КАК СТАТЬ АВТОРОМ

4.45 Оценка

2365.67 Рейтинг

Selectel

IT-инфраструктура для бизнеса

Подписаться

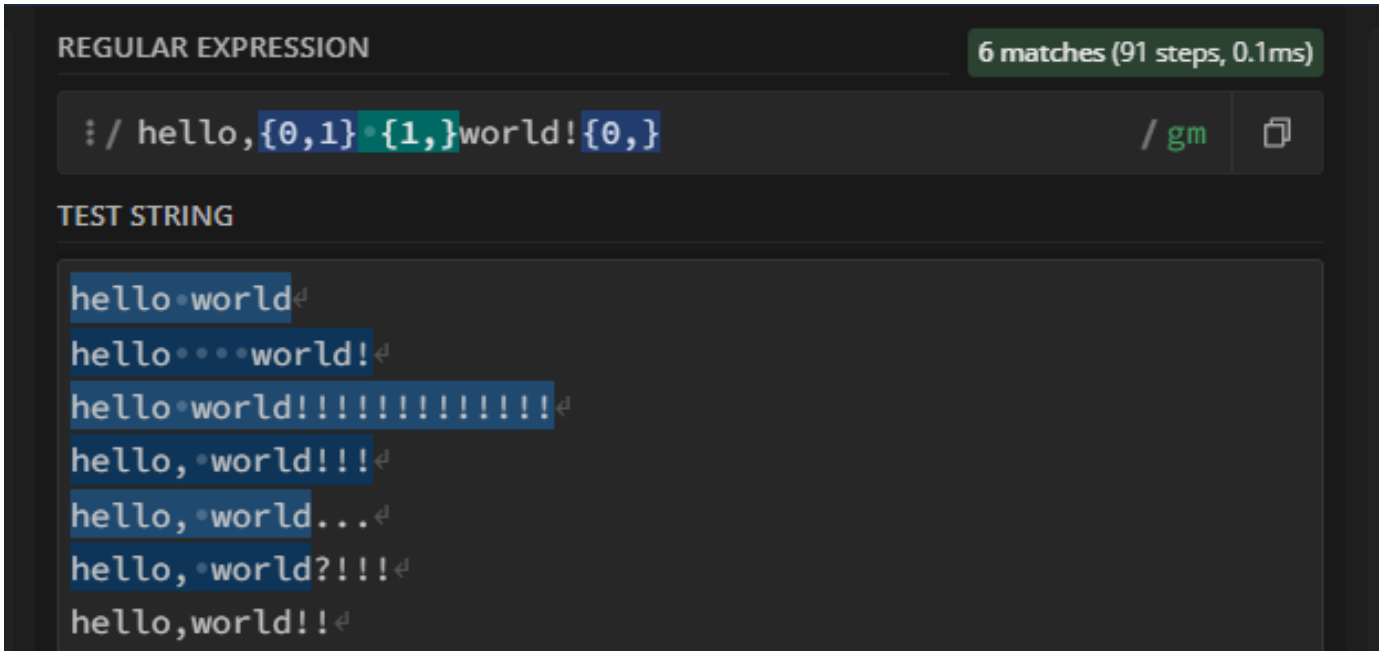
DandyDan
7 окт 2024 в 15:00

Регулярные выражения простыми словами. Часть 1

Простой 7 мин 52K

Блог компании Selectel, IT-стандарты*, Программирование*, Учебный процесс в IT

Туториал Технотекст 7



Разработчики делятся на два типа: тех, кто уже понимает регулярные выражения и порой решает сложные задачи одной строкой, и тех, кто все еще боится и всячески их избегает. Эта статья специально для вторых, чтобы им было проще стать первыми. Она либо поможет преодолеть «регекспофобию», либо усугубит ее. В любом случае, добро пожаловать под кат.

Используйте навигацию, если не хотите читать текст полностью:

Введение

Регулярное выражение описывает некоторый образец (на английском — pattern), которому текстовые строки могут или соответствовать, или нет. Основные области применения: поиск, валидация, парсинг и устрашение.

- **Поиск**. Найти все email-адреса в тексте~~, чтобы отправить им письма счастья~~.
- **Валидация**. Проверить, что введенный в форме email-адрес хотя бы отдаленно похож на настоящий.
- **Парсинг**. Разбить email-адрес на имя пользователя и домен.
- **Устрашение**. Наиболее полное регулярное выражение для валидации email-адресов можно посмотреть на этой странице.

Важно помнить, что хотя регулярные выражения и являются мощным инструментом, все-таки это не серебряная пуля и полнотой по Тьюрингу они не обладают. Следовательно, не все задачи можно решить с их помощью. Например, на Stack Overflow объясняется, почему ни в коем случае нельзя парсить HTML с помощью регулярных выражений.

С другой стороны, иногда регулярные выражения позволяют решать задачи, для которых они вообще не предназначены. Например, на этой странице описан крайне неэффективный, но работающий способ проверять число на простоту.

> Каждый раз писать «регулярное выражение» утомляет. В англоязычном сленге прижились термины regex и regexp. Однако оба звучат как имена злодеев из аниме, поэтому в этой статье я буду время от времени использовать слово «регулярки».
>
> Согласно легенде, «regular expressions» означает «обычные выражения», а регулярными их назвали, потому что поленились при переводе.
>
> Кстати, есть мнение, что термин regex не всегда может быть синонимом regular expression. Но иногда может. Подробности можно прочитать в короткой статье.

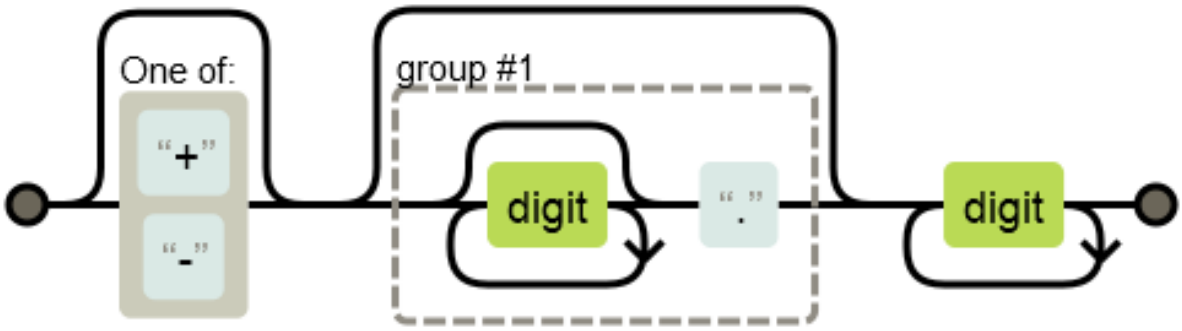
Инструменты

Безусловно, из всех своих задач регулярки лучше всего справляются именно с устрашением. Однако благодаря работе ведущих египтологов появились сервисы, помогающие расшифровать эти загадочные письмена. Я чаще всего пользуюсь двумя: один красивый, другой — полезный. На самом деле оба красивые и полезные.

