Сергей Куринный

# Mozilla

## Введение в веб программирование

# Mozilla. Введение в веб программирование

**Установка базового программного обеспечения** **3**
Какие инструменты используют профессионалы? 3
Какие инструменты на самом деле нужны мне прямо сейчас? 4

**Каким должен быть ваш веб-сайт?** **5**
Перво-наперво: планирование 5
Сделайте набросок вашего дизайна 6
Выберите свои активы 7

**Работа с файлами** **12**
Где веб-сайт должен располагаться на компьютере? 12
Небольшое отступление о регистре и пробелах 13
Какую структуру должен иметь ваш веб-сайт? 13
Путь к файлам 14
Что должно быть сделано? 17

**Основы HTML** **17**
Что такое HTML на самом деле? 17
Изображения 22
Разметка текста 23
Ссылки 25

**Основы CSS** **28**
Что такое CSS на самом деле? 28
Шрифты и текст 33
Блоки, блоки и ещё раз блоки 35

**Основы JavaScript** **41**
Что такое JavaScript на самом деле? 41

Пример "hello world" 42
Ускоренный курс по основам языка 44
Прокачаем пример нашего веб-сайта 51

**Обзор инструментов разработки в браузерах 56**
Как открыть инструменты веб-разработчика в браузере? 56
Inspector: DOM обозреватель и CSS редактор 59
Консоль JavaScript 62

**Что дальше? 64**

JavaScript — это язык программирования, который дает возможность реализовывать сложное поведение веб-страницы. Каждый раз, когда вы видите веб-страницу, она не только отображает статическое содержимое, но и делает большее - своевременно отображает обновление контента, выводит интерактивные карты, 2D/3D анимацию, прокручивает видео и т.д. - будьте уверены, здесь не обошлось без JavaScript.

Считается, что JavaScript сложнее изучить, чем связанные с ним технологии, наподобие HTML и CSS. Поэтому, перед изучением JavaScript, настоятельно рекомендуем сначала ознакомиться хотя бы с этими двумя технологиями.

Необходимо много работать, чтобы создать профессиональный веб-сайт, так что, если вы новичок в веб-разработке, мы рекомендуем начать с малого. Вы не будете создавать свой Facebook прямо сейчас, однако создать свой личный простой веб-сайт в Интернете не так уж и сложно, так что мы начнем с этого.

# Установка базового программного обеспечения

## Какие инструменты используют профессионалы?

- КОМПЬЮТЕР. Может быть, это звучит очевидно для некоторых людей, но некоторые из вас читают эту статью с телефона или библиотечного компьютера. Для серьезной веб-разработки, лучше приобрести настольный компьютер (Windows, Mac или Linux).

- ТЕКСТОВЫЙ РЕДАКТОР, ЧТОБЫ ПИСАТЬ КОД. Это может быть бесплатный текстовый редактор, например:

Notepad++ (http://notepad-plus-plus.org/),
Brackets (http://brackets.io/),
Atom (https://atom.io/),
Visual Studio Code (https://code.visualstudio.com/).

- ВЕБ-БРАУЗЕРЫ, ДЛЯ ТЕСТИРОВАНИЯ КОДА. В настоящее время наиболее часто используемые браузеры это
Firefox (https://www.mozilla.org/ru/firefox/new/),
Chrome (https://www.google.com/chrome/browser/),
Opera (http://www.opera.com/).
Вы также должны тестировать ваш сайт на то, как он работает на мобильных устройствах и на любых старых браузерах, которые ваша целевая аудитория может все ещё широко использовать (например, IE 6-8).

- ГРАФИЧЕСКИЙ РЕДАКТОР, ЧТОБЫ СОЗДАВАТЬ ИЗОБРАЖЕНИЯ для ваших веб-страниц.
The Gimp (http://www.gimp.org/)

Paint.NET (http://www.getpaint.net/)

- СИСТЕМА КОНТРОЛЯ ВЕРСИЙ, чтобы сотрудничать над проектом с командой, делиться кодом и вкладами, и избегать редакционных конфликтов. Сейчас Git (http://git-scm.com/) является наиболее популярным инструментом контроля версий, и репозиторий кода Github (https://github.com/), основанный на Git, также является очень популярным.

- FTP ПРОГРАММА, чтобы загружать веб-страницы на сервер для публичного просмотра.

- СИСТЕМА АВТОМАТИЗАЦИИ, такая как Grunt (http://gruntjs.com/) или Gulp (http://gulpjs.com/), для автоматического выполнения повторяющихся задач, например, минимизации кода и запуска тестов.

- ШАБЛОНЫ, БИБЛИОТЕКИ, ФРЕЙМВОРКИ и т. д., чтобы ускорить написание общей функциональности.

## Какие инструменты на самом деле нужны мне прямо сейчас?

Этот список выглядит страшновато, но, к счастью, вы можете начать веб-разработку, не зная ничего о большинстве из того, что в нем написано. В этой статье мы установим базовый минимум - текстовый редактор и некоторые современные веб-браузеры.

УСТАНОВКА ТЕКСТОВОГО РЕДАКТОРА
У вас, наверное, уже есть базовый текстовый редактор на вашем компьютере. По умолчанию Windows включает БЛОКНОТ и Mac OS X поставляется с TEXTEDIT. Linux дистрибутивы варьируются; Ubuntu поставляется с GEDIT по умолчанию.

Для веб-разработки вам, вероятно, понадобится больше, чем могут предложить Блокнот или TextEdit. Мы рекомендуем начать с Visual Studio Code. Она бесплатна, кроссплатформенна и более функциональна, чем Блокнот и TextEdit.

УСТАНОВКА СОВРЕМЕННЫХ ВЕБ-БРАУЗЕРОВ
На данный момент, мы установим только пару настольных веб-браузеров, чтобы проверять наш код. Выберите вашу операционную систему ниже и нажмите на соответствующие ссылки для загрузки установочных программ ваших любимых браузеров:
Linux: Firefox, Chrome, Opera.
Windows: Firefox, Chrome, Opera.
Mac: Firefox, Chrome, Opera, Safari.
Firefox (https://www.mozilla.org/ru/firefox/new/),
Chrome (https://www.google.com/chrome/browser/),
Opera (http://www.opera.com/).

Прежде, чем идти дальше, вам следует установить, по крайней мере, два из этих браузеров, чтобы использовать их для тестирования.

# Каким должен быть ваш веб-сайт?

Обдумайте план и дизайн веб-сайта, прежде чем приступить к написанию кода, в том числе:

- Какую информацию будет содержать мой веб-сайт?
- Какие шрифты и цвета я хочу использовать?
- Что будет делать мой сайт?

## Перво-наперво: планирование

Перед тем как делать что-то, вам нужны идеи. Что ваш веб-сайт должен фактически делать? По существу, ваш веб-сайт может делать все, что угодно, но для вашей первой попытки, вы должны придерживаться простых вещей. Мы начнем с создания простой веб-страницы, содержащую заголовок, изображение и несколько

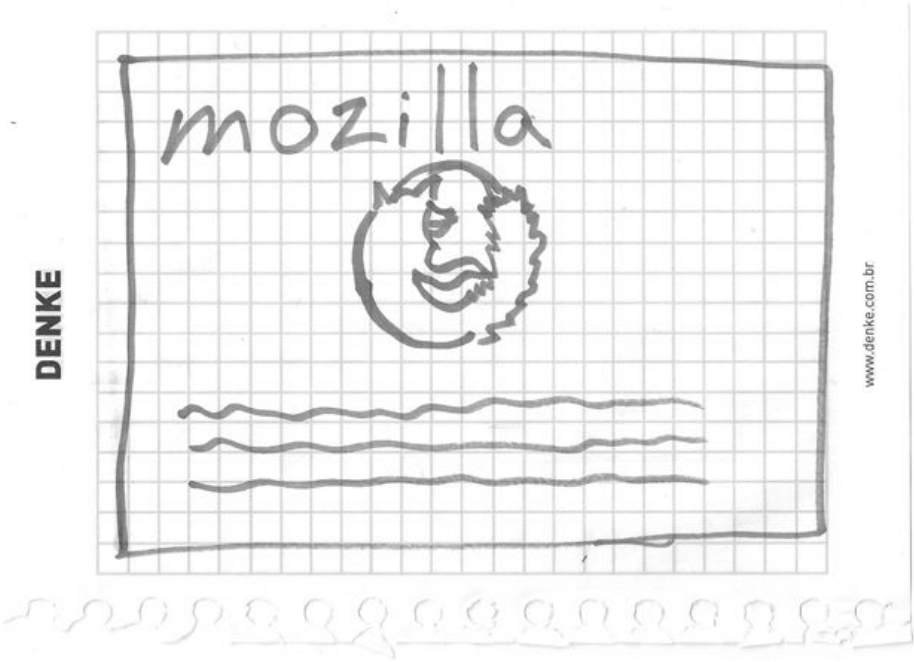
абзацев. Для начала, вам будет нужно ответить на следующие вопросы:

1. О чем ваш веб-сайт? Вам нравятся собаки, Нью-Йорк или Pacman?
2. Какую информацию вы предоставляете о предмете? Напишите заголовок и несколько абзацев, и подумайте над изображениями, которые вы хотите показать на своей странице.
3. Как будет выглядеть ваш веб-сайт, в простых терминах высокого уровня. Какой цвет фона? Какой вид шрифта будет уместен: деловой, мультяшный, жирный и кричащий или тонкий?

Комплексные проекты нуждаются в детализированных руководствах, которые включают все детали цветов, шрифтов, расстояния между элементами на странице, соответствующий стиль письма и так далее. Их иногда называют руководствами по проектированию или бренд-бук.

## Сделайте набросок вашего дизайна

Затем, возьмите ручку и бумагу и сделайте примерный набросок того, как вы хотите, чтобы выглядел ваш сайт. Для вашей первой веб-страницы должен получиться небольшой набросок, и вы должны взять это в привычку. Это действительно помогает, и вам не нужно быть Ван Гогом!




Даже в реальных, сложных веб-сайтах, команда разработчиков обычно начинает с наброска на бумаге и потом строит цифровые макеты используя графические редакторы или веб-технологии. Веб-команда часто включает в себя пару графических дизайнеров и дизайнера с опытом взаимодействия (user-experience (UX) designer). Графические дизайнеры, очевидно, работают вместе над визуализацией веб-сайта. UX дизайнеры играют более абстрактную роль, обращаясь к тому как пользователи будут пользоваться и взаимодействовать с веб-сайтом.

