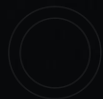

SHIFT

 FIAP



000000



FRONT-END SPECIALIST

DESENVOLVIMENTO WEB COM ANGULAR & REACT

LÓGICA DE PROGRAMAÇÃO

JAVASCRIPT



ALEX SANDER RESENDE DE DEUS

Professor

- Há 25 anos ensinando programação, é um apaixonado por tecnologia. Atualmente é coordenador de cursos na ETEC Albert Einstein do Centro Paula Souza.
- Na FIAP é professor da FIAP School e FIAP Corporate, lecionando C#, Banco de Dados e Desenvolvimento Mobile.

✉ profalex.deus@fiap.com.br

AGENDA

1

AULA 1

Definição e histórico

Fundamentos Básicos da Programação – Tipos de dados, variáveis e estruturas básicas; Usando o GIT

2

AULA 2

Estruturas condicionais, operadores lógicos e switch case

3

AULA 3

Laços de Repetição: For, while,

4

AULA 4

Objetos nativos JavaScript – Date, String, Math

AGENDA

5

AULA 5

Arrays, Filter, Map

Reduce, Splite, forEach, for in, for of

6

AULA 6

Funções: tipos, declarações, escopo, retorno, clousers, call-back, IIFE, factory

7

AULA 7

DOM – Document Object Model

8

AULA 8

DOM – Document Object Model

AGENDA

9

AULA 9

Eventos

Objetos: Criação, prototype, getters, setters



10

AULA 10



Classes JavaScript Assíncrono

AULA 9

POO - PROGRAMAÇÃO ORIENTADA A OBJETOS



A orientação a objetos nunca foi um ponto forte do Javascript, porém, esse contexto mudou um pouco com a chegada das classes, que funcionam como um **açúcar sintático** para a real POO (Programação Orientada a objetos) da linguagem, além do mais, isso traz uma sofisticação a sintaxe da linguagem, permitindo com que você crie projetos muito mais organizados e bem elaborados utilizando todos os recursos possíveis de POO.



Antes do **ES6**, você poderia criar estruturas parecidas com classes no Javascript, porém utilizando funções que deixava toda a semântica do código extremamente confusa. Agora, após a chegada do ES6, vamos ver como criar uma classe “de verdade” no javascript.

CRIANDO UMA CLASSE

Como em qualquer outra linguagem, basta começar digitando a palavra reservada `class` seguido do nome da classe.

```
JS script.js > ...  
1  class Heroi {  
2  
3  }  
4
```

DEFININDO ATRIBUTOS

Bem diferente de outras linguagens, onde você declara-os diretamente no corpo da classe, no JavaScript você deve declarar os atributos dentro do método construtor, que é um método que toda classe deve ter, e que é o primeiro a ser executado sempre quando você cria um novo objeto daquela classe.

```
JS script.js > ...  
1  class Heroi{  
2      constructor(){  
3          this.nome;  
4          this.forca;  
5      }  
6  }
```

DEFININDO ATRIBUTOS

Também podemos passar parâmetros para o construtor, permitindo assim, definir valores para sua classe durante a criação de uma instância.

```
JS script.js > ...  
1  class Heroi{  
2      constructor(pNome,pForca){  
3          this.nome=pNome;  
4          this.forca=pForca;  
5      }  
6  }
```

DEFININDO MÉTODOS

Os métodos são funções que existem dentro de classes, eles funcionam exatamente da mesma forma de **funções** convencionais. Podem receber parâmetros, printar informações, retornar dados e etc. São definidos diretamente no **corpo** da classe quase da mesma forma de funções, só que sem precisar utilizar a palavra chave **function**.

```
JS script.js > ...
1  class Heroi{
2      constructor(pNome,pForca){
3          this.nome=pNome;
4          this.forca=pForca;
5      }
6
7      saudacao(){
8          return `Olá, eu sou o ${this.nome}`;
9      }
10 }
```

CRIANDO UM OBJETO

Um objeto é uma instância de uma classe, ou seja, um indivíduo dela.

Exemplo:

```
JS script.js > ...
1  class Heroi{
2      constructor(pNome,pForca){
3          this.nome=pNome;
4          this.forca=pForca;
5      }
6
7      saudacao(){
8          return `Olá, eu sou o ${this.nome}`;
9      }
10 }
11
12 const hulk=new Heroi();
13 const homemAranha=new Heroi();
14
```

DEFININDO VALORES PARA OS ATRIBUTOS

1 – Instanciando o objeto e passando valores posteriormente

```
const hulk=new Heroi();  
hulk.nome="Hulk";  
hulk.forca=100;  
  
const homemAranha=new Heroi();  
homemAranha.nome="Homem Aranha";  
homemAranha.forca=80;
```

```
const hulk=new Heroi("Hulk",100);  
  
const homemAranha=new Heroi("Homem Aranha",80);
```

