CitiusTech

Session 2 : Manipulating Elements in JQuery

February 2012

This document is confidential and contains proprietary information, including trade secrets of CitiusTech. Neither the document nor any of the information contained in it may be reproduced or disclosed to any unauthorized person under any circumstances without the express written permission of CitiusTech.

Contents

Manipulating Element Properties (1/2)

- Each is the command used to traverse through a set of elements.

Examples	Explanations

`$('img').each(function(n){`
`   this.alt='This is image['+n+'] with an id of '+this.id;`
`});`
`});`
	• This command can be used to easily set a property value onto all elements in a matched set.

`var allAlts = new Array();`
`$('img').each(function(){`
`   allAlts.push(this.alt);`
`});`
	• Similarly, we can collect all values for a specific property into an array using each()

`var altValue = $('#myImage')[0].alt;`	• We can also obtain the property value of a single element.

CitiusTech

Manipulating Element Properties (2/2)

Each Syntax

Command	Explanation
each (iterator)	• Traverses all elements in the matched set invoking the passed iterator function for each.
Parameter	**iterator** (Function) • A function called for each element in the matched set. • The parameter passed to this function is set to the zero-based index of the element within the set, and the element itself is available as the this property of the function.
Returns	• The wrapped set.

Contents

Attribute Values - Fetching (1/2)

- The attr() command can be used to either fetch the value of an attribute from the first element in the matched set or set attribute values onto all matched elements.

Attr Syntax

Command	Explanation
attr (name)	• Obtains the values assigned to the specified attribute for the first element in the matched set.
Parameter	**name** (String) • The name of the attribute whose value is to be fetched.
Returns	• The value of the attribute for the first matched element. • The value undefined is returned if the matched set is empty or the attribute doesn’t exist on the first element.

Attribute Values - Fetching (2/2)

Examples	Explanations
<img id="myImage" src="image.gif" alt="An image" class="someClass" title="This is an image" **custom="some value"/>**	• Even though we usually think of attributes as predefined by HTML, we can use attr() with custom attributes set through JavaScript or HTML markup. • To illustrate this, let's amend the <img> element of our previous example with a custom markup attribute (highlighted in bold):
$("#myImage").attr("custom")	• Note that we have added a custom attribute, unimaginatively named custom, to the element. We can retrieve that attribute's value, as if it were any of the standard attributes, with getAttribute() and setAttribute() methods.

Attribute Values - Setting (1/2)

- There are two ways to set attributes onto elements in the wrapped set with JQuery

Command	Parameter	Returns
attr (name, value) • Sets the named attribute onto all elements in the wrapped set using the passed value.	**name** (String) • The name of the attribute whose value is to be fetched.	• The value of the attribute for the first matched element. • The value undefined is returned if the matched set is empty or the attribute doesn't exist on the first element.
attr(attributes) • Sets the attributes and values specified by the passed object onto all elements of the matched set	**attributes** (Object) • An object whose properties are copied as attributes to all elements in the wrapped set	• The wrapped set

Attribute Values - Setting (2/2)

Examples	Explanations

`$('*').attr('title',function(index) {`
`  return 'I am element ' + index + ' and my name is ‘ +`
`        (this.id ? this.id : 'unset');`
`});`
	• This command will run through all elements on the page, setting the title attribute of each element to a string composed using the index of the element within the DOM and the id attribute of each specific element.

`$('input').attr(`
`  { value: '', title: 'Please enter a value' }`
`);`
	• This statement sets the value of all <input> elements to the empty string, as well as sets the title to the string - Please enter a value.

Attribute Values - Removing

- In order to remove an attribute from DOM elements, JQuery provides the removeAttr() command.

removeAttr Syntax

Command	Explanations
removeAttr(name)	• Removes the specified attribute from every matched element
Parameter	**name** (String) • The name of the attribute to be removed
Returns	• The wrapped set

Example	Explanations
$("form").submit(function() { $(":submit",this).attr("disabled", "disabled"); });	• Within the body of the event handler, we grab all submit buttons that are inside our form with the : submit selector and modify the disabled attribute to the value "disabled“. • when building the matched set, we provide a context value (the second parameter) of this.

CitiusTech

Contents

Styles - Changing Elements (1/4)

- We have 2 options to change to change the styling of an element
  - We can add or remove a CSS class, causing the existing stylesheet to restyle the element based on its new classes.
  - we can operate on the DOM element itself, applying styles directly.

Adding and removing class names

- Adding class names to all the elements of a matched set is an easy operation with the following addClass() command:

addClass Syntax

Command	Explanations
addClass(names)	• Adds the specified class name or class names to all elements in the wrapped set
Parameter	**name** (String) • A string containing the class name to add or, if multiple class names are to be added, a space-delimited string of class names
Returns	• The wrapped set

CitiusTech

Styles - Changing Elements (2/4)

- Removing class names is using a command removeClass().

removeClass() Syntax

Command	Explanations
removeClass(names)	• Removes the specified class name or class names from each element in the wrapped set
Parameter	**name** (String) • A string containing the class name to remove or, if multiple class names are to be removed, a space-delimited string of class names
Returns	• The wrapped set

Styles - Changing Elements (3/4)

- We may want to switch a set of styles back and forth, perhaps to indicate a change between two states or for any other reasons that make sense with our interface.
- JQuery makes it easy with the toggleClass() command.

toggleClass() Syntax

Command	Explanations
toggleClass(name)	• Adds the specified class name if it doesn't exist on an element, or removes the name from elements that already possess the class name.
Parameter	**name** (String) • A string containing the class name to toggle.
Returns	• The wrapped set

Styles - Changing Elements (4/4)

Example:

Example	Explanations

function swap() {
    $('tr').toggleClass('striped');
}
 	
<table onmouseover="swap();" nmouseout="swap();">
	• This function uses the toggleClass() command to toggle the class named stripe for all <tr> elements.
	
	• The result is that every time the mouse cursor enters or leaves the table, all <tr>elements with the class striped will have the class removed, and all <tr> elements without the class will have it added.

Styles - Getting and Setting (1/5)

- Although modifying the class of an element allows us to choose which predetermined set of defined stylesheet styles should be applied, sometimes we want to override the stylesheet altogether.
- Applying styles directly on the elements themselves will automatically override stylesheets, giving us more fine-grained control over individual elements and their styles.
- The css() method works similarly to the attr() method, allowing us to set an individual CSS property by specifying its name and value, or a series of elements by passing in an object.

Styles - Getting and Setting (2/5)

CSS Syntax

Command	Explanations
css(name,value)	• Sets the named CSS style property to the specified value for each matched element.
Parameter	**name** (String) The name of the CSS property to be set. **value** (String|Number|Function) • A string, number, or function containing the property value. • If a function is passed as this parameter, it will be invoked for each element of the wrapped set with its return value serving as the value for the CSS property. • The this property for each function invocation will be set to the element being evaluated.
Returns	• The wrapped set.

