**javascript form validation**

There's nothing more troublesome than receiving orders, guestbook entries, or other form submitted data that are incomplete in some way. You can avoid these headaches once and for all with JavaScript's amazing way to combat bad form data with a technique called "form validation".

The idea behind JavaScript form validation is to provide a method to check the user entered information before they can even submit it. JavaScript also lets you display helpful alerts to inform the user what information they have entered incorrectly and how they can fix it. In this lesson we will be reviewing some basic form validation, showing you how to check for the following:

* If a text input is empty or not
* If a text input is all numbers
* If a text input is all letters
* If a text input is all alphanumeric characters (numbers & letters)
* If a text input has the correct number of characters in it (useful when restricting the length of a username and/or password)
* If a selection has been made from an HTML select input (the drop down selector)
* If an email address is valid
* How to check all above when the user has completed filling out the form

This lesson is a little long, but knowing how to implement these form validation techniques is definitely worth the effort on your part. Remember to check out Tizag's [HTML forms](http://www.tizag.com/htmlT/forms.php) lesson if you need to brush up on your form knowledge.

# form validation - checking for non-empty

This has to be the most common type of form validation. You want to be sure that your visitors enter data into the HTML fields you have "required" for a valid submission. Below is the JavaScript code to perform this basic check to see if a given HTML input is empty or not.

## JavaScript Code:

// If the length of the element's string is 0 then display helper message

function notEmpty(elem, helperMsg){

if(elem.value.length == 0){

alert(helperMsg);

elem.focus(); // set the focus to this input

return false;

}

return true;

}

The function *notEmpty* will check to see that the HTML input that we send it has something in it. *elem* is a HTML text input that we send this function. JavaScriptstrings have built in properties, one of which is the *length* property which returns the length of the string. The chunk of code *elem.value* will grab the string inside the input and by adding on length *elem.value.length* we can see how long the string is.

As long as *elem.value.length* isn't 0 then it's not empty and we return true, otherwise we send an alert to the user with a *helperMsg* to inform them of their error and return false.

## Working Example:

<script type='text/javascript'>

function notEmpty(elem, helperMsg){

if(elem.value.length == 0){

alert(helperMsg);

elem.focus();

return false;

}

return true;

}

</script>

<form>

Required Field: <input type='text' id='req1'/>

<input type='button'

onclick="notEmpty(document.getElementById('req1'), 'Please Enter a Value')"

value='Check Field' />

</form>

## Display:

Top of Form

Required Field: 

Bottom of Form

# form validation - checking for all numbers

If someone is entering a credit card, phone number, zip code, similar information you want to be able to ensure that the input is all numbers. The quickest way to check if an input's string value is all numbers is to use a regular expression /^[0-9]+$/ that will only *match* if the string is all numbers and is at least one character long.

## JavaScript Code:

// If the element's string matches the regular expression it is all numbers

function isNumeric(elem, helperMsg){

var numericExpression = /^[0-9]+$/;

if(elem.value.match(numericExpression)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

What we're doing here is using JavaScript existing framework to have it do all the hard work for us. Inside each string is a function called *match* that you can use to see if the string matches a certain regular expression. We accessed this function like so: elem.value.match(expressionhere).

We wanted to see if the input's string was all numbers so we made a regular expression to check for numbers [0-9] and stored it as *numericExpression*.

We then used the *match* function with our regular expression. If it is numeric then *match* will return true, making our if statement pass the test and our function*isNumeric* will also return true. However, if the expression fails because there is a letter or other character in our input's string then we'll display our *helperMsg* and return false.

## Working Example:

<script type='text/javascript'>

function isNumeric(elem, helperMsg){

var numericExpression = /^[0-9]+$/;

if(elem.value.match(numericExpression)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

</script>

<form>

Numbers Only: <input type='text' id='numbers'/>

<input type='button'

onclick="isNumeric(document.getElementById('numbers'), 'Numbers Only Please')"

value='Check Field' />

</form>

## Display:

Top of Form

Numbers Only: 

Bottom of Form

# form validation - checking for all letters

This function will be identical to *isNumeric* except for the change to the regular expression we use inside the *match* function. Instead of checking for numbers we will want to check for all letters.

If we wanted to see if a string contained only letters we need to specify an expression that allows for both lowercase and uppercase letters: /^[a-zA-Z]+$/ .

## JavaScript Code:

// If the element's string matches the regular expression it is all letters

function isAlphabet(elem, helperMsg){

var alphaExp = /^[a-zA-Z]+$/;

if(elem.value.match(alphaExp)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

## Working Example:

<script type='text/javascript'>

function isAlphabet(elem, helperMsg){

var alphaExp = /^[a-zA-Z]+$/;

if(elem.value.match(alphaExp)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

</script>

<form>

Letters Only: <input type='text' id='letters'/>

<input type='button'

onclick="isAlphabet(document.getElementById('letters'), 'Letters Only Please')"

value='Check Field' />

</form>

## Display:

Top of Form

Letters Only: 

Bottom of Form

# form validation - checking for numbers and letters

By combining both the *isAlphabet* and *isNumeric* functions into one we can check to see if a text input contains only letters and numbers.

