

# Javascript – Módulo 3 – B7Web

## Sumário

Aula 01 – Evento de Clique.....	2
Aula 02 – Evento de Teclado.....	3
Aula 03 – Mudança de Estilo.....	4
Aula 04 – Array.....	5
Aula 05 – Objeto (1/2).....	5
Aula 06 – Objeto (2/2).....	6
Aula 07 – Projeto: Menu de Navegação.....	7
Aula 08 – Switch.....	7
Aula 09 – Loop For.....	8
Aula 10 – Loop While.....	9
Aula 11 – querySelector e querySelectorAll.....	9

# Aula 01 – Evento de Clique

**onclick** → Executa JavaScript ao clicar sobre o elemento determinado, podendo este elemento ser um button ou qualquer outra tag HTML.

O atributo 'onclick' pode receber código JavaScript diretamente em si mesmo, desta forma, podemos digitar o JavaScript aqui, diretamente no HTML. Obs.: Acredito que esse não seja um modo usual (não deve ser uma boa prática). Ao invés de digitarmos o JavaScript diretamente no texto HTML, utilizaríamos uma função que seria chamada de um documento JavaScript.

HTML:

```
<h1 id="titulo">M03a01</h1>
<input id="campo" type="text" name="usuario" value="Nome do usuário">
<!--<button onclick="clique()">Clique em mim</button-->
<button onclick="document.querySelector('#titulo').innerHTML='Texto2'">Clique em mim</button>
```

No HTML acima, ao clicar no botão 'Clique em mim', o texto do <h1> é alterado para 'Texto2'.

Exemplo da função 'onclick' em um <h2>

```
<h2 id="titulo2" onclick="alert('Clicou')">Título 2</h2>
```

No exemplo acima, ao clicarmos no <h2> 'Título 2', o alert 'Clicou' é exibido.

---

**this** → Este comando seleciona o próprio elemento. É muito utilizado. Usá-lo, como no exemplo abaixo, seria equivalente ao dar um id para este button e no onclick chamar a ele mesmo através de um querySelector, por exemplo. Porém o comando this é muito mais simplificado.

HTML: `<button onclick="this.innerHTML='Clicou'">Botão 2</button>`

No HTML acima, ao clicar no botão 'Botão 2', o texto do próprio botão é alterado para 'Clicou'.

---

**onmouseover** → Com o comando 'onmouseover', podemos determinar que um evento aconteça quando passamos o cursor do mouse sobre o elemento escolhido.

**onmouseout** → Com o comando 'onmouseout', podemos determinar que um evento aconteça quando retiramos o cursor do mouse do elemento.

HTML:

```
<h2 id="titulo3" onmouseover="this.innerHTML='Passou o Mouse Sobre'"
onmouseout="this.innerHTML='Tirou o Mouse'">Título 3</h2>
```

No exemplo acima, ao passarmos o cursor do mouse sobre o <h2> 'Título 3', o texto do próprio Título 3 é alterado para 'Passou o Mouse Sobre'. E ao retirarmos o cursor do mouse do <h2>, o texto do próprio <h2> é alterado para 'Tirou o mouse'.

---

# Aula 02 – Evento de Teclado

**onkeydown** → Este comando aciona um evento no momento em que o usuário pressiona uma determinada tecla do teclado

HTML:

```
<input onkeydown="digitou1()" type="text" id="campo1" name="nome1" placeholder="Primeiro Nome">
<div id="res1"></div>
```

JavaScript:

```
function digitou1(){
    document.querySelector('#res1').innerHTML = 'Pressionou alguma tecla!'
}
```

Neste exemplo acima, ao pressionar alguma tecla no <input> 'nome1', o comando 'onkeydown' executa a função 'digitou1' do nosso JavaScript.

**onkeyup** → Este comando aciona um evento no momento em que o usuário solta uma determinada tecla do teclado que estava pressionada.

HTML:

```
<input onkeyup="digitou2()" type="text" id="campo2" name="nome2" placeholder="Segundo Nome">
<div id="res2"></div>
```

JavaScript:

```
function digitou2(){
    document.querySelector('#res2').innerHTML = 'Soltou alguma tecla!'
}
```

Neste exemplo acima, ao soltarmos uma tecla que estava pressionada no <input> 'nome2', o comando 'onkeyup' executa a função 'digitou2' do nosso JavaScript.

**onkeypress** → Este comando aciona um evento no momento em que o usuário pressiona e solta uma determinada tecla do teclado.

HTML:

```
<input onkeypress="digitou3()" type="text" id="campo3" name="nome3" placeholder="Terceiro Nome">
<div id="res3"></div>
```

JavaScript:

```
function digitou3(){
    document.querySelector('#res3').innerHTML = `Função Press! ${++i1}`
}
```

Neste exemplo acima, ao realizar o movimento completo de pressionar e soltar alguma tecla no <input> 'nome3', o comando 'onkeypress' executa a função 'digitou3' do nosso JavaScript.

