Я верстальщик

Веб-верстальщик

Арсений Матыцин

12+

Арсений Матыцин

Я верстальщик. Веб-верстальщик

«ЛитРес: Самиздат»

2018

Матыцин А. О.

Я верстальщик. Веб-верстальщик / А. О. Матыцин — «ЛитРес: Самиздат», 2018

Данный материал предназначен в первую очередь для тех, кто только начал осваивать веб, не знает за какие ниточки потянуть и что сделать, чтобы понять, как это все работает. Не менее полезен материал будет и дизайнерам, которые рисуют макеты и слабо представляют, что с этим потом делает верстальщик. Скорее, он даже очень нужен, так как давно существует проблема взаимопонимания между дизайнерами и верстальщиками. В целом данный материала создавался для повышения компьютерной грамотности среди людей, хоть как-то связанных с версткой.

Содержание

Арсений Матыцин
Я верстальщик. Веб-верстальщик

Введение

Для кого

Данный материал предназначен в первую очередь для тех, кто только начал осваивать веб, не знает за какие ниточки потянуть и что сделать, чтобы понять, как это все работает. Не менее полезен материал будет и дизайнерам, которые рисуют макеты и слабо представляют, что с этим потом делает верстальщик.

Скорее, он даже очень нужен, так как давно существует проблема взаимопонимания между дизайнерами и верстальщиками.

В целом данный материала создавался для повышения компьютерной грамотности среди людей хоть как-то связанных с версткой.

Что такое верстка

Существует несколько видов верстки. Изначально этот термин использовался для обозначения монтажа текста, изображений на макете, который затем переносился на бумагу, и впоследствии, с приходом интернета, версткой стали называть процесс создания гипертекстовой разметки (HTML кода).

Верстка сайта предполагает перенос дизайна макета в формат удобоваримый для браузера. Задача верстки сегодня – создать сайт, который одинаково хорошо будет смотреться на всех устройствах.

Будем откровенны, задача не из легких и, порой, в принципе неосуществима. Все дело в том, что устройства разные, разработчики железа постоянно что-то меняют, создают новое [иначе бы не было прогресса], но верстальщику из-за этого приходится подстраиваться. Не считая того, что технологии и стандарты верстки тоже постоянно дополняются и меняются. О концепциях и подходах поговорим позже.

В чем отличие от типографской верстки

Когда ты работаешь со статическим, аналоговым носителем, таким как бумага, ты знаешь наверняка, как будет выглядеть верстка в конечном счете.

Если ты берешь визитку, то необходимо просто определиться с ее форматом, настроить макет, сверстать и напечатать. На этом процесс можно закончить и переходить к следующему макету.

Как я указал выше – в вебе никогда не можешь быть уверен наверняка, с какого устройства пользователь посмотрит на плоды твоей работы.

Кто такой верстальщик

Термин верстальщик идет из типографии, где представитель профессии набирал полосы в соответствии с созданным макетом. В принципе, в плане веб, ничего не изменилось, просто поменялись инструменты, и верстальщик стал работать по сути только с текстом. Теперь же, вместо бумаги результат работы видят в браузере.

Также верстальщика можно называть более конкретно – верстальщик веб-страниц. Не так давно появился термин пришедший извне – frontend-разработчик.

Могу сказать, это мое личное мнение, но то, что должен уметь фронтедщик и якобы хорошо знать верстальщик, все это не серьезно. Просто front-end хорошо сочетается с back-end и имеет оттенок ориентированности не только на веб, но и на приложения.

Основа профессии верстальщика/фронтендщика – грамотность, знание основ, их правильное использование в разных проектах, от верстки одностраничника – до разработки крупного приложения. А также возможность изучать новые технологии для достижения и преодоления новых высот и сложностей.

Стандарты

Стандарты

Ранее создавалось множество версий HTML. Среди них были XHTML, строгий и переходной режим HTML4. Для обозначения используемого стандарта используется доктайп. Следующим образом:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

Включен строгий режим стандарта HTML 4.01

Сегодня для верстки сайтов актуален только один формат-стандарт – HTML5. И отлично, что к нему пришли. Самым первым его преимуществом является формат записи стандарта в коде:

<!DOCTYPE html>

Включен режим HTML 5.x

Ко второму преимуществу можно отнести семантику. Верстать можно и просто с использованием div-ов. Визуально никакой разницы, вот только в верстке важно не только как это выглядит. Очень важно, чтобы ваш код был семантически корректно сверстан. Это облегчит будущее существование сайта на просторах интернета.

Впрочем, если вы верстаете какое-нибудь веб-приложение, удел которого быть закрытым продуктом, то семантической версткой можно пренебречь.

Почему стоит следовать стандартам

Как я уже указывал ранее, задача верстальщика – обеспечить оптимальное отображение информации на всех устройствах. Стандарты, создаваемые и опекаемые организацией W3C, призваны служить именно этой цели.

Почему я об этом говорю? Потому что можно заставить браузер отображать верстку и без объявления доктайпа. Даже можно не указывать тег html. И большинство устройств спокойно съедят такой код. Но я очень не рекомендую так делать.

Маленький совет

Этим правилом, писать по стандартам, можно пренебречь, если вам необходимо, скажем, отдельно от всего сделать верстку таблицы, которую в дальнейшем вам необходимо вставить в код.

Табличная верстка

Это сильно устаревшая техника верстки, которую я застал, когда обучался дизайну. Тем не менее нельзя обойти стороной то, из чего выросла технология.

Раньше, чтобы расположить элементы в строку рядом друг-с-другом применялись таблицы. Вот пример:

```
<!DOCTYPE html>
<html lang="ru">
    <head>
        <meta charset="UTF-8">
        <title>Example</title>
    </head>
    <body>
        <table width="300" border>
        <tr>
            <td>1</td>
            <td>2</td>
            <td>3</td>
        </tr>
        </table>
    </body>
</html>
```

Пример кода табличной верстки

Результат обработки кода браузером Edge

Сложность такого метода заключалась в том, что невозможно было обойтись без пространственного понимания сетки. Если вы создавали сетку 5 на 5, то вынуждены были следовать ее логике, чтобы разместить информацию внутри. Или создать еще одну таблицу внутри.

Принципы табличной верстки хорошо демонстрируются на примере обычной таблицы в редакторе.

Тут в строке tr вложить элемент столбца td, который занимает место 5 ячеек.

Эта часть таблицы объединяет в себе 2 ячейки по горизонтали и 4 ячейки по вертикали А справа →,

в ячейку вложена таблица 5 на 5, которая по умолчанию отбивается от границ ячейки отступами.

Если возникала необходимость добавить одну ячейку во второй строке, то она выбивалась из ряда. Чтобы этого избежать приходилось либо в каком-нибудь столбце указывать объединение по вертикали на одну ячейку больше, либо переделывать сетку на 6 в ширину, или же создавать вложенную таблицу.

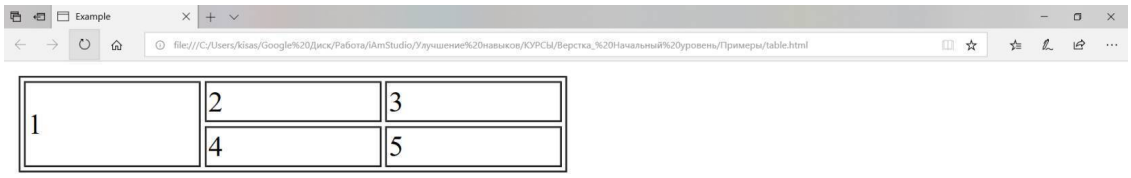
Такие манипуляции по склейке ячеек производятся с помощью атрибутов:

colspan – объединение ячеек по ширине

rowspan – объединение ячеек по высоте

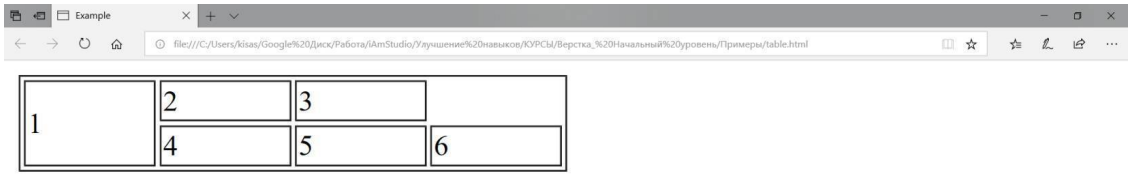
Пример:

```
<table width="300" border="1">
    <tr>
        <td rowspan="2">1</td>
        <td>2</td>
        <td>3</td>
    </tr>
    <tr>
        <td>4</td>
        <td>5</td>
    </tr>
</table>
```



Результат обработки в браузере

Формально это таблица з на 2, но так как я указал первой ячейке расшириться на 2, то было бы ошибкой во второй строке написать такое же количество ячеек td, как и в первой. Результат был бы тогда такой:



Результат ошибочного кода в браузере

Эти атрибуты легко запомнить, если разбить их на составляющие и знать перевод. Например span, который есть в обоих словах – это пролет, объединение, диапазон. В то же время row – это строка, а col (column) – это колонка.

Да и в целом это неплохая практика понимать, что ты пишешь. Необходимо знать английский хотя бы на уровне терминов и тегов, или навыков использования переводчика.

В целом, если вспомнить принцип дизайнера и верстальщика, то это можно воспринимать не как ошибку, а как фичу.

С приходом адаптивного дизайна и верстки над существованием таблиц навис большой вопрос. Так как все попытки отразить сколько-нибудь большую таблицу на экране мобильного телефона оказалось не так просто.

Исторический пример

Таблицы активно использовались в частности для таких вещей, как склейка изображений, создание навигации.

Склейка изображений это вообще обалденная штука, в свое время я увлекался таким сбором, разбором, но лучше всего был фактор оптимизации кода. Изображение, допустим, тигра можно было нарезать и во всех ячейках кроме глаза использовать сжатый формат jpg. А в случае с глазом использования gif-анимацию. Можно сделать gif высокого, насколько это возможно, качества и как результат смотреться будет эффектно и весить не так много, если бы весь тигр был гифкой. (Вес изображения при этом сильно зависит от дизайнера)

Впрочем казалось бы, что такого в табличной верстке и на кой оно вам надо. Ответ прост – она не умерла, а перешла в иное качественное состояние. Сегодня табличная верстка активно используется для создания html-рассылок. В данном случае, в отличие от разработчиков браузеров, разработчики софта для работы с письмами так и не пришли к единому консенсусу и 100% у вас на каждом отдельном устройстве откроются совершенно разные виды на вашу верстку. Порой ужасные.

Чтобы этого избежать там используется не стандарт HTML5, а строгий старый режим HTML4. К нему, вдобавок, идут набором ныне морально устаревшие теги типа center, и приходится использовать ухищрения, которые ни один верстальщик сайтов никогда в здравом уме использовать не будет.

Блочная верстка

Блочная верстка

В отличие от табличной, блочная верстка позволяет набрать необходимое количество элементов страницы без условностей и требований первого метода. Да и в целом этот тип можно отнести к инновационным. Наряду с блочной версткой стали активно использовать внешние файлы css, которые призваны были описывать внешний вид div-ов. Притом эта тенденция (выносить данные) дошла до того, что инлайновые стили стали неким моветоном. Лично я с этим конвейерным и неразумным подходом не согласен.

Разумно использовать внешние стили для повторяющихся элементов, для классов, для медиа запросов и во многих других случаях. Но если в конкретно этом span надо прописать *color:red*, зачем создавать отдельный класс или id и писать под него стили!?

Float:left

Самое частое свойство, которое мне приходилось использовать во время верстки сайта по блочному методу. Так как по-умолчанию div занимает всю область по ширине, куда он вписан, то, чтобы выстроить в линию несколько блоков необходимо дать им размер – ширину. Но тогда они просто выстроятся один под одним с указанными размерами. Чтобы выполнить «обтекание» нужно использовать свойство *float*.

Float:left заставляет элемент отобразиться таким образом, чтобы предыдущий был по левую сторону от него.

Небольшой хак

Существует иной способ решить этот вопрос, переноса блоков. Достаточно сменить ему свойство *display* с дефолтного *block* на *inline-block*. Тогда браузер будет воспринимать данный *div*, как символ в предложении.

Данная тема постепенно приводит нас к каскадным стилям. О них я расскажу после того, как опишу работу с тегами.

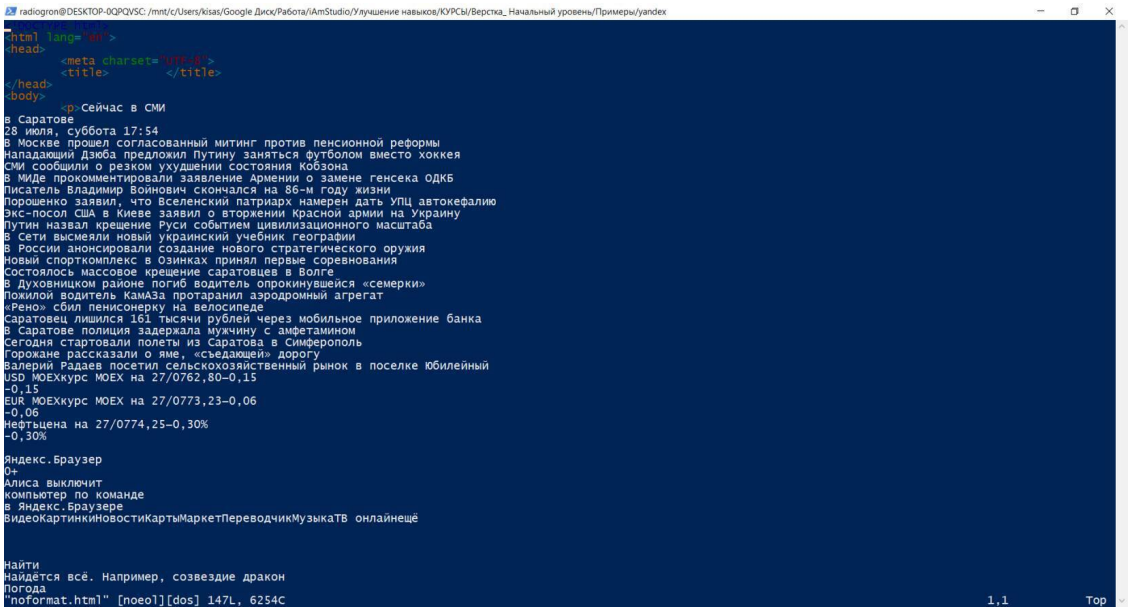
Редакторы кода

Выбор редактора кода (IDE)

Прежде чем мы приступим к практике, вам необходимо выбрать редактор. Иными словами IDE. IDE – это интегрированная среда разработки. Иными словами – среда в которой разработчик (тут верстальщик может почувствовать гордость, что он на одной ступени с программистами) пишет код и ему в этом помогают всякие утилиты, виджеты и прочая вспомогательная ерунда. Если у вас подсвечивается код в соответствии с синтаксисом выбранного языка – это IDE.

Но будем откровенны. Изначально мы работали в редакторах исходного кода. Мы это верстальщики, программисты, все, кто работал с кодом. Таким редактором долгое время был, и наверное остался блокнот, который поставлялся и поставляется до сих пор с операционной системой Windows. Со средой разработки в Linux не все так просто. Там есть Vim – редактор, позволяющий обрабатывать содержимое файлов прямо в режиме консоли.

Если открыть в Vim файл формата html с примером кода, описываемого в главе Теги, то можно увидеть вот это:



Редактор Vim в консоли Windows PowerShell с установленной подсистемой Ubuntu

Никто не знает, как выйти из вима
Сей факт породил тонны шуток и мемов в духе:

На самом деле все просто, достаточно ввести двоеточие и символ q в английской раскладке. Никакой интриги.

Язык не поворачивается назвать Vim IDE, так что я скорее отнесу его к редактору исходного кода. К слову, если блокнотом сейчас никто не пользуется, то Vim более чем здравствует. Если вы начнете использовать Linux систему, или, как я Windows, с установленной подсистемой Ubuntu и будете использовать систему хранения версий, то скорее всего столкнетесь с этим редактором.

Минутка истории

Самым первым редактором исходного кода можно считать что угодно, вплоть до гальки на песке, которая описывает алгоритм. Но более близкое к нашему времени можно считать дырокол (вспоминаем перфокарты) и лист бумаги\тетради. Именно с использованием обычной аналоговой бумаги мои родители писали и сдавали программы во время обучения в институте.

Но вернемся к вопросу выбора. Выбор редактора (возьмем в качестве этого термина объединение IDE и редактор исходного кода) очень важен как для верстальщика, так и для программиста.

Я уверен, в первую очередь он должен быть удобным. Для меня удобство – это качественные шрифты и цветовая схема, которая не вырывает мне глаза. Лучшим редактором в плане цветового решения для меня был и остается Notepad++. Там информация отображается на белом фоне, по-умолчанию используется хорошо читаемый шрифт. Но он сильно пасует перед продвинутыми редакторам в плане возможностей, хотя я достаточно часто к нему прибегаю, когда необходимо решить сложные вопросы легким путем.

Полноценная IDE, на которой я остановился на момент написания этой книги – Brackets. Это достаточно удобный редактор для работы над проектами. Когда много файлов, и нужно все их видеть не отходя от кассы, т.е. от редактора. В тот же notepad такую возможность ввели где-то в середине 2018-го года.

Иной раз и при особых задачах пасует и Brackets. Тогда на помощь может прийти WEBStorm. WEBStorm создала компания JetBrains и у них есть еще много редакторов под разные нужды. В основном эти нужды выражены языками.

В целом можно выделить следующие редакторы:

Notepad – Встроенный редактор Windows. Никакой подсветки, открывается очень быстро, позволяет обнулять стили.

Notepad++ (NPP) – Грубо говоря, расширенный редактор Notepad, позволяет полноценно редактировать код, и мы все когда-то с ним успели поработать.

Sublime – Редактор из Linux, настолько знаменит, что о нем не все знают. В целом, удобная альтернатива Notepad++. Или наоборот!? Рекомендую ознакомиться.

Dreamweaver – Хоррор из прошлого. Некогда запомнился мне инновационной подачей «пишу и вижу», что позволяло сразу видеть, как код написан. Вот только показывал он его както по своему, в своей реальности. Лет 7 назад посоветовал бы не связываться с ним, а что он представляет сейчас не знаю.

Brackets – Мощный и удобный редактор для верстальщика. Является неким средним явлением между notepad++ и phpstorm. Можно ставить дополнения и вообще достаточно гибок в настройке.

Visual Studio Code – Я когда-то учил язык С и тестировал знания в редакторе Visual Studio. Так вот с тем, к чему я привык, там эта «облегченная версия» не имеет ничего общего. Удобный расширяемый редактор, который я поставлю на одно место с brackets. По умолчанию умеет работать с Git.

PHPStorm – Ориентирован изначально на работу php-программиста, но отлично подходит и верстальщику. Удобная работа с проектами, некоторые интересные фичи типа редактирования открывающего и закрывающего тега. Но не умеет редактировать много строк сразу, да и тяжелый, как мастодонт. Будет грузится едва ли меньше, чем фотошоп на вашем компьютере. Рекомендую только в том случае, если вы работаете в паре с php-программистом и вам не обойтись без единовременного редактирования кода.

WebStorm – Редактор с упором на javascript проекты. Должен быть удобен для возведения конструкций с использованием фреймворков/библиотек типа Vue, Angular, React.

PyCharm – Это уже третий редактор от JetBrains, и он вроде не по теме, но только до того момента, как вы не начинаете работать с, допустим, фреймворком Django. Это уже относится к продвинутому уровню, так что не будем долго останавливаться.

Остальными критериями для выбора редактора я считаю расширяемость и в принципе возможности, о которых не думаешь в первую очередь. Например в sublime и brackets можно редактировать несколько строк одновременно.

Ни в коем случае не используйте для верстки визивиги!

WYSIWYG – это визуальный редактор, работающий по принципу «что видишь, то и получишь». Такой редактор уместен во многих других направлениях, например для сбора презентации в Point, или при составлении текста в Google Docs, но только не в верстке. Уже придумали, но еще не создали такой визивиг, который мог бы вам помочь в жизни.

К широко распространенному визивигу можно отнести редактор в WordPress. Представьте ситуацию: вы написали шикарный код, читаемый, с отступами, как надо, вам необходимо разместить его на страницу, идете в редактор вордпресса, вставляете, сохраняете. Потом какой-то очень умный человек переключается в текстовый режим и обратно. Все, код поехал.

Настройка IDE

Эти действия я буду описывать на личном примере с использованием редактора Brackets.

Одно из первых действий, что я делаю при установке чистой программы – качаю тему. Часто скин под Notepad++, иногда темную тему, иногда вообще что-то новое.

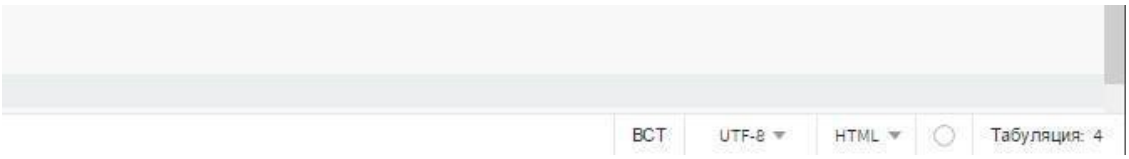
Далее ставлю Emmet – данное дополнение позволяет использовать краткую форму записи тегов.

Сокращенный код в Emmet: *div.classname#ddd>label+input*

Преобразованный в HTML: * <label for=""></label><input type="text"> *

И в обязательном порядке устанавливаю плагин Show Whitespace в исполнении Dennis Kehrig (В магазине несколько таких плагинов и этот, на мой взгляд, самый адекватный в исполнении). Этот плагин позволяет подсвечивать пробелы и табуляцию – таким образом круче управлять своим кодом.

После установки плагинов я устанавливаю настройку отступов в формат табуляции со значением в 4.



Настройка находится в нижнем правом углу редактора Brackets.

Лично мне легче работать с кодом именно таким образом. Так отступы всегда формируют читаемый код, что особенно важно при работе, например с препроцессором SASS (не путать с SCSS). Другой популярный метод – пробелы в 2 единицы.

Настраивать и выбирать вам, но так как вам либо работать в команде, либо отдавать свой код заказчику, с которым работать потом другому верстальщику или программисту, придерживайтесь культуре написания кода и делайте код читаемым хотя бы с помощью адекватных отступов.

Это в общем все. Для комфортного написания кода стоит просто настроить свое пространство для этого самого комфорта. К теме также можно отнести и выбор хорошего кресла, но это уже совсем другая история.

Не забывайте, кроме фактора удобства и популярности существуют требования языка. Например YAML требует запись в 2 пробела.

Когда спасает Notepad++

Автозамена в 20+ файлах – это просто волшебный редактор, когда необходимо провести автозамену в тонне файлов, любой другой навороченный редактор заглохнет и начнет чихать.

Редактирование отдельно-стоящих файлов. Незачем грузить полноценную IDE для редактирования одного файла, например при использовании FTP-клиента. На всех компьютерах, на которых я работал, notepad загружается крайне быстро. В то время, как тот же Brackets приходится ждать. Чего уж говорить про PHPStorm.

Необходимо обнулить стили. Если я беру какой-то текст с сайта, то вместе с ним тянутся и стили, заголовки и иное форматирование. И не всегда связка Ctrl+Shift+V спасает. В таком случае я запускаю Notepad ++.

Палочка-выручалочка

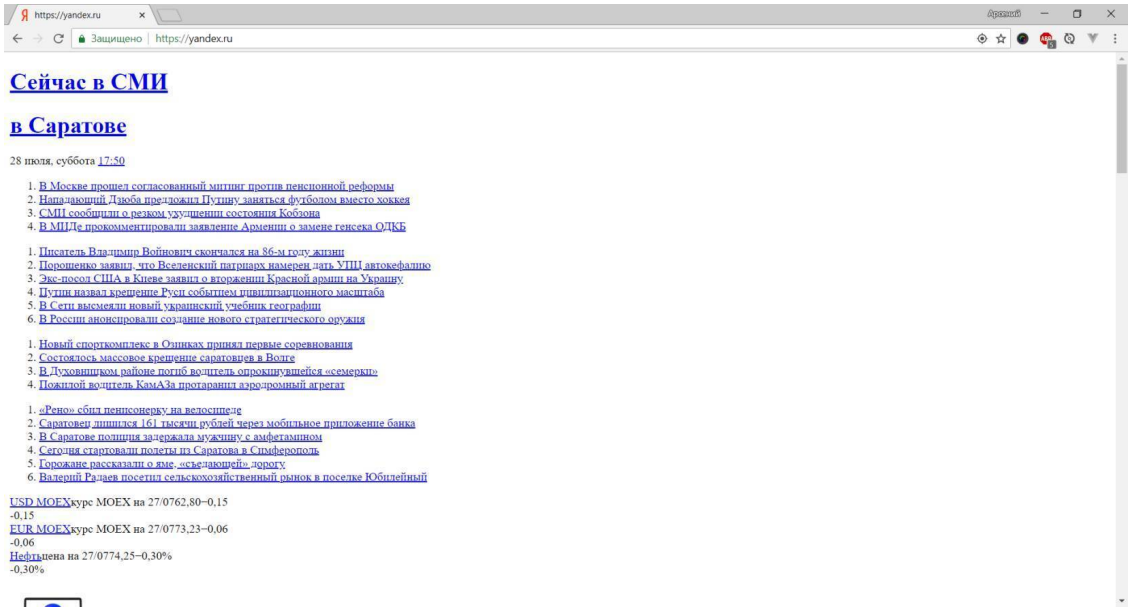
Если вы открыли ту самую тонну файлов в notepad и пытаетесь их закрыть, в меню Файл есть волшебная кнопка «Закрыть все».

HTML

Теги

Тег HTML – это уже и есть HTML. Теги используются для того, чтобы обозначить начало и конец элементов на будущей странице. Использование разметки изначально необходимо для того, чтобы облегчить читаемость.

Для примера я решил взять главную страницу Яндекса:

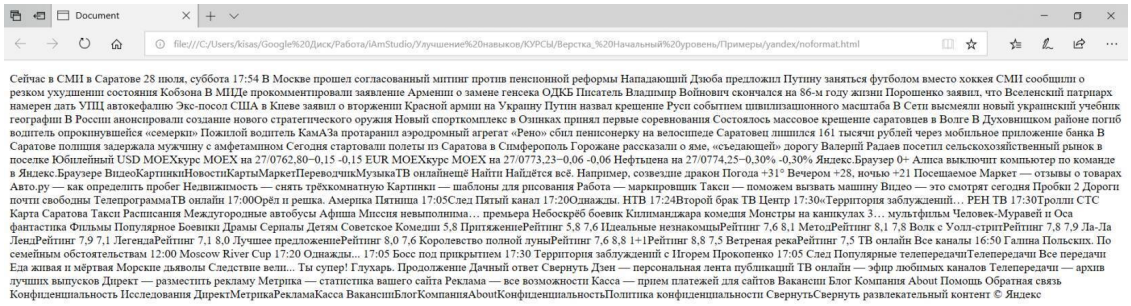


Главная Яндекса без стилей.

Для наглядности я удалил тег head вместе со всем содержимым. Внутри были стили и скрипты. И много всякой полезной ерунды, без которой мы можем видеть отформатированный, де-факто сверстанный текст. В данном случае текст и информацию вывел сервер, так что если вы проделаете то же самое у себя на компьютере, то увидите другой текст на главной странице.

Важно!: кроме head я удалил шапку и содержимое, которое выводило мою личную информацию.

Вот то же самое, только без участия тегов. Все содержимое я скопировал и вставил в тег p.



Главная Яндекса без верстки вообще.

Внимательный читатель обратит внимание, что 2 страницы браузера не сильно похожи одна на другую. Таким образом можно утверждать, что верстка – это перевод текста [читай контента] в другое качественное состояние. Точно так же, как это делают в типографии в классическом исполнении верстки.

В данном примере используются теги заголовков (H1), параграфа (p) и нумерованного списка (ol).

Обратите внимание:

Все теги форматирующие текст по-умолчанию имеют свое оформление. Например у списка есть отступ слева + нумерация, заголовки отличительно больше, чем основной текст, ссылки подсвечиваются и подчеркиваются и т.д. Первое желание начинающего дизайнера – поменять все стили на какие-нибудь «крутые». Ссылки сделать в цвет радуги, заголовкам дать новые отступы или учудить еще какую-нибудь ересь. Окститесь, ребята, в большинстве случаев не стоит так делать. Для того и создавались стандарты, чтобы пользователь мог понять, что ссылка это ссылка, заголовок это заголовок, который относится именно к вот этому абзацу. В этом деле существует много исключений, но работая с текстом всегда нужно взвесить все за и против, прежде, чем необдуманно броситься делать все «красивенько».

К слову, теги используют не только для верстки сайтов, но и для оформления текста, например в вики-разметке раньше существовали (сейчас скорее всего тоже где-то используются) bb-коды, созданные для оформления текста на форумах.

Сегодня, с ростом качественных редакторов типа zen-editor, мы все реже видим и используем теги в их первозданном виде при обычном редактировании текста в веб-интерфейсах. Но не обманывайтесь, под капотом редактора происходит оборачивание текста в теги.

HTML дескрипторы

Специально для написания этой книги, кроме знаний, которые я получил опытным путем в разработке разного рода проектов, я активно ищу теоретическую информацию. Таким образом я узнал, что слово тег – это сленг, который используется вместо термина дескриптор.

Еще одним важным примером будет XML разметка. Она вам пригодится при разработке сайта на продвинутом уровне, например при создании священного файла sitemap.xml. Священный он потому что при его отсутствии на сайте поисковики будут ставить вам палки в колеса.

Идя дальше можно говорить про разработку для мобильных платформ. Там вам тоже пригодится знание и понимание тегов.

Как применять теги

Существует 2 вида тегов. Те, которые надо закрывать и те, которые закрыты сами. Отличить их просто, те, что надо закрывать одинаковые, идут парно и последний, т.е. второй имеет одну отличительную черту. Косую. Закрываемые теги выглядят так:

<h1>Между тегами содержимое</h1>

Открывающий тег / Закрывающий тег с косой чертой

В то же время есть самозакрытые теги. Их также называют одиночными, пустыми. И если обычный тег надо открыть и закрыть, то одиночный необходимо просто разместить.

**
**

Вот и все

Открывающий тег, он же закрывающий

На фоне парных тегов количество одиночных весьма ограничено, и их легко запомнить. В разборе значений и предназначений тегов я разделю парные и одиночные теги на 2 группы.

С приходом стандарта HTML5 изменилась форма записи в одиночных тегах. Раньше требовалось указывать косую черту перед закрывающей скобкой тега вот так –
, а сейчас нет. Это долгое время мешало мне, так как я привык писать по старым стандартам, и валидатор W3C писал мне предупреждения.

Теги можно спокойно, с оглядкой всего-лишь на правила, вкладывать друг-в-друга. Самым простым примером должна стать ссылка внутри абзаца:

** Текст, внутри которого находится ссылка **

Пример вложенных тегов

Кроме вида тегов к ним применимы атрибуты. Атрибут тега – это свойство, которое может иметь значение присвоенное, или просто свойство, которое не имеет значения, но определяет, как будет себя вести на странице тот или иной тег.

Основные теги

Тегов существует великое множество и нет никакого смысла их все запоминать. Для постоянной работы вам потребуется несколько основных: div, span, p, a, img.

Также обращаю ваше внимание, что данный список не является конечным, так как для повышения семантики и для удобства верстки их дополняют, а многие выходят из обихода. Подробный список можно найти на сайте w3schools.com/tags/.

Парные теги

Самые первые парные теги, которые вам нужно запомнить – те, которые формируют каркас будущей страницы:

html – Данный тег определяет страницу целиком. В него оборачивается все содержимое страницы.

head – Данный тег предназначен для «невидимой» части страницы. В него помещается заголовок страницы (который отображается на вкладке браузера), определение правил и почти всегда ссылки на зависимости.

body – Само тело [body] содержит видимую часть страницы, грубо говоря отформатированный текст.

Тут же следует сразу показать, как их использовать:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Пример</title>
    </head>
    <body>
        Содержимое
    </body>
</html>
```

Пример использования парных тегов

Первым делом мы указываем доктайп, затем открываем html тег, в котором размещаем head внутри которого можно разместить title. Закрываем и тут же открываем body, в котором уже пишем весь код. В конце теги закрываются в порядке вложенности.

Маленький хак от Emmet

Возвращаясь к теме редакторов IDE можно упростить себе жизнь с плагином Emmet. Он позволяет вызвать шаблон-заготовку страницы html посредством следующих действий:

Пишем знак восклицания – !

Нажимаем таб

Получаем кусок кода, готовый к эксплуатации

Я такое часто применяю в редакторе Brackets.

Далее давайте разберем остальные теги.

** ** – Самый популярный тег, который используется для формирования страницы. Является блочным по-умолчанию. Т.е. растягивается по всей ширине занимаемого пространства и сразу понимает присвоенным значения. Например, если тегу а указать размеры, то он их не сможет понять до тех пор, пока вы в стилях не укажите ему другой display, например inline-block. Div это умеет делать сразу.

<h1>–<h6> – Тег заголовка [англ. heading – отсюда сокращение h], варьируется от уровня заголовка. H1 – самый важный и, соответственно, самый большой. А h6 уже меньше размера основного текста.

Заголовки жизненно необходимы для разделения текста на части, чтобы упростить чтение материала. Они также используются поисковыми роботами для определения содержимого страницы. Впрочем, и человек в первую очередь пробегается глазами по заголовкам и только потом читает сам текст (если читает).

Заголовок является блочным элементом.

На странице может быть не больше одного тега h1!

<p> – Данный тег используется для оформления текста в формате абзаца [англ. paragraph]. Чтобы абзац выглядел, как должен, а именно: быть единым целым, единицей текста, и иметь перенос текста в конце. Текст без обертки в параграф превращается в однородное полотно, которое трудно читать. Параграф не может быть обернут в тег ссылки.

Конец ознакомительного фрагмента.