# ВЕБ-ТЕХНОЛОГИИ

## Методические указания к выполнению лабораторных работ

Для студентов специальности 1-40 01 01 Программное обеспечение информационных технологий

# Лабораторная работа №1.

# Тема: Разработка макета Web-сайта средстами HTML

## Основные понятия

**HTML** *(HyperText Markup Language)* - язык разметки гипертекста - предназначен для создания Web-страниц. Под ***гипертекстом*** в этом случае понимается текст, связанный с другими текстами указателями-ссылками.

HTML представляет собой достаточно простой набор кодов, которые описывают структуру документа. HTML позволяет выделить в тексте отдельные логические части (заголовки, абзацы, списки и т.д.), поместить на Web-страницу подготовленную фотографию или картинку, организовать на странице ссылки для связи с другими документами.

HTML не задает конкретные и точные атрибуты форматирования документа. Конкретный вид документа окончательно определяет только ***программа-броузер*** на компьютере пользователя Интернета. HTML также не является языком программирования, но web-страницы могут включать в себя встроенные программы-скрипты на языках ***Javascript*** и ***Visual Basic Script*** и программы-апплеты на языке ***Java***.

Даже, если вы не собираетесь в дальнейшем редактировать "вручную" текст HTML (предполагая использовать графические редакторы), знание языка HTML даст вам возможность как лучше использовать эти средства, так и увеличит ваши шансы сделать HTML-документ более доступным и "читаемым" при просмотре броузерами разных фирм.

Основными компонентами HTML являются:

- **Тег (tag).** Тег HTML это компонент, который командует Web- броузеру выполнить определенную задачу типа создания абзаца или вставки изображения.
- **Атрибут** (или **аргумент**). Атрибут HTML изменяет тег. Например, можно выровнять абзац или изображение внутри тега.
- **Значение**. Значения присваиваются атрибутам и определяют вносимые изменения. Например, если для тега используется атрибут выравнивания, то можно указать значение этого атрибута. Значения могут быть текстовыми, типа *left* или *right*, а также числовыми, как например ширина и высота изображения, где значения определяют размер изображения в пикселях.

Теги представляют собой зарезервированные последовательности символов, начинающиеся с < (знака меньше) и заканчивающиеся > (знаком больше). Закрытие тега отличается от открытия только наличием символа '/'.

Предположим, у нас есть гипотетический атрибут форматирования текста, управляемый кодом **<X>**, и мы хотим применить его к словам "Это мой текст". HTML-последовательность кодов и собственно текста будет выглядеть так:

```
<X>Это мой текст</X>
```

Теги могут вкладываться друг в друга иерархически, но без пересечений, то есть допустимо вложение вида **<*teg1*><teg2></teg2> <*/teg1*>**, но не **<*teg1*><teg2> <*/teg1*></teg2>**.

Действие вложенных тегов объединяется. Например, если внутрь тега, создающего жирное начертание шрифта, вложен тег курсива, то в результате получится жирный курсив.

**Первое правило**

Первое правило HTML: закрывайте все, что вы открыли!
НО! Из этого правила, как и из всех остальных, существуют исключения.

HTML- программа должна начинаться тегом **<HTML>** и заканчиваться тегом **</HTML>**

```
<HTML>
          ......     (здесь будут другие теги программы)
  </HTML>
```

HTML- программы состоят из двух основных частей: заголовка и тела. Заголовок ограничивается парой тегов**<HEAD>** и **</HEAD>**, а тело - парой тегов**<BODY>** и **</BODY>**.

В результате HTML- программа выглядит следующим образом:

```
<HTML>
       <HEAD>
              ...   (здесь будет заголовок)
        </HEAD>
        <BODY>
           ....     (здесь будут другие теги тела
программы)
        </BODY>
</HTML>
```

Кроме того, каждая HTML- программа имеет заголовок, который помещается в заголовок окна броузера. Заголовок окна броузера создается при помощи двух тегов **<TITLE>** и **</TITLE>** и содержится между тегами **<HEAD>** и**</HEAD>**.

Тогда программа принимает следующий вид:

```
<HTML>
            <HEAD>
                <TITLE> Основы HTML </TITLE> ;
               </HEAD>
             <BODY>
```

```
                ....    (здесь будут другие теги тела
программы)
              </BODY>
</HTML>
```

---

***Некоторые авторы, пишущие об языке HTML, советуют записывать теги прописными буквами, другие - используют строчные. Редактор HTML - Allaire HomeSite 4.5.1, например, использует по умолчанию нижний регистр для записи тегов. При создании моих страниц использовались оба варианта написания тегов. Как видите, допустимо и то и другое. Современные браузеры допускают запись тегов в любом регистре.***
***Существуют теги и атрибуты "чувствительные" к написанию прописными или строчными буквами. Это регламентируется стандартами языка HTML, определенными Консорциумом W3C.***
***Обращайтесь к первоисточнику! Хорошее знание технического английского обязательно!***

---

При написании HTML-программ возникает необходимость вставки ***комментариев*** - поясняющих текстов, которые невидны при загрузке документа в браузер. Для этой цели служит тег **<!>**. Все, что заключено между символами **<!** и > считается комментарием и не отображается в браузере.

---

Еще один тег, который очень важен в HTML-программе, но так же не предназначается для отображения какого-либо объекта в броузере - тег **<*META*>**. Этот тег служит специальным целям, а именно - указания языка, на котором написан документ, его кодовой страницы, ключевых слов, используемых поисковыми системами для классификации этого документа и т.п. Теги **<*META*>** обычно вставляются в HTML-программу на заключительном этапе создания Web-страницы -*публикации*.

---

Для вставки в HTML-программу фрагмента программ, написанных на языке JavaScript или Viual Basic Script *сценариев* используют теги **<*SCRIPT*>** и **</*SCRIPT*>**.

---

## Общая структура HTML-файла

Суммируя вышесказанное приведем общую структуру HTML-файл :

```
<HTML>
<HEAD>
<Мета-теги>
<Функции скриптов>
<TITLE>Заголовок документа</TITLE>
</HEAD>
<BODY>
Основная часть документа
</BODY>
</HTML>
```

# Заголовки, абзацы, разрывы строк

## Заголовки

Каждый пользователь компьютера, работающий в текстовом редактором Microsoft Word знаком с понятием ***стиля заголовка***. В HTML тоже применяется это понятие для структурирования документа и выделения важности заголовка. Всего существуют 6 стилей заголовка. Каждый из них обозначается в HTML-документе парными тегами **<*Hi*>** и **</*Hi*>.** Здесь ***i*** обозначает важность стиля. ***H1*** обозначает самый важный стиль заголовка, ***H2*** - стиль заголовка второго уровня, а ***H6*** - стиль заголовка самого нижнего уровня.

В подавляющем большинстве случаев для заголовков Web-страниц используют три первых уровня заголовков **<*H1*>**, **<*H2*>** и **<*H3*>**. Объясняется это тем, что размеры шрифтов оставшихся заголовуов (теги **<*H4*>** - **<*H5*>**) меньше размера обычного шрифта Web-страницы.

Вот как в документ можно добавить очень важный заголовок.

```
<H1>An important heading</H1>
```

А вот результат.

# An important heading

Посмотрим другие примеры:

## Это заголовок второго уровня

### Это заголовок третьего уровня

Это обычный текст

### Это заголовок четвертого уровня

#### Это заголовок пятого уровня

##### Это заголовок шестого уровня

## Абзацы

Понятие абзаца в HTML-документе также аналогично понятию абзаца в Microsoft Word. Абзац обозначается в документе парными тегами **<*P*>** и **</*P*>** . Впрочем, применение закрывающего тега не является строго обязательным. НО! Специфика тега **<*P*>** заключается в том, что после текста, который

находится в его пределах, пустая строка добавляется ***автоматически***. Следует помнить и о другой особенности текстовых абзацев: когда текст достигает правой границы окна Web-броузера. переход на новую строку осуществляется автоматически, независимо от расположения тега **<*P*>**. Для отдельного абзаца можно указать тип, размер и цвет шрифта отличным от стиля остального документа.

Например:

```
<P ALIGN="CENTER"><B><I><FONT FACE="Arial" SIZE=6
COLOR="#C07080">My greetings to you!</P>
```

А вот и результат -

# *My greetings to you!*

**Разрывы строк**

Если в середине строки появилась необходимость ее разорвать - используйте **одиночный** тег переноса строки **<*BR*>**.(Это соответствует нажатию клавишной комбинации [Shift]-[Enter] в текстовых процессорах Word). Код <BR> не означает конца логического абзаца, и за строкой с этим кодом дополнительная пустая строка не появится.

Вот и нарушено первое правило!

Кроме тега **<*BR*>** не требует закрытия тег добавления изображения ***<IMG>***.

Использование закрывающего тега абзаца **</*p*>** также не является строго обязательным.

***HTML довольно "демократичен": неправильный тег или неправильное вложение тегов обычно не приводят к "зависаниям" браузера, а только вызывает сообщение об ошибке в строке состояния окна браузера вашего Интернет-читателя. Разумеется ошибки могут привести к неправильному форматированию HTML-документа.***

Примером использования тега **<*BR*>** может служить написание почтового адреса или стихотворения.

Браузеры показывают текст в своем окне, автоматически осуществляя перенос слов. Поэтому, если вы считаете необходимым запретить разрыв блока текста с пробелами между словами - воспользуйтесь специализированным символом *** *** - символом неразрывного пространства (non-breaking space). Например,

```
Освежающий  и бодрящий напиток Coca Cola,
приобрел широкую популярность в нашей стране.
```

А вот и результат:

Освежающий и бодрящий напиток Coca Cola, приобрел широкую популярность в нашей стране.

---

*Совет*
Не следует использовать цепочку символов *** *** для выравнивания текста в окне браузера. Для этих целей рекомендуется использовать *таблицы стилей*.

---

### Предварительно отформатированный текст

Достоинством браузеров является их способность самостоятельно распределять текст в окне браузера. Но иногда вы не нуждаетесь в этой услуге, хотите самостоятельно определить представление вашего текста в окне браузера. Например, вы хотите представить код программы в наиболее удобочитаемом виде. Такую возможность вам предоставляет тег ***<pre>***. Посмотрим пример:

```
<pre>
    void Node::Remove()
    {
        if (prev)
            prev->next = next;
        else if (parent)
            parent->SetContent(null);

        if (next)
            next->prev = prev;

        parent = null;
    }
</pre>
```

Это выполнится так

```
    void Node::Remove()
    {
        if (prev)
            prev->next = next;
        else if (parent)
            parent->SetContent(null);

        if (next)
```

```
            next->prev = prev;

        parent = null;
    }
```

Это сделано при помощи *таблицы стилей*.

Обратите внимание на то, что в случае использования тега **<pre>** , текст отображается браузером точно в таком виде, как он был создан в HTML-документе. Сохраняются все пробелы, табуляции и переводы строк. Исключением является только новая строка, следующая сразу же за тегом **<pre>**. Таким образом, эти два примера кода HTML на экране дисплея будут показаны одинаково:

```
<pre>предварительно отформатированный текст</pre>

<pre>
предварительно отформатированный текст
</pre>
```

А именно:

```
предварительно отформатированный текст
предварительно отформатированный текст
```

У этого тега существует необязательный атрибут, указывающий желаемый размер строки в символах, а именно:

```
<PRE width="N">
Предмет истории есть жизнь народов и человечества.
Непосредственно уловить и обнять словом - описать жизнь
не только
человечества, но одного народа, представляется
невозможным.
```

Обычно, для отображения предварительно отформатированного текса используются моноширинные шрифты, все символы которого имеют одинаковую ширину. Для того, чтобы браузер "не забыл" об этом следует применить *таблицы стилей*. Например, это можно сделать так:

```
<style type="text/css">
  pre { color: green; background: white; font-family:
monospace; }
</style>
```

Результат такого определения стиля для тега <pre> вы можете видеть на этой странице.

---

*Совет*

Если вы устанавливаете цвет текста, желательно также установить цвет фона. Это поможет вам избежать ситуации, когда буквы будут практически неразличимы на близком к ним по цвету фоне.

---

Совет № 2. Вместо того, чтобы устанавливать цвет фона для элемента <pre>, установите цвет фона для элемента <body>. Например, это можно сделать так:

```
<style type="text/css">
  body { color: black; background: white; }
  pre { color: green; font-family: monospace; }
</style>
```

**Выделенный текст**

В тех случаях, когда необходимо обратить особое внимание пользователя на тот или иной фрагмент текста, следует использовать стандартные средства форматирования.

