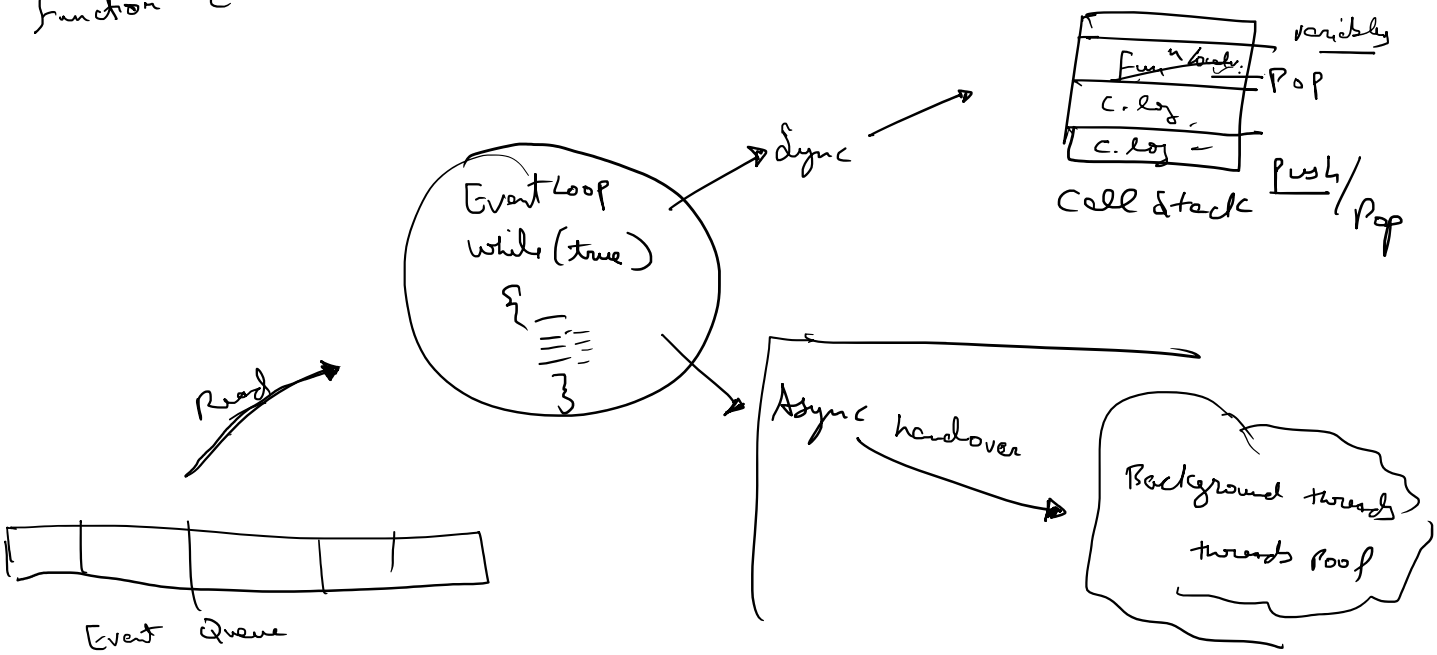
22 September 2024 14:00

Closures

Function countLikes() { }




```
function init(){
let count=0; // Lexical Scope
var countLikes= function (){
  count++;
  return count;
}
  return countLikes;
}
undefined
var c=init()
undefined
c();
```

var c = init()

