COMPX322-24A Assignment Two

Due Date: Monday April 29th, 10am

Object Oriented JavaScript : Towns and Weather Graphs

PART ONE

For this coursework you are to implement an interactive page that supports the use of JavaScript web page components (widgets). The widgets allow a user to view information about specific towns in New Zealand and to generate graphs of weather over a specified time period for selected towns. Multiple widgets can be displayed on the page at any time. 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 external APIs
- PHP to make database queries and return results
- Use of an external JavaScript library for graph functions – Chart.js
- Use of an external API to provide historical weather data - app.tomorrow.io

Application description

The application will be implemented in the form of an HTML page which includes a dashboard component for the widgets and a canvas element for displaying the graph. The dashboard is used to display one or more 'widgets' – JavaScript objects that can be inserted into the web page and which manage their own data and UI elements.
A user can select towns from a select item which results in a widget for that town being added to the dashboard. The widget displays information about the selected town and also has UI elements which enable a user to request a graph to be displayed for the weather of that town and to remove the widget from the dashboard.
The page also enables the user to select a 'Comparison Graph' option which allows them to choose two of the currently selected towns for creating a comparison weather graph.

Functional Requirements

- When the HTML page loads it displays an empty canvas element and a drop-down list of towns (populated from an sql query as described below).
- When the user selects a town from the drop down list a town widget is created and added to the page. The widget contains information about the town and buttons which allows the user to display a graph of recent weather (see below) and to remove the widget.
- The user can add, and remove as many widgets as they like (to a maximum of one per town).
- When the graph button of a widget is clicked, a graph is displayed showing the temperatures for the selected town for the last 24 hours.
- Each time the user selects a different town graph, the canvas is cleared and replaced with a graph for the selected town.
- If the user chooses to display a comparison graph they can then choose from the currently displayed towns which two (or more if you choose to implement this) should be used for a comparison graph of recent weather.

Non-Functional Requirements

- The Town widget is be implemented as a JavaScript object using a constructor function, this 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 town database table (provided in the town.sql file) to get **all** information. The returned town data must be stored in an **Array of object literals** (one for each town).
- Once the data has been read from the database and stored, the array of town objects is sorted alphabetically by town name.
- The names of the towns are then used to populate a drop-down list.
- When a town is selected from the drop-down list a town widget is created and added to the page.
- When the graph button on a widget is selected, a request is made to the app.tomorrow.io for Weather Recent History, the hourly history for the last 24 hours for the selected town.
- When weather data is returned from the asynchronous request it is used to create a line graph which is built using the Charts.js library and displayed in the canvas element on the page.
- When a user requests a comparative graph, a request is made to the app.tomorrow.io for Weather Recent History, the hourly history for the last 24 hours for each of the selected towns, all results are then displayed on one graph.

You have been provided with `towns.sql`. Import this into your database for a table of town information.

Documentation for the app.tomorrow.io can be found at:
https://docs.tomorrow.io/reference/historical
You will need to register for a free API key.

Documentation for Graphs.js library can be found at:
https://www.chartjs.org/docs/latest/

What you need to do:

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

(b) Register for an account with tomorrow.io and get a key. The free key restricts the number of requests you can make (25 per hour, 500 per day), I have provided two representative data files (townOneData.js and townTwoData.js) and recommend you use these while you are developing your code and switch to live requests once you have everything else working.

(c) Develop appropriate asynchronous request code to invoke a request to your php script to access the town 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 Town widget which includes the town data and constructs the page elements needed to display it on the page with the required UI elements (buttons etc).

(f) Write asynchronous request(s) to retrieve weather data for a selected town.

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

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	**30 marks**
• Asynchronous request is used to retrieve town data from the database table	**5 marks**
• Town object array is sorted client-side	**5 marks**
• Town names are used to populate a drop-down list of towns that a user can select from	**5 marks**
• Asynchronous requests are used to retrieve weather data from app.tomorrow.io	**10 marks**
• Chart.js is used to create line graphs of the selected data (individually and in combination)	**15 marks**
• Object Orientation is used appropriately to logically group elements of the solution following the given structure:	
○ Town data is stored in an array of object literals	**5 marks**
○ Constructor function is used for the Town widget	**10 marks**
○ Data objects for the Charts	**5 marks**
• All code is clearly structured, formatted and commented	**10 marks**
TOTAL	**100 marks**

Oral Assessment Component

For each assignment a group of students will be randomly selected to give a short (10 mins) oral presentation to explain your code and describe your solution. Selected students will be contacted after the submission date to arrange this.

Note: marks will be deducted if we have to edit your PHP scripts to correctly connect to the UoW MySQL server so make sure you test in that environment and submit code that will successfully connect.