МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Белорусский национальный технический университет

Кафедра «Технология и методика преподавания»

ПРОГРАММНЫЕ СРЕДСТВА СОЗДАНИЯ ИНТЕРНЕТ-ПРИЛОЖЕНИЙ

Пособие
для студентов учреждений высшего образования,
обучающихся по специальности 1-08 01 01
«Профессиональное обучение (по направлениям)»,
направление специальности 1-08 01 01-07
«Профессиональное обучение (информатика)»

Рекомендовано учебно-методическим объединением по профессионально-техническому обучению

Минск
БНТУ
2021

УДК 004.415 + 004.42 (075.8)
ББК 32.973.202-018.2я7
П78

Составитель *Н. И. Астапчик*

Рецензенты:
кафедра электронных вычислительных средств УО «Белорусский государственный университет информатики и радиоэлектроники», (зав. каф., д-р техн. наук, доцент *И. С. Азаров*);
главный научный сотрудник лаборатории математического и естественно-научного образования научно-методического учреждения «Национальный институт образования» Министерства образования Республики Беларусь, д-р пед. наук, профессор *В. В. Казачёнок*

П78 **Программные** средства создания Интернет-приложений : пособие для студентов учреждений высшего образования, обучающихся по специальности 1-08 01 01 «Профессиональное обучение (по направлениям)», направление специальности 1-08 01 01-07 «Профессиональное обучение (информатика)» / сост. Н. И. Астапчик. – Минск : БНТУ, 2021. – 108 с.
ISBN 978-985-583-262-2.

Пособие содержит лабораторнае работы по HTML и CSS. Каждая лабораторная работа включает теоретический материал, примеры решения задач, а также контрольные вопросы.

Пособие рекомендовано к изданию учебно-методическим объединением по профессионально-техническому обучению.

УДК 004.415 + 004.42 (075.8)
ББК 32.973.202-018.2я7

ISBN 978-985-583-262-2

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В наше время требования к сайтам очень изменились: для успешного продвижения компании в Интернете необходимо предоставлять своим посетителям обратную связь, подписки, рассылки, поиск по сайту и многое другое.

Основой любого сайта является язык разметки HTML. По сравнению с другими языками, HTML наиболее прост и понятен в изучении. Знание HTML-разметки пригодится при настройке и продвижении страницы, особенно, когда необходимо выйти за границы базовых шаблонов.

С помощью языка HTML легко и быстро можно сделать форму для отправки какого-либо запроса. Однако при нажатии на такую кнопку в большинстве случаев ничего не произойдет, потому что не был задан сценарий действий, которые следует выполнить, чтобы получить результат. Поэтому создание интерактивных компонентов – это задача уже для веб-программирования.

Целью дисциплины «Программные средства создания Интернет-приложений» является изучение основ написания сайтов и программирования для сети Интернет.

Пособие содержит 9 лабораторных работ по HTML и CSS. В каждой лабораторной работе приводятся теоретические сведения, изучение которых помогает выполнить задания, выдаваемые преподавателем.

В ходе выполнения работ студент должен изучить теоретический материал, ответить на контрольные вопросы, выполнить практические задания, приведенные в качестве примеров, оформить отчет и защитить работу.

Пособие написано в соответствии с учебным планом и программой учебной дисциплины «Программные средства создания Интернет-приложений» для направления специальности 1-08 01 01-07 «Профессиональное обучение (информатика)».

Лабораторная работа № 1

ОСНОВЫ ЯЗЫКА HTML

Цель работы: изучить основы языка HTML, структуру HTML-документа, теги для оформления шрифта и абзацев, закрепить полученные знания на практике.

Теоретические сведения

Суть и составные части Web технологии

Базовые элементы технологии Web:

– Internet – это всемирная сеть разнородных компьютерных сетей, взаимодействующих по протоколу TCP/IP;

– Web является одним из приложений Internet (наряду с электронной почтой, новостями и прочими электронными сервисами), предназначенных для массового распространения разнообразной информации;

– носителями информации в Web служат Web-страницы, содержащие текст, графику, мультимедийные элементы и гиперссылки на другие ресурсы Web или Internet;

– для передачи гипертекста Web-страниц в Internet используется специально разработанный протокол – HTTP (Hyper Text Transfer Protocol);

– для разработки Web-страниц используется специальный язык разметки гипертекста – HTML (Hyper Text Markup Language).

Для просмотра Web-страниц используется специальная клиентская программа Web-браузер. В окне Web-браузера отображаются результаты интерпретации языка HTML с Web-страниц, полученных во время навигации по гиперссылкам.

Основы языка разметки гипертекста – HTML

Базовым элементом языка разметки гипертекста является – ТЕГ (дескриптор, маркер). Тег всегда заключен между скобками < > и имеет следующий вид:

```
<ТЕГ параметр1= "ЗНАЧЕНИЕ" ... параметрN="ЗНАЧЕНИЕ">
```

Теги бывают одиночными и контейнерными. Контейнером называется пара: открывающий <ТЕГ> и закрывающий </ТЕГ>.

```
<ТЕГ> Содержимое контейнера </ТЕГ>
```

Открывающий тег служит для указания программе-браузеру начала какого-либо объекта или задания свойств объектов, помещенных в контейнер. Закрывающий – для указания о конце объекта или применения свойств, заданных в открывающем теге. Параметры (атрибуты) тега задают значения свойств объекта или объектов помещенных в контейнер. Значения свойств, содержащие пробелы, берутся в кавычки, в остальных случаях кавычки можно опустить.

Текст HTML-документа набирается в любом текстовом редакторе, например, **Блокнот**. Имена элементов, их атрибуты и значения можно набирать большими или малыми буквами. Элементы принято набирать большими буквами.

Алгоритм набора и просмотра HTML-документа заключается в следующем. Во-первых, открыть **Блокнот** и сохранить пустой файл под заданным именем и в требуемую папку. Для этого выбрать пункт меню **Файл–Сохранить**.

Появится диалоговое окно, в котором обязательно в поле **Тип файла** выбрать **Все файлы**. В поле **Имя файла** указать вместе с расширением имя файла латинскими буквами. Расширением может быть .htm или .html, но предпочтительно последнее. После сохранения пустого файла **Блокнот** закройте.

Далее необходимо двойным щелчком мыши открыть файл в браузере.

Для изменения содержимого HTML-документа нужно выбрать в браузере пункт меню **Вид–Просмотр HTML кода**. Откроется **Блокнот** с содержимым документа (в данном случае документ пустой).

HTML-документ представляет собой обычный текстовый файл, содержащий маркированный тегами форматирования текст, а также заданные специальными тегами ссылки на графические и прочие файлы мультимедиа, документы HTML и ресурсы Internet.

Документ HTML начинается открывающим тегом <HTML> и заканчивается закрывающим тегом </HTML>. Между данной парой контейнерных тегов располагаются две другие основные части HTML

документа: *заголовок*, заключенный в контейнер <HEAD>...</HEAD> и *тело* документа в контейнере <BODY>...</BODY>.

Таким образом, *структура простого HTML-документа* выглядит примерно так:

```
<!DOCTYPE html> <!-- Объявление формата документа -->
<HTML>
<HEAD> <!-- Техническая информация о документе -->
<META charset="UTF-8"> <!-- Определяем кодировку символов документа -->
<TITLE>...</TITLE> <!-- Задаем заголовок документа -->
<LINK rel="stylesheet" type="text/css" href="style.css"> <!-- Подключаем внешнюю таблицу стилей -->
<SCRIPT src="script.js"></SCRIPT> <!-- Подключаем сценарии -->
</HEAD>
<BODY>
<!-- Основная часть документа -->
</BODY>
</HTML>
```

Объявление <!DOCTYPE>:

Элемент <!DOCTYPE> должен первым указываться в документе HTML. Он отвечает за корректное отображение Web-страницы браузером и сообщает Web-серверу способ обработки документа и то, какие дескрипторы могут находиться на странице, хотя чаще всего он игнорируется браузерами. Поэтому его применение не обязательно.

Синтаксис:

```
<!DOCTYPE HTML "текст" "URL">
```

Здесь текст определяет версию HTML, а URL позволяет браузерам пользователей загрузить DTD, например:

```
<! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 5.0// EN"
   "http://WWW.W3.ORG/TR/REC-HTML/strict.dtd">
```

Тэг <HTML>:

Тэг <HTML> является корневым элементом документа, ему соответствует конечный тэг </HTML>. Все, что находится за пределами тегов, не воспринимается браузером как код HTML и никак им не обрабатывается. Как и <!DOCTYPE>, тэги <HTML> и </HTML> не являются строго обязательными. Но их использование является правилом хорошего тона, т. к. браузеры у пользователей могут быть различными и не известно, насколько корректно они визуализируют такой код.

Заголовочные тэги

Элемент <HEAD>:

Раздел <HEAD>...</HEAD> содержит техническую информацию о странице: заголовок, описание, ключевые слова для поисковых машин, кодировку и т. д. Введенная в нем информация не отображается в окне браузера, однако содержит данные, которые указывают браузеру, как следует обрабатывать страницу.

Элемент <TITLE>:

Обязательным тегом раздела <HEAD> является <TITLE>. Текст, размещенный внутри этого тега, отображается в строке заголовка веб-браузера. Длина заголовка должна быть не более 60 символов. Текст заголовка должен содержать максимально полное описание содержимого веб-страницы.

Синтаксис:

```
<TITLE>последовательность символов</TITLE>
```

Элемент <META>:

Необязательным тегом раздела <HEAD> является одинарный тег <META>. С его помощью можно задать описание содержимого страницы и ключевые слова для поисковых машин, автора HTML-документа и прочие свойства метаданных. Элемент <HEAD> может содержать несколько элементов <META>, потому что в зависимости от используемых атрибутов они несут различную информацию.

Синтаксис:

```
<META NAME=имя элемента метаинформации
    CONTENT=содержимое информации>
```

или

```
<META HTTP-EQUIV=имя элемента метаинформации
      CONTENT=содержимое информации>
```

Элемент <STYLE>:

Внутри этого элемента на языке CSS задаются стили, которые используются на странице. Таких элементов на странице может быть несколько.

Внутри <STYLE> можно записывать код форматирования как данного элемента, так и веб-страницы целиком.

Элемент <LINK>:

Подключить файл со стилями к веб-странице можно при помощи элемента <LINK>, не требующего закрывающего тега. <LINK> определяет связь между текущей страницей и другими документами. Таких элементов может быть несколько. Запись будет иметь следующий вид:

```
<link rel="stylesheet" href="style.css"
            type="text/css">
```

Элемент <SCRIPT>:

Элемент <SCRIPT> позволяет присоединять к документу различные сценарии. Закрывающий тег обязателен, при этом текст сценария может располагаться либо внутри этого элемента, либо во внешнем файле. Если текст сценария расположен во внешнем файле, то он подключается с помощью атрибутов элемента.

Основные элементы тела документа

Элемент <BODY>:

Элемент <BODY> предназначается для выделения той части документа, которая будет визуализирована для пользователя. Он имеет как начальный, так и конечный теги. Начальный тег <BODY> может иметь несколько атрибутов.

Вложенные атрибуты элемента <BODY>:

BackGround – атрибут, задающий графическое изображение, которое, как черепица, заполнит фон документа.

Синтаксис:

```
<BODY BACKGROUND="(URL)(путь) имя файла картинки">
```

Браузер воспринимает картинки со следующими расширениями графических файлов: .bmp, .gif, .jpg, .jpeg. Файл картинки должен быть помещен в папку, где находится файл данного HTML-документа. В «имя файла картинки» необходимо указывать полное имя файла вместе с расширением, например, BackGround=”aaa.gif”.

BgColor – этот атрибут задает цвет фона документа при помощи шестнадцатеричных значений интенсивности цветов RGB или строчного литерала, соответствующего названию цвета.

Синтаксис:

```
<BODY BGCOLOR="#ff0000">или <BODY BGCOLOR="RED">
```

Text – этот атрибут задает используемый по умолчанию цвет текста, который не является гиперссылкой. По умолчанию такой текст будет черным.

Синтаксис:

```
<BODY TEXT="цвет">
```

Значениями атрибутов *Text* и *BgColor* могут быть зарезервированные названия цветов на английском языке (значение не надо заключать в кавычки) или палитра цветов в виде #C1D191 (значение надо заключать в кавычки).

Палитра цветов определяется в шестнадцатеричной системе исчисления, где первые два символа определяют красный цвет (00 – нет красного, FF – наибольший процент красного), вторые два – зеленый цвет, третьи два – синий цвет. Зарезервированными значениями цветов могут быть:

– aqua (цвет морской волны) #00FFFF;
– black (черный) #000000;
– blue (голубой) #0000FF;
– fuchsia (фуксин) #FF00FF;
– gray (серый) #808080;
– green (зеленый) #008000;
– lime (ярко зеленый) #00FF00;

– maroon (темно-бордовый) #800000;
– navy (темно-синий) #000080;
– purple (фиолетовый) #800080;
– red (красный) #FF0000;
– silver (серебряный) #C0C0C0;
– yellow (желтый) #FFFF00;
– white (белый) #FFFFFF.

К каждому зарезервированному слову может быть без пробела добавлена приставка *dark* (темно) или *light* (светло), например, в виде *darkBlue*.

Пример задания атрибутов в элементе <Body>:

```
<Body BgColor=yellow Text=”#C0C0C0”>
```

Кроме задания палитры цветов в шестнадцатеричной системе исчисления, можно указывать процентное содержимое красного, зеленого и синего:

```
<Body BgColor=yellow Text=”70%,10%,50%”>,
```

где для атрибута *Text* задан цвет с 70 % красного, 10 % зеленого и 50 % синего.

Если процентное содержание всех цветов одинаковое, то это будет серый цвет, т. е. если процентное содержание всех цветов равно нулю – это черный, если процентное содержание всех цветов равно ста – это белый.

1. *LINK* – этот атрибут задает цвет гиперссылки. В большинстве браузеров он задан по умолчанию темно-синим.

Синтаксис:

```
<BODY LINK="цвет">
```

2. *ALINK* – этот атрибут задает цвет активной гиперссылки. Он меняет цвет гиперссылки в момент щелчка по ней мышью. Нежелательно задавать ему цвет фона.

Синтаксис:

```
<BODY ALINK="цвет">
```

3. *VLINK* – этот атрибут задает цвет посещенной гиперссылки. Нежелательно задавать ему цвет фона и цвет атрибута LINK.

Синтаксис:

```
<BODY VLINK="цвет">
```

В отличие от редакторов, оформительская часть текста Web-страницы (например, курсив или жирный текст) должна быть указана определенным элементом, поэтому нет смысла набирать в редакторе текст отличный от обычного его написания.

При наборе текста переносы в окне редактора не будут соответствовать переносам в окне браузера. Переносы в окне браузера выполняются после заполнения соответствующей строки окна браузера. Это же относится и к пробелам между словами – всегда между словами будет один пробел независимо от их количества в редакторе. Принудительный перенос на новую строку в браузере осуществляется элементом
, например, в виде:

```
первая строка первая строка<BR>вторая строка
          вторая строка и т. д.
```

В окне браузера этот текст буде выглядеть следующим образом:

первая строка первая строка
вторая строка вторая строка и т. д.

Как видим, перенос на новую строку в окне браузера выполняется по
 несмотря на то, что в редакторе этот текст записан в одной строке.

Основные оформительские элементы HTML

Элемент базового шрифта:

```
<BaseFont
     Color=red
     Face=”Times New Roman”
     Size=3>
```

Данный элемент задает тип, размер и цвет шрифта, которые применяются по умолчанию для всего документа. Этот элемент необходимо размещать сразу же за элементом <BODY>.

Атрибуты элемента <BaseFont>:

1. *Color* – в этом атрибуте указывается цвет символов одним из допустимых английских слов (в этом случае слово можно не заключать в кавычки) или в палитре красный-зеленый-синий (в этом случае значение надо заключать в кавычки).

2. *Face* – в этом атрибуте указывается название шрифта с соблюдением его принятого написания, т. е. с учетом прописных и заглавных букв в соответствующих местах.

3. *Size* – в этом атрибуте указывается число от 1 до 7, где 1 – самый малый размер шрифта, 7 – самый большой.

По умолчанию выставляется шрифт Times New Roman размером 3 черного цвета.

Элемент фонового звука:

```
<BgSound
     Balance=0
     Loop=100
     Volume=-500
     SRC="aaa.wav">
```

Атрибуты тега <BgSound>:

1. *SRC* указывает имя звукового файла вместе с расширением, предназначенного для воспроизведения. Форматами файла могут быть .wav, .au, .mid. Звуковой файл должен быть помещен в ту же папку, где находится файл данного документа.

2. *Loop* указывает на то, сколько раз будет воспроизводиться звуковой файл. Если вместо числа указать ключевое слово *infinite*, то файл будет воспроизводиться бесконечно.

3. *Balance* может содержать число от –10 000 до 10 000 и балансирует звук между динамиками (значение 0 – одинаковое звучание в обоих динамиках).

4. *Volume* может принимать значение от –10000 до 0 и определяет громкость звука.

Элемент комментария:

```
<!-- комментарий -->
```

Файл HTML может содержать комментарии, дающие пояснения для человека, читающего код. Комментарии не влияют на представление документа, т. е. они игнорируются браузером.

Элемент заголовков:

```
<H1 Align = Right> текст </H1>
```

Текст, помещенный между <H1> и </H1>, будет являться заголовком.

Вместо *H1* можно указывать *H2*, *H3*, *H4*, *H5*, *H6*. Элемент *H1* отображает заголовок самым большим шрифтом, соответственно элемент *H6* – самым малым шрифтом.

Атрибут *align* используется для выравнивания заголовка относительно окна браузера. Значение *right* выравнивает заголовок по правой границе окна браузера, *center* – по центру, *left* – по левой границе окна браузера, *justify* – по ширине окна.

Элемент центрирования текста:

```
<Center>текст</Center>
```

Элемент вставки горизонтальной линии:

```
<HR Align = Center Width = "50%" Size =
            = 2 Color=Red>
```

Атрибуты тега <HR>:

1. *Align* определяет выравнивание линии (см. элемент *H1*).

2. *Width* определяет длину горизонтальной линии в пикселах или в процентах относительно ширины экрана.

3. *Size* определяет толщину горизонтальной линии в пикселах.

4. *Color* определяет цвет линии (см. описание значений цветов для элемента *Body*).

Шрифтовые элементы:

Действие любого из приведенных ниже элементов распространяется на текст, который этот элемент ограничивает:

1. <DFN> текст </DFN> – используется с целью обозначения терминов и определений по типу словарей или глоссариев.

2. <CITE> текст </CITE> – используется с целью обозначения источника информации, из которого взята цитата.

3. <I> текст </I> – курсив.

4. <B> текст </B> – выделенный (полужирный) текст.

5. <Strong> текст </Strong> – сильное выделение (жирный) текста.

6. <U> текст </U> – подчеркивание.

7. <Strike> текст </Strike> – зачеркнутый текст.

8. <TT> текст </TT> – моноширинный шрифт (как на пишущей машинке).

9. <Big> текст </Big> – большой шрифт. Этот элемент увеличивает размер текста на определенную величину. Можно использовать вложенные друг в друга элементы <Big> для многократного увеличения размера шрифта.

10. <Small> текст </Small> – малый шрифт. Этот элемент уменьшает размер шрифта на определенную величину. Можно использовать вложенные друг в друга элементы <Small> для многократного уменьшения размера шрифта. В данном случае

<Big><Small> текст</Small></Big>

текст будет печататься стандартным размером шрифта, так как <Big> увеличивает размер шрифта, а <Small> уменьшает его на эту же величину.

11. _{текст} – подстрочный текст (нижний индекс).