Regexper позволяет превратить регулярку почти любой степени запутанности в красиво оформленный граф. Например, есть такая регулярка для вещественных чисел, изобретенная древними шумерами:

```
[+-]?(\d*\.)?\d+
```

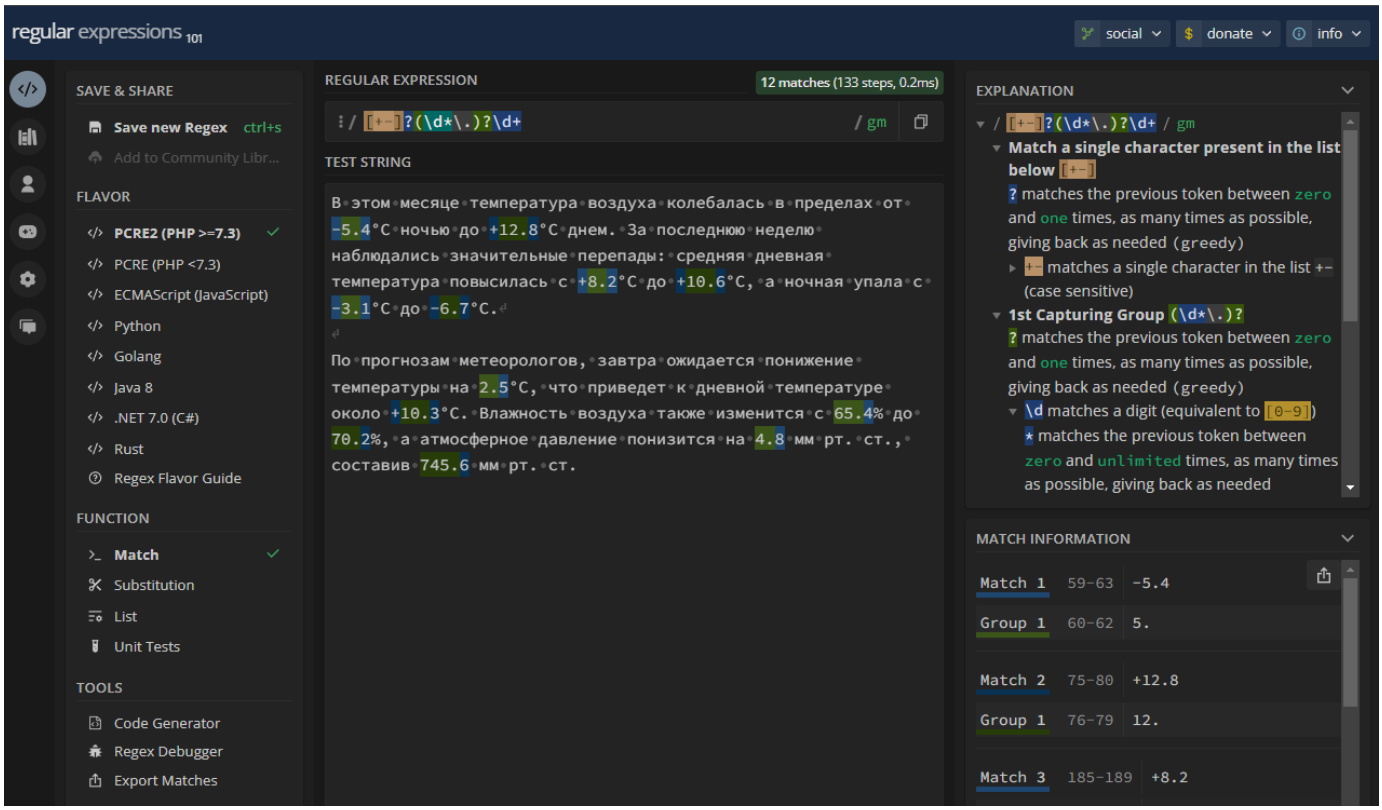
И вдруг она обретает смысл в виде интуитивно-понятной инфографики:



Если вы где-то откопали странную регулярку и хотите быстрее понять, что она делает, смело закидывайте в Regexper.

Правда, для особо запутанных случаев и граф будет непростым. Например, попробуйте отправить в сервис вышеупомянутую регулярку для email (правда, сперва нужно будет записать ее в одну строчку). Я не стал прикреплять картинку здесь, потому что она имеет 24 621 пикселей в ширину.

Второй крайне полезный ресурс — regex101. Помогает увидеть, как работает регулярка. А если она не работает или работает не так, то можно понять, в чем дело. В сервисе есть даже пошаговый дебаггер!

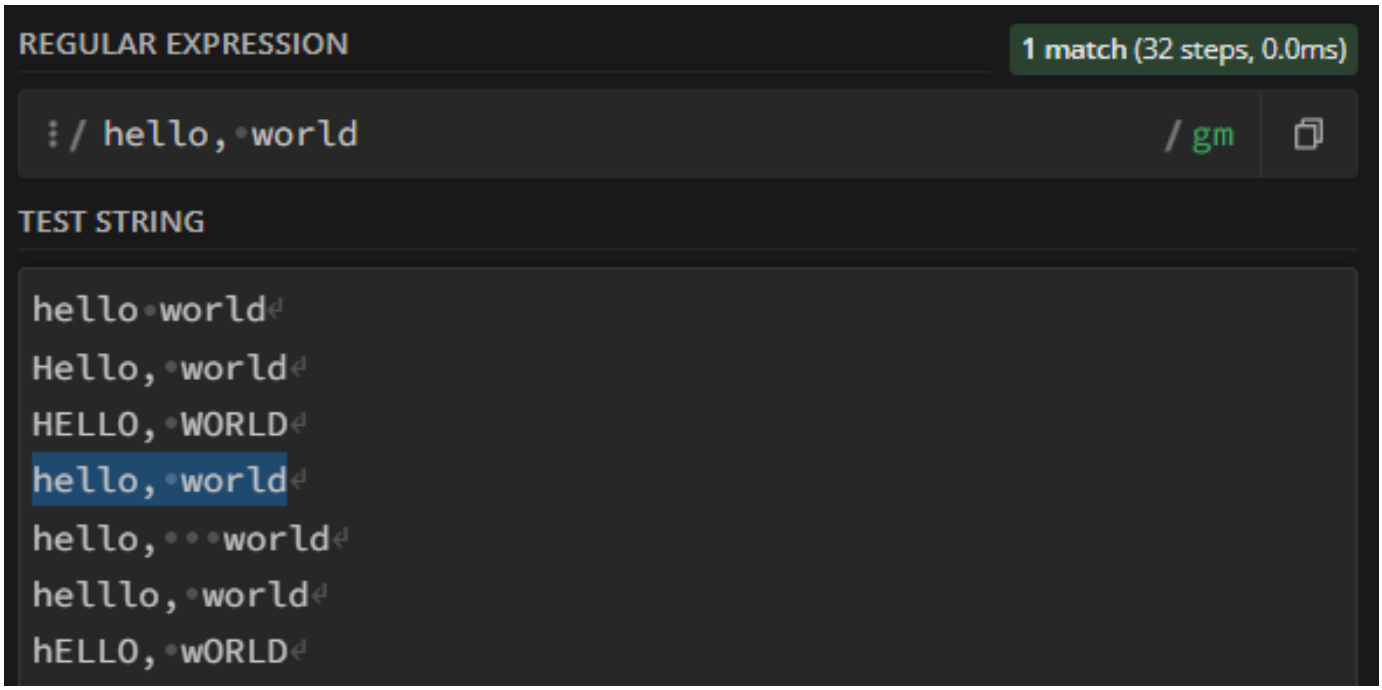


Hello, world

Хоть язык регулярных выражений и не является языком программирования, все-таки будем соблюдать традицию. Итак, открываем regex101 и вводим регулярное выражение:

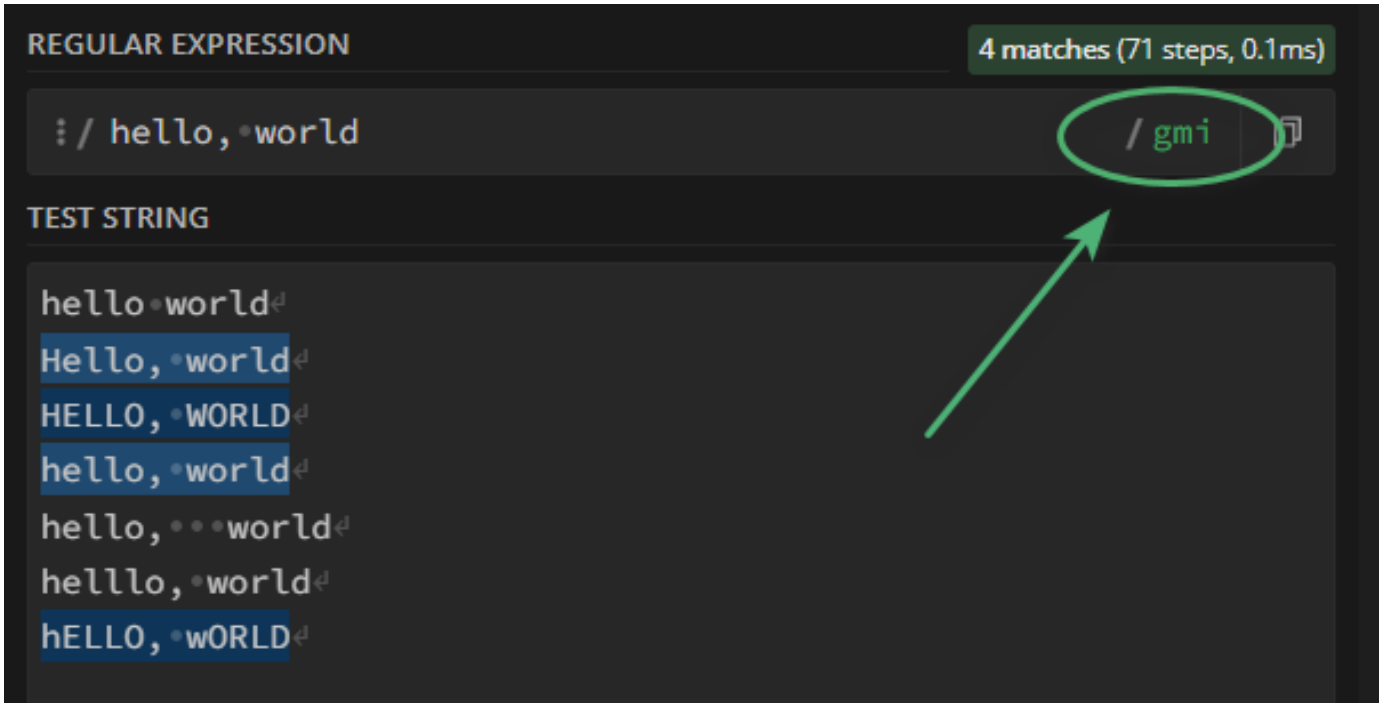
```
hello, world
```

Да-да, регулярка может выглядеть как человеческий текст, а не только как клингонский некролог. Регулярному выражению «hello, world» соответствует одна-единственная строка — «hello, world».



(В некоторых|Во многих) языках программирования регулярному выражению можно задавать опции. Если ей будет Case Insensitive (обычно обозначается как i), то подойдут все варианты регистра: «Hello world», «Hello World», «HELLO WORLD», «hElLo wOrLd» и т. д.

В regex101 для этого достаточно кликнуть по буквам с опциями справа от строки ввода.



Ни для кого не секрет, что в слове «hello» две буквы «l». Давайте этот занимательный факт запишем с помощью языка регулярных выражений:

```
hel{2}o, world
```

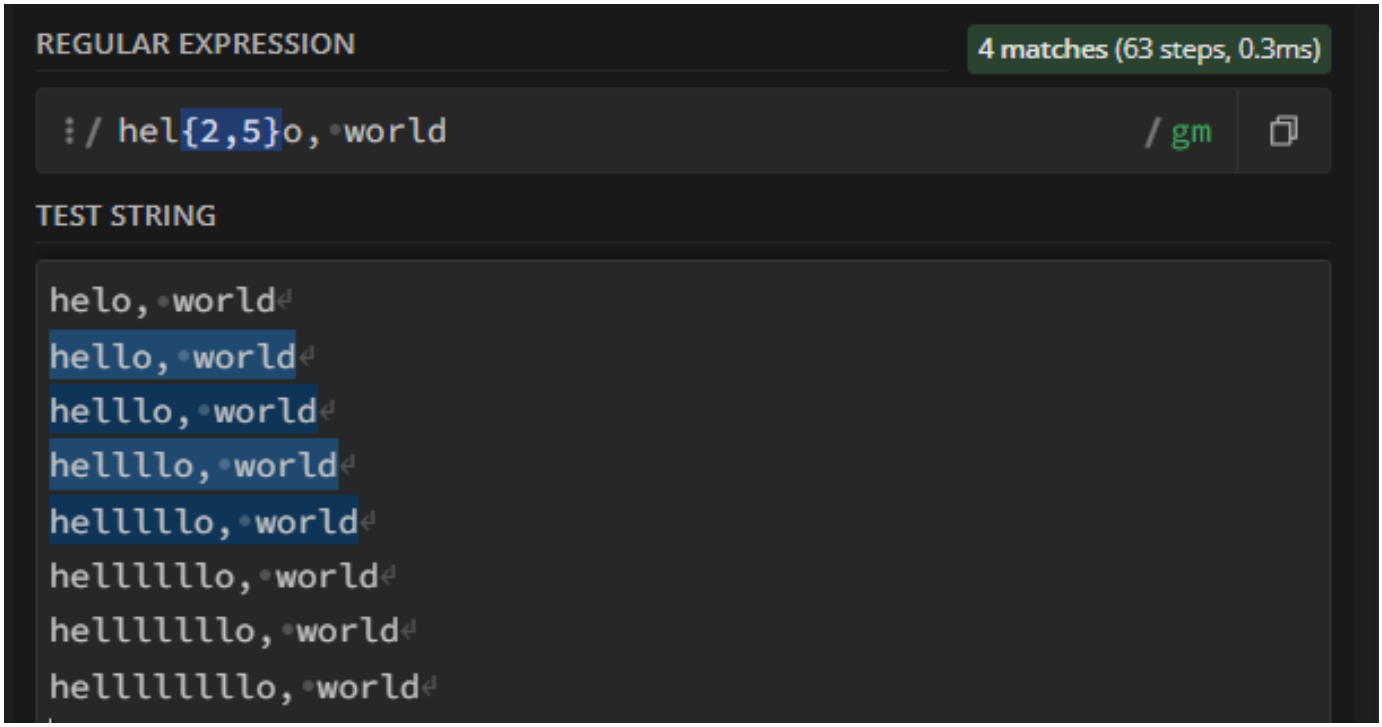
Это то же самое выражение, что и предыдущее, но теперь уже слегка пробуждает Ктулху. Обратите внимание, что {2} работает только на букву «l», а не на все буквы, стоящее ранее.

Зачем нам это нужно? Не проще ли было оставить просто «ll»? Безусловно, в данном случае проще. Однако в реальной практике могут быть более сложные ситуации. Во-первых, символ может повторяться не два раза, а, например, тысячу. Во-вторых, есть возможность применить операцию не на один символ, а на группу, но об этом в другой раз.

