

ГЛАВА 2. HYPERTEXT MARKUP LANGUAGE (HTML)

§ 2.1. Понятие HTML

Всемирная паутина World Wide Web (WWW) соткана из Webстраниц, которые создаются с помощью так называемого языка разметки гипертекста HTML (Hypertext Markup Language). Хотя многие говорят о программировании на этом языке, HTML вовсе не является языком программирования в традиционном понимании. HTML — язык разметки документа. При разработке HTML-документа выполняется разметка текстового документа точно так же, как это делает редактор при помощи красного карандаша. Эти пометки служат для указания формы представления информации, содержащейся в документе: где будут заголовки, что выделить цветом или может подчеркнуть, а где будут находиться иллюстрации.

HTML язык представляет собой совокупность команд, которые называемых тегами (от английского слова tag). Их добавление в простой тестовый документ позволяет получить единообразно оформленные и связанные между собой ссылками «web-страницы». Документ же при этом получает название гипертекстового, как содержащего внутри себя ссылки на другие документы.

Для просмотра HTML-документов используются специальные программы, которые называют браузерами. Они позволяют интерпретировать файлы, размеченные по правилам языка HTML, форматировать их в виде Web-страниц и отображать на экране компьютера пользователя. Существует большое количество программбраузеров, разработанных различными компаниями, каждый из которых имеет свои уникальные особенности[11]. На сегодняшний день наибольшей популярностью пользуются Google Chrome, Firefox, Internet Explorer и Opera.

Google Chrome (<http://www.google.com/chrome?hl=ru>) — один из самых свежих браузеров, появившийся на рынке в конце 2008 года, но уже завоевавший признание пользователей. Он снабжен поиском в системе Google из адресной строки, почтой, Google диском и другими расширениями от компании Google.

Mozilla Firefox (<http://mozilla-russia.org/products/firefox/>) - перспективный и развивающийся браузер, также получивший признание во всем мире. Его особенность — простота и расширяемость, которая получается за счет специальных расширений. Изначально Firefox имеет набор только самых необходимых функций, но, устанавливая желаемые расширения, в итоге можно нарастить браузер до системы, выполняющей все необходимые для вашей работы действия. Браузер Firefox является открытой системой, разрабатываемой группой Mozilla.

Microsoft Internet Explorer (IE) (<http://www.microsoft.com/ruru/windows/internet-explorer/>) - один из старейших браузеров, который бесплатно поставляется вместе с операционной системой Windows. Это и определило его популярность. Начиная с версии IE 7, программа по удобству приблизилась к своим давним конкурентам, в частности, появились вкладки. К сожалению, этот браузер хуже всех поддерживает спецификацию HTML, поэтому для корректного отображения в IE приходится порой отдельно отлаживать код специально под него.

Opera (<http://ru.opera.com/download/>) - быстрый и удобный браузер, поддерживающий множество дополнительных возможностей, повышающих комфортность работы с сайтами.

Safari (<http://www.apple.com/ru/safari/>) - разработанный компанией Apple этот браузер встроен в iPhone и операционную систему MacOS на компьютерах Apple. Также имеется версия под Windows.

Яндекс. Браузер (<http://browser.yandex.ru/>) — самый свежий из браузеров на сегодня, анонсированных в 2012 году компанией Яндекс. В него уже интегрированы сервисы компании: поиск, почта, переводчик, облачное хранилище, средства по просмотру файлов формата PDF и SWF.

Создавая web-страницы, стоит помнить об огромном разнообразии представленных средств отображения гипертекста, и добиваться корректного (универсального) отображения во всех браузерах, просматривая страницы в большинстве из них и соответствующим образом корректируя.

Современная web-индустрия предоставляет целый ряд онлайн-сервисов, которые позволяют «воочию» убедиться, корректно ли

отображаются страницы сайта в различных браузерах[12]. Среди них бесплатная служба с русским интерфейсом Browsershots (<http://browsershots.org/>), разработанная немецким программистом Йоханом Рохоллом, Browsrcamp для генерации скриншотов сайта, открытого в браузере Safari(<http://www.browsrcamp.com/>), IE NetRenderer - бесплатный сервис с двуязычным интерфейсом (английский и немецкий), «фотографирующий» страницы сайтов с использованием браузеров Internet Explorer редакций 5.5, 6.0 и 7.0 (<http://ipinfo.info/netrenderer/>) и многие, многие другие[12].

Создание HTML-документов

HTML-документ можно создавать в любом текстовом редакторе, хоть в блокноте (notepad.exe). Для получения web-страницы, достаточно добавить соответствующие теги и сохранить файл с расширением HTML.

1. В ОС Windows достаточно выполнить следующие действия:
Пуск > Программы > Стандартные > Блокнот.
2. Набрать код в Блокноте.
3. Сохранить готовый документ (Файл > Сохранить как...) под именем, например, primer.html, при этом поставив в диалоговом окне сохранения тип файла: «Все файлы» и кодировку «UTF-8» (Рисунок 1). Обратите внимание, что расширение у файла должно быть именно html.

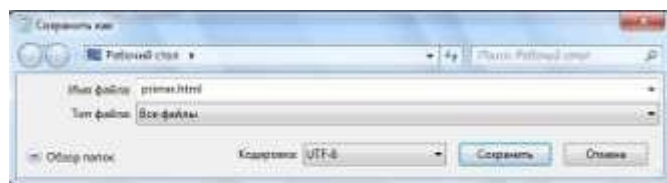


Рисунок 1. Создание html страницы.

4. Запустить файл двойным нажатием. Он откроется в браузере, установленном по умолчанию и отобразит результат вашей работы.

Блокнот можно использовать для создания гипертекстовых документов, но, для полноценной работы желательна поддержка следующих возможностей:

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

Среди наиболее известных редакторов HTML-кода выделяют

- EditPlus (<http://www.editplus.com>)
- HtmlReader (<http://manticora.ru/download.htm>),
- Notepad++ (<http://notepad-plus.sourceforge.net/ru/site.htm>),
- PSPad (<http://www.pspad.com/ru/download.php>),

Теги. Параметры тегов

Итак, основным элементом HTML-документа является HTML-тег, он записывается в угловых скобках и состоит из имени, за которым может следовать список параметров (атрибутов).

```
<тег атрибут1="значение" атрибут2="значение">
<тег атрибут1="значение"
атрибут2="значение">...</тег>
<тег параметр1="значение" параметр2="значение"> <тег
параметр1="значение" параметр2="значение">
...</тег>
```

Теги бывают двух типов — одиночные и парные (контейнеры). Одиночный тег используется самостоятельно, а парный может включать внутри себя другие теги или текст.

Парные теги, называемые по-другому контейнеры, состоят из двух частей — открывающий и закрывающий тег. Открывающий тег обозначается, как и одиночный — <тег>, а в закрывающем - используется

слеш — `</тег>`. Допускается вкладывать в контейнер другие теги, однако следует соблюдать их порядок вложенности[11]. Таким образом, действовать необходимо, как в детской игре «Матрешка».

Существует определенная иерархия вложенности тегов. Например, тег `<title>` должен находиться внутри контейнера `<head>` и нигде иначе. Чтобы не возникло ошибки, следите за тем, чтобы теги располагались в коде правильно. Если теги между собой равноценны в иерархии связи, то их последовательность не имеет значения. Так, можно поменять местами теги `<title>` и `<meta>`, на конечном результате это никак не скажется.

Работая с закрывающими тегами, стоит помнить также и следующее: закрывающий тег бывает обязательным или не обязательным. Обязательный закрывающий тег должен присутствовать всегда, иначе это приведет к ошибке при отображении документа. Необязательный закрывающий тег говорит о том, что разработчик может его, как добавить, так и опустить, к ошибке это не приведет. Однако рекомендуем закрывать все подобные теги, включая необязательные, это дисциплинирует, создает более стройный и строгий код, который легко модифицировать.

У тегов допустимы различные параметры, которые разделяются между собой пробелом. Впрочем, есть теги без всяких дополнительных параметров. Условно параметры можно подразделить на обязательные, они непременно должны присутствовать, и необязательные, их добавление зависит от цели применения тега.

Все значения параметров тегов желательно указывать в двойных ("пример") или одинарных кавычках ('пример'). Отсутствие кавычек не приведет к ошибкам, браузеры во многих случаях достаточно корректно обрабатывают код и без кавычек, за исключением текста, содержащего пробелы.

Любые теги, а также их параметры нечувствительны к регистру, поэтому форму записи вы вольны выбирать сами, как писать — `
`, `
` или `
`. В любом случае рекомендуется придерживаться выбранной формы записи на протяжении всех страниц сайта. Заметим также, что текст, полностью набранный прописными символами, читается хуже, чем текст со строчными символами или смешанный[11].

Внутри тега между его параметрами допустимо ставить перенос строк. Хотя ошибки при переносе текста в подобном случае и не возникнет, рекомендуем писать теги в одну строку, иначе ухудшается восприятие кода и его становится сложнее править.

Если какой-либо тег или его параметр был написан неверно, то браузер проигнорирует подобный тег и будет отображать текст так, словно тега и не было.

Структура HTML документа

Итак, для представления текста в браузере необходимо создать файл с расширением HTML, в который будет записан этот текст, как, например обычный текстовый документ, хранящийся в TXT-файле, и определенный набор слов – тегов. Строение web-страницы подобно строению человека: она содержит «голову» и «тело».

Сначала мы объявляем, что перед нами web-страница, используя теги `<HTML>` и `</HTML>`. Данная пара тегов является самой внешней, между его начальным и конечным тегами должна находиться вся Web-страница.

Раздел HTML определяет специфику документа, содержание которого будет интерпретироваться браузером, дает браузеру информацию о том, что документ разработан с помощью языка разметки HTML. Чаще всего тэг `<HTML>` используется без параметров. Большинство современных браузеров могут опознать документ и не содержащий тэгов `<HTML> </HTML>`, все же их употребление крайне желательно.

Итак, как мы уже говорили, у страницы есть «голова» `<HEAD></HEAD>`. Так же, как и у человека, от названия которой и произошло название тега `<HEAD>`, она расположена над «телом». И также существует она в единственном экземпляре (а то какой-то «Змей Горыныч» получится). Все, что здесь размещено, на странице мы практически не увидим, но, тем не менее, эта информация очень важна. Именно здесь определяется и имя страницы, и на каком языке она будет «разговаривать», да и вообще вся система управления, подобно нервной

системе, также находится «в голове». Иногда этот тег называют также заголовком страницы.

В теге <HEAD> могут размещаться следующие теги: <TITLE>, <BASE>, <LINK>, <META>, <STYLE>, <SCRIPT>.

«На теле» же страницы размещаются те элементы, которые предстают перед нашим взором, когда мы открываем ее в браузере подобно одежде человека, цвету его глаз, цвету кожи и т.п. Так же, как и «голова», тело у страницы всего одно. Большинство из рассматриваемых в этой главе тегов расположены именно в теле страницы.

Итак, подводя итог, для задания «скелета» страницы достаточно указать три основных тега (Рисунок 2).



Рисунок 2. «Скелет» web-страницы
Теги заголовка <HEAD>

```
<TITLE></TITLE>
```

Тег предназначен для указания имени созданному электронному документу. Следует помнить, что под именем документа в данном случае имеется в виду не файловое наименование, а название, отображаемое в заголовке окна браузера.

Название может иметь неограниченную длину и содержать любые символы, кроме некоторых зарезервированных (желательно не более 60 символов).

По умолчанию этот текст используется при создании закладки (bookmark) для документа. Избегайте безликих названий (Home Page, Index и т. д.).

<BASE>

Тег предназначен для документов, в которых используется относительный адрес и эти документы могут переноситься в другую папку или даже на другой компьютер без потери связи. Браузер ищет тег <base>, определяет полный адрес документа и корректно загружает его[11].

Например, если адрес документа указан как <base href="http://www.narfu.ru/dir1">, то при добавлении рисунков достаточно использовать относительный адрес .

При этом полный путь к изображению будет http://www.narfu.ru/dir1/images/lada.gif, что позволяет браузеру всегда находить графический файл, независимо от того, где находится текущая web-страница.

Также можно применять и иерархическую систему пути с двумя точками. Так, если изображение добавляется как , то полный путь к файлу будет http://www.narfu.ru/images/lada.gif.

Тег имеет один обязательный параметр HREF, после которого указывается полный URL-адрес документа.

```
<BASE HREF="http://ww.narfu.ru/">
```

<LINK>

Устанавливает связь с внешним документом вроде файла со стилями или со шрифтами. Он состоит из URL-адреса и параметров,

конкретизирующих отношения документов. Заголовок документа может содержать любое количество тэгов <LINK>.

```
<LINK HREF="styles/main.css" TYPE="text/css"
REL="stylesheet">
```

Возможные параметры тега <LINK> представлены в таблице 2.

Таблица 1. Возможные параметры тега <LINK>

Параметр	Назначение
HREF	Путь к связываемому файлу.
MEDIA	Определяет устройство, для которого следует применять стилевое оформление.
REL	Определяет отношения между текущим документом и файлом, на который делается ссылка.
TYPE	MIME-тип данных подключаемого файла.

Например,

```
<LINK REL="STYLESHEET" TYPE="TEXT/CSS"
HREF="HB.CSS">
```

определяет, что подключаемый файл хранит таблицу стилей (CSS).

<META>

Так называемые Мета – определения. Они предназначены для описания внутренних свойств HTML-файла, не отражающихся при просмотре Web-страницы, но использующихся для хранения информации, предназначенной для браузеров и поисковых систем. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных. Один из немногих тегов, не являющихся парным.

Каждый элемент META содержит два основных параметра, первый из которых определяет тип данных (NAME, HTTP-EQUIV), а второй — содержание (CONTENT). Также иногда используется и дополнительный параметр URL. Заголовок документа может содержать любое количество тэгов META.

Параметры типа мета-определений HTTP-EQUIV представлены в таблице 3.

Таблица 2. Параметры типа мета-определений HTTP-EQUIV

Параметр	Функция	Подпараметры	Примеры
expires	Устанавливает дату и время, после которой информация в документе будет считаться устаревшей (актуально для Интернет-ресурсов с динамически изменяющимся содержанием). По истечении срока, браузер будет перенаправлен к источнику для обновления информации.	День недели (Mon, и т.д.), число (01, 02, и т.д.), месяц (Jan, и т.д.), год, время (ч., мин., сек.) часовой пояс (GMT)	<META HTTP-EQUIV="expires" CONTENT="Sat, 25 Jan 2010 15:30:00 GMT">
refresh	Перезагрузка/ переадресация через заданный промежуток времени (в секундах) на указанный адрес. Полезно, если данные на странице часто обновляются.	URL	<META HTTP-EQUIV="refresh" CONTENT="10 ; URL=http://www.site.ru">
Параметр	Функция	Подпараметры	Примеры

Content-Type	<p>Определение типа и кодировки документа</p> <p>Наиболее употребляемые кодировки Windows-1251, KOI8-R</p>	charset	<pre><META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=Windows-1251"></pre>
--------------	--	---------	---

Это не полный перечень существующих параметров типа метаопределений HTTP-EQUIV, мы рассмотрели лишь наиболее используемые.

Теперь рассмотрим основные конструкции мета-определений второго типа — NAME (таблица 4).

Таблица 3. Основные конструкции мета-определений второго типа — NAME.

Параметр	Функция	Пример
keywords	задает набор ключевых слов документа, предназначенных для индексирования поисковыми системами (добавления информации о документе в базы данных). Слова указываются через запятую.	<pre><META NAME="keywords" CONTENT="HTML, Web, WWW, Webстраница, Интернет"></pre>
description	Указание краткого описания документа, также необходимого для поисковых систем (при поиске информации описание выводится рядом со ссылкой на найденный Интернет - ресурс).	<pre><META NAME="description" CONTENT="HTML. Язык разметки"></pre>
author	Информация об авторе (авторах) документа	<pre><META NAME="Author" CONTENT="A.M."></pre>
robots	<p>Составление правил для индексирования документа поисковыми системами (роботами)</p> <p>Подпараметры: ALL, NONE, INDEX, NOINDEX, FOLLOW, NOFOLLOW</p> <p>По умолчанию ALL</p>	<pre><META NAME="robots" CONTENT="ALL"></pre>

copyright	Установление авторского права и перечисление условий	<META name="Copyright">
Параметр	Функция	Пример
	распространения документа	content="BHV, All Right Reserved">
generator	Информация о программном обеспечении, с помощью которого создавался документ	<meta name=Generator content="Microsoft Word 10">

- ALL - разрешение индексирования документа со всеми присутствующими в нем гиперсвязями (ссылками);
- NONE запрет индексирования документа со всеми присутствующими в нем гиперсвязями;
- INDEX разрешение индексирования документа;
- NOINDEX запрет индексирования документа;
- FOLLOW разрешение индексирования присутствующих в документе гиперсвязей;
- NOFOLLOW запрет индексирования присутствующих в документе гиперсвязей.

Следует помнить о том, что использование абсолютно всех метаопределений не целесообразно. В зависимости от типа документа, его содержания и прочих факторов, необходимо выбирать только самые необходимые теги метаданных. Минимальный набор таких тегов: параметры для поисковых систем и кодировка документа.

<STYLE></STYLE>

Необязательный тег. Тег <style> применяется для определения стилей элементов web-страницы. Тег <style> необходимо также использовать внутри контейнера <head>. Можно задавать более чем один тег <style>, но желательно переносить описания в отдельные файлы *.css «таблицы описания стилей».

Во второй главе мы рассмотрим его более подробно.

<SCRIPT></SCRIPT>

Содержит код исполняемых сценариев (скриптов). Его рассмотрению посвящена третья глава.

Параметры тега <BODY>

Один из самых важных компонентов любого HTML-документа, в нем располагается содержательная часть, которая выводится браузером на экран монитора пользователя. Внутри элемента BODY можно использовать все элементы, предназначенные для дизайна Webстраницы. Также как и заголовок, тело страницы присутствует в коде в единственном экземпляре – как и человек, имеет одно тело.

Тег имеет ряд параметров, которые условно можно разделить на четыре основные группы (параметры фона, границ документа, текста и гиперссылок). Если эти параметры не указывать, то соответствующие свойства страницы — отступы, цвета и др. — будут такими, какие заданы по умолчанию в браузере.

Все параметры, которые задаются с помощью тега <BODY> можно разделить на четыре группы:

- параметры фона,
- параметры текста, □ параметры гиперссылок
- параметры границ документа.

Параметры фона

К параметрам, отвечающим за оформление фона страницы, относят следующие: цвет фона, фоновое изображение и его прокрутка.

Параметры и их синтаксис представлен в таблице 5.

Таблица 4. Параметры фона.

Параметр	Функция	Примеры
BGCOLOR	Определение цвета фона	Задание белого фона <BODY BGCOLOR="white"> <BODY BGCOLOR="#FFFFFF"> <BODY BGCOLOR="255,255,255">
Параметр	Функция	Примеры

BACKGROUND	Указание полного пути нахождения фонового рисунка	<BODY BACKGROUND="images/bg.gif">
BGPROPERTIES	Изменение свойств фона (например, фиксирование фонового рисунка) НЕ поддерживается Firefox	<BODY BACKGROUND="images/bg.gif" BGPROPERTIES="fixed"> прокручивая содержание документа, фоновое изображение остается в зафиксированном виде.

Описанные параметры не являются обязательными, однако использование BGCOLOR рекомендуется по следующей причине: пользователь в настройках своего браузера может поставить любой цвет фона, а разработчик, полагая, что белый цвет является основным по умолчанию, может не указать этот параметр. В результате вместо подразумеваемого белого цвета, фон может оказаться черным, зеленым и т. д., что способно привести к нарушению оформления документа. Также наряду с графическим изображением фона рекомендуется использовать и параметры цвета на тот случай, если рисунок не загрузится (тогда браузер отобразит цвет).

При использовании BACKGROUND стоит помнить, что изображение изначально отображается в натуральную величину. Если рисунок по размеру меньше окна браузера, то картинка повторяется по горизонтали вправо и вниз, создавая эффект мозаики. По этой причине на месте стыка фоновых картинок могут возникнуть видимые перепады, заметные для посетителей сайта. Исправить эту ситуацию возможно только с помощью каскадных таблиц стилей, о которых будет поведено в третьей главе. При выборе фонового рисунка убедитесь также, что обеспечен достаточный контраст между ним и текстом web-страницы – иначе текст просто потеряется на фоне графического изображения. В качестве фона допускается использовать, конечно, и анимированные изображения в формате GIF, но они отвлекают внимание читателей. Вообще, в качестве фонового изображения желательно использовать неброские цвета, дабы не отвлекать пользователя от основного элемента страницы – текста.

Пример задания фонового изображения.

```
<html>
  <head>
    <title>Тег BODY, параметр background</title>
  </head>
  <body background="images/snow.gif">
    ...

</body>
</html>
```

В данном случае, картинка «snow.gif» расположена в папке с именем «images», папка лежит рядом с файлом созданной webстраницы.

Параметры текста

Из параметров текста документа в большей степени применяется только один — TEXT. Он задает цвет основного текста на странице (значение параметра может быть введено аналогично цвету фона документа как с помощью словесного задания цвета, так и в шестнадцатеричном представлении или раскладке цветов rgb):

```
<BODY TEXT="black">
```

Параметры гиперссылок

С помощью тега <BODY> возможно также задать, как будут выглядеть гиперссылки на странице. За это отвечают следующие параметры (таблица 6).

Таблица 5. Параметры гиперссылок.

Параметр	Функция	Примеры
ALINK	Устанавливает цвет активной ссылки. (Ссылка становится активной, когда на нее нажимают курсором мыши или выделяют с помощью клавиатуры).	<BODY LINK="#0000FF" ALINK="#0000FF" VLINK="blue">

Параметр	Функция	Примеры
LINK	Определяет цвет непосещенных ссылок	
VLINK	Определяет цвет посещенных ссылок, т.е., таких ссылок, на которые пользователь уже «нажимал».	

Параметры границ документа

Эти параметры определяют отступы от края документа до содержимого страницы. Значения задаются в пикселях, значение по умолчанию 10 пикселей. Синтаксис употребления параметров представлен в таблице 7.

Таблица 6. Параметры границы документа

Параметр	Функция	Примеры
TOPMARGIN	от верхнего края	<BODY TOPMARGIN="5" BOTTOMMARGIN="5" LEFTMARGIN="10" RIGHTMARGIN="10">
BOTTOMMARGIN	от нижнего края	
LEFTMARGIN	от левого края	
RIGHTMARGIN	от правого края	

Например,

```
<BODY TOPMARGIN="5" BOTTOMMARGIN="5"
LEFTMARGIN="10" RIGHTMARGIN="10" >
```

Итак, мы рассмотрели основные разделы любого HTML-документа. Можем приступать к практике.

☒ Вопросы для самоконтроля

1. Из каких основных элементов состоит web-страница и в скольких экземплярах они могут быть представлены?
2. Какие параметры можно задать для фона страницы?
3. Что такое «Мета-теги» и для чего они нужны?

4. Как создать web-страницу?

§ 2.2. Работа с текстом

Текст — единственный объект Web-страницы, который не требует специального определения. Иными словами, произвольные символы интерпретируются по умолчанию как текстовые данные.

Но, тем не менее, работая с текстом, на веб-странице стоит помнить ряд правил:

□□ О расстановке пробелов и переносе строк.

□ Сколько бы ни стояло пробелов между словами при написании кода страницы, это никак не повлияет на конечный вид текста. На странице в браузере будет отображен лишь один.

Это же правило относится к символам табуляции и переносу текста — в браузере они не видны. Поэтому не ставьте лишних пробелов, поскольку это лишь увеличит общий объем файла, но никак не изменит вид документа в браузере.

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

□□ О переносах в тексте.

□ HTML не поддерживает расстановку переносов в словах, иначе говоря, все слова пишутся целиком без их разбиения. Это неудобство наиболее явно проявляется при выравнивании текста по ширине.

□□ Текст размещается по всей ширине окна браузера.

□ Если в тексте есть пробелы или дефисы, браузер автоматически разобьет его на строчки так, чтобы текст уместился в окне

□ Если же в тексте нет знаков пробела или дефиса, браузер отобразит ее в одну строку, добавив горизонтальную полосу прокрутки, если она шире окна браузера.

Для форматирования текста существует большое количество всевозможных тегов, составляющих две основные группы — теги логического форматирования и теги физического форматирования.

Теги физического форматирования позволяют разработчику HTML-документа визуально изменять вид текста, варьируя его параметры и значения. Они предназначены для выделения отдельных текстовых фрагментов различными способами, установленными автором документа.

В группу тегов логического форматирования входят теги, отображающие на экране монитора элементы документа, таким образом, как установлено по умолчанию в спецификации языка разметки HTML. Переопределить их параметры или свойства нельзя, за исключением ситуаций использования стилевых шаблонов CSS и обособления тегами физического форматирования. Основное предназначение этих тегов - логическое выделение отдельных элементов HTML.

Рассмотрим подробнее наиболее используемые теги из каждой группы.

Теги физического форматирования

Все теги видимого (физического) форматирования текста можно разделить в свою очередь на два класса. Первые, называемые часто тегами структурного форматирования, позволяют определить структуру документа, названия основных его элементов. Вторые же, в свою очередь, отвечают за выделение отдельных слов, фраз, предложений.

Структурное форматирование. Абзацы, отступы и переносы.

Как известно, любой текст имеет свою структуру. Книги разделены на части, главы и разделы. Газеты и журналы имеют отдельные рубрики и подзаголовки, которые, в свою очередь, включают фрагменты текста, также имеющие свою собственную внутреннюю структуру — абзацы, отступы, параграфы и пр.

Текст, размещенный в HTML-документе, не исключение, он также должен иметь логичную, понятную каждому пользователю структуру. Ведь от того, насколько просто и удобно будет восприниматься текст на

Web-странице, зависит многое, прежде всего то, какое впечатление о HTML-документе сложится у посетителя[11].

Структурное форматирование в HTML подразумевает разбиение текстовых фрагментов электронного документа на логические блоки с информацией, которым соответствует определенный формат: абзац, текстовый блок, центрирование, отступы и перенос строки, горизонтальный разделитель, предварительно отформатированный текст и комментарии.

Разбиение текста на отдельные абзацы, выражающие законченную мысль, - один из первых принципов составления любых документов. HTML-документы не являются исключением из этого правила. При создании документов с помощью текстовых редакторов разбиение на абзацы выполняется вводом символа перевода строки. Большинство редакторов реализует это при нажатии клавиши <Enter>. Как уже было сказано выше, в HTML-документах символы перевода строки не приводят к образованию нового абзаца[11].

Для обозначения абзаца используется специальный тег <p>, который разделяет фрагменты текста вертикальным отступом. Внутри этого тега могут находиться только теги форматирования текста (логические и физические).

Например, код, представленный ниже, демонстрирует эту возможность. Результат работы можно увидеть на рисунке (Рисунок 3).

```
<html>
<head>
  <title>Применение абзацев</title>
</head>
<body>
<p>Позови меня с собой, </p>
<p>Я пройду сквозь злые ночи, </p>
<p>Я отправлюсь за тобой, </p>
<p>Что бы путь мне ни пророчил. </p>
<p>Я приду туда, где ты</p>
<p>Нарисуешь в небе солнце, </p>
<p>Где разбитые мечты</p> <p>Обретают
снова силу высоты. </p>
```

```
<p> Татьяна Снежина</p>
</body></html>
```

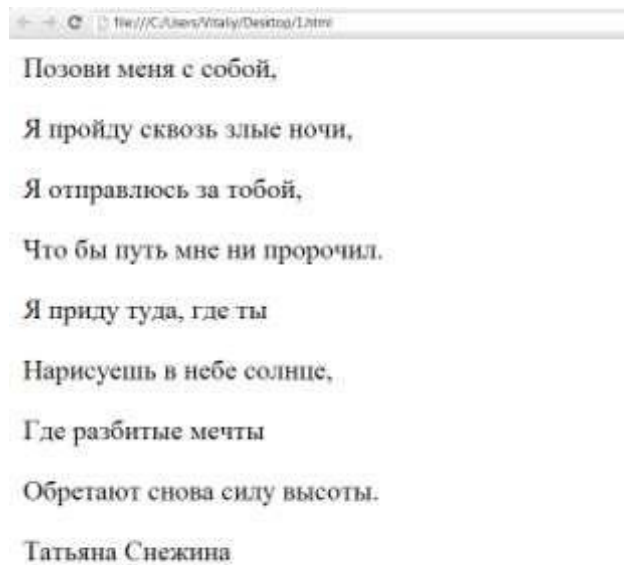


Рисунок 3. Результат отображения представленного кода в браузере.

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

Итак,

```
<body>
<p>Позови меня с собой, <br>
Я пройду сквозь злые ночи, <br>
Я отправлюсь за тобой, <br>
Что бы путь мне ни пророчил. <br>
Я приду туда, где ты<br>
Нарисуешь в небе солнце, <br>
Где разбитые мечты<br> Обретают
снова силу высоты.
</p>
<p> Татьяна Снежина</p>
</body>.
```

Результат представлен на рисунке (Рисунок 4).

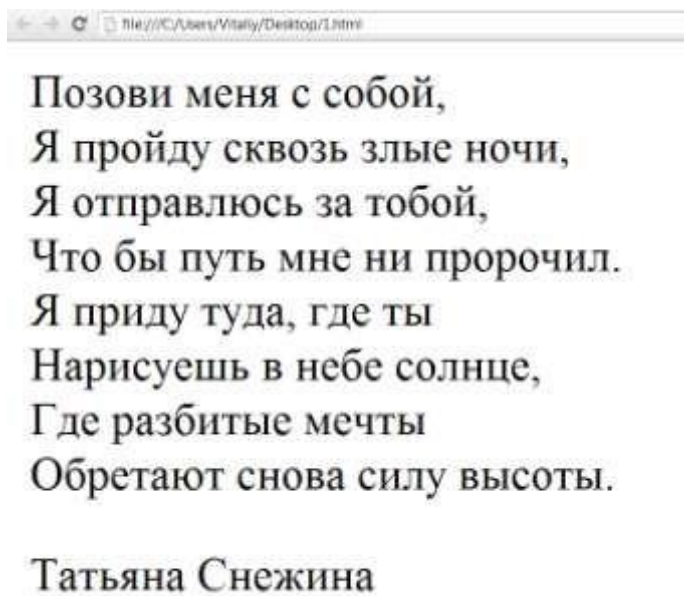


Рисунок 4. Обновленная страница

Возникают ситуации, когда необходимо выровнять текст не по левому краю, как по умолчанию выполняет браузер, а например, по центру или как в нашем примере по правому краю для указания автора строк. Для этого возможно использование тега `<p>` с параметром `ALIGN`.

Существуют следующие значения параметра `ALIGN`, отвечающего за тип горизонтального выравнивания текста в окне браузера.

- `ALIGN="left"` — текст выровнен по левому краю (значение параметра, принятое по умолчанию).
- `ALIGN="center"` — текст располагается посередине окна браузера. Не рекомендуется при работе с большими фрагментами текста, т. к. из-за различной длины слов восприятие абзаца затрудняется;
- `ALIGN="right"` — выравнивание текста по правому краю. Идеально подходит для создания эпиграфов, подписей, заголовков и пр. При работе с большими текстовыми фрагментами нежелательно;
- `ALIGN="justify"` — выравнивание по ширине окна браузера. Это значение является общепринятым стандартом оформления

текстовых блоков делового содержания. Рекомендовано к использованию.

Таким образом, второй абзац должен принять следующий вид:

```
<p align="right"> Татьяна Снежина </p>
```

Часто во многих печатных и книжных изданиях мы можем наблюдать, что первая строка абзаца начинается после небольшого отступа. В текстовом редакторе эта возможность регулируется клавишей табуляции «Tab» или перемещением ползунка линейки настроек рабочей области документа. Четыре следующих элемента позволяют внести некоторую определенность в формат абзаца HTML-документа.

□ Отступы.

В HTML размер отступа определяется кодовой конструкцией ` `, которая визуально представляет собой обычный пробел. Создавая текст в документе, не обязательно между каждым словом вставлять ` `, — браузер и так поймет, что нужно сделать отступ. Но если вам потребуется сделать отступ большего размера, и вы прямо в коде создадите несколько пробелов, браузер интерпретирует такие пропуски в виде единственного пробела.

□ Перевод строки `
`.

Как мы уже рассматривали, `
` используется для принудительного перевода строки. Этот тег один из немногих, не является парным и поэтому не имеет соответствующего закрывающего тэга. В месте его размещения строка заканчивается, а оставшийся текст печатается с новой строки. Например, такой подход может использоваться для создания структур типа списков без использования специальных тэгов разметки списка или как в нашем примере для оформления стихотворения. Для переноса на конкретное число строк тег ставится нужное количество раз.

□ Запрет перевода строки `<NOBR>`.

Тег этот парный и по своему действию является прямой противоположностью предыдущего. Текст, размеченный этим тэгом, будет гарантированно располагаться в одной строке, независимо от ее длины. Если при этом получающаяся строка будет выходить за пределы окна просмотра браузера, то появится горизонтальная полоса прокрутки. Использовать его следует крайне осторожно и только в случае крайней необходимости.

□ Мягкий перевод строки.

Размечая текст с помощью тэга неразрывной строки <NOBR> можно получить очень длинные строки. Чтобы этого избежать, можно указать браузеру место возможного перевода строки, что будет выполнено только при необходимости. Это можно осуществить, поставив в нужном месте текста тэг <WBR> (Word BReak), который так же, как и тэг
, не нуждается в закрывающем тэге.

Выделение отдельных слов, фраз, предложений.

Второй класс тегов физического форматирования позволяет задать акцент в отображении того или иного текста. Часть из тегов не рекомендуется к использованию спецификацией HTML 4.0 по целому ряду причин, или же вообще отменены, однако они продолжают поддерживаться браузерами. В таблице ниже (Таблица 7) приведены данные теги, а также рекомендуемые к использованию за место них.

Таблица 7. Выделение отдельных текстовых фрагментов.

Тег	Функция	Рекомендуется
 	Отображение полужирным шрифтом.	
<I> </I>	Отображение курсивом	, <VAR> <CITE>,
<TT> </TT>	Отображение моноширинным шрифтом	<CODE>, <SAMP>, <KBD>.

К тегам, работающим с размерами текста относят следующие (Таблица 8):

Таблица 8. Теги, управляющие размером шрифта.

Тег	Функция
Тег	Функция
<BIG></BIG>	Отображение текста шрифтом большего (чем непомеченная часть текста) размера.
<SMALL> </SMALL>	Отображение текста шрифтом меньшего размера
	Сдвиг текста ниже уровня строки и вывод его (если возможно) шрифтом меньшего размера. Удобно использовать для математических индексов.
	Сдвиг текста выше уровня строки и вывод его (если возможно) шрифтом меньшего размера. Удобно использовать для задания степеней чисел.

Одним из основных тегов физического форматирования текста, отображающий свойства шрифтов является . Тег является парным, для него могут использоваться следующие параметры (Таблица 9).

Таблица 9. Параметры тега .

Параметр	Функция
FACE	<p>Служит для указания типа шрифта, которым программа просмотра пользователя будет выводить текст.</p> <p>Значение параметра – название шрифта, которое должно в точности совпадать с названием шрифта, имеющего у пользователя.</p> <p>Если такого шрифта не будет найдено, то данное указание будет проигнорировано и будет использован шрифт, установленный по умолчанию.</p> <p>Можно указать как один, так и несколько названий шрифтов, разделяя их запятыми. Список шрифтов просматривается слева направо.</p> <p> Пример задания названия шрифта. Или же Пример задания названия шрифта.</p>

SIZE	Служит для указания размеров шрифта в условных единицах от 1 до 7. Конкретный размер шрифта зависит от браузера, по умолчанию – значение 3. Размер шрифта указывается как абсолютной величиной (SIZE=2), так и относительной (SIZE=+1).
COLOR	Служит для указания цвета шрифта. Значения могут быть заданы словами, числом в шестнадцатеричной системе, тройкой чисел RGB
Параметр	Функция
	текст будет написан красным цветом.

Например,

```
<FONT FACE="Tahoma", "Helvetica" COLOR="000000"
SIZE="2">Этот текст будет показан одним из 3-х
указанных шрифтов, черным цветом и размером
"2".</FONT>
```

В завершение разговора о тегах физического форматирования стоит добавить, что разные теги могут использоваться одновременно в отношении одного текстового фрагмента. Например, часть текста и даже отдельного слова может быть написана и курсивом, и жирным начертанием, да еще подчеркнута и т. д. Единственное, о чем следует помнить, это сохранение последовательности указания закрывающих тегов. Другими словами, не рекомендуется, открыв фрагмент текста тегами и , закрывать HTML-конструкцию в том же порядке (т. е. порядок закрывающих тегов должен быть обратным) – должен работать эффект «Матрешки».

Теги логического форматирования

Что мы делаем, когда хотим обратить внимание читателя на что-то важное? Правильно, выделяем важное курсивом. Или полужирным. Или подчеркиваем. Или еще как-нибудь.

Но страница, где встречается сразу несколько вариантов выделения, — и подчеркивание, и курсив, и еще что-нибудь — выглядит, мягко говоря, любительски, или даже неопрятно. Избежать беспорядка будет

гораздо проще, если вместо разнообразных вариантов выделения использовать логическую разметку (Таблица 10).

Таблица 10. Теги логического форматирования

Тег	Функция	Отображение на экране
	От английского слова «EMphasis» — постановка ударения, выделение, подчеркивание. Используется для интонационного выделения определенного текстового фрагмента.	курсив (так же, как теги <CITE> и <I>).
	Используется для выделения фрагментов текста, на которых необходимо акцентировать внимание пользователя. Тэгом обычно размечают более важные фрагменты текста, чем те, что размечены тэгом 	жирное начертание
<H1>, <H2>... <H6>	<H></H> с указанием одной из цифр (от 1 до 6) задает определенный размер заголовка всего текста целиком или его конкретного фрагмента.	Заголовок <H6> будет минимальным<H1> — самым большим. Подразумевается отступ от текстовой части.

Вы, наверное, обратили внимание на то, что некоторые элементы дают одинаковый результат. Более того, часть элементов может никак не изменять представление фрагмента текста на экране. Может возникнуть законный вопрос: для чего создано такое разнообразие элементов форматирования?

Ответ — в названии этой группы элементов. Они предназначены для расстановки логических ударений, выделения логических частей и подчеркивания сути высказываний. Их использование весьма актуально, поскольку, вероятно, в ближайшем будущем возможности браузеров возрастут, например, будет возможен поиск цитат на Webпространстве и т.п. Кроме того, авторам документов ничто не мешает уже сегодня,

применяя таблицы стилей, задать желаемое отображение для любого из тэгов, переопределив значения по умолчанию.

Так, например, следующие средства позволяют создать у посетителя впечатление интенсивной работы над обновлением страницы. Что обычно является первым признаком того, что над бумажным документом много и плодотворно поработали? Измятая бумага? Конечно, можно имитировать потертости, вмятины и жирные пятна за счет соответствующей графики... Но сейчас не об этом. Многочисленные исправления! Их-то можно без особого труда внести и в электронный документ. Да, но ведь на то он и электронный, чтобы даже после основательной переделки не содержать помарок! Их остается только имитировать. Для этого предназначены теги, представленные в таблице (Таблица 11).

Таблица 11. Теги имитации «бурной» деятельности.

Тег	Функция	Отображение на экране
	Используется для обозначения удаленного текста. Имеет необязательные для указания параметры DATETIME (дата удаления - YYYY-MMDDThh:mm:ssTZD, часовой пояс Time Zone) и CITE (URL-адрес документа - источник причины удаления). <DEL DATETIME=2013-1029T16:12:53+0.00>3.2	Зачеркнутый
Тег	Функция	Отображение на экране
<INS> </INS>	Используется для обозначения вставленного текста. Имеет необязательные для указания параметры DATETIME (дата вставки - YYYY-MMDDThh:mm:ssTZD, часовой пояс Time Zone) и CITE (URL-адрес документа - подробности внесенных дополнений).	подчеркивание

HTML придумал программист... Именно такая мысль приходит в голову, когда встречаешь в HTML целый «букет» средств для описания различных программных кодов и результатов их работы. Впрочем, использовать эти средства по назначению вовсе не обязательно. Кстати, есть такое правило: если в тексте — например, по физике или математике — встречается переменная, то она выводится курсивом.

Теги, представленные в таблице ниже (Таблица 12) позволяют это сделать.

Таблица 12. Теги для оформления кодов и листингов.

Тег	Функция	Отображение на экране
<CODE></CODE>	Используется для форматирования небольшого фрагмента программы	моноширинный шрифт
<SAMP></SAMP>	Используется при иллюстрации, примеров (sample), вывода данных на экран.	моноширинный шрифт
<VAR> </VAR>	Используется для выделения имен переменных (variable).	моноширинный шрифт
<PRE></PRE>	Используется для выделения больших фрагментов (листингов) кода.	моноширинный шрифт
<KBD> </KBD>	Предназначен для выделения текста, вводимого с клавиатуры (keyboard).	моноширинный шрифт в полужирном начертании

Есть одно различие в использовании тэгов <CODE> и <PRE>. В коде программ часто бывает важно наличие нескольких идущих подряд пробелов. Их отображение будет сохранено только при использовании тэга <PRE>, об этом мы уже ранее говорили.

Сокращения.

Вам, вероятно, знакома ситуация: вы читаете текст, изобилующий сокращениями, их значение разъясняется где-то далеко вначале, искать все это долго, а иначе непонятно, о чем речь...

В Web-документах такая проблема легко решается с помощью следующих тегов (таблица 14).

Таблица 13. Задание аббревиатур.

Тег	Функция	Отображение на экране
<ACRONYM> </ACRONYM> <ABBR></ABBR>	Используется для расшифровки аббревиатур. Реализуется через параметр TITLE.	Значение параметра выводится на экран при наведении на интересующее слово

```
<ACRONYM TITLE="HyperText Markup Language"> HTML
</ACRONYM>
```

Цитаты.

Согласно правилам русского языка, цитаты заключаются в кавычки, в электронных публикациях они добавляются также самостоятельно, как если бы мы писали ручкой на бумаге. Но, кроме кавычек, цитаты обычно выделяются форматированием.

Для коротких — не больше одного предложения — цитат, как правило, достаточно курсива. Более длинные — на несколько предложений — цитаты часто выделяют в отдельный абзац, чтобы текст лучше воспринимался.

За работу с цитатами отвечают следующие теги (таблица 15):

Таблица 14. Теги для работы с цитатами

Тег	Функция	Отображение на экране
<CITE> </CITE>	Используется для разметки	курсив, визуально
Тег	Функция	Отображение на экране
	коротких цитат, высказываний, названий	аналогичен тегам и <I>

	библиографических источников и пр.	
<BLOCKQUOTE> </BLOCKQUOTE>	Используется для разметки больших цитат. Имеет параметр CITE, в качестве значения которого можно указать источник цитаты.	отдельный абзац с отступом
<Q> </Q>	Используется для разметки коротких цитат. Имеет параметр CITE, в качестве значения которого можно указать источник цитаты.	курсив

Центрирование.

Центрирование любых элементов HTML-документа может быть осуществлено с помощью тега <CENTER>. Все данные, размещенные внутри этого парного тега, подлежат горизонтальному выравниванию посередине окна браузера. В сущности, тег <CENTER> представляет собой аналог значения ALIGN="center" того тега, результат отображения которого будет отцентрирован на экране монитора. Полезно использовать этот тег для центрирования таких элементов, как, например, таблиц, где неприменимо назначение ALIGN="center".

Горизонтальные линии.

Другим методом деления документа на части является проведение горизонтальных линий <HR>. Они визуальнo подчеркивают законченность той или иной области страницы. Часто используют рельефную, вдавленную линию, чтобы обозначить «объемность» документа.

Тег не требует парного закрывающего. До и после линии автоматически вставляется пустая строка.

Основными параметрами этого тега являются (таблица 16):

Таблица 15. Параметры тега <HR>

Параметр	Функция
Параметр	Функция

ALIGN	Выравнивает по краю или центру; имеет значения LEFT,CENTER, RIGHT
WIDTH	Устанавливает длину линии в пикселях или процентах от ширины окна браузера
SIZE	Устанавливает толщину линии в пикселях
NOSHADE	Отменяет рельефность линии
COLOR	Указывает цвет линии. Используется формат RGB или стандартное имя цвета.

<HR ALIGN=CENTER WIDTH=50% NOSHADE>

Комментарии.

Если горизонтальная линия является визуальным разделителем фрагментов текста, то комментарии невидимы на экране монитора, но в то же время могут нести большой объем информации.

Для чего же они нужны? Во-первых, они позволяют размечать структуру HTML-кода, давая заголовки различным логическим блокам элементов. Впоследствии такая разметка поможет быстро сориентироваться и найти необходимый фрагмент кода. Во-вторых, в комментариях можно указать информацию об авторском праве и прочие персональные данные, которые будут проиндексированы поисковыми системами.

Любой HTML-комментарий должен начинаться с конструкции <!-- и заканчиваться -->. Между ними может находиться любой текст, цифры, символы и прочее, за исключением тегов.

```
<!-- Начало блока новостей --> Код
блока новостей...
<!-- Окончание блока новостей -->
```

При всем удобстве включения в HTML-код комментариев необходимо помнить, что просмотреть код электронного документа может практически любой пользователь Интернета, открыв Webстраницу в текстовом редакторе. Поэтому размещать в комментариях информацию конфиденциального, коммерческого или личного характера не рекомендуется.

Текстовые блоки <DIV> </DIV> и

В случае необходимости указания специальных свойств отдельному фрагменту текста используются теги текстовых блоков. Изменение свойств осуществляется посредством назначения выбранному фрагменту текста стиля CSS, например:

```
<DIV STYLE="COLOR: GRAY;">Фрагмент текста, набранный  
серым цветом </DIV>
```

Между <DIV> и имеются существенные отличия.

- <DIV> является исключительно структурным тегом, а берет начало в области физического форматирования текста[10].
- <DIV> создает принудительный перенос строки на одну позицию после своего закрывающего тега (в отличие от тега абзаца <P>, который осуществляет перенос на две позиции), поэтому задавать с его помощью отдельные свойства фрагмента внутри абзаца нельзя — это вызовет принудительный перенос строки.

Возможно добавление параметра типа выравнивания. Применяется в случае, когда выделенному фрагменту текста необходимо присвоить определенные свойства, а к помощи никакого логического тега прибегнуть нельзя.

Позднее мы еще остановимся на данных тегах.

☒ Вопросы для самоконтроля

1. Чем отличаются теги физического и логического форматирования?
2. Что такое мягкий перенос строки и когда его применяют?
3. Могут ли теги быть вложенными друг в друга? Что при этом стоит помнить?
4. Для чего нужны комментарии?

§ 2.3. Специальные символы

В ходе создания того или иного HTML-документа часто бывает необходимо отобразить в браузере символы, используемые в спецификации языка HTML: угловые скобки, обозначающие теги (< >), знак амперсанда (&) и т. д. Использование таких символов, непосредственно введенных в HTML-документ, будет интерпретировано совсем не так, как задумал автор. Чаще всего это случается при добавлении в исходный документ фрагмента HTML-кода, который впоследствии должен отображаться в окне браузера именно как кодовый листинг, а не результат его исполнения.

Тогда нам на помощь приходят специальные кодовые комбинации. Эти коды состоят из символа амперсанда (&) и следующим за ним именем символа или его десятичным или шестнадцатеричным значением. Заканчиваться специальный символ должен знаком «точка с запятой».

В спецификации HTML приводятся целые таблицы со специальными символами и их значениями[3,10,13]. На сегодняшний день таблицы общедоступны, поэтому приводить их полностью излишне. Отметим лишь некоторые символы, употребление которых актуально.

Условно все специальные символы HTML можно разделить на три большие категории:

- символы, отображающие элементы HTML-форматирования (таблица 17);
- символы оформления документа (таблица 18); □ буквы иностранных алфавитов (таблица 19).

Причем для каждого специального символа существуют два способа записи — кодовый и числовой. К примеру, конструкции § и § совершенно одинаково будут выводиться в окне браузера в виде значка параграфа. Первый вариант задается путем указания амперсанда (&), знака диеза (#) и числового кода, а второй с помощью специальной кодовой конструкции (знак амперсанда (&) и соответствующее наименование).

Символы, отображающие элементы HTML-форматирования.

Таблица 16. Символы HTML-форматирования.

Запись специального символа	Назначение
<	Знак «меньше»
>	Знак «больше»
&	Амперсанд
"	Знак «кавычки»

Символы оформления документа

К так называемым символам оформления HTML-документов относятся значки авторского права, зарегистрированной торговой марки, символы иностранных валют, математические обозначения («плюс-минус», «умножить» и др.), а также прочие символы. Наиболее употребляемые из них представлены в таблице 18.

Таблица 17. Символы оформления документа.

Запись специального символа	Назначение
®	Зарегистрированная торговая марка
©	Знак copyright – (авторских прав)
± ±	Знак «плюс-минус»
§ §	Знак параграфа
×	Знак «умножить»
«	Левая кавычка «елочка»
»	Правая кавычка «елочка»
 	Пробел
-	Дефис
­	Мягкий перенос
← ←	Стрелка влево
↑ ↑	Стрелка вверх
→ →	Стрелка вправо
↓ ↓	Стрелка вниз
↔ ↔	Стрелка влево-вправо

Зная конструкции символов оформления, можно существенно облегчить процесс создания HTML-документов. Так, если необходимо разместить в HTML-документе некоторые математические данные, то существенно лучше будет выглядеть уравнение, где знак «умножить» задан с помощью специальной кодовой конструкции, нежели там, где в место него поставлена буква «х».

Буквы иностранных алфавитов

Когда создается HTML-документ на русском или английском языке, никаких затруднений с отображением букв в браузере, разумеется, возникнуть не может. Все они есть на раскладке клавиатуры компьютера. Представим ситуацию, в которой необходимо разработать электронный документ, к примеру, на французском языке. Алфавит данного языка содержит некоторые буквы, начертание которых характерно только лишь для данного языка. Или столь популярные в математическом языке α , β , γ , π .

Как можно выйти из такого положения? Существует несколько вариантов. Первый заключается в том, чтобы добавить в раскладку клавиатуры новый язык, включить в используемом текстовом редакторе поддержку французского и, выбрав на Панели задач значок подключения языка, спокойно набирать текст. Но как быть с греческими буквами? Добавить еще один язык или того хуже – сделать их картинками. Но мы забыли о конечной цели ввода такого текста — быть интерпретированным, причем корректно, браузером.

Лучший вариант подразумевает использование специальных кодовых или числовых конструкций, предназначенных для корректной визуализации нестандартных букв иностранного алфавита.

Таблица 18. Буквы иностранных алфавитов.

Запись специального символа	Назначение
α	Строчная буква альфа
β	Строчная буква бета
γ	Строчная буква гамма
π	Строчная буква пи
λ	Строчная буква лямбда

☑ Вопросы для самоконтроля

1. Можно ли один и тот же символ задать разными способами?
2. Какие символы относят к символам оформления документа?
3. Как задать значок авторских прав?
4. Как задать значок параграфа?
5. Можно ли добавить на страницу символу греческого алфавита?

§ 2.4 Графика

Возможность использования графики трудно переоценить в приложении к любому виду публикации, в том числе и для Webдокументов. Без иллюстраций документ однообразен, вял и скучен. Расчетливо подобранная и правильно размещенная в документе графика делает его визуально привлекательнее и, что самое важное, передает одну из основных идей документа.

Изображения могут сделать текст вашего документа более содержательным. Представьте некий сухой HTML-документ, содержащий, например, описание придуманного вами технического устройства. Если он состоит из одного текста, то многим покажется скучным и порой непонятным. А если его «разбавить» несколькими иллюстрациями, размещенными в нужных местах, документ станет более читабельным и визуально привлекательным[11].

Изображения помогают лучше передать суть и содержание документа. Все мы как дети любим книжки с картинками. Ведь не даром народная мудрость гласит «лучше один раз увидеть, чем сто раз услышать» (в данном случае — прочитать).

Однако во всем нужно чувство меры. Это правило лишний раз подтверждается при просмотре ряда Web-страниц. Довольно часто встречаются Web-документы, загроможденные фоновыми изображениями, ничего не выражающей графикой и раздражающей анимацией. Планируя разместить на своей странице то или иное

изображение, убедитесь, что оно действительно необходимо. Если при просмотре печатных материалов вам не составит труда перевернуть страницу, то для Web-документов часто приходится дожидаться окончания его загрузки с тем, чтобы двинуться дальше. Загроможденность графикой также плохо, как и полное ее отсутствие.

