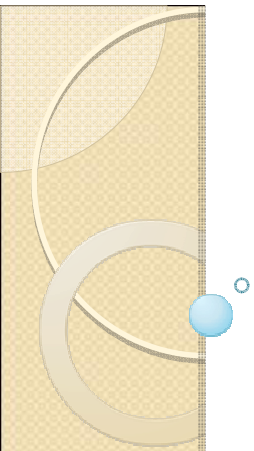

jQuery

CS 380: Web Programming

Downloading and using jQuery UI

```
<script
          src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"
          type="text/javascript"></script>
<script
          src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.8.21/jquery-ui.min.js"
          type="text/javascript"></script>
<!-- If you want the default ui widget stylings -->
<link
href="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.21/themes/ui-lightness/jquery-ui.css"
rel="stylesheet" type="text/css" />
```

- or download it, extract its .js files to your project folder
- documentation available on the jQuery UI API page
- the CSS is optional and only needed for widgets at the end

Looping over the DOM

- Using the DOM

```
var elems = document.querySelectorAll("li");
for (var i = 0; i < elems.length; i++) {
        var e = elems[i];
        // do stuff with e
}
```

- Using jQuery

```
$("li").each(function(idx, e) {
        // do stuff with e
});
```

Inside the jQuery each loop

```
$("li").each(function(idx, e) {
        // do stuff with e
});
```

- return false to exit the loop early
- e is a plain old DOM object
  - We can upgrade it again using $ if we want

```
$("li").each(function(idx, e) {
    e = $(e); // do stuff with e
});
```

Modifying DOM nodes

HTML attributes	DOM fields
title	.title
id	.id
class	.className
style="prop: value"	.style.prop = value

Getting/Setting CSS classes

```
function highlightField() {
        // turn text yellow and make it bigger
        var elem = document.getElementById("id");
        if (!elem.className) {
                elem.className = "highlight";
        } else if (elem.className.indexOf("invalid") < 0) {
                elem.className += " highlight";
        }
}
```

- JS DOM's className property corresponds to HTML class attribute
- somewhat clunky when dealing with multiple space-separated classes as one big string
- className is just a string, not an array like we would want

Getting/setting CSS classes in jQuery

```
function highlightField() {
        // turn text yellow and make it bigger
        if (!$("#myid").hasClass("invalid")) {
                $("#myid").addClass("highlight");
        }
}
```

- addClass, removeClass, hasClass, toggleClass ma nipulate CSS classes
- similar to existing className DOM property, but don't have to manually split by spaces

Adjusting styles with the DOM

```
<button id="clickme">Color Me</button>
window.onload = function() {
        document.getElementById("clickme").onclick = changeColor;
};
function changeColor() {
        var clickMe = document.getElementById("clickme");
        clickMe.style.color = "red";
}
```

Property	Description
style	lets you set any CSS style property for an element

Problems with reading/changing styles

```
<button id="clickme">Click Me</button>
window.onload = function() {
        document.getElementById("#clickme").onclick = biggerFont;
};
function biggerFont() {
        var size =
parseInt(document.getElementById("#clickme").style.fontSize);
        size += 4;
        document.getElementById("#clickMe").style.fontSize = s
ize + "pt";
}
```

- style property lets you set any CSS style for an element
- problem: you cannot (usually) read existing styles with it

Accessing styles in jQuery

```
function biggerFont() {
        // turn text yellow and make it bigger
        var size = parseInt($("#clickme").css("font-size"));
        $("#clickme").css("font-size", size + 4 + "pt");
}
```

- css function of the jQuery object allows reading pre-existing styles
- gives us the familiar font-size syntax instead of fontSize
- css(*property*) gets the property value, css(*property*, *value*) sets the property value

Exercise

- Find something with CSS and changing styles
- Write it with DOM and jQuery

jQuery method behavior

- Getters typically operate only on the first of the jQuery object's selected elements.

```
<ul>
        <li style="font-size: 10px">10px font size</li>
        <li style="font-size: 20px">20px font size</li>
        <li style="font-size: 30px">30px font size</li> </ul>

$("li").css("font-size"); // returns '10px'
```

