**JavaScript Event Loop**

Diagram

Description automatically generated

1. **The Heap**

A block of memory where we store objects in **unordered manner**. JS variables and objects currently in use will be stored in the heap

1. **Call Stack** (single-thread)

The call stack manages the execution of the functions being called using a last-in-first-out structure. Once a script is run, the first function is placed into the call stack. If the first function calls a second function, then that is stacked on top of it. Let’s say that the second function calls console.log('hello world');. This would be placed on top of the second function. If there is no other functions being called, the console.log executes and, once finished, is popped off the stack. Then the second function executes and is also popped off. Finally, the first one behaves the same way leaving the call stack to be empty. This is the single-threaded language in action.

1. **Web APIs (and Node)**

**When a JavaScript runtime engine reaches an asynchronous function in the call stack, it will immediately return and pop out of the stack.** While the call stack moves on to the next function, the web APIs handle the asynchronous function. It is because of the Web APIs built into our browsers that JavaScript is able to perform asynchronous tasks. The callback function and any metadata associated with it is registered to an event table and then passed to the task queue. For example

setTimeout(function() {  
 console.log('Logged after a 1 second delay');  
}, 1000);

When it is popped off the stack and passed to the Web API, the Web API will set a timer based on the second argument and, once expired, passes the callback into the **task queue**.

1. **Task Queue**

The task queue operates on a first-in-first-out structure. The first function in must wait for the event loop’s permission to pass. The event loop is an algorithm constantly checking the call stack and will not let any of the callback functions into the stack until it is empty. Once call stack is empty, it will take the first callback out of the task queue and push it into the call stack allowing it to execute. Once the stack is empty again, the event loop will allow the next callback from the queue into the stack and repeat this process until the queue itself is empty.