Call back Function
A callback function is a function where we pass parameters as another function, which is called from a function.

Callback function mhnje apn yeka function la parameter mhnun dusr function deto ani te function madhun call hot

Or callback function means function call from another function

Example ->
1st parameter ->name mhnje nav ghetoy
2nd parameter (callback) -> ha reference ahe SayBye() function cha

```
function greet(name, callback) {
   console.log("Hello " + name);
   callback();                          //callback() ni sayBye function call kel
}

function Bye() {                 => he call back function ahe kran te greet
   console.log("Goodbye!");           function madhun call kel ahe
}

//calling greet function and pass parameter name and ref of sayBye
function
greet("Sai", bye);
```

What is the benefit of callback?
1-to handle asynchronous operations
2-improve code flexibility

Disadvantage
1->call back hell problem
2->difficult to read and debug or understand

Example->Map,filter ,reduce,setTimeOut ,setInterval,forEach

CallBack Hell

When we call one function with another function and another function call in other function is call callback hell

Callback Hell happens when multiple functions are nested inside each other and calling each other using callback and making the code hard to read, understand, and maintain.
It is also called the “Pyramid of Doom”.

Callback hell in js mhnje jevha yek function dusrya function la call krt te dusr function kontya tri tisrya function la call krt as chalu ch ast tr ya process la call back hell mhntya

Callback hell kadhi hoto jevha yeka peksha jast function yeka madhe yek astya an yek mekana call krtya using callback
Ya mule code samjayla ,read ani maintain krayla avgad jato
Call Back hell la ch pyramid of doom mhntya
Real time example->break fast kraycha ahe →pani tapvl =>chaha bnvla=>khari anli =>chaha ghetla =>break fast zala

Example->

```
function boilwater(callback) {
  console.log("boil water");
  callback();
}

function makeTea(callback) {
  console.log("making tea");
  callback();
}

function maketost(callback) {
  console.log("making toast");
  callback();
}
```

```
function serv(callback) {
  console.log("breakfast done");
  callback();
}

// Nested callbacks
boilwater(() => {
  makeTea(() => {
    maketost(() => {
      serv(() => {
        console.log("done");
      });
    });
  });
});

// boil water
// making tea
// making toast
// breakfast done
// done
```

Why callback hell not use or why it is bad

1->hard to maintain
2->hard to read
3->difficult to error handling and debug
4->if one function get error all other function are stop automatically
Jr function error al tr sagl band hot

How to avoid Call back hell

1->use named function (not mainly used)
2->use promise
3->use Async and await (mostly use)

Promise

A Promise is an object that represents the result of an asynchronous operation where it is complete or fail
It allow to handle Asynchronous operations
It used in API call,file handling and fetching data
It solve problem of older method like call back hell

Promise ha js madhe yek object ahe jo result represent krto asynchronous operation ch ki te success ahe ki fail

हे प्रामुख्याने API call, data fetch, file read, database operation यासाठी वापरलं जातं.

State of Promise
1->Pending ->operation in progress not completed
Operation chalu ahe pn completed zalel nahiy

2->Fulfilled or resolve ->operation completed

3->Rejected ->operation is not completed
Error alya mul operation completed zalel nahiy

Real Life example
Promises are like ordering food online.
Order placed → Pending
Food delivered → Fulfilled
Order cancelled → Rejected

How to create Promise

```
let promise = new Promise((resolve, reject) => {
   //he 2 parameter compalrsary ahe jr use kraych nsel tr _ uesed kraych
resolve->sucess reject->error

  // some async operation   here promise work as constructor
  let success = true;
  if (success==true) {
```

```
    resolve("Task completed successfully!");
  } else {
    reject("Something went wrong!");
  }
});
```

how to use promise

```
promise
  .then((message) => {
    console.log("Success:", message);
  })
  .catch((error) => {
    console.log("Error:", error);
  })
 .finally(()=>{
    console.log("done");
    });
```

//example of odd even

```
let p = new Promise((resolve, reject) => {
  let n = 9;
  if (n % 2 === 0) {
    resolve("number is even");
  } else {
    reject("number is odd");
  }
});
```

Use promise

```
p.then((msg) => {
  console.log(msg);
}).catch((error) => {
  console.log(error);
});
//output number is odd
```

Advance Method of Promise

1-promice.all();

it will execute all the promice.if any one is reject then result rejected promise data is print

he method input as array gheti karan multipal promice gheyche ahe mhnun

2-promice.allSetled();

this will wait to complete the all promise wether it will success or reject it return all promise status

result in the form of array ani yacha input type pn array ahe

3-promice.race();

It return first promice that is settles mhnje ch 1st promice je resolve zalel ahe
Race ha concept timing vr depend age
input type as array gheto return jo resolve zala tyala print krto

4-promice.any();

It return the first fulfill promise mhnje jo 1st solve zala to yat time concept nst

Note->jr sagle promice reject astil tr yeto yeto =>aggrigateError

Example

```
const pp1 = Promise.resolve("Promice pp1");
const pp2 = Promise.resolve("Promice pp2");
const pp3 = Promise.reject("Promice pp3");
const pp4 = Promise.resolve("Promice pp4");

Promise.all([pp1, pp2, pp3, pp4])
  .then((data) => {
    console.log(data);
  })
  .catch((error) => {
    console.log(error);
  });

Promise.allSettled([pp1, pp2, pp3, pp4])
  .then((data) => {
    console.log(data);
  })
  .catch((error) => {
    console.log(error);
  });

Promise.race([pp1, pp2, pp3, pp4])
```

```
  .then((data) => {
    console.log(data);
  })
  .catch((error) => {
    console.log(error);
  });

Promise.any([pp1, pp2, pp3, pp4])
  .then((data) => {
    console.log(data);
  })
  .catch((error) => {
    console.log(error);
  });
```

Promise chaining
Promise Chaining means calling multiple .then() methods one after another,
where each .then() returns a value or another promise that the next .then() can use.

Promise chaining म्हणजे एका .then() नंतर दुसरा .then() असे एकामागून एक promise वापरणे.
प्रत्येक .then() काहीतरी value return करतो आणि पुढचा .then() ती value वापरतो.

Advantage ->It help us to avoid callback hell

Example->
1->fetch user data form API
2->get user order
3->get users total

disadvantage ->jr madhlya kontya pn function madhe error ala tr sagle stop hotya

async and await

In js async and await use to handle asynchronous operations in clean,readable way to solve promise chaining problem
They make asynchronous code look and behave more like synchronous code

Js madhe async and await use krto asynchronous operation clean,readable way madhe solve krayla ani promise chaining solve krayla

key component

1->async

declare function as async

async function always return promise

jr aplya await use kryahcha asel tr compulsory function aplyala async declare krayla lagt

2-await

it pause the execution of async function until the promise is resolved or reject

it only used inside the async function

Note->async and await always try and catch madhe ch lihaycha

advantage

1-simple code

2-faster to read and easy to maintain

3-easy ti error handling using try catch

Limitions

1-required modern browser and node 7.0+ version required

2-must always used with promise and async function

Example

```
//function bnvl mhnje apl logic tya madhe return promise
function msg() {
  return new Promise((resolve, reject) => {
    let s = true;
    if (s == false) {
      resolve("sucessful");
    } else {
      reject("un sucessful");
    }
  });
}
//new aysnc function bnvl tyat te call kel ani dislay
async function dis() {
  let res = await msg();
  console.log(res);
}
//asycn function call
dis();
```

Ajax in Java Script =>IMP

---

full form->asynchronous java script and xml
It allows web pages to send and receive data from a server in the background without reloading the entire page.

AJAX म्हणजे अशी पद्धत ज्यामुळे वेब पेज सर्व्हरकडून डेटा आणू किंवा पाठवू शकते
तेही पेज रीलोड न करता.

jr aplya page refresh n krta data anaycha kiva dynamically data change kraycha asel tr apn ajax use krto

old prb->
jr apn page madhe submit button drop down select,radio button use kele ani ani tyala data select kelya vr total page refresh hoych ani adhi fill kelela data sagla nigun jaycha jo apn text box madhe vagre bharel asaycha to

why use ajax
1->by using ajax we can display the data without refreshing the page
2->It used to communicate the server asynchronous without reloading or refreshing the page
3->It allow the web application to send and retrieve data from server in background
4->it improve the user experience

=>Purpose / Use:
Fetch data from a server (like APIs)
Submit forms without reloading the page
Update only part of a web page dynamically

To run the ajax we Required server
1->apache tomcate
2->xammp
3->live server (we mainly use)

key Features
1->asynchronous communication
  data exchange in background without disturbing user
2->partial page update
  only specific section of webpage are update instead of reloading entire page
3->support for multiple data formates

Ajax can handle JSON,XML,HTML,text,etc data

What happens if Ajax is not used
1->full page reload problem
2->slower interation--webpage reload many time make slower
3->poor user experience
4->high bandwidth usage --again again data transfer then consume more bandwidth

What is API?

API stands for Application Programming Interface

An API is a bridge that allows to communicate Frond end and backend .