- Setters typically operate on all of the selected DOM elements.

```
$("li").css("font-size", "15px");
// sets all selected elements to ‘15px'
<ul>
        <li style="font-size: 15px">10px font size</li>
        <li style="font-size: 15px">20px font size</li>
        <li style="font-size: 15px">30px font size</li>
</ul>
```

jQuery method parameters

- **getter syntax:**

```
$("#myid").css(propertyName);
```

- **setter syntax:**

```
$("#myid").css(propertyName, value);
```

- **multi-setter syntax:**

```
$("#myid").css({
        'propertyName1': value1,
        'propertyName2': value2,
        ...
        });
```

- **modifier syntax:**

```
$("#myid").css(propertyName, function(idx, oldValue) {
        return newValue;
});
```

jQuery method returns

method	return type
$("#myid");	jQuery object
$("#myid").children();	jQuery object
$("#myid").css("margin-left");	String
$("#myid").css("margin-left", "10px");	**jQuery object**
$("#myid").addClass("special");	**jQuery object**

More node manipulation with jQuery

jQuery method	functionality
.hide()	toggle CSS display: none on
.show()	toggle CSS display: none off
.empty()	remove everything inside the element, innerHTML = ""
.html()	get/set the innerHTML without escaping html tags
.text()	get/set the innerHTML, HTML escapes the text first
.val()	get/set the value of a form input, select, textarea, ...
.height()	get/set the height in pixels, returns a Number
.width()	get/set the width in pixels, return a Number

Creating new nodes

name	description
document.createElement("*tag*")	creates and returns a new empty DOM node representing an element of that type
document.createTextNode("*text*")	creates and returns a text node containing given text

```
// create a new <h2> node
var newHeading = document.createElement("h2");
newHeading.innerHTML = "This is a heading";
newHeading.style.color = "green";
```

Modifying the DOM tree

name	description
appendChild(*node*)	places given node at end of this node's child list
insertBefore(*new*, *old*)	places the given new node in this node's child list just before old child
removeChild(*node*)	removes given node from this node's child list
replaceChild(*new*, *old*)	replaces given child with new node

```
var p = document.createElement("p");
p.innerHTML = "A paragraph!";
document.getElementById("myid").appendChild(p);
```

Removing a node from the page

```
var bullets = document.getElementsByTagName("li");
for (var i = 0; i < bullets.length; i++) {
        if (bullets[i].innerHTML.indexOf("child") >= 0) {
                bullets[i].parentNode.removeChild(bullets[i]);
        }
}
```

jQuery manipulation methods

- http://api.jquery.com/category/manipulation/

Create nodes in jQuery

- The $ function to the rescue again

```
var newElement = $("<div>");

$("#myid").append(newElement);
```

- The previous example becomes with jQuery

```
$("li:contains('child')").remove();
```

Creating complex nodes in jQuery

- **The terrible way, this is no better than innerHTML hacking**

```
$("<p id='myid' class='special'>My paragraph is awesome!</p>")
```

- **The bad way, decent jQuery, but we can do better**

```
$("<p>")
        .attr("id", "myid")
        .addClass("special")
        .text("My paragraph is awesome!");
```

- **The good way**

```
$("<p>", {
        "id": "myid",
        "class": "special",
        "text": "My paragraph is awesome!"
});
```

jQuery $ function signatures

- Responding to the page ready event
  - `$(function);`
- Identifying elements
  - `$("selector", [context]);`
- Upgrading DOM
  - `elements=$(elements);`
- Creating new elements
  - `$("<html>", [properties]);`

Practice: Codeacademy

http://www.codecademy.com/courses/web-beginner-en-GfjC6/0?curriculum_id=50a3fad8c7a770b5fd0007a1#!/exercises/0

jQuery Visual Effects

Visual Effects

- Appear
  - show
  - fadeIn
  - slideDown
  - slide effect
- Disappear
  - hide
  - fadeOut
  - slideUp
  - Blind effect
  - Bounce effect
  - Clip effect
  - Drop effect
  - Explode effect
  - Drop effect
  - Explode effect
  - Fold effect
  - Puff effect
  - Size effect

Visual effects

- Getting attention
  - Highlight effect
  - Scale effect
  - Pulsate effect
  - Shake effect

Applying effects to an element

```
element.effect(); // for some effects

element.effect(effectName); // for most effects

$("#sidebar").slideUp();

// No need to loop over selected elements, as usual
$("#results > button").effect("pulsate");
```

