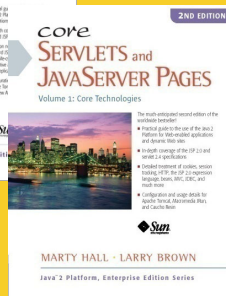
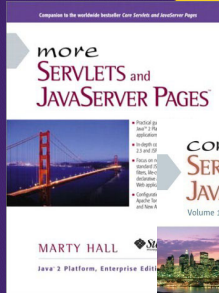




JavaScript: A Crash Course Part II: Browser-Specific Features

Originals of Slides and Source Code for Examples:
<http://courses.coreservlets.com/Course-Materials/ajax.html>

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 1.x & JSF 2.0, Struts Classic & Struts 2, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



For live Ajax & GWT training, see training courses at <http://courses.coreservlets.com/>.



Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
 - Java 6, intermediate/beginning servlets/JSP, advanced servlets/JSP, Struts, JSF 1.x & 2.0, Ajax, GWT, custom mix of topics
 - Ajax courses can concentrate on one library (jQuery, Prototype/Scriptaculous, Ext-JS, Dojo) or survey several
- Courses developed and taught by coreservlets.com experts (edited by Marty)
 - Spring, Hibernate/JPA, EJB3, Ruby/Rails

Contact hall@coreservlets.com for details

Topics in This Section

- **XML**
 - Getting document
 - Ajax
 - HTML page
 - From string (for interactive testing)
 - Document, Element, and Node classes
- **HTML**
 - HTMLDocument and HTML_ELEMENT classes
 - Specialized HTML_ELEMENT classes
 - The Window class
- **Event Handling**
 - General event-handling attributes
 - Element-specific event-handling attributes

5

© 2009 Marty Hall



Intro

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 1.x & JSF 2.0, Struts Classic & Struts 2, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Browser-Specific Classes

- **Previous lecture**
 - Described core JavaScript syntax and objects
- **This lecture**
 - Describes JavaScript specifically for browser applications
- **Reminder re references**
 - Online
 - <http://www.w3schools.com/js/>
 - http://www.w3schools.com/html/dom/dom_reference.asp
 - http://www.devguru.com/technologies/ecmascript/QuickRef/javascript_intro.html
 - Book
 - *JavaScript, the Definitive Guide* by David Flanagan

7

© 2009 Marty Hall



XML

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 1.x & JSF 2.0, Struts Classic & Struts 2, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Overview

- **API is very similar to Java DOM API**
 - If you know Java org.w3c.dom classes and methods, they are almost the same in JavaScript
- **Document class**
 - Represents top-level document
 - Also a specialized version representing the HTML page
- **Element class**
 - Represents XML/HTML element
 - Inherits Node methods plus has some extras
- **Node class**
 - Represents node in XML tree
 - Element is main node type, but there are also text nodes, CDATA notes, and a few others
 - Most Element methods inherited from here

9

Getting Document: Real Life

- **Getting general XML document**
 - You get XML document as the result of an Ajax request
 - `var xmlDocument = request.responseXML;`
- **Getting XML document representing HTML page**
 - Use builtin "document" variable
 - Many special features apply
 - `getDocumentById` method, `innerHTML` property for elements, forms and anchors and images properties, case-insensitivity for `getElementsByTagName`, lots more specific to HTML
 - http://www.w3schools.com/html/dom_obj_document.asp

10

Getting Document: Practice and Learning

- **Firefox**

```
var xmlString = "<blah>...</blah>";  
var parser = new DOMParser();  
var xmlDocument =  
    parser.parseFromString(xmlString, "application/xml");
```

- **Internet Explorer**

```
var xmlDocument =  
    new ActiveXObject("Microsoft.XMLDOM");  
xmlDocument.async = false;  
xmlDocument.loadXML(xmlString);  
return (xmlDocument);
```

- **Figuring out which is which**

- Check if (typeof DOMParser != "undefined")
 - True: use Firefox approach
 - False: use IE approach
- Warning: just for practice

11

Syntax Summary

- **Document class**

- Properties
 - documentElement
- Methods
 - getElementById (HTML only), getElementsByTagName

- **Element class**

- Methods
 - getAttribute, getElementsByTagName, hasAttribute

- **Node**

- Properties
 - attributes, childNodes, firstChild, lastChild, nextSibling, nodeName, nodeType, nodeValue, parentNode, previousSibling
- Methods
 - hasAttributes, hasChildNodes, normalize

