A regular expression is an object that describes a pattern of characters. Usually the term is abbreviated *regex* of *regexp*. JavaScript uses *RegExp*. They are used to perform pattern matching and search-and-replace functions on text. The syntax looks like this:

```
var myText = new RegExp(pattern, modifier);
```

A simpler syntax looks like this:

```
var myText = /pattern/modifier;
```

The term pattern specifies an expression's arrangement and deliver's a rule for that arrangement. The term modifier stipulates if a search should be global, case-sensitive or multiline.

Here is a reference for regexes as they are used in JavaScript.

Modifiers

Modifiers are used to perform case-insensitive and global searches:

Modifier	Description
I	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Brackets

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character not between the brackets
[0-9]	Find any digit from 0 to 9
[a-z]	Find any character from lowercase a to lowercase z
[A-Z]	Find any character from uppercase A to uppercase Z
[a-Z]	Find any character from lowercase a to uppercase Z
[adgk]	Find any character in the given set
[^adgk]	Find any character outside the given set
[red|blue|green]	Find any of the alternatives specified

Metacharacters

Metacharacters are characters with a special meaning:

Copyright © 2010 by Paul Roberts • All rights reserved

Metacharacter	Description
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word
\B	Find a match not at the beginning/end of a word
\0	Find a NUL character
\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character
\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd
\uxxxx	Find the Unicode character specified by a hexadecimal number xxxx

Quantifiers

Quantifier	Description
n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n
n?	Matches any string that contains zero or one occurrences of n
n{X}	Matches any string that contains a sequence of X n's
n{X,Y}	Matches any string that contains a sequence of X or Y n's
n{X,}	Matches any string that contains a sequence of at least X n's
n$	Matches any string with n at the end of it
^n	Matches any string with n at the beginning of it
?=n	Matches any string that is followed by a specific string n
?!n	Matches any string that is not followed by a specific string n

RegExp Object Properties

Property	Description
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

Property	Description
source	The text of the RegExp pattern

RegExp Object Methods

Method	Description
compile()	Compiles a regular expression
exec()	Tests for a match in a string. Returns a result array
test()	Tests for a match in a string. Returns true or false

Example

Suppose you want to specify the pattern for a U.S. ZIP code. Here is how you can do it:

```
^\d{5}(-\d{4})?$
```

Explanation

RegExp	Interpretation
^	Match any string with what follows ^ at the beginning of the string
\d	Match any digit
{5}	Match a sequence of five of what is before this
(xxxxxxxx)	Matches the group within the parentheses, adding the group to the foregoing
-	Hyphen literal
\d	Match any digit
{4}	Match a sequence of four of what is before this
?	Match zero or one occurrence of the foregoing
$	Match whatever is at the end of the foregoing