Frontend Developer

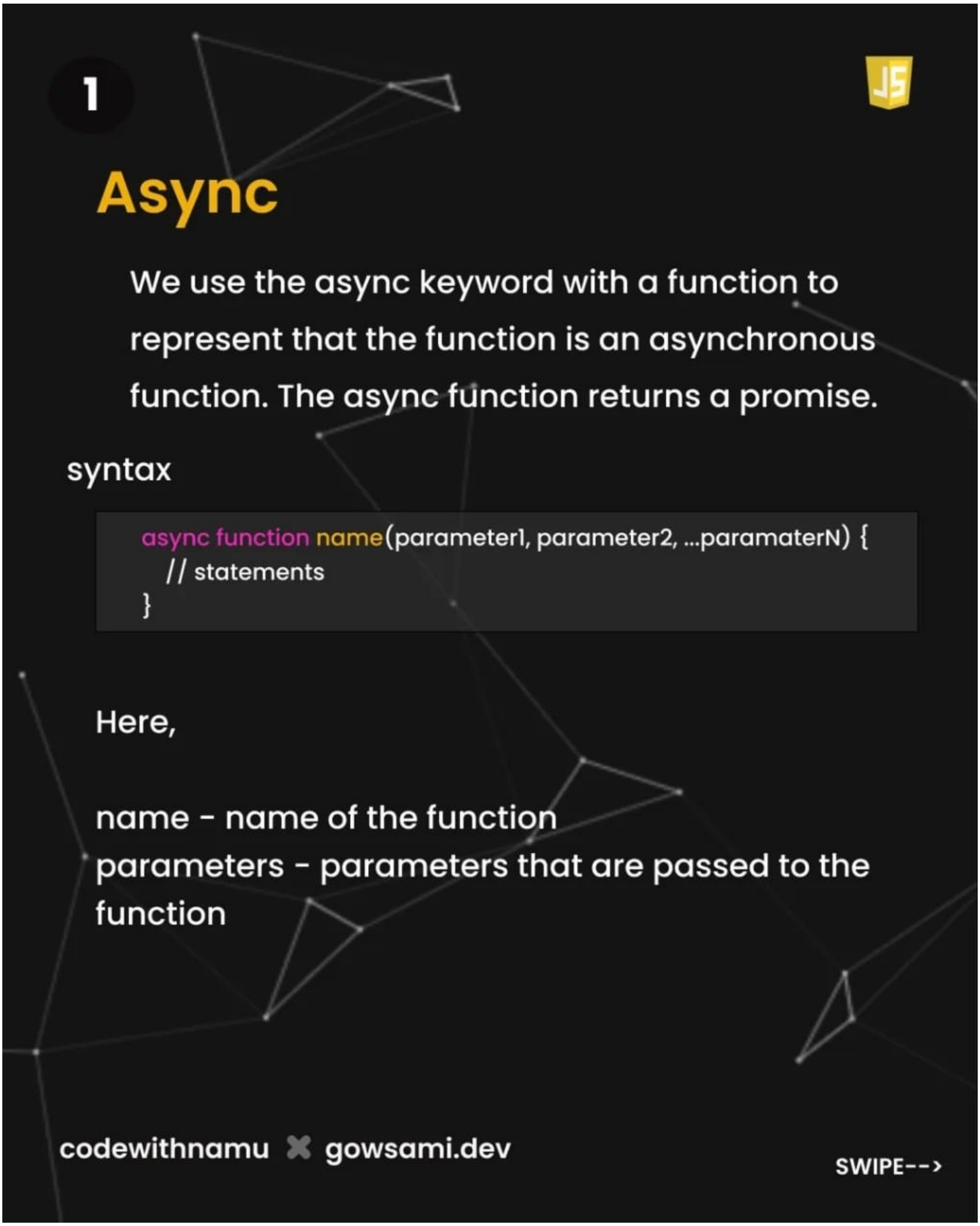
JS

Async/Await

In Javascript

codewithnamu ✖ gowsami.dev

SWIPE-->



1

Async

