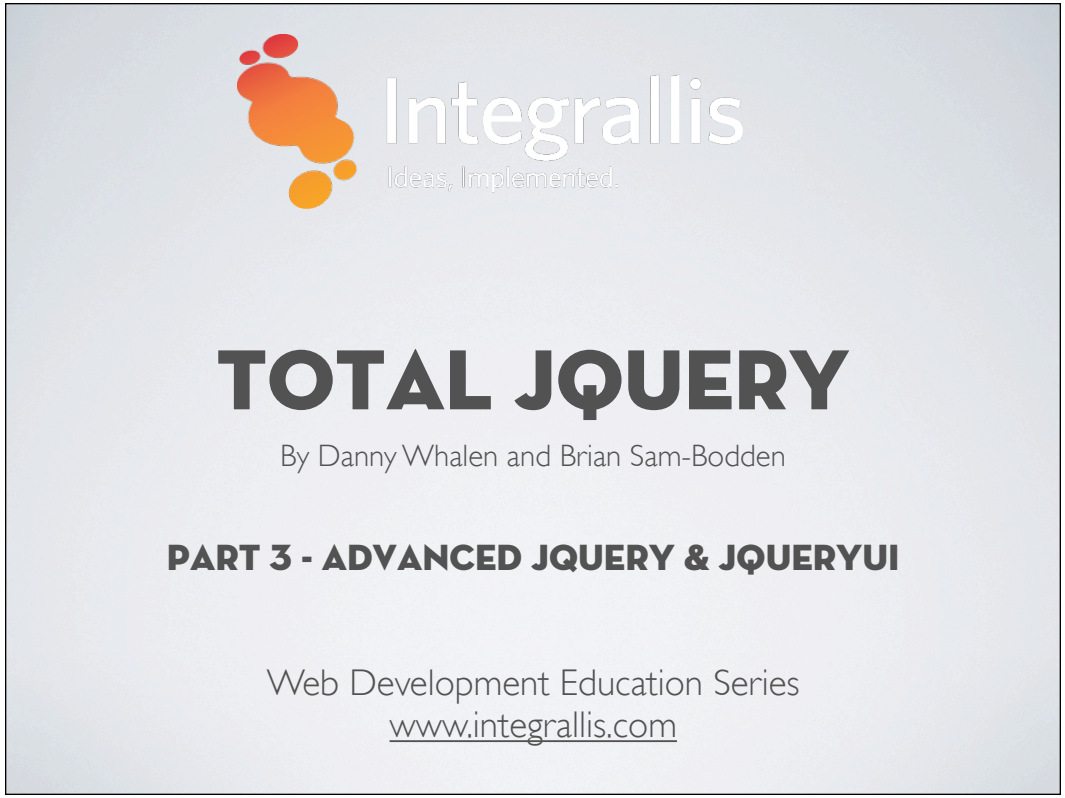


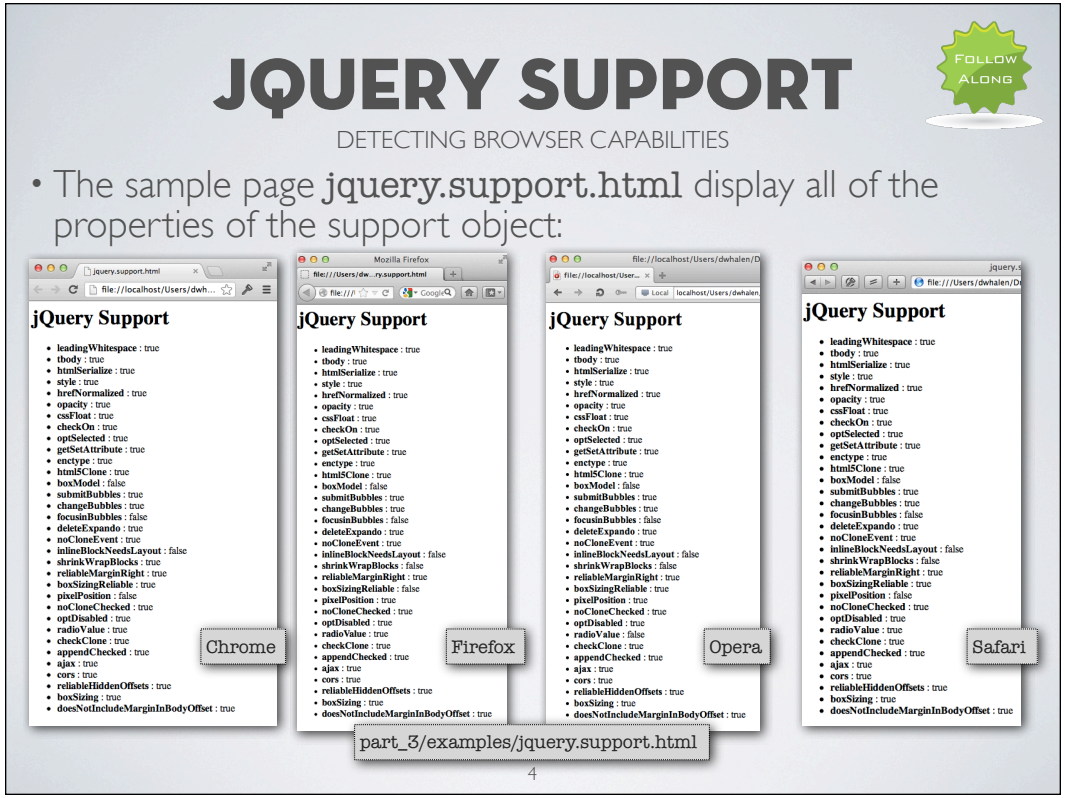
A publication of Integrallis Software, LLC.
www.integrallis.com/en/training
info@integrallis.com

OVERVIEW & OBJECTIVES

- Part 3 will cover:
  - Detecting Browser Capabilities
  - Live jQuery queries
  - jQuery Event Model
  - jQuery and AJAX
  - JQueryUI: Guided Tour
  - Drag and Drop

DETECTING BROWSER CAPABILITIES

JQUERY SUPPORT OBJECT



See the full documentation for jQuery.support at http://api.jquery.com/jQuery.support/

"Rather than using $.browser to detect the current user agent and alter the page presentation based on which browser is running, it is a good practice to use feature detection. To make this process simpler, jQuery performs many such tests and sets properties of the jQuery.support object."

JQUERY LIVE QUERIES

DEALING WITH PAGE CHANGES DYNAMICALLY

JQUERY LIVE QUERIES

DEALING WITH PAGE CHANGES DYNAMICALLY

- Most jQuery selectors works on elements that current exist on the DOM tree
- But what happens when a new element matching an existing selector is dynamically added?
- The `on()` method allows you to process events from descendant elements that are added to the document at a later time. The `live()` method is deprecated.

JQUERY LIVE QUERIES

DEALING WITH PAGE CHANGES DYNAMICALLY

- Let's illustrate the need for `on()` with a simple example
- The example below attempts to dynamically create buttons that can themselves create buttons

part_3/examples/without_on.html

```
<html>
    <head>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.js"></script>
        <script>
        function make_a_button() {
            $('body').append('<input class="maker" type="button" value="Eric says, make me a button!" />');
        }
        $(document).ready(function(){
            make_a_button();
            $('input:button.maker').click(function(){
                make_a_button();
            });
        });
        </script>
    </head>
    <body>
        <h1>Live Buttons</h1>
    </body>
</html>
```

Testing this page reveals that only the first button gets the onclick event handler!

JQUERY LIVE QUERIES

DEALING WITH PAGE CHANGES DYNAMICALLY

- Using `on()` we can make all buttons with the matching class get the `click` handler to be attached, regardless of when they were created:

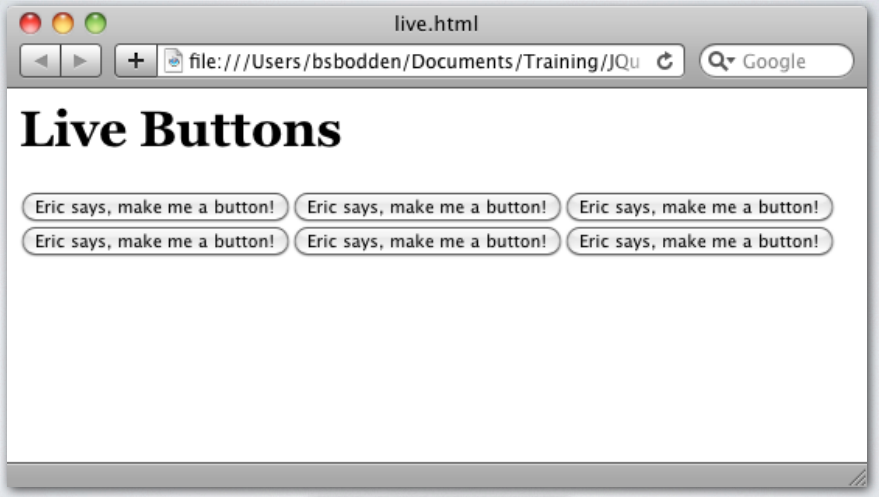
part_3/examples/on.html

```
<html>
    <head>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.js"></script>
        <script>
        function make_a_button() {
            $('body').append('<input class="maker" type="button" value="Eric says, make me a button!" />');
        }
        $(document).ready(function(){
            make_a_button();
            $('body').on('click', 'input:button.maker', function() {
                make_a_button();
            });
        });
        </script>
    </head>
    <body>
        <h1>Live Buttons</h1>
    </body>
</html>
```

JQUERY LIVE QUERIES

DEALING WITH PAGE CHANGES DYNAMICALLY

- Clicking any button creates new buttons:



part_3/examples/on.html

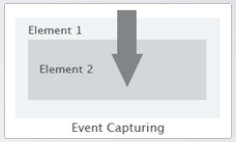
EVENTS, EVENTS, EVENTS

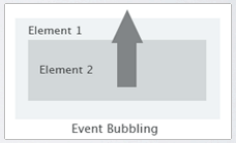
THE JQUERY EVENT MODEL

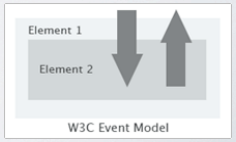
EVENTS

DOM EVENT MODEL

- Event Handling, as many other important areas of the DOM functioning has been hotly debated between the browser implementers:
  - **Event Capturing:** Supported by Netscape states that if an event happens in a child element the event is registered first with the parent element and then the children
  - **Event Bubbling:** Supported by Microsoft states the event if handled first in the child element and then propagated upwards to the parents
  - **W3C Event Model:** Takes a middle of the road approach by breaking the event propagation in two phases: Capture phase from the outermost parent to the event target and then the Bubbling phase in which the events are bubbled up back to the outermost parent



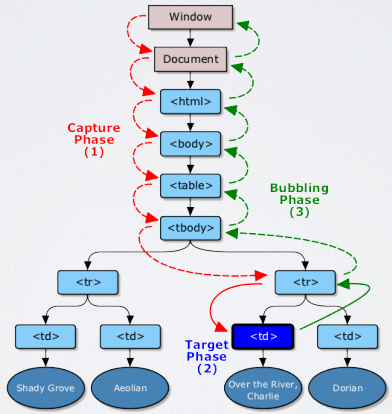




EVENTS

DOM EVENT MODEL

- The following diagram taken from the official W3C page on DOM Level-3 Events (http://www.w3.org/TR/DOM-Level-3-Events/) depicts an event on an element being handled:



EVENTS

DOM EVENT MODEL

- The DOM provides the addEventListener method that provides the following signature:
  `addEventListener(eventType, listener, useCapture)`

- In which `eventType` is a string representing the event type, the `listener` is an object implementing the EventListener interface of just a function and `useCapture` indicates whether this event listener will be triggered in the capture phase (true) or the bubbling phase (false)

- The `useCapture` parameter is not optional in all browser versions

EVENTS

DOM EVENT MODEL

- Let's look at an example: The **event_handling_phases.html** page shows a set of nested **div** elements styled with different colors:

```
<body>
       <h1>Event Handling in the DOM</h1>

       <div id="level_1">
        Level 1
        <div id="level_2">
            Level 2
            <div id="level_3">
                Level 3
                <div id="level_4">
                    Level 4
                </div>
            </div>
        </div>
    </div>


       <div id="output"></div>
</body>
```

```
div#level_1 { width: 300px; border: 5px solid gray; }
div#level_2 { width: 250px; border: 5px solid red; }
div#level_3 { width: 200px; border: 5px solid yellow; }
div#level_4 { width: 150px; border: 5px solid green; }
```

part_3/examples/event_handling_phases.html

EVENTS

DOM EVENT MODEL

- We can write a small snippet of JS to add a couple click event handlers to each element in the page, one using **useCapture** set to **true** and one set to **false**

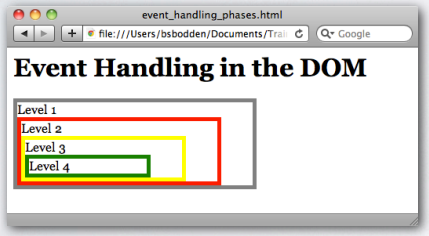
```
function logIt(text) {
    $('#output').append('<p>' + text + '</p>');
}

$(document).ready(function(){
    $('*').each(function() {
      var current = this;
      this.addEventListener('click', function(event) {
          logIt('Capturing ' + current.tagName + ' with id ' + current.id + ' for target ' + event.target.id);
      }, true);
      this.addEventListener('click', function(event) {
          logIt('Bubbling ' + current.tagName + ' with id ' + current.id + ' for target ' + event.target.id);
      }, false);
    });
});
```

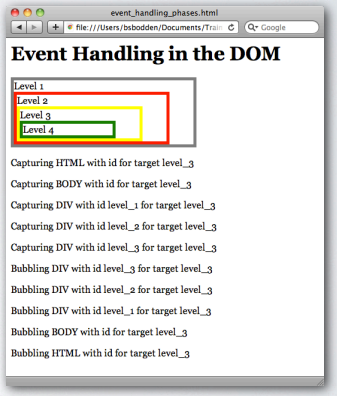
part_3/examples/event_handling_phases.html

EVENTS

DOM EVENT MODEL

- Clicking on any of the `div`s will reveal the chain of events behind the event handling:





part_3/examples/event_handling_phases.html

JQUERY EVENT MODEL

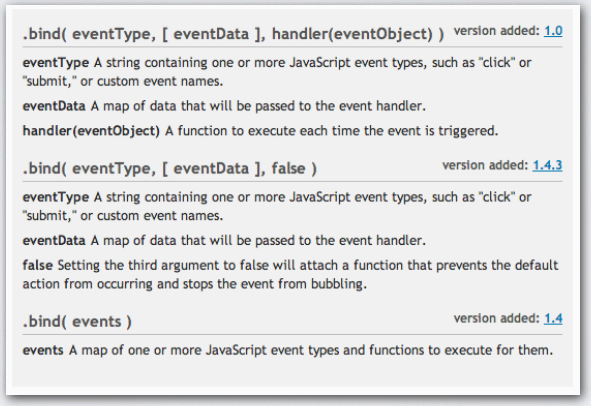
JQUERY TO THE RESCUE

- jQuery provides a thin, unifying wrapper over the DOM event model
- The jQuery event API enables you to set multiple event handlers/listeners per event type per element
- It provides an enhanced Event object to the handlers/listeners
- Provides a clean API for event canceling and default action blocking

JQUERY EVENT MODEL

BINDING AND UNBINDING

- Event handlers or listeners can be created for a DOM element using the **bind** method (http://api.jquery.com/bind/):
  `$('#clickable').bind('click', function(event) { ... });`



.bind(eventType, [eventData], handler(eventObject)) version added: 1.0

eventType A string containing one or more JavaScript event types, such as "click" or "submit," or custom event names.

eventData A map of data that will be passed to the event handler.

handler(eventObject) A function to execute each time the event is triggered.

.bind(eventType, [eventData], false) version added: 1.4.3

eventType A string containing one or more JavaScript event types, such as "click" or "submit," or custom event names.

eventData A map of data that will be passed to the event handler.

false Setting the third argument to false will attach a function that prevents the default action from occurring and stops the event from bubbling.

.bind(events) version added: 1.4

events A map of one or more JavaScript event types and functions to execute for them.

JQUERY EVENT MODEL

MANY DIFFERENT WAYS TO BIND

- Let's illustrate the many different ways to bind an event handler or listener to a DOM element using the following HTML page:

```
<body>
    <h1>Live Buttons</h1>
    <input id="button_1" type="button" value="Button One" />
    <input id="button_2" type="button" value="Button Two" />
    <input id="button_3" type="button" value="Button Three" />
    <input id="button_4" type="button" value="Button Four" />
    <input id="button_5" type="button" value="Button Five" />
    <input id="button_6" type="button" value="Button Six" />
    <br/><br/>
    <div id="enter">
        Enter
    </div>
    <div id="leave">
        Leave
    </div>
</body>
```

part_3/examples/jquery.bind.html

JQUERY EVENT MODEL

MANY DIFFERENT WAYS TO BIND

- We can bind using the bind method and passing the event name, in the case below 'click' and passing an anonymous function:

```
// bind click using the bind method
$('#button_1').bind('click', function() {
    alert('You just clicked #1');
});
```

part_3/examples/jquery.bind.html

JQUERY EVENT MODEL

MANY DIFFERENT WAYS TO BIND

- Alternatively for some events we can use the shortcut binding methods:

```
// bind click using the shortcut method
$('#button_2').click(function() {
    alert('You just clicked #2');
});
```

part_3/examples/jquery.bind.html

- jQuery provides many shortcut binding methods, including: blur, focus, focusin, focusout, load, resize, scroll, unload, click, dblclick, mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter, mouseleave, change, select, submit, keydown, keypress, keyup, error

JQUERY EVENT MODEL

MANY DIFFERENT WAYS TO BIND

- If you are binding multiple events to the same element you can pass an object literal:

```
// multiple events - different handlers
$('#button_3').bind({
    mouseenter: function() {
        $('#enter').css('opacity', '0.2');
        $('#leave').css('opacity', '1.0');
    },
    mouseleave: function() {
        $('#enter').css('opacity', '1.0');
        $('#leave').css('opacity', '0.2');
    }
});
```

part_3/examples/jquery.bind.html

JQUERY EVENT MODEL

MANY DIFFERENT WAYS TO BIND

- If you want to bind multiple events using the same handler you can pass a space separated list of event type names:

```
// multiple events - same handler
$('#button_4').bind('mouseenter mouseleave', function() {
    $(this).toggleClass('entered');
});
```

part_3/examples/jquery.bind.html

JQUERY EVENT MODEL

MANY DIFFERENT WAYS TO BIND

- And of course you can reuse a handler to be attached to many elements

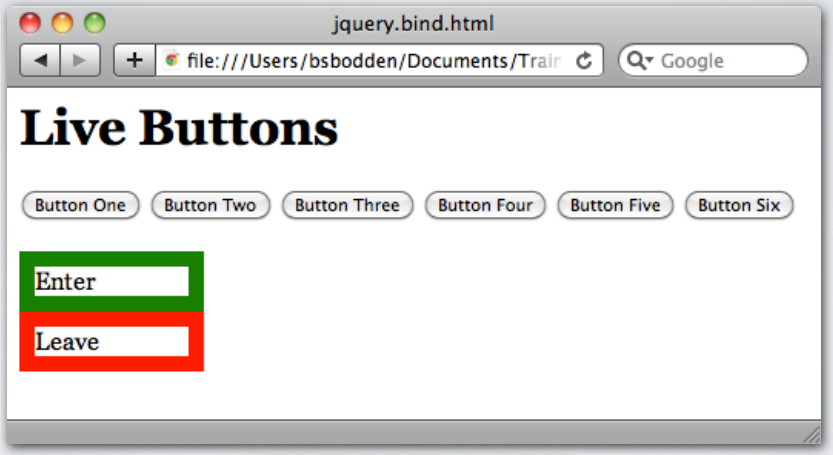
```
// reusable handler
function myHandler(event) { alert("You clicked '" + $(this).val() + "'"); }
$('#button_5').click( myHandler );
$('#button_6').click( myHandler );
```

part_3/examples/jquery.bind.html

JQUERY EVENT MODEL

MANY DIFFERENT WAYS TO BIND

- The `jquery.bind.html` example shows the different ways to bind an event handler:

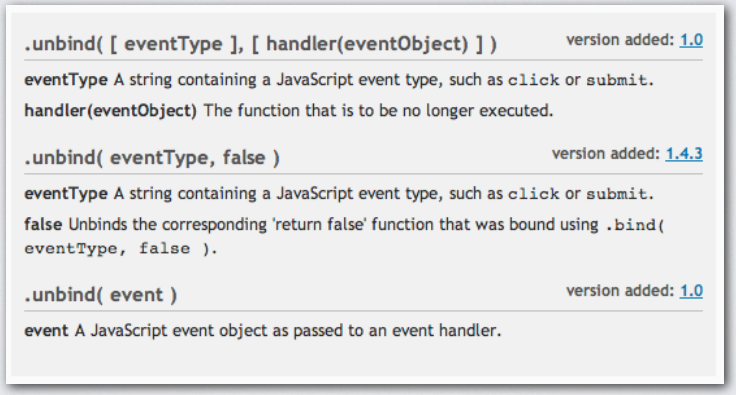


part_3/examples/jquery.bind.html

JQUERY EVENT MODEL

BINDING AND UNBINDING

- Similarly jQuery provides an unbind method to remove an event handler/listener
- See http://api.jquery.com/unbind/



.unbind([eventType], [handler(eventObject)]) version added: 1.0

eventType A string containing a JavaScript event type, such as `click` or `submit`.

handler(eventObject) The function that is to be no longer executed.

.unbind(eventType, false) version added: 1.4.3

eventType A string containing a JavaScript event type, such as `click` or `submit`.

false Unbinds the corresponding 'return false' function that was bound using `.bind( eventType, false )`.

.unbind(event) version added: 1.0

event A JavaScript event object as passed to an event handler.

JQUERY EVENT MODEL

CUSTOM EVENTS

- jQuery supports the binding of custom events to an element
- A custom event can the be triggered using the trigger method
- Additionally you can pass extra data in the trigger method

```
<h1>Custom Events</h1>
<input id="source" type="button" value="Button One" />
<br/><br/>
<div id="target"></div>
```

```
$(document).ready(function(){
    $('#target').bind('custom', function(event, param1, param2) {
       $(this).text(param1 + " " + param2)
    });

    $('#source').bind('click', function() {
        $('#target').trigger('custom', ['Hello', 'World']);
    });
});
```

part_3/examples/custom_events.html

JQUERY EVENT MODEL

CUSTOM EVENTS

- Clicking the button fires the event with the `#target div` as the source which is it's handled by the listener and uses the data passed to add a text node to the `div`

part_3/examples/custom_events.html

JQUERY AND AJAX

ASYNCHRONOUS COMMUNICATIONS WITH THE SERVER

JQUERY AND AJAX

ASYNCHRONOUS COMMUNICATIONS WITH THE SERVER

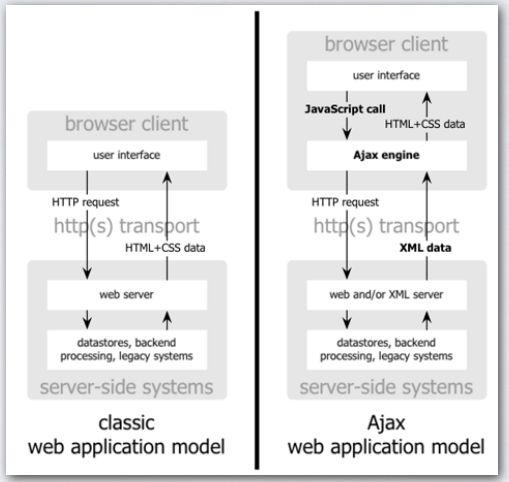
- AJAX stands for **A**synchronous **J**avaScript **A**nd **X**ML
- AJAX refers to any JavaScript technique for asynchronous interactions with the server using XML
  - The foundations for AJAX were championed by Microsoft back in 1998 (using ActiveX as part of OWA*)
- Eventually, the non-MS browsers implemented a standard way encapsulated in the XHR (XMLHttpRequest) object
- A more loose definition of AJAX is any operation that updates a page without a full page refresh using data obtained from a remote call

* Outlook Web Access (according to jQuery in Action by Manning)

JQUERY AND AJAX

ASYNCHRONOUS COMMUNICATIONS WITH THE SERVER

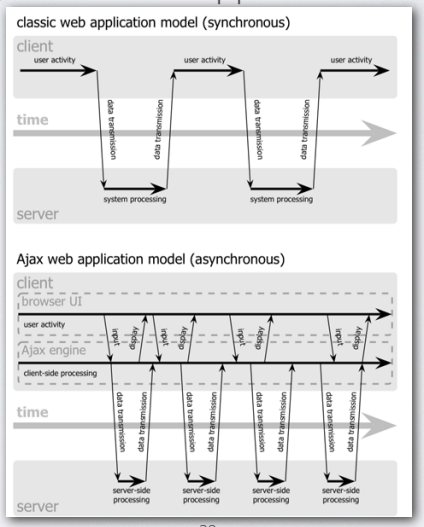
- AJAX (formal) application model, server returns XML, page is not refreshed and changes are applied dynamically with JS



JQUERY AND AJAX

ASYNCHRONOUS COMMUNICATIONS WITH THE SERVER

- In an AJAX application we typically see more and smaller requests, resulting in "chattier" applications



Images from JavaLobby article at http://www.javalobby.org/articles/ajax/

JQUERY AND AJAX

ASYNCHRONOUS COMMUNICATIONS WITH THE SERVER

- jQuery AJAX capabilities hide most of the complexity involved in performing an AJAX request
- One of the common uses of AJAX is to dynamically load a server-side generated snippet of HTML into an element of the page
- jQuery provides the `load()` method specifically for this purpose
- The `load()` method takes the URL of the resource to be loaded

http://api.jquery.com/load/

JQUERY AND AJAX

ASYNCHRONOUS COMMUNICATIONS WITH THE SERVER

- The `load()` method can load the contents of a server-side document into a given DOM element:

```
$('#my_div').load('some_page.html');
```

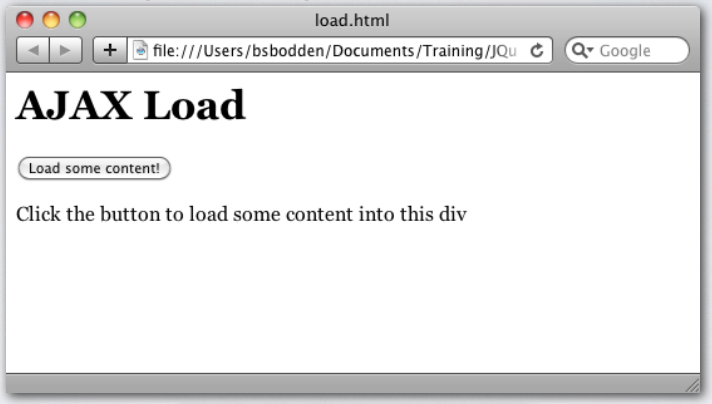
- The `load()` method can also take a callback function that is invoked when the loading has completed

```
$('#my_div').load('some_page.html', function() {
    $('p#message').text('loaded!');
});
```

JQUERY AND AJAX

ASYNCHRONOUS COMMUNICATIONS WITH THE SERVER

- We'll start with the behavior-less, markup-only `load.html` page and use jQuery to dynamically load content in the `load_area div` while showing a "loading..." animation



part_3/examples/ajax_load/load.html

JQUERY AND AJAX

ASYNCHRONOUS COMMUNICATIONS WITH THE SERVER

- The **load.html** page has two div elements; one with **id** of "**load_area**" containing a paragraph of text and one with and **id** of "**loader**" in which we'll show the image **loading.gif**

```
<html>
    <head>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.js"></script>
        <script>
        $(document).ready(function(){
        });
        </script>
    </head>
    <body>
        <h1>AJAX Load</h1>
        <input class="loader" type="button" value="Load some content!" />
        <div id="load_area">
            <p>Click the button to load some content into this div</p>
        </div>
        <div id="loader" class="loading"></div>
    </body>
</html>
```

part_3/examples/ajax_load/load.html

JQUERY AND AJAX

ASYNCHRONOUS COMMUNICATIONS WITH THE SERVER

- Let's start by styling the div meant to show the loading image:

```
<style>
div#loader {
    width: 40px;
    height: 40px;
    overflow: hidden;
    display: none;
}

div#loader.loading {
    background: url(loading.gif) no-repeat center center;
}
</style>
```

part_3/examples/ajax_load/load_1.html

JQUERY AND AJAX

ASYNCHRONOUS COMMUNICATIONS WITH THE SERVER

- The behavior can be achieved with 6 lines of jQuery
- On click we reveal the loader, fade out the load_area, asynchronously load the new contents, fade our the loader and finally reveal the new contents to the user

```
$('input:button.loader').click(function() {
    $('#loader').show();
    $('#load_area').fadeOut(2000, function() {
        $(this).load('content.html', function() {
            $('#loader').fadeOut(2000);
        });
    })
    .fadeIn(2000);
});
```

part_3/examples/ajax_load/load_1.html

JQUERY AND AJAX

ASYNCHRONOUS COMMUNICATIONS WITH THE SERVER

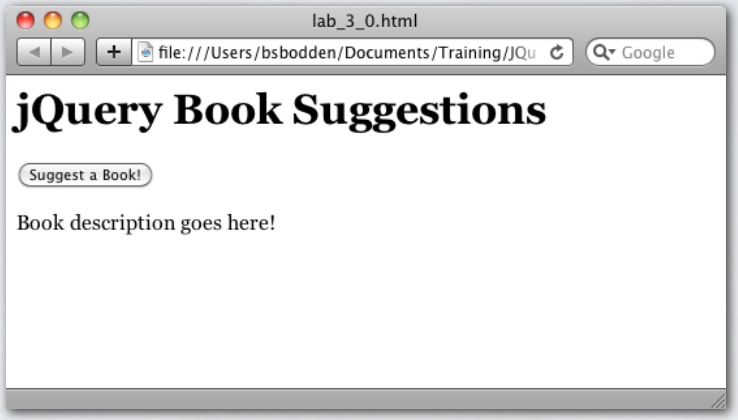
- Let's confirm the functionality of the load page:

part_3/examples/ajax_load/load_1.html

LAB 3.0

AJAX WITH JQUERY

- In Lab 3.0 you will work with jQuery AJAX function load to dynamically load some XML content:



part_3/examples/lab_3_0/lab_3_0.html

LAB 3.0

AJAX WITH JQUERY

- In Lab 3.0 you are asked to enhance the `lab_3_0.html` page to dynamically load content from an XML file
  - When the user clicks the **"Suggest a Book!"** button, load the contents of the remote XML file in a **hidden** div (**temp_loading_area**)
  - Using JavaScript generate a **random number** based on the number of book elements in the XML data
  - Load the chosen book description and image into the provided divs using some jQuery transitions

JQUERY AND AJAX

ASYNCHRONOUS COMMUNICATIONS WITH THE SERVER

- Other jQuery AJAX methods:
  - **ajax:** The most basic and configurable way to perform an AJAX request (http://api.jquery.com/jQuery.ajax/)
  - **getJSON:** Loads JSON data using a GET request (http://api.jquery.com/jQuery.getJSON/)
  - **get:** Loads data using a GET request (http://api.jquery.com/jQuery.get/)
  - **post:** Loads data using a POST request (http://api.jquery.com/jQuery.post/)
  - **getScript**: Loads a JavaScript file from the server and executes it (http://api.jquery.com/jQuery.getScript/)

LAB 3.0.1

AJAX AND WEB SERVICES

- Ajax, Web Services and Maps
  - Use http://jquery-ui-map.googlecode.com to create a page with an interactive map that can be use to select a geo location
  - Use JQuery getJSON to hit the geonames WS at:
    - api.geonames.org/neighbourhoodJSON
  - Show the information returned from the service in a div below the map and use an animation to alert the user that the results have arrived

JQUERY UI

WIDGETS AND UI UTILITIES

JQUERY UI

WIDGETS AND UI UTILITIES

- The jQueryUI project contains the "official" widgets and utilities for jQuery (in addition to the hundreds of Open Source ones)
- jQuery UI provides a collection of widgets and components
- Easy to prototype with and equally easy to use in production
- Customizable theming support provides for a consistent look and feel

JQUERY UI

WIDGETS AND UI UTILITIES

- Lean, clean, minimalistic UI API
- Provides a framework for creating UI Widgets/Components
- Adding jQuery UI:
  http://ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js
- Adding a Theme:
  http://ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/themes/cupertino/jquery-ui.css

JQUERY UI

WIDGETS AND UI UTILITIES

- Available jQuery UI Widgets:

Accordion	Autocomplete	Button
Datepicker	Dialog	Progressbar
Slider	Tabs	...more

JQUERY UI

WIDGETS AND UI UTILITIES

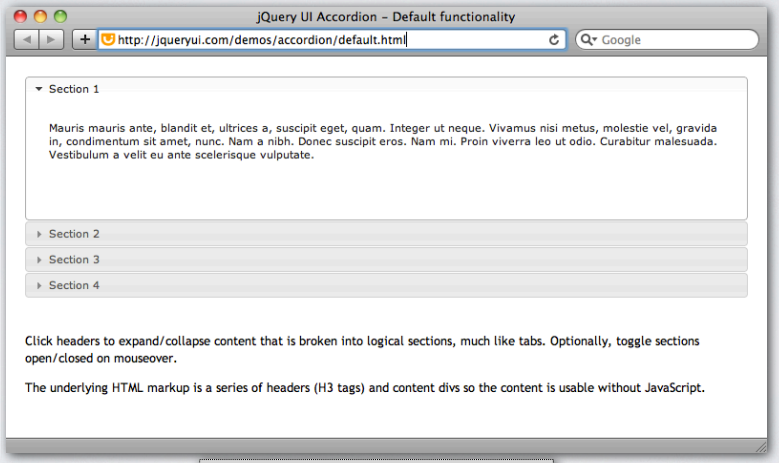
- To use jQuery UI widgets we follow the familiar jQuery pattern:
  - Find some applicable elements and apply and configure the widget on it:

```
<h1>Find Approved Housing Quotes for a Given Date</h1>
<script type="text/javascript">
$(function() {
    $("#datepicker").datepicker({
       dateFormat: 'yy/mm/dd',
       onSelect: function(dateText, inst) {
         window.location.href = "/quotes/housing/approved/" + dateText;
       }
    });
});
</script>
Date: <div id="datepicker"></div>
```

ACCORDION

JQUERY UI WIDGET TOUR

- The jQueryUI accordion is an outlook bar/panel style widget that allow content to be hidden in a sliding panel:

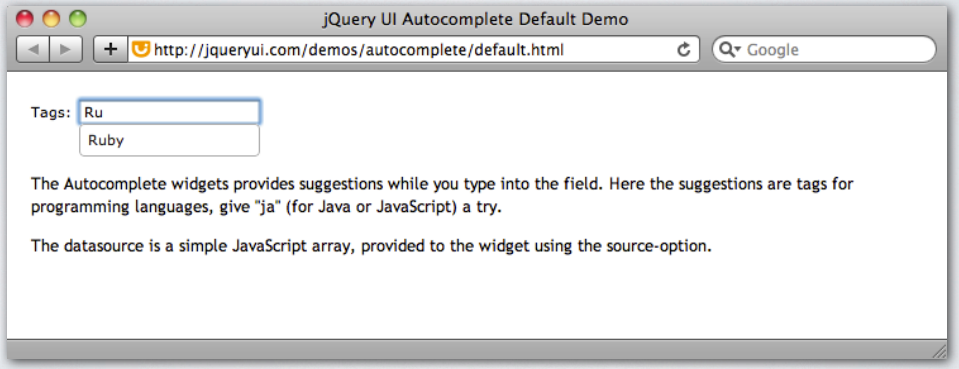


http://jqueryui.com/demos/accordion/

AUTOCOMPLETE

JQUERY UI WIDGET TOUR

- The jQueryUI autocomplete provides autocomplete functionality for a text field with data coming from the server via AJAX or from a local JSON/JS Array:

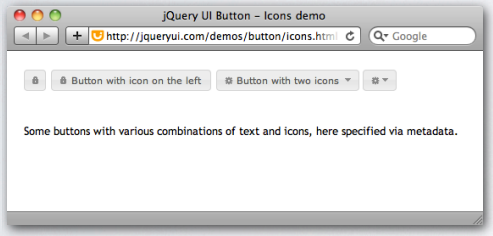


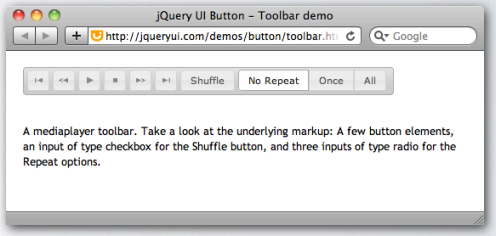
http://jqueryui.com/demos/autocomplete/

BUTTONS

JQUERY UI WIDGET TOUR

- The jQueryUI buttons provides clean, css themed buttons:



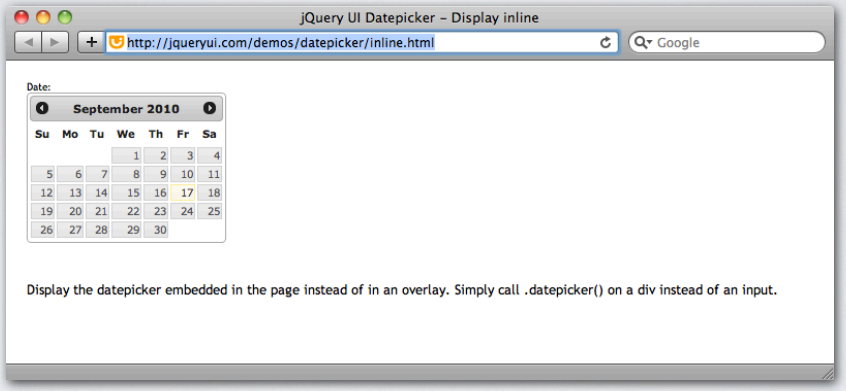


http://jqueryui.com/demos/button/

DATEPICKER

JQUERY UI WIDGET TOUR

- The jQueryUI datepicker provides a embedded calendar date picker or a drop-down calendar date picker:

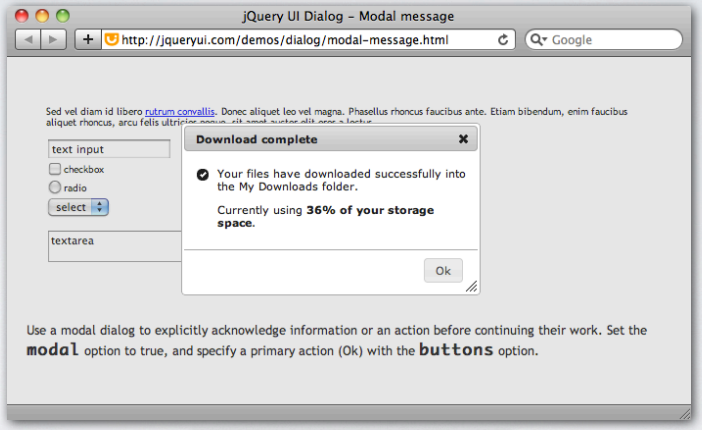


http://jqueryui.com/demos/datepicker/

DIALOG

JQUERY UI WIDGET TOUR

- The jQueryUI dialog provides a variety of modal and modeless dialogs for your applications:

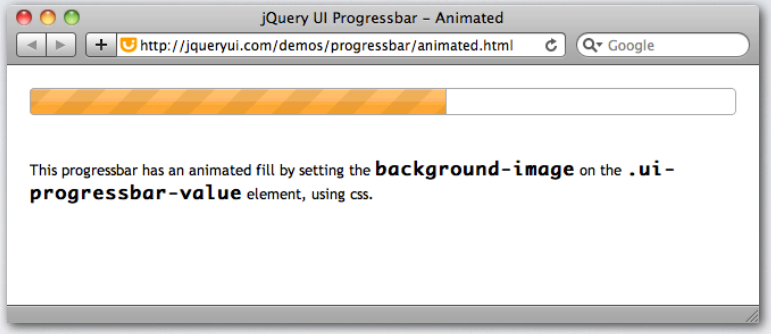


http://jqueryui.com/demos/dialog/

PROGRESSBAR

JQUERY UI WIDGET TOUR

- The jQueryUI progressbar show % of completion for a process, it can be used statically or dynamically (updated via AJAX)

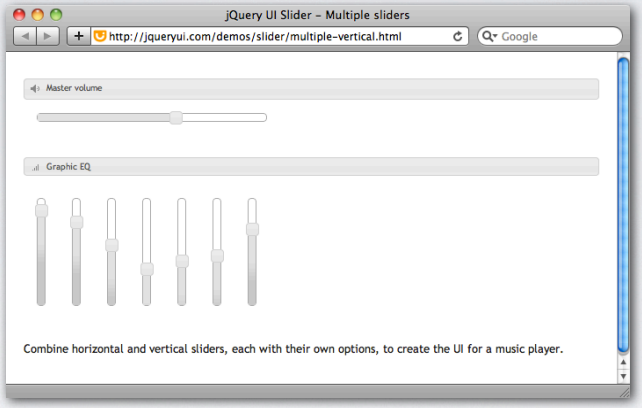


http://jqueryui.com/demos/progressbar/

SLIDER

JQUERY UI WIDGET TOUR

- The jQueryUI slider widget can turn an element into a highly configurable slider:

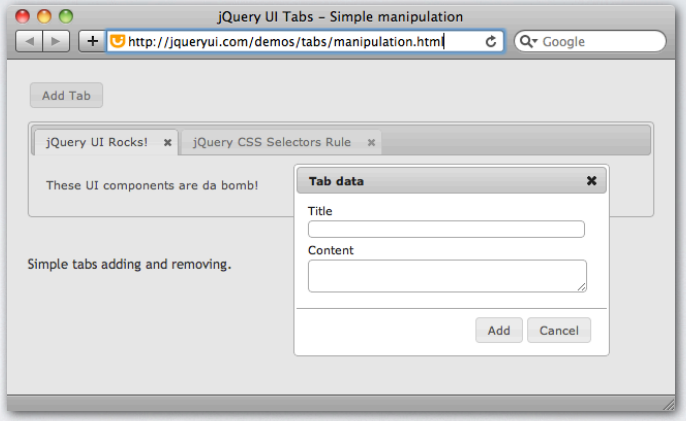


http://jqueryui.com/demos/slider/

TABS

JQUERY UI WIDGET TOUR

- The jQueryUI tabs can turn div, ul or li into tabs that can hold content and can even load content dynamically using AJAX:



http://jqueryui.com/demos/tabs/

EFFECTS

JQUERY UI EFFECTS

- The jQueryUI project contains the "official" widgets and utilities for jQuery (in addition to the hundreds of Open Source ones)
- jQuery UI provides a collection of widgets and components
- Easy to prototype with and equally easy to use in production
- Customizable theming support provides for a consistent look and feel

LAB 3.1

JQUERY UI

- Create simple page using a few of the jQuery UI components
  - Add a tab component
  - Add a slider component
  - As the user increments the slider, tabs are added and vice-versa; up to a maximum of 5 tabs and a minimum of 1 tab

DRAG AND DROP

JQUERY UI DRAG AND DROP SUPPORT

DRAG AND DROP

JQUERY UI DRAG AND DROP SUPPORT

- jQuery UI provides several ways to achieve drag and drop support in your applications by providing the following "interactions":
  - Draggable: Makes an element draggable using the mouse
  - Droppable: Makes an element droppable onto a target
  - Resizable: Makes an element resizable
  - Selectable: Makes a list of table element selectable
  - Sortable: Makes a list or table sortable

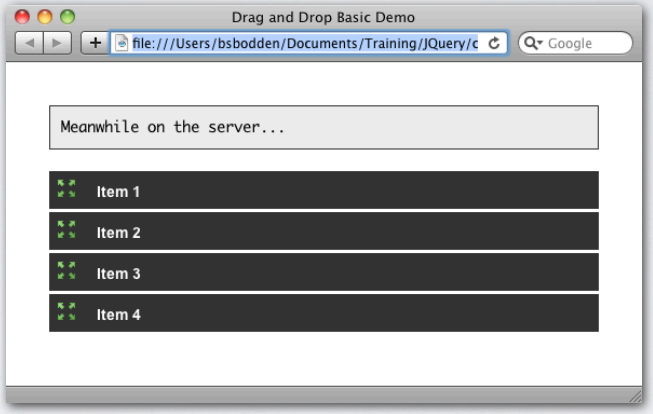
- http://jqueryui.com/demos/draggable/
- http://jqueryui.com/demos/droppable/
- http://jqueryui.com/demos/resizable/
- http://jqueryui.com/demos/selectable/
- http://jqueryui.com/demos/selectable/

-

DRAG AND DROP

JQUERY UI DRAG AND DROP SUPPORT

- The page below will serve as our framework for basic drag and drop interaction



part_3/examples/drag_and_drop/drag_and_drop.html

DRAG AND DROP

JQUERY UI DRAG AND DROP SUPPORT

- Let's enhance the DND example page below to allow for items in a ordered list to be rearranged using drag and drop gestures by making use of the sortable interaction

```
<html>
<head>
    <title>Drag and Drop Basic Demo</title>
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.js"></script>
    <script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js"></script>
    <link rel='stylesheet' href='styles.css' type='text/css' media='all' />
    <script type="text/javascript">
    $(document).ready(function() {

    });
    </script>
</head>
<body>
<pre><div id="info">Meanwhile on the server...</div></pre>
<ul id="test-list">
    <li id="item_1"><img src="arrow.png" alt="move" class="handle" /><strong>Item 1 </strong></li>
    <li id="item_2"><img src="arrow.png" alt="move" class="handle" /><strong>Item 2</strong></li>
    <li id="item_3"><img src="arrow.png" alt="move" class="handle" /><strong>Item 3</strong></li>
    <li id="item_4"><img src="arrow.png" alt="move" class="handle" /><strong>Item 4</strong></li>
</ul>
</body>
</html>
```

part_3/examples/drag_and_drop/drag_and_drop.html

http://jqueryui.com/demos/sortable/

DRAG AND DROP

JQUERY UI DRAG AND DROP SUPPORT

```css
* { margin: 0; padding: 0; }

body {
    font: 0.9em Arial;
    padding: 40px;
}

#info {
    display: block;
    padding: 10px; margin-bottom: 20px;
    border: 1px solid #333;
    background-color: #efefef;
}

#test-list {
    list-style: none;
}

#test-list li {
    margin: 0 0 3px;
    padding:8px;
    background-color:#333;
    color:#fff;
    list-style: none;
}

#test-list li img.handle {
    width: 16;
    height: 16;
    margin-right: 20px;
    cursor: move;
}
```

- A simple stylesheet is used to style the list

part_3/examples/drag_and_drop/styles.css

DRAG AND DROP

JQUERY UI DRAG AND DROP SUPPORT

- We make the list sortable, make the images (with class 'handle') the drag handle and provide a function handler to display the new list order in the info div

```
$("#test-list").sortable({
    handle : '.handle',
    update : function () {
        // simulate server communication
        setTimeout(function() {
            var order = [];
            $('#test-list li').each(function() {
                order.push($(this).attr("id"));
            });
            $("#info").text(order.join(" >> "));
            $("#info").effect( "bounce", {}, 1000);
        }, 1000);
    }
});
```

part_3/examples/drag_and_drop/drag_and_drop.html

LAB 3.2

DRAG AND DROP SHOPPING CART

- In this lab we will create a jQuery powered shopping cart from the ground up
  - Create the markup with two lists styled to appear side-by-side and an element showing the total amount of the items in the shopping cart
  - Each retail item will be represented by a div containing an image (drag handle), encode the item price in the div
  - As items are dragged into the cart, use an effect to alert the user of the cart total amount changes
  - Use the module of prototype pattern to separate the UI interactions from the business interactions
  - Extra Credit: As items are dropped into the cart, add controls to the items to increase the quantity