Developing JavaScript Based Rich Web UI with jQuery

Course # 22865

- Subhadeep Chatterjee

Introduction

Fundamentals

Fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

Advantages

jQuery allows elegantly (and efficiently) find and manipulate HTML elements with minimum lines of code. It uses a querying approach based on CSS concept.

Installation

All of the code is available in two formats:

- Compressed (which allows you to have a significantly smaller file size) http://code.google.com/p/jqueryjs/downloads/detail?name=jquery-1.3.2.min.js and
- Uncompressed (good for debugging and to understand what is behind the magic) http://code.google.com/p/jqueryjs/downloads/detail?name=jquery-1.3.2.js

Introduction

The Basics

Find elements (through css and/or xpath type selectors) and do something with them (through jQuery methods).

Find elements -

- by element type
- by element id
- by element class name
- by element hierarchy
- by a combination of these

Execute multiple methods on the selector by chaining their sequence together.

Execute methods -

- by visual appearance(s)
- by event execution
- by a combination of these

Introduction

Example - Do something with a button element

- Find button elements
- Attach click event handler

```
<html>
    <head>
        <script src="javascript/jquery-1.3.2.min.js"></script>
    </head>
    <body>
        <div class="buttonContainer">
            <button id="ok_button" class="buttonClass">OK</button>
        </div>
        <button id="cancel_button" class="buttonClass">Cancel</button>
    </body>
    <script>
        var buttonEls = document.getElementsByTagName('button');
            for(var i=0; i<buttonEls.length; i++){
                var buttonEl = buttonEls.item(i);
                buttonEl.onclick = function(){
                    alert("I clicked the button!");
                };
            }
    </script>

</html>
```

Introduction

Example - Do something with a button element

- Find button elements
- Attach click event handler

```
<html>
    <head>
        <script src="javascript/jquery-1.3.2.min.js"></script>
    </head>
    <body>
        <div class="buttonContainer">
            <button id="ok_button" class="buttonClass">OK</button>
        </div>
        <button id="cancel_button" class="buttonClass">Cancel</button>
    </body>

    <script>
        jQuery("button").click( function(){
            alert("I clicked the button!");
        });
    </script>


</html>
```

Introduction

jQuery provides a new class, appropriately named jQuery, that serves as a wrapper around other objects in order to provide extended operations upon those objects (often known as adapter pattern).

```
<script>
    jQuery("button").click( function(){
        alert("I clicked the button!");
    });
</script>
```

In order to make expressions and statements containing jQuery wrappers more compact, the jQuery class is mapped to $ as an alias.

```
<script>
    $("button").click( function(){
        alert("I clicked the button!");
    });
</script>
```

Introduction

- Lets add some styling(css)

```
<html>
    <head>
        <style>
            .taskBar { padding: 5px; border: 1px solid #ccc; float: left; width: auto; }
            .primaryButtonContainer { float: left; margin-right: 5px; }
            .buttonClass { background-color: #f0f0f0; border: 1px solid #cccccc; }
            #ok_button { font-weight: bold; }
            #cancel_button { color: #333333; }
        </style>
        <script src="javascript/jquery-1.3.2.min.js"></script>
    </head>
    <body>
        <div class="taskBar">
            <div class="primaryButtonContainer">
                <button id="ok_button" class="buttonClass">OK</button>
            </div>
            <button id="cancel_button" class="buttonClass">Cancel</button>
        </div>
    </body>

    <script>
        $("button").click( function(){
            alert("I clicked the button!");
        });
    </script>

</html>
```