- the effect will begin to animate on screen (asynchronously) the moment you call it
- One method is used behind the scenes to do most of the work, animate()

Effect options

```
element.effect(effectName, {

        option: value,
        option: value,
        ...
});

$("#myid").effect("explode", {
        "pieces": 25
});
```

Effects chaining

```
$('#demo_chaining')
      .effect('pulsate')
      .effect('highlight')
      .effect('explode');
```

- Effects can be chained like any other jQuery methods
- Effects are queued, meaning that they will wait until the previous effects finish

Effect duration

- You can specify how long an effect takes with the duration option
- Almost all effects support this option
- Can be one of slow, normal, fast or any number in milliseconds

```
$('#myid').effect('puff', {}, duration)
```

Custom effects - animate()

```
$('#myid').animate(properties, [duration]);
```

- You can animate any numeric property you want
- You can also animate these
  - color
  - background-color

```
$('#myid')
        .animate({
                'font-size': '80px',
                'color': 'green'
        }, 1000);
```

Custom effects easing

```
$('#myid')
        .animate(properties, [duration], [easing]);
```

- Your animations don't have to progress linearly
- There are many other options
  - slide
  - easeInSin

```
$('#myid')
        .animate({
                'font-size': '80px',
                'color': 'green'
        }, 1000, 'easeOutBounce');
```

Better Custom Effects* - toggleClass()

- * if you don't need easing or special options
- use the toggleClass method with its optional duration parameter

```
.special {
        font-size: 50px;
        color: red;
}
$('#myid').toggleClass('special', 3000);
```

Adding delay()

```
$('#myid')
        .effect('pulsate')
        .delay(1000)
        .slideUp()
        .delay(3000)
        .show('fast');
```

Effect complete event

```
$("#myid").effect('puff', [options], [duration], [function]);
```

- All effects can take a fourth optional callback parameter that is called when the animation ends
- the callback can use the this keyword as usual to address the element the effect was attached to

```
$('#myid').effect('clip', {}, 'default', function() {
      alert('finished');
});
```

Drag and drop

jQuery UI provides several methods for creating drag-and-drop functionality:

- Sortable : a list of items that can be reordered
- Draggable : an element that can be dragged
- Dropable : elements on which a Draggable can be dropped

Sortable

```
$('#myid ul').sortable([options]);
```

- specifies a list (ul, ol) as being able to be dragged into any order
- with some stylings you can get rid of the list look and sort any grouping of elements
- implemented internally using Draggables and Droppables
- to make a list un-sortable again, call .sortable('destroy') on the sortable element

Sortable

- options:
  - disabled
  - appendTo
  - axis
  - cancel
  - connectWith
  - containment
  - cursor
  - cursorAt
  - delay
  - distance
  - dropOnEmpty
  - forceHelperSize
  - opacity
  - revert
  - tolerance

Sortable demo

```
<ol id="simpsons">
        <li>Homer</li>
        <li>Marge</li>
        <li>Bart</li>
        <li>Lisa</li>
        <li>Maggie</li>
</ol>


$(function() {
        $("#simpsons").sortable();
});
```

Sortable list events

event	description
change	when any list item hovers over a new position while dragging
update	when a list item is dropped into a new position (more useful)

```
$(function() {
	$("simpsons").sortable({
		'update': function(event, ui) {
			// Do stuff here
		 }
	});
});
```

Sortable list events example

```
$(function() {
	$("#simpsons").sortable({
		'update': listUpdate
	});
});

function listUpdate(event, ui) {
	// can do anything I want here; effects,
	//an Ajax request, etc.
	ui.item.effect('shake');
}
```

Sortable "methods"

```
$('#my_list').sortable('methodName', [arguments]);

// Some examples
$('#my_list').sortable('destroy');
$('#my_list').sortable('option', 'cursor', 'pointer');
```

- jQuery plugins, like jQuery UI have an odd syntax for methods
- sortable methods
  - destroy
  - disable
  - enable
  - option
  - refresh
  - cancel

Draggable

```
$('#myid').draggable([options]);
```

- specifies an element as being able to be dragged

Draggable

- Options:
  - disabled
  - appendTo
  - addClasses
  - connectToSortable
  - delay
  - distance
  - grid
- Methods:
  - destroy
  - disable
  - enable
  - option
  - widget
- Events:
  - create
  - start
  - drag
  - stop

