Advanced Javascript

Day 6 - Asynchronous Programming

Program Execution

But first...

Let’s understand multitasking.

A computer can multitask, but can a computer program too?

Synchronous Execution

One line at a time.

Javascript executes code line by line. Almost every programming language executes code line by line.

If all languages are synchronous, isn’t that a bad thing?

There is a way….

Asynchronous Execution

Multiple resource. Multiple execution at a time.

Js can act asynchronous with the help of browser.

Enter Browser API

Whenever Js Engine encounters a piece of code that will take a long time to execute, it hands over it’s execution to browser.

Browser now takes care of that task and when its finished, it pings the Js engine that the task is done.

That’s how Js engine and Browser work together to make Javascript capable of asynchronous programming.

setTimeout

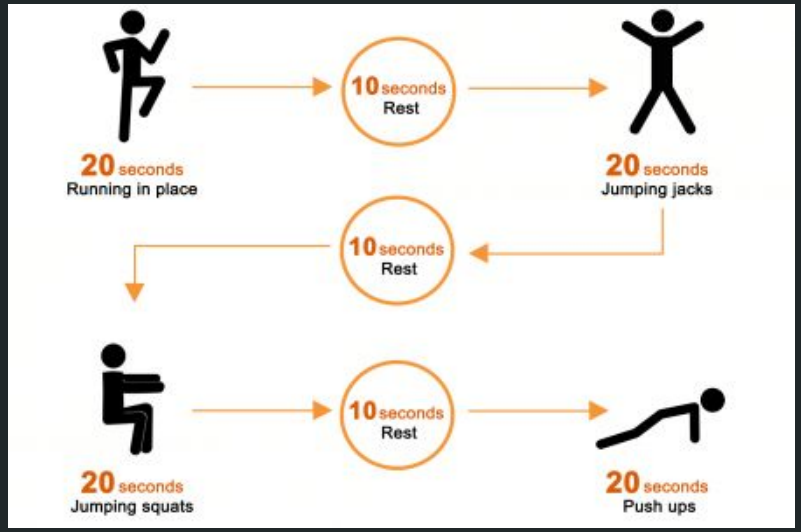
What is setTimeout()?

setTimeout sets a timer which executes a function or specified piece of code once the timer expires.

Whenever setTimeout() is encountered in a Js code, Js engine offloads that code to browser.

Now, it’s browser’s responsibility to keep track of counter and remind Js engine that this code should be executed next.

setInterval()



What is setInterval()?

setInterval() method repeatedly calls a function or executes a code snippet, with a fixed time delay between each call.

But how does Js engine keeps track of what to execute next?

Enter Stack and Queue

Let’s revisit stack and see where the new concept called ‘queue’ fits in.

Stack

When you call a function in JavaScript all the instructions within that function get loaded into a Stack.

Javascript then executes the instructions in each function by popping them from the stack.

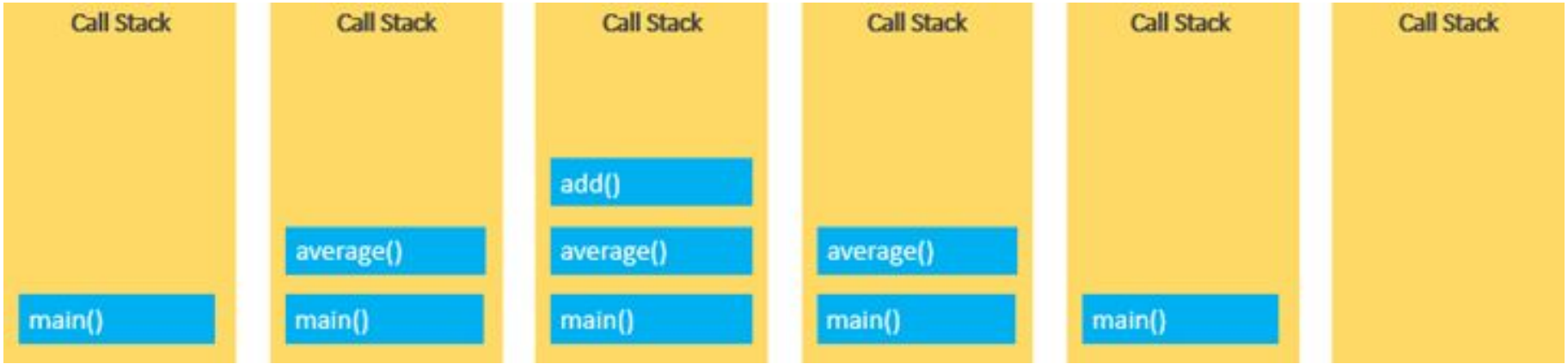
```
function main()
{
average()

}

function average()
{
add()
}

function add()
{
}

main()
```