Introduction

- Find elements using css concept to jQuery selector

```
<html>
    <head>
        <style>
            .taskBar { padding: 5px; border: 1px solid #ccc; float: left; width: auto; }
            .primaryButtonContainer { float: left; margin-right: 5px; }
            .buttonClass { background-color: #f0f0f0; border: 1px solid #cccccc; }
            #ok_button { font-weight: bold; }
            #cancel_button { color: #333333; }
        </style>
        <script src="javascript/jquery-1.3.2.min.js"></script>
    </head>
    <body>
        <div class="taskBar">
            <div class="primaryButtonContainer">
                <button id="ok_button" class="buttonClass">OK</button>
            </div>
            <button id="cancel_button" class="buttonClass">Cancel</button>
        </div>
    </body>

    <script>
        $("#ok_button).click( function(){
            alert("I clicked the button!");
        });
    </script>

</html>
```

Introduction

- Execute multiple methods to a jQuery selector

```
<html>
    <head>
        <style>
            .taskBar { padding: 5px; border: 1px solid #ccc; float: left; width: auto; }
            .primaryButtonContainer { float: left; margin-right: 5px; }
            .buttonClass { background-color: #f0f0f0; border: 1px solid #cccccc; }
            #ok_button { font-weight: bold; }
            #cancel_button { color: #333333; }
        </style>
        <script src="javascript/jquery-1.3.2.min.js"></script>
    </head>
    <body>
        <div class="taskBar">
            <div class="primaryButtonContainer">
                <button id="ok_button" class="buttonClass">OK</button>
            </div>
            <button id="cancel_button" class="buttonClass">Cancel</button>
        </div>
    </body>
   <script>
        $("#ok_button").click( function(){
            alert("I clicked the button!");
        });
        $("#ok_button").css("border-color", "blue");
   </script>
</html>
```

Introduction

- Execute multiple methods to a jQuery selector

```
<html>
    <head>
        <style>
            .taskBar { padding: 5px; border: 1px solid #ccc; float: left; width: auto; }
            .primaryButtonContainer { float: left; margin-right: 5px; }
            .buttonClass { background-color: #f0f0f0; border: 1px solid #cccccc; }
            #ok_button { font-weight: bold; }
            #cancel_button { color: #333333; }
        </style>
        <script src="javascript/jquery-1.3.2.min.js"></script>
    </head>
    <body>
        <div class="taskBar">
            <div class="primaryButtonContainer">
                <button id="ok_button" class="buttonClass" title="OK" >OK</button>
            </div>
            <button id="cancel_button" class="buttonClass">Cancel</button>
        </div>
    </body>

    <script>
        $("#ok_button").click( function(){
            alert("I clicked the button!");
        }).css("border-color", "blue");
    </script>

</html>
```

jQuery selectors

jQuery selectors

```
...
<style>
    .taskBar { padding: 5px; border: 1px solid #ccc; float: left; width: auto; }
    .primaryButtonContainer { float: left; margin-right: 5px; }
    .buttonClass { background-color: #f0f0f0; border: 1px solid #cccccc; }
    #ok_button { font-weight: bold; }
    #cancel_button { color: #333333; }
</style>
...
```

- Select element by id

  $("#ok_button")

- Select element by class

  $(".buttonClass")

- Select element tag name

  $("button")

- Select element by descendant

  $("div button")

- Select element by direct children

  $("div > button")

jQuery selectors

Way of identifying page elements quickly and easily by leveraging css syntax paradigm.

- Basic
- Hierarchy
- Attribute filters
- Basic filters
- Child filters
- Content filters
- Form filters
- Visibility filters

jQuery selectors

- Basic

(“.class”)
Selects all elements with the given class.

(“element”)
Selects all elements with the given tag name.

(“#id”)
Selects a single element with the given id attribute.

(“selector1, selector2, selectorN”)
Selects the combined results of all the specified selectors.

(“*”)
Selects all elements.

jQuery selectors

- Hierarchy

(“parent > child”)
Selects all direct child elements specified by "child" of "parent" elements.

(“ancestor descendant”)
Selects all elements that are descendants of a given ancestor.

(“prev + next”)
Selects all next elements matching "next" that are immediately preceded by a sibling "prev".

