Advanced JavaScript

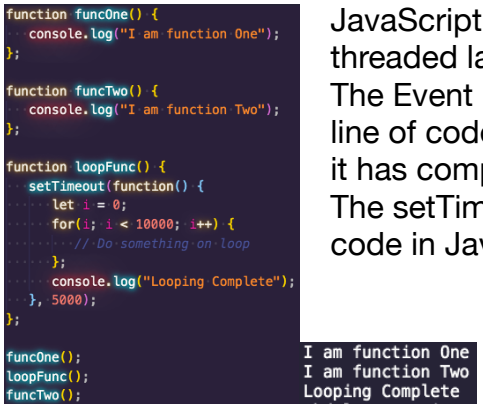
Section 3: Synchronous and Asynchronous Execution, Callbacks and Higher Order Functions & Strict Mode

---

1.1 Synchronous and Asynchronous Execution

JavaScript at its core is a synchronous language which mans that it executes code in sequence i.e. line by line. This therefore means the JavaScript code will not execute the next line of code (i.e. statement) until the first line of code has completed execution. For example if we had two functions of functionOne and functionTwo, the functionTwo cannot execute until functionOne has completely finished its execution.

The downside of synchronous code is that it can block execution of the next code until the current function is completely finished. Asynchronous code on the other hand allows multiple code to run at the same time asynchronously without having any issues of code blocking. This therefore allows different code to execute and complete at different times regardless the order the function was called.



JavaScript can have asynchronous execution but it is still a single threaded language and this is due to the JavaScript Event Loop. The Event Loop does not block execution and will allow the next line of code to execute and will only call the callback function once it has completed its execution and has returned something back. The setTimeout( ) API allows us to demonstrate asynchronous code in JavaScript.

In real world websites/webapps the code will make a request to the server to either fetch or write to a database. This server request generally takes some time to complete its execution and/or return something back. While this server request is occurring you would want to move onto the next line of code and do something else (i.e. you do not want users to sit around and wait for something to happen which would make for a very poor user experience). This is how a loader spinner is added when fetching data using the asynchronous code pattern.