Web 2.0 應用技術-CSS, JavaScript 及 HTML

CM310.1-03-2013-C

Lesson 3

Outline

- Traversing and Manipulating
- Events
- Effects
- Ajax

HTML vocabulary

```
<ul>
   <li>
      <span>
         <i>Foo</i>
      </span>
   </li>
   <li>Bar</li>
</ul>
```

- The first list item is a child of the unordered list.
- The unordered list is the parent of both list items.
- The span is a descendant of the unordered list.
- The unordered list is an ancestor of everything inside of it.
- The two list items are siblings.

Travesal

- Filtering Selections
- Finding elements relative to a selection

http://api.jquery.com/category/traversing/

Filtering Selection

```
var listItems = $( 'li' );

// filter the selection to only items with a class of 'special'
var special = listItems.filter( '.special' );

// filter the selection to only items without a class of 'special'
var notSpecial = listItems.not( '.special' );

// filter the selection to only items that contain a span
var hasSpans = listItems.has( 'span' );
```

Finding elements relative to a selection

```
// get the first list item on the page
var listItem = $( 'li' ).first(); // also: .last()

// get the siblings of the list item
var siblings = listItem.siblings();

// get the next sibling of the list item
var nextSibling = listItem.next(); // also: .prev()

// get the list item's parent
var list = listItem.parent();
```

```
// get the list items that are immediate children of the list
var listItems = list.children();

// get ALL list items in the list, including nested ones
var allListItems = list.find( 'li' );

// find all ancestors of the list item that have a class of "module"
var modules = listItem.parents( '.module' );

// find the closest ancestor of the list item that has a class of "module"
var module = listItem.closest( '.module' );
```

Manipulating Style

- Querying CSS

  .css; function(a,c){if(arguments...

  .css(); TypeError: Cannot call method

  .css(“font-size”); ”16px”

- Setting CSS Explicitly

  $(“p”).css(“font-size”,“24px”); [<p>...

Setting CSS Relatively

var original = $(“p”).css(“font-size”);

$(“p”).css(“font-size”, original + 8);

No actual change!

$(“p”).css(“font-size”, original + 8 + “px”);

Setting Multiple Attributes

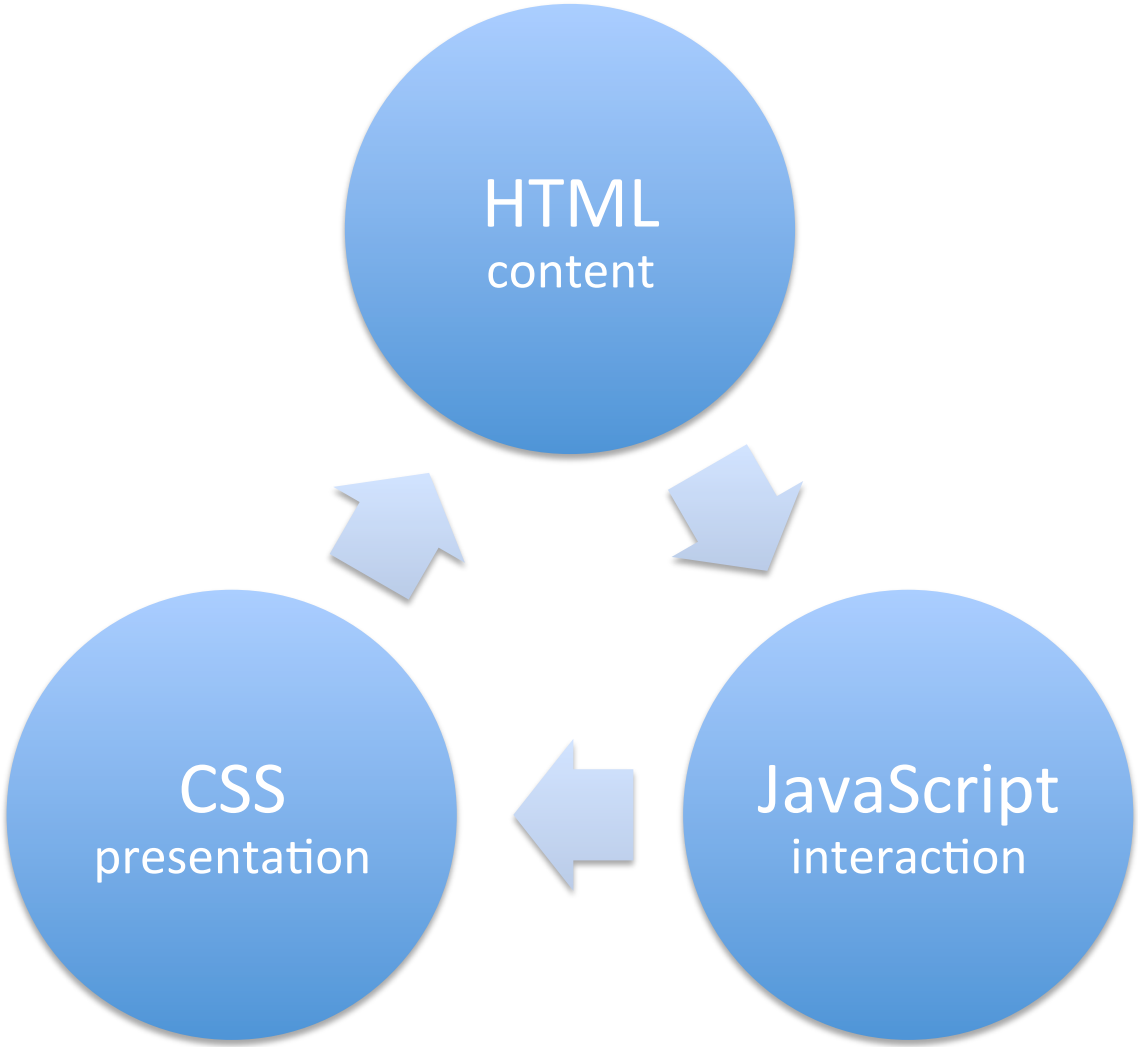
- Using Method Chaining

```
$("p").css("font-size","24px").
        css("font-weight","bold").
        css("line-height","32px");
```

