Regular Expressions

Regular Expressions

- A regular expression is a sequence or pattern of characters that specifies a rule.
- Regular expressions are used to do pattern matching, which is used in form validation.
- 2 ways to create regular expression in javascript
  - Using literal
    - var reExample = /pattern/;
  - Using the RegExp() constructor
    - var reExample = new RegExp("pattern");

Regular expression methods

- Exec()
- Test()

String methods which use regular expressions

- Match()
- Search()
- Replace()
- Split()

The exec() Method

- The exec() method takes one argument, a string, and checks whether that string contains one or more matches of the pattern specified by the regular expression.
- If one or more matches is found, the method returns a result array with the starting points of the matches.
- If no match is found, the method returns null.

Test method()

- The test() method also takes one argument, a string, and checks whether that string contains a match of the pattern specified by the regular expression.
- It returns true if it does contain a match and false if it does not.

```
var reexp = /^\d+$/;
reexp.test(“1234567”);
```

The search() Method

- The search() method takes one argument: a regular expression.
- It returns the index of the first character of the substring matching the regular expression.
- If no match is found, the method returns -1.1
- “University".search(/sit/); //returns 6

The split() Method

- The split() method takes one argument: a regular expression. It uses the regular expression as a delimiter to split the string into an array of strings.
- “University".split(/[aeiou]/);
- /*
- returns an array with the following values:
- “n", “v", “r", “s", “t“, ”y”
- */

The replace() Method

- The replace() method takes two arguments: a regular expression and a string.
- It replaces the first regular expression match with the string.
- If the g flag is used in the regular expression, it replaces all matches with the string.
- “HelloWorld".replace(/World/, “VIT"); //returns

  HelloVIT
- “Hello".replace(/[aeiou]/g, "x"); //returns Hxllx

The match() Method

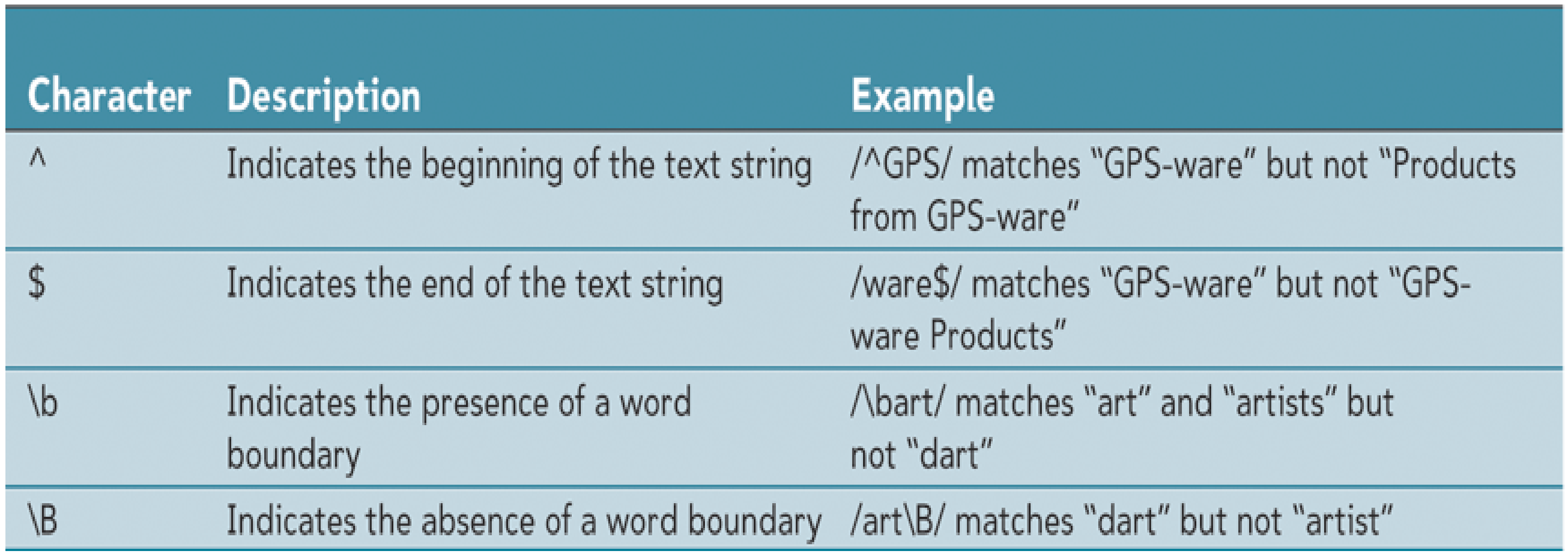
- The match() method takes one argument: a regular expression.
- It returns each substring that matches the regular expression pattern.
- “University".match(/[aeiou]/g);