Дело усугубляет наличие рекламы в Интернет, которая появляется на страницах многих сайтов в виде привлекательных графических плакатов (рекламных баннеров), которые обычно размещаются до основной содержательной части документов. Обычная реакция пользователей на рекламу в Web-документах точно такая же, как и на рекламу, вставляемую посреди вашей любимой телевизионной передачи.

В предыдущих параграфах мы узнали об основных методах построения HTML-документов. Этот параграф дополнит инструментарий средствами включения изображений в страницы, позволяющих сделать их более привлекательными и информативными.

Изображения на Web-страницах могут использоваться двумя способами: в качестве фонового изображения, на котором располагаются элементы основного документа, и изображения, встраиваемые в документ.

Для начала рассмотрим общие вопросы, возникающие на первом этапе работы с графическими изображениями на Web-страницах. Принимая решение о целесообразности включения в документ тех или иных иллюстраций, нужно иметь в виду следующее.

- Графические файлы могут иметь значительные размеры, что требует времени для их загрузки.
- Поисковые системы плохо могут индексировать графику. Поэтому если на ваших страницах расположены только иллюстрации без текстовых пояснений, то читатели такие страницы, скорее всего, не обнаружат.
- Также стоит не забывать, что пользователи могут работать с различным разрешением экрана монитора и различной глубиной цвета. Страницы, хорошо смотрящиеся при одном разрешении, могут выглядеть совершенно по-другому при ином разрешении.

- Еще один пункт, который также не стоит забывать при работе с изображениями – многие из них защищены законом об авторских правах. Публикация изображений без санкции автора может привести к неприятностям.

Способы хранения изображений

Существует много способов описания графической информации, соответственно имеется значительное количество форматов хранения графических файлов, — порядка нескольких десятков.

Все форматы хранения графической информации можно разделить на два типа: векторный и растровый.

Файлы векторной графики содержат математические данные о том, как перерисовать изображение с помощью отрезков прямых (векторов) при выводе его на экран. Процесс вывода требует дополнительной обработки, но такое представление графической информации имеет важное преимущество: масштаб изображения может быть изменен без потери качества, так как не существует фиксированной связи между тем, как он определен в файле и выводом точек на экран.

Векторная графика, как правило, употребляется для изображений с четкими геометрическими формами. Примером ее применения являются системы автоматизированного проектирования (CAD). В векторном виде хранится информация для некоторых типов шрифтов.

Растровая графика предполагает хранение данных о каждой точке изображения. Для отображения растровой графики не требуется сложных математических расчетов, достаточно лишь получить данные о каждой точке и отобразить их на экране. Все наши фотографии, сделанные на современных фотоаппаратах сохраняются именно в растровых форматах. При масштабировании растровой графики обычно происходит потеря разрешения, что ухудшает качество изображения.

На Web-страницах в подавляющем большинстве случаев используется растровая графика в трех форматах: GIF, JPG и PNG. Именно эти форматы непосредственно поддерживаются популярными браузерами, а для использования большинства других графических форматов потребуются специальные средства.

Ранее популярный формат BMP является стандартом MS Windows, однако его употребление не может быть рекомендовано, так как данный формат не поддерживает сжатие данных.

Разработанный недавно формат PNG был призван заменить растровый формат GIF, однако, на данный момент он занял свою нишу качественных изображений с прозрачным фоном. Иные графические форматы (кроме GIF и JPG) в HTML-документах на WWW-серверах практически не встречаются, хотя принципиально это возможно.

GIF (Graphics Interchange Format)

На сегодняшний день стандарт GIF по-прежнему является используемым и популярным в ходе разработки электронных документов. Стандарт был разработан компанией CompuServe Inc. для передачи графической информации в пределах определенных компьютерных сетей (разработка велась еще до появления Интернета).

К положительным качествам формата можно отнести

- ☐ возможность хранения множественных изображений,
- ☐ возможность сделать часть изображения прозрачной,
- ☐ наличие специального алгоритма сжатия данных позволяет подвергать компрессии файлы GIF без изменения качества изображения,
- ☐ способность чередования кадров анимированного GIF-файла приводит к тому, что в пределах одного рекламного носителя возможно разместить большой объем информации,
- ☐ по сравнению со статичными изображениями анимированный GIF-объект привлекает большее внимание со стороны пользователей,
- ☐ поддержка прозрачности позволяет экономить на исходном размере файла.

К недостаткам следует отнести

- ☐ ограниченное количество цветов (не более 256), реализованных в виде палитры 24-битовых цветов,
- ☐ отсутствие возможностей по хранению градаций серого и данных цветовой коррекции,
- ☐ хранению данных в моделях, отличных от RGB (например, CMYK или HSI).
- ☐ Прозрачным может стать только один цвет.

JPEG (Joint Photographic Experts Group)

Данный формат был создан специальной группой экспертов в области фотографии и предназначался для хранения графических изображений с большой глубиной цвета.

В стандарте применен специальный алгоритм компрессии данных — при повышении степени сжатия качество изображения ухудшается за счет изъятия «ненужной» информации (в отличие от алгоритма сжатия GIF, который позволяет производить подобную процедуру практически без потерь).

На сегодняшний день стандарт JPEG занимает лидирующее место по популярности и используется для создания изображений, в композицию которых входят фотографии, сложные коллажи (компьютерный монтаж нескольких разнородных графических объектов), объекты, подвергнутые действию различных графических эффектов и фильтров.

Не останавливаясь на деталях хранения информации в этом формате, отметим, что файлы JPG следует использовать, прежде всего, для хранения фотореалистичных изображений. Ограничение в 256 цветов, присущее GIF, может снизить качество изображения, что исключается при использовании JPG. Поскольку JPG основан на сжатии данных с потерями, учитывающими особенности восприятия изображения человеком, то без значительного ухудшения картинки можно обеспечить значительную степень сжатия и, как следствие, небольшой размер файла.

Различают две схемы хранения — обычная и прогрессивная (progressive). Прогрессивная схема хранения такова, что при выводе таких изображений создается впечатление постепенного проявления рисунка на экране с всё большим уточнением отдельных деталей.

Файлы формата JPG, в отличие от файлов формата GIF, не могут иметь несколько изображений, которые будут при просмотре сменять друг друга.

Кроме того, для них нет возможности назначить прозрачный цвет. Недостатком формата является возможность искажения цвета в результате сжатия данных.

PNG (Portable Network Graphics)

Стандарт разрабатывался с учетом особенностей Интернета. PNG выбрал в себя наиболее сильные стороны двух предыдущих стандартов и исключил их недостатки.

В стандарте реализованы следующие средства:

- прозрачный фон; □ строчное чередование; □ сжатие без потерь и др.

Однако PNG, несмотря на свои преимущества, не получил пока такого признания пользователями Интернета, как стандарты GIF и JPEG (одна из возможных причин — отсутствие поддержки анимации). В Сети можно встретить графические файлы с расширением png, но достаточно редко.

Какой формат предпочесть — GIF, PNG или JPG?

Итак, резюмируем, в каких случаях предпочтительнее использование формата GIF, а в каких — JPG и PNG.

Формат GIF лучше всего подходит для следующих типов изображений:

- изображений с ограниченным количеством используемых цветов;
- изображений, имеющих четкие границы и края, что свойственно большинству изображений типа меню, кнопок и графиков;
- изображений, в состав которых входит текст.

Формат JPG больше подходит для хранения следующих изображений:

- фотографий, полученных со сканера или цифровой камеры, а также фотореалистичных изображений, построенных на основе компьютерных расчетов (ray-tracing rendering);
- графики со сложным сочетанием цветов и оттенков;
- любое изображение, которое требует более 256 цветов.

Формат PNG стоит применять при работе

- со слоями и другими элементами со сложными краями

□ с фотореалистичными изображениями с неограниченным количеством используемых цветов

Эти рекомендации не носят абсолютного характера и могут не приниматься во внимание, тем более что не всегда можно провести строгое разграничение между многоцветными фотореалистичными изображениями и рисованной графикой в стиле «line art». Преобразование GIF в JPG и PNG может ухудшить качество изображения за счет алгоритма сжатия с потерями. Размер файла при таком преобразовании несложных рисунков может даже увеличиться. Преобразование JPG, PNG в GIF ограничит палитру цветов до 256 и в подавляющем большинстве случаев приведет к увеличению размера файла.

Фоновые изображения

Напомним, что для задания цвета фона употребляется параметр BGCOLOR тэга <BODY>, а фоновое изображение включается в документ при помощи параметра BACKGROUND. В качестве значения параметра BGCOLOR указывается название цвета или его составляющие в шестнадцатеричном коде. В качестве фонового изображения должен использоваться графический файл формата GIF или JPG и PNG.

Часто используемым вариантом является фоновое изображение в виде бледного рельефного логотипа. Такая графика ясно идентифицирует сайт и не мешает восприятию материала.

Приведем пример записи тэга <BODY> с указанием фонового цвета и фонового изображения:

```
<BODY BACKGROUND=texture.jpg BGCOLOR=gray>
```

На первый взгляд может показаться, что указание фонового цвета излишне при задании фонового изображения. В действительности все наоборот. Можно рекомендовать всегда указывать цвет фона документа, если задается фоновое изображение. Дело в том, что при загрузке документа, прежде всего, отображается текстовая часть, а на следующем проходе будут загружаться изображения, в том числе и изображение,

используемое в качестве фонового. До момента загрузки и отображения фонового изображения цвет фона документа будет определяться значением параметра BGCOLOR или устанавливаться по умолчанию. До загрузки фонового изображения порой проходит определенный промежуток времени, в течение которого пользователь знакомится с уже загруженным текстом. В какой-то момент проявляется фоновое изображение, изменяя гамму цветов документа. Чтобы предотвратить резкое изменение гаммы цветов, следует задавать значение цвета фона близким к цветам фонового изображения.

Встраивание изображений в HTML-документы

Для встраивания изображений в HTML-документ следует использовать непарный тэг . Данный тэг может иметь ряд параметров.

Самый главный из них – SRC (от английского «source» – «ресурс»). Это единственный параметр, который является обязательным для указания, который выполняет важную роль в графическом изображении на странице — он задает путь (относительный или абсолютный) к рисунку, то есть определяет URL-адрес файла с изображением.

Формат указания следующий:

```
<IMG SRC="picture.gif">
```

Заметим, что в данном случае браузер станет искать файл «picture.gif» в том же каталоге, где находится и HTML-документ, ссылающийся на этот рисунок. Обычно для графических изображений выделяется отдельная папка (в нашем случае images) :

```
<IMG SRC="images/picture.gif">
```

Замечено, что самыми частыми ошибками, приводящими к тому, что изображение не отображается в окне браузера, являются:

- не правильное написание параметра SRC. Уж очень часто любят менять местами последние две буквы.

- Не правильное указание расширения графического файла. Например, jpg вместо jpeg.
- Не правильное написание имени файла с точки зрения регистра. picture.gif и picture.GIF для большинства браузеров – разные файлы.

Если и Ваше изображение куда-то затерялось, проверьте, может это и ваш случай.

Задание размеров выводимого изображения

Единственным обязательным параметром тега является, как уже было описано выше, SRC. Использование остальных параметров не обязательно, однако рекомендуется по ряду причин.

Так, параметры WIDTH и HEIGHT:

- позволяют во время загрузки изображения сразу зарезервировать на странице столько места, сколько это необходимо для отображения рисунка. Многие разработчики пренебрегают этим правилом, в результате чего при загрузке под рисунок изначально отводится слишком мало места, а потом страница начинает «скакать», поскольку браузер одновременно пытается в это пространство вместить реальные размеры файла;
- поместить рисунок в окне браузера, не прибегая к помощи полос прокруток, которые автоматически возникают, если размеры изображения слишком велики. В таком случае, достаточно в HTML-коде указать пропорционально уменьшенные размеры этого рисунка. В этом случае браузеры автоматически при загрузке изображений выполняют его перемасштабирование. Заметим, что неаккуратное задание параметров может привести к изменению пропорций рисунка и, как следствие, к его искажению.

Значения параметров WIDTH и HEIGHT могут указываться как в пикселах, так и в процентах от размеров окна просмотра. Последний

вариант дает команду браузеру растянуть или сузить рисунок в соответствии с размерами окна страницы.

```
<IMG SRC="images/clock.gif" BORDER="1" WIDTH="300"
HEIGHT="200">
```

Любой из этих параметров может быть опущен. Если задан только один из параметров, то при загрузке рисунка второй параметр будет вычисляться автоматически из условий сохранения пропорций.

Рекомендуем все-таки указывать действительные размеры. Вопервых, это позволяет читателю, работающему в режиме отключения загрузки изображений, иметь представление о размерах иллюстраций по пустому прямоугольнику, выдаваемому на экран вместо изображения (если размеры не будут указаны, то браузер, не зная их, выведет маленькую пиктограмму и форматирование страницы будет нарушено). А во-вторых, это ускоряет верстку документа на экране. Обычно браузеры должны загрузить все встроенные изображения прежде, чем отформатировать текст на экране. Указание размеров встроенных изображений позволяет выполнить форматирование документа до полной загрузки файлов с изображениями.

Выравнивание изображений и отделение их от текста

При включении графического изображения в документ можно указывать его расположение относительно текста или других элементов страницы. Способ выравнивания изображения задается значением параметра ALIGN тэга . Возможные значения этого параметра приведены в таблице 20.

Таблица 19. Значения параметров ALIGN

Значение параметра ALIGN	Действие параметра
TOP	Верхняя граница изображения выравнивается по самому высокому элементу текущей строки
TEXTTOP	Верхняя граница изображения выравнивается по самому высокому текстовому элементу текущей строки

MIDDLE	Выравнивание середины изображения по базовой линии текущей строки
ABSMIDDLE	Выравнивание середины изображения посередине текущей строки
BASELINE или BOTTOM	Выравнивание нижней границы изображения по базовой линии текущей строки
ABSBOTTOM	Выравнивание нижней границы изображения по нижней границе текущей строки
Значение параметра ALIGN	Действие параметра
LEFT	Изображение прижимается к левому полю окна. Текст обтекает изображение с правой стороны
RIGHT	Изображение прижимается к правому полю окна. Текст обтекает изображение с левой стороны

Поясним действие параметров выравнивания, приведенных в таблице 7. Все значения параметров выравнивания изображений можно условно разделить на две группы по их принципу действия.

К одной группе относятся два значения параметра — LEFT и RIGHT. При использовании любого из этих параметров мы получаем так называемое «плавающее» изображение, прижимающееся к соответствующему краю окна просмотра браузера. При этом последующий текст (или другие элементы) «обтекают» изображение с противоположной стороны. Здесь текст, размещаемый рядом с изображением, может занимать несколько строчек. Для принудительного прекращения обтекания в заданном месте текста используется тэг принудительного прерывания строки
 с параметром CLEAR.

В качестве значений параметра CLEAR можно использовать следующие: LEFT, RIGHT или ALL. Например,

```
<BR CLEAR=right>
```

Текст, следующий далее, будет размещаться ниже изображения с новой строки.

К другой группе значений параметров относятся параметры, отвечающие за расположение изображения относительно строки текста,

куда оно встраивается. В отличие от плавающих изображений, здесь изображение является обычным элементом строки, только большим наподобие буквы.

Возникает вопрос, в чем разница между базовой линией и нижней границей строки или между самым высоким элементом строки и самым высоким текстовым элементом строки? Результат действия этих параметров может отличаться в зависимости от содержимого рассматриваемой строки.

Базовая линия (BASELINE или BOTTOM) — это нижняя часть линии текста, которая проводится, например, букв типа j, q, y. В отличие от выравнивания по базовой линии, при задании значения ABSBOTTOM выравнивание выполняется по нижней части самого низкого элемента в строке, т. е. по одному из символов строки, имеющему элементы, лежащие ниже базовой линии.

Аналогично обстоит дело с различием между параметрами TOP и TEXTTOP. Например, самым высоким элементом в строке может быть графическое изображение, в то время как самым высоким текстовым элементом строки является, как правило, заглавная буква.

При «обтекании» и «встраивании» изображения в текст последний размещается чересчур близко от края изображения. Это, пожалуй, не совсем красиво. Было бы лучше отступить на пару миллиметров.

Как это сделать? Конечно, можно, открыть Photoshop и внести соответствующие изменения в графический файл: собственноручно нарисовать рамку, добавить отступы... Но уйдет много времени, да и размер графического увеличится, а значит и время загрузки.

Гораздо проще задать отступы в HTML-коде. Для того в теге можно задавать параметры HSPACE и VSPACE, значения которых определяют отступы от изображения, оставляемые пустыми, соответственно, по горизонтали и вертикали. Это гарантирует, что между текстом и изображением останется пространство, необходимое для нормального восприятия.

```
<IMG SRC=arh.gif ALIGN=left HSPACE=20 VSPACE=20>
```


Необходимо обратить внимание на то, что значение, указанное для параметров HSPACE и VSPACE, устанавливается с обеих сторон графического изображения.

Рамки вокруг изображений

Изображение, встраиваемое на страницу, можно поместить также в рамку различной ширины. Для этого служит параметр BORDER. В качестве значения параметра используется число, означающее толщину рамки в пикселях. По умолчанию рамка вокруг изображений не рисуется (BORDER =0).

```
<IMG SRC="images/clock.gif" BORDER="5">
```

Исключением из этого правила является случай, когда изображение является ссылкой. В этом случае браузер автоматически отобразит вокруг рисунка рамку толщиной в 1 пиксель (кроме этого, некоторые браузеры делают рамку определенного цвета, обычно синего).

Поэтому, если никакой необходимости в рамке вокруг графического указателя ссылки нет, следует дать браузеру соответствующую инструкцию:

```
<A HREF="clock.html"> <IMG SRC="clock.gif"  
BORDER="0"> </A>
```

Альтернативный текст и подсказки

Для того чтобы дать пользователям неграфических браузеров или пользователям, работающим в режиме отключения загрузки изображений, возможность получить некоторую текстовую информацию о встроенных изображениях используется параметр ALT, определяющий альтернативный текст. При отключенных изображениях вместо них на экране появится текст, определенный значением параметра ALT. Значение этого параметра имеет смысл и для случаев, когда загрузка изображений будет выполняться. Поскольку загрузка изображений выполняется на втором проходе после отображения текстовой

информации, то изначально на экране на месте изображения появится альтернативный текст, который по мере загрузки будет сменяться изображением. Например, для рисунка `arh.gif`, демонстрирующего символику Архангельска, будет уместным добавить альтернативный текст «Герб Архангельска».

```
<IMG SRC=arh.gif ALT="Герб Архангельска" WIDTH=150  
HEIGHT=174>
```

При этом бесспорно, что для прозрачной распорки, размещенной в пустой табличной ячейке, совершенно бессмысленно писать что-то вроде «Прозрачная графическая распорка».

Параметр же `TITLE` предназначен для отображения всплывающих подсказок к рисункам. В отличие от `ALT` при незагруженном изображении он не предоставит пользователю никакой информации.

☑ Вопросы для самоконтроля

1. Какие форматы изображений применяются для размещения на интернет-страницах? В каких случаях стоит применять тот или иной формат?
2. Какие способы размещения изображений на странице существуют?
3. Почему рекомендуется задавать реальные размеры для изображений?
4. Можно ли избавиться от синей рамки у картинки-ссылки?
5. В чем отличие параметров `TITLE` и `ALT`?

§ 2.5. Списки

Давайте задумаемся — как часто мы сталкиваемся в нашей повседневной жизни с перечнем какой-то информации? Если грядет череда новогодних праздников и необходимо совершить множество всевозможных покупок для праздничного стола — мы составляем список всех необходимых продуктов. Планируя свой день, мы указываем в

ежедневнике перечень дел, мероприятий и встреч. Очутившись в кафе, мы знакомимся с предложенным меню, состоящим также из списка наименований тех блюд и напитков, которые можем заказать.

Таким образом, различные списки играют далеко не последнюю роль в нашей повседневной жизни. Возникает вопрос — а почему мы так часто прибегаем к этому средству представления информации?

Во-первых. Информация в виде списка позволяет разбить большие массивы данных на отдельные, четкие фрагменты, которые человеку гораздо удобнее воспринимать, нежели весь поток целиком.

Во-вторых. Списки позволяют создавать понятную и логичную внутреннюю структуру информационных данных, ориентироваться в которой – дело простое и удобное.

В-третьих. Использование списков удобно для отображения определенных пошаговых и прочих последовательных процессов[11].

Даже сейчас мы с Вами обратились к одному из списков. При работе с текстом в различных программах типа Microsoft Word создание списков не представляется сложным делом — надо всего лишь указать область данных, которую текстовому редактору необходимо превратить в список.

Автоматическое создание списков — очень удобная функция текстовых процессоров и причина первых восторгов у многих новичков. Это понятно: вместо того чтобы следить за нумерацией и отступами, можно уделить больше внимания логическому построению перечня и другим более важным вещам. Программа сама расставит нужные цифры, проследит за их форматированием. Кроме того, вместо плюсов и черточек, которые используют в нумерованных списках машинистки, можно подобрать приличествующие случаю кружки, «птички» и т.п.

Как создать список в HTML? Конечно, можно поступить просто, «по-машинистски», т.е. как секретарь-машинистка: самому ввести цифры и маркеры, лично проследить за отступами. Получится почти как на пишущей машинке. Но, это неудобно, и очень велика вероятность ошибиться.

Есть другой, более простой способ. В языке разметки HTML за организацию списков отвечает целый ряд тегов. Достаточно лишь определить нумерованный, маркированный или так называемый «список определений» перед нами.

Нумерованные списки

Одним из типов списков, реализованных в языке HTML, является нумерованный (упорядоченный) список. Последнее название произошло от формального перевода названия соответствующего тэга ``, с помощью которого и организуются списки такого типа в HTML-документах (OL — Ordered List, упорядоченный список). Списки данного типа обычно представляют собой упорядоченную последовательность отдельных элементов. Отличием от маркированных списков является то, что в нумерованном списке перед каждым его элементом автоматически проставляется порядковый номер. Вид нумерации зависит от браузера и может задаваться параметрами тэгов списка.

Каждый элемент нумерованного списка должен располагаться в отдельной паре тегов `... ` (List Item — элемент списка). Хотя последнее время его применяют и как одиночный.

Открывающий и закрывающий тэги списка `` и `` обеспечивают отступ до и после списка, отделяя, таким образом, его от основного содержимого документа.

Итак, для создания списка ингредиентов праздничного салата достаточно написать следующий код:

```
<OL>
<LI>морковь
<LI>2 яйца
<LI>картофель
<LI>крабовое мясо
</OL>
```

При отображении в браузере мы увидим стандартный нумерованный список.

А если вдруг нам захочется список отобразить не арабскими цифрами, а например, римскими, и нумерацию с трех – то и это тоже возможно.

Для задания номера начала списка используется параметр `START`.

По умолчанию нумерация списка всегда начинается с единицы (`start=1`). Для того чтобы она начиналась, например, с нуля (как любят

программисты), нужно указать в теге параметр start=0. В качестве значения параметра START всегда должно указываться натуральное число, записанное арабскими цифрами вне зависимости от вида нумерации списка.

Попробуем теперь заменить нумерацию с арабской на римскую. Для этого нам понадобится параметр type. Он может принимать одно из пяти значений: A, a, I, i или 1, которые соответствуют типам нумерации (таблица 21):

Таблица 20. Типы нумерации

Тип	Значение TYPE
большими латинскими буквами (A, B, C, ...)	A
малыми латинскими буквами (a, b, c, ...)	a
большими римскими цифрами (I, II, III, IV, ...)	I
малыми римскими цифрами (i, ii, iii, iv, ...)	i
арабскими цифрами (1, 2, 3, ...)	1

По умолчанию используется значение TYPE = 1, т. е. нумерация при помощи арабских цифр. Это касается и вложенных нумерованных списков.

К сожалению, в HTML пока нет способа автоматической разметки списка, пронумерованного кириллицей: а, б, в, и т.д. Как нет и способа изменить символ, стоящий после буквы или цифры: это всегда точка.

Изменение вида нумерации списка и значений номеров допустимо производить и для любого элемента списка. Тэг для нумерованных списков разрешает использовать параметры TYPE и VALUE. Параметр TYPE может принимать такие же значения, как и для тэга .

Пример записи:

```
<LI TYPE = A>
```

Значение параметра VALUE тэга позволяет изменить номер данного элемента списка. При этом изменяется нумерация и всех последующих элементов. Типичным применением являются списки с пропуском некоторых элементов.

Т.е. при помощи параметра value, если использовать его для первого в списке элемента LI, можно добиться того же эффекта, что и при помощи параметра start.

Маркированные списки

Как известно, кроме нумерованных списков, т.е. таких, где важен порядок следования элементов, есть такие, где важен только их перечень. Вместо цифр или букв в них используют маркеры — точки, черточки, кружки. Такие списки называют маркированными или ненумерованными (неупорядоченными).

Для разметки маркированных списков в HTML применяется тот же принцип, что и для нумерованных списков, только вместо дескриптора используется дескриптор . Как нетрудно догадаться его название происходит от английского «unordered list» — «неупорядоченный список».

Что получится, если в рассмотренном выше примере заменить дескриптор на ? Нумерация исчезнет, а вместо цифр в начале каждой строки появятся характерные жирные точки.

Заметим, что кроме элементов списка, отмечаемых тэгом внутри тега , могут присутствовать и другие HTML-элементы, например, обычный текст, не являющийся пунктом списка, а играющий роль его заголовка.

В тэге может быть указан аналогичный параметр TYPE, используемый для задания типа маркера.

Конкретный вид маркера будет зависеть от используемого браузера. Типичными вариантами отображения являются следующие:

- TYPE = disc - маркеры отображаются закрашенными кружками;
- TYPE = circle — маркеры отображаются не закрашенными кружками;
- TYPE = square — маркеры отображаются закрашенными квадратиками. Например,

```
<UL TYPE = circle>
```

Значением, используемым по умолчанию, является TYPE = disc.

Для вложенных маркированных списков на первом уровне по умолчанию используется значение disc, на втором - circle, на третьем и далее — square.