Кроме того, можно указать не только конкретное число повторений, но и пределы: минимум и максимум. Например:

```
hel{2,5}o, world
```

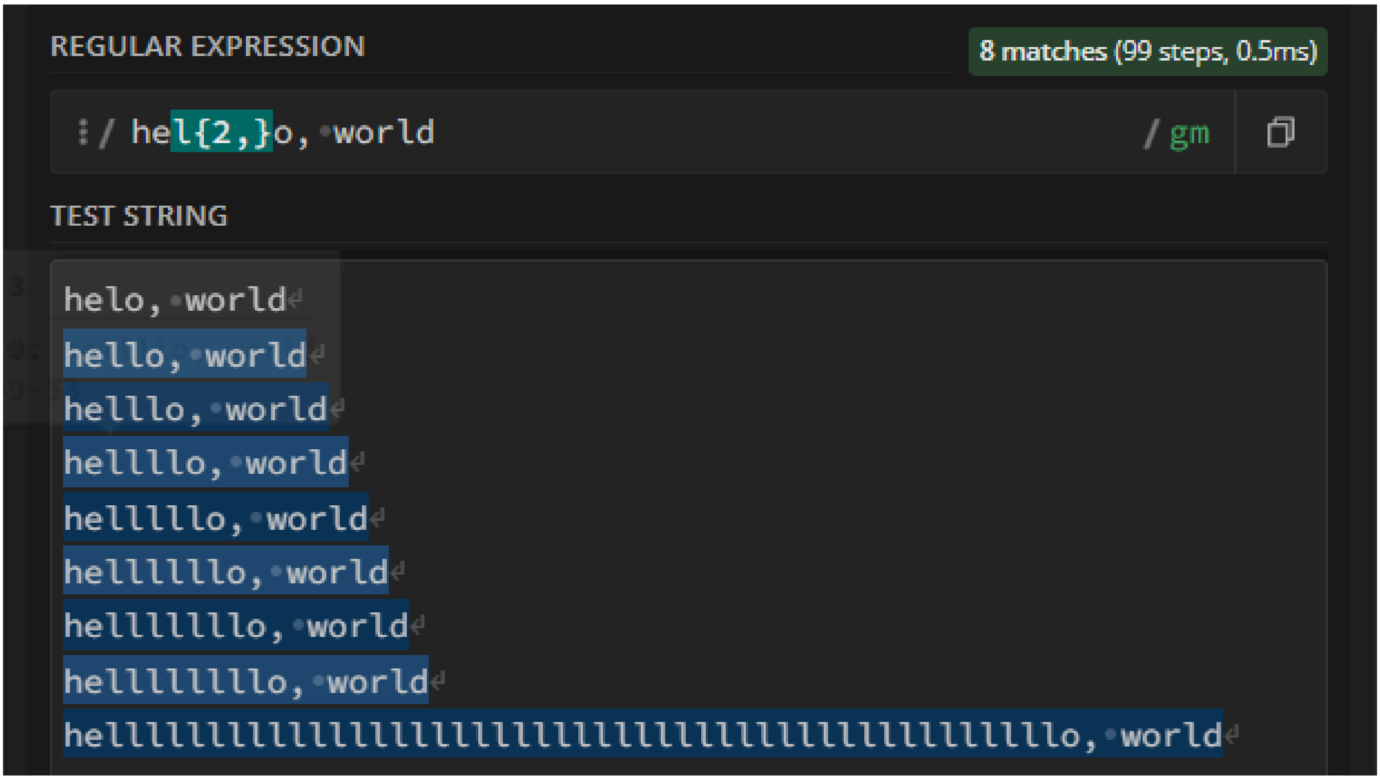
Такая запись означает, что буква «L» может повторяться от двух до пяти раз.



А если второе число опустить, но запятую оставить, это уже будет означать «до бесконечности».
Например:

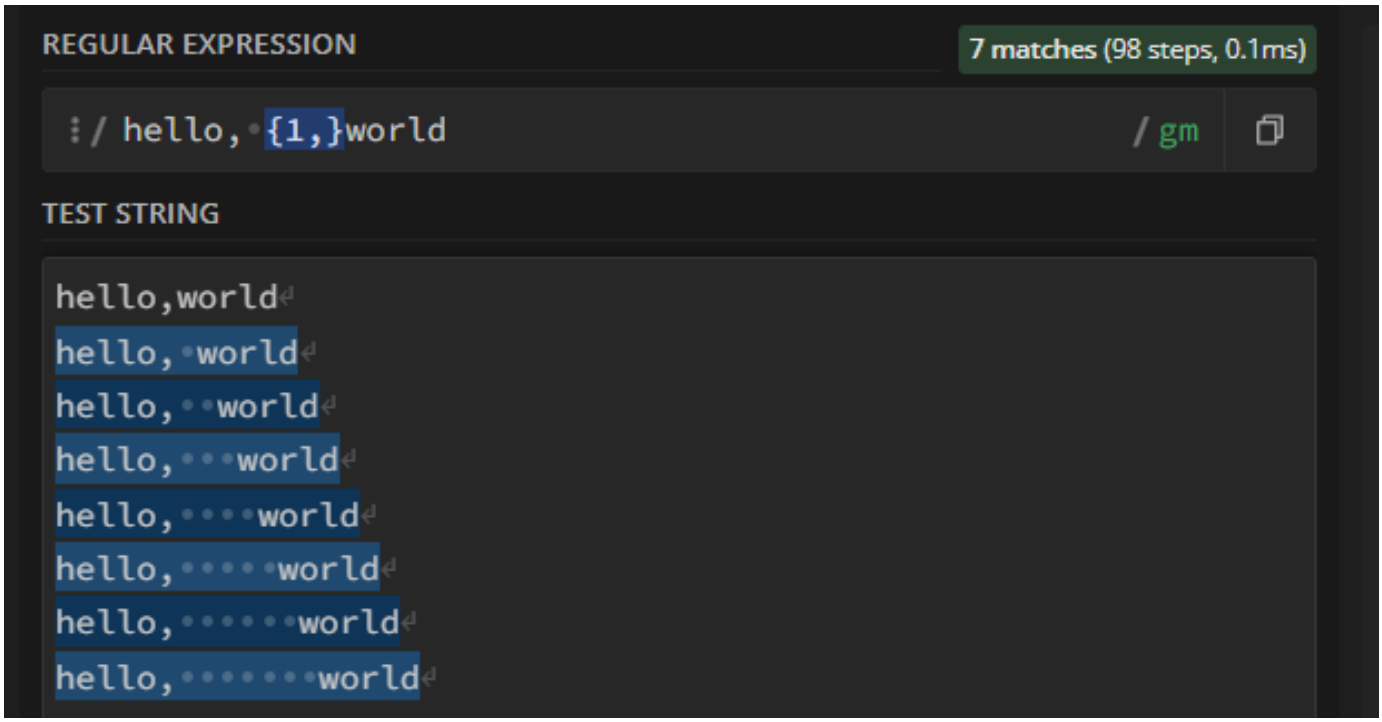
```
hel{2,}o, world
```

Такая запись уже означает, что в слове может быть любое количество букв «l», но не менее двух.