## JavaScript Code:

// If the element's string matches the regular expression it is numbers and letters

function isAlphanumeric(elem, helperMsg){

var alphaExp = /^[0-9a-zA-Z]+$/;

if(elem.value.match(alphaExp)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

# form validation - restricting the length

Being able to restrict the number of characters a user can enter into a field is one of the best ways to prevent bad data. For example, if you know that the zip code field should only be 5 numbers you know that 2 numbers is not sufficient.

Below we have created a *lengthRestriction* function that takes a text field and two numbers. The first number is the minimum number of characters and the second is the maximum number of a characters the input can be. If you just want to specify an exact number then send the same number for both minimum and maximum.

## JavaScript Code:

function lengthRestriction(elem, min, max){

var uInput = elem.value;

if(uInput.length >= min && uInput.length <= max){

return true;

}else{

alert("Please enter between " +min+ " and " +max+ " characters");

elem.focus();

return false;

}

}

Here's an example of this function for a field that requires 6 to 8 characters for a valid username.

## Working Example:

<script type='text/javascript'>

function lengthRestriction(elem, min, max){

var uInput = elem.value;

if(uInput.length >= min && uInput.length <= max){

return true;

}else{

alert("Please enter between " +min+ " and " +max+ " characters");

elem.focus();

return false;

}

}

</script>

<form>

Username(6-8 characters): <input type='text' id='restrict'/>

<input type='button'

onclick="lengthRestriction(document.getElementById('restrict'), 6, 8)"

value='Check Field' />

</form>

## Display:

Top of Form

Username(6-8 characters): 

Bottom of Form

# form validation - selection made

To be sure that someone has actually selected a choice from an HTML select input you can use a simple trick of making the first option as helpful prompt to the user and a red flag to you for your validation code.

By making the first option of your select input something like "Please Choose" you can spur the user to both make a selection and allow you to check to see if the default option "Please Choose" is still selected when the submit the form.

## JavaScript Code:

function madeSelection(elem, helperMsg){

if(elem.value == "Please Choose"){

alert(helperMsg);

elem.focus();

return false;

}else{

return true;

}

}

## Working Example:

<script type='text/javascript'>

function madeSelection(elem, helperMsg){

if(elem.value == "Please Choose"){

alert(helperMsg);

elem.focus();

return false;

}else{

return true;

}

}

</script>

<form>

Selection: <select id='selection'>

<option>Please Choose</option>

<option>CA</option>

<option>WI</option>

<option>XX</option>

</select>

<input type='button'

onclick="madeSelection(document.getElementById('selection'), 'Please Choose Something')"

value='Check Field' />

</form>

## Display:

Top of Form

Selection:     

Bottom of Form

# form validation - email validation

And for our grand finale we will be showing you how to check to see if a user's email address is valid. Every email is made up for 5 parts:

1. A combination of letters, numbers, periods, hyphens, plus signs, and/or underscores
2. The at symbol @
3. A combination of letters, numbers, hyphens, and/or periods
4. A period
5. The top level domain (com, net, org, us, gov, ...)

Valid Examples:

* bobby.jo@filltank.net
* jack+jill@hill.com
* the-stand@steven.king.com

Invalid Examples:

* @deleted.net - no characters before the @
* free!dom@bravehe.art - invalid character !
* shoes@need\_shining.com - underscores are not allowed in the domain name

The regular expression to check for all of this is a little overkill and beyond the scope of this tutorial to explain thoroughly. However, test it out and you'll see that it gets the job done.

## JavaScript Code:

function emailValidator(elem, helperMsg){

var emailExp = /^[\w\-\.\+]+\@[a-zA-Z0-9\.\-]+\.[a-zA-z0-9]{2,4}$/;

if(elem.value.match(emailExp)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

## Working Example:

<script type='text/javascript'>

function emailValidator(elem, helperMsg){

var emailExp = /^[\w\-\.\+]+\@[a-zA-Z0-9\.\-]+\.[a-zA-z0-9]{2,4}$/;

if(elem.value.match(emailExp)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

</script>

<form>

Email: <input type='text' id='emailer'/>

<input type='button'

onclick="emailValidator1(document.getElementById('emailer'), 'Not a Valid Email')"

value='Check Field' />

</form>

## Display:

Top of Form

Email: 

Bottom of Form

# validating a form - all at once

If you've made it this far I commend you, but we're not done yet! The final step is to be able to perform all of these validation steps when the user is ready to submit their data.

Each form has a JavaScript event called *onSubmit* that is triggered when its*submit* button is clicked. If this even returns 0 or false then a form cannot be submitted, and if it returns 1 or true it will always be submitted. Wouldn't it be perfect if we could somehow make an if statement that said "If the form is valid submit it (1) else don't submit it (0)"? Well with a master *formValidator* function we can do just that.

*formValidator* will be somewhat like a list of checks that we want to do before a form is submitted. But before we can decide what we want to check for, we need to have our form!

