JavaScript Mouse Events

Summary: in this tutorial, you will learn about the basic mouse events and their properties in JavaScript.

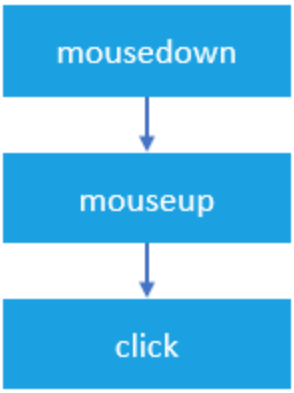
Introduction to JavaScript mouse events

Mouse events fire when you use the mouse to interact with the elements on the page. DOM Level 3 events define nine mouse events.

mousedown, mouseup, and click events

When you `click` an element, there are no less than three mouse events fire in the following sequence:

1. The `mousedown` fires when you press the mouse button on the element.
2. The `mouseup` fires when you release the mouse button on the element.
3. The `click` fires when one `mousedown` and one `mouseup` detected on the element.



If you press the mouse button on an element, move your mouse cursor off the element, and then release the mouse button. The only `mousedown` event fires on the element.

Likewise, if you press the mouse button, move the mouse over the element, and release the mouse button, the only `mouseup` event fires on the element.

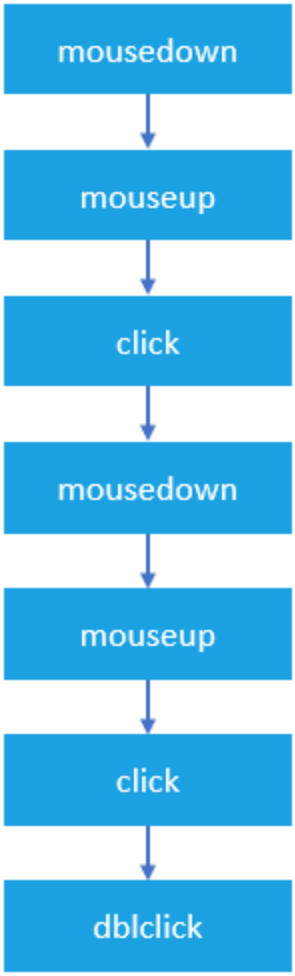
In both cases, the `click` event never fires.

dblclick event

In practice, you rarely use the `dblclick` event. The `dblclick` event fires when you double-click over an element.

It takes two click events to cause a `dblclick` event to fire. The `dblclick` event has four distinct events fired in the following order:

1. `mousedown`
2. `mouseup`
3. `click`
4. `mousedown`
5. `mouseup`
6. `click`
7. `dblclick`



The picture shows that the `click` events always occur before the `dblclick` event.

If you register both `click` and `dblclick` event handlers on the same element, you will not know whether the user has clicked or double-clicked the element.

mousemove

The `mousemove` event fires repeatedly whenever you move the mouse cursor around an element. This `mousemove` event fires many times per second as the mouse is moved around, even if it is just by one pixel. This may lead to a performance issue if the event handler function is complex.

To avoid the performance issue, it is a good practice to add `mousemove` event handler only when you need it and remove it as soon as it is no longer needed, like this:

```
element.onmousemove = mouseMoveEventHandler;
// ...
```

```
//  later, no longer use
element.onmousemove = null;
```

mouseover / mouseout

The `mouseover` fires when the mouse cursor is outside of the element and then moves inside the boundaries of the element.

The `mouseout` fires when the mouse cursor is over an element and then moves another element.

mouseenter / mouseleave

The `mouseenter` fires when the mouse cursor is outside of an element and then moves inside the boundaries of the element.

The `mouseleave` fires when the mouse cursor is over an element and then moves to the outside of the element's boundaries.

Both `mouseenter` and `mouseleave` does not bubble and does not fire when the mouse cursor moves over descendant elements.

Registering mouse event handlers

To register a mouse event, you use these steps:

- First, select the element by using `querySelector()` or `getElementById()` method.
- Then, register the mouse event using the `addEventListener()` method.

For example, suppose that you have the following button:

```
<button id="btn">Click Me!</button>
```

To register a mouse-click event handler, you use the following code:

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

btn.addEventListener('click',(event) => {
```

```
    console.log('clicked');
});
```

or you can assign a mouse event handler to the element's property:

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

btn.onclick = (event) => {
    console.log('clicked');
};
```

In legacy systems, you may find that the event handler is assigned in the HTML attribute of the element:

```
<button id="btn" onclick="console.log('clicked')">Click Me!</button>
```

It's a good practice to use the `addEventListener()` to register a mouse event handler.

Detecting mouse buttons

The `event` object passed to the mouse event handler has a property called `button` that indicates which mouse button was pressed on the mouse to trigger the event.

The mouse button is represented by a number:

- 0: the main mouse button is pressed, usually the left button.
- 1: the auxiliary button is pressed, usually the middle button or the wheel button.
- 2: the secondary button is pressed, usually the right button.
- 3: the fourth button is pressed, usually the Browser Back button.
- 4: the fifth button is pressed, usually the *Browser Forward* button.

