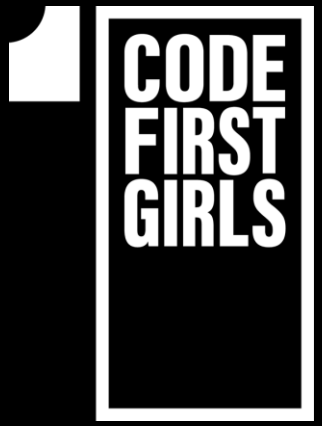
WELCOME TO CFG

YOUR INTRODUCTION TO WEB DEVELOPMENT



TECH SHOULDN'T JUST BE A BOYS CLUB.

COURSE JOURNEY

MODULE 01	MODULE 02	MODULE 03	MODULE 04	MODULE 05	MODULE 06
HTML	CSS	Recap Project design	**Javascript** + Overview & Data-types	Github pages Frameworks	Project presentations Careers in web development

04

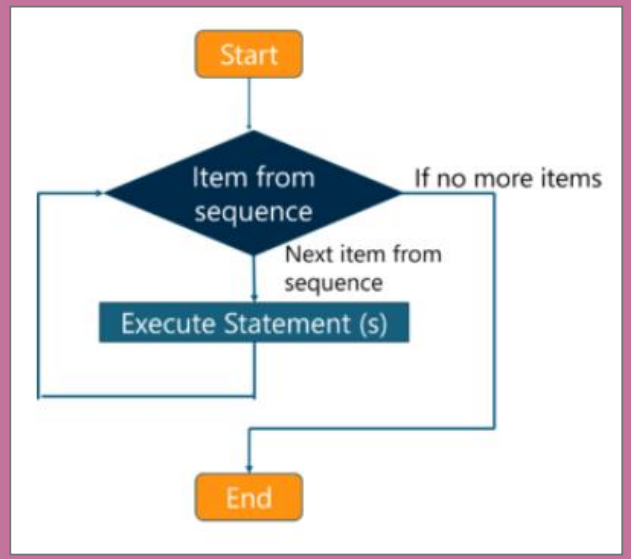
JAVASCRIPT PART 2

JavaScript coding practice

Programming concept: ‘FOR LOOP’

JavaScript: writing functions

FOR LOOP



The FOR loop repeats condition a certain number of times

```
for (begin; condition; step) {
   loop code;
}
```

```
for (i=0; i<5; i++) {
   console.log("The number is " + i);
}
```

NOW LET'S PRACTICE TOGETHER

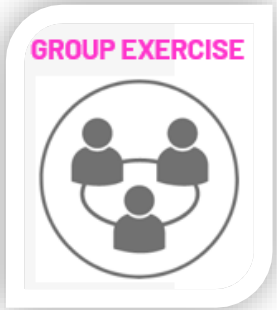
5 MINS

Exercise 5.1

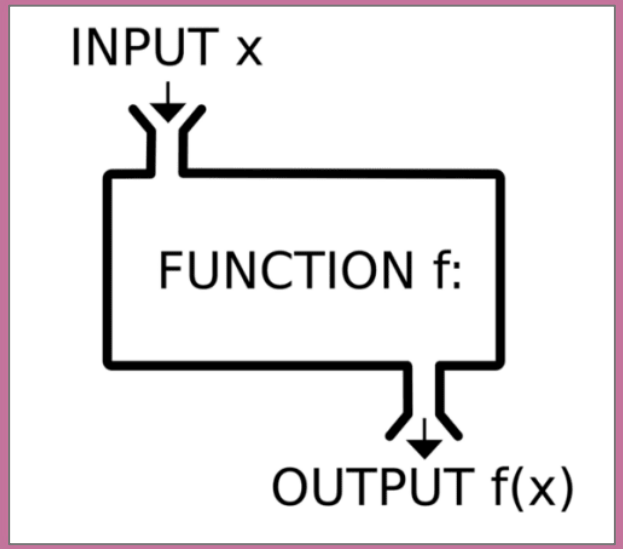
We've seen how a for loop can start at 0 and count up

