**Submitting AJAX Forms with jQuery**

[Chris Sevilleja](http://scotch.io/author/chris)  January 8, 2014  ·[ajax](http://scotch.io/tag/ajax), [forms](http://scotch.io/tag/forms), [javascript](http://scotch.io/tag/javascript-2), [jquery](http://scotch.io/tag/jquery-2)  [9 Comments](http://scotch.io/tutorials/javascript/submitting-ajax-forms-with-jquery#disqus_thread)

[View Code](https://github.com/scotch-io/ajax-forms-jquery)

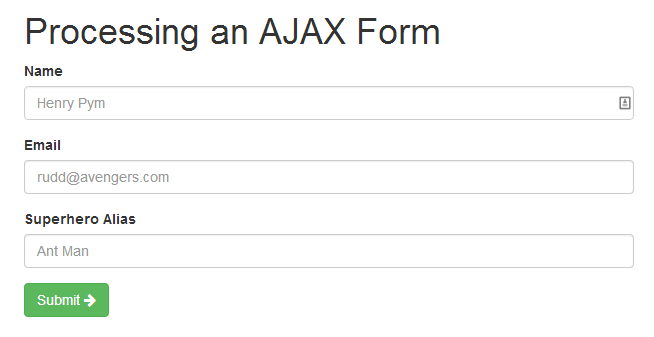
Lately I’ve noticed from looking through our Google Analytics and our search popularity, we have been coming up a lot for **AJAX Forms**. Now we have this other article, [Submitting AJAX Forms: The AngularJS Way](http://scotch.io/tutorials/javascript/submitting-ajax-forms-the-angularjs-way). That seems to be why people find us when searching for AJAX forms, but the Angular way just isn’t the way they want.

Today we’ll be looking at AJAX forms, the non-Angular way and using jQUery. This is to hopefully help those people coming to our site and looking for something other than the Angular version.

What We’ll Build

There are a few basic requirements usually asked from forms. We want:

* Process a form without a page refresh using AJAX
* Client side validation
* Server side validation (validation from the PHP side)
* Showing a processing icon while our form is… processing
* Showing errors if there are any
* Showing success if the form is submitted correctly

[](http://scotch.io/wp-content/uploads/2014/01/processing-ajax-forms-with-jquery.png)

[**Getting Started**](http://scotch.io/tutorials/javascript/submitting-ajax-forms-with-jquery#getting-started)

Let’s get started processing our form. We will need three different things for this:

* A way to process the form (we’ll be using PHP, but you can use anything you want)
* A view to show the form
* The JavaScript and AJAX (jQuery flavored) to make it all work

[**Processing the Form**process.php](http://scotch.io/tutorials/javascript/submitting-ajax-forms-with-jquery#processing-the-form-process.php)

This will be the easiest part. Since we’re using PHP, we can just create a simple script to process our form. Let’s call it process.php (clever I know), and make the code for that.

We will provide support to send back JSON data (which is what our JavaScript code will be processing after our PHP or any server side code sends back information). We will also send back errors and error messages in the data that we send back.

We will just check to see if a variable exists or not. There are many more validations that you can do like checking to see if there is a [valid email](http://en.kioskea.net/faq/1254-php-checking-the-format-of-an-email-address), but we’ll just focus on the required status.

// process.php

<?php

$errors = array(); // array to hold validation errors

$data = array(); // array to pass back data

// validate the variables ======================================================

// if any of these variables don't exist, add an error to our $errors array

if (empty($\_POST['name']))

$errors['name'] = 'Name is required.';

if (empty($\_POST['email']))

$errors['email'] = 'Email is required.';

if (empty($\_POST['superheroAlias']))

$errors['superheroAlias'] = 'Superhero alias is required.';

// return a response ===========================================================

// if there are any errors in our errors array, return a success boolean of false

if ( ! empty($errors)) {

// if there are items in our errors array, return those errors

$data['success'] = false;

$data['errors'] = $errors;

} else {

// if there are no errors process our form, then return a message

// DO ALL YOUR FORM PROCESSING HERE

// THIS CAN BE WHATEVER YOU WANT TO DO (LOGIN, SAVE, UPDATE, WHATEVER)

// show a message of success and provide a true success variable

$data['success'] = true;

$data['message'] = 'Success!';

}

// return all our data to an AJAX call

echo json\_encode($data);

Notice in PHP, to pass data back to our JavaScript call, you want toecho json\_encode(). Using a return like you normally would in a function wouldn’t provide JSON data back.