(“prev ~ siblings”)
Selects all sibling elements that follow after the "prev" element, have the same parent,
and match the filtering "siblings" selector.

jQuery selectors

- Attribute filters

Identifying page elements by their attributes.

[name=value]
Selects elements that have the specified attribute with a value exactly equal to a certain value.

[name!=value]
Select elements that either don't have the specified attribute, or do have the specified attribute but not with a certain value.

[name*=value]
Selects elements that have the specified attribute with a value containing the a given substring.

[name=value][name2=value2]
Matches elements that match all of the specified attribute filters.

```
<script>
    $("#ok_button").click( function(){
        alert("I clicked the button!");
    }).css("border-color", "blue");
</script>
```

```
<script>
    $("div[title='OK']").click( function(){
        alert("I clicked the button!");
    }).css("border-color", "blue");
</script>
```

jQuery selectors

- Basic filters

:first Selector
Selects the first matched element.

:last Selector
Selects the last matched element.

:even Selector
Selects even elements, zero-indexed.

:odd Selector
Selects odd elements, zero-indexed.

jQuery selectors

- Child filters

:first-child
Selects all elements that are the first child of their parent.

:last-child
Selects all elements that are the last child of their parent.

:nth-child
Selects all elements that are the nth-child of their parent.

:only-child
Selects all elements that are the only child of their parent.

jQuery selectors

- Content filters

:contains()
Select all elements that contain the specified text.

:empty
Select all elements that have no children (including text nodes).

:has()
Selects elements which contain at least one element that matches the specified selector.

:parent
Select all elements that are the parent of another element, including text nodes.

jQuery selectors

- Form filters

:checked
Matches all elements that are checked.

:disabled
Selects all elements that are disabled.

:enabled
Selects all elements that are enabled.

:radio
Selects all elements of type radio.

jQuery selectors

- Visibility filters

:hidden
Selects all elements that are hidden.

:visible
Selects all elements that are visible.

Traversing HTML DOM with jQuery selectors

Traversing HTML DOM

HTML DOM traversal can happen in two basic ways using jQuery,

- Filtering

.filter(selector)
Reduce the set of matched elements to those that match the selector or pass the function's test.

.first()
Reduce the set of matched elements to the first in the set.

.last()
Reduce the set of matched elements to the final one in the set.

etc...

Traversing HTML DOM

- Tree traversing

.children([selector])
Get the children of each element in the set of matched elements, optionally filtered by a selector.

.closest(selector)
Get the first ancestor element that matches the selector, beginning at the current element and progressing up through the DOM tree.

.find(selector)
Get the descendants of each element in the current set of matched elements, filtered by a selector.

.next([selector])
Get the immediately following sibling of each element in the set of matched elements, optionally filtered by a selector.

etc ...

Traversing HTML DOM

- Tree traversing example with “closest()”

.closest()
Get the first ancestor element that matches the selector, beginning at the current element and progressing up through the DOM tree.

```
<form name=”primaryUserForm” action=”primaryAction”>
   . . .
   <div class="taskBar">
          <div class="primaryButtonContainer">
               <button class="okButtonClass" title="OK" >OK</button>
          </div>
          <button class="cancelButtonClass">Cancel</button>
      </div>
</form>
<form name=”secondaryUserForm” action=”secondaryAction”>
   . . .
   <div class="taskBar">
          <div class="primaryButtonContainer">
               <button class="okButtonClass" title="OK" >OK</button>
          </div>
          <button class="cancelButtonClass">Cancel</button>
      </div>
</form>

<script>
   $(".okButtonClass").click( function(){ $(this).closest(“form”).submit(); });
</script>
```

Manipulating HTML & DOM

Manipulating HTML & DOM

Methods to modify entire elements (or groups of elements) themselves—inserting, copying, removing, and so on.

Among all the manipulative methods, few act both as ‘getters’ and as well as ‘setters’.