Styles - Getting and Setting (3/5)

Other usage of Css/Width/Height

Command	Parameter	Returns
css(properties) • Sets the CSS properties specified as keys in the passed object to their associated values for all matched elements	**properties** (Object) • Specifies an object whose properties are copied as CSS properties to all elements in the wrapped set	• The wrapped set
css(name) • Retrieves the computed value of the CSS property specified by name for the first element in the wrapped set	**name** (String) • Specifies the name of a CSS property whose computed value is to be returned	• The wrapped set
width(value)/height(value) • Sets the width or height of all elements in the matched set	**value** (Number) • The value to be set in pixels	• The wrapped set
width()/height() • Retrieves the width or height of the first element of the wrapped set	**None**	• The computed width or height as a number

CitiusTech

Styles - Getting and Setting (4/5)

Examples	Explanations

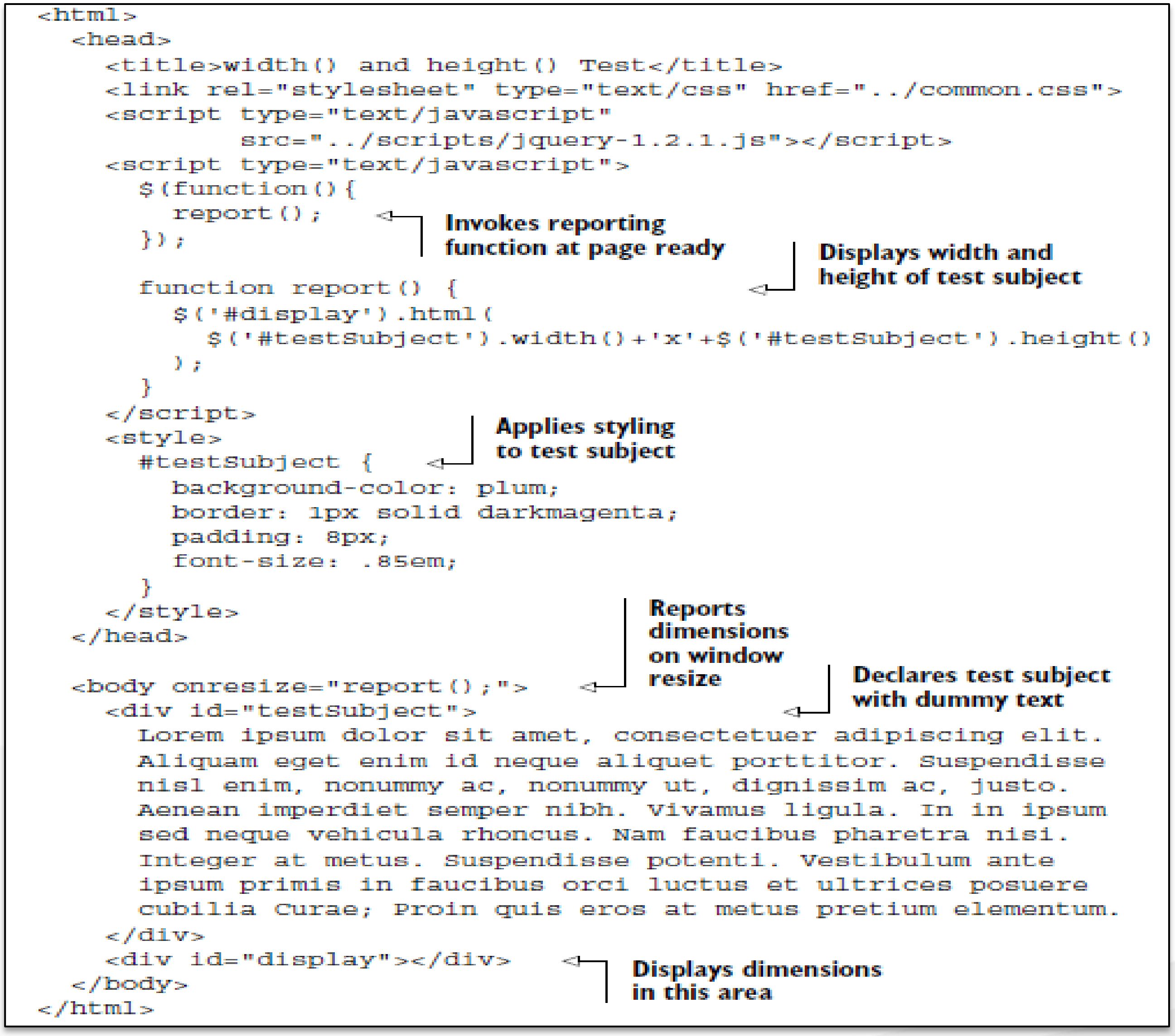
`$("div.expandable").css("width",function() {`
`      return $(this).width() + 20 + "px";`
`});`
	• we are expanding the width of all elements in the wrapped set by 20 pixels.

`$("div.myElements").width(500)`
OR
`$("div.myElements").css("width","500px“)`
	• Shortcut to css() function

`function report() {`
`   $('#display').html(`
`$('#testSubject').width()+'x'+$('#testSubject')`
`.height()`
`   );`
`}`
	• We are defining a function that will use the width() and height() commands to obtain the dimensions of the test subject <div> (named testSubject) and display the resulting values in the second <div> (named display).

CitiusTech

Styles - Getting and Setting (5/5)



```
<html>
  <head>
    <title>width() and height() Test</title>
    <link rel="stylesheet" type="text/css" href="../common.css">
    <script type="text/javascript"
            src="../scripts/jquery-1.2.1.js"></script>
    <script type="text/javascript">
      $(function(){
        report();
      });

      function report() {
        $('#display').html(
          $('#testSubject').width()+'x'+$('#testSubject').height()
        );
      }
    </script>
    <style>
      #testSubject {
        background-color: plum;
        border: 1px solid darkmagenta;
        padding: 8px;
        font-size: .85em;
      }
    </style>
  </head>

  <body onresize="report();">
    <div id="testSubject">
      Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
      Aliquam eget enim id neque aliquet porttitor. Suspendisse
      nisl enim, nonummy ac, nonummy ut, dignissim ac, justo.
      Aenean imperdiet semper nibh. Vivamus ligula. In in ipsum
      sed neque vehicula rhoncus. Nam faucibus pharetra nisi.
      Integer at metus. Suspendisse potenti. Vestibulum ante
      ipsum primis in faucibus orci luctus et ultrices posuere
      cubilia Curae; Proin quis eros at metus pretium elementum.
    </div>
    <div id="display"></div>
  </body>
</html>
```

Invokes reporting function at page ready

Displays width and height of test subject

Applies styling to test subject

Reports dimensions on window resize

Declares test subject with dummy text

Displays dimensions in this area

Styles - More

Examples	Explanations
$("p:first").hasClass("surpriseMe")	• We are trying to check that the element in the matched set has the specified class.
$("p:first").attr("class").split(" ");	• We are trying to obtain the list of classes defined for a particular element as an array instead of the cumbersome space-separated list.

hasClass Syntax

Command	Explanations
hasClass(name)	• Determines if any element of the matched set possesses the passed class name
Parameter	**name** (String) • The class name to be checked
Returns	• Returns true if any element in the wrapped set possesses the passed class name; false if not.

CitiusTech

Contents

Setting Element Content (1/12)

Replacing HTML or text content

- First is the simple html() command, which allows us to retrieve the HTML contents of an element when used without parameters and with the parameter lets us select the

Html Syntax's

Command	Parameter	Returns
html() • Obtains the HTML content of the first element in the matched set.	• None	• The HTML content of the first matched element. • The returned value is identical to accessing the innerHTML property of that element.
html(text) • Sets the passed HTML fragment as the content of all matched elements	**Text** (String) • The HTML fragment to be set as the element content	• The wrapped set

Setting Element Content (2/12)

Example

Examples	Explanations

`<ul id="theList">`
    `<li>One</li>`
    `<li>Two</li>`
    `<li>Three</li>`
    `<li>Four</li>`
`</ul>`
`var text = $('#theList').text();`
	• We can also set or get only the text contents of elements.
	• The text()command, when used without parameters, returns a string that’s the concatenation of all text.
	• results in variable text being set to OneTwoThreeFour.

text Syntax

Command	Parameter	Returns
text() • Concatenates all text content of the wrapped elements	• None	• The concatenated string
text(content) • Sets the text content of all wrapped elements to the passed value.	**content** (String) • The text content to be set into the wrapped elements.	• The wrapped set.