- returns an array with the following values:

  “U", “i", “e", “i“

Position matching

- Defining character positions

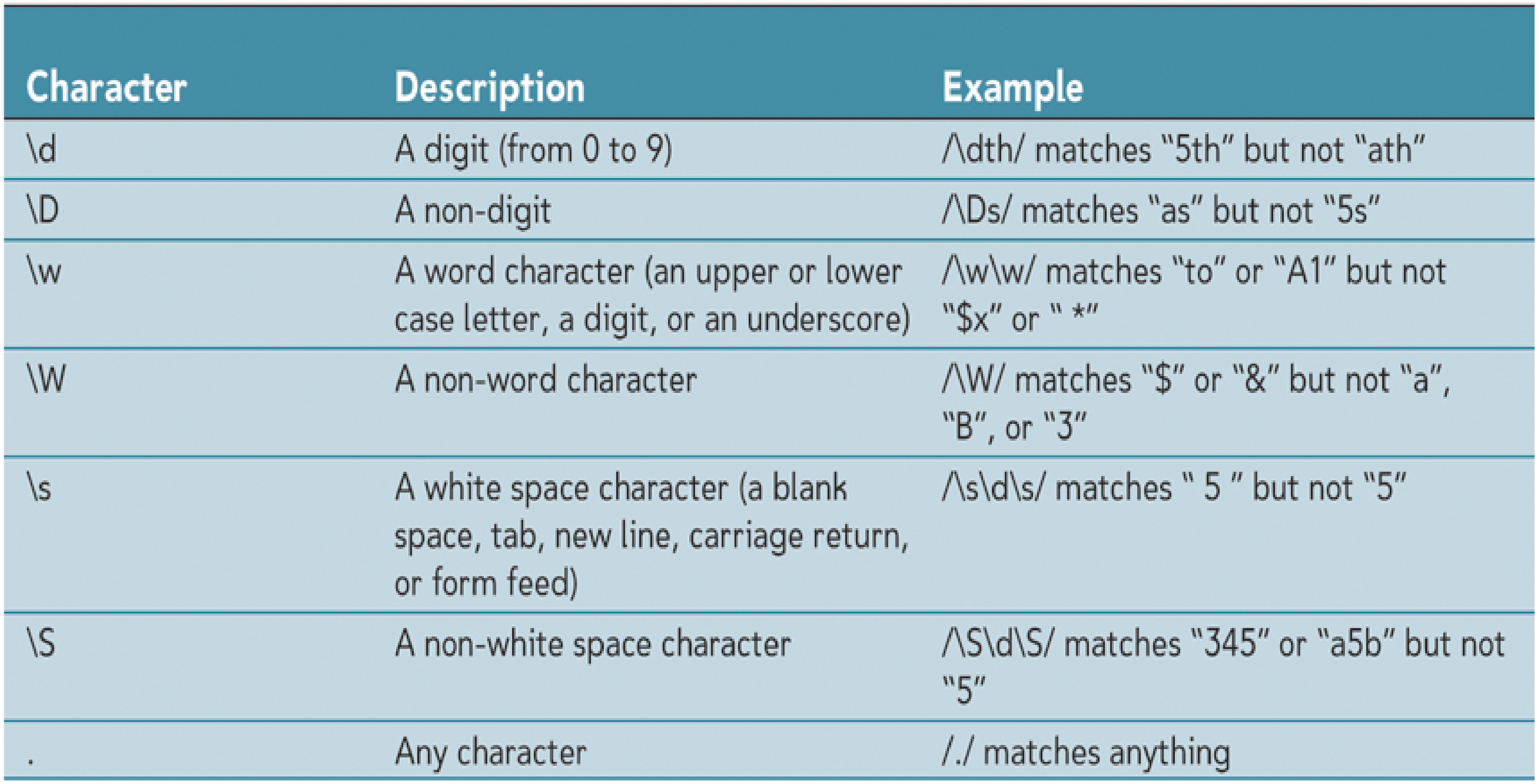


Character	Description	Example
^	Indicates the beginning of the text string	/^GPS/ matches "GPS-ware" but not "Products from GPS-ware"
$	Indicates the end of the text string	/ware$/ matches "GPS-ware" but not "GPS-ware Products"
\b	Indicates the presence of a word boundary	/\bart/ matches "art" and "artists" but not "dart"
\B	Indicates the absence of a word boundary	/art\B/ matches "dart" but not "artist"

Literals

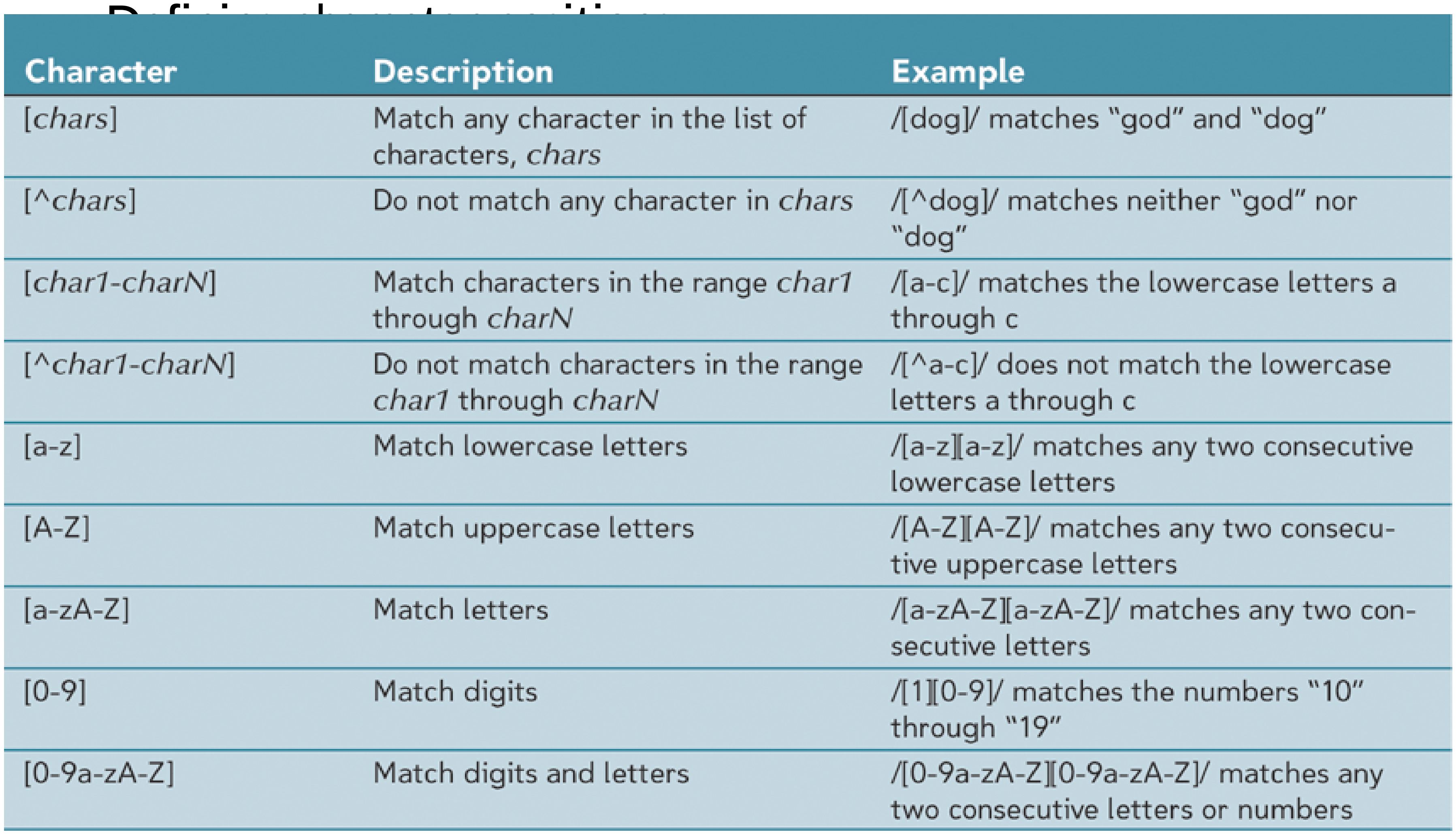
Symbol	Description
Alphanumeric	All alphabetical and numerical characters match themselves literally. So /2 days/ will match "2 days" inside a string.
\n	Matches a new line character
\f	Matches a form feed character
\r	Matches carriage return character
\t	Matches a horizontal tab character
\v	Matches a vertical tab character
\xxx	Matches the ASCII character expressed by the octal number xxx. "\50" matches left parentheses character "("
\xdd	Matches the ASCII character expressed by the hex number dd. "\x28" matches left parentheses character "("
\uxxxx	Matches the ASCII character expressed by the UNICODE xxxx. "\u00A3" matches "£".

