

Подпишитесь на нас в Telegram



admin

26 декабря 2017



15



1



0



0

6 пунктов, которые помогут легко разобраться с регехр

Давно хотели изучить регехр? Это небольшое руководство поможет разобраться с ними в 6 этапов, а обилие примеров позволит закрепить материал.



15

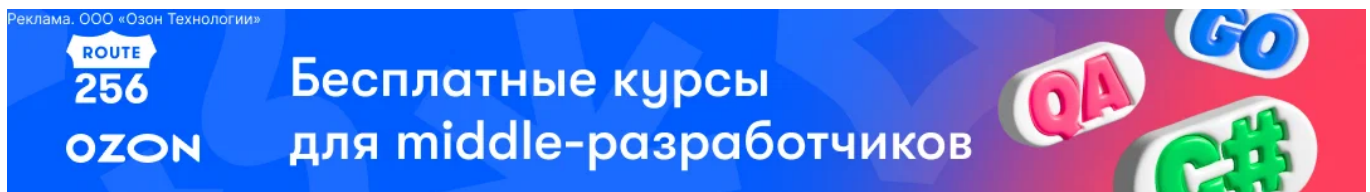


3



8

Реклама. ООО «Озон Технологии»



Что такое регехр?

Регехр представляет собой группу символов или знаков, которая используется для поиска определенного текстового шаблона.

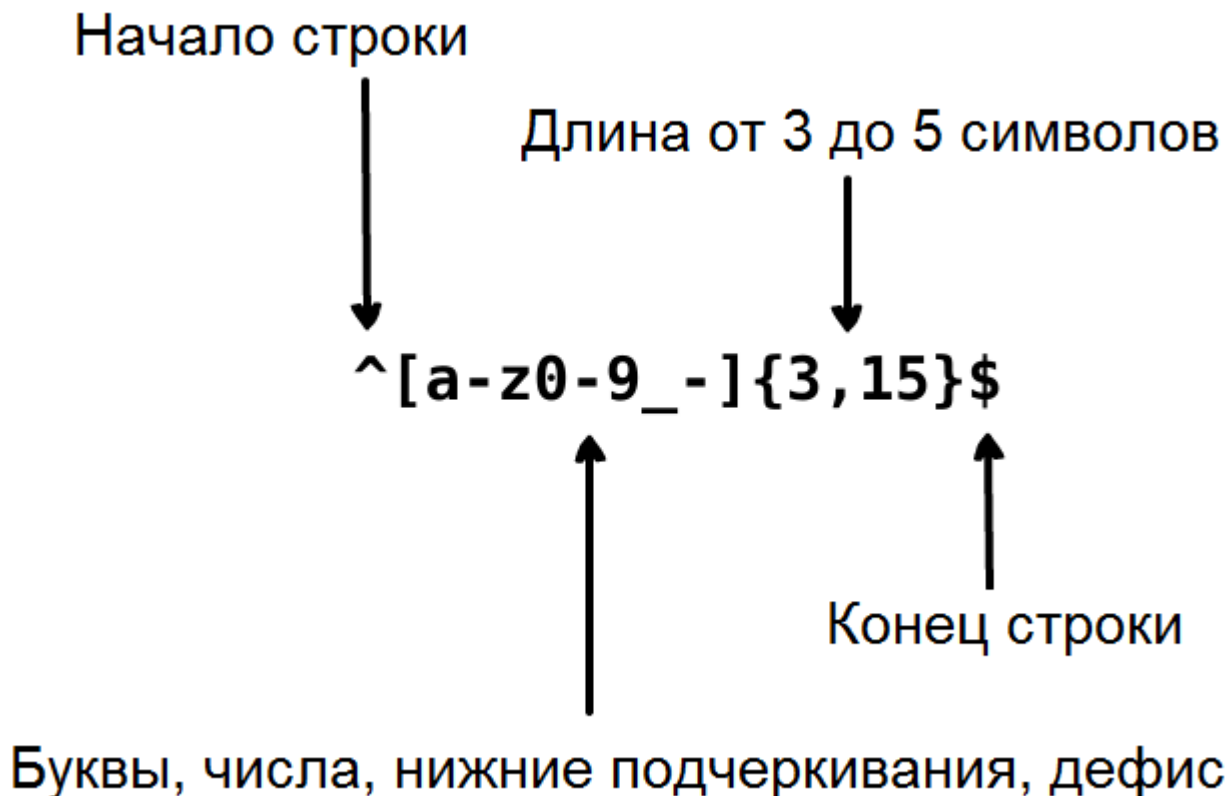
Регулярное выражение – это шаблон, который сравнивается с предметной строкой слева направо. Словосочетание “regular expression” применяется не так широко, вместо него обычно употребляют “регех” и “регехр”. Регулярное выражение используется для замены текста внутри строки, проверки формы, извлечения подстроки из строки на основе

Наш сайт использует файлы cookie для вашего максимального удобства.

