Lecture Sheet on "PHP+MySQL"
Day-19: (jQuery)

Objectives	**Topics**
❖ Introduction jQuery ❖ The Basics, using with CSS, Ajax & PHP	❖ What is jQuery and it's Features ❖ The Basics ❖ AJAX interface/menu using jQuery/PHP

jQuery is a lightweight JavaScript library that emphasizes interaction between JavaScript and HTML. It was released January 2006 at BarCamp NYC by John Resig. Dual licensed under the MIT License and the GNU General Public License, jQuery is free and open source software.

Features

jQuery contains the following features:

- **DOM**
- DOM traversal and modification, (including support for CSS 1-3 and basic **XPath**)
- Events
- CSS
- Effects and animations
- **Ajax**
- Extensibility
- Utilities - such as browser version and the each function.
- JavaScript Plugins

The Basics

This is a basic tutorial, designed to help you get started using jQuery. If you don't have a test page setup yet, start by creating a new HTML page with the following contents:

```
<html>
<head>
  <script type="text/javascript" src="path/to/jquery.js"></script>
  <script type="text/javascript">
    // Your code goes here
  </script>
</head>
<body>
  <a href="http://jquery.com/">jQuery</a>
</body>
</html>
```

Edit the `src` attribute in the script tag to point to your copy of jquery.js. For example, if jquery.js is in the same directory as your HTML file, you can use:

```
<script type="text/javascript" src="jquery.js"></script>
```

You can download your own copy of jQuery from the **http://docs.jquery.com/Downloading_jQuery** page.

Launching Code on Document Ready

The first thing that most Javascript programmers end up doing is adding some code to their program, similar to this: window.onload = function(){ alert("abc") }
Inside of which is the code that you want to run right when the page is loaded. Problematically, however, the Javascript code isn't run until all images are finished downloading (this includes banner ads). The reason for using window.onload in the first place is due to the fact that the HTML 'document' isn't finished loading yet, when you first try to run your code.
To circumvent both problems, jQuery has a simple statement that checks the document and waits until it's ready to be manipulated, known as the ready event:

```
$(document).ready(function(){
  // Your code here
});
```

The remaining jQuery examples will need to be placed inside the ready event so that they are executed when the document is ready to be worked on. Add the next section of code:

```
$("a").click(function(){
  alert("Thanks for visiting!");
});
```

Save your HTML file and reload the test page in your browser. Clicking the link on the page should make a browser's alert pop-up, before leaving to go to the main jQuery page.
For click and most other **events**, you can prevent the default behaviour - here, following the link to jquery.com - by calling event.preventDefault() in the event handler:

```
$("a").click(function(event){
  event.preventDefault();
  alert("As you can see, the link no longer took you to jquery.com");
});
```

Adding a Class

Another common task is adding (or removing) a CSS (Cascading Style Sheet) class. For example: $("a").addClass("test");
if you were to add some style information into the header of your script,

like this:

```
<style type="text/css"> a.test { font-weight: bold; } </style>
```

and then added the above **addClass** call - all your a elements would now be bold. To remove the class, you'd use **removeClass**

Special Effects

In jQuery, a couple handy **effects** are provided, to really make your web site stand out. To put this to the test, change the click that you added earlier to this:

```
$("a").click(function(event){
  event.preventDefault();
  $(this).hide("slow");
});
```

Now, if you click any link, it should make itself slowly disappear.

Chainability (The Magic of jQuery)

jQuery uses an interesting concept to make its code short and simple. For those familiar with object-oriented programming, this concept should be rather straightforward.

In a nutshell: Every method within jQuery returns the query object itself, allowing you to 'chain' upon it, for example:

$("a").addClass("test").show().html("foo");

Each of those individual methods (addClass, show, and html) return the jQuery object, allowing you to continue applying methods to the current set of elements.

