Содержание

Пояснительная записка

Список рекомендуемой литературы

***Раздел 1. Основы создания веб-страницы.***

Тема 1.1. Создание веб-страницы  
Тема 1.2. Организация текста внутри HTML-документа

Тема 1.3. Цветовая гамма HTML - документа

Тема 1.4. Графика

Тема 1.5. Таблицы

Тема 1.6. Гипертекстовый документ.

***Раздел 2. Веб-дизайн***

Тема 2.1. Дополнительные возможности создания веб-страниц

Тема 2.2. Свойства CSS

Тема 2.3. Модель форматирования CSS

**Пояснительная записка**

Данные методические указания составлены в соответствии с рекомендациями Министерства образования науки России по разработке методических указаний для студентов образовательных учреждений среднего профессионального образования и в соответствии с рабочей программой по специальности 035002 Издательское дело (углубленной подготовки).

Цель методических указаний – реализация государственных требованиям к уровню подготовки выпускников по специальности при заочной форме обучения по дисциплине «Создание web-сайтов» в условиях действия государственного образовательного стандарта среднего профессионального образования.

Методические указания для 5 курса заочного отделения и 3 курса очного отделения состоят из 2 разделов. В каждой теме раздела методического пособия рассматривается теоретический материал, который знакомит студентов с разработкой и созданием сайтов, раскрывает основные теоретические аспекты программирования на языках HTML и CSS.

Изучение дисциплины предусматривает аудиторные (лекционные и практические) и самостоятельные домашние занятия. Последнее – это изучение теории по рекомендуемой литературе, отработка практических навыков, конспектирование материала, так как методические указания только организуют и направляют самостоятельную работу студента.

В список рекомендуемой литературы вошли книги, которые помогут в изучении предмета, а также дадут дополнительную информацию

**Список рекомендуемой литературы**

**Основная литература**

1. Комолова Н.В. HTML. Самоучитель. – СПб.: Питер, 2011.
2. Прохоренок Н.А. HTML, JavaScript, PHP, и MySQL. Джентльменский набор Web-мастера + CD.– СПБ.: БХВ-Петербург, 2011.
3. Титтел Э., Ноубл Д. HTML, XHTML и CSS для чайников. – М.: Вильямс, 2011.

**Дополнительная литература.**

1. Мультимедийный самоучитель на CD-ROMTeacherProWeb-дизайн / Под ред. Катхановой Ю.Ф., Гринберг Т.С., Антонова Р., Соловьева И. – М, ММТИДД, 2007.

2. Марчук Е.В. HTML и CSS. . Самоучитель – М: Издательский дом «Вильямс». 2007.

3. Создание Web-сайтов без посторонней помощи / под ред. В. Н. Печникова. – М.: Технический бестселлер, 2006.

6. http://www.mrdesign.krasline.ru/

7. http://www.postroika.ru/

8. http://www.design.ru/kovodstvo/

9. http://www.citforum.ru/

10. http://netdesigner.ru/

11. http://book.promo.ru/

***Раздел 1. Основы создания веб-страницы.***

Тема 1.1. Создание веб-страницы

***Содержание программы***

Основные сведения о Web- проектировании. Знакомство с HTML. Теги HTML. Структура веб-страницы. Заголовок документа. Тело документа. Атрибуты тегов. Заголовки. Абзацы. Разделительная линия.

***Методические указания***

**HTML (HyperText Markup Language)** – язык разметки (маркировки) гипертекста.

Именно язык разметки дает браузеру необходимые инструкции о том, как отображать тексты и другие элементы страницы на мониторе. Важно заметить, что не только различные браузеры, но и различные их версии могут по-разному воспринимать и отображать на экране код. Следовательно, некоторые элементы корректно выглядящие в браузере **Operа** могут выглядеть иначе в **Internet Explorer** и других браузерах.

При верстке (грубо говоря, создании страниц) HTML-страниц это необходимо помнить, и просматривать их в самых популярных браузерах типа Mozilla Firefox, Internet Explorer и Operа.

Код разметки в HTML состоит из так называемых «тегов». Теги предоставляют информацию браузерам о форматировании и разметке страницы. НТМL-тег состоит из имени, за которым может следовать необязательный список атрибутов тега. Текст тега заключается в угловые скобки (< и >). Простейший вариант тега - имя, заключенное в угловые скобки, например <HEAD> или <i>. Для более сложных тегов характерно различие атрибутов, которые могут иметь конкретные значения для видоизменения функции тега.

Атрибуты тега следуют за именем и отделяются друг от друга одним или несколькими знаками табуляции, пробелами или символами возврата к началу строки. Порядок записи атрибутов в теге значения не имеет. Значение атрибута, если таковое имеется, следует за знаком равенства, стоящим после имени атрибута. Если значение атрибута - одно слово или число, то его можно просто указать после знака равенства, не выделяя дополнительно. Все остальные значения необходимо заключать в одинарные или двойные кавычки, особенно если они содержат несколько разделенных пробелами слов. Длина значения атрибута ограничена 1024 символами. Регистр символов в именах тегов и атрибутов не учитывается, чего нельзя сказать о значениях атрибутов. Особенно важно использовать нужный регистр при вводе URL других документов в качестве значения атрибута HREF. Чаще всего НТМL-теги состоят из начального и конечного компонентов, между которыми размещаются текст и другие элементы документа. Имя конечного тега идентично имени начального, но перед именем конечного тега ставится косая черта (/) (например, для тега стиля шрифта - курсив <i> закрывающая пара представляет собой </i>, для тега заголовка <ТIТLЕ> закрывающей парой будет </ТIТLЕ>). Конечные теги никогда не содержат атрибутов.

НТМL-тег состоит из имени, за которым может следовать необязательный список атрибутов тега. Текст тега заключается в угловые скобки (< и >). Простейший вариант тега - имя, заключенное в угловые скобки, например <HEAD> или <i>. Для более сложных тегов характерно различие атрибутов, которые могут иметь конкретные значения для видоизменения функции тега.

Атрибуты тега следуют за именем и отделяются друг от друга одним или несколькими знаками табуляции, пробелами или символами возврата к началу строки. Порядок записи атрибутов в теге значения не имеет. Значение атрибута, если таковое имеется, следует за знаком равенства, стоящим после имени атрибута. Если значение атрибута - одно слово или число, то его можно просто указать после знака равенства, не выделяя дополнительно. Все остальные значения необходимо заключать в одинарные или двойные кавычки, особенно если они содержат несколько разделенных пробелами слов. Длина значения атрибута ограничена 1024 символами. Регистр символов в именах тегов и атрибутов не учитывается, чего нельзя сказать о значениях атрибутов. Особенно важно использовать нужный регистр при вводе URL других документов в качестве значения атрибута HREF. Чаще всего НТМL-теги состоят из начального и конечного компонентов, между которыми размещаются текст и другие элементы документа. Имя конечного тега идентично имени начального, но перед именем конечного тега ставится косая черта (/) (например, для тега стиля шрифта - курсив <i> закрывающая пара представляет собой </i>, для тега заголовка <ТIТLЕ> закрывающей парой будет </ТIТLЕ>). Конечные теги никогда не содержат атрибутов.

