



Н. А. ТЕТЕРУКОВА

ИНТЕРНЕТ-ПРОГРАММИРОВАНИЕ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

ЧАСТЬ 1

WEB-СЕРВЕР

ЯЗЫК ГИПЕРТЕКСТОВОЙ РАЗМЕТКИ HTML

ЯЗЫК ПРОГРАММИРОВАНИЯ НА СТОРОНЕ

КЛИЕНТА JAVASCRIPT

МИНСК 2007

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«МИНСКИЙ ГОСУДАРСТВЕННЫЙ ВЫСШИЙ
РАДИОТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

Н. А. ТЕТЕРУКОВА

ИНТЕРНЕТ-ПРОГРАММИРОВАНИЕ

Лабораторный практикум
для учащихся специальности 2-40 01 01
«Программное обеспечение информационных технологий»

В двух частях

ЧАСТЬ 1

WEB-СЕРВЕР

ЯЗЫК ГИПЕРТЕКСТОВОЙ РАЗМЕТКИ HTML

ЯЗЫК ПРОГРАММИРОВАНИЯ НА СТОРОНЕ

КЛИЕНТА JAVASCRIPT

МИНСК 2007

УДК 681.3.06(075)
ББК 32.973.202.я7
Т37

Рекомендовано и изданию кафедрой информатики и Научно-методическим советом Учреждения образования «Минский государственный высший радиотехнический колледж» (протокол № 1 от 26.09.2007 г.)

Рецензенты

Ю. А. Скудняков, заведующий кафедрой информатики
МГВРК, канд. техн. наук, доцент

А. В. Пашенко, доцент кафедры, канд. техн. наук, УО «БНТУ»

Тетерукова Н. А.

Т37 Интернет-программирование : лаб. практикум для учащихся специальности 2-40 01 01 «Программное обеспечение информационных технологий» : в 2 ч. / Н. А. Тетерукова. – Мн. : МГВРК, 2007– .

ISBN 978-985-6851-15-8

Ч. 1 : Web-сервер. Язык гипертекстовой разметки HTML. Язык программирования на стороне клиента JavaScript / Н. А. Тетерукова. – Мн. : МГВРК, 2007. – 88 с.

ISBN 978-985-6851-16-5 (ч. 1)

Данный лабораторный практикум содержит краткие теоретические сведения и практические задания по языку гипертекстовой разметки HTML и языку программирования на стороне клиента JavaScript.

Предназначен для преподавателей и учащихся колледжа.

УДК 681.3.06(075)
ББК 32.973.202.я7

ISBN 978-985-6851-16-5 (ч. 1)
ISBN 978-985-6851-15-8

© Тетерукова Н. А., 2007

© Оформление. Учреждение образования «Минский государственный высший радиотехнический колледж», 2007

Предисловие

Современный специалист в области разработки программного обеспечения должен уметь пользоваться технологиями Интернет и владеть основами создания Web-приложений различного уровня сложности.

Целью дисциплины «Интернет-программирование» является изучение основ создания сайтов и программирования для сети Интернет.

Данный практикум является руководством для проведения лабораторных работ по курсу «Интернет-программирование».

Описание каждой лабораторной работы содержит теоретические сведения, необходимые для ее выполнения, примеры решения простейших задач, порядок выполнения работы, задания и контрольные вопросы.

При защите лабораторной работы необходимо предоставить отчет, продемонстрировать результат выполнения работы и ответить на контрольные вопросы.

Для удобства использования данный практикум представлен в двух частях.

В первой части практикума содержатся задания по разделам «Web-сервер», «Язык гипертекстовой разметки HTML» и «Язык Javascript».

Вторая часть практикума включает задания по разделам «Интерфейс CGI» и «Язык программирования PHP».

ЛАБОРАТОРНАЯ РАБОТА 1

УСТАНОВКА И НАСТРОЙКА WEB-СЕРВЕРА APACHE

Цель работы: получение практических навыков установки и настройки Web-сервера Apache.

Рекомендации по выполнению лабораторной работы

В связи с тем, что настройка отдельно установленного Web-сервера Apache является трудоемким процессом, занимающим значительное время, для проведения данной лабораторной работы целесообразно использовать самонастраивающийся пакет

«Денвер». В этом случае изменение отдельных директив конфигурационного файла сервера позволит проследить результат каждого действия и, в случае ошибки, быстро найти и устранить ее. Кроме того, комментарии конфигурационных файлов Apache в данном пакете переведены на русский язык, что обеспечивает осознанное изменение настроек учащимися.

Теоретические сведения

Web-сервер – это программа, позволяющая просматривать результат выполнения серверных скриптов и осуществлять их отладку на локальном компьютере.

«Денвер» («Джентльменский набор Web-разработчика» или сокращенно «Д.н.в.р.») представляет собой набор дистрибутивов и программную оболочку, используемые Web-разработчиками для отладки сайтов на локальном компьютере без необходимости выхода в Интернет. Базовый пакет «Денвер», кроме Web-сервера Apache, содержит интерпретатор PHP, Perl, сервер MySQL и другие компоненты, обеспечивающие работу системы в целом.

Для установки пакета «Денвер» необходимо запустить файл дистрибутива *Denver.exe* и следовать дальнейшим инструкциям (будет предложено выбрать директорию, в которую нужно установить пакет, указать имя нового виртуального диска и т. д.).

Перед началом работы с Web-сервером его необходимо запустить. Для запуска, перезапуска или остановки сервера Apache нужно запустить файлы *Run.exe*, *Restart.exe*, *Stop.exe* соответственно. Данные файлы располагаются в подкаталоге */etc* корневого каталога пакета «Денвер» (обычно корневым является каталог *WebServers*).

Настройка сервера Apache заключается в редактировании главного конфигурационного файла *httpd.conf*, который обычно расположен в подкаталоге *usr/local/apache/conf* корневого каталога пакета «Денвер». Конфигурация сервера определяется значениями конфигурационных переменных, называемых *директивами*. Комментарии в конфигурационном файле обозначаются символом #.

П р и м е ч а н и е. Для того, чтобы изменения в файле *httpd.conf* вступили в силу, необходимо перезапустить Web-сервер.

Основные директивы файла *httpd.conf* [5]:

ServerRoot задает вершину дерева каталогов, в которых содержатся файлы конфигурации, регистрации и отслеживания ошибок.

Например: *ServerRoot /usr/local/apache*

ServerName задает имя сервера, которое будет использоваться при переедресации URL.

Например: *ServerName localhost*

ServerAdmin – адрес, по которому будут направляться сообщения о проблемах с сервером.

Например: *ServerAdmin webmaster@localhost*

ServerAlias задает альтернативные имена Web-узла.

Например: *ServerAlias my_site.ru*

DocumentRoot задает путь к корневому каталогу документов Web-сервера.

Например: *DocumentRoot /home/localhost/www*

Alias указывает псевдоним каталога локальной файловой системы, находящегося за пределами поддерева, определяемого директивой *DocumentRoot*.

Например: *Alias /images/ /public/img/*

При запросе по адресу <http://localhost/images/1.gif> сервер вернет файл */public/img/1.gif*.

ScriptAlias определяет псевдоним каталога для CGI-сценариев.

Например: *ScriptAlias /cgi-bin/ /home/cgi-glob/*

ErrorLog указывает путь к файлу, в который будут записываться диагностические сообщения об ошибках.

Например: *ErrorLog logs/error.log*

TransferLog указывает путь к файлу, в котором сервер регистрирует входящие запросы.

Например: *TransferLog logs/transfer.log*

П р и м е ч а н и е. Путь в директивах **ErrorLog** и **TransferLog** может указываться относительно каталога, заданного директивой **ServerRoot**.

Существует возможность создания нескольких Web-узлов на одной машине-сервере и обслуживания их одним установленным экземпляром сервера Apache. Другими словами, сервер может обрабатывать запросы к множеству различных Web-узлов. Данная возможность сервера получила название *виртуальный хостинг* [5]. Каждый из созданных узлов (виртуальных

хостов) может иметь свою конфигурацию, необходимую для решения конкретных задач.

Для создания и конфигурирования виртуального хоста используется блок директив `<VirtualHost name / ip_address>` `</VirtualHost>`.

Примечание. В корневом каталоге документов Web-сервера должен быть создан каталог, соответствующий имени хоста.

Пример описания виртуального хоста:

```
<VirtualHost myhost.com>
ServerAdmin webmaster@myhost.com
DocumentRoot /home/myhost.com/www
ServerName myhost.com
ScriptAlias/cgi-bin/ /home/myhost.com/cgi-bin/
ErrorLog /home/myhost.com/error.log
TransferLog /home/myhost.com/transfer.log
</VirtualHost>
```

Создать виртуальный хост можно не только «вручную», определив директиву `<VirtualHost>`. В «Денвере», для того чтобы добавить новый виртуальный хост со стандартными настройками, достаточно создать структуру его каталогов в директории `/home`. Сервер Apache автоматически «увидит» изменения при следующем запуске. При этом изменять файл `httpd.conf` не нужно.

Порядок выполнения работы

1. Установите пакет «Денвер»:
 - устанавливайте в директорию по умолчанию (`C:\WebServers`);
 - создайте виртуальный диск с именем `Z`;
 - выберите вариант установки `1`;
 - создайте на Рабочем столе ярлыки для запуска пакета.
2. Запустите Web-сервер.
3. Для проверки работоспособности сервера наберите в адресной строке браузера `http://localhost`. Если сервер работает правильно, должна загрузиться HTML-страница `index.html` из корневого каталога документов Web-сервера (рис. 1.1).
4. Откройте конфигурационный файл `httpd.conf`.
5. Создайте директорию `номер_группы.conf` на диске `Z`. Скопируйте в созданный каталог содержимое каталога `Z:\usr\local\apache`. Измените значение директивы `ServerRoot` – укажите путь к каталогу `номер_группы.conf`.

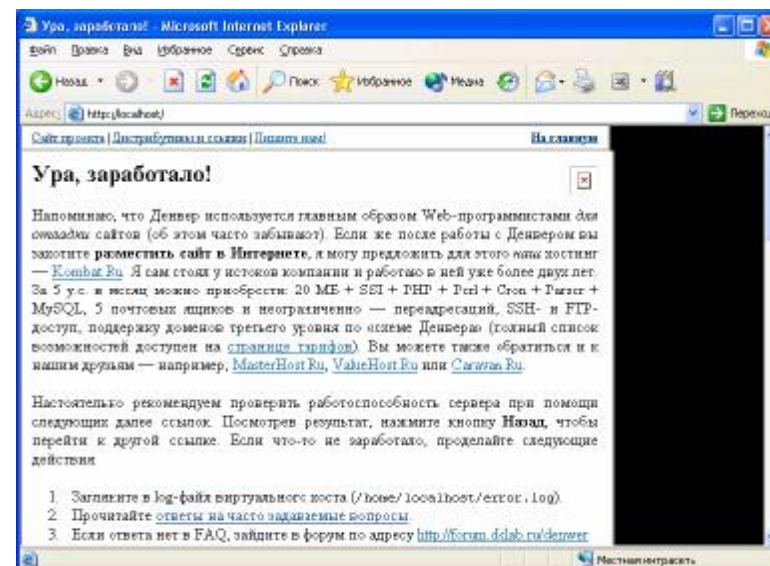


Рис. 1.1. Страница `index.html`

6. Создайте папку `img` на диске `D`. В графическом редакторе Paint создайте изображение и сохраните его в созданной папке. В файле `httpd.conf` определите псевдоним для папки `img` и откройте созданный файл в браузере.
7. Используя блок директив `<VirtualHost>` `</VirtualHost>`, создайте виртуальный хост с именем, соответствующим номеру группы: `номер_группы.by`.
8. Разместите на созданном виртуальном хосте файл `index.html` следующего содержания:

```
<html>
<body>
WEB-сервер установлен и настроен
<br>
Данную работу выполнили: ФИО учащихся
</body>
</html>
```
9. Проверьте работоспособность виртуального хоста `номер_группы.by`.
10. Добавьте виртуальный хост `myserver.com` путем создания структуры его каталогов в директории `home`.

11. Проверьте работоспособность виртуального хоста *myserver.com*.
12. Покажите результаты работы преподавателю.
13. Остановите Web-сервер.

Контрольные вопросы

1. Объясните назначение Web-сервера.
2. Какие программные средства входят в состав пакета «Денвер»?
3. Как запустить (перезапустить, остановить) сервер Apache?
4. Каким образом осуществляется настройка сервера?
5. Что такое директива?
6. Перечислите известные вам директивы конфигурационного файла *httpd.conf*.
7. Объясните назначение директив **ServerRoot** и **DocumentRoot**.
8. Объясните назначение директив **Alias** и **ScriptAlias**.
9. Что такое виртуальный хостинг?
10. Как можно создать виртуальный хост?

Отчетность по лабораторной работе

1. Измененный файл *httpd.conf* на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - измененные или добавленные строки файла *httpd.conf* с подробными комментариями;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 2

СОЗДАНИЕ HTML-ДОКУМЕНТОВ

Цель работы: формирование практических умений создания простейших HTML-документов.

Теоретические сведения

Для создания и форматирования Web-документов используется язык гипертекстовой разметки HTML (Hyper Text Markup Language).

В HTML определен стандартный набор тегов (дескрипторов) – команд, определяющих форматирование документа. Теги заключаются в треугольные скобки < >. Большинство тегов, как правило, используются парами. Сначала указывается открывающий тег, который объясняет браузеру, что делать с после-

дующим текстом. Затем следует закрывающий тег, ограничивающий область действия первого. Закрывающий тег отличается от открывающего наличием косой черты (слэша). В некоторых случаях закрывающий тег не требуется.

В HTML регистр символов, определяющих теги, не учитывается.

Web-документ ограничивается тегами <HTML> и </HTML>, которые определяют соответственно начало и конец документа.

В структуре HTML-документа выделяются заголовок (<HEAD> </HEAD>) и тело документа (<BODY> </BODY>). Заголовок может содержать заключенное в теги <TITLE> </TITLE> заглавие (или название) страницы, а также META-информацию.

Для создания HTML-документа нужно выполнить следующую последовательность действий:

- 1) запустить приложение «Блокнот»;
- 2) набрать исходный текст документа, например:

```
<HTML>
<HEAD><TITLE> Заглавие документа </TITLE></HEAD>
<BODY>
    Содержимое документа
</BODY>
</HTML>
```

- 3) сохранить файл с расширением .html или .htm, например *first.html* [6].

Для просмотра HTML-документа откройте созданный файл в браузере (например, Internet Explorer, Opera). Документ *first.html*, открытый в браузере Internet Explorer (IE), представлен на рис. 2.1.

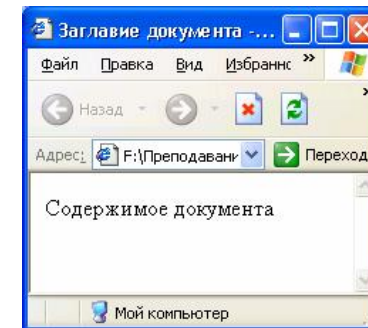


Рис. 2.1. Документ *first.html* в браузере Internet Explorer

Атрибуты тега **<BODY>** [2] представлены в табл. 2.1.

Т а б л и ц а 2.1

| Атрибут | Назначение |
|------------|---|
| BACKGROUND | Указывает адрес изображения, которое следует использовать в качестве фона документа. Если вместе с этим атрибутом используется атрибут BGPROPERTIES со значением fixed, фоновое изображение не будет прокручиваться |
| BGCOLOR | Определяет цвет фона документа |
| TEXT | Определяет цвет текста в документе |
| LINK | Определяет цвет ссылок, которые не были посещены |
| ALINK | Определяет цвет активных ссылок (ссылка является активной в момент нажатия на нее) |
| VLINK | Определяет цвет ссылок на просмотренные документы |
| TOPMARGIN | Определяет ширину (в пикселях) верхнего поля документа |
| LEFTMARGIN | Определяет ширину (в пикселях) левого поля документа |

При определении цвета для документа HTML могут использоваться названия цветов или их обозначения в шестнадцатеричной системе кодирования RGB. Например, следующие строки идентичны:

```
<BODY BGCOLOR="#FFFFFF" >
<BODY BGCOLOR="WHITE" >
```

Для включения комментариев в HTML-код используются последовательности символов `<!--` и `-->`.

```
<!-- Это комментарий -->
```

Порядок выполнения работы

1. Создайте документ **Lab2.html**, содержащий краткие сведения о вашей биографии.
2. Установите название документа, соответствующее вашей фамилии, имени и номеру группы.
3. Установите для документа серый цвет фона и синий цвет текста.
4. Определите ширину верхнего и нижнего полей документа, равную 10 пикселям.
5. Определите ширину левого и правого полей документа, равную 15 пикселям.

6. Скопируйте созданный документ в файл **Lab2_new.html**.
7. Установите изображение в качестве фона документа **Lab2_new.html**.
8. Используя шестнадцатеричные коды, измените цвет текста данного документа.