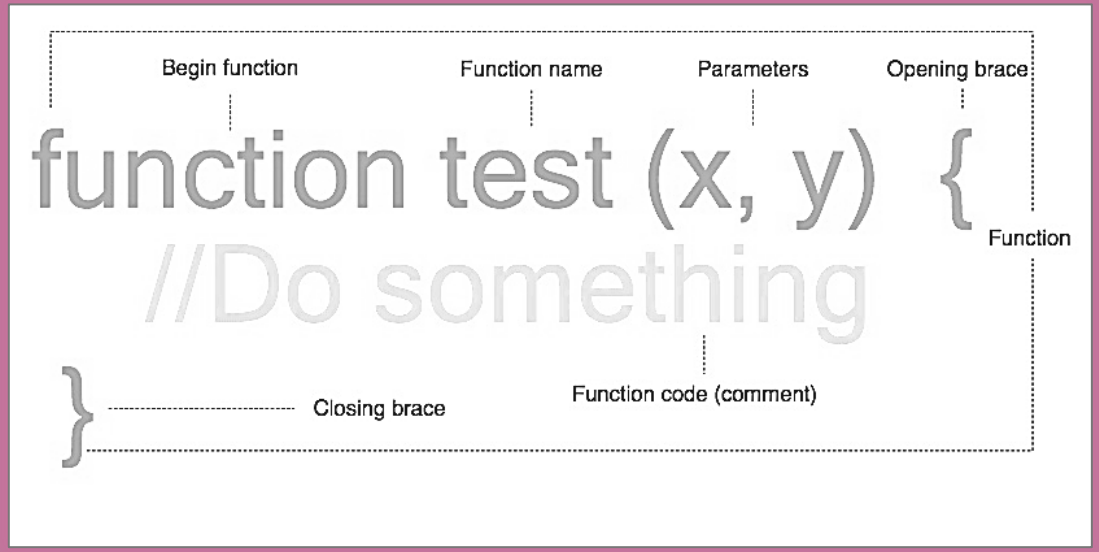
Create a new for loop which counts down instead

Extension: Have the console.log() do some simple arithmetic (addition, subtraction etc) on each loop around



FUNCTION





A function is a block of organised, reusable code that is used to perform single, related action

```
function product(a, b){
   return a * b;
}
```

```
product (8, 2);
```

```
var value = product (8, 2);
```

INSTRUCTOR DEMO

PRACTICE ALONG WITH THE INSTRUCTOR

Exercise 5.2

Create a function called **myName** that uses a **prompt** to get the users name

Call that function and return the value in a variable called **name**

Display that value in a **console.log**

Questions?

FUNCTION ARGUMENTS EXAMPLE

```
function product(a, b){
  return a * b;
}
```

```
var value1 = product (2, 4);
```

```
var value2 = product (3, 5);
```

```
var finalResult = product (value1, value2);
```

```
var finalResult = product (8, 15);     // finalResult = 120
```

INSTRUCTOR DEMO

PRACTICE ALONG WITH THE INSTRUCTOR

Exercise 5.3

Declare a new function called **movieDetails**

Give it two arguments - **title** and **released**

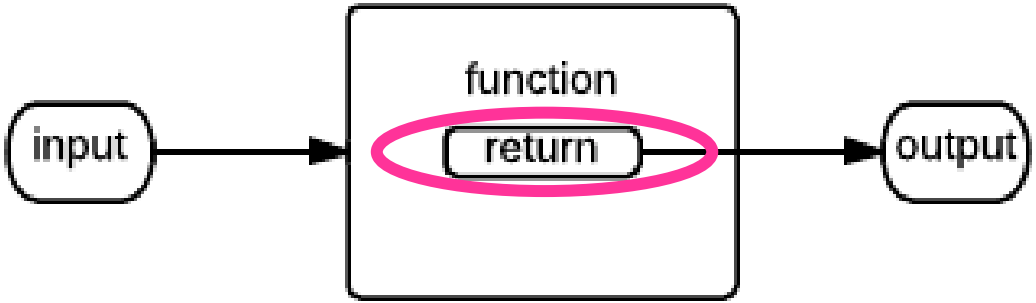
Inside the function body, **return** a sentence that includes both arguments

Create a variable called **movie1** and invoke the function giving it some movie details

Console log the variable

Questions?

FUNCTION LOGIC BODY




```
function verifyDiscount(name, age){
   if (age < 18){
      return "Child discount applied for " + name;
   } else {
      return name + "  is not eligible for discount";
}
```

INSTRUCTOR DEMO

PRACTICE ALONG WITH THE INSTRUCTOR

Exercise 5.4

Write another function with control flow called **shoppingCart** which decides whether or not you can afford a new item of clothing

It should take 3 arguments: **item**, **cost** and **balance** and **return** whether or not you can afford it based on your **balance**

Questions?

