

Funções

O que é uma função ?

Uma função é uma “parte” do programa ou código que pode ser reaproveitado ou chamado várias vezes.

Exemplo:

```
function hello(){  
  alert('Olá Mundo');  
}
```

Essa função por si só nunca será executado, para executar uma função é necessário invocá-la. Uma função é invocada da seguinte maneira:

```
hello();
```

Esse tipo de função é conhecida como funções `void`, pois elas não retornam nada. Esse tipo de função é excelente quando se quer apenas reaproveitar código.

Problema

Pedir para o usuário digitar um número e mostrar a tabuada desse número de 1 até 1000. Deve ter a possibilidade do usuário digitar outro número e também de limpar a tabuada gerada previamente.

Resolução:

```
var numero = 1;  
var $areaTabuada = document.querySelector('#tabuada');  
var tabuada = '';  
  
function gerarTabuada(){  
  numero = parseInt(prompt('informe um número'));  
  
  for(var i = 1; i <= 1000; i++){  
    tabuada += numero + ' X ' + i + ' = ' + (numero * i) + '<br>';  
  }  
}
```

```

    $areaTabuada.innerHTML = tabuada;

}

function limparTabuada(){
    tabuada = ''
    $areaTabuada.innerHTML = tabuada;
}

```

Execício

Montar uma mini calculadora. Ao ser carregada ela deve solicitar um número ao usuário, em seguida deve aparecer 4 botões, soma, multiplicação, subtração e divisão, a baixo dos botões o numero digitado. Ao clicar em cada operação a aplicação deve solicitar novamente o número ao usuário e em seguida realizar a operação.

```

var numero = parseFloat(prompt('Digite um número'));
var $mostrador = document.querySelector(".mostrador");

$mostrador.innerHTML = numero;

function somar(){
    var somar = parseFloat(prompt('Digite o número a ser somado'));

    numero = somar + numero;

    $mostrador.innerHTML = numero;
}

function subtracao(){
    var subtracao = parseFloat(prompt('Digite um número para ser subtraído'));

    numero = numero - subtracao;

    $mostrador.innerHTML = numero;
}

function multiplicacao(){
    var multiplica = parseFloat(prompt('Digite um número a ser multiplicado'));

    numero = numero * multiplica;

    $mostrador.innerHTML = numero;
}

function divisao(){
    var divisao = parseFloat(prompt('Digite um número para ser dividido'));

```

```
numero = numero / divisao;

$mostrador.innerHTML = numero;
}
```

Funções que recebem parâmetros

Uma função pode receber valores, conforme exemplo abaixo:

```
function mostrar(x){
    alert(x);
}
```

Esse valores são denominados de parâmetros.

Ao invocar uma função que recebe parâmetros, deve-se informar entre os parenteses, conforme exemplo:

```
monstrar('Hello');
```

O valor passado será interpretado no lugar no parâmetro.

Uma função também pode receber mais de um parâmetro, nesse caso cada parâmetro é separado por virgula, conforme exemplo:

```
function somar(n1, n2){
    var resultado = n1 + n2;
    alert(resultado);
}
```

Exercício 02

Pedir para o usuário digitar dois números que representam os lados de um retângulo assim que a página é carregada. Mostrar na tela a área total quando clicar num botão.

Resolução:

```

var altura = parseFloat(prompt('Informe a altura do retangulo: '));
var largura = parseFloat(prompt('Informe a largura de um retangulo'));

function calcularArea(largura, altura){
    var area = altura * largura;

    var $resultado = document.querySelector('.resultado');

    $resultado.innerHTML = 'A área total do retangulo é: ' + area;
}

```

Exercício 03

Repetir o problema do conteúdo de função, porém dessa vez ao invés de ter várias funções realizando diferentes operações, vamos ter apenas uma única função que recebe um parâmetro e identifica qual será a operação.

```

var numero1 = parseFloat(prompt('Digite um número:'));

var $resultado = document.querySelector('.resultado');

$resultado.innerHTML = numero1;

function calcular(operador){
    var numero2 = parseFloat(prompt('Digite um número:'));

    switch(operador){
        case '+':
            numero1 += numero2;
            break
        case '-':
            numero1 -= numero2;
            break
        case '*':
            numero1 *= numero2;
            break
        case '/':
            numero1 /= numero2;
    }

    $resultado.innerHTML = numero1;
}

```

Funções que retornam valores

As funções que retornam valores são úteis para que o desenvolvedor tenha autonomia sobre os valores retornados. O valor retornado pode ser armazenado em uma variável e reutilizado em outras partes do código.

