

Lecture Notes 6

JavaScript Event Handling

Model

Anand Tripathi

CSci 4131

Internet Programming

JavaScript Event Handling

Form Objects

A form object has the following objects in its elements[] array.

- **Buttons**
- **Checkboxes**
- **Hidden**
- **Password**
- **Reset**
- **Select**
- **Submit**
- **Text**
- **Textarea**

Events

Event	Occurs	For Objects
onclick	Mouse click by the user. Handler can return <u>false</u> to cancel the default action.	Links, buttons, elements
onload	Window document's body or image loading is completed.	Window, image
onmouseover	Mouse moves an element	Link, image, element
onmouseout	Mouse moves off an element.	Link, image, element
onreset	Form reset button is clicked	Form
onsubmit	Form submit button is clicked	Form

Events

Event	Occurs	For Objects
onabort	Image loading aborted	Image
onblur	Element loses focus	input, label, select, textarea, body
onfocus	Element loses focus	input, label, select, textarea, body
onchange	value changed	input, select, textarea

Forms Object

- `document.forms`
 - Represents an array containing all forms object in the document
- `document.forms[0]` is the first form
- `document.forms[document.forms.length-1]`
 - is the last form
- Event Handlers are associated with the forms object as well as with the elements of the form

Event Handler for Form

- These are defined in the FORM tag.
- onSubmit
 - It is executed when the form is submitted.
 - If it returns false, the submission will NOT be made
- onReset
 - This is executed when the user tries to reset a form.

Example: verifying form data

- This example has one text-field where the user can enter a number.
- The JavaScript code will check if the age looks reasonable:
 - It should be a number
 - In the range 0 -- 150
- The example code is an incomplete solution!
 - Why?
- [Click to see execution of the example code here.](#)

Example code: form verification

```
<script type="text/JavaScript">
```

```
function checkInputAge ( ) {  
    if ( isNaN( parseInt( document.myForm.age.value ) ) ) {  
        alert("Not a number!");  
    } else {  
        if ( document.myForm.age.value < 0 ) {  
            alert ( "You must be in your past life" );  
        }  
    }  
}
```


Example code: form verification

```
else {  
    if ( document.myForm.age.value > 150 ) {  
        alert( "You seem to be too old to be still alive!" );  
    }  
    else alert ( " Age looks OK ! " );  
}  
}  
}  
</script>
```

Example code: form verification

```
<body>
<form name="myForm">
AGE: <input type="text" name="age" value="20" > <br/>
<input type="button" name="checkAge" value="Click to Check Age"
  onClick="checkInputAge()" > <br/>
</form>
</body>
```

To do proper testing of the age to avoid "21B" to be taken as a valid age, we would need to use `String.search()` function to make sure that it contains only numeric characters 0..9.

Example

```
<form name="myForm" onSubmit="return validateForm();">  
...  
</form>
```

Another way to write this

```
document.myForm.onSubmit = validateForm;
```

```
<form name="myForm"  
  onReset='return confirm("Do you want to clear all data");'>  
...
```

Explicit Invoking of Handler

- Suppose that you want to validate a form before the user clicks the submit button.
- For this, you can explicitly execute the handler associated with onsubmit.
- Example:
- `document.myform.onsubmit()`

Form Elements

The various objects in a form, such as checkboxes, radio buttons, text, text area, buttons are called elements of the form.

```
<form name="survey">
```

```
...
```

```
</form>
```


The elements in this form can be accessed using the `elements[]` array as follows:

```
document.survey.elements[0]
```

```
document.survey.elements[1]
```

Example

```
<form action="register.cgi"
  onsubmit="if (this.element[0].value.length==0)
    return false;">
  <input type="text" name="firstname" />
  </form>
```



Two things to note in this example:

- Use of “**this**”, it refers to the object corresponding to element in which it occurs.
 - Here it refers the “form” object
- Accessing a property of an element, e.g. **this.element[0].value**
 - Here it refers to the “input” element, which is the first element of this form.

