

Conteúdo do dia!

Site: [Geração Tech](#)
Curso: Formação em Desenvolvedor Web - Online
Livro: Conteúdo do dia!

Impresso por: JOÃO VITOR DE MELO FREITAS
Data: quinta-feira, 18 jul. 2024, 22:35

Índice

1. JavaScript

- 1.1. Estruturas Condicionais em JavaScript
- 1.2. Continuação Estrutura Condicionais
- 1.3. Switch Case
- 1.4. Operador Ternário

1. JavaScript

Introdução

Até agora, vimos variáveis, constantes e como usar o `prompt` para obter valores através do navegador. Hoje, vamos explorar as estruturas condicionais em JavaScript.

Estruturas Condicionais


Estruturas condicionais são usadas para tomar decisões em um programa. Elas envolvem um teste lógico que determina qual caminho o programa deve seguir. Um exemplo simples seria: você se pesou na balança e não gostou do que viu. Se o peso está acima do desejado, você decide começar um regime; se está dentro do esperado, você mantém sua rotina atual. Essa tomada de decisão é o que chamamos de estrutura condicional.

1.1. Estruturas Condicionais em JavaScript

Sintaxe Básica

Em JavaScript, a estrutura condicional básica é feita com a palavra-chave `if`. Vamos ver um exemplo:

javascript

 Copiar código

```
let numero1 = 4; if (numero1 > 2) { console.log("O número é maior que 2"); }
```

Neste exemplo, estamos verificando se `numero1` é maior que 2. Se a condição for verdadeira, a mensagem "O número é maior que 2" será exibida no console.

Teste Lógico

Um teste lógico compara dois valores e retorna `true` (verdadeiro) ou `false` (falso). Os operadores de comparação incluem:

- `==` : igual a
- `!=` : diferente de
- `>` : maior que
- `<` : menor que
- `>=` : maior ou igual a
- `<=` : menor ou igual a

javascript


 Copiar código

```
console.log(2 + 2 == 4); // true console.log(2 + 2 == 5); // false
```

Estruturas Condicionais com `if`, `else if` e `else`

Podemos combinar múltiplos testes lógicos usando `else if` e `else`.

javascript

 Copiar código

```
let numero = parseInt(prompt("Digite um número:")); if (numero > 5) { console.log("O número é maior que 5"); } else if (numero < 5) { console.log("O número é menor que 5"); } else { console.log("O número é igual a 5"); }
```

Estrutura Condicional Completa

Vamos melhorar nosso exemplo para abranger todas as possibilidades de comparação.

javascript


 Copiar código

```
let numero = parseInt(prompt("Digite um número:")); if (numero > 5) { console.log("O número é maior que 5"); } else if (numero < 5) { console.log("O número é menor que 5"); } else { console.log("O número é igual a 5"); }
```

Exemplos Práticos

Exemplo 1: Maior ou Menor que 5

javascript

 Copiar código

```
let numero1 = parseInt(prompt("Digite um número:")); if (numero1 > 5) { console.log("O número é maior que 5"); } else if (numero1 < 5) { console.log("O número é menor que 5"); } else { console.log("O número é igual a 5"); }
```

Depuração de Código


Para depurar e testar nosso código, podemos utilizar o console do navegador. Acesse-o pressionando **Ctrl + Shift + J** (Windows) ou **Cmd + Option + J** (Mac). O console nos permite ver mensagens e erros gerados pelo código.

Local Storage

O `localStorage` permite armazenar dados no navegador de forma persistente.

Armazenando Dados


javascript

 Copiar código

```
localStorage.setItem("nome", "Gleidson");
```

Recuperando Dados

javascript

 Copiar código


```
let nome = localStorage.getItem("nome"); console.log(nome); // Gleidson
```

setInterval

A função `setInterval` executa uma função repetidamente em um intervalo de tempo especificado.

Exemplo de setInterval

javascript

 Copiar código

```
function mostrarHora() { let agora = new Date(); document.getElementById("relogio").innerHTML = agora.toLocaleTimeString();  
} setInterval(mostrarHora, 1000);
```

Referência MDN

A documentação MDN (Mozilla Developer Network) é uma excelente fonte para aprender mais sobre JavaScript. Ela fornece tutoriais, exemplos e explicações detalhadas sobre várias funcionalidades da linguagem.

dia 12 javascript vd 1



Conclusão

Compreender e utilizar estruturas condicionais é essencial para a programação. Elas nos permitem tomar decisões com base em condições lógicas, tornando nosso código mais dinâmico e eficiente. Além disso, ferramentas como `localStorage` e `setInterval` ampliam as possibilidades de interação e funcionalidade das nossas aplicações.

