Eman Fathi

Handling Events

Events

DOM Events are sent to **notify code** of interesting things that have taken place.

JavaScript can **respond to Events**.

Events can represent everything from basic **user interactions** to automated **notifications** of things happening.

Events

Network Events	
Event Name	**Fired When**
online	The browser has gained access to the network.
offline	The browser has lost access to the network.
Focus Events	
Event Name	**Fired When**
focus	An element has received focus (does not bubble).
blur	An element has lost focus (does not bubble).

Events

Resource Events	
Event Name	**Fired When**
cached	The resources listed in the manifest have been downloaded, and the application is now cached.
error	A resource failed to load.
abort	The loading of a resource has been aborted.
load	A resource and its dependent resources have finished loading.
beforeunload	The window, the document and its resources are about to be unloaded.
unload	The document or a dependent resource is being unloaded.

Events

Network Events	
Event Name	**Fired When**
open	A WebSocket connection has been established.
message	A message is received through a WebSocket.
error	A WebSocket connection has been closed with prejudice (some data couldn't be sent for example).
close	A WebSocket connection has been closed.
Form Events	
Event Name	**Fired When**
reset	The reset button is pressed
submit	The submit button is pressed

Events

View Events	
Event Name	**Fired When**
fullscreenchange	An element was turned to fullscreen mode or back to normal mode.
fullscreenerror	It was impossible to switch to fullscreen mode for technical reasons or because the permission was denied.
resize	The document view has been resized.
scroll	The document view or an element has been scrolled.
Clipboard Events	
Event Name	**Fired When**
cut	The selection has been cut and copied to the clipboard
copy	The selection has been copied to the clipboard
paste	The item from the clipboard has been pasted

Events

Mouse Events	
Event Name	**Fired When**
mouseenter	A pointing device is moved onto the element that has the listener attached.
mouseover	A pointing device is moved onto the element that has the listener attached or onto one of its children.
mousemove	A pointing device is moved over an element. (Fired continously as the mouse moves.)
mousedown	A pointing device button (ANY button) is pressed on an element.
mouseup	A pointing device button (ANY button) is released over an element.
click	A pointing device button (ANY button) has been pressed and released on an element.
dblclick	A pointing device button is clicked twice on an element.

Events

Mouse Events	
Event Name	**Fired When**
contextmenu	The right button of the mouse clicked (before context menu is displayed).
wheel	A wheel button of a pointing device is rotated in any direction.
mouseleave	A pointing device is moved off the element that has the listener attached.
mouseout	A pointing device is moved off the element that has the listener attached or off one of its children.
select	Some text is being selected.

Events

Keyboard Events	
Event Name	**Fired When**
keydown	ANY key is pressed
keypress	ANY key except Shift, Fn, CapsLock is in pressed position. (Fired continously.)
keyup	ANY key is released

Events

Drag & Drop Events	
Event Name	**Fired When**
dragstart	The user starts dragging an element or text selection.
drag	An element or text selection is being dragged (Fired continuously every 350ms).
dragend	A drag operation is being ended (by releasing a mouse button or hitting the escape key).
dragenter	A dragged element or text selection enters a valid drop target.
dragover	An element or text selection is being dragged over a valid drop target. (Fired continuously every 350ms.)
dragleave	A dragged element or text selection leaves a valid drop target.
drop	An element is dropped on a valid drop target.

Events

- Media Events
- Progress Events
- Storage events
- Update events
- SVG events
- Database events
- Notification events
- CSS events
- Script events
- Window events
- Document events
- Popup events
- Tab events
- Battery events
- Call events
- Sensor events
- Smartcard events
- SMS and USSD events
- Touch events

Setting Event Handler Properties

The simplest way to register an event handler is by setting a property of the event target to the desired event handler function.

event handler properties have names that consist of the word "**on**" followed by the event name: onclick, onchange, onload, onmouseover, and so on. Note that these property names are case sensitive and are written in all lowercase.

```
//Pass a function reference – do not add '()' after it, which would call the function!
el.onclick = modifyText;


//Using a function expression
element.onclick = function(event) {
     ... function logic ...
};
```

Setting Event Handler Properties

```
//Pass a function reference – do not add '()' after it, which would call the function!
el.onclick = modifyText;
//Using a function expression
element.onclick = function(event) {
     ... function logic ...
};
```

- **The single argument passed to the specified event handler function is a Event object.**
- **Within the handler, this will be the element upon which the event was triggered.**

This method replaces the existing click event listener(s) on the element if there are any.

Setting Event Handler Attributes

The event handler properties of a document element can also be set as attributes on the corresponding HTML tag. If you do this, the attribute value should be a string of JavaScript code.

```
<button onclick="displayDate()">The time is?</button>
```

Event Handler Argument

The **event object** is always **passed to the handler** and contains a lot of useful **information** what has happened.

- event.type - type of the event, like “click”
- event.target - the reference to clicked element.
- event.clientX / event.clientY - coordinates of the pointer at the moment of click.
- … Information about which button was clicked and other properties. We’ll cover them in details later.

- Now our aim is to get the event object. There are two ways.

Event Handler Argument

```
function myhandler(event) {
    //...
}
element.onclick = myhandler;

element.onclick = function (event) {
    // process data from event
    event.type;  // type of the event, like "click"
    event.target; // the reference to clicked element.
    event.clientX;
    event.clientY; //coordinates of the pointer at the moment of click.
}
```

Event

The Event interface represents any event of the DOM. It contains common properties and methods to any event.

Property	Description
bubbles	Read only A boolean indicating whether the event bubbles up through the DOM or not.
cancelBubble	A nonstandard alternative to Event.stopPropagation().
cancelable	Read only A boolean indicating whether the event is cancelable.
currentTarget	Read only A reference to the currently registered target for the event.
deepPath	Returns an Array of nodes through which the event bubbled.
defaultPrevented	Read only Indicates whether or not event.preventDefault() has been called on the event.

Event

Property	Description
eventPhase	Read only Indicates which phase of the event flow is being processed.
returnValue	A nonstandard alternative to Event.preventDefault() (IE)
srcElement	A nonstandard alias for Event.target. (old Internet Explorer-specific)
target	Read only A reference to the target to which the event was originally dispatched.
timestamp	Read only The time that the event was created.
type	Read only The name of the event (case-insensitive).
isTrusted	Read only Indicates whether or not the event was initiated by the browser (after a user click for instance) or by a script (using an event creation method, like event.initEvent)

Event

Property	Description
initEvent()	Initializes the value of an Event created. If the event has already being dispatched, this method does nothing.
preventBubble()	Obsolete Prevents the event from bubbling. use event.stopPropagation instead.
preventDefault()	Cancels the event (if it is cancelable).
stopPropagation()	Stops the propagation of events further along in the DOM.

MouseEvent Object

Property	Description
altKey	Returns whether the "ALT" key was pressed
button	Returns which mouse button was pressed
buttons	Returns which mouse buttons were pressed
clientX	Returns the horizontal coordinate of the mouse pointer, relative to the current window
clientY	Returns the vertical coordinate of the mouse pointer, relative to the current window
ctrlKey	Returns whether the "CTRL" key was pressed
Detail	Returns a number that indicates how many times the mouse was clicked

MouseEvent Object

Property	Description
pageX	Returns the horizontal coordinate of the mouse pointer, relative to the document
pageY	Returns the vertical coordinate of the mouse pointer, relative to the document
relatedTarget	Returns the element related to the element that triggered the mouse event
screenX	Returns the horizontal coordinate of the mouse pointer, relative to the screen
screenY	Returns the vertical coordinate of the mouse pointer, relative to the screen
shiftKey	Returns whether the "SHIFT" key was pressed when an event was triggered
Which	Returns which mouse button was pressed

KeyboardEvent Object

Property	Description
altKey	Returns whether the "ALT" key was pressed
ctrlKey	Returns whether the "CTRL" key was pressed
charCode	Returns the Unicode charcode of the key that triggered onkeypress event
key	Returns the key value of the key represented by the event
keyCode	Returns the Unicode character code of the key that triggered the onkeypress event, or the Unicode key code of the key that triggered the onkeydown or onkeyup event
location	Returns the location of a key on the keyboard or device
metaKey	Returns whether the "meta" key was pressed
shiftKey	Returns whether the "SHIFT" key was pressed when the key event was triggered
which	Returns the Unicode character code of the key that triggered the onkeypress event, or onkeydown or onkeyup event

onchange

```
<label>Choose an ice cream flavor: </label>
<select size="1" onchange="changeEventHandler(event);">
    <option>chocolate</option>
    <option>strawberry</option>
    <option>vanilla</option>
</select>

function changeEventHandler(event) {
    alert('You like ' + event.target.value + ' ice cream.');
}
```