Контрольные вопросы

1. Какую структуру имеет HTML-файл?
2. Каким тегом обозначается заголовок документа?
3. Какой тег используется для определения тела html-документа?
4. Перечислите известные вам атрибуты тега **<BODY>**.
5. Укажите способы определения цвета в HTML-документе.

Отчетность по лабораторной работе

1. Файлы **Lab2.html** и **Lab2_new.html** на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - структуру документа **Lab2.html**;
 - атрибуты тега **<BODY>**, используемые в документе **Lab2_new.html**, с указанием их назначения;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 3

ВСТАВКА ТЕКСТА В HTML И ЕГО ПОСЛЕДУЮЩЕЕ ЛОГИЧЕСКОЕ И ФИЗИЧЕСКОЕ ФОРМАТИРОВАНИЕ

Цель работы: формирование практических умений физического и логического форматирования текста в HTML.

Теоретические сведения

Различают теги логического и физического форматирования текста. Теги логического форматирования определяют, какую смысловую нагрузку несет выделенный фрагмент, т. е. чем он является в документе (заголовком, абзацем, цитатой и т. д.). Теги физического форматирования определяют в первую очередь, как будет выглядеть выделенный фрагмент.

Теги логического форматирования [2] представлены в табл. 3.1.

Т а б л и ц а 3.1

| Тег | Назначение |
|--------------|---|
| <CITE> | Используется для выделения цитаты (обычно курсив) |
| <CODE> | Выделяет программный код |
| | Используется для выделения фрагмента текста, имеющего большое значение (обычно курсив) |
| <KBD> | Выделяет текст, который предлагается набрать на клавиатуре (обычно моноширинный шрифт) |
| <SAMP> | Используется для выделения знаков, на которых необходимо акцентировать внимание (обычно моноширинный шрифт) |
| | Используется для выделения очень важного фрагмента текста (обычно полужирный) |
| <VAR> | Используется для выделения имени переменной (обычно курсив) |
| <DFN> | Используется для выделения определения (обычно курсив) |
| <BLOCKQUOTE> | Выделяет цитату (обычно используется отступ от левого поля) |
| <P> | Выделяет отдельный абзац в тексте |
| <H1> ...<H6> | Выделяют заголовки 6 уровней (H1 – самый важный) |

Использование тегов логического форматирования демонстрируется в приведенном ниже примере, результат приведен на рис. 3.1.

```
<HTML>
<HEAD>
<TITLE>Логическое форматирование</TITLE></HEAD>
<BODY>
<H2>
Теги логического
форматирования </H2>
<P>
<CITE>Цитата</CITE></P>
<P><BLOCKQUOTE>
Еще одна цитата</BLOCKQUOTE></P>
<P><CODE>
Программный код </CODE></P>
<P><STRONG>
Важный текст</STRONG></P>
<P><DFN>
Определение </DFN></P>
</BODY>
</HTML>
```

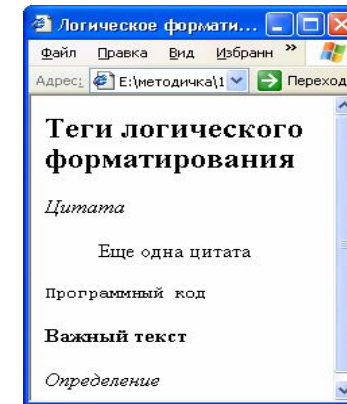


Рис 3.1. Использование тегов логического форматирования

Теги физического форматирования представлены в табл. 3.2:

Т а б л и ц а 3.2

| Тег | Назначение |
|---------------|--|
| <BASEFONT> | Определяет цвет, размер и тип основного шрифта документа (соответственно атрибуты color, size, face). Данный тег не является контейнером |
| | Позволяет изменять цвет, размер и тип шрифта |
| <I> | Выделяет текст курсивом |
| | Выделяет текст жирным шрифтом |
| <U> | Подчеркивает текст |
| <S>, <STRIKE> | Зачеркивает текст |
| <BIG> | Отображает текст увеличенным шрифтом (относительно текущего) |
| <SMALL> | Отображает текст уменьшенным шрифтом (относительно текущего) |
| <SUP> | Отображает текст со сдвигом вверх (верхний индекс) |
| <SUB> | Отображает текст со сдвигом вниз (нижний индекс) |
| <TT> | Отображает текст моноширинным шрифтом |
| | Переход на новую строку. Используется без парного закрывающего тега |
| <CENTER> | Горизонтальное выравнивание текста по центру |
| <PRE> | Предварительное форматирование: вывод текста в том виде, в котором он представлен в исходном документе |

Для добавления на страницу горизонтальной линии используется тег <HR>.

Использование тегов физического форматирования демонстрируется в приведенном ниже примере, результат – на рис. 3.2.

```
<HTML>
<HEAD>
<TITLE>Физическое форматирование</TITLE></HEAD>
<BODY>
<H2> Теги физического форматирования </H2>
  <FONT size=5 color=red>
    Текст красного цвета<BR>
  <BIG>Увеличенный</BIG>размер шрифта<BR>
  <SMALL>Уменьшенный </SMALL>размер шрифта<BR>
</FONT>
  <B>жирный</B> шрифт<BR>
  <I>курсив</I><BR>
  <U>подчёркнутый</U>текст<BR>
  <STRIKE>зачеркнутый</STRIKE>текст<BR>
  <SUP>Верхний</SUP> индекс <BR>
  <SUB>Нижний</SUB> индекс <BR>
  <B>Абзацы могут располагаться</B>
  <P align=left>слева,</P>
  <P align=center>по центру</P>
  <P align=right>или справа</P>
</BODY>
</HTML>
```

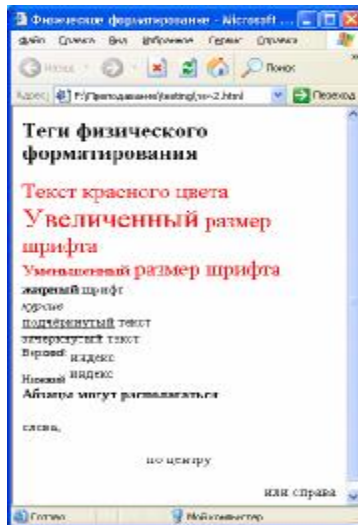


Рис. 3.2. Использование тегов физического форматирования

Порядок выполнения работы

1. Создайте HTML-документ **Lab3.html**.
2. Установите название документа «Форматирование текста».
3. Определите цвет текста и фона документа.
4. Определите параметры основного шрифта документа.
5. Наберите и отформатируйте текст индивидуального задания, выданного преподавателем, используя теги логического и физического форматирования.
6. Добавьте горизонтальную линию, установите ее цвет, толщину, длину и способ выравнивания.
7. Используя тег предварительного форматирования **<PRE>**, добавьте на страницу HTML-код выполненного задания.

Контрольные вопросы

1. Чем отличается физическое форматирование от логического?
2. Перечислите известные вам теги логического форматирования.
3. Перечислите известные вам теги физического форматирования.
4. Какие теги форматирования не являются контейнерами?
5. Укажите отличия использования тегов **<P>** и **
**.
6. Какой атрибут тега **<P>** позволяет выровнять текст по горизонтали?
7. Что такое предварительное форматирование?

Отчетность по лабораторной работе

1. Файл **Lab3.html** на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - структуру документа **Lab3.html** (без указания тегов, используемых для форматирования текста индивидуального задания);
 - теги логического и физического форматирования, используемые для выполнения индивидуального задания, с указанием их назначения;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 4 ОРГАНИЗАЦИЯ СИСТЕМЫ ССЫЛОК И ВНЕДРЕНИЕ В HTML ГРАФИКИ

Цель работы: формирование практических умений организации системы гиперссылок и использования графических изображений в HTML-документах.

Теоретические сведения

Ссылки

Ссылка состоит из двух частей:

- элемент привязки (или якорь – *anchor*) – место в документе, отмеченное как ссылка. Существуют два типа элементов привязки: текстовый и графический;
- ссылка на URL – сообщает браузеру, какой документ нужно загружать при щелчке на ссылке [2].

Для организации ссылок в HTML используется тег **<A>**.

Рассмотрим структуру ссылки на конкретном примере (рис. 4.1).

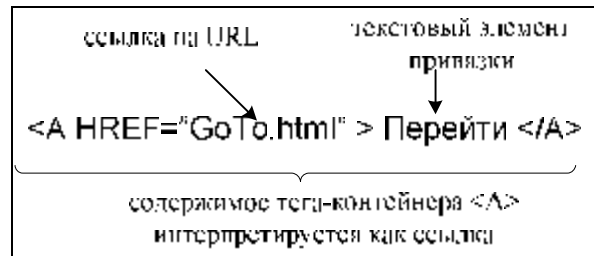


Рис. 4.1. Структура ссылки

Могут использоваться относительные и абсолютные ссылки на URL файла.

Относительной называется ссылка на файл, находящийся на том же компьютере. Это означает, что URL указывается относительно компьютера и каталога, из которого браузер первоначально загружает Web-страницу [2].

Например:

```
<A href="next.html">  
Перейти на страницу next.html</A>
```

Абсолютной называется ссылка, в которой указан полный путь к файлу (компьютер, каталог, имя файла) [2].

Например:

```
<A href="http://www.yandex.ru/index.html">  
Перейти на портал Yandex</A>
```

Можно создать ссылку не только на другой документ, но и на конкретный раздел текущего. Такие ссылки называют *внутренними* ссылками или *закладками*.

Пример внутренней ссылки приведен ниже:

```
<HTML>  
<HEAD><TITLE>Внутренние ссылки</TITLE></HEAD>  
<BODY>  
<H2> Создание внутренних ссылок</H2>  
<P><a href=#five>Переход к разделу 5</a></P>  
<BR><BR>  
<P> Раздел 1  
...</P>  
<P> Раздел 2  
...</P>  
<P> Раздел 3  
...</P>  
<P> Раздел 4  
...</P>  
<P> <A name=five>Раздел 5</A>  
...</P>  
</BODY>  
</HTML>
```

Таким образом, при помощи тега **<A>** создается элемент привязки, т. е. определяется место в документе, к которому нужно перейти. С помощью атрибута **NAME** элементу привязки присваивается имя **five**. Элемент привязки, созданный таким способом, в тексте выделяться не будет.

Для ссылки на созданную закладку атрибуту **HREF** тега **<A>** присваивается значение следующего вида:

путь_к_документу#имя_закладки.

Если закладка находится в том же документе, что и ссылка, путь к документу можно не указывать.

Вставка изображений

Для внедрения графики на HTML-страницу используется тег ****, который может включать в себя следующие атрибуты [8] (табл. 4.1):

Т а б л и ц а 4.1

| Атрибут | Назначение |
|---------|---|
| SRC | Обязательный атрибут, указывает адрес файла с изображением |
| HEIGHT | Определяет ширину изображения |
| WIDTH | Определяет высоту изображения |
| HSPACE | Определяет отступ изображения по горизонтали от других объектов документа |
| VSPACE | Определяет отступ по вертикали от других объектов документа |
| ALIGN | Указывает способ выравнивания изображения в документе (LEFT, RIGHT) |
| NAME | Определяет имя изображения, уникальное для данного документа |
| ALT | Определяет альтернативный текст |
| BORDER | Определяет ширину рамки вокруг изображения |

Пример добавления изображения *pict.jpg* на html-страницу приведен ниже, результат – на рис. 4.2.

```
<HTML>
<HEAD><TITLE>Тег IMG</TITLE></HEAD>
<BODY>
<H1> Использование тега
        &lt;IMG&gt;</H1>
<IMG src="pict.JPG" width="160"
      height="192" border="2"
      alt="картинка" align="right"
      hspace="20" vspace="30">
</BODY>
</HTML>
```

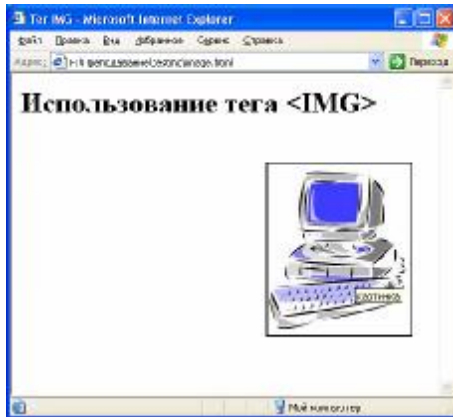


Рис. 4.2. Вставка изображения на html-страницу

Порядок выполнения работы

1. Создайте документ *Lab4_1.html*, содержащий перечень рабочих дней недели.
2. Установите название документа, цвет фона и текста.
3. Создайте документ *Lab4_2.html*, содержащий расписание занятий на неделю.
4. Установите название документа *Lab4_2.html*, цвет фона и текста.
5. Организуйте систему ссылок таким образом, чтобы при выборе определенного дня недели в документе *Lab4_1.html* осуществлялся переход к расписанию занятий данного дня в документе *Lab4_2.html*.
6. Добавьте в начало документа *Lab4_2.html* ссылки на расписание определенного дня недели.
7. Установите цвет ссылок для документов *Lab4_1.html* и *Lab4_2.html*.
8. Вставьте изображение на страницу *Lab4_1.html*.
9. Установите размер изображения, ширину рамки, выравнивание, отступ по горизонтали и вертикали, а также альтернативный текст.
10. Используйте добавленное изображение в качестве графического элемента привязки ссылки на документ *Lab4_2.html*.

Контрольные вопросы

1. Что такое элемент привязки?
2. Какой тег используется для создания ссылок?
3. В чем различие абсолютных и относительных ссылок?
4. Перечислите известные вам атрибуты тега `<A>`.
5. Как можно создать внутреннюю ссылку?
6. Какой тег используется для вставки изображения на html-страницу?
7. Как можно создать ссылку с графическим элементом привязки?
8. Какие атрибуты тега `` вы знаете?

Отчетность по лабораторной работе

1. Файлы *Lab4_1.html* и *Lab4_2.html* на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - структуру документов *Lab4_1.html* и *Lab4_2.html* с комментариями, поясняющими назначение конкретного тега и его атрибутов;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 5

РАБОТА С ТАБЛИЦАМИ. СОЗДАНИЕ СПИСКОВ

Цель работы: формирование практических умений работы с таблицами и списками в HTML.

Теоретические сведения

Таблицы

Таблица в HTML имеет следующий вид (рис. 5.1):

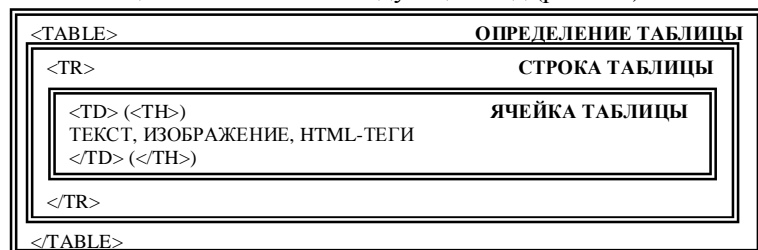


Рис. 5.1. Таблица в HTML

Атрибуты тега **<TABLE>** [8] приведены в табл. 5.1.

Т а б л и ц а 5.1

| Атрибут | Назначение |
|-------------|--|
| ALIGN | Определяет способ горизонтального выравнивания таблицы (LEFT, RIGHT, CENTER) |
| VALIGN | Определяет способ вертикального выравнивания таблицы (TOP, MIDDLE, BOTTOM) |
| BORDER | Определяет толщину рамки таблицы |
| CELLPADDING | Определяет расстояние между рамкой ячейки таблицы и ее содержимым |
| CELLSPACING | Определяет расстояние между границами соседних ячеек |
| WIDTH | Определяет ширину таблицы |
| HEIGHT | Определяет высоту таблицы |
| BGCOLOR | Определяет цвет фона ячеек таблицы |
| BACKGROUND | Заполняет фон таблицы изображением |

Ниже приведен пример создания простейшей таблицы, состоящей из двух строк и двух столбцов, результат представлен на рис. 5.2.

```
<HTML>
<HEAD><TITLE>Пример таблицы</TITLE></HEAD>
<BODY>
<TABLE width=50%
      align=center
      bgcolor=#a0dda0>
<TR>
  <TD>Ячейка 1</TD>
  <TD>Ячейка 2</TD>
</TR>
<TR>
  <TD>Ячейка 3</TD>
  <TD>Ячейка 4</TD>
</TR>
</TABLE>
</BODY>
</HTML>
```

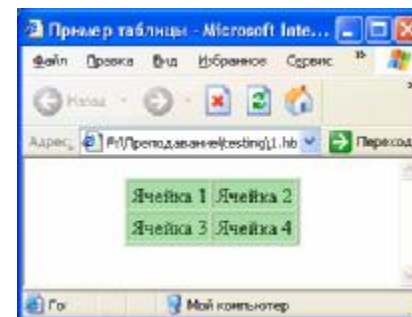


Рис. 5.2. Отображение созданной таблицы в Internet Explorer

Атрибуты тега **<TR>** [8] приведены в табл. 5.2:

Т а б л и ц а 5.2

| Атрибут | Назначение |
|---------|---|
| ALIGN | Определяет способ горизонтального выравнивания содержимого всех ячеек данного ряда (LEFT, RIGHT, CENTER, JUSTIFY) |
| VALIGN | Определяет способ вертикального выравнивания содержимого всех ячеек данного ряда |
| BGCOLOR | Определяет цвет фона для всех ячеек данного ряда |

Тег **<TH>** используется для определения заголовков в первой строке или в первом столбце таблицы.

