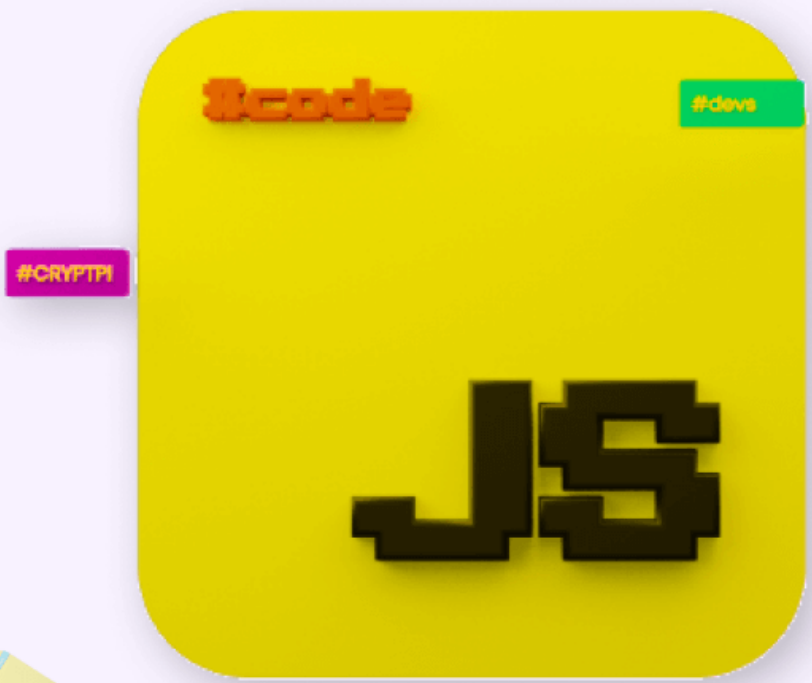
@CODE.CLASH

Javscript async/await



NEXT

Hey Everyones 👋

In this post, you will learn about JavaScript async/await keywords with the help of examples.

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

NEXT →

async/await

- We use the async keyword with a function to represent that the function is an asynchronous function.
- The async function returns a promise.

```
async function name(parameter1, parameter2, ...paramaterN) {
    // statements
}
```

- **name** - name of the function.
- **parameters** - parameters that are passed to the function.

NEXT →

Async Function

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

- Since this function returns a promise, you can use the chaining method then().

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

NEXT →

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.

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

Output

```
Promise resolved
hello
```

NEXT →

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

```
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();
```

waits for promise to complete

calling function

NEXT →

Error Handling

- While using the async function, you write the code in a synchronous 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(function (resolve, reject) {
    setTimeout(function () {
    resolve('Promise resolved')}, 4000);
});
// 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
```

NEXT →

Benefits of Using async Function

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

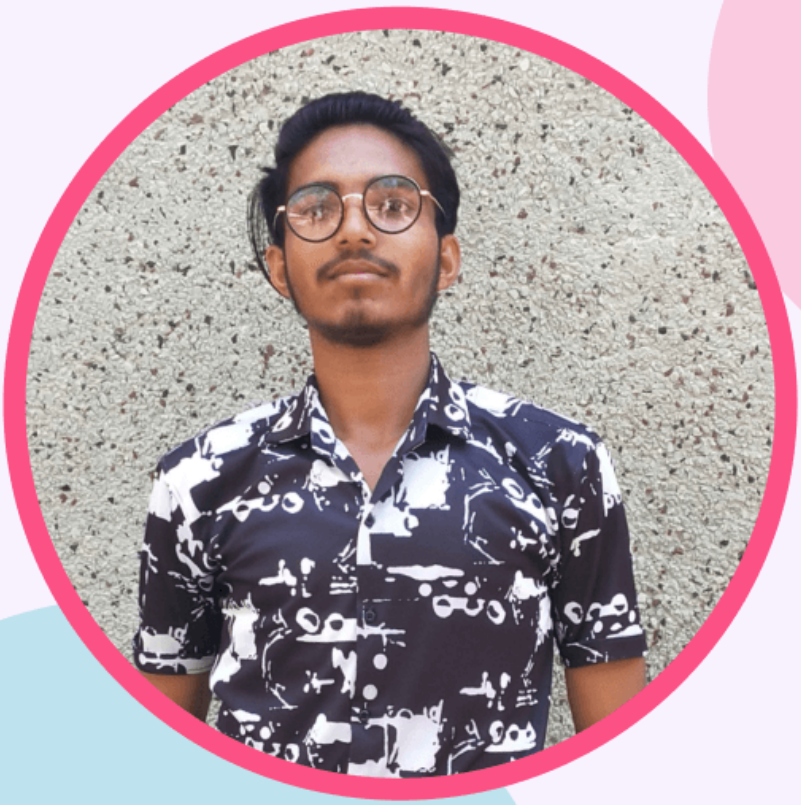
Note: You can use `await` **only inside of** `async` **functions.**

Best Of Luck :)

NEXT →

@CODE.CLASH

THANKS FOR YOUR ATTENTION



CODE.CLASH

IMTIYAZ NANDASANIYA

LIKE AND SAVE IT FOR LATER