СЕВАСОПОЛЬСКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Создание статического сайта на HTML и Javascript

Методические указания
к выполнению лабораторных работ
по дисциплине «Технологии Web - приложений»
для студентов очной формы обучения
по направлению 09.03.01 «Информатика и вычислительная техника»
Часть 1

Севастополь

2014

УДК 621.396.6

МЕТОДИЧЕСКИЕ УКАЗАНИЯ к выполнению лабораторных работ по дисциплине «Технологии Web - приложений» для студентов очной формы обучения по направлению 09.03.01 «Информатика и вычислительная техника» / Сост. доцент кафедры ИТКС Лелеков С.Г. – Севастополь: Изд-во СГУ, 2014. – 65с.

Целью методических указаний является оказание помощи студентам в выполнении лабораторных работ по дисциплине «Технологии Web - приложений».

Приведены варианты заданий, рекомендации по выполнению, требования к оформлению работ.

Методические указания рассмотрены и утверждены на заседании кафедры информационных технологий и компьютерных систем, протокол №11 от 30 июня 2014г.

Допущено учебно-методическим центром СГУ в качестве методических указаний.

Рецензент: Шаталова Ю.Г. доцент кафедры «информационных технологий и компьютерных систем».

Содержание

1. ЦЕЛИ И ЗАДАЧИ ЛАБОРАТОРНОГО ПРАКТИКУМА

Выполнение лабораторных работ предусмотрено рабочим планом подготовки бакалавров по направлению «Информатика и вычислительная техника» для студентов очной формы обучения.

Выполнение цикла лабораторных работ преследует следующие цели:

В первой части (лабораторные работы 1-5) - общее знакомство с методами и средствами разработки статических Web - сайтов, языком разметки HTML, стилевому оформлению страниц с помощью CSS, основам языка Javascript.

Во второй части (лабораторные работы 6-12) - общее знакомство с методами и средствами разработки динамических Web – сайтов с использованием средств языка сценариев PHP и СУБД MySQL.

2. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Методические указания по выполнению лабораторной работы содержат теоретическое введение, пример выполнения заданий лабораторной работы, индивидуальное задание, требования к оформлению отчета. Все варианты индивидуальных заданий имеют примерно одинаковую трудоемкость. Способ выбора варианта определен в заданиях.

Перед выполнением лабораторной работы необходимо внимательно прочитать текст методических указаний, ознакомиться с рекомендуемой литературой. Предлагаемый пример выполнения задания желательно повторить, чтобы уже на его основе выполнять индивидуальное задание. При этом допускается использование других инструментальных средств и способов выполнения заданий, не рассмотренных в данных методических указаниях.

По результатам выполнения работы оформляется печатный отчет, который обязательно должен содержать:

- титульный лист;
- описание выполнения индивидуального задания;
- список использованной литературы.

Требования к описанию выполнения индивидуального задания в каждой работе свои и изложены в методических указаниях к ее выполнению.

3.1. Лабораторная работа №1. Использование HTML для разработки простого статического Web - сайта

Введение

World Wide Web – это средство представления информации в Интернет. Более точно WWW можно определить как интерактивную мультимедийную гипертекстовую среду, использующую язык разметки и поддерживающую множество протоколов Интернет. Обсудим основные моменты в этом определении.

Мультимедийная среда – возможность размещать в Сети не только и не столько текстовую информацию, но и графику, звук, видеоклипы.

Гипертекстовая среда – возможность использования в тексте ссылок на другие порции информации (текст, графику, звуковые и видеоклипы и т.д.), дополнительные источники. Многообразие таких ссылок и порождает Всемирную паутину WWW!

В Web эти ссылки называют «Унифицированным указателем ресурсов URL». Каждый документ или файл в Интернет имеет собственный URL, что позволяет легко ссылаться на него из других документов.

Язык разметки HyperText Markup Language (HTML) является основой World Wide Web. С помощью конструкций языка создаются Web страницы (сайты), которые можно просматривать с помощью специальных программ - обозревателей, или как их еще называют - браузеров. Наиболее известными браузерами являются Microsoft Internet Explorer, Mozilla Firefox, Opera, Google Chrome.

Протоколы Internet. WWW использует собственный протокол связи между отдельными узлами: HTTP (HyperText Transfer Protocol). Любой протокол - это набор правил, которые используются компьютерами для обмена информацией. Кроме HTTP можно использовать и другие протоколы, получая доступ и другим приложениям Internet: UseNet - для доступа к группам новостей, FTP - для загрузки файлов; POP3, SMTP - электронной почте, Skype для телефонного общения и др.

WWW - в отличие от телевидения, интерактивная среда. Пользователи сами определяют, какой из узлов посещать, как долго изучать полученную информацию. Более того, на Web страницах часто можно ввести диалог с их авторами.

Web использует технологию "клиент - сервер". Это означает, что где – то в сети существует компьютер, на котором работает программное обеспечение Web - сервера, управляющее доступом к информации. Различные пользователи со всего мира являются клиентами сервера и получают от него информацию с помощью своих браузеров.

Необходимые программные средства для выполнения работы.

Для выполнения задания 1 потребуется:

1. Любой Web обозреватель (браузер), например, Intenet Explorer, Mozilla Firefox, Opera и др.
2. Так как HTML -документы чисто текстовые, для создания Web - страниц можно использовать практически любой текстовый редактор, от Windows Notepad до Microsoft Word. Одним из наиболее удобных для работы можно считать **Notepad++.** Скачать **Notepad++** можно, пройдя по этой ссылке: http://notepad-plus.sourceforge.net/ru/site.htm

Создание HTML - файлов

Основным понятием языка HTML является понятие тег - инструкция обозревателю, как отображать текст.

Независимо от того, как выглядит ваша домашняя страница, и какую информацию вы хотите отобразить, существует три тега, которые в соответствии со стандартами HTML и WWW должны присутствовать в каждой Web-странице:

- <html> Сообщает браузеру, что документ написан на языке HTML.
- <head> Отмечает вводную и заголовочную части HTML-документа.
- <body> Отмечает основной текст и информацию.

Эти теги необходимы Web-браузеру для определения различных частей HTML-документа, но они не оказывают прямого влияния на внешний вид Web-страницы. Они необходимы для того, чтобы последующие нововведения в HTML правильно интерпретировали Web - страницу, а также для того, чтобы она выглядела одинаково во всех Web-браузерах. Например, на Web-сервере запущена программа, которая просматривает все HTML-документы и составляет их список. Она просматривает только текст, расположенный внутри тегов <head>, поскольку здесь находится название документа. Таким образом, если в вашей домашней странице нет тегов <head> и </head>, она не будет включена в список. Так работает большинство средств поиска наиболее популярных Web-серверов. Они берут информацию из тегов <head> (и других).

Теги <html> и </html>

Эти теги сообщают браузерам, что текст между ними следует интерпретировать как текст HTML. Поскольку документы HTML чисто текстовые, тег <html> говорит о том, что файл написан на языке гипертекстовых ссылок.

Задание 1.Откройте блокнот и наберите <html> в самом начале файла. Затем наберите его напарника </html>. Теперь весь текст, который появится между этими тегами, будет отмечен как текст формата HTML. Вы обратили внимание на символ “/” во втором теге? Прямая косая черта используется для обозначения *закрывающих* тегов. Большинство тегов HTML парные, открывающий и закрывающий, они охватывают помечаемый ими текст. Закрывающий тег в паре всегда начинается в прямой косой черты. Сейчас ваша домашняя страница выглядит так:

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

Теги <head> и </head>

Теперь введем теги <head> и </head>. Они служат для размещения в вашем HTML-документе служебной информации..

Задание 2. Наберите <head> на экране между тегами <html>, затем в следующей строке наберите его пару – тег </head>.

Теги <body> и </body>

Теги <body> и </body>, как и <head>, предназначены для обозначения отдельной части HTML-документа. Текст, охваченный тегами <body>, является основной частью документа.

Сюда войдут большая часть текста и иная информация, составляющие основу документа.

Задание 3. Введите в вашу Web-страницу теги <body> и </body>, и документ примет следующий вид:

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

Теги <address> и </address>

Теперь рассмотрим еще пару тегов - <address>. Они охватывают информацию о том, к кому нужно обращаться в отношении данной страницы в том случае, если у кого-нибудь возникнут вопросы или замечания.

Теги <address> используются для того, чтобы отделить эту информацию от основного блока.

Задание 4. Между тегами <body> и </body> наберите ваше имя и адрес электронной почты, например:

Петр Иванов-e-mail: Petr@Ivanov.com

Теперь введите тег <address> на строчке выше имени и адреса.

Введите закрывающий тег </address> после имени и адреса (вся контактная информация может занимать несколько строк).

Ваш HTML-файл теперь выглядит следующим образом:

```
<html>
        <head>
        </head>
        <body>
        <address>
              Петр_Иванов_e-mail:Petr@Ivanov.com
        </address>
        </body>
</html>
```

Сохраните этот файл с расширением имени .html . (Меню Файл/Сохранить, Тип файла выбрать – «Все файлы», Имя файла – first.html).

Откройте его в Web-браузере.

Теги заголовка

В заголовке чаще всего размещают название страницы (тег <title>), служебную информацию для браузера: теги <meta>, <link>.

Название домашней страницы выводится браузером в строке заголовка окна. Название вводится с помощью тегов <title> и </title> размещаемых в заголовочной части между тегами <head> и </head>. Каждый документ может иметь только одно название.

Добавьте название Вашей домашней страницы, например, так:

<title> Домашняя страница Петра Иванова </title>

Тег <meta> может иметь, например, такой вид:

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

Тег <meta> здесь определяет тип содержимого документа. Атрибут http-equiv и его значение Content-type определяет, что сейчас будет описан тип документа. Атрибут content и его значение text/html; charset = windows-1251 сообщают браузеру, что данный документ является HTML-документом (text/html), и кодировка этого документа кириллица (windows-1251).

Задание 5. Добавьте в текст страницы тег <title>. Сохраните файл с расширением .html и откройте его в браузере.

Параметры тега <BODY>

В таблице 1.1. представлены атрибуты тега <BODY> предназначенные для изменения фона html-страницы.

Таблица 1.1.Параметры фона

Параметр	**Функция**	**Примеры**
BGCOLOR	Определение цвета фона	Задание белого фона

		<BODY BGCOLOR="white"> <BODY BGCOLOR="#FFFFFF"> <BODY BGCOLOR="255,255,255">
BACKGROUND	Указание фонового рисунка	<BODY BACKGROUND= "images/bg.gif">
BGPROPERTIES	Изменение свойств фона	BGPROPERTIES= "fixed" Прокручивая содержание документа, фоновое изображение остается в зафиксированном виде
TEXT	Определение цвета текста	`TEXT = "green"`

Описанные параметры не являются обязательными, однако использование BGCOLOR рекомендуется по следующей причине: пользователь в настройках своего браузера может поставить любой цвет фона, а разработчик, полагая, что белый цвет является основным по умолчанию, может не указать этот параметр. В результате вместо подразумевающегося белого цвета, фон может оказаться черным, зеленым и т. д., что способно привести к нарушению оформления документа.

Также наряду с графическим изображением фона рекомендуется использовать и параметры цвета на тот случай, если рисунок не загрузится (тогда браузер отобразит цвет).

Таблица 1.2. **Параметры границ документа**

Параметр	**Функция**	**Примеры**
TOPMARGIN	Определяет отступ от верхнего края документа	<BODY TOPMARGIN="5" BOTTOMMARGIN="5" LEFTMARGIN="10" RIGHTMARGIN="10" MARGINWIDTH="10" MARGINHEIGHT="5">
BOTTOMMARGIN	Определяет отступ от нижнего края документа	
LEFTMARGIN	Определяет отступ от левого края документа	
RIGHTMARGIN	Определяет отступ от правого края документа	
MARGINWIDTH	Определяет горизонтальный отступ	
MARGINHEIGHT	Определяет вертикальный отступ	

Примечание: Все значения параметров задаются в пикселах.

Заголовки

Существует шесть размеров шрифта заголовков (они пронумерованы от одного до шести, причем первый соответствует самому крупному шрифту).

Для ввода заголовка самым крупным первым номером, требуется поместить текст заголовка между тегами <h1> и </h1>. Если нужен заголовок размером поменьше, то необходимо воспользоваться тегами <h2> и </h2> и т.д.

Добавьте в Вашу страницу между тегами <body> </body> текст заголовка
Добро пожаловать на сайт Петра Иванова!
Обозначьте этот текст тегами <h1> и </h1>.
Введите подзаголовок
Приглашаются все желающие!
Обозначьте этот текст тегами <h2> и </h2>.

Задание 6. Добавьте на свою страницу теги заголовков. Сохраните файл и откройте или обновите Вашу страницу в браузере.

Ввод текста и иной информации

Теперь, когда вы ввели название и заголовки, введите информацию, рассказывающую посетителям о Вас. Ввод текста - это, возможно, самый простой этап в процессе создания домашней страницы, поскольку вы можете набрать текст непосредственно в HTML - файле.

Задание 7. Наберите, например, следующий текст: «Я рад приветствовать Вас на моей Web - странице. Я живу в славном городе Севастополе. Учусь в университете. С помощью Интернет хочу приобрести новых друзей.»

Сохраните файл и откройте Вашу страницу в браузере.

Однако, так как браузеры реагируют только на разметку HTML, то весь Ваш набранный текст будет отображаться в виде одного большого не отформатированного абзаца.

Для форматирования текста можно использовать следующие теги:

Тег нового абзаца - предписывает браузеру разделить два фрагмента текста пустой строкой. Ввести тег нового абзаца просто. Нужно установить курсор в то место, где Вы хотите вставить тег, и набрать . В конце абзаца поставьте тег .

Тег может содержать параметр ALIGN, отвечающий за тип горизонтального выравнивания текста в окне браузера.

ALIGN="left" — текст выровнен по левому краю (значение параметра, принятое по умолчанию);

ALIGN="center" — текст располагается посередине окна браузера;

ALIGN="right" — выравнивание текста по правому краю. Идеально подходит для создания

эпиграфов, подписей, заголовков и пр.;

ALIGN="justify" — выравнивание по ширине окна браузера.

Например, Текст абзаца

Тег перевода строки
 - аналогичен тегу , только он не вводит пустой строки. После этого тега текст продолжает отображаться с начала следующей строки.

Тег горизонтальной линии <hr> помогает разделить страницу на смысловые части.

Задание 8. Отформатируйте свой текст, используя теги ,
, <hr>. При этом горизонтальной линией отчеркните адрес электронной почты. Сохраните файл и откройте Вашу страницу в браузере.

Различные виды линий можно получить, используя атрибуты тега <hr>:

width - задает ширину линии, например, width =”50%” определяет линию шириной в половину экрана;

align - задает выравнивание линии. Возможны следующие варианты: align=”left”, align=”center”, align=”right”;

size- задает толщину линии. Обычная ширина составляет 2 пикселя (size=”2”), но можно указать любую ширину от 1 до 175 пикселей.

Чтобы указать атрибут, его необходимо включить в тег <hr>, например, <hr size=”5” width=”75%” align=”left”>.

Если в тексте требуется разместить несколько подряд идущих пробелов, то лучше использовать специальную комбинацию символов ** ** столько раз подряд, сколько требуется пробелов.

Стилевое оформление текста

Управление стилем текста осуществляется парными тегами. Форматируемый текст заключается между тегом начала и тегом конца.

Теги   служат для выравнивания текста по центру экрана.

Теги и позволяют отображать текст документа полужирным шрифтом.

Теги и позволяют отображать текст документа курсивом.

Теги <u> и </u> позволяют подчеркивать текст.

Также полезными являются теги для изменения размера текста в html-документе. В таблице 2.2.приведены их описание.

Таблица 2.2.Теги изменения размеров текста.

Тег	**Функция**
<BIG></BIG>	Отображение текста шрифтом большего, чем непомеченная часть текста размера
<SMALL></SMALL>	Отображение текста шрифтом меньшего размера
****	Сдвиг текста ниже уровня строки и вывод его (если возможно) шрифтом меньшего размера. Удобно использовать для математических индексов
****	Сдвиг текста выше уровня строки и вывод его (если возможно) шрифтом меньшего размера. Удобно использовать для задания степеней чисел
<FONT></FONT>	один из основных тегов форматирования текста, отображающий свойства шрифтов. Для него могут использоваться следующие параметры, приведенные в таблице 2.3.

Таблица 2.3.Атрибуты тега <FONT>

Параметр	**Функция**
FACE	Служит для указания типа шрифта Например, **<FONT FACE="Verdana", "Arial", "Helvetica">**
SIZE	Служит для указания размеров шрифта в условных единицах от 1 до 7.

	Конкретный размер шрифта зависит от браузера. По умолчанию – норма равна 3. Размер шрифта указывается абсолютной величиной, например SIZE=2, так и относительной - SIZE=+1
COLOR	Служит для указания цвета шрифта. Значения могут быть заданы словами, числом в шестнадцатеричной системе, тройкой чисел RGB. Например, **<FONT COLOR="#FF0000"> Текст будет написан красным цветом </FONT>**

Задание 9. Отформатируйте свой текст, используя теги управления стилем. Сохраните файл и откройте Вашу страницу в браузере.

