**ЗАНЯТИЕ 3. ВЕРСТКА: ПЕРВЫЕ ШАГИ. СТРУКТУРА HTML-ДОКУМЕНТА**

**3.1 От макета к веб-странице**

После того, как мы научились делать простые PSD-макеты, возникает закономерный вопрос о том, как же перенести их на веб-страницу. В течение следующих нескольких занятий мы поговорим о верстке и научимся верстать стандартными средствами: HTML и CSS.

Итак, начнем с того, что же такое HTML (HyperText Markup Language) – это язык разметки гипертекста, система верстки, которая определяет, как и какие элементы должны располагаться на веб-странице. Информация на сайте, способ ее представления и оформления зависят исключительно от разработчика и тех целей, которые он перед собой ставит. Собственно, нам необходимо научиться корректно разбивать макет на условные блоки для верстки и правильно их располагать, однако об этом немного позже.

Для работы нам понадобится ряд инструментов:

* Текстовый редактор.
* Браузер для просмотра результатов.
* Графический редактор.

Пара слов о текстовом редакторе. Я использовала Notepad++ и Dreamweaver от Adobe. В конечном итоге все же остановилась на первом варианте, во-первых – по причине того, что ПО бесплатное, во-вторых – простота и удобство, абсолютно ничего лишнего. Впрочем, это дело вкуса, но с редактором необходимо определиться.



Что касается браузера – лучше использовать популярные, так как некоторые браузеры могут не поддерживать различные стили и даже js-команды. Из-за возможных различий в отображении браузерами одного и того же кода возникает проблема идентификации браузера и его версии, чтобы «подсунуть» ему персональный код. Браузер IE поддерживает специальную технологию определения версии под названием «условные комментарии». Об этом можно подробнее прочесть [здесь](http://htmlbook.ru/samlayout/internet-explorer/uslovnye-kommentarii).

Графических редакторов не счесть, поэтому здесь почти полная свобода действий. Я использую Photoshop в виду удобства работы с макетом. К тому же, чаще всего, если вы будете верстать по уже готовому дизайну, макет будет получать именно в PSD-формате, это тоже весомый аргумент.

**3.2 Первый HTML-документ**

Итак, начнем непосредственно разговор об HTML. Ключевой единицей является тег, он используется для того, чтобы браузер при отображении документа понимал, что имеет дело не с простым текстом, а с элементом форматирования.

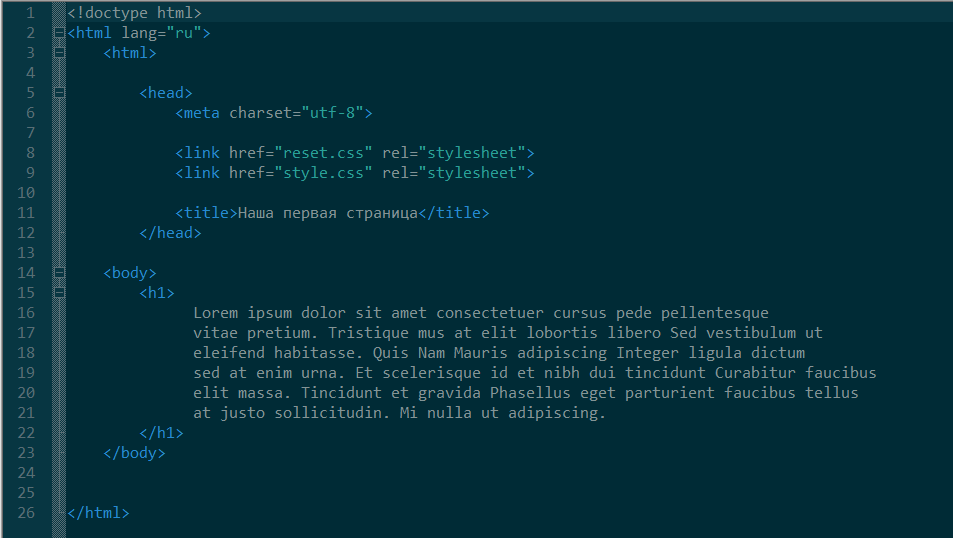
Общий синтаксис написания тегов:

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

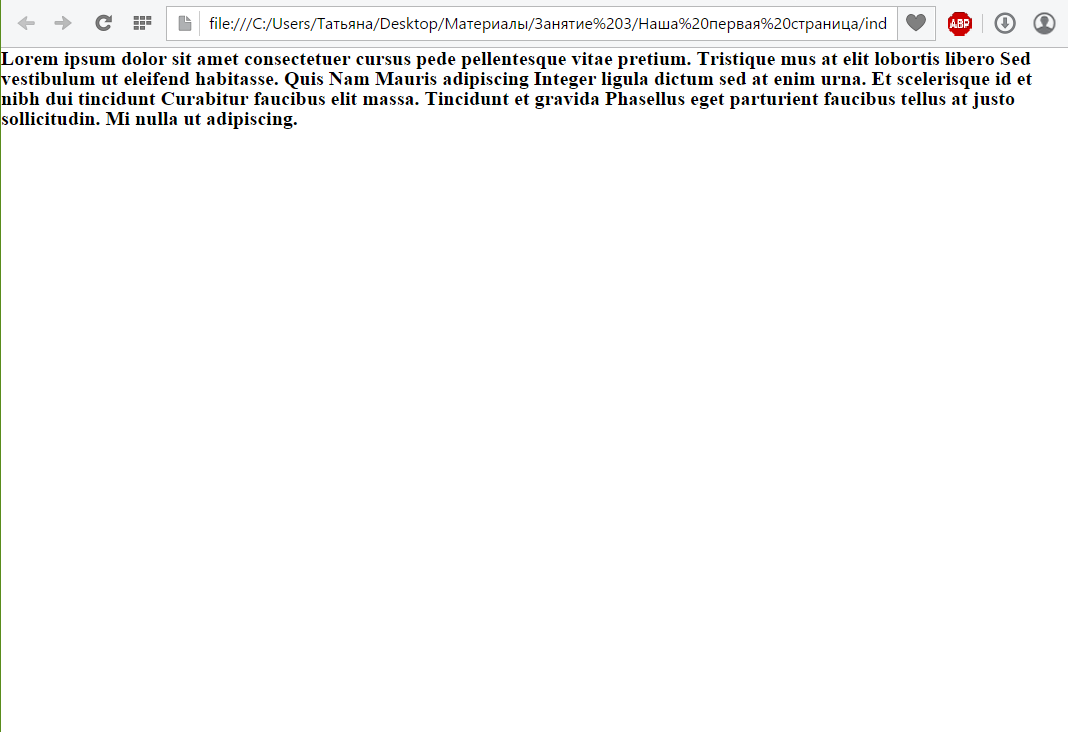
**<тег атрибут1="значение" атрибут2="значение">...</тег>**

Теги делятся на парные и одиночные. Одиночный используется самостоятельно, а парный может содержать в себе другие элементы. Парные теги, называемые по-другому контейнеры, состоят из двух частей — открывающий и закрывающий тег. Открывающий тег обозначается как и одиночный — **<тег>**, а в закрывающем используется слэш — **</тег>**. Также теги могут иметь атрибуты, в зависимости от типа тега атрибуты бывают обязательными и необязательными.

Создадим нашу первую html-страничку и разберем основные теги. Используются одиночные теги **<meta>** и **<link>**  а парных тегов сразу несколько: **<html>**, **<head>**, **<title>**, **<body>,** и **<h1>**.



А вот как это выглядит в браузере:



Для чего все эти теги? Сейчас разберемся.

Все начинается со строчки <!doctype html>. Элемент <!doctype> предназначен для указания типа текущего документа — DTD (document type definition, описание типа документа). Это необходимо, чтобы браузер понимал, как следует интерпретировать текущую веб-страницу, поскольку HTML существует в нескольких версиях, кроме того, имеется XHTML (EXtensible HyperText Markup Language, расширенный язык разметки гипертекста), похожий на HTML, но различающийся с ним по синтаксису. Чтобы браузер «не путался» и понимал, согласно какому стандарту отображать веб-страницу и необходимо в первой строке кода задавать **<!doctype>**.

Далее открывающий тег **<html>.** Тег **<html>** определяет начало HTML-файла, внутри него хранится заголовок (**<head>**) и тело документа (**<body>**).

Заголовок документа, как еще называют блок **<head>**, может содержать текст и теги, но содержимое этого раздела не показывается напрямую на странице, за исключением контейнера **<title>** (это название страницы, которое будет отображено как имя вкладки в браузере). В пределах **<head>** встречается еще два тега**: <meta> и <link>.** Первый указывает кодировку документа. Атрибут введен в HTML5 и предназначен для сокращения формы тега **<meta>**, которая задавала кодировку в предыдущих версиях HTML и XHTML. Второй - устанавливает связь с внешним документом вроде файла со стилями или со шрифтами.

Элемент **<body>** предназначен для хранения содержания веб-страницы (контента), отображаемого в окне браузера. Информацию, которую следует выводить в документе, следует располагать именно внутри контейнера **<body>**. К такой информации относится текст, изображения, теги, скрипты JavaScript и т.д. Открывающий и закрывающий теги **<body>** на веб-странице не являются обязательными, однако хорошим стилем считается их использование, чтобы определить начало и конец HTML-документа.

HTML предлагает шесть заголовков разного уровня, которые показывают относительную важность секции, расположенной после заголовка. Так, тег **<h1>** представляет собой наиболее важный заголовок первого уровня, а тег **<h6>** служит для обозначения заголовка шестого уровня и является наименее значительным. По умолчанию, заголовок первого уровня отображается самым крупным шрифтом жирного начертания, заголовки последующего уровня по размеру меньше. Теги **<h1>**,...,**<h6>** относятся к блочным элементам, они всегда начинаются с новой строки, а после них другие элементы отображаются на следующей строке. Кроме того, перед заголовком и после него добавляется пустое пространство.

Последним элементом в коде всегда идет закрывающий тег **</html>**.

**Вот и все теги, требующиеся для создания простейшей страницы.**

Несколько слов о комментариях:

<!-- Комментарий -->

Некоторый текст можно спрятать от показа в браузере, сделав его комментарием. Хотя такой текст пользователь не увидит, он все равно будет передаваться в документе, так что, посмотрев исходный код, можно обнаружить скрытые заметки.

Комментарии нужны для внесения в код своих записей, не влияющих на вид страницы. Начинаются они тегом **<!--** и заканчиваются тегом **-->**. Все, что находится между этими тегами, отображаться на веб-странице не будет.

**3.3 Виды верстки**

Одним из самых популярных споров между верстальщиками - это, какая вёрстка лучше: **табличная или блочная**. Вопрос этот очень спорный и каждый по-своему прав.

Начну с **преимуществ и недостатков табличной вёрстки**:

* Таблицы не перекрываются друг с другом при маленьких разрешениях.
* Легко делать кроссбраузерный дизайн.
* Гораздо проще блочной вёрстки.
* Очень много лишнего кода, ввиду бесконечного создания строк и столбцов.
* Далеко не каждый дизайн можно создать с помощью таблиц.

Теперь о **преимуществах и недостатках блочной вёрстки**:

* Значительно меньше **HTML-кода** и, как следствие, уменьшение веса страницы.
* Блоки загружаются быстрее таблиц (особенно больших таблиц).
* В отличии от таблиц, блоки - универсальное средство для создания любого дизайна.
* Гораздо сложнее **табличной вёрстки**.
* Огромные проблемы с **кроссбраузерностью**.
* Блоки начинают наезжать (либо спадать) друг на друга при маленьких разрешениях экрана.

**3.4 Блочная верстка**

С помощью блочной верстки можно создать практически любой дизайн, поэтому рассмотрим именно этот метод. Выражение «блочная вёрстка» или вёрстка с помощью слоёв заключается в конструктивном использовании тегов **<div>** и стилей. При этом придерживаются следующих принципов.

**Разделение содержимого и оформления**

Код HTML должен содержать только теги разметки и теги логического форматирования, а любое оформление выносится за пределы кода в стили. Такой подход позволяет независимо управлять видом элементов страницы и её содержимым. Благодаря этому над сайтом может работать несколько человек, при этом каждый выполняет свою функцию самостоятельно от других. Дизайнер, верстальщик и программист работают над своими задачами автономно, снижая время на разработку сайта.

**Активное применение тега <div>**

При блочной вёрстке существенное значение уделяется универсальному тегу **<div>**, который выполняет множество функций. Фактически это основа, на которую «навешиваются» стили, превращая её то в игрушку, то в зверушку. Совершенно не значит, что применяется только один этот тег, нужно ведь и рисунки вставлять и оформлять текст. Но при вёрстке с помощью слоёв тег **<div>** является кирпичиком вёрстки, её базовым фундаментом.

Благодаря этому тегу HTML-код распадается на ряд чётких наглядных блоков, код при этом получается более компактным, чем при табличной вёрстке, к тому же поисковые системы его лучше индексируют.

**Таблицы применяются только для представления табличных данных**

При блочной вёрстке, конечно же, используются таблицы, но только в тех случаях, когда они нужны, например, для наглядного отображения чисел и других табличных данных. Вариант, когда от таблиц предлагается отказаться вообще, является нецелесообразным и, более того, вредным.

При блочном подходе уделяется тегу **<div>**, с которым у большинства людей и ассоциируются слои. В каком-то смысле это является верным, поэтому договоримся в дальнейшем употреблять термин «слой» к тегу **<div>** для которого указан стилевой идентификатор или класс. Таким образом, выражение «слой с именем content» подразумевает, что используется тег <div id="content"> или <div class="content">.

В HTML5 добавлено несколько новых тегов разметки для обозначения разных типовых блоков страницы. К примеру, **<header>** и **<footer>**используются для создания «шапки» и «подвала», **<nav>** для навигации, **<aside>** для боковой панели. Включение в спецификацию HTML подобных элементов призвано снизить доминирование тега **<div>** и придать больше смысла разметке. Поэтому в вёрстке на HTML5 активно применяется термин «элемент», под которым подразумевается соответствующий тег и элемент который он создаёт.

Изложенные выше принципы блочной вёрстки при этом сохраняются за исключением того момента, что **<div>** в некоторых случаях заменяется более осмысленными тегами.

Ниже приведена таблица с основными тегами, нужными при разработке:

|  |  |
| --- | --- |
| Тег | Описание |
| <script> | Содержит внутри скрипты, которые иногда располагаются во внешнем файле. |
| <title> | Определяет заголовок страницы, отображается в имени вкладки браузера. |
| <head> | Является контейнером для элементов, которые помогают браузеру по работе с данными. |
| <body> | Содержит контент веб-страницы, который будет отображаться в окне браузера. |
| <p> | Текстовый абзац |
| <a> | Тег для создания ссылок. В зависимости от атрибутов устанавливает ссылку или якорь. |
| <b> | Делает шрифт жирным |
| <h1> | Заголовок первого уровня |
| <br> | Перевод на новую строку |
| <hr> | Начертание горизонтальной линии |
| <nav> | Задает навигацию по сайту |
| <ul> | Создание маркированного списка, используется для создания меню. |
| <li> | Элемент списка |
| <img> | Отображение графических изображений на веб-страницах |

|  |  |
| --- | --- |
| <form> | Устанавливает форму на странице для обмена данными между сервером и пользователем. |
| <label> | Связь между меткой и элементом формы. |
| <input> | Обеспечивает взаимодействие пользователем. Служит для создания текстовых полей, различных кнопок, переключателей и флажков. |
| <canvas> | Создает область, в которой при помощи Javascript можно делать анимации, графики и т.д. |
| <select> | Создает выпадающий список |
| <option> | Определяет пункты списка |