**CENG 256**

**Internet Programming**

**LAB 3**

**JQuery Form Basics**

Excerpt from (<http://www.htmlgoodies.com>)

So what is jQuery? From the jQuery web site: "*jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.*"

Okay, but what does that mean to me? Put simply, jQuery will help you write JavaScript faster using a simpler syntax. Instead of writing extra lines of code or the writing the same code over and over, you can use jQuery to consolidate the code. jQuery will do the heavy lifting while you can concentrate on the more important stuff.

jQuery also supports the idea of plugins. Plugins allow people to create mini-libraries that complement jQuery. The plugins can be anything from form validation to picture slide shows. We will look at plugins in future articles.

**Where to Start**

The first thing you will need to do to start using jQuery is to decide if you want to host the jQuery library or use a content delivery network (CDN). If you plan on hosting the jQuery library, you will need to download it from the jQuery web site (http://jquery.com) and upload it to your web server.

The best way to use jQuery is to use a CDN. Both Microsoft and Google offer jQuery on their CDNs. There are several advantages to doing this. The first is that a CDN is spread out over the Internet. When someone comes to your web site and requests the jQuery library, it will be provided to them by the closest hosting site. This will help speed up the download of the library to their local machine. Which brings us to another point, the jQuery library is cached on the user's machine to help with speed.

The second reason is that if the user has been to another web site that had this CDN reference, the user will already have the jQuery file on their system. This will speed up the loading of your web page since they will not need to download the file again.

When you are looking at the different libraries for jQuery, you will notice there are two. One of them has a ".min" in the file name. This is a "minified" version of the file. It is smaller and will load faster. Why two versions? The non-minified version is easier to read, but since there is a lot of white space, it makes the file larger. It is used to help with debugging.

So whenever possible, always use the CDN and the minified version. This will help speed up your web site.

Next we need to create a reference to the jQuery library. This is done by adding a line in between the opening (<head>) and closing (</head>) HTML header tags. We will use the Microsoft CDN.

<head>

<script type="text/javascript" src="https://ajax.microsoft.com/ajax/jQuery/jquery-1.4.2.min.js"></script>

</head>

jQuery works like normal JavaScript in the fact that you use <script> tags to denote the script. These can be placed in the head section or the body section of your web page.

<script>

Your script goes here

</script>

**A Quick Example**

First, start by creating an HTML document. You can call this anything you want. Enter the code below:

<html>

<head>

<script type="text/javascript" src="https://ajax.microsoft.com/ajax/jQuery/jquery-1.4.2.min.js"></script>

<script type="text/javascript">

</script>

</head>

<body>

<a href="#">Click Me</a>

</body>

</html>

Currently, the link "Click Me" does nothing. Let us change that. Enter the code below in between the script tags.

$(document).ready(function(){

$("a").click(function(event){

alert("You clicked me!");

});

});

What we did here was add a click event to the "Click Me" link. When it was clicked, it showed a JavaScript alert.

**What is Up with the $**

The dollar sign ($) denotes a jQuery constructor. It can be written either as $() or jQuery(). The dollar sign is just a shortcut. All jQuery constructors must start with this.

The constructor starts with a $ and the selector is enclosed in parenthesis. A selector can be an element (tag), an element ID, an element class name. This is similar to the way CSS works. In the example above, we used the anchor tag (a) as a selector. When you use a selector, jQuery searches through the Document Object Model (DOM) to find all selectors that match. In the case of the example above, if we had more than one anchor tag, all anchor tags would now have a click event.

Since selectors work similar to CSS, you will need to specify the selectors in a similar way. When referencing HTML elements, you will just use the element as is. See the following examples:

$("a") - all anchor tags

$("p") - all paragraph tags

$("p.subject") - all paragraph tags with a class of subject

When referencing class names, you must specify the period (.) before the class name as in the following examples:

$(".subject") - all elements with the class name of subject

$("p.subject") - all paragraph tags with a class of subject

And last, when referencing IDs, you must specify the # before the ID name:

$("#subject") - the element with the ID of subject

There are a few other types of selectors and we will talk about those in future articles. For now, these basic selectors should get you started.