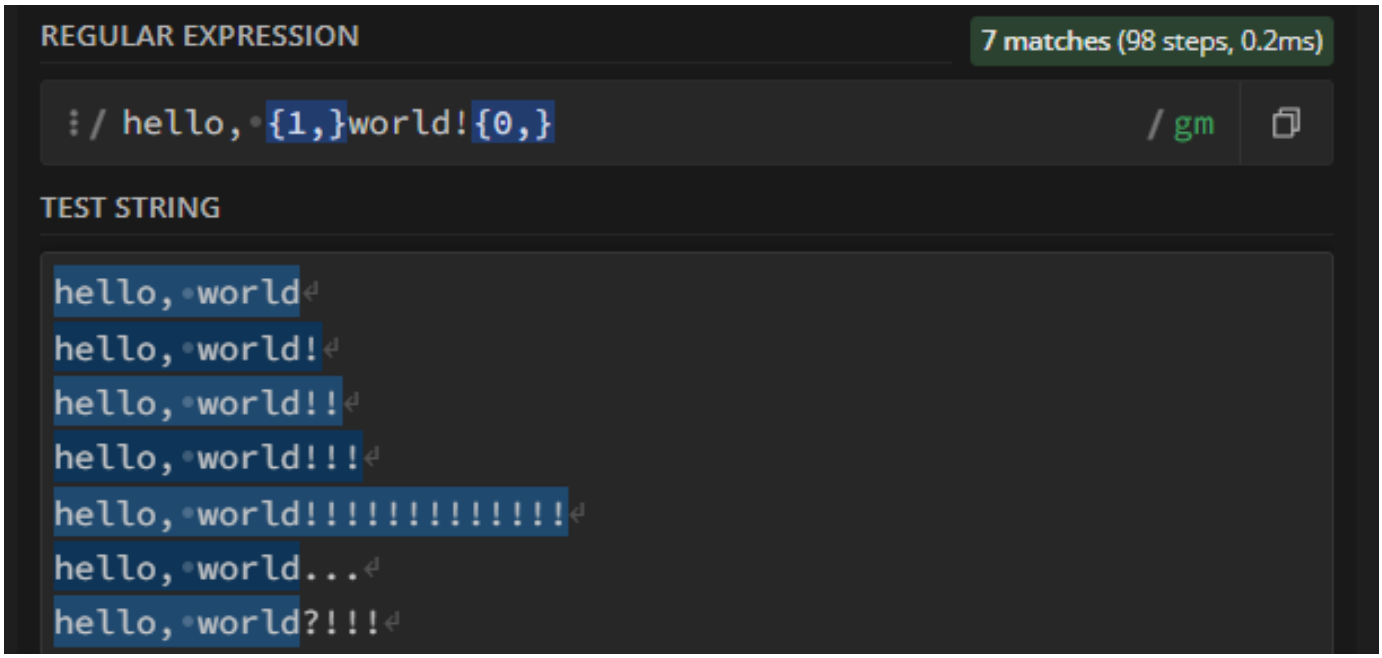
Оставим пока букву «l» в покое. В нашем примере вариативность по-настоящему пригодится, когда мы будем работать с пробелом. Допустим, нам нужно обрабатывать и те случаи, когда между словами не один пробел, а несколько:

```
hello, {1,}world
```



Срочная правка от заказчика: в конце фразы могут стоять восклицательные знаки, может быть даже сразу три. А может быть бесконечность, а может быть ни одного. Так и запишем:

```
hello, {1,}world!{0,}
```



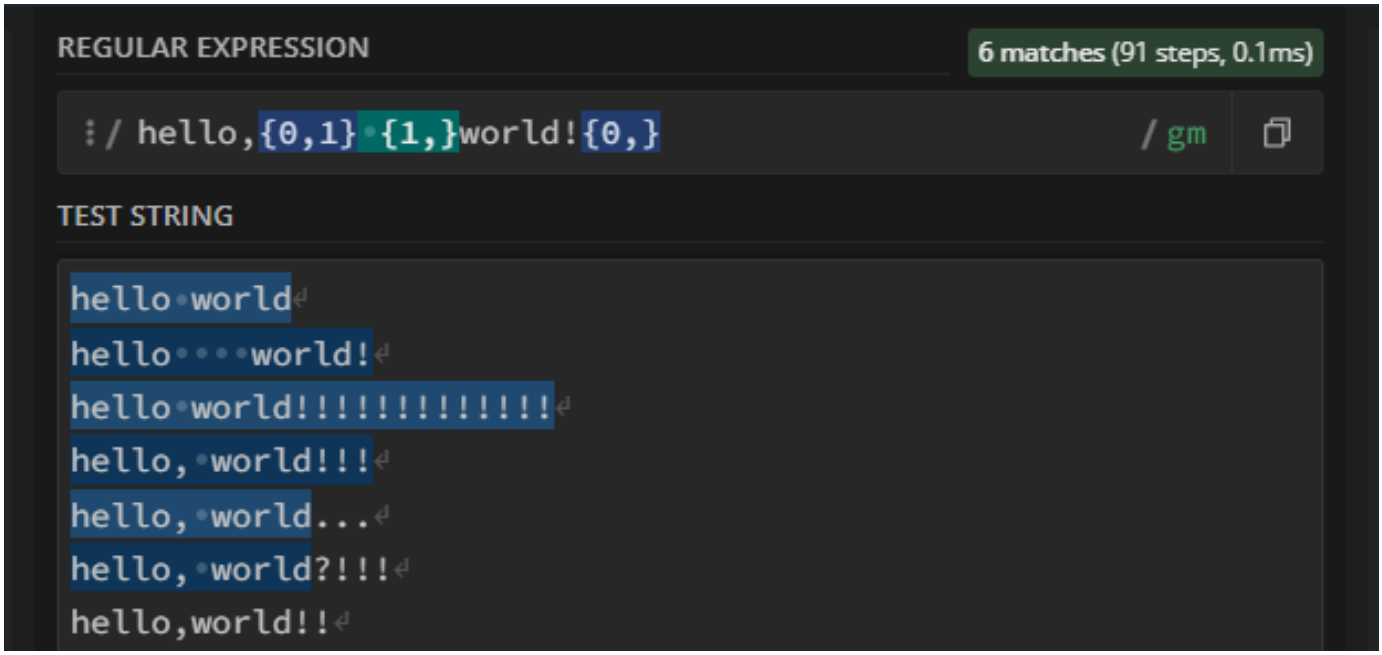
Обратите внимание на последние две строчки. Если начинаются символы, не соответветствующие образцу, это не отбраковывает всю строку целиком, а только ту часть, которая не подходит. Иногда это здорово, а иногда это совсем не то, что нам нужно. Что с этим делать, обсудим позже.

Раз уж зашла речь про знаки препинания: а что, если кто-то забудет поставить запятую, а мы все равно хотим найти такие варианты? Значит, запятая у нас должна встречаться или ноль, или один раз:

```
hello,{0,1} {1,}world!{0,}
```

Вот теперь это похоже на друидскую бухгалтерию, как и положено хорошей регулярке!{3,} Она уже достойна того, чтобы посмотреть ее в Regexper:

1. Берем слово «hello».
2. Запятую или берем, или обходим стороной.
3. Пробел один берем точно, затем крутимся по нему любое число раз.
4. Берем слово «world».
5. Восклицательный знак или обходим стороной, или крутимся по нему.



Обратите внимание, что строка, где вообще нет пробела, не соответствует образцу. Дело в том, что мы указали, что хотя бы один пробел должен быть. Разберем этот случай чуть позже.

Спецсимволы

Чтобы не переполнять регулярку цифрами, ее можно переполнить спецсимволами. Для некоторых наиболее распространенных вариантов есть сокращенный способ записи:

{0,1}	?	Символ или есть, или нет
{0,}	*	Символ встречается любое количество раз, включая 0
{1,}	+	Символ встречается не менее одного раза

Перепишем выражение короче:

```
hello,? +world!*
```

Теперь выражение выглядит почти как изначальное «hello, world», но насколько функциональнее оно стало!

