In JavaScript, events are actions or occurrences that happen in the browser, and they are a fundamental part of creating interactive and dynamic web pages. Some of the most frequently used event types in JavaScript include:

**Type of event**

1. \*\*Click Event:\*\*

- Triggered when a mouse button is pressed and released on an element.

```javascript

element.addEventListener('click', function() {

// Your code here

});

```

2. \*\*Mouseover and Mouseout Events:\*\*

- Mouseover is triggered when the mouse pointer enters an element.

- Mouseout is triggered when the mouse pointer leaves an element.

```javascript

element.addEventListener('mouseover', function() {

// Your code here

});

element.addEventListener('mouseout', function() {

// Your code here

});

```

3. \*\*Keydown, Keypress, and Keyup Events:\*\*

- Keydown is triggered when a key is pressed down.

- Keypress is triggered when a key is pressed.

- Keyup is triggered when a key is released.

```javascript

document.addEventListener('keydown', function(event) {

// Your code here

});

document.addEventListener('keypress', function(event) {

// Your code here

});

document.addEventListener('keyup', function(event) {

// Your code here

});

```

4. \*\*Change Event:\*\*

- Triggered when the value of an input element changes (e.g., input, select, textarea).

```javascript

element.addEventListener('change', function() {

// Your code here

});

```

5. \*\*Submit Event:\*\*

- Triggered when a form is submitted.

```javascript

form.addEventListener('submit', function(event) {

event.preventDefault(); // Prevents the form from submitting in the traditional way

// Your code here

});

```

6. \*\*Load Event:\*\*

- Triggered when a resource and its dependent resources have finished loading.

```javascript

window.addEventListener('load', function() {

// Your code here

});

```

These are just a few examples of the many events available in JavaScript. Understanding and effectively using events are crucial for building interactive and responsive web applications.

Event: timeStamp property

The timeStamp read-only property of the Event interface returns the time (in milliseconds) at which the event was created.

Value

This value is the number of milliseconds elapsed from the beginning of the time origin until the event was created. If the global object is Window, the time origin is the moment the user clicked on the link, or the script that initiated the loading of the document. In a worker, the time origin is the moment of creation of the worker.

// Events Timestamp

// document.querySelector("body").addEventListener("keypress",function(e){

// console.log((e.timeStamp/1000));

// })

Event: target property

The read-only target property of the Event interface is a reference to the object onto which the event was dispatched. It is different from Event.currentTarget when the event handler is called during the bubbling or capturing phase of the event.

Value

The associated EventTarget.

Example

// Make a list

const ul = document.createElement("ul");

document.body.appendChild(ul);

const li1 = document.createElement("li");

const li2 = document.createElement("li");

ul.appendChild(li1);

ul.appendChild(li2);

function hide(evt) {

// evt.target refers to the clicked <li> element

// This is different than evt.currentTarget, which would refer to the parent <ul> in this context

evt.target.style.visibility = "hidden";

}

// Attach the listener to the list

// It will fire when each <li> is clicked

ul.addEventListener("click", hide, false);

MouseEvent: clientX property

The clientX read-only property of the MouseEvent interface provides the horizontal coordinate within the application's viewport at which the event occurred (as opposed to the coordinate within the page).

For example, clicking on the left edge of the viewport will always result in a mouse event with a clientX value of 0, regardless of whether the page is scrolled horizontally.

Value

A double floating point value.

MouseEvent: clientY property

The clientY read-only property of the MouseEvent interface provides the vertical coordinate within the application's viewport at which the event occurred (as opposed to the coordinate within the page).

For example, clicking on the top edge of the viewport will always result in a mouse event with a clientY value of 0, regardless of whether the page is scrolled vertically.

Value

A double floating point value.

**MouseEvent: screenX property**

The screenX read-only property of the MouseEvent interface provides the horizontal coordinate (offset) of the mouse pointer in screen coordinates.

Value

A double floating point value.

**MouseEvent: screenY property**

The screenY read-only property of the MouseEvent interface provides the vertical coordinate (offset) of the mouse pointer in screen coordinates.

Value

A double floating point value.

MouseEvent: altKey property

The MouseEvent.altKey read-only property is a boolean value that indicates whether the alt key was pressed or not when a given mouse event occurs.

Be aware that the browser can't always detect the alt key on some operating systems. On some Linux variants, for example, a left mouse click combined with the alt key is used to move or resize windows.

// altkey

// document.querySelector("body").addEventListener("click",function(e){

// console.log(`Alt key is press ${e.altKey}`);

// },false)

**MouseEvent: ctrlKey property**

The MouseEvent.ctrlKey read-only property is a boolean value that indicates whether the ctrl key was pressed or not when a given mouse event occurs.

On Macintosh keyboards, this key is labeled the control key. Also, note that on a Mac, a click combined with the control key is intercepted by the operating system and used to open a context menu, so ctrlKey is not detectable on click events.

// Cltrlkey

document.querySelector("body").addEventListener("click",function(e){

console.log(`cltrl key is press ${e.ctrlKey}`);

})

**MouseEvent: shiftKey property**

The MouseEvent.shiftKey read-only property is a boolean value that indicates whether the shift key was pressed or not when a given mouse event occurs.

Value

A boolean value, where true indicates that the key is pressed, and false indicates that the key is not pressed.

document.querySelector("body").addEventListener("click",function(e){

console.log(`shift key is press ${e.shiftKey}`);

},false)