12. ^{текст} – надстрочный текст (верхний индекс).

13. <Font Color=red Size=4>текст</Font> – используется для указания характеристик шрифта. Чаще всего используются два атрибута элемента <Font>: атрибут задания цвета символов *color* (значения этого атрибута такие же, как и для элемента <BODY>) и атрибут указания размера шрифта *size* (значением этого атрибута должно быть число от 1 (для самого малого размера шрифта) до 7. Вместо элемента <Font> рекомендуется использовать листы стилей.

Блочные элементы:

<P align=отступ> Текст абзаца </P> – этот элемент задает один из способов разбиения текста на абзацы. Он может иметь вложенный атрибут *align*, который указывает отступ *left*, *center*, *right*, *justify*. Каждый следующий абзац игнорирует, заданное для предыдущего абзаца значение *align*.

<BlockQuote> текст </BlockQuote> – текст отображается с отступом от края листа с нового абзаца.

<BR cleaR=обтекание> задает разрыв текста с переходом на новую строку. Он может иметь вложенный атрибут *cleaR*, который принимает значения *left*, *all* или *right*, тем самым указывая обтекание текста вокруг плавающих изображений, вставленных в текст нестандартным способом. Каждый следующий абзац игнорирует, заданное для предыдущего абзаца значение *cleaR*. Текст может быть отменен тэгами <NOBR> и </NOBR>.

<PRE> текст </PRE> определяет предварительно отформатированный текст. Переносы на новую строку и количество пробелов между словами будут такими же, как и в редакторе **Блокнот**.

<DIV align=отступ> текст раздела </DIV> позволяет выделить в структуре документа несколько разделов. Он является блочным элементом, функционирующим во многом подобно <P>. Если закрывающий тэг </P> опущен, то <DIV> эффективно заменяет его и начинает новый абзац. Он может иметь атрибут *align*, который указывает отступ *left*, *center* или *right*. Каждый следующий раздел игнорирует заданное для предыдущего раздела значение *align*.

<ADDRESS> контактная информация </ADDRESS> используется для оформления контактной информации текущего документа, например, адрес электронной почты или полный почтовый адрес с номером телефона.

 текст используется с целью выделения особым шрифтом слова или текста.

<acronym Title=”Республика Беларусь”>РБ</acronym> – аббревиатура и ее расшифровка. Расшифровка определяется атрибутом *title* и появляется при указании курсором мыши на аббревиатуру.

<marquee behavior=”alternate” scrolldelay=100 width=500 height=30> текст </marquee> – движение текста (справа налево, до упора в левую границу, и наоборот) в блоке размером 500×30 со скоростью, определенной временем задержки = 100 миллисекунд. Если опустить атрибут *behavior*, то движение текста будет только справа налево.

<cite Title=”Источник”> цитата </cite> указывает, что текст, помещенный в него, является цитатой из книги или другого источника. Текст выводится при этом курсивом. С использованием листов стилей тексту можно придать нужное оформление. Атрибутом *title* можно указывать источник цитаты. Эта информация будет появляться при установке мыши на цитату.

Примеры выполнения заданий

Пример 1. Создать страницу с названием *Заголовки*.

```
<HTML>
<HEAD> <TITLE> Заголовки </TITLE> </HEAD>
<BODY>
  <H1> Заголовок первого уровня </H1>
  <H2> Заголовок второго уровня </H2>
  <H5> Заголовок пятого уровня </H5>
  <H6> Заголовок шестого уровня </H6>
</BODY>
</HTML>
```

Пример 2. Создать страницу с названием *Авторское форматирование*.

```
<HTML>
<HEAD> <TITLE> Авторское форматирование </TITLE>
</HEAD>
<BODY>
  <H1 align=center> Омар Хайам</H1>
  <HR>
  <PRE>
     Чтоб мудро жизнь прожить, знать надобно немало.
     Два важных правила запомни для начала:
     Уж лучше голодать, чем что попало есть,
     И лучше одному, чем вместе с кем попало!
  </PRE>
</BODY>
</HTML>
```

Пример 3. Создать страницу с названием *Размер шрифта*.

```
<HTML>
<HEAD> <TITLE> Размер шрифта </TITLE> </HEAD>
<BODY>
  <p><font size="1">Шрифт размера 1</font></p>
  <p><font size="2">Шрифт размера 2</font></p>
```

```
  <p><font size="3">Шрифт размера 3</font></p>
  <p><font size="7">Шрифт 7</font></p>
  <p><font size="6">У</font>тро красит нежным
    светом</p>
  <p><font size="+2">С</font>тены
    <small>ста-рого</small> <big>Кремля</big></p>
</BODY>
</HTML>
```

Пример 4. Создать страницу с названием *Логические стили*.

```
<HTML>
<HEAD> <TITLE> Логические стили </TITLE> </HEAD>
<BODY>
   <h4>Примеры форматирования текста с помощью
       логических стилей</h4>
   <ABBR>ABBR - отмечает текст как аббревиатуру
       (ABBReviation)</abbr><br>
   <ACRONYM title="Белорусский национальный тех-
       нический университет">
   БНТУ</ACRONYM> - отметка аббревиатуры с атри-
       бутом в виде подсказки<br>
   <CITE>CITE - отметка цитат, названий книг и
       т.д.</cite><br>
   <CODE>CODE - для отметки небольшого фрагмента
       программного кода</code><br>
   <DEL  datetime="2013-02-14t21:40:53+0.00">DEL
       - пример удаленного текста</del><br>
   <DFN> DFN - отмечает текстовый фрагмент как
       определение</dfn><br>
   <INS> INS - отмечает текст как вставку
       </ins><br>
   <EM>EM - от английского emphasis - акцент
       </EM><BR>
   <KBD>KBD - от английского keyboard - клавиа-
       тура</KBD><BR>
   <Q> Q - отмечает короткие цитаты в тексте
       </q><br>
```

```
    <SAMP>SAMP - от английского sample - образец
        </SAMP><BR>
    <STRONG>STRONG - от английского strong empha-
        sis - сильный акцент </STRONG><BR>
    <VAR>VAR - от английского variable - перемен-
        ная </VAR></P>
</BODY>
</HTML>
```

Контрольные вопросы

1. Перечислите базовые элементы технологии Web.
2. Что такое тег?
3. Какой вид имеет тег?
4. Назовите виды тегов.
5. В чем заключается алгоритм набора и просмотра HTML-документа?
6. Какова структура простого HTML-документа?
7. Охарактеризуйте элемент <!DOCTYPE>.
8. Охарактеризуйте элемент <HTML>.
9. Охарактеризуйте элемент <BODY>.
10. Перечислите заголовочные теги.
11. Охарактеризуйте элементы <HEAD>, <TITLE>, <META>, <STYLE>, <LINK>, <SCRIPT>.
12. Каково назначение атрибутов *background*, *bgcolor*, *text*, *link*, *alink*, *vlink*?
13. Назовите и охарактеризуйте элемент базового шрифта.
14. Назовите и охарактеризуйте элемент фонового звука.
15. Назовите и охарактеризуйте элемент комментария.
16. Назовите и охарактеризуйте элемент заголовка.
17. Назовите и охарактеризуйте элемент центрирования текста.
18. Назовите и охарактеризуйте элемент горизонтальной линии.
19. Перечислите шрифтовые элементы. Каково их назначение?
20. Перечислите блочные элементы. Каково их назначение?

Лабораторная работа № 2

СОЗДАНИЕ СПИСКОВ

Цель работы: изучить способы организации различных видов списков.

Теоретические сведения

Организация ненумерованного списка

Ненумерованный список определяется элементом <UL>, внутри которого может находиться элемент заголовка списка <LH> и обязательно один или несколько элементов <LI>, непосредственно самих элементов списка. Фрагмент документа для формирования ненумерованного списка имеет следующий вид:

```
<UL>
   <LH> заголовок списка</LH>
   <LI> первый элемент списка
   <LI> второй элемент списка
   <LI> третий элемент списка
</UL>
```

В элементе <UL> можно указывать тип маркера списка следующим образом:

`<UL Type=Disc>` – заполненный кружок;
`Square` – заполненный квадрат;
`Circle` – незаполненный кружок

В качестве маркера может быть использована любая картинка, подключение которой выполняется с использованием листов стилей.

Пример фрагмента формирования ненумерованного списка приведен ниже:

```
<ul>
    <lh>список клиентов:
    <li>Иванов
    <li>Петров
    <li>Сидоров
</ul>
```

В окне браузера этот список будет выглядеть следующим образом:

список клиентов:
- Иванов
- Петров
- Сидоров

Организация нумерованного списка

Нумерованный список формируется с использованием элемента <OL>, внутри которого может находиться элемент заголовка списка <LH> и обязательно один или несколько элементов <LI>, непосредственно самих элементов списка. Фрагмент документа для формирования нумерованного списка имеет следующий вид:

```
<OL>
    <LH> заголовок списка </LH>
    <LI> первый элемент списка
    <LI> второй элемент списка
    <LI> третий элемент списка
</OL>
```

По умолчанию нумерация осуществляется арабскими цифрами, начиная с 1. Нумерацию можно изменить указанием атрибута *type* в виде:

<OL Type=a>

Атрибут *type* может принимать следующие значения:

1 – арабские цифры;

a – строчные буквы латинского алфавита;

A – прописные буквы латинского алфавита;

I – прописные римские цифры;

i – строчные римские цифры.

Параметр *start* (только для <OL>) определяет начало нового отсчета, например, если нужно не «1, 2, 3», а «24, 25, 26»:

```
<OL START=24>
```

Кроме этого нумерацию можно изменить в элементах <LI>:

`<LI VALUE=3>` – дальнейшая нумерация будет с 3-го символа.

Атрибут *reversed* задает отображение списка в обратном порядке (например, 9, 8, 7…).

Многоуровневый нумерованный список

Многоуровневый список используется для отображения его элементов на разных уровнях, с различными отступами.

Чтобы сделать вложенную нумерацию, нужно использовать следующие свойства:

– *counter-reset* сбрасывает один или несколько счетчиков, задавая значение для сброса;

– *counter-increment* задает значение приращения счетчика, т. е. с каким шагом будет нумероваться каждый последующий пункт;

– *content* – генерируемое содержимое, отвечающее за вывод номера перед каждым пунктом списка.

Пример создания многоуровневого списка:

```
ol {
    /* убираем стандартную нумерацию */
list-style: none;
    /* Идентифицируем счетчик и даем ему имя li.
    Значение счетчика не указано - по умол-
    чанию оно равно 0 */
counter-reset: li;
}
li:before {
    /* Определяем элемент, который будет нуме-
    роваться — li. Псевдоэлемент before указы-
    вает, что содержимое, вставляемое при по-
```

```
        мощи свойства content, будет располагаться
        перед пунктами списка. Здесь же устанавли-
        вается значение приращения счетчика (по
        умолчанию равно 1) */
    counter-increment: li;
        /* С помощью свойства content выводится но-
        мер пункта списка. counters() означает, что
        генерируемый текст представляет собой значе-
        ния всех счетчиков с таким именем. Точка в
        кавычках добавляет разделяющую точку между
        цифрами, а точка с пробелом добавляется пе-
        ред содержимым каждого пункта списка */
    content: counters(li,".") ". ";
    }
```

Список определений

Элемент <DL> списка определений в отличие от нумерованных и ненумерованных списков состоит из двух частей: первая служит для задания термина, вторая – для вывода его определения. В одном <DL> может находиться определение нескольких терминов:

```
<DL>
    <LH> заголовок списка </LH>
    <DT> термин 1
    <DD> определение термина 1
    <DT> термин 2
    <DD> определение термина 2
</DL>
```

Текст после элемента <DD> будет отображаться с отступом от левой границы окна браузера.

Внутри <DD> могут находиться нумерованные и ненумерованные списки.

Примеры выполнения заданий

Пример 1. Создать документ, содержащий нумерованный и маркированный списки.

```
<HTML>
<HEAD> <TITLE> Списки </TITLE> </HEAD>
<BODY>
     <H1 align=center><U>Устройства ввода </U></H1>
     <HR>
     <OL type=I>
          <LI>Клавиатура
          <LI>Сканер
          <LI>Джойстик
          <LI>Веб-камера
          </OL>
     <H1 align=center><U>Устройства вывода </U></H1>
     <HR>
     <UL Type=Circle>
          <LI> Монитор
          <LI> Принтер
          <LI> Колонки
          </UL>
</BODY>
</HTML>
```

Задание. Примените для списка элементы форматирования шрифта и абзаца из лабораторной работы №1.

Пример 2. Список определений.

```
<HTML>
<HEAD> <TITLE> Список определений </TITLE> </HEAD>
<BODY>
    <DL>
    <LH> Компьютерные сети </LH>
    <DT> Протокол обмена -
    <DD>
    это набор правил (соглашение, стандарт), оп-
    ределяющий принципы обмена данными между раз-
    личными компьютерами в сети.
    <DT> Провайдер –
    <DD>
    (англ. обеспечиватель, полностью ISP – Inter-
    net Service Provider) <br> это организация,
```

```
    которая обеспечивает другим организациям и
    частным лицам доступ к сети Internet.
    <DT> Универсальный указатель ресурса или URL
    (Universal Resource Locator)
    <DD>
    включает в себя протокол доступа к документу,
    доменное имя или IP-адрес сервера, на котором
    находится документ, а также путь к файлу и
    имя файла: <br> protocol://domain_name/path/
    file_name
    </DL>
</BODY>
</HTML>
```

Задание. Примените для списка элементы форматирования шрифта и абзаца из лабораторной работы № 1.

Контрольные вопросы

1. Что определяет элемент <UL>?
2. Что определяет элемент <LH>?
3. Что определяет элемент <LI>?
4. Какой вид имеет фрагмент документа для формирования ненумерованного списка?
5. Как указать тип маркера? Перечислите типы маркеров.
6. Что определяет элемент <OL>?
7. Какой вид имеет фрагмент документа для формирования нумерованного списка?
8. Как указать тип номера?
9. Какие значения может принимать атрибут *type* для нумерованного списка?
10. Для чего предназначен атрибут *reversed*?
11. Как задать начальное значение номера для нумерованного списка?
12. Как задать значение номера для нумерованного списка?
13. Что определяет элемент <DL>?
14. Какой вид имеет фрагмент документа для формирования списка определений?

Лабораторная работа № 3

ВСТАВКА ГРАФИКИ И МУЛЬТИМЕДИА

Цель работы: изучить теги для подключения рисунков различных форматов, аудио- и видеофайлов.

Теоретические сведения

Вставка графики

Форматы графических файлов:

– *JPG (Joint Photographic Experts Group)*. JPEG используется в основном для хранения фотографий. Поддерживает 24-битовый цвет и сохраняет все полутона в фотографиях неизменными. Но JPG не поддерживает прозрачность и искажает мелкие детали и текст в изображениях. Файлы этого формата имеют расширения .jpg, .jpe, .jpeg.

– *GIF (Graphics Interchange Format)*. Главное достоинство этого формата – возможность хранить сразу несколько изображений в одном файле. Это позволяет создавать целые анимированные ролики. Также GIF-файлы позволяют установить один из цветов прозрачным, благодаря чему фон веб-страницы может проявляться через часть изображения. Формат подходит для сжатия изображений, в которых есть области со сплошным цветом, например, логотипов. Главный недостаток – это поддержка всего лишь 256-цветов, что не годится для хранения фотографий. GIF в основном используется для хранения логотипов, баннеров, изображений, содержащих прозрачные участки или текст. Файлы этого формата имеют расширение .gif.

– *PNG (Portable Network Graphics)*. Включает в себя лучшие черты GIF- и JPEG-форматов. Содержит 256 цветов и дает возможность сделать один из цветов прозрачным, при этом сжимает изображения в размер несколько меньше, чем GIF-файл.

– *APNG (Animated Portable Network Graphics)*. Формат изображения, основанный на формате PNG. Позволяет хранить анимацию, а также поддерживает прозрачность.

– *SVG (Scalable Vector Graphics)*. SVG-рисунок состоит из набора геометрических фигур, описанных в формате XML: линия, эллипс,

многоугольник и т. п. Поддерживается как статичная, так и анимированная графика. Набор функций включает в себя различные преобразования, альфа-маски, эффекты фильтров, возможность использования шаблонов. Изображения в формате SVG могут изменяться в размере без снижения качества.

– *BMP (Bitmap Picture)*. Представляет собой несжатое (оригинальное) растровое изображение, шаблоном которого является прямоугольная сетка пикселей. Bitmap-файл состоит из заголовка, палитры и графических данных. В заголовке хранится информация о графическом изображении и файле (глубина пикселей, высота, ширина и количество цветов). Палитра указывается только в тех Bitmap-файлах, которые содержат палитровые изображения (8 и менее бит) и состоят не более чем из 256 элементов. Графические данные представляют саму картинку. Глубина цвета в данном формате может быть 1, 2, 4, 8, 16, 24, 32, 48 бит на пиксель.

– *ICO (Windows icon)*. Формат хранения значков файлов в Microsoft Windows. Также, Windows icon используется как иконка на сайтах в интернете. Именно картинка такого формата отображается рядом с адресом сайта или закладкой в браузере. Один ICO-файл содержит один или несколько значков, размер и цветность каждого из которых задаётся отдельно. Размер значка может быть любым, но наиболее употребимы квадратные значки со сторонами 16, 32 и 48 пикселей.

Для внедрения графики на HTML-страницу используется тег <IMG>, который может включать в себя следующие атрибуты:

1. *Src* определяет имя графического файла вместе с его расширением. Сам графический файл должен находиться в той же папке, что и HTML-документ. Синтаксис:

```
src="flower.jpg"
```

2. *Align* задает расположение изображения относительно текста документа:

– *left* – изображение располагается слева от текста;

– *right* – изображение располагается справа от текста;

– *top* – вверх изображения по строке текста;

– *bottom* – (умолчание) низ изображения по строке текста;

– *middle* – середина изображения по строке текста.

3. *Height* – высота изображения. Может указываться в пикселях или % (например, `height: 300px`). Если это свойство не задано, то высота изображения определяется самим файлом.

4. *Width* – ширина изображения в пикселях или %. Если это свойство не задано, то ширина изображения определяется самим файлом.

5. *Border* – ширина рамки в пикселях вокруг изображения (0 – нет рамки).

6. *Hspace* – ширина незаполненного пространства (расстояние от границы изображения до текста) в пикселях слева и справа от изображения.

7. *Vspace* – ширина незаполненного пространства в пикселях снизу и сверху от изображения.

8. *Alt* добавляет альтернативный текст для изображения. Выводится на месте появления изображения до его загрузки или при отключенной графике, а также всплывает подсказкой при наведении курсора мыши на изображение.

9. *SrcSet* создает список источников для изображения, которые будут выбраны, исходя из разрешения экрана. Может использоваться вместе или вместо атрибута *src*. Значением атрибута является одна или несколько строк, разделенных запятой. Например:

```
<img src="flower.jpg"
  srcset="
    img/flower-mobile.jpg 320w,
    img/flower-wide-mobile.jpg 480w,
    img/flower-tablet.jpg 768w,
    img/flower-desktop.jpg 1024w,
    img/flower-hires.jpg 1280w"
  sizes="
    (max-width: 20em) 30vw,
    (max-width: 30em) 60vw,
    (max-width: 40em) 90vw"
  alt="Роза в моём саду">
```

10. *Longdesc* – URL расширенного описания изображения, дополняющее атрибут *alt*. Синтаксис:

```
longdesc = "http: // www.example.com
          /description.txt"
```

11. *Usemap* определяет изображение в качестве карты-изображения. Значение обязательно должно начинаться с символа #. Значение ассоциируется с атрибутами *name* или *id* тега <Map> и создает связь между элементами <Img> и <Map>. Атрибут нельзя использовать, если элемент <Map> является потомком элемента <A> или <Button>. Синтаксис:

```
usemap="#mymap"
```

Вставка звука и видео

Звуковые эффекты на Web-страницах создаются с помощью аудиофайлов. Наиболее распространены следующие форматы:

– *MP3* – самый популярный аудиоформат, использующий сжатие с потерями и позволяющий уменьшить размер файла в несколько раз;

– *AAC (Advanced Audio Codec)* – закрытый кодек, аналог MP3, но, по сравнению с последним, поддерживает более высокое качество звука при сходном размере файла;

– *Ogg Vorbis* – бесплатный формат с открытым кодом, поддерживается в Firefox, Opera и Chrome. Обеспечивает хорошее качество звука, но недостаточно широко поддерживается аппаратными проигрывателями;

– *Wav* – стандартный формат Windows-приложений;

– *Midi* – этот формат содержит запись нот и коды музыкальных инструментов, по которым синтезируется звук.