.html()
Get the HTML contents of the first element in the set of matched elements.

.html(html string)
Set the HTML contents of each element in the set of matched elements.

.text()
Get the combined text contents of each element in the set of matched elements, including their descendants.

.text(text string)
Set the content of each element in the set of matched elements to the specified text.

.val()
Get the current value of the first element in the set of matched elements. Primarily used to get the values of form elements.

.val(value)
Set the value of each element in the set of matched elements. Typically used to set the values of form fields.

Manipulating HTML & DOM

Methods that get and set DOM attributes of elements.

.attr(attribute name)
Get the value of an attribute for the first element in the set of matched elements.

.attr(attribute name, value)
Set one or more attributes for the set of matched elements.

```
<form name=”secondaryUserForm” action=”secondaryAction”>
   . . .
   <div class="taskBar">
        <div class="primaryButtonContainer">
             <button class="okButtonClass" title="OK">OK</button>
        </div>
        <button class="cancelButtonClass">Cancel</button>
    </div>
</form>

<script>
   $(“.cancelButtonClass”).attr(“title”, “Click to cancel”);
</script>
```

jQuery Event

jQuery Event

Way to associate behaviors to take effect when the user interacts with the browser, and manipulate those associated behaviors as well.

.bind(event type, [event data], handler)
Attach a handler to an event for the elements. Primary means of attaching behavior to a document.

```
<script>
   $(".okButtonClass").bind('click', function(){ $
   (this).closest("form").submit(); });
</script>
```

There are shortcut methods for binding the standard event types:

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

Event handler takes a callback function.

The DOM element in context is referred as '`this`'. To make it available as a jQuery object, it is referred as `$(this)`.

Event data (optional)

This parameter is used to inject any additional information to the event handler closure.

```
<script>
   var message =  "I clicked the button!";
   $("#ok_button").bind(“click”, {msg:message}, function(event){
        alert(event.data.msg);
    });
</script>
```

jQuery Event

Manual event execution.

Any event handlers attached with `.bind()` can be fired manually with the `.trigger()` method.

.trigger(event type, extra parameters)
Execute all handlers and behaviors attached to the matched elements for the given event type.

```
<form name=”secondaryUserForm” action=”secondaryAction”>
    . . .
    <input id=”inputField” type=”text” name=”field1”/>
    <div class="taskBar">
            <div class="primaryButtonContainer">
                 <button class="okButtonClass" title="OK">OK</button>
            </div>
            <button class="cancelButtonClass">Cancel</button>
       </div>
</form>

<script>
    $(“.okButtonClass”).click(function(){ alert(“I clicked OK”); });

    $(“#inputField”).click(function(){
        $(“.okButtonClass”).trigger(“click”);
    });
</script>
```

jQuery Event

Few key methods.

ready(handler)

Specify a function to execute when the DOM is fully loaded where `handler` is a function to execute after the DOM is ready.

```
<script>
    $(document).ready(function(){

        $(“.okButtonClass”).click(function(){ alert(“I clicked OK”); });
        $(“#inputField”).click(function(){
            $(“.okButtonClass”).trigger(“click”);
        });

    });
</script>
```

jQuery Event

Few key methods.

event.preventDefault()

When used along with an event handler, the default action of the event will not be triggered.

```
<style>#detailContent { display: none; }</style>

<div>
   <a id="detailLink" href="Click to see more">More</a>
</div>
<div id="detailContent">
   This is detailed content. It'll be visible only
   when user clicks the "More" link.
</div>

<script>
   $(document).ready(function(){

      $("#detailLink").click(function(event){
         event.preventDefault();

         $("#detailContent").css("display","block");
      });

   });
</script>
```

jQuery Event

Few key methods.

event.target

The DOM element that initiated the event.

```
<div class="taskBar">
     <div class="primaryButtonContainer">
          <button class="okButtonClass" title="OK">OK</button>
     </div>
      <button class="cancelButtonClass">Cancel</button>
   </div>

<script>
   $(document).ready(function(){

      $(“button”).click(function(event){
         $targetElement = $(event.target);
         if( $targetElement.hasClass(“okButtonClass”) ){
            alert(“I clicked OK button”);
         }
      });

   });
</script>
```

jQuery Effects (animation)

jQuery Effects (animation)

Effects methods to be used to perform animations on DOM elements.

By default, it provides number of frequently used effects as well as mechanism to achieve custom ones.

Few preset effects.

.show()

```
<script>
    $(document).ready(function(){

        $(“#detailLink”).click(function(event){
            event.preventDefault();

            $(“#detailContent”).show();
        });

    });
</script>
```

.hide()

```
<script>
    $(document).ready(function(){

        $(“#detailLink”).click(function(event){
            event.preventDefault();

            $(“#detailContent”).hide();
        });

    });
</script>
```

jQuery Effects (animation)

Custom effects using `animate` method.

`.animate`(properties, [duration], [easing], [callback])

Perform a custom animation of a set of numeric CSS properties.

```
<script>
   $(document).ready(function(){

      $(“#detailLink”).click(function(event){
         event.preventDefault();

      $(".okButtonClass").animate(
         { left: ‘50px’, height: ‘20px’ },
         300,
         function(){ alert(‘Done!’); }
      )});

   });
</script>
```

jQuery Effects (animation)

Custom effects using `animate` method.

`.animate`(properties, [duration], [easing], [callback])

Perform sequential effects based on a set of numeric CSS properties.

```
<script>
   $(document).ready(function(){

      $(“#detailLink”).click(function(event){
         event.preventDefault();

      $(".okButtonClass").animate(
         { left: ‘50px’ },
         300,
         function(){
            $(".okButtonClass").animate( { opacity: “0” };
         }
      )});

   });
</script>
```

AJAX with jQuery

AJAX with jQuery

Asynchronous HTTP requests.

Shorthand methods

jQuery.get(url, [data], [callback(data, textStatus, XMLHttpRequest)], [dataType])

url A string containing the URL to which the request is sent.
data A map or string that is sent to the server with the request.
callback (data, textStatus, XMLHttpRequest)A callback function that is executed if the request succeeds.
dataType The type of data expected from the server.

```
<script>
   $(document).ready(function(){
      $.get({
         “halloajax.html”,
         { text: “Hi” },
         function(data){
            alert(“Request sent”);
         },
         “html”
      });
   });
</script>
```

AJAX with jQuery

Asynchronous HTTP requests.

Shorthand methods

jQuery.post(url, [data], [success(data, textStatus, XMLHttpRequest)], [dataType])

url

A string containing the URL to which the request is sent.

data

A map or string that is sent to the server with the request.

success(data, textStatus, XMLHttpRequest)

A callback function that is executed if the request succeeds.

dataType

The type of data expected from the server.

```
<script>
    $(document).ready(function(){
        $.post({
            “halloajax.html”,
            { text: “Hi” },
            function(data){
                alert(“Request sent”);
            },
            “html”
        });
    });
</script>
```

AJAX with jQuery

Asynchronous HTTP requests.

Low level method

jQuery.ajax(settings)

settings A set of key/value pairs that configure the Ajax request. All options are optional.

Few frequently used settings:

type *String*
'GET' (default) or 'POST'

url *String*

data *Object / String*
Data to be sent to the server. It is converted to a query string, if not already a string.

dataType *String*
The type of data that you're expecting back from the server. If none is specified, jQuery will intelligently try to get the results, based on the MIME type of the response.

AJAX with jQuery

Asynchronous HTTP requests.

Low level method

jQuery.ajax(settings)

settings A set of key/value pairs that configure the Ajax request. All options are optional.