**Making Sure the Document is Ready**

As with JavaScript, you may want to execute your jQuery code after the HTML document has completely loaded and is ready to be manipulated. This is where the $(document).ready function comes in to play. This will keep the JavaScript and jQuery code unloaded until the document is ready. Once the document has loaded, the function will execute. This keeps the code from executing before the document is ready which could cause issues. If you tried to manipulate a DOM element before it was ready, you would lose that functionality.

**Complete Code Example**

<html>

<head>

<script type="text/javascript" src="https://ajax.microsoft.com/ajax/jQuery/jquery-1.4.2.min.js"></script>

<script type="text/javascript">

$(document).ready(function(){

$("a").click(function(event){

alert("You clicked me!");

});

});

</script>

</head>

<body>

<a href="#">Click Me</a>

</body>

</html>

View full article here-

http://www.htmlgoodies.com/tutorials/forms/article.php/3895776/HTML-Forms-jQuery-Basics---Getting-Started.htm

**PART1: Core Validation**

What We're Building

In this example, we have a simple contact form with name, email, and phone number. The form submits all the fields to a php script without page refresh, using native jQuery functions (native meaning, you don't need to download any extra plugins to make it work.

Step 1 - Build the HTML Form

Let's take a look at our html markup. We begin with our basic html form:

<div id="contact\_form">

<form name="contact" action="">

  <fieldset>

    <label for="name" id="name\_label">Name</label>

    <input type="text" name="name" id="name" size="30" value="" class="text-input" />

    <label class="error" for="name" id="name\_error">This field is required.</label>

    <label for="email" id="email\_label">Return Email</label>

    <input type="text" name="email" id="email" size="30" value="" class="text-input" />

    <label class="error" for="email" id="email\_error">This field is required.</label>

    <label for="phone" id="phone\_label">Return Phone</label>

    <input type="text" name="phone" id="phone" size="30" value="" class="text-input" />

    <label class="error" for="phone" id="phone\_error">This field is required.</label>

      <br />

    <input type="submit" name="submit" class="button" id="submit\_btn" value="Send" />

  </fieldset>

</form>

</div>

You might notice that I have included a div with id contact\_form that wraps around the entire form.  
Be sure to not miss that div in your own form as we will be needing this wrapper div later on. You  
might also notice that I have left both the action and the method parts of the form tag blank. We  
actually don't need either of these here, because jQuery takes care of it all later on.

Another important thing to be sure to include is the id values for each input field. The id values  
are what your jQuery script will be looking for to process the form with.

I've added some css styles and a background image in Photoshop to produce the following form:



Step 2 - Begin Adding jQuery

The next step in the process is to add some jQuery code. I'm going to assume that you have [downloaded jQuery](http://docs.jquery.com/Downloading_jQuery), uploaded to your server, and are [referencing it in your webpage](http://docs.jquery.com/How_jQuery_Works).

Next, open up another new javascript file, reference it in your html as you would any normal javascript file,  
and add the following:

$(function() {

$(".button").click(function() {

// validate and process form here

});

});

What the first function() does is, it loads the events inside, as soon as the html document is ready. If you have done any work in jQuery previously, the function is the same as jQuery's [document.ready](http://docs.jquery.com/How_jQuery_Works#Launching_Code_on_Document_Ready) function. So we start with that, and inside we have our click function that executes on clicking the submit button with class name of "button". Ultimately what we have accomplished with these lines of code is the same as if we were to add an onclick event to the submit button in the html. The reason we do it with jQuery is for clean separation of our presentation from our scripts.

