Event

Ba Nguyễn

Introduction

Event **-** ***sự kiện*** là tín hiệu khi một điều gì đó xảy ra trên **DOM**, tất cả các ***node*** trong **DOM** đều có thể kích hoạt các ***event***

Để cung cấp một hành động phản ứng lại khi một ***event*** được kích hoạt, chúng ta sử dụng các trình xử lý sự kiện - ***event handlers***

💡 ***Event handler*** là một ***function*** được chạy khi một ***event*** được kích hoạt

Các cách để chỉ định một ***handler*** cho một ***event*** cụ thể:

- ***HTML Attribute***
- ***DOM Property***
- `addEventListener()/removeEventListener()`

HTML Attributes

Các handlers có thể gán cho một event cụ thể trên một element thông qua attributes ***on<event>***

```
<body onkeyup="greeting()">
    <p onclick="alert('Hello babe ❤️')">Click me!</p>
    <p onmouseover="let x = 1, y = 2; alert(x + y)">
        Hover me
    </p>
    <script>
      function greeting() {
        alert("Hello babe 😍");
      }
    </script>
</body>
```

DOM Properties

Tương tự attributes, có thể gán các handlers thông qua DOM properties ***on<event>***

```
p.onclick = function () {
    alert("Hello babe 💔");
};
p.onmouseover = function () { alert(1 + 2); };

function greeting() {
    alert("Hello babe 😍");
}
document.body.onkeyup = greeting;
```

Element - this

Giá trị của `this` trong handler chính là element kích hoạt event

```
<p onclick="alert(this.textContent)">
    Click me!
</p>
<!— Click me! —>
```

💡 Khi gán một handler qua ***attribute*** là gán ***function call***, gán qua ***property*** là ***funciton reference***

💡 Không thể gán một handler thông qua phương thức `setAttribute()`

EventListener

Khi gán các event, chỉ có thể có một handler duy nhất cho một event khi gán thông qua attribute hoặc property. Một số event đặc biệt thậm chí không thể gán thông qua attribute hoặc property

JavaScript cung cấp 2 phương thức đặc biệt `addEventListener()/removeEventListener()` để quản lý các event tốt hơn, không gặp phải các hạn chế giống như attribute hoặc property.

```
// Cú pháp
el.addEventListener("event", handler , options);
el.removeEventListener("event", handler, options);

// Options:
// capture: false, passive: false, once: false
```

EventListener

```
let handler = function () {
    alert(this.textContent);
};

p.addEventListener("click", handler);
p.removeEventListener("click", handler);
```

💡 Lưu ý gán handler là ***function reference***, không phải ***function call***
💡 Khi thêm options trong `addEventListener()` thì `removeEventListener()` cũng cần truyền options tương ứng

EventObject

Handlers được gọi với một tham số đặc biệt - ***event***, ***event*** chứa tất cả thông tin về sự kiện, như phần tử nào kích hoạt, phím được nhấn, ...

```
p.addEventListener("click", function(e) {
    alert(e.currentTarget); // p
    alert(e.currentTarget.textContent);
    alert(e.type); // click
});
```

💡 Đối với arrow function, `e.currentTarget` tương tự `this`

Browser Default Actions

Có rất nhiều sự kiện mặc định với các thành phần trên trang sẽ được thực hiện bởi trình duyệt, đồng thời khi một sự kiện được kích hoạt trên một phần tử, sự kiện sẽ được *lan truyền* tới các phần tử cấp cao hơn.

```
// Ngăn sự kiện mặc định
e.preventDefault();

// Ngăn sự kiện lan truyền đi
e.stopPropagation();
```