Now that we have the form processing worked out, we’ll create our form.

[**Showing the Form**index.html](http://scotch.io/tutorials/javascript/submitting-ajax-forms-with-jquery#showing-the-form-index.html)

We will be using [Bootstrap](http://getbootstrap.com/) to build out our views. It’s just easier since all their classes are already pre-built and we can create our view quickly.

We will place all of the code we need for our view and inject errors into the view later.

<!-- index.html -->

<!doctype html>

<html>

<head>

<title>Look I'm AJAXing!</title>

<link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css"> <!-- load bootstrap via CDN -->

<script src="//ajax.googleapis.com/ajax/libs/jquery/2.0.3/jquery.min.js"></script> <!-- load jquery via CDN -->

<script src="magic.js"></script> <!-- load our javascript file -->

</head>

<body>

<div class="col-sm-6 col-sm-offset-3">

<h1>Processing an AJAX Form</h1>

<!-- OUR FORM -->

<form action="process.php" method="POST">

<!-- NAME -->

<div id="name-group" class="form-group">

<label for="name">Name</label>

<input type="text" class="form-control" name="name" placeholder="Henry Pym">

<!-- errors will go here -->

</div>

<!-- EMAIL -->

<div id="email-group" class="form-group">

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

<input type="text" class="form-control" name="email" placeholder="rudd@avengers.com">

<!-- errors will go here -->

</div>

<!-- SUPERHERO ALIAS -->

<div id="superhero-group" class="form-group">

<label for="superheroAlias">Superhero Alias</label>

<input type="text" class="form-control" name="superheroAlias" placeholder="Ant Man">

<!-- errors will go here -->

</div>

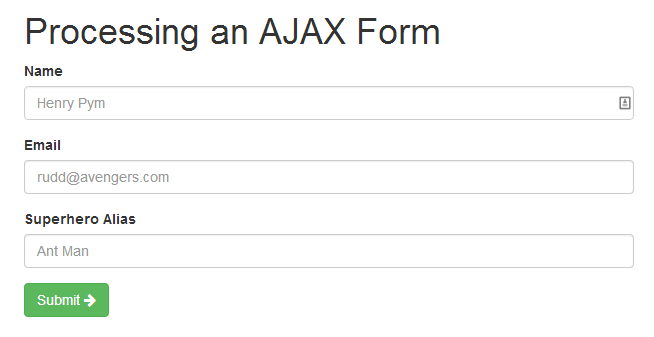
<button type="submit" class="btn btn-success">Submit <span class="fa fa-arrow-right"></span></button>

</form>

</div>

</body>

</html>

[](http://scotch.io/wp-content/uploads/2014/01/processing-ajax-forms-with-jquery.png)

[**Submitting the Form using AJAX**magic.js](http://scotch.io/tutorials/javascript/submitting-ajax-forms-with-jquery#submitting-the-form-using-ajax-magic.js)

To submit a form via AJAX, we will need to do certain things in our JavaScript file.

* Capture the form submit button so that the default action doesn’t take place
* Get all of the data from our form using jQuery
* Submit using AJAX (we’ll go through a few ways)
* Show errors if there are any

We’ll start our JavaScript file by getting data from our form and sending a POSTrequest to our already create PHP script (process.php).

// magic.js

$(document).ready(function() {

// process the form

$('form').submit(function(event) {

// get the form data

// there are many ways to get this data using jQuery (you can use the class or id also)

var formData = {

'name' : $('input[name=name]').val(),

'email' : $('input[name=email]').val(),

'superheroAlias' : $('input[name=superheroAlias]').val()

};

// process the form

$.ajax({

type : 'POST', // define the type of HTTP verb we want to use (POST for our form)

url : 'process.php', // the url where we want to POST

data : formData, // our data object

dataType : 'json', // what type of data do we expect back from the server

encode : true

})

// using the done promise callback

.done(function(data) {

// log data to the console so we can see

console.log(data);

// here we will handle errors and validation messages

});

// stop the form from submitting the normal way and refreshing the page

event.preventDefault();

});

});

Now when we press submit on our form, our JavaScript code will get all the**input values** that we need and send a POST request to process.php.

