GeekBrains

Vue CLI

Vue.js

Оглавление

Введение

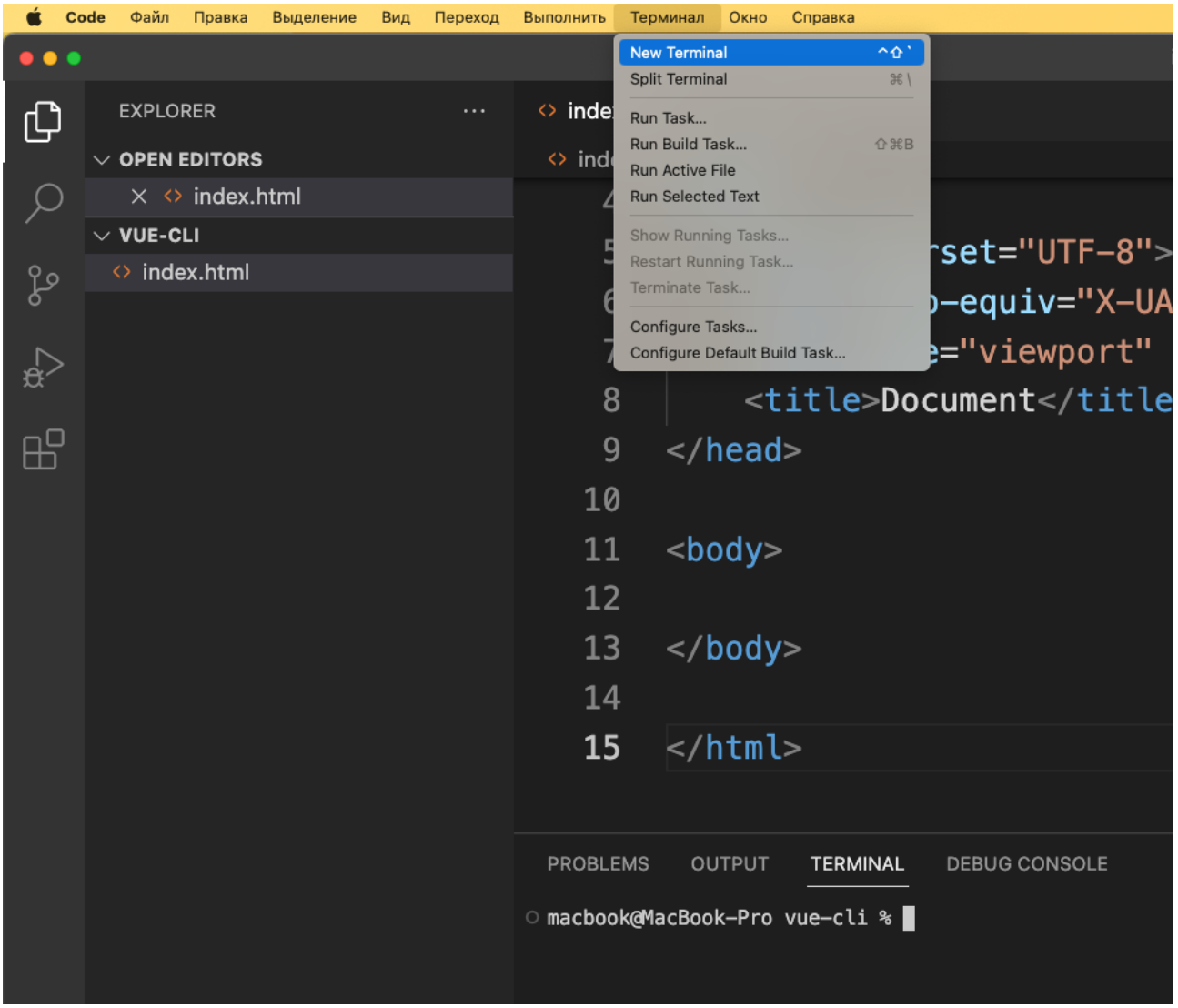
Давайте подумаем как много информации мы уже знаем, мы уже знаем что такое Vue.js и для каких целей он используется, рассмотрели разные виды подключения данного фреймворка, конечно уже умеем работать с данными и методами внутри Vue.js ну и конечно уже знаем основы работы с компонентами, но что мы сразу можем заметить, что наш код становится очень объемным, особенно когда компонент станет намного больше, тут мы будем путаться в огромном количестве файлов и конечно всё нужно упорядочить, но ведь для того чтобы упорядочить большое количество данных нам потребуется много времени, а самое главное что каждый программист будет делать это по своему, получается что наш код невозможно будет читать и поддерживать, вот тут к нам на помощь и приходят Vue CLI который должен решить все вышеперечисленные проблемы и конечно расставить всё по своим местам. Поэтому давайте разбираться с тем что это за инструмент такой и конечно как же его установить

