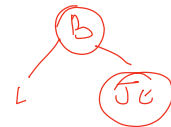# Day28(single thread , async programming , web apis, event loop , callbacks , callback hells)

JS-PL → weakly typed
        → dynamically

JS → PL

B
L ── JS
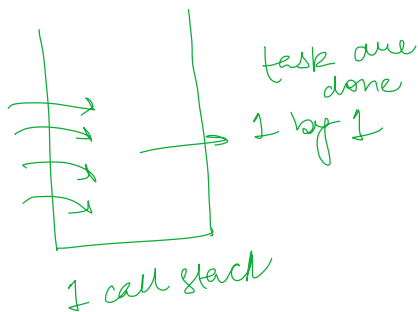
single thread
JS engine 1 baar mei sirf
1 statement ko
execute kar
sakta hai.

① weakly typed
② dynamically
③ Synchronous
④ Single threeded

PL

1 callstack

task are done
1 by 1

1 call stack

Synchronous → execution
done in order.

① ② ③

BF   P   ed

statement
time , tide & JS wait for none

eg. clg ('about');
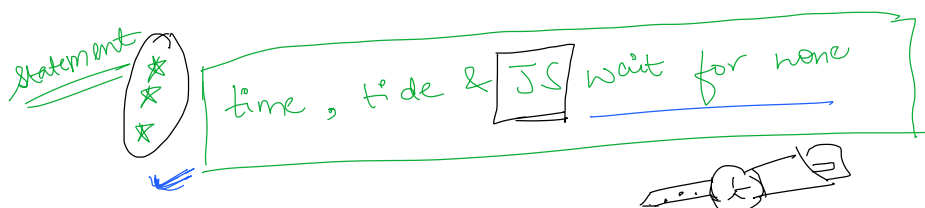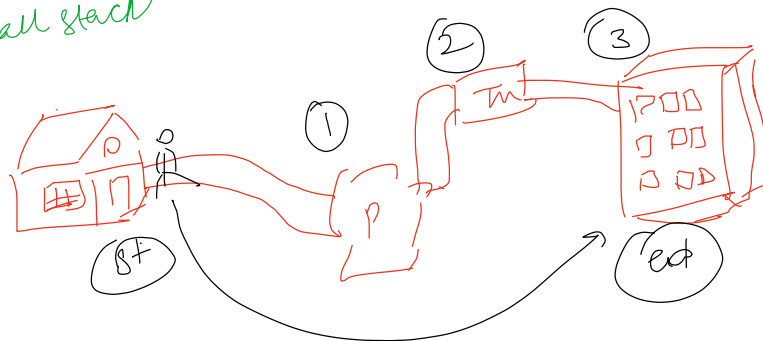
Prr kabhi na kabhi hame jarurat padh sakti hai ki hame code ka flow kuch der ke liye rokhna padhe aur JS m koi aisa tarika nahi hai jiske through hum code ka flow rokh sake. Toh fir kaise karenge ?.

(Top handwritten)

```
# -()
  clg ('after 4secs')  ✓
  clg ('end');  ✓
```

JS ✗✗
Katai Nai

method
↓
(count time)

☑ SetTimeout ( ) → Browser

WebAPT
Browser

No, this method
it is present inside
[browser] ✓

→ accepts 2 param.

Set Timeout (() ⇒ { } , 4000)   ↑ in (ms)

The window object is the main entry point to all the Web APIs provided by the browser. When the browser loads a web page, it creates a window object this object includes properties and methods that give you access to the browser's capabilities.

SetTimeOut method is present inside browser not in JavaScript.

Agar ye JS ke ander present nahi hai toh hum isse use kaise kar paate hai ?

Kyuki jab bhi JS ka code execute hota hai toh GEC banta hai aur uske sath ek Global object bhi banta hai aur ye method global object ke ander present hai aur hamare browser ke pass global object ka access hota hai jiski wajeh se hum iss method ko use kar pate hai

Jab bhi Browser web page ko load karta hai tab GEC create hota hai aur GEC ke sath ek Global object bhi create hota hai.

Global object ke pass browser ki jitni bhi properties or methods hai unn sab ka access hota hai.

```
4s → 4000 ms
10s → 10000 ms
```

Browser

JS ✗

Call Stack

Location,
Local storage
set Timeout
set Interval
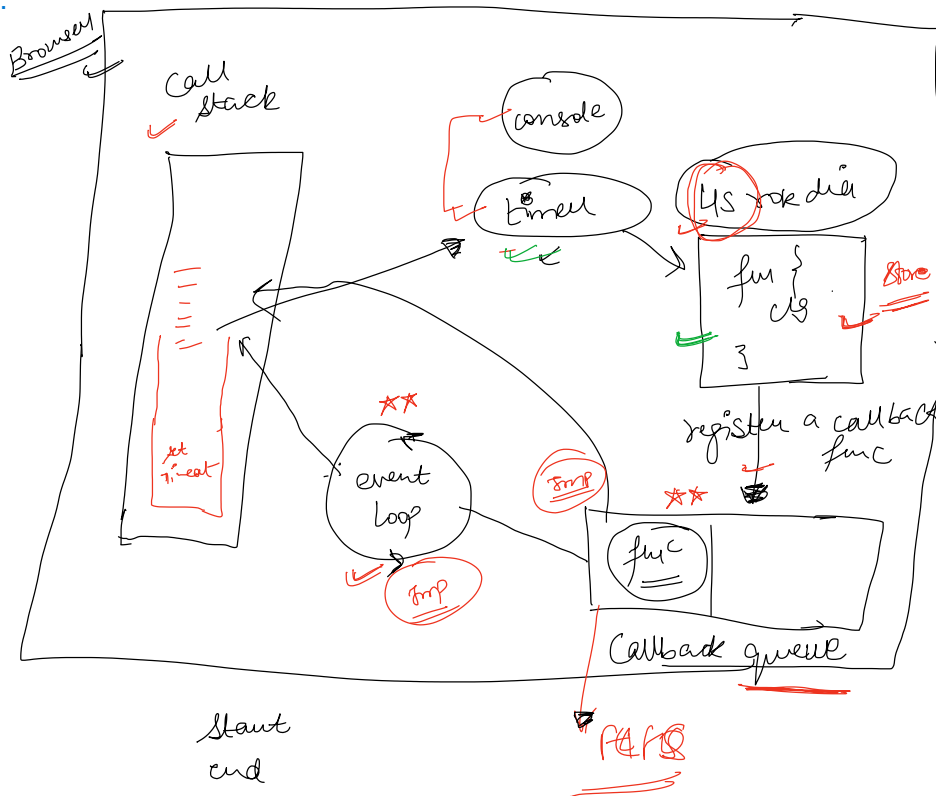null
console

