**Unit 12 Lab – Advanced JavaScript**

**Exercise 1 – Carousel Slider as a Class**

Take the code for the carousel slider in exercise-1 of unit-10 (Dynamic HTML) and repackage it as a class. This exercise only affects the JavaScript code and so the rest of the code (HTML, CSS, images) is unaffected

1. Create a new folder called “unit 12” containing a folder called exercise-1-class-carousel-slider
2. Copy either your own solution code for exercise-1 of unit-10 or the solution code as provided
3. In the main.js file, create a new Class called “CarouselSlider”
4. Create private properties for any global variables in the original code. Use the same names
5. Copy the initCarousel function code into the class constructor
6. Copy the code for the showSlides, incrementIndex functions and make them private
7. Modify the code so that it will work with the new Class structure
8. In the initCarousel function, create a new instance of the CarouselSlider class
9. Test your code. It should function as before

**Exercise 2 – Table for Data as a Class**

Take the code for generating the table in exercise- 6 of unit-10 (Dynamic HTML) and repackage is as a class. This exercise only affects the JavaScript code and so the rest of the code (HTML, CSS, images) is unaffected

1. Create a new folder called exercise-2-class-table-from-data
2. Copy either your own solution code for exercise-6 of unit-10 or the solution code as provided to the new folder
3. In the main.js file, create a new Class called “DataTable”
4. Create private properties for any global variables in original. Use the same names
5. The Constructor has an argument called “data” which takes an object and save it as a private property called “data”
6. Copy the renderTable code and rename it to “render”. The “render” function has no arguments.
7. Refactor (modify) the render function code to use the new “data” property
8. Refactor the existing “init” function to create a new DataTable object and call the “render” function
9. Test your code. It should function as before

**Exercise 3 – Table for Data as a Class using Fetch**

Building on exercise 2, use fetch() to load the table data dynamically

1. Create a new folder called exercise-3-class-table-from-dynamic-data
2. Copy the code from exercise-2-class-table-from-data
3. Create a new async private method called “loadHolidayData” using the code on slide 46 and load the table data from <https://date.nager.at/api/v3/PublicHolidays/2024/IE>
4. To update the DOM, call the “render” method after the response is received. Otherwise the “render” method will not wait for the response is received and so it will display an empty table.
5. In the Contructor, call the “loadHolidayData” method
6. Test your code. It should function as before

**Exercise 4 – Product Table using Fetch**

Building on exercise 3, use fetch() to load the products from <https://mdn.github.io/learning-area/javascript/apis/fetching-data/can-store/products.json> and display as a table

1. Create a new folder called exercise-4-product-table-using-fetch
2. Using the code for exercise 3 as a template create a “DataTable” class which accepts the products URL above as an argument
3. Display the data as a table
4. The images can be loaded from here -> <https://raw.githubusercontent.com/mdn/learning-area/main/javascript/apis/fetching-data/can-store/images/>
5. Test your code

**Exercise 5 (Challenge) – Generic Table using Fetch**

An enhancement to the previous exercises is to make the DataTable code generic, i.e. refactor the DataTable class so that it will display data from either the holidays or products URL’s

In order for this to work the format of the data returned by the fetch() method needs to be similar which is why we are restricting this exercise to just these two URL’s

Use the property names of either the public holiday or products as the table headings. Use the method “Object.getOwnPropertyNames()” to return a list of the properties in the data object

Show the tables for both URL’s

Note: for the product data, you won’t be able to show the image as the API doesn’t include a full url for the image property. Display the image name instead

**A screenshot of a computer

Description automatically generated**