Function init() { }

return fn

Function + Lexical Scope ⇒ Closure

Function + Lexical Scope

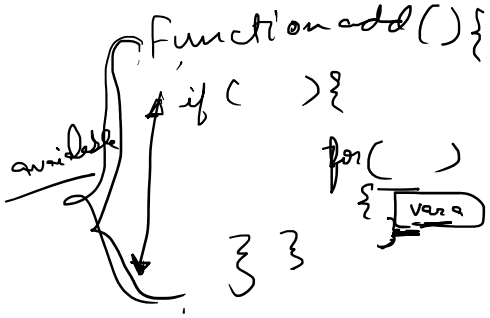
Function return

let, Const → 2015 ES6

Block scope

IIFE → Likes counter

Block level scope

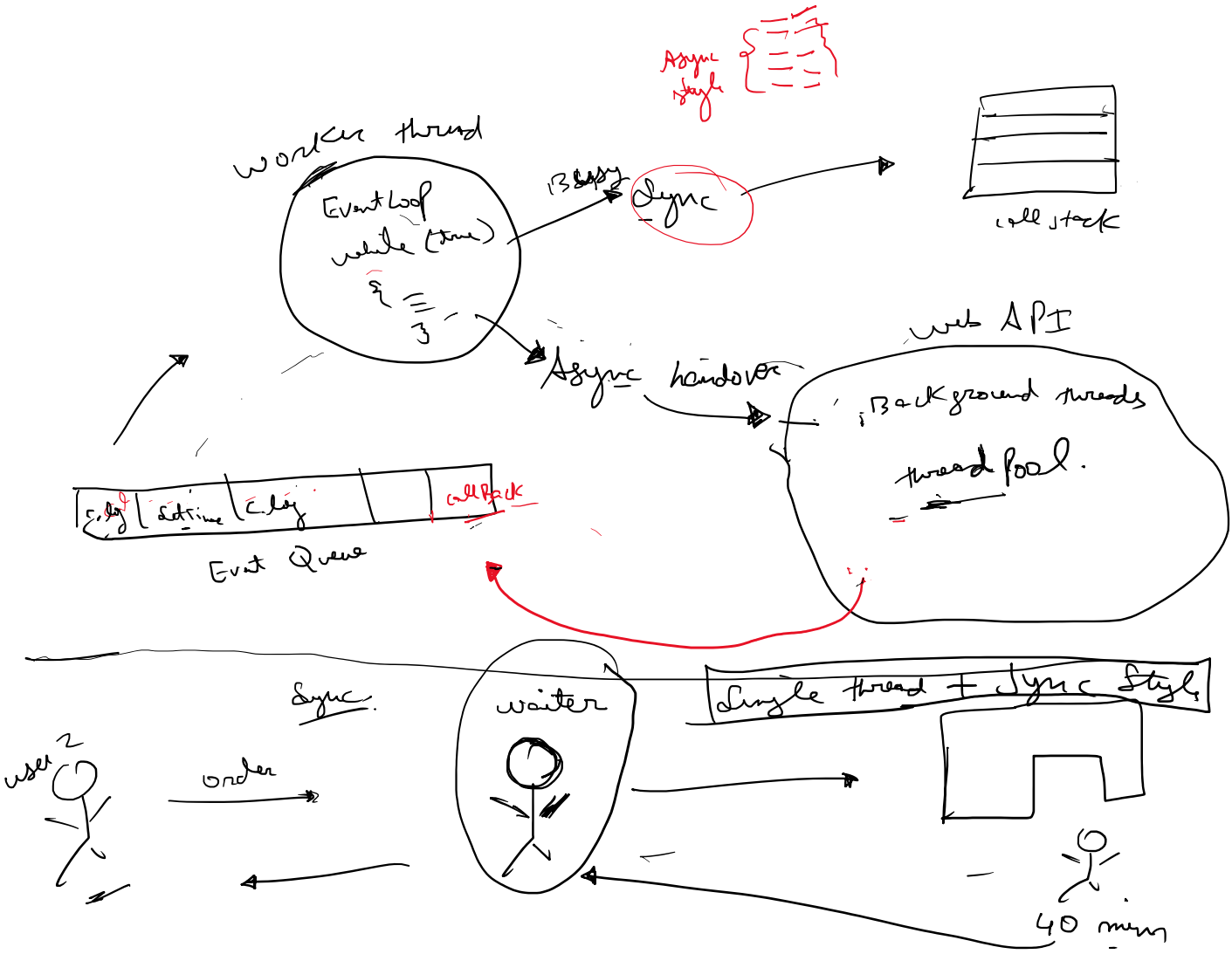


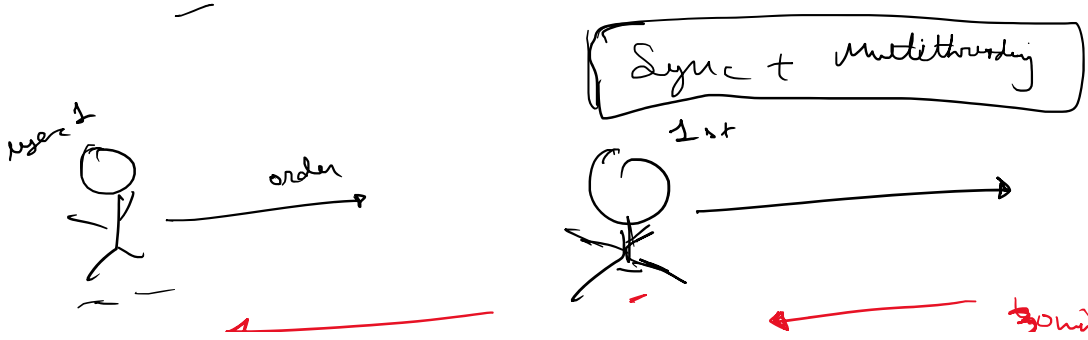
Blocking

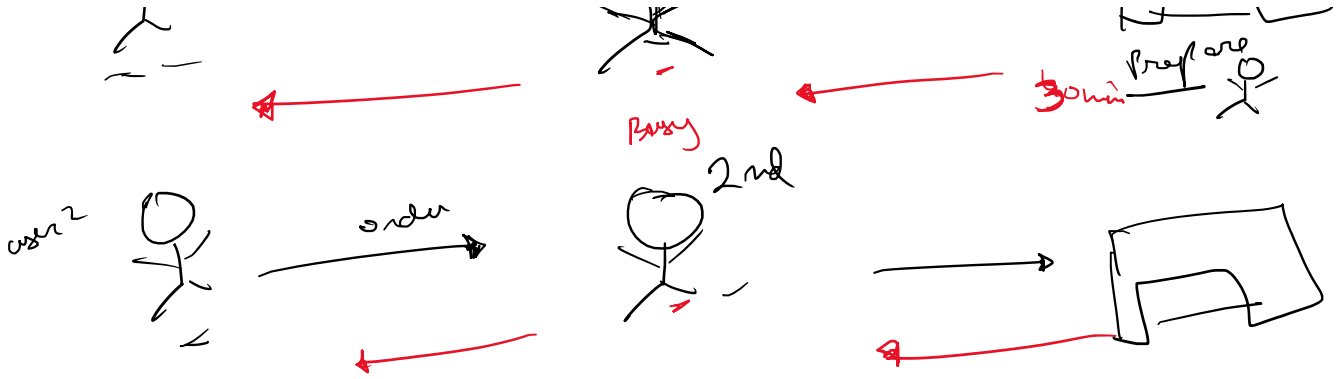
ASYNC PROGRAMMING

