МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

КАРАЧАЕВО-ЧЕРКЕССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. У.Д. Алиева

WEB-программирование: HTML, CSS и JAVASCRIPT

Учебно-методическое пособие

Карачаевск 2013

ББК-32.973.26-018.2

УДК-681.3.06

Печатается по решению
редакционно-издательского совета
Крачаево-Черкесского
государственного университета

Web-программирование HTML, CSS и JavaScript/ Учебно-методическое пособие. –Карачаевск: изд-во К-ЧГУ, 2013.- 124с.

В пособии излагаются методические рекомендации к выполнению лабораторных работ по дисциплине “Web-программирование ”. Предназначено для студентов, обучающихся по всем профилям подготовки бакалавров направления*: Информатика, Прикладная информатика, Прикладная математика и информатика*

Составители: Лепшокова А.Н., к.п.н.
Эльканова А.А., к.п.н.

Рецензенты: Салпагаров Х.М., к.ф-м.н., профессор
Кубекова Б.С., к.ф-м.н., доцент

Оглавление

Введение

Всемирная Паутина (World Wide Web) спровоцировала революцию в информатике, предоставив любому пользователю возможность публикации HTML-документов. До недавнего времени информация в этих документах была в большинстве случаев статической, что требовало реакции сервера на действия пользователя. С введением динамического HTML парадигма Web сместилась от взаимодействия с сервером в сторону создания интерактивных Web-узлов и Web-приложений. Поскольку динамический HTML обеспечивает возможность взаимодействия HTML-документов с пользователем и полного их изменения на клиентском компьютере, вы можете создавать Web-приложения с богатыми возможностями.

В результате изучения данного пособия, проводимого под руководством преподавателя, студенты познакомятся с:

• технологиями и концепцией создания статических web-страниц;
• принципами разработки структуры web-страниц;
• основными элементами языка;
• средствами оформления страниц;
• способами передачи данных от удаленного пользователя наСервер;
• технологиями и основными принципами объектно-ориентированного программирования;
• принципами создания динамических Web-документов;
• основными элементами языка;
• взаимосвязью языков скриптов и таблицей стилей для оформления Web-документов;
• организацией проверки данных введенных пользователем.

По окончании данного курса студенты смогут:

• создавать несложные статические сайты и наполнять их форматированным и структурированным содержимым;
• иметь представление об основах технологии объектно-ориентированного программирования, необходимых для Web-разработки;
• использовать стилевое форматирование для оформления HTML-документов.
• иметь представление о языке создания сценариев (то есть уметь понимать конструкции языка и интерпретировать результат);
• создавать Web-документы с динамически изменяемым содержимым;
• использовать стилевое форматирование совместно с языками сценариев для расширения возможностей оформления документов.

ГЛАВА I. ОБЗОР HTML И CSS.

1. Обзор возможностей языка HTML

Термин HTML (HyperText Markup Language) означает "язык разметки гипертекстов", с помощью которого верстаются Web-страницы. В самом начале развития WWW регулировать стандарты разработки была призвана международная общественная организация, включающая в себя представителей фирм-разработчиков и исследовательских институтов — W3C (World Wide Web Consortium). Более полную информацию по этому языку можно посмотреть по адресу: http://www.w3.org.

Характеристики HTML:

• Разработан специально для Web;
• Открытый стандарт;
• В HTML включен гипертекст;
• В HTML включена поддержка мультимедиа;

Со времени создания первой версии HTML претерпел некоторые изменения. Известны спецификации 2.0, 3.0, 3.2, 4.0, 5.0. Текущую спецификацию HTML всегда можно найти на сервере W3C (http://www.w3.org/). Все что необходимо, чтобы прочитать HTML-документ - это Web-браузер, который интерпретирует элементы HTML и воспроизводит на экране документ в виде, который ему придает автор. Основное преимущество HTML заключается в том, что документ может быть просмотрен на Web-браузерах различных типов и на различных платформах. HTML-документы могут быть созданы при помощи любого текстового редактора или специализированных HTML-редакторов и конвертеров. С другой стороны можно использовать специальные программы – редакторы HTML текстов.

1.1. Структура документа

Основными конструкциями языка являются тэги. Все тэги HTML начинаются с "<" (левой угловой скобки) и заканчиваются символом ">" (правой угловой скобки). Как правило, существует стартовый тэг и завершающий тэг.

<title> Заголовок документа </title>

Завершающий тэг выглядит также, как стартовый, он отличается от него прямым слэшем перед текстом внутри угловых скобок. Некоторые тэги, такие, как <hr> (тэг, определяющий горизонтальную линию), не требуют завершающего тэга. HTML не реагирует на регистр символов, на синтаксис. Тэги либо распознаются браузером,

либо нет. Когда Web-браузер получает документ, он определяет, как документ должен быть интерпретирован. Самый первый тэг, который встречается в документе, должен быть тэгом <html>. Простейший HTML-документ будет выглядеть так:

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

Основные элементы страницы

Заголовочная часть документа <head> . Элементы, находящиеся внутри раздела head (кроме названия документа, записываемого с помощью раздела title), не видны на экране (во всяком случае, напрямую). Элементы, содержащиеся внутри раздела head документа, нужны для того, чтобы:

• Дать документу название.
• Определить отношения между несколькими документами.
• Дать указание браузеру создать форму для поиска.
• Добавить динамическую составляющую.

Стартовый тэг <head> помещается непосредственно перед тэгом <title>, а завершающий тэг </head> размещается сразу после окончания описания документа. Например:

```
<head>
<title> Список сотрудников </title>
</head>
```

Заголовок документа <title> .

Раздел title служит для того, чтобы дать документу название. но обычно показывается в заголовке окна 6раузера. Название документа записывается между тэгами <title> и </title> и представляет собой текстовую строку.

Тело документа <body> . Тело документа должно находиться между тэгами <body> и </body>. Атрибуты этого тега определяют общий облик вашего документа. Они перечислены в Таблице 1.

Таблица 1.

Атрибут	Назначение
background	Указывает на URL-адрес изображения, которое используется в качестве фонового.

bgcolor	Определяет цвет фона документа
bgproperties	Если установлено значение fixed, фоновое изображение не прокручивается.
alink	Определяет цвет активной ссылки.
link	Определяет цвет еще не просмотренной ссылки.
vlink	Определяет цвет уже просмотренной ссылки.
text	Определяет цвет текста.
topmargin	Устанавливает границу верхнего поля в пикселах.
leftmargin	Устанавливает границу левого поля в пикселах.

Комментарии.

Как любой язык, HTML позволяет вставлять в тело документа комментарии, которые сохраняются при передаче документа по сети, но не отображаются браузером. Синтаксис комментария:

<!-- Это комментарий -->

Комментарии могут встречаться в документе где угодно и в любом количестве.

Элемент address.

Этот элемент служит для идентификации автора документа. Сюда же обычно помещаются сведения об авторских правах. Этот элемент располагается либо в начале, либо в самом конце документа. Элемент address состоит из текста, помещенного между тэгами <address> и </address>. Текст, заключенный между этими тэгами, обычно показывается в окне браузера курсивом.

Цветовое оформление документа

Если цвета составных частей документа не определены автором, то используются цвета по умолчанию. Они определяются установками программы просмотра. В HTML цвета определяются цифрами в шестнадцатеричном коде. Цветовая система базируется на трех основных цветах – красном, зеленом и синем – и обозначается RGB. Для каждого цвета задается шестнадцатеричное значение в пределах от 00 до FF, что соответствует диапазону 0 – 255 в десятичном исчислении. Затем эти значения объединяются в одно число, перед которым ставится символ #. Например, число #800080 обозначает фиолетовый цвет. Для простоты в HTML определены 16 стандартных цветов, которые вместе с их шестнадцатеричными кодами приведены в Таблице 2.

Таблица 2.

Цвет	Код	Цвет	Код
Black (черный)	#000000	Silver (серебряный)	#C0C0C0
Maroon (темно-бордовый)	#800000	Red (красный)	#FF0000
Green (зеленый)	#008000	Lime (известь)	#00FF00
Olive (оливковый)	#808000	Yellow (желтый)	#FFFF00
Navy (темно-синий)	#000080	Blue (синий)	#0000FF
Purple (фиолетовый)	#800080	Fuchsia (фуксия)	#FF00FF
Teal (чирок)	#008080	Aqua (аква)	#00FFFF
Gray (серый)	#808080	White (белый)	#FFFFFF

Атрибут bgcolor отвечает за цвет фона документа. Атрибут text определяет цвет текста документа. Атрибут link используется браузером для показа ещё непросмотренных ссылок. Атрибут vlink служит для показа уже просмотренных ссылок. Как правило, их окрашивают более темным оттенком того же цвета, что и непросмотренные ссылки. Атрибут alink определяет цвет ссылки, активной в текущий момент.

Установка полей

Атрибут leftmargin устанавливает расстояние между левым краем текста и левым краем окна браузера, которое измеряется в пикселах. Атрибут topmargin служит для установки расстояния между верхним краем текста и верхним краем окна браузера.

Специальные символы

Некоторые специальные символы не входят в базовую часть таблицы кодов ASCII. К ним относятся буквы алфавитов части европейских языков, математические и некоторые другие символы. Однако они тоже могут быть введены в ваш HTML-документ при помощи символа & (амперсант) и имени символа. Например, пробел: .

1.2. Форматирование документов

Любые опубликованные материалы имеют определенную структуру.

Разделение текста на абзацы

Поместите открывающий тэг в начало каждого нового абзаца вашего текста, и программа просмотра отделит абзацы друг от друга пустой строкой. Использование закрывающего тэга необязательно. Атрибут align, имеющий значения, представленные в Таблице 3. По умолчанию происходит выравнивание по левому краю.

Таблица 3.

Значение	Функция
left	Выравнивание текста по левой границе окна браузера.
center	Выравнивание по центру окна браузера.
right	Выравнивание по правой границе окна браузера.

В HTML несколько стоящих подряд тэгов не дают дополнительного пространства между абзацами.

Перевод строки

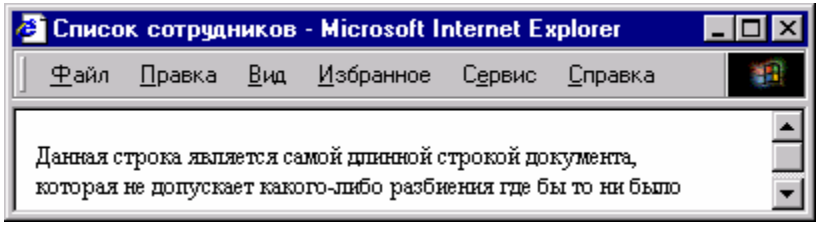
Для того чтобы перейти на следующую строку в любом нужном вам месте текущей строки, в HTML существует тэг разрыва строки
. Он заставляет программу просмотра выводить стоящие после него символы сначала новой строки. В отличие от тэга абзаца, тэг
 не добавляет
пустую строку.

Пример:

```
<body>
Эта и следующая строка
разделены тэгом конца абзаца
<p>
А эти две строки разделены тэгом
конца строки.
<br>
Разница видна?
</body>
```

Бывают случаи, когда возникает надобность в операции противоположного назначения – запретить перевод строки. Текст, заключенный между тэгами <nobr> и </nobr>, будет гарантированно располагаться в одной строке без переноса на другую. Во избежание

неприятностей с элементом <nobr> вы можете предложить браузеру читателя альтернативное место перевода строки при помощи тэга <wbr> ("мягкий" перевод строки). Эта инструкция будет выполнена только в том случае, если браузер не сможет вывести вашу фразу одной строкой в пределах окна просмотра.
Пример:
<nobr> Данная строка является самой длинной строкой документа,<wbr> которая не допускает какого-либо разбиения где бы то ни было </nobr>



В данном случае перевод строки будет произведен только после запятой.
Структурирование текста
Для удобства читателей текст рекомендуется разбить на логические части, каждая из которых посвящена отдельной теме.
Заголовки.
Элемент "заголовок" является контейнером и поэтому должен иметь открывающий (<h1>) и закрывающий (</h1>) тэги. HTML располагает шестью уровнями заголовков: h1 (самый верхний), h2, h3, h4, h5 и h6 (самый нижний). Программа просмотра выводит каждый из них, но вы не можете точно знать, в каком именно виде. Это обстоятельство является частью философии HTML: вы, как автор, отвечаете за содержание, а читатель определяет окончательный облик документа. Предназначение заголовков – показать структуру вашего документа. Точно так же, как и в теге "абзац", в заголовках можно использовать атрибут align.
Горизонтальные линии.
Элемент <hr> позволяет провести рельефную горизонтальную линию в окне большинства программ просмотра. Этот тэг не является контейнером, поэтому не требует закрывающего тэга. До и после линии автоматически вставляется пустая строка.

Таблица 4.

Атрибут	Назначение
align	Выравнивает по краю или центру; имеет значения left,

	center, right.
width	Устанавливает длину линии в пикселах или процентах от ширины окна браузера; в последнем случае добавляется символ %
size	Устанавливает ширину линии в пикселах.
noshade	Отменяет рельефность линии.
color	Указывает цвет линии. Используется формат RGB или стандартное имя.

Пример:

<body>
<hr align=center>
< hr align=center width=100>
< hr align=center width=50%>
< hr align=center width =100 size=5>
< hr align=center width =100 size=5 color=red>
</body>

Использование предварительно отформатированного текста.

Наиболее употребляемым является контейнер <pre>. Текст внутри него может записываться в любой форме. Поддерживаются также тэги <p> и
. Универсальность этого контейнера позволяет создавать таблицы и ровные колонки текста. Текст внутри контейнера <pre> может содержать любые элементы физического и логического форматирования. Однако запрещено использование тэга <address> и тэгов заголовка. Самым большим недостатком контейнера <pre> является возможность вывода преформатированного текста только моноширинным шрифтом.

Пример:

<pre>
HR ALIGN=CENTER
HR ALIGN=CENTER WIDTH=100
HR ALIGN=CENTER WIDTH=50%
HR ALIGN=CENTER WIDTH=100 SIZE=5
HR ALIGN=CENTER WIDTH=100 SIZE=5 COLOR=RED
</pre>

Цитата <blockquote>.

Данный тэг предназначен для обозначения в документе цитаты из другого источника. Текст, обозначенный тэгом <blockquote>, отступает от левого края документа на 8 пробелов.

<body>
На открытии данной конференции глава представительства произнес:
<blockquote>
Сегодня один из величайших дней для нашей компании.
</ blockquote>
</body>
При отображении браузером текст будет выглядеть так:

1.3. Форматирование текста

Логическое форматирование

Элементы логического форматирования.

Таблица 5.

<cite>	Используется для выделения цитат или названий книг и статей, при этом текст обычно выводится курсивом. <cite>Tom Sawyer</cite> remains one of the classics of American literature.
<code>	Применяется для вывода небольшого куска программного кода (для больших листингов используется тэг <pre>) шрифтом фиксированной ширины.
	Этот элемент обычно используется для выделения важных фрагментов текста. Браузеры обычно отображают такой текст курсивом. The actual line reads, "Alas, poor Yorick. I knew him, Horatio."
<kbd>	Элемент, выделяющий шрифтом фиксированной ширины текст, вводимый пользователем с клавиатуры. To run the decoder, type <kbd>Restore</kbd> followed by your password.
<samp>	Используется для выделения нескольких символов шрифтом фиксированной ширины. The letters <samp>AEIOU</samp> are the vowels of the English

	language.
	Этот элемент обычно используется для выделения важных фрагментов текста. Браузеры обычно отображают такой текст полужирным шрифтом. The most important rule to remember is Don't panic!
<var>	Используется для отметки имен переменных. Обычно такой текст отображается курсивом. The sort routine rotates on the <var>I</var>th element.

Все перечисленные элементы являются контейнерами и требуют закрывающего тэга.

Физическое форматирование

Последней "инстанцией" определения внешнего вида документа является программа просмотра читателя. Вы имеете ограниченные возможности повлиять на этот процесс с помощью элементов физического форматирования, список которых приведен ниже. Физическое форматирование считается “не рекомендованным“ оформления Web- страниц.

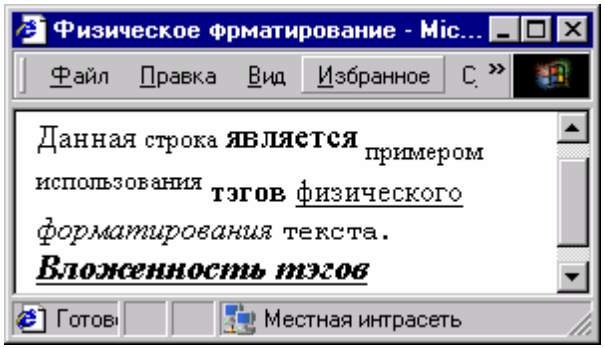
В последних версиях языка вместо него используются каскадные таблицы тилей. Но тем не менее, тэги физического форматирования можно встретить еще во многих документах.

	Выделяет текст полужирным шрифтом.
	This is in bold text.
	Выделяет текст курсивом.
	This is initalic text.
<tt>	Выводит текст шрифтом фиксированной ширины.
	This is in <tt>teletype</tt> text.
<u>	Элемент подчёркивания.
	This text is <u>underlined</u>.

<strike>	Элемент зачеркивания. Отображается текст, перечеркнутый горизонтальной линией.
	This is a <strike>strikethough</strike> example.
<big>	Выводит текст шрифтом большего размера.
	This is <big>big</big> text.
<small>	Выводит текст шрифтом меньшего размера.
	This is small text.
<sub>	Сдвигает текст ниже уровня строки и выводит его (если возможно) шрифтом меньшего размера.
	This is a _{subscript}.
<sup>	Сдвигает текст выше уровня строки и выводит его (если возможно) шрифтом меньшего размера.
	This is a ^{superscript}.

Все элементы физического форматирования являются контейнерами, т. е. требуют закрывающего тэга. Элементы физического форматирования
могут быть вложенными друг в друга.

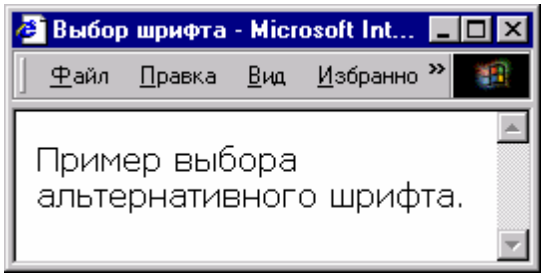
```
<body>
Данная
<small>строка</small>
<b><big>является</big></b>
<sub> примером</sub>
<sup>использования</sup>
<b>тэгов</b>
<u> физического</u>
<i> форматирования </i>
<tt>текста.</tt>
<b> <i> <u>
<big>Вложенность тэгов</big>
</u> </i> </u>
</body>
```



Шрифты.

Элемент font представляет собой контейнер, т. е. требует как открывающего, так и закрывающего тэгов. После стартового тэгаобязательно указание атрибутов, без которых элемент не оказывает никакого влияния на текст, помещенный в контейнер. Элемент font может использоваться внутри любого другого текстового контейнера. Атрибут face данного элемента позволяет вам указать тип шрифта, которым программа просмотра выведет ваш текст. Параметром атрибута служит название шрифта, которое должно в точности совпадать с названием шрифта, имеющегося у пользователя. Если нужного шрифта нет, программа проигнорирует запрос и будет использовать шрифт, установленный по умолчанию. Атрибут face позволяет указать как один, так и несколько шрифтов (через запятую). Весь список будет просмотрен слева направо, и первый из имеющихся на машине пользователя будет использован для вывода документа.

```
<html>
<head>
<title>Выбор шрифта</title>
</head>
<body>
<font face="Verdana", "Arial", "Helvetica">
Пример выбора альтернативного шрифта. </font>
</body>
</html>
```



Атрибут size служит для указания размера шрифта в условных единицах от 1 до 7. Принято считать, что размер "нормального" шрифта соответствует числу 3. Размер может быть указан как абсолютной величиной (size=5), так и относительной (size=+2).

```
<body>
<font size=1>Size 1</font><br>
<font size=-1>Size 2</font><br>
<font size=3>Size 3</font><br>
<font size=4>Size 4</font><br>
<font size=+2>Size 5</font><br>
<font size=6>Size 6</font><br>
<font size=+4>Size 7</font><br>
</body>
```

Атрибут color устанавливает цвет шрифта, который может быть указан как в формате RGB, так и стандартным именем.

```
<body>
<font color="#FF0000">Этот текст будет выведен красным
цветом</FONT><BR>
<font color="green"> Этот текст будет выведен зеленым цветом
</FONT><BR>
</body>
```

Тэг <basefont> служит для указания размера, типа и цвета шрифта, стандартных для данного документа. Эти величины обязательны для всего документа, если только не переназначаются при помощи элемента font. После закрытия элемента font значения тэга <basefont> восстанавливаются. Значения атрибутов basefont могут быть изменены другим тэгом <basefont> в любом месте документа. Это не контейнер и закрывающего тэга не существует. Тэг <basefont> использует те же самые атрибуты, что и элемент font.

```
<body>
Это текст по умолчанию.<br>
<basefont size=4 face="georgia">
А этот текст использует тэг.<br>
BASEFONT, который меняет
параметры шрифта
```

```
<font size=-3> А это
исключение</font>.<br>
</body>
```

1.4. Списки

HTML имеет специальные элементы-контейнеры для представления данных в виде списков. Основными типами списков являются нумерованные и маркированные списки, списки определений. Для получения дополнительных эффектов различные типы списков могут вкладываться друг в друга.

Упорядоченный (нумерованный) список.

В HTML список состоит из тэга-контейнера списка, определяющего его тип, и стандартных тэгов <li>, предваряющих каждый пункт списка. Упорядоченный список используется для нумерованного перечисления отдельных пунктов или указания последовательности каких-то действий. Когда программа просмотра встречает тэг упорядоченного списка, она последовательно нумерует пункты списка: 1, 2, 3 и т.д. Упорядоченный список открывается тэгом <ol>, а каждый его пункт начинается стандартным тэгом <li>. Для создания заголовка списка используется специальный тэг <lh>. Список закрывается тэгом . Открывающий и закрывающий тэги обеспечивают перевод строки до и после списка, отделяя таким образом список от остальн ого текста. Тэги абзаца можно использовать для отделения пунктов списка друг от друга. Внутри списка можно применять тэги стилей (как физические, так и логические, например, или <i>), и другие элементы HTML.

```
<body>
<ol>
<lh><em>Цвета радуги:</em><br>
<li>Red
<li>Orange
<li>Yellow
<li>Green
<li>Blue
<li>Indigo
<li>Violet
</ol>
</body>
```

Упорядоченные списки допускают вложения друг в друга Атрибуты тэга <ol> позволяют устанавливать вид маркеров элементов списка, а также задавать начальный маркер списка. В Таблице 6 приведены атрибуты этого тэга и их назначение.

Таблица 6.

Атрибут	Назначение
compact	Представляет список в более компактном виде.
type=A	Устанавливает маркер в виде прописных букв.
type=a	Устанавливает маркер в виде строчных букв.
type=I	Устанавливает маркер в виде больших римских цифр.
type=i	Устанавливает маркер в виде маленьких римских цифр.
type=1	Устанавливает маркер в виде арабских цифр.
start=n	Устанавливает начальный маркер в текущем списке.

Неупорядоченный (маркированный) список.

