Javascript with jQuery, Animation, AJAX

Aryo Pinandito


jQuery
write less, do more.

Why jQuery?

- Lightweight – 14kb (Minified and Gzipped)
- Cross-browser support (IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+)
- CSS-like syntax – easy for developers/non-developers to understand
- Active developer community
- Extensible – plugins

JQuery

- Powerful JavaScript library
  - Simplify common JavaScript tasks
  - Access parts of a page
    - using CSS or XPath-like expressions
  - Modify the appearance of a page
  - Alter the content of a page
  - Change the user's interaction with a page
  - Add animation to a page
  - Provide AJAX support
  - Abstract away browser quirks

JQuery

- Powerful JavaScript library
  - Access parts of a page using CSS or XPath-like expressions
  - Modify the appearance of a page
  - Alter the content of a page
  - Change the user's interaction with a page
  - Rich library of methods for AJAX development (AJAX = Asynchronous JavaScript and XML)
  - With jQuery and AJAX, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post.

Basic JQuery

- Selecting part of a document is a fundamental operation
- A JQuery object is a wrapper for a selected group of DOM nodes
- $() function is a factory method that creates JQuery objects
- $("dt") is a JQuery object containing all the "dt" elements in the document

Basic JQuery

- .addClass() method changes the DOM nodes by adding a ‘class’ attribute
- The ‘class’ attribute is a special CSS construct that provides a visual architecture independent of the element structures
- $("dt").addClass("emphasize") will change all occurrences of <dt> to <dt class=“emphasize”>

Basic JQuery

- To make this change, put it in a function and call it when the document has been loaded and the DOM is created. Example Function:

```
function doEmph() {
    $("dt").addClass("emphasize")
}
<body onLoad="doEmph()">
```

- Structure and appearance should be separated!

Basic JQuery

- JQuery provides an independent scheduling point after DOM is created and before images are loaded:

```
$(document).ready(doEmph);
```

- No HTML changes required. All done in the script.
- Better solution:

```
$(document).ready(function(){
   $("dt").addClass("emphasize")
});
```

```
<html><head>
<script src="jquery.js" type="text/javascript"></script>
<script src="test.js" type="text/javascript"></script>
…
```

JQuery Changes DOM

- `.attr({'name', 'value'})`
  - sets a new attribute (or many)
- `$('<i>hello</i>')`
  - Creates a new element
- `$('<i>hello</i>').insertAfter('div.chapter p');`
  - Creates element and inserts it into the document
- `.html() or .text() or .empty()`
  - will replace matched elements with newly created elements

JQuery Selectors

- CSS
  - p element name
  - #id identifier
  - .class classname
  - p.class element with class
  - p a anchor as any descendant of p
  - p > a anchor direct child of p

JQuery Selectors

- XPath
  - /html/body//div paths
  - a[@href] anchor with an href attr
  - div[ol] div with an ol inside
  - //a[@ref='nofollow'] any anchor with a specific value for the ref attribute

JQuery Filters

• p:first	first paragraph
• li:last	last list item
• a:nth(3)	fourth link
• a:eq(3)	fourth link
• p:even or p:odd	every other paragraph
• a:gt(3) or a:lt(4)	every link after the 4th or up to the fourth
• a:contains(‘click’)	links that contain the word click

Example

- JQuery uses chaining as follows

```
$(‘a:contains("ECS")’).parent().addClass("emphasize")
```

```
$(‘p.data’).parent().hide('slow')
```

JQuery Events

- bind(eventname, function) method
  - `'click'`
  - `'change'`
  - `'resize'`
-
```
$("a[@href]").bind('click',function(){
    $(this).addClass('red');
});
```

Other JQuery Effects using CSS

```
.css('property', 'value')
.css({'prop1':'value1', 'prop2':'value2', …})
```

- Example:

```
p.css('color', 'red')
```

Animation using JQuery

Ex: Show/Hide The Old Way

```
<script>
function toggle_visibility(id) {
       var e = document.getElementById(id);
       if(e.style.display == 'block')
          e.style.display = 'none';
       else
          e.style.display = 'block';
}


<div id="foo">This is FOO</div>

<a href="#" onclick="toggle_visibility('foo');">
Click here to toggle visibility of #foo</a>
```

Ex: Show/Hide with jQuery

```
$().ready(function(){
      $("a").click(function(){
            $("#foo").toggle('slow');
            return false;
      });
 });
```

JQuery provides simple show and hide animation using .toggle() and can be wrapped on event using anonymous function.

http://api.jquery.com/toggle/

Controlling Animation Speed

`.hide(speed)` or `.show(speed)`

- Where speed is 'slow', 'normal' or 'fast'
- Or you can specify using integer number value which is translated into milliseconds

`.hide(500)` or `.show(200)`

Custom Animation Effects using jQuery .animate()

