Оглавление

[Лабораторная работа №1 3](#_Toc99358599)

[**Введение в HTML и CSS** 3](#_Toc99358600)

[Язык HTML 3](#_Toc99358601)

[Что такое HTML теги? 3](#_Toc99358602)

[Атрибуты 3](#_Toc99358603)

[Язык CSS 4](#_Toc99358604)

[Основная структура страницы HTML 4](#_Toc99358605)

[**Работа с основными HTML тегами** 5](#_Toc99358606)

[Абзацы 5](#_Toc99358607)

[Заголовки страницы 5](#_Toc99358608)

[**Жирный** 6](#_Toc99358609)

[**Курсив** 6](#_Toc99358610)

[Замечания 7](#_Toc99358611)

[**Упорядоченные и неупорядоченные списки в HTML** 7](#_Toc99358612)

[**Списки с номерами** 8](#_Toc99358613)

[**Замечания** 8](#_Toc99358614)

[**Лабораторная работа № 2** 9](#_Toc99358615)

[**Работа со ссылками на HTML** 9](#_Toc99358616)

[**Ссылки на страницы вашего сайта** 9](#_Toc99358617)

[**Работа с изображениями на HTML** 10](#_Toc99358618)

[**Ширина картинки** 10](#_Toc99358619)

[**Высота картинки** 11](#_Toc99358620)

[**Одновременно ширина и высота** 11](#_Toc99358621)

[Лабораторная работа №3 13](#_Toc99358622)

[**Работа с таблицами на HTML** 13](#_Toc99358623)

[**Ширина и высота таблицы** 14](#_Toc99358624)

[**Ячейки-заголовки** 15](#_Toc99358625)

[Создать станицу сайта. Разметить так, как указано в примере: 17](#_Toc99358626)

[**Главный заголовок страницы** 17](#_Toc99358627)

[**Второстепенный заголовок 1** 17](#_Toc99358628)

[**Второстепенный заголовок 2** 17](#_Toc99358629)

[**Второстепенный заголовок 3** 17](#_Toc99358630)

[Создать станицу сайта. Разметить так, как указано в примере: 17](#_Toc99358631)

[**Что такое CMS** 18](#_Toc99358632)

[**Какие бывают cms** 18](#_Toc99358633)

[**Примеры cms** 18](#_Toc99358634)

[**Что нужно знать, чтобы делать сайты** 18](#_Toc99358635)

[**PHP и JavaScript** 18](#_Toc99358636)

[**Виды скриптов** 18](#_Toc99358637)

[Лабораторная работа №4 20](#_Toc99358638)

[**Работа с формами** 20](#_Toc99358639)

[Приложение 1 25](#_Toc99358640)

Лабораторная работа №1

**Введение в HTML и CSS**

Для того, чтобы сделать сайт, нужно знать много разных веб языков.

Языки HTML и CSS предназначены для верстки сайтов (верстка - это размещение элементов сайта по нужным местам). Язык PHP нужен для программирования сайта (с его помощью можно, к примеру, сделать регистрацию пользователей). Язык JavaScript нужен для того, чтобы "оживить" сайт: к примеру, сделать меняющиеся картинки (слайдер).

Язык HTML

Язык HTML - это основа web сайтов, с его помощью создается каркас страницы, которую вы видите в браузере. Если сравнивать страницу сайта и обычную бумажную книгу, то на сайте, как и в книге, есть абзацы и заголовки.

В книге есть название всей книги (по сути самый главный заголовок), есть названия глав, параграфов в этих главах и так далее. Заголовки, абзацы и другие блоки можно выделить и на странице сайта. Это делается с помощью HTML *тегов*.

Что такое HTML теги?

HTML теги - это специальные команды для браузера. Они говорят ему, что, к примеру, следует считать заголовком страницы, а что абзацем.

Теги строятся по такому принципу: уголок <, потом имя тега, а потом уголок >. Имя тега может состоять из английских букв и цифр. Примеры тегов:

<h1>

<p>

<b>

Теги обычно пишутся парами - открывающий тег и соответствующий ему закрывающий. Разница между открывающим и закрывающим тегами в том, что в закрывающем теге после уголка < стоит слеш /.

К примеру, <p> - так я открыл тег, а так - </p> - я его закрыл. Все, что попадает между открывающим и закрывающим тегами, подпадает под воздействие нашего тега.

Бывают теги, которые не нужно закрывать, например, <br> или <img>.

Атрибуты

В тегах также могут размещаться *атрибуты* - специальные команды, которые расширяют действие тега. Атрибуты размещаются внутри открывающего тега в таком формате:

<тег атрибут1="значение" атрибут2="значение">

Кавычки могут быть любыми - одинарными или двойными, допустимо их вообще их не ставить, если значение атрибута состоит из одного слова (но это не желательно).

Язык CSS

Язык CSS расширяет возможности языка HTML. Он позволяет менять цвета, шрифты, фон, в общем заниматься красотой сайта. А HTML, соответственно, отвечает за структуру сайта.

Основная структура страницы HTML

Страница сайта - это обычный текстовый файл с расширением .html. Внутри этого файла и хранится текст HTML страницы вместе с тегами. Этот файл обязательно должен иметь следующие теги: тег <html>, который должен содержать в себе текст всего сайта (все, что написано вне этого тега, браузером будет проигнорировано), а внутри него должны быть еще два тега: тег <head> для служебного содержимого страницы и тег <body> - для основного текста, который и виден на экране браузера.

В служебное содержимое, которое располагается внутри тега <head>, входит много различных вещей, но пока нам нужны только две из них. Это тег <title>, задающий название страницы, которое будет видно во вкладке браузера, и тег <meta>, который задает кодировку страницы (она ставится в атрибуте charset и обычно имеет значение utf-8).

Кроме того, перед тегом <html> обычно пишется конструкция doctype, которая указывает версию языка HTML, на которой сделан сайт. Актуальная сейчас версия языка имеет номер пять и доктайп для нее должен выглядеть так - <!DOCTYPE html>.

Итак, давайте посмотрим на основную структуру страницы:

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Это заголовок тайтл</title>

</head>

<body>

Это основное содержимое страницы.

</body>

</html>

**Задача 1**

Создайте файл с расширением .html. Скопируйте в него основную структуру страницы. Сохраните ваш файл. Затем откройте его в браузере перетянув файл мышкой или выполнив двойной клик по нему.

**Работа с основными HTML тегами**

Давайте теперь изучим основные теги, которые используются на странице. В примерах далее я для простоты не буду приводить структуру всей страницы, считайте, что мы всегда работаем внутри тега body, если не указано иное.

Абзацы

Одним из основных элементов страницы являются *абзацы*. Их можно сравнить с абзацами в книге - каждый абзац начинается с новой строки и имеет так называемую красную строку (это когда первая строка текста абзаца немного отступает вправо). По умолчанию красной строки нет, но ее легко сделать (об этом чуть позже).

Абзац создается с помощью тега <p> таким образом:

<p>

Это абзац.

</p>

<p>

Это еще один абзац.

</p>

<p>

И еще один абзац.

</p>

Задача 1

Из приложения 1 возьмите текст. Вставьте этот текст между егами <body>.Посмотрите, что получилось. Теперь с помощью тегов <p></p> разметьте ваш текст. Посмотрите, как эти абзацы будут выглядеть в браузере.

Заголовки страницы

Кроме абзацев важное значение на странице имеют *заголовки*. Их также можно сравнить с заголовками из книги - каждая глава имеет свой заголовок (название этой главы) и разбита на параграфы, которые тоже имеют свои заголовки. Ну, а основной текст страницы располагается в абзацах.

Заголовки создаются с помощью тегов <h1>, <h2>, <h3>, <h4>, <h5>, <h6>. Они имеют разную степень важности. В заголовке h1 следует располагать название всей HTML страницы, в h2 - название блоков страницы, в h3 - название подблоков и так далее.

Все заголовки по умолчанию жирные и имеют разный размер (это можно поменять через CSS, но об этом позже). Для примера сделаем заголовки всех уровней:

<h1>Заголовок h1</h1>

<h2>Заголовок h2</h2>

<h3>Заголовок h3</h3>

<h4>Заголовок h4</h4>

<h5>Заголовок h5</h5>

<h6>Заголовок h6</h6>

Задача 2

Скопируйте приведенный выше код себе и запустите его в браузере. Теперь разметьте с помощью тегов заголовков заголовки в вашем тексте.

Сделайте заголовок h1. Ниже сделайте заголовок h2, а под ним - абзац с текстом. После абзацев сделайте еще один заголовок h2, а под ним - 2 абзаца с текстом.

**Жирный**

Вы уже знаете, что заголовки по умолчанию **жирные**. Однако, можно сделать жирным и обычный текст - достаточно взять его в тег <strong></strong> или<b>(устарел). Смотрите пример:

<p>

Это обычный текст, а это <b>жирный</b> текст.

</p>

Задача 4

Проверьте работу тега <b> на вашей странице.

**Курсив**

Кроме жирного можно сделать также и *курсив* с помощью тега <em></em> или <i> (устарел):

<p>

Это обычный текст, а это <i>курсивный</i> текст.

</p>

Задача 5

Проверьте работу тегов <strong></strong> и <em></em> на вашей странице.

Замечания

Обратите внимание на то, что теги  <strong></strong> и <em></em> следует использовать внутри какого-либо другого тега, например абзаца. В этом случае абзацы создают общую структуру страницы (абзацы и заголовки), а указанные теги делают жирным или курсивом отдельные кусочки текста.

**Упорядоченные и неупорядоченные списки в HTML**

Наряду с абзацами и заголовками существует еще один важный элемент страницы - это *списки*. Такие элементы наверняка знакомы всем пользователям интернета. Они представляют собой перечисление чего-либо (список) по пунктам. Возле каждого пункта списка обычно стоит закрашенный кружок (он называется *маркер* списка).

