**JAVASCRIPT – CLICK EVENT**

1. Traditional event registration like:

element1.onclick = doSomething2;

registers handler for bubbling phase.

2. EVENT Object Properties :

a). ***target/srcElement*** : Element that initiated event dispatch.

3. We can write as many event handlers as we want by multiple writing ***Ele.addEventListener(\_)***and in IE it is writing multiple ***Ele.attachEvent(\_).***

4. Removes the event handler :

***element.removeEventListener('click',spyOnUser,false)***

***element.detachEvent(‘click’, spyOnUser);***

5. In its very recent DOM Level 3 Events W3C adds an ***eventListenerList*** to store a list of event handlers

that are currently registered on an element.

6. When we use like :- element.onclick = doSomething;

The function “doSomething” is copied in its entirety to the onclick property (which now becomes a method). So here this refers to the element on which handler is called. When multiple elements use the same function as event handler, multiple copies of the original “doSomething” method is made which individually keeps track of its “this” keyword(the element reslectively).

BUT when we use inline event registration like :- <element onclick="doSomething()">

Then it does not copy the function rather just references the original “doSomething” function and “this” here refers to the global window object.

BUT this problem can be solved by passing “this” in arguments :

<element onclick="***doSomething(this)">***

7. For W3C browsers :  
 function doSomething(e) {

alert(***e.type***);

}

For IE :  
 function doSomething() {

alert(***window.event.type***);

}

8.

**Cross-Browser**

1. Event Registration:

FOR W3C : Element.addEventListener(‘click’,doSomething,false);

FOR IE<=8 : ***Element.attachEvent(‘onclick’,doSomething);***

2. For IE<=8 ***window.event.cancelBubble = true***

For Others : ***e.stopPropagation();***

3. For W3C : ***Event.currentTarget*** -> returns element that currently handling the event.

FOR IE<=8 :

4. Drawbacks of Microsoft Event Model :  
 a.) Events always bubble, no capturing possibility.

b.) The event handling function is referenced, not copied, so the this keyword always refers to the window and is completely useless.

5. For W3C : ***‘this’ keyword*** refers to the element currently handling the event. While in IE “this” refers to the global window object.

6. Crossbrowser usage of Event object :

element.onclick = doSomething;

function doSomething(e) {

if (!e) var e = window.event;

// e gives access to the event in all browsers

}

For Inline Events :

<pre onclick="doSomething(event)">

function doSomething(e) {

alert(e.type);

}

7. Cross-Browser implementation of target and srcElement :  
 function doSomething(e) {

var targ;

if (!e) var e = window.event;

if (e.target) targ = e.target;

else if (e.srcElement) targ = e.srcElement;

if (targ.nodeType == 3) // defeat Safari bug –> nodeType=3 is textNode

targ = targ.parentNode;

}

8.

**Optimization**

1. Anonymous functions should not be given as event handlers as they cannot be removed afterwards whenever required.

Ele.addEventListener(“click”, function(){-----}, false); -> Wrong Method

**JAVASCRIPT – MOUSE EVENTS**

1. ***MOUSEDOWN***, user presses the mouse button on this element.

***MOUSEUP***, user releases the mouse button on this element.

***CLICK*** , one mousedown and one mouseup detected on this element.

1. Using “mousedown” event is better than using “click” event
2. Do not register both an onclick and an ondblclick event handler on the same HTML element.

**CASCADING STYLE SHEET (CSS)**

1. "border-collapse" css property in html table.

***border-collapse:collapse;***

***border-collapse:seperate;***

**JAVASCRIPT - COMMON**

1. Javascript Window Object does not have style object associated to it.