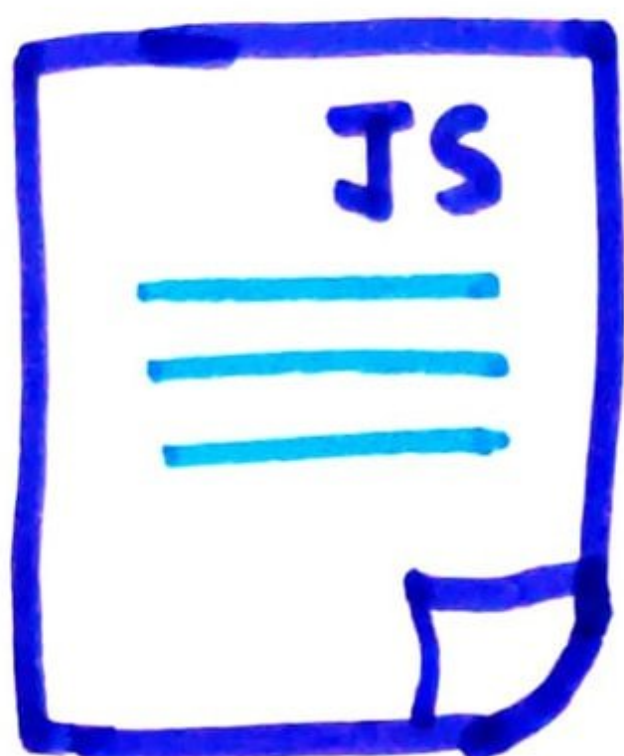


JavaScript Engine



Hey, I'm JavaScript
Can you help me run



Did someone
say anything?
I don't understand!

Okay... So the browser doesn't
understand JavaScript.

What it understands is bits
(1's and 0's)

Who can help us here?

Yes!! The JavaScript Engine



codeWithSimran



codeWithSimran_

Sold to
himanshurik57@gmail.com



There are a lot of JavaScript Engines out there written by really smart people!

For example :- V8 engine is written in C++ (yes they're programmed too and can be in a different language)

Okay, so what's inside this JavaScript Engine?

JE

Memory heap

This is where all the memory gets allocated
e.g. `var a = 5;`
memory allocated to variable `a`

Call Stack

This is where your program executes. It keeps track of where we are in the code



So ever heard of a memory leak?

A memory heap has limited space. When you have too much of unused memory that you don't free up the space gets filled.

No wonder, global variables are bad (They remain throughout the execution of the code)

You must've heard of stack overflow!!

Well that's when your call stack overflows as it also has limited space.

