ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАЗРАБОТКА ВЕБ САЙТОВ ДЛЯ НАЧИНАЮЩИХ

HTML CSS JAVASCRIPT

MOZILLA

ОСНОВЫ ПРОГРАММИРОВАНИЯ

MOZILLA. РАЗРАБОТКА ВЕБ САЙТОВ ДЛЯ НАЧИНАЮЩИХ (HTML, CSS, JAVASCRIPT)

ВЕРСИЯ 2.0

- У фирмы Mozilla на сайте есть руководство по HTML, CSS и JavaScript. Мне нужно было сделать небольшую методичку, чтобы было удобно распечатывать и читать детям.
- Я не автор текста, просто взял информацию с разных страничек сайта Mozilla и оформил красиво. Эта методичка хорошо идет для школьников, которым я преподаю.
- Замечания и комментарии присылайте на:
  [general.loki.2018@gmail.com] (Сергей Куринный)

ВВЕДЕНИЕ

ЧТО ТАКОЕ HTML?

- HTML (Hypertext Markup Language) - код, который используется для структурирования и отображения веб-страницы и ее контента (содержимого).
- Контент может быть структурирован внутри множества параграфов, маркированных списков или с использованием изображений и таблиц данных.
- HTML не является языком программирования. Это язык разметки, и используется, чтобы сообщать 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 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 строку: **Мой кот скучает**
- Если мы хотим строку, стоящую саму по себе, мы можем указать, что это параграф, заключая ее в тег элемента параграфа ( ): ** Мой кот скучает **

ЧТО ТАКОЕ CSS?

- CSS (Cascading Style Sheets) - код, который вы используете для стилизирования веб-страницы.
- Мы ответим на такие вопросы: Как я могу сделать текст черным или красным? Как я могу сделать так, чтобы контент появлялся в разных местах экрана? Как украсить веб-страницу фоновыми изображениями и цветами?

ЧТО ТАКОЕ JAVASCRIPT?

- JavaScript - язык программирования, который дает возможность реализовывать сложное поведение сайта.

- Каждый раз, когда вы видите веб-страницу, которая не только отображает статическое содержимое, но и делает больше (своевременно отображает обновление контента, выводит интерактивные карты, 2D/3D анимацию или прокручивает видео), будьте уверены, здесь не обошлось без JavaScript.
- Считается, что JavaScript сложнее изучить, чем связанные с ним технологии, наподобие HTML и CSS. Перед изучением JavaScript, рекомендуем сначала ознакомиться с ними.
- 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 простой веб-сайт в Интернете несложно.

КАКИЕ ИНСТРУМЕНТЫ НУЖНЫ?

- КОМПЬЮТЕР. Может быть, это звучит очевидно для некоторых людей, но некоторые из вас читают эту книгу с телефона или планшета. Для серьезной веб-разработки, лучше приобрести компьютер. Причем экран ноутбука должен быть не меньше 15 дюймов. Оперативная память не меньше 8 гигабайт. В процессоре не меньше двух ядер.
- ТЕКСТОВЫЙ РЕДАКТОР. Это может быть бесплатный текстовый редактор, например:
  - Notepad++ (http://notepad-plus-plus.org/).
  - Visual Studio Code (https://code.visualstudio.com/).
- ВЕБ-БРАУЗЕР. Наиболее часто используемые браузеры:
  - Firefox (https://www.mozilla.org/ru/firefox/new/).
  - Chrome (https://www.google.com/chrome/browser/).
- ГРАФИЧЕСКИЙ РЕДАКТОР. Создавать изображения.
  - Paint.NET (http://www.getpaint.net/)

ПЛАНИРОВАНИЕ

- Обдумайте план и дизайн веб-сайта, прежде чем приступить к написанию кода, в том числе:
  - Какую информацию будет содержать веб-сайт?
  - Что будет делать сайт?
- Перед тем как делать что-то, вам нужны идеи. Что веб-сайт должен фактически делать?

Веб-сайт может делать все, что угодно, но для первой попытки, вы должны придерживаться простых вещей.

- Мы начнем с создания простой веб-страницы, содержащую заголовок, изображение и несколько абзацев. Для начала, нужно ответить на следующие вопросы:
  - О чем веб-сайт?
  - Какую информацию вы предоставляете о предмете?
- Напишите заголовок и несколько абзацев, и подумайте над изображениями, которые вы хотите показать на странице.
- Как будет выглядеть веб-сайт, в простых терминах высокого уровня. Какой цвет фона? Какой вид шрифта будет уместен: деловой, мультяшный, жирный 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.

НАБРОСОК ДИЗАЙНА:

- Возьмите ручку и бумагу и сделайте примерный набросок того, как вы хотите, чтобы выглядел сайт. Для веб-страницы должен получиться небольшой набросок, и вы должны взять это в привычку.
- 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.

АКТИВЫ

- На данном этапе неплохо начать собирать воедино весь контент, который будет располагаться на веб-странице.
- У вас должен быть текст, разбитый на заголовки и параграфы. Придерживайтесь этого правила.

ЦВЕТОВАЯ СХЕМА:

- Перейдите в инструмент выбора цвета и выберите цвет, который вам нравится. [https://tinyurl.com/vdalnvl]
- Когда вы щелкните по цвету, вы увидите странный код из шести цифр, например, #660066.