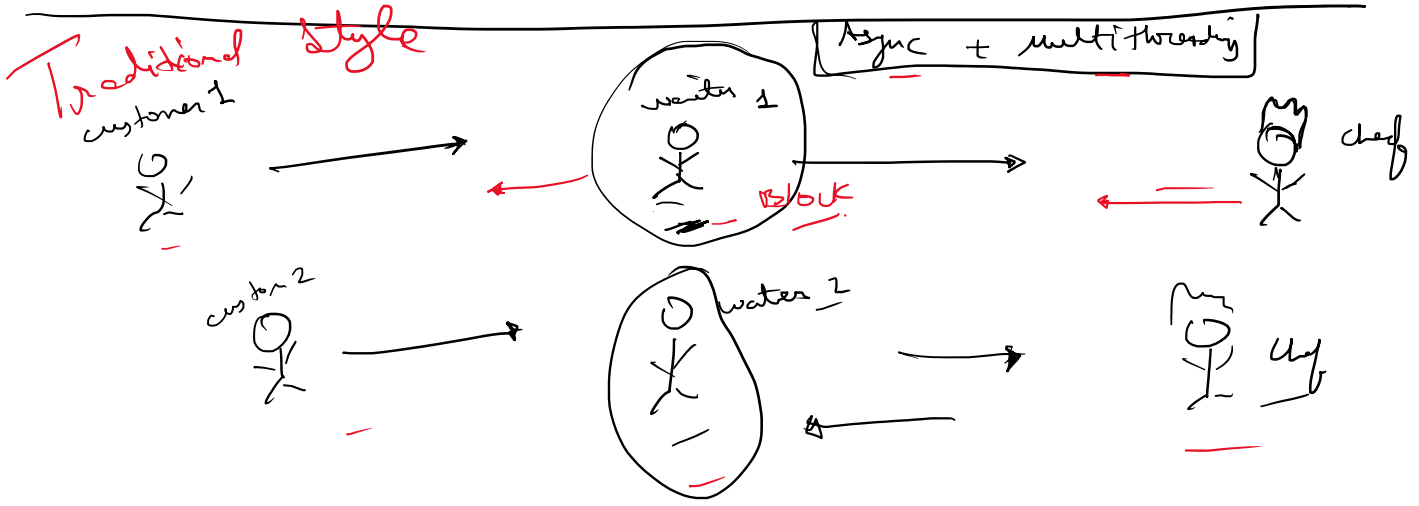
Sync → Blocking
waiting mode
call + wait + result

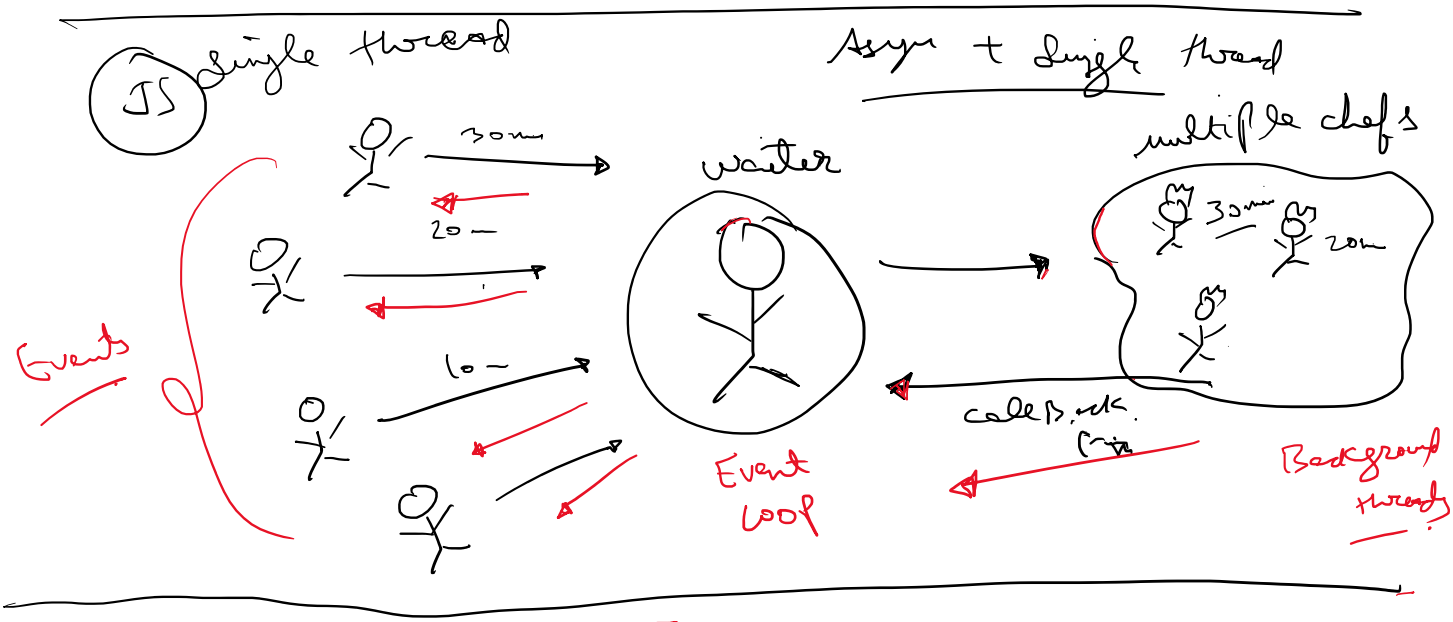
Async → Non-Blocking
call + forget + callBack











