Linguagens de Programação I

JavaScript - Parte 1

Linguagem JavaScript

Prof. Maurício Moreira Neto

Cursos
Sistemas da Informação
Análise e Desenvolvimento de Sistemas

Centro Universitário Christus

Unichristus

JavaScript

- É uma linguagem de programação interpretada
- Possui tipagem dinâmica
- Possui funções de primeira classe
- É case sensitive
- Suporte a programação assíncrona → código mais eficiente
- Manipulação da DOM
- É suportado tanto pelo lado cliente quanto pelo servidor (nodejs)

Como usar o JavaScript na aplicação web?

JavaScript dentro do HTML

- Utiliza-se a tag <*script*>

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Testing JS</title>
    <script>
        window.alert("Hello From JavaScript!")
    </script>
</head>
<body>
    <h2>Using JavaScript</h2>
    <p>Just click in the pop to see how JS is awesome!</p>
</body>
</html>
```

Unichristus

Chamando um arquivo JavaScript externo

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Testing JS</title>
    <script
src="js/hello.js"></script>
</head>
<body>
    <h2>Using JavaScript</h2>
    <p>Just click in the pop to see
how JS is awesome!</p>
</body>
</html>
```

hello.js

```
window.alert("Hello From
JavaScript");
```

Unichristus

Vamos fazer nosso Hello World em JS

```
console.log("Hello World from JS!")
```

E como iremos executar esse código?

1. No console do navegador (não muito recomendado para códigos extensos)
2. Softwares que iremos utilizar
   a. NodeJs
   b. Visual Studio Code
   c. Vamos utilizar os seguintes plugins
      i. JavaScript (ES6) Code Snippets
      ii. Code Runner

Criando um projeto Node

1. Crie uma pasta do projeto

2. Abra o terminal →Vá até a pasta do projeto

3. Digite no terminal
   a. **npm init -y** (Será criado o arquivo package.json)

4. Para executar o arquivo js basta escrever no terminal
   a. **node nome_arquivo.js**

5. Se quiser adicionar algum pacote basta escrever no terminal
   a. **npm install nome_pacote**

use strict

- O JavaScript mantém algumas funcionalidades desativadas por padrão
  - Garante o funcionamento do código antigo

- Ativa as novas funcionalidades do ECMAScript 5 (ES5)

- A diretiva pode ser usada de duas formas:
  - "use strict"
  - 'use strict'

use strict

- **Browser**

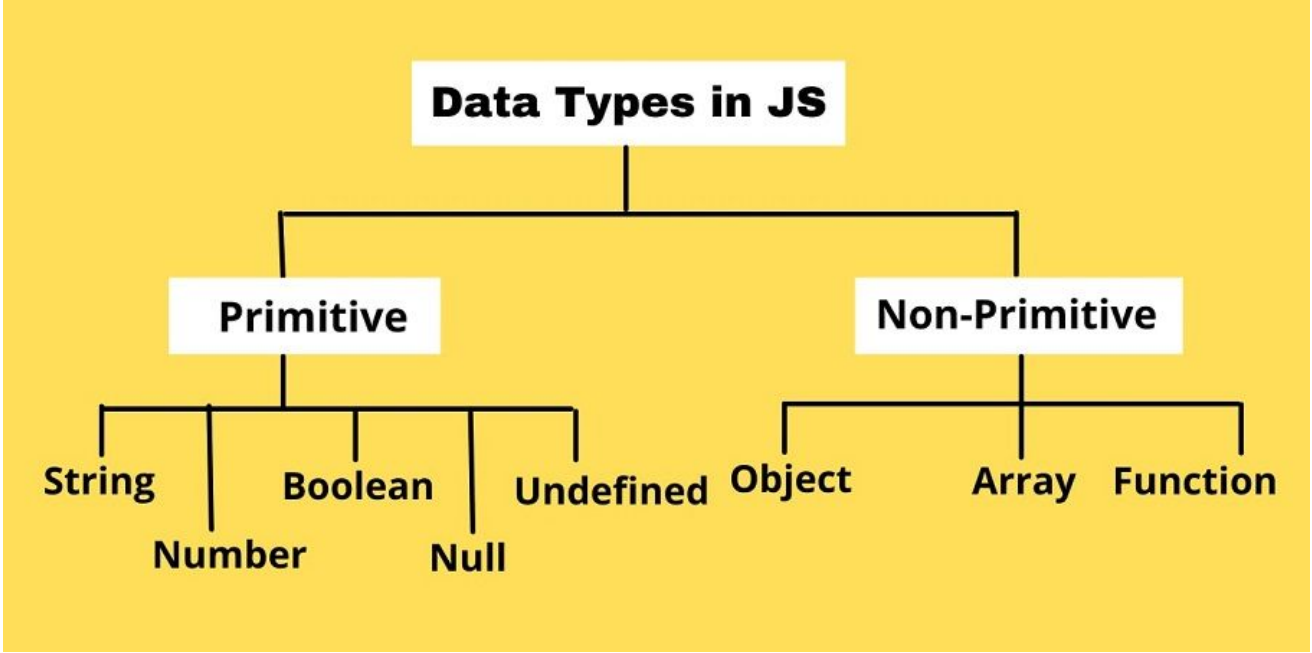
```
'use strict';
//  JS Code
<Enter to run>
```