Properties of Form Elements

- type** a read-only string for type of the element
- form** a read-only reference to the Form object in which it appears
- name** specified in the HTML document
- value**

Form Elements Event Handlers

<code>onclick</code>	executed when the user clicks the mouse on this element
<code>onchange</code>	When user changes the value represented by the element
<code>onfocus</code>	Triggered when the element receives the input focus
<code>onblur</code>	Triggered when the Form element loses the input focus

Event Handlers as Properties of Objects in a Document

One function can be used as event handler at many places:

Example from Flanagan's book:

```
function confirmLink(){
    return confirm("Do you want to visit "+ this.href +"?");
}
function setConfirm(){
    for (var i = 0; i < document.links.length; i++) {
        document.links[i].onclick = confirmLink;
    }
}
<body onload ="setConfirm()">
.....
```

[Click here to see this cde execution.](#)

Modifying an existing handler

- Suppose that you want to add some additional function to be executed on some event.
- See the example code: (Flanagan's book page 358)

```
var b = document.myform.somebutton;
```

```
// Save old handler
```

```
var oldhandler = b.onclick;
```

```
// assign a new handler that executes the old code first
```

```
// and then executes a newhandler code function
```

```
b.onclick = function () {oldhandler(); newhandler();}
```

Use of “this” in event handler

Two things to note in this example:

- Use of “this” in an event handler
- addEventListener method

In the W3 DOM model, the use of the keyword “this” refers to the element in which the event handler call appears.

```
function changeColor() {  
    this.style.backgroundColor = 'red';  
}  
elementA.addEventListener('click', changeTextColor, false);  
elementB.addEventListener('click', changeTextColor, false);
```

When you click on any of these two elements, that element’s background color changes to red. “this” refers the element in whose context the code is executed.

onFocus, onBlur, onSubmit, onReset (1)

Example 13.7 (Fourth Edition)

This example is from Deitel's book (see online edition, 4th edition or view page-source when example is loaded)

In the Fifth Edition, see example in Section 13.5 (Form Processing with focus and blur) It installs event-handlers using addEventListners.

```
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title>A Form Using onsubmit and onreset</title>
    <style type = "text/css">
      .tip { font-family: sans-serif;
        color: blue;
        font-size: 12px }
    </style>
```

onFocus, onBlur, onSubmit, onReset (2)

Example 13.7

```
<script type = "text/javascript">
```

```
var helpArray =
```

```
  [ "Enter your name in this input box.",
```

```
    "Enter your e-mail address in this input box in the format user@domain.",
```

```
    "Check this box if you liked our site.",
```

```
    "In this box, enter any comments you would like us to read.",
```

```
    "This button submits the form to the server-side script.",
```

```
    "This button clears the form.",
```

```
    "" ];
```

onFocus, onBlur, onSubmit, onReset (3)

Example 13.7

```
function helpText( messageNum )  
  {  
    document.getElementById( "tip" ).innerHTML =  
      helpArray[ messageNum ];  
  } // end function helpText
```

onFocus, onBlur, onSubmit, onReset (4)

Example 13.7

```
function registerEvents()
{
    document.getElementById( "myForm" ).onsubmit = function()
    {
        return confirm( "Are you sure you want to submit?" );
    } // end function

    document.getElementById( "myForm" ).onreset = function()
    {
        return confirm( "Are you sure you want to reset?" );
    } // end function
} // end function registerEvents
</script>
```

Another way to specify event handlers

```
document.getElementById( "myForm" ).addEventListener  
  ( "submit",  
    function() {  
      return confirm( "Are you sure you want to submit?" );  
    },  
    false  
  )
```

If the JavaScript code executed on “submit” **returns false**, then the form is **NOT** submitted.

onFocus, onBlur, onSubmit, onReset

Example 13.7

```
<body onload = "registerEvents()">
  <form id = "myForm" action = "">
    <div>
      Name: <input type = "text" name = "name"
        onfocus = "helpText(0)" onBlur = "helpText(6)" /><br />
      E-mail: <input type = "text" name = "e-mail"
        onfocus = "helpText(1)" onBlur = "helpText(6)" /><br />
      Click here if you like this site
      <input type = "checkbox" name = "like" onfocus =
        "helpText(2)" onBlur = "helpText(6)" /><br /><hr />
```

onFocus, onBlur, onSubmit, onReset

Example 13.7

Any comments?


```
<textarea name = "comments" rows = "5" cols = "45"  
  onfocus = "helpText(3)" onBlur = "helpText(6)"></textarea>
```

```
<br />
```

```
<input type = "submit" value = "Submit" onfocus =  
  "helpText(4)" onBlur = "helpText(6)" />
```

```
<input type = "reset" value = "Reset" onfocus =  
  "helpText(5)" onBlur = "helpText(6)" />
```

```
</div>
```

```
</form>
```

```
<div id = "tip" class = "tip"></div>
```

```
</body>
```

```
</html>
```

