**Regular Expression Syntax**

Regular expression syntax includes the use of special characters (do not confuse with the [HTML special characters](https://www.tutorialrepublic.com/html-tutorial/html-entities.php)). 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.

The following sections describe the various options available for formulating patterns:

**Character Classes**

Square brackets surrounding a pattern of characters are called a character class e.g. [abc]. 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]. Let's look at some examples of character classes:

| **RegExp** | **What it Does** |
| --- | --- |
| [abc] | Matches any one of the characters a, b, or c. |
| [^abc] | Matches any one character other than a, b, or c. |
| [a-z] | Matches any one character from lowercase a to lowercase z. |
| [A-Z] | Matches any one character from uppercase a to uppercase z. |
| [a-Z] | Matches any one character from lowercase a to uppercase Z. |
| [0-9] | Matches a single digit between 0 and 9. |
| [a-z0-9] | Matches a single character between a and z or between 0 and 9. |

**Repetition Quantifiers**

In the previous section we've learnt how to match a single character in a variety of fashions. But what if you want to match on more than one character? For example, let's say you want to find out words containing one or more instances of the letter p, or words containing at least two p's, and so on. This is where quantifiers come into play. With quantifiers you can specify how many times a character in a regular expression should match.

The following table lists the various ways to quantify a particular pattern:

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

**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.

| **RegExp** | **What it Does** |
| --- | --- |
| ^p | Matches the letter p at the beginning of a line. |
| p$ | Matches the letter p at the end of a line. |

**Email Validation Example:**

<?php

$string="ram@gmail.com";

$pattern = "[a-zA-Z0-9]+@[a-zA-Z0-9]+.[a-z0-9A-Z]";

// Returns true if "abc" is found anywhere in $string.

$result=ereg($pattern, $string);

if($result){

echo 'Email Valid';

}

else{

echo 'unsuccess';

}

?>