# Инструменты создания приложения

# HTML

Язык гипертекстовой разметки (Hypertext Markup Language, HTML) - это код, который используется для структурирования веб-содержимого.

HTML не является языком программирования; это язык разметки, и используется, чтобы сообщать браузеру, как отображать веб-страницы, которые посещают пользователи. HTML может быть сложным или простым, в зависимости от дизайна сайта. HTML состоит из ряда элементов, которые используется, чтобы вкладывать или оборачивать различные части контента, чтобы заставить контент отображаться или действовать определенным образом. Ограждающие теги могут сделать слово или изображение ссылкой на другую страницу, могут сделать слова курсивом, сделать шрифт больше или меньше и так далее. Пример тега можно увидеть на (Рис. 1).

Рис. 1 Конструкция элемента html

* Открывающий тег (Opening tag): состоит из имени элемента, в данном случае «p», заключенного в открывающие и закрывающие угловые скобки. Открывающий тег указывает, где элемент начинается или начинает действовать, в данном случае - где начинается абзац;
* Закрывающий тег (Closing tag): это то же самое, что и открывающий тег, за исключением того, что он включает в себя косую черту перед именем элемента. Закрывающий элемент указывает, где элемент заканчивается, в данном случае - где заканчивается абзац;
* Контент (Content): это контент элемента, который в данном случае является просто текстом;
* Элемент (Element): открывающий тег, закрывающий тег и контент вместе составляют элемент.

Элементы также могут иметь атрибуты (Рис. 2).

Рис. 2 Атрибут html

Атрибуты содержат дополнительную информацию об элементе, которая показывается 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:

* Пробел между ним и именем элемента (или предыдущим атрибутом, если элемент уже имеет один или несколько атрибутов);
* Имя атрибута, за которым следует знак равенства;
* Значение атрибута, заключенное с двух сторон в кавычки.

Также можно располагать элементы внутри других элементов - это называется вложением. Если необходимо выделить слово, то его можно заключить в элемент <strong>, который указывает, что слово должно быть сильно акцентированно:

«<p>Моя кошка <strong>очень</strong> раздражена.</p>»

Однако, необходимо убедитесь, что элементы правильно вложены: в примере выше был открыт первым элемент <p>, затем элемент <strong>, потом должны закрыться сначала элемент <strong>, затем <p>. Приведенное ниже неверно:

«<p>Моя кошка <strong>очень раздражена.</p></strong>»

Некоторые элементы не имеют контента, и называются пустыми элементами. Например, элемент <img>:

«<img src="images/firefox-icon.png" alt="Моё тестовое изображение">»

Элемент содержит два атрибута, но не имеет закрывающего тега </img>, 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.

Вот так отдельные элементы объединяются в целую HTML страницу.

«<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Моя тестовая страница</title>

</head>

<body>

<img src="images/firefox-icon.png" alt="Моё тестовое изображение">

</body>

</html>»

Здесь находится:

* «<!DOCTYPE html>» - доктайп. В прошлом, 1991/1992, данный тег должен был выступать в качестве ссылки на набор правил, которым HTML страница должна была следовать. HTML, означал автоматическую проверку ошибок и другие полезные вещи.;
* «<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 необходимо включить в HTML страницу, но не являющегося контентом, который показывается пользователям страницы. К ним относятся такие вещи, как ключевые слова и описание страницы, которые будут появляться в результатах поиска, CSS стили контента, кодировка и другое;
* «<body></body>» - элемент <body>. В нем содержится весь контент, который показывается пользователям, когда они посещают страницу, будь то текст, изображения, видео, игры, проигрываемые аудиодорожки;
* «<meta charset="utf-8">» - этот элемент устанавливает UTF-8 кодировку документа, которая включает в себя большинство символов из всех языков. Благодаря этому документ может обрабатывать любой текстовый контент, который в него вложен. Нет причин не устанавливать её, так как это может помочь избежать некоторых проблем;
* «<title></title>» - элемент <title>. Этот элемент устанавливает заголовок для страницы, который является названием, появляющимся на вкладке браузера загружаемой страницы, и используется для описания страницы, когда происходит добавление ее в закладки - избранное.

# CSS

Как и HTML, CSS на самом деле не является языком программирования. Это не язык разметки - это язык таблицы стилей. Это означает, что он позволяет применять стили выборочно к элементам в документах HTML. Например, чтобы выбрать все элементы абзаца на HTML странице и изменить текст внутри них с черного на красный (Рис. 3).

«p {

color: red;

}»

Чтобы подключить css файл к html нужно прописать в <head> строчку: «<link href="styles/style.css" rel="stylesheet" type="text/css">»



Рис. 3 Структура css

* Селектор (Selector). Имя HTML элемента в начале набора правил. Он выбирает элемент(ы) для применения стиля (в данном случае, элементы p). Для стилизации другого элемента, необходимо просто изменить селектор;
* Объявление (Declaration). Единственное правило, например, color: red; указывает, какие из свойств элемента стилизовать;
* Свойства (Properties). Способы, которыми можно стилизовать определенный HTML элемент (в данном случае, color является свойством для элементов <p>). В CSS выбирается, какие свойства нужно затронуть в правиле;
* Значение свойства (Property value). Справа от свойства, после двоеточия, есть значение свойства, который выбирает одно из множества возможных значений для данного свойства (существует множество значений color, 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 нужно использовать точку с запятой (;), чтобы отделить каждое объявление от следующего.
  1. JavaScript

JavaScript это кросс-платформенный, объектно-ориентированный, скриптовый язык. JavaScript - это маленький и легковесный язык; он спроектирован не для создания законченных, полноценных приложений, а для легкого встраивания в другие продукты и приложения, например, в веб браузеры. Внутри основного приложения, JavaScript может соединяться с объектами этого приложения и осуществлять программный контроль над ними.

Ядро JavaScript содержит набор базовых объектов, например, Array, Date и Math, и набор элементов языка: операторов, управляющих структур, и выражений. Ядро JavaScript может быть расширено для различных целей с помощью дополнений, например:

* Client-side JavaScript расширяет ядро языка предоставляя объекты управления браузером (Navigator или другой веб-браузер) и его Document Object Model (DOM). Например, клиентские расширения позволяют приложению размещать элементы в HTML-форме и реагировать на пользовательские события, такие как щелчок мыши, ввод данных в форму и навигация по страницам;
* Server-side JavaScript расширяет ядро языка предоставляя объекты для запуска JavaScript на сервере. Например, server-side расширения позволяют приложению соединяться с реляционной базой данных, обеспечивать непрерывность информации между вызовами приложения, или выполнять операции над файлами на сервере.

Используя функции JavaScript LiveConnect, можно позволить коду на Java и JavaScript общаться между собой. Можно создать экземпляр объекта Java из JavaScript и получить доступ к его публичным методам и свойствам. Из Java, можно получить доступ к объектам, свойствам и методам JavaScript.

Впервые JavaScript был использован в браузерах Netscape.

JavaScript и Java схожи в некоторых отношениях, но принципиально отличаются в других. Язык JavaScript напоминает Java, но не имеет статической типизации и строгой проверки типов. В основном, JavaScript следует большей части синтаксиса Java в выражениях, именованиях и основного потока управления конструкциями, что стало причиной, почему он был переименован из LiveScript в JavaScript.

В отличии от системы компилированных классов в Java, построенной на объявлениях, JavaScript поддерживает систему исполнения, основанную на небольшом количестве типов данных, представляющих числовые, логические и строковые значения. JavaScript обладает моделью объектов на базе прототипов вместо более общей модели объектов на базе классов. Модель объектов на базе прототипов делает возможным динамическое наследование; то есть, то, что унаследовано, может различаться для отдельных объектов. JavaScript также поддерживает функции без каких-либо специальных декларативных требований. Функции могут быть свойствами объектов, выполняться как слабо типизированные методы.