Пользуясь сайтом, вы даете свое согласие с [условиями пользования](#)
[cookie](#)

[Согласен](#)

цифры, нижнее подчеркивание и дефисы. Также нам бы хотелось ограничить количество символов в имени пользователя, чтобы оно не выглядело уродливым. Поэтому для проверки будем использовать следующее регулярное выражение:



Это выражение принимает строки `john_doe`, `jo-hn_doe` и `john12_as`. Однако имя пользователя `Jo` не будет соответствовать этому выражению, потому что оно содержит прописную букву, а также является слишком коротким.

1. Базовые совпадения

Регулярное выражение - это всего лишь шаблон из символов, который мы используем для выполнения поиска в тексте. Например, регулярное выражение `the` означает букву `t`, за которой следует буква `h`, за которой следует буква `e`.

"the" => The fat cat sat on the mat.

Регулярное выражение `123` соответствует строке `123`. Регулярное выражение сопоставляется входной строке путем сравнения каждого символа в регулярном выражении с каждым символом входной строки. Регулярное выражение и входная строка сравниваются посимвольно. Обычно базовые совпадения чувствительны к регистру, поэтому `The` не соответствует `123`.

Наш сайт использует файлы cookie для вашего максимального удобства.

Пользуясь сайтом, вы даете свое согласие с [условиями пользования](#) [cookie](#)

Тестировать выражение

2. Метасимволы

Метасимволы служат строительными блоками regex. Они не являются независимыми и обычно интерпретируются каким-либо образом. Некоторые метасимволы имеют особое значение, а потому помещаются в квадратные скобки. Метасимволы:

Метасимволы	Описание
.	Любой единичный символ, исключая новую строку.
[]	Поиск набора символов, помещенных в скобки.
[^]	Negated character class. Matches any character that is not contained between the square brackets
*	0 или больше повторений предшествующего символа.
+	1 или больше повторений предшествующего символа.
?	Делает предшествующий символ опциональным.
{ n,m }	Возвращает как минимум "n", но не более "m" повторений предшествующего символа.
(xyz)	Находит группу символа в строго заданном порядке.
	Разделяет допустимые варианты.
\	Исключает следующий символ. Позволяет искать служебные символы [] () { } . * + ? ^ \$ \
^	Находит начало введенной строки.
\$	Находит конец введенной строки.

2.1 Точка

Наш сайт использует файлы cookie для вашего максимального удобства.
Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

«.ar» => The **car** parked in the **garage**.

[Тестировать выражение](#)

2.2 Интервал символов

Интервал или набор символов также называют символьным классом. Для его обозначения используются квадратные скобки. Чтобы указать диапазон символов внутри класса, необходимо поставить знак тире. Порядок ряда символов в наборе неважен. Так, например, регулярное выражение `[Tt]he` означает: `T` или `t`, за которым следует буква `h`, за которой следует буква `e`.

«`[Tt]he` » => **The** car parked in **the** garage.

[Тестировать выражение](#)

Стоит отметить, что точка, помещенная в квадратные скобки, означает именно точку, а ничто другое. Таким образом регулярное выражение `ar[.]` означает строчный символ `a`, за которым следует буква `r`, за которой следует точка `.`.

«`ar [.]`» => A garage is a good place to park a **car**.

[Тестировать выражение](#)

2.2.1 Отрицание набора символов

Обычно символ `^` представляет начало строки, но когда он внутри квадратных скобок, все символы, которые находятся после него, исключаются из шаблона. Например, выражение `[^c]ar` поможет отыскать все символы кроме `c`, за которыми следуют `a` и `r`.

"`[^c]ar`" => The car **parked** in the **garage**.

