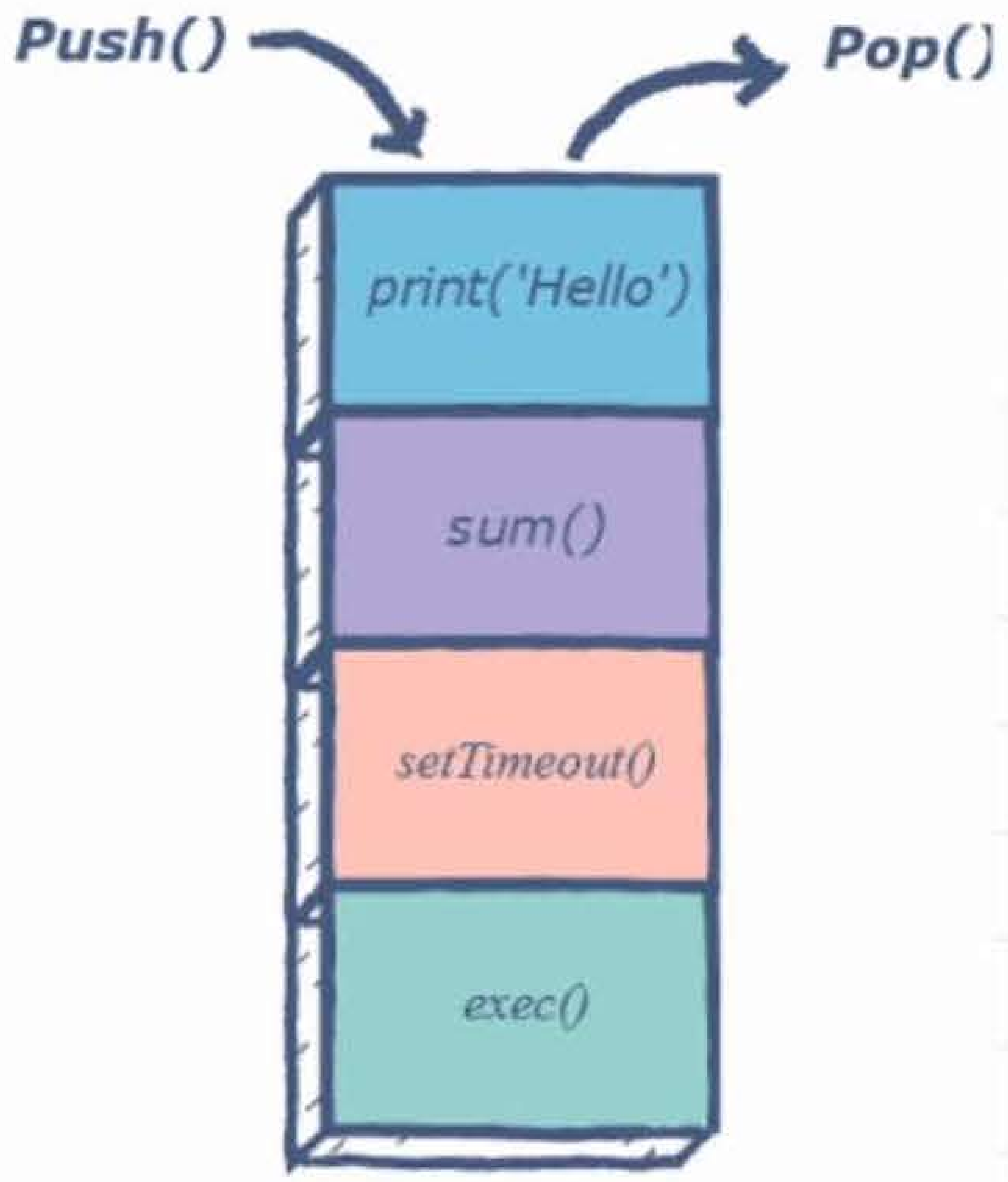
Press Esc to exit full screen

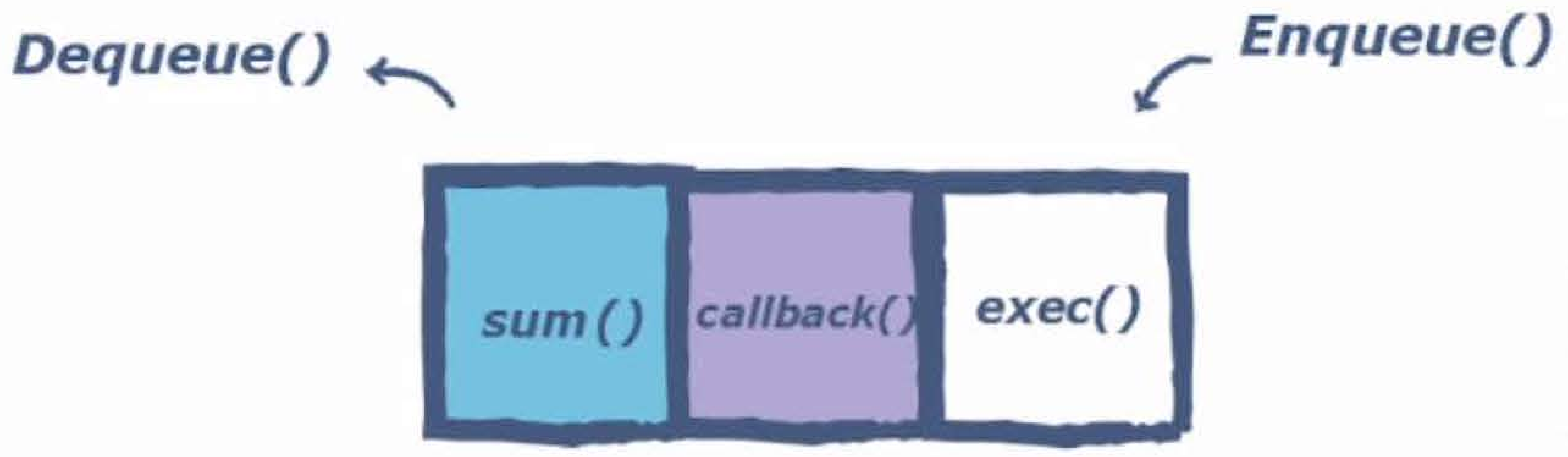
Call Stack

- Keeps track of all the operations in line to be executed.
- Whenever a function is finished, it is popped from the stack.

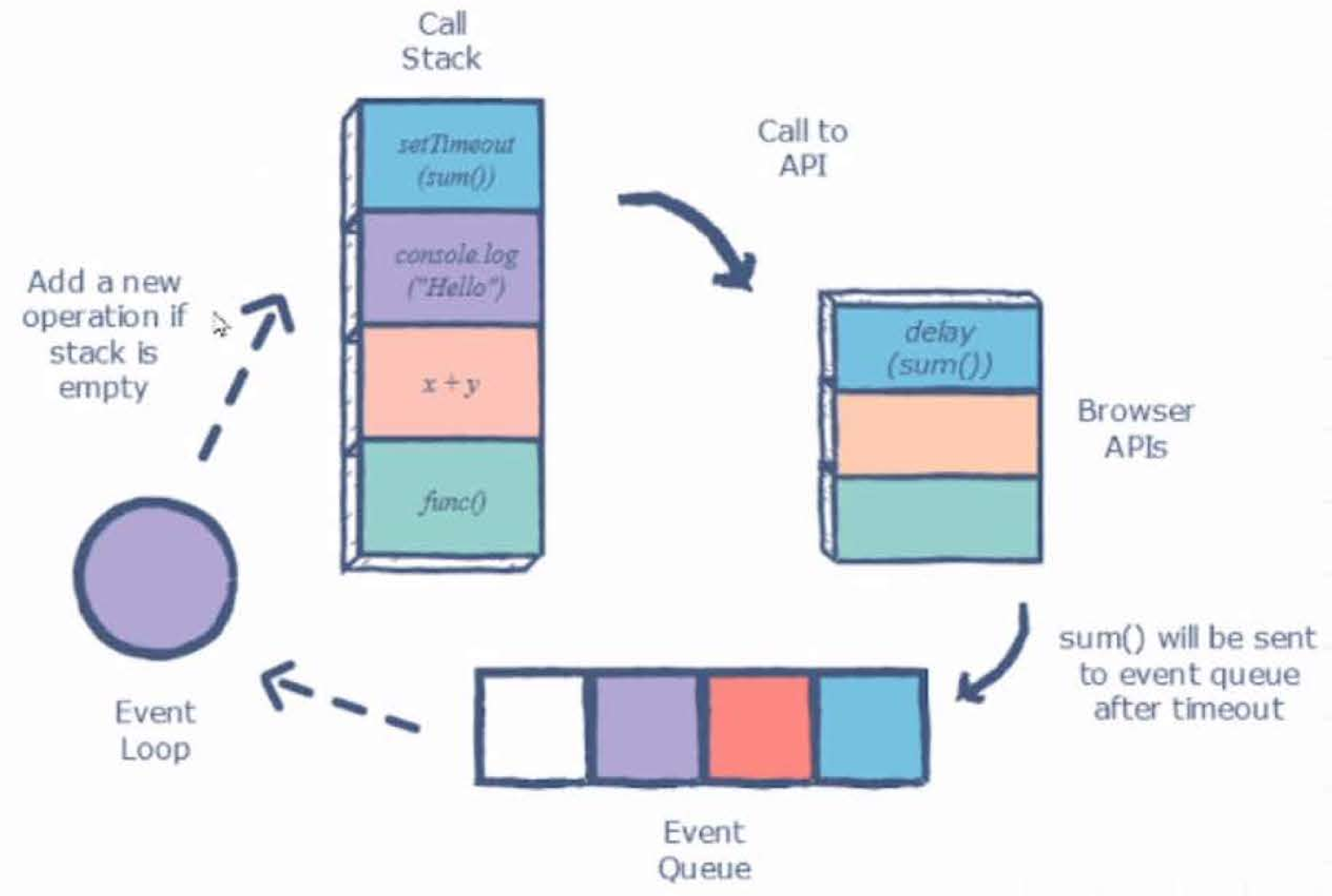


Event Queue

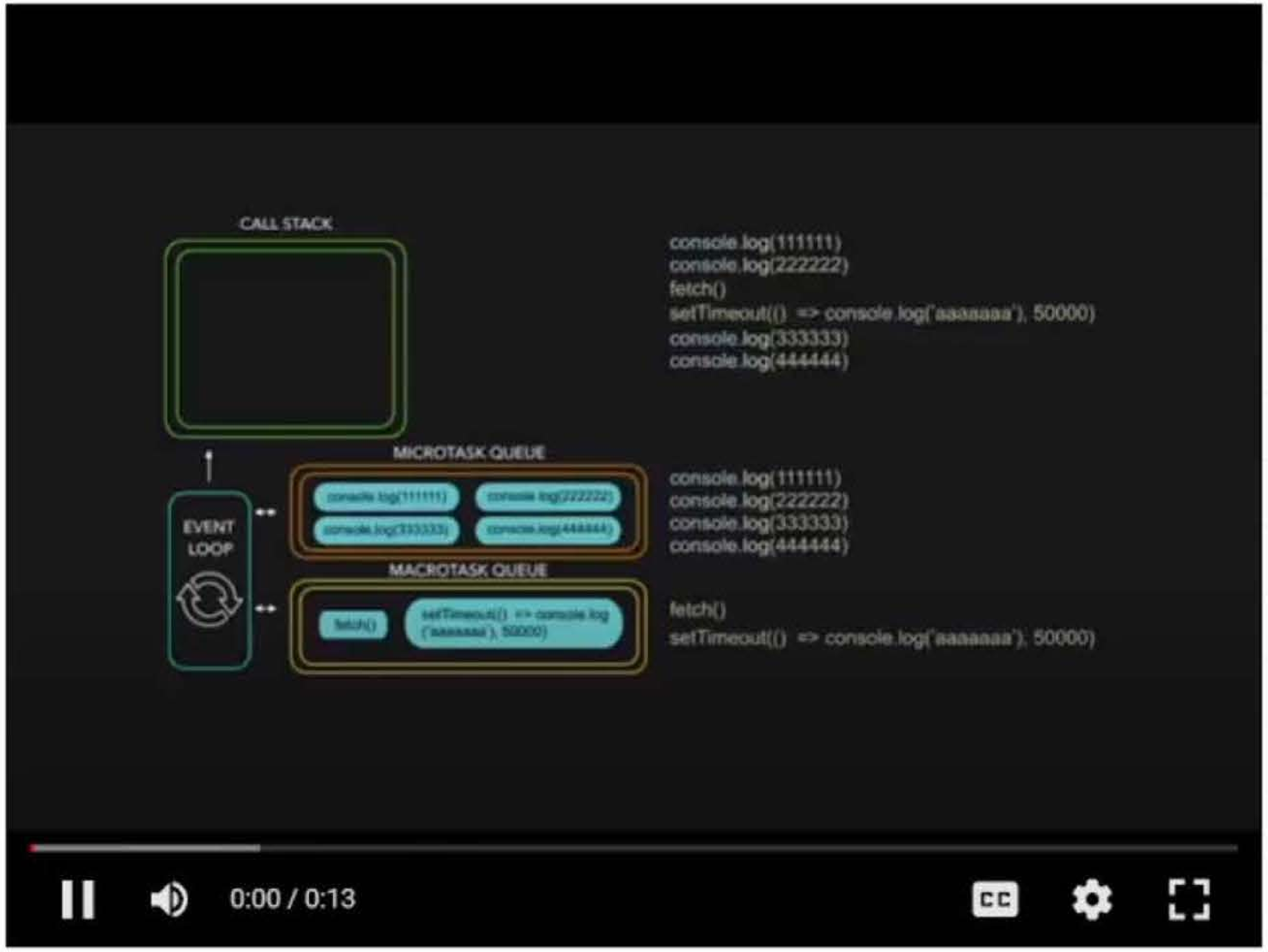
- Sends new functions to the stack for processing.
- Follows the queue data structure.
- Maintains the correct sequence in which all operations should be sent for execution.



Event Loop Execution



Event Loop Execution

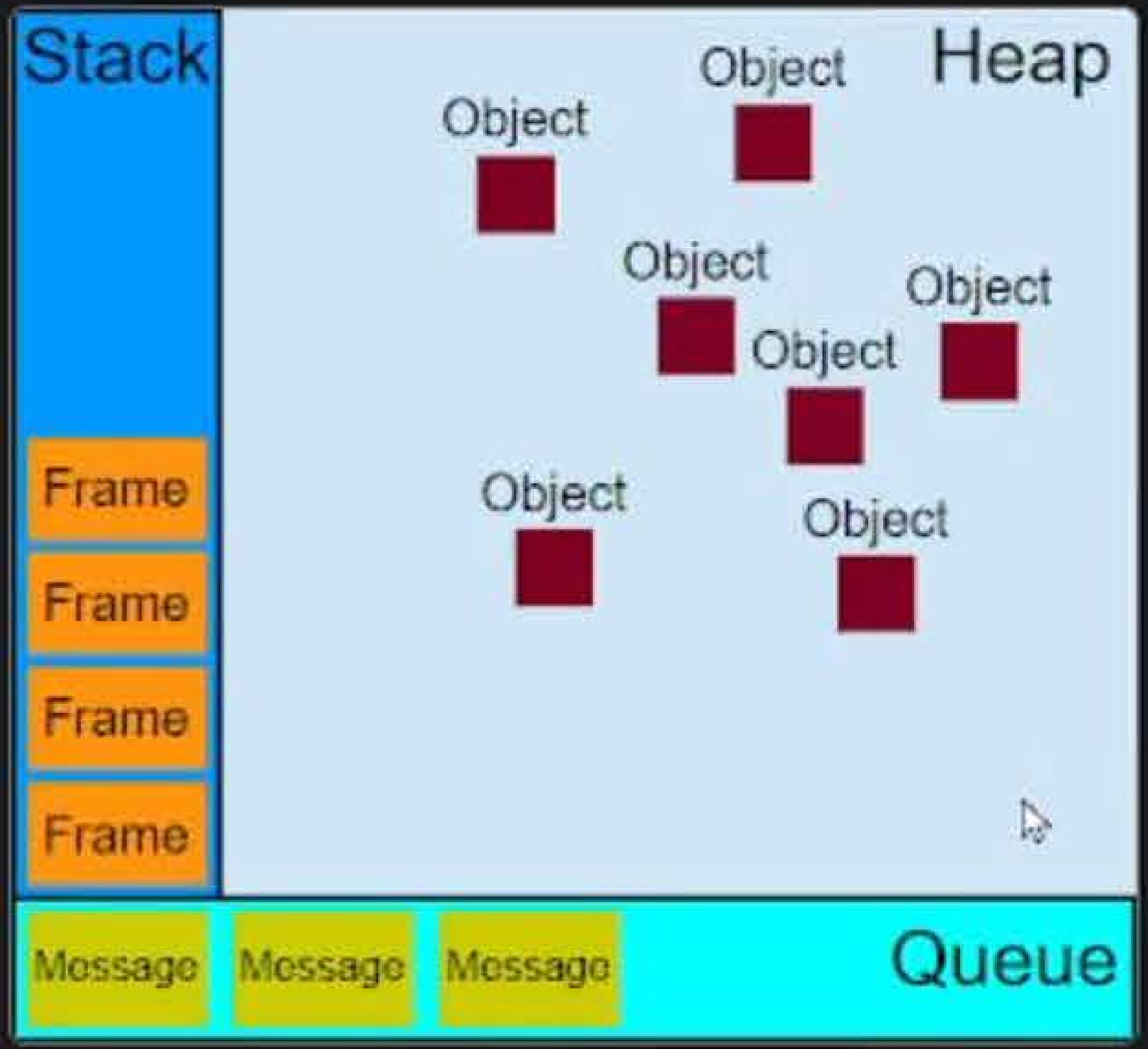


```
console.log(111111)
console.log(222222)
fetch()
setTimeout(() => console.log('aaaaaaa'), 50000)
console.log(333333)
console.log(444444)
```

```
console.log(111111)
console.log(222222)
console.log(333333)
console.log(444444)
```

```
fetch()
setTimeout(() => console.log('aaaaaaa'), 50000)
```

Visual representation



Stack

Function calls form a stack of *frames*.

```
function foo(b) {
  const a = 10;
  return a + b + 11;
```

What the heck is the event loop anyway? | Philip Roberts | JSConf EU

2.8M views 8 years ago
JavaScript programmers like to use words like, "event-loop", "non-blocking", "callback", "asynchronous", "single-threaded" and "concurrency".

We say things like "don't block the event loop", "make sure your code runs at 60 frames-per-second", "well of course, it won't work, that function is an asynchronous callback!" Show more

1,605 Comments Sort by

Add a comment...