Development Tools

Mari Good

CS 233JS

Topics

- JavaScript Development Tools
  - node.js
  - npm
  - webpack
- 2 applications
  - ToDoList
    - Introduces local storage
    - Reviews HTML5 form validation
  - Bookmarker

Our First Application

- The next example we're going to do is the ToDoList. Let me show you what it does.
- I've given you the starting files on github.
  - index.html contains all of the html for the application.
  - styles.css contains all of the css
  - index.js contains a bazillion comments that are intended to help us write the code for the application.
- You can, of course, just watch as we build this first application together. You can even ask a classmate for his/her/their code for the application. BUT you'll want to make sure that you understand what you're doing and why. The lab for this week includes the ToDoList and then the BookMarker application that uses the same concepts and skills.

Modern JS Development Tools

- Modern browsers support ES6+ features reasonably well. If you run the apps that we've written so far in Chrome, Firefox they will behave exactly as expected.
- http://kangax.github.io/compat-table/es6/ gives detailed information about browser support for ES6+ features
- BUT most professional developers use a variety of JS tools, one of which is designed to convert newer features in the code to ES5 which is fully supported by browsers.

Node.js

- Node.js is a JS runtime this that let's you run JS outside of the browser.
  - It is often used to create command line tools that support all kinds of tools that are used as part of the JavaScript "ecosystem"
  - It is also frequently used to write server-side code, most frequently web services or apis that are consumed by front end js application. We'll see this in action later in the term.
  - It can be downloaded from https://nodejs.org/en/.
    - The installation process is a little different in each operating system.
    - I'll assume that you've already installed node on your machine.
    - You can tell if node is already installed on a machine if you run node –v from the command line and it returns a version number.

NPM

- Node comes with a tool called npm, node package manager.
  - It's a command line tool that helps you create and manage modules that are used in your JS applications.
- npm init
  - Creates a file called package.json. It's a text file that contains all kinds of information about the node modules that are part of your application.
- npm install *somenodemodule*
  - Downloads a copy of the somenodemodule into the node_modules folder in your application folder. It also adds a line to your package.json file for the installed module.
  - npm install by itself reads the package.json file and downloads all of the node modules listed into the application directory.

Node Modules You Should Know

- I'm going to use several node modules in this application. You'll use lots more as the term progresses.
  - babel/core and babel/cli - "transpiles" code into an earlier version of JS. It will monitor our src folder for changes in our JS and automatically "transpile" it
  - babel-preset-env – specifically translates new JS code into ES5
  - html-webpack-plugin – copies your html file into the dist (dist is where the translated files will be written) folder and adds the script element to it automatically
  - copy-webpack-plugin – copies files from one folder to another. In our case we want to move the css file from src to dist. You might also move image files.
  - webpack – manages the process of using all of these node modules as you're writing code
  - webpack-dev-server – runs a little web server on your machine so that your app behaves during development like it will when deployed. It also "watches" your js code for changes and transpiles and loads your code in the browser automatically as it changes.

Setting Up Your Development Environment

- Get the starting files in the usual way.
  - Accept the invitation in moodle
  - Copy the url of your remote repo from github
  - Clone the repo to make a local copy of the repo on your machine
- I could show you how to set all of this up manually but the process is very error prone. Instead, I've given you a
  - package.json file – lists all of the node modules you'll need
    - npm install will download all of the node modules to your machine. Do this now.
  - webpack.config.js – tells webpack how to work with all of the node modules you've installed. I won't talk about the details of this file this week but I will in the next lab. By the end of the term you'll be able to read the file.

npm Scripts

- The scripts section of the package.json allows you to create “macros” that run command line tools for you.

```
"scripts": {
  "webpack": "webpack",
  "build": "webpack --config webpack.config.js",
  "watch": "webpack serve --open"
}
```

Our Scripts

- npm run webpack – will ask webpack to use the webpack.config.js file to transpile our code and put it in the dist folder. It will also make a copy of the html file and the css files in the dist folder.
- npm run build – does the same thing as run webpack but specifies the name of the configuration file explicitly.
- npm run watch – starts up the development server so that it “watches” your code for changes. It also opens a new browser window with your app loaded.

Build Our Code

- Type either npm run build or npm run webpack from the command line. It should
  - Make a new folder called dist
  - Copy the index.html file into dist
  - Copy the css folder and its contents into dist
  - Transpile our code and create a file called index.bundle.js in dist
  - Edit the html file to add a script element that links to the index.bundle.js file
- Verify that all of that happens on your machine.

Start the Development Server

- Type either npm run watch from the command line. It should
  - Transpile our code and store a copy in memory.
  - Start the dev server so that it runs our app on port 8080
  - Open a browser tab that displays our app
- Verify that all of that happens on your machine.
- Add a console.log to the js file. The server should transpile your code and reload your app (you should see the console log) automatically.

Back to the Code for ToDoList

- Index.js contains a bunch of comments that will help us write the code for the app together. In the process I'll introduce
  - Local storage
  - HTML form validation and javascript validation. (bootstrap validation styles too)

New Syntax in ToDoList

- ES6 template literals
- json object literals
- JSON.parse() and JSON.stringify()
- try and catch blocks for exception handling
- event.preventDefault()
- Array.splice() and Array.push()
- Array.reduce() and Array.forEach()
- classList.add() and classList.remove() and the is-invalid Bootstrap style
- localStorage[“tasks”] or
- localStorage.getItem(“tasks”) and localStorage.setItem(“tasks”, this.tasks)

Production Code

- The translated (tranpiled) code is in the dist folder but we need to rebuild it now that our app is finished.
  - npm run build will do that.
  - Load the html file in the browser without the dev server and make sure that everything still works.
  - Copy the dist folder to citstudent
- By the way, the .gitignore file that's in the repository ignores the node_modules and the dist folder.
- The last part of reading quiz 3 asks you to answer some objective questions about node, npm, package.json and the specific node modules that are used in this lab.

Bookmarker app

- The next example we're going to do is the Bookmarker application. Let me show you what it does.
- I've given you the starting files on github.
  - index.html
  - styles.css
  - index.js contains a bazillion comments that are intended to help you write the code for the application.
- Let's look at those things together now.
- It is essentially the same problem as the ToDoList.
  - Start by setting up your development environment
  - Feel free to "steal" code from that application and modify it to complete this one.
- The ToDoList and the Bookmarker applications make up lab 3. Don't forget git and the evaluation!

What's Next

- MemeCreator App
  - CSS Flexbox
  - HTML5 Canvas
  - Using client side apis – FileReader
  - Asynchronous calls and Promises
  - Lab 4 – just this one problem that we'll do together.
  - Reading Quiz 4
- Don't forget
  - Reading Quiz 3
  - Participation Score 3
  - Lab 3