# Структура HTML-документа

Содержание простейшего HTML-кода с пояснениями представлено в таблице 1.

Таблица 1. Пример HTML-кода

|  |  |
| --- | --- |
| **Текст HTML-программы** | **Комментарий** |
| <HTML>  <HEAD>  <META content="charset=windows-1251">  <TITLE>Упражнение 1</TITLE>  </HEAD>  <BODY>  <H1>Первый HTML-документ</H1>  <HR>  <P>  Корова не похожа на лошадь.  А лошадь не похожа на корову.  Именно это сходство  мы и берем за основу.  </P>  </BODY>  </HTML> | Начало HTML-документа.  Начало заголовка.  Информация о документе.  Название документа.  Конец заголовка.  Начало тела.  Заголовок.  Горизонтальная линия.  Начало абзаца.  Абзац.  Конец абзаца.  Конец тела.  Конец HTML-документа. |

Команды языка (теги) заключаются в угловые скобки. Как правило, теги парные. Первый тег открывает описание команды, определяет ее начало, второй, отличающийся от первого наличием косой черты "/" перед ключевым словом (именем тега) - закрывает его. Тег <HTML> должен открывать программу, а тег </HTML> - закрывать ее.

## Заголовок программы

Между парой тегов <TITLE> и </TITLE> располагается имя HTML-документа. Это имя браузер использует в заголовке окна Windows, в котором показывает документ. Пример описания заголовка отображен на рисунке 1.1.

Microsoft Internet Explorer

Рисунок 1.1 – Пример заголовка HTML-страницы

### Тело программы

Между тегами <BODY> и </BODY> располагаются команды, следуя которым браузер выводит информацию в окно документа.

# Форматирование текста

Для форматирования текста используются различные теги, позволяющие задавать параметры для текста. Форматирование текста позволяет создавать более интересные и индивидуальные WEB-документы, с параметрами задаваемыми разработчиком.

## Заголовки

Заголовки являются важной частью документа. Они разделяют информацию на отдельные логические части и поэтому существенно улучшают ее восприятие.

Заголовки бывают разные по значению, или по уровню. В HTML-документе поддерживаются заголовки шести уровней при помощи конструкции:

<Hn>Текст заголовка</Hn>

Здесь в качестве n следует использовать одну из цифр 1-6, обозначающую уровень заголовка.

Заголовки выделяются браузером при показе HTML-документа размером шрифта и жирностью.

### Горизонтальная линия

Линия - простейший графический элемент, который умеет строить браузер. Горизонтальная линия задается тегом <HR>, и этот тег не имеет парного закрывающего. Браузер выполняя эту команду, выведет на экран черту.

У тега <HR> существуют свойства, задав которые можно изменять стиль линии.

1. size - толщина линии. Задается в абсолютных единицах измерения (cm- сантиметры, mm- миллиметры, px-пиксели, pt-пункты);
2. color – цвет линии.

Горизонтальные линии удобно использовать для дополнительного выделения (или разделения) элементов документа на экране.

#### Абзац

Задается абзац тегами <P> и </P> , между которыми помещается текст.

Пример:

<HTML>

<BODY>

<P>Красная строка используется </P>

</BODY>

</HTML>

##### Принудительный разрыв строки

Команда <BR> заставляет браузер продолжить вывод абзаца с новой строки. Тег <BR> используется одиночно, без парного тега.

Пример:

<HTML>

<BODY>

Разрыв строки не используется

Разрыв строки используется <BR>

</BODY>

</HTML>

***Вопросы для самопроверки***

1. Что такое HTML?
2. Что такое тег?
3. Назовите основные теги HTML – документа.

**Тема 1.2. Организация текста внутри HTML-документа**

***Содержание программы***

Теги маркированных списков, нумерованных списков, теги для списков определений, вложенные списки

## 

## *Методические указания*

## Маркированный список

Каждый элемент этого списка браузер предваряет меткой в начале строки, а сами строки смещает вправо.

Команда UL, задающая маркированный список имеет следующую структуру, представленную в таблице2:

Таблица2. Структура команды для создания маркированных списков

|  |  |
| --- | --- |
| Описание команды: | Пример: |
| <UL> <LI>первый элемент списка</LI> <LI>второй элемент списка</LI>. . <LI>последний элемент списка</LI> </UL> | <UL>  <LI> Литература </LI>  <LI>Русский язык </LI>  <LI>Математика </LI>  </UL> |

Примечание. Конечный тег </LI> является необязательным.

Вид метки, которую браузер помещает перед каждым элементом списка, настраивается при помощи атрибута type. Этот атрибут может отсутствовать или принимать одно из трех значений:

|  |  |
| --- | --- |
| Начальный тег: | Вид метки на экране: |
| <UL> | По умолчанию: диск |
| <UL type = circle> | Окружность |
| <UL type = disc> | Диск |
| <UL type = square> | Квадрат |

### 

### Нумерованный список

В нумерованных списках каждый элемент снабжен номером, вид и начальное значение которого настраивается специальным атрибутом.

Нумерованный список задается при помощи команды <OL>. Описание структуры команды представлена в таблице3.

Таблица 3. Структура команды для создания нумерованных списков

|  |  |
| --- | --- |
| Описание команды | Пример |
| <OL>    <LI>первый элемент списка</LI>    <LI>второй элемент списка</LI>    . . .    <LI>последний элемент списка</LI> </OL> | <OL>    <LI>Один</LI>   <LI>Два</LI>    . . .    <LI>N</LI> </OL> |

Замечание. Конечный тег </LI> является необязательным.

Вид номера определяется значением атрибута type.

|  |  |
| --- | --- |
| Начальный тег: | Вид номера на экране: |
| <OL> | Нумерация выполняется арабскими цифрами (1, 2, 3, ...). |
| <OL type = 1> | Нумерация выполняется арабскими цифрами (1, 2, 3, ...). |
| <OL type = A> | Нумерация выполняется прописными буквами (A, B, C, ...). |
| <OL type = a> | Нумерация выполняется строчными буквами (a, b, c, ...). |
| <OL type = I> | Нумерация выполняется большими римскими цифрами (I, II, III, IV, ...). |
| <OL type = i> | Нумерация выполняется малыми римскими цифрами (i, ii, iii, iv, ...). |

Естественную нумерацию элементов списка (с единицы) можно изменить при помощи атрибута start = n. Число n задает начальное значение нумерации. Если для нумерации используются буквы или римские цифры, то стартовое значение равно n-му элементу в системе нумерации (например, 4 означает D или IV).

Пример:

<HTML>

<BODY>

<OL start=4>

<LI>четвертый пункт</LI>

<LI>Пятый пункт</LI>

<LI>пункт n</LI>

</OL>

</BODY>

</HTML>

# Списки определений

Интересной разновидностью списков являются списки определений. Как можно догадаться из названия, первоначально эти списки были введены для более наглядного представления определений терминов.

Список определений задается внутри HTMLэлемента DL (для его задания используются парные теги <DL> и </DL>). Каждый элемент списка определений состоит из двух частей: из термина (элемент DT) и определения термина (элемент DD). Пример текста HTML-документа, содержащего список определений, приведен ниже.

**Пример: Список определений**

<TITLE>Список определений</TITLE>

<BODY>

<DL>

<DT>HTML

<DD>Широко используемый язык гипертекстовой разметки

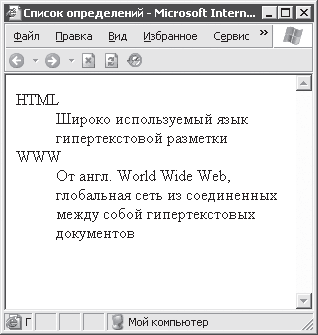
<DT>WWW

<DD>От англ. World Wide Web, глобальная сеть из соединенных между собой гипертекстовых документов

</DL>

</BODY>

Как можно видеть браузер по-разному отображает сами термины и определения этих терминов, причем делает это так, что сразу понятно, где определение, а где термин.



Список определений

Все особенности использования закрывающих тегов </DT> и </DD> аналогичны особенностям использования тега </LI>, то есть, несмотря на то что эти теги определены, использовать их совершенно не обязательно.

#### Вложенные списки

Элемент любого списка сам может быть списком. Уровень вложенности не ограничен, это позволяет использовать списки для отображения сложных [иерархических](../AppData/Local/Temp/Rar$DIa0.701/Robot_Htm/spravka/ierarh.htm)структур.

***Вопросы для самопроверки:***

1. Назовите теги маркированных списков.

2. Назовите теги нумерованных списков.

3. Назовите теги списков определений.

***Тема 1.3. Цветовая гамма HTML – документа***

***Содержание программы***

HTML-метки, определяющие цветовую гамму.   
Список атрибутов внутри метки <BODY>. Цвет фона.

###### **Методические указания**

###### **Цвет фона и цвет шрифта**

Цвета для фона и шрифта на всем протяжении документа можно задать при помощи атрибутов bgcolor и text в теге <body>, например:

|  |  |  |
| --- | --- | --- |
| |  | | --- | | <HTML>  <HEAD>  ....  </HEAD>  <BODY  bgcolor=aqua text=black>  </BODY>  </HTML> | | Это означает, что в документе заданы черные символы (black) на бирюзовом фоне (aqua). |

Атрибуты цветов, заданные в теге <BODY> действуют до парного тега </BODY>, то есть до конца документа. Цвет фона можно задать только один раз, а цвет шрифта можно менять внутри документа много раз при помощи команды <FONT> ...</FONT>. Установка цвета в этой команде выполняется при помощи атрибута color, например:

|  |  |  |
| --- | --- | --- |
| |  | | --- | | <HTML>  <HEAD>  ....  </HEAD>  <BODY bgcolor=aqua text=black>  ....  <FONT color=maroon>  ....  </FONT>  ....  </BODY>  </HTML> | | Заданы цвета для всего документа    Задан новый цвет шрифта (коричневый)  Восстановлен цвет шрифта (черный) |

Данным кодом можно назначить абсолютно любой оттенок. Если у цвета или оттенка есть название, то вместо***цифр и букв***можно написать слово, например красный***#FF0000***или***"red"***. Вот, к примеру, часто встречающиеся цвета: (#FF0000 - это нули 0000, а не буквы OOOO, не путайте)

[*Белый*](http://www.dostavka.ru/Kris-84-sm-id_6520314?partner_id=admitad&utm_source=admitad&utm_medium=cpa&utm_campaign=&utm_content=6520314)#FFFFFF White

Черный #000000 Black

Голубой #0000FF Blue

Зеленый #00FF00 Green

Желтый #FFFF00 Yellow

Оранжевый #FFA500 Orange

Красный #FF0000 Red

В HTML действует общее важное правило: конструкции языка могут быть вложенными, то есть, внутри одной команды могут располагаться другие. Указания вложенной команды могут изменять указания внешних команд, но когда вложенная команда заканчивается, восстанавливаются полномочия внешней инструкции.

***Вопросы для самопроверки:***

1. Каким тегом задается цвет фона документа?

2. Каким тегом задается цвет текста в документе?

***Тема 1.4. Графика***

***Содержание программы***

Параметры и форматы графических файлов.   
Изображение в HTML документе.

***Методические указания***

Для создания более интересных интерфейсов WEB-страниц используются различные изображения. Браузеры распознают различные форматы изображений, самыми распространенными из них являются два графических формата GIF и JPG.

## 

## Программирование картинки

Картинку можно помещать в документ почти так же, как текстовый символ при помощи команды [<IMG>](../AppData/Local/Temp/Rar$DIa0.701/Robot_Htm/add/tags/html32.htm#IMG). Команда не имеет парного закрывающего тега, но имеет много атрибутов.

*Атрибут src = имя файла*

Самым главным атрибутом команды является атрибут src, при помощи которого можно задать имя файла с картинкой. Например, команда <IMG src = img1.gif> - браузер отобразить на экране графический файл img1.gif из текущего каталога. Обычно графические файлы не смешивают с HTML-текстами, а помещают в отдельный каталог PIC, который является подкаталогом для каталога с программами (html-файлами). Тогда команда вывода графики будет иметь вид: <IMG src = ./pic/img1.gif>

*Атрибут alt = "текст надписи"*

Если браузер не находит картинки в указанном месте на диске, он вместо нее рисует на экране маленький прямоугольник и вписывает в него надпись, которая задана атрибутом alt: <IMG src=Ivan Fedorov.jpg  alt="Первопечатник ">. Атрибут alt рекомендуется использовать всегда. Даже если картинка найдена браузером, задание надписи не будет лишним: стоит пользователю остановить курсор на картинке, как надпись появится в маленьком окошке и сообщит дополнительную информацию.

*Атрибуты width = n и height = m*

Эти атрибуты задают ширину и высоту (в [пикселах](../AppData/Local/Temp/Rar$DIa0.701/Robot_Htm/spravka/pixel.htm))прямоугольника, в который выводится картинка. Если атрибуты не заданы, картинка рисуется в естественных размерах. Когда размеры прямоугольника не совпадают с размерами картинки, браузер производит масштабирование картинки, подгоняя ее под указанные величины.

*Атрибут border = n*

Атрибут позволяет задать рамку иллюстрации толщиной в n пикселов.   
При отсутствии атрибута, или при значении n = 0 рамка не рисуется.

*Атрибут align*

Этот атрибут позволяет определять положение иллюстрации по отношению к соседним элементам документа. Используются следующие значения атрибута:

align=top - вертикальное выравнивание по верхнему краю;

align=middle - вертикальное выравнивание по центру;

align=bottom - вертикальное выравнивание по нижнему краю;

align=left - горизонтальное выравнивание по левому краю;

align=right - горизонтальное выравнивание по правому краю.