Стандартные средства форматирования представлены специальными тегами, которые ***обязательно*** являются ***парными*** (т.е. имеются открывающий и закрывающий теги).

Чтобы увеличить размер шрифта на один пункт, используйте тег <big>.

Чтобы выделить текст **полужирным шрифтом**, воспользуйтесь **тегом <b>** или **тегом **.

Чтобы выделить текст *курсивом*, воспользуйтесь *тегом <i>* или ***тегом ***.

***Вы заметили, что в данном случае выделение текста тегами <i> и  различаются?***

***Объяснение простое - в HTML-коде этой страницы используется внедренная таблица стилей, в которой для тега  указан шрифт: полужирный курсив.***

`Шрифт пишущей машинки` можно имитировать с помощью `тега <tt>`.

С помощью тега <small> размер шрифта можно уменьшить на один пункт.

Существуют и другие теги, которые также предназначены для выделения текста.

Кроме того, в HTML включена поддержка математических символов и научных обозначений. Для построения простейших равенств и выражений вам могут пригодиться два тега **<*sub*>** (нижний индекс) и **<*sup*>** (верхний индекс). Например:

$$A^2+B^2=C^2$$

$CO_2$=углекислый газ

В HTML-коде это записано так:

```
<p align="center">A<sup>2</sup>+B<sup>2</sup>
=C<sup>2</sup></p>
<p align="center">CO<sub>2</sub>=углекислый газ</p>
```

**Добавление изображений**

Изображения делают страницу привлекательной. Теги изображений представляет собой исключение из правил - для них них не требуется закрывающего тега, зато у них есть ряд важных атрибутов. Теги изображения состоят из тега ***<img>*** и источника изображения (адрес графического файла), указанного в атрибуте ***src***. В качестве адреса графического файла указывается либо URL-адрес, либо имя файла (возможно с указанием пути). Например,

```
<IMG SRC="images/"Tourgojk_lake.jpg>
```

Рекомендуется включать в тег следующие атрибуты для каждого изображения:

- **width="x".** Определяет ширину изображения
- **heigth="x".** Определяет высоту изображения
- **border="x".** Значение *x* в данном случае определяет толщину рамки, которую браузер будет рисовать вокруг изображения.

---

*Совет*
Для этого атрибута нужно использовать значение,не равное нулю.

---

- **alt="x".** Атрибут *alt* позволяет вводить описание изображения. Это очень полезно для людей, отключающих изображение.

**Увеличение скорости загрузки изображения**

Для увеличения скорости передачи графического изображения в теге <IMG> можно использовать атрибут *lowsrc*, который принимает в качестве аргумента адрес графического файла. Для использования этого атрибута необходимо создать два файла: один с высоким разрешением (формат JPG), другой - с низким (формат GIF). Тогда тег

**<img src = "images/big_flower.jpg" align="middle"**
**lowsrc="images/small_flower.gif">**

предпишет браузеру сначала загрузить файл small_flower.gif, а затем по мере приема заменить его файлом big_flower.jpg.

Другой способ ускорения загрузки заключается в задании размеров изображения с помощью атрибутов *width* (ширина) и *heigth* (высота), измеряемые в пикселах. Если указать эти атрибуты, то броузер сначала выделит место под графику, подготовит макет документа, отобразит текст, и

только потом загрузит графику. При этом браузер сжимает или растягивает изображение, встраивая его в рамки указанного размера.

## Обтекание текстом изображения

При создании HTML-документа, вы можете выбрать как используемое вами изображение будет расположено относительно текста, а именно:

1. в одну линию с текстом;
2. слева от текста;
3. справа от текста.

Для управления расположением изображения относительно текста используется аттрибут *align*. Этот атрибут может принимать значения: *left* - изображение выравнивается по левому полю, *right* - изображение выравнивается по правому полю, *middle* - посередине.

**Задание №1.** Создайте web-сайт (3-4 связанные между собой html-страницы), согласно требованиям представленным в таблице 1. Наполните страницу небольшим контентом, согласно тематике.

Таблица 1 – Варианты для выполнения задания №1.

| **Вариант** | **Тематика страницы** |
|---|---|
| 1 | Спорт |
| 2 | Автомобили |
| 3 | Электроника |
| 4 | Программирование |
| 5 | Развлечения |
| 6 | Наука |
| 7 | Недвижимость |
| 8 | Книги |
| 9 | Животные |
| 10 | Игры |

**Задание №2.**

1) Создайте документ содержащий HTML-словарь тегов форматирования приведенных ниже. При создании словаря используйте список определений.

<abbr><acronym><address>**<article><aside>**
<b><bdi><bdo><big><blockquote><br>
<cite><code>**<command>**
**<datalist>**<del>**<details>**<dfn>

**<figcaption><figure><footer>**

<h1><h2><h3><h4><h5><h6>**<header>**
<i><ins>
<kbd>**<keygen>**
**<main><mark>**<menu>**<meter>**
**<nav>**<nobr>
**<output>**
<pre>**<progress>**<q>
<s><samp><small>**<source>**<strike><sub>**<summary>**<sup>
**<time><track>**<tt>
<u>
<var><video>
**<wbr>**

Жирным выделены теги поддержка которых есть только в HTML5. Создайте 5 страниц с подробным описанием тегов (синткаксис, поддержка браузеров, их возможные атрибуты, пример использования) согласно вашему варианту.

| **Вариант** | **Теги** |
|---|---|
| 1 | abbr code **mark time** video |
| 2 | acronym dfn **keygen progress** small |
| 3 | address blockquote **main summary** var |
| 4 | big **command header** ins samp |
| 5 | **datalist footer** nobr strike var |
| 6 | code **figcaption meter** pre sup |
| 7 | **figure** kbd  nobr small **wbr** |
| 8 | cite **header** menu strong var |
| 9 | big del **time track** var |
| 10 | div i **output progress** q |

2) В верхней части документа создайте внутренние ссылки (в качестве ссылки выступает заглавная буква английского алфавита), которые ведут в соответствующие букве части словаря. Добавьте возможность вернуться в начало документа.

# Лабораторная работа №2.

# Тема: Каскадные таблицы стилей CSS

## Теоретический материал

### 1. Что такое CSS?

CSS это акроним для Cascading Style Sheets/Каскадных таблиц стилей.

CSS это язык стилей, определяющий отображение HTML-документов. Например, CSS работает с шрифтами, цветом, полями, строками, высотой, шириной, фоновыми изображениями, позиционированием элементов и многими другими вещами. Потерпите, и увидите! HTML может (неправильно) использоваться для оформления web-сайтов. Но CSS предоставляет бóльшие возможности и более точен и проработан. CSS, на сегодняшний день, поддерживается всеми браузерами (программами просмотра).

В чём разница между CSS и HTML? HTML используется для структурирования содержимого страницы. CSS используется для форматирования этого структурированного содержимого.

По мере развития Web дизайнеры начали искать возможности форматирования онлайновых документов. Чтобы удовлетворить возросшим требованиям потребителей, производители браузеров (тогда - Netscape и Microsoft) изобрели новые HTML-тэги, такие, например, как <font>, которые отличались от оригинальных HTML-тэгов тем, что они определяли внешний вид, а не структуру.

Это также привело к тому, что оригинальные тэги структурирования, такие как <table>, стали всё больше применяться для дизайна страниц вместо структурирования текста. Многие новые тэги дизайна, такие как <blink>, поддерживались только одним браузером. "Вам необходим браузер X для просмотра этой страницы" - такой отказ стал обычным явлением на web-сайтах.

CSS был создан для исправления этой ситуации путём предоставления web-дизайнерам возможностей точного дизайна, поддерживаемых всеми браузерами. Одновременно произошло разделение представления и содержимого документа, что значительно упростило работу.

Появление CSS стало революцией в мире web-дизайна. Конкретные преимущества CSS:

- управление отображением множества документов с помощью одной таблицы стилей;
- более точный контроль над внешним видом страниц;
- различные представления для разных носителей информации (экран, печать, и т. д.);
- сложная и проработанная техника дизайна.

**2. Как работает CSS?**

Вы этой чсти вы создадите свою первую таблицу стилей/style sheet. Вы узнаете об основах базовой модели CSS и о том, какие коды необходимо использовать для CSS в HTML-документе.

Многие свойства, используемые в Cascading Style Sheets (CSS), аналогичны свойствам HTML. Таким образом, если вы используете HTML для дизайна страниц, вы, наверняка узна́ете многие коды. Посмотрим на конкретном примере.

***Базовый синтаксис CSS***

Скажем, нам нужен красный цвет фона web-страницы:

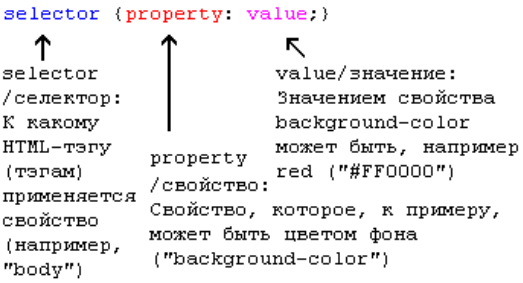
В HTML это можно сделать так:

```
<body bgcolor="#FF0000">
```

С помощью CSS того же самого результата можно добиться так:

```
body {background-color: #FF0000;}
```

Как видите, эти коды более или менее идентичны в HTML и CSS. Этот пример также демонстрирует фундаментальную модель CSS:




***Применение CSS к HTML-документу***

Есть три способа применить правила CSS к HTML-документу. Ниже мы рассмотри эти три метода. Мы рекомендуем сосредоточиться на третьем – то есть внешней/external таблице стилей.

*Метод 1: Инлайн/In-line (атрибут style)*

Можно применять CSS к HTML с помощью HTML-атрибута style. Красный цвет фона можно установить так:

```
<html>
          <head>
            <title>Example</title>
          </head>
          <body style="background-color: #FF0000;">
            <p>This is a red page</p>
          </body>
        </html>
```

*Метод 2: Внутренний (тэг style)*

Второй способ вставки CSS-кодов – HTML-тэг <style>. Например:

```
<html>
          <head>
            <title>Example</title>
            <style type="text/css">
              body {background-color: #FF0000;}
            </style>
          </head>
          <body>
            <p>This is a red page</p>
          </body>
        </html>
```

*Метод 3: Внешний (ссылка на таблицу стилей)*

Рекомендуемый метод – создание ссылки на так называемую внешнюю таблицу стилей. Внешняя таблица стилей - это просто текстовый файл с расширением .css. Вы можете поместить таблицу стилей на ваш web-сервер или на жёсткий диск, как и другие файлы.

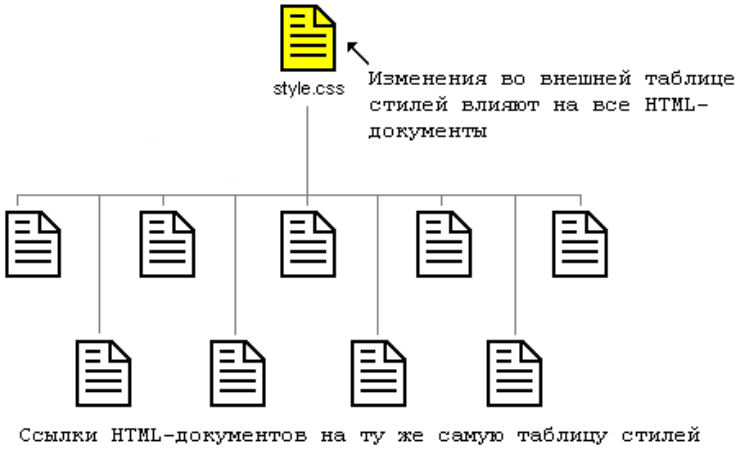
Например, скажем, ваша таблица стилей называется style.css и находится в папке style. Это можно проиллюстрировать так:


Весь фокус состоит в том, чтобы создать ссылку из HTML-документа (default.htm) на таблицу стилей (style.css). Это можно сделать одной строчкой HTML-кода:

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

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

Самое важное здесь то, что несколько HTML-документов могут ссылаться на одну таблицу стилей. Иначе говоря, один CSS-файл можно использовать для управления отображением множества HTML-документов.



