

A close-up photograph of two business people shaking hands. One person is wearing a grey suit jacket over a white shirt with a pink striped tie. The other person is wearing a dark blue suit jacket over a white shirt. The background is blurred.

Promises

As promises são objetos responsáveis  
por modelar comportamento assíncrono,  
permitindo o seu tratamento de uma  
forma mais fácil e direta

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor window.

The terminal window (top right) has the title "TERMINAL" and "1: bash". It contains the following text:

```
rodrigobranas:javascriptmasterclass $ node promises/promises_1.js
4
rodrigobranas:javascriptmasterclass $
```

The code editor window (left side) has the title "promises\_1.js — javascriptmasterclass". It contains the following JavaScript code:

```
JS promises_1.js ×
1  function sum(a, b) {
2    return a + b;
3  }
4  const result = sum(2, 2);
5  console.log(result);
6
```

promises\_2.js — javascriptmasterclass

JS promises\_2.js x

```
1 function delayedSum(a, b) {
2     setTimeout(function() {
3         return a + b;
4     }, 1000);
5 }
6 const result = delayedSum(2, 2);
7 console.log(result);
8
```

TERMINAL ... 1: bash

```
rodrigobranas:javascriptmasterclass $ node promises/promises_2.js
undefined
rodrigobranas:javascriptmasterclass $
```

promises\_3.js — javascriptmasterclass

JS promises\_3.js x

```
1 function delayedSum(a, b, callback) {
2     setTimeout(function() {
3         callback(a + b);
4     }, 1000);
5 }
6 delayedSum(2, 2, function(result) {
7     console.log(result);
8 });
9
```

TERMINAL ... 1: bash

rodrigobranas:javascriptmasterclass \$ node promises/promises\_3.js  
4  
rodrigobranas:javascriptmasterclass \$

A screenshot of a Mac OS X desktop environment showing a terminal window. The window has three panes: a file editor on the left, a terminal on the right, and a status bar at the top.

The title bar of the window says "promises\_4.js — javascriptmasterclass".

The file editor pane contains the following JavaScript code:

```
JS promises_4.js x
1 function delayedSum(a, b, callback) {
2     setTimeout(function() {
3         callback(a + b);
4     }, 1000);
5 }
6 delayedSum(2, 2, function(a) {
7     delayedSum(4, 4, function(b) {
8         delayedSum(a, b, function(result) {
9             console.log(result);
10        });
11    });
12 });
13
```

The terminal pane shows the output of running the script:

```
rodrigobranas:javascriptmasterclass $ node promises/promises_4.js
12
rodrigobranas:javascriptmasterclass $
```

Para criar uma promise basta instanciá-la, executando a função **resolve** em caso de sucesso, sendo tratado por meio de **then**

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor.

The terminal window (top right) has tabs for "TERMINAL" and "1: bash". The command `node promises/promises\_5.js` is run, followed by the output "12".

The code editor window (left) shows a file named "promises\_5.js" with the following content:

```
promises_5.js — javascriptmasterclass
JS promises_5.js x
1 function delayedSum(a, b) {
2     return new Promise(function (resolve) {
3         setTimeout(function() {
4             resolve(a + b);
5         }, 1000);
6     });
7 }
8 delayedSum(2, 2).then(function(a) {
9     delayedSum(4, 4).then(function(b) {
10        delayedSum(a, b).then(function(result) {
11            console.log(result);
12        });
13    });
14 });
15
```

Em caso de fracasso, a função **reject** deve ser executada, sendo tratada por meio de **catch**

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor side-by-side.

The terminal window on the right shows the command `node promises/promises_6.js` being run. It outputs several lines of text, including an `UnhandledPromiseRejectionWarning`, a `DeprecationWarning` about unhandled promise rejections, and a final prompt `rodrigobranas:javascriptmasterclass $`.

The code editor window on the left contains the file `promises_6.js`. The code defines a function `delayedSum` that returns a Promise. If either `a` or `b` is not provided, it rejects with the message "Invalid input". Otherwise, it resolves after a 1-second delay with the sum of `a` and `b`. The code then demonstrates chaining multiple `delayedSum` calls to print the result of `2 + 2 + 4 + 4`.

```
JS promises_6.js x
promises_6.js — javascriptmasterclass
1  function delayedSum(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  delayedSum(2, 2).then(function(a) {
10     delayedSum(4, 4).then(function(b) {
11         delayedSum().then(function(result) {
12             console.log(result);
13         });
14     });
15 });
16 
```

```
TERMINAL ... 1: bash + = 1
rodrigobranas:javascriptmasterclass $ node promises/promises_6.js
(node:62173) UnhandledPromiseRejectionWarning: Invalid input
(node:62173) [DEP0018] DeprecationWarning: Unhandled promise rejection (rejection id : 2): Invalid input
In the future, promise rejections that are not handled will terminate the Node.js process with a non-zero exit code.
rodrigobranas:javascriptmasterclass $
```

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor window.

