

A photograph of a space shuttle launching from a launch pad. The shuttle is white with a dark blue thermal protection system on its nose and wings. It is mounted on a large white external fuel tank with two solid rocket boosters attached. To the left of the shuttle is a tall, multi-tiered metal tower with various equipment and a circular window at the top. A massive plume of white smoke and fire erupts from the base of the shuttle, partially obscuring the launch pad. The background is a clear blue sky.

Function

Uma função é um objeto que contém um bloco de código executável

A screenshot of a macOS desktop environment. On the left is a code editor window titled "function_1.js — javascriptmasterclass". The editor shows the following JavaScript code:

```
JS function_1.js x
1  function sum(a, b) {
2    return a + b;
3  }
4  console.log(sum(2, 2));
5
```

The code defines a function named "sum" that takes two arguments, "a" and "b", and returns their sum. It then logs the result of calling "sum" with arguments 2 and 2 to the console.

To the right of the code editor is a terminal window titled "TERMINAL" with a tab labeled "1: bash". The terminal output is:

```
rodrigobranas:javascriptmasterclass $ node function/function_1.js
4
rodrigobranas:javascriptmasterclass $
```

The terminal command "node function/function_1.js" was run, and the output "4" was displayed, indicating the correct execution of the function.

A screenshot of a macOS desktop environment. On the left is a code editor window titled "function_2.js — javascriptmasterclass". The file contains the following JavaScript code:

```
JS function_2.js x
1 const sum = function(a, b) {
2     return a + b;
3 }
4 console.log(sum(2, 2));
5
```

The code editor has standard OS X window controls (red, yellow, green) at the top-left. To the right of the editor is a terminal window titled "TERMINAL" with a tab labeled "1: bash". The terminal output shows:

```
rodrigobranas:javascriptmasterclass $ node function/function_2.js
4
rodrigobranas:javascriptmasterclass $
```



Qual é a diferença entre function
declaration e expression?

A screenshot of a macOS desktop environment showing a terminal window and a code editor. The terminal window is titled 'function_3.js — javascriptmasterclass' and contains the command 'node function/function_3.js' followed by the output '4'. The code editor window shows a file named 'function_3.js' with the following content:

```
JS function_3.js x  
function sum(a, b) {  
    return a + b;  
}  
1 console.log(sum(2, 2));  
2  
3  
4  
5
```

function_4.js — javascriptmasterclass

JS function_4.js x

```
1 console.log(sum(2, 2));
2 const sum = function(a, b) {
3     return a + b;
4 }
5
```

TERMINAL ... 1: bash

rodrigobranas:javascriptmasterclass \$ node function/function_4.js
/Users/rodrigobranas/development/workspace/javascriptmasterclass/function/function_4.js:1
(function (exports, require, module, __filename, __dirname) { console.log(sum(2, 2));
^
ReferenceError: sum is not defined
 at Object.<anonymous> (/Users/rodrigobranas/development/workspace/javascriptmasterclass/function/function_4.js:1:75)
 at Module._compile (module.js:643:30)
 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 \$ █

Na linguagem JavaScript, **as funções**
são de primeira classe, ou seja, podem
ser atribuídas a uma variável, passadas
por parâmetro ou serem retornada de
uma outra função

function_5.js — javascriptmasterclass

JS function_5.js x

TERMINAL ... 1: bash + =

```
1 let sum = function (a, b) {  
2     return a + b;  
3 };  
4 let subtract = function (a, b) {  
5     return a - b;  
6 };  
7 let calculator = function (fn) {  
8     return function (a, b) {  
9         return fn(a, b);  
10    }  
11 };  
12 console.log(calculator(sum)(2, 2));  
13 console.log(calculator(subtract)(2, 2));  
14
```

rodrigobranas:javascriptmasterclass \$ node function/function_5.js
4
0
rodrigobranas:javascriptmasterclass \$

É possível invocar uma função com menos ou mais parâmetros, não necessariamente seguindo o que está declarado

function_6.js — javascriptmasterclass

JS function_6.js x

```
1  function sum(a, b) {  
2      return a + b;  
3  }  
4  console.log(sum(2, 2));  
5  console.log(sum(5));  
6  console.log(sum(1, 2, 3));  
7
```

TERMINAL ... 1: bash

rodrigobranas:javascriptmasterclass \$ node function/function_6.js
4
NaN
3
rodrigobranas:javascriptmasterclass \$

Podemos definir **valores padrão** para cada um dos parâmetros de uma função

function_7.js — javascriptmasterclass

JS function_7.js x

TERMINAL ... 1: bash + = 1

```
1 function sum(a = 1, b = 1) {  
2     return a + b;  
3 }  
4 console.log(sum(2, 2));  
5 console.log(sum(5));  
6 console.log(sum());  
7
```

rodrigobranas:javascriptmasterclass \$ node function/function_7.js
4
6
2
rodrigobranas:javascriptmasterclass \$

Por meio da variável implícita
arguments é possível acessar os
parâmetros da função invocada

function_8.js — javascriptmasterclass

JS function_8.js x

TERMINAL ... 1: bash + =

```
1 let sum = function () {  
2     let total = 0;  
3     for(let argument in arguments) {  
4         total += arguments[argument];  
5     }  
6     return total;  
7 };  
8 console.log(sum(1,2,3,4,5,6,7,8,9));  
9
```

rodrigobranas:javascriptmasterclass \$ node function/function_8.js
45
rodrigobranas:javascriptmasterclass \$

Também é possível acessar os parâmetros da função invocada por meio do **rest parameter**

function_9.js — javascriptmasterclass

JS function_9.js x

TERMINAL ... 1: bash + =

```
1 let sum = function (...numbers) {  
2     let total = 0;  
3     for(let number of numbers) {  
4         total += number;  
5     }  
6     return total;  
7 };  
8 console.log(sum(1,2,3,4,5,6,7,8,9));  
9
```

rodrigobranas:javascriptmasterclass \$ node function/function_9.js
45
rodrigobranas:javascriptmasterclass \$



CAUTION

O rest parameter deve ser sempre o
último da lista de parâmetros

function_10.js — javascriptmasterclass

JS function_10.js x

TERMINAL ... 1: bash + =

```
1 let sum = function (a, b, c, ...numbers) {
2     let total = a + b + c;
3     for(let number of numbers) {
4         total += number;
5     }
6     return total;
7 };
8 console.log(sum(1,2,3,4,5,6,7,8,9));
9
```

rodrigobranas:javascriptmasterclass \$ node function/function_10.js
45
rodrigobranas:javascriptmasterclass \$

function_11.js — javascriptmasterclass

JS function_11.js x

TERMINAL ... 1: bash + =

```
1 let sum = function (...numbers, a, b, c) {  
2     let total = a + b + c;  
3     for(let number of numbers) {  
4         total += number;  
5     }  
6     return total;  
7 };  
8 console.log(sum(1,2,3,4,5,6,7,8,9));  
9
```

rodrigobranas:javascriptmasterclass \$ node function/function_11.js
/Users/rodrigobranas/development/workspace/javascriptmasterclass/function/function_11.js
:1
(function (exports, require, module, __filename, __dirname) { let sum = function (...numbers, a, b, c) {
^
SyntaxError: Rest parameter must be last formal parameter
 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 \$ █