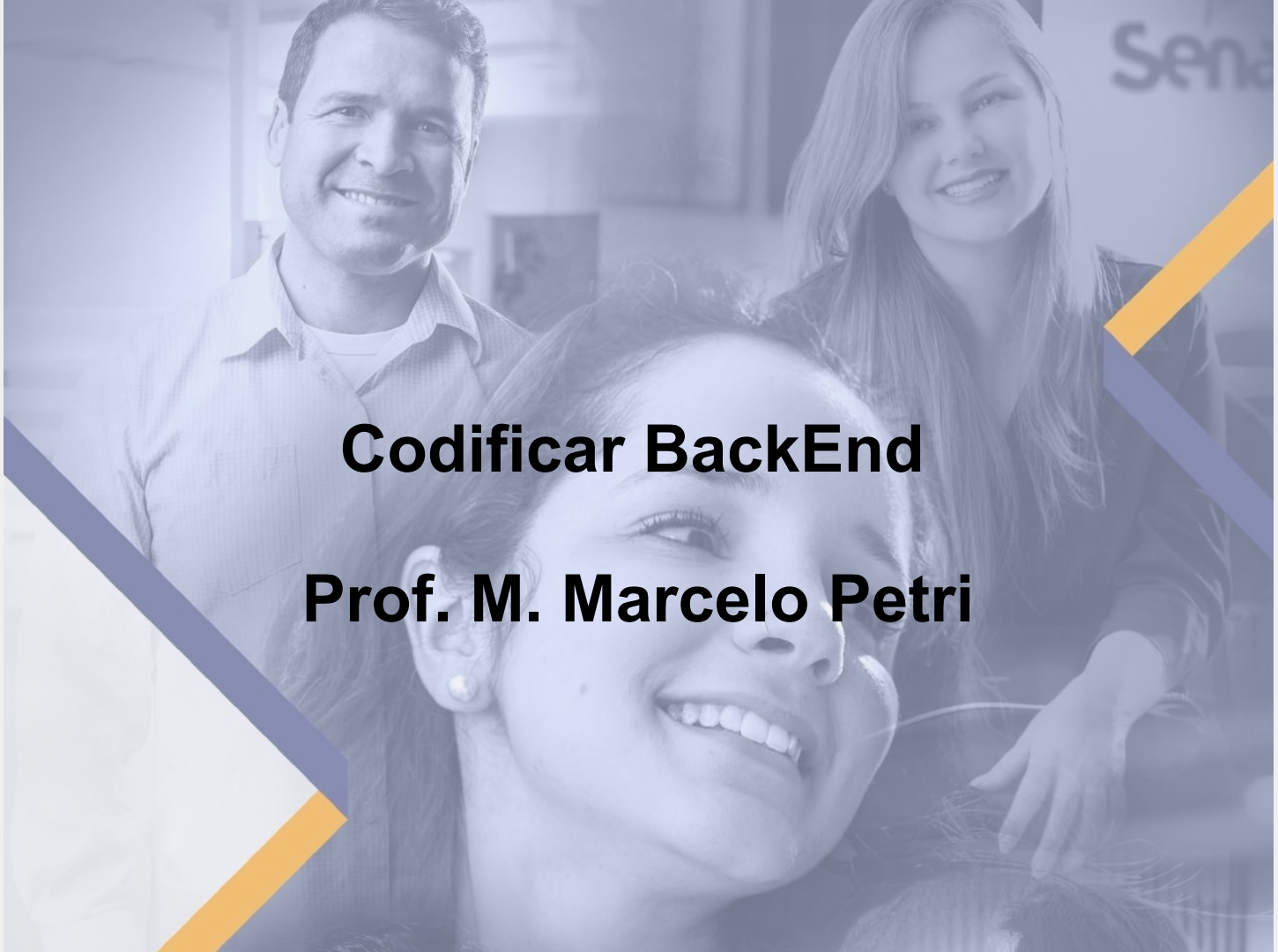


Codificar BackEnd

Prof. M. Marcelo Petri

Olá! Sejam bem vindos

DISCIPLINA/UNIDADE CURRICULAR	Codificar BackEnd
TEMA DA AULA	Introdução a Disciplina
INDICADOR	1 - Configura o ambiente de desenvolvimento conforme as funcionalidades e características do projeto
CRONOGRAMA	01 - Apresentação 02 - Revisão de JavaScript 03 - Orientação a Objetos / Classes
ATIVIDADE	Atividades Práticas
ORIENTAÇÕES	Respeite o horário de início e final de aula, guarde seu celular, não converse durante a explicação, levante a mão quando tiver dúvidas

