COMP 3512
Assignment #1: Single-Page App

Version: 1.1 February 17
Due Saturday March 6, 2021 at midnightish

Overview

This assignment provides an opportunity for you to demonstrate your ability to generate a dynamically updateable single page web application using JavaScript. **You can work in pairs or by yourself.**

Beginning

I feel foolish saying this in a third-year university course, but it is your responsibility to read all the assignment instructions thoroughly and carefully. If you are unclear about something, ask me. But before you do, read the material in question again!

Starting files will be found (eventually) at:

```
https://github.com/mru-comp3512-archive/w2021-assign1.git
```

Grading

The grade for this assignment will be broken down as follows:

Visual Design	20%
Programming Design	15%
Functionality (follows requirements)	65%

Data Files

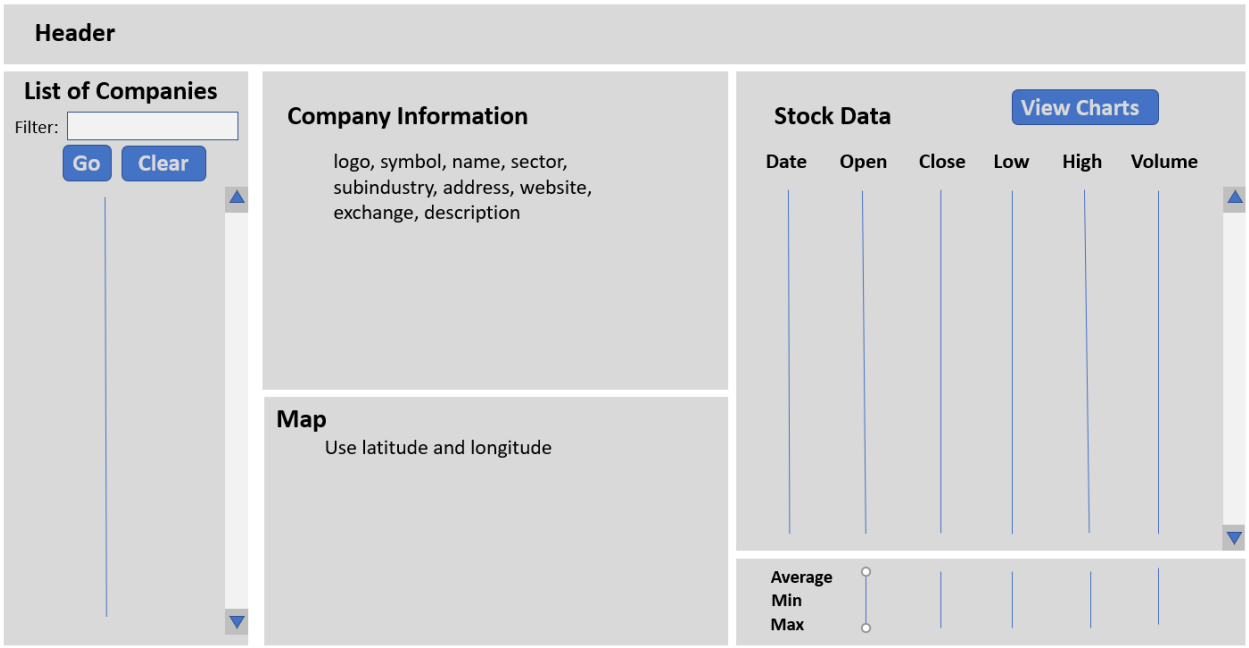
Images for the logos have been provided. Data is from one of my APIs.

Requirements

This assignment consists of a single HTML page (hence single-page application or SPA) with the following functionality:

1. Your assignment should have just a single HTML page which **must** be named `index.html`.
2. I expect your page will have significantly more additional CSS styling compared to `lab10-test05.html`. (Every year students receive minimal design marks due to minimal effort in CSS styling). If you make use of CSS recipes you found online, you must provide references (i.e., specify the URL where you found it) via comments within your CSS file. **Failure to properly credit other people's work in your CSS will likely result in a zero grade.** You can make use of a third-party CSS library; if you do, be sure to indicate this in the header. Attractive styling with your own CSS will result in a higher style mark than if you use a third-party library; poor styling with your own CSS will likely result in a lower style mark than if you use a third-party library.