## Выберите свои активы

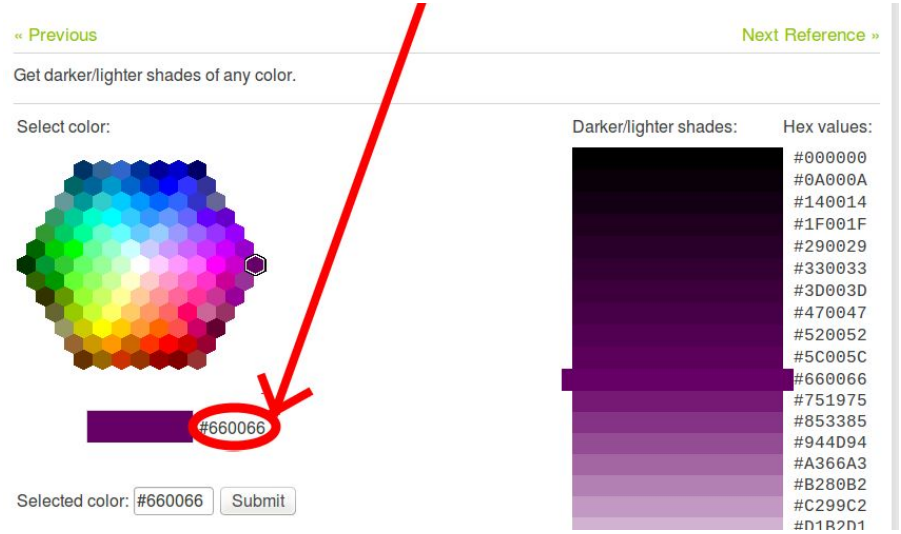
На данном этапе неплохо начать собирать воедино весь контент, который в конечном итоге будет располагаться на вашей веб-странице.

ТЕКСТ

У вас должен быть текст, разбитый на заголовки и параграфы. Придерживайтесь этого правила.

ЦВЕТОВАЯ СХЕМА

Чтобы выбрать цвет, перейдите в [инструмент выбора цвета] и выберите цвет, который вам нравится. Когда вы щёлкните по цвету, вы увидите странный код из шести цифр, например, #660066. Это называется шестнадцатеричный код (hex(adecimal) code) и он представляет ваш цвет. Скопируйте это код куда-нибудь прямо сейчас.



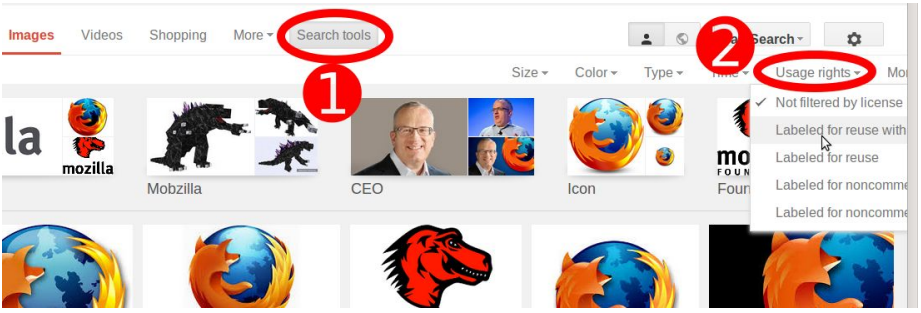

ИЗОБРАЖЕНИЯ

Чтобы выбрать изображение, перейдите в Google Картинки (https://www.google.com/imghp?gws_rd=ssl) и найдите что-нибудь подходящее.

1. Когда вы найдете нужное изображение, которое хотели, щёлкните по изображению.
2. Нажмите кнопку В полном размере (View image).
3. На следующей странице, правым щелчком мыши на изображении (Ctrl + клик на Mac), выберите Сохранить изображение как... (Save Image As...), и укажите место для хранения вашего изображения. В качестве альтернативы,

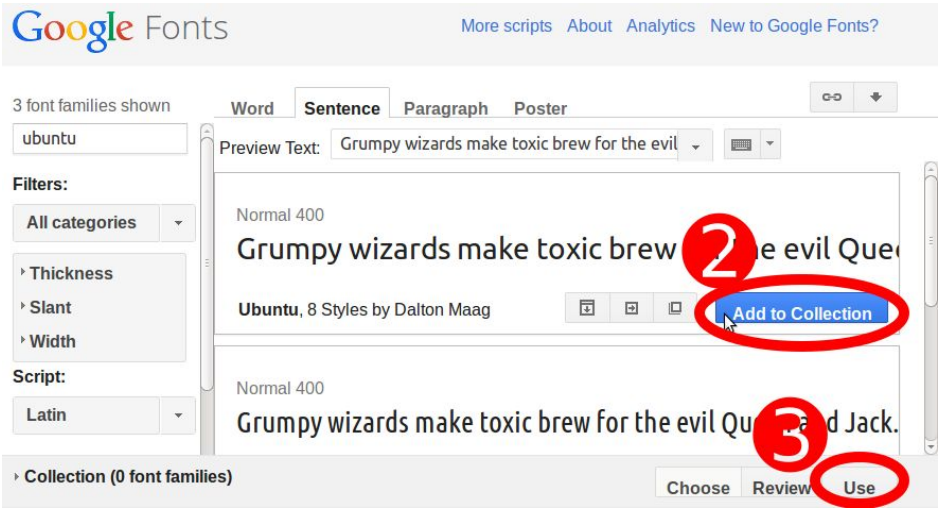
скопируйте адрес изображения из адресной строки браузера для последующего использования.


Большинство изображений в Интернете, использованных в Google Картинках имеют авторские права. Для снижения вероятности нарушения авторских прав, используйте фильтр лицензии Google. Для этого: 1) кликните на Инструменты поиска (Search tools), затем на 2) Права на использование (Usage rights):




ЧТОБЫ ВЫБРАТЬ ШРИФТ:

1. Перейдите на Google Fonts (http://www.google.com/fonts) и прокрутите список вниз, пока не найдете тот шрифт, который вам понравится. Вы также можете использовать элементы управления слева для дальнейшей фильтрации результатов.
2. Щёлкните по кнопке Add to collection рядом со шрифтом, который вы хотите выбрать.
3. Щёлкните по кнопке Use на панели в нижней части страницы.
4. На следующей странице прокрутите вниз к разделам 3 и 4 и скопируйте строки кода Google для последующего сохранения в вашем текстовом редакторе.


Google Fonts
More scripts About Analytics New to Google Fonts?
3 font families shown
ubuntu
Filters:
All categories
Thickness
Slant
Width
Script:
Latin
Word Sentence Paragraph Poster
Preview Text: Grumpy wizards make toxic brew for the evil
Normal 400
Grumpy wizards make toxic brew
Ubuntu, 8 Styles by Dalton Maag
Add to Collection
2
Normal 400
Grumpy wizards make toxic brew for the evil Qu
d Jack.
3
Collection (0 font families)
Choose Review Use


Google Fonts - Mozilla Firefox
What should yo...
What should yo...
Google Fonts
www.google.com/fonts#UsePlace:use
Google
Google Fonts
More scripts About Analytics New to Google Fonts?
3. Add this code to your website:
<link href='http://fonts.googleapis.com/css?fa
Collection into your web page, copy the code as the first element in the <head> of your HTML document.
» See an example
4
4. Integrate the fonts into your CSS:
The Google Fonts API will generate the necessary browser-specific CSS to use the fonts. All you need to do is add the font name to your CSS styles. For example:
font-family: 'Ubuntu', sans-serif;
Instructions: Add the font name to your CSS styles just as you'd do normally with any other font.
Example:
h1 { font-family: 'Metrophobic', Arial, serif; font-weight: 400; }
Collection (1 font family)
Choose Review Use

# Работа с файлами

Веб-сайт состоит из множества файлов: текстового контента, кода, стилей, медиа-контента, и так далее. Когда вы создаете веб-сайт, вы должны собрать эти файлы в рациональную структуру на вашем локальном компьютере, убедитесь, что они могут общаться друг с другом, и весь ваш контент выглядит правильно, прежде чем вы, в конечном итоге загрузите их на сервер. Работая с файлами, обсуждайте некоторые вопросы, о которых вы должны быть в курсе, чтобы вы могли рационально настроить файловую структуру для вашего веб-сайта.

## Где веб-сайт должен располагаться на компьютере?

Когда вы работаете на веб-сайте локально на вашем компьютере, вы должны держать все связанные файлы в одной папке, которая отражает файловую структуру опубликованного веб-сайта на сервере. Эта папка может располагаться где угодно, но вы должны положить её туда, где вы сможете легко её найти, может быть, на вашем рабочем столе, в домашней папке или в корне вашего жесткого диска.

1. Выберите место для хранения проектов веб-сайта. Здесь, создайте новую папку с именем web-projects (или аналогичной). Это то место, где будут располагаться все ваши проекты сайтов.
2. Внутри этой первой папки, создайте другую папку для хранения вашего первого веб-сайта. Назовите ее test-site (или как-то более творчески).

## Небольшое отступление о регистре и пробелах

Вы заметите, что в этой статье, мы просим вас называть папки и файлы полностью в нижнем регистре без пробелов. Это потому что:

1. Многие компьютеры, в частности веб-серверы, чувствительны к регистру. Так, например, если вы положили изображение на свой веб-сайт в test-site/MyImage.jpg, а затем в другом файле вы пытаетесь вызвать изображение как test-site/myimage.jpg, он может не сработать.
2. Браузеры, веб-серверы и языки программирования не обрабатывают пробелы последовательно. Например, если вы используете пробелы в имени файла, некоторые системы могут отнестись к имени файла как к двум именам файлов. Некоторые серверы заменяют пробелы в вашем имени файла на "%20" (символьный код для пробелов в URI), нарушая все ваши ссылки. Лучше разделять слова с помощью тире и нижнего подчеркивания: my-file.html или my_file.html.

По этим причинам, лучше всего приобрести привычку писать названия ваших папок и файлов в нижнем регистре и без пробелов, по крайней мере, пока вы не поймете, зачем это нужно. Так вы столкнетесь с меньшим количеством проблем.

## Какую структуру должен иметь ваш веб-сайт?

Далее давайте взглянем на то, какую структуру должен иметь наш тестовый сайт. Наиболее распространенные вещи, которые есть в любом сайте, создаваемом вами: индексный файл HTML, папки, содержащие изображения, файлы стилей и файлы скриптов. Давайте создадим их сейчас:

1. **index.html**: Этот файл обычно содержит контент домашней страницы, то есть текст и изображения, которые люди видят,

когда они впервые попадают на ваш сайт. Используя ваш текстовый редактор, создайте новый файл с именем index.html и сохраните его прямо внутри вашей папки test-site.

2. **Папка images**: Эта папка обычно содержит все изображения, которые вы используете на вашем сайте. Создайте папку с именем images внутри вашей папки test-site.
3. **Папка styles:** Эта папка обычно содержит CSS код, используемый для стилизации вашего контента (например, настройка текста и цвета фона). Создайте папку с именем styles внутри вашей папки test-site.
4. **Папка scripts:** Эта папка обычно содержит весь JavaScript код, используемый для добавления интерактивных функций на вашем сайте (например, кнопки которые загружают данные при клике). Создайте папку с именем scripts внутри вашей папки test-site.

На компьютерах под управлением Windows у вас могут возникнуть проблемы с отображением имен файлов, поскольку у Windows есть надоедливая настройка с названием Скрывать расширения для известных типов файлов, включенная по умолчанию. Обычно вы можете отключить ее, перейдя в проводник, выбрать вариант Свойства папки... и снять флажок Скрывать расширения для зарегистрированных типов файлов, затем щёлкнуть OK. Для получения более точной информации, охватывающей вашу версию Windows, выполните поиск в Интернете.