Звук на страницах часто используется для озвучения событий: нажатия кнопки, переход по гиперссылке, открытия окна. Часто страницы сопровождаются звуковым фоном с помощью тега <BgSound>.

На страницу можно встроить аудиоплеер с помощью тега <EMBED> с атрибутами:

1. *SRC* – URL-адрес звукового файла.

2. *Width*, *height* – размеры панели управления проигрывателя (в пикселях или процентах от полного размера).

3. *Autostart* – можно применить два значения: *true* – начало воспроизведения сразу после загрузки страницы – и *false* – включение звука пользователем.

Для внедрения звукового контента в веб-страницы также используется HTML5-элемент <AUDIO>. В общем виде HTML-разметка имеет следующий вид:

```
<audio src="name.ogg" controls></audio>
```

Атрибуты тега <AUDIO>:

1. *Autoplay* обеспечивает автоматическое воспроизведение аудиофайла сразу же после загрузки страницы.

2. *Controls* указывает браузеру, что нужно отобразить базовые элементы управления воспроизведением (начинать и останавливать воспроизведение, переходить в другое место записи, регулировать громкость).

3. *Loop* – циклическое воспроизведение аудиофайла.

4. *Muted* выключает звук при воспроизведении аудиофайла.

5. *Preload* – атрибут, отвечающий за предварительную загрузку аудиоконтента. Не является обязательным; некоторые браузеры игнорируют его. Возможные значения:

– *auto* (браузер загружает аудиофайл полностью, чтобы он был доступен, когда пользователь начнет его воспроизведение);

– *metadata* (браузер загружает небольшую часть аудиофайла, чтобы определить его основные характеристики);

– *none* (отсутствие автоматической загрузки аудиофайла. Содержит абсолютный или относительный URL-адрес аудиофайла.

Любой видеофайл является файловым контейнером, в котором хранятся другие файлы. Аудио- и видеодорожки объединяются для воспроизведения видеоролика. Метаданные содержат информацию о данном видеоролике: изображение обложки, субтитры и пр. К популярным форматам видеоконтейнеров относятся следующие:

– *Ogg* (.ogv, .oga, .ogx, .ogg) – формат-контейнер с открытым исходным кодом для видеокодека Theora и аудиокодека Vorbis. Работает в Firefox, Chrome и Opera. MIME-тип: video/ogg.

– *MPEG 4* (.mp4) – формат-контейнер для видеокодека H.264 и аудиокодека AAC. Работает в Safari и Chrome. Кодирует видео, в том числе высокой четкости, для полного спектра устройств таких, как iPhone, iPod и iPad. MIME-тип: video/mp4.

– *WebM* (.webm) – формат-контейнер с открытым исходным кодом для видеокодека VP8 от Google и аудиокодека Ogg Vorbis. Работает в Firefox, Chrome, Opera и Adobe Flash Player. MIME-тип: video/webm.

– *Audio Video Interleave* (.avi) – формат, предназначенный для записи звука и движущихся изображений, соответствует спецификации RIFF. MIME-тип: video/vnd.avi, video/avi, video/msvideo, video/x-msvideo.

– *Matroska* (.mkv) – популярный видеоконтейнер, может содержать видео в формате H.264, VP8 или Theora. MIME-тип: video/x-matroska, audio/x-matroska.

На данный момент браузеры поддерживают три основных видео формата: .mp4, .ogg/.ogv, .webm.

Видео в формате *.avi* на сайте средствами HTML5 не воспроизводится. Поэтому его необходимо перекодировать в один из этих трех форматов с соответствующими видео- и аудиокодеками для вывода на сайте.

При просмотре видеопроигрыватель должен его декодировать. Одни используют программное декодирование видеопотока, другие – аппаратное декодирование.

Каждый браузер поддерживает определенный кодек, поэтому, чтобы обеспечить воспроизведение видеоконтента во всех браузерах, видеофайл нужно размещать в нескольких форматах.

H.264 – высококачественный кодек от фирмы MPEG. Делится на профили для поддержки устройств как с минимальными возможностями, так и высокого разрешения.

Ogg Theora – открытый бесплатный стандарт для видео (качество и производительность несколько ниже H.264).

VP8 – открытый бесплатный кодек, сходный по качеству с H.264. Поддерживается в Firefox, Chrome и Opera.

Для вставки видеофайлов можно воспользоваться тегом <EMBED> с атрибутом *SRC* – URL-адрес видеофайла.

Тег <VIDEO> (закрывающий тег обязателен) добавляет, воспроизводит и управляет настройками видеоролика на веб-странице. Путь к файлу задается через атрибут *SRC* или вложенный тег <SOURCE>.

Синтаксис:

```
<video src="video.ogv" controls></video>
```

Атрибуты:

1. *Autoplay* – видео начинает воспроизводиться автоматически после загрузки страницы.

2. *Controls* добавляет панель управления к видеоролику.

3. *Height* задает высоту области для воспроизведения видеоролика.

4. *Loop* повторяет воспроизведение видео с начала после его завершения.

5. *Poster* указывает адрес картинки, которая будет отображаться, пока видео не доступно или не воспроизводится.

6. *Preload* используется для совместной загрузки веб-страницы и видео.

7. *Src* указывает путь к воспроизводимому видеоролику.

8. *Width* задает ширину области для воспроизведения видеоролика.

9. *Muted* выключает звук при воспроизведении видеофайла.

Альтернативные медиаресурсы

Элемент <SOURCE> используется для указания нескольких медиаресурсов для <AUDIO> и <VIDEO>. Элемент добавляет альтернативные видео / аудио, которые браузер может выбрать из предложенных на основании поддерживаемого им типа носителя или кодека.

Атрибуты тега <SOURCE>:

1. *Media* определяет тип медиаустройства (т. е. для каких устройств оптимизирован файл).

2. *Src* содержит абсолютный или относительный URL-адрес медиафайла.

3. *Type* определяет MIME-тип медиафайла.

Примеры выполнения заданий

Пример 1. Выравнивание изображения с помощью параметра *align*.

а)
```
<HTML>
<HEAD> <title> Параметр align="right" </title>
</HEAD>
<BODY>
    <img src="../images/fish.gif"
    align="right">
    Картинка справа, а текст обтекает ее
    слева, и этот текст может занимать не-
    сколько строчек.
</BODY>
</HTML>
```

Результат (рис. 3.1):

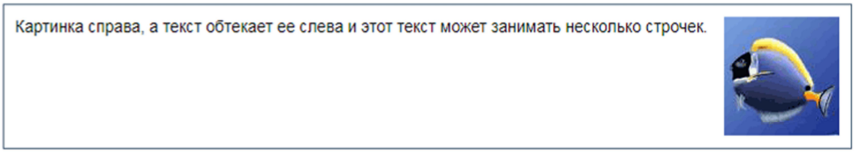

Рис. 3.1

б)
```
<HTML>
<HEAD> <title> Параметр align="middle" </title>
</HEAD>
<BODY>
    <img src="../images/fish.gif"
    align="middle">
    <font size=7>Выравнивание</font>
    середины изображения по базовой линии
    текущей строки.
</BODY>
</HTML>
```

Результат (рис. 3.2):

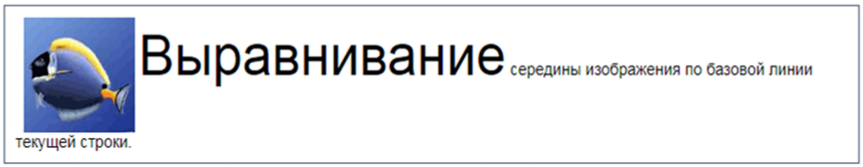

Рис. 3.2

Пример 2. Если необходимо, чтобы текст располагался под картинкой, то нужно использовать тег
 с параметром *clear*, который запрещает обтекание. Обтекание можно запретить с правой стороны (*right*), с левой стороны (*left*) и с обеих сторон (*all*).

```
<HTML>
<HEAD>
    <title>Запрет обтекания картинки</title>
</HEAD>
<BODY>
    <img src="../images/fish.gif">
    <br clear="all"> Остальные элементы документа
</BODY>
</HTML>
```

Результат (рис. 3.3):

Рис. 3.3

Пример 3. Отделение изображения от текста.

```
<HTML>
<HEAD>
    <title>Тег img с отступами </title>
</HEAD>
<BODY>
    <img src="../images/fish.gif" align="left"
    hspace="20" vspace="20">
    Остальное содержимое документа теперь не
    прилипает к изображению.
</BODY>
</HTML>
```

Результат (рис. 3.4):

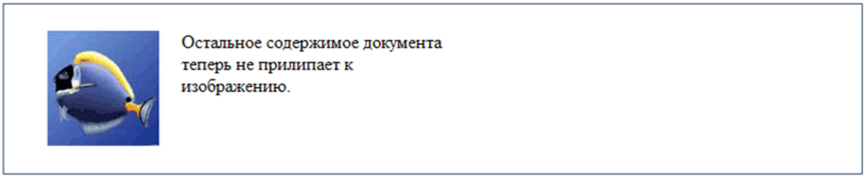

Рис. 3.4

Пример 4. Размеры изображений. Параметры *width* и *height* задаются в пикселях или в процентах.

```
<HTML>
<HEAD>
    <title>Тег img  </title>
</HEAD>
<BODY>
    <img src="../images/fish.gif">
    <img   src="../images/fish.gif"   width="50"
    height="50">
    <img   src="../images/fish.gif"   width="25"
    height="25">
</BODY>
</HTML>
```

Результат (рис. 3.5):

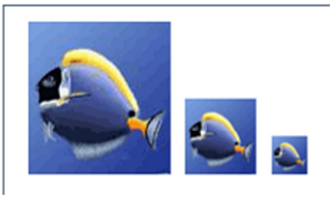

Рис. 3.5

Пример 5. Создать HTML-документ, содержащий аудиоплеер (с использованием тега <Embed>).

```
<HTML>
<HEAD>
    <title>Пример встраивания айдиоплеера </title>
</HEAD>
<BODY>
    <H2> Пример встраивания аудиоплеера </H2>
    <Center>
    <Embed SRC=animals.mp3 WIDTH=287 HEIGHT=213
    autostart=true>
    </Center>
</BODY>
</HTML>
```

Задание. Измените значение параметра *autostart*.

Пример 6. Создать HTML-документ, содержащий аудиплеер (с использованием тега <Audio>).

В настоящий момент не существует аудиоформата, который бы работал во всех браузерах. Поэтому для обеспечения доступности контента максимально широкой аудитории рекомендуется включать несколько источников звука, представленных с использованием атрибута *src* элемента <source>. Одновременно можно добавить резервный контент для браузеров, которые не поддерживают элемент <audio>.

```
<HTML>
<HEAD>
<TITLE> Пример встраивания аудиоплеера</TITLE>
    </HEAD>
<BODY>
<H2> Пример встраивания аудиоплеера </H2>
<audio controls >
     <source src="animals.ogg" type="audio/ogg" >
     <source src="animals.mp3" type="audio/mpeg" >
</audio>
</BODY>
</HTML>
```

Пример 7. Создать HTML-документ для воспроизвдения видеофайла.

Как и в случае с аудиофайлами, рекомендуется перечислять в <source> все форматы, начиная с более предпочтительного.

```
<HTML>
<HEAD>
  <TITLE>video</TITLE>
</HEAD>
<BODY>
    <video controls width="400" height="300">
      <source src="clip.mp4" type="video/mp4">
    <!-- MP4 для Safari, IE9, iPhone, iPad, An-
       droid, и Windows Phone 7 -->
```

```
      <source src="clip.webm" type="video/webm">
    <!-- WebM/VP8 для Firefox4, Opera, и Chrome -->
      <source src="clip.ogv" type="video/ogg">
    <!-- Ogg/Vorbis для старых версий браузеров
       Firefox и Opera -->
      <object data="clip.swf" type="application/x-
       shockwave-flash">
    <!-- добавляем видеоконтент для устаревших
       браузеров, в которых нет поддержки эле-
    мента video -->
        <param name="movie" value="video.swf">
      </object>
</video>
</BODY>
</HTML>
```

Контрольные вопросы

1. Перечислите форматы графических файлов.
2. С помощью какого тега осуществляется вставка графики?
3. Каково назначение атрибутов *src*, *align*, *height*, *width*, *border*, *hspace*, *vspace*, *alt*, *longdesc*, *srcset*, *usemap*?
4. Перечислите возможные варианты расположения рисунка относительно текста?
5. Какой атрибут тега <BODY> применяется для использования картинки в качестве обоев?
6. Перечислите некоторые аудио-форматы.
7. Какой тег предназначен для добавления аудиоплеера?
8. Перечислите атрибуты тега <EMBED>. Каково их назначение?
9. Какой HTML5-элемент используется для внедрения звукового контента в веб-страницы?
10. Перечислите атрибуты тега <AUDIO>. Каково их назначение?
11. Перечислите популярные видеоформаты.
12. Какие видеоформаты поддерживают браузеры?
13. Охарактеризуйте тег <VIDEO>. Каков его синтаксис?
14. Перечислите атрибуты тега <VIDEO>. Каково их назначение?

Лабораторная работа № 4

СОЗДАНИЕ ГИПЕРССЫЛОК

Цель работы: изучить возможные варианты организации гиперссылок и применить их на практике.

Теоретические сведения

Гиперссылки

HTML-ссылки создаются с помощью элементов <A>, <Area> и <link>. Ссылки представляют собой связь между двумя ресурсами, одним из которых является текущий документ.

Ссылки можно поделить на две категории: *ссылки на внешние ресурсы* – ссылки, которые создаются с помощью тега <link> и используются для расширения возможностей текущего документа при обработке браузером; *гиперссылки* – ссылки на другие ресурсы, которые пользователь может посетить или загрузить.

Гиперссылки используются для ускоренного перехода внутри одного документа, для вызова в окно браузера других документов и для просмотра скачивания объектов, не являющихся Web-страницами.

Гиперссылки создаются с помощью парного тега <A></A>. Внутрь тега помещается текст, который будет отображаться на вебстранице. Текст ссылки отображается в браузере с подчеркиванием, цвет шрифта – синий, при наведении на ссылку курсор мыши меняет вид.

Ссылка состоит из двух частей – *указателя* и *адресной части*. Указатель ссылки представляет собой фрагмент текста или изображение, видимые для пользователя. Адресная часть ссылки пользователю не видна. Она представляет собой адрес ресурса, к которому необходимо перейти.

Адресная часть ссылки состоит из URL. *URL* (*Uniform Resource Locator*) – унифицированный адрес ресурса. При создании адресов для разделения слов между собой рекомендуется использовать дефис, а не символ подчеркивания. В общем виде URL имеет следующий формат:

```
метод доступа://имя сервера:порт/путь
```

Метод доступа, или *протокол*, осуществляет обмен данными между рабочими станциями в разных сетях. Наиболее распространенные протоколы передачи данных приведены в табл. 4.1:

Таблица 4.1

Протоколы передачи данных

Название протокола	Назначение	Пример использования
file	обеспечивает чтение файла с локального диска	`file:/gallery/pictures/summer.html`
http	предоставляет доступ к веб-странице по протоколу HTTP	`http://site.ru/`
https	специальная реализация протокола HTTP, использующая шифрование	`https://site.ru/`
ftp	осуществляет запрос к FTP-серверу на получение файла	`ftp://pgu/directory/library`
mailto	запускает сеанс почтовой связи с указанным адресатом и хостом	`mailto: nika@gmail.com`

Имя сервера описывает полное имя машины в сети, например, site.ru. Если имя сервера не указано, то ссылка считается локальной, т. е. она относится к той же машине, на которой находится HTML-документ, содержащий ссылку.

Номер порта TCP, на котором функционирует веб-сервер представляет собой число, которое необходимо указывать, если метод доступа его требует (отдельные серверы могут иметь свой отличительный номер порта). Если порт не указан, по умолчанию используется номер 80. Стандартными портами являются 21 FTP, 23 Telnet, 70 Gopher, 80 HTTP.

Путь содержит имя папки, в которой находится файл.

Атрибуты тега <a>:

1. *Name* задает имя элемента. Имя используется, например, для создания ссылок между фреймами.

2. *Href* задает URL ресурса, на который должен перейти пользователь, щелкнув по ссылке. Атрибут может указывать как на внешний документ, так и на элемент внутри данного документа.

3. *Download* дополняет атрибут *href* и сообщает браузеру, что ресурс должен быть загружен в момент, когда пользователь щелкает по ссылке, вместо того, чтобы, например, предварительно открыть его (как PDF-файл). Задавая имя для атрибута, мы, таким образом, задаем имя загружаемому объекту. Разрешается использовать атрибут без указания его значения:

```
<a href="/images/logo.png" download><img
    src="/images/logo.png" alt="logo"></a>
<a href="/images/logo.png" down-
    load="logo"><img src="/images/logo.png"
    alt="logo"></a>
<a href="files/20022014.pdf" download="Отчет
    Февраль 2014.pdf">Загрузить отчет за
    Февраль 2014</a>
```

4. *Hreflang* определяет язык связанного веб-документа. Используется только вместе с атрибутом *href*. Принимаемые значения – аббревиатура, состоящая из двух букв, обозначающих код языка, например:

```
<a href="http://www.anysite.ru" hre-
flang="en">Anysite</a>
```

5. *Target* указывает на то, в каком окне должен открываться документ. Принимает следующие значения:

– *_self* (страница загружается в текущее окно);

– *_blank* (страница открывается в новом окне браузера);

– *_parent* (страница загружается во фрейм-родителе);

– *_top* (страница загружается в полное окно браузера).

6. *Rel* дополняет атрибут *href* информацией об отношении между текущим и связанным документом. Принимаемые следующие значения:

– *alternate* – ссылка на альтернативную версию документа (например, печатную форму страницы, перевод);

– *author* – ссылка на автора документа;

– *bookmark* – постоянный URL-адрес, используемый для закладок;

– *help* – ссылка на справку;

– *license* – ссылка на информацию об авторских правах на данный веб-документ;

– *next* / *prev* указывает связь между отдельными URL. Благодаря этой разметке, Google может определить, что содержимое данных страниц связано в логическую последовательность;

– *nofollow* запрещает поисковой системе переходить по ссылкам на данной странице или по конкретной ссылке;

– *noreferrer* указывает, что при переходе по ссылке браузер не должен посылать заголовок HTTP-запроса (Referrer), в который записывается информация о том, с какой страницы пришел посетитель сайта;

– *prefetch* указывает, что целевой документ должен быть кэширован, т. е. браузер в фоновом режиме загружает содержимое страницы к себе в кэш;

– *search* указывает, что целевой документ содержит инструмент для поиска;

– *tag* указывает ключевое слово для текущего документа.