Это поможет вам сэкономить уйму времени и сил. Если вы, например, хотите изменить цвет фона web-сайта из 100 страниц, таблица стилей избавит вас от необходимости вручную изменять все сто HTML-документов. Используя CSS, эти изменения можно сделать за несколько секунд, просто изменив один код в центральной таблице стилей.

**3. Цвет и фон**

В этой части вы научитесь, как использовать цвета и фон на ваших web-сайтах. Мы рассмотрим также продвинутые методы позиционирования и управления фоновым изображением. Будут разъяснены следующие CSS-свойства:

- color
- background-color
- background-image
- background-repeat
- background-attachment
- background-position
- background

***Цвет переднего плана : свойство 'color'***

Свойство color описывает цвет переднего плана элемента.

Например, представьте, что мы хотим сделать все заголовки документа тёмно-красными. Все заголовки обозначаются HTML-элементом <h1>. В нижеприведённом коде цвет элемента <h1> устанавливается красным.

```
h1 {
color: #ff0000;
}
```

Цвета можно указывать как шестнадцатеричные значения, как в примере (#ff0000), либо вы можете использовать названия цветов ("red") или rgb-значения (rgb(255,0,0)).

***Свойство 'background-color'***

Свойство background-color описывает цвет фона элемента.

В элементе <body> размещается всё содержимое HTML-документа. Таким образом, для изменения цвета фона всей страницы свойство background-color нужно применить к элементу <body>.

Вы можете также применять это свойство к другим элементам, в том числе – к заголовкам и тексту. В следующем примере различные цвета фона применяются к элементам <body> и <h1>.

```
body {
background-color: #FFCC66;
}

h1 {
color: #990000;
background-color: #FC9804;
}
```

Заметьте, что устанавливает два свойства для <h1>, разделяя их точкой с запятой.

***Фоновые изображения [background-image]***

CSS-свойство background-image используется для вставки фонового изображения.

Ниже мы используем в качестве фонового изображение бабочки. Вы можете загрузить это изображение и использовать его на вашем компьютере.

Для вставки рисунка бабочки в качестве фонового изображения web-страницы просто примените свойство background-image в тэге <body> и укажите местоположение рисунка.

```
body {
background-color: #FFCC66;
background-image: url("butterfly.gif");
}

h1 {
color: #990000;
background-color: #FC9804;
}
```

NB: Обратите внимание, что мы специфицируем место, где находится файл как url("butterfly.gif"). Это означает, что он находится в той же папке, что и таблица стилей. Вы, разумеется, можете ссылаться и на файлы изображений в других папках, используя, например, url("../images/butterfly.gif"), или даже на файлы в Internet, указывая полный адрес файла : url("http://www.html.net/butterfly.gif").

***Повторение/мультипликация фонового изображения [background-repeat]***

Вы заметили в предыдущем примере, что изображение бабочки повторяется по умолчанию по горизонтали и вертикали, заполняя весь экран? Свойство background-repeat управляет этим.

В таблице указаны четыре значения background-repeat.

| Значение | Описание |
|---|---|
| Background-repeat: repeat-x | Рисунок повторяется по горизонтали |
| background-repeat: repeat-y | Рисунок повторяется по вертикали |
| background-repeat: repeat | Рисунок повторяется по горизонтали и вертикали |
| background-repeat: no-repeat | Рисунок не повторяется |

Например, для отмены повторения/мультипликации фонового рисунка мы должны записать такой код:

```
body {
background-color: #FFCC66;
background-image: url("butterfly.gif");
background-repeat: no-repeat;
}

h1 {
color: #990000;
background-color: #FC9804;
}
```

***Блокировка фонового изображения [background-attachment]***

Свойство background-attachment определяет, фиксируется ли фоновый рисунок, или прокручивается вместе с содержимым страницы.

В таблице указаны два значения background-attachment. Попробуйте на примере почувствовать разницу между scroll и fixed.

| Значение | Описание |
|---|---|
| Background-attachment: scroll | Изображение прокручивается вместе со страницей - разблокировано |
| Background-attachment: fixed | Изображение блокировано |

Например, следующий код фиксирует изображение.

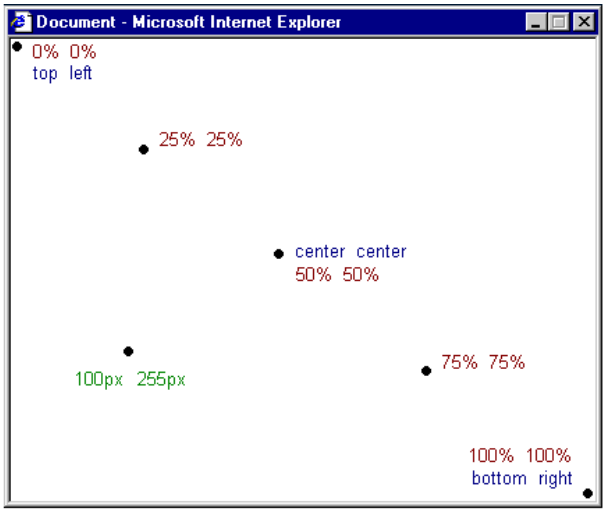
```
body {
background-color: #FFCC66;
background-image: url("butterfly.gif");
background-repeat: no-repeat;
background-attachment: fixed;
}

h1 {
color: #990000;
background-color: #FC9804;
}
```

***Расположение фонового рисунка [background-position]***

По умолчанию фоновый рисунок позиционируется в левом верхнем углу экрана. Свойство background-position позволяет изменять это значение по умолчанию, и фоновый рисунок может располагаться в любом месте экрана.

Есть много способов установить значение background-position. Тем не менее, все они представляют собой набор координат. Например, значение '100px 200px' располагает фоновый рисунок на 100px слева и на 200px сверху в окне браузера.

Координаты можно указывать в процентах ширины экрана, в фиксированных единицах (пикселы, сантиметры, и т. п.), либо вы можете использовать слова top, bottom, center, left и right. Модель ниже иллюстрирует сказанное:




В таблице дано несколько примеров.

| Значение | Описание |
|---|---|
| background-position: 2cm 2cm | Рисунок расположен на 2 cm слева и на 2 cm сверху |
| background-position: 50% 25% | Рисунок расположен по центру и на четверть экрана сверху |
| background-position: top right | Рисунок расположен в правом верхнем углу страницы |

В примере кода фоновое изображение располагается в правом нижнем углу экрана:

```
body {
background-color: #FFCC66;
background-image: url("butterfly.gif");
background-repeat: no-repeat;
background-attachment: fixed;
background-position: right bottom;
}

h1 {
color: #990000;
background-color: #FC9804;
}
```

### *Сокращённая запись [background]*

Свойство background входит в состав всех свойств, перечисленных в этой части.

С помощью background вы можете сжимать несколько свойств и записывать ваши стили в сокращённом виде, что облегчает чтение таблиц.

Например, посмотрите на эти строки:

```
background-color: #FFCC66;
background-image: url("butterfly.gif");
background-repeat: no-repeat;
background-attachment: fixed;
background-position: right bottom;
```

Используя background, того же результата можно достичь одной строкой кода:

```
background: #FFCC66 url("butterfly.gif") no-repeat fixed right bottom;
```

Порядок свойств этого элемента таков:

[background-color] | [background-image] | [background-repeat] | [background-attachment] | [background-position]

Если свойство отсутствует, оно автоматически получает значение по умолчанию. Например, если background-attachment и background-position нет в данном примере:

```
background: #FFCC66 url("butterfly.gif") no-repeat;
```

то этим двум неспецифицированным свойствам будут присвоены значения по умолчанию – scroll и top left.

**4. Шрифты**

В этой части вы изучите работу со шрифтами с помощью CSS. Мы рассмотрим также вопрос о том, что конкретный шрифт, выбранный для web-сайта, может отображаться только в том случае, если этот шрифт установлен на PC, с которого выполняется доступ к этому web-сайту. Дано описание следующих CSS-свойств:

- font-family
- font-style
- font-variant
- font-weight
- font-size
- font

***Семейство шрифта [font-family]***

Свойство font-family указывает приоритетный список шрифтов, используемых для отображения данного элемента или web-страницы. Если первый шрифт списка не установлен на компьютере, с которого выполняется доступ к сайту, ищется следующий шрифт списка, пока не будет найден подходящий.

Для категоризации шрифтов используются два типа имён: имя семейства/family-name и общее/родовое семейство/generic family. Эти два термина объясняются далее.

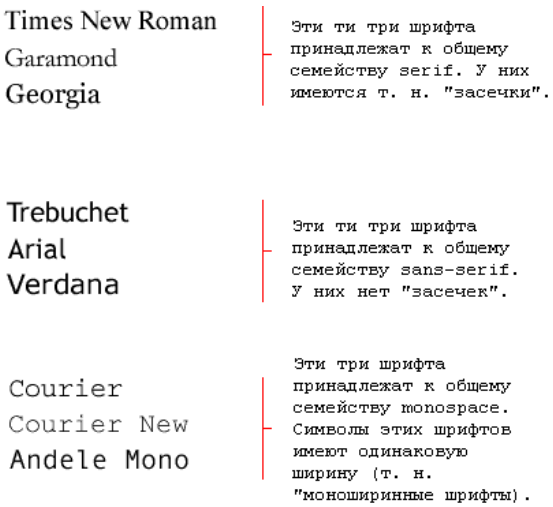
*Family-name*

Пример family-name (часто называемое просто "шрифт") это, например, "Arial", "Times New Roman" или "Tahoma".

*Generic family*

Его можно проще описать как группу family-names, имеющих характерные общие черты. Пример  sans-serif, набор шрифтов без "засечек/feet".

Разницу можно также проиллюстрировать так:




При указании шрифтов для вашего web-сайта вы, естественно, начинаете с предпочтительного шрифта, а затем перечисляете альтернативные. Рекомендуем в конце списка указывать родовое имя. Тогда страница, как минимум, будет отображена шрифтом того же семейства, если отсутствуют все специфицированные конкретные шрифты.

Список шрифтов может выглядеть так:

```
h1 {font-family: arial, verdana, sans-serif;}
h2 {font-family: "Times New Roman", serif;}
```

Заголовки <h1> будут отображаться шрифтом "Arial". Если он не установлен на пользовательской машине, будет использоваться "Verdana". Если недоступны оба шрифта, для показа заголовков будет использован шрифт семейства sans-serif.

Обратите внимание, что имя шрифта "Times New Roman" содержит пробелы, поэтому указано в двойных кавычках.

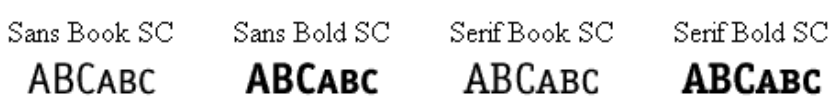
***Стиль шрифта [font-style]***

Свойство font-style определяет normal, italic или oblique. В примере все заголовки <h2> будут показаны курсивом italic.

```
h1 {font-family: arial, verdana, sans-serif;}
h2 {font-family: "Times New Roman", serif; font-style: italic;}
```

***Вариант шрифта [font-variant]***

Свойство font-variant используется для выбора между вариантами normal и small-caps. Шрифт small-caps использует малые заглавные буквы (upper case) вместо букв нижнего регистра. Непонятно? Смотрите примеры:



Если font-variant имеет значение small-caps, а шрифт small-caps недоступен, браузер, скорее всего, отобразит текст буквами верхнего регистра.

```
h1 {font-variant: small-caps;}
h2 {font-variant: normal;}
```

***Вес шрифта [font-weight]***

Свойство font-weight описывает, насколько толстым, или "тяжёлым", должен отображаться шрифт. Шрифт может быть normal или bold. Некоторые браузеры поддерживают даже числовые значения 100-900 (в сотнях) для описания веса шрифта.

```
p {font-family: arial, verdana, sans-serif;}
td {font-family: arial, verdana, sans-serif; font-weight: bold;}
```