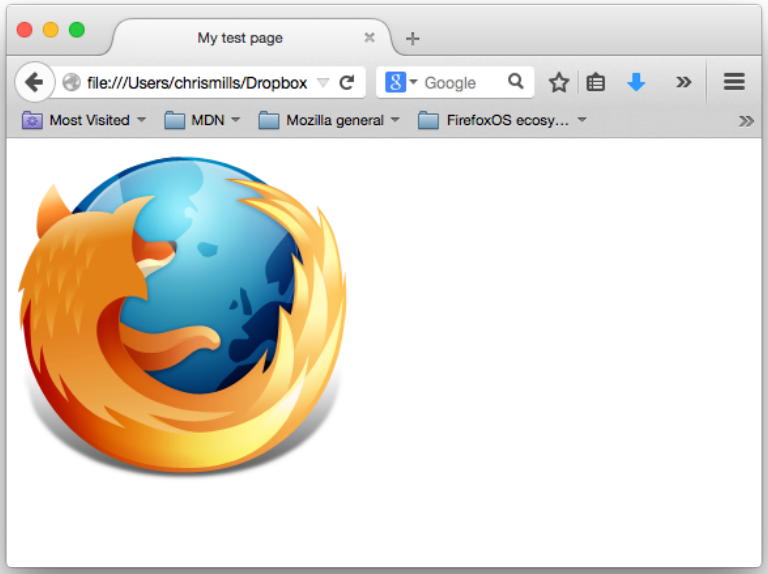
## Путь к файлам

Для того, чтобы файлы общались друг с другом, вы должны указать файлам путь между ними - обычно один файл знает, где находится другой. Чтобы продемонстрировать это, мы вставим немного HTML в наш файл index.html и научим его отображать изображение, которое вы выбрали ранее.

- Скопируйте изображение, которое вы выбрали ранее, в папку images.
- Откройте ваш файл index.html и вставьте следующий код в файл именно в таком виде. Прямо сейчас не беспокойтесь о том, что все это значит - позже в этом руководстве мы рассмотрим структуру более подробно.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <img src="" alt="My test image">
  </body>
</html>
```

- Строка <img src="" alt="My test image"> - это HTML код, который вставляет изображение на страницу. Мы должны сказать HTML, где находится изображение. Изображение находится внутри папки images, которая находится в той же директории что и index.html. Пройдя вниз по файловой структуре от index.html до нашего изображения, получим путь к файлу, который нам нужен. Он выглядит как images/your-image-filename. Например наше изображение, названное firefox-icon.png, имеет такой путь к файлу: images/firefox-icon.png.
- Вставьте путь к файлу в ваш HTML код между двойными кавычками src="".
- Сохраните ваш HTML файл, а затем загрузите его в вашем браузере (двойной щелчок по файлу). Вы должны увидеть вашу новую веб-страницу, отображающую ваше изображение!




Несколько правил о путях к файлам:

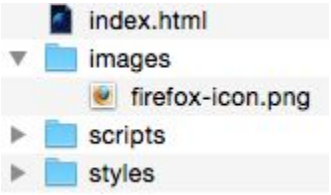
- Для ссылки на целевой файл в той же директории, что и вызывающий HTML файл, просто используйте имя файла, например, my-image.jpg.
- Для ссылки на файл в поддиректории, напишите имя директории в начале пути, плюс косую черту (forward slash, слеш), например: subdirectory/my-image.jpg.
- Для ссылки на целевой файл в директории выше вызывающего HTML файла, напишите две точки. Например, если index.html находится внутри подпапки test-site, а my-image.png - внутри test-site, вы можете обратиться к my-image.png из index.html, используя ../my-image.png.
- Вы можете комбинировать их так, как вам нравится, например ../subdirectory/another-subdirectory/my-image.png.

На данный момент это все, что вам нужно знать

Файловая система Windows стремится использовать обратный слеш (backslash), а не косую черту, например C:\windows. Это не имеет значения, даже если вы разрабатываете веб-сайт на Windows, вы все равно должны использовать обычные слеши в вашем коде.

## Что должно быть сделано?

К настоящему моменту структура вашей папки должна выглядеть примерно так:




# Основы HTML

HTML (Hypertext Markup Language) - это код, который используется для структурирования и отображения веб-страницы и её контента. Например, контент может быть структурирован внутри множества параграфов, маркированных списков или с использованием изображений и таблиц данных. Как видно из названия, эта статья даст вам базовое понимание HTML и его функций.

## Что такое HTML на самом деле?

HTML не является языком программирования; это язык разметки, и используется, чтобы сообщать вашему браузеру, как отображать веб-страницы, которые вы посещаете. Он может быть сложным или простым, в зависимости от того, как хочет веб-дизайнер. HTML состоит из ряда элементов, которые вы используете, чтобы вкладывать или оборачивать различные части контента, чтобы

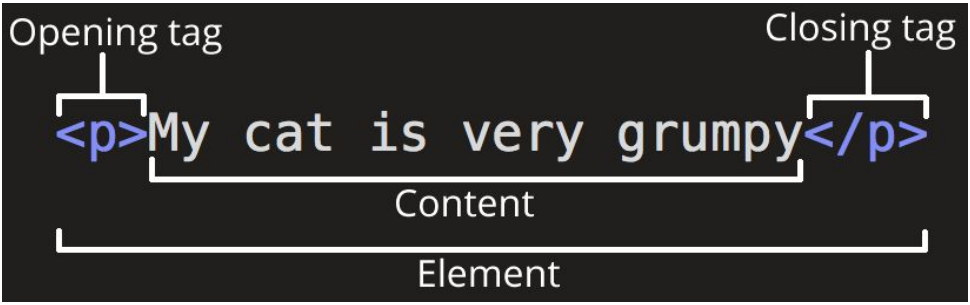
заставить контент отображаться или действовать определенным образом. Ограждающие теги могут сделать слово или изображение ссылкой на что-то еще, могут сделать слова курсивом, сделать шрифт больше или меньше и так далее. Например, возьмем следующую строку контента:

My cat is very grumpy

Если мы хотим строку, стоящую саму по себе, мы можем указать, что это параграф, заключая ее в тег элемента параграфа ( ):

My cat is very grumpy

АНАТОМИЯ HTML ЭЛЕМЕНТА
Давайте рассмотрим элемент абзаца более подробно.




Главными частями нашего элемента являются:

1. Открывающий тег: Состоит из имени этого элемента (в данном случае "p"), заключенного в открывающие и закрывающие угловые скобки. Указывает, где элемент начинается или начинает действовать, в данном случае — где начинается параграф.
2. Закрывающий тег: Это то же самое, что и открывающий тег, за исключением того, что он включает в себя косую черту перед именем элемента. Указывает, где элемент заканчивается, в данном случае — где заканчивается параграф. Отсутствие

закрывающего тега является одной из наиболее распространенных ошибок начинающих и может приводить к странным результатам.

3. Контент: Это контент элемента, который в данном случае является просто текстом.
4. Элемент: Открывающий тег плюс закрывающий тег, плюс контент вместе составляют элемент.

Элементы также могут иметь атрибуты, которые выглядят так:


Атрибуты содержат дополнительную информацию об элементе, которую вы не хотите показывать в фактическом контенте. В данном случае, class это имя атрибута, а editor-note это значение атрибута. Класс позволяет дать элементу идентификационное имя, которое может позже использоваться, чтобы обращаться к элементу с информацией о стиле и прочих вещах.
Атрибут всегда должен иметь:

1. Пространство между ним и именем элемента (или предыдущим атрибутом, если элемент уже имеет один или несколько атрибутов)
2. Имя атрибута, а затем знак равенства
3. Значение атрибута, заключенное с двух сторон в кавычки

ВЛОЖЕННЫЕ ЭЛЕМЕНТЫ
Вы можете располагать элементы внутри других элементов — это называется вложением. Если мы хотим заявить, что наша кошка ОЧЕНЬ раздражена, мы можем заключить слово "very" в элемент  , который указывает, что слово должно быть сильно акцентированно:

My cat is very grumpy.

Вы также должны убедиться, что ваши элементы вложены правильно: в примере выше мы открыли первым  элемент, затем  элемент, потом мы должны закрыть сначала  элемент, затем . Приведенное ниже неверно:

My cat is very grumpy.

Элементы должны открываться и закрываться правильно, поэтому они явно располагаются внутри или снаружи друг друга. Если они перекрываются, как в примере выше,  ваш веб-браузер будет пытаться сделать наилучшее предположение на основе того, что вы пытались сказать, и вы можете получить непредсказуемые результаты. Так что не стоит этого делать!

ПУСТЫЕ ЭЛЕМЕНТЫ
Некоторые элементы не имеют контента, и называются пустыми элементами. Возьмем  элемент <img>, который уже имеется в нашем HTML:

<img src="images/firefox-icon.png" alt="My test image">

Он содержит два атрибута, но не имеет закрывающего </img> тега, и никакого внутреннего контента. Это потому, что элемент изображения не оборачивает контент для влияния на него. Его целью является вставка изображения в HTML страницу в нужном месте.

АНАТОМИЯ HTML ДОКУМЕНТА
Мы завершили изучение основ отдельных HTML элементов, но они не очень полезны сами по себе. Теперь мы посмотрим, как отдельные элементы объединяются в целую HTML страницу. Давайте вернемся к коду, который мы записывали в наш index.html:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <img src="images/firefox-icon.png" alt="My test image">
  </body>
</html>
```

Здесь мы имеем:

- <!DOCTYPE html> — доктайп. В прошлом, когда HTML был молод (около 1991/1992), доктайпы должны были выступать в качестве ссылки на набор правил, которым HTML страница должна была следовать, чтобы считаться хорошим HTML, что могло означать автоматическую проверку ошибок и другие полезные вещи. Однако в наши дни, никто не заботится об этом, и он на самом деле просто исторический артефакт, который должен быть включен для того, что бы все работало правильно. На данный момент это все, что вам нужно знать.
- <html></html> — элемент <html>. Этот элемент оборачивает весь контент на всей странице, и иногда известен как корневой элемент.
- <head></head> — элемент <head>. Этот элемент выступает в качестве контейнера для всего, что вы пожелаете включить на HTML страницу, но не являющегося контентом, который вы показываете пользователям вашей страницы. К ним относятся такие вещи, как ключевые слова и описание страницы, которые будут появлялись в результатах поиска, CSS стили нашего контента, кодировка и другое.
- <body></body> — элемент <body>. В нем содержится весь контент, который вы хотите показывать пользователям, когда они посещают вашу страницу, будь то текст, изображения, видео, игры, проигрываемые аудиодорожки или что-то еще.

- <meta charset="utf-8"> — этот элемент устанавливает utf-8 кодировку вашего документа, которая включает в себя большинство символов из всех известных человечеству языков. По сути, теперь документ может обрабатывать любой текстовый контент, который вы в него вложите. Нет причин не устанавливать её, так как это может помочь избежать некоторых проблем в дальнейшем.
- <title></title> — это устанавливает заголовок для вашей страницы, который является названием, появляющимся на вкладке браузера загружаемой страницы, и используется для описания страницы, когда вы добавляете ее в закладки/избранное.

