**********events==========
UI Events:
==========

Mouse event types:

1.mousedown/mouseup :Mouse button is clicked/released over an element.
2.mouseover/mouseout :Mouse pointer comes over/out from an element.
3.mousemove   :Every mouse move over an element triggers that event.
4.click   :Triggers after mousedown and then mouseup over the same element if the left mouse button was used.
5.dblclick   :Triggers after two clicks on the same element within a short timeframe. Rarely used nowadays.
6.contextmenu   :Triggers when the right mouse button is pressed.

The events are categorized mainly into four groups:

1.mouse events
2.keyboard events
3.form events
4.document/window events

Mouse events:

A Mouse Event gets triggered when the user interacts with a mouse (pointing device) on the webpage,
                                          such as by clicking on an element or moving the mouse over the element.

The MouseEvent interface derives from the UIEvent interface, which in turn derives from the Event interface.

The Click Event:

The click event gets fired when a mouse is both pressed and released on a single element.

we can say when the user clicks on a button on the webpage, the click event is fired.

We can handle this event using an onclick event handler.

Syntax:

```
<element onclick="myFunction">

object.onclick = function() { myFunction() };

object.addEventListener("click", myFunction);
```

The Contextmenu Event:

The contextmenu event gets fired when the user attempts to open a context menu.

This event is typically triggered when the user right-clicks on an element to open the context menu.

We can handle this event using an oncontextmenu event handler.

Syntax:

```
<element oncontextmenu="myFunction">

object.oncontextmenu = function(){myFunction()};

object.addEventListener("contextmenu", myFunction);
```

The Mousedown Event:

The mousedown event gets fired when the user presses a mouse button on an element.

It differs from the click event as the click event gets fired when the mouse button is pressed and released while the pointer remains inside the same element.

The mousedown event is triggered the moment the mouse button is initially pressed.

It enables you to distinguish between the left, middle (scrolling wheel) and right mouse buttons.

We can handle this event using an onmousedown event handler.

Syntax:

```
<element onmousedown="myFunction">

object.onmousedown = function(){myFunction()};

object.addEventListener("mousedown", myFunction);
```

The Mouseup Event:

The mouseup event gets fired when the user releases the mouse button over an element.

If the user clicks outside an element, drags the mouse onto it, and releases the button, it is still counted as a mouseup event.

This method is often used together with the mousedown() method.

We can handle this event using an onmouseup event handler.

Syntax:

```
<element onmouseup="myFunction">

object.onmouseup = function(){myFunction()};

object.addEventListener("mouseup", myFunction);
```

The Mouseenter Event:

The mouseenter event gets fired when the user moves the mouse button inside or the mouse enters the element.

It is used to create nice effects with images as well as with text.

We can handle this event using an onmouseenter event handler.

Syntax:

```
<element onmouseenter="myFunction">

object.onmouseenter = function(){myFunction()};

object.addEventListener("mouseenter", myFunction);
```

mousleave event:

The mouseleave event gets fired when the mouse button moves outside or leaves the element.

This method is often used together with the mouseenter() method.

And both mouseenter and mouseleave events do not bubble and do not get fired when the mouse cursor enters or leaves its child elements.

We can handle this event using an onmouseleave event handler.

Syntax:

```
<element onmouseleave="myFunction">

object.onmouseleave = function(){myFunction()};

object.addEventListener("mouseleave", myFunction);
```

The Mouseover Event:
==========

The mouseover event gets fired when the user moves the mouse button inside the element or one of its children.

It is similar to the mouseenter event, but the mouseover event differs as it gets triggered every time the mouse enters the div element

or its child elements. On the other hand, the mouseenter event gets triggered only when the mouse enters the div element.

We can handle this event using an onmouseover event handler.

```
<element onmouseover="myFunction">

object.onmouseover = function(){myFunction()};

object.addEventListener("mouseover", myFunction);
```

The Mouseout Event:

The mouseout event gets fired when the user moves the mouse button outside the element or one of its children.