***Размер шрифта [font-size]***

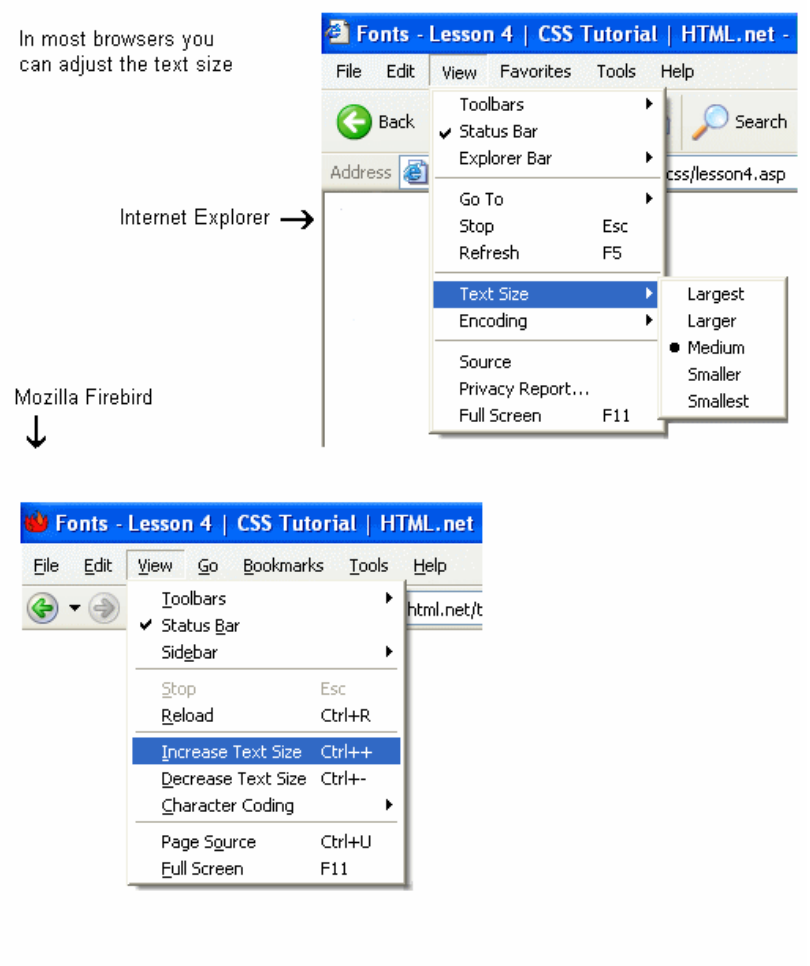
Размер шрифта устанавливается свойством font-size.

Используются различные единицы измерения (например, пикселы и проценты) для описания размера шрифта. Мы будем использовать самые распространённые и удобные единицы измерения. Вот примеры:

```
h1 {font-size: 30px;}
h2 {font-size: 12pt;}
h3 {font-size: 120%;}
p {font-size: 1em;}
```

Есть одно отличие в указанных единицах измерения: 'px' и 'pt' дают абсолютное значение размера шрифта, а '%' и 'em' – относительные. Многие пользователи не могут читать мелкий текст, по разным причинам. Чтобы сделать ваш web-сайт доступным для всех, вы должны использовать относительные значения, такие как '%' или 'em'.

Вот иллюстрация того, как настроить размер шрифта в Mozilla Firefox и Internet Explorer.




***Сокращённая запись [font]***

Используя сокращенную запись font, можно указывать все свойства шрифта в одном стилевом правиле.

Например, вот четыре строки описания свойств шрифта для <p>:

```
p {
font-style: italic;
font-weight: bold;
font-size: 30px;
font-family: arial, sans-serif;
}
```

Используя сокращённую запись, код можно упростить:

```
p {
font: italic bold 30px arial, sans-serif;
}
```

Порядок свойств font таков:

font-style | font-variant | font-weight | font-size | font-family

**5. Текст**

Форматирование и установка стиля текста - ключевая проблема для любого web-дизайнера. В этой части вы увидите впечатляющие возможности CSS при отображении текста. Будут рассмотрены следующие свойства:

- text-indent
- text-align
- text-decoration
- letter-spacing
- text-transform

***Отступы [text-indent]***

Свойство text-indent позволяет выделить параграф с помощью установки отступа для его первой строки. В примере 30px применяется ко всем параграфам <p>:

```
p {
text-indent: 30px;
}
```

***Выравнивание текста [text-align]***

CSS-свойство text-align соответствует атрибуту, используемому в старых версиях HTML. Текст может быть выровнен left, right, centred или justify.

В примере текст заголовочных ячеек таблицы <th> выравнивается вправо, а в ячейках данных <td> - по центру. Кроме того, нормальные параграфы - justify:

```
th {
text-align: right;
}

td {
text-align: center;
}

p {
text-align: justify;
}
```

***Декоративный вариант [text-decoration]***

Свойство text-decoration позволяет добавлять различные "декоративные эффекты". Например, можно подчеркнуть текст, провести линию по или над текстом и т. д. В примере <h1> подчёркнуты, <h2> - имеют черту над текстом, а <h3> - перечёркнуты.

```
h1 {
text-decoration: underline;
}

h2 {
text-decoration: overline;
}

h3 {
text-decoration: line-through;
}
```

***Интервал между буквами [letter-spacing]***

Интервал между буквами текста можно специфицировать свойством letter-spacing. Значение – нужная величина. Например, если вам необходимо 3px между буквами в параграфах <p> и 6px - в заголовках <h1>, то используется такой код:

```
h1 {
letter-spacing: 6px;
}

p {
letter-spacing: 3px;
}
```

***Трансформация текста [text-transform]***

Свойство text-transform управляет регистром символов. Можно выбрать capitalize, uppercase или lowercase, в зависимости от того, как выглядит текст в оригинальном HTML-коде.

Например, слово "headline" можно показать "HEADLINE" или "Headline". Имеются четыре возможных значения text-transform:

*Capitalize* - Капитализирует каждое слово. Например: "john doe" станет "John Doe".

*Uppercase* - Конвертирует все символы в верхний регистр. Например: "john doe" станет "JOHN DOE".

*Lowercase* - Конвертирует все символы в нижний регистр. Например: "JOHN DOE" станет "john doe".

*None* Трансформации нет - текст отображается так же, как в HTML-коде.

Для примера мы используем список имён. Все имена выделены с помощью <li> (list-item). Давайте капитализируем все имена и отобразим все заголовки верхним регистром.

```
h1 {
text-transform: uppercase;
}

li {
text-transform: capitalize;
}
```

**6. Ссылки**

Всё изученное в предыдущих частях вы можете применять и для ссылок/links (например изменять шрифт, цвет, подчёркивание и т. д). Новым будет то, что в CSS эти свойства можно определять по-разному, в зависимости от того, посетили уже ссылку, активна ли она, находится ли указатель мыши над ссылкой. Это позволяет добавить интересные эффекты на ваш web-сайт. Для этого используются так называемые псевдоклассы.

*Что такое псевдокласс?*

Псевдокласс позволяет учитывать различные условия или события при определении свойств HTM-тэга.

Рассмотрим пример. Как вы знаете, ссылки специфицируются в HTML тэгом <a>. В CSS мы также можем использовать a в качестве селектора:

```
a {
color: blue;
}
```

Ссылка может иметь разные состояния. Например, её уже посетили/visited или ещё нет. Можно использовать псевдоклассы для установки разных стилей посещённых и непосещённых ссылок.

```
a:link {
color: blue;
}

a:visited {
color: red;
}
```

Используйте |a:link| и |a:visited| для непосещённых и посещённых ссылок, соответственно. Активные ссылки имеют псевдокласс a:active, и a:hover, когда указатель - над ссылкой.

Мы рассмотрим каждый их этих четырёх псевдоклассов на примерах и с объяснениями.

***Псевдокласс: link***

Псевдокласс :link используется для ссылок на страницы, которые пользователь ещё не посещал.

В примере кода непосещённые ссылки - синие.

```
a:link {
color: #6699CC;
}
```

***Псевдокласс: visited***

Псевдокласс :visited используется для ссылок на страницы, которые пользователь посетил. В примере кода посещённые ссылки - фиолетовые.

```
a:visited {
color: #660099;
}
```

***Псевдокласс: active***

Псевдокласс :active используется для активных ссылок.

В примере активные ссылки имеют жёлтый фон.

```
a:active {
background-color: #FFFF00;
}
```

***Псевдокласс: hover***

Псевдокласс :hover используется для ссылок, над которыми находится указатель мыши. Это можно использовать для создания интересных эффектов. Например, если мы хотим, чтобы ссылки становились оранжевыми и курсивными при прохождении указателя над ними, то наш CSS должен выглядеть так:

```
a:hover {
color: orange;
font-style: italic;
}
```

**Пример 1: Эффект при нахождении указателя над ссылкой**

Эффекты для положения указателя мыши над ссылкой стали уже общим местом. Мы рассмотрим несколько дополнительных примеров для псевдокласса :hover.

***Пример 1a: Расстояние между буквами***

Как вы помните из Части 5, расстояние между символами можно установить свойством letter-spacing. Это можно применить для ссылки:

```
a:hover {
letter-spacing: 10px;
font-weight:bold;
color:red;
}
```

***Пример 1b: UPPERCASE и lowercase***

В Части 5 мы рассмотрели свойство text-transform, которое может переключать символы с верхнего на нижний регистр. Это также можно использовать для создания эффектов на ссылке:

```
a:hover {
text-transform: uppercase;
font-weight:bold;
color:blue;
background-color:yellow;
}
```

**Пример 2: Удаление подчёркивания ссылок**

Вы должны точно определить, нужно ли убрать подчёркивание ссылок, так как это может значительно снизить использование вашего web-сайта. Люди привыкли видеть на web-страницах синие подчёркнутые ссылки и знают, что по ним нужно щёлкать. Даже моя мама знает это! Если вы уберёте подчёркивание и измените цвет ссылок, весьма вероятно,

что это смутит пользователей и они не получат доступа ко всему содержимому вашего сайта web-сайт.

Вообще-то удалить подчёркивание ссылок очень просто. Как вы, может быть, помните из Части 5, свойство text-decoration можно использовать для определения подчёркивания текста. Для удаления подчёркивания просто установите в text-decoration значение none.

```
a {
text-decoration:none;
}
```

Альтернативно можно также установить text-decoration, наряду с другими свойствами, для всех четырёх псевдоклассов.

```
a:link {
color: blue;
text-decoration:none;
}

a:visited {
color: purple;
text-decoration:none;
}

a:active {
background-color: yellow;
text-decoration:none;
}

a:hover {
color:red;
text-decoration:none;
}
```

**7. Идентификация и группирование элементов (class и id)**

Иногда вам нужно будет применить особый стиль к определённому элементу или конкретной группе элементов. В этой части мы подробно разберём, как можно использовать class и id для специфицирования свойств выбранных элементов.

Как изменить цвет конкретного заголовка отдельно от других заголовков на вашем web-сайте? Как группировать ссылки по категориям и задавать для каждой категории особый стиль? Это лишь примерные вопросы, на которые мы ответим в этом уроке.

***Группирование элементов с помощью class***

Предположим, у нас есть два списка ссылок сортов винограда - для белого и для красного вина. HTML-код может быть таким:

```
<p>Виноград для белого вина:</p>
<ul>
<li><a href="ri.htm">Рислинг</a></li>
<li><a href="ch.htm">Шардонэ</a></li>
<li><a href="pb.htm">Пино Блан</a></li>
</ul>

<p>Виноград для красного вина:</p>
<ul>
<li><a href="cs.htm">Каберне Совиньон</a></li>
<li><a href="me.htm">Мерло</a></li>
<li><a href="pn.htm">Пино Нуар</a></li>
</ul>
```

Далее, мы хотим, чтобы ссылки на белое вино были жёлтого цвета, на красное вино - красного, а остальные ссылки на этой же странице оставались синими.

Для достижения этой цели мы разделим ссылки на две категории с помощью присвоения класса каждой ссылке атрибутом class.

Попробуем установить классы для предыдущего примера:

```
<p>Виноград для белого вина:</p>
<ul>
<li><a href="ri.htm" class="whitewine">Рислинг</a></li>
<li><a href="ch.htm" class="whitewine">Шардонэ</a></li>
<li><a href="pb.htm" class="whitewine">Пино Блан</a></li>
</ul>

<p>Виноград для красного вина:</p>
<ul>
<li><a href="cs.htm" class="redwine">Каберне Совиньон</a></li>
<li><a href="me.htm" class="redwine">Мерло</a></li>
<li><a href="pn.htm" class="redwine">Пино Нуар</a></li>
</ul>
```

Далее мы можем определить специальные свойства для ссылок whitewine и redwine, соответственно.

***Идентификация элемента с помощью id***

Помимо группирования элементов вам может понадобиться идентифицировать один уникальный элемент. Это можно реализовать с помощью атрибута id.

