

**Министерство образования Республики Беларусь**

**Учреждение образования  
«Гомельский государственный технический  
университет имени П. О. Сухого»**

**Институт повышения квалификации  
и переподготовки кадров**

**Кафедра «Информатика»**

**Т. В. Тихоненко, А. И. Рябченко**

## **ВЕРСТКА WEB-СТРАНИЦ**

**ЛАБОРАТОРНЫЙ ПРАКТИКУМ  
по одноименной дисциплине  
для слушателей специальности 1-40 01 74  
«Web-дизайн и компьютерная графика»  
заочной формы обучения**

**Гомель 2013**

УДК 004.42+004.43(075.8)  
ББК 32.973.26-018.1я73  
Т46

*Рекомендовано кафедрой «Информатика» ГГТУ им. П. О. Сухого  
(протокол № 5 от 25.11.2012 г.)*

Рецензент: д-р техн. наук, проф. каф. «Информационные технологии»  
ГГТУ им. П. О. Сухого *И. А. Мурашко*

**Тихоненко, Т. В.**

Т46 Верстка web-страниц : лаборатор. практикум по одноим. дисциплине для слушателей специальности 1-40 01 74 «Web-дизайн и компьютерная графика» заоч. формы обучения / Т. В. Тихоненко, А. И. Рябченко. – Гомель : ГГТУ им. П. О. Сухого, 2013. – 59 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: <http://library.gstu.by/StartEK/>. – Загл. с титул. экрана.

Представлены задания к лабораторным работам, с помощью которых слушатели приобретают навыки работы с языком гипертекстовой разметки HTML и основами каскадных таблиц стиля CSS. В практикуме прослеживаются три направления: изучение языка HTML, основы CSS и блочная верстка с помощью HTML и CSS.

Все задания к лабораторным работам снабжены предварительными теоретическими сведениями, которые будут полезны для выполнения заданий.

Для слушателей специальности 1-40 01 74 «Web-дизайн и компьютерная графика» заочной формы обучения.

УДК 004.42+004.43(075.8)  
ББК 32.973.26-018.1я73

© Учреждение образования «Гомельский  
государственный технический университет  
имени П. О. Сухого», 2013

## Лабораторная работа № 1 Основные понятия и определения языка html

**Цель работы:** изучить основные теги языка html, получить первоначальные навыки создания Web-страниц средствами html; научиться форматировать текст с помощью языка html.

### Теоретические сведения

#### *Понятие тега. Типы тегов*

Любой язык разметки состоит из двух основных частей:

1. информация, которую нужно передать (текст, который размечается);
2. данные разметки.

Данные разметки можно узнать по угловым скобкам < и >. Текст между угловыми скобками – это не что иное, как разметка HTML-документа. Такого образа конструкции, называются *теги*. Другими словами, тег – это инструкция браузеру, указывающая способ отображения текста.

Теги могут быть двух видов:

- 1) <.> одноэлементные (одионочные). Элементы, образованные при помощи одионочных тегов, называют *пустыми*;
- 2) <...>...</...> парные. Парный тег влияет на текст, с того места, где он употреблен, и до того места, где указан признак окончания его действия. Признаком завершения служит тот же тег, только начинающийся с символа « / » (обратный слеш).

Есть еще специальный вид тега, который в HTML называется *декларация*. Он используется исключительно для служебной информации. Одним из видов декларации является комментарий. Любой HTML-комментарий должен начинаться с конструкции <!-- и заканчиваться -->. Между ними может находиться любой текст, цифры, символы и прочее, за исключением тегов. Вложенность комментариев недопустима.

#### *Структура и правила оформления HTML-документа*

Структура HTML-документа состоит из нескольких правил. Абсолютно в любом языке разметки есть единственный корневой элемент. *Корневым* называют элемент, внутри которого находятся все остальные элементы. В HTML корневым элементом является <html>...</html>.

Языком HTML предусмотрены еще два обязательных раздела: заголовок и содержание.

**<head> ... </head>** - заголовок HTML-программы. Заголовок выполняет функцию рабочего заголовка HTML-документа и является, по сути, «невидимым».

**<body> ... </body>** - содержание HTML-программы. Основное содержание страницы, которое отображается браузером, помещается в тег **<body> ... </body>**. Его также называют телом программы.

Эти три элемента (**html**, **head**, **body**) являются обязательными.

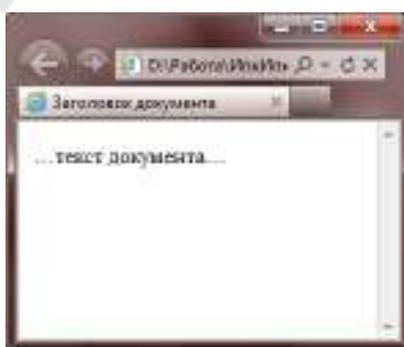
В разделе **head** может прописываться парный тег **<title>...</title>**, предназначенный для указания имени созданного электронного документа.

Для создания HTML-документов можно воспользоваться любым текстовым редактором, например Notepad (Блокнот). Все HTML-файлы следует сохранять с расширением **.htm** или **.html**, что укажет компьютеру, что файл содержит коды HTML.

Таким образом, любой HTML-документ имеет следующую структуру:

```
<html>  
<head>  
    <title>Заголовок документа</title>  
</ head >  
<body>  
    ...текст документа...  
</ body >  
</ html>
```

После создания HTML-файла, его необходимо просмотреть браузером, например Opera, см. рис. 1.1.



*Рисунок 1.1 – Вид HTML-документа в окне браузера Opera*

### **Основные элементы форматирования текста. Элементы блочные и строчные**

Теперь перейдем к изучению тегов, которые прописываются внутри раздела **body**. Все элементы внутри раздела **body** можно

разделить на две основные группы: элементы *строчные* (inline) и *блочные* (block). Строчные элементы используются для оформления кусков текста, а блочные для структуры блоков текста на странице.

Теги *текстовых блоков* `<span>` и `<div>` являются яркими представителями строчных и блочных элементов соответственно.

Метки вида `<hi>` ( $i = 1 \div 6$ ) описывают *заголовки* шести различных уровней. На рис. 1.2 показано, что заголовок `<h6>` будет минимальным, а `<h1>` - самым большим. Справа на рис. 1.3 изображен вид заголовков различных уровней в окне браузера, а слева их HTML-код.

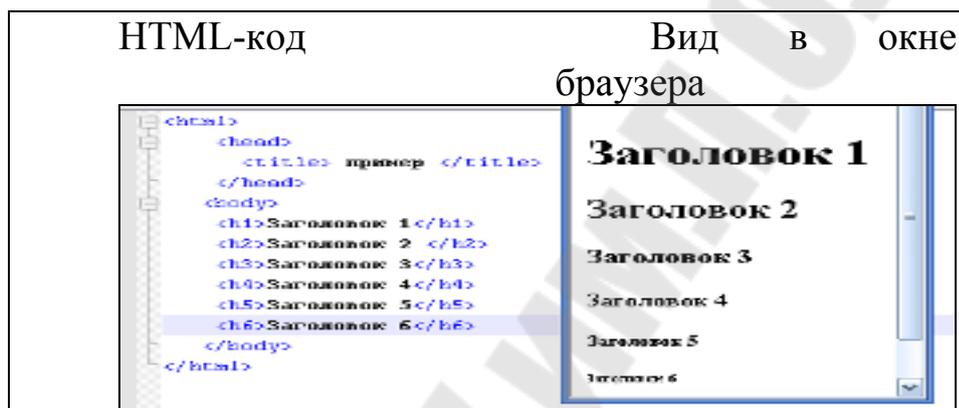


Рисунок 1.2 – Заголовки различных уровней

Элемент `<hr>` (*горизонтальная линия*) является одиночным пустым и блочным. Этот элемент образует полосу и выполняет функцию визуального разделителя фрагментов. Элемент `<hr>`, поскольку он является блочным, занимает по ширине все доступное пространство. На рис. 1.3 приведен пример горизонтальной линии.

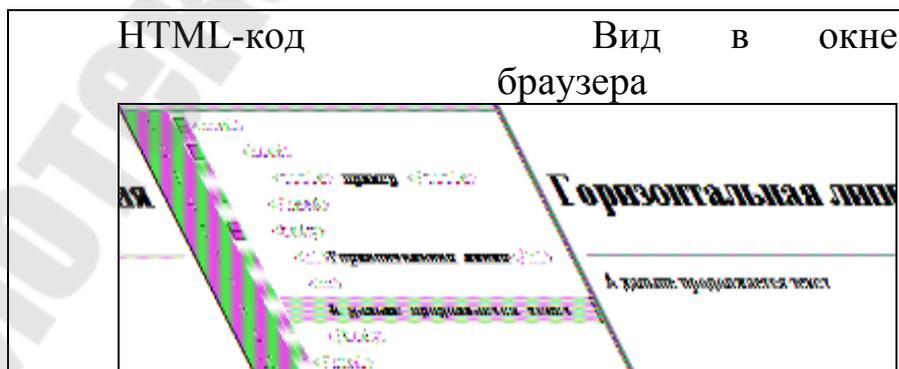


Рисунок 1.3 – Горизонтальная линия

При создании HTML-документа для обозначения *абзаца* (*параграфа*) используется специальный парный тег `<p>..</p>`, который разделяет фрагменты текста вертикальным отступом. Простой перевод

строки в данном случае не поможет: браузер, интерпретируя код, не воспримет отступ как команду создания абзаца, и при выводе содержания на экран монитора объединит фрагменты текста вместе.

Если необходимо перейти на новую строку, не прерывая абзаца, лучше использовать строчный пустой элемент `<br>`. Использование тега `<br>` очень удобно при публикации стихов.

Если вам необходимо отобразить текстовую строку фиксированной длины без переносов, используется парный тег `<nobr>..</nobr>`. При использовании данного тега в случае, если длина строки превышает ширину экрана, в нижней части окна браузера появится горизонтальная полоса прокрутки.

Для авторского форматирования используется блочный элемент `<pre>...</pre>`, внутри которого могут находиться только строчные элементы. В этом случае все, что расположено между тегами `<pre>...</pre>` отобразится в окне браузера с сохранением всех введенных разработчиком пробелов, переводов строк и отступов.

Все элементы непосредственного форматирования текста делятся на две категории: *элементы логического форматирования* и *элементы визуального форматирования*.

К элементам визуального оформления относятся теги, представленные на рис.1.4.

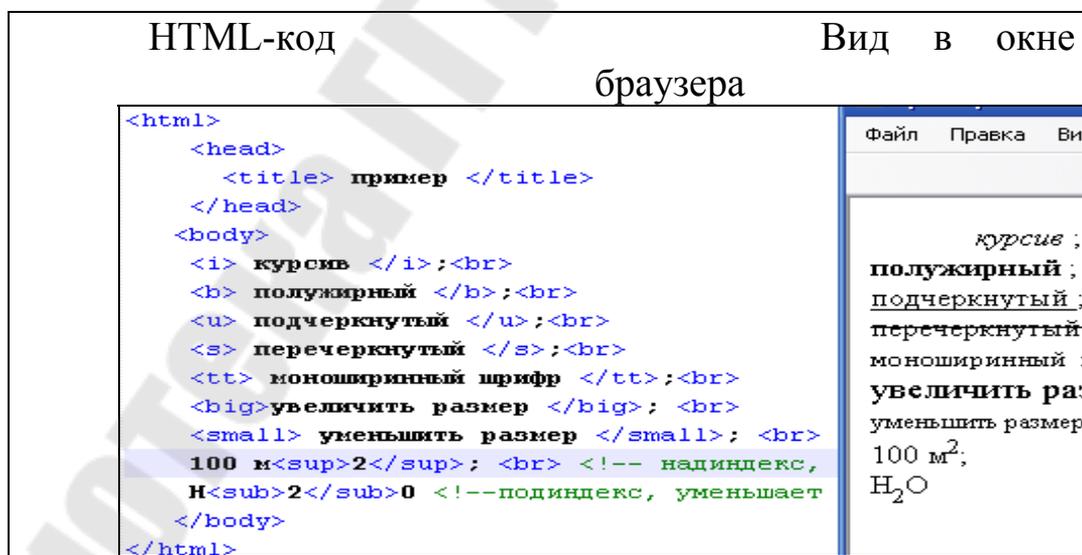


Рисунок 1.4 – Элементы визуального форматирования

Данные пары тегов можно вкладывать друг в друга. Однако, необходимо соблюдать порядок вложенности тегов. Например, полужирный курсив будет выглядеть так: `<b><i>Полужирный курсив</i></b>`.

Все элементы визуального форматирования, кроме `<i>` и `<b>`, не вошли в HTML5.

К элементам логического форматирования относятся следующие элементы:

`<em>` – выделение важных фрагментов текста *курсивом*.

`<strong>` – выделение наиболее важных фрагментов **полужирным**.

Рекомендуется заменять `<i>` на `<em>`, а `<b>` на `<strong>`.

`<ins>` – выделение фрагмента подчеркиванием, когда требуется визуально показать, что текст был вставлен после опубликования документа.

`<del>` – выделение фрагмента перечеркиванием, когда требуется визуально показать, что текст был удален после опубликования документа.

Элементы `<ins>`, `<del>` могут выступать и как строчные, и как блочные элементы.

`<cite>` – выделение цитат *курсивом*.