Создание изображения-ссылки

Чтобы сделать кликабельное изображение, необходимо поместить элемент <Img>внутрь тега <a>. Чтобы ссылка открывалась в другом окне, нужно добавить атрибут target="_blank" для ссылки.

```
<a  href="http://www.fast-torrent.ru/film/gran-za-
     granyu-tv.html" target="_blank">
<img src="//html5book.ru/wp-
     content/uploads/2014/07/Fringe.jpg"
     alt="Fringe">
</a>
```

Переходы в пределах одного HTML-документа

Якоря, или *внутренние ссылки*, создают переходы на различные разделы текущей веб-страницы, позволяя быстро перемещаться между ними. Это оказывается очень удобным в случае, когда на странице слишком много текста. Внутренние ссылки также создаются при помощи тега <a> с разницей в том, что атрибут *href* содержит имя указателя, так называемого якоря, а не URL-адрес. Перед именем указателя всегда ставится знак #.

Шаг 1. Создайте в точке, куда будет осуществляться переход, элемент:

```
<A Name = "xxx"></A>
```

где xxx – имя метки, куда осуществляется переход.

Метка может быть любым текстом, состоящим из букв и цифр. Но она должна быть уникальной среди всех меток данного документа.

Шаг 2. Создайте в точке, из которой осуществляется переход, элемент:

```
<A HREF="#xxx">текст перехода</A>
```

При просмотре HTML-документа браузером *текст перехода* будет выделен подчеркиванием и отображаться чаще всего синим цветом. Щелчок клавишей мыши по *тексту перехода* приведет к переходу на метку *xxx* в пределах данного HTML-документа.

Вместо <A Name = "xxx"></A>, в качестве метки перехода можно использовать идентификатор *id*. Идентификатор *id* доступен для любого элемента и определяет уникальную во всем документе метку именно этого объекта, например, <B ID=”b1”>, где *b1*, может быть любым текстом, состоящим из латинских букв, с соблюдением регистра, цифр и знаков подчеркивания. Первым символом в *b1* может быть только латинская буква.

Следующая разметка создаст оглавление с быстрыми переходами на соответствующие разделы:

```
<h1>Времена года</h1>
<h2>Оглавление</h2>
<a href="#p1">Лето</a> <!--создаём якорь, указав
   #id элемента-->
<a href="#p2">Осень</a>
<a href="#p3">Зима</a>
<a href="#p4">Весна</a>
<p id="p1">...</p> <!--добавляем соответствующий
   id элементу-->
<p id="p2">...</p>
<p id="p3">...</p>
<p id="p4">...</p>
```

Если нужно сделать ссылку с одной страницы сайта на определенный раздел другой, то необходимо задать *id* для этого раздела страницы, а затем добавить его к абсолютному адресу ссылки:

```
<th id="about-color">color</th>
<a href="https://html5book.ru/css-shrifty/#about-
   color" class="site" target="_blank">color</a>
```

Вызов в окно браузера нового HTML-файла

Вызов в окно браузера нового HTML-файла осуществляется с помощью элемента

```
<A HREF="new.html" Title = "моя биогра-
          фия">текст</A>
```

Атрибут *Title* используется для отображения всплывающей подсказки при установке указателя мыши на текст. Конечно, если элемент <A> окаймляет текст, то всплывающая подсказка не имеет смысла (сам текст уже содержит в себе какой-то смысл). Однако, вместо текста, в качестве ссылки может использоваться картинка, и в этом случае всплывающая подсказка может оказаться необходимой.

Текст указывает то место, откуда будет загружаться HTML-файл: new.html, указанный в атрибуте HREF. Файл new.html должен находиться в той же папке, что и файл данного HTML-документа

Вместо текста элемент <A> может использовать графический файл:

```
<A HREF="new.html" Title = "моя биография">
        <Img src=”aaa.gif” ></A>
```

Загрузка документов MS Office

Загрузка документов Word, Excel, Access осуществляется с помощью элемента

```
<A HREF=”aaa.doc”>текст</A>
```

В данном примере будет открыта программа Word с загруженным в нее файлом aaa.doc.

Ссылка на изображение

Ссылка на картинку необходима, если требуется просмотреть изображение в натуральную величину в сравнении с ее уменьшенным изображением на сайте:

```
<A Target=”_blank” HREF="xxx.gif">
<img src=”xxx.gif”  width=50 height=50 ></A>
```

Ссылка на адрес электронной почты, номер телефона

```
<A Href=”mailto:Ivanov@Narod.ru”> Ivanov@Na-
                    rod.ru</A>
<a href="tel:+74951234567">+7 (495) 123-45-67</a>
```

При размещении гиперссылок в одной строке между ними не образуется пробела. Для организации пробела между гиперссылками используется неразрывный пробел :

```
<a ….>текст1</a> <a ….>текст2</a>
```

В данном примере *текст2* будет отделен от *текст1* одним пробелом.

Новый документ или сайт можно загрузить по кнопке, а не по ссылке:

```
<form>
     <input  type="button"  value="  Текст  на
     кнопке " onclick="HomeButton()">
     <script>
          function HomeButton()
          {
          location.href="tut.by";
          }
     </script>
</form>
```

Вместо сайта (в данном примере tut.by) для *onClick* можно задать все, что допустимо для внешней гиперссылки <a>.

Кнопку можно использовать и для переходов внутри документа:

```
<form>
    <input type=button value="надпись на кнопке"
    onClick="location.href='index.html#b1';">
</form>
```

где *index.html* – имя документа с данной кнопкой, а *b1* – метка в этом документе, куда делается переход.

Структура HTML-приложения

Цепочка. Это самая простая структура (рис. 4.1). Странички просматриваются последовательно. На каждой странице предусмотрены ссылки на следующую страницу и на предыдущую.

Рис. 4.1

Простое меню. В этой структуре на главной станице расположены гиперссылки на дополнительные страницы (рис. 4.2). С каждой из них можно перейти только на главную станицу.

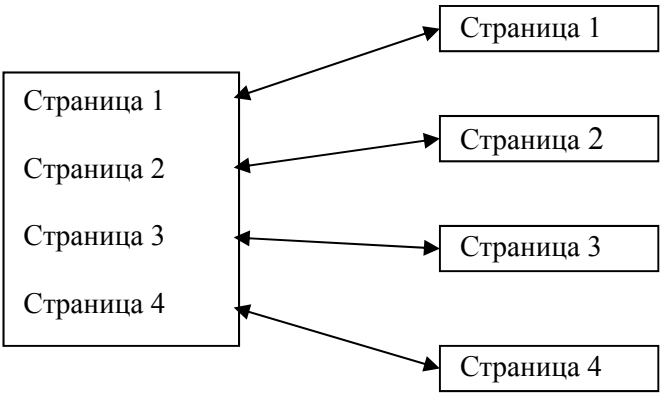

Рис. 4.2

Иерархия. Это зависимость по подчинению или включению одних объектов в другие.

Примеры выполнения заданий

Пример 1. Создайте 3 HTML-документа, отличающиеся цветом фона. Сохраните их под именами Start1.html, Middle.html, Finish.html. Создайте гиперссылки с первого документа на второй, со второго на первый и третий, с третьего на второй.

Первый HTML-документ:

```
<HTML>
<HEAD> <TITLE> Синий цвет</TITLE> </HEAD>
<BODY bgcolor=blue text=yellow>
    <H1> Страница СТАРТ </H1>
    <HR>
    <A href=Middle.html>
    <IMG align=right src=StrPr.png> </A>
</BODY>
</HTML>
```

Второй HTML-документ:

```
<HTML>
<HEAD> <TITLE> Голубой цвет</TITLE> </HEAD>
<BODY bgcolor=cyan text=red>
    <H1> Страница Середина </H1>
    <HR>
    <A href=Finish.html>
    <IMG align=rigth src=StrPr.png> </A>
    <A href=Start1.html>
    <IMG align=left src=StrL.png> </A>
</BODY>
</HTML>
```

Третий HTML-документ создайте самостоятельно по аналогии с предыдущими.

Пример 2. Создайте гипертекстовый документ.

Указание. В своей папке создайте еще одну «Гиперссылки». Скопируйте в нее рисунки Tigr.jpg, Loshad.jpeg, Slon.jpg и любую

фоновую картинку. Создайте следующие документы и свяжите их гипертекстовой связью.

Документ Start.html. В качестве фона – рисунок; заголовок – «Мой зоопарк» синим цветом; горизонтальная линия произвольной ширины. Маркированный список: Тигр, Лошадь, Слон. На каждый элемент списка гиперссылка на документ Zoo1.html, Zoo2.html, Zoo3.html соответственно.

```
<HTML>
<HEAD> <TITLE> Гиперссылки</TITLE> </HEAD>
<BODY background=Fon.jpg>
    <H1 align=center>
    <Font color=blue> Мой зоопарк</Font>
    </H1>
    <HR>
    <UL type=disk>
        <LI> <a href=zoo1.html> Тигр </A>
        <LI> <a href=zoo2.html> Лошадь </A>
        <LI> <a href=zoo3.html> Слон</A>
    </UL>
</BODY>
</HTML>
```

Документ Zoo1.html. В качестве фона – цвет Lime; заголовок – «Лошадь» синим цветом; горизонтальная линия произвольной ширины. Картинка Tigr.jpg прижата к правому краю. Текст с авторским форматированием: «Эй, не стойте слишком близко: я тигренок, а не киска!» Под горизонтальной линией текст «Вернуться назад» с гиперссылкой на документ Start.html.

```
<HTML>
<HEAD> <TITLE> Гиперссылки</TITLE> </HEAD>
<BODY bgcolor=lime>
    <H1 align=center>
        <Font color=blue> Тигр </Font>
    </H1>
    <HR>
    <IMG src=Tigr.jpg  align=right>
    <Pre> <Font Size=+3 color=red>
```

```
        <B>
        Эй, не стойте слишком близко:
           я тигренок,
                а не киска!
        </B></Font>
    </Pre>
    <HR size=7 width =50% align=left nashade>
    <P>
    <A href=Start.html> Вернуться назад </A></P>
</BODY>
</HTML>
```

Документ Zoo2.html. В качестве фона – цвет Teal; заголовок – «Тигр» синим цветом; горизонтальная линия произвольной ширины. Картинка Loshad.jpg прижата к левому краю. Текст с авторским форматированием: «Я люблю свою лошадку. Причешу ей шерстку гладко. Гребешком приглажу хвостик и верхом поеду в гости!» Под горизонтальной линией текст «Вернуться назад» с гиперссылкой на документ Start.html.

Документ Zoo3.html. создайте самостоятельно по аналогии с предыдущими документами.

Задание. Измените документы Zoo1.html, Zoo2.html, сделав якорем не текст, а картинки.

Контрольные вопросы

1. С помощью каких элементов создаются HTML-ссылки?
2. Назовите категории ссылок?
3. Для чего используются гиперссылки?
4. Какой элемент используется для создания гиперссылок?
5. Из каких частей состоит ссылка?
6. Из чего состоит адресная часть?
7. Какой формат имеет URL?
8. Для чего предназначен протокол?
9. Что представляет собой имя сервера?
10. Что представляет собой путь?
11. Что такое якорь?

12. Охарактеризуйте атрибуты *name*, *href*, *download*, *hreflang* тега <A>.

13. Охарактеризуйте атрибут *target* тега <A>. Какие он может принимать значения?

14. Охарактеризуйте атрибут *rel* тега <A>. Какие он может принимать значения?

15. Как реализуется переход в пределах одного HTML-документа?

16. Что можно использовать в качестве метки перехода вместо <A Name = "xxx"></A>?

17. Как осуществляется вызов в окно браузера нового HTML-файла?

18. Как осуществляется загрузка документов Word, Excel, Access?

19. Как реализуется ссылка на графическую картинку?

20. Как реализуется ссылка на адрес электронной почты, номер телефона?

21. Как организовать пробел между гиперссылками?

22. Как загрузить новый документ или сайт по кнопке?

23. Какие структуры HTML-приложения Вам известны?

24. Охарактеризуйте структуру *Цепочка* HTML-приложения.

25. Охарактеризуйте структуру *Простое меню* HTML-приложения.

26. Охарактеризуйте *Иерархическую* структуру HTML-приложения.

Лабораторная работа № 5

ИСПОЛЬЗОВАНИЕ СЛОЕВ И КАРТ-ИЗОБРАЖЕНИЙ

Цель работы: приобрести практические умения использования слоев и карт-изображений.

Теоретические сведения

Слои

Универсальным способом создания слоев является сочетание тега <DIV> и каскадных таблиц стилей (CSS).

Шаблон HTML-кода слоя выглядит следующим образом:

```
<DIV STYLE="Свойства слоя">
    Содержимое слоя
</DIV>
```

В контейнере <DIV> расположены HTML-теги, определяющие элементы, из которых состоит слой.

Свойства слоя записываются следующим образом:

```
свойство:значение.
```

Свойства слоев определены в табл. 5.1.

Таблица 5.1

Свойства слоев

Свойство	Назначение
position: absolute | static | relative	Точка начала отсчета координат
top: число	Y-координата верхнего левого угла слоя
left: число	X-координата верхнего левого угла слоя
width: число	Ширина слоя
height: число	Высота слоя
color: цвет	Цвет текста
background: цвет	Цвет фона слоя
background-image: url (путь)	Фоновое изображение слоя
visibility: visible | hidden	Первоначальная видимость слоя
z-index: число	Порядок отображения слоев

Карты-изображения

Многие Web-страницы для организации ссылок используют так называемые *карты-изображения*. Реализация этой возможности предусмотрена языком HTML и позволяет привязывать гипертекстовые ссылки к различным областям изображения. Такой подход нагляднее, чем применение обыкновенных текстовых связей, поскольку пользователь может не читать словесное описание связи, а сразу понять ее смысл по графическому образу.

Концепция карты-изображения на Web-страницах может быть реализована в двух различных вариантах: серверный (server-side imagemap) и клиентский вариант (client-side imagemap). Последнее название часто используют в виде аббревиатуры CSIM. Первым появился и получил распространение серверный вариант карт-изображений, который впервые был реализован в браузере Mosaic.

В использовании карт-изображений есть как положительные, так и отрицательные моменты.

Карты-изображения наиболее удобно использовать в следующих ситуациях:

1. Для представления пространственных связей, например, географических координат, которые было бы трудно задать отдельными кнопками или текстом. В качестве примера может быть приведена карта Северной Америки, на которой выбор каждого из штатов ведет к переходу на соответствующую страницу.

2. В качестве меню верхнего уровня, появляющегося на каждой странице. Наличие такого меню предоставляет возможность перехода в интересующий раздел сервера с любой страницы и в любой момент. Создание общего графического меню позволит сократить время разработки HTML-документов, поскольку будет использоваться один и тот же файл описания ссылок. Вместо того, чтобы на каждой странице устанавливать связи с различными частями начальной страницы, достаточно сослаться на общее меню.

К недостаткам карт-изображений можно отнести следующие:

1. Если не предусмотрено альтернативное текстовое меню, то не остается никаких средств навигации для пользователей, которые не могут загрузить графику.

2. Картам-изображениям свойственны общие недостатки, присущие использованию изображений на Web-страницах, а именно зна-

чительное увеличение времени загрузки по сравнению с чисто текстовыми документами.

3. Неудачно спроектированные изображения могут внести путаницу. Иногда бывает трудно определить области на изображении, являющиеся активными. Особенно это трудно сделать в серверном варианте. При реализации клиентского варианта ситуация упрощается, так как есть возможность перемещать мышь в пределах изображения и следить за появляющимися адресами ссылок в нижней части окна браузера.

4. При использовании карт-изображений браузер не имеет возможности отмечать другим цветом уже пройденные ссылки так, как это делается для текстовых ссылок.

Для определения карты-изображения, обрабатываемой клиентом, используются теги <MAP> и <AREA> (рис. 5.1).

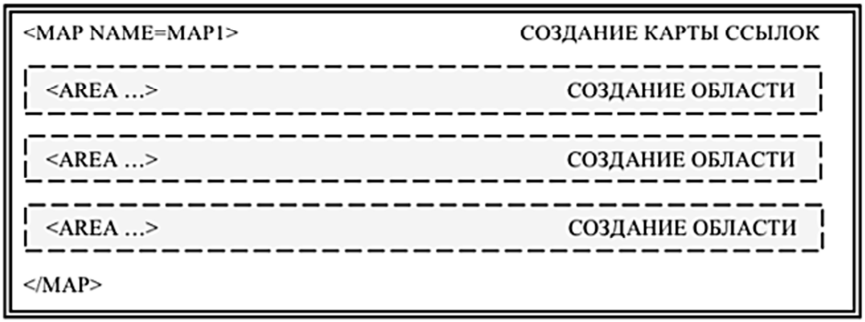

Рис. 5.1. Структура карты ссылок

Тэг <MAP>

Для описания конфигурации областей карты-изображения используется специальный тег <MAP>, единственным параметром которого является *name*. Значение параметра *name* определяет имя, которое должно соответствовать имени в *usemap*. Тег <MAP> требует закрывающего тега </MAP>. Внутри этой пары тегов должны располагаться описания активных областей карты, для чего используется специальный тег <AREA>.

Тэг <AREA>

Каждый отдельный тег <AREA> задает одну активную область. Завершающий тег не требуется. Активные области могут перекрываться. В случае если некоторая точка относится одновременно к нескольким активным областям, то будет реализована та ссылка, описание которой располагается первым в списке областей.

Атрибутами тега <AREA> являются *shape*, *coords*, *href*, *nohref*, *target* и *alt*. Рассмотрим назначение этих параметров.

Атрибут *shape* определяет форму активной области. Может принимать следующие значения:

– *rect* (активная область прямоугольной формы);

– *circle* (активная область в форме круга);

– *poly* (активная область в форме многоугольника);

– *default* (активная область занимает всю площадь изображения).

Если параметр *shape* опущен, то по умолчанию предполагается значение *rect*, т. е. область в виде прямоугольника.

Атрибут *coords* задает координаты отдельной активной области. Значением параметра является список координат точек, определяющих активную область, разделенных запятыми. Координаты записываются в виде целых неотрицательных чисел. Начало координат располагается в верхнем левом углу изображения, которому соответствует значение 0,0. Первое число определяет координату по горизонтали, второе – по вертикали. Список координат зависит от типа области:

– для области типа *rect* задаются координаты верхнего левого и правого нижнего углов прямоугольника;

– для области типа *circle* задаются три числа – координаты центра круга и радиус;

– для области типа *poly* задаются координаты вершин многоугольника в нужном порядке. Заметим, что последняя точка в списке координат не обязательно должна совпадать с первой. Если они не совпадают, то при интерпретации данных для этой формы области браузер автоматически соединит последнюю точку с первой. Различные редакторы карт-изображений в этом отношении работают по-разному: одни добавляют первую точку в конец списка, а другие – нет. Количественные ограничения на число вершин довольно велики и покрывают практически все мыслимые потребности. По крайней мере многоугольник, имеющий 100 вершин, уве-

ренно обрабатывается всеми ведущими браузерами. Есть ограничение, связанное с самим языком HTML, согласно которому список не может содержать более 1024 значений;

– область типа *default* не требует задания координат.

Атрибуты *href* и *nohref* являются взаимоисключаемыми. Если не задан ни один из этих параметров, то считается, что для данной области не имеется ссылки. То же самое явно определяет параметр *nohref*, не требующий значения. Параметр *href* определяет адрес ссылки, которая может записываться в абсолютной или относительной форме. Правила записи ссылок те же, что в тэге <A>.

