<!-- ! fetch -->

<!--? Synchronous -->
Synchronous means the code runs in a particular sequence of instructions given in the program.
Each instruction waits for the previous instruction to complete its execution.

<!--? Asynchronous -->
Due to synchronous programming, sometimes imp instructions get
blocked due to some previous instructions, which causes a delay in the UI.
Asynchronous code execution allows to execute next instructions
immediately and doesn't block the flow

1. What is the fetch() API?

- The fetch() API is a modern and powerful way to make asynchronous HTTP requests to servers in JavaScript.
- It returns a **Promise** that resolves to the Response object representing the response to the request.

2. Syntax

```
fetch(url, options)
    .then(response => {
        // Handle the response
    })
    .catch(error => {
        // Handle any errors
    });
```

- **url**: The URL to which the request is sent.
- **options**: An optional object containing custom settings for the request, such as method, headers, body, etc.

3. Handling the Response

- **response.json()**: Converts the response body into JSON. This also returns a Promise.

4. Example Explained

```
let fetchedData = fetch("https://api.github.com/users");
console.log(fetchedData); // Logs the Promise object

fetchedData.then((data) => {
    // First .then block to handle the response
    let jsonData = data.json();  // Convert response to JSON
    console.log(jsonData);  // Logs the Promise of the parsed JSON

    jsonData.then((finalData) => {
        // Second .then block to handle the parsed JSON data
        console.log(finalData);  // Logs the final parsed data

        finalData.map((e) => {
            console.log(e.login);  // Logs the 'login' property of each user
        });
    });
});
```

5. Chaining Promises

- The `fetch()` method returns a Promise. After calling `fetch()`, you can chain multiple `.then()` methods to handle the response and data.
- **Nested Promises**: In the example, the first `.then()` handles the `Response` object and parses it to JSON, while the second `.then()` handles the actual data.

```
// !   taking all the data from api and print on the ui


let fetchedData1 =   fetch("https://api.github.com/users")

let container = document.querySelector(".container")

fetchedData1.then((data)=>{

    let jsonData = data.json()

    jsonData.then((finalData)=>{

    finalData.map((ele)=>{

        let li = document.createElement("li")
        li.innerText = ele.login;

        container.append(li)
    })
    }).catch((err)=>{
        console.log(err)
    })

}).catch((err)=>{
    console.log(err)
})
```