

VOODOO'S INTRODUCTION TO JAVASCRIPT

© 1996, 1997 by Stefan Koch

Part 7: Forms

Validating form input

Forms are widely used on the Internet. The form input is often being sent back to the server or via mail to a certain e-mail account. But how can you be certain that a valid input was done by the user? With the help of JavaScript the form input can easily be checked before sending it over the Internet. First I want to demonstrate how forms can be validated. Then we will have a look at the possibilities for sending information over the Internet.

First of all we want to create a simple script. The HTML-page shall contain two text-elements. The user has to write his name into the first and an e-mail address into the second element. If the user has entered his name (for example 'Stefan') into the first text-field the script creates a popup window with the text 'Hi Stefan!'.

(The online version lets you test this script immediately)

Concerning the first input element you will receive an error message when nothing is entered. Any input is seen as valid input. Of course, this does not prevent the user from entering any wrong name. The browser even accepts numbers. So if you enter '17' you will get 'Hi 17!'. So this might not be a good check. The second form is a little bit more sophisticated. Try to enter a simple string - your name for example. It won't work (unless you have a @ in your name...). The criteria for accepting the input as a valid e-mail address is the @. A single @ will do it - but this is certainly not very meaningful. Every Internet e-mail address contains a @ so it seems appropriate to check for a @ here.

What does the script for those two form elements and for the validating look like? Here it goes:

```
<html>
<head>
<script language="JavaScript">
<!-- Hide

function test1(form) {
  if (form.text1.value == "")
    alert("Please enter a string!")
  else {
    alert("Hi "+form.text1.value+"! Form input ok!");
  }
}

function test2(form) {
```

```

    if (form.text2.value == "" ||
        form.text2.value.indexOf('@', 0) == -1)
        alert("No valid e-mail address!");
    else alert("OK!");
}
// -->
</script>
</head>

<body>
<form name="first">
Enter your name:<br>
<input type="text" name="text1">
<input type="button" name="button1" value="Test Input" onClick="test1(this.form)">
<P>
Enter your e-mail address:<br>
<input type="text" name="text2">
<input type="button" name="button2" value="Test Input" onClick="test2(this.form)">
</body>
</html>

```

First have a look at the HTML-code in the body-section. We just create two text elements and two buttons. The buttons call the functions *test1(...)* or *test2(...)* depending on which button is pressed. We pass *this.form* to the functions in order to be able to address the right elements in the functions later on. The function *test1(form)* tests if the string is empty. This is done via *if (form.text1.value == "")... . 'form'* is the variable which receives the *'this.form'* value in the function call. We can get the value of the input element through using *'value'* in combination with *form.text1*. In order to look if the string is empty we compare it with *""*. If the input string equals *""* then no input was done. The user will get an error message. If something is entered the user will get an ok.

The problem here is that the user might enter only spaces. This is seen as a valid input! If you want to, you can of course check for these possibilities and exclude them. I think this is quite easy with the information given here. Now have a look at the *test2(form)* function. This function again compares the input string with the empty string *""* to make sure that something has been entered. But we have added something to the if-command. The *||* is called the OR-operator. You have learned about it in part 6 of this introduction. The if-command checks if either the first or the second comparison is true. If at least one of them is true the whole if-command gets *true* and the following command will be executed. This means that you will get an error message either if your string is empty or if there isn't a *@* in your string. The second operation in the if-command looks if the entered string contains a *@*.

Checking for certain characters

Sometimes you want to restrict the form input to certain characters or numbers. Just think of a telephone number - the input should only contain digits (we assume that the telephone number does not contain any characters). We could check if the input is a number. But most people use different symbols in their telephone number - for example:

01234-56789, *01234/56789* or *01234 56789* (with a space inbetween). The user should not be forced to enter the telephone number without these symbols. So we have to extend our script to check for digits and some symbols. This is demonstrated in the next example which is taken

from my JavaScript book:

(The online version lets you test this script immediately)

```
<html>
<head>
<script language="JavaScript">
<!-- hide

// *****
// Script from Stefan Koch - Voodoo's Intro to JavaScript
// http://rummelplatz.uni-mannheim.de/~skoch/js/
// JS-book: http://www.dpunkt.de/javascript
// You can use this code if you leave this message
// *****

function check(input) {
  var ok = true;

  for (var i = 0; i < input.length; i++) {
    var chr = input.charAt(i);
    var found = false;
    for (var j = 1; j < check.length; j++) {
      if (chr == check[j]) found = true;
    }
    if (!found) ok = false;
  }

  return ok;
}

function test(input) {

  if (!check(input, "1", "2", "3", "4",
    "5", "6", "7", "8", "9", "0", "/", "-", " ")) {

    alert("Input not ok.");
  }
  else {
    alert("Input ok!");
  }
}

// -->
</script>
</head>

<body>
<form>
Telephone:
<input type="text" name="telephone" value="">
```

```

<input type="button" value="Check"
  onClick="test(this.form.telephone.value)">
</form>
</body>
</html>

```

The function *test()* specifies which characters are valid.

Submitting form input

What different possibilities do exist for submitting form input? The easiest way is to submit the form input via e-mail. This is the method we are going to look at a little bit closer. If you want the form input to be handled by the server you need to use CGI (Common Gateway Interface). This allows you to process the form input automatically. The server might for example build up a database from the input received by some customers. Another example are index-pages like Yahoo. They usually have a form for making a search in their database. The user gets a response quickly after the submit button was hit. He does not have to wait until the people maintaining this server read the input and then look up the information requested. This is done automatically by the server. JavaScript cannot do things like this. You cannot create guestbooks with JavaScript because JavaScript isn't able to write to a file on the server. You can only do this through CGI. Of course you can create a guestbook with the people answering via e-mail. You have to enter the feedback manually though. This is ok if you don't expect to get 1000 feedback mails a day.

This script here is plain HTML. So no JavaScript is needed here! Only, of course, if you want to check the input before the form is submitted you will need JavaScript. I have to add that the mailto-command does not work everywhere - for example the Microsoft Internet Explorer 3.0 does not support it.

```

<form method=post action="mailto:your.address@goes.here" enctype="text/plain">
Do you like this page?
  <input name="choice" type="radio" value="1">Not at all.<br>
  <input name="choice" type="radio" value="2" CHECKED>Waste of time.<br>
  <input name="choice" type="radio" value="3">Worst site of the Net.<br>
  <input name="submit" type="submit" value="Send">
</form>

```

The property *enctype="text/plain"* is used in order to send plain text without encoded parts. This makes it much easier to read the mail.

If you want to validate the form before it is sent over the net you can use the *onSubmit* event-handler. You have to put this event-handler into the *<form>* tag. This looks like this:

```

function validate() {
  // check if input ok
  // ...

  if (inputOK) return true
  else return false;
}

```

...

```
<form ... onSubmit="return validate()">
```

...

With this code the form isn't being sent over the Internet if the form input was wrong.

Setting the focus to a certain form-element

With the help of the *focus()* method you can make your form a little bit more user-friendly. You can define which element is in focus at the beginning. Or you could tell the browser to focus on the form where the input was wrong. This means that the browser will set the cursor into the specified form-element so the user does not have to click on the form before entering anything. You can do this with the following piece of script:

```
function setfocus() {  
    document.first.text1.focus();  
}
```

This script would set the focus to the first text-element in the script I have shown above. You have to specify the name of the whole form - which is called *first* here - and the name of the single form element - here *text1*. If you want to put the focus on this element when the page is being loaded you can add an *onLoad*-property to your *<body>* tag. This looks like this:

```
<body onLoad="setfocus()">
```

We can extend this with the following code:

```
function setfocus() {  
    document.first.text1.focus();  
    document.first.text1.select();  
}
```

(The online version lets you test this script immediately)

The text-element gets the focus and the text contained in this text-element is being selected.

©1996,1997 by Stefan Koch
e-mail:skoch@rumms.uni-mannheim.de
http://rummelplatz.uni-mannheim.de/~skoch/
My JavaScript-book: http://www.dpunkt.de/javascript