Shorthand Character Classes

Since certain character classes are used often, a series of shorthand character classes are available. `\d` is short for `[0-9]`. In most flavors that support Unicode, `\d` includes all digits from all scripts. Notable exceptions are Java, JavaScript, and PCRE. These Unicode flavors match only ASCII digits with `\d`.

`\w` stands for "word character". It always matches the ASCII characters `[A-Za-z0-9_]`. Notice the inclusion of the underscore and digits. In most flavors that support Unicode, `\w` includes many characters from other scripts. There is a lot of inconsistency about which characters are actually included. Letters and digits from alphabetic scripts and ideographs are generally included. Connector punctuation other than the underscore and numeric symbols that aren't digits may or may not be included. XML Schema and XPath even include all symbols in `\w`. Again, Java, JavaScript, and PCRE match only ASCII characters with `\w`.

`\s` stands for "whitespace character". Again, which characters this actually includes, depends on the regex flavor. In all flavors discussed in this tutorial, it includes `[ \t\r\n\f]`. That is: `\s` matches a space, a tab, a carriage return, a line feed, or a form feed. Most flavors also include the vertical tab, with Perl (prior to version 5.18) and PCRE (prior to version 8.34) being notable exceptions. In flavors that support Unicode, `\s` normally includes all characters from the Unicode "separator" category. Java and PCRE are exceptions once again. But JavaScript does match all Unicode whitespace with `\s`.

Shorthand character classes can be used both inside and outside the square brackets. `\s\d` matches a whitespace character followed by a digit. `[\s\d]` matches a single character that is either whitespace or a digit. When applied to `1 + 2 = 3`, the former regex matches ` 2` (space two), while the latter matches `1` (one). `[\da-fA-F]` matches a hexadecimal digit, and is equivalent to `[0-9a-fA-F]` if your flavor only matches ASCII characters with `\d`.

Negated Shorthand Character Classes

The above three shorthands also have negated versions. `\D` is the same as `[^\d]`, `\W` is short for `[^\w]` and `\S` is the equivalent of `[^\s]`.

Be careful when using the negated shorthands inside square brackets. `[\D\S]` is *not* the same as `[^\d\s]`. The latter matches any character that is neither a digit nor whitespace. It matches `x`, but not `8`. The former, however, matches any character that is either not a digit, or is not whitespace. Because all digits are not whitespace, and all whitespace characters are not digits, `[\D\S]` matches any character; digit, whitespace, or otherwise.

More Shorthand Character Classes

While support for `\d`, `\s`, and `\w` is quite universal, there are some regex flavors that support additional shorthand character classes. Perl 5.10 introduced `\h` and `\v`. `\h` matches horizontal whitespace, which includes the tab and all characters in the "space separator" Unicode category. It is the same as `[\t\p{Zs}]`. `\v` matches "vertical whitespace", which includes all characters treated as line breaks in the Unicode standard. It is the same as `[\n\cK\f\r\x85\x{2028}\x{2029}]`.

PCRE also supports `\h` and `\v` starting with version 7.2. PHP does as of version 5.2.2, Java as of version 8, and the JGsoft engine as of version 2. Boost supports `\h` starting with version 1.42. No version of Boost supports `\v` as a shorthand.

In many other regex flavors, `\v` matches only the vertical tab character. Perl, PCRE, and PHP never supported this, so they were free to give `\v` a different meaning. Java 4 to 7 and JGsoft V1 did use `\v` to match only the vertical tab.

Java 8 and JGsoft V2 changed the meaning of this token anyway. The vertical tab is also a vertical whitespace character. To avoid confusion, the above paragraph uses `\cK` to represent the vertical tab.

Ruby 1.9 and later have their own version of `\h`. It matches a single hexadecimal digit just like `[0-9a-fA-F]`. `\v` is a vertical tab in Ruby.

XML Character Classes

XML Schema, XPath, and JGsoft V2 regular expressions support four more shorthands that aren't supported by any other regular expression flavors. `\i` matches any character that may be the first character of an XML name. `\c` matches any character that may occur after the first character in an XML name. `\I` and `\C` are the respective negated shorthands. Note that the `\c` shorthand syntax conflicts with the control character syntax used in many other regex flavors.

You can use these four shorthands both inside and outside character classes using the bracket notation. They're very useful for validating XML references and values in your XML schemas. The regular expression `\i\c*` matches an XML name like `xml:schema`.

The regex `<\i\c*\s*>` matches an opening XML tag without any attributes. `</\i\c*\s*>` matches any closing tag. `<\i\c*(\s+\i\c*\s*=\s*("[^"]*"|'[^']*'))*\s*>` matches an opening tag with any number of attributes. Putting it all together, `<(\i\c*(\s+\i\c*\s*=\s*("[^"]*"|'[^']*'))*|/\i\c*)\s*>` matches either an opening tag with attributes or a closing tag.

No other regex flavors discussed in this tutorial support XML character classes. If your XML files are plain ASCII , you can use `[_:A-Za-z]` for `\i` and `[-._:A-Za-z0-9]` for `\c`. If you want to allow all Unicode characters that the XML standard allows, then you will end up with some pretty long regexes. You would have to use `[:A-Z_a-z\u00C0-\u00D6\u00D8-\u00F6\u00F8-\u02FF\u0370-\u037D\u037F-\u1FFF\u200C-\u200D\u2070-\u218F\u2C00-\u2FEF\u3001-\uD7FF\uF900-\uFDCF\uFDF0-\uFFFD]` instead of `\i` and `[-.0-9:A-Z_a-z\u00B7\u00C0-\u00D6\u00D8-\u00F6\u00F8-\u037D\u037F-\u1FFF\u200C-\u200D\u203F\u2040\u2070-\u218F\u2C00-\u2FEF\u3001-\uD7FF\uF900-\uFDCF\uFDF0-\uFFFD]` instead of `\c`.

Make a Donation

Did this website just save you a trip to the bookstore? Please make a donation to support this site, and you'll get a **lifetime of advertisement-free access** to this site!

Page URL: https://www.regular-expressions.info/shorthand.html
Page last updated: 24 January 2020
Site last updated: 08 April 2020