APRESENTAÇÕES

Professor: Marcelo Petri

Formação

Experiência

E-mail: marcelo.petri@prof.sc.senac.br

APRESENTAÇÕES

PLANO DA UNIDADE CURRICULAR

COMUNICAÇÃO

Use o **CHAT do TEAMS**

E-mail da Professor: **marcelo.petri@prof.sc.senac.br**

Empregabilidade em TI

Verificação da Aprendizagem

EXERCÍCIOS

PESQUISAS

ESTUDOS DE CASOS

DESENVOLVIMENTO

Introdução

- JavaScript é uma linguagem de script orientada a objetos, multiplataforma.
- É uma linguagem pequena e leve.
- Dentro de um ambiente de host (por exemplo, um navegador web) o JavaScript pode ser ligado aos objetos deste ambiente para prover um controle programático sobre eles.
- Front End, Back End, Mobile

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Introduction

Atividade Revisão

Nosso programa deve controlar a lista de atendimentos dos pacientes de um consultório odontológico – como se fosse um painel em exposição em uma tv do consultório.

Atividade Revisão

- **Funções**
  - Função são blocos de construção fundamentais em JavaScript.
  - Uma função é um procedimento de JavaScript - um conjunto de instruções que executa uma tarefa ou calcula um valor.
  - Para usar uma função, você deve defini-la em algum lugar no escopo do qual você quiser chamá-la.

Atividade Revisão

- **Funções - Exemplos**

```
//Function Hoisting
function falaOi(){
    console.log("Oi");
}
falaOi();
```

```
//First-class Objects
const umDado = function(){
    console.log("Sou um dado.");
}
umDado();

function executaFuncao(funcao){
    funcao();
}
executaFuncao(umDado);
```

```
//Arrow Function
const funcaoArrow = () => {
    console.log("Sou uma Arrow
Function");
}
funcaoArrow();
```

Funções (Parâmetros)

- Argumentos que sustentam todos os argumentos enviados
- *Arguments* não funcionam em Arrow Functions

```
function funcao() {
    console.log(arguments);
}
funcao('Valor', 123, true);
```

```
function funcao(a, b, c) {
    let total = 0;
    for (let argumentos of arguments){
        total += argumentos;
    }
    console.log(total, a, b, c);
}
funcao(1, 2, 3, 4, 5, 6, 7, 8, 9);
```

```
function funcao(a, b, c=10) {
    let total = 0;
    for (let argumentos of arguments){
        total += argumentos;
    }
    console.log(total, a, b, c);
}
funcao(1, 2);
```

```
function funcao(a, b=2, c=4) {
    console.log(a + b + c);
}
funcao(1, '', 2);
funcao(1, null, 2);
funcao(1, undefined, 2);
```

Funções Fábricas (Factory Function)

```
function criaPessoa(nome, sobrenome, alt, peso){
    return {
        nome,
        sobrenome,
        fala(assunto) {
            return `${this.nome} está ${assunto}.`;
        },
        altura: alt,
        peso: peso,
        imc() {
            const indice = this.peso / (this.altura**2);
            return indice.toFixed(2);
        }
    };
}
const p1 = criaPessoa('Marcelo', 'Petri', 1.74, 78);
console.log(p1.imc());
```