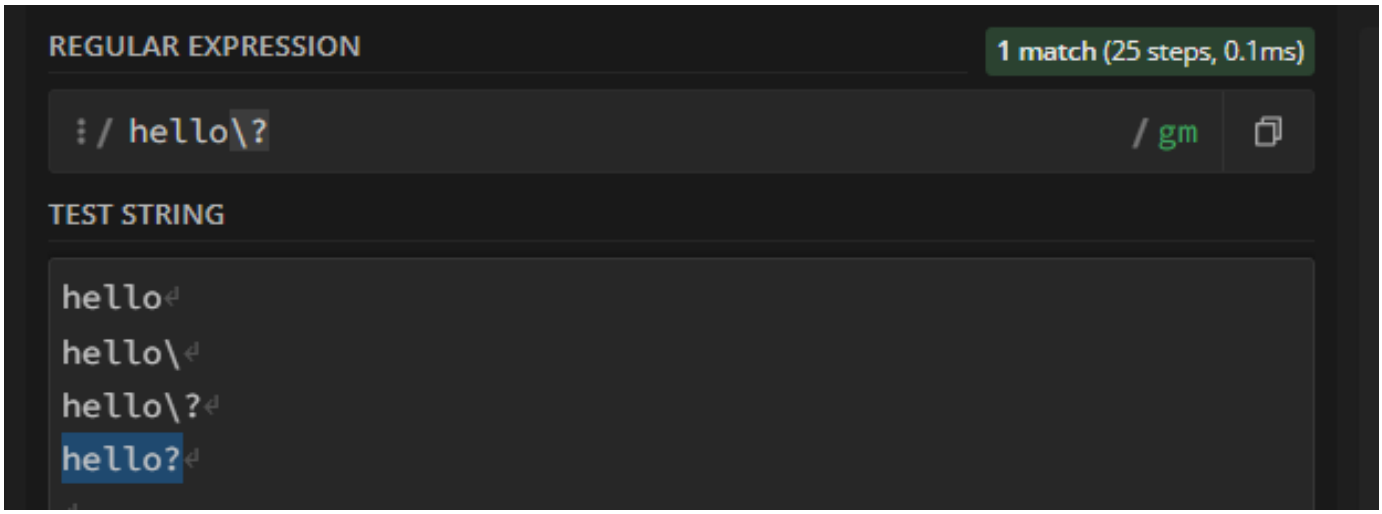
> Кстати, регулярное выражение, которому соответствуют оба имени злодеев англоязычных термина — regexp?

А что, если нужно воспринимать спецсимволы как обычные: плюс как плюс, вопросик как вопросик? Мы можем их экранировать с помощью escape-символов (в англоязычной литературе это называется escape character). Косая черта перед спецсимволом означает, что он теряет сверхспособности и становится обычным. Такой криптонит в мире регулярок.

Пример:

```
hello\?
```

Соответствует строке «hello?», а вовсе не «hello\» и «hello».

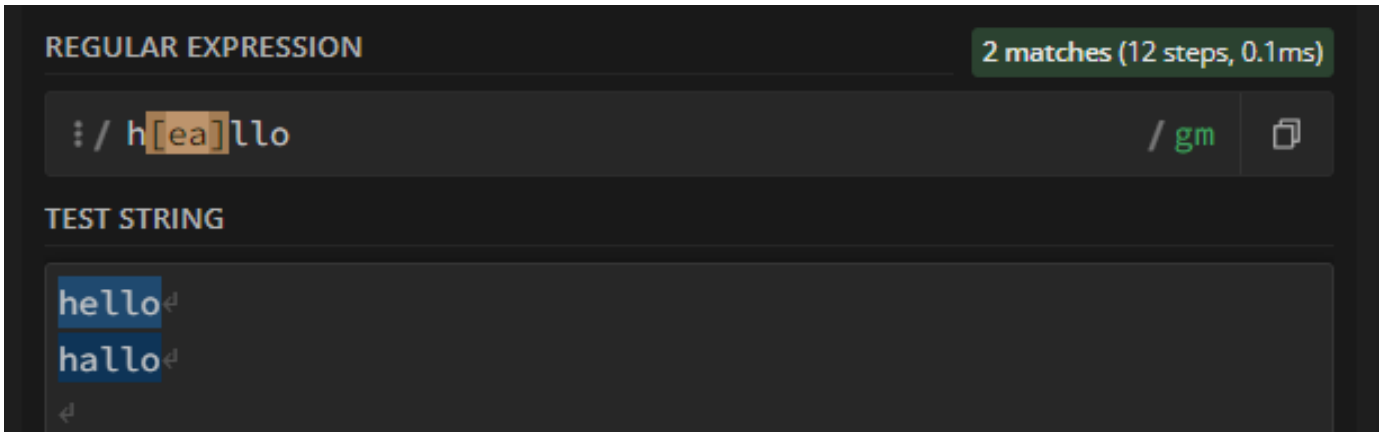


Косая черта может экранировать и саму себя. Чтобы это сработало, ее нужно записать дважды: \\

> В некоторых языках программирования косая черта уже является escape-символом в строке, поэтому, чтобы она стала escape-символом регулярного выражения, нужно написать \\. А чтобы она стала просто косой чертой, нужно написать \\\\. Да, чтобы косая черта осталась самой собой, ее нужно повторить четыре раза. Расскажите это людям, которые не понимают, за что айтишникам столько платят.

Возвращаемся к «hello, world». Кто-то может случайно написать не «hello», а «hallo». Давайте этот случай тоже обработаем.

Чтобы выбрать один символ из нескольких, нужно перечислить их внутри квадратных скобок. Выражению h[ea]llo соответствуют и «hello», и «hallo».

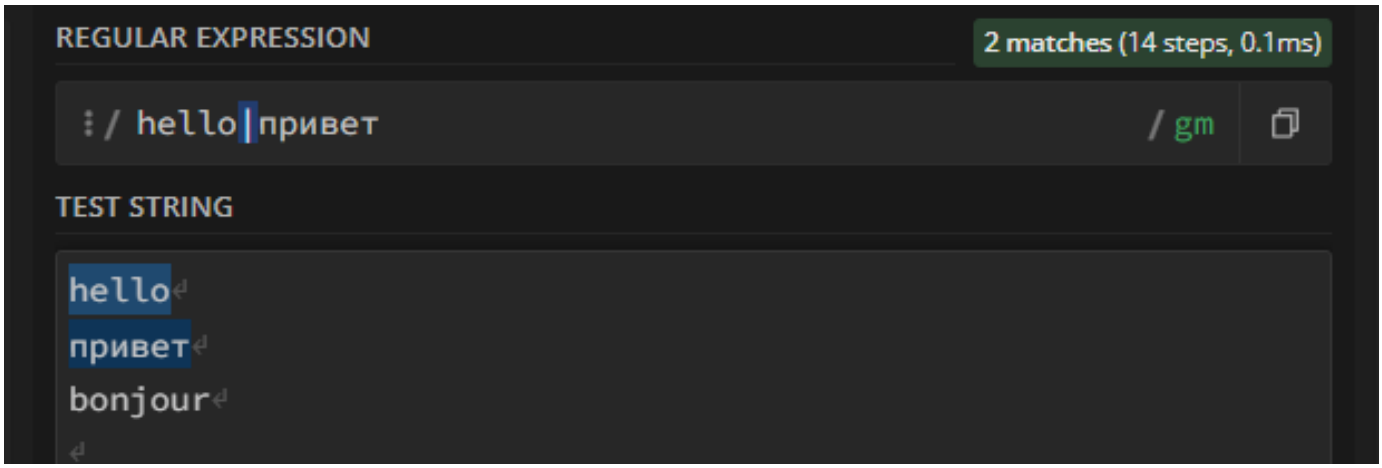


Раз уж мы заговорили о других языках, давайте и о русском подумаем. Как сделать, чтобы подходило и «hello», и «привет»?

Если написать [helloпривет], это выражение будет соответствовать просто одному символу из набора h,e,l,o, п, р, и, в, е, т. То есть одна буква подходит, а все слово — уже нет. Как

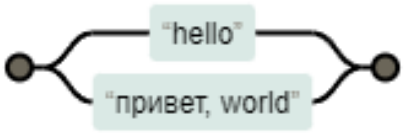
быть?

Для этого есть спецсимвол |. Запишем hello|привет.



Стоп, а в регулярках можно использовать кириллицу? Конечно! Хоть емоджи. Хотя, конечно, это зависит от приложения или языка программирования, в которым вы будете их использовать, но прямого запрета на этот счет нет.