## HTML Form Code:

<form onsubmit='return formValidator()' >

First Name: <input type='text' id='firstname' /><br />

Address: <input type='text' id='addr' /><br />

Zip Code: <input type='text' id='zip' /><br />

State: <select id='state'>

<option>Please Choose</option>

<option>AL</option>

<option>CA</option>

<option>TX</option>

<option>WI</option>

</select><br />

Username(6-8 characters): <input type='text' id='username' /><br />

Email: <input type='text' id='email' /><br />

<input type='submit' value='Check Form' /><br />

</form>

That's a lot of data to verify and the first thing we would probably want to check is that each field was at least filled out. To check for completion we will ensure no fields are empty and that the *SELECT* field has a selection. Here are the starting pieces of our master validation function *formValidator*.

## JavaScript Code:

function formValidator(){

// Make quick references to our fields

var firstname = document.getElementById('firstname');

var addr = document.getElementById('addr');

var zip = document.getElementById('zip');

var state = document.getElementById('state');

var username = document.getElementById('username');

var email = document.getElementById('email');

// Check each input in the order that it appears in the form!

if(isAlphabet(firstname, "Please enter only letters for your name")){

if(isAlphanumeric(addr, "Numbers and Letters Only for Address")){

if(isNumeric(zip, "Please enter a valid zip code")){

if(madeSelection(state, "Please Choose a State")){

if(lengthRestriction(username, 6, 8)){

if(emailValidator(email, "Please enter a valid email address")){

return true;

}

}

}

}

}

}

return false;

}

The first part of this function is where we create easy references to our HTML inputs using the *getElementById* function. These quick references will make our next block of code much easier to read!

The second part uses a bunch of embedded if statements to see whether or not each field has the correct type of data. If every single one of those fields we check validates, then we'll return true and the form will be submitted successfully.

However, if just one of those if statements fails then the *return false* at the end of the function is reached and prevents the form for being submitted.

As you can see this function really does do quite a lot, definitely earning the title of *formValidator*. Notice how this one function references all of the functions we have covered in this lesson. By placing all of these checks in a central location you make your code easier to read and easier to change around in the future.

Now let's put all the necessary and HTML together and try it out!

# all together now

Below we have taken the HTML form code and the new function *formValidator*and plugged in all the other form validation functions taught in this lesson that are referenced in *formValidator*.

## HTML & JavaScript Code:

<script type='text/javascript'>

function formValidator(){

// Make quick references to our fields

var firstname = document.getElementById('firstname');

var addr = document.getElementById('addr');

var zip = document.getElementById('zip');

var state = document.getElementById('state');

var username = document.getElementById('username');

var email = document.getElementById('email');

// Check each input in the order that it appears in the form!

if(isAlphabet(firstname, "Please enter only letters for your name")){

if(isAlphanumeric(addr, "Numbers and Letters Only for Address")){

if(isNumeric(zip, "Please enter a valid zip code")){

if(madeSelection(state, "Please Choose a State")){

if(lengthRestriction(username, 6, 8)){

if(emailValidator(email, "Please enter a valid email address")){

return true;

}

}

}

}

}

}

return false;

}

function notEmpty(elem, helperMsg){

if(elem.value.length == 0){

alert(helperMsg);

elem.focus(); // set the focus to this input

return false;

}

return true;

}

function isNumeric(elem, helperMsg){

var numericExpression = /^[0-9]+$/;

if(elem.value.match(numericExpression)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

function isAlphabet(elem, helperMsg){

var alphaExp = /^[a-zA-Z]+$/;

if(elem.value.match(alphaExp)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

function isAlphanumeric(elem, helperMsg){

var alphaExp = /^[0-9a-zA-Z]+$/;

if(elem.value.match(alphaExp)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

function lengthRestriction(elem, min, max){

var uInput = elem.value;

if(uInput.length >= min && uInput.length <= max){

return true;

}else{

alert("Please enter between " +min+ " and " +max+ " characters");

elem.focus();

return false;

}

}

function madeSelection(elem, helperMsg){

if(elem.value == "Please Choose"){

alert(helperMsg);

elem.focus();

return false;

}else{

return true;

}

}

function emailValidator(elem, helperMsg){

var emailExp = /^[\w\-\.\+]+\@[a-zA-Z0-9\.\-]+\.[a-zA-z0-9]{2,4}$/;

if(elem.value.match(emailExp)){

return true;

}else{

alert(helperMsg);

elem.focus();

return false;

}

}

</script>

<form onsubmit='return formValidator()' >

First Name: <input type='text' id='firstname' /><br />

Address: <input type='text' id='addr' /><br />

Zip Code: <input type='text' id='zip' /><br />

State: <select id='state'>

<option>Please Choose</option>

<option>AL</option>

<option>CA</option>

<option>TX</option>

<option>WI</option>

</select><br />

Username(6-8 characters): <input type='text' id='username' /><br />

Email: <input type='text' id='email' /><br />

<input type='submit' value='Check Form' />

</form>

## Display:

Top of Form

First Name:   
Address:   
Zip Code:   
State:            
Username(6-8 characters):   
Email:   


Bottom of Form