По сравнению с Java, JavaScript - это язык с очень свободной формой языка. Вам не надо объявлять переменные, классы или методы. Не стоит беспокоиться, на счет того: является ли метод public, private или protected, нет необходимости реализовывать интерфейсы. Переменные, параметры и возвращаемые функциями типы не являются явно типизированными.

Java - язык программирования на основе классов, предназначенный для быстрого выполнения и безопасности типов. Безопасность типов означает, например, что нельзя взять тип Java integer и привести его к типу object reference или получить доступ к закрытой памяти изменяя байт-код Java. Ориентированная на классы модель Java означает, что программы состоят исключительно из классов и их методов. Наследование классов и строгая типизация в Java обычно требуют тесно связанную иерархию объектов. Эти требования делают программирование на Java более сложным чем на JavaScript.

По духу JavaScript происходит от нескольких небольших, динамически типизированных языков программирования, таких как HyperTalk и dBASE. Эти скриптовые языки предлагают инструменты программирования для гораздо более широкой аудитории благодаря более простому синтаксису, специальной встроенной функциональности и минимальным требованиям для создания объектов.

* 1. Electron

Создание программ под ПК стало возможным, после появления библиотек подобных Electron JS.

Приложения в вебе становятся всё мощнее, здесь прогресс на лицо. Тем не менее значительную долю разработки занимают стандартные приложения, которые имеют полный доступ к физическому оборудованию ПК. Уже сегодня есть возможность объединить обе технологии и написать десктопное приложение на известных языках веб-программирования, вроде HTML, JS и Node.js. Это всё можно поместить в исполняемый файл, который можно использовать на Mac OS X, Windows, Linux.

Приложение на Electron – это обычная Node.js программа, поэтому ей нужно добавить npm. Благо, это выполняется предельно легко.

Следует запустить терминал и находясь в каталоге целевого проекта выполнить команду:

npm install

В результате появится папка с названием node\_modules, в которой установлены все нужные зависимости для программы. Дальше стоит ввести ещё одну команду.

npm start

После неё приложение запустится в новом окне. Нужно заметить, что в нём будет исключительно верхнее меню.

Несложно заметить, что приложение включается крайне неудобно для рядового пользователя. Это лишь один из способов запуска, который скорее подходит для разработчика, чем пользователя. После упаковки программы, пользователь сможет включить приложение стандартным способом – дважды кликнуть по ярлыку.

Разработка программы - это создание главного JS файла, а также HTML и CSS файлов, содержащих все стили и разметку для страниц программы.

Существует ещё один важный момент, который помогает достичь целевого пользователя. Вам нужно запереть всё содержимое в исполняемый файл, как раз его и можно включить двойным кликом. Важно создать уникальный дистрибутив под каждую ОС: Windows, OS X, Linux. Как раз в этом и пригодится Electron Packager.

Здесь следует уделить внимание тому, что в готовый файл также добавятся ресурсы, это обусловлено платформой Node JS, и обрезанная копия webkit веб-обозревателя. На выходе должен получиться файл весом около 50 Мб. Это весьма большой вес для обычного приложения в несколько строк. Вопрос с весом программы теряет актуальность при разработке крупных приложений со сложными алгоритмами работы.

Главное из значимых отличий от NW JS сводится к тому, что в NW.js входной файл - HTML, в то время как в Electron - JavaScript-файл. Таким образом Electron дарит больше возможностей по контролю. На его основе можно создать приложение с несколькими окнами, и настроить перенос данных между ними.

* 1. React

React - это JavaScript-библиотека для создания пользовательских интерфейсов. Именно библиотека, а не фреймворк. React часто называют фреймворком, но это ошибка. Во-первых, его использование ни к чему не обязывает, не формирует «фрейм» проекта. Во-вторых, React выполняет единственную задачу: показывает на странице компонент интерфейса, синхронизируя его с данными приложения, и только этой библиотеки в общем случае недостаточно для того, чтобы полностью реализовать проект.

Вскоре после появления React и подобные ему решения (Vue.js, Svelte) практически захватили мир фронтенда: потому что они помогают решать проблемы, основываясь на идее декларативного программирования, а не на императивном подходе.

Декларативный подход состоит в описании конечного результата.

При императивном подходе описываются конкретные шаги для достижения конечного результата.

Оказалось, что декларативный подход отлично подходит для создания интерфейсов, и он прижился в сообществе. Этот подход работает не только в вебе: сравнительно недавно компания Apple представила фреймворк SwiftUI, основанный на тех же принципах.

Возможности данной библиотеки:

* С её помощью можно построить интерфейс из отдельных компонентов, которые легко поддерживать;
* Она добавляет удобный слой абстракции, избавляя от необходимости работать с DOM напрямую;
* React — это уже не новая библиотека, за ней стоит крупная компания Facebook и большое сообщество разработчиков. Поэтому она отлично протестирована, регулярно поддерживается и стабильно обновляется, а переход к новым версиям проходит максимально гладко;
* Благодаря сообществу у React хорошо проработанная документация и большой опыт, накопленный в статьях, курсах и конференциях. Это значительно облегчает не только изучение библиотеки новичками, но и поиск ответов на всевозможные вопросы в процессе работы;
* На Гитхабе можно найти готовые React-компоненты почти на все случаи жизни. А если их нет, но есть нужные независимые библиотеки, то можно поискать интеграцию либо сделать её самостоятельно;
* В React-сообществе со временем сформировались определённые подходы и соглашения по организации проекта, кода, решению частых проблем. Для разработчиков это значит, что нужно тратить меньше времени на обсуждение некоторых договорённостей — можно использовать готовые и уже устоявшиеся;
* Хотя в проекте, скорее всего, будет налажен процесс сборки с помощью webpack, parcel, rollup или другого упаковщика, стоит иметь в виду, что это необязательно для использования React. Работая с этой библиотекой, пишется на чистом JS, и никакие диалекты HTML, CSS или JS изучать не нужно. Конечно, с React почти всегда используется JSX, но и это тоже [опционально](https://ru.reactjs.org/docs/react-without-jsx.html);
* React - это проект с открытым исходным кодом. Благодаря этому его можно безопасно использовать даже в коммерческих приложениях.

Однако при использовании React есть**особенности, которые важно учитывать:**

* React увеличивает размер приложения, которое нужно загрузить пользователям (~40 kB для пакетов React и React-dom).
* Загруженный код нужно выполнить в браузере: для пользователя это означает, что приложение будет запускаться медленнее. Подробнее про это можно почитать в [материале](https://v8.dev/blog/cost-of-javascript-2019) «The cost of JavaScript in 2019» от команды V8 (перевод: «[Цена JavaScript в 2019 году](https://habr.com/ru/company/ruvds/blog/459296/)»).
* Вместе с виртуальным DOM возникают свои накладные расходы: во-первых, по времени выполнения (сравнение виртуальных деревьев происходит не моментально); во-вторых, по памяти, ведь виртуальные деревья нужно где-то хранить и не в единственном экземпляре. С увеличением количества элементов на странице эти расходы растут, что может стать реальной проблемой на мобильных устройствах. Поэтому при изучении React важно уделить внимание способам оптимизации рендеринга приложения – необходимые инструменты для этого есть в самой библиотеке.
* Средний порог входа в React. Чтобы начать разрабатывать на React, программист должен не только изучить саму библиотеку, но и привыкнуть к используемой парадигме. Про это есть хорошая [статья](https://ru.reactjs.org/docs/thinking-in-react.html) «Философия React» на официальном сайте.
  1. Redux

Redux - это менеджер состояний. Чаще всего его используют с React, но его возможности не ограничиваются одной этой библиотекой. Хотя в React есть собственный метод управления состояниями, он плохо масштабируется. Перемещение состояния вверх по дереву работает для простых приложений, но в более сложных архитектурах изменение состояния производится через свойства (props). Ещё лучше делать это через внешнее глобальное хранилище.