`<code>` – отображение фрагментов программного кода моноширинным шрифтом.

`<kbd>` – текст, вводимый с клавиатуры: отображается моноширинным шрифтом.

`<var>` – название переменных: отображается *курсивом*.

`<samp>` – выделение нескольких символов моноширинным шрифтом.

`<dfn>` – определение вложенного термина *курсивом*.

Все элементы логического форматирования вошли в HTML5.

**Важно!** Чтобы лучше ориентироваться в большом коде, его нужно правильно и грамотно форматировать. Открывающие и закрывающие теги должны стоять полвертикали на одном уровне. Внутренние элементы можно сдвигаем с помощью табуляции. Правильное форматирование кода может помочь увидеть, где допущена ошибка, а также снизить процент возможных ошибок.

### ***Понятие элементов и атрибутов***

Один и тот же тег может интерпретироваться браузером по-разному. Это зависит дополнительных параметров. Эти параметры называются *атрибутами (параметрами) тега*. Атрибуты уточняют действие тегов. Каждый тег имеет свой набор допустимых атрибутов. Атрибуты указываются только в открывающем теге. Для одного тега можно использовать несколько атрибутов, разделенных пробелами. Отметим, что имена атрибутов заранее определены, а вот значение атрибуту присваивается заданное или произвольное.

Синтаксис записи тега и соответствующими атрибутами следующий:

`<тег имя_атрибута_1= "значение" ...  
имя_атрибута_n="значение">`

Например, `<hr align="left" width="50%" size=4>`.

Все атрибуты можно разделить на три основные группы:

1. общие. К ним относятся следующие атрибуты: `id`, `class`, `lang`, `dir`, `title`, `style`. Общие атрибуты могут стоять практически у подавляющего большинства элементов;

2. атрибуты событий. Эта группа атрибутов нужна для того, чтобы вызвать действие в ответ на действие пользователя или на какие-то системные события;

3. уникальные. К этой группе относятся индивидуальные атрибуты некоторых элементов.

Корневой элемент `<html>` атрибутов не имеет.

Рассмотрим некоторые атрибуты для изученных элементов.

Атрибут **align** может применяться только для блочных элементов и имеет следующие значения: `left`, `right`, `center`, `justify`.

Для заголовков `<h1> ... </h1>` (`i` – цифра от 1 до 6) атрибут `align` означает, что его содержимое будет выравниваться по левому краю при `align="left"` (по умолчанию), по правому краю при `align="right"`, все строки будут по центру при `align="center"`, выравнивание произойдет как по левому, так и по правому краю при `align="justify"`. Для маленьких кусков текста `justify` не имеет смысла.

У пустого элемента `<hr>` есть 4 атрибута: `align`, `noshade`, `size`, `width`, `color`.

Для тега `<hr>` атрибут `align` означает выравнивание самой горизонтальной линии и принимает следующие значения: `right`, `left`, `center`.

Выравнивание для элемента `<hr>` имеет смысл только тогда, когда прописан атрибут **width** с заданным значением. Значение атрибута `width` может быть задано в пикселях или процентах. Например, тег `<hr width=15>` определяет горизонтальную линию в 15 пикселей. Если значение атрибута `width` задано в процентах, то ширина линии вычисляется относительно ширины окна браузера. Например, `<hr width="40%">`. По умолчанию, значение атрибута `width` имеет 100%.

Атрибут **size** задает высоту горизонтальной линии в пикселях. Значение данного атрибута может варьировать от 1 до 175. По умолчанию, это значение равно `2px`.

Атрибут **noshade** (не отбрасывать тень) является одиночным и не имеет никакого значения. В разных браузерах использование этого атрибута может привести к различным результатам.

Атрибут **color** определяет цвет горизонтальной линии.

Значение цвета в атрибутах языка HTML может задаваться несколькими способами.

– по шестнадцатеричному значению

Шестнадцатеричная система базируется на числе 16. Цифры будут следующие: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Например, числу 255 в десятичной системе соответствует число FF в шестнадцатеричной системе. Чтобы не возникало путаницы в определении системы счисления, перед шестнадцатеричным числом ставят символ решетки #, например #666999. Каждый из трех цветов — красный, зеленый и синий — может принимать значения от 00 до FF. Таким образом, обозначение цвета разбивается на три составляющие #rrggbb, где первые два символа отмечают красную компоненту цвета, два средних — зеленую, а два последних — синюю. Допускается использовать сокращенную форму вида #rgb, где каждый символ следует удваивать (#rrggbb). К примеру, запись #fe0 расценивается как #ffee00.

– по названию

Браузеры поддерживают некоторые цвета по их названию. В таблице 1.1 приведен шестнадцатеричный код, название и описание. Более полную таблицу цветов можно найти в книге [2].

Таблица 1.1 – 15 стандартных цветов

Код	Название	Описание
#000000	ЧЕРНЫЙ	black
#FFFFFF	БЕЛЫЙ	white
#FF0000	КРАСНЫЙ	red
#008000	ЗЕЛЕНый	green
#0000FF	СИНИЙ	blue
#808000	ОЛИВКОВЫЙ	olive
#FFFF00	ЖЕЛТЫЙ	Yellow
#C0C0C0	СЕРЕБРИСТЫЙ	Silver
#808080	СЕРЫЙ	Gray
#FF00FF	ФУКСИНОВЫЙ	Fuchsia
#000080	УЛЬТРАМАРИН	Navy
#008080	СИЗЫЙ	Teal
#00FF00	СВЕТЛО-ЗЕЛЕНый	Lime
#800080	ПУРПУРНЫЙ	Purple
#800000	КАШТАНОВЫЙ	maroon

– с помощью RGB

Можно определить цвет, используя значения красной, зеленой и синей составляющей в десятичном исчислении. Значение каждого из трех цветов может принимать значения от 0 до 255. Также можно задавать цвет в процентном отношении. Вначале указывается ключевое

слово `rgb`, а затем в скобках, через запятую указываются компоненты цвета, например `rgb(255, 0, 0)` или `rgb(100%, 20%, 20%)`.

Для вставки цвета фона в строку с тегом `<body>` нужно добавить атрибут **bgcolor** и указать его значение – название цвета или его шестнадцатеричный вид. Фон страницы станет красным, если прописать одну из следующих строк.

```
<body bgcolor="red"> <!-- использовано название цвета-->  
<body bgcolor="#FF0000"> <!--использован шестнадцатеричный вид  
цвета-->
```

Для задания цвета текста в HTML есть специальный атрибут **text**, который может применяться только в теге `<body>`.

Тег `<body bgcolor="silver" text="red">` задаст цвет фона HTML-страницы серым, а цвет текста красным.

Для определения свойства шрифта в HTML есть специальный элемент `<font>...</font>`. Этот элемент имеет 3 атрибута: `color`, `size`, `face`.

Синтаксис элемента `font` следующий:

```
<font color="значение_1" size="значение_2" face="значение_3"> любой текст </font>
```

Чтобы написать текст красным цветом, необходимо вставить перед ним парный тег `<font>...</font>` и присвоить его атрибуту **color** значение `red` или `#FF0000`. Например, `<font color="red">Красный текст</font>` или `<font color="#FF0000">Красный текст</font>`.

Вторым атрибутом элемента `<font>...</font>` является размер шрифта – **size**. Параметр этого атрибута может быть описан либо абсолютной, либо относительной величиной. Абсолютная величина подразумевает использование в качестве параметра целого числа, указывающего на высоту шрифта в пунктах. Относительная же величина, обозначаемая целым числом со знаком плюс или минус (например, `+2` или `-1`), – это количество пунктов, которые следует прибавить или отнять от размера шрифта, используемого браузером (по умолчанию `12pt`). Так, запись `<font size="+1">` говорит о том, что размер помеченного таким образом текста будет на один пункт больше, чем обычный текст документа. Кроме того, размер шрифта можно записывать целыми числами от 1 до 7 (самый мелкий шрифт имеет размер 1, а самый большой – 7; `size="3"` – значение шрифта по умолчанию).

С помощью атрибута **face** тега `<font>` можно менять стиль написания (гарнитуру) шрифта, например:

```
<font face="Times New Roman"> Текст с начертанием Times New Roman </font>
```

Существует 5 основных семейств шрифтов:

1. Шрифты с засечками (семейство serif). Например, Georgia, Times New Roman, Palatino и др.;
2. Рубленые шрифты или шрифты без засечек (семейство sans-serif). Например, Arial, Helvetica, Verdana, Tahoma и др.;
3. Моноширинные шрифты (семейство monospace). Например, Courier, Courier New, Lucida Sans, Consolas и др.;
4. Fantasy;
5. Курсив (italic).

### **Специальные символы языка HTML**

К специальным символам HTML относятся символы, не входящие в состав стандартных ASCII-кодов. Их реализация в HTML-документах возможна при помощи отдельных кодовых конструкций или числовых комбинаций.

Условно все специальные символы HTML можно разделить на три большие группы:

1. *Символы, отображающие элементы HTML-форматирования.* Часто бывает необходимо отобразить в браузере символы, используемые в спецификации языка HTML: теги (< >), знак амперсанда (&) и т. д. Если необходимо в браузере отобразить сам код, нужно все левые угловые скобки в коде заменить на *&lt;*, а правые — на *&gt;*.

2. *Символы оформления документа.* К так называемым символам оформления HTML-документов относятся значки авторского права, зарегистрированной торговой марки, символы иностранных валют, математические обозначения («плюс-минус», «умножить» и др.) и пр. Приведенный на рис. 1.5 пример, может быть получен за счет вставки в код документа следующих конструкций:

HTML-код	Вид в окне браузера
<code>&amp;pound;</code> - Фунт стерлингов; <code>&lt;br&gt;</code>	£ - Фунт стерлингов;
<code>&amp;reg;</code> - Зарегистрированная торговая марка; <code>&lt;br&gt;</code>	® - Зарегистрированная торговая марка;
<code>&amp;plusminus;</code> - Знак «плюс-минус»	± - Знак «плюс-минус»

*Рисунок 1.5– Некоторые символы оформления HTML-документов*

3. *Буквы иностранных алфавитов.* Когда создается HTML-документ на русском или английском языке, никаких затруднений с отображением букв в браузере не возникает. Трудности появляются, если необходимо разработать электронный документ на языке, алфавит

которого содержит некоторые буквы, чье начертание характерно только лишь для данного языка (например, португальский).

Выйти из такого положения можно несколькими способами.

- a) добавить в раскладку клавиатуры новый язык.
- b) прибегнуть к услугам визуальных web-редакторов, которые включают мультиязыковую поддержку.
- c) использовать специальные кодовые или числовые конструкции, предназначенные для корректной визуализации нестандартных букв иностранного алфавита.

Полный перечень всех специальных символов, а также их кодовые и числовые конструкции можно найти в [2].

### **Практическое задание**

#### **Задание 1.**

Скопируйте нижеприведенный текст в Notepad++. Оформите код HTML согласно предписаниям в комментариях, добавив все необходимые разделы. Прделанную работу покажите преподавателю.

```
Мой первый документ <!-- это должно быть названием HTML страницы -->
<!-- здесь должна находится горизонтальная линия -->
Мой первый HTML-документ <!-- Этот текст сделать заголовком первого уровня-->
Евгений Онегин <!-- заголовок второго уровня -->
А.С. Пушкин (отрывок) <!-- заголовок третьего уровня -->
<!-- следующие четыре строчки поместить внутрь параграфа,
каждая строчка должна начинаться с новой строки -->
Мой дядя самых честных правил,
Когда не в шутку занемог,
Он уважать себя заставил
И лучше выдумать не мог.
```

#### **Задание 2.**

В папке mod1 откройте с помощью программы notepad++ файл lab1-1a.html. Выполните действия, описанные в комментариях. Результаты работы покажите преподавателю.

#### **Задание 3.**

Откройте файл reklama.gif и создайте аналогичную html-страницу.

### **Вопросы для контроля по лабораторной работе № 1**

1. Опишите структуру тега. Назовите виды тегов. Приведите примеры.
2. Какова структура HTML-документа?
3. Опишите структурные теги (<html> , <head> , <title>, <body>).

4. Дайте характеристику и назовите атрибуты тега <body>.
5. Каков порядок создания HTML-документа?
6. Дайте характеристику тегов <h1>...</h1> ÷ <h6>...</h6>, <nobr> ... </nobr> и <p>...</p> .
7. Дайте характеристику тегов <br> , <hr>, <pre>...</pre>.
8. Назовите элементы визуального и логического форматирования. Какие из них вошли в спецификацию HTML5?
9. Дайте характеристику тега <font>...</font>.
10. Опишите три основные категории специальных символов.

## Лабораторная работа № 2

### Создание и использование гиперссылок. Карты изображений

**Цель работы:** научиться создавать внешние и внутренние ссылки, якоря. Освоить принципы создания картинок-ссылок. Научиться создавать и использовать карты изображений.

#### Теоретические сведения

##### *Элемент <a>...</a> и его атрибуты*

С помощью ссылок осуществляется связь текста или картинки с другими гипертекстовыми документами. Текст ссылки, как правило, выделяется цветом и (или) оформляется подчеркиванием. Ссылка создается с помощью тега <a>...</a> и его атрибутов и имеет следующую конструкцию:

**<a href="URL" target="Окно" title="Подсказка">Название ссылки</a>**

Элемент <a>...</a> имеет следующие атрибуты:

**href** – принимает значение URL (Uniform Resource Locator, унифицированный локатор ресурсов) – адрес любого файла в Интернете. Может быть абсолютными, то есть указывается полный адрес странички (например, <http://tut.by/index.html>) и относительным, указывается файл относительно текущего (например, [index.html](#)).