В HTML существует возможность создания неупорядоченных списков, т. е. таких, в которых отношения между пунктами не определены. (Списки такого типа называют также ненумерованными или маркированными.) Неупорядоченный список вместо буквенной или цифровой нумерации предполагает использование различных символов, называемых маркерами списка (bullets). Как и в упорядоченных списках, здесь также обеспечивается перевод строки до и после списка, а также допускается вложенность списков. Список располагается внутри контейнера <ul>. Программы просмотра создают автоматический отступ для вложенных списков и сами разнообразят маркеры, вид которых зависит от типа программы. Как и в случае тэга <ol>, для тэга <ul> в HTML можно устанавливать вид маркеров в неупорядоченных списках при помощи атрибута type, который допускает три значения: disc, square и circle Как и тэг <ol>, тэг <ul> имеет атрибут compact, позволяющий выводить неупорядоченный список в более компактном виде.

Список определений.

Списки определений, также называемые словарями специальных терминов (глоссариями), являются особым видом списков HTML. Они представляют текст в форме словарной статьи, состоящей из определяемого термина и абзаца, раскрывающего его значение. Элемент списка определений dl является контейнером и обеспечивает отделение списка от остального текста пустыми строками. Внутри контейнера тэгом <dt> помечается определяемый

термин, а тэгом <dd> – абзац с его определением. Тэги <dt> и <dd> не являются контейнерами и поэтому не имеют закрывающих тэгов. Базовый шаблон списка определений выглядит
следующим образом:

```
<dl>
<dt>Термин
<dd>Определение данного термина
</dl>
```

Списки определений могут включать другие элементы HTML. Часто применяются элементы стилей (физические и логические).

Комбинирование списков

Бывают ситуации, когда в список одного типа требуется включить список (или списки) другого типа. HTML позволяет производить любое комбинирование типов списков.

```
<body>
<ol>
<lh><em>Planets of the Solar System:</em><br>
<li>Mercury
<ul> <ul>
<li>Roman god of commerce, travel, and thievery
<li>Dictionary Definition
<dl>
<dt>Mercury
<dd>The smallest of the planets and the one
nearest the sun, having a sidereal period of
revolution about the sun of 88.0 days at a
mean distance of 58.3 million kilometers (36.2 million miles) and a mean
radius of appropriately
2,414 kilometers (1,500 miles).
</dl>
</ul> </ul>
<li>Venus
<ul> <ul>
<li>Roman goddess of sexual love and physical beauty
<li>Dictionary Definition
<dl>
<dt>Venus
<dd>The second planet from the sun, having an
average radius of 6,052 kilometers (3,760 miles),
a mass 0.815 times that of Earth, anda sidereal
period of revolution about the sun of 224.7 days
at a mean distance of approximately 100.1 million
kilometers (67.2 million miles).
```

</dl>

</body>
Отступ для каждого списка устанавливает программа просмотра, однако, если это необходимо, для его увеличения можно добавить "пустой" список:

Простой список
 Простой список

нужно написать

Список с увеличенным отступом
 Список с увеличенным отступом

Дополнительные возможности форматирования списков

Вы можете создать собственные маркеры для использования в ненумерованных списках. Контейнер ul информирует браузер о необходимости интерпретировать заключенный в нем текст как неупорядоченный список. Если не нужны стандартные маркеры, то тэги <li> не используется. Вместо него нужно записать код, определяющий ваш новый маркер, например:
<img src= "cube.gif" align=top>Red
Тэг <img> указывает на графический файл используемого маркера и метод выравнивания изображения.

1.5. Гиперссылки

Значение ссылок во Всемирной паутине трудно переоценить. Читая книгу, вы всегда имеете ее под рукой. Работая в WWW, вы понятия не имеете, где находится та или иная нужная вам страница. Поэтому ссылки здесь являются единственной возможностью перейти от одного документа к другому.

Гипертекст и гипермедиа

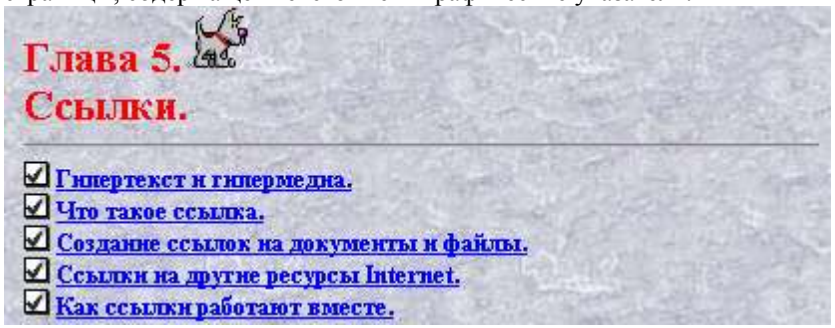
Гипертекстовый документ – это документ, содержащий ссылки на другие документы, позволяющие при помощи нажатия кнопки мыши быстро перемещаться от одного документа к другому. Гипермедийный документ основан на гипертекстовом документе, но в дополнение к тексту содержит разнообразную графику, видео и

аудиообъекты. В таких документах в качестве ссылок часто используются изображения.

Ссылка состоит из двух частей. Первая из них – это то, что вы видите на Web-странице; она называется указатель (anchor). Вторая часть, дающая инструкцию браузеру, называется адресной частью ссылки (URL-адресом) Когда вы щелкаете мышью по указателю ссылки, браузер загружает документ, адрес которого определяется URL-адресом.

Указатели.

Указатели бывают двух типов – текстовые и графические. Текстовые указатели представляют собой слово или слова, подчеркнутые прямой линией. На рисунке приведен пример Web-страницы, содержащей текстовые и графические указатели.



Графические указатели в принципе мало отличаются от текстовых. Они не подчеркиваются и не выделяются цветом, но вокруг них можно сделать рамку.

Маркеры списка (Bullets).

Графические указатели часто имеют вид точки, звездочки или другого маркера. Как правило, строка рядом с маркером также является указателем, имеющим тот же самый URL-адрес. Второй частью ссылки является URL-адрес. Это не что иное, как адрес Web-страницы, которая будет загружена, если щелкнуть на указателе ссылки кнопкой мыши. Указание адреса может быть относительным или абсолютным.

Относительные указатели.

URL-адрес файла, расположенный на том же компьютере, что и документ, в котором находится указатель этой ссылки, называется относительным. Это означает, что если браузер загрузил страницу, находящуюся по адресу http://www.mysite.com/page, то относительный адрес /picture подразумевает адрес http://www.mysite.com/page/picture, т. е. подкаталог, расположенный на той же машине. Абсолютные указатели. URL-адрес, полностью определяющий компьютер, каталог и файл, называется абсолютным. В отличие от относительных адресов,

абсолютные адреса могут ссылаться на файлы, расположенные на других компьютерах.

Создание ссылок на документы и файлы

Для создания ссылки необходимо сообщить браузеру, какой элемент страницы является указателем, и указать адрес документа, на который ссылаетесь. Оба действия выполняются при помощи тэга <a>. Тэг <a> имеет атрибут href, в котором размещается URL– адрес.

anchor

Для облегчения работы с относительными указателями в HTML введен тэг <base>. Он располагается в начале документа и содержит URL- адрес, относительно которого в документе построена вся адресация. Это указание влияет на любой нижестоящий тэг, включая <a>, <img> и т.д. Если, например, вы вставите строку

<base href="http://www.server.com">

вся относительная адресация в дальнейшем будет базироваться на этом адресе. Так, относительный указатель на файл "images/face.gif будет подразумевать адрес http://www.server.com/images/face.gif. Если тэг <base> отсутствует, относительная адресация строится на URL-адресе текущего документа.

Некоторые программы просмотра выводят в маленьком окошке текст, содержащийся в атрибуте title тэга <a>, если задержать курсор мыши на указателе ссылки.

 Me Page

Создание графического указателя аналогично созданию текстового указателя. Вместо текста в тэге <a> нужно разместить имя файла изображения. В следующей строке кода атрибут href опять указывает адрес домашней страницы, только вместо текстового анкера при помощи тэга <img> создается графический указатель, представляющий собой маленькую каринку

<img src="comp.jpg">

Кроме ссылок на другие документы, часто бывает полезно включить ссылки на разные части текущего документа. Для построения внутренней ссылки сначала нужно создать указатель, показывающий место назначения. Для этого нужно разместить там указатель и дать ему имя при помощи атрибута name тэга <a>. При этом атрибут href не используется, и браузер не выделяет содержимое тэга <a>.

Middle Section in Web page

После того как указатель получил имя, можно приступить к созданию ссылки на него. Для этого, вместо указания в атрибуте href адреса документа, как это делалось ранее, поместим туда имя

указателя со специальным префиксом (#), говорящим о том, что это внутренняя ссылка. Jump to the middle
Теперь, если пользователь щёлкнет кнопкой мыши на словах Jump to the middle, программа просмотра выведет среднюю часть документа, причем указатель ссылки будет расположен в верхней строке окна.

Файлы и другие встраиваемые объекты

Когда пользователь щелкает мышью на ссылке, указывающей на другую Web-страницу, она выводится непосредственно в окне браузера. Если же ссылка указывает на документ иного типа, программа просмотра принимает документ и затем решает, что с ним делать потом. Связь с электронной почтой (e-mail). Если вас интересует отклик читателей на содержание вашего документа, вы захотите поместить на странице свой адрес e-mail.
Send me E-mail.
После щелчка мышью на ссылке на ваш адрес браузер откроет собственное окно для работы с электронной почтой.
Связь с FTP.
Ссылка на сайт FTP похожа на другие гипертекстовые ссылки. Вместо http: Нужно поставить ftp:, а вместо URL-адреса – //sitename/path. Вы должны удостовериться, что сайт, на который вы ссылаетесь, разрешает анонимный доступ. Практически все браузеры работают с FTP без всяких проблем. Ваша ссылка может иметь следующий вид:
You can get the FAQ <a href="ftp://ftp.mysite.com/pub/FAQ">here< /a>.
Если не указывать имя файла, браузер выведет перечень всех файлов в каталоге.

Таблица 7.

Ссылка на	Формат	Пример
Web-страницу	http://sitename/	http://www.mysite.com/
e-mail	mailto:address	mailto:me@mysite.com
Newsgroup	news:newsgroupname	news:news.newusers.questions
FTP	ftp://sitename/	ftp://ftp.mysite.com/
Telnet	telnet://sitename/	telnet://bbs.mysite.com/

1.6. Таблицы

Для лучшего представления информации вы можете использовать таблицы. Элемент table представляет собой тэг-контейнер, в котором размещается содержимое таблицы. Таблица строится по строкам: для обозначения строки используется контейнер tr, для обозначения заголовков столбцов (строк) – контейнер th, а для данных в ячейках – контейнер td. Заголовки выделяются полужирным шрифтом и центрируются в своих ячейках. Данные имеют обычный шрифт и выравниваются по левой стороне ячейки.

Таблица 8.

Тэг	Описание
<table></table>	Контейнер таблицы.
<tr></tr>	Контейнер строки таблицы. Допускается отсутствие закрывающего тэга.
<td></td>	Контейнер ячейки таблицы. В ячейку можно включить другую таблицу. Закрывающий тэг может быть опущен.
<th></th>	Контейнер заголовка, располагающегося обычно в первой строке, либо в первом столбце таблицы. Закрывающий тэг также необязателен.

Пример:

```
<table border=2>
<tr>
<th>Colors</th><th>Of</th><th>The Rainbow</th>
</tr>
<tr>
<td>Red</td><td>Orange</td><td>Yellow</td>
</tr>
<tr>
<td>Green</td><td>Blue</td><td>Violet</td>
</tr>
</table>
<hr>
<table border=2>
<caption>My Favorite Groups</caption>
```

<tr><th>Rock</th>
<td>Pink Floyd</td>
<td>Led Zepplin</td>
<td>The Dobbie Brothers</td></tr>
<tr><th>Soft</th><td>Simon and Garfunkel</td>
<td>Peter, paul, & Mary</td>
<td>Neil Young</td></tr>
<tr><th>New Age</th><td>Enya</td>
<td>Clannad</td>
<td>Steamroller</td></tr>
</table>

Если надо разнообразить заголовки, то можно применить тэги физического форматирования. Таблицы всегда должны быть прямоугольными. Другие формы не допускаются.

Размещение данных внутри ячеек

При помощи атрибутов align и valign можно по-разному размещать данные относительно границ ячейки. Эти атрибуты используются совместно с элементами <caption>, <tr>, <TH> и <td> в самых различных комбинациях. Ниже приведены значения атрибутов для перечисленных элементов.

Таблица 9.

<caption>	Атрибут align может иметь значения top и bottom (по умолчанию – top); размещает заголовок таблицы сверху или снизу.
<tr>	Атрибут align может принимать значения left, center и right (по умолчанию – left для данных и center для заголовков); он определяет горизонтальное выравнивание данных в ячейках и действует на всю строку, если не отменяется тем же атрибутом в отдельной ячейке. Атрибут valing может иметь значения top, bottom, middle и baseline (по умолчанию – middle); он регулирует положение данных относительно верхней и нижней границ ячейки и влияет на всю строку, если не отменяется таким же атрибутом в отдельной ячейке, baseline применяется ко всем элементам строки и выравнивает их по базовой линии.
<th>	Атрибут align может принимать значения left, center и

	right (по умолчанию – center). Атрибут valing может иметь значения top, bottom и middle (по умолчанию – middle).

Применение этих атрибутов:

```
< table border>
<caption align=bottom>A Really Ugly Table</caption>
<tr>
<th></th><th>##########</th><th>##########</th>
<th>##########</th>
</tr>
<tr align=right>
<th>Row 1</th><td>XX<br>XX</td><td align=center>X
</td><td>XXX</td>
</tr>
<tr valing=baseline>
<th aling=left>Second Row</th><td>XXX<br>XXX</td><td>XXX</td>
<td>XXX<br>XXXXX<br>XXX</td>
</tr>
<tr aling=left>
<th>This Is<br>The Bottom Row of <br>The Table</th>
<td valing=bottom>XXXXX</td>
<td valing=top>XXX<br>XXXXX</td>
<td valing=middle>XXXXX</td>
</tr>
</table>
```

.

При отсутствии атрибута border рамка не прорисовывается. Такие таблицы можно использовать для табличной верстки старниц.

Объединение ячеек

Смежные ячейки таблицы могут объединяться с целью размещения большего количества данных. Например, в таблице из пяти строк и пяти столбцов все ячейки первой строки можно объединить и поместить в этой строке красивый заголовок таблицы. Возможно также объединение нескольких строк или создание пустой прямоугольной области. Для соединения двух смежных ячеек в одном столбце нужно использовать атрибут rowspan тэга <th> или <td>, например:

<td rowspan=2>

Для объединения двух смежных ячеек в одной строке нужно использовать атрибут colspan тех же тэгов, например:

<td colspan=2>

Пустые ячейки

Существует разница между пустой ячейкой и ячейкой с невидимым содержанием

Таблица 10.

width	Определяет ширину всей таблицы в пикселах, либо в процентах от ширины окна браузера. Может также использоваться для отдельной ячейки.
height	Определяет высоту всей таблицы в пикселах, либо в процентах от высоты окна браузера. Может также использоваться дня отдельной ячейки.
border	Устанавливает ширину рамки таблицы в пикселях, например, border=2.
cellpading cellspasing	Добавляют свободное пространство между данными внутри ячейки и ее границами и, соответственно, между ячейками внутри всей таблицы. Если рамка отсутствует, то результат их действия одинаков.

Использование цветов

Вы можете изменить цвет фона ячейки при помощи атрибута bgcolor перед размещением в ней текста или изображения, а также использовать атрибут bordercolor для изменения цвета рамки ячейки. Тэги <table>, <td>, <th> и <tr> допускают использование в них указанных атрибутов. Теги <table> и <tr> поддерживают аттрибут background.

Альтернативные способы представления табличных данных

• Список
• Использоваие изображения.
• Предварительно отформатированный текст.

<pre>

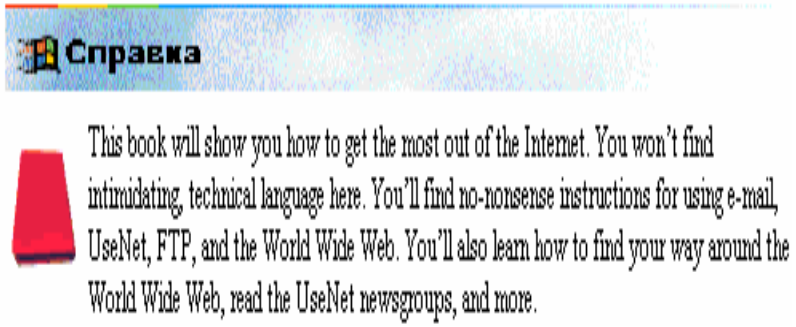
```
+------------------+----------------------+--------------------+
| Offense | Defense | Goalie |
+------------------+----------------------+--------------------+
| Husmann | O’Donnell | |
| Popplewell| |
| McGilly | Longo | Weinberg|
| Donahue | Seymour | |
```

```
| Camillo | Walsh | |
+------------------+----------------------+--------------------+
</pre>
```

Использование изображения в качестве заголовка таблицы

Вы можете украсить свою таблицу, поместив в ее заголовок изображение вместо текста.

```
<table width=500 cellspacing=0 cellpadding=2 border=0>
<tr>
<th colspan=2>
<img src="cloud.gif" border=0 height=25 width=500>
</th>
</tr>
<tr>
<td valing=top>
<img src="an59.gif">
</td>
<td valing=top>
This book will show you how to get the most out of the
Internet. You won't find intimidating, technical language
here. You'll find no-nonsense instructions for using e-mail,
UseNet, FTP, and the World Wide Web. You'll also learn how
to find your way around the World Wide Web, read the UseNet
newsgroups, and more.
</td>
</tr>
</table>
```



1.7. Использование графики

Изображения могут сделать текст вашего документа более содержательным. Изображения помогают лучше передать суть и содержание документа.

Вставка изображения в документ

Для вставки изображения на страницу следует воспользоваться тэгом <img> совместно с атрибутом src, поместив их в надлежащее место вашего HTML-документа:

```
<img src="URL">
```

По умолчанию браузер выводит изображение немедленно после текста или другого объекта, описанного предыдущими инструкциями

```
<body>
<p>
<img src="book.gif">
This text immediately follows the image.
<p>
This text is interrupted <img src="book.gif">
by the image.
<p>
In this case, the image appears inline after this text.
<img src="book.gif">
</body>
```

Предусмотрите хранение файлов графики в одном определенном каталоге (не корневом) вашего Web-сайта. Тогда вы сможете использовать относительную адресацию в комбинации с тэгом <base>, а не указывать полный URL-адрес. Атрибут src определяет не только какое изображение, но и где хранится это изображение. В примере, который приведен ниже. src="copper.gif" означает, что браузер будет искать изображение с именем copper.gif в той же самой папке (или каталоге), где непосредственно расположен html документ.

-src="images/copper.gif" означает, что изображение из папки нижнего уровня, по сравнению с папкой HTML- документа, который запросил его. Это можно продолжить по уровням вниз настолько это необходимо.

- src="../copper.gif" означает, что изображение находится в папке верхнего уровня, по сравнению с папкой HTML-документа, который запросил его.
- src="../../copper.gif" означает, что изображение в папке на два уровня выше, чем папка HTML-документа, который запросил его.
- src="../images/copper.gif" означает, что изображение и HTML-документ в папках одного уровня.
- src="../../../other/images/copper.gif" - …

Выравнивание текста по краю изображения

По умолчанию, когда изображение вставляется в строку текста, строка выравнивается по низу изображения. Изменить эту установку можно при помощи атрибута align тэга <img>.

Таблица11.

Значение	Описание.
top	Выравнивает текст по верху изображения.
middle	Выравнивает текст по середине изображения.
bottom	Выравнивает текст по низу изображения.

```
<body>
<p>
<img src="comp.jpg" align=top>
This text is aligned with the top of the image.
</p>
<p>
<img src="comp.jpg" align=middle>
This text is aligned with the middle of the image.
</p>
<p>
<img src="comp.jpg" align=bottom>
This text is aligned with the bottom of the image.
</p>
</body>
```

Позиционирование изображения на странице

По умолчанию программа просмотра выводит изображение в текущей строке. Текст не "обтекает" его. Однако при помощи атрибута align тэга <img> изображение можно сделать "плавающим", т. е. заставить текст расположиться вокруг изображения.

Таблица 12.

Значение	Описание
left	Обтекаемое текстом изображение прижато к левой стороне окна браузера.
right	Обтекаемое текстом изображение прижато к правой стороне окна браузера.


```
<p>
<img src="comp.jpg" align=left>
This text will wrap around the right-hand and bottom of the image.
This text will wrap around the right-hand and bottom of the image.
This text will wrap around the right-hand and bottom of the image.
<img src="comp.jpg" align=right>
This text will wrap around the left-hand and bottom of the image.
This text will wrap around the left-hand and bottom of the image.
This text will wrap around the left-hand and bottom of the image.
</p>
```

При помощи тэга <img> программе просмотра можно сообщить размеры изображения, которое затем размещено на странице:

```
<p>
<img src="comp.jpg">
<img src="comp.jpg" width=140 height=100>
</p>
```

Для указания размеров изображения (в пикселях) служат атрибуты height и width тэга <img>. Если указанные размеры не совпадают с размерами изображения, программа просмотра изменяет масштаб изображения. Альтернативное описание изображения:

```
<img src="filename" alt="Description">
```

Если браузер читателя не выводит изображение, на его месте будет помещено альтернативное описание. Если изображение выводится, это описание будет находиться на месте иллюстрации до начала ее загрузки. Еще лучше использовать эту возможность совместно с указанием размеров.

Если указатель мыши задержать на иллюстрации на одну-две секунды, этот же текст будет выведен в специальном всплывающем окошке в виде подсказки.

Помещение изображения в рамку.

Эта простая операция предполагает применение атрибута border тэга <img>. По умолчанию программа просмотра использует рамку, которая указана в соответствующей ссылке. Введите ширину рамки в пикселах, как показано в примере:

```
<body>
<img src="comp.jpg" border=0>
<img src="comp.jpg" border=5>
<img src="comp.jpg" border=10>
</body>
```

Отделение изображения от текста.

Вам может не понравиться, что текст слишком близко подходит к изображению Если это так, используйте атрибуты vspace и hspace для указания расстояния (по вертикали и горизонтали) между кромкой текста и краями иллюстрации.

```
<body>
<p>
<img src="comp.jpg" aling=left hspace=40 vspace=20>
This text will wrap around the right-hand and bottom of the image.
This text will wrap around the right-hand and bottom of the image.
This text will wrap around the right-hand and bottom of the image.
This text will wrap around the right-hand and bottom of the image.
This text will wrap around the right-hand and bottom of the image.
This text will wrap around the right-hand and bottom of the image.
</p>
<p>
<img src="comp.jpg" align=left>
This text will wrap around the left-hand and bottom of the image.
This text will wrap around the left-hand and bottom of the image.
This text will wrap around the left-hand and bottom of the image.
</p>
</body>
```

Карта ссылок

Сегодня многие Web-страницы располагают интересной разновидностью меню – изображениями-картами, т.е. изображениями, чувствительными к нажатию кнопки мыши (imagemaps – от англ. Слов image – изображение и map – карта, план.). Разные части изображения на странице могут быть связаны с разными URL-адресами. Так как пользователь должен знать, где расположены "чувствительные" области изображения, они часто выделяются рамками, тоже являющимися частью изображения. Изображения-карты бывают двух типов: обслуживаемые сервером и программой-клиентом (браузером). Синтаксис определения изображения-карты, обслуживаемой клиентом следующий:

```
<map name="mapname" >
<area [shape="shape"] coords="x,y,..." [href'="URL"] [nohref]>
</map>
```

Определение начинается тэгом <map> и заканчивается тэгом </map>. Для того чтобы ссылаться на это определение позже в тэге <img>, оно должно иметь имя, задаваемое при помощи атрибута name. Для задания чувствительных областей используется тэг <area>.

Атрибуты тэга <area>.

Таблица 13.

Атрибут	Описание

shape	Определяет форму чувствительной зоны. Имеет значения rect, poly, circle, rect.
coords	В этом атрибуте перечисляются через запятую пары координат – x1,y1,x2,y2. Между парами также ставится запятая.
href	Определяет URL-адрес ссылки. Относительные адреса задаются относительно документа, содержащего тэг <map>. Если в этом же документе используется тэг <base>, адресация рассчитывается относительно URL, указанного в этом тэге.
nohref	Указывает нечувствительную зону, т.е. зону, не связанную с другими документами или ресурсами. Атрибуты nohref и href взаимоисключающие.

Ссылка на изображение-карту:

```
<map name=mymap>
<area shape=rect coords="0,0,100,100" href=item1.html>
<area shape=rect coords="101,0,200,100" href=item2.html>
<area shape=rect coords="201,0,300,100" href=item3.html>
</map>
<img src=mymap.gif usemap=#mymap>
```

Этот тэг загружает изображение mymap.gif Атрибут usemap указывает на имя определения изображения-карты, которое было присвоено при помощи атрибута name тэга <map>.

1.8. Вставка объектов мультимедиа

Вставка видео

Для вставки видео-фрагментов в код HTML-страницы и его дальнейшего просмотра средствами браузера используется уже известный тэг <img> с новым атрибутом dynsrc. Например, строка кода: <img src=”image.gif” dynsrc=”video.avi”> выведет в окне браузера сначала картинку image.gif и весь текст, а затем, пока пользователь будет читать текст, догрузит видеофайл и запустит его вместо картинки. При помощи атрибута loop можно задать количество раз воспроизведения клипа. Значение этого атрибута равное –1 или

символьной константе infinite, позволяет прокручивать данное изображение не ограниченное количество раз. Данный атрибут принимает целочисленные значения. Атрибут start позволяет указать событие после совершения которого начнется воспроизведение клипа. Например, <img src=”image.gif” dynsrc=”video.avi” start=mouseover/fileopen> Для просмотра видеоклипа также можно использовать универсальный тэг <embed>. Он позволяет воспроизводить и видео и аудио клипы в обоих браузерах. Используя такие атрибуты данного тэга, как height и width вы можете изменять высоту и ширину видеоклипа в пикселах. Используя ключевое слово autostart можно автоматически загружать клип на выполнение сразу после его загрузки. Атрибут loop=yes задает бесконечное количество воспроизведений данного клипа.

Вставка аудио

Один из наиболее популярных способов вставки аудиоклипов – использование простой ссылки на звуковой файл:

<img src=”images/sound.gif” align=left>

 Мое приветствие

В данном случаи загрузка и последующее воспроизведение данного аудиофайла будет происходить только после щелчка мышкой по ссылке.На экран этот файл выведен не будет. Также используется тег <bgsound>. При использовании этого тега в вашем HTML-документе при его загрузке сначала происходит загрузка его текстовой и графической части и лишь затем, загрузка и воспроизведение аудиофайла указанного в атрибутах тега <bgsound>. Например:

<bgsound=”myaudio.wav” loop=10>

Значение атрибута loop равное infinite указывает на бесконечное количество повторений данного аудиофрагмента (пока пользователь не покинет страницу). Для вставки звукого файла может быть использован тег <embed>. При выводе аудиоклипа броузер выводит на экран набор кнопок, позволяющий управлять воспроизведением клипа. Эти кнопки управления появятся на вашем экране только после того, как будет загружена вся HTML-страница, и скачан целиком аудио файл.

<embed src=”lion.wav” height=60 width=140 autostart=true loop=true>

Данная строка выводит звуковой файл и устанавливает размер панели управления клипа. Атрибут loop позволяет задать количество раз проигрывания клипа. Иногда еще используется атрибут hidden=true (по умолчанию – false), который скрывает вывод на экран управляющик кнопок. В этом случаи просто звучит звук и посетители не могут им управлять.

1.9. Таблицы стилей

Таблицы стилей HTML предназначены для:

• Изменение расстояний между строками, словами и отдельными символами.

• Установка левого, правого, верхнего и нижнего полей элемента (блока текста контейнера HTML).

• Установка отступа элемента.

• Изменение размера, стиля и других атрибутов шрифта элемента.

• Установка рамки вокруг элемента.

• Включение фонового изображения и фонового цвета в элемент.

Связывание документа с таблицей стилей

Большим преимуществом таблиц стилей HTML является возможность отделить операцию форматирования от содержания документа. Сначала вы определяете, как должен выглядеть текст в том или ином месте страницы, а затем вводите сам текст. Если вы позднее решите, например, заменить цвет шрифта заголовков на синий, для этого будет достаточно поменять только стиль этих заголовков. Делать изменения в тексте нет необходимости.

Способы применения таблицы стилей.

Таблица 14.

Связывание (Linking).	Можно связать HTML-документ с таблицей стилей, хранящейся в отдельном файле.
Встраивание (Embedding).	Можно встроить таблицу стилей в HTML-документ с помощью контейнера <style>.
Задание стиля для Отдельного фрагмента документа (Inline).	

Можно определять элементы стиля "на лету", т. е. указывать их в тэгах HTML, например, в тэге абзаца <p>. Создания таблицы стилей в виде отдельного файла для применения его ко всем страницам сайта. Этот метод упрощает создание сайта. Вы можете даже разработать единую таблицу стилей, которую могли бы использовать все сайты вашей организации. Хранить таблицу стилей следует в текстовом файле с расширением .css. Его можно создать при помощи любого текстового редактора. У вас не будет никаких трудностей. Для

связывания таблицы стилей с документом HTML используйте тэг <link> следующим образом:

```
<link rel=stylesheet href=www.myserver.com/mysheet.css
type="text/css">
```

Укажите в атрибуте href URL-адрес вашей таблицы стилей. Дайте атрибуту type значение "text/css", что позволит программам просмотра, не поддерживающим таблицы стилей, проигнорировать указанный адрес.

Встраивание таблицы стилей в документ.

Таблицу стилей необязательно хранить в виде отдельного файла. Ее можно встроить непосредственно в документ, однако в этом случае она будет действовать только внутри файла этого документа. Для распространения действия таблицы на другие документы ее необходимо скопировать в каждый из них.

Для включения таблицы стилей в документ воспользуйтесь контейнером <style>. Он размещается между тэгами <html> и <body>:

```
<head>
</head>
<style type="text/css">
Style definitions go here
</style>
<body>
</body>
```

Тэг <style> имеет единственный атрибут type, определяющий тип mime (Multipurpose Internet Mail Extension, стандарт электронной почты Internet). Определяйте его как "text/css" для того, чтобы браузеры, не поддерживающие таблицы стилей, могли игнорировать тэг <style>. Ниже приведен конкретный пример тэга <style>.

```
<style type="text/css">
h1 {color: blue;}
</style>
```

Задание стиля для отдельного фрагмента документа.

Вы можете определять стиль, что называется, "на лету", оперативно внося требуемые изменения. Например, если вы определили стиль документа с заголовком одного цвета, а потом решили выделить цветом какой-то элемент заголовка, вы можете это сделать внутри тэга заголовка, не изменяя общий стиль документа.

Такой метод действует внутри тэга, где определен или переопределен стиль при помощи атрибута style. Он поддерживается всеми подчиненными тэгами тега <body>. Для оперативного определения стиля добавьте к нужному тэгу атрибут style и присвойте ему строковое значение, указывающее новый стиль:

```
<h1 style="color: blue">
```

Используя атрибут style с тэгом , можно определять стиль части документа, расположенной в контейнере . Это работает благодаря принципу "наследования. Например, если вы хотите установить цвет шрифта для целого блока тэгов синим, вы можете расположить эти тэги внутри контейнера и определить цвет шрифта текста следующим образом:

```
<div style="color: blue;">
<h1>This is a heading</h1>
<p>This is a paragraph. It will look blue in the user's browser</p>
</div>
```

Для изменения стиля нескольких слов или даже символов можно использовать атрибут style совместно с тэгом , например:

```
This is a <span style="color: blue;">simple</span> block of text
```

Синтаксис таблиц стилей

Таблицы стилей хранятся в текстовых файлах, удобных для редактирования. Их нетрудно создавать вручную, однако, как и для HTML-документов, существуют специальные редакторы таблиц стилей. Таблицы стилей позволяют определять стиль для одного или нескольких тэгов. Например, вы можете создать таблицу стилей, определяющую стили для тэгов <h1>, <h2>, <p> и . Каждое определение называется правилом (rule). Правило содержит селектор (тэг HTML), за которым следует декларация (определение стиля). Селектор является связующим звеном между определением и тэгом. Ниже приведен пример правила, указывающего стиль для каждого из тэгов заголовка <h1>: hl {color: blue;}. Декларация заключается в фигурные скобки. Каждая декларация имеет две части: название свойства и присваиваемое ему значение, разделенные двоеточием. В HTML включены десятки свойств (font-size, font-style, color, margin-right и т. д.). Каждое свойство может принимать несколько значений, одно из которых приписывается ему по умолчанию. В предыдущем примере было указано лишь одно свойство color. Однако ничто не мешает определить целый ряд свойств в одном тэге, отделив их друг от друга точкой с запятой:

```
Hl {color: blue; font-size: 12pt; text-line: center;}
```

В этом примере программа просмотра выведет каждый заголовок первого уровня синим шрифтом размером 12 пунктов и выровняет их по центру окна. Для всех прочих свойств будут использоваться значения по умолчанию (например, свойству font-style будет присвоено значение normal).

Группирование селекторов.

Если вы хотите определить один и тот же стиль для нескольких тэгов, вы можете перечислить их в отдельном списке:

p {font-size: 12pt;}
ul {font-size: 12pt;}
li {font-size: 12pt;}

HTML позволяет сделать то же самое и в более компактном виде – в
одной строке:
p, ul, li {font-size: 12pt;}

Запятая здесь является обязательным элементом. Если она опущена, смысл правила изменится. Комментирование таблицы стилей. По мере сложнения таблицы стилей, скорее всего, понадобится включить в нее дополнительные сведения о назначении того или иного правила. Комментарии располагаются между символами /* и */ и игнорируются программами просмотра, например:
body {margin-left: lin;} /* Отступ на 1 дюйм */
h1 {margin-left: -lin;} /* Сдвиг влево на 1 дюйм */
h2 {margin-left: -lin;} /* Сдвиг влево на 1 дюйм */
Комментарии могут иметь вид блоков текста, дающих подробное описание стиля страницы, например:
/*------

Свойство margin-left тэга <body> установлено в 1 дюйм. Так как все внутренние тэги наследуют это значение, то вся страница будет иметь отступ в 1 дюйм. Заголовки первого и второго уровней имеют отрицательный отступ (-1 дюйм), т.е. будут прижаты к левой границе окна браузера.
-----*/
body {margin-left: lin;} /* Отступ на 1 дюйм */
h1 {margin-left: -lin;} /* Сдвиг влево: на 1 дюйм */
h2 {margin-left: -lin;} /* Сдвиг влево на 1 дюйм */
Наследование свойств.

В HTML подчиненные тэги наследуют некоторые свойства родительских тэгов. Например, все тэги контейнера <body> (<p> и <ul>) будут обладать некоторыми свойствами тэга <body>. Точно так же тэг
наследует свойства тэга <ul>. Рассмотрим следующий код:
<style type="text/css">
p{color: blue;}
</style>
<body>
<p>Hello. This is a paragraph of text. This is emphasized<p>
</body>

Таблица стилей этого документа устанавливает цвет в тэге <p> иним, однако, цвет для тэга явно не определен (по умолчанию – это черный цвет). Здесь не о чем беспокоиться, так как этот тэг

находится в родительском контейнере <p> и наследует таким образом синий цвет.

Применение контекстных селекторов.

Иногда возникает необходимость определения двух (и более) стилей для одного тэга. Например, может понадобиться указание двух стилей для эга <li>: один для случая, когда он подчинен тэгу <ul>, и второй, когда он подчинен тэгу <ol>. Это возможно сделать с помощью контекстных селекторов. Контекстный селектор определяет точную последовательность тэгов, для которых будет применен тот или иной стиль. Другими словами, вы можете указать, что какой-то стиль должен применяться, например, в тэге <li> только в том случае, если этот тэг является подчиненным тэгу <ol>:

```
ol li{list-style-type: decimal;}.
```

Для того же тэга <li> можно определить другой стиль, действительный только в случае подчиненности тэгу <ul>:

```
ul li {list-style-type: square;}
```

Заметьте, что список селекторов не разделен запятыми. В противном случае всем тэгам списка будет приписан один и тот же стиль.

Почему таблицы стилей HTML называются каскадными

В рекомендациях W3C таблицы стилей называются "каскадными таблицами стилей" потому, что для верстки Web-страницы можно применять не одну, а сразу несколько таблиц. При этом программа просмотра сама определяет последовательность использования таблиц и разрешает конфликты между ними по принципу каскадирования. Как это работает? Каждому правилу браузер приписывает весовой коэффициент. При интерпретации каждого тэга программа просматривает все правила этого тэга и сортирует их по величине весового коэффициента. Выигрывает самое "весомое" правило.

Существуют следующие общие принципы разрешения конфликтов между таблицами стилей: Таблица стилей автора страницы "весомее" таблицы стилей читателя, которая, в свою очередь, "весомее" установок браузера по умолчанию. Старшинство типов таблиц стилей в документе (по убыванию): текущее задание стиля (inline), встраивание (embedding) связывание (linking).

Использование классов в таблицах стилей

Классом называется определение нескольких стилей одного элемента, каждый из которых может использоваться в нужном месте страницы. Например, вы можете определить три вариации стиля заголовка h1. Определение вариаций похоже на указание стиля, только к названию тэга добавляется произвольное имя класса, отделенное точкой:

```
h1.blue {color: blue;}
```

h1.red {(color: red;}
h1.black {color: black;}
Теперь, включая в документ тэг <h1>, можно указать в нём конкретный стиль при помощи атрибута class:
<h1 class=red>Red Heading</h1>
Вы можете разрешить обратиться к какому-либо классу из любого тэга, если при определении данного класса опустить в селекторе имя тэга, например:
.red {color: red;}

1.10. Блочная верстка страниц

Отличия блочной верстки от табличной

Верстка блоками, с помошью такого тега как div, имеет ряд больших преимущест по сравнению с версткой таблицами (с помощью элемента table), среди них:
1. Дизайн сверстаный блоками быстрее загружается.
2. Содержимое блоков, в отличии от содержимого ячеек таблиц отображается по мере загрузки(содержимое таблиц же напротив, отображается только тогда, когда загрузиться все содержимое таблицы).
3. Код написаный блоками имеет более читаемый вид.

Управление положением блоков на странице

Для решения этого вопроса используется такое свойство как float. Свойство float может принимать три значения:
1. left - выравнивание элемента по левому краю страницы;
2. right - выравнивание элемента по правому краю страницы;
3. none - элемент страницы ни куда не перемещается, то есть будет там где он должен быть. Это значение используется по умолчани. Так же нам понадобиться рассмотреть еще одно свойство - clear.
Свойство clear может принимать четыре значения:
1. left - установка элемента ниже любого предыдущего, перемещенного влево блока;
2. right - установка элемента ниже любого предыдущего, перемещенного вправо блока;
3. both - установка элемента ниже любого предыдущего перемещенного блока;
4. none - нет ни каких ограничений на положение блока относительно перемещаемых блоков.

1.11. Формы

С помощью HTML вы можете создавать простые формы, предполагающие ответы типа "да" и "нет", вы можете разрабатывать сложные формы для заказов или для того, чтобы получить от своих

читателей какие-либо комментарии и пожелания. Форма представляет собой несколько полей, где пользователь может ввести некоторую информацию, либо выбрать какую-то опцию. После того как пользователь отправит информацию, она обрабатывается программой (скриптом), размещённой на сервере. Скрипт – это короткая программа, специально созданная для обработки каждой формы. Формы HTML позволят вам получать информацию от читателей. В форму могут заноситься мнения сторон дискуссионной группы.

Работа с тэгами форм

В HTML существует три тэга для создания различного типа полей в форме: <textarea>, <select> и <input>. Любое их количество может быть размещено в контейнере между тэгами <form> и </form>. Ниже дано их краткое описание (подробнее они будут рассмотрены в соответствующих разделах этой главы).

Таблица 15.

<textarea>	Определяет поле, в которое пользователь вводит многострочную текстовую информацию.
<select>	Позволяет пользователю сделать выбор в окне с полосой прокрутки либо в раскрывающемся меню.
<input>	Обеспечивает некоторые другие виды ввода информации: ввод одной строки текста, установку и сброс флажков (checkboxes), выбор переключателя (radio buttons) и нажатие кнопки для отправки данных или очистки формы.

Каждая форма начинается тэгом <form>. В нем нужно определить два атрибута, указывающих используемый скрипт и метод посылки данных:

action	Определяет URL, который примет и обработает данные формы. Если этот атрибут не определен, данные отправляются но адресу страницы, на которой помещена форма.
method	Указывает форме, как послать информацию соответствующей программе обработки (скрипту). Обычно он получает значение

	post, тогда информация формы посылается отдельно от URL. Если указано значение get, информация формы посылается вместе с URL. Get для посылки данных до 256 символов, а post для посылки данных длинной более 256 символов. Например:

```
<form method="post" action="/cgi-bin/comment script">
...
</form>
```

В этом примере дано указание браузеру отправить заполненную форму для обработки скриптом comment script, расположенным в каталоге cgi-bin вашего сервера, и использовать метод посылки post. Метод get – данные, посылаемые браузером серверу, включают модифицированный URL адрес: метод(http), servr:port и в конец добавляется символ ?, далее следует строка запроса. Метод post – запрос серверу посылается как mime-данные, при этом пробелы заменяются символом +, поля разделяются & и символы кодируются в шестнадцатиричном коде в виде %xx. На странице можно расположить любое число форм, однако нужно следить за тем, чтобы не поместить одну форму в другую.

Тэг <textarea> предназначен для построения поля для ввода многострочной текстовой информации. При помощи атрибутов rows и cols этого тэга можно построить поле любого размера. В контейнере textarea допускается размещать любой текст, который будет выведен в поле ввода по умолчанию.Поле textarea удобно тем, что пользователь может ввести в него любое количество информации. Тэг <textarea> имеет следующие атрибуты.

Таблица 16.

name	Обязательный атрибут, определяющий название информации.
rows	Устанавливает высоту поля, т.е. число строк в нём.
cols	Устанавливает ширину поля, т.е. длину строки.

Хотя атрибуты rows и cols не являются обязательными, они не имеют определенных значений по умолчанию (для каждого браузера эти значения различны), поэтому лучше их всегда указывать. Данный тэг имеет еще один атрибут – wrep. Значения которые он может принимать приведены в Таблице 17.

Таблица 17.

off	Переход на новую строку только при нажатии на Enter. Данные отправляются одной строкой.
soft	Автоматический перенос текста на новую строку, при достижении границы текстового поля. При этом текст на сервер передается одной строкой. Это значение по умолчанию.
hard	Визуально так же как soft, но на сервер будет отправленно несколько строк.

```
<form>
<textarea name="comments" rows=4 cols=40>
Текст по умолчанию
В данном поле текст форматируется,
как в теге <pre>
</textarea>
</form>
```

Тэг <select> используется для создания всплывающего меню или списка опций с полосой прокрутки. Список опций и пункты меню располагаются внутри контейнера select. Как и тэг <textarea>, тэг <select> требует обязательного определения имени в атрибуте name. Количество опций указывается в атрибуте size. Ниже перечислены атрибуты тэга <select>.

Таблица 18.

name	Определяет имя поля
size	Определяет вертикальный размер окна для опций выбора. Если атрибут опущен или его значение равно 1, выводится всплывающий список опций(раскрывающийся список). Если указано число больше единицы, то опции выводятся в окне с полосой прокрутки. Если значение атрибута больше, чем фактическое количество элементов списка, добавляются пустые строки. При их выборе пользователем возвращаются пустые поля.
multiple	Этот атрибут позволяет производить выбор сразу

	нескольких опций.

Список опций включается в контейнер <select> при помощи тэгов
<option>. Этот тэг имеет два атрибута.

Таблица 19.

value	Указывает значение, возвращаемое программе обработки (скрипту) в случае выбора опции пользователем.
selected	Указывает на опцию, выделенную по умолчанию.

```
<form>
<select name="network">
<option selected value="ethernet"> Ethernet
<option value="token16"> Token Ring - 16MB
<option value="token4"> Token Ring - 4MB
<option value="localtalk"> LocalTalk
</select>
</form>
```

Тэг <select> можно использовать как дополнительное средство навигации. Для этого в него нужно включить список URL-адресов. После выбора одного из адресов и нажатия кнопки Отправить (Submit), скрипт, размещенный на вашем сервере или на машине читателя, загрузит требуемую страницу. Тэг <input>, в отличие от <textarea> и <select>, является одиночным тэгом. Он предназначен для сбора информации различными способами, включая текстовые поля, поля для ввода пароля, переключатели, флажки, кнопки для отправки данных (Submit) и для очистки формы (Reset, Clear). Тэг <input> располагает следующими атрибутами.

Таблица 20.

namesize	Указывает размер поля ввода в символах.
maxlenght	Определяет максимально возможное число символов, вводимых в поле.
value	Для текстового поля определяет текст, выводимый по умолчанию. Для флажков и переключателей указывает значение, возвращаемое программе обработки. Для кнопок отправки и очистки формы определяет надпись на кнопке.
checked	Устанавливает флажок или переключатель во включенное состояние по умолчанию. С другими типами тэгов <input> не употребляется.
type	Устанавливает тип поля ввода (см. ниже).

Атрибут type тэга <input> может принимать следующие значения:

1. text

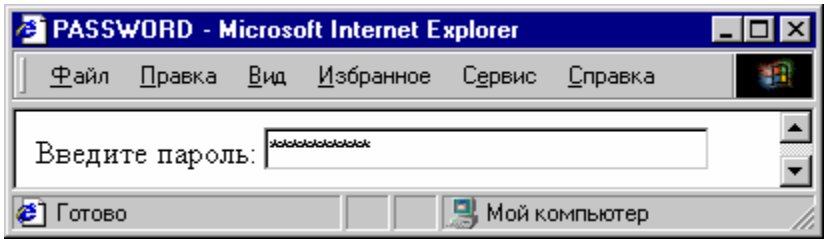
Является значением по умолчанию и предполагает создание одной строки для ввода данных. Для этого типа поля ввода употребляются атрибуты name (обязательный), size, maxlenght и value.

```
<form>
Please specify:<input type="text" name="network_other">
</form>
```

2. password

Позволяет заменять вводимые символы пароля звездочками. Для этого типа поля ввода используется атрибуты name (обязательный), size, maxlenght и value.