- Это шестнадцатеричный код и он представляет цвет. Скопируйте это код куда-нибудь прямо сейчас.

ИЗОБРАЖЕНИЯ:

- Чтобы выбрать изображение, перейдите в Google Картинки и найдите что-нибудь подходящее.
- Когда вы найдете нужное изображение, которое хотели, щелкните по изображению.
- Нажмите кнопку "В полном размере" (View image).
- На следующей странице, правым щелчком мыши на изображении, выберите "Сохранить изображение как..." (Save Image As...), и укажите место для хранения 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.

ЧТОБЫ ВЫБРАТЬ ШРИФТ:

- Перейдите на Google Fonts (http://www.google.com/fonts) и прокрутите список вниз, пока не найдете шрифт, который вам понравится. Вы также можете использовать элементы управления слева для дальнейшей фильтрации результатов.
- Щелкните по кнопке плюс рядом со шрифтом, который вы хотите выбрать. Далее, щелкните на панели в нижней части страницы, которая содержит список выбранных шрифтов.
- Когда панель откроется, скопируйте код для вставки в веб страничку. Он будет похож на:
- **<link href="https://fonts.googleapis.com/css?family=Gelasio&display=swap" rel="stylesheet">**

ФАЙЛЫ И ПАПКИ

ГДЕ РАЗМЕСТИТЬ НА КОМПЬЮТЕРЕ?

- 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 локальном компьютере.
- Когда вы работаете на веб-сайте локально на 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 ее туда, где вы сможете легко ее найти, может быть, на рабочем столе, в домашней папке или в корне жесткого диска.
- Выберите место для хранения проектов веб-сайта. Здесь, создайте новую папку с именем **web-projects** (или аналогичной). Это место, где будут храниться все проекты.
- Внутри этой первой папки, создайте другую папку для хранения первого веб-сайта. Назовите ее **test-site** (или как-то более творчески).

О РЕГИСТРЕ И ПРОБЕЛАХ:

- Вы заметите, что мы просим называть папки и файлы полностью в нижнем регистре без пробелов. Многие компьютеры и веб-серверы, чувствительны к регистру.
- Например, если вы положили изображение на 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.
- Браузеры, веб-серверы и языки программирования не обрабатывают пробелы последовательно. Если вы используете пробелы в имени файла, некоторые системы могут отнестись к имени файла как к двум именам файлов.
- Некоторые серверы заменяют пробелы в имени файла на "%20" (код для пробелов в URI), нарушая все ссылки.
- Лучше разделять слова с помощью тире и нижнего подчеркивания: **my-file.html** или **my_file.html**.
- Лучше всего приобрести привычку писать названия папок и файлов в нижнем регистре и без пробелов. Так вы столкнетесь с меньшим количеством проблем.

СТРУКТУРА САЙТА:

- В любом сайте есть: индексный файл HTML, папки с картинками, файлы стилей и файлы скриптов. Создайте их!
- **index.html:** Этот файл обычно содержит контент домашней страницы, то есть текст и изображения, которые люди видят, когда они впервые попадают на сайт. Используя текстовый редактор, создайте новый файл с именем index.html и сохраните его прямо внутри папки test-site.
- **Папка images:** Эта папка обычно содержит изображения, которые вы используете на сайте. Создайте папку с именем images внутри папки test-site.
- **Папка styles:** Эта папка содержит CSS код, используемый для стилизации контента (например, настройка текста и цвета фона). Создайте папку styles внутри test-site.
- **Папка scripts:** Эта папка содержит весь JavaScript код, используемый для добавления интерактивных функций на сайте. Создайте папку с именем scripts внутри test-site.
- На компьютерах под управлением Windows могут возникнуть проблемы с отображением имен файлов, поскольку у Windows есть 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 известных типов файлов**, затем щелкнуть 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 файл **index.html** и научим его отображать изображение.
- Скопируйте изображение, которое вы выбрали, в **images**.
- Откройте файл index.html и вставьте следующий код в файл именно в таком виде. Не беспокойтесь о том, что все это значит - позже мы рассмотрим код более подробно.