Атрибуты тегов **<TD>** и **<TH>** [8] приведены в табл. 5.3:

Т а б л и ц а 5.3

| Атрибут | Назначение |
|------------|---|
| ALIGN | Определяет способ горизонтального выравнивания содержимого ячейки |
| VALIGN | Определяет способ вертикального выравнивания содержимого ячейки |
| WIDTH | Определяет ширину ячейки |
| HEIGHT | Определяет высоту ячейки |
| COLSPAN | Определяет количество столбцов, объединенных данной ячейкой |
| ROWSPAN | Определяет количество рядов, объединенных данной ячейкой |
| NOWRAP | Отменяет автоматический перенос слов в пределах текущей ячейки |
| BGCOLOR | Определяет цвет фона ячейки |
| BACKGROUND | Заполняет фон ячейки изображением |

Пример использования атрибутов COLSPAN и ROWSPAN приведен ниже, результат представлен на рис. 5.3.

```
<TABLE border=1>
<TR>
    <TD colspan="2">
        Элемент1</TD>
    </TR>
<TR>
    <TD>Элемент 2</TD>
    <TD rowspan="2">
        Элемент 3</TD>
    </TR>
<TR>
    <TD>Элемент 4</TD>
</TR>
</TABLE>
```

| | |
|-----------|-----------|
| Элемент 1 | Элемент 3 |
| Элемент 2 | |
| Элемент 4 | |

Рис. 5.3. Объединение строк и столбцов таблицы

Списки

Существуют следующие виды списков:

- упорядоченные (пронумерованные);
- неупорядоченные (непронумерованные);
- списки определений.

Для создания списков используются следующие теги [8] (табл. 5.4):

Т а б л и ц а 5.4

| Тег | Назначение |
|---------------|--|
| | Создает неупорядоченный список |
| | Создает упорядоченный список |
| | Определяет пункт меню внутри элементов OL или UL |
| <MENU>, <DIR> | Создает неупорядоченный список, подобный UL |
| <DL> | Ограничивает список определений |
| <DT> | Создает термин в списке определений внутри элемента DL |
| <DD> | Создает определение термина внутри элемента DL |

Атрибуты тега [8] приведены в табл. 5.5.

Т а б л и ц а 5.5

| Атрибут | Назначение |
|---------|--|
| START | Определяет первое число, с которого начинается нумерация пунктов |
| TYPE | Определяет стиль нумерации пунктов |

Значения атрибута TYPE для упорядоченного списка могут быть следующими:

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

В теге также может использоваться атрибут TYPE (со значениями DISC, SQUARE или CIRCLE).

Для изменения порядка нумерации элементов списка в теге **LI** используется атрибут VALUE.

Пример создания списков различных видов приведен ниже, результат представлен на рис. 5.4.

```
<HTML>
<HEAD><TITLE>Создание списков</TITLE></HEAD>
<BODY>
    <B>Ненумерованный список</B>
    <UL type=square>
        <LI>Элемент 1
        <LI>Элемент 2
        <LI>Элемент 3
        <LI>Элемент 4
    </UL>
```

```

<B>Нумерованный список</B>
<OL type=I start=3>
  <LI>Элемент 1
  <LI>Элемент 2
  <LI value=1>Элемент 3
  <LI>Элемент 4
</OL>
<B>Список определений</B>
<DL>
  <DT>Термин1
  <DD>Определение1
  <DT>Термин2
  <DD>Определение2
</DL>
</BODY>
</HTML>

```

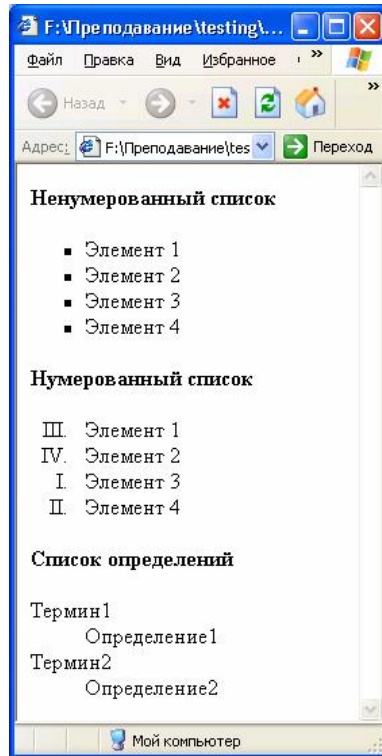


Рис. 5.4. Использование списков на HTML-странице

Порядок выполнения работы

1. Создайте документ **Lab5.html**, содержащий таблицу, соответствующую заданию, выданному преподавателем.
2. Установите название документа, цвет фона и текста.
3. Определите выравнивание таблицы по горизонтали и по вертикали.
4. Определите ширину и высоту таблицы.
5. Определите цвет фона ячеек таблицы.
6. Добавьте на созданную html-страницу списки в соответствии с заданием, выданным преподавателем.

Контрольные вопросы

1. Какие теги создания таблиц вы знаете?
2. Чем отличаются теги **<TD>** и **<TH>**?
3. Какие атрибуты могут использоваться для тегов **<TR>** и **<TD>**?
4. Как можно определить выравнивание таблицы?
5. Какие виды списков вы знаете?
6. Какие теги используются для создания упорядоченных списков?
7. Какие теги используются для создания неупорядоченных списков?
8. Как определить стиль нумерации?
9. Как изменить порядок нумерации?
10. Можно ли создавать вложенные списки?
11. Для чего используются списки определений?
12. Какие теги используются для создания списков определений?

Отчетность по лабораторной работе

1. Файл **Lab5.html** на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - листинг файла **Lab5.html** с комментариями, поясняющими назначение используемых тегов и их атрибутов;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 6

ИСПОЛЬЗОВАНИЕ СЛОЕВ И ИЗОБРАЖЕНИЙ-КАРТ

Цель работы: формирование практических умений использования слоев и изображений-карт.

Теоретические сведения

Слой

Для создания слоев может использоваться тег **<LAYER>**. Однако данный тег поддерживается только браузером Netscape Navigator. Более универсальным способом создания слоев является сочетание использования 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
```

Можно определить следующие свойства слоя (табл. 6.1):

Т а б л и ц а 6.1

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

Пример создания слоев приведен ниже, результат – на рис 6.1.

```
<DIV style="position:absolute; top:350;
left:10; width:180; height:50;
background:yellow; color:blue; z-index:1">
    <H2 align="center">Планета</H2>
</DIV>
<DIV style="position:absolute; top:0;
left:0; z-index:0">
    <IMG src="Planet.jpg" width="400" height="400"
border="0">
</DIV>
```

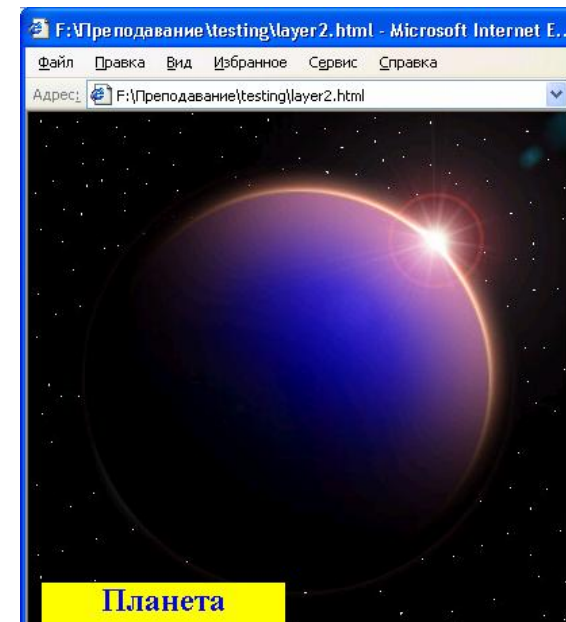


Рис. 6.1. Слои на HTML-странице

Изображения-карты

Различают два вида изображений-карт (карт ссылок):

- обрабатываемые сервером;
- обрабатываемые клиентом.

Для определения изображения-карты, обрабатываемой клиентом, используются теги **<MAP>** и **<AREA>** (рис. 6.2).

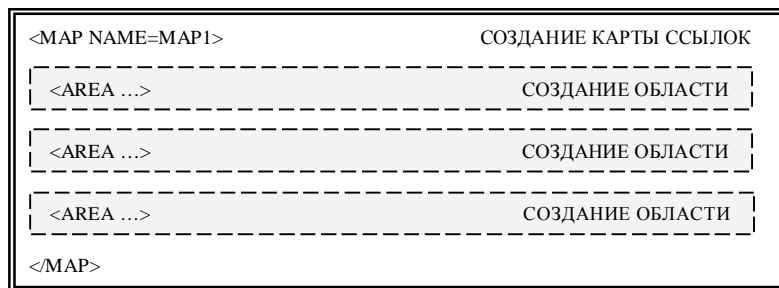


Рис. 6.2. Структура карты ссылок

Атрибуты тега **<AREA>** [8] приведены в табл. 6.2.

Т а б л и ц а 6.2

| Атрибут | Назначение |
|---------|---|
| SHAPE | Форма области (rect, circle, poly) |
| COORDS | Координаты области на карте |
| HREF | URL, на который ссылается область |
| NOHREF | Определяет область как неактивную (является атрибутом-флагом) |
| ALT | Альтернативный текст-подсказка для данной области |

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

Например:

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

Пример определения и использования карты ссылок приведен ниже, результат – на рис. 6.3.

```
<MAP NAME="rectangles">
  <AREA shape="rect" coords="95,55,295,185"
        alt="orange" href="2.html">
  <AREA shape="rect" coords="0,0,175,120"
        alt="yellow" href="1.html">
  <AREA>
</MAP>
<IMG src="Kard.gif"
      border="0" USEMAP="#rectangles">
```

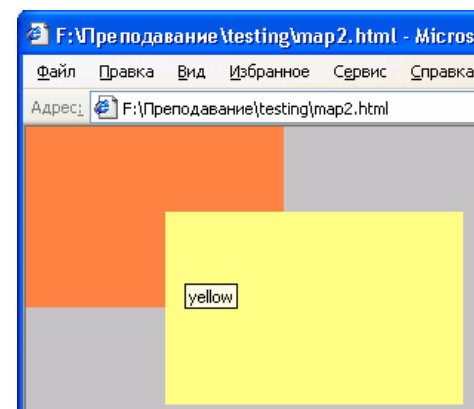
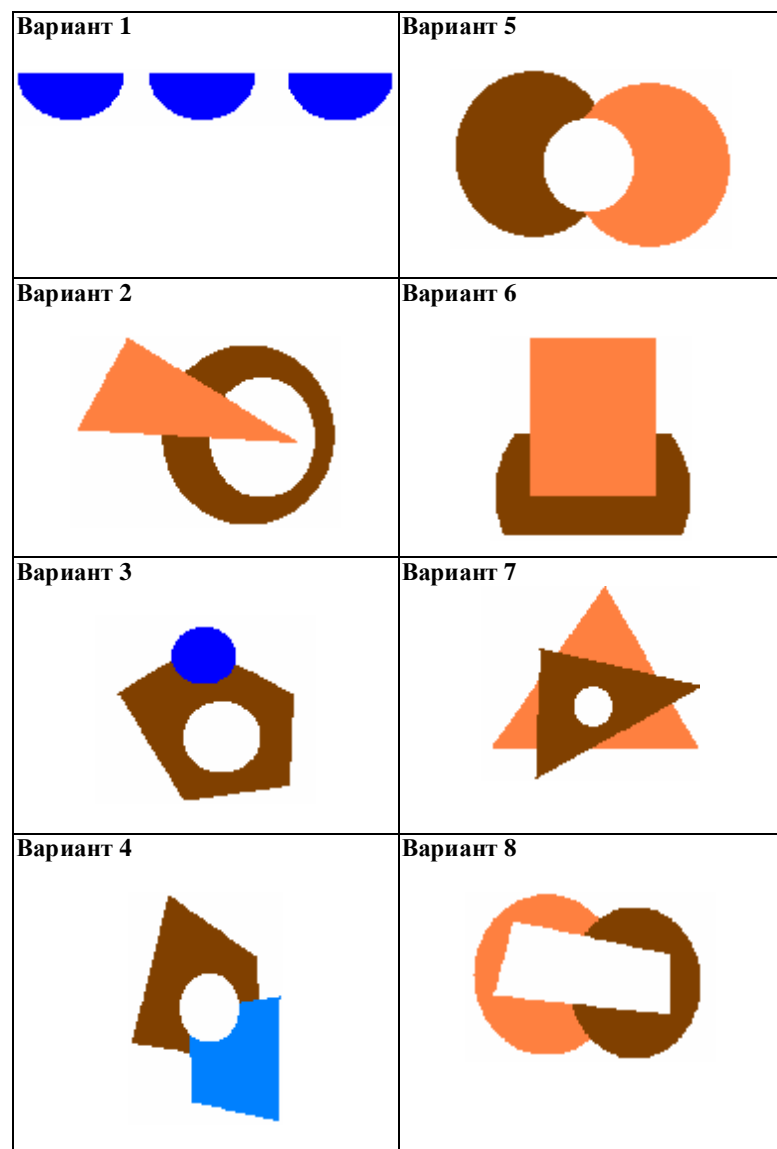


Рис. 6.3. Созданная карта ссылок

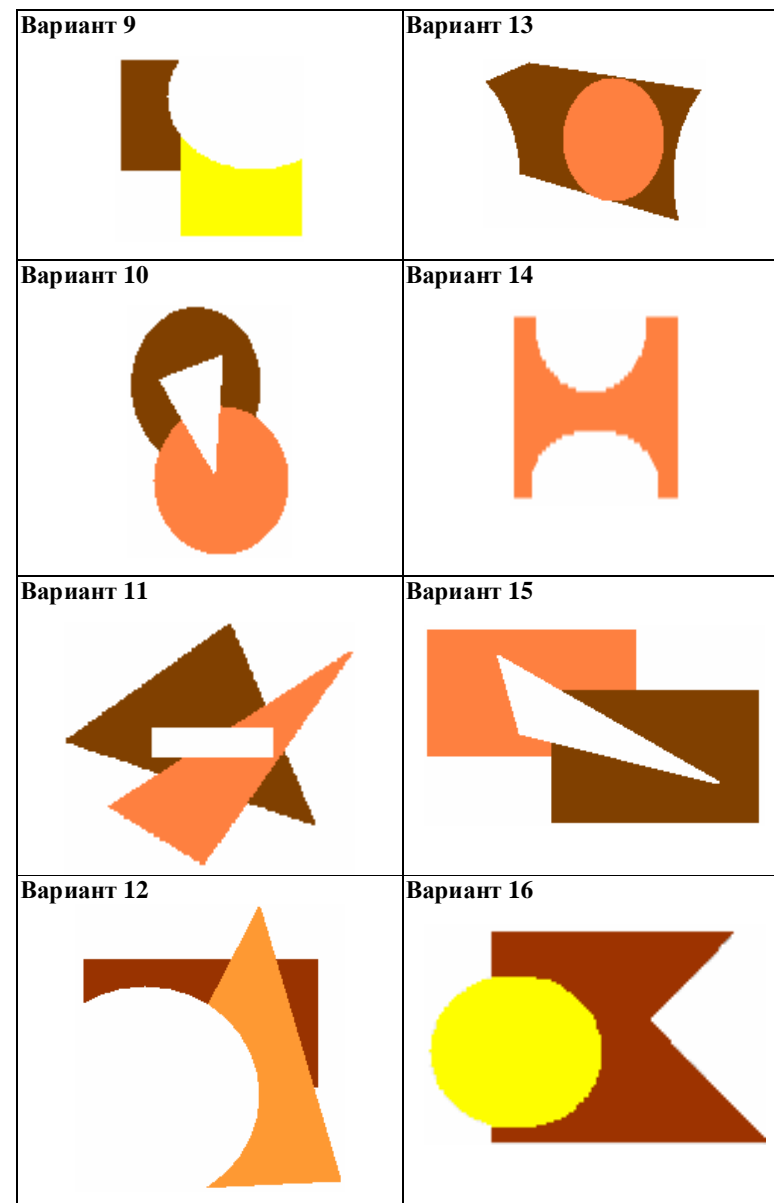
Порядок выполнения работы

1. Создайте документ **Lab6_layer.html**, определите его название, цвет фона и текста.
2. Создайте слой, содержащий графическое изображение, и расположите его в верхнем левом углу окна браузера.
3. Создайте слой, содержащий текст, и установите координаты верхнего левого угла слоя, равными 0,50.
4. Установите ширину и высоту слоя с текстом, определите цвет фона и текста.
5. Последовательно измените значение свойства **position** созданных слоев, сравните и проанализируйте полученные результаты.
6. Измените порядок наложения слоев таким образом, чтобы сначала слой с изображением перекрывал текстовый, а затем наоборот.
7. Установите фоновое изображение для слоя с текстом.
8. Создайте документ **Lab6_map.html**, содержащий карту ссылок, соответствующую вашему варианту (табл. 6.3).
Примечание. Области, ссылающиеся на различные документы, выделены разными цветами. Области, закрашенные белым цветом, не являются ссылками.
9. Определите текстовые подсказки для каждой области, используя атрибут **ALT**.

Т а б л и ц а 6.3



Окончание табл. 6.3



Контрольные вопросы

1. Назовите преимущества использования слоев.
2. Перечислите способы создания слоев. В чем их отличие?
3. Перечислите свойства слоя.
4. Как определяется порядок наложения слоев?
5. Какие значения может принимать свойство position? В чем их отличия?
6. Дайте определение понятия «изображение-карта».
7. Какие виды изображений-карт вы знаете?
8. Как создаются изображения-карты, обрабатываемые клиентом?
9. Перечислите атрибуты тега **<AREA>**.
10. Какой атрибут указывает, что изображение является картой ссылок?

Отчетность по лабораторной работе

1. Файлы *Lab6_layer.html* и *Lab6_map.html* на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - листинги файлов *Lab6_layer.html* и *Lab6_map.html* с комментариями, поясняющими назначение используемых тегов и их атрибутов;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 7

СОЗДАНИЕ СТРАНИЦ

С ИСПОЛЬЗОВАНИЕМ ФРЕЙМОВ

Цель работы: формирование практических умений использования фреймов при создании HTML-страниц.

Теоретические сведения

Фреймы используются для разделения окна браузера на несколько областей, в каждую из которых загружается отдельный HTML-документ.

В файле с фреймами не указываются теги **<BODY>** **</BODY>**.

Фреймовая структура создается следующим образом (рис. 7.1):

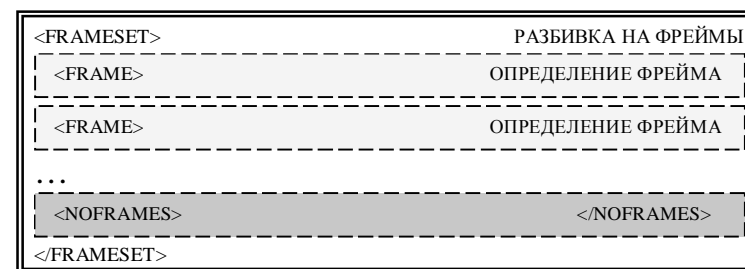


Рис. 7.1. Фреймовая структура

Контейнер **<NOFRAMES>** **</NOFRAMES>** обеспечивает альтернативную информацию для браузеров, не работающих с фреймами.

Атрибуты тега **<FRAMESET>** [8] приведены в табл. 7.1.

Т а б л и ц а 7.1

| Атрибут | Назначение |
|-------------|--|
| ROWS | Определяет количество и размеры горизонтальных фреймов в окне браузера |
| COLS | Определяет количество и размеры вертикальных фреймов в окне браузера |
| FRAMEBORDER | Определяет наличие рамок у содержащихся внутри FRAMESET фреймов |
| BORDER | Ширина рамок фреймовой структуры |
| BORDERCOLOR | Цвет рамок фреймовой структуры |

Атрибуты тега **<FRAME>** [8] приведены в табл. 7.2.

Т а б л и ц а 7.2

| Атрибут | Назначение |
|--------------|---|
| SRC | Указывает адрес HTML-файла, загружаемого в данный фрейм |
| NAME | Определяет имя фрейма |
| MARGINWIDTH | Определяет ширину левого и правого полей фрейма |
| MARGINHEIGHT | Определяет ширину верхнего и нижнего полей фрейма |
| SCROLLING | Определяет наличие полос прокрутки (YES NO AUTO) |
| NORESIZE | Не позволяет изменять размеры фрейма |
| FRAMEBORDER | Определяет наличие рамок у фрейма |

Фреймы могут быть вложенными друг в друга.

Рассмотрим пример создания вложенных фреймов. Создадим три HTML-документа *header.html*, *menu.html* и *info.html*, каждый из которых будет отображаться в отдельном фрейме.

Листинг документа **header.html**:

```
<HTML>
<HEAD> <TITLE> header.html </TITLE> </HEAD>
<BODY>
<H1>Использование фреймов в HTML</H1>
</BODY>
</HTML>
```

Листинг документа **menu.html**:

```
<HTML>
<HEAD> <TITLE> menu.html </TITLE> </HEAD>
<BODY>
<BR>
<UL>
  <LI>Ter FRAMESET<BR>
  <LI>Ter FRAME<BR>
</UL>
</BODY>
</HTML>
```

Листинг документа **info.html**:

```
<HTML>
<HEAD> <TITLE> info.html </TITLE> </HEAD>
<BODY>
<BR><BR><BR>
<CENTER> <I><B>
Фреймы используются для разделения окна
браузера на несколько областей, в каждую из
которых загружается отдельный HTML-документ.
</B></I> </CENTER>
</BODY>
</HTML>
```

В документе **main.html** определим фреймовую структуру:

```
<HTML>
<HEAD> <TITLE> Фреймы </TITLE> </HEAD>
<FRAMESET ROWS="30%,70%">
  <FRAME SRC="header.html">
    <FRAMESET COLS="30%,70%">
      <FRAME SRC="menu.html">
      <FRAME SRC="info.html">
    </FRAMESET>
  </FRAMESET>
</FRAMESET>
<NOFRAMES>
Ваш браузер не показывает фреймы.
</NOFRAMES>
</HTML>
```

Внешний вид документа **main.html** представлен на рис. 7.2

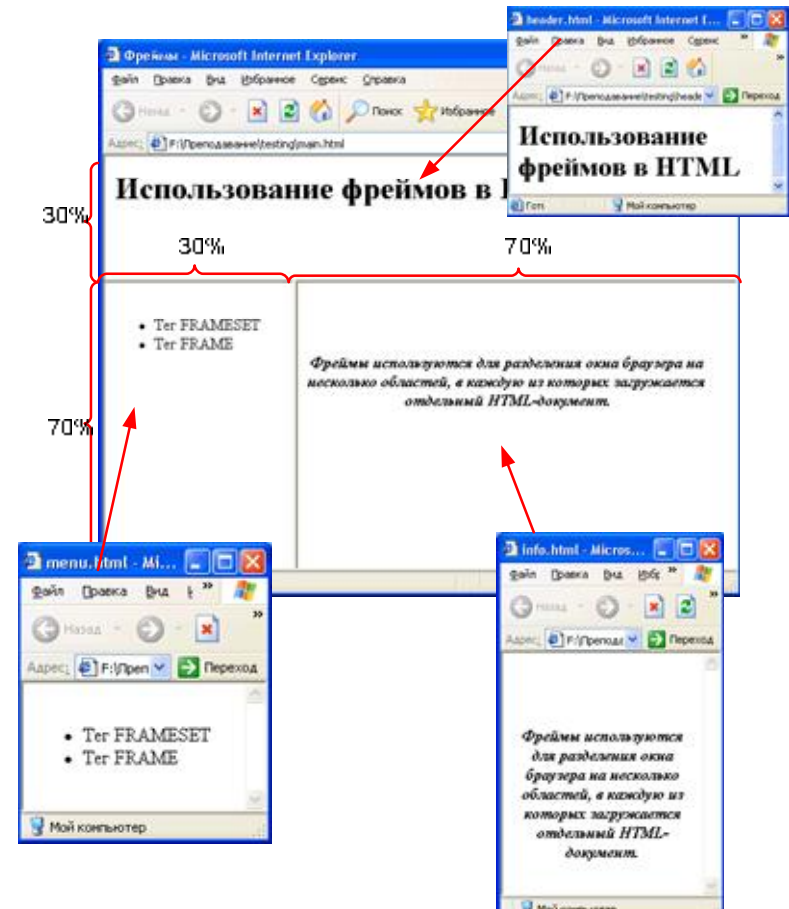


Рис. 7.2. Фреймовая структура документа **main.html**

Изменим документ **menu.html**: преобразуем элементы списка в ссылки на соответствующие документы **frame.html** и **frameset.html**:

```
<HTML>
<HEAD> <TITLE> menu.html </TITLE> </HEAD>
<BODY>
<BR>
<UL>
```

```
<LI><A href="frameset.html">Ter FRAMESET</A><br>
<LI><A href="frame.html">Ter FRAME</A><br>
</UL>
</BODY>
</HTML>
```

Листинг документа **frame.html**:

```
<html>
<head> <title> frame.html </title> </head>
<body>
  <H3> Атрибуты тега <b>FRAME</b></H3>
</body>
</html>
```

Листинг документа **frameset.html**:

```
<html>
<head> <title> frameset.html </title></head>
<body>
<H3>Атрибуты тега <b>FRAMESET</b></H3>
</body>
</html>
```

В этом случае при выборе одной из ссылок соответствующий документ будет загружен в левый нижний фрейм, т. е. заменит документ **menu.html**.

Чтобы определить окно, в которое будет загружаться документ, необходимо выполнить следующие действия:

- 1) используя атрибут **NAME**, определить имя фрейма, в который нужно загрузить документ;
- 2) в теге **<A>** указать атрибут **TARGET** и присвоить ему значение, соответствующее имени фрейма.

Определим имя **information** для правого нижнего фрейма. Измененный документ **main.html** представлен ниже:

```
<HTML>
<HEAD> <TITLE> Фреймы </TITLE> </HEAD>
<FRAMESET ROWS=" 30%,70%" >
  <FRAME SRC="header.html">
    <FRAMESET COLS=" 30%,70%">
      <FRAME SRC="menu.html">
        <FRAME SRC="info.html" NAME="information">
      </FRAMESET>
    </FRAMESET>
  </FRAMESET>
</NOFRAMES>
Ваш браузер не показывает фреймы. </NOFRAMES>
</HTML>
```

Изменим описание ссылок в документе **menu.html**:

```
<LI><A href="frameset.html" TARGET="information">
  Тер FRAMESET</A><br>
<LI><A href="frame.html" TARGET="information">
  Тер FRAME</A><br>
```

Переход по ссылкам демонстрируется на рис. 7.3.

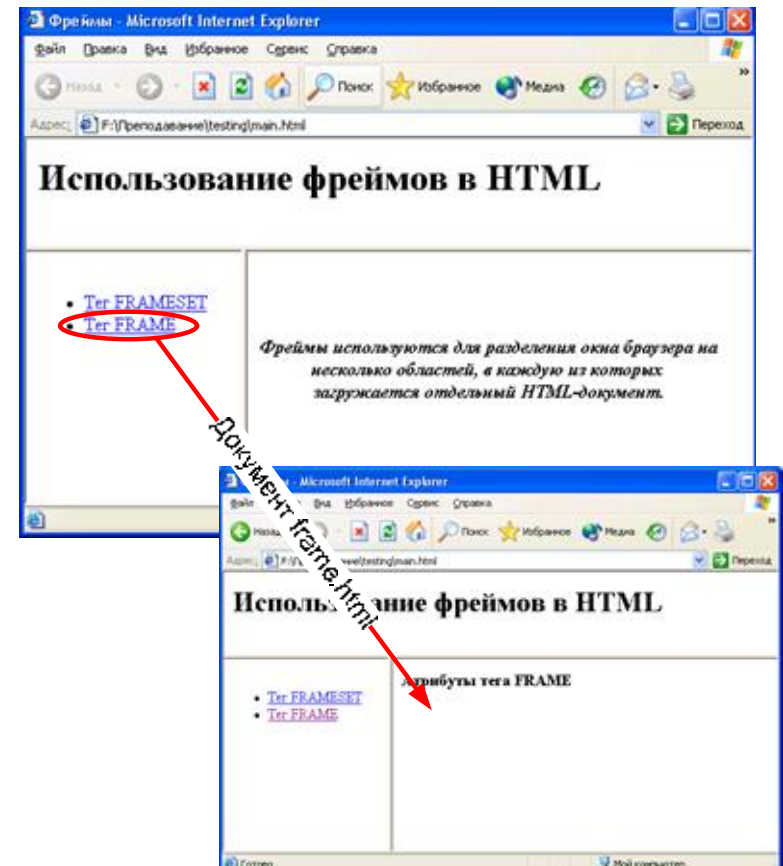


Рис. 7.3. Демонстрация перехода по ссылке в документе **main.html**

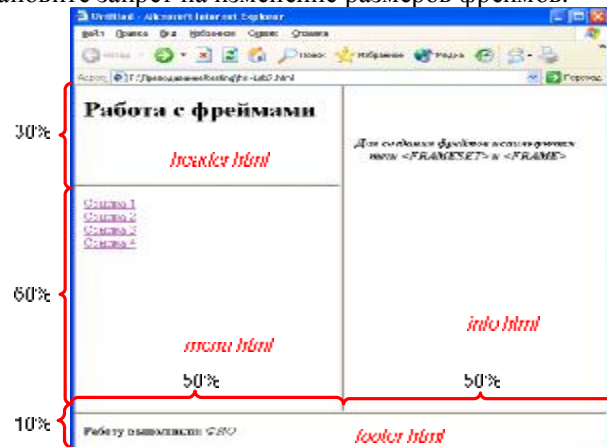
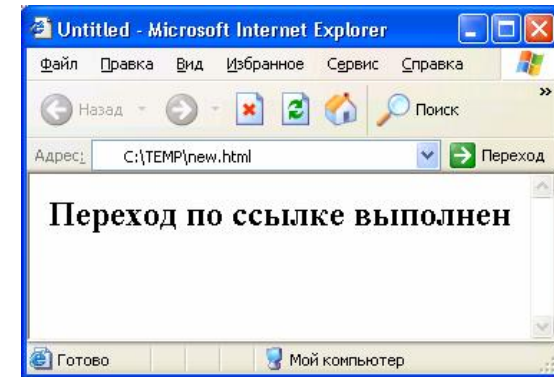
Значением атрибута **TARGET** может быть одно из неявных имен фреймов. В HTML зарезервированы четыре неявных имени [2], начинающихся с символа подчеркивания (табл. 7.3).

Т а б л и ц а 7.3

| Неявное имя | Назначение |
|---------------|---|
| blank | Загружает документ в новое окно |
| self | Загружает документ в текущее окно (установлено по умолчанию) |
| parent | Загружает документ в окно родительской фреймовой структуры |
| top | Загружает документ в окно фреймовой структуры верхнего уровня по отношению к данному фрейму |

Порядок выполнения работы

1. Создайте документ **Lab7.html** с фреймовой структурой, представленной на рис. 7.4.
2. Создайте документ **new.html** (рис. 7.5), который будет загружаться:
 - при выборе **Ссылки 1** в правый фрейм;
 - при выборе **Ссылки 2** в текущий фрейм;
 - при выборе **Ссылки 3** в новое окно;
 - при выборе **Ссылки 4** в текущее окно браузера, заменяя фреймовую структуру.
3. Установите ширину и цвет рамок фреймов.
4. Установите ширину полей нижнего фрейма.
5. Установите значение атрибута FRAMEBORDER, равным NO, затем измените на YES; сравните полученные результаты.
6. Добавьте полосы прокрутки для правого фрейма.
7. Измените размеры фреймов путем перетаскивания их границ. Установите запрет на изменение размеров фреймов.

Рис. 7.4. Фреймовая структура документа **Lab7.html**Рис. 7.5. Документ **new.html**

Контрольные вопросы

1. Для чего используются фреймы?
2. Какой тег используется для определения фреймовой структуры?
3. Какой тег используется для определения конкретного фрейма?
4. Возможно ли использование тега **<BODY>** в документах, определяющих фреймовую структуру? Какие могут быть последствия?
5. Перечислите атрибуты тега **<FRAME>** и укажите их назначение.
6. Перечислите атрибуты тега **<FRAMESET>** и укажите их назначение.
7. Какой атрибут тега **<A>** используется для указания окна, в которое необходимо загружать документ?
8. Перечислите известные вам неявные имена фреймов.

Отчетность по лабораторной работе

1. Файлы **header.html**, **menu.html**, **info.html**, **footer.html** и **Lab7.html** на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - листинги файлов **menu.html** и **Lab7.html** с комментариями, поясняющими назначение используемых тегов и их атрибутов;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 8 РАБОТА С ФОРМАМИ

Цель работы: формирование практических умений создания форм на HTML-странице.

Теоретические сведения

Формы используются для получения определенной информации от пользователя с целью ее последующей обработки.

Для создания форм используется парный тег **<FORM>** **</FORM>**.

Тег **<FORM>** может содержать следующие атрибуты [2] (табл. 8.1):

Т а б л и ц а 8.1

| Атрибут | Назначение |
|---------|--|
| NAME | Имя формы |
| ACTION | URL, по которому следует передать введенную информацию для последующей обработки |
| METHOD | Метод передачи данных из формы |

Для определения элементов формы могут использоваться следующие теги (табл. 8.2):

Т а б л и ц а 8.2

| Тег | Назначение |
|--|------------------------------------|
| <TEXTAREA> </TEXTAREA> | Многострочное текстовое поле ввода |
| <INPUT> | Поля ввода |
| <SELECT> </SELECT> | Меню-список |

Тег **<TEXTAREA>** [6] имеет следующие атрибуты (табл. 8.3):

Т а б л и ц а 8.3

| Атрибут | Назначение |
|---------|------------------------------|
| NAME | Имя поля ввода |
| ROWS | Число строк в поле ввода |
| COLS | Ширина поля ввода в символах |

Тег **<INPUT>** имеет следующие атрибуты [2, 6] (табл. 8.4):

Т а б л и ц а 8.4

| Атрибут | Назначение |
|-----------|--|
| CHECKED | Элемент формы CHECKBOX или RADIO будет отмечен |
| SIZE | Размер поля ввода в символах |
| MAXLENGTH | Количество символов, которое можно ввести в поле ввода |
| NAME | Имя поля ввода |
| SRC | Указывает путь к изображению (используется вместе со значением IMAGE атрибута TYPE) |
| VALUE | Устанавливает текст по умолчанию для поля ввода текста или пароля. Для флажка или переключателя указывает значение, возвращаемое серверу в случае выбора флажка или переключателя. Для кнопок определяет надпись |
| TYPE | Определяет тип поля ввода; по умолчанию создается однострочное текстовое поле ввода |

Возможные значения атрибута TYPE [2] (табл. 8.5):

Т а б л и ц а 8.5

| Атрибут | Назначение |
|----------|---|
| CHECKBOX | Флажок; может принимать значение <i>ON</i> (отмечен) или <i>OFF</i> (не отмечен) |
| HIDDEN | Скрытое поле |
| IMAGE | Изображение |
| TEXT | Однострочное поле ввода |
| PASSWORD | Модифицированное текстовое поле (при вводе текста вместо символов отображаются звездочки) |
| RADIO | Переключатель (используется для выбора одного варианта из нескольких)* |
| RESET | Кнопка, при нажатии на которую поля формы принимают значения по умолчанию |
| SUBMIT | Кнопка отправки данных |
| BUTTON | Кнопка, определенная пользователем (т. е. конкретного действия за данной кнопкой не закреплено, оно задается пользователем) |
| FILE | Поле ввода и кнопка «Обзор» для поиска файла на диске |

* Если переключателям даны одинаковые имена (в атрибуте NAME), они объединяются в группу; из группы переключателей можно выбрать только один.

Тег **<SELECT>** имеет следующие атрибуты [8] (табл. 8.6):

3. Создайте регистрационную форму, содержащую:
 - текстовое поле для ввода имени;
 - текстовое поле для ввода фамилии;
 - поле для ввода пароля;
 - меню-список для выбора страны проживания;
 - группу выпадающих списков для указания даты рождения;
 - переключатель для указания пола;
 - переключатели-флажки для указания интересов;
 - кнопку очистки формы;
 - кнопку отправки данных формы;
 - многострочное текстовое поле для ввода примечания.
4. Установите надписи на кнопках.
5. В поле для примечания укажите текст по умолчанию.
6. Для полей ввода имени и фамилии укажите размер поля и максимальное количество вводимых символов.
7. Добавьте на страницу графическое изображение, используя значение IMAGE атрибута TYPE.

Контрольные вопросы

1. Для чего используются формы?
2. Какой тег используется для создания формы?
3. Перечислите атрибуты тега **<FORM>**.
4. Перечислите известные вам теги определения отдельных элементов формы?
5. Какие атрибуты тега **<TEXTAREA>** вы знаете?
6. Перечислите атрибуты тега **<INPUT>**.
7. Перечислите возможные значения атрибута TYPE тега **<INPUT>**.
8. Как можно создать выпадающий список значений?
9. Для чего используется атрибут MULTIPLE?
10. При помощи какого тега определяется отдельный элемент меню-списка?

Отчетность по лабораторной работе

1. Файл **Lab8.html** на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - листинг файла **Lab8.html** с комментариями, поясняющими назначение используемых тегов и их атрибутов;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 9

СОЗДАНИЕ И ВНЕДРЕНИЕ В WEB-СТРАНИЦЫ ТАБЛИЦ СТИЛЕЙ

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

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;
- 2) встраивание;
- 3) оперативное определение;
- 4) импорт.

Связывание – это установка связи HTML-документа с таблицей стилей, хранящейся в отдельном файле с расширением css. Для связывания используется тег **<LINK>**.

Например:

```
<LINK REL=STYLESHEET
      HREF="http://www.myserver.com/mysheet.css"
      TYPE="text/css">
```

Таблицу стилей можно определять не только в отдельном файле, но и в документе, в котором она будет применяться. Включение таблицы стилей в документ называется *встраиванием* (используется тег-контейнер **<STYLE>**). Описание стилей размещается между тегами **<HTML>** и **<BODY>**.

Оперативное определение стиля используется, если нужно определить свойства для конкретного фрагмента HTML-документа, отличные от установленных по умолчанию для всего документа. Новые свойства указываются в атрибуте STYLE тега, для которого определяются параметры оформления.

Например:

```
<H1 STYLE="color: blue">
```

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>
```

Каждое определение стилей называется *правилом (rule)* [2].
Формат правила CSS следующий:

селектор{свойство1:значение1; свойство2: значение2;...}

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> [2]
```

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 [7].

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

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 будет изменяться на зеленый [2].

Основные свойства CSS приведены в приложении А.

Пример использования CSS в документе *Style.html* приведен ниже, результат – на рис. 9.1.

```
<HTML>
<HEAD>
<TITLE> CSS </TITLE>
<STYLE type="text/css">
  H2{color:white; background-color:#0099FF}
  H3{color:white; background-color: #cc0000}
</STYLE>
</HEAD>
<BODY>
<H2>Заголовок побольше на синем фоне</H2>
<H3>Заголовок поменьше на красном фоне</H3>
<P STYLE="font-style:italic;
      text-transform:uppercase; color:#cc0000">
Курсив красного цвета, все буквы заглавные</P>
<P STYLE="font-weight:bold;
      text-transform:lowercase; color:#0099ff">
Жирный шрифт синего цвета, все буквы строчные</P>
<P align="center" STYLE="border-width:medium;
      border-color:#0099ff; border-style:solid;
      color:#cc0000">
Текст в рамке</P>
</BODY>
</HTML>
```

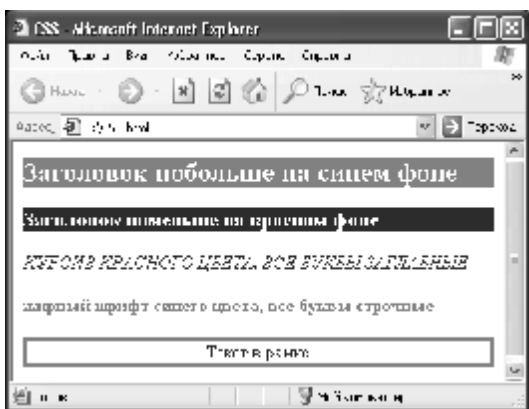


Рис. 9.1. Документ *style.html* в Internet Explorer

Порядок выполнения работы

1. Создайте документ *Lab9.html*, определите его название.
2. В файле *style.css* определите цвет фона и текста документа, свойства полос прокрутки.
3. Подключите файл *style.css* к документу *Lab9.html* (методом связывания).
4. Получите вариант задания у преподавателя.
5. Используя тег `<STYLE>`, определите стиль отдельных тегов в соответствии с вариантом, а также необходимые классы.
Примечание. При выполнении данной работы нельзя использовать теги физического форматирования.
6. Наберите текст задания, используя теги логического форматирования.
7. Для изменения свойств отдельных фрагментов текста используйте метод оперативного определения.
8. В файле *style.css* определите цвет непосещенной, активной, посещенной ссылок, а также цвет ссылки при наведении курсора мыши.
9. Создайте документ *Lab9_2.html* и скопируйте в него содержимое документа *Lab9.html*.
10. В документе *Lab9_2.html* для подключения файла, содержащего стилиевые правила, используйте импортирование.
11. Добавьте в *Lab9.html* ссылку на *Lab9_2.html*.
12. Измените вид курсора для элемента H1 в документе *Lab9.html*.

13. Установите фоновое изображение для документа *Lab9_2.html*, определите свойства данного фонового изображения (background-attachment и т. д.).
14. Добавьте на страницу *Lab9.html* изображение. Создайте в *style.css* стилевое правило для элемента IMG, определяющее размеры изображения и его положение на странице.

Контрольные вопросы

1. Для чего используется CSS?
2. Перечислите способы использования каскадных таблиц стилей. Укажите преимущества и недостатки каждого из них.
3. Укажите формат правила CSS.
4. Что такое селектор?
5. Что такое классы в CSS? Укажите их назначение.
6. Как определить и применить класс?
7. Как создать класс, не связанный с определенным тегом?
8. Что такое псевдокласс?

Отчетность по лабораторной работе

1. Файлы *style.css*, *Lab9.html* и *Lab9_2.html* на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - листинги файлов *style.css*, *Lab9.html* и *Lab9_2.html* с комментариями;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 10 СОЗДАНИЕ DHTML-СТРАНИЦ

Цель работы: получение практических умений создания документов, используя DHTML.

Теоретические сведения

Динамический HTML (DHTML) представляет собой комбинацию языка гипертекстовой разметки HTML, стилей и сценариев, используемую для динамического изменения содержания и оформления HTML-страницы, создания интерактивных документов, взаимодействующих с пользователем [2].

Можно выделить две группы сценариев: клиентские и серверные.

Клиентские сценарии выполняются на компьютере-клиенте. Многие браузеры (IE, Opera и т. д.) используют встроенный интерпретатор для выполнения клиентских сценариев.

В рамках курса «Интернет-программирование» будет рассматриваться язык программирования на стороне клиента JavaScript.

Механизм DHTML связан с обработкой событий. Существуют следующие категории событий:

- стандартные события (могут быть связаны с любым элементом HTML);
 - события для элементов, получающих фокус ввода;
 - события, специфические для отдельных элементов.
- Стандартные события приведены в табл. 10.1.

Т а б л и ц а 10.1

| Событие | Действие |
|--------------------|--|
| onclick | Щелчок левой кнопкой мыши по области элемента |
| ondblclick | Двойной щелчок левой кнопкой мыши по области элемента |
| onmousedown | Левая кнопка мыши нажата, когда указатель находится в области элемента |
| onmousemove | Указатель мыши перемещается по области элемента |
| onmouseup | Левая кнопка мыши отпущена, когда указатель находится в области элемента |
| onmouseover | Указатель мыши входит в пределы области элемента |
| onmouseout | Указатель мыши выходит за пределы элемента |
| onkeydown | Нажата и удерживается клавиша на клавиатуре |
| onkeypress | Нажата и отпущена клавиша на клавиатуре |
| onkeyup | Отпущена клавиша на клавиатуре |
| onscroll | Элемент прокручивается |

События для элементов, получающих фокус ввода, приведены в табл. 10.2:

Т а б л и ц а 10.2

| Событие | Действие |
|----------------|--|
| onenter | Произошел переход к элементу |
| onfocus | Устанавливается фокус ввода для элемента |
| onexit | Произошел выход из области элемента |
| onblur | Снимается фокус с элемента |

События, используемые для отдельных элементов, приведены в табл. 10.3.

Т а б л и ц а 10.3

| Событие | Действие |
|-----------------|---|
| onabort | Пользователь прерывает прием изображения |
| onchange | Объект изменяется при вводе данных пользователем (для элементов формы <INPUT type=file text password radio checkbox> , <SELECT> , <TEXTAREA>) |
| onload | Элемент (<BODY> или) полностью загружен |
| onreset | Нажата кнопка Reset, т. е. пользователь очистил форму |
| onselect | Отпущена кнопка мыши при выделении текста пользователем |
| onsubmit | Нажата кнопка Submit, т. е. форма отправлена на обработку |
| onunload | Документ начал выгружаться |

Для того чтобы связать объект с обработчиком события, нужно в теге, определяющем данный объект, указать атрибут, соответствующий событию, и в качестве значения записать программный код (например, на JavaScript) или имя функции.

Например:

```
<input type=button value="Click here"
      onClick="window.alert('Hello')">
```

При нажатии на кнопку будет появляться окно сообщения с текстом Hello (рис. 10.1).



Рис. 10.1. Окно сообщения

Используя DHTML, можно изменять оформление элемента при возникновении определенных событий.

Например:

```
<H1 onmouseover="this.style.color='red'">
Эта строка покраснеет</H1>
```

Ключевое слово **this** указывает на то, что изменяется стиль элемента, в теге которого определен обработчик события, в данном случае элемент H1.

Если обработка события предполагает изменение ряда свойств элемента, можно определить два различных класса и применять их при возникновении событий:

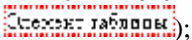
```
<HTML>
<HEAD> </HEAD>
<STYLE>
.red{color:red;font-style:italic}
.green{color:green;font-weight:bold}
</STYLE>
<BODY>
<P onmouseover="this.className='green'"
onmouseout="this.className='red'">
Этот текст будет изменяться при наведении указателя мыши</P>
</BODY>
</HTML>
```

При наведении указателя мыши будет применяться класс **green**, при выходе указателя за пределы элемента – класс **red**.

Порядок выполнения работы

1. Создайте документ **Lab10.html**, содержащий таблицу следующего вида:

| | | | | | |
|----------|----------|----------|----------|----------|----------|
| Элемент1 | Элемент2 | Элемент3 | Элемент4 | Элемент5 | Элемент6 |
|----------|----------|----------|----------|----------|----------|

2. Определите следующие классы:
 - **cl1** – текст синего цвета, шрифт Courier New, 12 px;
 - **cl2** – текст красного цвета, полужирный;
 - **cl3** – текст зеленого цвета, шрифт Times New Roman, курсивное начертание;
 - **cl4** – текст черного цвета, в красной рамке (например, );
 - **cl5** – шрифт синего цвета, полужирный, курсивное начертание.
3. Ко всем элементам таблицы примените класс **cl1**.
4. При наведении указателя мыши на *Элемент1* примените класс **cl2**. Когда указатель мыши выходит за пределы данного элемента таблицы, должен применяться класс **cl5**.
5. При двойном щелчке на *Элемент2* примените класс **cl3**.
6. При одинарном щелчке на *Элемент3* примените класс **cl4**.
7. При наведении курсора мыши на *Элемент4* измените цвет текста на желтый.

8. При двойном щелчке на *Элемент5* измените цвет фона ячейки.
9. При одинарном щелчке на *Элемент6* измените начертание на курсивное.
10. Установите размеры ячеек таблицы таким образом, чтобы при изменении стиля элемента не перемещались границы ячеек.

Контрольные вопросы

1. Поясните суть механизма DHML.
2. Чем отличаются клиентские и серверные сценарии?
3. Какие языки программирования на стороне клиента вы знаете?
4. Перечислите категории событий.
5. Какие стандартные события вы знаете?
6. Перечислите известные вам события для элементов, получающих фокус ввода.
7. Перечислите известные вам события, используемые для отдельных элементов.
8. Как изменить стиль элемента при возникновении определенного события?

Отчетность по лабораторной работе

1. Файл **Lab10.html** на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - листинг файла **Lab10.html** с комментариями;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 11

СОЗДАНИЕ ПРОСТЕЙШИХ СКРИПТОВ И ВНЕДРЕНИЕ ИХ В HTML-ДОКУМЕНТ

Цель работы: формирование практических умений создания простейших скриптов на языке программирования JavaScript и добавления их на HTML-страницу.

Теоретические сведения

Сценарии (скрипты) JavaScript размещаются непосредственно в коде HTML-страницы.

Включение кода JavaScript в состав HTML-документа может осуществляться при помощи тега **<SCRIPT>**. Рассмотрим на

примере варианты использования данного тега. Создадим HTML-документ *pr11_1.html*:

```
<HTML>
<HEAD><TITLE>Использование JavaScript</TITLE></HEAD>
<BODY>
<P>Обычный текст HTML-документа</P>
<SCRIPT language="JavaScript">
document.write("<b>Текст, созданный сценарием");
document.write("JavaScript</b>");
</SCRIPT>
</body>
</html>
```

Полученный HTML-документ представлен на рис. 11.1.

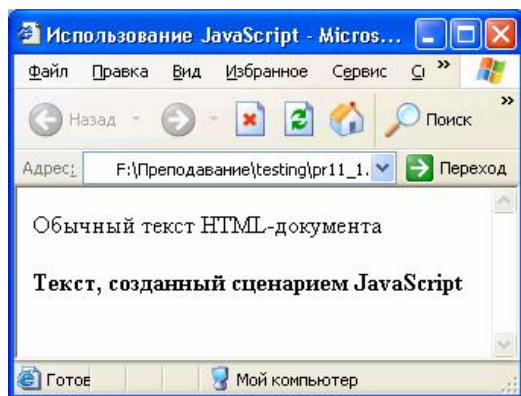


Рис.11.1. Документ *pr11_1.html* в Internet Explorer

Атрибут **language** в данном случае можно не указывать, он по умолчанию принимает значение **JavaScript**.

Код JavaScript также можно разместить в отдельном файле с расширением *.js* и подключить его к основному HTML-документу. Для этого в теге **<SCRIPT>** указывается атрибут **src** со значением, соответствующим адресу файла со сценарием JavaScript.

Например:

```
<SCRIPT SRC="Text.js"> </SCRIPT>
```

Примечание. В JavaScript учитывается регистр букв, т. е. **document.write("Hello")** \neq **Document.Write("Hello")**. Во втором случае будет выдана ошибка.

Для управления HTML-страницами и их элементами JavaScript использует объектную модель документов DOM (Document Object Model). Упрощенная схема объектной модели представлена на рис. 11.2.

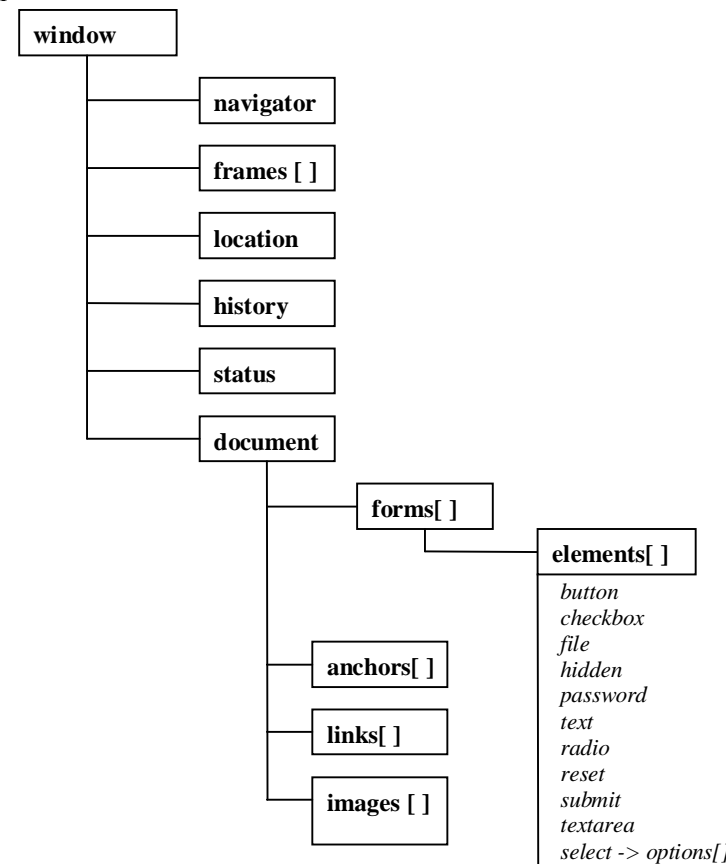


Рис. 11.2. Объектная модель документа

Объекты появляются после загрузки документа браузером или как результат работы программы. Каждый объект характеризуется набором методов, свойств и возможных событий.

Свойства объектов соответствуют атрибутам определенных тегов; их использование позволяет получить или изменить характеристики окна браузера, загруженных документов и элемен-

тов HTML-страницы. Например, к свойствам изображения относятся **src**, **width**, **height** и т. д.

Методы объекта представляют собой действия, которые могут быть выполнены по отношению к данному объекту. Например, одним из методов объекта **document** является **write**, который помещает в документ определенный текст.

Создадим документ *pr_DOM.html*:

```
<HTML>
<HEAD> <TITLE> DOM </TITLE> </HEAD>
<BODY>
<FORM name=Sum>
Первое число <INPUT type="text" name="number1"><BR>
Второе число <INPUT type="text" name="number2"><BR>
<INPUT type="button" name="but"
                value="Вычислить сумму"><BR><BR>
Сумма <INPUT type="text" name="result">
</FORM>
</BODY>
</HTML>
```

Результат создания документа *pr_DOM.html* приведен на рис. 11.3.

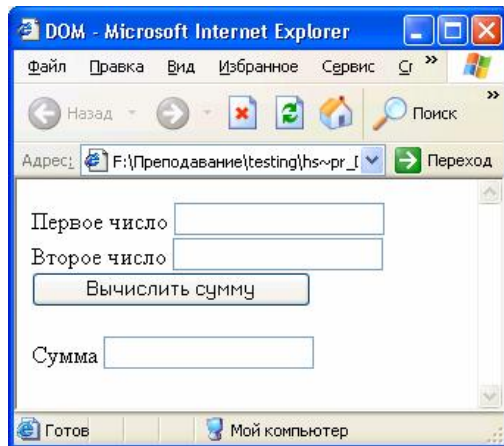


Рис. 11.3. Документ *pr_DOM.html* в Internet Explorer

При открытии данного документа создается объектная модель, представленная на рис. 11.4.

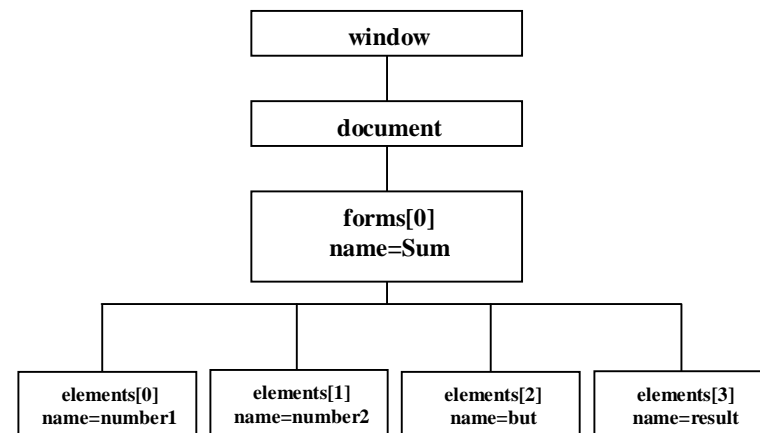


Рис. 11.4. Иерархия объектов документа *pr_DOM.html*

Изменим описание кнопки на странице следующим образом:

```
<input type="button" name="but"
value=" Вычислить сумму "
onClick="document.forms[0].elements[3].value=
parseInt(document.forms[0].elements[0].value)+
parseInt(document.forms[0].elements[1].value);">
```

В результате изменений при нажатии кнопки значения, введенные пользователем в два первых текстовых поля, будут переводиться в числовой тип данных (функцией **parseInt**) и суммироваться; результат будет помещен в третье текстовое поле.

Объект **window** можно не указывать (как в данном примере), если имеется в виду текущее окно браузера.

Для обращения к объектам страницы можно также использовать имя, указанное в атрибуте **name**.

Например, обработку нажатия кнопки можно определить следующим образом:

```
<input type="button" name="but"
value=" Вычислить сумму "
onClick="document.Sum.result.value=
parseInt(document.Sum.number1.value)+
parseInt(document.Sum.number2.value);">
```

Для расчета суммы можно определить отдельную функцию

summa() и вызвать ее при помощи атрибута **onClick**:

```
<html>
<head>
  <title> DOM </title>
  <script>
    // определяем функцию summa()
    function summa(){
      // объявляем переменные n1 и n2
      var n1=parseInt(document.Sum.number1.value);
      var n2=parseInt(document.Sum.number2.value);
      /*рассчитываем сумму и записываем ее
        в поле result*/
      document.Sum.result.value=n1+n2;
    }
  </script>
</head>
<body>
<form name=Sum>
  ...
  <input type="button" name="but"
    value="Вычислить сумму" onClick="summa()">
  ...
</form>
</body>
</html>
```

Работа с окнами

Для открытия нового окна используется метод **open** объекта **window**.

```
Например:
My_window=window.open("doc.html","wind",
  "width=250,height=100,status=no,
  toolbar=no,menubar=no,scrollbars=no");
```

В результате выполнения данного кода будет открыто окно, представленное на рис. 11.5.



Рис. 11.5. Внешний вид открытого окна

My_window является переменной, с помощью которой можно получить доступ к открытому окну (например, для того чтобы закрыть его). В новое окно с именем **wind** будет загружен документ **doc.html**. Третий параметр метода **open** определяет характеристики окна.

Возможные свойства окна приведены в табл. 11.1

Т а б л и ц а 11.1

| Свойство | Изменяемый параметр окна |
|------------|---|
| width | Ширина окна в пикселях |
| height | Высота окна в пикселях |
| toolbar | Наличие панели инструментов |
| location | Наличие строки адреса |
| status | Наличие строки состояния |
| menubar | Наличие строки меню |
| scrollbars | Наличие полос прокрутки |
| resizable | Разрешает/запрещает изменение размеров окна |

Для того чтобы передать фокус некоторому окну, используется метод **focus**. Например:

```
My_window.focus( );
передает фокус окну, с которым связана переменная My_window.
Для закрытия окна используется метод close( ). Например:
window.close( ) // закрывает текущее окно
My_window.close( )
/* закрывает окно, с которым связана переменная
My_window*/
window.opener.close( )
//закрывает окно, из которого было открыто текущее
```

В JavaScript определены методы для вызова стандартных окон: **alert**, **confirm** и **prompt**.

Метод **alert("message")** вызывает диалоговое окно с сообщением, определенным параметром **message**, и кнопкой **OK**.

Метод **confirm("message")** вызывает диалоговое окно с указанным сообщением **message** и кнопками **OK** и **Cancel**.

Пример приведен ниже, результат – на рис. 11.6.

```
confirm("Хотите продолжить работу?");
```

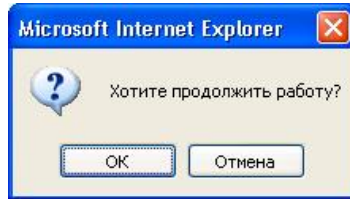


Рис. 11.6. Внешний вид окна **confirm** в Internet Explorer

Метод **prompt** (**"message"** [, **"inputDefault"**]) отображает диалоговое окно ввода текста. Параметр **message** определяет текст запроса пользователю, **inputDefault** указывает текст по умолчанию в поле ввода.

Пример приведен ниже, результат – на рис. 11.7.

```
prompt ("Как Вас зовут?",  
        "Неизвестный пользователь");
```

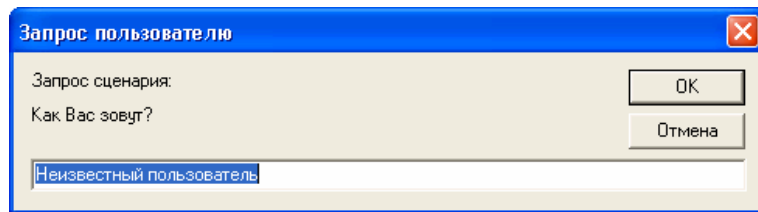


Рис. 11.7. Внешний вид окна **prompt** в Internet Explorer

Для выполнения заданий лабораторной работы нужно знать:

- типы данных JavaScript;
- правила определения имени переменной;
- операторы JavaScript;
- особенности использования функций eval, parseInt и parseFloat;
- способы доступа к объектам HTML-страницы;
- особенности обращения к отдельным элементам формы (например, спискам SELECT, переключателям RADIO и т. д.).

Порядок выполнения работы

1. Создайте HTML-страницу **Lab11_1.html**. Установите название

документа, цвет фона и текста.

2. Создайте сценарий JavaScript, который при загрузке данного документа будет вызывать окно, запрашивающее имя пользователя (используйте метод **prompt**).
3. Введенное имя сохраните в переменной **name**.
4. Выведите на страницу приветствие вида «Здравствуйте, **name!**». Если имя не введено, приветствие должно выглядеть следующим образом: «Здравствуйте! Было бы приятно с Вами познакомиться».
5. При щелчке левой кнопкой мыши на приветствии должно появляться окно **confirm**, в котором содержится вопрос «Вы хотите изменить имя пользователя?». Если ответ утвердительный, вновь вызовите окно запроса имени и соответствующим образом измените имя пользователя на странице.
6. Создайте документ **Lab11_2.html**, в соответствии с вашим вариантом (табл. 11.2). Определите соответствующие поля формы для получения входных данных. Пользователю должна быть предоставлена возможность выбора формы предъявления результата: в новом окне, в стандартном окне **alert** или на той же странице.

Т а б л и ц а 11.2

| Вариант | Задание |
|---------|--|
| 1 | Вычислите сумму n натуральных чисел |
| 2 | Определите n -й член ряда Фибоначчи |
| 3 | Вычислите факториал числа |
| 4 | Рассчитайте сумму двух чисел |
| 5 | Рассчитайте разность двух чисел |
| 6 | Рассчитайте произведение двух чисел |
| 7 | Определите частное двух чисел |
| 8 | Найдите корни квадратного уравнения |
| 9 | Определите, является ли високосным год, введенный пользователем |
| 10 | Определите сумму n членов ряда Фибоначчи |
| 11 | Определите длину окружности по ее радиусу |
| 12 | Определите результат вычисления арифметического выражения, введенного в текстовое поле |
| 13 | Найдите наименьший общий делитель двух чисел |
| 14 | Найдите наибольший общий делитель двух чисел |

| | |
|----|---|
| 15 | Найдите площадь прямоугольника по его ширине и высоте |
|----|---|

- Измените способ доступа к элементам формы, получите доступ к элементам всеми известными вам способами и проанализируйте особенности использования каждого из них.
- Добавьте на страницу *Lab11_1.html* кнопку, при нажатии на которую должно открываться новое окно с загруженной страницей *Lab11_2.html*.
- На страницу *Lab11_2.html* поместите ссылку, при нажатии на которую окно с данным документом будет закрываться.

Контрольные вопросы

- Укажите способы использования кода JavaScript в составе HTML-страницы.
- На HTML-странице определена следующая форма:

```
<FORM name="f1">
<INPUT type="text" name="txt"> <br>
<INPUT type="button" name="but"> <br>
</FORM>
```

Каким образом можно поместить в поле ввода **txt** некоторый текст, изменить размер данного поля, изменить надпись на кнопке?

- Как в JavaScript определяются комментарии?
- Перечислите методы создания стандартных окон JavaScript.
- Укажите параметры окна, которое открывается при выполнении следующего кода:

```
msgWindow=window.open("1.html", "displayWindow",
    "width=300,height=300,location=no,
    status=yes,toolbar=yes,menubar=no,
    scrollbars=no,resizable=no")
```

Как можно загрузить в данное окно документ *new.html*, используя атрибут TARGET тега <A>? Как можно закрыть данное окно?

Отчетность по лабораторной работе

- Файлы *Lab11_1.html* и *Lab11_2.html* на магнитном носителе.
- Отчет, содержащий:
 - цель работы;
 - листинги файлов *Lab11_1.html* и *Lab11_2.html* с комментариями;
 - иерархию объектов документа *Lab11_2.html*;

- заключение.

ЛАБОРАТОРНАЯ РАБОТА 12 РАБОТА СО СТАНДАРТНЫМИ ОБЪЕКТАМИ JAVASCRIPT

Цель работы: получение практических умений работы со стандартными объектами **Date**, **Math** и **Array**.

Теоретические сведения

К стандартным объектам, определенным в JavaScript, относятся объекты **Date**, **Array** и **Math**.

Объект **Date** позволяет работать с временем и датами, например, получить текущее время или определить, сколько дней осталось до некоторой знаменательной даты.

Экземпляр объекта **Date** создается следующим образом:

```
varName = new Date([parameters]);
```

Квадратные скобки [] в вышеприведенном выражении обозначают, что указание параметров **parameters** является необязательным.

В следующем примере [4] осуществляется вывод на страницу текущего времени, результат представлен на рис. 12.1.

