

# Client side validation with forms using JavaScript

When creating forms to gather visitor feedback from your site, it's important you receive the information that is required before allowing the submission to complete. In most cases with web forms, there is usually one field that is required. When the visitor presses the submit button, if this field is empty developers should implement functionality to alert the user for the missing information.

There are two ways to validate form data:

- Client side
- Server side

Client side validation is done by JavaScript, which is executed inside a visitor's browser. Server side validation is performed by server-side scripts which can be ASP, ASP.NET, PHP, or Cold fusion. In this article, we will examine how to validate form fields using JavaScript.

## See Finished Page

<http://midwestwebdesign.net/tutorials/clientvalidation/form.html>

## Download Project Files

<http://midwestwebdeisgn.net/tutorials/clientvalidation/clientvalidation.zip>

## Before we begin...

Web developers **SHOULD NOT RELY** solely on JavaScript for their validation. Our reason is simple - if the visitor has JavaScript turned off (disabled) in their browser, validation will not work. Furthermore, relying on JavaScript for validation opens your site to hackers who can easily exploit your code, which in turns can corrupt the data you receive, which in turn is often used to insert results into a database table.

When working with forms, it's recommended doing client and server side validation. By performing server side validation a hacker could not (at least not easily) exploit your server-side script, since it's parsed before resulting output (normal HTML) is sent to the browser. One additional note, we will not rely on JavaScript to ensure actually getting the information we want such as numbers for a social security field. We will only ensure a visitor has entered something into the form field. We let server-side scripts check for correct input, using regular expressions. Please research regular expression via Google for more information.

## Open the file

We will avoid discussing in-depth how the form is created, and instead opt to examine validation techniques with JavaScript. You may open the file in your text editor or web browser, kick the tires, take it for a test drive and if you have questions, feel free to ask me using the link at the bottom of this article. One important piece of code in this file is shown below:

```
<form name="contact" method="post" onsubmit="return userinformation(this);">
```

When the visitor presses the submit button, we call the **onsubmit** attribute. Inside the parenthesis, we call **userinformation** function, which checks for empty fields which are required and displays

an appropriate error message. We pass a parameter in the function call in order for the form to submit once required fields are entered.

### Create the JavaScript file

In your text editor, create a new file named **contactus.js**. Once completed, between the opening and closing head tags of **form.html**, link to the JavaScript file:

```
<head>  
<script type="text/javascript" src="contactus.js"></script>  
</head>
```

### Write the validation function

In **contactus.js**, write the following:

```
function userinformation(form) {  
  
}
```

We create a function called **userinformation**. This function will have multiple validation checks which will be called each time the submit button is pressed to ensure all required fields are not blank. Since we provide a parameter for the function in the opening form tag, we need to pass a parameter in the JavaScript file as well.

### Two methods to display our error message

The first method and arguably the easiest way to display the error message can be done by writing the following conditional statement:

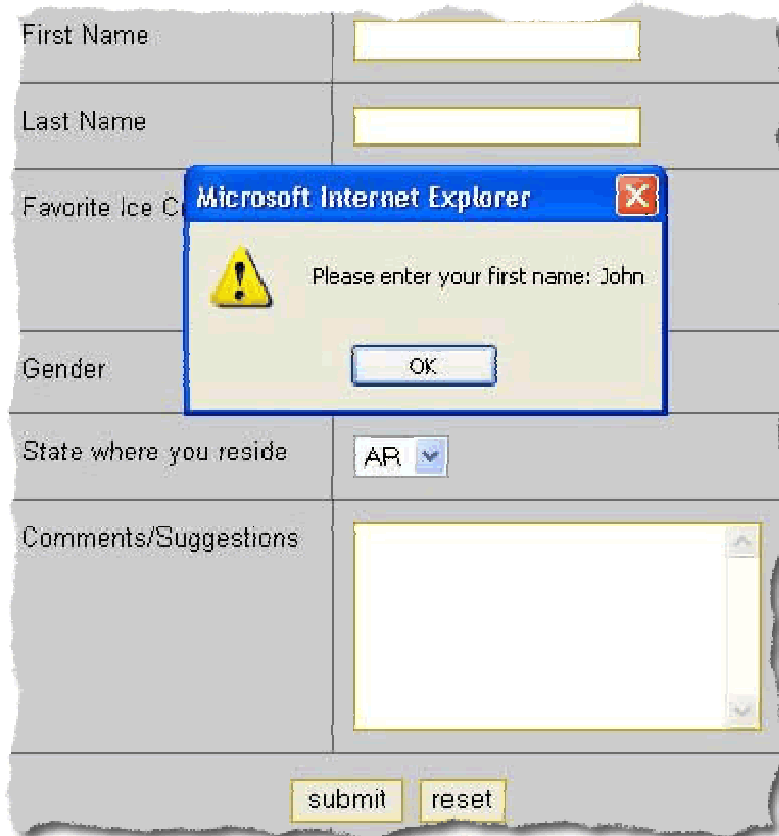
```
function userinformation(form) {  
    if(document.contact.fname.value=="") {  
        alert("Please enter your first name: John");  
    }  
}
```

