Aula Prática 2 – JavaScript

Professor: Luiz Carlos Felix Carvalho

Eventos

Continuamos com o assunto 'eventos'. Recordando, na última aula, vimos como adicionamos um evento em um elemento HTML. No primeiro exemplo, vimos como adicionar um comportamento a um elemento select:

```
<label for="selectTime">Quantos mundiais tem o: </label>
<select id="selectTime">
    <option value="">--Selecione--</option>
    <option value="corinthians">Corinthians</option>
    <option value="palmeiras">Palmeiras</option>
</select>

<p id="paragrafo"></p>
```

```
<script>
    let select = document.getElementById( "selectTime" );
    let paragrafo = document.getElementById( "paragrafo" );

    select.addEventListener( "change", changeValue );

    function changeValue() {

        var time = select.value;

        if ( time === "corinthians") {

            paragrafo.textContent =
            "O Corinthians tem DOIS mundiais!!!";
        }
        else if ( time === "palmeiras") {

            paragrafo.textContent =
            "O Palmeiras NÃO tem mundial!!!";
        }
    }
</script>
```

No segundo exemplo, adicionamos um comportamento em um botão:

```
<button>Change color</button>
```

```
var btn = document.querySelector("button");

function random(number) {
  return Math.floor(Math.random() * (number + 1));
}

btn.onclick = function () {
  var rndCol =
    "rgb(" + random(255) + "," + random(255) + "," + random(255) + ")";
  document.body.style.backgroundColor = rndCol;
};
```

Vimos, também, como trabalhar adição de comportamento a um elemento HTML, utilizando um código JavaScript inline:

```
function changeValue2() {
    let select = document.getElementById( "selectTime2" );
    let paragrafo = document.getElementById( "paragrafo2" );

   var time = select.value;

   if ( time === "corinthians") {

      paragrafo.textContent =
      "O Corinthians tem DOIS mundiais!!!";
   }
   else if ( time === "palmeiras") {

      paragrafo.textContent =
      "O Palmeiras NÃO tem mundial!!!";
   }
}
```

```
    <label for="selectTime2">Quantos mundiais tem o: </label>
    <select id="selectTime2" onchange="changeValue2()">
        <option value="">--Selecione--</option>
        <option value="corinthians">Corinthians</option>
        <option value="palmeiras">Palmeiras</option>
    </select>

    <p id="paragrafo2"></p>
```

Vamos, então, entender mais sobre eventos:

Segundo o site Mozilla, "**eventos** são ações ou ocorrências que acontecem no sistema que estamos desenvolvendo — o sistema irá disparar algum tipo de sinal quando o evento acontecer, além de prover um mecanismo pelo qual alguma ação automática possa ser executada (ou seja, rodar algum código) quando o evento ocorrer".

São diversos os eventos que podemos observar em uma página HTML. Para se ter uma referência, no fim desse documento, são listados os eventos que são padrão no HTML 5. Porém, os mais utilizados são: onclick, onchance, onkeypress, onfocus, onblur, onmouseover, onmouseout, onselect, onsubmit.

Nos exemplos apresentados, note que foram apresentadas duas formas de adicionar um evento a um elemento: utilizando a propriedade em que se quer adicionar o evento (btn.onClick) e utilizando o método addEventListener (select.addEventListener).

Da mesma forma que adicionamos um evento, podemos retirá-lo, por exemplo, utilizando o método removeEventListener.

```
select.removeEventListener( "change", changeValue );
```

Exercício 1:
Vimos duas formas distintas de adicionar eventos, utilizando a propriedade e utilizando o método addEventListener. Vamos, então, realizar uma simples comparação para entendermos uma diferença entre eles. Implemente o seguinte código (um arquivo HTML com um JavaScript):

```
<html>
    <head>

    </head>
    <body>
        <label for="selectTime">Quantos mundiais tem o: </label>
        <select id="selectTime">
```

```
            <option value="">--Selecione--</option>
            <option value="corinthians">Corinthians</option>
            <option value="palmeiras">Palmeiras</option>
        </select>

        <p id="paragrafo"></p>

        <p id="paragrafo2"></p>
    </body>
    <script>
        let select = document.getElementById( "selectTime" );
        let paragrafo = document.getElementById( "paragrafo" );
        let paragrafo2 = document.getElementById( "paragrafo2" );

        select.addEventListener( "change", changeValue );

        function changeValue() {

            var time = select.value;

            if ( time === "corinthians") {

                paragrafo.textContent =
                "O Corinthians tem DOIS mundiais!!!";
            }
            else if ( time === "palmeiras") {

                paragrafo.textContent =
                "O Palmeiras NÃO tem mundial!!!";
            }
        }

    </script>
</html>
```

Execute o código em um navegador. Agora, vamos alterar o código para adicionar o evento utilizando a propriedade do elemento. Troque a linha de código select.addEventListener por:

```
      //select.addEventListener( "change", changeValue );
      select.onchange = changeValue;
```

Execute novamente o código em um navegador. Veja que não houve diferença no comportamento.

