Function

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

function_1.js — javascriptmasterclass

function_1.js

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

TERMINAL

1: bash

```
rodrigobranas:javascriptmasterclass $ node f
unction/function_1.js
4
rodrigobranas:javascriptmasterclass $
```

function_2.js — javascriptmasterclass

function_2.js

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

TERMINAL

1: bash

```
rodrigobranas:javascriptmasterclass $ node f
unction/function_2.js
4
rodrigobranas:javascriptmasterclass $
```

Qual é a diferença entre function declaration e expression?

function_3.js — javascriptmasterclass

function_3.js

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

TERMINAL

1: bash

```
rodrigobranas:javascriptmasterclass $ node f
unction/function_3.js
4
rodrigobranas:javascriptmasterclass $
```

function_4.js — javascriptmasterclass

JS function_4.js

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

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node f
unction/function_4.js
/Users/rodrigobranas/development/workspace/j
avascriptmasterclass/function/function_4.js:
1
(function (exports, require, module, __filen
ame, __dirname) { console.log(sum(2, 2));

                                ^


ReferenceError: sum is not defined
    at Object.<anonymous> (/Users/rodrigobra
nas/development/workspace/javascriptmastercl
ass/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:68
4:10)
    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

function_5.js

```
let sum = function (a, b) {
    return a + b;
};
let subtract = function (a, b) {
    return a - b;
};
let calculator = function (fn) {
    return function (a, b) {
        return fn(a, b);
    }
};
console.log(calculator(sum)(2, 2));
console.log(calculator(subtract)(2, 2));

```

TERMINAL

1: bash

```
rodrigobranas:javascriptmasterclass $ node f
unction/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

```
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 f
unction/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

```
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
```

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node f
unction/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

function_8.js

```
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
```

TERMINAL

1: bash

```
rodrigobranas:javascriptmasterclass $ node f
unction/function_8.js
45
rodrigobranas:javascriptmasterclass $
```

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

function_9.js — javascriptmasterclass

function_9.js

```
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
```

TERMINAL

1: bash

```
rodrigobranas:javascriptmasterclass $ node f
unction/function_9.js
45
rodrigobranas:javascriptmasterclass $
```

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

function_10.js — javascriptmasterclass

JS function_10.js

```
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
```

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node f
unction/function_10.js
45
rodrigobranas:javascriptmasterclass $
```

function_11.js — javascriptmasterclass

function_11.js

```
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
```

TERMINAL

1: bash

```
rodrigobranas:javascriptmasterclass $ node f
unction/function_11.js
/Users/rodrigobranas/development/workspace/j
avascriptmasterclass/function/function_11.js
:1
(function (exports, require, module, __filen
ame, __dirname) { let sum = function (...num
bers, a, b, c) {


    ^

SyntaxError: Rest parameter must be last for
mal 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:68
4:10)
    at startup (bootstrap_node.js:187:16)
    at bootstrap_node.js:608:3
rodrigobranas:javascriptmasterclass $
```