Let's examine the code in greater detail.

First, inside the function we have a conditional statement which does the following:

- It checks to see if the value of the first name field in our form is equal to a null string by using a Boolean check (double equal's sign)
  - If it's true, we display an alert message
  - If it's false, we do nothing

Save your file. If you open the file in your browser and press the submit button without entering a value in the first name field, you receive the following error message:



While this method works and is simple to create, imagine this scenario: each time a visitor presses the submit button, one error message will show for each field which is required, and then the visitor clicks OK. Imagine if you have multiple fields in a form which require validation and making a visitor go through this mind-numbing experience each time they press the submit button.

A better method and one we'll use requires a little more work on our part, but a visitor will thank us in the end. Here's how it works:

- Create two variables
  - One named **message**, which holds our concatenated error message string
  - One named **noerrors**, which is set to the variable above

After performing our validation checks, we perform a simple conditional check with the **message** variable using a Boolean check (double equal's sign):

- If it's equal to **noerrors**, we do nothing
- If it's not equal to **noerrors**, we call the alert function and pass the **message** variable as a parameter which will inform a visitor which fields are missing

This method is more efficient and user friendly because we provide a visitor with one error message informing them of all errors which have occurred. As a visitor fills in the missing fields,

the error message will only show the remaining fields required, allowing for a more intuitive approach and experience for the visitor.

### Create message and noerrors variables

Inside our function, add the following code:

```
function userinformation(form){
    var message="Please complete the following: \n\n";
    var noerrors=message;
}
```

First, we declare two variables, `message` and `noerrors` by using the **var** keyword. We assign our **message** variable a text string informing our visitor what they need to complete. We set **noerrors** variable to the value of **message**. As mentioned before, we'll use a simple conditional check to determine the following:

- If **message** is equal to **noerrors**, we do nothing
- If **message** is not equal to **noerrors** we show an error dialog

We use a character escape sequence common to C derivative languages, `\n`, which causes a new line for each error which occurs.

### Validate first name field

We check the value of the first name field for a value by adding the following code:

```
function userinformation(form){
    var message="Please complete the following: \n\n";
    var noerrors=message;
    if(document.contact.fname.value==""){
        message+="Please enter your first name: John\n";
    }
    if(message==noerrors){
        return true;
    }else{
        alert(message);
        return false;
    }
}
```

In the conditional statement, we do the following:

- Check the value of the first name field (`fname`):
- If it's equal to a null string:
  - We assign and concatenate our **message** variable with an error message, followed by a new line

We can use shorthand syntax for the error message variable (+=) to keep our code more readable. The equivalent is:

```
message=message + "Please enter your first name: John\n";
```

Save your JavaScript file and preview the results.

