



INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 08 - OBJETOS LITERAIS

O QUE IREMOS APRENDER

01

PROGRAMAÇÃO ORIENTADA A OBJETOS

02

OBJETOS LITERAIS

03

DESESTRUTURANDO OBJETOS

04

ARRAY DE OBJETOS

Revisão da Aula Anterior

- MÉTODOS AUXILIARES DE ARRAYS
- UTILIZANDO IA



PROGRAMAÇÃO ORIENTADA A OBJETOS

A Programação Orientada a Objetos (**POO**) é um **paradigma que organiza o código em torno de objetos**, que combinam dados (propriedades) e comportamentos (métodos).

Em POO, classes são usadas como modelos para criar objetos, permitindo reutilização de código e melhor organização.

Principais conceitos incluem herança, encapsulamento, polimorfismo e abstração, que ajudam a criar sistemas modulares, escaláveis e de fácil manutenção.

{ POO }

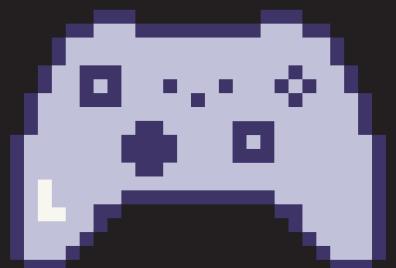
Programação Orientada a Objetos



APLICAÇÕES



◆ Desenvolvimento de Aplicações Web



◆ Desenvolvimento de Jogos



◆ Desenvolvimento de Aplicativos Desktop

PRÓS E CONTRAS DA POO

Vantagens:

- **Reutilização de Código:** Criação de classes reutilizáveis, reduzindo a duplicação de código.
- **Modularidade:** O código é dividido em objetos e classes, facilitando manutenção e gerenciamento.
- **Facilidade de Expansão:** Novos recursos podem ser adicionados sem modificar o código existente, tornando-o escalável.
- **Modelagem do Mundo Real:** Representa conceitos do mundo real de forma intuitiva no código.
- **Encapsulamento e Segurança:** Protege dados internos, garantindo que apenas o código autorizado possa acessá-los.

Desvantagens:

- **Desempenho:** Pode introduzir overhead, resultando em menor desempenho em comparação com paradigmas mais simples.
- **Complexidade Inicial:** Requer mais planejamento, o que pode ser excessivo para projetos pequenos.
- **Superengenharia:** Desenvolvedores inexperientes podem criar classes desnecessárias, complicando o código.
- **Dificuldade em Projetos Pequenos:** A estrutura da POO pode ser complexa demais para projetos simples, tornando o desenvolvimento menos eficiente.



CLASSES E OBJETOS



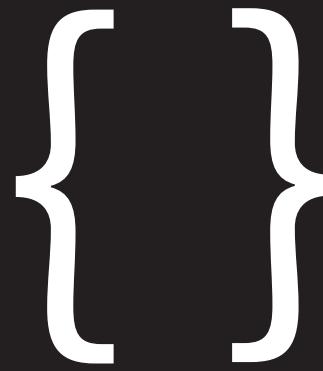
Classe: É como um modelo ou molde que define a estrutura e o comportamento de um conjunto de objetos.

Ela descreve quais propriedades (atributos) e métodos (funções) os objetos desse tipo terão. Classes não ocupam memória até que sejam instanciadas em objetos.



Objeto: É uma instância de uma classe. Ou seja, quando você cria um objeto, você está criando uma cópia baseada na classe com valores específicos para seus atributos. Objetos ocupam memória e têm seus próprios dados.

OBJETOS LITERAIS



Objetos literais em JavaScript são **estruturas de dados que permitem armazenar pares de chave-valor**.

Eles são uma maneira simples e direta de criar objetos, onde as **chaves (também chamadas de propriedades)** são strings que identificam os valores armazenados.

Os valores podem ser de qualquer tipo, incluindo números, strings, arrays, funções ou outros objetos.

CRIANDO OBJETOS

Um objeto literal é definido usando {} e as propriedades são escritas no formato chave: valor, separadas por vírgulas.

Exemplo

```
● ● ●  
1 let pessoa = {  
2   nome: "João",  
3   idade: 30,  
4   profissão: "Desenvolvedor"  
5 };
```

Neste exemplo, pessoa é um objeto literal com três propriedades: **nome**, **idade**, e **profissão**, cada uma com seu respectivo valor. Objetos literais são fundamentais em JavaScript para organizar e estruturar dados de maneira fácil e acessível.

MANIPULANDO OBJETOS

Manipular objetos literais em envolve acessar, modificar, adicionar e remover suas propriedades.
Aqui estão algumas operações comuns:



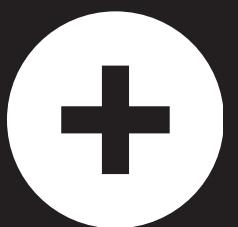
Acessando um valor:

```
1 let pessoa = { nome: "João", idade: 30 };
2 console.log(pessoa.nome); // "João"
```



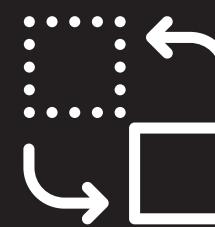
Apagando propriedades

```
1 delete pessoa.idade;
```



Adicionando uma nova propriedade:

```
1 pessoa.profissão = "Desenvolvedor";
```



Alterando um valor:

```
1 pessoa.idade = 31;
2 pessoa["nome"] = "Maria";
```

ATIVIDADE PRÁTICA

Atividade 01

Crie um objeto chamado carro que contenha as seguintes propriedades:

marca: uma string representando a marca do carro.

modelo: uma string representando o modelo do carro.

ano: um número representando o ano de fabricação.

Após criar o objeto, exiba no console cada uma das propriedades separadamente (marca, modelo, e ano).

Objetivo:

Praticar a criação de objetos e o acesso às suas propriedades.

ATIVIDADE PRÁTICA

Atividade 02

Crie um objeto chamado pessoa com as propriedades nome, idade, e profissao.

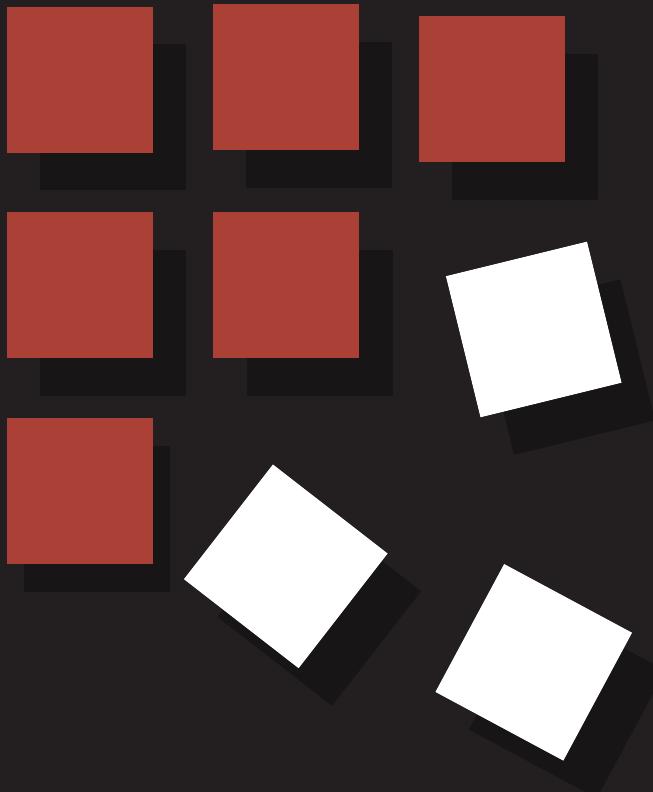
Em seguida:

- 1 Altere a propriedade profissao para um novo valor.
- 2 Adicione uma nova propriedade chamada cidade ao objeto.
- 3 Exiba no console o objeto completo após as modificações.

Objetivo:

Praticar a manipulação de objetos, alterando propriedades existentes, adicionando novas propriedades, e exibindo o objeto atualizado.

DESESTRUTURAÇÃO DE OBJETOS



A desestruturação de objetos é uma forma concisa de extrair valores de um objeto e atribuí-los a variáveis individuais.

Facilita o acesso a propriedades específicas de um objeto, tornando o código mais limpo e legível.

EXTRAINDO PROPRIEDADES

Neste código, utilizamos a desestruturação de objetos para extrair as propriedades nome e idade do objeto pessoa, atribuindo-as a variáveis com o mesmo nome.



```
1 let pessoa = { nome: "João", idade: 30, profissão: "Desenvolvedor" };
2 let { nome, idade } = pessoa;
3 console.log(nome); // "João"
4 console.log(idade); // 30
```

let { nome, idade } = pessoa;

Desestrutura o objeto pessoa, criando as variáveis nome e idade com os valores correspondentes.

console.log(nome);: Exibe "João" no console.