Библиотека Redux — это способ управления состоянием приложения. Она основана на нескольких концепциях, изучив которые, можно с лёгкостью решать проблемы с состоянием.

Redux идеально использовать в средних и крупных приложениях. Им стоит пользоваться только в случаях, когда невозможно управлять состоянием приложения с помощью стандартного менеджера состояний в React или любой другой библиотеке.

Простым приложениям Redux не нужен.

В Redux общее состояние приложения представлено одним объектом JavaScript — state (состояние) или state tree (дерево состояний). Неизменяемое дерево состояний доступно только для чтения, изменить ничего напрямую нельзя. Изменения возможны только при отправке action (действия).

Действие (action) - это JavaScript-объект, который лаконично описывает суть изменения.

Единственное требование к объекту действия — это наличие свойства type, значением которого обычно является строка.

В простом приложении тип действия задаётся строкой. По мере разрастания функциональности приложения лучше переходить на константы.

При запуске действия обязательно что-то происходит и состояние приложения изменяется. Это работа редукторов.

Редуктор (reducer) — это чистая функция, которая вычисляет следующее состояние дерева на основании его предыдущего состояния и применяемого действия.

Чистая функция работает независимо от состояния программы и выдаёт выходное значение, принимая входное и не меняя ничего в нём и в остальной программе. Получается, что редуктор возвращает совершенно новый объект дерева состояний, которым заменяется предыдущий.

* 1. Node js

Подружить две технологии, react и electron, можно несколькими способами, например, создать проект CRA (create react app) и установить туда с помощью пакетного модуля npm или yarn «голый» electron. Или же можно пойти от обратного и создать проект electron forge и туда установить react.

Но в любом случае разработка не была бы возможна без технологии node js, который позволяет вытащить js из браузера.

Node.js это программный пакет, основанный на JavaScript-движке Google V8. Он включает уровень абстракции платформы – библиотеку libuv, и базовую библиотеку, которая сама главным образом написана на JavaScript. Кроме того, стоит отметить, что создатель Node.js Райан Даль намеревался создавать сайты, работающие в реальном времени с технологией push, вдохновившись такими приложениями, как Gmail. В Node.js он предоставил разработчикам инструмент для работы с парадигмой неблокирующего событийно-ориентированного ввода/вывода.

Основная идея Node.js заключается в использовании неблокирующего событийно-ориентированного ввода/вывода, чтобы оставаться легковесным и эффективным при работе с приложениями, обрабатывающими большие объемы данных в реальном времени и работающими на распределенных устройствах.

При разговоре о Node.js очень важно не упустить одну важную деталь. Это встроенная поддержка управления пакетами с помощью инструмента NPM, который по умолчанию идет в любой установке Node.js. Принцип модулей NPM схож с принципом Ruby Gems – это набор повторно используемых компонентов, находящихся в открытом доступе. Их можно легко установить из сетевого репозитория, с поддержкой управления версиями и зависимостями.

То есть если подвести итоги, то node js позволяет с помощью библиотеки electron js, создать настольное приложение, интерфейс которому будет рисовать react js.

* 1. Webpack

Webpack - это сборщик модулей JavaScript с открытым исходным кодом. Он создан в первую очередь для JavaScript, но может преобразовывать внешние ресурсы, такие как HTML, CSS и изображения, если включены соответствующие загрузчики. Webpack принимает модули с зависимостями и генерирует статические ресурсы, представляющие эти модули.

