Javascript

Classes

Introdução

- Como vimos anteriormente, construtores podem nos ajudar a criar objetos, porém no Javascript moderno existem as classes, que nos fornecem mais funcionalidades para trabalhar com orientação a objetos.

- Vamos explorar esse construtor de objetos a seguir.

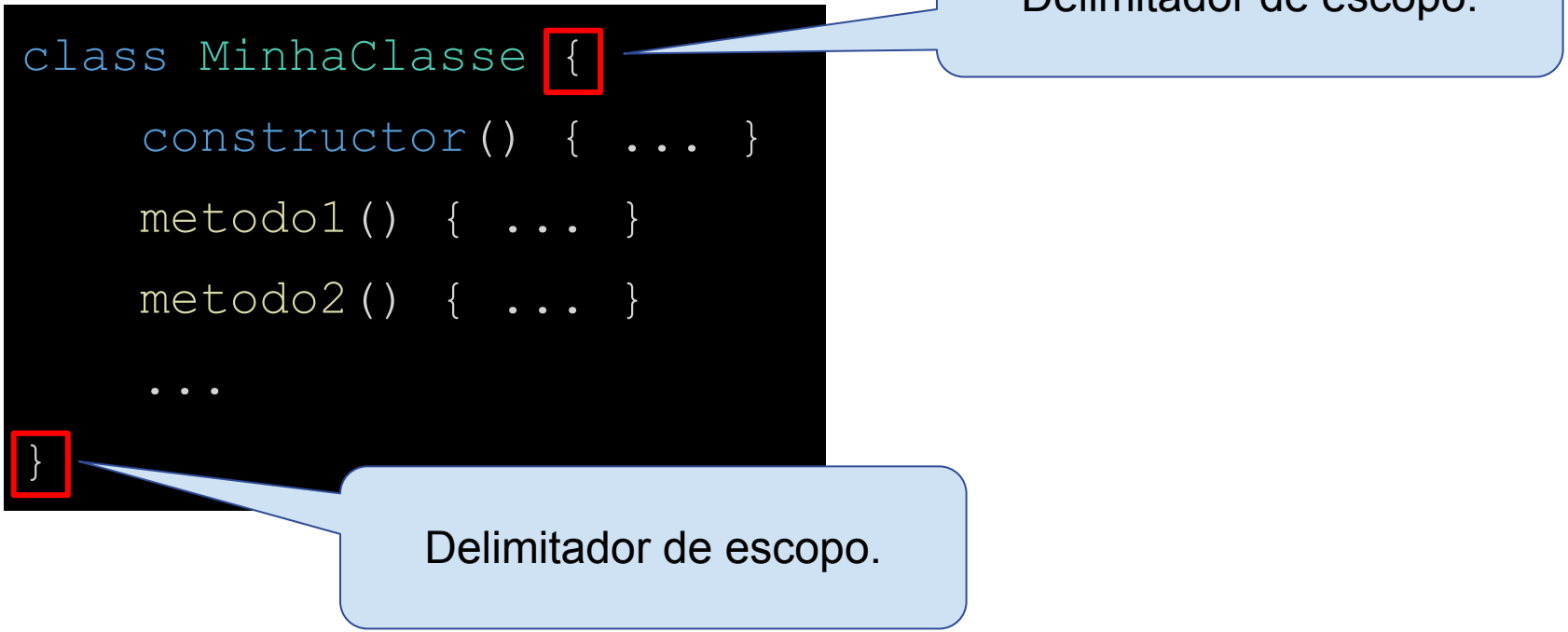
- Em ambos os casos, um objeto vazio (sem propriedades) foi criado.

Sintaxe básica de uma classe

```
class MinhaClasse {
    constructor() { ... }
    metodo1() { ... }
    metodo2() { ... }
    ...
}
```

Nome da classe.

Sintaxe básica de uma classe



Sintaxe básica de uma classe

```
class MinhaClasse {
    constructor() { ... }
    metodo1() { ... }
    metodo2() { ... }
    ...
}
```

Método construtor, que pode ou não receber parâmetros. Neste método inicializamos as propriedades do objeto.