**target** – определяет, в каком окне загрузить гиперссылку. Может иметь значения:

**\_top** – загружает гиперссылку на всем пространстве окна браузера (если до этого существовало разбиение на фреймы, то оно исчезнет).

**\_blank** – загружает гиперссылку в новом окне браузера.

**\_self** – загружает содержимое страницы, в окно, которое содержит эту ссылку (используется по умолчанию).

*\_parent* – загружает содержимое страницы, заданной ссылкой, в окно, являющееся непосредственным владельцем набора фреймов.

*title* – задает текст подсказки, который будет появляться при наведении мышки на гиперссылку. Параметр не обязательный.

Для примера создадим ссылку в документе menu.html на заранее созданный документ index.html (см. пример 2.1). Предполагается, что оба документа находятся в одной папке.

**Пример 2.1** – Пример создание ссылки

```
<p align="center">  
<a href="index.html" target="self" title="Пример ссылки"> Ссылка </a>  
</p>
```

Щелкнув на ссылку, откроется другой документ, в данном случае index.html.

Вы можете также создать ссылку на e-mail, в это случае нужно прописать следующее:

```
<a href="mailto:po4ta@tut.by">po4ta@tut.by</a>
```

Почтовая гиперссылка имеет несколько параметров (не обязательных):

&subject - тема письма;

&body - текст вашего сообщения;

&cc - копии письма через запятую;

&bcc - скрытые копии письма через запятую;

Атрибут title="Выпадающая подсказка" ставиться по желанию и располагается отдельно от параметров почтовой ссылки:

```
<a href="mailto:po4ta@tut.by &subject=Письмо &body=text  
&cc=copy@tut.by &bcc=hidden_copy  
@tut.by" title="Пример почтовой гиперссылки">Почта</a>
```

Иногда возникает непобедимость сделать *ссылку на определенное место* в том же или в другом документе. Чтобы нажав по какой-нибудь ссылке можно было попасть в определенное место данного документа, необходимо создать *закладки*.

Ссылка на закладку в том же документе имеет следующий вид:

```
<a href="#Имя закладки">Название раздела</a>.
```

А так выглядит ссылка на закладку в другом документе:

```
<a href="Имя документа#Имя закладки">Название раздела</a>
```

Сама закладка будет такой:

```
<a name="Имя закладки"> ...текст... </a> или  
<тег id="Уникальное имя закладки">...текст...</тег>
```

Щелкнув на фразу «Название раздела» пользователь будет попадать на определенную вами закладку.

Для того, чтобы при наведении на ссылку курсором и при клике она меняла свой цвет, в тег `<body>` нужно добавить еще несколько параметров: `text` – цвет текста, `link` – цвет ссылки, `vlink` – цвет пройденной ссылки, `alink` – цвет активной ссылки, когда подводится к ней курсор.

Например,

```
<body text="black" link="blue" vlink="purple" alink="red">
```

Данные атрибуты определяют свойства для всего документа. Поместив такой код в HTML-документ, уже не надо будет назначать каждый раз цвет текста и цвет ссылок, т.к. везде он будет таким, каким определен в теге `<body>`.

### ***Размещение иллюстраций на web-странице. Типы файлов иллюстраций***

На данный момент практически все браузеры широко поддерживают три формата: GIF, JPEG, PNG.

**Формат GIF** (Graphic Interchange Format, формат обмена графикой) является первым форматом, который появился для обмена медийной информацией. Изображение, кодированное в формат GIF, всегда содержит от 2 до 256 цветов. Такая палитра цветов называется *индексированная* или *фиксированная*.

**Формат JPEG** (Join Photographic Experts Group, объединенная группа экспертов в области фотографии) – это графический стандарт, созданный на основе одноименного алгоритма сжатия, изображений с потерей качества, кодирующего не идентичные элементы, а межпиксельные интервалы. Особенностью формата JPEG является то, что он всегда искажает картинку.

**Формат PNG** (Portable Network Graphics, портативная сетевая графика) – это формат, который поддерживает «черезстрочность» не только по горизонтали, но и по вертикали. Данный формат имеет три разновидности *PNG-8*, *PNG-24*, *PNG-32*. Данный формат есть в описании на сайте консорциума <http://www.w3.org/Graphics/PNG/>.

### ***Элемент IMG и его атрибуты***

Картинку можно вставить как объект на web-странице, что осуществляется с помощью элемента `<img>`. Чтобы показать (загрузить) картинку необходимо прописать следующую строку:

```

```

Например, `` или ``, или ``.

Далее, браузер при загрузке картинки определяет ее истинные размеры, на которые и «раздвигает» выделяемую строчную область.

Если необходимо, чтобы браузер сразу выделял область нужного размера, то указывают атрибуты `width` и `height`.

Например, ``

Изображение можно делать ссылкой. Для этого вместо названия ссылки нужно прописать графический элемент (См. пример 2.2). В этом случае вокруг изображения появится рамка в 3 пикселя (по умолчанию), цвета, определенного для гиперссылок.

### Пример 2.2.

```
<a href="URL" target="окно">  
  
</a>
```

В примере 2.2. атрибут `title` может быть только у рисунка.

Атрибут `alt` нужен для альтернативного представления, а `title` для подсказок.

Рассмотрим вопрос расположения изображений относительно текста. На рис. 2.1 представлен самый простой случай, при котором рядом с рисунком располагается только одна строка текста.



Рисунок 2.1 – Расположение изображений относительно текста

Как же сделать так, чтобы текст располагался весь рядом с изображением, а не только одна его строка? В этом случае нам поможет атрибут `align`: `` - это означает, что картинка будет прижата к левому краю экрана, а текст будет обтекать ее справа. Чтобы сделать наоборот (картинка справа, текст слева) надо прописать: ``.

Атрибут `align` у картинок может принимать и другие значения. Текст может располагаться внизу картинки (по умолчанию) - ``, посередине - ``, и вверху - ``.

Кроме параметра `align` существует еще несколько атрибутов:

- атрибут `vspace`. Например, `` - задает расстояние между текстом и рисунком (по вертикали).

- параметр *hspace*. Например, `` - тоже задает расстояние между текстом и рисунок (по горизонтали).

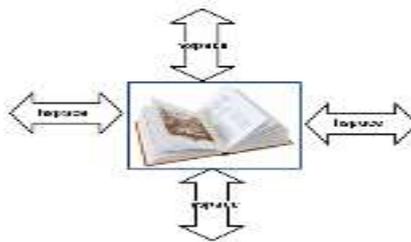


Рисунок 2.2 – Графическое представление атрибутов *hspace* и *vspace*

Изображение можно использовать не только как элементы, но и как заливку в качестве фона. Для применения картинок в качестве фона используются специальный атрибут `background` элемента `body`:

`<body background =”путь_к_файлу”>`.

Заливка фоновой картинкой начинается с левого верхнего угла, и начинает повторяться по двум осям, по оси *x* и по *y*.

### **Навигационные карты**

Уже говорилось об изображениях и о том, как изображение сделать ссылкой. Пользователи интернета знают, что можно сделать так, чтобы при нажатии на разные области (части) одного и того же изображения, можно было попадать на разные страницы, это называется – *навигационная карта*.

Навигационные карты задаются элементом `<map>...</map>`. Тэг `<map>` включает в себя тэг(и) `<area>`, которые определяют геометрические области внутри карты и ссылки, связанные с каждой областью (т.е. то куда вы попадете после нажатия). Конструкция элементов, при создании кликабельных областей может быть следующей:

```
<map>
  <area ...>
  <area ...>
  ...
  <area ...>
</map>
```

Кликабельные области бывают трех различных форм:

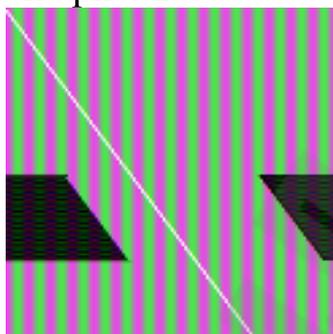
**Rect** – прямоугольная форма, координаты задаются в пикселях и записываются через запятую *x1, y1, x2, y2*. (*x1,y1*) – координаты левого верхнего угла изображения и (*x2,y2*) – правый нижний угол.

**Circle** – форма окружности, через запятую задаются координаты центра окружности (x,y) и величина радиуса R.

**Poly** – многоугольник, координаты задаются последовательно для каждой точки. Заканчиваются той же точкой из которой вышли.

Рассмотрим эти области подробнее.

*Прямоугольники.* Для начала нам нужно изображение. Возьмем изображение, представленное на рис. 2.3.



*Рисунок 2.3 – Прямоугольная клиткабельная область*

Картинка простая - всего лишь серый квадрат. Представьте, что черный прямоугольник, нарисованный на нем, - это изображение какой-то кнопки, а серое - какой-то сложный фон. Чтобы не создавать сложную таблицу, вы решили не резать картинку на много частей и не вычленять кнопки, Вы решили поступить проще - создать навигационную карту, где область прямоугольника будет ссылкой.

Итак, геометрические области и то, куда пользователь попадет при нажатии на них, определяет тэг `<area>`, естественно, при помощи своих параметров. Это параметры `shape` и `coords`.

Параметр `shape` - определяет форму области (будет ли она прямоугольником (`shape="rect"`), кругом (`shape="circle"`) или многоугольником (`shape="poly"`)). Сейчас мы работаем с прямоугольной областью, поэтому:

```
<map>  
<area shape="rect">  
</map>
```

Параметр `coords` - определяет координаты (положение нашей геометрической формы). Число координат и порядок их значений зависят от выбранной нами формы (см. рисунок 2.4).

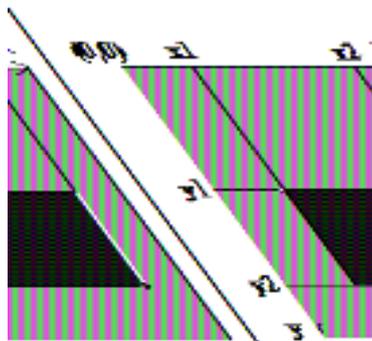


Рисунок 2.4 – Определение координат прямоугольной области

Отсчет ведется от левого верхнего угла картинке, считайте его началом координат (0;0). Если работаем с прямоугольной областью, то нужны координаты верхнего-левого и нижнего-правого углов области. Порядок записи координат для параметра coords следующий: `<area shape="rect" coords="x1,y1,x2,y2">`.

В нашем примере у прямоугольника координаты такие:  $x1=83$ ,  $y1=122$ ,  $x2=177$ ,  $y2=185$ .

Значит, код будет выглядеть следующим образом:

```
<map>
<area shape="rect" coords="83,122,177,185">
</map>
```

Теперь пропишем, куда будет ссылаться наша область, для этого нам понадобится уже знакомый параметр

href:

```
<map>
<area href="drugoy_document.html" shape="rect"
coords="83,122,177,185">
</map>
```

Однако этого все еще не достаточно, чтобы картинка стала ссылкой, нужно еще указать имя карты и связать ее с картинкой.

У тэга `<map>` есть параметр `name` - имя карты, назовем карту - `karta1`:

```
<map name="karta1">
<area href="drugoy_document.html" shape="rect"
coords="83,122,177,185">
</map>
```

Для того, чтобы связать карту с картинкой, надо использовать атрибут `usemap="#имя_карты"` для картинки:

```


...текст...
<map name="karta1">
  <area href="drugoy_document.html" shape="rect"
coords="83,122,177,185">
</map>

```

*Круги.* Для создания круглой области нужны будут координаты ее центра (x, y) и длина радиуса R в пикселях. Порядок записи следующий: <area shape="circle" coords="x, y, R">.

Работать будем с геометрической областью, изображенной на рис. 2.5.

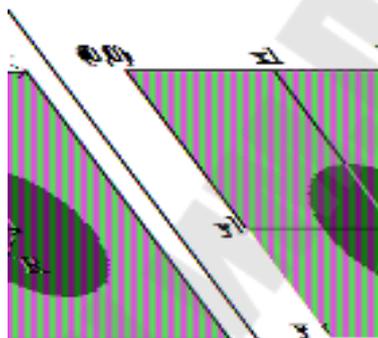


Рисунок 2.5 – Округлая клиткабельная область

В нашем случае координаты для круга будут такие: x=46, y=48; а длина радиуса - R=35. Запишем:

```

<map>
<area shape="circle" coords="46, 48, 35">
</map>

```

Теперь, когда самое главное записано, пропишем имя карты, куда она ссылается и привяжем карту к рисунку:

```


...текст....
<map name="karta2">
<area href="drugoy_document_2.html" shape="circle" coords="46, 48,
35">
</map>

```

Для карты можно прописать alt для каждой области:

```


...текст....
<map name="karta2">
  <area href="drugoy_document_2.html" shape="circle" coords="46, 48,
35" alt="круг">

```

</map>

Теперь при наведении на область будет всплывать подсказка.

*Многоугольники.* Указывая точки (координаты углов), они как бы соединяются, и можно получить очень разнообразные фигуры (рис. 2.6). Используя значение атрибута **shape** poly, можно делать самые разнообразные области, от скромного треугольника до шикарной звезды.