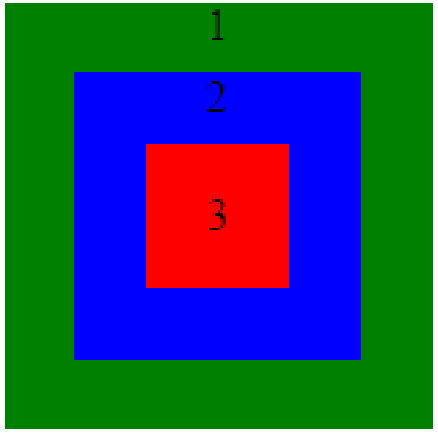
Event Bubbling

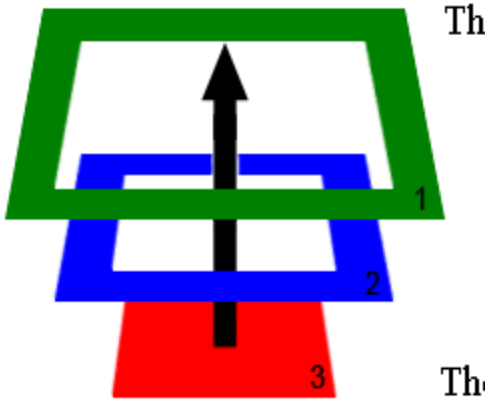
DOM elements can be nested inside each other. And somehow, the handler of the parent works even if you click on it's child.

```
<div onclick="alert('Div handler worked!')">
    <em>Click here triggers on nested <code>EM</code>, not on <code>DIV</code></em>
</div>
```

That's because an event *bubbles* from the nested tag up and triggers the parent.

After an event triggers on the deepest possible element, it then triggers on parents in nesting order.





this and event.target

The deepest element which triggered the event is called the **target** or, the originating element.

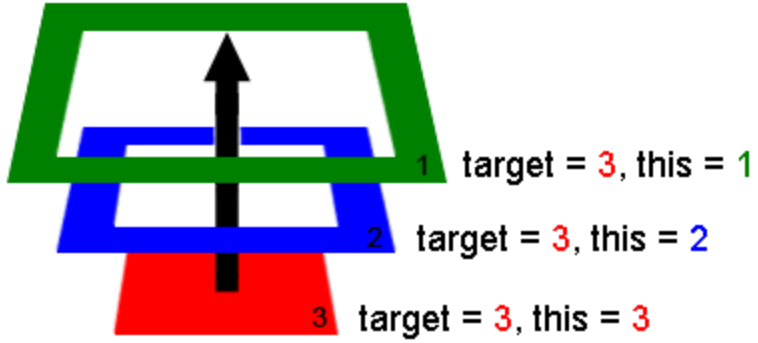
Internet Explorer has the **srcElement** property for it, all W3C-compliant browsers use event.target. The cross-browser code is usually like this:

```
var target = event.target || event.srcElement;
```

When handlers trigger on parents:

event.target/srcElement - remains the same originating element.

this - is the current element, the one event has bubbled to, the one which runs the handler.



JS

Stopping the bubbling

The bubbling goes right to the top. When an event occurs on an element - it will bubble up to <HTML>, triggering handlers on it's way.