2 – Passando valores na instância

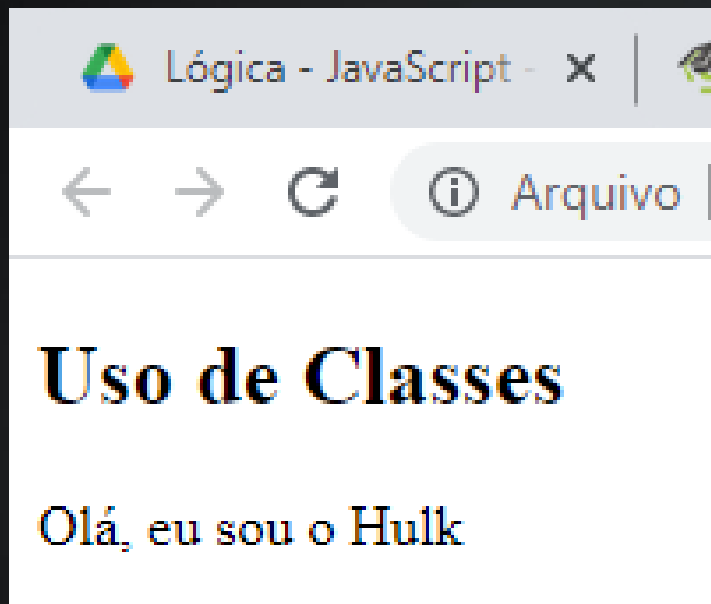
TESTANDO NO NAVEGADOR

```
<> index.html > ...
1  <!DOCTYPE html>
2  <html>
3  <meta charset="UTF-8">
4  <head>
5      <title>Classes</title>
6  </head>
7  <body>
8      <h2>Uso de Classes</h2>
9
10     <p id="saudacaoHeroi"></p>
11     <script type="text/JavaScript" src="js/script.js"></script>
12 </body>
13 </html>
```

TESTANDO NO NAVEGADOR

```
> JS script.js > ...
1  class Heroi{
2      constructor(pNome,pForca){
3          this.nome=pNome;
4          this.forca=pForca;
5      }
6
7      saudacao(){
8          return `Olá, eu sou o ${this.nome}`;
9      }
10 }
11
12 const hulk=new Heroi("Hulk",100);
13
14 document.getElementById("saudacaoHeroi").innerHTML=hulk.saudacao();
15
```

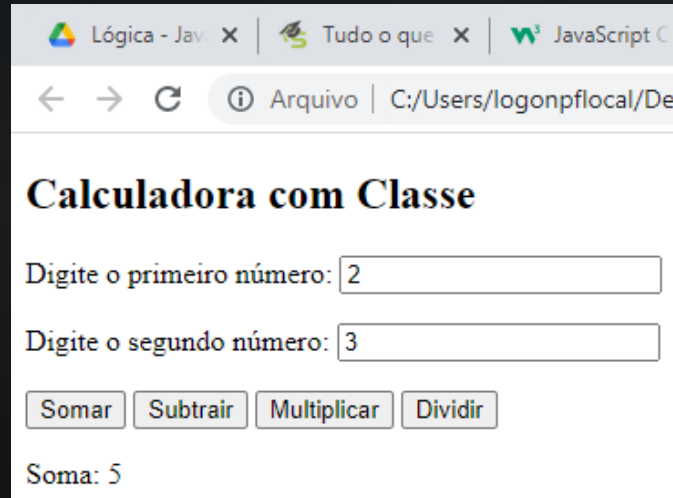
TESTANDO NO NAVEGADOR



MOMENTO **HANDS ON**

Crie a classe calculadora com 3 atributos (n1, n2 e res) e 4 métodos (soma, subtração, multiplicação e divisão).

Crie duas inputs para digitação dos valores, 4 botões com as respectivas operações e exiba o resultado.



The screenshot shows a web browser window with three tabs: 'Lógica - Java', 'Tudo o que', and 'JavaScript C'. The address bar shows the path 'C:/Users/logonpflocal/Des'. The main content area has the title 'Calculadora com Classe' in bold. Below the title, there are two input fields. The first is labeled 'Digite o primeiro número:' and contains the value '2'. The second is labeled 'Digite o segundo número:' and contains the value '3'. Below these inputs are four buttons: 'Somar', 'Subtrair', 'Multiplicar', and 'Dividir'. At the bottom, the text 'Soma: 5' is displayed.