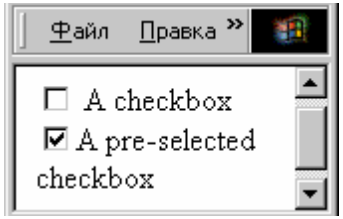
```
<form>
Введите пароль: <input type="password" name="secret_word" size="10"
maxlenght="30">
</form>
```



3. checkbox

Позволяет вывести поле для установки флажка в виде маленького квадратика, в котором может быть произведена отметка опции "галочкой". Может использоваться совместно с атрибутами NAME (обязательный), VALUE и CHECKED (определяет установленный по умолчанию флажок). Флажки обычно употребляются, когда можно выбрать сразу несколько опций из числа предложенных.

```
<form>
<input type="checkbox" name="checkbox1" value="checkbox_value1">
A checkbox
<input type="checkbox" name="checkbox2" value="checkbox_value2"
checked>
A pre-selected checkbox
</form>
```

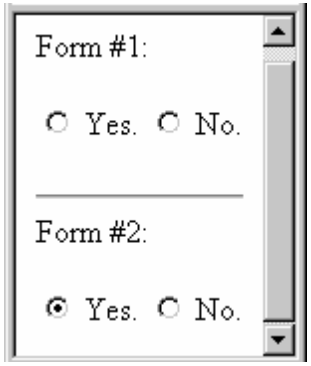


4. radio

Позволяет выбрать только одну из представленного числа опций. Переключатели можно группировать, задавая одно и то же значение атрибута name (обязательный). Также используются атрибуты value и checked. Значение атрибута value отправляется на сервер для обработки скриптом.

```
Form #1:
<form>
<input type="radio" name="choice" value="choice1"> Yes.
<input type="radio" name="choice" value="choice2"> No.
</form>
<hr>
Form #2:
<form>
<input type="radio" name="choice" value="choice1" checked> Yes.
```

<input type="radio" name="choice" value="choice2"> No.
</form>



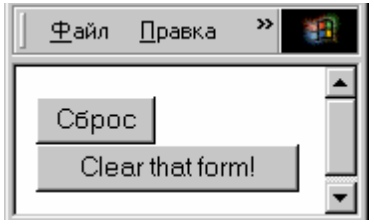
Если опций для выбора слишком много, для экономии места лучше использовать тэг <select>.

5. reset

Позволяет создать кнопку для очистки формы. Атрибут value может быть использован здесь для наименования этой кнопки (по умолчанию кнопка имеет надпись “Сброс” или “Reset”).

<form>
<input type="reset">

<input type="reset" value="Clear that form!">
</form>

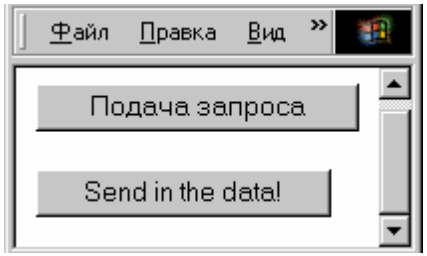


6. submit

Используется для создания кнопки, по нажатию которой введенные данные отправляются на сервер для обработки программой-скриптом. В атрибуте value может быть указано название для этой кнопки (по умолчанию – “Submit” или “Подача запроса”).

<form>
<input type="submit">

<input type="submit" value="Send in the data!">
</form>



7. button

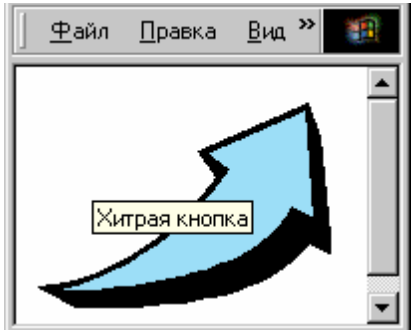
Используется для создания кнопки при нажатии на которую будет выполнятся отдельный скрипт. В качестве параметров можно задать размеры кнопки и ее подпись.

```
<form>
<input type="button" value="TEST" width="200" height="100">
</form>
```

8. image

Используется для создания кнопки в виде картинки, при щелчке левой кнопкой мыши по которой, все данные из формы будут отправлены на сервер. Таким образом это подобие кнопки submit, только оформленной по другому.

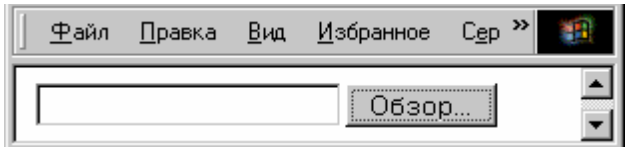
```
<form>
<input type="image" src="trigger.gif" align="right" alt="Хитрая
кнопка">
</form>
```



9. file

Используется для обращения к файловой структуре диска и выбора файла, который необходимо переслать по сети. Пересылка осуществляется в двоичном коде.

```
<form>
<input type="file">
</form>
```



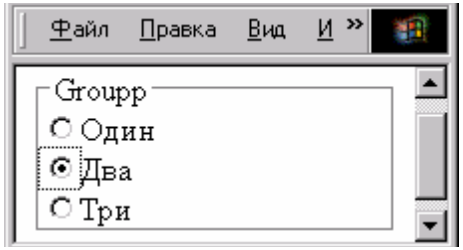
10. hidden

Используется для передачи данных на сервер из многоступенчатых форм без участия пользователя. Данные в этих полях на экране не отображаются.

Выделение нескольких элементов в группу

Парный тэг <fieldset> предназначен для выделения нескольких элементов управления в группу. В паре с этим тэгом работает парный тэг <legent>, при помощи которого задается подпись для этой группы. У тэга <legent> есть атрибут align, который может принимать значения left, right, center и предназначенный для выравнивания подписи группы относительно экрана.

```
<fieldset>
<legend align=left>Groupp</legend>
<input type="radio" name="radio1" value="1">Один<br>
<input type="radio" name="radio1" value="2">Два<br>
<input type="radio" name="radio1" value="3">Три<br>
</fieldset>
```



Размещение в документе форм нескольких типов

Расположение на странице форм разных типов способно сделать ее более выразительной и понятной. Размещение на странице нескольких форм требует их визуального разделения. Для этого можно использовать тэг <hr>, создающий горизонтальные линии, или тэг <img> для включения узкого горизонтально расположенного изображения.

2. Практика

Постановка задачи

Необходимо создать Web-сайт, состоящий из нескольких (не менее трех) связанных между собой статических HTML-страниц и использующий следующие возможности языка HTML:

• структурирование и оформление текста;
• списки;
• табличная и блочная верстка;
• графические и текстовые ссылки;
• вставка графических объектов, видео и аудио;
• формы с полями различных типов (списки, текстовые поля, переключатели и селекторы), а также кнопками, позволяющими очистить форму и передать ее содержимое на сервер.

Оформление страниц должно выполняться при помощи каскадных таблиц стилей. Навигация по сайту должна осуществляться с помощью меню. Все страницы необходимо создавать в виде исходных текстов, не пользуясь графическими средствами разработки. Для выполнения практической работы Вам необходимо иметь текстовый редактор (возможна работа в специальных редакторах Web-документов, например Adobe Dreamweaver), несколько браузеров для просмотра Ваших страниц (Internet Explorer, Mozilla Firefox, Opera и другие). Ниже, на Рис. 1-3, приведены примеры нескольких страниц. Подробные инструкции по созданию описанного выше сайта приведены во второй части каждой практической работы (см. раздел “Задание”). В первой части каждой практической работы даны простые упражнения, на примере которых можно рассмотреть отдельные элементы языка HTML.

2.1. Практическая работа №1

Упражнение 1. Изменеие строки заголовка

1. Откройте текстовый редактор. Введите текст:

```
<html>
<head>
<title>Упражнение 1</ title>
</head>
<body>
--- Это моя страница ---
</body>
</html>
```

2. Сохраните документ с именем Ex1.html в рабочей папке
3. Вызовите браузер. Откройте созданный файл. (File Open …)

Упражнение 2. Использование заголовков в документе

1. Перейдите в текстовый редактор. Введите текст:

```
<html>
<head>
<title>Упражнение 2</ title>
</head>
<body>
<h1 align=center> Заголовок первого уровня</h1>
здесь идет обычный текст
<h6 align=center> Заголовок последнего уровня</h6>
</body>
</html>
```

2. Сохраните документ с именем Ex2.html в рабочей папке.
3. Вызовите браузер. Откройте созданный файл. (File □ Open □…).

Упражнение 3. Логическое форматирование текста на абзацы и отделение их друг от друга горизонтальными линиями.

```
<html>
<head>
<title>Упражнение 2</ title>
</head>
<body>
<p>
Здесь
Вы
видите все
подряд<br>здесь<br>прервали<br>строку<br>
Это предложение отделено от следующего </p> Это другой параграф.
<p> Это один параграф </p><p> Это другой параграф </p>
<h1> <i>Линии</i></h1>
<hr width=50>
< hr width =100>
< hr width =20% align=left size=1>
< hr width =50% align=right size=5>
< hr width =100% align=center size=8 color="#ff0000" noshade >
</body>
</html>
```

Сохраните документ с именем Ex3.html в рабочей папке.

Упражнение 4. Использование упорядоченных и неупорядоченных списков

```
<html>
<head>
```

<title>Упражнение 4</ title>
</head>
<body>
<h2> Пример упорядоченного списка</h2>
Единицы измерения информации
<ol start=2 type=1>
биты
байты
<l>ікилобайты

....
<li value=7>мегабайты
<il>гигабайты

<h2> Пример неупорядоченного списка
</h2>
Единицы измерения информации

биты
байты
килобайты
мегабайты
гигабайты

</body>
</html>
Сохраните документ с именем Ex4.html в рабочей папке.

Упражнение 5. Использование внешних гиперссылок
<html>
<head>
<title>Упражнение 5</ title>
</head>
<body>
Перейти к LIMTU

Напишите мне по электронной почте

Открытие диска или папки < a href ="C:\">C:
</body>
</html>
Сохраните документ с именем Ex5.html в рабочей папке.

Упражнение 6. Использование внутренних гиперссылок

```
<html>
<head>
<title>Упражнение 6</ title>
</head>
<body>
Перейти к <a href="Ex5.html">упражнению 5</a>
</body>
</html>
```

Сохраните документ с именем Ex6.html в той же папке, что и документ Ex5.html.

Упражнение 7. Использование ссылок в одном документе

```
<html>
<head>
<title>Упражнение 2</ title>
</head>
<body>
<a name="Chapt1">Глава1 </a>, самая главная,
к ней мы еще вернемся <br>
  <br>   <br>   <br>   <br>   <br>  
<br>   <br>   <br>   <br>   <br>   <br>
 
<br>   <br>   <br>   <br>   <br>
глава2 <br> глава3 <br>
можно вернуться из той же страницы
<a href="#Chapt1">возврат</a>
</body>
</html>
```

Сохраните документ с именем Ex7.html в рабочей папке.

Упражнение 8. Создание файла для возврата к определенному месту документа (якорю)

```
<html>
<head>
<title>Упражнение 8</ title>
</head>
<body>
<a href="Ex7.html#Chapt1">перейдем к 1 главе</a>
</body>
</html>
```

Сохраните документ с именем Ex8.html в той же папке, что и документ Ex7.html.

Задание 1. Создание и наполнение страниц будущего сайта

В данном задании Вы начнете создание страниц будущего сайта.

1. Выберите тему для будущего сайта, предлагается сделать сайт “о себе” или своих увлечениях.
2. Сделайте макет будущего сайта: продумайте, какие страницы войдут в сайт, как они будут взаимосвязаны, какую информацию будут содержать, и как эта информация будет располагаться на каждой странице.
3. Подготовьте наполнение страниц - текстовые фрагменты, графические файлы, видео- и аудио – файлы, приведите информацию к необходимому формату, задайте нужные размеры изображениям. Создайте папку для сайта и скопируйте туда всю подготовленную информацию.
4. В папке сайта создайте HTML-документы, соответсвующие главной и другим страницам сайта.
5. Разместите в документах текстовую информацию, отформатируйте ее нужным образом.
6. Добавьте на страницы сайта навигационное меню и организуйте переходы по гиперссылкам между страницами сайта.

2.2. Практическая работа №2

Упражнение 9. Задание фонового изображения

1. Выберите подходящее для фона изображение и скопируйте его в рабочую папку. Переименуйте его в img1.gif.

```
<html>
<head>
<title>Упражнение 9</ title>
</head>
<body background=”img1.gif”>
55
--- Это моя страница, но уже с интересным фоном ---
</body>
</html>
```

2. Сохраните документ с именем Ex9.html в рабочей папке.

Упражнение 10. Вставка изображений

1. Выберите подходящее изображение и скопируйте его в рабочую папку. Переименуйте его в img2.gif.

```
<html>
<head>
<title>Упражнение 10</ title>
</head>
```

<body>
Вставка изображения
<img src="img2.gif">
< img src="img2.gif" width=82 height=68>
< img src="img2.gif" width=200 height=68>
< img src="img2.gif" width=20>
</body>
</html>
2. Сохраните документ с именем Ex10.html в рабочей папке.

Упражнение 11. Создания таблицы

```
<html>
<head>
<title>Упражнение 11</ title>
</head>
<body>
<table border cellspacing=0 cellpadding=10>
<caption align=top> Оргтехника
</caption>
<tr>
<th>Наименование</th>
<th> Цена </th>
<th> Количество </th>
</tr>
<tr>
<th> Принтер </th>
<td align =right> 350 </td>
<td align =right> 2 </td>
</tr>
<tr>
<th> Ксерокс</th>
<td align =right> 1250 </td>
<td align =right> 1 </td>
</tr>
<tr align=center>
<th> Факс</th>
<td align =right> 250 </td>
<td align =right> 2 </td>
</tr>
</table>
</body>
</html>
```

Сохраните документ с именем Ex11.html в рабочей папке.

Упражнение 12. Создание вложенных таблиц

```
<html>
<head>
<title>Упражнение 12</ title>
</head>
<body>
<table border =3 width=200 height=100>
<tr>
<td>
<table border =3>
<tr>
<td>Ed</ td>
</tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```

Сохраните документ с именем Ex12.html в рабочей папке.

Задание 2: Верстка одной из страниц сайта и графическое оформление страниц

В ходе выполнения данного практического задания необходимо сверстать одну из страниц сайта и добавить графическую информацию на страницы.

1. Оформите одну из страниц сайта (например, главную страницу), с помощью табличной верстки, то есть используя для взаимного расположения элементов на странице таблицы с невидимыми границами. Для этого продумайте, как будут располагаться элементы на странице относительно друг друга, создайте таблицу и расположите элементы в ее ячейках.
2. Задайте фоновые изображения на страницах, если это необходимо.
3. Поместите графическую информацию на страницы сайта, а также добавьте аудио- и видео-файлы.

2.3. Практическая работа №3

Упражнение 13. Создание и использование каскадных таблиц стилей

1. Внутренние таблицы стилей задают стиль только одному элементу документа при помощи атрибута style в HTML теге.

```
<html>
```

<head>
<title>Упражнение 13a</title>
</head>
<body>

Пример физического форматирования.

<p style=”color:green; font-size:12pt; font-family:Arial”>
Пример встроенной таблицы стилей.

</body>
</html>

2. Сохраните документ с именем Ex13a.html в рабочей папке.

3. Глобальные стили (внедренные таблицы) задают вид элементов всего документа. Для этого используется тег <style type="text/css">. Он размещается в заголовке документа.

<html>
<head>
<title>Упражнение 13b</title>
<style type="text/css">
h1{color:red; font-style:italic; font-size:32px;}
p{color:blue}
</style>
</head>
<body>
Пример внедренной таблицы стилей.
<h1> Этот заголовок написан крупным красным курсивом. </h1>
 Это предложение выделено синим цветом.
</body>
</html>

4. Сохраните документ с именем Ex13b.html в рабочей папке.

5. Использование внешних стилевых таблиц — самый экономный и обобщающий способ задания правил оформления однотипных элементов для любого количества страниц. То есть одну таблицу стилей можно использовать для форматирования многих страниц, что приводит к единообразному отображению различных документов и придает некоторую системность серверу разработчика. Осуществляется связывание отдельного файла, содержащего стилевые правила, с множеством гипертекстовых документов. Связываемый файл содержит набор правил.

body{background:grey; font-size:14pt; color:red; font-family:Arial;}
h1, p{color:green;}

6. Сохраните документ с именем example_styles.css в рабочей папке.
7. В самих же HTML документах делается ссылка на этот файл при помощи тега <link>. Выглядит это так: <link rel="stylesheet"
type="text/css"
href="адрес файла со стилями">
<html>
<head>
<title> Упражнение 13c</title>
<link rel="stylesheet" type="text/css" href="example_styles.css">
</head>
<body>
Пример внешней таблицы стилей
<h1> Этот заголовок выделен зеленым цветом. </h1>
 И это предложение тоже.
</body>
</html>
8. Сохраните документ с именем Ex13c.html в рабочей папке.

Упражнение 14. Использование классов в создании каскадных таблиц стилей
<html>
<head>
<title> Упражнениеи 14</title>
<style>
.blue {color:blue; font-style:italic;}
#boldunderline {text-decoration:underline; font-weight:bold;}
</style>
</head>
<body>
 Здравствуйте, это моя домашняя страница.
 Пока еще в стадии разработки ...

... Но скоро откроется
</body>
</html>
Сохраните документ с именем Ex14.html в рабочей папке.

Упражнение 15. Использование блочной верстки
1. Откройте новый HTML-документ.
2. Создайте в нем три блока разного цвета:

1

2

3

3. Задайте ширину для каждого блока:

1

2

3

4. Теперь предлагается каждому из блоков добавить свойство float со значением, например left, блоки должны выстроиться в один ряд:

1

2

60
3

5. Попробуйте добавить ниже еще один блок другого цвета:

1

2

3

4
4

6. Так как выше у нас блоки со свойством float, то для того, чтобы четвертый блок оказался под уже созданными тремя, надо для четвертого

блока задать свойство clear:
 1
2
3
4
4
7. Для того, чтобы убрать отступ между четвертым блоком и первыми тремя добавьте для всей страницы стилевые свойства margin и padding:
<style type="text/css">
*{ margin:0px; padding:0px; }
</style>
8. Должен получиться следующий вариант страницы:
<html>
<head>
<title>Упражнение 15</title>
<style type="text/css">
* { margin:0px; padding:0px; }
</style>
</head>
<body>
1
2
3
4
4
4
</body>
</html>
9. Сохраните документ с именем Ex15.html в рабочей папке.

Задание 3: Верстка последующих страниц сайта и стилевое оформление

На данном этапе создания сайта необходимо завершить верстку всех страниц сайта выбранными способами и оформить страницы, используя таблицы стилей.

1. Используйте блочную верстку для одной из отавшихся страниц сайта. Для этого сначала скомпануте блоки на странице, затем поместите в них содержание.
2. Для остальных страниц сайта используйте по желанию либо блочную, либо табличную верстку. При этом заготовьте отдельную страницу, которая может представлять собой либо страницу обратной связи, либо анкету, либо регистрацию, для создания формы.
3. Задайте с помощью внешних каскадных таблиц стилей одинаковой стилевое оформление страниц сайта.

4. Если необходимо, уже с помощью внедренных и встроенных таблиц стилей оформите отдельные элементы на страницах сайта.

2.4. Практическая работа №4

Упражнение 16. Текстовое поле и поле для ввода пароля

```
<html>
<head>
<title> Упражнение16 </title>
</head>
<body>
Содержание Документа
<form>
<input type=text name="login" value="user" size=40>
<input type =password name="pas" value="123" size=10
maxlenght=10>
</form>
</body>
</html>
```

Сохраните документ с именем Ex16.html в рабочей папке.

Упражнение 17. Текстовая область

```
<html>
<head>
<title>Упражнение17</title>
</head>
<body>
<form>
<textarea name="comments" rows=5 cols=30>
Комментарий 1
Комментарий 2
</textarea>
</form>
</body>
</html>
```

Сохраните документ с именем Ex17.html в рабочей папке.

Упражнение 18. Работа с флажком

