Asynchronous JavaScript :-

→ **Synchronous** :- matlab ek ke baad dusra hoga, jab tak ek command complete naa ho, dusra shuru nahi hoga.

→ **Asyn** :- matlab saare kaam ek saath shu
kardo, jiska answer pahle aajaye uska jawaab dedena.

set Timeout
set Interval
promises
fetch
axios → package used for api call ie. third party data taking.
XML HttpRequest

} Asynchronous

console.log ("12") - synchronous

// Kai baar aapka final code depended hota hai kisi aur ke server par, is case mein humein nahi pata hota ki answer uske server se kab laut kar aayega, to

Async JS

→ setTimeout - isko code kuchh der baad chalta hai.

→ setInterval - isko code kuch der baad chalta hai baar baar ek particular interval time mein.

→ Fetch API - ye kissi aur url par jaa kar kuch data laayega ya data humaare paas se us url par lekar jaayega.

→ Axios (or other HTTP libraries) - ye bhi wahi karega jo fetch karta hai bas ye thoda jaada developer friendly hai.

→ Promise - ye janab ke andar jo code likhoge wo apna kaam karega aur khud side stack mein chale jaayega uss code ko lekar aur jab andar se code resolve kiya jaayega tab ye chalenge.

1 setTimeout

eg)
```
set Timeout ( function () {
        console. log ("hi");
    }, 1000)
        ↑ in ms
```

eg)
```
~~set~~ console. log ("hey 1");
    console. log ("hey 2");
    set Timeout ( function () {
            console. log ("hey 3");
        }, 2000)
    console. log ("hey 4");
```

o/p will be

```
key 1
key 2
key 4
key 3
```

as key3 is in async functn, to ye side stack mein rahega jab tak main stack pura execute no hojaye.

② setInterval

Eg)

```
setInterval ( function () {
    console.log ("key1");
}, 1000)
```

o/p :- har 1 sec baad key1 console mai print hoga

Then how to stop it

```
const humaraInterval = setInterval (function () {
    console.log ("key 1");
}, 1000)

clearInterval ( humaraInterval)
```

// lekin ye code ka o/p nahi milega as ye turant execute hoga ☹

```
var count = 1;
const humaraInterval = setInterval(function(){
            ++count;
            console.log(count);
            if (count === 4) clearInterval(humara
                                          interval);
        3, 1000)
```

// iss tareke se code ko roka jata hai.

③ Fetch API.

Samajhne keliye API ek url hota hai jo ki data deta hai.

one of best api is random user api

~~# kyuki ye internet par~~

// ye kisi aur url par jaa kar kuch data laayega ya data humaare paas se us url par lekar jaayega.

// Kyuki ye internet par jaayega aur fir data ko lekar aayega to issmein time lagta hai to by default hi js mein fetch ko async banaaya gaya hai. Kyuki fetch ka kaam hai data laana wo bhi kisi url se ab aisa ho sakta hai us url ki website slow ho, to data laane mein time lage. aur agar fetch synchronous hota to uske baad ka code tab tak nahi chalta jab tak uska data nahi aajata, which is a big problem, poora code atak sakta ~~hai~~ tha.

→ fetch('url') ← ye jaayega side stack mein

console.log("hey") ← ye jaayega main stack mein

ye problem create kar sakta hai. so to avoid we can use then method.

→ fetch('url')
 .then(raw() => console.log(raw));

ye data raw data ke format mai hoga jo ki readable nahi hai for ourselves.

raw ek blob hai jo ki byte readable format mai hota hai which only be understandable by computers. so to make human understand- format we will use .json()

```
fetch ('url')
.then (raw => raw.json())
. then (readable => console.log(readable))
```

④ Axious

issmein raw data nahi milta ; direct json data milta hai.

- paste the cdn then write

```
axious . get ('url')
. then ( result => console.log (result))
```

$\underbrace{\quad=>\quad}$ fat arrow functn.

get or post. jyada backend mein use hota hai

⑤ Promises

// promise ke andar koi bhi async code likhdo jo man mein aaye aur promise aapko ek parchi dedeta hai and wo parchi par by default likha hota hai waiting, parchi par do events hote hai mainly ek event ka naam hai then aur ek event ka naam hai catch, agar apka data aagaya to paichi pe resolved likhjaayega waiting ki jagah and then chalega aur agar data mein dikkat aayi to catch chalega and waiting ki jagah rejected likhjaayega.

