REGULAR EXPRESSIONS AND REGEXP OBJECT

A regular expression is an object that describes a pattern of characters.

The JavaScript **RegExp** class represents regular expressions, and both String and **RegExp** define methods that use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.

Syntax

A regular expression could be defined with the **RegExp** constructor, as follows −

```
var pattern = new RegExp(pattern, attributes);

or simply

var pattern = /pattern/attributes;
```

Here is the description of the parameters −

- **pattern** − A string that specifies the pattern of the regular expression or another regular expression.
- **attributes** − An optional string containing any of the "g", "i", and "m" attributes that specify global, case-insensitive, and multiline matches, respectively.

Brackets

Brackets [] have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

Expression	Description
[...]	Any one character between the brackets.
[^...]	Any one character not between the brackets.
[0-9]	It matches any decimal digit from 0 through 9.
[a-z]	It matches any character from lowercase a through lowercase z.
[A-Z]	It matches any character from uppercase A through uppercase Z.
[a-Z]	It matches any character from lowercase a through uppercase Z.

The ranges shown above are general; you could also use the range [0-3] to match any decimal digit ranging from 0 through 3, or the range [b-v] to match any lowercase character ranging from **b** through **v**.

Quantifiers

The frequency or position of bracketed character sequences and single characters can be denoted by a special character. Each special character has a specific connotation. The +, *, ?, and $ flags all follow a character sequence.

Expression	Description
p+	It matches any string containing at least one p.

p*	It matches any string containing zero or more p's.
p?	It matches any string containing one or more p's.
p{**N**}	It matches any string containing a sequence of **N** p's
p{2,3}	It matches any string containing a sequence of two or three p's.
p{2, }	It matches any string containing a sequence of at least two p's.
p$	It matches any string with p at the end of it.
^p	It matches any string with p at the beginning of it.

Examples

Following examples explain more about matching characters.

Expression	Description
[^a-zA-Z]	It matches any string not containing any of the characters ranging from **a** through **z** and **A** through Z.
p.p	It matches any string containing **p,** followed by any character, in turn followed by another **p**.
^.{2}$	It matches any string containing exactly two characters.
. *	It matches any string enclosed within and .
p*hp**	It matches any string containing a **p** followed by zero or more instances of the sequence **hp**.

Literal characters

Character	Description
Alphanumeric	Itself
\0	The NUL character \u0000
\t	Tab \u0009
\n	Newline \u000*A*
\v	Vertical tab \u000*B*
\f	Form feed \u000*C*
\r	Carriage return \u000*D*
\xnn	The Latin character specified by the hexadecimal number nn; for example, \x0A is the same as \n
\uxxxx	The Unicode character specified by the hexadecimal number xxxx; for example, \u0009 is the same as \t
\cX	The control character ^X; for example, \cJ is equivalent to the newline character \n

Metacharacters

A metacharacter is simply an alphabetical character preceded by a backslash that acts to give the combination a special meaning.

For instance, you can search for a large sum of money using the '\d' metacharacter: **/[\d]** + **000/**, Here **\d** will search for any string of numerical character.

The following table lists a set of metacharacters which can be used in PERL Style Regular Expressions.

Character	Description
.	a single character
\s	a whitespace character *space, tab, newline*
\S	non-whitespace character
\d	a digit 0 – 9
\D	a non-digit
\w	a word character a-z, A-Z, 0-9, _
\W	a non-word character
[\b]	a literal backspace *specialcase*.
[aeiou]	matches a single character in the given set
[^aeiou]	matches a single character outside the given set
foo|bar|baz	matches any of the alternatives specified

Modifiers

Several modifiers are available that can simplify the way you work with **regexps,** like case sensitivity, searching in multiple lines, etc.

Modifier	Description
i	Perform case-insensitive matching.
m	Specifies that if the string has newline or carriage return characters, the ^ and $ operators will now match against a newline boundary, instead of a string boundary
g	Performs a global matchthat is, find all matches rather than stopping after the first match.

RegExp Properties

Here is a list of the properties associated with RegExp and their description.

Property	Description
constructor	Specifies the function that creates an object's prototype.
global	Specifies if the "g" modifier is set.

ignoreCase	Specifies if the "i" modifier is set.
lastIndex	The index at which to start the next match.
multiline	Specifies if the "m" modifier is set.
source	The text of the pattern.

In the following sections, we will have a few examples to demonstrate the usage of RegExp properties.

RegExp Methods

Here is a list of the methods associated with RegExp along with their description.

Method	Description
exec	Executes a search for a match in its string parameter.
test	Tests for a match in its string parameter.
toSource	Returns an object literal representing the specified object; you can use this value to create a new object.
toString	Returns a string representing the specified object.

In the following sections, we will have a few examples to demonstrate the usage of RegExp methods.