- Defining character positions



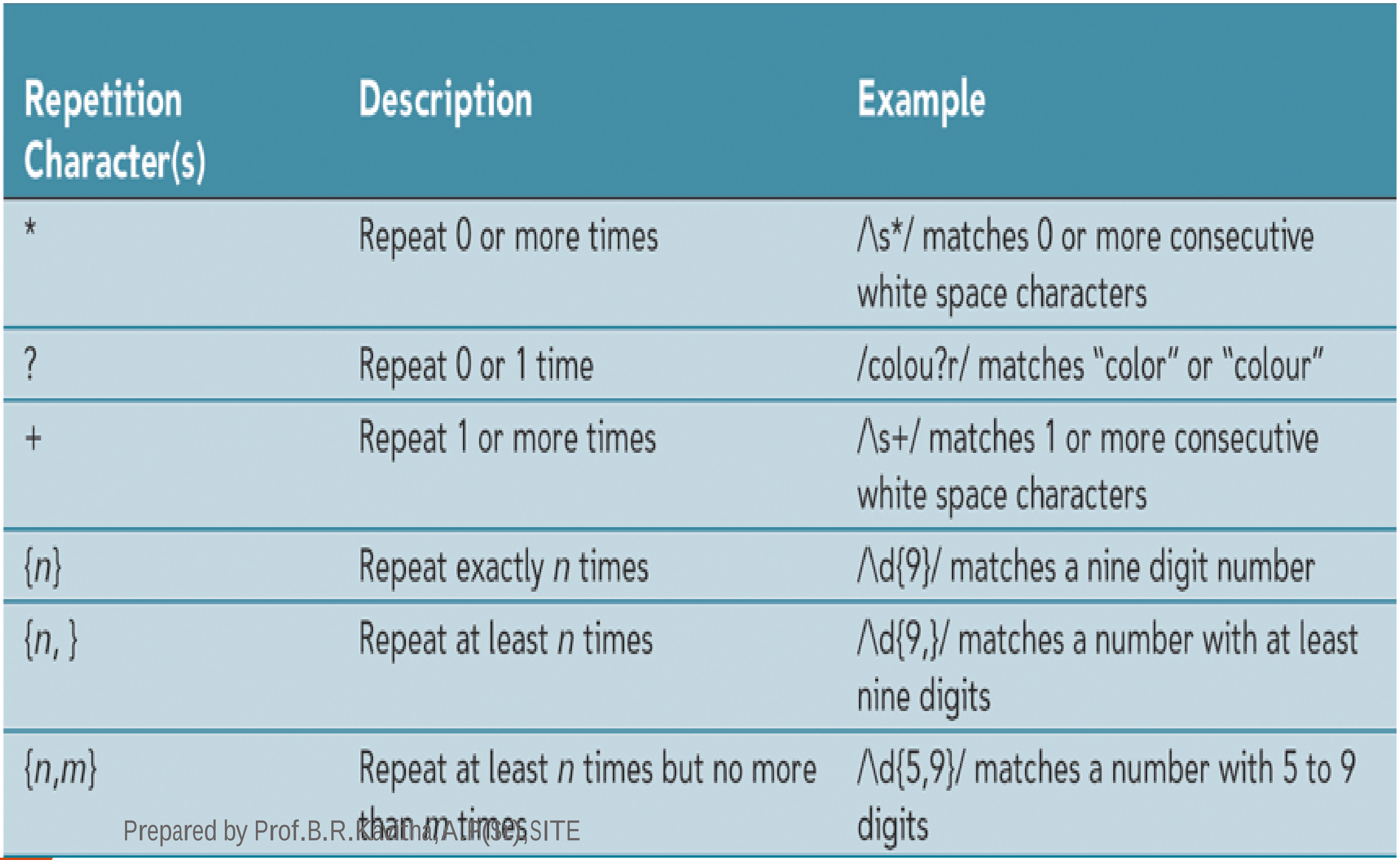
Character	Description	Example
\d	A digit (from 0 to 9)	/\dth/ matches "5th" but not "ath"
\D	A non-digit	/\Ds/ matches "as" but not "5s"
\w	A word character (an upper or lower case letter, a digit, or an underscore)	/\w\w/ matches "to" or "A1" but not "$x" or " *"
\W	A non-word character	/\W/ matches "$" or "&" but not "a", "B", or "3"
\s	A white space character (a blank space, tab, new line, carriage return, or form feed)	/\s\d\s/ matches " 5 " but not "5"
\S	A non-white space character	/\S\d\S/ matches "345" or "a5b" but not "5"
.	Any character	/./ matches anything