12

The Document Class

- **documentElement** property
 - The root Element of the document
- **getElementById** method
 - Returns the Element with the specified ID.
 - For HTML documents only!
 - Refers to attribute that the DTD defines as an “id attribute”, not necessarily named “id”. Does not match attributes named “id” in regular XML docs.
 - Is supposed to be case-sensitive, but IE is case-insensitive
- **getElementsByTagName**
 - Returns an array of Elements that have that tag name
 - Can use “*” for all Elements in document
 - Unsupported in IE 5
 - Is case-sensitive for regular XML documents
 - Is case-insensitive for HTML documents
 - Even when using XHTML

13

The Document Class: Examples (Code)

```
function getXmlDoc(xmlString) {
    var parser = new DOMParser();
    var xmlDocument =
        parser.parseFromString(xmlString, "application/xml");
    return(xmlDocument);
}
var test =
    "<customers rating='vip'>" +
    "  <customer id='a1234'>" +
    "    <firstName>Rafael</firstName>" +
    "    <lastName>Nadal</lastName>" +
    "  </customer>" +
    "  <customer id='a1235'>" +
    "    <firstName>Roger</firstName>" +
    "    <lastName>Federer</lastName>" +
    "  </customer>" +
    "</customers>";
var testDoc = getXmlDoc(test);
```

14

The Document Class: Examples (Results)



```
Firebug - Document Class
File View Help
Inspect Clear Profile
Console HTML CSS Script DOM Net Options
>>> testDoc.documentElement.nodeName;
"customers"
>>> testDoc.documentElement.getAttribute("rating");
"vip"
>>>
testDoc.getElementsByTagName("lastName")[1].firstChild.nodeValue;
"Federer"
>>> var rafie = testDoc.getElementsByTagName("customer")[0];
>>> rafie.getAttribute("id");
"a1234"
>>> testDoc.getElementById("a1234");
null
>>> |
```

15

The Element Class

- **getAttribute**
 - Gets value of designated attribute.
 - E.g., if element refers to `<foo bar="a" baz="b">...</foo>`, `element.getAttribute("baz")` returns "b"
- **getElementsByTagName**
 - Returns an array of subelements that have this tag name
 - Subelements can be arbitrarily nested
- **hasAttribute**
 - Tests if element has attribute of given name
- **Also inherits from Node class**
 - See next slides
 - All Elements are Nodes, but not vice versa

16

The Node Class: Properties

- **attributes**
 - An array of the attributes (call nodeName on each to get names)
- **childNodes**
 - An array of direct child nodes. 0-length if no children.
- **firstChild, lastChild, parentNode**
 - Specific child nodes. Parent node (null for top element).
- **nextSibling, previousSibling**
 - Related children of the parent node
- **nodeName**
 - For Element nodes, the XML element name
- **nodeType**
 - Node.ELEMENT_NODE, Node.TEXT_NODE, Node.ATTRIBUTE_NODE, Node.CDATA_SECTION_NODE, and a few other options. **Fails on some IE versions!**
- **nodeValue**
 - For Text nodes, the body content.
 - Call normalize first. See next slide.

17

Fixing Node Types in Internet Explorer

- **Problem**
 - Official standards Node.TEXT_NODE etc., unsupported through Internet Explorer 6 (OK in IE 7)
- **Solution**
 - Redefine them
 - <http://www.ibm.com/developerworks/xml/library/x-matters41.html>

```
if (!window['Node']) {
    window.Node = new Object();
    Node.ELEMENT_NODE = 1;
    Node.ATTRIBUTE_NODE = 2;
    Node.TEXT_NODE = 3;
    Node.CDATA_SECTION_NODE = 4;
    ...
    Node.DOCUMENT_NODE = 9; ... }
```