It is always confused with the mouseleave event, but they both differ.

In the case of the mouseleave event, if the parent element (the element to which the event is attached) has any descendants or child elements,

this event gets fired only when the mouse leaves the parent element and not the child elements.

On the other hand, the mouseout event gets triggered every time the mouse leaves the parent element as well as its child elements.

We can handle this event using an onmouseout event handler.

```
<element onmouseout="myFunction">

object.onmouseout = function(){myFunction()};

object.addEventListener("mouseout", myFunction);
```

The Mousemove Event;

The mouseout event gets fired when the user moves the mouse button or hovers over an element.

We can handle this event using an onmousemove event handler.

```
<element onmousemove="myFunction">
```

```
object.onmousemove = function(){myFunction()};

object.addEventListener("mousemove", myFunction);
```

**********Keyboard Events**********

A Keyboard Event gets triggered when the user interacts with any keys on the keyboard on the webpage.
The KeyboardEvent interface also derives from the UIEvent interface, which in turn derives from the Event interface.
The common events that belong to the KeyboardEvent are:

The KeyDown Event:

The keydown event gets fired when the user presses any key (on the keyboard).

We can handle this event using an onkeydown event handler.

```
<element onkeydown="myFunction">

object.onkeydown = function(){myFunction()};

object.addEventListener("keydown", myFunction);
```

The KeyUp Event:

The keyup event gets fired when the user releases a key (on the keyboard).

When you press any key on the keyboard, the keydown event gets triggered first, and if you hold down the key, the keydown event gets

triggered multiple times.

Only when you release the key, does the keyup event gets fired. We can handle this event using an onkeyup event handler.

Syntax:

```
<element onkeyup="myFunction">

object.onkeyup = function(){myFunction()};

object.addEventListener("keyup", myFunction);
```

Form events:

A javascript Form event gets triggered when a user interacts with a form in a website.

Form events are used to make the form filling process more interactive and informative for the user.

These forms events include:

The Focus Event:

The focus event gets fired once the element has received focus.

The most common job is to change the background color of the HTML element on focus.

It clearly distinguishes it from the others and makes it stand out that this element is the one that is presently on focus.

It belongs to the FocusEvent interface, which inherits all the properties and methods from the UIEvent interface,
which in turn derives from the Event interface. We can handle this event using an onfocus event handler.

```
<element **onfocus**="myFunction">

object.**onfocus** = function(){myFunction()};

object.addEventListener("**focus**", myFunction);
```

The Blur Event:

The blur event gets fired once the element loses its focus.

It belongs to the FocusEvent interface, which inherits all the properties and methods from the UIEvent interface,

which in turn derives from the Event interface. We can handle this event using an onblur event handler.

Syntax:

```
<element onblur="myFunction">

object.onblur = function(){myFunction()};

object.addEventListener("blur", myFunction);
```

The Submit Event:

The submit event gets fired when the user submits a form.

It is used to validate the data entered before the user submits the request to the server-side.

It belongs to the SubmitEvent interface, which inherits all the properties and methods from the Event interface.

We can handle this event using an onsubmit event handler.

Syntax:

We can handle the submit event either by using the onsubmit event handler in HTML or Javascript
and assigning a JavaScript function to it or by using JavaScript's addEventListener() method to attach the submit event to the HTML element.

```
<element onsubmit="myFunction">

object.onsubmit = function(){myFunction()};

object.addEventListener("submit", myFunction);
```

The Change Event:

The change event gets fired when the user changes the value of an HTML element (i.e., <input >, <select >, and <textarea >) on the webpage. It belongs to the Event interface.

We can handle this event using an onchange event handler.

Syntax:

We can handle the change event either by using the onchange event handler in HTML or Javascript and assigning a JavaScript function to
it or by using JavaScript's addEventListener() method to attach the change event to the HTML element.

```
<element onchange="myFunction">

object.onchange = function(){myFunction()};

object.addEventListener("change", myFunction);
```

Window/Document Events:

A Window event gets triggered when the user does something that affects an entire browser window.

For example, when the webpage loads or when the user resizes the browser window. These window events include:

The Load Event:

The load event gets fired when the whole page has loaded, including all dependent resources such as stylesheets and images.

It can also be used to deal with cookies. It belongs to the Event interface and we can handle this event using an onload event handler.

Syntax:

We can handle the load event either by using the onload event handler in HTML or Javascript
 and assigning a JavaScript function to it or by using JavaScript's addEventListener() method to attach the load event to the HTML element.

```
<element onload="myFunction">

object.onload = function(){myFunction()};

object.addEventListener("load", myFunction);
```

****
***
There are many different types of events in javascript.
The following are some of the most common types of events:

Window Events
Mouse Events
Keyboard Events
Form Events
Storage Events
Media Events
Drag Events
Clipboard Events

onload event:

If you want to execute a function when the page is loaded, you can use the onload event.

This event is triggered when the page is loaded.

```
<body onload="alert('Page loaded!')">
  <p>As soon a page loaded it alerts "Page loaded!" message.</p>
</body>
```

onresize event:

The onresize event is triggered when you resize the browser window.

```
<body onresize="resizeW()">

  <script>
    function resizeW() {
      console.log('The window has been resized');
    }
  </script>
</body>
```

onbeforeunload event:

The onbeforeunload event is triggered when the user attempts to close the browser tab.

Example

```
<body onbeforeunload="return confirm('Are you sure you want to close the tab?')">
  <p>If you close the tab, it will ask you to confirm.</p>
</body>
```

onoffline event:

The onoffline event is triggered when the browser is offline.

Example

```
<body onoffline="offLine()">
  <p id="demo"></p>

  <script>
    function offLine() {
      const demo = document.getElementById("demo");
      demo.innerHTML = "You are offline";
      demo.style.color = "red";
    }
  </script>
</body>
```

ononline event:

The ononline event is triggered when the browser is online.

Example

```
<body ononline="onLine()">
  <p id="demo"></p>

  <script>
    function onLine() {
```

```
      const demo = document.getElementById("demo");
      demo.innerHTML = "You are online";
      demo.style.color = "green";
    }
  </script>
</body>
```

Mouse events in javascript:

Mouse events are those events that are triggered on some activity by the mouse, like clicking, hovering, dragging, etc.

The following is the list of mouse events in JavaScript:

Event	Action	Description
click	Mouse click	when the mouse is clicked over an element
dblclick	Mouse double click	Mouse double clickwhen the mouse is double-clicked over an element
mouseover	Mouse over	Triggered when the user moves the mouse over an element
mouseout	Mouse out	Triggered when the user moves the mouse out of an element
mousedown	Mouse button down	Triggered when the user presses the mouse button.
mouseup	Mouse button up	Triggered when the user releases the mouse button.
mousemove	Mouse move	Triggered when the user moves the mouse over an element
mouseenter	Mouse enter	Triggered when the user moves the mouse over an element
mouseleave	Mouse leave	Triggered when the user moves the mouse out of an element

click event:

The click event is triggered when the user clicks the mouse button.

Example

```
<p id="elm" onclick="changeColor()">Click this element</p>

<script>
  function changeColor() {
    document.getElementById("elm").style.color = "red";
  }
</script>
```

dblclick event:

The dblclick event is triggered when the user double click the mouse button.

Note: When you double click the mouse button then the click event is triggered first 2 times and then the dblclick event is triggered.

Example

```
<p id="elm" ondblclick="changeColor()">Double click this element</p>

<script>
  function changeColor() {
    document.getElementById("elm").style.color = "red";
  }
</script>
```

mouseover event:

When you take the mouse cursor over an element then the mouseover event is triggered.

You can use this event to perform some action when the mouse cursor is over an element like changing the color of the element.

Example

```
<button onmouseover="changeColor()" id="btn">Change Color</button>

<script>
  function changeColor() {
    document.getElementById("btn").style.backgroundColor = "red";
  }
</script>
```

mouseout event:

When you take the mouse cursor out of an element then the mouseout event is triggered.

You can also use it to reverse the action of the mouseover event.

Example

```
<button onmouseover="changeColor()" onmouseout="reverseColor()" id="btn">Change
Color</button>

<script>
  function changeColor() {
    document.getElementById("btn").style.backgroundColor = "red";
  }
  function reverseColor() {
    document.getElementById("btn").style.backgroundColor = "white";
  }
</script>
```

mousedown event:

The mousedown event is triggered when the user presses the mouse button.

You can use this event to open some pop-up or to perform some action when the mouse button is pressed over an element.

Example

```
<button onmousedown="openPopup()" id="btn">Open Popup</button>

<div id="popup">
  <p>This is a popup</p>
</div>

<script>
  function openPopup() {
    document.getElementById("popup").style.display = "block";
  }
</script>
```

mouseup event:_

The mouseup event is triggered when the user releases the mouse button.

You can use this event to close some pop-up or to perform some action when the mouse button is released over an element.

Example

```
<button onmousedown="openPopup()" onmouseup="closePopup()" id="btn">Open/close
Popup</button>

<div id="popup">
  <p>This is a popup</p>
</div>
```

```
<script>
  function openPopup() {
    document.getElementById("popup").style.display = "block";
  }
  // Close the popup
  function closePopup() {
    document.getElementById("popup").style.display = "none";
  }
</script>
```

mousemove event:

The mousemove event is triggered when the user moves the mouse over an element.

In the example below, we have used the mousemove event to track the position of the mouse cursor.

Example

```
<button onmousemove="trackMouse()" id="btn">Track Mouse</button>

<script>
  function trackMouse() {
    var x = event.clientX;
    var y = event.clientY;
    document.getElementById("btn").innerHTML = "Mouse position: " + x + ", " + y;
  }
</script>
```

mouseenter event:

The mouseenter event is triggered when the mouse enters an element.

To check this event in action we will change the background color of an element when the mouse enters it.

Example

```
<div onmouseenter="changeColor()" id="div">
  <p>This is a div</p>
</div>

<script>
  function changeColor() {
    document.getElementById("div").style.backgroundColor = "blue";
  }
</script>
```

mouseleave event:

The mouseleave event is triggered when the mouse leaves an element.

You can also use this event to reverse the changes made by the mouseenter event or can do some other action.

Example

```
<div onmouseenter="enterColor()" onmouseleave="leaveColor()" id="div">
  <p>This is a div</p>
</div>

<script>
  function enterColor() {
    document.getElementById("div").style.backgroundColor = "blue";
  }
  function leaveColor() {
    document.getElementById("div").style.backgroundColor = "red";
  }
</script>
```

Keyboard events in javascript:

Keyboard events are those events that are triggered when you press a key on your keyboard.

There are many different types of keyboard events in javascript. Some of them are:

Event	Action	Description
onkeydown	Button Down	event is fired when any keyboard button is down
onkeyup	Button released	event is fired when any keyboard button is released
onkeypress	Button pressed	event is fired when any keyboard button is pressed

keydown event:

The keydown event is triggered when a key is pressed down.

As soon as you press a key, the browser will fire the keydown event.

Example

```
<p id="output"></p>
<textarea onkeydown="keyDown()"></textarea>

<script>
```

```
  var count = 0;
  function keyDown() {
    var output = document.getElementById('output');
    count++;
    output.innerHTML = "You pressed: " + event.key + ", numberof times: " + count;
  }
</script>
```

keyup event:

The keyup event is triggered when a key is released.

As soon as you release a key, the browser will fire the keyup event.