Списки

Список задает некоторое перечисление. Каждый элемент перечисления располагается с новой строки или нумеруется, или отмечается каким - нибудь символом (маркером). В первом случае список называется нумерованным, во втором - маркированным.

Чтобы задать маркированный список в HTML используется парный тег

 . Позиции списка отмечаются непарным тегом <li>. Допускается использование заголовка списка, который заключается в парный тег <lh> </lh>. Например, Вы хотите список своих увлечений оформить в виде маркированного списка. Это можно сделать например так:

 <lh> Мои увлечения </lh>
 Общение с друзьями
 Музыка
 Книги

Чтобы задать нумерованный список в HTML используется парный тег

 . Позиции списка отмечаются также непарным тегом <li>.

Для нумерованного списка можно выбрать тип нумерации позиций. По умолчанию это арабские цифры, допускаются также буквы английского алфавита, римские цифры. Для указания типа используется атрибут type.

Например, тег <ol type=”a”> определяет нумерацию в виде букв английского алфавита, а <ol type=”1”> определяет нумерацию в виде арабских цифр.

Задание 10. Добавьте на свою страницу список. Сохраните файл и откройте Вашу страницу в браузере.

Таблицы

Принцип создания таблиц в HTML таков: создаётся таблица, потом создаётся строка, потом все столбцы данной строки(т.е ячейки), потом очередная строка, снова все очередные столбцы данной строки и так далее.

Таблица создаётся с помощью тега <table>, а заканчивается тегом </table>. У таблицы есть строки и столбцы, поэтому их необходимо создать. Теперь согласно принципу создания таблиц, необходимо создать строку. Строка создаётся с помощью тега <tr>. Соответственно, сигналом к окончанию строки является закрывающий тег </tr>. Внутри тега <tr> необходимо создавать столбцы, которые создаются с помощью тега <td>. И уже

внутри этого тега находятся те элементы, которые должны быть расположены внутри данной ячейки. После того, как все элементы уложены, то можно закрывать столбец с помощью тега </td>. Дальше открывается новый тег <td>. В него снова помещаются элементы, после этого закрывается </td>. Это сигнал к концу второго столбца. И так далее, столько столбцов, сколько Вам нужно. В конце закрывается строка тегом </tr>. Затем следующая строка и так далее. А заканчивается всё закрывающим тегом </table>.

Задание 11. Добавьте на страницу таблицу, содержащую типовой распорядок Вашего выходного дня. Напрмер:

№	Мероприятие	Время	
		Начало	Конец
1	Подъем	9-00	10-00
2	Утренняя гимнастика и водные процедуры	10-00	10-30
3	Завтрак	10-30	11-00
	. . .	. . .	. . .

```
<h1><center> Мой распорядок дня </h1></center>

<table width="75%" border="1" align=”center” bgcolor=”pink” style=”color:blue;”>
 <tr>
  <td width="11%" align="center" rowspan="2">№ </td>
   <td width="55%" align="center" rowspan="2">Мероприятие </td>
   <td width="17%" align="center" colspan="2">Время</td>
 </tr>
 <tr>
   <td width="17%" align="center">Начало</td>
   <td width="17%" align="center">Конец</td>
 </tr>
<tr>
   <td width="11%" align="center">1</td>
   <td width="55%">Подъем</td>
   <td width="17%" align="center">9-00</td>
   <td width="17%" align="center">10-00</td>
 </tr>
 <tr>
   <td width="11%" align="center">2</td>
   <td width="55%">Утренняя гимнастика и водные процедуры</td>
   <td width="17%" align="center">10-00</td>
   <td width="17%" align="center">10-30</td>
 </tr>
 <tr>
   <td width="11%" align="center">3</td>
   <td width="55%">Завтрак</td>
   <td width="17%" align="center">10-30</td>
   <td width="17%" align="center">11-00</td>
 </tr>
</table>
```

Продолжите создание таблицы самостоятельно.

Теперь об атрибутах. Сначала атрибуты тега <table>.

1) Атрибут "border", значение которого задаёт толщину рамки таблицы в пикселях. По умолчанию, рамки вообще нет. Давайте поставим значение этого атрибута равное "2".

2) Атрибуты "width" и "height" задают ширину и высоту таблицы соответственно. Размер может быть указан, как в абсолютных единицах (пиксели, px), так и в относительных (проценты, %). Относительный размер, чем хорош, так это тем, что он всегда подстроится под любое разрешение монитора пользователя и любой браузер. А абсолютные тем хороши, что при любых браузерах и любых разрешениях монитора не будет сюрпризов с дизайном, связанные, например, с растягиванием элементов (если монитор широкоэкранный, к примеру). Давайте поставим значение атрибутов "width" и "height" по "30%".

3)Атрибут "bgcolor" задает цвет фона таблицы. Можно использовать, например, следующие значения: aqua, black, blue, fuchsia, gray, grey, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow.

4) Атрибут "style" позволяет задать форматирование таблицы в стиле CSS (см. задание 3).

Атрибуты тега <tr>:

1) Атрибут "height". Заметьте, что у тега <tr> нет атрибута "width", впрочем, это логично, ведь тег <tr> отвечает за строку, а, следовательно, за высоту. А за ширину отвечают столбцы.

Собственно, даже атрибут "height" не особо-то и используется, поэтому можно сказать, что атрибутов у тега <tr> и вовсе нет.

Наиболее богатым по количеству атрибутов является тег <td>, отвечающий за ячейку таблицы:

1) Атрибут "width". Объяснение то же, что и для атрибута тега <tr>. Соответственно, атрибута "height" нет.

2) Атрибут "colspan". Значение этого атрибута означает количество столбцов, которое занимает данная ячейка.

3) Атрибут "rowspan". Значение этого атрибута означает количество строк, которое занимает данная ячейка.

4) Атрибут "align". Значение этого атрибута означает выравнивание элемента внутри ячейки по горизонтали. Бывают три значения: "left" (по левому краю), "center" (по центру), "right" (по правому краю). По умолчанию стоит выравнивание по левому краю.

5) Атрибут "valign". Значение этого атрибута означает выравнивание элемента внутри ячейки по вертикали. Снова имеются только три значения: "top" (по верху), "middle" (по середине), "bottom" (по низу). По умолчанию стоит значение "middle".

Размещение изображений на Web - странице

В качестве рисунка может быть использован любой файл с расширением *.gif или *.jpg или *.png. Для размещения изображений на странице используется тег <img src="Имя

файла">. Например, <img SRC=”Tigers.gif” alt=”Здесь долен быть тигр”>. Так можно указывать только те файлы, которые расположены в той же папке, что и исходная страница.

Для выравнивания изображения относительно текста используется атрибут align.

Для масштабирования изображения можно использовать атрибуты width и height. Например, <img src=”Tigers.gif” align=right width=300 height=200>. Здесь значения высоты и ширины изображения приводится в пикселях(точках). **Чтобы изображение сохранило свои пропорции при масштабировании надо использовать только один из этих атрибутов.**

Атрибут "alt" указывает текст, который описывает картинку. Этот же текст будет показываться в случае, если картинка по каким-либо причинам будет не найдена, либо у пользователя в браузере отключён показ картинок. Очень желательно этот атрибут ставить, так как это помогает оптимизации сайта.

Изображение можно разместить в виде фона. Для этого нужно задать значение атрибута background для тега <body>. Например,

<body background="bd.jpg">.

Задание 12. Добавьте на страницу какие-либо изображения, фотографии. Сохраните файл и откройте Вашу страницу в браузере.

Ссылки на Web - страницах

Гиперссылки на Web - страницы - одно из основных свойств WWW. С помощью таких ссылок и родилась Всемирная паутина.

Для создания гипертекстовой ссылки используется специальный парный тег

 . Рассмотрим подробнее структуру гиперссылки на примере:

 Новости СНТУ .

Здесь http://www.sevntu.com.ua/news.html - представляет URL - адрес файла в Internet.

«Новости СНТУ» - текст, который видит пользователь на экране обозревателя. Этот текст обычно выделен синим цветом, и при наведении на него мыши появляется характерный указатель в виде указательного пальца.

Если документ, на который ссылается данная ссылка, находится в той же самой директории, что и исходный документ, то в качестве URL - адрес файла можно просто указать имя файла, например.

 Следующая страница .

Разумеется, ссылаться можно не только на HTML-страницы, но и на любые файлы, будь то картинки, будь то фильмы, будь то музыка, будь то архивы, будь то ещё всё, что угодно.

Последний атрибут - это "title". Этот атрибут задаёт текст, который будет виден при наведении мышки на ссылку.

Что касается того, что внутри тега, то в большинстве случаев - это обычный текст. Очень часто делают картинку внутри тега **<a>**, тогда эта ссылка будет в виде картинки.

Тег **** означает конец гиперссылки.

Задание на самостоятельную работу

1. Создайте личный Web - сайт с информацией о себе, включающий, по крайней мере, две страницы. На первой странице поместите общую информацию о себе, фотографию. На второй более частные подробности, например, информацию о своих

планах на следующую неделю, предыдущие места учебы, работы, Ваших увлечениях и т.п. Впрочем, Ваша фантазия ничем не ограничивается.

2. Подготовьте отчет, в который включите: HTML – тексты, разработанных страниц, скриншоты страниц, описания использованных тегов и их атрибутов.

3.3. Лабораторная работа №2.1. Использование CSS при разработке Web - сайта. Часть 1. Основы табличной верстки

Табличная верстка

В основу табличной верстки положено представление сайта в виде таблицы. Как известно подавляющее большинство сайтов имеют типовую структуру, которую можно детализировать под конкретную задачу. Весь макет представляется некоторой таблицей, а каждая ячейка наполняется своим информационным содержанием.

Достоинства и недостатки табличной верстки:
Проста в понимании, не требует более глубоких знаний HTML и CSS (+)
Интуитивно понятна при построении, минимум CSS правил (+)
Трудно разбираться в HTML коде при более сложной структуре сайта (-)
Пока вся таблица не загрузиться она не будет показана на экране (-)
Сложный дизайн с перекрытием элементов не реализуем (-)
Много лишнего кода (-)

Для иллюстрации приемов табличной верстки возьмем задачу разработки сайта об автомобилях. Сайт преднзначен для посещения автолюбителей, интересующихся историей появления автомобиля, новинками автомобилестроения и т.п.

Сайт будет состоять из нескольких страниц, структуру которых надо спроектировать заранее.

Собственно проект страницы может выглядеть следующим образом (рис.2):

<table>
<tr><td colspan="2">Шапка страницы</td></tr>
<tr><td>Главное (Top) меню</td><td>Форма авторизации</td></tr>
<tr><td>Содержимое страницы (Контент)</td><td>Банеры</td></tr>
<tr><td colspan="2">Подвал</td></tr>
</table>

Рис.2 Верстка страниц сайта об автомобилях

Далее создадим, например в «Блокноте», файл index.php. Разместить его требуется в папке с именем «www».

Подготовим заголовок:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd>
<html>
<head>
    <title> Сайт об автомобилях </title>
    <meta http-equip="Content-type" content="text/html; charset=1251" />
    <link rel="stylesheet" type="text/css" href="styles/main.css" />
</head>
<body>
</body>
</html>
```

Для того, чтобы применить таблицу стилей к HTML-документу, можно избрать один из трёх способов, либо комбинировать их:

1.Применить *внешние стили* (в виде отдельного текстового .*css*-файла) с помощью тега <link>.

2.*Встроить стили* непосредственно в HTML-документ (в виде блока css-текста) с помощью тега <style>.

3.Применить *inline-стиль*, то есть назначить стиль конкретному HTML-элементу непосредственно в документе, с помощью HTML-атрибута style (см. задание 1).

Наиболее распространенными являются первый и второй способы.

В нашем примере будет применен первый способ, с описанием стилей в отдельном файле, расположенном в папке styles с именем main.css.

В основу шаблона сайта составляет таблица, то добавим ее создание в тег <body>.

```
<body>
    <table id="main">
        <tr>
            <td colspan="2" id="header">
            <h1> Сайт об автомобилях</h1>
            <p>
                <img src= "images/header.png" alt ="Шапка сайта" />
            </p>
            </td>
        </tr>
        <tr>
            <td colspan="2">
                <hr />
            </td>
        </tr>
    </table>

</body>
```

Проверьте результаты своей работы, открыв файл index.php в браузере. Так как он имеет расширение имени *php*, то автоматически он не откроется!

Среди параметров некоторых тегов появился параметр “ID=”. Это один из параметров, позволяющих использовать стили.

Создадим в папке *www* папку *styles*. В ней создадим файл *main.css*.

В файле main.css для тега <body> обнулим отступы(margin, padding):

```
body {
  margin: 0;
  padding: 0;
}
```

Обнулим отступы для прямой (тег <hr>):

```
hr {
  margin: 0;
}
```

Используя ID селекторы, растянем основную часть на весь экран и сделаем выравнивание заголовка по центру.

```
#main {
  width: 100%;
}
#header {
  text-align: center;
}
```

Дополнительно можно убрать отступы в заголовке (использованы контекстные селекторы):

```
#header h1, #header p {
  margin: 0 ;
}
```

Проверьте результаты своей работы, открыв файл index.php в браузере.

Закончив с заголовком, переходим к следующей части таблицы: верхнего меню и форме авторизации (рис.2).

Откройте вновь файл index.php

В соответствии с проектом страницы добавим в таблицу следующую строку и разобьем ее на две ячейки

```
<tr>
  <td>  </td>
  <td>  </td>
</tr>
```

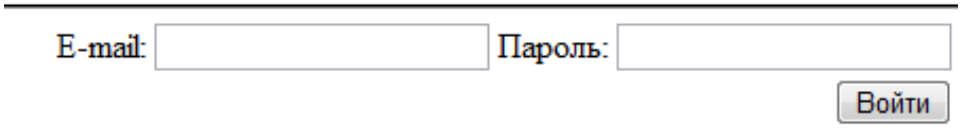
В первой ячейке будет меню из четырех пунктов: Главная, Регистрация, Статьи, Гостевая книга. Оформим его в виде таблицы из одной строки.

```
</tr>   <!-- конец заголовка -->
<tr>
    <td>    <!-- верхнее меню -->
        <table id="topmenu">
            <tr>
                <td>
                    <a href ="index.php"> Главная </a>
                </td>
                <td>
                    <a href ="reg.php"> Регистрация </a>
                </td>
                <td>
                    <a href ="articles.php"> Статьи </a>
                </td>
                <td>
                    <a href ="guestbook.php"> Гостевая книга </a>
                </td>

            </tr>
        </table>

    </td>
```

Файлы reg.php, articles.php, guestbook.php создадим позже.

Займемся формой авторизации. Она находится в ячейке рядом с главным меню.

Содержимым формы будут два поля для ввода E-mail и пароля и ниже кнопки «Войти».



Оформим форму в виде таблицы.

```
</td>
<td class="right">  <!-- форма авторизации -->
    <form name="auth" action="auth.php" method="post">
    <table>
        <tr>    <!-- 1-я строка формы авторизации -->
            <td>E-mail:</td>
            <td>
                <input type="text" name="email" />
            </td>
            <td>Пароль: </td>
            <td>
                <input type="password" name="password" />
            </td>
        </tr>
        <tr>    <!-- 2-я строка формы авторизации -->
            <td colspan="4">
            <input type="submit" name="button_auth" value="Войти">
            </td>

        </tr>

    </table>
    </form>
    <tr>    <!-- прямая линия -->
        <td colspan="2"> <hr /> </td>
    </tr>
</td>
```

Строка меню и формы авторизации готова. Внизу проведена ограничительная горизонтальная линия.

Теперь займемся оформлением главного меню и формы авторизации, используя стили. Откройте папку styles и в ней файл main.css. Добавьте следующие стили.

```
#topmenu{
    font-size:150%;
    width:100%
}

#topmenu td{
    text-align:center;
}
.right{
    float:right;
}

.right td{
    text-align:right;
}
```

Следующая часть разметки страницы Content и Banners. Вновь откройте файл index.php. Добавим после горизонтальной черты отделяющей верхнее меню и форму авторизации еще одну строку и ячейку. В ячейке поместим таблицу.

```
<tr>
  <td colspan=”2”>
        <table cellpadding=”0” cellspacing=”0” id=”content”>
```

```
        </table>
  </td>
</tr>
```

Содержимым таблицы будет строка из двух ячеек.

```
<tr>
  <td>   </td> <!—1-я ячейка -->
  <td>   </td> <!—2-я ячейка -->

</tr>
```

В первой ячейке будет располагаться основной текст об автомобиле и его фотография.

