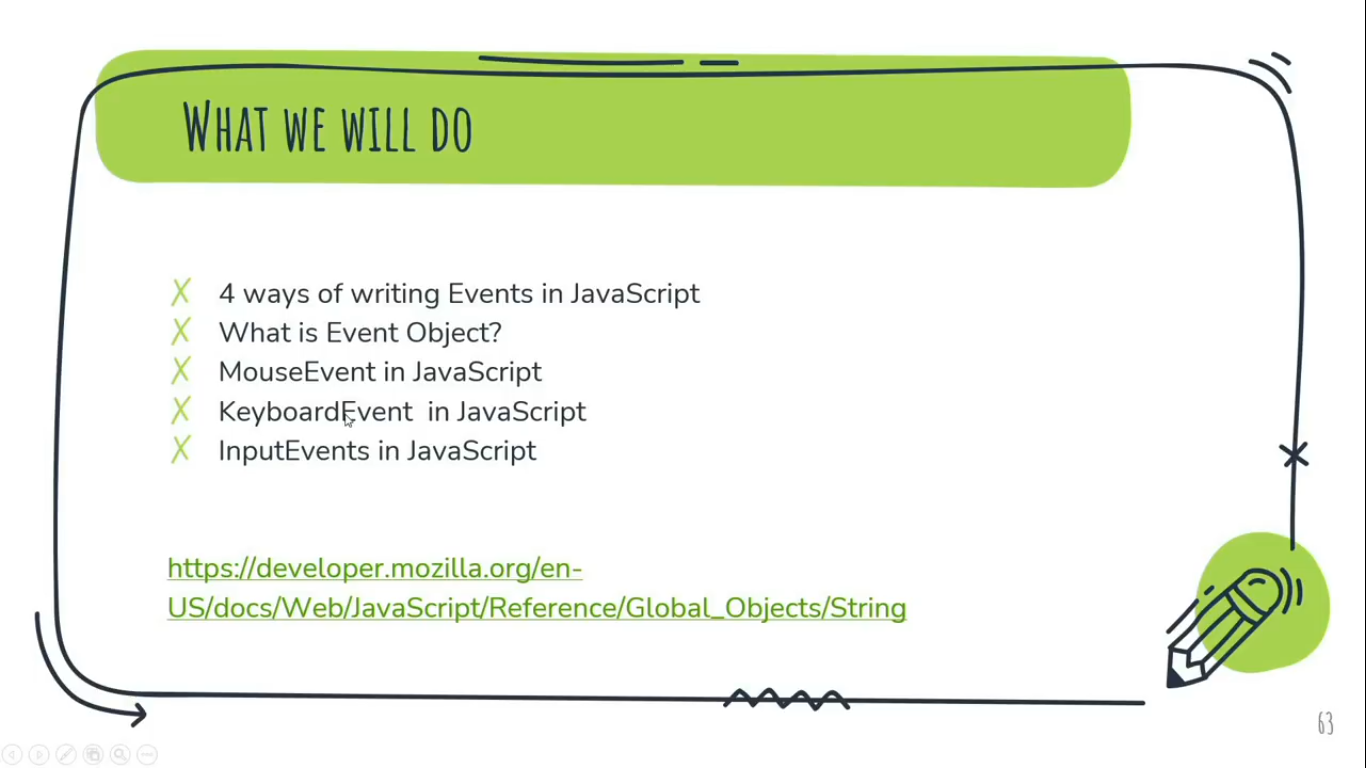
**Events**

****

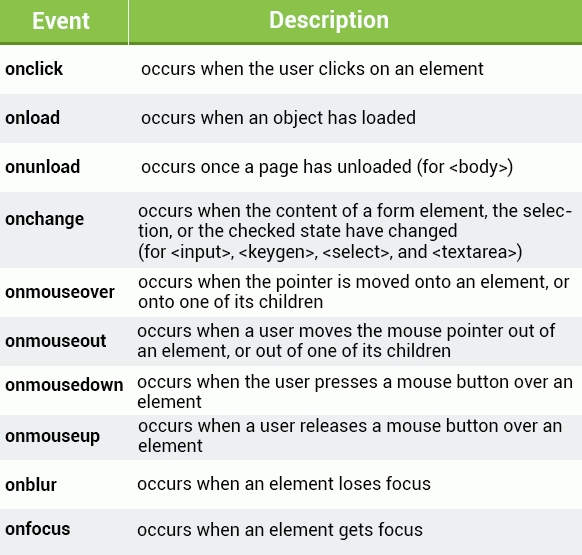
An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

* An HTML web page has finished loading.
* An HTML input field was changed.
* An HTML button was clicked.
* Often, when events happen, we may want to do something.

