Funções em Javascript

Formas de declaração de uma função

```
// Função definida de forma direta
function exemplo1(args){
  //Código a ser executado
}

exemplo1(x); // Chamada da função

// Função por expressão não nomeada
const exemplo2 = function(args){
  // Código a ser executado
}

exemplo2(x); // Chamada da função

// Função por expressão nomeada
const exemplo3 = function ex3(args){
  // Código a ser executado
}

exemplo3(x); // Chamada da função

// Arrow Functions
const exemplo4 = (args) => retorno;

exemplo4(x); // Chamada da função
```

Parâmetros de uma função

- Uma função **pode** receber parâmetros em sua chamada e esses podem ser acessados dentro do bloco de código da função.
- Existe uma grande liberdade sobre o que pode ser passado como parâmetro para uma função, como: **variáveis, valores, arrays, objetos, outras funções**.
- **Parâmetro Default** (padrão) de uma função: Quando não passamos um dos parâmetros definidos em uma função, por padrão, ele recebe o valor `undefined` . Para alterar esse comportamento é possível atribuir um valor default a um parâmetro na definição da função.

```
function exemplo(a, b = 2){
  console.log(a*b);
}
```

```
exemplo(5, 3); // Imprime 15 (5*3)
exemplo(5); // Imprime 10 (5*2)
```

- **Parâmetro Rest** de uma função (`...arg`): O parâmetro Rest de uma função é definido por três pontos finais antecedendo o **último parâmetro** da função. Ele vai tratar todos os parâmetros passados na chamada da função que excedem o que foi definido como um Array.

```
function exemplo(a, ...b){
  console.log("valor 'a' = "+ a);
  for(let i = 0; i < b.lenght; i ++){
    console.log(b[i]);
  }
}

exemplo(5, 3, 4, 6, 7, 8); // Imprime valor 'a' = 5 e 3, 4, 6, 7, 8
exemplo(4, 5, 6); // Imprime valor 'a' = 4 e 5, 6
```

Observações

- Uma função pode ser definida dentro de outra função, dessa forma ela pode acessar os parâmetros da função pai.
- Uma função pode fazer uma chamada a si mesma dentro de seu bloco de código. **RECURSIVIDADE**