Defining character positions



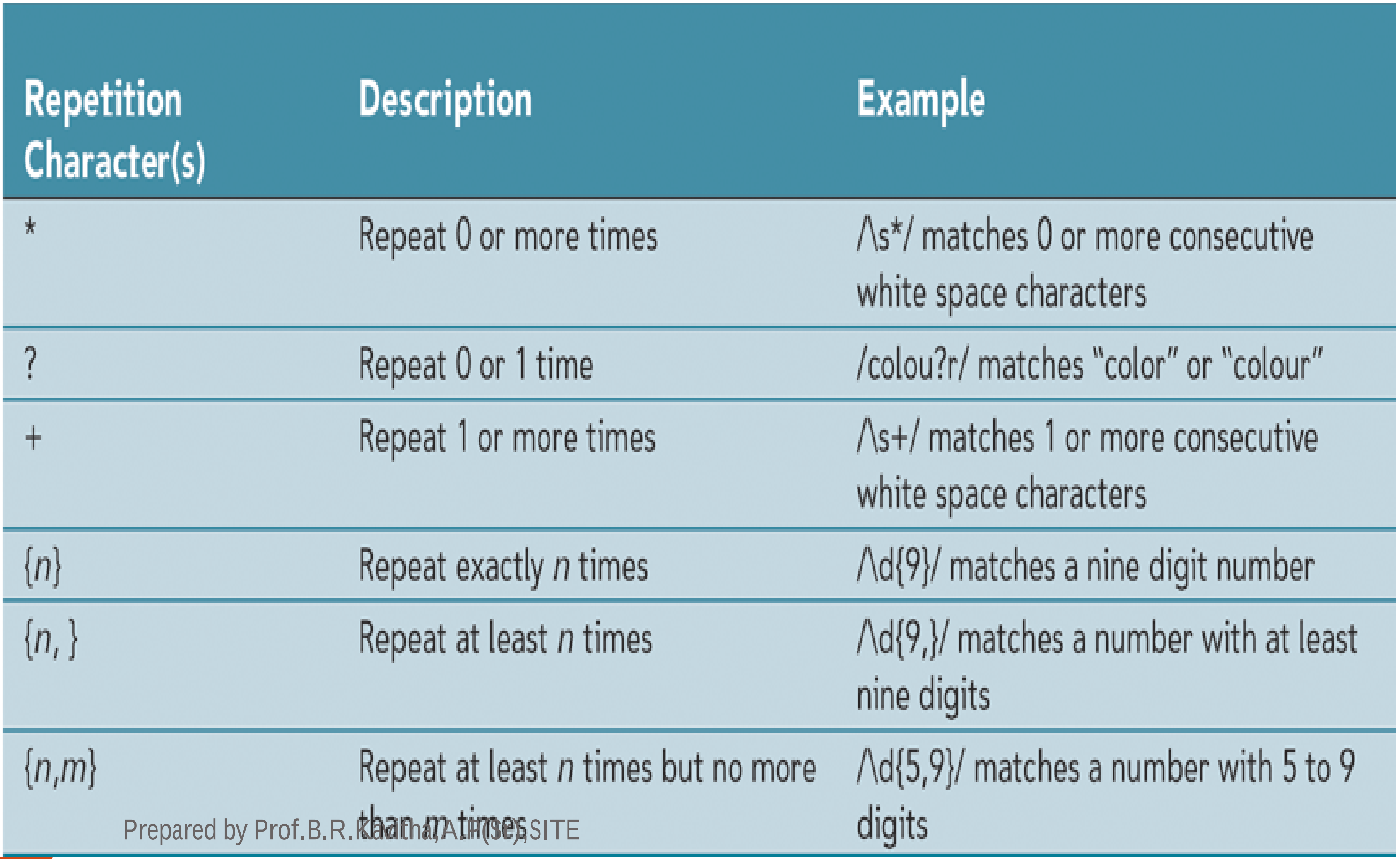
Character	Description	Example
[*chars*]	Match any character in the list of characters, *chars*	/[dog]/ matches "god" and "dog"
[^*chars*]	Do not match any character in *chars*	/[^dog]/ matches neither "god" nor "dog"
[*char1-charN*]	Match characters in the range *char1* through *charN*	/[a-c]/ matches the lowercase letters a through c
[^*char1-charN*]	Do not match characters in the range *char1* through *charN*	/[^a-c]/ does not match the lowercase letters a through c
[a-z]	Match lowercase letters	/[a-z][a-z]/ matches any two consecutive lowercase letters
[A-Z]	Match uppercase letters	/[A-Z][A-Z]/ matches any two consecutive uppercase letters
[a-zA-Z]	Match letters	/[a-zA-Z][a-zA-Z]/ matches any two consecutive letters
[0-9]	Match digits	/[1][0-9]/ matches the numbers "10" through "19"
[0-9a-zA-Z]	Match digits and letters	/[0-9a-zA-Z][0-9a-zA-Z]/ matches any two consecutive letters or numbers

- Repeating characters



Repetition Character(s)	Description	Example
*	Repeat 0 or more times	/\s*/ matches 0 or more consecutive white space characters
?	Repeat 0 or 1 time	/colou?r/ matches "color" or "colour"
+	Repeat 1 or more times	/\s+/ matches 1 or more consecutive white space characters
{*n*}	Repeat exactly *n* times	/\d{9}/ matches a nine digit number
{*n*, }	Repeat at least *n* times	/\d{9,}/ matches a number with at least nine digits
{*n*,*m*}	Repeat at least *n* times but no more than *m* times	/\d{5,9}/ matches a number with 5 to 9 digits