```
<HTML>
<HEAD> <TITLE> Объект Date </TITLE> </HEAD>
<BODY>
<script language="JavaScript">
var now= new Date();
document.write("Time: " + now.getHours() + ":" +
now.getMinutes() + "<br>");
document.write("Date:" +now.getDate() + "/" +
(now.getMonth() + 1) + "/" + (now.getYear()));
</BODY>
</HTML>
```



Рис.12.1. Вывод текущей даты и времени

В примере, приведенном выше, создается экземпляр объекта **Date** с именем **now**, позволяющий получить доступ к текущим дате и времени.

Обратите внимание на выражение `(now.getMonth()+1)`. Поскольку месяцы представлены числами от 0 (январь) до 11 (декабрь), для привычного отображения номера месяца к результату, возвращаемому методом `getMonth()`, нужно прибавить 1.

Объект **Array** позволяет создавать массивы и работать с ними.

Экземпляр объекта **Array** создается следующим образом:

`arrayName=new Array([arrayLength | values]),`

где **arrayLength** – количество элементов массива;

values – значения элементов.

Параметры **arrayLength** и **values** являются взаимно-исключающими.

Основные методы объекта **Array** [1] представлены в табл. 12.1

Т а б л и ц а 12.1

| Метод | Действие |
|----------------|---|
| sort | Сортирует элементы массива |
| reverse | Изменяет порядок следования элементов массива на обратный |
| push | Добавляет один или несколько элементов в конец массива |
| pop | Удаляет последний элемент массива |
| unshift | Добавляет один или несколько элементов в начало массива |
| shift | Удаляет первый элемент массива |
| join | Объединяет элементы массива в строку |
| concat | Объединяет два массива в один |

Для определения количества элементов массива используется свойство **length** объекта **Array**.

Рассмотрим пример использования объекта **Array**, результат которого приведен на рис. 12.2.

```
<HTML>
<HEAD> <TITLE> Объект Array </TITLE>
<SCRIPT language="JavaScript">
function max(){
var My_array=new Array;
//в переменную str записываем содержимое поля
//array_str
var str=document.array_form.array_str.value;
//делим строку str по разделителю «пробел»
//и записываем полученные подстроки в массив
My_array=str.split(' ');
// находим максимальный элемент
var max=parseInt(My_array[0]);
for(var i=1;i<My_array.length-1;i++)
if (parseInt(My_array[i])>max)
max=parseInt(My_array[i]);
alert ('Максимальный элемент: '+ max);
}
</SCRIPT></HEAD>
<BODY>
<FORM name="array_form">
Введите элементы массива через пробел <br>
<INPUT type="text" name="array_str"
size="30"><br> <br>
<INPUT type="button" name="but"
value="Максимальный элемент"
onClick="max();">
</FORM>
</BODY>
</HTML>
```

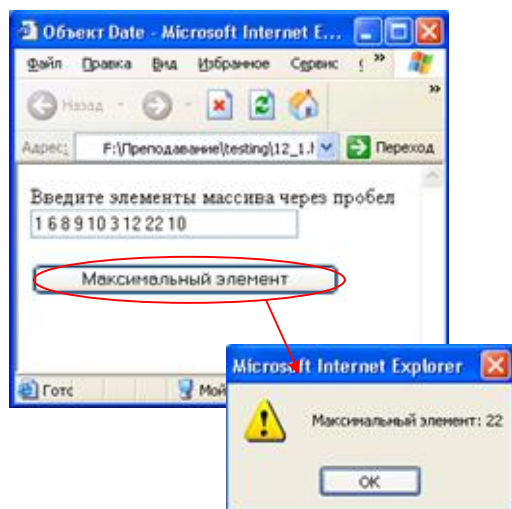


Рис. 12.2. Демонстрация выполнения функции **max()**

Объект **Math** предоставляет доступ к математическим константам и функциям. Основные свойства и методы данного объекта представлены в таблицах 12.2 и 12.3 соответственно.

Т а б л и ц а 12.2

| Свойство | Математическая константа |
|----------|---|
| E | Экспонента (основание натуральных логарифмов) |
| LN2 | Натуральный логарифм числа 2 |
| LN10 | Натуральный логарифм числа 10 |
| LOG2E | Логарифм с основанием 2 числа E |
| LOG10E | Логарифм с основанием 10 числа E |
| PI | Число π |
| SQRT1_2 | Квадратный корень из 0.5 |
| SQRT2 | Квадратный корень из 2 |

Т а б л и ц а 12.3

| Метод | Функция | Пример |
|-------|----------------------------|----------------------------|
| sin | Тригонометрические функции | Math.sin(Math.PI/2) |
| cos | | |
| tan | | |
| asin | | |
| acos | | |
| atan | | |

| | | |
|--------|---|-----------------------------------|
| abs | Абсолютное значение | Math.abs(-10) |
| ceil | Ближайшее целое число, большее или равное аргументу | Math.ceil(45.95) // 46 |
| exp | Значение e^{number} | Math.exp(1) // 2.718 |
| floor | Ближайшее целое число, меньшее или равное аргументу | Math.floor(45.95) // 45 |
| log | Натуральный логарифм для положительного аргумента | Math.log(1) // 0 |
| max | Наибольшее из двух чисел | Math.max(10,20) |
| min | Наименьшее из двух чисел | Math.min(10,20) |
| pow | Основание, возведенное в степень | Math.pow(7,2) // 49 |
| random | Случайное число в интервале между 0 и 1 | Math.random() |
| round | Значение, округленное до ближайшего целого числа | Math.round(20.49) // 20 |
| sqrt | Квадратный корень аргумента | Math.sqrt(9) |

Для выполнения заданий лабораторной работы нужно знать:

- виды параметров, указываемых при создании экземпляра объекта **Date**;
- методы объекта **Date**;
- особенности представления значений даты и времени;
- особенности использования свойств `innerHTML` и `innerHTML`.

Порядок выполнения работы

1. Создайте HTML-страницу **Lab12.html**. Установите название документа, цвет фона и текста.
2. Создайте сценарий JavaScript, выводящий на страницу при ее загрузке приветствие, связанное с временем суток, например, «Доброе утро» с 7.00 до 12.00, «Добрый день» с 12.00 до 17.00, «Добрый вечер» с 17.00 до 00.00, «Доброй ночи» с 00.00 до 7.00.
3. Добавьте на страницу **Lab12.html** форму (рис. 12.3) для расчета значений функции, соответствующей вашему варианту (табл. 12.4).

Т а б л и ц а 12.4

| Вариант | Задание | Вариант | Задание |
|---------|--|---------|----------------------------------|
| 1 | $y = \frac{x^2}{2x}$ | 9 | $y = \frac{1}{(5x-4)(5x-1)^2}$ |
| 2 | $y = \frac{1}{(2x+1)(2x^2+3)}$ | 10 | $y = \frac{1}{x \cdot 3^x}$ |
| 3 | $y = \frac{2^x}{(2x+1)3x}$ | 11 | $y = \frac{x}{(x+3)^3}$ |
| 4 | $y = x \cdot \left(1 - \cos \frac{1}{\sqrt{x}}\right)$ | 12 | $y = \frac{2^x}{3^x x}$ |
| 5 | $y = \sin \frac{1}{x} \cdot \operatorname{tg} \frac{1}{x}$ | 13 | $y = \left(\frac{2}{x}\right)^2$ |
| 6 | $y = \frac{2^x}{(2x-3)(2x-2)^3}$ | 14 | $y = \frac{1}{(3x+1)^3}$ |
| 7 | $y = \frac{x+1}{2^x(x-1)}$ | 15 | $y = \frac{1}{(2x-3)^2}$ |
| 8 | $y = \frac{1}{x\sqrt{x}}$ | 16 | $y = \frac{x-1}{x^2}$ |

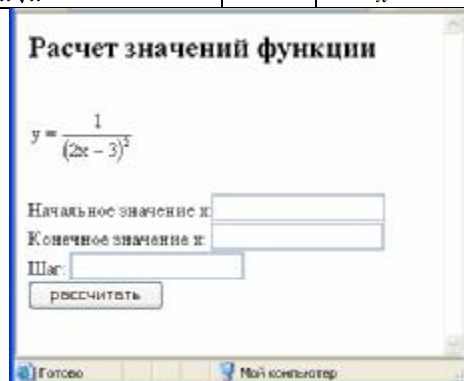


Рис. 12.3. Пример формы для расчета значений заданной функции

- Осуществите проверку введенных данных: числа должны быть положительными, начальное значение меньше конечного, шаг не должен превышать разницы между начальным и

конечным значениями.

- На основе введенных данных сформируйте массив **znach**, содержащий значения переменной x .
- Рассчитайте значения функции y для каждого x и сохраните результаты в массиве **func**.
- Выведите результаты расчетов на страницу **Lab12.html** в виде таблицы.

Контрольные вопросы

- Какие стандартные объекты вы знаете?
- Что обозначает следующая запись:
`myDate = new Date(97,02,10);` ?
- Перечислите известные вам методы объекта **Date**.
- Перечислите известные вам методы и свойства объекта **Math**.
- Перечислите известные вам методы и свойства объекта **Array**.

Отчетность по лабораторной работе

- Файл **Lab12.html** на магнитном носителе.
- Отчет, содержащий:
 - цель работы;
 - листинг файла **Lab12.html** с комментариями;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 13 СОЗДАНИЕ СОБСТВЕННЫХ ФУНКЦИЙ. ОРГАНИЗАЦИЯ УПРАВЛЕНИЯ ВРЕМЕНЕМ

Цель работы: формирование практических умений определения и использования функций в JavaScript, работы с функциями управления временем **setTimeout**, **clearTimeout**, **setInterval** и **clearInterval**.

Теоретические сведения

Формат определения пользовательской функции выглядит следующим образом:

```
function f_name([arg1,arg2,...])
{
  /* function body */
},
```

где **function** – ключевое слово, при помощи которого опре-

деляют функцию;

f_name – имя функции;

arg1, arg2, ... – необязательные параметры функции;

{ } – операторные скобки, ограничивающие тело функции.

Рассмотрим пример расчета площади круга по его радиусу.

Результат приведен на рис. 13.1.

```
<HTML>
<HEAD>
  <TITLE>Function_pr1</TITLE>
<SCRIPT language="JavaScript">
  function calc(){
    var rad=parseFloat(document.data.radius.value);
    var s=Math.pow(rad,2)*Math.PI;
    document.data.result.value=s;
  }
</SCRIPT>
</HEAD>
<BODY>
  <H3>Расчет площади окружности</H3>
  <FORM name="data">
    Введите радиус окружности:
    <BR><INPUT type="text" name="radius"><BR>
    <INPUT type="button" name="but"
      value="Рассчитать площадь"
      onClick="calc();" ><BR><BR>
    Результат: <INPUT type="text" name="result">
  </FORM>
</BODY>
</HTML>
```

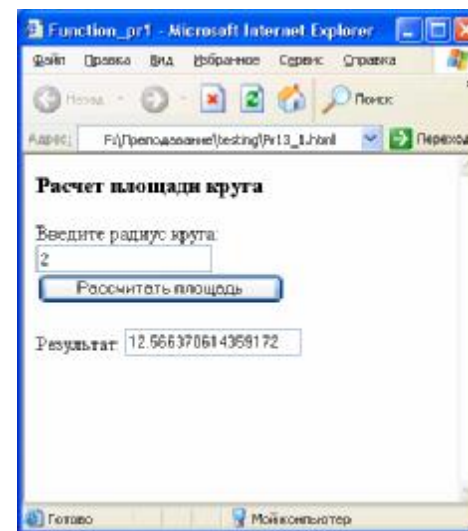


Рис. 13.1. Результат работы функции **calc()**

В результате выполнения функции **calc()** вычисляется площадь окружности, которая записывается в текстовое поле результата. Решим ту же задачу таким образом, чтобы функция только возвращала результат, а его запись в поле **result** осуществлялась при помощи оператора присваивания в параметре обработки события **onClick**:

```
...
function calc(){
  var rad=parseFloat(document.data.radius.value);
  var s=Math.pow(rad,2)*Math.PI;
  return s;
}
...
<INPUT type="button" name="but"
  value="Рассчитать площадь"
  onClick="document.data.result.value=calc();" >
<BR><BR>
...
```

Для того чтобы функция могла выполняться для различных значений, при ее описании используются так называемые формальные параметры, а при вызове – фактические параметры [1].

Изменим функцию **calc()** из примера, приведенного выше,

таким образом, чтобы в нее в качестве параметра передавался введенный пользователем радиус окружности:

```
...
function calc(rad){
var rad=parseFloat(rad);
var s=Math.pow(rad,2)*Math.PI;
document.data.result.value=s;
}
...
<INPUT type="button" name="but"
value="Рассчитать площадь"
onClick="calc(document.data.radius.value);">
...
```

В качестве параметра может передаваться также объект, соответствующий форме:

```
...
function calc(obj){
var rad=parseFloat(obj.radius.value);
var s=Math.pow(rad,2)*Math.PI;
obj.result.value=s;
}
...
<INPUT type="button" name="but"
value="Рассчитать площадь"
onClick="calc(document.data);">
...
```

Для управления временем в сценариях JavaScript используются функции **setTimeout**, **setInterval**, **clearInterval** и **clearTimeout**, являющиеся методами объекта window.

Рассмотрим пример, в котором через 5 секунд после загрузки страницы появляется окно сообщения с текстом «Время истекло!»:

```
<html>
<head>
<title>Функция SetTimeout</title>
<script language="JavaScript">
function timer() {
    setTimeout("alert('Время истекло!')", 5000);
}
</script>
</head>
<body onLoad="timer( )">
...
```

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

Первый параметр функции **setTimeout** определяет код JavaScript, который должен выполняться по истечении времени, указываемого вторым параметром (в миллисекундах).

В следующем примере функция **setTimeout** используется для создания бегущей строки в поле ввода (документ *scroll_msg.html*):

```
<HTML>
<HEAD>
<TITLE>Функция SetTimeout</title>
<SCRIPT Language="JavaScript">
// текст бегущей строки
var msg = "Бегущая строка ";
function scrollMsg(){
//записываем в поле Msg текст бегущей строки
document.scrollerForm.Msg.value = msg;
/*изменяем строку msg таким образом, чтобы ее
первый символ переместился в конец строки*/
msg=msg.substring(1,msg.length)+
msg.substring(0,1);
/* функция scrollMsg() вызывает сама себя на
выполнение через 150 миллисекунд*/
setTimeout("scrollMsg()", 150);
}
</SCRIPT>
</head>
<BODY ONLOAD="scrollMsg()">
<FORM NAME="scrollerForm">
<INPUT TYPE="text" NAME="Msg" VALUE="" size=15>
</FORM>
</BODY>
</HTML>
```

Добавим на страницу *scroll_msg.html* кнопки **Стоп** и **Пуск** для остановки и возобновления движения строки.

```
<HTML>
<HEAD>
<TITLE>Функция SetTimeout</TITLE>
<SCRIPT Language="JavaScript">
var msg = "Бегущая строка ";
var id;
function scrollMsg(){
document.scrollerForm.welcomeMsg.value=msg;
msg=msg.substring(1,msg.length)+msg.substring(0,1);
```

[illegible]

В данном примере идентификатор потока вычислений, создаваемого функцией `setTimeout`, сохраняется в глобальной переменной `id`: `id=setTimeout("scrollMsg()", 150)`.

Функция **clearTimeout** уничтожает поток вычислений с идентификатором, указанным в качестве параметра **clearTimeout(id)**, т. е. в данном случае останавливает движение строки.

Для управления временем может также использоваться функция **setInterval**("Код JavaScript",time), которая вызывает программный код JavaScript, определенный в первом параметре, не один раз как **setTimeout**, а каждый раз по истечении времени **time**. Для уничтожения потока вычислений, созданного функцией **setInterval**, используется функция **clearInterval**.

Порядок выполнения работы

1. Обеспечьте движение строки в поле ввода, используя функцию **setInterval**; результат сохраните в файле **Lab13_1.html**.
2. Создайте документ **Lab 13_2.html**, содержащий поле ввода с «идущими» часами.
3. Установите запрет на изменение данного поля ввода.
4. Добавьте на страницу кнопки **Стоп** и **Пуск** для остановки и запуска часов.

5. Определите необходимые функции без параметров.
6. Измените функцию запуска часов таким образом, чтобы в качестве параметра передавался объект страницы – поле ввода, в котором будут размещены часы. Результат сохраните в документе ***Lab13_21.html***.
7. В документе ***Lab13_21.html*** для пуска и остановки часов используйте только одну кнопку, надпись на которой будет меняться в зависимости от их текущего состояния.

Контрольные вопросы

1. Укажите формат определения пользовательской функции.
2. Определите различие между формальными и фактическими параметрами.
3. Для чего используются функции **setTimeout** и **clearTimeout**?
4. В чем различие функций **setTimeout** и **setInterval**?

Отчетность по лабораторной работе

1. Файлы ***Lab13_1.html***, ***Lab13_2.html*** и ***Lab13_21.html*** на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - листинги файлов ***Lab13_1.html***, ***Lab13_2.html*** и ***Lab13_21.html*** с комментариями;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 14

РАБОТА СО СЛОЯМИ И ИЗОБРАЖЕНИЯМИ

Цель работы: формирование практических умений работы со слоями и изображениями, используя JavaScript.

Теоретические сведения

JavaScript расширяет возможности использования слоев. Динамическое изменение свойств слоя позволяет перемещать объекты, находящиеся на данном слое, или делать их невидимыми.

Чтобы получить доступ к слою, созданному при помощи тега `<div>`, ему необходимо назначить уникальное имя `id`.

Таким образом, если задать слой

```
<div id=myLayer ...>
...
</div>,
```

то в дальнейшем можно получить доступ к нему с помощью конструкции **document.all["myLayer"]**.

JavaScript предоставляет возможность изменения параметров слоя (расположение, размеры и т. д.) путем изменения свойств стиля данного слоя.

