Event Listeners

Event Listners are used to add and remove **JavaScript Events**. The most preferred and recommended way of dealing with **javascript events** is to use **event listeners**. These **event listeners** were introduced in ECMA 5. Thus all modern web browsers support them (*Except I.E 8 and below*).

To use event listeners, we use addEventListener() function (to add event),
and removeEventListener() function (to remove an event).

Advantage of using Event Listeners

1. Can add multiple functions on same event.
2. Event can be removed using removeEventListener function.
3. Event Propagation can be changed.
4. Can use custom events, like touch events.

Drawback of Event Listners

1. No support for IE 8 and below.

Type of Event Listeners

addEventListener
removeEventListener

---

addEventListener

addEventListener is the most preferred and recommended way to handle **javascript events**.

Inside **addEventListener** function, first parameter is event type string, like click, mouseover, mouseout, scroll etc and second parameter is function that will be called when event occurs. Second parameter can also be an anonymous function. Third parameter is boolean.

---

addEventListener with callback function

```
document.body.addEventListener("click",doThis);

function doThis(){
    alert(this.textContent);
```

```
    }
```

or

addEventListener with anonymous function

```
document.body.addEventListener("click",function(){});
```

In this example, we are using *click event* on body element. *addEventListener* use two parameters. First event type, and second is callback function, that is invoked when event occurs. Third parameter (*Boolean value true or false*) is option as it is used only when we change event propagation.

```
document.querySelector("button").addEventListener("click",function(){
    //action
});

    or

document.querySelector("button").addEventListener("click", myFunction);

function myFunction(){
    //action
}
```

removeEventListener

Event listeners can be removed using **removeEventListener()** method. This is another advantage of using event listeners. Here is an example.

Click Click here once

```
var btn=document.querySelector("button");
btn.addEventListener("click", remove);

function remove(){
    alert("Clicked");
    btn.removeEventListener("click",remove)
}
```

preventDefault

preventDefault() function is used to stop default action on event. For example, if user click an hyperlink, first JS function will be executed, and then page will redirect to href. But using **preventDefault** function can stop default action of hyperlink.

preventDefault() is used on click of hyperlinks and form submit. By default, clicking submit button can form data even if it is invalid, but **preventDefault()** can stop default functionality.

Link

```
document.querySelector("a").addEventListener("click",function(e){
    //  e is event name here
    e.preventDefault();
    alert("Link clicked");
})
```

Submit

```
document.querySelector("form").addEventListener("submit",function(e){
    e.preventDefault();
    alert("Submit button clicked");
})
```

Event Propagation

Event Propagation is the order that the events fire. By default events propagates from child to parent node, also known as bubbling.

Type of event propagations

Bubbling
Capturing

Event Bubbling

Event Bubbling is when event fire on an elements and then bubbles up to parent Elements till it reach root node. By default all events are bubbling type. In example show, click events is used on both parent div, and child button. But button will call first.

Event Bubbling

```
<div id="div1">
    <button id="button1">Button 1</button>
</div>

<script>
document.querySelector("#div1").addEventListener("click",function(){
            alert("you clicked div");
        });
```

```
document.querySelector("#button1").addEventListener("click",function(){
            alert("you clicked button");
        });
</script>
```

Event capturing

Event capturing is when root node is fired first, and then Propagate downwards until it reach targeted element. Capturing is possible by using third boolean parameter (true) in event listener. Here is an example.

Event Capturing

```
<div id="div2">
    <button id="button2">Button 1</button>
</div>

<script>
document.querySelector("#div2").addEventListener("click",function(){
            alert("you clicked div");
        },true);
document.querySelector("#button2").addEventListener("click",function(){
            alert("you clicked button");
        },true);
</script>
```

Touch Events

ES5 introduced **Touch Based events in javascript** . Touch events are applicable for Touch Based Devices only.

Type of Touch Events

1. **touchstart**
2. **touchmove**
3. **touchend**

Touch Event Example

Touch not Supported


```
var img=document.querySelector("img);

img.addEventListener("touchstart",function(){
    this.style.boxShadow="0px 0px 5px 1px #000";
});
img.addEventListener("touchmove",function(e){
   e.preventDefault();
   if (e.targetTouches.length == 1) {
    var touch = e.targetTouches[0];
    // Place element where the finger is
    obj.style.left = (touch.clientX-obj.clientLeft) + 'px';
    obj.style.top = (touch.clientY-obj.clientTop-350) + 'px';
    // position of image should be relative in css

});
img.addEventListener("touchend",function(){
    this.style.boxShadow="0px 0px 5px 1px transparent";
});
```

Event Listeners are not supported in IE 8 and below browser. For Old Browsers support, use property based click events, or Jquery.