**Para saber mais: addEventListener('click') vs onclick**

Em nossa trajetória pelo mundo dos eventos em JavaScript, um assunto intrigante merece nossa atenção: a diferença entre usar elemento.addEventListener('click', mostrarAlerta) e elemento.onclick = mostrarAlerta.

Essas abordagens podem parecer semelhantes, mas têm particularidades importantes.

**Vamos pegar um botão como exemplo:**

**const btnExemplo = document.querySelector('#btnExemplo');**

Agora vamos analisar qual a diferença pra nós, devs frontend, e pro navegador. Vem comigo.

## Diferença para o navegador

**addEventListener:** Com ele, você pode adicionar mais de um ouvinte para o mesmo evento. Imagine que você quer que o botão exiba um alerta e também mude de cor quando clicado. Confira:

**function mostrarAlerta() {**

**alert('Botão clicado!');**

**}**

**function mudarCor() {**

**btnExemplo.style.backgroundColor = 'red';**

**}**

**btnExemplo.addEventListener('click', mostrarAlerta);**

**btnExemplo.addEventListener('click', mudarCor);**

**onclick:** Se você tentar definir mais de uma ação usando onclick, a última definida é a que permanecerá, sobrescrevendo as anteriores. Veja o risco:

**btnExemplo.onclick = function() {**

**alert('Botão clicado!');**

**}**

**btnExemplo.onclick = function() {**

**btnExemplo.style.backgroundColor = 'red';**

**}**

Neste caso, apenas a função que muda a cor de fundo será executada. O alerta foi sobrescrito!

## Diferença para o desenvolvedor

* Flexibilidade com addEventListener: Com ele, você tem mais controle e liberdade para adicionar diversas reações a um mesmo evento, ideal para scripts ou módulos distintos.
* Controle simplificado com onclick: Pode parecer mais direto, mas o risco é grande em aplicações robustas! Ao sobrescrever funções inadvertidamente, você pode acabar com comportamentos inesperados.

Reflexão: Em um projeto grande, com vários desenvolvedores trabalhando e talvez até com bibliotecas externas, como garantir que não estamos sobrescrevendo um evento crucial usando **onclick**?

Essa exploração nos ajuda a escrever códigos mais conscientes e eficazes. A **curiosidade** é uma das maiores ferramentas da pessoa desenvolvedora.