

Send AJAX Form With JQuery

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:

```
1. <div id="contact_form">
2. <form name="contact" action="">
3.   <fieldset>
4.     <label for="name" id="name_label">Name</label>
5.     <input type="text" name="name" id="name" size="30" value="" class="text-input" />
6.     <label class="error" for="name" id="name_error">This field is required.</label>
7.
8.     <label for="email" id="email_label">Return Email</label>
9.     <input type="text" name="email" id="email" size="30" value="" class="text-input" />
10.    <label class="error" for="email" id="email_error">This field is required.</label>
11.
12.    <label for="phone" id="phone_label">Return Phone</label>
13.    <input type="text" name="phone" id="phone" size="30" value="" class="text-input" />
14.    <label class="error" for="phone" id="phone_error">This field is required.</label>
15.
16.    <br />
17.    <input type="submit" name="submit" class="button" id="submit_btn" value="Send" />
18.  </fieldset>
19. </form>
20. </div>
```

Step 1 -Build the HTML form

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.

Send AJAX Form With JQuery

Step 1 -Build the HTML form

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



The image shows a web form with a light gray background and a thin orange border. On the left side, there are three input fields stacked vertically. The first field is labeled 'Name' and has a light orange background. The second field is labeled 'Return Email' and the third is labeled 'Return Phone'. Below these fields is a 'Send' button with a white background and an orange border. To the right of the form is a dark orange background featuring a wireframe illustration of an open rectangular box and the text 'MOG MACHINE' in a bold, black, sans-serif font.

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, uploaded to your server, and are referencing it in your webpage.

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

```
1. $(function() {  
2.     $(".button").click(function() {  
3.         // validate and process form here  
4.     });  
5. });
```

Step 2 - Begin adding jQuery

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` 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

```
1. $(function() {
2.     $('.error').hide();
3.     $(".button").click(function() {
4.         // validate and process form here
5.
6.         $('.error').hide();
7.         var name = $("input#name").val();
8.         if (name == "") {
9.             $("label#name_error").show();
10.            $("input#name").focus();
11.            return false;
12.        }
13.        var email = $("input#email").val();
14.        if (email == "") {
15.            $("label#email_error").show();
16.            $("input#email").focus();
17.            return false;
18.        }
19.        var phone = $("input#phone").val();
20.        if (phone == "") {
21.            $("label#phone_error").show();
22.            $("input#phone").focus();
23.            return false;
24.        }
25.
26.    });
27. });
```

Step 3 - Write some form validation

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).

Step 3 - Write some form validation

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:

```
1. 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":

```
1. if (name == "") {  
2.     $("label#name_error").show();  
3. }
```

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

```
1. if (name == "") {  
2.     $("label#name_error").show();  
3.     $("input#name").focus();  
4.     return false;  
5. }
```

Step 3 - Write some form validation

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 with jQuery's ajax function

Now we get to the heart of the tutorial -- submitting our form without page refresh, which sends the form values to a php 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):

```
1. var dataString = 'name='+ name + '&email=' + email + '&phone=' + phone;
2. //alert (dataString);return false;
3. $.ajax({
4.   type: "POST",
5.   url: "bin/process.php",
6.   data: dataString,
7.   success: function() {
8.     $('#contact_form').html("<div id='message'></div>");
9.     $('#message').html("<h2>Contact Form Submitted!</h2>")
10.    .append("<p>We will be in touch soon.</p>")
11.    .hide()
12.    .fadeIn(1500, function() {
13.      $('#message').append("<img id='checkmark' src='images/check.png' />");
14.    });
15.  }
16. });
17. return false;
```

Step 4 - Process our form submission with jQuery's ajax function

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:

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

Step 4 - Process our form submission with jQuery's ajax function

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;
```

I've commented out an alert that I sometimes use to be sure I am grabbing the right values, which you may find helpful in the process. If you uncomment that alert and test your form, assuming everything has gone right so far, you should get a message containing all data values in a query string

Step 4 - Process our form submission with jQuery's ajax function

Now we get to our main ajax function, the star of today's show. This is where all the action happens, so pay close attention!

```
1. $.ajax({
2.   type: "POST",
3.   url: "bin/process.php",
4.   data: dataString,
5.   success: function() {
6.     //display message back to user here
7.   }
8. });
9. return false;
```

Basically what's going on in the code is this:

The `.ajax()` function processes the values from our string called `dataString` (`data:dataString`) with a php script called `process.php` (`url:"bin/process.php"`), using the 'POST' method (`type:"POST"`).

If our script processed successfully, we can then display a message back to the user, and finally return `false` so the page does not reload.

That's it! The entire process is handled right there in these few lines!

Step 4 - Process our form submission with jQuery's ajax function

There are more advanced things you can do here, other than sending an email and giving a success message. For example you could send your values to a database, process them, then display the results back to the user. So if you posted a poll to users, you could process their vote, then return the voting results, all without any page refresh required.