Call Stack
main()
Call Stack
average()
main()
Call Stack
add()
average()
main()
Call Stack
average()
main()
Call Stack
main()
Call Stack

Message Queue

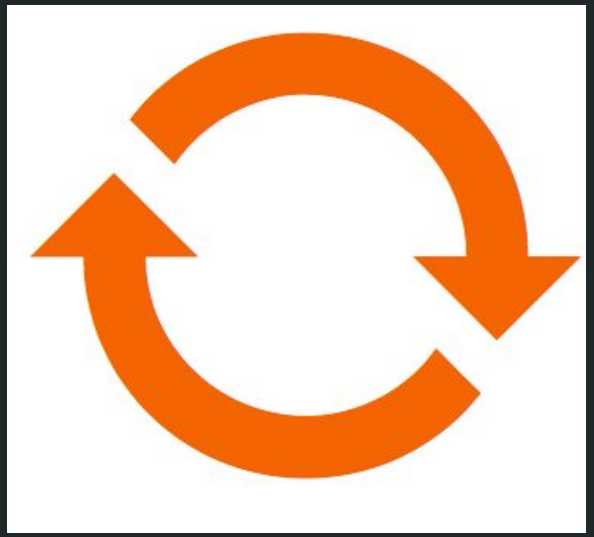
It’s a one more data structure that Js to add next code to execute.

Such code are called tasks, since they need more time to execute.

Tasks are added to the queue whenever Js finds a function which will take some time to execute.

Whenever Js finds Functions like setTimeout and setInterval which requires a particular time to execute, it also add them to the message queue.

