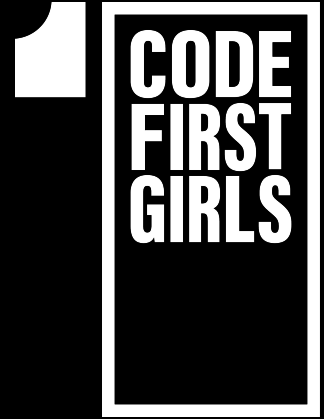


# **WELCOME TO CFG** **YOUR INTRODUCTION** **TO WEB DEVELOPMENT**



**TECH SHOULDN'T JUST BE A BOYS CLUB.**

# COURSE JOURNEY

MODULE 6: UI/UX

**MODULE 01**

HTML

**MODULE 02**

CSS

**MODULE 03**

Recap  
Project  
design

**MODULE 04**

Bootstrap

**MODULE 05**

JavaScript



**MODULE 06**

UI/UX  
Accessibility

**MODULE 07**

Github  
Pages  
Project  
work

**MODULE 08**

Project  
presentations

**What is Javascript?**

**Javascript Basics**

**DOM Manipulation**


**Complete interesting practical exercises**

# WHAT IS JAVASCRIPT?

Javascript is known as **the language of the web**. It works alongside HTML and CSS to give websites interactive features, and also is commonly used to work with stored data or API's.

It's made up of **data types, variables and functions** and is a **high-level language**, meaning that it's easier to use and incorporates some natural language elements.

We can do a vast array of things with Javascript, including adding **animation**, working with **user input**, **storing data** given to us by a user, and generally making websites **dynamic**. Dynamic websites mean that they can display different content to different users, as opposed to static websites that always stay the same.

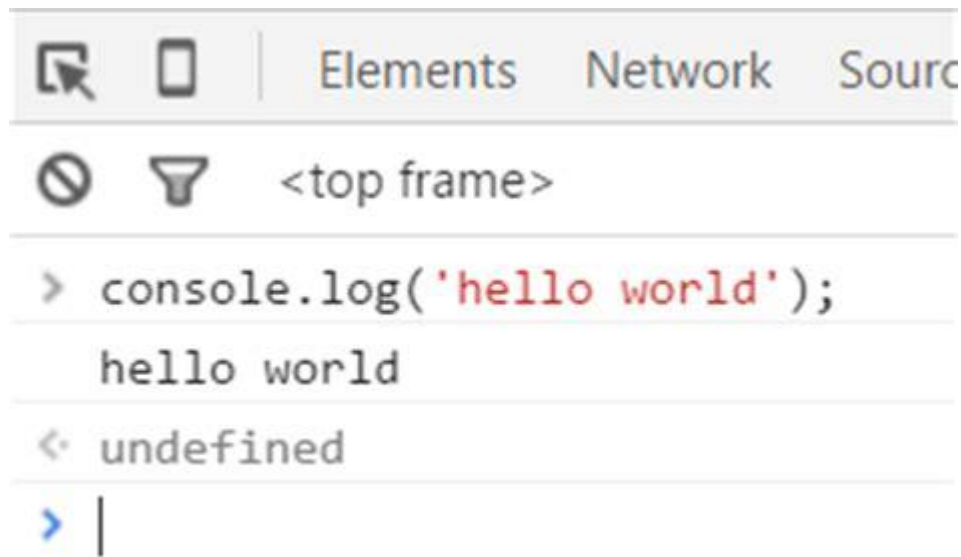
A large, bold, black 'JS' logo is centered on a solid yellow rectangular background. The letters are thick and sans-serif, with the 'J' and 'S' connected slightly at the bottom.

# THE BASICS OF JAVASCRIPT: CONSOLE.LOG()

Before we learn more about the syntax and structure of javascript as a language, we're going to look at one of its most important built-in tools.

This is `console.log()`, a function that comes built-in to javascript, meaning we don't have to create it ourselves. You can recognise that it's a function because of the brackets at the end.

`Console.log` is essentially a print statement. Whatever is inside those brackets, whether a variable or a string, i.e "Hello World" will be printed in the console.



TEST OUT CONSOLE.LOG() IN YOUR BROWSER BY USING THE DEVELOPER TOOLS UNDER THE VIEW TAB. SELECT YOUR JAVASCRIPT CONSOLE AND OFF YOU GO!

