

Mobile Application Development

Prof. Fernando Pinéo

Sejam bem-vindos

Apresentação

FIAP

Fernando Pinéo

Teacher1



- Formado em Ciência da Computação
- Pós Graduado em Segurança e Defesa Cibernética
- Cursando Pós em Cibersegurança e Governança de dados

- Professor Universitário



- Instrutor de Form. Profis



- Professor Técnico



Conteúdo programático

Conteúdo programático



CONTEÚDO

Introdução a Programação JavaScript

Ambiente de desenvolvimento, NPM e YARN, e dependências

Views e Componentes

Componentes Funcionais

Componentes Funcionais

Check point 4

Persistência de Dados (Async Storage)

React Hooks - useState

React Hooks - useEffect

Conteúdo programático



React Hooks - useContext

Check point 5

Requisições HTTP (Axios)

Firebase

Firebase

Check point 6

Introdução a JavaScript

Para essa prática, será necessário a instalação do node.js e vscode

Declaração de variáveis

```
2 var nome = "Fernando"  
3 let sobreNome = "Abreu"  
4 const anoAtual = 2025
```

Qual a diferença entre VAR e LET?

VAR : Tem escopo de função ou global.

LET : Tem escopo de bloco, ou seja, ela só é acessível dentro do bloco { } onde foi declarada (como em loops ou condicionais).

VAR vs LET

FIAP

```
VS Code interface showing a JavaScript file named 'introducao.js' with the following code:

1 function exemploVar() {
2   if (true) {
3     var x = 10;
4   }
5   console.log(x);
6 }
7
8 exemploVar();
9 console.log(x);
```

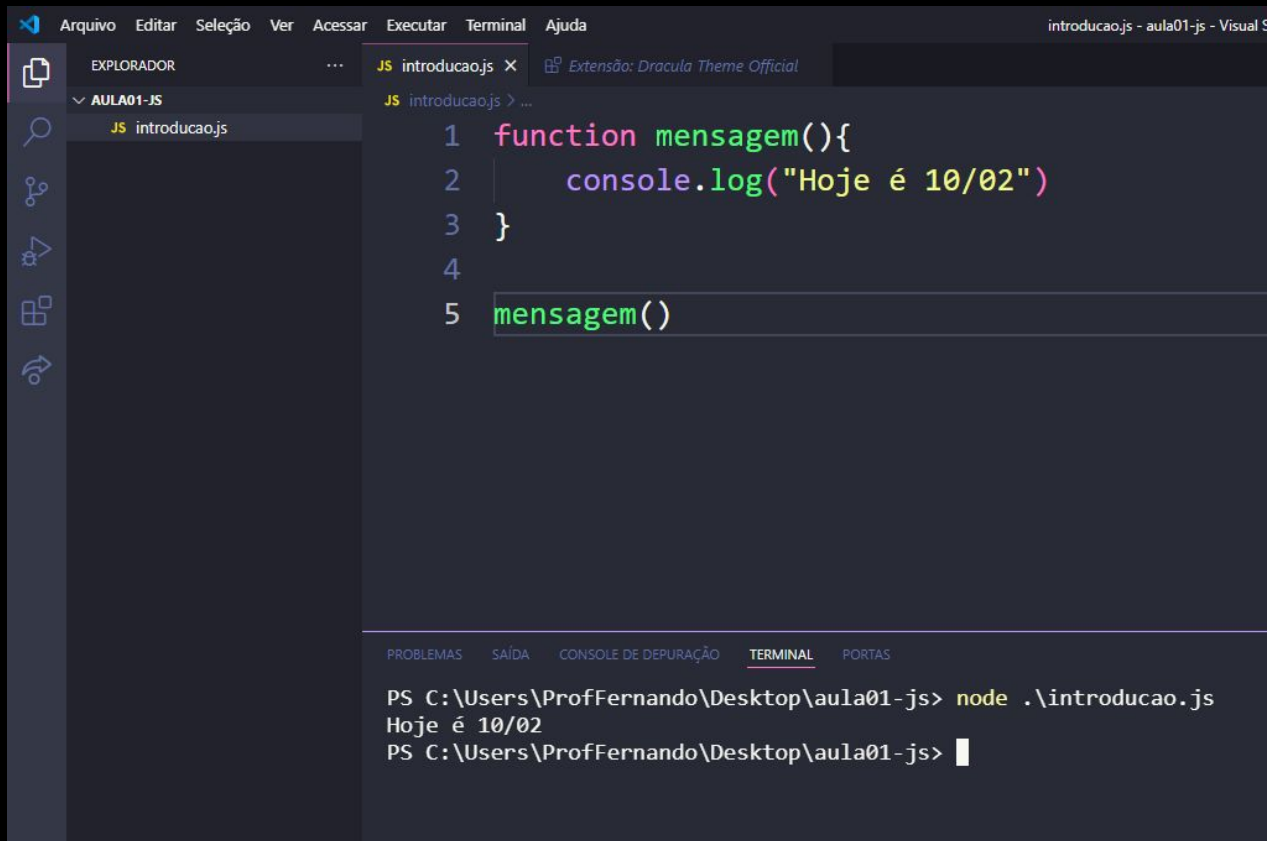


```
VS Code interface showing a JavaScript file named 'introducao.js' with the following code:

1 function exemploLet() {
2   if (true) {
3     let y = 20;
4   }
5   console.log(y);
6 }
7
8 exemploLet();
9 console.log(y);
```



