***REGULAR EXPRESSIONS:***

EXPRESSION FLAGS:

* If object is supplied for the pattern the flag will replace any of the objects flag and last value is reset to 0.
* If flag is not specified then the regular expressions object is supplied , then that flag value of last index value will be copied over.
* The value is a string containing any combination of the values like:

1. g(global match)
2. i(ignore case)
3. m(multi line)
4. u(Unicode)
5. y(sticky).

* Advantage of this is a large number of parsing problems are easier to solve with it, and also easy to modify the all values by changing it to global.

CHARACTER POSITIONS:

* It is nothing but a finding the exact position of the character in a given string or conditions.
* The caret(^) matches the positions before the first character in the string applying ^abc only the caret will apply to the a not to bc because it positions to first string.
* Similarly $ matches the last character in a string.

Eg:

Var village= formers;

Village.match(/ers$/) it produce [“ers”, index:4, input:”formers”]

Village.match(/^for/) it produce [“for”,index:0, input:”formers”]

* The main us of it will be the identification of required text content in a big part of content.

REPREATING CHARACTERS:

* The repeating character will contain certain number of conditions which will provide the repeating and representation of a particular character.
* It basically contain \*,+,?.
* \*= repeat it zero or more times.
* += repeat 1 or more times.
* ?= repeat 0 or 1 time.

EXAMPLE:

/\*ab/ it matches abbb.

/+abc/ it matches abccc

/?123abc/ it matches 123

* It will be used to validate the forms or other things by analysing this pattern match.

ESCAPE SEQUANCE:

* A backslash that proceeds a non-special characters indicates that the next character is special and is not be interpreted literally.
* A backslash that proceeds a special characters indicates that the next character is special and it should be interpreted literally.
* By, contrast the pattern /z\\*/ it remove the ‘\*’ to enable matches with string like ‘\*z’.

Eg:

name //b/ : // it will cause ambiguity so, to resolve this , problem the concept of escape sequence is introduced.

Name / \b/.

ALTERNATIVE PATTERN AND GROUPING:

* The most common use for re is to search for patterns in text. The regular expression consists of a pattern and optional flags.

New RegExp is allows to construct patterns dynamically.