📌 **In simple words:**

API = Request + Response system between client and server

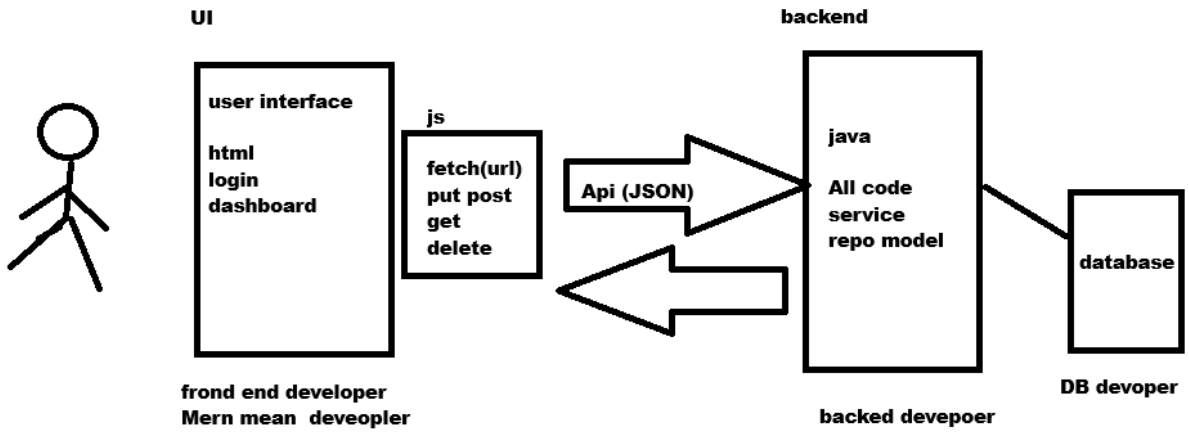
2️⃣ Real-Life Example 🍽️ (Very Important)

Restaurant Example:

- 👩‍🦱 **You → Client**
- 📋 **Menu → API**
- 👨‍🍳 **Kitchen → Server**
- 🍔 **Food → Response**

You don’t go to the kitchen directly
You place order via menu (API)
Kitchen sends food back

💡 **Same way:**
Frontend → API → Backend → Database



Why Do We Need an API?

✓ Frontend & Backend separation
✓ Data sharing between systems
✓ Secure communication
✓ Reusability
✓ Platform independent (Web, Mobile, Desktop)

JSON

JSON stands for JavaScript Object Notation.
It is a lightweight data format used to store and transfer data between client and server.

JSON is used to send data in text form.

Example

```
{
  "name": "Sai",
  "age": 22,
  "course": "MSc Computer Science"
}
```

Rules of JSON

Data is in key : value pairs

Keys must be in double quotes

Values can be: String ,Number ,Boolean,Array,Object,null

```
{
  "name": "Amit",
  "age": 21,
  "isStudent": true,
  "skills": ["Java", "SQL", "JS"],
  "address": {
    "city": "Pune",
    "pin": 411001
  }
}
```

JSON → Object

```
let obj = JSON.parse('{"name":"Saurabh","age":22}');

Object ->JSON
let json = JSON.stringify(obj);

//json
let student = {
  name: "Sai",
  age: 22,
  course: "MSc Computer Science",
};
console.log(student);
//object to string
console.log(JSON.stringify(student));
```

Why JSON is Used?

- **Fast data transfer**
- **Language independent (multiple use kru shkto)**
- **Easy to read & write**
- **Used in APIs (REST)**

Javascript object

```
let obj = {
  name: "Saurabh",
  age: 22
};
```

What difference between JSON and Javascript object

Point	JSON	JavaScript Object
Definition	**JSON is a text-based data format used to store and transfer data between systems.**	**A JavaScript object is a data structure used inside JavaScript programs to store data and logic.**
Usage	**JSON is mainly used for sending and receiving data between client and server.**	**A JavaScript object is mainly used for writing program logic and handling data.**
Data Type	**JSON data is always in string format.**	**A JavaScript object exists in memory as an object.**
Quotes	**In JSON, all keys must be written inside double quotes.**	**In a JavaScript object, keys can be written with or without quotes.**
Language Support	**JSON is language independent**	**A JavaScript object works only in JavaScript.**
Comments	**JSON does not allow comments.**	**A JavaScript object allows comments in the code.**

`IMP`

`Ajax method`

`fetch()` is a **built-in JavaScript method** used to **request data from an API** (server) and get a **response**

`fetch()` returns a Promise

Q How to fetch data from API

Syntax of fetch

```
fetch("API_URL")
  .then(response => response.json())  //response data
  .then(data => {  //data string
    console.log(data);  //print
  })
  .catch(error => {    //error
    console.log(error);
  });
```

Example

1->normal console vr data display kraycha asel tr

```
fetch("https://jsonplaceholder.typicode.com/users")
  .then(response => response.json())   // convert response to JSON
  .then(data => {
    console.log(data);                 // fetched data
  })
  .catch(error => {
    console.error("Error:", error);
  });
```

2->html chya page vr display kraycha asel tr

Abc.html ->as yek list bnvaychi

**<ul id="userList">
**

main.js

```
fetch("https://jsonplaceholder.typicode.com/users")
  .then(response => response.json())
  .then(users => {
    let list = document.getElementById("userList");

    users.forEach(user => {
      let li = document.createElement("li");
      li.innerText = user.name + " - " + user.email;
      list.appendChild(li);
    });
  })
  .catch(err => console.log(err));
```

Using wait

```
async function fetchUsers() {
  try {
    let response = await fetch("https://jsonplaceholder.typicode.com/users");
    let data = await response.json();
    console.log(data);
  } catch (error) {
    console.log("Error:", error);
  }
}

fetchUsers();
```