# Фоновые изображения

Используя атрибут bgcolor команды BODY, можно выкрасить фон документа в один из 16-ти основных цветов, или использовать более сложные цвета, задавая их в виде шестнадцатеричных чисел. Вместо одноцветной заливки в качестве фона документа можно использовать картинку. Эта картинка будет, повторяясь, укладываться в экранное поле наподобие паркетных плиточек. Чтобы "паркет" смотрелся красиво, необходима визуальная стыковка плиточек между собой. Это надо учитывать при создании фоновой иллюстрации, так же важно, чтобы фон оставался фоном, а не переходил на передний план восприятия, не отвлекал внимание пользователя от главных элементов документа, не мешал читать текст.

Фоновая картинка задается в команде BODY атрибутом background:

<BODY background = "имя файла с картинкой и тип файла">

Пример:

<BODY background = "./pic/fon1.jpg">

***Вопросы для самопроверки:***

1. Назовите тег вставки иллюстрации в документ?

2. Перечислите основные атрибуты тега вставки иллюстрации.

3. Назовите принцип вставки фонового изображения в документ.

***Тема 1.5. Таблицы***

***Содержание программы***

Как устроена таблица, метки таблицы в HTML.   
Атрибуты таблицы.

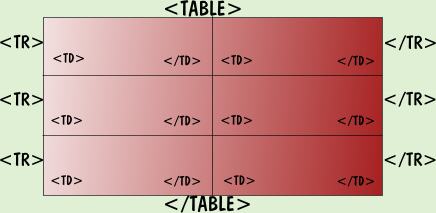
***Методические указания***

В практическом HTML-программировании таблицы незаменимы для создания левых и правых полей страницы, выравнивания элементов на экране, многоколонной верстки, наложения картинок друг на друга, а так же для отображения на экране табличного материала.

Таблица представляет собой прямоугольник, расчерченный на клетки. Если на клетки смотреть по вертикали, то они образуют столбцы, если по горизонтали - строки. Каждая строка и столбец в таблице может иметь свой заголовок. Заголовки записываются в первой строке и первом столбце. В общем случае таблицы используют для хранения информации с двумя характерными признаками. Значения одного признака расписано в заголовках строк, другого - в заголовках столбцов. А в каждую клетку таблицы заносят значение, одновременно обладающее признаком строки и столбца, на пересечении которых она расположена.

## Программирование таблицы

Таблица задается командой[<TABLE> ... </TABLE>](../AppData/Local/Temp/Rar$DIa0.701/Robot_Htm/add/tags/html32.htm#TABLE).Внутри этих тэгов задаются строки командами <TR> ... </TR>, внутри строк задаются клетки (ячейки) командами <TD> ... </TD>. Таким образом, по строкам, описывается вся структура таблицы. На рисунке ниже изображена структура построение таблиц в HTML-документах.



Структура таблицы

Пример:

|  |
| --- |
| <TABLE border=1>  <TR> <!-- Первая строка -->  <TD>(1,1)</TD> <!-- Первая ячейка -->  <TD>(1,2)</TD> <!-- Вторая ячейка -->  </TR>  <TR> <!-- Вторая строка -->  <TD>(2,1)</TD> <!-- Первая ячейка -->  <TD>(2,2)</TD> <!-- Вторая ячейка -->  </TR>  <TR> <!-- Третья строка -->  <TD>(3,1)</TD> <!-- Первая ячейка -->  <TD>(3,2)</TD> <!-- Вторая ячейка -->  </TR>  </TABLE> |

Атрибуты команды TABLE:

|  |  |
| --- | --- |
| Атрибут: | Назначение: |
| align= (left, right) | Выравнивание по горизонтали. |
| width= (число или процент) | Ширина таблицы. |
| cellpadding= (число) | Расстояние между содержимым  ячейки и рамкой. |
| cellspacing= (число) | Расстояние между ячейками таблицы. |
| bgcolor= (цвет) | Цвет фона таблицы. |
| background= (файл) | Фоновая картинка. |
| border= (число) | Ширина линий рамки. |
| bordercolor= (цвет) | Цвет линий рамки. |
| bordercolordark= (цвет) | Цвет рамки (снизу и справа). |
| bordercolorlight= (цвет) | Светлый цвет рамки (сверху и слева). |

### Атрибуты команды TR

HTML-таблица состоит из строк, каждая из которых задается командой

<TR> описание ячеек табличной строки </TR>. Варьируя атрибутами команды TR, можно установить те или иные свойства одновременно для всех ячеек строки.

|  |  |
| --- | --- |
| Атрибут: | Назначение: |
| align= (left, center, right) | Выравнивание по горизонтали. |
| valign=(top, center,bottom, baseline) | Выравнивание по вертикали. |
| bgcolor= (цвет) | Цвет фона. |
| bordercolor= (цвет) | Цвет линий рамки. |
| bordercolordark= (цвет) | Цвет рамки (снизу и справа). |
| bordercolorlight= (цвет) | Светлый цвет рамки (сверху и слева). |

#### Атрибуты команды TD (TH)

Каждая табличная строка состоит из ячеек, которые последовательно описываются командами TD (обычная ячейка) и (или) TH (ячейка-заголовок):

<TR> описание содержимого ячейки </TR>

<TH> описание содержимого ячейки </TH>

Содержимое обычной ячейки по умолчанию горизонтально выравнивается по левому краю, а ячейки-заголовка - по центру. Кроме того, текст в ячейке-заголовке выделяется полужирным шрифтом. Команды TD и TH имеют следующие атрибуты:

|  |  |
| --- | --- |
| Атрибут: | Назначение: |
| align= (left, center, right) | Выравнивание по горизонтали. |
| valign=(top, center,bottom, baseline) | Выравнивание по вертикали. |
| width= (число или процент) | Ширина ячейки. |
| bgcolor= (цвет) | Цвет фона. |
| background= (файл) | Фоновая картинка. |
| bordercolor= (цвет) | Цвет линий рамки. |
| bordercolordark= (цвет) | Цвет рамки (снизу и справа). |
| bordercolorlight= (цвет) | Светлый цвет рамки (сверху и слева). |
| nowrap | Выключение автоматического разрыва строк. |
| colspan= (число) | Ширина большой ячейки (в столбцах (объединение)). |
| rowspan= (число) | Высота большой ячейки (в строках (объединение)). |

##### 

###### **Таблицы, вложенные друг в друга**

Используя таблицы, можно выровнять информационные блоки на экране и разместить их в соответствии с дизайн-проектом документа. Метод вложенных таблиц позволяет создавать более интересные интерфейсы WEB-страниц.

