SIMULADOS & TUTORIAIS

TUTORIAIS JAVASCRIPT

Como utilizar funções em JavaScript

TUTORIAIS JAVASCRIPT

Como utilizar funções em JavaScript

Introdução

Uma função é um bloco de código que será executado quando a mesma for chamada por algum comando. Por exemplo:

```
<!DOCTYPE html>
<html>
<head>
<script>
function minhaFuncao()
{
alert("Olá Pessoal!");
}
</script>
</head>
<body>
<button onclick="minhaFuncao()">Clique aqui</button>
</body>
</html>
```

Esse exemplo, após executado, surgirá um botão, que, ao ser clicado chamará a função **minhaFunção()** e executará o código inserido no seu interior.

A sintaxe de uma função em JavaScript

Uma função é escrita como um bloco de código dentro de duas chaves ({}), precedida pela palavra chave **function**. Sua sintaxe é a seguinte:

```
function nome_da_função()
{
código a ser executado
}
```

Uma função pode ser chamada diretamente quando um evento ocorre, tal como o clique de um botão, por exemplo, ou pode ser chamada de qualquer lugar por um codigo JavaScript.

Nota: A palavra chave **function** deve ser escrita com letras minúsculas, e deve ser chamada da mesma forma que foi escrita, caso contrário, causará um erro.

Chamando uma função com argumentos

Quando você chamar uma função, você também pode passar alguns valores para ela. Esses valores são chamados de argumentos ou parâmetros. Esses argumentos podem ser usados dentro da função. Você pode enviar quantos argumentos forem necessários, separando-os por vírgulas, da seguinte forma:

```
function minhaFuncao (argumento1, argumento2)
{
```

```
código a ser executado.
}
```

Os argumentos podem ser variáveis, expressões, etc., depende da finalidade da função. Esses argumentos ou parâmetros devem obedecer a mesma ordem em que foram criados, caso contrário o resultado poderá não ser o esperado.

Vejamos um exemplo prático:

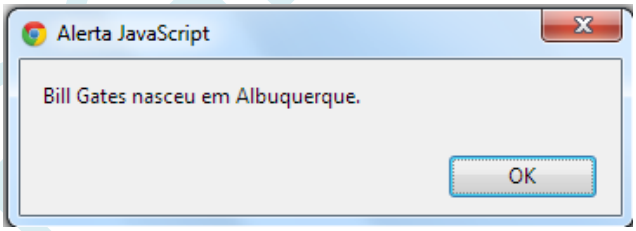
Exemplo 1

Nesse exemplo vamos criar uma função com dois argumentos, que após executada mostrará uma caixa de diálogo com a mensagem descrita no corpo da função utilizando esses argumentos. Para isso:

1. Digite o código abaixo no seu editor de texto e salve-o como: **js6-exemplo1.html**.

```
<!DOCTYPE html>
<html>
<body>
<p>Clique no botão para chamar a função com argumentos.</p>
<button onclick="minhaFuncao('Bill Gates','Albuquerque')">Clique aqui</button>
<script>
function minhaFuncao(nome, cidade)
{
alert(nome + " nasceu em " + cidade + ".");
}
</script>
</body>
</html>
```

2. Esse código após executado no browser e o botão for clicado exibirá a seguinte janela de diálogo:



Você poderá usar quaisquer argumentos para essa função, desde que seja compatível com a mensagem que será exibida.

Funções com valor de retorno

Às vezes você deseja que sua função retorne um valor qualquer e não simplesmente exibir uma mensagem, de forma que você possa utilizar esse valor para uma determinada finalidade, fazer um cálculo, por exemplo. Isso é possível utilizando-se uma declaração chamada **return**. Vejamos um exemplo:

```
function minhaFuncao()
{
var x = 5;
return x;
}
```

Essa função após executada retornará o valor 5. E com esse valor retornado você poderá usá-lo em qualquer situação e várias vezes quantas forem necessárias em toda a sua página.

Você também poderá armazenar essa função em uma varável, por exemplo:

```
var valor = minhaFuncao();
```

Na verdade você está armazenando o valor 5, que é o resultado da função.

Funções com valor de retorno e com argumentos

Você também poderá utilizar esse tipo de função com argumentos. Vejamos um exemplo:

```
function soma(x, y)
{
var soma = x + y;
return soma;
}
```

Vejamos um exemplo prático:

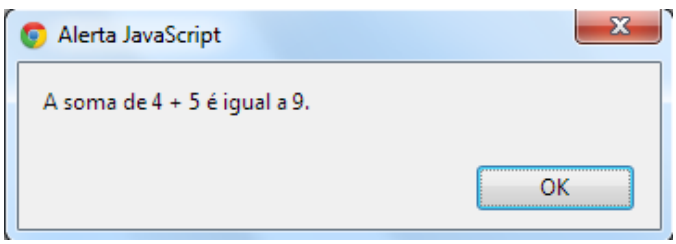
Exemplo 2

Nesse exemplo vamos criar uma função chamada **soma** com duas variáveis. Quando ela for executada retornará a soma dessas variáveis e você poderá fazer qualquer um outro cálculo com esse resultado. Para isso:

1. Digite o código abaixo no seu editor de texto e salve-o como: **js6-exemplo2.html**.

```
<!DOCTYPE html>
<html>
<body>
<p>Clique no botão para chamar a função com argumentos.</p>
<button onclick="soma(4, 5)">Clique aqui</button>
<script>
function soma(x, y)
{
var soma = x + y;
alert("A soma de " + x + " + " + y + " é igual a " + soma + ".")
return soma;
}
</script>
</body>
</html>
```

2. Esse código após executado e o botão for clicado exibirá a seguinte janela de diálogo:



E você ainda poderá usar o valor de retorno (9) para outro propósito, como por exemplo, extrair a raiz quadrada, fazer algum cálculo com o valor de outra função, entre outras coisas.

A declaração **return** não serve somente para retornar valores de funções, serve também para sair do controle de uma função caso uma condição não seja satisfeita, mas não é obrigatório. Veja um exemplo:

```
function minhaFuncao(x, y)
{
if (x > y)
  {
  return;
  }
x = x + y;
}
```

Nesse caso o resto da função será ignorada caso **x** seja maior do que **y**.

Vejamos um exemplo prático:

Exemplo 3

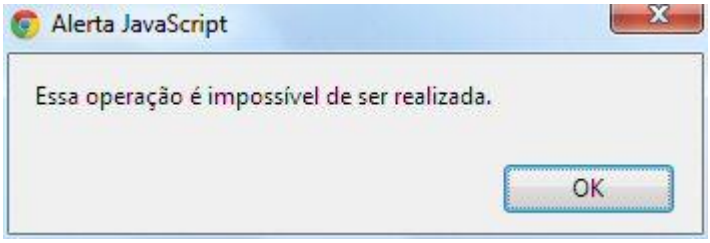
Nesse exemplo vamos criar uma função chamada **dividir** com duas variáveis (**x** e **y**). Nesse caso se **y** for igual a zero – como não é possível essa operação – a função será encerrada, caso contrário o valor da divisão será retornado. Para isso:

1. Digite o código abaixo no seu editor de texto e salve-o como: **js6-exemplo3.html**.
2. Execute no seu browser preferido e confira o resultado.

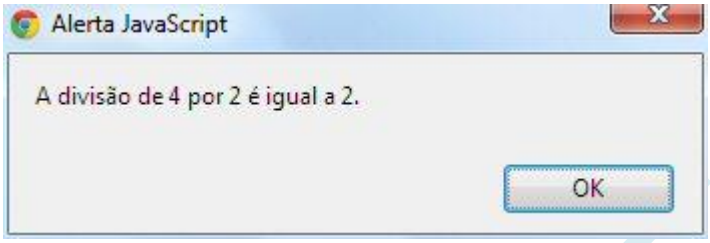
```
<!DOCTYPE html>
<html>
<body>
<p>Clique no botão para chamar a função.</p>
<button onclick="dividir(4, 0)">Clique aqui</button>
<script>
function dividir(x, y)
{
if (y == 0) {
alert("Essa operação é impossível de ser realizada.")
}
else
{
var resultado = x / y;
alert("A divisão de " + x + " por " + y + " é igual a " + resultado + ".");
}
return resultado;
}
</script>
</body>
</html>
```

Esse código após executado e o botão for clicado exibirá as seguintes janelas de diálogo:

Para o caso de **y** ser igual a 0:



Para o caso de **y** for diferente de 0:



Variáveis locais em JavaScript

Uma variável declarada dentro de uma função JavaScript usando a palavra chave **var** é considerada **LOCAL**, e só pode ser acessada dentro dessa função. Você pode ter variáveis locais com o mesmo nome em diferentes funções, porque variáveis locais são somente reconhecidas pela função na qual elas foram declaradas. Variáveis locais são descartadas assim que a função é completada. Vejamos um exemplo:

```
function somar(x, y)
{
var resultado = x + y;
return resultado;
}
function dividir(x, y)
{
var resultado = x / y;
return resultado;
}
```

Variáveis globais em JavaScript

Variáveis declaradas fora de uma função são consideradas **GLOBAIS**, e todos os scripts e funções da página poderão acessá-las. Por exemplo:

```
var raiz = 9;
function calcular(x)
{
if(raiz <= 5)
{
var resultado = x + raiz;
}
else
{
var resultado = x / raiz;
}
alert("O resultado da operação é: " + resultado + ".");
}
```

Vejamos um exemplo prático:

Exemplo 4

Nesse exemplo vamos criar uma função chamada **calcular** com duas variáveis (uma local, **x**, e uma global, **raiz**). Nesse caso se **x** for menor ou igual a **5** o resultado da operação será uma **soma**, caso contrário será uma **divisão**. Note que foi utilizada uma variável global que está sendo usada dentro da função, e apenas uma caixa de diálogo será exibida para qualquer um dos resultados. Para isso:

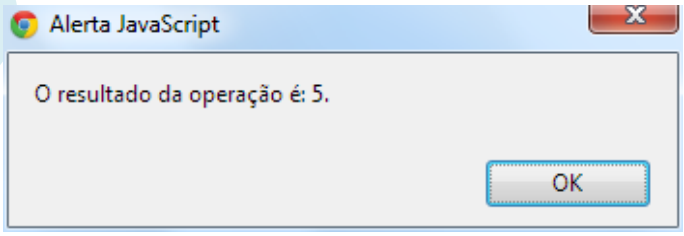
1. Digite o código abaixo no seu editor de texto e salve-o como: **js6-exemplo4.html**.

```
<!DOCTYPE html>
<html>
<body>
<p>Clique no botão para chamar a função.</p>
<button onclick="calcular(5)">Clique aqui</button>
<script>
var raiz = 0;
function calcular(var x)
{
if(raiz <= 20)
{
var resultado = x + raiz;
}
else
{
var resultado = x / raiz;
}
alert("O resultado da operação é: " + resultado + ".");
}
</script>
</body>
</html>
```

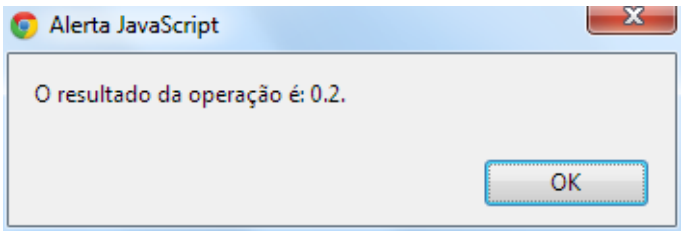
2. Esse código após executado e o botão for clicado exibirá apenas uma janela de diálogo com um dos seguintes resultados:

Considerando x = 5:

Para o caso de **raiz** ser menor ou igual a **20** (por exemplo: **0**):



Para o caso de **raiz** ser maior do que **20** (por exemplo: **25**):



Tempo de vida das variáveis em JavaScript

O tempo de vida das variáveis em JavaScript começa quando elas são declaradas. Variáveis locais são descartadas quando a função é completada. Variáveis globais são descartadas quando uma página é fechada.

Atribuindo valores a variáveis não declaradas em JavaScript

Se você atribuir um valor a uma variável que não tenha ainda sido declarada, essa variável será automaticamente declarada como variável **GLOBAL**. Por exemplo:

A declaração:

```
nomeCarro = "Fiat";
```

Será considerada uma variável global, mesmo se for executada dentro de uma função.

*****

Exercícios de fixação

1) Uma variável declarada dentro de uma função JavaScript usando a palavra chave **var** é considerada **LOCAL**, e só pode ser acessada dentro dessa função. Essa declaração é:

   a) Verdadeira
   b) Falsa

2) Que palavra-chave devemos utilizar para criarmos uma função?

   a) myFunction
   b) var
   c) function
   d) varFunction

3) Variáveis globais são aquelas declaradas fora de uma função, e todos os scripts e funções da página poderão acessá-las. Essa declaração é:

   a) Verdadeira
   b) Falsa

4) Variáveis locais são descartadas quando a função é iniciada. Variáveis globais são descartadas quando uma página é fechada. Essa declaração é:

   a) Verdadeira
   b) Falsa

5) O tempo de vida das variáveis em JavaScript começa quando elas são declaradas. Essa declaração é:

   a) Verdadeira
   b) Falsa

6) Que declaração devemos utilizar para que uma função retorne um valor?

   a) value
   b) return
   c) back
   d) returnValue

7) Que valor a função abaixo retornará após ser executada. Considerando x=3 e y=4?

```
function minhaFuncao(x, y)
{
if (x > y)
  {
  return x;
  }
else {
  return y;
  }
}
```

   a) 2
   b) 3
   c) 4
   d) 5

8) Uma função deve ter obrigatoriamente pelo menos um parâmetro ou argumento. Essa declaração é:

   a) Verdadeira
   b) Falsa

9) Uma função em JavaScript é escrita como um bloco de código dentro de:

   a) colchetes
   b) parênteses
   c) chaves
   d) aspas

10) Os argumentos de uma função só podem ter valores numéricos. Essa declaração é:

   a) Verdadeira
   b) Falsa

*****

Exercícios propostos

Desenvolva as seguintes funções e mostre os resultados em uma janela de diálogo:

1) Escreva uma função que calcule o maior de três números dados.
2) Escreva uma função que calcule o cubo de um número.
3) Escreva uma função que calcule o fatorial de um número dado.
4) Escreva uma função com 5 argumentos (números inteiros) e mostre somente se o número for par.
5) Escreva uma função cujo argumento seja uma string e o valor de retorno seja o tamanho dessa string.

*****