МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»
ВОЛЖСКИЙ ФИЛИАЛ

Кафедра гуманитарных и естественнонаучных дисциплин

Методические указания к лабораторным работам
по дисциплине

ИНТЕРНЕТ ПРОГРАММИРОВАНИЕ

Направление подготовки

09.03.01 Информатика и вычислительная техника

Направленность (профиль, специализация) образовательной программы

«Автоматизированные системы обработки информации и управления»

Квалификация

бакалавр

Чебоксары
2019

ОГЛАВЛЕНИЕ

Введение

Для написания кода HTML будем использовать программу Notepad++.

Notepad++ - текстовый редактор, предназначенный для программистов и всех тех, кого не устраивает скромная функциональность входящего в состав Windows Блокнота.

Основные особенности Notepad++:

- Подсветка текста и возможность сворачивания блоков, согласно синтаксису языка программирования.
- Поддержка большого количества языков (C, C++, Java, XML, HTML, PHP, Java Script, ASCII, VB/VBS, SQL, CSS, Pascal, Perl, Python, Lua, TCL, Assembler).
- WYSIWYG (печатаешь и получаешь то, что видишь на экране).
- Настраиваемый пользователем режим подсветки синтаксиса.
- Авто-завершение набираемого слова.
- Одновременная работа с множеством документов.
- Одновременный просмотр нескольких документов.
- Поддержка регулярных выражений Поиска/Замены.
- Полная поддержка перетягивания фрагментов текста.
- Динамическое изменение окон просмотра.
- Автоматическое определение состояния файла.
- Увеличение и уменьшение.
- Заметки.
- Выделение скобок при редактировании текста.
- Запись макроса и его выполнение.

HTML (Hyper Text Markup Language) – язык разметки гипертекста, включает в себя способы оформления гипертекстовых документов, стандартный язык разметки документов во Всемирной паутине. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Гипертекст – структура, позволяющая устанавливать смысловые связки между элементами текста и другими документами.

Теги

Тег – начальный или конечный маркер элемента записывается в угловых скобках и состоит из имени, за которым может следовать список атрибутов (все атрибуты располагаются в начальном теге). Большинство тегов имеют два компонента: открывающий и закрывающий. Закрывающий компонент имеет то же название, но при записи перед названием ставится символ « / ».

Назначение HTML – тегов:

- форматирования текста;
- описания кадров и форм;
- форматирования таблиц и списков;
- организации ссылок на другие ресурсы;
- вставки изображений и расширений HTML.

1. Знакомство с языком HTML в текстовом редакторе Notepad++

***Цели и задачи данной лабораторной работы*:**

- Ознакомиться с текстовым редактором Notepad ++.
- Получить базовые представления о языке HTML.
- Создать простейший документ на языке HTML с применением тегов для текста, заголовков и картинок.

Базовые представления о языке HTML

HTML-документ — это один большой контейнер, который начинается с тега <HTML> и заканчивается тегом </HTML>:

```
<HTML>Содержание документа</HTML>
```

Контейнер HTML или гипертекстовый документ состоит из двух других вложенных контейнеров: *заголовка* документа (HEAD) и тела документа (BODY). Рассмотрим простейший пример классического документа.

```
<HTML>
<HEAD>
<TITLE>Простейший документ</TITLE>
</HEAD>
<BODY TEXT=#0000ff BGCOLOR=#f0f0f0>
<H1>Пример простого документа</H1>
<HR>
Формы HTML-документов
<UL>
<LI>Классическая
<LI>Фреймовая
</UL>
<HR>
</BODY>
</HTML>
```

Компания Netscape Communication расширила классическую форму документа возможностью организации фреймов (кадров), позволяющих разделить рабочее окно программы просмотра на несколько независимых фреймов. В каждый фрейм можно загрузить свою страницу HTML. Приведем пример документа с фреймами.

```
<HTML>
<HEAD>
<TITLE>Документ с фреймами</TITLE>
</HEAD>
<FRAMESET COLS="30%,*">
<FRAME SRC=frame1.htm NAME=LEFT>
<FRAME SRC=frame2.htm NAME=RIGHT>
</FRAMESET>
</HTML>
```

Назначение заголовка

Заголовок HTML-документа является необязательным элементом разметки. В HTML 2.0 предлагалось вообще отказаться от элементов *HEAD* и BODY. В то время в HTML не было элементов, которые использовались одновременно и в *заголовке*, и в теле документа. Современная практика HTML-разметки такова, что почти в каждом документе есть HTML-*заголовок*.

Первоначально существование *заголовка* определялось необходимостью именования окна браузера. Это достигалось за счет элемента разметки *TITLE*:

```
<HTML>
 <HEAD>
  <TITLE>Это заголовок</TITLE>
 ...
 </HEAD>
 <BODY>
 ...
 </BODY>
</HTML>
```

Отображение содержания элемента TITLE

Однако задумывался *заголовок* для несколько иных целей. Исходя из общих соображений, связанных с теорией и практикой разработки и эксплуатации гипертекстовых систем, все гипертекстовые связи информационных узлов принято разделять на контекстные и общие.

Контекстные гипертекстовые связи соответствуют определенному месту документа — контексту. В HTML такие связи реализованы в виде гипертекстовых ссылок (элемент A (anchor)). Фактически до реализации таблиц описателей стилей в современных браузерах это был единственный вид связей, которыми мог управлять автор HTML-документа.

Общие гипертекстовые связи определяются не частью документа (контекстом), а всем документом целиком. Например, быть предыдущим по отношению к другому документу или следующим — это общая гипертекстовая связь, которая позволяет организовать так называемый "линейный" просмотр информационных узлов гипертекстовой сети.

Реализация такого сорта ссылок уже давно является частью проектов W3C (Arena, Amaya). В коммерческих браузерах такой механизм реализован только для описателей стилей (элемент разметки ***LINK***).

Важную роль *заголовок* HTML-документа играет в JavaScript. Существует принципиальная разница между *заголовком* и телом документа при использовании элемента разметки *SCRIPT*. Она заключается в определении зоны видимости функций и переменных. Переменные и функции, определенные в *заголовке* документа, относятся ко всему окну браузера. Это значит, что к ним можно обратиться из любого места документа и изменить их значения. Кроме того, к ним можно обратиться из другого окна или фрейма. Фактически, это глобальные переменные. При работе с многослойными документами переменные и функции тела относятся к слоям, что делает доступ к ним неудобным.

Основные контейнеры заголовка

Основные контейнеры заголовка — это элементы HTML-разметки, которые наиболее часто встречаются в *заголовке* HTML-документа, т.е. внутри элемента разметки *HEAD*.

Мы рассмотрим только восемь элементов разметки, включая сам элемент разметки *HEAD*:

- *HEAD* (элемент разметки *HEAD*);
- *TITLE* (заглавие документа);
- *BASE* (база URL);
- **ISINDEX** (поисковый шаблон);
- *META* (метаинформация);
- *LINK* (общие ссылки);
- ***STYLE*** (описатели стилей);
- ***SCRIPT*** (скрипты).

Чаще всего применяются элементы *TITLE, SCRIPT, STYLE*. Использование элемента *META* говорит об осведомленности автора о правилах индексирования документов в поисковых системах и возможности управления HTTP-обменом данными. *BASE* и **ISINDEX** в последнее время практически не применяются. *LINK* указывают только при использовании внешних относительно данного документа описателей стилей.

Элемент разметки HEAD

Элемент разметки ***HEAD*** содержит *заголовок* HTML-документа. Данный элемент разметки не является обязательным. При наличии тега начала элемента разметки желательно использовать и тег конца элемента разметки. По умолчанию элемент *HEAD* закрывается, если встречается либо тег начала контейнера BODY, либо тег начала контейнера FRAMESET. Атрибутов у тега начала контейнера нет, хотя в DTD HTML один необязательный атрибут прописан. Синтаксис контейнера *HEAD* в общем виде выглядит следующим образом:

```
<HEAD profile="http://www.intuit.ru/help">
Это пример из документации по сайту Интернет-
Университета Информационных Технологий
</HEAD>
```

Контейнер *заголовка* служит для размещения информации, относящейся ко всему документу в целом. Необязательный атрибут PROFILE указывает на файл профиля, с описанием META-тегов которые не описаны в стандартной спецификации. В качестве значения этого атрибута указывается URL данного файла.

Элемент разметки TITLE

Элемент разметки ***TITLE*** служит для именования документа в World Wide Web. Более прозаическое его назначение — именование окна браузера, в котором просматривается документ. Состоит контейнер из тега начала, содержания и тега конца. Наличие тега конца обязательно. Тег начала элемента не имеет специфических атрибутов.

В различных браузерах алгоритм отображения элемента *TITLE* может отличаться. Так, в некоторых руководствах предлагается создать бегущую строку в *заголовке* документа, указав несколько последовательных контейнеров *TITLE*:

```
<TITLE>И</TITLE>
<TITLE>Ин</TITLE>
<TITLE>Инт</TITLE>
<TITLE>Инте</TITLE>
<TITLE>Интер</TITLE>
...
<TITLE>Интернет-Университе</TITLE>
<TITLE>Интернет-Университет</TITLE>
```

Такой механизм в современных браузерах не работает. При этом следует учитывать, что в отличие от реализации "бегущей" строки средствами JavaScript, лидирующие пробелы в *заголовке* игнорируются.

При выборе текста для содержания контейнера *TITLE* следует учитывать, что отображается он системным шрифтом, так как является *заголовком* окна браузера. В нелокализованных версиях операционных систем и графических оболочек русский текст содержания элемента *TITLE* будет отображаться абракадаброй.

Синтаксис контейнера *TITLE* в общем виде выглядит следующим образом:

```
<TITLE>название документа</TITLE>
```

Заголовок не является обязательным контейнером документа. Его можно опустить. Роботы многих поисковых систем используют содержание элемента *TITLE* для создания поискового образа документа. Слова из *TITLE* попадают в индекс поисковой системы. Из этих соображений элемент *TITLE* всегда рекомендуется использовать на страницах Web-узла.

Задание 1. «Создать простейший документ на языке HTML с применением тегов для текста, заголовков и картинок»

Открываем программу Notepad++ и создаем новый документ

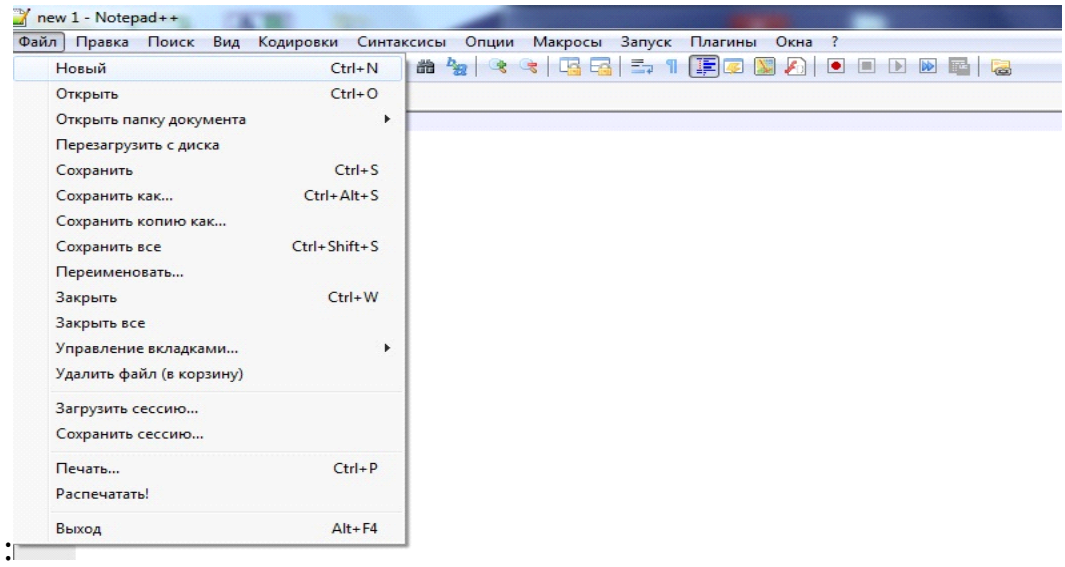

Рис. 1 Создание нового документа

Выбираем кодировку. Это нужно для того, чтобы текст отображался корректно. Под ОС Windows кодировку выбираем Windows-1251.

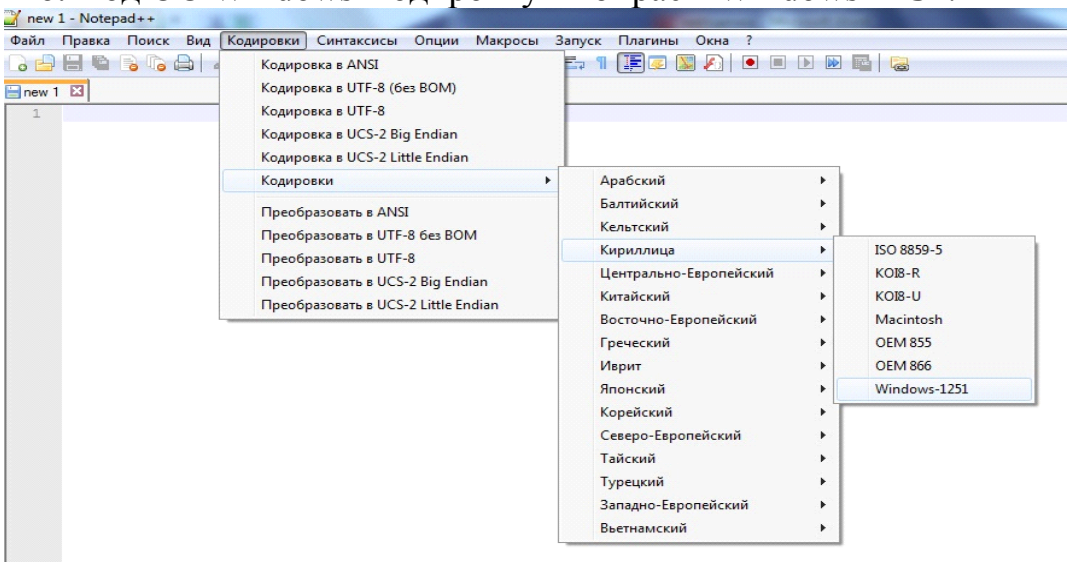

Рис. 2 Выбор кодировки

Пишем каркас html документа. Он состоит из следующих тегов:

<!DOCTYPE html> – предназначен для указания типа текущего документа, необходим, чтобы браузер понимал, как следует интерпретировать текущую веб-страницу, поскольку HTML существует в нескольких версиях

<html> … </html> – начало и конец разметки документа;

<head> … </head> – внутри контейнера находятся метатеги, которые используются для хранения информации предназначенной для браузеров и поисковых систем;

<title> … </title> –содержание заголовка;

<body> … </body> – содержание документа.

<meta charset> – <meta> – информация для браузера, charset – атрибут задает кодировку документа.

Пример:

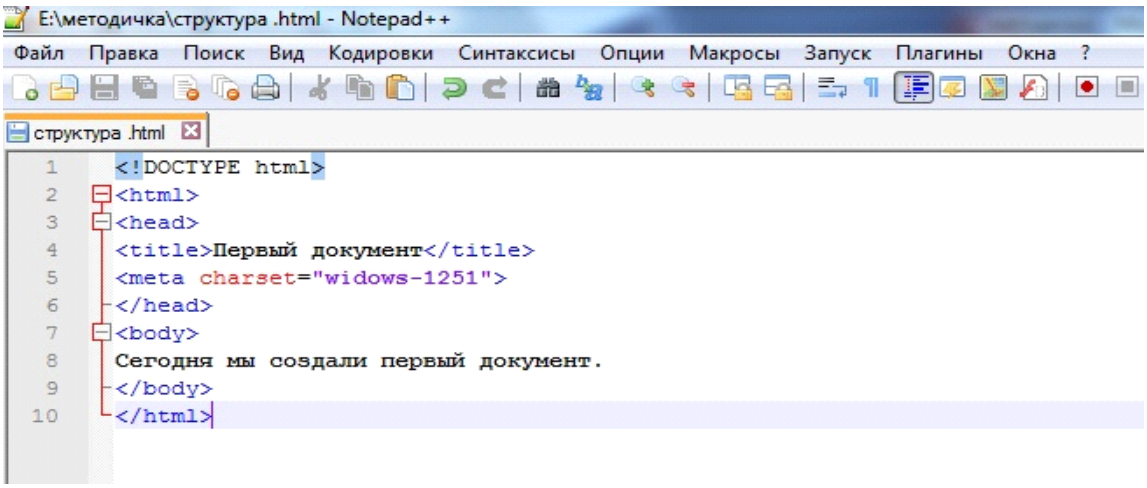

Рис. 3 Структура документа

Элемент горизонтальная линия

Тег <hr> рисует горизонтальную линию, которая по своему виду зависит от используемых параметров, а также браузера. Тег <hr> относится к блочным элементам, линия всегда начинается с новой строки, а после нее все элементы отображаются на следующей строке.

Атрибуты:

align - определяет выравнивание линии.
color - цвет линии.
noshade - рисует линию без трехмерных эффектов.
size - толщина линии.
width - ширина линии.

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Ter HR</title>
 </head>
 <body>

 <hr>
  <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh
  euismod tincidunt ut lacreet dolore magna aliguam erat volutpat. Ut wisis enim
  ad minim veniam, quis nostrud exerci tution ullamcorper suscipit lobortis nisl
  ut aliquip ex ea commodo consequat.</p>
 <hr>

 </body>
</html>
```

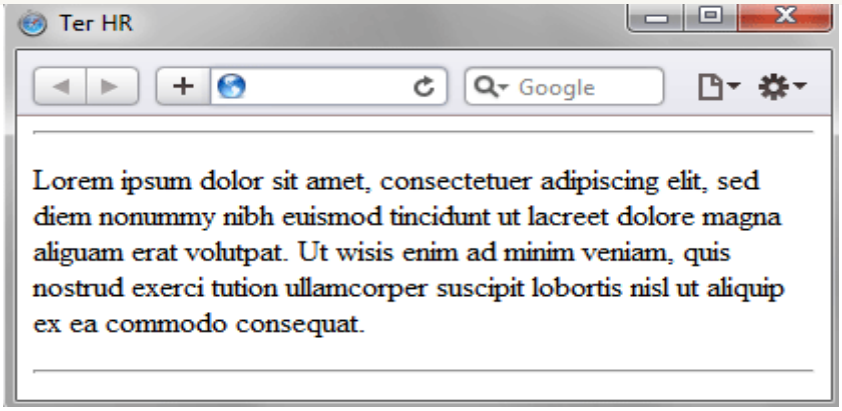

Рис. 4 Применение тега <hr>

Текст

Предварительно отформатированный текст можно вставить в html-страницу с помощью тегов <pre>…</pre>. Браузер покажет текст, заключенный между этими тегами в том виде, в каком он был в текстовом редакторе.

Вывести на экран первую букву своего имени при помощи тега <pre>.

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Ter PRE</title>
 </head>
 <body>
  <pre>
      -                    -----
     -----               |-----
    ||----|              ----||
    ||-----            -----||
    ||-----|          |-----||
    || -----           ------||
    ||- ----|        |------||
    ||---||--        -------||
    ||--|| --|     |-------||
    || -|| |--     --- - --||
    || -||  --|-|--| - ---|
    |---||   |-----|  |-----|
    |---||    |----   |-----|
    |----|     ---    |-----|
    |-----            ------|
  </pre>
 </body>
</html>
```

Рис. 5 Использование тега <pre>

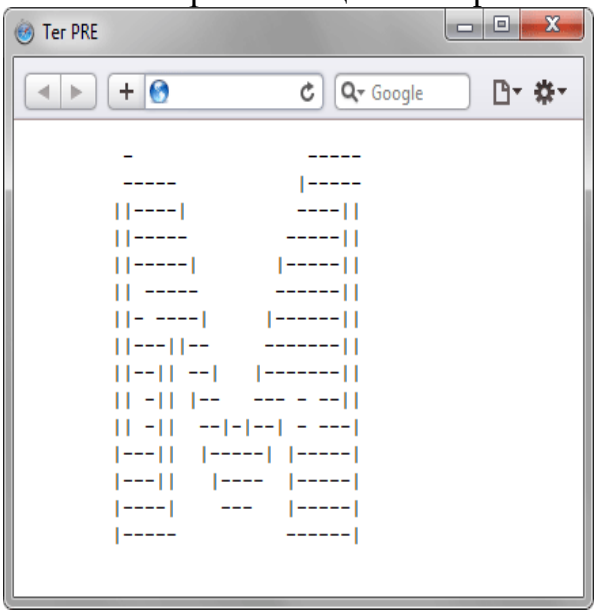

Рис. 6 Тег <pre>

Самыми популярными тегами для форматирования текста являются:
< b>< /b> — используется для выделения текста жирным шрифтом.

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="Windows-1251">
  <title>Тег B</title>
 </head>
 <body>
  <p><b>Lorem ipsum dolor sit amet</b></p>
  <p><b>Lorem ipsum</b> dolor sit amet, consectetuer adipiscing elit,
sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat  volutpat.
<b>Ut wisis</b> enim ad minim veniam, quis nostrud exerci
tution ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.</p>
 </body>
</html>
```

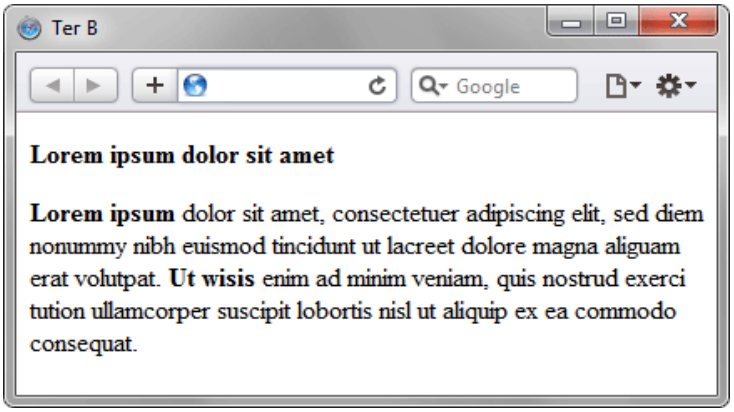

Рис. 7 Применение тега <b>

< strong>< /strong> - также может применяться для акцентирования текста. Устанавливает жирное выделение текста.

< I>< /I> — используется для получения курсивного текста.

```
<!DOCTYPE HTML>
<html>
 <head>
   <meta charset="Windows-1251">
  <title>Тег I</title>
 </head>
 <body>
  <p><b><i>Lorem ipsum dolor sit amet</i></b></p>
  <p><i>Lorem</i> ipsum dolor sit amet, consectetuer adipiscing elit,
  sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat
  volutpat. <i>Ut</i> wisis enim ad minim veniam, quis nostrud exerci
  tution ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.</p>
 </body>
</html>
```

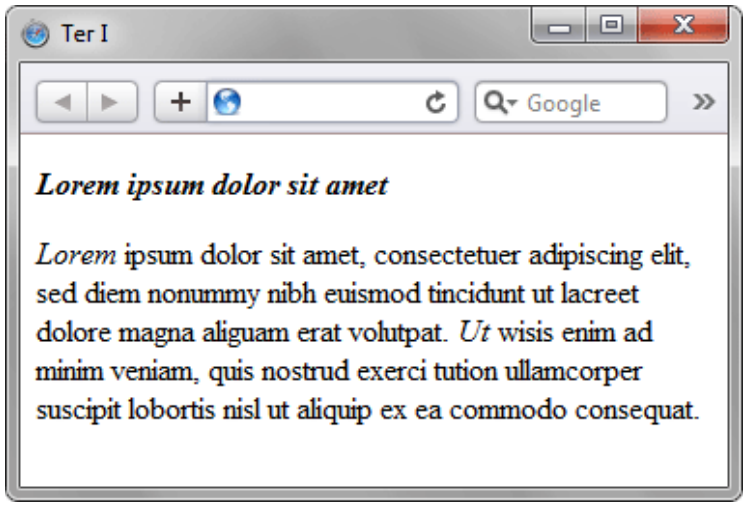

Рис. 8 Курсив , тег <i>

< U>< /U> — позволит подчеркнуть текст. Тут главное, чтобы пользователь не перепутал подчёркнутый текст с ссылкой.

<STRIKE>< /STRIKE> — делает текст перечёркнутым.

В HTML абзацы создаются с помощью тега < p>.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="Windows-1251">
  <title>Применение абзацев</title>
 </head>
 <body>
   <p>В одних садах цветёт миндаль, в других метёт метель.</p>
   <p>В одних краях ещё февраль, в других - уже апрель.</p>
```

```
  <p>Проходит время, вечный счёт: год за год, век за век...</p>
  <p>Во всём - его неспешный ход, его кромешный бег.</p>
  <p>В году на радость и печаль по двадцать пять недель.</p>
  <p>Мне двадцать пять недель февраль, и двадцать пять - апрель.</p>
  <p>По двадцать пять недель в туман уходит счёт векам.</p>
  <p>Летит мой звонкий балаган куда-то к облакам.</p>
  <p><i>М. Щербаков</i></p>
 </body>
</html>
```

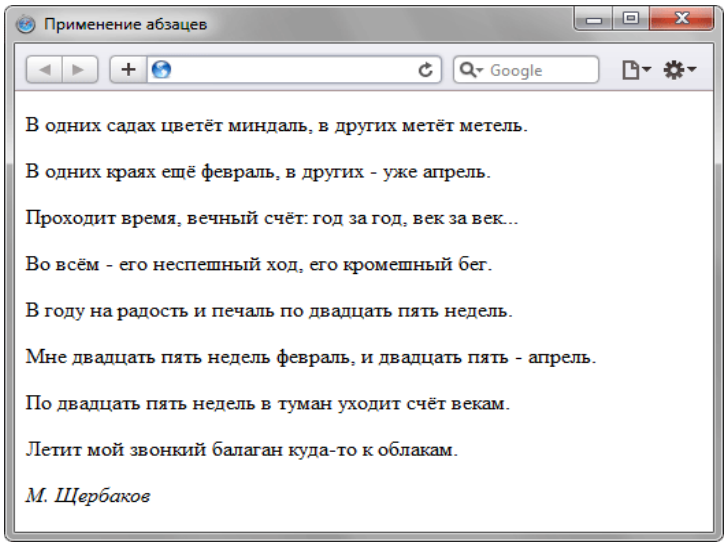

Рис. 9 Абзацы

Заголовки

Заголовки выполняют важную функцию. Во-первых, они показывают важность раздела, к которому относятся. Во-вторых, с помощью различных заголовков легко регулировать размер текста. Чем выше уровень заголовка, тем больше размер шрифта. Самым верхним уровнем является уровень 1 (<h1>), а самым нижним — уровень 6 (<h6>). И, в-третьих, поисковики добавляют рейтинг тексту, если он находится внутри тега заголовка.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="Windows-1251">
  <title>Заголовки в тексте</title>
 </head>
 <body>
  <h1>Заголовок первого уровня</h1>
  <h2>Заголовок второго уровня</h2>
  <h3>Заголовок третьего уровня</h3>
  <h4>Заголовок четвертого уровня</h4>
  <h5>Заголовок пятого уровня</h5>
  <h6>Заголовок шестого уровня</h6>
 </body>
```

```
</html>
```

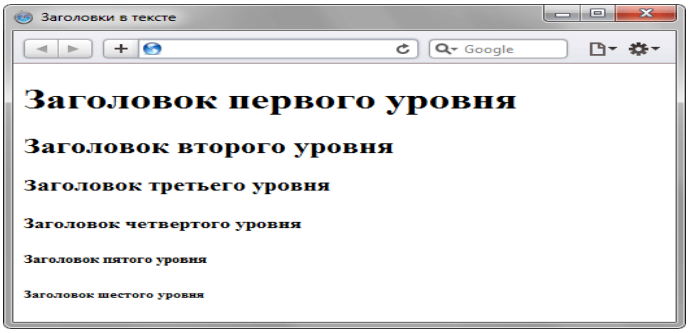

Рис. 10 Заголовки

Ссылки и гиперссылки

Ссылки являются основой гипертекстовых документов и позволяют переходить с одной веб-страницы на другую. Для создания ссылки необходимо сообщить браузеру, что является ссылкой, а также указать адрес документа, на который следует сделать ссылку. Оба действия выполняются с помощью тега <a>. Общий синтаксис создания ссылок следующий.

текст ссылки

Атрибут href определяет URL (Universal Resource Locator, универсальный указатель ресурса), иными словами, адрес документа, на который следует перейти, а содержимое контейнера <a> является ссылкой. Текст, расположенный между тегами <a> и , по умолчанию становится синего цвета и подчеркивается.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="Windows-1251">
  <title>Ссылки на странице</title>
 </head>
 <body>
  <p><a href="dog.html">Собаки</a></p>
  <p><a href="cat.html">Кошки</a></p>
 </body>
</html>
```

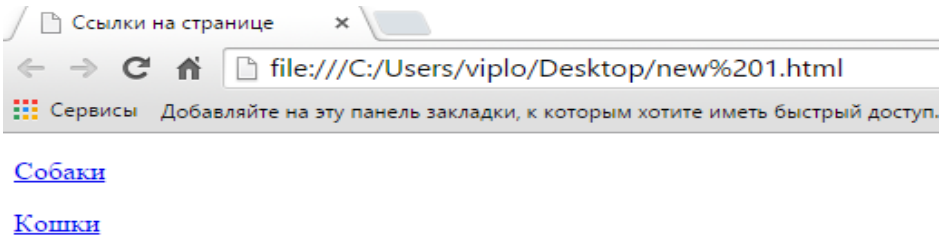

Рис. 11 Гиперссылки

Добавление изображений на страницу

Для добавления изображения в документ применяется тег <img>, его атрибут src определяет путь к графическому файлу, который должен быть в формате GIF, PNG или JPEG. Также для тега <img> необходимо указать

обязательный атрибут alt, он описывает альтернативный текст видимый в процессе загрузки изображения или при отключении картинок в браузере.

<img src="путь к файлу" alt="альтернативный текст">

Примечание: Если графический файл находится в одной папке с HTML-документом, то в качестве пути достаточно указать только имя файла.

<img src="i am.jpg" alt="альтернативный текст" title="Мое фото">

< img src="http://www.seoded.ru/image/seoded.png">

Рис. 12 Тег <img>

Рис. 13 Изображение

2 Контейнеры тела документа

Теги тела документа

Теги тела документа предназначены для управления отображением информации в программе интерфейса пользователя. Они описывают гипертекстовую структуру *базы данных* при помощи встроенных в текст контекстных гипертекстовых ссылок. ***Тело документа*** состоит из:

- иерархических контейнеров и заставок;
- заголовков (от H1 до H6);
- *блоков* (параграфы, списки, формы, таблицы, картинки и т.п.);
- горизонтальных подчеркиваний и адресов;
- текста, разбитого на области действия стилей (подчеркивание, выделение, курсив);
- математических описаний, графики и гипертекстовых ссылок.

Тело документа – контейнер BODY

Описание тегов тела документа следует начать с тега BODY. В отличие от тега HEAD, тег BODY имеет атрибуты.

Атрибут BACKGROUND определяет фон, на котором отображается текст документа. Так, если источником для фона *HTML* - документа является графический файл image.gif, то в открывающем теге тела *BODY* появляется соответствующий атрибут:

<BODY BACKGROUND="image.gif">

Как видно из этого примера, в качестве значения данного атрибута используется адрес в сокращенной форме URL. В данном случае это адрес локального файла. Следует заметить, что разные интерфейсы пользователя поддерживают различные дополнительные атрибуты для тега BODY.

Таблица 2.1. Атрибуты

Атрибут	Значение
BGCOLOR=#FFFFFF	Цвет фона
TEXT=#0000FF	Цвет текста
VLINK=#FF0000	Цвет пройденных гипертекстовых ссылок
LINK=#008000	Цвет гипертекстовой ссылки

В данной таблице строка #XXXXXX определяет цвет в терминах RGB в шестнадцатеричной нотации. Также имеется возможность задавать цвета по названию. Далее в таблице приведены названия цветов, определенные в стандарте *HTML* 4 и соответствующие им RGB-коды. Отметим, что многие современные браузеры выходят за рамки стандартов и поддерживают гораздо больше названий цветов.

Таблица 2.2. Цвета

Название	Код	Название	Код
aqua	#00FFFF	navy	#000080
black	#000000	olive	#808000
blue	#0000FF	purple	#800080
fuchsia	#FF00FF	red	#FF0000
gray	#808080	silver	#C0C0C0
green	#008000	teal	#008080
lime	#00FF00	white	#FFFFFF
maroon	#800000	yellow	#FFFF00

Так, значения атрибутов в таблице 2.1 определяют цвет текста как синий, фона — белый, пройденные ссылки красные, а новые ссылки зеленые. Если в качестве атрибутов тега BODY указать

<BODY BGCOLOR=#FFFFFF TEXT=#0000FF
VLINK=#FF0000 LINK=#00FF00>,

то цвет фона будет белым, текст будет синим, ссылки — зелеными, а пройденные ссылки станут красными. Однако пользоваться этими атрибутами следует крайне осторожно, так как у пользователя может оказаться другой интерфейс, который эти параметры не интерпретирует.

Microsoft Internet Explorer и Netscape Navigator допускают применение атрибутов LEFTMARGIN=n и TOPMARGIN=n в теге <BODY>. Атрибут LEFTMARGIN= задает левое поле для всей страницы. TOPMARGIN= определяет верхнее поле. Число n показывает ширину поля в пикселах. Например, тег <BODY LEFTMARGIN ="40"> создаст на всей странице левое поле шириной 40 пикселов. При n, равном 0, левое поле отсутствует.

Теги управления разметкой

Заголовки

Заголовок обозначает начало раздела документа. В стандарте определено 6 уровней заголовков: от H1 до H6. Текст, окруженный тегами <H1></H1>, получается большим — это основной заголовок. Если текст окружен тегами <H2></H2>, то он выглядит несколько меньше (подзаголовок); текст внутри <H3></H3> еще меньше и так далее до <H6></H6>. Некоторые программы позволяют использовать большее число заголовков, однако реально более трех уровней встречается редко, а более 5 — крайне редко.

Ниже на рисунке показан результат использования следующих заголовков: (открыть)

<H1>Заголовок 1</H1>
<H2>Заголовок 2</H2>

Тег <P>

Тег <P> применяется для разделения текста на параграфы. В нем используются те же атрибуты, что и в заголовках.

Атрибут ALIGN

Атрибут ALIGN позволяет выровнять текст по левому или правому краю, по центру или ширине. По умолчанию текст выравнивается по левому краю. Данный атрибут применим также к графике и таблицам.

Далее приведены возможные значения атрибута ALIGN:

ALIGN=justify выравнивание по левому и правому краям. Реализовано не во всех программах интерпретации.

ALIGN=left выравнивание по левому краю. По умолчанию текст *HTML* выравнивается по левому краю и не выравнивается по правому, то есть начало строк находится на одном уровне по вертикали, а концы — на разных. Чаще всего, получающийся при этом текст с равными промежутками между словами выглядит лучше. Поскольку выравнивание по левому краю задается автоматически, атрибут ALIGN=left можно опустить.

ALIGN=right выравнивание по правому краю. Текст, выровненный по правому краю и не выровненный по левому — концы строк находятся на одном уровне, а начало на разных, — часто используется с целью создать оригинальный дизайн. Для этого задается атрибут ALIGN=right в обычных тегах, например в теге <P>.

ALIGN=center центрирование текста и графики. Есть несколько способов отцентрировать текст или графику. В спецификациях *HTML*3.0 предлагается пользоваться тегом <ALIGN="center">. Однако этот тег применим не ко всем объектам *HTML* -страницы, поэтому разработчики Netscape добавили тег <CENTER>, который центрирует любые объекты и поддерживается браузерами Netscape Navigator 3.0, Microsoft Internet Explorer 3.0 и другими. К тегу <CENTER> нужно относиться с осторожностью. Какой-нибудь браузер может его вообще проигнорировать, и на странице окажется текст, выровненный по левому краю.

Обтекание графики текстом. С помощью атрибута ALIGN можно заставить текст "обтекать" графический объект. Для этого следует поместить тег <IMG SRC="/путь/файл.gif"> туда, где должен быть графический объект, и добавить атрибут ALIGN=left, ALIGN=right или ALIGN=center. Кроме того, с помощью атрибутов HSPACE и VSPACE (они описаны ниже) задается ширина горизонтальных и *вертикальных полей*, отделяющих изображение от текста. Можно также создать рамку вокруг картинки или обрамление таблицы текстом. Чтобы текст не "обтекал" графику, а прерывался, необходимо применить тег
 с атрибутом CLEAR.

***Использование тега
***

Принудительный перевод строки используется для того, чтобы нарушить стандартный порядок отображения текста. При обычном режиме интерпретации программа интерфейса пользователя отображает текст в рабочем окне, автоматически разбивая его на строки. В этом режиме концы строк текста игнорируются. Иногда для большей выразительности требуется

начать печать с новой строки. Для этого и нужен тег BR. Атрибут CLEAR в теге
 используется для того, чтобы остановить в указанной точке обтекание объекта текстом и затем продолжить текст в пустой области за объектом. Продолжающийся за объектом текст выравнивается в соответствии со значениями LEFT, RIGHT или ALL атрибута CLEAR:

<BR CLEAR=left> Текст будет продолжен, начиная
с ближайшего пустого левого поля.
<BR CLEAR=right> Текст будет продолжен, начиная
с ближайшего пустого правого поля.
<BR CLEAR=all> Текст будет продолжен, как
только и левое, и правое поля окажутся пустыми.

