MetaCharacters for repeated pattern Matching

The metacharacters can also be used to match more than one characters, for that purpose we use what is called as***quantifiers***

For eg: to match multiple uppercase letters , the regular expression to be used is **/[A-Z]+/** here the **+ sign** means ***"one or more than one of previous characters"***. The table below provides a list of all quantifiers.

Table: Regular Expression : Quantifiers - Greedy Metacharacters.

| Metacharacter | Description |
| --- | --- |
| *x?* | *To match 0 or 1 of x* |
| *(xyz)?* | *To match zero or one pattern of xyz.* |
| *x\** | *To match 0 or more of x* |
| *(xyz)\** | *To match zero or more patterns of xyz* |
| *x+* | *To match 1 or more of x* |
| *(xyz)+* | *To match one or more patterns of xyz* |
| *x{m,n}* | *To match at least m of x and no more than n of x.* |

Javascript Regular Expressions Metacharacters: Greed Factor

A quantifier is called as ***greedy*** if it matches the largest possible set of characters from left to right, checking the last possible character that satisfies the pattern.

The demo below , the regular expression ***/[A-Z][a-z]\*\s*** checks for an uppercase letter ***[A-Z]*** , followed by a zero or more lowercase characters ***[a-z]\**** and a space, ***\s***. There can be zero or more lowercase characters.

<script>

function check(){

var regex = /[A-Z][^a-z]\*\s/;

var userString = document.forms["regform"]["myinput"].value;

var output = regex.test(userString);

if(output)

{

window.alert("every thing good");

return false

}

else

{

window.alert("this is not a valied formate");

return false

}

}

</script>

Javascript Regular Expressions Metacharacters: Greed Factor

A quantifier is called as ***greedy*** if it matches the largest possible set of characters from left to right, checking the last possible character that satisfies the pattern.

The demo below , the regular expression ***/[A-Z][a-z]\*\s*** checks for an uppercase letter ***[A-Z]*** , followed by a zero or more lowercase characters ***[a-z]\**** and a space, ***\s***. There can be zero or more lowercase characters.

## JavaScript Form Validation: Checking for Zip Codes

A simple six digit zipcode can be checked using regular expression which matches exactly six digits : **/^d{6}$/**

Another way can be **/^[0-9][0-9][0-9][0-9][0-9][0-9]$/** , for zipcode with dash in between it can be **/^\d{6}-?\d{5}$** .

<script>

function check\_Zip(zipCode){

var regex = /^\d{5}$/;

if(regex.test(zipCode.value) == false){

alert("The lenght of zip code must be atleast five digits");

zipCode.focus();

return false;

}

if(zipCode.value == " "){

alert("Zip code Field cannot be left empty");

zipCode.focus();

return false;

}

return true;

}

</script>

## Javascript Form Validation: Alphabetic Data Input

There may be some form fields whose values must be strictly alphabetic characters, eg: Name, Country, State.

In the below demo the regular expression looks for one or more uppercase or lowercase letters within the character class ***[A-Za-z]***, followed with an end of a line anchor ***$***

<script>

function check\_Alpha(letters){

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

if(regex.test(letters.yourname.value) == false){

alert("Name must be in alphabets only");

letters.yourname.focus();

return false;

}

if(letters.yourname.value == " "){

alert("Name Field cannot be left empty");

letters.yourname.focus();

return false;

}

return true;

}

</script>

## JavaScript Form Validation: Removing Spaces and Dashes

Some undesired spaces and dashes from the user input can be removed by using the string object ***replace()*** method.

The regular expression is used to find the characters and then replace them with empty spaces.

<script>

var demoString = "123-45-5675";

var regex = /[ -]+/g;

var replacedString = demoString.replace(regex, " ");

document.write("The Original String was : " + demoString);

document.write("<br>The replaced String : " + replacedString);

</script>

## JavaScript Form Validation: Removing Unwanted Parentheses

Unwanted parentheses surrounding area codes or telephone numbers, can be removed using regular expression and method***replace()***

The regular expression searches for one or more parentheses, spaces or dashes , globally.

<script>

var demoString = "(123)-45-5675";

var regex = /[() -]+/g;

var replacedString = demoString.replace(regex, " ");

document.write("The Original String was : " + demoString);

document.write("<br>The replaced String : " + replacedString);

</script>

## JavaScript Form Validation: Removing Nondigits

Nondigit(i.e character that is not a digit ) can be removed using the regular expression ***[^0-9]*** or ***\D***. Eg: nondigits in zipcodes, phone numbers etc.

<script>

var demoString = "(123)-//[45%6]-5675@4%&";

var regex = /\D/g;

var replacedString = demoString.replace(regex, " ");

document.write("The Original String was : " + demoString);

document.write("<br>The replaced String : " + replacedString);

</script>

## javaScript Form Validation: Phone Numbers

An USA phone number has ten digits, it comprises of : a three digit area code, subscriber number of seven digit.

The area code may have a parentheses around the area code, and dashes or spaces seperating the numbers in the subscriber number.

<script>

function check\_phone(phno){

var regex = /^\(?\d{3}\)?-?\s\*-?\d{4}$/;

if(regex.test(phno.yourphone.value)){

return true;

}

else{

alert("This is not a valid phone number");

return false;

}

}

</script>

The phone number is of type ***(XXX)-XXX-XXXX*** , ***XXX-XXX-XXXX*** or ***XXXXXXXXXX***