Função simples



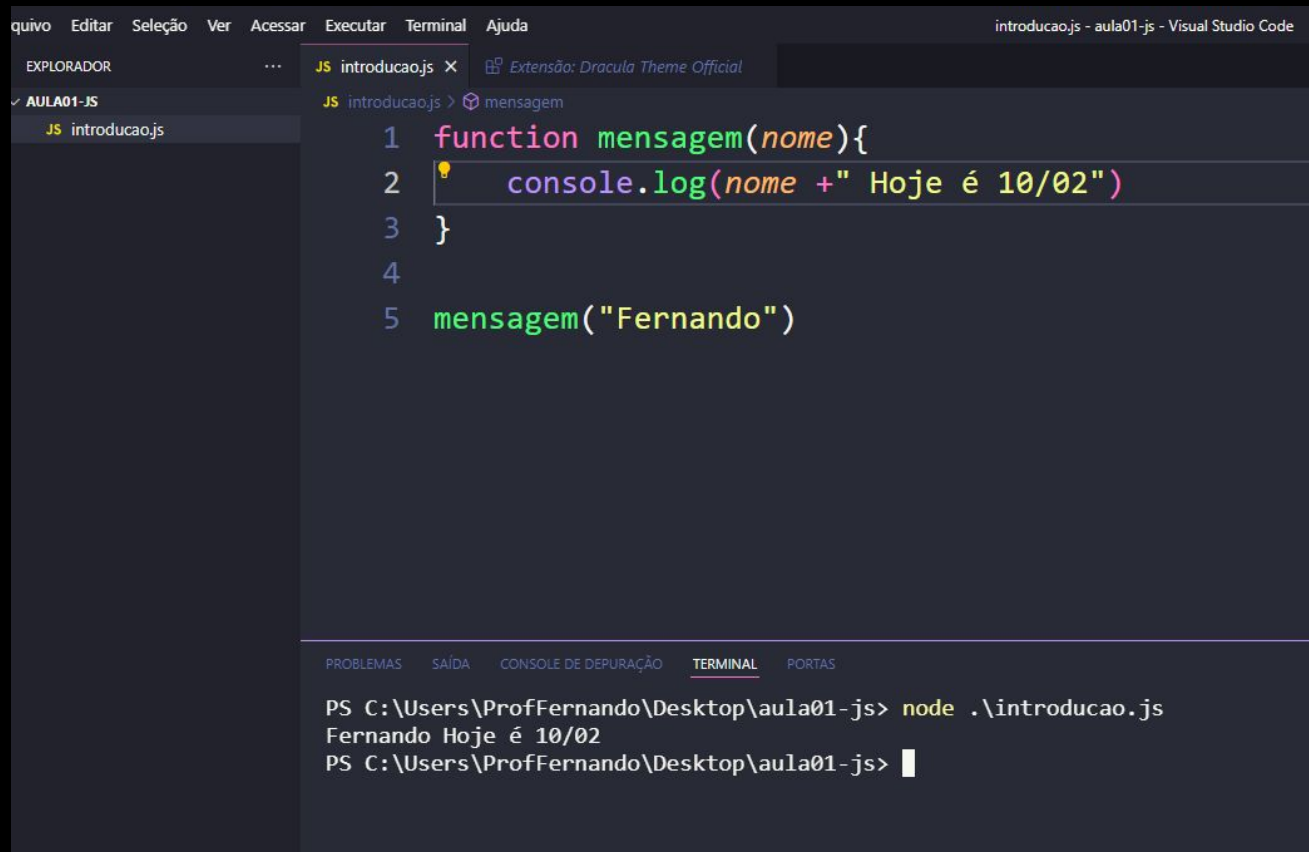
The image shows a screenshot of the Visual Studio Code editor interface. The top menu bar includes 'Arquivo', 'Editar', 'Seleção', 'Ver', 'Acessar', 'Executar', 'Terminal', and 'Ajuda'. The Explorer sidebar on the left shows a project named 'AULA01-JS' with a file 'introducao.js'. The main editor area displays the following JavaScript code:

```
1 function mensagem(){
2     console.log("Hoje é 10/02")
3 }
4
5 mensagem()
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\introducao.js
Hoje é 10/02
PS C:\Users\ProfFernando\Desktop\aula01-js> 
```

