**Описание первых семи лабораторных работ по курсу «Архитектура информационных систем (ИС)»**

1. ***Введение***

Современные ИС имеют самые разнообразные архитектуры. Одной из самых распространенных и, одновременно, одной из старейших, является архитектура «клиент - сервер».

Суть этой архитектуры состоит в том, что два компонента ИС, клиент и сервер, взаимодействуют таким образом, что клиент формирует запрос серверу и интерпретирует получаемый ответ, а сервер обрабатывает запрос клиента и выполняет обработку данных, формируя ответ. Оба этих компонента могут быть модулями одной программы, двумя различными процессами одной операционной системы, или даже двумя приложениями, выполняемыми на разных компьютерах (хостах), в этом случае возникает вопрос о правилах передачи данных, - транспортном протоколе.

Типичным случаем реализации клиент- серверной архитектуры является ситуация, когда сервер хранит специальным образом организованные данные (оформленные с использованием средств разметки и форматирования) , а клиент, получая такие данные, восстанавливает вид документа, задуманный его автором. При этом достигается платформенная независимость ИС, т.к. клиент и сервер могут работать на разных программно-аппаратных платформах.

Как известно, таким средством разметки и форматирования является язык HTML, а в качестве клиента может выступать приложение – браузер (известные всем Internet Explorer, Firefox, Opera, Chrome etc.).

Изучению возможностей, предоставляемых языком HTML и некоторых аспектов работы серверного ПО посвящен настоящий цикл лабораторных работ.

1. ***Введение в HTML***
   1. Что такое HTML

Если открыть любой html-документ на редактирование в обычном текстовом редакторе, то результат будет несколько отличаться от того, что все привыкли наблюдать, переключая страницы в интернете: перед нашим взором предстанет «сырой» текст без красочных картинок и форматирования. Этот текст и является кодом, записанным с использованием HTML, а он в свою очередь может интерпретироваться браузерами, которые преобразуют html-код, написанный web-программистом, в удобочитаемый вид. HTML (от англ. «HyperText Markup Language») – это язык разметки документов во Всемирной паутине.

* 1. Структура HTML-документа
     1. Теги

Любой документ на языке HTML представляет собой набор элементов, у каждого из таких элементов есть начало и конец, которые обозначаются с помощью специальных пометок – тегов. Элементы могут содержать какие-либо данные, которые заключаются между тегами начала и конца, а также возможен вариант, когда элемент не нуждается в дополнительных данных, в этом случае тег окончания опускается. Элементы, в записи которых указывается открывающий и закрывающий тег называются *контейнерными*, а элементы, состоящие только из открывающего тега, называются *одиночными* тегами.

Приведем несколько примеров:

* <p>Мой первый абзац, написанный с использованием HTML</p>

Эта строчка говорит о том, что мы определяем отдельный абзац в нашем документе. Текст абзаца заключен между специальным тегом <p>. Причем <p> – это тег открывающий, а </p> – тег закрывающий. Отличаются они в одном символе «/», и этот символ одинаков для всех закрывающих тегов. Всегда по наличию этой косой черты, можно определить какой тег открывающий, а какой закрывающий.

* <br />

Этот тег служит для переноса строки. Очевидно, что такому тегу не нужны дополнительные данные, поэтому он сам по себе самостоятельный элемент и в закрывающем теге не нуждается. Причем в записи этого тега бывают расхождения. Дело в том, что язык HTML постоянно развивается и улучшается, что приводит к тому, что для определенных контрольных точек в развитии этого языка вводится своя версия. Так существуют HTML 4 и 5 версии. Введение версий несколько осложняет алгоритм работы браузеров, и существует такая вероятность, что «свежие» по дате выпуска теги, будут отображаться не во всех браузерах корректно. Поэтому html-код желательно проверять во многих браузерах, а не только в одном. В зависимости от версии HTML корректной записью может быть как <br> так и <br />.

Внутри открывающих тегов могут дополнительно использовать атрибуты для более гибкой настройки форматирования (или прочих свойств). Например:

<img src=”photo.jpg”>

Тег «img» (от англ. «Image») предназначен для добавления на сайт картинки, но как указать какую именно картинку следует добавить? Для этого задается специальный атрибут этого тега «src» (от англ. «Source»), в котором указывается путь до файла с необходимым изображением. Атрибутов может быть несколько, а может и вообще не быть – все зависит от свойств того тега, который будет использоваться. Значение атрибутов зачастую всегда берутся в двойные кавычки, однако использовать их для числовых значений не обязательно.

*Примечание: Теги можно записывать в любом регистре – для единообразия будем использовать нижний регистр.*

* + 1. Каркас HTML-документа

На простейшем примере рассмотрим, какие теги должны обязательно присутствовать в каждом HTML-документе.

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Первый заголовок</title>

</head>

<body>

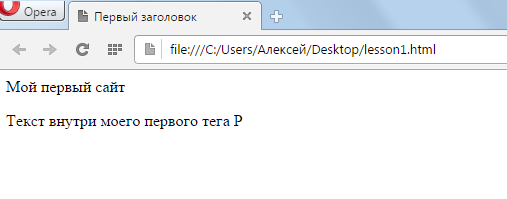
Мой первый сайт

<p>Текст внутри моего первого тега P</p>

</body>

</html>

Результат обработки этого текста браузером будет следующий:



Прежде чем перейти к разбору записанного кода, определимся со стилем написания и средой исполнения HTML-кода.

* + - 1. Сохранение файла и просмотр результатов в браузере

Для записи кода будем использовать обычный текстовый редактор (блокнот Windows подойдет). Но, разумеется, никто не ограничивает вас в выборе среды разработки – существуют и более удобные редакторы с подсветкой синтаксиса. В частности, к таким удобным редакторам относится свободно- распространяемое Notepad++

Для того, чтобы сохранить записанный текст, как html-документ, нужно выбрать из списка предлагаемых расширений \*.html или указать это расширение вручную, сбросив выбранное расширение на «All Files \*».

После того, как файл сохранен с расширением \*.html, он автоматически будет открываться с помощью браузера по умолчанию, если этого не происходит, то следует выбрать браузер в пункте контекстного меню «Открыть с помощью», которое появляется после нажатия на правую кнопку мыши по ярлыку сохраненного файла.

* + - 1. Стиль написания

Будем придерживаться следующих рекомендаций:

* Все теги записываются в едином регистре (верхнем или нижнем).
* Каждый контейнерный тег на отдельной строке.
* Тег, вложенный в другой тег, имеет отступ на одну табуляцию больше по сравнению с родительским тегом. Пример записи, оформленной в соответствии со «стилем»:

<html>

<body>

<p></p>

</body>

</html>

Уважайте себя и того, кто будет просматривать ваш код. В коде, написанным единим стилем, намного легче разбираться.

* + - 1. Обязательные теги

Вернемся к примеру и разберем все те теги, из которых он состоит.

* <!DOCTYPE html>. На самом деле, это не тег, а декларация, которая используется для обозначения версии, используемой при написании html-документа. Данная запись указывает браузерам на то, что при написании кода использовался HTML 5.
* <html></html>. Контейнерный тег, который задает область поиска для браузера. Все что находится за закрывающим тегом </html>, в большинстве браузеров будет проигнорировано.
* Любой html-документ состоит из двух частей: <head> и <body>, - которые указываются внутри контейнерного тега <html>.

Все то, что отображается на странице в браузере, находится внутри тега <body>. Это может быть текст, управляющие кнопки, ссылки, изображения и т.д., то есть те данные, которые должен увидеть пользователь. В нашем примере мы добавили две строчки с текстом. Одну написали прямо в теге <body>, а вторую добавили с помощью тега <p>, который отвечает за создание абзаца в документе.

*Примечание: Обратите внимание, что если опустить тег <p> и написать вторую строку без его использования внутри тега <body>:*

*<body>*

*Мой первый абзац*

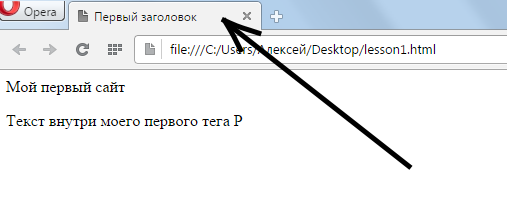
*Мой второй абзац*

*</body>*

*То обе эти строчки будут отображены браузером на одной строке. Это происходит потому, что все переносы строк, табуляции и пробелы сокращаются искусственно до одного пробела. Поэтому для этих целей необходимо либо использовать специальные теги, либо явно экранировать такие символы.*

Тег <head> предназначен для метаданных, в нем указывается такая вспомогательная информация, как: используемая на странице кодировка, подключаемые файлы (файлы скриптов или стилей), - все это, пользователь не может увидеть на странице. Однако без этого невозможно добиться нормального функционирования вашего сайта. Если убрать тег <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />, то большинство браузеров не сможет отобразить русский текст в читаемом виде. Придется либо менять настройки браузера, либо отказаться от использования русского текста. Все эти варианты доставляют неудобство как разработчику так и посетителям сайта. Поэтому существует возможность явно указывать браузерам, в какой именно кодировке нужно отображать текст на странице.

Пожалуй, единственный элемент, который можно «увидеть» на странице, относящийся к тегу <head> - это заголовок страницы, который определяется внутри контейнерного тега <title>. Заголовок страницы отображается на вкладке браузера и используется для создания аннотаций поисковыми роботами.



* 1. Задания

Ответьте на вопросы:

* Что такое HTML?
* Что такое тег?
* Какие вы знаете обязательные теги для любого html-документа?
* С помощью чего можно создавать html-документы?
* С помощью чего можно просматривать html-документы?
* Какие html-теги вам известны?
* Индивидуальное задание

1. ***Первые шаги***
   1. Создание абзацев

Внимательный читатель уже научился создавать абзацы, освоив предыдущий материал, но повторение не бывает лишним.

Для определения абзаца внутри документа служит контейнерный тег <p>. Создадим с его помощью три абзаца:

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Учебная страница</title>

</head>

<body>

<p>Мы только начали изучать HTML, поэтому мне нечем вас удивить.</p>

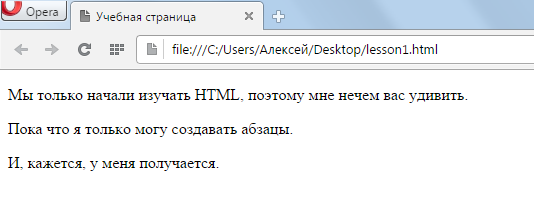
<p>Пока что я только могу создавать абзацы.</p>

<p>И, кажется, у меня получается.</p>

</body>

</html>

Такой незамысловатый код выполняет поставленную задачу.



* 1. Создание списков

Списки используются в основном для перечисления чего-либо. Бывают нумерованные и маркированные списки.

Пример нумерованного списка:

1. Первый элемент списка
2. Второй элемент списка

Пример маркированного списка

* Элемент списка
* Элемент списка

Для создания нумерованного списка используется контейнерный тег <ol>, а для создания маркированного списка - <ul>. Оба списка состоят из «элементов списка», которые указываются внутри списка с помощью контейнерного тега <li> (от англ. «List Item»).