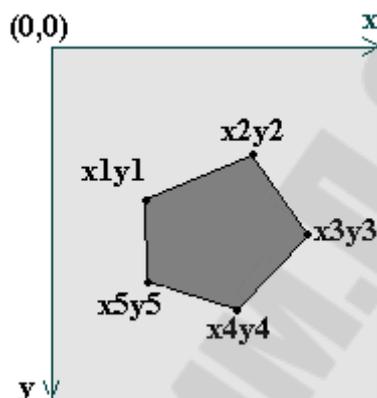


Рисунок 2.6 – Определение координат многоугольной области

Зададим тип области:

```
<map>
```

```
<area shape="poly">
```

```
</map>
```

Координаты пишутся по следующему принципу: `<area shape="poly" coords="x1,y1,x2,y2,...,xN,yN">`.

Расшифровывается это так: "координаты первого угла (x1,y1), координаты второго угла (x2,y2), еще много углов и их координат (...), координаты последнего угла (xN,yN)". Т.е. для пятиугольника запись полностью будет выглядеть так: `<area shape="poly" coords="x1,y1,x2,y2,x3,y3,x4,y4,x5,y5">`.

Теперь подставим реальные значения координат в код:

```
<map>
```

```
<area shape="poly" coords="168,9,232,29,200,97,223,129,153,119">
```

```
</map>
```

Дальше уже прописываем ссылку, имя карты, и привязываем карту к рисунку (это везде по одному и тому же принципу):

```

```

```
...текст...
```

```
<map name="karta3">
```

```

<area href="drugoy_document_3.html" shape="poly"
coords="168,9,232,29,200,97,223,129,153,119">
</map>

```

При создании кликабельных областей могут возникнуть некоторые нюансы:

1. Можно одновременно использовать разные области, например круг и многоугольник:

```


...текст...
<map name="karta3">
<area href="drugoy_document_3.html" shape="circle"
coords="46,48,35,">
<area href="drugoy_document_3.html" shape="poly" cords = "168, 9,
232, 29, 200, 97, 223, 129, 153, 119" >
</map>

```

2. Области могут пересекаться. В этом случае при нажатии на область пересечения приоритет имеет область, которая указана первой (т.е. пользователь пойдет на страницу, куда она ссылается). Используя две перекрывающиеся окружности можно создать фигуру, изображенную на рисунке 2.8.

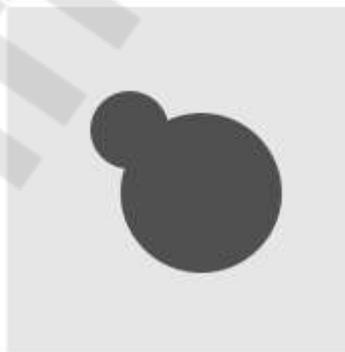


Рисунок 2.7 – Перекрывающиеся окружности

```


<map name="my_map4">
<area href="index.html" shape="circle" coords="111, 107, 40"
alt="Две окружности" />
<area href="index.html" shape="circle" coords="75, 76, 20"
alt="Две окружности" />
</map>

```

3. Что бы не мучиться с картами, можно найти на просторах интернета специальную программу, с помощью которой без труда

можно расчертить карту и не прописывать все вручную. Это может сэкономить время.

4. Существует еще одна область – default – оставшаяся часть, которая не попала в области, описанные с помощью элемента <area>, но тоже может стать кликабельной. Атрибут default прописывается в конце всех элементов <area> и не поддерживается в Internet Explorer.

5. Иногда возникнет ситуация, когда необходимо вычесть одну область из другой или, другими словами, сделать в области «отверстие».



*Рисунок 2.9 – Некликабельная круглая область в центре*

Элемент `nohref` позволяет сделать часть области некликабельной.

Например, сделаем кольцо внутри, которого будет некликабельная область.

```
<area shape="circle" coords="50, 30, 15" nohref>  
<area shape="circle" coords="50, 30, 35" href=http://yandex.ru  
title="Привет">
```

### **Практическое задание**

#### **Задание 1**

- a) Откройте папку lab 2-1, а в ней с помощью программы notepad++ откройте файл lab 2-1.html. Создать оглавление, с каждого пункта которого осуществляется переход к нужной статье. Оформите html-документ по всем правилам оформления html-кода. Оглавление html-страницы, должно содержать не менее 5 строк, которые необходимо сделать гиперссылками на соответствующие разделы. После нажатия на гиперссылку, текст документа должен прокручиваться таким образом, чтобы соответствующий текст раздела отображался максимально высоко к верхнему краю окна области просмотра.
- b) Откройте папку lab 2-2, а в ней файл index.html. В index.html оформите меню в виде гиперссылок на заранее созданные файлы, соответствующие названиям пунктов меню. Оформите html-документ.

- с) Откройте папку lab 2-3 и создайте файл menu.html в папку lab 2-3.

Скопируйте содержимое lab 2-2\index.html в menu.html. В файле menu.html оформите меню в виде гиперссылок на файл content.html с показом статьи, соответствующей названиям пункта меню.

Например:

```
<a href="../lab 2-1/lab 2-1.html#p1">Звезды</a><br>
```

### **Задание 2.**

а) Вставьте изображения на web-страницу. В качестве изображений могут быть выбраны любые картинки. Не добавляйте на страницы слишком больших изображений. Используйте различные варианты обтекания картинки текстом.

1. Создайте изображение в качестве ссылки.

2. Создайте фоновой рисунок

б) Сделать рисунки (таблица 1) картой ссылок, каждая часть которого является ссылкой на соответствующий документ или раздел документа.

1. Используйте файл menu.html из папки lab 2-3.

2. Заменить меню из файла menu.html на карту ссылок.

Например,



### **Вопросы для контроля по лабораторной работе № 2**

1. Опишите атрибуты элемента `<a>...</a>`.
2. Как сделать картинку ссылкой?
3. Какие Вы знаете типы файлов иллюстраций.
4. Опишите атрибуты элемента `<img>`.
5. Что такое навигационные карты и для чего они используются?
6. Какие теги используются для создания кликабельных областей?
7. Опишите основные виды кликабельных областей.
8. Для чего нужны элементы `default` и `noreferrer`?

## Лабораторная работа № 3 Создание списков

**Цель работы:** научиться создавать нумерованные и маркированные списки, а также списки определений с помощью средств HTML.

### Теоретические сведения

#### *Типы списков*

Язык HTML предлагает несколько механизмов создания списков. В каждом списке должен быть один или несколько элементов списков. Списки могут содержать:

- неупорядоченную информацию;
- упорядоченную информацию;
- определения.

*Неупорядоченные списки* создаются с помощью элемента `<ul>...</ul>`, внутри которого может находиться не менее одного элемента `<li>...</li>`:

```
<ul>  
<li>первый элемент;</li>  
<li>второй элемент;</li>  
<li>третий элемент.</li>  
</ul>
```

Для того, чтобы управлять внешним видом маркера, необходимо применить атрибут **type** со значением “circle” или “square”.

Например, `<ul type="circle">` - маркер в виде окружности,

`<li type="square">` - маркер в виде квадрата.

На рис. 3.1 приведен пример HTML-кода неупорядоченного списка с разными видами маркера.

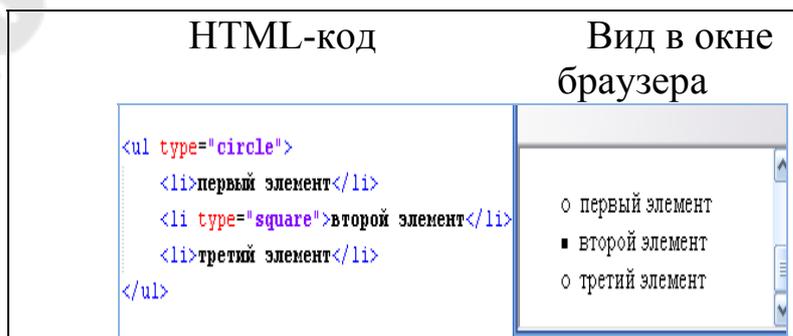


Рисунок 3.1 – Неупорядоченный список

Упорядоченный список, создаваемый с помощью элемента `<ol>...</ol>`, может содержать информацию, в которой важен порядок, например, рецепт:

1. Тщательно смешать сухие ингредиенты.
2. Влить жидкость.
3. Смешивать 10 минут.
4. Выпекать в течение часа при температуре 300 градусов.

Реализация в HTML упорядоченного списка представлена на рисунке 3.2.

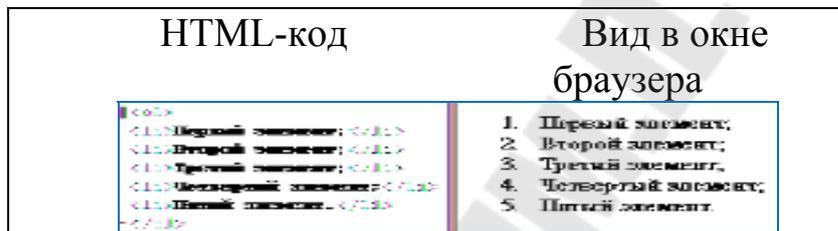


Рисунок 3.2 – Упорядоченный список

Если необходимо начать нумерацию не с первого номера, а, например, с 5, то используйте атрибут **start** со значением “5” (рис.3.3).



Рисунок 3.3 – Упорядоченные списки

Для управления внешним видом нумерованного списка, необходимо применить атрибут **type** со значениями, приведенными в таблице 3.1.

Таблица 3.1- Типы нумераций упорядоченного списка

Начальный тег	Вид номера на экране
<code>&lt;ol&gt;</code>	Нумерация выполняется арабскими цифрами (1, 2, 3, ...)
<code>&lt;ol type="1"&gt;</code>	Нумерация выполняется арабскими цифрами (1, 2, 3, ...)
<code>&lt;ol type="A"&gt;</code>	Нумерация выполняется прописными буквами (A, B, C, ...)
<code>&lt;ol type="a"&gt;</code>	Нумерация выполняется строчными буквами (a, b, c, ...)

<code>&lt;ol type="I"&gt;</code>	Нумерация выполняется большими римскими цифрами (I, II, III, ...)
<code>&lt;ol type="i"&gt;</code>	Нумерация выполняется малыми римскими цифрами (i, ii, iii, ...)

*Списки определений*, создаваемые с помощью элемента `<dl>...</dl>`, могут содержать ряд пар термин/определение (хотя списки определений могут иметь и иные применения). Определение состоит из термина, определенного тегом `<dt>...</dt>`, и его описания, определенного тегом `<dd>...</dd>`.

Все эти элементы – блочные. Внутри элемента `dt` может стоять только строчный контент. Внутри элемента `<dd>...</dd>` можно ставить как строчные, так и блочные элементы. Элемент `<dd>...</dd>` визуально автоматически делает отступ с левой стороны на 40 пикселей, тем самым он отделяет термин от описания. Внутри элемента `<dl>...</dl>` элементы `<dt>...</dt>` и `<dd>...</dd>` могут стоять в любом порядке и в любом количестве. Других элементов в элементе `<dl>...</dl>` быть не может.

Например, список определений можно использовать в рекламе изделия:

**Низкая цена**

Новая модель этого изделия существенно дешевле предыдущей!

**Проще работа**

Мы изменили изделие, так что с ним теперь легко работать!

**Безопасно для детей**

Вы можете оставить своих детей в комнате, и изделие не причинит им вреда (не гарантируется).

На языке HTML он определяется следующим образом:

```
<dl>
  <dt>Низкая цена</dt>
  <dd> Новая модель этого изделия существенно дешевле
  предыдущей!</dd>
  <dt>Проще работа</dt>
  <dd>Мы изменили изделие, так что с ним теперь легко
  работать!</dd>
  <dt>Безопасно для детей </dt>
  <dd> Вы можете оставить своих детей в комнате, и изделие не
  причинит им вреда (не гарантируется).</dd>
</dl>
```

Списки могут быть *вложенными* (смешанными). Разные типы списков можно использовать вместе, как в следующем примере

(рисунок 3.4), где список определений содержит неупорядоченный список (новость дня) и упорядоченный список (новость ночи).

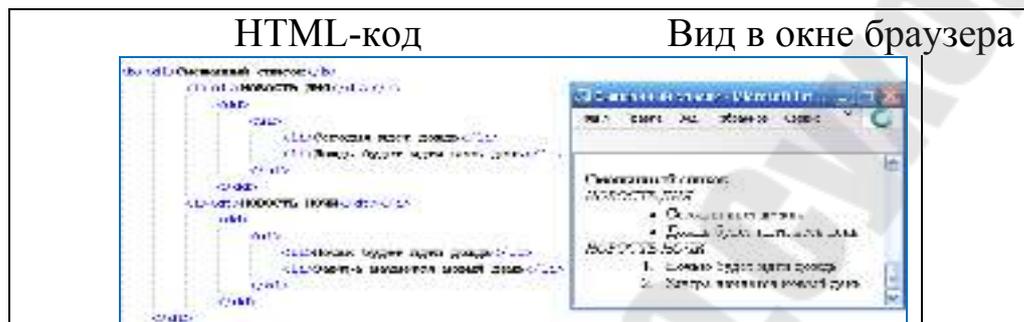


Рисунок 3.4 – Смешанный список

Не стоит использовать списки для создания отступов в тексте. Это делается с помощью каскадных таблиц стилей (CSS).