Função com parâmetro



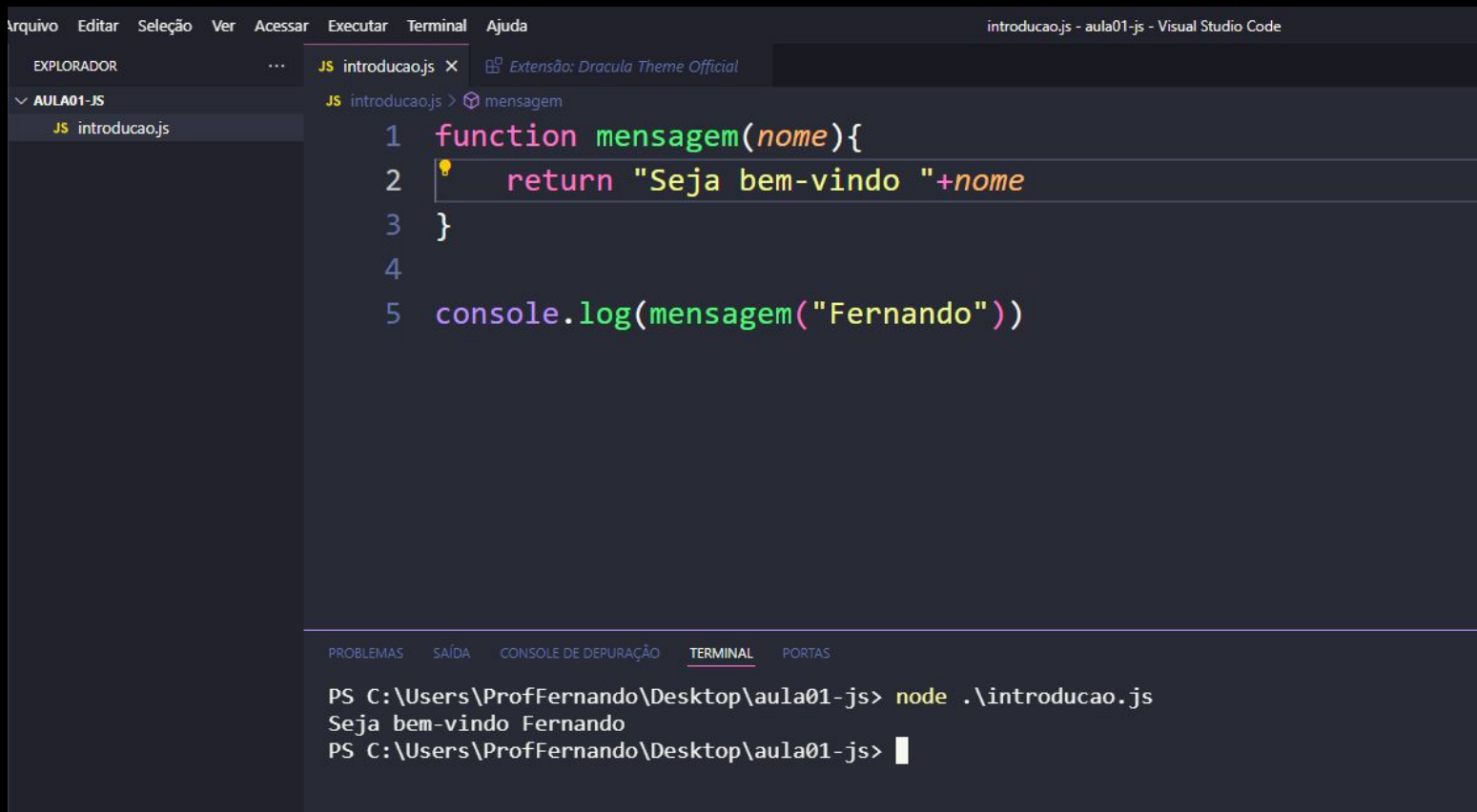
The image shows a screenshot of the Visual Studio Code editor interface. The top menu bar includes 'Arquivo', 'Editar', 'Seleção', 'Ver', 'Acessar', 'Executar', 'Terminal', and 'Ajuda'. The title bar indicates the file is 'introducao.js - aula01.js - Visual Studio Code'. The Explorer sidebar on the left shows a project named 'AULA01-JS' with a file 'introducao.js'. The main editor area displays the following JavaScript code:

```
1 function mensagem(nome){  
2     console.log(nome + " Hoje é 10/02")  
3 }  
4  
5 mensagem("Fernando")
```

Below the editor, the TERMINAL tab is active, showing the command prompt output:

```
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\introducao.js  
Fernando Hoje é 10/02  
PS C:\Users\ProfFernando\Desktop\aula01-js> |
```

Função com retorno



The image shows a screenshot of the Visual Studio Code editor. The top menu bar includes 'Arquivo', 'Editar', 'Seleção', 'Ver', 'Acessar', 'Executar', 'Terminal', and 'Ajuda'. The title bar reads 'introducao.js - aula01.js - Visual Studio Code'. The Explorer sidebar on the left shows a folder named 'AULA01-JS' containing a file 'introducao.js'. The main editor area displays the following JavaScript code in a file named 'introducao.js':