Syntax

```
new Promise( function ( resolve, reject) {

})
```

Eg)

```
const v = new Promise ( function ( resolve, reject) {
        fetch ('https://randomuser.me/api/')
          .then ( raw => raw.json())
          .then ( result => {
          if ( result.results [0].gender === "male")
                                              resolve();
          else reject();
          });
    });

v.then (function (){
          console.log("hara sultan daba");
    })
```

Callbacks

callback sirf ek function hota hai, bas thoda special jo hai wo ye hai ki ise pass kiya jaata hai as an argument jab particular async code chal jaaye.

eg)
```
function abcd ( a, b) {
        b();
}
abcd( 1, function () { console . log ("callback
                                   chala")});
```
callback functn

o/p :- callback. chala

how to use

callback pahli cheez to ek functn hai, aap iss functn wo sab likhdo jo aapko chalaana ho jab answer aajaye, aur ise tab chalao jab aapka async code chal chuka ho

eg)
```
function doAsync (url, callback) {

        fetch (url)
        .then ( raw => raw. json ())
        . then ( result => {
                callback ()
        })
}
doAsync (`url`, function () {

})
```

Eg) // user se kuch data mangaao and jab data aajaye to us data ke name, gender and email ko print karo.

```
function getData (url, callback) {
    fetch (url)
    .then ( raw => raw.json())
    .then ( result => {
        callback ( result)
    })
}
getData ("https://randomuser.me/api", function(result){
    console.log ( result.results[0].gender,
        result.results[0].email, ~~result~~
    result.results[0].name.first)
})
```

✓ Async / Await

Koi bhi functn banaalo and uske andar jo man mein aaye wo async code likhdo, ab jab async likhte ho to baad wali line pahle chal jaati hai kyuki async side stack pe hote hai aur baad waali line agar async ke basis par hui to apka code fail ho jaayega, wo isliye kyuki apka code depend karta hai async code par jo ki baad mein chalega sync code chalne ke baad.

// with async await aap async code bhi aise likh skte ho jaise ki aap normal synchronous code likh rahe ho.

// await ko work karane ke liye uske parent par async likhna padega.

Eg) async function abcd () {

let a = await fetch (`https://randomuser/api/`);

// uss url ke data fetch karne ke baad hi agli line chalega.

a = await a.json();

// yahan par firse likhna pada warna to ye line pehele chal jaati. ~~aur~~

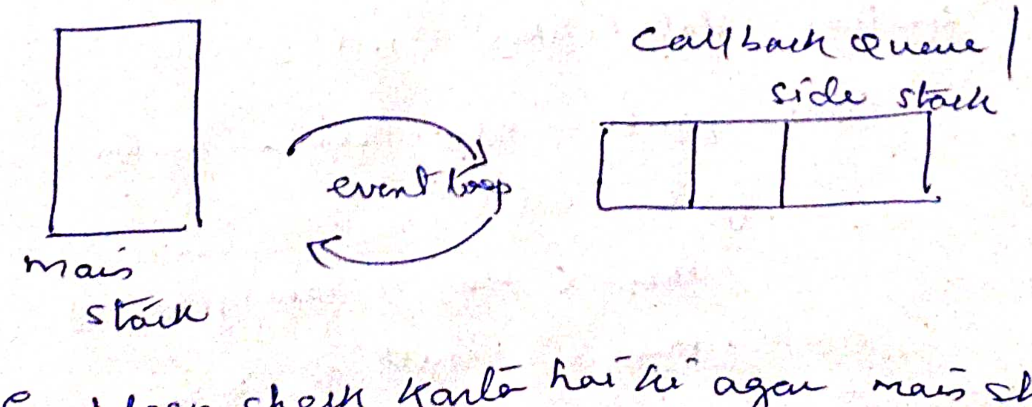
console.log (a);

}

abcd ();

AJAX — = Fetch & Axious

Event loop



Event loop check karta hai ki agar main stack khali ho tab side stack se code ko main stack par lekar ayega.

Callbacks vs Promises vs Async/Await

Q) ek url se data lekar aao and usey console par show karo.

Callback.

```
function datafetcher ( url, callback ) {
        fetch ( url)
        .then ( raw => raw. json())
        . then ( result => {
              callback ( result );
        })
  }

data Fetcher (" https://randomuser..", function (result)
    {
        console.log(result);
  })
```

promises

```
function dataFetcher (url) {
    const parchi = new Promise (function (resolve, reject) {
        fetch (url)
          . then ( raw => raw.json ())
          . then ( result => {
                resolve (result );
          })
  })
```

```
    return parchi ;
  }

var a = dataFetcher ("https :// randomuser.me/api/")
      .then ( function (result) {
            console. log (result) ;
        }
```

async/await

```
async function dataFetcher (url) {
    let data = await fetch (url);
    let result = await data.json();
return result;
 }

async function hh () {
   const data = await dataFetcher (`https://...`);
        console.log(data);
  }
  hh ();
```

hum async functn ko direct print nahi kar sakte hain so ek functn use karenge & then uss data ko print karenge.