Пример кода создания вложенных таблиц:

<html><body>

<table border=0 cellpadding=0 cellspacing=0 width=100%>

<tr>

<td><img src="dino.gif" width=110 height=111 align=center></td>

<td width="99%" bgcolor="#006600" align=center>

<font size="6"face="comic sans ms" color="#66ff66"><strong>

Вас приветствует "Красный уголок"</strong></font></td>

</tr><tr>

<td bgcolor="#006600" align=center>

<font face="comic sans ms" size=7 color="#66ff66" style="comic sans ms">

выбери<br>тему:

<td>

<table border=0 bordercolor="#ffffff" cellpadding=0 cellspacing=1>

<tr>

<td align=center><img src="704.jpg" width=180 height=160></td>

<td width=100% align="center">

<font face="comic sans ms" size=6 color="#006600">звери</font></td>

</tr><tr>

<td align=center><img src="469017.jpg" width=180 height=160></td>

<td width=100% align="center"><font face="comic sans ms" size=6 color="#006600">птицы</font></td>

</tr> <tr>

<td align=center><img src="i427047.jpg" width=180 height=160></td>

<td width=100% align="center"><font face="comic sans ms" size=6 color="#006600">рыбы</font></td>

</tr>

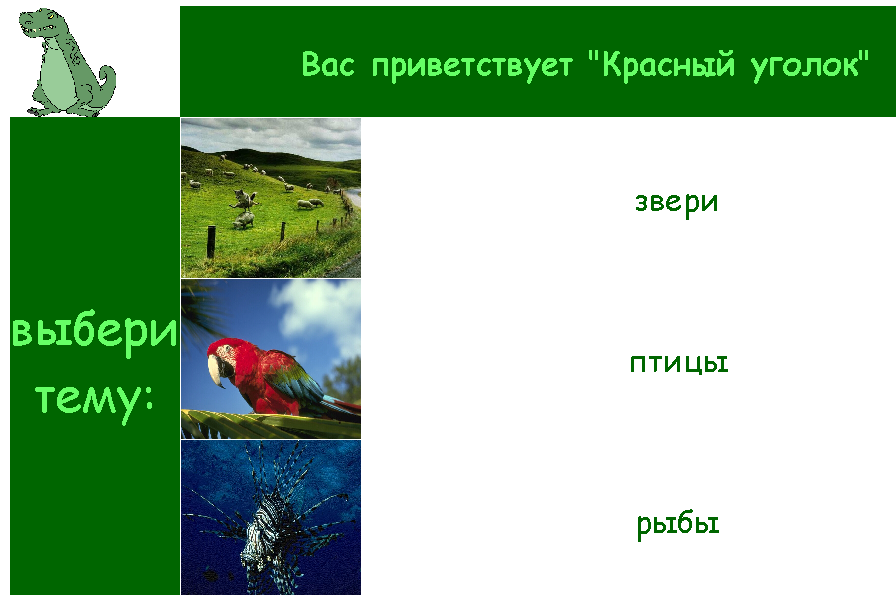
</table>

</td>

</tr>

</table></html>

Результат исполнения кода представлен на рисунке



Пример использования вложенных таблиц для оформления HTML-страниц

***Вопросы для самоконтроля:***

1. Назовите тег таблицы и основные атрибуты.

2. Назовите атрибуты строк.

3. Назовите атрибуты ячеек.

**Тема 1.6. Гипертекстовый документ**

***Содержание программы***

Способы организации гипертекстовых документов. Текстовые ссылки. Изображения-ссылки.

***Методические указания***

Для навигации по сайту используются ссылки. Для перехода по ссылке необходимо щелкнуть по ней мышкой, для возврата из ссылки необходимо использовать навигационную кнопку браузера "Назад" ("Back"). Если ссылки образуют вложенную цепочку, то кнопки "Назад" ("Back") и "Вперед" ("Forward") можно использовать для движения по пройденному ссылочному пути в обе стороны. Они работают как традиционные операции "откатка" и "накатка" в большинстве прикладных программ.

Для задания гипертекстового перехода внутри документа используют две команды [<A>](../AppData/Local/Temp/Rar$DIa0.701/Robot_Htm/add/tags/html32.htm#A). Первая команда с атрибутом href является источником перехода, вторая с атрибутом name - приемником.

|  |  |
| --- | --- |
| Общий вид программы | Комментарий |
| ...  <A href=#метка>текст</A>  ...  ...  ...  <A name=метка></A>  ... | Задание перехода по метке  На экран выводится ссылка:[текст](../AppData/Local/Temp/Rar$DIa0.701/Robot_Htm/04.HTM#метка1#метка1)  Метка. Сюда браузер приходит по ссылке.  На экране ничего не отображается. |

Для организации перехода внутри документа нужно выбрать имя для метки. Имя должно быть уникальным в HTML-программе, т.е. других меток с таким именем в программе быть не должно, т.к. для одинаковых меток браузер выполняет переход на самую первую метку в программе. Имя может заключаться в кавычки. Имя может состоять из любых символов. В том случае, когда имя содержит пробелы или ключевые символы HTML, кавычки обязательны.

Запрограммировать переход по метке можно согласно примеру:

<Ahref=#метка>текст</A>,   
где:

**#** - ключевой символ;

**метка** - выбранное имя для метки;

**текст** - запись, которая будет выглядеть на экране браузера как ссылка.

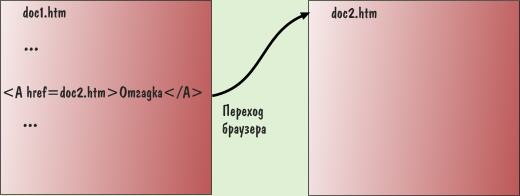
Установить тег с меткой в нужном месте HTML-программы можно следующим образом: перед фрагментом HTML-программы, на который должен выполниться переход записать команду-метку: <A name =метка>.

При этом имя метки должно быть то же самое, что и в команде, задающей переход, парный тег </A>может быть опущен.

##### Переход к другому документу

Для того, чтобы браузер загрузил в свое окно новый HTML-документ, необходимо прописать в программе ссылку при помощи команды <A> с атрибутом href=имя\_файла.

|  |  |
| --- | --- |
| Общий вид программы | Комментарий |
| ...  <A href=имя\_файла>текст</A>  ... | Выполнить файл "имя\_файла".  На экран выводится ссылка:[текст](../AppData/Local/Temp/Rar$DIa0.701/Robot_Htm/04.HTM#метка2#метка2)  При щелчке пользователя на ссылке  браузер строит на экране документ  по программе, заданной в файле  "имя\_файла". |
| На рисунке представлена структура перехода по гиперссылке от одной странице к другой. | | |



Структура перехода по гиперссылке

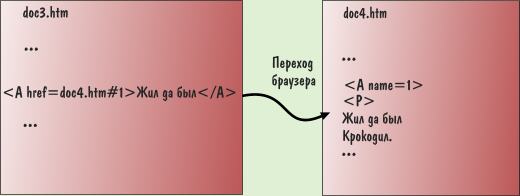
###### 

###### **Переход к метке другого документа**

Существует метод для объединения задач передачи управления в другую часть документа и передачи управления в другую программу (HTML-файл). Этот метод заключается в передачи управления из одного документа к метке внутри другого.

|  |  |
| --- | --- |
| Общий вид программы | Комментарий |
| Первый HTML-файл  ...  <A href=имя\_файла#метка>  Текст  </A>  ...  Второй HTML-файл  ...  <A name=метка>  ... | Приступить к показу фрагмента  с меткой "метка" в файле  "имя\_файла".  На экран выводится ссылка: [текст](../AppData/Local/Temp/Rar$DIa0.701/Robot_Htm/04.HTM#метка3#метка3)  При щелчке пользователя на ссылке  браузер строит на экране документ  по программе, заданной в файле  "имя\_файла", начиная с фрагмента  с меткой "метка" |

На рисунке представлена структура перехода к метке другого HTML документа.