Списки создаются с помощью тега <ul>, внутри которого обязательно должны идти теги <li>. Тег ul задает сам список, а в теги li следует размещать пункты списка (то есть одному li соответствует один маркер списка). Смотрите пример:

<ul>

<li>пункт списка</li>

<li>пункт списка</li>

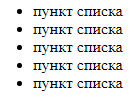
<li>пункт списка</li>

<li>пункт списка</li>

<li>пункт списка</li>

</ul>

Результат выполнения кода:



Задача 1



**Списки с номерами**

Списки, созданные через тег ul, называются *неупорядоченными* списками. Такое название им дано потому, что существуют еще и *упорядоченные* списки, у которых вместо маркеров-точек будут маркеры-цифры. Такие списки вместо тега ul имеют тег <ol>, а пункты таких списков также создаются через теги li.

Удобство упорядоченных списков в том, что я могу вставить новый пункт списка в любое место - и нумерация сама перестроится (то есть вам не приходится за ней следить в случае каких-либо изменений, как было бы в том случае, если бы вы расставляли ее вручную).

Давайте сделаем упорядоченный список, используя тег ol:

<ol>

<li>пункт списка</li>

<li>пункт списка</li>

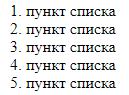
<li>пункт списка</li>

<li>пункт списка</li>

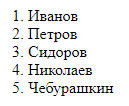
<li>пункт списка</li>

</ol>

Результат выполнения кода:



Задача 2



Также на вашей странице есть параграфы с нумерацией – превратите из в списки.

**Замечания**

Отдельно теги li использовать нельзя, только в составе списков. Кроме того, нельзя вкладывать списки в абзацы и наоборот. Списки в списки вкладывать можно, такое практикуется.

**Лабораторная работа № 2**

**Работа со ссылками на HTML**

Сейчас мы с вами будем разбираться со *ссылками*. Ссылки являются теми элементами, которые делают из интернета интернет. Нажимая на ссылки, мы можем переходить с одной страницы сайта на другую. Если бы их не было - интернет был бы просто набором страниц, никак не связанных друг с другом.

Ссылка создается с помощью тега <a>. В этом теге обязательно должен быть атрибут href, в который следует записывать адрес той страницы, на которую ведет ссылка.

Давайте для примера сделаем ссылку на гугл. Для этого нужно открыть этот сайт в браузере, скопировать адрес страницы из адресной строки и вставить в атрибут href ссылки:

<a href="https://www.google.com">ссылка</a>

Результат выполнения кода:



Задача 1

Сделайте на вашей странице ссылку на яндекс и ссылку на ютуб.

**Ссылки на страницы вашего сайта**

Выше мы делали ссылки на страницы чужих сайтов, копируя их адрес из адресной строки браузера. Пусть теперь вы хотите связать ссылками страницы своего сайта.

В таком случае копировать их адреса из адресной строки нельзя. Дело в том, что такие адреса будут содержать полный путь к вашей странице от корневого диска вашего компьютера, и при переносе в другую папку такие ссылки просто сломаются.

Для страниц своего сайта следует практиковать другой подход. Изучим его на примере. Пусть в корневой папке вашего сайта расположен файл index.html и файл page.html. Свяжем эти файлы ссылками. Для этого в атрибуте href ссылок следует просто написать имена этих файлов (с их расширением). Давайте сделаем это.

На странице index.html сделаем ссылку на страницу page.html:

<h1>страница index</h1>

<a href="page.html">ссылка на page</a>

А на странице page.html сделаем ссылку на страницу index.html:

<h1>страница page</h1>

<a href="index.html">ссылка на index</a>

Задача 2

Сделайте три HTML страницы в корне вашего сайта. Назовите их 1.html, 2.html, 3.html. Свяжите эти страницы ссылками друг с другом.

Задача 3

Назовите вашу главную страницу index.html. Разместите на ней ссылки на страницы 1.html, 2.html, 3.html.

Задача 4

Модифицируйте предыдущую задачу так, чтобы ссылки располагались в списке ul.

**Работа с изображениями на HTML**

Давайте теперь разберемся с тем, как разместить **изображение** на странице вашего сайта. Для этого предназначен тег <img>, имеющий обязательный атрибут src, в котором следует задавать путь к файлу картинки. При этом сам тег не требует закрывающего тега.

Давайте посмотрим на примере. Пусть в корневой папке нашего сайта находится файл smile.png с картинкой. Давайте выведем ее на странице нашего сайта:

<img src="smile.png">

Результат выполнения кода:



Задача 1

Разместите на странице вашего сайта какую-нибудь картинку.

**Ширина картинки**

По умолчанию картинки отображаются своим реальным размером. Например, та картинка со смайлом, которую мы выводили выше, имеет реальный размер 131 на 121 пикселей и такой же размер картинки мы и видим в браузере.

При желании можно изменить размер картинки, задав ей ширину. Это делается с помощью атрибута width

Давайте для примера зададим нашей картинке ширину 100 пикселей. Для этого значением атрибута width укажем число 100, вот так: width="100".

Какого же размера будет картинка на экране? Можно подумать, что следующая: ширина 100 пикселей, как мы задали, и реальная высота 130, так как мы ее не задавали. На самом деле, это будет не так.

Ширина действительно будет 100 пикселей, а вот высота подстроится так, чтобы пропорции картинки не исказились, то есть также станет 100 пикселей.

Давайте запустим и убедимся в этом:

<img src="smile.png" width="100">

Результат выполнения кода:



Задача 2

Установите вашей картинке ширину в 300 пикселей.

