COMP3006 – Full-Stack Development

Workshop 3 – JavaScript Objects

Autumn 2020

This week's workshop will require you to implement some JavaScript programs to test your knowledge of the objects following this week's lecture. You can reuse the test harness from last week to run these programs if you like, or create a new HTML file for each question.

You will also extend your work on the mini project by adding to the JavaScript functionality you began developing last week.

Before the Workshop

Exercise 1

This week is all about objects, so the first thing to do is to make sure you can define your own objects. Write an object that stores your forename, surname and date of birth in an object. Use the `console.log` pattern to print out the whole object, and each key/value pair.

> Have a look at the lecture – slide 7 will be helpful.

Exercise 2

Last week you wrote a function that returns an array containing the unique values within an array that it receives as an argument. I was intending for you to do this using conditionals and loops, but commented that JavaScript has an inbuilt *set* method for doing this. Write a function that takes an array of values and constructs a `Set` object that can be returned by the function.

> We haven't explicitly covered the **Set** object in the lectures or labs – you are expected to do some research to find out how it works.

During the workshop

Exercise 3

Write a program that generates the current system time with a `Date` object. If the current time is before 12pm then the program should write into the HTML page *"Good morning"*. If it is after 12pm and before 6pm then the HTML should read *"Good afternoon"*. Otherwise it should read *"Good evening"*.

Exercise 4

Define the following data structure – you can use constructors or classes. A *lecturer* has a name, email address and office number. A *student* has a name, email address and student number. Examples are:

- **Lecturer:** David Walker, `david.walker@plymouth.ac.uk`, PSQ A322.
- **Student:** Jane Smith, `jane.smith@students.plymouth.ac.uk`, 1234567

Your task is to extract details from a form, use them to instantiate an object of either type `Lecturer` or `Student`, and then use the object to populate a table.

- You should abstract the common data to a class or object called `Person`, which is inherited by the `Lecturer` and `Student`.

> It might be a good idea to think about how you break this into functions – maybe have a function to add the data to the appropriate table.

- In the HTML you should have a form for adding a student and a form for adding a lecturer. There should be a table for each type too (you will need to give the table an ID so that you can use the jQuery `append` method to add rows to it).
- Each form should have an add button so that the current form values are used to create an object of the appropriate type, before being added to the appropriate table when the button is clicked. The last operation in the event handler should clear the form for the next new record.

Once you have it working you should extend the code so that new items appear in a page using HTML.

Mini Project

This week you will extend the mini project by adding what you have learned this week about objects and classes in JavaScript.

Exercise 5

Your task is to create an appropriate data structure with which to represent a message. Think about what data you will want to store – I imagine there will be a message body, along with details of who sent the message and when. Treat these as bare minimum requirements, you should think about how to extend it.

For now you should do the same as in the previous exercise – when the send button is clicked, retrieve the content from the user interface (use dummy variables where something is missing – for example, you will learn how to record the current user later in the module) and create an instance of the data structure you have defined. Use the `console.log` pattern to print out the data and ensure it works.

Extension task

Exercise 6

A University of Plymouth email address can take the following forms:

- `david.walker@plymouth.ac.uk`
- `david.walker-1@plymouth.ac.uk`
- `david.walker@students.plymouth.ac.uk`
- `david.walker-1@students.plymouth.ac.uk`

Write a program that provides an input with which the user can type in their email address. Then use JavaScript's regular expression provision to verify that it is or isn't a valid Plymouth email address. You should show a message in the page to say that it either is or isn't.