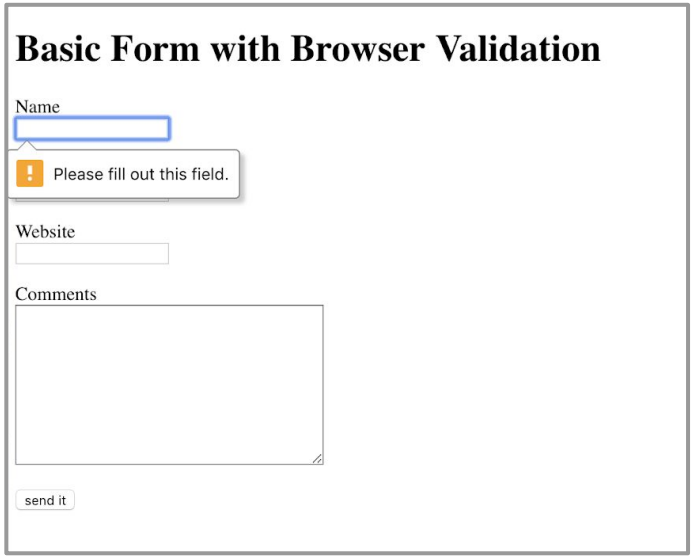
Form Validation Basics

Built in Browser Validation, JavaScript and jQuery Validation Options

Built In Browser Validation - Demo

If you go try this example, you will see that the browser does the validation. This validation is built into modern browsers, but older versions of browsers don't have this feature.

(ex0 in the example files)



Built In Browser Validation - Using Attributes in the HTML

Look at the HTML and notice the pieces added to make the browser do this validation.

The attribute, required, looks a little funny sitting out there all by itself. It is really just a shortcut for writing the attribute like this:

required = "required"

The W3C decided that was just redundant and that it can be shortened down to just required.

```
<form action="processor.php" method="post" id="myform">

    <p><label for="name">Name</label><br>
    <input id="name" name="name" type="text" required></p>

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

    <p><label for="url">Website</label><br>
    <input id="url" name="url" type="url"></p>

    <p><label for="comments">Comments</label><br>
    <textarea id="comments" name="comments" cols="40" rows="10" required></textarea></p>

    <p><input name="send" type="submit" value="send it"></p>

</form>
```

Using Input Type Values

The type attribute on the email and website fields is providing further validation for those types of data.

This is useful for such a simple form, but if you want more control over the error messages, or have a more complex form, you will need JavaScript to help validate the forms.

```
<form action="processor.php" method="post" id="myform">

    <p><label for="name">Name</label><br>
    <input id="name" name="name" type="text" required></p>

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

    <p><label for="url">Website</label><br>
    <input id="url" name="url" type="url"></p>

    <p><label for="comments">Comments</label><br>
    <textarea id="comments" name="comments" cols="40" rows="10" required></textarea></p>

    <p><input name="send" type="submit" value="send it"></p>

</form>
```

Processing the Form

To make matters worse, there is no validation on the processing end. If the data makes it past the browser, this processor will just print out what was sent without checking anything.

For a more secure interface, you need to validate on both the backend (with PHP, or some other server side technology), and on the front end with JavaScript.

(ex0 in the example files)

```
<body>

    <h1>Processed Data</h1>

    <?php

    if( isset( $_POST['comments'] ) )
    {
        $name = $_POST['name'];
        $email = $_POST['email'];
        $url = $_POST['url'];
        $comments = $_POST['comments'];
        $submit = $_POST['send'];

        echo "<p>Name: $name</p>";
        echo "<p>Email: $email</p>";
        echo "<p>URL: $url</p>";
        echo "<p>Comments: $comments</p>";
        echo "<p>$submit</p>";
    }
    ?>

</body>
```

Backend Validation with PHP

If you look at the next version of the simple form, you will see that I have removed browser validation, but added some on the PHP file.