Установка Vue CLI

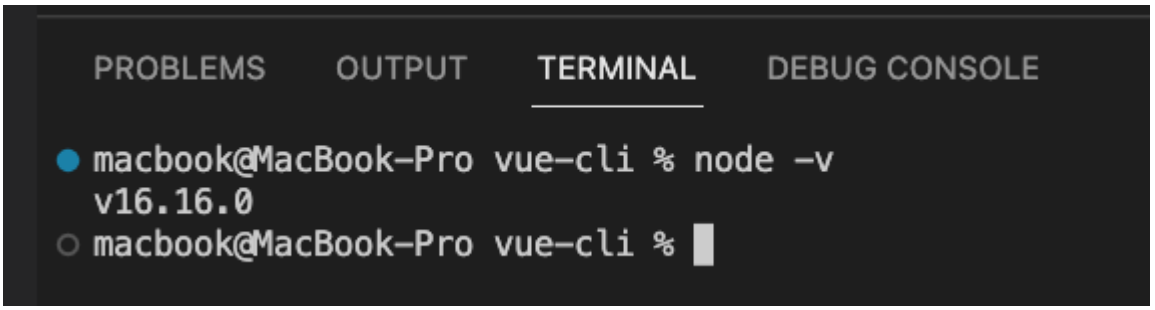
Vue CLI это набор инструментов для командной строки, значительно упрощающий жизнь разработчикам, которые хотят запустить новый проект на Vue.js. Он позволяет с помощью одной команды сгенерировать основу (boilerplate) для нового

приложения, подключить плагин и даже создать новое приложение с помощью графического интерфейса в браузере (vue ui).
Vue CLI из коробки поддерживает Babel, TypeScript, ESLint, PostCSS, а также unit и end-to-end тесты.

Первым делом вам потребуется убедиться что у вас установлен node.js для этого вам необходимо открыть командную строку в редакторе кода Visual Studio Code и открыть терминал



Далее вводим команда у терминале, который открылся в нижней части редактора кода



Рекомендуется использовать последнюю стабильную версию, так что если у вас показывает версию node.js значит у вас всё установлено и работает корректно

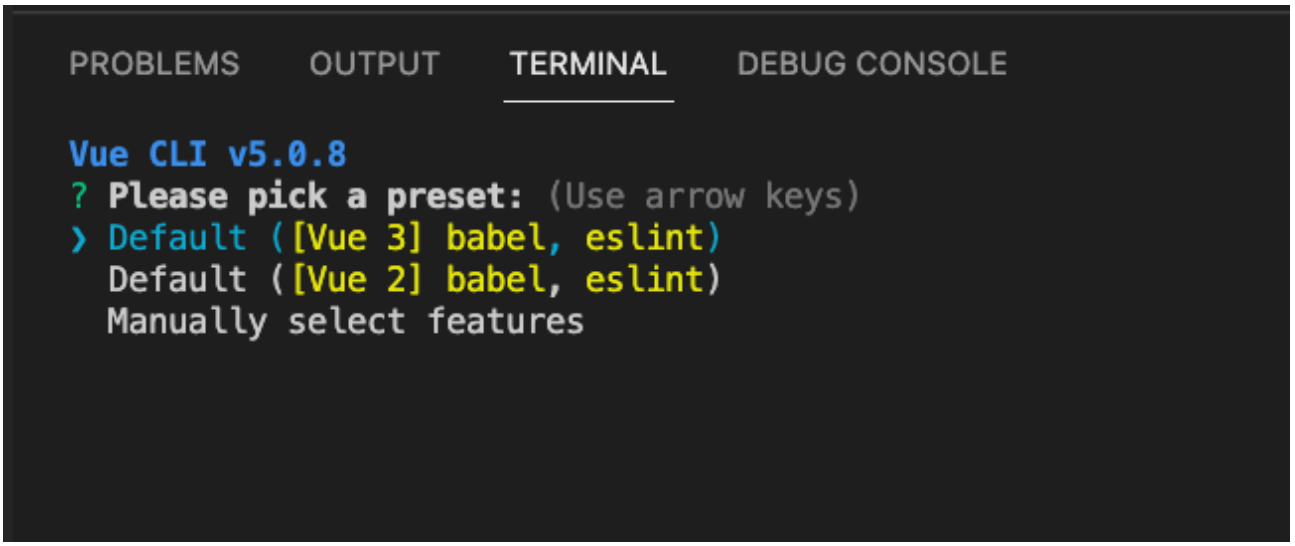
Важно: Лучше создать отдельную папку, в которой вы будете работать с vue cli чтобы ничего не удалить или не сломать при установке

Для установки vue-cli используем команду установки NPM пакета:

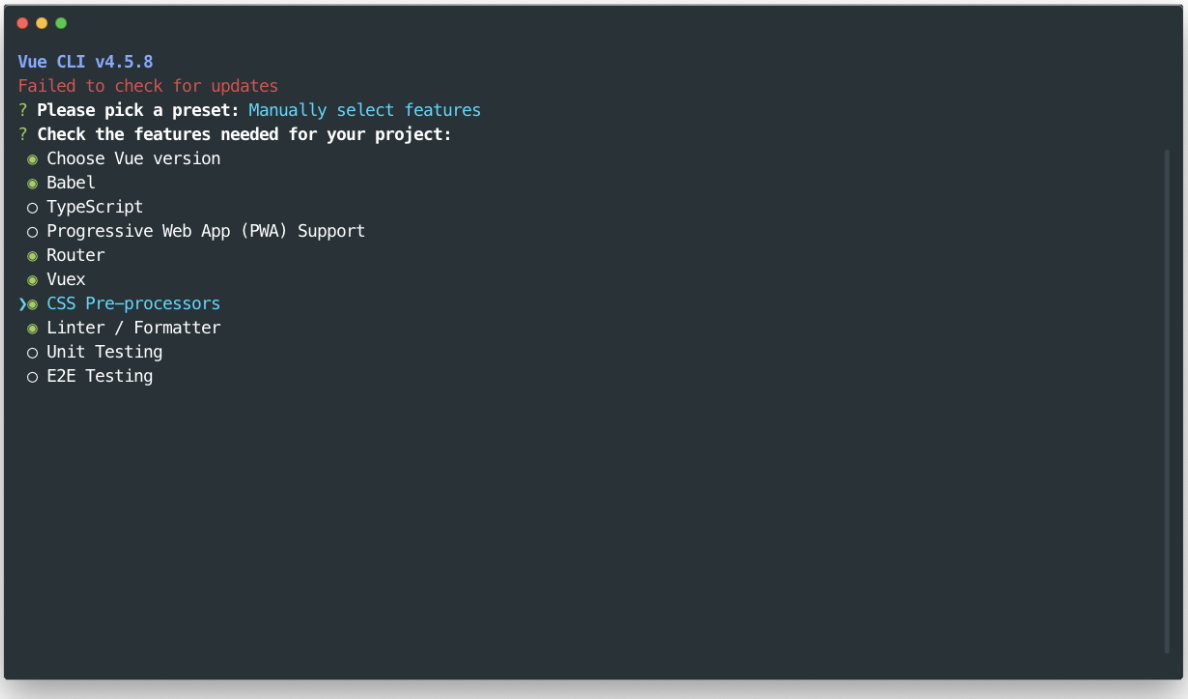
```
npm install -g @vue/cli
```

Создание нового проекта Vue

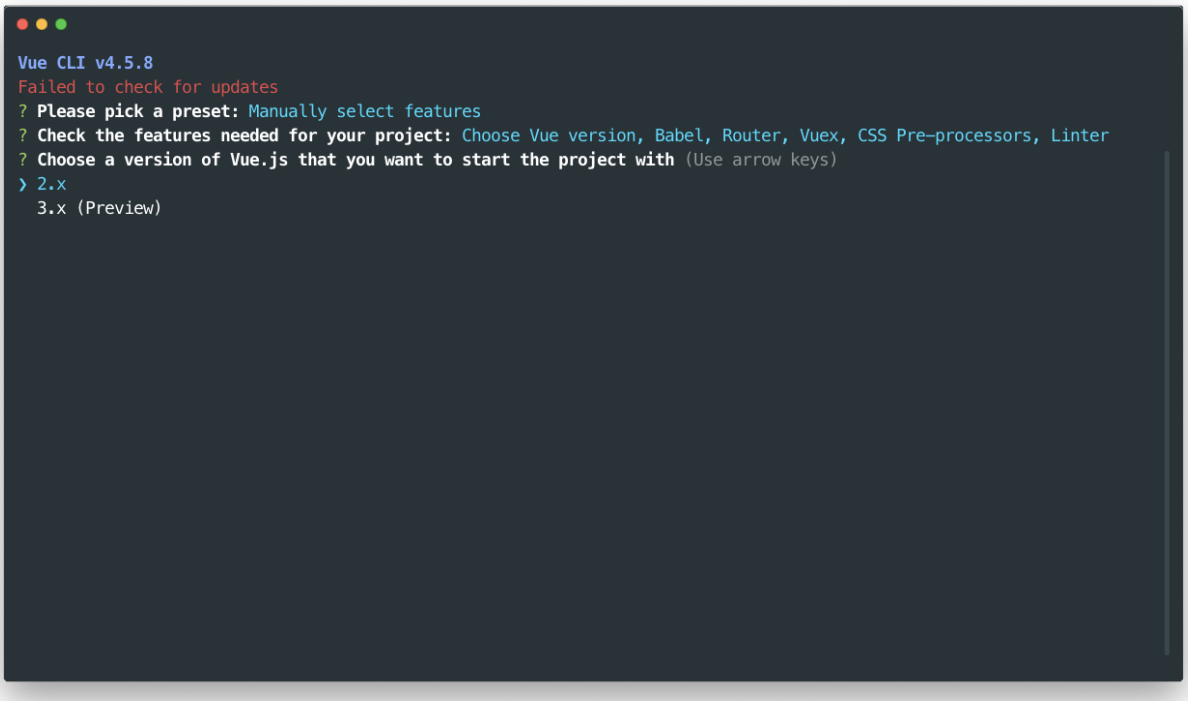
Теперь для создания нового проекта во Vue запустим команду **vue create first-project**. Появится вопрос, о том, какие инструменты мы хотим использовать в проекте. Выберем вариант “Manually select features”, чтобы увидеть, какие возможности у нас есть.