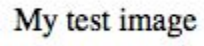
## Изображения

Давайте снова обратим наше внимание на элемент изображения:

<img src="images/firefox-icon.png" alt="My test image">

Как было сказано раньше, код встраивает изображение на нашу страницу в нужном месте. Это делается с помощью атрибута src (source), в котором содержится путь к нашему файлу изображения. Мы также включили атрибут alt (alternative). В этом атрибуте, вы указываете поясняющий текст для пользователей, которые не могут увидеть изображение, возможно, по следующим причинам:

1. У них присутствуют нарушенния зрения. Пользователи со значительным нарушением зрения часто используют инструменты, называемые Screen Readers ("экранные дикторы"), которые читают для них альтернативный текст.
2. Что-то пошло не так, в результате чего изображение не отобразилось. Например, попробуйте намеренно изменить путь в вашем атрибуте src, сделав его неверным. Если вы сохраните и перезагрузите страницу, то вы должны увидеть что-то подобное вместо изображения:




Альтернативный текст - это "пояснительный текст". Он должен предоставить читателю достаточно информации, чтобы иметь представление о том, что передает изображение. В этом примере наш текст "My test image" не годится. Намного лучшей альтернативой для нашего логотипа Firefox будет "The Firefox logo: a flaming fox surrounding the Earth" ("Логотип Firefox: огненный Лис вокруг Земли"). Сейчас попробуйте придумать более подходящий альтернативный текст для вашего изображения.

## Разметка текста

В этом разделе рассмотрим некоторые из основных HTML элементов, которые вы будете использовать для разметки текста.

ЗАГОЛОВКИ
Элементы заголовка позволяют вам указывать определенные части вашего контента в качестве заголовков или подзаголовков вашего контента. Точно так же, как книга имеет название, названия глав и подзаголовков, HTML документ может содержать то же самое. HTML включает шесть уровней заголовков <h1>–<h6>, хотя обычно вы будете использовать не более 3-4 :

```
<h1>My main title</h1>
<h2>My top level heading</h2>
<h3>My subheading</h3>
<h4>My sub-subheading</h4>
```

Теперь попробуйте добавить подходящее название для вашей HTML страницы, чуть выше  элемента <img>.

ПАРАГРАФЫ

Как было сказано раньше, в  элементах  содержатся параграфы текста; вы будете использовать их регулярно при разметке текстового контента:

This is a single paragraph

Добавьте свой текст в один или несколько параграфов, расположенных прямо под элементом <img>.

СПИСКИ

Большая часть веб-контента является списками и HTML имеет специальные элементы для них. Разметка списка всегда состоит по меньшей мере из двух элементов. Наиболее распространенными типами списков являются нумерованные и ненумерованные списки:

1. Ненумерованные списки - это списки, где порядок пунктов не имеет значения, например список покупок. Они оборачиваются в элемент <ul>.
2. Нумерованные списки - это списки, где порядок пунктов имеет значение, например рецепт. Они оборачиваются в  элемент <ol>.

Каждый пункт внутри списков располагается внутри  элемента <li> (list item).
Например, если мы хотим включить часть следующего фрагмента параграфа в список:

At Mozilla, we’re a global community of technologists, thinkers, and builders working together ... 

Мы могли бы изменить разметку на эту:

At Mozilla, we’re a global community of

<ul>

```
  <li>technologists</li>
  <li>thinkers</li>
  <li>builders</li>
</ul>

<p>working together ... </p>
```

Попробуйте добавить упорядоченный или неупорядоченный список на свою страницу.

## Ссылки

Ссылки имеют очень большое значение — они являются тем, что из Web делает WEB. Для добавления ссылки нужно использовать простой элемент — <a> — а это сокращение от "anchor" ("якорь"). Для того, чтобы текст в вашем параграфе стал ссылкой, выполните следующие действия:

1. Выберите некоторый текст. Мы выбрали текст "Mozilla Manifesto".
2. Оберните текст в элемент <a>, например так:
3. <a>Mozilla Manifesto</a>
4. Задайте элементу <a> атрибут href, например так:
5. <a href="">Mozilla Manifesto</a>
6. Заполните значение этого атрибута веб-адресом, на который вы хотите указать ссылку:
7. <a href="https://www.mozilla.org/en-US/about/manifesto/">Mozilla Manifesto</a>

Вы можете получить неожиданные результаты, если в самом начале веб-адреса вы опустите https:// или http:// часть, называемую протоколом. После создания ссылки, кликните по ней, чтобы убедиться, что она работает так, как вы хотели.
href сначала может выглядеть довольно непонятым выбором для имени атрибута. Если у вас возникли проблемы с тем, чтобы

запомнить его, можете запомните, что атрибут href образуется как hypertext reference ("гипертекстовая ссылка").
Теперь добавьте ссылку на вашу страницу, если вы еще не сделали этого.

ЗАКЛЮЧЕНИЕ
Если вы следовали всем инструкциям в этой статье, то вы должны увидеть в конечном итоге страницу, аналогичную рисунке ниже. Вы можете посмотреть ее здесь: (http://mdn.github.io/beginner-html-site/).


Если вы застряли, вы всегда можете сравнить свою работу с нашим готовым примером кода на GitHub.

# Основы CSS

CSS (Cascading Style Sheets) — это код, который вы используете для стилизирования вашей веб-страницы. В Основах CSS мы расскажем то, что вам нужно знать, чтобы начать. Мы ответим на такие вопросы: Как я могу сделать мой текст черным или красным? Как я могу сделать так, чтобы мой контент появлялся в разных местах экрана? Как украсить мою веб-страницу фоновыми изображениями и цветами?

## Что такое CSS на самом деле?

Как и HTML, CSS не является языком программирования. Это язык таблицы стилей, то есть он позволяет выборочно применять стили к элементам в HTML документах. Например, чтобы выбрать все элементы параграфа на HTML странице и превратить их текст в красный, вы должны написать на CSS следующее:

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

Давайте попробуем: вставить эти три строки CSS в новый файл в вашем текстовом редакторе, а затем сохраним файл как style.css в вашей папке styles.

Однако нам нужно применять CSS к нашему HTML документу, в противном случае CSS стиль не повлияет на то, как ваш браузер отображает HTML документ.

Откройте ваш файл index.html и вставьте следующую строку в шапку, то есть между <head> и </head> тегами:

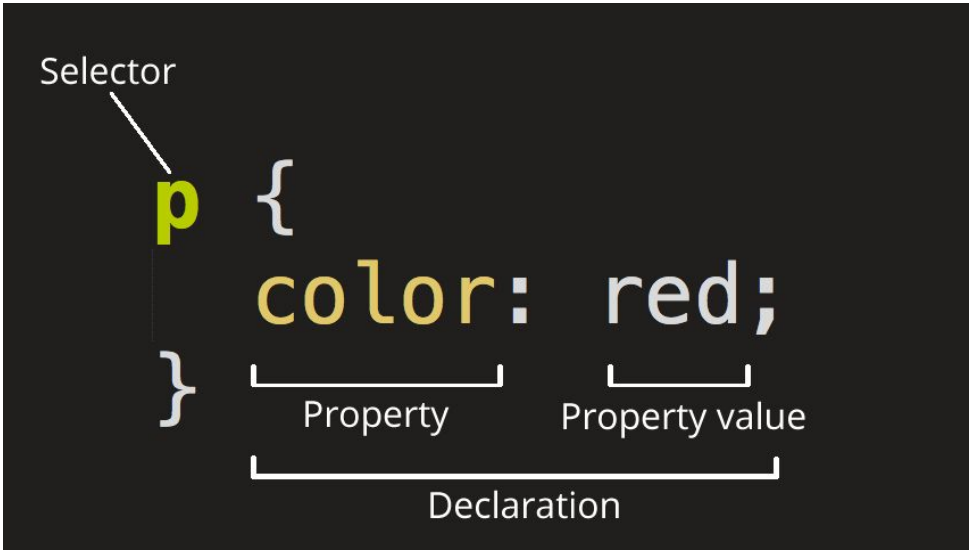
1. <link href="styles/style.css" rel="stylesheet" type="text/css">

2. Сохраните index.html и загрузите его в вашем браузере. Вы должны увидеть что-то вроде этого:


Если текст вашего параграфа  теперь красный, поздравляем. Вы написали свой первый успешный CSS!

АНАТОМИЯ НАБОРА ПРАВИЛ CSS

Давайте посмотрим на вышеупомянутый CSS немного более подробно:




Вся структура называется набором правил (но зачастую для краткости "правило" ). Отметим также имена отдельных частей:

Селектор (Selector)
Имя HTML элемента в начале набора правил. Он выбирает элемент(ы) для применения стиля (в данном случае, элементы p ). Для стилизования другого элемента, просто измените селектор.

Объявление (Declaration)
Одно правило, например color: red; указывает, какие из свойств элемента вы хотите стилизовать.

Свойства (Properties)
Способы, которыми вы можете стилизовать данный HTML элемент (в данном случае color является свойством для элементов p). В CSS вы можете выбрать, какие свойства вы хотите затронуть в вашем правиле.

Значение свойства (Property value)
Справа от свойства, после двоеточия, находится значение свойства, в котором выбирается одно из множества возможных значений для данного свойства (у свойства color есть множество значений, помимо red).

Обратите внимание на другие важные части синтаксиса:

- Каждый набор правил (кроме селектора) должен быть обернут в фигурные скобки ({}).
- В каждом объявлении необходимо использовать двоеточие (:), чтобы отделить свойство от его значений.
- В каждом наборе правил вы должны использовать точку с запятой (;), чтобы отделить каждое объявление от следующего.

Таким образом, чтобы изменить несколько значений свойств сразу, вам просто нужно написать их, разделяя точкой с запятой, например так:

```
p {
  color: red;
  width: 500px;
  border: 1px solid black;
}
```

МНОЖЕСТВЕННЫЙ ВЫБОР ЭЛЕМЕНТОВ
Вы также можете выбрать множество элементов разного типа и применить единый набор правил для всех из них. Добавьте несколько селекторов, разделенных запятыми. Например:

```
p,li,h1 {
  color: red;
}
```

РАЗНЫЕ ТИПЫ СЕЛЕКТОРОВ

Существует множество различных типов селектора. Выше мы рассматривали только селектор элементов, который выбирает все элементы данного типа в HTML документе. Но мы можем сделать выбор более конкретным. Вот некоторые из наиболее распространенных типов селекторов:

| Имя селектора | Что выбирает | Пример |
|---|---|---|
| Селектор элемента (иногда называемый селектором тега или типа) | Все HTML элемент(ы) указанного типа. | p<br>Выбирает <p> |
| ID селектор | Элемент на странице с указанным ID (на одной HTML странице, может быть только один элемент с каким-либо ID). | #my-id<br>Выбирает <p id="my-id"> или <a id="my-id"> |
| Селектор класса | Элемент(ы) на странице с указанным классом (множество экземпляров класса может объявляться на странице). | .my-class<br>Выбирает <p class="my-class"> и <a class="my-class"> |
| Селектор атрибута | Элемент(ы) на странице с указанным атрибутом. | img[src]<br>Выбирает <img src="myimage.png"> но не <img> |

| | | |
|---|---|---|
| Селектор псевдокласса | Указанные элемент(ы), но только в случае определенного состояния, например, при наведении курсора. | a:hover Выбирает <a>, но только тогда, когда указатель мыши наведен на ссылку. |

Существует много селекторов. Вы можете найти более подробный список в нашем Руководстве селекторов.