See the following example:

```
<!DOCTYPE html>
<html>
<head>
    <title>JS Mouse Events - Button Demo</title>
</head>
<body>
    <button id="btn">Click me with any mouse button: left, right, middle, ...</button>
    <p id="message"></p>
    <script>
        let btn = document.querySelector('#btn');

        // disable context menu when right-mouse clicked
        btn.addEventListener('contextmenu', (e) => {
            e.preventDefault();
        });

        // show the mouse event message
        btn.addEventListener('mouseup', (e) => {
            let msg = document.querySelector('#message');
            switch (e.button) {
```

```
                case 0:
                    msg.textContent = 'Left mouse button clicked.';
                    break;
                case 1:
                    msg.textContent = 'Middle mouse button clicked.';
                    break;
                case 2:
                    msg.textContent = 'Right mouse button clicked.';
                    break;
                default:
                    msg.textContent = `Unknown mouse button code: ${event.button}`;
            }
        });
    </script>
</body>
</html>
```

In this example, when you click the button with your mouse (left-click, right-click, and middle-click), it shows a corresponding message on the `<div>` element.

Output

Modifier keys

When you click an element, you may press one or more modifier keys: Shift, Ctrl, Alt, and Meta.

Note the Meta key is the Windows key on Windows keyboards and the Command key on the Apple keyboard.

To detect if these modifier keys have been pressed, you can use the `event` object passed to the mouse event handler.

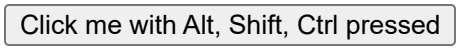
The `event` object has four Boolean properties, where each is set to `true` if the key is being held down or `false` if the key is not pressed.

See the following example:

```
<!DOCTYPE html>
<html>
<head>
    <title>JS Modifier Keys Demo</title>
</head>
<body>
    <button id="btnKeys">Click me with Alt, Shift, Ctrl pressed</button>
    <p id="messageKeys"></p>

    <script>
        let btnKeys = document.querySelector('#btnKeys');

        btnKeys.addEventListener('click', (e) => {
            let keys = [];

            if (e.shiftKey) keys.push('shift');
            if (e.ctrlKey) keys.push('ctrl');
            if (e.altKey) keys.push('alt');
```

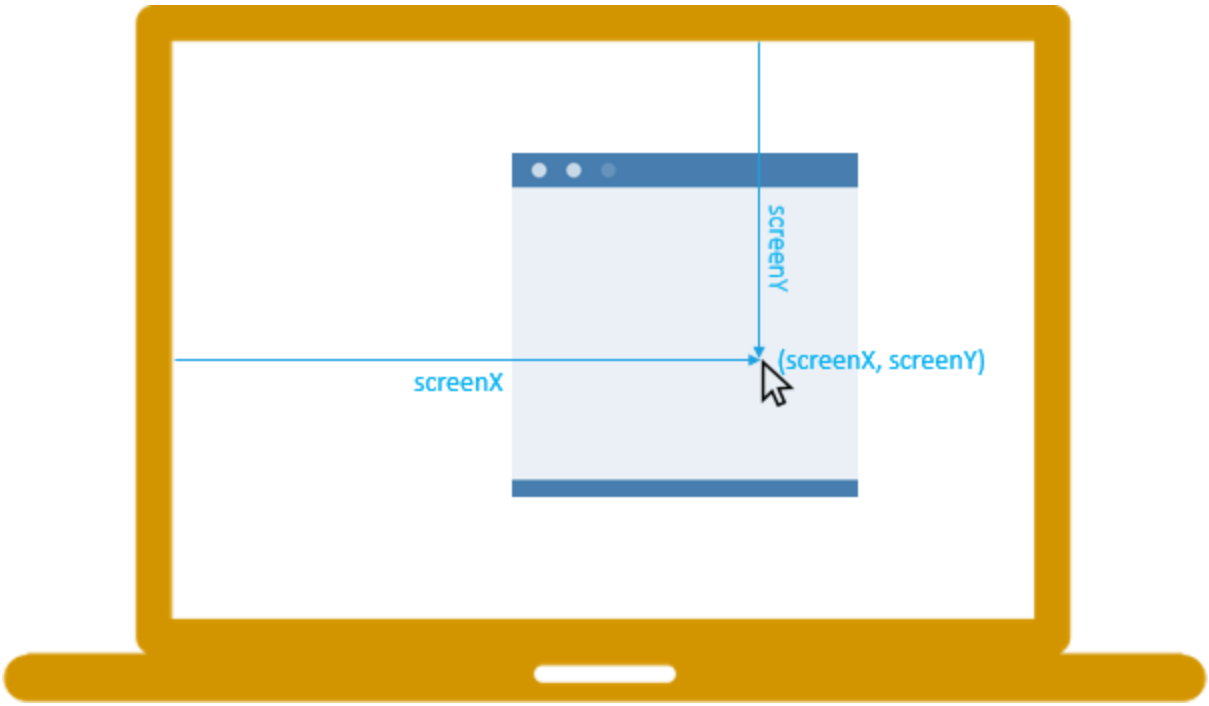
```
            if (e.metaKey) keys.push('meta');

            let msg = document.querySelector('#messageKeys');
            msg.textContent = `Keys: ${keys.join('+')}`;
        });
    </script>
</body>
</html>
```