```
<tr>    <!-- начало контента  -->
    <td colspan="2">
        <table  id="content">
            <tr>
                <td> <!--1-я ячейка -->
                    <div class="article">
                        <h1> Автомобиль </h1>
                        <p class="article_img">
                        <img src="images/main.png" alt="Статья 1"/>
                        </p>
                        <p>Текст…</p>
                        <p>Текст…</p>
                        <p>Текст…</p>
                        <p>Текст…</p>
                    </div>
                </td>
```

Во второй ячейке будут располагаться баннеры.

```
                <td id="banners_240"> <!--2-я ячейка -->
                    <div class="banner">
                        <a href="#">
                            <img src="images/banner_1.jpg" alt="Баннер1" />
                        </a>
                    </div>
                    <hr />
                    <div class="banner">
                        <a href="#">
                        <img src="images/banner_2.jpg" alt="Баннер2" />
                        </a>
                    </div>
                    <hr />
                </td>
```

Ссылки на реальные сайты пока отсутствуют.

Теперь оформим стили. Откройте файл main.css. Для контента и подвала добавим в него:

```
#content {
    width: 100%;
    padding: 0;
    margin: 0;
}
#content td{
    vertical-align: top;
}
.article h1, .article_img {
    text-align: center;
}
#banners_240 {
    border-left: 1px solid  #000;
    width: 240px;
}
#footer p {
    text-align: center;
}
```

Последней строкой является Подвал. Отделим его чертой:

```
        <tr>
            <td colspan="2">
                <hr />
            </td>
        </tr>
        <tr>
            <td colspan="2" id="footer">
                <p>Все права защищены &copy; "Сайт об автомобилях"  </p>
            </td>
        </tr>


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

Страница готова. Наполним ее контентом из прилагаемого файла.

Для этого замените фрагмент

Текст…
Текст…
Текст…
Текст…

абзацами первой статьи из приложения 2.

Файлы рисунков можно найти в папке images, которую также следует разместить в папке www создаваемого Web – сайта.

Задание на самостоятельную работу.

1.Создать сайт об автомобилях, руководствуясь изложенным выше материалом;

2.Создать собственный сайт согласно варианту, который определяется следующим образом: найти остаток от деления двух последних цифр зачетной

книжки (индивидуального плана) студента на 19; к полученному остатку добавить единицу.

Например, две последние цифры 34. Тогда остаток от деления 34 на 19 равен 15 плюс 1 - получим 16 – й вариант. Или, если две последние цифры 12, то остаток также равен 12 плюс 1 - получим 13.

№ варианта	**Организация, для которой выполняется проектирование учебного сайта**
1	магазин;
2	ресторан;
3	санаторий;
4	заказ такси;
5	гостиница;
6	поликлиника;
7	офис по продаже недвижимости;
8	бюро по трудоустройству и профориентации
9	атлетический клуб
10	больница (стационар)
11	автомобильные грузоперевозки
12	продажа железнодорожных билетов
13	малое производственное предприятие;
14	деканат ВУЗа
15	автосервис
16	туристическая фирма
17	парикмахерская
18	троллейбусный парк (организация движения)
19	библиотека

3.Подготовить отчет, включающий постановку задачи, структурную схему сайта, исходные тексты, скриншоты, выводы.

Более подробную информацию о CSS можно получить, например, на сайте www.css.manual.ru

3.2. Лабораторная работа №2.2. Использование CSS при разработке Web - сайта. Часть 2. Основы блочной верстки

Введение

CSS (Cascading Style Sheets, каскадные таблицы стилей) – технология описания внешнего вида документа, написанного языком разметки. Преимущественно используется как средство оформления веб-страниц в формате HTML и XHTML, но может применяться с любыми видами документов в формате XML, включая SVG и XUL.

CSS используется создателями веб-страниц для задания цветов, шрифтов, расположения и других аспектов представления документа. До появления CSS оформление веб-страниц осуществлялось непосредственно внутри содержимого документа (см. лабораторную работу №1). Однако, с появлением CSS стало возможным принципиальное разделение содержания (написанного на HTML или другом языке разметки) и представления (написанного на CSS) документа. Это разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печать, чтение голосом (специальным голосовым браузером или программой чтения с экрана) и др.

Чтобы применить CSS-оформление к HTML-элементу или множеству элементов, обычно используются *селекторы* – специальные указатели на HTML-объекты, к которым мы планируем применить css-правило.

CSS правило состоит из селектора и списка свойств и их значений заключенного в фигурные скобки.

Вот три основных вида селекторов.

1) HTML селекторы

Это простейший случай – в качестве селектора мы используем имя того html-элемента, который хотим изменить. Например, для тега `<strong>` селектором будет `strong`. Соответственно, для тега `<h1>` селектором будет `h1`, и так далее. Теперь мы можем переопределить внешний вид всех таких элементов в нашем документе:

```
strong {font-weight: normal; color: red;}
h1 { font: bold 10pt verdana; }
```

2) Селекторы класса

«Класс» - это некое имя, строка, которое мы можем применить к любым HTML-тегам, чтобы впоследствии ссылаться на них по имени класса. В качестве имени класса вы можете использовать практически любую строку. Удобство таких селекторов в том, что можно присвоить одно имя класса множеству html-тегов в документе и затем управлять их внешним видом, обращаясь к ним по имени класса:

Пусть есть фрагмент HTML – кода, сожержащий несколько абзацев, например:

```
<p class=”myClass”>  Текст1  </p>
<p>Текст2  </p>
<p class=”myClass”>  Текст3  </p>
```

Тогда, если применить стиль к классу .myClass { font: bold 10pt verdana; }, то шрифт изменится, только у первого и третьего абзаца.

3) ID селекторы (или идентификаторы)

Любой идентификатор (ID) – это некое имя, которое можно применить, так же, как и в случае с классами, к любому HTML-тегу. Основное отличие – ID должен быть уникален в рамках html-документа:

```
#myObj { margin: 1em; } /* изменяем поля для элемента, у которого id=”myObj” */
span#today { margin: 1em; } /* изменяем поля для элемента span, у которого id=”today” */
```

Список основных свойств, используемых для CSS правил, приведен в Приложении 1.

Работа по созданию сайта начинается с разработки макетов страниц. Этот этап называется версткой. Различают блочную и табличную верстки.

Блочная верстка

Блочная верстка базируется на том, что блочные элементы HTML, как правило, располагаются по вертикали, сверху вниз друг за другом в том порядке, в котором они встречаются в HTML коде. Кроме этого блокам можно задавать свойство плавучести (float:left | right | none | inherit). Если блоку указать свойство float:left, то он будет выровнен по левому краю, а все остальные блоки будут игнорировать его, как будто этого блока нет, за исключение текста, остальные блоки, которым задано это же свойство будут обтекать его справа, на сколько это позволяет ширина экрана или элемента внутри которого они находятся. Следует заметить, что любой элемент можно сделать блочным, заданием ему свойства display:block, изначально только элементы div по умолчанию считаются блочными.

float:right выровнит блок по правому краю, а все остальные блоки будут игнорировать его, либо обтекать, если им задано это же свойство и если в коде идут подряд два или несколько блоков с указанным свойством, то первым вправо встанет тот блок, который идет первым в коде, остальные обтекают его слева.

Свойство float:none отменяет эффект плавучести для блока, но это не значит что блок будет располагаться как обычно сверху вниз, если выше расположен блок с эффектом плавучести, то нижний блок будет игнорировать верхний и встанет под него, чтобы этого не было нужно задать этому блоку свойство clear:both.

float:inherit - задает свойство плавучести, такое же, как у родительского блока (блока в который вложен данный дочерний блок).

Достоинства и недостатки блочной верстки:

Меньший вес страницы за счет меньшего кода (+)

Реализация сложного дизайна с перекрывающимися блоками (+)

Трудно освоить, табличная верстка проще (-)

Чаще приходится решать вопросы кроссбраузерности. блоки могут перекрываться при изменении разрешения экрана, масштабировании (-).

Ход работы

Используем блочную верстку для создания web- страницы с динамическим меню навигации при помощи списков и правил CSS.

Необходимость в такое меню с набором гиперссылок - одна из часто встречающихся задач в практике веб-дизайнера. Меню двухуровневое: ссылки обычно группируют по категориям, а в основное меню включают только категории. При выборе категории должны отобразиться соответствующие ссылки (или подкатегории).

Как правило, такие динамические меню принято создавать средствами языка Javascript, позволяющего совершать любой сложности манипуляции с элементами веб-страницы. Однако, существует и решение на CSS - довольно простое и красивое, хотя и не любую фантазию дизайнера выполняющее. Рассмотрим его пошагово.

В этой работе (и нескольких следующих) воспользуемся специализированным редактором, например, Microsoft Visual Web Developer. Мы рекомендуем именно его, поскольку он бесплатен, а, совершенствуясь профессионально, вы можете перейти на Microsoft Visual Studio - одно из лучших средств разработчика самого широкого круга приложений. Для студентов Visual Studio в настоящее время также предоставляется бесплатно (см. http://dreamspark.ru).

1. Запустите Visual Web Developer (или Visual Studio) и создайте новый документ HTML.

2. Внутри элемента head введите подходящий заголовок страницы(тег <title>).

3. Также внутри элемента head введите тэг <style>. Закрывающий тег </style> появится автоматически.

4. Создайте в теле документа набор категорий в виде неупорядоченного списка (ul), каждый элемент которого содержит список ссылок.

```
  <ul id="MainMenu">
            <li><a href="#" title="Популярные блюда">Популярные блюда</a>
            <ul>
            <li><a href="/Italian.htm">Итальянская кухня</a></li>
            <li><a href="/Snack.htm">Закуски</a></li>
            <li><a href="/Breakfast.htm">Закуски</a></li>
            <li><a href="/Sweets.htm">Сласти</a></li>
            <li><a href="/Fruits.htm">Фрукты</a></li>
            </ul>
            </li>
            <li><a href="#" title="Подарки к празднику">Подарки к
празднику</a>
            <ul>
            <li><a href="/Anniversary.htm">Юбилеи</a></li>
            <li><a href="/Baby.htm">Малышам</a></li>
            <li><a href="/Birthday-Food.htm">День рождения</a></li>
            <li><a href="/Congratulations.htm">Поздравляем!!!</a></li>
            </ul>
            </li>
            <li><a href="#" title="Выбор по цене">Выбор по цене</a>
            <ul>
            <li><a href="Under-1000.htm">Меньше 1000 руб</a></li>
            <li><a href="1000-To-1500.htm">От 1000 до 1500 руб</a></li>
            <li><a href="1500-To-2000.htm">От 1500 до 2000 руб</a></li>
            <li><a href="2000-To-3000.htm">От 2000 до 3000 руб</a></li>
            <li><a href="3000-up.htm">Дорого</a></li>
            </ul>
            </li>
            <li><a href="#" title="Фрукты и Овощи">Фрукты и Овощи</a>
            <ul>
```

```
                <li><a href="/Dried-Fruits-Nuts.htm">Сухофрукты и
Орехи</a></li>
                <li><a href="/Fruit-Baskets.htm">Корзины фруктов</a></li>
                <li><a href="/Fruit-Towers.htm">Горы фруктов</a></li>
                <li><a href="/Healthy-Food.htm">Здоровая пища</a></li>
                <li><a href="/Organic-Food.htm">Без пестицидов</a></li>
                </ul>
                </li>
                <li><a href="#" title="Садовые и комнатные цветы">Цветы</a>
                <ul>
                <li><a href="/Fresh-Flowers.htm">Живые цветы</a></li>
                <li><a href="/Plants-And-Dish-Gardens.htm">Садовые
цветы</a></li>
                <li><a href="/Sympathy-Flowers.htm">"Говорящие"
цветы</a></li>
                </ul>
                </li>
                <li><a title="Корпоративные подарки" href="/Corporate-
Food.htm">
                Корпоративные подарки</a> </li>
                    </ul>
```

Названия ссылок и категорий придумайте самостоятельно. Важно, чтобы атрибут id внешнего списка имел значение MainMenu - далее ему будет назначен особый стиль по этому идентификатору).

В таблице стилей добавьте следующее правило:

```
#MainMenu > li {
        float: left;
        list-style-type: none;
      }
```

После применения указанного стилевого правила пункты внешнего списка (li) расположились горизонтально (за счёт обтекания):

Пусть вложенные списки ссылок будут невидимыми (добавьте им свойство display:none;) и появляются только при наведении курсора на название соответствующей категории. Следующее правило с селектором псевдокласса hover имеет такой смысл: у списка (ul), вложенного в пункт списка (li), на который наведён указатель (:hover) и который вложен в элемент с id=#MainMenu, способ отображения следует сделать блочным (а не невидимым):

```
#MainMenu li:hover ul {
    display:block;
   }
```

Сохраните разрабатываемый документ, откройте его в браузере и убедитесь, что меню работает правильно.

5. Принципиально механизм работает - осталась эстетическая сторона.

- Назначьте якорям любого уровня вложенности в меню (правило #MainMenu a) желаемый цвет (color), гарнитуру (font), а также уберите подчёркивание (text-decoration).

- Назначьте элементам списка категорий (правило #MainMenu > li) фоновый цвет (background), внутренний отступ (padding) и рамку справа (border-right).

- Назначьте элементам вложенного списка ссылок (правило #MainMenu li li) такой же фоновый цвет, как и в списке категорий, а также небольшой

отступ и рамку снизу. Кроме того, уберите маркировку списка (list-style-type).

- Уберите у списка ссылок (правило #MainMenu li ul) поля и отступы (margin и padding).

Небольшой проблемой является то, что некоторые элементы списка категорий увеличиваются в ширину при наведении на них указателя. Это является следствием того, что ширина элемента списка определяется шириной всего содержимого - включая вложенный список. Однако если выбросить вложенный список из нормального потока и позиционировать его абсолютно, то его ширина более не будет влиять на ширину родительского элемента. Поэтому укажите для списка ссылок абсолютное позиционирование, а для элементов списка категорий - относительное. Проверьте в браузере. Затем уточните положение вложенного списка относительно его контейнера путём задания значения свойств left и top (не опускайте выпадающий список слишком низко, иначе он станет пропадать при попытке выбрать в нём ссылку).

Зачем нужно было позиционировать элементы списка категорий относительно? Уберите соответствующее правило, и вам всё станет ясно.

6. Добавьте последний штрих: пусть элементы обоих списков при наведении указателя немного изменяют цвет фона (правило #MainMenu li:hover).

Окончательный результат должен быть подобен следующему:

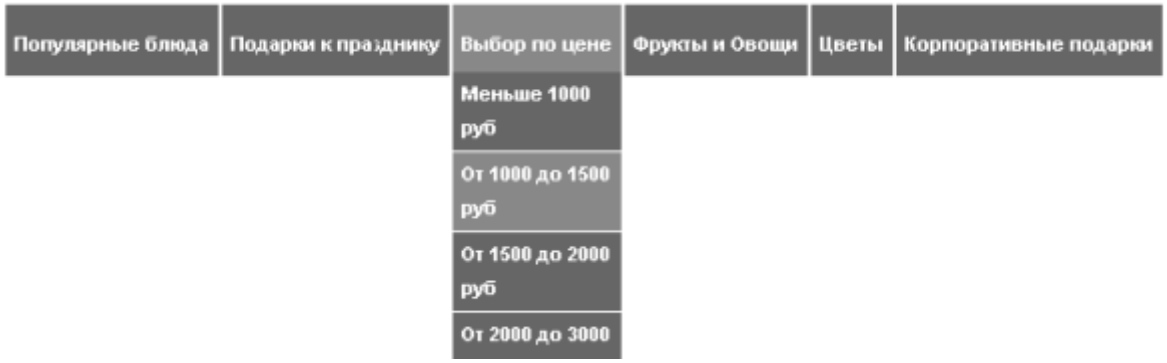
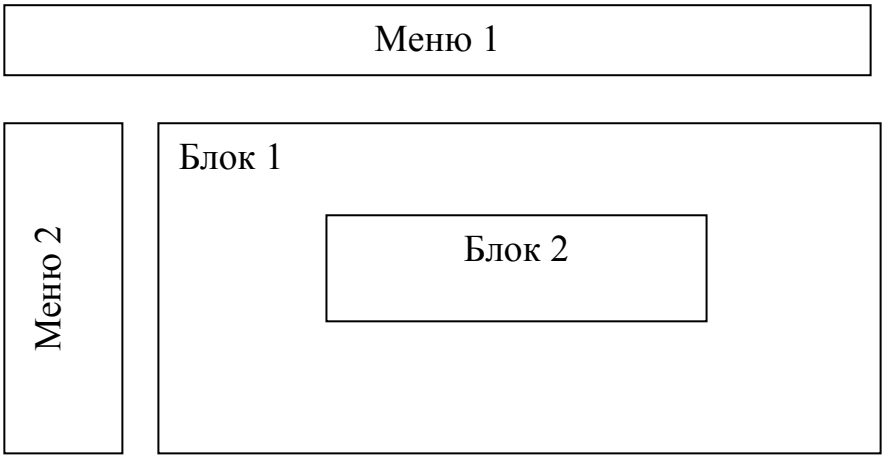


Рис. 7.2. Меню с выпадающими списками ссылок в действии

Задание на самостоятельную работу

1. Создать макет web – страницы.

Вариант 1



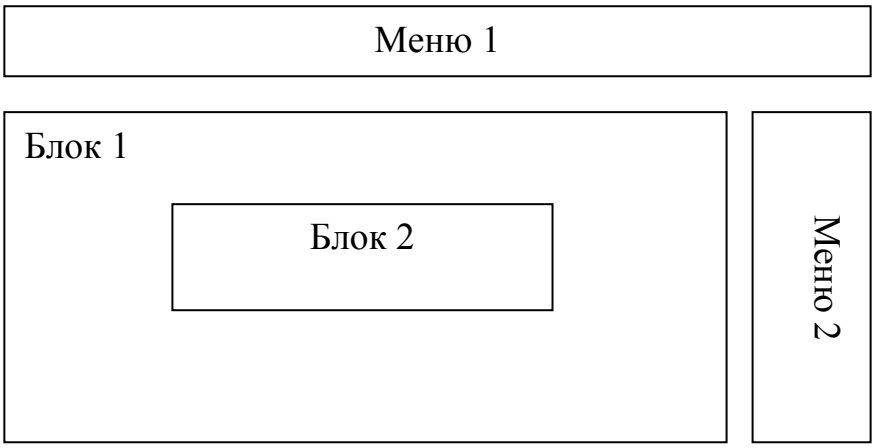
Меню1 двухуровневое, с горизонтальным расположением категорий; списки выпадают вниз;

Меню2 двухуровневое, с вертикальным расположением категорий; списки выпадают справа и вниз;

Блок 1 используется как фон для блока 2;

Блок 2 текстовый, располагается в центре блока 1.

Вариант 2



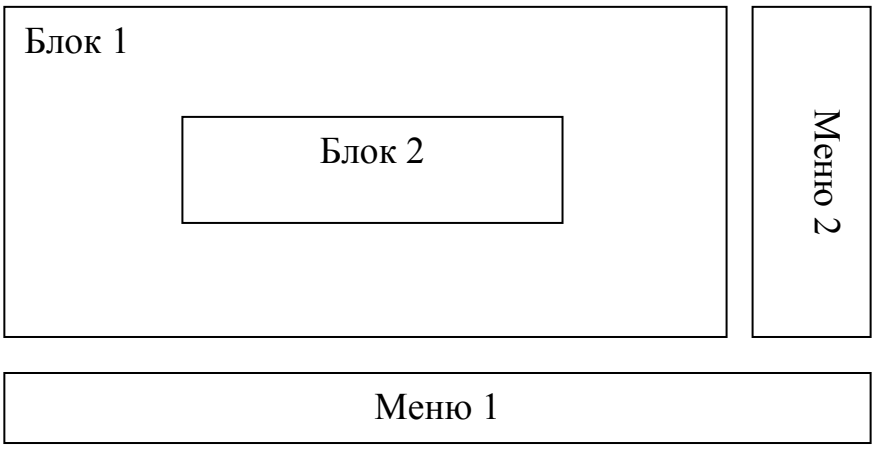
Меню1 двухуровневое, с горизонтальным расположением категорий;

Меню2 двухуровневое, с вертикальным расположением категорий; списки выпадают слева и вниз;

Блок 1 используется как фон для блока 2;

Блок 2 текстовый, располагается в центре блока 1.

Вариант 3



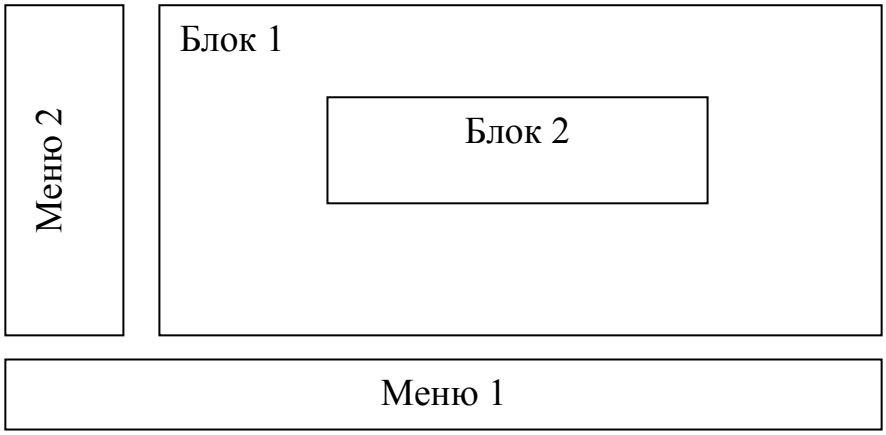
Меню1 двухуровневое, с горизонтальным расположением категорий; списки выпадают вверх;

Меню2 двухуровневое, с вертикальным расположением категорий; списки выпадают слева и вниз;

Блок 1 используется как фон для блока 2;

Блок 2 текстовый, располагается в центре блока 1.

Вариант 4



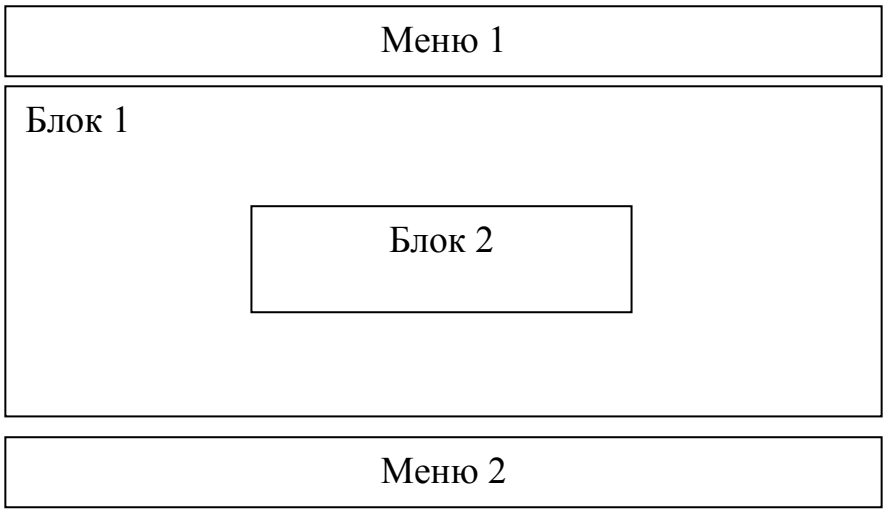
Меню1 двухуровневое, с горизонтальным расположением категорий; списки выпадают вверх;

Меню2 двухуровневое, с вертикальным расположением категорий; списки выпадают справа и вниз;

Блок 1 используется как фон для блока 2;

Блок 2 текстовый, располагается в центре блока 1.

Вариант 5



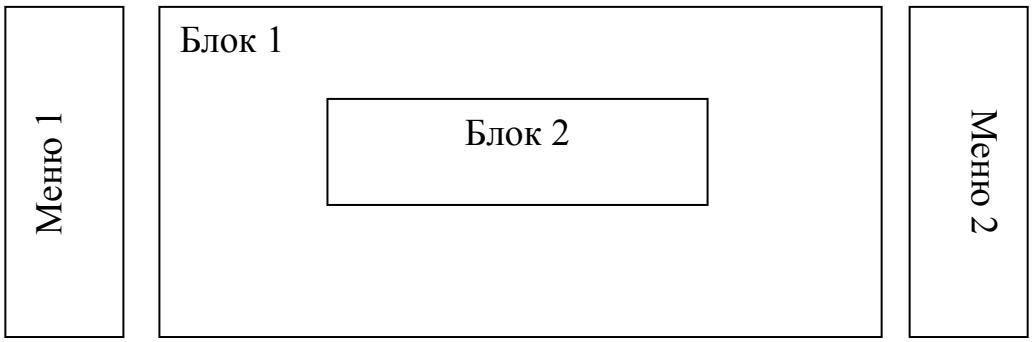
Меню1 двухуровневое, с горизонтальным расположением категорий; списки выпадают вниз;

Меню2 двухуровневое, с горизонтальным расположением категорий; списки выпадают вверх;

Блок 1 используется как фон для блока 2;

Блок 2 текстовый, располагается в центре блока 1.

Вариант 6



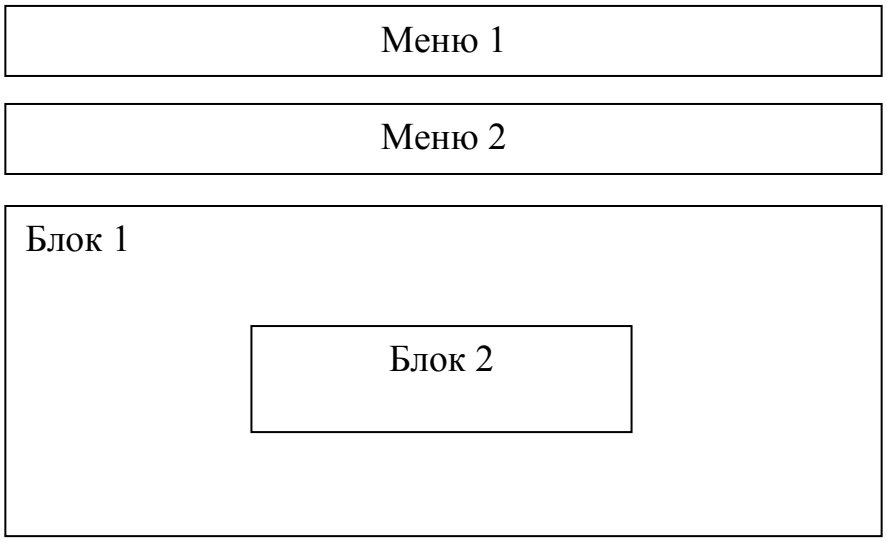
Меню1 двухуровневое, с вертикальным расположением категорий; списки выпадают справа и вниз;

Меню2 двухуровневое, с вертикальным расположением категорий; списки выпадают слева и вниз;

Блок 1 используется как фон для блока 2;

Блок 2 текстовый, располагается в центре блока 1.

Вариант 7



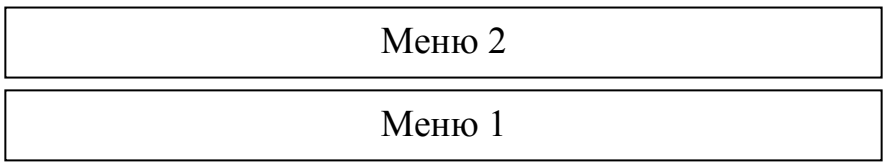
Меню1 двухуровневое, с горизонтальным расположением категорий; списки выпадают вниз;

Меню2 двухуровневое, с горизонтальным расположением категорий; списки выпадают вниз;

Блок 1 используется как фон для блока 2;

Блок 2 текстовый, располагается в центре блока 1.

Вариант 8



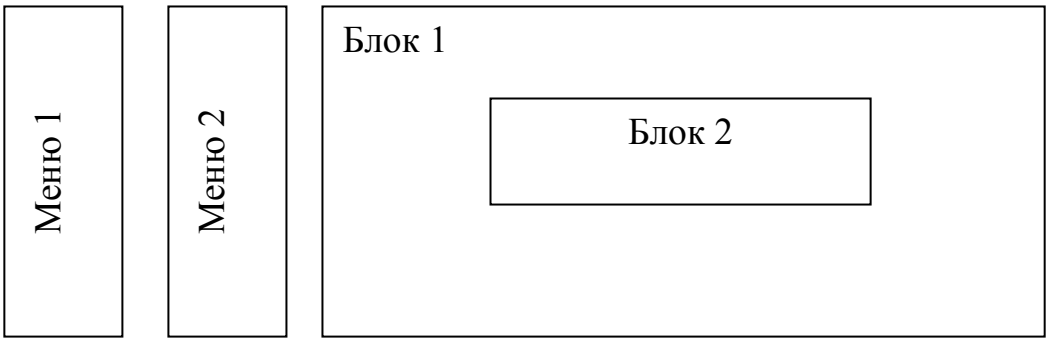
Меню1 двухуровневое, с горизонтальным расположением категорий; списки выпадают вверх;

Меню2 двухуровневое, с горизонтальным расположением категорий; списки выпадают вверх;

Блок 1 используется как фон для блока 2;

Блок 2 текстовый, располагается в центре блока 1.

Вариант 9



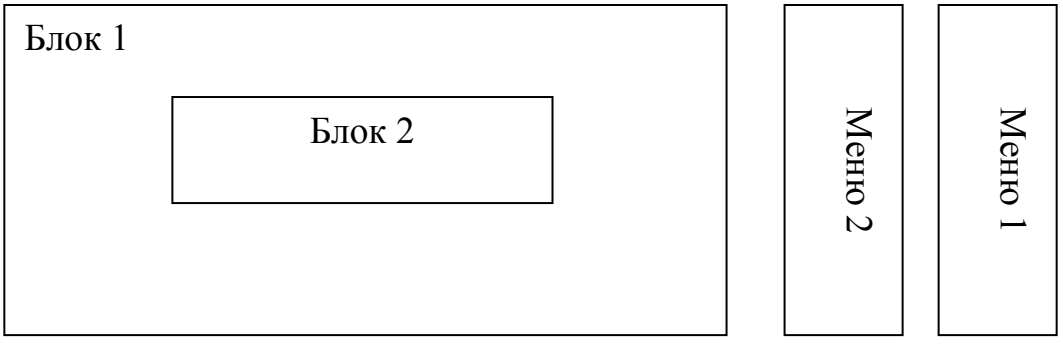
Меню1 двухуровневое, с вертикальным расположением категорий; списки выпадают справа и вниз;

Меню2 двухуровневое, с вертикальным расположением категорий; списки выпадают справа и вниз;

Блок 1 используется как фон для блока 2;

Блок 2 текстовый, располагается в центре блока 1.

Вариант 10



Меню1 двухуровневое, с вертикальным расположением категорий; списки выпадают слева и вниз;

Меню2 двухуровневое, с вертикальным расположением категорий; списки выпадают слева и вниз;

Блок 1 используется как фон для блока 2;

Блок 2 текстовый, располагается в центре блока 1.

3.4. Лабораторная работа №3. Взаимодействие с пользователем на стороне клиента

Введение

В WWW вся информация размещается на Web - страницах, написанных на языке HTML или его расширениях. В содержимое Web - страницы может входить текстовая информация, ссылки на другие Web - страницы, графические изображения, аудио- и видеоинформация и другие данные. Эти страницы хранятся на Web - серверах.

Для доступа к Web - страницам используются специальные клиентские программы - обозреватели (браузеры), находящиеся на компьютерах пользователей Интернета. Обозреватель формирует запрос на получение требуемой страницы или другого ресурса с помощью специального URL - адреса. В задачу обозревателя входит отображение Web - страниц, которые формирует Web - сервер. При этом обозреватель устанавливает соединение с требуемым Web - узлом, используя протокол передачи данных HTTP.

Для расширения возможностей клиентской части (обозревателя) и серверной части создаются программы расширения обозревателя и сервера. Схема взаимодействия обозревателя и сервера с использованием программ расширения приведена на рисунке 1.

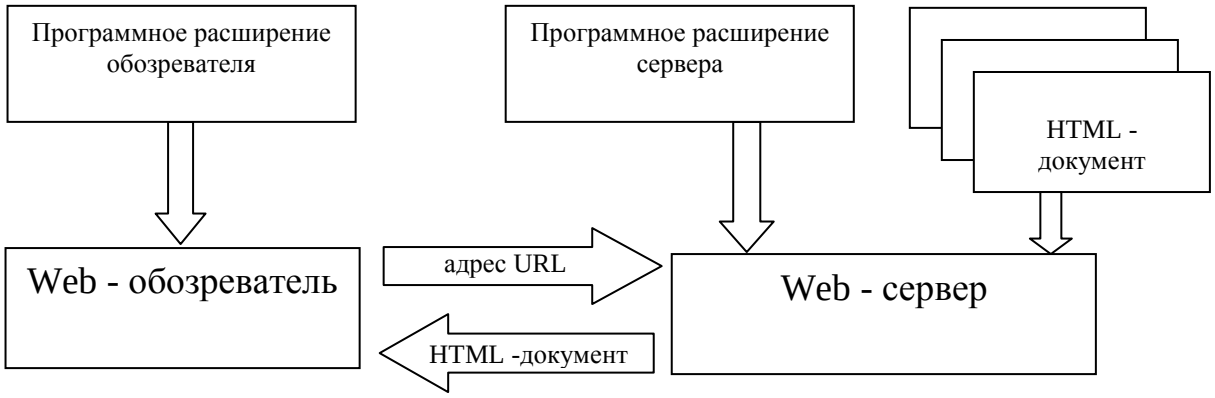


Рис.1. Схема взаимодействия обозревателя и сервера.

Для организации такого взаимодействия могут использоваться следующие средства:

- сценарии, подготовленные на различных языках сценариев (JavaScript, VBScript) и вставляемые в обычный Web - документ;
- аплеты и сервлеты Java;
- элементы управления ActiveX;
- программы, реализованные с использованием интерфейса CGI;
- динамические библиотеки, реализованные с использованием интерфейса ISAPI;
- динамические страницы IDC/HTX, ASP, PHP.