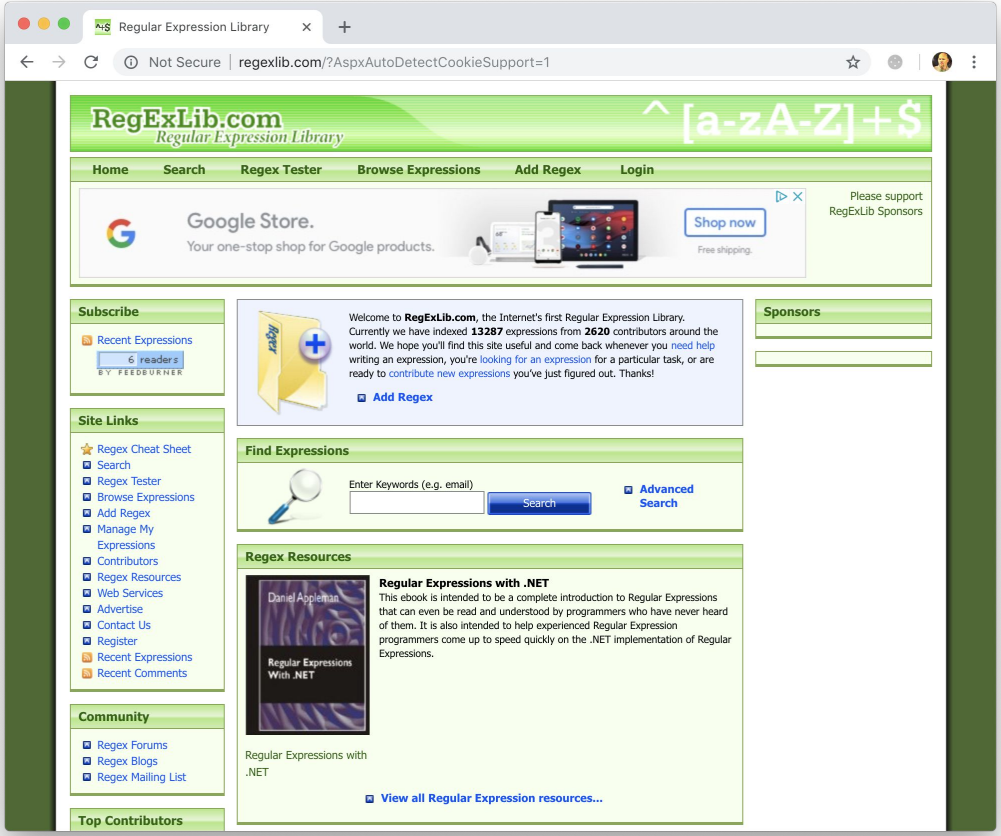
This version of the script makes sure the required fields are not empty, then uses a regular expression to make sure the email address is really an email address.

(ex1 in the example files)

```php
if( isset( $_POST['comments'] ) )
{
    $name = $_POST['name'];
    $email = $_POST['email'];
    $url = $_POST['url'];
    $comments = $_POST['comments'];
    $submit = $_POST['send'];

    if( !empty($name) && !empty($email) && !empty($comments) )
    {

        $re_email = "/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,4})+$/";

        if( preg_match($re_email, $email) )
        {

            echo "<p>Name: $name</p>";
            echo "<p>Email: $email</p>";
            echo "<p>URL: $url</p>";
            echo "<p>Comments: $comments</p>";
            echo "<p>$submit</p>";
        }
        else { echo "<p>you did not format your email properly</p>"; }
    }
    else { echo "<p>You did not fill in the required fields</p>"; }
}
```

Regular Expressions

Regular expressions allow us to check for certain patterns, like the pattern of an email address, or a web url or an american phone number. You can use the regex library to locate regular expressions that other people have written, or you can learn how to write them yourself.



Plain JavaScript Validation - HTML

If you look at this next version, which includes JavaScript validation on the front end, you will see that error messages show up if the form is not filled out properly and you try to submit it.