FUNCTION SCOPE

LOCAL SCOPE

```
// code here can NOT use animal

function myFunction() {

    var animal = "Fox"

    // code here CAN use animal
}
```

GLOBAL SCOPE

```
var animal = "Fox"

// code here can use animal

function myFunction() {

    // code here can also use animal
}
```

INSTRUCTOR DEMO

PRACTICE ALONG WITH THE INSTRUCTOR

Exercise 5.5

Find the folder: exercise 5.5 -starter-code in Slack

Download the code, unzip and move it to a similar location

Open the entire folder (not individual files) in VScode

Go through the code and follow along with the instructions in app.js

Questions?

HOMEWORK

+ **Homework Task:** Build a Calculator

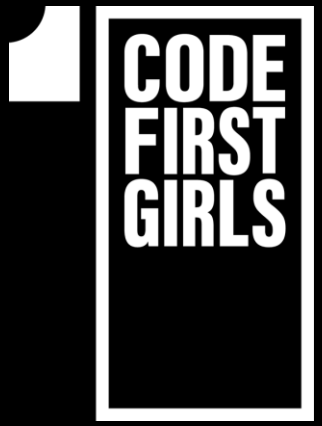
Write the calculator functions for the following operations

- Sum
- Subtract
- Divide
- Multiply
- Power
- Square Root

+ **Extended Learning Task:** Calculator with user input

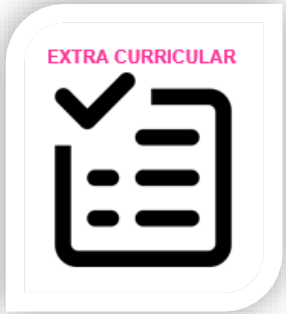
Use prompt() to get user information to do the calculations and alert() to display the result

THANK YOU
HAVE A GREAT WEEK!

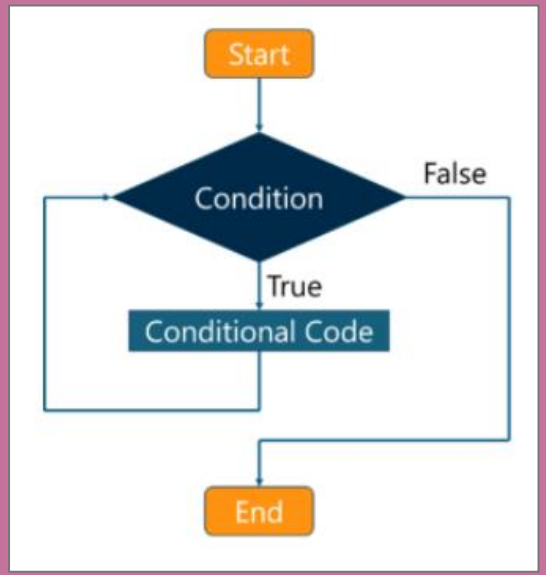


SELF STUDY TOPICS

THESE CONCEPTS ARE USEFUL TO KNOW –
WE ENCOURAGE YOU TO SELF STUDY



WHILE LOOP

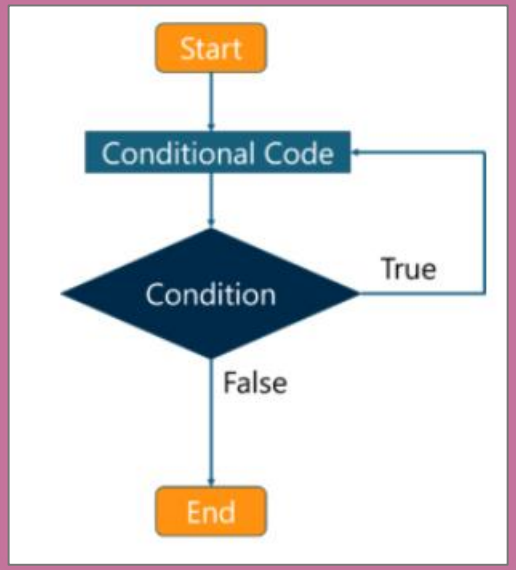


While the **condition is true**, the code within the loop is executed

```
while (condition) {
  loop code;
}
```

Research and write some examples for WHILE loop

DO WHILE LOOP



This loop will **first execute the code**, then check the condition and while the condition holds true, execute repeatedly.

```
do {
   loop code;
} while (condition)
```

