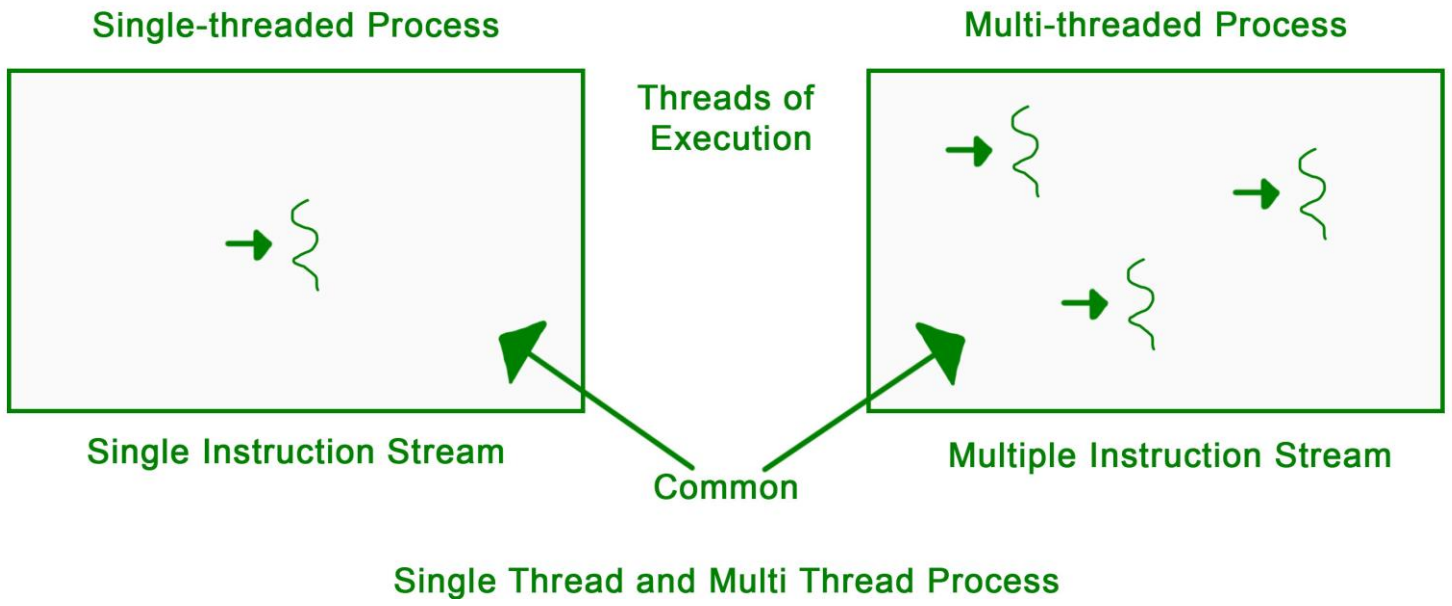
Thread

A thread is a sequence of instructions that execute one after another.

A single is basically just the code being executed top to bottom, as we would be reading the code that we write

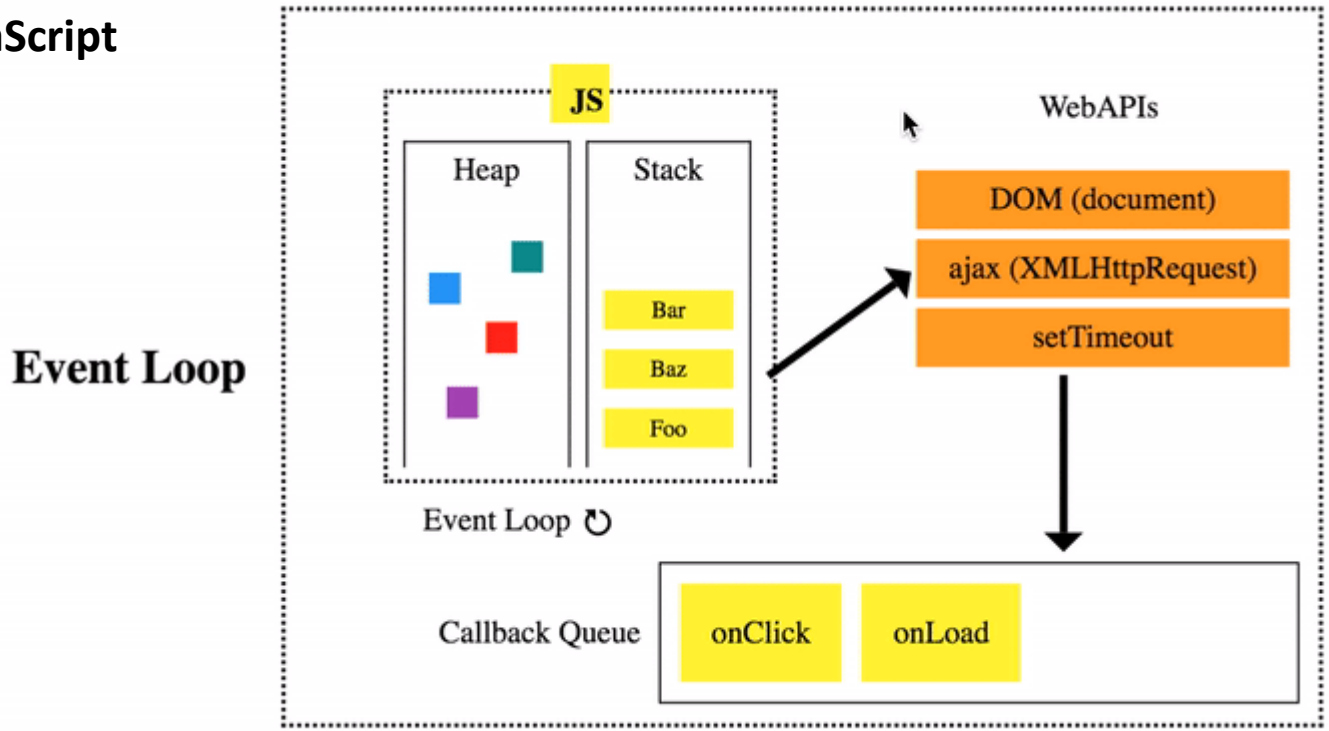


Single Thread and Multi Thread Process

So, if JavaScript is single threaded, why does it look like things can happen simultaneously?

Enter, the **event loop.** The event loop will place callback functions inside the callback queue into the stack for execution, when the stack is empty. We have this idea of WebAPIs, which are other threads that our web browser handles, that can place callback functions into the callback queue.

This concept is Asynchronous JavaScript



Fetch API

- Fetch API is one of the WebAPIs that allows the browser to send HTTP requests and receive HTTP responses
- Whenever our program interacts with the Fetch API to send a request, the program DOES NOT WAIT for the response to be received. Instead, the waiting for the response will happen over in the browser's other threads.
- Whenever the response is done, a callback function will then be placed with the HTTP response data into the callback queue
- IF our execution stack is empty, then the callback function with the HTTP response will then be executed in our own stack.

Whenever you send an HTTP request, you don't know when you're going to receive it. It could take 200ms, 1 second, 5 seconds, etc.

SO, the reason we need to have the Fetch API work with the event loop and provide this asynchronous "concurrency" (at the same time) is to provide the user with a fluid experience. You don't want your JavaScript code to be pausing for 10 seconds for that slow response to finally arrive.

We want other things to continue to happen while we are waiting for the HTTP response to arrive.