The only difference in the form is the additional spans to hold the error messages.

The PHP file is exactly the same.

(ex2 in the example files)

```
<h1>Basic Form with JavaScript Validation</h1>

<form action="processor.php" method="post" id="myform">

    <p><label for="name">Name</label><br>
    <input id="name" name="name" type="text"><span id="name-err"></span></p>

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

    <p><label for="url">Website</label><br>
    <input id="url" name="url" type="text"><span id="url-err"></span></p>

    <p><label for="comments">Comments</label><span id="comments-err"></span><br>
    <textarea id="comments" name="comments" cols="40" rows="10"></textarea></p>

    <p><input name="send" type="submit" value="send it"></p>

</form>
```

Plain JavaScript Validation - Function

The basic guts of this version is that there is a counter that starts at zero. If any field does not pass muster, then the counter is incremented. At the end of the script, check to see if the counter is still zero or a higher number. If it is higher, return false which keeps the form from going through to the processor.php file.

```
    document.getElementById('myform').onsubmit = validateForm;

    function validateForm()
    {
        var reName = /^[a-zA-Z]+(([\'\- ][a-zA-Z])?[a-zA-Z]*)*$/;
        var reEmail = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,4})+$/;
var reURL = /(ftp|http|ht...
        var counter=0;

        if( !reName.test(document.getElementById('name').value) )
        {
            fixForm( document.getElementById('name'), "Please provide a proper name" );
            counter++;
        }

        if( !reEmail.test(document.getElementById('email').value) )
        {
            fixForm( document.getElementById('email'), "Please type add a valid email address" );
            counter++;
        }

var url = document.getEle...
        if(counter > 0)
        {
            return false;
        }
        else
        {
            return true;
        }
    }
```

Plain JavaScript Validation - Fix Form Function

The fix form function takes a specific field and a message as arguments, and then adds the correct error message into the correct span in the form, if there is an error for that field.

```
if( !reName.test(document.getElementById('name').value) )
{
    fixForm( document.getElementById('name'), "Please provide a proper name" );
    counter++;
}
```

```
function fixForm(field, message)
{
    var errorMessage = field.id + "-err";
    document.getElementById(errorMessage).innerHTML = message;
    document.getElementById(errorMessage).style.color = "red";
    document.getElementById(field.id).focus();
}
```

jQuery Form Validation

If you look at the next version, it works pretty much the same way, but the script has been converted to using jQuery. That makes it a little bit shorter and easier to work with, but the basic strategy is the same: If the counter increments, then stop the processing of the form and highlight the incorrect fields.

(ex3 in the example files)

```
$("#myForm").submit( function( event ){
    var name = $('#name').val();
    var email = $('#email').val();
    var url = $('#url').val();
    var comments = $('#comments').val();

    $('#myForm span').html('').css('color', 'red');
    counter = 0;

    if( !re_name.test(name) )
    {
        $('#name-err').html("Please enter a valid name");
        counter++;
    }

    if( !re_email.test(email) )
    {
        $('#email-err').html("Please enter a valid email");
        counter++;
    }

    if( url !="" && !re_url.test(url) )
    {
        $('#url-err').html("Please enter a valid URL");
        counter++;
    }

    if( comments == '' )
    {
      $('#comments-err').html(" Please give me a comment!");
      counter++;
    }

    if( counter > 0){ event.preventDefault(); }

} );
```

jQuery Validator Plugin

To both simplify and supercharge your form validation, you can use the jQuery Validator plugin.

There are both simple and advanced ways it can be used that give you more control over error messages and how everything is displayed than what you can get from the built in browser validation.

Check out this version for a working example of a simple use of the plugin.

(ex4 in the example files)

Adding jQuery

```
</body>

<!-- add link to jQuery library here -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

<!-- add validator plugin here -->
<!-- Instantiate the validator plugin with your own script here -->
```

To get started with the jQuery validator, first add jQuery. A good place to get it is from the Google hosted libraries site.