ou

```
(function() {
  'use strict';
  // JS Code
})()
```

Tipos de Dados

- Number
- String
- Boolean
- Object
- Null
- Undefined



Usando o typeof

number	'number'
string	'string'
boolean	'boolean'
function	'function'
object	'object'
array	**'object'**
null	**'object'**
undefined	'undefined'

```
var a = "teste";
console.log(typeof a); // string
console.log(typeof 95.8); //number
console.log(typeof 5); // number
console.log(typeof false); // boolean
console.log(typeof true); //boolean
console.log(typeof null); // object
var b;
console.log(typeof b); //undefined
```

Declarando Variáveis

- Utiliza-se a palavra reservada `var` para definir variáveis

- Em JavaScript, as variáveis são fracamente tipadas, ou seja, o tipo é definido a partir de uma atribuição

- **Exemplos**

```
var x = 5; // declaração de variáveis com atribuição de
valor
var y; // declaração de uma variável sem atribuição
y = 2; // atribuicão
console.log(x + y);
```

Declarando Variáveis

- `var variable_name`

- **Number**
  - `var fnumber = 2;`
  - `fnumber = 3 + 2.1;`
  - `fnumber = 28 % 6;`

- **Object Math**
  - `d = Math.PI * r * r;`

Declarando Variáveis

- `var` x `let` x `const`

- `let`
  - Devem ser declaradas antes do uso
  - Possuem nível de bloco
  - Não podem ser re-declaradas

- `const`
  - Não podem ser re-declaradas
  - Não podem ser re-atribuídas
  - Possuem nível de bloco

Declarando Variáveis

- **let**

```
let x = “Mauricio Neto”;
let x = 0;
// Error: x já foi declarado

{
    let num = 10;
}
// num não pode ser visto fora do bloco
```

Declarando Variáveis

- const

```
const PI = 3.1415926;
PI = 3.14; // Erro: não pode ser feito essa
atribuição
PI = PI + 10; // Erro: não pode ser feito
essa atribuição
```

Exercício

- Qual a saída do seguinte código?

```
for (let i = 0; i < 10; i++) {
    console.log(i);
}
console.log("value i = " + i);
```

Exercício

- E se trocássemos o **let** pela **var**? o erro continuaria?

```
for (var i = 0; i < 10; i++) {
    console.log(i);
}
console.log("value i = " + i);
```

Convertendo para numéricos

- **parseInt**
  - Conversão de string para número (define a base):
    - `parseInt("123", 10);`
      - `123`
    - `parseInt("010", 10);`
      - `10`
    - `parseInt("11", 2);`
      - `3`

- **parseFloat**
  - Conversão de string para número flutuante
    - `parseFloat("123");`
      - `123`

Conversão de Tipos

- `var textoInteiro = "10";`
- `var inteiro = parseInt(textoInteiro);`
- `var textoFloat = "10.22";`
- `var numberfloat = parseFloat(textoFloat);`
- `var milNumber = 1000;`
- `var milString = milNumber.toFixed(2); // recebe o retorno da função`
- `console.log(milString); // imprime a string "1000.00"`