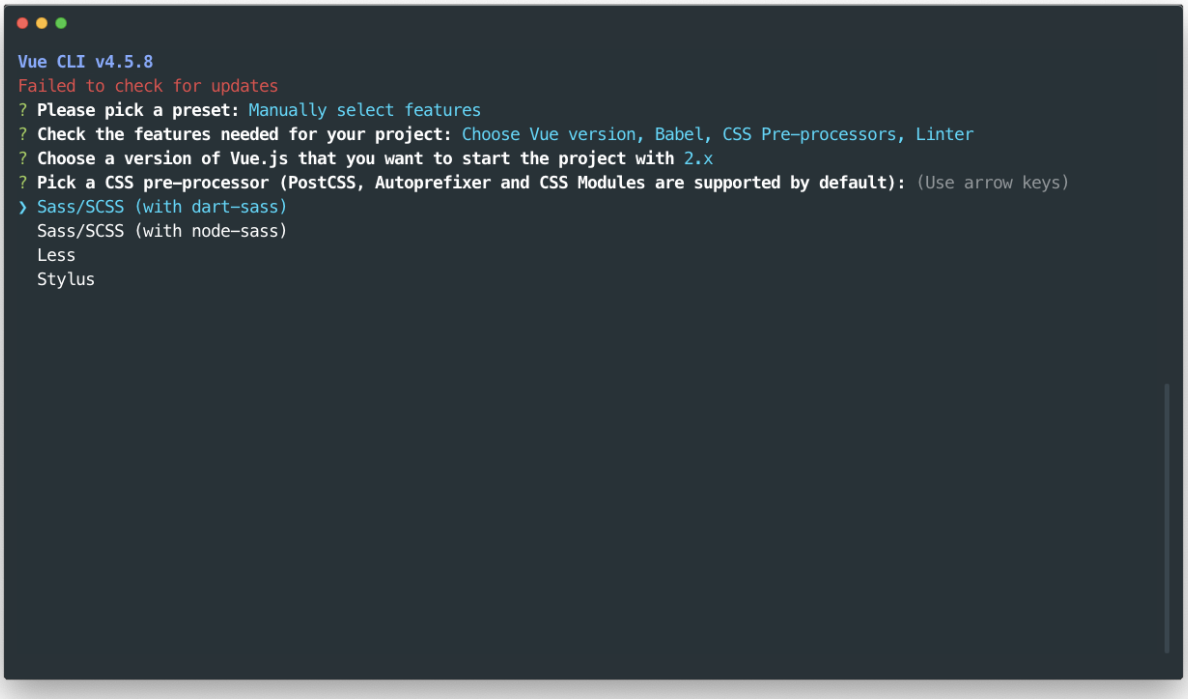
Выберем часть опций - в частности: Babel, Linter и CSS pre-processors.



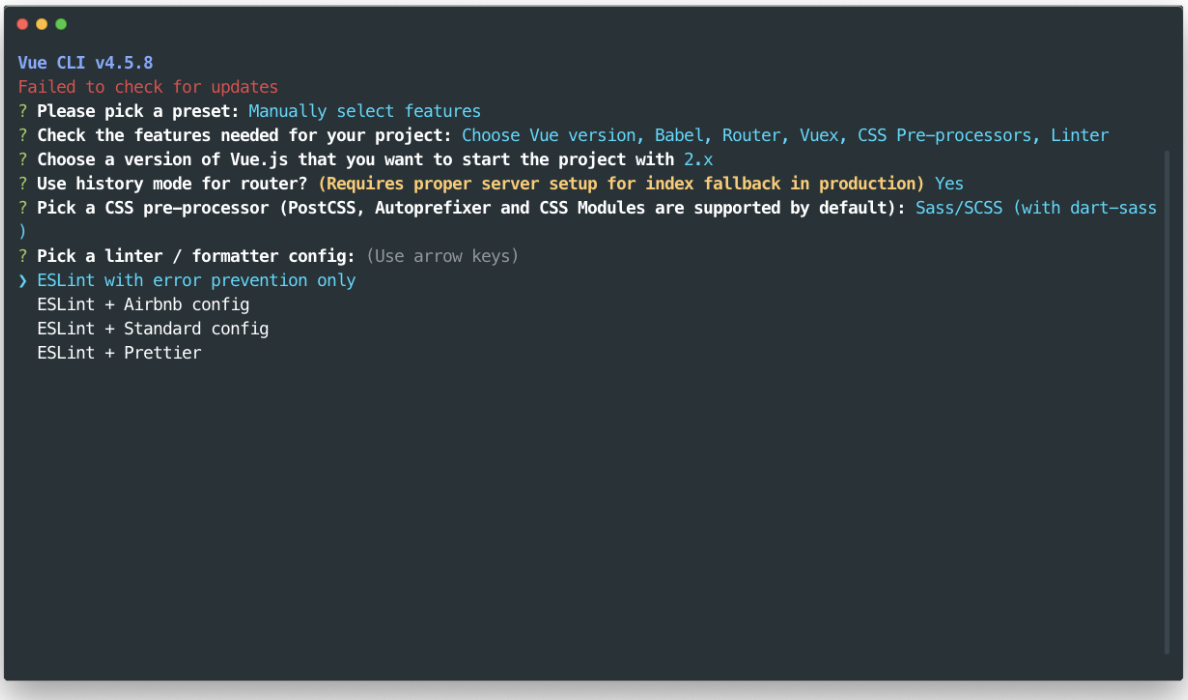
Так как нас сегодня уже доступна 3я версия Vue, следующий экран предлагает выбрать, что именно установить. Выберем пока версию 2.x