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>Это заголовок</title>
        <meta name="viewport"
        content="width=device-width, initial-scale=1" />
</head>
<body>
        <img src="" alt="Моя картинка">
</body>
</html>
```

- Строка **<img src="" alt="Моя картинка">** - 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 изображения, получим путь к файлу, который нам нужен.
- Он выглядит как **images/your-image-filename**. Например изображение, названное **firefox-icon.png**, имеет такой путь к файлу: **images/firefox-icon.png**.
- Вставьте путь к файлу в HTML код между двойными кавычками **src=""**.
- Сохраните HTML файл и загрузите его в браузере (двойной щелчок по файлу). Вы должны увидеть веб-страницу, отображающую изображение!

НЕСКОЛЬКО ПРАВИЛ О ПУТЯХ К ФАЙЛАМ:

- Для ссылки на файл в той же директории, что и вызывающий HTML файл, используйте имя файла без ничего, например, **my-image.jpg**.
- Для ссылки на файл в поддиректории, напишите имя директории в начале пути, плюс косую черту. Например: **subdirectory/my-image.jpg**.
- Для ссылки на 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.
- Файловая система Windows стремится использовать обратный слеш, 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 коде.

СТРУКТУРА ПАПКИ ДОЛЖНА ВЫГЛЯДЕТЬ ТАК:

HTML

АНАТОМИЯ ЭЛЕМЕНТА

- **ОТКРЫВАЮЩИЙ ТЕГ:** состоит из имени 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.
- **ЗАКРЫВАЮЩИЙ ТЕГ:** 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.
- **КОНТЕНТ:** контент элемента, который в данном случае является просто текстом.
- **ЭЛЕМЕНТ:** открывающий тег плюс закрывающий тег, плюс контент вместе составляют элемент.
- Элементы могут иметь атрибуты, они выглядят так:

- Атрибуты содержат дополнительную информацию об элементе, которую вы не хотите показывать в 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:
  - Пространство между ним и именем элемента (или предыдущим атрибутом, если элемент уже имеет один или несколько атрибутов).
  - Имя атрибута, а затем знак равенства.
  - 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 кошка ОЧЕНЬ скучает, мы можем заключить слово "очень" в элемент **** , который указывает, что слово должно быть акцентированно:
  ** Моя кошка очень скучает. **
- Вы также должны убедиться, что элементы вложены правильно: в примере выше мы открыли первым элемент, затем элемент, потом мы должны закрыть сначала элемент, затем . Это неверно:
  ** Моя кошка очень скучает. **

- Элементы должны открываться и закрываться правильно.
- Располагайте их явно внутри или снаружи друг друга. Если они перекрываются, веб-браузер будет пытаться сделать наилучшее предположение на основе того, что вы пытались сказать, и вы получите неправильные результаты.

ПУСТЫЕ ЭЛЕМЕНТЫ

- Некоторые элементы не имеют контента, и называются пустыми элементами. Возьмем элемент <img>, который уже имеется в HTML:
  <img src="images/firefox-icon.png"
  alt="Моя тестовая картинка">
- 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 ДОКУМЕНТА

- Вернемся к коду, который мы записывали в index.html.
- **<!DOCTYPE html>** - доктайп. В прошлом доктайпы должны были выступать в качестве ссылки на набор правил, которым HTML страница должна была следовать. 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.
- **<html></html>** - оборачивает весь контент на странице, он известен как корневой элемент.
- **<head></head>** - элемент выступает в качестве контейнера для всего, что вы желаете включить на HTML страницу, но не являющегося контентом, который вы показываете пользователям страницы. К ним относятся такие вещи, как ключевые слова и описание страницы, которые будут появлялись в результатах поиска, CSS стили контента, кодировка и другое.
- **<body></body>** - элемент содержит весь контент, который вы хотите показывать пользователям, когда они посещают страницу, будь то текст, изображения, видео, игры, проигрываемые аудиодорожки или что-то еще.

- **<meta charset="utf-8">** - элемент устанавливает utf-8 кодировку документа, которая включает в себя большинство символов из всех известных языков. Теперь документ может обрабатывать любой текстовый контент, который вы в него вложите. Нет причин не устанавливать ее, так как это поможет избежать проблем.
- **<title></title>** - устанавливает заголовок для страницы, который является названием, появляющимся на вкладке браузера загружаемой страницы, и используется для описания страницы, когда вы добавляете ее в избранное.
- **<meta name="viewport" content="width=device-width, initial-scale=1" />** - специальный код, который нужен для правильного отображения страницы на мобильных устройствах.

ИЗОБРАЖЕНИЯ

- Обратим внимание на элемент изображения: **<img src="images/firefox-icon.png" alt="Моя картинка">**