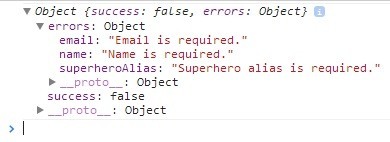
**Promise Callbacks**: We will be using the .done callback to handle a successful AJAX request. This used to be called .success but that has since been deprecated in jQuery 1.8+.

Another Way to Get Form Data serialize

You could also use [serialize](http://api.jquery.com/serialize/) instead of pulling the form information individually.

Seeing JSON Data Come Back

Our PHP script will process the inputs that our AJAX call sent and spit back the$data[] array that we created. You can see this in your browser’s console after you submit your form.

[](http://scotch.io/wp-content/uploads/2014/01/ajax-form-console-error.jpg)

You can access any of these attributes as part of your data object. Just use the. to get to a specific item.

Handling Errors

In our PHP script, we are only checking to make sure that all the fields are required. If a field is not present, we send an error back. We’ll want to inform our user of this error.

We’re going to add errors under each of the inputs. The Bootstrap classes help us do this so let’s go ahead and handle those errors and inject them into our page if they happen.

// magic.js

...

// process the form

$.ajax({

type : 'POST', // define the type of HTTP verb we want to use (POST for our form)

url : 'process.php', // the url where we want to POST

data : formData, // our data object

dataType : 'json' // what type of data do we expect back from the server

})

// using the done promise callback

.done(function(data) {

// log data to the console so we can see

console.log(data);

// here we will handle errors and validation messages

if ( ! data.success) {

// handle errors for name ---------------

if (data.errors.name) {

$('#name-group').addClass('has-error'); // add the error class to show red input

$('#name-group').append('<div class="help-block">' + data.errors.name + '</div>'); // add the actual error message under our input

}

// handle errors for email ---------------

if (data.errors.email) {

$('#email-group').addClass('has-error'); // add the error class to show red input

$('#email-group').append('<div class="help-block">' + data.errors.email + '</div>'); // add the actual error message under our input

}

// handle errors for superhero alias ---------------

if (data.errors.superheroAlias) {

$('#superhero-group').addClass('has-error'); // add the error class to show red input

$('#superhero-group').append('<div class="help-block">' + data.errors.superheroAlias + '</div>'); // add the actual error message under our input

}

} else {

// ALL GOOD! just show the success message!

$('form').append('<div class="alert alert-success">' + data.message + '</div>');

// usually after form submission, you'll want to redirect

// window.location = '/thank-you'; // redirect a user to another page

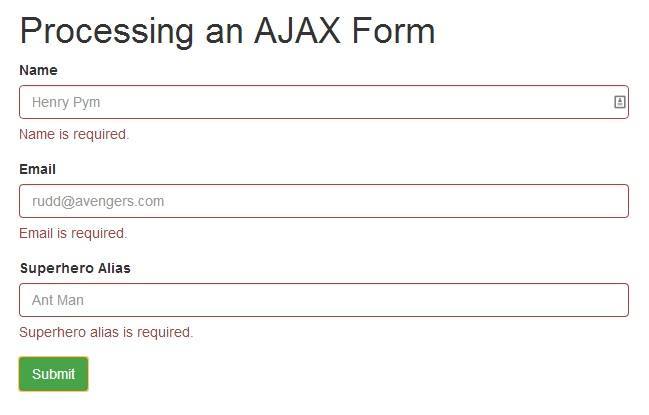
alert('success'); // for now we'll just alert the user

}

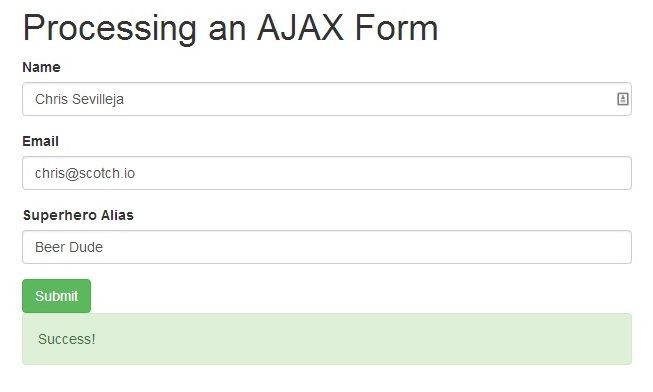
});

...

