Eventos no JS

Link: https://cherry-client-b8f.notion.site/Eventos-no-JS-253911d84e0d80b5b617f61154d1eb38?pvs=73

Um **evento** é uma **ação** ou **ocorrência** que acontece no navegador e que o JavaScript pode "ouvir" e responder.

Exemplos comuns:

- O usuário **clicou** em um botão → evento `click`
- O usuário **digitou** algo em um campo → evento `input` ou `keydown`
- A página foi **carregada** → evento `load`
- O mouse passou por cima de um elemento → evento `mouseover`

Como ouvir eventos?

Nós podemos ouvir eventos diretamente via html, utilizando o prefixo `on` e adicionando o nome do evento:

```
<button onclick="clicar()">Clique aqui </button>
```

Porém a forma mais utilizada é via JS, utilizando o `addEventListener`:

```
document.getElementById("meuBotao").addEventListener("click", function(event) {
  alert("Botão clicado!");
});
```

Onde:

- `"click"` → tipo de evento
- `function(event) {...}` → **função callback** que será executada quando o evento ocorrer
- `event` → objeto que contém **informações sobre o evento**

Conhecendo mais sobre o objeto event

Sempre que um evento acontece, o navegador cria um **objeto** `event` e o passa para a função de callback.

Esse objeto contém **informações detalhadas sobre o evento**.

Propriedades que podem ser acessadas

Para **mouse** (`click` , `mousemove` , `mouseover` ...)

- `event.clientX` , `event.clientY` → posição do mouse na janela
- `event.button` → qual botão do mouse (0 = esquerdo, 1 = meio, 2 = direito)
- `event.altKey` , `event.ctrlKey` , `event.shiftKey` → se uma tecla modificadora estava pressionada

Para **teclado** (`keydown` , `keyup`)

- `event.key` → a tecla pressionada (`"a"` , `"Enter"` , `"Escape"`)
- `event.code` → código físico da tecla (`"KeyA"` , `"Enter"`)
- `event.ctrlKey` , `event.shiftKey` → se o usuário estava pressionando essas teclas junto

Para **formulários** (`submit` , `input` , `change`)

- `event.target.value` → valor do campo que disparou o evento
- `event.preventDefault()` → impede o comportamento padrão (ex: envio de formulário)

Para **janela/página** (`load` , `resize` , `scroll`)

- `event.target` → geralmente é o `window` ou `document`

- `event.type` → qual evento disparou

Delegação de eventos

A **Delegação de eventos** é uma técnica em JavaScript para lidar com eventos de muitos elementos filhos usando apenas um listener no elemento pai.

Imagine que você tem **100 botões** numa lista.

Se você adicionar um `addEventListener` em cada um, terá 100 listeners na memória. Isso é:

- Ineficiente
- Difícil de manter quando os elementos são criados dinamicamente

A ideia da **delegação**

Em vez de escutar cada botão, você coloca **um único listener no elemento pai** (ex: a `<ul>` ou `<div>` que contém os botões).

Quando o usuário clica, o evento "bolha" (propaga para cima) até o pai, e lá você verifica **quem foi o filho que disparou**.

```
document.getElementById("lista").addEventListener("click", function(event) {
  if (event.target.classList.contains("botao")) {
    console.log("Botão clicado:", event.target.textContent);
  }
});
```

Como funciona tecnicamente

1. O evento dispara no **elemento alvo** (ex: o botão).
2. Ele sobe na árvore do DOM (fase de *bubbling*).
3. O listener no pai recebe o evento.
4. Dentro do listener, usamos `event.target` para saber **qual filho exato** gerou o evento.

Aplicações

É possível verificar se um elemento corresponde à um seletor CSS com `.matches`

```
document.querySelector(".lista").addEventListener("click", function(event) {
  if (event.target.matches("li.item")) {
    console.log("Clicou em um item:", event.target.textContent);
  }
});
```

Também é possível subir na hierarquia do DOM até encontrar o ancestral que corresponda ao seletor.
Útil quando você clica em algo dentro do alvo real (ex: botão dentro de um card).

```
document.querySelector(".cards").addEventListener("click", function(event) {
  const card = event.target.closest(".card");
  if (card) {
    console.log("Clicou em um card:", card.dataset.id);
  }
});
```

Referências

https://dmitripavlutin.com/javascript-event-delegation/

https://www.freecodecamp.org/news/event-delegation-javascript/

https://www.w3schools.com/jsref/dom_obj_event_prop.asp

https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/Events