Not a Number (NaN) e Infinity

- `parseInt("hello", 10);`
  - NaN
- `NaN + 5;`
  - NaN
- `isNaN(NaN);`
  - true
- `1 / 0;`
  - Infinity

Unichristus

String

- O objeto String trabalha com uma série de caracteres
- JS converte automaticamente entre string primitiva e Objetos String
- Criando um objeto String:

```
var obj_str = new String(string);
```

Template String

- São strings literais que permitem expressões embutidas
- Utiliza-se acentos graves (ou *back-ticks*)
- **Exemplo**

```
var greeting = `Hello, Good Evening!`;
```

- Utiliza-se **${}** para inserir as expressões ou variáveis

```
var name = "Maurício";
console.log(`Hello, Good Evening, Mr. ${name}.`);
```

Template String

- Expressões

```
var fnumber = 20;
var snumber = 44;
let resultMessage = `The sum of ${fnumber} and
${snumber} is ${fnumber + snumber}`;
console.log(resultMessage);
```

- Também é possível utilizar em múltiplas linhas

```
let name = "Maurício"
let message = `Hi, Mr. ${name}!
How are you?
I hope you are fine.
Thank you for your attention.`
console.log(message);
```

Unichristus

String

String	`length`	Returns the number of characters in a string
	`concat( )`	Joins two or more strings
	`indexOf( )`	Returns the position of the first occurrence of a specified string value in a string
	`lastIndexOf( )`	Returns the position of the last occurrence of a specified string value, searching backward from the specified position in a string
	`match( )`	Searches for a specified string value in a string
	`replace( )`	Replaces some characters with others in a string
	`slice( )`	Extracts a part of a string and returns the extracted part in a new string
	`split( )`	Splits a string into an array of strings
	`substring( )`	Extracts the characters in a string between two specified indexes
	`toLowerCase( )`	Displays a string in lowercase letters
	`toUpperCase( )`	Displays a string in uppercase letters

Retirado de: MDN JavaScript String

String

- `"hello".length;`
  - `5`
- `"hello".charAt(0);`
  - `h`
- `"hello, world".replace("hello", "good night");`
  - `good night, world`
- `"hello".toUpperCase()`
  - `HELLO`

Exercício

- Faça um programa em JavaScript que receba uma frase do usuário e coloque em letras maiúsculas.
  - **1. Dica:** use a biblioteca prompt-sync do nodejs (https://www.npmjs.com/package/prompt-sync).
  - **2. Dica:** para usarmos as bibliotecas do node precisamos iniciar o projeto com **npm init -y** (no terminal)
  - **3. Dica:** após criar o package.json, digite no terminal **npm install prompt-sync**

Boolean

- **true** ou **false**
- **false**: 0 - "" - NaN - null - undefined
- 0 restante é **true**
- Operações: &&, || e !

Operadores

- **Numéricos:** +, -, *, / e %
- **Atribuição:** =, +=, -=, *=, /=, %=
- **Incremento/decremento:** a++, ++a, b--, --b
- **Concatenação de string**
  - "hello" + " world"
    - hello world
- **Coerção de tipos:**
  - "3" + 4 + 5 -> 345
  - 3 + 4 + "5" -> ?

Operadores

- **Numéricos:** +, -, *, / e %
- **Atribuição:** =, +=, -=, *=, /=, %=
- **Incremento/decremento:** a++, ++a, b--, --b
- **Concatenação de string**
  - "hello" + " world"
    - hello world
- **Coerção de tipos:**
  - "3" + 4 + 5 -> 345
  - 3 + 4 + "5" -> 75

Comparação

- **Para números e strings:** <, >, <= e >=
- **Igualdade:** == e !=
  - Faz conversão de tipos se necessário
    - "dog" == "dog" -> true
    - 1 == true -> true
- **Identidade:** === e !==
  - Não faz conversão de tipos
  - Se forem de tipos diferentes, o resultado será falso
    - 1 === true -> false
    - true === true -> true

Operadores Condicionais e Lógicos

Operador	Descrição	Tipo
==	Comparação de igualdade	Comparação
===	Comparação de igualdade de valor e tipo	Comparação
!=	Comparação de desigualdade	Comparação
!==	Comparação de desigualdade de valor ou de tipo	Comparação
>	Maior que	Comparação
<	Menor que	Comparação
>=	Maior ou igual	Comparação
<=	Menor ou igual	Comparação
?	Operador ternário	Comparação
&&	“E” and lógico	Lógico
||	“OU” or lógico	Lógico
!	Negação	Lógico