Now if there are any errors that come back from our server, we can show them like so:

[](http://scotch.io/wp-content/uploads/2014/01/ajax-form-error.jpg)

And if our AJAX form was successful:

[](http://scotch.io/wp-content/uploads/2014/01/ajax-form-success1.jpg)

Clearing Errors

Every time we submit the form, our errors from our previous submission are still there. We just need to clear them by removing them as soon as the form is submitted again.

// magic.js

...

// process the form

$('form').submit(function(event) {

$('.form-group').removeClass('has-error'); // remove the error class

$('.help-block').remove(); // remove the error text

...

[**Showing Server Errors**](http://scotch.io/tutorials/javascript/submitting-ajax-forms-with-jquery#showing-server-errors)

If there is an error in our server-side code, we won’t get any feedback from our AJAX call. We’ll just be left there scratching our heads. There’s a simple solution to that: we’ll add an errorattribute to our AJAX call.

// magic.js

...

// process the form

$.ajax({

type : 'POST', // define the type of HTTP verb we want to use (POST for our form)

url : 'process.php', // the url where we want to POST

data : formData, // our data object

dataType : 'json' // what type of data do we expect back from the server

})

// using the done promise callback

.done(function(data) {

...

})

// using the fail promise callback

.fail(function(data) {

// show any errors

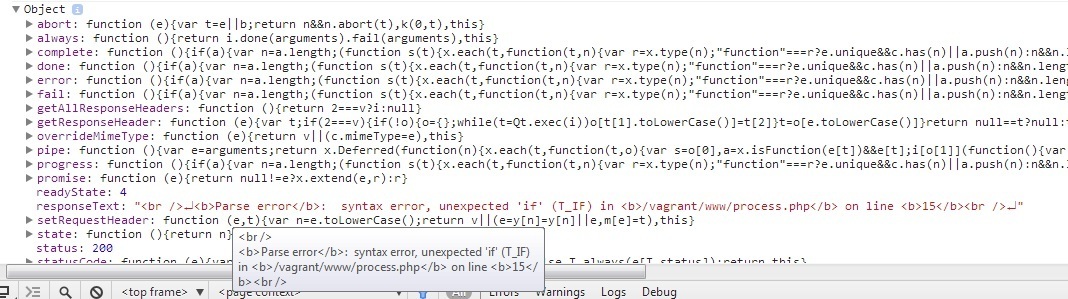
// best to remove for production

console.log(data);

});

...

Now if there are any errors in our PHP code (maybe a missing semicolon), they will show. You’ll probably want to not show errors when you move your application live though so remember to handle that accordingly.

[](http://scotch.io/wp-content/uploads/2014/01/ajax-form-jquery-fail.jpg)

[**Different Ways To Do AJAX Calls**](http://scotch.io/tutorials/javascript/submitting-ajax-forms-with-jquery#different-ways-to-do-ajax-calls)

If you don’t want to write all that AJAX code out every time, there’s an easier way! You can use the [jQuery POST Shorthand](http://api.jquery.com/jQuery.post/) method. This let’s you simplify your code and still get the same results.

The above code would be the equivalent of:

$.post('process.php', function(formData) {

// place success code here

})

.fail(function(data) {

// place error code here

});

Much cleaner!

[**Conclusion**](http://scotch.io/tutorials/javascript/submitting-ajax-forms-with-jquery#conclusion)

Submitting forms using AJAX is a fairly easy process and hopefully this helped you on your journey in creating forms and processing them without a page refresh.

This article only covers the server-side validation part of forms. You’ll want to look at doing client-side validation as well and doing more than just required validation. There are plenty of resources out there to push forward and as always, ask questions in the comments if you need any help!

**Moving Forward**: If you want to see another cool way to process an AJAX form, check out our other article that uses [AngularJS](http://scotch.io/tutorials/javascript/submitting-ajax-forms-the-angularjs-way) to submit forms.

[View Code](https://github.com/scotch-io/ajax-forms-jquery)