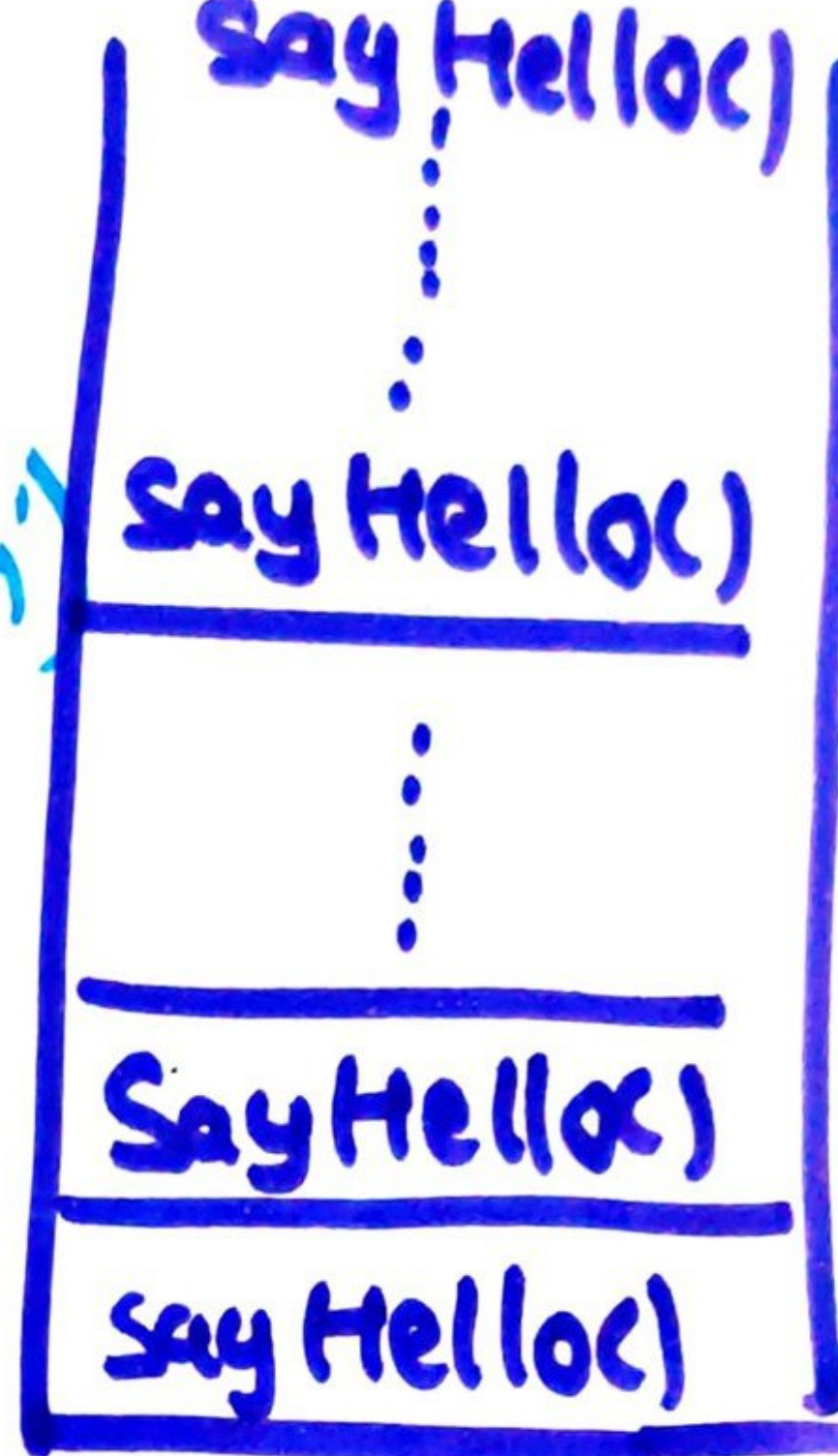



```
function SayHello()
```

```
{ console.log('Simran')  
  SayHello();
```

```
}
```

Well that went
into an infinite
recursion and we
have stack overflow



JavaScript is a single threaded
language?

Well that means it has only
ONE CALL STACK and therefore

it can only execute one task
at a time

Okay But Why single threaded?
It's quite easy and no complications

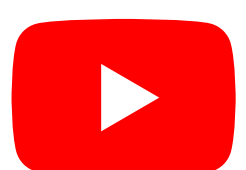


Okay... Wait! I've heard of asynchronous programming. If JavaScript can do that, how is it single threaded? Let's take an example!

```
setTimeout ( ) => {
```

```
  console.log("setTime out is asyn"  
  }, 2000) → wait for 2 seconds
```

setTimeout is given to us by Web APIs (It gives us various APIs) It's technically not a part of JavaScript.




```
console.log('1')
```

```
setTimeout(() => {
```

```
  console.log('2');
```