**Высота картинки**

Можно вместо атрибута width написать атрибут height - в этом случае уже высота будет задана как 100 пикселей, а ширина подстроится под нее так, чтобы не исказить пропорции:

<img src="smile.png" height="100">

Результат выполнения кода:



Задача 3

Установите вашей картинке высоту в 100 пикселей.

**Одновременно ширина и высота**

Если задать одновременно и ширину, и высоту, то картинку можно исказить. Например, если задать ширину 100 пикселей, а высоту 70px, то наш смайл станет растянутым по ширине:

<img src="smile.png" width="100" height="70">

Результат выполнения кода:

Картинка не обязательно исказится, ведь мы можем просто задать и ширину, и высоту по 100 пикселей, в этом случае наша картинка станет меньше, чем была (а была она 130), но пропорции ее не исказятся:

<img src="smile.png" width="100" height="100">

Результат выполнения кода:



Задача 4

Установите вашей картинке ширину и высоту так, чтобы пропорции изображения исказились.

Задача 5

Установите вашей картинке ширину и высоту так, чтобы пропорции изображения НЕ исказились.

Задача 6

Теперь сделайте вашу картинку ссылкой.

# Лабораторная работа №3

**Работа с таблицами на HTML**

Сейчас мы с вами научимся делать таблицы на HTML. Я думаю, вы представляете, что такое таблица в обычной жизни - это набор строк и столбцов, на пересечении которых находятся ячейки.

В HTML таблицы создаются по похожему принципу. Там тоже есть столбцы и строки с ячейками, однако HTML код таблиц при первом взгляде может показаться непривычным: таблицы создаются по рядам - сначала первый ряд, потом второй и так далее.

Код таблицы имеет жесткую структуру: главным является тег table, внутри которого должны лежать теги tr, которые создают ряды таблицы, а внутри них - теги td, которые создают ячейки. Тег table может иметь атрибут border, который задает границу таблице и ее ячейкам.

Давайте для примера сделаем таблицу с тремя рядами и тремя ячейками в каждом ряду:

<table border="1">

<!--Это будет первый ряд таблицы:-->

<tr>

<td>ячейка 1</td>

<td>ячейка 2</td>

<td>ячейка 3</td>

</tr>

<!--Это будет второй ряд таблицы:-->

<tr>

<td>ячейка 4</td>

<td>ячейка 5</td>

<td>ячейка 6</td>

</tr>

<!--Это будет третий ряд таблицы:-->

<tr>

<td>ячейка 7</td>

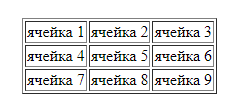
<td>ячейка 8</td>

<td>ячейка 9</td>

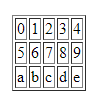
</tr>

</table>

Результат выполнения кода:



Задача 1



**Ширина и высота таблицы**

С помощью атрибутов width и height можно задать ширину и высоту таблице. Зададим, например, нашей таблице ширину в 300 пикселей, а высоту - в 200:

<table border="1" width="300" height="200">

<tr>

<td>ячейка 1</td>

<td>ячейка 2</td>

<td>ячейка 3</td>

</tr>

<tr>

<td>ячейка 4</td>

<td>ячейка 5</td>

<td>ячейка 6</td>

</tr>

<tr>

<td>ячейка 7</td>

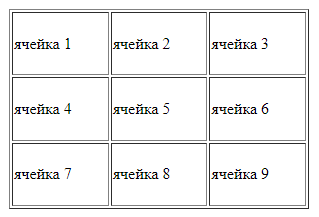
<td>ячейка 8</td>

<td>ячейка 9</td>

</tr>

</table>

Результат выполнения кода:



Задача 2

Добавьте вашей таблице ширину 400 пикселей и высоту 300.

**Ячейки-заголовки**

Кроме тегов td существуют также теги th, которые также создают ячейки. Но это уже будут не обычные ячейки, а *ячейки-заголовки*, которые указывают, что находится в данном столбце (или строке) таблицы. По умолчанию текст в таких ячейках th будет жирный и расположен по центру.

Давайте посмотрим на примере. Пусть у нас есть вот такая таблица с работниками:

<table border="1" width="300">

<tr>

<td>Иван</td>

<td>Иванов</td>

<td>200$</td>

</tr>

<tr>

<td>Николай</td>

<td>Петров</td>

<td>300$</td>

</tr>

<tr>

<td>Владимир</td>

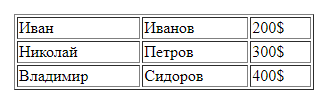
<td>Сидоров</td>

<td>400$</td>

</tr>

</table>

Результат выполнения кода:



Как вы видите, в первом столбце хранится имя работника, во втором столбце - фамилия, а в третьем - зарплата. Давайте сделаем еще один ряд в начале таблице, в котором разместим заголовки колонок в тегах th:

<table border="1" width="300">

<tr>

<th>Имя</th>

<th>Фамилия</th>

<th>Зарплата</th>

</tr>

<tr>

<td>Иван</td>

<td>Иванов</td>

<td>200$</td>

</tr>

<tr>

<td>Николай</td>

<td>Петров</td>

<td>300$</td>

</tr>

<tr>

<td>Владимир</td>

<td>Сидоров</td>

<td>400$</td>

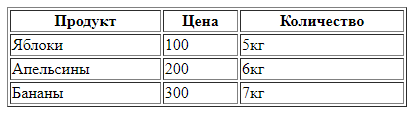
</tr>

</table>

Результат выполнения кода:



Задача 3



**САМОСТОЯТЕЛЬНО**

Задача 1

Создать станицу сайта. Разметить так, как указано в примере:

**Главный заголовок страницы**

Абзац с текстом.

