Arial;
        font-size: 12pt;
        color: red;
}
a.external:visited { color: blue; }
a.external:hover { color: orange; }
a.external:active { color: green; }
```

Имя тега <a> указывать необязательно.

Если набор ссылок располагается в одной области веб-страницы, то можно применить селекторы потомков, позволяющие адресовать стиль к тегу, вложенному в другой тег. Предположим, имеется пять ссылок, которые ведут на основные разделы сайта.

Они представляют собой главную панель навигации, и необходимо придать им характерный вид. Для этого следует за-

ключить эти ссылки в тег и применить к нему стилевой класс или стиль с идентификатором:

```
<div id="nav">.
```

Теперь появилась возможность произвести выборку и отформатировать только эти ссылки:

```
#nav a:link {color: red;}
#nav a:visited {color: blue;}
#nav a:hover {color: black;}
#nav a:active {color: yellow;}
```

Если на веб-странице тег <a> используется не только для гиперссылок, но и для якорей, то часто возникает необходимость стилизовать их отдельно. Одним из вариантов является использование селекторов атрибутов – учитывая, что у гиперссылки есть атрибут href, а у якоря – name.

Например:

```
a[href] {color: white;}
a[name] {color: aqua;}
```

Подчеркивание ссылок

Еще с самого начала развития Интернет ярко-синий подчеркнутый текст сигнализировал: «Нажмите здесь, чтобы перейти туда-то». Однако стандартные атрибуты подчеркивания и цветового оформления ссылок – это то, что в первую очередь изменяет любой веб-дизайнер. Рассмотрим ряд вариантов.

Полное удаление подчеркивания. Чтобы убрать стандартное подчеркивание, следует использовать свойство text-decoration со значением none:

```
a:link {text-decoration: none;}
```

Естественно, полное удаление подчеркивания может смутить посетителей сайта. Поэтому надо каким-то образом выделять гиперссылки в тексте, чтобы посетитель догадывался о значении данного элемента.

Добавление подчеркивания только при наведении на ссылку указателя мыши. Некоторые веб-дизайнеры убирают подчеркивание для всех ссылок, выделяют их каким-то другим способом, а затем включают атрибут подчеркивания только при наведении указателя мыши. Чтобы обеспечить такой эффект, следует удалить подчеркивание ссылок, а затем повторно ввести его в псевдокласс : hover:

```
a:link, a:visited {
        text-decoration: none;
        background-color: lightblue;
}

a:hover {
        background-color: transparent;
        text-decoration:underline;
}
```

Использование свойства нижней границы. Существует возможность управлять цветом, толщиной или стилем стандартной линии подчеркивания ссылок, которое всегда отображается в виде линии толщиной 1 пиксель того же цвета, что и текст.

Большего разнообразия можно добиться путем использования вместо подчеркивания свойства border-bottom. Скрыть обычное подчеркивание и добавить черту в виде пунктирной линии-границы можно следующим образом:

```
a {
        text-decoration: none;
        border-bottom: dashed 2px red;
}
```

Создание ссылок-кнопок

Можно придать ссылкам вид кнопок, присутствующих в окнах и на панелях инструментов компьютерных программ. Для этого следует воспользоваться свойствами border, background-color и padding.

Предположим, существует ссылка, которой присвоен класс «button»:

```
<a href=”price.html” class=”button”>Прайс-лист</a>
```

Для добавления рамки-контура требуется добавить следующий стиль:

```
a.button {
        text-decoration: none;
        border: solid 1px red;
}
```

Можно окрасить кнопку путем установки цвета фона:

```
a.button {
        text-decoration: none;
        border: solid 1px red;
        background-color: gold;
}
```

Также можно добавить отступы, чтобы текст ссылки не примыкал к рамке:

```
a.button {
        text-decoration: none;
        border: solid 1px red;
        background-color: gold;
        padding: 3px 5px 3px 5px;
}
```

Можно добиться, чтобы кнопка выглядела объемной путем использования разных стилей для линий границ.

Широко применяемая дизайнерская методика превращения плоской кнопки в объемную предполагает использование четырех границ разного цвета. Придание кнопке объемности не представляет сложности – главное следует помнить, что освещение заставляет прел выглядеть таким. Необходимо представить, что свет падает на одну из четырех сторон; эта обращенная к источнику света сторона – самая светлая, а противоположная — самая темная. Остальные две стороны дол иметь цвета промежуточных оттенков между «светлыми» и «темными».

Например, можно попробовать такой вариант:

```
a.button {
        background-color: gold;
        color: green;
        font-weight: bold;
        border-width: 3px;
        border-style: solid;
        border-top-color: lightblue;
        border-left-color: deepskyblue;
        border-right-color: blue;
        border-bottom-color: darkblue;
        text-decoration: none;
        padding: 5px;
}
```

Примеры кнопок ссылок, созданных, с помощью данного стиля приведены на рис. 21.

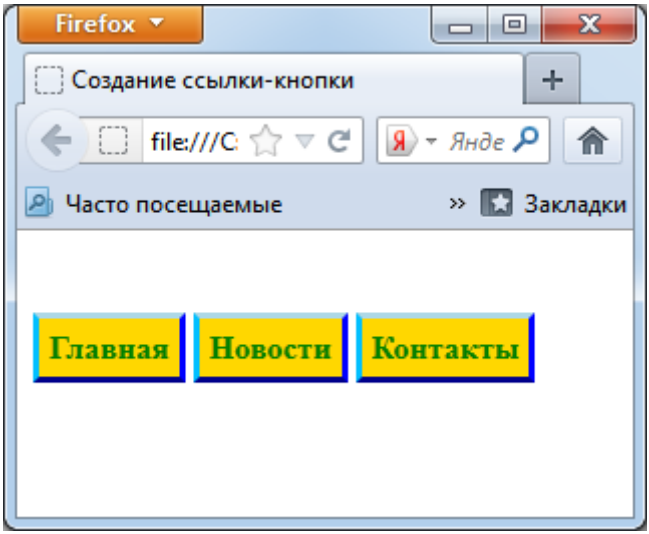

Рис. 21. Примеры кнопок-ссылок, созданных с помощью CSS

Также можно создать стиль для состояния :hover. Кнопки будут реагировать на наведение посетителем на ссылку указателя мыши, обеспечивая интерактивность и обратную связь. В случае с объемной кнопкой при ее нажатии очень эффективен метод инверсной смены цветов: темный фон кнопки дол стать светлым, светлая граница — темной и т. д.

Например, чтобы инвертировать цвета рамок можно создать следующий стиль:

```
a.button:hover {
        border-top-color: darkblue;
        border-left-color: blue;
        border-right-color: deepskyblue;
        border-bottom-color: lightblue;
}
```

Если необходимо создать кнопки фиксированного размера, можно преобразовать ссылку в блочный элемент, после чего задать фиксированную ширину и выравнивание надписи на кнопке посередине. Для этого следует добавить в стиль a.buttons следующие свойства:

```
display: block;
width: 120px;
text-align: center;
```

При преобразовании ссылки из из встроенного элемента в блочный в браузере Internet Explorer 6 возникает ошибка – по-прежнему можно щелкать только на тексте ссылки, остальная часть кнопки является неактивной. Чтобы исправить ситуацию, следует добавить стиль, явно определяющий высоту для тега <a> (причем высота может быть любой, вплоть до 1 пикселя). Поскольку он уменьшает ссылки до нечитаемых размеров в в более поздних версиях Internet Explorer и остальных браузерах, необходимо скрыть от них стиль. Один из способов использовать конструкцию *html:

```
*html a.buttons {height: 1px;}
```

Использование изображений. Добавление изображений – достаточно легкий и эффективный способ улучшения навигации сайта. Существует множество методики вариантов дизайна, но следует заметить, что в хороших грамотных методов не используется тег <img>.

Вместо этого используется прием с CSS-свойством background-image, с помощью которого можно добавить фоновое изображение к ссылке. При этом следует руководствоваться следующими прицип:

– грамотно использовать значение свойства no-repeat;

– управлять позиционированием изображения с помощью свойства background-position;

– добавлять отступы с помощью свойства padding (если используется фоновое изображение как значок слева или справа от надписи на кнопке);

– использовать псевдоклассы (например, для смены фонового изображения для посещенной ссылки или при наведении на нее).

2.9. Создание панелей навигации

Каждому сайту требуется хорошая управляющая панель с элементами навигации: во-первых, чтобы привести посетителей к нужной им информации, а во-вторых, чтобы им понравилось управление сайтом и они вернулись сюда вновь. Содержимое большинства сайтов организовано в виде разделов. Например, продукция, контактная информация, журнал отзывов и т. д. Такая организация позволяет посетителям хорошо ориентироваться в содержимом сайта, и они точно знают, что и где можно найти. В большинстве случаев ссылки на основные разделы сайта можно найти на панели навигации.

В качестве основы для панели навигации можно воспользоваться маркированным списком, который можно стилизовать и преобразовать в вертикальную или горизонтальную панель навигации.

Например, существует такой маркированный список гиперссылок, помеченный соответствующим классом:

```
<ul class="nav">
  <li><a href="main.html">Главная</a></li>
  <li><a href="comp.html">О компании</a></li>
  <li><a href="prod.html">Продукция</a></li>
  <li><a href="price.html">Прайс-лист</a></li>
  <li><a href="contacts.html">Контакты</a></li>
</ul>
```

Для начала необходимо провести определенную подготовительную работу.

1. Удалить маркеры. Для этого можно создать стиль с соответствующим свойством:

```
ul.nav {
        list-style-type:none;
}
```

2. Убрать отступы и поля. В стиль надо добавить еще пару свойств:

```
ul.nav {
        list-style-type:none;
        margin-left: 0;
        padding-left: 0;
}
```

Теперь можно начинать создавать навигационную панель.

Вертикальная панель навигации

Для создания вертикальной панели навигации необходимо осуществить следующие действия.

1. Отобразить ссылки в виде блочных элементов. Чтобы выровнять элементы панели навигации по ширине, их следует преобразовать в блочные, создав следующий стиль:

```
ul.nav a {
        display: block;
}
```

2. Ограничить ширину кнопок. В данный момент ширина кнопок панели навигации равна ширине родительского элемента. Для ограничения ширины следует добавить в созданный стиль свойство width:

```
ul.nav a {
        display: block;
        width: 8em;
}
```

Если текст кнопки занимает всего одну строку, то можно его центрировать вертикально, чтобы над и под ним были одинаковые промежутки. Для этого следует установить высоту ссылки и назначить такое же значение свойства межстрочного интервала:

```
ul.nav a {
        display: block;
        width: 8em;
        height: 1.25em;
        line-height: 1.25em;
}
```

В принципе не обязательно явно назначать свойство width, если панель навигации вложена в элемент разметки web-страницы, для которого уже явно установлена ширина – например, боковое меню.

3. Устранить ошибки отображения панели в старых версиях Internet Explorer. Во-первых, убрать лишние промежутки между элементами панели навигации, добавив стиль:

```
ul.nav li {
        display: inline;
}
```

Во-вторых, как уже говорилось ранее, исправить неверное отображение ссылки при преобразовании ее в блочный элемент, добавив:

```
*html ul.nav a {height: 1px;}
```

Далее можно заняться стилизацией кнопок панели навигации – добавить отступы, поля, изображения, изменить фоновый цвет и т.д. Должен получиться результат, похожий на рис. 22.

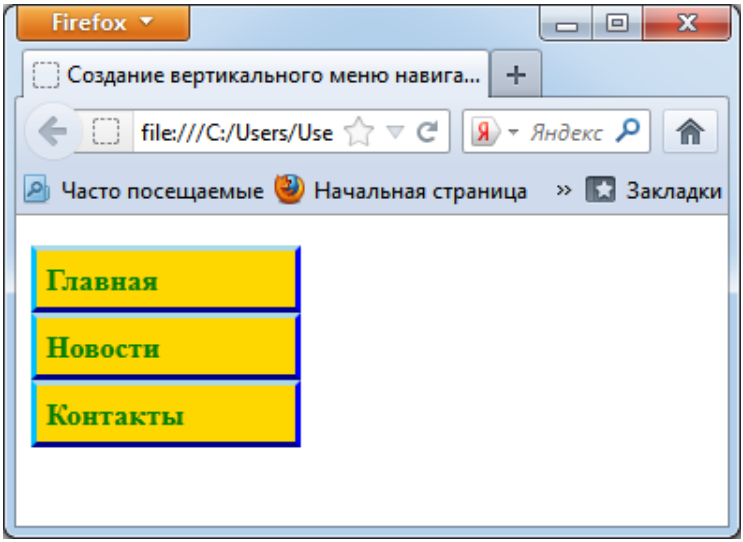

Рис. 22. Вертикальная панель навигации

Горизонтальные панели навигации

Существует два основных метода создания навигационной панели – принципиальная разница заключается в том, что в одном случае получатся кнопки разных размеров (зависит от надписи на кнопке), во втором случае можно получить кнопки одинакового размера.

Вне зависимости от метода, изначально следует удалить маркеры, поля и отступы слева от элементов списка.

Первый способ создания горизонтального меню

1. Преобразовать пункты списка во встроенные элементы. Для этого следует добавить следующий стиль:

```
ul.nav li {
        display: inline;
}
```

Желательно, чтобы панель навигации была не слишком большой – иначе элементы в ней могут разместиться на двух или нескольких строках.

2. Стилизовать ссылки. Можно добавить границу кнопок, убрать подчеркивание ссылок, добавить цвет фона, добавить отступы, а также назначить поле справа, если надо отдалить кнопки друг от друга. Например:

```
ul.nav a {
        border: 1px dashed red;
        text-decoration: none;
        background-color: aqua;
        color: darkgreen;
        padding: 5px 10px 5px 10px;
        margin-right:10px;
}
```

3. Добавить для тега <ul> верхний и нижний отступы. Поскольку теги <a> – встроенные элементы, добавление для них верхнего и нижнего отступов фактически не увеличит высоты ссылок. Вместо этого границы и фон ссылок наложатся на соседние вышестоящие и нижестоящие элементы.

Чтобы решить проблему, следить добавить отступ для тега <ul>, который создает соответствующее пустое пространство, предотвращающее наложение границ и фона ссылок. Для нижнего отступа <ul> следует использовать такое же значение, как размер нижнего отступа ссылки.

Для определения верхнего отступа тега следует добавить 1 пиксель к значению верхнего отступа ссылки (при использовании em просто надо удостовериться в том, что отступ тега <ul> больше).

Например, стиль тега <ul>, соответствующий ссылкам в шаге 2, должен выглядеть следующим образом:

```
ul.nav {
        margin-left: 0;
        list-style: none;
        padding-left: 0;
```

```
        padding-top: 6px;
        padding-bottom: 5px;
}
```

Полученный результат отображен на рис. 23.

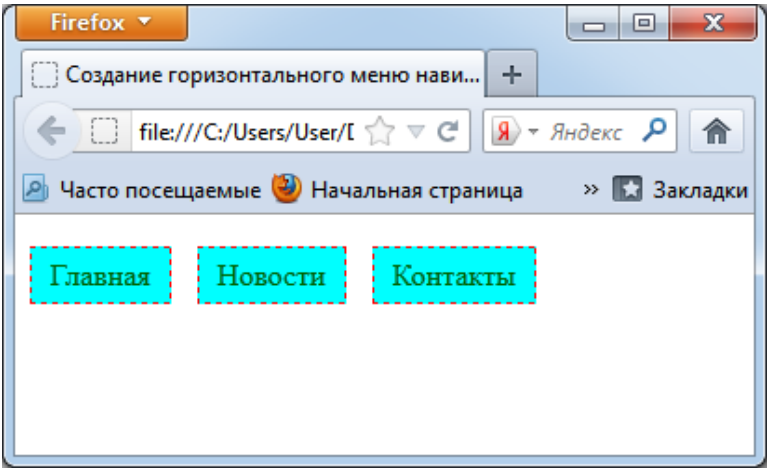

Рис. 23. Горизонтальная панель навигации

Второй способ создания горизонтального меню

Хотя методика с использованием display: inline для создания горизонтальной панели навигации проста, у нее есть один существенный недостаток: нет способа выровнять ширину кнопок. Установка ширины тегов или <a> не приведет к требуемому результату, поскольку это встроенные элементы. Чтобы решить проблему, нужно прибегнуть к более сложному и хитрому решению — применению плавающих элементов.

Последовательность действий следующая:

1. Преобразовать пункты списка в плавающие элементы. Для этого следует добавить стиль:

```
ul.nav li {
        float: left;
}
```

2. Преобразовать ссылки в блочные элементы. Для этого следует добавить стиль:

```
ul.nav a {
        display: block;
}
```

3. Стилизовать ссылки. Аналогично шагу 2 предыдущего способа.

4. Определить ширину кнопок. В стиль, описывающий тег <a> надо добавить свойство width (для него рекомендуется использовать параметр em, так как он наиболее хорошо масштабируем).

5. Назначить для тега <ul> свойство float. Если для тега заданы границы, цвет фона или изображение, придется установить для него свойство float с выравниванием по левому краю. Если этого не сделать, то высота элементов списка выйдет за пределы размеров <ul>.

6. Очистить параметр float для последующего содержимого. Самый простой способ – создать стилевой класс .clear, например:

```
.clear {
        clear: both;
}
```

Созданный класс следует применить к первому тегу, следующему за списком.

В итоге может получиться примерно следующее:

```
ul.nav {
        list-style-type:none;
        margin-left: 0;
        padding-left: 0;
        float: left;
}
```

```
ul.nav li {
        float: left;
}

ul.nav a {
        display: block;
        border: 1px dashed red;
        text-decoration: none;
        background-color: aqua;
        color: darkgreen;
        margin-right: 2px;
        width: 7em;
        text-align:center;
}

.clear {
        clear: both;
}
```

Полученный результат отображен на рис. 24.

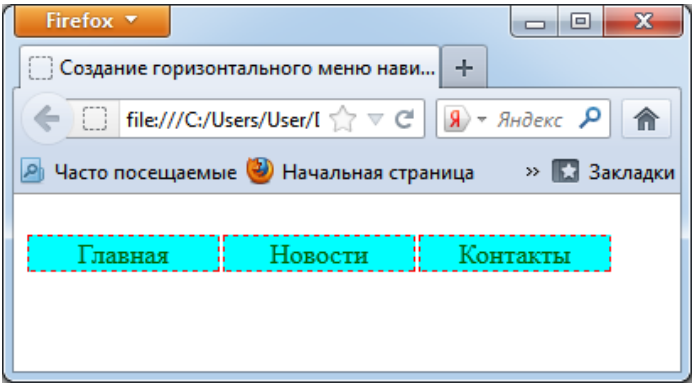

Рис. 24. Горизонтальная панель навигации (второй вариант)

2.10. Позиционирование элементов на веб-странице

Свойства позиционирования

CSS-свойство position позволяет управлять тем, где и как браузер отобразит определенные элементы. Используя его, вы можете, например, поместить боковое меню на странице там, где вы этого желаете, или убедиться, что навигационная панель наверху страницы останется там, даже когда пользователь прокручивает страницу вниз.

CSS предлагает четыре типа позиционирования.

absolute – абсолютное. Такое позиционирование позволяет определять расположение элемента, задавая позиции left, right, top или bottom в пикселях, em или процентах. Вы можете поместить блок на расстоянии 20 пикселей от верхнего и 200 пикселей от левого края страницы.

Следует заметить, что абсолютно размещенные элементы полностью отделены от потока страницы, определенного HTML-кодом. Другими словами, остальные элементы на странице не знают о существовании данного элемента. Они даже могут частично или полностью исчезнуть со страницы, попав под абсолютно размещенный элемент.

relative – относительное. Элемент с таким позиционированием размещается относительно его текущего положения в потоке HTML. Например, устанавливая значение top – 20 пикселей и значение left – 200 пикселей для относительно размещенного заголовка, вы переместите его на 20 пикселей вниз и на 200 пикселей влево от того места, где он появился бы на странице.

В отличие от абсолютного позиционирования элемент из HTML-потока не исключается, и остальные элементы страницы регулируют старое размещение относительно позиционированного объекта. Соответственно, перемещение объекта с относительным позиционированием оставляет «дыру», на месте, где он должен был находиться.

Основная польза относительного позиционирования заключается в том, что кроме перемещения элемента, с его помо-

щью можно установить новую точку привязки для относительно позиционируемых элементов, что будет рассмотрено далее.

fixed – фиксированное. Фиксированный элемент запирается в определенном месте на экране. Определение такого позиционирования сходно с установкой значения fixed для свойства background-attachment. Когда посетитель прокручивает страницу, фиксированные элементы остаются на экране на одной и той же заранее заданной позиции.

static – статическое. Статическое позиционирование просто означает, что элемент показывается в рамках обычного HTML-потока, т.е. так, как элементы позиционируются на странице по умолчанию.

Итак, чтобы изменить расположение любого элемента, просто используйте свойство position, которому присваивается одно из четырех значений: static, absolute, relative, fixed.

Установка значений расположения

Область отображения браузера, также называемая областью просмотра, имеет верхний, нижний, правый и левый края. Для каждого из этих четырех краев есть соответствующее CSS-свойство: top, bottom, left и right. Нет необходимости задавать их все – обычно достаточно двух, чтобы определить позицию элемента на странице. В качестве значения может использоваться любая из действующих в CSS единиц измерений: пиксели, em, проценты и т.д. Можно использовать и отрицательные значения.

Таким образом, для размещения элемента следует указать свойство position и два из вышеуказанных свойств. Можно также управлять шириной элемента, установив свойство width.

Например, чтобы поместить баннер шириной 760 пикселей в определенном месте относительно левого и верхнего краев страницы, можно создать следующий стиль:

```
#banner {
        position: absolute;
        left: 100px;
```

```
        top: 50px;
        width: 760px;
}
```

Если необходимо размещать баннер относительно правого края страницы, то надо изменить стиль следующим образом:

```
#banner {
        position: absolute;
        right: 100px;
        top: 50px;
        width: 760px;
}
```

Можно с помощью одновременного применения свойств left и right косвенно задать ширину элемента. Но следует иметь в виду, что данный метод не работает в браузере Internet Explorer 6 и ниже – левая позиция показывается корректно, а правая игнорируется.

Комбинирование абсолютного и относительного позиционирования

На самом деле абсолютно позиционированный элемент помещается относительно границ его ближайшего предка. Иначе говоря, если уже был создан элемент с абсолютным расположением (например, тег , который появится на расстоянии 100 пикселей от вершины окна браузера), то любые абсолютно позиционированные элементы внутри этого тега размещаются относительно верхнего, нижнего, левого и правого краев области тега.

Например, есть тег с id d1 и к нему применен следующий стиль:

```
#d1 {
        background-color: lightgray;
```

```
        position: absolute;
        top: 5em;
        left: 5em;
}
```

Есть также тег , вложенный в этот блок. Применение абсолютного позиционирования для этого тега позволит поместить его относительно абсолютно позиционированного родительского элемента. Добавление свойства bottom со значением 0 переместит блок не к основанию экрана, а к основанию предка. Установка свойства right в значение 5em будет применяться к правому краю его родителя.

Позиционирование подчиняется следующим правилам:

– тег расположен относительно окна браузера, если у него абсолютное позиционирование и он не находится внутри любого другого тега, к которому применено абсолютное, относительное или фиксированное позиционирование;

– тег определен относительно сторон другого элемента, если он находится внутри другого тега с абсолютным, относительным или фиксированным позиционированием.

Как уже говорилось ранее, относительное позиционирование элемента может создавать новую точку привязки для других абсолютных элементов, размещенных внутри него.

Например, если мы хотим создать заголовок первого уровня, а его верхнем правом углу поместить изображение, например вот так:

```
<h1>
Заголовок
<img src=”images/pic01.jpg”>
</h1>
```

, то можно сделать два следующих стиля:

```
h1 {
        position: relative;
}
```

```
img {
        postion: absolute;
        top: 0;
        right: 0;
}
```

Наложение элементов

При размещении абсолютно позиционируемых элементов на странице, может возникнуть такая ситуация, что они будут частично или полностью перекрывать друг друга (или накладываться друг на друга).

Обычно порядок наложения элементов соответствует их порядку в HTML-коде страницы. На странице с двумя абсолютно позиционированными тегами второй тег из HTML появится выше первого. Но можно управлять порядком наложения элементов, используя свойство CSS z-index. Свойство получает числовое значение, например:

```
z-index: 3;
```

Чем больше значение, тем ближе к вершине набора появится элемент. Элемент со свойством z-index, имеющим наибольшее значение появится над всеми другими.

Использование CSS-позиционирования для разметки страницы

Позиционирование – это весьма мощный инструмент CSS, который можно (и нужно) использовать для верстки страницы. Рассмотрим несколько примеров.

Предположим, что необходимо разработать схему из 5 блоков - баннера, основного содержания, двух боковых меню и подножия. Также надо будет добавить тег обертки, содержащий все блоки, находящиеся под баннером.

Блоки будут размещаться на странице в следующем порядке

1 – баннер;
2 – основное содержание;
3 – левое боковое меню;
4 – правое боковое меню;
5 – подножие.

Общая схема будет выглядеть следующим образом (рис. 25).

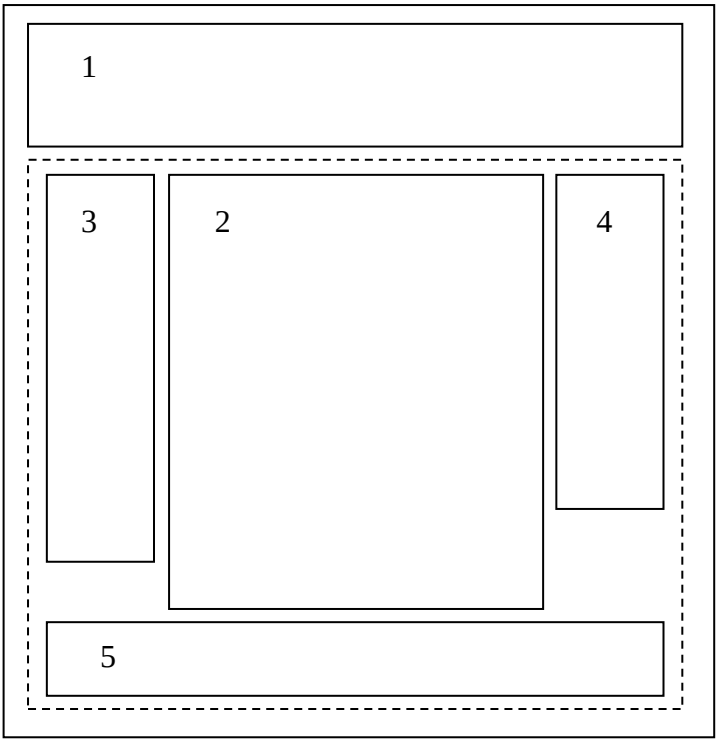

Рис. 25. Типовая схема веб-страницы с двумя боковыми меню

Процесс верстки будет состоять из следующих этапов.

1. Добавление тегов для каждого раздела страницы. Каждый тег должен иметь свой уникальный ID, чтобы для каждого блока можно было создать свой стиль. Теги

 для разных блоков должны быть расположены в порядке, указанном выше.

Можно присвоить тегам следующие ID:

banner – баннер;

mainContent – основное содержание;

leftSidebar – левое меню;

rightSidebar – правое меню;

footer – подножие.

2. Включение всего HTML-кода для главного содержимого, боковых меню и нижнего колонтитула в другой тег – обертку. Ему следует тоже присвоить свой ID, например contentWrapper.

3. Задание тегу обертки относительной позиции. Для этого следует создать стиль следующего вида:

```
#contentWrapper {
        position: relative;
}
```

4. Применение абсолютного позиционирования для боковых меню. Для размещения боковых меню в нужных позициях следует добавить два стиля:

```
#leftSidebar {
        position: absolute;
        left: 0;
        top: 0;
        width: 175px;
}
#rightSidebar {
        position: absolute;
        right: 0;
        top: 0;
width: 180px;

}
```

Следует заметить, что в данных стилях надо обязательно задать фиксированную ширину, чтобы ограничить боковые меню и оставить место для основного содержимого.

5. Создание стиля для главного содержимого. Самое главное – добавить нужные значения для отступов слева и справа, где размещаются боковые меню.

```
#mainContent {
        margin-left: 175px;
        margin-right: 180px;
}
```

Отступы следует сделать больше или равными ширине соответствующих меню.

6. При необходимости задать поля для нижнего колонитула, чтобы он не перекрывался боковыми меню. Другой вариант – заполнять страницы таким образом, чтобы основное содержимое по высоте всегда было больше боковых меню.

Верстка с использованием фиксированного позиционирования

Во многих случаях бывает нужно, чтобы некоторые элементы страницы были постоянно видимыми, например, навигационная панель, поле для поиска или логотип сайта. При слишком большой высоте содержимого страницы эти элементы во многих случаях «терялись» при прокрутке.

В свое время решением подобных проблем было использование фреймов. Но фреймы обладали множеством недостатков и постепенно вышли из употребления. Но с помощью свойства CSS position со значением fixed можно имитировать фреймы, фиксируя некоторые элементы страницы в нужных местах, а остальным разрешать прокрутку.

Предположим, что необходимо создать страницу из 4-х блоков:

1 – баннер;

2 – основное содержимое;

3 – боковое меню;

4 – подножие.

Основное содержимое можно будет прокручивать, а остальные элементы будут фиксированными (рис. 26).

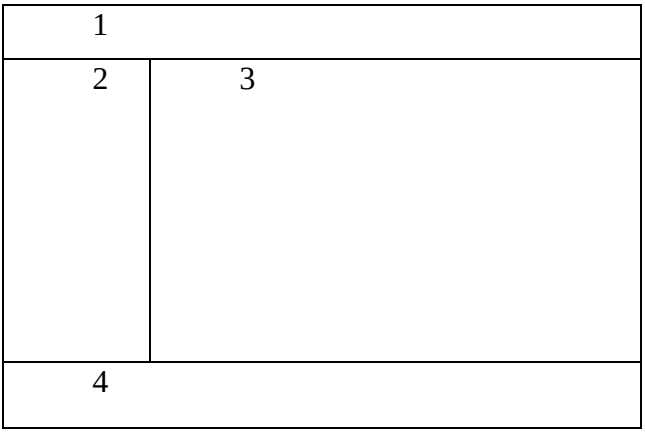

Рис. 26. Типовая схема веб-страницы с одним боковым меню

Верстка будет состоять из следующих этапов.

1. **Добавление тегов с атрибутами ID** для каждого раздела страницы. Допустим будут указаны следующие ID:

banner – баннер;

sidebar – левое меню;

main – основное содержание;

footer – подножие.

2. Добавление материала к каждому тегу . Проверку работоспособности верстки лучше проводить на наглядном примере. Желательно, чтобы основное содержимое по высоте было больше высоты окна браузера.

3. Создание стилей для всех фиксированных элементов. Для баннера, левого меню и подножия следует создать соответствующие стили:

```
#banner {
        position: fixed;
left: 0;
top: 0;
width: 100%;
}
#sidebar {
        position: fixed;
        left: 0;
        top: 110px;
        width: 175px;
}
#footer {
        position: fixed;
        bottom: 0;
        left: 0;
        width: 100%;
}
```

4. Создать стиль для прокручиваемой области содержимого. Главная – задача избежать того, чтобы содержимое тега основного содержания не появлялось под фиксированными элементами. Для этого нужно добавить соответствующие поля с нужных сторон.

```
#main {
        margin-left: 175px;
        margin-top: 100px;
}
```

5. Отрегулировать разметку для Internet Explorer 6 и ниже. Так как IE 6 не понимает фиксированное позиционирование, то придется указать ему, чтобы он воспринимал фиксированные элементы как абсолютные. Это можно сделать, добавив стили, работающие только в IE 6 и ниже:

```
*html #banner {
        position: absolute;
}
*html #sidebar {
        position: absolute;
}
```

footer в данном случае так позиционировать не стоит – он будет «ездить» по окну браузера при прокрутке. А так он будет находиться в самом низу страницы.

3. НОВЫЕ ТЕХНОЛОГИИ WEB-ДИЗАЙНА

3.1. Общие сведения о HTML 5

Считается, что разработка стандарта началась в 2004 году Рабочей группой по разработке Гипертекстовых Прикладных Технологий в Веб (Web Hypertext Application Technology Working Group), когда W3C сосредоточилась на разработке XHTML 2.0. В 2009 году от разработки последнего решено было отказаться и обе группы объединились в работе над будущим стандартом HTML.

Следует заметить, что стандарт HTML 5 до сих пор находится в разработке и, соответственно, существует в виде черновика. Но большинство современных браузеров поддерживают те нововведения, которые внесены этим стандартом в разметку web-документов. Принятие полной спецификации HTML ожидается в 2014 году.

К новым особенностям HTML, любопытных в плане Web-дизайна можно отнести:

– добавление тегов, облегчающих отображение мультимедийной и графической информации;

– добавление тегов, отображающих семантическое содержание документа;

– добавление новых видов полей для формы.

HTML 5 потенциально дает много возможностей web-дизайнеру. Некоторые даже утверждают, что он заменит JavaScript и CSS. На самом деле это не так – за HTML остается структурирование содержимого веб-страницы, CSS управляет отображением элементов на странице, а JavaScript «оживляет» элементы web-страницы, добавляя им анимацию и делая их интерактивными.

Основные различия между HTML 4.01 и HTML 5

Каким образом можно увидеть, что мы имеем дело со страницей, созданной с помощью HTML 5? Самый верный способ – обратиться к элементу <!DOCTYPE>.

Элемент <!DOCTYPE> предназначен для указания типа текущего документа – DTD (document type definition, описание типа документа). Чтобы браузер «не путался» и понимал, согласно какому стандарту отображать веб-страницу и необходимо в первой строке кода задавать <!DOCTYPE>.

В старых версиях <!DOCTYPE> представлял собой весьма громоздкую конструкцию, состоящую из множества частей. В HTML 5 он пишется достаточно просто:

```
<!DOCTYPE html>
```

И все. Если у документа сверху стоит такой <!DOCTYPE>, то он интерпретируется браузером как документ HTML 5.

Прежде, чем перейти к описанию новых тегов, стоит огласить список старых тегов, использование которых осуждается спецификацией HTML 5 (Таблица валидности тегов в разных стандартах HTML приведена в прил. 5).

Теги, не вошедшие в спецификацию HTML 5 (возможно, в окончательной спецификации будет другой список):

<acronym>	<isindex>
<applet>	<marquee>

<basefont>
<bgsound>
<big>
<blink>

<comment>
<dir>
<font>
<frame>
<frameset>
<nobr>
<noembed>
<noframes>
<plaintext>
<s>
<strike>
<tt>
<u>
<xmp>

3.2. Теги для отображения мультимедийных объектов

Тег <audio>

Тег <audio> Добавляет, воспроизводит и управляет настройками аудиозаписи на веб-странице. Путь к файлу задается через атрибут src или вложенный тег <source>. Внутри контейнера <audio> можно написать текст, который будет выводиться в браузерах, не работающих с этим тегом. Данный тег будет работать в браузерах IE (с 9-й версии), Chrome 10.0 и выше, Opera 10.50 и выше, Firefox 4.0 и выше, Safari 3.1 и выше.

Имеет синтаксис:

```
<audio src="URL"></audio>
```

или

```
<audio>
        <source src="URL">
</audio>
```

Список кодеков и поддерживаемых форматов несколько ограничен и зависит от вида и версии браузера (см. прил. 6).

Тег <audio> имеет ряд важных атрибутов.

autoplay – при добавлении этого атрибута к тегу звук начинает проигрываться сразу при загрузке страницы. Можно добавить этот атрибут просто без значения, а можно написать autoplay=”autoplay”.

controls – добавляет панель управления к аудиотреку. Вид панели и ее содержимое зависит от браузера и может в себя включать: кнопку воспроизведения, паузы, перемотки, ползунок для изменения уровня громкости и др. Способ добавления – аналогичен autoplay.

loop – зацикливает воспроизведение аудио, чтобы оно повторялось с начала после завершения. Способ добавления – аналогичен autoplay.

preload – используется для загрузки аудиофайла вместе с загрузкой веб-страницы. Этот атрибут игнорируется, если установлен autoplay. Имеет 3 значения: none – не загружать аудиофайл (значение по умолчанию); metadata – загружать только служебную информацию (продолжительность звучания и др.); auto – загружать файл при загрузке страницы.

Пример использования:

```
<audio autoplay preloadp="metadata">
  <source src="audio/music.ogg" type='audio/ogg;
            codecs=vorbis'>
  <source src="audio/music.mp3" type="audio/mpeg">
            Тег audio не поддерживается вашим браузером.
  <a href="audio/music.mp3">
            Скачайте музыку
  </a>
</audio>
```

Результат использования тега <audio> показан на рис. 27.

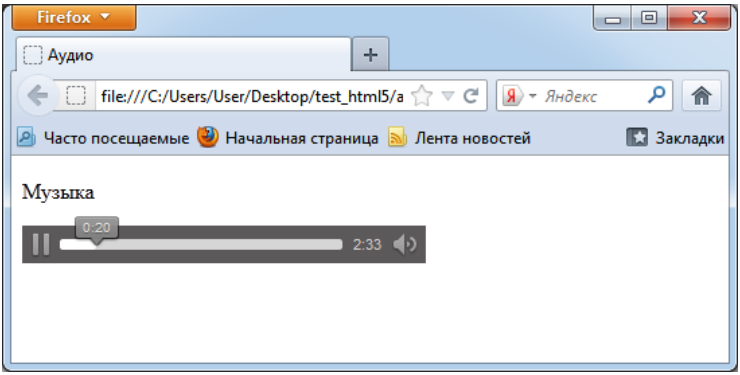

Рис. 27. Результат использования тега <audio>
в браузере Firefox

Для универсального воспроизведения во всех браузерах аудио кодируют с помощью разных кодеков и добавляют файлы одновременно через тег <source>. Браузер просмотрит все теги <source> выберет из них файл, который он сможет воспроизводить.

Прежде, чем перейти к тегу для воспроизведения видео обратим внимание, на упомянутый выше тег <source>.

Тег <source>

Данный тег предназначен для добавления ссылки на файл. Обычно используется в связке с тегами для отображения аудио и видео.

Синтаксис:

```
<audio>
   <source src="URL">
</audio>

<video>
   <source src="URL">
</video>
```

Имеет ряд следующих атрибутов.

media – тип устройства, на котором будет воспроизводиться данный файл. Имеет следующие значения:

all – все устройства;

braille – устройства, основанные на системе Брайля и предназначенные для слепых людей;

handheld – наладонники, смартфоны, устройства с малой шириной экрана;

print – печатающее устройство вроде принтера;

screen – экран монитора;

speech – речевые синтезаторы, а также программы для воспроизведения текста вслух. Сюда же входят речевые браузеры;

projection – проектор;

tty – телетайпы, терминалы, портативные устройства с ограниченными возможностями экрана;

tv – телевизор.

src – адрес файла.

type – MIME-тип медийного источника. Существует большое количество MIME-типов, определяющих вид добавленного на страницу файла.

Например:

type="audio/mpeg" – файл mp3;
type=”video/mpeg” – файл mpeg;
type=”text/css” – файл CSS и т.д.

Тег <video>

Добавляет, воспроизводит и управляет настройками видеоролика на веб-странице. Путь к файлу задается через атрибут src или вложенный тег <source>. Список поддерживаемых браузерами видеокодеков ограничен (См. прил. 6).

Синтаксис:

```
<video>
<source src="URL">
</video>
```

Тег имеет следующие атрибуты:

autoplay, controls, loop, preload, src – аналогичны тегу <audio>;

height и width – задают высоту и ширину области для просмотра видеоролика;

poster – указывает адрес картинки, которая будет отображаться пока видео недоступно или не воспроизводится.

Пример добавления видео на страницу:

```
<video width="400" height="300" controls="controls" poster="video/myvid.jpg">

<source src="video/myvid.ogv" type='video/ogg;
codecs="theora, vorbis"'>

<source src="video/myvid.mp4" type='video/mp4;
codecs="avc1.42E01E, mp4a.40.2"'>

<source src="video/myvid.webm" type='video/webm;
codecs="vp8, vorbis"'>

Тег video не поддерживается вашим браузером.

<a href="video/myvid.mp4">
  Скачайте видео
</a>
</video>
```

Пример использования тега <video> показан на рис. 28.

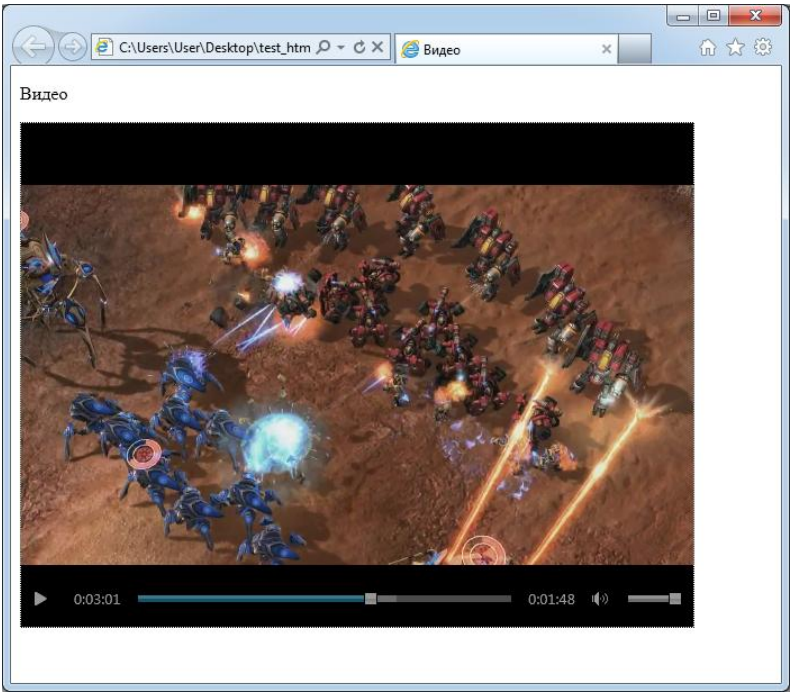

Рис. 28. Результат использования тега <video>
в браузере Internet Explorer

Тег <canvas>

Данный тег предназначен для создания области, в которой при помощи JavaScript можно рисовать разные объекты, выводить изображения, трансформировать их и менять свойства.

Синтаксис:

<canvas id="идентификатор">
</canvas>

Обратившись к ID элемента можно применить к нему сценарий JavaScript и добавить в него рисунки и анимацию.

Имеет два атрибута – width и height – устанавливающие высоту и ширину области для рисования.

3.3. Теги для обозначения блоков страницы

При блочной верстке страниц, содержимое страницы делилось на отдельные части (блоки) с помощью тегов . Каждый тег помечался определенным идентификатором, который вместе со стилем CSS определял, что это за блок и где он будет размещаться. Например, блок заголовка страницы помечался тегом с ID “header”, блок подножия – тегом с ID “footer”.

В HTML 5 разработчики решили пойти навстречу веб-дизайнеру и добавили несколько аналогов тегу , имена которых призваны отразить предназначение (смысл или семантику) блока. Действительно, более удобным и наглядным будет вот такой код:

```
<header>
  Заголовок сайта
</header>
```

чем

```
<div id=”header”>
  Заголовок сайта
</div>
```

Далее веб-дизайнер может создать стиль для тега <header> и отформатировать заголовок сайта, как он считает нужным.

Для удобства верстки в HTML 5 добавлены следующие теги:

<article> – задает содержание сайта вроде новости, статьи, записи блога, форума или др. Атрибутов нет.

<aside> – определяет блок сбоку от контента для размещения рубрик, ссылок на архив, меток и другой информации. Такой блок, как правило, называется «сайдбар» или «боковая панель». Применимы универсальные атрибуты.

<details> – используется для хранения информации, которую можно скрыть или показать по требованию пользователя. По умолчанию содержимое тега не отображается, для изменения статуса применяется атрибут open (например, <details open=«open»>Текст</details>).

<footer> – задаёт «подвал» сайта или раздела, в нём может располагается имя автора, дата документа, контактная и правовая информация. Атрибутов нет.

<header> – задает «шапку» сайта или раздела, в которой обычно располагается заголовок. Атрибутов нет.

<hgroup> – используется для группирования заголовков веб-страницы или раздела. Внутри располагаются теги заголовков от <h1> до <h6>. Атрибутов нет.

<menu> – предназначен для отображения списка пунктов меню. Аналогично тегам <ol> и <ul> внутри контейнера <menu> список формируется с помощью тегов <li>. Атрибут label устанавливает видимую метку для меню, атрибут type задает тип меню (значения: context – контекстное меню, toolbar – меню в виде панели инструментов, list – меню в виде списка).

<nav> – задает навигацию по сайту. Если на странице несколько блоков ссылок, то в <nav> обычно помещают приоритетные ссылки. Также допустимо использовать несколько тегов <nav> в документе. Запрещается вкладывать <nav> внутрь <address>. Атрибутов нет.

<section> – задает раздел документа, который может включать в себя заголовки, шапку, подвал и текст. Допускается вкладывать один тег <section> внутрь другого.

<summary> – указывает заголовок для тега <details>, по которому можно щелкать для разворачивания/сворачивания информации. Тег <summary> должен идти первым внутри <details>. Синтаксис:

```
<details>
<summary>Текст</summary>
</details>
```

В дополнение можно упомянуть следующие теги полезные для использования на странице:

<mark> – помечает текст как выделенный. Такой текст ничем не отличается от обычного, но его вид может быть изменен с помощью стилей. В браузере Chrome и Firefox фоновый цвет текста внутри <mark> выделяется желтым цветом.

<figure> – используется для группирования любых элементов, например, изображений и подписей к ним.

<figcaption> – содержит описание для тега <figure>. Тег <figcaption> должен быть первым или последним элементом в группе.

Пример – создание изображения с подписью:

```
<figure>
   <div><img src="LZ2.jpg" alt="Led Zeppelin II"></div>
   <figcaption>Led Zeppelin II <br> 1969 </figcaption>
</figure>
```

Теперь создадим стили для тегов <figure> и <figcaption>:

```
figure {
   background: #5f6a72; /* Цвет фона */
   padding: 5px; /* Поля вокруг */
   display: block; /* Блочный элемент */
   width: 350px; /* Ширина */
   float: left; /* Блоки выстраиваются по горизонтали */
   margin: 0 10px 10px 0; /* Отступы */
   text-align: center; /* Выравнивание по центру */
  }
figcaption {
   color: #fff; /* Цвет текста */
  }
```

Получившийся результат показан на рис. 29.

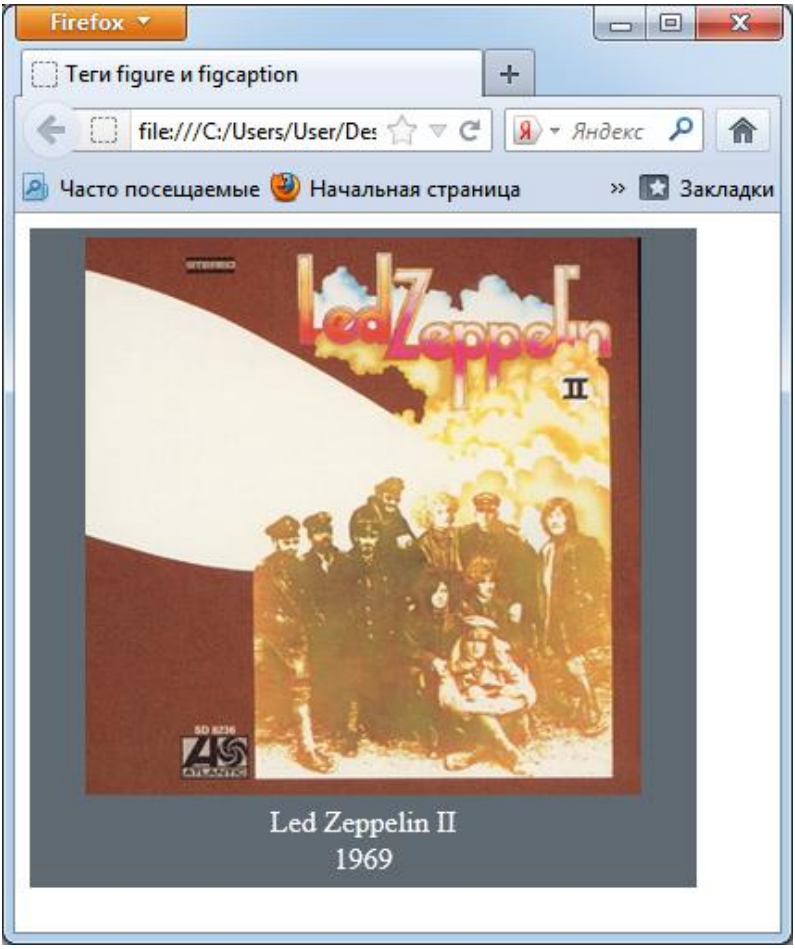

Рис. 29. Результат использования тегов <figure> и <figcaption>

3.4. Новые элементы веб-форм

Поскольку формы являются одним из наиболее важных элементов веб-страниц в HTML 5 добавлено много новых возможностей для улучшения веб-форм. К сожалению, многие новшества, связанные с формами не поддерживаются, боль-

шинством браузеров – фактически только Opera способна отобразить большую часть новых тегов и атрибутов.

В тег <input> добавлены новые атрибуты. Укажем некоторые из них.

autocompete – этот атрибут помогает заполнять поля форм текстом, который был введён в них ранее. Значения сохраняет и подставляет браузер, при этом автозаполнение по соображениям безопасности может отключаться пользователем в настройках и не может в таком случае управляться атрибутом autocomplete.

autofocus – указывает, что данный элемент должен быть активным сразу после загрузки страницы.

form – Связывает поле с формой по её идентификатору. Такая связь необходима в случае, когда поле располагается за пределами <form>, например, при создании её программно или по соображениям дизайна.

formnovalidate – отменяет встроенную проверку данных введенных пользователем в форме на корректность перед отправкой формы. Такая проверка делается браузером автоматически для полей <input type="email">, <input type="url">, а также при наличии атрибута pattern или required у тега <input>.

list – указывает на список вариантов, созданный с помощью тега <datalist>, которые можно выбирать при наборе текста. Изначально этот список скрыт и становится доступным при получении полем фокуса.

Например:

```
<input type="text" list="mark">

<datalist id="mark">
  <option value="3">Удовлетворительно</option>
  <option value="4">Хорошо</option>
  <option value="5">Отлично</option>
</datalist>
```

min и **max** – устанавливают минимальное и максимальное значение для ввода числа или даты. Значение – целое число или дата в формате ГГГГ-ММ-ДД.

multiple – позволяет указывать одновременно несколько файлов в поле для загрузки файлов, а также несколько адресов электронной почты. При использовании двух и более почтовых адресов они должны перечисляться через запятую.

pattern – указывает регулярное выражение, согласно которому требуется вводить и проверять данные в поле формы. Если присутствует атрибут pattern, то форма не будет отправляться, пока поле не будет заполнено правильно. Например [0-9]{5,10} – ввести от 5 до 10 цифр, \D [^0-9] – любой символ кроме цифры и т.д.

placeholder – выводит текст внутри поля формы, который исчезает при получении фокуса. Значение – текстовая строка.

required – указывает, что данное поле обязательно должно быть заполнено перед отправлением формы.

step – устанавливает шаг изменения числа для ползунков и полей ввода чисел.

Кратко рассмотрим значения атрибута **type** тега <input>, позволяющие создавать новые элементы веб-форм.

color – поле для выбора цвета.

date – поле для выбора календарной даты.

datetime – указание даты и времени.

datetime-local – указание местной даты и времени.

email – поле для адресов электронной почты.

number – ввод чисел.

range – ползунок для выбора чисел в указанном диапазоне

search – поле для поиска.

tel – поле для телефонных номеров.

time – поле для времени.

url – поле для веб-адресов.

month – поле для выбора месяца.

week – выбор недели.

Вид страницы, иллюстрирующей новые элементы формы, показан на рис. 30.

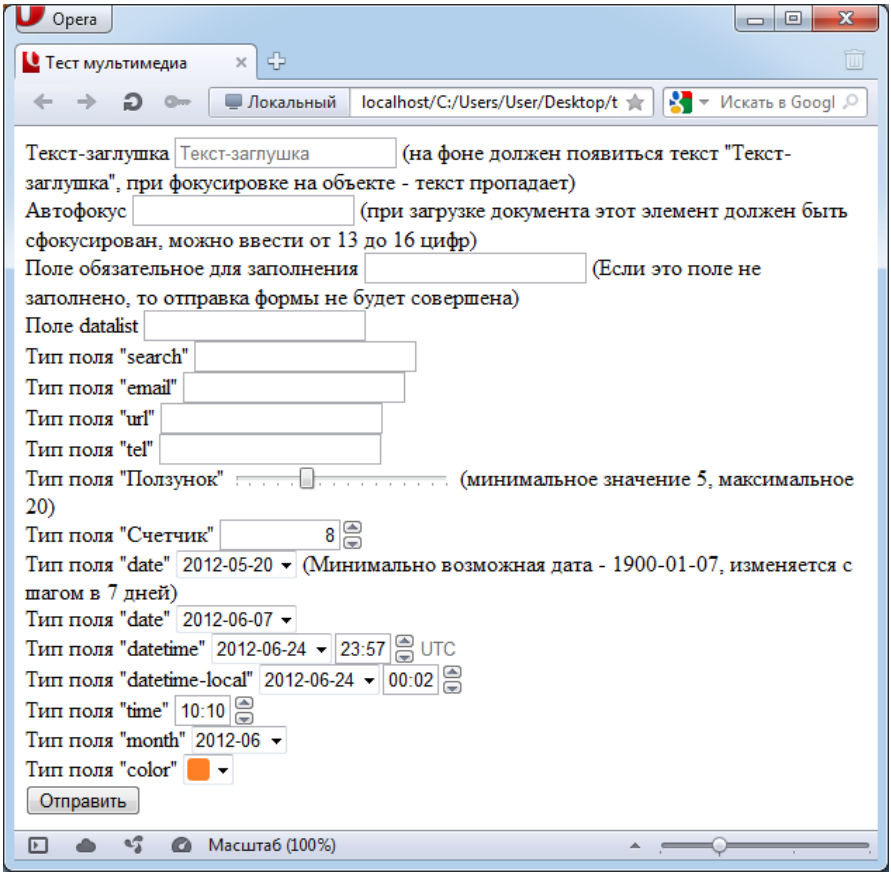

Рис. 30. Примеры новых элементов формы в браузере Opera

3.5. Общие сведения о CSS 3

С 1996 года CSS значительно «повзрослел» и стал основным инструментом веб-дизайнеров. Благодаря их стремлению делать все более красивые веб-сайты последующие версии CSS приобретали больше свойств и, как следствие, большую слож-

ность. Версию CSS 3 W3C разделил на независимые модули, каждый из которых описывает отдельные наборы свойств. Назначение некоторых модулей известно, например CSS Transitions (анимация или трансформация переходов между двумя состояниями – подсветка текста при наведении на него указателя мыши) или CSS Web Fonts (инструкции для изменения различных свойств шрифта). Функции других пока точно не определены, например CSS Math или CSS Namespace.

Основная цель подобного модульного деления состоит в том, чтобы дать возможность CSS развиваться постепенно, не заставляя всех ждать, когда же наконец будет готов один гигантский набор инструкций. Каждый модуль развивается отдельно, а производители браузеров могут использовать отдельные инструкции модулей сразу, как только те будут готовы.

Хотя CSS 3 еще не стал общепринятым стандартом, многие наиболее привлекательные свойства CSS 3 нашли свое применение в браузерах.

Некоторые новые возможности CSS 3:

– введение новых видов селекторов для более гибкой выборки элементов на странице;

– новые виды оформления текста (тень, разбиение текста на колонки и т.д.);

– настройка прозрачности элементов страницы;

– создание скругленных границ элементов;

– новые свойства, специально предназначенные для элементов форм.

3.6. Селекторы в CSS 3

Селекторы дочерних элементов

CSS 3 предоставляет новые возможности по форматированию тегов, которые являются дочерними по отношению к остальным.

Рассмотрим основные свойства, связанные с дочерними элементами.

:first-child – данный псевдокласс применяет стилевое оформление к первому дочернему элементу своего родителя. На самом деле, :first-child появился в стандарте CSS 2, но остальные подобные селекторы добавились в CSS 3, поэтому уместно разместить его описание здесь.

:last-child –псевдокласс задает стилевое оформление последнего элемента своего родителя.

В качестве примера, можно использовать данные селекторы для оформления первого и последнего элемента маркированного списка:

```
ul :first-child {text-decoration: underline}
ul :last-child { text-decoration: line-through}
```

Результат использования данных стилей показан на рис. 31.

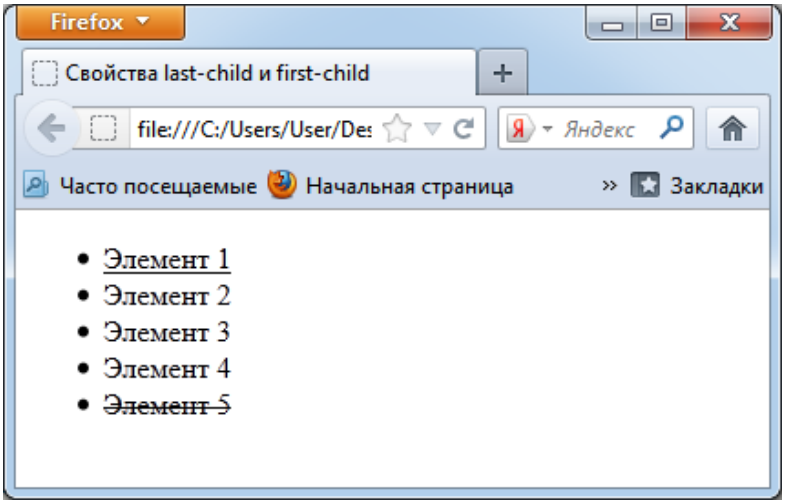

Рис. 31. Результат применения селекторов :first-child и last-child к маркированному списку

Следует обратить внимание на пробел, расположенный между ul и :first-child. Эта конструкция означает «найти первый

дочерний элемент тега ». Конструкция вида ul:first-child без пробела имеет совершенно другой смысл: «найти все теги <ul>, которые являются первыми дочерними для какого-то другого тега».

:nth-child() – используется для добавления стиля к элементам на основе нумерации в дереве элементов. Синтаксис данного свойства выглядит следующим образом:

элемент :nth-child(odd | even | <число> | <выражение>) {...}

Как видно в скобках после название свойства должно быть указано число или выражение для определения стилизуемых элементов. Возможны следующие значения:

odd – все нечетные номера элементов;

even – все четные номера элементов;

число – порядковый номер дочернего элемента относительно своего родителя. Нумерация начинается с 1, это будет первый элемент в списке.

выражение – задается в виде an+b, где a и b целые числа, а n – счетчик, который автоматически принимает значение 0, 1, 2...

Если a равно нулю, то оно не пишется и запись сокращается до b. Если b равно нулю, то оно также не указывается и выражение записывается в форме an. a и b могут быть отрицательными числами, в этом случае знак плюс меняется на минус, например: 5n-1.

За счет использования отрицательных значений a и b некоторые результаты могут также получиться отрицательными или равными нулю. Однако на элементы оказывают влияние только положительные значения из-за того, что нумерация элементов начинается с 1.

Примеры выражений:

2n – 2, 4, 6, 8, 10… – все четные элементы, аналог значения even.

2n+1 – 1, 3, 5, 7, 9… – все нечетные элементы, аналог значения odd.

3n+2 – 5, 8, 11, 14 и т.д.
-n+3 – 3, 2, 1.

Применим данный селектор к элементам маркированного списка, воспользовавшись значениями odd и even:

ul :nth-child(odd) {text-decoration: underline}
ul :nth-child(even) { text-decoration: line-through}

Результат использования данных стилей показан на рис. 32.

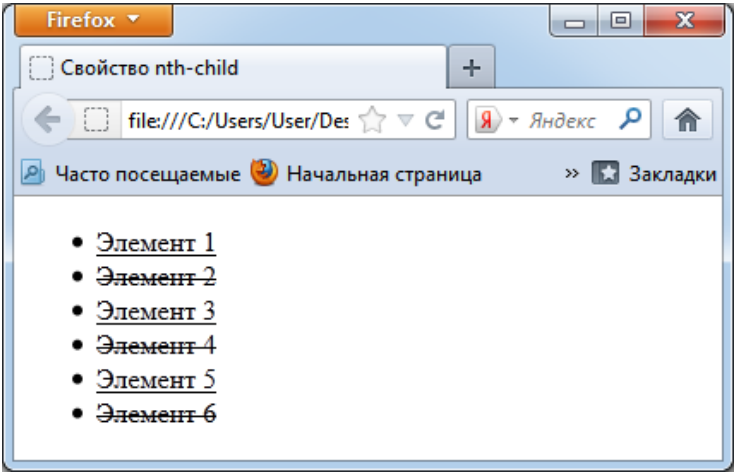

Рис. 32. Результат применения селекторов :nth-child

:nth-last-child – используется для добавления стиля к элементам на основе нумерации в дереве элементов. В отличие от псевдокласса :nth-child отсчет ведется не от первого элемента, а от последнего. В остальном принцип использования данного свойства тот же самый.

:only-child – применяется к дочернему элементу, только если он единственный у родителя. Аналогичен использованию :first-child:last-child или :nth-child(1):nth-last-child(1).

Селекторы типов

В CSS 3 введены новые селекторы, которые работают так же, как и селекторы дочерних элементов за тем лишь исключением, что позволяют указать на конкретный тип выбираемого тега. Предположим, что необходимо оформить первый абзац внутри боковой панели особым образом, однако на некоторых страницах первым абзацем является заголовок, начинающийся с тега <h2>, в то время как на других страницах это тег <p>. В данном случае не получиться использовать :first-child для выбора абзаца, поскольку в некоторых случаях он будет являться вторым после тега <h2>. Тем не менее тег <p> всегда является первым абзацем, даже если перед ним находятся другие теги. Поэтому можно будет воспользоваться селектором типа :first-of-type.

Рассмотрим данные селектор и подобные ему селекторы типов.

:first-of-type – работает точно так же, как и :first-child, но позволяет указать на конкретный тип дочернего тега. Например, есть боковая панель, в которой id="sidebar". Чтобы оформить первый абзац в этой панели, можно воспользоваться следующим стилем:

```
#sidebar p:first-of-type
```

Следует обратить внимание на то, что именно тег <p> является первым нужным типом.

:last-of-type – работает точно так же, как и :last-child, но указывает на последний тег конкретного типа.

:nth-of-type – работает точно так же, как и :nth-child, но указывает на последний тег конкретного типа. Данный селектор удобно использовать, например, для изображений находящихся в объемном тексте. Чтобы разместить изображения в тексте в шахматном порядке можно добавить следующие стили:

```
img:nth-of-type(odd) {float: left}
img:nth-of-type(even) {float: right}
```

:nth-last-of-type – используется для добавления стиля к элементам указанного типа на основе нумерации в дереве элементов. В отличие от псевдокласса :nth-of-type отсчет ведется не от первого элемента, а от последнего.

:only-of-type – применяется к дочернему элементу указанного типа, только если он единственный у родителя. Аналогичен использованию :first-of-type:last-of-type или :nth-of-type(1):nth-last-of-type(1).

3.7. Форматирование текста в CSS 3

Создание тени для текста

В CSS 3 включено свойство, которое позволяет добавить тень тексту, чтобы придать ему глубины (например, заголовкам, спискам или отдельным абзацам). Это свойство называется **text-shadow**.

Синтаксис:

text-shadow: none | тень [,тень]

где тень:

<цвет> <сдвиг по x> <сдвиг по y> <радиус размытия>

Данное свойство может принимать следующие значения.

none – отменяет добавление тени.

<цвет> – цвет тени в любом доступном CSS формате. По умолчанию цвет тени совпадает с цветом текста. Необязательный параметр.

<сдвиг по x> – смещение тени по горизонтали относительно текста. Положительное значение этого параметра задает сдвиг тени вправо, отрицательное – влево. Обязательный параметр.

<сдвиг по y> – смещение тени по вертикали относительно текста. Также допустимо использовать отрицательное значение, которое поднимает тень выше текста. Обязательный параметр.

<радиус> – задает радиус размытия тени. Чем больше это значение, тем сильнее тень сглаживается, становится шире и светлее. Если этот параметр не задан, по умолчанию устанавли-

вается равным 0. Учтите, что алгоритм сглаживания в браузерах обычно разный, поэтому вид тени может несколько различаться в зависимости от заданных параметров сглаживания.

Допускается указывать несколько параметров тени, разделяя их между собой запятой. В CSS3 учитывается следующий порядок: первая тень в списке размещается на самом верху, последняя в списке – в самом низу.

В качестве примера зададим следующий стиль:

```
p {
        text-shadow: #999 -4px 4px 3px;
}
```

Первое значение задает цвет тени. Значение «-4px» можно прочесть как «поместить тень на 4 пикселя левее текста». Следующий параметр – «4px» – размещает тень на 4 пикселя ниже текста. Значение «3px» определяет смазанность тени. Нулевое значение (полное отсутствие смазанности) даст очень четкуб и резкую тень, в то время как большие значения сделают ее более размытой.

Можно добавить две тени сразу, например:

```
.sh1 {
        text-shadow: -4px 4px 3px #666, 1px -1px 2px #000;
}
```

Результат использования данных стилей показан на рис. 33. К верхней надписи применен первый стиль, к нижней - второй, с двойной тенью.

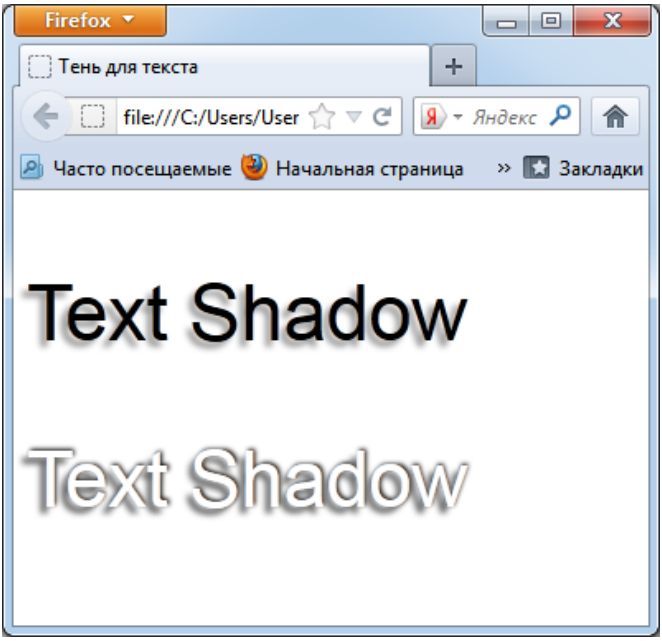

Рис. 33. Пример использования свойства text-shadow

Разбиение текста на колонки

В CSS 3 добавлен ряд свойств позволяющий разбить многострочный текст на две или большее количество колонок. Рассмотрим данные свойства.

column-count – Определяет количество колонок в тексте. Следует заметить, что данное свойство поддерживается только в браузере Internet Explorer версии 10 и выше, а также в браузере Opera, начиная с версии 11.1. Браузер Firefox использует нестандартное свойство **-moz-column-count**, Safari, Chrome и iOS поддерживают нестандартное свойство **-webkit-column-count.**

Чтобы разбиение текста на колонки сработало во всех браузерах можно использовать вот такой стиль:

```
.book {
-moz-column-count: 2; /* Для Firefox */
```

```
-webkit-column-count: 2; /* Для Safari и Chrome */
column-count: 2;
}
```

В качестве значения используется либо целое число, либо значение auto (число колонок вычисляется автоматически на основе других свойств: column-width, column-gap и т.д.)

column-width – задаёт оптимальную ширину колонки в многоколоночном тексте. Реальная ширина колонки может быть больше введённого значения, сокращение указанной ширины приводит к отмене нескольких колонок, текст при этом выстраивается в одну колонку. Браузер Firefox использует нестандартное свойство **-moz-column-width**, Safari, Chrome и iOS поддерживают нестандартное свойство **-webkit-column-width.** Значение задается в абсолютных величинах, например, в пикселях. Также можно использовать значение auto (ширина колонок вычисляется автоматически на основании других свойств).

column-gap – задаёт расстояние между колонками в многоколоночном тексте. Браузер Firefox использует нестандартное свойство **-moz-column-gap**, Safari, Chrome и iOS поддерживают нестандартное свойство **-webkit-column-gap.** Значение задается в абсолютных величинах, например, в пикселях. Также можно использовать значение normal (межколоночный интервал задается браузером).

column-rule – в многоколоночном тексте отрисовывает линию между колонками. Синтаксис:

column-rule: <border-width> || <border-style> || <цвет>

Значения:

border-width определяет толщину линии. Для управления видом линии предоставляется несколько значений border-style (значения аналогичны видам границ элемента: solid – сплошная линия, double – сплошная двойная линия, dashed – штриховая линия, dotted – линия из точек). <цвет> устанавливает цвет ли-

нии, значение может быть в любом допустимом для CSS формате.

columns – универсальное свойство, которое позволяет одновременно задать ширину и количество колонок многоколоночного текста. Браузер Firefox использует нестандартное свойство **-moz-columns**, Safari, Chrome и iOS поддерживают нестандартное свойство **-webkit-columns.**Синтаксис:

columns: [column-width] || [column-count]

В качестве значений выступает комбинация свойств column-width и column-count. Порядок значения не имеет.

Пример использования свойства column:

columns: 200px auto;

Ширина колонок 200 пикселей, количество задаётся браузером.

columns: 3 12em;

Три колонки минимальной ширины 12em каждая.

Результат разбиения текста на три колонки шириной 12em показан на рис. 34.

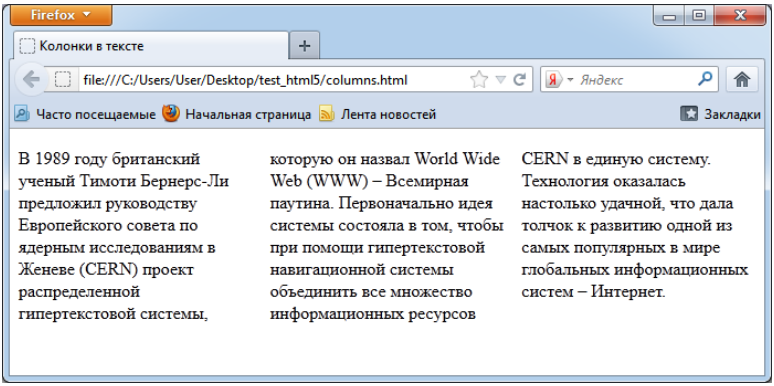

Рис. 34. Текст, разбитый на колонки с помощью CSS

Подключение собственного шрифта

Благодаря директиве **@font-face** перед веб-дизайнерами открываются новые возможности. Используя, данное нововведение, можно разместить на сервере файл шрифта в формате

Open Type (.otf) или True Type (.ttf), а затем воспользоваться директивой @font-face, чтобы браузер посетителя скачал указанный шрифт на время просмотра сайта.

Пример типичного использования директивы @font-face:

```
@font-face {
        font-family: Lavoisier;
        src: url(‘lavoisier.otf’);
}
```

Свойство font-family определяет имя, которое будет использоваться в стилях. В данном случае это Lavoisier. Оно необязательно должно совпадать с настоящим именем шрифта, который будет использоваться. Свойство src используется для указания URL-адреса файла шрифта.

Лишь после определения имени шрифта его возможно использовать в своем файле стилей. Например:

```
h1 {
        font-family: Lavoisier, Arial, Helvetica, sans-serif;
        color: #FF993E;
        font-size: 48px;
}
```

В данном случае рекомендуется в свойстве font-family указать дополнительные шрифты, которые браузер пользователя применит к элементу, если не поймет директиву @font-face.

3.8. Работа с прозрачностью и фоном элемента в CSS 3

Прозрачность

В CSS 3 добавлено специальное свойство **opacity**, которое позволяет настроить степень «прозрачности» видимого элемента на странице.

Данное свойство может принимать значение в диапазоне от 0 (полностью прозрачный) до 1 (непрозрачный). Например,

если необходимо сделать какое-либо изображение полупрозрачным, то можно использовать следующий стиль:

```
.ghost {
        opacity: 0.5;
}
```

Firefox до версии 3.5 поддерживает нестандартное свойство **–moz-opacity**.

Пример использования свойства opacity показано на рис. 35. Слева приведено непрозрачное изображение, справа - изображение с прозрачностью 0.5.

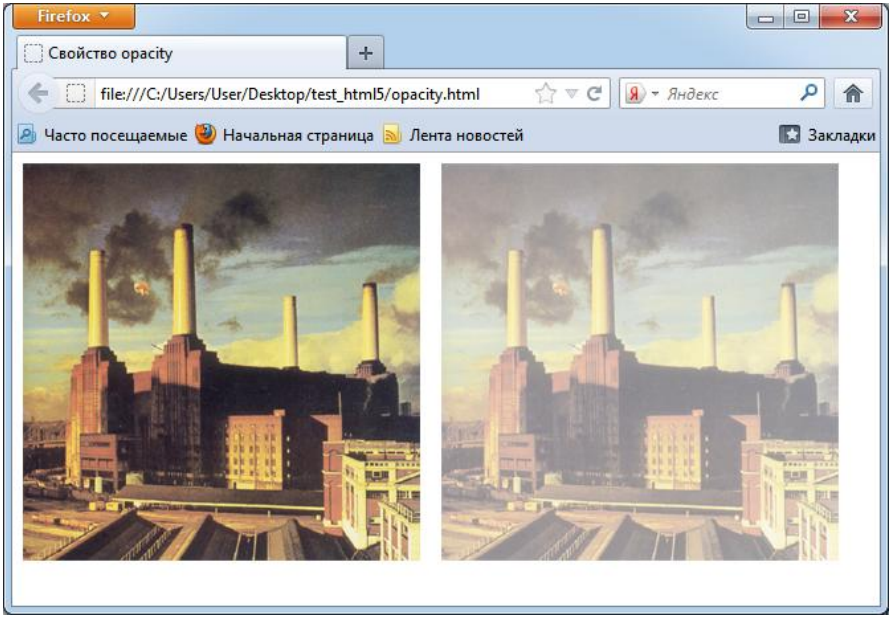

Рис. 35. Пример использования свойства opacity

Можно применять данное свойство вместе с псевдоклассом :hover, чтобы реализовать эффект выделение ссылок или изображений при наведении на них курсора мыши.

Добавление тени элемента

Свойство **box-shadow** добавляет тень к элементу. Допускается использовать несколько теней, указывая их параметры через запятую, при наложении теней первая тень в списке будет выше, последняя ниже. Добавление тени увеличивает ширину элемента, поэтому возможно появление горизонтальной полосы прокрутки в браузере.

Синтаксис:

box-shadow: none | <тень> [,<тень>]

где <тень>:

inset <сдвиг по x> <сдвиг по y> <радиус размытия> <растяжение> <цвет>

Для настройки вида тени используются следующие значения.

none – отменяет добавление тени.

inset – тень выводится внутри элемента. Необязательный параметр.

<сдвиг по x> – смещение тени по горизонтали относительно элемента. Положительное значение этого параметра задает сдвиг тени вправо, отрицательное – влево. Обязательный параметр.

<сдвиг по y> – смещение тени по вертикали относительно элемента. Положительное значение задает сдвиг тени вниз, отрицательное – вверх. Обязательный параметр.

<размытие> – задает радиус размытия тени. Чем больше это значение, тем сильнее тень сглаживается, становится шире и светлее. Если этот параметр не задан, по умолчанию устанавливается равным 0, тень при этом будет четкой, а не размытой.

<растяжение> – положительное значение растягивает тень, отрицательное, наоборот, ее сжимает. Если этот параметр

не задан, по умолчанию устанавливается 0, при этом тень будет того же размера, что и элемент.

<цвет> – цвет тени в любом доступном CSS формате, по умолчанию тень черная. Необязательный параметр.

Допускается указывать несколько теней, разделяя их параметры между собой запятой. Учитывается следующий порядок: первая тень в списке размещается на самом верху, последняя в списке – в самом низу.

Firefox до версии 4.0 поддерживает нестандартное свойство **–moz-box-shadow**. Safari до версии 5.1, Chrome до версии 10.0 и iOS поддерживают свойство **-webkit-box-shadow**.

Пример использования:

```
.shadow {
  background: #fc0; /* Цвет фона */
  -moz-box-shadow: 0 0 10px #999; /*Для Firefox*/
  -webkit-box-shadow: 0 0 10px #999; /* Для Safari и Chrome
*/
  box-shadow: 0 0 10px #999; /* Параметры тени */
  padding: 10px;
}
```

Результат показан на рис. 36.

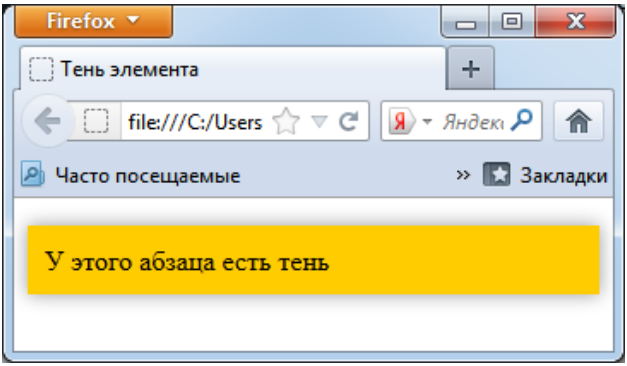

Рис. 36. Пример создания тени элемента средствами CSS

Работа с фоновым изображением

Рассмотрим несколько новых свойств для работы с фоновыми изображениями.

background-size – масштабирует фоновое изображение согласно заданным размерам.

Синтаксис:

background-size: [<значение> | <проценты> | auto]{1,2} | cover | contain

Может принимать следующие значения.

<значение> – задает размер в любых доступных для CSS единицах – пиксели (px), сантиметры (cm), em и др.

<проценты> – задает размер фоновой картинки в процентах от ширины или высоты элемента.

auto – если задано одновременно для ширины и высоты (auto auto), размеры фона остаются исходными; если только для одной стороны картинки (100px auto), то размер вычисляется автоматически исходя из пропорций картинки.

cover – масштабирует изображение с сохранением пропорций так, чтобы его ширина или высота равнялась ширине или высоте блока.

contain – масштабирует изображение с сохранением пропорций таким образом, чтобы картинка целиком поместилась внутрь блока.

Если установлено одно значение, оно задает ширину фона, второе значение принимается за auto. Пропорции картинки при этом сохраняются. Использование двух значений через пробел задает ширину и высоту фоновой картинки.

Safari до версии 4.1 и Chrome до версии 3.0 используют нестандартное свойство **-webkit-background-size**. Opera до версии 10.53 использует нестандартное свойство **-o-background-size**. Firefox до версии 4.0 использует нестандартное свойство **-moz-background-size**.

background-origin – определяет область позиционирования фонового рисунка. Это свойство не применяется, когда значение background-attachment задано как fixed.

Синтаксис:

background-origin: [padding-box | border-box | content-box] [, [padding-box | border-box | content-box]]

Может принимать следующие значения.

padding-box – фон позиционируется относительно края элемента с учетом толщины границы.

border-box – фон позиционируется относительно границы, при этом линия границы может перекрывать изображение.

content-box – фон позиционируется относительно содержимого элемента.

Значений может быть несколько (для каждого из множественных фоновых рисунков), при этом значения разделяются между собой запятой.

Если фон задан один, а значений background-original несколько, то браузеры покажут разное поведение. Firefox и Opera используют первое значение, Chrome и Safari создадут несколько фоновых рисунков. Chrome до версии 3.0 и Safari до версии 5.0 поддерживают нестандартное свойство **-webkit-background-origin**. Firefox до версии 4.0 поддерживает нестандартное свойство **-moz-background-origin**.

background-clip – определяет, как цвет фона или фоновая картинка должна выводиться под границами. Эффект заметен при прозрачных или пунктирных границах.

Синтаксис:

background-clip: [padding-box | border-box | content-box] [, [padding-box | border-box | content-box]]

Может принимать следующие значения.

padding-box – фон отображается внутри границ.

border-box – фон выводится под границами.

content-box – фон отображается только внутри контента.

Значений может быть несколько (для каждого из множественных фоновых рисунков), при этом значения разделяются между собой запятой.

Safari понимает только нестандартное свойство **-webkit-background-clip**, также с ним работает и Chrome. Opera не поддерживает значение content-box. Firefox до версии 4.0 поддерживает нестандартное свойство **-moz-background-clip**.

3.9. Стилизация границ элемента в CSS 3

Одной из интересных особенностей CSS 3 стала возможность создавать элементы с закругленными уголками, делая их более визуально красивыми, с помощью специального свойства **border-radius**.

Свойство border-radius устанавливает радиус скругления уголков рамки. Если рамка не задана, то скругление также происходит и с фоном.

Синтаксис:

border-radius: <радиус>{1,4} [/ <радиус>{1,4}]

Разрешается использовать одно, два, три или четыре значения, перечисляя их через пробел. Также допустимо писать два значения через слэш (/). В качестве значений указываются числа в любом допустимом для CSS формате. В случае применения процентов, отсчет ведется относительно ширины блока.

Можно ввести следующие количества значений для свойства border-radius:

1 – радиус указывается для всех четырех уголков;

2 – первое значение задает радиус верхнего левого и нижнего правого уголка, второе значение – верхнего правого и нижнего левого уголка;

3 – первое значение задает радиус для верхнего левого уголка, второе – одновременно для верхнего правого и нижнего левого, а третье – для нижнего правого уголка;

4 – по очереди устанавливает радиус для верхнего левого, верхнего правого, нижнего правого и нижнего левого уголка.

В случае задания двух параметров через слэш, то первый задает радиус по горизонтали, а второй по вертикали (эллиптические уголки). На рис. 37 показана разница между обычным скругленным уголком и эллиптическим уголком.

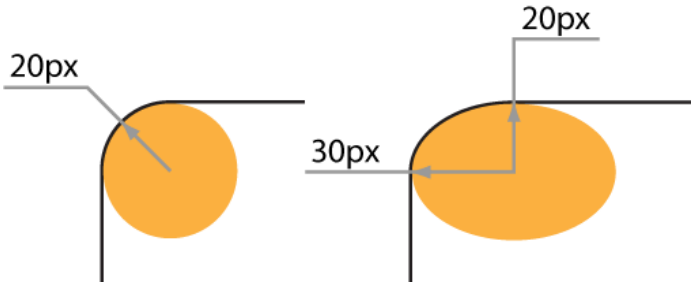

Рис. 37. Радиус скругления для создания разных типов уголков

Пример использования различных вариантов свойства border-radius приведен на рис. 38. Использовались следующие значения (сверху вниз):

