View All Pages

Code Execution in Js

Synchronous Execution

Explore Synchronous nature of Js

- Run a for loop upto a million iteration

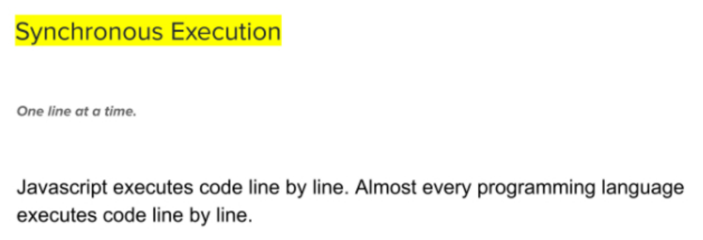
```
console.log('before loop')
for(let i=0;i<100000000000;i++)
{

}
console.log('after loop')
```

Understand Synchronous behaviour of Js

- There will be some code that might take time to complete, it can block other code if we don't have services of async programming. For e.g making a network request

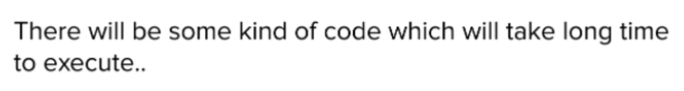
- Js cannot do multiple things at same time. Js needs a friend, that friend is browser.



Synchronous Execution

One line at a time.

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



There will be some kind of code which will take long time to execute..

Js can act asynchronous with the help of a friend.

Asynchronous Execution

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.

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

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.

Message Queue/Callback 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.

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.

◂ Previous

Next ▸

setTimeout and setInterval

Used to simulate payment form (cvv, otp etc) using sync prog functions studied so far. Waiting for payment conformation

Understand the tools setTimeout and setInterval as per WE Session

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

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

Account
Dashboard
Courses
Calendar
Inbox
Help

Home
Announcements
Assignments
Discussions
Grades
Pages
Syllabus
Quizzes
Modules

View All Pages

Event Loop

Call Stack

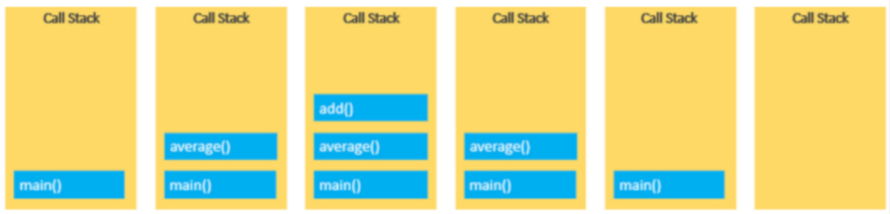
Understand Call stack using Loupe

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 add()
{
}
function average()
{
add()
}

main()
```



Example 1

Example 2

Experience with websites

- What kind of features on any website takes time to execute?

- For e.g Completing a payment, Booking a ticket, loading images but text gets loaded.

Understand Web API

- In the case of setTimeout, setInterval, network request, it is sent to Web API and sent back to call stack. this mechanism is called Event Loop

- Understand, WEB API (Browser) cannot execute the code, it can just keep track of time and understand network response. Ultimately, Js is the only one who can execute the code.

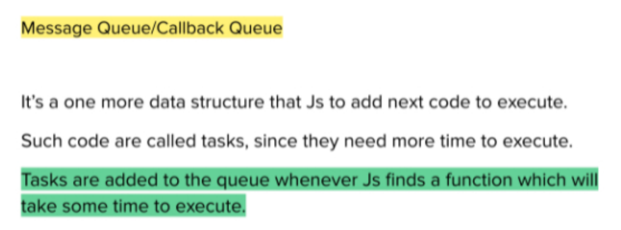
Whenever setTimeout() is encountered in a Js code, Js engine offloads that code to browser.

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

Callback Queue/Message Queue/Task Queue

Understand Callback Queue

- Whenever timer expires for setTimeout/setInterval or when you get a response from api call WEB API, it add them to the callback/task/message queue.



Message Queue/Callback Queue

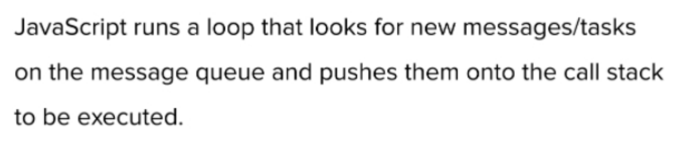
It's a one more data structure that Js to add next code to execute.

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

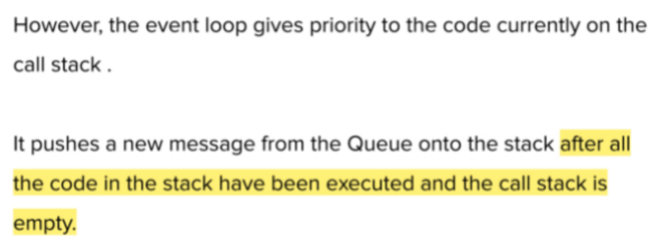
Tasks are added to the queue whenever Js finds a function which will take some time to execute.

Event Loop - Connecting the dots

Understand Event Loop



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.

Event loop visualised

◂ Previous

Next ▸