Eventos em JavaScript

> Os eventos são ações realizadas em um elemento desejado, eles te alertam sobre essas ações para que possamos responder de alguma forma. Um exemplo usado na maioria das aplicações é o evento de clique, onde você pode definir alguma função para o clique ou não.

O Javascript nos possibilita o uso de diversos eventos, eles são:

- onBlur;
- onChange;
- onClick;
- onFocus;
- onKeyPress;
- onLoad;
- onMouseOver;
- onMouseOut;
- onSubmit;
- onInput;

Antes de vermos os exemplos devemos saber alguns detalhes:

- O método addEventListener() é responsável por adicionar alguma função à um elemento ou algo do tipo. É usado para adicionarmos eventos dinamicamente via Javascript.
  Sua sintaxe:

  ```
  addEventListener('eventoDesejado', funcaoDesejada);
  ```

- Apesar dos eventos possuírem o prefixo 'on', no método acima devemos inseri-los sem ele. Por exemplo:

  ```
  // onClick:
  addEventListener('click', funcaoDesejada);

  // onload:
  addEventListener('load', funcaoDesejada);

  // onSubmit:
  addEventListener('submit', funcaoDesejada);
  ```

- Podemos chamar com o prefixo 'on' pelo Javascript também, porém não utilizando o addEventListener:

  ```
  // onClick:
  variavel.onclick = funcaoQualquer;


  // Exemplo:
  const btn = document.querySelector('button');

  btn.onclick = funcaoQualquer;
  ```

- Caso chame o evento diretamente no HTML temos que utilizar o prefixo, além de colocarmos parênteses '()' no final do nome da função desejada. Por exemplo:

  ```
  <!-- onClick: -->
  <button onclick="funcaoQualquer()">
  ```

```
  Clique aqui
</button>

<!-- onFocus: -->
<input type="text" name="nomeDoInput" onfocus="funcaoQualquer()">
```

---

onBlur →

Esse evento é chamado ao retirarmos o foco de determinado elemento. Podemos usar um input como exemplo, ao ser clicado o foco da página será atribuído a ele e, assim que o usuário clicar em outro local da aplicação seu foco será retirado.

Por exemplo:

```
<input type="text" name="nomeInput">
```

```
let input = document.querySelector('input');

input.onblur = perdeuOFoco;

function perdeuOFoco() {
  input.value = 'Foco removido';
};

// Podemos adicionar o evento de qualquer modo
// (diretamente no HTML, addEventListener ou .evento)
```

---

onChange →

É chamada quando o usuário confirma uma mudança de valor, clicando fora do controle em questão ou apertando a tecla `Tab`(para alterar de controle).

Por exemplo:

```
<input type="text" placeholder="Insira seu nome e veja-o logo abaixo." size="28">

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

```
let input = document.querySelector('input');
let paragrafo = document.getElementById('paragrafo');

input.onchange = confirmaMudanca;

function confirmaMudanca() {
  paragrafo.innerText = `O nome inserido é ${input.value}.`;
}
```

---

onClick →

Um dos eventos mais usados, é responsável por chamar determinada função ao ser clicado. É comum vermos esse evento em envios, confirmações e até mesmo interação com o usuário.

Por exemplo:

```
<label for="nome">
  Preencha seu nome:
</label>

<input type="text" name="nomeInput" id="nome">

<button onclick="enviarDados()">
  Enviar
</button>
```

```
const btn = document.querySelector('button');
let input = document.querySelector('input');
```

```
function enviarDados() {
  alert(
    `Obrigado por preencher o formulário, ${input.value}.`
  );
}

// Perceba que o evento foi inserido diretamente no HTML
```

---

onFocus →

Esse evento acontece ao colocarmos o foco da página em determinado elemento. É o inverso do evento onBlur, sendo chamado quando colocamos o foco em algo.

Por exemplo:

```
<input type="text">
```

```
let input = document.querySelector('input');

input.onfocus = recebeuFoco;

function recebeuFoco() {
  input.value = 'Foco recebido';
};
```

---

onKeyPress →

É ativado quando o usuário clica em alguma tecla do teclado. Algumas exceções como Ctrl, Shift, Esc e Alt não acionam o evento em si.