**Obs.:** o evento **onkeypress** não é disparado para todas as teclas (por exemplo, ALT, CTRL, SHIFT, ESC) em todos os navegadores. Para detectar apenas se o usuário pressionou uma tecla, use o evento onkeydown, porque ele funciona para todas as teclas. (fonte: [https://www.w3schools.com/jsref/event\\_onkeypress.asp](https://www.w3schools.com/jsref/event_onkeypress.asp))

**event** → 'event' não é um simples parâmetro, e sim um comando. Assim como o 'this' trata do próprio elemento, o 'event' trata do próprio evento.

Obs.: O comando 'event' retorna um object em JavaScript e podemos acessar qualquer informação deste objeto. Veja o exemplo de como acessamos as propriedades do 'event' através do JavaScript.

HTML:

```
<input onkeypress="digitou4(event)" type="text" id="campo4" name="nome4" placeholder="Quarto Nome">
<div id="res4"></div>
```

## JavaScript:

```
function digitou4(e){
    document.querySelector('#res4').innerHTML = `Tecla: ${e.key} - Código: ${e.keyCode}`
    console.log(e)
}
```

Neste exemplo acima, ao executarmos o 'onkeypress' no <input> 'nome4', o comando 'onkeypress' executa a função 'digitou4' e envia o comando 'event' como um parâmetro para o nosso JavaScript e recebemos dentro do JavaScript com o nome de 'e'. Aqui, imprimimos na tela a tecla que foi digitada utilizando o parâmetro '.key' e imprimimos também o código da tecla utilizando o parâmetro '.keyCode'

Neste próximo exemplo, realizamos determinadas ações, de acordo com o código de alguma tecla previamente escolhida. Neste caso, utilizaremos o código da tecla 'Enter' que é o número 13.:

## HTML:

```
<input onkeypress="digitou5(event)" type="text" id="campo5" name="nome5" placeholder="Quarto Nome">
<div id="res5"></div>
```

## JavaScript:

```
function digitou5(e){
    if(e.keyCode == 13){// 13 é o código da tecla 'Enter'
        let texto = document.querySelector('#campo5').value
        let c = texto.length
        document.querySelector('#res5').innerHTML = `${c} Caracteres <br> Você digitou: ${texto}`
    }
}
```

No exemplo acima, enviamos novamente o comando 'event' para a function digitou5 que recebeu o texto digitado na variável 'texto', somou os caracteres da variável com o '.length' e retornou o texto e o total de caracteres que ele contém.

---

# Aula 03 – Mudança de Estilo

Realizando mudanças de Classe de um elemento HTML através do comando **classList** do JavaScript

**Obs.:** Este conceito do comando classList já tínhamos visto na **aula 05 do módulo 02**, porém aqui, aprimoramos o código, dividindo as da troca de classes em funções. Veja o exemplo:

## HTML:

```
<h1 id="titulo">Seja bem vindo!</h1>
<button onclick="azul()">Azul</button>
<button onclick="vermelho()">Vermelho</button>
<button onclick="verde()">Verde</button>
```

## CSS:

```
.azul{
    background-color: #0000FF;
    color: white;
}
.vermelho{
    background-color: #ff0000;
    color: white;
}
.verde{
    background-color: green;
    color: black;
}
```

## JavaScript:

```
function azul(){
    limpar()
    document.querySelector('#titulo').classList.add('azul')
}

function vermelho(){
    limpar()
    document.querySelector('#titulo').classList.add('vermelho')
}

function verde(){
    limpar()
    document.querySelector('#titulo').classList.add('verde')
}

function limpar(){
    document.querySelector('#titulo').classList.remove('azul')
    document.querySelector('#titulo').classList.remove('vermelho')
    document.querySelector('#titulo').classList.remove('verde')
}
```

Nos arquivos desta aula tem ainda mais uma tentativa de um exercício com botão que foi proveitoso, mas não foi bem sucedido. Veja os arquivos da aula.

---

## Aula 04 – Array

Obs.: É possível, em JavaScript, armazenar um array dentro de outro array.

```
let ingredientes = [['uva', 'pera', 'maçã'], ['arroz', 'feijão', 'macarrão']]
console.log(ingredientes)
// Resultado -> [Array(3), Array(3)]
console.log(ingredientes[1])
// Resultado -> ['arroz', 'feijão', 'macarrão']
console.log(ingredientes[1][0])
// Resultado -> 'arroz'
```

## Aula 05 – Objeto (1/2)

Diferença básica entre Array e Object:

- **Array** é uma listagem **numerada**
- **Object** é uma listagem **nomeada**

Sintaxe:

- Declaração (sintaxe) de um array → **let nomeDoArray = [ ]**
- Declaração (sintaxe) de um object → **let nomeDoObject = { }**

**OBS.:** No Object declaramos o nome do campo, que é chamado de **propriedade**, adicionamos **:**. Depois colocamos a variável que será armazenada nesta propriedade. **Separamos uma propriedade da outra com vírgulas.**

## EXEMPLO:

```
let carro = {  
  marca: 'Fiat',  
  modelo: 'Uno',  
  peso: '800kg',  
  ligar: function(){  
    console.log('VRUM VRUM')  
  },  
  acelerar: function(){  
    console.log('RIRIRHIHIHIH')  
  }  
}
```

Podemos acessar os valores do Object de duas maneiras:

```
console.log(carro['marca'])  
console.log(carro.modelo) // Esta é a mais usual.
```

Acessamos uma **function** armazenada em um object da seguinte forma:

```
carro.ligar()  
carro.acelerar()
```

---

## Aula 06 – Objeto (2/2)

**this** → Comando utilizado para referenciar o próprio object, após o this, utilizamos o ponto a adicionamos o nome da propriedade que queremos acessar

Veja a sua utilização no exemplo abaixo:

```
let carro = {  
  marca: 'Fiat',  
  modelo: 'Uno',  
  peso: '800kg',  
  ligado: false,  
  ligar: function(){  
    this.ligado = true  
    console.log(`Ligando o ${this.modelo}`)  
    console.log('VRUM VRUM')  
  },  
  acelerar: function(){  
    if(this.ligado == true){  
      console.log('RIRIRHIHIHIH')  
    }else{  
      console.log(`O ${this.marca} ${this.modelo} não está ligado`)  
    }  
  }  
}  
  
carro.ligar()  
carro.acelerar()
```

---

## Armazenando um object dentro de um array:

Sintaxe:

```
let carros = [
  {marca: 'Fiat', modelo: 'Pálio'},
  {marca: 'Volks', modelo: 'Fusca'},
  {marca: 'Toyota', modelo: 'Corolla'},
  {marca: 'Jeep', modelo: 'Compass'}
]
```

**Obs.:** Interessante observar, que dentro de um array, o object em si, não recebe um nome específico. Neste caso, o object é acessado através do nome do array e adicionando a sua posição (numérica) em que está localizado.

---

### Formas de acessar o array e os objects contidos nele:

→ Desta forma acessamos todo o array:

```
console.log(carros)
```

E temos o seguinte retorno:

```
[{...}, {...}, {...}, {...}]
```

---

→ Deste forma, acessamos um object por completo que está localizado em uma determinada posição dentro deste array. Neste exemplo acessamos a posição [2]

```
console.log(carros[2])
```

E temos o seguinte retorno:

```
{marca: 'Toyota', modelo: 'Corolla'}
```

---

→ E, desta forma, podemos acessar uma propriedade específica que está dentro de um object determinado pertencente a este array: Neste exemplo acessamos a propriedade modelo do object que está na posição [3] do nosso array.

```
console.log(carros[3].modelo)
```

Temos o seguinte retorno:

```
Compass
```

---

## Aula 07 – Projeto: Menu de Navegação

Desafio do Menu expansível.

Olhar o código para analisar.

---

## Aula 08 – Switch

Quando usamos o **switch ao invés do if**:

Quando temos um valor específico e precisamos, a partir deste valor, fazer várias condicionais diferentes.

Exemplo de código:

Declaração de variáveis:

```
let dia = 7
let diaNome
let semana
```

```

switch(dia){
  case 1:
    diaNome = 'Domingo'
    break
  case 2:
    diaNome = 'Segunda-feira'
    break
  case 3:
    diaNome = 'Terça-feira'
    break
  case 4:
    diaNome = 'Quarta-feira'
    break
  case 5:
    diaNome = 'Quinta-feira'
    break
  case 6:
    diaNome = 'Sexta-feira'
    break
  case 7:
    diaNome = 'Sábado'
    break
  default:
    diaNome = 'Erro'
}

```

```

switch (dia) {
  case 1:
    semana = 'final de semana'
    break;
  case 2:
  case 3:
  case 4:
  case 5:
    semana = 'dia normal'
    break;
  case 6:
    semana = 'dia da preparação'
    break;
  case 7:
    semana = 'dia do SENHOR!'
    break;
  default:
    semana = ''
}

```

Obs.: Dois pontos para observar:

**default** → O comando default dentro do switch é utilizado para declararmos o que ocorrerá caso nenhuma das condições acima forem satisfeitas.

**case** → Empilhamento de cases na segunda figura acima. Utilizado para casos em que valores diferentes necessitam retornar um mesmo resultado. Não é obrigatório a declaração nesta sintaxe, mas desta maneira o nosso código fica mais enxuto.

---

## Aula 09 – Loop For

Exercitado for loop e for loop array:

**JavaScript:**

```

//for loop
//for loop array
let texto = ''
for(let i=1; i<=50; i++){
  texto += i + '<br/>'
}
let carros = ['Ferrari', 'Fusca', 'Pálio', 'Corolla', 'Lamborghini']
let html = '<ul>'
for(let i in carros){
  html += '<li>' + carros[i] + '</li>'
}
html += '</ul>'
document.querySelector('#amostra').innerHTML = html
document.querySelector('#amostra').innerHTML += texto

```



---

## Aula 10 – Loop While

Exemplo de **while** e comparação com o **for**

```
let html = ''
let c=1
while(c<=10){
  html += "Número"+c+"<br>"
  c++
}
for(let c=1; c<=10; c++){
  html += 'Número'+c+'<br>'
}
document.querySelector('#amostra').innerHTML = html
```

---

## Aula 11 – querySelector e querySelectorAll

Obs. Interessante:

**querySelectorAll** → retorna todos os itens em um array.