* Refactoring old code
* Pending promises [illegible] settle [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] while [illegible] & [illegible] reject promise [illegible] [illegible] [illegible] state [illegible] pending after [illegible] complete

JS 12

Async functions

- Async fun^n :

```
async function greet(){
        return "hello world";
}
let hello = async() => {}; return promise
```

```
async function greet(){
    return "hello"
}
greet();
Promise {
```

- fun^n
  - ↙ normal execute ↓ return promise → state → fulfilled. + return
  - ↘ error ↓ return promise ↓ rejected state.

For more understanding study async.html.

```
async function greet (){
      abc.abc()
      return ("hello")
}
//Error abc not defined.
        OR
async fun^n greet (){
   throw "some random error"
   return "hello";
}
greet()
=> Promise {
```

Await (only in async fun^n)

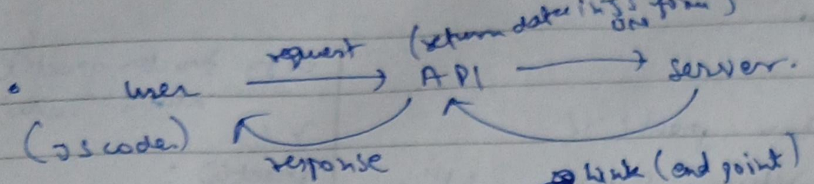
⇒ pause the execution of its surrounding async fun^n until the promise is settled (resolved or rejected)

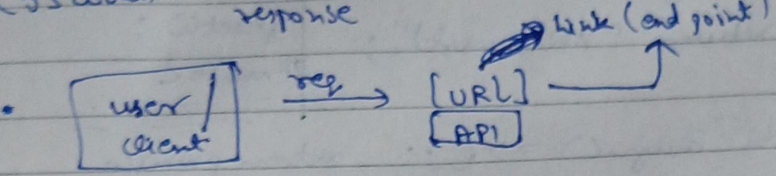
```
→ function getNum(){
    return new promise ((resolve, reject) =>{
        setTimeout(() => {
            let num = math.floor (math.random()*10) +1
            console.log(num);
            resolve();
        },1000);
    });
function demo() {     //async bnaya in order to use await)
    Num ()            // इन तीनों से problem है कि [illegible]
    Num ()            1 second में execute
    Num
```

// In order to do one item after 1 second another
await getNum();
" " ;
" " ;
;

* Handling rejections in await.
use try and catch.

* API → Application programming interface.





* Accessing some APIs.
⇒ Some random → (i) https://catfact.ninja/fact.
(ii) www.boredapi.com/api/activity.

→ JSON data made to be read by computer.
(iii) https://dog.ceo/api/breeds/image/random

JSON

(Javascript object Notation www.json.org.)
- JSON → contain key value pair.
  ⇒ All keys are string.

```
Let obj = {
    "string" : n;
    "a"      : undefined     // Error. [In JS it will work]
}
```

// json validator // konsa json ka code sahi konsa galat.

Accessing

Data from JSON

- JSON.parse(data) (method to parse a string data into JS object)
- JSON.stringify([illegible]) method to parse a JS object [illegible] JS[illegible]

```
> let jsonRes =
> let validRes = JSON.parse(jsonRes);
> console.log(validRes.fact)
> let student = {
      name: "shradha",
      class: 5,
  };
  console.log(JSON.stringify(student));
  // convert student into json.
```

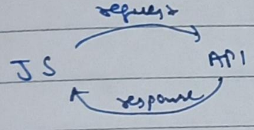
Testing API request.

Tools

- hoppscotch (we use it)
- Postman

Ajax (request & response calls)

→ asynchronous javascript and XML



HTTP verbs

Ex: • get • post • Delete.

Status codes

- Ex: 200 — OK
- 400 — Bad req.
- 404 — Not found
- 500 — Internal server error.

& many more are there.

Adding information in URL's

→ https://www.google.com/search?q=mango
search for mango.
→ harry + potter. (harry potter)

→ name = shradha & marks = 95 ;
key ↗ (name), value ↑ (shradha)

> In harry potter :-

https://api.potterdb.com

// for accessing movies :-

https://api.potterdb.com/v1/movies

// If we want to access single movie :-

https://api.potterdb.com/v1/movies/"id of that movie"

(As it is written like "/v1/characters/:id",

:id is the thing which specifies what is written)

* headers :-

Accept	Application/json	to get data in JSON format

* Our first Request

```
• let url = "https://catfact.ninja/fact";
fetch(url)
    .then((res) => {
        console.log(res);
    })
    .catch((err) => {
        console.log("error -", err);
})
```

// for accessing data :-

```
→ res.json().then((data) => {
        console.log(data);
});
```

(OR)

```
→ return res.json();
.then( (data) => {
        console.log(data);
})
```

// console.log(data.fact) // for printing fact of data only

// for again calling.

```
.then ((data) => {
        console.log(data)
        return fetch(url);
})
.then((res) => {
        return res.json();
})
.then ((data2) => {
        console.log(data2)
})
```

fetch with async and await

```
let url =

async function getfacts() {
        try {
        let res = await fetch(url);
        let data = await res.json();
        console.log(res)
        }
        catch(e) {
        console.log(e)
        }
}
```

undefined

// let res = ~~await~~ fetch(url)
// we get undefined ~~response~~ using above.

Axios

=> Library to make HTTP request
=> CDN copy to body.
=>

```
async fun getfacts() {
        try {
            let res = await axios.get(url) // return promise
            console.log(res).
        }
        catch(e) {
            console.log(e)
        }
}
```

// fetch से json से data को convert करना पड़ता but this is not done in axios.
(As fetch से readable stream data है, is से data)
// response.data (to get data)
// " ". fact (to get fact present in data)

```
btn.addEventListener("click", async () => {
    let fact = await getFacts();
    console.log(fact)
});
```

// Hyr async & await use nhi kia to promise return hoga.

* Sending headers using axios.

```
let url = "https://dog.ceo/api/breeds/image/random";

async function getDogs() {
    try {
        const config = { headers: { Accept: "application/json" } };
        let res = await axios.get(url);
        console.log(res) // promise
        // (res.data) // in HTML format.
        // for JSON
    } catch(e) {
        console.log(e)
    }
}
```

TERMINAL