jQuery Validator Plugin

Advanced Options

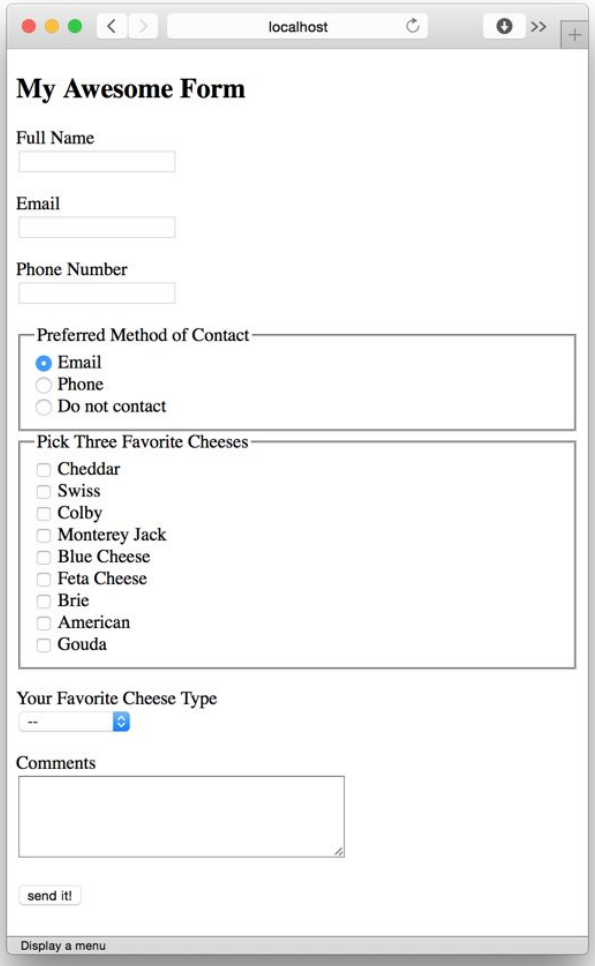
Introduction

For this exercise, you have a more complex form. It has some text fields, some radio buttons, some checkboxes, a select list and a textarea.

Try a few of the jQuery Validator plugin's more advanced features.

The processor.php file has also been updated to get this data.

Remember to put this project on a server if you want to see the actual form get processed.



Form Markup

The HTML on that makes up the form is fairly simple and does not contain much extra markup or attributes.

```
<div class="form-section">
    <p><label for="name">Full Name</label><br>
    <input type="text" id="name" name="name"></p>

    <p><label for="email">Email</label><br>
    <input type="text" id="email" name="email"></p>

    <p><label for="phone">Phone Number</label><br>
    <input type="text" id="phone" name="phone"></p>

    <fieldset>
        <legend>Preferred Method of Contact</legend>
        <input type="radio" name="contacttype" value="email" id="emailcontact" checked>
        <label for="emailcontact">Email</label><br>
        <input type="radio" name="contacttype" value="phone" id="phonecontact">
        <label for="phonecontact">Phone</label><br>
        <input type="radio" name="contacttype" value="none" id="nocontact">
        <label for="nocontact">Do not contact</label><br>
    </fieldset>
</div>
```

Dealing with Checkboxes

The only thing that might seem a little unusual is the square brackets on the checkboxes. Because this set of elements all have the same name, and multiple checkboxes can be checked (unlike radio buttons), they go through to the processor.php file as an array.

If you look at the top of the processor.php file, $cheeses is actually an array with zero or more elements in it.

```
<fieldset>
    <legend>Pick Three Favorite Cheeses</legend>

    <input type="checkbox" id="cheddar" name="cheese[]" value="Cheddar">
    <label for="cheddar">Cheddar</label><br>

    <input type="checkbox" id="swiss" name="cheese[]" value="Swiss">
    <label for="swiss">Swiss</label><br>

    <input type="checkbox" id="colby" name="cheese[]" value="Colby">
    <label for="colby">Colby</label><br>
```

```
$contacttype = $_POST['contacttype'];
$cheeses = $_POST['cheese'];
$cheesetype = $_POST['cheesetype'];
$comments = $_POST['comments'];
```

Loop for Getting the Checkbox Data

This PHP code loops through the array and puts the items into a simple comma separated list. The last bit just removes the final comma and space from the end of the list.

Let's get to the validation...

```php
// for handling the array
$cheese_list ="";

foreach($cheeses as $eachcheese)
{
    $cheese_list .= $eachcheese . ", ";
}

//cuts off the final ', '
$cheese_list = substr($cheese_list, 0, -2);
```

Starting the Script

As before, use the validate function built into the validator plugin. But this time you will add some curly braces so that you can pass in some options.

The syntax for adding options is to add an object. The name of option group, followed by a colon, followed by curly braces. Inside the curly braces, you have an option in that group.

For email, you want two options. It is both required and you want to test that it is an email address. Notice the comma on the end of the name line.

```
<script type="text/javascript">

    $("#myform").validate( {} );

</script>
```

```
$("#myform").validate( {

    rules: {

        name: "required"

    }

} );
```

```
rules: {

    name: "required",
    email: {required: true, email: true}

}
```

Checking the Checkboxes

The cheese field is the checkboxes, which is an array, so you have to include the square braces and put it in quotes. Notice that you can make it required as well as say how many checkboxes are required by setting a minlength and maxlength for the array.

```
rules: {

    name: "required",
    email: {required: true, email: true},
    "cheese[]": {required: true, minlength: 3, maxlength: 3}

}
```

Checking the Phone Number

Checking the phone number is a bit more challenging. The validator script does not have a regular expression for phone numbers built in, but you can add one.

