

MÓDULO

JAVASCRIPT/AJAX

UNIDADE

FUNÇÕES JAVASCRIPT

ÍNDICE

OBJETIVOS.....	3
INTRODUÇÃO.....	4
1. DEFINIÇÃO DE FUNÇÃO.....	5
2. ARGUMENTOS DAS FUNÇÕES.....	9
3. CHAMADAS PARA AS FUNÇÕES.....	12
4. FUNÇÕES JAVASCRIPT PREDEFINIDAS.....	13
4.1. PARSEINT.....	13
4.2. PARSEFLOAT.....	14
4.3. ISNAN.....	14
4.4. ISFINITE.....	16
4.5. NÚMERO E STRING.....	16
4.6. EVAL.....	17
CONCLUSÃO.....	19
AUTOAVALIAÇÃO.....	21
SOLUÇÕES.....	25
PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO.....	26
BIBLIOGRAFIA	27

OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Definir o conceito de função em JavaScript.
- Criar funções com e sem argumentos.
- Conhecer as funções predefinidas em JavaScript.

INTRODUÇÃO

A programação linear tem uma desvantagem particular: todo o código é executado seguindo a linha de tempo da execução. Graças às funções, pode definir o código que não é executado nesta ordem linear, mas é declarado para ser usado mais tarde. Deste modo, pode criar programas JavaScript mais úteis e funcionais.

1. DEFINIÇÃO DE FUNÇÃO



Definição

Uma **função** é um conjunto de instruções de código agrupadas sob um nome exclusivo e que pode ser chamado de qualquer lugar no código JavaScript. Estas funções podem gerar variáveis, modificar dados, alterar código HTML ou devolver resultados.

A criação de funções consiste na palavra reservada `function`, seguida por:

- **Nome da função:** elemento obrigatório e exclusivo, ou seja, não pode haver duas funções com o mesmo nome.
- **Argumento da função:** elemento entre parênteses e separado por vírgulas. Estes argumentos não são obrigatórios e o nome utilizado por eles será a variável com a qual irá trabalhar.
- **Código a ser executado:** elemento entre chavetas (`{}`), onde é colocado o código que pretende que seja executado dentro da função.

Se pretender que uma função devolva algum dado, terá de utilizar a palavra reservada `return`. Para isso, basta escrever `return` seguido pelos dados a serem devolvidos.

```
function funcao(){  
    return 4 * 5  
}
```

As funções podem receber tantos parâmetros quanto precisar de indicar. Estes parâmetros são indicados como valores e deve ter-se em consideração que, ao serem indicados como valores, as alterações nestes valores serão apenas consideradas localmente; fora da função continuarão a ser válidos como antes de os indicar para a função, a menos que modifique um valor global dentro da função.

Exemplo 01

Segue-se um exemplo de como uma função usa as variáveis locais e como atualiza uma variável global.

Exemplo:

```
<html>  
<head>  
<script type="text/javascript">  
function modVariables(contador){  
    contador = contador + 1;  
    alert(contador);  
}  
var contador;  
contador = 9;  
modVariables(contador);  
alert(contador);  
</script>  
</head>
```



```
<body>  
</body>  
</html>
```

Ao abrir este código num navegador, surgirá o alert do contador, que dentro da função devolve 10, enquanto o alert de fora da função, declarado antes da função, devolve 9, pois a função modifica a variável do contador localmente.



Visualização do método alert(), criado dentro da função.



Visualização do método alert(), criado fora da função.

No entanto, no próximo exemplo não é colocada a variável como um argumento da função. Além de defini-la como global antes de definir a função, se abrir este código, verá que em ambos os casos o valor devolvido pelo alert é 10.

Exemplo:

```
<html>
<head>
<script type="text/javascript">
var contador;
contador = 9;
function modVariables()
{
    contador = contador + 1;
    alert(contador);
}
modVariables();
alert(contador);
</script>
</head>
<body>
</body></html>
```



Visualização da página JavaScript.

2. ARGUMENTOS DAS FUNÇÕES

Os argumentos de uma função são os valores na forma de variáveis que se pretende passar para a função. Esses argumentos são locais e só são válidos na função.

Internamente, as funções armazenam os argumentos recebidos dentro de um array chamado `arguments` que tem as mesmas características de um array normal.

```
arguments[i]
```

Como qualquer outro array, cada um dos seus elementos pode ser utilizado através do índice que os define, sendo o primeiro valor deste índice **zero (0)**.