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

syntax

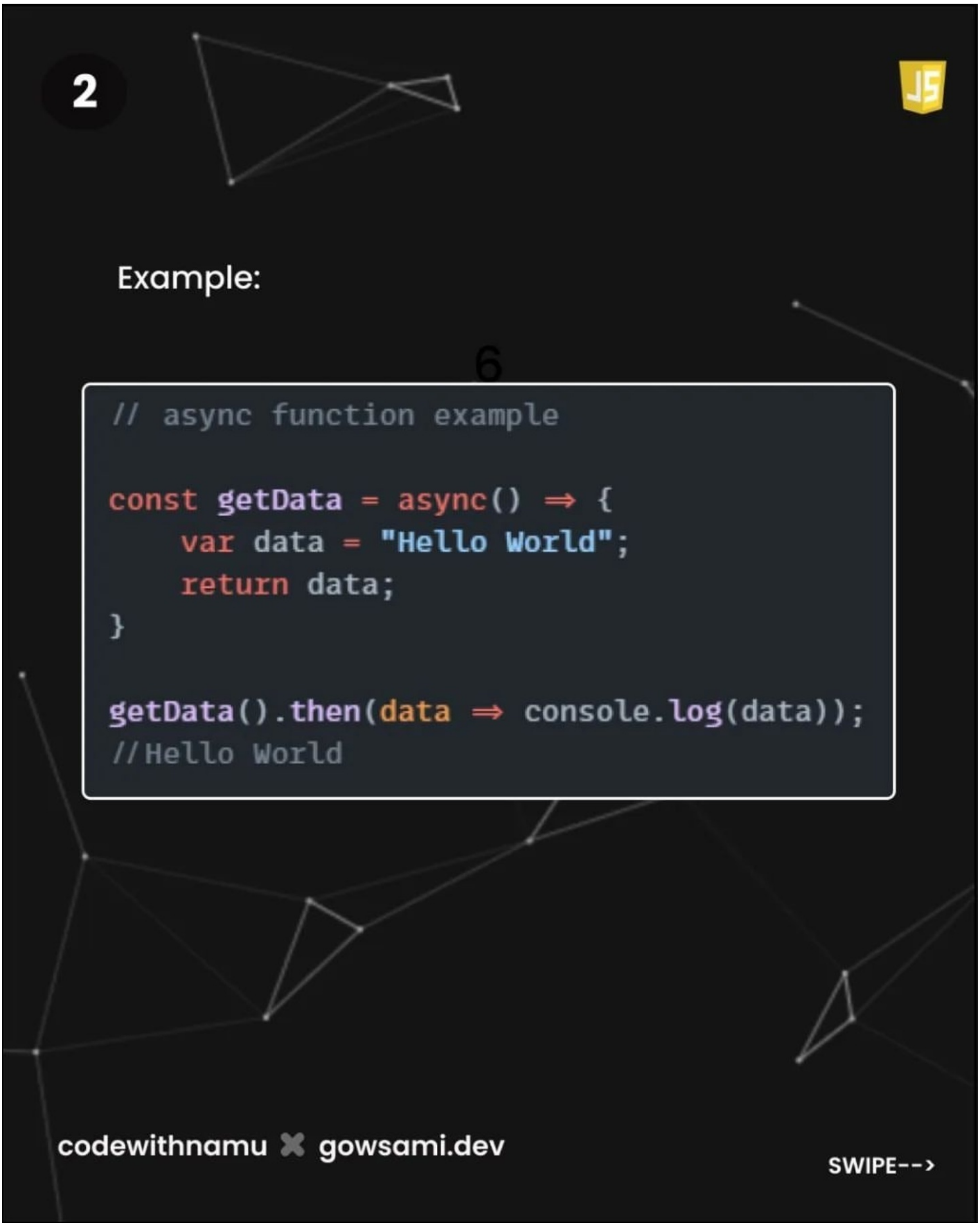
```
async function name(parameter1, parameter2, ...paramaterN) {
    // statements
}
```