1.Взаимодействие с пользователем на стороне клиента

Целью данного задания является ознакомление с существующими Internet технологиями, предоставляющими следующие возможности:

- разработка и использование в HTML-документах пользовательских форм;
- расширение клиентского программного обеспечения средствами

предварительной обработки пользовательских форм.

2. Создание и использование пользовательских форм

Формы HTML являются основным средством организации интерактивного взаимодействия в Интернете. Практически не всех сайтах, куда бы Вы не зашли, Вы встретите формы, которые предлагается заполнить. Они служат для пересылки данных от пользователя к Web-серверу. С помощью элементов HTML-документа можно создавать простые формы, предполагающие ответы типа "да" и "нет", а также разрабатывать сложные формы для заказов или для того, чтобы получить от пользователей какие-либо комментарии и пожелания и т. д.

В HTML-документе формы задаются с помощью контейнерного тега <form> и состоят из нескольких полей ввода, которые обозреватель отображает в виде графических элементов управления: кнопок выбора, переключателей, строк ввода текста, управляющих кнопок и т. д. Для создания полей ввода предназначены теги <input>, <select>, <textarea>, которые задаются внутри тега <form>.

Тег <form> в общем виде записывается в следующем формате:

<form action="url" метнod=”метод_передачи”
name=”имя_формы”, onsubmit="имя_программы" >
содержимое формы (теги – дескрипторы)
</form>

Параметр action является обязательным. В качестве его значения задается адрес URL программы, которая будет обрабатывать извлеченную из формы информацию.

Параметр method определяет метод пересылки данных формы от обозревателя к Web-серверу и может принимать значение get или post. Метод get более прост, но имеет ряд ограничений, поэтому мы далее будем использовать более универсальный метод post.

Параметр name задает имя формы.

Параметр onsubmit определяет программу, написанную на языках JavaScript или VBScript, которая выполняется до отправки данных формы на сервер. Используется обычно для проверки правильности заполнения клиентом формы.

Теги – дескрипторы

Рассмотрим использование тегов-дескрипторов <textarea>, <select> и <input>, применяемых в HTML для создания интерфейсных объектов внутри формы.

Теги-дескрипторы имеют следующее назначение:

• <textarea> позволяет вводить многострочную текстовую информацию;

• <select > используется для выбора нужной строки в окне с полосой прокрутки либо в раскрывающемся меню;

• <input> является многоцелевым тегом и позволяет вводить текстовую информацию в виде строки, создавать интерфейсные объекты в виде кнопок по установке и сбросу флагов (checkboxes) и переключателей (radiobuttons), обычные кнопки и другие объекты.

Тег<textarea>

Тег <textarea> является контейнерным и используется для создания внутри формы поля ввода многострочного текста. Это поле может отображаться в виде ограниченной прямоугольной области с горизонтальной и вертикальной полосами прокрутки. Тег имеет следующий формат:

<textarea name=”имя” rows=”число” cols= “число” >

…

Текст по умолчанию

…

</textarea>

Атрибуты rows и cols определяют число строк и число столбцов видимого текста соответственно. Между открывающим и закрывающим тегами может быть помещен текст, отображаемый в поле ввода по умолчанию.

Тег <select>

Тег <select> является контейнерным и позволяет организовать внутри формы выбор одного или нескольких значений (параметров) из списка значений.

Тег <select> записывается в следующем формате:

<select name=”имя_поля” size=число multiple>

…

Элементы меню или списка

…

</select>

Здесь: параметр size определяет количество видимых элементов выбора, при size=1 используется раскрывающееся меню, при size =”число” большем 1 используется список прокрутки, для которого "число" определяет количество видимых элементов; задание параметра multiple позволяет выполнять выбор из меню или списка одновременно несколько элементов.

Элементы меню в теге <select> задаются с помощью тега <option>, записываемого в следующем формате:

<option selected value=”значение”> Текст

<option> Текст

<option> Текст

…

Атрибут selected задает выбор элемента по умолчанию. Параметр value содержит значение, пересылаемое серверу, если данный элемент выбран в меню или списке. Текст, находящийся в конце тега, выводится на экран обозревателя на месте элемента в списке.

Тег<input>

Тег <input> предназначен для создания внутри формы различных кнопок, полей для ввода текста, пароля, имен файлов. Тег имеет следующий формат:

<input type=”тип_поля_ввода” name=”имя_поля_ввода” дополнительные_параметры>

Первые два параметра являются обязательными. Параметр type определяет тип поля ввода: кнопка выбора, кнопка передачи и др. Параметр name задает имя поля, которое используется как идентификатор передаваемого Web-серверу значения. Состав дополнительных параметров зависит от значения параметра type, определяющего тип поля.

Атрибут type тега <input> может принимать следующие значения:

• text — является значением по умолчанию. В этом случае создается интерфейсный элемент в виде одной строки для ввода данных. Дополнительные параметры: value=значение, отображаемое в поле по умолчанию; size=n, где n - максимальное количество отображаемых символов;

• password — аналогичен предыдущему типу, но заменяет вводимые символы пароля звездочками;

• checkbox — выводит поле для установки или сброса флажка. В этом поле может использоваться атрибут checked, который определяет установленный по умолчанию флаг, а также атрибут value, определяющий строку, которая будет передана серверу, если элемент будет выбран ;

• radio — аналогичен предыдущему типу поля, но имеет отличный внешний вид (аналогичный интерфейсному виду Radio-кнопки — графическому элементу операционной

системы Windows) и не позволяет отменять сделанный выбор. Эти элементы используются группой, из которых может быть выбран только один. Все элементы группы должны иметь общее имя (name);

• reset — позволяет создать кнопку для обновления формы. Атрибут value позволяет задать надпись на кнопке;

• submit — используется для создания кнопки, по нажатию которой введенные данные отправляются на сервер для обработки программой-сценарием.

Атрибут name у всех типов используется для получения доступа к заданному элементу HTML-документа.

Рассмотрим пример создания формы:

```
<form>
Введите ваше имя:
<input type="text" name="My" value="Имя" size="10" maxlength="20"><br>
Введите пароль:
<input type="password" name="pass1" Size="30" maxlength="30"><BR>
Выберите тип действия: <br>
<input type="checkbox" name="mycheckboxl" value="Параметр1" checked> Действие 1
<input type="checkbox" name="mycheckbox2" value="Параметр2"> Действие 2 <br>
<input type="radio" name="my1" value=" Параметр2"> Действие 3
<input type="radio" name="my1" value =" Параметр3"> Действие 4 <br>
<textarea name="My_textarea " rows="4" cols="40">
Справочный текст, внутри которого <br>
можно ввести собственные данные <br>
Эта форма посылает данные серверу
</textarea> <br>
<select name="menu">
<option selected value ="Действие1"> Действие1
<option value=" Действие2"> Действие2
<option value=" Действие3"> Действие3
</select>
<input type="reset" value="Reset ">
<input type="submit" value ="Послать данные!">
</form>
```

Сформируйте HTML – документ (см. задание 1), включите в него текст примера формы и посмотрите результат в обозревателе (Рис.2).

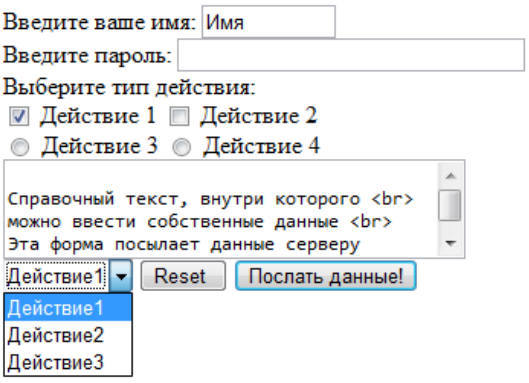


Рис.2. Окно обозревателя с текстом примера.

3. Проверка правильности заполнения формы на стороне клиента

Как было сказано выше, основное назначение форм - позволить клиенту заполнить требование собственника сайта на предоставление некоторой услуги. После заполнения форма отсылается на сервер, где и обрабатывается. Однако, если по какой - то причине, клиент неправильно заполнил форму, то сервер будет вынужден вновь запросить данные от клиента. Такой обмен данными приводит к нерациональному использованию ресурсов сети. Поэтому более логично выполнять проверку правильности заполнения формы перед отправкой ее серверу на клиентской машине.

Такая проверка осуществляется с помощью специальных программ, записанных на языках JavaScript или VBScript. Текст программ помещается в HTML - документ с помощью парного тега <script> </script>.

Пример 1.

Пусть клиенту предлагается небольшая форма с предложением указать свое имя и возраст. Текст HTML - документа представлен ниже.

```
<html>
<head>
<script Language="JavaScript">
<!--
function IsEmpty (data){
  if (data.length==0) return true
  else return false
}
function IsNumber(data){
  var NumStr="0123456789"
  var ch
  var count=0
  for(var i=0; i<data.length; i++){
          ch=data.substring(i,i+1)
          if (NumStr.indexOf(ch)!=-1) count++
  }
  if ((count==data.length)&&(data.length!=0)){
          n=parseInt(data)
          if ((n>5)&&(n<100)) return true
          else return false
```

```
  }
  else return false
}
function IsFormOK(){
        if (IsEmpty(document.F.Myname.value)) {
        alert("Вам следует ввести имя!")
        return false}
        if (!IsNumber(document.F.Myage.value)) {
        alert("Возраст должен быть указан целым числом от5 до100.")
        return false}

        return true


}
//-->
</script>
</head>
<body>
<form name="F" onSubmit="return IsFormOK()">
Введите ваше имя:
<input type="text"  name="Myname"  ><BR>
Введите Ваш возраст:
<input type="text"  name="Myage"  ><BR>
<input type="submit" VALUE ="Послать данные!">
</form>
</body>
</html>
```

Рассмотрим основные части этого документа. Его внешний вид определяет форма с именем F. В ней задаются два поля для ввода имени и возраста клиента и кнопка "Submit" для отправки данных на сервер (В целях упрощения изложения адрес обрабатывающего расширения на сервере не указан, т.е. атрибут action опущен). Перед отправкой данных формы на сервер производится проверка правильности заполнения формы с помощью атрибута

onSubmit="return IsFormOK()".

Для выполнения проверки вызывается функция IsFormOK(). Функция IsFormOK() и сопутствующие функции написаны на языке JavaScript. Тексты всех функций помещены между тегами <script> и </script>.

Главной функцией является функция IsFormOK().

```
function IsFormOK(){
        if (IsEmpty(document.F.Myname.value)) {
        alert("Вам следует ввести имя!")
        return false}
        if (!IsNumber(document.F.Myage.value)) {
        alert("Возраст должен быть указан целым числом от5 до100.")
        return false}
        return true
```

```
}
```

Для доступа к полям формы используется следующая конструкция:

```
document.имя_формы.имя_поля.value;
```

То есть сначала обращаемся к **объекту Document**, затем к его **дочернему объекту Form** (через имя формы), потом к **имени поля** данной формы, и, наконец, к **значению поля**.

Функция IsFormOK() включает два условных оператора проверки правильности заполнения текстовых полей формы. Первый проверяет на пустоту поле, содержащее имя клиента. Эта проверка осуществляется вызовом функции IsEmpty (document.F.Myname.value), в качестве аргумента которой взято значение поля Myname.

```
if (IsEmpty(document.F.Myname.value)) {
        alert("Вам следует ввести имя!")
        return false}
```

В случае если функция IsEmpty вернет значение истина, т.е. значение поля Myname будет пустым, то клиенту будет выдано сообщение "Вам следует ввести имя!" (оператор alert), и функция IsFormOK() примет значение ложь (оператор return false). В результате форма не будет отослана серверу.

Аналогично проверяется правильность заполнения поля возраст. Здесь используется функция IsNumber(document.F.Myage.value), которая проверяет, что пользователь указал в поле возраст (там должно быть целое число в интервале от 5 до 100). Восклицательный знак перед именем функции в условном операторе означает отрицание. Другими словами, запись

if (!IsNumber(document.F.Myage.value)) означает: "Если значение поля Myage не является числом, то".

В случае если обе проверки окончились благополучно, функция IsFormOK() примет значение истина (оператор return true) и данные формы будут отосланы на сервер.

Рассмотрим подробнее функции IsEmpty и IsNumber.

Функция IsEmpty проверяет на пустоту строковый аргумент.

```
function IsEmpty (data){
  if (data.length==0) return true
  else return false
}
```

Для этих целей использован условный оператор и свойство length - длина. Таким образом запись

if (data.length==0) означает: "Если длина строки data равна нулю, т. е. если строка пустая, то...".

Функция IsNumber проверяет, является ли строковый аргумент записью числа в интервале от 5 до 100.

```
function IsNumber(data){
  var NumStr="0123456789"
  var ch
  var count=0
  for(var i=0; i<data.length; i++){
        ch=data.substring(i,i+1)
        if (NumStr.indexOf(ch)!=-1) count++
  }
  if ((count==data.length)&&(data.length!=0)){
```

```
            n=parseInt(data)
            if ((n>5)&&(n<100)) return true
            else return false
        }
        else return false
    }
```

Первые три оператора служат для описания и инициализации внутренних переменных:

```
    var NumStr="0123456789"
    var ch
    var count=0
```

Переменная NumStr содержит перечень всех арабских цифр, переменная ch будет содержать значение очередного символа исходной строки, переменная count будет содержать количество цифр в исходной строке.

Далее с помощью оператора цикла

for(var i=0; i<data.length; i++){

организован просмотр по очереди всех символов исходной строки data. В качестве переменной цикла используется переменная i. Запись i++ означает, что с каждым проходом цикла переменная i должна увеличиваться на 1.

В цикле с помощью метода substring в переменную ch записывается очередной i - й символ исходной строки data.

ch=data.substring(i,i+1)

Далее с помощью метода indexOF проверяется, является ли символ, записанный в ch цифрой. Если да, то переменная count увеличивается на 1.

if (NumStr.indexOf(ch)!=-1) count++

На этом циклический участок заканчивается.

```
for(var i=0; i<data.length; i++){
            ch=data.substring(i,i+1)
            if (NumStr.indexOf(ch)!=-1) count++
    }
```

После выполнения цикла переменная count будет содержать количество цифр в исходной строке. Поэтому если сравнить ее значение с длиной исходной строки, можно узнать все ли символы в исходной строке - цифры. Вторым условием правильной записи числа является наличие хотя бы одной цифры в его записи. Для проверки этих двух условий служит оператор

if ((count==data.length)&&(data.length!=0)){

Здесь && обозначает логическое И.

Если число записано верно, то остается проверить лежит ли оно в диапазоне от 5 до 100. Выполняется эта проверка следующими операторами:

```
        n=parseInt(data)
        if ((n>5)&&(n<100)) return true
        else return false
```

Внутренняя функция parseInt(data) служит для перевода числа из исходной строковой записи в числовую.

Пример 2.

Исходная форма:

```
    <form name = 'form1' action = 'handler.php' method = 'post'
onSubmit=”return  check(document.form1)”>
  Ваше имя:
  <input type = 'text' name = 'firstname' />
```

```
  <br />
  Ваш пароль:
  <input type = 'password' name = 'pass' />
  <br />
  Ваш пол:
  <input type = 'radio' name = 'sex' value = 'male' checked />M
  <input type = 'radio' name = 'sex' value = 'female' />F
  <br />
  Выберите число:
  <select name = 'number'>
    <option selected value = '1'>1</option>
    <option value = '2'>2</option>
    <option value = '3'>3</option>
  </select>
  <br />
  Ваше сообщение:
  <textarea name = 'message'></textarea>
  <br />
  Согласитесь с нашими правилами:
  <input type = 'checkbox' name = 'rules' />
  <br />
  <input type = 'hidden' value = 'hidefield' />
  Загрузите файл:
  <input type = 'file' name = 'fileupload' />
  <br />
  <input type = 'submit' value = 'Отправить' name = 'sub' />
</form>
```

В теге <form> атрибут "onSubmit" со значением "return check(document.form1); " делает следующее: перед отправкой формы вызывает функцию, которая должна будет вернуть либо true, либо false. Если она вернёт true, то форма отправится на сервер, а если false, то форма не будет отправлена.

