Javascript
Engine

Content

Engine - Runtime	Call Stack	Event Loop
Asynchronus	Memory	Debugging

Javascript engine

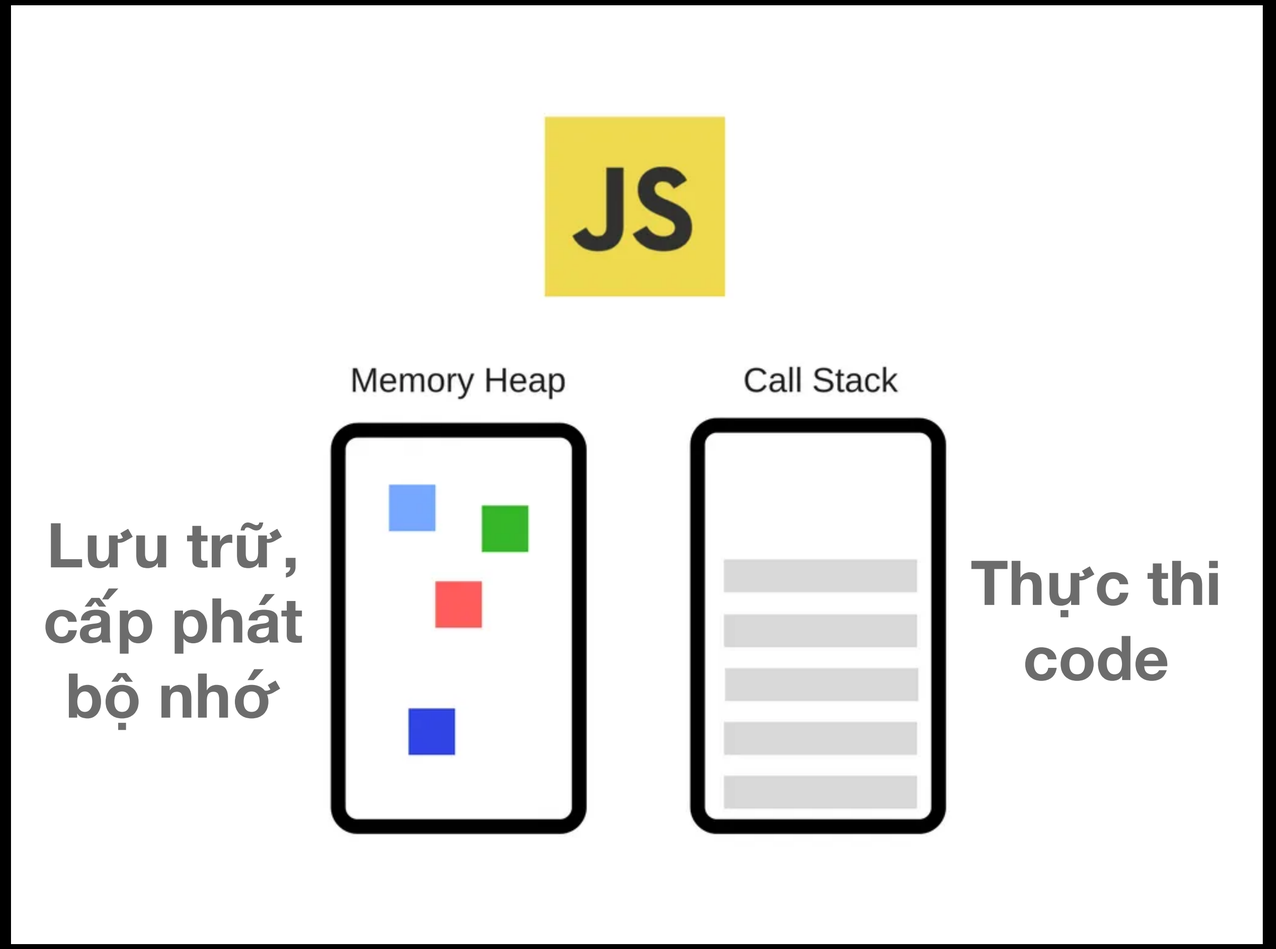
- V8 - Google (Chrome - Nodejs)
- Spider Monkey (FireFox)
- JavascriptCore (Webkit - Safari)
- Charka (IE - Edge)

=>

```
const value = obj["key"];
// safari
// ReferenceError: Can't find variable: obj

const value = obj["key"];
// Chrome
// Uncaught ReferenceError: obj is not defined
```

