CSS 481 Final Project: Stock Website

By Phuong Vu

Part 2: Development Instruction

A. Required Dependencies

1. **Font Awesome:** icon for website
   - Sign up for an account with Font Awesome
   - Create your own account
   - Once your account is created, create a Font Awesome Kit.
   - Replace the Javascript Embed Code for your kit in the Script tag in the "head" tag of all html files

2. **Polygon.io:** Stock data API
   - Sign up for an account with Polygon.io
   - Create your own account
   - Once your account is created, choose basic free plan
   - Get your own secret API Key
   - Insert your API in the URL in the index file

3. Install **live-server** for localhost:
   - "npm install live-server --save-dev" at the project folder Terminal
   - In "package.json" file, add in:

```
  "scripts": {
"start": "live-server ."
  }          ,
```

   - Then, in the Terminal, type in "npm start" to start hosting the website on localhost

4. **Building Stock chart**:
   - Follow the instructions in this link

5. **Website Hosting:**
   - Create an account at W3school here
   - After creating an account, create a Space
   - Upload all the code to the space
   - The website link is generated at the top left of the project's space page

- The website is host at this link: https://stock-portfolio-web.w3spaces.com

B. Development Instruction

Notes: Please have all the dependencies stated above installed and register before moving on to this instruction section

1. Writing static website with HTML and CSS

a. Set up project:
   - Create a repository on Github
   - Name the repository project name.
   - On desired folder/path of computer, using Terminal to git clone the project link.
   - Project is ready in the folder to begin with.
   - Create a folder called images to store images for the website
   - Create a folder called data to store datafiles for publishing to Github later
   - Create a html folder to store html files of different pages for the website

b. HTML:
   - Create an index.html file with all the required tags in the head tag including meta, title, link, and script. The meta and title tags are required for html file setup. The link and script tags are required to link with stylesheet, favicon image, external package for icon, and JavaScript files.
   - Create a div of header part, which include the navigation bar and welcome page div.
   - Create the navigation bar with all necessary link to different pages. The href of the a tag should be the path of current page to other pages. Include in the navigation bar two icons: one for closing and one for opening navigation when screen is too small to show all the navigation contents.
   - Insert background image into the front page of index.html.
   - Put in welcome message with short cut button on top of the background image.
   - Create a footer for copyright information.

- In the html folder, create 4 different html file for different pages of the website: companyinfo.html, price-chart.html, stocknews.html, and stockprice.html
- Copy the content in the head tag, navigation div and copyright/footer div from index.html into the 4 newly created html files. Correct the path to different pages/files accordingly to current file location.

- Companyinfo.html:
  + Create a div in the body of the page between the navigation bar and footer for page content
  + Create a search bar to put into the navigation bar
  + Create a heading for the page
  + Create a div to store the company information
  + Put in Apple's information as default on the page
  + Inside the information div, there should be company name, description and 11 other information to be implemented.
  + For each information, it should have an id to be altered based on users' search later

- Stockprice.html:
  + Create a div in the body of the page between the navigation bar and footer for page content
  + Create a search bar and graph button(linked to the price-chart.html) to put into the navigation bar
  + Create a heading for the page
  + Create a div to store the company information
  + Put in Apple's information as default on the page
  + Inside the information div, there should be company name, open, close, highest, lowest price, number of transaction, trading volume, volume weighted average price.
  + For each information, it should have an id to be altered based on users' search later

- Stocknews.html:
  + Create a div in the body of the page between the navigation bar and footer for page content
  + Create a refresh button to put into the navigation bar
  + Create a heading for the page
  + Create a div to store all the article

+ Put in 10 news article as default on the page
+ Create a div for each article. Each div should have image, link to the article as title, description, span tag to display ticker name, and additional information like publisher, author, publication date
+ For each information, it should have an id to be altered based on users' search later

- Price-chart.html: Follow the tutorial in the Dependencies #4 stated above for this section
  + Create a div in the body of the page between the navigation bar and footer for page content
  + Create a heading for the page
  + Create a label and select tag to create a list of drop down company for user to choose from
  + Create a div to store the graph
  + Put in Apple's graph as default on the page
  + For each information, it should have an id to be altered based on users' search later
  + At the end of body tag, create a script tag to put in javascript code to fetch information and create the graph accordingly
  + Set the default value for company and stock ticker to be Apple and AAPL.
  + As the document load, draw the graph. When there is a change to the drop down button, update the value of company and stock ticker variables to be user's choice. Then, redraw the graph with user's input
  + Follow the tutorial stated above to draw graph.

c. CSS:
   - Set general styling rules for all the pages including margin, padding, font-family and box-sizing.
   - Set general styling rules for all the head, html body tag
   - Set general styling rules for all the navigation bar
   - Set styling rules for the front page including background image and welcome page
   - Set general styling rules for all html files in the html folder for their body-container div and heading inside the div
   - Set styling rules for the rest of the components inside the body-container to stay in the center of the page with proportional size.

- After all the styling for all pages in full screen is complete, open Chrome's Dev Tool to inspect screen on the smaller screen
- Alter styling rules to better accommodate smaller screen sizes.
  + Hide two opening and closing menu icon when the screen is big enough to show the navigation bar
  + These two icons would be shown when screen is too small to show all the items in navigation bar. Add styling so that the menu content in small screens goes in vertical order.

2. **User interaction with JavaScript**
   - Create the index.js file in strict mode
   - Add event listener for window when the dom is load and attach behaving function to be domLoaded()
   - Inside domLoaded() is all the behaviors would be executed when the page is loaded.
     + Add event listener and handling function for opening and closing menu button, which would be visible when the screen is small like mobile screen.
     + Add event listener and handling functions for users' input on the search bar and refresh button (stock news page) if the current page has the respective element
   - Create handling function to close and open menu for small screen size
   - Create handling functions to handle users' input on the search bar of company information and stock price page. In each of these function will call to another function to fetch API links
   - Create functions to fetch API links for the handling functions above, then render the received information to the page corresponding
   - Create handling function for the stock news page when users click on the refresh button. Inside this function, fetch API links and get the 10 newest article and render these article to the stock news page.

3. **Hosting Website**
   - In the terminal of the folder, use npm to install live-server. Testing the website interation on local host and ensure all the render information matches with Postman's result for testing purposes.
   - Upload all the code files to project's space on W3School. The website is hosted automatically with link at the top of the project's space page.
   - The website is hosted at: https://stock-portfolio-web.w3spaces.com/index.html