Event Listener

Explicação dos eventos de mouse e teclado em JavaScript

O QUE SÃO EVENTOS DE MOUSE E TECLADO EM JAVASCRIPT ?

Os eventos são basicamente um conjunto de ações que são realizadas em uma determinada página web, seja ele, um clique, pressionar teclas, ou interagir com os elementos, seja ele um texto, imagem, ou uma div, por exemplo. Todas as interações do usuário com o conteúdo da página pode ser considerada um evento.

O QUE É UM EVENT LISTENER E COMO ADICIONÁ-LO VIA JS?

Definição

O método JavaScript addEventListener () permite configurar funções a serem chamadas quando um evento especificado acontece, como quando um usuário clica em um botão.

Sintaxe

```
target.addEventListener(event, listener, useCapture)
```

target: Elemento DOM que desejamos aplicar o evento

event: Nome do evento a ser aplicado

listener: Geralmente, a função a ser executada quando o evento ocorrer

useCapture: Pode omitir na declaração, é um boolean de valor padrão false; Se true, useCapture indica que o usuário deseja iniciar uma captura. Depois de iniciada a captura, todos os eventos do tipo especificado serão enviados à listener (que pode ser um objeto)

PRINCIPAIS EVENTOS

blur	**O evento ocorre quando um elemento perde o foco**
change	**O evento ocorre quando o conteúdo de um elemento de formulário, a seleção ou o estado verificado são alterados (para <input>, <select> e <textarea>**
click	**O evento ocorre quando o usuário clica em um elemento**
focus	**O evento ocorre quando um elemento recebe o foco**
keyPress	**O evento ocorre quando o usuário pressiona uma tecla**
load	**O evento ocorre quando um objeto é carregado**
mouseOver	O evento ocorre quando o ponteiro é movido para um elemento ou para um de seus filhos
mouseOut	O evento ocorre quando um usuário move o ponteiro do mouse para fora de um elemento ou de um de seus filhos
onSubmit	O evento ocorre quando um formulário é enviado

O QUE É UMA FUNÇÃO DE CALLBACK (UTILIZADA NO EVENT LISTENER)?

nº 1

Basicamente, callback é um tipo de função que só é executada após o processamento de outra função ou quando é invocada em resposta a ocorrência de um evento.. Na linguagem JavaScript, quando uma função é passada como um argumento de outra, ela é, então, chamada de callback.

nº 2

Nesse sentido, callback no event listenet é aquela que passamos como parâmetro e ela somente é executada quando aquele evento ocorrer.

QUAL A DIFERENÇA DE UM EVENT LISTENER DE ATRIBUTOS HTML (COMO ONCLICK)?

A principal diferença é que com os atributos HTML, como o onclick, você consegue cadastrar apenas uma função como callback do evento. Já com o addEventListener() é possível utilizar quantas funções quiser que todas elas serão ativadas quando o evento ocorrer. Justamente por isso o addEventListener() é mais indicado.

O QUE É O OBJETO EVENT QUE É RECEBIDO NA FUNÇÃO DE CALLBACK?

Event.bubbles	Um booleano que indica se o evento surge em bolha pela DOM ou não.
Event.isTrusted	Indica se ou não o evento foi iniciado pelo navegador (depois de um clique do usuário, por exemplo) ou por um script (usando um método de criação de evento, como event.initEvent)
Event.initEvent()	Inicializa o valor de um evento criado. Se o evento já está sendo despachado, este método não faz nada.
Event.stopPropagation()	Para a propagação de eventos mais adiante no DOM.

O QUE É EVENT BUBBLING EM JS?

O Event bubbling ocorre quando um usuário interage com um elemento no HTML e o evento se propaga como “bolhas” por todos os elementos que estão aninhados a ele.
Para explicar esse processo, imagine um copo com refrigerante aonde as bolhas borbulham do fundo do copo até o ponto mais alto. O princípio do efeito bubbling no JavaScript é exatamente o mesmo. Quando um evento “bubbling” é realizado sobre um determinado elemento, todos os ancestrais dele também serão acionados em ordem crescente de aninhamento até chegar ao último elemento:

O QUE FAZ A FUNÇÃO EVENT.STOPPROPAGATION?

No exemplo anterior é possível verificar que o efeito percorre todos os elementos, sempre partindo do item selecionado até o elemento mais alto.

- event.preventDefault() impede que o evento padrão ocorra (ex.: seguir um link);
- event.stopPropagation() impede que o evento seja propagado para os handlers dos elementos DOM pais;
- return false faz as duas coisas (e ainda interrompe a execução do handler imediatamente, sem executar as instruções que vêm depois).