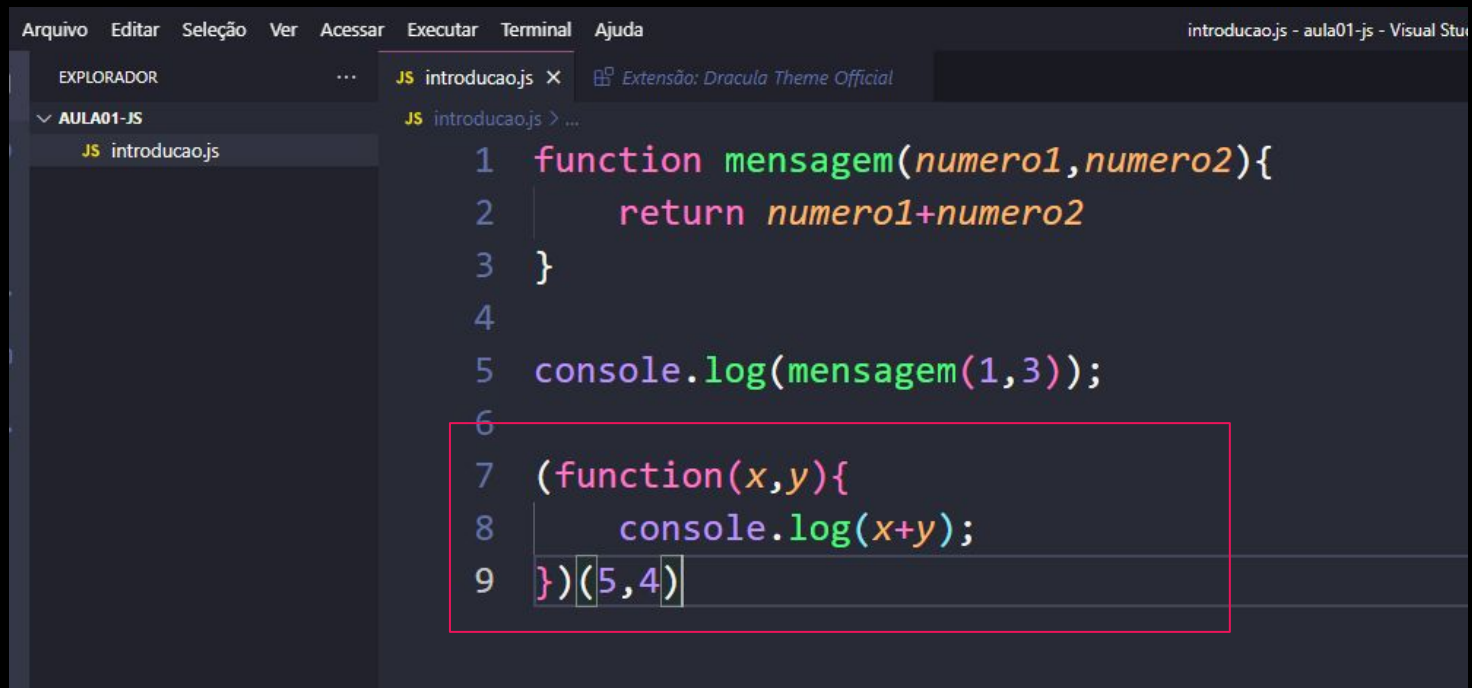
```
1 function mensagem(nome){  
2     return "Seja bem-vindo "+nome  
3 }  
4  
5 console.log(mensagem("Fernando"))
```

Below the editor, the TERMINAL panel is active, showing the command prompt output:

```
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\introducao.js  
Seja bem-vindo Fernando  
PS C:\Users\ProfFernando\Desktop\aula01-js> 
```

