JavaScript

Programa do curso

- Aula 1: Introdução e sintaxe
- Aula 2: Funções, arrays e objetos
- Aula 3: JavaScript integrado a HTML
- Aula 4: DOM, seletores e elementos
- **Aula 5: Eventos**
- Aula 6: Formulários
- Aula 7: Ajax
- Aula 8: Exercício integrador

1. Eventos

DOM - Eventos

Um evento é uma coisa que acontece no navegador ou algo que o usuário faz.

Alguns exemplos:

- A página terminou de carregar
- A entrada de um formulário foi alterada
- Um botão foi clicado

O JavaScript permite agir quando esses eventos acontecem.

DOM - Eventos

Existem duas formas de registrar um evento:

1) A primeira é estabelecendo uma propriedade no objeto ou document

```
elemento.onNomeDoEvento = function () {
    console.log('Clicou no botao')
}
```

As mais usadas são:

- onclick
- onchange
- onmouseover
- onmouseout
- onkeydown
- onload

DOM - Eventos

2) A segunda é utilizando **addEventListener**.

```
objeto.addEventListener(tipoDeEvento, funcaoGerenciadora);
```

tipoDeEvento: é uma string com o nome do tipo de evento

funcaoGerenciadora: é uma função invocada quando o evento acontece.

Por exemplo: Para que serve isto?

```
document.addEventListener("click", function(){
        alert("Ai! Você clicou em mim!");
});
```

Eventos - this

Podemos utilizar a palavra reservada **this**, que neste contexto faz referência ao objeto que executou o evento.

```
function minhaFuncao() {
    console.log(this)  // this é o elemento que executou o evento
}
elemento.addEventListener('click', minhaFuncao);
```

Eventos - removeEventListener

Para remover um *addEventListener* inserido, utilizamos:

```
objeto.removeEventListener(tipoDeEvento, funcaoGerenciadora);
```

- tipoDeEvento: é uma string com o nome do tipo de evento
- funcaoGerenciadora: é uma função invocada quando o evento acontece.

Onclick VS EventListener

A principal diferença entre onclick e EventListener é que addEventListener permite que um elemento seja escutado por muitos canais, ou seja, muitos listeners.

Por sua vez, o onclick tem apenas um evento relacionado

Eventos - preventDefault()

Algumas vezes queremos pausar a execução padrão de um determinado elemento, por exemplo um <form>.

Para evitar a execução de um evento utilizamos:

event.preventDefault()

```
<form id="form" action=”/salvarDados” method=”post”>
  <input id="inputAtor" type=”text” name=”ator” />
  <button type=”submit”>Salvar</button>
</form>

<script>
  const inputAtor = document.querySelector("#inputAtor")
  document.querySelector("#form").onsubmit = function (event) {
    event.preventDefault();

    if (!inputAtor.value) {
      alert('Preencha o campo corretamente!')
    } else {
      form.submit()
    }
  }
</script>
```

Eventos - Mouse

O objeto **event** associado ao mouse tem atributos que permitem saber a posição dele com **clientX** e **clientY**.

```
elemento.addEventListener('click', function(event) {
    event.clientX;
    event.clientY;
});
```

Eventos - Teclado

Também é possível controlar os eventos disparados quando as teclas são pressionadas, utilizando os eventos keypress, keydown e keyup.

```
elemento.addEventListener('keypress', function(event) {
        var x = event.keyCode;
        if (x == 27) { // 27 é o escape
                alert(“Você apertou escape!!”);
        }
});
```

Vamos praticar!

Prática 5 - **Eventos**

2. Timers

Timers - setTimeout

O JavaScript tem funções nativas que permitem atrasar a execução de qualquer código.

A função **setTimeout** é utilizada quando queremos que o código seja executado uma vez depois de um tempo estabelecido.

```
setTimeout(funcao, atraso);
```

Exibe um alert depois de **3** segundos (3000 milissegundos):

```
setTimeout(function(){ alert("Hello"); }, 3000);
```

Timers - setInterval

Por meio dessa função, podemos executar o mesmo código várias vezes em um intervalo regular.

```
setInterval(função, atraso);
```

A cada 3 segundos, aparece o alerta:

```
setInterval(function(){ alert("Hello"); }, 3000);
```

Timers - clearTimeout / clearInterval

Para interromper um *timeout,* utilizamos:

```
var myVar = setTimeout(function(){ alert("Hello"); }, 3000);
clearTimeout(myVar);
```

Para interromper um *interval,* utilizamos:

```
var myVar = setInterval(function(){ alert("Hello"); }, 3000);
clearInterval(myVar);
```

Vamos praticar!

Prática 5 - **Timers**

Obrigado!

Perguntas?