Few frequently used settings (contd.):

success(data, textStatus) *Function*
A callback function to be called if the request succeeds. The function gets passed two arguments: The data returned from the server, formatted according to the 'dataType' parameter; a string describing the status.

error(XMLHttpRequest, textStatus, errorThrown) *Function*
A callback function to be called if the request fails. The function is passed three arguments: The XMLHttpRequest object, a string describing the type of error that occurred and an optional exception object, if one occurred.

complete(XMLHttpRequest, textStatus) *Function*
A function to be called when the request finishes (after success and error callbacks are executed). The function gets passed two arguments: The XMLHttpRequest object and a string describing the status of the request.

AJAX with jQuery

Asynchronous HTTP requests.

Simplest method

.load(url, [data], [complete(responseText, textStatus, XMLHttpRequest)])

url

A string containing the URL to which the request is sent.

data

A map or string that is sent to the server with the request.

complete(responseText, textStatus, XMLHttpRequest)

A callback function that is executed when the request completes.

For loading a page fragment, a simple implementation can be:

```
<script>
   $(document).ready(function(){
      $("#myContent").load("halloajax.html #contentDiv);
   });
</script>
```

Using plugin

Using plugin

Way of packaging reusable codes.

jQuery plugin is a reusable component created by extending jQuery.

It allows methods to be added to its library. These methods are passed the jQuery object within the JavaScript ‘this’ object.

The method should return ‘this’ (jQuery object), so other functions can be chained.

Two ways to do it

- Creating new function to jQuery namespace
- Creating jQuery object methods

Using plugin

Creating new function to jQuery namespace

Define the plugin (Create a jQuery debug plugin that will use console or alert appropriately)

```
<script>

    jQuery.myDebugPlugin = function(obj){
        if( window.console ){
            console.log(obj);
        } else {
            alert(obj);
        }
    };

</script>
```

Call the plugin

```
<script>

    jQuery.myDebugPlugin(obj);

</script>
```

Using plugin

Creating jQuery object methods

Define the plugin (Add a ‘navigation’ role attribute to elements with class called ‘aria-nav’)

```
<script>

    jQuery.fn.myAriaPlugin = function(){
        // Make it chainable
        return this.each( function(){
            $thisElement = jQuery(this);
            if( $thisElement.hasClass(‘aria-nav’) ){
                $thisElement.attr(‘role’, ‘navigation’);
            }
        });
    };

</script>
```

Call the plugin

```
<script>

    $(“div”).myAriaPlugin();

</script>
```

Using plugin

Using configuration/options

Plugins with configuration parameters should have their own default which can be updated with user defined one. Updating configuration parameters happen with `jQuery.extend`

Define the plugin (A plugin that will expand to a given height upon click)

```
<script>
   jQuery.fn.myExpandPlugin = function(options){
      // Define defaults and update them with available options
      configOptions = jQuery.extend({ height: “200” }, options);

      // Make it chainable
      return this.each( function(){
         $thisElement = jQuery(this);
         $thisElement.click(function(){

            // Use css height to expand with animation
            $thisElement.animate({ height: configOptions.height }, 300);
         });
      });
   };
</script>
```

Call the plugin

```
<script>
   $(“div”).myExpandPlugin({height:’150’});
</script>
```

Using plugin

Making it all together

- Wrap your code within an anonymous function so that it is scoped within itself
- Use $ alias by passing `jQuery` object to the function
- Do not assume configuration options are always being passed by user
- Always return the `jQuery` object

```
<script>
    (function($){
        $.fn.myExpandPlugin = function(options){
            // Define defaults
            configOptions = { height: “200” };

            // Update configuration with available options
            if(options){ $.extend({ height: “200” }, options); }
            this.each( function(){
                $thisElement = $(this);
                $this.click(function(){

                    // Use css height to expand with animation
                    $thisElement.animate({ height: configOptions.height }, 300);
                });
            });

            // Make it chainable
            return this;
        };
    })(jQuery);
</script>
```