Calculadora com Classe

Digite o primeiro número:

Digite o segundo número:

Soma: 5

index.html > ...

```
1 <!DOCTYPE html>
2 <html>
3 <meta charset="UTF-8">
4 <head>
5   <title>Classes</title>
6 </head>
7 <body>
8   <h2>Calculadora com Classe</h2>
9   <p>Digite o primeiro número: <input type="text" id="txtN1"></p>
10  <p>Digite o segundo número: <input type="text" id="txtN2"></p>
11  <p>
12    <button id="btnSomar" onclick="somar()">Somar</button>
13    <button id="btnSubtrair" onclick="subtrair()">Subtrair</button>
14    <button id="btnMultiplicar" onclick="multiplicar()">Multiplicar</button>
15    <button id="btnDividir" onclick="dividir()">Dividir</button>
16  </p>
17  <p id="resultado"></p>
18  <script type="text/JavaScript" src="js/script.js"></script>
19 </body>
20 </html>
```

```
JS script.js > Calculadora
1 class Calculadora {
2   constructor(pN1, pN2) {
3     this.n1 = pN1;
4     this.n2 = pN2;
5     this.res = 0;
6   }
7
8   somar() {
9     this.res = this.n1 + this.n2;
10  }
11 }
12
13
14 function somar() {
15   const calc = new Calculadora();
16   calc.n1 = parseFloat(document.getElementById("txtN1").value);
17   calc.n2 = parseFloat(document.getElementById("txtN2").value);
18   calc.somar();
19   document.getElementById("resultado").innerHTML = `Soma: ${calc.res.toString()}`;
20 }
```

POO - **ENCAPSULAMENTO**

Encapsulamento é um dos assuntos mais importantes na POO, pois ele vai garantir a **integridade** de seus objetos. A boa notícia é que ele existe no Javascript, e a má notícia é que o encapsulamento dentro da linguagem é puramente fictício e funciona apenas como **açúcar sintático**.

Um dos principais recursos do encapsulamento são os **getters** e **setters**, e para criá-los, primeiramente você precisa ter um **atributo**, e após isso, crie dois **métodos** utilizando as palavras chave **get** e **set** na frente de cada um respectivamente.

```
JS script.js > ...
1 class Heroi{
2     constructor(pNome,pForca){
3         this._nome=pNome;
4         this._forca=pForca;
5     }
6     get nome(){
7         return this._nome;
8     }
9     set nome(valor){
10        _nome=valor;
11    }
12
13    get forca(){
14        return this._nome;
15    }
16    set forca(valor){
17        _forca=valor;
18    }
19    saudacao(){
20        return `Olá, eu sou o ${this.nome}`;
21    }
22 }
23
24 const hulk=new Heroi();
25 hulk.nome="Hulk";
26 hulk.forca=100;
27
28 const homemAranha=new Heroi();
29 homemAranha.nome="Homem Aranha";
30 homemAranha.forca=80;
```

Utilizando o get/set note que seus atributos não ficam tão expostos. A manipulação deles é feita via métodos get/set. Nestes métodos é possível implementar outras linhas de código visando a segurança destes dados.

MOMENTO **HANDS ON**

Atualize o projeto da calculadora, criando `get` e `set` para todos os `atributos` da classe.

POO - **HERANÇA**

Técnica da POO é onde a classe filha (**subclasse**) herda tudo da classe mãe (**superclasse**), e automaticamente já tem todos os atributos e métodos da mãe, sem precisar reescrevê-los. Esta técnica permite uma grande reutilização de código.

```
JS script.js > ...
1  class Pessoa{
2      constructor(pNome, pCpf){
3          this.nome = pNome;
4          this.cpf = pCpf;
5      }
6      saudacao(){
7          return "Olá mundo";
8      }
9  }
10
11 class Professor extends Pessoa{
12     constructor(pNome, pCpf, pMateria){
13         // Chamada do construtor da superclasse(Pessoa)
14         // o super é uma representação do construtor da super classe
15         // E sempre deve ser a primeira linha na sub classe
16         super(pNome, pCpf);
17         this.materia = pMateria;
18     }
19     darAula(){
20         return `Agora vamos dar aula de ${this.materia}`;
21     }
22 }
```

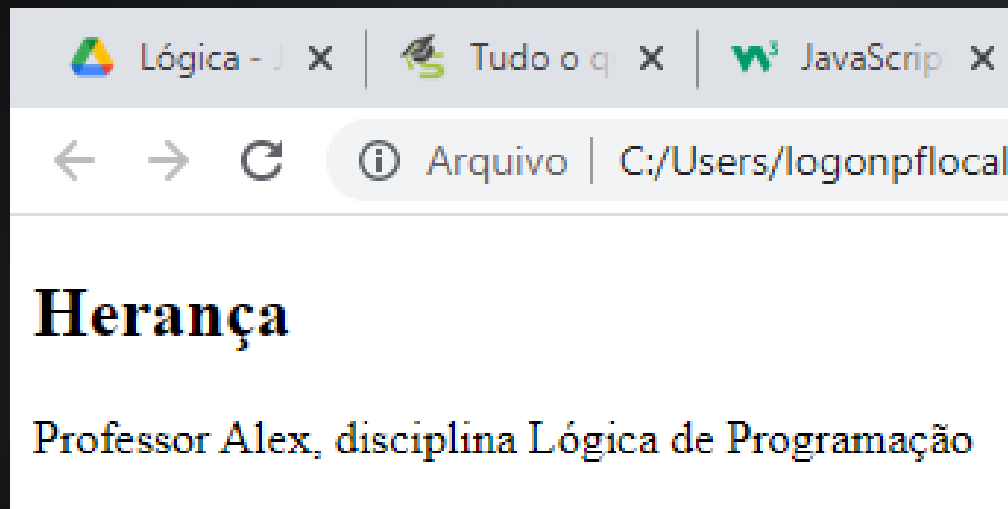
A classe professor herda de Pessoa. Portanto, esta classe tem como atributos a matéria (específico dela), e também nome e cpf (da super classe). O mesmo ocorre com os métodos: saudação herdado da super classe e dar Aula específico dela.