Атрибут *nohref* полезно использовать для исключения части активной области. Пусть, например, необходимо создать активную область в виде кольца. Такой тип области не предусмотрен в списке возможных областей, однако он может быть реализован путем задания двух круговых областей. Для этого сначала следует задать область меньшего радиуса и указать в качестве параметра *nohref*. Далее нужно задать область большего радиуса с центром в той же точке и указать нужную ссылку. Область внутри кольца, определенная двумя окружностями различного радиуса, не будет иметь ссылку. Использование подхода, основанного на взаимном перекрытии областей, позволит строить области весьма разнообразной формы.

Атрибут *target* указывает, куда будет загружен документ при переходе по ссылке. Принимает следующие значения:

– *_self* (страница загружается в текущее окно);

– *_blank* (страница открывается в новом окне браузера);

– *_parent* (страница загружается во фрейм-родитель);

– *_top* (страница загружается в полное окно браузера).

Атрибут *alt* позволяет записать альтернативный текст для каждой из активных областей изображения. По существу этот текст будет играть лишь роль комментария для создателя документа. Альтернативный текст, записанный для всего изображения (в тэге <IMG>), выдается на экран при работе с отключенной загрузкой изображений, поэтому для активных областей он никогда не появится.

Примеры задания областей различных типов:

```
<MAP NAME="logo">
    <AREA SHAPE=rect COORDS="33,60,191,246"
       HREF="r.htm" ALT="Прямоугольная область">
```

```
    <AREA SHAPE=circle COORDS="366,147,109"
       HREF="c.htm" ALT="Круговая область">
    <AREA SHAPE=poly
       COORDS="534,62,699,62,698,236,626,261,
       534,235"
    HREF="p.htm" ALT="Многоугольник"> <AREA
       SHAPE=default HREF="default.htm">
</MAP>
```

При добавлении изображения, являющегося картой ссылок, используется атрибут *usemap*, значение которого – имя карты, указанное в теге <MAP>. Перед именем карты ставится символ #.

Например:

```
<IMG SRC=MAP.JPG
 WIDTH=100 HEIGHT=100 USEMAP=#MAP1>
```

Алгоритм создания карты-изображения

1. Размечаем исходное изображение на активные области нужной формы. Координаты областей можно вычислить с помощью программы для обработки фотографий, например, Adobe Photoshop или Paint.

2. Задаем имя карты, добавив его в тег <Map> с помощью атрибута *name*. Это же значение присваиваем атрибуту *usemap* тега <Img>.

3. Добавляем ссылки на веб-страницы или части веб-документа для каждой активной области, по которым пользователь будет переходить при нажатии на активную область изображения.

Примеры выполнения заданий

Пример 1. Создание слоев, результат на рис. 5.2.

```
<HTML>
<HEAD>
    <title>Слои  </title>
</HEAD>
<BODY>
```

```
    <div style="position:absolute; top:450;
    left:300; width:180; height:50;
     background:yellow; color:blue; z-index:1">
        <H2 align=center>  Наша планета </H2>
    </div>
    <div style="position:absolute; top:0; left:0;
    z-index:0">
        <img src="zemlja.jpg" width="500"
        height="500" border=0>
    </div>
</BODY>
</HTML>
```

Рис. 5.2. Слои на HTML-странице

Пример 2. Создание карты-изображения с регионами клика (рис. 5.3).

```
<HTML>
<BODY>
    <p>Кликните на солнце или на одной из планет,
    чтобы увидеть их ближе:</p>
    <img src="planets.gif" width="145" height="126"
    alt="Планеты" usemap="#planetmap" />
    <map name="planetmap">
```

```
        <area  shape="rect"  coords="0,0,82,126"
        alt="Солнце" href="sun.gif" />
        <area  shape="circle"  coords="90,58,3"
        alt="Меркурий" href="mer.gif" />
        <area  shape="circle"  coords="124,58,8"
        alt="Венера" href="ven.gif" />
    </map>
</BODY>
</HTML>
```

Кликните на солнце или на одной из планет, чтобы увидеть их ближе:

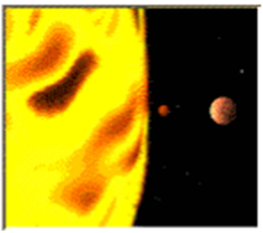

Рис. 5.3. Созданная карта ссылок

Контрольные вопросы

1. Назовите универсальный способ создания слоев.
2. Как выглядит шаблон HTML-кода для создания слоя?
3. Как записываются свойства слоя?
4. Каково назначение свойства *position* слоя? Какие значения может принимать это свойство?
5. Каково назначение свойств слоя *top*, *left*, *width*, *height*, *color*, *background*, *background-image*, *z-index*?
6. Каково назначение свойства *visibility* слоя? Какие значения оно может принимать?
7. Каким образом может быть реализована концепция карты-изображения на Web-страницах?
8. В каких ситуациях наиболее удобно использовать карты-изображения?
9. Перечислите недостатки карт-изображений?

10. Какие теги используются для определения карты-изображения, обрабатываемой клиентом?

11. Перечислите атрибуты тега <AREA>. Каково их назначение?

12. Какой атрибут используется при добавлении изображения, являющегося картой ссылок?

13. Какой символ ставится перед именем карты?

Лабораторная работа № 6

ТАБЛИЦЫ В HTML

Цель работы: ознакомиться с элементами создания простых и сложных таблиц, а также с элементами их оформления.

Теоретические сведения

Таблицы являются одними из самых используемых при формировании HTML-документа элементов. Часто таблицы используются при формировании меню, а еще чаще для рационального и компактного размещения информации на странице.

Элемент <Table> используется для указания начала и конца таблицы, ее общих свойств.

Элемент <Table> может содержать следующие атрибуты:

1. *Width* – ширина таблицы в пикселях (width=200) или в процентах от ширины страницы (width=50%).

2. *Align* устанавливает расположение таблицы по отношению к левой и правой границам документа. Допустимые значения:

– *left* (выравнивание по левой границе);

– *center* (выравнивание по центру);

– *right* (выравнивание по правой границе).

3. *Border* устанавливает ширину рамки вокруг таблицы в пикселях (по умолчанию – 0).

4. *BorderColor* – цвет рамки таблицы.

5. *Cellspacing* устанавливает расстояние между рамками ячеек таблицы в пикселях (по умолчанию – 2).

6. *Cellpadding* устанавливает расстояние от границы ячейки до информации в ней (по умолчанию – 1).

7. *Background* – фоном всей таблицы будет картинка.

8. *BgColor* – цвет фона таблицы.

9. *Frame* указывает какие внешние стороны таблицы должны отображаться с рамкой. Атрибут может принимать следующие значения:

– *above* – только верхний край таблицы;

– *below* – только нижний край таблицы;

– *LHS* – только левый край таблицы;

– *RHS* – только правый край таблицы;

– *hsides* – верхний и нижний края таблицы;

– *vsides* – левый и правый края таблицы;

– *box* – все рамки таблицы; присваивается по умолчанию, если значение атрибута *border* больше нуля;

– *void* отключает вывод всех внешних рамок.

10. *Rules* управляет выводом внутренних рамок таблицы. Атрибут может принимать следующие значения:

– *all* – выводятся все рамки ячеек;

– *cols* – выводятся только разделительные линии столбцов;

– *groups* – выводятся только разделительные линии между группами ячеек, заданных с помощью элементов <THead>, <TBody>, <TFoot> и <ColGroup>;

– *rows* – отображаются линии только для строк;

– *none* отключает вывод всех внутренних линий.

Элемент <TR> определяет начало и конец строки таблицы, общие свойства ее ячеек.

Элемент <TR> может содержать следующие атрибуты:

1. *Align* устанавливает выравнивание текста в ячейках по горизонтали (значения см. выше для *align* таблицы).

2. *Valign* устанавливает выравнивание текста в ячейках по вертикали. Может принимать следующие значения:

– *top* – выравнивание по верхнему краю;

– *middle* – выравнивание по центру;

– *bottom* – выравнивание по нижнему краю.

3. *Bgcolor* – цвет фона ячеек строки.

4. *BorderColor* – цвет рамок ячеек таблицы.

Вместо элемента <TR> можно использовать элементы <THEAD> для строки, являющейся шапкой таблицы, и <TFOOT> для строки, являющейся подведением итогов таблицы. Эти элементы особенно полезны при динамическом формировании таблиц.

Пример использования элемента <THEAD>:

```
<Table border=1 BgColor=red >
    <THead Align=Center BGColor=Yellow>
        <TD>Столбец1</TD><TD>Столбец2</TD>
    </THead>
    <TR>
        <TD>Текст первого столбца</TD>
        <TD>Текст второго столбца</TD>
</TR>
</Table>
```

Элемент <TD> используется для указания ячейки в строке, ее индивидуальных свойств и может содержать следующие атрибуты:

1. *Align* устанавливает выравнивание текста в ячейке по горизонтали. Может принимать значения *left*, *right*, *center* (см. выше для *align* таблицы). Кроме этих значений может быть значения *justify* – выравнивание по ширине ячейки – и *char* – выравнивание по указанному символу (это значение поддерживается не во всех браузерах). Если используется *char*, то необходимо указать атрибут *char* и *charSet*.

2. *CharSet* указывает смещение при выравнивании по символу (см. значение *char* атрибута *align*: <TD Border=2 Align= Char Char=”.” CharSet=2>.

3. *Valign* устанавливает выравнивание текста в ячейке по вертикали. Значения те же, что и для *align*.

4. *Colspan* объединяет ячейки по горизонтали. Например *colspan*=2 означает, что ячейка простирается на две колонки.

5. *Rowspan* объединяет ячейки по вертикали. Например *rowspan*=2 означает, что ячейка простирается на две строки.

6. *Width* – ширина ячейки в пикселях или в процентах от ширины всей таблицы.

7. *Height* – высота ячейки в пикселях или в процентах от высоты всей таблицы.

8. *Bgcolor* – цвет фона ячейки.

9. *Background* – фоном ячейки будет картинка.

10. *BorderColor* – цвет рамки ячейки.

Вместо элемента <TD> можно использовать элемент <TH>. Этот элемент указывает на то, что данная ячейка является «шапкой» таблицы. Браузером элемент <TH> отображается несколько другим стилем, чем элемент <TD>, более жирным. Элемент <TH> содержит тот же набор атрибутов и свойств, что и элемент <TD>.

В том случае, когда группа ячеек таблицы, за исключением нижней и верхней строк, должна иметь одинаковые атрибуты, они объединяются элементом <TBODY>.

Пример использования элемента <TBODY>:

```
<Table>
   <TBody BgColor=red Align=Center>
      <TR>
         <TH>ЭВМ</TH><TH>Быстродействие</TH>
      </TR>
      <TR>
         <TH>Crey</TH><TH>1000000</TH>
         <TH>IBM 9000</TH><TH>10000</TH>
      </TR>
   </TBody>
</Table>
```

Как можно заметить, атрибут *border* содержится только в элементе <Table> и его действие распространяется на все ячейки таблицы. Что же делать, если, например, рамка должна быть не только во всей таблице, но и в ее шапке? В этом случае необходимо использовать атрибуты *frame* и *rules* с соответствующими значениями.

Таблицы часто используются для организации меню. Этим приемом пользуются в сайтах без применения фреймов. Вертикальное меню организуется разбиением всей страницы на две ячейки таблицы: левую для меню и правую для информации. Такой прием очень хорош, если вся информация в правой ячейке помещается на экран. Если же для просмотра будет использовать скроллинг, то вместе с ним в правой ячейке будет выполняться пролистывание меню, которое будет при этом исчезать из области экрана.

Также таблицы используются для записи текста в несколько колонок (в газетном стиле).

Как можно было заметить, в вышеприведенном материале отсутствовала информация о форматировании в едином стиле столбцов таблицы. Этот пробел возмещают <ColGroup> и <Col>. <ColGroup> используется для задания одинаковых свойств, включаемых в него по элементам <Col> столбцам. Элемент <Col> задает отличные от определяемых в <ColGroup> свойства.

Таблица может содержать один или несколько элементов <ColGroup>, которые необходимо размещать непосредственно за открывающим тегом <Table>. Внутри <ColGroup> может располагаться один или несколько непарных элементов <Col>.

Пример использования этих элементов приведен ниже:

```
<Table BgColor=yellow border=1>
   <ColGroup Span=2 Align=left></ColGroup>
   <ColGroup Style={background:lightblue}>
      <Col  Align=center>
      <Col Align=Right>
   </ColGroup>
   <tr>
      <td>желтый фон</td><td> желтый фон</td>
      <td>синий фон</td><td>синий фон</td>
      <td>желтый фон</td>
   </tr>
   <tr>
      <td>слева</td><td>слева</td>
      <td>центр</td><td>справа</td>
      <td>слева</td>
   </tr>
</Table>
```

Этот фрагмент документа в браузере будет отображаться следующим образом (рис. 6.1):

желтый фон	желтый фон	синий фон	синий фон	желтый фон
слева	слева	центр	справа	слева

Рис. 6.1

Несмотря на то, что третий и четвертый столбцы определены с синим фоном, рамка вокруг ячеек в этих местах все равно остается желтого цвета. Более того, если установить вывод без рамки, все равно между ячейками видна рамка желтого цвета (рис. 6.2):

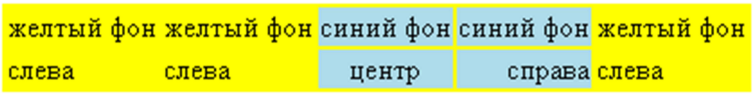

Рис. 6.2

Для устранения этого недостатка нужно использовать свойство *border-collapse* таблиц стилей:

```
<Table BgColor=yellow style='border-
        collapse:collapse;'>
```

Такое изменение в элементе <Table> устранит рамку между ячейками (рис. 6.3):

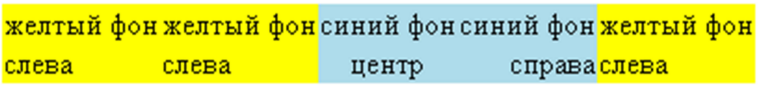

Рис. 6.3

Элемент <ColGroup> может содержать следующие атрибуты:

1. *Align* определяет горизонтальное выравнивание содержимого ячеек всей группы. Кроме стандартных, для этого значения используется еще значение *char* (это значение поддерживается не во всех браузерах), которое определяет выравнивание по символу, заданному в атрибуте *char*.

2. *Valign* определяет выравнивание по вертикали во всех ячейках группы.

3. *Char* указывает символ, по которому производится выравнивание. Этим символом может быть, например, десятичная точка, которую необходимо указывать в кавычках.

4. *Charoff* указывает в символах смещение влево или вправо данных, выравненных по символу в атрибуте *Char*.

5. *Width* задает ширину столбцов группы в пикселях или в процентах. В качестве значения допускается использование значения 0, которое означает минимальную для размещения данных ширину столбцов.

6. *Span* задает принятое по умолчанию количество столбцов в группе. По умолчанию это значение равно 1. Этот атрибут игнорируется, если в группе присутствует хотя бы один элемент <Col>.

Элемент <Col> может содержать атрибуты аналогичные <ColGroup>. В атрибуте *span* указывается число текущих столбцов с одинаковыми свойствами.

При создании таблиц чаще всего возникают трудности с расстановкой в нужных местах соответствующих рамок. В общем случае здесь не обойтись без использования таблиц стилей.

Примеры выполнения заданий

Пример 1. Создать HTML-документ, содержащий таблицу.

```
<HTML>
<HEAD></HEAD>
<BODY>
    <H1> Простейшая таблица</H1>
    <Table Border=1> <!--это начало таблицы -->
        <Caption Align=Center>У таблицы может
           быть заголовок</Caption>
        <TR> <!-- это начало первой строки -->
           <TD><!-- это начало первой ячейки
           (столбца) -->
           Первая строка, первая колонка
           </TD><!-- это конец первой ячейки
           (столбца) -->
           <TD><!-- это начала второй ячейки
           (столбца) -->
           Первая строка, вторая колонка
           </TD><!-- это конец второй ячейки
           (столбца) -->
        </TR><!-- это конец первой строки -->
        <TR> <!-- это начала второй строки -->
```

```
            <TD><!-- это начало первой ячейки
            (столбца) -->
            Вторая строка, первая колонка
            </TD><!-- это конец первой ячейки
            (столбца) -->
            <TD><!-- это начала второй ячейки
            (столбца) -->
            Вторая строка, вторая колонка
            </TD><!-- это конец второй ячейки
            (столбца) -->
        </TR><!-- это конец первой строки -->
    </Table> <!-- это конец таблицы -->
</BODY>
</HTML>
```

В окне браузера этот текст документа будет выглядеть следующим образом (рис. 6.4):

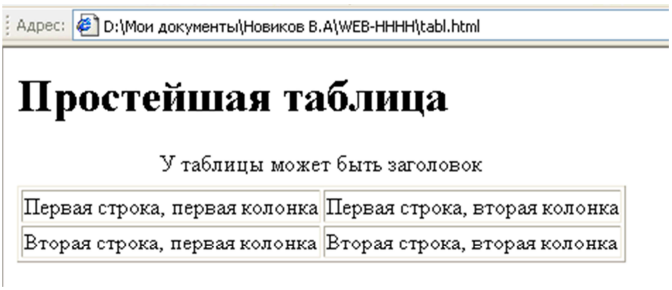

Рис. 6.4

Если ячейка должна быть пустой, то необходимо использовать неразрывный пробел :

```
<TD> </TD>
```

Пример 2. Использование атрибутов *frame* и *rules* с соответствующими значениями.

```
<HTML>
<HEAD> </HEAD>
<BODY>
    <Table Border=1 Frame=Box Rules=Groups>
        <ColGroup Span=1></ColGroup>
        <THead>
            <TD>Заголовок1</TD><TD>Заголовок2</TD>
        </THead>
        <TR>
            <TD width=80>0.5</TD><TD>1000</TD>
        </TR>
        <TR>
            <TD width=80>0.7</TD><TD>500</TD>
        </TR>
    </Table>
</BODY>
</HTML>
```

В браузере этот документ будет выглядеть следующим образом (рис. 6.5):

Заголовок1	Заголовок2
0.5	1000
0.7	500

Рис. 6.5

Пример 3. Организовать меню сайта с помощью таблицы.

```
<HTML>
<HEAD> </HEAD>
<BODY Bgcolor = "#COC8C8">
    <table border=3 height="100%" width="100%">
        <tr valign=center>
        <td  colspan=2>
        <center><strong><big>Меню
        сайта</big></strong></center><br><br>
```

```
        <table border=3 align=center>
        <tr>
        <td widht=80 align=center>
        <a target="_top"
        Href="фамилия.html">биография<Br></a>
        </td>
        </tr>
        </table> <br>
        <table border=3 align=center>
        <tr>
        <td widht=80 align=center>
        <a target="_top" Href="vuz.html"> мой ВУЗ
         <Br></a>
        </td>
        </tr>
        </table>
        </td>
        </tr>
    </table>
</BODY>
</HTML>
```

Задание. Измените текст документа, чтобы вокруг ссылки рамка не изображалась. Для этого удалите соответствующие элементы таблиц. Сохраните этот файл под именем menu1.html.

Пример 4. Вложенная таблица.

