Async/Await in Javascript | JavaScript Tutorial In Hindi #43

In this tutorial, we will learn about what JavaScript Async/Await keywords do with examples. Promises provide us an easier way to deal with asynchronous programming. If you have not watched the tutorial on promises, then check ***tutorial#39***. Now let us move towards our main topic. **Async/await functions**, which is a new addition with ES2017, that help us even more in allowing us to write completely synchronous-looking code while performing asynchronous tasks behind the scenes.
The functionality we can achieve using async functions can also be achieved by combining promises with generators, but async functions give us what we need without any extra complex code. As the async/await keywords were introduced in the newer version of JavaScript (ES8). Some older browsers may not support the use of async/await.

The Async Keyword:-

We use the async keyword with a function to represent that the function is asynchronous. The async function always returns a promise.
Here is the syntax of async function is:

```
async function name(param1, param2, ...paramN) {
    // statements
}
```

- **name** - This is the name of the function
- **param** – This is the parameters that are passed to the function

Example: Async function

```
// async function example
```

```
async function func() {
    console.log('Async/Await tutorial.');
    return Promise.resolve(1);
}
func();
```

In the above example, the async keyword is used to represent that the function is asynchronous. Since this func() function returns a promise, we can use the chaining method then() like this:

```
async function func() {
    console.log('Async/Await tutorial.');
    return Promise.resolve(1);
}
func().then(function(res) {
    console.log(res)
});
```

Output:-

Async/Await tutorial.
1

The await keyword:-

The await keyword is used to wait for the asynchronous operation. This keyword is used inside the async function. Here is the syntax to use await is:

```
let result = await promise;
```

The await pauses the async function until the promise returns a result value.
When we want to call this function, we prepend await, and the calling code will stop until the promise is resolved or rejected. Here is another example:

```
async function func1() {
let promise = new Promise((resolve, reject) => {
setTimeout(() => resolve("Done with Async/Await!"), 1000)
});
let result = await promise; // wait until the promise resolves
console.log(result);
}
f();
```

Benefits of using an async function:-

- Debugging using promises sometimes is very hard because the debugger will not step over the asynchronous code. But the Async/await makes this very easy because it is just like synchronous code to the compiler.
- As we can see from the example above, the code looks very simple compared to the code using plain promises, with chaining and callback functions.
- Error handling is simpler in async functions.

So, in this tutorial, we have learned about async and await keywords. Async/await provides a nice, simplified way to write async code that is easy to read and maintain.

Here is a quick review of what we have studied in this lecture:

- An async function is a function that is declared with the asynckeyword. Async functions are the instances of AsyncFunction constructor, and the await keyword is permitted within them.
- The async and await keywords enable the asynchronous, promise-based behavior to be written cleaner.

Code file tut43.js as described in the video

Copy

```
console.log("This is tutorial 43");

async function harry(){
    console.log('Inside harry function');
    const response = await fetch('https://api.github.com/users');
    console.log('before response');
    const users = await response.json();
    console.log('users resolved')
    return users;

    // return "harry";
}

console.log("Before calling harry")
let a = harry();
console.log("After calling harry")
console.log(a);
a.then(data => console.log(data))
console.log("Last line of this js file")
```

Previous Next

CodeWithHarry

Copyright © 2022 CodeWithHarry.com