CitiusTech

Setting Element Content (3/12)

Moving and Coping Elements

- To add content to the end of existing content, the append() command is available.

Append Syntax

Command	Explanations
append(content)	• Appends the passed HTML fragment or elements to the content in all matched elements.
Parameter	**content** (String|Element|Object) • A string, element, or wrapped set to append to the elements of the wrapped set.
Returns	• The wrapped set.

Setting Element Content (4/12)

Examples	Explanations
<$('p').append('<b>some text<b>');	• This statement appends the HTML fragment created from the passed string to the end of the existing content of all <p> elements on the page.
$("p.appendToMe").append($("a.appendMe"))	• This statement appends all links with the class appendMe to <p> elements with the class appendToMe. • The disposition of the original elements depends on the number of elements serving as the target of the append. • If there is a single target, the element is removed from its original location—performing a move operation of the original element to a new parent. • In the case where there are multiple targets, the original element remains in place and copies of it are appended to each of the targets—a copy operation.
var toAppend = $("a.appendMe")[0]; $("p.appendToMe").append(toAppend);	• Whether the element identified by toAppend is moved or copied again depends on the number of elements identified by $("p.appendToMe"): • Move if one element is matched, Copy if more than one element is matched.

Setting Element Content (5/12)

- If we want to move or copy an element from one place to another, a simpler approach uses the appendTo() command, which allows us to grab an element and move it somewhere else in the DOM.

appendTo Syntax

Command	Explanations
append(target)	• Moves all elements in the wrapped set to the end of the content of the specified target(s).
Parameter	**target** (String|Element) • A string containing a JQuery selector or a DOM element. • Each element of the wrapped set will be appended to that location. • If more than one element matches a string selector, the element will be copied and appended to each element matching the selector.
Returns	• The wrapped set.

Setting Element Content (6/12)

- A number of related commands work in a fashion similar to append() and appendTo()

Command	Explanations
prepend() and prependTo()	• Works like append() and appendTo(), but insert the source element before the destination target’s contents instead of after
before() and insertBefore()	• Insert the element before the destination elements instead of before the destination’s first child.
after() and insertAfter()	• Insert the element after the destination elements instead of after the destination’s last child.

Setting Element Content (7/12)

Wrapping elements

- Another type of DOM manipulation is to wrap an element (or series of elements) in some markup.

Wrap Syntax

Command	Explanations
wrap(wrapper)	• Wraps the elements of the matched set with the passed HTML tags or a clone of the passed element.
Parameter	**wrapper** (String|Element) • The opening and closing tags of the element with which to wrap each element of the matched set, or an element to be cloned and server as the wrapper.
Returns	• The wrapped set.

CitiusTech

Setting Element Content (8/12)

Example

Examples	Explanations
$("a.surprise").wrap("<div class='hello'></div>")	• To wrap each link with the class surprise in a <div> with the class hello
$("a.surprise").wrap($("div:first")[0]);	• To wrap the link in a clone of the first <div> element on the page

- When multiple elements are collected in a matched set, the wrap() method operates on each one individually.
- If we'd rather wrap all the elements in the set as a unit, we can use the wrapAll() method instead:

Setting Element Content (9/12)

Command syntax: wrapAll/ wrapInner

Command	Parameter	Returns
wrapAll (wrapper) • Wraps the elements of the matched set, as a unit, with the passed HTML tags or a clone of the passed element.	**wrapper** (String|Element) • The opening and closing tags of the element with which to wrap each element of the matched set.	• The wrapped set
wrapInner(wrapper) • Wraps the contents, to include text nodes , elements of the matched set with the passed HTML tags or a clone of the passed element.	**wrapper** (String|Element) • The opening and closing tags of the element with which to wrap each element of the matched set.	• The wrapped set

Setting Element Content (10/12)

Removing/Empty elements

- If we want to empty or remove a set of elements, this can be accomplished with the remove() command whose syntax is as follows:

Remove Syntax

Command	Parameter	Returns
Remove() • Removes all elements in the wrapped set from the page DOM	• None	• The wrapped set
Empty() • Removes the content of DOM elements in matched set	• None	• The wrapped set

Examples	Explanations
$("div.elementToReplace").after("<p>I am replacing the div</p>").remove(); **OR** $("div.elementToReplace").replaceWith("<p>I am replacing the div</p>");	• Because the after() function returns the original wrapped set containing the <div>, we can then remove it, resulting in a replacement of the original <div> with the newly created <p> element.

CitiusTech

Setting Element Content (11/12)

Cloning elements

- One more way that we can manipulate the DOM is to make copies of elements to attach elsewhere in the tree.
- JQuery provides a handy wrapper method for doing so with its clone() command.

Clone Syntax

Command	Explanations
clone(copyHandlers)	• Creates copies of the elements in the wrapped set and returns a new wrapped set that contains them. • The elements and any children are copied. • Event handlers are optionally copied depending upon the setting of the copyHandlers parameter.
Parameter	**copyHandlers** (Boolean) • If true, event handlers are copied. If false, or omitted, handlers are not copied.
Returns	• The newly created wrapped set.

Setting Element Content (12/12)

Example

Examples	Explanations
$('img').clone().appendTo('fieldset.photo');	• This statement makes copies of all image elements and appends them to all <fieldset> elements with the class name photo.
$('ul').clone().insertBefore('#here');	• This command chain performs a similar operation but the targets of the cloning operation—all <ul> elements—are copied, including their children.
$('ul').clone().insertBefore('#here').end().hide();	• This statement performs the same operation as the previous example, but after the insertion of the clones, the end() command is used to select the original wrapped set (the original targets) and hide them. • This emphasizes how the cloning operation creates a new set of elements in a new wrapper.

CitiusTech

Contents

Form Element Values (1/3)

- Val() is used to retrieve the value attribute of a form element for the first element in the wrapped set.

Val Syntax

Command	Explanation
Val()	• Returns the value property of the first element in the matched set. • When the element is a multi-select element, the returned value is an array of all selections.
Parameter	• None
Returns	• The fetched value or values

Few Limitations of Val Command

- If the first element in the wrapped set isn't a form element, a JavaScript error is thrown.
- This command also doesn't distinguish between the checked or unchecked states of check boxes and radio buttons, and will return the value of check boxes or radio buttons as defined by their value attribute, regardless of whether they are checked or not.

Form Element Values (2/3)

- The val() command is also used for supplying a value.