Estrutura de Decisão - if

Condição composta

```
if ( condição1 ) {
  // instruções caso verdade
} else {
  // instruções caso falso
}
```

if - else - if

```
if ( condição1 ) {
  // instruções A
} else if( condição2 ) {
  // instruções B
} else {
  // instruções C
}
```

Unichristus

Estrutura de Decisão - if

- **Exemplo**

```
var name = "mauricio";
if (name.toLowerCase() == "mauricio") {
   console.log("you can pass");
} else if(name.toLowerCase() == "admin") {
   console.log("you can access with high privileges");
} else {
   console.log("You can not pass");
}
```

Exercícios

- Faça um programa que receba um valor ou um conjunto de valores e verifique se cada número é par ou ímpar. O programa deve exibir:
  - 3 is odd
  - 4 is even

- Pode ser feito com entrada de dados pelo prompt-sync ou com um vetor estático inicial.

Exercício

- Escreva um programa em JavaScript que aceite dois inteiros e mostre o maior valor.
  - **Observação:** utilize a função de casting *parseInt(value, 10)*
    - https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/parseInt

Exercício (Resposta)

```html
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible"
content="IE=edge">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>First Exercise with JavaScript</title>
</head>
<body>
   <h1>Which the bigger value?</h1>
   <label for="fvalue">First Value:</label>
   <input type="number" id="fvalue">
   <br>
   <label for="svalue">Second Value:</label>
   <input type="number" id="svalue">
   <br/>
   <button onclick="biggerThan()">Click Here!</button>
   <p id="result">...</p>
</body>
```

```html
<script>
   function biggerThan() {
       var output = "the bigger number is ";
       fnumber =
document.getElementById("fvalue").value;
       snumber =
document.getElementById("svalue").value;
       result = document.getElementById("result");
       console.log(snumber);
       console.log(fnumber);
       if (parseInt(fnumber, 10) > parseInt(snumber,
10)) {
           result.innerText = output + fnumber;
       }  else {
           if (parseInt(snumber, 10) > parseInt(fnumber,
10)) {
               result.innerText = output + snumber;
           } else {
               result.innerText = "They are equals";
           }
       }
   }
</script>
</html>
```

Unichristus

Exercício

- Vamos fazer um alarme o qual é possível configurar hora-minuto e a mensagem que será exibido quando o tempo determinado for atingido

- **Observação:** crie uma função para verificar os valores de entrada.

- Iremos utilizar o método setInterval para criarmos o nosso alarme!
  - https://developer.mozilla.org/en-US/docs/Web/API/setTimeout

Exercício (Resposta)

```
function isNumber(input, message) {
     if (isNaN(input)) {
     window.alert(message);
     console.log(message);
     }
}

let setAlarm = setInterval(function() {
     var hour = document.getElementById("hour").value;
     isNumber(hour, "Hour value is not applicable!");
     var minutes = document.getElementById("minutes").value;
     isNumber(minutes, "Minutes value is not applicable!");
     var message = document.getElementById("message").value;

     if (new Date().getHours() == hour && new Date().getMinutes() == minutes) {
     document.getElementById("resultado").innerHTML = message;
     window.alert(message);
     }
}, 1000);
```

Unichristus

Exercício (Resposta)

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exercise JavaScript</title>
    <script>
        function isNumber(input, message) {
            if (isNaN(input)) {
                window.alert(message);
                console.log(message);
            }
        }
        let setAlarm = setInterval(function () {
            var hour = document.getElementById("hour").value;
            isNumber(hour, "Hour value is not applicable!");
            var minutes = document.getElementById("minutes").value;
            isNumber(minutes, "Minutes value is not applicable!");
            var message = document.getElementById("message").value;

            if (new Date().getHours() == hour && new Date().getMinutes() == minutes) {
                document.getElementById("resultado").innerHTML = message;
                window.alert(message);
            }
        }, 1000);
    </script>
</head>
<body>
    <h1>My JavaScript Alarm</h1>
    <form>
        <div>
            <label for="hour">Hour</label> <br />
            <input type="text" id="hour" />
        </div>
        <div>
            <label for="minutes">Minutes</label> <br />
            <input type="text" id="minutes" />
        </div>
        <div>
            <label for="message">Message</label> <br /  >
            <textarea name="message" id="message" cols="40" rows="20"></textarea>
        </div>
    </form>
    <button onclick="setAlarm()">Setup</button>
    <div id="resultado"></div>
</body>
</html>
```

Unichristus

Estrutura de Decisão - switch