Draggable example

```
<div id="draggabledemo1">Draggable demo 1. Default options
</div>
<div id="draggabledemo2">Draggable demo 2.
      {'grid': [40,40], 'revert': true}
</div>


$('#draggabledemo1').draggable();
$('#draggabledemo2').draggable({
      'revert': true,
      'grid': [40, 40]
});
```

Droppable

```
$('#myid').droppable([options]);
```

- specifies an element as being able to receive draggables

Droppable

- Options:
  - disabled
  - accept
  - activeClass
  - hoverClass
  - scope
  - greedy
  - tolerance
- Methods:
  - destroy
  - disable
  - enable
  - option
  - widget
- Events:
  - create
  - over
  - out
  - **drop**
  - activate
  - deactivate

Drag/drop shopping demo

```
<img id="shirt" src="images/shirt.png" alt="shirt" />
<img id="cup" src="images/cup.png" alt="cup" />
<div id="droptarget"></div>


$('#shirt').draggable();
$('#cup').draggable();
$('#droptarget').droppable({
        'drop': productDrop
});

function productDrop(event, ui) {
        alert("You dropped " + ui.item.attr('id'));
}
```

Auto-completing text fields

Scriptaculous offers ways to make a text box that auto-completes based on prefix strings :

- Local Autocompleter

```
var data = ["foo", "food", "foobar", "fooly", "cake"];
$('#my_text_input').autocompleter({
        'source': data
});
```

- Ajax Autocompleter: The autocompleter will make AJAX calls to the given URL providing a term parameter with the current value of the input field

```
$('#my_text_input').autocompleter({
        'source': 'http://foo.com/webservice.php'
});
```

Using a local autocompleter

```
var data = ["foo", "food", "foobar", "foolish", "foiled", "cake"]
$('#myid').autocompleter({
        'source': data
});
```

- pass the choices as an array of strings
- You can also pass an array of objects with label and value fields

```
var data = [ {'label': 'Track and Field', 'value': 'track'},
             {'label': 'Gymnastics', 'value': 'gymnastics'},
             ...
           ];
```

- the widget injects a ul elements full of choices as you type
- use the appendTo option to specify where the list is inserted

Local autocompleter demo

```
<input id="bands70s" size="40" type="text" />
<div id="bandlistarea"></div>


$('#bands70s').autocomplete({
        'source': data,
        'appendTo': '#bandlistarea'
});
```

Using an AJAX autocompleter

```
$('#my_input').autocomplete({
        'source': 'http://foo.com/webservice.php'
});
```

```
if (!isset($_GET['term'])) {
        header('HTTP/1.1 400 Invalid Request -
        No term parameter provided');
        die('No term parameter provided.');
}
$term = $_GET['term'];
$results = getCompleterResults($term);
// an array() return value print
json_encode($results);
```

- when you have too many choices to hold them all in an array, you can instead fetch subsets of choices from a server using AJAX
- instead of passing choices as an array, pass a URL from which to fetch them
  - the AJAX call is made with a term parameter
  - the choices are sent back from the server as a JSON array of strings or array of objects with label and value fields

accordion widget

- your HTML should be pairs of headers with anchors and containers
- make the parent of these pairs an accordion

```
<div class="accordion">
      <h1><a href="#">Section 1</a></h1>
      <div>Section 1 Content</div> ...
</div>

 $(function() {
       $( "#accordion" ).accordion();
 });
```

tabs widget

- your HTML should be a list of link to element on your page
- the href attributes should match ids of elements on the page

```
<div class="tabs">
        <ul>
                <li><a href="#tab1">Tab 1</a></li>
                <li><a href="#tab2">Tab 2</a></li> ...
        </ul>
<div id="tab1">Tab 1 Content</div>
<div id="tab2">Tab 2 Content</div> ... </div>
```

```
$(function() { $( "#tabs" ).tabs(); });
```

jQuery UI theming

- jQuery UI uses classes gratuitously so that we can style our widgets however we want
- there are two kinds of classes used
  - framework classes which exist for all widgets
  - widget specific classes

kind	classes
Layout Helpers	.ui-helper-hidden, .ui-helper-reset, .ui-helper-clearfix
Widget Containers	.ui-widget, .ui-widget-header, .ui-widget-content
Interaction States	.ui-state-default, .ui-state-hover, .ui-state-focus, .ui-state-active