[http://midwestwebdesign.net/tutorials/clientvalidation/form\\_1.html](http://midwestwebdesign.net/tutorials/clientvalidation/form_1.html)

### Validate last name field

We check the value of the last name field for a value by adding the following code:

```
function userinformation(form){
    var message="Please complete the following: \n\n";
    var noerrors=message;
    if(document.contact.fname.value==""){
        message+="Please enter your first name: John\n";
    }
    if(document.contact.lname.value==""){
        message+="Please enter your last name: Doe\n";
    }
    if(message==noerrors){
        return true;
    }else{
        alert(message);
        return false;
    }
}
```

In the conditional statement, we do the following:

- Check the value of the last name field (lname):
- If it's equal to a null string:
  - We assign and concatenate our **message** variable with an error message, followed by a new line

Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/clientvalidation/form\\_2.html](http://midwestwebdesign.net/tutorials/clientvalidation/form_2.html)

### Validating checkboxes

When working with check boxes, we take a slightly different approach to determine which one has been selected. Since check boxes can have one or multiple values selected, they are treated as an array. An array is simply a variable which holds different values. We check the value of the ice cream field for a value by adding the following code:

```

function userinformation(form){
    var message="Please complete the following: \n\n";
    var noerrors=message;
    if(document.contact.fname.value==""){
        message+="Please enter your first name: John\n";
    }
    if(document.contact.lname.value==""){
        message+="Please enter your last name: Doe\n";
    }
    var icecream = false;
    for (i = 0; i < document.contact.icecream.length; i++) {
        if (document.contact.icecream[i].checked)
            icecream = true;
    }
    if (!icecream) {
        message+= "Please select your favorite ice cream: Chocolate/Strawberry/Butt
Pecan\n";
    }
    if(message==noerrors){
        return true;
    }else{
        alert(message);
        return false;
    }
}

```

Let's examine the code in greater detail:

- We declare a variable **icecream** and assign a value of **false**
- Since there can be multiple checkboxes selected, we use a loop:
  - Our for loop works as follows:
    - Creates a counter variable, i, and sets the value to 0
    - We use a condition which checks:
      - Whether our counter variable is less than the number of checkboxes
      - If this is true, we continue looping and increment the counter each time through the loop
      - Then we check if one check box is checked:
        - If its checked, we set our icecream variable to true

Next, we create another condition using the negation operator. The negation operator (!) takes a value of true or false and switches it based on the value it receives. The condition works as follows:

- If the value of icecream is not equal to true, show an error message

Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/clientvalidation/form\\_3.html](http://midwestwebdesign.net/tutorials/clientvalidation/form_3.html)

### Validating radio buttons

Since radio buttons are Boolean values (true or false), we can check the value of our gender field by adding the following code:

```
function userinformation(form) {
    var message="Please complete the following: \n\n";
    var noerrors=message;
    if(document.contact.fname.value==""){
        message+="Please enter your first name: John\n";
    }
    if(document.contact.lname.value==""){
        message+="Please enter your last name: Doe\n";
    }
    var icecream = false;
    for (i = 0; i < document.contact.icecream.length; i++) {
        if (document.contact.icecream[i].checked)
            icecream = true;
    }
    if (!icecream) {
        message+= "Please select your favorite ice cream: Chocolate/Strawberry/Button
Pecan\n";
    }
    var gender=document.contact.gender;
    if(!gender[0].checked && !gender[1].checked){
    message+="Please select your gender: Male/Female\n";
    }
    if(message==noerrors){
        return true;
    }else{
        alert(message);
        return false;
    }
}
```

Let's examine the code in greater details:

- First, we declare a new variable named **gender** and assign it the gender input field
- Next, we use a condition to determine the following:
  - Using the negation operator, we check gender for a value. If false, we show an error message

The reason we access the first and second element of our gender array is we need to check that either male or female was selected. Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/clientvalidation/form\\_4.html](http://midwestwebdesign.net/tutorials/clientvalidation/form_4.html)

## Validating our drop down list

A drop down list is similar to a checkbox in that it's considered an array. However, only one option can be selected. We can check the value of our state field by adding the following code:

```
function userinformation(form){
    var message="Please complete the following: \n\n";
    var noerrors=message;
    if(document.contact.fname.value==""){
        message+="Please enter your first name: John\n";
    }
    if(document.contact.lname.value==""){
        message+="Please enter your last name: Doe\n";
    }
    var icecream = false;
    for (i = 0; i < document.contact.icecream.length; i++) {
        if (document.contact.icecream[i].checked)
            icecream = true;
    }
    if (!icecream) {
        message+= "Please select your favorite ice cream: Chocolate/Strawberry/Button
Pecan\n";
    }
    var gender=document.contact.gender;
    if(!gender[0].checked && !gender[1].checked){
        message+="Please select your gender: Male/Female\n";
    }
    var state=document.contact.state.selectedIndex;
    if(document.contact.state.options[state].value=="null"){
    message+="Please select a state: MO\n";
    }
    if(message==noerrors){
        return true;
    }else{
        alert(message);
        return false;
    }
}
```