Work with the ex04 example files!

Adding the Plugin

```
<!-- add link to jQuery library here -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

<!-- add validator plugin here -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.17.0/jquery.validate.min.js"></script>
```

Then add the jQuery Validator plugin. You can find the hotlink for the plugin on the jQuery Validator website. Get the minified version for quicker download on your site.

jQuery Validator Plugin - Running the Validator

```
<!-- add link to jQuery library here -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

<!-- add validator plugin here -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.17.0/jquery.validate.min.js"></script>

<!-- Instantiate the validator plugin with your own script here -->
<script>

    $("#myForm").validate();

</script>
```

Then in your own script tag, run the validator.

Adding Required Attributes

```
<p><label for="name">Name</label><br>
<input id="name" name="name" type="text" required></p>

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

<p><label for="url">Website</label><br>
<input id="url" name="url" type="url"></p>

<p><label for="comments">Comments</label><br>
<textarea id="comments" name="comments" cols="40" rows="10" required></textarea></p>

<p><input id="send"  name="send" type="submit" value="send it"></p>
```

On the form, add the required attribute to require those fields.

jQuery Validator Plugin - Testing the Validation

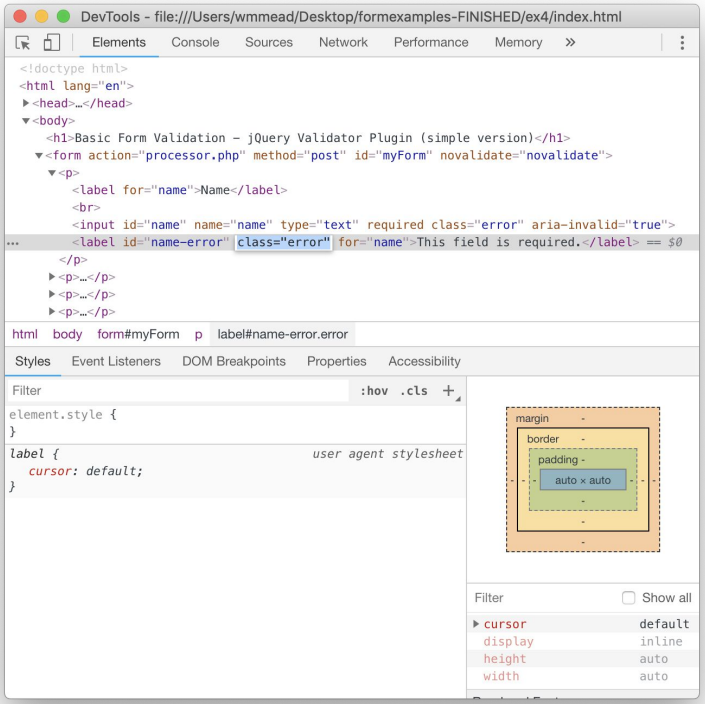
Name
This field is required.

Email
This field is required.

Website
asdas Please enter a valid URL.

Comments
This field is required.

send it



If you test it. It should be working. Notice that the validator adds a label with a class you can hook into with CSS.

jQuery Validator Plugin - Styling and Custom Error Messages

```css
label.error {
    color:#D40004;
    display:block;
    border: 2px solid #D40004;
    padding:15px;
    font-size:24px;
    text-transform:uppercase;
}
```

You can turn on the CSS at the top of the page to style the error messages.

And you can add title attributes to the required fields to specify custom error messages.

```html
<p><label for="name">Name</label><br>
<input id="name" name="name" type="text" required title="yo, give me a name!"></p>
```

Summary

In this lesson you have seen how you can perform client side form validation in a number of ways, including using the built in browser features, plain JavaScript, jQuery or a simple use of the jQuery validator plugin.

The jQuery validator plugin offers a lot more advanced options as well.