<https://www.tutorialrepublic.com/php-tutorial/php-regular-expressions.php>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions>

**[]** denote a character class. **()** denotes a capturing group.

Regular Expression:

**Regular Expression Syntax:**

Regular expression syntax includes the use of special characters (do not confuse with the HTML special characters).

The characters that are given special meaning within a regular expression, are: **. \* ? + [ ] ( ) { } ^ $ | \**. You will need to backslash these characters whenever you want to use them literally. For example, if you want to match ".", you'd have to write \.. All other characters automatically assume their literal meanings.

**Character Classes:**

A character class always matches a single character out of a list of specified characters that means the expression [abc] matches only a, b or c character.

Negated character classes can also be defined that match any character except those contained within the brackets. A negated character class is defined by placing a caret (^) symbol immediately after the opening bracket, like this [^abc]

You can also define a range of characters by using the hyphen (-) character inside a character class, like [0-9]

**Predefined Character Classes:**

|  |  |
| --- | --- |
| . | Matches any single character except newline \n. |
| \d | matches any digit character. Same as [0-9] |
| \D | Matches any non-digit character. Same as [^0-9] |
| \s | Matches any whitespace character (space, tab, newline or carriage return character). Same as [ \t\n\r] |
| \S | Matches any non-whitespace character. Same as [^ \t\n\r] |
| \w | Matches any word character (definned as a to z, A to Z,0 to 9, and the underscore). Same as [a-zA-Z\_0-9] |
| \W | Matches any non-word character. Same as [^a-zA-Z\_0-9] |

Capturing Groups:

Boundaries:

|  |  |
| --- | --- |
| ^ | Matches beginning of input. If the multiline flag is set to true, also matches immediately after a line break character. For example, /^A/ does not match the "A" in "an A", but does match the first "A" in "An A". |
| $ | Matches end of input. If the multiline flag is set to true, also matches immediately before a line break character. For example, /t$/does not match the "t" in "eater", but does match it in "eat". |
| \b | A **word boundary** character ( \b) helps you search for the words that begins and/or ends with a pattern. For example, the regexp /\bcar/ matches the words beginning with the pattern car, and would match cart, carrot, or cartoon, but would not match oscar.  Similarly, the regexp /car\b/ matches the words ending with the pattern car, and would match scar, oscar, or supercar, but would not match cart. Likewise, the /\bcar\b/ matches the words beginning and ending with the pattern car, and would match only the word car. |
| \B |  |

Pattern Modifiers:

|  |  |
| --- | --- |
| g | Global match |
| i | Case-insensitive search |
| m | Multi-line search |
| s | Allows . to match newline characters. |
|  |  |

**Quantifier:** Indicates number of characters or expressions to match.

**Repetition Quantifiers:**

|  |  |
| --- | --- |
| n+ | Matches one or more occurrences of the letter n. |
| n\* | Matches zero or more occurrences of the letter n. |
| n? | Matches zero or one occurrences of the letter n. |
| n{2} | Matches exactly two occurrences of the letter n. |
| n{2,3} | Matches at least two occurrences of the letter n, but not more than three occurrences of the letter n. |
| n{2,} | Matches two or more occurrences of the letter n. |
| n{,3} | Matches at most three occurrences of the letter n. |

**Position Anchors:**

There are certain situations where you want to match at the beginning or end of a line, word, or string. To do this you can use anchors. Two common anchors are caret (^) which represent the start of the string, and the dollar ($) sign which represent the end of the string.

|  |  |
| --- | --- |
| ^p | Matches the letter p at the beginning of a line. |
| p$ | Matches the letter p at the end of a line. |

**Assertions:** Indicates in some way that a match is possible. Assertions include look-ahead, look-behind, and conditional expressions.

**Look around:**

<https://www.regular-expressions.info/lookaround.html>

?: is for non capturing group

?= is for positive look ahead

?! is for negative look ahead

?<= is for positive look behind

?<! is for negative look behind

|  |  |
| --- | --- |
| Characters | Meaning |
| x(?=y)  **Positive Lookahead** | Matches "x" only if "x" is followed by "y". For example, /Jack(?=Sprat)/ matches "Jack" only if it is followed by "Sprat".  /Jack(?=Sprat|Frost)/ matches "Jack" only if it is followed by "Sprat" or "Frost". However, neither "Sprat" nor "Frost" is part of the match results. |
| x(?!y)  **Negative lookahead** | Matches "x" only if "x" is not followed by "y". For example, /\d+(?!\.)/ matches a number only if it is not followed by a decimal point. /\d+(?!\.)/.exec('3.141') matches "141" but not "3". |
| (?<=y)x  **Lookbehind assertion** | Matches "x" only if "x" is preceded by "y". For example, /(?<=Jack)Sprat/ matches "Sprat" only if it is preceded by "Jack". /(?<=Jack|Tom)Sprat/ matches "Sprat" only if it is preceded by "Jack" or "Tom". However, neither "Jack" nor "Tom" is part of the match results. |
| (?<!y)x  **Negative lookbehind assertion** | Matches "x" only if "x" is not preceded by "y". For example, /(?<!-)\d+/ matches a number only if it is not preceded by a minus sign. /(?<!-)\d+/.exec('3') matches "3". /(?<!-)\d+/.exec('-3') match is not found because the number is preceded by the minus sign. |