## **PROJECT: RESTAURANT PAGE**

### **Introduction**

Let’s use what we’ve learned and take a chance to continue practicing DOM manipulation by dynamically rendering a simple restaurant homepage! By the end, we are going to be using JavaScript alone to generate the entire contents of the website!

Note: DOM elements should be created using JavaScript but styling can be done in a separate CSS file.

### **Assignment**

1. Start the project the same way you began the webpack tutorial project.
   1. run npm init in your project directory to generate a package.json file.
   2. run npm install webpack webpack-cli --save-dev to install webpack to the node\_modules directory of your project.
      1. Quick tip: the node\_modules folder can get *really* big. It is customary to add a .gitignore file to your project so that you don’t have to sync the contents of node\_modules to github. The dependencies that are stored there can be installed from your package.json by running npm install, so you don’t need to sync them.
   3. Create a src and dist directory with the following contents:
      1. an index.js file in src
      2. an index.html file in dist. Go ahead and link the main.js file in a script tag. main.js is the file that will be generated by webpack.
      3. create a webpack.config.js file that looks just like our file from the [tutorial](https://webpack.js.org/guides/getting-started/#using-a-configuration).
2. Set up an HTML skeleton inside of dist/index.html with single <div id="content">.
3. Inside of src/index.js write a simple console.log or alert statement and then run npx webpack. Load up dist/index.html in a browser to make sure everything is working correctly.
   1. Quick tip #2: if you run npx webpack --watch you will not have to rerun webpack every time you make a change.
4. Create a bare-bones homepage for a restaurant. Include an image, headline, and some copy about how wonderful the restaurant is. It’s okay to hard-code these into the HTML for now just to see how they look on the page.
5. Now remove those elements from the HTML (so leave only the <html>, <body>, and <div id="content"> tags) and instead create them by using JavaScript only, e.g. by appending each new element to div#content once the page is first loaded. Since we’re all set up to write our code in multiple files, let’s write this initial page-load function inside of its own module and then import and call it inside of index.js.
6. Next, set up your restaurant site to use tabbed browsing to access the Contact and Menu pages. [Look at the behavior of this student solution](https://eckben.github.io/bearysBreakfastBar/) for visual inspiration.
   1. Put the contents of each ‘tab’ inside of its own module. Each module will export a function that creates a div element, adds the appropriate content and styles to that element and then appends it to the DOM.
   2. Write the tab-switching logic inside of index.js. You should have event listeners for each tab that wipes out the current contents and then runs the correct ‘tab module’ to populate it again.
7. If you are using GitHub pages to host your completed page you need to do a tiny bit more work to get it to show up. After running webpack the full bundled version of your site is available in the dist folder, but GH pages is looking for an index.html in the root directory of your project.
   1. Simply follow the instructions on [this gist](https://gist.github.com/cobyism/4730490). EZPZ!
   2. Recall that the source branch for GitHub Pages is set in your repository’s settings.

### **Examples**

* <https://michalosman.github.io/restaurant-page/>
* <https://igorashs.github.io/restaurant-page/>
* <https://sultanbadri.github.io/restaurant-page/>
* <https://ginnerzapata.github.io/restaurant/>