Если написать hello|привет, world, то слово «world» останется в правой части. То есть мы получим или «hello», или «привет, world».

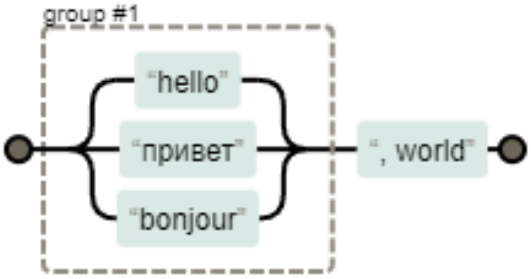


А можно «world» как-то оставить общим (хоть это и странно будет смотреться в сочетании с приветом)? Чтобы ограничить действие вертикальной черты, можно использовать круглые скобки. Вообще, это далеко не единственная, и более того — не главная задача круглых скобок, но об этом позже. А пока:

```
(hello|привет), world
```

Можно писать сколько угодно вариантов:

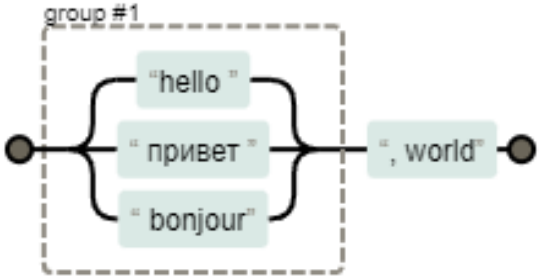
```
(hello|привет|bonjour), world
```

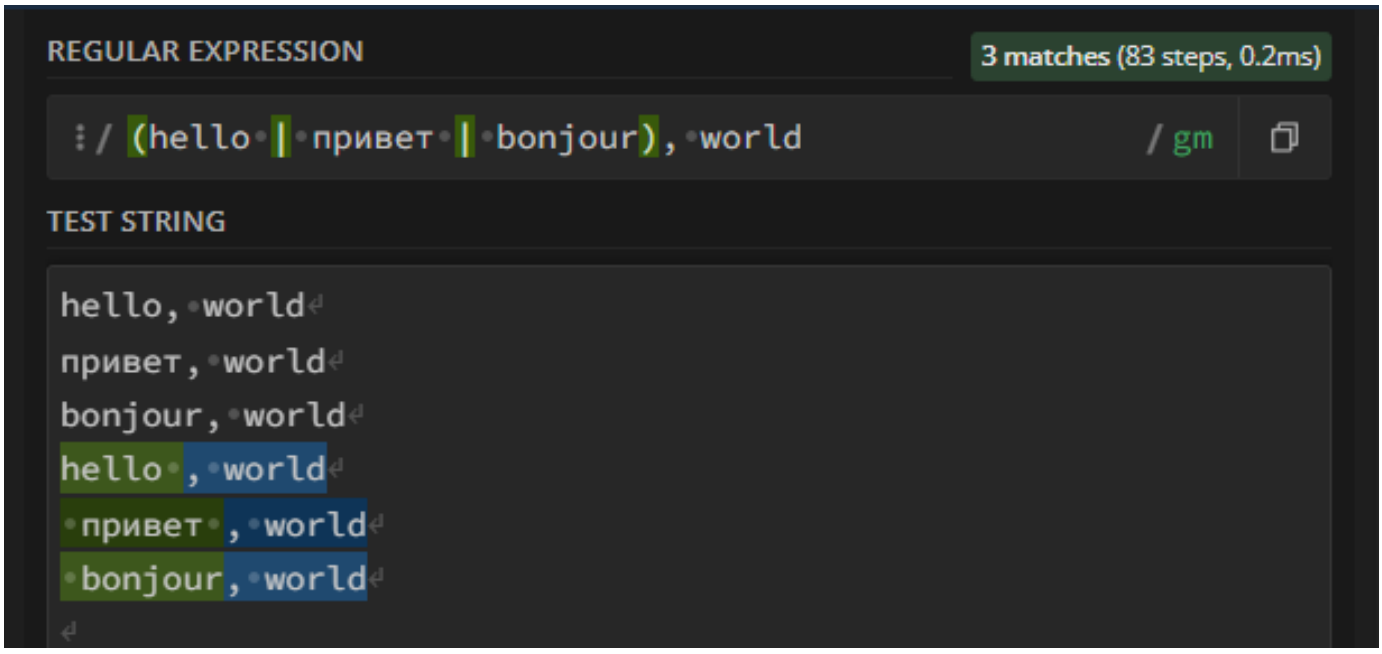


Обратите внимание, что можно поставить пробелы вокруг прямой черты, чтобы было красиво:

```
(hello | привет | bonjour), world
```

Но в таком случае изменится и выражение, поскольку пробелы не просто для красоты, а являются частью выражения. Поэтому не вставляйте пробелы, если не уверены, что они нужны.





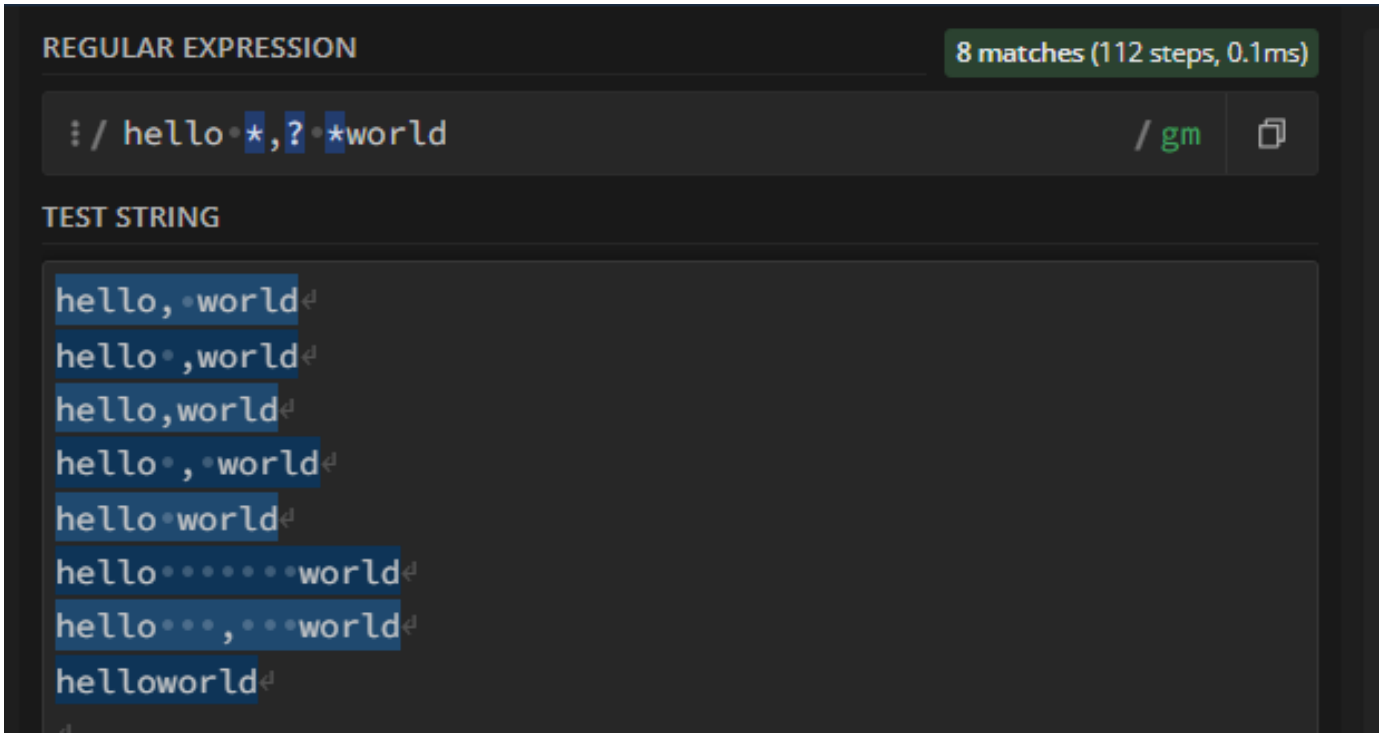
Кстати, по поводу пробелов. Есть люди, которые не ставят пробел после запятой, а есть те, кто ставят перед ней. Как сказал бы Шелдон Купер, в мире не хватит ромашкового чая,

чтобы унять ярость в моей груди. Но что поделать? Такие случаи тоже возможно нужно обрабатывать.

