Boas-vindas!

Benvindo caro aspirante a dev!
Nesta aula, você irá aprender um pouco sobre funções recursivas em javascript.

Ao final deste módulo você deverá:

- Entender a estrutura de uma função e os seus propósitos;
- Entender função recursiva;
- Capturar erros e emiti-los utilizando uma 'template string'.

Referências básicas:

- Template string
  - https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Template_literals
- Function
  - https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Functions
- Recursive function
  - https://developer.mozilla.org/en-US/docs/Glossary/Recursion
  - https://linuxhint.com/javascript-recursive-function/
  - https://www.javascripttutorial.net/javascript-recursive-function/
  - https://www.freecodecamp.org/news/what-is-recursion-in-javascript/

Exercícios:

1. Crie uma função em JavaScript que permita a soma de dois números inteiros positivos. Caso não seja possível a soma, emitir um erro (throw) do tipo `[sum] Impossible to sum ${num1} + ${num2}` ('template string' no qual num1 é o primeiro parâmetro e num2 é o segundo parâmetro);
2. Utilizando a função do exercício 1, crie uma função em JavaScript que permita a subtração de dois números inteiros positivos, considerando:
   a. O primeiro número é maior ou igual ao segundo número;
   b. Não se deve utilizar estrutura de looping (for, while, do ...);
   c. Não se deve utilizar Math ou funções de terceiros;
   d. Não se deve utilizar operações de soma e subtração ('+' ou '-'), isto é, se desejar somar, utilizar a função de soma criada no exercício 1;
   e. Caso seja impossível realizar subtração (capturar o erro se existir), deve-se emitir um erro (throw) do tipo `[subtract] Impossible to subtract ${num1} - ${num2}`;
   f. Dica: utilizar chamada de função recursiva.
3. Utilizando as funções do exercício 1 e 2, crie uma função que multiplique dois valores inteiros positivos, de forma que:
   a. Não se deve utilizar estrutura de looping (for, while, do ...);
   b. Não se deve utilizar Math ou funções de terceiros;
   c. Não se deve utilizar operações de soma e subtração ('+' ou '-'), isto é, se desejar somar ou subtrair, utilizar as funções criadas nos exercícios 1 e 2;
   d. Caso seja impossível realizar soma ou subtração (capturar o erro se existir), deve-se emitir um erro (throw) do tipo `[multiply] Impossible to multiply ${num1} * ${num2}`
   e. Dica: utilizar chamada de função recursiva.
4. Utilizando as funções do exercício 1, 2 e 3, crie uma função que exponencie dois valores inteiros positivos, de forma que:
   a. Não se deve utilizar estrutura de looping (for, while, do ...);

   b. Não se deve utilizar Math ou funções de terceiros;
   c. Não se deve utilizar operações de soma, subtração e multiplicação (‘+’ ou ‘-‘ ou ‘*’), isto é, se desejar somar ou subtrair ou multiplicar, utilizar as funções criadas nos exercícios 1, 2 e 3;
   d. Caso seja impossível realizar soma ou subtração (capturar o erro se existir), deve-se emitir um erro (throw) do tipo `[multiply] Impossible to multiply ${num1} * ${num2}`
   e. Dica: utilizar chamada de função recursiva.
5. Utilizando as funções dos exercícios 1, 2 e 3, crie uma função exiba o valor inteiro da divisão de dois valores inteiros positivos, de forma que:
   a. Não se deve utilizar estrutura de looping (for, while, do ...);
   b. Não se deve utilizar Math ou funções de terceiros;
   c. Não se deve utilizar operações de soma, subtração e multiplicação (‘+’ ou ‘-‘ ou ‘*’), isto é, se desejar somar ou subtrair ou multiplicar, utilizar as funções criadas nos exercícios 1, 2 e 3;
   d. Caso seja impossível realizar soma ou subtração (capturar o erro se existir), deve-se emitir um erro (throw) do tipo `[divide] Impossible to divice ${num1} / ${num2}`, ou caso o denominador seja zero, `[divide] Division by zero`.
   e. Dica: utilizar chamada de função recursiva.

Em síntese:

Neste módulo você aprendeu um pouco sobre o uso de funções, principalmente funções recursivas. Também exercitou a emissão, captura e exibição de erros em JavaScript utilizando ‘try...catch’, ‘throw’ e ‘template strings’.