- Этот код встраивает изображение на страницу в нужном месте. Это делается с помощью атрибута **src**, в котором содержится путь к файлу изображения.
- Мы также включили атрибут **alt**. В этом атрибуте, вы указываете поясняющий текст для пользователей, которые не могут увидеть изображение по следующим причинам:
- У них присутствуют нарушенния зрения. Пользователи с нарушением зрения часто используют "экранные дикторы", которые читают для них альтернативный текст.
- Что-то пошло не так, в результате чего изображение не отобразилось. Например, попробуйте намеренно изменить путь в атрибуте src, сделав его неверным. Если вы сохраните и перезагрузите страницу, то должны увидеть альтернативный текст вместо изображения.
- Альтернативный текст - "пояснительный текст". Он должен предоставить достаточно информации, чтобы иметь представление о том, что передает изображение.
- В этом примере текст "**Моя картинка**" не годится. Намного лучшей альтернативой для логотипа Firefox будет "**Логотип Firefox: огненный Лис вокруг Земли**".

- Сейчас попробуйте придумать более подходящий альтернативный текст для изображения.

ЗАГОЛОВКИ

- Элементы заголовка позволяют указывать определенные части контента в качестве заголовков или подзаголовков 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>Главный заголовок</h1>**
- **<h2>Заголовок верхнего уровня</h2>**
- **<h3>Подзаголовок</h3>**
- **<h4>Подзаголовок Подзаголовка</h4>**
- Теперь попробуйте добавить подходящее название для HTML страницы, чуть выше элемента <img>.

ПАРАГРАФЫ

- В элементах содержатся параграфы текста. Вы будете использовать их регулярно при разметке текста:
  ** Это параграф **
- Добавьте 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.
- Ненумерованные списки - это списки, где порядок пунктов не имеет значения, например список покупок. Они оборачиваются в элемент **<ul>**.
- Нумерованные списки - списки, где порядок пунктов имеет значение. Они оборачиваются в элемент **<ol>**.
- Каждый пункт внутри списков располагается внутри элемента **<li>**.

- Например, если мы хотим включить часть следующего фрагмента параграфа в список:
  ** Mozilla, мы являемся мировым сообществом технологов, мыслителей и строителей, работающих вместе ... **
- Мы могли бы изменить разметку на эту:
  ** Mozilla, мы являемся мировым сообществом **
  <ul>
  **
технологов
**
  **
мыслителей
**
  **
строителей
**
  **
**
  ** работающих вместе ... **
- Попробуйте добавить упорядоченный или неупорядоченный список на страницу.

ССЫЛКИ

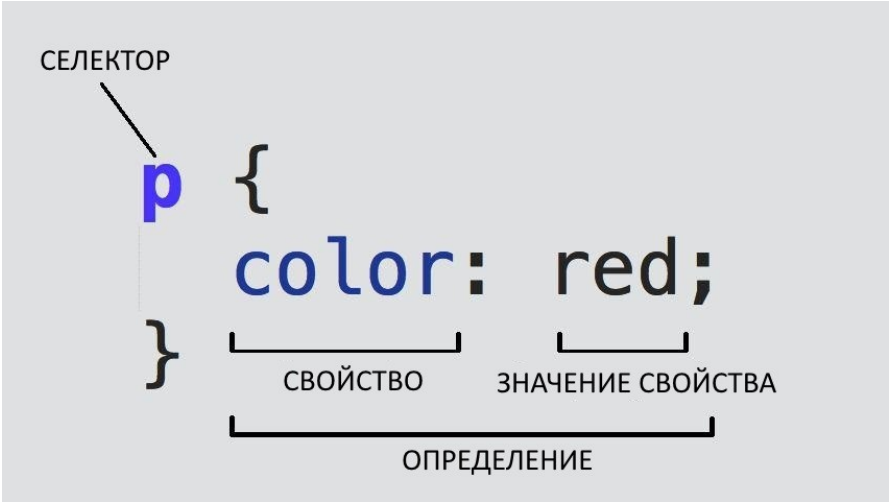
- Для добавления ссылки нужно использовать простой элемент - **<a>**. Для того, чтобы текст в параграфе стал ссылкой, выполните следующие действия:
- Выберите текст, например "Google".
- Оберните текст в элемент <a>, например так:
  <a>Google
- Задайте элементу <a> атрибут href, например так:
  Google
- Заполните значение этого атрибута веб-адресом, на который вы хотите указать ссылку:
  Google
- 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.
- Добавьте ссылку на страницу, если еще не сделали этого.

CSS

- CSS не является языком программирования. Это язык таблицы стилей, 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 следующее:
  p {
  color: red;
  }
- Давайте попробуем: вставить эти три строки CSS в новый файл в текстовом редакторе, а затем сохраним файл как **style.css** в папке **styles**.
- Нам нужно применить CSS к HTML документу, в противном случае CSS стиль не повлияет на то, как браузер отображает HTML документ.
- Откройте файл index.html и вставьте следующую строку в шапку, между <head> и </head> тегами:
  <link href="styles/style.css" rel="stylesheet" type="text/css">
- Сохраните index.html и загрузите его в браузере.
- Если текст параграфа теперь красный, поздравляем. Вы написали первый успешный CSS!
- Вы можете определить CSS прямо внутри HTML файла. Для этого используйте тег **<style></style>**. Внутри этого тега пишите так как будто это CSS файл:
  <style>
  p {
  color: red;
  }
  </style>
- Вы можете также определять стиль прямо внутри любого элемента. Например так:
  <table style="width: 100%; margin-bottom: 0.5em;">

АНАТОМИЯ НАБОРА ПРАВИЛ

- Давайте посмотрим на CSS более подробно. Вся структура называется набором правил (для краткости "правило"). Отметим имена отдельных частей:



- **СЕЛЕКТОР.** 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.
- **ОПРЕДЕЛЕНИЕ.** Одно правило, например **color: red;** укажет, какие из свойств элемента вы хотите стилизовать.
- **СВОЙСТВА.** 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 правиле.
- **ЗНАЧЕНИЕ СВОЙСТВА.** 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).

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.

- Чтобы изменить несколько значений свойств сразу, вам просто нужно написать их, разделяя точкой с запятой:
  p {
  color: red;
  width: 500px;
  border: 1px solid black;
  }
- Вы также можете выбрать множество элементов разного типа и применить единый набор правил для всех из них. Добавьте несколько селекторов, разделенных запятыми.
  p,li,h1 {
  color: red;
  }
- Все в CSS документе между **/*** и ***/** является комментарием, который браузер игнорирует, когда он обрабатывает код. Это место, где вы пишите заметки о том, что делаете.

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.
- Вы можете найти более подробный список в руководстве селекторов. [https://tinyurl.com/ql9686a]

СЕЛЕКТОР ЭЛЕМЕНТА:

- Иногда называется селектором тега или типа.
- ВЫБИРАЕТ: Все HTML элемент(ы) указанного типа.
- ПРИМЕР: **p**
- ДЕЛАЕТ: Выбирает все <p>

ID СЕЛЕКТОР:

- ВЫБИРАЕТ: Элемент с указанным ID (на одной странице, может быть только один элемент с каким-либо ID).
- ПРИМЕР: **#my-id**
- ДЕЛАЕТ: Выбирает <p id="my-id"> или <a id="my-id">

СЕЛЕКТОР КЛАССА:

- ВЫБИРАЕТ: Элемент(ы) на странице с указанным классом (множество экземпляров класса может быть на странице).
- ПРИМЕР: **.my-class**
- ДЕЛАЕТ: Выбирает <p class="my-class"> и <a class="my-class">

СЕЛЕКТОР АТРИБУТА:

- ВЫБИРАЕТ: Элемент(ы) на странице с указанным атрибутом.
- ПРИМЕР: **img[src]**
- ДЕЛАЕТ: Выбирает <img src="myimage.png"> но не <img>

СЕЛЕКТОР ПСЕВДО-КЛАССА:

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

ШРИФТЫ И ТЕКСТ

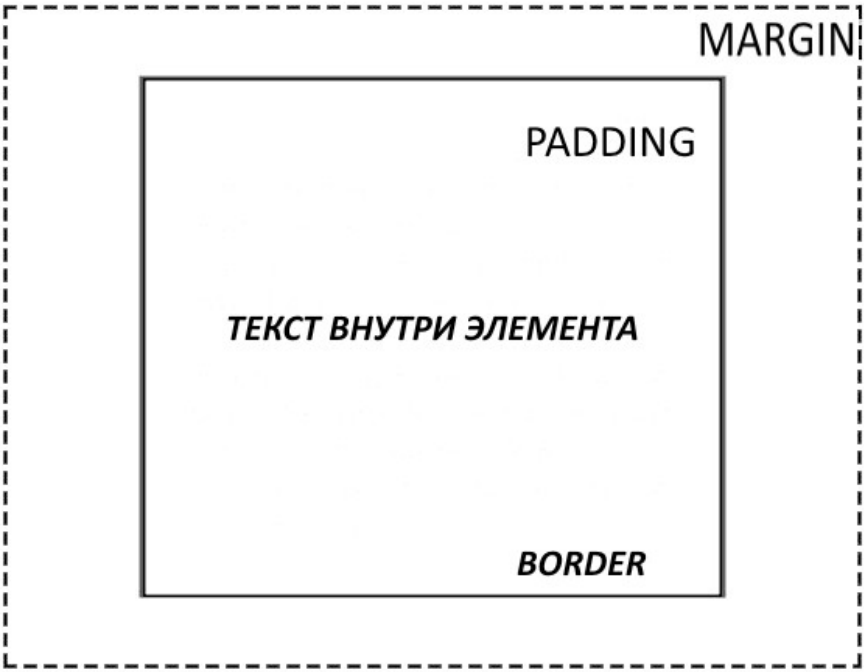
- Давайте сделаем шрифты и текст немного лучше.
- Прежде всего, вернитесь и найдите вывод из Google Fonts, который вы уже где-то сохранили.
- Добавьте элемент <link ... > внутри шапки index.html (в любом месте между <head> и </head>).
- Это будет выглядеть так (все в одной строке кода): **<link href='http://fonts.googleapis.com/css?family=Open+Sans' rel='stylesheet' type='text/css'>**
- Затем удалите существующее правило в style.css файле. Это был хороший тест, но красный текст на самом деле не выглядит очень хорошо.
- 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 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):
  html {
  font-size: 10px; /* шрифт будет 10 пикселей*/
  font-family: [placeholder]
  }
- В комментариях объяснения, что означает "px".
- Теперь мы установим размер шрифта для элементов, содержащих текст внутри HTML тела (<h1>, <li>, и <p>). Мы также отцентрируем текст заголовка и установим высоту строки и расстояние между буквами в теле документа, чтобы сделать его более удобным для чтения:
  h1 {
  font-size: 60px;
  text-align: center;
  }
  p, li {
  font-size: 16px;
  line-height: 2;
  letter-spacing: 1px;
  }

БЛОКИ

- В написании CSS постоянно используются блоки - установка их размера, цвета и расположения.
- Большинство HTML элементов на странице могут быть представлены как блоки, расположенные друг на друге.
- Каждый из блоков имеет свойства.
- **padding -** пространство только вокруг контента (например, вокруг параграфа текста).
- **border -** сплошная линия, расположена рядом с padding.
- **margin -** пространство вокруг внешней стороны элемента.
- **width** - ширина элемента.
- **background-color -** цвет позади контента.
- **color -** цвет контента элемента (обычно текста).
- **text-shadow** - устанавливает тень текста внутри элемента.

- **display** - устанавливает режим отображения элемента.
- Давайте добавим больше CSS на странице!



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 сайта.
- Выбрать цвета [https://htmlcolorcodes.com/].
- Браузер понимает английские названия более 100 цветов.
- Например Green (зеленый), Blue (синий), Orange (оранжевый), Red (красный), Yellow (желтый), Purple (фиолетовый), White (белый), Black (черный), Pink (розовый), Turquoise (бирюзовый), Violet (фиолетовый), SkyBlue (небесно-голубой), PaleGreen (светло-зеленый).

- Полный список названий цветов можно найти на сайте CSS-Tricks: [https://tinyurl.com/grpk652]

РАЗБИРАЕМСЯ С ТЕЛОМ

- **body {**
  width: 600px;
  margin: 0 auto;
  background-color: #FF9500;
  padding: 0 20px 20px 20px;
  border: 5px solid black;
  }
- **width: 600px;** - задает ширину тела в 600 пикселей.
- **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 (**auto** является особым значением, которое делит пространство по горизонтали поровну слева и справа). Вы можете использовать один, два, три или четыре значения.
- **background-color: #FF9500;** - устанавливает цвет фона элемента. Мы использовали 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 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;** - устанавливает сплошную черную рамку шириной 5 пикселей со всех сторон.

СТИЛИЗАЦИЯ ЗАГОЛОВКА

- У нас есть ужасный разрыв в верхней части. Это происходит потому, что браузеры применяют некоторый стиль по умолчанию для <h1> элемента.