Research and write some examples for WHILE loop

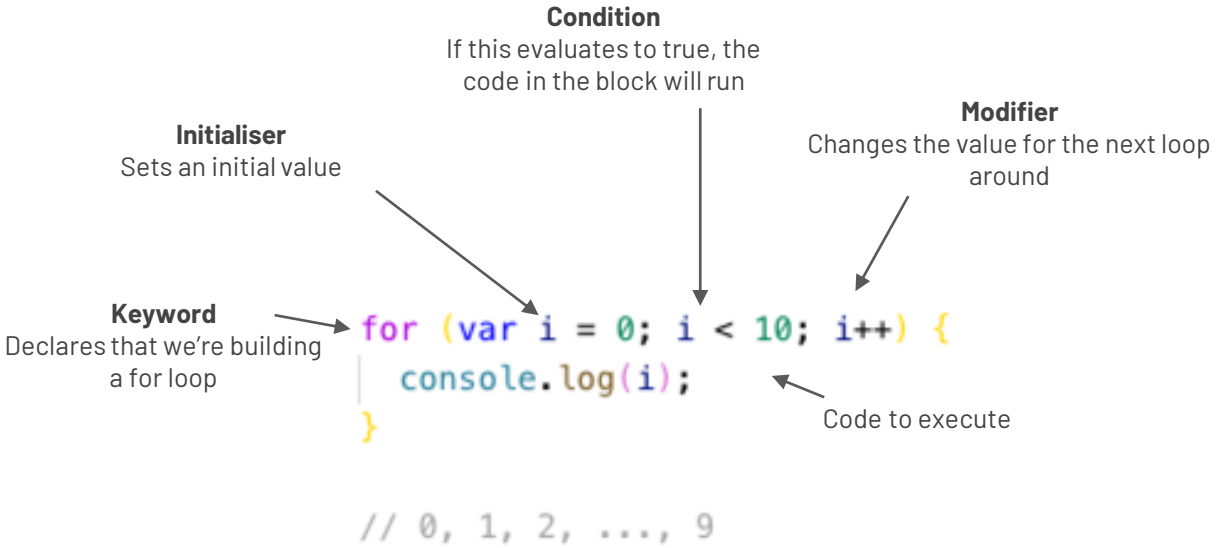
REFERENCE MATERIALS

LOOPS

Used to repeat a process a certain number of times

Not exclusively used with arrays, but particularly well suited to them

There are other loops that will require your own research



LOOPS


```
for(initialise; check condition;
modifier if condition is true)
for (var i = 0; i < 10; i++) {
  console.log(i);
}

0,1,2,3,4,5,6,...9

When using with arrays, we use the
array length to get the index
for (var i = 0; i < fruit.length; i++)
{
  console.log(fruit[i]);
}

'apple', 'banana', 'pineapple', 'pears'
```

FUNCTIONS

Making Code reusable

A function is a crucial feature of almost every single programming language

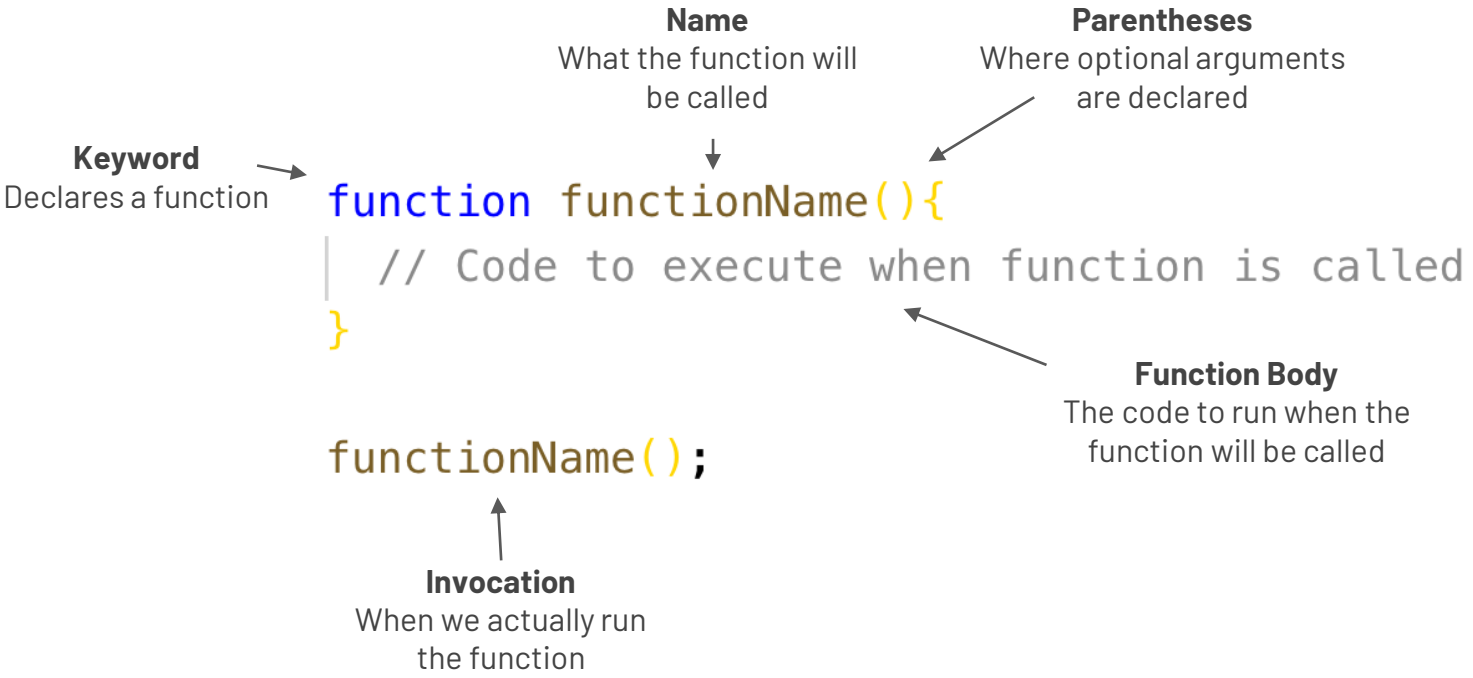
A function body is defined to carry out more of one processes, and then that function can be called as many times as you need

This help us to keep our code D-R-Y (Don't Repeat Yourself)

Less code means it runs more efficiently which means it's **faster** and **cheaper** at enterprise levels

FUNCTIONS

Anatomy of a function



WRITING A FUNCTION

Declare a new function called helloWorld

Inside the function block, console log a **string** “Hello World”

If you run the code then you’ll notice nothing happens. This is because we haven’t **called** or **invoked** the function

Underneath, call the function with **helloWorld()** and you’ll see “Hello World” in the console!

```
function helloWorld() {
  console.log('Hello world');
}

helloWorld();

// 'Hello world'
```

RETURNING A VALUE

There are many cases when we will need to store the result of a function (eg a calculation)

The return keyword is how we do this

We can save the returned value in a variable and then pass that variable around our code

```
function returnHello() {
  return 'Hello world';
}

var greeting = returnHello();

console.log(greeting);

// 'Hello world'
```

ARGUMENTS

Simply put, arguments are placeholders for actual values

Arguments are where functions can really demonstrate their reusability in writing DRY code

In the block {...} we treat the argument as the data type we EXPECT it to be

We can pass variables as arguments too

```
function sum(number1, number2) {
  return number1 + number2;
}

var sumAnswer1 = sum(2, 4);
console.log(sumAnswer1); // 6

var sumAnswer2 = sum(12, 3);
console.log(sumAnswer2); // 15

// We can even pass variables as arguments when we invoke the
function
var sumAnswer3 = sum(sumAnswer1, sumAnswer2);
console.log(sumAnswer3); // 21
```

ADDING LOGIC TO A FUNCTION

Very often we will want a function to return different things depending on what we pass to it

In this example, we are passing **name** (which will be a string) and **age** (which will be a number)

We check if the age is under 18: if true, we return a concatenated string with a message saying they're old enough

If false, then we return a concatenated string saying they have to be 18 to drive

```
function verifyAge(name, age) {
  if (age > 18) {
    return name + ' is old enough to drive';
  } else {
    return name + ' has to be 18 to drive';
  }
}

var John = verifyAge('John', 16);
console.log(John); // 'John has to be 18 to drive'
var Jane = verifyAge('Jane', 20);
console.log(Jane); // 'Jane is old enough to drive'
```

WHAT IS SCOPE?

Scope, in essence, is what a function can **see** when it is invoked

Functions have their own scope, meaning that any variables declared within the block are only visible **within** that block

If that doesn't make sense, don't worry: scope can be tricky

A basic rule of thumb is that code can look out but not in

Think of it like a 2-way mirror - variables **outside** a function see the mirror, variables **inside** the function see the glass

In this example, Lucy can be **'seen'** from inside the fence, but **prisoner1** is showing as **not defined**

```
var freePerson = 'Lucy';

function outerFence() {
  var prisoner1 = 'Tim';
  console.log(freePerson);
}

console.log(prisoner1);
```