Também pode saber o número exato de argumentos utilizados, graças à função `length`:

```
arguments.length
```

Esta função é muito útil, já que o JavaScript permite criar funções que recebem vários argumentos de acordo com a definição da função e mais tarde indicam mais argumentos do que os estabelecidos por esta função. Isto pode ser muito útil nos casos em que não tenha a certeza de quantos valores vai passar para a função.

Exemplo 02

No exemplo seguinte, verá uma função que está definida com menos argumentos do que os que enviará posteriormente. Utilizando as funcionalidades de argumentos, irá aceder a todos os valores.

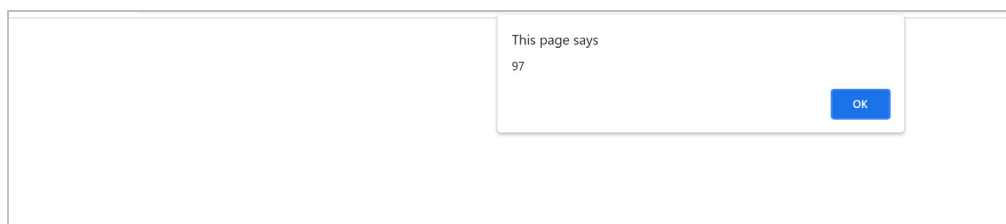
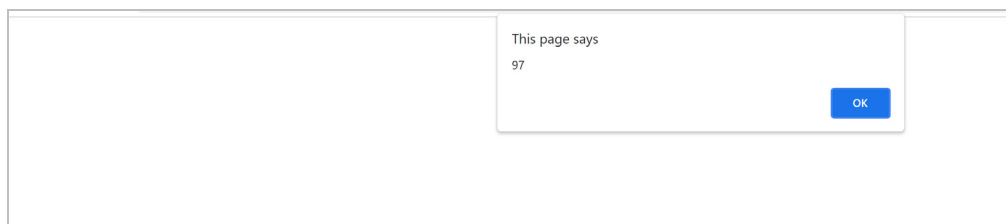
Exemplo:

```
<html>
<head>
<script type="text/javascript">
function calculadora(operador) {
  resultado = 0;
  for (var i = 1; i < arguments.length; i++){
    switch (operador){
      case "+":
        resultado = resultado + parseFloat(arguments[i]);
        break;
      case "-":
        resultado = resultado - parseFloat(arguments[i]);
        break;
      case "*":
        if (resultado == 0) {resultado = 1;} //evitar multiplicar
por 0
        resultado = resultado * parseFloat(arguments[i]);
        break;
    }
  }
  return resultado;
}
</script>
</head>
<body>
<script type="text/javascript">
```

```
alert(calculadora("+", "6", "23", "1", "67"));  
alert(calculadora("-", "567", "2", "32", "2"));  
alert(calculadora("*", "3", "2", "34", "2"));  
</script>  
</body>  
</html>
```

Neste exemplo, passará alguns argumentos que indicam a operação matemática a ser realizada e os valores a operar.

Se olhar com atenção, verá que a função espera por apenas um argumento, embora receba mais.



Visualização da página JavaScript.

3. CHAMADAS PARA AS FUNÇÕES

Uma função criada em JavaScript pode ser chamada de qualquer ponto do código, desde que a tenha definido antes, ou seja, não pode chamar uma função antes de a declarar. Por exemplo:

```
variable = somarDados(3,4);  
function somarDados(dado1,dado2){  
    return dado1 + dado2;  
}
```

Neste exemplo, a função “somarDados” é chamada antes de ser criada, o que irá gerar um erro em JavaScript e evitar que o resto do código seja executado.

Exceto no caso anterior, uma função pode ser chamada:

- A partir do código JavaScript.
- Dentro de outras funções.
- De qualquer evento HTML.
- Através de um link (<a href>) em HTML.
- Através de si mesma (funções recursivas).

4. FUNÇÕES JAVASCRIPT PREDEFINIDAS

O JavaScript possui várias funções predefinidas que ajudam a criar código de uma forma mais rápida. Segue-se uma lista com as principais e mais usadas:

4.1. PARSEINT

Esta função recebe uma string como argumento e devolve um valor numérico inteiro da string. A sua sintaxe é:

```
parseInt(texto,base);
```

Esta função lê no argumento da string e vai tentar convertê-lo num número inteiro, removendo as casas decimais do resultado se necessário, ou seja, se a string for por exemplo "3.1", irá apenas devolver o número 3.