Here,

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

codewithnamu ✖ gowsami.dev

SWIPE-->



2

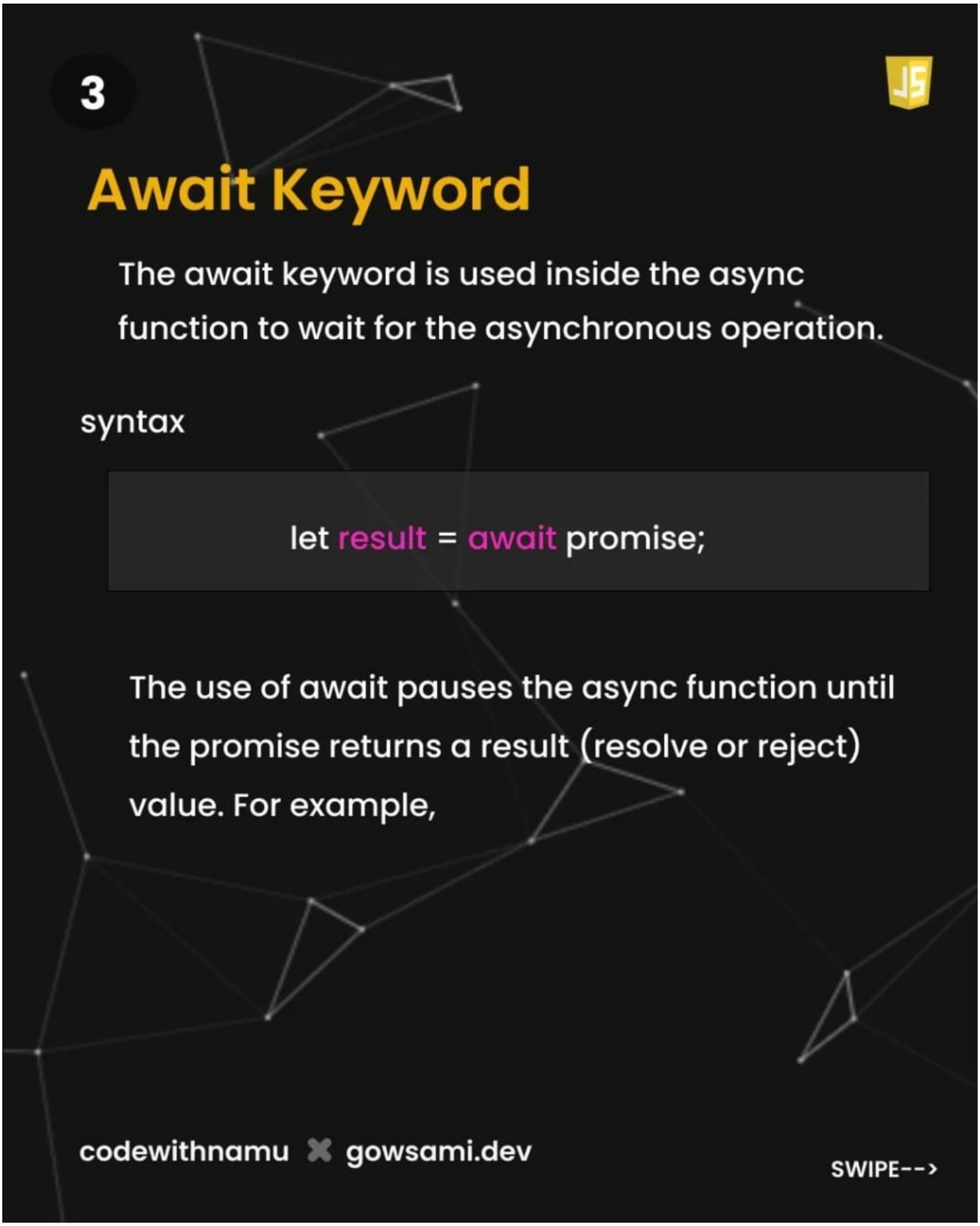
Example:

6

```
// async function example

const getData = async() => {
    var data = "Hello World";
    return data;
}

getData().then(data => console.log(data));
//Hello World
```

codewithnamu ✖ gowsami.dev

SWIPE-->



3

Await Keyword

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

syntax

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

The use of await pauses the async function until the promise returns a result (resolve or reject) value. For example,

codewithnamu ✖ gowsami.dev

SWIPE-->

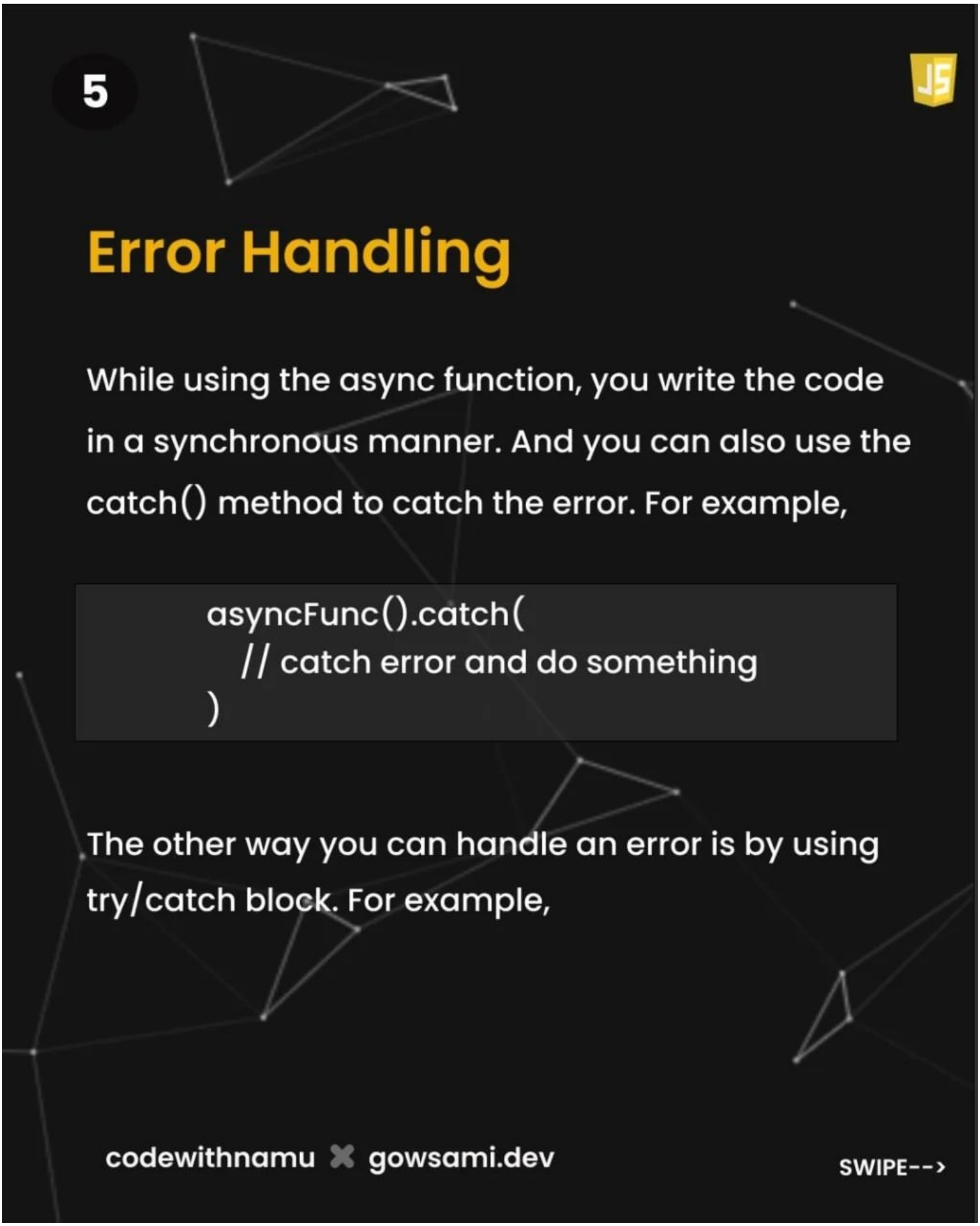
4

Example:

```
// 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();
```

codewithnamu ✖ gowsami.dev

SWIPE-->



5

Error Handling

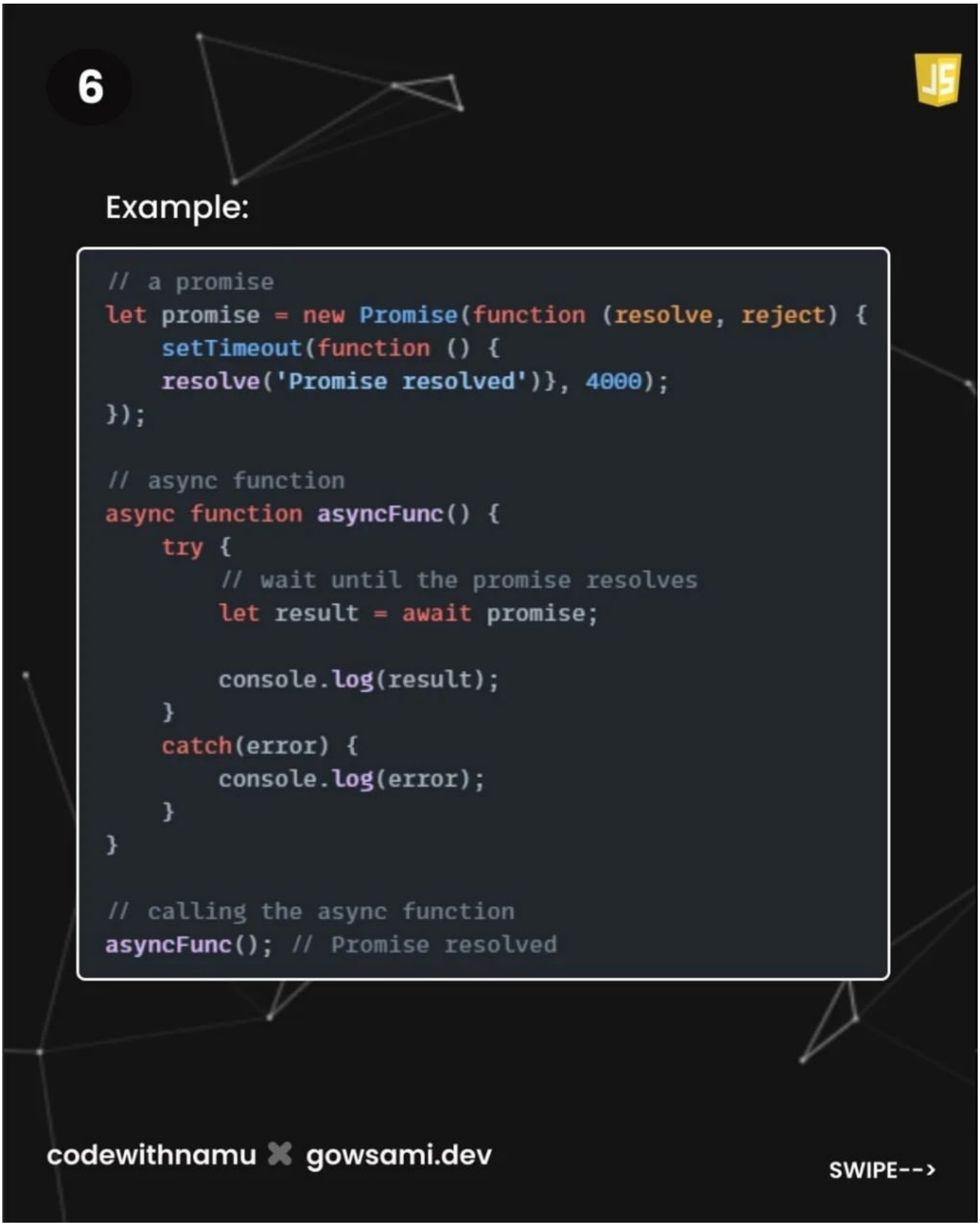
While using the async function, you write the code in a synchronous manner. And you can also use the catch() method to catch the error. For example,