Vamos criar uma nova função de comportamento para essa seleção:

```
        function changeValue2() {
```

```
        var time = select.value;

        if ( time === "corinthians") {

            paragrafo2.textContent =
            "Você sabia que o Palmeiras NÃO tem mundial?";
        }
        else if ( time === "palmeiras") {

            paragrafo2.textContent =
            "Você sabia que o Corinthians tem DOIS mundiais?";
        }
    }
```

Vamos, então, adicionar o comportamento dessa função para ser executado, também, no evento onselect:

```
//select.addEventListener( "change", changeValue );
select.onchange = changeValue;
select.onchange = changeValue2;
```

Nossa intenção é que os dois métodos sejam chamados: changeValue e changeValue2. Execute o código em um navegador e veja o resultado.

Quando adicionamos o segundo comportamento, sobrescrevemos o primeiro.

Vamos trocar por addEventListener a forma de adicionar um evento ao elemento:

```
select.addEventListener( "change", changeValue );
select.addEventListener( "change", changeValue2 );
//select.onchange = changeValue;
//select.onchange = changeValue2;
```

Execute o código em um navegador e veja o resultado. Com addEventListener, ambos os comportamentos (os dois métodos) são adicionados ao mesmo elemento.

Evento como parâmetro

Quando um evento é disparado e um comportamento é invocado para ser executado, há a possibilidade de se receber, na função de comportamento o 'objeto evento'. Tal objeto é sempre passado como parâmetro, se a função o declara como parâmetro, por exemplo:

```
function changeValue( event ) {

    //(…)
}
```

Tal objeto pode ser útil, principalmente, para acessar o elemento que originou o evento e para impedir que um comportamento padrão seja executado. A propriedade do objeto de evento que retorna o elemento que originou a ação é 'target' (event.target). O método para evitar que um comportamento padrão seja executado é preventDefault() (event.preventDefault()).

A utilização de **preventDefault()** pode ser considerada, por exemplo, quando quiser impedir que o submit de um formulário aconteça. Quando um formulário é criado em HTML, a ação de submit é sempre realizada caso disparada. Uma forma de evitar é utilizando **event.preventDefault()**, que cancela tal ação.

Exercício 2

Vamos alterar o exemplo relacionado com o select, para obtermos a opção selecionada com auxílio do objeto de evento. Altere o método changeValue para receber o objeto evento e depois acesse o valor através da propriedade value de target do objeto de evento:

```
function changeValue( event ) {

    //var time = select.value;
    var time = event.target.value;

    // (…)
```

Exercícios (mais...)

3) Crie um arquivo HTML com um formulário. Nesse formulário vamos criar os seguintes campos:

a. Estado: um select com as opções de estados disponíveis (utilize o exemplo como base para criar um select com algumas opções de estados adicionadas);
b. Cidade: outro select com as opções de cidades disponíveis, segundo a seleção realizada do estado (ou seja, ao selecionar o estado, este campo será preenchido com as opções de cidades daquele estado).

Depois de criar o arquivo HTML com os campos select's, crie um arquivo JavaScript que adicione um comportamento no select do item a, de forma que ao selecionar o estado, opções de cidades sejam adicionadas no campo select do item b.

Orientações:

Para criar um elemento HTML, você deve utilizar **document.createElement('<tagHTML>')**. Para adicionar um texto a um elemento, utilize a propriedade **textContent** de tal elemento. Para adicionar um elemento HTML a outro você deve utilizar o método **appendChild**.

Assim, temos que para adicionar uma opção a um select devemos:

```
var op = document.createElement( "option" );
op.textContent = "Uberlândia";
op.value = 'ub';
selectCid.appendChild( op );
```

Note que, a cada seleção de estado você deve limpar o select de cidades e preenchê-lo novamente. Uma das formas de se remover uma opção de um select é:

```
select.remove( i );
```

Onde i é o índice do elemento option no select. Para acessar a quantidade de opções de um select, utilize:

```
select.options.length
```

Assim, para remover as opções de um select, podemos fazer um loop iterando a quantidade de opções do select, começando do último elemento para o primeiro, sendo que a cada iteração se remove o elemento de índice da iteração.

4) Outra forma de se adicionar uma opção em um select é através do objeto Option:

```
var option = new Option( '<texto da opção>', '<value>' );
select.add(option);
```

Altere o exercício 1 para utilizar essa forma de adicionar uma opção a um select.

5) Altere a implementação do exercício 2, para criar funções dentro de seu código, de modo a melhor organizar a implementação. Deve-se criar, pelo menos mais duas funções e invocá-las dentro da implementação. Por exemplo: uma função que adicione uma opção a um select; e, outra função para eliminar as opções de um select.