Sintaxe básica de uma classe

```
class MinhaClasse {
    constructor() { ... }
    metodo1() { ... }
    metodo2() { ... }
    ...
}
```

Outros métodos podem ser implementados. Note que a sintaxe é um pouco diferente da utilizada na criação direta do objeto, pois não se separam os métodos por vírgula.

```
class Pessoa {
    constructor(nome, sobrenome) {
        this._nome = nome;
        this._sobrenome = sobrenome;
    }
    get nome() {
        return this._nome;
    }
    set nome(valor) {
        this._nome = valor.toUpperCase();
    }
    get sobrenome() {
        return this._sobrenome;
    }
    set sobrenome(valor) {
        this._sobrenome = valor.toUpperCase();
    }
    get nomeCompleto() {
        return this._nome + " " + this._sobrenome;
    }
}

let p = new Pessoa("", "Falcão");
p.nome = "Camillo";
console.log(p.nomeCompleto);
```

Exemplo do uso de uma classe

Exemplo do uso de uma classe

```
class Pessoa {
    constructor(nome, sobrenome) {
        this._nome = nome;
        this._sobrenome = sobrenome;
    }
    get nome() {
        return this._nome;
    }
    set nome(valor) {
        this._nome = valor.toU
    }
    get sobrenome() {
        return this._sobrenome;
    }
    set sobrenome(valor) {
        this._sobrenome = valor.
    }
    get nomeCompleto() {
        return this._nome + " " + this._sobrenome;
    }
}

let p = new Pessoa("", "Falcão");
p.nome = "Camillo";
console.log(p.nomeCompleto);
```

No construtor é possível acessar diretamente o campo desejado (this._nome, por exemplo) ou a propriedade que encapsula este campo (this.nome, por exemplo).

Exemplo do uso de uma classe

```
class Pessoa {
    constructor(nome, sobrenome) {
        this._nome = nome;
        this._sobrenome = sobrenome;
    }
    get nome() {
        return this._nome;
    }
    set nome(valor) {
        this._nome = valor.toUpperCase();
    }
    get sobrenome() {
        return this._sobrenome;
    }
    set sobrenome(valor) {
        this._sobrenome = valor.toUpperCase();
    }
    get nomeCompleto() {
        return this._nome + " " + this._sobrenome;
    }
}

let p = new Pessoa("", "Falcão");
p.nome = "Camillo";
console.log(p.nomeCompleto);
```

Propriedades podem ser utilizadas normalmente.

Exemplo do uso de uma classe

```
class Pessoa {
    constructor(nome, sobrenome) {
        this._nome = nome;
        this._sobrenome = sobrenome;
    }
    get nome() {
        return this._nome;
    }
    set nome(valor) {
        this._nome = valor.toUpperCase();
    }
    get sobrenome() {
        return this._sobrenome;
    }
    set sobrenome(valor) {
        this._sobrenome = valor.toUpperCase();
    }
    get nomeCompleto() {
        return this._nome + " " + this._sobrenome;
    }
}

let p = new Pessoa("", "Falcão");
p.nome = "Camilo";
console.log(p.nomeCompleto);
```

Para instanciar um objeto, é necessário utilizar o operador “new” em frente ao nome da classe. Podem ser passados parâmetros para o construtor do objeto.

Exemplo do uso de uma classe

```
class Pessoa {
    constructor(nome, sobrenome) {
        this._nome = nome;
        this._sobrenome = sobrenome;
    }
    get nome() {
        return this._nome;
    }
    set nome(valor) {
        this._nome = valor.toUpperCase();
    }
    get sobrenome() {
        return this._sobrenome;
    }
    set sobrenome(valor) {
        this._sobrenome = valor.toUpperCase();
    }
    get nomeCompleto() {
        return this._nome + " " + this._sobrenome;
    }
}

let p = new Pessoa("", "Falcão");
p.nome = "Camillo";
console.log(p.nomeCompleto);
```

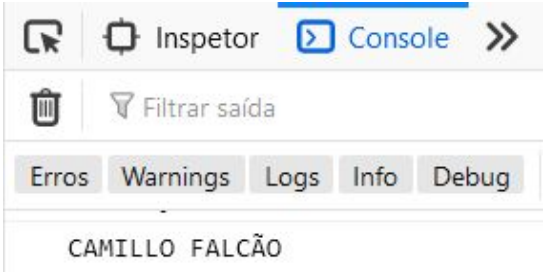
Propriedades e campos podem ser utilizados normalmente.

Exemplo do uso de uma classe

```
class Pessoa {
    constructor(nome, sobrenome) {
        this._nome = nome;
        this._sobrenome = sobrenome;
    }
    get nome() {
        return this._nome;
    }
    set nome(valor) {
        this._nome = valor.toUpperCase();
    }
    get sobrenome() {
        return this._sobrenome;
    }
    set sobrenome(valor) {
        this._sobrenome = valor.toUpperCase();
    }
    get nomeCompleto() {
        return this._nome + " " + this._sobrenome;
    }
}

let p = new Pessoa("", "Falcão");
p.nome = "Camillo";
console.log(p.nomeCompleto);
```

Resultado após execução:



Herança

```
class Conta {
    constructor(saldo) {
        this._saldo = saldo;
    }
    get saldo() {
        return this._saldo;
    }
    set saldo(valor) {
        this._saldo = valor;
    }
}
```

É possível herdar de uma classe em Javascript através do comando “extends”. Veja a superclasse ao lado.