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

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

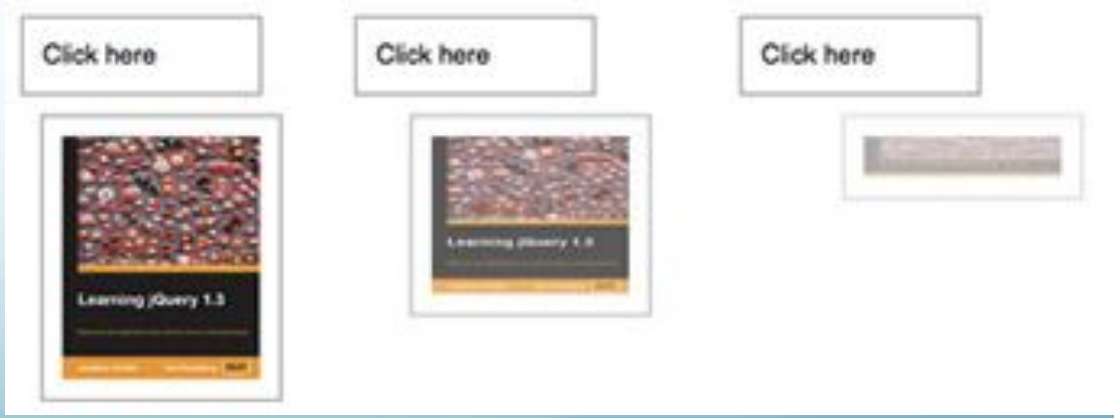
- More information
- http://api.jquery.com/animate/

jQuery .animate() Example

```
<div id="clickme">
  Click here
</div>
<img id="book" src="book.png" alt="" width="100" height="123"
  style="position: relative; left: 10px;">
```

To animate the opacity, left offset, and height of the image simultaneously:

```
$( "#clickme" ).click(function() {
  $( "#book" ).animate({
    opacity: 0.25,
    left: "+=50",
    height: "toggle"
  }, 5000, function() {
    // Animation complete.
  });
});
```



Asynchronous JavaScript and XML (AJAX)

'Old' Way of Getting/Sending Data

- In traditional JavaScript coding, to fetch or send information to or from a database or a file on webserver requires making an HTML form and GET or POST data to the server
- User clicks "Submit" button to send/get the information, waits for the server to respond, then a complete new page loads with results
- Since server returns a new page each time user submits input, traditional web apps can run slowly and tend to be less user-friendly

Things change...

- Until recently, we didn't have any alternative to this load/wait/respond method of web browsing.
- Ajax (sometimes written AJAX) is a means of using JavaScript to communicate with a web server without submitting a form or loading a new page.
- Ajax makes use of a built-in object, XMLHttpRequest, to perform this function.
  - This object is not yet part of the DOM (Document Object Model) standard
  - The term "Ajax" was coined in 2005, but the XMLHttpRequest object was first supported by Internet Explorer several years before this.

AJAX

- Ajax stands for “Asynchronous JavaScript and XML”.
- The word “asynchronous” means that the user isn't left waiting for the server the respond to a request, but can continue using the web page.

Typical Method Using AJAX

1. A JavaScript creates an XMLHttpRequest object, initializes it with relevant information as necessary, and sends it to the server. The script (or web page) can continue after sending it to the server.
2. The server responds by sending the contents of a file or the output of a server side program (written, for example, in PHP).
3. When the response arrives from the server, a JavaScript function is triggered to act on the data supplied by the server.
4. This JavaScript response function typically refreshes the display using the DOM, avoiding the requirement to reload or refresh the entire page.

The Back End

- The part of the Ajax application that resides on the web server is referred to as the "back end".
- This back end could be simply a file that the server passes back to the client, which is then displayed for the user.
- Alternatively, the back end could be a program, written in PHP, Perl, Ruby, Python, C, or some other language that performs an operation and sends results back to the client browser.
- An XMLHttpRequest object can send information using the GET and POST methods to the server in the same way that an HTML form sends information.

Example – Ajax The Old Way

```
function GetXmlHttpObject(handler) {
        var objXmlHttp = null;    //Holds the local xmlHTTP object instance

        //Depending on the browser, try to create the xmlHttp object
        if (is_ie){
                var strObjName = (is_ie5) ? 'Microsoft.XMLHTTP' : 'Msxml2.XMLHTTP';
                try{
                        objXmlHttp = new ActiveXObject(strObjName);
                        objXmlHttp.onreadystatechange = handler;
                }
                catch(e){
                //Object creation errored
                alert('Verify that activescripting and activeX controls are enabled');
                return;
                }
        }
                else{
                // Mozilla | Netscape | Safari
                objXmlHttp = new XMLHttpRequest();
                objXmlHttp.onload = handler;
                objXmlHttp.onerror = handler;
        }
        //Return the instantiated object
        return objXmlHttp;
}
```

Example – Ajax with jQuery

GET

- ```
  $.get("process.php", { name: "John", time:
  "2pm" }, function(data){
      alert("Data Loaded: " + data);
  });
  ```

POST

- ```
  $.post("process.php", { name: "John", time:
  "2pm" }, function(data){
      alert("Data Loaded: " + data);
  });
  ```

Assignment

Student Assignment

- Made a simple login page using HTML5 and CSS3 that consists of two text inputs for:
  - Username
  - Password
- Using jQuery:
  - Validate the form that the username input is a valid email and may not be left blank
  - Validate the password that it should have numbers, lowercase, and uppercase letters. The password should consists of 8 or more letters.
  - Send the login data to a web server using jQuery's AJAX
- The server side script in web server should process the data and returns a success or failure message depends on the username and password received.
- The login page should display any messages given from server side scripts to the user.

References

- http://www.w3schools.com/ajax/default.asp
- http://www.internetnews.com/dev-news/article.php/3676226
- http://daniel.lorch.cc/docs/ajax_simple

Questions?