```
switch(action) {
    case 'example1':
        // comandos
        break;
    case 'example2':
        // comandos
        break;
    default:
        // comandos do caso default
        break;
}
```

Estrutura de Decisão - Operador Ternário

- Normalmente utilizado para substituir uma condicional composta
  - `var allowed = (age > 18) ? "yes" : "no";`

Exercícios

- Faça um programa em JS que verifique o sinal de um valor recebido. Por exemplo: 4 -5 -3 6
  - Saída:
    - 4: sinal +
    - -5: sinal -
    - -3: sinal -
    - 6: sinal +

- O dado por ser recebido pelo terminal ou dado por um vetor.

Estruturas de Repetição

Sem variável de controle

```
while (expressão) {
    // comandos
}

do {
    // comandos
} while (expressão)
```

Com variável de controle

```
for (var i = 0; i < 5; i++) {
    // comandos
}
```

Exercício

Faça um programa usando a linguagem JavaScript que cria uma função que receba dois valores e diga qual os valores pares entre esses dois números

Exercício (Resposta)

```
function valores_pares(inic, fim) {
    for (var i = inic; i <= fim; i++) {
        if (i % 2 == 0) {
            console.log('Par: ' + i);
        }
    }
};
valores_pares(1, 10);
```

Tratamento de Exceções

```
try {
  Bloco de Código do try
}
catch(err) {
  Bloco de código para tratar o erro
}
finally {
  Bloco de código que será executado independente do
resultado do try / catch
}
```

Exercício

- Crie uma função chamada calculadora que recebe dois números e uma operação como parâmetros. A função deve realizar a operação matemática correspondente (adição, subtração, multiplicação ou divisão) e retornar o resultado. No entanto, o código deve ser robusto o suficiente para tratar possíveis erros, como divisão por zero ou operação inválida.

Exercício (Resposta)

```
function calculadora(num1, num2, operacao) {
  if (typeof num1 !== 'number' || typeof num2 !== 'number') {
    throw new Error('Os parâmetros num1 e num2 devem ser números.');
  }
  switch (operacao) {
    case '+':
      return num1 + num2;
    case '-':
      return num1 - num2;
    case '*':
      return num1 * num2;
    case '/':
      // Trata a divisão por zero
      if (num2 === 0) {
        throw new Error('Divisão por zero não é permitida.');
      }
      return num1 / num2;
    default:
      throw new Error('Operação inválida.');
  }
}
// Exemplo de uso:
try {
  const resultadoSoma = calculadora(10, 5, '+');
  console.log('Resultado da Soma:', resultadoSoma);
  const resultadoDivisao = calculadora(10, 0, '/');
  console.log('Resultado da Divisão:', resultadoDivisao);
} catch (error) {
  console.error('Erro:', error.message);
}
```

Unichristus

Objetos

- Estruturas que possuem propriedades e métodos
- Simples pares key-value

- **Criação de objetos:**
  - `var obj = new Object();`
  - `let obj = {};`

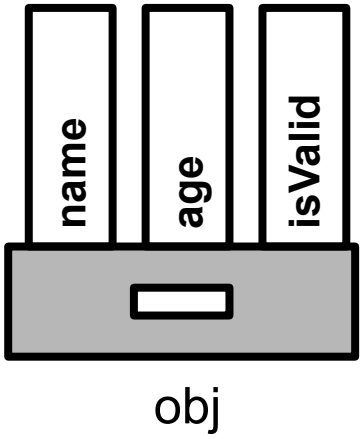
- **Manipulação de objetos**
  - `obj.name = "Mauricio"`
    OU
  - `obj["name"] = "Mauricio";`

Iteração de um Objeto

- **Pode-se iterar pelas chaves de um objeto**