Попробуем такой вариант:

```
hello *,? *world
```

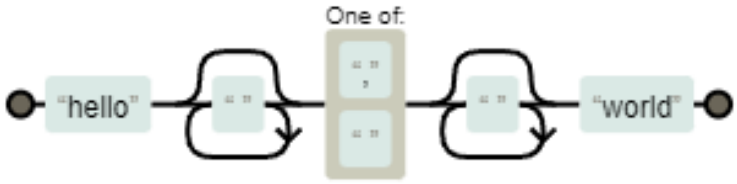
Работает.



Но есть две проблемы. Первая — строчка «helloworld». Так как у нас пробелы со звездочкой и запятая под вопросом, то подходят даже те варианты, в которых вообще нет разделителей. В тренировочном примере ничего страшного, но в реальной практике это распространенная проблема: как сделать, чтобы среди множества опциональных вариантов хотя бы один все-таки присутствовал? Это можно сделать разными способами: квадратным и круглым.

Квадратный способ

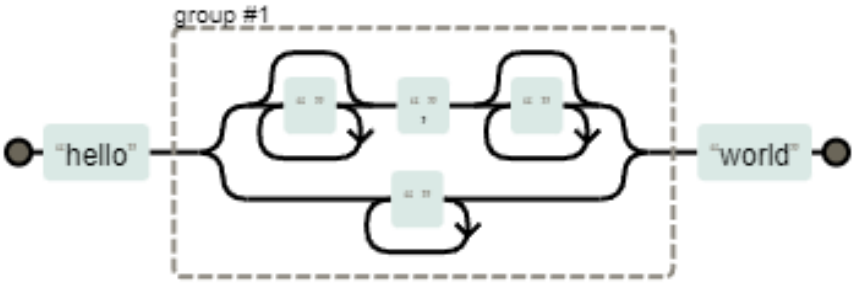
```
hello *[, ] *world
```



После «hello» могут идти пробелы (возможно, ни одного), затем точно должен быть какой-то разделитель (запятая или пробел), затем — еще пачка пробелов, возможно пустая.

Круглый способ

```
hello( *, *| +)world
```



Здесь у нас есть два варианта на выбор:

- запятая, с которой могут быть пробелы (а могут и не быть);
- пробелы, не менее одного.

Какой из способов более предпочтительный, зависит от конкретного случая.

Первую проблему решили, но есть вторая: в самом выражении пробел недостаточно ярко выражен. Не сразу очевидно, что звездочка относится именно к нему.

Помните, мы говорили, что косая черта превращает спецсимволы в простых смертных? Так вот, наоборот тоже работает. Если поставить косую черту перед простым магловским символом, он обретет сверхспособности. Конечно, не с каждым это работает, только с теми, у кого достаточно мидихлориан.

Например, буква s прекрасна тем, что с нее начинается слово space, поэтому \s — это спецсимвол, обозначающий пробел. И не только пробел, а вообще все пробелообразное:

Tab и при определенных опциях перенос строк.

Перепишем наше выражение:

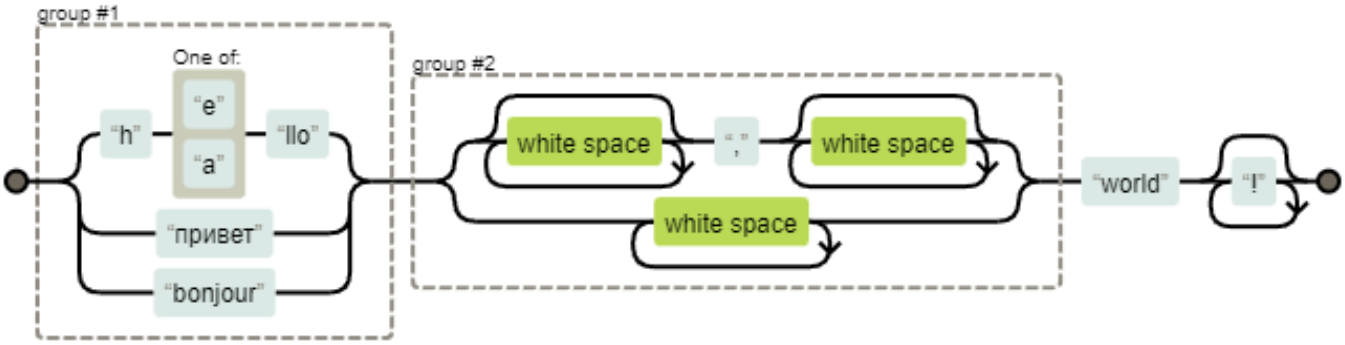
```
hello(\s*,\s*|\s+)world
```

Уже начинает походить на древнеегипетскую глаголицу, как и положено достойному регулярному выражению. Но при этом мы все еще примерно понимаем, что оно делает! В этом главный секрет регулярок — кто их пишет, тот их и понимает.

Заглядываем в бездну

Теперь соберем все вместе:

```
(h[ea]llo|привет|bonjour)(\s*,\s*|\s+)world!*
```



Уровень оккультизма достиг приемлемого.

В статье присутствует от нуля до бесконечности неточностей и упрощений. Некоторые из них допущены умышленно, чтобы сохранить развлекательный дух статьи и не отпугнуть вас излишней педантичностью и академичностью.

В следующей публикации рассмотрим различные возможности регулярных выражений более подробно.

Теги: selctel, regex, regexp, регулярные выражения, oarticle

Хабы: Блог компании Selectel, IT-стандарты, Программирование, Учебный процесс в IT

Selectel

IT-инфраструктура для бизнеса

ВКонтакте Telegram Сайт

52 Карма 0.1 Рейтинг

Даниил Тутубалин @DandyDan

Пользователь

Подписаться

Комментарии 57

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

Maximov_psy
12 часов назад

Что я узнал, проконсультировав 100 айтишников

12 мин 8.4K

+44 49 53

ru_vds
14 часов назад

Как защищают фильмы и доставляют их в кинотеатры

Средний 12 мин 3.2K

Обзор Перевод

+36 30 6

MrSotnik
17 часов назад

Новый язык от 1С: Зачем? Кому? Стоит ли лезть?

5 мин 21K

+36 34 76

RationalAnswer
22 часа назад

Тотальный блэкаут на юге Европы, а также чудеса нейро-лизоблюдия от ChatGPT

11 мин 13K

Дайджест

+34 17 70

alizar
18 часов назад

Ян Лекун, создатель LeNet, формата DjVu и адвокат опенсорса

Средний 7 мин 1.8K

Обзор

+28 13 6

El_Gato_Grande
18 часов назад

Как ИИ-контент проклял интернет и почему это закономерно

8 мин 4.3K

+25 10 9

BaDInMe
18 часов назад

ML-обработка видео в web-браузере для видеоконференций SaluteJazz

Средний 14 мин 237

Кейс

+21 5 0

full_moon
17 часов назад