



# Async/Await

O `async/await` facilita a **interação com chamadas assíncronas, aguardando o retorno de uma determinada promise**

The screenshot shows a Mac OS X desktop environment with a terminal window open. The terminal window has a title bar "async\_await\_1.js — javascriptmasterclass". The main pane displays the code for "async\_await\_1.js", which contains several functions for handling promises and asynchronous operations using the `yield` keyword. The code includes a `sum` function that returns a promise, an `async` wrapper function, an `asyncR` helper function, and an `async` generator function that performs two additions and logs the result. The right side of the terminal window shows the command being run in the terminal: "node a sync\_await/async\_await\_1.js", followed by the output "12".

```
JS async_await_1.js x
async_await_1.js — javascriptmasterclass
1  function sum(a, b) {
2      return new Promise(function(resolve) {
3          setTimeout(function() {
4              resolve(a + b);
5          }, 1000);
6      });
7  }
8  function async(fn) {
9      const gen = fn();
10     asyncR(gen);
11 }
12 function asyncR(gen, result) {
13     const obj = gen.next(result);
14     if (obj.done) return;
15     obj.value.then(function(result) {
16         asyncR(gen, result);
17     });
18 }
19 async(function* () {
20     const a = yield sum(2, 2);
21     const b = yield sum(4, 4);
22     const result = yield sum(a, b);
23     console.log(result);
24 });
25

TERMINAL ... 1: bash +
rodrigobranas:javascriptmasterclass $ node a sync_await/async_await_1.js
12
rodrigobranas:javascriptmasterclass $
```

async\_await\_2.js — javascriptmasterclass

JS async\_await\_2.js ×

```
1  function sum(a, b) {
2      return new Promise(function(resolve) {
3          setTimeout(function() {
4              resolve(a + b);
5          }, 1000);
6      });
7 }
8
9 (async function () {
10    const a = await sum(2, 2);
11    const b = await sum(4, 4);
12    const result = await sum(a, b);
13    console.log(result);
14 })();
15
```

TERMINAL ... 1: bash

```
rodrigobranas:javascriptmasterclass $ node a
sync_await/async_await_2.js
12
rodrigobranas:javascriptmasterclass $
```

The screenshot shows a Mac OS X desktop environment with a terminal window open. The terminal window has a title bar "async\_await\_3.js — javascriptmasterclass". The main pane contains the following code:

```
JS async(await_3.js x
1  function sum(a, b) {
2      return new Promise(function(resolve, reject) {
3          if (!a || !b) return reject("Invalid input");
4          setTimeout(function() {
5              resolve(a + b);
6          }, 1000);
7      });
8 }
9
10(async function () {
11    const a = await sum(2);
12    const b = await sum(4, 4);
13    const result = await sum(a, b);
14    console.log(result);
15})();
```

To the right of the code editor is a terminal window titled "1: bash". It displays the following output:

```
rodrigobranas:javascriptmasterclass $ node async(await/async(await_3.js
(node:179) UnhandledPromiseRejectionWarning: Unhandled promise rejection (reject
ion id: 2): Invalid input
(node:179) [DEP0018] DeprecationWarning:
  Unhandled promise rejections are deprec
ated. In the future, promise rejections
that are not handled will terminate the
Node.js process with a non-zero exit cod
e.
rodrigobranas:javascriptmasterclass $
```

Para tratar possíveis exceções associadas a chamadas assíncronas é possível utilizar um bloco try/catch

The screenshot shows a terminal window with the following content:

```
rodrigobranas:javascriptmasterclass $ node a
de async_await/async_await_4.js
Invalid input
rodrigobranas:javascriptmasterclass $
```

The terminal window has tabs labeled "TERMINAL" and "1: bash". The command entered is "node a", followed by the path "de async\_await/async\_await\_4.js". The output shows an error message "Invalid input".

The code editor window contains a file named "async\_await\_4.js" with the following content:

```
function sum(a, b) {
  return new Promise(function(resolve, reject) {
    if (!a || !b) return reject("Invalid input");
    setTimeout(function() {
      resolve(a + b);
    }, 1000);
  });
}

(async function () {
  try {
    const a = await sum(2);
    const b = await sum(4, 4);
    const result = await sum(a, b);
    console.log(result);
  } catch (e) {
    console.log(e);
  }
})();
```



É possível **iterar** utilizando `async/await`?

