# HTML and JavaScript Typical Setup and Usage

|  |
| --- |
| ***javascriptdemo.html***  <!DOCTYPE html>  <html>  <head>  <title>JavaScript Demo</title>  <meta charset="utf-8" />  </head>  <body>  <header><h1>JavaScript Demo</h1></header>  <script src="myjs.js"></script>  </body>  </html> |
| ***myjs.js***  var h1s = document.getElementsByTagName('h1');  h1s[0].style.backgroundColor = "red"; |
|  |

# Functions

1. ***Write a function*** that accepts an array of numbers as its only parameter. The function returns the largest number in the array.
   1. Make up a test case and then show that the function works
2. ***Write a function as an expression*** that accepts an array of numbers as its only parameter. The function returns the average of the numbers.
   1. Make up a test case and then show that the function works
3. Write a function that accepts an object as its only parameter. The object has the following properties: name, age, height, weight, telephone, email. The function simply logs all the properties to the console.
   1. Demonstrate that the function works.
4. Write a function called ‘areaOfCircle’ that calculates and returns the area of a circle (). The function takes no parameters but should assume that there is an object with property ‘radius’ (I.e. use this.radius).
   1. Create object called circle1 with radius 10.
   2. Create an object called circle2 with radius 100.
   3. Use context binding to output the area of circle1 and circle2.
5. Use closures to implement a function that returns a unique string each time it is called.
   1. Demonstrate the function
6. Use closures to generate 10 links. Each link’s text should read ‘Link <n>’, where <n> is a multiple of 10, starting at 10 and ending at 100. When the link is clicked, change an area on the page to read ‘You clicked link <n>” where n is the link’s number.
   1. Demonstrate that it works

# Objects

1. Create an object called ‘**car**’ with the following properties:
   1. **yearModel**: a string that holds the car’s model year
   2. **make**: a string that holds the car’s make
   3. **speed**: a number that holds the car’s current speed
   4. **accelerate**: a method that adds 5 to the **speed** property
   5. **brake**: a method that subtracts 5 from the **speed** property
   6. Demonstrate the object in action
   7. Demonstrate how to remove the **make** property from **car**
2. Create a constructor function called ‘**Car**’. The function should take 3 parameters: **yearModel**, **make**, and **speed**.
   1. **yearModel**: a private string that holds the car’s model year
   2. **make**: a private string that holds the car’s make
   3. **speed**: a private number that holds the car’s current speed
   4. Use the constructor function to create 3 different instances of Car and then log each instance**.**
   5. Call accelerate on one of the instances.
   6. Use prototyping to add the **accelerate** and **brake** methods.
      1. **accelerate**: a method that adds 5 to the **speed** property
      2. **brake**: a method that subtracts 5 from the **speed** property
   7. Call the **accelerate** and **brake** methods on the instances and then output the **speed** value of each instance.
3. Use constructor functions, prototyping, closures, and context binding to implement the following class diagram. Use a separate file to define the classes.

|  |
| --- |
|  |

* 1. Instantiate an **Employee** and output its values.
     1. Try outputting the employee’s pay (call **getPay**)
  2. Instantiate a **ProductionWorker** and output its values. Output the worker’s pay after working for 10 hours.