Также как и у нумерованных списков, параметр TYPE с теми же значениями может употребляться для указания вида маркеров отдельных элементов списка. Для этого необходимо соответствующее значение указать в тэге элемента списка .

```
<LI TYPE = circle>
```

Итак, для задания книг в домашней библиотеке в виде кружков необходимо написать

```
<UL TYPE = circle>
<li> М. Цветаева. Избранное
<li> А.А. Фадеев. Разгром
<li> Л.Н. Толстой. Анна Каренина
<li> И.Л. Гончаров. Обрыв
</UL>
```

Если же Вам недостаточно указанных выше трех видов маркеров, то можно задать и так называемые «графические маркеры» с элементами-картинками. Это открывает широчайший простор для творчества дизайнеров.

Для указания браузеру того, что в качестве маркера будет использовано графическое изображение, внутри тега-контейнера вместо указателя элемента размещается обыкновенная HTML-конструкция описания графических объектов, т.е. .

```
<UL>
<IMG SRC="marker.gif">Corel DRAW<BR>
<IMG SRC="marker.gif">Adobe Illustrator<BR>
<IMG SRC="marker.gif">Macromedia Free<BR>
</UL>
```

Браузер поймет такую конструкцию как команду задать отступ от левого края документа до начала элементов маркированного списка (где повторяющимся маркером является файл рисунка marker.gif)[11].

В данном случае после каждого пункта маркированного списка стоит тег переноса строк
. Если его не указывать, все элементы списка будут выстроены в один ряд. Это связано с тем, что создание графических маркеров не относится к каким-то особым HTML-конструкциям, а реализовано за счет обособления рисунков-маркеров тегом . Поскольку тег элемента списка отсутствует, браузер не делает переноса строки.

Вложенные списки

Бывают случаи, когда в элемент списка одного типа требуется включить целый список такого же типа или другого. При этом будут организованы многоуровневые или вложенные списки. В HTML допустимо произвольное вложение различных типов списков, однако при их организации следует проявлять аккуратность. Здесь действует все тот же «принцип матрешки» вложенности тегов.

В этом нашем примере в каждый элемент маркированного списка вложен свой нумерованный список.

```
<UL>
<B>Спутники некоторых планет
</B>
<LI>Земля
<OL>
<LI>Луна </OL>
<LI>Марс
<OL>
<LI>Фобос
<LI>Деймос
</OL>
<LI>Уран
<OL>
<LI>Ариэль
```



```
<LI>Умбриэль  
<LI>Титания  
<LI>Оберон  
</OL>  
</UL>
```

В качестве совета добавим, что при составлении сложных вложенных списков особое внимание следует уделять корректному указанию закрывающих тегов `` и ``. Для того чтобы не запутаться в разветвленной структуре HTML-кода вложенных списков, каждый последующий уровень вложенности лучше всего отделять от предыдущего небольшим отступом от левого края документа.

Списки определений

Об этом виде списков вспоминают достаточно редко. В большинстве случаев, когда хочется сделать красиво, воображение «зацикливается» на выборе того или иного маркера. Как будто единственное, от чего зависит красивый и аккуратный дизайн, — это «птички» или кружочки!

На самом деле достаточно часто — гораздо чаще, чем, кажется на первый взгляд — можно использовать еще один вид списков. Это так называемые списки определений. Эти списки строятся по принципу: сначала короткая фраза или слово (как правило, выделенное полужирным шрифтом или как-нибудь иначе), потом более подробное разъяснение. Такие фрагменты можно организовать по-разному, в зависимости от общего вида страницы: можно как обычные абзацы, а можно — как списки определений[11].

Весь список определений заключается внутрь парного дескриптора `<DL>` (от английского *definition list* — «список определений»), каждый термин — внутрь дескриптора `<DT>` (от английского *definition term* — «определяемый термин») и каждое описание — внутрь дескриптора `<DD>` (от английского *definition description* — «описание определения»). Вот и все, что необходимо сделать.

Итак, список определений записывается следующим образом:

```
<DL>
<DT>Термин
<DD>Определение термина
</DL>
```

В тексте после тэга <DT> не могут использоваться элементы уровня блока, такие как, например, тэги абзаца <p> или заголовков <H1>—<H6>. Как правило, текст определяемого термина должен располагаться в одной строке.

Текст, содержащий определение термина, выводится, начиная со следующей строки (или через строку для некоторых браузеров) после определения термина с отступом вправо. В информации, помещенной после тэга <DD>, могут располагаться элементы уровня блока. Отсюда следует, в частности, что списки определений могут быть вложенными.

Приведем пример HTML-документа, в котором использован список определений:

```
<DL>
<CENTER>
<H3>Классификация типичных темпераментов человека,
основанная на воззрениях Гиппократ</H3></CENTER>
<DT>Флегматик
<DD>Пассивный, очень трудоспособный, медленно
приспосабливающийся; настроение устойчивое, мало
поддается внешнему влиянию; вялость эмоциональных
реакций и медлительность в волевой деятельности
<DT>Сангвиник
<DD>Активный, энергичный, легко
приспосабливающийся; живость и подвижность
эмоциональных реакций, быстрота и сила волевых
проявлений <DT>Холерик
<DD>Активный, очень энергичный, настойчивый;
порывистость и сила эмоциональных реакций, бурные
волевые проявления
<DT>Меланхолик
<DD>Пассивный, легко утомляющийся, тяжело
приспосабливающийся; слабость волевых проявлений и
```

преобладание подавленного настроения, неуверенность в себе
</DL>

Отображение приведенного HTML-документа в браузере приведено на рисунке (Рисунок 5).

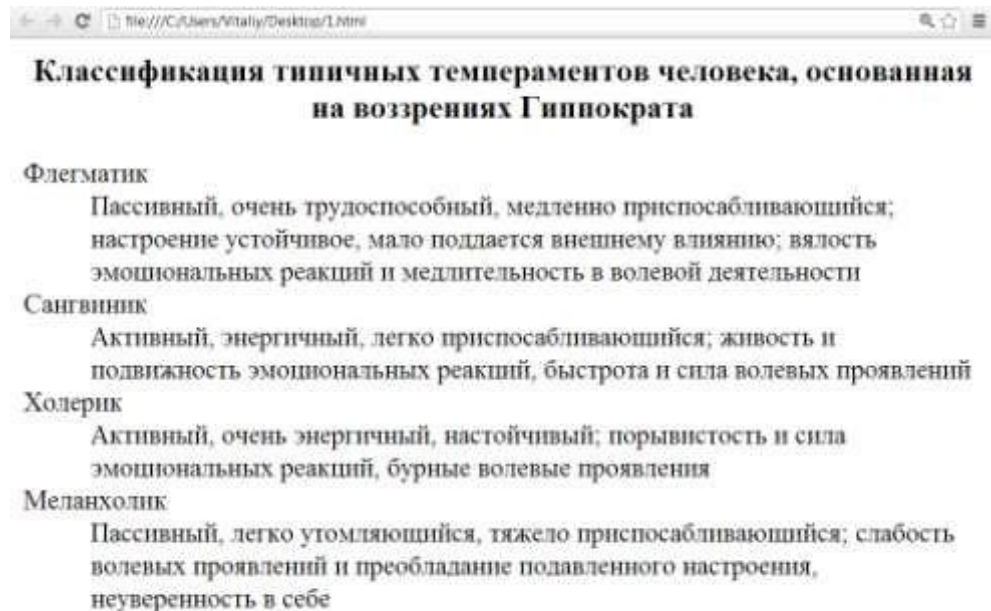


Рисунок 5. Список определений

☒ Вопросы для самоконтроля

1. В каких случаях применять списки определений?
2. Позволяет ли HTML работать с вложенными списками?
3. Что может выступать в роли маркера списка?
4. Как задать список, нумеруемый римскими цифрами?

§ 2.6. Табличное представление данных

Одним из наиболее мощных и гибких средств представления информационных данных в HTML по праву являются таблицы. В повседневной жизни, решая те или иные задачи, мы часто сталкиваемся с таблицами. Однако в HTML таблицы не ограничиваются удобным средством структурирования информации. Сегодня таблица остается

основой структуры web-страницы., содержащей внутри себя самые разнообразные элементы HTML.

Первая версия языка HTML не предусматривала специальных средств для отображения таблиц, так как была в основном предназначена для написания простого текста. С развитием сфер применения HTML-документов стала актуальной задача представления данных, для которых типично наличие ряда строк и столбцов. Создание документов, содержащих выровненные по колонкам данные, на первых порах осуществлялось использованием «преформатированного» текста, внутри которого необходимое выравнивание обеспечивалось введением нужного количества пробелов. Напомним, что текст внутри пары тэгов `<PRE>` и `</PRE>` выводится моноширинным шрифтом, и все пробелы и символы табуляции являются значащими. Работа по выравниванию такого текста выполнялась вручную, что существенно замедляло создание документов. Поддержка табличного представления данных стала стандартом де-факто, поскольку изначально была реализована во всех ведущих браузерах и лишь по прошествии значительного времени была закреплена в спецификации HTML 3.2[8,11].

Создание простейших HTML-таблиц

Рассмотрим сначала минимальный набор тэгов и их параметров, необходимый и достаточный для создания несложных таблиц, а затем перейдем к их детальному описанию.

Описание таблиц должно располагаться внутри раздела документа `<BODY>`. Документ может содержать произвольное число таблиц, причем допускается вложенность таблиц друг в друга.

Каждая таблица должна начинаться тэгом `<TABLE>` и завершаться тэгом `</TABLE>`. Внутри этой пары тэгов располагается описание содержимого таблицы. Любая таблица состоит из одной или нескольких строк, в каждой из которых задаются отдельные ячейки.

Для задания строк используется тег `<TR>` (от английского «Table Row» - строка таблицы), внутри которого расположены ячейки `<TD>` (от английского «Table Data» - данные, или содержимое таблицы). После

того как встретится новый элемент TR, начнется описание следующей строки, и т. д. до конца таблицы (тега </TABLE>).

Завершающие коды </TR>, </TD> могут быть опущены. Однако все же рекомендуется их указывать для предотвращения ошибок, которые могут возникнуть при создании сложных вложенных таблиц.

Завершающий тэг таблицы </TABLE> является обязательным.

Этих сведений вполне достаточно для построения элементарных таблиц.

Приведем пример простейшей таблицы, состоящей из двух строк и двух столбцов.

```
<TABLE>
<TR>
<TD>Ячейка 1 строки 1</TD>
<TD>Ячейка 2 строки 1</TD>
</TR>
<TR>
<TD>Ячейка 1 строки 2</TD>
<TD>Ячейка 2 строки 2</TD>
</TR>
</TABLE>
```

Результат работы представлен на рисунке (Рисунок 6).



Ячейка 1 строки 1	Ячейка 2 строки 1
Ячейка 1 строки 2	Ячейка 2 строки 2

Рисунок 6. Простейшая таблица

«Где же таблица?» - спросите Вы. Таблица есть, только она имеет невидимые рамки (границы). По умолчанию рамки не рисуются, и на экране пользователь увидит лишь ровно расположенный текст ячеек таблицы. Существует немало ситуаций, когда использование таблиц без рамок вполне оправданно, например, для многоколонных списков, реализованных при помощи таблиц, или задания точного взаимного расположения рисунков и текста. Однако в большинстве случаев для

традиционного использования таблиц ее ячейки полезно отделить друг от друга линиями сетки, что облегчает восприятие и понимание информации, содержащейся в таблице.

Отображение рамок и их цвет

Итак, для того чтобы отобразить линии сетки таблицы, и вокруг всей таблицы необходимо указать параметр BORDER для тега <TABLE>. Можно указать просто BORDER или задать ему численное значение.

Например, <TABLE BORDER> или <TABLE BORDER=3>.

Численное значение параметра определяет толщину рамки в пикселях, рисуемую вокруг всей таблицы, однако на толщину рамок вокруг каждой ячейки это значение не влияет. При отсутствии численного значения оно принимается равным 1. Расположить ячейки как можно ближе друг к другу возможно заданием BORDER=0, что означает отсутствие рамок. Это значение и принято по умолчанию. Возможность независимого управления отображением рамки вокруг всей таблицы и рамками вокруг ячеек пока отсутствует.

По умолчанию рамка черно-серая, для того чтобы изменить ее цвет используется параметр BORDERCOLOR. Значение указывается в шестнадцатеричном формате (например, #FFCC00), раскладке rgb (например, rgb(100,200,4)) или в виде наименования (например, green).

```
<TABLE border=15 bordercolor=green>
```

Если в ячейке ничего нет, то рамки вокруг нее тоже не будет, как ни старайтесь. Для того чтобы ячейка выглядела пустой, но имела рамку, нужно «положить» в нее нечто невидимое. Как правило, такими «невидимыми» объектами являются неразрывный пробел тег принудительного переноса строки
 или прозрачный GIF-файл размером 1x1 пиксель.

При использовании этих тегов следует помнить, что размер пустой (в визуальном плане) ячейки будет зависеть от свойств шрифта, заданных для HTML-документа в целом или же предопределенных для

конкретного блока (, <DIV>) или ячейки цветные текстовые блоки. Разместив в ячейке, которая должна визуально казаться пустой, текст небольшого объема цветом, аналогичным значению параметра BGCOLOR тега <TD> или <TH>, при просмотре такого документа в браузере мы получим «пустую» табличную ячейку. При использовании так называемой «графической распорки» искусственно увеличить или уменьшить размеры ячейки можно будет путем переназначения его параметров WIDTH и HEIGHT.

Итак, если мы хотим получить рамку, нужно использовать параметр border, если не хотим — просто пропускаем его. А если хотим, но не везде? Например, как быть, если мы хотим оставить видимыми только вертикальные границы ячеек, как между газетными столбцами?

В теге <TABLE> есть два параметра, позволяющие «поиграть» с отображением разных частей рамок.

RULES

Параметр RULES управляет прорисовкой внутренних линий сетки таблицы. Возможные значения параметра RULES приведены в таблице 22.

Таблица 21. Возможные значения параметра RULES

Значение	Функция
ALL	Отображение линейки целиком
GROUPS	Часть линейки, разделяющая сгруппированные данные
COLS	Часть линейки, разделяющая столбцы
ROWS	Часть линейки, разделяющая строки
NONE	Отсутствие линейки

FRAME

Управление прорисовкой рамок вокруг таблицы осуществляется параметром FRAME. Возможные значения параметра FRAME представлены в таблице 23.

Таблица 22. Возможные значения параметра FRAME.

Значение	Функция
BOX	Рамка с четырех сторон
BORDER	Рамка с четырех сторон

ABOVE	Рамка только сверху
BELOW	Рамка только снизу
LHS	Только левая часть рамки
RHS	Только правая часть рамки
VSIDES	Левая и правая части рамки
ALL	границы ячеек будут видимы
HSIDES	Верхняя и нижняя части рамки
VOID	Нет рамок

Прорисовка линий сетки таблицы и рамок будет осуществляться только при наличии параметра BORDER тэга <TABLE>. При отсутствии этого параметра или его нулевом значении линии сетки и рамки будут отсутствовать при любых значениях параметров FRAME и RULES.

Выравнивание таблицы и элементов в ячейках таблицы

Стандарт HTML позволяет задать горизонтальное выравнивание всей таблицы относительно ширины HTML-документа. За это отвечает аналогичный работе с текстом и картинками параметр ALIGN.

Существует три возможных значения параметра ALIGN:

- align="left" - выравнивание таблицы по левому краю документа. Значение принято по умолчанию и в случае отсутствия параметра ALIGN таблице будет присвоено именно это значение;
- align="right" - выравнивание таблицы по правой границе документа;
- align="center" — центрирование таблицы относительно ширины документа.

Помимо функции выравнивания таблицы по горизонтали документа, параметр ALIGN позволяет получать эффект «обтекания» таблицы текстом аналогично «обтеканию» рисунков.

Для получения такого эффекта нужно выбрать тип выравнивания таблицы (в данном случае, кроме центрирования), назначить ширину

таблицы менее 100% или зафиксировать ее (подробнее об изменении параметра ширины таблицы будет рассказано чуть позже). Текст, который должен огибать таблицу, размещается сразу же после закрывающего тега `</TABLE>`.

Каждую отдельную ячейку внутри таблицы можно рассматривать как область для независимого форматирования. Все правила, которые действуют для управления отображением текста, могут быть использованы для форматирования текста внутри ячейки. Внутри ячейки допустимо использование практически всех элементов HTML, которые могут появляться внутри тела документа `<BODY>`, в том числе тэги, управляющие расположением текста - `<p>`, `
`, `<HR>`, коды заголовков — от `<H1>` до `<H6>`, тэги форматирования символов — ``, `<I>`, ``, `<BIG>`, ``, ``, ``, тэги вставки графических изображений ``, гипертекстовых ссылок `<A>` и т. д.

Задание параметров выравнивания на уровне кода `<TR>` определяет выравнивание для всех ячеек данной строки, при этом в каждой отдельной ячейке строки может быть определены свои параметры, переопределяющие действие параметров, заданных в `<TR>` (по умолчанию `LEFT` для `<TD>` и `CENTER` для `<TH>`).

Параметр `VALIGN` также определяет тип выравнивания содержимого ячеек таблицы, но по вертикали. Он может принимать следующие значения:

- `valign="middle"` — выравнивание посередине ячейки
(значение по умолчанию);
- `valign="top"` — выравнивание по верхнему краю ячейки;
- `valign="bottom"` — выравнивание по нижней границе ячейки;
- `valign="baseline"` — выравнивание по базовой линии
(привязка текста одной строки к единому уровню).

Что произойдет, если для всей таблицы задан один тип выравнивания, а для ячейки — другой? Здесь действует общее правило для одноименных параметров: если значение параметра, определенного

для всей таблицы <TABLE>, не совпадает со значением такого же параметра в <TR>, то для данной строки используется параметр, определенный в <TR>. Аналогичным образом, если значение параметра для некоторой ячейки отличается от значения такого же параметра для содержащей ее строки или для всей таблицы, то приоритет имеет значение, заданное для ячейки.

Размеры таблицы и ячеек

Для того чтобы вид страницы не так сильно зависел от размеров окна браузера, в дескрипторе <TABLE> задают минимальную ширину и высоту таблицы. Для этого используются параметры WIDTH и HEIGHT, соответственно. Габариты задаются как в абсолютных значениях (в пикселях), так и в относительных (в процентах от ширины окна).

```
<TABLE WIDTH="500" HEIGHT="50%">
```

Способ верстки Web-страниц, при котором размеры таблиц задаются в процентах, иногда называют «резиновым» дизайном за то, что при изменении размеров окна такие страницы как бы растягиваются или сжимаются, сохраняя основные пропорции. Размер ячеек таблицы будет уменьшаться/увеличиваться пропорционально заявленному значению в процентах[11,12]. Впрочем, эта мера действенна только до определенной степени. Можно сузить окно так, чтобы в левой части осталось по одному слову в каждой строке, и слова все равно будут «налезать» на таблицу. Что тогда делают Web-дизайнеры? Тяжело вздыхают и... строят новую таблицу, чтобы вписать «строптивый кусок текста» в ее левый столбец.

Задание этих же параметров на уровне <TR> и <TD> позволяет определить размеры конкретных строк или ячеек. А при задании в процентном соотношении – пропорций в отображении столбцов.

Фон таблицы и ячеек

Если фоном таблицы или ячейки служит обычная однотонная заливка, то ее цвет определяется параметром bgcolor. Значением этого

параметра является код цвета — такой же, как при определении цвета шрифта.

А как быть, если хочется «подложить» под текст ячейки более сложный фон, например, с переходами цвета или узором? Для любых видов фоновой заливки, кроме однородного цвета, используются готовые графические файлы, которые подключаются к HTML-странице с помощью параметра `background`.

```
<TABLE BACKGROUND=texture.jpg BGCOLOR=gray>
```

Выбирая фон для таблицы, нужно учитывать следующую особенность. Что произойдет, если изображение окажется больше, чем нужно? Скорее всего, вы сами можете ответить на этот вопрос: ничего хорошего — просто часть картины окажется «обрезанной». А что будет, если картинка меньше области, занимаемой таблицей? В этом случае браузер всегда пытается «размножить» его как мозаику. Последнее правило касается любых фоновых изображений — не только для таблиц, но и для всей HTML-страницы. Об этом мы уже говорили в четвертом параграфе.

Действие данных параметров в отношении тега ряда `<TR>` и тегов ячейки `<TD>` и `<TH>` аналогично типу выравнивания `ALIGN` и `VALIGN`.

Параметр `BGCOLOR` используется для тегов `<TR>`, `<TD>` и `<TH>`, а параметр `BACKGROUND` применим только к тегам ячеек `<TD>` и `<TH>`.

И цвет фона, и фоновое изображение могут быть переопределены как для отдельных строк таблицы, так и для ячеек. Однако здесь следует учесть одну особенность: если фон, заданный для всей таблицы, заполняет также и промежутки между ячейками, то фон, определенный для строк и ячеек, заполняет только внутреннюю часть самих ячеек. При этом промежутки между ячейками (разумеется, если таковые существуют) остаются заполненными тем фоном, который был определен для всей таблицы.

Расстояние между ячейками и внутренние отступы

Для того чтобы понять, что такое расстояние между ячейками рассмотрим внимательнее к Web-таблицам. В отличие от других, знакомых нам таблиц, их рамки двойные — каждая ячейка как бы заключена в собственное «окошко». Между соседними «окошками» обычно имеется некий зазор.

Ширину этого зазора по вертикали и горизонтали определяет параметр CELLSPACING. Значение параметра указывается в пикселях, и по умолчанию равно нулю. При конструкции CELLSPACING="0" ячейки таблицы сольются.

```
<TABLE BORDER="2" CELLSPACING="2">
<TR>
<TD>Ячейка 1</TD>
<TD>Ячейка 2</TD>
</TR>
</TABLE>
```

Внутренним отступом называют расстояние между границей ячейки и границей текста. Не начинать же текст прямо от рамки - это неаккуратно и некрасиво. Только в том случае, если рамка невидима, можно себе позволить сделать это расстояние нулевым, и то не всегда.

Для создания и регулирования отступа между рамкой ячейки и ее содержимым (по вертикали и горизонтали) используется параметр CELLPADDING.

Форма указания значения аналогична параметру CELLSPACING.

```
<TABLE BORDER="2" CELLSPACING="2" CELLPADDING="7">
<TR>
<TD>Ячейка 1</TD>
<TD>Ячейка 2</TD>
</TR>
</TABLE>
```

При значении параметра CELLPADDING, равном нулю, содержимое ячейки будет вплотную прижато к рамке таблицы, что ухудшит восприятие информации (это утверждение еще более актуально

для таблиц с прозрачной рамкой — в этом случае текст соседних ячеек сольется друг с другом).

Расстояния между соседними ячейками и отступы между рамкой ячейки и ее содержимым (как по вертикали, так и по горизонтали) всегда будут одинакового размера, т. к. HTML не позволяет назначать разные значения в пределах данного параметра.

Заголовки таблицы, строк и столбцов. Колонтитулы таблицы

Для заголовка страницы можно использовать логический тег <H1>. Но при желании его можно включить в структуру таблицы — в качестве заголовка — с помощью тега <CAPTION>.

Зачем это может понадобиться? Если некий заголовок является неизменным спутником именно этой таблицы, то гораздо лучше жестко «связать» их, чтобы случайно не потерять при очередном копировании. К тому же, <CAPTION> позволяет описать положение заголовка относительно таблицы. По умолчанию выравнивание задано по центру, если же мы захотим изменить режим выравнивания, то нам следует задать значение уже знакомого параметра align для тега <CAPTION> явно: left или right.

Для перемещения надписи вниз, под таблицу лучше использовать второй параметр тега <CAPTION> — valign. Этот параметр имеет всего два значения — top и bottom. Они определяют положение заголовка соответственно над или под таблицей.

```
<TABLE border=15 bordercolor=red >
<CAPTION ALIGN=LEFT valign=bottom>Заголовок,
располагаемый внизу с выравниванием влево</
CAPTION>
<TR>
<TD>Ячейка 1 строки 1</TD>
<TD>Ячейка 2 строки 1</TD>
</TR>
<TR>
<TD>Ячейка 1 строки 2</TD>
<TD>Ячейка 2 строки 2</TD>
```

```
</TR>
</TABLE>
```

У многих таблиц есть также «шапки» — заголовки строк и столбцов, оформление которых чем-то отличается от остальной таблицы. Конечно, можно задать это форматирование в каждой ячейке отдельно:

```
<TABLE>
<TR>
<TD align=center><B>ССЫЛКИ</B></TD>
<TD align=center><B>НОВОСТИ</B></TD>
</TR>
</TABLE>
```

Но если «заголовочных» ячеек слишком много, то такое занятие может и надоесть. Конечно, выравнивание по центру можно вынести в тег `<TR>`:

```
<TABLE>
<TR align=center>
<TD><B>ССЫЛКИ</B></TD>
<TD><B>НОВОСТИ</B></TD>
</TR>
</TABLE>
```

Но как быть в этом случае полужирным шрифтом или если «шапка» затрагивает не только первую строку таблицы, но и, например, левый столбец?

Для того чтобы разметка таблиц не была слишком утомительна, в HTML предусмотрены специальные теги для «заголовочных» ячеек — `<TH>` (от table header — «заголовок таблицы»). Тег `<TH>` полностью подобен дескриптору `<TD>`, за исключением того, что его содержимое обычно выводится полужирным шрифтом и выравнивается по центру, поэтому его можно использовать в любом месте таблицы вместо `<TD>`, а вовсе не обязательно только для верхней строки или левого столбца.

Браузеры позволяют также использовать и другие дополнительные теги структурирования табличных данных, а именно - `<THEAD>`,

<TBODY> и <TFOOT>. Эти теги предназначены для создания колонтитулов таблицы различных уровней (соответственно для верхнего, основного (содержательного) и нижнего уровней таблицы).

Теги верхнего и нижнего колонтитулов <THEAD> и <TFOOT> могут быть использованы в структуре таблицы лишь единожды, причем для них необязательно наличие закрывающих тегов.

Верхний и нижний колонтитулы функционально очерчивают логические заголовки соответствующего уровня и применимы, в основном, в больших таблицах, не помещающихся в пределах одной страницы электронного документа.

Тег основного колонтитула <TBODY> может встречаться неоднократно в пределах одной таблицы, однако требует своего закрывающего тега.

Основные колонтитулы выполняют функцию, аналогичную тегам группировки, расставляя логические метки по ходу изложения основной содержательной части таблицы.

При использовании новых тэгов появляется возможность более гибко управлять рамками и линиями сетки таблицы.

Слияние ячеек и принудительные переносы в ячейках

Вот, наконец, мы и пришли к одному из самых интересных моментов, связанных с Web-таблицами: как создаются ячейки нестандартного размера, занимающие собой несколько столбцов или несколько строк?