The terminal window (top right) has the title "TERMINAL" and "1: bash". It contains the following text:

```
rodrigobranas:javascriptmasterclass $ node promises/promises_7.js
Invalid input
rodrigobranas:javascriptmasterclass $
```

The code editor window (left side) has the title "promises\_7.js — javascriptmasterclass". It contains the following JavaScript code:

```
1  function delayedSum(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  delayedSum(2, 2).then(function(a) {
10     delayedSum(4, 4).then(function(b) {
11         delayedSum().then(function(result) {
12             console.log(result);
13         }).catch(function (e) {
14             console.log(e);
15         });
16     });
17 });
18 }
```

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor. The terminal window is titled 'TERMINAL' and has a tab labeled '1: bash'. It displays the output of a Node.js script named 'promises\_8.js'. The script contains a function 'delayedSum' that returns a promise. If either input is falsy, it rejects the promise with the message 'Invalid input'. Otherwise, it resolves the promise after a 1-second delay with the sum of the inputs. The terminal shows the script being run, followed by an unhandled promise rejection warning from Node.js version 6.21.91, which states that unhandled promise rejections are deprecated and will terminate the process with a non-zero exit code.

```
promises_8.js — javascriptmasterclass
JS promises_8.js x
1  function delayedSum(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  delayedSum(2, 2).then(function(a) {
10     delayedSum().then(function(b) {
11         delayedSum().then(function(result) {
12             console.log(result);
13         }).catch(function (e) {
14             console.log(e);
15         });
16     });
17 });
18
```

```
rodrigobranas:javascriptmasterclass $ node promises/promises_8.js
(node:62191) UnhandledPromiseRejectionWarning: Unhandled promise rejection (rejection id : 2): Invalid input
(node:62191) [DEP0018] DeprecationWarning: Unhandled promise rejections are deprecated.
In the future, promise rejections that are not handled will terminate the Node.js process with a non-zero exit code.
rodrigobranas:javascriptmasterclass $
```

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor window.

The terminal window (top right) has the title "TERMINAL" and "1: bash". It contains the following text:

```
rodrigobranas:javascriptmasterclass $ node promises/promises_9.js
Invalid input
rodrigobranas:javascriptmasterclass $
```

The code editor window (left side) has the title "promises\_9.js — javascriptmasterclass". It contains the following JavaScript code:

```
1  function delayedSum(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 delayedSum(2, 2).then(function(a) {
10     delayedSum().then(function(b) {
11         delayedSum().then(function(result) {
12             console.log(result);
13         }).catch(function (e) {
14             console.log(e);
15         });
16     }).catch(function (e) {
17         console.log(e);
18     });
19 });
20
```

A screenshot of a Mac OS X desktop environment showing a terminal window. The terminal window has a title bar "promises\_10.js — javascriptmasterclass". The main pane displays the contents of a file named "promises\_10.js". The code defines a function "delayedSum" that returns a Promise. If the input is invalid (either a or b is null or undefined), it rejects with the message "Invalid input". Otherwise, it resolves after a 1-second delay with the sum of a and b. This function is then chained with three .then blocks and two .catch blocks, each logging the result or error to the console.

```
JS promises_10.js x
promises_10.js — javascriptmasterclass
1  function delayedSum(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  delayedSum().then(function(a) {
10     delayedSum().then(function(b) {
11         delayedSum().then(function(result) {
12             console.log(result);
13         }).catch(function (e) {
14             console.log(e);
15         });
16     }).catch(function (e) {
17         console.log(e);
18     });
19 });
20 
```

The right-hand terminal pane shows the output of running the script with node. It includes an UnhandledPromiseRejectionWarning and a DeprecationWarning about unhandled promise rejections being deprecated.

```
rodrigobranas:javascriptmasterclass $ node promises/promises_10.js
(node:62223) UnhandledPromiseRejectionWarning: Invalid input
(node:62223) [DEP0018] DeprecationWarning: Unhandled promise rejections are deprecated.
In the future, promise rejections that are not handled will terminate the Node.js process with a non-zero exit code.
rodrigobranas:javascriptmasterclass $
```

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor. The terminal window is titled '1: bash' and contains the command 'node promises/promises\_11.js' followed by the error message 'Invalid input'. The code editor window is titled 'promises\_11.js — javascriptmasterclass' and displays a JavaScript file named 'promises\_11.js'.

```
JS promises_11.js × promises_11.js — javascriptmasterclass TERMINAL ... 1: bash + = 1

1 function delayedSum(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 delayedSum().then(function(a) {
10     delayedSum().then(function(b) {
11         delayedSum().then(function(result) {
12             console.log(result);
13         }).catch(function (e) {
14             console.log(e);
15         });
16     }).catch(function (e) {
17         console.log(e);
18     });
19 }).catch(function (e) {
20     console.log(e);
21 });
22
```

```
rodrigobranas:javascriptmasterclass $ node promises/promises_11.js
Invalid input
rodrigobranas:javascriptmasterclass $
```

É possível **centralizar** o tratamento de uma promise encadeando seus retornos

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor window.

