

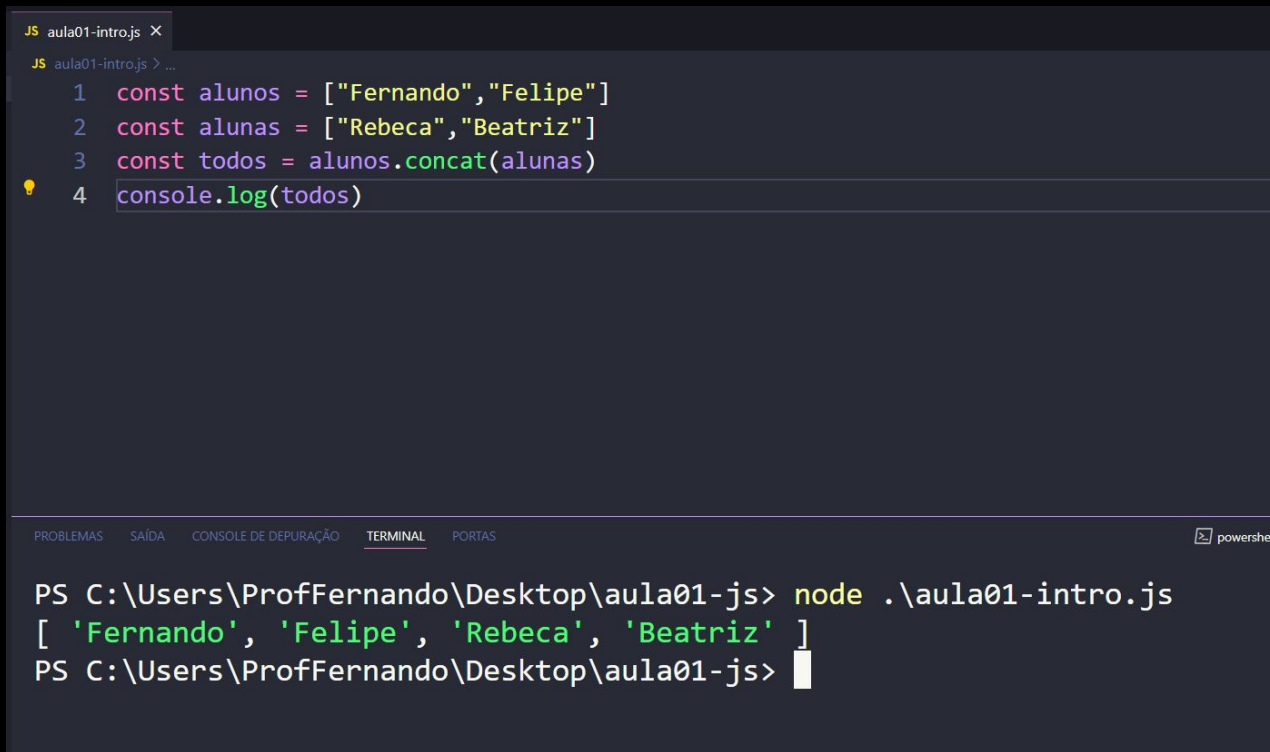
# Mobile Application Development

Prof. Fernando Pinéo

# Introdução JS - Parte II

# Função Concat

No JavaScript, a função concat é usada para combinar duas ou mais strings em uma única string.



```
JS aula01-intro.js x
JS aula01-intro.js > ...
1 const alunos = ["Fernando", "Felipe"]
2 const alunas = ["Rebeca", "Beatriz"]
3 const todos = alunos.concat(alunas)
4 console.log(todos)

PROBLEMAS SAÍDA CONSOLE DE DEPURACÃO TERMINAL PORTAS powershe

PS C:\Users\ProfFernando\Desktop\aula01-js> node .\aula01-intro.js
[ 'Fernando', 'Felipe', 'Rebeca', 'Beatriz' ]
PS C:\Users\ProfFernando\Desktop\aula01-js>
```

# Array em js

Array é heterogêneo, porém é recomendável utilizar dados do mesmo tipo no array.

js

```
alunos = ["Fernando", "Ana", "Rebeca"]  
console.log(alunos)
```

# Incluindo um novo elemento no array

## Método "push"

```
alunos = ["Fernando", "Ana", "Rebeca"]  
alunos.push("Jeferson")  
console.log(alunos)
```

# Ordenando o vetor com método sort

FIA/P

```
alunos = ["Fernando", "Ana", "Rebeca"]  
alunos.push("Jeferson")  
alunos.sort()  
console.log(alunos)
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

```
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\aula01-intro.js  
[ 'Ana', 'Fernando', 'Jeferson', 'Rebeca' ]
```

Saída - Terminal

# Deletando um item do vetor

```
alunos = ["Fernando", "Ana", "Rebeca"]  
delete alunos[1]  
console.log(alunos)
```

PROBLEMAS SAÍDA CONSOLE DE DEPUÇÃO TERMINAL PORTAS

```
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\aula01-intro.js  
[ 'Fernando', <1 empty item>, 'Rebeca' ]
```

Saída - Terminal

# Removendo o último item da lista - método pop()

```
alunos = ["Fernando", "Ana", "Rebeca"]  
alunos.pop()  
console.log(alunos)
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

```
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\aula01-intro.js  
[ 'Fernando', 'Ana' ]  
PS C:\Users\ProfFernando\Desktop\aula01-js> █
```



# Removendo o primeiro item da lista - método shift()

```
alunos = ["Fernando", "Ana", "Rebeca"]  
alunos.shift()  
console.log(alunos)
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

```
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\aula01-intro.js  
[ 'Ana', 'Rebeca' ]  
PS C:\Users\ProfFernando\Desktop\aula01-js> █
```

# Adicionando e removendo com o método splice

```
5  
alunos = ["Fernando", "Ana", "Rebeca"]  
alunos.splice(1,1)  
console.log(alunos)
```

Índice

Qtd a ser deletado a partir do índice

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

```
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\aula01-intro.js  
[ 'Fernando', 'Rebeca' ]
```

Saída - Terminal

# Duplicando um pedaço(slice) do array

FIAP

```
alunos = ["Fernando", "Ana", "Rebeca", "Pedro", "João"]  
algunsAlunos = alunos.slice(1,4)  
console.log(algunsAlunos)
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

```
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\aula01-intro.js  
[ 'Ana', 'Rebeca', 'Pedro' ]
```

Saída - Terminal

# For each

```
alunos = ["Fernando", "Ana", "Rebeca", "Pedro", "João"]
alunos.forEach(function(nome, indice) {
  console.log(nome, indice)
});
```

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS

```
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\aula01-intro.js
Fernando 0
Ana 1
Rebeca 2
Pedro 3
João 4
```

# Utilizando o filter

```
const carros=[
  {modelo:"Golf", preco:70000, ano:2015},
  {modelo:"Amarok", preco:110000, ano:2020},
  {modelo:"Toro", preco:120000, ano:2024},
  {modelo:"Creta", preco:150000, ano:2022},
]
console.log(carros.filter(function(e){
  return e.preco>110000
})))
```

PROBLEMAS

SAÍDA

CONSOLE DE DEPURAÇÃO

TERMINAL

PORTAS

```
[
  { modelo: 'Toro', preco: 120000, ano: 2024 },
  { modelo: 'Creta', preco: 150000, ano: 2022 }
]
```

Saída - Terminal

# Aplicando dois filters

```
const carros=[
  {modelo:"Golf", preco:70000, ano:2015,flex:false},
  {modelo:"Amarok", preco:110000, ano:2020,flex:false},
  {modelo:"Toro", preco:120000, ano:2024,flex:true},
  {modelo:"Creta", preco:150000, ano:2022,flex:true},
]

const caro = carro => carro.preco >=110000
const flex = carro => carro.flex

console.log(carros.filter(caro).filter(flex))
```

PROBLEMAS SAÍDA CONSOLE DE DEPURACÃO TERMINAL PORTAS

```
PS C:\Users\ProfFernando\Desktop\aula01-js> node .\aula01-intro.js
[
  { modelo: 'Toro', preco: 120000, ano: 2024, flex: true },
  { modelo: 'Creta', preco: 150000, ano: 2022, flex: true }
]
```

Saída - Terminal

Crie uma arrow function que receba dois números como parâmetros e retorne o produto desses números.

```
const multiplicar = (num1, num2) => num1 * num2;  
  
console.log(multiplicar(4, 5));
```



Crie uma função que receba dois números e retorne a soma, subtração, multiplicação e divisão desses números em um objeto.

```
function calcularOperacoes(a, b) {  
  return {  
    soma: a + b,  
    subtracao: a - b,  
    multiplicacao: a * b,  
    divisao: a / b  
  };  
}
```

```
console.log(calcularOperacoes(10, 5));
```

Crie uma função que receba uma lista de números e retorne o maior número dessa lista.

```
function maiorNumero(arr) {  
    return Math.max(...arr);  
}
```

```
console.log(maiorNumero([3, 5, 9, 1, 7]));
```

Crie uma função que receba um número e retorne "Positivo" se o número for maior que 0, "Negativo" se o número for menor que 0, e "Zero" se o número for igual a 0, usando o operador ternário.

```
function verificarNumero(num) {  
    return num > 0 ? 'Positivo' : num < 0 ? 'Negativo' : 'Zero';  
}  
  
console.log(verificarNumero(5));  
console.log(verificarNumero(-3));  
console.log(verificarNumero(0));
```

Crie uma função que receba um objeto com as propriedades nome, idade e cidade, e use destructuring para retornar apenas nome e cidade.

# Resposta



```
function extrairDados({ nome, cidade }) {  
  return `Nome: ${nome}, Cidade: ${cidade}`;  
}  
  
const pessoa = { nome: 'Carlos', idade: 30, cidade: 'São Paulo' };  
console.log(extrairDados(pessoa)); // Nome: Carlos, Cidade: São Paulo
```



Crie uma função que receba um array de objetos representando pessoas, cada um contendo nome, idade e profissao. Use destructuring para acessar as propriedades e filtre as pessoas que são maiores de idade e têm a profissão de "Desenvolvedor", retornando um novo array com esses dados.

```
[  
  {nome:"João",idade:25,profissao:"Desenvolvedor"},  
  {nome:"Maria",idade:17,profissao:"Designer"},  
  {nome:"Carlos",idade:30,profissao:"Desenvolvedor"},  
  {nome:"Ana",idade:22,profissao:"Desenvolvedor"},  
]
```

# Resposta

```
const pessoas = [
  { nome: 'João', idade: 25, profissao: 'Desenvolvedor' },
  { nome: 'Maria', idade: 17, profissao: 'Designer' },
  { nome: 'Carlos', idade: 30, profissao: 'Desenvolvedor' },
  { nome: 'Ana', idade: 22, profissao: 'Desenvolvedor' }
];

function filtrarDesenvolvedores(pessoas) {
  return pessoas.filter(({ idade, profissao }) => idade >= 18 && profissao === 'Desenvolvedor');
}

console.log(filtrarDesenvolvedores(pessoas));
```

Dúvidas?