**Understanding how JavaScript Runtime Works**

JavaScript is a single-threaded programming language which means it has a single call stack and a single memory heap. This means that the language can only run only one thing at a time. The JavaScript runtime environment provides programs access to built-in libraries and objects so that the code work. It is comprised of the following elements:

* Call stack
* Memory heap
* The Callback queue
* The Event Loop
* Web APIs

**The Memory heap**, also called ‘The heap’ is a preserved computer memory (main storage) that a program process can use to store data or for allocation of objects and variables.

**The Call stack** is a data structure that keeps track of the current location of where we are in the program and runs in a last-in, first-out way. Each object in the stack is called a stack frame. The frame at the top of the call stack is the JavaScript engine runs.

As a JavaScript engine gets to a function, it is pushed onto the stack, when the functions return a value or gets sent to the Web APIs, it is popped off the stack and continues with the next stack frame.

**Web APIs** let developers manipulate documents, draw and manipulate graphics and fetch data from servers. Features like event listeners, timing functions and AJAX requests are contained in the Web APIs waiting for an action to be triggered. When a request finish receiving data, or a timer reaches its set time, or a click happens; this triggers a call back function to be sent to the Callback queue.

**The Callback** queue stores the callback functions from the Web APIs in the order in which they were sent. The queue is a data structure that runs first-in, first-out. Callback functions would have to sit in the queue until the Call stack is empty and they are then moved into the Call stack.

The job of the **Event loop** is to constantly monitor the state of the Call-stack and the Callback queue and put it onto the Call Stack, scheduling it for execution.