- Using a Map

```
$("p").css({"font-size":"24px",
            "font-weight":"bold",
            "line-height":"32px"});
```

WAIT!!



The CSS Interface

- DOM and CSS communicate via IDs and Classes
- Manipulate IDs and Classes
- More reliable and maintainable

CSS Class

- Querying CSS Class

$(“p:first”).hasClass(“plan”);

true / false

- Adding A CSS Class

$(“p:first”).addClass(“bigger”);

- Removing A CSS Class

```
$(“p:first”).removeClass(“bigger”);
```

- Removing all CSS Classes

```
$(“p:first”).removeClass();
```

- Toggling a Class

```
$(“p:first”).toggleClass(“bigger”);
```

Visibility

$(“p:first”).hide();

...

$(“p:first”).show();

...

$(“p:first”).toggle();

switches between show and hide

Manipulating Content

- Setting Text Content
  $(“p#time span.label”).text(“Departs At:”);
  [Departs At:”]

- Text is just Text
  $(“p#time”).text();
  “Departs At: 10:36AM”

$(“p#time”).text(“Departs At: 10:36AM”);

[Departs At: 10:36AM]

Queries & Setting with html()

$(“p#time”).html();

“Departs At:10:36AM”

$(“p#time”).html(“Departs At: 10:36AM”);

“Departs At: 10:36AM”

Prepend & Append

$(“span.label”).prepend(“Departure”);

“Departure Time: 10:36AM”

$(“p#time”).append(“ from the Gate”);

“ Departure Time: 10:36AM from the Gate ”

Inserting siblings with before & after

```
$(“p#time”).before(“<h3>Departure Details</
h3>”);
```

```
$( "h1" ).after(“<h3>Departure Details</h3>”);
```

Manipulating Nodes

- Removing Nodes

  $("span.label").remove();

- Moving Nodes

  var gate = $("p.gate").remove();

  $("p#time").before(gate);

Review

function	without parameter	with parameter
.text	query the plain text	set the plain text
.html	query html and text	set html and text
	description	
.prepend	add element inside the beginning of the selection	
.append	add element inside the end of the selection	
.before	add element before selection	
.after	add element after selection	
.remove	remove element from the DOM	

jQuery Events

- jQuery provides simple methods for attaching event handlers to selections. When an event occurs, the provided function is executed. Inside the function, this refers to the element that was clicked
- http://api.jquery.com/category/events/

Connecting Events to Elements

Binding an event

Binding an event

```
$('#foo').bind('click', function() {
    alert('User clicked on "foo."');

});
```

Convenience methods for most common events

```
$('p').click(function() {
    console.log
            ('click');
```

Disconnecting Events

- To disconnect an event handler, you use the $.fn.unbind method and pass in the event type to unbind. If you attached a named function to the event, then you can isolate the unbinding to that named function by passing it as the second argument.