```
var obj = { 'name': 'mauricio', 'age': 32 };
obj.isValid = true;

for (attr in obj) {
   console.log (attr + ' = ' + obj[attr]);
}
```



Exercício

1. Crie um objeto chamado pessoa com as seguintes propriedades: nome, idade e profissão. Em seguida, imprima no console uma mensagem no seguinte formato: "Olá, meu nome é {nome}, tenho {idade} anos e sou {profissao}."

2. Crie um objeto chamado estoque que armazene informações sobre produtos disponíveis em uma loja. Cada produto deve ter as propriedades nome, preço e quantidade. Em seguida, crie uma função chamada verificarEstoque que recebe o nome de um produto como parâmetro e retorna uma mensagem informando se o produto está disponível em estoque e quantas unidades estão disponíveis.

Arrays

- Tipo especial de objeto: as chaves são números e não strings.
- Sintaxe [ ]:

```
let/var a = new Array();
a[0] = "dog";
a[1] = "cat";
a[2] = "horse";
a.length
   = 3
```

Arrays - Uso em Filas (Queue)

- A fila é um dos casos mais comuns de um array

- Coleção de dados ordenado que suporta duas operações:
  - **push** - Adiciona um elemento ao final
  - **shift** - retorna um elemento do começo, avançando a fila, tornando o segundo elemento no primeiro

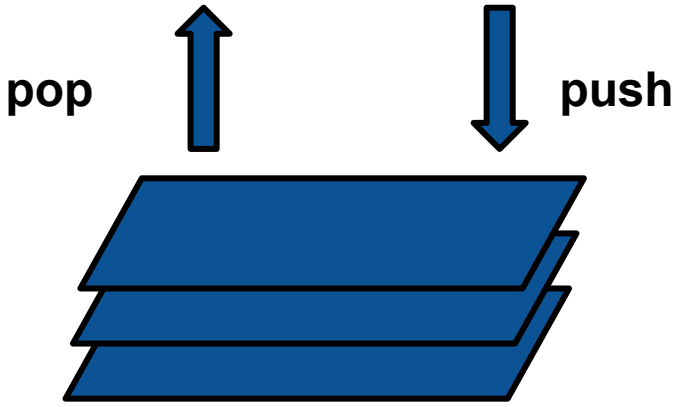
Adicionando Elementos no Array

```
var palavras = ["unichristus", "programming"];
palavras.push("javascript");
// adiciona a string "Javascript"
console.log(palavras);
= ["programming", "javascript"];
palavras.shift();
= ["programming", "javascript"];
```

Arrays - Uso em Pilhas (Stack)

- A pilha é outro caso comum de um array

- Suporta duas operações:
  - **push** - Adiciona um elemento ao final
  - **pop** - retorna o elemento do final



Adicionando Elementos no Array

```
var palavras = ["unichristus", "programming"];
palavras.push("javascript");
// adiciona a string "Javascript"
console.log(palavras);
= ["unichristus", "programming", "javascript"];
palavras.pop();
= ["unichristus", "programming"];
```

Iteração em um Array

```
for (var i = 0; i < array.length; i++) {
    // acesse os elementos com índice array[i]
}

for (let item of array) {
    // acesse o item com vetor array
}
```

Arrays Javascript - Diversos tipos

```
var example = ["Hello", true, 123, {name: "mauricio"},
function(){console.log("Hello From Array");}]

example[4]();
```

- Resultado
  "Hello From Array"

Métodos sobre Arrays - forEach

- **Sintaxe**

```
myArray.forEach(function(currentValue, index, arr), thisValue)
```

  - *`function()`* - Será a função que executará a cada elemento do array
  - *`currentValue`* - Valor atual do elemento
  - *`index`* - O índice do elemento atual
  - *`arr`* - array do elemento atual
  - *`thisValue`* - [opcional] valor passado para a função como valor do operador *this*

Exercício

Crie uma página HTML simples com um botão e uma lista não ordenada (<ul>) contendo alguns itens. Use JavaScript para adicionar um evento ao botão que, ao ser clicado, deve alterar o texto de todos os itens da lista para "Item modificado". Use a função *forEach* para iterar pelos elementos da lista e alterar o texto de cada item.

Resposta

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Manipulação da DOM com forEach</title>
</head>
<body>
    <button id="btnModificar">Modificar Itens da Lista</button>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
        <li>Item 4</li>
        <li>Item 5</li>
    </ul>
    <script src="script.js"></script>
</body>
</html>
```