Função anônima

Uma função anônima em JavaScript é uma função sem nome, geralmente utilizada em situações onde a função não precisa ser reutilizada e é passada como argumento para outras funções ou executada imediatamente.



```
Arquivo  Editar  Seleção  Ver  Acessar  Executar  Terminal  Ajuda  introducao.js - aula01-js - Visual Studio Code

EXPLORADOR  ...  JS introducao.js  Extensão: Dracula Theme Official

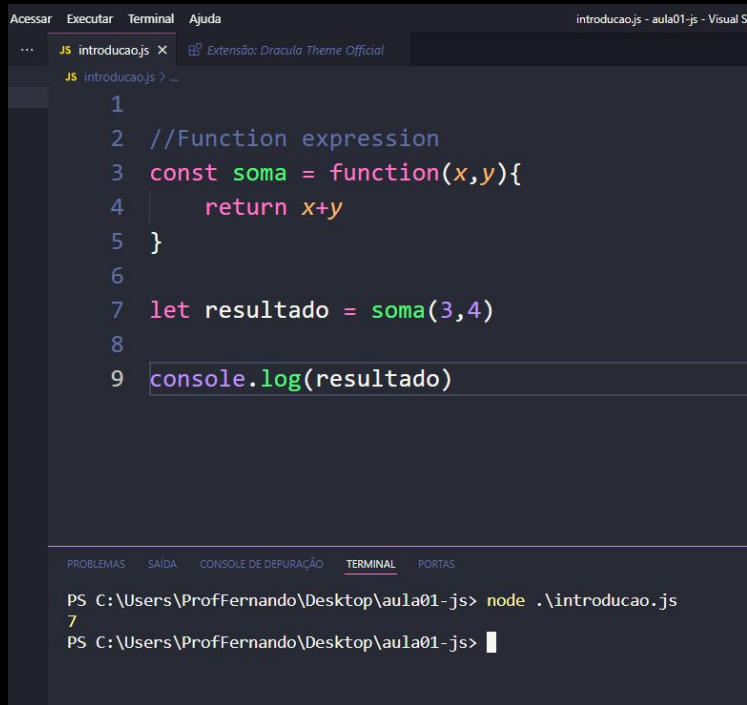
AULA01-JS
  JS introducao.js

1  function mensagem(numero1,numero2){
2      return numero1+numero2
3  }
4
5  console.log(mensagem(1,3));
6
7  (function(x,y){
8      console.log(x+y);
9  })(5,4)
```