HTML allows event handler attributes, with JavaScript code, to added to HTML elements.

You can write JavaScript code that executes when an **event**occurs, such as when a user clicks an HTML element, moves the mouse, or submits a form.  
When an event occurs on a target element, a **handler**function is executed.  
Common HTML events include:



**Q.** The type of function that executes when an event occurs is called:  
Ans:- **event handler function**

The **onload**and **onunload**events are triggered when the user enters or leaves the page. These can be useful when performing actions after the page is loaded.

Ex: <body **onload**="doSomething()">

Similarly, the **window.onload** event can be used to run code after the whole page is loaded.

**window.onload** = function() {   
//some code   
}

The **onchange**event is mostly used on textboxes. The event handler gets called when the text inside the textbox changes and focus is lost from the element.

**4 ways of writing Events in JS:-**

1. Using inline events alert();
2. By Calling a function (Most common way of writing)
3. Using Inline events (HTML onclick=”” property and element.onclick or say using by reference)
4. Using Event Listeners (addEventListener and IE’s attachEvent)

**1st Way,**

<button onclick=”alert(‘hello’)”>click Me</button>

**2nd Way,**

<button onclick=”check”>click Me</button>

<script>

//Function Expression  
const check = () => { //Anonumous function with Fat Arrow Function  
alert(“Hello”);  
}  
</script>

**3rd way,**

<button class="btn-2">Me Click</button>

<script>  
//**Using by Reference**  
const fire = document.querySelector('.btn-2');  
fire.onclick = function () { **// here fire is the reference**  
alert("hi");   
console.log("Check Console");}  
</script>

**4th way,**

<button class="btn-2">Me Click</button>

<script>  
//**Using by Reference with addEventListener method or Function**  
const fire = document.querySelector('.btn-2');  
fire.addEventListener(‘click’, ()=>{   
alert("hi");  
console.log("Check Console");  
})  
//**passing two Parameters ‘Event as click’, ‘function as CallBack Function i.e. (Function under Function/Method)**  
</script>

**Note:**

In the 3rd way, it’s impossible to execute multiple statement under the same function by firing event because the last statemet override the top statement.

On the other hand, In the 4th way, It’s possible to execute multiple statements under the same function one by one from Top to Bottom.

Onclick will always work, but addEventListener doesn’t work in Interner Explorer before version 9.

This is What the **Difference** between **onclick** and **addEventListener.**

**Q. What is Event Object?**

Event Object is the parent object of the event objects. Or All html events are the part of Event Objects.  
For Example:  
MouseEvent, focusEvent, KeyboardEvent, InputEvent etc.

**// To get Event Object**

<button class="btn-2">Me Click</button>  
<script>  
const Newfire = document.querySelector('.btn-2');  
Newfire.addEventListener('click', ()=>{  
console.log(event); **//** **event** **is the pre-defined object in JS**  
console.log(event.type); **//** **event** **with its type property**   
}  
);  
</script>

**MouseEvent in JS:**

The MouseEvent Object  
Events that occur when the mouse interacts with the HTML document belongs to the MouseEvent Object.  
Sample Events:  
onmousedown //mouse click down  
onmouseup //mouse click up or leave click  
onmouseenter //mouse arrow hover in  
onmouseleave //mouse arrow hover out

**KeyboardEvent in JS:**

The KeyboardEvent Object  
Events that occur when user presses a key on the keyboard, belongs to the KeyboardEvent Object.  
Sample Event:  
onkeypress //key press on keyboard  
onkeyup //key up on keyboard  
onkeydown //key down on keyboard  
onmouseleave //mouse arrow hover out

For Example:  
<input type="text" class="field"/>  
<div class="container"></div>

<script>  
const fire =document.querySelector('.field');  
fire.addEventListener('keypress', ()=>{  
document.querySelector('.container').innerHTML=`You Pressed ${event.key} and its code is ${event.code}`;  
});  **//Output: You Pressed C and its code is KeyC**

const fire =document.querySelector('.field');  
fire.addEventListener('keydown', ()=>{  
document.querySelector('.container').innerHTML=`Key is Down`;  
});

fire.addEventListener('keyup', ()=>{  
document.querySelector('.container').innerHTML=`Key is UP`;  
}); **//Output: Key is Down OR Key is UP**  
</script>

**InputEvents in JS:**

The InputEvents Object  
The onchange event occurs when the value of an element has been changed.

