COMPX322-23A Assignment Two

Due Date: Monday April 24th, 10am

Object Oriented JavaScript : Commodity Prices Widget

PART ONE

For this coursework you are to implement an interactive page that supports the use of JavaScript web page components (widget). The widgets allow a user to view information about commodities and to generate graphs of prices over time for selected commodities. Multiple widgets can be displayed on the page in the form of a. You will use:

- HTML without in-line formatting, and with well-formed open and closing tags for HTML elements
- Cascading Style Sheets (CSS)
- JavaScript, coded in an **object-oriented style**
- DHTML (modifying web page content using JavaScript)
- Asynchronous requests to a server-side database and an external API
- PHP to make a database query and return results
- Use of an external JavaScript library for graph functions – Chart.js
- Use of an external API to provide commodity data – alphavantage.co

Application description

The application will be implemented in the form of an HTML page which includes a dashboard component and a canvas element for displaying graphs. The dashboard is used to display one or more 'widgets' – JavaScript components that can be inserted into the web page and which manage their own data and UI elements.
A user can select commodities from a select item which results in a widget for that commodity being added to the dashboard. The widget provides methods for a user to request a graph to be displayed for the prices of that widget, and also for a comparative graph to be displayed which compares the commodity prices with the currently graphed commodity.

Functional Requirements

- When the HTML page loads it displays an empty canvas element and a drop-down list of commodities (populated from an sql query as described below).
- When the widget loads a drop-down list is provided with a list of available commodities.
- When the user selects a commodity from the drop down list a commodity widget is created and added to the page. The widget contains information about the commodity and a button which allows the user to display a graph for the commodity's prices over time.
- The user can add, and remove as many widgets as they like (to a maximum of one per commodity)
- When the graph button of a widget is clicked, a graph is displayed showing monthly (default) historic prices for that commodity.
- Each time the user selects a different commodity graph, the canvas is cleared and replaced with a graph for the selected commodity.
- When a graph is displayed, the user is able to select a second commodity to add to the graph to provide a comparative view of the two sets of prices over time.

Non-Functional Requirements

- The Commodity widget is be implemented as a JavaScript object using a constructor function, this does not have to be in a separate file from the rest of the JS and does not need to be self-contained (so it only needs to be usable within the context of this assignment)
- When the page loads, an asynchronous request is made to the commodity database table to get all information. The returned commodity data must be stored in an **Array of object literals** (one for each commodity).
- Once the data has been read from the database and stored, the array of commodity objects is sorted alphabetically by commodity name.
- The names of the commodities are then used to populate a drop-down list.
- When a commodity is selected from the drop-down list a commodity widget is created and added to the page.
- When the graph button on a widget is selected, a request is made to the AlphaVantage API.
- When data is returned from the asynchronous request it is used to create a line graph which is built using the external Charts.js library and displayed in a canvas element on the page.
- Each time a new commodity is selected an asynchronous request is made for the data and a new chart is displayed.

- When a user requests a comparative graph, a request is made for the second commodity's data and the results are used in combindation with the existing graph data to create a graph showing both sets of prices

You have been provided with `commodities.sql`. Import this into your database for a table of commodity information, including the code required by alphavantage.co

Documentation for the alphavantage API can be found at:
https://www.alphavantage.co/documentation/

Make sure you look at the instructions for Commodities. You will need to register for a free API key.

What you need to do:

(a) Import the SQL file **commodities.sql** into your database. This will construct and populate a table of sample data.

(b) Register for an account with alphavantage.co and get a key

(c) Develop appropriate asynchronous request code to invoke a request to your php script to access the commodities database table and handle the response. You can use either AJAX or Fetch. The returned data should be stored client-side in an array of object literals.

(d) Write the JavaScript code needed to sort the array of object literals

(e) Write a constructor function for the Commodity widget which includes the commodity data and constructs the page elements needed to display it on the page.

(f) Write asynchronous request(s) to retrieve price data for a selected commodity. The code for the commodity will be retrieved from your array of commodity objects.

(g) Write methods to create the graphs from the commodity price data (both individual and combined) using Chart.js.

PART TWO

Sign up for a lab session to demonstrate your solution. You will be required to answer some questions about your code and show how you would refactor part of it based on a given change request. You can book an early lab session as soon as you have all of the required functionality complete (you don't need to wait until your solution is totally finished). Final demo slots will be available to book in the supervised labs the week of the deadline.

What to submit and how

All of your material for this assignment must be submitted electronically using Moodle.

Assuming that all parts of your application are within a directory called **compx322assn2** within your **course_html** directory.

Compress this folder using tar/gzip/zip (depending on platform) to create a compx322assn2.zip, compx322assn2.tar.gz or compx322assn2.tar.xv file.

In the COMPX322 Moodle site, you will see a link **Assignment 2** to the submission page. This link allows you to upload your zip file. You can do this as many times as you want up to the submission deadline for the assignment.

When you submit a file Moodle will ask you to confirm that what you have submitted is your own work, and will provide you with a 'receipt' that establishes that you have indeed submitted something. No other mechanism for submission will be accepted.

How your work will be assessed

The assignment will be marked out of **100** as follows:

- Application meets minimum functional requirements **20 marks**
- Asynchronous request is used to retrieve commodity data from the database table **5 marks**
- Commodity object array is sorted client-side **5 marks**
- Commodity names are used to populate a drop-down list of commodities that a user can select from **5 marks**
- Asynchronous requests are used to retrieve price data from alphavantage.co **20 marks**
- Chart.js is used to create line graphs of the selected data (individually and in combination) **20 marks**
- Object Orientation is used appropriately to logically group elements of the solution following the given structure:
    - Commodity data is stored in an array of object literals **5 marks**
    - Constructor function is used for the Commodity widget **5 marks**
- The rest of the JavaScript code is clearly structured, formatted and commented **5 marks**

- Lab demonstration (**mandatory**) **10 marks**

TOTAL **100 marks**