

# Async-Await

JAVASCRIPT

JS

Hey Everyone 🙌

JavaScript is everywhere. Millions of webpages are built on JS.

A few examples will help you understand the JavaScript Async await Keywords in this post.

Do Like, save and Share This Post If You Found This Helpful.

# async-await

We use the **async** keyword with a function to represent that the function is an **asynchronous** function.

The **syntax of async** function is:

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

Here,

- **codeclash** - name of the function
- **param** - parameters that are passed to the function

# Async Function

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

Since this function **returns a promise**, you can use the chaining method **then()** like this:

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

func().then(function(result) {
  console.log(result)
});
// Output
// Async function
// 1
```

# Await Keyword

The `await` keyword is used inside the `async` function to wait for the `asynchronous` operation.

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

The use of `await` pauses the `async` function until the promise `returns a result` value.

```
// a promise
let promise = new Promise(function (resolve, reject) {
  setTimeout(function () {
    resolve('Promise resolved'), 4000);
});
// async function
async function asyncFunc() {
  // wait until the promise resolves
  let result = await promise;
  console.log(result);
  console.log('hello');
}
// calling the async function
asyncFunc();

// Output
// Promise resolved
// hello
```

- In the above program, a **Promise** object is created and it gets resolved after **4000 milliseconds**.
  - Here, the **asyncFunc()** function is written using the **async** function.
  - Hence, **hello** is displayed only after **promise value** is available to the **result** variable.
  - The **await keyword** waits for the promise to be complete.

```
let promise = new Promise(function (resolve, reject) { ←
    setTimeout(function () {
        resolve('Promise resolved');
    }, 4000);
});

async function asyncFunc() { →
    let result = await promise; ←
    console.log(result);
    console.log('hello');
}

asyncFunc();
```

calling function

waits for promise to complete

# Error Handling

While using the `async` function, you write the code in **asynchronous manner**.

And you can also use the `catch()` method to **catch the error**.

```
asyncFunc().catch(  
    // catch error and do something  
)
```

The other way you can handle an error is by using **try/catch** block.

```
// a promise  
let promise = new Promise(...);  
// async function  
async function asyncFunc() {  
    try { // wait until the promise resolves  
        let result = await promise;  
        console.log(result);  
    }  
    catch(error) {  
        console.log(error);  
    }  
}  
// calling the async function  
asyncFunc(); // Promise resolved
```

# Benefits Of Using Async Function

- The code is more readable than using a callback or a promise.
- Error handling is simpler.
- Debugging is easier.

@CODE.CLASH



**Check Out My Profile For  
Such Excellent Posts On  
Web Development.**



IMTIYAZ NANDASANIYA