Put this code above the validator function.

This code is in a code snippet on the assignment page, so you can copy and paste it.

```
<script type="text/javascript">

    $.validator.addMethod("phoneRegex", function(value, element) {
        return this.optional(element) || /^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$/i.test(value);
    }, "Please type a valid U.S. Phone Number.");

    $("#myform").validate( {
```

Adding To the Rules

Now when you add the validator for the phone number, you can use the phoneRegex regular expression you added above.

Finally, cheesetype is also required. Notice, no comma at the end of that line.

```
rules: {

    name: "required",
    email: {required: true, email: true},
    "cheese[]": {required: true, minlength: 3, maxlength: 3},
    phone: {required: true, phoneRegex: true}

}
```

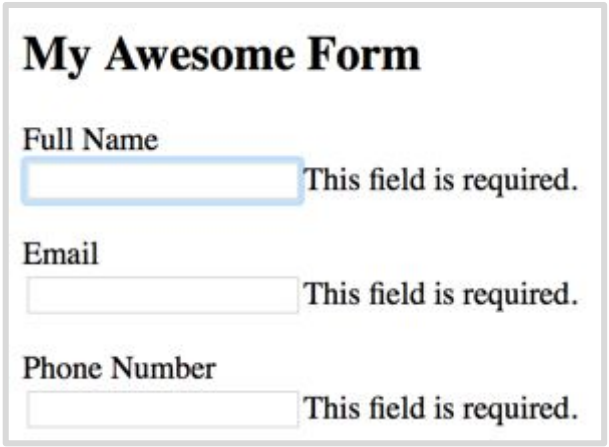
```
rules: {

    name: "required",
    email: {required: true, email: true},
    "cheese[]": {required: true, minlength: 3, maxlength: 3},
    phone: {required: true, phoneRegex: true},
    cheesetype: "required"

}
```

Testing the Validation So Far

If you test it, it should work, but we are getting generic error messages.

An added benefit of using the plugin this way is that your markup is completely clean.

You did not have to add “required” or set the type attributes for the different fields.



Adding Custom Messages

To set custom messages, add a group of options for messages.

Don’t forget the comma at the end of rules!

```
$("#myform").validate( {

    rules: {

        name: "required",
        email: {required: true, email: true},
        "cheese[]": {required: true, minlength: 3, maxlength: 3},
        phone: {required: true, phoneRegex: true},
        cheesetype: "required"

    },

    messages: {

    }

} );
```

Adding the Messages

The syntax here is the same as in the rules section.

For email, you need to provide two messages. One for if it is empty, and one for if it is not an email address.

The array is again dealt with slightly differently.

```
messages: {

    name: "Dude, gimme your name.",
    email: {required: "Email is required", email: "Valid email address please"},
    "cheese[]": "You must choose three cheeses"
}
```

Error Message for Phone

Phone is also a little different, because we defined the message to get when we added the record, it will use that message. If you want to change it, simply change it above.

Inside the messages options, we just leave an empty string.

```
$.validator.addMethod("phoneRegex", function(value, element) {
    return this.optional(element) || /^\(?[\d]{3}\)?[\s-]?[\d]{3
}, "Please type a valid U.S. Phone Number.");
```

```
messages: {

    name: "Dude, gimme your name.",
    email: {required: "Email is required", email: "Valid email address please"},
    "cheese[]": "You must choose three cheeses",
    phone: {required: "Phone is required", phone: ""}
}
```

The Final Message for the Select Box

The final field just gets a simple message. Notice, no comma on the end.

```
messages: {

    name: "Dude, gimme your name.",
    email: {required: "Email is required", email: "Valid email address please"},
    "cheese[]": "You must choose three cheeses",
    phone: {required: "Phone is required", phone: ""},
    cheesetype: "Type of prefered cheese is required"
}
```

Add some HTML & CSS

At this point, it is working and you could go from here to styling and getting the error message to look the way you want.

However, there is one more option I want to show you.

Add this markup to the bottom of the form and then add a rule to the stylesheet.

Notice that this messagebox div is set to display:none initially.

```
<div id="messagebox">
    <h4>Please Fix These Errors</h4>
    <ul></ul>
</div>

</form>
```

```
<style>

#messagebox {
    position:absolute;
    top:20px;
    right:20px;
    width: 250px;
    padding:15px;
    border:2px solid #666;
    display:none;
}

</style>
```

A Few More Settings in the Script

Add a comma after messages and then add these three settings. This will tell the script that we want to put the error messages inside #messagebox, and that each error message will go inside a list item, inside the messagebox unordered list.

```
messages: {

    name: "Dude, gimme your name.",
    email: {required: "Email is required", email: "Valid email address please"},
    "cheese[]": "You must choose three cheeses",
    phone: {required: "Phone is required", phone: ""},
    cheesetype: "Type of prefered cheese is required"
},

errorContainer: "#messagebox",
errorLabelContainer: "#messagebox ul",
wrapper: "li"
```

Finished Version

This will put the errors in a box on the side. Notice that as you fix the errors, the disappear from the list. When they are all gone, the error box goes back to display:none.

There are plenty of other options to explore for the jQuery Validator plugin, but this gives you a good sense of kinds of things it can do.