COLSPAN показывает, на сколько ячеек по горизонтали следует расширить <TD> или <TH>, в котором указан данный параметр.

ROWSPAN делает то же самое, но увеличивает область <TD> или <TH> по вертикали. Параметры COLSPAN и ROWSPAN применяются только в тегах ячейки <TD> и <TH>.

При использовании параметров COLSPAN и ROWSPAN особое внимание следует уделять корректному объединению соседних ячеек таблицы, а также своевременному указанию закрывающих тегов ячеек и рядов. Возникновение ошибки может привести к нарушению структуры

таблицы, «заползанию» одних ячеек на другие, перекрытию текста и даже невозможности отобразить таблицу в браузере.

Например, для задания таблицы на рисунке ниже (Рисунок 7) достаточно задать таблицу из двух строк, в верхней строке задать одну ячейку и указать, что необходимо объединить два столбца в ней.

Шапка	
Левая ячейка	Правая ячейка

Рисунок 7. Таблица с объединенными столбцами.

```
<TABLE>
<TR align=center>
<TD COLSPAN=2>Шапка</TD>
</TR>
<TR>
<TD> Левая ячейка</TD>
<TD> Правая ячейка</TD>
</TR>
</TABLE>
```

Наоборот, если мы хотим объединить строки в одну, как в таблице ниже (Рисунок 8), для первой строки задаем две ячейки, для второй из которых указываем, что она «растягивается» на две строки. Во второй строке задаем только одну ячейку – вторая уже была «украдена» первой строкой.

Левая ячейка	Объединенная
Правая ячейка	

Рисунок 8. Таблица с объединёнными строками.

```
<TABLE>
<TR align=center>
<TD> Левая ячейка</TD> <TD
ROWSPAN=2>Объединенная</TD>
```



```
</TR>
<TR>
<TD> Правая ячейка</TD>
</TR>
</TABLE>
```

Параметр NOWRAP запрещает принудительный перенос строки в ячейке или табличном ряде. Не рекомендуется использовать данный параметр во всех ячейках, т. к. это может сильно понизить уровень масштабируемости таблицы (при условии, что значение ширины и/или высоты таблицы указано в процентах). Параметр NOWRAP применим в <TR>, <TD> и <TH>.

Группировка столбцов

При построении таблиц мы можем легко задать единый тип выравнивания для отдельной ячейки и даже целой табличной строки. Задать же единое оформление для столбца становится проблематично, так как столбец в таблице — это последовательность ячеек, располагающихся в разных рядах.

Стандартными средствами HTML нам пришлось бы задавать один и тот же тип выравнивания для отдельно взятой ячейки, формирующей нужный столбец:

```
<TABLE>
<TR>
<TD align=right>Ячейка 1 с выравниванием вправо</TD>
<TD align=center>Ячейка 1 с выравниванием по
центру</TD>
</TR>
<TR>
<TD align=right >Ячейка 2 с выравниванием
вправо</TD>
<TD align=center>Ячейка 2 с выравниванием по
центру</TD> </TR>
</TABLE>
```

Но труд разработчика электронного документа может быть значительно облегчен за счет таких тегов, как `<COL>` и `<COLGROUP>`. Эти теги предназначены для определения свойств отображения столбцов, сгруппированных по конкретному признаку.

Каждый из тегов имеет параметр `SPAN`, задающий количество соседних столбцов подлежащих форматированию[11].

```
<TABLE ALIGN="center" BORDER="2" CELLSPACING="0"
CELLPADDING="5"
WIDTH="100%" HEIGHT="200">
<COLGROUP ALIGN="center" SPAN="2">
<COLGROUP ALIGN="right" SPAN="2">
<TR>
<TD>Ячейка 1</TD><TD>Ячейка 2</TD><TD>Ячейка
3</TD><TD>Ячейка 4</TD>
</TR>
<TR>
<TD>Ячейка 5</TD><TD>Ячейка 6</TD><TD>Ячейка
7</TD><TD>Ячейка 8</TD>
</TR>
</TABLE>
```

Таблица состоит из четырех столбцов, в которых данные сгруппированы по заданному признаку. Первые два столбца имеют тип выравнивания по центру, последние два — по правому краю.

Как можно увидеть из примера для сгруппированных столбцов мы можем задавать единый тип выравнивания с помощью параметра `ALIGN`. Возможные значения и формат записи аналогичны тегам `<TD>` и `<TH>`: по левому краю, по правому краю, по центру.

Помимо этого теги `<COL>` и `<COLGROUP>` могут содержать дополнительный параметр вертикального выравнивания данных — `VALIGN`. Возможные значения и формат записи также аналогичны тегам `<TD>` и `<TH>`: по верхнему краю, по нижнему краю, посередине. К сожалению, работает это не для всех браузеров.

Для того чтобы изменить форматирование, например, второго и третьего столбцов таблицы необходимо пойти на маленькую хитрость

поставить тег `<COLGROUP>` без параметров форматирования, указав лишь, сколько столбцов нужно «пропустить»:

```
<COLGROUP SPAN=K>
```

`<COLGROUP` параметры следующей группы столбцов `>`

Кроме того, возможно задание единой ширины, высоты и цвета ячеек определенного столбца. Для этого для тега `<COLGROUP>` указываем соответственно `width`, `height` и `bgcolor`. Значения этих параметров и правила их присваивания такие же, как для ячеек `<TD>`.

Предположим, нам требуется конкретизировать свойства определенного столбца внутри группы. Например, выровнять новости по верхней границе ячеек, а даты — по нижней. Для этого используется тег `<COL>`, расположенный внутри `<COLGROUP>`, он описывает параметры отдельного столбца, принадлежащего группе. Значения параметров, заданные в тэге `<COL>`, переопределяют значения из тэга `<COLGROUP>`.

Пусть в таблице первые два столбца должны быть выровнены вправо, а третий — посередине, причем все три столбца необходимо объединить в группу. Последующие три столбца также должны быть объединены в группу и иметь выравнивание, аналогичное первой группе.

Для решения этой задачи следует записать такой фрагмент HTMLкода:

```
<TABLE>
<COLGROUP>
<COL SPAN=2 ALIGN=RIGHT>
<COL ALIGN=CENTER>
<COLGROUP>
<COL SPAN=2 ALIGN=RIGHT>
<COL ALIGN=CENTER>
(данные для таблицы)
</TABLE>
```

Очень просто и быстро, не правда ли? Заметим, что в тэге `<COLGROUP>` в данном примере, в отличие от предыдущего,

отсутствует параметр SPAN. Здесь его употребление бессмысленно, так как количество элементов в группе будет определяться следующими за тэгом <COLGROUP> тэгами <COL>. Поэтому любое заданное значение параметра SPAN тэга <COLGROUP> будет переопределено.

Добавление параметра rules="groups" к тегу <table> позволяет рисовать линию только между колонками, созданными с помощью <colgroup>.

Вложенные таблицы

Отдельные ячейки таблицы могут содержать практически любые тэги языка и данные, разрешенные в разделе <BODY> документа. В том числе, внутри ячейки таблицы может быть целиком размещена другая таблица. Такие таблицы называются вложенными.

Правила построения вложенных таблиц ничем не отличаются от создания таблиц одного уровня — используются те же теги и параметры, задаются те же свойства и значения. Единственное, о чем нельзя забывать в ходе создания сложных вложенных таблиц, это:

- каждая таблица последующего уровня размещается внутри тега-контейнера <TD> или <TH> таблицы предыдущего уровня;
- вложенная таблица не может быть создана за пределами вышеназванных тегов ячейки таблицы;
- таблица одного уровня может содержать любое количество вложенных таблиц другого уровня, идущих друг за другом в пределах тега ячейки таблицы верхнего уровня;
- количество тегов таблиц всех уровней должно соответствовать количеству закрывающих тегов этих же таблиц.

Отметим, что не все браузеры, поддерживающие таблицы, правильно отражают сложные таблицы с несколькими уровнями вложенности, поэтому их использование требует осторожности[11,14].

Особенность вложенных таблиц, в отличие от других способов представления данных в электронном документе, позволяет более точно

размещать отдельные элементы страницы относительно друг друга и относительно границ самого документа, отображаемого браузером.

Например, два разнородных блока текста и нумерованный список, размещенные внутри тега <BODY>, невозможно разместить на одном уровне, а тем более на одном уровне со смещением в какую-либо сторону. Использование таблиц с легкостью решает эту проблему, позволяя располагать различные элементы и их комбинации в разных местах документа посредством видимых и невидимых ячеек рядов таблицы.

Вот почему многие HTML-документы создаются на основе таблиц, где в качестве несущей основы берется таблица с невидимыми краями, содержащая вложенные таблицы с разным оформлением, отличающимися значениями параметров.

Подводя итог сказанному, можно выделить следующие преимущества вложенных таблиц:

- гибкая масштабируемость структуры электронного документа в целом;
- широкие возможности позиционирования отдельных элементов страницы;
- многоуровневое представление разнородных информационных данных;
- расширенные оформительские возможности; □ поддержка популярными браузерами.

Подводя итог данной теме, перечислим еще раз, какие параметры и для каких элементов таблицы могут быть применимы (таблица 24).

Таблица 23. Основные параметры тега <TABLE>

Параметр	Функция	Применение
ALIGN	Выравнивание таблицы (содержимого ячейки или ряда) по горизонтали	<TR>, <TD>, <TH>, <TABLE>
VALIGN	Выравнивание таблицы	<TR>, <TD>, <TH>

	(содержимого ячейки или ряда) по вертикали	
WIDTH	Определение ширины таблицы (ячейки или ряда)	<TR>, <TD>, <TH>, <TABLE>
HEIGHT	Определение высоты таблицы (ячейки или ряда)	<TR>, <TD>, <TH>, <TABLE>
BGCOLOR	Указание цвета для фона таблицы (ячейки или ряда)	<TR>, <TD>, <TH>, <TABLE>
Параметр	Функция	Применение
BACKGROUND	Указание рисунка для фона таблицы (ячейки)	<TD>, <TH>, <TABLE>
NOWRAP	Запрет принудительного переноса строки в ячейке или ряду	<TR>, <TD>, <TH>
COLSPAN	Объединение соседних ячеек по горизонтали	<TD>, <TH>
ROWSPAN	Объединение соседних ячеек по вертикали	<TD>, <TH>

☑ Вопросы для самоконтроля

1. Можно ли задать фон отдельно для каждой ячейки?
2. Видна ли граница для таблицы заданной с помощью <table>?
3. Можно ли задать расстояние между ячейками?
4. Для чего применяют группировку столбцов?
5. Чем хороша табличная верстка?
6. В чем особенности работы с таблицами, содержащими объединенные ячейки?

§ 2.7. Гиперссылки

Ну вот, мы и подобрались к основному элементу web-страниц – гиперссылкам. Гипертекстовый документ — именно так иногда называют web-страницы – это документ, содержащий ссылки на другие документы, позволяющие при помощи нажатия кнопки мыши быстро перемещаться от одного документа к другому. Часто подобные ссылки можно увидеть и в файлах помощи современных программных

продуктов. За основу гипертекста взят принцип организации энциклопедических словарей, в которых во многих статьях есть ссылки на другие.

Читая книги, мы часто видим, как автор, раскрывая ту или иную тему, ссылается на другой раздел или вообще на совершенно другой печатный источник. В Интернете размещены миллионы электронных документов, часто похожих по тематике и ориентированных на одну и ту же пользовательскую аудиторию. Поэтому ссылки здесь являются единственной возможностью перейти от одного документа к другому[11].

По сути, любая гипертекстовая ссылка — указатель внутри гипертекстового документа, указывающий на другой документ (связывает с другим документом), который также может быть гипертекстовым. Однако успешный переход по ссылке возможен в двух случаях — если ресурс, на который ссылается документ, существует и если структура гиперссылки верна с точки зрения HTML.

Первый фактор является объективным, не зависящим от нас, т. к. разработчик электронного документа, однажды поставив в нем ссылку на внешний ресурс, может и не знать о том, что этот ресурс прекратил свое существование, или же переместился на другой адрес, или же временно закрыт и пр.

Второй фактор относится к разряду субъективных, так как только от создателя HTML-документа может зависеть, сумеет ли посетитель перейти по ссылке, или она составлена неверно. Чтобы последнего не произошло, рассмотрим структуру и правила описания гипертекстовых ссылок.

Структура гиперссылок

Любая гиперссылка состоит из двух важных частей: указателя ссылки («якоря») и адреса ресурса, на который необходимо осуществить переход. Да-да, якоря. Дело в том, что название тега <A>, с помощью которого создаются гиперссылки, происходит от английского слова anchor — «якорь». В самом деле, тег как бы цепляется за объект, на который указывает ссылка. И вот уже страница надежно связана с этим

объектом, даже если последний находится от нас за сотни серверов и тысячи километров ...

Внешне отличить гиперссылку от обычного текста очень просто: при наведении курсора мыши на ссылку указатель принимает вид руки с указательным пальцем, как бы показывающим, что этот текст содержит гиперссылку. Сама ссылка подчеркивается (в случае если указателем является текст).

В качестве указателя ссылки может выступать текст и графические изображения. Текстовые указатели обычно представляют собой слово или несколько слов, выделенных на экране подчеркиванием. Цвет текстового указателя может регулироваться автором и установками браузера.

В качестве ссылки можно использовать графическое изображение. По принципу действия графические ссылки ничем не отличаются от текстовых. Они не подчеркиваются и не выделяются цветом, а для их выделения браузеры обычно вокруг такого изображения рисуют рамку. Но и от нее можно избавиться – об этом мы ранее уже говорили в § 2.4.

В ряде случаев возможно объединение графики и текста в рамках единого указателя ссылки.

Итак, указатель ссылки описывается парным тегом <A>, а адрес перехода реализован с помощью параметра HREF этого тега. Все, что находится между <A> и , и является указателем ссылки или, как еще говорят, «гиперссылкой».

Значением параметра HREF является путь к тому или иному Интернет - ресурсу. Все гиперссылки разделяют на два типа: внешние и внутренние. Внешние ссылки ведут на другие ресурсы глобальной сети или другие документы одного Web-сайта, а внутренние позволяют посетителю путешествовать в пределах одного HTML-документа.

Указание адреса Интернет-ресурса может быть относительным или абсолютным. Если указываем папку и имя файла – такой путь называется относительным (относительно нашего файла). Если же указываем протокол для передачи файла, имя сервера, на котором расположен файл, и полный путь к нему от корневой директории на сервере – перед нами абсолютная ссылка.

Относительные указатели удобны в использовании. Намного проще вставить только имя файла, а не весь длинный URL-адрес. Они также позволяют перемещать файлы в пределах сервера без больших изменений в межстраничной адресации.

```
<A HREF="2.htm"> Ссылка на страницу 2 со страницы 1</A>
```

Такой формат записи внешней ссылки подразумевает расположение файла 2.htm (на который указывает гиперссылка) в том же каталоге, что и файл 1.htm (с которого будет осуществляться переход). В нашем случае оба файла расположены в каталоге about, находящемся на Web-сайте www.narfu.ru.

Теперь попытаемся усложнить ситуацию и представим, что файл 2.html размещен в корне сайта, а файл 1.html по-прежнему находится в каталоге about. В этом случае запись внешней ссылки с использованием относительного пути к файлу примет следующий вид:

```
<A HREF="../2.html">Ссылка на страницу 2 со страницы 1</A>
```

С помощью относительного пути перехода по ссылке мы велели браузеру подняться на уровень вверх (из уровня файлов, лежащих в about, на уровень размещения самого каталога about). Следует помнить, что внешняя ссылка, путь к которой прописан относительно, может ссылаться только на документы в пределах одного сайта.

При этом относительный указатель будет работать по-другому, если в HTML-документе используется тэг <BASE>. Он располагается в начале документа в разделе HEAD и содержит URL-адрес, относительно которого в документе построена вся адресация. Это указание влияет на любой тэг документа, в котором используется относительная адресация.

Внешние ссылки

Понятие Интернета как глобальной информационной сети подразумевает не только World Wide Web (WWW, Всемирная Паутина), в состав которой входит большинство Интернет - ресурсов. Интернет — это более емкая инфраструктура, включающая в себя различные информационные сервисы, работа которых реализуется с помощью так называемых протоколов — наборов технологических правил взаимодействия документов друг с другом.

Например, WWW работает на основе протокола HTTP (HyperText Transfer Protocol, протокол передачи гипертекста). Помимо этого, существуют такие технологии, как FTP (File Transfer Protocol, протокол передачи файлов), E-mail (служба электронной почты), News (группы новостей), Gopher и др.

Вполне возможна ситуация, когда разработчику HTML-документа понадобится поставить ссылку на другие, отличные от HTTP сервисы. В этом случае структура гиперссылки остается прежней — указатель (текстовый и/или графический) и адресная часть. Однако последняя может меняться в зависимости от выбранной технологии передачи данных[11].

Итак, для внешних ссылок, указывающих на web-страницы, указываем протокол HTTP

```
<A HREF="http://www.yandex.ru/">Ссылка на поисковую систему</A>
```

А, например, столь популярная ссылка на адрес электронной почты, может выглядеть так:

```
<A HREF="mailto:mail@site.ru">Ссылка на адрес электронной почты</A>
```

Если пользователь нажмет на такую ссылку, на его компьютере запустится установленная по умолчанию программа чтения и отправления электронной почты (например, Microsoft Outlook, The Bat!, Netscape Messenger, Eudora и др.).

Кроме того, можно сразу добавить тему письма.

 Ссылка на адрес электронной почты

Такая гиперссылка не только запустит почтовую программу и откроет окно нового письма, но и добавит в поле заголовка электронного сообщения Subject обращение "Здравствуйте!".

Правила составления гиперссылок на различные информационные сервисы представлены в таблице 25.

Таблица 24. Гиперссылки на различные информационные сервисы

Название информационного сервиса	Пример гиперссылки
http — доступ к WWW	http://www.site.ru/
mailto — отправка сообщения по электронной почте;	mailto:mail@site.ru
ftp — доступ к архивам файлов по протоколу передачи файлов	ftp://ftp.site.ru/
file — доступ к файлу на локальном диске;	file//file.site.ru/hit.mp3

Гипертекстовая ссылка далеко не всегда ссылается на гипертекстовый документ, в качестве значения параметра href можно указать ссылку на любой объект.

Например, на сайте творческого конкурса можно поместить анкету для участников, составленную в формате Microsoft Word, на сервере бесплатного программного обеспечения — на архив программы (расширения zip, rar и пр.), на ресурсе, содержащем коллекцию музыкальных файлов, — на композиции в формате mp3.

Организации структуры гиперссылок при этом остается неизменной:

Коммерческий договор

```
<A HREF = "http:// www.narfu.ru/program.zip">Архив  
программы </A>  
<A HREF= "http:// www.narfu.ru/hit.mp3">Музыкальный  
ролик</A>
```

В отношении же загрузки запрашиваемого файла в формате, отличном от HTML, возможны две ситуации:

- браузер поддерживает запрашиваемый формат. Тогда он откроет файл в своем рабочем окне (например, рисунки формата GIF, JPEG и пр.). В некоторых случаях браузер может дополнительно использовать подключаемый программный модуль (plug-in), без которого задача не была бы решена;
- браузер не поддерживает запрашиваемый формат. В этом случае браузер откроет стандартное диалоговое окно с возможными вариантами действий относительно открываемого файла неизвестного формата — «Открыть» этот файл из текущего состояния или «Сохранить» этот файл на диске или «Отмена» действий. При нажатии на «Сохранить» файл загружается в соответствующую директорию, при нажатии на «Открыть» - браузер переадресовывает запрос операционной системе, которая запускает программу, предназначенную для просмотра данного типа.

Внутренние ссылки и закладки

Кроме ссылок на другие документы, часто бывает полезно включить ссылки на разные части текущего документа. Например, большой документ читается лучше, если он имеет оглавление со ссылками на соответствующие разделы. Такое оформление можно встретить на популярном энциклопедическом ресурсе «Википедия» (<http://ru.wikipedia.org/>). Такой подход также бывает полезен, например, при составлении технической документации или юридических договоров, где быстрый переход от одного пункта документа к другому играет первостепенную роль.

Структура внутренней гиперссылки включает две части — сама ссылка и ее именной идентификатор (# и имя элемента, аналогичное значению параметра HREF самой гиперссылки), позволяющий переместиться в нужное место электронного документа.

Например, если вы хотите сделать ссылку на текст определенной главы документа, нужно разместить там указатель и дать ему имя при помощи параметра NAME тэга <A>. При этом параметр HREF не используется, и браузер не выделяет содержимое тэга <A>. Например:

```
<A NAME=chapter_5> </A>
```

Обратите внимание, что в приведенном примере отсутствует содержимое тэга <A>. Обычно именно так и делают, поскольку здесь нет необходимости как-то выделять текст, а требуется лишь указать местоположение. После того как место назначения определено, можно приступить к созданию ссылки на него. Для этого, вместо указания в параметре HREF адреса документа, как это делалось ранее, поместим туда имя ссылки с префиксом #, говорящим о том, что это внутренняя ссылка.

```
<A HREF="#chapter_5">Глава 5</A>
```

Теперь, если пользователь щелкнет кнопкой мыши на словах «глава 5», браузер выведет соответствующую часть документа в окне просмотра.

Действительно, гиперссылка — могучее средство. В сущности, это готовый инструмент, решающий одну из самых важных задач в любой книге, будь она бумажная или электронная, — задачу перехода в нужную точку из оглавления или предметного указателя. В бумажной литературе, последняя задача так и не решена: сначала надо найти по ссылке в предметном указателе страницу, где объясняется нужное понятие, а потом еще нужно как-то отыскать на странице сам термин. Иногда бывает непросто. То ли дело гипертекстовая ссылка! Переход по ней происходит мгновенно и сразу в нужную точку.

В нужную точку? До сих пор мы выяснили только, как перейти на нужную страницу. Конечно, можно разбить электронную книгу так, чтобы каждая глава размещалась в отдельном файле. (Кстати, именно так очень часто и поступают, чтобы уменьшить размер загружаемого файла.) Но как поступить, если мы ищем некий термин, описываемый в середине главы? Понадеяться на встроенную в браузер функцию контекстного поиска?

Конечно, можно поступить и так. Но это значит — заставить посетителя страницы каждый раз вручную вводить искомый термин в окно поиска. Поверьте на слово: ему это надоест, и очень быстро.

Как организовать переход не только на нужную страницу, но в нужную точку страницы? В текстовых редакторах для этого предусмотрен специальный инструмент — закладки. То есть некие именованные метки, по которым можно отыскать соответствующую им точку документа.

В HTML такие закладки тоже есть. По реализации они очень похожи на метки, используемые программистами для переходов внутри программы. Для перехода по такой закладке в HTML-коде требуется создать два объекта.

В «точке выхода» создается закладка, которой присваивается какое-нибудь имя. Это имя может содержать любые латинские буквы, а также цифры. Например, метку для перехода в начало страницы можно назвать `begin`, а для перехода в конец — `end`. В соответствующих точках кода страницы ставятся дескрипторы `<A>` с параметром `name`, которому присвоены эти значения:

```
<!-- начало страницы>  
<A name=top>  
<A name=end>  
<!-- конец страницы>
```

Лучше выбирать меткам осмысленные имена. Браузеру все равно, но вам потом будет проще читать и редактировать код.

Теперь, если мы хотим сослаться не просто на файл, а на конкретное место этого файла, нужно, кроме имени файла, указать имя закладки. По правилам HTML разделителем между этими именами служит знак #:

```
<A href="page2.html#top">Переход в начало страницы  
page2.html</A>  
<A href="page2.html#end"> Переход в конец страницы  
page2.html</A>
```

Обратим внимание: в наших руках теперь средство, позволяющее «перепрыгивать» не только с одной страницы на другую, но между разными точками одной и той же страницы. В последнем случае указание имени файла в ссылке становится излишним, и его можно опустить. Если переход по ссылке происходит в пределах одного файла, достаточно ограничиться указанием только имени метки:

```
<A href="#top"> Переход в начало текущей  
страницы</A>
```

Дополнительные параметры тега <A>

Помимо HREF, тег <A> может содержать еще и другие параметры, например, STYLE, CLASS, NAME, TITLE И TARGET. Первые два параметра используются для создания стилевых шаблонов гиперссылки, третий параметр предназначен для указания имени внутренней ссылки.

Параметр TITLE используется для создания всплывающей подсказки при наведении курсора мыши на гиперссылку. Подсказка может выступать в качестве комментария или описания к ссылке, ведущей на электронный документ:

```
<A HREF="http:// http://ru.wikipedia.org/" TITLE  
="Вики"> Ссылка на Википедию /A>
```

Параметра TARGET позволяет определить назначение перехода по ссылке. Например, нажав на гиперссылку, можно открыть документ в текущей или в отдельной (новой) вкладке. Это в ряде случаев является принципиальным моментом. Принято, что если пользователь находится

в пределах одного сайта (сервера), открывать все ссылки в том же окне. Если же по ссылке будет осуществлен переход на другой сервер – использовать загрузку в новую вкладку.

```
<A HREF="http:// http://ru.wikipedia.org/"  
TARGET="_blank"> Ссылка на Википедию /A>
```

Наличие знака нижнего подчеркивания перед значением blank не является обязательным параметром для большинства популярных браузеров, однако Консорциум W3C настоятельно рекомендует указывать полную конструкцию параметра TARGET.

Если же target=_self, то объект открывается в том же окне, что и документ, содержащий гиперссылку. Этот режим используется по умолчанию, так что указывать его специально не обязательно.

В завершение следует добавить, что параметр TARGET и некоторые другие его значения (помимо _blank) используются в описании так называемой фреймовой структуры, о чем будет рассказано позже.

Итак, систематизируем основные параметры тега <A> для лучшего усвоения материала (табл. 26).

Таблица 25. Основные параметры тега <A>

Параметр	Функция
HREF	Указание адреса перехода по гиперссылке
TARGET	Определение места назначения перехода (текущее/новое окно)
STYLE	Задание стилевого шаблона
CLASS	Указание класса стилевого шаблона
NAME	Именной идентификатор внутренней гиперссылки

☒ Вопросы для самоконтроля

1. Какие гиперссылки бывают?
2. Из каких элементов состоит гиперссылка?
3. Для каких ссылок желательно делать переход в новую вкладку? Как его реализовать?

4. Как реализовать переход к середине определенной страницы с помощью гиперссылок?