The screenshot shows a terminal window with the following details:

- Title Bar:** async\_await\_5.js — javascriptmasterclass
- Terminal Tab:** 1: bash
- Code Content:** The code defines a function `sum` that returns a Promise. It then uses an `async` function to call `sum` twice, awaiting each result and pushing it into an array. Finally, it logs the array. A catch block handles any errors.

```
JS async_await_5.js x
1  function sum(a, b) {
2      return new Promise(function(resolve, reject) {
3          if (!a || !b) return reject("Invalid input");
4          setTimeout(function() {
5              resolve(a + b);
6          }, 1000);
7      });
8
9
10 (async function () {
11     try {
12         const functions = [
13             sum(2, 2),
14             sum(4, 4)
15         ];
16         const results = [];
17         functions.forEach(function(fn) {
18             const result = await fn;
19             results.push(result);
20         });
21         const [a,b] = results;
22         const result = await sum(a, b);
23         console.log(result);
24     } catch (e) {
25         console.log(e);
26     }
}
```

- Output:** The terminal shows the command `node a sync\_await/async\_await\_5.js` being run. It then displays a stack trace for a `SyntaxError` at line 18, column 10, pointing to the `fn` identifier. The error message is "SyntaxError: Unexpected identifier".

```
rodrigobranas:javascriptmasterclass $ node a sync_await/async_await_5.js
/Users/rodrigobranas/development/workspace/javascriptmasterclass/async_await/async_await_5.js:18
                const result = await fn;
                                         ^
SyntaxError: Unexpected identifier
    at createScript (vm.js:80:10)
    at Object.runInThisContext (vm.js:139:10)
)
    at Module._compile (module.js:607:28)
    at Object.Module._extensions..js (module.js:654:10)
    at Module.load (module.js:556:32)
    at tryModuleLoad (module.js:499:12)
    at Function.Module._load (module.js:491:3)
    at Function.Module.runMain (module.js:684:10)
    at startup (bootstrap_node.js:187:16)
    at bootstrap_node.js:608:3
rodrigobranas:javascriptmasterclass $
```

The screenshot shows a terminal window with the following output:

```
rodrigobranas:javascriptmasterclass $ node a
sync_await/async_await_6.js
Invalid input
4
8
rodrigobranas:javascriptmasterclass $
```

The terminal window is titled "async\_await\_6.js — javascriptmasterclass". The code editor on the left contains the following JavaScript code:

```
1  function sum(a, b) {
2      return new Promise(function(resolve, reject) {
3          if (!a || !b) return reject("Invalid input");
4          setTimeout(function() {
5              resolve(a + b);
6          }, 1000);
7      });
8 }
9
10 (async function () {
11     try {
12         const functions = [
13             sum(2, 2),
14             sum(4, 4)
15         ];
16         const results = [];
17         functions.forEach(async function(fn) {
18             const result = await fn;
19             console.log(result);
20             results.push(result);
21         });
22         const [a,b] = results;
23         const result = await sum(a, b);
24         console.log(result);
25     } catch (e) {
26         console.log(e);
27     }
28 })()
```

The screenshot shows a macOS desktop environment with a terminal window open. The terminal window has a title bar "async\_await\_7.js — javascriptmasterclass". The main pane displays the code for "async\_await\_7.js". The code defines a "sum" function that returns a Promise. It includes a timeout of 1000ms. The main block uses an async function to call "sum" twice, then logs the results. The terminal pane shows the command "node async\_await/async\_await\_7.js" being run, followed by the output "12".

```
JS async_await_7.js x
async_await_7.js — javascriptmasterclass
1  function sum(a, b) {
2      return new Promise(function(resolve, reject) {
3          if (!a || !b) return reject("Invalid input");
4          setTimeout(function() {
5              resolve(a + b);
6          }, 1000);
7      });
8
9
10 (async function () {
11     try {
12         const functions = [
13             sum(2, 2),
14             sum(4, 4)
15         ];
16         const results = [];
17         for (let fn of functions) {
18             const result = await fn;
19             results.push(result);
20         }
21         const [a,b] = results;
22         const result = await sum(a, b);
23         console.log(result);
24     } catch (e) {
25         console.log(e);
26     }
}
```

```
TERMINAL ... 1: bash +
rodrigobranas:javascriptmasterclass $ node a
sync_await/async_await_7.js
12
rodrigobranas:javascriptmasterclass $
```