**Второстепенный заголовок 1**

Абзац с текстом.

Абзац с текстом.

Абзац с текстом.

**Второстепенный заголовок 2**

Абзац с текстом.

Абзац с текстом.

Абзац с текстом.

**Второстепенный заголовок 3**

Абзац с текстом.

Абзац с текстом.

Абзац с текстом.

Задача 2

Создать станицу сайта. Разметить так, как указано в примере:

**Что такое CMS**

**CMS** - "система управления контентом" (**движок**) - написанная PHP-программистами основа для сайта, с помощью которой вы сможете управлять сайтом (добавлять контент, менять пункты меню и т.п.) не зная HTML и CSS.

Однако, для того чтобы сделать сайт с помощью **CMS** *потребуются услуги* и программиста, и дизайнера, и верстальщика. И капиталовложения.

**Какие бывают cms**

Бывают различные системы управления контентом: для интернет-магазинов, для блогов, для форумов и т.д.

**Примеры cms**

*Примеры популярных CMS*: Joomla, WordPress (для блогов), PhpBB (для форумов).

**CMS-ки** бывают *платные* и *бесплатные*.

Задача 3

Создать станицу сайта. Разметить так, как указано в примере:

**Что нужно знать, чтобы делать сайты**

1. **HTML**
2. *CSS*
3. PHP
4. SQL
5. JavaScript
6. jQuery
7. Flash
8. SEO

**PHP и JavaScript**

Языки программирования **PHP** и **JavaScript** позволяют сделать сайт динамичным, то есть реагирующим на действия пользователя. Например, можно сделать красивую выпадающую менюшку или слайдер.

**Виды скриптов**

Для этого пишутся скрипты (англ. *script* - "сценарий") - программы, позволяющие реагировать на действия пользователя. Скрипты бывают двух видов:

* те, которые выполняются на сервере, а результат их выполнения приходит в браузер к пользователю уже в готовом виде. Это скрипты, написанные на языке **PHP**. На нем пишуться **CMS-ки** – системы управления контентом.
* те, которые выполняются прямо в браузере пользователя. Это скрипты, написанные на языке **JavaScript**. Они чаще всего используются для, того чтобы сделать страницу более удобной и красивой.

# Лабораторная работа №4

**Работа с формами**

***Цель работы:*** Изучить основы организации форм в HTML.

***Базовые сведения:***

Формы были созданы и используются в WWW для получения отклика пользователя на предоставленную информацию и сбора данных о пользователе. После заполнения пользователем формы и запуска процесса ее обработки информация из нее попадает к программе, работающей на сервере. Простота использования тега <MAILTO:> в формах позволяет даже владельцам небольших страниц получать отклик от своих читателей. Для обработки большого количества откликов используются программы, поддерживающие Common Gateway Interface (CGI) и расположенные на сервере, в адрес которого поступают отклики. Таким образом пользователь может интерактивно взаимодействовать с Web-сервером через Internet.

**Задание формы — элемент FORM**

Элемент FORM обозначает документ как форму и определяет границы использования других тегов, размещаемых в форме. Тег <FORM> определяется последовательностью тегов <INPUT>, размещенных внутри пары <FORM> и </FORM>. В форме используется как метод (method), так и действие (action) для описания обработки данных, вводимых пользователем в форму. Метод (GET или POST) определяет, как должны обрабатываться входные данные из формы, а действие указывает на URI программы, ответственной за обработку этих данных.

<FORM METHOD="post" ACTION="mailto:yourname@your.email.address">

**Определение элементов управления формы — тег <INPUT>**

Данный тег используют для определения области внутри формы, куда вводятся данные. Он формирует поле для ввода информации пользователем. Это может быть текстовое поле, опция, изображение или кнопка. Вид поля ввода определяется значением атрибута TYPE.

**Атрибут TYPE=text**

Когда пользователю необходимо ввести небольшое количество текста (одну или несколько строк), используется тег <INPUT>, и атрибут TYPE устанавливается в значение text. Это значение принято по умолчанию и указывать его необязательно. Кроме того, задается атрибут NAME для определения наименования переменной поля.

Ваше имя <INPUT NAME="Name" SIZE="35">

Имеется еще три дополнительных атрибута, которые можно использовать. Первый называется MAXLENGTH, он ограничивает число символов, вводимых пользователем в текущее поле. По умолчанию данное число не ограничено. Вторым атрибутом является SIZE, определяющий размер видимой на экране области, занимаемой текущим полем. Значение по умолчанию определяется типом браузера. Если значение MAXLENGTH больше, чем SIZE, браузер будет прокручивать данные в окне. Последним из дополнительных атрибутов является атрибут VALUE, обеспечивающий начальное значение поля ввода.

