Internet Software Technologies
Dynamic HTML – part two

IMCNE

A.A. 2008/09

Gabriele Cecchetti

Events: another simple example

- Every element on a web page has certain events which can trigger JavaScript functions.
  For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button:

```
<html><head><title>Event Handlers</title>
<script>
function ciao() {
  window.alert(“You clicked button!");
}
</script></head>
<body>
<button onClick="ciao()"> Click Here ! </button>
</body></html>
```

Examples of actions which can trigger JavaScript functions

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input box in an HTML form
- Submitting an HTML form
- A keystroke

- **Note:** the triggered function will not be executed before the event occurs!

Main event handlers (1/2)

Event handler	Description
onAbort	abort loading an <img>
onFocus	acquire window focus or form's element focus
onBlur	loss form's element focus
onChange	change <input> and <textarea> content)
onClick	click on form's elements or on link
onDblClick	double click on same elements
onKeyDown	Key down
onKeyUp	Release a key
onKeyPress	Press and hold a key
onLoad	page loaded
onUnload	page unloaded

Main event handlers (2/2)

Event handler	Description
onMouseMove	cursor moving
onMouseOut	cursor going out from map, link or area
onMouseOver	cursor over map, link or area
onMouseDown	mouse button pressed and cursor over a link
onMouseUp	mouse button released and cursor over a link
onMove	window moving
onResize	window resizing
onSubmit	form submitting – submit button pressed
onReset	form resetting – reset button pressed
onselect	selecting content on a page

Mouse and Keyboards attributes

Property	Description
altKey	Returns whether or not the "ALT" key was pressed when an event was triggered
button	Returns which mouse button was clicked when an event was triggered
clientX	Returns the horizontal coordinate of the mouse pointer when an event was triggered
clientY	Returns the vertical coordinate of the mouse pointer when an event was triggered
ctrlKey	Returns whether or not the "CTRL" key was pressed when an event was triggered
metaKey	Returns whether or not the "meta" key was pressed when an event was triggered
relatedTarget	Returns the element related to the element that triggered the event
screenX	Returns the horizontal coordinate of the mouse pointer when an event was triggered
screenY	Returns the vertical coordinate of the mouse pointer when an event was triggered
shiftKey	Returns whether or not the "SHIFT" key was pressed when an event was triggered

Other event attributes

Property	Description
bubbles	Returns a Boolean value that indicates whether or not an event is a bubbling event
cancelable	Returns a Boolean value that indicates whether or not an event can have its default action prevented
currentTarget	Returns the element whose event listeners triggered the event
eventPhase	Returns which phase of the event flow is currently being evaluated
target	Returns the element that triggered the event
timeStamp	Returns the time stamp, in milliseconds, from the epoch (system start or event trigger)
type	Returns the name of the event

Example: what is the unicode of the key pressed?

```
<html><head>
<script type="text/javascript">
function whichButton(event)
{
alert(event.keyCode);
}
</script></head>
<body onkeyup="whichButton(event)">
<p>Press a key on your keyboard.<br>
  An alert box will alert the unicode of the key
   pressed.</p>
</body></html>
```

Example: what are the coordinates of the cursor?

```
<html><head>
<script type="text/javascript">
function show_coords(event)
{
x=event.clientX;
y=event.clientY;
alert("X coords: " + x + ", Y coords: " + y);
}
</script>
</head>
<body onmousedown="show_coords(event)">
<p>Click in the document. An alert box will alert the x
  and y coordinates of the mouse pointer.</p>
</body></html>
```


Example: which mouse button was clicked ?

```
<html><head>
<script type="text/javascript">
function whichButton(event)
{
if (event.button==2)
  alert("You clicked the right mouse button!");
else
  alert("You clicked the left mouse button!");
}
</script></head>
<body onmousedown="whichButton(event)">
<p>Click in the document. An alert box will alert which
  mouse button you clicked.</p>
</body></html>
```

Adding events via scripting

- This is an alternative technique to classic event handler.
- You can assign and set up event handlers to elements using scripting, and inside your script.
- This allows for the event handlers to be dynamically set up, without having to mess around with the HTML codes on the page.
- When setting up event handlers for an element directly inside your script, the code to execute for the events must be defined inside a function.

