**Event Loop: (it has a counter named reff which increase by one whenever new event register minimum no of reff must be 1 and process.exit function call to stop the event loop)**

The [setTimeout()](https://www.javascripttutorial.net/javascript-bom/javascript-settimeout/), fetch requests, and [DOM](https://www.javascripttutorial.net/javascript-dom/) events are parts of the [Web APIs](https://www.javascripttutorial.net/web-apis/) of the **web browser**.

**Timers(setTime/intervals call-backs),pending [ I/O related call-backs[network operation or read/write file operations], Poll[check the poll and try to run the new call back function first but if not possible then recommend them as pending],setImmediate** call-backs **and close Callbacks.**

**Web API:** These Web APIs allow you to use the modern functions provided by the web browsers in your web applications.

When you call the [setTimeout()](https://www.javascripttutorial.net/javascript-bom/javascript-settimeout/) function, make a [fetch request](https://www.javascripttutorial.net/javascript-fetch-api/), or click a button, the web browser can do these activities concurrently and asynchronously.

[**https://www.javascripttutorial.net/javascript-event-loop/**](https://www.javascripttutorial.net/javascript-event-loop/)

**Call Stack:**

The **call stack** is responsible for keeping track of all the operations in line to be executed. Whenever a function is finished, it is popped from the stack.

**Event Queue / Task Queue:**

The **event queue** is responsible for sending new functions to the stack for processing. It follows the queue data structure to maintain the correct sequence in which all operations should be sent for execution.

**The language itself is single-threaded, but the browser APIs act as separate threads.**