18

The Node Class: Methods

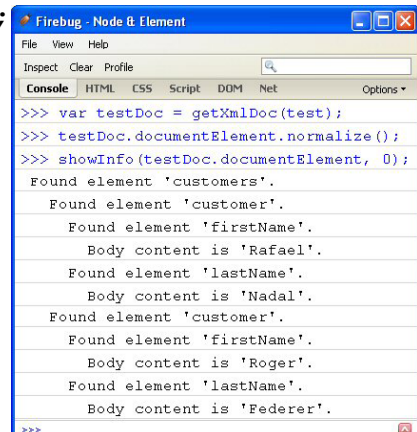
- **hasAttributes**
 - Does this Node have any attributes at all?
- **hasChildNodes**
 - Does this Node have any children at all?
- **normalize**
 - Merges multiline text nodes.
 - Important if element has body content that spans multiple lines or has extra white space.
 - But you can still have empty text nodes
 - You can call it on root element just once.
 - `xmlDoc.documentElement.normalize();`

19

Node and Element: Example

```
function showInfo(node, indent) {
  if (node.nodeType == Node.TEXT_NODE) {
    console.log("%s Body content is '%s'.",
      spaces(indent), node.nodeValue);
  } else if (node.nodeType == Node.ELEMENT_NODE) {
    console.log("%s Found element '%s'.",
      spaces(indent), node.nodeName);
    var children = node.childNodes;
    for(var i=0; i<children.length; i++) {
      showInfo(children[i], indent+1);
    }
  }
}

function spaces(n) {
  var indentString = "  ";
  var result = "";
  for(var i=0; i<n; i++) {
    result =
      result.concat(indentString);
  }
  return(result);
}
```



```
Firebug - Node & Element
File View Help
Inspect Clear Profile
Console HTML CSS Script DOM Net Options
>>> var testDoc = getXmlDoc(test);
>>> testDoc.documentElement.normalize();
>>> showInfo(testDoc.documentElement, 0);
Found element 'customers'.
  Found element 'customer'.
    Found element 'firstName'.
      Body content is 'Rafael'.
    Found element 'lastName'.
      Body content is 'Nadal'.
    Found element 'customer'.
      Found element 'firstName'.
        Body content is 'Roger'.
      Found element 'lastName'.
        Body content is 'Federer'.
```

20



HTML and Browser Capabilities

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 1.x & JSF 2.0, Struts Classic & Struts 2, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

HTMLDocument: Properties

- **Obtain with predefined 'document' variable**
 - Specialized subclass of Document class discussed earlier
- **Properties**
 - title, domain, URL
 - Info about the document. URL is same as `window.location.href` unless redirection occurs
 - body
 - The body element
 - anchors, applets, forms, images, links
 - Arrays of subelements, in the order they appear in the document. Usually better to find elements by ids.
 - cookie, lastModified, referrer
 - In Ajax, it is usually better to manipulate these on server
 - blah
 - Element that has `name="blah"` (first one if repeated)

HTMLDocument: Methods

- **write, writeln**
 - Dynamically insert text into document
 - Used from <script> tag that has body content
 - Not used by Ajax response handlers
 - Use HTMLDocument.innerHTML property instead
- **getElementsByName**
 - Returns array of elements that have given name attribute
- **getElementsByTagName**
 - Returns array of elements that have given element name
 - Case insensitive
 - Inherited from Document class (see earlier slide)
- **getElementById (inherited from Document)**
 - Finds element with specified id attribute
 - Inherited from Document class (see earlier slide)

23

HTMLElement

- **Subclass of Element class. Obtain with**
 - document.body, document.getElementsByTagName, document.getElementsByName, document.images. etc.
 - otherElement.getElementsByTagName, otherElement.childNodes, otherElement.firstChild, etc.
- **Most important properties**
 - id
 - The id attribute
 - nodeName
 - Element name (inherited from Node class).
 - name
 - The name attribute (for HTML elements with "name" only)
 - **innerHTML**
 - Read/write property giving HTML text inside element
 - style
 - CSS2Properties object representing element styling
 - className
 - Space-separated list of CSS class names
- **Method**
 - scrollIntoView
 - Scrolls browser so element is visible

24

Form Class

- **Obtaining reference**
 - document.forms array, “form” variable for code invoked by input element
 - Any method or property that returns Node (getElementById, childNodes, etc.)
- **Properties**
 - elements
 - Array of all input elements in form
 - action, encType, method, name, target
 - Corresponds to HTML attributes
- **Methods**
 - submit, reset

25

Image Class

- **Obtaining reference**
 - document.images array, document.nameOfImage
 - Any method or property that returns Node (getElementById, childNodes, etc.)
- **Properties**
 - src
 - Read/write property. Changing this changes the image.
 - Preload images first with new Image(...) and setting its href, so browser will have images cached.
 - name, align, alt, border, height, hspace, ismap, usemap, vspace, width
 - Corresponds to HTML attributes
 - Inherited ones described earlier
 - Node, Element, HTMLInputElement

26

Input Class

- **Obtaining reference**
 - theForm.elements array
 - Any method or property that returns Node (getElementById, childNodes, etc.)
- **Properties**
 - name, id, value, type, disabled, form (enclosing form)
 - For all input elements
 - defaultValue
 - Initial value as given in the HTML
 - Type-specific properties (see online API)
 - checked, maxLength, useMap, etc.
- **Methods**
 - blur/focus (all), click (buttons, checkboxes, radio buttons), select (file, password, text)

27

Window Class: Properties

- **Obtaining reference**
 - Use “window” or “self”
- **history**
 - History object. Not writable.
- **location**
 - Location object.
 - location.href = "new address" redirects browser
- **status**
 - Status line value. Writable.
- **Size/position/scrolling**
 - innerWidth, innerHeight, outerWidth, outerHeight, screenX (or screenLeft), screenY (or screenTop)

28

Window Class: Methods

- **alert, confirm, prompt**
 - Pops up dialog boxes of various sorts
- **print**
 - Invokes print dialog
- **setInterval, clearInterval [repeated actions]**
 - setInterval(someFunction, milliseconds)
- **setTimeout, clearTimeout [one-time actions]**
 - setTimeout(someFunction, milliseconds)
- **getComputedStyle**
 - Get style info for specified element
- **Movement**
 - Lots of methods for opening, closing, resizing windows

29

© 2009 Marty Hall



Event Handling

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 1.x & JSF 2.0, Struts Classic & Struts 2, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Idea

- **Most HTML elements have *onblah* attributes**
 - Designates function to run when specified event occurs
- **General event handlers**
 - onclick, ondblclick
 - User single-clicks or double-clicks over element
 - onkeydown, onkeypress, onkeyup
 - User presses key when element has the focus
 - onmousedown, onmouseup
 - User presses mouse over element
 - onmouseover, onmouseout
 - Mouse moves over or leaves being over element
 - onmousemove
 - Mouse moves while over element

31

Approaches to Assigning Event Handlers

- **Assign to property directly in HTML**
 - `<input type="button" onclick="someFunctionCall()"/>`
 - Note the parens: this is a function *call*
 - Inside the function, “this” refers to the window
- **Assign to property indirectly**
 - `var element = document.getElementById("blah");`
 - `element.onclick = someFunctionName;`
 - Notes: no parens: this is a function *value*. Inside the function, “this” refers to the element.
 - This process often done from window.onload
- **Use `addEventListener` or `attachEvent`**
 - DOM 2: `addEventListener`; IE: `attachEvent`
 - Lack of portability makes this approach hard to manage

32

Pros and Cons of Event-Handling Approaches

- **Assigning to property directly in HTML**
 - `<input ... onclick="someFunctionCall()"/>`
 - Simpler, especially for beginners
 - Slightly easier to pass arguments to the handler
- **Assigning to property indirectly**
 - `someElement.onclick = someFunctionName;`
 - This approach is sometimes known as “unobtrusive JavaScript”.
 - Better separation: all JavaScript in JS files. HTML file contains only HTML.
 - More work to set up unless you use a JavaScript library like Prototype or jQuery. Using `window.onload` is tricky (see later slides).
 - Slightly more work to pass arguments to function (use anonymous function)

33

Example Code: Directly Assigning Event Handler

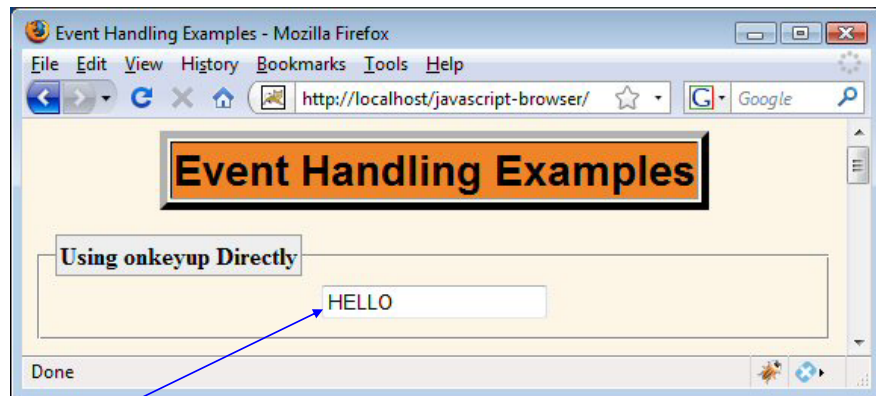
- **JavaScript**

```
function makeUpperCase(textfield) {
    textfield.value = textfield.value.toUpperCase();
}
```
- **HTML**

```
<input type="text" onkeyup="makeUpperCase(this)"/>
```

34

Example Output: Directly Assigning Event Handler



Input was "Hello"
(not "HELLO")

35

Example Code: Indirectly Assigning Event Handler

- **JavaScript**

```
function makeUpperCase(textfield) {  
    textfield.value = textfield.value.toUpperCase();  
}  
  
function makeMeUpperCase() {  
    makeUpperCase(this);  
}  
  
window.onload = function() {  
    document.getElementById("uppercase-field").onkeyup =  
        makeMeUpperCase;  
}
```

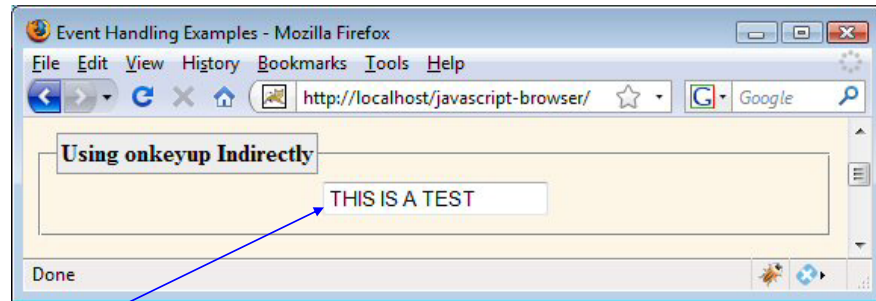
See slide near end about making
window.onload safer when loading
multiple JavaScript libraries.

- **HTML**

```
<input type="text" id="uppercase-field"/>
```

36

Example Output: Indirectly Assigning Event Handler



Input was "This Is a Test"
(not "THIS IS A TEST")

37

Passing Events to Event Handlers

- **Idea**
 - JavaScript automatically passes an event object as the last argument to event handler functions
 - Although you had to use `window.event` in old IE versions
 - Sometimes you need the event object for extra info
 - **General**
 - `event.type` ("click", "mouseover", etc.)
 - `event.target` (element on which event occurred)
 - **Mouse events**
 - `event.button` (0 = left, 1 = middle, 2 = right)
 - `event.altKey`, `event.ctrlKey`, `event.metaKey`, `event.shiftKey`
 - » Booleans indicating if keys were down when mouse event occurred
 - `event.clientX`, `event.clientY`, `event.screenX`, `event.screenY`
 - **Keyboard events**
 - `event.charCode`, `event.keyCode` (see later example)

38

Keyboard Events (onkeypress)

- **Internet Explorer**

- Use “event” argument in newer versions
- Use “window.event” in older versions
- event.charCode is numeric character code
 - For printable characters, convert to character with String.fromCharCode

- **Other browsers**

- Use “event” in all versions
- event.charCode is numeric character code if character was printable
 - Convert to character with String.fromCharCode
- event.keyCode is numeric character code if character was nonprintable (arrow, ENTER, Control, etc.)
 - You must compare to numeric values here

39

Example: Portable Character Checking

- **Goal**

- Recognize when Down Arrow is pressed
- When pressed, do whatever pushbutton would have done

- **Main ideas**

- Define handler to take “event” as argument
- Use “event” if defined, otherwise “window.event”
- Use charCode if defined, otherwise keyCode
- Don’t repeat code: programmatically look up the button’s onclick handler and call it

40

Example JavaScript: Portable Character Checking

```
function showValue(inputID, resultID) {
    var html =
        "<div class='sample'>" +
        document.getElementById(inputID).value +
        "</div>"
    document.getElementById(resultID).innerHTML =
        html;
}

function doOnClickOf(buttonID, event) {
    var e = event || window.event;
    var code = e.charCode || e.keyCode;
    if (code == 40) { // 40 is Down Arrow
        var f = document.getElementById(buttonID).onclick;
        f();
    }
}
```

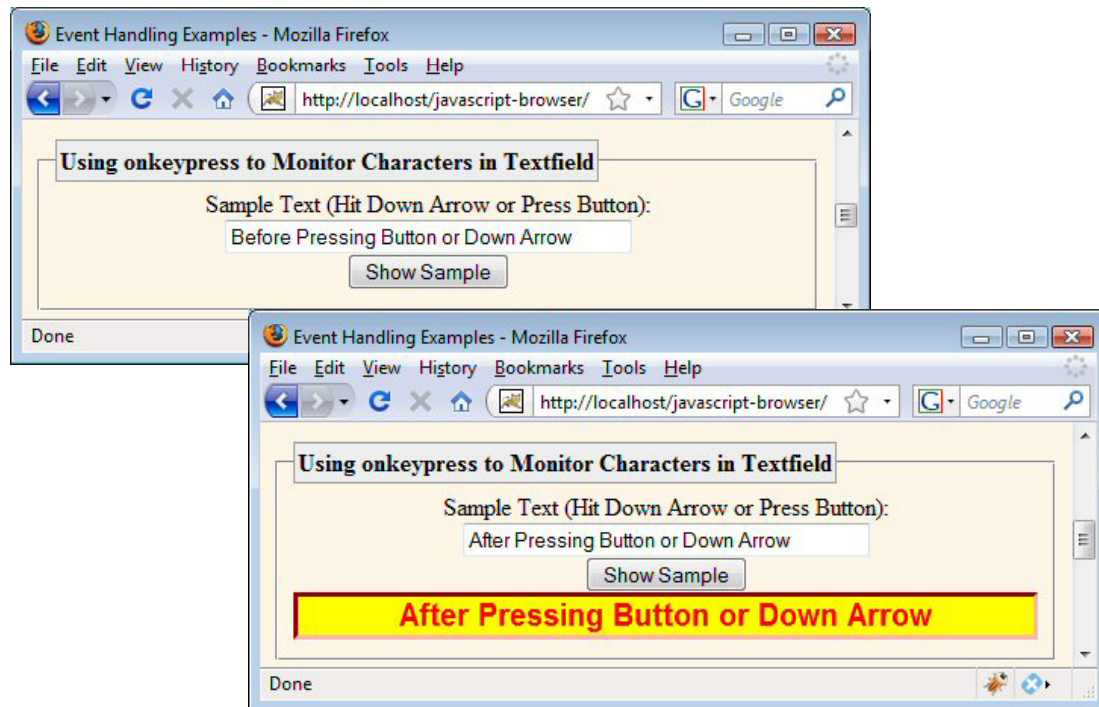
41

Example JavaScript: Portable Character Checking

```
<form action="#">
    <label for="textfield-1">
        Sample Text (Hit Down Arrow or Press Button):
    </label>
    <input type="text" id="textfield-1" size="40"
        onkeyup="doOnClickOf('button-1', event)"/><br/>
    <input type="button" id="button-1" value="Show Sample"
        onclick="showValue('textfield-1', 'div-1')"/><br/>
    <div id="div-1"></div>
</form>
```

42

Example Output: Portable Character Checking



43

Mouse Events

- **Can capture mouse events on *any* element**
 - Images, links, input elements, even normal text
- **Event Properties**
 - To access properties, use event handler that takes event as final argument.
 - To support old browsers, use window.event if event undefined, as in previous example
 - altKey, ctrlKey, metaKey, shiftKey
 - Booleans that tell if key was down when event occurred
 - button
 - 0 for left button, 1 for middle, 2 for right
 - clientX, clientY
 - The x and y coordinates relative to the browser window
 - screenX, screenY
 - The x and y coordinates relative to the user's monitor

44

Mouse Event Handlers: Example (JavaScript)

```
function on(event) {
  var e = event || window.event;
  var message =
    "<ul class='sample'>" +
    "  <li>clientX: " + e.clientX + "</li>" +
    "  <li>clientY: " + e.clientY + "</li>" +
    "  <li>screenX: " + e.screenX + "</li>" +
    "  <li>screenY: " + e.screenY + "</li>" +
    "</ul>";
  var region = document.getElementById("messageRegion");
  region.innerHTML = message;
}

function off() {
  var region = document.getElementById("messageRegion");
  region.innerHTML = "";
}
```

