Funções

Bloco de código que pode ser executado e reutilizado.

Valores podem ser passados por uma função e a mesma retorna outro valor.

Function

```
function areaQuadrado(lado) {
    return lado * lado;
}
areaQuadrado(4) // 16
areaQuadrado(5) // 25
areaQuadrado(2) // 4
```

Função

```
function pi() {
    return 3.14;
}

var total = 5 * pi(); // 15.7
```

Parâmetros e Argumentos

Parâmetros

```
// peso e altura são os parâmetros
function imc(peso, altura) {
    const imc = peso / (altura ** 2);
    return imc;
}

imc(80, 1.80) // 80 e 1.80 são os argumentos
imc(60, 1.70) // 60 e 1.70 são os argumentos
```

Os parâmetros são separados por vírgulas

Fazer a chamada usando parênteses executa a função:

Executar função

```
function corFavorita(cor) {
    if(cor === 'azul') {
        return 'Você gosta do céu';
    } else if(cor === 'verde') {
        return 'Você gosta de mato';
    } else {
        return 'Você não gosta de nada';
    }
}
corFavorita(); // retorna 'Você não gosta de nada'
```

Argumentos podem ser funções

callbacks

```
addEventListener('click', function() {
    console.log('Clicou');
});
```

> A função possui dois argumentos:
>
> - Primeiro é a string 'click'
> - Segundo é uma função anônima

Chamadas de **Callback**, geralmente são funções que ocorrem após algum evento.

Funções anônimas são aquelas em que o nome da função não é definido, escritas como `function() {}` ou `() => { }`.

Pode ou não retornar um valor

```
function imc(peso, altura) {
    const imc = peso / (altura ** 2);
}

imc(80, 1.80); // retorna o imc
console.log(imc(80, 1.80)); // retorna undefined
```

Quando não definimos o `return`, ela irá retornar `undefined`.

O código interno da função é executado normalmente, independente de existir valor de return ou não.

Valores retornados

```
function terceiraIdade(idade) {
    if(typeof idade !== 'number') {
        return 'Informe a sua idade!';
    } else if(idade >= 60) {
        return true;
    } else {
        return false;
    }
}
```

Uma função pode retornar qualquer tipo de dado e até outras funções.

⚠ **Cuidado, retornar diferentes tipos de dados na mesma função não é uma boa prática.**

Escopo

Variáveis e funções definidas dentro de um bloco {} , não são visíveis fora dele.

Escopo

```
function exerciciosResolvidos(resolvidos) {
    var totalExercicios = 10;
    return `Ainda faltam ${totalExercicios - resolvidos} exercícios para resolver`;
}

console.log(totalExercicios); // erro, totalExercicios não definido
```

Escopo Léxico

Funções conseguem acessar variáveis que foram criadas no contexto pai

Escopo

```
var profissao = 'Desenvolvedor';

function dados() {
    var nome = 'Joãozinho';
    var idade = 28;
    function outrosDados() {
        var endereco = 'Rio de Janeiro';
        var idade = 29;
        return `${nome}, ${idade}, ${endereco}, ${profissao}`;
    }
    return outrosDados();
}

dados(); // Retorna 'Joãozinho, 29, Rio de Janeiro, Desenvolvedor'
outrosDados(); // retorna um erro
```

Hoisting

Antes de executar uma função, o JS 'move' todas as funções declaradas para a memória

Hoisting

```
imc(80, 1.80); // imc aparece no console
function imc(peso, altura) {
    const imc = peso / (altura ** 2);
    console.log(imc);
}
```

Exercício

- Crie uma função para verificar se um valor é `True`
- Crie uma função matemática que retorne o perímetro de um quadrado.

  Lembrando: perímetro é a soma dos quatro lados do quadrado
- Crie uma função que retorne o seu nome completo, ela deve possuir os parâmetros: nome e sobrenome
- Crie uma função que verifica se um número é par
- Crie uma função que retorne o tipo de dado do argumento passado nela (typeof)
- `addEventListener` é uma função nativa do JavaScript, o primeiro parâmetro é o evento que ocorre e o segundo o Callback.

  Utilize essa função para mostrar no console o seu nome completo quando o evento 'scroll' ocorrer.
- Crie uma função e utilize as estruturas de controle e repetição que aprendemos para imprimir no console a letra da música Um elefante incomoda, utilize a função para verificar se o número é par neste desenvolvimento.