Step 4 - Process our form submission with jQuery's ajax function

Let's summarize what happened in our example, to be sure we have covered everything. We grabbed our form values with jQuery, and then placed those into a string like this:

```
1. var name = $("input#name").val();  
2. var email = $("input#email").val();  
3. var phone = $("input#phone").val();  
4. var dataString = 'name='+ name + '&email=' + email + '&phone=' + phone;
```

Step 4 - Process our form submission with jQuery's ajax function

Then we used jQuery's ajax function to process the values in the dataString. After that process finishes successfully, we display a message back to the user and add return false so that our page does not refresh:

```
1. $.ajax({
2.   type: "POST",
3.   url: "bin/process.php",
4.   data: dataString,
5.   success: function() {
6.     $('#contact_form').html("<div id='message'></div>");
7.     $('#message').html("<h2>Contact Form Submitted!</h2>")
8.     .append("<p>We will be in touch soon.</p>")
9.     .hide()
10.    .fadeIn(1500, function() {
11.      $('#message').append("<img id='checkmark' src='images/check.png' />");
12.    });
13.  }
14. });
15. return false;
```

Step 4 - Process our form submission with jQuery's ajax function

Then we used jQuery's ajax function to process the values in the dataString. After that process finishes successfully, we display a message back to the user and add return false so that our page does not refresh:

```
1. $.ajax({
2.   type: "POST",
3.   url: "bin/process.php",
4.   data: dataString,
5.   success: function() {
6.     $('#contact_form').html("<div id='message'></div>");
7.     $('#message').html("<h2>Contact Form Submitted!</h2>")
8.     .append("<p>We will be in touch soon.</p>")
9.     .hide()
10.    .fadeIn(1500, function() {
11.      $('#message').append("<img id='checkmark' src='images/check.png' />");
12.    });
13.  }
14. });
15. return false;
```

Step 4 - Process our form submission with jQuery's ajax function

The success part of the script has been filled in with some specific content that can be displayed back to the user.

But as far as our ajax functionality goes, that's all there is to it.

For more options and settings be sure to check out [jQuery's documentation on the ajax function.](#)

The example here is one of the simpler implementations, but even so, it is very powerful as you can see.

Step 5 - Display a message back to the user

Let's briefly look at the part of the code that displays our message back to the user, to finish out the tutorial.

First, we change the entire contents of the contact_form div (remember I said we would be needing that div) with the following line:

```
1. $('#contact_form').html("<div id='message'></div>");
```

Step 5 - Display a message back to the user

What that has done is replaced all the content inside the contact_form div, using jQuery's html() function.

So instead of a form, we now just have a new div inside, with id of 'message'. Next, we fill that div with an actual message -- an h2 saying "Contact Form Submitted":

```
$('#message').html("<h2>Contact Form Submitted!</h2>")
```

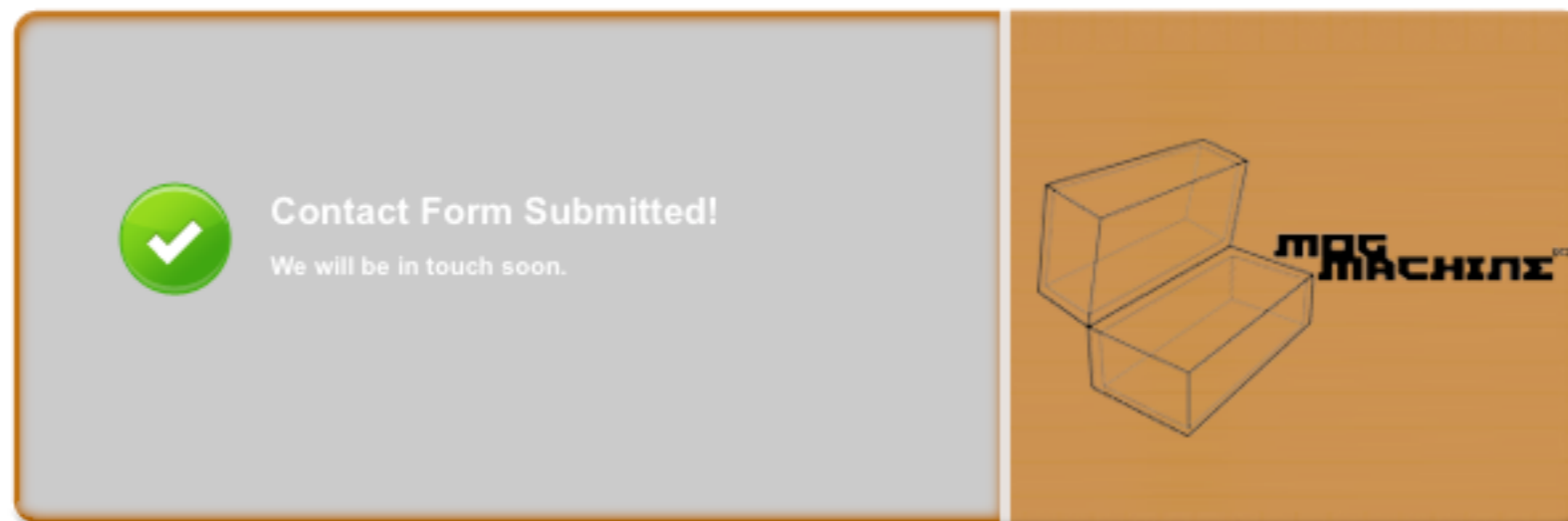
Step 5 - Display a message back to the user

We next add even more content to the message div with jQuery's `append()` function, and to top everything off we add a cool effect by hiding the message div with the jQuery `hide()` function, then fade it all in with the `fadeIn()` function:

```
1. .append("<p>We will be in touch soon.</p>")
2. .hide()
3. .fadeIn(1500, function() {
4.     $('#message').append("<img id='checkmark' src='images/check.png' />");
5. });
```

Step 5 - Display a message back to the user

So the user ends up seeing the following after they submit the form:



Send AJAX Form With JQuery

By now, I think you will have to agree that it is incredibly easy to submit forms without page refresh using jQuery's powerful ajax function. Just get the values in your javascript file, process them with the ajax function and return false, and you are good to go. You can process the values in your php script just like you would any other php file, the only difference being that the user does not have to wait for a page refresh - it all happens silently in the background.

So if you have a contact form on your website, a login form, or even more advanced forms that process values through a database and retrieve results back, you can do it all easily and efficiently, and perhaps most importantly, you enhance the end user experience with your website by providing interactivity without their having to wait for the page to reload.