The screenshot shows a Visual Studio Code editor window with a file named 'introducao.js'. The code defines a function 'mensagem' and calls it with 'console.log(mensagem(1,3))'. Below that, an anonymous function is defined and immediately invoked with the arguments '5' and '4'. This invocation is highlighted with a red box.

Function Expression

é uma forma de declarar uma função, onde a função é definida como parte de uma expressão e, geralmente, atribuída a uma variável. Ela pode ser anônima (sem nome) ou nomeada.



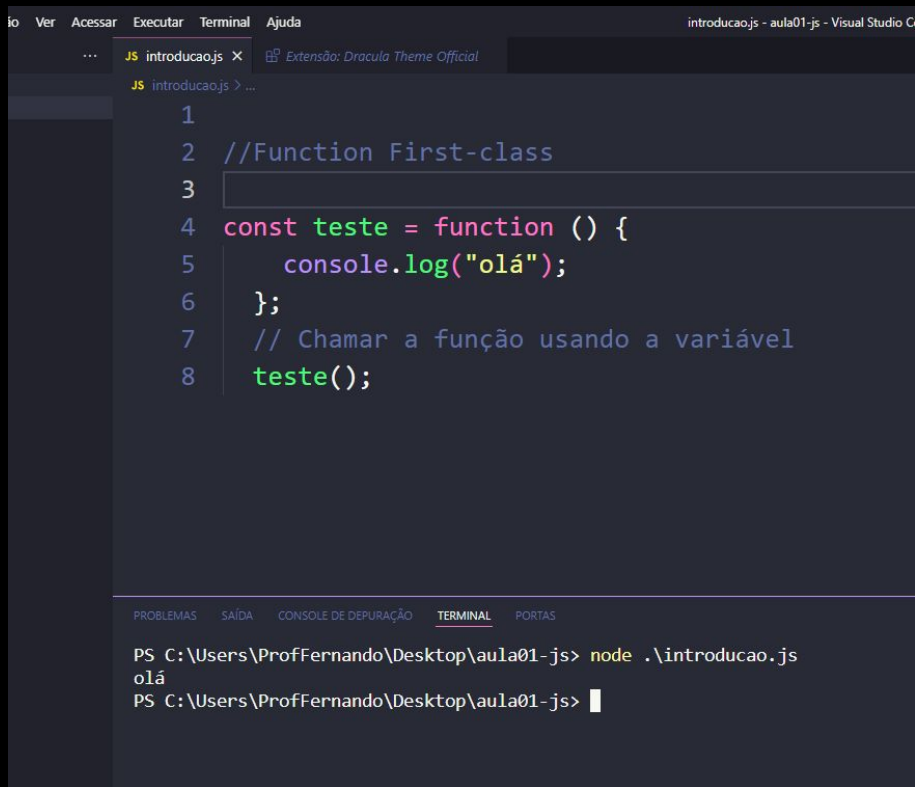
```
1
2 //Function expression
3 const soma = function(x,y){
4     return x+y
5 }
6
7 let resultado = soma(3,4)
8
9 console.log(resultado)
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO **TERMINAL** PORTAS