Val() Syntax

Command	Parameter	Returns
Val (value) • Sets the passed value as the value of all matched form elements	**value** (String) • A string that's set as the value property of each form element in the wrapped set	• The wrapped set
Val(values) • Causes any check boxes, radio buttons, or options of <select> elements in the wrapped set to become checked or selected if their values match any of the values passed in the values array.	**values** (Array) • An array of values that will be used to determine which elements are to be checked or selected.	• The wrapped set

Form Element Values (3/3)

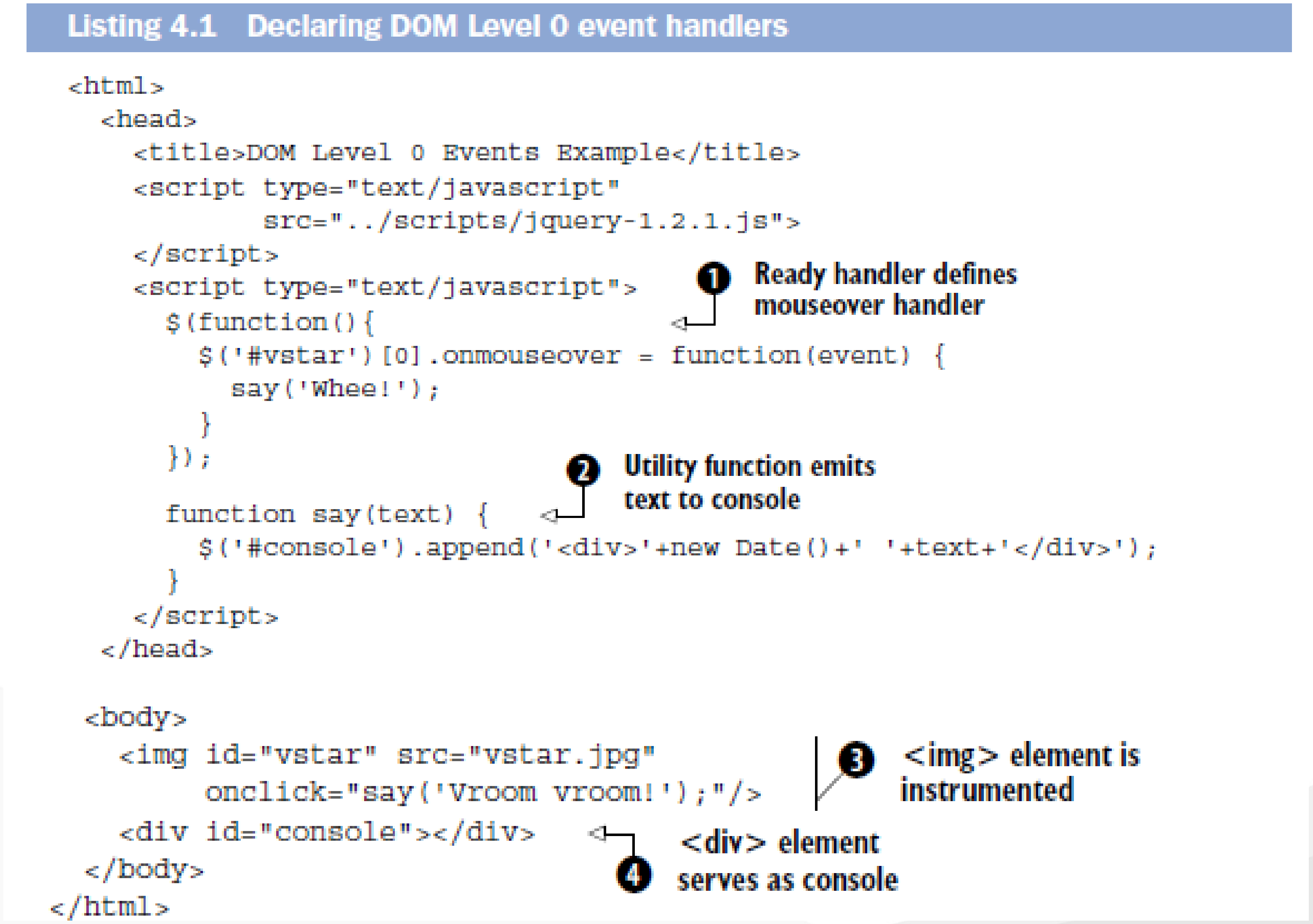
Examples	Explanations
$('[name=radioGroup]:checked').val()	• It will return values of all radioGroups which are checked
$('input,select').val(['one','two','three']);	• This statement will search all the <input> and <select> elements on the page for values that match any of the input strings: one, two or three. • Any check boxes or radio buttons that are found to match will become checked, and any options that match will become selected. • This makes val() useful for much more than just text elements.

Contents

DOM Event Model - Level 0 (1/4)

- The DOM Level 0 Event Model is probably the event model that most web developers employ on their pages.
- In addition to being somewhat browser-independent, it’s fairly easy to use.
- Under this event model, event handlers are declared by assigning a reference to a function instance to properties of the DOM elements.
- These properties are defined to handle a specific event type;
  - for example, a click event is handled by assigning a function to the onclick property, and a mouseover event by assigning a function to the onmouseover property of elements that support these event types.
- The browsers allow us to specify the body of an event handler function as attribute values in the DOM elements’ HTML, providing a shorthand for creating event handlers.

DOM Event Model - Level 0 (2/4)



Listing 4.1 Declaring DOM Level 0 event handlers

```
<html>
  <head>
    <title>DOM Level 0 Events Example</title>
    <script type="text/javascript"
            src="../scripts/jquery-1.2.1.js">
    </script>
    <script type="text/javascript">
      $(function(){
        $('#vstar')[0].onmouseover = function(event) {
          say('Whee!');
        }
      });

      function say(text) {
        $('#console').append('<div>'+new Date()+' '+text+'</div>');
      }
    </script>
  </head>

 <body>
   <img id="vstar" src="vstar.jpg"
        onclick="say('Vroom vroom!');"/>
   <div id="console"></div>
 </body>
</html>
```

1. Ready handler defines mouseover handler
2. Utility function emits text to console
3. <img> element is instrumented
4. <div> element serves as console

DOM Event Model - Level 0 (3/4)

Event Bubbling

- When an event is triggered on an element in the DOM tree, the event-handling mechanism of the browser checks to see if a handler has been established for that particular event on that element and, if so, invokes it. But that’s hardly the end of the story.
- After the target element has had its chance to handle the event, the event model checks with the parent of that element to see if it has established a handler for the event type, and if so, it’s also invoked—after which its parent is checked, then its parent, then its parent, and on and on, all the way up to the top of the DOM tree.
- Because the event handling propagates upward like the bubbles in a champagne flute (assuming we view the DOM tree with its root at the top), this process is termed event bubbling.

DOM Event Model - Level 0 (4/4)

```
<html id="greatgreatgrandpa">
  <head>
    <title>DOM Level 0 Events Example</title>
    <script type="text/javascript"
            src="../scripts/jquery-1.2.1.js">
    </script>
    <script type="text/javascript">
      $(function(){
        $('*').each(function(){
          var current = this;
          this. onclick = function(event) {
            if (!event) event = window.event;
            var target = (event.target) ?
                         event.target : event.srcElement;
            say('For ' + current.tagName + '#'+ current.id +
                ' target is ' + target.id);
          }
        });
      });

      function say(text) {
        $('#console').append('<div>'+text+'</div>');
      }
    </script>
  </head>

  <body id="greatgrandpa">
    <div id="grandpa">
      <div id="pops">
        <img id="vstar" src="vstar.jpg"/>
      </div>
    </div>
    <div id="console"></div>
  </body>
</html>
```

❶ Selects every element on the page

❷ Applies onclick handler to every selected element

DOM Event Model - Level 2 (1/7)

- DOM Level 2 Event Model was designed to address problems like multiple handlers against events.
- It lets us set how event handlers, even multiple handlers, are established on DOM elements under this more advanced model.

Establishing events

- Rather than assigning a function reference to an element property, DOM Level 2 event handlers—also termed listeners—are established via an element method.
- Each DOM element defines a method named addEventListener() that's used to attach event handlers (listeners) to the element.

DOM Event Model - Level 2 (2/7)

- The format of this method is as follows:

```
addEventListener(eventType,listener,useCapture)
```

  - The eventType parameter is a string that identifies the type of event to be handled
  - This string is, generally, the same event names we used in the DOM Level 0 Event Model without the on prefix: for example, click, mouseover, keydown, and so on.
  - The listener parameter is a reference to the function (or inline function) that’s to be established as the handler for the named event type on the element.
  - As in the basic event model, the Event instance is passed to this function as its first parameter.
  - The final parameter, useCapture, is a Boolean value

DOM Event Model - Level 2 (3/7)

```
<html>
  <head>
    <title>DOM Level 2 Events Example</title>
    <script type="text/javascript"
            src="../scripts/jquery-1.2.1.js">
    </script>
    <script type="text/javascript">
      $(function(){
        var element = $('#vstar')[0];
        element.addEventListener('click',function(event) {
          say('Whee once!');
        },false);
        element.addEventListener('click',function(event) {
          say('Whee twice!');
        },false);
        element.addEventListener('click',function(event) {
          say('Whee three times!');
        },false);
      });

      function say(text) {
        $('#console').append('<div>'+text+'</div>');
      }
    </script>
  </head>

  <body>
    <img id="vstar" src="vstar.jpg"/>
    <div id="console"></div>
  </body>
</html>
```

❶ **Establishes three event handlers!**

DOM Event Model - Level 2 (4/7)

Event Propagation

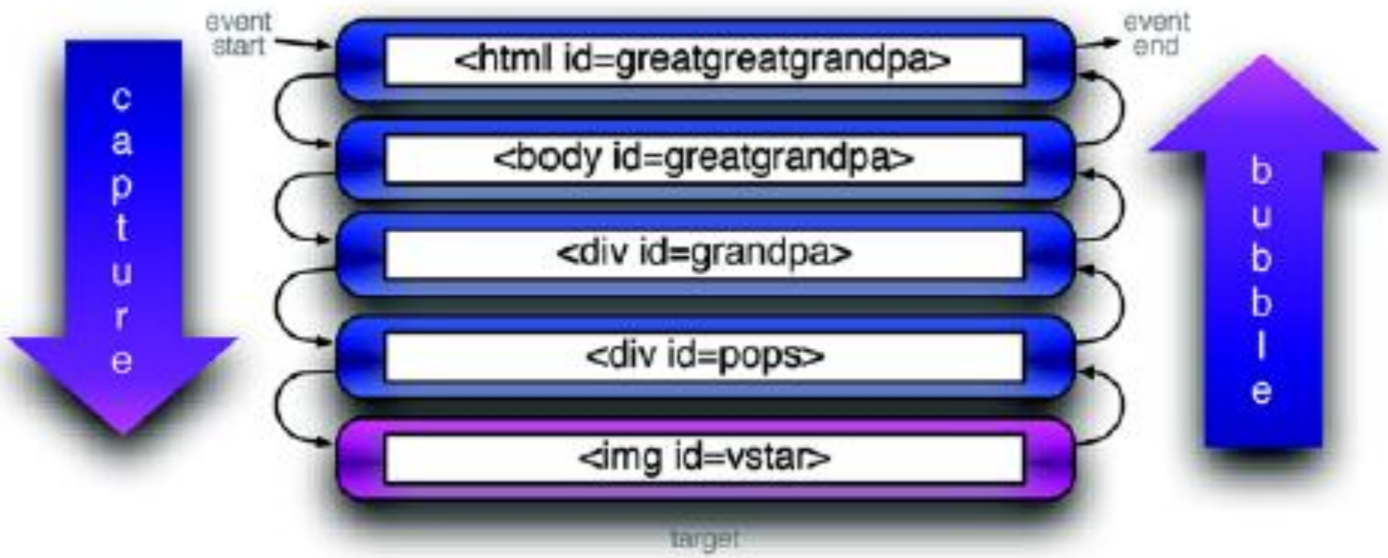
- We saw earlier that, with the Basic Event Model, once an event was triggered on an element the event propagated from the target element upwards in the DOM tree to all the target's ancestors.
- The advanced Level 2 Model also provides this bubbling phase but ups the ante with an additional phase: capture phase.
- Under the DOM Level 2 Event Model, when an event is triggered, the event first propagates from the root of the DOM tree down to the target element and then propagates again from the target element up to the DOM root.
- The former phase (root to target) is called capture phase, and the latter (target to root) is called bubble phase.

DOM Event Model - Level 2 (5/7)

Event Propagation

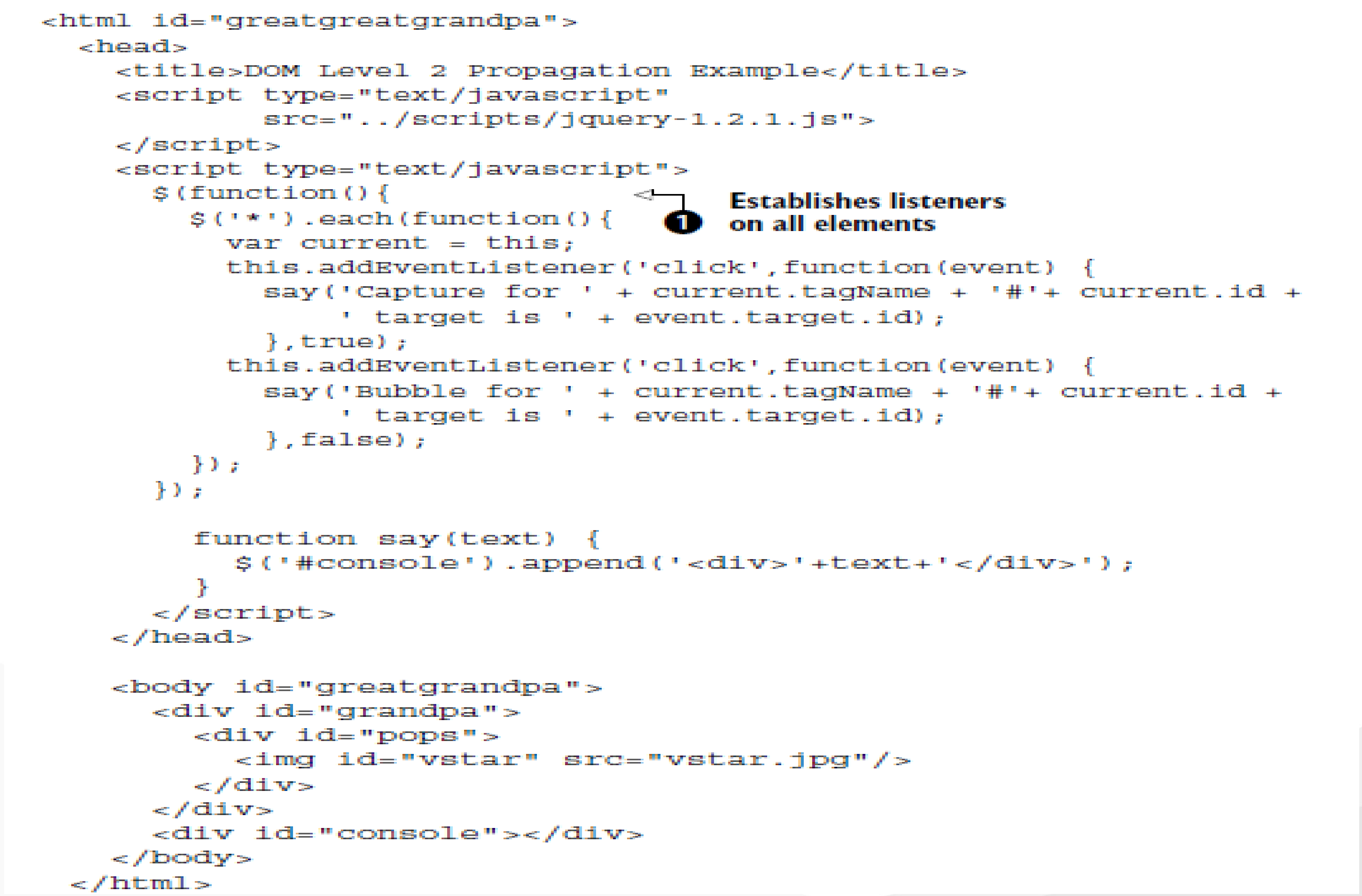
- When a function is established as an event handler, it can be flagged as a capture handler in which case it will be triggered during capture phase, or as a bubble handler to be triggered during bubble phase.
- As you might have guessed by this time, the useCapture parameter to addEventListener() identifies which type of handler is established.
- A value of false for this parameter establishes a bubble handler, whereas a value of true registers a capture handler.