Example

```
<p id="output"></p>
<textarea onkeyup="keyUp()"></textarea>

<script>
  var count = 0;
  function keyUp() {
    var output = document.getElementById('output');
    count++;
    output.innerHTML = "You released: " + event.key + ", numberof times: " + count;
  }
</script>
```

keypress event:
==========
The keypress event is triggered when a key is pressed and released.

As soon as you press and release a key, the browser will fire the keypress event.

When you compare the keypress event to the keydown event then keydown event is triggered before keypress event.

Example

```
<p id="output"></p>
<textarea onkeypress="keyPress()"></textarea>

<script>
  var count = 0;
  function keyPress() {
    var output = document.getElementById('output');
    count++;
```

```
    output.innerHTML = "You pressed: " + event.key + ", numberof times: " + count;
  }
</script>
```

Form events in javascript

Forms in HTML have many different types of events that can be triggered at different action points.

The form events in javascript are:

Event	Action	Description
submit	onsubmit	event is fired when the form is submitted
focus	onfocus	event is fired when the cursor is focused on form
blur	Focus Removed	event is fired when the focus is removed from input section
change	onchange	event is fired when the value of the input is changed
reset	onreset	event is fired when the form is reset
select	onselect	event is fired when a selection is made

submit event:

The submit event is triggered when you submit the form.

Note: submit event by default refreshes the page when the form is submitted.

Example

```
<form onsubmit="submitForm(event)">
  <input type="text" name="name" placeholder="name">
  <input type="text" name="email" placeholder="email">
  <input type="submit" value="submit">
</form>

<script>
  function submitForm(event) {
    event.preventDefault();
    alert('form submitted');
  }
</script>
```

focus event:

The focus event is triggered when the input gains focus.

Example

```
<input type="text" onfocus="focusInput(this)">

<script>
  function focusInput(event) {
    event.style.backgroundColor = 'orange';
  }
</script>
```

blur event:

The blur event is triggered when the input loses focus.

Example

```
<input type="text" onblur="blurInput()">

<script>
  function blurInput() {
    alert('focus removed');
  }
</script>
```

change event:

The change event is triggered when the value of an input changes.

Example

```
<input type="text" onchange="changeInput()">
<p id="output"></p>

<script>
  function changeInput() {
    document.getElementById('output').innerHTML = this.value;
  }
</script>
```

reset event:

The reset event is triggered when the form is reset.

There lies an input type called reset, which is not a submit button. It is used to reset the form.

Example

```
<form onreset="resetForm()">
  <input type="text" name="name" placeholder="name">
  <input type="text" name="email" placeholder="email">
  <input type="reset" value="reset">
</form>

<script>
  function resetForm() {
    alert('form reset');
  }
</script>
```

select event:

The select event is triggered when a new input is selected.

In the example below, we are going to use the select element to select the input field.

Example

```
<select name="cars" onchange="selectInput()">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="mercedes">Mercedes</option>
  <option value="audi">Audi</option>
</select>

<script>
  function selectInput() {
    alert('you selected: '+this.value);
  }
</script>
```

Storage Event in JavaScript:

The Storage Event is triggered when the storage area is changed.

Note: The storage event triggered only when the storage area of another window is changed not the current window.

The following example shows how to use the Storage Event in JavaScript.

Example

```
<button onclick="addItemLocalStorage()">Add item to local storage</button>
```

```
<script>
  // add new item to local storage
  function addItemLocalStorage() {
    var newWindow = window.open("");
    newWindow.localStorage.setItem("name", "John");
    newWindow.close();
  }

  window.addEventListener("storage", storageEvent);

  function storageEvent() {
    alert("Storage changed");
  }
</script>
```

Media Events in JavaScript:

Media events are events that are triggered by media elements. These events are triggered when different actions are taken or occur on the media element.

The list of media events is shown in the table below.

