**Preparation**

Exercises will refer to my week 4 GitHub repo which is at [**https://github.com/oit-gaden/Web-Development-2020-Winter/tree/master/Week4/app1**](https://github.com/oit-gaden/Web-Development-2020-Winter/tree/master/Week4/app1)

**Exercise 1 - Install Node and NPM**

1. Install Node from [**https://nodejs.org/**](https://nodejs.org/)  
   You can install either current or latest.  NPM is installed along with the installation of Node.  No need to install Node if you know you already have it installed.
2. Open a command/shell window and run "**node --version**" to make sure you have successfully installed Node.
3. In the same command/shell window run "**npm --version** to make sure you have successfully installed npm.
4. Run "**npm help**" to see the several commands available.  You can also find documentation at [**https://docs.npmjs.com/cli-documentation/**](https://docs.npmjs.com/cli-documentation/)If you're not familiar with npm then take a minute to look at the documentation for the following commands which you will be using:  
     
   **npm init**  
   **npm install**

**Exercise 2 - Create Week 4 version of your Web app**

1. Create a Week 4 folder.
2. Copy the entire contents of your Week 3 folder to your Week 4 folder.  Just include the contents of your Web app.  Put your JavaScript in a folder named "js".  Put your css in a folder named "css".  Put your image(s) in a folder named "images".  Leave your html in the root of your week 4 lab folder.  These are not required folders but just one way to organize your application assets.  These will be the folder used in the Gulp build below.  You'll need to adjust the references to your JavaScript, CSS and image files in your html files.
3. Copy the following from my GitHub repo (see link above) to the root of your week 4 folder:  
     
   **.dockerignore**When you run "docker build" all files and folders in "." are sent to the Docker engine process to be used in the building of the Docker images.  Server development only files are not required so there is no reason to slow down the build by sending them to the Docker engine.  The .dockerignore file tells Docker what files/folders are not needed for the build.  
     
   **gulpfile.js  
   .gitignore** (You probably already have this.  If so, make sure it excludes the folder **node\_modules**).
4. From a terminal/command window in your week 4 lab folder run the following command:  
     
   **npm init**You can take all of the default answers or fill in answers as you wish.  This will create the file **package.json** in your week 4 lab folder.

**Exercise 3 - Install build tools for your Web app**

1. Add the gulp CLI.  From a terminal/command window in your lab folder run the following command:  
     
   **npm install -g gulp-cli**  
     
   The -g installs the gulp command globally so you can use it for all projects.  Check the installation by running the following command:  
     
    **gulp --version**
2. Install gulp and gulp plugins used for development build.  You'll find all of the development packages you'll need to install listed under "devDependencies" in my "package.json" file.  Open a command/shell window in your week 4 lab folder and run the following command:  
     
   **npm install --save-dev [package-name]**for each of the packages.  For example, to install dependency "gulp", run the following command:  
     
   **npm install --save-dev gulp**  
     
   Check your package.json file to make sure all of the "devDependencies"  listed match what is my package.json file.

**Exercise 4 - Build you Web app**

1. At the top of each .html file, surround your .css link and .js script lines with the same **removeIf** and **endRemoveIf** HTML comment lines that are at the top of my app's index.html file.  This causes your development only CSS and JavaScript references to be removed when the app is built using gulp (gulp inject plugin).
2. At the top of each .html file containing CSS references to CSS add the same inject:css and endinject HTML comment lines that are at the top of my files that contain CSS references.  This causes the build version of your .css to be "injected" into your .html files when the app is built using gulp (gulp inject plugin).
3. At the top of each .html file that contains references to JavaScript add the same inject:js and endinject HTML comment lines that are at the top of my files that contain JavaScript references.  This causes the build version of your .js to be "injected" into your .html files when the app is built using gulp (gulp inject plugin).
4. Run the following command in your week 4 lab folder to build your app:  
     
   **gulp**  
     
   You shouldn't see any errors displayed.  You should see "Starting 'watch\_files'".  **If you used the ECMAScript keyword "let" to declare a variable then you'll need to change it to "var" otherwise you will see errors in this step.**  You can hit "Control-C" to exit from the gulp command watching files once it stops.
5. Using VSCode open to your week 4 lab folder or the file explorer, look for the folder "**build**" and open it up. You should see your images copied to the images folder, the file app.min.css (bundled, minified version of your .css files)  with its associate source map file in the css folder  and the file app.min.js (bundled, minified version of your JavaScript files) with its associated source map file in the js folder  I would encourage you to look at the app.min.css and app.min.js files to see the minification applied (basically removal of white space).
6. View the gulpfile.js file and see if you can figure out how these files were copied/created.  No need to include this in your lab submission.
7. Using VSCode, you should now be able to click on your home page in the build folder, run live server and see your home page displayed.  You should also be able to navigate around your Web app using the page links at the top of each page. Styling from your .css file should be applied as expected.

**Exercise 5 - Watch app rebuilt and refreshed as you develop your Web app**

1. Invoke gulp again as above if you stopped it in Exercise 4.
2. Start LiveServer in VSCode starting with the home page in the build folder if you had stopped it.
3. Make changes to any of your assets (HTML, JavaScript and/or CSS).  Gulp sees the changes and rebuilds the app.  LiveServer sees the new built file(s) and refreshes the browser so you can see your changes in action.

**Exercise 6 - Build a Docker image with you built app**

1. Edit the Dockerfile and to the following:  
     
   Replace the line: "**COPY .  /usr/share/nginx/app/**" with "**COPY ./build  /usr/share/nginx/app/**"
2. Build the Docker image.
3. Run the Docker image to test.
4. Push your Docker image to Docker Hub.

**Other things to look at/try**

1. Completely remove the node\_modules folder and then run "**npm install**" in you Week 3 folder.  See that the node\_modules folder gets recreated as expected.  You should then be able to continue working as before.  Since node\_modules is not pushed to your GIT repo it is not available after cloning the repo.  So you run npm install after cloning.
2. Add the build script  from my package.json file (under scripts section) and try them by running "npm run [script name]".
3. Looking at the package.json file you'll see a "^" prior to each version number.  You can find out what that means at [**http://www.semver.org**](http://www.semver.org/)**.**