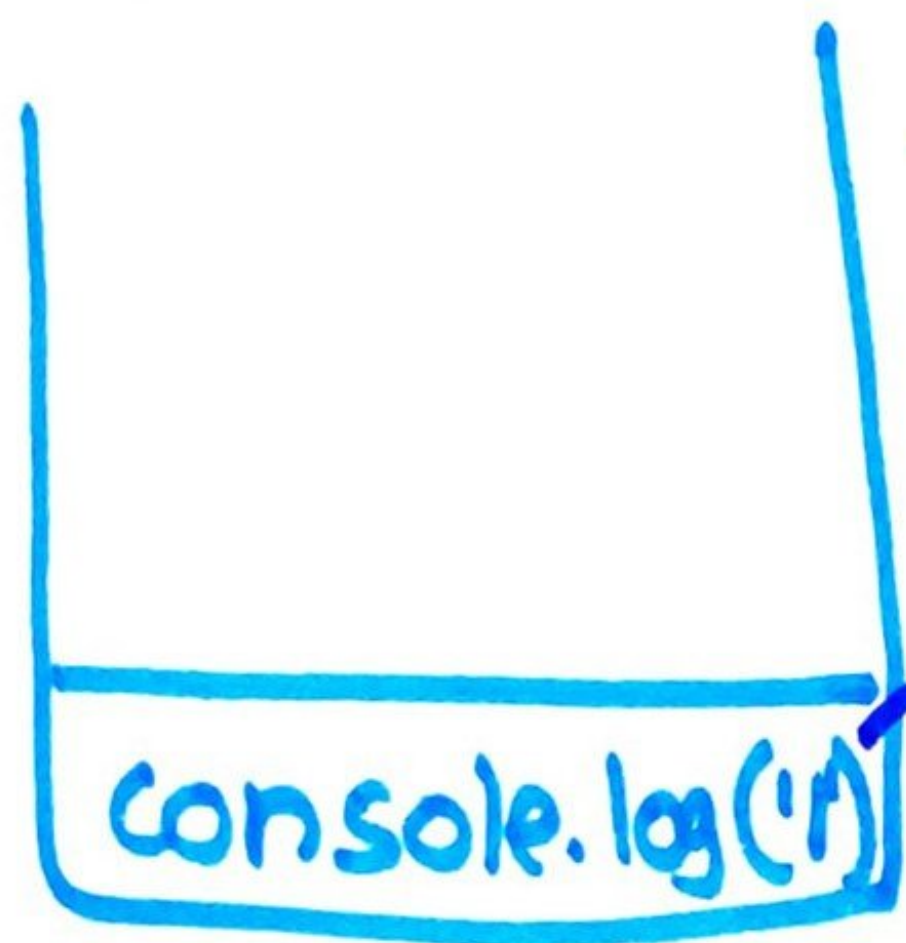
```
});  
console.log('3');
```

Output: 1
3
2

[since setTimeout
waits for 2 secs
it's printed in the
end]

Behind the scenes

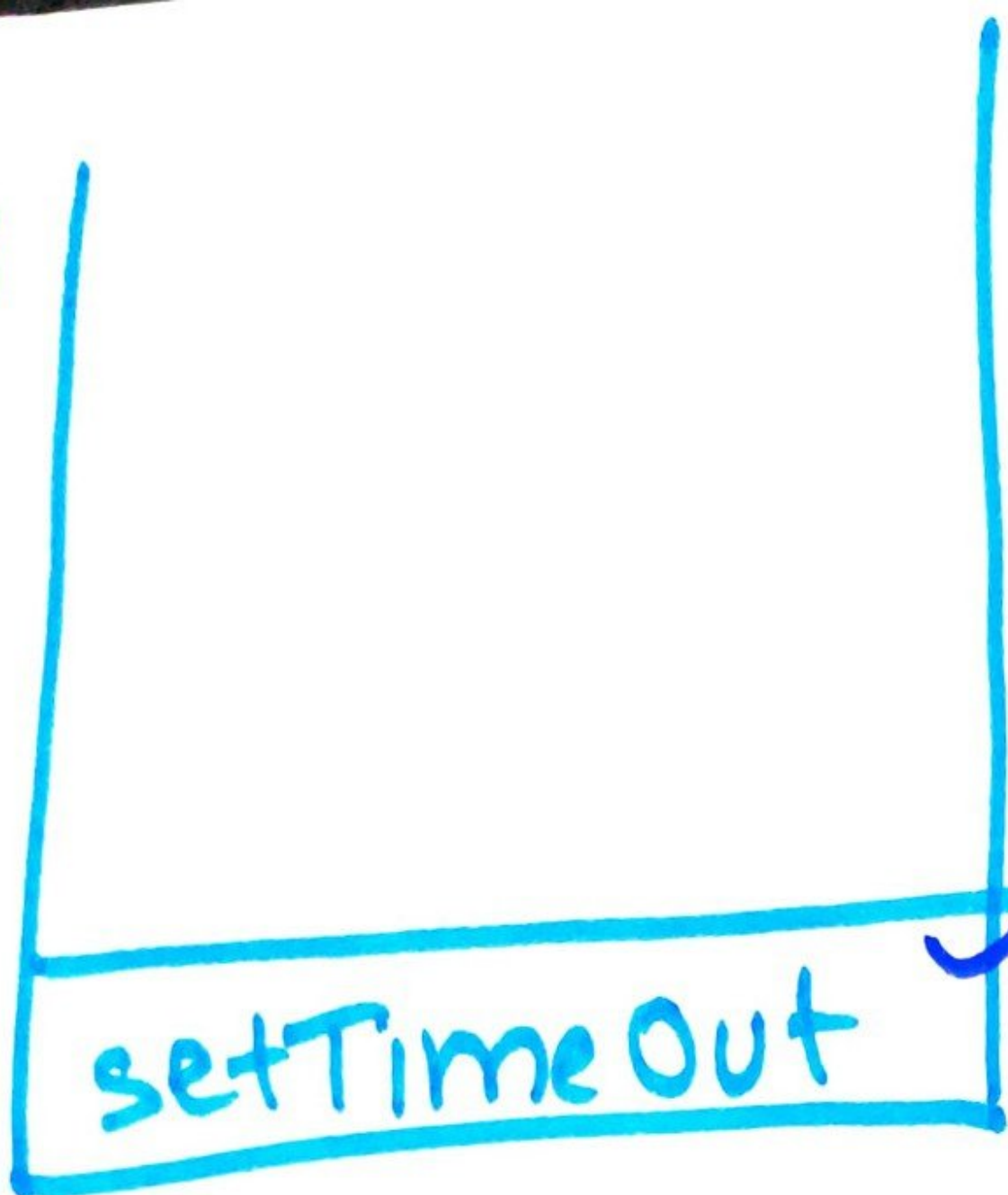
① → Call Stack



→ gets executed
and gets popped
So ~~the~~ stack is now
empty and output
is 1

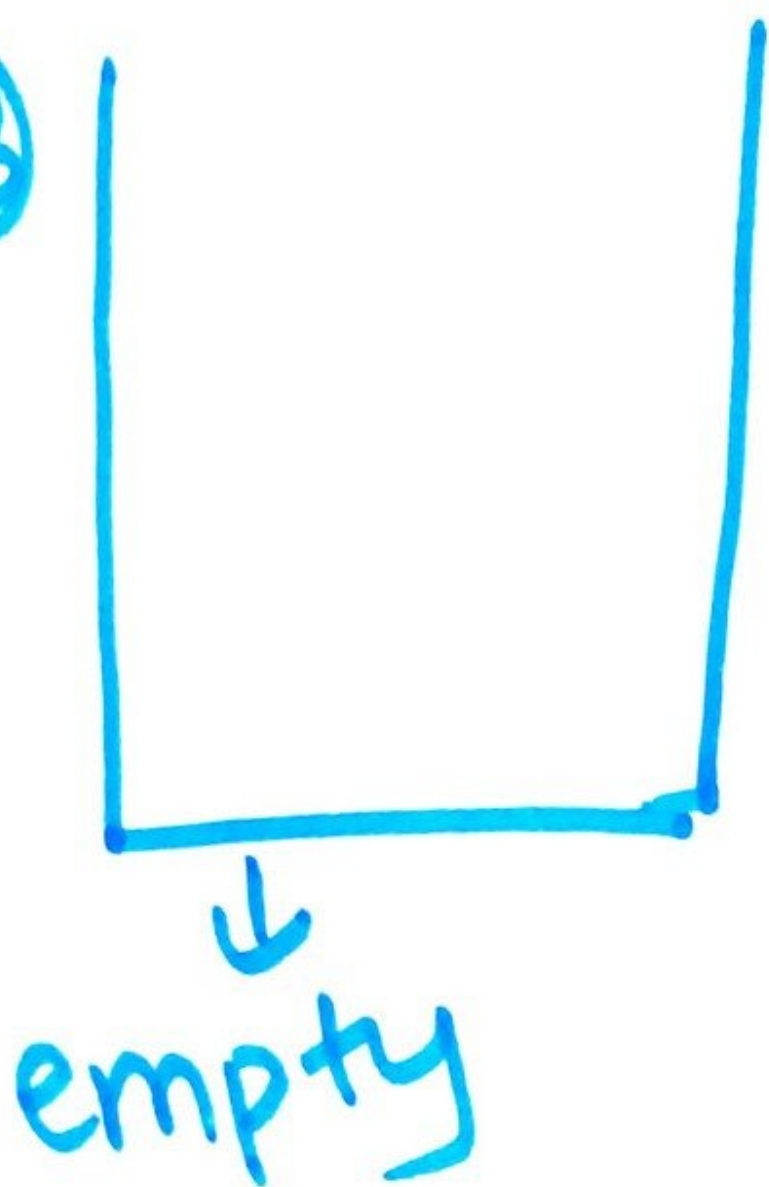


②



OH! this is
given by the
Web API Let me
Send it to web API

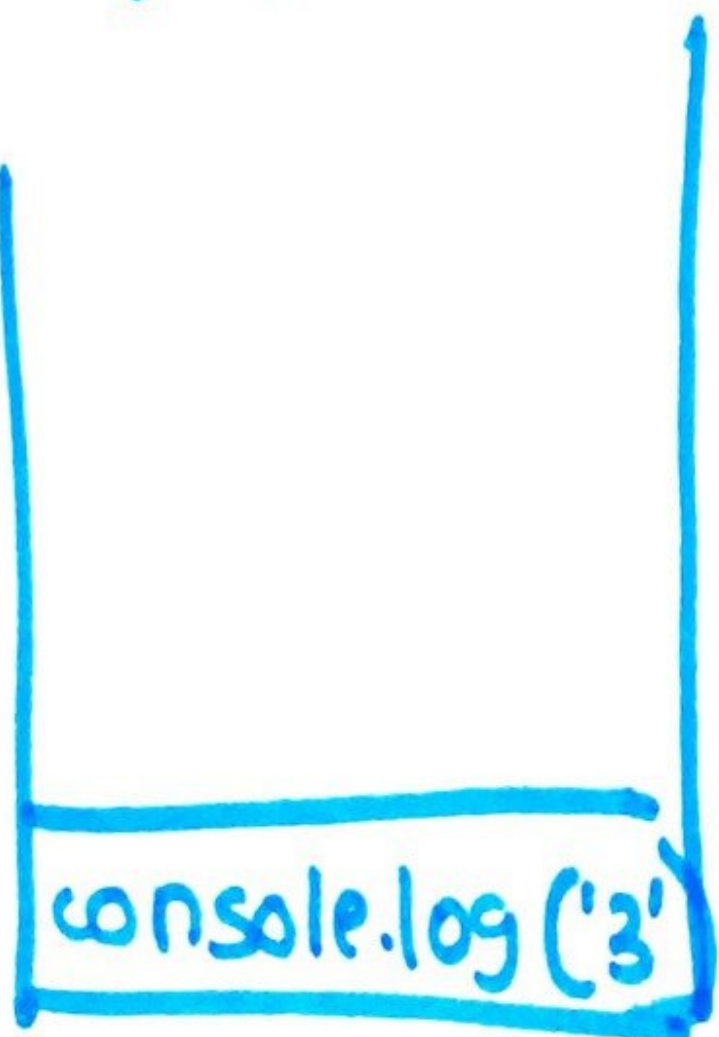
③



Web API

I've setTimeout
with me and I
should execute it
after 2 seconds

④



> Output will
print 3

So far

1
3

Web API is still waiting



⑤ After 2 seconds are over
WEB API ← On its console.log() that should be * executed.
This is basically a callback that is executed after 2 secs.

WEB API will send this to callback Queue saying there's a callback please proceed.



callback queue

This queue basically keeps track of all callbacks that need to be executed.



Now, there's something called as event loops which keeps checking if stack is empty

Well now it's empty so the event loop will take a callback from callback queue and put it in the stack

`console.log(2)`

> prints 2
So finally we have
1
3
2

Recap of setTimeout

- ① Pushed to stack → ② Passed to WEB API
- ④ Pushes callback to callback queue ← ③ waits for 2 seconds
- ⑤ Event loop check if stack empty and pushes to stack

