



Exception

R esumo

Em JavaScript, uma exceção é um objeto que representa uma condição anormal que ocorre durante a execução de um programa. Quando um erro ou uma exceção é lançado, interrompe o fluxo normal de execução e é tratado por um bloco de código especial chamado de bloco ‘try-catch’.

Aqui está um exemplo básico de como usar o bloco “try-catch” em JavaScript:

```
```javascript
try {
 // Código que pode gerar uma exceção
 throw new Error("Opa, algo deu errado!");
} catch (e) {
 // Código para tratar a exceção
 console.log("Ocorreu uma exceção: " + e.message);
}
```
```

No exemplo acima, estamos usando a palavra-chave “throw” para lançar uma exceção do tipo “Error” com a mensagem “Opa, algo deu errado!”. Em seguida, capturamos a exceção no bloco “catch” e imprimimos a mensagem de erro no console.

Além de lançar exceções explicitamente com “throw”, várias operações em JavaScript podem gerar exceções automaticamente. Por exemplo, ao tentar acessar uma propriedade inexistente de um objeto ou chamar uma função que não foi definida, ocorrerão exceções.

A utilização de exceções é útil para lidar com erros e situações excepcionais de forma estruturada. Ao envolver o código que pode gerar uma exceção no bloco “try”, você tem a oportunidade de capturar e tratar a exceção no bloco “catch”. Isso permite que você tome medidas apropriadas, como exibir mensagens de erro, registrar informações relevantes, tomar ações alternativas ou simplesmente interromper a execução de forma controlada.


Você também pode usar os blocos “finally” e “throw” em conjunto com o bloco “try-catch” para ter um controle mais refinado sobre o fluxo de execução e garantir que determinadas ações sejam sempre realizadas, independentemente de uma exceção ser lançada ou não.

Em resumo, as exceções em JavaScript fornecem um mecanismo para lidar com erros e situações excepcionais durante a execução do seu código, permitindo que você escreva um código mais robusto e resiliente.

Para ampliar o controle sobre o fluxo de execução do código, JavaScript fornece um bloco adicional chamado ‘finally’ que pode ser anexado após o bloco ‘try-catch’. O bloco ‘finally’ é executado independentemente de uma exceção ter sido lançada ou capturada.

Aqui está um exemplo de como usar o bloco ‘finally’:

Resumo precisa ter, no mínimo, 2 páginas.




```
try {
  // Código que pode gerar uma exceção
  throw new Error("Opa, algo deu errado!");
} catch (e) {
  // Código para tratar a exceção
  console.log("Ocorreu uma exceção: " + e.message);
} finally {
  // Este bloco de código será sempre executado
  console.log("Finalizando a execução...");
}
```

Fonte: autoral

Neste exemplo, se uma exceção é lançada no bloco 'try' e capturada no bloco 'catch', o bloco 'finally' ainda é executado. Mesmo se nenhuma exceção é lançada, o bloco 'finally' será executado, tornando-se um local perfeito para a limpeza de código ou outras tarefas que devem ser executadas independentemente do sucesso ou falha da execução.

Outro aspecto importante das exceções em JavaScript é a habilidade de definir e lançar tipos personalizados de exceções, além das exceções padrão, como o tipo 'Error'. Isso permite que você crie um sistema mais detalhado e específico de tratamento de erros, que pode diferenciar entre diferentes tipos de exceções e responder adequadamente a cada uma delas.^[1]

Por exemplo, você pode definir uma exceção 'InvalidUserInputException' para casos em que a entrada do usuário não é válida e uma exceção 'DatabaseException' para erros que ocorrem ao interagir com um banco de dados.

O tratamento de exceções fornece uma maneira eficiente de lidar com erros e situações inesperadas que podem ocorrer em seu código. 

Ao capturar e tratar exceções, você pode evitar o comportamento indesejado ou a falha de seu programa e garantir que ele continue funcionando corretamente sob diferentes circunstâncias. [\[2\]](#)

Conteúdo Bônus

Aqui está um exemplo de função que verifica uma regra de negócio para a compra de um carro e lança uma exceção se a condição não for atendida:


```
```javascript
function comprarCarro(idade, possuiCarteira) {
 if (idade < 18) {
 throw new Error("Você deve ter pelo menos 18 anos para comprar um carro.");
 }

 if (!possuiCarteira) {
 throw new Error("Você precisa ter uma carteira de motorista para comprar um carro.");
 }

 // Restante da lógica de compra do carro...
 console.log("Parabéns! Compra de carro realizada com sucesso.");
}

try {
 comprarCarro(20, true);
} catch (error) {
 console.log("Ocorreu uma exceção: ", error.message);
}
...
```
```

Nesse exemplo, a função `comprarCarro` recebe dois parâmetros: `idade` e `possuiCarteira`. A função verifica se a idade é maior ou igual a 18 anos e se a pessoa possui carteira de motorista. Se alguma dessas condições não for atendida, a função lança uma exceção usando a palavra-chave `throw` e uma mensagem de erro específica.

No bloco try, chamamos a função comprarCarro com valores que atendem às regras de negócio (idade de 20 anos e carteira  motorista). Se ocorrer uma exceção dentro da função, o fluxo de execução será interrompido e capturado no bloco catch, onde podemos tratar a exceção e exibir a mensagem de erro no console.

Você pode adaptar essa lógica de regra de negócio de compra de carro de acordo com os requisitos específicos do seu programa. O uso de exceções nesse contexto permite que você identifique e lide com possíveis problemas ou condições inválidas durante o processo de compra.

Referência Bibliográfica

FLANAGAN, David. **JavaScript: O Guia Definitivo**. 6ª Ed. Porto Alegre: Bookman, 2013.

FREEMAN, Eric. **Use a cabeça!: programação JavaScript**. 1ª Ed. São Paulo: Alta Books, 2016.

Ir para exercício