Async. Network calls()

call →
← data

to N.C.

Sync style

JS. 2 functions

1. setTimeOut ()

② setTimeout()


```
function doBiggerTask(){
  setTimeout(()=>{
      console.log('inside timeout fn');
      for(var i=1;i<100000;i++){
      for(var j=1;j<100000;j++){}
          console.log('big task!!!!!!!!!!!!!!!')
  return "bigger task done";
  },3000);
}
undefined
console.log('1st line');
console.log(doBiggerTask());
console.log('3rd line')
```

handover → thread pool.

call + forget + callback

receive X

1st line → undefined

3rd line

---

match score , weather Data

clock.

Dependent calls.

order() → Payments()

Payments() → deliveryInfo()

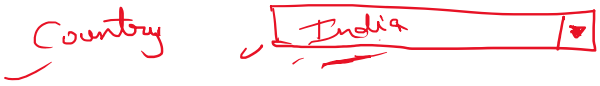
order() {

dependent calls → Payment() {

deliveryInfo() {

tracking() {

} } }



Lajpat Nagar,
Shalimar Bagh,
Defence Colony,

Login → Dashboard

login() {
  Dashboard() { }
}

async → callBack → async → callBack → async → callBack

country → result → State → result → Zone → ...

Callback Hell

Call + forget + callBacks

Pyramid Problem

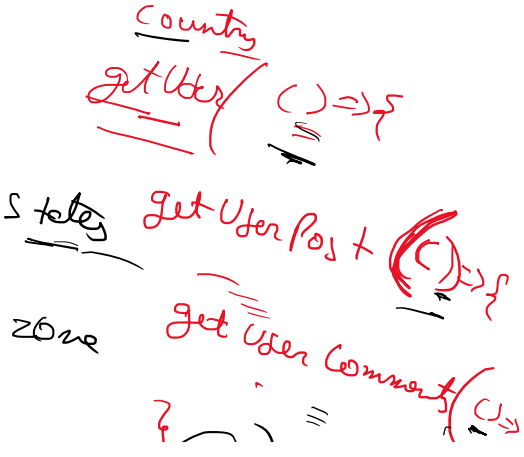
↳ hard to read

↳ hard to manage.

↳ Difficult to find Bugs

```
// Simulating async operations using setTimeout
function getUser(userId, callback) {
 setTimeout(() => {
 console.log('User found');
 callback({ id: userId, name: 'John' });
 }, 1000);
}

function getUserPosts(userId, callback) {
 setTimeout(() => {
 console.log('Posts for user found');
 callback(['Post 1', 'Post 2']);
 }, 1000);
}
function getPostComments(postId, callback) {
 setTimeout(() => {
 console.log('Comments for post found');
 callback(['Comment 1', 'Comment 2']);
 }, 1000);
}
// Callback hell structure
getUser(1, (user) => {
 getUserPosts(user.id, (posts) => {
 getPostComments(posts[0], (comments) => {
 console.log('User:', user);
 console.log('Posts:', posts);
 console.log('Comments:', comments);
 });
 });
});
```



```
    console.log('Comments:', comments);
  });
});
});
```

Promise

Promise ← total ()

return

Mcd.

orderPlace → receipt / Promise

resolve → success

reject → error.

Promise

→ Pending — } - 2 states

→ fullfilled / Reject -

3 states

Pending, fulfilled, reject