## Continuing at The End of The Previous Match

The anchor `\G` matches at the position where the previous match ended. During the first match attempt, `\G` matches at the start of the string in the way `\A` does.

Applying `\G\w` to the string `test string` matches `t`. Applying it again matches `e`. The 3rd attempt yields `s` and the 4th attempt matches the second `t` in the string. The fifth attempt fails. During the fifth attempt, the only place in the string where `\G` matches is after the second `t`. But that position is not followed by a word character, so the match fails.

## End of The Previous Match vs. Start of The Match Attempt

With some regex flavors or tools, `\G` matches at the start of the match attempt, rather than at the end of the previous match. This is the case with Ruby and the Just Great Software applications. In EditPad Pro `\G` matches at the position of the text cursor. When a match is found, EditPad Pro will select the match, and move the text cursor to the end of the match. The result is that `\G` matches at the end of the previous match result only when you do not move the text cursor between two searches. All in all, this makes a lot of sense in the context of a text editor.

The distinction between the end of the previous match and the start of the match attempt is also important if your regular expression can find zero-length matches. Most regex engines advance through the string after a zero-length match. In that case, the start of the match attempt is one character further in the string than the end of the previous match attempt. .NET, Java, and Boost advance this way and also match `\G` at the end of the previous match attempt. Thus `\G` fails to match when .NET, Java, and Boost have advanced after a zero-length match.

## \G Magic with Perl

In Perl, the position where the last match ended is a “magical” value that is remembered separately for each string variable. The position is not associated with any regular expression. This means that you can use `\G` to make a regex continue in a subject string where another regex left off.

If a match attempt fails, the stored position for `\G` is reset to the start of the string. To avoid this, specify the continuation modifier `/c`.

All this is very useful to make several regular expressions work together. E.g. you could parse an HTML file in the following fashion:

```
while ($string =~ m/</g) {
  if ($string =~ m/\GB>/c) {
    # Bold
  } elsif ($string =~ m/\GI>/c) {
    # Italics
  } else {
    # ...etc...
  }
}
```

The regex in the while loop searches for the tag’s opening bracket, and the regexes inside the loop check which tag we found. This way you can parse the tags in the file in the order they appear in the file, without having to write a single big regex that matches all tags you are interested in.

## \G in Other Programming Languages

This flexibility is not available with most other programming languages. E.g. in Java, the position for \G is remembered by the Matcher object. The Matcher is strictly associated with a single regular expression and a single subject string. What you can do though is to add a line of code to make the match attempt of the second Matcher start where the match of the first Matcher ended. Then \G will match at this position.

## Start of Match Attempt

Normally, \A is a start-of-string anchor. But in Tcl, the anchor \A matches at the start of the match attempt rather than at the start of the string. With the GNU flavors, \` does the same. This makes no difference if you're only making one call to `regexp` in Tcl or `regexec()` in the GNU library. It can make a difference if you make a second call to find another match in the remainder of the string after the first match. \A or \` then matches at the end of the first match, instead of failing to match as start-of-string anchors normally do. Strangely enough, the caret does not have this issue in either Tcl or GNU's library.

## Make a Donation



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