# THE BASICS OF JAVASCRIPT: VARIABLES



WHILE A VALUE SUCH AS 'CAT' COULD BE STORED IN THE FRUIT VARIABLE, IT IS ALWAYS BEST TO STORE DATA IN VARIABLES WITH APPROPRIATE NAMES (i.e PET = "CAT")

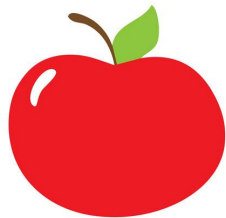
A variable in Javascript is essentially a **container** that holds **reusable data**. The most popular way to think of them is like boxes. We store different data types within these boxes, and then label them so that we can use them again and again.

For example, think of a cardboard box. We can store whatever we want inside it, like the solution to a problem, a number or anything else we can think of. If we intend to store apples, pears and bananas in the box, we might want to write 'fruit' as the label, or variable name.

The significance of the variable is its label. You can change what's inside this box, but not the label. It's only a name given to a memory location.

# HOW TO CREATE VARIABLES

We can create a variable using the **var** keyword, followed by the name we want to give to that container. We then use the = sign to assign a value to the variable. In this case, we are using the word apple, and placing it inside a box called fruit.



```
var fruit = “apple”;
```

# NOW, INSTEAD OF WRITING “APPLE” AND ADDING IT AS A NEW ITEM EVERY TIME IN OUR CODE, WE CAN JUST REFERENCE “FRUIT” AND GET THIS VALUE INSTANTLY!



# Variable Syntax & Structure

variable identifier

End of the statement

start with

assignment operator

value

```
var name = 'James Bond';
```

The diagram illustrates the syntax of a variable assignment statement. The code `var name = 'James Bond';` is shown with several annotations: a red arrow points from 'variable identifier' to the word 'name'; a red arrow points from 'start with' to the word 'var'; a red arrow points from 'assignment operator' to the equals sign '='; a red arrow points from 'value' to the string 'James Bond'; and a red arrow points from 'End of the statement' to the semicolon ';'. The words 'var' and 'name' are colored orange and purple respectively, the equals sign is green, and the string is dark red.

# ALWAYS MAKE SURE THE VARIABLE NAME MAKES SENSE AND PROPERLY REPRESENTS THE VALUE YOU WANT TO STORE!



# VARIABLES: LET AND CONST

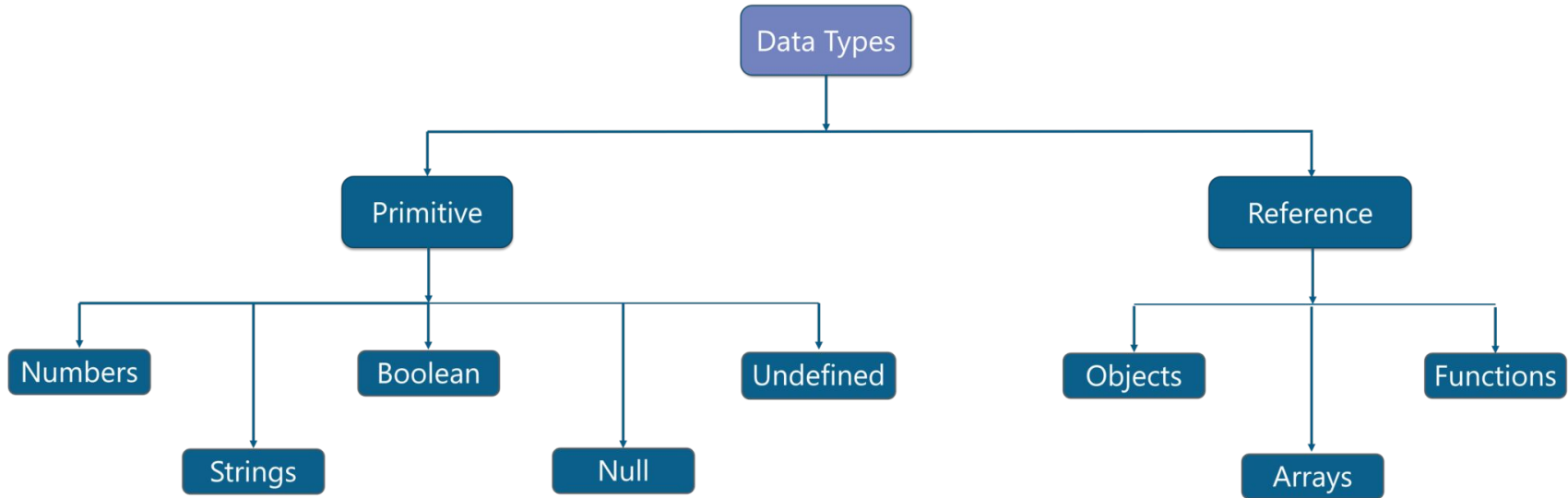
We know that we can create a variable using the **var** global keyword, which can be redefined and reassigned. We also have two different keywords for variables that we can use to not only make our code more efficient. These are **let** and **const**. The simplest way to distinguish between them is as follows:

- **Let** is the closest to var - it has a scope that can be restricted to within its own block of code. Lets are also mutable, meaning the value can be reassigned.
- **Const** is a constant value - it cannot be redefined or reassigned. It also has a local scope, meaning it is restricted to a block of code.

# MUTABLE BASICALLY MEANS CHANGEABLE.

```
1  const name = 'Nia';  
2  let age = 25;  
3  // can reassign age  
4  age = 27;  
5  // cannot reassign name  
6  name = 'Jane';  
7  
8
```

# THE BASICS OF JAVASCRIPT: DATA TYPES



Data types are just what they say they are - types of data we use in our programmes. They are divided into two categories, primitive and reference. Some of the most common data types we will be using are **numbers** (1,2,3), **booleans** (true or false) and **strings** ("isn't this fun?"). We can then assign these values to variables...

# JAVASCRIPT DATA TYPES: THE BIG 4

"Hello World"		45.892	
"Salut tout le monde"		1091	
"世界您好"	TRUE	375	
"Hola Mundo"	FALSE	2.8	No Value
<b>string</b>	<b>boolean</b>	<b>number (int or float)</b>	<b>null</b>

# JOINING STRINGS

In Javascript, there are particular ways we can join variables. In the case of strings, these methods are concatenation and interpolation.

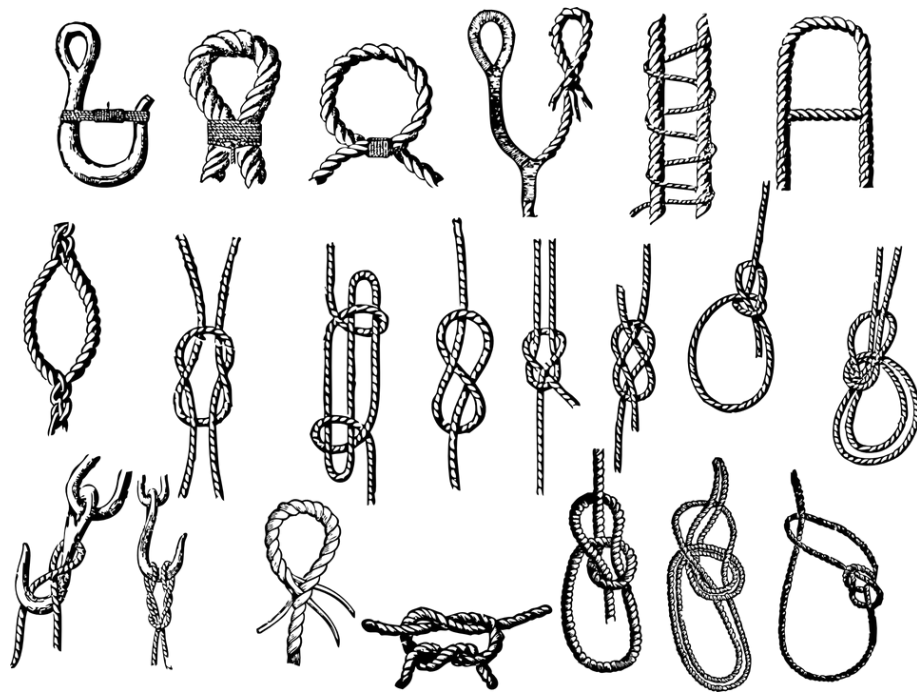
Concatenation uses the + sign to join two string values together, as you can see below.

Interpolation is when you use template literals to join pieces of data together. You can recognise template literals as they use the \$ sign, followed by the curly brackets {}.