Event	Description
abort	Triggered when the loading of an audio/video is aborted.
canplay	Triggered when the browser can start playing the audio/video.
canplaythrough	Triggered when the browser can play through the audio/video without stopping for buffering.
durationchange	Triggered when the duration of the audio/video is changed.
emptied	Triggered when the current playlist is empty.
ended	Triggered when the current playlist is ended.
error	Triggered when an error occurs.
loadeddata	Triggered when the browser has loaded the current frame of the audio/video.
loadedmetadata	Triggered when the browser has loaded meta data for the audio/video.
loadstart	Triggered when the browser starts looking for the audio/video.
pause	Triggered when the audio/video is paused.
play	Triggered when the audio/video is played.
playing	Triggered when the audio/video is playing.
progress	Triggered when the browser is downloading the audio/video.
ratechange	Triggered when the playing speed of the audio/video is changed.
sought	Triggered when the user is finished moving/skipping to a new position in the audio/video.
seeking	Triggered when the user starts moving/skipping to a new position in the audio/video.

stalled	Triggered when the browser is trying to get media data, but data is not available.
suspend	Triggered when the browser is intentionally not getting media data.
timeupdate	Triggered when the current playback position has changed.
volumechange	Triggered when the volume has been changed.
waiting	Triggered when the video stops because it needs to buffer the next frame.

Drag Events in JavaScript:

Drag events are events that are triggered by drag and drop actions.

When you drag a draggable element, the browser fires the dragstart event. This event is followed by a number of events, depending on the type of drag action.

The list of drag events is shown in the table below.

Event	Description
ondrag	Triggered when an element is being dragged
ondragend	event occurs when an element stops being dragged
ondragenter	event occurs when an element or text selection is being dragged over a valid drop target (every few hundred milliseconds).
ondragleave	event occurs when an element or text selection leaves a valid drop target.
ondragover	event occurs when an element or text selection is being dragged over a valid drop target (every few hundred milliseconds).
ondragstart	event occurs when an element is being dragged
ondrop	event occurs when an element or text selection is being dropped on a valid drop target.

Here is an example that shows different drag events in action.

Example

```
<p draggable="true" ondrag="onDrag()" ondragstart="dragStart()"
ondragend="dragEnd()">Drag me</p>
<textarea ondragenter="dragEnter()" ondragleave="dragLeave()"
ondragover="dragOver(event)"></textarea>
<p id="output"></p>
<script>
  const output = document.getElementById("output");
  function onDrag() {
    output.innerHTML += "Dragging...<br>";
  }
  function dragStart() {
    output.innerHTML = "Dragging started...<br>";
```

```
  }
  function dragEnd() {
    output.innerHTML += "Dragging ended...<br>";
  }
  function dragEnter() {
    output.innerHTML += "Dragging entered...<br>";
  }
  function dragLeave() {
    output.innerHTML += "Dragging left...<br>";
  }
</script>
```

Clipboard Events in JavaScript:

Clipboard events are events that are triggered by the copy, cut, and paste actions.

The list of clipboard events is shown in the table below.

Event	Description
oncopy	Triggered when the user copies text to the clipboard.
oncut	Triggered when the user cuts text to the clipboard.
onpaste	Triggered when the user pastes text from the clipboard.

copy event:

When you copy text to the clipboard, the browser fires the oncopy event.

Example

```
<p oncopy="copy()">Copy me!</p>

<script>
  function copy() {
    alert("Copied!");
  }
</script>
```

cut event:

When you cut text to the clipboard, the browser fires the oncut event.

Example

```
<input type="text" oncut="cut()" value="Cut Me!">

<script>
```

```
  function cut() {
    alert("Text cut!");
  }
</script>
```

paste event:

When you paste text from the clipboard, the browser fires the onpaste event.

Example

```
<input type="text" onpaste="pasteHandler()">

<script>
  function pasteHandler() {
    alert("Paste event is fired");
  }
</script>
```