Теперь создадим функцию check(), которая должна, во-первых, проверять полностью форму, сообщать пользователю об ошибках, а также возвращать true (если форма полностью правильная), либо false (если форма содержит ошибки).

```
    function check(form) {
 var firstname = form.firstname.value;
      var pass = form.pass.value;
      var message = form.message.value;
      var rules;
      var file = form.fileupload.value;
      var select=form.number.value;
      var sex1;

      if (document.form1.sex[0].checked) sex1 = document.form1.sex[0].value;
    else sex1 = document.form1.sex[1].value;
    if (document.form1.rules.checked) rules = 'yes';
    else rules = 'no';
  var file = form.fileupload.value;
  var bad = "";
  if (firstname.length < 3)
     bad += "Имя слишком короткое" + "\n";
  if (firstname.length > 32)
    bad += "Имя слишком длинное" + "\n";
  if (pass.length < 3)
    bad += "Пароль слишком короткий" + "\n";
  if (pass.length > 32)
    bad += "Пароль слишком длинный" + "\n";
  if (message.length < 3)
    bad += "Сообщение слишком короткое" + "\n";
  if (rules != "yes")
```

```
    bad += "Вы не согласились с правилами" + "\n";
  if (file.length == 0)
    bad += "Вы не выбрали файл для загрузки" + "\n";
  if (bad != "") {
    bad = "Неверно заполнена форма:" + "\n" + bad;
    alert(bad);
    return false;
  }
  return true;
}
```

В данном скрипте мы сначала получаем все данные, нуждающиеся в проверке, и записываем их в переменные. Затем создаём переменную bad, в которую будем записывать все некорректные данные. Затем идёт целый набор IF, которые проверяют поля в форме и записывают все ошибки в переменную bad. Затем, если переменная bad не пустая (то есть были ошибки), то выводим окно (alert()) с ошибками. И возвращаем false, чтобы форма не была отправлена. Если переменная bad пустая, то дальше просто возвращаем true, чтобы форма была отправлена уже на обработку на сервер в "handler.php".

Задания на самостоятельную работу

1. Выполнить пример 1
2. Выполнить пример 2
3. Выполнить индивидуальное задание согласно варианту, который определяется по последней цифре зачетной книжки (индивидуального плана) студента.
4. Подготовить отчет, включающий: постановку задачи в соответствии с вариантом, HTML – текст с текстами функций, скриншоты, ответы на контрольные вопросы.

Варианты индивидуальных заданий.

Подготовить форму, содержащую следующие элементы, и осуществить проверку корректности их заполнения на стороне клиента:

№варианта	Элементы формы	Условия корректности
1	Text1	Непусто, Целое число от 20 до 50
	Text2	Непусто, Кол-во символов >5
2	Password1	Непусто
	Password2	Непусто, Password1= Password2
3	Checkbox1	
	Checkbox2	Один или оба флажка должны быть включены
4	Checkbox	Флажок должен быть включен
	Textarea	Содержит не менее 10 символов
5	Select из опций, значениями которых являются буквы:’a’,’b’,’c’	Выбрана буква ‘b’
	Checkbox	Флажок должен быть выключен
6	Text1	Непусто, Целое число меньше 5000
	Password1	Целое число
7	Checkbox	Флажок должен быть выключен
	Textarea	Содержит не более 20 символов
8	Radio из трех элементов	Выбран второй переключатель
	Text	Непусто, Кол-во символов от 5 до 10

9	Select из опций, значениями которых являются названия месяцев в году	Выбран «сентябрь»
	Textarea	Непусто
10	Text	Содержит строку «www». *Использовать функцию indexOf(«искомая строка»)*.

Контрольные вопросы

1. Для чего предназначен контейнерный тег FORM?
2. Назначение атрибута ACTION.
3. Назначение и правила определения кнопки SUBMIT.
4. В каких случаях используют скриптовые языки JavaScript и VBScript?
5. Как правильно оформить процедуру проверки правильности заполнения формы клиентом?
6. Для чего предназначена функция alert()?
7. Почему неправильно заполненную форму лучше обработать сразу на стороне клиента?

3.5.Лабораторная работа 4. Обработка событий при помощи сценариев Javascript

Основные теоретические сведения

Введение

Практически все JavaScript-приложения выполняют те или иные действия, откликаясь на различные события.

Событие - это сигнал от браузера о том, что что-то произошло.

Есть множество самых различных событий:

- DOM-события, которые инициируются элементами DOM. Например, событие click происходит при клике на элементе, а событие mouseover - когда указатель мыши появляется над элементом,
- События окна. Например событие resize - при изменении размера окна браузера,
- Другие события, например load, readystatechange. Они используются, скажем, в технологии AJAX.

Именно DOM-события связывают действия, происходящие в документе, с кодом JavaScript, тем самым обеспечивая динамический веб-интерфейс.

Назначение обработчиков

Для того, чтобы скрипт реагировал на событие - нужно назначить хотя бы одну функцию-обработчик. Обычно обработчики называют "on+имя события", например: onclick.

JavaScript - однопоточный язык, поэтому обработчики всегда выполняются последовательно и в общем потоке. Это значит, что при установке обработчиков двух событий, которые возникают на элементе одновременно, например mouseover (мышь появилась над элементом) и mousemove (мышь двигается над элементом), их обработчики будут выполнены последовательно.

Существует несколько способов назначить обработчик на конкретное событие элемента.

1. Через атрибут HTML-тега.

Список атрибутов, которые определены для HTML тэгов приводится ниже:

onabort	Прерванная загрузка изображения
onblur	утрата фокуса элементом
onchange	Изменение содержимого в поле ввода
onclick	Щелчок мыши на объекте
onerror	Ошибка при загрузке изображения или документа
onfocus	Получение фокуса элементом
onkeypress	Клавиша нажата
onload	Завершение загрузки страницы или изображения
onmousedown	Нажатие кнопки мыши
onmousemove	Перемещение курсора мыши
onmouseover	Наведение курсора мыши на объект
onreset	Кнопка "Reset" нажата
onresize	Изменение размера окна
onsubmit	Кнопка "Submit" нажата
onunload	Уход с веб-страницы

Обработчик события можно указать в виде inline-записи, прямо в атрибуте в виде «on+событие».

Например, для обработки события click на кнопке input, можно назначить обработчик onclick так:

```
<input id="b1" value="Нажми Меня" onclick="alert('Спасибо!');" type="button"/>
```

Можно назначить и функцию. Например, пусть при клике на кнопку input запускается функция count_rabbits(). Для этого запишем вызов функции в атрибут onclick:

```
<script type="text/javascript">
   function count_rabbits() {
      …
   }
</script>

<body>
   <input type="button" onclick="count_rabbits()" value="Считать кролей!"/>
</body>
```

Напомним, что имена атрибутов HTML-тегов нечувствительны к регистру, поэтому атрибут oNcLiCk сработает также, как onClick или onclick.

Но если вы хотите придерживаться хорошего стиля (или спецификации XHTML), то имена тегов и их атрибуты должны быть указаны в нижнем регистре.

2. Через свойство объекта

Для этого нужно:

- получить элемент;
- назначить обработчик свойству «on+имя».

Вот пример установки обработчика события click на элемент с id="myElement":

```
document.getElementById('myElement').onclick = function() {
```

```
    alert('Спасибо');}
<input id="myElement" type="button" value="Нажми меня"/>
```

Объект "событие" (event)

Объект событие всегда передается обработчику и содержит массу полезной информации о том, где и какое событие произошло. Из объекта события обработчик может узнать, на каком элементе оно произошло, каковы были координаты мыши (для событий, связанных с мышью), какая клавиша была нажата (для событий, связанных с клавиатурой), и извлечь другую полезную информацию.

Например, для события по клику мыши (onclick), свойство event.target(в IE event.srcElement) содержит DOM-элемент, на котором этот клик произошел.

Способы передачи этого объекта обработчику существует зависят от браузера.

1) В браузерах, работающих по рекомендациям W3C (Mozilla, Opera и др.), объект события всегда передается в обработчик первым параметром.

Например:

```
function doSomething(event) {
   // event - будет содержать объект события
}
element.onclick = doSomething;
```

При вызове обработчика объект события event будет передан ему первым аргументом.

Можно назначить и вот так:

```
element.onclick = function(event) {
  // event - объект события
}
```

2) В Internet Explorer существует глобальный объект window.event, который хранит в себе информацию о последнем событии. А первого аргумента обработчика просто нет.

То есть, все должно работать так:

```
// обработчик без аргументов
function doSomething() {
   // window.event - объект события
}

element.onclick = doSomething;
```

3) Кросс-браузерное решение

Можно кросс-браузерно получить объект события, использовав такой приём:

```
function doSomething(event) {
    event = event || window.event

    // Теперь event - объект события во всех браузерах.
}
```

```
element.onclick = doSomething
```

Практическая часть

В данной работе предлагается создать HTML-таблицу, представляющую калькулятор, и создать ряд обработчиков событий - нажатий пользователя на кнопки, которые управляют логикой работы калькулятора.

Часть 1. Создание разметки

1. Создание таблицы знаков калькулятора.

То, как предлагается оформить калькулятор в данной работе, показано на рис.1

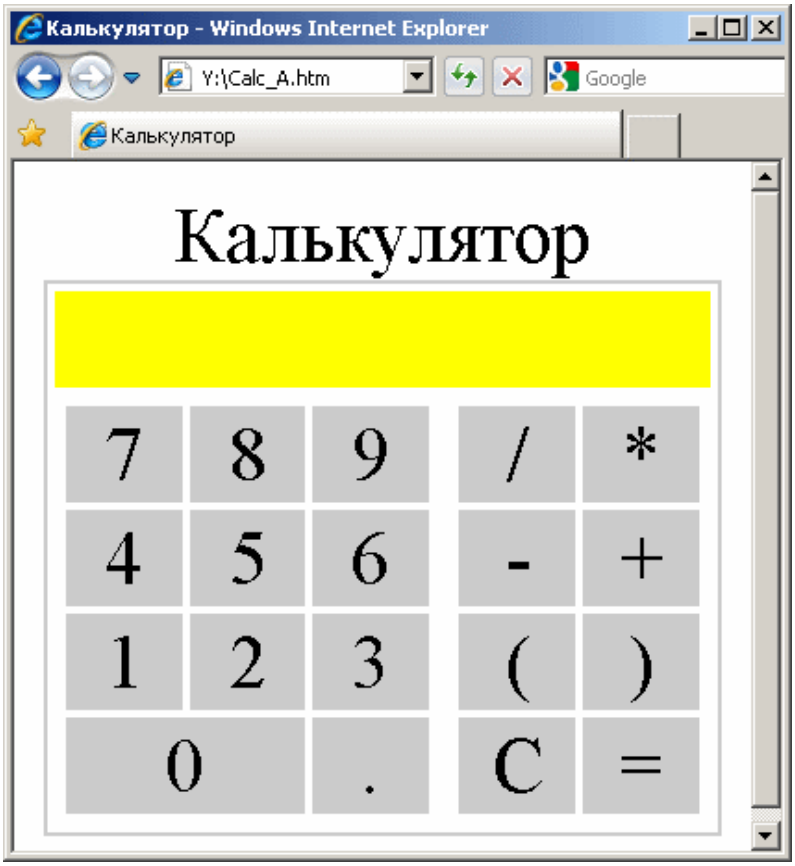


Рис. Законченный вид таблицы-калькулятора

Очевидно, таблица - наиболее подходящий элемент для представления набора элементов управления в виде выровненной сетки. Регулярность таблицы нарушается в нескольких местах: ячейка для результата шире всех остальных; клавиши операций отделены от цифр; нуль занимает две смежных ячейки. С точки зрения поведения также имеются отдельные смысловые группы: клавиши цифр должны обрабатываться одним образом; кнопки сброса и знак равенства - другим; ячейка результата клавишей не является. Поэтому сгруппируем элементы визуально и функционально в отдельные таблицы следующим образом.

Сначала создайте в новом файле с именем Lab4.htm **таблицу знаков** следующего вида:

/	*
-	+
(	)
C	=

Ячейкам таблицы (элементам td) назначено стилевое правило, устанавливающее серый фон, выравнивание текста по центру, ширину и курсор в виде указателя:

```
cursor: pointer;
```

2. Затем создайте таблицу цифр:

7	8	9
4	5	6
1	2	3
0		.

Здесь нуль занимает две смежных ячейки, и необходимо использовать атрибут colspan

3. Создание внешней таблицы.

Теперь создайте таблицу с двумя строками. Во второй строке создайте две ячейки - в первую переместите таблицу цифр, а во вторую - таблицу знаков. В первой строке нужна всего одна ячейка - для результата - на всю ширину строки. Воспользуйтесь еще раз атрибутом colspan.

Ячейке результата назначьте атрибут id="result" и создайте для нее стиль (используя селектор идентификатора), назначив подходящий цвет фона, высоту и выравнивание текста по правому краю.

Внешней таблице добавьте тонкую границу серого цвета и заголовок (элемент caption).

Часть 2. Создание обработчиков событий

1. Обработка цифр и знаков.

Назначьте таблицам цифр и знаков атрибут id со значениями digits и signs соответственно. Ячейке с кнопкой сброса назначьте id='clear', а ячейке с кнопкой '=' - id='calculate'.

Вставьте в раздел head документа скрипт, определяющий следующую функцию.

```
//функция, выполняемая автоматически по окончании загрузки страницы
window.onload = function() {
    //переменные, определенные здесь, доступны и во вложенных функциях
    var oDigits = document.getElementById('digits');
```

```
        var oSigns = document.getElementById('signs');
        var oResult = document.getElementById('result');
        var oClear = document.getElementById('clear');
        var oCalc = document.getElementById('calculate');

        //Обрабатываем щелчок мыши по цифрам и знакам
        oDigits.onclick =
        oSigns.onclick =
        function(e) {//Mozilla FireFox и ряд других браузеров отправляет в
обработчик событий объект-событие
                //Internet Explorer не отправляет ничего
                var elem = null; //элемент, на котором произошло событие
                if (e) elem = e.target; //работает в FireFox
                else elem = window.event.srcElement; //работает в Internet
Explorer
                if (elem && elem.tagName.toLowerCase() == 'td') {
                        oResult.innerHTML += elem.innerHTML;
                }
        }
    }
```

Принципиально весь приведенный код делает две вещи: определяет действия, которые нужно проделать сразу по завершению загрузки документа браузером (т.е. при наступлении события window.onload), а также определяет действия, которые нужно выполнять в ответ на щелчок мышью по таблицам цифр и знаков (т.е. по событию onclick, возникающему в этих элементах). Действия представляют собой блоки кода, помещенные в функцию. Каждый оператор в блоке кода имеет свой смысл - в коде он пояснен комментариями, которые следует внимательно изучить.

Проверьте работу скрипта: при щелчке мышью по обрабатываемым символам они должны добавляться к строке результата.

2. Обработка клавиши "=".

Следом за функцией-обработчиком цифр и знаков (но внутри обработчика onload) добавьте следующую функцию.

```
    //кнопке Равно (=) требуется специальная обработка
    oCalc.onclick = function(e) {
       if (!e) e = window.event;
       e.cancelBubble = true; //запрещаем передачу события родительскому
элементу
       try { //перехватываем возможное исключение
               oResult.innerHTML = eval(oResult.innerHTML); //выполняем расчет
выражения, записанного в строке
       }
       catch (ex) { }
    }
```

Здесь идея заключается в том, что в ячейке result должно находиться арифметическое выражение. Чтобы вычислить его результат, нужно провести разбор выражения и пройти по дереву вычислений, что обычно реализуется относительно громоздким кодом. Но Javascript предоставляет один особенный метод - eval, который выполняет любой код Javascript, в том числе и арифметическое выражение! На случай, если выражение введено неправильно (не соответствуют скобки и т.п.), вызов метода eval следует заключить в блок try/catch - "попытайся"/"перехвати". Если исключительная ситуация возникнет, она будет перехвачена блоком catch, который не делает ничего (главное в том, что исключение считается в таком случае обработанным, и браузер не будет показывать сообщения об ошибке). Можно было бы, например, в этом блоке очистить поле result. Еще один важный момент: обработчик onclick назначен как ячейке calculate, так и содержащей ее таблице signs. Если специально не предотвратить передачу события родительскому элементу, то оно

будет обработано дважды - сначала ячейкой, затем таблицей. Посмотрите, как бы это выглядело.

3. Самостоятельно напишите обработчик кнопки Очистить (C).

4. Последний штрих:

Сделайте, чтобы обрабатывались еще два события - onmousedown и onmouseup - "нажатие" и "отжатие" левой клавиши мыши. Пусть при нажатии цифра (или знак) изменяют цвет, а при отжатии возвращают первоначальный.

Необходимые для этого команды приведены ниже, а обработчики, в которых они будут запускаться, напишите самостоятельно.

```
elem.style.color = 'red';
elem.style.color = '';
```

Задание на самостоятельную работу

Дополните калькулятор панелью вычисления стандартных математических функций в соответствии с вариантом.

Варианты для самостоятельной работы.

№	Стандартные функции	Расположение панели
1	Math.abs, Math.acos, Math.asin, Math.atan	сверху
2	Math.exp, Math.PI, Math.random, Math.sqrt	сверху
3	Math.log, Math.round, Math.floor, Math.ceil	сверху
4	Math.sin, Math.cos, Math.tan, Math.pow	сверху
5	Math.max, Math.E, Math.LN2, Math.LOG2E	сверху
6	Math.min, Math.SQRT12, Math.SQRT2, Math.LN10	сверху
7	Math.abs, Math.PI, Math.floor, Math.pow	снизу
8	Math.acos, Math.random, Math.ceil, Math.sin	снизу
9	Math.atan, Math.max, Math.SQRT12, Math.tan	снизу
10	Math.min, Math.E, Math.tan, Math.abs	снизу
11	Math.LN2, Math.random, Math.ceil, Math.max	снизу
12	Math.cos, Math.pow, Math.LN10, Math.PI	снизу

Примечание. Если нажата кнопка с одноместной функцией, то выражение в окне результата должно меняться, например, на Math.sin (выражение). При вводе двухместных функций дополнительно потребуется кнопка «запятая».