DOM Event Model - Level 2 (6/7)



Propagation in the DOM Level 2 Event Model traverses the DOM hierarchy twice: once from top to target during capture phase and once from target to top during bubble phase.

DOM Event Model - Level 2 (7/7)



```
<html id="greatgreatgrandpa">
  <head>
    <title>DOM Level 2 Propagation Example</title>
    <script type="text/javascript"
            src="../scripts/jquery-1.2.1.js">
    </script>
    <script type="text/javascript">
      $(function(){
        $('*').each(function(){
          var current = this;
          this.addEventListener('click',function(event) {
            say('Capture for ' + current.tagName + '#'+ current.id +
                ' target is ' + event.target.id);
            },true);
          this.addEventListener('click',function(event) {
            say('Bubble for ' + current.tagName + '#'+ current.id +
                ' target is ' + event.target.id);
            },false);
        });
      });

      function say(text) {
        $('#console').append('<div>'+text+'</div>');
      }
    </script>
  </head>

  <body id="greatgrandpa">
    <div id="grandpa">
      <div id="pops">
        <img id="vstar" src="vstar.jpg"/>
      </div>
    </div>
    <div id="console"></div>
  </body>
</html>
```

1 Establishes listeners on all elements

Contents

JQuery Event Model (1/13)

- Although it’s true that the creation of Rich Internet Applications requires a hefty reliance on event handling, the thought of writing event-handling code on a large scale while dealing with the browser differences is enough to daunt even the most intrepid of page authors.
- JQuery’s event model addresses all the issues
- JQuery’s event model exhibits the following features:
  - Provides a unified method for establishing event handlers
  - Allows multiple handlers for each event type on each element
  - Uses standard event-type names: for example, click or mouseover
  - Makes the Event instance available as a parameter to the handlers
  - Normalizes the Event instance for the most often used properties
  - Provides unified methods for event canceling and default action blocking

JQuery Event Model (2/13)

Binding event handlers using JQuery

- Using the JQuery Event Model, we can establish event handlers on DOM elements with the bind() command.

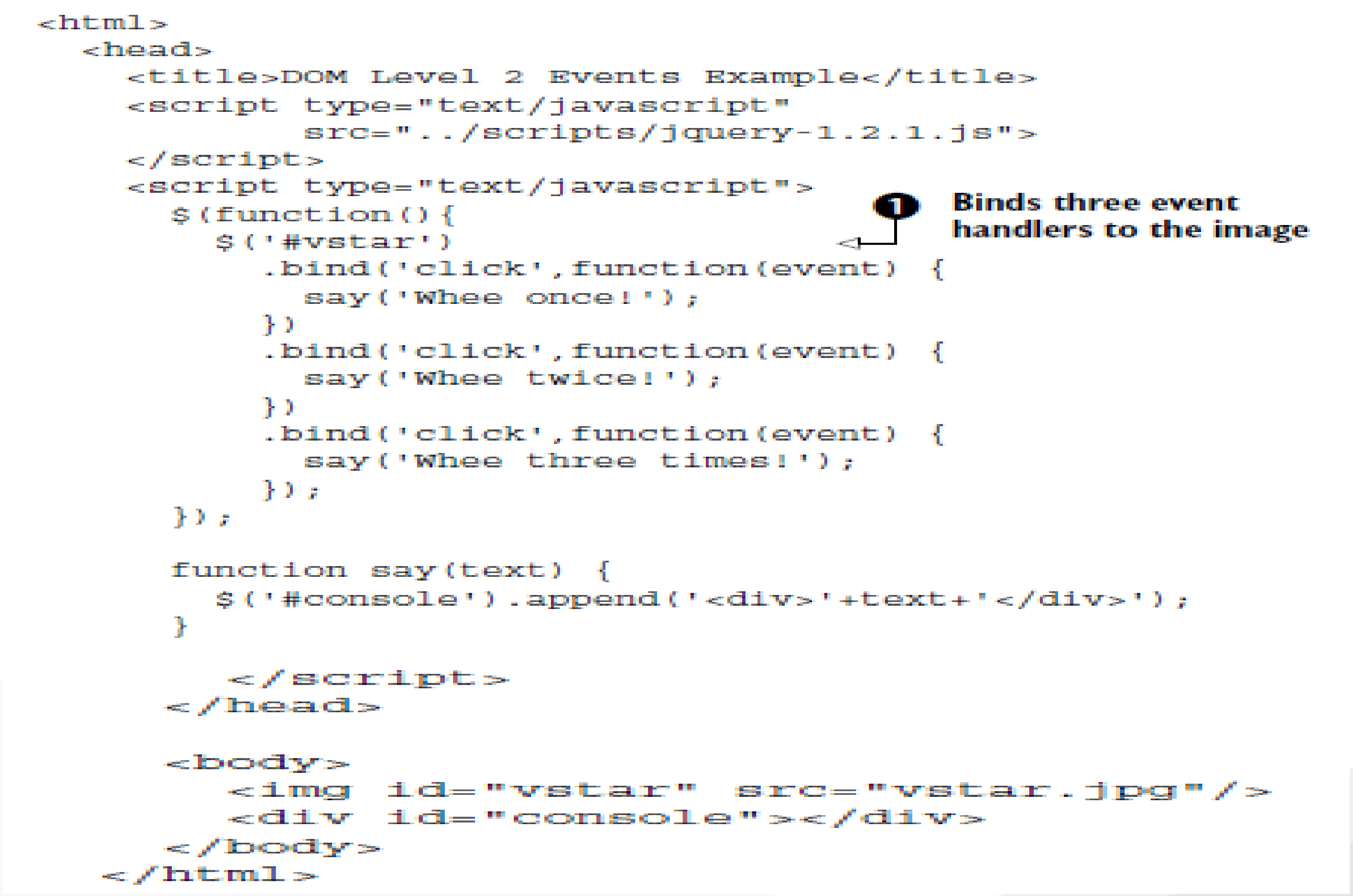
```
$('img').bind('click',function(event){alert('Hi there!');});
```

- This statement binds the supplied inline function as the click event handler for every image on a page.

Bind Syntax

Command	Parameter	Returns
bind(eventType,data,listener) • Establishes a function as the event handler for the specified event type on all elements in the matched set.	**eventType** (String) • Specifies the name of the event type for which the handler is to be established. **data** (Object) • Caller-supplied data that's attached to the event instance as a property named data for availability to the handler functions. **listener** (Function) • The function that's to be established as event handler.	• The wrapped set.