Funções Construtora (Constructor Function) - Getter/Setter

```
function criaPessoa(nome, sobrenome, alt, peso){
    return {
        nome,
        sobrenome,
        fala(assunto) {
            return `${this.nome} está ${assunto}.`;
        },
        altura: alt,
        peso: peso,
        get imc() {
            const indice = this.peso / (this.altura**2);
            return indice.toFixed(2);
        }
    };
}
const p1 = criaPessoa('Marcelo', 'Petri', 1.74, 78);
console.log(p1.imc);
```

Funções Construtoras

- Cria novos objetos similar a função fábrica;

```
function Pessoa (nome, sobrenome){
    const ID=123; //Privados
    const metodoInterno = function(){
        console.log('Método interno');
    }
    //Publicos
    this.nome = nome;
    this.sobrenome = sobrenome;
    this.metodo = function(){
        console.log('Sou um método!');
    }
}

const p1 = new Pessoa('Marcelo', 'Petri');
console.log(p1);
p1.metodo();
```

Funções (Retorno)

- Return => retorna um valor e termina a função;

```
function soma(a, b){
    return a + b;
}
let s1 = soma(2, 5);
console.log(s1);
```

```
function criaPessoa(nome, sobrenome){
    return { nome: nome, sobrenome: sobrenome};
}
const p1 = criaPessoa('Marcelo', 'Petri');
```

- Não retorna nada

```
document.addEventListener('click', function (){
    document.body.style.backgroundColor = 'red';
});
```

Arrays - Básico

```
1   //                0         1         2
2   const nomes = ['Marcelo', 'Maria', 'João'];
3   nomes[2] = 'João'; //adiciona conteudo no indice
4   delete nomes[2]; //apaga o conteudo do indice
5   const removido = nomes.pop(); //Remove do final
6   const removido1 = nomes.shift(); //Remove do inicio
7   nomes.push('Petri'); // Adiciona no final
8   nomes.unshift('Petri'); // Adiciona no inicio
9   const nome = nomes.join(', '); //Array em String
10  const nomes1 = nome.split(', '); //String em Array
11
12  console.log(nomes1);
13
14  // Outra forma de criar array
15  const nomes = new Array('Marcelo', 'Maria', 'João');
```

Objetos

```js
const pessoa = {
    nome: 'Marcelo',
    sobrenome: 'Petri'
};
//Notação de ponto
console.log(pessoa.nome);
console.log(pessoa.sobrenome);

//Notação de conchetes e chaves
const chave = 'nome'
console.log(pessoa[chave]);
console.log(pessoa['sobrenome']);
```

```js
const pessoa = new Object();
pessoa.nome = 'Marcelo';
pessoa.sobrenome = 'Petri';
pessoa.idade = 44;
pessoa.falarNome = function(){
    console.log(`${this.nome} falou seu nome.`);
}
pessoa.getDataNascimento = function(){
    const dataAtual =  new Date();
    return dataAtual.getFullYear() - this.idade;
}

pessoa.falarNome();
console.log(pessoa.getDataNascimento());
console.log(pessoa);
```

Objetos

```
1   //Factory functions
2   function criaPessoa(nome, sobrenome){
3       return {
4           nome,
5           sobrenome,
6           nomeCompleto(){
7               return `${this.nome} ${this.sobrenome}`
8           }
9       }
10  }
11  const p1 = criaPessoa('Marcelo', 'Petri');
12  console.log(p1.nomeCompleto());
```