Resposta

```
// Selecione o botão e a lista
const btnModificar = document.getElementById('btnModificar');
const listaItens = document.querySelectorAll('ul li');

// Adicione um ouvinte de evento ao botão
btnModificar.addEventListener('click', () => {
    listaItens.forEach(item => {
        item.textContent = 'Item modificado';
    });
});
```

Métodos sobre Arrays - Filter

- Método que cria um novo array preenchido com os elementos que passam pela condição da função
- Este método não altera o array original

```
var ages = [32, 33, 16, 40];
function checkAdult(age) {
  return age >= 18;
}
console.log(ages.filter(checkAdult));
```

Exercício

Considere um conjunto de números inteiros. Crie um programa em JavaScript que recebe um array de números inteiros e retorna um novo array contendo apenas os números pares. Implemente a função *selecionarNumerosPares(numeros)* que recebe um array de números inteiros como parâmetro e utiliza o método filter para criar um novo array contendo apenas os números pares.

Métodos sobre Arrays - Map

- Método cria um novo array com a chamada da função para cada elemento do array

- Não modifica o array original

```
var numbers = [65, 44, 12, 4];
var newArray = numbers.map(myFunction)
function myFunction(num) {
  return num * 10;
}
console.log(newArray);
```

Exercício

- Uma estação meteorológica registrou a temperatura em graus Celsius ao longo de um dia. Crie um programa em JavaScript que recebe um array de temperaturas em graus Celsius e retorna um novo array contendo as temperaturas equivalentes em graus Fahrenheit.
  - **A fórmula para converter de Celsius para Fahrenheit é: Fahrenheit = (Celsius * 9/5) + 32.**
- Implemente a função *converterCelsiusParaFahrenheit(tempCelsius)* que recebe um array de temperaturas em graus Celsius como parâmetro e utiliza o método map para criar um novo array contendo as temperaturas equivalentes em graus Fahrenheit.

Métodos sobre Arrays - Reduce

- Método aplicado em arrays para realizar operações de redução nos elementos

- A operação envolve a iteração pelos elementos de um array e a acumulação de um valor

- **Exemplo**

```
let value = arr.reduce(function(accumulator, item, index, array) {
// ...
}, [initial]);
```

Métodos sobre Arrays - Reduce

- **Argumentos**
  - **accumulator** - é o resultado da chamada da função anterior, igual a initial na primeira vez (se inicial for fornecida)
  - **item** - é o item atual da matriz
  - **index** - é a sua posição
  - **array** - é o seu array

- A medida que a função é aplicada, o resultado da chamada de função anterior é passada para a próxima como o primeiro argumento

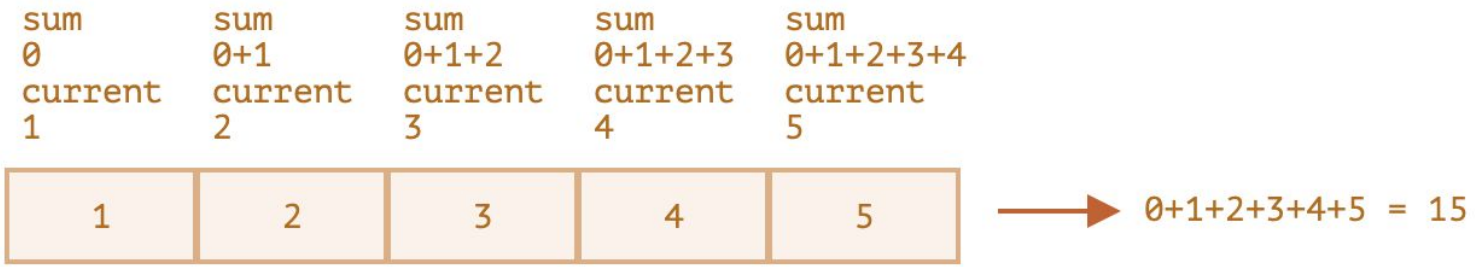
Exemplo de Reduce

- Este exemplo apresenta o uso do reduce para realizar o somatório dos valores de um array.

```
let arr = [1, 2, 3, 4, 5];

let result = arr.reduce((sum, current) => sum + current, 0);

console.log(result); // 15
```



Exercício

- Você tem um array de números inteiros. Sua tarefa é criar uma função em JavaScript chamada **encontrarMaiorValor** que utiliza o método reduce() para encontrar o maior valor no array.