§ 2.8. Карты-изображения

Ряд сайтов, особенно географической тематики, для организации ссылок используют так называемые карты-изображения (Imagemap).

Карта-изображение внешне выглядит как обычное встроенное изображение, но при выборе с помощью курсора мыши той или иной области на этом изображении выполняется переход на разные страницы.

Реализация этой возможности предусмотрена языком HTML и позволяет привязывать гипертекстовые ссылки к различным областям изображения. Такой подход нагляднее, чем применение обыкновенных текстовых связей, поскольку пользователь может не читать словесное описание связи, а сразу понять ее смысл по графическому образу.

Карты-изображения предоставляют пользователям дружественный интерфейс для перехода на другие Web-страницы. Чтобы выполнить переход по такой ссылке, следует просто выбрать нужное место на изображении и щелкнуть мышью. Наличие такого развитого графического интерфейса является одним из значительных преимуществ Web-страниц по сравнению с другими ресурсами Интернета. Вместо текстовых меню, пользователи получают наглядное графическое представление информации.

Карты-изображения можно применять в самых разнообразных областях компьютерных технологий. Наиболее распространенными из них являются:

- геоинформационные и картографические системы;
- баннерные рекламные сети и системы электронной коммерции;
- электронный и сотовый банкинг, платежные системы;
- игровые трехмерные и двумерные интернет-ресурсы;
- корпоративные серверы;
- интернет-ресурсы широкого профиля.

Как видно из списка, диапазон применения карт-изображений может охватывать практически все отрасли современных технологий.

Компоненты карт-изображений

Основными компонентами этих карт являются: само изображение, которое будем называть опорным для данной карты-изображения и описание конфигурации активных областей.

Эти изображения могут иметь любой допустимый формат (GIF или JPG). При этом в формате GIF может использоваться прозрачный цвет, а также режим чередования строк. Для того чтобы изображение могло использоваться в качестве опорного для карты-изображения, формально не накладывается никаких дополнительных ограничений. Обычно на изображении указывается, где следует сделать щелчок, чтобы перейти на ту или иную страницу. Существует несколько путей указания границ областей, реализующих различные ссылки. Часто используется рамка или какой-либо иной разделитель.

Конфигурация карты-изображения записывается в виде обычного текста, который в зависимости от используемого формата может быть сохранен в отдельном файле или являться частью HTML-документа. Описание конфигурации содержит координаты для каждой из активных областей изображения, а также URL-адреса, связанные с каждой из этих областей.

Активные области могут иметь форму прямоугольников, кругов и многоугольников. Допускается любая комбинация этих фигур. Также может задаваться одно значение URL-адреса, определенное для случая, когда пользователь выполняет щелчок в пределах изображения, но вне любой из заданных активных областей.

Исторически существовало два варианта реализации картизображений, но сегодня применяют только так называемый клиентский вариант (client-side imagemap). Это название часто используют в виде аббревиатуры CSIM.

Конфигурирование карты-изображения

Клиентский вариант карты-изображения позволяет поместить всю информацию о конфигурации карты в HTML-файле, в который встроено изображение.

Для указания того, что встроенное изображение является опорным для карты, используется параметр USEMAP тэга . Значением параметра USEMAP является ссылка на описание конфигурации карты.

Для описания конфигурации областей карты-изображения используется специальный тэг <MAP>, единственным параметром которого является NAME.

Значение параметра NAME определяет имя, которое должно соответствовать имени в USEMAP. Тэг <MAP> требует закрывающего тэга </MAP>. Внутри этой пары тэгов должны располагаться описания активных областей карты, для чего используется специальный тэг <AREA>.

Необходимо отметить, что описание карты-изображения следует сразу же после указания тега рисунка :

```
<IMG SRC="picture.gif" USEMAP="#mymap">  
<MAP NAME="mymap">  
Координаты активных областей...  
</MAP>
```

Итак, каждый отдельный тэг <AREA> задает одну активную область, при этом активные области могут перекрываться. В случае если некоторая точка относится одновременно к нескольким активным областям, то будет реализована та ссылка, описание которой располагается первым в списке областей.

Параметр SHAPE определяет форму активной области:

- ☐ rect - прямоугольник,
- ☐ circle - круг,
- ☐ poly – многоугольник,
- ☐ default - все точки области.

Описание области default должно располагаться последним в списке активных областей. Если параметр SHAPE опущен, то по умолчанию предполагается значение rect.

Параметр COORDS задает координаты отдельной активной области. Значением параметра является список координат точек,

определяющих активную область, разделенных запятыми. Координаты записываются в виде целых неотрицательных чисел.

Начало координат размещается в верхнем левом углу графического изображения, которому соответствует начальное значение 0, 0. Первое число определяет координату по горизонтали, второе — по вертикали. Список координат зависит от типа области (табл. 27).

Таблица 26. Типы областей и их координаты.

Область	Задаваемые координаты
rect	Координаты верхнего левого и правого нижнего углов прямоугольника.
circle	координаты центра круга и радиус.
poly	вершины многоугольника (до 100 вершин) в нужном порядке. Если последняя и первая точка не совпадают, то при интерпретации данных для этой формы области браузер автоматически их соединит.
default	не требует задания координат.

Параметры HREF и NOHREF являются взаимоисключающими. HREF определяет наличие гиперссылки для данной области и собственно ссылку, а NOHREF – ее отсутствие (не требует значения). Правила записи полностью совпадают с правилами записи ссылок в тэге <A>. Если не задан ни один из этих параметров, то считается, что для данной области не имеется ссылки.

Параметр NOHREF полезно использовать для исключения части активной области. Пусть, например, необходимо создать активную область в виде кольца. Такой тип области не предусмотрен в списке возможных областей, однако он может быть реализован путем задания двух круговых областей. Для этого сначала следует задать область меньшего радиуса и указать в качестве параметра NOHREF. Далее нужно задать область большего радиуса с центром в той же точке и указать нужную ссылку. Тогда область внутри кольца, определенная двумя окружностями различного радиуса, будет иметь необходимую ссылку.

Использование подхода, основанного на взаимном перекрытии областей, позволит строить области весьма разнообразной формы.

Параметр TARGET также используется для указания типа перехода (внешнего или внутреннего), загружаемого по данной ссылке.

ALT позволяет записать альтернативный текст для каждой из активных областей изображения.

Пример задания карты-изображения.

```
<IMG SRC="box.jpg" BORDER="1" WIDTH="300"
HEIGHT="234" ALT="" USEMAP="#gift">
<MAP NAME="gift">
<AREA SHAPE="rect" COORDS="60, 26, 222, 98"
HREF="bantik.html" ALT="Бантик">
<AREA SHAPE="rect" COORDS="63, 88, 135, 188"
HREF="left.html" ALT="Левая сторона">
<AREA SHAPE="rect" COORDS="151, 102, 225, 200"
HREF="right.html" ALT="Правая сторона">
</MAP>
```

В завершении добавим, что для быстрого определения координат достаточно открыть изображение в любом из графических редакторов, например, MS Paint из стандартного набора программ ОС Windows и «счесть» их из строки состояния программы.

Преимущества и недостатки карт-изображений

В использовании карт-изображений есть как положительные, так и отрицательные моменты. Большинство из них носит эстетический характер, но некоторые имеют и технические аспекты. Для создания хороших Web- страниц важно понимать преимущества и недостатки карт-изображений.

Карты-изображения наиболее удобно использовать в следующих ситуациях:

- для представления пространственных связей, например географических координат, которые было бы трудно задать отдельными кнопками или текстом в качестве примера -

карта Северной Америки, на которой выбор каждого из штатов ведет к переходу на соответствующую страницу.

- более удобного средства, чем карта-изображение, для создания сложных навигационных меню (в особенности географических, топографических и прочих карт) не найти. Процесс создания и пространственного размещения на странице нескольких десятков кнопок для обозначения, например, всех областей Российской Федерации, чрезвычайно сложен и потребует больших временных затрат;
- использование карты-изображения в качестве навигационных меню на каждой странице интернетпроекта может существенно сократить время загрузки электронных документов и сэкономить место на Webсервере;
- для использования карты-изображения потребуется изготовить всего один рисунок;
- использование карт-изображений позволит разработчику HTML-документов реализовать самые смелые дизайнерские задумки. Можно создавать графические объекты любой сложности и формы, не задумываясь об их пространственном размещении на странице, что способно придать интернет-ресурсу оригинальность и сделать его более запоминающимся для посетителей.

Несмотря на то, что карты-изображения стали популярны, очевидно, что они не являются неотъемлемым атрибутом Web-страниц и используются далеко не на всех страницах. Есть ситуации, когда не следует использовать карты-изображения.

К недостаткам карт-изображений можно отнести:

- Картам-изображениям свойственны общие недостатки, присущие использованию изображений на Web-страницах, а именно, значительное увеличение времени загрузки по сравнению с чисто текстовыми документами.

- Неудачно спроектированные изображения могут внести путаницу. Иногда бывает трудно определить области, являющиеся активными на изображении.
- Для посетителей, которые экономят свое время пребывания в Интернете, переход к HTML-документам по ссылкам, указанным в конфигурации карты-изображения, не позволяет отслеживать страницы, на которых они уже побывали, поскольку гиперссылки карт-изображений не изменяют цвет после посещения их пользователем;
- Если параллельно с картой-изображением не предусмотрено дублирующее текстовое меню, то посетители, которые по каким-либо причинам не могут загрузить графику или отключили ее в своем браузере, останутся за бортом.
- С картами-изображениями нельзя сделать разные эффекты, которые доступны при разрезании одного рисунка на фрагменты: эффект перекатывания, частичная анимация, индивидуальная оптимизация картинок для их быстрой загрузки.

Интересной альтернативой использования карт-изображений являются flash-ролики или разрезание изображений.

Во flash-роликах можно создавать разные области ссылок, используя возможности векторной графики. Благодаря широким возможностям, на Flash можно создавать потрясающие меню и средства навигации.

Разрезание изображений - это одно из популярных средств в современном дизайне. Одно изображение в этом случае разрезается с помощью специальных программ на фрагменты, которые окончательно сводятся вместе, создавая иллюзию цельной картинке. Хотя области разрезания могут быть только прямоугольные, в большинстве случаев этого вполне достаточно для создания ссылок. Для каждого фрагмента можно выбрать наиболее подходящий графический формат, в котором он будет сохранен, параметры оптимизации, добавить альтернативный текст. Тогда даже при отключенном показе картинок, будут хорошо видны границы областей и замещающий изображение текст.

☑ Вопросы для самоконтроля

1. Что такое карта-изображение, и из каких компонент оно состоит?
2. Какие изображения подойдут для карты-изображения?
3. Когда целесообразно использовать карты-изображения?
4. Какие виды областей можно задать для карты?

§ 2.9. Что такое пользовательские формы

Web-сайт — это почти всегда диалог. Конечно, встречаются «односторонние» сайты, авторы которых стремятся только показать, но это скорее исключение из правил. Даже на таких сайтах редко обходится без ссылки на почту.

Но чаще всего все-таки нас интересует мнение посетителей сайта. Для этой цели используются формы — это совокупность стандартных HTML-конструкций ввода текстовой и прочей информации и программы-обработчика этой информации, работающей на Webсервере. Иными словами, пользовательская форма (или HTML-форма) служит для передачи информационных данных серверу.

Значение пользовательских форм трудно переоценить — они являются особым средством HTML, дающим посетителю возможность не только пассивно просматривать информацию, но и быть задействованным в содержании Web-сайта. Такое свойство принято называть интерактивностью, которая на сегодняшний день встречается практически во всех электронных документах.

Подобно таблицам или текстовым блокам форма является контейнером, который может включать различные элементы: поля для ввода, различные выпадающие меню, списки, «галочки», кнопочки...

Все это многообразие объединяет их предназначение — получение данных от пользователя. И таких контейнеров на странице может быть несколько, главное, чтобы они не пересекались.

Задается форма с помощью парного тега <FORM>.

<FORM параметры>

</FORM>

Параметр NAME позволяет идентифицировать форму уникальным именем для дальнейшего использовать в обработчике данных.

<FORM NAME="reg">

Текстовые блоки

Поля для ввода имени, пароля, поля, где можно прокомментировать ту или иную информацию на странице – все это текстовые блоки.

Наиболее распространенным и способным отобразить широкий набор элементов управления пользовательской формой является тег <INPUT>. Он может «превращаться» и текстовую строку, поле ввода пароля, и многие другие элементы, о которых мы поговорим чуть позже. Тег не является парным и содержит ряд параметров.

Строки для ввода текста на HTML-страницах встречаются сплошь и рядом. Это и строка для ввода поискового запроса, и поле для ввода «логина» и т.д. Представляет собой узкий вытянутый прямоугольник, внутри которого можно ввести с клавиатуры одну строку текста.

Задается с помощью значения параметра TYPE равного "TEXT" для тега <INPUT>.

<INPUT TYPE="text" NAME="name">

Данный элемент содержит ряд дополнительных параметров, описанных в таблице 28.

Таблица 27. Дополнительные параметры <INPUT TYPE="text" >

Параметр	Описание
SIZE	Определение максимального количества символов, видимых в текстовой строке без перемещения курсора
MAXLENGTH	Определение максимального количества символов, допустимых для ввода в текстовой строке. По умолчанию количество вводимых символов не ограничено
NAME	Присвоение идентификационного имени для программы-обработчика

VALUE	Указание значения текстовой строки (при просмотре в браузере выводится в виде обычного текста в самой строке)
READONLY	Позволяет создать элемент, недоступный для редактирования

Рассмотрим несколько вариантов использования элемента текстовой строки.

```
<FORM NAME="mail" ACTION="mail.cgi" METHOD="post">
Ваше имя:<BR>
<INPUT TYPE="text" SIZE="20" MAXLENGTH="50"
NAME="name"> <BR><BR>
Ваш E-mail:<BR>
<INPUT TYPE="text" SIZE="30" MAXLENGTH="35"
NAME="email">
</FORM>
```

Это элемент ввода пользовательского пароля ничем не отличается от обыкновенной текстовой строки, за исключением того, что набранный текст отображается в виде звездочек или точек. Задается он `<INPUT TYPE="password">`.

```
<INPUT TYPE="password" SIZE="30" NAME="password">
```

Напоминаем, что данные, вводимые в это поле, при использовании типа передачи GET будут отображаться в ссылке запроса браузера.

Не всегда текст, который нужно ввести, помещается в одной строке. Бывает, что он растягивается на несколько строк или даже абзацев. Конечно, можно обойтись текстовой строкой «бесконечной» длины (без указания значения параметра `maxlength`). Однако выглядит такая строка — без начала, без конца — неэстетично, а пользоваться ею очень неудобно.

Поэтому для ввода крупных блоков текста предусмотрен другой элемент формы — поле ввода. Именно он и применяется при создании полей для комментариев.

Для создания текстового поля используется парный тег `<TEXTAREA>`. Он создает внутри формы поле для ввода

многострочного текста, отображаемое в окне браузера в виде прямоугольной области с горизонтальной и вертикальной полосами прокрутки. Внутри него помещается текст, который должен оказаться в поле ввода по умолчанию.

```
<TEXTAREA COLS="25" ROWS="5" NAME="comment">Ваш  
комментарий...</TEXTAREA>
```

Основные параметры тега `<TEXTAREA>` представлены в таблице 29

Таблица 28. Основные параметры тега `<TEXTAREA>`.

Параметр	Описание
COLS	Определение количества столбцов текстового поля
ROWS	Определение количества рядов текстового поля
NAME	Присвоение уникального имени, для необходимого идентификации программой- обработчиком
READONLY	Позволяет создать элемент, недоступный для редактирования
Параметр	Описание

WRAP	<p>Способ представления текста, вводимого в окно.</p> <ul style="list-style-type: none"> <input type="checkbox"/> Virtual - в окне текст автоматически разбивается на строки, но при передаче эта автоматическая разбивка не сохраняется, если вы ввели все одной строкой, то оно так и будет передано. Используется по умолчанию. <input type="checkbox"/> Off - Если мы хотим, чтобы переход на новую строку в окне происходил только тогда, когда пользователь нажимает «Enter». <input type="checkbox"/> Hard - если мы хотим, чтобы переход на новую строку происходил автоматически, и эта разбивка сохранялась при передаче текста на обработку
------	---

Элементы выбора

В нашей жизни мы часто выбираем: при регистрации или поиске – город из выпадающего списка, при выборе одежды – наш пол или размер, при подборе моделей в интернет-магазине – интересующие характеристики искомого товара. Все эти действия объединяет наличие элементов выбора (списков) с заведомо определенным набором значений и возможностью выбрать один или несколько из них.

Тег `<INPUT>` поддерживает два типа списков, из которых посетителю страницы предлагается что-то выбрать. В первом случае допускается выбор нескольких вариантов, во втором — только одного. Первый тип называют списком вариантов, а второй — списком переключателем.

Обычно пункты списков вариантов снабжены квадратными «окошками», в которых при выборе появляются «галочки» (checkbox). Для создания такого списка используется тег `<INPUT>` с параметром `type=checkbox`, причем задается столько `<INPUT>`, сколько в списке есть вариантов, только имена у них должны быть разные — у каждого варианта свое. Дело в том, что, хотя логически список является цельным элементом формы, с точки зрения кода это всего лишь набор

разрозненных, несвязанных тегов. У каждого — свое имя и значение. А для того чтобы стало ясно, что именно нам предлагают выбрать, нужно снабдить эти квадратики пояснениями. Пользователь может выбирать несколько вариантов поля CHECKBOX, значение каждого из которых будет передано программой-обработчиком на Web-сервер.

Тег может содержать ряд дополнительных параметров (Таблица 29).

Таблица 29. Дополнительные параметры <INPUT> с параметром type=checkbox

Параметр	Описание
NAME	Указание общего для всех вариантов выбора идентификационного имени
VALUE	Определение значения для конкретного варианта выбора (обязательный параметр). Не должен повторяться, т. к. при установке флажка передается на Web-сервер
CHECKED	Данный вариант является выбранным по умолчанию. Значений он не имеет.

```
<FORM NAME="select" ACTION="strana.php" METHOD="get">
  <INPUT TYPE="checkbox" NAME="strana" VALUE="france"
CHECKED>Франция <BR>
  <INPUT TYPE="checkbox" NAME="strana" VALUE="usa"> США
  <BR>
  <INPUT TYPE="checkbox" NAME="strana" VALUE="canada">
Канада<BR>
</FORM>
```

Очень часто эти элементы служат орудием для завязывания различного рода спама, когда посетитель не дочитав длинный-длинный текст, отправил данные на сервер, согласившись на очередную подписку, учтиво «заполненную» по умолчанию.

За это отвечает параметр без значений CHECKED, добавляемый в тег <INPUT>, как Франция в примере выше.

В списках-переключателях слева от пунктов имеются кружки, причем в центре кружка, соответствующего выбранному пункту, появляется жирная точка (radiobutton).

Для создания таких списков используется тот же тег `<INPUT>`, но параметру `type` уже присваивается значение `radio`. Как и в списке вариантов, каждому элементу списка-переключателя соответствует отдельный тег `<INPUT>`. Однако, в отличие от списка вариантов, все элементы списка-переключателя должны иметь одно имя, для того чтобы быть «единым целым» — выбор одного элемента списка отменяет выбор другого. Наконец, здесь, как и в списке вариантов, для выбора одного из пунктов по умолчанию используется параметр `checked`.

Элемент также имеет дополнительные параметры `NAME`, `VALUE` и `CHECKED`, действие которых аналогично опции выбора `CHECKBOX`.

```
<FORM NAME="select_age" ACTION="age.php"
METHOD="get">
  <INPUT TYPE="radio" NAME="age" VALUE="baby"
CHECKED>0-7 лет<BR>
  <INPUT TYPE="radio" NAME="age" VALUE="child">7-15
лет <BR>
  <INPUT TYPE="radio" NAME="age" VALUE="junior">15-
18 лет <BR>
</FORM>
```

Когда списки бывают очень длинными и скучными и занимают много места, стоит задуматься воспользоваться раскрывающимся списком. Представляет собой поле с некоторым значением и справа небольшую кнопку со стрелкой. Если нажать на стрелке, «выпадает» список. Щелкаем на одном из его пунктов — и список сворачивается обратно, а в строке появляется выбранный пункт.

Подобный список создается с помощью тега `<SELECT>`, а его элементы — с помощью тегов `<OPTION>`.

```
<SELECT name= day>
<OPTION value="Не помню">Не помню
<OPTION value= Понедельник > Понедельник
<OPTION value =Вторник>Вторник
<OPTION value=Среда>Среда
<OPTION value=Четверг>Четверг
<OPTION value=Пятница>Пятница
```

```
<OPTION value=Суббота>Суббота  
<OPTION value=Воскресенье>Воскресенье  
</SELECT>
```

Непарный тег `OPTION` в свою очередь имеет следующие параметры `Selected` - выбранное по умолчанию значение и `value` для обработки данных на стороне сервера.

Однако на этом возможности раскрывающегося списка не исчерпываются. По умолчанию список представляет собой одну строку, которая «раскрывается» при щелчке на кнопке. Его можно модифицировать в окошко с полосой прокрутки и при желании выбором нескольких значений. Для этого нам потребуется параметр `size` тега `<SELECT>`. Этот параметр определяет количество строк, из которых состоит список в «закрытом» состоянии, и по умолчанию равен единице.

Для того чтобы список позволял выбрать сразу несколько вариантов, используется параметр `multiple`. Как и `checked`, этот параметр не имеет значений - в результате программе-обработчику передается несколько значений с одинаковым именем. Для выбора нескольких вариантов работает принцип работы в Windows: если эти пункты следуют подряд, выбираем первый, нажимаем клавишу `Shift` и, удерживая ее, выбираем последний пункт. Если же пункты разбросаны, выбираем их в произвольном порядке, удерживая нажатой клавишу `Ctrl`.

При разработке электронной формы рекомендуется придерживаться следующего правила: то, что посетитель страницы видит на экране одновременно, должно представлять собой законченный блок информации. Тому, чтобы форма выглядела эстетично, и чтобы ею было удобно пользоваться, очень способствует рациональная комбинация различных списков.

Кнопки

Задумывались ли Вы, сколько кнопок нас окружает? Кнопка на будильнике, кнопки на телефоне, клавиатуре, микроволновой печи, чайнике... Этакая «кнопочная» жизнь. Не обошли стороной они и

Интернет-страницы. Кнопки — главный элемент и любой электронной формы. Выделяют четыре основных вида кнопок.

Как это ни забавно, кнопки создаются с помощью того же тега `<INPUT>`, что и текстовые строки. Однако значение параметра `type` в этом случае другое — в зависимости от назначения кнопки.

Это кнопка отправления пользовательских данных на Web-сервер имеет тип `TYPE="SUBMIT"`. При нажатии на нее запускается программа-обработчик, которая анализирует введенные пользователем данные и отправляет результат на сервер. Обычно она отображается в виде прямоугольной кнопки с надписью «Submit». Может иметь имя, задаваемое в параметре `NAME`, и надпись, сохраненную в параметре `VALUE`.

Графический аналог стандартной кнопки отправления данных формы на Web-сервер задается все тем же тегом `<INPUT>` с параметром `TYPE="IMAGE"`. Здесь фантазия дизайнера уже ничем не ограничена.

Для данного элемента используются и некоторые дополнительные параметры, характерные для изображений (табл. 31).

Таблица 30. Дополнительные параметры `<INPUT>` с параметром `type="image"`.

Параметр	Описание
SRC	Указание относительного или абсолютного пути на сервере к файлу графического изображения, служащего кнопкой передачи данных формы;
ALIGN	Указание типа выравнивания текста относительно графической кнопки отправления данных формы;
BORDER	Определение толщины рамки кнопки (как правило, значение равно нулю);
ALT	Указание альтернативного текста/подсказки для кнопки отправления данных.

При работе с многочисленными текстовыми строками и опциями выбора пользователь может, допустив ошибку, пожелать заново заполнить форму, тогда ему придется либо перезагружать страницу, либо вручную удалять текст из каждого поля формы. Для этого случая существует кнопка для сброса введенных пользователем данных HTML-формы и восстановление всех установленных по умолчанию значений полей формы. И вновь нам помогает тег `<INPUT>`, но уже со значением параметра `TYPE="RESET"`.

По умолчанию названием кнопки сброса является «Reset». Изменить наименование элемента можно с помощью дополнительного параметра `VALUE`.

Пример использования кнопок отправления и сброса данных приведен ниже.

```
<FORM NAME="user" ACTION="anketa.php"
METHOD="post">
<p>Анкета пользователя:</p>
<P><B>Контактные данные:</B></P>
Ваше имя:<BR>
<INPUT TYPE="text" SIZE="15" MAXLENGTH="50"
NAME="name">
<BR>Ваш E-mail:<BR>
<INPUT TYPE="text" SIZE="30" MAXLENGTH="35"
```

```
NAME="email">
<P ALIGN="center">
<INPUT TYPE="submit" VALUE="Отправить данные">
 
<INPUT TYPE="reset" VALUE="Очистить поля формы">
</P>
</FORM>
```

Этот тег `<BUTTON>` `</BUTTON>` позволяет создавать кнопки так же, как и тег `INPUT`. Но, в отличие от последнего, он является контейнером - его содержимое может быть достаточно сложным, например, комбинацией текста и графики.

```
<BUTTON name="bt" value="button" type="submit">
Кнопка<IMG src="img.gif"></BUTTON>
```

Параметр тега `type` может принимать следующие значения:

- `BUTTON` - кнопка, щелчок на которой вызывает запрограммированную разработчиком реакцию;
- `RESET` - кнопка для очистки формы.
- `SUBMIT` - кнопка, подтверждающая, что форма заполнена.

Дополнительные параметры тега `<INPUT>`

Кроме основных параметров, в теге `<INPUT>` могут присутствовать некоторые дополнительные параметры, используемые с помощью вспомогательных средств по отношению к HTML.

Это элемент выбора файла `TYPE="FILE"`, расположенного на локальном компьютере пользователя, предназначенного для загрузки на Web-сервер. Из дополнительных параметров действуют только `SIZE` и `MAXLENGTH`. Для корректной передачи файла необходима конструкция `enctype="multipart/form-data"` и указание типа передачи данных `POST` в теге `<FORM>`. В противном случае браузер передаст не выбранный файл, а путь к нему на компьютере пользователя.

Следует добавить, что средств HTML, способных изменять название кнопки просмотра содержимого локального компьютера, не существует. Значение может быть только одно и зависит от модели браузера: «Browse» или «Обзор».

