

STEP Javascript homework 016

1. Construa uma hierarquia de código entre pessoas em um curso. P.ex: Alunos, professores, coordenador de curso, vendedores. Para cada hierarquia que pensou, crie dois objetos. Exiba infos (atributos, chamadas de método, etc) sobre a criação e sobre estados desses objetos na página.

Requerido: usar Herança (utilize o método `call()`)

Requerido: você deve ter um botão para criar um objeto desses ('criar um aluno', p.ex).

2. Faça uma função no código acima que liste os atributos de cada tipo de objeto que você pensou. Não os seus valores, mas seus nomes.

DICA: tente usar o método `Object.getOwnPropertyNames(<objeto>)`

Requerido em todos: os scripts devem estar separados, na subpasta js (a partir de agora não vamos mais fazer scripts 'embebidos no html' mas importados pela tag `script` - veja no código ao final deste doc

=====

Entregue a pasta inteira (a pasta com o arquivo .html e a subpasta 'js' com o arquivo .js dentro)

=====

Código desenvolvido em sala de aula

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <script src=".//js/aula016.js"></script>
  <title>Document</title>
</head>
<body>
```

```
</body>  
</html>
```

JAVASCRIPT

```
function Pessoa (primeiro, ultimo, idade, genero, interesses) {  
  
    this.nome = {primeiro, ultimo};  
    this.idade = idade;  
    this.genero = genero;  
    this.interesses = interesses;  
  
    // this.saudacao = saudacao;  
    // function saudacao() {  
    //     alert("Ola! Eu sou o " + this.nome.primeiro + ".")  
    // }  
  
    // variação do método acima  
    // this.saudacao = function () {  
    //     alert("Ola! Eu sou o " + this.nome.primeiro + ".")  
    // }  
}  
  
// substitui o metodo saudacao acima  
// O prototype é uma maneira de generalizar métodos  
// para vários objetos diferentes  
Pessoa.prototype.saudacao = function () {  
    document.write("Ola! Eu sou o " + this.nome.primeiro + ".")  
}  
  
// Professor "herda" de Pessoa. O método 'call' chama o construtor  
// da outra função construtora, sendo que a assinatura tem que ser  
// respeitada E o primeiro parametro é adicional e obrigatoriamente 'this'  
function Professor (primeiro, ultimo, idade, genero, interesses, assunto) {  
    Pessoa.call(this, primeiro, ultimo, idade, genero, interesses);  
    this.assunto = assunto;
```

```
}
```

```
let pessoa = new Pessoa(  
    "Arthur",  
    "Souza",  
    20,  
    "M",  
    ["Futebol", "Cozinha"]  
)  
console.log(pessoa.saudacao())  
let professor = new Professor("dudu", "prof", 172, "M", ["prog", "debug"],  
"javascript")  
  
// isto aqui está gerando erro  
// resolveremos isso na próxima aula  
console.log(professor.saudacao())  
  
// inspeciona os atributos  
console.log(Object.getOwnPropertyNames(professor))  
console.log(Object.getOwnPropertyNames(pessoa))
```