Web APIs

All these are Browser's method not JS methods

In Web Api's ko hamara JS directly access/use kar sakta hai kyuki browser usko access provide karta hai.

Browser (methods)

Browser

Diagram 1 (top)

console

Kaam

Call stack

start
end
Output

Diagram 2

Call stack

Console

Browser

Laam

func

CB register

1. Jab bhi Browser Web Api's use hoti hai JS ke code ke ander toh Javascipt Browser se inn properties ko access karta hai aur browser inn Api's ko storage provide kar deta hai utne time ke liye jitne time ke baad ye execute hongi.
2. Aur execute hone ke baad ye callback queue m aake store ho jati hai.

Diagram 3

Browser

Call stack

No

No

console

timer

JS ok dia

fun {
CS
}

Store

set timeout

event loop

tmp

register a callback func

func

tmp

Callback queue

start
end

FIFO

Callback queue ka kaam hai ki brower ki web Api's ko store karke rakhega aur unn functions/ callback ko rokke rakhega jab tk unn ki turn na aa jaye apne ander(means jab tk event loop unhe callstack m nahi bhej deta).

WebAPI's (super power)

Store (storage) → jitna intzaar

callback queue? →

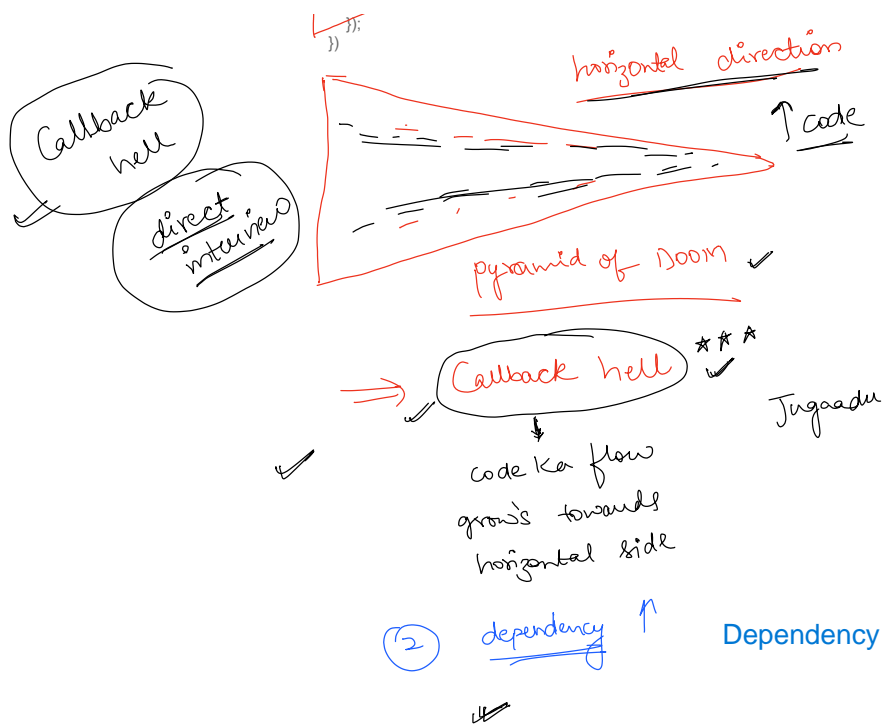| f1 | f2 | f3 | f4 |

event loop?

→ to check ki kya CS khali?

Event loop check karega ki callstack khali hai ya nahi agar khali hai toh callback queue m check karega agar usse koi function milta hai callback queue m toh usse callstack m bhej dega.

2S
4S

timer
2S
$f(2S)$
$-2S$
4S
$-2S$
$-2S$
$-2S$
② $f(4S)$
store
JS
$clg(end)$
$f(4S)$
$clg(start)$
CS

$| f(2S) | f(4S) |$ → cbqueue

API

Callback nesting

advantage →

Pros

1st func^n Khatam

vske based   2nd func^n

vske based   3rd func

wfk based   4th func^n

Helps in performing sequential task.

Cons   disadvantage   ② problem

① callback hell

②

loop

step1(function(image){
  step2(image , function(filteredImage){
    step3(filteredImage , function(finalImage){
      step4(finalImage);
    })
  })
})

These callback are growing towards horizontal diection. It is also called pyramid of DOOM or Callback Hell.

└ }};
    })

horizontal direction

↑ code

Callback
hell

direct
intuview

pyramid of Doom ✓

⟹ Callback hell ★ ★ ★ ✓

Jugaadu

code ka flow
grow's towards
horizontal side

② dependency ↑   Dependency b/w all callback are high.

Solution ?

table

Callbacks ✓

↓

Promises

↓

Async - Await

Imp°