Eventos HTML 5

Atributo	Valor	Descrição
off	script	Aciona quando o documento fica off-line
onabort	script	Triggers em um evento de aborto
onafterprint	script	Acionado depois que o documento é impresso

onbeforeonload	script	Triggers antes do carregamento do documento
onbeforeprint	script	Dispara antes que o documento seja impresso
onblur	script	Aciona quando a janela perde o foco
oncanplay	script	Aciona quando a mídia pode iniciar a reprodução, mas pode ter que parar para armazenar em buffer
oncanplaythrough	script	Aciona quando a mídia pode ser reproduzida até o final, sem parar para armazenar em buffer
onchange	script	Aciona quando um elemento é alterado
onclick	script	Gatilhos em um clique do mouse
oncontextmenu	script	Aciona quando um menu de contexto é acionado
ondblclick	script	Gatilhos em um mouse, clique duas vezes
ondrag	script	Aciona quando um elemento é arrastado
ondragend	script	Gatilhos no final de uma operação de arrastar
ondragenter	script	Aciona quando um elemento foi arrastado para um destino de soltar válido
ondragleave	script	Aciona quando um elemento está sendo arrastado sobre um destino de soltar válido
ondragover	script	Gatilhos no início de uma operação de arrastar
ondragstart	script	Gatilhos no início de uma operação de arrastar

ondrop	script	Aciona quando o elemento arrastado está sendo solto
ondurationchange	script	Aciona quando o comprimento da mídia é alterado
onemptied	script	Aciona quando um elemento de recurso de mídia fica vazio de repente.
onended	script	Aciona quando a mídia chega ao final
onerror	script	Acionadores quando ocorre um erro
onfocus	script	Aciona quando a janela recebe foco
onformchange	script	Aciona quando um formulário é alterado
onforminput	script	Aciona quando um formulário recebe entrada do usuário
onhaschange	script	Aciona quando o documento foi alterado
input	script	Aciona quando um elemento recebe entrada do usuário
oninvalid	script	Aciona quando um elemento é inválido
onkeydown	script	Aciona quando uma tecla é pressionada
onkeypress	script	Aciona quando uma tecla é pressionada e liberada
onkeyup	script	Aciona quando uma chave é liberada
loading	script	Acionadores quando o documento é carregado
onloadeddata	script	Aciona quando os dados de mídia são carregados

onloadedmetadata	script	Aciona quando a duração e outros dados de mídia de um elemento de mídia são carregados
onloadstart	script	Aciona quando o navegador começa a carregar os dados de mídia
onmessage	script	Aciona quando a mensagem é acionada
onmousedown	script	Aciona quando um botão do mouse é pressionado
onmousemove	script	Aciona quando o ponteiro do mouse se move
onmouseout	script	Aciona quando o ponteiro do mouse sai de um elemento
onmouseover	script	Aciona quando o ponteiro do mouse se move sobre um elemento
onmouseup	script	Acionadores quando um botão do mouse é liberado
onmousewheel	script	Aciona quando a roda do mouse está sendo girada
onoffline	script	Aciona quando o documento fica off-line
onoine	script	Aciona quando o documento fica on-line
ononline	script	Aciona quando o documento fica on-line
onpagehide	script	Aciona quando a janela está oculta
onpageshow	script	Aciona quando a janela se torna visível
onpause	script	Aciona quando os dados de mídia são pausados

onplay	script	Aciona quando os dados de mídia começarão a ser reproduzidos
playing	script	Aciona quando os dados de mídia começam a ser reproduzidos
onpopstate	script	Aciona quando o histórico da janela é alterado
progress	script	Aciona quando o navegador está buscando os dados da mídia
onratechange	script	Aciona quando a taxa de reprodução dos dados de mídia foi alterada
onreadystatechange	script	Aciona quando as alterações do estado pronto
onredo	script	Aciona quando o documento realiza um refazer
onresize	script	Aciona quando a janela é redimensionada
onscroll	script	Aciona quando a barra de rolagem de um elemento está sendo rolada
onseeked	script	Aciona quando o atributo em busca de um elemento de mídia não é mais verdadeiro e a busca terminou
onsearch	script	Aciona quando o atributo de busca de um elemento de mídia é verdadeiro e a busca foi iniciada
onselect	script	Aciona quando um elemento é selecionado
onstalled	script	Aciona quando há um erro ao buscar dados de mídia

onstorage	script	Acionadores quando um documento é carregado
onsubmit	script	Aciona quando um formulário é enviado
onsuspend	script	Aciona quando o navegador estiver buscando dados de mídia, mas parou antes que todo o arquivo de mídia fosse buscado
ontimeupdate	roteiro	Aciona quando a mídia muda sua posição de reprodução
onundo	script	Aciona quando um documento executa um desfazer
onunload	script	Aciona quando o usuário sai do documento
onvolumechange	script	Aciona quando a mídia altera o volume, também quando o volume está definido como “mudo”
waiting	script	Aciona quando a mídia parou de ser reproduzida, mas espera-se que seja retomada

Exercícios: