Regex Cheat Sheet

```
# shorthand character classes
regex = /d/; # matches any digit, short for [0-9]
regex = /D/; # matches non-digits, short for [^0-9]
regex = /S/; # matches non-white space character
regex = /s/; # matches any white space character
regex = /w/; # matches character, short for [a-zA-Z_0-9]
regex = /W/; # matches non-word character [^w]
regex = /b/; # Matches a word boundary where a word character is [a-zA-Z0-9_]
# These meta characters boast a pre-defined meaning and make various typical patterns easi
er to use.

# matching using quantifiers
regex = /X./; # matches any character
regex = /X*/; # Matches zero or several repetitions of letter X, is short for {0,}
regex = /X+-/; # matches one or more repetitions of letter X, is short for {1,}
regex = /X?/; # finds no or exactly one letter X, is short for is short for {0,1}.
regex = // d{3}; # matches three digits. {} describes the order of the preceding liberal
regex = // d{1,4} ; # means d must occur at least once and at a maximum of four
# A quantifies helps developers to define how often an element occurs.

# character ranges
regex = /[a-z]/; # matches all lowercase letters
regex = /[A-Z]/; # matches all uppercase letters
regex = /[e-l]/; # matches lowercase letters e to l (inclusive)
regex = /[F-P]/; # matches all uppercase letters F to P (inclusive)
regex = /[0-9]/; # matches all digits
regex = /[5-9]/; # matches any digit from 5 to 9 (inclusive)
regex = / [a-d1-7]/; # matches a letter between a and d and figures from 1 to 7, but not d
1
regex = /[a-zA-Z]/; # matches all lowercase and uppercase letters
regex = /[^a-zA-Z]/; # matches non-letters

# matching using anchors
regex = / ^The/; # matches any string that starts with “The”
regex = / end$/; # matches a string that ends with end
regex = / ^The end$/; # exact string match starting with “The” and ending with “End”

# escape characters
regex = / a/; # match a bell or alarm
regex = / e/; # matches an escape
regex = / f/; # matches a form feed
regex = / n/; # matches a new line
regex = / Q…E/; # ingnores any special meanings in what is being matched
regex = / r/; # matches a carriage return
regex = / v/; # matches a vertical tab
# It is critical to note that escape characters are case sensitive
```

```
# matching using flags
regex = / i/; # ignores the case in pattern ( upper and lower case allowed)
regex = / m/; # multi-line match
regex = / s/; # match new lines
regex = / x/; # allow spaces and comments
regex = / j/; # duplicate group names allowed
regex = / U/; # ungreedy match
# Besides the regular expressions, flags can also be used to help developers with pattern
 matching.

# matching a specific string
regex = /sing/; # looks for the string between the forward slashes 9case-sensitive)… match
es “sing”, “sing123”
regex = /sing/i; # looks for the string between the forward slashes (case-insensitive)...
 matches "sing", "SinNG", "123SinNG"
regex = /hello/g; # looks for multiple occurrences of string between the forward slashe
s...

# groups
regex = /it is (sizzling )?hot outside/; # matches "it is sizzling hot outside" and "it is
hot outside"
regex = /it is (?:sizzling )?hot outside/; # same as above except it is a non-capturing gr
oup
regex = /do (dogs) like pizza 1/; # matches "do dogs like pizza dogs"
regex = /do (dogs) like (pizza)? do 2 1 like you?/; # matches "do dogs like pizza? do pizz
a dogs like you?"

# look-ahead and look-behind
regex = /d(?=r)/; # matches 'd' only if it is followed by 'r', but 'r' will not be part of
the overall regex match
regex = / (?<=r)d /; # matches 'd' only if it is proceeded by an 'r', but 'r' will not be
 part of the overall regex match
```