You can take this even further, by adding or removing elements from the selection, modifying those elements and then reverting to the old selection, for example:

```
$("a")
  .filter(".clickme")
    .click(function(){
      alert("You are now leaving the site.");
    })
  .end()
  .filter(".hideme")
    .click(function(){
      $(this).hide();
      return false;
    })
  .end();
```

Which would work against the following HTML:

```
<a href="http://google.com/" class="clickme">I give a message when you leave</a>
<a href="http://yahoo.com/" class="hideme">Click me to hide!</a>
<a href="http://microsoft.com/">I'm a normal link</a>
```

Methods that modify the jQuery selection, and that can be undone with end(), are the following:

- add() ,
- children() ,
- eq() ,
- filter() ,
- find() ,
- gt() ,
- lt() ,
- next() ,
- not() ,
- parent() ,
- parents() and
- siblings() .

Callbacks and Functions

A callback is a function that is passed as an argument to another function and is executed after its parent function has completed. The special thing about a callback is that functions that appear after the "parent" can execute before the callback executes.
Another important thing to know is how to properly pass the callback. This is where I have often forgotten the proper syntax.

Callback *without* arguments

For a callback with no arguments you pass it like this:

```
$.get('myhtmlpage.html', myCallBack);
```

Note that the second parameter here is simply the function name (but *not* as a string and without parentheses). Functions in Javascript are 'First class citizens' and so can be passed around like variable references and executed at a later time.

Callback *with* arguments

"What do you do if you have arguments that you want to pass?", you might ask yourself.

Wrong

The Wrong Way (will **not** work!)

```
$.get('myhtmlpage.html', myCallBack(param1, param2));
```

This will not work because it calls

```
myCallBack(param1, param2)
```

and then passes the return value as the second parameter to $.get().

Right

The Right Way (note the use of function(){ preceding myCallBack())

```
$.get('myhtmlpage.html', function(){
  myCallBack(param1, param2);
});
```

So, by passing in an anonymous function (the part with... AJAX interface/menu using jQuery/PHP

```
 function(){
   myCallBack(param1, param2);
 }
```

More Reading

jQuery users have developed a number of helpful **tutorials** to help guide you, the new jQuery user, through the process of learning how to best use this library. Again, if we could not describe properly the features of the jQuery-library... simply let us know about it.

AJAX interface/menu using jQuery/PHP

This tutorial teaches you how to present data using AJAX (using jQuery) with some neat effects. We can use PHP/MySQL to store the data and call it. In this tutorial I`ll teach you how to do so with PHP. So lets begin !

HTML Page

Our HTML file will contain the javascript as well as the CSS styles embedded within it. We can also create the effect by creating individual files for each i.e one for javascript functions and the other for the CSS. Anyways in this tutorial, both of them will be embedded within the html page. Now First fire up your favourite HTML editor like Dreamweaver.
We will be using The jQuery Framework - jquery.js

Step 1 - The CSS

Lets add our css first. First create a new HTML file, call it interface.html.
The following is the css code.

```
<style type="text/css">
body { margin:0 auto 0 auto; color:#000; font-family:Georgia, "Times New Roman", Times, serif; font-size:16px; background-color:#666666; }

```

```
.page { margin:100px auto 0 auto; width:750px;}

/*MENU START*/
#menu  { list-style:none; margin:0px; padding:0px;}

#menu li { list-style:none; display:inline; }
        li.active a { background-color:#FFF;color:#000; }

        #menu li a,#menu li a:link { float:left; background-color:#3c3c3c; margin-right:5px; padding:7px; color:#FFFFFF; text-decoration:none; width:6em; text-align:center;}
        #menu li a:visited { }
        #menu li a:hover { background-color:#327cc8 }
        #menu li a:active { background-color:#FFF;color:#000; }
        /* MENU END*/

        #outcontent {clear:both; background-color:#FFFFFF; }
        .content {background-color:#FFF; padding:10px; height:300px; margin:0px; }

        #loading { clear:both; background:url(wait.gif) center top no-repeat; text-align:center;padding:33px 0px 0px 0px; font-size:12px;display:none; font-family:Verdana, Arial, Helvetica, sans-serif; }
        </style>
```

The CSS is pretty straight forward.

