

Fundamentos do Desenvolvimento de Software

Programação Web com JavaScript I

Agenda

Etapa 7: Funções em Javascript.

- Funções.
- Bibliotecas.
- Praticando Códigos.



The background is an abstract geometric pattern composed of numerous triangles of varying sizes and shades of green and blue. The colors transition from a dark, almost blackish-blue on the left to a vibrant green on the right, with various intermediate shades of teal and forest green. The triangles are arranged in a way that creates a sense of depth and movement.

Funções

Funções são blocos nomeados de código projetados para fazer um trabalho específico: **devem ter somente uma finalidade**.

Quando você deseja realizar uma tarefa específica que definiu em uma função, você chama a função responsável por ela.

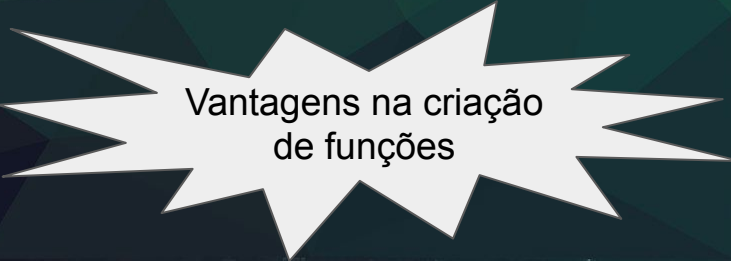
Se precisar executar essa tarefa várias vezes em todo o programa, não precisará digitar todo o código para a mesma tarefa repetidamente: você apenas chama a função dedicada a lidar com essa tarefa.

Você descobrirá que o uso de funções torna seus programas mais fáceis de escrever, ler, testar e corrigir.

Modularização: facilita o desenvolvimento, pois implementa-se uma unidade menor, concentrando-se nela, até que ele esteja funcionando com alto grau de confiabilidade.

Organização: o código fica melhor organizado e portanto mais fácil de ser entendido e mantido.

Reaproveitamento: sempre que precisar aplicar o código encapsulado em qualquer outra parte do programa, pode-se utilizar aquele que já foi implementado.



Vantagens na criação
de funções

script.js x +



> ...

```
1 //Definição da função
2 ▼ function olaMundo() {
3
4     alert("Olá Mundo!");
5 }
6
7 //Chamada da função
8 olaMundo();
```


Modificada ligeiramente, a função **olaMundo** não vai apenas dizer ao usuário Olá! mas também cumprimentá-los pelo nome.

Para que a função faça isso, você insere o nome de usuário entre parênteses da definição da função.

Ao adicionar nome de usuário, você permite que a função aceite qualquer valor de nome de usuário que especificar.

Os valores que são fornecidos às funções são chamados de parâmetros.

script.js x +



> f olaMundo

1 //Definição da função

2 ▼ function olaMundo(pessoa) {

3

4 💡 alert("Olá Mundo, " + pessoa + "!");

5 }

6

7 //Chamada da função

8 olaMundo("Machado de Assis");

script.js x +



> ...

1 //Definição da função

2 **function** **descreverPet**(tipo_pet, nome_pet) {

3

4 **alert**("Eu tenho um " + tipo_pet + "\nE o nome dele é " + nome_pet);

5 }

6

7 //Chamada da função

8 **descreverPet**("Hamster", "Willie");

Programa



prompt

```
script.js x +  
> ...  
1 ▾ /*  
2   Crie um programa que receba na entrada o valor de três notas de  
3   um aluno - com valor entre 0 e 10 e informe a média entre elas.  
4  */  
5  let nota1 = Number(prompt("Digite a primeira nota"));  
6  let nota2 = Number(prompt("Digite a segunda nota"));  
7  let nota3 = Number(prompt("Digite a terceira nota"));  
8  let media = (nota1 + nota2 + nota3) / 3;  
9  alert(media);
```

alert

Função



parâmetros

```
script.js x +
> ...
1 //Definição da função
2 function somar(numero1, numero2) {
3
4   return numero1 + numero2;
5 }
6
7 //Chamada da função
8 alert(somar(10, 20));
```

Quando for criar uma função, pense em algo mais complexo que pode ser “agrupado” em um módulo.

Não crie funções para operações ou expressões simples.

return

The background is an abstract geometric pattern composed of numerous triangles of varying sizes and shades of green and blue. The colors transition from a dark, almost blackish-blue on the left to a vibrant green on the right, with various intermediate shades of teal and forest green in between. The triangles are arranged in a way that creates a sense of depth and movement.

Bibliotecas

As **Bibliotecas de Funções** são coleções de subprogramas às quais um programador pode recorrer quando escreve um programa de computador.

A principal vantagem resulta da possibilidade de utilização compartilhada, consequência direta da sua estrutura modular.

Para iniciarmos com esse conceito, vamos criar uma biblioteca simples com várias funções de validação básicas.



Search

1



Files

index.html

script.js

style.css

validacoes.js

2

validacoes.js x +



1

validacoes.js x +

3

```
1 //-----
2 //Biblioteca com funções de validação de campos
3 //-----
4 function isEmpty(field) {
5   let valid = false;
6   if(field != null && field != undefined && field.length > 0) {
7     valid = true;
8   }
9   return valid;
10 }
```

A propriedade **length** de uma String contém o comprimento da string.

script.js x +

> ...

```
1 let nome; // = prompt("Digite o seu nome");
2 isEmpty(nome) ? alert("OK") : alert("NOK");
```

4

Praticando Códigos

Inclua na sua biblioteca duas novas funções: uma para validar se o campo tem um mínimo de caracteres e outra para validar se o campo tem um máximo de caracteres.



Inclua na sua biblioteca uma função para validar se é um número inteiro e outra para validar se é um número real.