## CONCATENATION

```
var name = 'Jane'
```

```
var greeting = 'Hi' + name
```



## INTERPOLATION

```
var name = 'Jane'
```

```
var greeting = 'Hi ${name}'
```

# LINKING JAVASCRIPT TO OUR PROJECT FILES

```
<!DOCTYPE html>
<html>
<head>
  <title>javascript link</title>
</head>
<body>
<h1>Here is your code</h1>
<script src="yourfile.js"></script>

<script type="text/javascript">

  alert('Hello coder');
</script>

</body>
</html>
```

There are two ways to use Javascript with your project files. These both rely on using the <script> tags within HTML.

## OPTION ONE:

We create a .js file within our project folder. Then, within our html page, we identify a script source, selecting the new js file we've created. And done! Now everything you have in that file can be accessed within the HTML doc.

## OPTION TWO:

We create some Javascript code within the body of our HTML doc itself. We specify the type, i.e. javascript and then within those tabs, we create our code directly.

# NOW LET'S PRACTICE TOGETHER

## LINKING JS AND CREATING VARIABLES

MODULE 5: JAVASCRIPT

5 MINS

### Exercise 1.0 - LINKING JS

\*Create an HTML file and a JS file and link them to each other

### Exercise 1.1

Within the JS file, create a new variable called name and assign it with the value of your name. Use your new knowledge of var, let and const.

### Exercise 1.2

Create a new variable and store your age.

### Exercise 1.3

Create a new variable called "future job" and assign it "web developer".

### Exercise 1.4

Connect these bits of information together with either concatenation or interpolation, so that it creates the phrase "My name is (name), I am (age) years old and I will soon become a web developer."

9 MINS

### GROUP EXERCISE



# CONDITIONAL STATEMENTS: IF / ELSE STATEMENTS

A conditional statement is a set of rules performed if a certain condition is met. This helps us gain more control over our code. For example, if A is true, do this. If B is true, do that.

This type of conditional flow is made up of three distinct parts, and relies on boolean values, i.e. true or false.

**IF** = The if statement is used when we want a block of code to be run as long as the condition is true.

**ELSE IF** = This condition runs when the condition before it is false.

**ELSE** = The else condition only runs if everything before it is false.

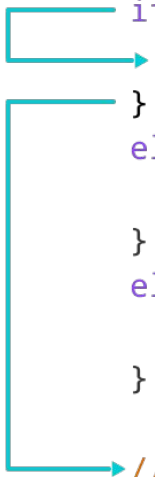


# JUST THINK OF CONDITIONAL STATEMENTS AS GIVING OUR CODE OPTIONS

# CONDITIONAL STATEMENTS: IF / ELSE STATEMENTS

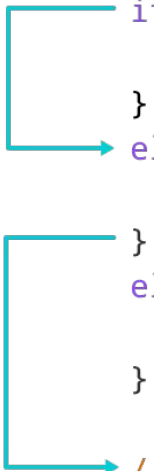
## 1st Condition is true

```
let number = 2;  
if (number > 0) {  
  // code  
}  
else if (number == 0){  
  // code  
}  
else {  
  //code  
}  
//code after if
```



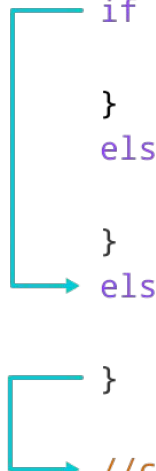
## 2nd Condition is true

```
let number = 0;  
if (number > 0) {  
  // code  
}  
else if (number == 0){  
  // code  
}  
else {  
  //code  
}  
//code after if
```



## All Conditions are false

```
let number = -2;  
if (number > 0) {  
  // code  
}  
else if (number == 0){  
  // code  
}  
else {  
  //code  
}  
//code after if
```





# NOW LET'S PRACTICE TOGETHER

## CONDITIONAL STATEMENTS

MODULE 5: JAVASCRIPT

4 MINS

### Exercise 2.0

Let's create an if/else statement! First, create a variable called `allergic_to_prawns`. Give it the value of `TRUE`.

### Exercise 2.1

Imagine the scenario that we are at a seafood restaurant. If we're allergic to prawns, print the following statement - "Guess we're having breadsticks!"

### Exercise 1.2

If we're not allergic to prawns, print the following statement - "Bring on the shellfish!"

### Exercise 1.3

Change the `allergic_to_prawns` variable and run the code again. See what happens?