**Атрибут TYPE=checkbox**

При создании форм часто требуется получить ответ пользователя на вопрос типа "Да/Нет". Для создания независимых кнопок в формах HTML используется тег <INPUT> с атрибутом TYPE=checkbox. В зависимости от содержания формы пользователь может отметить несколько флагов. Когда форма использует тег <INPUT> с атрибутом CHECKBOX, в нем должны присутствовать и атрибуты NAME, и VALUE. Атрибут NAME указывает на наименование данного поля (флага) ввода. В атрибуте VALUE будет содержаться значение поля.

Россия<INPUT NAME="Страна" TYPE=checkbox VALUE="Россия"> %

Страны СНГ<INPUT NAME="Страна" TYPE=checkbox VALUE="СНГ">

В некоторых случаях необходимо инициализировать данный флаг, как уже отмеченный. В таких случаях тег <INPUT> должен содержать атрибут CHECKED.

**Атрибут TYPE=radio**

В некоторых случаях требуется организовать выбор одного из нескольких возможных значений. Для создания формы ввода при выборе пользователем одного значения из нескольких возможных необходимо использовать тег <INPUT> с атрибутом TYPE=radio. Когда в форме применяется данный атрибут, в теге <INPUT> должны быть указаны атрибуты NAME и VALUE. Атрибут NAME указывает наименование соответствующего поля (кнопки). Атрибут VALUE содержит значение поля.

Пол мужской<INPUT NAME="Пол" TYPE=radio VALUE="Мужской"> %

Пол женский<INPUT NAME="Пол" TYPE=radio VALUE="Женский">

**Атрибут TYPE=image**

В зависимости от содержимого формы может случиться так, что пользователю потребуется щелкнуть мышью на изображении, чтобы завершить работу с формой. Для этого программисты используют тег <INPUT> с атрибутом TYPE=image. Когда пользователь щелкает мышью по изображению, браузер сохраняет координаты соответствующей точки экрана. Далее он "обрабатывает" введенную в форму информацию. Когда форма использует атрибут image, тег <INPUT> должен содержать также атрибуты NAME и SRC. NAME указывает наименование поля ввода формы. Атрибут SRC содержит URI файла — источника изображения. Атрибут ALIGN является дополнительным и используется аналогично тому же атрибуту тега <IMG>.

<BR>Выберите точку<INPUT TYPE=image NAME=point SRC="image.gif">

**Атрибут TYPE=password**

Если в форме требуется организовать ввод пароля, то атрибут TYPE можно установить в значение password (TYPE=password). Используя данный тип, можно организовать ввод пароля без вывода на экран составляющих его символов. При этом следует помнить, что введенные данные передаются по незащищенным каналам связи и могут быть перехвачены.

Подпись<INPUT NAME="login"> %

Пароль <INPUT TYPE=password NAME="pass">

**Атрибут TYPE=reset**

Когда пользователь заполняет форму, ему может потребоваться начать все сначала. На такой случай существует кнопка Reset, по которой пользователь может щелкнуть мышью, чтобы вернуться к первоначальным значениям полей. Когда пользователь выбирает данную кнопку, форма восстанавливает первоначальные значения всех элементов, в которых присутствует атрибут TYPE=reset. Для создания кнопки Reset используется тег <INPUT> с атрибутом TYPE=reset. Браузер в свою очередь будет выводить изображение данной кнопки. Если в форме используется атрибут reset, тег <INPUT> может дополнительно содержать атрибут VALUE. Данный атрибут определяет надпись на изображении кнопки.

<INPUT TYPE=reset VALUE="Очистить форму">

**Атрибут TYPE=submit**

Используя форму HTML для ввода информации от пользователя, необходимо обеспечить пользователю возможность завершить ввод данных. Для этого используется тег <INPUT> с атрибутом TYPE=submit. Браузер, в свою очередь, выводит данный элемент, как кнопку, по которой пользователь может щелкнуть, чтобы завершить процесс редактирования. Когда в форме используется тег <INPUT> с атрибутом submit, данный элемент может содержать два дополнительных атрибута: NAME и VALUE. Атрибут NAME хранит значение переменной поля в вашей форме. Атрибут VALUE — указывает наименование кнопки Submit.

<BR><INPUT TYPE=submit VALUE="Отправить сообщение"> %

**Атрибут TYPE=hidden**

Скрытые поля. Добавление в тег INPUT атрибута TYPE=hidden позволит включить в отправляемую форму значения атрибутов NAME и VALUE, которые пользователь изменить не может. Такие метки полезны при наличии нескольких форм для дальнейшей обработки данных.

**Создание многострочных областей ввода текста — тег <TEXTAREA>**

В зависимости от типа формы может потребоваться организовать ввод большого количества текста. В таких случаях используется тег <TEXTAREA> для создания текстового поля из нескольких строк. Данный тег использует три атрибута: COLS, NAME и ROWS.