Engine Components



Runtime

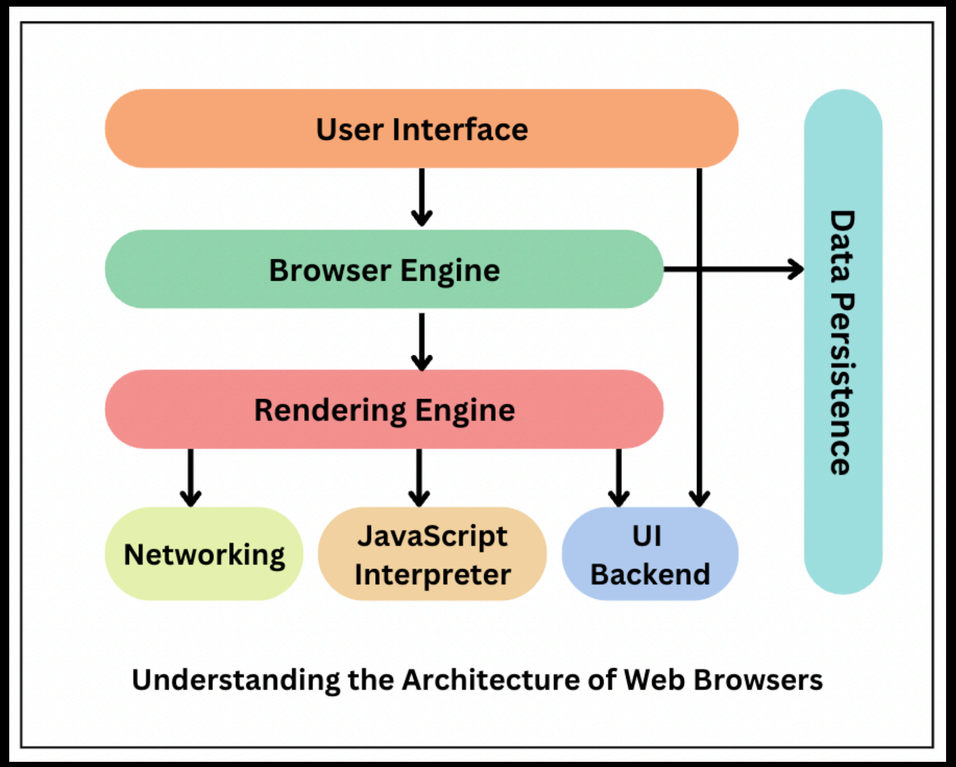
Do you think they are javascript?

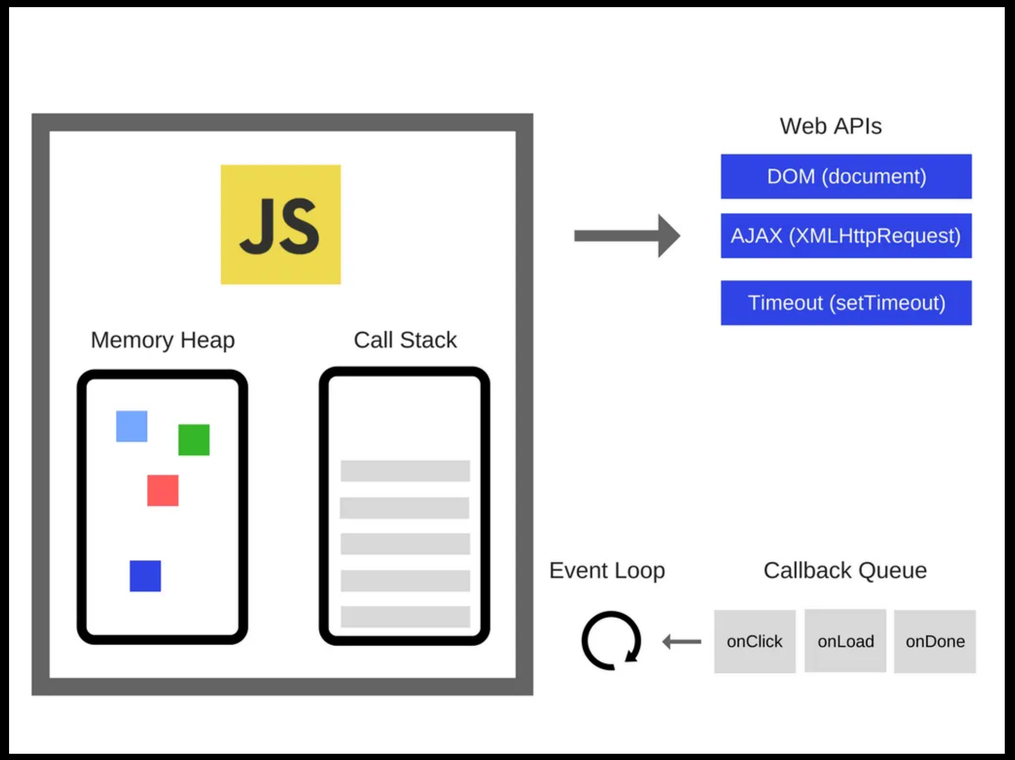
- setTimeout
- setInterval
- console.log, console.error
- document.getElementById()

No, they are not. They are web APIs.

Runtime

Javascript engine not working alone



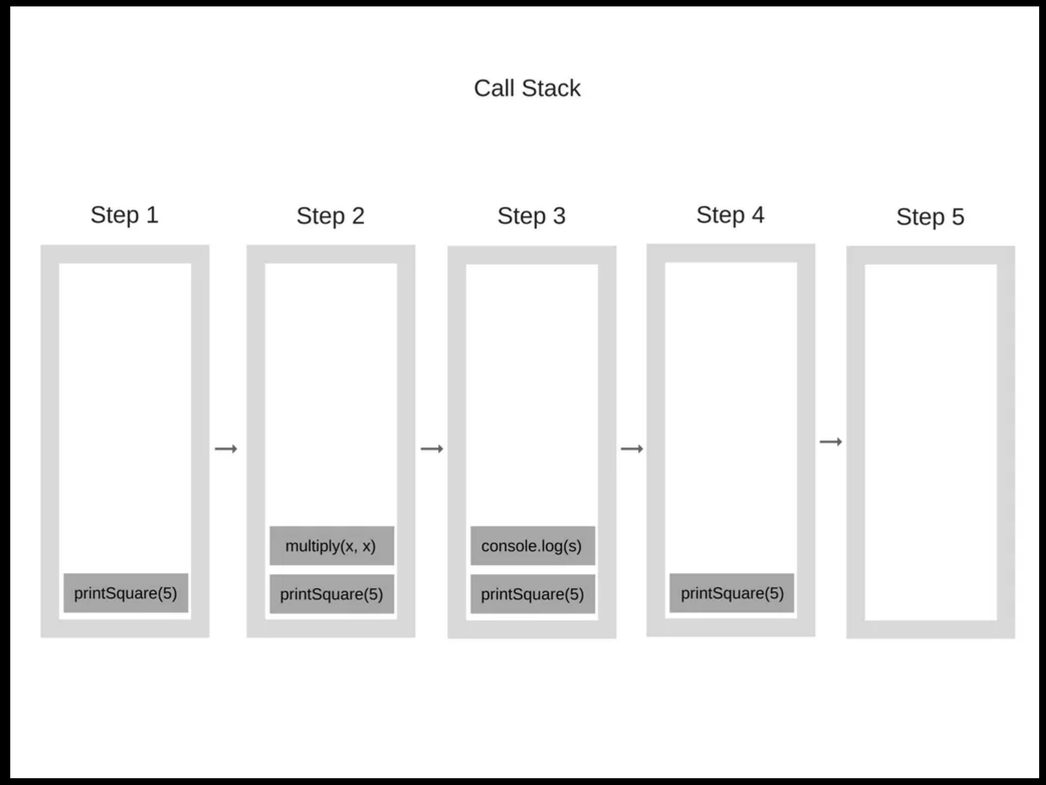


It work with many things inside runtime

Call Stack

```
function multiply(x, y) {
    return x * y;
}
function printSquare(x) {
    var square = multiply(x, x);
    console.log(square);
}
printSquare(5);
```

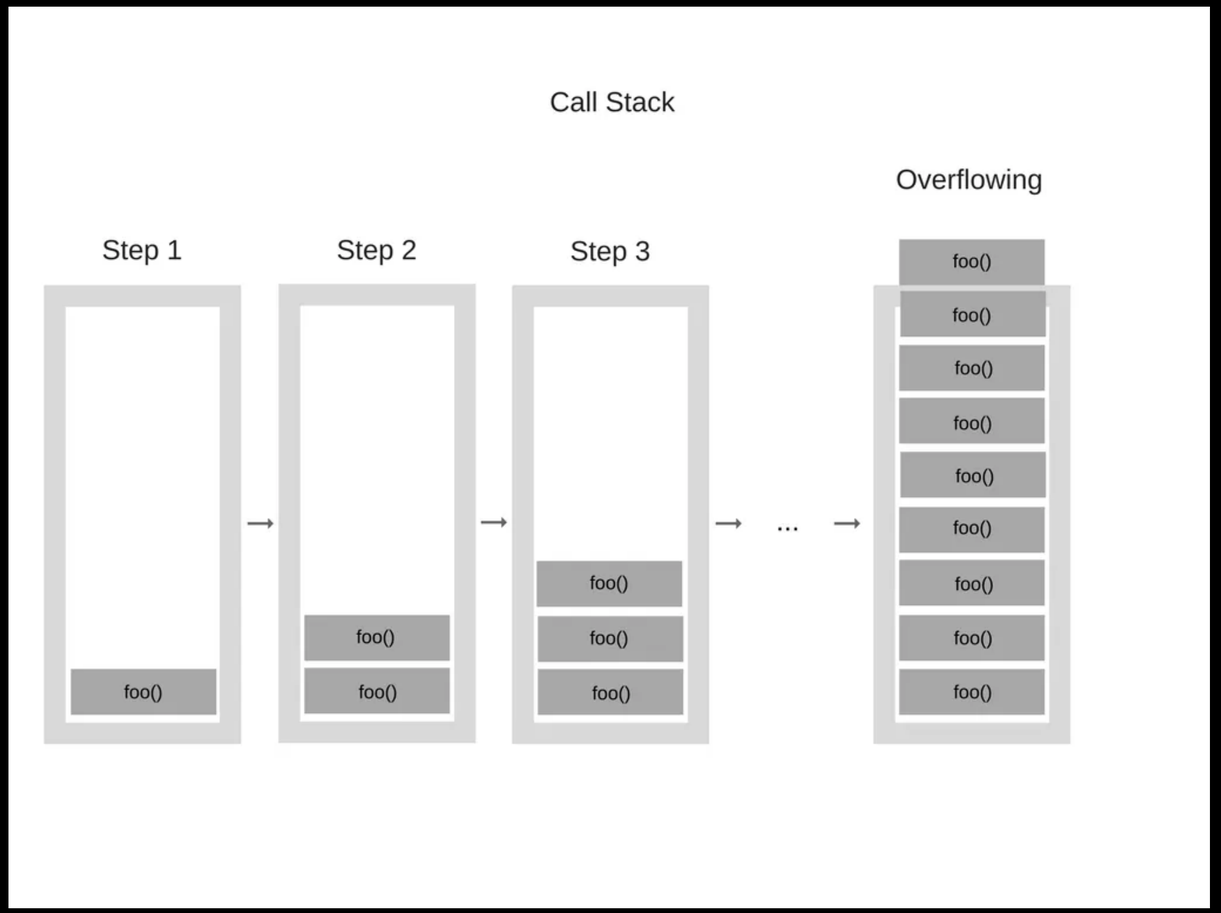
Where we are?



Maximum Call Stack

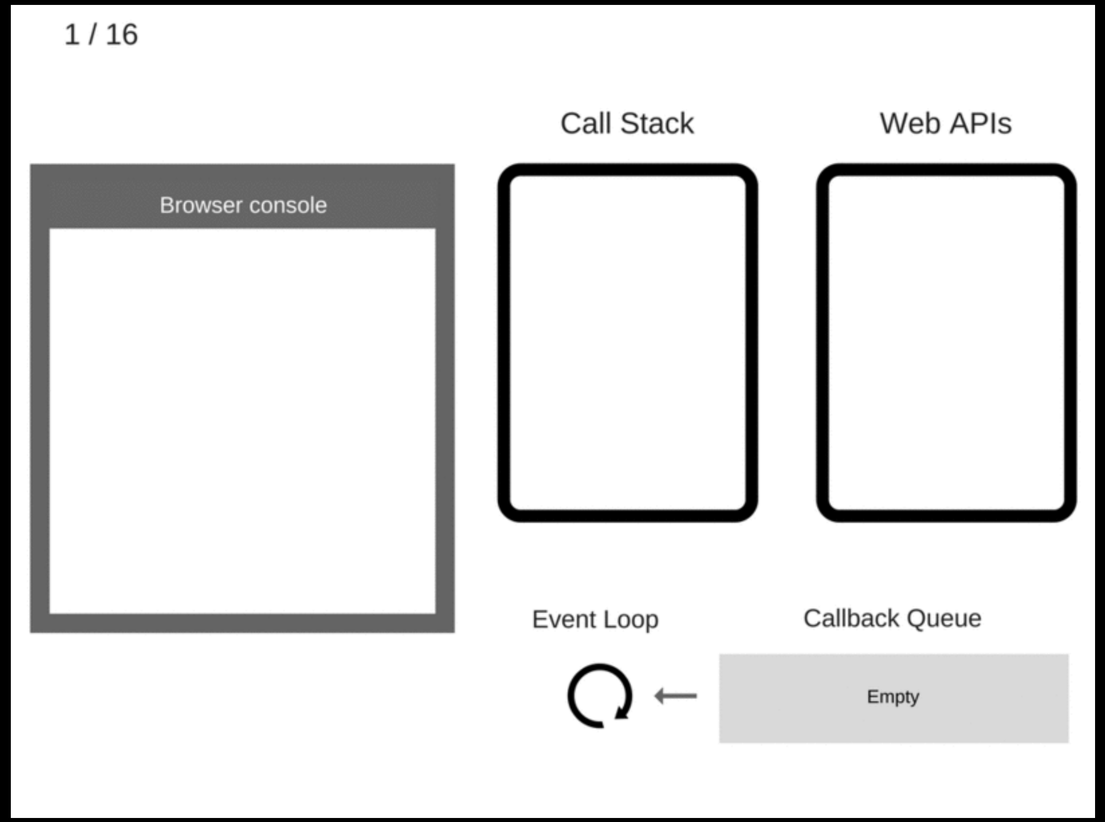
```
function foo() {
    foo();
}
foo();
```

```
▶ Uncaught RangeError: Maximum call stack size exceeded
    at foo (<anonymous>:2:5)
    at foo (<anonymous>:2:5)
    at foo (<anonymous>:2:5)
    at foo (<anonymous>:2:5)
    at foo (<anonymous>:2:5)
    at foo (<anonymous>:2:5)
    at foo (<anonymous>:2:5)
    at foo (<anonymous>:2:5)
    at foo (<anonymous>:2:5)
    at foo (<anonymous>:2:5)
```



See how setTimeout work

```
console.log('Hi');
setTimeout(function cb1(){
    console.log('cb1');
}, 0);
console.log('Bye');
```



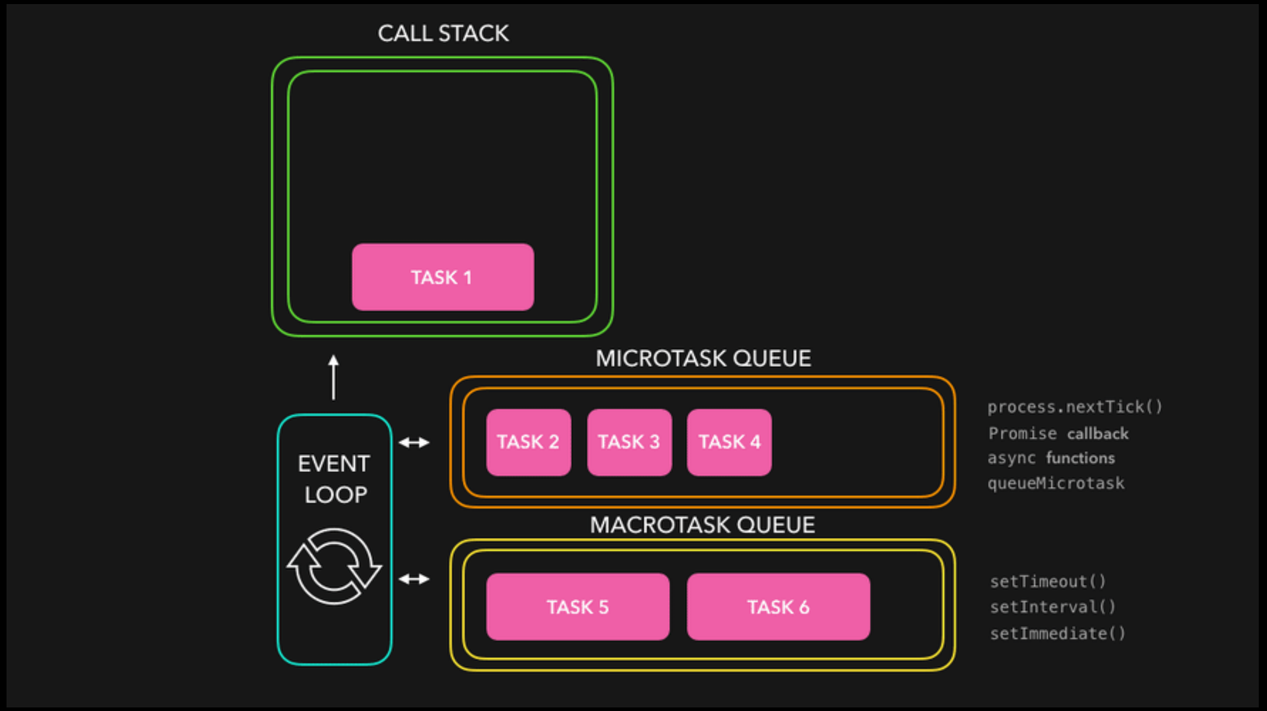
How setTimeout work

- setTimeout do not put callback to the event loop queue.
- it set a timer
- Time up, runtime push callback to callback queue
- Event loop will pickup callback from callback queue to execute

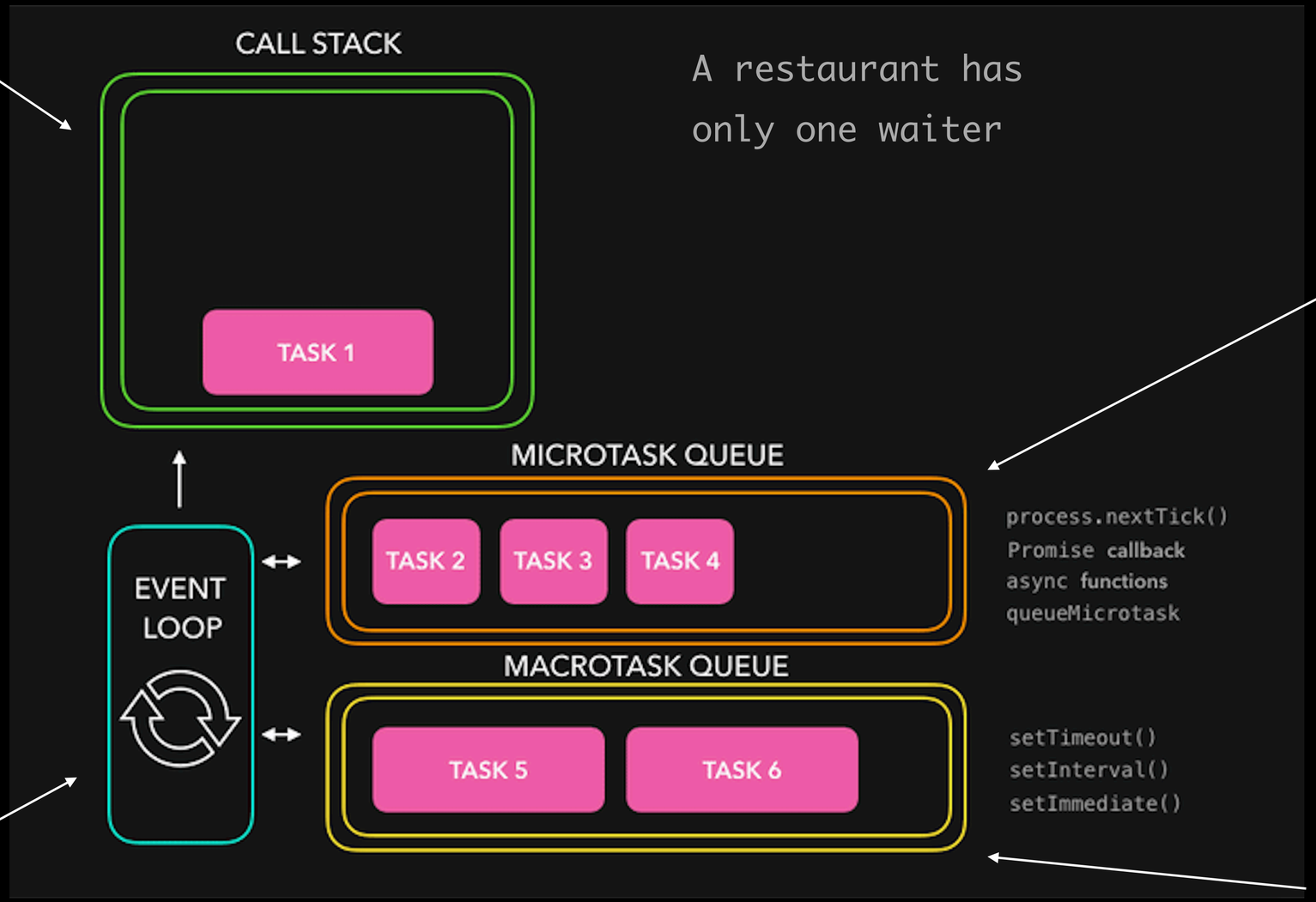
Event Loop

- javascript just do the synchronous code
- one thing at a time
- event loop: decide what to do next

  (when call stack empty)



Waiter



Vip

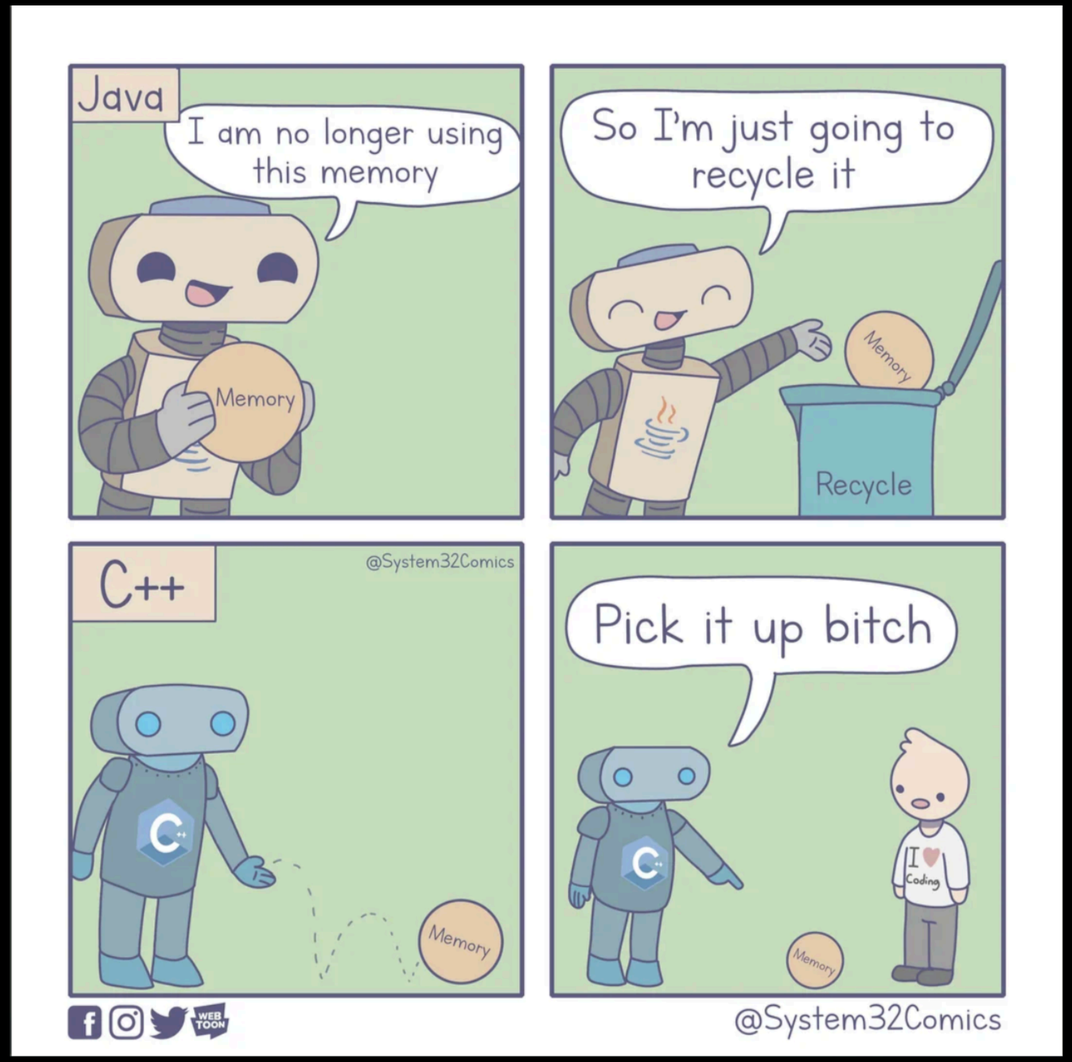
Manager

Member

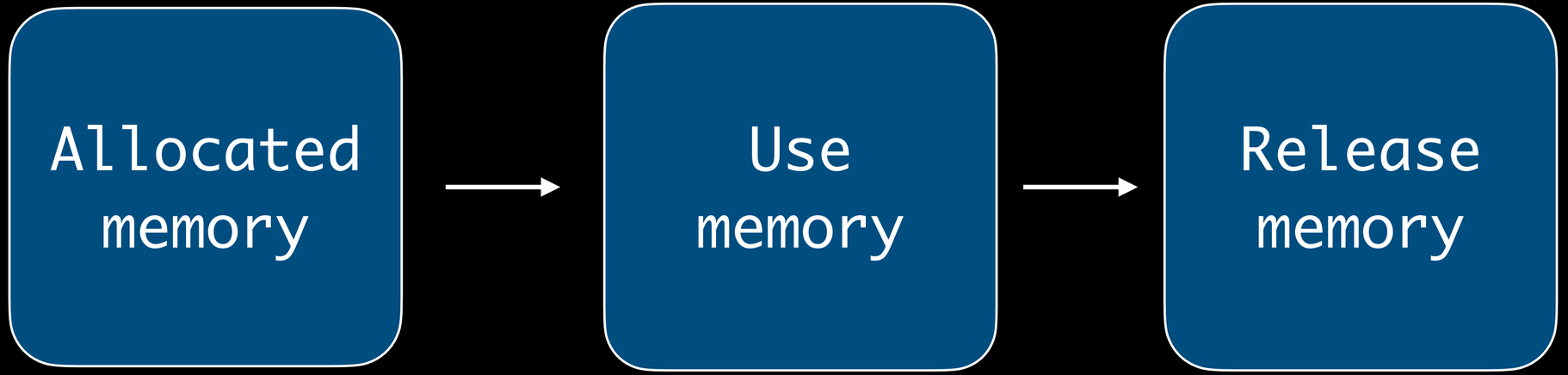
Memory management in Javascript

But if you choose not to care

about it, this was a big mistake



Memory Life


Allocated memory
Use memory
Release memory
Problem happen here

Release memory in javascript

Release memory

Garbage Collector

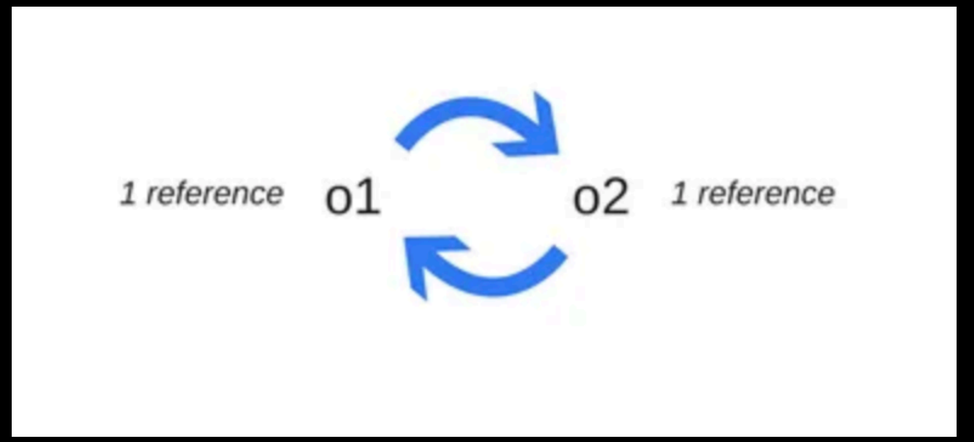
What is no longer using?

Reference Counting

An object is considered “garbage collectible”

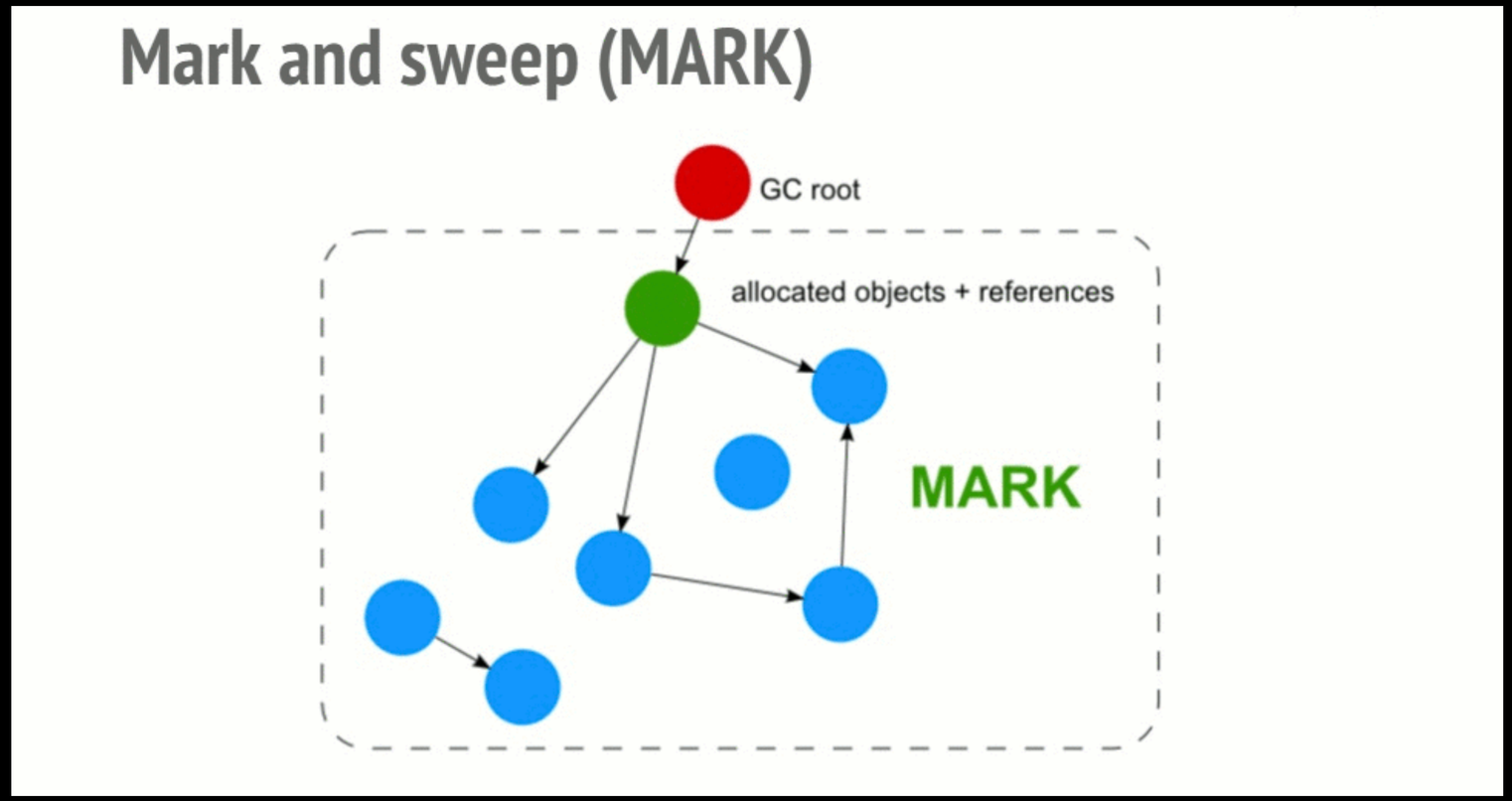
if there are no references pointing to it.

Issue with circle reference

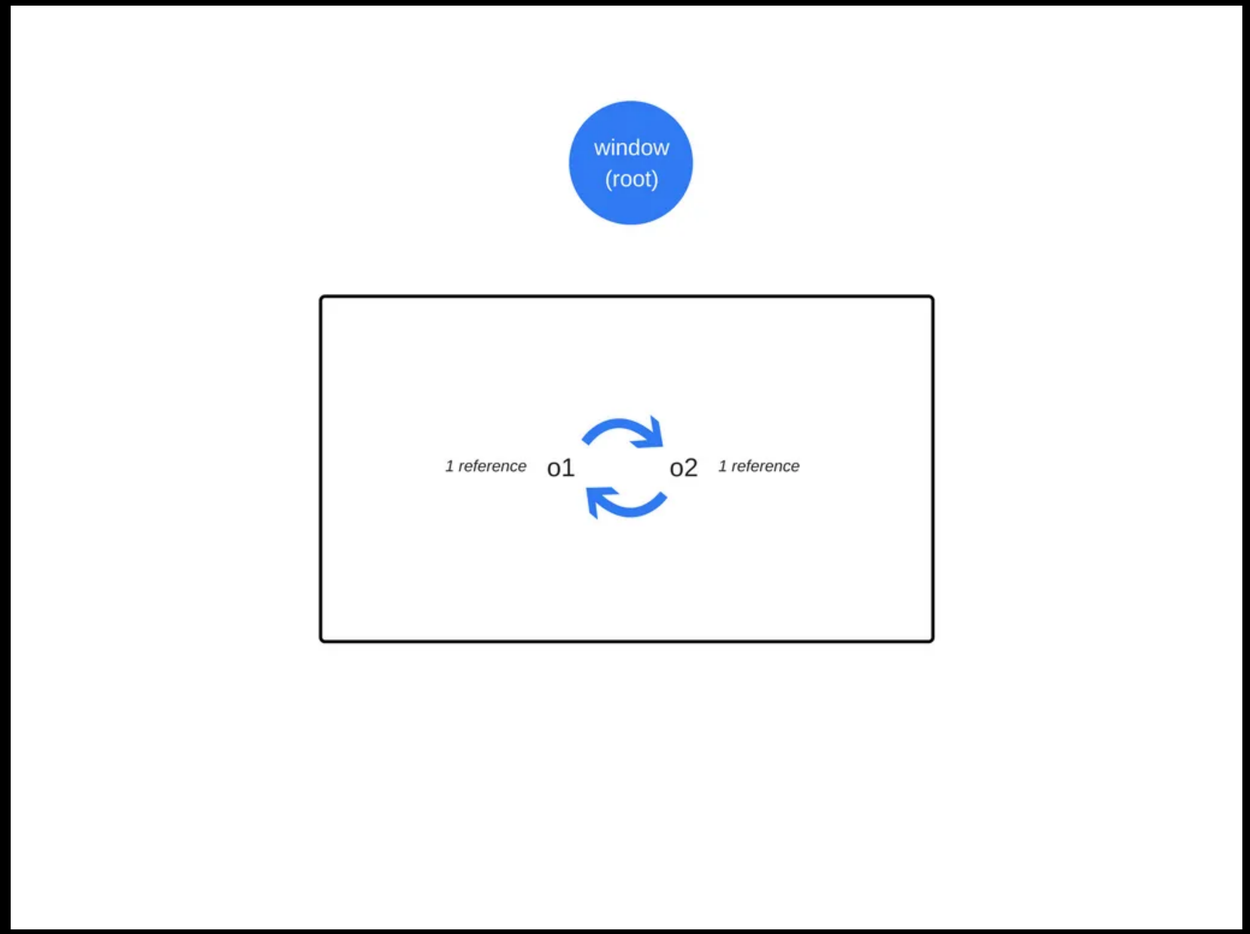


Mark and sweep

1. Root: Global variables
2. All reference from root



Circle issue solved



Common memory leak

- global variables
- Timers or callbacks that are forgotten
- Do not remove listeners
- Do not clear reference to unused object

Game Loop

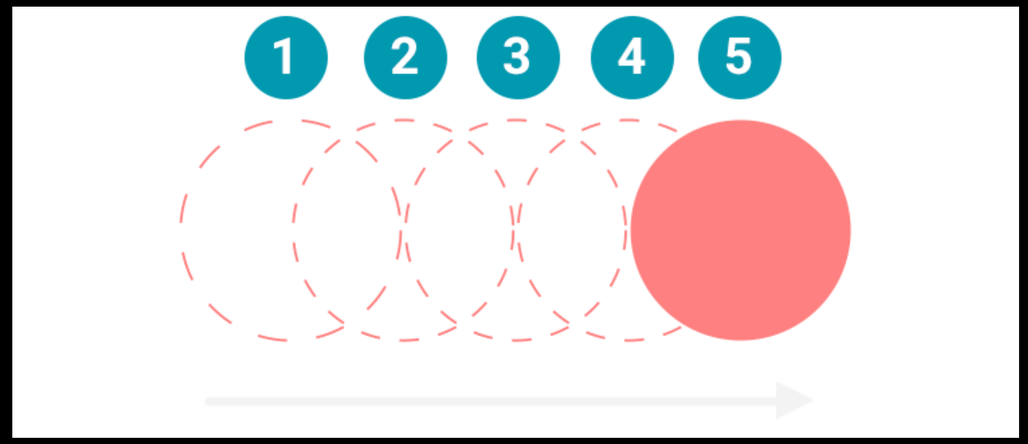
```
// A bad game loop
while (running) {
    draw();
}

// Another bad game loop
setInterval(gameLoop, 16);

function gameLoop() {
    draw();
}

setTimeout(gameLoop, 16);

function gameLoop() {
    // update
    setTimeout(gameLoop, 16)
}
```




```
let lastTime;
function gameLoop(timeStamp) {
    lastTime = lastTime || timeStamp;
    let dt = timeStamp - lastTime;
    draw(dt);
    window.requestAnimationFrame(gameLoop);
}
window.requestAnimationFrame(gameLoop);
```

Debugging

- Type of Errors
  - Syntax errors
  - Logical errors
  - Runtime Errors
  - Engine, framework errors

Strategy to debug

1. Reproduce the bug (find steps to make bug happen)
2. Trace the error (find the line make error)
   - breakpoint, debugger, console.log, console.trace
   - Watching call stacks, variables ’s scope
3. Fix bug
4. Test with reproduce steps