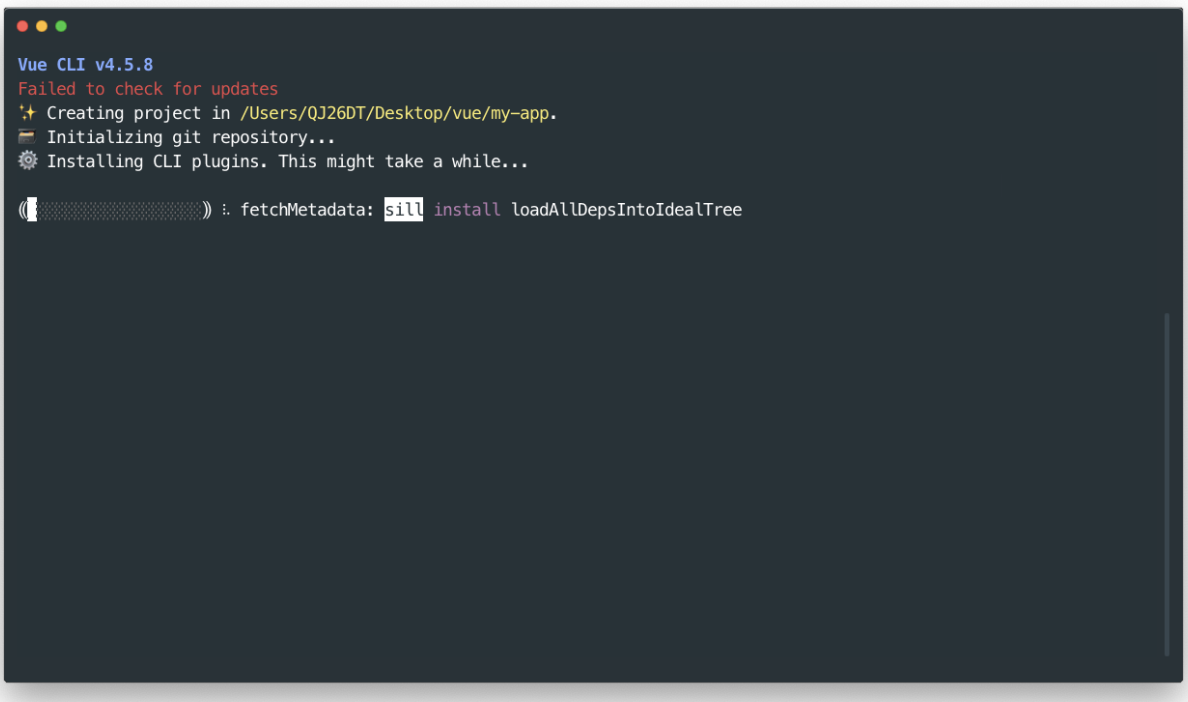
Затем из вариантов CSS pre-processor’ов выбираем Sass/SCSS. (pre-processor’ы CSS нужны для того, чтобы сделать код стилей более кратким и удобочитаемым. Этот код во время сборки приложения конвертируется в обычный код CSS. Делается это именно с помощью препроцессоров).



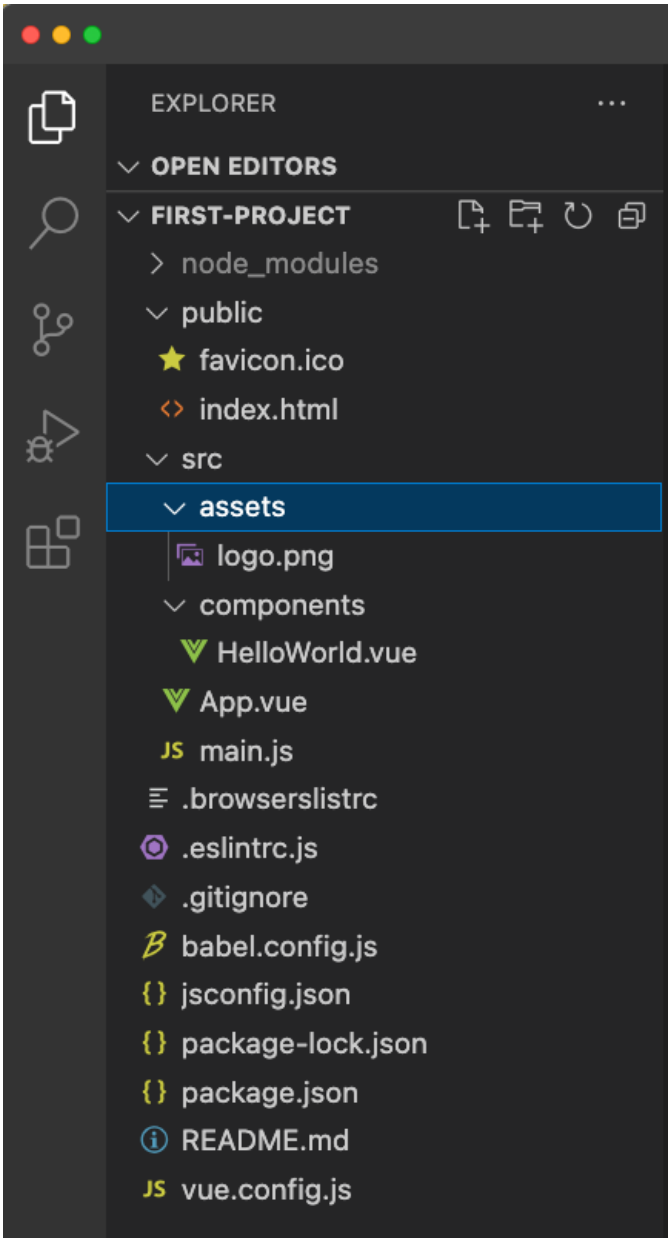
Затем, в шаге настройки линтера/форматтера выбираем ESLint with error prevention only. После этого выбираем опцию Lint on save и хранение конфигураций а отдельных файла



Получив от нас первоначальные инструкции, vue-cli начнет инициализацию проекта, установку необходимых модулей и генерацию структуры проекта из папок и файлов. В конце должно появиться сообщение об успешной установке проекта, а также инструкция дальнейших действий.



В результате наших действий, у нас создалась папка с именем будущего проекта (first-project) содержащую в себе стандартную структуру файлов.



