Async/Await

CS445 Modern Asynchronous Programming

Maharishi University of Management
Department of Computer Science

Teaching Faculty: Assistant Professor Umur Inan

Prepared by: Associate Professor Asaad Saad

Async/Await

Async/Await is just syntactic sugar for `.then()` calls on a promise. It's a shorthand that lets you write async code with fewer lines of code.

The **`async`** function declaration defines an asynchronous function, which operates asynchronously via the event loop, using an implicit Promise to return its result.

The **`await`** keyword is only valid inside **`async`** functions

If the awaited expression isn’t a promise, it’s wrapped into a promise.

Await

`await` puts the code of its scope into an invisible **`.then()`** handler, it allows you to save the writing of the **`.then()`** handler and gives the code a synchronous look.

```
myPromise().then(() => {
    console.log("hello");
});

await myPromise();
console.log("hello");
```

Example

```
async function myFunction() {
    const result = await new Promise(resolve => resolve(true));
    console.log(result);
}
myFunction();
console.log(`Finish`);
```

Keep it Safe

```
async function myFunction() {
    try{
        const result = await new Promise((resolve,reject) => reject(false));
        console.log(result);
    }catch(error){
       console.log(error);
    }
}
myFunction();
console.log(`Finish`);
```

Since a promise can reject, we better have a try/catch around **`await`** to catch and handle that rejection.

Example - Promise

```
const phoneCall = function(){
   return new Promise(resolve=>resolve(true))
};


const orderPizza = function () {
    phoneCall()
        .then(console.log)
        .catch(console.log);
};

orderPizza();
console.log('Finish homework')



// Finish homework
// true
```

Example – Async & Await

1

2

3

```
async function orderPizza() {
    try {
        console.log('Before making the call');
        let results = await phoneCall();
        console.log(results);
        console.log('After making the call');
    } catch (error) {
        console.log(error);
    }
}

orderPizza();
console.log('Finish homework')
```

```
Before making the call
Finish homework
true
After making the call
```

```
const phoneCall = function(){
    return new Promise(resolve=>resolve(true))
};
```

Example

```
console.log(`Start`);
async function myFunction() {
    console.log(`A`)
    await fakePromise()
    console.log(`B`)
}
myFunction()
console.log(`End`);

function fakePromise() {
    console.log(`Fake`)
}
```

If the awaited expression isn’t a promise, it’s wrapped into a promise.

Example

```
console.log(`Start`);
async function myFunction() {
    console.log(`A`)
    await realPromise()
    console.log(`B`)
}
myFunction()
console.log(`End`);

function realPromise() {
    return new Promise(resolve => {
        console.log(`C`)
        resolve(`Real`)
        console.log(`D`)
    })
}
```