Отдельные ячейки таблицы могут содержать практически любые теги языка и данные, разрешенные в разделе <BODY>. В том числе внутри ячейки таблицы может быть размещена другая таблица. Такие таблицы называются *вложенными*. Правила их построения не отличаются от построения простых таблиц.

```
<HTML>
<HEAD> <TITLE>Вложенные таблицы в HTML</TITLE>
</HEAD>
<BODY>
 <table border="3">
  <tr>
```

```
    <td>Ячейка 1.1</td>
    <td colspan="3">Ячейки 1.2 - 1.4</td>
   </tr>
   <tr>
    <td>Ячейка 2.1</td>
    <td colspan="2" rowspan="2">Ячейки 2.2, 2.3
    и 3.2, 3.3</td>
    <td>Ячейка 2.4</td>
   </tr>
   <tr>
    <td>Ячейка 3.1</td>
    <td>
     <table border="2">
      <tr><td>Ячейка 1.1</td><td>Ячейка
1.2</td><td>Ячейка 1.3 </td>    </tr>
      <tr><td>Ячейка 2.1</td><td>Ячейка
2.2</td><td>Ячейка 2.3 </td>    </tr>
     </table>
    </td>
   </tr>
   <tr>
    <td colspan="4">Ячейки 4.1 - 4.4</td>
   </tr>
  </table>
  </BODY>
 </HTML>
```

В окне браузера таблица будет выглядеть следующим образом (рис. 6.6):

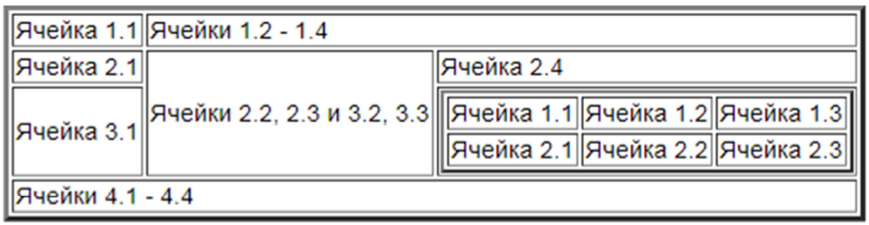

Рис. 6.6

Контрольные вопросы

1. Для чего используется элемент <Table>?
2. Перечислите атрибуты элемента <Table>.
3. Каково назначение атрибута *width* элемента <Table>?
4. Каково назначение атрибута *align* элемента <Table>? Какие значения он может принимать?
5. Каково назначение атрибутов *border*, *bordercolor*, *cellspacing*, *cellpadding*, *background*, *bgcolor* элемента <Table>?
6. Каково назначение атрибута *frame* элемента <Table>? Какие значения он может принимать?
7. Каково назначение атрибута *rules* элемента <Table>? Какие значения он может принимать?
8. Для чего используется элемент <TR>?
9. Перечислите атрибуты элемента <TR>.
10. Каково назначение атрибута *valign* элемента <TR>? Какие значения он может принимать?
11. Каково назначение атрибута *bgcolor* элемента <TR>?
12. Каково назначение атрибута *bordercolor* элемента <TR>?
13. Какие элементы можно использовать вместо элемента <TR>?
14. Для чего используются элементы <THEAD>, <TFOOT>, <TD>?
15. Перечислите атрибуты элемента<TD>.
16. Каково назначение атрибута *align* элемента <TD>? Какие значения он может принимать?
17. Каково назначение атрибутов *charset*, *colspan*, *rowspan*, *width*, *height*, *bgcolor*, *background*, *bordercolor* элемента <TD>?
18. Какой элемент можно использовать вместо элемента <TD>? На что указывает этот элемент?
19. В каком случае используется элемент <TBODY>?
20. Для чего используется свойство *float* листов стилей?
21. Для чего используются элементы <ColGroup> и <Col>?
22. Перечислите атрибуты элемента <ColGroup>.
23. Каково назначение атрибутов *align*, *valign*, *char*, *charoff*, *width*, *span* элемента <ColGroup>?
24. Перечислите атрибуты элемента <Col>.
25. Каково назначение атрибута *span* элемента<Col>?

Лабораторная работа № 7

СОЗДАНИЕ ФОРМ В HTML

Цель работы: ознакомиться с созданием форм в HTML, с элементами, которые могут находиться в форме и их добавлением.

Теоретические сведения

Элемент <FORM>

HTML-формы являются элементами управления, которые применяются для сбора информации от посетителей веб-сайта.

Веб-формы состоят из набора текстовых полей, кнопок, списков и других элементов управления, которые активизируются щелчком мыши. Технически формы передают данные от пользователя удаленному серверу.

Для получения и обработки данных форм используются языки веб-программирования такие, как PHP, Perl.

Определяются формы тэгом <FORM>. В одном документе может быть определено несколько форм для заполнения, но они не должны быть вложенными одна в другую.

Формат контейнера <FORM> следующий:

```
<FORM action="url" method="POST">...</FORM>
```

Тэг <FORM> имеет атрибуты, перечисленные в табл. 7.1.

Таблица 7.1

Атрибуты тега <FORM>

Атрибут	Назначение
name	Имя формы
action	Обязательный атрибут, указывающий URL обработчика формы на сервере, которому передаются данные. Представляет из себя файл (например, action.php), в котором описано, что нужно делать с данными формы. Если значение атрибута не будет указано, то после перезагрузки страницы элементы формы примут значения по умолчанию. В случае, если вся работа будет выполняться на стороне клиента сценариями JavaScript, то для атрибута action можно указать значение #.

	Также можно сделать, чтобы заполненная посетителем форма приходила вам на почту. Для этого нужно внести следующую запись: `<form action="mailto:адрес вашей электронной почты" enctype="text/plain"></form>`

method	Определяет метод, используемый для посылки содержания заполненной формы на сервер. Возможные варианты при этом следующие:
	1. Метод *get* передает данные на сервер через адресную строку браузера. При формировании запроса к серверу все переменные и их значения формируют последовательность вида:
	`www.anysite.ru/form.php?var1=1&var2=2.`
	Имена и значения переменных присоединяются к адресу сервера после знака ? и разделяются между собой знаком &. Все специальные символы и буквы, отличные от латинских, кодируются в формате %nn, пробел заменяется на +. Этот метод нужно использовать, если вы не передаете больших объемов информации. Если вместе с формой предполагается отправка какого-либо файла, этот метод не подойдет.
	2. Метод *post* применяется для пересылки данных больших объемов, а также конфиденциальной информации и паролей. Данные, отправляемые с помощью этого метода, не видны в заголовке URL, так как они содержатся в теле сообщения.
	`<form action="action.php" enctype="multipart/form-data" method="post"></form>`

enctype	Задает тип кодирования содержимого заполненной формы. Если значение атрибута не установлено, по умолчанию предполагается application/x-www-form-urlencoded
accept-charset	Значение атрибута представляет собой разделенный пробелами *список кодировок символов*, которые будут использоваться для отправки формы, например, <form accept-charset="ISO-8859-1">
target	Указывает окно, в которое будет направлена информация: _blank – новое окно, _self – тот же фрейм, _parent – родительский фрейм (если он существует, если нет, то в текущий), _top – окно верхнего уровня по отношению к данному фрейму. Если вызов происходит не из дочернего фрейма, то в тот же фрейм

Обработку данных, вводимых в формы, осуществляет программа, адрес которой указывается в атрибуте *action* тэга <FORM>. Но далеко не все сервера, предоставляющие место для личных сайтов, разрешают работу таким программам. Поэтому для демонстрации использования форм они будут отправляться не на сервер, а по электронной почте адресату, которого вы сами укажите. Для этого в атрибуте *action* прописывается макрокоманда отправки почтового сообщения:

```
<FORM ACTION="mailto:freeuser@mail.net">,
```

где freeuser@mail.net – адрес электронной почты получателя. Для отправки содержимого формы в данном случае используется почтовая программа отправителя.

Внутри контейнера <FORM> могут находиться любые теги, кроме другого контейнера <FORM>. Для создания полей для ввода данных внутри формы применяют теги:

– <INPUT>;
– <SELECT>;
– <TEXTAREA>.

Создание полей формы

Тег <INPUT> используется для задания простого элемента ввода. Атрибуты элемента отличаются в зависимости от типа поля. Могут быть использованы следующие атрибуты:

– *accept* определяет тип файла, разрешенных для отправки на сервер. Указывается только для `<input type="file">`. Возможные значения:

а) file_extension разрешает загрузку файлов с указанным расширением, например, accept=".gif", accept=".pdf", accept=".doc";

б) audio/* разрешает загрузку аудиофайлов;

в) video/* разрешает загрузку видеофайлов;

г) image/* разрешает загрузку изображений;

д) media_type указывает на медиатип загружаемых файлов;

– *alt* определяет *альтернативный текст* для изображений, указывается только для `<input type="image">`;

– *checked* проверяет, установлен ли флажок по умолчанию при загрузке страницы для полей типа `type="checkbox"` и `type="radio"`;

– *disabled* отключает возможность редактирования и копирования содержимого поля;

– *maxlength="n"* задает максимальное количество символов, вводимых в поле. Значение по умолчанию 524288 символов;

– *name="идентификатор"* – имя для создаваемого поля ввода, используемое для идентификации при обработке данных сервером;

– *readonly* не позволяет пользователю изменять значения элементов формы, выделение и копирование текста при этом доступно. Указывается без значения атрибута;

– *size="n"* задает физический размер поля ввода в символах, действует для элементов ввода текста или пароля;

– *value*="значение" используется для полей ввода текста *text* или пароля *password*, для задания начального содержания поля, а для *checkbox* или *radio* задает значение, когда элемент находится в отмеченном состоянии;

– *src* задает URL изображения, используемого в качестве кнопки отправки данных формы. Указывается только для поля `<input type="image">`;

– *type="параметр"* задает тип создаваемого элемента ввода; этот параметр может принимать одно из значений, указанных в табл. 7.2.

Таблица 7.2

Возможные значения атрибута *type* и их применение

Значение атрибута *text*	**Назначение**	**Пример**

text	Создает поле для ввода текста. Это значение используется по умолчанию	`<form method="get" action="/cgi-bin/handler.cgi">`
		`<b>Заполните Ваши данные:</b>`
		`Имя: <input type=text maxlength=25 size=20>`
		`Фамилия:<input type=text maxlength=25 size=20>`
		`</form>`

Продолжение табл. 7.2

Значение атрибута *type*	**Назначение**	**Пример**

password	Создает поле ввода пароля, аналогичное типу *text*, но при этом вводимые символы отображаются на экране звездочками:
	`<input type=password параметры>`
		`<form method= "get" action="/cgi-bin/handler.cgi">`
		`<b>Логин: </b><input type=text maxlength=25 size=20 name="text">`
		`<b>Пароль:</b> <input type=password maxlength=15 size=20 name="pass">`
		`</form>`

file	Создает поле для ввода имени локального файла, который необходимо куда-то послать. Сопровождается созданием кнопки «Обзор»	–
hidden	Создает скрытый элемент, не показываемый пользователю. Информация, хранящаяся в скрытом поле, всегда пересылается на сервер и не может быть изменена ни пользователем, ни браузером. Может применяться, например, при обработке нескольких форм, поочередно отправляемых пользователем и каким-либо образом связанных между собой	–

radio	Создает радиокнопку – принимающий положения «включено» и «выключено» элемент-переключатель в составе группы, из которой может быть выбран только один элемент	`<form action="/cgi-bin/handler.cgi"> <b>Какое время года Вы больше любите?)</b>`
		`<input type=radio name=answer value=a1>Весну`
		`<input type=radio name=answer value=a2>Лето`
		`<input type=radio name=answer value=a3>Осень`
		`<input type=radio name=answer value=a4>Зиму`
		`</form>`

Окончание табл. 7.2

Значение атрибута *type*	**Назначение**	**Пример**

checkbox	Создает переключатель-поле для установки флажка «включено» / «выключено»	`<form action="/cgi-bin/handler.cgi">`
		`<b>С какими операционными системамы вы знакомы?</b>`
		`<input type=checkbox name=option1 value=a1 checked>Windows 95/98`
		`<input type=checkbox name=option2 value=a2>Windows 2000`
		`<input type=checkbox name=option3 value=a3>Linux`
		`</form>`

submit	Создает кнопку передачи данных, действие которой сводится к отсылке содержимого заполненной формы на сервер	`<form action="/cgi-bin/handler.cgi">` `<p><input type=submit value="Отправить данные">`

image	Определяется элемент в виде графического изображения, действующий аналогично кнопке «submit»	`<form action="/cgi-bin/handler.cgi">`
		`<table align=center>`
		`<tr><td colspan=2>Введите ваше имя: <tr>`
		`<td><input type=text width=25></td>`
		`<td><input type=image src=../school/examples3/ser.gif width=90 height=68 border=0></td>`
		`</tr>`
		`</table>`
		`</form>`

reset	Создает кнопку, которая сбрасывает значения введенных полей в исходное состояние	`<form action="/cgi-bin/handler.cgi">`
		`<input type=text width=20> <input type=reset value="Сбросить данные">`
		`</form>`

Раскрывающийся список

Раскрывающиеся списки создаются при помощи элемента <SELECT>...</SELECT>. Списки дают возможность расположить большое количество пунктов компактно и выбрать одно или несколько значений из предложенного множества. «По умолчанию» в поле списка отображается его первый элемент.

Атрибуты тега <SELECT> следующие:

1. *Autofocus* устанавливает автоматический фокус на элементе при загрузке страницы.

2. *Disabled* отключает раскрывающийся список.

3. *Form* определяет форму, которой принадлежит данный список. В качестве значения атрибута указывается идентификатор формы.

4. *Multiple* дает возможность выбора одного или нескольких пунктов (для этого при выборе нужно нажать и удерживать клавишу Ctrl).

5. *Name* определяет имя для выпадающего списка. Значение атрибута содержит название, отражающее тематику.

6. *Required* выводит сообщение о том, что пользователь должен выбрать значение из раскрывающегося списка перед отправкой формы.

7. *Size* задает количество одновременно видимых на экране элементов списка. Если количество элементов списка превышает установленное, появляется полоса прокрутки. Значение атрибута задается целым положительным числом.

Для добавления в список пунктов используются элементы <OPTION>...</OPTION>, которые располагаются внутри <SELECT>.

Атрибуты тега <OPTION>:

1. *Disabled* делает недоступным для выбора элемент списка.

2. *Label* задает укороченную версию для элемента, которая будет отражаться в выпадающем списке. Значение атрибута содержит текст, описывающий соответствующий пункт выпадающего списка.

3. *Selected* устанавливает выбранный элемент списка по умолчанию при загрузке страницы браузером.

4. *Value* указывает значение, которое будет отправлено вместе с формой на сервер.

Для систематизации списков применяется элемент <OPTGROUP>...</OPTGROUP>, который создает заголовки в списках.

Атрибуты тега <OPTGROUP>:

1. *Disabled* отключает данную группу элементов списка для выбора.

2. *Label* задает заголовок для группы элементов выпадающего списка. Значение атрибута содержит текст, недоступный для выбора, который будет располагаться над соответствующим списком. Текст выделяется в браузере жирным начертанием.

Для списков возможно изменить шрифт, его размер, цвет и другие свойства, а также добавить границы, цвет фона и фоновое изображение.

Пример:

```
<form action="/cgi-bin/handler.cgi">
    <b>Какое время года Вы больше любите? :-)</b>
    <p align=center>
    <select name= время года>
        <option>Весну</option>
        <option>Лето</option>
        <option>Осень</option>
        <option>Зиму</option>
    </select>
</form>
```

Текстовые поля ввода

Контейнер <TEXTAREA> может быть использован для расположения многострокового поля ввода с необязательным содержимым в форме.

Основными атрибутами тега <TEXTAREA> являются:

1. *Autofocus* устанавливает фокус на нужном начальном текстовом поле автоматически.

2. *Cols* устанавливает ширину через количество символов. Если пользователь вводит больше текста, появляется полоса прокрутки.

3. *Disabled* отключает возможность редактирования и копирования содержимого поля.

4. *Form* – значение атрибута должно быть равно значению *id* элемента <form> в этом же документе. Определяет одну или несколько форм, которым принадлежит данное текстовое поле.

5. Maxlength – атрибут, задаеющий максимальное число символов для ввода в поле.

6. *Name* задает имя текстового поля.

7. *Placeholder* определяет короткую текстовую подсказку, которая описывает ожидаемое вводимое значение.

8. *Readonly* отключает возможность редактирования содержимого поля.

9. *Required* выводит сообщение о том, что данное поле является обязательным для заполнения.

10. *Rows* указывает число, которое означает, сколько строк должно отображаться в текстовой области.

11. *Wrap* определяет, должен ли текст сохранять переносы строк при отправке формы. Значение *hard* сохраняет перенос, а значение *soft* – нет. Если используется *hard*, то должно указываться значение атрибута *cols*.

Пример:

```
<form method="get" action="/cgi-bin/handler.cgi">
<b>Введите ваш отзыв: </b>
<textarea rows=10 cols=45></textarea>
</form>
```

Группировка элементов формы

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

Браузеры для повышения наглядности отображают результат использования тега <FIELDSET> в виде рамки, вид которой зависит от операционной системы, а также от используемого браузера.

Синтаксис:

```
<form>
  <fieldset>...</fieldset>
</form>
```

Атрибуты тега <FIELDSET>:

1. *Disabled* делает группу связанных элементов формы, находящихся внутри контейнера <fieldset>, недоступными для заполнения

и редактирования. Используется для ограничения доступа к некоторым полям формы, содержащим ранее введенные данные. Атрибут используется без указания значения: <fieldset disabled>..

2. *Form* связывает группу с формой. Значение атрибута должно быть равно id элемента <form> в этом же документе. Указывает на одну или несколько форм, к которым принадлежит данная группа элементов. На данный момент атрибут не поддерживается ни одним браузером.

3. *Name* определяет имя, которое будет использоваться для ссылки на элементы в JavaScript или на данные формы после заполнения и отправки формы. Является аналогом атрибута *id*.

Также для этого тега доступны универсальные атрибуты и события.

Каждой группе элементов можно присвоить название с помощью элемента <LEGEND>, который идет сразу за тегом <FIELDSET>. Название группы проявляется слева в верхней границе <FIELDSET>.

Пример:

```
<form action=" ">
  <fieldset>
   <legend>Работа со временем</legend>
     <input type="checkbox"> создание пунктуаль-
         ности (никогда не  будете никуда опаз-
         дывать);<br>
     <input type="checkbox"> излечение от пунк
         туальности (никогда
      никуда не будете торопиться);<br>
     <input type="checkbox"> изменение восприя
         тия времени и часов.
      <p><input type="submit"></p>
  </fieldset>
 </form>
```

Подписи к полям формы

Подписи к элементам формы создаются с помощью элемента <LABEL>...</LABEL>.

Атрибут *for* тега <LABEL> определяет, к какому полю формы привязан данный элемент. Можно создавать поясняющие подписи к следующим элементам формы: <input>, <textarea>, <select>. Значение атрибута содержит идентификатор поля формы.

Существует два способа группировки надписи и поля. Если поле находится внутри элемента <LABEL>, то атрибут *for* указывать не нужно.

```
<!-- с указанием атрибута for -->
<label for="comments">Когда вы последний раз летали на самолете?</label>
<textarea id="comments"></textarea>

<!-- без атрибута for -->
<p><label>Кошка<input id="cat"
type="checkbox"></label></p>
```

Кнопки

Элемент <BUTTON>...</BUTTON> создает кликабельные кнопки. В отличие от кнопок, созданных <input> (`<input type="submit"></input>`, `<input type="image">`, `<input type="reset">`, `<input type="button">`), внутрь элемента <BUTTON> можно поместить контент текст или изображение.

Для корректного отображения элемента <BUTTON> разными браузерами нужно указывать атрибут *type*, например, `<button type="submit"></button>`.

Кнопки позволяют пользователям передавать данные в форму, очищать ее содержимое или предпринимать какие-либо другие действия. Можно создавать границы, изменять фон и выравнивать текст на кнопке.

Атрибуты тега <BUTTON> представлены в виде табл. 7.3.