TESTANDO NO NAVEGADOR

<> index.html > ...

```
1  <!DOCTYPE html>
2  <html>
3  <meta charset="UTF-8">
4  <head>
5      <title>Classes</title>
6  </head>
7  <body>
8      <h2>Herança</h2>
9
10     <p id="teste"></p>
11     <script type="text/JavaScript" src="js/script.js"></script>
12 </body>
13 </html>
```


TESTANDO NO NAVEGADOR



TESTANDO NO NAVEGADOR

```
JS script.js > ...
1 class Pessoa {
2   constructor(pNome, pCpf) {
3     this.nome = pNome;
4     this.cpf = pCpf;
5   }
6   saudacao() {
7     return "Olá mundo";
8   }
9 }
10
11 class Professor extends Pessoa {
12   constructor(pNome, pCpf, pMateria) {
13     // Chamada do construtor da superclasse(Pessoa)
14     // o super é uma representação do construtor da super classe
15     // E sempre deve ser a primeira linha na sub classe
16     super(pNome, pCpf);
17     this.materia = pMateria;
18   }
19   darAula() {
20     return `Agora vamos dar aula de ${this.materia}`;
21   }
22 }
23
24 const prof = new Professor();
25 prof.nome = "Alex";
26 prof.cpf = "123.456.789-00";
27 prof.materia = "Lógica de Programação";
28 document.getElementById("teste").innerHTML = `Professor ${prof.nome},
29 disciplina ${prof.materia}`;
```

MOMENTO **HANDS ON**

Crie a classe **Funcionário** com os atributos: código, nome, valorHora e qtdeHoras. Crie o método calcularSalario que multiplica o valor da hora pela quantidade de horas e armazene o resultado em outro atributo: salário.

Crie uma sub classe de Funcionário chamada **Segurança**. Esta classe terá um atributo chamado bonificação e um método chamado calcularBonificacao. O valor desta bonificação é de 15% do salário calculado.

PARA **TREINAR MAIS**

01. Escreva uma classe cujos objetos representam alunos matriculados em uma disciplina. Cada objeto

dessa classe deve guardar os seguintes dados do aluno: matrícula, nome, 1 nota de prova e 1 nota de trabalho. Escreva os seguintes métodos para esta classe:

Media: calcula a média final do aluno.

Final: calcula quanto o aluno precisará para a prova final (quando sua média for menor do que 6, deve-se exibir quanto ele deverá tirar na recuperação para ter média 6, ou seja, subtraia sua média de 12). Retorna zero se ele não for para a prova final.

02. Crie uma classe chamada **Ingresso** que possui um valor em reais e um método `imprimeValor()`.

a. crie uma classe **VIP**, que herda **Ingresso** e possui um valor adicional. Crie um método que retorne o valor do ingresso VIP (com o adicional incluído).


b. crie uma classe **Normal**, que herda **Ingresso** e possui um método que imprime: "Ingresso Normal".

LEMBRE-SE DE VERSIONAR
SEUS PROJETOS NO GIT

BIBLIOGRAFIA

- <https://developer.mozilla.org/pt-BR/docs/Aprender/JavaScript>
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/String
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Math
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array
- https://developer.mozilla.org/pt-BR/docs/DOM/Referencia_do_DOM
- <https://www.w3schools.com/>
- <http://docplayer.com.br/17393758-Javascript-eventos-e-objetos-nativos.html>
- <https://www.adalgisa-souza.appspot.com/javascript/>
- <https://blog.schoolofnet.com/tudo-o-que-voce-precisa-saber-sobre-classes-no-javascript/>

OBRIGADO

 profalex.deus@fiap.com.br

 /alexsanderresende

FIAP

Copyright © 2020 | Alex Sander Resende de Deus

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento,
é expressamente proibido sem consentimento formal, por escrito, do professor/autor.

SHIFT

 FIAP



000000