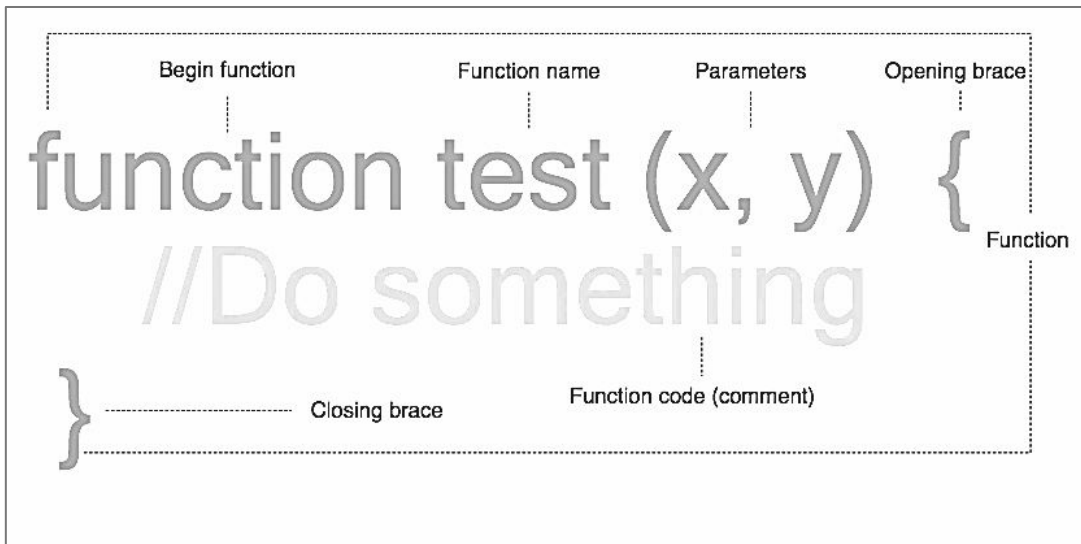
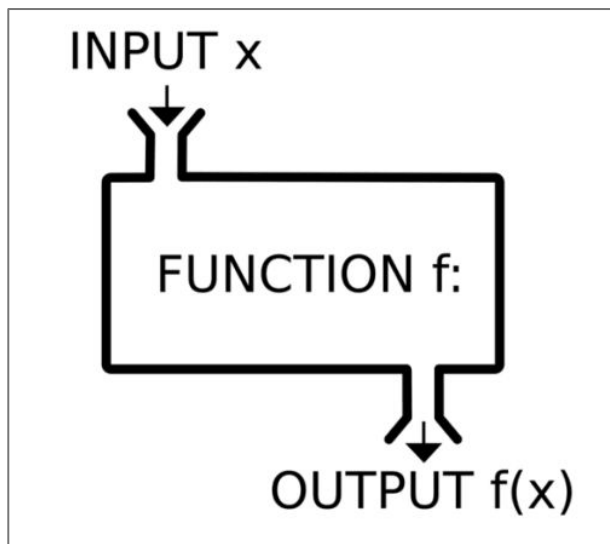
4 MINS