```
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\introducao.js
7
PS C:\Users\ProfFernando\Desktop\aula01-js> 
```

Função First-class

função first-class (ou função de primeira classe) significa que as funções podem ser tratadas como valores. Isso significa que você pode atribuir uma função a uma variável, passar uma função como argumento para outra função, dentre outros usos.



```
io  Ver  Acessar  Executar  Terminal  Ajuda  introducao.js - aula01-js - Visual Studio C...
... JS introducao.js X Extensão: Dracula Theme Official
JS introducao.js > ...
1
2 //Function First-class
3
4 const teste = function () {
5     console.log("olá");
6 };
7 // Chamar a função usando a variável
8 teste();

PROBLEMAS  SAÍDA  CONSOLE DE DEPURACÃO  TERMINAL  PORTAS
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\introducao.js
olá
PS C:\Users\ProfFernando\Desktop\aula01-js> 
```


Funções Arrow

```
JS introducao.js X
JS introducao.js > ...
1 let triplo = function(a){
2   return 3*a
3 }
4
5 console.log(triplo(3))

PROBLEMAS SAÍDA CONSOLE DE DEPURACÃO TERMINAL PORTAS
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\introducao.js
9
PS C:\Users\ProfFernando\Desktop\aula01-js> █
```

Normal

```
ssar Executar Terminal Ajuda introducao.js
JS introducao.js X
JS introducao.js > ...
1 let triplo = (a) =>{
2   return 3*a
3 }
4
5 console.log(triplo(4))

PROBLEMAS SAÍDA CONSOLE DE DEPURACÃO TERMINAL PORTAS
PS C:\Users\ProfFernando\Desktop\aula01-js> █
```

Primeira forma

```
JS introducao.js X
JS introducao.js > ... triplo
1 let triplo = a => 3*a
2 console.log(triplo(5))

PROBLEMAS SAÍDA CONSOLE DE DEPURACÃO TERMINAL PORTAS
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\introducao.js
15
PS C:\Users\ProfFernando\Desktop\aula01-js> █
```

Segunda forma

Função invocada


```
JS introducao.js X
JS introducao.js
1 //função invocada
2 (()=>{ console.log("olá mundo")})();
```

Funções com parâmetros rest

Utilizada quando não sabemos a qtd de valores que será passada por parâmetro

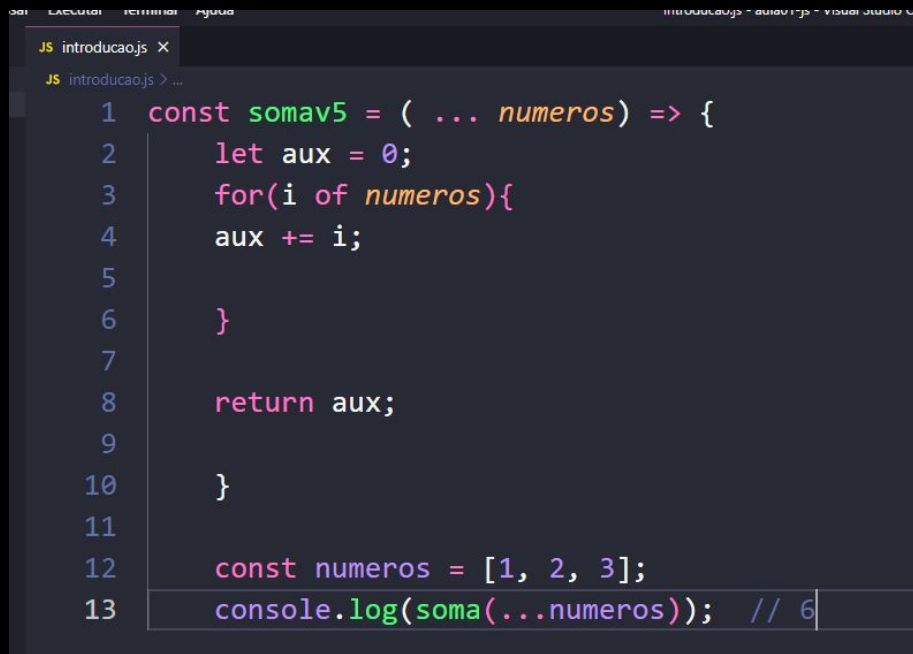
JS introducao.js X

JS introducao.js > ...

