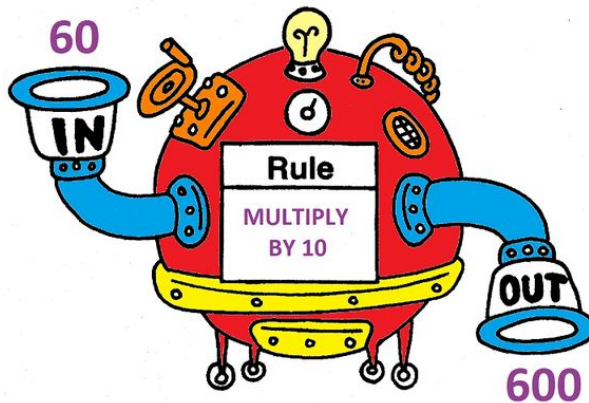


Desenvolvimento de Sistemas : Funções

Prof. Ovídio Francisco



Plano de aula

Introdução

Funções em Javascript

Parâmetros

Retorno

Exercícios

O contexto

Imagine um programa para uso estatístico.

Para criar esse programa o desenvolvedor terá que encontrar a **média aritmética** das **alturas** de um grupo de pessoas. Então ele cria um código que cumpre essa tarefa.

Depois ele vê a necessidade de calcular a média aritmética dos **pesos** das pessoas. Então ele cria um código que cumpre essa tarefa.

Ele ainda encontra a necessidade de calcular a média aritmética dos **salários**. E novamente cria um código que cumpre essa tarefa e percebe que está se repetindo.

O que fazer?

O contexto

Imagine um programa para uso estatístico.

Para criar esse programa o desenvolvedor terá que encontrar a **média aritmética** das **alturas** de um grupo de pessoas. Então ele cria um código que cumpre essa tarefa.

Depois ele vê a necessidade de calcular a média aritmética dos **pesos** das pessoas. Então ele cria um código que cumpre essa tarefa.

Ele ainda encontra a necessidade de calcular a média aritmética dos **salários**. E novamente cria um código que cumpre essa tarefa e percebe que está se repetindo.

O que fazer?



O contexto

Ele encontra várias situações onde precisa calcular a médias e aproveita seu código **copiando** as partes que já havia escrito. Assim, seus cálculos estatísticos estão todos prontos. Bastou copiar e colar.

Certo?

Mas algum tempo depois, descobre que deveria ter calculado as **médias ponderadas** ao invés da média aritmética.

O que fazer?

[] - Encontrar e alterar todos os códigos que calculam a média.

[] - Criar um comando que calcula a média e usá-lo desde o começo.

Mas por quê não?

Pontos negativos em copiar e colar códigos:

- Trabalho de copiar e colar
- Dificuldade em manutenção
- Dificuldade na legibilidade
- Dificuldade em ver o todo
- O software fica maior

O contexto

Outro exemplo:

Imagine uma escola que precisa cadastrar alunos, responsáveis e professores.

Um sistema tem módulos (tela) para cadastro de cada uma dessas entidades onde sempre deve-se verificar se o CPF, caso informado, é válido.

Imagine ainda que esses cadastros podem ser feitos por várias interfaces como web, desktop, mobile ou ainda carregadas de um arquivo ou banco de dados.

Nesse caso, o programador pode criar uma função e reutilizá-la sempre que precisar.

Pontos positivos em usar funções:

- Reaproveitamento de código
- Concentração da lógica
- Torna o código mais **organizado, sucinto e legível**.
- **Evita retrabalho.**
- **Facilita manutenção.**
- O software fica menor

Funções em Javascript

Funções em C

Você certamente já usou funções. Exemplos:

```
prompt("Informe ...");
```

Solicita dados

```
alert("Olá mundo!");
```

Exibe uma mensagem

```
Number("234");
```

Converte para número

Funções em C

- Funções são usadas para **modularizar** um programa.
- Ao criar uma função, cria-se um novo **comando**.
- Torna o código mais **organizado, sucinto e legível**.
- **Evita retrabalho**.
- **Facilita manutenção**.

Funções em C

Imagine algo bem simples:

Você precisa de uma função que escreve “Olá” na tela.