GROUP EXERCISE



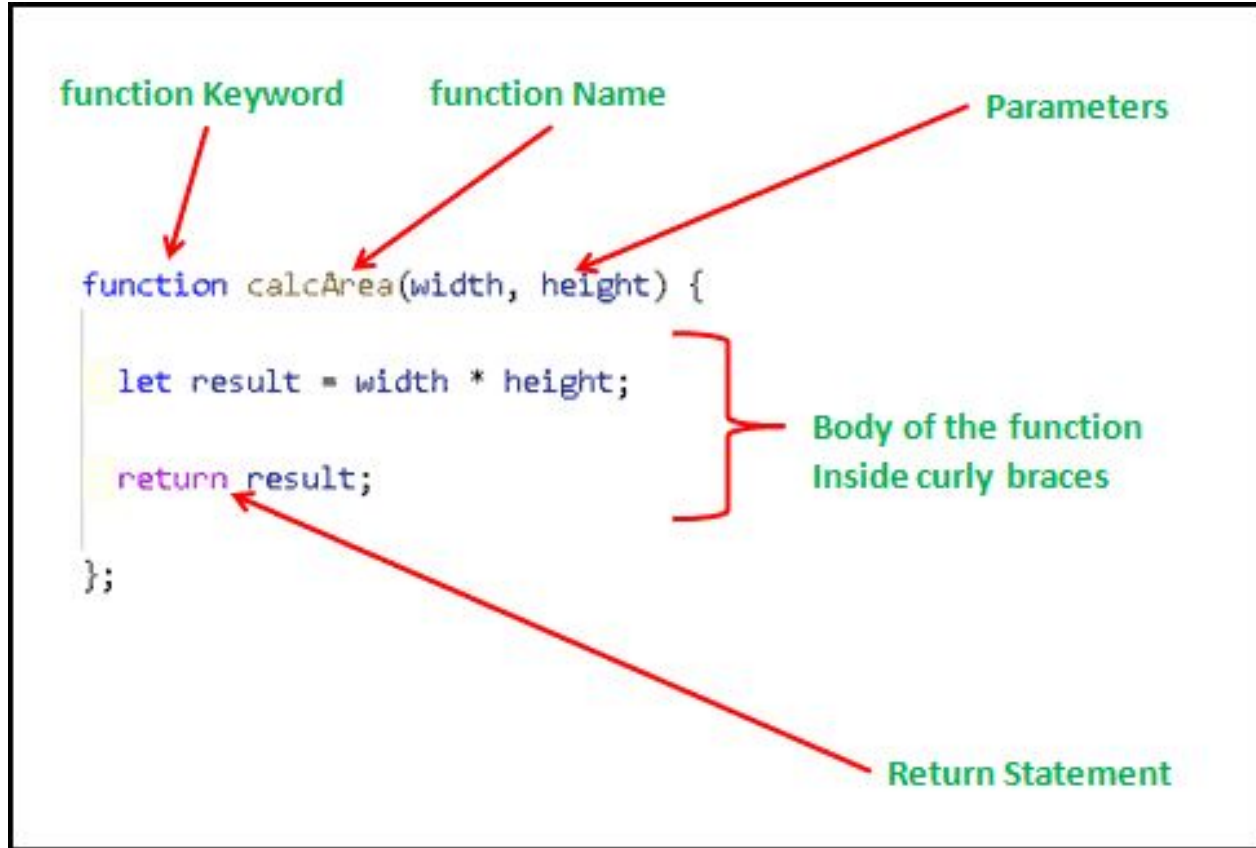
# JAVASCRIPT BASICS: FUNCTIONS

A function is basically a reusable block of code. It can include its own variables and operations, and often has parameters so that you can feed it information and get new information in return. Functions are some of the most powerful elements in any programming language. `Console.log()` is one such function we've seen already. You can spot a function by the use of these brackets. In its simplest terms, think of a function as a reusable tool to perform an action or operation.





# Function Syntax & Structure



# DOM MANIPULATION WITH JAVASCRIPT

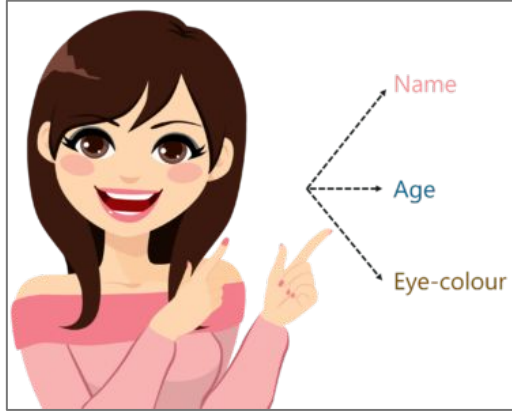
One of the most exciting things about learning Javascript is something called DOM Manipulation. The HTML DOM is the **Document Object Model**, the object created when a webpage first loads that contains all of the information relevant to the construction of that page.

Through this model, Javascript gains the power to alter, or manipulate elements, styles and information. The DOM is a standard format for how we can get, change, add or delete elements.

This also includes creating events, for example, adding interaction whenever a button is clicked, animating an element or storing data given to us by a user. In short, it brings our website to life!



# What's an Object?



## ACCESS VALUE

```
girl['name'];  
girl.age;  
girl.friends[0];  
girl['address']['street'];  
girl.address.city;  
girl.sayHello();
```

In its simplest form, an object is a data structure used to store information. It is made up of key value pairs, in the same way a variable stores a value. In fact, think of an object as a variable, just one that can store many values.

So, instead of having separate variables for name, age, height, etc, we could instead store all of these in one object and call it "girl" or something to signify what these attributes belong to.