É possível utilizar o bloco **for-await-of**  
para iterar sobre um iterator de promises

The screenshot shows a terminal window with the following details:

- Title Bar:** async\_await\_8.js — javascriptmasterclass
- Terminal Tab:** 1: bash
- Content:** A stack trace for a SyntaxError:

```
rodrigobranas:javascriptmasterclass $ node a
sync_await/async_await_8.js
/Users/rodrigobranas/development/workspace/j
avascriptmasterclass/async_await/async_await
_8.js:17
    for await (let result of functions)
{
    ^^^^^^

SyntaxError: Unexpected reserved word
  at createScript (vm.js:80:10)
  at Object.runInThisContext (vm.js:139:10)
)
  at Module._compile (module.js:607:28)
  at Object.Module._extensions..js (module
.js:654:10)
  at Module.load (module.js:556:32)
  at tryModuleLoad (module.js:499:12)
  at Function.Module._load (module.js:491:
3)
  at Function.Module.runMain (module.js:68
4:10)
  at startup (bootstrap_node.js:187:16)
  at bootstrap_node.js:608:3
rodrigobranas:javascriptmasterclass $
```

The terminal output indicates that the script was run with node, and it failed at line 17 due to an unexpected reserved word 'for'.

**Code Editor Area:** The code editor shows the contents of `async_await_8.js`:

```
JS async_await_8.js x
1  function sum(a, b) {
2      return new Promise(function(resolve, reject) {
3          if (!a || !b) return reject("Invalid input");
4          setTimeout(function() {
5              resolve(a + b);
6          }, 1000);
7      });
8
9
10 (async function () {
11     try {
12         const functions = [
13             sum(2, 2),
14             sum(4, 4)
15         ];
16         const results = [];
17         for await (let result of functions) {
18             results.push(result);
19         }
20         const [a,b] = results;
21         const result = await sum(a, b);
22         console.log(result);
23     } catch (e) {
24         console.log(e);
25     }
26 })()
```



CAUTION

Para utilizar é necessário usar a  
flag --harmony-async-iteration

The screenshot shows a Mac OS X desktop environment with a terminal window open. The terminal window has a title bar "async\_await\_8.js — javascriptmasterclass". The main pane displays the code for "async\_await\_8.js", which contains two functions: a synchronous helper function "sum" and an asynchronous iteration function using "async function ()". The "sum" function returns a Promise that resolves after 1000ms with the sum of its arguments. The iteration function uses "try", "const", "for await", "push", "const", "result", "catch", and "console.log" statements. The right pane of the terminal window shows the command being run in the terminal: "node --harmony-async-iteration async\_await/async\_await\_8.js". The output of the command is "12", indicating the result of the sum operation.

```
JS async_await_8.js x
async_await_8.js — javascriptmasterclass
1  function sum(a, b) {
2      return new Promise(function(resolve, reject) {
3          if (!a || !b) return reject("Invalid input");
4          setTimeout(function() {
5              resolve(a + b);
6          }, 1000);
7      });
8
9
10 (async function () {
11     try {
12         const functions = [
13             sum(2, 2),
14             sum(4, 4)
15         ];
16         const results = [];
17         for await (let result of functions) {
18             results.push(result);
19         }
20         const [a,b] = results;
21         const result = await sum(a, b);
22         console.log(result);
23     } catch (e) {
24         console.log(e);
25     }
26 })()
TERMINAL ... 1: bash + = 1
rodrigobranas:javascriptmasterclass $ node --harmony-async-iteration async_await/async_await_8.js
12
rodrigobranas:javascriptmasterclass $
```