Herança

```
class ContaCorrente extends Conta{
    constructor(saldo,limite) {
        super(saldo);
        this._limite = limite;
    }
    get limite() {
        return this._limite;
    }
    set limite(valor) {
        this._limite = valor;
    }
}
```

Herança

```
class ContaCorrente extends Conta{
    constructor(saldo,limite) {
        super(saldo);
        this._limite = limite;
    }
    get limite() {
        return this._limite;
    }
    set limite(valor) {
        this._limite = valor;
    }
}
```

A classe ContaCorrente possui saldo e limite, sendo “saldo” herdado da classe Conta.

instanceof

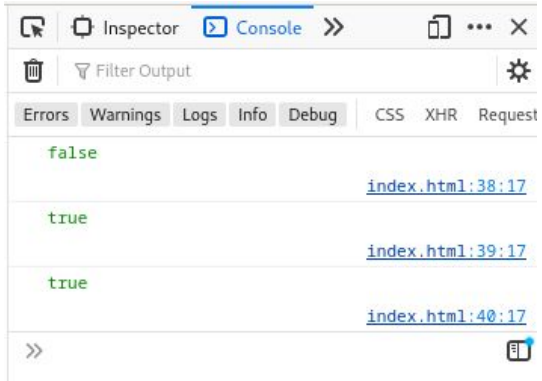
- O operador *instanceof* permite saber se um objeto pertence a uma classe informada ou se o objeto em questão pertence a uma classe que herda da classe informada.

- Veja à seguir o operador *instenceof* sendo utilizado em instâncias das classes Conta e ContaCorrente, apresentadas anteriormente.

instanceof

```
let obj1 = new Conta(100);
let obj2 = new ContaCorrente(10, 0);

console.log(obj1 instanceof ContaCorrente); //false
console.log(obj1 instanceof Conta);         //true
console.log(obj2 instanceof ContaCorrente); //true
```



Métodos Estáticos

- Métodos estáticos pertencem à classe e não aos objetos.

- Veja em seguida a classe Aluno e dois métodos estáticos implementados:
  - compararPorMatricula
  - compararPorNome

Métodos Estáticos

```
class Aluno {
 constructor(matricula, nome) {
       this. matricula = matricula;
       this._nome = nome;
   }
   get matricula() {
       return this._matricula;
   }
   get nome() {
       return this._nome;
   }
   static compararPorMatricula(obj1, obj2)
   {
     return obj1.matricula - obj2.matricula;
   }
   static compararPorNome(obj1, obj2)
   {
     if (obj1.nome == obj2.nome) {
       return 0;
     }
     return obj1.nome > obj2.nome ? 1 : -1;
   }
}
```

Este método retorna um número negativo quando a matrícula do primeiro objeto recebido por parâmetro for menor que a do segundo, retorna um número positivo se a matrícula do primeiro for maior ou retorna zero caso as matrículas sejam iguais.

Métodos Estáticos

```
class Aluno {
 constructor(matricula, nome) {
       this. matricula = matricula;
       this._nome = nome;
   }
   get matricula() {
       return this._matricula;
   }
   get nome() {
       return this._nome;
   }
   static compararPorMatricula(obj1, obj2)
   {
     return obj1.matricula - obj2.matricula;
   }
   static compararPorNome(obj1, obj2)
   {
     if (obj1.nome == obj2.nome) {
       return 0;
     }
     return obj1.nome > obj2.nome ? 1 : -1;
   }
}
```

Considerando a ordem lexicográfica, este método retorna -1 se o nome do objeto da esquerda for menor que o do objeto da direita, retorna 1 se o nome da esquerda for maior que o da direita. Retorna 0 caso os nomes sejam iguais.

Métodos Estáticos

- Agora veja o resultado da execução do código abaixo:

```
const alunos = [
 new Aluno(123, "Bruno"),
 new Aluno(120, "Carlos"),
 new Aluno(121, "Ana"),
];

alunos.sort(Aluno.compararPorMatricula);
console.log(alunos);
```

Métodos Estáticos

- Agora veja o resultado da execução do código abaixo:

```
const alunos = [
 new Aluno(123, "Bruno"),
 new Aluno(120, "Carlos"),
 new Aluno(121, "Ana"),
];

alunos.sort(Aluno.compararPorNome);
console.log(alunos);
```

Fim de material

Dúvidas?