Let's examine the code in greater detail:

- First, we declare a variable named **state** and assign it the selected index of our drop down list, which is a numerical value
- Next, in our condition we do the following:
  - We check the value of our drop down list:
    - If it's equal to a null value, we show an error message

Save your file and preview the results.

### Validating a text area field

We can check the value of a text area field the same way as an input field by adding the following code:

```
function userinformation(form) {
    var message="Please complete the following: \n\n";
    var noerrors=message;
    if(document.contact.fname.value==""){
        message+="Please enter your first name: John\n";
    }
    if(document.contact.lname.value==""){
        message+="Please enter your last name: Doe\n";
    }
    var icecream = false;
    for (i = 0; i < document.contact.icecream.length; i++) {
        if (document.contact.icecream[i].checked)
            icecream = true;
    }
    if (!icecream) {
        message+= "Please select your favorite ice cream: Chocolate/Strawberry/Buttton
Pecan\n";
    }
    var gender=document.contact.gender;
    if(!gender[0].checked && !gender[1].checked){
        message+="Please select your gender: Male/Female\n";
    }
    var state=document.contact.state.selectedIndex;
    if(document.contact.state.options[state].value=="null"){
        message+="Please select a state: MO\n";
    }
    if(document.contact.comments.value==""){
message+="Please provide us with some comments \n";
}
    if(message==noerrors){
        return true;
    }else{
        alert(message);
        return false;
    }
}
```

- In our condition we check the value of our comments field for a null value:
  - If it's equal to a null string, we show an error message

Save your file and preview the results.

### One add-on which might be useful

One addition to this form which makes the visitor experience a bit better is adding a reset check. If the visitor has entered information into form fields, and they accidentally press the reset button, instead of deleting their information from the form fields, we show an alert asking them to confirm the reset action.

### Create the reset file

Inside your text editor, create a new file named **resetcheck.js** and insert the following code:

```
// JavaScript Document
function confirmReset(){
var resetForm=confirm('Are you sure you want to reset the form?');
if(resetForm==true){
return true;
}else{
return false;
}
}
```

First, we create a function named **confirmReset**. Next, we create a variable named **resetForm**. Then we check the value of our variable using a condition to determine:

- If the variable is equal to true, we show the confirmation message
- If the variable is equal to false, we do nothing

In our HTML file, between the opening and closing head tags we link to the file:

```
<head>
<script type="text/javascript" src="contactus_7.js"></script>
<script src="resetcheck.js" type="text/javascript"></script>
</head>
```

Inside the opening form tag, we add an **onReset** attribute with a call to the **confirmReset** function which now will be called each time a visitor presses the reset button:

```
<form name="contact" method="post" onsubmit="userinformation(this);return false"
onreset="return confirmReset();">
```

Save your file and preview the results.

[http://midwestwebdesign.net/tutorials/clientvalidation/form\\_7.html](http://midwestwebdesign.net/tutorials/clientvalidation/form_7.html)

### Some general thoughts

If you have Dreamweaver, it will validate form fields except checkboxes and radio buttons. There are some extensions which will validate these fields if you prefer not to manually program it. You will need to a Google search to find which extension suits you best.

## Summary

In this article you learned how to:

- Create, save and link to an external JavaScript file
- Two different methods of showing error messages
- How to validate:
  - Input fields
  - Radio buttons
  - Checkboxes
  - Drop down lists and
  - Text area
- JavaScript syntax including how to create variables, loops, logical operators, and concatenation

Take the knowledge gained here and validate your form to your heart's content!

If you have questions, please follow the link below.

<http://midwestwebdesign.net/tutorials/contact.aspx>