WATS-4000: Up and Running

Submitted on 04/08/2018 by Kelly Colht

Read through the site code and note the following:

- What directories do you see? How do you interpret their names?
- Where is the Vue app defined? (Which file?)
- What is listed in package.json?
- What do you see in the build and config directories?

The top-level folders in the project directory are called build, config, src, and static. The build folder contains configuration files for webpack, a build tool which packages a projects assets and dependencies into a production ready bundle. I'm not sure what everything in each of the contained files does, but know that together they manage the building of dependency graphs, loaders for pre-processing multiple file types into modules, destinations for bundles generated by webpack, and plugins to extend the capabilities of webpack. To meet the goals of different developmnet stages, there are shared and separate settings for production and development builds.

The main configuration files for the project are enclosed in the config folder. The index.js file provides settings for both a development build of the project served locally and the production build.

Running npm install at the root of the project directory adds a node_modules folder to contain dependencies specified in the top-level file, package.json. In addition to the named dependencies, any dependencies of those dependencies (and so forth) are also added.

The src folder contains the files related to the app itself. components is where reusable aspect of the project reside and can be referenced from and assets is used to contain and organize assets for the site separate from the main app files.

The static folder is empty save for a .gitkeep file. My understanding is that this file exists only because Git cannot track an empty folder and adding a .gitignore file would be a little misleading.

The project directory also includes a few top-level files. I've already mentioned the package.json file listing project dependencies. This file also includes references to build scripts and other metadata like project name, description, and version.

There is also an index.html file which is the "front page" of the app when it is built (the div in this file with the "app" id pulls in the application from src/App.vue).

There is a JavaScript compiler called babel in this project which uses a plugin called istanbul. I am not sure why this is entered under the env.test object.

There is a .postcssrc.js file which is pretty empty; I don't know much about what it's for other than it having something to do with css.

A .gitignore file is at the toplevel and specifies to Git which folders, files, and file-types not to track. The listed include the entire node_modules folder (which can be easily reproduced based on the package.json file), and some others I'm not sure about.

The last file to mention is .editorconfig which looks to be a helpful file for whatever text editor is used for the project to keep spaces, tabs, white space, etc. standardized.

Run npm run build –report and take a look at both the webpage that comes up and the output in the console. Consider the following questions:

- What are you looking at in the “build report” that pops up in the web browser?
- Can you tell which files are the largest in the project?
- Does the custom code of the app (look for the blue box) make up the largest of the filesize? If not, what creates the most bulk in terms of file downloads?
- What do you see in the console output? How do you interpret that information?
- When you’re finished you can exit the build report by typing ctrl-c to quit.

npm run build --report starts the Webpack Bundle Analyzer which generates and analyzes a test build of the app and then produces a report.

The report is displayed as a treemap with some labels that on first look resemble the names of files I came across in the previous step along with some I don’t recognize. On closer examination, nested blocks hierarchically represent different folders and files produced during the test build process. Hovering over each block displays three different sizes (stat, parsed, and gzipped) and the target’s path.

On the right side of the treemap are folders and files exported by webpack into the static folder. To the left are folders and files appearing in the node_modules folder of the finished, test build. These are the vendor files which are required by the app but were not created as part of the app.

I think the purpose of separating vendor vs. app files out is to allow the report to distinguish between the total size of the application when installed, with dependencies included; the size of the app when shipped without dependencies, the small blue part to the right; and the size taken up in the total application by the dependencies, the larger tan part to the left. For this particular app, the size of dependencies was many times larger than the core app itself.

Seeing the finished test build visualized this way in the report helped me better better understand how the build process works through finding out what makes it into the final build. A conclusion I’ve come to is that requiring dependencies in your project comes at a cost.

Look at the directories in your project again and notice that there is a new one called dist.

- Explore the dist directory. What do you see?
- Do you see the filenames of the static files? What seems odd about those filenames?
- Do you see the contents of your JS and CSS files? What has happened to those contents?
- Describe (in words or with a flowchart/diagram) what happens when the npm run build command is executed to the best of your ability.

Generating the test build created a new folder called dist in the project directory. There are several .js and .map files which include hashes specific to the test build and are familiar from what was printed to the console after the run command. The prefixes for these files are app, vendor, and manifest. On opening these files I notice they’ve been uglified, I’m guessing for the purposes of minimizing filesize. I don’t recognize much of what’s in these files but assume its a production-build of the app bundled for distribution!