Lógica de Programação - Aula 8

Santander Coders 2024

Try-Catch

Try- Catch

O conceito de exceções refere-se a situações anormais ou imprevistas que podem ocorrer durante a execução de um programa. Essas situações geralmente representam erros ou condições inesperadas que podem comprometer a execução normal do código. Em linguagens de programação, como JavaScript, o tratamento de exceções é uma abordagem fundamental para lidar com essas situações e manter a robustez do programa.

A execução de um programa pode ser interrompida por diversas razões, tais como:

Erros de Sintaxe: O código pode conter erros de sintaxe que impedem sua interpretação correta pelo interpretador da linguagem.

Erros de Tempo de Execução: São erros que ocorrem durante a execução do programa, como divisões por zero, acesso a propriedades inexistentes, entre outros.

Condições Inesperadas: Situações que não foram previstas durante o desenvolvimento do programa, como falhas de rede, falta de permissões, entre outras.

Try Catch

O tratamento de exceções permite que os desenvolvedores capturem esses erros e tomem medidas apropriadas para lidar com eles, evitando que o programa seja encerrado de forma abrupta. Em JavaScript, a estrutura básica para o tratamento de exceções é construída em torno das palavras-chave try, catch, e finally.

O bloco **try** contém o código que pode gerar uma exceção.
O bloco **catch** captura e trata a exceção caso ela ocorra.
O bloco **finally** contém código que será executado independentemente de ocorrer ou não uma exceção.

Ao lidar com exceções, os desenvolvedores podem fornecer mensagens de erro significativas, registrar informações detalhadas sobre o problema e, em alguns casos, realizar ações corretivas para mitigar os impactos da exceção. Isso contribui para a criação de programas mais robustos e resistentes a situações inesperadas, melhorando a experiência do usuário e facilitando a manutenção do código.

```
try {
    // Código que pode gerar exceções
    let resultado = 10 / 0; // Tentativa de divisão por zero
} catch (erro) {
    // Bloco de código para lidar com a exceção
    console.error("Ocorreu um erro: " + erro.message);
}
```

```
try {
    // Código que pode gerar exceções
    let resultado = 10 / 0; // Tentativa de divisão por zero
} catch (erro) {
    // Bloco de código para lidar com a exceção
    console.error("Ocorreu um erro: " + erro.message);
} finally {
    // Bloco de código que será executado sempre
    console.log("Bloco finally executado.");
}
```

Forçar um erro

```
function dividir(x, y) {
    if (y === 0) {
        throw new Error("Não é possível dividir por zero.");
    }
    return x / y;
}

try {
    let resultado = dividir(10, 0);
    console.log("Resultado: " + resultado);
} catch (erro) {
    console.error("Ocorreu um erro: " + erro.message);
}
```

Bora Praticar

Teste de Lógica 1

01 - Dados cinco números inteiros positivos, encontre os valores mínimo e máximo que podem ser calculados somando exatamente quatro dos cinco números inteiros. Em seguida, imprima os respectivos valores mínimo e máximo como uma única linha de dois inteiros separados por espaço.

Exemplo:

arr - [1,3,5,7,9]

a soma mínima é: 1 + 3 + 5 + 7 = 16
a soma máxima é: 3 + 5 + 7 + 9 = 24

Saída: [16,24]

Teste:

arr = [5,1,6,3,9] ⇒ Saída ⇒ [15, 23]
arr = [9,3,3,5,8] ⇒ Saída ⇒ [19, 25]

02 - Detalhe da escada: Esta é uma escada de tamanho n=4

```
   #
  ##
 ###
####
```

sua base e altura são iguais a **n**. É desenhado usando os símbolos **#** e **espaços**. A última linha não é precedida por nenhum espaço. Escreva um programa que imprima uma escada do tamanho N.

Exemplo de entrada: n ⩾ 1

1 ou 4 ou 5 ou 7

Entrada:6 Saída:

```
     #
    ##
   ###
  ####
 #####
######
```