Em Javascript, o código completo pode ser algo assim:

```
<script>

    function digaOlá() {
        |     document.write("Olá!<br>")
    }

    digaOlá();

</script>
```

Para utilizar funções é preciso:

1. Definir (criar) a função
2. Fazer chamadas (executar a função)

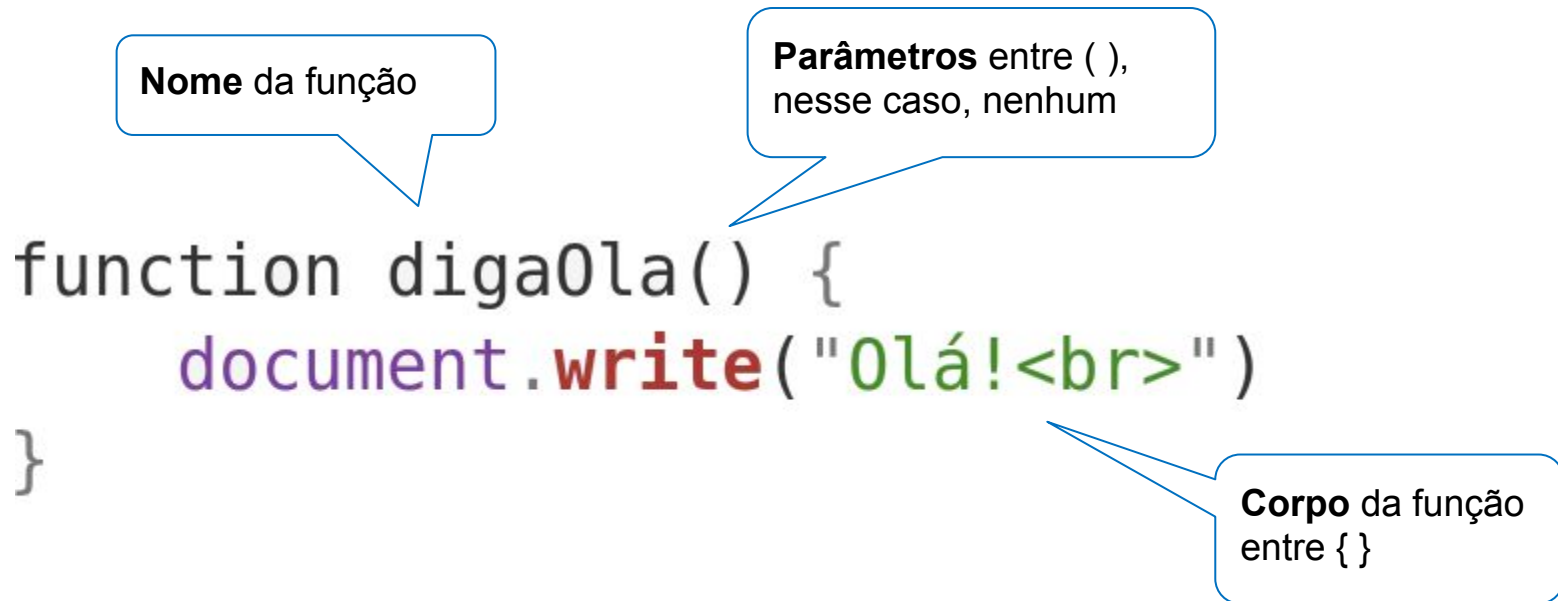
Funções em C

Antes de utilizar uma função, é preciso definí-la, como mostrado abaixo.

```
function digaOlá() {  
    document.write("Olá!<br>")  
}
```

Funções em C

Um função deve ter um **nome**, lista de **parâmetros** (*veremos mais detalhes adiante*) e **corpo** (código).



The diagram illustrates the syntax of a function in C. It features a code snippet with three callout boxes explaining its components:

- Nome da função**: Points to the function name `digaOla()`.
- Parâmetros entre (), nesse caso, nenhum**: Points to the empty parentheses `()` following the function name.
- Corpo da função entre { }**: Points to the curly braces `{ }` that enclose the function body.

```
function digaOla() {  
    document.write("Olá!<br>")  
}
```

Funções em C

Uma vez criada, você chama (executa) a função **como** um **comando**. Veja o exemplo:

```
<script>
```

```
function digaOla() {  
    document.write("Olá!<br>")  
}
```

Declaração
da função

```
digaOla();
```

Chamada para a
função

```
</script>
```

Chamar uma função, significa **executar** o código da função no **ponto onde é chamada**.

