Web Technologies II

Introducing JQuery – Part B

Advanced Selectors

jQuery also lets you use more complicated CSS selectors to accurately pinpoint the tags you wish to select

- Descendant selectors

provide a way to target a tag inside another tag. For example, say you've created an unordered list of links and added an ID name of navBar to the list's <ul> tag like this: <ul id="navBar">. The jQuery expression $('a') selects all <a> tags on the page. However, if you want to select only the links inside the unordered list, you use a descendant selector like this:

```
$('#navBar a')
```

- Child selectors

target a tag that's the child of another tag. A child tag is the direct descendant of another tag. For example, in the HTML diagrammed in previous slides the <h1> and <p> tags are children of the <body> tag, but the <strong> tag is not (because it's wrapped by the <p> tag).

Advanced Selectors

jQuery also lets you use more complicated CSS selectors to accurately pinpoint the tags you wish to select

- Descendant selectors

provide a way to target a tag inside another tag. For example, say you've created an unordered list of links and added an ID name of navBar to the list's <ul> tag like this: <ul id="navBar">. The jQuery expression $('a') selects all <a> tags on the page. However, if you want to select only the links inside the unordered list, you use a descendant selector like this:

$('#navBar a')

- Child selectors

target a tag that's the child of another tag. A child tag is the direct descendant of another tag. For example, in the HTML diagrammed in previous slides the <h1> and <p> tags are children of the <body> tag, but the <strong> tag is not (because it's wrapped by the <p> tag).

You create a child selector by first listing the parent element, followed by a >, and then the child element. For example, to select <p> tags that are the children of the <body> tag, you'd write this:

```
$('body > p')
```

Adjacent sibling

selectors let you select a tag that appears directly after another tag. For example, say you have an invisible panel that appears when you click a tab. In your HTML, the tab might be represented by a heading tag (say <h2>), while the hidden panel is a <div> tag that follows the header. To make the <div> tag (the panel) visible, you'll need a way to select it. You can easily do so with jQuery and an adjacent sibling selector:

```
$('h2 + div')
```

Attribute selectors

let you select elements based on whether the element has a particular attribute, and even check to make sure the attribute matches a specific value. With an attribute selector, you can find <img> tags that have the alt attribute set, or even match an <img> tag that has a particular alt text value.

You add the attribute selector after the name of the element whose attribute you're checking. For example, to find <img> tags that have the alt attribute set, you write this:

$('img[alt]')

There are a handful of different attribute selectors:

[attribute] selects elements that have the specified attribute assigned in the HTML. For example, $('a[href]') locates all <a> tags that have an href attribute set

[attribute="value"] selects elements that have a particular attribute with a specific value. For example, to find all text boxes in a form, you can use this:

$('input[type="text"]')

[attribute^="value"] matches elements with an attribute that begins with a specific value. For example, if you want to find links that point outside your site, you can use this code:

$('a[href^="http://"]')

[attribute$="value"] matches elements whose attribute ends with a specific value, which is great for matching file extensions. For example, with this selector, you can locate links that point to PDF files

$('a[href$=".pdf"]')

[attribute*="value"] matches elements whose attribute contains a specific value anywhere in the attribute

$('a[href*="amazon.ca"]')

jQuery Filters

jQuery also provides a way to filter your selections based on certain characteristics.For example, the :even filter lets you select every even element in a collection. In addition, you can find elements that contain particular tags, specific text, elements that are hidden from view, and even elements that do not match a particular selector.

To use a filter, you add a colon followed by the filter's name after the main selector. For example, to find every even row of a table, write your jQuery selector like this:

```
$('tr:even')
```

This code selects every even <tr> tag. To narrow down the selection, find every even table row in a table with class name of striped. You can do that like this:

```
$('.striped tr:even')
```

- :even and :odd select every other element in a group.
- :first and :last select the first or the last element in a group.
- You can use :not() to find elements that don't match a particular selector type

:has() finds elements that contain another selector.

```
$('li:has(a)')
```

:contains() finds elements that contain specific text

```
$('a:contains(Click Me!)')
```

:hidden locates elements that are hidden

```
$('div:hidden').show();
```

:visible is the opposite of :hidden

CHAINING FUNCTIONS

Sometimes you'll want to perform several operations on a of elements. For example, say you want to set the width and height of a <div> tag (with an ID of popUp) using JavaScript. Normally, you'd have to write at least two lines of code. But jQuery lets you do it with a single line:

```
$('#popUp').width(300).height(300);
```

jQuery uses a useful principle called chaining , which lets you add functions one after the other. Each function is connected to the next by a period, and operates on the same jQuery collection of elements as the previous function

Adding Content to a Page

jQuery provides many functions for manipulating elements and content on a page, from simply replacing HTML, to precisely positioning new HTML in relation to a selected element, to completely removing tags and content from the page.