Exemplo:

```
function multiplicacao(n1, n2){  
  return n1 * n2;  
}
```

Invocando a função e armazenando seu valor:

```
var resultado = multiplicacao(2, 5)
```

A variável resultado irá armazenar o `return` da função.

Exercício 04

Pedir para o usuário digitar dois números que representem os lados de um retângulo. Mostrar na tela a área total de duas formas distintas: em um alert ou em uma div (depende do botão selecionado). Esse código não precisa ser executado quando a página é aberta.

Resolução:

```
function exibirAlert(){  
  var altura = parseFloat(prompt('Digite a altura:'));  
  var largura = parseFloat(prompt('Digite a largura:'));  
  
  var area = calcularArea(largura, altura);  
  
  alert('A área total é: ' + area);  
}  
  
function exibirTela(){  
  var altura = parseFloat(prompt('Digite a altura:'));  
  var largura = parseFloat(prompt('Digite a largura:'));
```

```

    var area = calcularArea(largura, altura);

    document.querySelector('.area').innerHTML = 'A área total é: ' + area;
}

function calcularArea(largura, altura){
    return largura * altura;
}

```

Exercício 05

Repetir o exercício 03 só que agora realizando as funções de forma desacoplada.

Resolução:

```

    var numero1 = parseFloat(prompt('Digite um número:'));

    escrever(numero1);

    function iniciar(operador){
        var numero1 = document.querySelector('.resultado').textContent;
        var numero1 = parseFloat(numero1);
        var numero2 = parseFloat(prompt('Digite outro número:'));

        var mensagem = calcular(operador, numero1, numero2);

        escrever(mensagem);
    }

    function calcular(operador, x, y){

        var resultadoCalculo = null

        switch(operador){
            case '+':
                resultadoCalculo = x + y;
                break
            case '-':
                resultadoCalculo = x - y
                break
            case '*':
                resultadoCalculo = x * y;
                break
            case '/':
                resultadoCalculo = x / y;
        }

        return resultadoCalculo;
    }

```

```
function escrever(x){  
    document.querySelector('.resultado').innerHTML = x;  
}
```

Avançando mais um pouco....

Uma função pode ser literal...isso se diz respeito a forma como ela é declarada, por exemplo:

```
function exemplo(parametro){  
    //linhas de código  
    //Corpo da função  
}
```

aqui é utilizado a palavra reservada `function`, que defini que será uma função. O nome da função. E entre parenteses, alguns parâmetros no qual pode ser trabalhando dentro da função. Dentro da função temos as linhas de código, que é conhecido como corpo da função.

OBS: Uma função pode ter seu nome ocultado, criando assim uma função anonima.

```
function (parametro){  
    //Código.  
}
```

Uma função também é um elemento de primeira classe. O que significa que eu posso tratar uma função como se fosse um objeto. Sendo assim é possível definir uma propriedade a uma função da seguinte forma:

```
function exemplo (){  
    //linhas de código.  
}  
  
exemplo.propriedade = "Demonstração";
```

Um função também pode receber outra função como parâmetro, exemplo:

```
function exemplo (parametro){  
    parametro();  
}  
  
function hello(){  
    alert('Olá, mundo.');}  
  
exemplo(hello());
```

O código acima é totalmente válido.

Escopo de uma função:

Quando iniciamos um código e declaramos uma variável ela é declarada de forma global ou seja, todo o código tem acesso a essa variável.

Agora quando uma variável é declarada dentro de uma função ela existe apenas lá dentro, o que chamamos de forma local.

```
var x = 0;  
  
function exemplo(){  
    var x = 10;  
}  
  
console.log(x);
```

No exemplo acima o valor printado no console será 0, pois o x declarado dentro da função só existe lá. Caso de forma global eu tenha uma variável com o mesmo nome, essas variáveis não entram em conflito.

Existe um problema, caso queira definir uma variável apenas de forma local em uma função e esqueça de colocar a palavra var, essa variável irá vazar de forma global, exemplo:

```
var x = 0;  
  
function exemplo(){  
    y = 10;  
}
```



```
console.log(y);
```

Conforme o exemplo a cima o Y será exibido no console, pois não foi declarado a palavra var.

