Sự kiện trong Dom - JavaScript

Các sự kiện trong JavaScript, bắt sự kiện trong DOM, lắng nghe sự kiện như khi click, change, tạo ra sự kiện và kích hoạt một sự kiện

Bắt sự kiện trên phần tử HTML

Bạn có thể viết mã JavaScript thi hành khi một sự kiện nào đó xảy ra, ví dụ khi người dùng bấm vào một phần tử cụ thể, khi di chuyển chuột trên phần tử, khi submit form ...

Các sự kiện này coi như thuộc tính của phần tử, thuộc tính sẽ gán tên hàm được thi hành.

Ví dụ sự kiện khi bấm chuột vào phần tử tương ứng với thuộc tính `onclick`, vậy tạo một phần tử `p` khi người dùng bấm chuột vào sẽ thi hành hàm bạn định nghĩa `myfunc` viết HTML như sau:

```
<p onclick="myfunc()">some text</p>
```

Bảng các sự kiện HTML hay dùng

Sự kiện	Mô tả
`onclick`	xảy ra khi bấm chuột vào phần tử
`onload`	xảy ra khi phần tử được tải
`onunload`	xảy ra khi trang un load (phần tử body)
`onchange`	xảy ra khi thay đổi nội dung phần tử trong `form` như khi chọn select, khi người dùng bấm radio, áp dụng cho các phần tử (input, select, textarea)
`onmouseover`	xảy ra khi chuột di chuyển trên phần tử, hoặc phần tử con của phần tử
`onmouseout`	khi chuột di chuyển ra khỏi phần tử
`onmousedown`	xảy ra khi bấm chuột trên phần tử
`onmouseup`	xảy ra khi nhả bấm chuột trên phần tử
`onblur`	xảy ra phi phần tử mất focus
`onfocus`	xảy ra khi phần tử nhận focus (phần tử đang kích hoạt nhận dữ liệu)

Ví dụ bắt sự kiện trên phần tử HTML

Sự kiện có thể được bắt bằng cách khai báo ngay từ thuộc tính trong HTML, ví dụ sau hiện thị popup khi người dùng bấm chuột vào phần tử

```
<button onclick="show()">Click Me</button>
<script>
    function show() {
        alert("Hi there");
    }
</script>
```

Click Me

Sự kiện đã được bắt bằng đoạn mã: `onclick="show()"`

Bắt sự kiện trên phần tử DOM - HTMLElement

Cách thứ 2 kết quả tương tự là bạn bắt sự kiện bằng mã JavaScript, thông qua đối tượng HTMLElement tìm được trên DOM

```
var x = document.getElementById("demo");
x.onclick = function () {
  document.body.innerHTML = Date();
}
```

Sự kiện đã được bắt bằng đoạn mã: `x.onclick = function() {}` hoặc bằng `x.onclick = functionname();`

onload, onunload

`onload` và `unload` xảy ra khi người dùng mở trang và rời trang.

```
<body onload="doSomething()">
```

Tương tự `window.onload` có thể được dụng để bắt dự kiện khi trang được tải.

```
window.onload = function() {
   //mã
}
```

onchange

`onchange` dùng phổ biến trong hộp nhập dữ liệu văn bản. Sự kiện xảy ra khi nội dung textbox thay đổi và mất focus

```
<input type="text" id="name" onchange="change()">

<script>
    function change() {
         var x = document.getElementById("name");
         x.value= x.value.toUpperCase();
    }
</script>
```

Lắng nghe sự kiện trên DOM

Phương thức `addEventListener()` sẽ gắn thêm hàm (Listener) vào một phần tử để lắng nghe sự kiện của phần tử mà không loại bỏ các hàm sự kiện đã gắn trước. Điều này giúp cho một sự kiện xảy ra có thể gọi nhiều hàm Listener gắn với sự kiện đó.

```
element.addEventListener(event, listener, useCapture);
```

Các tham số là:

- `event` tên sự kiện ví dụ `"click"`, `"mousedown"`, `"load"`, `"change"`, `"mouseover"`, `"blur"` ...
- `listener` tham số thứ 2 là hàm do bạn định nghĩa, muốn thi hành khi sự kiện xảy ra
- `useCapture` tham số thứ 3 là giá trị `true`, `false` đây là một tùy chọn sẽ giải thích sau

Chú ý sử dụng cách này tên các sự kiện sẽ bỏ đi tiền tố `on` ví dụ sử dụng `"click"` chứ không phải `onclick` như phần trên.

```
element.addEventListener("click", myFunction);
element.addEventListener("mouseover", myFunction);

function myFunction(event) {
  alert("Hello World!");
}
```

removeEventListener

Hàm dùng để loại bỏ hàm (listener) đã gắn vào sự kiện trên phần tử

```
element.removeEventListener(name_event, listener);
```

Ví dụ sau gắn một hàm (listener) vào phần tử để lắng nghe sự kiện click, khi sự kiện đó xảy ra thì loại bỏ listener đó (không lắng nghe nữa - chỉ bấm được 1 lần).

```
<button id="demo">Start</button>
```

```
<script>
    var btn = document.getElementById("demo");
    btn.addEventListener("click", myListener);

    function myListener(e) {
      alert(Math.random());
      btn.removeEventListener("click", myListener);
    }
</script>
```

Start

Kiểu lan truyền sự kiện

Ở đây giải thích tham số thứ 3 trong hàm **addEventListener**, tham số **useCapture**

Có hai kiểu lan truyền sự kiện, `bubbling` và `capturing`

Để giải thích giả sử có phần tử `div` bên trong nó chứa phần tử `p` nếu vậy khi bấm chuột vào `p` thì xảy ra sự kiện `click`. Vậy phần tử `p` hay phần tử `div` sẽ bắt được sự kiện trước?

Nếu là `capturing` nghĩa là tham số thứ 3 `useCapture` là `true` thì phần tử `div` nhận được sự kiện trước, sau đó mới đến `p`

Nếu là `bubbling` nghĩa là tham số thứ 3 `useCapture` là `false` thì phần tử `p` nhận được sự kiện trước, sau đó mới đến `div`

capturing (useCapture = true) - sự kiện đi từ trên xuống dưới của DOM
bubbling (useCapture = false mặc định) - sự kiện truyền từ dưới lên trên trong cây DOM

```
//Capturing - gốc đến ngọn
elem1.addEventListener("click", myFunction, true);

//Bubbling - từ ngọn đến gốc
elem2.addEventListener("click", myFunction, false);
```

Listener khi bắt các sự kiện

Nói kỹ hơn về các hàm để gắn vào sự kiện mà ta gọi là các Listener. Các hàm này có giao diện triển khai từ giao diện `eventListener.handleEvent(event);`

Có nghĩa là một hàm Javascript có 1 tham số, tham số đó chứa thông tin sự kiện gửi đến, mọi trường hợp bạn có định nghĩa ra các hàm Listener với cấu trúc như sau:

```
/***
 *
 * @param event Event
 */
function myListener(event) {

}
```

Khi Listener của bạn bắt được một sự kiện nào đó, nó luôn nhận được tham số chứa thông tin sự kiên, ở ví dụ trên lưu trong tham số `event`, tham số này là đối tượng kiểu Event

Event có nhiều phương thức, có một số mà bạn tham khao luôn ở đây như:

`preventDefault()`	Ngăn cản ứng xử thông thường xảy ra trên phần tử. Ví dụ bạn bấm vào link, gọi phương thực này sẽ ngăn trình duyệt chuyển đến trang chỉ ra bởi link đó.
	`function myListener(event) {`
	`  event.preventDefault();`
	`  //Các code khác ...`
	`}`

`stopPropagation()`	Dừng lan truyền sự kiện, ví dụ nếu một sự kiến có nhiều sự kiện đang lắng nghe, bạn muốn sau khi Listener của bạn bắt được thì các Lister khác không còn nhận được nữa thì gọi phương thức này.

Tạo và phát sự kiện Event

Ở ví dụ này, ta tạo ra một sự kiện có tên `eventxinchao`, sự kiện đó có chứa dữ liệu là dòng chữ `"Dữ liệu của sự kiện"`, sau đó đối tượng DOM `document` phát đi sự kiện đó.

Mặt khác ta cũng gắn vào `document` một Listener đang lắng nghe sự kiện có tên `eventxinchao`, để mỗi khi sự kiện này xảy ra thì Listener này được gọi

```
<script>
    //Tạo ra Listener ngồi nghe sự kiện eventxinchao
    document.addEventListener("eventxinchao", function (e) {
        alert(event.dulieu);
    });


    //Mỗi khi hàm này được gọi sẽ phát đi sự kiện eventxinchao
    function guiEventXinchao() {
        //Tạo ra một Event
        event = document.createEvent('Event');

        //Khởi tạo Event để thiết lập tên cho nó là eventxinchao
        event.initEvent('eventxinchao', true, true);
```

```
        event.dulieu = "Dữ liệu của sự kiện";

        //Phát sự kiện
        document.dispatchEvent(event);
    }

</script>

<button onclick="guiEventXinchao();">Gửi Event - eventxinchao</button>
```