1.2. Continuação Estrutura Condicionais


Introdução

Vamos continuar nosso projeto da aula 4 e explorar mais algumas coisas sobre estruturas condicionais em JavaScript. Dentro dos parênteses do `if`, sempre teremos um teste lógico. Esse teste lógico compara um valor com outro valor usando operadores. Esses operadores têm tipos diferentes, mas não da mesma forma que os tipos de dados.

Operadores de Atribuição

Operadores de atribuição são usados para atribuir valores a variáveis. O operador mais comum é o `=`, que atribui um valor a uma variável.

javascript

 Copiar código

```
let numero = 2; numero += 2; // número agora é 4
```

Operadores de Incremento e Decremento

Você pode incrementar ou decrementar o valor de uma variável usando os operadores `++` e `--`. Esses operadores são atalhos para operações mais longas.

javascript

 Copiar código

```
let numero = 2; numero += 2; // número agora é 4 numero -= 2; // número agora é 2
```

Operadores Aritméticos

Os operadores aritméticos são usados para realizar operações matemáticas:

- `+` : Adição
- `-` : Subtração
- `*` : Multiplicação
- `/` : Divisão
- `%` : Módulo (resto da divisão)

javascript

 Copiar código


```
let soma = 2 + 3; // 5 let subtracao = 5 - 2; // 3 let multiplicacao = 2 * 3; // 6 let divisao = 6 / 2; // 3 let modulo = 5 % 2; // 1
```

Operadores de Comparação

Operadores de comparação são usados para comparar dois valores:

- `==` : Igual a (compara valor)
- `===` : Estritamente igual a (compara valor e tipo)
- `!=` : Diferente de (compara valor)
- `!==` : Estritamente diferente de (compara valor e tipo)
- `>` : Maior que
- `<` : Menor que
- `>=` : Maior ou igual a
- `<=` : Menor ou igual a

javascript

 Copiar código


```
console.log(2 == "2"); // true console.log(2 === "2"); // false console.log(2 != "2"); // false console.log(2 !== "2"); // true console.log(2 > 1); // true console.log(2 < 3); // true console.log(2 >= 2); // true console.log(2 <= 1); // false
```

Operadores Lógicos

Operadores lógicos são usados para combinar expressões booleanas:

- `&&` : E lógico (AND)
- `||` : Ou lógico (OR)
- `!` : Não lógico (NOT)

javascript


 Copiar código

```
let a = true; let b = false; console.log(a && b); // false console.log(a || b); // true console.log(!a); // false
```

Exemplo Prático de Estruturas Condicionais

Vamos criar um exemplo prático usando `if`, `else if` e `else` para testar números.

javascript

 Copiar código

```
let numero = parseInt(prompt("Digite um número:")); if (numero > 5) { console.log("O número é maior que 5"); } else if (numero < 5) { console.log("O número é menor que 5"); } else { console.log("O número é igual a 5"); }
```

Armazenamento Local (Local Storage)

O `localStorage` permite armazenar dados no navegador de forma persistente.

Armazenando Dados

javascript

 Copiar código

```
localStorage.setItem("nome", "Gleidson");
```

Recuperando Dados

javascript

 Copiar código

```
let nome = localStorage.getItem("nome"); console.log(nome); // Gleidson
```

Função `setInterval`

A função `setInterval` executa uma função repetidamente em intervalos de tempo especificados.

Exemplo de `setInterval`

javascript

 Copiar código

```
function mostrarHora() { let agora = new Date(); document.getElementById("relógio").innerHTML = agora.toLocaleTimeString(); } setInterval(mostrarHora, 1000);
```

Referência MDN

A documentação MDN (Mozilla Developer Network) é uma excelente fonte para aprender mais sobre JavaScript. Ela fornece tutoriais, exemplos e explicações detalhadas sobre várias funcionalidades da linguagem.