Exercise Problem

- Write code for a form which displays two text areas, with default string values as "Apples" and "Oranges", and one button labeled "swap".
- If you click "SWAP", the contents of the two text boxes are interchanged. [Click here to see execution.](#)

Apples	Oranges	SWAP
Oranges	Apples	SWAP

Event Model

- Picture is bit complicated due to different models supported by different browsers:
- See page 351
- 1. **Original Event Model:** Part of HTML 4 standard and supported by DOM (Document Object Model) Level 1.0
- 2. **Standard Event Model:** Standardized in W3C DOM Level 2.0. It is supported by Netscape 6 and Mozilla browsers.
- 3. **Internet Explorer Event Model:** Implemented by IE 4 and later versions. It does not have all features of standard model.

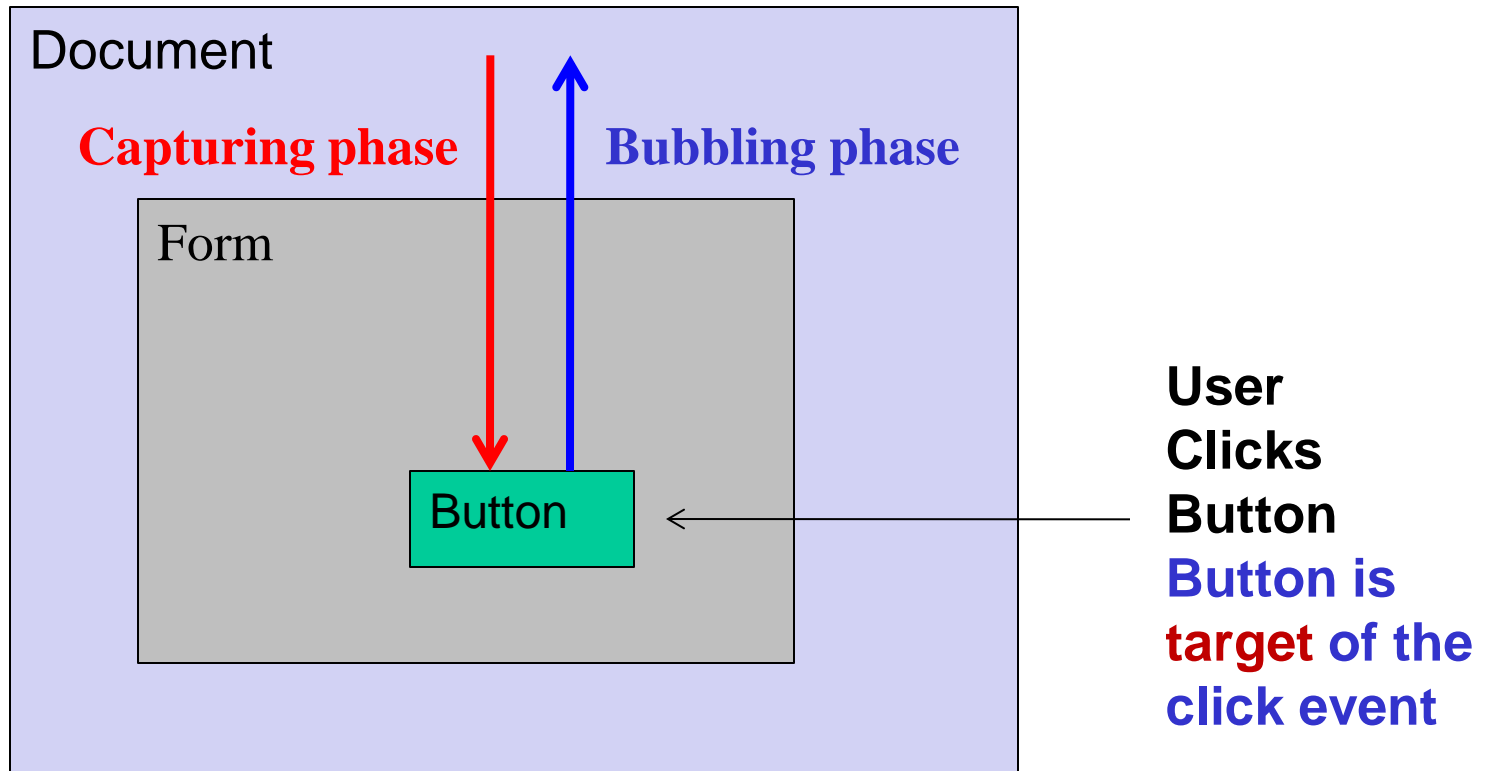
Scope of Event Handler

- JavaScript is lexically (statically) scoped language.
- A function executes in the scope in which it is defined, and not in the scope where it is called.
