JavaScript - Day 13

Page No.
Date

ASYNC & AWAIT (ES8)

→ It is an extension to promise itself.

→ Async also returns a promise.

→ So it's basically syntactic sugar, does the same thing as promises but code looks much more redable.

// with promises [following do the same thing].

```
performTask(task1)
    .then(() => performTask(task2)
    .then(() => performTask(task3)
    .then(() => performTask(task4)
```

// async await

do some async task — Inside this function

```
async function playGame() {
    const task1 = await performTask(task1)
    await performTask(task2);
    await performTask(task3);
    await performTask(task4);
}
```

→ await keyword basically indicates were waiting for someone to receive something to give us a response.

→ Instead of chaining with .then() we're awaiting for every task to perform before next one.

* **Practical Example :-**

→ The fetch() methods that let's you make an API call, is actually a promise, so we can do the following,

```
fetch('http://jsonoutput/users')
    .then(response => response.json())
    .then(console.log(response));
```

→ Because fetch is a promise we'are able to do .then() on it.

* **NOTE :** response.json(), this json() to convert to json format is also a promise, hence again a .then() after it.

• Now, Let's convert this to async await.

```
async function fetchStudents() {
    const response = await fetch('http://....')
    const data = await response.json()
    console.log(data);
}
```

* **Error handling in async await.**

```
async function fetchStudent() {
    try {
        await fetch(URL)
        :
}
```

```
    catch (error) {
        console.log(error);
    }
}
```

→ Error handling with async await can be achived using try catch block.

→ put your fetch call (await) inside try and it any error occurs within try then it will be caught in catch block.