Webpack принимает зависимости и генерирует граф зависимостей, позволяющий веб-разработчикам использовать модульный подход для разработки своих веб-приложений. Его можно использовать из командной строки или настроить с помощью файла конфигурации с именем webpack.config.js. Этот файл используется для определения правил, плагинов и т. д. для проекта. Webpack сильно расширяем с помощью правил, которые позволяют разработчикам писать задачи, которые они хотят выполнять при объединении файлов.

Для использования webpack требуется Node.js.

* 1. JSX

JSX — это код JavaScript с комбинацией синтаксиса xml. Тег JSX имеет имя тега, атрибуты и дочерние элементы, которые делают его похожим на xml.

React использует JSX для шаблонов вместо обычного JavaScript. Нет необходимости использовать его, однако, ниже приведены некоторые плюсы, которые идут с ним:

* Это быстрее, потому что он выполняет оптимизацию при компиляции кода в JavaScript;
* Это также типобезопасно, и большинство ошибок могут быть обнаружены во время компиляции;
* Это облегчает и ускоряет написание шаблонов, если есть представление о HTML.
  1. Visual Studio Code

Visual Studio Code - редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

Visual Studio Code основан на Electron и реализуется через веб-редактор Monaco, разработанный для Visual Studio Online.

Visual Studio Code поддерживает ряд языков программирования, подсветку синтаксиса, IntelliSense, рефакторинг, отладку, навигацию по коду, поддержку Git и другие возможности. Многие возможности Visual Studio Code недоступны через графический интерфейс, зачастую они используются через палитру команд или JSON-файлы (например, пользовательские настройки). Палитра команд представляет собой подобие командной строки, которая вызывается сочетанием клавиш.

Visual Studio также позволяет заменять кодовую страницу при сохранении документа, символы перевода строки и язык программирования текущего документа.

С 2018 года появилось расширение Python для Visual Studio Code с открытым исходным кодом. Оно предоставляет разработчикам широкие возможности для редактирования, отладки и тестирования кода.

На март 2019 года посредством встроенного в продукт пользовательского интерфейса можно загрузить и установить несколько тысяч расширений только в категории «programming languages» (языки программирования).

Также расширения позволяют получить более удобный доступ к программам, таким как Docker, Git и другие. В расширениях можно найти линтеры кода, темы для редактора и поддержку синтаксиса отдельных языков.

Visual Studio Code имеет встроенный emmet

Emmet (ранее Zen Coding) — набор плагинов для текстовых редакторов, которые в некоторой степени ускоряют написание кода HTML, XML, XSL, а также кода на некоторых других языках. Проект был начат Вадимом Макеевым в 2008 году и активно разрабатывается Сергеем Чикуенком начиная с 2009 года, а также сообществом пользователей Zen Coding на основе идеи Zen Coding 2.0. C 2015 года за опыт пользователей в проекте отвечает Микаэл Гелецян.

Инструменты Zen Coding были вложены в популярные текстовые редакторы, а некоторые плагины, разработанные командой Zen Coding и другими, сделаны независимыми друг от друга. Zen Coding написан на JavaScript и поэтому хорошо переносим между платформами.

Zen Coding является программным обеспечением с открытым исходным кодом и распространяется под лицензией MIT.

Visual Studio Code собирает данные об использовании (телеметрию) и отправляет их в Microsoft, но эта функция является отключаемой (опция File > Preferences > Settings, «telemetry.enableTelemetry»). Хотя предоставление данных не является обязательным, и вы можете отказаться от передачи персональных данных, некоторые возможности, такие как персонализация, использующие такие данные, будут недоступны для отключения. Данные могут передаваться контролируемым филиалам Microsoft, дочерним компаниям и правоохранительным органам в соответствии с заявлением о конфиденциальности.

Данная IDE помимо встроенного emmet’а поддерживает большое количество плагинов, позволяющие сильно экономить время. Например, Prettier.