Then, instead of referencing multiple variables one by one, we just need to access them using the object name, followed by the key. For example, `girl['name']` would bring up the girl's name.

It's a good idea to utilise these to simplify storing and accessing data!

# Example of a Javascript Object

```
let girl = {  
  name: 'Anne', // string  
  age: 23, // integer  
  friends: ['Susan', 'Marie', 'Jenny'], // array  
  address: {  
    //object  
    number: 123,  
    street: 'Cambridge Road',  
    city: 'London'  
  },  
  sayHello: function() {  
    // function  
    console.log('Hola Amigos!');  
  }  
};
```

KEY IDENTIFIERS:

USES {} CURLY BRACKETS

KEY-VALUE PAIRS  
(I.E. FRIENDS: (NAMES))

CAN STORE DIFFERENT DATA TYPES  
IN ONE AREA  
(NUMBERS, LISTS, STRINGS)

IT CAN ALSO STORE FUNCTIONS  
SPECIFIC TO THAT OBJECT, WHICH  
ARE KNOWN AS **METHODS**.

# DOM Events

When \_\_\_\_\_ happens, do \_\_\_\_\_.

When a page load happens, do play the video of a cat sliding into cardboard.

When a click happens, do submit my online purchase.

When a mouse release happens, do hurl the giant/not-so-happy bird.

When a delete key press happens, do send this file to the Recycle Bin.

When a touch gesture happens, do apply this old timey filter to this photo.

When a file download happens, do update the progress bar.

# Event Listeners

Event listeners are an important part of DOM manipulation. They're exactly what the name suggests - lines of code we create to listen for certain events to happen, and then trigger reactions. There is a two part process for creating an event listener:

- We need to specify the element we're watching - in this case, called "ourElement":

We do this by using the **document.getElementById("ourElement")**

- We add our event listener, and specify the kind of event we're looking for, as well as what we want that to trigger:

**.addEventListener("click", displayTime);**

Here, we are activating the displayTime function when this is clicked. Put that all together:

**document.getElementById("ourElement").addEventListener("click", displayTime);**



# Example of DOM Manip: The Carousel

One example of DOM manipulation can be found in the bootstrap import you've already learned about. The picture carousel in bootstrap utilises a javascript plugin to work. This javascript plugin through the .carousel class allows us to cycle through elements, creating an interactive gallery.

When we import the bootstrap javascript bundle, we can use these class identifiers to manually switch between pictures - changing the information being displayed in the active tab / section of the gallery. Let's look at an example...

```
<script src="script.js"></script>
<!-- Bootstrap JS -->
<!-- JavaScript Bundle with Popper -->
<!-- this allows us to use the carousel -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"
```

# NOW LET'S PRACTICE TOGETHER

## DOM MANIPULATION

MODULE 5: JAVASCRIPT

5 MINS

### Exercise 3.0

Let's create a button in our HTML file. We can give this element the id "myButton" and have the text say "Click me!"

### Exercise 3.1

Create a function called `pressAlert()`. The purpose of this function is to print an alert to the console that a button has been clicked.

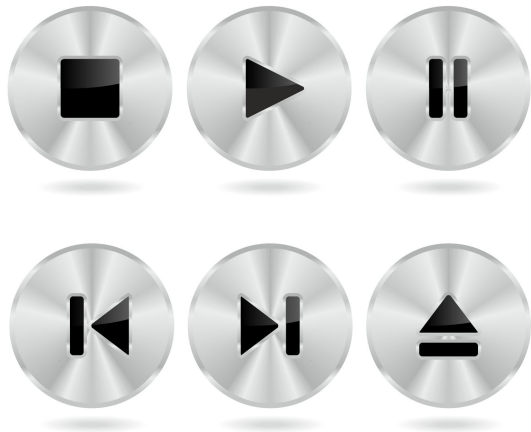
### Exercise 3.2

Write an event listener within your Javascript file to prompt this function when the button is clicked. Edit the button itself with an `onClick()` statement in the tags of the HTML element.

### Exercise 3.3

Save this js alert to a variable called 'alert'

4 MINS



### GROUP EXERCISE



# HOMEWORK

# "Hello World"

## + Homework Task

Add some interaction to your website files using javascript. Create a function that takes a name input from the user and displays it in a "Hello" alert statement. Think about input types, buttons and more!

**THANK YOU**  
**HAVE A GREAT**  
**WEEK!**