Um outro exemplo:

```
var x = 10;
var y = 20;

function exemplo(){
  var x = 30;
  var y = 40;

  console.log('X local: ' + x);
  console.log('Y local: ' + y);
}

exemplo();

console.log('X global: ' + x);
console.log('Y global: ' + y);
```

Resultado:

X local: 30

Y local: 40

X global: 10

Y global: 20

Funções auto executáveis

As funções auto executáveis tem um o objetivo que contorna um problema, o problema da poluição no escopo global. Imagine que dois arquivos diferentes compartilhe a mesma variável no escopo global, a chance de acontece um bug é imensa.

Para isso utilizamos uma função para envolver todo o nosso código, dessa forma quando o código for executado todas as variáveis serão limpas.

Exemplo:

```
iniciar();
```

```
function iniciar(){  
    //Código aqui  
}
```

O problema nessa implementação é que outros códigos também podem ter uma função chamada iniciar, para contornar esse problema utilizamos a seguinte solução:

```
//IIFE ou Função auto executável  
(function(){  
    //Código aqui  
})();
```

Uma outra boa pratica ao se utilizar IIFE (Immediately-invoked function expression) é utilizar uma string que é a: 'use strict'. Ela serve para impedir que uma variável seja declarada sem a palavra var ou let.

```
(function(){  
    'use strict';  
    //código aqui  
})
```

Propriedade arguments

A propriedade arguments é uma propriedade que nos permite visualizar os parâmetros passados em uma espécie de array. Isso nos traz uma maior flexibilidade a função, exemplo:

```
somar(10, 20, 10, 10)  
  
function somar(){  
    var resultado = 0;  
    for(var i = 0; i < argument.length; i++){  
        resultado += argument[i];  
    }  
    return resultado  
}
```

No exemplo acima a função não esperava receber nem um parâmetro de forma explícita, mas recebeu e foi tratado pela propriedade argument.

Exercício 06

Criar uma função que retorne a média de números passados por parâmetros. Essa função pode receber quantos números forem necessários.

exemplos:

calcularMedia(2, 4);

calcularMedia(1, 1, 2, 3, 5);

Resolução:

```
(function(){
  'use strict';

  console.log(calcularMedia(2, 4));
  console.log(calcularMedia(10, 10, 20, 0, 50));

  function calcularMedia(){
    var soma = 0;

    for(var i = 0; i < arguments.length; i++){
      soma += arguments[i];
    }

    return soma /= arguments.length;
  }
})();
```

Exercício 07

Crie uma função que retorne o índice de massa corporal. Crie uma outra função que receba esse índice e retorne uma string informando sua classificação.

$$\text{IMC} = \text{peso} / \text{altura}^2$$

Classificação	IMC
Muito abaixo do peso	16 a 16,9 kg/m2
Abaixo do peso	17 a 18,4 kg/m2
Peso normal	18,5 a 24,9 kg/m2
Acima do peso	25 a 29,9 kg/m2
Obesidade Grau I	30 a 34,9 kg/m2
Obesidade Grau II	35 a 40 kg/m2
Obesidade Grau III	maior que 40 kg/m2

Fonte: <https://www.tuasaude.com/imc/>

Resolução:

```
(function(){
    'use strict';

    var peso = parseFloat(prompt('Informe seu peso:'));
    var altura = parseFloat(prompt('Informe sua altura'));

    var massa = calcularMassaCorporal(peso, altura);

    classificacaoImc(massa);

    function calcularMassaCorporal(peso, altura){
        var calculo = peso / (altura ** 2);
        return calculo;
    }

    function classificacaoImc(indice){
        if(indice >= 16 && indice < 17){
            console.log('Muito abaixo do peso');
        }else if(indice >= 17 && indice < 18.5){
            console.log('Abaixo do peso');
        }else if(indice >= 18.5 && indice < 30){
            console.log('Peso normal');
        }else if(indice >= 25 && indice < 30){
            console.log('Acima do peso');
        }else if(indice >= 30 && indice < 35){
            console.log('Obesidade de primeiro grau');
        }else if(indice >= 35 && indice < 41){
            console.log('Obesidade de segundo grau');
        }else if(indice > 40){
            console.log('Obesidade de terceiro grau');
        }
    }
})
```

}()