**Атрибут COLS** Указывает (число символов) число колонок, содержащихся в текстовой области.

**Атрибут NAME** Определяет наименование поля.

**Атрибут ROWS** Задает количество видимых строк текстовой области.

<TEXTAREA NAME=тема COLS=38 ROWS=3> </TEXTAREA>

**Использование списков в форме — тег <SELECT>**

Когда формы HTML становятся более сложными, в них часто включают списки с прокруткой и выпадающие меню. Для этого используют тег SELECT с атрибутом TYPE=select. Для определения списка пунктов используют тег <OPTION>. Тег <SELECT> поддерживает три необязательных атрибута: MULTIPLE, NAME и SIZE.

**Атрибут MULTIPLE** Позволяет выбрать более чем одно наименование.

**Атрибут NAME** Определяет наименование объекта.

**Атрибут SIZE** Определяет число видимых пользователю пунктов списка. Если в форме установлено значение атрибута SIZE=1, то браузер выводит на экран список в виде выпадающего меню. В случае SIZE > 1 браузер представляет на экране обычный список.

В форме может использоваться тег <OPTION> только внутри тега <SELECT>. Эти теги поддерживают два дополнительных атрибута: SELECTED и VALUE.

**Атрибут SELECTED** Используется для первоначального выбора значения элемента по умолчанию.

**Атрибут VALUE** Указывает на значение, возвращаемое формой после выбора пользователем данного пункта. По умолчанию значение поля равно значению тега <OPTION> (открыть).

Выбор <SELECT NAME="Выбор"> %

<OPTION>Вариант 1 </OPTION>%

<OPTION>Вариант 2 </OPTION>%

<OPTION VALUE="Вариант 3">Вариант 3 </OPTION>%

<OPTION SELECTED>Вариант 4 </OPTION>%

</SELECT>

***Задание на лабораторную работу***

1. Создайте форму следующего вида:

|  |
| --- |
| Image img0401 |

* **Лабораторная работа № 5**
* **HTML+CSS**
* Цель: изучить основные способы подключения CSS; возможности оформления текста, шрифта, фона на web-странице; научиться использовать селекторы классов и идентификаторов.

**Краткая теория**

Подключение CSS.

Внешнее. Самый мощный и удобный способ определения стилей и правил для сайта. Стили хранятся в отдельном файле, который может быть использован для любых веб-страниц. Для подключения таблицы связанных стилей используется тег <link>в заголовке страницы.

Внутреннее. Стиль определяется в самом документе и обычно располагается в заголовке веб-страницы. По своей гибкости и возможностям этот способ использования стиля уступает предыдущему, но также позволяет размещать все стили в одном месте. В данном случае, прямо в теле документа. Определение стиля задается тегом <style>.

Строковое. Внутренний стиль являются по существу расширением для одиночного тега, используемого на веб-странице. Для определения стиля используется атрибут style, а его значения указываются с помощью языка таблицы стилей.

Текст.

Цвет текста задается с помощью свойства color. Можно задавать разными способами: по шестнадцатеричному значению, по названию, в формате RGB, RGBA, HSL, HSLA.

Горизонтальное выравнивание текста в пределах элемента определяет свойство text-align.

Значения:

- сenter – выравнивание текста по центру, текст помещается по центру горизонтали окна браузера или контейнера, где расположен текстовый блок;

- justify – выравнивание по ширине, что означает одновременное выравнивание по левому и правому краю;

- left – выравнивание текста по левому краю, строки текста выравнивается по левому краю, а правый край располагается «лесенкой»;

- right – выравнивание текста по правому краю, этот способ выравнивания выступает в роли антагониста предыдущему типу;

- auto – не изменяет положение элемента;

- inherit – наследует значение родителя.

Отступ между словами и буквами в тексте:

- word-spacing – устанавливает интервал между словами; если для текста задано выравнивание через text-align со значением justify (выравнивание по ширине), то интервал между словами будет установлен принудительно, но не меньше значения, указанного через word-spacing;

- letter-spacing – определяет интервал между символами в пределах элемента. Значения устанавливаются так же, как и в предыдущем свойстве.

Свойство text-decoration добавляет оформление текста в виде его подчеркивания, перечеркивания, линии над текстом и мигания. Одновременно можно применить более одного стиля, перечисляя значения через пробел.

Значения:

- blink – устанавливает мигающий текст. Такой текст периодически, примерно раз в секунду исчезает, потом вновь появляется на прежнем месте.

- line-through – создает перечеркнутый текст;

- overline – линия проходит над текстом;

- underline – устанавливает подчеркнутый текст;

- none – отменяет все эффекты, в том числе и подчеркивания у ссылок, которое задано по умолчанию.

Шрифт

Свойство font-family устанавливает семейство шрифта, которое будет использоваться для оформления текста содержимого. Список шрифтов может включать одно или несколько названий, разделенных запятой. Если в имени шрифта содержатся пробелы, например, Trebuchet MS, оно должно заключаться в одинарные или двойные кавычки.

Cвойство font-size определяет размер шрифта элемента. Разрешается использовать любые допустимые единицы CSS: em (высота шрифта элемента), ex (высота символа х), пункты (pt), пикселы (px), проценты (%) и др. За 100% берется размер шрифта родительского элемента. Отрицательные значения не допускаются.

Свойство font-style – определяет начертание шрифта (обычное, курсивное или наклонное).

Значения:

- normal – обычное начертание текста;

- italic – курсивное начертание;

- oblique –наклонное начертание,курсив и наклонный шрифт при всей их похожести не одно и то же.

Свойство font-weiqht устанавливает насыщенность шрифта. Значение устанавливается от 100 до 900 с шагом 100. Сверхсветлое начертание, которое может отобразить браузер, имеет значение 100, а сверхжирное – 900. Нормальное начертание шрифта (которое установлено по умолчанию) эквивалентно 400, стандартный полужирный текст – значению 700.

Насыщенность шрифта задается с помощью ключевых слов: bold – полужирное начертание, normal – нормальное начертание. Также допустимо использовать условные единицы от 100 до 900. Значения bolder и lighter изменяют жирность относительно насыщенности родителя, соответственно, в большую и меньшую сторону.

Фон.

Свойство background-attachment устанавливает, будет ли прокручиваться фоновое изображение вместе с содержимым элемента. Изображение может быть зафиксировано и оставаться неподвижным, либо перемещаться совместно с документом.

Свойство background-color определяет цвет фона элемента.

Свойство background-image устанавливает фоновое изображение для элемента. Если одновременно для элемента задан цвет фона, он будет показан, пока фоновая картинка не загрузится полностью. В качестве значения используется путь к графическому файлу, который указывается внутри конструкции url(). Путь к файлу при этом можно писать как в кавычках (двойных или одинарных), так и без них.

Свойство background-position задает начальное положение фонового изображения, установленного с помощью свойства background-image.

У свойства background-position два значения, положение по горизонтали (может быть – left, center, right) и вертикали (может быть – top, center, bottom). Кроме использования ключевых слов положение также можно задавать в процентах, пикселах или других единицах. Если применяются ключевые слова, то порядок их следования не имеет значения, при процентной записи вначале задается положение рисунка по горизонтали, а затем, через пробел, положение по вертикали.

Свойство background-repeat определяет, как будет повторяться фоновое изображение, установленное с помощью свойства background-image. Можно установить повторение рисунка только по горизонтали, по вертикали или в обе стороны.

**Задания к лабораторной работе № 5**

Задания выполняются со своими HTML-документами, созданными ранее.

Задание 1.

а) Стили для оформления заголовков на web-страницах описать во внешнем файле. В одном правиле задать одинаковый цвет и вид шрифта для заголовков h1, h2, h3

б) Выделить в тексте несколько предложений (любых на выбор) тегом span и задать для этих предложений с помощью классов жирность, курсив и в одном из выделенных предложений перечеркнутый текст.

Задание 2. Абзацы текста. Задать классы для абзацев. Для каждого абзаца создать свой стиль, в котором использовать перечисленные выше свойства текста, шрифта. У каждого абзаца должно быть разное выравнивание, отступ между словами и буквами, разный вид и размер шрифта. Продемонстрировать разные способы (три) указания цвета текста и размера шрифта.

Задание 3. Изменить параметры элементов span второго и третьего выделенного предложения (цвет текста, стиль шрифта, оформление). Использовать селекторы идентификаторов.

Задание 4. Для первой web-страницы задать цветной фон. Для второй web-страницы задать в качестве фона изображение, которое постоянно будет размещено в правом верхнем углу, используя краткую форму записи.

# Приложение 1

**How To Avoid Wasting Your Most Valuable Online Asset - TIME!**

If you're involved in an internet marketing business, then

you've probably already come face-to-face with your #1

enemy - "Time Wasting".

What you need to know is that "Time Is Money". And this

applies even more to the online world, than in regular

business.

You see, anyone can become an online entrepreneur. It

doesn't involve start-up capital or expensive equipment to

start. A low grade PC and a net connection is all you need.

And since cash isn't really a barrier, time-management has

become all the more critical. Your challenge for success is

simply to get more done - in less time.

So what are the greatest online time wasters and what can

you do to avoid them?

1. Retyping the same old thing time and time again.

2. Failing to manage your email effectively.

3. Getting sucked into pointless forum discussions.

4. Failing to set a goal, before you switch the computer on.

**Retyping the same old thing time and time again.**

------------------------------------------------

This sounds simple. But stop and think how many times you

retype the same old simple information.

Most people have cottoned on to automating their email

signature, but you can go far beyond that when it comes to

communicating words quickly.

Get yourself a clickboard or shortcut extender. This will

enable you to write much faster and prevent you from ever

having to type the same thing twice.

I use this utility http://www.howtocorp.com/typepilot , but

there are lots of other useful typing assistant utilities

available.

**Failing to manage your email effectively.**

-----------------------------------------

I know, there's so much spam flying around, but you should

be attempting to win this war with a few weapons of your

own installed on your computer.

i. Make sure you have a working piece of anti-spam software

set up. Whilst sometimes the wrong messages will get

trapped, the best software will still remove 98% of the

emails that you don't want. Make sure they are filtered

into a junk email file, so you can manually sift

through them afterwards and pull back any that have

been incorrectly assumed to be unwanted.

ii. Set up folders in your inbox and get yourself properly

organized. I keep a folder for SALES, another one for

FEEDBACK and others for JOINT VENTURES + many more.