```
<html>
<head>
<title>Упражнение18 </title>
</head>
<body>
```

<form action=”URL-адрес сервера” method=post
enctype=multipart/form-data>
Выберите нужные вам опции:
1 <input type=checkbox checked name="check1" value=1>

2 <input type=checkbox name="check2" value=2>
</form>
</body>
</html>
Сохраните документ с именем Ex18.html в рабочей папке.

Упражнение 19. Работа с переключателями
<html>
<head>
<title> Упражнение19</title>
</head>
<body>
<form action=”URL-адрес сервера” method=post
enctype=multipart/form-data >
Выберите цвет:
<input type=radio name="r1" value=1>красный
< input type=radio name="r1" value=2>желтый
< input type=radio name="r1" value=3 checked>синий
</form>
</body>
</html>
Сохраните документ с именем Ex19.html в рабочей папке.
Упражнение 20. Работа со списками
<html>
<head>
<title> Упражнение20</title>
</head>
<body>
<form action=”URL-адрес сервера” method=post
enctype=multipart/form-data >
Владею языком программирования
<select name=="language" multiple size=2>
<option value="A">Java
< option value="C" selected>C++
< option value="B">C#
</select>

<input type=submit name=”ok”>

<input type=reset name=”cancel”>
</form>
</body>
</html>
Сохраните документ с именем Ex20.html в рабочей папке.

Задание 4: Последние страницы сайта

Сделав данное задание, Вы доделаете страницы своего сайта, состоящего из статических HTML-документов.

1. Доделаем заготовленную раннее для формы страницу. Поместим в отведенное для формы место на странице,саму форму, состоящую из различных элементов. Не забудьте добавить элементы кнопок отправки данных и сброса результатов.
2. С помощью каскадных таблиц стилей оформите вновь добавленные элементы.
3. По желанию добавьте еще несколько страниц сайта и завершите оформление.

ГЛАВА II. СОЗДАНИЕ ИНТЕРАКТИВНЫХ СТРАНИЦ.

2.1. Обзор возможностей языка JavaScript

Взаимодействие клиента и сервера в Интернете осуществляется с помощью запросов, посылаемых клиентом серверу, и ответов сервера на запрос клиента. Его основу составляют HTTP-сообщения, подразделяемые на:

• запрос (request) клиента к серверу;
• ответ (response) сервера клиенту.

Стандартный язык разметки HTML позволяет легко создавать статичные Web–страницы. Пользователь не может менять их содержимое, не может взаимодействовать с ними. Для того чтобы сделать страницу по- настоящему интерактивной, нужен еще один язык, выполняемый в контексте браузера, - скриптовый язык. Исследования работы приложений интернета показали, что для выполнения определенных действий пользователя нет необходимости постоянно обращаться к серверу - эти действия можно реализовать на стороне клиента, если бы он позволял каким-то образом их запрограммировать. Так появился встроенный в программу просмотра Web-страниц (браузер) язык JavaScript, который расширил возможности языка разметки HTML, предоставляя разработчику возможность встраивать в документ HTML код программы, выполняющейся на клиенте. Скриптовый язык используется для создания интерактивных страниц. Обычно он не содержит всех возможностей настоящих языков программирования, таких, например, как работа с файлами или управление графикой. Созданные с помощью скриптовых языков программы не могут выполняться самостоятельно - они работают только в контексте браузера, поддерживающего выполнения скриптовых программ. Создаваемые на скриптовых языках программы, называются сценариями или скриптами, включаются в состав Web-страниц и распознаются и обрабатываются браузером отдельно от остального HTML - кода.

Язык программирования JavaScript - объектно-ориентированный язык разработки встраиваемых приложений, выполняющихся как на стороне клиента, так и на стороне сервера. Веб-обозреватель, работающий на компьютере-клиенте, обеспечивает среду, в которой JavaScript имеет доступ к объектам, которые представляют собой окна, меню, диалоги, текстовые области и т. д. Кроме того, обозреватель позволяет присоединить сценарии на языке JavaScript к таким событиям, как загрузка и выгрузка страниц и графических образов, нажатие клавиш и движение мыши, выбор текста и пересылка форм. При этом программный код сценариев только реагирует на события и поэтому не нуждается в главной

программе. Набор объектов, предоставляемых обозревателем, известен под названием Document Object Model (DOM). Основная идея JavaScript состоит в возможности изменения значений атрибутов HTML-контейнеров и свойств среды отображения в процессе просмотра HTML-страницы пользователем. При этом перезагрузки страницы не происходит.

Основные области использования JavaScript при создании интерактивных HTML- страниц:

• Динамического создания содержимого страницы во время ее загрузки или уже после того, как она полностью загружена;

• Отображения диалоговых панелей и сообщений в статусной строке браузера;

• Оперативная проверка достоверности заполняемых пользователем полей форм HTML до передачи их на сервер;

• Создание динамических HTML-страниц совместно с каскадными таблицами стилей и объектной моделью документа (DHTML);

1.1. Общий обзор языка

Основные определения

Любая программа оперирует некими данными: именем стилевого класса, размерами элемента, цветом шрифта и прочие. JavaScript может манипулировать данными, относящимися к разным типам. Тип данных описывает их возможные значения и набор применимых к ним операций. Типы данных бывают простыми и сложными. Сущность, относящаяся к простому типу данных может хранить только одно значение (это строковые, числовые и логические типы данных). Сущность сложного типа данных может хранить сразу несколько значений. Например – массивы. Другой пример сложного типа данных – объекты. Для создания механизма управления страницами на клиентской стороне было предложено использовать объектную модель документа. Суть модели в том, что каждый HTML-контейнер - это объект, который характеризуется тройкой:

- Свойства
- Методы
- События

Существование программных объектов самих по себе не имеет никакого смысла. Они дадут преимущества при программировании тогда, когда можно организовать их взаимодействие.

1. Объекты – это сложные сущности, позволяющие хранить сразу несколько значений разных типов данных, они представляют собой блоки, из которых строится JavaScript. Применяются для возвращения значений и изменения состояния форм, страниц,

браузера и определенных программистом переменных. Объекты можно сопоставить с существительными. Кошка, автомобиль, дом, компьютер, форма – все это существительные, они могут быть представлены как объекты.

2. Экземпляры объекта – сущности, хранящие реальные данные и созданные на основе этого объекта. То есть конкретный, реально существующий дом, находящийся по заданному адресу можно рассматривать, как экземпляр объекта типа дом.

3. Свойства – набор внутренних параметров объекта. Используются для того, чтобы различать экземпляры одного объекта – например, все экземпляры типа дом. Свойства сравнимы с прилагательными и ссылаются на уникальные для каждого экземпляра объекта особенности. Один и тот же объект может обладать многими свойствами: дом может быть большим и маленьким, синим и красным. Разные объекты могут обладать одинаковыми свойствами: дерево, так же, как и дом, может быть большим и маленьким, синим и красным... Большинство свойств объекта мы можем изменять, воздействуя на них через методы.

4. Методы - это действие или способ, при помощи которого мы можем изменять определенные свойства объекта, то есть управлять этими объектами, а также в некоторых случаях менять их содержимое.

5. События – это очень важное в программировании на JavaScript понятие. События главным образом порождаются пользователем, являются следствиями его действий. Если пользователь нажимает кнопку мыши, то происходит событие, которое называется Click. Если экранный указатель мыши движется по ссылке HTML- документа, происходит событие MouseOver. Существует несколько различных событий.

6. Оператор - это команда, инструкция для компьютера. Встретив в программе тот или иной оператор, машина четко его выполняет.

7. Функция - это определенная последовательность операторов, то есть набор команд, последовательное выполнение которых приводит к какому-то результату. Например, выполнение кем-то заданной Вами функции (процедуры) "возьми стакан, открой кран, набери в него воды и принеси мне" приведет к результату: Вы получите стакан воды из-под крана.

8. Переменная - в языках программирования переменные используются для хранения данных определенного типа, например параметров свойств объекта. Каждая переменная имеет свое имя (идентификатор) и хранит только одно значение, которое может меняться в ходе выполнения программы. Данные могут быть разных

типов: целое число, десятичная дробь, логическая константа, текстовая строка.

Понятие объектной модели применительно к JavaScript

При загрузке HTML-страницы в браузер интерпретатор языка создает объекты со свойствами, определенными значениями тэгов страницы. Для правильного использования объектных моделей следует четко понимать, как браузер компонует страницы и, тем самым, создает иерархию объектов. При загрузке страницы просматриваются сверху вниз, тем самым последовательно происходит компоновка страницы и ее отображение в окне браузера. А это означает, что и объектная модель страницы также формируется последовательно, по мере ее обработки. поэтому невозможно обратится из сценария, расположенного ранее какой- либо формы на странице, к элементам этой формы. Всегда следует помнить том, что браузер последовательно сверху вниз интерпретирует содержимое HTML-страницы.

Еще один аспект работы с объектами языков сценариев заключается в том, что нельзя изменить свойства объектов. Браузер обрабатывает страницу только один раз, компонуя и отображая ее. Поэтому попытка в сценарии изменить свойство отображенного элемента страницы, обречена на провал. Только повторная загрузка страницы приведет к желаемому результату.

Размещение операторов языка JavaScript на странице

Встроить сценарий JavaScript в HTML-страницу можно несколькими способами.

1. Задание операторов языка внутри тэга <script> языка HTML.

Для внедрения в HTML-страницу сценария JavaScript в спецификацию языка HTML был введен тэг-контейнер <script>...</script>, внутри которого могут располагаться операторы языка JavaScript. Обычно браузеры, не поддерживающие какие-нибудь тэги HTML, просто их игнорируют, анализируя, однако, содержимое пропускаемых тэгов с точки зрения синтаксиса языка HTML, что может приводить к ошибкам при отображении страницы. Во избежание подобной ситуации следует помещать операторы языка JavaScript в контейнер комментария <!-- ... //-- >, как показано ниже<script (language="javascript")>

```
<!--
операторы javascript
//-->
</script>
```

Параметр language задает используемый язык сценариев. В случае языка JavaScript его значение задавать не обязательно, так как этот язык используется браузерами по умолчанию. Примечание:

символы // перед закрывающим тэгом комментария --> являются оператором комментария JavaScript. Он необходим для правильной работы интерпретатора. Документ может содержать несколько тэгов <script>, расположенных в любом месте документа. Все они последовательно обрабатываются интерпретатором JavaScript по мере отображения частей документа в окне браузера. В связи с этим ссылка на переменную, определенную в сценарии, размещенном в конце документа, может привести к генерации ошибки интерпретатора при обращении к такой переменной из сценария в начале документа.

2. Задание файла с кодом JavaScript.

Тэг <script> имеет параметр src, позволяющий связать встраиваемый сценарий с внешним файлом, содержащим программный код на языке JavaScript. В качестве значения параметра задается полный или относительный URL-адрес ресурса. Задание закрывающего тэга </script> обязательно, независимо от того, заданы или нет операторы внутри тэга. Следующий фрагмент кода связывает документ HTML с файлом- источником, содержащим некоторый набор функций:

```
<script language="JavaScript" src="http://url/file.js">
операторы javascript
</script>
```

Примечание: связываемый внешний файл не должен содержать тэгов HTML и должен иметь расширение .js.

3. Использование выражений JavaScript в качестве значений параметров тэгов HTML. Переменные и выражения JavaScript можно использовать в качестве значений ьпараметров тэгов HTML. Например:

```
<a href="javascript: window.open('name.htm', '_self')">
<img src="images/ch1.png" border="0">
</a>
```

4. Определение обработчика событий в тэге HTML.

1.2. Язык ядра JavaScript

Синтаксис языка

Язык JavaScript чувствителен к регистру. Приложение JavaScript представляет собой набор операторов языка (команд), последовательно обрабатываемых встроенным в браузер интерпретатором. Каждый оператор можно располагать в отдельной строке. В этом случае разделитель ‘;’, отделяющий один оператор от другого, не обязателен. Его используют только в случае задания нескольких операторов на одной строке. Любой оператор можно расположить в нескольких строках без всякого символа продолжения. Например, следующие два вызова функции alert эквивалентны:

…
alert("Подсказка");
alert(
"Подсказка"
);
…
Нельзя перемещать на другую строку единый строковый литерал – он должен располагаться полностью на одной строке текста программы или разбит на два строковых литерала, соединенных операцией конкатенации
‘+’:
…
alert("Подсказка"); // правильно
alert("Под
сказка"); // не правильно
alert("Под" +
"сказка"); // правильно (но браузер выведет текст одной строкой!)
…
Пробельные символы в тексте программы являются незначащими, если только они не используются в строковых литералах. В JavaScript строковые литералы можно задавать двумя равноправными способами - последовательность символов, заключенная в двойные или одинарные кавычки:
"Анна"
'Анна'
В строковых литералах можно использовать ESC-последовательности, которые начинаются с символа обратной наклонной черты, за которой следует обычный символ. Некоторые подобные комбинации трактуются как один специальный символ.

Таблица 1.

Esc-последовательности	Символ
\b	Возврат на один символ
\f	Переход на новую страницу
\n	Переход на новую строку
\r	Возврат каретки
\t	Горизонтальная табуляция Ctrl-I

\'	Апостроф
\"	Двойные кавычки
\\	Обратная наклонная черта

ESC-последовательности форматирования используются при отображении информации в диалоговых окнах, отображаемых функциями alert(), prompt() и confirm(), а также, если методом document.write() записывается содержимое элемента pre.

Комментарии в программе JavaScript двух видов - однострочные и многострочные:

// комментарий, расположенный на одной строке.

/*
комментарий, расположенный
на нескольких строках.
*/

Ссылка на объект осуществляется по имени, заданному параметром name тэга HTML, с использованием точечной нотации. Например, пусть в документе задана форма с двумя полями ввода:

<form name="form1">
Фамилия: <input type = "text" name = "student" size = 20>
Курс: <input type = "text" name = "course" size = 2>
</form>

Для получения фамилии студента, введенного в первом поле ввода, в программе JavaScript следует использовать ссылку document.form.student.value, а чтобы определить курс, на котором обучается студент, необходимо использовать ссылку document.form.course.value.

Переменные и литералы в JavaScript

В JavaScript все переменные вводятся с помощью одного ключевого слова var. Синтаксическая конструкция для ввода в программе новой переменной с именем name1 выглядит следующим образом: var name1;

Объявленная таким образом переменная name1 имеет значение 'undefined' до тех пор, пока ей не будет присвоено какое-либо другое значение, которое можно присвоить и при ее объявлении:

var name1 = 5;
var name1 = "новая строковая переменная";

JavaScript поддерживает четыре простых типа данных:

• Целый
• Вещественный

• Строковый
• Логический (булевый)

Для присваивания переменным значений основных типов применяются литералы – буквальные значения данных соответствующих типов.

Выражения JavaScript

Выражение – комбинация переменных, литералов и операторов, в результате вычисления которой получается одно единственное значение. Переменные в выражениях должны быть инициализированы.

1. Присваивание

Оператор присваивания (=) рассматривается как выражение присваивания, которое вычисляется равным выражению правой части, и в то же время он присваивает вычисленное значение выражения переменной заданной в левой части: var name2=10;

2. Арифметическое выражение.

Вычисляемым значением арифметического выражения является число. Создаются с помощью арифметических операторов.

Таблица 2.

Оператор	Действие
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Остаток от деления целых чисел
++	Увеличение значения на единицу
--	Уменьшение значения на единицу

3. Логическое выражение

Вычисляемым значением логического выражения может быть true или false. Для создания используются операторы сравнения или логические
операторы, применяемые к переменным любого типа.

Таблица 3.

Операторы
сравнения Значение Логические

Операторы
Значение
= = Равно && логическое И
! = Не равно | | логическое ИЛИ
>= Больше или равно ! логическое НЕ
<= Меньше или равно
> Строго больше
< Строго меньше

4. Строковые выражения

Вычисляемым значением строкового выражения является число. В JavaScript существует только один строковый оператор – оператор конкатенации (сложения) строк:
string1 = “Моя ” + “строка”

1.3. Управляющие конструкции языка JavaScript

Операторы JavaScript

Операторы служат для управления потоком команд в JavaScript. Блоки операторов должны быть заключены в фигурные скобки.
1. Операторы выбора
• условный оператор if
Эта управляющая структура используется, когда необходимо выполнить некий программный код в зависимости от определенных условий. Также предусмотрена конструкция if-else (если-тогда-иначе).

```
if (условие_1)
{
оператор_1; // эти операторы выполняются, если условие_1
верно
оператор_2;
}
else
{
оператор_3; // эти операторы выполняются, если условие_1 ложно °74
в73 оператор_4;
}
```

Условие для проверки (вопрос компьютеру) записывается сразу после слова if в круглых скобках. После этого в фигурных скобках пишется то, что будет предприниматься в случае выполнения условия. Далее else и снова в фигурных скобках то, что выполнится в случае, если условие не сработает. Количество различных действий между фигурными скобками неограниченно, фактически можно выполнить две различные программы. При сравнении можно использовать логические выражения. Например:

```
<script language="JavaScript">
```

```
var x = 5;
var y = 10;
if (x>y) {
alert('x - максимальное число')
}
else
{
alert('y - максимальное число')
}
</script>
```

• оператор выбора switch

Это фактически несколько условных операторов, объединенных в одном. В данном операторе вычисляется одно выражение и сравнивается со значениями, заданными в блоках case. В случае совпадения выполняются операторы соответсвующего блока case.

```
switch (выражение) {
case значение1:
оператор_1;
break;
case значение2:
оператор_2;
break;
…..
default:
оператор;
}
```

Если значение выражения не равняется ни одному из значений, заданных в блоках case, то вычисляется группа операторов блока default, если этот блок задан, иначе происходит выход из оператора switch. Необязательный оператор break, задаваемый в блоках case, выполняет безусловный выход их оператора switch.

2. Операторы цикла

Оператор цикла повторно выполняет последовательность операторов JavaScript, определенных в его теле, пока не выполниться некоторое заданное условие.

• цикл for (цикл со счетчиком)

```
for (i=1; i<10; i++){
<тело цикла>
}
```

Первый параметр (i=1) определяет счетчик и указывает его начальное значение. Этот параметр называется начальным выражением, поскольку в нем задается начальное значение счетчика (начальное

значение в данном случае равно единице). Это выражение инициализации выполняется самым
первым и всего один раз. Второй параметр (i<10) - это условие, которое должно быть истинным, чтобы цикл выполнялся, как только условие цикла становится ложным, работа цикла завершается. Он называется условием цикла. Проверка условия цикла осуществляется на каждом шаге; если условие истинно, то выполняется тело цикла (операторы в теле цикла). Цикл в данном случае выполнится только девять раз так как задано условие i<10. Третий параметр (i++) - это оператор, который выполняется при каждом последовательном прохождении цикла. Он называется выражением инкремента, поскольку в нем задается приращение счетчика (приращение счетчика в данном случае равно единице). Пример автоматической прорисовки нескольких линий с помощью цикла for.

```
<script language="JavaScript" type="text/JavaScript">
for (var i=1; i<10; i++){
document.write("<hr align='center' width='100'>");
}
</script>
```

• цикл while (цикл с предусловием)

```
while (условие)
{
<тело цикла>
}
```

Пока значение условия - true (истинно), выполняется тело цикла. Тело цикла может быть представлено простым или составным оператором. Оператор while содержит в скобках все необходимые параметры условия цикла (логическое выражение). После определения всех параметров цикла вводится открывающая фигурная скобка, символизирующая начало тела цикла. Закрывающая фигурная скобка вводится в конце тела цикла. Все операторы, введенные в скобках, выполняются при каждом прохождении цикла.

```
<script language="JavaScript">
var i=1;
while(i<=10){
document.write('число='+i+'<br>');
i=i+2;
}
</script>
```

• прерывание и перезапуск цикла

Оператор прерывания break позволяет прервать выполнение цикла и перейти к слудующему за ним выражению:

```
a = 10;
```

```
i = 1;
while (a<100){
17
a = a * i;
if (i>4) break;
++i;
}
```
Если значение i превысит 4, то прерывается выполнение цикла. Оператор перезапуска continue позволяет перезапустить цикл, т.е. оставить невыполненными все последующие выражения, входящие в тело цикла, и запустить выполнение цикла с самого начала.
```
a = 10;
i = 1;
while (a<100){
++i;
if (i>2 && i<11) continue;
a = a * i;
}
```
Создание и вызов функций в JavaScript

В JavaScript функцией называется именованная часть программного кода, которая выполняется только при обращении к ней посредством указания ее имени. Функции создаются с помощью ключевого слова function. Обычно функции располагают в секции <head>. Такое расположение функций в HTML-документе гарантирует их полную загрузку до того момента, когда их можно будет вызвать из секции <body>. После названия функции (func_name) ставятся двойные круглые скобки, программный код при этом заключается в фигурные скобки:
```
<script language="JavaScript">
function func_name()
{
программный код функции (тело функции)
}
</script>
```
Для того, чтобы вызвать функцию в нужном месте, необходимо просто указать ее имя в тексте:
```
<script language="JavaScript">
func_name();
</script>
```
Второй вариант вызова функции непосредственно в HTML теге:
```
<a href="javascript:func_name()">Текст ссылки</a>
```
Ниже приведен код страницы HTML, после загрузки которой каждые

три секунды будет появляться сообщение, генерируемое вызовом функции myMessage():

```
<script>
function myMessage()
{
alert("My Message")
}
</script>
<body onload='setTimeout ("myMessage()",3000)'>
<p>Каждые три секунды будет появляться сообщение</p>
</body>
```

Метод setTimeout() запускает выполнение кода JavaScript, задаваемого первым строковым параметром, через определенный промежуток времени после выполнения метода. Интервал задается в миллисекундах (1000 соответствует 1 секунде).

1.4. Стандартные объекты и функции ядра JavaScript

Объект Array

Массив - упорядоченный набор однородных данных, к элементам которого можно обращаться по имени и индексу. Язык JavaScript не имеет строенного типа данных для создания массивов, поэтому для решения используется объект Array и его методы. Для создания объекта Array вызывается оператор new и конструктор массива - системная функция (ее имя совпадает с именем объекта), инициализирующая элементы массива:

```
m=new Array();
```

Заполнение массива происходит позже. Например:

```
<script language="JavaScript">
//создание нового массива
m=new Array();
//заполнение массива
m[0]=1;
m[1]=2;
m[2]=4;
m[3]=56;
</script>
```

В приведенном выше примере с помощью команды new создается массив m, а затем происходит его заполнение - каждому элементу присваивается определенное значение.

```
m=new Array(1,2,4,56);
```

Вызывается команда new и сразу задаются значения всех элементов массива.

```
<script language="JavaScript">
```

//создание нового массива и его заполнение
m=new Array(1,2,4,56)
</script>
Объявление строковых массивов проводится тем же способом, что и объявление числовых массивов.

Таблица 4.

Методы объектаArray	Действие
join()	Объединяет все элементы массива в одну строку с указанием разделителя.
reverse()	Изменяет порядок элементов в массиве - первый элемент становится последним, последний - первым
sort()	Выполняет сортировку элементов массива
split()	Разделяет строку на составные части
concat()	Объединяет два массива в один
slice()	Выделяет часть массивы
toString()	Возвращает строку - результат конкатенации всех элементов массива. Элементы массива в строке разделены запятой.
Свойство length Возвращает	длину массива (число элементов в нем).

Пусть определены два массива:
array1 = new Array("Первый","Второй","Третий");
array2 = new Array("Один","Два","Три");
Тогда метод join() первого массива array1.join() возвратит строку:
"Первый,Второй,Третий"
Метод sort() первого массива array1.sort() упорядочит элементы массива array1 (переставив их местами непосредственно в самом массиве
array1) в алфавитном порядке:
array1[0] = "Второй";
array1[1] = "Первый";
array1[2] = "Третий";

Поскольку некоторые методы массива возвращают массив, то к нему можно сразу же применить какой-либо метод, продолжив "точечную" нотацию. Например, array1.concat(array2).sort() объединит два массива в один новый и отсортирует его.

Объект Date

Используется для представления дат в программах JavaScript. Время храниться в виде числа миллисекунд, прошедших от 1 января 1970 года. Данный объект создается также, как и любой объект в JavaScript – с помощью оператора new и конструктора, в данном случае Date(): date1 = new Date(); // значением переменной date1 будет текущая дата Параметром конструктора может быть строка, в которой записана нужная дата:

date1 = new Date(“january 14, 2000, 12:00:00”);

Можно задать список параметров:

date1 = new Date(2000, 1, 14, 12, 0, 0);

Объект Math

В свойствах данного объекта хранятся основные математические константы, а его методы вычисляют основные математические функции. При обращении к данному объекту, создавать его не надо, но необходимо явно указывать его имя Math. Например:

p = Math.PI; // хранится значение числа пи.

Объект String

Можно явно создавать строковый объект, используя оператор new и конструктор:

myString = new String(“Hello!”);

Данный объект имеет единственное свойство length, хранящее длину строки, содержащейся в строковом объекте, и два типа методов: одни непосредственно влияют на содержание самой строки, вторые возвращают отформатированный HTML-вариант строки.

Стандартные функции верхнего уровня

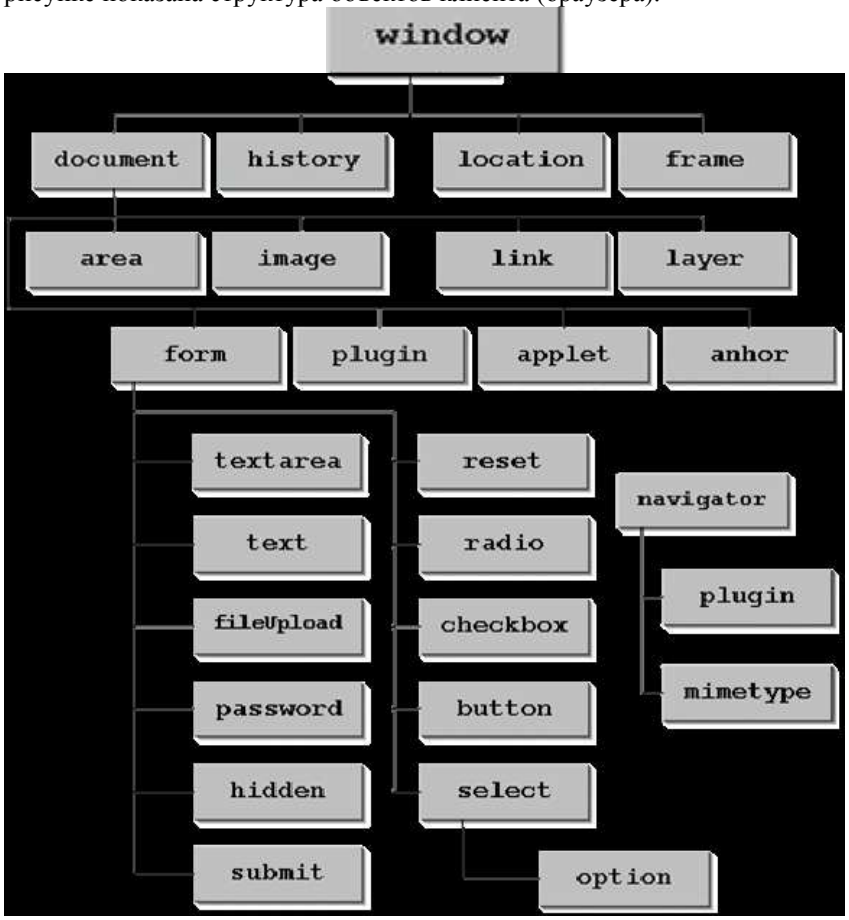
В JavaScript существуют несколько функций, для вызова которых не надо создавать никакого объекта, она находятся вне иерархии объектов. Функция parseFloat(parameter) анализирует значение переданного ей строкового параметра на соответствие представлению вещественного числа. Функция parseInt(parameter, base) пытается возвратить целое число по основанию, заданному 5 н вторым параметром. Эти функции полезны при анализе введенных пользователем данных в полях формы до их передачи на сервер. Функции Number(object) и String(object) преобразуют объект, заданный в качестве его параметра в число или строку.

1.5. Объекты клиента

При интерпритации страницы HTML браузером создаются объекты JavaScript, свойтсва которых представляют значения параметров тэгов языка HTML.

Иерархия объектов

Созданные объекты существуют в виде иерархической структуры, отражающей структуру самой HTML-страницы. На верхнем уровне расположен объект window, представляющий собой активное окно браузера. Далее вниз по иерархической лестнице следуют объекты frame, document, location и history и т.д. Значения свойств объектов отражают значения соответствующих параметров тэгов страницы или установленных системных параметров. На рисунке показана структура объектов клиента (браузера).



Особняком стоит объект navigator с двумя дочерними (подчиненными) объектами. Он относится к самому браузеру, и его свойства позволяют определить характеристики программы просмотра. Каждая страница в добавление к объекту navigator обязательно имеет еще четыре объекта:

 window — объект верхнего уровня, свойства которого применяются ко всему окну, в котором отображается документ;
 document — свойства которого определяются содержимым самого документа: связи, цвет фона, формы и т. д.;
 location — свойства которого связаны с url-адресом отображаемого документа;
 history — представляет адреса ранее загружавшихся HTML-страниц.

Кроме указанных объектов страница может иметь дополнительные объекты, зависящие от ее содержимого, которые являются дочерними объектами объекта document. Если на странице расположена форма, то все ее элементы являются дочерними объектами этой формы. Для задания очного имени объекта используется точечная нотация с полным указанием всей цепочки наследования объекта. Наиболее общий объект высшего уровня находится слева в выражении, и слева направо происходит переход к более частным объектам, являющимся при этом наследниками высших в иерархии объектов. Кроме этих классов объектов пользователь может создавать и свои собственные. Но обычно большинство программ используют эту систему классов и не создают новых.

Объект navigator

Этот объект применяется для получения информации о версиях.

Синтаксис:

navigator.name_properties

Методы и события, догадаться не определены для этого объекта. Да и свойства только для чтения, так как ресурс с информацией о версии недоступен для редактирования.

Свойства

• appCodeName - кодовое имя браузера;
• appName - название браузера;
• appVersion - информация о версии браузера;
• userAgent - кодовое имя и версия браузера;

Ниже приведен пример использования объекта navigator.

