Scripts and objects

JavaScript provides pre-built functionality through an object-oriented interface. Because of this, we have to understand how to use objects.

Scripts:

It is good practice to include JavaScript code in separate files with the extension **.js**. We can include our script into our HTML file using the **<script>** tag. We would include the script called **lab2b.js** by inserting

```
<script src="lab2b.js" defer></script>
```

into the **<head>** element of the HTML file. The **defer** attribute guarantees that the web page will finish loading before the browser attempts to run the script. The web browser will interpret the HTML page line by line from top to bottom. Every time the browser sees an element it creates it in memory. If our script were to run before the page finishes loading, then some of the elements will appear missing to your program and lead to errors.

Object:

An object is a container that represents things like people, elements on the page, or library. This container can hold two types of things:

- Properties
- Functionality/Methods

We can represent people as objects. These objects may have properties like name or height. They may also have functionality like walk or run.

The Math Object

References:

- http://www.w3schools.com/js/js_math.asp
- https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Math

The math object holds mathematical constants as properties and mathematical functions as methods. Constants are in all uppercase.

Accessing Properties:

If we wanted to access the constant π, we would write `Math.PI`. Notice the "." between **Math** and **PI**. Math is the name of the object, while **PI** is the name of the property. This patterns hold for all objects. We can access an object property by first writing the name, followed by a "." and then the name of the property.

Accessing Methods:

There are many special functions in math, that are in the Math object. Like Mathematical functions, functions in JavaScript also have parameters. Let's access the **sin** function.

If we wanted to calculate sin $(\frac{\pi}{4})$ we would write `Math.sin(Math.PI/4)`. Once again we write the object name followed by a ".". Next we write the method name followed by a pair of parenthesis. Finally, the value of our argument goes into the parenthesis, in this case `Math.PI/4.` Different methods have different number of parameters, so you should check the documentation mentioned above.

Generating random numbers

To generate an integer from 1 to 4 we can write:

```
Math.floor(Math.random()*4)+1;
```

`Math.random()` generates floating point number from 0 to 1, not including 1.
Multiplying by 4 scales the random value to 0 to 4, not including 4.
Math.floor, rounds the number down to an integer between 0 and 3.
Adding 1 moves the range of integer up by 1, giving us a random value of between 1 and 4.
We can replace 4 with any number, to give us an integer between 1and any number.

Element objects

Elements on the pages are represented as objects in JavaScript. If we give the element an **id** we can access element using that **id.**

HTML:

```
<div id="divExample"> <p>Filler text</p></div>
```

In Java Script we would access the above element using **divExample.**

JavaScript:

```
divExample = document.getElementById("divExample");
divExample.innerHTML = "<p>New Text </p>";
```

Like all objects, element objects have properties and methods.

document object and getElementById :

The document object represents the entire webpage. It is possible to locate elements on the webpage via their **id** attribute. Using the **getElementById** method, we can obtain the reference to the element.

innerHTML property:

The **innerHTML** property seen above, allows us to access the HTML nested inside an element. If we assign new text to the **innerHTML** property, we can replace the nested HTML on the webpage for that element.

The full reference to element objects can be found at
https://developer.mozilla.org/en-US/docs/Web/API/Element

Example:

The following example demonstrates how to fill in text in an HTML page using JavaScript, the use of objects and expressions.
The HTML contains several place holder ** ** elements questions and answer via JavaScript.

Example2b.html:

```
<!DOCTYPE html>
<head>
   <title>Example 2b</title>
   <meta charset="UTF-8">
   <script src = "Example2b.js" defer ></script>
</head>
<body>
      <h1>Example 2b: Filling in text with
        JavaScript </h1>
    <article class = "practiceProblem">
      <h2>Problem 1</h2>
      <div id = "question1" class = "question"></section>
      <div id = "answer1" class = "answer"></section>
    </article>

    <article class = "practiceProblem">
      <h2>Problem 2</h2>
      <div id = "question2" class = "question"></section>
      <div id = "answer2" class = "answer"></section>
    </article>

</body>
```

Example2b.js:

```
//Code for Problem 1
var peppers = Math.floor(Math.random() * 5) + 1;
var question1 = document.getElementById("question1");
var answer1 = document.getElementById("answer1");
question1.innerHTML = "Peter ate "+ peppers +
                      " peppers. What is his name?";
answer1.innerHTML = "Answer = Peter"

//Code for Problem 2
var base = Math.random() * 4;
var height = base / 4;
var question2 = document.getElementById("question2");
var answer2 = document.getElementById("answer2");
question2.innerHTML = "What is the area of a"+
                      " Triangle  with  a base of " +
                       base + " and a height of " +
                       height;
answer2.innerHTML = "Answer = " + base * height / 2;
```

To modify the element contain in an element the innerHTML property is used. The procedure is as follows:

1. Get a reference to the element using **getElementById**
2. Assign the new HTML text you want to be nested within the element to the **innerHTML** property.