Таблица 7.3

Атрибуты тега <BUTTON>

Атрибут	Назначение
autofocus	Устанавливает фокус на кнопке при загрузке страницы
disabled	Отключает кнопку, делая ее некликабельной
form	Указывает на одну или несколько форм, которым принадлежит данная кнопка. Значение атрибута – идентификатор соответствующей формы

Продолжение табл. 7.3

Атрибут	Назначение
formaction	Значение атрибута содержит URL-адрес обработчика отправляемых данных формы. Только для кнопки типа `type="submit"`. Переопределяет значение атрибута *action*, указанного для элемента <form>
formenctype	Задает тип кодировки данных формы перед отправкой на сервер при нажатии на кнопки типа `type="submit"`. Переопределяет значение атрибута *enctype*, указанного для элемента <form>. Возможные значения: – `application/x-www-form-urlencoded` – значение по умолчанию. Все символы перед отправкой будут закодированы; – `multipart/form-data` – символы не кодируются. Используется в случае, когда с помощью формы загружаются файлы; – `text/plain` – символы не кодируются, а пробелы заменяются на символ +
formmethod	Атрибут определяет метод, который браузер будет использовать для отправки формы. Переопределяет значение атрибута *method*, указанного для элемента <form>. Указывается только для кнопок типа `type="submit"`. Возможные значения: – *get* – данные из формы (имя / значение) добавляются в URL-адрес и отправляются на сервер. Данный способ имеет ограничения по размеру отправляемых данных и не подходит для отправки паролей и конфиденциальной информации; – *post* – данные из формы добавляются в виде HTTP-запроса. Метод является более надежным и безопасным, чем *get* и не имеет ограничений по размеру
formnovalidate	Атрибут задает, что данные формы не должны проверяться при отправке. Указывается только для кнопок типа `type="submit"`
formtarget	Атрибут задает, в каком окне выводить результат после отправки формы. Указывается только для кнопок типа type="submit". Переопределяет значение атрибута *target*, указанного для элемента <form>. Принимаемые значения:

Окончание табл. 7.3

Атрибут	Назначение
	– _blank (загружает ответ в новое окно / вкладку); – _self (загружает ответ в то же окно (значение по умолчанию)); – _parent (загружает ответ в родительский фрейм); – _top (загружает ответ во весь экран); – framename (загружает ответ во фрейм с указанным именем)
name	Задает имя кнопки; значение атрибута – текст. Используется для ссылки на данные формы после того, как форма была отправлена, или на данную кнопку (кнопки) в JavaScript
type	Определяет тип кнопки. Возможные значения: – button – кликабельная кнопка; – reset – кнопка сброса, возвращает первоначальное значение; – submit – кнопка для отправки данных формы
value	Задает значение по умолчанию при нажатии на кнопку отправки

Примеры выполнения заданий

Пример 1. Создать HTML-документ, содержащий форму.

```
<HTML>
<HEAD> <Title> Формы в  HTML </Title> </HEAD>
<BODY>
<div align="center">
  <form name="form1" action="mailto:??????">
    Ваше имя: <input type="text" name="firstname"
   value="Тут Ваше имя">
    <br>
    Ваш пароль: <input type="password" name=
   "pass" value="Тут Ваш пароль">
    <br>
    Выберите вариант:
    <select name="choice">
```

```
        <option value="1">Вариант 1
        <option value="2">Вариант 2
        <option value="3">Вариант 3
      </select>
      <br>
      Напишите что-нибудь:
      <br>
      <textarea name="message" rows="10" cols=
     "15">Сообщение</textarea>
      <br>
      Выберите что-нибудь одно:
      <input type="radio" name="choiceradio" value
     ="1">Вариант 1
      <input type="radio" name="choiceradio" value
     ="2">Вариант 2
      <input type="radio" name="choiceradio" value
     ="3">Вариант 3
      <br>
      Вы согласны с нашими правилами:
      <input type="checkbox" name="terms" value=
     "yes">
      <br>
      Выберите файл для загрузки:
      <input type="file" name="path">
      <br>
      <input type="button" name="start" value=
     "Начать">
      <br>
      <input type="submit" value="Отправить">
    </form>
  </div>
  </BODY>
  </HTML>
```

Пример 2. Варианты использования тэга <SELECT>, создающего списки в форме.

```
<HEAD>
<TITLE>Создание списка</TITLE>
</HEAD>
<BODY>
<TABLE>
<TR>
 <TD>Выберите операционную систему, которую вы
   используете:</TD>
 <TD>Выберите операционную систему, которую вы
   используете:</TD>
 <TD>Выберите операционную систему, которую вы
   используете:</TD>
</TR>
<TR>   <TD>    <FORM>
   <SELECT>
    <OPTION>Windows XP
    <OPTION>Windows 7
    <OPTION>Windows 8
    <OPTION>Linux
    <OPTION>Unix
   </SELECT>
  </FORM>
 </TD>
 <TD>
  <FORM>
   <SELECT size=3>
    <OPTION>Windows XP
    <OPTION>Windows 7
    <OPTION>Windows 8
    <OPTION>Linux
    <OPTION>Unix
   </SELECT>
  </FORM>
 </TD>
 <TD>    <FORM>
   <SELECT multiple>
    <OPTION>Windows XP
    <OPTION>Windows 7
```

```
      <OPTION>Windows 8
      <OPTION>Linux
      <OPTION>Unix
     </SELECT>
    </ FORM>
  </TD>
</TR>
</TABLE> </BODY>  </HTML>
```

Контрольные вопросы

1. Для чего предназначены формы?
2. Каким тегом определяются формы?
3. Напишите формат контейнера <FORM>.
4. Какие атрибуты имеет тег <FORM>?
5. Для чего предназначен атрибут *action*?
6. Для чего предназначен атрибут *method*?
7. Какие значения может принимать атрибут *method*? В каком случае используется то или иное значение?
8. Для чего предназначен атрибут *enctype*?
9. Какие теги применяются для создания полей для ввода данных внутри формы?
10. Каково назначение тега <INPUT>?
11. Перечислите атрибуты тега <INPUT>. Каково их назначение?
12. Какие значения может принимать атрибут *type* тега<INPUT>?
13. Какой элемент будет добавлен в форму, если атрибут *type* тега <INPUT> имеет значение *text*, *password*, *file*, *hidden*, *radio*, *checkbox*, *submit*, *image*, *reset*?
14. Каково назначение тега <SELECT>?
15. Перечислите атрибуты тега <SELECT>. Каково их назначение?
16. Каково назначение тега <OPTION>?
17. Перечислите атрибуты тега <OPTION>. Каково их назначение?
18. Каково назначение тега <OPTGROUP>?
19. Перечислите атрибуты тега <OPTGROUP>. Каково их назначение?
20. Каково назначение тега <TEXTAREA>?
21. Перечислите атрибуты тега <TEXTAREA>. Каково их назначение?

22. Для чего предназначен элемент <FIELDSET>?

23. Перечислите атрибуты тега <FIELDSET>. Каково их назначение?

24. С помощью какого элемента создаются подписи к элементам формы?

25. Назовите элемент тега <LABEL>. Для чего он предназначен?

26. Как задать кнопку в форме?

27. Перечислите атрибуты тега <BUTTON>. Каково их назначение?

Лабораторная работа № 8

ФРЕЙМЫ В HTML

Цель работы: получить знания о видах фреймов и способах их создания. Выработать практические умения по разделению страницы на несколько фреймов, добавлению плавающих фреймов, отображению информации в области фрейма.

Теоретические сведения

Окно браузера может быть разбито на произвольное число независимых подокон, в каждом из которых могут отображаться разные HTML страницы. Такие подокна называются *фреймами*. Разбиение окна браузера на фреймы осуществляется парным дескриптором <FRAMESET>...</FRAMESET>. Каждый созданный фрейм посредством этого же дескриптора может быть также разделен. Деление окна браузера на фреймы не является телом HTML документа, поэтому дескриптор <BODY>...</BODY> не используется.

Дескриптор <FRAMESET> имеет два атрибута: *cols* и *rows*. Они позволяют разделять окно на вертикальные (*cols* – колонки) либо горизонтальные полосы (*rows* – строки). Размер полосы указывается в процентах от вертикального (горизонтального) размера разделяемой области или в экранных пикселях. Каждому значению *cols* и *rows* в порядке их следования должен соответствовать либо <FRAME>, либо новый <FRAMESET>.

Показ HTML страниц во фреймах

При описании фрейма дескриптором <FRAME> нужно указать атрибут *src*, задающий имя файла, который будет открыт в этой области, и атрибут *name*, определяющий имя фрейма, под которым он будет известен в пределах создаваемого Web-сайта. Имя обеспечивает возможность кликать по гиперссылкам в одном фрейме, а просматривать файлы, открывающиеся по ссылкам, – в другом. Это прекрасная возможность создавать системы меню для сайтов.

Атрибуты дескриптора <FRAME>:

– *frameborder* позволяет отобразить или скрыть границу фрейма; чтобы убрать границу между двумя фреймами, нужно скрыть ее в каждом из них. Если атрибут не указан, граница отображается. Frameborder=0 скрывает (Internet Explorer отображает тонкую границу), а frameborder=1 отображает границу;

– *noresize* исключает возможность изменения размеров фрейма мышью (если атрибут не указан, такая возможность есть);

– *scroling* отображает или запрещает отображение полос прокрутки. Если информация не умещается во фрейме, то на правой и нижней границе фрейма могут появляться полосы прокрутки. Scroling=auto (используется по умолчанию) – если информация помещается во фрейме, полосы прокрутки не отображаются, если не умещается, отображаются; scroling=yes – полосы прокрутки всегда отображаются; scroling=no – полосы прокрутки не отображаются;

– *marginwidth* определяет отступ во фрейме от левого края до содержимого;

– *marginheight* определяет отступ во фрейме от верхнего края до содержимого;

– *bordercolor* указывает цвет границы фрейма (ширина границы задается параметром *framespacing*).

Плавающие фреймы в HTML

При создании многооконной Web-страницы, помимо применения тэга <FRAMESET>...</FRAMESET>, также могут быть использованы плавающие фреймы. В этом случае в HTML-документе вместо тэга <FRAMESET>...</FRAMESET> используется, как и во всех остальных Web-страницах, тэг <BODY>...<BODY>.

Плавающий фрейм, подобно изображению, будет располагаться на экране в том месте, которое соответствует расположению тэга с его описанием в HTML-документе. Для задания плавающего фрейма используется тэг-контейнер <IFRAME>...</IFRAME>. Плавающий фрейм объединяет свойства и фрейма, и изображения. Одна часть его параметров (*marginheight*, *marginwidth*, *name*, *scroling*, *src*, *bordercolor*) свойственна фреймам, другая (*align*, *height*, *width*, *hspace*, *vspace*) – изображениям.

Примеры выполнения заданий

Пример 1. Создать HTML-документ, содержащий 3 горизонтальных фрейма.

```
<HTML>
<HEAD>
    <Title> Три горизонтальных фрейма </Title>
</HEAD>
    <frameset rows ="50%, 40,*">
        <frame> <!-- Фрейм, соответсвующий 50%
            --!>
        <frame> <!-- Фрейм, соответсвующий 40
            пикселям --!>
        <frame> <!-- Фрейм, соответсвующий звез-
            дочке --!>
    </frameset>
</HTML>
```

Задание:

1. Скройте границы между фреймами.
2. Создайте HTML-документ, содержащий 3 вертикальные фрейма.

Пример 2. Создать HTML-документ pr3.html, отображающий содержимое 3 файлов: title.html (с текстом «Заголовок»); menu.html (с текстом «Меню»); main.html (с текстом «Информация») в 3 различных фреймах (рис. 8.1).

Рис. 8.1. Вид документа, содержащего 3 произвольно расположенных фрейма, отображающих информацию из 3 различных файлов

А)
```
<HTML>
  <HEAD>
  <Title> Пример структуры сайта </Title>
  </HEAD>
  <frameset rows="100,*">
   <frame name="Title" src="title.html">
   <frameset cols="25%,*">
   <frame name="menu" src="menu.html">
   <frame name="info" src="main.html">
   </frameset>
   </frameset>
  </HTML>
```

Б)
```
<HTML>
  <HEAD>
      <title>Фреймы в html</title>
  </HEAD>
  <frameset rows="100,*">
  <frame  name="Title"  src="title.html"  margin-
width="50"  marginheight="20" noresize>
      <frameset cols =" 25%,*">
```

```
      <frame  src="menu.html"  marginwidth="100"
scrolling="no">
      <frame src="main.html" name="info" margin-
width="200" marginheight="100">
      </frameset>
  </frameset>
  </html>
```

Пример 3. Создать HTML-документ, содержащий плавающий фрейм (рис. 8.2).

Указания:

1. Добавить в папку необходимые файлы из предыдущих лабораторных работ (задание № 1 (лабораторная работа № 4), задание № 1 (лабораторная работа № 6)).

2. Сделать так, чтобы выбираемый во фрейме *Menu* файл задания отображался во фрейме *Info*.

3. Пока не выбран ни один пункт меню, во фрейме *Info* отображается плавающий фрейм, содержащий файл Zoo1.html.

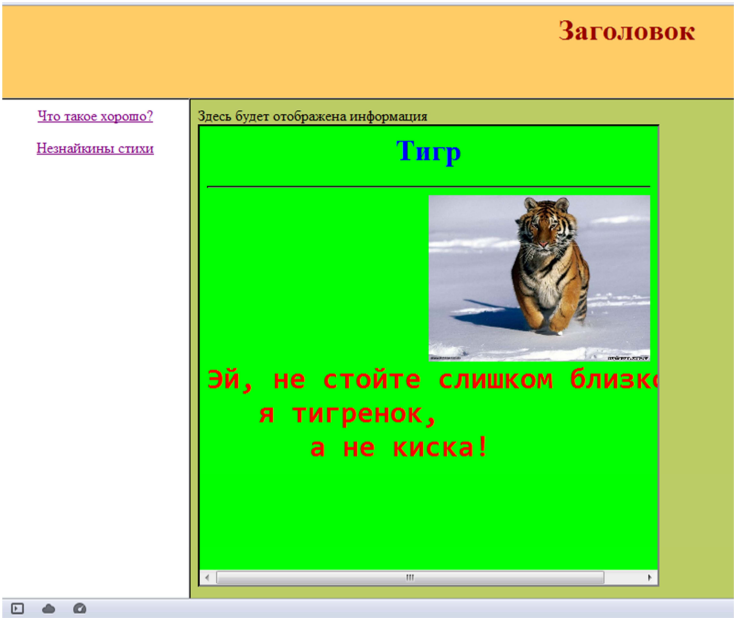

Рис. 8.2. Документ, содержащий плавающий фрейм.

```
<HTML>
    <HEAD>
        <title>Меню сайта</title>
    </HEAD>
    <body bgcolor="#BBCC66">
 Здесь будет отображена информация
<BR>
<IFRAME NAME="ff" SRC="zoo1.html" HEIGHT=500
WIDTH=500>
 </IFRAME>
    </BODY>
</HTML>
```

Контрольные вопросы

1. Что такое фрейм?
2. Как осуществляется разбиение окна браузера на фреймы?
3. Какие атрибуты имеет дескриптор <FRAMESET>?
4. Для чего предназначен атрибут *cols*?
5. Для чего предназначен атрибут *rows*?
6. В каких единицах указываются размеры полос?
7. После разделения на фреймы как должен быть определен каждый фрейм?
8. Перечислите атрибуты дескриптора <FRAME>.
9. Для чего предназначены атрибуты *src*, *name*, *frameborder*, *noresize*, *scroling* дескриптора <FRAME>?
10. Какие значения может принимать атрибут *scroling*?
11. Для чего предназначены атрибуты *marginwidth*, *marginheight*, *bordercolor* дескриптора <FRAME>?
12. Какой тег используется для задания плавающего фрейма?
13. В чем особенность плавающих фреймов?

Лабораторная работа № 9

СОЗДАНИЕ И ВНЕДРЕНИЕ В WEB-СТРАНИЦЫ ТАБЛИЦ СТИЛЕЙ

Цель работы: формирование практических умений использования каскадных таблиц стилей для оформления HTML-документов.

Теоретические сведения

CSS (каскадные таблицы стилей) управляют внешним видом документа. Использование CSS позволяет отделить содержание документа от его оформления, т. е. сначала определяется, как будет выглядеть тот или иной элемент документа (например, заголовок, абзац и т. д.), а затем вводится его содержимое.

Существуют четыре способа применения таблиц стилей к документу:

1. *Связывание* позволяет использовать одну таблицу стилей для форматирования многих страниц HTML.
2. *Внедрение* позволяет задавать все правила таблицы стилей непосредственно в самом документе.
3. *Импортирование* позволяет встраивать в документ таблицу стилей, расположенную на сервере.
4. *Встраивание* в тэги документа позволяет изменять форматирование конкретных элементов страницы.

Встраивание CSS в HTML

CSS позволяют назначить собственный стиль визуального представления любому тегу HTML, в том числе <BODY>. Если стиль задан для тега <BODY>, он наследуется всеми элементами (абзацами, заголовками и т. д.), в случае отсутствия у них собственных стилей.

Пример. Необходимо, чтобы все абзацы отображались шрифтом – Times New Roman, размером – 12 пунктов, цветом – зеленым. Для этого следует указать атрибут *style* тега <BODY>, присвоив ему соответствующее значение:

```
<body style="font-family:'Times New Roman'; font-size:12pt; color:green">
```

В примере используется встраивание стиля непосредственно в тег документа – так называемый *inline*-стиль.

Внедрение таблицы стилей

Для применения одинакового форматирования к нескольким одинаковым элементам страницы необходимо создать в заголовке страницы (в любом месте между тегами <head> и </head>) внедренную таблицу стилей, в которой задаются требуемые правила оформления. Для этого создается тег-контейнер таблицы стилей, начинающийся открывающим тегом <style> и заканчивающийся закрывающим тегом </style>. Внутри этого тега-контейнера можно задать любое количество правил CSS.

Пример. Необходимо, чтобы все абзацы на странице выглядели, как в предыдущем примере, все заголовки первого уровня отображались шрифтом Arial синего цвета полужирного начертания размером 16 пунктов, а все заголовки второго уровня – шрифтом Helvetica размером 14 пунктов полужирного курсивного начертания желтого цвета.

```
<head>
...
<style>
<!--body {
font-family:'Times New Roman';
font-size: 12pt;
color:green;
}
h1 {
font-family:Arial;
font-size:16pt;
color:blue;
font-weight:bold;
}
h2 {
font-family:Helvetica;
font-size:14pt;
color:yellow;
font-weight:bold;
```

```
font-style:italic;
}
-->
</style>
...
</head>
```

Связывание таблицы стилей

Связывание позволяет хранить таблицу стилей в отдельном файле и присоединять ее к документам с помощью тэга <link>, задаваемого в разделе <head>:

```
<link rel="stylesheet" type="text/css"
        href="mystyles.css"/>
```

В этой строке указывается, что связываемый файл является таблицей стилей (`rel="stylesheet"`), формат этого файла – .css1 (`type="text/css"`) и находится он в той же директории, что и файл html под именем «mystyles.css» (`href="mystyles.css"`).

Импорт таблицы стилей

Для *импорта* таблиц стилей в HTML-файл используется ключевое слово *@import*. В данном случае импортируется только содержимое текстового файла, поэтому для того, чтобы этот текст интерпретировался как таблицы стилей, @import нужно поместить в контейнер <STYLE>.

Например:

```
<STYLE TYPE="text/css">
@import url(http://www.myserver.com/style.css);
</STYLE>
```

Приоритеты использования таблиц стилей

Браузер расставляет приоритеты таблиц стилей следующим образом:

1. Встроенные (inline-) стили (встроенные с помощью атрибута *style* непосредственно в теги документа) – наивысший приоритет.

Будут применены браузером в любом случае, даже если возникает конфликт с внедренными или внешними стилями.