- Repeating characters



Repetition Character(s)	Description	Example
*	Repeat 0 or more times	/\s*/ matches 0 or more consecutive white space characters
?	Repeat 0 or 1 time	/colou?r/ matches "color" or "colour"
+	Repeat 1 or more times	/\s+/ matches 1 or more consecutive white space characters
{*n*}	Repeat exactly *n* times	/\d{9}/ matches a nine digit number
{*n*, }	Repeat at least *n* times	/\d{9,}/ matches a number with at least nine digits
{*n*,*m*}	Repeat at least *n* times but no more than *m* times	/\d{5,9}/ matches a number with 5 to 9 digits

Grouping and Alteration

Symbol	Description
( )	Grouping characters together to create a clause. May be nested. Example: /(abc)+(def)/ matches one or more occurrences of "abc" followed by one occurrence of "def".
|	Alternation combines clauses into one regular expression and then matches any of the individual clauses. Similar to "OR" statement. Example: /(ab)|(cd)|(ef)/ matches "ab" or "cd" or "ef".

Pattern switches or flags

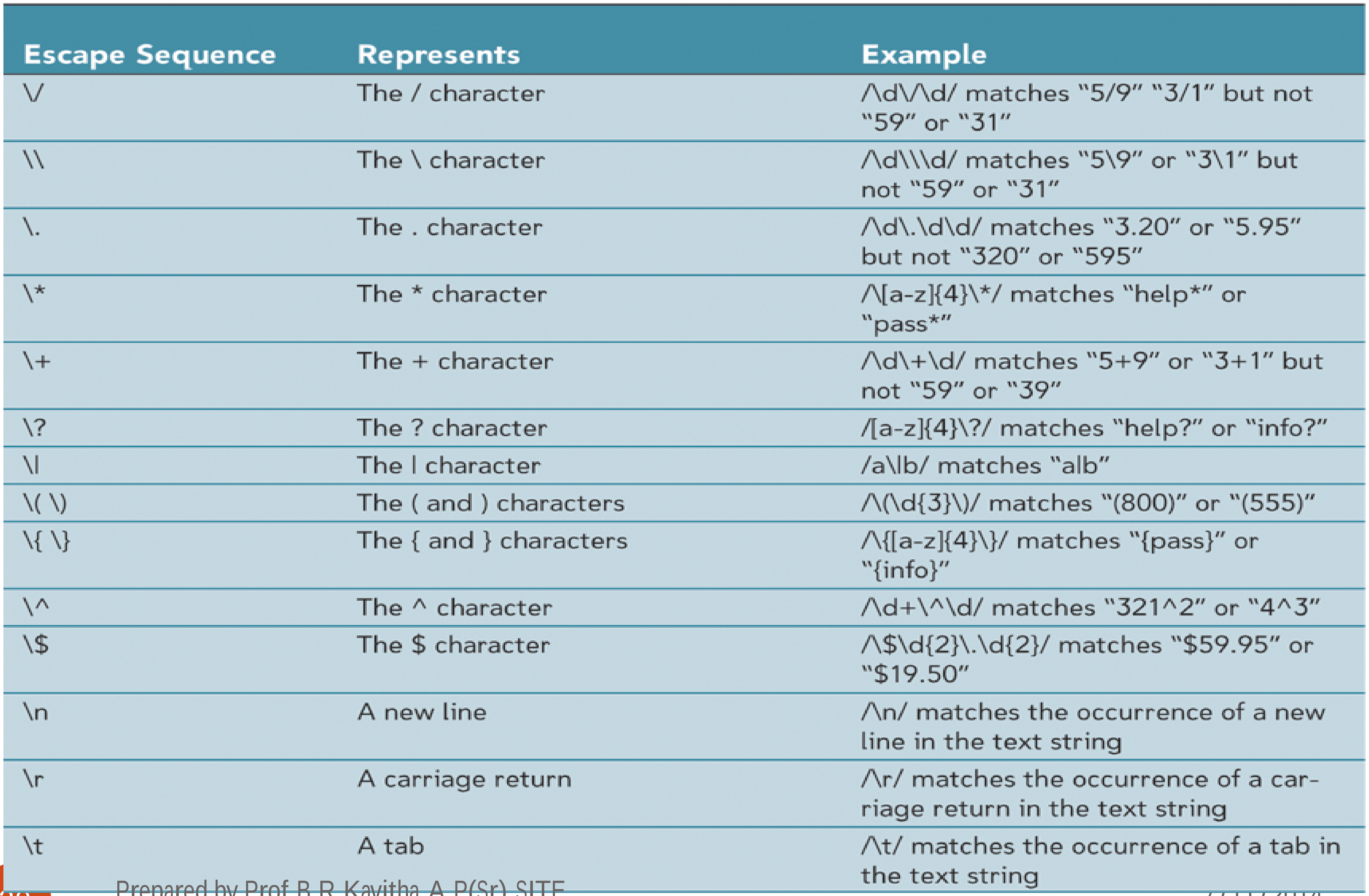
Property	Description
i	Ignore the case of characters. /The/i matches "the" and "The" and "tHe"
g	Global search for all occurrences of a pattern Ex: /ain/g matches both "ain"s in "No pain no gain", instead of just the first.
gi	Global search, ignore case. Ex: /it/gi matches all "it"s in "It is our IT department"

