<LAB365> SENAI

SEJAM BEM-VINDOS!

Jornada DEV AM. CE. RR

Formação gratuita
em Desenvolvimento
Front-End SENAI

Começamos em breve!

A aula vai iniciar em **5 minutos**.

Relaxe e aproveite a música enquanto aguardamos todos chegarem!

Lógica com JavaScript #12

SENAI

Missão de hoje

- Entender o que são métodos de instância
- Usar o this para acessar variáveis dentro de métodos
- Aplicar herança de classes para reaproveitar código
- Compreender como sobrescrever métodos
- Desenvolver projetos que utilizam conceitos de orientação a objetos.

Os quatro pilares da POO

- **Encapsulamento:** A ideia de agrupar os dados (atributos) e os comportamentos (métodos) relacionados dentro de um objeto. Isso permite que o objeto controle o acesso aos seus próprios dados, protegendo-os de interferências externas.
- **Herança:** Um mecanismo que permite que uma nova classe (derivada) herde atributos e métodos de uma classe existente (base). Isso promove a reutilização de código e cria hierarquias de classes, como uma classe "Cachorro" herdando de uma classe "Animal".
- **Polimorfismo:** Permite que objetos de diferentes classes respondam a uma mesma chamada de método de maneiras específicas para cada um. Por exemplo, tanto um "Cachorro" quanto um "Gato" podem ter um método "emitirSom()", mas cada um emitirá um som diferente (latido ou miado).
- **Abstração:** Consiste em simplificar a complexidade do mundo real, mostrando apenas os detalhes essenciais e relevantes para o objeto. Isso ajuda a criar modelos mais fáceis de entender e gerenciar.

O que é um método de instância?

É uma função definida **dentro de uma classe**, que pode ser chamada em uma **instância** dessa classe.

Serve para representar **ações** ou **comportamentos do objeto**.

Esse método pertence apenas a quem foi criado a partir dessa classe — é específico de cada instância.”

```
class Usuario {
  constructor(nome, sobrenome) {
    this.nome = nome;
    this.sobrenome = sobrenome;
  }

  obterNomeCompleto() {
    return "Nome completo aqui";
  }
}

const usuario1 = new Usuario("Fábio", "Lima");
console.log(usuario1.obterNomeCompleto());
```

Implementando métodos de instância

Um método de instância pode usar as variáveis da instância através do `this`.

```
class Usuario {
  constructor(nome, sobrenome) {
    this.nome = nome;
    this.sobrenome = sobrenome;
  }

  obterNomeCompleto() {
    return `${this.nome} ${this.sobrenome}`;
  }
}

const usuario = new Usuario("Juliana", "Cunha");
console.log(usuario.obterNomeCompleto()); // Juliana Cunha
```

Sempre use `this` para acessar propriedades do próprio objeto.

Exemplo prático com condição

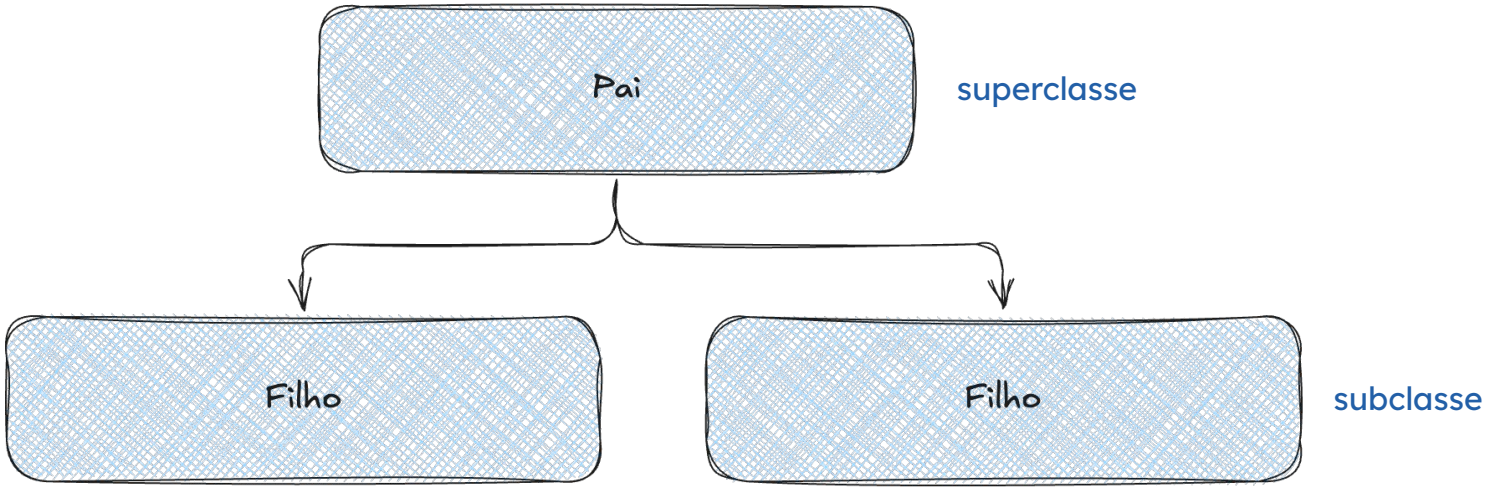
```
class Usuario {
  constructor(nome, idade) {
    this.nome = nome;
    this.idade = idade;
  }

  podeVotar() {
    return this.idade >= 16;
  }
}
```

Os métodos de instância não recebem parâmetros se as informações **já estão salvas** dentro da classe.