Event Loop



Waitress

Chef

Service

Chef's table

Waitress

Chef

Service

Chef's table

Made by Papi Diagne

Waitress

Chef

Service

Chef's table

Made by Papi Diagne

Waitress

Chef

Service

Chef's table

Made by Papi Diagne

Waitress

Chef

Service

SaladBowl

Chef's table

Made by Papi Diagne

Waitress

Chef

Service

Chef's table

Made by Papi Diagne

Waitress

Chef

Service

SaladBowl

Chef's table

Made by Papi Diagne

Waitress

Chef

Service

Chef's table

Made by Papi Diagne

Waitress

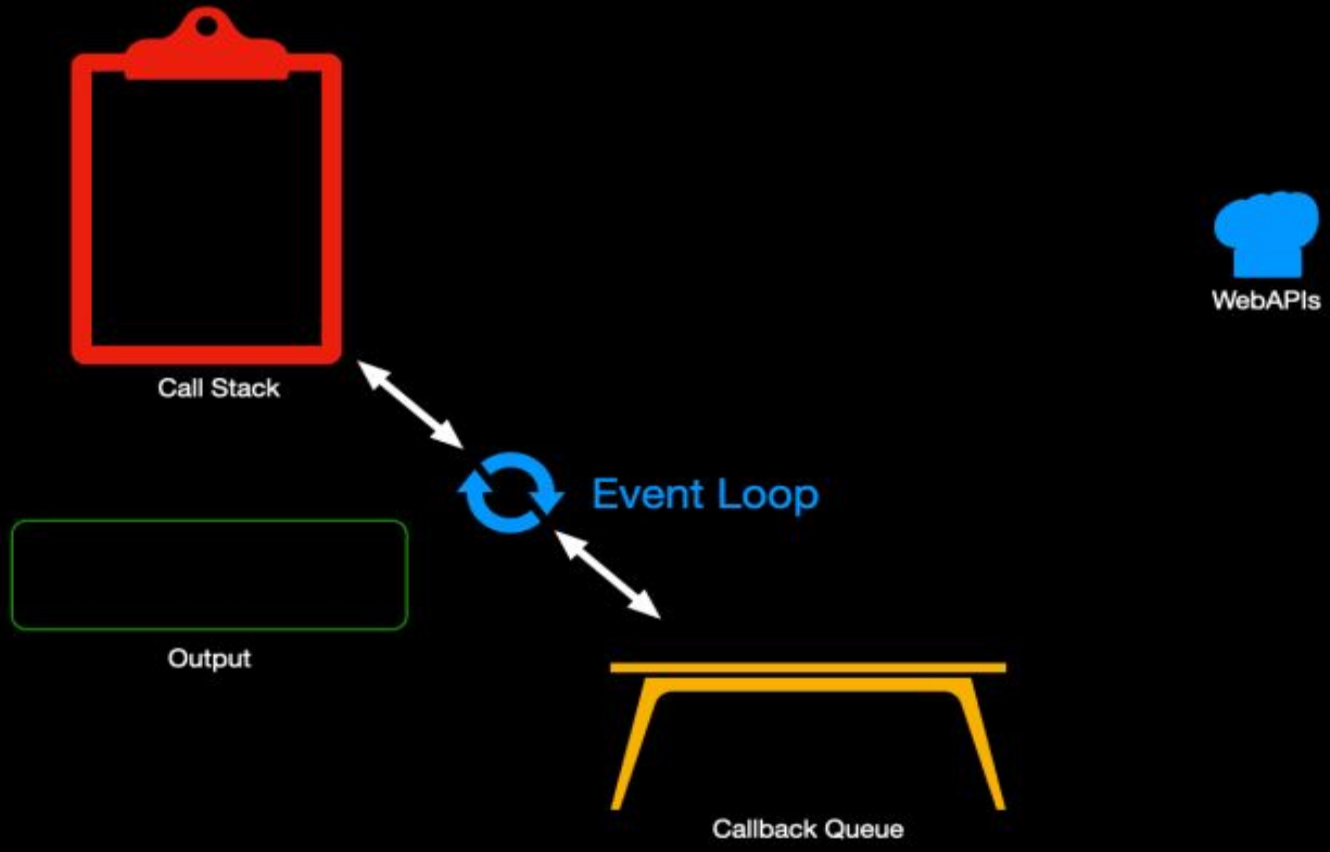
Chef

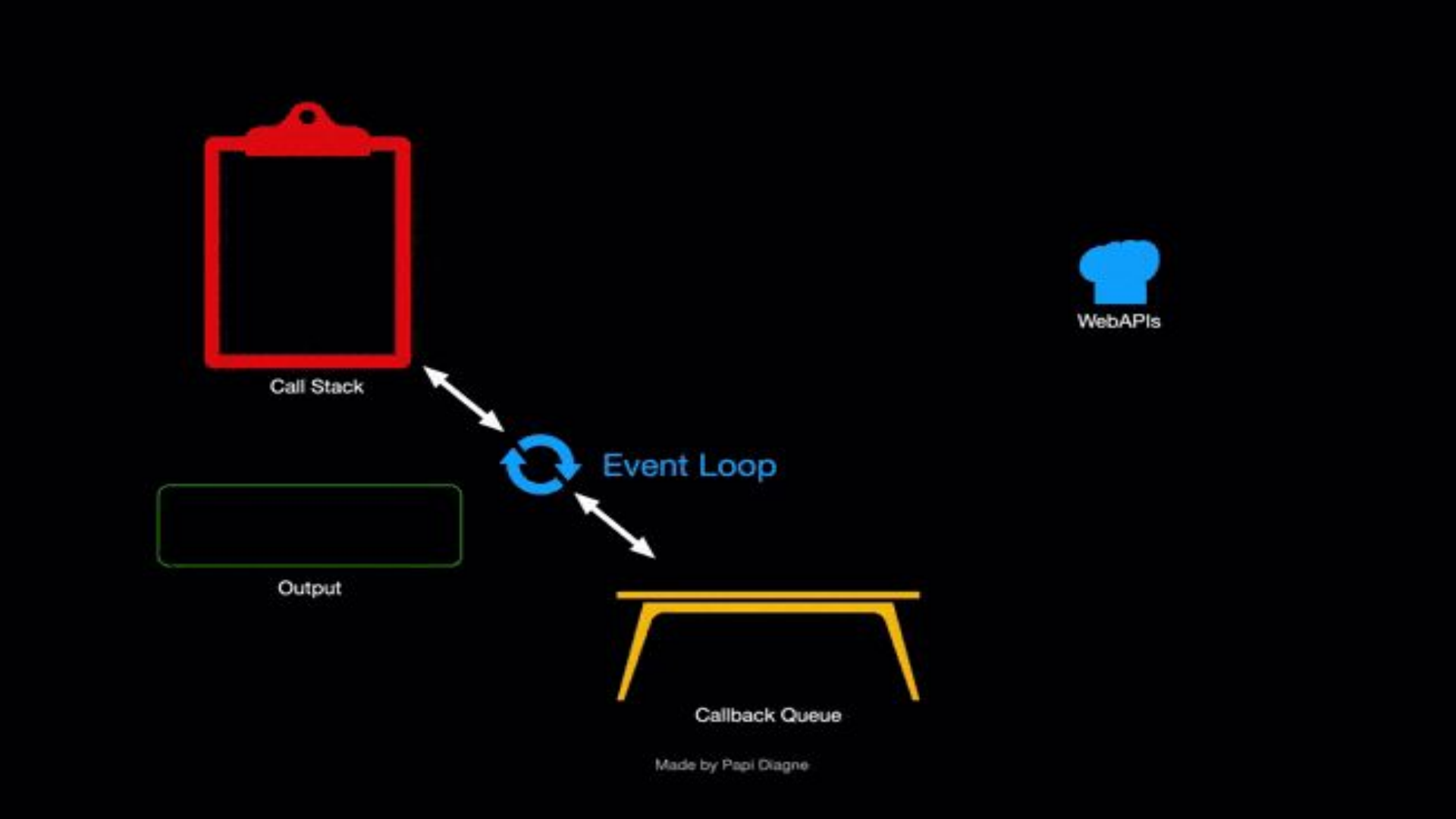
Service

Chef's table

Made by Papi Diagne

Let’s add our fancy words now and make sense of this model.


Call Stack
Event Loop
WebAPIs
Output
Callback Queue
Made by Papi Diagne


Call Stack
WebAPIs
Event Loop
Output
Callback Queue
Made by Papi Diagne

https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif

http://latentflip.com/loupe/

JavaScript runs a loop that looks for new messages/tasks on the message queue and pushes them onto the call stack to be executed.

However, the event loop gives priority to the code currently on the call stack .

It pushes a new message from the Queue onto the stack after all the code in the stack have been executed and the call stack is empty.

Queue holds all the code that will require a longer time to execute.

Especially, network requests.

But why asynchronous programing is important?

Most programs you write often tend to be synchronous. This means that actions in code happen one at a time.

For example HTTP requests could take any amount of time to receive a response from the server. There is even a chance you won't even get a response!