- Чтобы избавиться от разрыва, мы переопределили стиль по умолчанию, установив **margin: 0;**.
- **h1 {**
  margin: 0;
  padding: 20px 0;
  color: #00539F;
  text-shadow: 3px 3px 1px black;
  }
- Затем мы установили заголовку верхний и нижний **padding** на 20 пикселей, и сделали текст заголовка того же цвета, как и цвет фона html.
- Мы использовали свойство **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 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.

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;**.

ИНСТРУМЕНТЫ В БРАУЗЕРАХ

- Каждый браузер оснащен инструментами для веб-разработчика. Эти инструменты позволяют делать различные вещи, от изучения загруженных HTML, CSS и JavaScript до отображения в каких ресурсах нуждается страница и как долго она будет загружаться.

КАК ОТКРЫТЬ ИНСТРУМЕНТЫ РАЗРАБОТЧИКА?

- Панель разработчика находится в нижней или правой части браузера. Как ее отобразить? Есть три варианта:
- КЛАВИАТУРА. Ctrl + Shift + I
- ПАНЕЛЬ МЕНЮ. FIREFOX. Открыть меню > Веб-разработка > Инструменты разработки
- ПАНЕЛЬ МЕНЮ. CHROME. Открыть меню > Дополнительные инструменты > Инструменты разработчика
- КОНТЕКСТНОЕ МЕНЮ. Нажмите правой кнопкой мыши на любом участке веб-страницы, появится контекстное меню, в котором нужно выбрать пункт **"Исследовать Элемент"**. Этот способ отобразит код того элемента, на котором вы щелкнули правой кнопкой.

ОБЗОР DOM INSPECTOR:

- По-умолчанию, в панели открывается вкладка Inspector. Этот инструмент позволяет Вам видеть, как HTML-код выглядит на странице в настоящем времени, также как CSS, который применен к каждому элементу на странице.
- Он также позволяет в реальном времени редактировать HTML и CSS. Изменения можно увидеть в окне браузера.
- Если вы не видите Inspector, нажмите на его вкладку.
- 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. Полезно для отладки и тестирования.
- **:hover/:active/:focus**. Заставляет элементы переключить свое состояние на то, к которому применен стиль.
- **Копировать/Копировать как HTML**. Копирует текущий выделенный HTML.

- Попробуйте изменить что-нибудь через окно Inspector на 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 не сможете их сохранить.

ОБЗОР CSS РЕДАКТОРА:

- По-умолчанию, CSS редактор отображает CSS свойства примененные к текущему выбранному элементу.
- Свойства, примененные 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 элемент.
- Рядом с каждым свойством указаны имя файла и номер строки. где располагается это свойство.
  Щелчок по этому пути перенесет 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 сможете написать совершенно новую декларацию для страницы.
- Вы могли заметить другие вкладки в CSS редакторе:
  - **Вычислено**: Здесь указаны все вычисления свойств выделенного элемента (окончательные значения примененные браузером).

- **Блоковая модель**: Отображает блочную модель выделенного элемента. Здесь вы можете увидеть внешние и внутренние отступы, а также границы примененные к элементу, здесь также указан их размер.
- **Анимации**: На этой вкладке вы можете увидеть анимации примененные к выделенному элементу.

КОНСОЛЬ JAVASCRIPT:

- Консоль JavaScript - полезный инструмент для отладки. Она позволяет загружать JavaScript вопреки порядку загрузки скрипта в браузере, и докладывает об ошибках как только браузер пытается выполнить код.
- Для доступа к консоли нажмите на кнопку **Console**.
- Чтобы понять, что происходит, введите фрагменты кода в консоль один за другим (и затем нажмите Enter):
- **alert('Привет');**
- **document.querySelector('html').style.backgroundColor = 'purple';**
- Теперь попробуйте ввести следующую, неправильную версию кода и посмотрите, что из этого получится.
- **alert('Привет);**
- **document.cheeseSelector('html').style.backgroundColor = 'purple';**
- Вы увидите ошибки, которые сообщит браузер. Зачастую эти ошибки выглядят довольно загадочно, но они должны быть довольно простыми, чтобы можно было выяснить проблему!

JAVASCRIPT

- JavaScript – язык программирования, который добавляет интерактивность на веб-сайт (например: игры, отклик на нажатие кнопки или при вводе данных в формы, динамические стили, анимация).
- JavaScript ("JS" для краткости) - динамический язык программирования, который применяется к HTML документу.
- Он может обеспечить интерактивность на веб-сайтах. Его разработал Brendan Eich, сооснователь проекта Mozilla.
- 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, вы сможете создавать игры, анимированную 2D и 3D графику, приложения с базами данных и многое другое!
- Разработчиками написано много инструментов поверх основного JavaScript языка, которые разблокируют огромное количество дополнительных функций.

ДОПОЛНЕНИЯ:

- 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.
- Вы можете применить к HTML сторонние фреймворки и библиотеки, что позволит ускорить создание сайтов.

