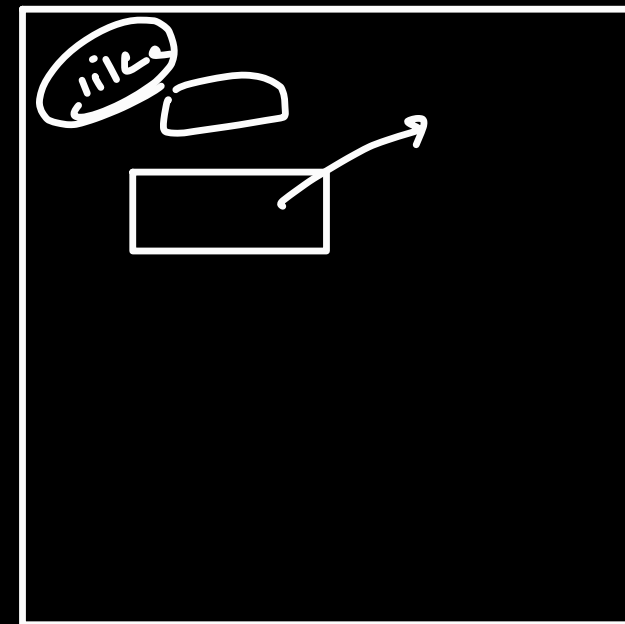


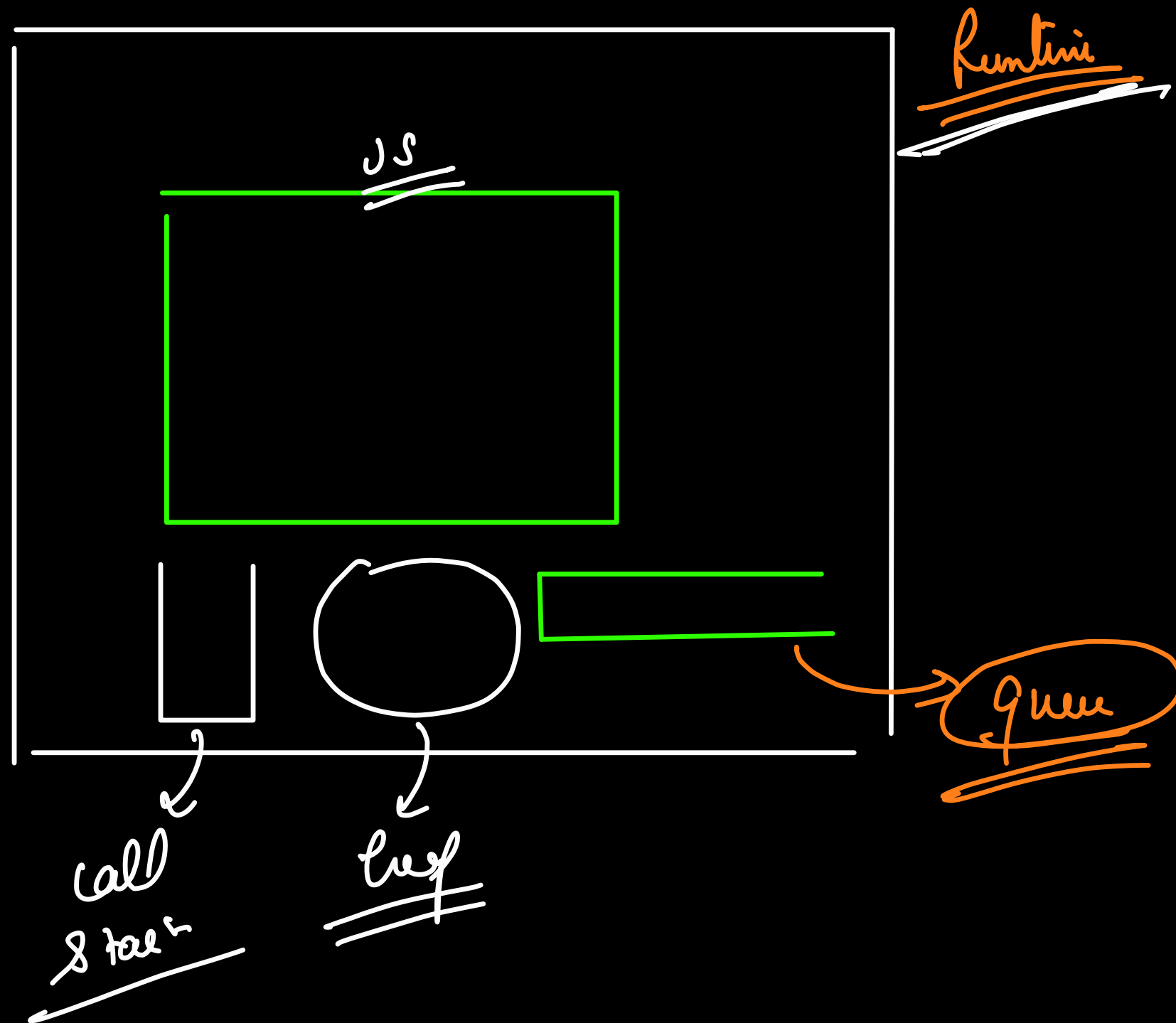
# Async Programming With JS

- 1) JS is sync in nature
- 2) JS is single threaded.