Элемент разметки <NOBR>

Тег <NOBR> (No Break, без обрыва) дает браузеру команду отображать весь текст в одной строке, не обрывая ее. Если текст, заключенный в <NOBR>, не поместится на экране, браузер добавит в нижней части окна документа горизонтальную полосу прокрутки. Если вы хотите оборвать строку в определенном месте, поставьте там тег
.

Теги управления отображением символов

Все эти теги можно разбить на два класса: теги, управляющие формой отображения (font style), и теги, характеризующие тип информации (information type). Часто внешне разные теги при отображении дают одинаковый результат. Это зависит главным образом от настроек интерпретирующей программы и вкусов пользователя.

Теги, управляющие формой отображения

Курсив, усиление, подчеркивание, верхний индекс, нижний индекс, шрифт большой, маленький, красный, синий, различные комбинации — все это делает страницы более интересными. Microsoft Internet Explorer и Netscape Navigator позволяют определить шрифт с помощью тега FONT. Теперь можно объединять на одной странице несколько видов шрифтов, вне зависимости от того, какой из них задан по умолчанию в браузере пользователя.

Теги <BIG> и <SMALL> — изменение размеров шрифта

Текст, расположенный между тегами <BIG></BIG> или <SMALL> </SMALL>, будет, соответственно, больше или меньше стандартного.

Верхние и нижние индексы

С помощью тегов <SUP> и <SUB> можно задавать верхние и нижние индексы, необходимые для записи торговых знаков, символов копирайта, ссылок и сносок. Рассматриваемые теги позволяют создать внутри текстовой области верхние или нижние индексы любого размера. Чтобы они казались меньше окружающего текста, можно использовать теги <SUP> и <SUB> с атрибутом FONT SIZE=-1, уменьшающим размер шрифта.

Атрибут SIZE

Атрибут SIZE тега <FONT> позволяет задавать размер текста в данной области. Если вы не пользуетесь тегом <BASEFONT SIZE=n>для задания определенного размера шрифта на всей странице, то по умолчанию

принимается 3. Некоторые браузеры тег <FONT> не поддерживают, поэтому желательно употреблять его только внутри текстовой области. В других случаях лучше использовать теги <H1>, <H2>, <H3> и т.д. Главное преимущество тега <FONT> состоит в том, что после окончания действия он не разбивает строку, как теги <Hn>. Поэтому тег <FONT> бывает очень полезен для изменения размера шрифта в середине строки.

Атрибут COLOR

Если вы хотите сделать свою страницу более красочной, можете воспользоваться атрибутом COLOR в теге FONT, и тогда единственным ограничением будет цветовая палитра на компьютере пользователя.

Теги, управляющие формой отображения, приведены в таблице.

Таблица 2.3. Теги, управляющие формой отображения

Тег	**Значение**
<I>...</I>	Курсив (Italic)
<B>...</B>	Усиление (Bold)
<TT>...</TT>	Телетайп
<U>...</U>	Подчеркивание
<S>...</S>	Перечеркнутый текст
<BIG>...</BIG>	Увеличенный размер шрифта
<SMALL>...</SMALL>	Уменьшенный размер шрифта
_{...}	Подстрочные символы
^{...}	Надстрочные символы

Таблица 2.4. Теги, характеризующие тип информации

Тег	**Значение**
<EM>...</EM>	Типографское усиление
<CITE>...</CITE>	Цитирование
<STRONG>...</STRONG>	Усиление
<CODE>...</CODE>	Отображает примеры кода (например, "коды программ")
<SAMP>...</SAMP>	Последовательность литералов
<KBD>...</KBD>	Пример ввода символов с клавиатуры
<VAR>...</VAR>	Переменная
<DFN>...</DFN>	Определение
<Q>...</Q>	Текст, заключенный в двойные кавычки

Эти теги допускают вложенность, поэтому все они имеют тег начала и конца. При использовании таких тегов следует помнить, что их отображение

зависит от настроек программы-интерфейса пользователя, которые могут и не совпадать с настройками программы-разработчика гипертекста.

Создание списков в HTML

Списки являются важным средством структурирования текста и применяются во всех языках разметки. В HTML имеются следующие виды списков: ненумерованный список (неупорядоченный) (*Unordered Lists* <UL>), нумерованный список (упорядоченный) (*Ordered Lists* <OL>) и список определений. Теги для ненумерованных и нумерованных списков — это основа *HTML*. *HTML* 3.2 добавляет несколько атрибутов к тегам списков для выбора разных типов маркеров в ненумерованных списках и разных схем нумерации в нумерованных. Можно включать такие атрибуты и в сами теги элементов списка (*List Item* <LI>), чтобы сменить тип маркера в середине списка. После появления нового атрибута все последующие маркеры в списке будут иметь такой же вид.

Неупорядоченные списки — тег <UL>

Ненумерованный список. Ненумерованный список предназначен для создания текста типа:

- первый элемент списка;
- второй элемент списка;
- третий элемент списка.

Записывается данный список в виде последовательности:

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

Теги <UL> и </UL> — это теги начала и конца ненумерованного списка, тег <LI> (*List Item*) задает тег элемента списка. Помимо этих тегов, существует тег, позволяющий именовать списки — <LH> (List Header).

Атрибуты маркеров в ненумерованном списке

Чтобы не применять одни и те же маркеры на разных уровнях вложенности, можно использовать атрибут TYPE. Вы можете задать любой тип маркера в произвольном месте списка. Можно даже смешивать разные типы маркеров в одном списке. Ниже перечислены теги с атрибутами стандартных маркеров:

```
<UL TYPE=DISC>Тег создает сплошные маркеры
такого типа, как в списках первого уровня по
умолчанию.
<UL TYPE=CIRCLE>Тег создает маркеры в виде
окружностей.
<UL TYPE=SQUARE>Тег создает сплошные квадратные
маркеры.
```

Упорядоченные списки — тег <OL>

Нумерованные списки. Тег <OL> вместе с атрибутом TYPE= в *HTML* 3.2 позволяет создавать нумерованные списки, используя в качестве номеров не только обычные числа, но и строчные и прописные буквы, а также строчные и прописные римские цифры. При необходимости можно даже смешивать эти типы нумерации в одном списке:

<OL TYPE=l> Тег создает список с нумерацией
в формате 1., 2., 3., 4. и т.д.
<OL TYPE=A> Тег создает список с нумерацией
в формате A., B., C., D. и т.д.
<OL TYPE=a> Тег создает список с нумерацией
в формате a., b., c., d. и т.д.
<OL TYPE=I> Тег создает список с нумерацией
в формате I., II., III., IV. и т.д.

Список определений — тег <DL>

Теги списка (Definition List: <DL>, <DT>, <DD>) используют для создания списка терминов и их определений. Схема использования тега следующая.

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

Определяемый термин записывается на одной строке, а его определение — на следующей, с небольшим отступом вправо. Тег <DL>позволяет создавать отдельные абзацы с отступом без нумерации или маркеров. Отступ делается от левого края. Если на странице несколько тегов <DL>, то текст постепенно сдвигается все больше вправо. В конце определения поместите закрывающий тег </DL>. Помните, что тег <DL> сдвигает только левую границу абзаца.

Задание 2 Работа со списками и таблицами в текстовом редакторе Notepad++

Маркированный список определяется тем, что перед каждым элементом списка добавляется небольшой маркер, обычно в виде закрашенного кружка. Сам список формируется с помощью контейнера <ul>, а каждый пункт списка начинается с тега <li>, как показано ниже.

```
<ul>
 <li>Первый пункт</li>
 <li>Второй пункт</li>
 <li>Третий пункт</li>
</ul>
```

В списке непременно должен присутствовать закрывающий тег , иначе возникнет ошибка. Закрывающий тег хотя и не обязателен, но ставится, чтобы четко разделять элементы списка.

```
<!DOCTYPE HTML >
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Маркированный список</title>
```

```
 </head>
<body>
<hr>
 <ul>
<li>Чебурашка</li>
<li>Крокодил Гена</li>
 <li>Шапокляк</li>
<li>Крыса Лариса</li>
</ul>
 <hr>
 </body>
</html>
```

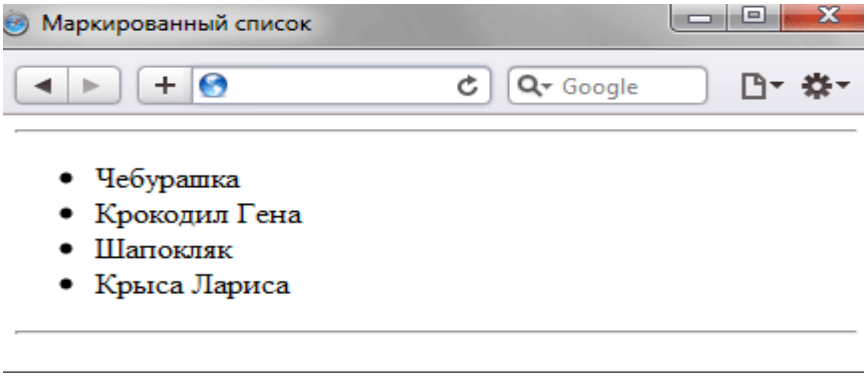

Рис. 14 Маркированный список

Нумерованные списки представляют собой набор элементов с их порядковыми номерами. Вид и тип нумерации зависит от атрибутов тега **<ol>**, который и применяется для создания списка. Каждый пункт нумерованного списка обозначается тегом **<li>**, как показано ниже.

```
<p><strong>Работа со временем</strong></p>
<ol>
  <li>создание пунктуальности (никогда не будете никуда опаздывать);</li>
  <li>излечение от пунктуальности (никогда никуда не будете торопиться);</li>
  <li>изменение восприятия времени и часов.</li>
</ol>
```

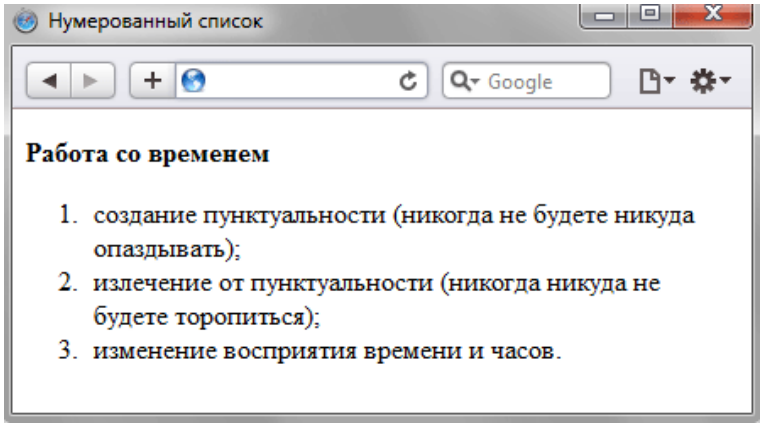

Рис. 15 Пример нумерованного списка

Таблицы

Элемент <table> служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются с помощью тегов <tr> и <td>. Внутри <table> допустимо использовать следующие элементы: <caption>, <col>, <colgroup>, <tbody>, <td>, <tfoot>, <th>, <thead> и <tr>.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML  4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Тег TABLE</title>
 </head>
 <body>
  <table border="1" width="100%" cellpadding="5">
   <tr>
    <th>Ячейка 1</th>
    <th>Ячейка 2</th>
   </tr>
   <tr>
    <td>Ячейка 3</td>
    <td>Ячейка 4</td>
  </tr>
 </table>
 </body>
</html>
```

Рис. 16 таблицы

Блоки

Элемент является блочным элементом и предназначен для выделения фрагмента документа с целью изменения вида содержимого. Как правило, вид блока управляется с помощью стилей. Чтобы не описывать каждый раз стиль внутри тега, можно выделить стиль во внешнюю таблицу стилей, а для тега добавить атрибут class или id с именем селектора.

```
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>Тег DIV, атрибут align</title>
  <style type="text/css">
   #layer1 {
    background: #fc0;
    padding: 5px;
   }
   #layer2 {
    background: #fff;
    width: 60%;
    padding: 10px;
   }
  </style>
 </head>
 <body>

  <div align="right" id="layer1">
  <div align="left" id="layer2">
   Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
   sed diem nonummy nibh euismod tincidunt ut lacreet dolore
   magna aliguam erat volutpat.
  </div>
  </div>

 </body>
</html>
```

Рис. 17 Блочное форматирование

3. Стили CSS

Интернет значительно упрощает *поиск* нужной информации, обратной стороной, правда, является то, что большинство *Интернет* - источников попросту пересказывают друг друга, если не дублируют полностью. К примеру, набрав в поисковике *CSS*, можно очень быстро узнать, что это:

1. каскадные таблицы стилей;
2. используется для управления внешним видом информации, предоставляемой сайтом.

Несколько сложнее узнать, как именно соотносится *CSS* с *HTML*, и еще сложнее – почему для оформления *HTML* документа нельзя обойтись собственными возможностями *HTML*, тем более, что соответствующие атрибуты присутствуют в большинстве *тегов*. В качестве введения к данной лекции, мы постараемся внести некоторую *ясность* в эти вопросы.

Возникновение CSS

Во второй лекции нашего курса приведена краткая история возникновения *HTML*. Частично написанное сейчас будет пересекаться с прошлым материалом.

Первый стандарт *HTML* не содержал структур, позволяющих отображать текст каким-либо особенным образом, то есть не было никаких средств управления внешним видом информации. Первоначальной задачей было обеспечение доступности информации для любых устройств.

С дальнейшим распространением *HTML* такие компании, как *Microsoft* и *Netscape* стали внедрять свои собственные *теги* или "улучшать" имеющиеся именно с целью управления оформлением *html* - документов. Часть внесенных изменений прижилась и "ушла" в массы.

После случилось логически неизбежное, а именно, *HTML* стал представлять собой набор несовместимых между собой *тегов* и расширений. Результатом исправления ситуации стал стандарт *HTML* 3.2, "узаконивший" ряд изменений и устранивший проблемы совместимости.

Таким образом, если несколько упростить, то можно сказать, что все приемы и *теги*, направленные на управление внешним видом предоставляемой информации, по своей сути, являются атавизмом предыдущих версий *HTML*.

Стандарт *HTML* 3.2 лишь исправил ряд наиболее серьезных недостатков предыдущих версий. Настоящим же решением проблемы явился стандарт *HTML* 4.0, в рамках которого было предложено отделить описание структуры *html* документа от его оформления.

Если уж совсем "рыться" в истории, то можно заметить, что и этот подход не нов. SGML, на котором и основывалась первая версия HTML, в числе прочего предполагал наличие отдельного "файла стилей" документа.

Таким образом, возникновение *каскадных таблиц стилей* является закономерным результатом эволюционного развития стандартов *HTML*.

Суть и преимущества CSS

CSS (*Cascading Style Sheets* – каскадные таблицы стилей) – язык описания внешнего вида документа, созданного при помощи *языка разметки*.

Концепция *каскадных таблиц стилей* была предложена Хоконом Виум Ли – норвежским ученым и специалистом в области *информационных технологий*, работавшим в то время на консорциум *W3C*.

Как правило, *CSS* применяется при работе с *HTML* и *XHTML* языками, и используется для задания цветов, параметров шрифтов, расположения блоков и иных элементов представления веб - страниц.

К преимуществам использования *CSS* относятся:

- *централизованное управление отображением множества* документов при помощи одной таблицы стилей;
- упрощенный контроль внешнего вида веб - страниц;
- наличие разработанных дизайнерских техник;
- возможность использования различных стилей для одного документа, в зависимости от устройства, при помощи которого осуществляется доступ к веб - странице.

Стандарт *HTML* 4.0 помимо *CSS* утвердил и объектную модель *браузера* (*Browser Object Model – BOM*), о которой следует сказать отдельно.

Объектная модель браузера описывает содержимое веб - документа, т.е. сама модель является набором объектов, описывающих указанное содержимое. Поскольку *BOM* уникальна для каждого *браузера*, возникали проблемы с межплатформенными приложениями. В конечном итоге, на *место объектной модели браузера* пришла *объектная модель документа* (*Document Object Model – DOM*), описывающая стандарт представления веб - страниц в виде набора объектов.

Применяются с помощью элемента link, который должен располагаться внутри элемента head и нигде более.

Пример подключения файла css в html документе:

<link rel=”stylesheet” type=”text/css” href=”mystyle.css” media=”all” />

Создаем документ с расширением **«*.css»** по аналогии с документом **«html»**

Единицы измерения:

«px, pt, em, %» и после каждого свойства ставится ***«;».***

Основные свойства

Список базовых свойств, которые должен знать даже начинающий веб-мастер:
margin, padding, border, background-color, color, font-family, font-size, float

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.

background-color - Устанавливает цвет фона для элемента.

background-image - Устанавливает фоновую картинку для элемента.

background-position - Устанавливает первоначальное положение для фоновой картинки.

background-repeat - Управляет циклическим повторением фоновой картинки.

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.

border-spacing - Устанавливает расстояние между ячейками таблицы.

Верхняя рамка

border-top - Краткий вариант доступа к свойствам border-top-width, border-top-style и border-top-color.

border-top-color - Устанавливает цвет верхнего бордюра.

border-top-style - Устанавливает стиль линии верхнего бордюра.

border-top-width - Устанавливает ширину верхнего бордюра.

Нижняя рамка

border-bottom - Краткий вариант доступа к свойствам border-bottom-width, border-bottom-style и border-bottom-color.

border-bottom-color - Устанавливает цвет нижнего бордюра.

border-bottom-style - Устанавливает стиль линии нижнего бордюра.

border-bottom-width - Устанавливает ширину нижнего бордюра.

Левая рамка

border-left - Краткий вариант доступа к свойствам border-left-width, border-left-style и border-left-color.

border-left-color - Устанавливает цвет левого бордюра.

border-left-style - Устанавливает стиль линии левого бордюра.

border-left-width - Устанавливает ширину левого бордюра.

Правая рамка

border-right - Краткий вариант доступа к свойствам volume, border-right-style и border-right-color.

border-right-color - Устанавливает цвет правого бордюра.

border-right-style - Устанавливает стиль линии правого бордюра.

volume - Устанавливает ширину правого бордюра.

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 (курсив).

font-variant - Управляет внешним видом строчных букв (строчные как прописные, "капитель").

font-weight - Управляет толщиной (насыщенностью) шрифта.

Позиционирование

position - Определяет порядок, в соответствии с которым элемент отображается на веб-странице.

bottom - Сдвигает элемент относительно нижнего края. Используется совместно со свойством position.

left - Сдвигает элемент относительно левого края. Используется совместно со свойством position.

page-break-before - 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.

Форматирование

clear - Запрещает/разрешает элементу обтекать "floated" объекты.

clip - Определяет область элемента, которая должна отображаться на странице.

display - Изменяет базовые свойства элементов.

float - Сдвигает элемент к правому или левому краю.

height - Определяет высоту прямоугольной области вокруг элемента.

overflow - Определят как отображать блочный элемент в случае, если его содержимое выходит за рамки родительского элемента.

visibility - Управляет настройкой видимости элемента.

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.

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.

text-indent - Определяет размер отступа первой строки в тексте.

text-transform - Управляет написанием прописных или строчных букв в тексте.

vertical-align - Определяет высоту содержимого внутри инлайн элемента или внутри ячейки таблицы.

white-space - Определяет как отображать пробелы, символы табуляции и пустой строки.

word-spacing - Определяет расстояние между словами.

Печать

widows - Позволяет избежать появления висячей строки.

orphans - Позволяет избежать появления одинокой первой строки.

page-break-after - Определяет наличие или отсутствие разрыва страницы после элемента при печати.

page-break-before - Определяет наличие или отсутствие разрыва страницы перед элементом при печати.

page-break-inside - Определяет наличие или отсутствие разрыва страницы внутри элемента при печати.

Поля

padding - Сокращенный способ задать следующие параметры: padding-top, padding-right, padding-bottom и/или padding-left.

padding-bottom - Определяет ширину пространства между содержимым элемента и нижним бордюром.

padding-left - Определяет ширину пространства между содержимым элемента и левым бордюром.

padding-right - Определяет ширину внешнего пространства между правым бордюром и невидимой границей прямоугольника.

padding-top - Определяет ширину внешнего пространства между верхним бордюром и невидимой границей прямоугольника.

Прочее

caption-side - Определяет, где будет отображаться заголовок таблицы: над ней или под ней.

color - Устанавливает цвет текста элемента.

content - Применяется для того, чтобы вставить текст или изображение до или после какого-либо элемента.

counter-increment - Задает значения приращений счетчика.

counter-reset - Устанавливает идентификатор, который хранит счетчик отображений какого-либо элемента и устанавливает начальное значение этого счетчика.

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.

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.

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.

Кому не понятно читать там: http://ru.html.net/tutorials/css/, http://htmlbook.ru/

Задание 3 «Разработка одностраничной web-визитки»

Написать одностраничный сайт о себе, пример ниже:

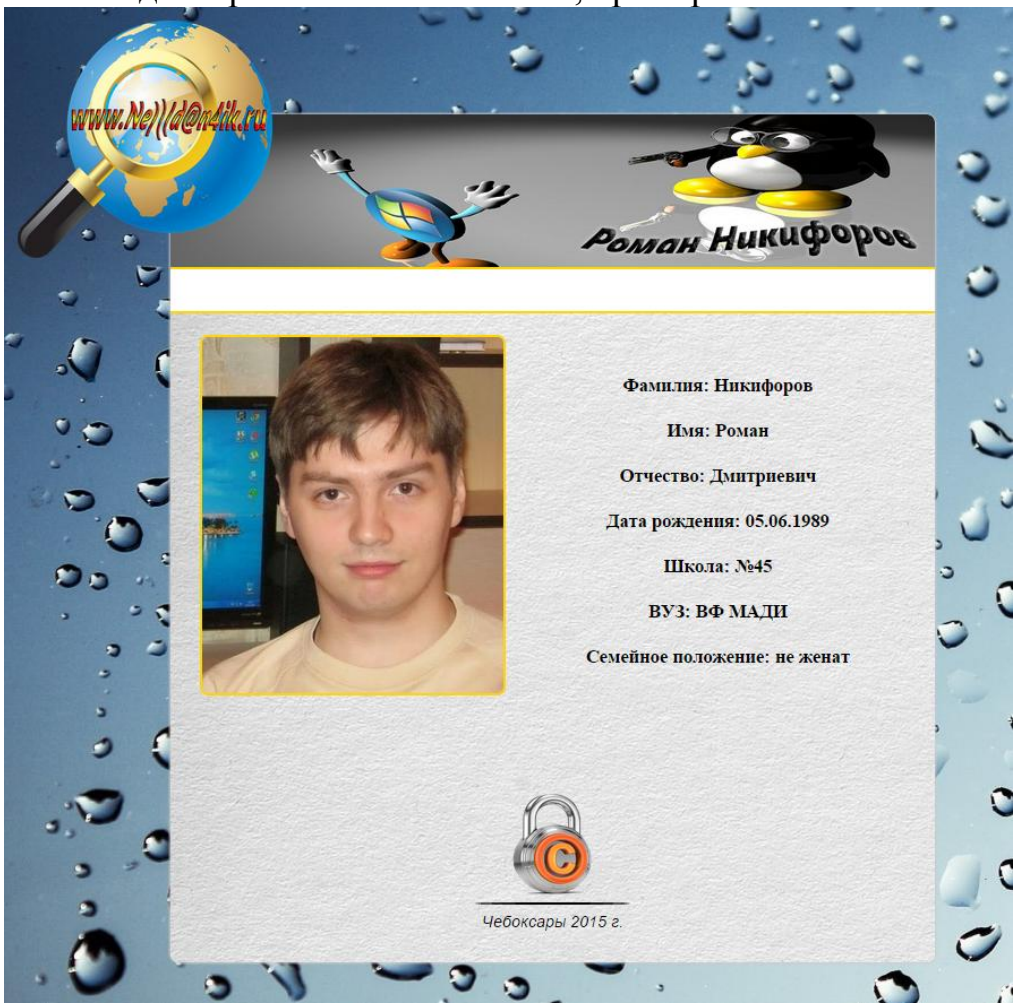

Рис. 18 Пример визитки

4. **Работа с веб - формами**

Функциональность сайтов, предоставляющих пользователю возможность ввода данных и получения результатов их обработки, обеспечивается использованием программ, работающих на стороне сервера — серверных приложений. Эти приложения обрабатывают полученные от посетителя Web-сайта данные и выдают результат в виде обычной Web-страницы. Именно для них в *HTML*предусмотрена возможность создания Web-форм и *элементов управления* — чтобы посетитель мог ввести данные, которые потом обработает *серверное приложение*

Вот основная схема работы серверного приложения.

1. Посетитель вводит в *элементы управления*, расположенные в Web-форме на Web-странице, нужные данные.
2. Введя данные, посетитель нажимает расположенную в той же Web-форме особую кнопку — кнопку *отправки данных*.
3. Web-форма кодирует введенные в нее данные и отправляет их серверному приложению, расположенному по указанному интернет-адресу.
4. Web-сервер перехватывает отправленные данные, запускает *серверное приложение* и передает данные ему.
5. *Серверное приложение* обрабатывает полученные данные.
6. *Серверное приложение* формирует Web-страницу с результатами обработки данных посетителя и передает ее Web-серверу.
7. Web-сервер получает сформированную серверным приложением Web-страницу и отправляет ее посетителю.

Для обеспечения взаимодействия между серверным приложением и пользователем используются веб - формы.

Форма предназначена для обмена данными между пользователем и сервером. Область применения форм не ограничена отправкой данных на *сервер*, с помощью клиентских скриптов можно получить *доступ* к любому элементу формы, изменять его и применять *по*своему усмотрению.

Документ может содержать любое количество форм, но одновременно на *сервер* может быть отправлена только одна форма. *По* этой причине данные форм должны быть независимы друг от друга.

Элементы управления

По сути, форма – *контейнер* для *элементов управления* (кнопок, списков, полей ввода).

Эти *элементы управления* уже "умеют" откликаться на действия посетителя: поля ввода — принимать введенные символы, флажки — устанавливаться и сбрасываться, *переключатели* — переключаться, списки — прокручиваться, выделять пункты, разворачиваться и сворачиваться, а кнопки — нажиматься. За обеспечение подобного функционала отвечает *браузер*, т.е. нет необходимости вручную писать соответствующий код.

К поддерживаемым *HTML* элементам управления относят:

- *Флажок* (*checkbox*) – предлагает пользователю ряд вариантов, и разрешает выбор нескольких из них.

- *Переключатель*(*radio*) – предлагает пользователю ряд вариантов, но разрешает выбрать только один из них.
- *Кнопка сброса формы*(*Reset*).При нажатии на кнопку сброса, все элементы формы будут установлены в то состояние, которое было задано в атрибутах по умолчанию, причем отправка формы не производится.
- *Выпадающий список* (select). Тэг <select> представляет собой выпадающий или раскрытый список, при этом одновременно могут быть выбраны одна или несколько строк.
- *Текстовое поле* (text). Позволяет пользователям вводить различную информацию.
- *Поле для ввода пароля* (password). Полностью аналогичен текстовому полю, за исключением того что символы, набираемые пользователем, не будут отображаться на экране.
- *Многострочное поле ввода текста* (*textarea*). Многострочное *поле ввода* текста позволяет отправлять не одну строку, а сразу несколько. По умолчанию *тег* создает пустое поле шириной в 20 символов и состоящее из двух строк.
- *Скрытое текстовое поле*. Позволяет передавать сценарию какую то служебную информацию, не отображая её на странице.
- *Кнопка отправки формы* (*submit*). Служит для отправки формы сценарию.
- *Кнопка для загрузки файлов* (*browse*). Служит для реализации загрузки файлов на сервер.
- *Рамка* (fieldset) Объект fieldset позволяет вам нарисовать рамку вокруг объектов. Имеет закрывающий тэг </fieldset>. Заголовок указывается в тэгах <*legend*> </*legend*>. Основное назначение объекта – задание различных *стилей оформления*.

Отправка данных на сервер

Для того чтобы успешно подготовить введенные посетителем данные и отправить их серверному приложению, Web-форма должна "знать" метод *отправки данных*, указывающий вид, в котором данные будут отправлены. Таких методов *HTML* поддерживает два:

1.Метод GET формирует из введенных посетителем данных набор пар вида <имя элемента управления>=<введенные в него данные>.

Эти пары добавляются справа к интернет-адресу серверного приложения, отделяясь от него символом ? (вопросительный знак); сами пары разделяются символами & (амперсанд). Полученный таким образом интернет-адрес отправляется Web-серверу, который извлекает из него интернет-адрес серверного приложения и сами данные.

2.Метод *POST* также формирует из введенных данных пары вида <имя элемента управления>=<введенные в него данные>. Но отправляет он их не в составе интернет-адреса, а вслед за ним, в качестве дополнительных данных.

Ввиду того, что курс посвящен основам клиентской разработки, мы не будем останавливаться на вопросах обработки информации сервером.

<form>

Тег <form> определяет форму *HTML* страницы.

Когда форма отправляется на *сервер, управление данными* передается программе, заданной атрибутом *action* тега *<form>*. Предварительно *браузер* подготавливает информацию в виде пары "имя=значение", где имя определяется атрибутом name тега *<input>*, а *значение* введено пользователем или установлено в *поле* формы *по* умолчанию.

Внутрь контейнера *<form>* помещаются другие теги, при этом сама форма никак не отображается на *веб-странице*, видны только ее элементы и результаты вложенных *тегов*.

Оформляется *тег* следующим образом:

<form>
.....
</form>

К атрибутам *<form>* относятся:

- *accept-charset* – определяет *кодировку*, в которой сервер может принимать и обрабатывать данные;
- *action* - определяет адрес программы или документа, который обрабатывает данные формы;
- autocomplete – включает *автозаполнение* полей формы;
- enctype – определяет способ *кодирования* данных формы. Может принимать следующие значения:
  - application/x-www-*form*-urlencoded – вместо пробелов ставится +, символы вроде русских букв кодируются их шестнадцатеричными значениями;
  - multipart/*form*-data – данные не кодируются;
  - text/plain – пробелы заменяются знаком +, буквы и другие символы не кодируются.
- *method* – метод протокола HTTP (GET или *POST*);
- name – определяет имя формы;
- novalidate – отменяет встроенную проверку данных формы на корректность ввода;
- *target* –задает *имя окна* или *фрейма*, куда обработчик будет загружать возвращаемый результат.

Задание 4. Разработка динамического web-сайта

Задание:

Написать сайт визитку состоящий из 5 и более страниц содержащих гиперссылки.

Сайт должен содержать следующие разделы (разделы создавать с помощью маркированного списка):

- Главная (Обо мне). Раздел должен содержать ваше фото и краткую информация о вас: ФИО, дату рождения, школу, ВУЗ, семейное положение.

- Хобби (Увлечения) - опишите ваши увлечения при оформлении добавьте изображения.
- Медиа (Галерея) - оформите фотоальбом.
- Контакты (Связь со мной) - укажите контактную информацию о вас: страна проживания, регион, город, улица, почтовый индекс, телефон, e-mail, skype. Оформите раздел используя таблицу.

Создайте два раздела на выбор с содержимым стилизованным на свое усмотрение. Используйте гиперссылки (сделать гиперссылки двумя способами: текстом и картинкой) чтобы перейти на внешний сайт, например для чтения книги.

В различных разделах должны быть применены различные методы оформления.

Пример содержания страниц:

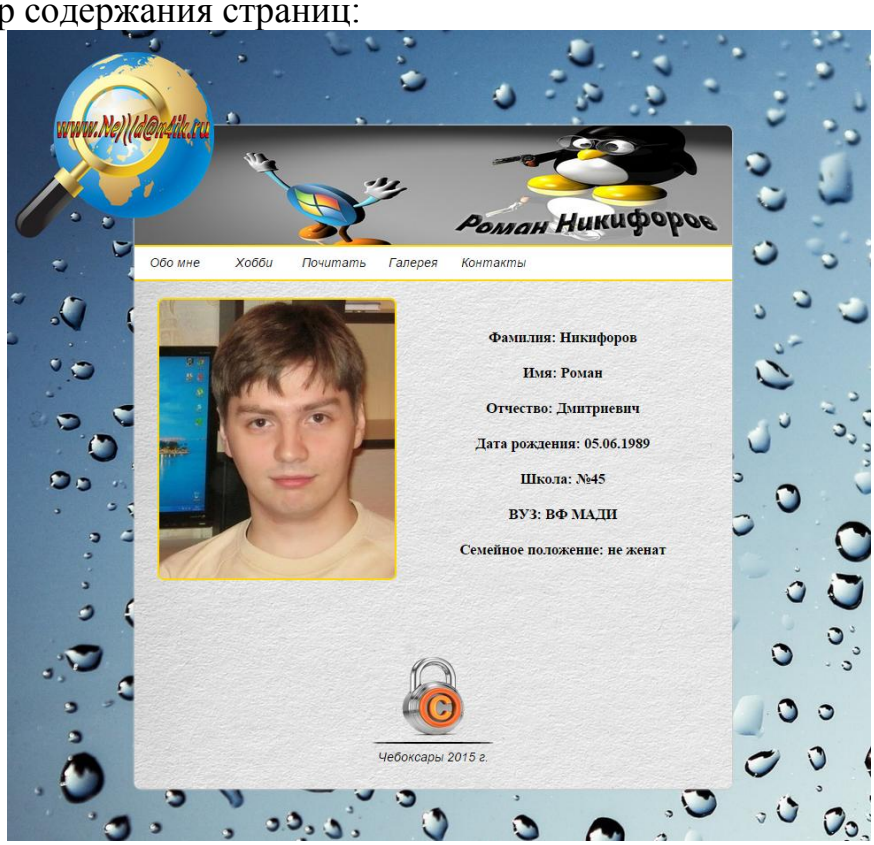

Рис. 19 Раздел обо мне

Рис. 20 Позиционирование элементов на странице

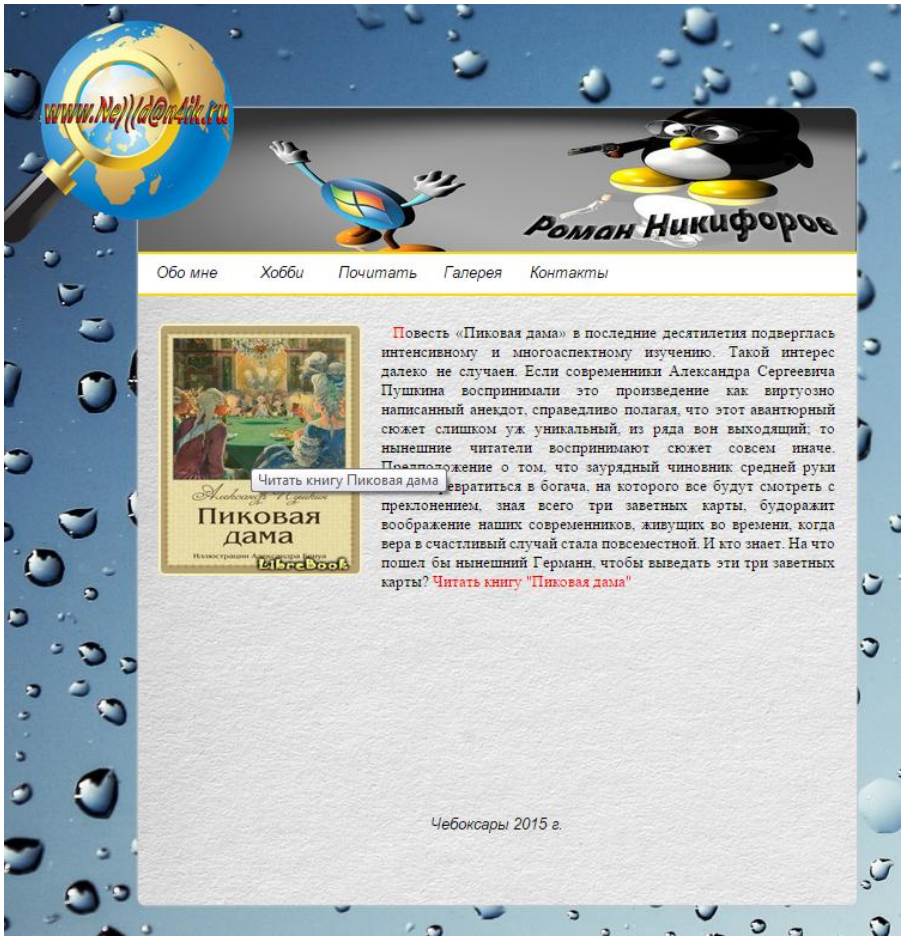

Рис. 21 Гиперссылки в виде картинки и текста

Рис. 22 Раздел галерея

Рис. 23 Оформление галереи

Рис. 24 Оформление контента в виде таблицы

Татьяна Анатольевна Изосимова
Инна Витальевна Ксенофонтова

ЛАБОРАТОРНЫЙ ПРАКТИКУМ
по дисциплине
«Интернет технологии»
для направления 09.03.01 «Информатика и вычислительная техника»
профиля
«Автоматизированные системы обработки информации и управления»

Компьютерная верстка – Ксенофонтова И.В.

Подписано в печать . Формат 60x84/16.
Бумага писчая. Печать оперативная. Усл. печ.л.4 .
Тираж экз. Заказ № . Цена свободная

Федеральное государственное бюджетное образовательное учреждение
Высшего профессионального образования
«Московский автомобильно-дорожный
государственный технический университет (МАДИ)»
Волжский филиал