JavaScript Regular Expression

Regular Expression in JavaScript

In the previous part of this JavaScript tutorial, you learnt to do simple things to manipulate a string by using the properties and methods of a string object.

For example, If you want to replace any digit character in a string "Java123JavaScript897C" with "Programming, ". How can you do that? It is hard to accomplish this task.

For another example, if you want to allow a user to enter a phone number in the format: (855)987-678. It is also hard to complete this task by using what you have learnt with the string object. Fortunately, JavaScript provides Regular Expression object that you can solve these problems. By using Regular Expression object, you can define a complex pattern of string to be matched.

For example, to solve the first problem, you can write a string pattern as below:
var Stpat=/\d+/g;

Then supply this string variable to the replace() method of your string object. A string pattern of regular expression object is written between the forward slash(/pattern/) to inform the browsers that it is regular expression.

Writing a string pattern for global searching and case-insensitivity

In general, Regular Expression object matches only the first string pattern. This means that when the string pattern is matched at the first time in another string then the searching process will stop.

For example, we have a string "JavaScript-HTML-CSS". We want to replace the minus signs(-) with commas. Our string and string pattern should be:

```
var st="JavaScript-HTML-CSS";

var stpat=/-/;
```

To replace the minus signs with commas, the replace() method is used:

```
var stnew=st.replace(stpat,",");
```

Now let see the full code:

```
<html>
<head>
<title>Regular Expression-a simple string pattern</title>

<script language="JavaScript" type="text/javascript">

var st="JavaScript-HTML-CSS";
var stpat=/-/;
var stnew=st.replace(stpat,",");
```

```
document.write(stnew+"<br>");

</script>

</head>
<body>
</body>
</html>
```

By executing this code on a browser, you will get:

JavaScript,HTML-CSS

It is not want we really want. We want all the minus signs are replaced by commas. To solve this problem, you need to use the **g** character with the string pattern:

```
var stpat=/-/g;
```

Therefore, the code above should be modified to:

```
<html>
<head>
<title>Regular Expression-a simple string pattern</title>

<script>

var st="JavaScript-HTML-CSS";
var stpat=/-/g;
var stnew=st.replace(stpat,",");
document.write(stnew+"<br>");

</script>

</head>
<body>
</body>
</html>
```

The Regular Expression object is case-sensitive. The lower case and upper case letters are treated differently. If you want a pattern to be matched in case-insensitive manner, you need to use the i character with your string pattern:

```
var stpat=/-/gi;
```

Writing a simple string pattern

Now let write a simple string pattern to split the following string object to an array--starr:

```
var st="JavaScript is powerful web scripting language";
```

This string can be split by the space character. Therefore, string pattern should be:

```
var stpat=/ /;
```

Then by supplying this string pattern as an argument of the split() method you will get an array of strings.

```
var starr=st.split(stpat);
```

Now you see the complete example code as shown below:

```
<html>
<head>
<title>Regular Expression-a simple string pattern</title>

<script language="JavaScript" type="text/javascript">

var st="JavaScript is powerful web scripting language";
var stpat=/ /;
var starr=st.split(stpat);
var i;
for(i=0;i<starr.length;i++)
document.write(starr[i]+"<br>");

</script>

</head>
<body>
</body>
</html>
```

Special characters

To write more complex string patterns, you need to know how to use special characters in the string patterns. The special characters and examples are listed below:

Special Character	Description
\d	Matches any digit character from 0 to 9. Example: /\d\d\d/ matches 123 but not 12a ore a12,...
\D	Matches any character that is not a digit. Example: /\D/ matches A but not 1.
\s	Matches any character that is a whitespace including tab, carriage return, formfeed, new line, and vertical tab. Example: /\s/ matches tab.
\S	Matches any character that is not a whitespace character. Example: /\S/ matches A but not tab.
\w	Matches any character that is a character from a to z, A to Z, and underscore. Example: /\w/ match _ but not 0.

\W	Matches any character that is a not a character from a to z, A to Z, and underscore. Example: /\W/ matches @ but not a.
[...]	Matches any character that is in brackets. Example: /[ab]/ matches a or b only.
[^...]	Matches any character that is not in bracket. Example:/ [^ab]/ matches A or B but not a or b.
.	Matches any character that is not a new line. Example /./ matches a,0, or %
?	Matches the previous item 0 time or 1 time. Example: /J?/ matches J or not at all.
*	Matches the previous item 0 time or more times. Example: /J*/ matches J, JJ, JJJ, or not at all
+	Matches the previous item 1 time or more times. Example: /J+/ matches J,JJ, or JJJ.
{n}	Matches the previous item n times. Example:/\d{2}/ matches 34, 44, or 89.
{n,}	Matches the previous item more than n times. Example: /\d{1,}/ matches 0, 10,287,3897, or 90000.
{n,m}	Matches the previous item from n to m times. Example: /\d{1,3}/ matches 1, 23, or 897.
^	Matches a pattern at the beginning of a string.
$	Matches a pattern at the end of a string.
\b	Matches a word boundary position.
\B	Matches a position that is not a word boundary.

For example, now you want a user to enter his/her phone number in a text box in the following example format: (855)89-256-724. To check the phone number validity(conforming to this format), your pattern should be:

```
var stpat=/[(]855[)]\d{2}-\d{3}-\d{3}/gi;
```

To help you understand the pattern, we break it in to small parts:

[(] and [)] match opening and closing parentheses.

/855/ matches 855.

/\d{2}/ matches any two digits.

/-/ matches the minus sign.

/\d{3}/ matches any three digits.

Now let see this task is done on the browser by executing the code:

```
<html>
<head>
<title>Text field element</title>
<script language="JavaScript" type="text/javascript">
function check(){
var ph=document.forms[0].phone.value;
var stpat=/[(]855[)]\d{2}-\d{3}-\d{3}/gi;
if(ph.search(stpat)!=-1) alert("Valid!");
else alert("Invalid!");
}
</script>
</head>
<body>
<form action="checkphone.html">
Enter phone number: <input type="text" name="phone" /><br />
<input type="button" onclick="check()" value="Submit">
</form>
</body>
</html>
```

Grouping Expressions

You can group more than one expressions together. For example, you have a string a like "Java Programming, JavaScript Programming are the same?". How can you write a pattern to match Java Programming and JavaScript Programming? This kind of problem can be solved by grouping Java and JavaScript patterns together in parentheses. Java and JavaScript both end with Programming, so the pattern should be:

```
var stpat=/\b(Java)?\b(JavaScript)\bProgramming\b/;
```

Reusing groups of characters

For example, you have a string like "JavaScript,HTML,HTML,CSS,PHP,ASP, CSS,CSS,CSS,CSS". In fact this string should contain only distinct words. How would you mark the duplicate words and replace them with "Removed"? You can solve this problem by providing group number(\1 is refers to group 1, \2 refers to group 2 and so on):

```
var stpat=/([a-z]+), \1/gi;
```

```
<html>
<head>
<title>Regular Expression-reusing groups of characters</title>

<script language="JavaScript" type="text/javascript">

var st= "JavaScript,HTML,HTML,JavaScript,CSS,PHP,ASP,CSS,CSS,CSS,CSS";
var stpat= /([a-z]+),\1/gi;
var stnew=st.replace(stpat,"Removed");
document.write(stnew);
</script>
```

```
</head>
<body>
</body>
</html>
```