if we execute valid  
ecmascript code viz  
given by the standards



Browsers

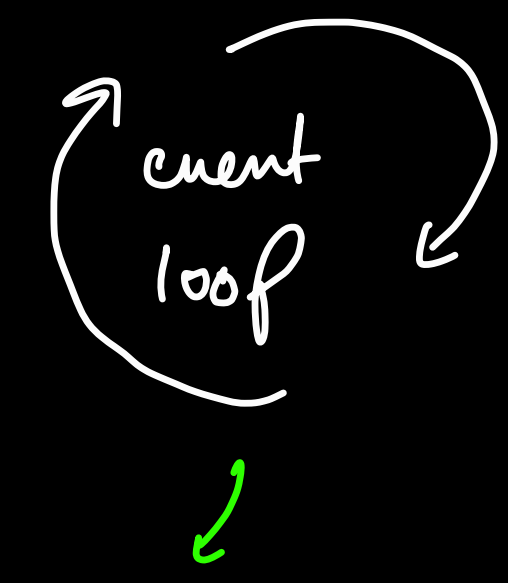
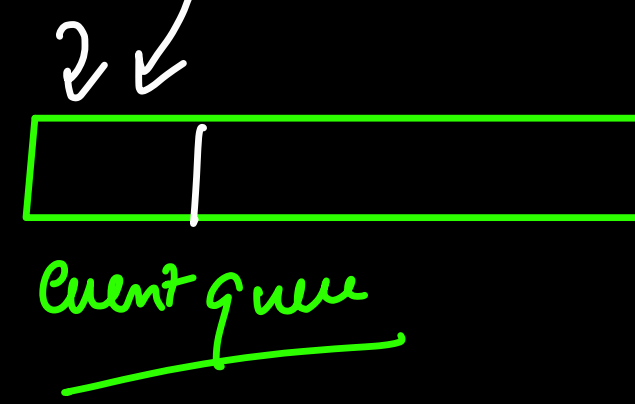
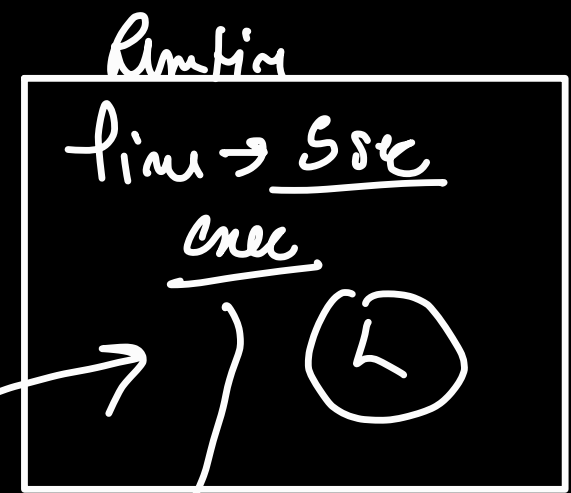


5 sec  
> 5 sec

```

11 function timeConsumingByLoop() {
12   console.log("loop starts");
13   for(let i = 0; i < 10000000000; i++) {
14     // some task
15   }
16   console.log("loop ends");
17 }
18
19 function timeConsumingByRuntimeFeature() {
20   console.log("Starting timer");
21   setTimeout(function exec() {
22     console.log("Completed the timer");
23   }, 5000);
24 }
25
26 console.log("Hi");
27
28 timeConsumingByLoop();
29 timeConsumingByRuntimeFeature();
30 timeConsumingByLoop();
31
32 console.log("By");

```



Hi  
 loop start  
 loop end  
 start timer  
 loop starts  
 loop ends  
 By  
captured line

it keeps on checking  
 whether the call stack is empty or not &  
 no global code is left.

```

1 function timeConsumingByLoop() {
2   console.log("loop starts");
3   for(let i = 0; i < 100000000000; i++) {
4     // some task
5   }
6   console.log("loop ends");
7 }
8 function timeConsumingByRuntimeFeature0() {
9   console.log("Starting timer");
10  setTimeout(function exec() {
11    console.log("Completed the timer0");
12    for(let i = 0; i < 100000000000; i++) {
13      // some task
14    }
15  }, 5000); // 5 sec timer
16 }
17 function timeConsumingByRuntimeFeature1() {
18   console.log("Starting timer");
19   setTimeout(function exec() {
20     console.log("Completed the timer1");
21   }, 0); // 0 s timer
22 }
23 function timeConsumingByRuntimeFeature2() {
24   console.log("Starting timer");
25   setTimeout(function exec() {
26     console.log("Completed the timer2");
27   }, 200); // 200 ms timer
28 }
29 console.log("Hi");
30 timeConsumingByLoop();
31 timeConsumingByRuntimeFeature0();
32 timeConsumingByRuntimeFeature1();
33 timeConsumingByRuntimeFeature2();
34 timeConsumingByLoop();
35 console.log("By");

```

10 sec

105 sec

Runtime

time0 → 5 sec, exec0

time1 → 0 sec, exec1

time2 → 200ms, exec2



Event queue



Non-Blocking

hi  
loop starts  
loop ends

Starting timer

Starting timer

Starting timer

loop starts

loop ends

By  
Completing time1

Completing time2

Completing time0

Call stack