The terminal window (top right) has the title "TERMINAL" and "1: bash". It contains the following text:

```
rodrigobranas:javascriptmasterclass $ node promises/promises_12.js
Invalid input
rodrigobranas:javascriptmasterclass $
```

The code editor window (left side) has the title "promises\_12.js — javascriptmasterclass". It contains the following JavaScript code:

```
1  function delayedSum(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 delayedSum(2, 2).then(function(a) {
10     return delayedSum(4, 4).then(function(b) {
11         return delayedSum().then(function(result) {
12             console.log(result);
13         });
14     });
15 }).catch(function (e) {
16     console.log(e);
17 });
18
```

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor window.

The terminal window (top right) has the title "TERMINAL" and "1: bash". It contains the following text:

```
rodrigobranas:javascriptmasterclass $ node promises/promises_13.js
12
performance: 3021.098ms
rodrigobranas:javascriptmasterclass $
```

The code editor window (left side) has the title "promises\_13.js — javascriptmasterclass". It contains the following JavaScript code:

```
1  function delayedSum(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  console.time("performance");
10 delayedSum(2, 2).then(function(a) {
11     return delayedSum(4, 4).then(function(b) {
12         return delayedSum(a, b).then(function(result) {
13             console.log(result);
14             console.timeEnd("performance");
15         });
16     });
17 }).catch(function (e) {
18     console.log(e);
19 });
20
```

Podemos executar várias promises ao mesmo tempo, retornando após todas terem sucesso usando **Promise.all**

A screenshot of a terminal window titled "promises\_14.js — javascriptmasterclass". The terminal shows the command \$ node promises/promises\_14.js being run, followed by the output 12 and performance: 2011.724ms.

```
JS promises_14.js x
promises_14.js — javascriptmasterclass
TERMINAL ... 1: bash + = 1

rodrigobranas:javascriptmasterclass $ node promises/promises_14.js
12
performance: 2011.724ms
rodrigobranas:javascriptmasterclass $
```

```
1  function delayedSum(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  console.time("performance");
10 Promise.all([
11     delayedSum(2, 2),
12     delayedSum(4, 4)
13 ]).then(function(values) {
14     let [a, b] = values;
15     return delayedSum(a, b).then(function(result) {
16         console.log(result);
17         console.timeEnd("performance");
18     });
19 }).catch(function (e) {
20     console.log(e);
21 });
22
```

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor.

The terminal window (top right) has the title "TERMINAL" and "1: bash". It contains the following text:

```
rodrigobranas:javascriptmasterclass $ node promises/promises_15.js
Invalid input
rodrigobranas:javascriptmasterclass $
```

The code editor window (left side) has the title "promises\_15.js — javascriptmasterclass". It contains the following JavaScript code:

```
1  function delayedSum(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  Promise.all([
10      delayedSum(2, 2),
11      delayedSum(4, 4)
12 ]).then(function(values) {
13     let [a, b] = values;
14     return delayedSum().then(function(result) {
15         console.log(result);
16     });
17 }).catch(function (e) {
18     console.log(e);
19 });
20
```

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor window.

The terminal window (top right) has the title "TERMINAL" and shows the command:

```
rodrigobranas:javascriptmasterclass $ node promises/promises_16.js
```

The output of the command is:

```
12
```

The code editor window (left side) has the title "promises\_16.js — javascriptmasterclass". It contains the following JavaScript code:

```
1  function delayedSum(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  Promise.all([
10      delayedSum(2, 2),
11      delayedSum(4, 4)
12 ]).then(function(values) {
13     let [a, b] = values;
14     return delayedSum(a, b).then(function(result) {
15         console.log(result);
16     });
17 }).catch(function (e) {
18     console.log(e);
19 });
20
```

Também podemos executar várias promises ao mesmo, retornando após a primeira ter sucesso usando `Promise.race`

A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor.

The terminal window (top right) has the title "TERMINAL" and "1: bash". It displays the command `node promises/promises\_17.js` followed by the output "16".

The code editor window (left side) has the title "promises\_17.js — javascriptmasterclass". It contains the following JavaScript code:

```
JS promises_17.js ×

1  function delayedSum(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          }, Math.random() * 1000);
7      });
8  }
9  Promise.race([
10     delayedSum(2, 2),
11     delayedSum(4, 4)
12 ]).then(function(value) {
13     return delayedSum(value, value).then(function(result) {
14         console.log(result);
15     });
16 }).catch(function (e) {
17     console.log(e);
18 });
19
```