```
border-radius: 50px 0 0 50px;
border-radius: 40px 10px;
border-radius: 13em/3em;
border-radius: 13em 0.5em/1em 0.5em;
border-radius: 8px;
```

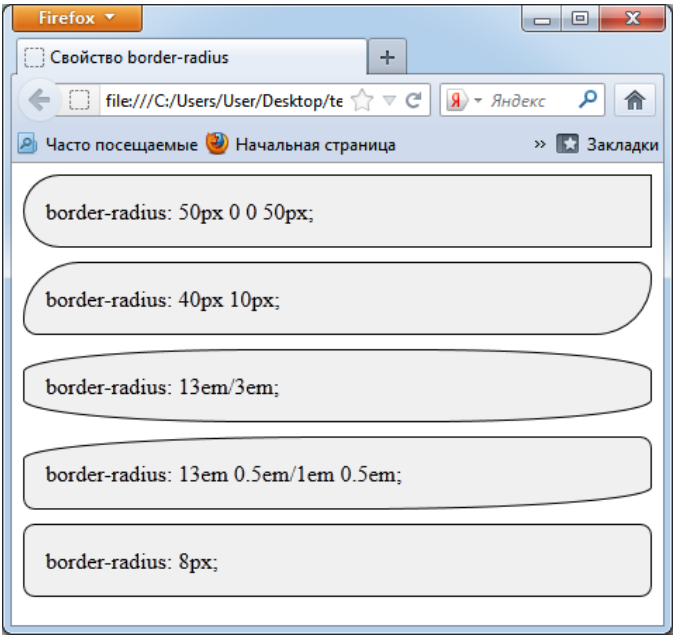

Рис. 38. Пример использования свойства border-radius

Chrome до версии 4.0, Safari до версии 5.0, iOS используют нестандартное свойство **-webkit-border-radius**.

Firefox до версии 4.0 использует нестандартное свойство **-moz-border-radius**.

Для управления отдельными границами используется следующая группа свойств:

border-bottom-left-radius
border-bottom-right-radius
border-top-left-radius
border-top-right-radius

Каждое свойство может иметь одно или два значения.

В качестве радиуса указывается любое допустимое в CSS значение (px, em и др.), а также проценты, в этом случае радиус скругления считается от ширины блока.

Необязательное второе значение предназначено для создания эллиптического уголка, первое значение при этом устанавливает радиус по горизонтали, а второе – радиус по вертикали.

Chrome до версии 4.0, Safari до версии 5.0, iOS используют нестандартные свойства вида **-webkit-border-bottom-left-radius** и т.д. Firefox до версии 4.0 использует свойства вида **-moz-border-radius-bottomleft** и т.д.

ЗАКЛЮЧЕНИЕ

Учебное пособие рассматривает основы разработки веб-сайтов с использованием языка гипертекстовой разметки HTML и каскадных таблиц стилей CSS.

Освоение базовых принципов разработки веб-сайтов, веб-страниц и их отдельных элементов позволит разрабатывать качественное статичное содержимое веб-сайтов различной тематической направленности и назначения.

Особое внимание в пособии уделено вопросам применения каскадных таблиц стилей CSS, изучение которых позволит освоить особенности оформления различных элементов веб-страниц для создания визуально эффектных и удобных для использования веб-сайтов.

Современные технологии Web-дизайна базируются на разработке структуры веб-страниц на основе блочной разметки (верстки). Особенности построения типичных структур страниц с использованием блочной верстки нашли свое отражение в учебном пособии.

Отдельная глава посвящена перспективным стандартам веб-разработки: HTML 5 и CSS 3. В пособии подробно рассмотрен ряд новых тегов и стилей, которые позволят оформлять элементы веб-страниц с учетом новых тенденций в Web-дизайне.

Таким образом, учебное пособие содержит подробное описание основных технологий HTML и CSS, применяемых при создании современных веб-сайтов.

ПРИЛОЖЕНИЕ 1

Специальные символы HTML

Описание символа	Обозначение	10-ный код	16-ный код	Вид
неразрывный пробел				
перевернутый восклицательный знак	¡	¡	¡	¡
цент	¢	¢	¢	¢
фунт стерлингов	£	£	£	£
знак денежной единицы	¤	¤	¤	¤
йена	¥	¥	¥	¥
вертикальная черта	¦	¦	¦	¦
параграф	§	§	§	§
диереза	¨	¨	¨	¨
знак авторского права	©	©	©	©
показатель женского рода	ª	ª	ª	ª
открывающая двойная угловая кавычка	«	«	«	«
знак отрицания	¬	¬	¬	¬
мягкий перенос				
охраняемый знак	®	®	®	®
надчеркивание	¯	¯	¯	¯
градус	°	°	°	°
плюс-минус	±	±	±	±
вторая степень	²	²	²	²
третья степень	³	³	³	³
острое ударение	´	´	´	´
знак микро	µ	µ	µ	µ
конец абзаца	¶	¶	¶	¶

средняя точка	·	·	·	·
седиль	¸	¸	¸	¸
единица в верхнем индексе	¹	¹	¹	¹
показатель мужского рода	º	º	º	º
закрывающая двойная угловая кавычка	»	»	»	»
одна четвертая	¼	¼	¼	¼
одна вторая	½	½	½	½
три четверти	¾	¾	¾	¾
перевернутый вопросительный знак	¿	¿	¿	¿
A с тупым ударением	À	À	À	À
A с острым ударением	Á	Á	Á	Á
A с циркумфлексом	Â	Â	Â	Â
A с тильдой	Ã	Ã	Ã	Ã
A с диерезой	Ä	Ä	Ä	Ä
A с кружком	Å	Å	Å	Å
лигатура AE	Æ	Æ	Æ	Æ
C с седилем	Ç	Ç	Ç	Ç
E с тупым ударением	È	È	È	È
E с острым ударением	É	É	É	É
E с циркумфлексом	Ê	Ê	Ê	Ê
E с диерезой	Ë	Ë	Ë	Ë
I с тупым ударением	Ì	Ì	Ì	Ì
I с острым ударением	Í	Í	Í	Í
I с циркумфлексом	Î	Î	Î	Î
I с диерезой	Ï	Ï	Ï	Ï
ETH	Ð	Ð	Ð	Ð
N с тильдой	Ñ	Ñ	Ñ	Ñ

O с тупым ударением	Ò	Ò	Ò	Ò
O с острым ударением	Ó	Ó	Ó	Ó
O с циркумфлексом	Ô	Ô	Ô	Ô
O с тильдой	Õ	Õ	Õ	Õ
O с диерезой	Ö	Ö	Ö	Ö
знак умножения	×	×	×	×
O перечеркнутое	Ø	Ø	Ø	Ø
U с тупым ударением	Ù	Ù	Ù	Ù
U с острым ударением	Ú	Ú	Ú	Ú
U с циркумфлексом	Û	Û	Û	Û
U с диерезой	Ü	Ü	Ü	Ü
Y с острым ударением	Ý	Ý	Ý	Ý
THORN	Þ	Þ	Þ	Þ
двойное s	ß	ß	ß	ß
a с тупым ударением	à	à	à	à
a с острым ударением	á	á	á	á
a с циркумфлексом	â	â	â	â
a с тильдой	ã	ã	ã	ã
a с диерезой	ä	ä	ä	ä
a с кружком	å	å	å	å
лигатура ae	æ	æ	æ	æ
c с седилем	ç	ç	ç	ç
e с тупым ударением	è	è	è	è
e с острым ударением	é	é	é	é
e с циркумфлексом	ê	ê	ê	ê
e с диерезой	ë	ë	ë	ë
i с тупым ударением	ì	ì	ì	ì
i с острым ударением	í	í	í	í
i с циркумфлексом	î	î	î	î
i с диерезой	ï	ï	ï	ï

Eth	ð	ð	ð	ð
n с тильдой	ñ	ñ	ñ	ñ
o с тупым ударением	ò	ò	ò	ò
o с острым ударением	ó	ó	ó	ó
o с циркумфлексом	ô	ô	ô	ô
o с тильдой	õ	õ	õ	õ
o с диерезой	ö	ö	ö	ö
знак деления	÷	÷	÷	÷
o перечеркнутое	ø	ø	ø	ø
u с тупым ударением	ù	ù	ù	ù
u с острым ударением	ú	ú	ú	ú
u с циркумфлексом	û	û	û	û
u с диерезой	ü	ü	ü	ü
y с острым ударением	ý	ý	ý	ý
thorn	þ	þ	þ	þ
y с диерезой	ÿ	ÿ	ÿ	ÿ
кавычка	"	"	"	"
амперсант	&	&	&	&
левая угловая скобка	<	<	<	<
правая угловая скобка	>	>	>	>
лигатура OE	Œ	Œ	Œ	Œ
лигатура oe	œ	œ	œ	œ
S с птичкой	Š	Š	Š	Š
s с птичкой	š	š	š	š
Y с диерезой	Ÿ	Ÿ	Ÿ	Ÿ
циркумфлекс	ˆ	ˆ	ˆ	ˆ
малая тильда	˜	˜	˜	˜
короткий пробел				
длинный пробел				
узкий пробел				

разделитель нулевой ширины	‌	‌	‌	
соединитель нулевой ширины	‍	‍	‍	
указатель слева направо	‎	‎	‎	
указатель справа налево	‏	‏	‏	
короткое тире	–	–	–	–
длинное тире	—	—	—	—
открывающая одинарная кавычка	‘	‘	‘	‘
закрывающая одинарная кавычка	’	’	’	’
нижняя одинарная кавычка	‚	‚	‚	‚
открывающая двойная кавычка	“	“	“	“
закрывающая двойная кавычка	”	”	”	”
нижняя двойная кавычка	„	„	„	„
кинжал	†	†	†	†
двойной кинжал	‡	‡	‡	‡
знак промилле	‰	‰	‰	‰
открывающая угловая кавычка	‹	‹	‹	‹
закрывающая угловая кавычка	›	›	›	›
евро	€	€	€	€
курсивное f	ƒ	ƒ	ƒ	*ƒ*
прописная альфа	Α	Α	Α	Α
прописная бета	Β	Β	Β	Β
прописная гамма	Γ	Γ	Γ	Γ

прописная дельта	Δ	Δ	Δ	Δ
прописной эпсилон	Ε	Ε	Ε	Ε
прописная дзета	Ζ	Ζ	Ζ	Ζ
прописная эта	Η	Η	Η	Η
прописная тета	Θ	Θ	Θ	Θ
прописная иота	Ι	Ι	Ι	Ι
прописная каппа	Κ	Κ	Κ	Κ
прописная ламбда	Λ	Λ	Λ	Λ
прописная мю	Μ	Μ	Μ	Μ
прописная ню	Ν	Ν	Ν	Ν
прописная кси	Ξ	Ξ	Ξ	Ξ
прописной омикрон	Ο	Ο	Ο	Ο
прописная пи	Π	Π	Π	Π
прописная ро	Ρ	Ρ	Ρ	Ρ
прописная сигма	Σ	Σ	Σ	Σ
прописная тау	Τ	Τ	Τ	Τ
прописная ипсилон	Υ	Υ	Υ	Υ
прописная фи	Φ	Φ	Φ	Φ
прописная хи	Χ	Χ	Χ	Χ
прописная пси	Ψ	Ψ	Ψ	Ψ
прописная омега	Ω	Ω	Ω	Ω
строчная альфа	α	α	α	α
строчная бета	β	β	β	β
строчная гамма	γ	γ	γ	γ
строчная дельта	δ	δ	δ	δ
строчная эпсилон	ε	ε	ε	ε
строчная дзета	ζ	ζ	ζ	ζ
строчная эта	η	η	η	η
строчная тета	θ	θ	θ	θ
строчная иота	ι	ι	ι	ι

строчная каппа	κ	κ	κ	κ
строчная ламбда	λ	λ	λ	λ
строчная мю	μ	μ	μ	μ
строчная ню	ν	ν	ν	ν
строчная кси	ξ	ξ	ξ	ξ
строчный омикрон	ο	ο	ο	ο
строчная пи	π	π	π	π
строчная ро	ρ	ρ	ρ	ρ
строчная сигма конечная	ς	ς	ς	ς
строчная сигма	σ	σ	σ	σ
строчная тау	τ	τ	τ	τ
строчная ипсилон	υ	υ	υ	υ
строчная фи	φ	φ	φ	φ
строчная хи	χ	χ	χ	χ
строчная пси	ψ	ψ	ψ	ψ
строчная омега	ω	ω	ω	ω
символ строчная тета	ϑ	ϑ	ϑ	ϑ
ипсилон с крючком	ϒ	ϒ	ϒ	ϒ
символ пи	ϖ	ϖ	ϖ	ϖ
маркер списка	•	•	•	•
многоточие	…	…	…	…
знак прим	′	′	′	′
знак двойной прим	″	″	″	″
надчеркивание	‾	‾	‾	‾
дробная черта	⁄	⁄	⁄	⁄
рукописная Р	℘	℘	℘	℘
мнимая часть числа	ℑ	ℑ	ℑ	ℑ
действительная часть числа	ℜ	ℜ	ℜ	ℜ
торговая марка	™	™	™	™

алеф	ℵ	ℵ	ℵ	ℵ
стрелка влево	←	←	←	←
стрелка вверх	↑	↑	↑	↑
стрелка вправо	→	→	→	→
стрелка вниз	↓	↓	↓	↓
стрелка влево-вправо	↔	↔	↔	↔
возврат каретки	↵	↵	↵	↵
двойная стрелка влево	⇐	⇐	⇐	⇐
двойная стрелка вверх	⇑	⇑	⇑	⇑
двойная стрелка вправо	⇒	⇒	⇒	⇒
двойная стрелка вниз	⇓	⇓	⇓	⇓
двойная стрелка влево-вправо	⇔	⇔	⇔	⇔
квантор всеобщности	∀	∀	∀	∀
знак дифференциала	∂	∂	∂	∂
квантор существования	∃	∃	∃	∃
пустое множество	∅	∅	∅	∅
набла	∇	∇	∇	∇
принадлежит множеству	∈	∈	∈	∈
не принадлежит множеству	∉	∉	∉	∉
является членом	∋	∋	∋	∋
n-арное произведение	∏	∏	∏	∏
n-арная сумма	∑	∑	∑	∑
знак минус	−	−	−	−
оператор звездочка	∗	∗	∗	∗
радикал	√	√	√	√
пропорционально	∝	∝	∝	∝
бесконечность	∞	∞	∞	∞
угол	∠	∠	∠	∠

логическое И	∧	∧	∧	∧
логическое ИЛИ	∨	∨	∨	∨
пересечение	∩	∩	∩	∩
объединение	∪	∪	∪	∪
интеграл	∫	∫	∫	∫
следовательно	∴	∴	∴	∴
оператор тильда	∼	∼	∼	∼
приблизительно равно	≅	≅	≅	≅
асимптотически равно	≈	≈	≈	≈
не равно	≠	≠	≠	≠
тождественно равно	≡	≡	≡	≡
меньше или равно	≤	≤	≤	≤
больше или равно	≥	≥	≥	≥
подмножество	⊂	⊂	⊂	⊂
надмножество	⊃	⊃	⊃	⊃
не подмножество	⊄	⊄	⊄	⊄
подмножество или равно	⊆	⊆	⊆	⊆
надмножество или равно	⊇	⊇	⊇	⊇
прямая сумма	⊕	⊕	⊕	⊕
векторное произведение	⊗	⊗	⊗	⊗
перпендикулярно	⊥	⊥	⊥	⊥
оператор точка	⋅	⋅	⋅	⋅
левый верхний угол	⌈	⌈	⌈	⌈
правый верхний угол	⌉	⌉	⌉	⌉
левый нижний угол	⌊	⌊	⌊	⌊
правый нижний угол	⌋	⌋	⌋	⌋
левая угловая скобка	⟨	〈	〈	⟨
правая угловая скобка	⟩	〉	〉	⟩
ромб	◊	◊	◊	◊
пики	♠	♠	♠	♠

Окончание прил. 1

трефы	♣	♣	♣	♣
червы	♥	♥	♥	♥
бубны	♦	♦	♦	♦

ПРИЛОЖЕНИЕ 2

Стандартные кодировки символов

Наимено-вание	**Идентифи-катор**	**Допустимые синонимы**	**Ко-довая стр.**
Unicode (UTF-7)	utf-7	csUnicode11UTF7, unicode-utf_7, unicode-1-1-utf-7, unicode-2-0-utf-7, x-unicode-2-0-utf-7	65000
Unicode (UTF-8)	utf-8	unicode-utf_8, unicode-utf_8-1_1, unicode-1-1-utf-8, unicode-2-0-utf-8, x-unicode-2-0-utf-8	65001
Unicode (UTF-16)	unicode-utf_16	csUnicode11, csUnicodeASCII, csUnicodeLatin1, ISO-10646-UCS-2, ISO-10646-UCS-BASIC, ISO-10646-Unicode-Latin1, ISO-10646, ISO-10646-J-1, unicode-utf_16-1.1	
Unicode (UTF-32)	unicode-utf_32	ISO-10646-UCS-4	
Арабский (ASMO 708)	ASMO-708		708
Арабский (DOS)	DOS-720		720
