Tratamento de exceções

O tratamento de exceções no JavaScript ou em qualquer outra linguagem de programação é o recurso que permite que os erros de execução do software sejam tratados antes que o programa quebre e fique sem condições de continuar sendo executado para o usuário.

O erro pode ocorrer por algum problema de digitação durante o desenvolvimento, um comportamento inesperado do servidor, uma entrada errada do usuário, uma requisição mal formatada e por tantos outros motivos que poderíamos descrever aqui.

Try Catch

O Try Catch é o recurso que utilizamos para tratar os erros de execução do nosso software.

A sintaxe do try catch é composta por dois blocos principais: O try onde o código é executado; e o catch, onde você recebe, via parâmetro na função, um objeto do tipo Error. Com este objeto você pode aplicar o tratamento que for adequado. Exemplo:

```
try {
  // seu código aqui
} catch(error) {
  // tratamento de erro aqui
}
```

Você também pode estender o try catch usando a cláusula finally. Este bloco será executado independente se houver ou não falha, ou seja, depois que o `try` ou o `catch` executar, este bloco será acionado. Isto pode ser útil, por exemplo, para fechar um arquivo que foi aberto para leitura, registrar algum log ou fechar alguma conexão. Veja a sintaxe de exemplo.

```
try {
  // seu código aqui
} catch (error) {
  // tratamento de erro aqui
} finally {
  // executa sempre
}
```

Entendendo o objeto Error

Quando alguma falha é detectada dentro do bloco `try` o JavaScript cria um objeto do tipo Error, que é enviado como parâmetro para o `catch`. O objeto Error, por padrão, possui duas propriedades: `name` e `message`.

A propriedade `name` é, de fato, o nome do erro. Por exemplo, um erro de sintaxe geraria um objeto com a propriedade `name` com o valor SyntaxError.

A propriedade `message` é um texto contendo a descrição detalhada do erro.

O JavaScript possui os seguintes erros nativos:

ReferenceError:
Lançado quando uma referência a uma variável ou função inexistente ou inválida é detectada.

TypeError:
Lançado quando um operador ou argumento passado para a função é de um tipo diferente do esperado.

SyntaxError:
Lançado quando ocorre algum erro de sintaxe ao interpretar o código, por exemplo, ao realizar o parse de um JSON.

URIError:
Lançado quando ocorre algum erro no tratamento de URI, por exemplo, enviando parâmetros inválidos no decodeURI() ou encodeURI().

RangeError:
Lançado quando um valor não está no conjunto ou intervalo de valores permitidos, por exemplo, um valor em string num array número.

Operador throw

Quando precisamos de uma exeção que não esteja revista no JavaScript, nós podemos lançar a nossa própria exeção. Para isso, nós usamos o operador `throw`. No código abaixo, vamos simular uma validação de CPF, e caso o CPF não seja validado, lançaremos uma exeção com o `throw`.

```
try{
  var cpfValido = false; //imagine aqui uma chamada para uma função que vali

  if (!cpfValido){
    throw new Error('O CPF informado não é válido!');
  }
} catch(error){
  console.log(error.message);
}
```

Perceba que ao executar o código acima, usamos o `throw` para lançar uma nova exceção e personalizamos o nosso parâmetro `message` do objeto Error. Quando o bloco `catch` for executado, a frase "O CPF informado não é válido!" será exibida na tela, porque assim a definimos como propriedade do objeto Error.