Содержание отчета

1. Титульный лист;
2. Постановка задачи на самостоятельную разработку;
3. Листинг исходных кодов с комментариями;
4. Скриншоты тестовых примеров и их описание;
5. Выводы

Контрольные вопросы

1. Что такое событие?
2. Обработчики событий и способы их задания?
3. Способы передачи информации о событиях в скрипты.
4. Проблемы кросс-браузерности.

3.6.Лабораторная работа 5. Изучение приёмов динамического формирования HTML-документа на стороне клиента

Цель работы. Изучить приёмы динамического формирования HTML-документа на стороне клиента

Основные теоретические сведения

HTML DOM (Document Object Model) – представляет собой стандарт консорциума W3C для программного доступа к документам HTML или XML. Фактически это платформо- и языково-нейтральный интерфейс, позволяющий программам и сценариям динамически обращаться и обновлять содержимое, структуру и стиль HTML-документа.

Согласно модели DOM:

- Весь документ представляется узлом документа;
- Каждый HTML тэг является узлом элемента;
- Текст внутри HTML элементов представляется текстовыми узлами;
- Каждому HTML атрибуту соответствует узел атрибута;
- Комментарии являются узлами комментариев.

Например:

```
<html>
  <head>
    <title>HTML документ</title>
  </head>
  <body>
    <h1>Заголовок </h1>
    <p>Просто текст</p>
  </body>
</html>
```

В этом примере корневым узлом является тэг <html>. Все остальные узлы содержатся внутри <html>. У этого узла имеется два дочерних узла: <head> и <body>. Узел <head> содержит узел <title>, а узел <body> содержит узлы <h1> и <p>.

В рамках DOM модели HTML можно рассматривать как множество узловых объектов. Доступ к ним осуществляется с помощью JavaScript или других языков программирования. Программный интерфейс DOM включает в себя набор стандартных свойств и методов.

К типичным свойствам DOM относятся следующие:

x.innerHTML – внутреннее текстовое значение HTML элемента x ;

x. nodeName – имя x ;

x.nodeValue – значение x ;

x.parentNode – родительский узел для x ;

x.childNodes – дочерний узел для x ;

x.attributes – узлы атрибутов x.

Узловой объект, соответствующий HTML элементу поддерживает следующие методы:

x.getElementById(id) – получить элемент с указанным id ;

x.getElementsByTagName(name) – получить все элементы с указанным именем тэга (name);

x.appendChild(node) – вставить дочерний узел для x ;

x.removeChild(node) – удалить дочерний узел для x.

Например:
Для получения текста из элемента <p> со значением атрибута id "demo" в HTML документе можно использовать следующий код:

```
txt = document.getElementById("demo").innerHTML
```

Тот же самый результат может быть получен по-другому:

```
txt=document.getElementById("demo").childNodes[0].nodeValue
```

В рамках DOM возможны 3 способа доступа к узлам:

1. С помощью метода getElementById(ID). При этом возвращается элемент с указанным ID.
2. С помощью метода getElementsByTagName(name). При этом возвращаются все узлы с указанным именем тэга (в виде индексированного списка). Первый элемент в списке имеет нулевой индекс.
3. Путем перемещения по дереву с использованием отношений между узлами.

Для определения длины списка узлов используется свойство length. Например:

```
x = document.getElementsByTagName("p");
for (i = 0; i < x.length; i++)
 {
  document.write(x[i].innerHTML);
  document.write("<br/>");
 }
```

В данном примере внутрь HTML документа вставляется в виде списка текстовое содержимое всех элементов соответствующих тэгу <p>.

Для навигации по дереву в ближайших окрестностях текущего узла можно использовать следующие свойства:

- parentNode ;
- firstChild ;
- lastChild.

Ниже приводится пример, в котором пользователь имеет возможность выбрать цвет текста с помощью диалогового элемента

```
<html>
<body>
// здесь будет отображаться текст
<div id="c" style="color:blue">Вы выбрали цвет текста: синий</div>

<script language="JavaScript">
// пользователь выбирает цвет текста
  var tcolor = prompt("Выберите цвет текста: red, blue, green, yellow, black","black");
// задается текст
  document.getElementById("c").innerHTML = "Вы выбрали цвет текста: " + tcolor;
// задается цвет текста
  document.getElementById("c").style.color = tcolor;
```

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

Методы, применимые к table:

- createCaption() - создать элемент caption
- deleteCaption() - удалить элемент caption
- createTHead() - создать элемент thead
- deleteTHead() - удалить элемент thead
- createTFoot() - создать элемент tfoot
- deleteTFoot() - удалить элемент tfoot

Методы, применимые к table, thead, tbody и tfoot:

- insertRow(index) - добавить строку в указанное место
- deleteRow(index) - удалить строку с указанным номером
- moveRow(fromIndex, toIndex) - переместить строку

Методы, применимые к tr:

- insertCell(index) - добавить ячейку в указанное место
- deleteCell(index) - удалить ячейку с указанным номером

Методы, имена которых начинаются на create и insert возвращают ссылку на созданный объект. Методы insertRow(index) и insertCell(index) позволяют в качестве номера передать число -1, и в этом случае элемент будет вставлен в конец. Все эти методы можно вызвать в коде сценария на Javascript.

Также полезно знать о некоторых свойствах, предоставляемых объектной моделью таблицы:

Свойства объекта table:

- tHead
- tBodies[]
- tFoot

rows[] - также применимо к thead, tbody и tfoot.

Свойство объекта tr:

- cells[]

Практическая часть

Введение

В данной работе будет создана страница формирования клиентом заказа из набора продуктов. При выборе клиентом продукта, указании количества единиц и нажатии кнопки "Добавить в корзину" к таблице заказа добавляется строка.

Кнопка "Удалить" позволяет удалять пункты из заказа по одному или группой. При всех манипуляциях подсчитывается число единиц заказанных товаров и их суммарная стоимость.

Часть 1. Создание разметки

Статическая часть страницы будет иметь следующий вид (рис. 1).

Рис. 1. Начальный вид формы заказа при открытии страницы

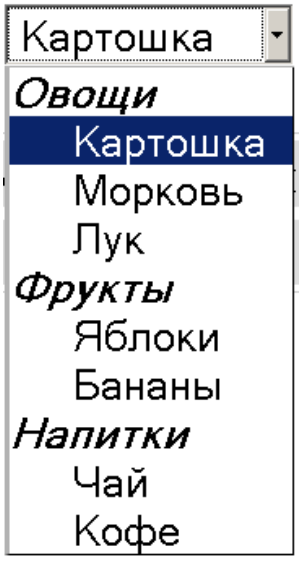
Рассмотрим действия по созданию этой формы пошагово.

1. Создание выпадающего списка (элемент select).

В теле нового документа с именем Lab5.htm создайте элемент div, в котором будет размещена вся разметка данного примера.

Первый элемент - select - должен содержать несколько групп (optgroup) по несколько пунктов (option). Каждый пункт содержит наименование какого-либо продукта, а его атрибут value - цену (произвольное в данном случае число).

В развёрнутом виде список должен выглядеть так:



Назначьте списку атрибут id="lstProducts".

2. Создание остальной разметки

Следом за выпадающим списком создайте два элемента ввода - input type="text" и input type="button" (примеры приведены в лекции). Текстовому элементу ввода назначьте атрибут id="txtQty", а кнопке - onclick="AddToCart()".

Последний элемент разметки - таблица с заголовком (элемент caption), верхним и нижним колонтитулом (элементы thead и tfoot) и телом (элемент tbody). Описания и примеры применения этих элементов также см. в соответствующей лекции. Таблице назначьте атрибут id="tblOrder".

Раздел tbody оставьте пустым (он будет формироваться динамически), а в колонтитулах, как видно на рисунке, следует расположить по одному элементу управления - checkbox и button. Им сразу задайте обработчики события click следующим образом:

```
<input type="checkbox" onclick="ToggleCheck(this)" />
<input type="button" value="Удалить отмеченные" onclick="RemoveSelected()"
/>
```

Часть 2. Создание сценария, манипулирующего таблицей

1. Добавление пунктов заказа

Создайте скрипт в конце тела документа, и сразу определите обработчики, назначенные элементам управления в предыдущем задании - пока пусть это будут функции с пустым телом.

В начало скрипта поместите следующие команды:

```
var tbl = document.getElementById('tblOrder');
var oList = document.getElementById('lstProducts');
```

Ссылки на элементы 'tblOrder' (таблица заказа) и lstProducts (список выбора продукта) будут часто использоваться в коде сценария, поэтому целесообразно определить их единожды.

При нажатии кнопки "Добавить в корзину" в таблицу должны добавляться строки, соответствующие пунктам заказа:

Функция-обработчик этой кнопки была названа AddToCart. Изучите её код и добавьте его в скрипт.

```
/* Добавление пунктов заказа */
function AddToCart() {
  /* Определяем значение, введённое в текстовое поле */
  var qty = document.getElementById('txtQty').value;
  /* Проверка: распознаётся ли значение как число? Если нет, считаем
единицей */
  if (parseFloat(qty) != qty)
    qty = 1;
  /* Вставляем строку в тело таблицы */
  var oRow = tbl.tBodies[0].insertRow(-1);
  /* В добавленную строку вставляем, во-первых, checkbox */
  oRow.insertCell(-1).innerHTML = '<input type="checkbox">';
  /* во-вторых, текст, взятый из списка выбора продуктов */
  oRow.insertCell(-1).innerHTML = oList.options[oList.selectedIndex].text;
  /* в-третьих, цена выбранного продукта */
  oRow.insertCell(-1).innerHTML = oList.value;
  /* далее, количество, указанное в текстовом поле */
  ...
  /* затем стоимость пункта заказа */
  ...
  /* и, наконец, кнопку "Удалить" */
  ...
  /* По окончании вставки строки необходимо пересчитать сумму заказа */
  //Calculate();
}
```

Код в тех строках, где оставлено многоточие, напишите самостоятельно, опираясь на комментарии. Вызов функции Calculate() оставьте пока закомментированным - эту функцию мы добавим позже, а сначала следует добиться верной работы AddToCart().

2. Функция Calculate.

Создайте функцию Calculate() и уберите комментарий с её вызова в функции AddToCart.

```
function Calculate() {
  /* Счётчики для количества единиц товара и общей стоимости */
  var qty = 0, amount = 0;
  /* Цикл по всем строкам в теле таблицы */
  for (var i = 0, n = tbl.tBodies[0].rows.length; i < n; i++) {
    /* Увеличиваем qty на значение в 3 столбце текущей строки */
    qty += parseFloat(tbl.tBodies[0].rows[i].cells[3].innerHTML);
    /* Увеличиваем amount на значение в 4 столбце текущей строки */
```

```
      amount += parseFloat(tbl.tBodies[0].rows[i].cells[4].innerHTML);
    }
    /* Записываем qty в 3 столбец нижнего колонтитула */
    ...
    /* Записываем amount в 4 столбец нижнего колонтитула */
    ...
  }
```

Код в тех строках, где оставлено многоточие, напишите самостоятельно, опираясь на комментарии.

3. Функция RemoveProduct(elem).

Кнопка "Удалить", динамически вставляемая в правую ячейку каждой строки заказа, должна выполнять свою работу - следовательно, необходимо назначить ей обработчик (назовём его RemoveProduct) и написать его код. Этот обработчик имеет существенное отличие от предыдущих, которое заключается в том, что смысл его работы зависит от контекста вызова: удалять нужно именно ту строку, в которой кнопка расположена. Таким образом, обработчик должен принимать параметр, определяющий контекст. Это можно сделать различными путями, и здесь мы предлагаем наиболее типичный. Если назначить кнопке обработчик следующим образом: onclick="RemoveProduct(this)", то функция RemoveProduct получит в качестве параметра ссылку на тот объект, который принял событие (в данном случае - щелчок мыши). Очевидно, обладая ссылкой на элемент в таблице, можно каким-то образом определить номер (index) строки таблицы, в которой он находится, и применить метод таблицы deleteRow(index). Отношение между строкой и кнопкой "Удалить" такое: строка является контейнером ячейки, которая является контейнером кнопки. Ссылку на контейнер элемента можно получить при помощи свойства parentNode этого элемента. С учётом этих соображений код рассматриваемой функции примет следующий вид:

```
  function RemoveProduct(elem) {
    tbl.deleteRow(elem.parentNode.parentNode.rowIndex);
    Calculate();
  }
```

Вставьте эту функцию в скрипт (и не забудьте правильно описать её вызов в динамическом определении кнопки - см. функцию AddToCart). Сохраните документ, обновите страницу в браузере и проверьте добавленную функциональность.

4. Функция RemoveSelected.

Код этой функции представлен ниже, и на нём следует остановиться подробнее, поскольку работа по удалению элементов из множества имеет свои тонкости.

```
  function RemoveSelected() {
    /* находим все элементы input в теле таблицы */
    var checks = tbl.tBodies[0].getElementsByTagName('input');
    var i = 0;
    /* начинаем перебор элементов в цикле */
    while (i < checks.length) {
      /* рассматриваем элемент лишь в том случае, если это checkbox и он
отмечен */
      if (checks[i].type == 'checkbox' && checks[i].checked)
        /* вызываем функцию, которая удалит строку с пунктом заказа -
передаём ей ссылку на checkbox */
        RemoveProduct(checks[i]);
      else
        /* счётчик увеличиваем лишь в том случае, если удаление не было
сделано */
        i++;
    }
  }
```

Множество ссылок на все элементы ввода (input) внутри тела таблицы (tbody) легко получить при помощи метода getElementsByTagName. Этот метод принимает в качестве параметра имя тэга и выдаёт массив ссылок. Важно отметить, что ссылками являются не

только элементы массива, но и сам массив - *ссылочный*, т.е. все изменения в документе автоматически отражаются в этом массиве. Таким образом, удалив строку из таблицы (и тем самым удалив из неё два элемента input - checkbox и button), мы тем самым уменьшим число элементов в массиве checks на 2. Это следует принимать во внимание при циклической обработке массива.

Номер рассматриваемого в цикле элемента массива обозначен в данном примере переменной i, и отсчёт, как обычно, начинается с нуля. Чтобы правильно сформулировать оператор увеличения этого i, следует задаться вопросом: "Рассмотрев элемент с номером i, элемент с каким номером следует рассматривать далее?" Ответ таков: "Если удаление не было выполнено, то i+1 ; иначе - вновь i (теперь это номер следующего *из оставшихся* элементов)". Поэтому цикл записан именно таким образом (а не с использованием оператора for).

6. Функция ToggleCheck.

Последняя функция данного сценария создаёт дополнительное удобство: можно отметить или сбросить сразу все галочки в пунктах заказа, щёлкнув по галочке (элементу checkbox) в заголовке таблицы. Этому элементу назначена функция-обработчик ToggleCheck, принимающая в качестве параметра this, т.е. ссылку на объект-источник события. Напишите эту функцию самостоятельно - все необходимые для этого приёмы уже рассмотрены.

Окончательный вид работающей страницы показан на рис.2.

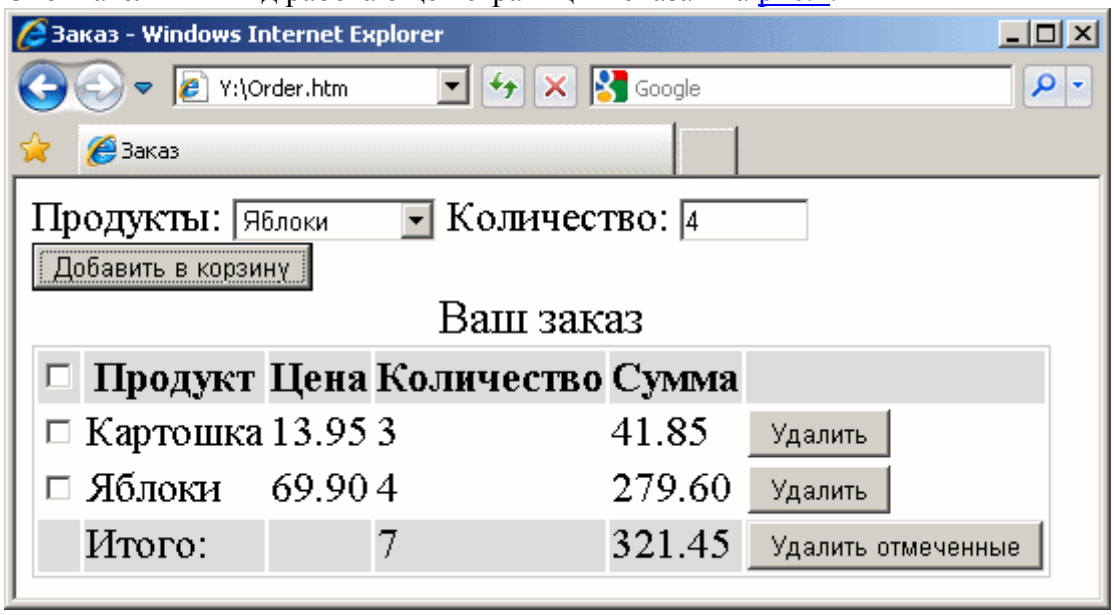


Рис. 2. Действующая страница заказа

Задание на самостоятельную работу

1) Сделайте так, чтобы при добавлении продукта, который в заказе уже присутствует, новая строка не добавлялась, а вместо этого увеличивалось бы количество уже заказанных единиц этого продукта.

2) Добавьте в таблицу заказа поля «Цена в у.е.» и «Сумма в условных единицах». Курс для пересчета валют требуется отдельно вводить в текстовом окне.

3) Добавьте в таблицу заказа поля «НДС» и «Сумма с НДС». НДС вычисляется по ставке 15%.

4) Добавить список регионов со значением километража до пункта доставки. В итоговой строке должно быть поле «Стоимость доставки» и «Общая стоимость».

5) Добавить на страницу список «Скидка». В итоговой строке должно быть поле «Общая стоимость со скидкой».

6) При заказе товаров на сумму более 1000р. становится активной кнопка «Добавить бонус» при нажатии на которую в заказ добавляется стока с бонусным товаром.

Варианты выполнения самостоятельного задания:

№варианта	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
№заданий	1,2	1,3	1,4	1,5	1,6	2,6	5,6	2,3	2,4	2,5	3,4	3,5	3,6	4,5	4,4

4. СПИСОК РЕКОМЕНДОВАННОЙ ЛИТЕРАТУРЫ И ЭЛЕКТРОННЫХ РЕСУРСОВ

Основная

1. А.А. Дуванов. Web-конструирование. HTML. — СПб.: БХВ-Петербург, 2003. — 325 с.
2. А.А. Дуванов. Web-конструирование. DHTML. — СПб.: БХВ-Петербург, 2003. — 512 с.
3. Молли Э. Хольцшлаг. Использование HTML 4: Пер. с англ.: Уч. пос. — М: Издательский дом «Вильямс», 2000. — 1008 с.
4. П.Б. Храмцов, С.А.Брик, А.М. Русак, А.И.Сурин, Основы WEB-технологий, М.: ИТУИТ.РУ, 2003. – 512 с.
5. Д.Роджерс, Программирование на Microsoft JScript.NET, “Вильямс”, 2002. - 352 с.
6. В.В. Дунаев, JavaScript, СПб. : “Питер”, 2003 .— 394 с.
7. К.Мельтцер, Разработка CGI-приложений на Perl, М. : “Вильямс”, 2001 .— 395 с.
8. И.Шапошников, PHP 5.1 : учебный курс, СПб [и др.] : “Питер”, 2007 .— 192 с.
9. Д.Шеперд, Освой самостоятельно XML за 21 день, М.: “Вильямс”, 2002. – 432 с.
10. А.Старыгин, XML: разработка Web-приложений, СПб. : “БХВ-Петербург”, 2003 .— 585 с.
11. Э.Троелсен, C# и платфома .NET. Библиотека программиста, СПб.: “Питер”, 2007.- 796 с.
12. М.Беллиньясо, Разработка Web-приложений в среде ASP.NET 2.0: с примерами на C#, “Диалектика”, 2007. - 640 с.
13. С.Шорт, Разработка XML Web-сервисов средствами Microsoft .NET, СПб. : “БХВ-Петербург”, 2003 .— 480 с.
14. Э.Ньюкомер, Веб-сервисы : XML, WSDL, SOAP и UDDL, СПб. : Питер, 2003.
15. И.Салмре, Программирование мобильных устройств на платформе .NET Compact Framework, “Вильямс”, 2006. - 736 с.
16. Д.Крейн, Э.Паскарелло, Д. Джеймс, AJAX в действии: технология - Asynchronous JavaScript and XML = AJAX in Action, М.: “Вильямс”, 2006. — С. 640.

Дополнительная

1. А. Матросов, А. Сергеев, М. Чаунин. HTML 4.0. Наиболее полное руководство.
2. М. Браун, Д. Ханикат. HTML 3.2 в подлиннике.
3. В.А. Остейковский. Информатика. — М.: ВШ, 2000. — 319 с.
4. В. Холмогоров. Основы Web–мастерства. Учебный курс. — СПб: Питер, 2001. — 352 с.
5. Использование HTML 4: Пер. с англ. / Луиза Паттерсон, Сью Шарльворс, Джоди Корнелиус и др.: Уч. пос. — М.: Издательский дом «Вильямс», 2000. — 400 с.
6. С.Н. Коржинский. Настольная книга Web–мастера: эффективное применение HTML, CSS и JavaScript. М.: Издательский дом «КноРус», 2000. — 320 с.

7. Б. Форта, Освой самостоятельно регулярные выражения. 10 минут на урок, М.: “Вильямс”, 2005. – 184 с.
8. http://www.help.mymoney.ru (материалы по первоначальным шагам в создании и продвижении сайта).
9. http://www.botik.ru/~robot/sale/web.htm (Роботландский университет).
10. http://www.webclub.ru (Российский клуб веб-дизайнеров. Множество материалов по веб-конструированию).
11. http://www.artlebedev.ru/kovodstvo/ — Артемий Лебедев. Руководство по дизайну сайта.
12. http://ru.html.net — учебники HTML, CSS
13. http://html.manual.ru/ — справочник
14. http://wcode.ru/ — учебники

Приложение 1. Перечень основных свойств CSS в соответствии со спецификацией CSS2.1

Свойства	Значение
Фон	
background	Сокращенный вариант записи для свойств background-color, background-image, background-repeat, background-attachment и background-position.
background-attachment	Устанавливает, должна ли фоновая картинка скролиться или должна быть зафиксирована в окне браузера (scroll/fixed).
background-color	Устанавливает цвет фона для элемента.
background-image	Устанавливает фоновую картинку для элемента(`url('...файл...')`).
background-position	Устанавливает первоначальное положение для фоновой картинки(процент% процент%)
background-repeat	Управляет циклическим повторением фоновой картинки(repeat/repeat-x/repeat-y)
Рамка (граница, бордюр)	
border	Краткий вариант записи для свойств border-width, border-style и border-color. Влияет на все четыре границы элемента.
border-color	Устанавливает цвет рамки со всех сторон элемента.
border-width	Устанавливает толщину рамки со всех сторон элемента.
border-style	Определяет стиль оформления рамки вокруг элемента.
border-collapse	Указывает ячейкам таблицы, иметь ли собственный бордюр или общий с соседней ячейкой(collapse/separate)
border-spacing	Устанавливает расстояние между ячейками таблицы.
Шрифт	
font	Краткий вариант записи свойств font-style, font-variant, font-weight, font-size, line-height и font-family.
font-family	Определяет шрифт(ы) для отображения текста.
font-size	Управляет размером шрифта.
font-style	Управляет наклоном шрифта (normal/italic).
font-variant	Управляет внешним видом строчных букв (строчные как прописные, "капитель").
font-weight	Управляет толщиной (насыщенностью) шрифта(normal/bold).
Позиционирование	
position	Определяет порядок, в соответствии с которым элемент отображается на веб-странице(absolute/relative/fixed).
bottom	Сдвигает элемент относительно нижнего края. Используется совместно со свойством position.
left	Сдвигает элемент относительно левого края. Используется совместно со свойством position.
top	Сдвигает элемент относительно верхнего края. Используется совместно со свойством position.
right	Сдвигает элемент относительно правого края. Используется совместно со свойством position.
z-index	Определяет порядок, в соответствии с которым элементы накладываются друг на друга, если необходимо отобразить их на одном месте(z-idex:2;).
Форматирование	
clear	Запрещает/разрешает элементу обтекать "floated" объекты(left/right/both/none).
clip	Определяет область элемента, которая должна отображаться на странице (rect (Y1,X1,Y2,X2))
display	Изменяет базовые свойства элементов (Значение none позволяет скрыть объект).
float	Сдвигает элемент к правому или левому краю (left/right).
height	Определяет высоту прямоугольной области вокруг элемента.
overflow	Определят как отображать блочный элемент в случае, если его содержимое выходит за рамки родительского элемента(visible/scroll/hidden).
visibility	Управляет настройкой видимости элемента(none/hidden/collapse).
width	Определяет ширину прямоугольной области вокруг элемента.

Списки	
list-style	позволяет одновременно задать три параметра для маркеров пунктов списка: list-style-type, list-style-position и/или list-style-image.
list-style-image	Устанавливает изображение, которое будет использоваться для маркировки пунктов списка, например, ul {list-style-image: url('/images/rhomb.gif')}
list-style-position	Определяет, как отобразить на странице маркер пункта в списке: внутри того же прямоугольника, в котором располагается элемент списка или вне его.
list-style-type	Определяет, какой вид будет иметь маркер пункта в списке..
Текст	
direction	Применяется при создании сайтов на языках, в которых чтение страницы идет справа налево.
letter-spacing	Определяет длину интервала между буквами.
page-break-inside	Определяет размер межстрочного интервала.
text-align	Выравнивает содержимое блочного элемента (текст или изображение) относительно границ блока, а так же содержимое ячеек таблицы по горизонтали.
text-decoration	Определяет, какой оформительский прием нужно применить к тексту(underline/overline/line-through/none).
text-indent	Определяет размер отступа первой строки в тексте.
text-transform	Управляет написанием прописных или строчных букв в тексте(uppercase/lowercase/capitalize/none).
vertical-align	Определяет высоту содержимого внутри инлайн элемента или внутри ячейки таблицы.
white-space	Определяет как отображать пробелы, символы табуляции и пустой строки (pre).
word-spacing	Определяет расстояние между словами.
Поля	
padding	Сокращенный способ задать следующие параметры: padding-top, padding-right, padding-bottom и/или padding-left.
Прочее	
caption-side	Определяет, где будет отображаться заголовок таблицы: над ней или под ней (top/bottom).
color	Устанавливает цвет текста элемента.
cursor	Определяет вид курсора при наведении мышки на некий элемент.
empty-cells	Определяет, нужно ли отображать границы и фон ячейки, если в ней нет содержимого (show/hide).
margin	Сокращенный способ задать следующие параметры: margin-top, margin-right, margin-bottom и/или margin-left
margin-bottom	Определяет ширину внешнего пространства между нижним бордюром и невидимой границей прямоугольника.
margin-left	Определяет ширину внешнего пространства между левым бордюром и невидимой границей прямоугольника.
margin-right	Определяет ширину внешнего пространства между правым бордюром и невидимой границей прямоугольника.
margin-top	Определяет ширину внешнего пространства между верхним бордюром и невидимой границей прямоугольника.
max-height	Определяет максимальную высоту элемента.
max-width	Определяет максимальную ширину элемента.
min-height	Определяет минимальную высоту элемента.
min-width	Определяет минимальную ширину элемента.
outline	Это быстрый способ задать следующие параметры: outline-width, outline-style и/или outline-color. Свойство аналогичное border.
outline-color	Определяет цвет контура вокруг элемента.
outline-style	Определяет вид контура вокруг элемента.
outline-width	Определяет ширину контура вокруг элемента.
quotes	Определяет вид открывающей и закрывающей кавычки в тексте.
table-layout	Определяет ширину столбцов в таблице.

Приложение 2. Примеры текстового контента

1. Автомобиль

Первые известные чертёжи автомобиля (с пружинным приводом) принадлежат Леонардо да Винчи (стр. 812R Codex Atlanticus), однако ни действующего экземпляра, ни сведений о его существовании до наших дней не дошло. В 2004 году эксперты Музея истории науки из Флоренции смогли восстановить по чертежам этот автомобиль, доказав тем самым правильность идеи Леонардо. В эпоху Возрождения и позже в ряде европейских стран «самодвижущиеся» тележки и экипажи с пружинным двигателем строились в единичных количествах для участия в маскарадах и парадах.

В 1769 году французский изобретатель Кюньо испытал первый образец машины с паровым двигателем[2], известный как «малая телега Кюньо», а в 1770 году — «большую телегу Кюньо». Сам изобретатель назвал её «Огненная телега» — она предназначалась для буксировки артиллерийских орудий.

«Тележку Кюньо» считают предшественницей не только автомобиля, но и паровоза, поскольку она приводилась в движение силой пара. В XIX веке дилижансы на паровой тяге и рутьеры (паровые тягачи, то есть безрельсовые паровозы) для обычных дорог строились в Англии, Франции и применялись в ряде европейских стран, включая Россию, однако они были тяжёлыми, прожорливыми и неудобными, поэтому широкого распространения не получили.

В 1791 году русским изобретателем Иваном Кулибиным была изготовлена «самокатная повозка».

Были отдельные случаи построения легковых автомобилей как предметов роскоши. Так, в историю вошёл La Marquise.

2. Polo Sedan

Новый Volkswagen Polo Sedan появился в продаже летом 2010 г. Автомобиль спроектирован специально для российского рынка на базе бюджетного бразильского VW Pointer (Gol) Sedan. Предшественником Фольксваген Поло седан был Volkswagen Pointer, однако появление этой модели не нашло широкого отклика у российских автолюбителей.

По словам создателей, при проектировании Polo Sedan учитывались климатические и дорожные условия нашей страны — у машины полностью оцинкованный кузов, усиленная подвеска и дорожный просвет 170 мм. Автомобиль оснащается только бензиновым двигателем объемом 1,6 л мощностью 105 л.с. Седан может оснащаться как 5-ступенчатой механической коробкой передач, так и 6-ступенчатой автоматической трансмиссией с функцией последовательного ручного переключения (Tiptronic). Volkswagen Polo sedan производится только на заводе ООО «Фольксваген Груп Рус» в промзоне Грабцево под Калугой.

На российском рынке модель приобрела широкую популярность, благодаря стильной внешности, комфорту, просторному салону и лояльным ценам.

3. Ford Focus

Ford Focus воплощает динамичный и инновационный стиль. Его стремительный дизайн дополняется выразительным оформлением салона, выдержанным в стиле кокпита спортивного автомобиля. Также Новый Focus может оснащаться целым рядом технологий для обеспечения комфорта, безопасности и развлечения водителя и пассажиров.

На сегодняшний день Новый Focus является одним из самых передовых автомобилей эконом-класса. Он оснащен рядом ультрасовременных систем и функций, которые делают каждую Вашу поездку безопаснее и комфортнее, эффективнее и экономичнее, легче и приятнее.

4. Ford Mondeo

Стильный автомобиль бизнес-класса, динамичный и спортивный Mondeo демонстрирует свой совершенный облик с любой точки зрения. И снаружи, и внутри. Это – подлинный шедевр технической мысли, произведенный с безукоризненной аккуратностью и воплощенный выразительным языком «кинетического дизайна» Ford.

Надежные и экономичные двигатели со сниженным уровнем выбросов CO2 служат идеальным дополнением к изысканному дизайну и интеллектуальным технологиям представительского Ford Mondeo, обеспечивая превосходные впечатления от управления автомобилем.

В современном, уже четвертом, поколении автомобиля еще ярче чувствуется бизнес-класс. Он стал крупнее своих предшественников, отличается повышенной шумо- и виброизоляцией, более совершенными системами безопасности. Последняя версия Ford Mondeo, представленная в 2010 году выделяется новыми линиями капота, решеткой радиатора измененной формы, стильными светодиодными огнями – и еще более динамичным обликом. Всю силу Ford Mondeo можно почувствовать, даже просто рассматривая его фотографии.

Какой бы Mondeo Вы ни выбрали, и Вы, и Ваши пассажиры будете чувствовать себя в полной безопасности. Ощущение защищенности создают отличное сцепление с дорогой, контролируемость движения, использованные в отделке салона материалы премиум-класса, спокойная атмосфера консервативного клуба и даже благородный звук, с которым закрываются двери комфортного салона.

5. Лада Калина

В июле 2007 года началось производство Лады Калины с новым 16-клапанным двигателем объемом 1,4 литра, а в сентябре того же года АвтоВАЗ приступил к выпуску модели с АБС.

В августе 2007 года выпущен первый автомобиль Lada Kalina универсал. Тогда же из-за дефекта в рулевой колонке пришлось отозвать 6200 автомобилей, произведённых в декабре 2005 — январе 2006 (они были отремонтированы дилером АвтоВАЗа компанией «Брянск-Лада»).

Из-за дефекта литья картера рулевого механизма АвтоВАЗ отозвал 171 автомобиль модели Lada Kalina, которые были выпущены 24 — 25 мая 2006 года. Кроме того, АвтоВАЗ отзывал 8400 автомобилей Lada Kalina из-за дефектов опор двигателя.