Herança de Classe

Por que precisamos de herança?

Quando duas classes compartilham código semelhante, podemos reaproveitar métodos e evitar duplicação.



DRY

Código antes da herança

```
class Funcionario {
  constructor(nome, sobrenome) {
    this.nome = nome;
    this.sobrenome = sobrenome;
  }

  obtemNomeCompleto() {
    return `${this.nome} ${this.sobrenome}`;
  }
}
class Administrador {
  constructor(nome, sobrenome) {
    this.nome = nome;
    this.sobrenome = sobrenome;
  }

  obtemNomeCompleto() {
    return `${this.nome} ${this.sobrenome}`;
  }
}
```

Usando herança (extends)

```
class Funcionario {
  constructor(nome, sobrenome) {
    this.nome = nome;
    this.sobrenome = sobrenome;
  }

  obtemNomeCompleto() {
    return `${this.nome} ${this.sobrenome}`;
  }
}

class Administrador extends Funcionario {
  pagarSalarios() {
    console.log("Pagando salários...");
  }
}

const admin = new Administrador("Regina", "Silva");
console.log(admin.obtemNomeCompleto()); // Regina Silva
admin.pagarSalarios(); // método da filha
```

A classe filha **herda tudo** da classe pai, e ainda pode ter suas próprias funções.

Intervalo!

Finalizamos o nosso primeiro período de hoje.

Nos vemos em 15min.

CE
Início: 20:00
Retorno: 20:15

AM. RR
Início: 19:00
Retorno: 19:15

Sobrescrevendo métodos da classe pai

```
class Funcionario {
  constructor(nome, sobrenome) {
    this.nome = nome;
    this.sobrenome = sobrenome;
  }

  obtemNomeCompleto() {
    return `${this.nome} ${this.sobrenome}`;
  }
}

class Administrador extends Funcionario {
  obtemNomeCompleto() {
    return `${this.nome} ${this.sobrenome} (admin)`;
  }
}

const regina = new Administrador("Regina", "Silva");
console.log(regina.obtemNomeCompleto()); // Regina Silva (admin)
```

Resumo:

- `extends` → indica herança
- A classe filha herda métodos da classe pai
- Pode sobrescrever para alterar o comportamento

Cuidados com herança

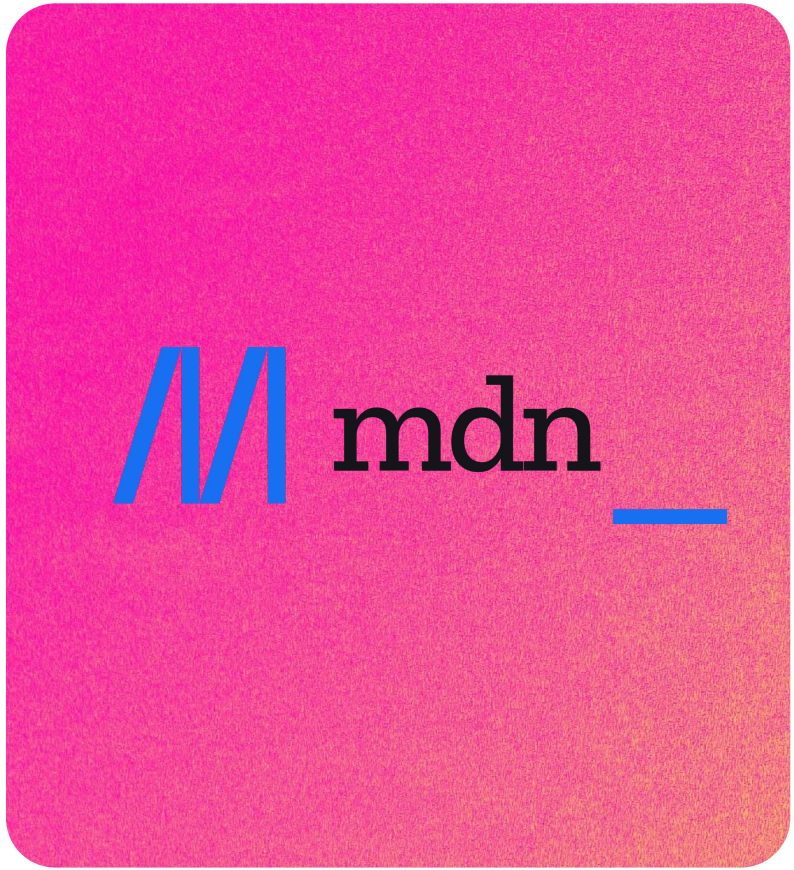
- A classe pai **precisa ser definida antes** da classe filha
- Sempre use `extends` seguido do nome da classe pai
- Se precisar usar o construtor da classe pai, use super()

Atividades

- https://app.devstart.tech/learn/logica-de-programacao/metodos-de-instancia-de-classe/ex1-introducao-aos-metodos-de-instancia
- https://app.devstart.tech/learn/logica-de-programacao/metodos-de-instancia-de-classe/ex1-implementando-metodos-de-instancia
- https://app.devstart.tech/learn/logica-de-programacao/projetos-com-classes/ex1-projeto-18-aplicativo-de-nomes-com-classes
- https://app.devstart.tech/learn/logica-de-programacao/projetos-com-classes/ex1-projeto-19-aplicativo-de-sala-de-aula-com-classes
- https://app.devstart.tech/learn/logica-de-programacao/heranca-de-classe/ex2-estendendo-a-classe-pai

Material complementar

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes/Private_elements
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes/static




<LAB365>
SENAI