Output

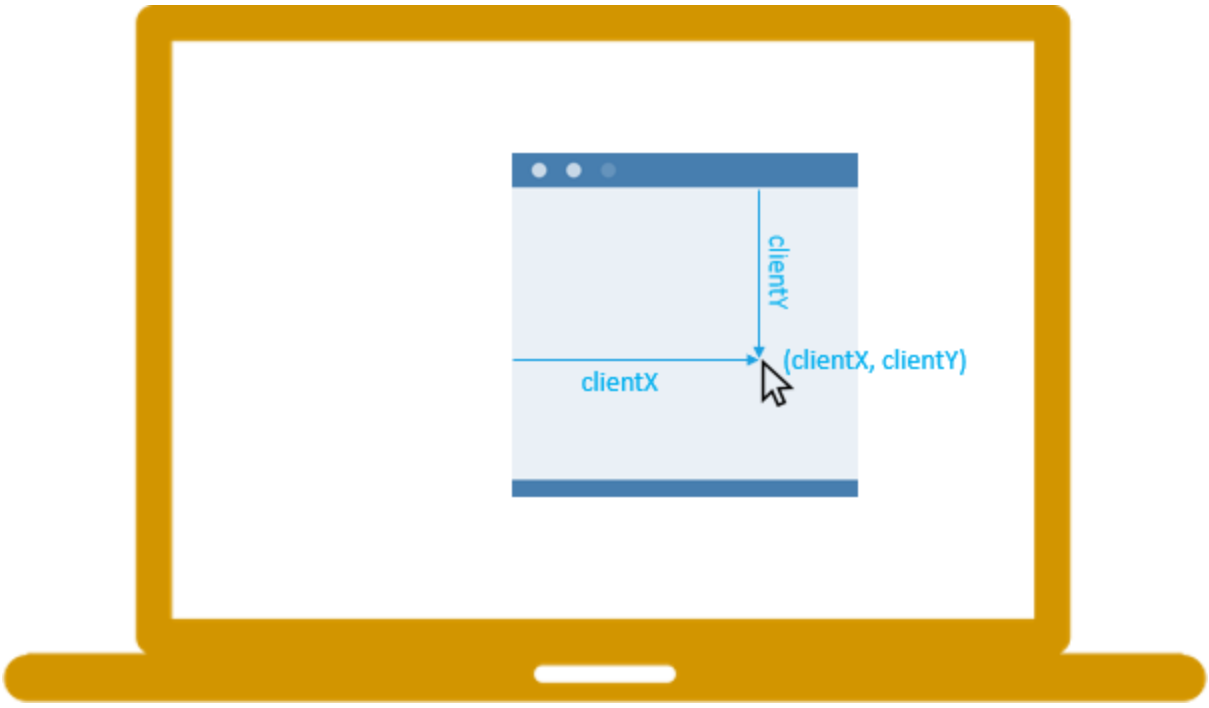


Getting Screen Coordinates

The `screenX` and `screenY` properties of the event passed to the mouse event handler return the screen coordinates of the location of the mouse in relation to the entire screen.



On the other hand, the `clientX` and `clientY` properties provide the horizontal and vertical coordinates within the application's client area at which the mouse event occurred:



See the following demo:

```
<!DOCTYPE html>
<html>
<head>
    <title>JS Mouse Location Demo</title>
```

```
    <style>
        #track {
            background-color: goldenrod;
            height: 200px;
            width: 400px;
        }
    </style>
</head>
<body>
    <p>Move your mouse to see its location.</p>
    <div id="track"></div>
    <p id="log"></p>

    <script>
        let track = document.querySelector('#track');
        track.addEventListener('mousemove', (e) => {
            let log = document.querySelector('#log');
            log.innerText = `
            Screen X/Y: (${e.screenX}, ${e.screenY})
            Client X/Y: (${e.clientX}, ${e.clientY})`
        });
    </script>
</body>
</html>
```

Output

Move your mouse to see its location.

Summary

- DOM Level 3 defines nine mouse events.
- Use `addEventListener()` method to register a mouse event handler.
- The `event.button` indicates which mouse button was pressed to trigger the mouse event.
- The modifier keys: alt, shift, ctrl, and meta (Mac) can be obtained via properties of the event object passed to the mouse event handler.
- The `screenX` and `screenY` properties return the horizontal and vertical coordinates of the mouse pointer in screen coordinates.
- The `clientX` and `clientY` properties of the `event` object returns horizontal and vertical coordinates within the application's client area at which the mouse event occurred.

Quiz