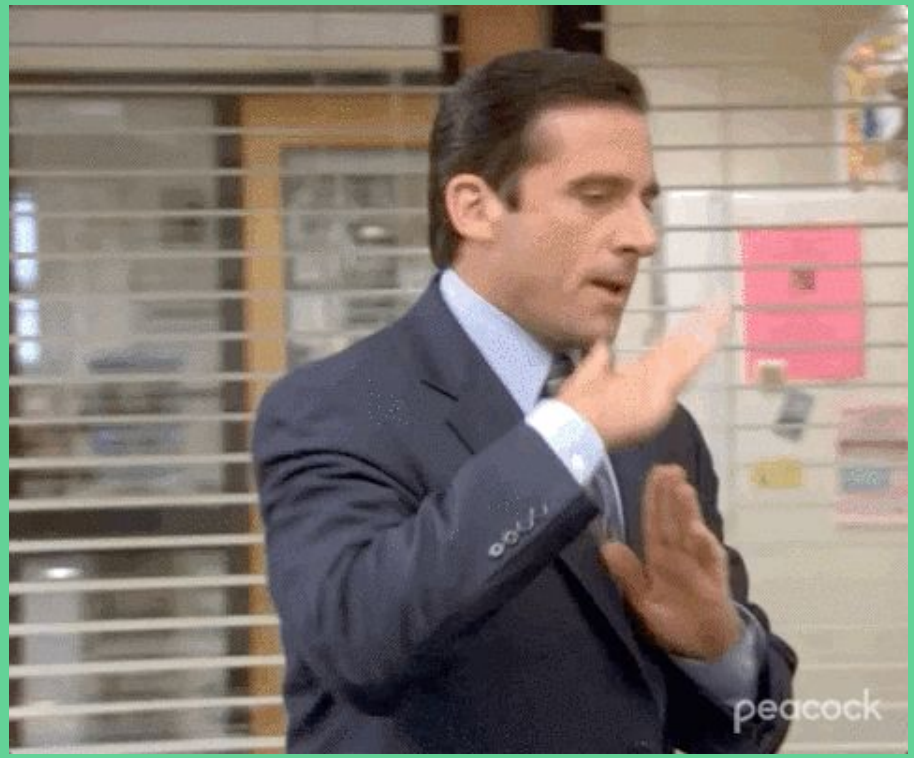
This is known as blocking and can slow down app performance.