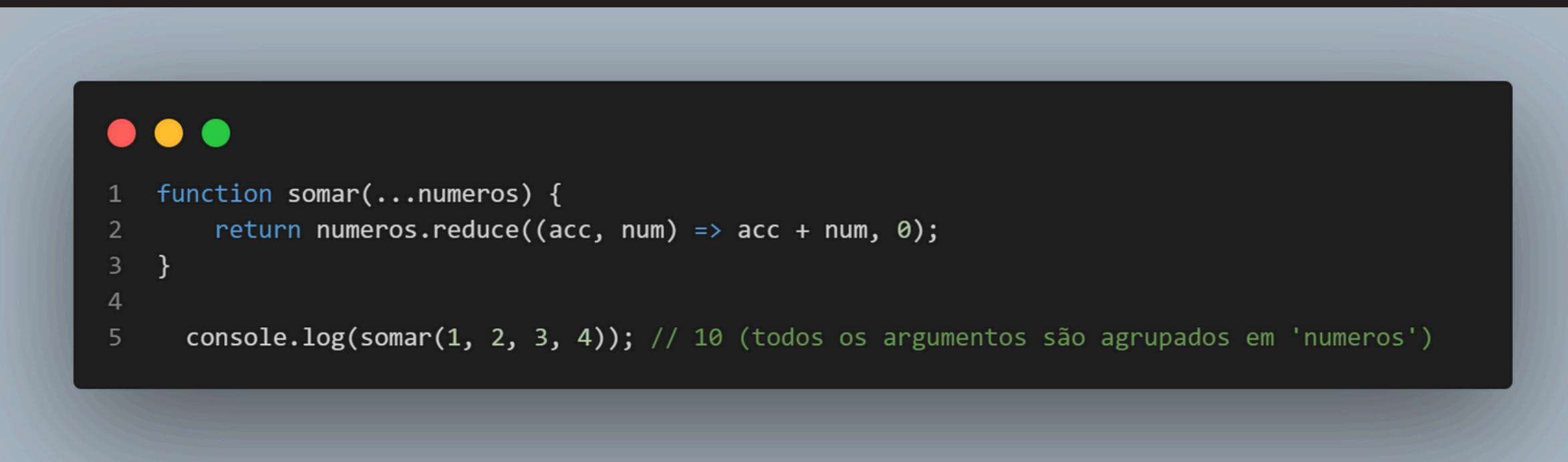
console.log(idade);: Exibe 30 no console.

REST OPERATOR (...) COM FUNÇÕES

Permite que você agrupe múltiplos argumentos ou elementos em um único array. Ele é usado principalmente em funções para coletar os argumentos restantes em um array, ou em objetos e arrays para agrupar elementos restantes.

Uso em Funções:

Quando usado em uma função, o rest operator captura todos os argumentos passados após os parâmetros definidos e os coloca em um array.



```
● ● ●  
1  function somar(...numeros) {  
2      return numeros.reduce((acc, num) => acc + num, 0);  
3  }  
4  
5  console.log(somar(1, 2, 3, 4)); // 10 (todos os argumentos são agrupados em 'numeros')
```

REST OPERATOR (...) COM OBJETOS

Uso em Arrays e Objetos:

Você pode usar o rest operator para capturar os elementos restantes de um array ou as propriedades restantes de um objeto.



```
1 const exibirDetalhesFilme = ({ titulo, diretor, ...detalhes } ) => {
2   console.log(`Título: ${titulo}`);
3   console.log(`Diretor: ${diretor}`);
4   console.log("Outros detalhes:", detalhes);
5 };
6
7 const filme = {
8   titulo: "Inception",
9   diretor: "Christopher Nolan",
10  ano: 2010,
11  genero: "Ficção Científica",
12  duracao: "148 minutos"
13 };
14
15 exibirDetalhesFilme(filme);
16 // Título: Inception
17 // Diretor: Christopher Nolan
18 // Outros detalhes: { ano: 2010, genero: 'Ficção Científica', duracao: '148 minutos' }
```

ATIVIDADE PRÁTICA

Atividade 03

Crie um objeto chamado livro que contenha as seguintes propriedades:

titulo: uma string representando o título do livro.

autor: uma string com o nome do autor.

ano: um número representando o ano de publicação.

editora: uma string com o nome da editora.