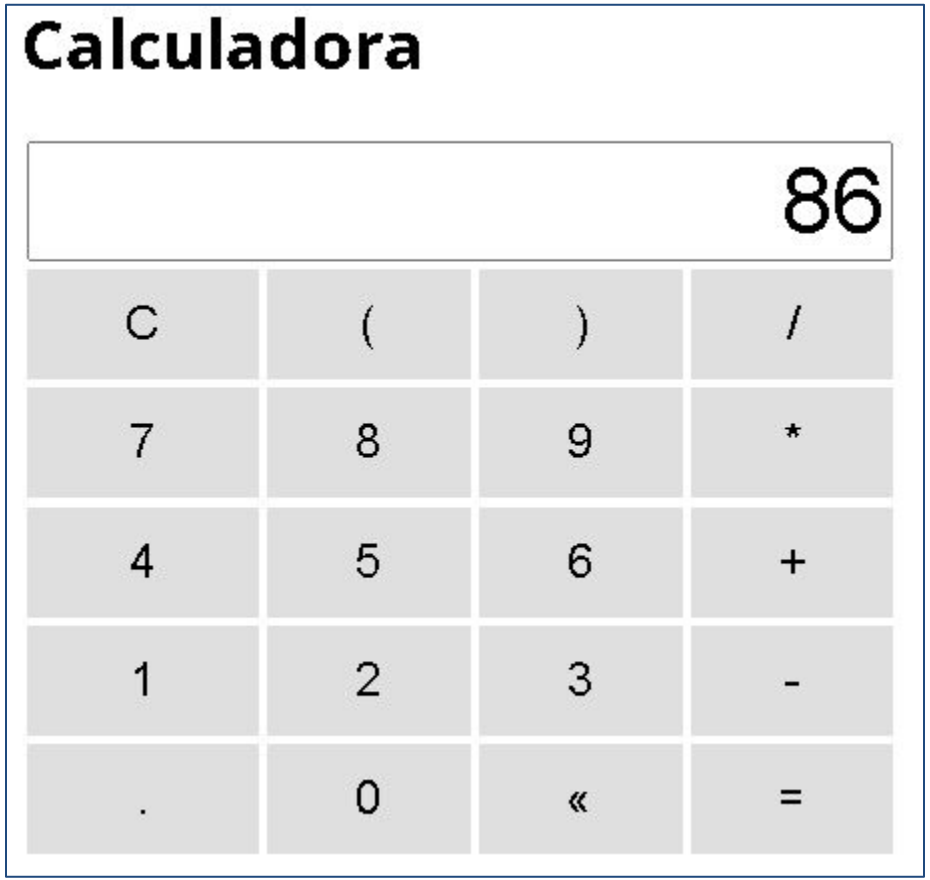
```
1   // Constructor functions
2   function Pessoa(nome, sobrenome){
3       this.nome = nome;
4       this.sobrenome = sobrenome;
5   }
6   const p2 = new Pessoa('Marcelo','Petri');
7   console.log(p2);
```

Getters e Setters

```
function Produto(nome, preco, estoque){
    this.nome = nome;
    this.preco= preco;

    let estoquePrivado = estoque;
    Object.defineProperty(this, 'estoque', {
        enumerable: true, //mostrar
        configurable: true,//configuravel
        get: function(){
            return estoquePrivado;
        },
        set: function(valor){
            if (typeof valor !== 'number'){
                throw new TypeError('Mensagem erro');
            }
            estoquePrivado = valor;
        }
    });

}

const p1 = new Produto('Camisa', 20, 3);
p1.estoque = 'abc'
console.log(p1);
```

```
function criaProduto(nome){
    return {
        get nome(){
            return nome;
        },
        set nome(valor){
            nome = valor;
        }
    };
}
const p1 = new criaProduto('Camisa');
console.log(p1.nome);
```

Atividades



Referências

Bibliografia Básica

MOLINARI, William. **Desconstruindo a Web**: As tecnologias por trás de uma requisição. São Paulo: Casa do Código, 2016.

MUELLER, John Paul. **Segurança para desenvolvedores web**. São Paulo: Novatec. 2016.

ARAÚJO, Everton Coimbra de. **ASP.NET Core MVC**: Aplicações modernas em conjunto com o Entity Framework. São Paulo: Casa do Código, 2018.

Bibliografia Complementar

ANDRADE, Sidney da Silva. **Aprenda Java EE8**: Aplicações para web com Spring MVC e Hibernate. São Paulo: Senai Editora, 2018.

DALL'OGLIO, Pablo. **PHP Programando com Orientação a Objetos**. 4. ed. São Paulo: Novatec, 2018.

DEITEL, Harvey; DEITEL, Paul. **Java**: Como Programar. 10.ed. São Paulo: Pearson, 2016.

POWERS, Shelley. **Aprendendo Node**: Usando JavaScript no Servidor. São Paulo: Novatec, 2017.