But a handler may decide that event is fully processed and stop the bubbling.

```
element.onclick = function(event) {
    event = event || window.event        cross-browser event

    if (event.stopPropagation) {
        event.stopPropagation();         W3C standard variant
    } else {
        event.cancelBubble = true;       IE variant
    }
}
```

Event Cancellation

```
<body>
    This example cancels the onclick event for the checkbox <br />
    and the onchange event for the selection list below.<br />
    The onclick event is cancelable, the onchange is not.
    <input type="checkbox" onclick="return false;" />
    Try to check this checkbox.
    <br /><br />
    Select an item from the following selection list.
    <select onchange="return false;">
        <option>First option</option>
        <option>Second option</option>
    </select>
    <a href="http://www.google.com" onclick="return false;">Google</a>
</body>
```

preventDefault()

Cancels the event if it is cancelable, without stopping further propagation of the event.

```
function stopDefAction(evt) {
    evt.preventDefault();
}
document.getElementById('my-checkbox').addEventListener( 'click', stopDefAction, false );
```

preventDefault()

Write Script to prevent the user to enter numbers in the textbox

```
<input type="text" id="username" maxlength="10" />

var txtUser = document.getElementById('username');
txtUser.addEventListener('keypress', function (evt) {
    var charCode = evt.charCode;
    if (charCode <= 48 && charCode >= 57) {
            evt.preventDefault();
    }
}, false);
```

this

```
element.onclick = function () {
    this.style.color = 'red';
}
```

When a function is used as an event handler, its **this** is set to the **element caused the event to fire**.

this

```
<img src="1.jpg" width="300" height="300"
        onmouseover="this.src ='2.jpg'"
        onmouseout="this.src='1.jpg'" />
```

When code is called from an in–line on-event handler, its **this** is set to the DOM **element** on which the listener is placed.

Document Load Events

window.onload

- By default, it is fired when the entire page loads, including its content (images, css, scripts, etc.)
- In some browsers it now takes over the role of document.onload and fires when the DOM is ready as well.

```
window.onload = function(e){
    console.log("window.onload", e, Date.now());
}
```

dcument.onload

- It is called when the DOM is ready which can be prior to images and other external content is loaded.

```
window.document.onload = function(e){
    console.log("document.onload", e, Date.now());
}
```

EventTarget.addEventListener

registers the specified listener on the EventTarget it's called on. The event target may be an Element in a document, the Document itself, a Window, or any other object that supports events.

Syntax:

target. AddEventListener(type, listener[, useCapture]);

- **type** A string representing the event type to listen for.
- **listener** The object that receives a notification (an object that implements the Event interface) or simply a JavaScript **function**.
- **useCapture Optional** A Boolean that indicates that events of this type will be dispatched to the registered listener before being dispatched to any Event Target beneath it in the DOM tree. Events that are bubbling upward through the tree will not trigger a listener designated to use capture. useCapture defaults to **false**.

addEventListener

The addEventListener() method attaches an event handler to the specified element.

The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.

You can add many event handlers to one element.

```
var btn = document.getElementById("btn");
btn.addEventListener("click", myFunction);
btn.addEventListener("click", mySecondFunction);
```

You can add many event handlers of the same type to one element, i.e two "click" events.

```
btn.addEventListener("mouseover", myFunction);
btn.addEventListener("mouseout", myThirdFunction);
```

addEventListener

You can add event listeners to any DOM object not only HTML elements. i.e the window object.

```
window.addEventListener("load", function(event) {
    console.log("All resources finished loading!");
});
```

The addEventListener() method makes it easier to control how the event reacts to bubbling.

```
document.getElementById("myDiv").addEventListener("click", myFunction, true);
```

removeEventListener

You can easily remove an event listener by using the removeEventListener() method.

```
btn.removeEventListener("mousemove", myFunction);
```

Firing Event

```
<input type="checkbox" id="myCheck" onmouseover="myFunction()" onclick="alert('clicked')">


function myFunction() {
    document.getElementById("myCheck").click(); // Click on the checkbox
}
```

- The click() method simulates a mouse-click on an element.
- This method can be used to execute a click on an element as if the user manually clicked on it.
- When click() is used with supported elements (e.g. one of the <input> types), it fires the element's click event.
- One exception: The click() method will not cause an <a> element to initiate navigation as if a real mouse-click had been received.

Event Delegation

Attach an event to a parent item or document and check in the parent if trigger is your element. You can use a class name or tag name etc.

```
// Get the element, add a click listener...
document.getElementById("parent-list").addEventListener("click",function(e) {
    // e.target is the clicked element! // If it was a list item
    if(e.target && e.target.nodeName == "LI") {
        // List item found!  Output the ID!
        console.log("List item ",e.target.id.replace("post-")," was clicked!");
    }
});
```

in this way the event also works for elements what will be **added afterwards**.

Thank You