For radiobuttons and checkboxes, the onchange event occurs when the checked state has been changed.   
Sample Event:  
onchange //data change of input field

For Example:

<input type="text" class ="nameField" name="" placeholder="Enter Your Name"/>

<select class="opt">  
<option vlaue="">Select</option>   
<option vlaue="Admin">Admin</option>  
<option vlaue="Teacher">Teacher</option>  
<option vlaue="Student">Student</option>  
</select>

<script>  
var optValue = document.querySelector('.opt');

optValue.addEventListener('change', ()=>{  
var nameField = document.querySelector('.nameField').value;  
var optValue = document.querySelector('.opt').value;   
document.querySelector('.container').innerHTML=`Hey, ${nameField}, You are ${optValue}`;

})  
</script>

**OR**

<input type="text" class ="nameField" name="" placeholder="Enter Your Name"/>

<select class="opt" onchange="changeValue()">  
<option vlaue="">Select</option>   
<option vlaue="Admin">Admin</option>  
<option vlaue="Teacher">Teacher</option>  
<option vlaue="Student">Student</option>  
</select>

<script>  
var changeValue = ()=>{  
var nameField = document.querySelector('.nameField').value;  
var optValue = document.querySelector('.opt').value;  
document.querySelector('.container').innerHTML=`Hey, ${nameField}, You are ${optValue}`;

}

</script>

**Event Listeners**  
  
The **addEventListener**() method attaches an event handler to an element without overwriting existing event handlers. You can add *many*event handlers to one element.  
You can also add many event handlers of the same type to one element, i.e., two "click" events.

element.**addEventListener**(event, function, useCapture);

The first parameter is the event's **type**(like "click" or "mousedown").  
The second parameter is the **function**we want to call when the event occurs.  
The third parameter is a Boolean value specifying whether to use event **bubbling**or event **capturing.**

**Event Propagation**  
  
There are two ways of event propagation in the HTML DOM: **bubbling**and **capturing**.  
  
Event propagation allows for the definition of the element order when an event occurs. If you have a <p> element inside a <div> element, and the user clicks on the <p> element, which element's "click" event should be handled first?  
  
In **bubbling**, the innermost element's event is handled first and then the outer element's event is handled. The <p> element's click event is handled first, followed by the <div> element's click event.  
  
In **capturing**, the outermost element's event is handled first and then the inner. The <div> element's click event is handled first, followed by the <p> element's click event.

**Note:** Capturing goes **down**the DOM.  
 Bubbling goes **up**the DOM.

**Capturing vs. Bubbling**

The **addEventListener**() method allows you to specify the propagation type with the "**useCapture**" parameter.

addEventListener(event, function, **useCapture**)

The default value is **false**, which means the bubbling propagation is used; when the value is set to **true**, the event uses the capturing propagation.

//Capturing propagation   
elem1.addEventListener("click", myFunction, **true**);   
  
//Bubbling propagation   
elem2.addEventListener("click", myFunction, **false**);

**Note:** This is particularly useful when you have the same event handled for multiple elements in the DOM hierarchy.

**Project *on EventListener:- (Bulb Turn-On & Turn-Off), (slider)****<file:///E:/xampp/htdocs/websites/Learning/JavaScript/Tutorial/Practical/Project/Event%20Listiner/Bulb%20On%20&%20Off/index.html>*(Open in Chrome Browser)

[*file:///E:/xampp/htdocs/websites/Learning/JavaScript/Tutorial/Practical/Project/Event%20Listiner/Bulb%20On%20&%20Off/index.html*](file:///E:/xampp/htdocs/websites/Learning/JavaScript/Tutorial/Practical/Project/Event%20Listiner/Bulb%20On%20&%20Off/index.html)(Open in Chrome Browser)