12

O que é má prática ontem pode ser boa prática hoje e vice-versa!

No mundo front-end, há uma separação clara entre HTML, CSS, JS. Tanto isso é verdade que a boa prática é que cada um fique no seu quadrado, ou seja, que cada um tenha seu arquivo dedicado. Essa separação visa facilitar a manutenção.

Contudo, quando manipulamos o DOM, vira e mexe precisamos associar eventos a elementos. Uma maneira de se fazer isso é encontrar o elemento que queremos associar o evento, e realizar essa associação via JavaScript. Vejamos um exemplo:

```
<p id="p1">Olá</p>
```

Temos um parágrafo com um `id` definido. No mundo JavaScript, se quisermos associar um evento `click` ao parágrafo, precisamos buscá-lo primeiro e depois associar o evento:

```
function mostra() {
    alert('Fui clicado');
}

document.querySelector('#p1').addEventListener('click', mostra);
```

O código anterior funciona perfeitamente, inclusive deixou marcante a separação entre HTML e JS, pois em nenhum momento no HTML referenciamos nosso JS (apenas a tag `<script>` que o carrega, claro).

Contudo, essa solução nos obriga a manipular o DOM toda vez que quisermos associar um evento com o elemento. Sendo assim, quando criamos **SPA** (**S**ingle **P**age **A**pplications), páginas que não se recarregam durante seu uso é muito comum usar a abordagem clássica, que é associar a função do evento diretamente na tag `<html>` da nossa página.

Qual das opções abaixo associa diretamente na tag `<p>` o evento `click` para a função `mostra` ?

A

```
<p id="p1" click="mostra()">Olá</p>
```

B

```
<p id="p1" click="mostra">Olá</p>
```

✓

```
<p id="p1" onclick="mostra()">Olá</p>
```

Dependendo do seu viés teórico, essa solução pode ser uma "heresia". Contudo, frameworks SPA, como Angular, adotam estrutura semelhante para associar a ação de um controller a um componente da página, dessa forma, removendo o desenvolvedor de ter que realizar essa associação manualmente.

PRÓXIMA ATIVIDADE