Структура перехода к метке HTML-документа

***Иллюстрация как ссылка***

Возможен вариант, когда картинка будет играть роль гиперссылки. Для этого достаточно вложить команду IMG внутрь команды A:<A href=переход> <IMG src=файл> </A> Картинка ведет себя так же, как текст. Курсор при попадании на картинку-ссылку меняет свою форму. Мышиный щелчок по картинке заставляет браузер выполнять переход. Отличие только в том, что картинка-ссылка не подчеркивается, как текст, и не выделяется цветом, а обрамляется в рамочку.

***Вопросы для самоконтроля:***

1. Как делается переход внутри документа?

2. Как создать ссылку на отдельный документ?

3. Как сделать иллюстрацию ссылкой?

**Раздел 2. Веб-дизайн**

**Тема 2.1. Дополнительные возможности создания веб-страниц**

***Содержание программы***

Назначение CSS. Базовые понятия CSS. Виды стилей

Стиль - это набор параметров, задающий внешнее представление некоторого объекта в той или иной среде.

CSS - это сокращение от Cascading Style Sheets - в переводе Каскадные таблицы стилей.

Основные преимущества CSS :

* управление дизайном любого количества документов с помощью одной таблицы стилей;
* более точный дизайн страниц, поддерживаемый всеми браузерами;
* разделение документа на две составляющие: структура и дизайн, благодаря чему исходный код становится чистым и легко читаемым;
* новые расширенные возможности по сравнению с обычным html.

**Синтаксис и принцип работы CSS**

Как и любой другой язык программирования, CSS имеет строго определенный синтаксис, т.е. правила по которым создаются таблицы стилей. Запомните, в CSS в отличие от HTML нет ни элементов, ни атрибутов, ни тегов. Основной структурной единицей здесь является правило, которое определяет, как будет выглядеть тот или иной элемент в документе.

Рассмотрим структуру правила:



Как видно из рисунка выше, сначала записывается так называемый **селектор**, показывающий к какому html тегу(тегам) применяется то или иное свойство.

Далее, непосредственно за селектором, пишется **блок объявления стилей**, который обязательно заключается в фигурные скобки.

Каждое объявление в свою очередь состоит из **свойства** и его **значения**. После свойства ставится двоеточие. Правило может содержать в себе несколько объявлений. В таком случае они должны быть отделены друг от друга точкой с запятой (см. рисунок) причем после последнего объявления точку с запятой можно не ставить.

Виды стилей:

**А) Встраиваемые стили**

Встраиваемые стили определяются в HTML-теге и оказывают влияние только на этот тег.

Пример применения стиля к абзацу

<html><head><title>ПРИМЕР ВСТРАИВАЕМОГО СТИЛЯ</title></head>

<body>

<p style="font: 17pt comic sans ms">Текст этого абзаца будет отображаться шрифтом Comic

sans ms размером 17 пунктов</p>

</body></html>

**Б) Подключаемые стили**

Подключаемые стили, которые также называют внутренними стилями, применяются ко всей веб-странице. Для этого служит тег <STYLE>, помещаемый в заголовок HTML-документа.

**<html>  
<head>  
<title>**Работаем со стилями**</title>**  
<style type="text/css">  
h1{color:blue;font-size:14px}  
</style>  
**</head>  
<body>**  
**<h1>**Это моя первая таблица стилей, и если все работает, то несмотря на то, что это заголовок первого уровня, он отобразиться высотой всего лишь 14 пикселей и будет голубого цвета**</h1>**  
**</body>  
</html>**

**В) Связываемые стили**

Связываемые, или внешние, стили применяются к документам всего веб-сайта. Создайте с помощью любого текстового редактора документ, содержащий только определения стилей, и сохраните его в файле с расширением .css. Созданные таким образом списки можно применить к любой веб-странице.

Тогда в каждом документе, который мы хотим подключить, в голове документа (между тегами <head> и </head>) необходимо прописать строчку:

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

Эта строка указывает браузеру, что он должен использовать правила отображения HTML-файла из CSS-файла.

**Попробуйте сами:**

Откройте Блокнот (или другой редактор) и создайте с помощью него два файла находящихся в одной папке: CSS файл - **style.css** и HTML файл - **index.htm**. Файлы примерно следующего содержания:

**index.htm**

<html>  
<head>  
<title>Работаем со стилями</title>  
<link rel="stylesheet" type="text/css" href="style.css" >  
</head>  
<body>  
<h1>Это моя первая таблица стилей, и если все работает, то несмотря на то, что это заголовок первого уровня, он отобразится высотой всего лишь 14 пикселей и будет голубого цвета</h1>  
</body>  
</html>

**style.css**

h1{color:blue;font-size:14px}

***Вопросы для самоконтроля:***

1. Что такая CSS?

2. Что показывает селектор?

3. Перечислите виды стилей.

**Тема 2.2. Свойства CSS**

***Содержание программы***

Разъяснение значений свойств. Свойства шрифта.   
Свойства текста. Свойства цвета и фона.

***Методические указания***

**Шрифт в CSS**

За шрифты в CSS отвечают следующие свойства:

* [font-family](http://zvirec.com/css_book.php?id=5#family)
* [font-style](http://zvirec.com/css_book.php?id=5#style)
* [font-variant](http://zvirec.com/css_book.php?id=5#variant)
* [font-weight](http://zvirec.com/css_book.php?id=5#weihgt)
* [font-size](http://zvirec.com/css_book.php?id=5#size)
* [font](http://zvirec.com/css_book.php?id=5#font)

**Свойство FONT-FAMILY**

Данное свойство определяет гарнитуру шрифта. Вообще FONT-FAMILY с англ. означает семейство шрифта. Шрифты по тем или иным признакам, объединяются в семейства. Три основных семейства:   
**Serif** - шрифты с характерными "засечками", наиболее яркий представитель - "Times New Roman";  
**Sans-serif** - шрифты рубленые, без засечек, например Arial или Verdana ;  
**Monospace** - моноширинные шрифты (с одинаковым расстоянием между символами, наподобие печатной машинки), такие как "Courier New";

 Если в названии шрифта встречаются пробелы, то его следует заключать в кавычки" .

**h1** {font-family: verdana, arial, sans-serif;}  
**p** {font-family: "Times New Roman", serif;}

Т.е. в начале пишется самый приоритетный шрифт, затем менее приоритетный, а затем желательно писать имя семейства.   
**h1** {**font-family**: **verdana, arial, sans-serif**;} означает, что все заголовки первого уровня будут отображаться шрифтом **verdana**. Если по каким либо причинам этот шрифт не установлен на компьютере, то браузер подставит шрифт **arial**, а если и его нет, то браузер подставит любой доступный шрифт из семейства **serif** .

**Свойство FONT-STYLE**

Данное свойство задает стиль шрифта. Может принимать три значения:

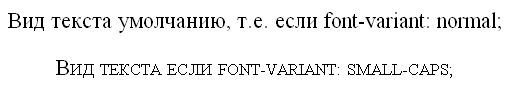
* **normal** - обычный;
* **italic** - курсивный;
* **oblique** - наклонный;

**h1**{font-family: verdana, arial, sans-serif;**font-style**:**normal**;}  
**h2**{font-family: verdana, arial, sans-serif; **font-style**:**italic**;}  
**h3**{font-family: verdana, arial, sans-serif;**font-style**:**oblique**;}

**Свойство FONT-VARIANT**

Это свойство используется для выбора варианта написания букв нижнего регистра. Может принимать два значения:

**normal** - значение по умолчанию, текст отображается обычным образом.  
**small-caps** - буквы нижнего регистра отображаются как уменьшенные заглавные.



Т.е. как видите большие(заглавные буквы) остаются без изменений, а маленькие - представляют собой полную копию заглавных букв, только слегка меньшего размера.

По умолчанию (т.е. если вообще не записывать это свойство) текст будет отображаться обычным начертанием.

Пример записи стиля:

**h1**{font-family: verdana, arial, sans-serif; **font-variant**:**small-caps**; }  
**h2**{font-family: verdana, arial, sans-serif;}

**Свойство FONT-WEIGHT**

Это свойство определяет насыщенность шрифта, т.е. с его помощью можно сделать шрифт жирным. Основные значения: **normal** - обычный и**bold** - жирный. Некоторые браузеры поддерживают числовые значения: 100, 200,300,400,500,600,700,800,900. Для справки: 400 равносильно normal, а 700 - bold . Но не советую вам пользоваться числами!

**P** {font-family: arial, verdana, sans-serif;}

**Свойство FONT-SIZE**

C его помощью можно регулировать размер шрифта. В качестве единиц измерения лучше всего использовать пикселы, т.к. это универсальный способ и во всех браузерах вы увидите одинаковый результат, а это самое главное.

Давайте прикольнемся над заголовками:

**h1** {font-family: arial, verdana, sans-serif;**font-size**: **18px**;}  
**h2**{font-family: arial, verdana, sans-serif;**font-size**: **36px**;color: red; }

# Текст в CSS

Свойства CSS отвечающие за форматирование текста.

* [text-align](http://zvirec.com/css_book.php?id=6#align)
* [text-decoration](http://zvirec.com/css_book.php?id=6#decor)
* [text-indent](http://zvirec.com/css_book.php?id=6#indent)
* [text-transform](http://zvirec.com/css_book.php?id=6#transform)
* [letter-spacing](http://zvirec.com/css_book.php?id=6#letter)
* [word-spacing](http://zvirec.com/css_book.php?id=6#word)

**Свойство TEXT-ALIGN**

Свойство выравнивания текста, аналогичное атрибуту ALIGN используемому в html. Может принимать четыре значения:

**left** - выравнивание по левому краю(значение по умолчанию);  
**right** - выравнивание по правому краю;  
**center** - выравнивание по центру;  
**justify** - выравнивание по ширине(по правому и левому краям одновременно).

Тут все очень просто, ну вот например:

**h1**{ **text-align**:**center**;}  
**h2** {**text-align**:**left**;}  
**h3** {**text-align**:**right**;}  
**p** {**text-align**:**justify**;}

**Свойство TEXT- DECORATION**

Позволяет оформлять текст определенным образом. Рассмотрим четыре основных значения данного свойства:

**none** - значение по умолчанию. Дополнительного оформления не происходит;  
**underline**- текст подчеркивается снизу;  
**overline**- текст надчеркивается сверху;  
**line-through** - текст перечеркивается;

**h1** {text-align:center; **text-decoration**:**underline**;}  
**h2** {text-align:center;**text-decoration**:**overline**;}  
**h3** {text-align:center; **text-decoration**:**line-through**;}

**Свойство TEXT-INDENT**

Это свойство пригодится нам для создания отступов первой строки, другими словами абзацев. Значение лучше задавать в пикселах, это универсальный способ.

**h1**{text-align:center;} **p**{**text-indent**: **40px**;}

Однако можно задать и в процентах от общей длины базовой строки(строки без отступа)

**h1**{text-align:center;} **p**{**text-indent**: **20%**;}

**Свойство TEXT-TRANSFORM**

C помощью этого свойства, можно видоизменять текст, а именно менять большие буквы на маленькие или наоборот. Может иметь такие значения:

**capitalize** - меняет регистр первых букв каждого слова так, чтобы они были заглавными.**Например:"создайте сайт сейчас" станет "Создайте Сайт Сейчас"**.  
Частенько это свойство используется при написании рекламных текстов, это больше привлекает внимания!

**uppercase**- делает из всех букв заглавные. **Например: "текст в css " станет " ТЕКСТ В CSS "**

**lowercase**- делает из всех букв маленькие. **Например: "TRANSFORM" cтанет "transform".**

**none** - не производит смены регистра; это значение используется по умолчанию.

**Свойство LETTER-SPACING**

C помощью этого свойства можно изменять расстояние между буквами. Значение лучше указывать в пикселях.

**h1** {**letter-spacing**: **10px**;}  
**p**{**letter-spacing** :**4px**; }

В примере выше для заголовков установлен интервал между буквами в 10 px, а для параграфов в 4px;

**Свойство WORD-SPACING**

Позволяет изменять расстояние между словами. Значение также лучше задавать в пикселах.

**h1** {**word-spacing**: **20px**;}  
**p**{**word-spacing** :**10px**; }

**Цвет и фон в CSS**

Цвета в CSS указываются так же, как и в html. Т.е. можно указывать шестнадцатиричное значение, например #ff3355, либо же название цвета (red, green, blue и др.)

Основными свойствами цвета и фона в CSS являются:

* [color](http://zvirec.com/css_book.php?id=4#color)
* [background-color](http://zvirec.com/css_book.php?id=4#backgroundcolor)
* [background-image](http://zvirec.com/css_book.php?id=4#backgroundimage)

**Свойство COLOR**

Задает основной цвет(цвет переднего плана) того или иного элемента. Например, если мы хотим сделать цвет всех заголовков первого уровня красным, а цвет текста параграфов зеленым, то таблица стилей будет выглядеть так:

# H1 {color: red ;} P {color: green ;}

**Свойство BACKGROUND-COLOR**

Задает фоновый цвет элемента. В отличие от html, в котором фоновый цвет можно использовать только для страницы или ячейки таблицы (т.е. имеющих атрибут bgcolor ) в CSS , фоновый цвет можно задавать для чего угодно: для таблиц, заголовков, параграфов, ссылок и др. Тут главное правильно определить что нам нужно.

