

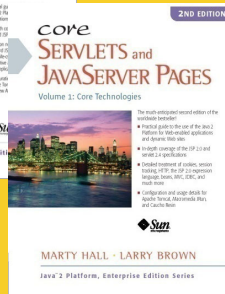
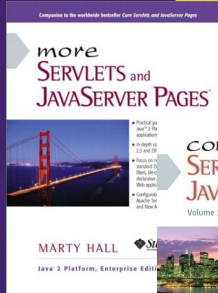


# JavaScript: A Crash Course

## Part III: 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 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 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, servlets/JSP (intermediate and advanced), Struts, JSF 1.x, JSF 2.0, Ajax, GWT 2.0 (with GXT), custom mix of topics
  - Ajax courses can concentrate on 1 library (jQuery, Prototype/Scriptaculous, Ext-JS, Dojo, Google Closure) or survey several
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Spring, Hibernate/JPA, EJB3, Web Services, Ruby/Rails

Contact [hall@coreservlets.com](mailto:hall@coreservlets.com) for details

# Topics in This Section

- **Manipulating the DOM**
  - Finding elements of interest
  - Inserting HTML into existing elements
- **Reading textfield values**
  - And other input elements
- **Assigning event handlers**
  - Directly in the HTML
  - From JavaScript, via `window.onload`
- **HTML-specific classes**

5

© 2010 Marty Hall



## Intro

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

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.

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

# Main Points

- **Most important topics**

- How to insert HTML into a page
  - `document.getElementById(...).innerHTML = ...;`
- How to read a textfield value
  - `escape(document.getElementById(...).value);`
- How to assign an event handler
  - `<input ... onclick="someFunctionCall()"/>`
    - This goes in the HTML page
  - `someElement.onclick = someFunction;`
    - This is done directly in JavaScript, probably from the `window.onload` handler.

» This section cover a lot of other topics, and unless you have JavaScript experience already, you won't follow all of it. But, for most Ajax applications, the topics above are by far the most important ones.

7

# Browser-Specific Classes

- **Previous two lectures**

- Described general JavaScript syntax and capabilities

- **This lecture**

- Describes JavaScript specifically for browser applications

- **Next lecture**

- Describes JavaScript capabilities for parsing XML

- **Reminder re references**

- Online
  - <http://www.w3schools.com/js/>
  - [http://www.w3schools.com/html/dom/dom\\_reference.asp](http://www.w3schools.com/html/dom/dom_reference.asp)
  - [http://www.devguru.com/technologies/ecmascript/QuickRef/javascript\\_intro.html](http://www.devguru.com/technologies/ecmascript/QuickRef/javascript_intro.html)
- Book
  - *JavaScript, the Definitive Guide* by David Flanagan

8



# Manipulating the DOM

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

## Summary

- **Most important topics**
  - How to find the element that has a given id
    - `var element = document.getElementById("...");`
  - How to insert HTML into a page
    - `element.innerHTML = "...";`
      - You can insert into all sorts of elements (div, span, p, h1, td, etc.).
      - But, be sure that what you insert would have been legal if it had been placed there to begin with. So, don't try to insert block-level elements (like h1) into inline elements (like span).
- **Auxiliary topics**
  - Finding HTML elements by name
  - Finding all HTML elements of a certain type
  - General info that represents entire page (URL, title, etc.)

# HTMLDocument Class

- **Overview**
  - Obtain reference with predefined “document” variable
  - Specialized subclass of the Document class that will be discussed in XML lecture
- **Properties**
  - title, domain, URL
    - Info about the page.
    - 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. Notice it is document.referrer, even though the HTTP header is Referer.
  - blah
    - Element that has name="blah" (first one if name repeated)

11

# HTMLDocument: Methods

- **getElementById**
  - Finds element with specified id attribute
- **write, writeln**
  - Dynamically insert text into document
  - Used from <script> tag that has body content
  - Not used by Ajax response handlers
    - Use HTMLElement.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 XML lecture)

12

# 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

13

# The Selectors API

- **W3C defined API for search based on CSS**
  - Find a single element that matches CSS selector
  - Find all elements that match CSS selector or selectors
  - Details at <http://www.w3.org/TR/selectors-api/>
  - Supported by Firefox 3.5 and recent Chrome releases
- **Main two methods**
  - document.querySelector
  - document.querySelectorAll
- **See jQuery lectures for details & examples**
  - jQuery used this approach before W3C API existed
    - For speed, jQuery now uses native methods if browser supports them

14



# Reading Values of Input Elements

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

## 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

# 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

17

# 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)

18



# The Window Class

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

## Idea

- **General information about the page**
  - History, URL, status line
  - But does not refer to the DOM (use document for that)
- **Most important for Ajax**
  - `window.onload = someFunction;`
    - Code to run on startup
  - `window.setInterval(someFunction, milliseconds);`
    - Code to run periodically
  - `window.alert("some message")`
    - Message to go in popup dialog box. You can just use `alert` instead of `window.alert`.

# Window Class

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

21

# 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

22

# Window Class: The onload Event Handler

- **Idea**

- Code to run after the page is loaded.
  - You can't directly do the following in the JavaScript file that is loaded in the head:
    - `document.getElementById("someID").innerHTML = "some HTML";`
    - `document.getElementById("someID").onclick = someFunction;`
  - The reason is that that DOM does not exist yet, so you can't find the elements and manipulate them.
- General form
  - `window.onload =`  
`function() { codeToRunAfterPageLoads(); }`

- **Details**

- See later section on event handling

23

© 2010 Marty Hall



## Event Handling

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

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 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

25

## 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;`
    - Note: 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

26

# 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;`
    - Better separation: all JavaScript in JS files. HTML file contains only HTML.
      - This approach is sometimes known as “unobtrusive JavaScript”.
    - 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)

27

# Example Code: Directly Assigning Event Handler

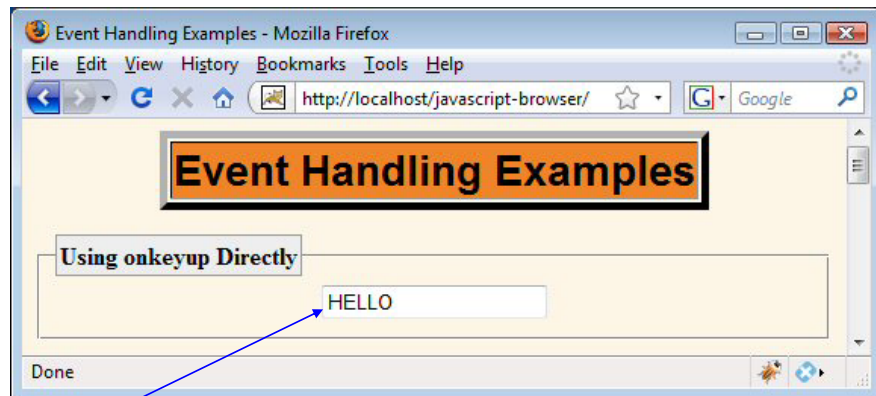
- **JavaScript**

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

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

28

## Example Output: Directly Assigning Event Handler



Input was "Hello"  
(not "HELLO")

29

## 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;  
}
```

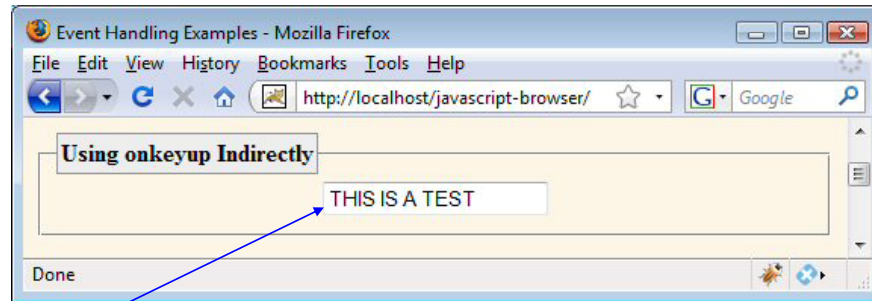
See slide near end about making  
window.onload safer when loading  
multiple JavaScript libraries.

- **HTML**

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

30

## Example Output: Indirectly Assigning Event Handler



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

31

## 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)

32

# 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

33

# 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

34

## 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();
    }
}
```