45

Mouse Event Handlers: Example (HTML)

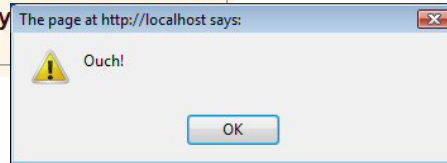
```
<fieldset>
  <legend>Using onclick on Arbitrary Elements</legend>
  <h2 onclick="alert('Ouch!')">
    Here is a heading. What happens when you click on it?
  </h2>
</fieldset>
<p/>
<fieldset>
  <legend>Using onmouseover and onmouseout</legend>
  <h2 onmouseover="on(event)" onmouseout="off()">
    Here is a heading. What happens when you move over it?
  </h2>
  <div id="messageRegion"></div>
</fieldset>
```

46

Mouse Event Handlers: Example (Results)

Using onclick on Arbitrary Elements

Here is a heading. What happens when you



Using onmouseover and onmouseout

Here is a heading. What happens when you move over it?

Here is a heading. What happens when you move over it?

- clientX: 205
- clientY: 541
- screenX: 1120
- screenY: 684

47

Specialized Event Handlers

- **input**
 - onclick
 - For pushbuttons and toggle buttons.
 - Also fires when button is invoked via keyboard.
 - onchange
 - For textfields, when change is committed (focus leaves)
 - Use onkeyup for individual characters
 - onblur, onfocus
- **form**
 - onsubmit, onreset
 - Return false to prevent form from really being submitted.
 - Widely used for client-side validation of form fields.
- **body**
 - onblur, onerror, onfocus, onload, onresize, onunload
- **img**
 - onabort, onerror, onload

48

Specialized Event Handlers: window.onload

- **Purpose**

- Run JavaScript code after page is done loading. Used to insert HTML in certain regions or to attach event handlers to certain HTML elements. Neither can be done until page is done loading

- **Simple example (myfile.js)**

```
window.onload = function() {  
    document.getElementById("...").onclick = ...;  
    document.getElementById("...").innerHTML = ...;  
}
```

49

window.onload and Multiple JavaScript Libraries

- **Problem**

- Assigning directly to window.onload replaces any existing window.onload function.
- Another library might already be using window.onload

- **Solution**

- See if window.onload exists.
 - On Firefox, if no window.onload (typeof window.onload == "undefined") ...
 - On IE, if no window.onload, window.onload is null
 - **Either way, you can test !window.onload**
- If so, grab the function
 - var oldWindowLoadFunction = window.onload;
- And call it before your window.onload functions
 - oldWindowLoadFunction();

50

Safer window.onload

```
if (!window.onload) {
  window.onload = function() {
    document.getElementById("...").onclick = ...;
    document.getElementById("...").innerHTML = ...;
  };
} else {
  var oldWindowLoadFunction = window.onload;
  window.onload = function() {
    oldWindowLoadFunction();
    document.getElementById("...").onclick = ...;
    document.getElementById("...").innerHTML = ...;  };
}
```

51

Additional window.onload Problem

- **Problem**
 - window.onload runs after the entire page (including images and style sheets) have been loaded. You have to wait until the DOM is parsed, but you shouldn't have to wait until after the images have been loaded.
- **Solutions**
 - Use window.addEventListener or window.attachEvent
 - But it is tricky to make this portable across browsers
 - Many JavaScript libraries (including Prototype and jQuery) have simple methods for defining code that runs after the DOM is loaded, but before images and style sheets are loaded

52

Summary

- **XML**
 - Obtaining document
 - request.responseXML or parseFromString (testing only)
 - Node and Element properties and methods
 - attributes, childNodes, firstChild, nodeName, nodeValue
 - getElementsByTagName, getAttribute, normalize
- **HTML**
 - Methods and properties
 - document.writeln, getElementById
 - name, id, value, type, disabled, innerHTML
- **Event Handling**
 - `<input type="button" onclick="someFunction()"/>`
 - `someElement.onclick = someFunction;`

53

© 2009 Marty Hall



Questions?

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 1.x & JSF 2.0, Struts Classic & Struts 2, Ajax, GWT, Spring, Hibernate/JPA, Java 5 & 6.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.