- #page will be assigned to a div, which will hold our menu and other divs
- #menu is the CSS to display our menu tabs.
- #loading is the CSS to display the AJAX loading image.
- #outcontent is the container to contain .content

Put the above code inside the HEAD tag of interface.html. So interface.html will look something like the following.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>interface.html</title>
<style type="text/css">
body { margin:0 auto 0 auto; color:#000; font-family:Georgia, "Times New Roman", Times, serif; font-size:16px; background-color:#666666; }

.page { margin:100px auto 0 auto; width:750px;}

```

```
#menu  { list-style:none; margin:0px; padding:0px;}

#menu li { list-style:none; display:inline; }
li.active a { background-color:#FFF;color:#000; }

#menu li a,#menu li a:link { float:left; background-color:#3c3c3c; margin-right:5px; padding:7px; color:#FFFFFF; text-decoration:none; width:6em; text-align:center;}
#menu li a:visited { }
#menu li a:hover { background-color:#327cc8 }
#menu li a:active { background-color:#FFF;color:#000; }

.content {
background-color:#FFF; background:url(ajaxinterface.jpg) bottom right no-repeat; padding:10px; height:300px; margin:0px; }

#loading { clear:both; background:url(wait.gif) center top no-repeat; text-align:center;padding:33px 0px 0px 0px; font-size:12px;display:none; font-family:Verdana, Arial, Helvetica, sans-serif; }

#outcontent {clear:both; background-color:#FFFFFF; }

</style>
</head>

<body>
</body>
</html>
```

The above is the HTML page interface.html with the CSS defined in it.

Step 2 - The HTML

Ok! Since now we have coded the CSS, lets create the HTML divs inside interface.html.

1. Inside the BODY tag lets create a div which will contain the menu and the subsequent divs.

```
<body>
   <div class="page">
   <!-- Menu and other divs go here !-->
   </div>
<body>
```

```
<body>
    <div class="page">
    <!-- Menu and other divs go here !-->
    </div>
</body>
```

2. Then we`ll create the menu Tabs.

```
<body>
   <div class="page">
   <ul id="menu">
   <!-- you can add more tabs as you wish !-->
          <li id="about"  ><a href="#" title="about">About</a></li>
         <li id="portfolio"><a href="#" title="portfolio">Portfolio</a></li>
         <li id="contact"><a href="#" title="contact">Contact</a></li>
   </ul>
   <br style="clear:both;" />
   </div>
</body>
```

3. Now we need a container to display our Loading text and our content, so we`ll create 2 divs and lets put these divs inside another div called outcontent. The following is how it`ll look.

```
<body>
   <div class="page">
      <div>
         <ul id="menu">
         <!-- you can add more tabs as you wish !-->
                  <li id="about"  ><a href="#" title="about">About</a></li>
                 <li id="portfolio"><a href="#" title="portfolio">Portfolio</a></li>
                 <li id="contact"><a href="#" title="contact">Contact</a></li>
         </ul>
      <br style="clear:both;" />
      </div>
      <!-- #outcontent - Overall div containing Loading div &amp; content div !-->
      <div id="outcontent">
          <div id="loading">LOADING</div>
         <div class="content"></div>
      </div>
      <!-- #outcontent end !-->
   </div>
</body>
```

interface.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>interface.html</title>
<style type="text/css">
body { margin:0 auto 0 auto; color:#000; font-family:Georgia, "Times New Roman", Times, serif; font-size:16px; background-color:#666666; }
```

```
.page { margin:100px auto 0 auto; width:750px;}

#menu   { list-style:none; margin:0px; padding:0px;}

#menu li { list-style:none; display:inline; }
li.active a { background-color:#FFF;color:#000; }

#menu li a,#menu li a:link { float:left; background-color:#3c3c3c; margin-right:5px; padding:7px;
color:#FFFFFF; text-decoration:none; width:6em; text-align:center;}
#menu li a:visited { }
#menu li a:hover { background-color:#327cc8 }
#menu li a:active { background-color:#FFF;color:#000; }

.content {
background-color:#FFF; padding:10px; height:300px; margin:0px; }

#loading { clear:both; background:url(wait.gif) center top no-repeat; text-align:center;padding:33px
0px 0px 0px; font-size:12px;display:none; font-family:Verdana, Arial, Helvetica, sans-serif; }

#outcontent {clear:both; background-color:#FFFFFF; }

</style>
</head>

<body>
<div class="page">
<div>
<ul id="menu">
                  <li id="about"><a href="#" title="about">About</a></li>
                <li id="portfolio"><a href="#" title="portfolio">Portfolio</a></li>
                <li id="contact"><a href="#" title="contact">Contact</a></li>
</ul>
<br style="clear:both;" />
</div>
<div id="outcontent">
<div id="loading">LOADING</div>
<div class="content"></div>
</div>
</div>
</body>
</html>
```

The page should look something like the following

Step 3 - Add the Javascript Magic !