### Практическое задание

#### **Задание 1.**

Откройте с помощью программы notepad++ файл lab 3.html.

В нижеприведенном тексте есть два элемента неупорядоченного списка, есть элементы упорядоченного списка, а также списки определений. Цифры, дефисы, отступы - это визуальная подсказка, того, как браузер должен показать этот текст. Оформите нижеприведенный текст с помощью тегов HTML.

- Книги

1. Куринария

Холодные закуски

И.П. Хромов "Любимые бутерброды", Р.О. Дроздов "Мои лучшие рецепты закусок",

К.Нетман "Быстро и просто. Холодные закуски"

Домашние рецепты

О. Островская "Едим дома", Л. Смит "Вкусная домашняя еда",

Д. Орлова "Супы. Просто, полезно, вкусно"

Кондитерские изделия

А.Л. Пиров "Пироги на любой вкус", Р. Агунеш, З. Гульдес

"Ассорти", Р.Л. Долидова "Лучшие кексы",

О. Бауман "Торты. Лучшие рецепты"

Старинная кухня

Автор настоящей книги собрал более полутора тысяч рецептов

приготовления блюд, вин, соков народных мастеров-кулинаров и виноделов за период 130 последних лет

2. Художественная литература

3. Юмористическая литература

4. Детективы

- 5. Поэзия
- 6. Историческая литература

- Специальные предложения
  - a. Бестселлеры
  - b. Скидки
  - c. Аудиокниги

**Задание 2.** Создайте свой собственный смешанный список.

### **Вопросы для контроля по лабораторной работе**

1. Какие Вы знаете типы списков?
2. Элементы ol, ul, dd и их атрибуты.
3. Как управлять видом маркеров в неупорядоченном списке?

## **Лабораторная работа № 4**

### **Создание таблиц. Вложенные таблицы**

**Цель работы:** научиться создавать простые таблицы в HTML. Освоить атрибуты colspan и rowspan. Научиться создавать горизонтальное и вертикальное меню с помощью средств HTML.

### **Теоретические сведения**

#### ***Таблица и ее элементы***

Рассмотрим создание простой таблицы поэтапно.

Сначала зададим две строки таблицы.

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

```
</table>
```

Теперь в каждой строке зададим по два столбца (ячейки):

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

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

Для начала рекомендуем нарисовать желаемую таблицу на листе бумаге, чтобы все наглядно видеть.

### **Правила задания атрибутов для таблицы и ее ячеек.**

#### **Объединение ячеек**

Тег **<table>** может включать следующие атрибуты:

**width** - определяет ширину таблицы в пикселях или процентах, по умолчанию ширина таблицы определяется содержимым ячеек.

**border** - устанавливает толщину рамки. По умолчанию таблица рисуется без рамки.

**bordercolor** - устанавливает цвет окантовки.

**bgcolor** - устанавливает цвет фона для всей таблицы.

**background** - заполняет фон таблицы изображением.

**cellspacing** - определяет расстояние между рамками ячеек таблицы в пикселях.

**cellpadding** - определяет расстояние в пикселях между рамкой ячейки и текстом.

**align** - определяет расположение таблицы в документе. По умолчанию таблица прижата к левому краю страницы. Допустимые значения атрибута align: *left* (слева), *center* (по центру страницы) и *right* (справа).

**frame** - управляет внешней окантовкой таблицы, может принимать следующие значения:

*void* - окантовки нет (значение по умолчанию).

*above* - только граница сверху.

*below* - только граница снизу.

*hsides* - границы сверху и снизу.

*vsides* - только границы слева и справа.

*lhs* - только левая граница.

*rhs* - только правая граница.

*box* - рисуются все четыре стороны.

*border* - также все четыре стороны.

**rules** - управляет линиями, разделяющими ячейки таблицы.

Возможные значения атрибута rules:

*none* - нет линий (значение по умолчанию).

*groups* - линии будут только между группами рядов.

*rows* - только между рядами.

*cols* - только между колонками.

*all* - между всеми рядами и колонками.

Таблица может включать заголовок, который располагается между тегами **<caption>...</caption>**. Он должен быть непосредственно после

тега `<table>`. К заголовку возможно применение атрибута **align**, определяющего его положение относительно таблицы со следующими значениями:

*top* - значение по умолчанию, заголовок над таблицей по центру.

*left* - заголовок над таблицей слева.

*right* - заголовок над таблицей справа.

*bottom* - заголовок под таблицей по центру.

Строки таблицы начинаются открывающимся тэгом `<tr>` и завершаются закрывающимся `</tr>`, а каждая ячейка таблицы начинается тэгом `<td>` и завершается `</td>`. Данные теги могут иметь такие атрибуты:

**align** - устанавливает горизонтальное выравнивание текста в ячейках строки. Может принимать значения:

*left* – выравнивание влево;

*center* – выравнивание по центру;

*right* – выравнивание вправо.

**valign** - устанавливает вертикальное выравнивание текста в ячейках строки. Принимает допустимые значения: *top* – выравнивание по верхнему краю;

*center* – выравнивание по центру (по умолчанию);

*bottom* – выравнивание по нижнему краю.

**Bgcolor** - устанавливает цвет фона строки или ячейки.

**background** - заполняет фон строки или ячейки изображением.

Следующие атрибуты могут применяться только для ячеек.

**width** - определяет ширину ячейки в пикселях.

**height** - определяет высоту ячейки в пикселях.

**nowrap** - присутствие этого атрибута показывает, что текст должен размещаться в одну строку.

**background** - заполняет фон ячейки изображением.

Кроме этого, любая ячейка таблицы может быть определена не тэгами `<td>...</td>`, а тэгами `<th>...</th>`. Текст внутри тегов `<th>...</th>` будет выделен полужирным шрифтом и отцентрирован. Однако, в настоящее время, тег `<th>` уже устарел, поскольку для работы с оформлением текста применяется CSS.

Если ячейка пустая, то вокруг нее рамка не рисуется. Если рамка все же нужна вокруг пустой ячейки, то в нее надо ввести символьный объект **&nbsp;**; (от англ. non-breaking space, неразрывающий пробел). Ячейка по-прежнему будет пуста, но рамка вокруг нее будет (&nbsp; -

обязательно должен набираться строчными буквами и закрываться точкой с запятой).

Теги, устанавливающие шрифт (`<b>`, `<i>`, `<font size="значение">`, `<font color="значение">`), необходимо повторять для каждой ячейки.

Подробнее остановимся на атрибутах `colspan` и `rowspan`.

**Colspan** - определяет количество столбцов, на которые простирается данная ячейка, а **rowspan** - количество рядов. Эти параметры могут принимать значение от 2 и более, т.е. наша ячейка может растягиваться на два и более столбца (ряда). Теперь, обратимся к примерам.

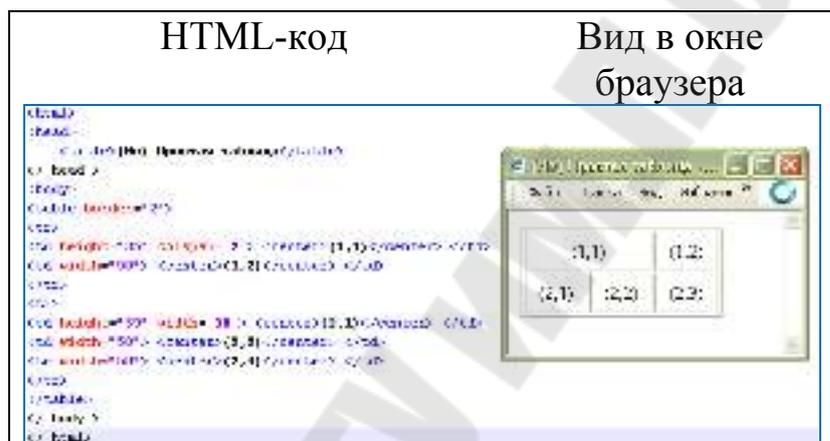


Рисунок 4.1 – Использование атрибута `colspan`

В таблице на рис. 4.1 использован параметр `colspan = 2`, прописанный для ячейки (1,1), где первая цифра - это номер ряда, а вторая - номер столбца (т.е. (1,2) - первый ряд, второй столбец и т.д.).

Обратите внимание, на то, что параметр `width` для ячейки (1,1) в примере на рис. 4.1 не указан. Если необходимо задать этот атрибут, то в данном примере для ячейки (1,1) его надо было бы прописать равным 100 пикселям, т.к. все-таки ячейка (1,1) длиннее других в два раза.

И второе на что следует обратить внимание, в рассмотренном примере на рис. 4.1 нет ячейки (1,3), т.е. в первом ряду всего лишь две ячейки, т.к. ячейка (1,1) равна сама по себе двум ячейкам по длине (что мы и указали параметром `colspan`). Если вы прописали ячейку (1,3), тогда у вас получилась бы такая «таблица», как на рис. 4.2.

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)

Рисунок 4.2 – Ошибочное построение таблицы

Теперь, рассмотрим параметр `rowspan`. Принцип действия тут тот же. Попробуйте самостоятельно прописать код для таблицы, что представлена на рис. 4.3.

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	

Рисунок 4.3 – Использование атрибута `rowspan`

### Практическое задание

**Задание 1.** В файле `lab 4-1.html` создайте простую таблицу, содержащую не менее 9 ячеек. Содержимое ячеек оформить самостоятельно. В одну из ячеек вставьте картинку. Задайте различную ширину столбцов с помощью элемента `col` или `colgroup`.

**Задание 2.** В файле `lab 4-1.html` создайте 4 таблицы, изображенные ниже. Используйте атрибуты `rowspan` и `colspan`. Заполните содержимым ячейки таблиц.

Разместите данные таблицы в один ряд двумя способами. Каждый способ продемонстрируйте преподавателю.

С помощью элемента `tbody` выровнять содержимое всех ячеек первой таблицы по левому краю, второй таблицы – по правому, в третьей – по верхнему краю, а в четвертой – по нижнему.

1	2		3	4	5	6
	5	6				

1		3	4	5	6
3	4				

1	2	3	4	5
3				

1		3	4	5
3	4			

**Задание 3.** В файле `lab 4-2.html` создайте вложенные таблицы с вертикальным и горизонтальным меню как показано на рисунках ниже, не используя атрибуты `rowspan` и `colspan`. Результаты работы покажите

600					
<table border="1"> <tr><td> </td></tr> <tr><td> </td></tr> <tr><td> </td></tr> <tr><td> </td></tr> </table>					400

<table border="1"> <tr><td> </td><td> </td><td> </td></tr> </table>					
<table border="1"> <tr><td> </td></tr> <tr><td> </td></tr> <tr><td> </td></tr> <tr><td> </td></tr> </table>					

## Вопросы для контроля по лабораторной работе № 4

1. Элемент table и его атрибуты.
2. Для чего нужны и как правильно использовать атрибуты colspan и rowspan?
3. Основные приемы создания вложенных таблиц?

## Лабораторная работа № 5

### Формы

**Цель работы:** научиться создавать формы на html-страницах с помощью средств языка HTML.

### Теоретические сведения

Форма — это инструмент, с помощью которого HTML-документ может отправить информацию в заранее определенную точку внешнего мира. Формы применяются для опроса посетителей, покупки чего-либо, отправки электронной почты.

Формы размещаются между тегами `<form>` `</form>`. HTML-документ может содержать в себе несколько форм, но они не должны находиться одна внутри другой. Тег `<form>` может содержать следующие атрибуты:

**Action** - Обязательный атрибут. Определяет, где находится обработчик формы.

**Method** - Определяет, каким образом данные из формы будут переданы обработчику. Допустимые значения: `method="post"` и `method="get"`. Если значение атрибута не установлено, по умолчанию предполагается `method="get"`.

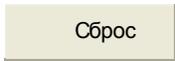
**enctype** - Определяет, каким образом данные из формы будут закодированы для передачи обработчику.

Для внесения информации пользователем в форму используется элемент `<input>` Это и есть поля, в которые пользователь вводит информацию. Каждый элемент `<input>` включает атрибут `name="имя"`, определяющий имя данного поля (идентификатор поля). В таблице 5.1 представлены основные типы, применяемых элементов `<input>`.