Ну вот например, чтобы изменить фоновый цвет всей страницы, нужно задать это свойство элементу BODY - т.к. именно он отвечает за тело документа, т.е. за всю страницу.

##### **BODY** {background-color : #FFEE8C **;**} **H1** {**color: red ;**background-color **:**blue **;**} **P** {**color: green ;**}

**Свойство BACKGROUND-IMAGE**

Данное свойство используется для задания фонового изображения

##### **BODY** {**background-color : #FFEE8C ;**background-image **:** url(smile.png) **;**} **H1** {**color: red ;background-color :blue ;**} **P** {**color: green ;**}

Как видите, в качестве значения свойства, указывается путь к изображению, но немного не так как в html. В начале пишется **URL** а затем сразу, **без пробелов**в круглых скобках положение картинки. Если она находится в той же папке, то пишем просто название картинки, как в примере выше. Если допустим в подпапке img , то пишем так url(img/smile.png).

Фоновое изображение в CSS можно задавать для любого элемента, не только для таблиц и всей страницы.

***Вопросы для самоконтроля:***

1. Назовите основные свойства шрифта.

2. Назовите основные свойства текста

3. Назовите основные свойства цвета и фона.

**Тема 2.3. Модель форматирования CSS**

***Содержание программы***

Поля и отступы. Блочные элементы. Плавающие элементы.

Поля и отступы

Поле (**margin**) - Это расстояние от внешней границы элемента до границы окна браузера или же элемента родителя, т.е. границы того блока в который вложен наш элемент.

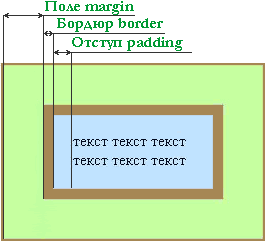
Отступом (**padding**) - Называют расстояние от внутренней границы элемента до его содержания текста, картинок таблиц.

На рисунке наглядно показаны эти расстояния, а так же место занимаемое бордюром (**border**) про него тоже не следует забывать.

Расстояния **margin** и **padding** указываются:

* **px** - В пикселях или любых других допустимых CSS единицах измерения.
* **%** - В процентах.
* **auto** - Размер полей и отступов автоматически рассчитывается браузером

**margin: 5px 40px 20px 1px;** -*четыре значения.*  
Первое значение - Устанавливает поле от верхней границы элемента.   
Второе значение - Устанавливает поле от правой границы элемента.   
Третье значение - Устанавливает поле от нижней границы элемента.   
Четвёртое значение - Устанавливает поле от левой границы элемента.



## Дочерние свойства margin и padding.

Свойства CSS **margin** и **padding** являются базовыми и имеют ряд дочерних свойств.

У **margin** зто:

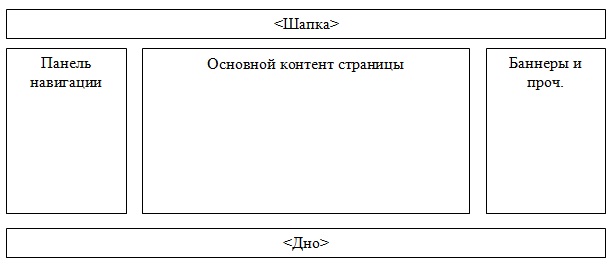
* **margin-top** - Поле от верхней границы элемента родителя,
* **margin-left**- Поле от левой границы элемента родителя,
* **margin-right** - Поле от правой границы элемента родителя,
* **margin-bottom** - Поле от нижней границы элемента родителя.

И у **padding** соответственно:

* **padding-top** - Отступ от верхней границы элемента до его содержания,
* **padding-left** - Отступ от левой границы элемента до его содержания,
* **padding-right** - Отступ от правой границы элемента до его содержания,
* **padding-bottom** - Отступ от нижней границы элемента до его содержания.

***Блочные и плавающие элементы***

Как и любой *блочный элемент*, *контент*  тега <div> начнется с новой строки. Содержимое нескольких *тегов* <div> будет расположено вертикально, друг под другом. Допустим, что мы хотим получить веб - страницу следующего вида:



Добавим следующий *html* - код для размещения пяти контейеров:

<div id="top">top</div>

<div id="left">left </div>

<div id="content">content</div>

<div id="right">right</div>

<div id="bottom">bottom</div>

В *таблицу стилей* добавим соответствующие стили для каждого контейнера:

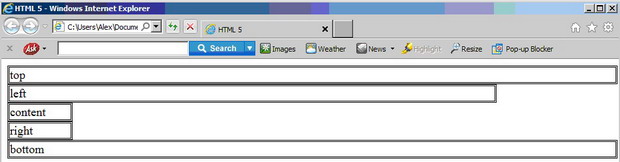
#top { height: 20px; width:412px; border: 3px double black}

#left { height: 200px; width:50px; border: 3px double black; }

#content { height: 200px; width:300px; border: 3px double black; }

#bottom { height: 20px; width:412px; border: 3px double black; }

Для наглядности, для каждого контейнера указаны параметры отрисовки его границ. В результате получим следующее:



Очевидно, что *размещение* контейнеров, используемое *по* умолчанию, не подходит для наших целей.

Для управления размещением элементов используется *атрибут* стиля *float*, принимающий следующие значения:

* left – элемент выравнивается по левому краю родителя, остальные элементы "обтекают" указанный по правой стороне;
* right – элемент выравнивается по правому краю родителя, остальные элементы "обтекают" указанный по левой стороне;
* none – обтекание элемента не указано;
* *inherit* – значение наследуется от родителя.

Для того чтобы элементы расположились *по* горизонтали один за другим, необходимо задать одно и то же *значение* *float* для следующих друг за другом элементов.

Для размещения блоков ниже выровненных *по* горизонтали, необходимо использовать *атрибут* стиля *clear*, явно указывающий на то, что данный блок должен располагаться ниже предшествующих ему контейнеров.

*Атрибут* *clear* может принимать следующие значения:

* left – элемент будет расположен ниже всех элементов, значение атрибута *float* у которых равно left;
* right – элемент будет расположен ниже всех элементов, значение атрибута *float* у которых равно right;
* both – элемент будет расположен ниже всех элементов, значение атрибута *float* у которых равно left или right;
* none – отмена свойств атрибута *clear*;
* *inherit* –значение наследуется от родителя.

Изменим стили контейнеров из предыдущего примера следующим образом:

#top { height: 20px; width:412px; border: 3px double black}

#left { height: 200px; width:50px; border: 3px double black; float: left}

#content { height: 200px; width:300px; border: 3px double black; float:left}

#right { height: 200px; width:50px; border: 3px double black; float:left}

#bottom { height: 20px; width:412px; border: 3px double black; clear: left}

Этого достаточно для того, чтобы получить желаемый результат.

***Вопросы для самоконтроля:***

1. Назовите основные свойства полей.

2. Назовите основные свойства отступов.

3 Назовите тег блочного элемента и его свойства.