2. Внедренные стили (перечисленные в теге-контейнере <style> в заголовке документа) – чуть меньший приоритет; будут применяться всегда, кроме случаев возникновения конфликта с inline-стилями (будут применены inline-стили).

3. Импортированные стили (стили внешнего файла .css, связанные с документом с помощью свойства @import в теге-контейнере <style>) будут применяться в тех случаях, когда отсутствуют аналогичные правила CSS среди встроенных и внедренных стилей.

4. Связанные стили (стили, присоединенные к html-файлу посредством тега <link>) – наименьший приоритет, будут применены только после того, как браузер убедится в отсутствии аналогичных правил во всех остальных типах стилей.

Каждое определение стилей называется *правилом* (rule).

Формат правила CSS следующий:

```
селектор{свойство1:значение1; свойство2: зна-
чение2;…}
```

Например:

```
H1 {color:blue}
```

В документе, для которого определено данное правило, все заголовки *H1* будут выделяться синим цветом.

Если заменить это правило на

```
H1,H2,H3 {color:blue},
```

синим цветом будут выделяться заголовки первого, второго и третьего уровней.

Класс определяет разновидность стиля, к которому можно обращаться в определенном теге, используя атрибут *class*.

Например, можно определить три разновидности стиля *H1* и затем использовать каждый из них в соответствующем контексте:

```
H1.blue {color: blue}
H1.red {color: red}
H1.black {color: black}
```

При добавлении тега <H1> в HTML-документ необходимо определить атрибут *class*, чтобы указать, какой именно стиль будет использоваться:

```
<H1 CLASS=red>Красный заголовок</H1>
```

Можно создавать класс, не связанный с определенным тегом. Например, если задать стилевое правило следующим образом:

```
.bold_and_italic{font-style:italic;font-
weight:bold}
```

и присвоить атрибуту *class* некоторого тега значение bold_and_italic, содержимое данного тега будет отображаться жирным шрифтом с курсивным начертанием.

Использование *псевдоклассов* позволяет указать внешний вид HTML-элемента в определенный момент времени. Синтаксис:

```
Селектор:псевдокласс {свойство:значение}
```

В CSS определены псевдоклассы для гиперссылок. Например:

```
/*непосещенная гиперссылка*/
A:link {color:blue}

/*активная гиперссылка*/
A:active {color:red}

/*посещенная гиперссылка*/
A:visited {color:yellow}

/*свойства гиперссылки при наведении курсора*/
A:hover {color:green}
```

Таким образом, непосещенная гиперссылка будет выделена синим цветом, активная – красным, посещенная – желтым, а при наведении курсора мыши цвет ссылки изменится на зеленый.

Основные свойства CSS приведены в приложении.

Примеры выполнения заданий

Пример 1. Установка цвета фона страницы

```
<HTML>
<HEAD>
    <style type="text/css">
       body
       {
       background-color:#b0c48e;
       }
    </style>
</HEAD>
<BODY>
    <h1>Моя CSS веб страница!</h1>
    <p>Привет мир! Это Uroki-CSS.ru пример.</p>
</BODY>
</HTML>
```

Пример 2. Установка изображения в качестве фона страницы.

```
<HTML>
<HEAD>
    <style type="text/css">
       body {background-image:url('paper.gif');}
    </style>
</HEAD>
<BODY>
    <h1>Привет Мир!</h1>
</BODY>
</HTML>
```

Пример 3. Установка цвета текста различных элементов.

```
<HTML>
<HEAD>
    <style type="text/css">
        body {color:red;}
        h1 {color:#00ff00;}
        p.ex {color:rgb(0,0,255);}
```

```
    </style>
</HEAD>
<BODY>
    <h1>Это заголовок 1</h1>
    <p>Это обычный параграф. Заметьте, что
    текст - красный. Цвет текста по умолчанию
    для страницы определяется в селекторе
    body.</p>
    <p class="ex">Это параграф с классом
    class="ex". Текст - голубой.</p>
</BODY>
</HTML>
```

Пример 4. Выравнивание текста.

```
<HTML>
<HEAD>
    <style type="text/css">
        h1 {text-align:center;}
        p.date {text-align:right;}
        p.main {text-align:justify;}
    </style>
</HEAD>
<BODY>
    <h1>CSS пример выравнивания текста</h1>
    <p class="date">Май, 2009</p>
    <p class="main">В делах нет лучшего сове-
    та, чем быть умеренным. Быть умеренным -
    значит предвосхищать. Предвосхищать - зна-
    чит быть подготовленным и сильным. Быть
    подготовленным и сильным - значит быть
    всегда преуспевающим. Быть всегда преуспе-
    вающим - значит иметь бесконечные возмож-
    ности. (Дао-де-дзин, глава 59)</p>
    <p><b>Замечание:</b> Значение "justify"
    выравнивает строки как в газете или журна-
    ле.</p>
</BODY>
</HTML>
```

Пример 5. Установка размера шрифта.

```
<HTML>
<HEAD>
    <style type="text/css">
        h1 {font-size:250%;}
        h2 {font-size:200%;}
        p {font-size:100%;}
    </style>
</HEAD>
<BODY>
        <h1>Это заголовок 1</h1>
        <h2>Это заголовок 2</h2>
        <p>Это параграф.</p>
</BODY>
</HTML>
```

Пример 6. Установка различных цветов для посещенных / непосещенных ссылок.

```
<HTML>
<HEAD>
    <style type="text/css">
        a:link {color:#FF0000;} /*непосещенная
        ссылка */
        a:visited {color:#00FF00;} /* посещен-
        ная ссылка */
        a:hover {color:#FF00FF;} /* курсор мыши
        над ссылкой */
        a:active {color:#0000FF;} /* выбранная
        ссылка */
    </style>
</HEAD>
<BODY>
    <p><b><a href="index.php" target="_blank">Это
       ссылка</a></b></p>
    <p><b>Замечание:</b> a:hover ДОЛЖНО идти
       после a:link и a:visited в CSS опреде-
       лении для того,
```

```
    чтобы все работало.</p>
    <p><b>Замечание:</b> a:active ДОЛЖНО идти
       после a:hover в CSS определении, чтобы
    все работало.</p>
</BODY>
</HTML>
```

Пример 7. Установка высоты изображения в пикселях.

```
<HTML>
<HEAD>
    <style type="text/css">
        img.normal
        {
        height:auto;
        }
        img.big
        {
        height:120px;
        }
        p.ex
        {
        height:100px;
        width:100px;
        }
    </style>
</HEAD>
<BODY>
    <img class="normal" src="images/logocss.gif"
       width="95" height="84" /><br />
    <img  class="big"  src="images/logocss.gif"
       width="95" height="84" />
    <p  class="ex">Высота  и  ширина  параграфа
       100px.</p>
    <p>Это некоторый текст в параграфе. Это
       некоторый текст в параграфе.
    Это некоторый текст в параграфе. Это неко-
       торый текст в параграфе.
```

```
    Это некоторый текст в параграфе. Это неко
       торый текст в параграфе.</p>
</BODY>
</HTML>
```

Пример 8. Различные маркеры пунктов списка.

```
<HTML>
<HEAD>
    <style type="text/css">
        ul.a {list-style-type:circle;}
        ul.b {list-style-type:disc;}
        ul.c {list-style-type:square;}

        ol.d {list-style-type:armenian;}
        ol.e {list-style-type:cjk-ideographic;}
        ol.f {list-style-type:decimal;}
        ol.g {list-style-type:decimal-leading-
           zero;}
        ol.h {list-style-type:georgian;}
        ol.i {list-style-type:hebrew;}
        ol.j {list-style-type:hiragana;}
        ol.k {list-style-type:hiragana-iroha;}
        ol.l {list-style-type:katakana;}
        ol.m {list-style-type:katagana-iroha;}
        ol.n {list-style-type:lower-alpha;}
        ol.o {list-style-type:lower-greek;}
        ol.p {list-style-type:lower-latin;}
        ol.q {list-style-type:lower-roman;}
        ol.r {list-style-type:upper-alpha;}
        ol.s {list-style-type:upper-latin;}
        ol.t {list-style-type:upper-roman;}
        ol.u {list-style-type:none;}
    </style>
</HEAD>
<BODY>
    <ul class="b">
       <li>Тип - диски</li>
       <li>Чай</li>
```

```
        <li>Кока Кола</li>
    </ul>
    <ul class="c">
        <li>Тип - квадраты</li>
        <li>Чай</li>
        <li>Кока Кола</li>
    </ul>
    <ol class="d">
        <li>Армянский тип</li>
        <li>Чай</li>
        <li>Кока Кола</li>
    </ol>
    <ol class="g">
        <li>Десятичный тип с нулем впереди</li>
        <li>Чай</li>
        <li>Кока Кола</li>
    </ol>
    <ol class="j">
        <li>Хираганский тип</li>
        <li>Чай</li>
        <li>Кока Кола</li>
    </ol>
    <ol class="o">
        <li>Греческий алфавит в нижнем регист-
            ре</li>
        <li>Чай</li>
        <li>Кока Кола</li>
    </ol>
</BODY>
</HTML>
```

Контрольные вопросы

1. Для чего используются таблицы стилей?
2. Перечислите способы применения таблиц стилей к документу.
3. Охарактеризуйте связывание документа с таблицей стилей. Какой тег используется для связывания?
4. Что называется внедрением? Какой тег используется для внедрения?

5. В каком случае используется встраивание CSS в тэги документа?

6. Охарактеризуйте импортирование таблиц стилей в HTML-файл? Приведите пример импортирования таблицы стилей?

7. Каким образом браузер расставляет приоритеты таблиц стилей?

8. Что такое правило?

9. Какой формат имеет правило?

10. Что определяет класс?

11. Как создать класс, не связанный с определенным тегом?

12. Для чего используются псевдоклассы?

13. Какой синтаксис у псевдокласса?

ЛИТЕРАТУРА

1. Квинт, И. Создаем сайты с помощью HTML, XHTML и CSS, включая HTML5 и CSS3 / И. Квинт. – 2-е изд. – СПб. : Питер, 2012. – 448 с. : ил., табл.

2. Макфарланд, Д. Новая большая книга CSS / Д. Макфарланд; пер. С. Черников. – СПб. : Питер, 2016. – 716 с. : ил., табл.

3. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон; пер. Н. Вильчинского. – 4-е изд. – СПб. : Питер, 2018. – 766 с. : ил., табл.

4. Новиков, В. А. Информационные системы и сети: с электрон. прил.: учебное пособие / В. А. Новиков, А. В. Новиков, В. В. Матвеенко. – Минск : Изд-во Гревцова, 2014. – 448 с. : ил.

5. Робсон, Э. Изучаем HTML, XHTML и CSS / Э. Робсон, Э. Фримен; пер. В. Черник. – 2-е изд. – СПб. : Питер, 2015. – 720 с. : ил., табл.

6. Сетевые технологии и базы данных: лабораторный практикум: в 2 ч. / сост. А. В. Манюкевич. – Минск : БНТУ, 2012. – Ч. 2: Разработка HTML-страниц. – 2012. – 61 с. : ил., табл.

7. Тетерукова, Н. А. Интернет-программирование: лабораторный практикум для учащихся специальности 2-40 01 01 «Программное обеспечение информационных технологий»: в 2 ч. / Н. А. Тетерукова. – Минск : МГВРК, 2007. – 90 с.

8. HTML5BOOK [Электронный ресурс]. – Режим доступа: https://html5book.ru. – Дата доступа: 26.08.2021.

9. Как создать свой сайт бесплатно [Электронный ресурс]. – Режим доступа: http://www.site-do.ru. – Дата доступа: 26.08.2021.

10. Уроки CSS [Электронный ресурс]. – Режим доступа: http://uroki-css.ru. – Дата доступа: 26.08.2021.

ПРИЛОЖЕНИЕ

Свойства CSS

Свойство	Описание	Возможные значения
Свойства шрифта		
font-family	Тип шрифта или семейство шрифтов	Шрифты: Arial, Verdana и т. д. Семейства: *serif* (с засечками), *sans-serif* (без засечек), *fantasy* (с украшениями), *monospace* (моноширинные) и т. д.
font-size	Размер шрифта	Стандартные единицы длины (px, cm и т. д.), проценты, ключевые слова (*xx-small*, *x-small*, *small*, *medium*, *large*, *x-large*, *xx-large*)
font-style	Вид начертания	*Normal* (нормальное начертание), *italic* и *oblique* (курсивное начертание)
font-variant	Регистр букв	*Normal*, *small-caps* (заменяет строчные буквы на маленькие заглавные)
font-weight	Жирность шрифта	Числа от 100 до 900, ключевые слова *bold*, *bolder*, *lighter*
Свойства текста		
letter-spacing	Расстояние между буквами	Стандартные единицы длины (px, cm и т. д.), ключевое слово *normal*
line-height	Расстояние между строками	Стандартные единицы длины (px, cm и т. д.), проценты, ключевое слово *normal*
text-align	Выравнивание текста	*Left* (по левому краю), *right* (по правому краю), *center* (по центру) и *justify* (по ширине страницы)
text-decoration	Выделение текста	*Underline* (подчеркивание), *overline* (линия над текстом), *line-through* (зачеркивание) и т. д.
text-indent	Отступ первой строки абзаца	Стандартные единицы длины (px, cm и т. д.), проценты
word-spacing	Расстояние между словами	Стандартные единицы длины (px, cm и т. д.), ключевое слово *normal*
text-transform	Изменение регистра букв	*Capitalize* (первая буква каждого слова выводится в верхнем регистре), *upper-case* (все в верхнем регистре), *lowercase* (все в нижнем регистре), *none* (регистр букв не изменяется)
Свойства цвета и фона		
color	Цвет содержимого тега, например, текста	Ключевое слово (*white*, *red* и т. д.), код RGB
background-color	Цвет фона элемента	Ключевое слово (*white*, *red* и т. д.), код RGB, *transparent* (прозрачный фон)
background-image	Фоновое изображение элемента	URL или ключевое слово *none*

Свойство	Описание	Возможные значения
background-attachment	Перемещение фонового изображения при прокрутке	*Scroll* (фоновое изображение перемещается вместе с текстом), *fixed* (изображение не двигается относительно окна браузера во время прокрутки документа)
background-position	Точка начала копирования фонового изображения вправо и вниз	Стандартные единицы длины (px, cm и т. д.), проценты, ключевые слова: горизонтальное смещение – *left, right*, *center*, вертикальное смещение – *top*, *center*, *bottom*
background-repeat	Способ копирования фонового изображения	*Repeat-x* (копирование изображения только по вертикали), *repeat-y* (копирование изображения только по горизонтали), *no-repeat* (фоновое изображение не копируется), *repeat* (копирование изображения по вертикали и горизонтали)
Свойства обрамления		
border-color	Цвет рамки	Может принимать от одного до четырех значений цвета. Если установлено только одно значение, то все четыре стороны рамки будут одного цвета. Четыре значения определяют цвет каждой из сторон рамки в следующем порядке: верх, правая сторона, левая сторона, низ
border-width	Толщина рамки	Стандартные единицы длины (px, cm и т. д.), ключевые слова: *thin* (тонкая), *medium* (средняя), *thik* (толстая). Может принимать от одного до четырех значений
border-style	Стиль рамки	*None*, *dotted* (точечная), *dashed* (штриховая), *double* (двойная), *solid* (сплошная), *outset* (приподнятая), *inset* (утопленная), *ridge* (объемная) и т. д. Может принимать от одного до четырех значений
padding	Расстояние между содержимым элемента и рамкой	Стандартные единицы длины (px, cm и т. д.), проценты. Может принимать от одного до четырех значений
margin	Ширина полей	Стандартные единицы длины (px, cm и т. д.), проценты, ключевое слово *auto*. Может принимать от одного до четырех значений
П р и м е ч а н и е. Все вышеперечисленные свойства обрамления можно применять отдельно для каждой стороны элемента, указывая через дефис *left*, *right*, *top*, или *bottom*, например, *margin-left*, *padding-right*, *border-top-color*, *border-bottom-width*		

Свойство	Описание	Возможные значения
Свойства позиционирования		
width	Ширина элемента	Стандартные единицы длины (px, cm и т. д.), проценты, ключевое слово *auto*
height	Высота элемента	Стандартные единицы длины (px, cm и т. д.), проценты
top	Y-координата верхнего левого угла элемента	Стандартные единицы длины (px, cm и т. д.), проценты, ключевое слово *auto*
left	X-координата верхнего левого угла элемента	Стандартные единицы длины (px, cm и т. д.), проценты, ключевое слово *auto*
float	Определяет область отображения тега как «плавающий» элемент и выравнивает его по горизонтали	*None* (отключает данное свойство), *left* (выравнивает по левому краю), *right* (выравнивает по правому краю)
clear	Расположение содержимого тега относительно «плавающего» элемента	*None* (может размещаться рядом с «плавающими» элементами по обе стороны), *left* (слева от «плавающего» элемента), *right* (справа от «плавающего» элемента), *both* (запрещает размещение рядом с «плавающим» элементом)
z-index	Порядок наложения элементов	Число (сверху будет отображаться элемент, имеющий большее значение), ключевое слово *auto* (меньше любого значения)
Визуальные свойства		
cursor	Вид курсора над элементом	*Crosshair*, *hand*, *move*, *e-resize*, *ne-resize*, *nw-resize*, *n-resize*, *sw-resize*, *se-resize*, *s-resize*, *w-resize*, *text*, *wait*, *help*, *auto* (стандартная форма курсора для данного элемента), *default* (стандартная форма курсора для окна браузера)
visibility	Видимость элемента	*Inherit* (значения данного свойства наследуются от родительского элемента), *none* (элемент невидим), *visible* (элемент видим)
display	Способ отображения элемента	*Block* (элемент отображается как блочный: пустая строка до и после элемента), *inline* (элемент отображается как встроенный), *none* (элемент не отображается, предыдущий и последующий элементы сдвигаются вместе) и т. д.

Свойство	Описание	Возможные значения
overflow	Стиль отображения текста, переполнившего границы элемента	*Scroll* (создать линейки прокрутки для просмотра не поместившегося содержимого), *hidden* (спрятать содержимое, не попавшее в видимую часть элемента), *visible* (все содержимое тега будет видимо, изменятся размеры элемента), *auto*
Свойства списков		
list-style-type	Стиль элементов списка	Для неупорядоченных списков: *disc* (диск), *circle* (окружность), *square* (квадрат) или *none* (нет маркера). Для упорядоченных списков: *decimal* (арабские цифры), *lower-roman* (маленькие римские цифры), *upper-roman* (большие римские цифры), *lower-alpha* (строчные латинские буквы), *upper-alpha* (заглавные латинские буквы) и *none* (нет нумерации)
list-style-image	Изображение, используемое вместо маркера	*Url* (URL изображения)
Свойства полос прокрутки		
scrollbar-arrow-color	Цвет стрелок на кнопке со стрелками	Ключевое слово (*white*, *red* и т. д.), код RGB
scrollbar-base-color	Цвет основных элементов полосы прокрутки: ползунка, кнопок со стрелками, дорожки для ползунка	Ключевое слово (*white*, *red* и т. д.), код RGB
scrollbar-face-color	Цвет ползунка и кнопок со стрелками	Ключевое слово (*white*, *red* и т. д.), код RGB
scrollbar-highlight-color	Цвет подсветки, создающий эффект объемности	Ключевое слово (*white*, *red* и т. д.), код RGB
scrollbar-shadow-color	Цвет тени для ползунка и кнопок со стрелками	Ключевое слово (*white*, *red* и т. д.), код RGB
scrollbar-track-color	Цвет дорожки для ползунка	Ключевое слово (*white*, *red* и т. д.), код RGB