Branch Reset Groups

Perl 5.10 introduced a new regular expression feature called a branch reset group. JGsoft V2 and PCRE 7.2 and later also support this, as do languages like PHP, Delphi, and R that have regex functions based on PCRE. Boost added them to its ECMAScript grammar in version 1.42.

Alternatives inside a branch reset group share the same capturing groups. The syntax is `(?|regex)` where `(?|` opens the group and `regex` is any regular expression. If you don't use any alternation or capturing groups inside the branch reset group, then its special function doesn't come into play. It then acts as a non-capturing group.

The regex `(?|(a)|(b)|(c))` consists of a single branch reset group with three alternatives. This regex matches either `a`, `b`, or `c`. The regex has only a single capturing group with number 1 that is shared by all three alternatives. After the match, `$1` holds `a`, `b`, or `c`.

Compare this with the regex `(a)|(b)|(c)` that lacks the branch reset group. This regex also matches `a`, `b`, or `c`. But it has three capturing groups. After the match, `$1` holds `a` or nothing at all, `$2` holds `b` or nothing at all, while `$3` holds `c` or nothing at all.

Backreferences to capturing groups inside branch reset groups work like you'd expect. `(?|(a)|(b)|(c))\1` matches `aa`, `bb`, or `cc`. Since only one of the alternatives inside the branch reset group can match, the alternative that participates in the match determines the text stored by the capturing group and thus the text matched by the backreference.

The alternatives in the branch reset group don't need to have the same number of capturing groups. `(?|abc|(d)(e)(f)|g(h)i)` has three capturing groups. When this regex matches `abc`, all three groups are empty. When `def` is matched, `$1` holds `d`, `$2` holds `e` and `$3` holds `f`. When `ghi` is matched, `$1` holds `h` while the other two are empty.

You can have capturing groups before and after the branch reset group. Groups before the branch reset group are numbered as usual. Groups in the branch reset group are numbered continued from the groups before the branch reset group, which each alternative resetting the number. Groups after the branch reset group are numbered continued from the alternative with the most groups, even if that is not the last alternative. So `(x)(?|abc|(d)(e)(f)|g(h)i)(y)` defines five capturing groups. `(x)` is group 1, `(d)` and `(h)` are group 2, `(e)` is group 3, `(f)` is group 4, and `(y)` is group 5.

Named Capturing Groups in Branch Reset Groups

You can use named capturing groups inside branch reset groups. If you do, you should use the same names for the groups that will get the same numbers. Otherwise you'll get undesirable behavior in Perl or Boost. PowerGREP treats mismatched group names as an error. PCRE only reliably supports named groups inside branch reset groups starting with version 8.00. This means Delphi only does so starting with XE7 and PHP starting with version 5.2.14.

`(?'before'x)(?|abc|(?'left'd)(?'middle'e)(?'right'f)|g(?'left'h)i)(?'after'y)` is the same as the previous regex. It names the five groups "before", "left", "middle", "right", and "after". Notice that because the 3rd alternative has only one capturing group, that must be the name of the first group in the other alternatives.

If you omit the names in some alternatives, the groups will still share the names with the other alternatives. In the regex `(?'before'x)(?|abc|(?'left'd)(?'middle'e)(?'right'f)|g(h)i)(?'after'y)` the group `(h)` is still named "left" because the branch reset group makes it share the name and number of `(?'left'd)`.

In Perl, PCRE, and Boost, it is best to use a branch reset group when you want groups in different alternatives to have the same name. That's the only way in Perl, PCRE, and Boost to make sure that groups with the same name really are one and the same group.

In PowerGREP, groups with the same name are always treated as one and the same group. So you don't really need to use a branch reset group in PowerGREP when using named capturing groups.

Day and Month with Accurate Number of Days

It's time for a more practical example. These two regular expressions match a date in m/d or mm/dd format. They exclude invalid dates such as 2/31.

```
^(?:(0?[13578]|1[02])/(3[01]|[12][0-9]|0?[1-9]) # 31 days
 |  (0?[469]|11)/(30|[12][0-9]|0?[1-9])         # 30 days
 |  (0?2)/([12][0-9]|0?[1-9])                   # 29 days
 )$
```

The first version uses a non-capturing group `(?:…)` to group the alternatives. It has six separate capturing groups. `$1` and `$2` hold the month and the day for months with 31 days. `$3` and `$4` hold them for months with 30 days. `$5` and `$6` are only used for February.

```
^(?|(0?[13578]|1[02])/(3[01]|[12][0-9]|0?[1-9]) # 31 days
 |  (0?[469]|11)/(30|[12][0-9]|0?[1-9])         # 30 days
 |  (0?2)/([12][0-9]|0?[1-9])                   # 29 days
 )$
```

The second version uses a branch reset group `(?|…)` to group the alternatives and merge their capturing groups. The 4th character is the only difference between these two regexes. Now there are only two capturing groups. These are shared between the tree alternatives. When a match is found `$1` always holds the month and 2 always holds the day, regardless of the number of days in the month.

Make a Donation

Page URL: https://www.regular-expressions.info/branchreset.html
Page last updated: 22 November 2019
Site last updated: 08 April 2020