- The scope chain for a handler is defined with the containment hierarchy for the object where the handler is defined.

Event Propagation Model

- *Target* is the object (node) in a document tree where an event occurs.
- There are 2 phases for event propagation along the components of a document object tree.
 - Capturing phase
 - Bubbling phase

Event Propagation Model



Event Capturing Model

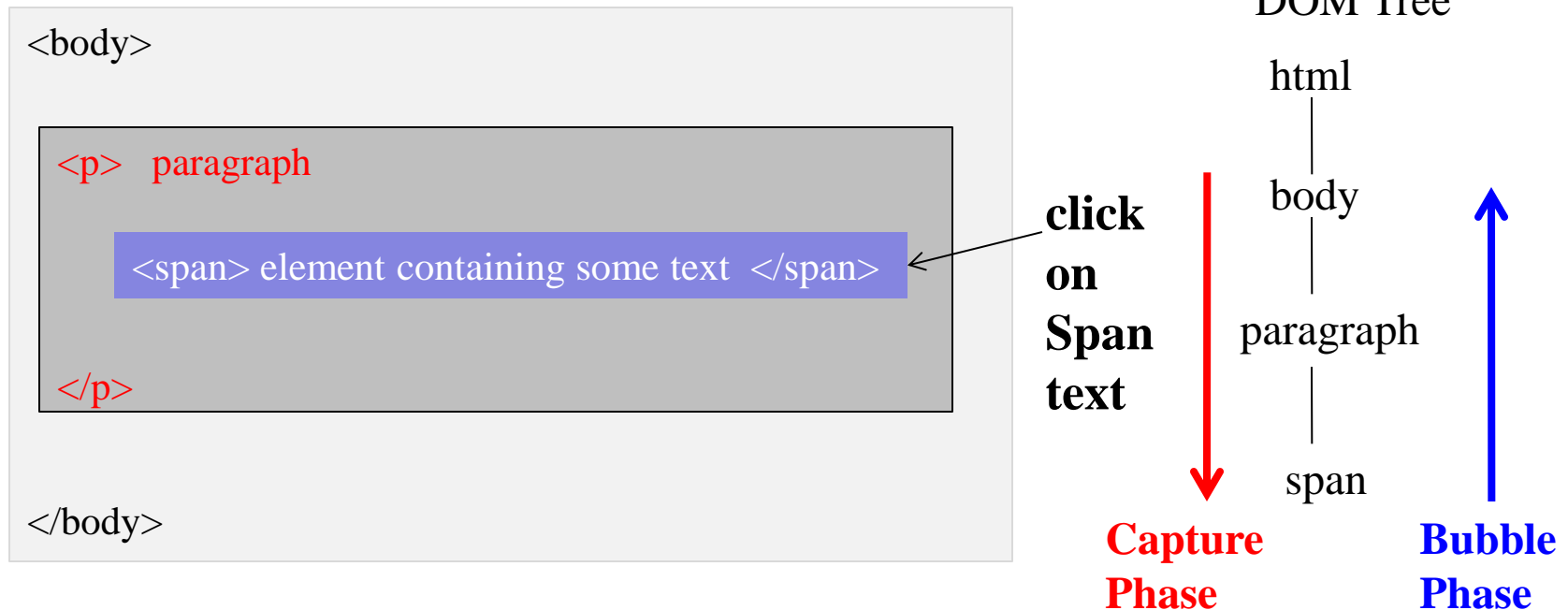
- The recommended method of event handling by W3 DOM 2 standard is to use `addEventListener()` method.

`addEventListener(event string, listener (function or an Object implementing EventListener), useCapture);`

- Where `useCapture` is a **boolean value**
 - “**true**” means that indicates that the event of the specified type (example: "click" event) to be sent to the designated listener by the designated element during the capture phase.
 - “**false**” means that the event would be handled by the listener during the bubble phase.

