JavaScript: Fundamentos para Desenvolvimento Web Interativo - Turma 2025B

3.2 Chamando funções

A definição de uma função não a executa. Chamar a função executa realmente as ações especificadas com os parâmetros indicados. Há diversas maneiras de chamar uma função, cada uma com seu comportamento particular em relação ao contexto, parâmetros e resultados. Vamos explorar as principais formas de chamar funções em JavaScript e os conceitos associados.

Chamada direta

Esta é a forma mais comum de chamar uma função. Basta usar o nome da função seguido de parênteses, contendo os argumentos (ou sem, se não houver parâmetros). Veja um exemplo no quadro abaixo:

```
function saudar(nome) {
    console.log("Olá, " + nome + "!");
}

saudar("Ana"); // Invoca a função com o argumento "Ana"
```

Chamada como método

Se uma função for parte de um objeto, chamá-la como método implica que o contexto (ou seja, o valor de **this**) será o objeto ao qual a função pertence. Veja um exemplo no quadro abaixo:

```
const pessoa = {
    nome: "Carlos",
    cumprimentar() {
        console.log("Olá, eu sou " + this.nome);
    }
};

pessoa.cumprimentar(); // Chamada como método, `this` refere-se a `pessoa`
```

Chamada com operador new

Quando uma função é chamada com o operador new, ela se comporta como um construtor, criando uma nova instância de um objeto. Neste caso, **this** refere-se à nova instância, e a função geralmente é usada para inicializar propriedades ou métodos da instância. Veja um exemplo no quadro abaixo:

```
function Carro(marca) {
    this.marca = marca;
}

const meuCarro = new Carro("Toyota"); // Cria uma nova instância de `Carro`
```

Chamada com funções de retorno (callbacks)

Em JavaScript, é comum usar funções como argumentos para outras funções, criando um mecanismo de "**callback**". Ao chamar uma função com um callback, o código pode ser assíncrono ou executar lógica adicional antes ou depois do callback. Veja um exemplo no quadro abaixo:

```
function executarComCallback(callback) {
    console.log("Antes do callback");
    callback(); // Chama a função de retorno
    console.log("Depois do callback");
}

executarComCallback(() => {
    console.log("Callback executado");
});
```

Neste exemplo, uma função anônima é passada como callback para ser chamada dentro da função **executarComCallback**.

◄ 3.1 Definindo funções

Seguir para...

3.3 Passagem de parâmetros ►