Арабский (ISO)	iso-8859-6	csISOLatinArabic, iso-ir-127, ISO_8859-6, ISO_8859-6:1987, iso-8859_6-1999, ECMA-114, arabic	28596
Арабский (Windows)	windows-1256	cp1256, windows-1256-2000	1256

Балтий-ские языки (Latin-4)	iso-8859-4	csISOLatin4, iso-ir-110, ISO_8859-4, ISO_8859-4:1988, iso-8859_4-1988, l4, latin4	28594
Балтий-ские языки (Windows)	windows-1257	windows-1257-2000	1257
Вьетнам-ский (Windows)	windows-1258	windows-1258-2000	1258
Греческий (ISO)	iso-8859-7	csISOLatinGreek, iso-ir-126, ISO_8859-7, ISO_8859-7:1987, iso-8859_7-1987, ECMA-118, ELOT_928, greek, greek8	28597
Греческий (Windows)	windows-1253	windows-1253-2000, x-cp1253	1253
Европа (Latin-3)	iso-8859-3	iso-8859_3-1999	28593
Европа (Latin-6)	iso-8859-10	iso-8859_10-1999	
Европа (Latin-7)	iso-8859-13	iso-8859_13-1999	
Европа (Latin-9)	iso-8859-15	iso-8859_15-1999	28605
Западная Европа (ASCII)	us-ascii	ANSI_X3.4-1968, ANSI_X3.4-1986, cp367, csASCII, IBM367, iso-ir-6, ISO646-US, ISO_646.irv:1991, ascii, us, us-ascii-1968, x-ansi	20127

Западная Европа (Latin-1)	iso-8859-1	cp819, ibm819, iso-ir-100, iso8859-1, iso_8859-1, iso_8859-1:1987, iso-8859_1-1998, ISO-8859-1-Windows-3.0-Latin-1, ISO-8859-1-Windows-3.1-Latin-1, latin1, l1	29591
Западная Европа (Mac)	mac-roman-2000	x-mac-roman	10000
Западная Европа (Windows)	windows-1252		1252
Иврит (DOS)	DOS-862		862
Иврит (ISO–визуаль-ный)	iso-8859-8	csISOLatinHebrew, iso-ir-138, ISO_8859-8, iso-8859_8-1999, ISO-8859-8 Visual, visual, hebrew	28598
Иврит (ISO–логиче-ский)	iso-8859-8-i		38598
Иврит (Windows)	windows-1255	ISO_8859-8:1988, iso-ir-138, log-ical, windows-1255-2000	1255
Кельтские языки (Latin-8)	iso-8859-14	iso-8859_14-1999	
Кириллица (DOS)	cp866	ibm866	866
Кириллица (ISO)	iso-8859-5	csISOLatinCyrillic, iso-ir-144, ISO_8859-5, ISO_8859-5:1988, iso-8859_5-1999, cyrillic	28595

Кириллица (Mac)	mac-cyrillic-2000	x-mac-cyrillic	10007
Кириллица (Windows)	windows-1251	csWindows31Latin5, iso-8859-5-windows-latin-5, windows-1251-2000, x-cp1251	1251
Кириллица (КОИ8-Р)	koi8-r	csKOI8R, koi, ru-koi8_r-2000	20866
Китайский – традиционное письмо (EUC)	x-euc-tw	osf-euc_tw-2000	51950
Китайский – традиционное письмо (Windows)	big5	csBig5, CN-Big5, windows-950, windows-950-2000, x-x-big5	950
Китайский – упрощенное письмо (GB2312)	gb2312	csGB2312, csISO58GB23128, iso-ir-58, GB2312, GBK, GB_2312-80, gb-3212-2000, chinese, CN-GB, CN-GB-ISOIR165	936
Китайский – упрощенное письмо (HZ)	hz-gb-2312		52936
Корейский (EUC)	euc-kr	csEUCKR, ks_c_5601, kcs5601, osf-euc_kr-2000	51949
Корейский (ISO)	iso-2022-kr	csISO2022KR, posix-2022_kr	50225
Корейский (Windows)	ks_c_5601-1987	csKSC56011987, korean, windows-949, windows-949-2000	949

Тайский (Windows)	iso-8859-11	windows-874, windows-874-2000	874
Турецкий (ISO)	iso-8859-9	csISOLatin5, iso-ir-148, l5, ISO_8859-9, ISO_8859-9:1989, iso-8859_9-1999, latin5	28599
Турецкий (Windows)	windows-1254	windows-1254-2000, x-cp1254	1254
Украин-ский (КОИ8-У)	koi8-u		21866
Централь-ная Европа (DOS)	ibm852	cp852	852
Централь-ная Европа (Latin-2)	iso-8859-2	csISOLatin2, iso-ir-101, iso8859-2, iso_8859-2, iso_8859-2:1987, iso_8859_2-1999, ISO-8859-2-Windows-Latin-2, l2, latin2	28592
Централь-ная Европа (Mac)	mac-ce-2000	x-mac-ce	10029
Централь-ная Европа (Windows)	windows-1250	x-cp1250	1250
Японский (EUC)	euc-jp	csEUCPkdFmtJapanese, Ex-tended_UNIX_Code_Packed_For mat_for_Japanese, osf-euc_jp-2000, x-euc, x-euc-jp	51932
Японский (JIS)	iso-2022-jp	csISO2022JP, csISO2022JP2, iso-2022-jp-2, posix-2022_jp-2000	50220

Японский (JIS – допускающий однобайтовую кану – SO/SI)	iso-2022-jp	csISO2022JP, _iso-2022-jp$SIO	50222
Японский (JIS – допускающий однобайтовую кану)	csISO2022JP	_iso-2022-jp$ESC	50221
Японский (Windows)	shift_jis	csShiftJIS, csWindows31J, ms_Kanji, shift-jis, x-ms-cp932, x-sjis, Windows-31J, windows-932-2000	932

ПРИЛОЖЕНИЕ 3

Коды и названия цветов

Название цвета	Красный	Зеленый	Синий	Валидность
aliceblue	F0	F8	FF	-
antiquewhite	FA	EB	D7	-
aqua	00	FF	FF	+
aquamarine	7F	FF	D4	-
azure	F0	FF	FF	-
beige	F5	F5	DC	-
bisque	FF	E4	C4	-
black	00	00	00	+
blanchedalmond	FF	EB	CD	-
blue	00	00	FF	+
blueviolet	8A	2B	E2	-
brown	A5	2A	2A	-
burlywood	DE	B8	87	-
cadetblue	5F	9E	A0	-
chartreuse	7F	FF	00	-
chocolate	D2	69	1E	-
coral	FF	7F	50	-
cornflowerblue	64	95	ED	-
cornsilk	FF	F8	DC	-
crimson	DC	14	3C	-
cyan	00	FF	FF	-
darkblue	00	00	8B	-
darkcyan	00	8B	8B	-
darkgoldenrod	B8	86	0B	-
darkgray	A9	A9	A9	-
darkgreen	00	64	00	-
darkkhaki	BD	B7	6B	-

darkmagenta	8B	00	8b	-
darkolivegreen	55	6B	2F	-
darkorange	FF	8C	00	-
darkochid	99	32	CC	-
darkred	8B	00	00	-
darksalmon	E9	96	7A	-
darkseagreen	8F	BC	8F	-
darkslateblue	48	3D	8B	-
darkslategray	2F	4F	4F	-
darkturquoise	00	CE	D1	-
darkviolet	94	00	D3	-
deeppink	FF	14	93	-
deepskyblue	00	BF	FF	-
dimgray	69	69	69	-
dodgerblue	1E	90	FF	-
firebrick	B2	22	22	-
floralwhite	FF	FA	F0	-
forestgreen	22	8B	22	-
fuchsia	FF	00	FF	+
gainsboro	DC	DC	DC	-
ghostwhite	F8	F8	FF	-
gold	FF	D7	00	-
goldenrod	DA	A5	20	-
gray	80	80	80	+
green	00	80	00	+
greenyellow	AD	FF	2F	-
honeydew	F0	FF	F0	-
hotpink	FF	69	B4	-
indiandred	CD	5C	5C	-
indigo	4B	00	82	-

ivory	FF	FF	F0	-
khaki	F0	E6	8C	-
lavender	E6	E6	FA	-
lavenderblush	FF	F0	F5	-
lawngreen	7C	FC	00	-
lemonchiffon	FF	FA	CD	-
ligtblue	AD	D8	E6	-
lightcoral	F0	80	80	-
lightcyan	E0	FF	FF	-
lightgoldenrodyellow	FA	FA	D2	-
lightgreen	90	EE	90	-
lightgrey	D3	D3	D3	-
lightpink	FF	B6	C1	-
lightsalmon	FF	A0	7A	-
lightseagreen	20	B2	AA	-
lightscyblue	87	CE	FA	-
lightslategray	77	88	99	-
lightsteelblue	B0	C4	DE	-
lightyellow	FF	FF	E0	-
lime	00	FF	00	+
limegreen	32	CD	32	-
linen	FA	F0	E6	-
magenta	FF	00	FF	-
maroon	80	00	00	+
mediumaquamarine	66	CD	AA	-
mediumblue	00	00	CD	-
mediumorchid	BA	55	D3	-
mediumpurple	93	70	DB	-
mediumseagreen	3C	B3	71	-
mediumslateblue	7B	68	EE	-

mediumspringgreen	00	FA	9A	-
mediumturquoise	48	D1	CC	-
medium violetred	C7	15	85	-
midnightblue	19	19	70	-
mintcream	F5	FF	FA	-
mistyrose	FF	E4	E1	-
moccasin	FF	E4	B5	-
navajowhite	FF	DE	AD	-
navy	00	00	80	+
oldlace	FD	F5	E6	-
olive	80	80	00	+
olivedrab	6B	8E	23	-
orange	FF	A5	00	-
orangered	FF	45	00	-
orchid	DA	70	D6	-
palegoldenrod	EE	E8	AA	-
palegreen	98	FB	98	-
paleturquose	AF	EE	EE	-
palevioletred	DB	70	93	-
papayawhop	FF	EF	D5	-
peachpuff	FF	DA	B9	-
peru	CD	85	3F	-
pink	FF	C0	CB	-
plum	DD	A0	DD	-
powderblue	B0	E0	E6	-
purple	80	00	80	+
red	FF	00	00	+
rosybrown	BC	8F	8F	-
royalblue	41	69	E1	-
saddlebrown	8B	45	13	-

salmon	FA	80	72	-
sandybrown	F4	A4	60	-
seagreen	2E	8B	57	-
seashell	FF	F5	EE	-
sienna	A0	52	2D	-
silver	C0	C0	C0	+
skyblue	87	CE	EB	-
slateblue	6A	5A	CD	-
slategray	70	80	80	-
snow	FF	FA	FA	-
springgreen	00	FF	7F	-
steelblue	46	82	B4	-
tan	D2	B4	8C	-
teal	00	80	80	+
thistle	D8	BF	D8	-
tomato	FF	63	47	-
turquose	40	E0	D0	-
violet	EE	82	EE	-
wheat	F5	DE	B3	-
white	FF	FF	FF	+
whitesmoke	F5	F5	F5	-
yellow	FF	FF	00	+
yellowgreen	9A	CD	32	-

ПРИЛОЖЕНИЕ 4

MIME-типы

Расширение файла	Тип данных
ai	application/postscript
aif	audio/aiff
aif	audio/x-aiff
aiff	audio/aiff
aiff	audio/x-aiff
ani	application/x-navi-animation
aos	application/x-nokia-9000-communicator-add-on-software
aps	application/mime
arc	application/octet-stream
arj	application/arj
arj	application/octet-stream
art	image/x-jg
asf	video/x-ms-asf
asm	text/x-asm
asp	text/asp
asx	application/x-mplayer2
asx	video/x-ms-asf
asx	video/x-ms-asf-plugin
au	audio/basic
au	audio/x-au
avi	application/x-troff-msvideo
avi	video/avi
avi	video/msvideo
avi	video/x-msvideo
bin	application/mac-binary

bin	application/macbinary
bin	application/octet-stream
bin	application/x-binary
bin	application/x-macbinary
bm	image/bmp
bmp	image/bmp
bmp	image/x-windows-bmp
boo	application/book
book	application/book
c	text/x-c
c++	text/plain
ccad	application/clariscad
class	application/java
class	application/java-byte-code
class	application/x-java-class
com	application/octet-stream
com	text/plain
conf	text/plain
cpp	text/x-c
cpt	application/mac-compactpro
cpt	application/x-compactpro
cpt	application/x-cpt
css	application/x-pointplus
css	text/css
dcr	application/x-director
def	text/plain
dif	video/x-dv
dir	application/x-director
dl	video/dl
dl	video/x-dl

Doc	application/msword
dot	application/msword
drw	application/drafting
dvi	application/x-dvi
dwg	application/acad
dwg	image/vnd.dwg
dwg	image/x-dwg
dxf	application/dxf
dxf	image/vnd.dwg
dxf	image/x-dwg
dxr	application/x-director
exe	application/octet-stream
gif	image/gif
gz	application/x-compressed
gz	application/x-gzip
gzip	application/x-gzip
gzip	multipart/x-gzip
h	text/plain
h	text/x-h
hlp	application/hlp
hlp	application/x-helpfile
hlp	application/x-winhelp
htc	text/x-component
htm	text/html
html	text/html
htmls	text/html
htt	text/webviewhtml
ice	x-conference/x-cooltalk
ico	image/x-icon
inf	application/inf

jam	audio/x-jam
jav	text/plain
jav	text/x-java-source
java	text/plain
java	text/x-java-source
jcm	application/x-java-commerce
jfif	image/jpeg
jfif	image/pjpeg
jfif-tbnl	image/jpeg
jpe	image/jpeg
jpe	image/pjpeg
jpeg	image/jpeg
jpeg	image/pjpeg
jpg	image/jpeg
jpg	image/pjpeg
jps	image/x-jps
js	application/x-javascript
js	application/javascript
js	application/ecmascript
js	text/javascript
js	text/ecmascript
latex	application/x-latex
lha	application/lha
lha	application/octet-stream
lha	application/x-lha
lhx	application/octet-stream
list	text/plain
lsp	application/x-lisp