Step 3 - Write Some Form Validation

$(function() {

$('.error').hide();

$(".button").click(function() {

// validate and process form here

$('.error').hide();

var name = $("input#name").val();

if (name == "") {

$("label#name\_error").show();

$("input#name").focus();

return false;

}

var email = $("input#email").val();

if (email == "") {

$("label#email\_error").show();

$("input#email").focus();

return false;

}

var phone = $("input#phone").val();

if (phone == "") {

$("label#phone\_error").show();

$("input#phone").focus();

return false;

}

});

});

Inside our function that loads when the page is ready, we add some form validation. But the first thing you see that got added is $('.error').hide();. What this does is hides our 3 labels with class name "error". We want these labels to be hidden not just when  
the page first loads, but also when you click submit, in case one of the messages was shown to the user previously. Each error message should only appear if validation doesn't work out.

We validate by first checking if the name field was left blank by the user, and if it is, we then show the label with id of name\_error. We then place the focus on the name input field, in case the user  
is at all confused about what to do next! (I have learned to never assume too much when it comes to form users).

To explain in more detail how we are making this happen, we set a variable 'name' to the value of the input field with id "name" -- all with one line of jQuery:

var name = $("input#name").val();

We then check if that value is blank, and if it is, we use jQuery's show() method to show the label with  
id "name\_error":

if (name == "") {

$("label#name\_error").show();

}

Next, we place the form focus back on the input field with id of "name", and finally return false:

if (name == "") {

$("label#name\_error").show();

$("input#name").focus();

return false;

}

Be sure to have return false in your code, otherwise the whole form gets submitted (which defeats  
the purpose of this tutorial)! What return false does is it prevents the user from proceeding any further  
without filling out the required field(s).

Step 4 - Process our Form Submission

Now we get to the heart of the tutorial -- submitting our form without page refresh, which sends the form values to a php a script in the background. Let's take a look at all the code first, then I will break down into more detail next. Add the following code just below the validation snippet we added previously (and before the button click function is closed out):

var dataString = 'name='+ name + '&email=' + email + '&phone=' + phone;

alert (dataString);return false;

We have a lot going on here! Let's break it all down - it's so simple and so easy to use once you understand the process. We first create a string of values, which are all the form values that we want to pass along to the script that sends the email.

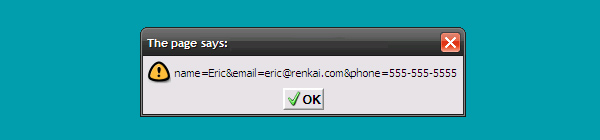
Recall previously, we had set a variable 'name' with the value of the input field with id "name", like so:

var name = $("input#name").val();

We can use that 'name' value again, as well as the 'email' and the 'phone' values, to create our dataString:

var dataString = 'name='+ name + '&email=' + email + '&phone=' + phone;

If you uncomment that alert and test your form, assuming everything has gone right so far, you should get a message similar to the following:



**Part 2: Effects and combinations**

(https://designshack.net/tutorials/creating-a-slide-in-jquery-contact-form)

**HTML CODE**

<html>

<head>

<script type="text/javascript" src="https://ajax.microsoft.com/ajax/jQuery/jquery-1.4.2.min.js">

</script>

<script src="effect.js"> </script>

<link rel="stylesheet" href="layout.css">

</head>

<body>

<div class="box">

<div id="contactFormContainer">

<div id="contactForm">

<fieldset>

<label for="Name">Name \*</label>

<input id="name" type="text" />

<label for="Email">Email address \*</label>

<input id="Email" type="text" />

<label for="Message">Your message \*</label>

<textarea id="Message" rows="3″ cols="20″></textarea>

<input id="sendMail" type="submit" name="submit" onclick="closeForm()" />

<span id="messageSent">Your message has been sent successfully!</span>

</fieldset>

</div>

<div id="contactLink">link here</div>

<div class="header">

<h1>

Company logo</h1>

</div>

</div>

</div>

</body>

</html>

**CSS:**

#contactFormContainer

{

position:absolute;

left:600px;

float:right;

