LAB 1: Working with core JavaScript

Contents

Introduction 1
Resources 1
Task 1. Objects and methods 2
Task 2. Classes 3
Task 3. Using subclasses 4

Introduction

In this lab you will practice working with objects and classes in JavaScript. You will also practice running and debugging JavaScript.

Resources

- Lecture notes and sample code from Module Website
- Visual Studio and Firefox/Firebug (or any combination of editor/browser)
- JavaScript reference, e.g. **https://developer.mozilla.org/en/JavaScript/Reference**

Task 1. Objects and methods

Create a new web page to contain the JavaScript you will write. If you are using Visual Studio you can create a new ASP.NET web project and add a new HTML file.

Add a <script> element to your web page. You can either write your JavaScript in the page, or refer to a separate .js file which you create for your script.

Write JavaScript to accomplish the results described below. You should test your script by viewing the page in your browser. You can use your debugger, e.g. Firebug, to help you.

1. Create an object called *team*, using an object initialiser, which represents a Formula 1 racing team. The object should have the following properties:

   name -"McLaren",
   location - "Woking",
   principal - "Martin Whitmarsh"

2. Write a representation of your object to the web page using:

   ```
   document.write(JSON.stringify(team, null, 4));
   ```

   You can also set a breakpoint in your debugger and inspect the object. You can use these approaches at subsequent stage to check your objects are as they should be.

3. Add a new property *drivers* to *team*, which consists of an array containing two objects with the following properties:

   name - "Jenson Button", dob - 19/1/1980
   name - "Lewis Hamilton", dob - 7/1/1985

4. Write the *dob* property of one of the drivers to the web page in a sensible format.

5. Add a new property *points* to each driver, with the following values:

   [25, 18, 0, 25]
   [12, 10, 25, 8]

6. Add a method *totalPoints* to team which calculates the total number of points held by both drivers combined. Call this method and write the result to the web page (should be 123).

7. Add a new points value, 25 and 10 respectively, to each driver. Call the *totalPoints* method again and write the result to the web page (should be 158).

Task 2. Classes

In this task you will define a class, or prototype, to represent a Formula 1 team, and create an instance of this class.

Add a new script to your project, either within a web page or in an external file which is referenced in your page. Write code to accomplish the following:

1. Create a constructor *Team* which takes parameters representing the name, location and principal of the team and the names of two drivers. An example call to this constructor would be:

   ```
   var mclaren = new Team("McLaren", "Woking", "Martin Whitmarsh", "Jenson
   Button", "Lewis Hamilton");
   ```

   The constructor should initialise an object with the same structure as *team* in Task 1, except that we will not include the *dob* birth property of the driver objects.

2. Add methods to the prototype of *Team* to as follows:

   addPoints(driver, points), where driver is the index of a driver, 0 or 1, and points is the new points value to add.

   totalPoints(), which as in Task 1 calculates the total number of points held by both drivers combined.

3. Test your class by calling the constructor and the methods as follows:

   ```
   var mclaren = new Team("McLaren", "Woking", "Martin Whitmarsh", "Jenson
   Button", "Lewis Hamilton");

   mclaren.addPoints(0, 25);
   mclaren.addPoints(0, 18);
   mclaren.addPoints(0, 0);
   mclaren.addPoints(0, 25);
   mclaren.addPoints(1, 12);
   mclaren.addPoints(1, 10);
   mclaren.addPoints(1, 25);
   mclaren.addPoints(1, 8);


   document.write(JSON.stringify(mclaren, null, 4));
   document.write("<br />");

   document.write("Points: " + mclaren.totalPoints());
   document.write("<br />");
   ```

   The value returned by *totalPoints* should be 123.

Task 3. Using subclasses

In this task you will refactor your *Team* class so that its *principal* and *drivers* properties are instances of classes *Principal* and *Driver*, which are subclasses of a class *TeamMember*. The classes should have the following properties/methods:

TeamMember
name property

Principal
name property, inherited from *TeamMember*

Driver
name property, inherited from *TeamMember*
points property, initially an empty array
addpoints(points), which adds the new points value to the driver's *points* array
totalpoints(), which returns the driver's points total

Add a new script to your project, either within a web page or in an external file which is referenced in your page. Write code to accomplish the following:

1. Create constructors *TeamMember*, *Principal* and *Driver.*

2. Add the required methods to the prototype of *Driver*.

3. Refactor the *Team* constructor so that it:

   - Can be called with the same parameter list as in task 2

   - Creates a new instance of *Principal* for the *principal* property, using the appropriate constructor parameter.

   - Creates an array containing two new instances of *Driver* for its *drivers* property, using appropriate constructor parameters for the drivers' names

4. Refactor the *addPoints* and *totalPoints* methods of *Team* so that they are called in the same way as in task 2, but delegate appropriately to the methods of *Driver*.

5. Test your class by calling the constructor and the methods using the same code as in task 2.

   The value returned by *totalPoints* should still be 123.