JQuery Event Model (3/13)



```
<html>
  <head>
    <title>DOM Level 2 Events Example</title>
    <script type="text/javascript"
            src="../scripts/jquery-1.2.1.js">
    </script>
    <script type="text/javascript">
      $(function(){
        $('#vstar')
          .bind('click',function(event) {
            say('Whee once!');
          })
          .bind('click',function(event) {
            say('Whee twice!');
          })
          .bind('click',function(event) {
            say('Whee three times!');
          });
      });

      function say(text) {
        $('#console').append('<div>'+text+'</div>');
      }

        </script>
      </head>

      <body>
        <img id="vstar" src="vstar.jpg"/>
        <div id="console"></div>
      </body>
    </html>
```

1 Binds three event handlers to the image

JQuery Event Model (4/13)

- Another little nifty event handling extra that JQuery provides for us is the ability to group event handlers by assigning them to a namespace.
- Unlike conventional namespacing (which assigns namespaces via a prefix), the event names are namespaced by adding a suffix to the event name separated by a period character.
- By grouping event bindings in this way, we can easily act upon them later as a unit.
- Take, for example, a page that has two modes: a display mode and an edit mode.
- When in edit mode, event listeners are placed on many of the page elements, but these listeners are not appropriate for display mode and need to be removed when the page transitions out of edit mode.
- We could namespace the edit mode events with code such as

JQuery Event Model (5/13)

Examples	Explanations
$('#thing1').bind('click.editMode',someListener); $('#thing2').bind('click.editMode',someOtherListener); ... $('#thingN').bind('click.editMode',stillAnotherListener);	• By grouping all these bindings into a namespace named editMode, we can later operate upon them as a whole. • For example, when the page leaves edit mode and it comes time to remove all the bindings we could do this easily.
$('*').unbind('click.editMode');	• This will remove all click bindings in the namespace editMode for all elements on the page.

JQuery Event Model (6/13)

- JQuery also provides a specialized version of the bind() command, named one(), that establishes an event handler as a one-shot deal.
- Once the event handler executes the first time, it's automatically removed as an event handler.

One syntax

Command	Parameter	Returns
one(eventType,data,listener) • Establishes a function as the event handler for the specified event type on all elements in the matched set. • Once executed, the handler is automatically removed.	**eventType** (String) • Specifies the name of the event type for which the handler is to be established. **data** (Object) • Caller-supplied data that's attached to the Event instance for availability to the handler functions. **listener** (Function) • The function that's to be established as the event handler.	• The wrapped set.

JQuery Event Model (7/13)

Removing Event Handler

- Unbind() is used to remove the binding

Unbind syntax

Command	Parameter	Returns
unbind(eventType,listener) • Removes events handlers from all elements of the wrapped set as specified by the optional passed parameters. • If no parameters are provided, all listeners are removed from the elements.	**eventType** (String) • If provided, specifies that only listeners established for the specified event type are to be removed. **listener** (Function) • If provided, identifies the specific listener that's to be removed. **event** (Event) • Removes the listener that triggered the event described by this Event instance.	• The wrapped set.

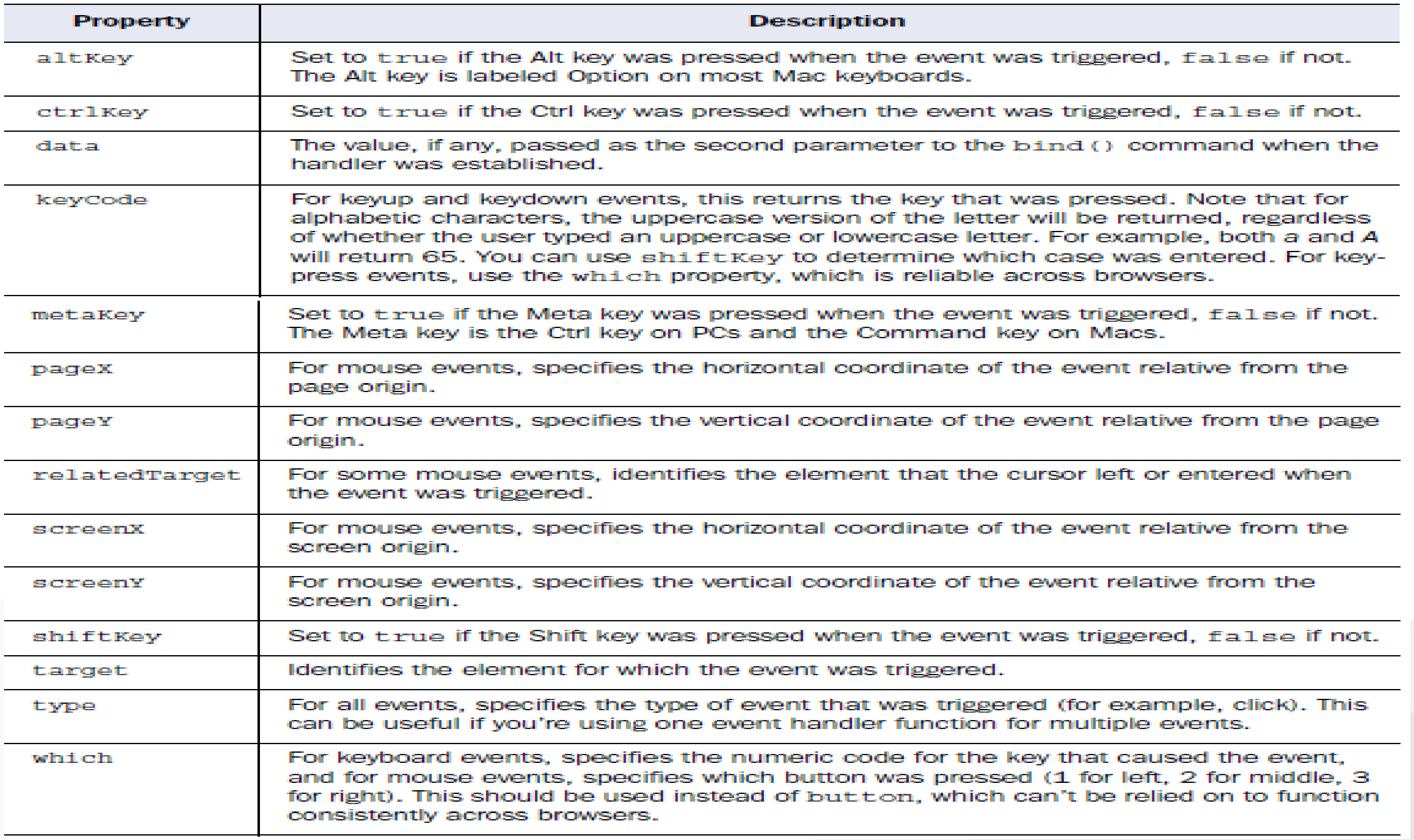
JQuery Event Model (8/13)

Inspecting the Event instance

- When an event handler established with the bind() command is invoked, the Event instance is passed to it as the first parameter to the function.
- This eliminates the need to worry about the window.event property under Internet Explorer, but what about accessing the divergent properties of the Event instance?
- Even when using JQuery to establish handlers, the Event instance passed to the event handler is a clone of the native object as defined by the browser.
- That means that in standards-compliant browsers, the Event instance will follow the standardized layout of properties, and under Internet Explorer, the instance will use the proprietary layout.
- Before the proprietary instance is passed to the event handler, JQuery does its best to fix up the object so that the most commonly accessed properties and methods of that object follow the standardized format.
- So once again, except for the most obscure of Event properties, we can write the code for our event handlers without regard for browser platform.

