Notes

Making a website, Unit 3

Introduction to a function

A function is a small bit of code that can be called and ran any point in your code. If you are going to use a bit of code all over your code, then you can just create that code in a function and call the function. They also accept parameters and will always return exactly 1 value.

Layout of a function

There are 4 parts of a function that you need to make.

The keyword function: function

The name of the function: funtionName

The parameters of the function: parameters

Lastly the body of the function that holds the core

Function functionName (parameters){

//function body

}

Calling a Function

Once you have created your function you can call them anytime with any parameter that you have created. You can do this by calling their name.

functionName (parameters);

The parameters

Unlike other code you can use a variable that is yet to be declared by passing it throught the parameters. This lets you accept many different inputs that can be used to change the outome of the function.

```
function showMessage (from, text){
    console.log(from +  ': ' + text)
}
showMessage('Ann', 'Hello!'); // Ann: Hello!
```

Function set up

```
function calcMath(number){
    number *= 4;
    number *= 13;
    return number;
}
console.log(calcMath(10))
```

onClick

Thye easiest way to incorporate this into your website is using the onClick event listener. This will allow you to call a function in the HTML to run some code on a button click.

<button onclick = “functionName()” > Click Me! </button>

Local Variables

If a variable is called in a function then the variable is called a local variable and can only be accessed inside of that function. This is kind of like how we use the “I” variable for every loop, withought conflict. We are able to use the same variable over and over in the function.

Global Variables

A global variable exists outside of the function and can be accessed and modified by all functions. If you try to call the same named variable in a function it will use the local version and not overwrite the global version. There are reasons to use both local and global variables so it is really up to the situation which is correct.

Default Values

When creating your parameters you can set them equal to something to make sure there is a default value. Then if nothing is entered in the parameters they will still have a value to input.

```
Function showMessage(from, text = “no text given”){

Console.log(from + “: “ + text);

}

showMessage(“Ann”); //Ann: no text given
```

Returning a Value

You can make a function assign a value as well. This used using the “return” keyword. There can only be one return value and as soon as it happens the function will end. No matter what. The best part of returning a value is the ability to directly assign that value to a variable.

Proper Naming for Functions

Since functions do something and serve a purpose you need to start using proper verbs:

“getFunction”: returns a value

“calcFunction”: calculate something

“createFunction”: creates something

“checkFunction”: checks something and returns a true or false

Commenting

Functions are usually very short and do a thing, only 1 thing. Because of this, there are usually lots of functions that are created and used in longer code and therefore your functions MUST HAVE COMMENTS. It may also be better to split your functions into several smaller functions.

Function Expression

A function expression is when you create your function inside of a variable of sorts. Ex.

```
Function sayHi (name, text){

Console.log(sayHi(name, text));

}

Let sayHi = function(name, text){

Console.log(sayHi(name, text));

}
```

Expression vs Declaration

Declarations (The first way we learned) has something called “hoisting”, which means that it can be used before it is created.

```
//this works

doStuff()

function doStuff(){}
```

Expression don’t have hoisting but they can be created and forgotten after use, saving your computer some brain power.

When to use each

This really depends on the code you want to run. Functon declarations are used if you want something usable in your entire code. Function expressions are used when you want something sm,aller and to be used only where the function is available.

Arrow Functions

Arrow Functions are a nicer way of writing functions expressions. These are not more or less efficient, they are just a more compact way of coding a function.

Basic Arrow Function

When creating the arrow function you can leave out the keyword function. The => tells the code what is returned from the function.

Ex

```
Let sum = function(a, b){

Return a + b;

}

//is the same as

Let sum = (a, b) => a + b;
```

More Ex>

```
Let double = function (n) {

Return n * 2;

}

//same as

Let double = n => n * 2;
```

Multiline Arrow Functions

You can use arrow functions for multiple lines, just like rgular functions.

```
Let sum = (a, b) => {

Let result = a + b;

Return result;

}

Console.log(sum(3, 5));
```

Use Strict

Every js file should start with the phrase “use strict”;

This is to make sure that the js code uses the modern versions for behavior and not the outdated older versions, called “compatibility” versions. This wont break the code but if you ever want to turn js intoan OOP instead of OBP language, this is how you do it.

//unrelated but Python is the most developed coding language because it is the closest to english