A Quick Tour of the generated content

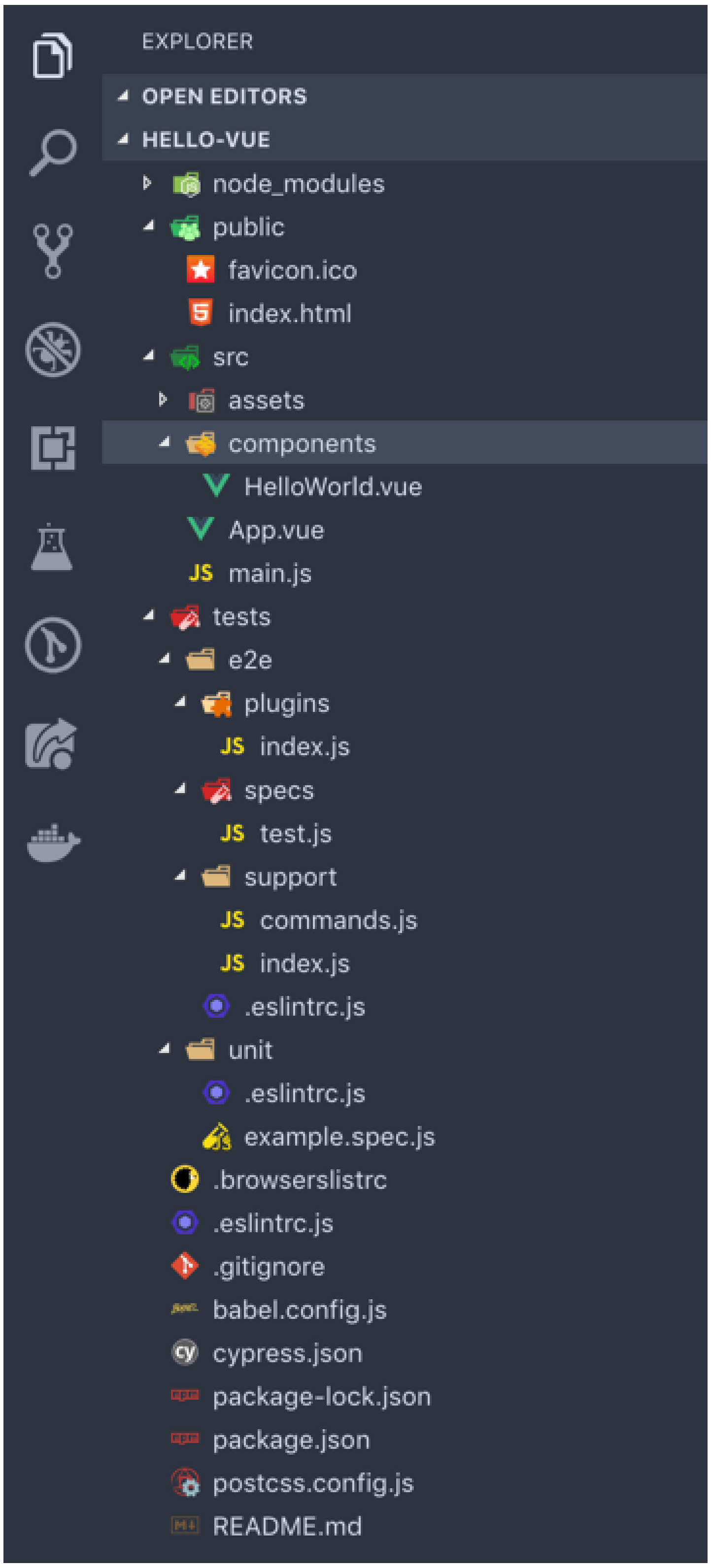
This document explains what each of the files and folders are used for in a typical Vue application. The sample project was created using the Vue CLI:

```
vue create hello-vue
```

Typically, you're presented with the following options during project creation:

- Manually Select Features
  - Babel
  - Linter / Formatter
  - Unit Testing
  - E2E Testing
- Linter / Formatter Config
  - ESLint with error prevention only
- Pick additional lint features:
  - Lint on save
- Pick a unit testing solution:
  - Mocha + Chai
- Pick a E2E testing solution:
  - Cypress (Chrome Only)
- Where do you prefer placing config for Babel, PostCSS, ESLint, etc.?
  - In dedicated config files
- Save this as a preset for future projects?
  - Yes or No (The default is No, but it might be a good idea to save this as a preset)

The overall structure of the generated project is shown below:



```
EXPLORER
OPEN EDITORS
HELLO-VUE
  node_modules
  public
    favicon.ico
    index.html
  src
    assets
    components
      HelloWorld.vue
    App.vue
    main.js
  tests
    e2e
      plugins
        index.js
      specs
        test.js
      support
        commands.js
        index.js
      .eslintrc.js
    unit
      .eslintrc.js
      example.spec.js
  .browserslistrc
  .eslintrc.js
  .gitignore
  babel.config.js
  cypress.json
  package-lock.json
  package.json
  postcss.config.js
  README.md
```

The node_modules directory

The `node_modules` directory contains all of the npm packages needed for your application to run. Every time you run the command `npm install some-package`, the package `some-package` downloads and is stored in this folder. From here, you can import dependencies into your Vue.js project or reference them manually in an HTML page.

The public directory

The public directory contains your `index.html` file that everything gets bootstrapped and injected into. For example, if you need to add a dependency like Bootstrap 4 into your application via a CDN, you can add its respective tags in this file. However, it’s best practice to always import them via the `node_modules` folder.

In this directory, you also have the `favicon.ico` icons file, an `img` directory to store static assets like app images, and a `manifest.json` file that contains some basic meta information about your application.

Note: the `img` folder and `manifest.json` file are optional and may not be added during the project creation.

The src directory

This is the most important directory in the generated project. In this folder, all of your single file components, stylesheets, assets, and more are stored here. This is where you'll work in 99.99% of the time during your application’s development.

Read the list below to learn more about the `src` directory and see what's new in Vue CLI 3:

- `assets/`: Stores all of your application’s assets like images, CSS, and scripts.
- `components/`: Stores all of the application’s components. These components are single `.vue` files, which contain `<template>`, `<script>`, and `<style>`.
- `App.vue`: The single component in which all other views and components get injected into. This is a great place to add global components that should be shared across the app like `Header.vue` and `Footer.vue`.
- `main.js`: This is your single Vue Instance in which the `App.vue`, routes, and all their components get injected into.

The tests directory

The `tests` directory contains all of the tests for your application. Depending on what you selected while creating your application, you might have end-to-end tests, unit tests, or both.

e2e Tests

Contains end-to-end test related files.

- `plugins/`: Plugins enable you to tap into, modify, or extend the internal behavior of Cypress.
- `specs/`: test spec files
- `support:/`
  - `index.js`: This is a great place to put global configuration and behavior that modifies Cypress.
  - `commands.js`: This example shows you how to create various custom commands and overwrite existing commands.
- `.eslintrc.js`: ESLint configuration for e2e tests

Unit tests

Contains unit test related files.

- `.eslintrc.js`: ESLint configuration for e2e tests
- `example.spec.js`: an example unit test spec

Everything else

These are the rest of the files in the root of the project:

- `.browserlistrc`: The config to share target browsers and Node.js versions between different front-end tools
- `.eslintrc.js`: ESLint configuration
- `.gitignore`: sensible defaults for Git
- `babel.config.js`: babel configuration
- `cypress.json`: Cypress (e2e testing) configuration
- `package-lock.json`: It describes the exact tree that was generated, such that subsequent installs are able to generate identical trees, regardless of intermediate dependency updates
- `package.json`: The NPM package meta file that contains all the build dependencies and build commands
- `postcss.config.js`: post css configuration (Vue CLI uses PostCSS internally)
- `README.md`: default readme file

NPM

npm is the world's largest software registry. The documentation states that, "open source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as well" ("About npm").

ESLint

ESLint is, "an open source JavaScript linting utility originally created by Nicholas C. Zakas in June 2013. Code linting is a type of static analysis that is frequently used to find problematic patterns or code that doesn't adhere to certain style guidelines. There are code linters for most programming languages, and compilers sometimes incorporate linting into the compilation process" ("About ESLint").

Babel

According to the documentation, "Babel is a toolchain that is mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript in current and older browsers or environments" ("What is Babel?").

Babel can:

- Transform syntax
- Polyfill features that are missing in your target environment (through @babel/polyfill)
- Source code transformations (codemods)

Check out these videos for inspiration.

Cypress

Cypress is a next generation front-end testing tool built for the modern web. According to the documentation, the tool "addresses key pain points developers and QA engineers face when testing modern applications. The tool makes it possible to set up tests, write tests, run tests, and debug tests" ("Why Cypress?", 2020).

The documentation adds that, "Cypress is most often compared to Selenium; however Cypress is both fundamentally and architecturally different. Cypress is not constrained by the same restrictions as Selenium. This enables you to write faster, easier, and more reliable tests" ("Why Cypress?", 2020).

Sources:

"About ESLint." Retrieved from https://eslint.org/docs/about/.

"About npm." Retrieved from https://docs.npmjs.com/about-npm/.

"What is Babel?" Retrieved from https://babeljs.io/docs/en/.

"Why Cypress?" Retrieved from https://docs.cypress.io/guides/overview/why-cypress.html#In-a-nutshell.