

# COMP 1842

Week 2 part 2 – JavaScript continued

Matt Prichard

# Introduction

Recap

Basic control structures

Functions

Interactivity

# Recap

- Last week we looked at how to include JavaScript into a web page.
  - Externally, like a CSS page
  - In the page in `<script></script>` tags
  - Inline within an HTML element

[https://www.w3schools.com/js/js\\_where.asp](https://www.w3schools.com/js/js_where.asp)

# Recap 2

- We saw how to use the var keyword to define a variable.
- We looked at naming conventions and dynamic data typing.
- We went through the various operators such as assignment, arithmetic, comparison and logical and saw how similar these were to PHP.

[https://www.w3schools.com/js/js\\_operators.asp](https://www.w3schools.com/js/js_operators.asp)

# Recap 3

- We looked at 3 ways to display output in JavaScript
  - `alert("hello world");` to make pop up boxes
  - `console.log("hello world");` to write to the JS console
  - `document.getElementById("demo").innerHTML = "Hello World!";` to write to an HTML element with an ID of “demo”.
- Finally we looked at the **importance** of using the console to debug our code.

# Crash course continues

As I have said this module will be focused on JavaScript, but when we begin using Vue.js (which is clearly written in JavaScript), I will discuss the code as we get to it.

These first 2 weeks are really just to get a flavour of the language. I can't stress enough the importance of working through some W3schools or Codecademy tutorials in your own time.

# let vs const vs var

I highly **recommend you read** this section

[https://www.w3schools.com/js/js\\_variables.asp](https://www.w3schools.com/js/js_variables.asp)

- Always declare JavaScript variables with var, let, or const.
- The var keyword is used in all JavaScript code from 1995 to 2015.
- The let and const keywords were added to JavaScript in 2015.
- If you want your code to run in an older browser, you must use var.

# const

- The const keyword was introduced in [ES6 \(2015\)](#).
- Variables defined with const cannot be Redeclared.
- Variables defined with const cannot be Reassigned.
- Variables defined with const have Block Scope.
- As a general rule, always declare a variable with const unless you know that the value will change. Use const when you declare:
  - A new Array, a new Object or a new Function



# let

- The let keyword was introduced in [ES6 \(2015\)](#).
- Variables defined with let cannot be Redeclared.
- Variables defined with let must be Declared before use.
- Variables defined with let have Block Scope.
- <https://www.freecodecamp.org/news/understanding-let-const-and-var-keywords/>

# Basic control structures

# Conditional statements

if, else and else if

As we can see PHP and JS are almost identical here. The brackets, the semi colons, the overall structure are the same. The only difference is php has 'elseif' and JS 'else if'. Notice the space between the words in JS

```
<?php
```

```
$t = date("H");
```

```
if ($t < "10") {  
    echo "Have a good morning!";  
} elseif ($t < "20") {  
    echo "Have a good day!";  
} else {  
    echo "Have a good night!";  
}  
?>
```

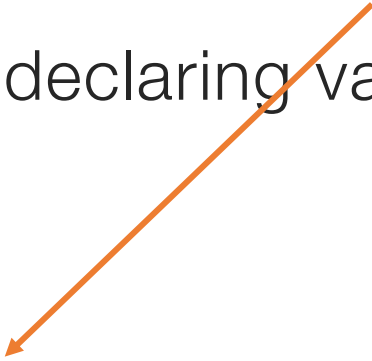
```
if (time < 10) {  
    greeting = "Good morning";  
} else if (time < 20) {  
    greeting = "Good day";  
} else {  
    greeting = "Good evening";  
}
```

---

# For loop

The for loop structure and syntax is the same in JS as PHP.

Notice the use of 'let' for the counter in the JS version, this is the new way of declaring variables that change, but var would still work.



```
for (let i = 0; i < 5; i++) {  
    text += "The number is " + i + "<br>";  
}
```

```
<?php  
for ($x = 0; $x <= 10; $x++) {  
    echo "The number is: $x <br>";  
}  
?>
```

# While loop

The while loop structure and syntax is the same in JS as PHP.

```
while (i < 10) {  
    text += "The number is " + i;  
    i++;  
}
```

```
<?php  
$x = 1;  
  
while($x <= 5) {  
    echo "The number is: $x <br>";  
    $x++;  
}  
?>
```

# Arrays

In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with a numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

JavaScript is slightly simpler. It is also now common practice to declare arrays with the `const` keyword, but again you can use `var`.

```
const cars = ["Saab", "Volvo", "BMW"];
```

```
var cars = ["Volvo", "BMW"];
```

functions

# Basic function syntax

Again the **basic** function syntax is the same for JS on the left and PHP on the right

```
function myFunction() {  
    alert("Hello World!");  
}  
myFunction();
```

```
<?php  
function writeMsg() {  
    echo "Hello world!";  
}  
  
writeMsg(); // call the function  
?>
```



JS functions in more detail

Functions consist of a series of statements that have been grouped together because they perform a specific task.

A method is the same as a function except it is part of an object.

# Declaring a simple function

Use the 'function' keyword to declare a function

Give the function a name (use variable naming conventions)

```
function sayHello() {  
    console.log("Hello!");  
}
```

The statements that perform the task sit in the code block in curly braces.

All functions must have a pair of brackets after the name

# Calling a simple function

To run or call the function simply use the function name followed by parentheses.

```
sayHello();
```

```
sayHello();
```



Examples/sayHello\_function.html

You can call or reuse the function as many times as you like.

# Same function into a <p> tag

```
sayHello_function2.html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Document</title>
6  </head>
7  <body>
8      <p id="text"></p>
9      <script type="text/javascript">
10         function sayHello(){
11             document.getElementById('text').innerHTML = "hello";
12         }
13         sayHello();
14     </script>
15 </body>
16 </html>
```

# Functions can have parameters

Items in the parentheses after the name are called parameters.

These name act like variables within the function. They can accept words or numbers depending on what the function is doing.

```
function getArea(width, height) {  
    area = width * height;  
    console.log(area);  
}
```

# Calling functions that have parameters

```
getArea(3, 5);
```

You can use variables as **arguments** too, this will do the same thing.

Call the function and pass '**arguments**' to the parameters. 3 to the width and 5 to the height.

```
var wallWidth = 3;  
var wallHeight = 5;  
getArea(wallWidth, wallHeight);
```

## Function - return

```
23
24     <script>
25         // Calculate the area of a rectangle
26         function calcRectangleArea(aWidth, aHeight) {
27             let area= aWidth * aHeight;
28             return area;
29         }
30         //process the button click
31         function doCalc(){
32             let newWidth = document.getElementById("width").value;
33             let newHeight = document.getElementById("height").value;
34             let newArea = calcRectangleArea(newWidth,newHeight)
35             document.getElementById("areaCalc").innerHTML+="the area of the
36         }
37
38     </script>
```



# Interactive area calculator

Use it in the labs for reference

<examples/areacaculator.html>

<examples/areacalculator-Commented.html>

– heavily commented version

enter height

enter width

the area of the rectangle 23 by 12 is 276

