academeez

async-await

Dealing with async code the easy way!

async-await and promises

- async functions are functions we can use to help us deal with promises in a really easy and clear way.
- In async code you write you write your code in a very sync structure

```
async function fetchFromServer() {
    const res = await fetch('https://some-server-url');
    const dataFromServer = await res.json();
    return dataFromServer
}

fetchFromServer().then((dataFromServer) => {
    // we get the data wrapped in a promise
})
```

Prerequisite - promises

- Since async-await is a tool used with promises, it is highly recommended to view the Promise lesson before going over this lesson

async - await

- async function is a special kind of function.
- This function always returns a promise
- The function has multiple entrance and exit points marked with **await**
- await is placed before a promise
- The function will exit until the promise will be resolved and than re-enter the function where we can choose to place the value in the promise and assign it to a variable

async - await - multiple exits entrances

academeez

- In this example the async function will exit two times before each promise
- It will enter again after those promise are resolved

```
async function fetchFromServer() {
    //entering the function
    // ...

    // exit the function until fetch is resolved
    const res = await fetch('https://some-server-url');

    // exit the function until json() is resolved
    const dataFromServer = await res.json();

    // return the data wrapped as promise
    return dataFromServer
}
```

Dealing with errors

academeez

- Our async code can also fail.
- With async function we can use try..catch for dealing with async errors

```
async function fetchFromServer() {
    try {
        const res = await fetch('https://some-server-url');
        const dataFromServer = await res.json();
        return dataFromServer
    } catch(err) {
        // if the call for the server fails
        // fallback to empty array response
        console.log(err.message);
        return [];
    }
}
```

Running Promises in parallel

- With await we can “wait” for a promise to be fulfilled, but what if we want to run promises in parallel and wait for multiple promises running together?

```
async function multipleExample() {
    const [promise1Value, promise2Value] = await Promise.all(
        [
            promise1,
            promise2
        ]
    )
}
```

Summary

academeez

- async - await functions allow us to deal with promises in a special functions where we can use await before each promise
- This allows us to write async code in a more sync way

academeez

Thank You

Next Lesson: Modules