```
document.all["myLayer"].style.visibility=hidden
// изменение видимости - слой становится невидимым
document.all["myLayer"].style.left=200
document.all["myLayer"].style.top=200
// изменение положения
document.all["myLayer"].style.width=100
document.all["myLayer"].style.height=150
// изменение размеров слоя
```

В следующем примере слой будет отображаться или скрываться при нажатии на кнопку (результат представлен на рис. 14.1):

```
<html>
<head>
<script language="JavaScript">
function showHide()
{
if (document.all["myLayer"].style.visibility=
"visible")
document.all["myLayer"].style.visibility="hidden"
else
document.all["myLayer"].style.visibility="visible";
}
</script>
</head>
<body>
<div id=myLayer style="visibility:visible">
<font size=4 color="#0000ff"><i>
Текст внутри слоя</i></font>
</div>
<form>
<input type="button" value="Show/Hide layer"
onClick="showHide()">
</form>
</body>
</html>
```

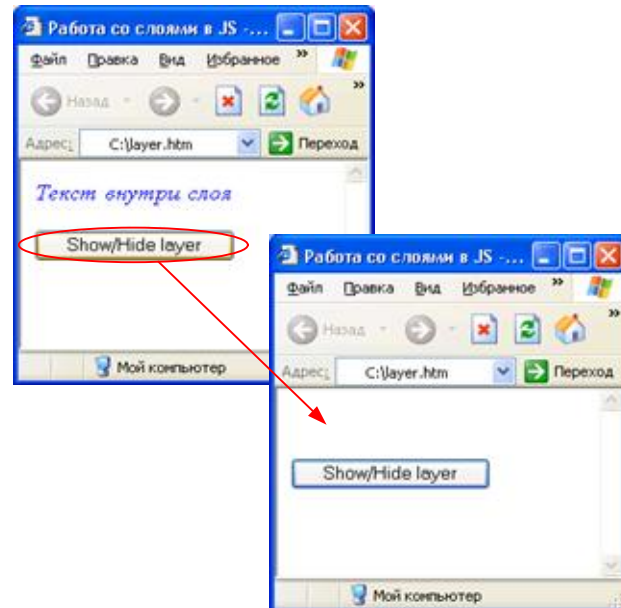


Рис.14.1. Динамическое изменение видимости слоя

В языке JavaScript все изображения организованы в массив, который называется **images** и является свойством объекта **document**. Каждое изображение на Web-странице получает порядковый номер: первое изображение получает номер 0, второе – номер 1 и т. д.

Каждое изображение в HTML-документе рассматривается как объект **Image**. Объект **Image** имеет определенные свойства (соответствующие атрибутам тега **IMG**), к которым можно обращаться посредством JavaScript. Например, определить, какой размер имеет изображение, можно обратившись к его свойствам **width** и **height** [3].

В следующем примере происходит смена изображений при наведении курсора мыши:

```
<html>
<head><title>Смена изображений в JS</title>
</head>
<body>

</body>
</html>
```

Для выполнения заданий лабораторной работы нужно знать:

- способы доступа к объектам HTML-страницы;
- сущность предварительной загрузки изображений;
- свойство **display**.

Порядок выполнения работы

1. Используя слои, в документе *Lab14_1.html* организуйте меню для доступа к выполненным лабораторным работам по дисциплине «Интернет-программирование» в соответствии с вариантом, предложенным преподавателем.

Вариант 1. Горизонтальное двухуровневое меню, раскрывающееся при наведении указателя мыши. Второй уровень меню скрывается, когда указатель мыши покидает область выбранного пункта или появившегося меню.

Вариант 2. Вертикальное двухуровневое меню, раскрывающееся при наведении указателя мыши. Второй уровень меню скрывается, когда указатель мыши покидает область выбранного пункта или появившегося меню.

Вариант 3. Горизонтальное двухуровневое меню, раскрывающееся при щелчке левой кнопкой мыши. Второй уровень меню скрывается, когда указатель мыши покидает область выбранного пункта или появившегося меню.

Вариант 4. Вертикальное двухуровневое меню, раскрывающееся при щелчке левой кнопкой мыши. Второй уровень меню скрывается, когда указатель мыши покидает область выбранного пункта или появившегося меню.

Вариант 5. Горизонтальное двухуровневое меню, раскрывающееся при щелчке левой кнопкой мыши. Второй уровень меню скрывается при повторном щелчке по пункту меню.

Вариант 6. Вертикальное двухуровневое меню, раскрывающееся при щелчке левой кнопкой мыши. Второй уровень меню скрывается при повторном щелчке по выбранному пункту.

Вариант 7. Текстовое двухуровневое меню: при выборе определенного пункта меню пункты верхнего уровня раздвигаются и

вставляются подпункты выбранного пункта, т. е. пункт меню «раскрывается».

2. Используя самостоятельно созданные изображения (например, в редакторе Paint), выполните задание в соответствии с вариантом, предложенным преподавателем, и сохраните результат в файле *Lab14_2.html*. Используйте предварительную загрузку изображений.

Вариант 1. Разместите на странице два изображения и кнопку **Переставить**. При нажатии на данную кнопку изображения должны поменяться местами.

Вариант 2. Разместите на странице четыре изображения, два поля ввода и кнопку **Переставить**. При нажатии на данную кнопку изображения с номерами, введенными в текстовые поля, должны поменяться местами.

Вариант 3. Организуйте последовательный просмотр изображений (не меньше четырех), используя кнопки **Предыдущее изображение** и **Следующее изображение**.

Вариант 4. Добавьте на страницу изображение; при наведении указателя мыши оно должно постепенно увеличиваться в размерах, создавая иллюзию приближения изображения (предельные значения высоты и ширины установите самостоятельно).

Вариант 5. Добавьте на страницу изображение; при наведении указателя мыши оно должно постепенно уменьшаться в размерах, создавая иллюзию удаления изображения (предельные значения высоты и ширины установите самостоятельно).

Вариант 6. Организуйте последовательный просмотр изображений (не меньше четырех). Смена изображений должна осуществляться через некоторые равные промежутки времени.

Вариант 7. Организуйте движение изображения слева направо.

Вариант 8. Организуйте движение изображения справа налево.

Вариант 9. Организуйте движение изображения из левого верхнего угла окна к правому нижнему.

Вариант 10. Организуйте движение изображения из правого верхнего угла окна к левому нижнему.

Вариант 11. Организуйте движение изображения сверху вниз.

Вариант 12. Организуйте движение изображения снизу вверх.

Контрольные вопросы

1. Каким образом можно получить доступ к слою, используя

JavaScript?

2. Перечислите известные вам свойства стиля, применимые для слоев.
3. Укажите способы доступа к изображению.
4. Перечислите известные вам свойства объекта Image.
5. Каковы преимущества предварительной загрузки изображений?

Отчетность по лабораторной работе

1. Файлы **Lab14_1.html** и **Lab14_2.html** на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - листинги файлов **Lab14_1.html** и **Lab14_2.html** с комментариями;
 - заключение.

ЛАБОРАТОРНАЯ РАБОТА 15

РЕШЕНИЕ СЛОЖНЫХ ЗАДАЧ ПОСРЕДСТВОМ JAVASCRIPT

Цель работы: закрепление умений создания и использования функций для работы с объектами страницы.

Для выполнения заданий лабораторной работы нужно знать:

- способы доступа к объектам HTML-страницы;
- особенности обращения к отдельным элементам формы;
- особенности создания и использования функций;
- функции для работы со строками.

Порядок выполнения работы

1. Используя возможности JavaScript, создайте инженерный калькулятор и сохраните его в файле **Lab15.html**. Должны быть реализованы следующие функции:

- простейшие арифметические операции (+, -, *, /);
- $\sin()$;
- $\cos()$;
- $\ln()$;
- \sqrt{x} ;
- $1/x$;
- x^2 ;

- e^x ;

- 10^x ;

- десятичная точка для ввода вещественных чисел.

В калькуляторе должны быть предусмотрены кнопки с цифрами для ввода операндов.

При выполнении какой-либо операции должна проверяться ее корректность (например, при делении нужно проверить, не является ли знаменатель нулем и т. д.).

В строке состояния должна отображаться текущая операция или выдаваться ошибка в случае ее возникновения.

Дополнительные функции реализуйте в соответствии с вашим вариантом (табл. 15.1).

Т а б л и ц а 15.1

| Вариант | Задание |
|---------|---|
| 1 | Запрет ввода чисел с клавиатуры |
| 2 | Удаление последнего символа |
| 3 | Занесение результата вычислений в память |
| 4 | Вычисление факториала |
| 5 | Вычисление x^n |
| 6 | Перевод из радиан в градусы |
| 7 | Перевод из градусов в радианы |
| 8 | Вычисление модуля числа |
| 9 | Вычисление остатка от деления |
| 10 | Округление |
| 11 | Вычисление суммы цифр числа |
| 12 | Вычисление процента от числа |
| 13 | Изменение знака числа |
| 14 | Генерация случайного числа от 0 до 1 |
| 15 | Генерация случайного целого числа от 1 до 100 |

Отчетность по лабораторной работе

1. Файл **Lab15.html** на магнитном носителе.
2. Отчет, содержащий:
 - цель работы;
 - листинг файла **Lab15.html** с комментариями;
 - заключение.

Список использованных источников

1. Дмитриева, М. В. JavaScript. Экспресс-курс / М. В. Дмитриева. – СПб. : БХВ-Петербург, 2005.
2. Использование HTML 4 : учеб. пособие : пер. с англ. / М. Р. Браун [и др.]. – 4-е изд. – М. : Вильямс, 1999.
3. Кох, Ст. Введение в JavaScript для мага : пер. с нем. / Ст. Кох. – 1996, 1997. www.math omsu. omskreg.ru/info/learn/js/koch/javascript.html
4. Николенко, Д. В. Практические занятия по JavaScript / Д. В. Николенко. – СПб. : Наука и техника, 2000.
5. Хокинс, Скотт. Администрирование Web-сервера Apache и руководство по электронной коммерции : пер. с англ. / Скотт Хокинс. – М. : Вильямс, 2001.
6. Ярошевич, А. О. Производственное обучение. Информатика : практикум для студентов спец. 1-08 01 01-07 «Профессиональное обучение. (Информатика)» : в 6 ч. Ч. 5. Сетевые технологии / А. О. Ярошевич. – Мн. : МГВРК, 2005.

Интернет-источники

7. css.manual.ru
8. html.manual.ru

Приложение А

(справочное)

Свойства CSS*

| Свойство | Описание | Возможные значения |
|------------------------|----------------------------------|--|
| Свойства шрифта | | |
| font-family | Тип шрифта или семейство шрифтов | <i>Шрифты: Arial, Verdana</i> и т. д. <i>Семейства: serif</i> (с засечками), <i>sans-serif</i> (без засечек), <i>fantasy</i> (с украшениями), <i>monospace</i> (моноширинные) и т. д. |
| font-size | Размер шрифта | Стандартные единицы длины (<i>px, cm</i> и т. д.), проценты, ключевые слова (<i>xx-small, x-small, small, medium, large, x-large, xx-large</i>) |
| font-style | Вид начертания | <i>normal</i> (нормальное начертание), <i>italic</i> и <i>oblique</i> (курсивное начертание) |
| font-variant | Регистр букв | <i>normal, small-caps</i> (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 | <i>Left</i> (по левому краю), <i>right</i> (по правому краю), <i>center</i> (по центру) и <i>justify</i> (по ширине страницы) |
| text-decoration | Выделение текста | <i>underline</i> (подчеркивание), <i>overline</i> (линия над текстом), <i>line-through</i> (зачеркивание) и т. д. |
| text-indent | Отступ первой строки абзаца | Стандартные единицы длины (<i>px, cm</i> 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 | <i>capitalize</i> (первая буква каждого слова выводится в верхнем регистре), <i>uppercase</i> (все в верхнем регистре), <i>lowercase</i> (все в нижнем регистре), <i>none</i> (регистр букв не изменяется) |

* css.manual.ru

| Свойство | Описание | Возможные значения |
|------------------------------|--|--|
| 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 |
| background-attachment | Перемещение фонового изображения при прокрутке | scroll (фоновое изображение перемещается вместе с текстом), fixed (изображение не движется относительно окна браузера во время прокрутки документа) |
| background-position | Точка начала копирования фонового изображения | 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 оформления | | |
| 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), thick (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 |

| Свойство | Описание | Возможные значения |
|--|--|---|
| 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 оформления 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 . | | |
| 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 | inherit (значения данного свойства |

| Свойство | Описание | Возможные значения |
|---------------------------------|---|---|
| | элемента | 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. |
| 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 <i>неупорядоченных</i> списков: disc (диск), circle (окружность), square (квадрат) или none (нет маркера) Для <i>упорядоченных</i> 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: ползунок, кнопок со стрелками, дорожки для ползунка | Ключевое слово (white , red и т. д.), код RGB |
| scrollbar- | Цвет ползунка и | Ключевое слово (white , red и т. д.), код |

| Свойство | Описание | Возможные значения |
|----------------------------------|--|--|
| face-color | кнопок со стрелками | RGB |
| scrollbar-highlight-color | Цвет подсветки, создающий эффект объемности | Ключевое слово (white , red и т. д.), код RGB |
| scrollbar-shadow-color | Цвет тени для ползунок 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 |

Содержание

| | |
|---|-----------|
| Предисловие | 3 |
| ЛАБОРАТОРНАЯ РАБОТА 1 | |
| Установка и настройка WEB-сервера Apache | 3 |
| ЛАБОРАТОРНАЯ РАБОТА 2 | |
| Создание HTML-документов | 8 |
| ЛАБОРАТОРНАЯ РАБОТА 3 | |
| Вставка текста в HTML и его последующее логическое и физическое форматирование | 11 |
| ЛАБОРАТОРНАЯ РАБОТА 4 | |
| Организация системы ссылок и внедрение в HTML графики ... | 16 |
| ЛАБОРАТОРНАЯ РАБОТА 5 | |
| Работа с таблицами. Создание списков | 20 |
| ЛАБОРАТОРНАЯ РАБОТА 6 | |
| Использование слоев и изображений-карт | 26 |
| ЛАБОРАТОРНАЯ РАБОТА 7 | |
| Создание страниц с использованием фреймов | 32 |
| ЛАБОРАТОРНАЯ РАБОТА 8 | |
| Работа с формами | 40 |
| ЛАБОРАТОРНАЯ РАБОТА 9 | |
| Создание и внедрение в Web-страницы таблиц стилей | 45 |
| ЛАБОРАТОРНАЯ РАБОТА 10 | |
| Создание DHTML-страниц | 49 |
| ЛАБОРАТОРНАЯ РАБОТА 11 | |
| Создание простейших скриптов и внедрение их в HTML-документ | 53 |
| ЛАБОРАТОРНАЯ РАБОТА 12 | |
| Работа со стандартными объектами JavaScript | 63 |
| ЛАБОРАТОРНАЯ РАБОТА 13 | |
| Создание собственных функций. Организация управления временем | 69 |
| ЛАБОРАТОРНАЯ РАБОТА 14 | |
| Работа со слоями и изображениями | 74 |
| ЛАБОРАТОРНАЯ РАБОТА 15 | |
| Решение сложных задач посредством JavaScript | 79 |
| Список использованных источников | 81 |
| Приложение А – Свойства CSS | 82 |

Учебное издание

Тетерукова Наталья Александровна

ИНТЕРНЕТ-ПРОГРАММИРОВАНИЕ

Лабораторный практикум

для учащихся специальности 2-40 01 01

«Программное обеспечение информационных технологий»

В двух частях

ЧАСТЬ 1

WEB-СЕРВЕР

ЯЗЫК ГИПЕРТЕКСТОВОЙ РАЗМЕТКИ HTML

ЯЗЫК ПРОГРАММИРОВАНИЯ НА СТОРОНЕ

КЛИЕНТА JAVASCRIPT

Зав. ред.-издат. отд. О. П. Козельская

Редактор Г. Л. Говор

Корректор Н. Г. Михайлова

Компьютерная верстка Н. М. Олейник

План издания 2007 г. (поз. 25)

Изд. лиц. № 02330/0131735 от 17.02.2004.

Подписано в печать 30.10.2007. Формат 60×84 1/16.

Бумага писчая. Гарнитура Таймс. Печать ризографическая.

Усл. печ. л. 5,12. Уч.-изд. л. 3,99. Тираж 100 экз. Заказ 203.

Издатель и полиграфическое исполнение Учреждение образования
«Минский государственный высший радиотехнический колледж»
220005, г. Минск, пр-т Независимости, 62.

