Funções

Definição de funções

Uma função é um bloco de códigos designado para realizar uma tarefa.

```
function Produto(n1, n2) {
    return n1 * n2;
// Retorna o produto de n1 com n2
}
```

Uma função é executada quando é invocada por alguém.

oProduto(2, 5); // Isso retornará 10

Funções como expressão

Podemos atribuir uma função anônima a uma variável

```
let nomeFuncao = function() {
    console.log("Olá");
};
```

nomeFuncao();

Funções aninhadas

Podemos ter uma função dentro de outra função.

A função aninhada será visível apenas dentro da função pai.

```
function circunferencia (raio)  {
    function diametro() { // função aninhada
      return 2*raio;   // raio é uma variável da função pai
     }

    return Math.PI * diametro(); // invocamos a função
}
```

Callbacks

Callbacks

Callback é uma função passada como parâmetro a outra função, que será executada após o término da primeira.

JavaScript é uma linguagem assíncrona. Isso significa que é possível executar uma invocação sem saber quando ela termina. Para poder administrar essa situação, utilizamos o padrão de design **callback.**

Os callbacks podem ser usados de várias formas. No próximo exemplo, utilizamos um callback com uma função anônima.

```
function a(callback){
        console.log( 'a vem primeiro');
        callback();

}

function b(){
  console.log( 'b vem depois' );
}

a( b );
```

O Resultado seria:

a vem primeiro
b vem depois

Arrays

Métodos de arrays - forEach

Exemplo:

```
let array = [1, 5, 7]

array.forEach(function(value, index) {
  console.log("No índice: " + index + " está o valor: " +
value);
});
```

Resultado:

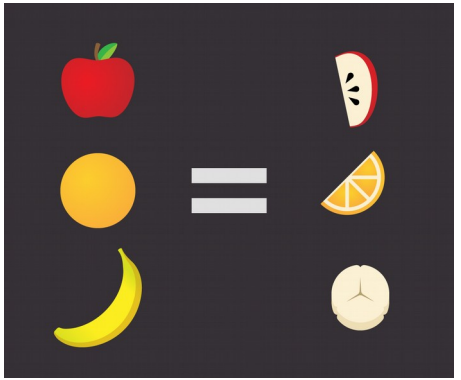
No índice: 0 está o valor: 1
No índice: 1 está o valor: 5
No índice: 2 está o valor: 7

Métodos de arrays - map

O **map** é usado para modificar **cada um** dos itens de um array utilizando uma função determinada.

```
let array = [1, 5, 7]

array.map(function(numero) {
        return numero * 2;
});
```



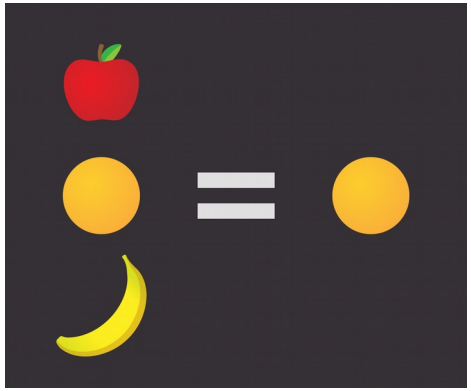
Resultado: [2, 10, 14]

Métodos de arrays - filter

O **filter** é usado para filtrar alguns elementos de um array.

```
let array = [13, 18, 20]

array.filter(function(numero) {
        return (numero >= 18);
});
```



Resultado: [18, 20]

Objeto literal

Definição de objetos

Um **objeto** é qualquer item do dia a dia representado em códigos.

```
let carro = {
        marca: "Chevrolet",
        modelo: "Corsa",
        quilometragem: 65000,
        cor: "Branco",
        donos: ["Luciana", "João", "Zezinho"]
        };
```

Os objetos têm **propriedades** usadas para descrevê-lo.

Elas podem ser de vários tipos (number, string, array, object, etc.)

Propriedades de objetos

É possível acessar uma **propriedade** de um objeto de várias formas:

- carro.marca; // "Chevrolet"
- carro["marca"]; // "Chevrolet"

É possível acessar utilizando variáveis:

```
let propMarca = "marca";
carro[propMarca]; // "Chevrolet"
```

Propriedades de objetos

Estabelecer o valor de uma propriedade:

```
carro.color = "Vermelho";
carro["cor"] = "Vermelho";
```

É possível adicionar propriedades a um objeto:

```
carro.velocidadeMax = 200;
carro[“velocidadeMax”] = 200;
```

Métodos de objetos

Também existem **métodos** para interagir com os objetos.

Um método é uma propriedade do objeto com uma **função** atribuída.

```
let carro = {
    marca: "Chevrolet",
    ligar: function(nome){ console.log("vrum vrum"); },
    modelo: "Corsa"
    };

    carro.ligar();
    // É possível utilizar carro["ligar"]();
```

Métodos de objetos e parâmetros

Como são funções, os métodos também podem receber parâmetros.

```
let carro = {
        marca: "Chevrolet",
        motorista: function(nome){
                console.log(nome+"está dirigindo");
        },
        modelo: "Corsa"
        };


        auto.motorista("Pedro");
```

“Pedro está dirigindo”