## Шрифты и текст

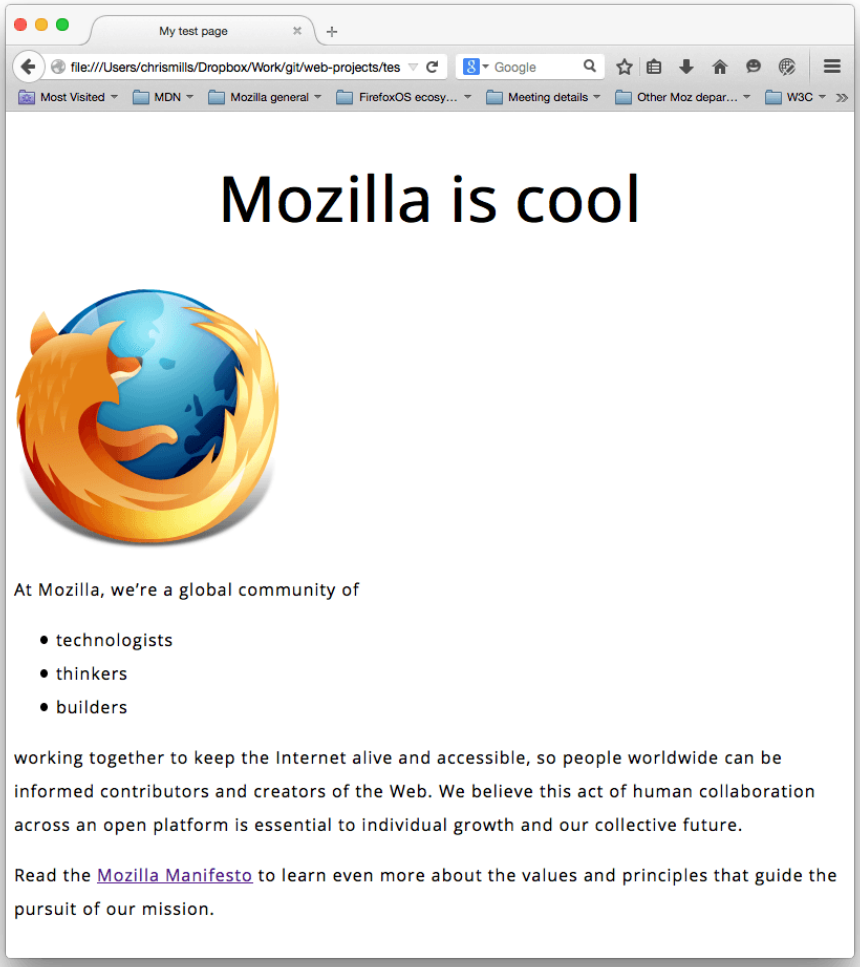
Теперь, когда мы изучили некоторые основы CSS, давайте добавим ещё несколько правил и информацию в наш файл style.css, чтобы наш пример выглядел хорошо. Прежде всего, давайте сделаем наши шрифты и текст немного лучше.

1. Прежде всего, вернитесь и найдите вывод из Google Fonts, который вы уже где-то сохранили. Добавьте элемент <link ... > где-нибудь внутри шапки вашего index.html (снова, в любом месте между тегами <head> и </head>). Это будет выглядеть примерно так:
2. <link href='http://fonts.googleapis.com/css?family=Open+Sans' rel='stylesheet' type='text/css'>
3. Затем удалите существующее правило в вашем style.css файле. Это был хороший тест, но красный текст на самом деле не выглядит очень хорошо.
4. Добавьте следующие строки в нужное место, заменив строку placeholder фактической font-family строкой, которую вы получили от Google Fonts. (font-family просто означает, какой шрифт(ы) вы хотите использовать для вашего текста). Это правило сначала устанавливает глобальный базовый шрифт и размер шрифта для всей страницы (поскольку <html> является родительским элементом для всей страницы, и все элементы внутри него наследуют такой же font-size и font-family):

5. html {
   font-size: 10px; /* px значит 'пиксели': базовый шрифт будет 10 пикселей в высоту */
   font-family: placeholder: здесь будет имя шрифта из Google fonts
   }
6. Примечание: Я добавил в комментарии объяснения, что означает "px". Все в CSS документе между /* и */ является CSS комментарием, который браузер игнорирует, когда он обрабатывает код. Это место, где вы пишите полезные заметки о том, что вы делаете.
7. Теперь мы установим размер шрифта для элементов, содержащих текст внутри HTML тела (<h1>, <li>, и <p>). Мы также отцентрируем текст нашего заголовка и установим некоторую высоту строки и расстояние между буквами в теле документа, чтобы сделать его немного более удобным для чтения:
8. h1 {
   font-size: 60px;
   text-align: center;
   }

   p, li {
   font-size: 16px;
   line-height: 2;
   letter-spacing: 1px;
   }

Вы можете настроить значения px по своему усмотрению, чтобы ваш дизайн выглядел так, как вы хотите, но, в общем, ваш дизайн должен выглядеть вот так:

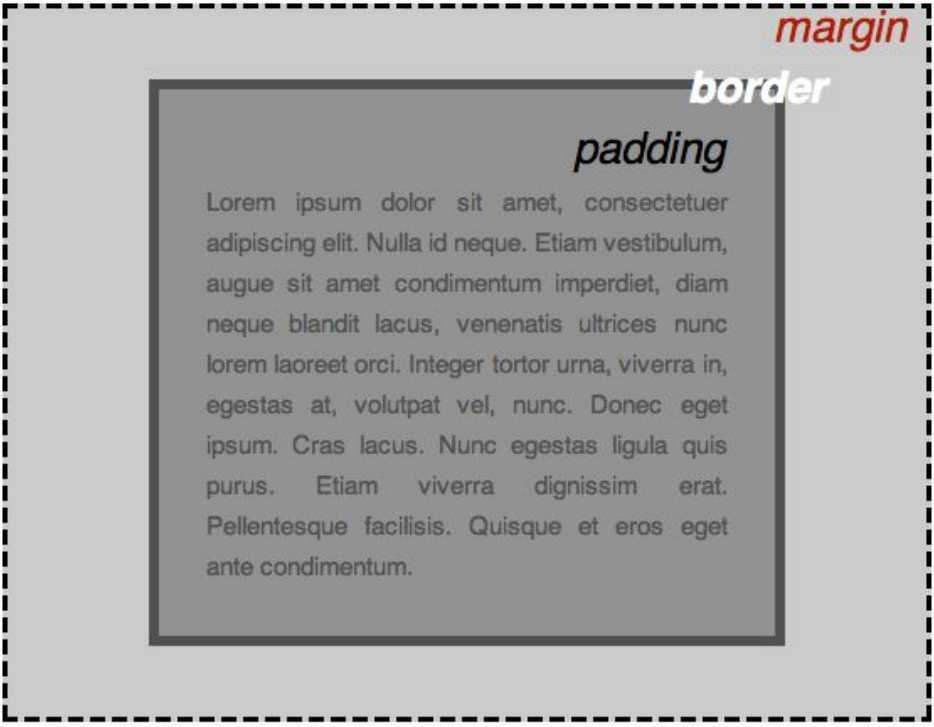



## Блоки, блоки и ещё раз блоки

Одну вещь, которую вы заметите в написании CSS -это то, что там постоянно используются блоки - установка их размера, цвета и расположения и т.д. Большинство HTML элементов на странице могут быть представлены как блоки, расположенные друг на друге.

Не удивительно, CSS макет основан главным образом на блочной модели (box model). Каждый из блоков, занимающий пространство на вашей странице имеет свойства, такие как:

- padding, пространство только вокруг контента (например, вокруг параграфа текста)
- border, сплошная линия, которая расположена рядом с padding
- margin, пространство вокруг внешней стороны элемента




В этом разделе мы также используем:

- width (ширину элемента)
- background-color, цвет позади контента и padding элементов
- color, цвет контента элемента (обычно текста)
- text-shadow: устанавливает тень на тексте внутри элемента
- display: устанавливает режим отображения элемента (не волнуйтесь об этом пока что)

Итак, давайте начнем и добавим больше CSS на нашей странице! Продолжайте добавлять эти новые правила, расположенные в нижней части страницы, и не бойтесь экспериментировать с изменением значений, чтобы увидеть, как это работает.

ИЗМЕНЕНИЕ ЦВЕТА СТРАНИЦЫ

```
html {
  background-color: #00539F;
}
```

Это правило устанавливает цвет фона для всей страницы. Измените код цвета сверху, на цвет который вы выбрали при планировании вашего сайта.

РАЗБИРАЕМСЯ С ТЕЛОМ

```
body {
  width: 600px;
  margin: 0 auto;
  background-color: #FF9500;
  padding: 0 20px 20px 20px;
  border: 5px solid black;
}
```

Теперь для body элемента. Здесь есть немало деклараций, так что давайте пройдем через них всех по одному:

- width: 600px; — заставляет тело быть всегда 600 пикселей в ширину.
- margin: 0 auto; — когда вы установите два значения для таких свойств как margin или padding, первое значение элемента влияет на верхнюю и нижнюю сторону (сделав их 0 в данном случае), и второе значение для левой и правой стороны (здесь, auto является особым значением, которое делит доступное пространство по горизонтали поровну слева и

справа). Вы также можете использовать один, три или четыре значения.

- background-color: #FF9500; — как и прежде, устанавливает цвет фона элемента. Я использовал красновато-оранжевый для тела, в отличие от темно-синего цвета для html элемента. Идите вперед и экспериментируйте. Не стесняйтесь использовать white или те, которые вы хотите.
- padding: 0 20px 20px 20px; — у нас есть четыре значения, установленные для padding, чтобы сделать немного пространства вокруг нашего контента. В этот раз мы не устанавливаем padding на верхней части тела, но делаем 20 пикселей слева, снизу и справа. Значения устанавливаются сверху, справа, снизу, слева, в таком порядке.
- border: 5px solid black; — просто устанавливает сплошную черную рамку шириной 5 пикселей со всех сторон тела.

ПОЗИЦИОНИРОВАНИЕ И СТИЛИЗАЦИЯ НАШЕГО ЗАГОЛОВКА ГЛАВНОЙ СТРАНИЦЫ

```
h1 {
  margin: 0;
  padding: 20px 0;
  color: #00539F;
  text-shadow: 3px 3px 1px black;
}
```

Вы, возможно, заметили, что есть ужасный разрыв в верхней части тела. Это происходит потому, что браузеры применяют некоторый стиль по умолчанию для <h1> элемента (по сравнению с другими), даже если вы не применяли какой-либо CSS вообще! Это может звучать как плохая идея, но мы хотим, чтобы веб-страница без стилей имела базовую читаемость. Чтобы избавиться от разрыва, мы переопределили стиль по умолчанию, установив margin: 0;.

Затем мы установили заголовку верхний и нижний padding на 20 пикселей, и сделали текст заголовка того же цвета, как и цвет фона html.
Здесь, мы использовали одно довольно интересное свойство это text-shadow, которое применяет текстовую тень для текстового контента элемента. Он имеет следующие четыре значения:

- Первое значение пикселей задает горизонтальное смещение тени от текста — как далеко она движется поперек: отрицательное значение должно двигать ее влево.
- Второе значение пикселей задает вертикальное смещение тени от текста — как далеко она движется вниз, в этом примере: отрицательное значение должно переместить ее вверх.
- Третье значение пикселей задает радиус размытия тени — большое значение будет означать более размытую тень.
- Четвертое значение задает основной цвет тени.

И вновь попробуйте поэкспериментировать с различными значениями, чтобы посмотреть, что можно придумать.

ЦЕНТРИРОВАНИЕ ИЗОБРАЖЕНИЯ

```
img {
  display: block;
  margin: 0 auto;
}
```

В заключение, мы отцентрируем изображение, чтобы сделать его лучше. Мы можем использовать margin: 0 auto уловку снова, как мы это делали раньше для body, но мы также должны сделать что-то еще. Элемент body является блочным, это означает, что занимает много места на странице и может иметь margin и другие значения отступов применяемые к нему. Изображения, наоборот, являются строчными элементами, то есть они этого не могут. Таким образом,

чтобы применять margin к изображению, мы должны дать изображению блочное поведение с помощью display: block;.

Не стоит беспокоиться, если вы еще не понимаете display: block; и блочное/строчное различие. Вы поймете, когда будете изучать CSS более подробно.

ЗАКЛЮЧЕНИЕ
Если вы застряли, вы всегда можете сравнить свою работу с нашим готовым примером кода на Github.

Если вы следовали всем инструкциям в этой статье, вы должны получить страницу, которая выглядит примерно так (вы также можете посмотреть нашу версию здесь):

# Основы JavaScript

JavaScript – это язык программирования, который добавляет интерактивность на ваш веб-сайт (например: игры, отклик на нажатие кнопки или при вводе данных в формы, динамические стили, анимация). Эта статья поможет вам начать работать с этим захватывающим языком и даст вам представление о том, на что он способен.

## Что такое JavaScript на самом деле?

JavaScript ("JS" для краткости) — это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах. Его разработал Brendan Eich, сооснователь проекта Mozilla, Mozilla Foundation и Mozilla Corporation.

Вы можете сделать очень многое с JavaScript. Вы можете начать с малого, с простых функций, таких как карусели, галереи изображений, изменяющиеся макеты и отклик на нажатие кнопок. Когда вы станете более опытным в языке, вы сможете создавать игры, анимированную 2D и 3D графику, полномасштабные приложения с базами данных и многое другое!

JavaScript сам по себе довольно компактный, но очень гибкий, и разработчиками написано много инструментов поверх основного JavaScript языка, которые разблокируют огромное количество дополнительных функций с очень небольшим усилием. К ним относятся:

- Программные Интерфейсы приложения (API) встроенные в браузеры, обеспечивающие различные функциональные возможности, такие как динамическое создание HTML и установку CSS стилей, захват и манипуляция видеопотоком,

работа с веб-камерой пользователя или генерация 3D графики и аудио сэмплов.
- Сторонние API позволяют разработчикам внедрять функциональность в свои сайты от других разработчиков, таких как Twitter или Facebook.
- Также вы можете применить к вашему HTML сторонние фреймворки и библиотеки, что позволит вам ускорить создание сайтов и приложений.

## Пример "hello world"

Предыдущий раздел звучит очень многообещающе, и это на самом деле так — JavaScript является одной из самых перспективных веб-технологий, и когда вы освоитесь и начнете использовать его, ваши веб-сайты перейдут в новое измерение мощи и креативности. Тем не менее, с JavaScript немного более сложно освоиться, чем с HTML и CSS, и вам придется начать с малого, продолжая изучение небольшими шагами.
Для начала, мы покажем вам, как добавить некоторые основы JavaScript на вашу страницу, чтобы создать "hello world!" пример (стандартный пример в основах программирования).

Важно: Если вы не следили за остальным нашим курсом, скачайте этот пример кода и используйте его в качестве стартовой точки.

1. Для начала, перейдите на ваш тестовый сайт и создайте новый файл с именем main.js. Сохраните его в вашей папке scripts.
2. Далее, перейдите в ваш index.html файл и введите следующий тег в новую строку прямо перед закрывающим тегом </body>:
3. <script src="scripts/main.js"></script>
4. Он, в основном, делает ту же работу, что и элемент <link> для CSS — добавляет JavaScript на страницу, позволяя ему взаимодействовать с HTML (и CSS, и чем-нибудь ещё на странице).

5. Теперь добавьте следующий код в файл main.js:
6. var myHeading = document.querySelector('h1');
   myHeading.textContent = 'Hello world!';
7. Теперь убедитесь, что HTML и JavaScript файлы сохранены, и загрузите index.html в браузере. Вы должны увидеть что-то вроде этого:


Причиной, по которой мы поставили элемент <script> в нижней части HTML файла, является то, что HTML загружается в браузере по порядку появления его в файле. Поэтому, если JavaScript загружается первым и ему нужно взаимодействовать с HTML ниже его, он не сможет работать, так как JavaScript будет загружен раньше, чем HTML, с которым нужно работать. Поэтому, располагать JavaScript в нижней части HTML страницы считается хорошей практикой.

ЧТО ПРОИЗОШЛО?
Итак, ваш заголовок текста был изменен на "Hello world!" с помощью JavaScript. Мы сделали это с помощью вызова функции querySelector(), захватив ссылку на наш заголовок и сохранив ее в переменной названной myHeading. Это очень похоже на то, что мы делали в CSS с помощью селекторов. Если вы хотите что-то сделать с элементом, то для начала вам нужно его выбрать.
После этого, мы устанавливаем значение переменной myHeading в textContent свойство (которое представляет собой контент заголовка) "Hello world!".

# Ускоренный курс по основам языка

Давайте познакомимся с некоторыми основными возможностями языка JavaScript, чтобы дать вам больше понимания, как это всё работает. Более того, эти возможности являются общими для всех языков программирования. Если вы сможете понять эти основы, вы будете в состоянии начать программировать, как ни в чём не бывало!

Важно: В этой статье, попробуйте вводить примеры строк кода в вашей JavaScript консоли, чтобы увидеть, что происходит.

ПЕРЕМЕННЫЕ

Переменные — это контейнеры, внутри которых вы можете хранить значения. Вы начинаете объявлять переменную с ключевым словом var, за которым следует любое имя, которым вы захотите ее назвать:

var myVariable;

Все инструкции в JavaScript должны заканчиваться точкой с запятой, чтобы указать, где заканчивается эта инструкция. Если вы не добавите их, вы можете получить неожиданные результаты.

Вы можете назвать переменную практически как угодно, но есть некоторые ограничения для её имени (смотрите в этой статье в правилах именования переменных.) Если вы не уверены, вы можете проверить имя вашей переменной, чтобы увидеть корректно ли оно.

JavaScript чувствителен к регистру — myVariable отличается от переменной myvariable. Если у вас возникают проблемы в вашем коде, проверьте регистр!

После объявления переменной, вы можете присвоить ей значение:

myVariable = 'Bob';

Вы можете сделать обе эти операции на одной и той же строке, если вы захотите:
var myVariable = 'Bob';

Вы можете получить значение, просто вызвав переменную по имени:

myVariable;

После установки значения переменной,  вы можете изменить его позже:

var myVariable = 'Bob';
myVariable = 'Steve';

Обратите внимание, что переменные имеют разные типы данных:

ТИП ДАННЫХ: String
Строка текста. Чтобы показать, что переменная является строкой, вы должны заключить ее в кавычки.
ПРИМЕР: var myVariable = 'Bob';

ТИП ДАННЫХ: Number
Числа. Числа не имеют кавычек вокруг них.
ПРИМЕР: var myVariable = 10;

ТИП ДАННЫХ: Boolean
Значение True(Правда)/False(Ложь). Слова true и false специальные ключевые слова в JS, и не нуждаются в кавычках.
ПРИМЕР: var myVariable = true;

ТИП ДАННЫХ: Array
Структура, которая позволяет хранить несколько значений в одной ссылке.
ПРИМЕР: var myVariable = [1,'Bob','Steve',10];
Обратиться к каждому элементу массива можно так:

myVariable[0], myVariable[1], и т.д.

ТИП ДАННЫХ: Object
В принципе что угодно. Все в JavaScript является объектом, и может храниться в переменной. Имейте это в виду.
ПРИМЕР: var myVariable = document.querySelector('h1');
Все это из вышеприведенных примеров.

Так для чего нам нужны переменные? Переменные должны были сделать что-нибудь интересное в программировании. Если значения не могли бы изменяться, то вы не могли бы ничего сделать динамическим, например, персонализировать приветственное сообщение или сменить изображение, отображаемое в галерее изображений.

КОММЕНТАРИИ

Вы можете поместить комментарии в JavaScript код, так же как вы делали это в CSS:
```
/*
Всё, что находится тут комментарий.
*/
```
Если ваш комментарий не содержит переноса строк, то зачастую легче поставить две косые черты, как тут:
```
// Это комментарий
```

ОПЕРАТОРЫ

operator — это математический символ, который производит результат, основанный на двух значениях (или переменных). В приведенной ниже таблице вы можете увидеть некоторые из наиболее простых операторов, наряду с некоторыми примерами, которые опробуйте в JavaScript консоли.

ОПЕРАТОР: сложение/конкатенация

Используется для сложения двух чисел или склеивания двух строк вместе.
СИМВОЛЫ: +
ПРИМЕР: 6 + 9;
"Hello " + "world!";

ОПЕРАТОР: вычитание, умножение, деление
Они делают то, что вы ожидаете от них в математике.
СИМВОЛЫ: -, *, /
ПРИМЕР: 9 - 3;
8 * 2; // умножение в JS это звездочка
9 / 3;

ОПЕРАТОР: оператор присваивания
Вы уже видели его: он присваивает значение переменной.
СИМВОЛЫ: =
ПРИМЕР: var myVariable = 'Bob';

ОПЕРАТОР: тождественный оператор
Делает проверку, если увидит, что два значения равны друг другу, то возвращает true/false (Boolean) результат.
СИМВОЛЫ: ===
ПРИМЕР: var myVariable = 3;
myVariable === 4;

ОПЕРАТОР: отрицание, неравенство
Возвращает логически противоположное значение, которое этому предшествует; превращает true в false, и т.д. Когда используется вместе с оператором равенства, оператор отрицания проверяет, являются ли два значения не равными.
СИМВОЛЫ: !, !==
ПРИМЕР: Основное выражение true, но сравнение возвращает false, потому что мы отрицаем его:
var myVariable = 3;
!(myVariable === 3);
Здесь мы проверяем "myVariable НЕ равно 3". Это возвращает false, потому что myVariable равно 3.

```
var myVariable = 3;
myVariable !== 3;
```

Существует намного больше операторов для изучения, но этих пока хватит. Смотрите их полный список в разделе Выражения и операторы.

Смешивание типов данных может привести к некоторым неожиданным результатам при выполнении вычислений, поэтому будьте осторожны, когда вы ссылаетесь на ваши переменные правильно, вы получаете результаты, которые вы ожидаете. Например, введите "35" + "25" в вашу консоль. Почему вы не получили результат, который вы ожидали? Потому, что кавычки превратили числа в строки, так что у вас в итоге получилась конкатенация строк, а не сложение чисел. Если вы введете, 35 + 25, то получите правильный результат.

УСЛОВИЯ

Условие — это кодовая структура, которая позволяет вам проверить истинно ли выражение, а затем выполнить другой код в зависимости от результата. Самая распространенная форма условия называется, if ... else. Например:

```
var iceCream = 'chocolate';
if (iceCream === 'chocolate') {
  alert('Yay, I love chocolate ice cream!');
} else {
  alert('Awwww, but chocolate is my favorite...');
}
```

Выражение внутри if ( ... ) — это проверка, которая использует тождественный оператор (как описано выше), чтобы сравнить переменную iceCream со строкой chocolate и увидеть равны ли они. Если это сравнение возвращает true, выполнится первый блок кода. Если нет, этот код пропустится и выполнится второй блок кода, после инструкции else.

ФУНКЦИИ

Функции являются способом упаковки функциональности, которую вы хотите использовать повторно, так что всякий раз, когда вы хотите получить функциональность, вы можете просто вызвать функцию, а не постоянно переписывать весь код. Вы уже видели некоторые виды использования функций выше, например:

1. var myVariable = document.querySelector('h1');
2. alert('hello!');

Эти функции, document.querySelector и alert, встроены в браузер для того, чтобы вы использовали их всякий раз, когда вам это необходимо.
Если вы видите что-то, что выглядит как имя переменной, но имеет скобки — () — после него, скорее всего это функция. Функции часто принимают аргументы — биты данных, с которыми они должны выполнить свою работу. Они передаются в скобки, и разделяются запятыми, если существует более одного аргумента.
Например, функция alert() делает всплывающий блок, появляющийся в окне браузера, но мы должны дать ему строку в качестве аргумента, чтобы сказать функции, что писать во всплывающем блоке.
Хорошей новостью является то, что вы можете определять свои собственные функции — в следующем примере мы напишем простую функцию, которая принимает два числа в качестве аргументов и умножает их:

```
function multiply(num1,num2) {
  var result = num1 * num2;
  return result;
}
```

Попробуйте запустить вышеупомянутую функцию в консоли, затем попробуйте использовать вашу новую функцию несколько раз, например:
multiply(4,7);
multiply(20,20);
multiply(0.5,3);

Инструкция return сообщает браузеру вернуть переменную result из функции, которую можно будет использовать. Это необходимо потому, что переменные определенные внутри функций доступны только внутри этих функций. Это называется областью видимости переменной. (Читайте больше о видимости переменных.)

СОБЫТИЯ
Для создания действительной интерактивности на веб-сайте, вам необходимы события — это структура, которая слушает то, что происходит в браузере, а затем позволяет вам запускать код в ответ на это. Наиболее очевидным является событие клика, которое вызывается браузером, когда мы щёлкаем по чему-то мышью. Для демонстрации этого, попробуйте ввести следующую команду в вашей консоли, а затем щёлкните по текущей веб-странице:

```
document.querySelector('html').onclick = function() {
    alert('Ouch! Stop poking me!');
}
```

Существуют множество способов прикрепить событие к элементу. Здесь мы выбираем HTML элемент и устанавливаем ему обработчик свойства onclick анонимной функцией (т.е. безымянной) которая содержит код, который мы хотим запустить, когда происходит событие клика.
Обратите внимание, что
document.querySelector('html').onclick = function() {};
эквивалентно

var myHTML = document.querySelector('html');
myHTML.onclick = function() {};
Просто, так короче.

## Прокачаем пример нашего веб-сайта

Теперь, когда мы прошли некоторые основы JavaScript, давайте добавим несколько крутых несложных функций в пример нашего сайта, чтобы дать вам некоторое представление о принципах работы.

ДОБАВЛЕНИЕ СМЕНЫ ИЗОБРАЖЕНИЯ

В этом разделе мы добавим ещё одно изображение на наш сайт и добавим некоторый простой JavaScript для переключения между двумя изображениями, когда по ним щелкнули.

1. В первую очередь найдите другое изображение, которые вы хотели бы показать на вашем сайте. Убедитесь что оно такого же размера, как ваше первое изображение или максимально близкое к нему.
2. Сохраните изображение в вашу папку images.
3. Перейдите в ваш файл main.js и введите следующий JavaScript. (Если ваш "hello world" JavaScript по-прежнему существует, удалите его.)
4. var myImage = document.querySelector('img');

```
myImage.onclick = function() {
    var mySrc = myImage.getAttribute('src');
    if(mySrc === 'images/firefox-icon.png') {
      myImage.setAttribute ('src','images/firefox2.png');
    } else {
      myImage.setAttribute ('src','images/firefox-icon.png');
    }
}
```

5. Сохраните все файлы и загрузите index.html в браузере. Теперь, когда вы щёлкните по изображению, оно должно измениться на другое!

Итак, мы сохраняем ссылку на наш элемент изображения в переменной myImage. Далее, мы создаём этой переменной обработчик события onclick с анонимной функцией. Теперь, каждый раз, когда на этот элемент изображения щёлкнут:

1. Мы получаем значение из атрибута src изображения.
2. Мы используем условие для проверки значения src, равен ли путь к исходному изображению:
   - Если это так, мы меняем значение src на путь ко 2 изображению, заставляя другое изображение загружаться внутри элемента <image>.
   - Если это не так (значит, оно должно было уже измениться), мы меняем значение src, возвращаясь к первоначальному пути изображения, каким он был изначально.

## ДОБАВЛЕНИЕ ПЕРСОНАЛЬНОГО ПРИВЕТСТВЕННОГО СООБЩЕНИЯ

Далее мы добавим немного другого кода, чтобы изменить заголовок страницы и включить персонализированное приветственное сообщение, когда пользователь впервые заходит на сайт. Это приветственное сообщение будет сохраняться, когда пользователь уходит с сайта, а затем приходит назад. Мы также добавим возможность изменять пользователя и, поэтому приветственное сообщение необходимо в любое время.

1. В index.html, добавьте следующую строку перед элементом <script>:
2. <button>Change user</button>
3. В main.js, добавьте следующий код в конец файла, точно так, как написано - он захватит ссылки на новую кнопку и заголовок, и сохранит их в переменные:
4. var myButton = document.querySelector('button');
   var myHeading = document.querySelector('h1');

5. Теперь добавьте следующую функцию для установки персонализированного приветствия - она ничего не будет делать, но мы будем использовать её позже:
6. function setUserName() {
   var myName = prompt('Please enter your name.');
   localStorage.setItem('name', myName);
   myHeading.innerHTML  = 'Mozilla is cool, ' + myName;
  }
7. Эта функция содержит функцию prompt(), которая вызывает диалоговое окно, немного похожее на alert() кроме того, что prompt() просит пользователя ввести некоторые данные, и сохраняет эти данные в переменной, после того как пользователь нажимает OK. В данном случае, мы просим пользователя ввести его имя. Далее, мы вызываем API под названием localStorage, которое позволяет нам сохранять данные в браузере и извлекать их позднее. Мы используем функцию setItem() из localStorage для создания и хранения данных в свойстве под названием 'name', и устанавливаем это значение в переменную myName, которая содержит имя введенное пользователем. В конце мы устанавливаем textContent заголовку в виде строки и имени пользователя.
8. Затем добавляем блок if ... else, чтобы мы могли вызвать код инициализации, приложение устанавливает выполняет его, когда оно впервые загружается:
9. if(!localStorage.getItem('name')) {
   setUserName();
  } else {
   var storedName = localStorage.getItem('name');
   myHeading.innerHTML  = 'Mozilla is cool, ' + storedName;
  }
10. Этот первый блок использует оператор отрицания (логическое НЕ) чтобы проверить, существуют ли данные в пункте name. Если нет, то функция setUserName() запускается для его создания. Если это так (то есть, пользователь установил его во время предыдущего посещения), мы извлекаем

сохраненное имя, используя getItem() и устанавливаем textContent заголовку в виде строки плюс имя пользователя, так же, как мы делали внутри setUserName().

11. В заключение, установим обработчик события onclick на кнопку, чтобы при щелчке по ней запускалась функция setUserName(). Это позволяет пользователю задавать новое имя, всякий раз, когда он захочет, нажав на кнопку:
12. myButton.onclick = function() {
    setUserName();
    }

Теперь, когда вы в первый раз посетите сайт, мы попросим вас ввести ваше имя пользователя, а затем дадим вам персональное сообщение. Затем вы можете изменить имя, в любой момент, нажав на кнопку. В качестве дополнительного бонуса, имя хранится внутри localStorage, оно сохранится после закрытия сайта, поэтому персонализированное сообщения все ещё будет там, когда вы откроете сайт снова!

ЗАКЛЮЧЕНИЕ

Если вы следовали всем инструкциям в этой статье, в конечном итоге вы должны получить страницу, которая выглядит примерно так (вы также можете посмотреть нашу версию здесь):


Если вы застряли, вы всегда можете сравнить свою работу с нашим готовым примером кода на Github.

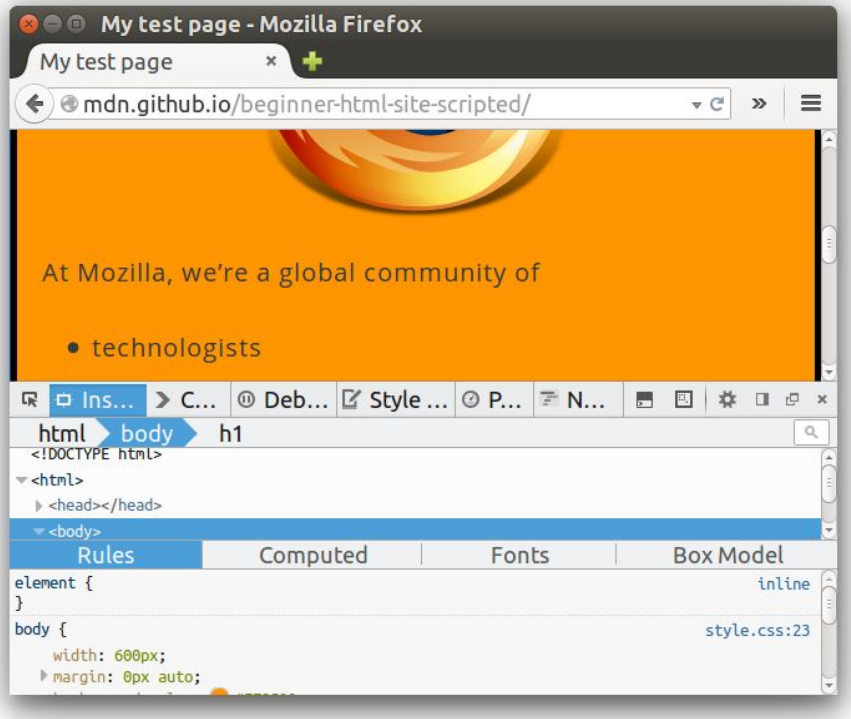
# Обзор инструментов разработки в браузерах

Каждый современный интернет-браузер оснащён мощными инструментами для веб-разработчика. Эти инструменты позволяют делать различные вещи, от изучения загруженных в настоящий момент HTML, CSS и JavaScript до отображения в каких ресурсах нуждается страница и как долго она будет загружаться. Эта статья научит Вас использовать базовые функции инструментов разработчика в Вашем браузере.

Прежде чем начать заниматься с примерами, откройте пример сайта для начинающих, с которым мы работали на предыдущих занятиях. Вам следует держать его открытым, чтобы выполнить описанные ниже действия.