Особенность id в том, что в документе не может быть более одного элемента с данным конкретным id. Каждый id должен быть уникальным. В других случаях используйте атрибут class. Теперь взглянем на пример использования id:

```
<h1>Глава 1</h1>
...
<h2>Глава 1.1</h2>
...
<h2>Глава 1.2</h2>
...
<h1>Глава 2</h1>
...
<h2>Глава 2.1</h2>
...
<h3>Глава 2.1.2</h3>
...
```

Это могут быть заголовки документа, разделённого на главы или параграфы. Естественным будет назначить id каждой главе:

```
<h1 id="c1">Глава 1</h1>
...
<h2 id="c1-1">Глава 1.1</h2>
...
<h2 id="c1-2">Глава 1.2</h2>
...
<h1 id="c2">Глава 2</h1>
...
<h2 id="c2-1">Глава 2.1</h2>
...
<h3 id="c2-1-2">Глава 2.1.2</h3>
...
```

Заголовок, скажем, chapter 1.2, должен быть красным. Это делается в соответствии с CSS:

```
#c1-2 {
color: red;
}
```

**8. Группирование элементов (span и div)**

Элементы  и  используются для структурирования документа, часто совместно с атрибутами class и id.

В этой части мы подробно рассмотрим, как использовать  и , поскольку эти элементы HTML имеют важнейшее значение в CSS.

- Группирование с помощью 
- Группирование с помощью 

***Группирование с помощью ***

Элемент  можно назвать нейтральным элементом, который ничего не добавляет к содержимому документа. Но, в сочетании с CSS,  может использоваться для визуальных эффектов применимо к отдельным блокам текста.

Пример – цитата из Бенджамина Франклина:

```
        <p>Кто рано ложится и рано встаёт,
тот будет здоровым, богатым и умным</p>
```

Скажем, мы хотим, чтобы Mr. Franklin увидел все преимущества бодрствования выделенные красным цветом. Для этого мы можем отметить эти преимущества с помощью . Каждому блоку span будет присвоен class, который затем можно определить в нашей таблице стилей:

```
        <p>Кто рано ложится и рано встаёт,
 тот будет <span class="benefit">здоровым</span>,
<span class="benefit">богатым</span>
и <span class="benefit">умным</span>.</p>
```

В CSS:

```
span.benefit {
color:red;
}
```

Разумеется, вы можете также использовать id для определения стиля -элементов. Не забывайте только, что вы должны установить уникальный id каждому из трёх -элементов.

***Группирование с помощью ***

В то время как  используется в элементах уровня блока, как в предыдущем примере,  применяется для группирования одного или более блок-элементов.

Кроме этого отличия, группирование с помощью  работает более или менее аналогично. Посмотрим на пример – два списка президентов США, сгруппированных по их политической принадлежности:

```
<div id="democrats">
<ul>
<li>Франклин Д. Рузвелт</li>
<li>Гарри Трумэн</li>
<li>Джон Ф. Кеннеди</li>
<li>Линдон Б. Джонсон</li>
<li>Джимми Картер</li>
<li>Билл Клинтон</li>
</ul>
</div>

<div id="republicans">
<ul>
<li>Дуайт Д. Эйзенхауэр</li>
<li>Ричард Никсон</li>
<li>Джэралд Форд</li>
<li>Рональд Рейган</li>
<li>Джордж Буш</li>
<li>Джордж У. Буш</li>
</ul>
</div>
```

В нашей таблице стилей мы можем использовать такое же группирование, как и раньше:

```
#democrats {
background:blue;
}

#republicans {
background:red;
}
```

**Задание:** Отформатировать при помощи CSS web-сайт, созданный в первой лабораторный работе.

**Лабораторная работа №3**

**Тема: «Средства мультимедиа в HTML»**

Упражнение 1. Задание фонового изображения

1. Выберите подходящее для фона изображение и скопируйте его в рабочую папку. Переименуйте его в img1.gif.

```
<html>
<head>
<title>Упражнение 1</ title>
</head>
<body background=”img1.gif”>
--- Это моя страница, но уже с интересным фоном ---
</body>
</html>
```

2. Сохраните документ с именем Ex1.html в рабочей папке.

Упражнение 2. Вставка изображений

1. Выберите подходящее изображение и скопируйте его в рабочую папку. Переименуйте его в img2.gif.

```
<html>
<head>
<title>Упражнение 2</ title>
</head>
<body>
```

Вставка изображения <br>

```
<img src="img2.gif">
< img src="img2.gif" width=82 height=68>
< img src="img2.gif" width=200 height=68>
< img src="img2.gif" width=20>
</body>
</html>
```

2. Сохраните документ с именем Ex2.html в рабочей папке.

Упражнение 3. Создания таблицы

```
<html>
<head>
<title>Упражнение 11</ title>
</head>
<body>
<table border cellspacing=0 cellpadding=10>
<caption align=top> Оргтехника
</caption>
 <tr>
          <th>Наименование</th>
 <th> Цена </th>
          <th> Количество </th>
 </tr>
 <tr>
          <th> Принтер </th>
          <td align =right> 350 </td>
```

```
          <td align =right> 2 </td>
 </tr>
 <tr>
          <th> Ксерокс</th>
          <td align =right> 1250 </td>
          <td align =right> 1 </td>
 </tr>
 <tr align=center>
          <th> Факс</th>
          <td align =right> 250 </td>
          <td align =right> 2 </td>
 </tr>
</table>
</body>
</html>
```

Сохраните документ с именем Ex3.html в рабочей папке.

Упражнение 4. Создание вложенных таблиц

```
    <html>
    <head>
    <title>Упражнение 12</ title>
    </head>
    <body>
          <table border =3 width=200 height=100>
    <tr>
     <td>
     <table border =3>
                          <tr>
     <td>Ed</ td>
     </tr>
                    </table>
                    </td>
     </tr>
    </table>
    </body>
    </html>
```

Сохраните документ с именем Ex4.html в рабочей папке.

**Задание.** Верстка одной из страниц сайта и графическое оформление страниц. В ходе выполнения данного задания необходимо сверстать одну из страниц сайта и добавить графическую информацию на страницы.

1. Оформите одну из страниц сайта (например, главную страницу), с помощью табличной верстки, то есть используя для взаимного расположения элементов на странице таблицы с невидимыми границами. Для этого продумайте, как будут располагаться элементы на странице относительно друг друга, создайте таблицу и расположите элементы в ее ячейках.

2. Задайте фоновые изображения на страницах, если это необходимо.

3. Поместите графическую информацию на страницы сайта, а также добавьте аудио- и видео-файлы.

Упражнение 5. Создание и использование каскадных таблиц стилей

1. Внутренние таблицы стилей задают стиль только одному элементу документа при помощи атрибута style в HTML теге.

```
<html>
<head>
<title>Упражнение 13a</title>
</head>
<body>
<p color ="blue" size="3" face="Arial">
Пример физического форматирования.
</p>
<p style=”color:green; font-size:12pt; font-family:Arial”>
Пример встроенной таблицы стилей.
</p>
</body>
</html>
```

2. Сохраните документ с именем Ex13a.html в рабочей папке.

3. Глобальные стили (внедренные таблицы) задают вид элементов всего документа. Для этого используется тег <style type="text/css">. Он размещается в заголовке документа.

```
<html>
<head>
 <title>Упражнение 5</title>
<style type="text/css">
h1{color:red; font-style:italic; font-size:32px;}
p{color:blue}
</style>
</head>
<body>
<p>Пример внедренной таблицы стилей.</p>
<h1> Этот заголовок написан крупным красным курсивом. </h1>
<p> Это предложение выделено синим цветом.</p>
</body>
</html>
```

4. Сохраните документ с именем Ex5.html в рабочей папке.

5. Использование внешних стилевых таблиц — самый экономный и обобщающий способ задания правил оформления однотипных элементов для любого количества страниц. То есть одну таблицу стилей можно использовать для форматирования многих страниц, что приводит к единообразному отображению различных документов и придает некоторую системность серверу разработчика. Осуществляется связывание отдельного файла, содержащего стилевые правила, с множеством гипертекстовых документов. Связываемый файл содержит набор правил.

```
body{background:grey; font-size:14pt; color:red; font-family:Arial;}
h1, p{color:green;}
```

6. Сохраните документ с именем example_styles.css в рабочей папке.

7. В самих же HTML документах делается ссылка на этот файл при помощи тега <link>. Выглядит это так: <link rel="stylesheet" type="text/css" href="адрес файла со стилями">

```
<html>
<head>
<title> Упражнение 13c</title>
<link rel="stylesheet" type="text/css"
href="example_styles.css">
</head>
<body>
```

Пример внешней таблицы стилей

```
<h1> Этот заголовок выделен зеленым цветом. </h1>
<p> И это предложение тоже.</p>
</body>
</html>
```

8. Сохраните документ с именем Ex5c.html в рабочей папке.

Упражнение 6. Использование классов в создании каскадных таблиц стилей

```
<html>
<head>
<title> Упражнениеи 6</title>
<style>
 .blue {color:blue; font-style:italic;}
 #boldunderline {text-decoration:underline; font-
weight:bold;}
</style>
</head>
<body>
<p class="blue"> Здравствуйте, это моя домашняя страница.
</p>
<p class="blue" id="boldunderline"> Пока еще в стадии
разработки ...
</p>
<p id="boldunderline">... Но скоро откроется </p>
</body>
</html>
```

Сохраните документ с именем Ex6.html в рабочей папке.

Упражнение 7. Использование блочной верстки

1. Откройте новый HTML-документ.
2. Создайте в нем три блока разного цвета:

```
<div style="background:#red;">
1
</div>
 <div style="background:#green;">
2
</div>
 <div style="background:#blue;">
3
</div>
```

3. Задайте ширину для каждого блока:

```
<div style="background:#red; width:100px;">
```

```
1
</div>
 <div style="background:#green; width:200px;">
2
</div>
 <div style="background:#blue; width:300px;">
3
</div>
```

4. Теперь предлагается каждому из блоков добавить свойство float со значением, например left, блоки должны выстроиться в один ряд:

```
<div style="background:#red; width:100px; float:left;">
1
</div>
 <div style="background:#green; width:200px; float:left;">
2
</div>
 <div style="background:#blue; width:300px; float:left;">
60
3
</div>
5. Попробуйте добавить ниже еще один блок другого цвета:
<div style="background:#red; width:100px; float:left;">
1
</div>
 <div style="background:#green; width:200px; float:left;">
2
</div>
 <div style="background:#blue; width:300px; float:left;">
3
</div>
 <div style="background:#orange;">4<br>4
</div>
```

6. Так как выше у нас блоки со свойством float, то для того, чтобы четвертый блок оказался под уже созданными тремя, надо для четвертого блока задать свойство clear:

```
 <div style="background:#red; width:100px; float:left;">
1
</div>
 <div style="background:#green; width:200px; float:left;">
2
</div>
 <div style="background:#blue; width:300px; float:left;">
3
</div>
 <div style="background:#orange; clear:left;">4<br>4
</div>
```

7. Для того, чтобы убрать отступ между четвертым блоком и первыми тремя добавьте для всей страницы стилевые свойства margin и padding:

```
<style type="text/css">
 *{ margin:0px; padding:0px; }
</style>
```

8. Должен получиться следующий вариант страницы:

```
<html>
<head>
<title>Упражнение 15</title>
<style type="text/css">
 * { margin:0px; padding:0px; }
</style>
</head>
<body>
<div style="background:#red; width:100px;
float:left;">1</div>
<div style="background:#green; width:200px;
float:left;">2</div>
<div style="background:#blue; width:300px;
float:left;">3</div>
<div style="background:#orange;
clear:left;">4<br>4<br>4</div>
</body>
</html>
```

9. Сохраните документ с именем Ex6.html в рабочей папке.

**Задание.** Верстка последующих страниц сайта и стилевое оформление.

На данном этапе создания сайта необходимо завершить верстку всех страниц сайта выбранными способами и оформить страницы, используя таблицы стилей.

1. Используйте блочную верстку для одной из оставшихся страниц сайта. Для этого сначала скомпонует блоки на странице, затем поместите в них содержание.

2. Для остальных страниц сайта используйте по желанию либо блочную, либо табличную верстку. При этом заготовьте отдельную страницу, которая может представлять собой либо страницу обратной связи, либо анкету, либо регистрацию, для создания формы.

3. Задайте с помощью внешних каскадных таблиц стилей одинаковой стилевое оформление страниц сайта.

4. Если необходимо, уже с помощью внедренных и встроенных таблиц стилей оформите отдельные элементы на страницах сайта.

