# Regular expression use cases

This article is meant to be a companion to the regular expressions resources article. Instead of telling you how to do certain things, this article aims to answer, “why would I want to do that?”

## Character classes

Syntax: `[aeiou]` or `[0-9]`

Matches a single character in a set.

These can be useful when you don’t need the full power of the OR (`|`) syntax, because you are only looking for a small variation. For example: `civili[sz]ation` will search for both the British and American spellings of the word “civilization”. It’s shorter than writing `civilisation|civilization`.

You can use ranges to look for digits: `[0-9]+`

You can even combine ranges. Here’s how you would search for a case-insensitive hexadecimal digit: `[0-9a-fA-F]`

## Negated character classes

Syntax: `[^aeiou]`

Matches a character NOT in a set.

Useful, for example, to detect non-alphanumeric characters: `[^0-9a-zA-Z]`

Here’s a more specific example, used to see if a word starts with an ‘s’ and is followed by a consonant: `s[^aeiou]` (this assumes you have already determined that the word is only made up of letters, as digits or other punctuation would also pass the test).

A word of caution: in the example above, you should NOT interpret `s[^aeiou]` to mean “match an s that isn’t followed by a vowel”. The two meanings are similar, but there are subtle differences.

The above example really means “match an s and the next character, provided the character exists and isn’t a vowel”.

## Negative lookahead

Syntax: `7(?!a)`

This doesn’t match anything. It’s simply a conditional check. Compare this to how `7$` character isn’t really matching a 7 followed by a newline, it’s simply checking that any potential matching ‘7’ is the last character of the line. `7(?!a)` tries to match a ‘7’, and the `(?!a)` part checks to make sure that there is no ‘a’ character after the ‘7’.

In the negated character classes example above, I told you what it isn’t used for. Well, if you actually want to match “an s that isn’t followed by a vowel”, here’s how you do it: `s(?![aeiou])`

Do you see the difference? With the negative lookahead, the character after ‘s’ isn’t included in the match. So if you are doing a replace, the character after ‘s’ won’t be touched. In addition, the negative lookahead version will match an ‘s’ if it’s the last character of the line, which the character class negation will not catch.

---

Revision #3
Created Tue, Jun 14, 2016 12:08 AM by Rozett, Keith
Updated Tue, Mar 14, 2017 1:55 AM by Rozett, Keith