Таблица 5.1 – Атрибуты элемента input

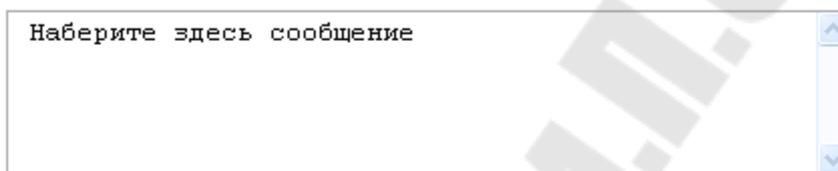
<p>type="text"</p>	<p>Определяет окно для ввода строки текста. Может содержать дополнительные атрибуты size="число" (ширина окна ввода в символах) и maxlength="число" (максимально допустимая длина вводимой строки в символах):</p> <pre>&lt;input type="text" size="20" name="User" value="логин"&gt;</pre> <div data-bbox="596 472 890 521" style="border: 1px solid black; padding: 2px; width: fit-content;">             логин         </div> <p>Определяет окно шириной 20 символов для ввода текста. По умолчанию в окне находится текст LENIN INC, который пользователь может изменить.</p>
<p>type="password"</p>	<p>Определяет окно для ввода пароля. Абсолютно аналогичен типу text, только вместо символов вводимого текста показывает на экране звездочки (*):</p> <pre>&lt;input type="password" name="PW" size="20" maxlength="10"&gt;</pre> <div data-bbox="596 1016 890 1066" style="border: 1px solid black; padding: 2px; width: fit-content;">             *         </div> <p>Определяет окно шириной 20 символов для ввода пароля. Максимально допустимая длина пароля — 10 символов.</p>
<p>type="radio"</p>	<p>Определяет радиокнопку. Может содержать дополнительный атрибут checked (показывает, что кнопка отмечена). В группе радиокнопок с одинаковыми именами может быть только одна помеченная радиокнопка:</p> <pre>&lt;input type="radio" name="Question" value="Yes" checked&gt; Да &lt;input type="radio" name="Question" value="No"&gt; Нет &lt;input type="radio" name="Question" value="Possible"&gt; Возможно</pre> <div data-bbox="596 1666 788 1783" style="margin-left: 20px;"> <input checked="" type="radio"/> Да  <input type="radio"/> Нет  <input type="radio"/> Возможно         </div> <p>Определяет группу из трех радиокнопок, подписанных Yes, No и Possible. Первоначально помечена первая из кнопок. Если пользователь не отметит другую кнопку, обработчику будет передана переменная Question со значением Yes. Если пользователь отметит другую кнопку, обработчику будет передана переменная Question со значением No или Possible.</p>

type=checkbox	<p>Определяет квадрат, в котором можно сделать пометку. Может содержать дополнительный атрибут checked (показывает, что квадрат помечен). В отличие от радиокнопок, в группе квадратов с одинаковыми именами может быть несколько помеченных квадратов:</p> <pre>&lt;input type=checkbox name=Comp value="CPU"&gt; Процессоры &lt;input type=checkbox name=Comp value="Video" checked&gt; Видеоадаптеры &lt;input type=checkbox name=Comp value="Scan"&gt; Сканеры &lt;input type=checkbox name=Comp value="Modem" checked&gt; Модемы</pre> <div data-bbox="587 730 882 913" style="border: 1px solid black; padding: 5px;"> <input type="checkbox"/> Процессоры  <input checked="" type="checkbox"/> Видеоадаптеры  <input type="checkbox"/> Сканеры  <input checked="" type="checkbox"/> Модемы </div> <p>Определяет группу из четырех квадратов. Первоначально помечены второй и четвертый квадраты. Если пользователь не произведет изменений, обработчику будут переданы две переменные: Comp=Video и Comp=Modem.</p>
type=hidden	<p>Определяет скрытый элемент данных, который не виден пользователю при заполнении формы и передается обработчику без изменений. Такой элемент иногда полезно иметь в форме, которая время от времени подвергается переработке, чтобы обработчик мог знать, с какой версией формы он имеет дело.</p> <pre>&lt;input type=hidden name=version value="1.1"&gt;</pre> <p>Определяет скрытую переменную version, которая передается обработчику со значением 1.1.</p>
type=submit	<p>Определяет кнопку, при нажатии на которую запускается процесс передачи данных из формы обработчику:</p> <pre>&lt;input type=submit value="Отправить"&gt;</pre> <div data-bbox="587 1832 746 1888" style="border: 1px solid black; padding: 2px; display: inline-block;">Отправить</div>
type=reset	<p>Определяет кнопку, при нажатии на которую очищаются поля формы. Поскольку при использовании этой кнопки данные</p>

	<p>обработчику не передаются, кнопка типа reset может и не иметь атрибута name:</p> <pre>&lt;input type=reset value=" Сброс "&gt;</pre> 

Формы могут содержать поля для ввода большого текста `<textarea>`:

```
<textarea name=address rows=5 cols=50> Наберите здесь сообщение</textarea>
```



*Рисунок 5.1 – Элемент `textarea`*

Атрибут `name` определяет имя, под которым содержимое окна будет передано обработчику. Атрибут `rows` устанавливает высоту окна в строках. Атрибут `cols` устанавливает ширину окна в символах. Текст, размещенный между тегами `<textarea></textarea>`, представляет собой содержимое окна по умолчанию. Пользователь может его отредактировать или просто стереть.

Кроме всего этого формы могут содержать меню выбора, которое начинается открывающимся тегом `<select>` (содержит обязательный атрибут `name`, определяющий имя меню) и завершается закрывающимся `</select>`. Между ними находятся теги `<option>`, определяющие элемент меню. Обязательный атрибут `value` устанавливает значение, которое будет передано обработчику, если выбран этот элемент меню. Тег `<option>` может включать атрибут `selected`, показывающий, что данный элемент выбран/отмечен по умолчанию.

```
<select name="имя">
  <option value="option_1" selected>текст 1
  <option value="option_2">текст 2
  <option value="option_n">текст n
</select>
```

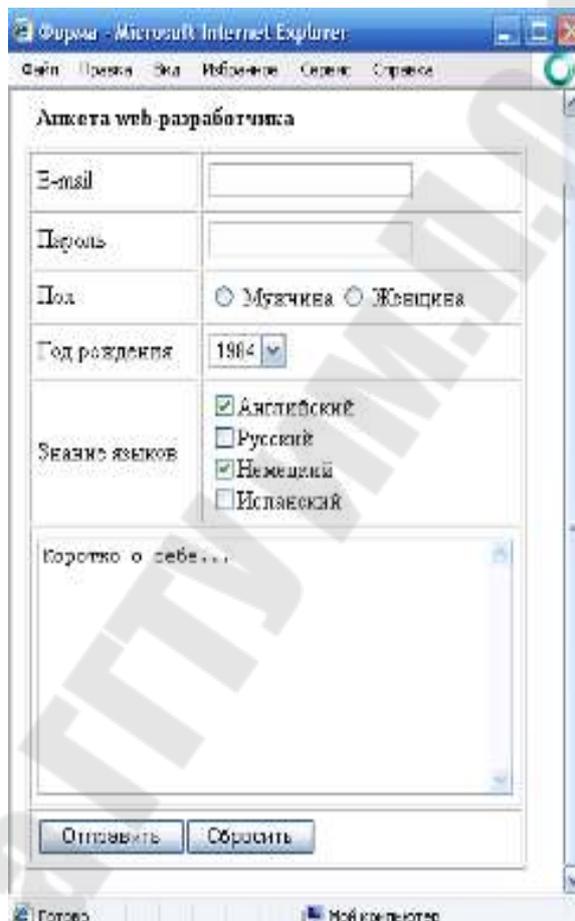


Тег `<select>` может также содержать атрибут `multiple`, присутствие которого показывает, что из меню можно выбрать несколько элементов. Большинство браузеров показывают меню `<select multiple>` в виде окна,

в котором находятся элементы меню. Высоту окна в строках можно задать атрибутом `size=число`.

### Практическое задание

**Задание 1.** Создайте форму «Анкета web-разработчика» по следующему примеру.



**Задание 2.** Создайте свою собственную форму.

### Вопросы для самоконтроля по лабораторной работе № 5

1. Что такое форма и для чего она нужна?
2. Перечислите основные элементы формы.
3. Перечислите методы отправки информации из полей формы.
4. Опишите элемент `input` и его атрибуты (`type`, `name`, `value`, `checked`, `size`, `maxlength`, `align`, `src`).
5. Как создать многострочное текстовое поле?
6. Перечислите атрибуты тега `<textarea>...</textarea>`.
7. Как создать поле выбора файлов?
8. Назовите способы создания флажков и переключателей.
9. Перечислите виды кнопок и опишите форматы их записи.

## Лабораторная работа №6

### Форматирование текста с помощью CSS

**Цель работы:** научиться форматировать текст с помощью языка стиливой разметки CSS.

#### Теоретические сведения

##### *Основные понятия и определения. Грамматика языка стилей*

Как уже было отмечено ранее, стилевые правила записываются в своем формате, отличном от HTML. Основным понятием выступает *селектор* – это некоторое имя стиля, для которого добавляются параметры форматирования. В качестве селектора выступают теги, классы и идентификаторы. Общий способ записи имеет следующий вид:

```
Селектор {  
    свойство1: значение;  
    свойство2: значение;  
    свойство3: значение;  
}
```

Вначале пишется имя селектора, например, h1, это означает, что все стилевые параметры будут применяться к тегу <h1>, затем ставятся фигурные скобки, в которых записывается стилевое свойство, а его значение указывается после двоеточия. Стилиевые свойства разделяются между собой точкой с запятой, в конце этот символ тоже ставится.

Форма записи правил зависит от желания разработчика, поскольку CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции. Так, в примере 6.1 показаны две разновидности оформления селекторов и их правил.

##### **Пример 6.1** – Использование стилей

```
<html>  
  <head>  
    <title>Заголовки</title>  
    <style type="text/css">  
      h1 {  
color: #a6780a;  
font-weight: normal;  
      }  
      h2 { color: olive; border-bottom: 2px solid black; }  
    </style>  
  </head>  
<body>  
  <h1>Заголовков 1</h1>
```

```
<h2>Заголовок 2</h2>
</body>
</html>
```

В данном примере свойства селектора h2 записаны в одну строку, а для селектора h1 каждое свойство находится на отдельной строке. В первом случае легче отыскивать нужные свойства и править их по необходимости, но при этом незначительно возрастает объем данных за счет активного использования пробелов и переносов строк. Так что в любом случае способ оформления стилевых параметров зависит от разработчика.

Далее приведены некоторые правила, которые необходимо знать при описании стиля.

*Правило 1.* Для селектора допускается добавлять каждое стилевое свойство и его значение по отдельности, как это показано в примере 6.2.

**Пример 6.2** – Расширенная форма записи

```
td { background: olive; }
td { color: white; }
td { border: 1px solid black; }
```

Очевидно, что такая запись не очень удобна. Приходится повторять несколько раз один и тот же селектор, да и легко запутаться в их количестве. Поэтому пишите все свойства для каждого селектора вместе. Указанный набор записей в таком случае получит следующий вид (пример 6.3). Такая форма записи более наглядная и удобная в использовании.

**Пример 6.3** – Компактная форма записи

```
td {
  background: olive;
  color: white;
  border: 1px solid black;
}
```

*Правило 2.* Если для селектора вначале задается свойство с одним значением, а затем то же свойство, но уже с другим значением, то применяться будет то значение, которое в коде установлено ниже (пример 6.4).

**Пример 6.4** – Разные значения у одного свойства

```
p {color: green;}
p {color: red;}
```

В данном примере для селектора p цвет текста вначале задается зеленым, а затем красным. Поскольку значение red расположено ниже, то оно в итоге и будет применяться к тексту.

На самом деле такой записи лучше вообще избегать и удалять повторяющиеся значения. Но подобное может произойти не явно, например, в случае подключения разных стилевых файлов, в которых содержатся одинаковые селекторы.

*Правило 3.* У каждого свойства может быть только соответствующее его функции значение. Например, для color, который устанавливает цвет текста, в качестве значений недопустимо использовать числа.

Комментарии нужны, чтобы делать пояснения по поводу использования того или иного стилового свойства, выделять разделы или писать свои заметки. Комментарии позволяют легко вспоминать логику и структуру селекторов, и повышают разборчивость кода. Вместе с тем, добавление текста увеличивает объем документов, что отрицательно сказывается на времени их загрузки. Поэтому комментарии обычно применяют в отладочных или учебных целях, а при выкладывании сайта в сеть их стирают.

Любой CSS-комментарий начинается с конструкции /\* и заканчивается конструкцией \*/.

**Пример 6.5** – Комментарии в CSS-файле

```
/*
Стиль для сайта mysite.by
*/
div {
    width: 200px; /* Ширина блока */
    margin: 10px; /* Поля вокруг элемента */
    float: left; /* Обтекание по правому краю */
}
```

Как следует из примера 6.5, комментарии можно добавлять в любое место CSS-документа, а также писать текст комментария в несколько строк. Вложенные комментарии недопустимы.

### **Ход выполнения работы**

Создать маркированный список, в котором один и тот же текст выводится с использованием различных стилей:

- первый вариант: цвет – красный, размер шрифта – 14pt, стиль – наклонный, выравнивание – по правому краю;
- второй вариант текста заключен в сплошную тонкую рамку зеленого цвета, цвет текста – зеленый, размер шрифта – 12 pt, выравнивание – по центру.

Текст: Тестирование – это защитная сеть вокруг вашей программы, последняя и порой единственная надежда на серьезное повышение ее



text-align:center;">

Тестирование - это защитная сеть вокруг вашей программы, последняя и порой единственная надежда на серьезное повышение ее качества до того, как она выйдет в свет. (Лу Гринзоу, "Философия программирования")

</li></ul></body></html>

### **Практическое задание**

**Задание 1.** Прodelайте комплекс тренировочных упражнений (файлы lab 7-1-1.html – 7-1-5.html). Откройте в блокноте (notepad++) файл lab 7-1-1.html и выполните предложенные задания (в комментариях). Внимательно прочитайте задания. Продемонстрируйте весь выполненный комплекс упражнений преподавателю.

После переходите к выполнению задания lab 7-1.html.