Funções em JavaScript

Funções Nomeadas

```
function add(x, y) {
    var total = x + y;
    return total;
    }
```

- **Passagem de parâmetros:**

  add(2, 3) -> 5

Funções Anônimas

```
var somaDoisNumeros = function (numero1, numero2) {
    return numero1 + numero2;
};

somaDoisNumeros(10, 20);
```

Funções - Arguments

```
function add(...arguments) {
    var total = 0;
    for (var index = 0; index <=arguments.length; index++) {
        total += index;
    }
    return total;
}
console.log(add(1,2,3));
```

add(2, 3,4) -> 9

Funções de Seta (Arrow Functions)

- Introduzidos no ES6
- Permite criar funções de forma clara comparada a funções regulares
- **Sintaxe**

```
let myFunction = (arg1, arg2, ..., argN) => {
    // commands
}
```

Funções de Seta (Arrow Functions)

- **Exemplos**

```
let hello = () => console.log('Hello');
hello();
```

```
let soma = (a, b) => {
   let resultado = a + b;
   return resultado;
}
let x = soma(5, 5);
console.log(x);
```

JSON

- JSON (*JavaScript Object Notation*) é um formato geral para representar valores e objetos
  - Padrão RFC 4627

- Inicialmente feito para o JS porém…outras linguagens a utilizam por ser um formato de dados simples para ser enviado

- Ou seja, é bastante usado para troca de dados entre o cliente que usa o JS e o servidor que usa Java/Ruby/C#, …

JSON

- O JavaScript prove dois métodos para o JSON:
  - **JSON.stringify**
    - Converte objetos em JSON
  - **JSON.parse**
    - Converte JSON de volta a objetos

```
let user = {
  name: 'Mauricio',
  age: 33,
  isProfessor: true,
  courses: ['programming 1', 'programming 2', 'mathematical logic 1'],
};
let user_json = JSON.stringify(user);
console.log('Data type: ' + typeof user_json);
console.log(user_json);
```

JSON

```
let value = JSON.parse(str, [function]);
```

- **str:** JSON a ser convertido em string
- **function:** Opcional! Função (chave, valor) que será chamada por cada par de chave-valor e que pode transformar o valor

```
let userJSON = '{ "name": "Mauricio", "age": 33, "isProfessor": true,
"courses": ["programming 1", "programming 2", "mathematical logic 1"]
}';

let userObject = JSON.parse(userJSON);

console.log(userObject.name); // mauricio
console.log(userObject.courses[2]); //mathematical logic 1
```

JSON

Exemplo de uso de uso da função do parse:

```
let schedule = `{
"meetups": [
  { "title": "Conference 1", "date": "2017-11-30T12:00:00.000Z" },
  { "title": "Conference 2", "date": "2017-04-18T12:00:00.000Z" }
]
}`;
schedule = JSON.parse(schedule, function(key, value) {
  if (key == 'date') return new Date(value);
  return value;
});
console.log("Date: " + schedule.meetups[1].date.getDate());
```

Exercício

- Um aplicativo de gerenciamento de tarefas armazena informações sobre as tarefas dos usuários em formato JSON. Crie um programa em JavaScript que recebe um array contendo objetos JSON representando tarefas. Cada objeto possui os campos titulo, descricao e concluida. O programa deve retornar um novo array contendo os títulos das tarefas concluídas.
    - Implemente a função *obterTitulosTarefasConcluidas(tarefas)* que recebe um array de objetos JSON de tarefas como parâmetro e utiliza manipulação de JSON para criar um novo array contendo os títulos das tarefas que estão marcadas como concluídas.

Obrigado pela atenção!

Dúvidas?

Contato

mauricio.moreira@unichristus.edu.br

Unichristus