Link para a Documentação MDN

dia 12 javascript vd 2



Conclusão

Compreender e utilizar estruturas condicionais é essencial para a programação. Elas nos permitem tomar decisões com base em condições lógicas, tornando nosso código mais dinâmico e eficiente. Além disso, ferramentas como `localStorage` e `setInterval` ampliam as possibilidades de interação e funcionalidade das nossas aplicações.

Espero que essa apostila ajude a consolidar o conhecimento sobre esses conceitos importantes em JavaScript. Vejo vocês no próximo encontro!



934A8.96 8.96 0 0 0 2112a8.96 8.96 0 0 0 -1.085-4.287 11 0 0 1 .402-1.356M15.799 7.9a11 0 0 1 1.4.2 6.48 6.48 0 0 1 1.3 3.9c0 1.313-.39 2.537-1.06 3.56a11 0 0 1 -1.673-1.096A4.47 4.47 0 0 0 16.5 12a4.47 4.47 0 0 0 -.9-2.7 11 0 0 1 .2-1.4" clip-rule="evenodd">



4o



1.3. Switch Case

Introdução


Dando continuidade ao nosso estudo de JavaScript, hoje aprenderemos sobre outra estrutura condicional: o `switch case`. Na aula anterior, vimos as estruturas condicionais `if` e `else if`, onde realizamos testes lógicos que, se retornassem verdadeiro, executavam um bloco de código específico.

Estrutura `switch case`

A estrutura `switch case` é utilizada quando temos um conjunto de opções definidas e queremos executar diferentes blocos de código com base no valor de uma variável.

Sintaxe Básica do `switch case`

javascript

 Copiar código

```
switch (expressao) { case valor1: // Bloco de código break; case valor2: // Bloco de código break; default: // Bloco de código }
```

Exemplo Prático

Vamos criar um exemplo prático utilizando `switch case`.

1. Crie um novo arquivo HTML e conecte-o com um arquivo JavaScript.
2. No arquivo JavaScript, vamos usar `switch case` para mudar a cor de fundo da página com base no valor de uma variável `modo`.

HTML


html

 Copiar código

```
<!DOCTYPE html> <html lang="pt-br"> <head> <meta charset="UTF-8"> <title>Exemplo Switch Case</title> <script src="script.js" defer></script> </head> <body> </body> </html>
```

JavaScript

javascript

 Copiar código

```
let modo = prompt("Digite um modo: dark, red ou blue:"); switch (modo) { case "dark": document.body.style.backgroundColor = "black"; document.body.style.color = "white"; break; case "red": document.body.style.backgroundColor = "red"; document.body.style.color = "white"; break; case "blue": document.body.style.backgroundColor = "blue"; document.body.style.color = "white"; break; default: document.body.style.backgroundColor = "pink"; document.body.style.color = "black"; }
```

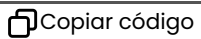
Explicação do Exemplo

1. Solicitamos ao usuário que insira um modo (`dark`, `red` ou `blue`) através do `prompt`.
2. A variável `modo` armazena a escolha do usuário.
3. O `switch` avalia o valor da variável `modo`:
 - Se `modo` for `"dark"`, o fundo da página será preto e o texto será branco.
 - Se `modo` for `"red"`, o fundo da página será vermelho e o texto será branco.
 - Se `modo` for `"blue"`, o fundo da página será azul e o texto será branco.
 - Se `modo` não corresponder a nenhum dos casos, o fundo da página será rosa e o texto será preto (`default`).

Importância do `break`

O `break` é essencial para interromper a execução do `switch case` após encontrar uma correspondência. Sem o `break`, o JavaScript continuará a executar os casos seguintes, mesmo que eles não correspondam.

javascript



Copiar código

```
switch (modo) { case "dark": document.body.style.backgroundColor = "black"; document.body.style.color = "white"; // Sem o
break, ele continuará executando os próximos casos case "red": document.body.style.backgroundColor = "red";
document.body.style.color = "white"; // Sem o break, ele continuará executando os próximos casos case "blue":
document.body.style.backgroundColor = "blue"; document.body.style.color = "white"; // Sem o break, ele continuará executando
os próximos casos default: document.body.style.backgroundColor = "pink"; document.body.style.color = "black"; }
```

Uso do default

O **default** é utilizado para definir um bloco de código a ser executado quando nenhuma das condições anteriores for atendida. Ele é semelhante ao **else** em uma estrutura **if else**.

javascript



Copiar código

```
default: document.body.style.backgroundColor = "pink"; document.body.style.color = "black";
```

dia 12 javascript vd 3



Conclusão

O **switch case** é uma estrutura condicional poderosa que simplifica a tomada de decisões quando há múltiplas opções possíveis. Ele é ideal para situações onde temos um conjunto definido de valores e queremos executar diferentes blocos de código com base nesses valores.

Vejo vocês no nosso próximo encontro para explorarmos mais sobre JavaScript!

1.4. Operador Ternário


Introdução

Continuando nosso aprendizado sobre JavaScript, vamos falar sobre uma última estrutura condicional. Até agora, aprendemos sobre `if`, `else if`, `else` e `switch case`. Essas estruturas nos permitem fazer decisões baseadas em testes lógicos. Hoje, apresento a vocês o operador ternário, uma forma compacta de escrever condicionais.

Operador Ternário

O operador ternário é uma maneira mais concisa de realizar testes lógicos em JavaScript. Ele é útil quando precisamos fazer testes simples e rápidos. A sintaxe do operador ternário é:

javascript

 Copiar código

```
condição ? expressão1 : expressão2
```

- **condição:** A expressão lógica a ser avaliada.
- **expressão1:** O resultado se a condição for verdadeira.
- **expressão2:** O resultado se a condição for falsa.

Exemplo Prático

Vamos usar o operador ternário com a variável `modo`.

javascript

 Copiar código

```
let modo = "dark"; // Usando operador ternário para verificar se a variável está definida let mensagem = modo ? "Está definida" : "Não está definida"; console.log(mensagem); // Está definida
```


Se a variável `modo` tiver um valor, a mensagem será "Está definida". Caso contrário, será "Não está definida".

Comparação com `if...else`

Vamos ver como o operador ternário simplifica a escrita em comparação com a estrutura `if...else`.

Usando `if...else`


javascript

 Copiar código

```
let modo = "dark"; let mensagem; if (modo) { mensagem = "Está definida"; } else { mensagem = "Não está definida"; } console.log(mensagem); // Está definida
```

Usando Operador Ternário

javascript

 Copiar código

```
let modo = "dark"; let mensagem = modo ? "Está definida" : "Não está definida"; console.log(mensagem); // Está definida
```

Valores Falsos e Verdadeiros

Em JavaScript, alguns valores são considerados "falsos" por padrão. Conhecer esses valores é importante para entender como os testes lógicos funcionam.

Valores Falsos


- `false`
- `0`
- `""` (string vazia)

- `null`
- `undefined`
- `NaN` (Not a Number)

Valores Verdadeiros

- Strings não vazias
- Números diferentes de zero
- Arrays
- Objetos

javascript

 Copiar código

```
console.log(Boolean(0)); // false console.log(Boolean("")); // false console.log(Boolean("texto")); // true
console.log(Boolean(123)); // true console.log(Boolean([])); // true console.log(Boolean({})); // true
```

Exemplos Adicionais com Operador Ternário

Podemos usar o operador ternário para outras situações além de verificar se uma variável está definida.

Exemplo 1: Verificar Igualdade


javascript

 Copiar código

```
let modo = "dark"; let mensagem = (modo === "dark") ? "Modo escuro ativado" : "Modo claro ativado"; console.log(mensagem);
// Modo escuro ativado
```

Exemplo 2: Verificar Número

javascript

 Copiar código

```
let numero = 5; let tipoNumero = (numero > 0) ? "Positivo" : (numero < 0) ? "Negativo" : "Zero"; console.log(tipoNumero); //
Positivo
```

dia 12 javascript vd 4



Conclusão

O operador ternário é uma ferramenta poderosa para escrever condicionais de maneira concisa e eficiente em JavaScript. Ele é especialmente útil para testes lógicos simples e rápidos. Conhecer os valores considerados "falsos" e "verdadeiros" em JavaScript é essencial para utilizar corretamente essa estrutura.

Espero que essa aula tenha ajudado a entender como utilizar o operador ternário em seus códigos JavaScript. Vejo vocês no próximo encontro para continuar nosso aprendizado!