Form Validation

Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button. If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This was really a lengthy process which used to put a lot of burden on the server.

JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions.

- **Basic Validation** − First of all, the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.
- **Data Format Validation** − Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.

Basic Form Validation

First let us see how to do a basic form validation. In the above form, we are calling **validate()** to validate data when **onsubmit** event is occurring. The following code shows the implementation of this validate() function.

Data Format Validation

The following example shows how to validate an entered email address. An email address must contain at least a '@' sign and a dot (.). Also, the '@' must not be the first character of the email address, and the last dot must at least be one character after the '@' sign.

The indexOf() method returns the position of the first occurrence of a specified value in a string.

This method returns -1 if the value to search for never occurs.

string.indexOf(*searchvalue*, *start*)

searchvalue	Required. The string to search for
start	Optional. Default 0. At which position to start the search

Return Value:	A Number, representing the position where the specified searchvalue occurs for the first time, or -1 if it never occurs

String methods:

String Length

The `length` property returns the length of a string:

```
var txt = "welcome";
var sln = txt.length;
```

Finding a String in a String

The `indexOf()` method returns the index of (the position of) the `first` occurrence of a specified text in a string:

```
ar str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
```

The `lastIndexOf()` method returns the index of the **last** occurrence of a specified text in a string:

```
var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("locate");
```

Both `indexOf()`, and `lastIndexOf()` return -1 if the text is not found.

The `search()` method searches a string for a specified value and returns the position of the match:

The two methods, `indexOf()` and `search()`, are **equal?**

They accept the same arguments (parameters), and return the same value?

The two methods are **NOT** equal. These are the differences:

- The `search()` method cannot take a second start position argument.
- The `indexOf()` method cannot take powerful search values (regular expressions).

Extracting String Parts

There are 3 methods for extracting a part of a string:

- `slice(`*`start`*`, `*`end`*`)`
- `substring(`*`start`*`, `*`end`*`)`
- `substr(`*`start`*`, `*`length`*`)`

The slice() Method

`slice()` extracts a part of a string and returns the extracted part in a new string.

The method takes 2 parameters: the start position, and the end position (end not included).

This example slices out a portion of a string from position 7 to position 12 (13-1):

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(7, 13);
```

If a parameter is negative, the position is counted from the end of the string.

This example slices out a portion of a string from position -12 to position -6:

The substring() Method

`substring()` is similar to `slice()`.

The difference is that `substring()` cannot accept negative indexes

Replacing String Content

The `replace()` method replaces a specified value with another value in a string:

By default, the `replace()` method replaces **only the first** match:

By default, the `replace()` method is case sensitive. Writing MICROSOFT (with upper-case) will not work:

Converting to Upper and Lower Case

A string is converted to upper case with `toUpperCase()`:

The concat() Method

`concat()` joins two or more strings:

String.trim()

The `trim()` method removes whitespace from both sides of a string:

The charAt() Method

The `charAt()` method returns the character at a specified index (position) in a string: