

## Funções com JavaScript

Funções são blocos de instrução responsáveis por procedimentos específicos realizados através de uma chamada que atende a uma necessidade da aplicação. Para declarar uma função se utiliza a palavra-chave `function` seguida por um nome e os parâmetros (argumentos) listados dentro dos parênteses e separados por vírgula, para finalizar se declara o procedimento (instruções) que será executado entre as chaves.

### Sintaxe:

```
function myFunction(parameter){  
    myScript;  
}
```

### Exemplo:

```
function quadrado(x){  
    return x * x;  
}
```

### Observação:

Quando o parâmetro é um dado primitivo alterações realizadas no valor do parâmetro não são refletidas globalmente. Quando o parâmetro é um objeto as alterações realizadas nos atributos (propriedades) do objeto são refletidas globalmente.

### Exemplo:

```
var a = 10; //variável primitiva  
var b = { valor: 10 }; //objeto inicializado com o atributo valor = 10  
function quadrado(x) {  
    x = x * x;  
    return x;  
}  
function cubo(x) {  
    x.valor = x.valor * x.valor * x.valor;  
    return x.valor;  
}  
var c = quadrado(a); //c = 100, a = 10  
var d = cubo(b); //d = 1000, b.valor = 1000
```

## Função lambda ou função flecha (=>).

A função lambda funciona da mesma forma que uma função callback (função de retorno). O operador representa uma forma simplificada de escrever uma função.

### Sintaxe:

```
(parameter) => { myScript; }
```

Os parênteses à esquerda são os parâmetros da função e o operador que está à direita é o corpo da função com expressão de retorno.

#### Exemplo:

```
var a = 10;
var b = (x) => { return x * x; }
var c = (x) => x * x; // expressão equivalente a linha anterior.
document.write(
  '<br>b: ' + b(a) +
  '<br>c: ' + c(a)
);
```

## Manipulação de Elementos HTML com JavaScript

Documentos do tipo HTML quando são carregados em um navegador de internet se transformam em objetos de documento. Para manipular elementos em documentos HTML utilizamos os métodos definidos pelo Modelo de Objeto de Documento (Document Object Model - DOM). Este modelo especifica os métodos utilizados para manipular o objeto **document** e seus componentes. Podemos acessar o objeto *document* de duas maneiras: [window.document](#) ou simplesmente [document](#). O window é o objeto que representa a janela do navegador. Veja a seguir as principais propriedades e métodos utilizados pelo DOM.

### Manipulação de elementos:

#### getElementById()

O método getElementById() retorna o elemento que possui o atributo id especificado no parâmetro

#### Sintaxe:

```
document.getElementById("idName");
```

#### Exemplo:

```
<!DOCTYPE html>
<html>
<body>
  <p id="myId"></p>
  <script>
    var myElement = document.getElementById("myId");
    alert(myElement.tagName);
  </script>
</body>
</html>
```

## getElementsByClassName()

O método `getElementsByClassName()` retorna um Array com uma coleção de elementos HTML (HTMLCollection) que possuem o nome da classe especificado no parâmetro.

### Sintaxe:

```
document.getElementsByClassName("className");
```

### Exemplo:

```
<!DOCTYPE html>
<html>
<body>
  <p id="myId" class="myClass"></p>
  <script>
    var myElement = document.getElementsByClassName("myClass");
    alert(myElement[0].tagName);
  </script>
</body>
</html>
```

## getElementsByTagName()

O método `getElementsByTagName()` retorna um Array com uma coleção de elementos HTML (HTMLCollection) que possuem a tag especificado no parâmetro.

### Sintaxe:

```
document.getElementsByTagName("tagName");
```

### Exemplo:

```
<!DOCTYPE html>
<html>
<body>
  <p id="myId" class="myClass"></p>
  <script>
    var myElement = document.getElementsByTagName("p");
    alert(myElement[0].tagName);
  </script>
</body>
</html>
```

## createElement()

O método `createElement()` cria o elemento especificado no parâmetro.

### Sintaxe:

```
document.createElement("tagName");
```

**Exemplo:**

```
<!DOCTYPE html>
<html>
<body>
  <p id="myId" class="myClass"></p>
  <script>
    var myElement = document.createElement("button");
    alert(myElement.tagName);
  </script>
</body>
</html>
```

**appendChild()**

O método appendChild() adiciona um novo elemento filho ao elemento HTML selecionado

**Sintaxe:**

```
elementParent.appendChild(elementChild);
```

**Exemplo:**

```
<!DOCTYPE html>
<html>
<head>
  <style>
    button{
      width:90px;
      height: 30px;
    }
  </style>
</head>
<body>
  <p id="myId" class="myClass"></p>
  <script>
    var myElement = document.createElement("button");
    document.getElementById("myId").appendChild(myElement);
  </script>
</body>
</html>
```

**remove()**

O método remove() remove o elemento selecionado.

**Sintaxe:**

```
element.remove();
```

**Exemplo:**

```
<!DOCTYPE html>
<html>
<body>
  <p id="myId">meu parágrafo</p>
  <button onclick="myFunction()">click aqui</button>
  <script>
    function myFunction() {
      var myElement = document.getElementById("myId");
      myElement.remove();
    }
  </script>
</body>
</html>
```

## activeElement

A propriedade activeElement retorna o elemento que está focado

### Sintaxe:

```
document.activeElement;
```

### Exemplo:

```
<!DOCTYPE html>
<html>
<body onclick="myFunction()">
  <input type="text" value="input-text">
  <button>botão</button>
  <script>
    function myFunction() {
      var myElement = document.activeElement;
      alert(myElement.tagName);
    }
  </script>
</body>
</html>
```

## Manipulação de Atributos:

### setAttribute(), getAttribute()

O método **setAttribute()** cria ou modifica o atributo especificado no parâmetro. O método **getAttribute()** retorna o valor do atributo especificado no parâmetro.

### Sintaxe:

```
element.setAttribute('attribute', 'attrValue');
element.getAttribute('attribute');
```

### Exemplo:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    #myId{
      color: white;
      background-color: black;
    }
  </style>
</head>
<body>
  <button id="myButton" onclick="addAttribute()">Aplicar Atributos</button>
  <button id="myButton" onclick="showAttribute()">Exibir Atributos</button>
  <script>
    function addAttribute() {
      var myElement = document.createElement("input");
      myElement.setAttribute('id', 'myId');
      myElement.setAttribute('placeholder', 'Digite aqui');
      document.body.appendChild(myElement);
    }
    function showAttribute() {
      alert(
        document.getElementById('myId').getAttribute('id') + '\n' +
        document.getElementById('myId').getAttribute('placeholder'));
    }
  </script>
</body>
</html>
```

### removeAttribute()

O método removeAttribute() remove o atributo especificado no parâmetro do elemento selecionado.

#### Sintaxe:

```
element.removeAttribute("attribute");
```

### Exemplo:

```
<!DOCTYPE html>
<html>
<body>
  <a id="myAnchor" href="https://www.google.com">link para o google</a>
  <button onclick="myFunction()">Remover link</button>
  <script>
    function myFunction() {
      document.getElementById("myAnchor").removeAttribute("href");
    }
  </script>
</body>
</html>
```

```
}  
</script>  
</body>  
</html>
```

## className

A propriedade **className** define ou retorna o valor do atributo class do elemento selecionado

### Sintaxe:

```
element.className = "className"; //define o valor  
element.className; //retorna o valor
```

### Exemplo:

```
<!DOCTYPE html>  
<html>  
<head>  
  <style>  
    .class01 {  
      background-color: lightcoral;  
    }  
    .class02 {  
      color: lightblue;  
    }  
  </style>  
</head>  
<body>  
  <div id="myId">  
    <p>Click no botão para inserir as classes.</p>  
  </div>  
  <button onclick="myFunction()">Click</button>  
  <script>  
    function myFunction() {  
      var myElement = document.getElementById("myId");  
      myElement.className = "class01 class02";  
    }  
  </script>  
</body>  
</html>
```

## style

A propriedade **style** define ou retorna o valor do atributo style do elemento selecionado. Para aplicar os valores é necessário informar a propriedade após o estilo através da seguinte sintaxe: **style.cssProperty**.

Propriedades compostas por mais de uma palavra devem ser escritas no formato *lowerCamelCase* (border-top deve ser escrito como borderTop).

#### Sintaxe:

```
element.style.cssProperty = "propertyValue"; //define o valor  
element.style.cssProperty; //retorna o valor
```

#### Exemplo:

```
<!DOCTYPE html>  
<html>  
<head>  
</head>  
<body>  
  <p id="myId">Estilo do meu parágrafo:</p>  
  <button onclick="myFunction()">Click</button>  
  <script>  
    function myFunction() {  
      document.getElementById("myId").style.color = "lightblue";  
      document.getElementById("myId").style.backgroundColor = "coral";  
      document.getElementById("myId").style.borderBottom = "6px solid red";  
    }  
  </script>  
</body>  
</html>
```

## Manipulação de conteúdo

### innerHTML

A propriedade **innerHTML** define ou retorna o conteúdo do elemento HTML selecionado

#### Sintaxe:

```
element.innerHTML; //retorna o conteúdo  
element.innerHTML = "conteúdo"; //define o conteúdo
```

#### Exemplo:

```
<!DOCTYPE html>  
<html>  
<head>  
  <style>  
    span{  
      color: blue;  
    }  
  </style>  
</head>  
<body>
```



```
<p id="myId">Este é meu parágrafo, </p>
<button onclick="myFunction()">Click</button>
<script>
  function myFunction() {
    document.getElementById("myId").innerHTML += "<span>este é meu span!</span>";
  }
</script>
</body>
</html>
```

## innerText

A propriedade **innerText** define ou retorna conteúdo de texto no elemento HTML selecionado

### Sintaxe:

```
element.innerText; //retorna o texto
element.innerText = "conteúdo"; //define o texto
```

### Exemplo:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    span{
      color: blue;
    }
  </style>
</head>
<body>
  <p id="myId">Este é meu parágrafo, </p>
  <button onclick="myFunction()">Click</button>
  <script>
    function myFunction() {
      document.getElementById("myId").innerText += "este é meu novo texto!";
    }
  </script>
</body>
</html>
```

## Como obter valores de um elemento <select> (dropdown) com JavaScript:

Para obter o elemento **option** (<option>) de um elemento **select** (<select>) utilizamos a expressão: **mySelect.option[mySelect.selectedIndex]**, e adicionamos as propriedades **value**, para retornar o atributo **value**, ou **text** para retornar o conteúdo textual do elemento **option**.

### Sintaxe:

```
select.options[select.selectedIndex].value; //retorna o atributo value
select.options[select.selectedIndex].text; //retorna o conteúdo do elemento
```

#### Exemplo:

```
<!DOCTYPE html>
<html>
<body>
  <select id="cor">
    <option value="#0000aa" selected>Azul</option>
    <option value="#00aa00">Verde</option>
    <option value="#aa0000">Vermelho</option>
  </select>
  <button onclick="myFunction()">Click aqui</button>
  <script>
    var select = document.getElementById('cor');
    var optionValue = select.options[select.selectedIndex].value;
    var optionText = select.options[select.selectedIndex].text;
    function myFunction() {
      alert('valor: ' + optionValue + '\nCor: ' + optionText);
    }
  </script>
</body>
</html>
```

## Eventos com JavaScript

### Eventos com teclado (keyboardEvent):

1. **keypress:** é acionado quando uma tecla que produz um caractere é pressionada. Teclas que não produzem caracteres não acionam o keypress.
2. **keydown:** é acionado quando uma tecla é pressionada, inclusive teclas que não produzem caracteres.
3. **keyup:** é acionado quando uma tecla é pressionada e liberada, inclusive teclas que não produzem caracteres.

### keypress

#### html:

```
<element onkeypress="myFunction()"></element>
```

#### JS (método onkeypress):

```
object.onkeypress = myFunction;
object.onkeypress = function(){ myScript; }
object.onkeypress = (event) => { myScript; }
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.onkeypress = myFunction;  
document.getElementById("myId").onkeypress = myFunction;
```

#### **JS (método addEventListener):**

```
object.addEventListener("keypress", myFunction);  
object.addEventListener("keypress", function(){ myScript; });  
object.addEventListener("keypress", (event) => { myScript; });
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.addEventListener("keypress", myFunction);  
document.getElementById("myId").addEventListener("keypress", myFunction);
```

### keydown

#### **html:**

```
<element onkeydown="myFunction()"></element>
```

#### **JS (método onkeydown):**

```
object.onkeydown = myFunction;  
object.onkeydown = function(){ myScript; }  
object.onkeydown = (event) => { myScript; }
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.onkeydown = myFunction;  
document.getElementById("myId").onkeydown = myFunction;
```

#### **JS (método addEventListener):**

```
object.addEventListener("keydown", myFunction);  
object.addEventListener("keydown", function(){ myScript; });  
object.addEventListener("keydown", (event) => { myScript; });
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.addEventListener("keydown", myFunction);  
document.getElementById("myId").addEventListener("keydown", myFunction);
```

### keyup

#### **html:**

```
<element onkeyup="myFunction()"></element>
```

#### **JS (método onkeyup):**

```
object.onkeyup = myFunction;  
object.onkeyup = function(){ myScript; }  
object.onkeyup = (event) => { myScript; }
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.onkeyup = myFunction;
```

```
document.getElementById("myId").onkeyup = myFunction;
```

### JS (método `addEventListener`):

```
object.addEventListener("keyup", myFunction);  
object.addEventListener("keyup", function(){ myScript; });  
object.addEventListener("keyup", (event) => { myScript; });
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.addEventListener("keyup", myFunction);  
document.getElementById("myId").addEventListener("keyup", myFunction);
```

### Evento com tecla específica

O método *event.key* retorna uma *string* com o valor da tecla pressionada. Dessa forma, podemos utilizar o *event.key* em uma estrutura condicional (if-else, switch-case ou ternário) para executar um bloco de instrução caso uma tecla específica seja pressionada.

#### Exemplo:

```
<!DOCTYPE html>  
<html>  
<body id="myBody">  
  <script>  
    document.onkeyup = function (event) {  
      if (event.key == 'Enter') {  
        alert('Você pressionou o: ' + event.key);  
      }  
      else {  
        switch (event.key) {  
          case 'ArrowUp':  
            alert('Up');  
            break;  
          case 'ArrowDown':  
            alert('Down');  
            break;  
          case 'ArrowLeft':  
            alert('Left');  
            break;  
          case 'ArrowRight':  
            alert('Right');  
            break;  
          default:  
            alert(event.key);  
            break;  
        }  
      }  
    }  
  }  
}
```

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

## Eventos com mouse (MouseEvent):

1. **click**: o evento disparado quando o elemento é clicado.
2. **dblclick**: o evento é disparado quando ocorre um click duplo no elemento.
3. **contextmenu**: o evento disparado quando o elemento é clicado com o botão direito do mouse
4. **mousedown**: o evento é disparado quando qualquer botão do mouse é clicado sobre o elemento.
5. **mouseup**: o evento é disparado quando qualquer botão do mouse é liberado sobre o elemento.
6. **mouseenter**: o evento é disparado quando o ponteiro do mouse entre no elemento.
7. **mouseleave**: o evento é disparado quando o ponteiro do mouse sai do elemento.

### click

#### html:

```
<element onclick="myFunction()"></element>
```

#### JS (método onclick):

```
object.onclick = myFunction;
object.onclick = function(){ myScript; }
object.onclick = (event) => { myScript; }
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.onclick = myFunction;
document.getElementById("myId").onclick = myFunction;
```

#### JS (método addEventListener):

```
object.addEventListener("click", myFunction);
object.addEventListener("click", function(){ myScript; });
object.addEventListener("click", (event) => { myScript; });
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.addEventListener("click", myFunction);
document.getElementById("myId").addEventListener("click", myFunction);
```

### dblclick

#### html:

```
<element ondblclick="myFunction()"></element>
```

#### JS (método ondblclick):

```
object.ondblclick = myFunction;
object.ondblclick = function(){ myScript; }
object.ondblclick = (event) => { myScript; }
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.ondbclick = myFunction;  
document.getElementById("myId").ondbclick = myFunction;
```

#### **JS (método addEventListener):**

```
object.addEventListener("dblclick", myFunction);  
object.addEventListener("dblclick", function(){ myScript; });  
object.addEventListener("dblclick", (event) => { myScript; });
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.addEventListener("dblclick", myFunction);  
document.getElementById("myId").addEventListener("dblclick", myFunction);
```

### contextmenu

#### **html:**

```
<element oncontextmenu ="myFunction()"></element>
```

#### **JS (método ondblclick):**

```
object.oncontextmenu = myFunction;  
object.oncontextmenu = function(){ myScript; }  
object.oncontextmenu = (event) => { myScript; }
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.oncontextmenu = myFunction;  
document.getElementById("myId").oncontextmenu = myFunction;
```

#### **JS (método addEventListener):**

```
object.addEventListener("contextmenu", myFunction);  
object.addEventListener("contextmenu", function(){ myScript; });  
object.addEventListener("contextmenu", (event) => { myScript; });
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.addEventListener("contextmenu", myFunction);  
document.getElementById("myId").addEventListener("contextmenu", myFunction);
```