No caso de não ser possível converter a string (por exemplo, se tiver uma letra por engano) vai devolver o valor NaN (Not a Number), que significa que não é um número.

Esta função recebe um segundo parâmetro opcional, denominado de base ou raiz. Se este parâmetro for especificado, a função tomá-lo-á como base específica para tentar converter a string. Por defeito, se este parâmetro não for inserido, a função assumirá uma base 10 ou decimal, mas é possível especificar 8 para base octal, 16 para hexadecimal, entre outros.

Se passar um parâmetro inválido para a base, a função irá ignorá-lo e tentar uma conversão de base 10.

4.2. PARSEFLOAT

Esta função recebe uma string como argumento e devolve um valor numérico da string, incluindo as casas decimais. A sua sintaxe é:

```
parseFloat (string);
```

Se no parâmetro string que passar para a função aparecer um carácter que não seja um número, um ponto decimal, um expoente ou os sinais de mais ou menos (+ ou -), ele assume o carácter inválido e descartará o resto para tentar converter a string.

Se o primeiro carácter na string for um carácter inválido, ele devolverá um NaN.

4.3. ISNAN

Esta função pega no parâmetro e verifica se não é um número, devolvendo "verdadeiro" se o valor do parâmetro não for um número.

Exemplo 03

Segue-se um exemplo do uso das funções parseFloat e isNaN. Para o fazer, converter-se-á uma série de dados, verificando se o resultado é "ok" e é um número.

Exemplo:

```
<html>
<head>
<script type="text/javascript">
```



```
function conversor(valor) {  
  valorConvertido=parseFloat(valor);  
  if (isNaN(valorConvertido)) {  
    alert('não é um número');  
  } else {  
    alert('é um número');  
  }  
}  
  
</script>  
</head>  
<body>  
<input type="text" name="valor" id="valor" /><br/>  
<input type="button" value="é um número ?" onclick =  
  "conversor(document.getElementById('valor').value)"/>  
</body>  
</html>
```

Neste exemplo, graças à função `document.getElement`, o valor é dinamicamente passado para a função JavaScript que vai tentar converter o valor e, dependendo do seu resultado, dirá se o valor é um número ou não.


Como pode ver na imagem seguinte, e verificar com este exemplo, a função `parseFloat` aceita uma string com letras, desde que não seja na primeira posição, e devolve o número que encontrar.



The screenshot shows a web browser window. On the left, there is a text input field containing the number '10'. Below it is a button labeled 'é um número ?'. On the right, an alert dialog box is displayed with the title 'This page says' and the message 'é um número'. There is an 'OK' button in the bottom right corner of the alert box.



The screenshot shows a web browser window. On the left, there is a text input field containing the string '5y'. Below it is a button labeled 'é um número ?'. On the right, an alert dialog box is displayed with the title 'This page says' and the message 'é um número'. There is an 'OK' button in the bottom right corner of the alert box.



The image shows a web form on the left with a text input field containing the letter 'x' and a label 'é um número?'. To the right of the form is a message box with the text 'This page says não é um número' and an 'OK' button.

Resultado do exemplo.

4.4. ISFINITE

Esta função permite passar um parâmetro para verificar se este é um número finito, devolvendo “verdadeiro”. Se o argumento for NaN ou infinito, irá devolver “falso”. A sua sintaxe é:

```
isFinite(número)
```

4.5. NÚMERO E STRING

Estas duas funções permitem converter um objeto num número ou string, por exemplo, uma data, uma hora, etc., em qualquer elemento do tipo “objeto”, (geralmente usado para objetos predefinidos em JavaScript, como Data).

A sintaxe correta é:

```
String(objeto)
```

```
Number(objeto)
```

Por exemplo, para converter o objeto PI da classe Math, é necessário:

```
D = Math.PI  
x = String(D)  
alert(x)
```

4.6. EVAL

A função eval passa uma string como parâmetro e a função tentará interpretar a string convertendo-a em código JavaScript, ou seja, tentando executar o conteúdo da string como se ela fosse um pedaço de código.

Exemplo 04

Neste exemplo será criada uma simples calculadora que utiliza um único campo para escrever as operações que pretende realizar e, graças ao método eval, devolve um resultado.

