JavaScript Custom Events

Summary: in this tutorial, you will learn about JavaScript custom events such as creating a custom event and dispatching it.

Introduction to JavaScript custom events

The following function highlights an element by changing its background color to `yellow`:

```
function highlight(elem) {
    const bgColor = 'yellow';
    elem.style.backgroundColor = bgColor;
}
```

To execute a piece of code after highlighting the element, you may come up with a callback:

```
function highlight(elem, callback) {
    const bgColor = 'yellow';
    elem.style.backgroundColor = bgColor;

    if(callback && typeof callback === 'function') {
        callback(elem);
    }
}
```

The following calls the `highlight()` function and adds a border to a `<div>` element:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>JS Custom Event Demo</title>
</head>
<body>
    <div class="note">JS Custom Event Demo</div>
    <script>
        function highlight(elem, callback) {
            const bgColor = 'yellow';
            elem.style.backgroundColor = bgColor;

            if (callback && typeof callback === 'function') {
                callback(elem);
            }
        }

        let note = document.querySelector('.note');
        function addBorder(elem) {
            elem.style.border = "solid 1px red";
        }

        highlight(note, addBorder);
    </script>
</body>
</html>
```

To make the code more flexible, you can use the custom event.

Creating JavaScript custom events

To create a custom event, you use the `CustomEvent()` constructor:

```
let event = new CustomEvent(eventType, options);
```

The `CustomEvent()` has two parameters:

- The `eventType` is a string that represents the name of the event.
- The `options` is an object has the `detail` property that contains any custom information about the event.

The following example shows how to create a new custom event called `highlight` :

```
let event = new CustomEvent('highlight', {
    detail: {backgroundColor: 'yellow'}
});
```

Dispatching JavaScript custom events

After creating a custom event, you need to attach the event to a DOM element and trigger it by using the `dispatchEvent()` method:

```
domElement.dispatchEvent(event);
```

JavaScript custom event example

Put it all together:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript Custom Event</title>
</head>
<body>
    <div class="note">JS Custom Event</div>
    <script>
        function highlight(elem) {
            const bgColor = 'yellow';
            elem.style.backgroundColor = bgColor;

            // create the event
            let event = new CustomEvent('highlight', {
                detail: {
                    backgroundColor: bgColor
                }
            });
```

```
            // dispatch the event
            elem.dispatchEvent(event);
        }

        // Select the div element
        let div = document.querySelector('.note');

        // Add border style
        function addBorder(elem) {
            elem.style.border = "solid 1px red";
        }

        // Listen to the highlight event
        div.addEventListener('highlight', function (e) {
            addBorder(this);

            // examine the background
            console.log(e.detail);
        });

        // highlight div element
        highlight(div);
    </script>
</body>
</html>
```

How it works:

- First, declare the `highlight()` function that highlights an element and triggers the `highlight` event.
- Second, select the `<div>` element by using the `querySelector()` method.
- Third, listen to the `highlight` event. Inside the event listener, call the `addBorder()` function and show the `detail` property in the Console.
- Finally, call the `highlight()` function that will trigger the `highlight` event.

Why use custom events

Custom events allow you to decouple code execution, allowing one piece of code to run after another completes.

For example, you can place event listeners in a separate script file and have multiple listeners for the same custom event.

Summary

- Use the `CustomEvent()` constructor to create a custom event and `dispatchEvent()` to trigger the event.