Events

Chapter 3
Objectives

• Learn how to work with events
Understanding Events

• An Event is a specific circumstance that is monitored by JavaScript, that the script can respond to in some way
• Events allow users to interact with your web pages
• Most common events are user actions, e.g. click
# Understanding Events...

<table>
<thead>
<tr>
<th>Event</th>
<th>Triggered when</th>
</tr>
</thead>
<tbody>
<tr>
<td>abort</td>
<td>The loading of an image is interrupted</td>
</tr>
<tr>
<td>blur</td>
<td>An element, such as a radio button, becomes inactive</td>
</tr>
<tr>
<td>click</td>
<td>The user clicks an element once</td>
</tr>
<tr>
<td>change</td>
<td>The value of an element, such as text box, changes</td>
</tr>
<tr>
<td>error</td>
<td>An error occurs when a document or image is being loaded</td>
</tr>
<tr>
<td>focus</td>
<td>An element, such as a command button, becomes active</td>
</tr>
<tr>
<td>load</td>
<td>A document or image loads</td>
</tr>
<tr>
<td>mouseout</td>
<td>The mouse moves off an element</td>
</tr>
<tr>
<td>mouseover</td>
<td>The mouse moves over an element</td>
</tr>
<tr>
<td>reset</td>
<td>A form’s fields are reset to its default values</td>
</tr>
<tr>
<td>select</td>
<td>A user selects a field in a form</td>
</tr>
<tr>
<td>submit</td>
<td>A user submits a form</td>
</tr>
<tr>
<td>unload</td>
<td>A document unloads</td>
</tr>
</tbody>
</table>

*Table 3-2* JavaScript events
Elements and Events

• Events are associated with XHTML tags, but those available to a tag vary

• Event handler
  – Code that executes in response to a specific event
  – Included as an attribute of the tag that initiates the event
Event Handler

• Syntax

\(<\text{element event_handler} = "JavaScript \text{code}">\)

• No need for \(<\text{script}>\)/\(<\text{script}>\) elements

• Event handler names are the same as the name of the event itself, plus a prefix of on

\(<\text{body on\text{load}=...}\)
# Elements and their Events

<table>
<thead>
<tr>
<th>&lt;a&gt; Anchor</th>
<th>onfocus, onblur, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;body&gt; Document body</td>
<td>onload, onunload, onclick, etc.</td>
</tr>
</tbody>
</table>
## Elements and their Events

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;a&gt;</code></td>
<td>Anchor</td>
<td><code>onfocus, onblur, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup</code></td>
</tr>
<tr>
<td><code>&lt;img&gt;</code></td>
<td>Image</td>
<td><code>onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup</code></td>
</tr>
<tr>
<td><code>&lt;body&gt;</code></td>
<td>Document body</td>
<td><code>onload, onunload, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup</code></td>
</tr>
<tr>
<td><code>&lt;form&gt;</code></td>
<td>Form</td>
<td><code>onsubmit, onreset, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup</code></td>
</tr>
<tr>
<td><code>&lt;input&gt;</code></td>
<td>Form control</td>
<td><code>tabindex, accesskey, onfocus, onblur, onselect, onchange, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup</code></td>
</tr>
<tr>
<td><code>&lt;textarea&gt;</code></td>
<td>Text area</td>
<td><code>onfocus, onblur, onselect, onchange, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup</code></td>
</tr>
<tr>
<td><code>&lt;select&gt;</code></td>
<td>Selection</td>
<td><code>onfocus, onblur, onchange</code></td>
</tr>
</tbody>
</table>

*Table 3-3*  XMLElement elements and their associated events
**click and dblclick events**

- Trigger event when clicking the left mouse button
- You can override an anchor tag’s automatic `onclick` event handler
  - Add to the `<a>` tag an `onclick` event handler that executes custom code
- The `dblclick` event works like the `click` event
Example 1

<input type="button" value="Press Me" onclick="window.alert('You clicked a button!')"/>

• `window.alert()` method displays a pop-up dialog box with an **OK** button

• You can include multiple JavaScript statements in an event handler, as long as semicolons separate the statements
Example 2

<a href="http://www.google.co.uk" onclick="return window.confirm('Are you sure?')">Click me</a>

- `window.confirm()` displays a pop-up dialog box with an **OK** and **Cancel** button and returns `true` when OK is pressed or `false` when Cancel is pressed.

- **links** already have an **onclick** event, but can be **overridden**. Link is only loaded if JavaScript program returns `true`. 
Example 3

```javascript
var count=0;
function increaseCounter() {
    count++;
    document.getElementById("counter").innerHTML="You've pressed the button "+count+" times";
}

<h1 id="counter">Please press the button</h1>
<input type="button" value="Click Me" onclick="increaseCounter()">
```
mouseover and mouseout

- Rollover occurs when your mouse moves over an element and is captured by `mouseover` event
- `mouseout` event occurs when the mouse moves off an element
- One common use is to change the text that appears in a browser `status bar`
Example 4

```html
<img src="slide0.jpg"
    onmouseover="this.setAttribute('src','slide1.jpg')"
    onmouseout="this.setAttribute('src','slide0.jpg')">

Note use of this. Alternative:

```html
<img src="slide0.jpg"
    onmouseover="this.src='slide1.jpg'
    onmouseout="this.src='slide0.jpg'">
```
mousedown/ mouseup events

• mousedown event occurs when you point to an element and hold the mouse button down

• mouseup event occurs when you release the mouse button
Example 5

```html
<img src="slide0.jpg"
onmousedown="this.src='slide1.jpg')"
onmouseup="this.src='slide0.jpg')">
```
Timeouts

• window object’s timeout and interval methods executes automatically
• setTimeout() method
  – Executes code after a specific amount of time
  – Executes only once

```javascript
var variable = setTimeout("code", milliseconds);
```
When used?

• Redirect users to a different page
  – "you will be redirected in 5 seconds"

• Give limited time to fill in form
Timeouts...

clearTimeout() method

  – Cancel a setTimeout() before its code executes

• Example

```javascript
var clock1 =
  setTimeout("window.alert('Time up!')"), 5000);
```
Intervals

`setInterval("code", ms) method`
- Repeatedly executes the same code after being called only once

`clearInterval() method`
- Used to clear a `setInterval()` method call

• Interval methods are most often used for starting animation code
Example: Clock

```javascript
var clockInterval;
function startClock() {
    clockInterval = window.setInterval("updateClock()",50);
}
function updateClock() {
    var t = new Date();
    document.getElementById("clock").innerHTML=t;
}
function stopClock() {
    window.clearInterval(clockInterval);
}

<body onload="startClock()">
    <h1 id="clock">
    </h1>
    <input type="button" value="Stop Clock" onclick="stopClock()"/>
</body>
```
Example: redirect

```javascript
var count;
function startCounter(numSeconds) {
    count = numSeconds;
    window.setInterval("countDown()",1000);
}
function countDown() {
    count--;
    document.getElementById("counterID").innerHTML=count;
    if (count <= 0)
        window.location.href="http://www.google.co.uk";
}
<body onload="startCounter(10)">
    <h1>You will be redirected in <strong id="counterID">10</strong> seconds</h1>
</body>
```
Summary

• An event is a specific circumstance that is monitored by JavaScript and that your script can respond to in some way

• Code that executes in response to a specific event is called an event handler