- Unbinding all click handlers on a selection

```
$('p').unbind('click');
```

- Unbinding a particular click handler

```
var foo = function() { console.log('foo'); };
var bar = function() { console.log('bar'); };
$('p').bind('click', foo).bind('click', bar);
$('p').unbind('click', bar); // foo is still bound to the click event
```

Mouse Events

- http://api.jquery.com/category/events/mouse-events/

- .click() / .dblclick()
- .hover()
- .mousedown() / .mouseup() // click and release mouse click
- .mouseover() / .mouseout() // cursor move around and move outside
- .mouseenter() / .mouseleave() // cursor go inside and outside
- .mousemove()

Inside the Event Handling Function

```
$('a').click(function(e) {
        $this = $(this);
    if ($this.attr('href').match('evil')) {
        e.preventDefault();
        $this.addClass('evil');
    }
});
```

Hover

The .hover() method binds handlers for both mouseenter and mouseleave events.

```
$('#menu li').hover(function() {
	$(this).toggleClass('hover');
});

// same thing:
$('#menu li').hover(
	function() {
		$(this).addClass('hover');
	}, function() {
		$(this).removeClass('hover');
	}
);
```

Form Events

- http://api.jquery.com/category/events/form-events/

- .blur() / .focus() // loses focus or gains focus
- .change() // value changed
- .select() // makes a text selection
- .submit() // submit a form

Browser Events

- http://api.jquery.com/category/events/browser-events/

- .resize() // the size of the browser window changed
- .scroll() // scrolls to a different place in the element

Document Loading

- http://api.jquery.com/ready/

- .ready() // when the DOM is fully loaded.

Triggering Event Handlers

- Trigger the event handlers bound to an element without any user interaction via the $.fn.trigger method

- $('#foo').bind('click', function() {
- alert($(this).text());
- });

- $('#foo').trigger('click');

Handling future events

- .one() // the handler is unbound after its first invocation
- .on() / .off()

Effects

- jQuery makes it trivial to add simple effects to your page
- http://api.jquery.com/category/effects/

Built-in Effects

- .show() / .hide()
- .fadeIn() / .fadeOut()
- .fadeTo()
- .toggle()
- .slideUp() / .slideDown()
- .slideToggle()

- $('h1').show();

Changing the Duration of Built-in Effects

```
$('h1').fadeIn(300); // fade in over 300ms
$('h1').fadeOut('slow'); // using a built-in speed definition

speeds: {
    slow: 600,
    fast: 200,
    // Default speed
    _default: 400
}
```

- Augmenting jQuery.fx.speeds with custom speed definitions

```
jQuery.fx.speeds.blazing = 100;
jQuery.fx.speeds.turtle = 2000;
```

Custom Effects with $.fn.animate

jQuery makes it possible to animate arbitrary CSS properties via the $.fn.animate method. The $.fn.animate method lets you animate to a set value, or to a value relative to the current value.

```
<style>
div.funtimes{
position:absolute;
background-color:#abc;
left:50px;
width:90px;
height:90px;
margin:5px;
}
</style>
$('div.funtimes').animate(
{
left : "+=50px",
opacity : 0.25
},
300, // duration
function() { console.log('done!'); // calback
});
```