lsp	text/x-script.lisp
lst	text/plain

Lzh	application/octet-stream
lzh	application/x-lzh
lzx	application/lzx
lzx	application/octet-stream
lzx	application/x-lzx
m3u	audio/x-mpequrl
man	application/x-troff-man
mid	application/x-midi
mid	audio/midi
mid	audio/x-mid
mid	audio/x-midi
mid	music/crescendo
mid	x-music/x-midi
midi	application/x-midi
midi	audio/midi
midi	audio/x-mid
midi	audio/x-midi
midi	music/crescendo
midi	x-music/x-midi
mod	audio/mod
mod	audio/x-mod
mov	video/quicktime
movie	video/x-sgi-movie
mp2	audio/mpeg
mp2	audio/x-mpeg
mp2	video/mpeg
mp2	video/x-mpeg
mp2	video/x-mpeq2a
mp3	audio/mpeg3
mp3	audio/x-mpeg-3

mp3	video/mpeg
mp3	video/x-mpeg
mpa	audio/mpeg
mpa	video/mpeg
mpeg	video/mpeg
mpg	audio/mpeg
mpg	video/mpeg
mpga	audio/mpeg
pas	text/pascal
pcl	application/vnd.hp-pcl
pcl	application/x-pcl
pct	image/x-pict
pcx	image/x-pcx
pdf	application/pdf
pic	image/pict
pict	image/pict
pl	text/plain
pl	text/x-script.perl
pm	image/x-xpixmap
pm	text/x-script.perl-module
pm4	application/x-pagemaker
pm5	application/x-pagemaker
png	image/png
pot	application/mspowerpoint
pot	application/vnd.ms-powerpoint
ppa	application/vnd.ms-powerpoint
pps	application/mspowerpoint
pps	application/vnd.ms-powerpoint
ppt	application/mspowerpoint
ppt	application/powerpoint

ppt	application/vnd.ms-powerpoint
ppt	application/x-mspowerpoint
ppz	application/mspowerpoint
ps	application/postscript
psd	application/octet-stream
pwz	application/vnd.ms-powerpoint
py	text/x-script.phyton
pyc	applicaiton/x-bytecode.python
qt	video/quicktime
qtif	image/x-quicktime
ra	audio/x-pn-realaudio
ra	audio/x-pn-realaudio-plugin
ra	audio/x-realaudio
ram	audio/x-pn-realaudio
rm	application/vnd.rn-realmedia
rm	audio/x-pn-realaudio
rpm	audio/x-pn-realaudio-plugin
rtf	application/rtf
rtf	application/x-rtf
rtf	text/richtext
rtx	application/rtf
rtx	text/richtext
rv	video/vnd.rn-realvideo
sgml	text/sgml
sgml	text/x-sgml
sh	application/x-bsh
sh	application/x-sh
sh	application/x-shar
sh	text/x-script.sh
shtml	text/html

shtml	text/x-server-parsed-html
ssi	text/x-server-parsed-html
tar	application/x-tar
tcl	application/x-tcl
tcl	text/x-script.tcl
text	application/plain
text	text/plain
tgz	application/gnutar
tgz	application/x-compressed
tif	image/tiff
tif	image/x-tiff
tiff	image/tiff
tiff	image/x-tiff
txt	text/plain
uri	text/uri-list
vcd	application/x-cdlink
vmd	application/vocaltec-media-desc
vrml	application/x-vrml
vrml	model/vrml
vrml	x-world/x-vrml
vsd	application/x-visio
vst	application/x-visio
vsw	application/x-visio
wav	audio/wav
wav	audio/x-wav
wmf	windows/metafile
xla	application/excel
xla	application/x-excel
xla	application/x-msexcel
xlb	application/excel

xlb	application/vnd.ms-excel
xlb	application/x-excel
xlc	application/excel
xlc	application/vnd.ms-excel
xlc	application/x-excel
xld	application/excel
xld	application/x-excel
xlk	application/excel
xlk	application/x-excel
xll	application/excel
xll	application/vnd.ms-excel
xll	application/x-excel
xlm	application/excel
xlm	application/vnd.ms-excel
xlm	application/x-excel
xls	application/excel
xls	application/vnd.ms-excel
xls	application/x-excel
xls	application/x-msexcel
xlt	application/excel
xlt	application/x-excel
xlv	application/excel
xlv	application/x-excel
xlw	application/excel
xlw	application/vnd.ms-excel
xlw	application/x-excel
xlw	application/x-msexcel
xm	audio/xm
xml	application/xml
xml	text/xml

Окончание прил. 4

z	application/x-compress
z	application/x-compressed
zip	application/x-compressed
zip	application/x-zip-compressed
zip	application/zip
zip	multipart/x-zip

ПРИЛОЖЕНИЕ 5

Валидность тегов в различных версиях HTML

Тег	HTML 4.01 / XHTML 1.0			HTML5	XHTML 1.1
	Transi-tional	Strict	Frameset		
<!-- -->	+	+	+	+	+
<a>	+	+	+	+	+
<abbr>	+	+	+	+	+
<acronym>	+	+	+	-	+
<address>	+	+	+	+	+
<applet>	+	-	+	-	-
<area>	+	+	+	+	+
<article>	-	-	-	+	-
<aside>	-	-	-	+	-
<audio>	-	-	-	+	-
<b>	+	+	+	+	+
<base>	+	+	+	+	+
<basefont>	+	-	+	-	-
<bdo>	+	+	+	+	+
<bgsound>	-	-	-	-	-
<big>	+	+	+	-	+
<blink>	-	-	-	-	-
<blockquote>	+	+	+	+	+
<body>	+	+	+	+	+
 	+	+	+	+	+
<button>	+	+	+	+	+
<canvas>	-	-	-	+	-
<caption>	+	+	+	+	+
	+	-	+	-	-
<cite>	+	+	+	+	+

<code>	+	+	+	+	+
<col>	+	+	+	+	+
<colgroup>	+	+	+	+	+
<command>	-	-	-	+	-
<comment>	-	-	-	-	-
<datalist>	-	-	-	+	-
<dd>	+	+	+	+	+
<del>	+	+	+	+	+
<details>	-	-	-	+	-
<dfn>	+	+	+	+	+
<dir>	+	-	+	-	-
	+	+	+	+	+
<dl>	+	+	+	+	+
<dt>	+	+	+	+	+
	+	+	+	+	+
<embed>	-	-	-	+	-
<fieldset>	+	+	+	+	+
<figcaption>	-	-	-	+	-
<figure>	-	-	-	+	-
<font>	+	-	+	-	-
<footer>	-	-	-	+	-
<form>	+	+	+	+	+
<frame>	-	-	+	-	-
<frameset>	-	-	+	-	-
<h1>, <h2>, <h3>, <h4>, <h5>, <h6>	+	+	+	+	+
<head>	+	+	+	+	+
<header>	-	-	-	+	-
<hgroup>	-	-	-	+	-

<hr>	+	+	+	+	+
<html>	+	+	+	+	+
<i>	+	+	+	+	+
<iframe>	+	-	+	+	-
<img>	+	+	+	+	+
<input>	+	+	+	+	+
<ins>	+	+	+	+	+
<isindex>	+	-	+	-	-
<kbd>	+	+	+	+	+
<keygen>	-	-	-	+	-
<label>	+	+	+	+	+
<legend>	+	+	+	+	+
	+	+	+	+	+
<link>	+	+	+	+	+
<map>	+	+	+	+	+
<mark>	-	-	-	+	-
<marquee>	-	-	-	-	-
<menu>	+	-	+	+	-
<meta>	+	+	+	+	+
<meter>	-	-	-	+	-
<nav>	-	-	-	+	-
<nobr>	-	-	-	-	-
<noembed>	-	-	-	-	-
<noframes>	+	-	+	-	-
<noscript>	+	+	+	+	+
<object>	+	+	+	+	+
	+	+	+	+	+
<optgroup>	+	+	+	+	+
<option>	+	+	+	+	+
<output>	-	-	-	+	-

<p>	+	+	+	+	+
<param>	+	+	+	+	+
<plaintext>	-	-	-	-	-
<pre>	+	+	+	+	+
<progress>	-	-	-	+	-
<q>	+	+	+	+	+
<rp>	-	-	-	+	-
<rt>	-	-	-	+	-
<ruby>	-	-	-	+	-
<s>	+	-	+	-	-
<samp>	+	+	+	+	+
<script>	+	+	+	+	+
<section>	-	-	-	+	-
<select>	+	+	+	+	+
<small>	+	+	+	+	+
<source>	-	-	-	+	-
	+	+	+	+	+
<strike>	+	-	+	-	-
	+	+	+	+	+
<style>	+	+	+	+	+
<sub>	+	+	+	+	+
<sup>	+	+	+	+	+
<summary>	-	-	-	+	-
<table>	+	+	+	+	+
<tbody>	+	+	+	+	+
<td>	+	+	+	+	+
<textarea>	+	+	+	+	+
<tfoot>	+	+	+	+	+
<th>	+	+	+	+	+
<thead>	+	+	+	+	+

Окончание прил. 5

<time>	-	-	-	+	-
<title>	+	+	+	+	+
<tr>	+	+	+	+	+
<tt>	+	+	+	-	+
<u>	+	-	+	-	-
	+	+	+	+	+
<var>	+	+	+	+	+
<video>	-	-	-	+	-
<wbr>	-	-	-	+	-
<xmp>	-	-	-	-	-

ПРИЛОЖЕНИЕ 6

Браузеры и поддерживаемые ими кодеки

Браузер	Internet Explorer	Chrome	Opera	Safari	Firefox
Аудио кодеки					
ogg/vorbis	-	+	+	-	+
wav	-	-	+	+	+
mp3	+	+	-	+	-
AAC	+	+	-	+	-
Видео кодеки					
ogg/theora	-	+	+	-	+
H.264	+	+	-	+	-
WebM	-	+	+	-	+

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Дронов В.А. Adobe Dreamweaver CS4 / В.А. Дронов. СПб.: БХВ-Петербург, 2009. 832 с.

2. Основы Web-технологий / П.Б. Храмцов, С.А. Брик, А.М. Русак, А.И. Сурин. М.: ИНТУИТ.РУ «Интернет-Университет Информационных Технологий», 2003. 512 с.

3. Прохоренок Н.А. HTML, JavaScript, PHP и MySQL. Джентельменский набор Web-мастера / Н.А. Прохоренок. СПб.: БХВ-Петербург, 2010. 912 с.

4. Макфарланд Д. Большая книга CSS. 2-е изд. / Д. Макфарланд. СПб.: Питер, 2011. 560 с.

5. Дронов В.А. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов / В.А. Дронов. СПб.: БХВ-Петербург, 2011. 416 с.

6. Сергеева Т.И. Web-дизайн: учеб. пособие / Т.И. Сергеева, М.Ю. Сергеев. Воронеж: ВГТУ, 2006. 185 с.

7. Сергеева Т.И. Введение в Web-дизайн/ Т.И. Сергеева, М.Ю. Сергеев. Воронеж: ВГТУ, 2007. 175 с.

8. Основы Web-дизайна / Т.И. Сергеева, М.Ю. Сергеев, С.С. Валеев, Ю.Г. Строкина. Уфа: УГАТУ, 2009. 160 с.

9. htmlbook.ru – справочник по HTML и CSS.

ОГЛАВЛЕНИЕ

Введение 3

1. Создание веб-страниц с помощью тегов HTML 4
   - 1.1. История развития HTML 4
   - 1.2. Структура HTML-документа 5
   - 1.3. Элементы разметки заголовка документа 8
   - 1.4. Определитель типа документа 13
   - 1.5. Тег <body> – контейнер тела документа 15
   - 1.6. Теги для оформления текста документа 17
   - 1.7. Теги для логического выделения фрагментов текста 27
   - 1.8. Теги для оформления таблиц 32
   - 1.9. Создание и оформление гиперссылок 41
   - 1.10. Добавление и оформление изображений 44
   - 1.11. Добавление на страницу различных объектов 49
   - 1.12. Создание элементов форм 52
   - 1.13. Универсальные атрибуты 65
   - 1.14. Особенности XHTML 68
2. Использование каскадных таблиц стилей CSS 71
   - 2.1. Способы применения стилей 71
   - 2.2. Управление селекторами стилей 75
   - 2.3. Стилизация текста и списков 82
   - 2.4. Управление полями, отступами и границами 91
   - 2.5. Стилизация фона элементов 96
   - 2.6. Управление размерами элементов. Плавающие элементы 99
   - 2.7. Разметка страницы на основе плавающих элементов 105
   - 2.8. Стилизация ссылок 117
   - 2.9. Создание панелей навигации 126
   - 2.10. Позиционирование элементов на веб-странице 134
3. Новые технологии web-дизайна 144
   - 3.1. Общие сведения о HTML 5 144
   - 3.2. Теги для отображения мультимедийных объектов 146

3.3 Теги для обозначения блоков страницы 152
3.4. Новые элементы веб-форм 158
3.5. Общие сведения о CSS 3 155
3.6. Селекторы в CSS 3 159
3.7. Форматирование текста в CSS 3 164
3.8. Работа с прозрачностью и фоном элемента в CSS 3 170
3.9. Стилизация границ элемента в CSS 3 175
Заключение 179
Приложение 1 180
Приложение 2 190
Приложение 3 196
Приложение 4 201
Приложение 5 211
Приложение 6 216
Библиографический список 217

Учебное издание

Сергеев Михаил Юрьевич
Сергеева Татьяна Ивановна

WEB-ДИЗАЙН:
СОЗДАНИЕ WEB-САЙТОВ
С ПОМОЩЬЮ HTML И CSS

В авторской редакции

Подписано в печать 30.06.2012.
Формат 60×84/16. Бумага для множительных аппаратов.
Усл. печ. л. 13.8. Уч.-изд. л. 11,4. Тираж 250 экз.
Заказ. №

ФГБОУ ВПО «Воронежский государственный
технический университет»
394026 Воронеж, Московский просп., 14