# Лабораторная работа №4.

# Тема: Формы в HTML

Назначением формы является сбор информации от пользователей. После того как пользователь заполнит форму и запускает процесс ее обработки, информация из нее попадает в программу, работающую на сервере. Другая программа под названием Common Gateway Interface (CGI) обрабатывает ее. Таким образом, пользователь может интерактивно взаимодействовать с сервером Web через Internet.

## Тэг < FORM >

Тэг < FORM > используется для обозначения документа как формы. Данный тэг определяет границы использования других тэгов, размещаемых в форме.

< FORM > определяется последовательностью тэгов < INPUT >, размещенных внутри пары <FORM> и </FORM>.

Данный тэг поддерживает атрибуты ACTION, ENCTYPE, METHOD:

- метод (METHOD) – принимает значения GET или POST – определяет, как программист должен обрабатывать входные данные из формы.
- действие (ACTION) – указывает на URI программы, ответственной за обработку данных.

## Сбор данных при помощи форм (тэг < INPUT >)

Тэг < INPUT > используется для определения области внутри формы, где собираются данные. Данный тэг представляет собой поле для ввода информации пользователем (обычно одна строка текста). В этом случае требуется наличие атрибута NAME для определения наименования переменной поля.

Можно использовать следующие атрибуты:

- MAXLENGTH – ограничивает число вводимых символов (по умолчанию ограничений нет).
- SIZE – размер видимой на экране области, занимаемой текущим полем. Если MAXLENGTH > SIZE, броузер будет прокручивать данные в окне.
- VALUE – определяет начальное значение поля ввода.

а также атрибуты: ALIGN , CHECKED , NAME , SRC , TYPE .

Пример 1 – простая форма для ввода:

```
<FORM>
Улица:  <INPUT NAME= "street"> <BR>
Город: <INPUT NAME= "city" SIZE= "20" MAXLENGTH= "20"> <BR>
Индекс:  <INPUT NAME= "zip"  SIZE= "5"  MAXLENGTH= "5" VALUE= "99999">
<BR>
</FORM>
```

В окне броузера это будет выглядеть так:

Улица:
Город:
Индекс: 99999

## Атрибут CHECKBOX

При создании форм часто бывает необходимо получить ответ пользователя на вопрос типа (Да/Нет) или (Правда/Ложь). Например, нужно выбрать из списка несколько значений. Для создания независимых кнопок в формах можно использовать атрибут CHECKBOX. В зависимости от содержания можно отметить несколько флагов.

Вместе с атрибутом CHECKBOX должны использоваться следующие атрибуты:

- CHECKED – инициализировать данный флаг, как отмеченный
- NAME – наименование поля ввода формы
- VALUE – значение поля ввода

Пример 2 (элемент «Котлеты» указан как заранее отмеченный):

```
Выберите Ваше любимое блюдо:
<FORM>
<INPUT TYPE="checkbox" NAME="food" VALUE="Пельмени"> Пельмени <BR>
<INPUT TYPE="checkbox" NAME="food" VALUE="Голубцы"> Голубцы <BR>
<INPUT TYPE="checkbox" NAME="food" VALUE="Котлеты" CHECKED> Котлеты <BR>
</FORM>
```

В окне броузера это будет выглядеть так:

Выберите Ваше любимое блюдо:

- ☐ Пельмени
- ☐ Голубцы
- ☑ Котлеты

## Атрибут PASSWORD

Данный атрибут используется для организации ввода пароля без вывода на экран составляющих его символов (вместо символов выводятся звездочки).

Пример 3

```
<FORM>
Ввведите имя:
<INPUT NAME="login"><BR>
Ввведите пароль:
<INPUT TYPE="password" NAME="p_word">
</FORM>
```

## Атрибут RADIO

Данный атрибут используется для организации выбора одного единственного варианта их нескольких возможных.

Вместе с атрибутом RADIO должны использоваться следующие атрибуты:

- CHECKED – инициализировать данный флаг, как отмеченный
- NAME – наименование поля ввода формы
- VALUE – значение поля ввода

Пример 4

```
За какую команду Вы болеете?
<FORM>
   <INPUT TYPE="radio" NAME="beer" VALUE="din"> Динамо<BR>
   <INPUT TYPE="radio" NAME="beer" VALUE="shak"> Шахтер<BR>
   <INPUT TYPE="radio" NAME="beer" VALUE="noteam"> Я не смотрю футбол<BR>
</FORM>
```

Результат:

За какую команду Вы болеете?

- ○ Динамо
- ○ Шахтер
- ○ Я не смотрю футбол

## Атрибут RESET

Данный атрибут используется для создания кнопки «Reset». При нажатии на эту кнопку форма восстанавливает первоначальные значения полей всех элементов <INPUT>, в которых присутствует атрибут RESET.

Вместе с атрибутом RESET может использоваться атрибут VALUE – значение поля ввода по умолчанию.

Пример 5

```
<FORM>
    Код: <INPUT NAME="cod"> <BR>
    Телефон: <INPUT NAME="phone" SIZE="6" MAXLENGTH="6" ><BR>
    <INPUT TYPE="RESET">
</FORM>
```

Результат:

Код:

Телефон:

Сбросить

## Атрибут SELECT

Для организации списков с прокруткой и выпадающим меню можно использовать атрибут <SELECT>. Для определения списка пунктов используются элементы <OPTION> внутри <SELECT>.

Вместе с атрибутом SELECT можно использовать следующие атрибуты:

- NAME – наименование объекта
- MULTIPLE – позволяет выбрать более чем одно наименование
- SIZE – определяет число пунктов, видимых для пользователя.

SIZE = 1 – браузер выводит список на экран в виде выпадающего меню (видно одно наименование)

SIZE > 1 – броузер представляет на экране обычный список (число - количество видимых наименований)

С элементом OPTION можно использовать следующие атрибуты:

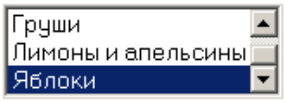
- SELECTED – для первоначального выбора значения элемента по умолчанию
- VALUE – значение, возвращаемое формой после выбора пользователем данного пункта. По умолчанию значение поля равно элементу < OPTION >.

Когда пользователь заполняет форму, атрибут NAME элемента <SELECT> состыковывается с атрибутом VALUE элемента для формирования наименования, выбранного пользователем.

Пример 6

```
<FORM>
    <SELECT NAME="fruct" SIZE="3">
        <OPTION VALUE="Plum"> Сливы </OPTION>
        <OPTION VALUE="Pear"> Груши </OPTION>
        <OPTION VALUE="Lemon_and_orange">Лимоны и апельсины </OPTION>
        <OPTION VALUE="Apple" selected> Яблоки </OPTION>
    </SELECT>
</FORM>
```

Результат:




## Атрибут SUBMIT

Данный атрибут используется при окончании ввода пользователем данных. Броузер, в свою очередь выводит данный элемент, как кнопку, на которой пользователь может щелкнуть, чтобы завершить процесс редактирования.

Вместе с атрибутом SUBMIT можно использовать следующие атрибуты:

NAME – наименование кнопки SUBMIT

VALUE – значение переменной поля в вашей форме

Для создания простой кнопки используется атрибут <BUTTON>

## Атрибут TEXTAREA

Данный атрибут используется для ввода большого количества текстовой информации (несколько строк).

Вместе с атрибутом TEXTAREA можно использовать следующие атрибуты:

- NAME – наименование поля
- COLS – число колонок (символов) в текстовой области
- ROWS – число видимых строк в текстовой области

```
<FORM>
<TEXTAREA NAME="adress" COLS="64" ROWS="6">
Украина
Мариуполь
пр.Нахимова, 99
</TEXTAREA>
</FORM>
```

Украина
Мариуполь
пр.Нахимова, 99

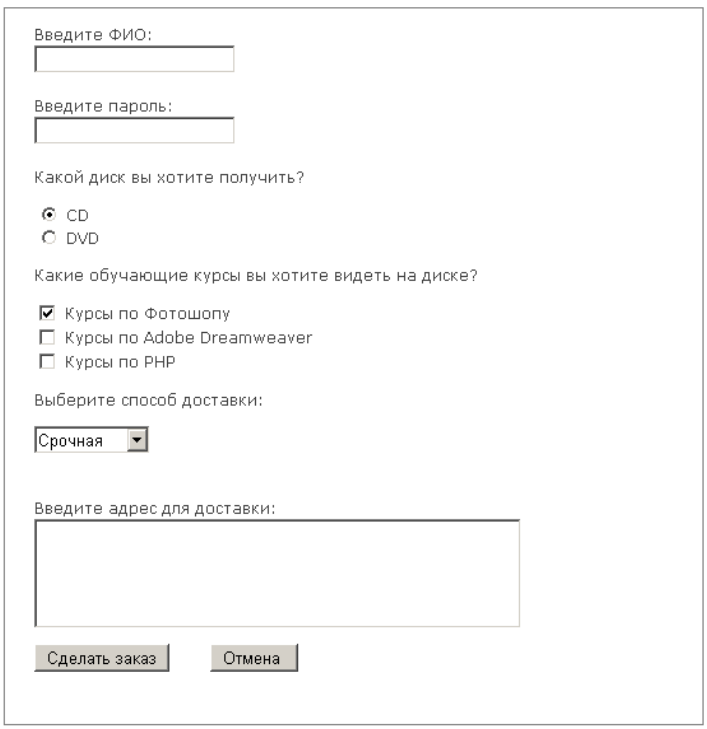
## Отправление файлов при помощи форм

Формы можно использовать для отправки не только небольших информационных сообщений ввиде параметров, а также и для отправки файлов.

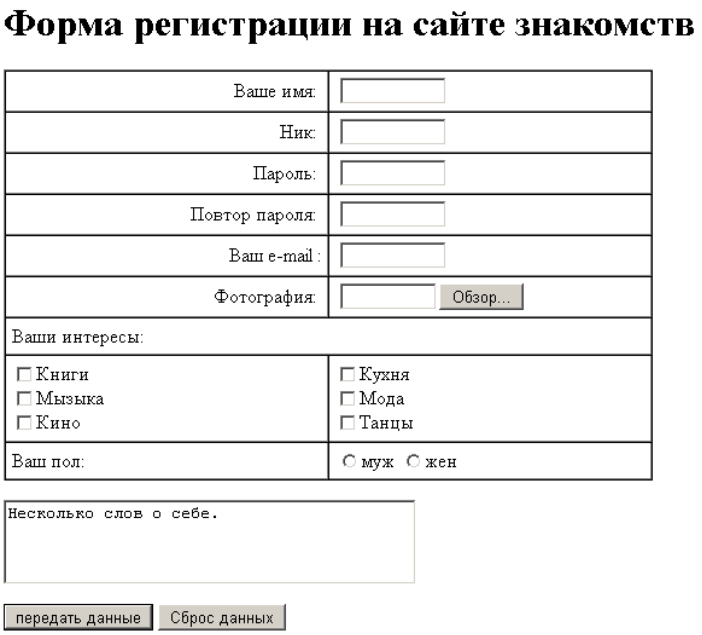
Например:

```
<FORM ENCTYPE="multipart/form-data" ACTION="url" METHOD=POST>
Отправить данный файл: <INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Отправить файл">
</FORM>
```

**Задание 1**. Создайте форму, изображенную на рисунке. Способ доставки принимает значения – срочная, не срочная, курьером.




**Задание 2**. Создайте форму, изображенную на рисунке




**Задание 3**. Создайте произвольную форму (отвечающую кругу ваших интересов). При создании формы воспользуйтесь как минимум пятью элементами формы из HTML5. Используйте шаблон ввода (атрибут pattern) на два элемента вашей формы.

# Лабораторная работа №4.
# Тема: Программное взаимодействие JavaScript с HTML-документами.

**Цели работы:** создать первый скрипт и ознакомится с основами создания и размещения JavaScript на веб-странице.

## Что такое JavaScript?

Во-первых, это не Java. Таким образом Java и JavaScript – это не одно и то же. Java – это язык программирования, разработанный в Sun Microsystems. A JavaScript является продуктом Netscape. Но это не единственное отличие.

Оба языка представляют собой OOP (Object Orientated Programming - объектно-ориентированный язык программирования). Это значит, что с их помощью можно строить небольшие объекты, из которых потом складывается целое. Главное отличие в том, что Java позволяет создавать совершенно самостоятельные события. «Java-applet» («приложеньице») может запускаться с веб-страницы, но на самом деле это полностью независимая программа, хоть и маленькая. К тому же ее нельзя просмотреть в виде текста. Для запуска ее необходимо «транслировать» в то, что называется «машинным языком».