```
<html>
<head>
<title> navigator </title>
<script language="javascript">
```

```
var firstn = window.prompt("Введите ваше имя: ","ваше имя")
function welcome(){
var appname=navigator.appName
var appver=navigator.appVersion
window.alert("Привет, " + firstn + ". Вы используете " +
appname + ". Версия " + appver + ". Спасибо за визит.");
}
function writename(){
document.write(firstn + ".");
}
</script>
</head>
<body onLoad="welcome()">
Добро пожаловать,
<script language="JavaScript">
writename();
</script>
</body>
</html>
```

Объект window

Объект window создается автоматически при запуске браузера, так как для отображения документа необходимо окно. Одно из назначений объекта окна - это создание нового окна. Новое окно браузера создается с помощью метода window.open(). Метод window.open() имеет ряд дополнительных аргументов, которые позволяют задать местоположение окна, его размер и тип, а также указывают, должно ли окно иметь полосы прокрутки, полосу команд и т. п. Помимо этого можно задавать и имя окна.

В общем виде данный метод можно представить следующим образом:

```
window.open('url', 'name', 'parameters')
```

Рассмотрим синтаксис более подробно:

• Первый параметр метода window.open() - это url документа, агружаемого в окне. Если его не заполнить, то окно останется пустым.

• Второй параметр определяет название окна (name). Это имя может использоваться для обращения к созданному окну.

• Третий параметр представляет список необязательных опций, разделенных запятой. С их помощью Вы определяете вид нового окна: наличие в нем панелей инструментов, строки состояния и других элементов. Приведем таблицу с описанием параметров нового окна, задаваемого третьим параметром (parameters) метода open().

Таблица 5.

Параметр	Значение		Описание
fullscreen	yes	no	указывает, показывается ли новое окно на полный экран или как обычное окно. По умолчанию показывается обычное окно
	1	0	
channelmode	yes	no	позволяет указать, отображается ли полоса каналов
	1	0	
toolbar	yes	no	позволяет указать, отображается ли полоса кнопок
	1	0	
location	yes	no	позволяет указать, отображается ли полоса для ввода адреса
	1	0	
directories	yes	no	позволяет указать, отображается ли полоса кнопок для выбора каталогов
	1	0	
status	yes	no	позволяет указать, отображается ли полоса статуса
	1	0	
menubar	yes	no	позволяет указать, отображается ли полоса меню
	1	0	
scrollbars	yes	no	задает отображение горизонтальной и вертикальной полос прокрутки
	1	0	
resizable	yes	no	позволяет указать, может ли окно изменять свой размер
	1	0	

width	yes	no	задает ширину окна в пикселах. Минимальное значение - 100
	1	0	
height	yes	no	задает высоту окна в пикселах. Минимальное значение - 100
	1	0	
top	yes	no	задает вертикальную координату левого
	1	0	верхнего угла окна
left	yes	no	задает горизонтальную координату левого
	1	0	верхнего угла окна

Объект window использует три метода отображения сообщений:

- метод prompt() – выводит диалоговое окно с полем ввода, куда пользователь может ввести информацию
- метод alert() – выводит на экран окно - сообщение с кнопкой ОК и определенным программистом текстом
- метод confirm() – выводит диалоговое окно с кнопками ОК и Cancel. Дает возможность пользователю продолжить или отменить предложенную операцию.

Сообщение, которое вы хотите вывести на экран, набирается в кавычках внутри круглых скобок. Данный скрипт запрашивает имя посетителя и выдает приветствие с введенным именем.

```
<script language="JavaScript">
name=window.prompt ("Введите, пожалуйста, свое имя", "Ваше имя");
window.alert ("Вас зовут, " + name);
</script>
```

В этом фрагменте кода метод prompt имеет следующие параметры: текст запроса и значение, заполняющее поле ввода по умолчанию; переменная name - имя переменной, куда сохраняется введенная информация (имя может быть любым).

Объект document

Объект document имеет дело прежде всего с телом HTML-страницы. Он имеет несколько дочерних объектов (коллекций): all, images, link, anchor и form. Пользуясь объектной моделью построения документа можно, например, обратиться к любой картинке на странице через следующий синтаксис:

document.images.name.src

Для document не существует никаких событий. Некоторые свойства и методы перечислены в таблице, из методов наиболее употребимы write и writeln

.

Таблица 6.

Свойства	Назначение
bgColor	Устанавливает цвет фона текущего документа. Этот цвет может иметь шестнадцатеричное представление #rrggbb или соответствующее название. Синтаксис: document.bgColor="#e7e6d8"
fgColor	Устанавливает цвет текста документа. Аналогичен по функциям свойству bgColor
referrer	Указывает url документа, на который ссылается пользователь в настоящее время. Например, если кто-то обратился по адресу: http://www.nm.org/welcome.htm с сервера http://www.someplace.com, то свойством referrer будет: http://www.someplace.com, если это свойство было в странице вышеупомянутого расположения; в противном случае оно обращается в null
location	Соответствует адресу url текущего документа

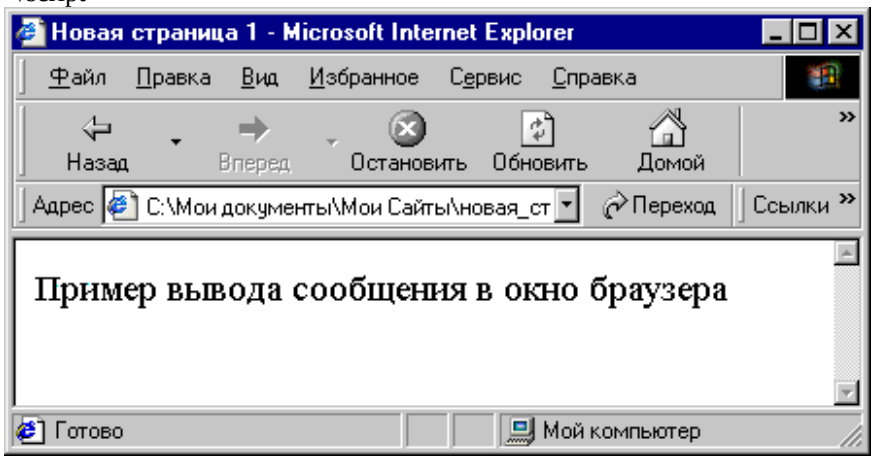
Таблица 7.

Методы	Назначение
write() writeln()	Записывает HTML-текст в текущий документ и должен вызываться, когда документ открывается для записи. Синтаксис: document.write('somestring'), где somestring может быть одной строкой, переменной или же несколькими связанными строками в формате HTML, которые выводятся на экран

lastModified()	Показывает дату последней модификации документа: date1 = document. lastModified
open()	Открывает документ для записи дополнительных строк в формате HTML: document.open()
close()	Закрывает документ, который вызывался методом document.open(): document.close().

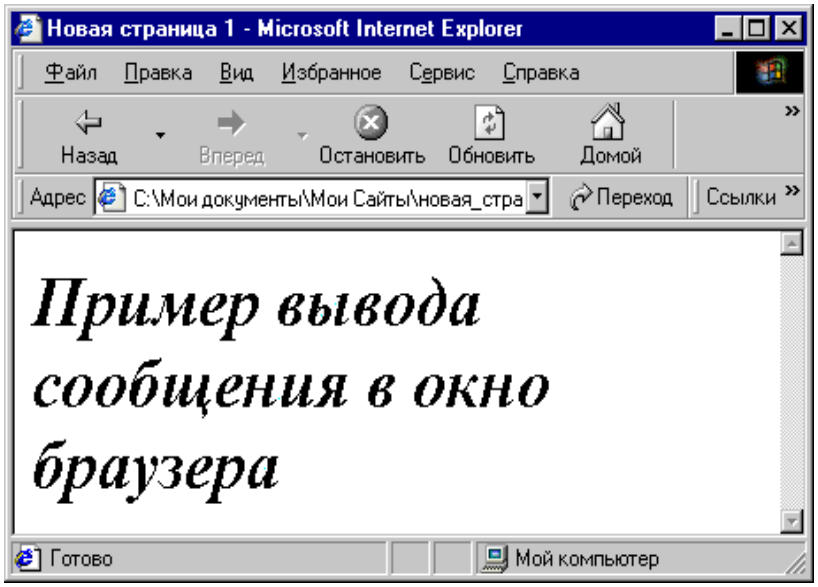
Методы write() / writeln().
Вызов метода document.write() с указанием определенных параметров приводит к отображению текста в окне браузера. В качестве параметра при вызове метода document.write() мы указываем строку, которую хотели бы увидеть на экране.

```
<script language="JavaScript">
document.write('Пример вывода сообщения в окно браузера')
</script>
```



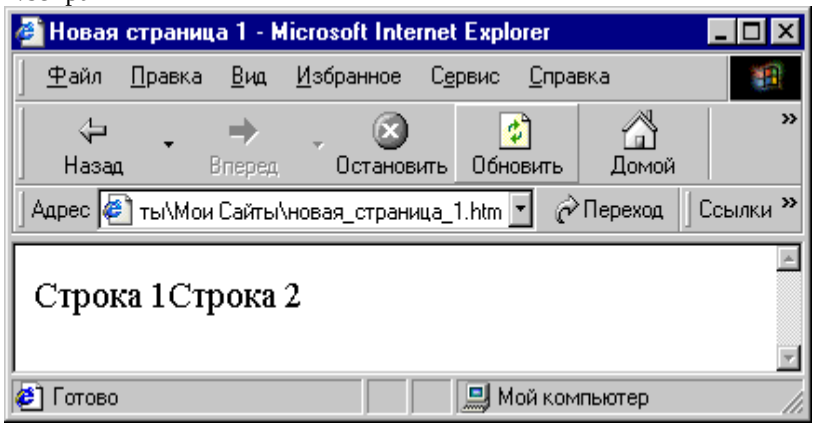
Выводимая строка может содержать и тэги языка HTML. В этом случае браузер выведет данную строку точно так же, как если бы она была размещена непосредственно в HTML документе.

```
<script language="JavaScript">
document.write('<h1><b>< i>Пример вывода сообщения в окно браузера</i></b></h1>')
</script>
```



При написании скрипта, содержащего несколько команд document.write() подряд, при выводе в браузер текст окажется на одной строке

```
<script language="JavaScript">
document.write('Строка 1') ;
document.write('Строка 2') ;
</script>
```



Для размещения каждого куска текста в новом абзаце можно использовать 2 способа:

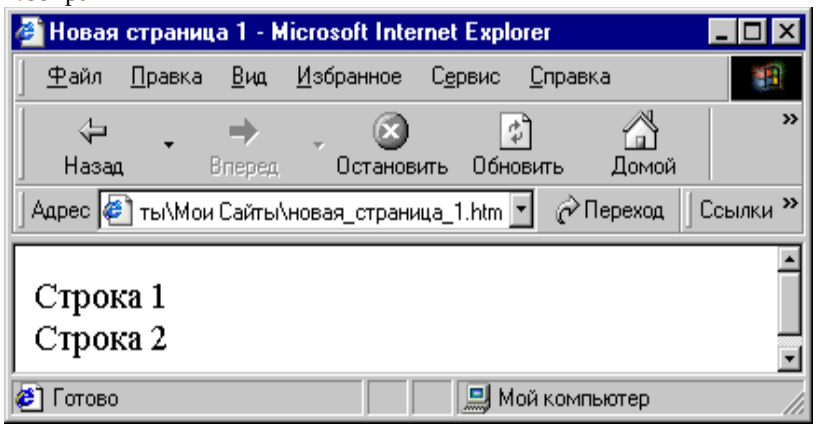
1. либо тег <p>, либо тег
, который включается в состав выводимой строки;

2. используя метод document.writeln().
Следующие примеры приведут к одинаковому результату:

```
<script language="JavaScript">
document.write('Строка 1<br>') ;
document.write('Строка 2иf22<br>') ;
</script>
```

и

```
<script language="JavaScript">
document.writeln('Строка 1') ;
document.writeln('Строка 2') ;
</script>
```



Объект location

Объект location содержит информацию о местонахождении текущего документа, т.е. его интернет-адрес. Его также можно использовать для перехода на другой документ и перезагрузки текущего документа.

Таблица 8.

Свойство	**Описание**
hash	Имя "якоря" в интернет-адресе документа, если оно есть.
host	Имя компьютера в сети интернет вместе с номером порта, если он указан.
hostname	Имя компьютера в сети Интернет.

href	Полный интернет-адрес документа
pathname	Путь и имя файла, если они есть.
port	Номер порта. Если не указан, возвращает номер 80 - стандартный порт, через который работает протокол http.
protocol	Идентификатор протокола. Если не указан, возвращается "http:".
search	Строка параметров, если она есть.

.

Таблица 9.

Метод	Описание
assign(url)	Загружает документ, адрес которого передан в качестве параметра. Поддерживаетс только IE начиная с 4.0
reload()	Перезагружает документ с Web-сервера.
replace(url)	Загружает документ, адерс которого передан в качестве параметра, и заменяет в списке истории Web-обозревателя адрес предыдущего документа адресом нового.

Пользуясь объектом location, можно загрузить другой документ на место текущего. Для этого просто необходимо присвоить значение нового интернет-адреса свойству href.

```
document.location.href = "http://www.---.ru";
```

Если вы хотите полностью заменить текущий документ, чтобы даже адрес его не появлялся в списке истории, воспользуйтесь методом replace:

```
document.location.replace("http://www.--.ru");
```

Объект form

Каждая форма в документе, определенная тегом <form>, создает объект form, порождаемый объектом document. Ссылка на этот

объект осуществляется с помощью переменной, определенной в атрибуте name тега <form>. В документе может быть несколько форм, поэтому для удобства ссылок и обработки в объект document введено свойство-массив forms, в котором содержатся ссылки на все формы документа. Ссылка на первую форму задается как document.forms[0], на вторую - document.forms[1] и т.д. Вместо индекса в массиве forms можно указывать имя формы. Например, если в документе присутствует единственная форма со значением атрибута name=form1, то любой из следующих операторов JavaScript содержит ссылку на эту форму: document.forms[0];
document.forms["form1"];
document.form1;
Последний оператор возможен в силу того, что объект document порождает объект form (как и все остальные объекты, соответствующие элементам HTML страницы) и ссылку на него можно осуществлять по обычным правилам наследования языка JavaScript.

Все элементы формы порождают соответствующие объекты, подчиненные объекту родительской формы. Таким образом, для ссылки на объект text (с параметром name = text1) формы form1 можно пользоваться любым из нижеприведенных операторов:
document.forms[0].text1;
document.forms["form1"].text1;
document.form1.text1;

Кроме имени элементы формы, имеют свойство value, значение которого определяется смыслом атрибута value элемента формы. Например, для элементов text и textarea значением этого свойства будет строка содержимого полей ввода этих элементов; для кнопки подтверждения - надпись на кнопке и т.д. Обратиться к свойству value можно по тому же принципу, например: document.form1.text1.value

1.6. Обработка событий

Использование языка JavaScript при обработке событий значительно расширило возможности языка HTML. Чаще всего программы создаются для обработки информации, вводимой пользователем в поля форм. Возможности управления элементами форм обеспечиваются главным образом за счет функций обработки событий, которые могут быть заданы для всех элементов формы. События делятся на несколько категорий:
• события, связанные с документами (события документа) - загрузка и выгрузка документов;
• события, связанные с гиперсвязью (события гиперсвязи) - помещение указателя мыши на гиперсвязь и активизация гиперсвязи;

• события, связанные с формой (события формы) –
о щелчки мыши на кнопках;
о получение и потеря фокуса ввода и изменение содержимого полей ввода, областей текста и списков;
о выделение текста в полях ввода и областях текста;

События, связанные с документами, возникают при загрузке и выгрузке документа, в то время как события гиперсвязей возникают при их активизации или при помещении на них указателя мыши. Чтобы обеспечить перехват события, необходимо написать функцию-обработчик события. В качестве обработчиков событий могут быть заданы целые функции языка JavaScript или только группы из одного или нескольких операторов. В таблице перечислены имена событий и условия их возникновения:

Таблица 10.

Имя события	Атрибут HTML	Условие возникновения события
Blur	onBlur	Потеря фокуса ввода элементом формы
Change	onChange	Изменение содержимого поля ввода или области текста, либо выбор нового элемента списка
Click	onClick	Щелчок мыши на элементе формы или гиперсвязи
Focus	onFocus	Получение фокуса ввода элементом формы
Load	onLoad	Завершение загрузки документа
Unload	onUnload	Выгрузка текущего документа и начало загрузки нового
MouseOver	onMouseOver	Помещение указателя мыши на гиперсвязь

MouseOut	onMouseOut	Помещение указателя мыши не на гиперсвязь
Select	onSelect	Выделение текста в поле ввода или области текста
Submit	onSubmit	Передача данных формы

Атрибут onClick

Атрибут onClick может использоваться в следующих тегах HTML:

• . . .
• <input type="checkbox" onClick=" function()">
• <input type="radio" onClick="function()">
• <input type="reset" onClick=" function()">
• <input type="submit" onClick=" function()">
• <input type="button" onClick=" function()">

Операторы языка JavaScript, заданные в атрибуте onClick, выполняются при щелчке мыши на таких объектах как гиперсвязь, кнопка перезагрузки формы или контрольный переключатель. Для контрольных переключателей и селекторных кнопок событие Click возникает не только при выборе элемента, но и при разблокировании. Разберем пример использования атрибута onClick для кнопок, определенных тегами
<input type="button"> в контейнере <form> . . . </form>:

```
…
<script language="JavaScript">
function but1() {
alert("Вы нажали первую кнопку");
}
function but2() {
alert("Вы нажали вторую кнопку");
}
</script>
…
<form>
<input type="button" value="Первая кнопка" onClick="but1()">
<input type="button" value="Вторая кнопка" onClick="but2()">
</form>
…
```

Работа с меню

Список в форме задается с помощью объекта select, обработка обытий выполняется с помощью следующих параметров:
onChange - вызывается при изменении выбора;
onBlur - вызывается при снятии фокуса с объекта;
onFocus - вызывается при перемещении фокуса на объект.
Рассмотрим следующий пример:

```
…
<script language="JavaScript">
function selectBlur()
{
document.myForm7.myText.value="Вы нажали поле вне списка ";
}
function selectFocus()
{
document.myForm7.myText.value="Вы нажали ту же кнопку ";
}
function selectChange()
{
document.myForm7.myText.value="Вы нажали другую кнопку ";
}
</script>
…
<form name="myForm7">
<input type="text" name="myText" size=40 value="Город"><br>
<select name="script" multiple onBlur="selectBlur()"
onFocus="selectFocus()" onChange="selectChange()">
<option value="town1" selected>Париж
<option value="town2">Лондон
<option value="town3">Рим
<option value="town4">Берлин
</select>
</form>
…
```

Управление логикой программного кода при помощи событий

В объектно-ориентированном программировании нет единой структуры управления работой программы. Есть независимые друг от друга объекты. Когда пользователь щелкает, например, по ссылке на экране, браузер передает событие Click объекту, тега <a>. Для события “щелчок мыши” в этом объекте предусмотрен стандартный обработчик — он загружает в окно новый документ. Давайте попробуем “перехватить” это событие:

документ page1

Если щелкнуть по ссылке, на экране возникнет надпись “Хода нет?”. Событие перехвачено, но, при закрытии окна alert, видим, что браузер по- прежнему грузит документ page1.htm. При помощи атрибута onClick мы установили в объекте, “отвод” на собственный обработчик. Но когда скрипт нашего обработчика выполнен, управление возвращается к стандартному обработчику, и это вызывает загрузку документа page1.htm. Отключение стандартной обработки кодируется так:

документ page1

Оператор return указывает возвращаемое функцией значение. Если ее операнд true, то документ загружается, если false, нет. Подтверждение активизации гиперсвязи. Аналогичный пример управления логикой программного кода при помощи событий рассмотрен и в следующем примере. Гиперссылка обычно всегда срабатывает по клику мыши, но иногда нужно, чтобы пользователь был уверен, что хочет перейти по ссылке в следующий документ. Для этого существует метод confirm(), который отображает на экране окно сообщения с кнопками "Ok" и "Cancel". Для перехвата события в теге <a href= ... > мы применим событие onClick. Рассмотрите пример подтверждения активизации гиперсвязи:

 Подтверждение активизации гиперсвязи

Определение событий формы

Объект form имеет два обработчика событий: onSubmit и onReset. В эти обработчики событий, задаваемые в пределах дескриптора <form>, добавляется группа операторов JavaScript или функция, управляющая формой. Если вы добавите оператор (или функцию) в обработчик onSubmit, то он (или она) вызывается до отправки данных в сценарий CGI. Для того чтобы отменить отправку данных на обработку сценарием CGI, обработчик событий onSubmit должен возвратить значение false. Если же он возвращает значение true, то данные отправляются на сервер. В некоторых случаях необходимо добавить в форму кнопку reset, запускающую обработчик событий onReset. Для формы одним из важных действий на странице является проверка правильности заполнения полей пользователем на машине клиента до пересылки их на сервер. В следующем примере разъясняется, как выполнять эту процедуру.

Рассмотрим скрипт, который будет проверять правильность заполнения формы. Необходимо проверить нет ли пустых строк и правильно ли введен e-mail:

```
<html>
<head>
<title>пример формы</title>
<script language="JavaScript">
function doSend(){
var v=document.user.e.value.indexOf("@",1)
if(document.user.f.value==""){
alert('Вы должны заполнить поле ФИО')
document.user.f.focus()
}
if(document.user.a.value==""){
alert('Вы должны заполнить поле адреса')
document.user.a.focus()
}
if(document.user.e.value==""){
alert('Вы должны заполнить поле e-mail')
document.user.e.focus()
}
if(v==-1){
alert('Адрес e-mail указан неверно')
document.user.e.select()
document.user.e.focus()
}
else
document.user.submit()
}
</script>
</head>
<body>
<p align="center"><font size=6>Данные о пользователе</font>
<form name="user">
<b>Пожалуйста, укажите данные о себе:</b>
<br>
ФИО<input type="text" name="f" size="30"><br>
Адрес<input type="text" name"a" size="35"><br>
e-mai<input type="text" name="e" size="30"><br>
<input type="button" value="Послать" onClick="doSend()">
<input type="reset" value="Отменить">
</form>
</p>
</body>
</html>
```

Вставка звука

Если вам необходимо озвучить страницу, вот простейшая инструкция:

```
<bgsound src="music/gimn.mid" loop=infinite>
```

С помощью JavaScript можно разнообразить страницы сайта.

Пример скрипта проигрывания музыки при наведении на заголовок текста:

```
…
<script>
function playHome() {
document.all.sound.src = "music/file.mid"}
</script>
…
<bgsound id=sound>
<h1 onMouseover=playHome()>Заголовок с музыкой</h1>
…
```

1.7. DHTML

DHTML (динамический HTML) – это набор средств, которые позволяют создавать более интерактивные Web-страницы без увеличения загрузки сервера. Другими словами, определенные действия посетителя ведут к изменениям внешнего вида и содержания страницы без обращения к серверу.

DHTML построен на объектной модели документа (Document Object Model, DOM), которая расширяет традиционный статический HTML- документ. DOM обеспечивает динамический доступ к содержимому документа, его структуре и стилям. В DOM каждый элемент Web- страницы является объектом, который можно изменять. DOM не определяет новых тэгов и атрибутов, а просто обеспечивает возможность программного управления всеми тэгами, атрибутами и каскадными таблицами стилей (CSS).

Каждой гиперссылке, заголовку или текстовому параграфу можно присвоить имя, атрибуты стиля или цвета текста и указать это имя в сценарии, имеющемся на данной странице. Сценарий может быть написан на любом существующем скриптовом языке, но здесь подразумевается использование языка JavaScript. Элементы страницы впоследствии могут изменяться в результате определенного события, например при наведении курсора мыши, при щелчке, нажатии клавиши на клавиатуре либо после истечения заданного промежутка времени. Изменяться может не только стиль и цвет текста, но и весь объект, текст или рисунок.

Объединение JavaScript и CSS

1. Пример изменения цвета текста.