Example of Bubble and Capture Phase

- In this example body contains one paragraph element, and the paragraph contains one span element



Example 1 – Event Handling in the Bubble Phase

[Click here to see this example](#)

```
<head>
<script language="JavaScript">
  var colors = new Array ("red", "blue", "green", "yellow", "magenta", "cyan");
  var i = 0;
  function changeTextColor( ) {
    var msg = colors[i];    alert( msg );
    this.style.color= colors[i];    i = (i+1) % 4;
  }
  function addListeners() {
    var span1 = document.getElementById("textSpan");
    var para1 = document.getElementById("Para01");
    para1.addEventListener("click", changeTextColor, false);
    span1.addEventListener("click", changeTextColor, false);
    document.body.addEventListener("click", changeTextColor, false);
  }
</script>
</head>
```

Example 1 – Event Handling in the Bubble Phase

```
<body onload= "addListeners()">
```

THIS EXAMPLE WILL WORK ON FIREFOX, BUT NOT ON Microsoft IE.

```
<br/>
```

This example illustrates how addEventListener is used. In this example click event will be processed in the BUBBLE phase.

```
<p id="Para01">
```

This is the first line of the paragraph.

```
<br/>
```

the special variable "this" (click on this line, it is enclosed in a span element)

```
<br/>
```

This last line of the paragraph.

```
</p>
```

When you click on the above text will be changed to red color.

The click event bubbles upward and the entire paragraph's font color is changed to blue. The event will bubble up, and the body text will be changed to green.

```
</body>
```

Example 2 – Event Handling in the Capture Phase

[Click here to see this example](#)

We make the following changes, highlighted in red, to the previous example:

```
function addListeners() {  
  var span1 = document.getElementById("textSpan");  
  var para1 = document.getElementById("Para01");  
  para1.addEventListener("click", changeTextColor, true);  
  span1.addEventListener("click", changeTextColor, true);  
  document.body.addEventListener("click", changeTextColor, true);  
}
```

Example 3 – Event Handling in both Capture and Bubble Phases

[Click here to see this example](#)

```
var colors = new Array ("red", "blue", "green", "yellow", "magenta", "cyan");
```

```
var i = 0;
```

```
function changeTextColor( ) {
```

```
    var msg = colors[i];
```

```
    alert( msg );
```

```
    this.style.color= colors[i];
```

```
    i = (i+1) % 6;
```

```
}
```

```
function addListeners() {
```

```
    var span1 = document.getElementById("textSpan");
```

```
    var para1 = document.getElementById("Para01");
```

```
    para1.addEventListener("click", changeTextColor, false);
```

```
    span1.addEventListener("click", changeTextColor, false);
```

```
    document.body.addEventListener("click", changeTextColor, false);
```

```
    para1.addEventListener("click", changeTextColor, true);
```

```
    span1.addEventListener("click", changeTextColor, true);
```

```
    document.body.addEventListener("click", changeTextColor, true);
```

```
}
```

Event Bubbling

[Click here to see this example](#) **Section 13.7**

```
<html>
<head>
  <meta charset="utf-8">
  <title>Event Bubbling</title>
  <script src = "bubbling.js" </script>
</head>
<body>
  <p id = "bubble">Bubbling enabled.</p>
  <p id = "noBubble">Bubbling disabled.</p>
</html>
</body>
```

Event Bubbling: bubbling.js

```
function documentClick()  
{   alert( "You clicked in the document." );  
} // end function documentClick
```

```
function bubble( e ) {  
    alert( "This will bubble." );  
    e.cancelBubble = false;  
} // end function bubble
```

```
function noBubble( e ) {  
    alert( "This will not bubble." );  
    e.cancelBubble = true;  
} // end function noBubble
```

Event Bubbling: bubbling.js

```
function registerEvents() {  
    // Third argument false means event handler to be  
    // executed in bubble phase  
    document.addEventListener("click", documentClick, false);  
    document.getElementById( "bubble" ).addEventListener(  
        "click" , bubble, false);  
    document.getElementById( "noBubble" ).addEventListener(  
        "click:", noBubble, false);  
} // end function registerEvents
```

```
Window.addEventListener("load", registerEvents, false);
```