### mousedown

#### **html:**

```
<element onmousedown="myFunction()"></element>
```

#### **JS (método ondblclick):**

```
object.onmousedown = myFunction;  
object.onmousedown = function(){ myScript; }  
object.onmousedown = (event) => { myScript; }
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.onmousedown = myFunction;  
document.getElementById("myId").onmousedown = myFunction;
```

#### **JS (método addEventListener):**

```
object.addEventListener("mousedown", myFunction);  
object.addEventListener("mousedown", function(){ myScript; });  
object.addEventListener("mousedown", (event) => { myScript; });
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.addEventListener("mousedown ", myFunction);  
document.getElementById("myId").addEventListener("mousedown ", myFunction);
```

### **mouseup**

#### **html:**

```
<element onmouseup="myFunction()"></element>
```

#### **JS (método ondblclick):**

```
object.onmouseup = myFunction;  
object.onmouseup = function(){ myScript; }  
object.onmouseup = (event) => { myScript; }
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.onmouseup = myFunction;  
document.getElementById("myId").onmouseup = myFunction;
```

#### **JS (método addEventListener):**

```
object.addEventListener("mouseup", myFunction);  
object.addEventListener("mouseup", function(){ myScript; });  
object.addEventListener("mouseup", (event) => { myScript; });
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.addEventListener("mouseup", myFunction);  
document.getElementById("myId").addEventListener("mouseup", myFunction);
```

### **mouseenter**

#### **html:**

```
<element onmouseenter="myFunction()"></element>
```

#### **JS (método ondblclick):**

```
object.onmouseenter = myFunction;  
object.onmouseenter = function(){ myScript; }  
object.onmouseenter = (event) => { myScript; }
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.onmouseenter = myFunction;
```

```
document.getElementById("myId").onmouseenter = myFunction;
```

#### JS (método `addEventListener`):

```
object.addEventListener("mouseenter", myFunction);  
object.addEventListener("mouseenter", function(){ myScript; });  
object.addEventListener("mouseenter", (event) => { myScript; });
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.addEventListener("mouseenter", myFunction);  
document.getElementById("myId").addEventListener("mouseenter", myFunction);
```

### mouseleave

#### html:

```
<element onmouseleave = "myFunction()"></element>
```

#### JS (método `onmouseleave`):

```
object.onmouseleave = myFunction;  
object.onmouseleave = function(){ myScript; }  
object.onmouseleave = (event) => { myScript; }
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.onmouseleave = myFunction;  
document.getElementById("myId").onmouseleave = myFunction;
```

#### JS (método `addEventListener`):

```
object.addEventListener("mouseleave", myFunction);  
object.addEventListener("mouseleave", function(){ myScript; });  
object.addEventListener("mouseleave", (event) => { myScript; });
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.addEventListener("mouseleave", myFunction);  
document.getElementById("myId").addEventListener("mouseleave", myFunction);
```

### Eventos com foco (Focus Event):

1. **focus:** o evento é disparado quando o elemento ganha o foco.
2. **blur:** o evento disparado quando o elemento perde o foco.

### focus

#### html:

```
<element onfocus = "myFunction()"></element>
```

#### JS (método `onfocus`):

```
object.onfocus = myFunction;  
object.onfocus = function(){ myScript; }  
object.onfocus = (event) => { myScript; }
```



**OBS:** object pode ser o próprio documento ou um elemento.

```
document.onfocus = myFunction;  
document.getElementById("myId").onfocus = myFunction;
```

**JS (método addEventListener):**

```
object.addEventListener("focus", myFunction);  
object.addEventListener("focus", function(){ myScript; });  
object.addEventListener("focus", (event) => { myScript; });
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.addEventListener("focus", myFunction);  
document.getElementById("myId").addEventListener("focus", myFunction);
```

## blur

**html:**

```
<element onblur ="myFunction()"></element>
```

**JS (método ondblclick):**

```
object.onblur = myFunction;  
object.onblur = function(){ myScript; }  
object.onblur = (event) => { myScript; }
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.onblur = myFunction;  
document.getElementById("myId").onblur = myFunction;
```

**JS (método addEventListener):**

```
object.addEventListener("blur", myFunction);  
object.addEventListener("blur", function(){ myScript; });  
object.addEventListener("blur", (event) => { myScript; });
```

**OBS:** object pode ser o próprio documento ou um elemento.

```
document.addEventListener("blur", myFunction);  
document.getElementById("myId").addEventListener("blur", myFunction);
```