Funções em C

Atenção: Definir uma função não faz com que o seu código seja executado. Será executado somente quando houver uma chamada à função.

```
<script>
```

```
function digaOlá() {  
    document.write("Olá!<br>")  
}
```

```
digaOlá();
```

```
</script>
```


Não executa ainda,
apenas define a função

A execução de função segue a seguinte ordem:

1. Chamada à função
2. Execução do código da função
3. Retorno ao ponto após a chamada

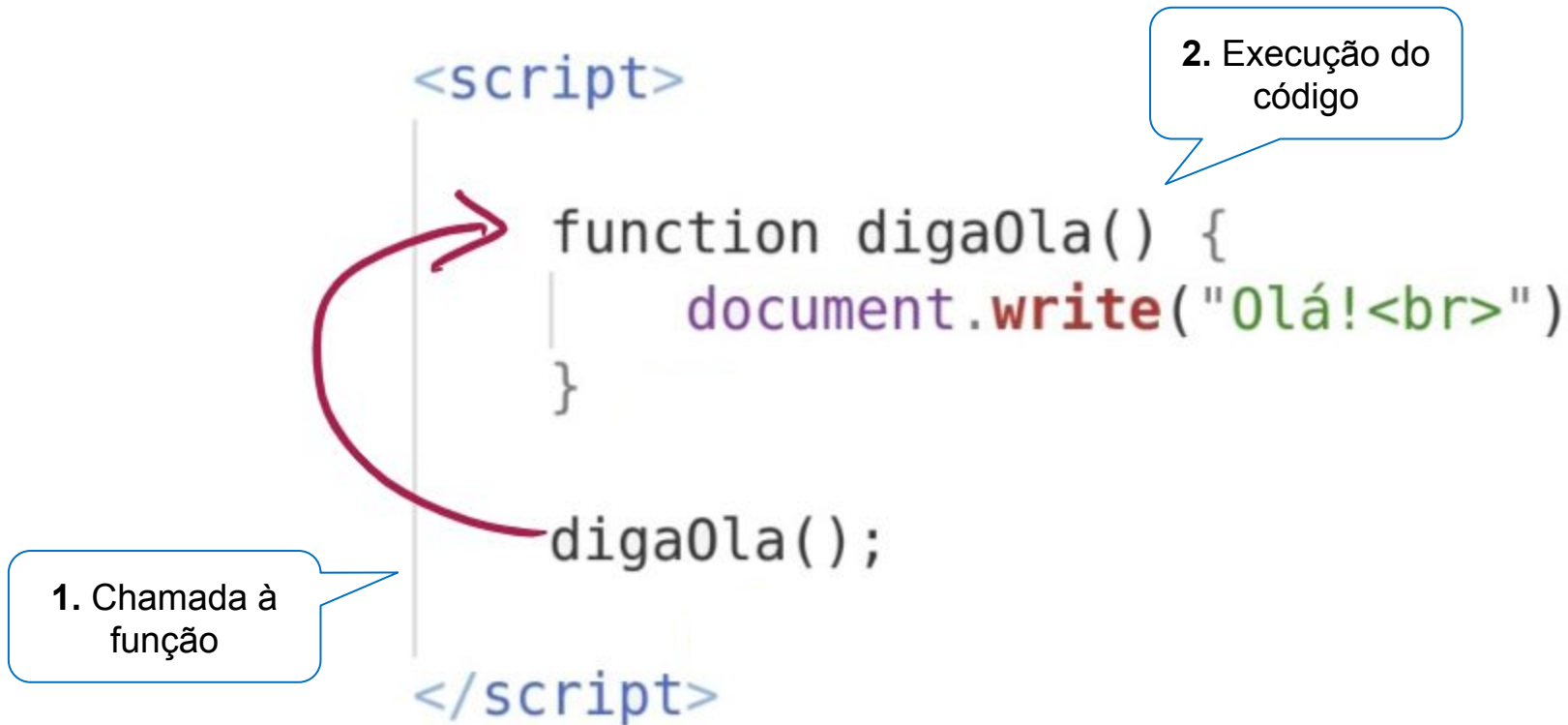
Funções em C

```
<script>  
    function digaOla() {  
        document.write("Olá!<br>")  
    }  
    digaOla();  
</script>
```

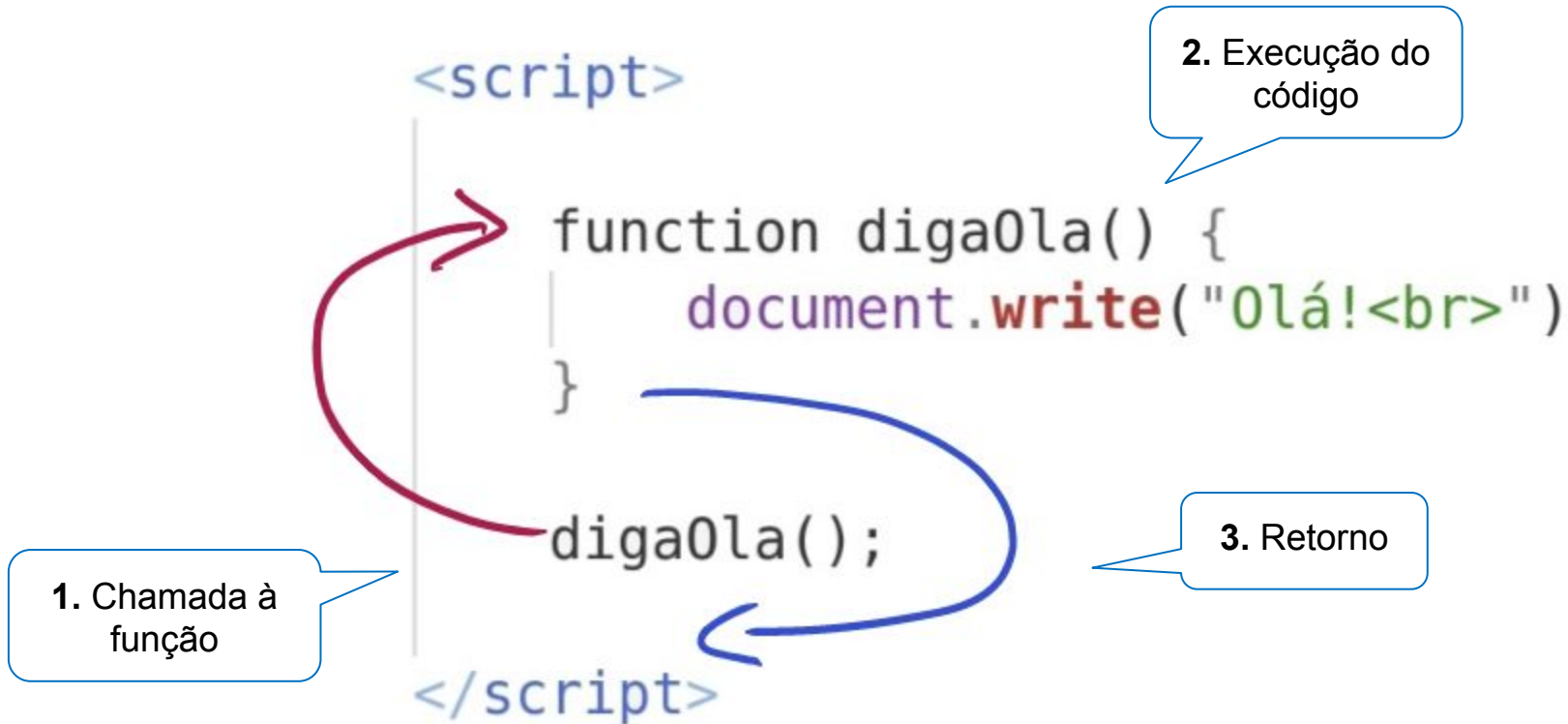


1. Chamada à
função

Funções em C



Funções em C



Parâmetros

Parâmetros

Agora, caso você queira que sua função não seja limitada a só um texto fixo. Você pode usar **parâmetros**.

Veja o exemplo:

```
<script>
```

```
function digaOla(nome) {  
    document.write("Olá " + nome + "!<br>")  
}
```

```
digaOla("Sebastian");
```

```
</script>
```

Parâmetros

- Parâmetros são **dados** enviados para a função.
- O código da função irá utilizar esses **dados**.
- Os parâmetros são **definidos na implementação** da função.
- Tornam-se variáveis dentro da função
- A chamada da função deve passar os **valores requeridos entre parênteses**.

Parâmetros

```
<script>
```

Parâmetro do tipo
string

```
function digaOla(nome) {  
    document.write("Olá " + nome + "!<br>")  
}
```

```
digaOla("Sebastian");
```

```
</script>
```

Esse texto é passado
para a função

Parâmetros

```
<script>
```

```
function digaOla(nome) {  
    document.write('Olá ' + nome + '!<br>')  
}
```

Parâmetro do tipo
string

```
digaOla("Sebastian");
```

```
</script>
```

Esse texto é passado
para a função

Parâmetros

Veja outro exemplo de função que recebe dois inteiros e escreve a soma:

```
<script>
```

```
function escreveSoma(a, b) {  
    var soma = a + b;  
    document.write("<p>A soma é: " + soma + "</p>");  
}
```

```
escreveSoma(5, 3);
```


```
</script>
```

Parâmetros

Veja outro exemplo de função que recebe dois inteiros e escreve a soma:

```
<script>
```

```
function escreveSoma(a, b) {  
    var soma = a  
    document.writ>A soma é: " + soma + "</p>");  
}
```



```
escreveSoma(5, 3);
```

```
</script>
```

Exercícios

1. Escreva uma função que mostra se um número é positivo ou negativo. Considere o zero positivo.
2. Escreva uma função que informa se um número é ímpar. Use o operador % (módulo).
3. Escreva uma função que recebe dois números e informa o produto.
4. Escreva uma função que recebe três números e informa o maior.
5. Escreva uma função recebe 4 números e informa a média aritmética;

Retorno

Retorno

Imagine agora que você precisa somar dois números. Na verdade, você precisará somar dois números muitas vezes.

