JavaScript dispatchEvent

Summary: in this tutorial, you'll learn how to programmatically create and dispatch events using `Event` constructor and `dispatchEvent()` method.

Typically, events are generated by user actions such as mouse clicks and key presses. In addition, events can be generated from code.

To generate an event programmatically, you follow these steps:

- First, create a new `Event` object using `Event` constructor.
- Then, trigger the event using `element.dispatchEvent()` method.

Event constructor

To create a new event, you use the `Event` constructor like this:

```
let event = new Event(type, [,options]);
```

The `Event` constructor accepts two parameters:

type

is a string that specifies the event type such as `'click'` .

options

is an object with two optional properties:

- `bubbles` : is a boolean value that determines if the event bubbles or not. If it is `true` then the event is bubbled and vice versa.
- `cancelable` : is also a boolean value that specifies whether the event is cancelable when it is `true` .

By default, the `options` object is:

```
{ bubbles: false, cancelable: false}
```

For example, the following creates a new `click` event with the default `options` object:

```
let clickEvent = new Event('click');
```

dispatchEvent method

After creating an event, you can fire it on a target element using the `dispatchEvent()` method like this:

```
element.dispatchEvent(event);
```

For example, the following code shows how to create the `click` event and fire it on a button:

HTML:

```
<button class="btn">Test</button>
```

JavaScript:

```
let btn = document.querySelector('.btn');

 btn.addEventListener('click', function () {
        alert('Mouse Clicked');
 });

let clickEvent = new Event('click');
btn.dispatchEvent(clickEvent);
```

Click this link to see the demo.

In this example, the event handler is executed as if the `click` event were generated by user actions.

If the event comes from the user actions, the `event.isTrusted` property is set to `true`. In case the event is generated by code, the `event.isTrusted` is `false`. Therefore, you can examine the value of `event.isTrusted` property to check the "authenticity" of the event.

The `Event` is the base type of `UIEvent` which is the base type of other specific event types such as `MouseEvent`, `TouchEvent`, `FocusEvent`, and `KeyboardEvent`.

It's a good practice to use the specialized event constructor like MouseEvent instead of using the generic `Event` type because these constructors provide more information specific to the events.

For example, the `MouseEvent` event has many other properties such as `clientX` and `clientY` that specify the horizontal and vertical coordinates at which the event occurred relative to the viewport:

```
let clickEvent = new MouseEvent("click", {
    bubbles: true,
    cancelable: true,
    clientX: 150,
    clientY: 150
});
```

The following link shows the full list of properties of the MouseEvent

Summary

- Use the specific event constructor such as `MouseEvent` and call `dispatchEvent()` method on an element to generate an event from code.
- Use `event.isTrusted` to examine whether the event is generated from code or user actions.