Theory

1) Callbacks
2) Promises
3) How Asynchronous code works
4) setTimeout, setInterval
5) Async/wait → intro
6) H.W. → automation

5) H.W
github-signup
Postman

7) H.W.
8) ✓

setTimeout(cb, 5000);

asynchronous

1)
2)
3)
4)

cb → watch
5000 ?
10 000 ms

browser — web API

node — node API

synchronous — asynchronous + other func^n

chrome → v8

JS engine → v8 — web API

console.log → browser console.log

setTimeout()

node → node → C++ → node runtime

console → object — backend

{.log, table} methods — API → // X

backend { Application Programming Interface

console.log( );

console.log( ) .table( )

1) 2) read exec → browser

web API

Keep checking for func^n in cb queue, if call stack is empty, then cb run from queue to call stack

cb — call stack — Event Loop — callback queue

```
function yolo(){
    var a = 10;
    function cb () {
        console.log("hello how are you ?");
    }
    // function to be executed(callback fun)
    // 1 second =1000 milli second
    setTimeout(cb, 5000);
    console.log(a);
}

yolo();
```

synchronous

console

10
undefined
hello how are you ?

```
for (var i = 1; i <= 10; i++){
    setTimeout(function () {
        console.log(i);
    }, 2000);
}
```

var

A	C
11	11

var i=1 to 10 → global scope

for(var i=1;i<=10;i++)

ST (cb,2000);

}

i=1 2 3 4 5 6 7 8 9 10 11

function cb(){ console.log(i); } — new variable

cb, cb, cb, cb, cb, cb, cb, cb, cb, cb

```
for (let i = 1; i <= 10; i++){
    setTimeout(function () {
        console.log(i);
    }, 2000);
}
```

let → block scope

console: 1, 2, 3, 4, 5

i=1 2 3 4 5 6

web API: cb,i,2000; cb,i,8000; cb,i,6000; cb,i,4000; cb,i,2000

CS: cb

EL

CQ: cb, cb

1 to 10 print

2 are 1, 2 — 2 are used

3 — 2 are

4 — 4 is print