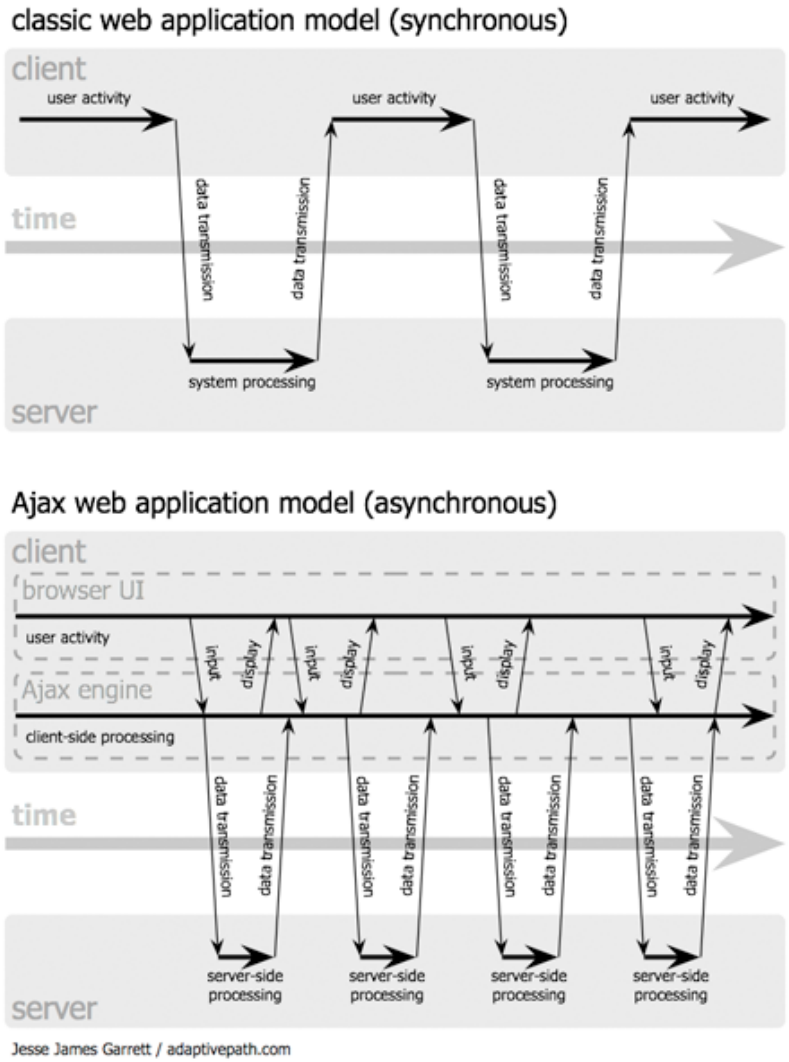
Custom Effects with $.fn.animate

- Easing - jQuery includes only two methods of easing: swing and linear. If you want more natural transitions in your animations, various easing plugins are available.
- http://jqueryui.com/demos/effect/easing.html
- It is possible to do per-property easing when using the $.fn.animate method.
- $('div.funtimes').animate(
  - –{
    - •left : ["+=50", "swing”],
    - •opacity : [0.25, "linear"]
  - –},
  - –300
-);

Managing Effects

- jQuery provides several tools for managing animations
- .stop() // Stop currently running animations on the selected elements.
- .delay() // Wait the specified number of milliseconds before running the next animation.
- $('h1').show(300).delay(1000).hide(300);
- jQuery.fx.off // If this value is true, there will be no transition for animations; elements will immediately be set to the target final state instead.
- jQuery.fx.off = true;

Ajax (Asynchronous JavaScript and XML)



Ajax Overview

- Most jQuery applications don't in fact use XML, despite the name "Ajax"; instead, they transport data as plain HTML or JSON (JavaScript Object Notation).
- Limitation: Ajax does not work across domains.

Ajax Methods

- jQuery provides Ajax support that abstracts away painful browser differences. It offers both a full featured $.ajax() method

- Simple convenience methods, such as
- $.get() / $.post()
- $.getScript()
- $.getJSON()
- $.fn.load()

Ajax Examples

```
// get plain text or html
$.get('/users.php', { userId : 1234 }, function(resp) {
    –console.log(resp);
});

// add a script to the page, then run a function defined in it
$.getScript('/static/js/myScript.js', function() {
    –functionFromMyScript();
});

// get JSON-formatted data from the server
$.getJSON('/details.php', function(resp) {
    –$.each(resp, function(k, v) {
        •console.log(k + ' : ' + v);
    –});
});
```

Same-Origin Policy and JSONP

- In general, Ajax requests are limited to the same protocol (http or https), the same port, and the same domain as the page making the request. This limitation does not apply to scripts that are loaded via jQuery's Ajax methods.
- The other exception is requests targeted at a JSONP service on another domain. In the case of JSONP, the provider of the service has agreed to respond to your request with a script that can be loaded into the page using a <script> tag, thus avoiding the same-origin limitation; that script will include the data you requested, wrapped in a callback function you provide:

- <script>

  –callback({ data: “here is the data you want from another domain”});

- </script>

Working with JSONP

- Using YQL and JSONP

```
$.ajax({
        –url : 'http://query.yahooapis.com/v1/public/yql',
        –// the name of the callback parameter,
        –// as specified by the YQL service
        –jsonp : 'callback',

        –// tell jQuery we're expecting JSONP
        –dataType : 'jsonp',

        –// tell YQL what we want and that we want JSON
        –data : {
            •q : 'select * from weather.forecast where woeid= 2165352',
            •format : 'json'
        –},

        –// work with the response
        –success : function(response) {
            console.log(response);
        –}
});
```

Ajax Events

- Often, you'll want to perform an operation whenever an Ajax requests starts or stops, such as showing or hiding a loading indicator. Rather than defining this behavior inside every Ajax request, you can bind Ajax events to elements just like you'd bind other events.

- http://docs.jquery.com/Ajax_Events

- Setting up a loading indicator using Ajax Events
- $('#loading_indicator').ajaxStart(function() { $(this).show(); })

  .ajaxStop(function() { $(this).hide(); });