Back references

- Backreferences are special wildcards that refer back to a subpattern within a pattern.
- They can be used to make sure that two subpatterns match.
- The first subpattern in a pattern is referenced as \1, the second is referenced as \2, and so on.
- Ex:([bmpw])o\1 matches "bob", "mom", "pop", and "wow", but not "bop" or "pow“
- ^\d{3}([\-]?)\d{2}\1\d{4}$ ---- in checkSSN program

Backslash

- The backslash (\) is also used when you wish to match a special character literally. For example, if you wish to match the symbol "$" literally instead of have it signal the end of the string, backslash it: /\$/



Escape Sequence	Represents	Example
\/	The / character	/\d\/\d/ matches "5/9" "3/1" but not "59" or "31"
\\	The \ character	/\d\\\d/ matches "5\9" or "3\1" but not "59" or "31"
\.	The . character	/\d\.\d\d/ matches "3.20" or "5.95" but not "320" or "595"
*	The * character	/\[a-z]{4}*/ matches "help*" or "pass*"
\+	The + character	/\d\+\d/ matches "5+9" or "3+1" but not "59" or "39"
\?	The ? character	/[a-z]{4}\?/ matches "help?" or "info?"
\\|	The | character	/a\\|b/ matches "a|b"
\(\)	The (and) characters	/\(\d{3}\)/ matches "(800)" or "(555)"
\{ \}	The { and } characters	/\{[a-z]{4}\}/ matches "{pass}" or "{info}"
\^	The ^ character	/\d+\^\d/ matches "321^2" or "4^3"
\$	The $ character	/\$\d{2}\.\d{2}/ matches "$59.95" or "$19.50"
\n	A new line	/\n/ matches the occurrence of a new line in the text string
\r	A carriage return	/\r/ matches the occurrence of a carriage return in the text string
\t	A tab	/\t/ matches the occurrence of a tab in the text string

Some Examples in form validation

- //contact no.

  var reg=/^([0-9]{7,12})$/;

- // following pattern specifies the rules for email

  /^([A-Za-z0-9_\-\.])+\@([A-Za-z0-9_\-\.])+\.([A-Za-z]{2,4})$/;

- // following pattern allows exactly 6 character long string having numaric characters only

  // "^" indicates begining of string, "$" indicates end of string.

  var reg = /^([0-9]{6})$/;

References

- http://www.advanced-javascript-tutorial.com/RegularExpressions.cfm#.UgCO86wVg1h

- http://www.javascriptkit.com/javatutors/re2.shtml
- Web Technologies – Uttam K.Roy