```
<input type="text" name="nomeInput">
<p></p>
```

```
let input = document.querySelector('input');
let resultado = document.querySelector('p');

input.onkeypress = renderizarTexto;

function renderizarTexto() {
  resultado.innerText = ` ${input.value}`;
}
```

---

onLoad →

É chamada assim que o carregamento da página é concluído. É frequentemente usado no elemento `body`.

Podemos usá-lo para alguma funcionalidade necessária assim que a página for totalmente carregada (imagens, vídeos, arquivos e etc), por exemplo a utilização dos famosos 'cookies'. Informando o usuário da ativação de cookies, ou pedindo sua permissão.

Por exemplo:

```
document.body.onload = alertarCookies;

function alertarCookies() {
  alert('Caro usuário, os cookies foram ativados com sucesso!');
}
```

---

onMouseOver e onMouseOut →

onMouseOver: o evento é acionado quando passamos o ponteiro do mouse por cima de um determinado elemento. Podemos compará-lo ao efeito de `hover` do CSS, que muda o estilo do elemento em questão, mas nesse caso será adicionado alguma funcionalidade.

onMouseOut: o evento é acionado ao movermos o ponteiro do mouse para "fora" do elemento em questão.

Em conjunto, conseguimos inseri-los num caso qualquer, por exemplo:

```
<p>Esse é um parágrafo qualquer</p>
```

```
let paragrafo = document.querySelector('p');

paragrafo.onmouseover = ponteiroEmCima;
paragrafo.onmouseout = ponteiroFora;

function ponteiroEmCima() {
  paragrafo.innerText = 'O mouse está nesse parágrafo';
}

function ponteiroFora() {
  paragrafo.innerText = 'O mouse não está nesse parágrafo';
}
```

---

onSubmit →

O evento é disparado ao enviarmos um formulário.

Por exemplo:

```
<form>
  <label for="alimento">Insira um alimento:</label>
  <input type="text" id="alimento" required>

  <button type="submit">Enviar</button>
</form>
<p id="sucesso" hidden>Dados enviados com sucesso!</p>
```

```
const form = document.querySelector('form');
const alimento = document.getElementById('alimento');
const mensagemSucesso = document.getElementById('sucesso');

form.onsubmit = enviarDados;

function enviarDados(event) {
  form.setAttribute('hidden', '');
  mensagemSucesso.removeAttribute('hidden');

  event.preventDefault();
}
```

1: selecionamos os elementos a serem utilizados;

2: adicionamos a função enviarDados ao evento de submit do formulário;

3: por sua vez, recebe como parâmetro um evento, responsável por não enviar o formulário de verdade, assim, impedindo o recarregamento da página (preventDefault()).

4: inserimos o atributo 'hidden' no fomulário e removemos o mesmo da mensagem de sucesso. O atributo hidden é usado quando queremos que algo seja classificado como "não relevante" para aquela situação, fazendo com que esconda o elemento em questão.

---

onInput →

Ocorre toda vez que um valor é inserido, é comum utilizarmos em tags como `input`, `textarea` e `select`.

Por exemplo:

```
<input
  type="text"
  placeholder="Insira palavras e confira a contagem dos caracteres abaixo."
  size="47">

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

```
let input = document.querySelector('input');
let contagem = document.getElementById('contagem');

input.oninput = contarCaracteres;

function contarCaracteres(e) {
  contagem.innerText = `Quantidade atual de caracteres: ${e.target.value.length}.`;
}
```

Bibliografias:

- https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Building_blocks/Events;
- https://developer.mozilla.org/en-US/docs/Web/API/GlobalEventHandlers/onfocus;
- https://developer.mozilla.org/en-US/docs/Web/API/Notification/onclick;
- https://developer.mozilla.org/en-US/docs/Web/API/GlobalEventHandlers/onchange;
- https://developer.mozilla.org/en-US/docs/Web/API/GlobalEventHandlers/onblur;
- https://developer.mozilla.org/en-US/docs/Web/API/GlobalEventHandlers/oninput;
- https://www.w3schools.com/jsref/event_onkeypress.asp;