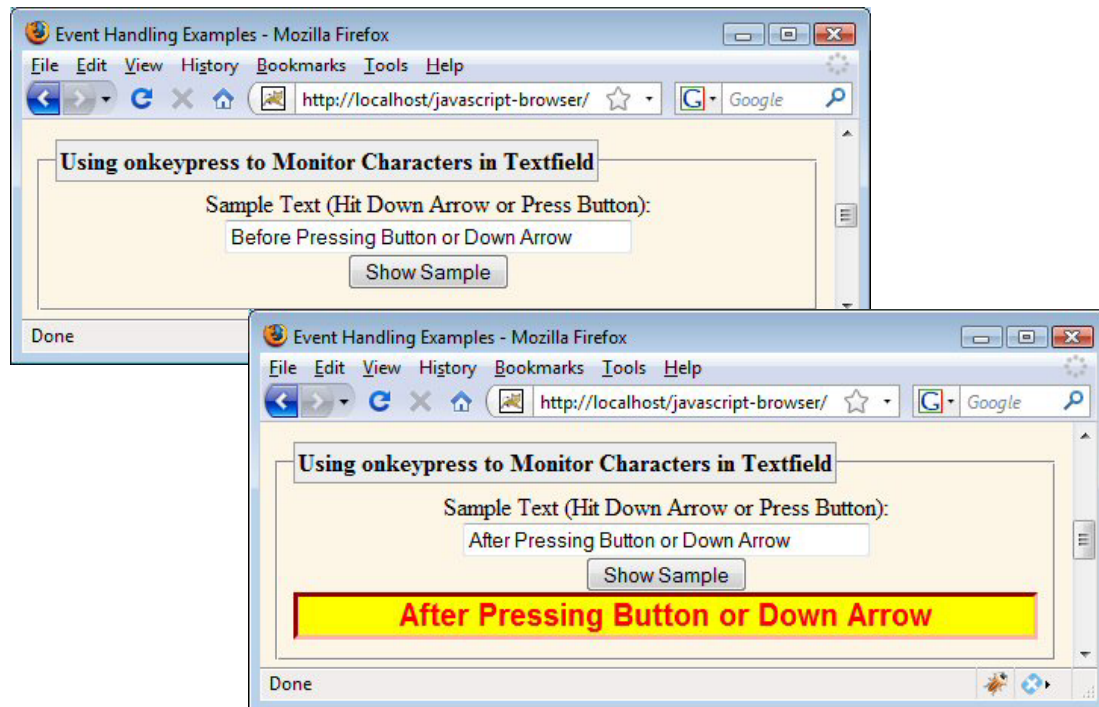
35

## 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>
```

36

# Example Output: Portable Character Checking



37

## 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

38

## 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 = "";
}
```

39

## Mouse Event Handlers: Example (HTML)

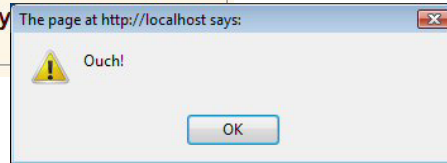
```
<fieldset>
  <legend>Using onclick on Arbitrary Elements</legend>
  <h2 onclick="alert('Ouch!')">
    Here is a heading. What happens when you click on it?
  </h2>
</fieldset>
<p/>
<fieldset>
  <legend>Using onmouseover and onmouseout</legend>
  <h2 onmouseover="on(event)" onmouseout="off()">
  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

40

# 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

41

# 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

42

# 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 = ...;  
}
```

43

# 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();

44

# 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 = ...;  };
}
```

45

# 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

46

# Summary

- **Manipulating the DOM**

```
var someElement = document.getElementById("...");
```

- Many ways to find elements other than by id

```
someElement.innerHTML = "...";
```

- **Reading textfield values**

```
var val = someElement.value;
```

- **Assigning event handlers**

```
<input type="button" onclick="someFunctionCall()"/>
```

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

47

© 2010 Marty Hall



## Questions?

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

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.

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