Разбор структуры проекта Vue

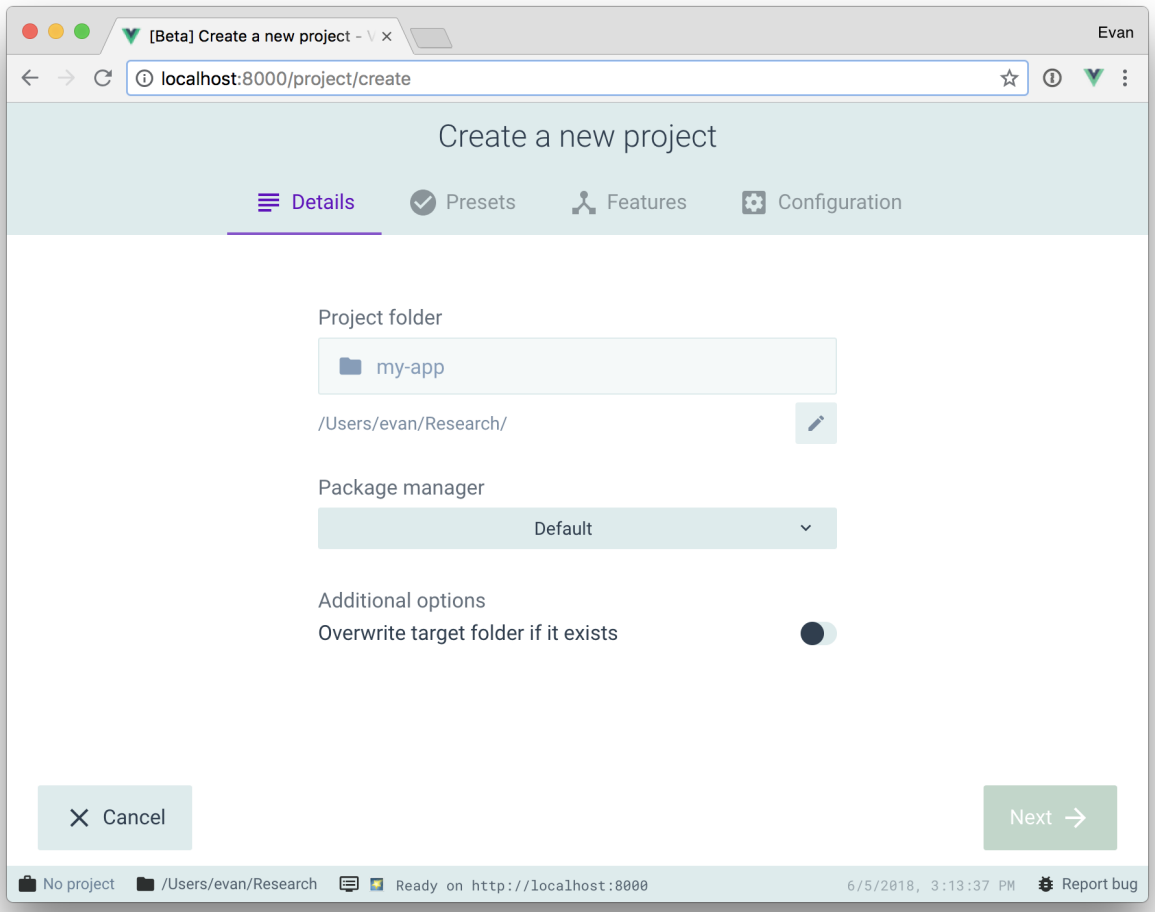
Рассмотрим файлы в созданном проекте, и что находится в каждой из папок:

- node_modules - в этой папке лежат коды и исполняемые файлы установленных внешних (не наших) модулей, ее не стоит открывать и менять. Также ее не стоит коммитить в репозиторий (поэтому она по умолчанию была добавлена в файл .gitignore)
- public - здесь лежат файлы, которые используются в проекте, но при сборке будут скопированы без изменений. Исключением является только файл

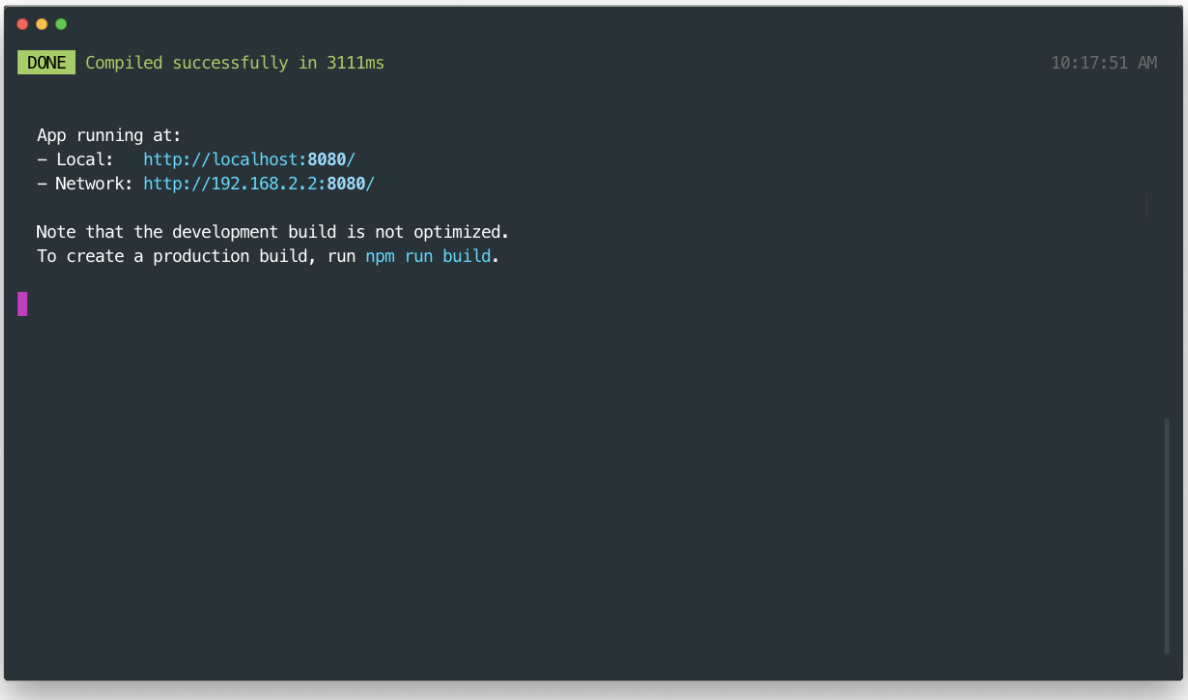
index.html - во время компиляции в него добавятся тэги <script>, подключающие трансформированные файлы .js.