```
<html>
<head>
```

<title>Простая страница</title>
</head>
<body>
<h1 style="color:red">Добро пожаловать на нашу страницу!</h1>
Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.
</body>
</html>

Данный пример применения CSS позволяет сделать заголовок красного цвета. Допустим, вы хотите, чтобы текст заголовка только тогда становился красным, когда пользователь наводит на него курсор. Этого можно добиться с помощью CSS и JavaScript.

Шаг 1. Удаление существующей информации о стиле Это действие может показаться вам шагом назад, но оно действительно необходимо:

<html>
<head>
<title>Простая страница</title>
</head>
<body>
<h1>Добро пожаловать на нашу страницу! </h1>
Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.
</body>
</html>

Шаг 2. Добавление идентификатора

Поскольку вам нужно как-то обращаться к элементу, с которым будут производиться манипуляции, необходимо в тэг <hl> добавить атрибут id - это краткое обозначение, позволяющее указать нужный элемент:

<html>
<head>
<title>Простая страница</title>
</head>
<body>
<hl id="head1">Добро пожаловать на нашу страницу!</h1>
Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.

3десь много интересной информации.
3десь много интересной информации. </p>
</body>
</html>

Шаг 3. Добавление обработчика событий

Следующий шаг — добавление обработчика событий. Этому действию соответствует событие onMouseover. Также следует указать имя функции, которая будет вызываться при выполнении события:

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<h1 id="head1" onMouseover ="colorchange ()">Добро пожаловать на
Нашу страницу!</h1>
<p>3десь много интересной информации.
3десь много интересной информации.
3десь много интересной информации.
3десь много интересной информации.
3десь много интересной информации. </p>
</body>
</html>
```

Шаг 4. Написание сценария JavaScript

Вам потребуется единственная строка, состоящая из следующих частей:

- имя объекта на странице, с которым должен выполняться ваш сценарий - в данном случае head1;
- применяемый аспект JavaScript - в данном случае style;
- атрибут стиля, который будет изменяться - color;
- новое значение, принимаемое атрибутом стиля - red.

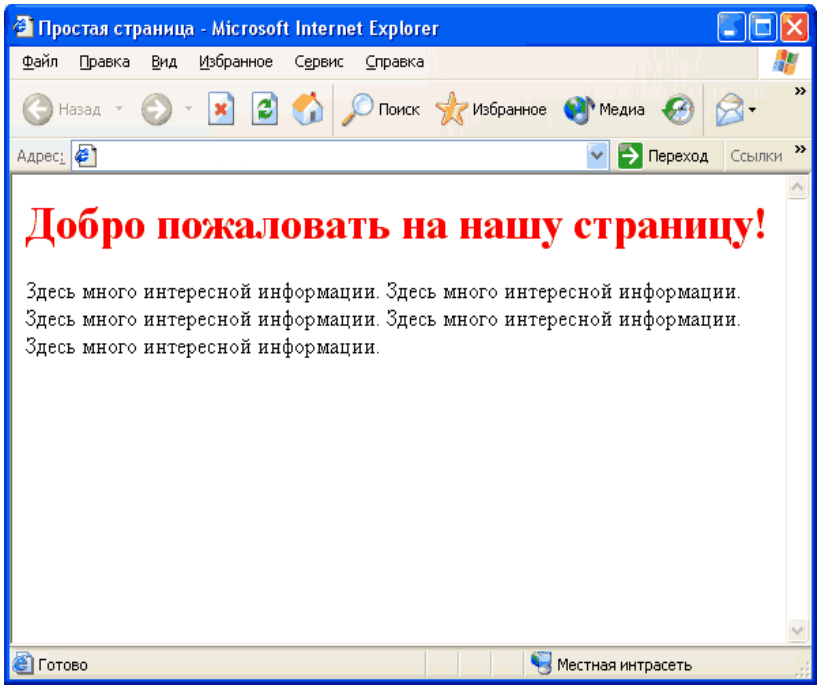
Соедините это, и получится следующая строка:

```
Head1.style.color = "red"
```

Добавьте ее в функцию и сохраните файл. В окончательном варианте страница должна выглядеть так:

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function colorchange()
{
head1.style.color = "red";
}
</script>
```

```
</head>
<body>
<h1 id="head1" onMouseover="colorchange()">Добро пожаловать на
нашу страницу!</h1>
<p>3десь много интересной информации.
3десь много интересной информации.
3десь много интересной информации.
3десь много интересной информации.
3десь много интересной информации. </p>
</body>
</html>
```



Откройте эту страницу в браузере и посмотрите, что происходит, когда вы наводите курсор на заголовок. Если все было сделано правильно, то цвет заголовка изменится. Но обратите внимание, что цвет текста не становится прежним, когда вы убираете курсор с заголовка.

2. Пример использования атрибута text-decoration.

Используя в сценариях JavaScript атрибуты, название которых пишется через дефис, убирайте дефис и пишите оба слова слитно, причем второе слово должно начинаться с заглавной буквы. Таким

образом, textdecoration в сценариях должно выглядеть как textDecoration.

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function addunderline()
{
head.style.textDecoration = "underline";
}
function removeunderline()
{
head.style.textDecoration = "none";
}
</script>
</head>
<body>
<h1 id="head" onMouseover="addunderline()"
onMouseout="removeunderline()">Добро пожаловать на нашу
страницу! </h1>
<p>Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.</p>
</body>
</html>
```

Необходимо обратить внимание на прописную букву в слове textDecoration — если все слово набрать в нижнем регистре, сценарий выполняться не будет. Теперь при наведении курсора заголовок станет подчеркнутым, а затем, если убрать курсор, вернется в прежнее состояние.

3. Пример точного позиционирования текста

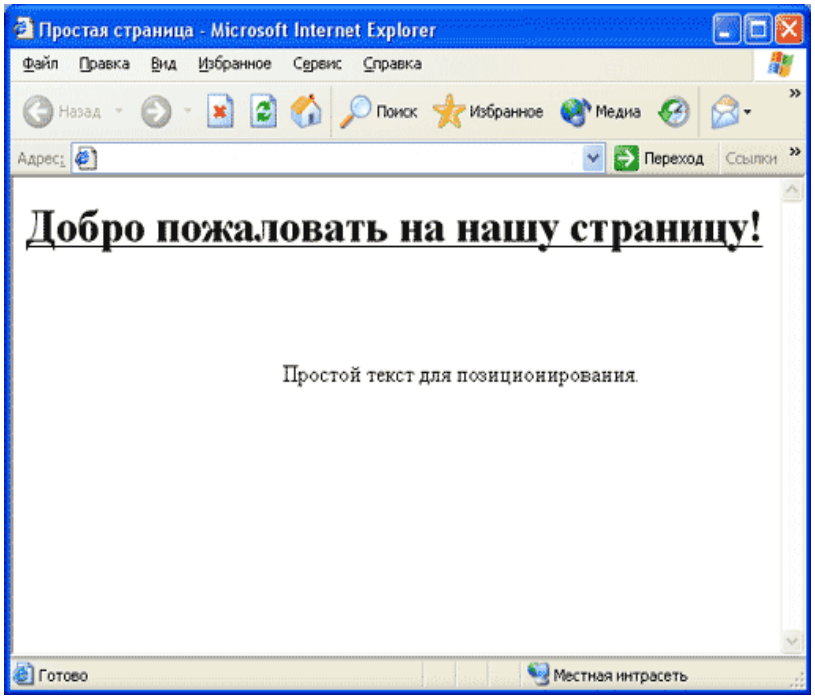
Сначала необходимо рассмотреть, каким образом осуществляется позиционирование текста. Наиболее часто применяются следующие атрибуты позиционирования:

 position - имеет два интересующих нас значения: absolute и relative (по умолчанию значение static). Для значения absolute в качестве точки отсчета используется верхний левый угол окна браузера, и все параметры местоположения отмеряются от него. В свою очередь, для relative точкой отсчета является то место, в котором разместился бы элемент страницы, если бы не было представлено никакой информации о местоположении;

 top - используется для указания вертикального смещения элемента от точки отсчета. Величина смещения может выражаться в различных единицах (пикселы, дюймы, сантиметры, миллиметры и т.п.). В наших примерах используются пикселы. Положительное значение top соответствует смещению элемента страницы вниз, в то время как отрицательное - по направлению к верхней границе окна браузера;
 left - подобен атрибуту top, но применяется для указания горизонтального направления. Положительное значение соответствует сдвигу элемента вправо, отрицательное - влево.

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<h1 style=" text-decoration: underline">Добро пожаловать на нашу страницу!</h1>
<p style="position:absolute; top:125px; left:200px">Простой текст для позиционирования.</p>
</body>
</html>
```

С помощью CSS текст расположен со значением absolute, то есть его положение отсчитывается от верхнего левого угла окна браузера. Значение атрибута top равно 125px, таким образом, текст будет расположен на 125 пикселов ниже верхнего края страницы. Значение атрибута left равно 200px, то есть текст будет сдвинут на 200 пикселов от левого края окна браузера.



1.8. Создание анимационных объектов

Анимация - это процесс «оживления» объекта. Анимация включает в себя две составляющие: расстояние между соседними кадрами, называемое скачком (jump), и временной промежуток между двумя последовательными скачками, называемый интервалом (interval). При больших скачках и длительных интервалах анимация выглядит медленной и грубой. Движение объектов кажется неестественным и воспринимается как мелькание. При малых скачках и кратких интервалах анимация выглядит более плавной, хотя, если чересчур увлечься, движение покажется нарочитым.

1. Пример перемещения текста слева направо

Сначала следует ввести текст в тэге , ограничивающем текст, добавить идентификатор id.

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<div id="anim">
```

```
Текст, шагом марш!
</div>
</body>
</html>
```

Затем воспользуемся CSS, чтобы поместить текст в начальное положение:

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

Далее начинаем работать над сценарием JavaScript. Поскольку не нужно, чтобы текст вечно двигался вправо, надо предусмотреть возможность контролирования этого процесса. Чтобы запустить сценарий на выполнение только при условии, если текст находится, например, менее чем в 500 пикселах от левой границы экрана, удобнее всеговоспользоваться оператором if. Для этого понадобится атрибут CSS
pixelLeft.

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelLeft < 500)
{
}
}
</script>
</head>
<body>
<div id="anil" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

Теперь рассмотрим операторы, управляющие анимацией. Прежде всего нужно задать скачок. Каждый раз текст будет перемещаться вправо на 50 пикселей. Атрибут pixelLeft используется не только для определения положения текста, но и для изменения положения на 50 пикселей:

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelLeft < 500)
{
anim.style.pixelLeft +=50;
}
}
</script>
</head>
<body>
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

Далее речь пойдет об интервале. Он задается с помощью метода setTimeout, позволяющего вновь запустить функцию после истечения определенного промежутка времени. Давайте установим интервал до повторного запуска функции moveTxt(), равным 5000 мс:

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
<!-- Маскируемся!
function moveTxt()
{
if (anim.style.pixelLeft < 500)
{
anim.style.pixelLeft +=50;
setTimeout("moveTxt()", 5000);
}
}
</script>
</head>
```

```
<body>
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

Процесс будет повторяться до тех пор, пока условие оператора if не станет ложным. Последнее, что нужно сделать, - запустить сценарий на выполнение. Для этого следует воспользоваться событием onLoad:

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function moveTxt()
{
if (anil.style.pixelLeft < 500)
{
anil.style.pixelLeft +=2;
setTimeout("moveTxt()", 50);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anil" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

2. Пример движения текста по диагонали

```
<html>
<head>
<title>Простая страница</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelTop < 500)
{
anim.style.pixelTop + = 2;
anim.style.pixelLeft +=2;
setTimeout("moveTxt()", 50);
}
```

```
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

1.9. Слои

Позиционирование слоя

Для создания слоев следует использовать тег или . Эти теги взаимозаменяемы и различаются лишь внешним видом в браузере. Если требуются отступы до и после текста, следует использовать элемент . При размещении текста внутри параграфа применяется тег . В Таблице 11 перечислены наиболее важные атрибуты.

Таблица 11.

Id	Имя слоя, используемое для указания его в <script>
Left	Позиция слоя по x координате
Top	Позиция слоя по y координате
Position	Задает относительную или абсолютную позицию относительно других объектов
z-index	Позиция слоя при наложении нескольких объектов друг на друга
Width	Ширина слоя в пикселах или %
Height	Высота слоя в пикселах или %
bgColor	Цвет фона слоя
background	Картинка фона
src	Внешний html документ, содержащийся в слое

Пример наложения текста:

```
<html>
<body>
Слой1 наверху
<div style="position:relative; font-size:50px; z-index:2; color: navy">
Слой 1
</div>
<div style="position:relative; top:-55; left:5; color:orange; font-size:80px;
z-index:1">
Слой 2
</div>
Слой 2 наверху
<div style="position:relative; font-size:50px; z-index:3; color: navy">
Слой 1
</div>
<div style="position:relative; top:-55; left:5; color:orange; font-size:80px;
z-index:4">
Слой 2
</div>
</body>
</html>
```

Тип позиционирования слоя определяется параметром position, положение элемента - двумя координатами top и left. Кроме тегов и абсолютное позиционирование поддерживают следующие элементы: <applet>, <input>, <button>, <object>, <select>, <fieldset>, <iframe>, <table>, <img>, <textarea>.

Параметр position:relative используется для смещения слоя относительно родительского элемента. Установка этого значения не изменяет размещение элемента, но если установлены значения свойств top или left, то слой смещается от своего нормального положения в документе. В то время как свойство position указывает тип системы координат, параметры top и left определяют точную позицию слоя. Значения этих параметров могут определяться в процентном отношении или пикселах, принимать положительные и отрицательные величины. Это дает возможность размещать контент выше или ниже на странице независимо от физической позиции кода HTML. То есть, в верхней части страницы можно поместить слой, который описан внизу HTML-документа.

Свойство z-index

Свойство z-index определяет порядок слоев, или их перекрытие по отношению к другим слоям. По умолчанию все слои позиционированы со значением z-index равным нулю. Другие слои могут размещаться ниже путем установки отрицательного значения z-index. Для слоев, у которых zindex не установлен, это значение

назначается неявно в соответствии с их положением в документе. Поэтому слой, который помещен в документ позже, размещается выше остальных элементов, позиционированных ранее.

Свойства visibility и display

Для отображения или скрытия слоя используется свойство visibility. Он может принимать значения visible, установленное по умолчанию, для показа слоя, и hidden, которое его прячет. Например, скрытый блок текста можно оформить следующим образом:

```
<div style="visibility: hidden">Спрятанный слой</div>
```

При этом, когда используется данное свойство для скрытия элемента, соотвествующий данному элементу блок занимает прежнее положение на
странице, но сома содержимо не отображается. Чтобы на странице не оставалось пустого блока, соотвествующегоскрываемому элементу, можно использовать свойство display со значением none. Для отображения элемента display равно block.

Динамическое управление слоями

Сценарии JavaScript позволяют динамически управлять параметрами установленных слоев. Это позволяет получить такие эффекты, как скрытие и отображение слоя, изменение порядка отображения, перемещение слоя в окне браузера. Все эти эффекты достигаются с помощью изменения соответствующих стилевых параметров установленных слоев. Для обращения к слоям из сценариев JavaScript, удобнее всего каждому слою дать собственное имя при помощи параметра id. Например:

```
<div id="div1">
...
</div>
```

Для того, чтобы скрыть отображение слоя div1, можно использовать следующую команду:

```
div1.style.visibility='hidden';
```

Для повторного отображения слоя следует выполнить следующее присвоение:

```
div1.style.visibility='visible';
```

Пример динамической смены слоев: в данном примере для отображения некоторого слоя следует нажать на соответствующую ссылку. Эту идею можно применить и для организации выпадающих меню.

```
<html>
<head>
<style type="text/css">
div {
```

```
position: absolute;
top: 20;
left: 160;
visibility: hidden;
}
</style>
<script language="JavaScript">
function show_d(d)
{
div1.style.visibility='hidden';
div2.style.visibility='hidden';
div3.style.visibility='hidden';
div4.style.visibility='hidden';
div5.style.visibility='hidden';
d.style.visibility='visible';
}
</script>
</head>
<body>
<a href="javascript:void(0)" onClick="show_d(div1);">
показать слой 1
</a><br>
<a href="javascript:void(0)" onClick="show_d(div2);">
показать слой 2
</a><br>
<a href="javascript:void(0)" onClick="show_d(div3);">
показать слой 3
</a><br>
<a href="javascript:void(0)" onClick="show_d(div4);">
Показать слой 4
</a><br>
<a href="javascript:void(0)" onClick="show_d(div5);">
показать слой 5
</a><br>
<div id="div1">
<h3>Слой номер один</h3>
Некоторый текст, на слое расположенный. Его можно скрыть и показать. Текст может содержать несколько строк.
</div>
<div id="div2">
<h3>Слой номер два</h3>
Содержит свой текст. Если показывается, то текст на других слоях не виден.
```

```
</div>
<div id="div3">
<h3>Слой номер три</h3>
```
Тоже текст. При работе со слоями надо следить, чтобы текст одного слоя не "выглядывал" из-под другого слоя при самых различных размерах окна браузера и используемых шрифтах.
```
</div>
<div id="div4">
<h3>Слой номер четыре</h3>
```
Здесь нет текста.
```
</div>
<div id="div5">
<h3>Слой номер пять</h3>
```
И тут тем более нет.
```
</div>
</body>
</html>
```

Динамическое изменение цвета фона ячеек

Использование стилей и управление ими с помощью JavaScript позволяет менять вид ячейки "на ходу", при выполнении определенных условий, таких как наведение курсора на ссылку или саму ячейку. Рассмотрим самый простой прием - цвет фона ячейки меняется, когда курсор мыши наводится на нее. Наведение мыши на область отслеживается событием onMouseover, а вывод мыши за ее пределы - событием onMouseout. Поскольку цвет фона меняется у той же самой ячейки, на которую наводим курсор мыши, то изменение стиля делается с помощью метода this.style.background.

