JavaScript Async-Await

*"async and await make promises easier to write"*

**async** makes a function return a Promise

**await** makes a function wait for a Promise

**async function**

An async function is a function declared with the async keyword, and the await keyword is permitted within it. The async and await keywords enable asynchronous, promise-based behavior to be written in a cleaner style, avoiding the need to explicitly configure promise chains.

**Syntax**

asyncfunctionname(param0)

{

statements  
}

asyncfunctionname(param0,param1)

{

statements  
}

asyncfunctionname(param0,param1,/\* … ,\*/paramN)

{

statements  
}

**Parameters**

name

The function's name.

param Optional

The name of an argument to be passed to the function.

statements Optional

The statements comprising the body of the function. The await mechanism may be used.

**Return value**

A Promise which will be resolved with the value returned by the async function, or rejected with an exception thrown from, or uncaught within, the async function.

**Description**

Async functions can contain zero or more await expressions. Await expressions make promise-returning functions behave as though they're synchronous by suspending execution until the returned promise is fulfilled or rejected. The resolved value of the promise is treated as the return value of the await expression. Use of async and await enables the use of ordinary try / catch blocks around asynchronous code.

**Example**

const getData = async() => {

**var** data = "Hello World";

**return** data;

}

getData().then(data => console.log(data));

**Output:**

Hello World

**Await:** Await function is used to wait for the promise. It could be used within the async block only. It makes the code wait until the promise returns a result. It only makes the async block wait.

**Example:**

const getData = async() => {

**var** y = await "Hello World";

    console.log(y);

}

console.log(1);

getData();

console.log(2);

**Output:**

1  
2  
Hello World

Notice that the console prints 2 before the **“Hello World”**. This is due to the usage of the await keyword.

Browser Support

ECMAScript 2017 introduced the JavaScript keywords async and await.

The following table defines the first browser version with full support for both:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Chrome | Edge | Firefox | Safari | Opera |
| Dec, 2016 | Apr, 2017 | Mar, 2017 | Sep, 2017 | Dec, 2016 |