```
1  const somav5 = ( ... numeros ) => {  
2      let aux = 0;  
3      for(i of numeros){  
4          aux += i;  
5  
6      }  
7  
8      return aux;  
9  
10 }  
11  
12  console.log("REST:", somav5(2,2))
```

Funções com spread syntax

Usado para "espalhar" ou expandir os itens de um array ou objeto em uma expressão



```
1 const somav5 = ( ... numeros ) => {  
2   let aux = 0;  
3   for(i of numeros){  
4     aux += i;  
5  
6   }  
7  
8   return aux;  
9  
10 }  
11  
12 const numeros = [1, 2, 3];  
13 console.log(soma(...numeros)); // 6
```

No exemplo acima, o `...numeros` "espalha" o conteúdo do array `numeros` como argumentos individuais na função `soma`.

Construa funções para somar, subtrair, dividir e multiplicar, mas cada uma delas só pode ocupar uma única linha.

JS introducao.js X

JS introducao.js > ...

```
1  const add = (a, b) => a+b;
2  const subtr = (a, b) => { return a-b; }
3  const mult = (a, b) => a*b;
4  const divi = (a, b) => a/b
5
6  console.log("Soma:", add(2,2))
7  console.log("Subtração:", subtr(10, 7))
8  console.log("Multiplicação:", mult(5, 4))
9  console.log("Divisão:", divi(10,2))
```

Operador Ternário

É uma forma concisa de escrever uma expressão condicional if-else.

```
condição ? valor_se_verdadeiro : valor_se_falso;
```

Operador Ternário

É uma forma concisa de escrever uma expressão condicional if-else.

```
x = 8

if (x % 2 == 0) {
  resultado = 'par'
}
else {
  resultado = 'impar'
}

console.log(resultado)
```

Normal

```
x = 8

const resultado = x % 2 == 0 ? 'par' : 'impar'
console.log(resultado)
```

Com operador ternário

Filter

Como o próprio nome diz, utilizamos quando desejamos realizar algum tipo de filtro, por exemplo, em array, obj, etc

```
// filter

meu_array = [1, 2, 3, 4, 5, 6, 7, 8, 9]

const par = i => i % 2 == 0

console.log(meu_array.filter(par))
```

Destructuring

O operador destructuring, extrair de dentro do objeto ou array, seu atributos ou elementos.

```
const pessoa = {
  nome: "Felipe",
  idade: 20,
  sobrenome: "Alves",
  endereco: {
    logradouro: "Aven. Paulista",
    numero: 5000
  }
}

//Esse seria o padrão que vc pegaria a informação
let nome = pessoa.nome
console.log(nome)

//Com o destructuring
const {sobrenome, idade} = pessoa
console.log(idade)

//Criando destructuring com identificadores
const {nome: n, idade: i} = pessoa
console.log(n)

//Pegando o logradouro
const {endereco: {logradouro}} = pessoa
console.log(logradouro)
```

Dúvidas?