}

#contactForm

{

height:277px;

width:351px;

background-image:url('bkg.jpg');

display:none;

}

#contactForm fieldset

{

padding:30px;

border:none;

}

#contactForm label

{

display:block;

color:#ffc400;

}

#contactForm input[type=text]

{

display:block;

border:solid 1px #4d3a24;

width:100%;

margin-bottom:10px;

height:24px;

}

#contactForm textarea

{

display:block;

border:solid 1px #4d3a24;

width:100%;

margin-bottom:10px;

}

#contactForm input[type=submit]

{

background-color:#4d3a24;

border:solid 1px #23150c;

color:#fecd28;

padding:5px;

}

#contactLink

{

height:40px;

width:351px;

background-image:url('slidein\_button.png');

display:block;

cursor:pointer;

}

#messageSent

{

color:#ff9933;

display:none;

}

**JQuery code**

$(document).ready(function(){

$("#contactLink").click(function(){

if ($("#contactForm").is(":hidden")){

$("#contactForm").slideDown("slow");

}

else{

$("#contactForm").slideUp("slow");

}

});

});

function closeForm(){

$("#messageSent").show("slow");

setTimeout('$("#messageSent").hide();$("#contactForm").slideUp("slow")', 2000);

}

## Conclusion

That’s it, you’re done! Now you have an idea how to set up form validation with jQuery. Please keep in mind that this doesn’t replace server-side validation. It’s still possible for a malicious user to manipulate or bypass the validation rules (e.g. using the browser’s developer tools)

Lab Question

## Step 1: Create the HTML Form

For the registration we want to collect the following user information:

1. First name
2. Last name
3. Email
4. Password

So, let’s create our form containing these input fields:

<div class="container">

<h2>Registration</h2>

<form action="" name="registration">

<label for="firstname">First Name</label>

<input type="text" name="firstname" id="firstname" placeholder="John"/>

<label for="lastname">Last Name</label>

<input type="text" name="lastname" id="lastname" placeholder="Doe"/>

<label for="email">Email</label>

<input type="email" name="email" id="email" placeholder="john@doe.com"/>

<label for="password">Password</label>

<input type="password" name="password" id="password" placeholder="&#9679;&#9679;&#9679;&#9679;&#9679;"/>

<button type="submit">Register</button>

</form>

</div>

When integrating this into a real application, don’t forget to fill in the action attribute, to make sure the form is submitted to the correct destination.

## Step 2: Create Styles for the Form

Create a new file, css/styles.css, and include it in the <head> section of your HTML file:

<link rel="stylesheet" href="css/style.css"/>

Copy the following styles into the newly created file:

/\* Styles \*/

\* {

margin: 0;

padding: 0;

}

body {

font-family: "Open Sans";

font-size: 14px;

}

.container {

width: 500px;

margin: 25px auto;

}

form {

padding: 20px;

background: #2c3e50;

color: #fff;

-moz-border-radius: 4px;

-webkit-border-radius: 4px;

border-radius: 4px;

}

form label,

form input,

form button {

border: 0;

margin-bottom: 3px;

display: block;

width: 100%;

}

form input {

height: 25px;

line-height: 25px;

background: #fff;

color: #000;

padding: 0 6px;

-moz-box-sizing: border-box;

-webkit-box-sizing: border-box;

box-sizing: border-box;

}

form button {

height: 30px;

line-height: 30px;

background: #e67e22;

color: #fff;

margin-top: 10px;

cursor: pointer;

}

form .error {

color: #ff0000;

}

Note the styles for .error, which will be used for validation error messages.

## Step 3: Create the Validation Rules

## Create a file called validation.js

## Using JQuery, write code to completely validate the form to the specifications outlined below:

## All fields are required

## The name fields must contain no numbers

## The password must be at least 8 characters long and contain at least one letter, number and special character (either an exclamation mark(!) or a dollar sign($))

## Upon correct validation, an Alert box should display the information entered in a neat format line by line.