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

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

codewithnamu ✖ gowsami.dev

SWIPE-->



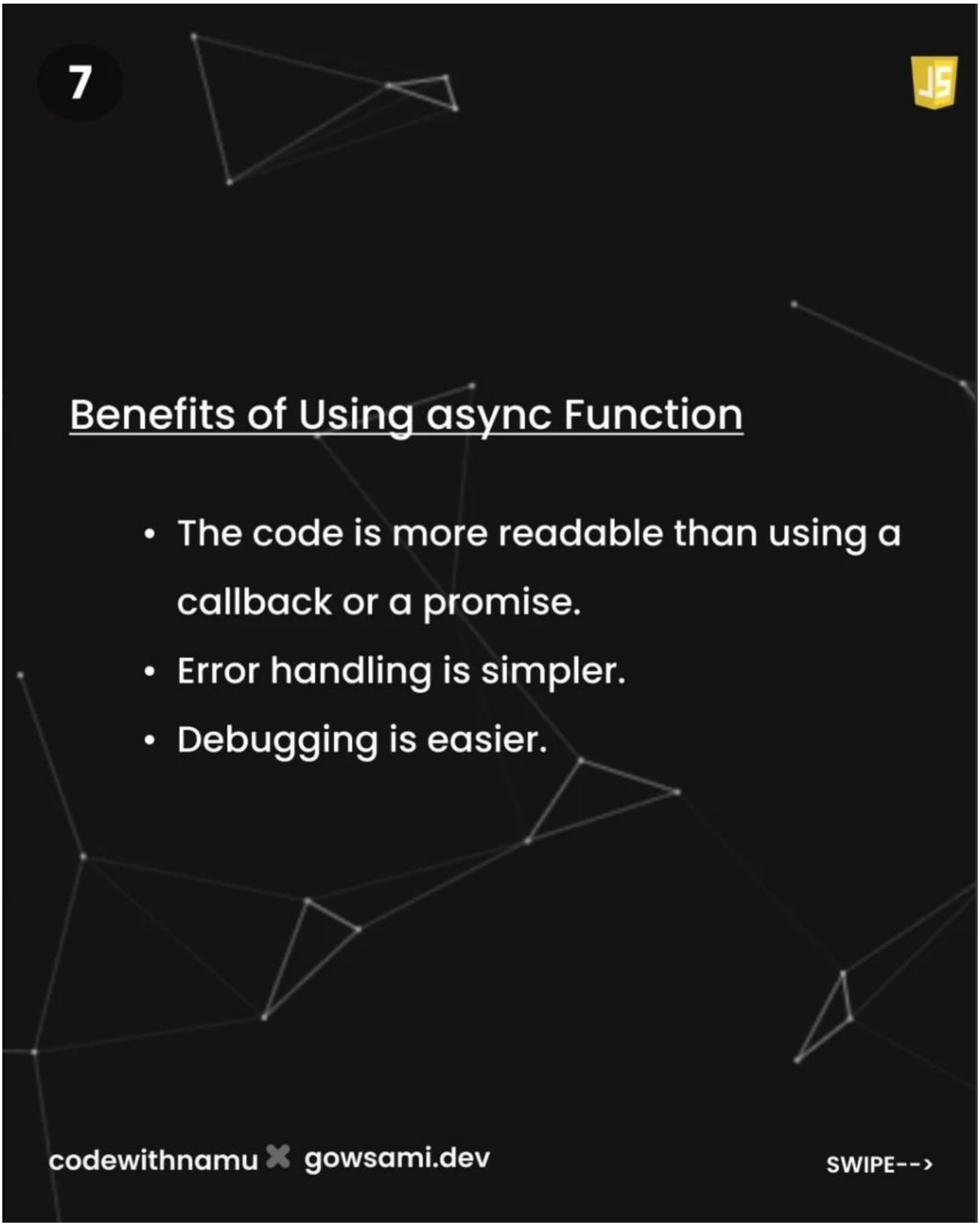
6

Example:

```
// 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
```

codewithnamu ✖ gowsami.dev

SWIPE-->



7

JS

Benefits of Using async Function

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

codewithnamu ✖ gowsami.dev

SWIPE-->