- src - папка со исходным кодом проекта. Здесь будет лежать вся логика нашего приложения
- src/assets - здесь обычно лежат изображения и css файлы со стилями, которые используются в приложении
- src/components - папка, содержащая отдельные Vue компоненты в виде файлов с расширением vue
- src/main.js - входной файл проекта во Vue, мы будем его изменять, когда захотим что-нибудь подключить к нашему приложению, например - плагин.
- src/App.vue - корневой компонент приложения, все остальные компоненты будут добавляться внутри него
- src/components/HelloWorld.vue - пример обычного компонента, который мы подключаем

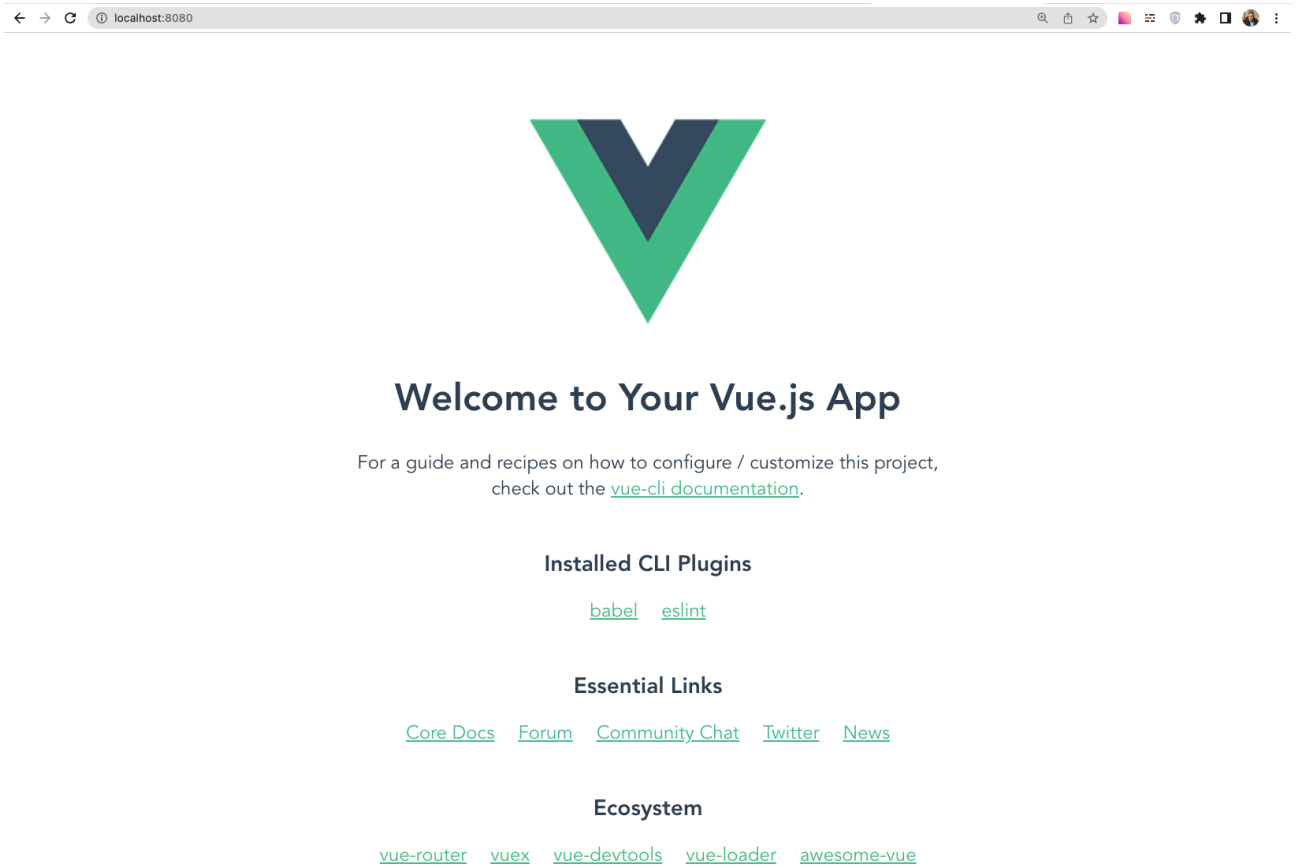
Если вы из тех, кто не очень любит работать в командной строке, тогда вы вполне можете воспользоваться инструментом с аналогичными возможностями, имеющим графический интерфейс Vue UI - больше про него можно прочитать тут - https://cli.vuejs.org/ru/guide/creating-a-project.html#%D0%B8%D1%81%D0%BF%D0%BEn%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-gui



Для запуска приложения перейдем в его папку (в терминале cd first-project) и запустим его с помощью команды npm run serve. Спустя несколько секунд после успешной сборки мы должны увидеть похожее сообщение:



Перейдем в браузер и убедимся, что приложение запущено и работает. Для этого в адресной строке впишем http://localhost:8080 . В браузере должны отображаться страница с текстом Welcome to Your Vue.js App.