JQuery Event Model (9/13)

Inspecting the Event instance



Property	Description
`altKey`	Set to `true` if the Alt key was pressed when the event was triggered, `false` if not. The Alt key is labeled Option on most Mac keyboards.
`ctrlKey`	Set to `true` if the Ctrl key was pressed when the event was triggered, `false` if not.
`data`	The value, if any, passed as the second parameter to the `bind()` command when the handler was established.
`keyCode`	For keyup and keydown events, this returns the key that was pressed. Note that for alphabetic characters, the uppercase version of the letter will be returned, regardless of whether the user typed an uppercase or lowercase letter. For example, both a and *A* will return 65. You can use `shiftKey` to determine which case was entered. For key-press events, use the `which` property, which is reliable across browsers.
`metaKey`	Set to `true` if the Meta key was pressed when the event was triggered, `false` if not. The Meta key is the Ctrl key on PCs and the Command key on Macs.
`pageX`	For mouse events, specifies the horizontal coordinate of the event relative from the page origin.
`pageY`	For mouse events, specifies the vertical coordinate of the event relative from the page origin.
`relatedTarget`	For some mouse events, identifies the element that the cursor left or entered when the event was triggered.
`screenX`	For mouse events, specifies the horizontal coordinate of the event relative from the screen origin.
`screenY`	For mouse events, specifies the vertical coordinate of the event relative from the screen origin.
`shiftKey`	Set to `true` if the Shift key was pressed when the event was triggered, `false` if not.
`target`	Identifies the element for which the event was triggered.
`type`	For all events, specifies the type of event that was triggered (for example, click). This can be useful if you're using one event handler function for multiple events.
`which`	For keyboard events, specifies the numeric code for the key that caused the event, and for mouse events, specifies which button was pressed (1 for left, 2 for middle, 3 for right). This should be used instead of `button`, which can't be relied on to function consistently across browsers.

JQuery Event Model (10/13)

Other event-related commands

- There are often interaction styles that are commonly applied to pages in Rich Internet Applications and are implemented using combinations of behaviors.
- JQuery provides a few event-related convenience commands that make it easier to use these interaction behaviors on our pages.

Toggling listeners

- The first of these is the toggle() command, which establishes a pair of click event handlers that swap off with each other on every other click event.

JQuery Event Model (11/13)

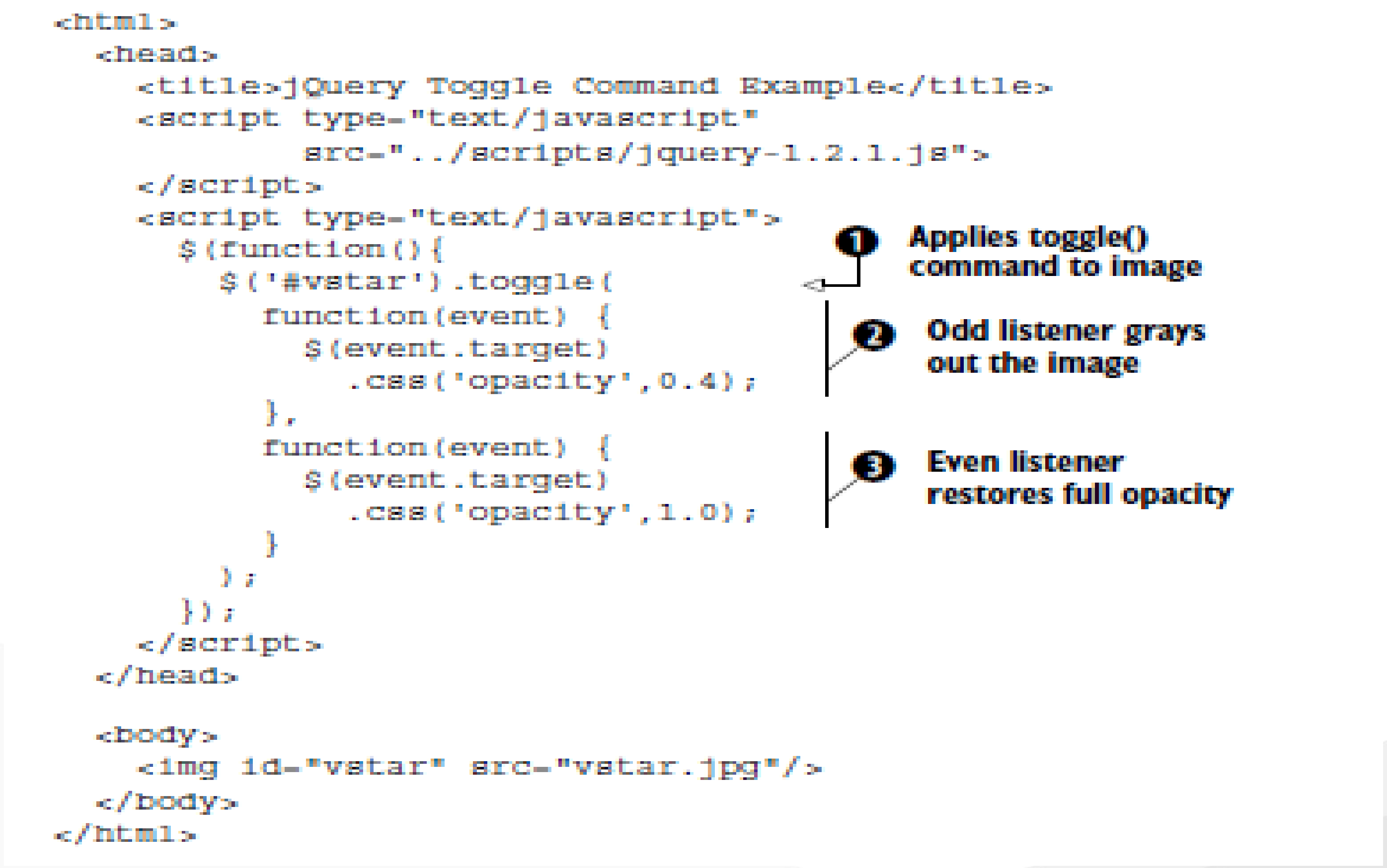
Toggle Syntax

Command	Parameter	Returns
Toggle • Establishes the passed functions as click event handlers on all elements of the wrapped set that toggle between each other with every other trigger of a click event	**listenerOdd** (Function) • A function that serves as the click event handler for all oddnumbered Clicks**.** **listenerEven** (Function) • A function that serves as the click event handler for all evennumbered clicks	• The wrapped set.

- A common use for this convenience command is to toggle the enabled state of an element based upon how many times it has been clicked.

JQuery Event Model (12/13)

Other event-related commands



```
<html>
  <head>
    <title>jQuery Toggle Command Example</title>
    <script type="text/javascript"
            src="../scripts/jquery-1.2.1.js">
    </script>
    <script type="text/javascript">
      $(function(){
        $('#vstar').toggle(
          function(event) {
            $(event.target)
              .css('opacity',0.4);
          },
          function(event) {
            $(event.target)
              .css('opacity',1.0);
          }
        );
      });
    </script>
  </head>

  <body>
    <img id="vstar" src="vstar.jpg"/>
  </body>
</html>
```

1 Applies toggle() command to image

2 Odd listener grays out the image

3 Even listener restores full opacity

JQuery Event Model (13/13)

Hovering Over Elements

- Events that inform us when the mouse pointer has entered an area, as well as when it has left that area, are essential to building many of the user interface elements that are commonly presented to users on our pages.

Hover Syntax

Command	Parameter	Returns
hover(overListener,outListener) • Establishes handlers for the mouseover and mouseout events for matched elements. • These handlers only fire when the area covered by the elements is entered and exited, ignoring transitions to child elements.	**overListener** (Function) • The function to become the mouseover handler. **outListener** (Function) • The function to become the mouseout handler.	• The wrapped set.

CurioCT - CitiusTech's Technology Q & A Forum

- In case of any questions please log on to http://interct/SitePages/CurioCTTechnology.aspx

CitiusTech

THANK YOU