To study the following examples of these functions, assume you have a page with the following HTML:

```
<div id="container">
        <div id="errors">
                <h2>Errors:</h2>
        </div>
</div>
```

Here are the five most useful jQuery functions for manipulating content on a page:

.html() can both read the current HTML inside an element and replace the current contents with some other HTML.

```
alert($('#errors').html());
$('#errors').html('<p>There are four errors in this form</p>');
```

.text() works like .html() but it doesn’t accept HTML tags.

It’s useful when you want to replace the text within a tag

$('#errors h2').text('No errors found');

.append() adds HTML as the last child element of the selected element

$('#errors').append('<p>There are four errors in this form</p>');

.prepend() is just like .append(), but adds HTML directly after the opening tag for the selection.

$('#errors').prepend('<p>There are four errors in this form</p>')

If you want to add HTML just outside of a selection, either before the selected element’s opening tag or directly after the element’s closing tag, use the .before() or .after() functions.

Replacing and Removing Selections

At times you may want to completely replace or remove a selected element

$('#popup').remove();

The .remove() function isn’t limited to just a single element. Say you want to remove all <span> tags that have a class of error; you can do this: $('span.error').remove();

Setting and Reading Tag Attributes

Adding, removing, and changing elements isn’t the only thing JQuery is good at, and it’s not the only thing you’ll want to do with a selection of elements. You’ll often want to change the value of an element’s attribute—add a class to a tag, for example, or change a CSS property of an element

Classes

Cascading Style Sheets are a very powerful technology, letting you add all sorts of sophisticated visual formatting to your HTML. Because web browsers process and implement CSS instructions very quickly and efficiently, simply adding a class to a tag can completely change that tag’s appearance—even make it disappear from the page.

jQuery provides several functions for manipulating a tag’s class attribute:

addClass() adds a specified class to an element.

$('a[href^="http://"]').addClass('externalLink');

<a href="http://www.oreilly.com/">

Becomes -><a href="http://www.oreilly.com/" Class= "external- --Link">

removeClass() is the opposite of addClass(). It removes the specified class from the selected elements. For example, if you wanted to remove a class named highlight from a <div> with an ID of alertBox, you'd do this:

```
$('#alertBox').removeClass('highlight');
```

Reading and Changing CSS Properties

jQuery's css() function also lets you directly change CSS properties of an element, so instead of simply applying a class style to an element, you can immediately add a border or background color, or set a width or positioning property.

To determine the current value of a CSS property, pass the name of the property to the css() function. For example, say you want to find the background color of a <div> tag with an ID of main:

```
var bgColor = $('#main').css('background-color');
```

The css() function also lets you set a CSS property for an element. To use the function this way, you supply two arguments to the function: the CSS property name and a value.

```
$('body').css('font-size', '200%');
```

If you want to change more than one CSS property on an element, you don’t need to resort to multiple uses of the .css() function

```
$('#highlightedDiv').css('background-color','#FF0000');
$('#highlightedDiv').css('border','2px solid #FE0037');
```

Another way is to pass what’s called an object literal to the .css() function

```
{ 'background-color' : '#FF0000', 'border' : '2px solid #FE0037' }
```

Acting on Each Element in a Selection

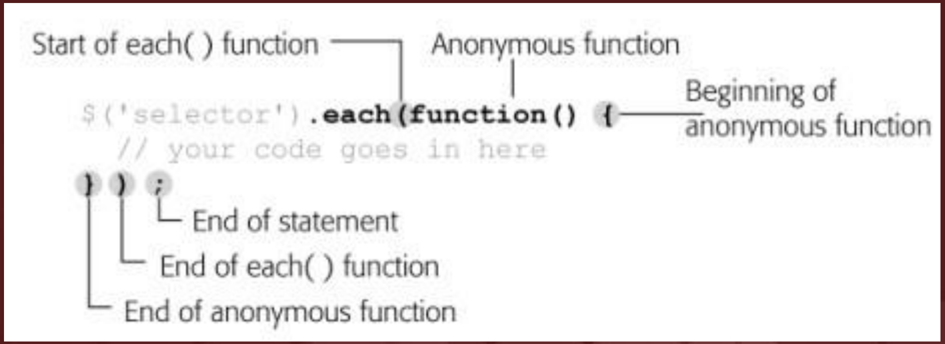
One of the unique qualities of jQuery is that most of its functions automatically loop through each item in a jQuery selection. For example, to make every <img> on a page fade out, you only need one line of JavaScript code:

```
$('img').fadeOut();
```

There are plenty of times when you’ll want to loop through a selection of elements and perform a series of actions on each element. jQuery provides the .each() function for just this purpose.

For example

```
$('selector').each(function() {
// code goes in here
});
```


Start of each( ) function
Anonymous function
Beginning of anonymous function
$('selector').each(function() {
// your code goes in here
});
End of statement
End of each( ) function
End of anonymous function

this and $(this)

The this keyword refers to whatever element is calling the anonymous function. So the first time through the loop, this refers to the first element in the jQuery selection, while the second time through the loop, this refers to the second element.

Questions ?