Создадим нумерованный и маркированный список с помощью HTML:

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Учебная страница</title>

</head>

<body>

<ol>

<li>Первый элемент списка</li>

<li>Текст второго элемента</li>

</ol>

<ul type="square">

<li>элемент списка</li>

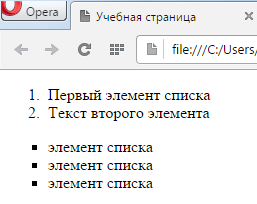
<li>элемент списка</li>

<li>элемент списка</li>

</ul>

</body>

</html>



Атрибут «type» отвечает за внешний вид маркера (для маркированного списка). Может принимать значения: circle, disc(по умолчанию) и square.

* 1. Заголовки

Для выделения отдельных смысловых участков на вашем сайте следует использовать заголовки. Заголовки – это более крупный текст по сравнению с обычным текстом. Такой текст позволяет пользователю быстрее ориентироваться в контенте вашего сайта.

В HTML можно создавать заголовки 6 видов: <h1>, <h2>, … , <h6>. Причем заголовок с меньшим номером крупнее, чем заголовок с большим номером.

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Учебная страница</title>

</head>

<body>

<h1>Заголовок H1</h1>

<h2>Заголовок H2</h2>

<h3>Заголовок H3</h3>

<h4>Заголовок H4</h4>

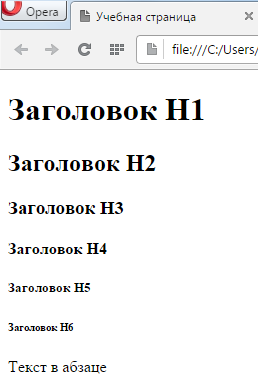
<h5>Заголовок H5</h5>

<h6>Заголовок H6</h6>

<p>Текст в абзаце</p>

</body>

</html>



* 1. Изменение цвета фона и свойств используемого шрифта

Для изменения цвета отдельных элементов на странице, как правило, используют специальные таблицы стилей - CSS. Мы будем их рассматривать позднее, поэтому сейчас воспользуемся теми тегами, которые позволяют редактировать цвет фона и изменять шрифт.

Цвет фона всей странице можно задать с помощью атрибута «bgcolor» тега <body>. Цвет можно указывать как в словесной форме, так и, например, в hex-коде (формат #RRGGBB). Например, bgcolor=”red” или bgcolor=”#FF0000”.

Изменять свойства шрифта можно, поместив необходимый текст, внутрь контейнерного тега <font>. С его помощью можно изменять цвет, размер и гарнитуру шрифта.

<font size=”5” color=”green” face=”Arial”>Зеленый текст</font>

Преобразовать обычный текст в курсивный можно, поместив его внутрь контейнерного тега <i>, если же необходимо получить полужирный шрифт, то следует использовать контейнерный тег <b>. Есть и другие специальные теги, с которыми вы можете ознакомиться в документации [2].

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Учебная страница</title>

</head>

<body bgColor="gray">

<font color="white" size="5" face="Arial">

<p>Текст в абзаце</p>

<i>Курсивный текст</i><br />

<b>Жирный текст</b><hr />

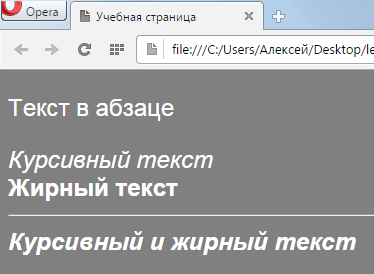
<b><i>Курсивный и жирный текст</i></b>

</font>

</body>

</html>

В данном примере для переноса текста на новую строку были использованы специальные теги <br /> и <hr />. <HR /> - помимо перехода на следующую строку рисует разделительную черту. Без использования этих тегов почти весь текст (кроме абзаца) будет на одной строке. Мы уже оговаривали ранее этот момент.



*Примечание: обратите внимание, что вложенные теги должны закрываться в том же порядке, что и открывались <b><i></i></b>, не должно быть перекрестных тегов. Нельзя записывать следующим образом <b><i></b></i>.*

* 1. Комментарии и ошибки

Комментирование кода позволяет легко вникнуть в суть происходящего не только сторонним людям, но и вам самим, если вы захотите вернуться к коду через месяцы, а методички не будет под рукой. Комментирование не только повышает читабельность вашего кода, но также и предотвращает возможные ошибки – вы еще раз взгляните на свой код, прежде чем начнете комментировать его. Если в вашем коде есть ошибка, то браузер никак не сообщит вам о том, на какой строчке она возникла, вы лишь увидите, что страница загрузилась не до конца и на каком-то моменте перестала прорисовываться дальше. В первую очередь исследуйте именно этот участок кода. Можно быстро получить код, который смог распознать браузер, используя опции самого браузера – получить код исходной страницы.

За добавление комментариев отвечает следующая конструкция:

<!-- Текст моего первого комментария -->

Мы вам рекомендуем пользоваться этой конструкцией часто, но все-таки совсем необязательно комментировать каждую строчку кода.

* 1. Вставка изображения на страницу

Изображение способно освежить любой сайт. Настала пора научиться добавлять картинку на наш сайт. Делается это с помощью одиночного тега <img>:

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Учебная страница</title>

</head>

<body >

<p>

<img src="D:/header\_html.png" alt="Не прогрузилась :(" width="160" height="57" align="left">

<h3> Занятие 2 </h3>

В этом занятии мы научимся добавлять картинку,

</p>

<p align="center">а заодно</p>

<p align="right">изучим горизонтальное выравнивание</p>

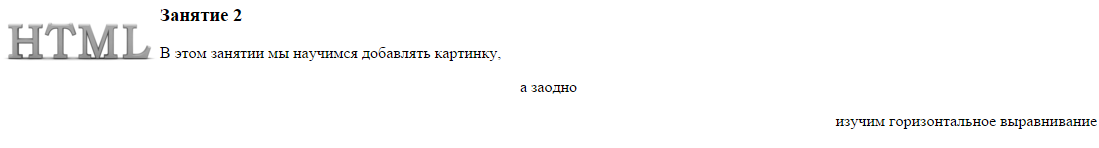
</body>

</html>

Не стоит пугаться такого обилия атрибутов у тега <img> - обязательным является только атрибут «src», а остальные можно опустить. Однако чем больше атрибутов мы знаем, тем ярче и функциональнее может стать наш сайт.

* src – файл с изображением
* alt – текст, который отобразиться, если картинка не прогрузится
* width и height – ширина и высота. С помощью этих атрибутов можно смасштабировать изображение до нужных размеров.
* align – горизонтальное выравнивание изображение. Может принимать значения: left, center, right. Отметим так же, что подобный атрибут есть и у тега <p>.

Обратите внимание на вложенность тега <img> в этом примере. Объясните себе почему сделано именно так.



* 1. Задания
* Создайте абзац и выделите в нем первую букву крупным красным шрифтом
* Создайте три абзаца, перед каждым из них добавьте заголовок. Последний абзац выровняйте по правому краю
* Создайте два абзаца, в первый добавьте картинку слева от текста абзаца, а во второй абзац добавьте картинку справа от текста
* Создайте маркированный список из изображений на сером фоне

1. ***Ссылки в HTML***
   1. Виды адресов, на которые можно ссылаться

Когда современное общество активно использует интернет-технологии, сложно отыскать человека, который не знаком со словом «ссылка». Такие поисковики, как: <https://www.google.com>, <https://www.yandex.ru>, - у всех на слуху. Однако в HTML ссылка может не только представлять из себя какой-то адрес в интернете (кликнув по такой ссылке, пользователь попадет на ту страницу, которая была указана в ссылке), а также может указывать на e-mail или даже на конкретное место в html‑документе. Правильное использование ссылок может позволить не только создавать переходы между страницами, но также позволит создать оглавление с переходами внутри страницы.

Ссылка на web-страницу:

<https://www.somepage.com>

Ссылка на e-mail (отправка электронного письма на указанный адрес):

<mailto:some@mail.com>

Ссылка на «именованный абзац» (якорь) другой страницы:

<https://www.somepage.com#paragraph_5>

Ссылка на «именованный абзац» (якорь) внутри текущей страницы:

#paragraph\_5

* 1. Простая ссылка

За добавление ссылки в HTML отвечает контейнерный тег <a> с обязательным атрибутом «href» или «name» в зависимости от назначения ссылки. Для создания простой ссылки на какую-либо web-страницу используется атрибут «href», с помощью которого задается адрес в интернете, соответствующий переходу по ссылке.

<a href=”https://www.google.com”>Search Google</a>

Между открывающим и закрывающим тегом задается видимый текст для ссылки, т.е. тот текст, который увидит пользователь. Кликнув по такой ссылке, пользователь будет переадресован на сайт Goolge.

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Учебная страница</title>

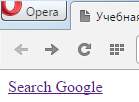
</head>

<body >

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

</body>

</html>



* 1. Якорь (Anchor)

Всем известно, что если лодку сносит течением, то надо бросить якорь. Но с этим якорем у ссылок мало общего. В IT-сфере термин «якорь» используется для обозначения привязки к чему-либо. Это может быть левая граница экрана или конкретное место в документе. Якоря в html-документах, как правило, используются для быстрого перехода к определенной части контента web‑страницы. С их помощью можно создать удобную навигацию по странице и добавить оглавление с переходами по указанным позициям.

Добавить на страницу якорь можно несколькими способами. Мы рассмотрим лишь один из них, который полностью опирается на использование тегов <a>. Для этого нам понадобиться применить тег <a> с различными атрибутами.

Одна из ссылок будет ссылаться на якорь (место в документе). Для этого будет использоваться атрибут «href», причем в адресе ссылки мы будем использовать имя того якоря, к которому должен перейти пользователь. Это достигается за счет добавления к адресу ссылки конструкции «#ИмяЯкоря».

Вторая ссылка будет непосредственно якорем. Чтобы задать имя якоря, воспользуемся специальным атрибутом «name».

<body>

<a href="#MyAnchor">Перейти к ...</a>

<!-- Добавьте в этом месте столько текста, чтобы он не помещался на страницу. Чтобы текст, который указан ниже пришлось «проскролить» -->

<a name="MyAnchor"></a>

<!-- За якорем добавьте текст, чтобы полностью наблюдать переход по ссылке к якорю -->

</body>

На месте комментариев вам следует добавить какую-нибудь объемную статью, которая бы скрывала якорь за границами экрана. Если вы все сделали правильно, то после перехода по ссылке страница пролистывается до якоря. Причем якорь оказывается в самом верху видимой области. Для более комфортного перехода желательно центрировать видимую область по якорю и добавлять эффект «пролистывания» страницы, но для учебных целей нам это не нужно.

* 1. Использование тегов внутри ссылки

Для добавления ссылки используется контейнерный тег <a>. Как и внутри любого контейнерного тега, внутри тега <a> может указываться не только текст, но и другие теги. Так, например, легко превратить любую картинку в ссылку.

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

<img src="D:/google.png" alt="Нажмите для перехода">

</a>

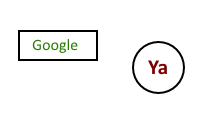
* 1. Карта изображения

Для красочных сайтов, в которых используется некий стиль, может возникать необходимость задания ссылок в качестве каких-то геометрических фигур. Например, ссылка в виде треугольника или сразу две разные ссылки на одном изображении. Для решения такой задачи к изображению необходимо добавить специальную карту с переходами. На такой карте попиксельно задаются участки изображения, которые являются ссылками. По умолчанию поддерживаются ссылки в виде прямоугольников, кругов и многоугольников.

Карта задается с помощью контейнерного тега <map> с обязательным атрибутом «name» (произвольное название этой карты). Карта состоит из областей, каждая из которых является ссылкой определенной формы и размера. Области задаются с помощью контейнерного тега <area>. Обязательными атрибутами этого тега являются:

* shape – форма области: rectangle, circle или poly (прямоугольник, круг или многоугольник).
* coords – координаты, в которых располагается указанная геометрическая форма на изображении. Для каждого значения shape форма записи этого атрибута своя. Для прямоугольника указываются координаты (x,y) сначала верхнего левого угла (в пикселях), а затем координаты нижнего правого угла (coords=”x1,y1,x2,y2”). Для круга указывается координата (x, y) центра окружности, а затем величина радиуса окружности в пикселях (coords=”x,y,r”). Для многоугольника последовательно перечисляются все координаты (x,y), из которых он состоит, причем последняя указанная пара координат автоматически будет замкнута на первую пару координат (coords=”x1,y1,x2,y2,x3,y3, . . . , xN, yN).
* href – адрес, на который будет переходить пользователь по клику в этой области.

Для демонстрации мы собрали все свои навыки рисования и получили на выходе следующее изображение



По нажатию на прямоугольник пользователь будет переходить на сайт Google, а по нажатию на круг будет происходить переход на сайт поисковика Yandex. Это единое изображение, а не два различных. Для того, чтобы применить карту к изображению, необходимо задать атрибут «usemap» для тега <img>. В атрибуте «usemap», подобно ссылке на якорь, указывается сслыка на карту с областями (usemap=”#ИмяКарты”).

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Учебная страница</title>

</head>

<body >

<img src="D:/navigation.png" alt="Карта переходов" usemap="#Navigation">

<map name="Navigation">

<area shape="rectangle" coords="16,30,93,58" href="https://www.google.com">

<area shape="circle" coords="154,66,25" href="https://www.yandex.ru">

</map>

</body>

</html>

* 1. Задания
* Создайте нумерованный список из ссылок
* Создайте небольшое оглавление с помощью ссылок и быстрым переходом по ним на соответствующие этим главам якоря. Посмотрите на изменения в адресной строке браузера. На основе полученных знаний создайте ссылку на якорь, который расположен в другом созданном вами html-документе.
* Поэкспериментируйте с вложенными внутрь тега <a> тегами. Что вы можете превратить на своей странице в ссылку?
* Нарисуйте треугольник, квадрат и круг. Сохраните изображение. Добавьте изображение на вашу страницу. Добавьте к изображению карту переходов. Используйте все типы областей (прямоугольник, круг и многоугольник). Вы можете использовать и другие фигуры, если захотите.

1. ***Таблицы в HTML***

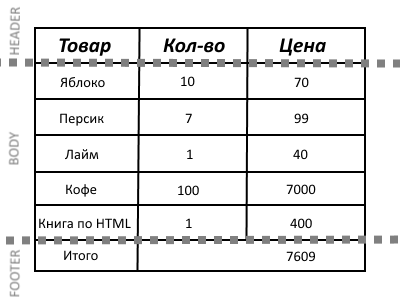
Если вы дочитали до этой главы, то уже представляете себе основные синтаксические конструкции языка html, знаете какие теги обязательные, без труда сможете добавить картинку или ссылку на сайт. Поэтому с этого момента мы будем записывать теги без явного указания родительских тегов(<body> или <head>), вы уже должны понимать, где следует прописывать такие теги.

* 1. Структура таблицы

Таблица в html состоит из трех частей:

* Заголовок или шапка (HEADER)
* Тело таблицы (BODY)
* Подвал (FOOTER)

Любая из этих частей состоит из строк, которые в свою очередь состоят из ячеек.



В приведенном примере шапка таблицы состоит из одной строки, тело таблицы (основная часть с данными) состоит из 5 строк, а подвал состоит из одной сроки. В зависимости от целей, шапка и подвал могут опускаться. В каждой из строк определенно по три столбца, в подвале два последних столбца объединены в один (именно так, количество столбцов в каждой из строк не должно изменяться, их можно только объединять между собой, а не удалять или добавлять). Объединять можно не только столбцы, но также и строки. Все это мы скоро рассмотрим. А сейчас перейдем к основам.

Для добавления таблицы на вашу страницу необходимо использовать контейнерный тег <table>. По умолчанию таблицы в html прорисовываются без границ, чтобы увидеть контуры добавленной таблицы следует использовать атрибут «border» тега <table>. В качестве значения этот атрибут принимает целое число, которое определяет толщину прорисовываемой границы (по умолчанию border=0). Как упоминалось ранее, таблица состоит из строк и ячеек. Строки таблицы задаются контейнерным тегом <tr> (от англ. «Table Row»), а внутри строк задаются ячейки таблицы с помощью контейнерного тега <td> (от англ. «Table Data»). Причем учитывайте то, что количество ячеек в каждой строке, должно быть одинаковым (так как, по сути, это описание значений в столбцах таблицы для строки, а столбцов в таблице константное число). Конечно, существуют расширенные опции (атрибуты) для создания таблиц, с помощью которых можно объединять несколько ячеек внутри одной строки или, аналогично, можно объединять несколько ячеек внутри одного столбца (в разных строках). Для этого в теге <td> указываются атрибуты «colspan» или «rowspan» соответственно. В качестве значений для этих атрибутов указывается количество ячеек (<td>), которые следует объединить. Помните, что количество ячеек в каждой строке должно быть неизменно, поэтому если для таблицы вы определили 4 столбца, то внутри каждого <tr> должно быть ровно 4 <td>. Если же вы укажите colspan=3, то есть для какой-то строки у вас будет ячейка, занимающая 3 столбца в строке, то помимо такой «широкой» ячейки должна быть определена еще только 1 ячейка (т.е. всего два <td>), так как 3+1=4, что нам и требуется. Внимательно изучите приведенный пример.

<table border=1>

<tr>

<th> Header 1 </th>

<th> Header 2 </th>

</tr>

<tr>

<td> Data 11 </td>

<td> Data 1 2 </td>

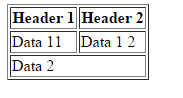
</tr>

<tr>

<td colspan=2> Data 2 </td>

</tr>

</table>



Обратите внимание, что в первой строке для определения ячеек мы использовали контейнерный тег <th>, а не <td>. Этот тег специализирован под заголовок таблицы. Текст, помещенный внутрь такой ячейки, выделяется жирным шрифтом. Хотя следует понимать, что заголовок можно задавать и обычными <td> тегами, при желании добавив к ним тег <b>.

Теперь мы с вами умеем создавать html-таблицу. Давайте рассмотрим более детальный вариант ее определения с указанием секций таблицы: заголовок, тело и подвал (thead, tbody и tfoot). Комментариев добавлять к коду не будем, так как вы уже должны уметь разбираться в том, что добавилось, и за что отвечают новые теги. Отметим только, что такая запись используется в основном в тех случаях, когда применяются специфические визуальные стили для таблиц, чтобы единым образом представлять шапки таблицы и их основное содержание. Поскольку мы с вами еще не знакомы со стилями, то такая запись в ближайшее время нам не пригодится, но это не значит, что мы можем позволить себе не знать о таком способе создания таблицы «по частям».

<table border=1>

<thead>

<tr>

<th> Header 1 </th>

<th> Header 2 </th>

</tr>

</thead>

<tbody>

<tr>

<td> Data 11 </td>

<td> Data 12 </td>

</tr>

<tr>

<td colspan=2> Data 2 </td>

</tr>

</tbody>

<tfoot>

<tr>

<td> Footer 11 </td>

<td rowspan=2> Footer \*2 </td>

</tr>

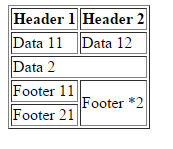
<tr>

<td> Footer 21 </td>

</tr>

</tfoot>

</table>

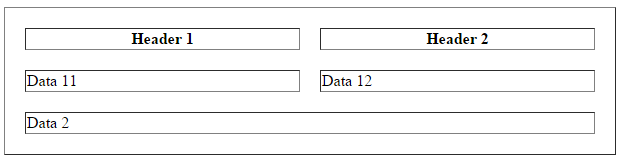


* 1. Форматирование таблицы

В данном разделе мы рассмотрим только дополнительные атрибуты тега <table>. Информация по тегам <tr> и <td> нам в ближайшее время не понадобится, однако мы рекомендуем вам ознакомиться с дополнительными атрибутами и для этих тегов на страницах официальной документации[2].

Опциональные атрибуты тега <table>:

* cellspacing. Используется для определения расстояния между ячейками таблицы. Например, для cellspacing=20 заметно, что ячейки не примыкают вплотную друг к другу, а разделяются как по вертикали, так и по горизонтали на расстояние в 20 пикселей.



* align. Используется для определения того к какой границе экрана должна примыкать таблица (left, center, right). Отметим, что для строк таблицы (тег <tr>) можно указывать не только горизонтальное выравнивание, но и вертикальное выравнивание для того содержимого, которое будет находиться внутри ячеек таблицы (halign и valign).
* width. По умолчанию таблицы растягивается на минимально возможную величину, которой достаточно, чтобы отобразить содержимое таблицы. Но что делать, если нам необходимо растянуть таблицу на весь экран или на определенное количество пикселей? Явное указание ширины таблицы спасет нас при наличии такой проблемы. Ширину таблицы можно указывать как в пикселях (<table width=”500”>), так и в процентном соотношении по отношению к ширине рабочей области браузера (<table width=”100%”). Данная запись может поддерживаться не всеми браузерами, скоро мы научимся добиваться того же эффекта при помощи стилей.
* height. Используется для определения высоты таблицы в пикселях.
* border. Используется для задания ширины прорисовываемой границы в пикселях для элементов таблицы.
  1. Блок Div

Одно из применений таблиц, кроме очевидного применения, это компоновка страницы. С помощью таблицы можно разделить всю страницу на части, в одной из ячеек будет меню сайта, в другой основной контент или рекламные баннеры. Однако в последнее время прослеживается тенденция использования для таких целей контейнерных тегов <div>. Блок <div> без дополнительных стилей и атрибутов отделяет контент построчно, но, применяя к нему такие стили как обтекание или выравнивание, можно добиться подобного разделения страницы на ячейки, как и при использовании таблицы. С применением тегов <div> проще управлять стилями, также их использование позволяет избежать лишних нагромождений из <table><tr><td> вложенностей. Еще у таблиц есть одна небольшая особенность: по умолчанию ширина столбцов в таблице одинакова для каждого столбца, в зависимости от контента эта величина может измениться, поэкспериментируйте с содержимым таблицы. Отметим только, что не всегда значение ширины ячейки выбирается желательным образом. Кроме того, в случае с таблицей, нам приходится держать в голове информацию о том, сколько ячеек должно быть в каждой строке, сколько строк для столбца мы объединили между собой и т.д., применение блока <div> позволит избежать этих проблем. На данном этапе мы несколько ограничены в вариантах использования блока <div>, поскольку еще не знакомы с применением стилей обтекания. Поэтому пока что блок div фактически ничем не отличается для нас от контейнерного тега <p> (атрибут «align» у него тоже есть).

* 1. Задания
* Приведите пример использования контейнерного тега <div>
* Добавьте на страницу таблицу умножения
* Добавьте на страницу таблицу со списком купленных товаров (вымышленным покупателем А). В таблице должна быть шапка (наименование товара, кол-во), основной контент таблицы (перечисление купленных позиций) и подвал (строка «Итого»)
* В этом задании от вас требуется скомпоновать свою страницу.
  + Выделить в ней шапку, подвал, меню сайта и основной контент.
  + Наполнить страницу содержимым (в шапке можно указать название сайта и добавить логотип, меню сайта может содержать список из ссылок, в подвале можно указать автора сайта, а контент сайта представьте так, как считаете нужным)

Используйте контейнерные теги <table> или <div>.

1. ***Фреймы***
   1. Что такое фрейм

Фреймы разделяют окно браузера на отдельные области, расположенные вплотную друг к другу. В каждую из таких областей загружается самостоятельная веб-страница определяемая с помощью тега **<**frame**>**. С помощью фреймов веб-страница делится на два или более документа, которые обычно содержат навигацию по сайту и его контент. Механизм фреймов позволяет вам использовать макеты для отдельных частей страницы. Предположим, ваш сайт состоит из семи html-страниц. Как любой хороший сайт эти страницы должны быть выдержаны в едином стиле. На каждой из этих страниц есть меню. Не будете же вы копировать один и тот же html-код по компоновке меню сайта на каждую страницу? Вы можете добавить восьмой файл, в котором определите только меню, а в остальных семи файлах будете подключать этот файл парой строк, не формируя каждый раз меню заново. Такой механизм используется редко, поскольку проще решить эту задачу с помощью конструкций языка php, но для учебных целей мы попробуем воссоздать и такой вариант. Чаще фреймы используются, чтобы на одном сайте прогрузить часть контента с другого сайта (чтобы пользователь мог использовать сторонние сервисы или увидел, что его ожидает при переходе по ссылке).

* 1. FrameSet

Тег <frameset> заменяет собой элемент <body> на веб-странице. Допустимо использовать вложенную структуру элементов, это позволяет разбить один фрейм на две и более области [2]. <frameset> состоит из фреймов <frame> или других <frameset>.

<frameset>

<frame>

</frameset>

Рассмотрим некоторые атрибуты тега <frameset>

* [border](http://htmlbook.ru/html/frameset/border). Задает толщину границы между фреймами.
* [bordercolor](http://htmlbook.ru/html/frameset/bordercolor). Задает цвет линии границы.
* [cols](http://htmlbook.ru/html/frameset/cols). Устанавливает ширину или пропорции фреймов в виде колонок.
* [rows](http://htmlbook.ru/html/frameset/rows). Задает размер (высоту) или пропорции фреймов в виде строк.

Особое внимание здесь следует уделить атрибутам «cols» и «rows», так как именно эти атрибуты определяют как будут располагаться фреймы на странице. Для этих атрибутов параметры задаются через запятую. Например, <frameset rows=”80,\*” cols=”\*”> означает, что рабочая область будет разделена на две строки и одну колонку. Символ «\*» означает, что ширина или высота будет эквивалентна оставшейся (не занятой другими фреймами) области на странице. Для данного примера первая строка занимает 80 пикселей в высоту и всю ширину экрана браузера, а вторая строка занимает оставшуюся область по высоте (все высота минус 80 пикселей) и также всю ширину экрана. Указывать ширину или высоту можно как в пикселях, так и в процентах (из расчета 100% на весь frameset), кроме того, можно воспользоваться специальным символом «\*».

Но сами по себе <frameset> ничего кроме разметки страницы не задают, чтобы в конкретной области появился фрейм, его необходимо задать с помощью одиночного тега <frame>.

Тег <frame> определяет свойства отдельного фрейма, на которые делится окно браузера. Этот элемент должен располагаться в контейнере <frameset> [2]. Рассмотрим некоторые атрибуты тега <frame>.

* [bordercolor](http://htmlbook.ru/html/frame/bordercolor). Определяет цвет линии границы вокруг фрейма.
* [frameborder](http://htmlbook.ru/html/frame/frameborder). Отображать рамку вокруг фрейма или нет. Может принимать значения 0 или 1.
* [name](http://htmlbook.ru/html/frame/name). Задает уникальное имя фрейма. Рекомендуем присваивать уникальные имена всем вашим фреймам.
* [noresize](http://htmlbook.ru/html/frame/noresize). Определяет, можно изменять размер фрейма пользователю или нет. Для нас этот атрибут примечателен тем, что он не требует указания значения (знак равно ставить не нужно). Например, <frame src=”test.html” name=”test” noresize>.
* [scrolling](http://htmlbook.ru/html/frame/scrolling). Определяет способ отображения полосы прокрутки во фрейме. Может принимать значения «no», «yes» или «auto».
* [src](http://htmlbook.ru/html/frame/src). Задает путь к файлу, предназначенному для загрузки во фрейме.

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Учебная страница</title>

</head>

<frameset cols="50%,50%" rows="\*">

<frame src="test.html" name="testFrame1">

<frameset cols="\*" rows="100,\*">

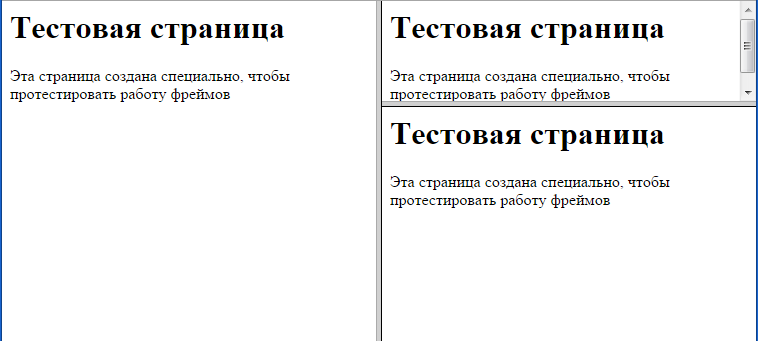
<frame src="test.html" name="testFrame2">

<frame src="test.html" name="testFrame3">

<frameset>

</frameset>

</html>



В данном примере мы загрузили во все фреймы одну и ту же страницу, разумеется, вы можете использовать и различные страницы для фреймов.

Обратите внимание на то, как скомпоновались фреймы. Разберемся в приведенном примере. Внутри <frameset> определен дочерний <frameset> на ровне с обычным фреймом. Первым тегом мы указали браузеру, что хотим разделить рабочую область браузера на две колонки, причем одинаковые по размеру. В первую колонку мы поместили фрейм, а вторую продолжили разделять. В итоге во второй колонке мы определили два фрейма в двух строках. Высота первой строки 100 пикселей, а второй все оставшееся пространство.

В большинстве браузеров по умолчанию для фреймов используется плавающая граница, то есть за рамку вокруг фрейма можно «потянуть» мышкой, изменяя размер фреймов.

Кодировка загружаемых во фреймах страниц своя, на нее не распространяется значение, указанное в <head> страницы-контейнера фреймов.

* 1. Плавающий фрейм

Механизм плавающих фреймов позволяет внедрять фреймы на ваш сайт, и при этом вы не ограничены лишь в использовании тегов <frameset> и <frame>. Плавающие фреймы размещаются внутри тега <body> и задаются с помощью контейнерного тега <iframe>, который позволяет загружать в область заданных размеров любые другие независимые документы. Некоторые версии браузеров могут не поддерживать тег <iframe>, и его содержимое будет проигнорировано. Для таких браузеров можно указать альтернативный текст, который увидят пользователи вместо плавающего фрейма. Этот текст необходимо размещать между открывающим и закрывающим тегом.

Перейдем к рассмотрению некоторых атрибутов тега <iframe>:

* [align](http://htmlbook.ru/html/iframe/align). Определяет как фрейм будет выравниваться по краю, а также способ обтекания его текстом.
* [frameborder](http://htmlbook.ru/html/iframe/frameborder). Устанавливает, отображать границу вокруг фрейма или нет.
* [height](http://htmlbook.ru/html/iframe/height). Задает высоту фрейма в пикселях.
* [name](http://htmlbook.ru/html/iframe/name). Определяет уникальное имя фрейма.
* [scrolling](http://htmlbook.ru/html/iframe/scrolling). Задает способ отображения полосы прокрутки во фрейме.
* [src](http://htmlbook.ru/html/iframe/src). Задает путь до файла, содержимое которого будет загружаться во фрейм.
* [width](http://htmlbook.ru/html/iframe/width). Определяет ширину фрейма в пикселях.

Полный перечень атрибутов вы всегда сможете найти в online-документации по языку HTML [2].

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Учебная страница</title>

</head>

<body>

<h1>Добавляем на форму плавающий фрейм</h1>

<p>Воспользуемся специальным тегом <b>&lt; iframe &gt; </b></p>

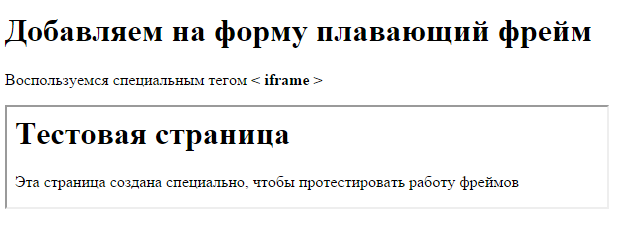
<iframe src="test.html" width="600" height="100" frameborder="1">

Ваш браузер не поддерживает плавающие фреймы. Нам очень жаль.

</iframe>

</body>

</html>



В приведенном примере для отображения плавающего фрейма, мы позволили себе воспользоваться специальным приемом, который вам еще не знаком. Несомненно, многие из вас задумывались о том, а как же вывести знак больше или меньше («>» или «<») как текст внутри страницы, если эти символы зарезервированы под определение тегов в языке html? Ответ очень прост: напрямую никак. Эти символы действительно зарезервированы только под определение тегов. Чтобы отобразить эти символы на странице необходимо «экранировать» их. То есть вводить код символа или специальную последовательность символов, которая преобразуется браузерами в нужный вам символ. Приведем пример наиболее распространенных последовательностей:

* **&lt;**

Отображает на странице знак меньше.

* **&gt;**

Отображает на странице знак больше.

* **&nbsp;**

Отображает на странице пробельный символ. Мы уже оговаривались, что браузеры все пробельные символы между элементами сокращают до одного пробела, но что делать, если вам все-таки надо поставить два пробельных символа подряд? В этом случае вы можете «экранировать» пробелы, явно указывая браузеру, что их не следует сокращать.

*Примечание: обратите внимание на запись: сначала ставится символ амперсант, затем идет последовательность букв, которая отвечает за свой символ и в конце последовательности ставится точка с запятой.*

* 1. Задания
* Составьте с помощью <frameset> таблицу 3х3 из фреймов (можете использовать один и тот же html-документ)
* Что такое фрейм?
* Как вы считаете, хорошей ли практикой является использование фреймов на своей странице? В каких случаях стоит использовать фреймы?
* В чем отличие в использовании тега <frame> от <iframe>?
* \*Вопрос, не относящийся к теме фреймов: расскажите об экранировании символов. Зачем оно может пригодиться, и какие способы экранирования вам известны?

1. ***CSS***
   1. Что такое CSS

CSS (от англ. «Cascading Style Sheets») – язык описания внешнего вида документа, написанного с использованием языка разметки. Дословно переводится как «каскадные таблицы стилей». В последнее время css набирает обороты не только в web-разработке, а также активно внедряется в дизайнеры форм популярных IDE (сред сразработки).

Хотим мы изменить размер какого-то элемента на странице, поменять цвет, сменить шрифт или обвести границей – все это легко делается с помощью применения стилей. Примечательно для учебных целей то, что многие html-теги поддерживают одни и те же свойства стилевого оформления. Допустим, вы узнали, как задать свой шрифт для блока div, с большой вероятностью с помощью эквивалентного параметра можно будет задать свой шрифт и другим тегам. Универсальность в изменении параметров тегов с помощью стилей, безусловно, приятная новость для нас. Конечно, из этого правила есть исключения.

Перейдем к конкретным примерам, как же выглядит синтаксис css.

p

{

background-color: green;

color: yellow;

}

Во-первых, стиль, как правило, применяется к каким-то элементам html-документа. В данном случае мы воспользовались именем тега – стиль будет определен для тега <p>. Видите, где мы указали тег? В самой первой строке «p», т.е. пишем теги без треугольных скобочек. Затем, те свойства стиля, которые необходимо переопределить для тега, указываются через точку с запятой внутри блока, ограниченного фигурными скобками (после названия тега ставится открывающая фигурная скобка, после переопределения свойств стиля тега ставится закрывающая фигурная скобка). Синтаксис привычен для тех людей, которые знакомы с Java, C, C++ или JavaSript.

Во-вторых, для тех элементов, которые мы хотим изменить, нужно указать какие именно свойства оформления следует изменить. В приведенном примере мы сменяем заливку всех тегов <p> на странице на зеленый цвет (background-color: green;), а цвет текста внутри этих тегов будет желтый (color: yellow;).

Подытоживая, приведем общий вид записи для задания своего стиля каким-то элементам в html-документе.

Селектор

{

свойство : значение;

свойство : значение;

}

Свойства стиля, в случае если они состоят из нескольких слов, записываются через дефис, как например, background-color. Для записи свойств используются только строчные буквы латинского алфавита и дефис. Свойство от его значения отделяется знаком двоеточие. После значения ставится знак «точка с запятой». Несколько позже мы пополним свой «словарь параметров стиля», а пока будем рассматривать примеры, в которых используются только изменения цвета текста или цвета фона.

На месте селектора может быть не только наименование тега. Возможные варианты мы вскоре рассмотрим.

* 1. Разные стили для разных элементов

Первое на что стоит обратить внимание, так это то, где прописывать эти стили, и как увидеть результат на моей страничке? Стили будем записывать в обычном файле, созданным любым из текстовых редакторов. Расширение у такого файла следует задать как «.css» (по такому расширению вы сразу поймете, какого содержания этот файл, более того, поймет это и редактор с подсветкой синтаксиса). На вопрос как применить ваши стили, записанные в файле «mySuperStyles.css», мы ответим чуть позже. А сейчас окунемся в синтаксис css с головой.

Стили можно задавать сразу для нескольких тегов, для этого нужно записывать известную нам конструкцию друг за дружкой.

body {

color: black;

}

p {

background-color: green;

}

В примере определен цвет текста для всех элементов, находящихся внутри тега <body>, и цвет заливки для всех тегов <p> на странице. Напрашивается справедливый вопрос, а что если я захочу сделать цвет у тега <body> одним, а у тега <p> другим? Они как-то перезаписывают друг друга? Ответ: да, перезаписывают. Текст внутри всего документа вы можете сделать зеленым (для тега <body>) , а внутри абзацев – красным (для тега <p>).

Теперь рассмотрим еще более частный случай. Допустим, вы не хотите, чтобы все теги <p> у вас были одного стиля, а вам хочется уметь задавать различный стиль для одинаковых тегов в зависимости от их вложенности в другие теги. Зададим один стиль для всех тегов <p> и другой стиль для более частной группы: тегов <p>, вложенных внутрь тега <div>. Более частный стиль перезапишет более общий, и мы получим требуемое стилевое оформление. За то, что один тег обязательно должен быть вложен в другой тег, отвечает специальный символ «>» (знак больше) между тегами в записи селектора. Данная связка для селектора называется селектором дочерних элементов.

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Учебная страница</title>

<style>

p {

color: green;

}

div {

color: red;

}

div > p {

color: blue;

}

</style>

</head>

<body>

<p> 1

<div> 2

<p> 3 </p>

</div>

</p>

</body>

</html>

Из приведенного примера видно, что стиль можно задавать в секции <head> html‑документа с помощью контейнерного тега <style>. Запись «div > p» означает, что мы определяем стиль только для таких тегов <p>, которые вложены в тег <div>. Результат обработки этого кода браузера будет следующим:

* единица – зеленая
* двойка – красная
* тройка – синяя

*Примечание: мы не уверены в том, что этот материал дойдет до вас в цветном оформлении, поэтому картинка с результатом не прикладывается.*

Обратите внимание, что полный путь до тега писать вовсе не обязательно, можно взять только какое-то окончание этого пути, которое однозначно определяет набор необходимых тегов.

«body > table > tr > td > a» и «td > a» - это эквивалентные записи селекторов для тех страниц, в которых нет таблиц, вложенных в другие таблицы (в этом случае вторая запись может затронуть и ссылки во вложенной таблице). Если вложенных таблиц на вашей странице нет, то нагляднее использовать все-таки второй вариант.

Что делать, если вы не хотите полностью помнить уровни вложенности тегов. Допустим, вы знаете, что где-то в блоке div есть тег <p>, но он вложен различными способами:

* <div><p>
* <div><span><p>
* <div><a><p>
* <div><table><tr><td><p>

В случае с селектором дочерних элементов ваши селекторы для стиля выглядели бы так:

* div > p
* div > span > p
* div > a > p
* div > table > tr > td > p

Не много ли одинаковых стилей придется прописать для одного единственного вида тегов? Программисты и разработчики – люди рациональные и, естественно, он придумали способ задавать селектор, который указывает на отношение родитель-потомок, но это связь не обязана быть прямой, а может включать в себя сколько угодно промежуточных звеньев. Такой вид селектора называет селектором потомков. При записи этого селектора опускается символ «>», т.е. теги записываются через пробел. Как итог, получаем всего один селектор:

* div p

Таким селектором мы задаем стиль для тегов <p> произвольного уровня вложенности в тег <div>. Стало лучше, не правда ли?

Последний вид селектора, который мы затронем в этой главе – это селектор атрибутов. Опять же, рассмотрим использование на примере какой-нибудь ситуации. Допустим, на вашем сайте, есть пять ссылок, переход по одной из этих ссылок является обязательным для пользователей. Естественно, вы захотите как-то выделить такую ссылку, например, с помощью придания ей особенного стилевого оформления. Определить конкретную ссылку, можно по значениям ее атрибутов, например, по значению атрибута «href». Запишем селектор атрибутов, который определит необходимую нам ссылку:

* a[href=”http://www.readme.ru”]

Значение атрибута вместе с самим атрибутом указывается в квадратных скобках, примыкающих к наименованию тега, для которого будет определяться стиль. Стоит отметить, что наименование тега можно опускать, только убедитесь в том, что стиль будет применен именно для тех элементов, которые вам нужны.

Рассмотрим еще одну ситуацию. Допустим, для тегов <p>,<div>,<ol> и тега <ul>, который вложен в тег <table>, на вашем сайте применяется одинаковый стиль. В таком случае вы можете воспользоваться расширенной записью стиля:

селектор, селектор, ..., селектор

{

свойство : значение;

свойство : значение;

}

Т.е. селекторы можно указывать через запятую и к каждому из них будет применен описанный ниже стиль. Для нашей ситуации селекторы будут выглядеть следующим образом:

* p, div, ol, table ul

p, div, ol, table ul

{

color: green;

}

*Примечание: видов селекторов достаточно много, на всех мы останавливаться не будем, вы можете подробнее почитать о видах селекторов в литературе [3] и [4]. Отметим только существование универсального селектора «\*», стиль для такого селектора применяется на все теги документа.*

* 1. Классы и идентификаторы

Класс или идентификатор может быть присвоен какому-нибудь элементу (тегу) HTML посредством атрибутов class или id этого элемента (тега):

<p id=”firstP”>

<div class=”infoLayout”> . . . </div>

</p>

Имена классов и идентификаторов, в отличие от названий тегов и их атрибутов, чувствительны к регистру ввода букв.

Основное отличие между классами элементов и идентификаторами элементов в том, что в документе какой-нибудь класс может быть присвоен сразу нескольким элементам, а идентификатор — только одному. Также отличие в том, что могут существовать множественные классы (когда класс элемента состоит из нескольких слов, разделённых пробелами). Для идентификаторов такое невозможно.

Свойства стиля классов и идентификаторов задаются с помощью соответствующих селекторов. Причём может быть задано как свойство класса в целом (в таком случае селектор начинается с «.»), или свойство идентификатора самого по себе (в таком случае селектор начинается с «#»), так и свойство какого-нибудь элемента этого класса или с этим идентификатором [4].

#firstP

{

color: black;

}

.infoLayout

{

background-color: yellow;

}

или

p#firstP

{

color: black;

}

div.infoLayout

{

background-color: yellow;

}

* 1. Внутренний стиль

Стиль для тегов можно задавать различными способами, сейчас мы рассмотрим, так называемый внутренний стиль. Внутренний стиль – это стиль, который задается непосредственно через атрибут «style» для тега.

<p style=”color:green;”> Некоторый текст </p>

Поскольку стиль указывается напрямую из тега, то селектор, определяющий элемент, ставить не нужно.

<div style=”color:blue; background-color:gray;”> . . . </div>

* 1. Глобальный стиль

Если на своей странице использовать только внутренний стиль, то при появлении ошибки в стилевом оформлении (блок div залез на пункт меню, ссылка занимает половину экрана и т.д.) возникнут определенные сложности. Некорректный стиль тега, может быть унаследован от родительских тегов, в таком случае, чтобы найти ошибку, придется просмотреть и проанализировать весь документ. Это неприятно. Поэтому предпочтительнее хранить определение всех своих стилей для документа где-то в одном месте.

CSS можно подключить к документу из секции <head> с помощью контейнерного тега <style>. В таком случае, стили прописываются в начале документа, и анализировать их намного удобнее.

<html>

<head>

<style>

p { color: green; }

div { color: blue; background-color: gray; }

</style>

</head>

<body>

<p> Некоторый текст </p>

<div> . . . </div>

</body>

</html>

Не забывайте прописывать заголовок страницы и кодировку.

* 1. Связанный стиль

Зачастую таблицы стилей для документов занимают сотни, а то и тысячи строк. Помещать тысячу строк в начало документа и потом пролистывать их постоянно, чтобы добавить какой-то контент на сайт – это не лучшая идея. Определение стилей лучше выносить в отдельный файл. Чтобы добавить на страничку таблицу стилей, в секции <head> используется одиночный тег <link>.

<head>

<link rel=”stylesheet” type=”text/css” href=”D:/mySuperStyles.css”>

</head>

Рассмотрим некоторые атрибуты тега <link>:

* rel – определяет отношение между html-документом и файлом, на который делается ссылка при помощи тега <link>. Значение «stylesheet» указывает на то, что подключаемый файл хранит в себе таблицу стилей css. Возможны и другие значения этого атрибута, например, так выглядит тег <link> для изменения иконки вашей страницы на вкладке браузера: <link rel=”shortcut icon” href=”favicon.ico”>.
* type – тип данных, подключаемого файла. Значение «text/css» используется для подключаемых таблиц стилей.
* href – путь до связываемого файла. В нашем примере таблица стилей сохранена на локальном диске D в корневом каталоге под именем «mySuperStyles.css».
* charset – позволяет указать кодировку связываемого документа.

*Примечание: тегом <link> можно добавлять на сайт не только определение стилей, но также добавлять и шрифты или задавать иконку для сайта. С помощью этого тега можно даже интегрировать ленту новостей на ваш сайт.*

Стоит понимать, что в одном документе может применяться как связанный, так и глобальный или внутренний стиль (вы можете комбинировать их использование). При этом для одного тега может быть определено несколько стилевых оформлений, что приводит к конфликту в определении стиля. Такие конфликты разрешаются на основе приоритетного порядка применения стилей.

* 1. Приоритет стилей

Приоритет применения CSS к элементам HTML основан на двух принципах наследования и каскадирования.

Принцип наследования заключается в том, что стиль родительского элемента распространяется на дочерние элементы до тех пор, пока этот стиль не будет явно переопределен у дочернего элемента. В этом случае уже стиль дочернего элемента будет распространяться на всех его потомков, пока у тех не будет задан свой стиль. Так мы можем задать цвет текста для тега <body> и он автоматически будет применяться для всех тегов на странице, пока они не обзаведутся своим собственным стилем. Однако, это касается не всех стилей, такой стиль как задание рамочки вокруг тега не переносится на дочерние элементы. Так что всегда проверяйте наследуется тот или иной стиль или нет.

Принцип каскадирования вступает в силу при возникновении наложения различных стилей на один и тот же элемент. Например, для одного и того же тега задан глобальный и внутренний стиль (то есть при возникновении конфликта в определении стиля). Для разрешения конфликтов вводится правило приоритета применения стиля. Далее в порядке увеличения приоритета указаны варианты задания стиля на html-странице (следующий приоритетнее предыдущего):

* собственный стиль браузера
* стиль, заданный пользователем в настройках браузера
* стиль, заданный для html-документа
  + наследуемые стили (принцип наследования)
  + связанные стили (подключаемые внешние таблицы стилей, тег <link>)
  + глобальные стили (тег <style>)
  + внутренние стили (атрибут «style»)

Этот порядок не полный, поскольку стили можно подключать также с помощью слова !import, а для селекторов существует формула расчета их приоритета. Нам в такие тонкости углубляться сейчас не нужно, поэтому оставим порядок таким.

* 1. Свойства для изменения стиля

В этой главе мы рассмотрим некоторые свойства стилевого оформления, которые позволят «оживить» ваши html-странички. Свойства сгруппированы по тематикам. Приступим.

* + 1. Цвет

Любой цвет в CSS может быть записан различными способами:

* Словесная запись. Примером, являются «простые цвета», у которых есть английское название. Для красного – это «red», для зеленого – «green» и т.д.
* Запись шестнадцатеричного кода. Каждый цвет можно представить в RGB‑палитре. От того в каких пропорциях смешать красный (R), зеленый (G) и синий (B) цвет, будет зависеть итоговый цвет. Значения для каждого цвета варьируются от 0 до 255. Если взять 255 красного и по нулям зеленого и синего, то в итоге мы получим ярко красный цвет. Если взять каждого цвета по нулям, то получится черный цвет. Если же каждого цвета взять по 255, то получится белый цвет.

Десятичной записи чисел от 0 до 255 соответствует шестнадцатеричная запись чисел от 00 до FF. Причем значения каждого цвета указываются друг за другом без пробелов в порядке RGB (красный, зеленый и синий). Например, FF0000 – это красный цвет, 00FF00 – зеленый, 808080 – серый (между черным и белым).

Чтобы браузер понял, что речь идет о величине, указанной в шестнадцатеричной системе исчисления, перед значением необходимо добавить символ решетки «#»: «#FF0000», «#00FF00» и «#808080»

* Запись в десятичной системе исчисления. Мы уже разобрались с тем, что концентрацию каждого цвета можно указывать в диапазоне от 0 до 255. Однако, для записи цвета в десятичной системе существует не только порядок RGB. Поэтому для того, чтобы явно указать браузеру, что мы задаем цвета в порядке «красный, зеленый и синий» необходимо воспользоваться конструкцией:

rgb(величина красного, величина зеленого, величина синего цвета). Например,  
rgb(255, 0, 0) для красного цвета. Если вы хотите дополнительно указывать альфа‑канал (прозрачность), то воспользуйтесь конструкцией вида:  
rgba(255, 255, 0, 128). Величина альфа-канала также варьируется от 0 до 255.

Рассмотрим свойства, которые позволят изменить цвет текста, цвет фона или добавить фоновое изображения для элемента.

* color. Свойство «color» позволяет изменять цвет текста внутри элементов, соответствующих заданному селектору.

p { color: red; }

div { color: #FF0000; }

span { color: rgb(255, 0, 0); }

* background-color. Свойство «background-color» позволяет задавать цвет фона для элемента (цвет заливки).

p { background-color: #00FFFF; }

* background-image. Свойство «background-image» позволяет задавать фоновое изображение.

div { background-image : url(‘D:/mySuperImage.png’); }

Путь до изображения указывается внутри специальной конструкции url(). При этом сам путь можно как записывать внутри кавычек (одинарных или двойных), так и без них.

* + 1. Геометрия

Под геометрией мы здесь будем понимать не форму элементов, а их размеры. Более конкретно, нас интересует только высота и ширина элементов. Ширину и высоту можно задавать как в пикселях: «100px» , - так и в процентном соотношении относительно размеров родительского элемента: «73%».

Для задания ширины используется свойство «width»:

.layout { width: 100%; }

Для изменения высоты элемента используется свойство «height»:

#mainLayout { height: 400px; }

* + 1. Шрифт

Среди свойств, изменяющих отображение текста на странице, нас в первую очередь будет интересовать изменение гарнитуры и размера шрифта.

Свойство «font-family» изменяет гарнитуру элемента. В этом свойстве допускается указывать сколько угодно семейств шрифтов через запятую. Делается это потому, что некоторые специализированные шрифты, могут не поддерживаться браузерами или могут не прогружаться по ряду причин. В этом случае браузер будет пытаться применить следующий по порядку, указанный вами шрифт. Важное правило: если в записи гарнитуры есть пробелы, то ее название обязательно надо брать в кавычки.

p { font-family: Verdana, ‘Times New Roman’, serif; }

Для того, чтобы изменить размер шрифта применяется специальное свойство «font-size». Причем задать размер можно различными способами:

* Используя набор констант.
  + Набор констант (xx-small, x-small, small, medium, large, x-large, xx-large) задает размер, который называется абсолютным. На значение этих констант влияют настройки браузера и операционной системы.
  + Набор констант (larger, smaller) устанавливает относительные размеры шрифта. Поскольку размер унаследован от родительского элемента, эти относительные размеры применяются к родительскому элементу, чтобы определить размер шрифта текущего элемента.
* Используя различные единицы измерения. 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 [3].

<h1 style=”font-size: 14pt;”>Текст заголовка</h1>

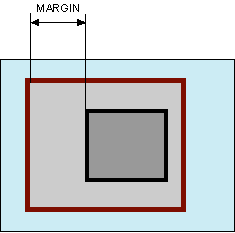
*Примечание: почти у всех свойств CSS можно указывать специальное значение «inherit», которое указывает на то, что значение необходимо копировать из родительского элемента.*

* + 1. Отступы

Порою возникают такие ситуации, когда нам необходимо сделать отступ в несколько вертикальных пробелов вокруг кого-либо тега, или разместить текст внутри тега с отступом от границ самого тега. За такие специальные расстояния для отступов отвечают свойства margin и padding.

* Margin

Устанавливает величину отступа от каждого края элемента. Отступом является пространство от границы текущего элемента до внутренней границы его родительского элемента [3].



*На рисунке показан отступ слева от границы элемента. Отступ может быть, и справа, и сверху, и снизу, и слева от элемента.*

* + margin-left. Задает величину отступа слева от границ элемента в пикселях или в процентах (возможны также значения inherit и auto – если вы понимаете о чем мы).
  + margin-top. Задает величину отступа от границ элемента сверху.
  + margin-right. Задает величину отступа справа от границ элемента.
  + margin-bottom. Задает величину отступа снизу от границ элемента.

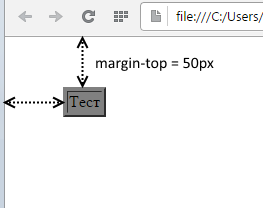
<table border=2 style=”background-color: gray; margin-left: 50px; margin-top: 50px;”>

<tr>

<td> Тест </td>

</tr>

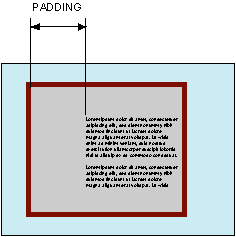
</table>



*Примечание: мы никогда не оговаривали такой момент, что не стоит копировать html-текст представленный здесь, если вы пользуетесь электронным вариантом этого материала. Дело все в том, что кавычки, которые устанавливает по умолчанию Microsoft Word © для своих документов, имеют не тот ASCII-код, который необходим html-документам. Поэтому, пожалуйста, если вы копируете код прямо в текстовый редактор – простите нас, но мы такой подход не одобряем – вам как минимум придется исправить кавычки (с косых на прямые).*

* Padding

Устанавливает значение полей вокруг содержимого элемента. Полем называется расстояние от внутреннего края рамки элемента до воображаемого прямоугольника, ограничивающего его содержимое [3].



*На рисунке показано поле слева от границы элемента. Поле может быть, и справа, и сверху, и снизу, и слева от границ элемента.*

* + padding-left. Задает величину поля элемента слева от его границы.
  + padding-top. Задает величину поля элемента сверху от его границы.
  + padding-right. Задает величину поля элемента справа от его границы.
  + padding-bottom. Задает величину поля элемента снизу от его границы.

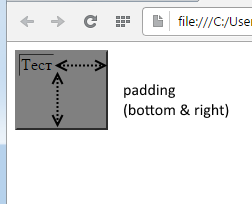
<table border=2 style="background-color: gray; padding-right: 50px; padding-bottom: 50px;">

<tr>

<td> Тест </td>

</tr>

</table>



*Примечание: обратите внимание, что у некоторых рассмотренных нами свойств есть одинаковые префиксы (первая часть в записи): font-size и font-family, background-color и background‑image, margin-left и margin-top. На самом деле все эти свойства – это конкретизация более общих свойств: font, background и margin. Существуют и такие общие записи, как: «margin: 10px 20px 10px 10px;», - в такой записи мы сразу задали значения для left, top, right и bottom величин у общего свойства margin (аналогично можно задать значения сразу для всех полей свойства padding). Чтобы пользоваться такой записью, надо знать все конкретные свойства более «общего» свойства (мы знаем все свойства, на которые разделено общее свойство margin). К сожалению, в рамках учебного материала, который ограничен по времени изучения, мы не можем останавливаться на всех «подсвойствах», поэтому зачастую будем опускать общие записи.*

* + 1. Выравнивание

Выравнивание элементов вполне можно задавать с помощью атрибута «align» для тех тегов, у которых он поддерживается. Здесь мы обратим внимание только на выравнивание текста внутри документа.

* text-align. Задает выравнивание текста по горизонтали, может принимать значения: «left» (выравнивание по левому краю), «center» (выравнивание по горизонтальному центру элемента), «right» (выравнивание по правому краю) и «justify» (выравнивание по ширине).

<table border=1>

<tr>

<td id="leftTD">Текст выравнивается слева</td>

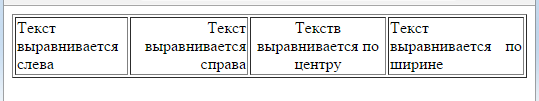
<td id="rightTD">Текст выравнивается справа</td>

<td id="centerTD">Текств выравнивается по центру</td>

<td class = "justify">Текст выравнивается по ширине</td>

</tr>

</table>



*Примечание: для выравнивания элементов по вертикали может применяться свойство vertical-align, принимающее значения: «top», «middle», «bottom», «sub», «super» и другие (sub и super служат для отображения надстрочного или подстрочного текста).*

* + 1. Границы

Каждому элементу можно задавать свою границу. В данном случае мы рассмотрим сразу полную запись свойства «border», опустив такие его «подсвойства» как, например, «border-width».

При задании границы с помощью свойства «border» сначала указывается толщина линии для границы в пикселях, затем указывается стиль линии (сплошная, пунктирная) и, наконец, задается цвет линии для границы. Цвет линии можно задать любыми известными нам способами, а что касается стиля линии, то он может принимать одно из следующих значений:



<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<title>Учебная страница</title>

<style>

.solidBorder { border: 3px solid black; }

.grayBackground { background-color: gray; }

#marginInside { margin: 50px 50px 50px 50px; }

</style>

</head>

<body>

<div class="solidBorder grayBackground">

<div id="marginInside" style="background-color: white;">Некоторый текст</div>

</div>

</body>

</html>



Этот пример может работать некорректно в некоторых браузерах, поскольку иногда к блокам div применяется стиль с обтеканием по умолчанию. В этом случае его необходимо отменять. Посмотрите с помощью инспектора элементов в браузере (правая кнопка по содержимому страницы в браузере, затем выберете просмотр кода элементов или инспектор элементов). Этот инспектор позволит вам просмотреть какие стили применены к тому или иному тегу. Если вы увидите там что-то лишнее, что вы не задавали, то советуем вам сбросить это значение. Обычно это можно сделать, задав соответствующему свойству значение «none». Однако, не всегда. Мы не можем рассмотреть тут все возможные варианты, но имейте в виду, что такая ситуация возможна.

*Примечание: надеемся, вы понимаете, что это далеко не полный перечень возможных свойств для изменения стилевого оформления. Так мы упустили из виду целый раздел «обтекание элементов», с помощью которого можно добиться такого эффекта, что абзацы и элементы не будут занимать строки целиком, а можно разместить несколько <div> или <p> в одной строке как колонки, заменяя ими таблицу. Если вы действительно хотите добиться каких-то успехов в web-разработке, то советуем вам воспользоваться ссылками [2] и [3].*

* 1. Задания
* Какие селекторы используются в записи

body > .infoLayout, #mainLink div.classA > img#mainImage, p

{

}

?

* Что такое классы и идентификаторы? В чем их отличие?
* Что такое CSS?
* Какими способами можно задать стиль?
* Объясните последний пример (в разделе границы).
* Что можно изменять с помощью стилей на странице?
* Создайте какую-нибудь html-страничку с применением CSS. Продемонстрируйте свой результат преподавателю, будьте готовы объяснить каждую строчку в коде (чем больше стилей вы охватите, тем лучше).

1. ***Форма в HTML***
   1. Что такое форма

Формы в html используются для обмена данными между пользователем и сервером. С помощью формы вы можете заполнить какую-нибудь анкету и отослать ее администратору сайта или отослать на почту. На форме могут быть размещены различные элементы: текстовые поля, кнопки, выпадающие списки, элементы выбора и другие. Если на вашем сайте нужна какая-либо обратная связь с пользователем (ввод логина и пароля, ввод своей почты, регистрация, сообщения на форуме и т.д.), то вам следует добавить форму с необходимыми элементами на вашу страничку.

Форму можно добавить при помощи контейнерного тега <form>. Документ может содержать несколько форм, но одновременно на сервер может отправиться только одна форма. Естественно, что при таком ограничении, ваши формы не должны напрямую зависеть друг от друга (за регистрацию отвечает одна форма, а за отправку сообщений для обратной связи – другая). При этом не рекомендуется использовать для различных целей одну форму, то есть объявление конструкции <body><form> … </form></body> - это очень плохая практика, никогда не оборачивайте всю страницу в форму. Используйте формы только в тех местах, где это необходимо. Если не соблюдать это правило, то роботы поисковых систем, скорее всего, не будут индексировать ваш сайт и посчитают, что на нем нет контента, поскольку формы им не интересны.

Каждый элемент формы связывается с каким-то системным именем (name), при заполнении этого элемента каким-то значением (value), образуется пара «name=value» (имя-значение). Эти пары передаются на сервер. Допустим, вы указали, что поле для ввода логина называется «LoginField», далее вы заполнили это поле значением «Admin», затем нажали кнопку «Отправить». В итоге на сервер отошлется пара «LoginField=Admin». Все символы, отличные от латинских букв, будут экранироваться записью «%hh», где hh – это шестнадцатеричный код символа в ASCII-таблице (русские символы будут экранироваться), пробел заменяется на знак «+».

Существует два основных метода передачи данных на сервер: POST и GET. Мы рассмотрим только одно из основных их отличий.

* GET. При передаче пар «имя-значение» этим способом, мы можем наблюдать изменения в адресной строке браузера. Параметры добавляются прямо в web-адрес. Допустим, мы передаем две пары: «name=Egor» и «age=47», со страницы <http://www.mypage.com>. После отправки этих данных методом GET адрес в строке станет выглядеть следующим образом: <http://www.mypage.com?name=Egor&age=47>.

Видите, как преобразился web-адрес? После начального адреса страницы добавился символ «?», который свидетельствует о том, что далее будут указываться параметры с помощью метода GET. Сами пары «имя=значение» разделяются символом «&». Вы наверняка уже встречали такие адреса при поиске информации в поисковых системах. Поле для ввода поискового запроса – это элемент формы.

* POST. При передаче данных методом POST, адрес ссылки не модифицируется. Пары «имя=значение» передаются в метаданных страницы. Эти данные можно найти средствами браузеров (если они передавались), просматривая свойства html‑страницы.

Метод передачи данных задается с помощью специального атрибута «method».

Рассмотрим некоторые из атрибутов тега <form>:

* action. Адрес документа, электронной почты или программы, которые будут обрабатывать данные формы. При этом данные могут обрабатываться и той же страницей, где находится форма, в этом случае можно, например, указать: action=”” или action=”#”.
* method. Метод протокола HTTP: POST или GET (например, method=”GET”).
* name. Задает имя формы. Рекомендуем вам указывать каждой форме ее уникальное имя.
* target. Имя окна или фрейма, куда обработчик будет загружать возвращаемый результат.

Сама форма не отображается на странице, отображаются только находящиеся в ней элементы. Пример использования контейнерного тега <form>:

<form action=”#” method=”GET” name=”RegistrationForm”> . . . </form>

* 1. Элементы формы

Элементы формы располагаются внутри контейнерного тега <form>. Под элементами формы понимаются различные поля ввода, списки с выбором, флаги и т.д. Основным тегом для задания элементов формы является одиночный тег <input>, у которого есть специальный атрибут «type», в зависимости от значения которого, на форме будут располагаться различные элементы.

* + 1. Строковое поле для ввода

Полю ввода соответствует тип «text» для одиночного тега <input>:

<body>

<form name=”testForm” action=”” method=”GET”>

<input type=”text”>

</form>

</body>



*Обратите внимание, что элементы формы прописываются внутри контейнерного тега <form>.*

Наиболее важными атрибутами, если мы говорим о передаче данных с формы на сервер, являются атрибуты «name» и «value». Не забывайте указывать имя для каждого элемента формы, иначе у вас возникнут трудности в определении нужных пар «имя-значение», когда вы будете просматривать очередную пачку данных пришедшую вам на сервер.

<form name="testForm" action="" method="GET">

<input type="text" name="textField" value="Некоторый текст">

</form>

При отправке данных в «value» запишется тот текст, который введен в текстовое поле с именем «name». При необходимости вы можете изначально выводить текстовые поля с некоторым текстом, заполнив нужным содержимым атрибут «value».



При отправке данных с такой формы, адрес преобразиться из «site.html» в «site.html?textField=Некоторый+текст» (в зависимости от браузера русские символы могут экранироваться в адресной строке).

Рассмотрим некоторые дополнительные атрибуты текстового поля:

* size. Задает ширину текстового поля («size=20»).
* maxLength. Определяет количество символов, разрешенных для ввода в текстовое поле.
* readonly. Указывает на то, что текст в поле нельзя изменять (только для чтения). Данное поле не требует значения.

<input type=’text’ value=’меня не поменять’ readonly>

* placeholder. Задает текст подсказки. Подсказка отображается до тех пор, пока в поле не будет введен текст.

<input type="text" placeholder="Введите логин">



Не огорчайтесь, если этот и некоторые другие атрибуты не заработают у вас в компьютерном классе, возможная причина может крыться в версии используемого браузера.

Бывают такие ситуации, когда пользователю приходится вводить какую-то конфиденциальную информацию в ваше текстовое поле, например, пароль. Для таких случаев лучше применять специализированное текстовое поле, которое заменит все вводимые символы на кружочки. Используйте тип элемента «password» для достижения такого эффекта.

<input type="password" size="50">



* + 1. Кнопка Submit

В принципе, мы с вами уже научились вводить какие-то данные через поле ввода. Теперь посмотрим, каким способом эти данные можно отправить на сервер, указанный в атрибуте «action» тега <form>. Оказывается, что специально для этих целей существует кнопка отправки данных. Это обычный элемент формы с типом «submit». Атрибут «value» задает текст кнопки. По нажатию левой кнопкой мыши на этот элемент все данные будут отправлены на сервер.

<form name="testForm" action="" method="GET">

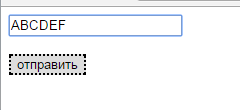
<input type="text" name="textField">

<br>

<br>

<input type="submit" value="отправить" style="border: 2px dotted black;">

</form>



После нажатия на кнопку, произойдут изменения в адресной строке (метод GET)



* + 1. Элемент «Флажок»

Используются для выбора вариантов из предложенных. Разрешается выбирать более одного элемента одновременно. Для создания такого элемента указываем тип «checkbox». При передаче на сервер будут учитываться только выбранные элементы.

<form name="testForm" action="" method="GET">

<h4>Что вам нравится?</h4>

<input type="checkbox" name="cofee" value="yes">Кофе

<br>

<input type="checkbox" name="tea" value="yes">Чай

<br>

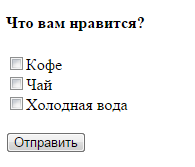
<input type="checkbox" name="water" value="cold">Холодная вода

<br>

<br>

<input type="submit">

</form>.



* + 1. Элемент «Переключатель».

Переключатели следует использовать, когда подразумевается единственный выбор среди предложенных. Поскольку на одной странице может располагаться много переключателей по разным тематикам, то переключатели нужно уметь как-то различать. Для этих целей переключатели объединяют в группы. Именно внутри одной группы возможно выбрать только один переключатель. Для того, чтобы объединить переключатели в группу для них необходимо использовать одно и тоже значение атрибута «name». Переключателю соответствует тип «radio». При передаче данных на сервер отправляется только одна пара «имя-значение» для каждой из групп переключателей. Эта пара соответствует выбранному переключателю.

<form name="testForm" action="" method="GET">

<b>Ваш пол?</b><br>

<input type="radio" name="gender" value="m"> Мужской <br>

<input type="radio" name="gender" value="f"> Женский <br>

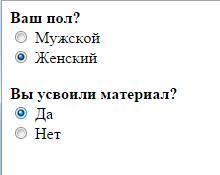
<br>

<b>Вы усвоили материал?</b><br>

<input type="radio" name="understand" value="yes"> Да <br>

<input type="radio" name="understand" value="no"> Нет <br>

</form>



* + 1. Кнопка сброса введенных данных

Для того, чтобы сбросить все введенные в форму данные, можно воспользоваться специальным элементом – кнопкой сброса. Для того, чтобы создать такую кнопу укажите тип «reset» для тега <input>. После нажатия на этот элемент, все данные в форме примут первоначальное значение.

<input type="reset" value="Сбросить значения">

* + 1. Кратко об оставшихся в тени типах тега <input>

Мы не можем так детально останавливаться на каждом значении атрибута в силу ряда причин. Поэтому на остальных значениях атрибута «type» для тега <input> не будем долго задерживаться. Возможные типы и их описание сведены в таблицу, взятую со страниц online‑документации [2].

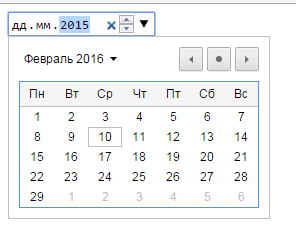
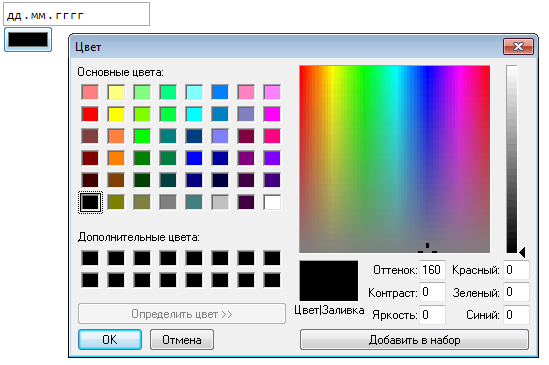
|  |  |
| --- | --- |
| **Тип** | **Описание** |
| button | Кнопка. |
| checkbox | Флажки. Позволяют выбрать более одного варианта из предложенных. |
| file | Поле для ввода имени файла, который пересылается на сервер. |
| hidden | Скрытое поле. Оно никак не отображается на веб-странице. |
| image | Поле с изображением. При нажатии на рисунок данные формы отправляются на сервер. |
| password | Обычное текстовое поле, но отличается от него тем, что все символы показываются звездочками. Предназначено для того, чтобы никто не подглядел вводимый пароль. |
| radio | Переключатели. Используются, когда следует выбрать один вариант из нескольких предложенных. |
| reset | Кнопка для возвращения данных формы в первоначальное значение. |
| submit | Кнопка для отправки данных формы на сервер. |
| text | Текстовое поле. Предназначено для ввода символов с помощью клавиатуры. |

С появлением HTML 5 были добавлены дополнительные элементы и значения для атрибута «type» соответственно.

|  |  |
| --- | --- |
| **Тип** | **Описание** |
| color | Виджет для выбора цвета. |
| date | Поле для выбора календарной даты. |
| datetime | Указание даты и времени. |
| datetime-local | Указание местной даты и времени. |
| email | Для адресов электронной почты. |
| number | Ввод чисел. |
| range | Ползунок для выбора чисел в указанном диапазоне. |
| search | Поле для поиска. |
| tel | Для телефонных номеров. |
| time | Для времени. |
| url | Для веб-адресов. |
| month | Выбор месяца. |
| week | Выбор недели. |

Что касается новых типов, то многие из них не поддерживаются в браузерах: Internet Explorer, Firefox и Android (встроенного браузера).

Приведем пример элементов «date» и «color» (браузер Opera).



* + 1. Многострочное поле для ввода

Помимо элементов формы, создаваемых с помощью тега <input>, можно добавлять также и контейнерные элементы. Например, многострочное поле ввода. Обычное текстовое поле не сохраняет переносы строк, что ограничивает пользователя в форматировании текста. Создается многострочное поле для ввода контейнерным тегом <textarea>. Между тегами <textarea> и </textarea> можно поместить любой текст, который будет отображаться внутри поля. Поскольку вводимый текст помещается внутри тега, то атрибут «value» у такого элемента отсутствует.

<form name="testForm" action="" method="GET">

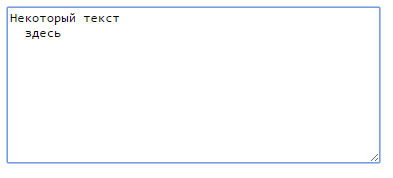
<textarea name="memoField" cols=50 rows=10>

Некоторый текст

здесь

</textarea>

</form>



Обратите внимание, что содержимое тега <textarea> мы указали без табуляций до уровня тега. Сделали мы это намерено, поскольку этот тег сохраняет форматирование: сколько в него ввести пробелов (даже в начале), столько пробельных символов он и отобразит.

Не забывайте указывать значение атрибута «name» для тега. В приведенном примере мы использовали два дополнительных атрибута «cols» и «rows».

* cols. Задает количество символов в строке, на которые рассчитана область для ввода.
* rows. Задает количество строк, на которые рассчитана область для ввода.

То есть эти два атрибута отвечают за ширину и высоту в символах для области ввода текста.

* + 1. Список выбора

В зависимости от количества элементов, которые доступны для выбора, различают множественный список выбора и раскрывающийся список. Чтобы создать раскрывающийся список необходимо воспользоваться специальным контейнерным тегом <select>. Элементы доступные для выбора, указываются внутри этого тега и создаются с помощью контейнерного тега <option>.

<form name="testForm" action="" method="GET">

<select name="country">

<option>Англия</option>

<option>Россия</option>

<option>Япония</option>

</select>

</form>



Несколько украсим наш список при помощи некоторых атрибутов.

<form name="testForm" action="" method="GET">

<select name="country">

<option disabled>Выберете страну</option>

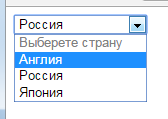
<option>Англия</option>

<option selected>Россия</option>

<option>Япония</option>

</select>

</form>



В приведенном примере появилась новая «опция» для выбора, используем ее как подсказу, которая сообщает пользователю, для чего предназначен этот список. Поскольку подсказка должна только отображаться («Выберете страну»), а выбор для нее должен быть недоступен, сделаем ее неактивной с помощью специального атрибута «disabled» (атрибут без значения – флаг). Также укажем значение по умолчанию, с помощью специального флага «selected» для страны «Россия».

Наш список стал выглядеть достойно, но это не все «хитрости», с помощью которых можно добиться приятных визуальных эффектов. Помимо простого перечисления вариантов выбора, теги <option> можно объединять в специальные группы <optgroup>. При использовании групп список принимает древовидную структуру, а названия групп выделяются особым шрифтом и не доступны для выбора. Название группы задается с помощью атрибута «label» тега <optgroup>.

<form name="testForm" action="" method="GET">

<select name="country">

<option disabled>Выберете страну</option>

<optgroup label="Азия">

<option>Китай</option>

<option>Северная Корея</option>

<option>Япония</option>

</optgroup>

<optgroup label="Африка">

<option>Ливия</option>

<option>Эфиопия</option>

</optgroup>

<optgroup label="Европа">

<option>Англия</option>

<option>Бельгия</option>

<option>Германия</option>

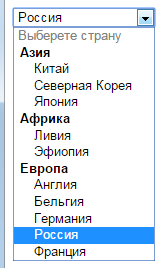
<option selected>Россия</option>

<option>Франция</option>

</optgroup>

</select>

</form>



Стало намного лучше, не так ли? Однако наш раскрывающийся список пока не несет никакой понятной информации для сервера (скрипта), который будет обрабатывать выбор пользователя. Для этого необходимо заполнить атрибут «name» у тега <select> и атрибут «value» у всех опций выбора <option>.

Для того, чтобы превратить раскрывающийся список выбора в множественный список выбора, необходимо заполнить значение атрибута «size», у тега <select> любым значением большим чем единица. Заданное значение указывает, сколько элементов списка отображать на экране одновременно (количество строк в списке, по умолчанию это значение равно единице).

<form name="testForm" action="" method="GET">

**<select name="country" size=11 multiple>**

<option disabled>Выберете страну</option>

<optgroup label="Азия">

<option value="CH">Китай</option>

<option value="NK">Северная Корея</option>

<option value="JA">Япония</option>

</optgroup>

<optgroup label="Африка">

<option value="LI">Ливия</option>

<option value="EP">Эфиопия</option>

</optgroup>

<optgroup label="Европа">

<option value="EN">Англия</option>

<option value="BL">Бельгия</option>

<option value="GE">Германия</option>

<option value="RU" selected>Россия</option>

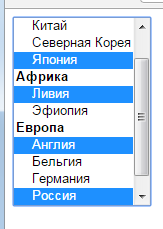
<option value="FR">Франция</option>

</optgroup>

</select>

</form>

Мы задали «size=11» и указали специальный атрибут «multiple», который позволяет совершать множественный выбор в списке (при зажатой кнопке «ctrl» или «shift» будет осуществляться соответствующее выделение).



* 1. Задания
* Зачем нужна форма?
* Какие методы передачи данных на сервер вы знаете?
* Какие элементы формы вам известны?
* Создайте форму для регистрации нового пользователя (обязательны к заполнению: логин, почта, пароль, повтор пароля, пол, страна, кнопка отправить).
* Создайте форму обратной связи (обязательны к заполнения: адрес почты, тема сообщения, выбор тематики ошибки из списка, текст сообщения, кнопка отправить).
* Создайте произвольную форму на ваше усмотрение (используйте не менее 7 различных видов элементов формы и кнопку отправить).