Netscape как бы упростил Java до набора более простых команд. JavaScript не может существовать сам по себе, он должен находиться внутри веб-страницы, а веб-страницу необходимо просматривать в браузере, который понимает язык JavaScript (скажем, Netscape Communicator и Internet Explorer).

JavaScript — это не HTML. У JavaScript и HTML очень похожие правила. Во-первых, JavaScript располагается внутри документа HTML. JavaScript сохраняется в виде текста вместе с документом HTML. Главная же разница в том, что в HTML имеет довольно расплывчатые правила. Не имеет значения, сколько пробелов вы оставляете между словами или абзацами. HTML можно было бы писать одной сплошной строкой. У JavaScript имеется четкая форма. И пренебрегать ею можно лишь изредка.

**Пример.**

```
<SCRIPT LANGUAGE="javascript">
document.write("<FONT COLOR='RED'>Это красный текст</FONT>")
</SCRIPT>
```

Например, вторая строка нашего скрипта выглядит следующим образом:

```
document.write("<font color='red'>Красный текст</font>")
```

То есть, целиком находится на одной линии и должна сохранять свою форму. Предположим, вы скопировали ее в текстовый редактор с узкими страницами, и поля разорвали строку:

```
document.write("<font color='red'>Красный текст</font
                                >")
```

Таким образом записанный скрипт имеет ошибки.

При написании скрипта желательно использовать текстовый редактор, например NotePad поскольку в нем нет полей. При написании скрипта обращайте внимание на регистр букв.

Ниже разбирается этот скрипт.

Начнем с первой строки:

```
<SCRIPT LANGUAGE="JavaScript">
```

Это код HTML, который дает браузеру понять, что с этого места начинается JavaScript. Все скрипты начинаются с такой команды. Запсись LANGUAGE(язык)="JavaScript" является необходимой.

Этим:

```
</SCRIPT>
```

...заканчивается любой JavaScript без исключений.

Скрипт начинается с <SCRIPT LANGUAGE="javascript"> и заканчивается </SCRIPT>. Основная часть скрипта имеет следующий вид:

```
document.write("<FONT COLOR='RED'>Это красный текст</FONT>")
```

Вот из чего состоит скрипт: указывается DOCUMENT (документ HTML) и те изменения, которые в нем произойдут – что-то будет написано (WRITE). То, что будет написано, находится в скобках.

DOCUMENT представляет собой object (объект). Слово WRITE (писать), отделенное точкой, называется method (методом объекта). Таким образом, скрипт попросту говорит: «Возьмите объект (что-то, уже существующее) и припишите что-то к нему».

Текст в скобках называется instance (примером метода), он передает то, что происходит, когда метод воздействует на объект. Имейте в виду, что текст внутри скобок находится в кавычках. Никогда нельзя про них забывать.

Текст в кавычках представляет собой простой HTML. Обратите внимание, что дальше идут одинарные кавычки. Если поставить двойные, JavaScript решит, что это конец строки, и получится, что только часть вашего текста будет применена к объекту, а это уже ошибка. Запомните: внутри двойных кавычек ставятся одинарные. JavaScript не перекрашивает текст, а текст перекрасил HTML.

**Пример. Работа с датами**

```
<SCRIPT LANGUAGE="JavaScript">
//Скрипт отмечает точную дату и время вашего прибытия на страницу
Now = new Date();
document.write("Сегодня " + Now.getDate()+
"-" + Now.getMonth() + "-" + Now.getFullYear() + ".
Вы зашли на мою страницу ровно в: " + Now.getHours() +
":" + Now.getMinutes() + " и " + Now.getSeconds() +
" секунд.")
</SCRIPT>
```

Обратите внимание, что синтаксис программы очень похож на программы написанные на C++.

**Пример. Работа с мышью**

```
<A HREF="http://www.newmail.ru"
onMouseOver="window.status='Бесплатный хостинг';
return true">Ссылка</A>
Пример 4. Команда onClick
<a href="http://www.jsp.newmail.ru" onClick="alert('Attention!');">
JavaScript it's easy</a>
```

**Пример. Команда onFocus**

```
<form>
<input type="text" size="30"
onFocus="window.status='Текст в строке состояния';">
</form>
```

**Пример. Команда onBlur**

```
<form>
<input type="text" size="45" value="Впишите свое имя и щелкните по
другой строке"
onBlur="alert('Вы изменили ответ — уверены, что он правильный?');">
</form>
```

**Пример. Команда onChange**

```
<form>
<input TYPE="text" size="45"
value="Измените текст и щелкните по другой строке"
onChange="window.status='Текст был изменен';">
</form>
```

**Пример. Команда onSubmit**

```
<form>
<input TYPE="submit"
onSubmit="parent.location='thanksalot.html'";>
</form>
```

**Пример. Получение информации от пользователя**

```
<SCRIPT LANGUAGE="javascript">
/* Скрипт предназначен для того, чтобы получить
от пользователя информацию и поместить ее на страницу */

var user_name = prompt ("Напишите свое имя","Здесь");
document.write("Добрый день, " + user_name + "! Добро
пожаловать!");
</SCRIPT>
```

**Пример. Функции в JavaScript**

```
<SCRIPT LANGUAGE="javascript">
<!-- Скрыть от браузеров, не читающих Javascript
function dateinbar()
{
 var d = new Date();
 var y = d.getFullYear();
 var da = d.getDate();
 var m = d.getMonth() + 1;
 var t = da + '/' + m + '/' + y;

 defaultStatus = "Вы прибыли на страницу " + t + ".";
 }
 // не скрывать -->
 </SCRIPT>
 ...и команда onLoad в <BODY>:
 <BODY BGCOLOR="xxxxxx" onLoad="dateinbar()">
```

**Индивидуальные задания.**

Напишите скрипт который делал бы следующее

| № | Вариант задания |
|---|---|
| 1. | Вывести две строки текста произвольного содержания друг под другом. Цвет первой строки должен быть красным, а второй синим. |
| 2. | Поместить на вашу страницу дату, разделенную дробями /. Приветственный текст должен быть зеленого цвета. |
| 3. | Метод, alert() (предупредить) вызывает небольшое диалоговое окно с текстом и кнопкой |

| | |
|---|---|
| | OK. Сделайте так, чтобы окно предупреждения всплывало при наведении курсора на ссылку. |
| 4. | Создать форму, которая будет взаимодействовать с пользователем. Форма должна иметь три элемента: поле ввода с просьбой ввести имя; два поля для флажков с вопросом о том, что пользователю больше нравиться учиться или отдыхать; кнопку отправки данных. При вводе имени, в строке состояния должны появиться слова: «Впишите сюда свое имя» |
| 5. | Создать форму, которая будет взаимодействовать с пользователем. Форма должна иметь три элемента: поле ввода с просьбой ввести имя; два поля для флажков с вопросом о том, что пользователю больше нравиться учиться или отдыхать; кнопку отправки данных. Два поля с флажками должны отослать в строку состояния слова: «Вы выбрали...» и выбор пользователя. |
| 6. | Создать форму, которая будет взаимодействовать с пользователем. Форма должна иметь три элемента: поле ввода с просьбой ввести имя; два поля для флажков с вопросом о том, что пользователю больше нравиться учиться или отдыхать; кнопку отправки данных. При нажатии на кнопку должно выскочить окно предупреждения, благодарящее пользователя за участие в опросе. |
| 7. | Создать ссылку на страницу на вашем сервере. Например, если вы находитесь на www.you.ru, JavaScript создаст ссылку на www.you.ru/index.html. Также, каким бы свойством вы ни воспользовались, присвойте ему переменную. |
| 8. | Создайте функцию, которая вызовет два запроса (prompt). (Подсказка: один следует за другим с новой строки.) Первый попросит пользователя ввести свое имя, второй — отчество. Затем та же функция должна вызвать окно предупреждения (alert) с текстом: **Hello, имя отчество, Welcome to web page, it's great site WWW!** |
| 9. | Напишите функцию, которая возвращает дату ее вызова и после пишет дату в TextBox. |
| 10. | Напишите функцию, которая при ее вызове меняет состояние у CheckBox и пишет некоторое содержание в TextBox. |

## Контрольные вопросы

1. Какая разница между HTML и JavaScript ?
2. Опишите работу с датами
3. Опишите работу с мышью
4. Опишите команду onClick
5. Опишите команду onFocus
6. Опишите команду onBlur
7. Опишите команду onChange
8. Опишите команду onSubmit
9. Как получить информацию от пользователя
10. Дайте пример представления функции в JavaScript.

## Лабораторная работа №5.

## Тема: JavaScript и DOM API

### Теоретические сведения

JavaScript позволяет "оживить" веб-страницу. Это реализуется путем добавления к статическому описанию фрагмента исполняемого кода. JavaScript-сценарий может взаимодействовать с любыми компонентами HTML-документа и реагировать на изменение их состояния.

JavaScript не является строго типизированным языком, в переменных могут храниться практически любые типы данных. Как и программа на языке Java, сценарий JavaScript выполняется под управлением интерпретатора. Однако если Java-приложение или Java-аплет компилируется в байтовый код, то сценарий JavaScript интерпретируется на уровне исходного текста.

Следует отметить, что языковые конструкции JavaScript совпадают с соответствующими средствами C++ и Java.

### Структура сценария

Сценарием JavaScript считается фрагмент кода, расположенный между дескрипторами <SCRIPT> и </SCRIPT>:

Текст HTML-документа
<SCRIPT>
Код сценария
</SCRIPT>
Текст HTML-документа

### Переменные

В сценариях JavaScript переменные могут хранить данные любых типов: числа, строки текста, логические значения, ссылки на объекты, а также специальные величины, например "нулевое" значение null или значение NaN, которое сообщает о недопустимости операции.

Переменная в языке JavaScript объявляется с помощью ключевого слова var. Так, например, выражение

```
var selected = "first item";
```

создает переменную с именем selected и присваивает ей в качестве значения строку символов " first item ". Переменные могут объявляться также автоматически. Это происходит при присвоении значения переменной, не встречавшейся ранее в данном сценарии. Так, в следующем примере создается переменная с именем rating, которой присваивается числовое значение, равное 512.5:

```
rating = 512.5;
```

### Объекты

В языке JavaScript не предусмотрены средства для работы с классами в том виде, в котором они реализованы в C++ или Java. Разработчик сценария не может создать подкласс на основе существующего класса, переопределить

метод или выполнить какую-либо другую операцию с классом. Сценарию, написанному на языке JavaScript, в основном доступны лишь готовые объекты. Построение нового объекта приходится выполнять лишь в редких случаях.

Объекты содержат свойства (свойства объектов можно сравнить с переменными) и методы. Объекты, а также их свойства и методы идентифицируются именами. Объектами являются формы, изображения, гипертекстовые ссылки и другие компоненты веб-страницы, HTML-документ, отображаемый в окне браузера, окно браузера и даже сам браузер. В процессе работы JavaScript сценарий обращается к этим объектам, получает информацию и управляет ими.

Кроме того, разработчику сценария на языке JavaScript доступны объекты, не связанные непосредственно с HTML-документом. Их называют предопределенными, или независимыми объектами. С помощью этих объектов можно реализовать массив, описать дату и время, выполнить математические вычисления и решить некоторые другие задачи.

Первый объект, с которым необходимо познакомиться, чтобы написать простейший сценарий, - это объект document, который описывает HTML документ, отображаемый в окне браузера. Ниже приведен исходный текст веб-страницы, содержащей сценарий, действия которого сводятся к выводу строки текста в окне браузера.

```
<HTML>
<HEAD>
<TITLE>Первый сценарий JavaScript</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
    document.write("Проверка сценария JavaScript");
</SCRIPT>
</BODY>
</HTML>
```

Имена чувствительны к регистрам символов, и если вы попытаетесь обратиться к текущему документу по имени Document, интерпретатор JavaScript отобразит сообщение об ошибке.

Основное назначение сценариев JavaScript - создавать динамически изменяющиеся объекты, корректировать содержимое HTML-документов в зависимости от особенностей окружения, осуществлять взаимодействие с пользователем и т.д.

**Операции**

Набор операторов в JavaScript, их назначение и правила использования в основном совпадают с принятыми в языке C++. Исключением является операция задаваемая символом " + ".

В JavaScript символ "+" определяет как суммирование числовых значений, так и конкатенацию строк.

Так, например, в результате вычисления выражения

sum = 47 + 21;

переменной sum будет присвоено значение 68, а после выполнения операции

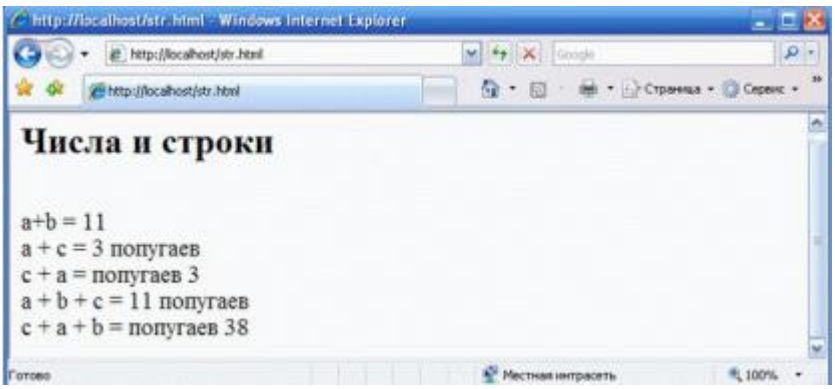
sum = "строка 1 " + "строка 2";

в переменную sum будет записана последовательность символов " строка 1 строка 2 ".

Рассмотрим еще один пример:

```
<HTML>
<BODY>
<H2>Числа и строки</H2><BR>
<SCRIPT LANGUAGE="JavaScript">
      var a = 3;
      var b = 8;
      var c = " попугаев ";
      document.write("a+b="); document.write(a + b);
      document.write("<BR>");
      document.write( "a + c = "); document.write(a+c);
      document.write("<BR>");
      document.write("c + a = "); document.write (c + a);
      document.write ("<BR>");
      document.write ("a + b + c = "); document.write(a + b + c);
      document.write("<BR>");
      document.write("c + a + b = "); document.write(c + a + b);
      document.write("<BR>");
</SCRIPT>
</BODY>
</HTML>
```

В окне браузера приведенный выше HTML-код выглядит так, как показано на скриншоте.




Первая строка отображает результат суммирования двух числовых значений, вторая и третья - результат конкатенации строки и символьного представления числа. Если операция суммирования чисел предшествует конкатенации, JavaScript вычисляет сумму чисел, представляет ее в символьном виде, затем производит конкатенацию двух строк. Если же первой в выражении указана операция конкатенации, то JavaScript сначала преобразует числовые значения в символьный вид, а затем выполняет конкатенацию строк.

**Управляющие конструкции**

Управляющие конструкции, используемые в языке C++, в основном применимы и в сценариях JavaScript.

В JavaScript дополнительно определены языковые конструкции, отсутствующие в C++, а именно: операторы for...in и with.

В пример 6.1 с помощью оператора цикла на веб-странице формируется таблица умножения чисел.

```
<html>
<body>
<table>
<script language="JavaScript">
document.write("<tr><td> </td>");
for (i = 1; i < 10; i++) document.write("<td>"+i+" </td>");
document.write("</tr>");
for (i = 1; i < 10; i++)
{
document.write("<tr><td>" + i + " </td>");
for (j = 1; j < 10; j++)
{
document.write("<td  bgcolor='#00ffa0'>" + (i*j) +
" </td>");}
document.write("</tr>");
}
</script>
</table>
</body>
</html>
```

**Пример 1.**

Отдельного внимания заслуживает оператор new. Несмотря на то, что большинство объектов уже созданы браузером и доступны сценарию, в некоторых случаях приходится создавать объекты в процессе работы. Это относится к предопределенным объектам и объектам, определяемым разработчиком сценария. Для создания объекта используется оператор new, который вызывается следующим образом:

переменная =  new   тип_ объекта (параметры)

**Функции**

Формат объявления функции выглядит следующим образом:

```
function имя функции ([ параметры]) тело функции
```

Объявление функции начинается с ключевого слова function. Так же, как и в языке C для идентификации функции используется имя, при вызове функции могут передаваться параметры, а по окончании выполнения возвращаться значение. Однако, в отличие от C, тип возвращаемого значения и типы параметров не задаются. Ниже показаны два способа вызова функции

• имя_функции ([параметры]);

• переменная = имя функции ([параметры]);

Во втором случае значение, возвращаемое функцией, присваивается указанной переменной.

**Область видимости переменных**

Работа с переменными в теле функции подчиняется следующим правилам.

- Если переменная объявлена с помощью ключевого слова var, доступ к ней осуществляется по правилам, подобным тем, которые используются в языке C.
- Переменная, объявленная внутри функции, считается локальной. Область видимости такой переменной ограничивается телом функции, в которой она объявлена.
- Переменная, объявленная вне функции, считается глобальной. К ней можно обращаться из любой точки сценария.
- • Если локальная и глобальная переменные имеют одинаковые имена, то в теле функции локальная переменная "маскирует" глобальную.
- Если переменная создается автоматически, т.е. если она не объявлена с помощью ключевого слова var, но присутствует в левой части оператора прямого присваивания, то она считается глобальной и становится доступной из любой точки сценария.

**HTML DOM**

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

В рамках данного стандарта можно выделить 3 части:

• Core DOM – стандартная модель любого структурированного документа

• XML DOM - стандартная модель XML документа

• HTML DOM - стандартная модель HTML документа DOM определяет объекты и свойства всех элементов документа и методы (интерфейс) для доступа к ним.

HTML DOM определяет объекты и свойства всех HTML элементов и методы (интерфейс) для доступа к ним. Иначе говоря, HTMLDOM описывает каким образом необходимо получать, изменять, добавлять и удалять HTML элементы.

В соответствии с моделью DOM все, что содержится внутри HTML документа - является узлом. То есть HTML документ представляется в виде дерева узлов, которыми являются элементы, атрибуты и текст.

**Узлы дерева HTML документа**

Согласно модели DOM:

- Весь документ представляется узлом документа;
- Каждый HTML тэг является узлом элемента;
- Текст внутри HTML элементов представляется текстовыми узлами;

- Каждому HTML атрибуту соответствует узел атрибута;
- Комментарии являются узлами комментариев.

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

**Пример 2.**

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

Следует обратить особое внимание на то, что текст, расположенный в узле элемента соответствует текстовому узлу. В примере<title>HTML документ</title> узел элемента <title> содержит текстовый узел " HTML документ ", то есть " HTML документ " не является значением элемента <title>. Тем не менее, в рамках HTML DOM значение текстового узла может быть доступно посредством свойства innerHTML.

Все узлы HTML документа могут быть доступны посредством дерева, при этом их содержимое может быть изменено или удалено, а также можно добавить новые элементы.

Все узлы дерева находятся в иерархических отношениях между собой. Для описания этих отношений используются термины родитель, дочерний элемент и потомок. Родительские узлы имеют дочерние узлы, а дочерние элементы одного уровня называются потомками (братьями или сестрами).

В отношении узлов дерева соблюдаются следующие принципы:

- Самый верхний узел дерева называется корневым;
- Каждый узел, за исключением корневого, имеет ровно один родительский узел;
- Узел может иметь любое число дочерних узлов;
- Конечный узел дерева не имеет дочерних узлов;
- Потомки имеют общего родителя.

**Программный интерфейс HTML DOM**

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

Свойства представляют некоторые сущности (например, <h1>), а методы - действия над ними (например, добавить <a>).

К типичным свойствам DOM относятся следующие:

- x.innerHTML – внутреннее текстовое значение HTML элемента x ;
- x. nodeName – имя x ;
- x.nodeValue – значение x;
- x.parentNode – родительский узел для x ;
- x.childNodes – дочерний узел для x ;
- x.attributes – узлы атрибутов x.

Узловой объект, соответствующий HTML элементу поддерживает следующие методы:

- x.getElementById(id) – получить элемент с указанным id ;
- x.getElementsByTagName(name) – получить все элементы с указанным именем тэга ( name );
- x.appendChild(node) – вставить дочерний узел для x ;
- x.removeChild(node) – удалить дочерний узел для x.

**Пример 3.**

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

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

Тот же самый результат может быть получен по-другому:

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

В рамках DOM возможны 3 способа доступа к узлам:

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

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

Пример 4.

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

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

- parentNode ;
- firstChild ;
- lastChild.

Для непосредственного доступа к тэгам можно использовать 2 специальных свойства:

- document.documentElement – для доступа к корневому узлу документа;

- document.body – для доступа к тэгу <body>.

**Свойства узлов**

В HTML DOM каждый узел является объектом, который может иметь методы (функции) и свойства. Наиболее важными являются следующие свойства:

- nodeName ;
- nodeValue ;
- nodeType.

Свойство nodeName указывает на имя узла. Это свойство имеет следующие особенности:

1. Свойство nodeName предназначено только для чтения;
2. Свойство nodeName узла элемента точно соответствует имени тэга;
3. Свойство nodeName узла атрибута соответствует имени атрибута;
4. Свойство nodeName текстового узла всегда равно #text
5. Свойство nodeName узла документа всегда равно #document

Замечание: nodeName всегда содержит имя тэга HTML элемента в верхнем регистре.

Свойство nodeValue указывает на значение узла. Это свойство имеет следующие особенности:

- Свойство nodeValue узла элемента не определено;
- Свойство nodeValue текстового узла указывает на сам текст;
- Свойство nodeValue узла атрибута указывает на значение атрибута.

Свойство nodeType возвращает тип узла. Это свойство предназначено только для чтения.

Наиболее важными типами узлов являются следующие:

| Тип элемента | Тип узла |
|---|---|
| Element | 1 |
| Attribute | 2 |
| Text | 3 |
| Comment | 8 |
| Document | 9 |

**Изменение HTML элементов**

HTML элементы могут быть изменены с посредством использования JavaScript, HTML DOM и событий.

В примере 5 показано, как можно динамически изменять текстовое содержимое тэга <p>:

```
<html>
<body>
<p id="p1">Hello World!</p>
<script type="text/javascript">
      document.getElementById("p1").innerHTML="New text!";
```

```
</script>
</body>
</html>
```
Пример 5.

**Диалоговые элементы**

В JavaScript поддерживается работа со следующими диалоговыми элементами интерфейса:

1. Alert. Применяется для уведомления пользователя, работающего с веб-браузером.

Синтаксис:

```
alert("сообщение");
```

2. Confirm. Применяется для выбора пользователем одного из двух вариантов ответа "Да/Нет".

Соответственно Confirm возвращает значение true/false.

Синтаксис:

```
confirm("вопрос");
```

3. Prompt. Применяется для ввода пользователем значения. При нажатии "OK" возвращается введенное значение, в случае "Cancel" возвращается значение null.

Синтаксис:

```
prompt("вопрос/запрос","значение по умолчанию");
```

Ниже приводится код веб-страницы, в которой пользователь имеет возможность выбрать цвет текста с помощью диалогового элемента

```
<html>
<body>
// здесь будет отображаться текст
<div id="c" style="color:blue">Вы выбрали цвет текста:
черный</div>
<script language="JavaScript">
// пользователь выбирает цвет текста
var tcolor = prompt("Выберитецветтекста: red, blue, green,
yellow,
black","black");
// задаетсятекст
  document.getElementById("c").innerHTML =
"Вывыбралицветтекста: " + tcolor;
// задаетсяцветтекста
  document.getElementById("c").style.color = tcolor;
</script>
</body>
</html>
```
Пример 6.

## Порядок выполнения работы

Используя Microsoft Expression Studio 2 (или другой редактор), подготовьте на основе примеров 1-2, 4-6 соответствующие веб-страницы и просмотрите их с помощью веб-браузера.

Контрольное задание. Создание таблицы случайно выбранных цветов

Взяв за основу сценарий построения таблицы умножения, постройте таблицу случайно выбранных цветов.

Цвет ячейки таблицы задается с помощью атрибута bgcolor. Цвет ячейки описывается в рамках трехкомпонентной модели RGB, например: <td bgcolor="#c0a145">. Для генерации каждой компоненты можно использовать генератор случайных чисел с помощью методов объекта Math и преобразование в шестнадцатеричный формат:

```
color = Math.round(255.0*Math.random());
r = color.toString(16);
```

Результирующий цвет образуется путем конкатенации компонентов:

```
color = r + g + b;
```

Примерный вид результата работы сценария: