**# Shopping Cart Functionality**

You will be creating the core functionality used for a storefront application. The visual of the storefront as well as the JavaScript necessary to connect the code you will write, to that visual have already been created.

Shopping carts are a fantastic tool to practice JavaScript fundamentals.

**## Getting Started**

There are two ways you can work locally. You can ignore all the excess files and strictly work within the src file. Or, you can take advantage of the live server and tests to improve your experience.

To get the starter repository you can either:

- Download the project from GitHub by clicking on the green button and selecting download (this is easiest!)

- Forking the repository and then clone it to your local machine (see info on forking in the GitHub documentation: [Fork a repo](https://docs.github.com/en/github-ae@latest/get-started/quickstart/fork-a-repo).

**### [Optional] Forking and Cloning**

To fork the repository, click the "Fork" button at the top right corner of the repository page. Once forked, view the repository under your repositories. Clone the your repository to your local machine by running:

```bash

git clone path/to/your/GitHub/repository

```

**### Installing Node**

If you would like to take advantage of the live server and/or the tests, you must have node.js installed on your system.

To check if you already have Node.js installed, open your terminal application and run the following code:

```bash

node -v

```

If Node.js is installed, a version will be returned--something like `v16.14.2`.

If Node.js is not installed, you can install it from their website: <a href="https://nodejs.org/en/" target="\_blank">Nodejs.org</a>. Make sure to select the **\*\*LTS\*\***, "Recommended for most users," version.

**### Accessing the Working Directory**

Once Node.js is installed, you will need to change your working directory to your project's directory which is the `starter` folder in this repo - all the scripts, like npm start or npm test should run within the directory. If using a tool like Visual Studio Code, the terminal window will automatically place you in your project folder. If using your terminal application, type `cd` then drag the root project folder onto the terminal pane and hit [\*\*enter\*\*]. The result should be something similar to the following:

```bash

$ cd /Users/username/Documents/cd2073-intro-to-js-1-project-starter/starter

```

The path you enter will differ from the above but do note the space between `cd` and the start of the path.

**### Install Dependencies**

Once in the working directory, you need to add all of the node.js dependencies for the live server and Jest to work. Anytime we download a Node.js project, this is the first step we take.

```bash

npm install

```

You will see a new folder has been created named `node\_modules`. You can ignore this folder. Just know it holds all the stuff that makes the live server and Jest work, and that it's pretty huge. Because of its size, we don't push it or pull it from repositories. We just rebuild it by running the following command.

**### Accesing the working files**

You will be doing all of your work in `script.js`, which is inside the `starter/src/assets` folder. The starter code file structure looks like this:

**\*\****\_Starter Code\_***\*\***

```bash

├── src

│ ├── assets

│ │ ├── front.js

│ │ ├── script.js <--- Where you will be doing your work

│ │ └── styles.css

│ ├── images

│ │ ├── cherry.jpg

│ │ ├── orange.jpg

│ │ └── strawberry.jpg

│ └── index.html

├── tests

│ └── script.test.js

├── package.json

└── package-lock.json

```

**## Running the Live Server**

The live server is a tool that will auto update every time you make a change to a file in the src directory. So if you update your script.js and you want to check whether it worked, there's no need to refresh, the live server will handle that for you. To view your project in the workspace, the live server is mandatory.

To start the live server, enter the following code into your terminal application or the terminal workspace pane:

```

npm run start

```

Once you have entered this code, the live server will continue to run. If you would like to stop the server, type **\*\*[control] [c]\*\***. You will see the terminal window exit the live server.

**## Using Jest to test your code**

Jest is a testing framework for JavaScript. Small, [unit tests](https://en.wikipedia.org/wiki/Unit\_testing) are written to check the pieces of functionality of an application. Tests are typically written first and then code is written to make them pass. You have the option of taking advantage or prewritten unit tests. To run the tests enter the following code into your terminal application or the terminal workspace pane:

```

npm run test

```

When the tests run, you will see a bunch of information come up in the terminal pane. It will look something like this. On the left, you can see the tests failing, on the right you can see the tests passing.

![Jest results](https://video.udacity-data.com/topher/2022/April/62506d85\_tests/tests.jpg)

**## Project Instructions**

Step 1: Review the starter code

Step 2: Build the products

Step 3: Create the cart functionality.

- `addProductToCart()`

- `increaseQuantity()`

- `decreaseQuantity()`

- `removeProductFromCart()`

Step 4: Create the Checkout Functionality.

- `cartTotal()`

- `pay()`

Step 5: REFACTOR!

Step 6: Re-test the user interface

Step 7: [Optional] Add extra features

- Remove all items from the cart using an `emptyCart` function.

- Integrate a currency switcher to switch between USD, EUR, and YEN.

- Implement currency formatting to accomodate USD, EUR, and YEN.

**## Built With**

- Node.js (https://nodejs.com) - A JavaScript runtime for creating JavaScript servers

- Live Server (https://www.npmjs.com/package/live-server) - Live reloading for your workspace

- Jest (https://jestjs.io/) - A JavaScript testing framework

**## License**

[License](LICENSE.txt)