ПРИМЕР "ПРИВЕТ МИР"

- 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. С JavaScript немного более сложно освоиться, чем с HTML и CSS, и вам придется начать с малого, продолжая изучение небольшими шагами.
- Мы покажем, как добавить некоторые основы JavaScript на страницу, чтобы создать "Привет мир!" пример.
- Для начала, перейдите на тестовый сайт и создайте новый файл с именем **main.js**. Сохраните его в папке **scripts**.
- Далее, перейдите в index.html и введите следующий тег в новую строку прямо перед закрывающим тегом </body>: **<script src="scripts/main.js"></script>**
- Он делает то же, что и элемент <link> для CSS - добавляет JavaScript на страницу, позволяя ему работать с ней.

- Теперь добавьте следующий код в файл main.js:
  var myHeading = document.querySelector('h1');
  myHeading.textContent = 'Привет Мир!';
- Убедитесь, что HTML и JavaScript файлы сохранены, и загрузите index.html в браузере.
- 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.
- 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, заголовок текста был изменен на "Привет Мир!" с помощью JavaScript. Мы сделали это с помощью вызова функции querySelector(), захватив ссылку на заголовок и сохранив ее в переменной 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) "Привет Мир!".
- Важно: Попробуйте вводить примеры строк кода в 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.
- Есть некоторые ограничения для имени переменной.

- Имя переменной может содержать только буквы, цифры или символы **$** (символ доллара) и _ (символ подчеркивания). Первый символ не должен быть цифрой.
- Называйте переменные английскими буквами, без пробелов.
- JavaScript чувствителен к регистру - myVariable отличается от переменной myvariable.
- После объявления переменной, вы можете присвоить ей значение: **myVariable = 'Bob';**
- Вы можете сделать обе эти операции в одной и той же строке: **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';**
- 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';**
- Вместо одинарных можно использовать двойные кавычки.

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, и не нуждаются в кавычках.
- ПРИМЕР: **var myVariable = true;**

ТИП ДАННЫХ: ARRAY

- Позволяет хранить несколько значений в одной ссылке.
- ПРИМЕР: **var myVariable = [1,'Bob','Steve',10];**
- Обратиться к элементу массива можно так: **myVariable[0]**. В квадратных скобках пишут номер элемента массива.

ТИП ДАННЫХ: OBJECT

- В принципе что угодно. Все в JavaScript является объектом, и может храниться в переменной. Имейте это в виду.
- ПРИМЕР: **var myVar = document.querySelector('h1');**

КОММЕНТАРИИ

- Вы можете поместить комментарии в JavaScript код:
  /*
  Все, что находится тут комментарий. */
- Если комментарий однострочный:
  // Это комментарий

ОПЕРАТОРЫ

- operator - математический символ, который производит результат, основанный на 2 значениях (или переменных).

ОПЕРАТОР: СЛОЖЕНИЕ/КОНКАТЕНАЦИЯ

- Используется для сложения чисел или склеивания строк.
- СИМВОЛ: **+**
- ПРИМЕР: **6 + 9** ; **alert("Hello " + "world!")**

ОПЕРАТОР: ВЫЧИТАНИЕ, УМНОЖЕНИЕ, ДЕЛЕНИЕ

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

ОПЕРАТОР ПРИСВАИВАНИЯ:

- Присваивает значение переменной.
- СИМВОЛ: **=**
- ПРИМЕР: **var myVariable = 'Bob';**

ТОЖДЕСТВЕННЫЙ ОПЕРАТОР:

- Делает проверку, если увидит, что два значения равны друг другу, то возвращает true/false (Boolean) результат.
- СИМВОЛЫ: **===**

- ПРИМЕР: **myVariable === 4**

ОПЕРАТОР: ОТРИЦАНИЕ, НЕРАВЕНСТВО

- Возвращает логически противоположное значение, которое этому предшествует; превращает true в false.
- Когда используется вместе с оператором равенства, 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;

ЧТО ЕЩЕ НАДО ЗНАТЬ:

- Существует намного больше операторов для изучения. Полный список в разделе [https://tinyurl.com/rbm7f8s].
- Смешивание типов данных может привести к неожиданным результатам при выполнении вычислений, поэтому будьте осторожны. Когда вы ссылаетесь на переменные правильно, вы получаете результаты, которые вы ожидаете.
- Например, введите **"35" + "25"** в консоль. Почему вы не получили результат, который вы ожидали? Потому, что кавычки превратили числа в строки, так что в итоге получилась конкатенация строк, а не сложение чисел. Если вы введете, 35+25, то получите правильный результат.

УСЛОВИЯ

- Условие - кодовая структура, которая позволяет проверить истинно ли выражение, а затем выполнить другой код в зависимости от результата.
- Самая распространенная форма условия if ... else. Пример:
- ```
  var iceCream = 'шоколадное';
  if (iceCream === 'шоколадное') {
    alert('Да, я люблю шоколадное мороженое!');
  } else {
    alert('Жаль, мое любимое - шоколадное...');
  }
  ```
  ```

