Session 2 exercises

Advanced JavaScript for Web Sites and Web Applications

Exercise 1

- Download and extract the *workshop2.zip* file to a location in your workspace.
- In `workshop2.js`. using the array/join string building technique discussed earlier, create the HTML string as shown below.
- You can add the assembled HTML string to the `workshop2.html` page, after the `<h1>` tag, using `insertAdjacentHTML`.
- The HTML:

```
<ul>
    <li>This is a first item</li>
    <li>This is a second item</li>
    <li>And another one!</li>
    <li>...and a last one</li>
</ul>
```

Exercise 2

- In the `workshop2` folder, you will find a sub-folder called `converter`
- Open `converter.html` and `converter.js` in your editor
- Can you tell what the code is doing?
- Preview `converter.html` in your browser to see it in action.
- This application works, but there is a problem
  - the rates are hard-coded in the function definition, making them hard to manage globally.
  - Plus, if we want to use the rates elsewhere in our application, we will have to duplicate them.
- Your Task: Resolve this issue by creating a new object with the rates as properties.
  - You should create this object in the *global scope* (i.e. not inside the function)
- When the rates are needed in the application, you can simply reference this object.
- Extra: if you have time, modify the `convert` function definition so that the rates object is passed to it as an argument. You will also have to modify the event listener function so that the rates object is passed to `convert()`

Exercise 3

- In `workshop2.js`, Use a `for...in` loop to iterate the following object and count the number of properties and the number of methods contained in it:

```
var basket = {
  items: 0,
  totalPrice: 0.50,
  addItem: function() {},
  removeItem: function() {}
};
```

- Remember:
  - you can use `typeof` to determine a variable's *type*, bearing in mind that *methods* are really *functions*
  - in the loop, you access the property using *bracket notation*
- E.g.:

```
if (typeof myObject[propertyName] === 'function') {
    // myObject[propertyName] is a function
} else {
    // myObject[propertyName] is not a function
}
```

- Everything that does not have *function* as it's type should be considered a *property*.
- Use `console.log()` to display the results

Exercise 4

- In `workshop2.js`, rewrite the code shown below so that the `message` string is constructed using the array/join technique we looked at earlier.
- Try to avoid any *concatenating* of variables and strings
  - i.e. don't use + or +=

```
var myObject= {
    a: 1,
    b: 2,
    c: 3
};
var prop, message;

for (prop in myObject) {
    // "prop" will hold the name of property
    message = "Property name: " + prop + ", ";
    // Use myObject[prop] to get it's value
    message += "Value: " + myObject[prop];
    console.log(message);
}
```

Exercise 5

- In the `workshop2` folder, there is a file `exercise_data.js` which contains the definition of an object holding information about various courses at City University.
- Task: Using a `for...in` loop and the array/join string building technique, create a `<ul>` list that will show the *name* of each course and the *weekday* it runs on.
  - The *name* should be linked to the respective URL.
- **TIP:** Create an empty array to start with. As you process the data, add new elements to this array. When done, `join()` the array elements.
  - Remember, you can use `.push()` to add new elements to an array
- Again, you can add the completed HTML string to the `workshop2.html` page after the `<h1>` tag, using `insertAdjacentHTML`.

Your completed HTML should look something like:

```
<ul>
    <li><a href="course_url">course_name</a> - <span>course_day</span></li>
    <li><a href="course_url2">course_name2</a> - <span>course_day2</span></li>
    <!-- etc.... -->
</ul>
```