Let’s Recap

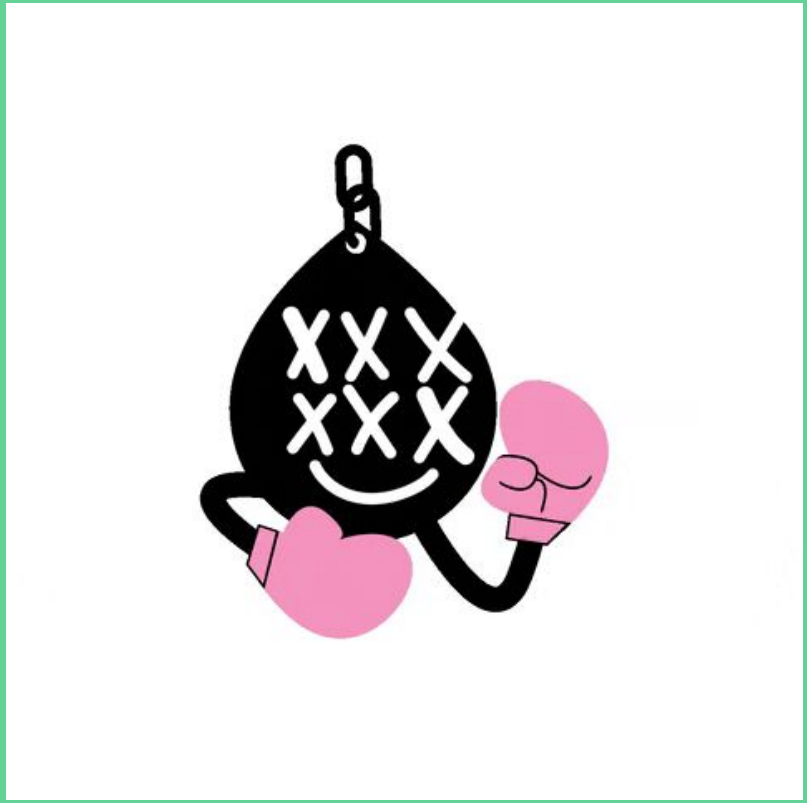
Synchronous Execution -

Asynchronous Execution

setTimeout

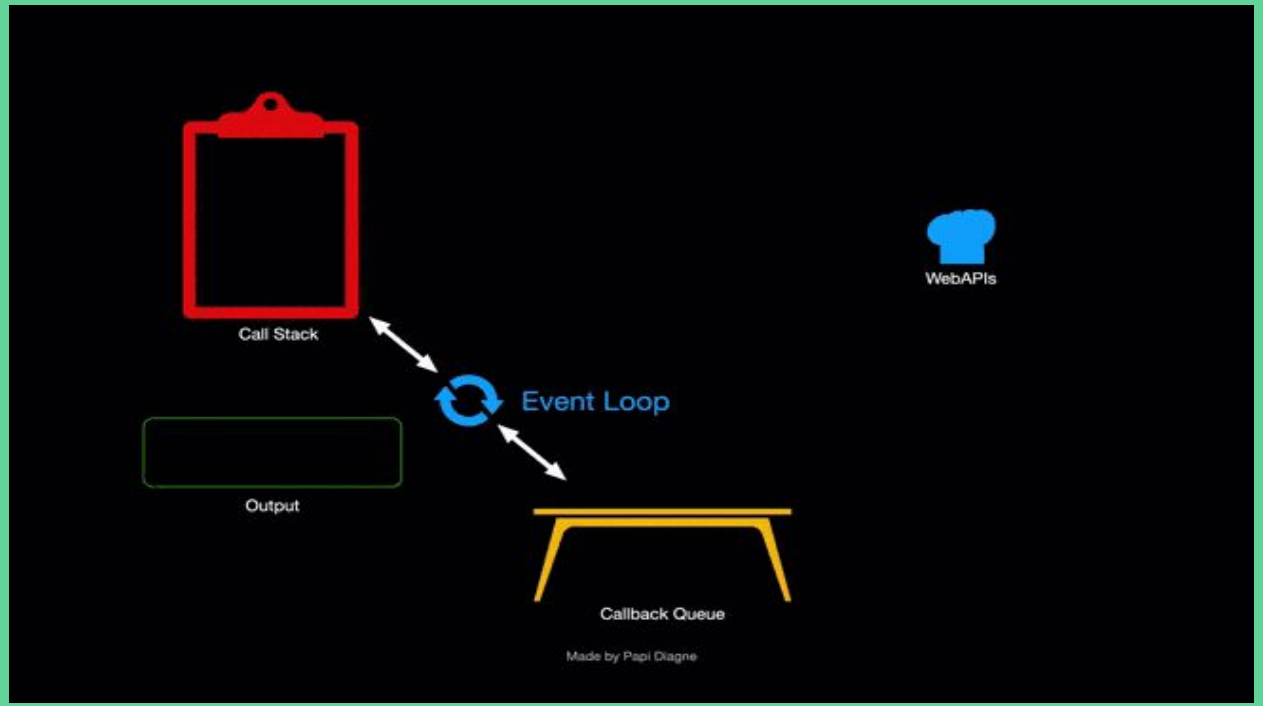


setInterval



Message/Task Queue

Event Loop


Call Stack
WebAPIs
Event Loop
Output
Callback Queue
Made by Papi Diagne