1. Definições Básicas

**Erro** - Falha ou engano que demonstra falta dos conhecimentos

**Exceção** - Algo excluído da maioria, que não pertence ao todo

**Diferença entre “erro” em linguagem natural e “exceção” em JavaScript** - O tratamento de exceções em JavaScript é uma ferramenta importante para criar código mais robusto e que possa lidar com situações inesperadas sem interromper a execução do programa. O tratamento de erros, por outro lado, envolve a detecção e correção de erros no código assim parando o código

1. **Erros sem Exceções?**

Quando foi usado uma String sem os parênteses

1. **]Confiabilidade Limitada**

Que se espera um numero deve se certificar de que tudo está tudo certo usando por exemplo o typeof

1. **Tipos de Erros em JS**

**SyntaxError** Erro de sintaxe aparece quando um código está malformado, quando há erros de digitação nas palavras-chave, parênteses ou colchetes incompatíveis, etc.

**ReferenceError:** Ocorre quando tentamos acessar uma função ou variável que não existe.

**TypeError :** Este tipo de erro ocorre quando um determinado valor não é do tipo esperado

1. **Try…Catch Básico**

function safeParse(jsonString) {

try {

return JSON.parse(jsonString);

} catch (erro) {

console.log("Ocorreu um erro ao converter JSON:", erro.message);

return null;

}

}

console.log(safeParse('{"nome": "tutty"}'));

console.log(safeParse('texto inválido'));

1. **Tratamento Condicional de Exceções**

function safeParse(jsonString) {

try {

return JSON.parse(jsonString);

} catch (err) {

if (err instanceof SyntaxError) {

} else {

throw err;

}

}

}

console.log(safeParse('{"nome": "tutty"}'));

console.log(safeParse('texto inválido'));

1. **Bloco Finally**

function safeParse(jsonString) {

try {

return JSON.parse(jsonString);

} catch (err) {

if (err instanceof SyntaxError) {

} else {

throw err;

}

}

finally {

console.log("Parse attempt finished")

}

}

console.log(safeParse('{"nome": "tutty"}'));

console.log(safeParse('texto inválido'));

1. **Lançando Erros Customizados**

class InvalidAgeError extends Error {

}

function checkAge(age) {

if (age < 0 || age > 120) {

throw new InvalidAgeError("Idade fora do intervalo (0–120)");

}

return "Idade válida";

}

console.log(checkAge(30));

console.log(checkAge(-5));

console.log(checkAge(200));

1. **Depuração com console.log**

Pois ele dá um valor apenas para a variável A e deixa a variável B como indefinida por isso que da esse erro.

1. **Preparação do Ambiente de Depuração**

Iria clicar na tecla f12 para abrir o depurador

1. Estava na pagina console testando o código e quando dei o enter fui redirecionada para a pagina Sources onde apareceu varias informações inclusive o valor do x e do y.
2. **Step Into:** Isso executa apenas a próxima instrução no código e a pausa novamente.

**Step Over:** Funciona como Step Into, exceto que, se for usado quando a próxima instrução for uma chamada de função, o código não pulará para o código da função

**Step Ou:** Isso nos permite sair imediatamente de uma função na qual o código está pausado.

1. **Call Stack**

Mostra que o externo foi chamado primeiro, e dentro dele, foi chamado o interno que está sendo executado antes do primeiro

1. **Depuração Sem debugger**