This page would do for static pages, but we are making an AJAX powered page ! Its time to add the magic.

1. First lets link to the jquery library

```
<script src="jquery.min.js" type="text/javascript"></script>
```

2. Next lets open the script tag again and add the following code. The following code is pretty explanatory. I have added a lot of comments to show what each and every line does.

```
<script>
 // When the document loads do everything inside here ...
   $(document).ready(function(){
   $('.content').load('boo.php'); //by default initially load text from boo.php
      $('#menu a').click(function() { //start function when any link is clicked
                  $(".content").slideUp("slow");
                   var content_show = $(this).attr("title"); //retrieve title of link so we can compare with php file
                     $.ajax({
                     method: "get",url: "boo.php",data: "page="+content_show,
                              beforeSend: function(){$("#loading").show("fast");}, //show loading just when link is clicked
                              complete: function(){ $("#loading").hide("fast");}, //stop showing loading when the process is complete
                              success: function(html){ //so, if data is retrieved, store it in html
                              $(".content").show("slow"); //animation
                              $(".content").html(html); //show the html inside .content div
                        }
                     }); //close $.ajax(
               }); //close click(
            }); //close $(
        </script>
```

Voila ! thats it.

Our Final interface.html page

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>interface.html</title>
<script src="jquery.min.js" type="text/javascript"></script>
 <script>
  // When the document loads do everything inside here ...
    $(document).ready(function(){
    $('.content').load('boo.php'); //by default initally load text from boo.php
```

```
        $('#menu a').click(function() { //start function when any link is clicked
                $(".content").slideUp("slow");
                 var content_show = $(this).attr("title"); //retrieve title of link so we c
an compare with php file
                  $.ajax({
                  method: "get",url: "boo.php",data: "page="+content_show,
                  beforeSend: function(){$("#loading").show("fast");}, //show loadin
g just when link is clicked
                  complete: function(){ $("#loading").hide("fast");}, //stop showing l
oading when the process is complete
                  success: function(html){ //so, if data is retrieved, store it in html
                  $(".content").show("slow"); //animation
                  $(".content").html(html); //show the html inside .content div
                }
             }); //close $.ajax(
        }); //close click(
     }); //close $(
   </script>
   <style type="text/css">
   body { margin:0 auto 0 auto; color:#000; font-family:Georgia, "Times New Roman"
, Times, serif; font-size:16px; background-color:#666666; }

   .page { margin:100px auto 0 auto; width:750px;}
   #menu  { list-style:none; margin:0px; padding:0px;}

   #menu li { list-style:none; display:inline; }
   li.active a { background-color:#FFF;color:#000; }
   #menu li a,#menu li a:link { float:left; background-color:#3c3c3c; margin-right:5px;
 padding:7px; color:#FFFFFF; text-decoration:none; width:6em; text-align:center;}
   #menu li a:visited { }
   #menu li a:hover { background-color:#327cc8 }
   #menu li a:active { background-color:#FFF;color:#000; }

   .content {
   background-color:#FFF; padding:10px; height:300px; margin:0px; }
   #loading { clear:both; background:url(wait.gif) center top no-repeat; text-align:cente
r;padding:33px 0px 0px 0px; font-size:12px;display:none; font-family:Verdana, Arial, Helv
etica, sans-serif; }

   #outcontent {clear:both; background-color:#FFFFFF; }

   </style>
   </head>

   <body>
   <div class="page">
   <div>
   <ul id="menu">
```

```
            <li id="about"  ><a href="#" title="about">About</a></li>
            <li id="portfolio"><a href="#" title="portfolio">Portfolio</a></li>
            <li id="contact"><a href="#" title="contact">Contact</a></li>
    </ul>
    <br style="clear:both;" />
    </div>
    <div id="outcontent">
    <div id="loading">LOADING</div>
    <div class="content"></div>
    </div>
    </div>
    </body>
    </html>
```