```
…
<table width=60% border=1 cellspacing=0 cellpadding=4
bordercolor=#333333 align=center>
<tr>
<td align=center bgcolor=#cccccc
onMouseover="this.style.background='#ffcc33'"
onMouseout="this.style.background='#cccccc'"><a href="#">Пункт
1</a></td>
<td align=center bgcolor=#cccccc><a href="#">Пункт 2</a></td>
50
</tr>
</table>

…
```

2.2. Практика

Постановка задачи

Необходимо выполнить практические работы №1-№5 и итоговое задание, предложенное в конце. Для выполнения итогового задания Вам необходимо иметь созданный в курсе «HTML» Web-сайт, состоящий из нескольких (не менее трех) связанных между собой статических HTML-страниц и использующий основные возможности языка HTML. Для выполнения всех практических работ Вам необходимо иметь текстовый редактор (возможна работа в специальных редакторах Web-документов, например Adobe Dreamweaver), несколько браузеров для просмотра Ваших страниц (Internet Explorer, Mozilla Firefox, Opera и другие).

2.1. Практическая работа №1. Размещение скриптов в HTML-документе.

Задание 1.

1. Создайте простой HTML-документ.
2. Добавьте два абзаца с произвольным текстом.
3. Организуйте между двумя абзацами вывод приветсвенного сообщения в диалоговом окне, задав необходимые команды внутри тэга <script>.
4. Добавьте команду вывода аналогичного приветсвенного сообщения в окно браузера после закрытия диалогового окна.
5. Сохраните документ с именем Ex1.html в рабочей папке.

Задание 2.

1. Создайте простой HTML-документ.
2. Добавьте два абзаца с произвольным текстом.
3. Организуйте между двумя абзацами вывод приветсвенного сообщения в диалоговом окне, задав необходимые команды JavaScript во внешем файле. Для этого:

• создайте новый текстовый файл,
• поместите в него код JavaScript,
• сохраните файл с именем main.js следующим образом: укажите тип файла “Все файлы”, кодировку “UTF-8”.

4. Добавьте ссылку на внешний скриптовый файл из рабочего HTML-документа.
5. Сохраните документ с именем Ex2.html в рабочей папке.

Задание 3.

1. Создайте простой HTML-документ.
2. Сохраните документ с именем Ex3.html в рабочей папке**.**
3. Добавьте в документ код JavaScript так, чтобы в диалоговом окне появлялось поле с надписью "Введите сюда своё имя" и со значением по умолчанию в поле "Введите имя". Для этого используйте метод

promt(...) объекта window. Для хранения введенного значения заведите новую переменную.
4. Организуйте вывод введенного значения имени в окно браузера в виде: "Ваше имя <…..>".
5. Дополните код, чтобы в новом диалоговом окне появилось надпись "Начать заново? " При положительном ответе появлялось диалоговое окно: "Не надоело? ", при отказе – "Ну и правильно!". Используйте для написания методы alert(…) и confirm(…) объекта window.

2.2. Практическая работа №2. Операторы управления, функции. Объекты ядра JavaScript.

Задание 4.

1. Рассмотрите пример скрипта:

```
<html>
<head>
<title>if</title>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
var x, y;
x=parseInt(prompt("Введите значение x",")); // метод parseInt()
переводит строку в целое
y=parseInt(prompt("Введите значение y",")); // число
if(x<y)
{
alert("Максимальное число - y")
}
else {
alert("Максимальное число - x")
}
</script>
</body>
</html>
```

2. Допишите скрипт так, чтобы при введении пользователем одинаковых чисел, открывалось сообщение "Введенные числа равны!".
3. Напишите скрипт, в котором пользователя просят ввести правильный пароль. При вводе правильного пароля, в окне браузера появляется сообщение о том, что пароль верен. При вводе неправильного пароля – выпадает сообщение о неправильно введенном пароле. Для выполнения задания введите переменную password, в которую сохраните верное значение пароля.
4. Сохраните документ с именем Ex4.html в рабочей папке.

Задание 5.

1. Рассмотрите пример скрипта:

```
<html>
<head>
<title>for</title>
</head>
<body>
<h1>Пример простой</h1>
<script language="JavaScript" type="text/JavaScript">
function line() {
document.writeln("<hr align='center' width='100'>");}
for (var i=1; i<10; i++)
line();
</script></body>
</html>
```

2. Создайте вариант прорисованных линий со следующим условием:

• десять линий должны располагаться друг под другом,

• первая должна быть длинной 10 пикселей,

• каждая последующая на 10 пикселей больше.

3. Сохраните документ с именем Ex5.html в рабочей папке.

Задание 6.

1. Создайте простой HTML-документ.
2. Сохраните документ с именем Ex6.html в рабочей папке**.**
3. Добавьте в документ код JavaScript так, чтобы в окне браузера была выведена таблица степеней двойки вида:

Степень	Результат
2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32

Для этого в сценарии используйте метод write(…) объекта document для формирования содержимого страницы. На каждой итерации цикла for сформируйте очередную строку таблицы, в первую ячейку которой заносится соответствующая степень двойки, а во вторую результат ее возведения в указанную степень. Для выполнения этого действия используется встроенный объект Math и его метод

pow(…), возводящий первый параметр в степень, заданную вторым параметром. Обратите внимание, что метод write(…) может вызываться с любым количеством фактических параметров. Результатом его работы в любом случае является вывод в документ строки, полученной конкатенацией всех параметров, переданных в метод.

Задание 7.

1. Рассмотрите пример скрипта:

```
<html>
<head>
<title>array</title>
</head>
<body>
<script language="JavaScript">
year=new Array("декабрь","январь","февраль","март","апрель","май",
"июнь","июль","август","сентябрь","октябрь","ноябрь");
summer=new Array(); //летние месяцы
summer=year.slice(6,9);
document.write(summer+"<br>");
</script>
</body>
</html>
```

2. Создайте массив, содержащий названия школьных предметов. Выделите из него два массива. Пусть к первому относятся предметы из раздела точных наук, а ко второму - из раздела гуманитарных наук. Для создания и вывода в окно браузера новых массивов используйте метод slice(…) и wtite(…) объекта document. Оформите исполняющий скрипт в виде отдельной функции, описанной в разделе <head> и вызванной в разделе <body>.

3. Сохраните документ с именем Ex7.html в рабочей папке**.**

Задание 8.

1. Создайте простой HTML-документ.

2. Сохраните документ с именем Ex8.html в рабочей папке**.**

3. Добавьте скрипт, на основе которого будут выполняться следующие условия:

• если на страницу зашел пользователь через браузер Microsoft Internet Explorer, перенаправьте его автоматически на страницу Ex1.html;

• если на страницу зашел пользователь через любой другой браузер, перенаправьте его на страницу Ex3.html.

Для выполнения задания используйте свойство appName объекта navigator.

2.3. Практическая работа №3. Объекты клиентских приложений. Обработка событий.

Задание 9.

1. Рассмотрите скрипт:

```
<html>
<head>
<title>document</title>
</head>
<body>
<script language="JavaScript" type="text/JavaScript">
document.write("Спасибо, что пришли к нам на курсы!");
</script>
</body>
</html>
```

2. Допишите скрипт так, чтобы
• цвет фона документа был #E7E6D8,
• цвет шрифта – красный,
• внизу выводилась дата последней модификации документа, используйте для этого слияние методов wtite(…) и lastModified(…) объекта document.
3. Сохраните документ с именем Ex9.html в рабочей папке.

Задание 10.

1. Рассмотрите пример скрипта открытия нового окна на странице:

```
<html>
<head><title>window</title></head>
<body>
<h1>Создание нового окна</h1><hr>
<script language="JavaScript" type="text/JavaScript">
window.open("http://www.google.com","","toolbar=no,scrollbars=yes,widt
h=250, height=250, resizable=yes, top=100, left=500")
</script>
</body></html>
```

2. Измените скрипт так, чтобы выполнялись следующие условия:
• открытие нового окна происходило при нажатии на ссылку с текстом: «Щелкните на ссылке для получения справочной информации»,
• размеры окна - 500x500,
• есть возможность изменения размеров окна.
Для выполнения задания используйте написание функции.

3. Сохраните документ с именем Ex10.html в рабочей папке.

Задание 11.

1. Создайте страницу с переадресацией на другой адрес (redirect).

2. Изменитескрипт так, чтобы переадресация на другой адрес была с задержкой 5 секунд.
3. Сохраните документ с именем Ex11.html в рабочей папке.

Задание 12.

1. Создайте HTML-документ, в котором будет 2 ссылки:
• первая ссылка должна ссылаться на PDF файл; при нажатии на нее выпадает сообщение с предупреждением о том, что для загрузки документа требуется программа Acrobat, и продолжить загрузку или нет; используйте для написания метод confirm(…) для подтверждения загрузки;
• вторая ссылка должна содержать такой код, чтобы при наведении на нее мыши менялся цвет фона документа на красный.
2. Сохраните документ с именем Ex12.html в рабой папке.

Задание 13.

1. Создайте HTML-документ, содержащий любую картинку.
2. Добавьте скрипт с условиями:
• при наведении курсора мыши на картинку она увеличивается,
• при отведении курсора мыши – уменьшается до исходного размера.
Постройте скрипт через использование функций и событий MouseOver и MouseOut.
3. Сохраните документ с именем Ex13.html в рабочей папке.

Задание 14.

1. Создайте HTML-страницу содержащую следующую форму заполнения данных:
Ваше имя: *
Пароль *
Подтверждение пароля*
Электронный адрес: *
Тема сообщения:
Сообщение:
Отправить Очистить
* - необходимые для заполнения поля
2. Добавьте скрипт, проверяющий следующие данные:
• заполнено ли поле имени,
• введен ли пароль и содержит ли он больше 4-х символов.
Используйте для этого свойство length данного поля,
• совпадают ли значения, введенные в оба поля для паролей,
• заполнено ли поле электронного адреса и содержит ли оно символ @,
• заполнено ли поле сообщения и содержит ли оно больше 10 символов,
3. При несоблюдении условий, курсор должен установиться в то поле, где пользователем введено неверное значение.

4. Сохраните документ с именем Ex15.html в рабочей папке.

2.4. Практическая работа №4. Объединение JavaScript и CSS.

Задание 15.

1. Рассмотрите скрипт:

```
<head>
<title>h1</title>
<script language="JavaScript">
function colorchange()
{head.style.color = "red";}</script>
</head><body>
<h1 id="head" onmouseover="colorchange()">Добро пожаловать на нашу страницу!</h1>
</body>
</html>
```

2. Допишите скрипт страницы таким образом, чтобы красный цвет исчезал после отвода курсора мыши с заголовка.

3. Сохраните документ с именем Ex15.html в рабочей папке.

Задание 16.

1. Рассмотрите скрипт:

```
<html>
<head>
<title>text decoration</title>
<script language="JavaScript">
function addunderline()
{
head.style.textDecoration = "underline";
}
function removeunderline()
{
head.style.textDecoration = "none";
}
</script>
</head>
<body>
<h1 id="head" onMouseover="addunderline()"
onMouseout="removeunderline()">
Добро пожаловать на нашу страницу!
</h1>
</body>
</html>
```

2. Допишите скрипт страницы таким образом, чтобы на одинарный щелчок мыши появлялось полоса над заголовком, а на двойной

щелчок – текст зачеркивался. Используйте события onclick, ondblclick и значения рассматриваемого свойства overline и linethrough.
3. Сохраните документ с именем Ex18.html в рабочей папке.

Задание 17.

1. Создайте HTML-документ, содержащий любое изображение.
2. Поместите изображение в тег . Задайте для него абсолютное позиционирование со смещением вниз и влево на 500 пикселей.
3. Сохраните документ с именем Ex17.html в рабочей папке.

2.4. Практическая работа №5. Слои. Движущиеся элементы.

Задание 18.

1. Рассмотрите скрипт:

```
<html>
<head>
<title>simple animation</title>
<script language="JavaScript">
function moveTxt()
{if (anil.style.pixelLeft < 500){
anil.style.pixelLeft +=50;
setTimeout("moveTxt()", 5000);}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anil" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

2. Измените скрипт страницы:
• добейтесь плавного передвижения текста;
• измените направление текста - задайте направление сверху вниз при помощи атрибута pixelTop.
3. Сохраните документ с именем Ex18.html в рабочей папке.

Задание 19.

1. Рассмотрите скрипт:

```
<head>
<title>anima1</title>
<script language="JavaScript">
function moveTxt()
{
if (anim.style.pixelTop <500)
```

```
{
anim.style.pixelTop +=2;
anim.style.pixelLeft +=2;
setTimeout("moveTxt()", 50);
}
}
</script>
</head>
<body onLoad="moveTxt()">
<div id="anim" style="position:absolute; left:10; top:10">
Текст, шагом марш!
</div>
</body>
</html>
```

2. Измените направление текста. Задайте направление с верхнего правого угла экрана (приблизительно) по диагонали к середине экрана.

3. Сохраните документ с именем Ex19.html в рабочей папке.

Задание 20.

1. Создайте HTML-страницу, на которой будет три слоя. Верхний и нижней представляют из себя статичные квадраты разного цвета с текстом, а меду ними должна проплывать любая картинка слева направо.

2. Сохраните документ с именем Ex20.html в рабочей папке.

Итоговое задание

1. Перейдите к Web-сайту, созданному в курсе “Web-программирование: HTML”.

2. Добавьте к странице, содержащей форму, скрипт, осуществляющий проверку введенных в форму данных.

3. Добавьте к остальным страницам скрипты на свое усмотрение.

Литература

1. Бен Хеник, HTML и CSS. Путь к совершенству. СПб: Питер, 2011 -336 с.
2. Дунаев В., HTML, скрипты и стили. СПб: БХВ-Петербург, 2011- 816 с.
3. Дронов В., HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. СПб: БХВ-Петербург, 2011 - 416 с.
4. Джон Поллок. JavaScript. Руководство разработчика. СПб: Питер,2011-544 с.
5. Дэвид Макфарланд. JavaScript. Подробное руководство. Эксмо, 2009-608 с.
6. Чак Муссиано, Билл Кеннеди. HTML и XHTML. Подробное руководство. Символ-Плюс, 2011 - 752 с.
7. Кисленко Н.П. HTML. Самое необходимое. СПб: БХВ-Петербург, 2012 – 352 с.
8. Комолова Н., Яковлева Е. HTML, XHTML и CSS. СПб: Питер, 2012 –304 с.
9. Пол Вилтон, Джереми МакПик. JavaScript. Руководство программиста. СПб: Питер, 2009-720 с.
10. Роберт Агулар. HTML и CSS. Основа любого сайта. Экспо, 2010 –320 с.
11. Стоян Стефанов. JavaScript. Шаблоны. СПб: Символ-плюс, 2011-272с.
12. Климов А. JavaScript на примерах. СПб: БХВ-Петербург, 2009-336 с.
13. Шафер С., HTML, XHTMLи CSS. Библия пользователя. М.: Вильямс, 2010 – 656 с.
14. Лабберс К., Олберс Н., Салим К.. HTML5 для профессионалов:мощные инструменты для разработки современных веб-приложений. М.: Вильямс, 2011 – 272 с.

Интернет-ресурсы

www.w3.org

ПРИЛОЖЕНИЕ. Цвета в HTML

Цвета выводятся с помощью смешения источников:

- RED (красного) цвета:
- GREEN (зеленого) цвета:
- BLUE (синего) цвета.

Значения цветов

Цвета в HTML определяют с помощью шестнадцатеричной записи комбинации значений красного, зеленого и синего цветов (RGB). Наименьшее значение, которое можно задать одному из источников равно 0 (hex #00), а максимальное - 255 (hex #FF).

Следующая таблица показывает результат объединения источников красного (R), зеленого (G) и синего (B) источников цвета:

Цвет	Цвет (HEX)	Цвет RGB
	#000000	rgb(0,0,0)
	#FF0000	rgb(255,0,0)
	#00FF00	rgb(0,255,0)
	#0000FF	rgb(0,0,255)
	#FFFF00	rgb(255,255,0)
	#00FFFF	rgb(0,255,255)
	#FF00FF	rgb(255,0,255)
	#C0C0C0	rgb(192,192,192)
	#FFFFFF	rgb(255,255,255)

Названия цветов

Некоторая совокупность названий цветов поддерживается большинством браузеров.

Примечание: Только 16 названий цветов поддерживается стандартом W3C для CSS (aqua (голубой), black (черный), blue (синий), fuchsia (фуксия), gray (серый), green (зеленый), lime (лайм), maroon (темно-бордовый), navy (темно-синий), olive (оливковый), purple (сиреневый), red (красный), silver (светло-серый), teal (сине-зеленый), white (белый),

и yellow (желтый)). Для всех других цветов необходимо использовать значение HEX цвета.

Цвет	Цвет (HEX)	Название цвета
	#FFF8DC	Cornsilk (кукурузные рыльца)
	#00008B	DarkBlue (темно-синий)
	#008B8B	DarkCyan (темно-голубой)
	#FFFFF0	Ivory (слоновая кость)
	#FFFFE0	LightYellow (светло-желтый)
	#FFA500	Orange (оранжевый)
	#4169E1	RoyalBlue (ярко-синий)

Названия цветов в HTML

На этой странице находится таблица названий цветов, которые поддерживаются большинством броузеров.

Примечание: Стандарт HTML консорциума W3C поддерживает только 16 имен цветов (aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow). Для всех остальных цветов необходимо использовать HEX (шестнадцатеричное) значение цвета.

Название цвета	HEX цвета	Цвет
AliceBlue (бледно-голубой)	#F0F8FF	
AntiqueWhite (античный белый)	#FAEBD7	
Aqua (голубой)	#00FFFF	
Aquamarine (аквамарин)	#7FFFD4	
Azure (лазурный)	#F0FFFF	
Beige (беж)	#F5F5DC	
Bisque (бисквитный)	#FFE4C4	
Black (черный)	#000000	
BlanchedAlmond (светло-кремовый)	#FFEBCD	
Blue (синий)	#0000FF	
BlueViolet (сине-фиолетовый)	#8A2BE2	
Brown (коричневый)	#A52A2A	

BurlyWood (старое дерево)	#DEB887	
CadetBlue (серо-голубой)	#5F9EA0	
Chartreuse (зеленовато-желтый)	#7FFF00	
Chocolate (шоколадный)	#D2691E	
Coral (коралловый)	#FF7F50	
CornflowerBlue (васильковый)	#6495ED	
Cornsilk (кукурузные рыльца)	#FFF8DC	
Crimson (малиновый)	#DC143C	
Cyan (светло-голубой)	#00FFFF	
DarkBlue (темно-синий)	#00008B	
DarkCyan (темно-голубой)	#008B8B	
DarkGoldenRod (темно-золотисто-коричневый)	#B8860B	
DarkGray (темно-серый)	#A9A9A9	
DarkGreen (темно-зеленый)	#006400	
DarkKhaki (темный хаки)	#BDB76B	
DarkMagenta (темно-пурпурный)	#8B008B	
DarkOliveGreen (темно-оливково-зеленый)	#556B2F	
Darkorange (темно-оранжевый)	#FF8C00	
DarkOrchid (темно-лиловый)	#9932CC	
DarkRed (темно-красный)	#8B0000	
DarkSalmon (темно-оранжево-розовый)	#E9967A	
DarkSeaGreen (темно-зеленое море)	#8FBC8F	
DarkSlateBlue (темный серовато-синий)	#483D8B	
DarkSlateGray (темный синевато-серый)	#2F4F4F	
DarkTurquoise (темно-бирюзовый)	#00CED1	
DarkViolet (темно-фиолетовый)	#9400D3	
DeepPink (темно-розовый)	#FF1493	
DeepSkyBlue (темно-небесно-голубой)	#00BFFF	
DimGray (тускло-серый)	#696969	
DodgerBlue (тускло-васильковый)	#1E90FF	
Feldspar (полевой шпат)	#D19275	
FireBrick (огнеупорный кирпич)	#B22222	
FloralWhite (цветочно-белый)	#FFFAF0	
ForestGreen (лесная зелень)	#228B22	
Fuchsia (фуксия)	#FF00FF	
Gainsboro (светло-серый)	#DCDCDC	
GhostWhite (туманно-белый)	#F8F8FF	
Gold (золотой)	#FFD700	

GoldenRod (золотисто-коричневый)	#DAA520	
Gray (серый)	#808080	
Green (зеленый)	#008000	
GreenYellow (зелено-желтый)	#ADFF2F	
HoneyDew (медовая роса)	#F0FFF0	
HotPink (ярко-розовый)	#FF69B4	
IndianRed (индийский-красный)	#CD5C5C	
Indigo (индиго)	#4B0082	
Ivory (слоновая кость)	#FFFFF0	
Khaki (хаки)	#F0E68C	
Lavender (бледно-лиловый)	#E6E6FA	
LavenderBlush (бледный розово-лиловый)	#FFF0F5	
LawnGreen (зеленая трава)	#7CFC00	
LemonChiffon (лимонный)	#FFFACD	
LightBlue (светло-синий)	#ADD8E6	
LightCoral (светло-коралловый)	#F08080	
LightCyan (светло-светло-голубой)	#E0FFFF	
LightGoldenRodYellow (светлый коричнево-желтый)	#FAFAD2	
LightGrey (светло-серый)	#D3D3D3	
LightGreen (светло-зеленый)	#90EE90	
LightPink (светло-розовый)	#FFB6C1	
LightSalmon (светлый оранжево-розовый)	#FFA07A	
LightSeaGreen (светло-зеленое море)	#20B2AA	
LightSkyBlue (светло-небесно-голубой)	#87CEFA	
LightSlateBlue (светлый серовато-синий)	#8470FF	
LightSlateGray (светлый синевато-серый)	#778899	
LightSteelBlue (светлый голубовато стальной)	#B0C4DE	
LightYellow (светло-желтый)	#FFFFE0	
Lime (лайм)	#00FF00	
LimeGreen (зеленый лайм)	#32CD32	
Linen (льняной)	#FAF0E6	
Magenta (пурпурный)	#FF00FF	
Maroon (каштановый)	#800000	
MediumAquaMarine (умеренно аквамариновый)	#66CDAA	
MediumBlue (умеренно синий)	#0000CD	
MediumOrchid (умеренно лиловый)	#BA55D3	

MediumPurple (умеренно пурпурный)	#9370D8	
MediumSeaGreen (умеренно-зеленое море)	#3CB371	
MediumSlateBlue (умеренно серовато-синий)	#7B68EE	
MediumSpringGreen (умеренная весенняя зелень)	#00FA9A	
MediumTurquoise (умеренно-бирюзовый)	#48D1CC	
MediumVioletRed (умеренно фиолетово красный)	#C71585	
MidnightBlue (ночной синий)	#191970	
MintCream (мятно-кремовый)	#F5FFFA	
MistyRose (туманно-розовый)	#FFE4E1	
Moccasin (песочный)	#FFE4B5	
NavajoWhite (темно-песочный)	#FFDEAD	
Navy (темно-синий)	#000080	
OldLace (старые кружева)	#FDF5E6	
Olive (оливковый)	#808000	
OliveDrab (тускло-оливковый)	#6B8E23	
Orange (оранжевый)	#FFA500	
OrangeRed (оранжево-красный)	#FF4500	
Orchid (лиловый)	#DA70D6	
PaleGoldenRod (бледно-золотисто-коричневый)	#EEE8AA	
PaleGreen (бледно-зеленый)	#98FB98	
PaleTurquoise (бледно-голубой)	#AFEEEE	
PaleVioletRed (бледно-фиолетово-красный)	#D87093	
PapayaWhip (дыни)	#FFEFD5	
PeachPuff (персиковый)	#FFDAB9	
Peru (коричневый)	#CD853F	
Pink (розовый)	#FFC0CB	
Plum (сливовый)	#DDA0DD	
PowderBlue (туманно-голубой)	#B0E0E6	
Purple (пурпурный)	#800080	
Red (красный)	#FF0000	
RosyBrown (розово-коричневый)	#BC8F8F	
RoyalBlue (ярко-синий)	#4169E1	
SaddleBrown (старая кожа)	#8B4513	
Salmon (оранжево-розовый)	#FA8072	

SandyBrown (песочно-коричневый)	#F4A460	
SeaGreen (зеленое море)	#2E8B57	
SeaShell (морская ракушка)	#FFF5EE	
Sienna (охра)	#A0522D	
Silver (свето-серый)	#C0C0C0	
SkyBlue (небесно-голубой)	#87CEEB	
SlateBlue (серовато-синий)	#6A5ACD	
SlateGray (синевато-серый)	#708090	
Snow (снежный)	#FFFAFA	
SpringGreen (весенняя зелень)	#00FF7F	
SteelBlue (сине-стальной)	#4682B4	
Tan (бронзы)	#D2B48C	
Teal (сине-зеленый)	#008080	
Thistle (чертополох)	#D8BFD8	
Tomato (томат)	#FF6347	
Turquoise (бирюзовый)	#40E0D0	
Violet (фиолетовый)	#EE82EE	
VioletRed (фиолетово-красный)	#D02090	
Wheat (пшеничный)	#F5DEB3	
White (белый)	#FFFFFF	
WhiteSmoke (белый дым)	#F5F5F5	
Yellow (желтый)	#FFFF00	
YellowGreen (желто-зеленый)	#9ACD32	

Web-программирование HTML, CSS и JavaScript
Учебно-методическое пособие

План университета 2013, поз 17
Редактор Н.В. Ефрюкова
Корректор А.А. Эльканова
Компьютерная верстка А.Н. Лепшокова

Подписано в печать __.11. 2013
Формат 60х84/16
Бумага офсетная
Объем: 8,3 физ.печ.л.,
7.7 усл.печ.л., 7.3 уч.изд.л.
Тираж 100 экз.

Издательство Карачаево-Черкесского
государственного университета им. У.Д. Алиева
369200, г. Карачаевск, ул. Ленина, 29

Отпечатано в типографии
Карачаево-Черкесского государственного университета
369202, Карачаевск, ул. Ленина,46