Javascript Callback Functions

by Remigijus Kutkaitis · Jun. 02, 16 · Web Dev Zone

Lately, I had issues trying to understand callback functions. I didn't quite get it how they work and how to use them in Javascript code. Here I'll try to explain them as simply as possible. I hope it helps someone.

Javascript Functions

In Javascript functions are objects, like strings, arrays, etc. So if it is an object, this means that you can pass wherever you want, as you would with every object. For example:

```
var calculateArea = function (a,  b) {
var area = a * b;
console.log("Calculated area of "+ a + " and " + b + " is equal to: " + area);
return area;
}

function rectangleCalculation(a, b, calculateFunction){
calculateFunction(a, b);
}

rectangleCalculation(5, 4, calculateArea);
```

In this short example, we see that the "calculateArea" function is passed to the "rectangleCalculation" function as one of the parameters, and it's called inside it. It's very convenient if you have some abstract functions with some common functionality and need to extract some specific functionality. For example:

```
var calculateArea = function (a,  b) {
var area = a * b;
console.log("Calculated area is equal to: " + area);
return area;
}

var calculatePerimeter = function (a, b) {
var perimeter = 2 * (a + b);
console.log("Calculated perimeter is equal to: " + perimeter);
return perimeter;
}

function rectangleCalculation(a, b, calculateFunction){
```

```
14  console.log("Input parameters: a = " + a + "; b = " + b + ";");
15  calculateFunction(a, b);
16  }
17
18  rectangleCalculation(5, 4, calculateArea);
19  rectangleCalculation(5, 4, calculatePerimeter);
```

At this point, I moved out parameters logging to the "rectangleCalculation" function as a rectangle cannot have more than two variables, a and b. I also added another function for perimeter calculation. From this you can see that I do not need anything to change, I can pass whatever I want to the "rectangleCalculation" function.

Javascript Callback Functions

A callback function is a function which is passed to another function, and it is it's responsibility when to call a passed function. This is very convenient when performing long I/O operations, as it allows asynchronous behaviour. Callback functions are widely used in most Javascript frameworks. In the beginning, especially for those who programmed in Java or a similar language, it's a little bit weird, because from a code sequence you cannot be sure that it will be executed line by line. For example:

```
1   var callDelayFunction = function(callbackFunction) {
2   setTimeout(function() {
3   callbackFunction();
4   }, 3000);
5   };
6
7
8   console.log("Starting");
9
10  callDelayFunction(function() {
11  console.log("Callback function");
12  });
13
14  console.log("Ending");
```

Here from the first look at the code, it looks like "Starting", "Callback function", and "Ending" should be called, but actually "Starting", "Ending", and, after 3 seconds, "Callback function" is called. So when writing code, you need to keep in mind, especially when it involves communication with database or files I/O, that the next code which will be executed must work without data, as it will usually not be presented yet.

Conclusion

Callback functions are very useful as they allow you to perform time-consuming operations without blocking code execution. Of course, it has some drawbacks, and you need to change your thinking a little bit if you have worked with Java or a similar language. But this is a good thing, as stepping out of your comfort zone always brings a lot of benefits.

Code from examples can be found here. The easiest way to launch it is using nodeJS, e.g. *node RectangleCalc-1*.

Topics: JAVASCRIPT, CALLBACK IN JAVASCRIPT, PROGRAMMING