Example of event handler via scripting

```
<a id="test" href=http://retis.sssup.it>
   retis.sssup.it</a>
<script type="text/javascript">
function changestatus(){
   window.status="Click here for RETIS site" return true
}
function changebackstatus() {
   window.status=''
}
document.getElementById("test").onmouseover=changestatus
document.getElementById("test").onmouseout=changebackstatus
</script>
```

Notice how we attached the two functions to execute for the two events- the function names without the **parenthesis**. *This is called a reference call to the function*.\When assigning a function to an event via scripting, always use a function call. If you were to include the parenthesis of the function inside the definition, an error will be generated.

The DOM event flow

- The DOM introduces a new concept for detecting events and assigning corresponding event handlers to react to them.
- Naturally, it also supports the 2 conventional techniques discussed earlier.

How events are handled in the DOM ?

- The DOM interprets all user action very differently from the user . Example: moving the mouse over a link: to the DOM the following events took place
  1) the mouse was moved over the document
  2) the mouse was moved over any tags containing the target <a> tag
  3) the mouse was moved over the <a> tag of the link in question
  4) the mouse was moved over any tags containing the target <a> tag
  5) the mouse was moved over the document.
- Summary: Mouse over document → Mouse over any containment tags of <A> → **Mouse over destination <A>** → Mouse over any containment tags of <A> → Mouse over document
- We call **the event** as it travels to the intended element ***event capture***, **and as it travels back** ***event bubble***.

Event capture and Event bubble

- Event capture → it basically refers to the process of an event as it travels to its destination element. Actually, it further refers to the ability to "capture", or intercept this event.
- Event bubble → it is the exact opposite of event capture, and points to the traveling upwards of an event from the source element to the topmost, or document.

Assigning event handlers in the DOM (1/2)

- In light of the new event flow (*event capture* and *event bubble*) that occurs inside the DOM, new methods for assigning event handlers are introduced.
- A concept for attaching event handlers is introduced- *event listeners.* Two methods:

```
object.addEventListener(event, function, capture)
object.removeEventListener(event, function, capture)
```

where the first method assigns an event handler, while the second allows you to then remove it.

Assigning event handlers in the DOM (2/2)

```
object.addEventListener(event, function, capture)
object.removeEventListener(event, function, capture)
```

where:

event should contain the event you wish to detect, such as **click** (onclick minus the "on" prefix)

function should contain the reference call to the function to execute for this event,
such as **dothis** (dothis() minus the parenthesis)

capture should contain the Boolean value "**true**" or "**false**";

- a value of "true" causes function to be executed when the event is detected at the *capture* phase.
- a value of "false" causes function to be executed when event is at *bubble* phase.

Example of using addEventListener

```
<div id="test"> Some text over div </div>
<script type="text/javascript">
function alertit(){
  alert("You moved your mouse over me!")
}
document.getElementById("test").addEventListener("mouse
  over",alertit,false)
</script>
```

- Moving your mouse over the DIV now will cause an alert message to popup.

Multiple addEventListener()

- It can be used multiple times to attach multiple functions to the *same* event for the same element.
- For example, we can have 3 separate onMouseover events for the previous DIV example:

```
document.getElementById("test").addEventListener("mouse
  over",alertthis,false)
document.getElementById("test").addEventListener("mouse
  over",alertdat,false)
document.getElementById("test").addEventListener("mouse
  over",alerttidat,false)
```

- And all three will be responded to onMouseover.
- Note that two scripts can both apply the same event handler to an element without the former being cancelled out.

References to event specification

- Look for them in Internet.
- Pay attention to the different browser implementations.

HTML DOM style specification

- The Style object represents an individual style statement.
- The Style object can be accessed from the document or from the elements to which that style is applied.
- **Syntax**:

```
document.getElementById("id").style.property="value"
```

- The *property* name *usually* is the style property as defined in the CSS specification.

Example: changing the case letter on the fly

```
<html>
<head>
<script type="text/javascript">
function changeText() {
  document.getElementById("p1").style.textTransform="ca
  pitalize";
}
</script>
</head>
<body>
<p id="p1">This is an example paragraph.</p>
<input type="button" onclick="changeText()"
  value="Convert first letter of each word" />
</body></html>
```