Этот файл должен отображаться в браузере так:

1. основные цвета: черный текст на белом фоне;  
(Напоминание: все элементы на странице имеют по умолчанию цвет фона прозрачным)
2. абзацы выравниваются справа;
3. заголовки выравниваются справа и записываются рубленым шрифтом  
красного цвета, заголовки должны быть записаны прописными буквами,  
сжатыми на  $-1em$ ;
4. термины, выделяемые курсивом, записываются зеленым цветом.

**Задание 2.** В блокноте (notepad++) откройте файл lab 7-2.html.

Задать области для следующих элементов:

для заголовков

1. задание внешнего отступа: сверху нет отступа, справа и слева по 50px, а снизу задать 100px;
2. внутренний отступ 50px со всех сторон;
3. нарисовать рамку шириной в 10px темно-синего цвета, стиль рамки - сплошная линия;
4. фон элемента задать светло-голубого цвета;

для параграфов

1. задание внешнего отступа: сверху 30px, справа и слева по 150px, а снизу задать 50px;
2. внутренний отступ 30px со всех сторон;

3. нарисовать рамку шириной в 10px темно-красного цвета, стиль рамки - сплошная линия;
4. фон элемента задать бледно розового цвета;

Используя этот файл, экспериментируйте со свойствами стилей:

font

size

color

background

border

margin

padding

Выполненное задание покажите преподавателю.

### **Вопросы для контроля по лабораторной работе № 6**

1. Какие три типа CSS Вы знаете?
2. В каком месте Web-страницы записываются встроенные CSS?
3. Где располагается описание внешних CSS?
4. В каком месте страницы описываются внедренные CSS?
5. Какое основное достоинство каскадных таблиц стилей?
6. Какое расширение должен иметь файл с описанием внешних CSS?

### **Лабораторная работа №7**

#### **Позиционирование. Создание объемного текста**

**Цель работы:** с помощью позиционирования научиться создавать объемную запись из обычного текста.

#### **Теоретические сведения**

При работе с элементами, предполагается, что элементы на web-странице будут идти в той же последовательности, в которой они стоят в коде. Это называется *нормальный поток*. Разделяют два правила, которые этот нормальный поток могут нарушить.

Это такие правила как:

1. *float : none | left | right;*
2. *position : static | absolute | relative;*

Рассмотрим эти правила подробнее.

Правило **float** определяет плавание. Как только элементу поставить правило `float: left|right;`, то этот элемент становится плавающим, что означает, что э элементом происходит следующее:

- элемент становится блочным;
- плавающий элемент свои размеры определяет в соответствии со своим контентом. Это значит, что если в плавающем элементе контента нет, а ширина и высота были заданы, то он просто «схлопнется»;
- блочные элементы плавающие игнорируют (исключением являются таблицы), а строчные элементы их обтекают;
- плавающие элементы в схемах с `margin colabs` не участвуют.

Можно сделать так, чтобы блочные элементы, идущие в коде за плавающими, учитывали эти плавающие элементы. Для этого воспользуемся специальным правилом

*clear : none | left | right | both;*

С помощью этого правила блочные элементы учитывают (слева, справа, с обеих сторон) плавающий элемент.

По умолчанию `float` и `clear` имеют значения `none`.

Плавающие элементы встают максимально высоко и максимально в левую сторону или в правую в зависимости от правила `float`. А поскольку плавающие элементы друг друга учитывают, то это позволяет поставить их в ряд.

Правило **position** тоже серьезно меняет взаимоотношения элементов. Элемент с позиционированием можно двигать относительно каких-то областей.

*Position : static;* - это правило по умолчанию стоит у всех элементов. Применение этого правила, дает возможность поставить все элементы друг за другом.

*Position : relation;* - это относительное позиционирование. В этом случае нормальный поток не нарушается. Однако с помощью задания смещения (`top`, `left`, `right`, `bottom`), мы можем смещать этот элемент относительно своего положения.

*top : auto | величина | %;*

*left : auto | величина | %;*

*right : auto | величина | %;*

*bottom : auto | величина | %;*

Плавающие элементы реализуются для колонок.

*Position : absolute;* - позволяет позиционировать элемент относительно каких-то других элементов на странице.

Для строчных элементов ширина, высота и вертикальный margin игнорируются.

Следует запомнить несколько свойств:

- Любой элемент с Position : absolute; становится блочным.
- Его размеры определяются его контентом.
- Ни один элемент на странице, кроме корневого, его не учитывает.

У правила position есть еще одно значение – *fixed*. По своему действию это значение близко к absolute, но в отличие от него привязывается к указанной свойствами left, top, right и bottom точке на экране и не меняет своего положения при прокрутке веб-страницы. Браузер Firefox вообще не отображает полосы прокрутки, если положение элемента задано фиксированным, и оно не помещается целиком в окно браузера. В браузере Opera хотя и показываются полосы прокрутки, но они никак не влияют на позицию элемента.

Следующие правила используют для динамических эффектов.

Правило *display* : none | block | inline | list-item по умолчанию есть у любого элемента. Это тип представления элемента. Например, для элементов div, p, h1, display находится в значении block. Для таких элементов как span, em, strong правило display имеет значение inline. Для элементов li – display: list-item;. Значение none говорит о том, что элемент не будет показываться вообще.

Правило *visibility* : hidden | visible | inherit; - это видимость нашего элемента.

visibility : hidden; – элемент не показывается, но место под его резервируется.

visibility : visible; – значение по умолчанию.

Правило *overflow* : auto | scroll | visible | hidden;. Переполнение контентом возникает тогда, когда заданные размеры не соответствуют размерам содержания.

overflow : visible; – значение по умолчанию. Использование этого правила, приведет к тому, что все что выйдет за пределы контента будет показано и не будет влиять на другие элементы.

overflow : hidden; – контент, который не вместился, скроется и у пользователя отсутствует возможность просмотреть его.

overflow : scroll; – появятся полосы прокрутки, которые позволят увидеть весь текст.

overflow : auto; – полосы прокрутки появляются сами в зависимости от размеров контента. Если контент вмещается в область, то никаких полос прокрутки добавлено не будет.

Правило *z-index* : auto | величина | inherit;

В качестве величины используются целые числа (положительные, отрицательные и ноль). Чем больше значение, тем выше находится элемент по сравнению с теми элементами, у которых оно меньше. При равном значении z-index, на переднем плане находится тот элемент, который в коде HTML описан ниже. Хотя спецификация и разрешает использовать отрицательные значения z-index, но такие элементы не отображаются в браузере Firefox до версии 2.0 включительно. Кроме числовых значений применяется auto — порядок элементов в этом случае строится автоматически, исходя из их положения в коде HTML и принадлежности к родителю, поскольку дочерние элементы имеют тот же номер, что и родительский элемент. Значение inherit указывает, что оно наследуется у родителя.

*Правило clip : auto | rect (top right bottom left);*

Определяет ту часть элемента, которую вы собираетесь показать. В статических страницах это правило используется только при сложной верстке. Сразу появляется возможность для элементов с position : absolute | fixed; показать часть элемента.

В качестве значения clip выступает некоторая область. Планировалось, что таких областей будет много, однако на практике используется только прямоугольная область - rect. Например, clip : rect (10px 20px 30px 10px); Все значения вычисляются относительно самой верхней левой точки области, в которой выделяем необходимую часть.

*Правило cursor: [url('путь к курсору'),] | [ auto | crosshair | default | e-resize | help | move | n-resize | ne-resize | nw-resize | pointer | progress | s-resize | se-resize | sw-resize | text | w-resize | wait | inherit ]*

Устанавливает форму курсора, когда он находится в пределах элемента. Вид курсора зависит от операционной системы и установленных параметров. Прежде чем воспользоваться возможностью переделать вид курсора, решите, а будет ли он использоваться к месту. Многих пользователей подобные изменения могут ввести в заблуждение, когда, например, вместо традиционной руки, появляющейся при наведении на ссылку, возникает нечто другое. В большинстве случаев, лучше оставить все как есть.

По умолчанию у всех видимых элементах используется правило cursor : auto;

Бывают и другие значения.

*Cursor: crosshair;* - перекрестная линия, оптический прицел. Данное правило используется там, где нужно четко попадать в какие-то координаты.

*Cursor: pointer;* - указатель-рука, которая является своеобразной подсказкой, что можно кликнуть. При наведении на элемент, у которого

прописано правило `cursor:pointed`; курсор будет меняться на изображение руки.

*Cursor:help*; - вместо курсора появится курсор со знаком вопроса. Показывает, что пользователь может кликнуть на элемент и появится всплывающая подсказка.

*Cursor:wait*; - курсор в виде песочных часов.

*Cursor:move*; - подсказка, что пользователь может передвигать элемент.

*Cursor:text*; - вид вертикальной черты при наведении на текст. Означает, что текст можно выделить и что-то с ним сделать.

Есть значения, которые говорят, как наш курсор расположить относительно сторон света. Всего 8 направлений: *cursor: e-resize | n-resize | ne-resize | nw-resize | s-resize | se-resize | sw-resize | w-resize*;

### **Практическое задание**

Проделайте комплекс тренировочных упражнений (файлы lab 8-1 – 8-8). Откройте в блокноте (notepad++) файл lab 8-1.html и выполните предложенные задания (в комментариях). Внимательно прочитайте задания. Продемонстрируйте весь выполненный комплекс упражнений преподавателю.

После выполнения упражнений переходите к выполнению задания lab 8.html. Создайте объемное изображение слов HTML и вашей фамилии, используя позиционирование в два и три слоя.

### **Вопросы для контроля по лабораторной работе № 7**

1. Для чего используются классы?
2. Какие программы используются для автоматизации создания таблиц стилей?
3. Какова зона действия встроенных CSS?
4. Какова зона действия внедренных CSS?
5. Какова зона действия внешних CSS?
6. Как с помощью CSS создать псевдографическое изображение?
7. Что такое стиль?

### **Лабораторная работа №8**

#### **Создание панелей. Верстка веб-страницы**

**Цель работы:** научиться создавать панели.

## Теоретические сведения

### *Создание блока*

В этом разделе вы узнаете об одном из способов создания стандартного блока на основе CSS, состоящего из «шапки» и основной части.

Для этого потребуется выполнить ряд простых действий:

- 1) Для удобства вынесем CSS в отдельный файл.
- 2) Вставьте в этот файл код, приведенный в примере 8.1.
- 3) Чтобы увидеть работу стилей, создадим в той же директории html-файл с кодом как в примере 6.2.
- 4) Сохраняем и смотрим, что получилось.

Для удобства чтения и лучшего понимания кода, мы прописываем пояснения к его командам в комментариях.

### **Пример 8.1** – Создание блока на CSS (css файл)

```
/* Задание стилей всего блока */
#block {
    width: 250px; /* Задание ширины блока */
}

/* Задание стилей заголовка */
.head {
    text-align:center; /* Выравнивание заголовка по центру
блока */
    color: #fff; /* Задание цвета заголовка (тут - белый) */
    background-color: #0274b0; /* Задание цвета фона (тут -
синий) */
    border: 2px solid #ffba00; /* Задание сплошной границы
блока шириной в 2 пикселя и её цвета */
    font-size: 15px; /* Задание размера шрифта заголовка */
    font-weight:bold; /* Задание полужирного начертания
шрифта */
    padding: 7px 0 7px 0; /* Задание верхнего и нижнего
отступов текста заголовка от границ блока */
}

/* Задание стилей основного блока */
.body {
    color:#333; /* Задание цвета текста */
    background-color: #d2efff; /* Задание цвета фона */
```

```

border: 2px solid #ffba00; /* Задание сплошной границы
блока шириной в 2 пикселя и её цвета */
border-top-style: none; /* Удаление верхней границы
блока */
font-size: 12px; /* Задание размера шрифта */
padding: 5px; /* Задание отступа в 5 пикселей со всех
сторон */
}

```

### Пример 8.2 – Создание блока на CSS (html-файл)

```

<html>
<head>
<link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div id="block">
<div class="head">
ЗАГОЛОВОК БЛОКА
</div>
<div class="body">
ОСНОВНОЙ БЛОК.<br>
Текст блока....
</div>
</div>
</body>
</html>

```

Если ошибок при написании кода вами допущено не было, то в окне браузера вы увидите, блок представленный на рис. 8.1.

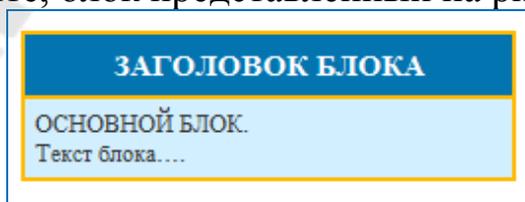


Рисунок 8.1 – Стандартный блок на CSS

### Создание текста в три колонки

В данном разделе поговорим об одном из самых простых способов вёрстки в несколько столбцов, без использования таблиц. Порядок действий тут такой же как и в разделе 8.1. Здесь мы лишь приведем содержание css-файла (пример 8.3) и html-файла (пример 8.4).