## Как открыть инструменты веб-разработчика в браузере?

Панель разработчика находится в нижней части Вашего браузера :




Как её отобразить? Есть три варианта:

КЛАВИАТУРА.

Ctrl + Shift + I, кроме

Internet Explorer. (клавиша - F12)

Mac OS X. (сочетание клавиш - ⌘ + ⌥ + I )

ПАНЕЛЬ МЕНЮ.

FIREFOX. Открыть меню ☰ ➤ Developer ➤ Инструменты разработки, илиИнструменты ➤ Веб-разработка ➤ Инструменты разработки

CHROME. Дополнительные инструменты ➤ Инструменты разработчика
SAFARI. Разработка ➤ Показать Web Inspector . Если Вы не видите меню "Разработка", зайдите в  Safari ➤ Настройки ➤ Дополнительно,  и проверьте стоит ли галочка напротив "Показать меню разработки".

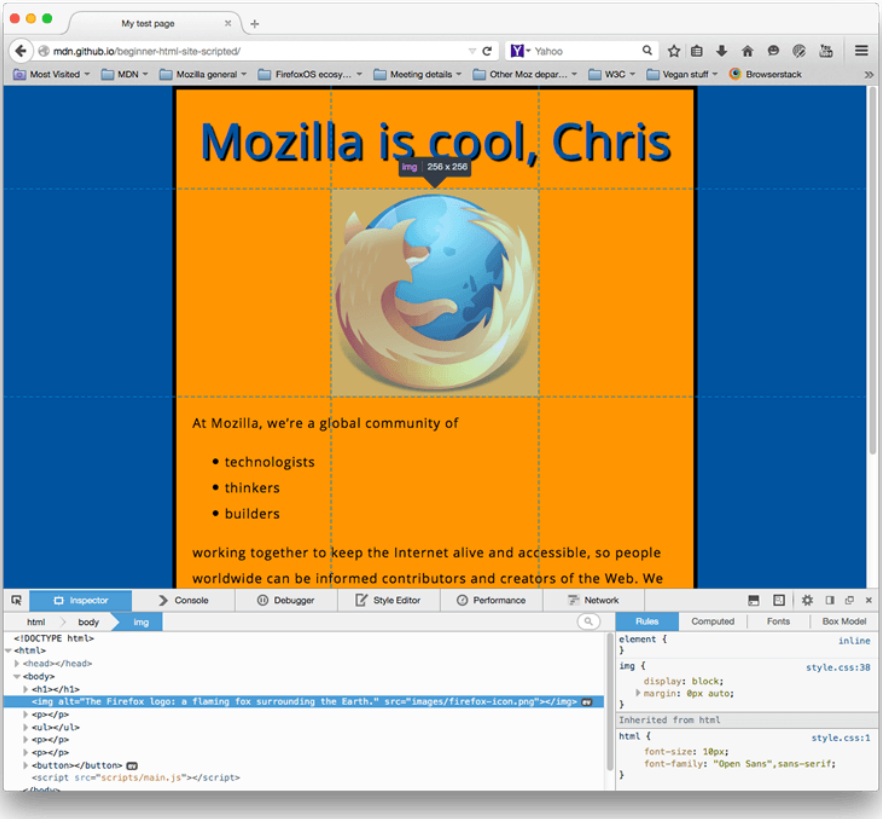
OPERA. Меню ➤ Разработка ➤ Инструменты разработчика. Если Вы не видите меню "Разработка",  включите его отображение, перейдя в Меню ➤ Другие инструменты ➤ Показать меню разработчика.

КОНТЕКСТНОЕ МЕНЮ.

Нажмите правой кнопкой мыши на любом участке веб-страницы (Ctrl-клик для Mac), появится контекстное меню, в котором Вам нужно выбрать пункт Исследовать Элемент. (дополнение: этот способ отобразит Вам код того элемента, на котором вы щёлкнули правой кнопкой.)

# Inspector: DOM обозреватель и CSS редактор

По-умолчанию, в панели открывается вкладка Inspector, Вы можете увидеть это на скриншоте снизу. Этот инструмент позволяет Вам видеть, как HTML-код выглядит на странице в настоящем времени, также как CSS, который применён к каждому элементу на странице. Это также позволяет Вам в реальном времени редактировать как HTML, так и CSS. Внесённые изменения можно увидеть непосредственно в окне браузера.



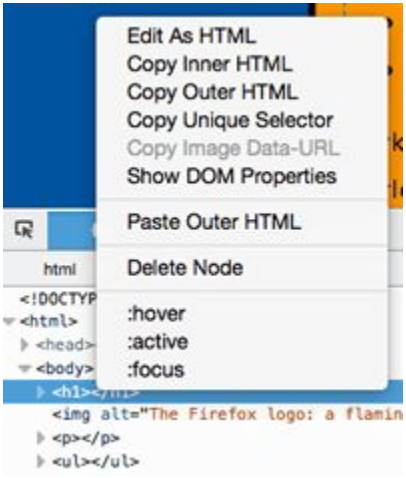

Если Вы не видите Inspector,

- Нажмите на вкладку Inspector .

- В Internet Explorer, нажмите на DOM Обозреватель, или нажмите Ctrl + 1.
- В Safari, элементы управления представлены не так чётко, но Вы должны увидеть HTML код, если Вы не выбрали что-то другое в окне разработки. Нажмите на кнопку Стиль, чтобы увидеть CSS.

ОБЗОР DOM INSPECTOR

Для начала, попробуйте нажать правой кнопкой мыши (Ctrl+клик) по элементу HTML в DOM inspector и посмотрите на контекстное меню. Пункты меню могут различаться в разных браузерах, но важными из них являются одни и те же:




- Удалить узел (иногда Удалить элемент). Удаляет текущий элемент.
- Править как HTML (иногда Добавить атрибут/Править текст). Позволяет редактировать HTML и видеть результат "вживую". Очень полезно для отладки и тестирования.
- :hover/:active/:focus. Заставляет элементы переключить своё состояние на то, к которому применён Ваш стиль.
- Копировать/Копировать как HTML. Копирует текущий выделенный HTML.

Попробуйте изменить что-нибудь через окно Inspector на Вашей странице прямо сейчас. Дважды кликните по элементу, или нажмите

правой кнопкой мыши и выберите Править как HTML из контекстного меню. Вы можете сделать любые изменения, какие захотите, но Вы не сможете их сохранить.

ОБЗОР CSS РЕДАКТОРА

По-умолчанию, CSS редактор отображает CSS свойства применённые к текущему выбранному элементу:


Эти функции особенно удобны:

- Свойства, применённые к текущему элементу, отображаются в порядке убывания приоритета.
- Можно убирать галочки напротив свойств для того чтобы видеть, что получится, если их удалить.
- Нажмите на маленькую стрелочку рядом со свойством, чтобы увидеть все его эквиваленты.
- Нажмите на имя свойства или его значение, чтобы открыть текстовое окошко, в котором Вы можете задать новые значения и увидеть, как изменится Ваш элемент с новыми значениями.
- Рядом с каждым свойством указаны имя файла и номер строки. где располагается это свойство. Щелчок по этому пути

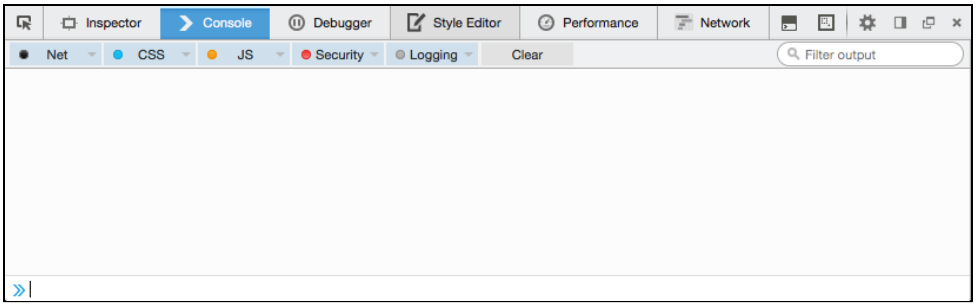
перенесёт Вас в окно, где можно редактировать этот CSS и сохранить.
- Вы можете также нажать на закрывающуюся фигурную скобку любого свойства, чтобы вывести текстовое поле на новую строку, где Вы сможете написать совершенно новую декларацию для Вашей страницы.

Вы должно быть уже заметили другие вкладки в  CSS редакторе:
- Вычислено: Здесь указаны все вычисления свойств выделенного элемента (окончательные, нормализованные значения применённые браузером).
- Блоковая модель: Отображает блочную модель выделенного элемента, здесь Вы можете увидеть внешние и внутренние отступы, а также границы применённые к элементу, здесь также указан их размер.
- Анимации: В Firefox, на вкладке Анимации Вы можете увидеть анимации применённые к выделенному элементу.

УЗНАТЬ БОЛЬШЕ ОБ INSPECTOR В РАЗЛИЧНЫХ БРАУЗЕРАХ:
- Firefox Page inspector
- IE DOM Explorer
- Chrome DOM inspector (Inspector в Opera схож с Inspector в Chrome)
- Safari DOM inspector and style explorer

## Консоль JavaScript

Консоль JavaScript невероятно полезный инструмент для отладки JavaScript, если он не работает, как ожидалось. Она позволяет Вам загружать JavaScript вопреки порядку загрузки скрипта в браузере, и докладывает об ошибках как только браузер пытается выполнить Ваш код. Для доступа к консоли из любого браузера просто нажмите на кнопку Console. (В Internet Explorer, нажмите Ctrl + 2.) Откроется окно, как показано ниже:




Чтобы понять, что происходит, попробуйте ввести фрагменты кода в консоль один за другим (и затем нажмите Enter):

1. alert('hello!');
2. document.querySelector('html').style.backgroundColor = 'purple';
3. var myImage = document.createElement('img');
   myImage.setAttribute('src','https://farm4.staticflickr.com/3455/3372925208_e1f2aae4e3_b.jpg');
   document.querySelector('h1').appendChild(myImage);

Теперь попробуйте ввести следующую, неправильную версию кода и посмотрите, что из этого получится.

1. alert('hello!);
2. document.cheeseSelector('html').style.backgroundColor = 'purple';
3. var myImage = document.createElement('img');
   myBanana.setAttribute('src','https://farm4.staticflickr.com/3455/3372925208_e1f2aae4e3_b.jpg');
   document.querySelector('h1').appendChild(myImage);

Вы увидите некоторые ошибки, которые сообщит Вам браузер. Зачастую эти ошибки выглядят довольно загадочно, но они должны быть довольно простыми, чтобы можно было выяснить проблему!

УЗНАТЬ БОЛЬШЕ О JAVASCRIPT КОНСОЛИ В РАЗЛИЧНЫХ БРАУЗЕРАХ:

- Firefox Web Console
- IE JavaScript console
- Chrome JavaScript Console (Inpector в Opera схож с Inspector в Chrome)
- Safari Console

# Что дальше?

Ресурсы по JavaScript на Mozilla Developer Network:
https://developer.mozilla.org/ru/docs/Web/JavaScript
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference

Введение в HTML от Mozilla Developer Network:
https://developer.mozilla.org/ru/docs/Web/Guide/HTML/Introduction

Введение в CSS от Mozilla Developer Network:
https://developer.mozilla.org/ru/docs/Web/Guide/CSS/Getting_started