Exemplo:

```
<html>
<head>
<script type="text/javascript">
function calcular(formulario){
    formulario.resultado.value = eval(formulario.operacao.value);
}
</script>
</head>
<body>
<form name="calculadora">
String da operação
<input type="text" name="operacao" /><br>
Resultado:
<input type="text" name="resultado" /><br>
<input type="button" value="Calcular" onClick="calcular(this.form)">
</form>
</body>
</html>
```

String da operação

Resultado:

Resultado do exemplo.

CONCLUSÃO

As funções são uma parte muito importante de qualquer linguagem de programação, uma vez que permitem criar excertos de código independentes, que por sua vez permitem criar uma rede de chamadas entre funções para gerar scripts mais complexos.

Apreendeu a criar, usar e chamar funções, com e sem argumentos, para poder desenvolver as primeiras peças de código independentes de HTML.

AUTOAVALIAÇÃO

1. O que é uma função?

- a) Um elemento JavaScript padrão, não dinâmico.
- b) Um conjunto de declarações agrupadas sob o mesmo nome. O referido nome pode ser repetido, mas apenas se os argumentos que recebe forem diferentes.
- c) Um conjunto de declarações agrupadas sob o mesmo nome que não podem ser repetidas.
- d) Um conjunto de declarações simples sob o mesmo nome que podem ser repetidas.

2. Como devem ser os argumentos de uma função?

- a) Devem ser exatos e separados por vírgulas (,).
- b) Devem ser exatos e separados por ponto e vírgula (;).
- c) Não precisam de ser exatos e são separados por vírgulas (,).
- d) Não precisam de ser exatos e são separados por ponto e vírgula (;).

3. **Como são os valores indicados como parâmetros para as funções?**
 - a) São locais e afetam apenas a função.
 - b) São globais se os definir antes em JavaScript.
 - c) São locais, mas modificam o valor global.
 - d) São locais fora da função.

4. **Qual é a palavra reservada utilizada para nomear a matriz de argumentos recebidos em uma função?**
 - a) parameters.
 - b) arguments.
 - c) values.
 - d) arrays.

5. **Para aceder à quantidade de dados inseridos num array, que propriedade se utiliza?**
 - a) count.
 - b) number.
 - c) items.
 - d) length.

6. **Em qual dos seguintes locais do código não se pode chamar funções?**
 - a) Dentro de outras funções.
 - b) Dentro da mesma função.
 - c) A partir de um link<a href>.
 - d) Em qualquer parte do código, mesmo fora da tag<script>.

- 7. A função predefinida parseInt pode receber um segundo parâmetro. O que é que ele indica?**
- a) A base da raiz para a qual converter em string.
 - b) A precisão decimal do número devolvido.
 - c) Se a função irá ou não arredondar o resultado.
 - d) O segundo parâmetro só é executado se o primeiro for falso.
- 8. Quando é que a função parseFloat devolve NaN?**
- a) Se algum dos caracteres da string não for um número.
 - b) Se algum dos caracteres da string não for um número, um ponto decimal, um expoente ou um sinal de mais ou menos.
 - c) Se algum dos caracteres da string não for um número ou ponto decimal.
 - d) A função parseFloat nunca devolve NaN.
- 9. Qual é a função utilizada para descobrir se um número é finito?**
- a) isnumber.
 - b) notIsInfinite.
 - c) isFinite.
 - d) Não existe uma função predefinida para isso.
- 10. O que é que as funções number e string permitem fazer?**
- a) Converter um objeto num number ou string.
 - b) Converter uma variável de tipo de texto num number ou string.
 - c) Devolver verdadeiro, se o parâmetro indicado for um number ou uma string.
 - d) Devolver falso, se o parâmetro indicado for um number ou uma string.

SOLUÇÕES

1.	c	2.	c	3.	a	4.	b	5.	d
6.	d	7.	a	8.	b	9.	c	10.	a

PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

Pode aceder a mais informações e alguns exemplos práticos (e editá-los) no seguinte website:

- https://www.w3schools.com/js/js_functions.asp

BIBLIOGRAFIA

- MDN Web Docs (2021). "Funções". Disponível em:
<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Fun%C3%A7%C3%B5es>. Consultado a 28 de janeiro de 2021.
- VV. AA. (2010). *JavaScript*. Madrid: AnayaMultimedia