Prettier - это средство для форматирования кода, которое нацелено на использование жёстко заданных правил по оформлению программ. Оно форматирует код автоматически

Вот какие возможности и особенности Prettier позволяют говорить о полезности этого инструмента:

* Приведение в порядок существующей кодовой базы. Подобное, с помощью Prettier, можно выполнить буквально одной командой. Ручная обработка больших объёмов кода займёт гораздо больше времени. Например, затраты труда, необходимые для того, чтобы вручную отформатировать 20000 строк кода;
* Prettier легко внедрить. Prettier использует «усреднённый», наименее спорный подход к стилю при форматировании кода. Так как проект это опенсорсный, многие внесли в него вклад, улучшая его и сглаживая острые углы.
* Prettier позволяет сосредоточиться на написании кода, а не на его форматировании. Многие просто не осознают того, как много времени и сил тратится на форматирование кода. Использование Prettier позволяет не думать о форматировании, а заниматься вместо этого программированием.

Или вот еще один очень полезный плагин «LiveServe», который поднимает локальный сервер с возможностью автоматического обновления страниц в браузере при их изменении. Запустить сервер можно посредством нажатия на иконку «Go Live» в строке состояния или выбора пункта «Open with Live Server» в контекстном меню области редактора.

* 1. Git и Github

Git - распределённая система контроля версий, которая даёт возможность разработчикам отслеживать изменения в файлах и работать над одним проектом совместно с коллегами. Она была разработана в 2005 году Линусом Торвальдсом, создателем Linux, чтобы другие разработчики могли вносить свой вклад в ядро Linux. Git известен своей скоростью, простым дизайном, поддержкой нелинейной разработки, полной децентрализацией и возможностью эффективно работать с большими проектами.

Подход Git к хранению данных похож на набор снимков миниатюрной файловой системы. Каждый раз, когда сохраняется состояние проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.

Преимущества Git:

* Бесплатный и open-source. Можно бесплатно скачать и вносить любые изменения в исходный код;
* Небольшой и быстрый. Выполняет все операции локально, что увеличивает его скорость. Кроме того, Git локально сохраняет весь репозиторий в небольшой файл без потери качества данных;
* Резервное копирование. Git эффективен в хранении бэкапов, поэтому известно мало случаев, когда кто-то терял данные при использовании Git;
* Простое ветвление. В других системах контроля версий создание веток - утомительная и трудоёмкая задача, так как весь код копируется в новую ветку. В Git управление ветками реализовано гораздо проще и эффективнее.

Теперь пора разобраться, что такое GitHub и как он работает с Git.

Как было разобрано выше, Git - это инструмент, позволяющий реализовать распределённую систему контроля версий.

GitHub - сервис онлайн-хостинга репозиториев, обладающий всеми функциями распределённого контроля версий и функциональностью управления исходным кодом — всё, что поддерживает Git и даже больше. Также GitHub может похвастаться контролем доступа, багтрекингом, управлением задачами и вики для каждого проекта.

Git-репозиторий, загруженный на GitHub, доступен с помощью интерфейса командной строки Git и Git-команд. Также есть и другие функции: документация, запросы на принятие изменений (pull requests), история коммитов, интеграция со множеством популярных сервисов, email-уведомления, эмодзи, графики, вложенные списки задач, система упоминаний, похожая на ту, что в Twitter, и т.д.

Кроме GitHub есть другие сервисы, которые используют Git, - например, Bitbucket и GitLab.

Чтобы лучше понимать, что такое Git и как он работает, нужно ещё знать, что такое система контроля версий.

Системы контроля версий (СКВ, VCS, Version Control Systems) позволяют разработчикам сохранять все изменения, внесённые в код. При возникновении проблем они могут просто откатить код до рабочего состояния и не тратить часы на поиски ошибок.

СКВ также позволяют нескольким разработчикам работать над одним проектом и сохранять внесённые изменения независимо друг от друга. При этом каждый участник команды видит, над чем работают коллеги