3. You must write your own JavaScript. That is, no jQuery, no Bootstrap, no other third-party JavaScript libraries. You have all the knowledge you need from the lectures and the three JavaScript labs to complete this assignment. If you do find yourself, however, making use of some small snippet of code you found online (and I do mean small), then you **must** provide references (i.e., specify the URL where you found it) via comments within your .js file. **Failure to properly credit other people's work in your JavaScript will result in a zero grade. Using ancient JavaScript found online will result in lower marks.** There is no need to credit code you found in the labs or lectures.
4. Most of the functionality in the app can be found in the two sketches shown on the next few pages. Each of these is described in more details below. The first shown below is the **Default View**.



5. **Header**. The page title should be COMP 3512 Assign1. Add an icon or label named Credits. When the user mouses over this icon/label, please display your name(s), the course, any third-party technology used (include 3rd party CSS if using, Google Maps, charting package, etc). This message should be formatted uniquely and should disappear after 5 seconds (use setTimeout).

6. **List of Companies**. This must display a list of companies (`name` field). This assignment uses two discrete APIs: one for retrieving the company list, the other for retrieving company information, quotes, and month's stock data. The URL for the company list API will be:

   ```
   https://www.randyconnolly.com/funwebdev/3rd/api/stocks/companies.php
   ```

   This service returns just the most basic information about a few hundred companies.

   The API will take some time to retrieve this data. Display a loading animation (there are many free animated GIF or CSS available) until the data is retrieved. For this and the other API, they are also available via HTTP.

   To improve the performance of your assignment, you must store the company data in local storage after you fetch it. Your page should thus check if this company list data is already saved in local storage: if it is then use local data, otherwise fetch it and store it in local storage. This approach improves initial performance by eliminating this first fetch in future uses of the application. Be sure to test that your application works when local storage is empty before submitting (you can empty local storage in Chrome via the Application tab in DevTools).

   In the list simply display the company names, sorted alphabetically. Each list item must be clickable (be sure to change mouse cursor to indicate they are clickable). To make this application more readable, add scrollbars to the fixed-height container using the `overflow-y` property.

   When the user clicks on a company name, then populate the Company Info, Map, and Stock Data sections with the data for the clicked stock/company.

   Finally, above the list, provide a filter textbox. When the user types into the textbox (change event), filter the list so it only shows companies whose symbol begins with the same letters typed into the box. Provide some mechanism for clearing the filter (i.e., seeing all the companies).

7. **Company Info**. When the user clicks on a company in the list, then your program will display the rest of the company information already fetched from the API.

   In this section, display the following information from the JSON: `logo, symbol, name, sector, subindustry, address, website, exchange, description`. The `website` should be an actual working link. Also display the logo (provided). I expect this info to be nicely formatted and laid out sensibly.

8. **Map**. Display a Google Map using the company latitude and longitude properties at the same zoom level as in the lab10 mapping exercise. You will need to make use of the Google Cloud Platform (GCP) credits provided for you.

   Sometimes students have difficulty getting their Google Map to display correctly; in such a case, check for an error message in the console. The usual problem is that you haven't yet enabled the API for this service. From the hamburger menu of the Google Cloud Platform dashboard, select the APIs & Services option and then the Library option. Look for Maps JavaScript API from the API Library. Select it then click on the appropriate options to enable your API.

9. **Stock Data**. Display stock data for the current company/stock from the following API:

   `https://www.randyconnolly.com/funwebdev/3rd/api/stocks/history.php?symbol=xxx`

   where xxx is the `Symbol` property value for the clicked-on company/stock. The API will take some time to retrieve this data. Display a loading animation until the data is retrieved.

   This API returns three months of real historical stock data from January 2, 2019 to March 29, 2019. This works out to 61 rows of data. To make this data more readable, add scrollbars to the fixed-height container using the `overflow-y` property.

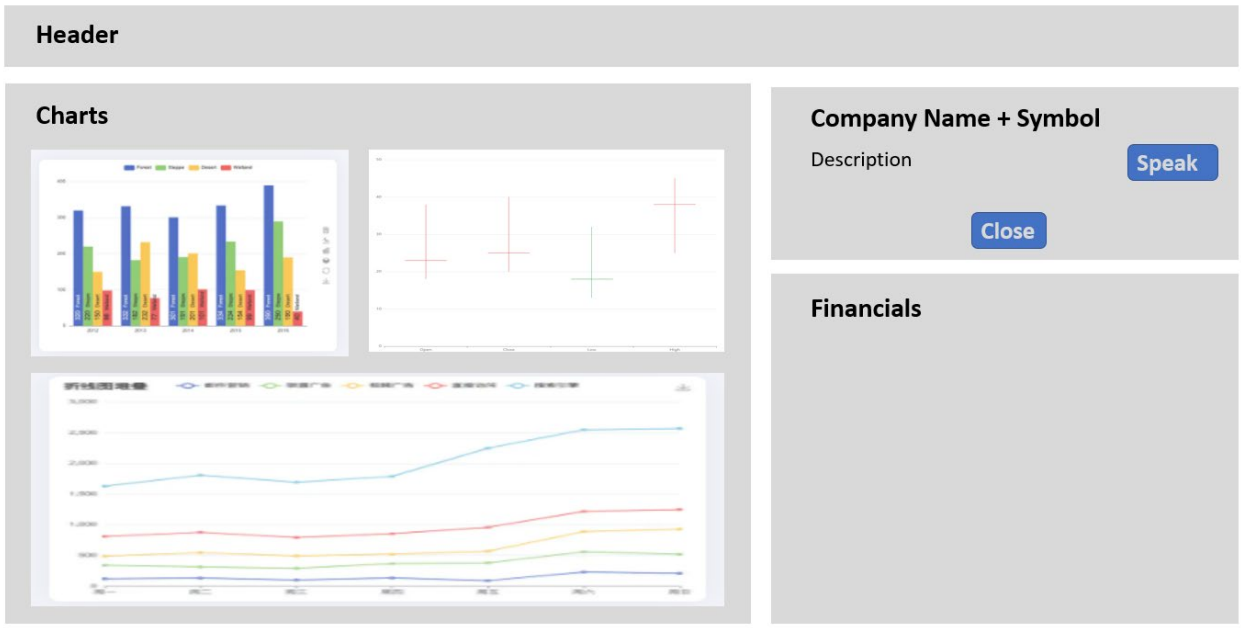
   Your program will display the `date`, `open`, `close`, `high`, `low`, and `volume` fields.

   The headings at the top of each column should be clickable (be sure to change mouse cursor to indicate they are clickable). When the user clicks on a column heading, the data will be redisplayed so that it is sorted on the value just clicked. When first displayed, sort by date. There should only be one initial fetch; the sorting should operate on the already fetched data.

   You will also need to calculate and display the following information from the stock data: average, minimum, and maximum values for open, close, low, high, and volume. I expect this information to be nicely formatted (e.g., currency as currency) and laid out sensibly.

   When the user clicks on the View Charts button, then sections 6,7,8, and 9 on the page will be hidden and replaced with the Chart View, described next.

10. **Chart View**. When the user clicks on the View Charts button, sections 2, 3, 4, 5 will be hidden and replaced with just the chart view for the stock:



For the Chart View, display the following fields: `symbol`, `name`, and `description`.

For the Financials area, display the 2017,2018, 2019 revenue, earnings, assets, and liabilities values in a table (this data is in the company data). I expect this information to be nicely formatted (e.g., currency as currency with commas for the big numbers) and laid out sensibly. **NOTE: only about 20% of the stocks have associated financial data, so you must be able to handle this elegantly: i.e., don't crash and display message in the page (not an alert).**

The Speak buttons will use the speech synthesizer to speak the `description` field content.

The Close button will hide this view and show instead the **Default View**.

For the charts, I would recommend using either echarts (https://echarts.apache.org/) or chart.js (https://www.chartjs.org/); you can use another package if you wish. These are analogous to Google Maps in that they are an external library with an API you will need to learn and discover. Both of these require either an empty or an empty <canvas> element which will end up being the container for the chart generated by the JavaScript charting library.

You will need the following charts:

- a bar or column chart which displays the 2017,2018, 2019 revenue, earnings, assets, and liabilities,
- a candlestick chart which displays the min, max, and average values for open, close, low, and high.
- a line chart which plots the close value and volume for each day in our data set. This will require different y-axis ranges for the close and volume lines. As well, since there are 61 data points, your x-axis labelling will need to be sensible.

Using these charting libraries will typically require examining the online documentation for the library and transforming your data into a form needed by the charting function.

Testing

Every year students lose many easy marks because they didn't read the requirements carefully enough. When your assignment is getting close to done, I would recommend going through each requirement step and carefully evaluate whether your assignment satisfies each specified requirement.

Submitting and Hosting

Your assignment will need to reside on a working public server and your source code on GitHub. Since your site only has static assets, you can make use of any free static file hosting provider. Some possibilities:

- Github Pages
- Netlify
- Firebase Hosting

These hosts work in conjunction with git and/or github. That is, you push to github; and then push to the host.

This step is going to take some time, so don't leave it till the very end. Be sure to test site on host to make sure it works.

I would strongly recommend getting your hosting to work a week before the due date. It's okay if your assignment is still not complete at that point: the idea here is to make sure hosting works ahead of time!

When your hosting is working and the assignment is ready to be marked, then send me an email with the following information:

- The URL of the home page of the site on your hosting platform.
- The names of the other people in the group
- The URL of the github repo so that I can mark the source code. If your repo is private, then add me as a collaborator.

Hints

1. Begin by testing the two APIs out first in the browser. Simply copy and paste the URLs into the browser address bar. The JSON can be hard to understand because it doesn't have white space, so use an online JSON formatter (such as https://jsonformatter.curiousconcept.com) via copy and paste to see the JSON structure in an easier to understand format. Alternately, install a json viewer extension in Chrome.
2. Remember that JSON and JavaScript are case sensitive. For instance, it is easy to type in `subIndustry` and it won't work because the JSON field is actually `subindustry`.
3. Your HTML will include the markup for both **Default View** and the **Chart View**. Initially, the later will initially use CSS to set its `display` to `none`. Your JavaScript code will programmatically hide/unhide (i.e., change the `display` value) the relevant markup container for the two views.
4. Most of your visual design mark will be determined by how much effort you took to make the two views look well designed. Simply throwing up the data with basic formatting will earn you minimal design marks.
5. Most of your programming design mark will be based on my assessment of your code using typical code review criteria. For instance, did you modularize your code using functions? Are your functions too long? Is the code documented? Do you have code duplication (you shouldn't)? Did you make use of object-oriented techniques? Are variables and functions sensibly named? Is your code inefficient (e.g., fetching the same data repeatedly)? Are you using outdated JavaScript techniques (e.g., inline coding, `XmlHttpRequest`, etc)? Is the code excessively reliant on found code from Stack Overflow, etc?
6. We didn't get a chance to discuss this yet in class, but when constructing single-page applications in JavaScript, you sometimes need to "insert" data into dynamic HTML elements that you modify/create in JavaScript. In HTML5, this is supported via the `data-X` attribute.
7. For the Speak buttons and the Close button, be sure to only add click event handlers for these buttons only once when the page loads (and not every time you display a chart).
8. Most years, students tend to get low marks on the design side of the assignment. I would recommend looking at other financial sites on the internet and examine how they present data. Notice the use of contrast (weight, color, etc) and spacing.