

Aula 14

Conversão de Tipos e Tratamento de Erros

Slide 1: O Que Vamos Aprender Hoje? (Parte 1)

Tema: Conversão e Comparação de Tipos

- Comparação Solta (`==`) vs. Estrita (`===`): Entender a diferença crucial que evita muitos bugs.
- O Perigo da Conversão Automática: Como o JavaScript pode te surpreender ao comparar `número` com `texto`.
- A Regra de Ouro: Por que `===` é seu melhor amigo para comparações seguras.

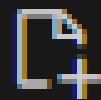
Slide 2: O Que Vamos Aprender Hoje? (Parte 2)

Tema: Tratamento de Erros

- Evitando a "Tela Branca": Como impedir que um erro quebre todo o seu programa.
- Sua Rede de Segurança (`try...catch`): A estrutura para "tentar" um código e "capturar" possíveis falhas.
- Criando Nossas Próprias Regras (`throw`): Como lançar seus próprios erros para validar dados em funções.
- Objetivo Final: Escrever um código mais forte, seguro e profissional.

Prática

✓ CODIGO



JS conversao.js


JS erros.js

<> index.html

<> index.html X

JS erros.js

JS conversao.js

<> index.html >  html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <h1>Conversão de Tipos e Tratamento de Erros</h1>
10
11      <script src="conversao.js"></script>
12      <script src="erros.js"></script>
13  </body>
14  </html>
```

JS conversao.js > ...

```
1  // AULA BÁSICA: Conversão e Erros
2  // Foco: Os conceitos mais fundamentais de forma isolada.
3  console.log("--- JS: aula-basica.js ---");
4
5  // =====
6  // EXEMPLO 1: A DIFERENÇA MAIS IMPORTANTE - '==' vs '==='
7  // =====
8  // Objetivo: Entender como o JavaScript compara valores.
9
10 console.log("\n--- EXEMPLO 1: '==' vs '===' ---");
11
12 let numero = 5;
13 let texto = "5";
14
15 console.log("Temos um número:", numero);
16 console.log("Temos um texto (string):", texto);
17
18 // Usando '==' (Comparação "solta"):
19 // O '==' tenta converter os tipos para serem iguais ANTES de comparar.
20 // Aqui, ele converte o texto "5" para o número 5 e depois compara.
21 console.log("\nComparando com '==' (tenta converter):");
22 console.log("numero == texto:", numero == texto); // true
23
24 // Usando '===' (Comparação "estrita"):
25 // O '===' é mais seguro! Ele NÃO converte os tipos.
26 // Ele compara o valor E o tipo. Se os tipos forem diferentes, o resultado é falso.
27 console.log("\nComparando com '===' (NÃO converte):");
28 console.log("numero === texto:", numero === texto); // false
29
30 // REGRA BÁSICA: Use SEMPRE '===' para evitar surpresas.
31
```

JS erros.js > ...

```
1  // EXEMPLO 2: CAPTURANDO UM ERRO COM 'try...catch'
2  // =====
3  // Objetivo: Evitar que o programa quebre quando um erro acontece.
4
5  console.log("\n--- EXEMPLO 2: Básico do 'try...catch' ---");
6
7  //funcaoQueNaoExiste();
8
9  console.log('continuação')
10
11  try {
12      // 1. O bloco 'try' tenta executar um código.
13      console.log("Vou tentar chamar uma função que não existe...");
14      funcaoQueNaoExiste(); // Este comando vai gerar um erro.
15
16      // Como o erro acontece na linha acima, esta linha nunca será executada.
17      console.log("Esta mensagem não vai aparecer.");
18
19  } catch (erro) {
20      // 2. Se um erro ocorrer no 'try', o bloco 'catch' é executado.
21      // A variável 'erro' contém informações sobre o que deu errado.
22      console.log("Opa! Um erro foi capturado com sucesso!");
23      console.log("Mensagem do erro:", erro.message); // Exibe a mensagem do erro.
24  }
25
26  console.log("\nO programa não quebrou e continuou executando. Isso é ótimo!");
27
```



```
30 // EXEMPLO 3: CRIANDO UM ERRO COM 'throw'
31 // =====
32 // Objetivo: Criar nossas próprias condições de erro dentro de uma função.
33
34 console.log("\n--- EXEMPLO 3: Básico do 'throw' ---");
35
36 function verificarIdade(idade) {
37     // Verificamos se a idade é válida.
38     if (idade < 18) {
39         // Se a idade for menor que 18, nós "lançamos" um erro.
40         // Isso interrompe a função e pode ser capturado por um 'try...catch'.
41         throw new Error("Acesso negado. A pessoa é menor de idade.");
42     }
43
44     // Este código só é executado se nenhum erro for lançado.
45     console.log("Idade válida. Acesso permitido.");
46 }
47
48 // Usando a função dentro de um 'try...catch'.
49
50 // Teste 1: Idade válida
51 try {
52     verificarIdade(25); // Isso não vai gerar erro.
53 } catch (e) {
54     console.log("ERRO CAPTURADO:", e.message);
55 }
56
57 // Teste 2: Idade inválida
58 try {
59     verificarIdade(15); // Isso VAI gerar um erro.
60 } catch (e) {
61     // O erro lançado pela função será capturado aqui.
62     console.log("ERRO CAPTURADO:", e.message);
63 }
64
65 console.log("\nFim da aula básica!");
```

Atividade

Exercício 1: Preveja a Comparação 😞

- Arquivo: `aula_basica/js/exercicios/exercicio1-comparacao.js`
- Enunciado:
 1. Crie um script que exiba no console o resultado (`true` ou `false`) das seguintes comparações.
 2. Antes de rodar o código, tente adivinhar o resultado e adicione um comentário no código explicando o porquê de cada resultado.
 - `console.log(0 == "");`
 - `console.log(0 === "");`
 - `console.log(1 == true);`
 - `console.log(1 === true);`
 - `console.log(null == undefined);`
 - `console.log(null === undefined);`

Exercício 2: Código à Prova de Falhas 🛡️

- Arquivo: `aula_basica/js/exercicios/exercicio2-try-catch.js`

- Enunciado:

1. O código abaixo vai gerar um erro porque a variável `aluno` é um objeto, mas tentamos acessá-la como uma função.

JavaScript



```
const aluno = { nome: "Carlos" };  
aluno(); // Isso vai gerar um TypeError
```

2. Seu trabalho é colocar esse código dentro de um bloco `try...catch`.
 3. No bloco `catch`, exiba uma mensagem amigável no console, como: "Ocorreu um erro, mas o programa não parou!" e também exiba a mensagem do erro (`erro.message`).
 4. Adicione um `console.log("Fim do programa.");` após o bloco `try...catch` para provar que a execução continuou.
-

Exercício 3: Validação de Nota 📝

- Arquivo: `aula_basica/js/exercicios/exercicio3-throw.js`
- Enunciado:
 1. Crie uma função chamada `avaliarNota(nota)`.
 2. Dentro da função, verifique se a `nota` está fora do intervalo de 0 a 10.
 3. Se a nota for inválida (menor que 0 ou maior que 10), a função deve lançar um erro com a mensagem "Nota inválida!".
 4. Se a nota for válida, a função deve apenas exibir no console "Nota válida.".
 5. Chame a função `avaliarNota` duas vezes dentro de blocos `try...catch`: uma vez com uma nota válida (ex: 8) e outra com uma nota inválida (ex: 11), tratando os possíveis erros.

Adapte o exercício 3
para receber a nota de
um formulário HTML
(DOM)