Veja o exemplo:

```
<script>

    a = 3;
    b = 5;
    c = a + b;
    d = a + c;
    c = b + d;
    d = d + c;

    document.write("d = " + d + "<br>");

</script>
```

Introdução

Para deixar seu código mais organizado, você pode criar uma função que **retorna** a soma de duas variáveis;

Veja como fica:

```
function soma(a, b) {  
    var s = a + b;  
    return s;  
}
```



Retorno do resultado

Introdução

Veja o exemplo:

```
<script>
```

```
    function soma(a, b) {  
        var s = a + b;  
        return s;  
    }
```

```
    a = 3;
```

```
    b = 5;
```

```
    c = soma(a, b);
```

```
    d = soma(a, c);
```

```
    c = soma(b, d);
```

```
    d = soma(d, c);
```

```
    document.write("d = " + d + "<br>");
```

```
</script>
```


Parâmetros

- Você pode entender o **retorno** como o **resultado** da função ou a **saída de dados da função**.
- Uma função pode **retornar um valor para o ponto onde foi chamada**.
- Esse valor pode ser atribuído a uma variável, usado para compor uma expressão ou como parâmetro para outra função.
- A palavra-chave **return encerra** a execução da função e **retorna o valor**.

Introdução

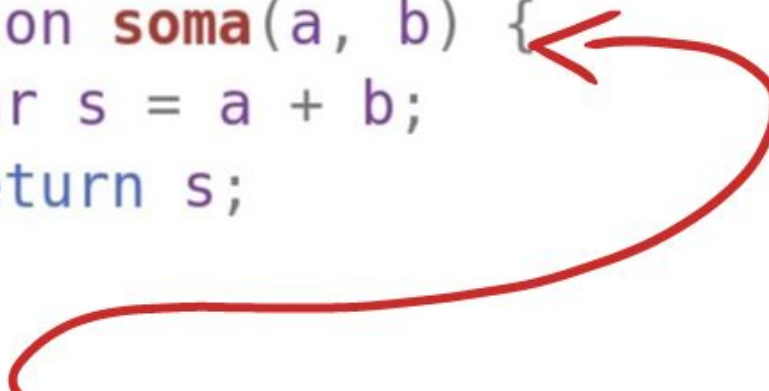
Veja o exemplo:

```
<script>
```

```
function soma(a, b) {  
    var s = a + b;  
    return s;  
}
```

```
c = soma(3, 6);
```

```
</script>
```



1. Chamada à
função

Introdução

Veja o exemplo:

```
<script>
```

2. Execução do código

```
function soma(a, b) {  
    var s = a + b;  
    return s;  
}
```

```
c = soma(3, 6);
```

```
</script>
```

1. Chamada à função

Introdução

Veja o exemplo:

```
<script>
```

2. Execução do código

```
function soma(a, b) {  
    var s = a + b;  
    return s;  
}
```

3. Retorno

```
c = soma(3, 6);
```

```
</script>
```

1. Chamada à função

Para saber mais...

- http://www.inf.ufpr.br/cursos/ci067/Docs/NotasAula/notas-11_Fun_c_coes.html
- https://www.tutorialspoint.com/cprogramming/c_functions.htm
- <https://www.geeksforgeeks.org/functions-in-c/>
- <https://www.programiz.com/c-programming/c-functions>
- <https://blog.cranksoftware.com/challenge-of-code-reuse-and-embedded-guis>
-

Exercícios

Exercícios

1. Escreva uma função que mostra se um número é positivo ou negativo. Considere o zero positivo.
2. Escreva uma função que recebe 3 números e retorna o produto.
3. Escreva uma função que recebe 4 números e informa a média aritmética;
4. Escreva uma função que retorna se um número é ímpar. Use o operador % (módulo).
*Opcional: retorne **true** ou **false**.*
5. Escreva uma função que recebe três números e retorna o maior.
6. Escreva uma função que recebe dois números, a e b, e retorna a^b .
7. Escreva uma função que retorna a tabuada de um número recebido por parâmetro.