Служебное поле, не видимое пользователю, не может быть изменено ни браузером, ни самим пользователем задается через TYPE="HIDDEN". Оно анализируется программой-обработчиком и отправляется на Web-сервер, причем отправляемое значение варьируется специальным параметром VALUE.

Помимо этого, для любого элемента, заданного тегом <INPUT> можно применяться следующие параметры (табл. 32).

Таблица 31. Дополнительные параметры тега <INPUT>.

Параметр	Описание
DISABLED	Параметр, отменяющий активность выбранного поля формы. При указании параметра DISABLED редактирование элемента формы будет недоступно. Используется в сложных, динамически изменяющихся HTML-формах в совокупности с такими технологическими средствами, как DHTML, CSS, JScript и др.;
TITLE	Параметр отображения всплывающей подсказки для элемента формы.
STYLE	Параметр определения стиля CSS для выбранного элемента формы;
CLASS	Параметр, устанавливающий класс стилевого шаблона CSS (ссылающийся на внешний или внутренний набор стилей).

Группировка элементов формы

Элемент <LABEL></LABEL> является контейнером для других элементов формы и позволяет их группировать. Например, можно объединить надпись и поле ввода:

```
<label> адрес: <INPUT type="text" id="address">
```

</label>

Если элемент LABEL и другой элемент находятся отдельно, используется параметр for, значение которого должно совпадать со значением параметра id соответствующего элемента:

```
<LABEL for="address">Ваш адрес: </label>  
<INPUT type="text" id="address">
```

Для каждого элемента LABEL создается один элемент формы. Иногда элементы удобно распределить по ячейкам таблицы, обеспечив тем самым выравнивание данных в форме.

Два элемента <FIELDSET> <LEGEND> </LEGEND> </FIELDSET> также предназначены для создания группы полей в форме.

С помощью элемента FIELDSET несколько элементов объединяются: пользователь видит их заключенными в рамку. Внутри группы элементы формы используются обычным способом.

Элемент LEGEND позволяет создать заголовок группы. Поскольку этот элемент является контейнером, в нем можно размещать другие элементы HTML. Например, заголовок группы можно составить из двух строк, если использовать тег
.

```
<FIELDSET>  
<LEGEND>Заголовок группы</LEGEND>  
Имя: <INPUT name="imya" type="text">  
Фамилия: <INPUT name="familiya" type="text"><BR>  
Телефон: <INPUT name="telefon" type="text"><BR>  
Текст подсказки  
</FIELDSET>
```

С помощью параметра align можно регулировать положение заголовка:

- top — заголовок сверху;
- bottom — заголовок внизу (что не всегда удастся реализовать);

- left — заголовок вверху и слева (значение по умолчанию);
- right — заголовок вверху и справа.

Итак, электронная форма — эффективное средство, благодаря которому HTML-страница превращается из статичной, лишь предоставляющей информацию пользователю, в «активную», взаимодействующую с ним, позволяющую принять информацию и передать ее для обработки.

Способ обработки и передачи данных определяется тегом `<FORM>`, внутри которого и заключается код формы. Средство обработки определяется параметром `action`, метод передачи — параметром `method`, а тип кодирования — параметром `enctype`.

Внутреннее содержание формы можно разделить на две части: активное и пассивное. Пассивными элементами формы являются все декоративные элементы, которые могут там содержаться. Это обычные составляющие HTML-страницы.

Активные элементы формы предназначены для ввода данных. Это строки и поля ввода, списки и кнопки. У каждого активного элемента формы — как и у всей формы — есть два основных параметра — `name` и `value`. Первый определяет имя элемента, по которому его можно отличить от других элементов формы, второй — значение, которое передается через этот элемент.

Данные, вводимые посредством формы, обрабатываются не средствами HTML. Они могут передаваться по электронной почте или непосредственно программе-обработчику. Язык, на котором может быть написана такая программа, значения не имеет. В частности, для обработки таких данных могут использоваться сценарии на языке JavaScript.

Процесс обработки и передачи данных

Специальная программа, которая будет обрабатывать введенные в форму данные, определяется параметром `ACTION`. Значением параметра выступает URL адрес (относительный или абсолютный), где она расположена. Эта программа может быть написана на любом серверном

языке, например, PHP, Python, C# , и расположена как на удаленном сервере, так и на компьютере посетителя страницы.

```
<form action="/reg/reg.php">
```

Впрочем, вместо адреса программы-обработчика значением параметра action может быть обычный адрес электронной почты. Тогда данные формы будут просто отправлены по этому адресу, а что с ними делать дальше — решит получатель.

```
<form action="mailto:form@narfu.ru.ru"  
enctype="text/plain">
```

Способ обработки данных задается параметром METHOD и может осуществляться двумя методами. Передача самих данных соответствует значению post (от английского «переслать»), а передача ссылки на данные — значению get («получить»). По умолчанию используется последний вариант, дабы излишне не нагружать сеть.

При использовании типа GET данные пользовательской формы пересылаются в составе адреса запроса браузера: после имени программы-обработчика ставится знак вопроса (?), обозначающий вывод запроса браузера к переменным HTML-формы, а также последовательность переменных и их свойств из самой формы. Последовательность переменных формы разделяется символом амперсанда —&.

Пример адреса запроса браузера с использованием типа передачи данных GET:

```
http://www.narfu.ru/reg/reg.php?Name  
=Petr&Email=Petr@narfu.ru
```

Из структуры ссылки, образовавшейся в ходе обработки данных формы, понятно, что пользователь ввел свое имя («Petr») и адрес электронной почты («Petr@narfu.ru»).

При использовании типа POST данные формы не отображаются в адресной строке браузера, а передаются в составе самого запроса

программы-обработчика. Таким образом, используя этот тип, мы получим следующую гиперссылку (учитывая те же данные формы, что и в случае с типом GET):

<http://www.narfu.ru/reg/reg.php>

Например, при передаче данных формы по электронной почте используется значение post.

Зачастую пользовательская форма содержит конфиденциальную информацию, в этом случае использование метода GET просто противопоказано. Иначе вся секретная информация, введенная пользователем, будет доступна для просмотра любому пользователю данного компьютера (просмотр истории перехода по Web-сайтам в браузере легко обнаружит ссылку с конфиденциальными данными).

Тип кодирования данных, введенных через форму, определяется параметром ENCTYPE (от английского encryption type — тип кодирования). Кодирование осуществляется браузером и используется для предотвращения разного рода искажений в процессе передачи на сервер. Возможными значениями параметра могут быть: application/xwww-form-urlencoded (по умолчанию) и multipart/form-data. Первое значение используется, если помимо текста необходимо передать на сервер данные иного типа (к примеру, графику или запакованные файлы). Значение multipart/form-data используется в редких специфических случаях, например, при необходимости предоставить пользователю возможность загрузки на сервер любого файла со своего локального компьютера, выбранный при помощи элемента формы <INPUT TYPE=FILE>.

Значение параметра определяется двумя параметрами: тип и подтип данных. Тип данных — текст (text), графика (image), программа (application). Подтип — это уже классификация данных внутри типа (image/gif, text/html).

Например, при отправке данных электронной почтой используется тип text/plain. Таким образом, для того чтобы данные формы передавались по электронной почте, код формы должен выглядеть так:

```
<FORM action="mailto:form@narfu.ru.ru"
enctype="text/plain" method="post">
</FORM>
```

Схема кодирования application/x-www-form-urlencoded одинакова для обоих методов пересылки (GET и POST) и заключается в следующем. Для каждого элемента формы, имеющего имя, заданное параметром NAME, формируется пара "name=value", где value — значение элемента, введенное пользователем или назначенное по умолчанию. Если значение отсутствует, соответствующая пара имеет вид "name——". Для радиокнопок и переключателей используются значения только выбранных элементов. Если элемент выбран, а значение параметра VALUE не определено, по умолчанию используется значение «on». Если флажки образуют группу, то передаваемым значением является строка разделенных запятыми значений параметра VALUE всех установленных флажков.

Все пары объединяются в строку, в качестве разделителя служит символ "&". Так как имена и значения представляют собой обычный текст, то они могут содержать символы, недопустимые в составе URL (метод GET пересылает данные как часть URL). Такие символы заменяются последовательностью, состоящей из символа % и их шестнадцатеричного ASCII-кода. Символ пробела может заменяться не только кодом %20, но и знаком + (плюс). Признаком конца строки, встречающийся в поле TEXTAREA, заменяется кодом %OD%OA. Такое кодирование называется URL-кодированием.

Закодированная информация пересылается серверу одним из методов GET или POST. При использовании метода GET данные формы пересылаются серверу в составе URL-запроса, к которому добавляются после символа "?", они называются строкой запроса. Тело запроса в этом случае является пустым. Web-сервер, получив запрос, присвоит переменной среды QUERY_STRING значение строки запроса и вызовет программу, обозначенную в первой части URL до символа "?".

Строка запроса — не единственный способ передачи данных через URL. Другим способом является дополнительная информация о пути (extra path information), представляющая собой часть URL,

расположенную после имени специальной программы. Сервер выделяет эту часть и сохраняет ее в переменной среды PATH_INFO. Программа обработки может затем использовать эту переменную для извлечения данных.

Для того чтобы описать, какие действия должен совершить браузер после отправки данных, какую страницу и в какое окно загрузить используют параметр TARGET. С уже знакомыми нам параметрами _blank, _self или именем фрейма.

☑ Вопросы для самоконтроля

1. Какие виды кнопок существуют на web-страницах?
2. Какие элементы позволяет создать универсальный тег `<INPUT>`?
3. Какие технологии для передачи данных форм существуют?
4. С помощью какого типа передачи стоит отправлять данные о пароле?

§ 2.10. Фреймы

Фрейм (от англ. «frame» — рамка, каркас, кадр) представляет собой отдельное рабочее окно браузера, разделенное еще на несколько различных окон по параметрам и размеру. Совокупность таких окон принято называть фреймовой структурой.

Каждый фрейм — представляет собой самостоятельную вебстраницу, загружаемую в соответствующем «окошке» общего пространства страницы браузера. Таким образом, HTML-документ, созданный на фреймовой основе, является набором взаимосвязанных электронных документов, параметры и свойства которых определяются настройками всей фреймовой структуры.

Многого общего фреймы имеют с таблицами — и те и другие осуществляют разбиение окна просмотра браузера на прямоугольные области, в которых располагается некоторая информация. Однако при помощи таблиц можно решить только задачу форматирования страниц документа, а с помощью фреймов — еще и организовать взаимодействие между ними[12].

Еще одно существенное различие между фреймами и таблицами состоит в том, что каждому фрейму должен соответствовать отдельный HTML-документ, в то время как содержимое всех ячеек таблицы всегда часть одного документа. Каждый фрейм по существу является отдельным «мини-браузером», любая отображаемая в нем страница может независимо от других прокручиваться при просмотре. Таблицы могут полностью не помещаться в окне и быть просмотрены только по частям, у фреймов же структура всегда представлена на экране. Отсюда следует вывод, что если в HTML-таблицах общее число ячеек практически не ограничено и может достигать нескольких сотен, то число фреймов в документе обычно не превосходит нескольких единиц.

В конечном итоге, выбор структуры документа — табличной или фреймовой — зависит от многих факторов и не может быть однозначно предопределен.

Сферы применения фреймов. Достоинства и недостатки

Сегодня вокруг фреймов существует множество споров о целесообразности их использования. Один говорят, что они давно уже устарели, другие утверждают — что их можно применять. Прежде чем принять окончательное решение рассмотрим их преимущества и недостатки.

Среди достоинств использования фреймов можно выделить[11,12,14]:

- Интуитивная простота в использовании. Благодаря двум независимым документам очень легко организовать навигацию. Один фрейм содержит ссылки пункты меню, другой — контент, загружаемый при нажатии на ссылки из первого фрейма.
- Уменьшение объема каждой страницы за счет отсутствия повторяющихся блоков меню. И как следствие уменьшение всей загружаемой с сайта информации.
- Возможность точного размещения информации в окне браузера. Вне зависимости от прокручивания содержимого

главного фрейма с контентом, боковые блоки сохраняют свое местоположение.

- Возможность изменения размеров фрейма во время загрузки, невозможная при традиционной верстке HTML.
- Возможность изменять только один «блок» страницы без перезагрузки других.
- Возможность обновить два и более фрейма нажатием одной гиперссылки (при помощи одного из скриптовых языков).

Противники применения фреймов в HTML-документах выделяют следующие недостатки[11,12,14]:

- Проблемы с навигацией. Пользователь может оказаться на сайте с так называемого «черного входа». Случается это при переходе из результатов поискового запроса, когда загружается лишь фрейм с контентом, а все вспомогательные фреймы, особенно фрейм с навигацией остается «за бортом». Ни навигации, не названия сайта, какой-то обрывок. Единственное, что остается пользователю – вручную редактировать путь в адресной строке, что в любом случае доставляет неудобство.
- Плохая индексация поисковыми системами. Поисковые системы плохо работают с фреймовой структурой, поскольку на страницах, которые содержат контент, чаще всего нет ссылок на другие документы.
- Внутренние страницы нельзя добавить в «Закладки». Пользователь всегда видит только адрес самой фреймовой структуры, а не составляющих ее страниц. Это затрудняет помещение понравившейся страницы в закладки браузера.
- Отсутствие кроссплатформенности браузеров. Некоторые параметры фреймов могут совершенно по-своему отображаться в различных браузерах. Порой до явного противоречия.
- Непрестижность. Сегодня сайты с фреймами считаются несолидными, а их авторы сразу исключаются из гильдии профессионалов, которые никогда не используют фреймы в

своих работах. Единственным исключением остаются чаты, где без фреймов обойтись, хотя и можно, но достаточно своеобразными и мудреными методами, а с помощью фреймов они создаются почти элементарно.

Стоит заметить, что некоторые приведенные недостатки вполне можно разрешить. Например, отдельный документ формируется со всех фреймовой структурой за счет применения скриптов. Да и показатели поисковых систем в области индексации фреймовых документов гораздо выше, чем несколько лет назад.

В настоящее время фреймы в каком-то смысле заняли свою нишу и применяются там, где недостатки фреймов не имеют особого значения, а преимущества наоборот, активно востребованы. Чаще всего это в системах администрирования и справке.

Создание фреймов

Для создания фреймовой структуры используется, по крайней мере, три HTML-документа: первый определяет фреймовую структуру и делит окно браузера на две части, а оставшиеся два документа загружаются в заданные окна.

Для создания фреймовой структуры используется тег `<frameset>`, который применяется для деления экрана на области и записывается в документе за место `<body>`. Его параметры `COLS` или `ROWS` задают соответственно количество горизонтальных или вертикальных фреймов и их размер. Размер можно определять, фиксировано в пикселях, или же в процентах от размера окна браузера.

Внутри данного тега находятся теги `<FRAME>`, которые определяют фреймы, которые будут загружены в эту структуру.

Наиболее типичное использование фреймовых структур, когда один фрейм служит оглавлением документов, а другой используется для загрузки их содержимого. Он представлен на рисунке (Рисунок 9).

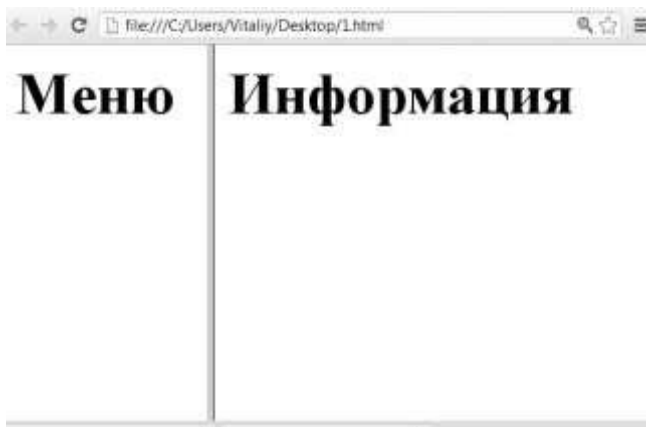


Рисунок 9. Пример разделения окна браузера на два фрейма

Для создания структуры, заданной на рисунке 11 понадобится главный файл со структурой, например, `index.htm`. Файл `left.htm` будет загружаться в левый фрейм, `right.html` — в правый фрейм. Задание файлов `left.htm` и `right.html` ничем не будет отличаться от ранее создаваемых web-страниц, потому их мы опустим.

Только `index.htm` отличается по виду своего кода от других файлов.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
  <head>
    <title>Фреймы</title>
  </head>
  <frameset cols="200,*">
    <frame src="left.htm" name="MENU">
    <frame src="right.htm" name="CONTENT">
  </frameset>
</html>
```

При использовании фреймов в первой строке кода указывается тип документа. Данный `<!DOCTYPE>` указывают браузеру, что он имеет дело с фреймами, эта строка кода является обязательной. Тег `<head>` содержит заголовок документа, который будет являться общим для всех файлов данной фреймовой структуры.

В данном примере окно браузера разбивается на две колонки с помощью параметра `cols`, левая колонка занимает 200 пикселей, а правая

— все оставшееся пространство, заданное символом звездочки. Ширину или высоту фреймов можно задавать и в процентном отношении к окну браузера.

Тег <FRAME> содержит имя HTML-файла, который будет загружен в указанную область с помощью параметра SRC. В левое окно будет загружен файл, названный left.html, а в правое — right.html. Каждому фрейму желательно задавать уникальное имя с помощью параметра NAME, тогда его можно будет динамически изменять. Имена фреймов должны начинаться с латинской буквы или цифры, при их написании учитывается регистр. Так, например, фреймы с именами "frame_A" и "Frame_a" будут различны.

Нередко можно увидеть и такие случаи использования фреймов: два смежных фрейма использованы для загрузки документов, которые удобно просматривать одновременно и сопоставлять друг с другом, как например, при совершении покупок бытовой техники, различающихся некоторыми характеристиками. В каждом из двух документов, загружаемых во фреймы, используется табличная форма для представления информации, которую можно совместно сравнивать, а также независимо просматривать и распечатать.

Изменение размеров фреймов

Для описания размеров фреймовых структур возможно задавать целый ряд дополнительных параметров.

Так, параметр value тега <FRAME> позволяет определить высоту/ширину фрейма. Ее можно задавать фиксировано, в пикселях, или в процентах. Первый вариант не рекомендуется, поскольку браузеры имеют различный размер рабочего поля, не говоря уже об экранных разрешениях у пользователя. Некорректно заданные процентные соотношения фреймов будут автоматически подправлены: если общая сумма процентов описываемых фреймов превысит 100%, то размеры всех фреймов пропорционально уменьшаться, в противном случае - пропорционально увеличиваются.

Наилучшим способом задания размера фрейма является применения символа «*», указывающего на то, что все оставшееся место

будет принадлежать данному фрейму. Если указывается два или более фрейма со значением «*», то оставшееся пространство делится поровну между этими фреймами. Если перед звездочкой стоит цифра, то она указывает пропорцию для данного фрейма (во сколько раз он будет больше аналогично описанного просто звездочкой). Например, описание "4*,*,*", говорит, что будет создано три фрейма с размерами 2/3 свободного пространства для первого фрейма и по 1/6 для двух других.

Параметры MARGINWIDTH и MARGINHEIGHT тега <FRAME> определяют соответственно размер (в пикселях) горизонтального или вертикального отступа между фреймом и его границей. В свою очередь, параметр SCROLLING отвечает за наличие полос прокрутки у фрейма. Если значение этого параметра равно «YES» - полосы всегда присутствуют на странице, «NO» - всегда отсутствуют, «AUTO» — появляются в случае необходимости, если информация в окне выходит за его рамки (режим по умолчанию).

Для того чтобы пользователь не смог изменить размеры фрейма или прокрутить информацию в нем, достаточно добавить для этого фрейма параметр NORESIZE.

Границы между фреймами

Тег <FRAMESET>, как уже говорилось ранее, определяет количество и структуру фреймов, составляющих страницу. Его логический параметр FRAMEBORDER определяет наличие границы между фреймами («YES» / «NO»), BORDER – ее толщину, а BORDERCOLOR – цвет. Задание этих параметров для тега <FRAME> позволяет задать стиль границы определенного фрейма. Правила употребления параметров BORDER и BORDERCOLOR аналогичны применимых при описании таблиц.

По умолчанию граница отображается в виде трехмерной линии. Чтобы ее скрыть используется параметр frameborder, которому указывают значение 0. Однако в браузере Opera граница хоть и становится в этом случае бледной, все же остается, потому для этого браузера требуется добавить framespacing="0". Также браузер Opera игнорирует и цвет границы, изображая ее всегда черной.

```
<frameset cols="200,*" frameborder="0"
framespacing="0">
  <frame src="menu.html" name="MENU">
  <frame src="content.html" name="CONTENT">
</frameset>
</html>
```

Взаимодействие между фреймами

Простейшая форма просмотра информации на WWW состоит в чтении страниц и переходах по ссылкам, при которых текущий документ в окне браузера замещается другим документом. При работе с фреймами можно организовать более удобную для пользователя схему загрузки документов.

Взаимодействие между фреймами заключается в возможности загрузки документов в выбранный фрейм по командам из другого фрейма. Для этой цели используется параметр TARGET тэга <A>. Данный параметр определяет имя фрейма или окна браузера, в которое будет загружаться документ, на который указывает данная ссылка. По умолчанию при отсутствии параметра TARGET документ загружается в текущий фрейм (или окно).

В качестве имени может задаваться имя существующего окна или фрейма, а может указываться новое имя, под которым будет открыто новое окно. Два значения («_blank», «_self») нами были уже рассмотрены для знакомства с гиперссылками. Значение «_parent» и «_top» загружает документ в родительское для фреймов окно — саму фреймовую структуру. Запоминаем, что зарезервированные значения «_blank», «_self», «_parent», «_top» должны записываться строчными латинскими буквами.

Итак, в примере выше (рис.11), во всех гиперссылках левого фрейма навигации left.html необходимо указывать фрейм с именем «CONTENT»:
 ссылка .

Рассмотрим еще один пример взаимодействия между фреймами и отдельными окнами браузера.


```

<HTML>
  <HEAD>
    <TITLE>Использование фреймов</TITLE>
  </HEAD>
  <FRAMESET COLS=2*,*,*>
    <FRAME SRC= a.htm NAME="A">
    <FRAME SRC=blank.htm NAME="B">
    <FRAME SRC=blank.htm NAME="C">
  </FRAMESET>
</HTML>

```

В этом HTML-документе дается описание структуры, состоящей из трех фреймов с именами "A", "B" и "C". Заметим, что задавать имя фрейму "A" в данном примере было необязательно потому, как гиперссылок на него не предусмотрено. При загрузке приведенной страницы в браузер во фреймах будет отображена информация, содержащаяся в файлах, определяемых параметром SRC. Во фрейм "A" попадет содержимое файла a.htm, а остальные два фрейма получат данные из файла blank.htm, не содержащих пока никакой информации.

Текст кода фрейма A (файл a.htm), являющегося стандартной страницей и содержащей гиперссылки для загрузки страницы test.htm.

```

<HTML>
  <HEAD>
    <TITLE>Меню</TITLE>
  </HEAD>
  <BODY>
    <A HREF="test.htm" TARGET="_blank">Загрузка в новое
    окно </A><P>
    <A HREF="test.htm" TARGET="_top">Загрузка в полное
    окно </A><P>
    <A HREF="test.htm" TARGET="_self">Загрузка в текущий
    фрейм </A>
    <A HREF="test.htm" TARGET="B">Загрузка во фрейм B
    </A><P>
    <A HREF="test.htm" TARGET="C">Загрузка во фрейм C
    </A><P>

```

```
<A HREF="test.htm" TARGET="D">Загрузка в окно с  
именем D</A><P>  
</BODY>  
</HTML>
```

Текст файла test.htm принципиального значения не имеет. При реализации ссылок, описанных в a.htm первая ссылка со значением TARGET="_blank" будет каждый раз создавать новое окно без имени и загружать туда указанный документ. Следующая ссылка со значением TARGET="_top" загрузит документ в полное окно вместо всей фреймовой структуры, вернуться обратно к фреймовой структуре будет возможно нажатием кнопки «Назад». Ссылка со значением TARGET="_self" загрузит документ во фрейм "A" на место документа со ссылками, в данном случае результат идентичен выполнению ссылки без параметра TARGET.

Но наиболее интересными, с точки зрения работы с фреймами, являются следующие три ссылки. Ссылка со значением TARGET="B" будет загружать файл test.htm во фрейм с именем "B", следующая ссылка выполнит те же действия для фрейма "C". А последняя из ссылок со значением TARGET="D" приведет к образованию нового окна браузера с именем "D" и загрузке в него файла test.htm, куда впоследствии и будут загружаться все гиперссылки с данным значением TARGET.

При работе с фреймами очень часто имена фреймов путают с самими загружаемыми файлами. Имена фреймов при просмотре нигде не видны, они требуются только для организации взаимодействия и поэтому скрыты от пользователя, а имена файлов отвечают за наполнение этих фреймов. Увидеть их можно только при просмотре исходного текста HTML-файлов[11].

Плавающие фреймы

Разговор о фреймах будет неполным без упоминания плавающих фреймов. Так называется фрейм, который можно добавлять в любое место страницы, причем он может иметь произвольные размеры. Еще одно его название — встроенный фрейм, он называется так из-за своей

особенности встраиваться прямо в тело web-страницы. На рисунке ниже (Рисунок 10) приведен пример такого фрейма.

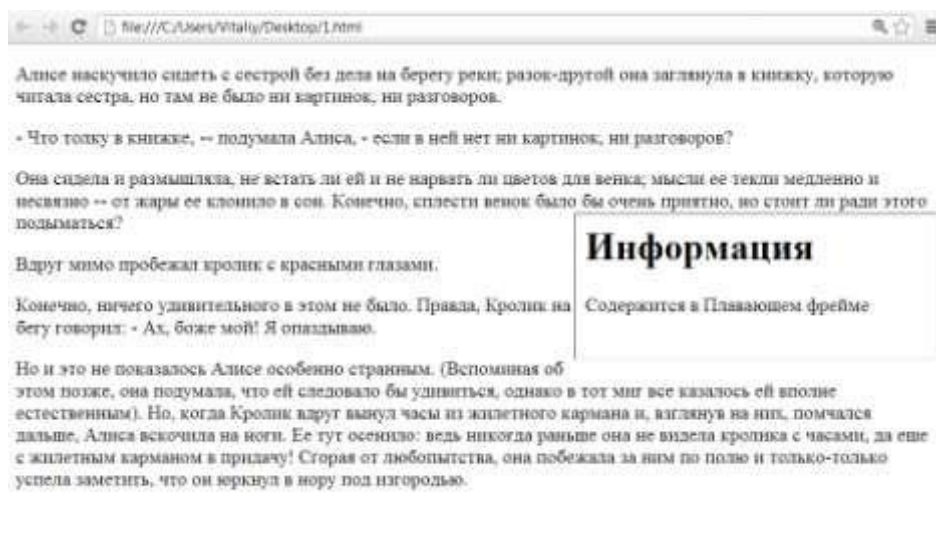


Рисунок 10. Плавающий фрейм на web-странице

Во фрейм можно загружать HTML-документ и прокручивать его содержимое независимо от остального материала на странице.

Добавление плавающего фрейма происходит с помощью тега `<IFRAME>`, имеющего обязательный параметр `SRC`, указывающий на адрес загружаемой во фрейм страницы.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01  
Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">  
<html>  
  <head>  
    <title>Плавающий фрейм</title>  
  </head>  
  <body>  
    <p><iframe src="1.html" width="300"  
height="120"></iframe></p>  
  </body></html>
```

В данном примере ширина и высота фрейма устанавливается через параметры `WIDTH` и `HEIGHT`. Сам загружаемый во фрейм файл называется `1.html`. Заметьте, что если содержимое не помещается целиком в отведенную область, появляются полосы прокрутки. Еще одно удобство плавающих фреймов состоит в том, что в него можно загружать

документы по ссылке. Для этого требуется задать имя фрейма через параметр NAME, а в теге <A> указать это же имя в параметре TARGET.

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

Синтаксис этого тега комбинирует в себе параметры фреймов – как родственников, и параметры вставки изображений – так как позволяет себя размещать в любом месте страницы.

Параметры, унаследованные от традиционных фреймов: FRAMEBORDER, SCROLLING, MARGINWIDTH, MARGINHEIGHT. От изображений этому тегу достались такие свойства, как Размеры окна WIDTH и HEIGHT, отступы при «обтекании» VSPACE и HSPACE.

☑ Вопросы для самоконтроля

1. Что представляют собой фреймы?
2. Раскройте понятие «плавающий фрейм».
3. Какие параметры фреймов можно регулировать?
4. В чем преимущества и недостатки использования фреймов?

§ 2.11. Добавление Google-карт

Часто встречаются ситуации, когда на сайте необходимо указать схему проезда до какого-либо места. Для этого достаточно подключить одну из карт, генерируемых поисковыми системами. Рассмотрим схему добавления карты на страницу на примере Google-карт.

Для этого на сайте www.google.ru выбираем раздел «Карты» или же напрямую заходим по ссылке maps.google.ru (Рисунок 11).

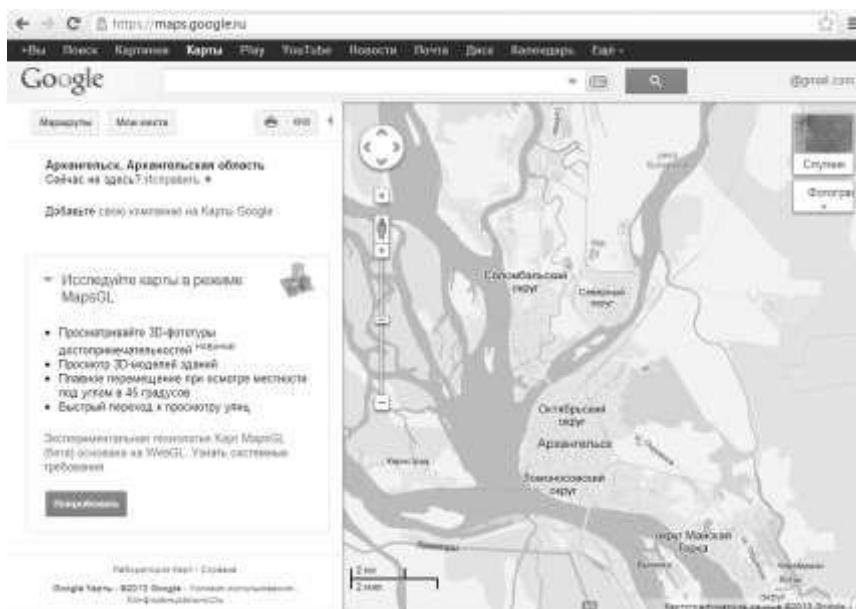


Рисунок 11. Ресурс maps.google.ru

Первое, что необходимо сделать - войти в систему под своим аккаунтом. Или же пройти систему регистрации, она бесплатна и не займет много времени. После того, как мы были идентифицированы системой, необходимо ввести в поисковую строку адрес, который хотим отметить на карте, или же один из двух адресов - если мы хотим указать схему проезда.

Например, «Урицкого д.68 В». Система автоматически подберет похожие адреса из своей базы и покажет популярные результаты (Рисунок 12).

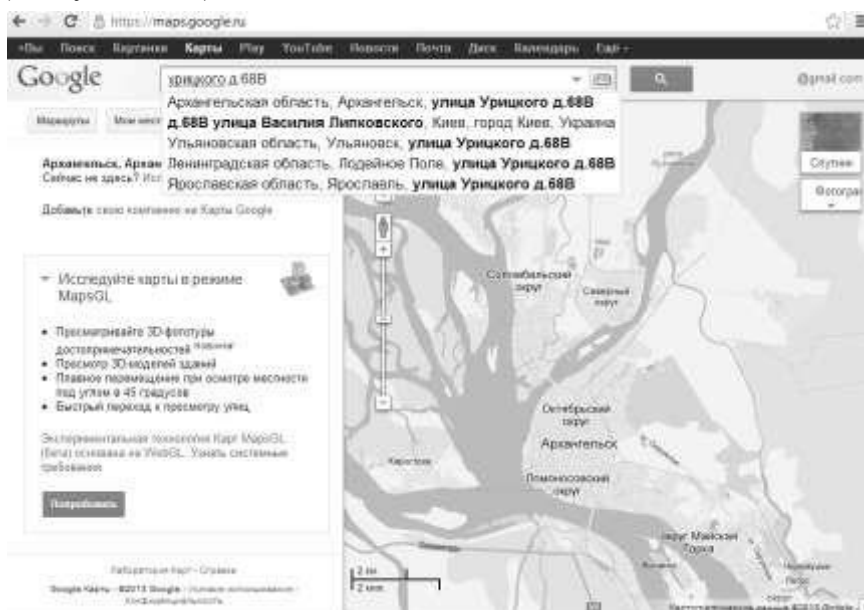


Рисунок 12. Выбор адреса.

После выбора нужного варианта на карте отобразится искомый адрес, а панели слева можно будет увидеть все организации, расположенные по данному адресу (Рисунок 13).

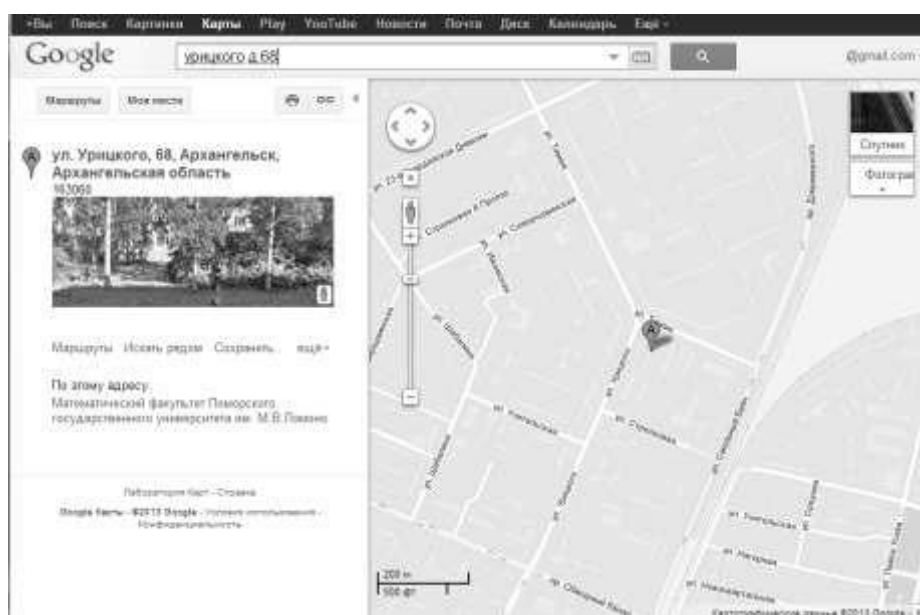


Рисунок 13. Вывод организаций, расположенных по данному адресу.

Теперь добавим второй адрес, до которого мы хотим продолжить маршрут. Для этого входим на вкладку «Маршруты» в меню слева. И добавляем второй адрес (Рисунок 14).



Рисунок 14. Добавление второго адреса.

И нажимаем кнопку «проложить маршрут». Схема проезда готова (Рисунок 15). Осталось добавить ее на нашу страницу. Для этого в правом верхнем углу нажимаем на кнопку «ссылка».

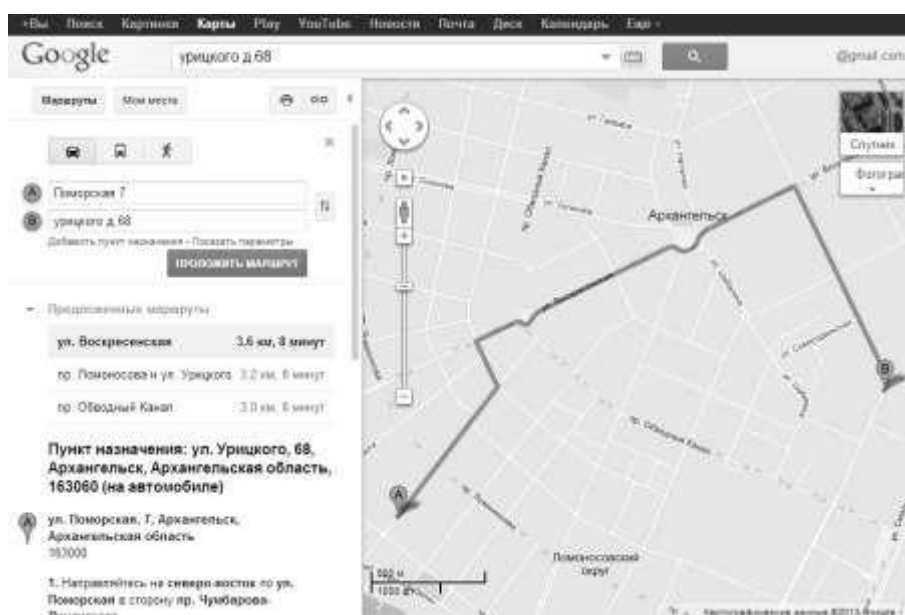


Рисунок 15. Прокладывание маршрута.

Выпадающее меню предлагает нам код для добавления на сайт в виде плавающего фрейма (Рисунок 16).



Рисунок 16. Получение ссылки на карту.

Для более тонкой настройки выбираем пункт «Настройка и предварительный просмотр». Здесь мы можем задать размер карты или

относительно и абсолютно (в пикселях), все зависит от выделенного на странице места под карту (Рисунок 17).



Рисунок 17. Панель «Настройка и предварительный просмотр».

Возможно, придется даже уменьшить масштаб карты, чтобы увидеть оба пункта назначения. После выбора необходимо варианта копируем его код и вставляем на нашу страницу.

☒ Вопросы для самоконтроля

1. Какие виды маршрутов может задать Google-карта?
2. Как добавить Google-карту на страницу?
3. Какие параметры для нее можно задать?

§ 2.12. HTML 5. Новые теги и параметры

Интернет сильно изменился с тех времен, как стандарт HTML 4.01 был утвержден в 1999 году. Все новые и новые задачи ставятся сегодня перед браузерами. Некоторые элементы в HTML 4.01 устарели, не используются или используются не так, как это задумывалось[14]. Так стандарт HTML 4.01 постепенно вытесняется новой технологией HTML5, для которой характерно:

- Наличие новых возможностей, основанных на традиционных HTML, CSS, DOM и JavaScript

- Снижение необходимости использования внешних плагинов (например, Flash)
- Независимость от устройств вывода
- Улучшенная разметка для сценариев
- Полная открытость процесса развития
- Улучшенная обработка ошибок
- Упрощение декларации `<!doctype>`
- Поддержка локального хранилища

В скором времени технология HTML5 будет признана новым стандартом HTML. Данный стандарт – это совместная работа консорциума World Wide Web Consortium (W3C) и Web Hypertext Application Technology Working Group (WHATWG), работавших в двух параллельных направлениях и решивших в 2006 году объединить свои усилия и создать новую версию языка разметки HTML (WHATWG работали над веб-формами и приложениями, а W3C разрабатывал стандарт XHTML 2.0). Их объединение позволило выйти на новый уровень развития, и уже сегодня большинство браузеров поддерживает целый ряд новых элементов, созданных в HTML5 и API: новые элементы для организации более грамотной структуры документа, семантики, рисования и для медиа-контента[5].

Так в HTML5 осталась только одна, очень простая, декларация `<!doctype>`:

```
<!DOCTYPE html>
```

Новые семантические элементы

HTML5 предлагает новые элементы для лучшей семантики и структурирования элементов документа.

Это `<header>` и `<footer>`, позволяющие определить заголовочный блок или «футер» подвал документа (секции), а также `<nav>` для работы с навигационное меню.

`<aside>`, который определяет элементы, которые не являются частью содержимого, но косвенно с ним связаны: цитаты, дополнительная

информация к статье, словарь с терминами, список ссылок и т.д. Кроме того, появился тег `<figure>`, позволяющий работать с иллюстрациями, диаграммами, фотографиями, листингами кода и т.п. и `<figcaption>` задающий подпись к ним.

Данный стандарт позволяет определить статью (`<article>`), группу заголовков `<h1>...<h6>`, когда заголовок включает в себя несколько уровней заголовков (`<hgroup>`), секцию документа (`<section>`).

Выделение, «подсветка» текста может осуществляться теперь с помощью тега `<mark>`, а форматирование элементов, содержащих дату и время с помощью тега `<time>`.

Новые медиа элементы

До сих пор не было стандарта для показа видео и аудио на веб-странице. Большинство сайтов воспроизводило их через плагины (например, Flash), или с помощью ограниченных в функциях `<embed>`. HTML5 определяет новый элемент для стандартного способа вставки аудио и видео файлов на web-страницу.

Элемент `<video>` отвечает за вставку видеофайла, на данный момент тег поддерживается Internet Explorer 9 +, Firefox, Opera, Chrome и Safari.

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg"> Ваш
браузер не поддерживает элемент video.
</video>
```

Параметр `controls` добавляет элементы управления видео, например, воспроизвести, поставить на паузу и управление звуком. А `width` и `height` задают размер отображения видеофайла.

В настоящее время существует три поддерживаемых формата видео для элемента `<video>`: MP4, WebM и Ogg. О поддержке данных форматов браузерами указано в приложении А.

HTML5 определяет также и новый элемент `<audio>`., который отвечающий за стандартный способ вставлять аудио на web-страницу.

```
<audio controls>  
  <source src="horse.ogg" type="audio/ogg">  
  <source src="horse.mp3" type="audio/mpeg"> Ваш  
браузер не поддерживает элемент audio.  
</audio>
```

В настоящее время существует три поддерживаемых формата файлов для `<audio>` элемента: MP3, WAV и Ogg. В приложении А указана таблица по поддержке данных форматов браузерами.

Вспомогательный тег `<source>` для этих элементов определяет медиаисточник.

А тег `<track>` позволяет авторам указать текстовую дорожку для медийных элементов `<audio>` и `<video>`. Такая дорожка обычно содержит субтитры на разных языках, комментарии, заголовки и др.

Новый элемент canvas

Один из самых интересных и функциональных тегов, появившихся в новом стандарте, это холст для рисования `<canvas>`, позволяющий создавать двумерные изображения, анимацию, а также динамически их трансформировать и форматировать с использованием скриптового языка JavaScript. Подробнее о данном теге и его возможностях будет рассказано в 4 главе, после знакомства с основным инструментом данного тега – JavaScript.

Новые элементы формы

Не осталась нетронутой и такая область интернет-приложений, как форма. Новый стандарт добавил множество новых элементов, контролирующих ввод и проверку введенных данных, позволяющих выбрать на календаре дату или с помощью бегунка визуально откорректировать тот или иной параметр.

HTML5 имеет несколько новых типов полей для ввода данных для форм. Эти новые функции позволяют лучше контролировать ввод и проверку введенных данных.

Больше всего нововведения коснулись многоликого и универсального элемента `<input>`, добавив новых значений параметру `type`.

Так `type=date` позволяет пользователю выбрать дату, а `type=color` – цвет с помощью традиционных диалоговых окон.

`type=email` определяет поле, которое должно содержать email адрес, а `type=url` - должно содержать url адрес. Значения, введенные в такие поля, автоматически проверяются перед отправкой на сервер.

С помощью `type=number` и `type=range` можно задать поля, которое должно содержать только числа, причем диапазон принимаемых чисел можно ограничить с помощью параметров `min` (минимальное допустимое число) и `max` (максимальное допустимое число). Также можно задать и шаг допустимых чисел (параметр `step`). В отличие от `type=number` элемент `type=range` отображается как ползунок, который можно перетаскивать мышкой.

Элемент `type=search` определяет поле поиска и может использоваться, например, для создания поиска по сайту.

Кроме того, существуют элементы, позволяющие поработать с датой и временем: поле дата-время (`Datetime`), локальное дата-время (`datetime-local`), поле месяц (`month`), поле время (`time`), поле неделя (`week`) и поле телефон (`tel`).

Примечание: не все основные браузеры на данный момент поддерживают новые типы полей ввода. Тем не менее, если они не поддерживаются, они будут вести себя как обычные текстовые поля.

Появились также и новые элементы форм, расширяющие функционал современных web-приложений. К ним относят `<datalist>`, который создает список вариантов, которые можно выбирать при наборе в текстовом поле. Изначально этот список скрыт и становится доступным при получении полем фокуса. Тег `<keygen>`, который используется для генерации пары ключей — закрытого и открытого. Когда форма отправляется на сервер, закрытый ключ сохраняется на локальном компьютере, а открытый ключ передается вместе с формой. Сами ключи необходимы для шифрования и расшифровки данных, создания и проверки цифровой подписи. И элемент `<output>`, определяющий

область, в которую выводится информация, преимущественно с помощью скриптов.

Как уже говорилось, многие элементы стандарта HTML 4.01 были пересмотрены и модернизированы, а некоторые и вовсе удалены из стандарта HTML5 как морально устаревшие:

- ☐ `<acronym>`
- ☐ `<applet>`
- ☐ `<basefont>`
- ☐ `<big>`
- ☐ `<center>`
- ☐ `<dir>`
- ☐ ``
- ☐ `<frame>`
- ☐ `<frameset>`
- ☐ `<noframes>`
- ☐ `<strike>`
- ☐ `<tt>`

На момент написания учебного пособия работа над спецификацией HTML5 все еще ведется.

☒ Вопросы для самоконтроля

1. Какими преимуществами обладает стандарт HTML5 по сравнению с HTML 4.01?
2. Зачем введены новые элементы `<header>`, `<nav>`, `<footer>`, `<article>`?
3. Какие видео и аудио форматы поддерживаются для размещения на web-страницах через элементы `<video>` и `<audio>`?

☒ Лабораторная работа №1

ВАРИАНТ 1.

Ранее уже говорилось о том, что HTML- страницы можно создавать в любом текстовом редакторе. Вы можете выбрать любой из предложенных или воспользоваться блокнотом.

☒ *Задание 1.*

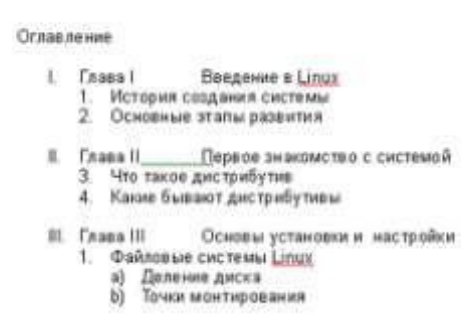
Создайте свою страницу, содержащую

1. заголовок, отображаемый в окне браузера,
2. информацию о создателе,
3. ключевые слова,
4. краткую информацию о содержании страницы,
5. обновление каждые 2 минуты,
6. информацию об используемой кодировке,
7. фоновую картинку и бледно-синий цвет фона, на тот случай, если картинки не отображаются,
8. темно-синий цвет текста.

В тексте страницы необходимо указать личные данные (ФИО, группа, направление подготовки) и отрывок из любимого художественного произведения не менее 3000 символов.

1. разбейте текст страницы на абзацы, вставив места возможных переносов.
2. выделите цветом, размером, шрифтом, начертанием важные моменты,
3. Добавьте к второму абзацу изображение, обтекаемое текстом слева и имеющее заданный размер.
4. Добавьте в текст страницы оформленное синим цветом небольшое стихотворение с указанием автора строк. Предварительно задайте название стихотворения в виде заголовка.
5. Добавьте меню для перехода между разделами данной страницы, расположите его в начале документа.
6. Добавьте ссылку на страницу, где размещена информация об авторе вашего рассказа.

7. Добавьте в конце страницы ссылку «Написать разработчику» на ваш электронный ящик. А также строку с создателем страницы и значком авторских прав.
8. Добавьте таблицу

Текст по центру (Оформите цитатой в кавычках-елочках Ваш жизненный принцип)		
снизу справа формула серной кислоты H_2SO_4 и формулу $\alpha^2 \pm \beta^2 = \gamma^2$,	Текст, размещенный по ширине ячейки	список с графическими маркерами на тему основные виды параметров тега META.
	Список: 	
Фоновое изображение		

☒ Задание 2.

Используя полученные знания, создайте страницу, содержащую

1. текстовую строку, позволяющую вводить не более 20 символов, причем видимыми являются 10 символов.
Изначально отображается текст «Введите Имя».
2. Поле для ввода секретного ключа не более 8 символов.
3. Список возможных предметов для изучения для выбора нескольких значений (по умолчанию – web-программирование) и список, позволяющий выбрать единственный номер курса (по умолчанию – 3 курс).
4. Многострочное поле для ввода информации о пользователе (например, курсы, которые были изучены и т.д.).

5. Текстовую строку, содержащую некоторую информацию и доступную только для чтения.
6. поле ввода возраста, поле с ползунком для выбора размера обуви, поле для ввода адреса сайта и поле для выбора даты рождения. Все эти поля оформите с помощью новых типов полей ввода.
7. Разместите на странице небольшой видеоролик с кнопками управления. Выведите текст об ошибке для тех браузеров, которые не поддерживают элемент `<video>`.
8. Добавьте Google-карту проезда от Вашего дома до одного из Ваших любимых мест города.

ВАРИАНТ 2.

Ранее уже говорилось о том, что HTML- страницы можно создавать в любом текстовом редакторе. Вы можете выбрать любой из предложенных или воспользоваться блокнотом.

☒ Задание 1.

Создайте свою страницу, содержащую

1. заголовок, отображаемый в окне браузера,
2. информацию о создателе,
3. ключевые слова,
4. обновление каждые 3 минуты,
5. информацию об используемой кодировке,
6. фоновую картинку и бледно-зеленый цвет фона, на тот случай, если картинки не отображаются,
7. темно-зеленый цвет текста.

В тексте страницы необходимо указать личные данные (ФИО, группа, направление подготовки) и отрывок из любимого художественного произведения не менее 3000 символов.

1. разбейте текст страницы на абзацы, вставив места возможных переносов.

2. выделите цветом, размером, шрифтом, начертанием важные моменты,
3. Добавьте к третьему абзацу изображение, обтекаемое текстом справа и имеющее заданный размер 100 пикселей по ширине и рамку в 3 пикселя.
4. Добавьте в текст страницы оформленное черным цветом небольшое стихотворение с указанием автора строк. Предварительно задайте название стихотворения в виде заголовка 2 уровня.
5. Добавьте меню для перехода между разделами данной страницы, расположите его в начале документа.
6. Добавьте ссылку на страницу, где размещена информация об авторе вашего рассказа.
7. Добавьте в конце страницы ссылку «Написать разработчику» на ваш электронный ящик. А также строку с создателем страницы и значком авторских прав.
8. Добавьте таблицу

<p style="text-align: center;">Текст по ширине (Оформите цитатой в кавычках-елочках Ваш жизненный принцип)</p>		
<p>снизу справа формула марганцовки KMnO_4 и формулу $\alpha^2 \pm \beta^2 = \gamma^2$,</p>		<p>Список:</p> <p>Оглавление</p> <p>I. Глава I Введение в <u>Linux</u></p> <p>1. История создания системы</p> <p>2. Основные этапы развития</p> <p>II. Глава II Первое знакомство с системой</p> <p>3. Что такое дистрибутив</p> <p>4. Какие бывают дистрибутивы</p> <p>III. Глава III Основы установки и настройки</p> <p>1. Файловые системы <u>Linux</u></p> <p>а) Деление диска</p> <p>б) Точки монтирования</p>
<p>список с графическими маркерами на тему основные виды гиперссылок</p>	<p>Текст, размещенный по ширине ячейки</p>	

☑ *Задание 2.*

Используя полученные знания, создайте страницу, содержащую

1. текстовую строку, позволяющую вводить не более 10 символов, причем видимыми являются 5 символов.

Изначально отображается текст «Введите номер договора».

2. Поле для ввода секретного ключа не более 8 символов.
3. Список возможных стран для посещения (3-4 страны), возможность для выбора нескольких значений (по умолчанию – Франция) и список, позволяющий выбрать единственный сезон года (по умолчанию – лето).
4. Многострочное поле для ввода информации о посещении пользователем страны.
5. Текстовую строку, содержащую информацию, доступную только для чтения.
6. поле ввода числа ночей для отдыха, поле с ползунком для выбора количества звезд гостиницы (2-5 звезд, 3 по умолчанию), поле для выбора даты начала и конца отдыха. Все эти поля оформите с помощью новых типов полей ввода.
7. Разместите на странице небольшой аудиофрагмент с кнопками управления. Выведите текст об ошибке для тех браузеров, которые не поддерживают элемент <audio>.
8. Добавьте Google-карту проезда от Вашего дома до одного из Ваших любимых мест города.