Сейчас приложение запущено в режиме live-reload, то есть, если мы изменим его код - изменения отобразятся на странице.

Разберем файлы приложения более детально:

- main.js - здесь создается инстанс приложения Vue, который привязывается к html элементу с идентификатором app. Мы можем найти этот div элемент в файле index.html. Функциональный параметр render использует единственный подгруженный компонент - корневой компонент App.vue.
  Обратите внимание на свойство el - с помощью него мы говорим Vue, в каком месте ему искать шаблон в DOM дереве. Vue автоматически найдет элемент и превратит все его содержимое в шаблон. С этого момента внутри шаблона нам станут доступны данные нашего приложения
- App.vue - В шаблоне находится корневой элемент div и два элемента внутри, один из которых - компонент HelloWorld. Его подключение производится в js-части с помощью кода

```
components: {
   HelloWorld,
}
```

Такая запись является ничем иным, как как сокращенным объявлением свойства ES6. Мы подключаем компонент в приложение и будем использовать его по имени свойства. Если нам необходимо использовать в шаблоне компонент под другим именем, то мы можем сделать это:

```
components: {
   "MySecondComponent": HelloWorld
}
```

- HelloWorld.vue - Шаблонная часть здесь гораздо больше, а часть с кодом js содержит неизвестное нам ранее свойство props, которое используется для передачи данных дочерним компонентам (HelloWorld является дочерним компонентом для компонента App).. Эти данные затем можно использовать в коде шаблона ({{ msg }}). Более подробно о передаче данных между компонентами мы поговорим с вами на следующих уроках.

Вернемся в код App.vue и изменим передаваемое значение пропса msg "Welcome to Your Vue.js App" на “Здесь могла быть ваша реклама”. После изменения файла изменения должны отобразиться в браузере, так как приложение запущено в режиме разработки с опцией живой перезагрузки.

```
<template>
 <div id="app">
   <img alt="Vue logo" src="./assets/logo.png">
   <HelloWorld msg="Здесь могла быть ваша реклама" />
 </div>
</template>


<script>
import HelloWorld from './components/HelloWorld.vue'


export default {
 name: 'App',
 components: {
   HelloWorld
 }
}
</script>
```

Как результат

Здесь могла быть ваша реклама

For a guide and recipes on how to configure / customize this project,
check out the vue-cli documentation.

Installed CLI Plugins

babel eslint

Essential Links

Core Docs Forum Community Chat Twitter News

Ecosystem

vue-router vuex vue-devtools vue-loader awesome-vue

Посмотрим еще раз более детально на файл App.vue - он, как и любой файл с расширением .vue это по умолчанию - Single File Vue Component или сокращенно SFC. Он имеет состоит из трех основных частей:

- Шаблон - разметка HTML внутри тегов <template></template>
- JS логика - код внутри тегов <script></script>
- Стили - css/sass код внутри тегов <style></style>

В коде компонента мы отделяем шаблон от логики, в действительности же, под капотом, Vue создает реактивную связь между данными и их представлением.

Благодаря этому, нам больше нет необходимости работать с DOM напрямую, Vue позаботится о максимально производительном обновлении DOM дерева в зависимости от изменений наших данных.

Давайте посмотрим на пример простого компонента:

```
<template>
 <div id="app">
   {{ message }}
 </div>
</template>


<script>
export default {
 name: 'MyComponent',
 data: () => ({
   message: 'Привет, Vue!'
 })
}
</script>
```

Свойство data - это функция, которая возвращает объект с начальными данными компонента (стейт, если вы работали с React), которые Vue автоматически делает реактивными. Что это значит? Мы поговорим об этом подробнее позже, но на данный момент достаточно упомянуть, что Vue создает список зависимостей от каждого свойства объекта data. С этого момента он автоматически будет перерисовывать те части шаблона, которые ссылаются на какое-то свойство в этом

объекте. При этом, перерисовка будет выполняться “по-умному”, с помощью виртуального DOM.

Помимо data у компонента есть ещё ряд свойств, например:

- name - для указания имени компонента;
- components - для декларации дочерних компонентов используемых в шаблоне;
- methods - для описания функций которые могут работать с данными компонента - как методы JS класса
- computed, watch - для специальных реактивных свойств;
- create, beforeDestroy - события жизненного цикла компонента;

На следующих занятиях мы подробно разберем каждый из них.

Пока важно понять что совокупность свойств объекта компонента Vue ещё называется Options API, то есть по сути список опций с помощью которых мы можем описать наш компонент.

Итоги урока

Мы подробно разобрали установку и запуск Vue CLI, разобрали возможности редактирования и конечно же рассмотрели все созданные элементы, не стоит бояться большого количества папок или файлов, ведь вам не нужно будет работать со всеми элементами сразу, плюс стандартный шаблон содержит в себе много дополнительного кода, который на старте можно смело удалить и тогда Vue CLI станет для вас самым оптимальным решением и уже использовать стандартное подключение через CDN точно не потребуется. Теперь остается только разбить проект на блоки и создать все необходимые компоненты, а как это сделать мы узнаем на следующем уроке

Используемая литература

1. Официальный сайт Vue.js - ссылка
2. Документация Vue CLI - ссылка