### Пример 8.3 – Создание текста в три колонки (css-файл)

```

/* Задание стилей всего блока */
#body {
    width:900px; /* Установка ширины блока в 900 пикселей */
}

/* Задание стилей левого столбца */
#left {
    float:left; /* Установка обтекания */
    width:300px; /* Установка ширины столбца */
    background:#aeddff; /* Установка фонового цвета */
    height: 300px; /* Установка высоты столбца */
}

/* Задание стилей правого столбца */
#right {
    float:right; /* Установка обтекания */
    width:300px; /* Установка ширины столбца */
    background:#ffecbe; /* Установка фонового цвета */
    height: 300px; /* Установка высоты столбца */
}

/* Задание стилей среднего столбца */
#center {
    margin-left:300px; /* Установка отступа */
    margin-right:300px; /* Установка отступа */
    background:#beffc0; /* Установка фонового цвета */
    height: 300px; /* Установка высоты столбца */
}

```

**Пример 8.4** - Создание текста в три колонки (html-файл)

```

<html>
<head>
<link rel="stylesheet" href="style.css" type="text/css" />
</head>
<body>
<div id="body">
    <div id="left">Текст в левом столбце</div>
    <div id="right">Текст в правом столбце</div>
    <div id="center">Текст в среднем столбце</div>
</div>
</body>
</html>

```

Способ довольно прост, тем не менее эффективен и гибок. Он допускает любое форматирование столбцов и их содержания.

В браузере перед вами появится вот такой вид, рис. 8.2.

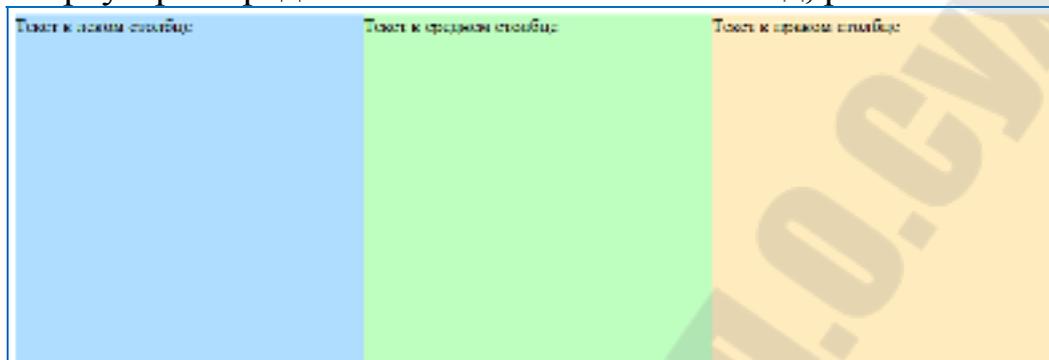


Рисунок 8.2 – Создание три колонки

В последующих двух разделах поговорим о различных способах создания шаблонов сайтов.

### ***Современная верстка сайта при помощи CSS***

Начнём с самого простого, стандартного типа шаблонов, рис. 8.3.

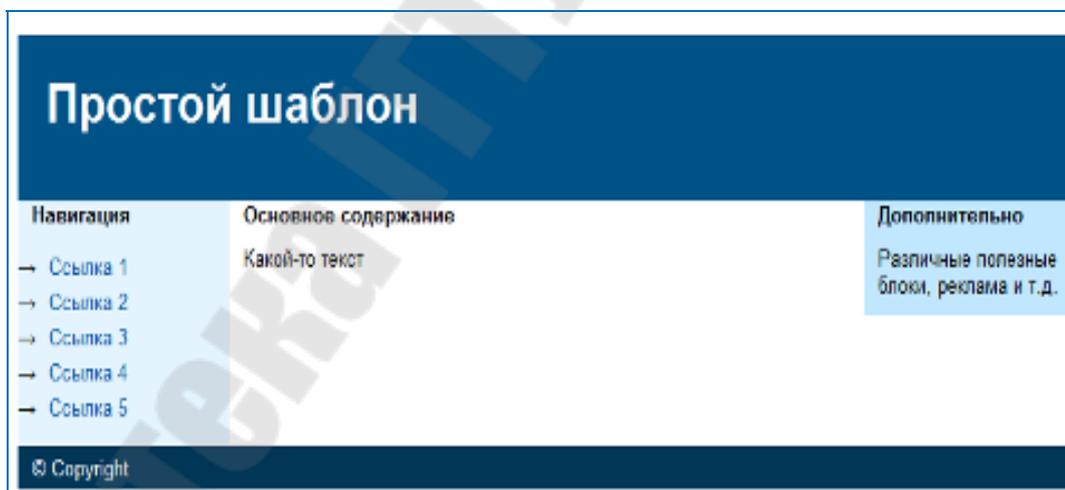


Рисунок 8.3 – Пример шаблона сайта

К особенностям шаблона на рис. 8.3 относятся:

- наличие 3-х колонок;
- стандартные поля заголовков;
- простое меню;
- отсутствие таблиц;
- отсутствие графики.

Этот шаблон будет очень удобен при изучении блочной вёрстки.

Как и раньше в примерах 6.5 и 6.6 приведем содержание css и html файлов соответственно с комментариями команд кода.

**Пример 8.5** – Простой шаблон (css-файл)

```
/* Задание стилей всего шаблона */  
body {  
    font: 80% Arial;  
    text-align:center;  
}  
  
/* Задание стилей новой строки */  
p {margin:0 10px 10px;}  
  
/* Задание стилей ссылок */  
a {  
    padding:5px;  
    text-decoration:none;  
    color:#0053a1;  
}  
  
/* Задание стилей ссылок при наведении */  
a:hover {  
    text-decoration:underline;  
    color:#067a00;  
}  
  
/* Задание стилей блока заголовка */  
div#header {  
    background-color:#005387;  
    color:#fff;  
    height:80px;  
    line-height:80px;  
    padding-left:20px;  
}  
  
/* Задание стилей всего шаблона */  
div#all {  
    text-align:left;  
    width:750px;  
    margin:0 auto;  
}
```

```
/* Задание стилей навигации */

```

```
float:left;
width:100%;
}
```

```
/* Задание стилей центрального столбца */
div#content {margin: 0 150px;}
```

### Пример 8.6 – Простой блок (html-файл)

```
<html>
<head>
<title>Шаблон сайта</title>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<div id="all">
<div id="header"><h1>Простой шаблон </h1></div>
  <div id="templ">
    <div id="content">
      <p><strong>Основное содержание</strong></p>
<p>Какой-то текст</p>
    </div>
  </div>
  <div id="navigation">

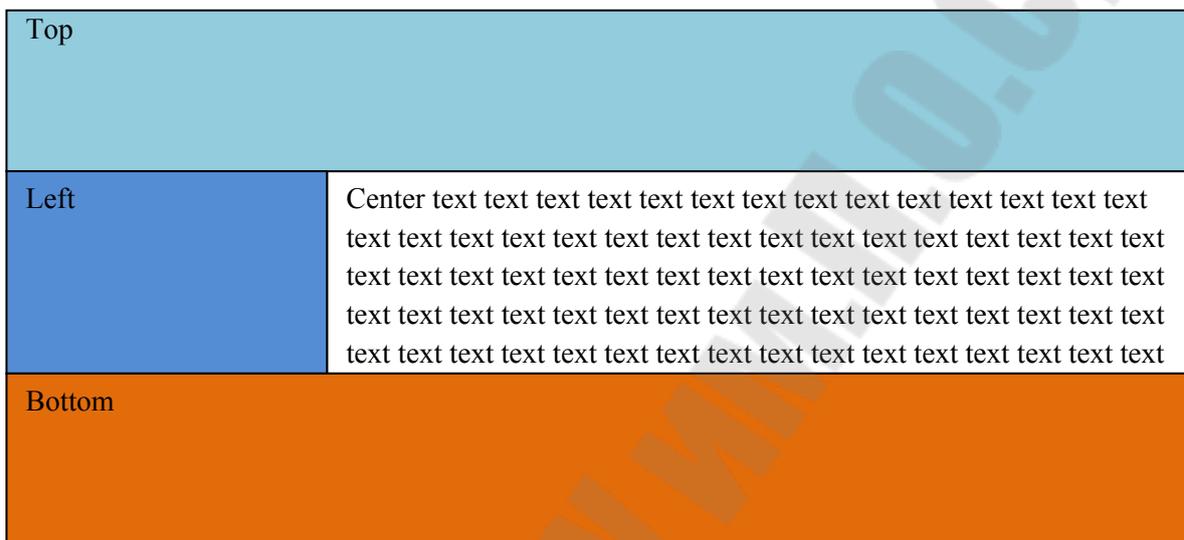
    <p><strong>Навигация</strong></p>
    <ul>
      <li>&#8594; <a href="#">Ссылка 1</a></li>
      <li>&#8594; <a href="#">Ссылка 2</a></li>
      <li>&#8594; <a href="#">Ссылка 3</a></li>
      <li>&#8594; <a href="#">Ссылка 4</a></li>
      <li>&#8594; <a href="#">Ссылка 5</a></li>
    </ul>
  </div>
  <div id="extra">
    <p><strong>Дополнительно</strong></p>
    <p>Различные полезные блоки, реклама и т.д.</p>
  </div>
  <div id="footer">
    <p>© Copyright</p>
  </div>
</div>
</body>
```

</html>

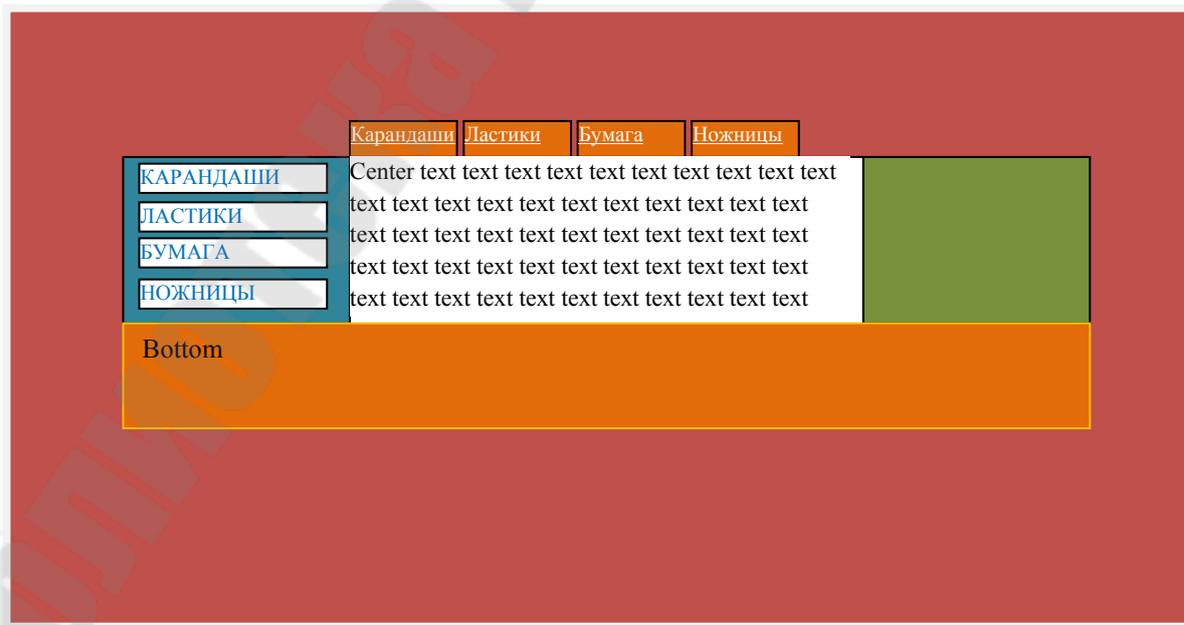
## Практическое задание

Создать следующие две панели, показанные ниже.

### Панель 1



### Панель 2



## Вопросы для контроля по лабораторной работе № 8

1. Почему таблицы стилей называют каскадными?
2. Какой тип CSS следует использовать, если предполагается изменение формата сразу нескольких страниц?
3. Какой из трех типов CSS следует считать наименее перспективным?
4. Как выполнить группировку селекторов?

## Литература

1 Нэш, К. Война Браузеров // эл. журнал: Сети / network world – № 01 – 1998.

2 Петюшкин, А. В. HTML в Web-дизайне // СПб.: БХВ-Петербург – 2004. – 400 с.: ил.

3 Дронов, В. HTML 5, CSS 3 и Web 2. Разработка современных web-сайтов / В. Дронов // СПб.: БХВ-Петербург, 2011. – 416 с.: ил. – (Профессиональное программирование).

4 Загуменов, А.П. Как раскрутить и разрекламировать Web-сайт в сети Интернет / А.П. Загуменов // М.: ДМК Пресс, 2005. – 384 с., ил.

## Оглавление

Лабораторная работа № 1 Основные понятия и определения языка HTML .....	3
Лабораторная работа № 2 Создание и использование гиперссылок. карты изображений ..	13
Лабораторная работа № 3 Создание списков .....	25
Лабораторная работа № 4 Создание таблиц. вложенные таблицы.....	29
Лабораторная работа № 5 Формы.....	34
Лабораторная работа №6 Форматирование текста с помощью css .....	39
Лабораторная работа №7 Позиционирование. создание объемного текста .....	44
Лабораторная работа №8 Создание панелей. вёрстка веб-страницы .....	48
Литература .....	58

**Тихоненко Татьяна Владимировна  
Рябченко Александр Иванович**

## **ВЕРСТКА WEB-СТРАНИЦ**

**Лабораторный практикум  
по одноименной дисциплине  
для слушателей специальности 1-40 01 74  
«Web-дизайн и компьютерная графика»  
заочной формы обучения**

Подписано к размещению в электронную библиотеку  
ГГТУ им. П. О. Сухого в качестве электронного  
учебно-методического документа 30.05.13.

Рег. № 75Е.  
<http://www.gstu.by>