Utilize a desestruturação para extrair as propriedades **titulo** e **autor** do objeto **livro**.

Em seguida, exiba essas propriedades no console.

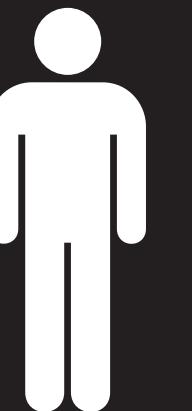
Objetivo:

Praticar a desestruturação de objetos em JavaScript, extraíndo propriedades e definindo valores padrão.

ARRAY DE OBJETOS

Em JavaScript, um array de objetos é uma **estrutura que combina as funcionalidades dos arrays e dos objetos**, permitindo armazenar e organizar múltiplos objetos dentro de um único array. Cada elemento do array é um objeto, o que facilita o agrupamento de dados relacionados em uma coleção.

Por exemplo, um array de objetos pode ser usado para representar uma lista de pessoas, onde cada pessoa é um objeto com propriedades como nome, idade, e profissão

pessoas = [   **]**

CONSTRUINDO UM ARRAY DE OBJETOS

Exemplo

```
1 const pessoas = [  
2   {nome: "Ana", idade: 25, profissão: "Engenheira"},  
3   {nome: "Bia", idade: 24, profissão: "Veterinaria"},  
4   {nome: "Carol", idade: 25, profissão: "Diretora"}  
5 ];
```

Esse formato é muito útil para manipular e acessar dados complexos de forma organizada, e é amplamente utilizado em aplicações modernas para trabalhar com conjuntos de dados.

SPREAD OPERATOR

O spread operator (...) é uma funcionalidade do JavaScript que permite "espalhar" os elementos de um array ou as propriedades de um objeto em outro array ou objeto.

O spread operator pode ser usado para copiar ou concatenar arrays.

Uso com Arrays



```
1 const arr1 = [1, 2, 3];
2 const arr2 = [...arr1, 4, 5]; // [1, 2, 3, 4, 5]
```

Permite copiar objetos ou combinar propriedades de vários objetos em um único objeto.

Uso com Objetos



```
1 const obj1 = { a: 1, b: 2};
2 const obj2 = { ... obj1, c: 3}; // {a: 1, b: 2, c: 3}
```

ATIVIDADE PRÁTICA

Atividade 04

Crie um array chamado `livros`, onde cada elemento do array é um objeto representando um livro, com as seguintes propriedades:

titulo: string com o título do livro.

autor: string com o nome do autor.

Após criar o array:

- Exiba no console todos os títulos dos livros.
- Adicione um novo objeto representando outro livro ao array.
- Exiba no console a lista completa de livros após a adição.

Objetivo:

Praticar a criação, manipulação e iteração de um array de objetos

ATIVIDADE PRÁTICA

Atividade 05

Crie dois objetos chamados representando dois carros com as seguintes propriedades:

marca: string com a marca do carro.

modelo: string com o modelo do carro.

ano: número representando o ano de fabricação.

Em seguida:

- Crie um novo objeto que combine as propriedades dos dois objetos usando o spread operator.
- Exiba o novo objeto no console.

Objetivo:

Praticar o uso do spread operator para combinar objetos.

PRATIQUE E APRENDA

Crie um objeto chamado aluno que deve conter as seguintes propriedades:

nome: uma string com o nome do aluno.

notas: um array com as notas do aluno.

Além das propriedades, adicione os seguintes métodos ao objeto aluno:

calcularMedia(): Este método deve calcular e retornar a média das notas armazenadas no array notas.

adicionarNota(nota): Este método deve permitir adicionar novas notas ao array de notas do aluno.

Após criar o objeto:

- Calcule a média das notas usando o método calcularMedia() e exiba o resultado no console.

Objetivo: Praticar a criação de objetos com métodos em JavaScript, manipulando dados dentro do objeto e aplicando lógica para calcular a média das notas.

SE LIGA NO CONTEÚDO DA PRÓXIMA AULA!

AULA 09 DE JAVASCRIPT.
REVISÃO.



INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Aula 09 - Revisão

FUNÇÕES

MÉTODOS AUXILIARES DE ARRAY

PROGRAMAÇÃO ORIENTADA A OBJETOS



INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 08 - OBJETOS LITERAIS