- Выражение внутри if (...) - это проверка, которая использует тождественный оператор, чтобы сравнить переменную iceCream со строкой **'шоколадное'** 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 else.
- If может использоваться без части else:

```
if (iceCream === 'шоколадное') {
  alert('Да, я люблю шоколадное мороженое!');
}
```

ФУНКЦИИ

- Функции являются способом упаковки кода, который вы хотите использовать повторно. Всякий раз, когда вы хотите сделать что-то, вы можете просто вызвать функцию, а не постоянно переписывать весь код. Вы уже видели некоторые виды использования функций выше, например:
- **var myVariable = document.querySelector('h1');**
- **alert('hello!');**
- Функции, **document.querySelector** и **alert**, встроены для того, чтобы вы использовали их, когда это нужно.
- Если вы видите что-то, что выглядит как имя переменной, но имеет скобки - **()** - после него, скорее всего это функция.
- Функции часто принимают аргументы - данные, 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 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 эту функцию в консоли, затем попробуйте использовать новую функцию несколько раз:
- **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.

СОБЫТИЯ

- Для интерактивных веб-сайтов нужны события.
- События - структура, которая слушает то, что происходит в браузере, а затем позволяет запускать код в ответ на это.
- Наиболее очевидным является событие клика, которое вызывается браузером, когда мы щелкаем по чему-то мышью. Для демонстрации этого, попробуйте ввести следующую команду в консоли, а затем щелкните по текущей веб-странице:
  document.querySelector('html').onclick = function()
  { alert('Ой! Не щелкай по мне!');
  }
- Существует 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.
- **document.querySelector('html').onclick = function() {};** эквивалентно
  var myHTML = document.querySelector('html');
  myHTML.onclick = function() {};

СМЕНА ИЗОБРАЖЕНИЯ

- Давайте добавим еще одно изображение на сайт и добавим код для переключения между изображениями, когда по ним щелкнули.

- Найдите другое изображение, которые вы хотели бы показать на сайте. Убедитесь что оно такого же размера, как первое изображение или близкое к нему.
- Сохраните изображение в папку **images**.
- Перейдите в файл main.js и введите JavaScript:

```
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');
}
}
```

- Сохраните файлы и загрузите index.html в браузере. Теперь, когда вы щелкните по картинке, она должно измениться!
- Итак, мы сохраняем ссылку на элемент изображения в переменной myImage. Далее, мы создаем этой переменной обработчик события onclick с анонимной функцией. Теперь, каждый раз, когда на этот элемент изображения щелкнут:
- Мы получаем значение из атрибута **src** изображения.
- 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.

ПРИВЕТСТВИЕ

- Далее добавим код, 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 нужно в любое время.
- В index.html, добавьте строку перед элементом <script>:

<button>Изменить пользователя</button>

- 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:
  var myButton = document.querySelector('button');
  var myHeading = document.querySelector('h1');
- Теперь добавьте следующую функцию для установки персонализированного приветствия - она ничего не будет делать, но мы будем использовать ее позже:
  function setUserName() {
  var myName = prompt('Введите ваше имя.');
  localStorage.setItem('name', myName);
  myHeading.innerHTML = 'Привет, ' + myName;
  }
- Эта функция содержит функцию prompt() , которая вызывает диалоговое окно, немного похожее на alert(). **prompt()** просит ввести данные, и сохраняет их в переменной, после того как пользователь нажимает OK.
- В данном случае, мы просим пользователя ввести его имя. Далее, мы вызываем API **localStorage**, которое позволяет 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 заголовок в виде строки и имени пользователя.
- Затем добавляем блок if ... else, чтобы мы могли вызвать код инициализации, приложение устанавливает выполняет его, когда оно впервые загружается:
  if(!localStorage.getItem('name')) {
  setUserName();
  } else {
  var storedName = localStorage.getItem('name');
  myHeading.innerHTML = 'Привет, ' + storedName;
  }
- 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 заголовок в виде строки плюс имя пользователя, так же, как мы делали внутри setUserName().

- В заключение, установим обработчик события onclick на кнопку, чтобы при щелчке по ней запускалась функция setUserName(). Это позволяет пользователю задавать новое имя, нажав на кнопку:
  myButton.onclick = function() {
  setUserName();
  }
- Теперь, когда вы в первый раз посетите сайт, мы попросим вас ввести имя пользователя, а затем дадим персональное сообщение.
- Затем вы можете изменить имя, в любой момент, нажав на кнопку. В качестве дополнительного бонуса, имя сохранится после закрытия сайта, поэтому персонализированное сообщения все еще будет там, когда вы откроете сайт снова!

ПОЛЕЗНЫЕ ССЫЛКИ

- Этот документ на сайте Mozilla на русском [https://tinyurl.com/rctyvsm].
- Этот документ на сайте Mozilla на украинском [https://tinyurl.com/utqyofp].
- Firefox инспектор [https://tinyurl.com/tx8u4ye]
- Firefox консоль [https://tinyurl.com/wgnlzw8]
- Chrome DOM инпектор [https://tinyurl.com/wmtxza7]
- Chrome консоль [https://tinyurl.com/yah3hejf]