RegEx

Topics Covered:

- What is RegEx?
- Brackets in RegEx.
- Quantifiers in RegEx.
- Characters in RegEx.
  - Characters.
  - Literal characters.
  - Metacharacters.
- Modifiers in RegEx.
- RegEx properties.
- RegEx methods.

Topics in Detail:

What is RegEX?

- A regular expression is an object that describes a pattern of characters.
- The JavaScript RegExp class represents regular expressions and defines methods that use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.
- A regular expression could be defined with the RegExp () constructor.
- Syntax:
  var pattern = new RegExp(pattern, attributes);
  or
  var pattern = /pattern/attributes;
- Where
  - 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.

Bracket	Description
[...]	Any one character between the bracket.
[^...]	Any one character not between the bracket.
[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.
- We can use the range [0-3] to match any decimal digit ranging from 0 through 3.
- The range [b-v] to match any lowercase character ranging from b through v.

Quantifiers:

- The frequency of 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 one or more p's.
p*	It matches any string containing zero or more p's.
p?	It matches any string containing at most one p.
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.

Matching Characters:

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.
<b>(.*)</b>	It matches any string enclosed within <b> and </b>.
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 (\u000A)
\v	Vertical tab (\u000B)
\f	Form feed (\u000C)
\r	Carriage return (\u000D)
\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 numeric characters.
- 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 (special case).
[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

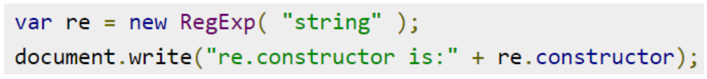
Modifiers:

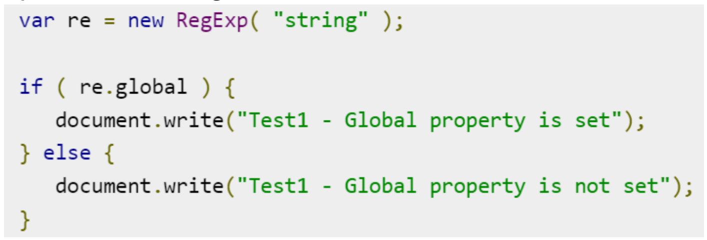
Several modifiers are available that can simplify the way you work with regexps, like case sensitivity, searching in multiple lines, etc.

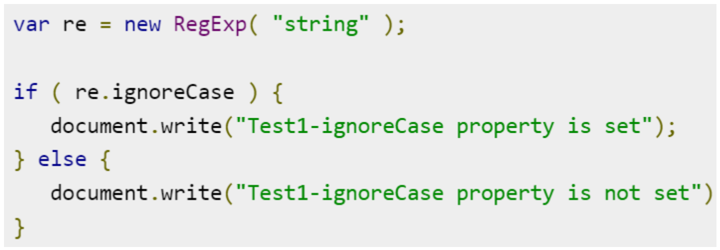
Modifiers	Description
i	Perform case-insensitive matching.
m	Specifies that if the string has newline or carriage return characters, the ^ and $ operators will now match against a newline boundary, instead of a string boundary
g	Performs a global matchthat is, find all matches rather than stopping after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

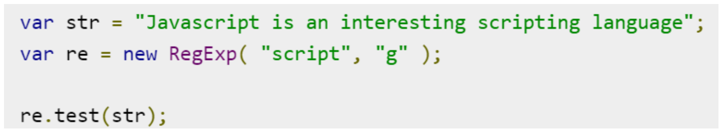
RegExp properties:

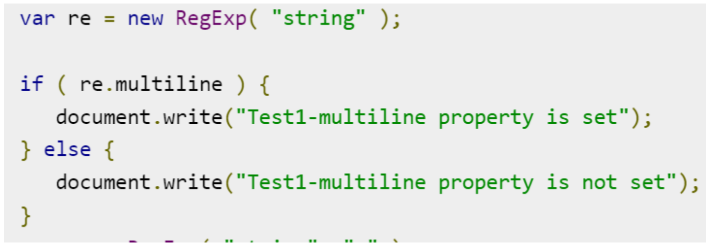
Properties	Description

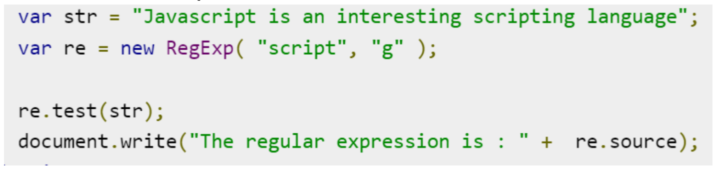
constructor	Specifies the function that creates an object's prototype.
	
	`var re = new RegExp( "string" );`
	`document.write("re.constructor is:" + re.constructor);`

global	Specifies if the "g" modifier is set.
	
	`var re = new RegExp( "string" );`
	
	`if ( re.global ) {`
	`   document.write("Test1 - Global property is set");`
	`} else {`
	`   document.write("Test1 - Global property is not set");`
	`}`

ignoreCase	Specifies if the "i" modifier is set.
	
	`var re = new RegExp( "string" );`
	
	`if ( re.ignoreCase ) {`
	`   document.write("Test1-ignoreCase property is set");`
	`} else {`
	`   document.write("Test1-ignoreCase property is not set")`
	`}`

lastIndex	The index at which to start the next match.
	
	`var str = "Javascript is an interesting scripting language";`
	`var re = new RegExp( "script", "g" );`
	
	`re.test(str);`

multiline	Specifies if the "m" modifier is set.
	
	`var re = new RegExp( "string" );`
	
	`if ( re.multiline ) {`
	`   document.write("Test1-multiline property is set");`
	`} else {`
	`   document.write("Test1-multiline property is not set");`
	`}`

source	The text of the pattern.
	
	`var str = "Javascript is an interesting scripting language";`
	`var re = new RegExp( "script", "g" );`
	
	`re.test(str);`
	`document.write("The regular expression is : " +  re.source);`

RegExp Methods:

Method	Description

exec()	Executes a search for a match in its string parameter.
	`var str = "Javascript is an interesting scripting language";`
	`var re = new RegExp( "script", "g" );`
	
	`var result = re.exec(str);`

test()	Tests for a match in its string parameter.
	`var str = "Javascript is an interesting scripting language";`
	`var re = new RegExp( "script", "g" );`
	
	`var result = re.test(str);`

toSource()	Returns an object literal representing the specified object; you can use this value to create a new object.
	
	`var str = "Javascript is an interesting scripting language";`
	`var re = new RegExp( "script", "g" );`
	
	`var result = re.toSource(str);`

toString()	Returns a string representing the specified object.
	`var str = "Javascript is an interesting scripting language";`
	`var re = new RegExp( "script", "g" );`
	
	`var result = re.toString(str);`