Callback Functions

Justice Reskill

What are Callbacks

Callbacks are when a function is passed into another function as a parameter, and the parent function will execute the passed in function, and potentially pass in variables to the child function.

Callbacks are an important and fundamental concept in Javascript.

Passing A Function As A Parameter

Callbacks start with passing one function into another function. When a function is a parameter we call it with the '()'.

Take an important note, in this example we are defining functions and assigning it to a variable.

```
//Define a function
var childFunction = function() {
        console.log('Hello world');
}

//Define another function with a parameter
function parentFunction(param1) {
        //Call the function
        param1();
}

//pass the child function into the parent function
parentFunction(childFunction);
```

Anonymous Function

An anonymous function is a function that is created on the fly. In the previous example, we assigned a function as a variable. As a variable, we can call the function anytime.

In this example, we are defining the function in the parameter! This means this function will only exist when the code is executed, and cannot be called again.

```
//Define parent function
function parentFunction(param1) {
        //Call the function
        param1();
}

//create the function the fly as the parameter
parentFunction(function() {
        console.log('Hello world');
});
```

Calculator Example

In this example, we've create a simple calculator object.

The object has one function called execute, which accepts 3 parameters. The first two parameters are passed into the 3 parameter, which is an assumed function.

```
var calculator = {
        execute : function(num1, num2, operand) {
                return operand(num1, num2);
        }
}
```

Adding/Subtracting

In this example, we define the functions add and subtract. And then we are able to pass them into the calculator object.

```
function add(num1, num2) {
        return num1 + num2;
}

function subtract(num1, num2) {
        return num1 - num2;
}

var calculator = {
        execute : function(num1, num2, operand) {
                return operand(num1, num2);
        }
}

calculator.execute(2,2, add);

calculator.execute(3,2, subtract);
```

Anonymous Multiplication

In this example, we create the multiplication callback function without assigning it to a variable. This is an anonymous function being created on the fly and passed as parameter.

```
var calculator = {
        execute : function(num1, num2, operand) {
                return operand(num1, num2);
        }
}

calculator.execute(2,2, function(num1, num2) {
        return num1 * num2;

});
```

Extending Functionality

With callbacks we can extend our functionality to meet almost any situation.

For example, the area of a triangle is ½ * base * height. Can we extend our calculator to calculate that?

```
function areaOfTriangle(num1, num2) {
        return .5 * num1 * num2;
}


var calculator = {
        execute : function(num1, num2, operand) {
                return operand(num1, num2);
        }
}

calculator.execute(2, 2, areaOfTriangle);
```

Numerous Callback Functions in Javascript

In Javascript, there are various instances where you will use a callback function. We are going go over a few them to help solidify the concept of callback functions.

Timeout

The timeout function is a built-in function that will execute after a certain amount of milliseconds. The function as two parameters:

setTimeout(callback,milliseconds);

Read more at:
https://www.w3schools.com/jsref/met_win_settimeout.asp

```
//Will print hello world after 3 seconds
setTimeout(function() {
        alert("Hello World!");
}, 3000);
```

Interval

The interval function will repeatedly call a function every given amount of milliseconds.

setInterval(callback,milliseconds);

Read more at:
https://www.w3schools.com/jsref/met_win_setinterval.asp

```
//Will print hello world every 10 seconds
setInterval(function() {
        alert("Hello World!");
}, 10000);
```

Event Listeners

Remember when in html we had an onclick event in html call javascript?

```
<button type="button" onclick="clickedButton()" >
        Click Me
</button>
```

We can have the same functionality in the DOM with Javascript's listener function.

addEventListener(event, callback);

Read more at:
https://www.w3schools.com/jsref/met_document_addeventlistener.asp

```
//Create an html element
var button = document.createElement('button');

//Add an event listener to that button
button.addEventListener("click", function() {
        alert("You clicked me ");
});
```

Array ForEach

Instead of iterating over an array with a for loop, arrays have a neat built in function call forEach. This function uses a forEach loop to iterate over the items in array.

foreach(function(currentValue, index))

Read more at:

https://www.w3schools.com/jsref/jsref_foreach.asp

```
//Create an array
var items = [1,2,3,4,5,6,7];

items.forEach(function(value, index)) {
        //Prints the current value
        console.log(value);
});
```