[Тестировать выражение](#)

2.3 Повторения

Следующие мета-символы `+`, `*` или `?` используются для того, чтобы обозначить

Наш сайт использует файлы cookie для вашего максимального удобства.
Пользуясь сайтом, вы даете свое согласие с [условиями пользования](#)
[cookie](#)

Этот символ поможет найти одно или более копий какого-либо символа. Регулярное выражение `a*` означает 0 или более повторений символа `a`. Но если этот символ появится после набора или класса символов, тогда будут найдены повторения всего сета. Например, выражение `[a-z]*` означает любое количество этих символов в строке.

`"[a-z]*" => The car parked in the garage #21.`

Тестировать выражение

Также символ может быть использован вместе с метасимволом `.` для подбора строки из любых символов `.*`.

Еще звездочку можно использовать со знаком пробела `\s`, чтобы подобрать строку из пробелов. Например, выражение `\s*cat\s` будет означать 0 или более пробелов, за которыми следует символ `c`, за ним `a` и `t`, а за ними снова 0 либо больше пробелов.

`"\s*cat\s*" => The fat cat sat on the concatenation.`

Тестировать выражение

2.3.2 Плюс

`+` соответствует одному или нескольким повторениям предыдущего символа. Например, регулярное выражение `c.+t` означает: строчная буква `c`, за которой следует хотя бы один символ, за которым следует строчный символ `t`. Необходимо уточнить, что буква `t` должна быть последней `t` в предложении.

`"c.+t" => The fat cat sat on the mat.`

Тестировать выражение

2.3.3. Вопросительный знак

В regex метасимвол `?` делает предшествующий символ необязательным. Этот символ соответствует полному отсутствию или же одному экземпляру предыдущего символа. Например, регулярное выражение `[T]?he` означает: необязательно заглавную букву `T`, за которой следует строчный символ `h`, за которым следует строчный символ `e`.

Наш сайт использует файлы cookie для вашего максимального удобства.

Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

[Тестировать выражение](#)

2.4 Скобки

Скобки в regex, которые также называются квантификаторами, используются для указания допустимого количества повторов символа или группы символов. Например, регулярное выражение `[0-9]{2,3}` означает, что допустимое количество цифр должно быть не менее двух цифр, но не более 3 (символы в диапазоне от 0 до 9).

```
"[0-9]{2,3}" => The number was 9.9997 but we rounded it off to 10.0.
```

[Тестировать выражение](#)

Мы можем убрать второе число. Например, выражение `[0-9]{2,}` означает 2 или более цифр. Если мы также уберем запятую, то тогда выражение `[0-9]{3}` будет находить только лишь 3 цифры, ни меньше и ни больше.

```
"[0-9]{2,}" => The number was 9.9997 but we rounded it off to 10.0.
```

[Тестировать выражение](#)

```
"[0-9]{3}" => The number was 9.9997 but rounded it off to 10.0.
```

[Тестировать выражение](#)

2.5 Символьная группа

Группа символов - это группа подшаблонов, которая записывается внутри скобок `(...)`. Как было упомянуто раньше, если в регулярном выражении поместить квантификатор после символа, он повторит предыдущий символ. Но если мы поставим квантификатор после группы символов, он просто повторит всю группу. Например, регулярное выражение `(ab)*` соответствует нулю или более повторениям символа «ab». Мы также можем использовать `|` - метасимвол чередования внутри группы символов. Например, регулярное выражение `(c|g|p)ar` означает: символ нижнего регистра `c`, `g` или `p`, за которым следует символ `a`, за которым следует символ `r`.

```
"(c|g|p)ar" => The car is parked in the garage.
```

Наш сайт использует файлы cookie для вашего максимального удобства.
Пользуясь сайтом, вы даете свое согласие с [условиями пользования](#)
[cookie](#)

В regex вертикальная полоса `|` используется для определения перечисления. Перечисление - это что-то вроде условия между несколькими выражениями. Можно подумать, что набор символов и перечисление работают одинаково, но это совсем не так, между ними существует огромная разница. Перечисление работает на уровне выражений, а набор символов на уровне знаков. Например, регулярное выражение `(T|t)he|car` означает: `T` или `t`, сопровождаемая строчным символом `h`, сопровождаемый строчным символом `e` или строчным символом `s`, а затем `a` и `i`.

`"(T|t)he|car" => The car is parked in the garage.`

[Тестировать выражение](#)

2.7 Исключение специального символа

Обратная косая черта `\` используется в regex, чтобы избежать символа, который следует за ней. Это позволяет нам указывать символ в качестве символа соответствия, включая зарезервированные `{ } [] / \ + * . $ ^ | ? .`. Чтобы использовать специальный символ в качестве подходящего, перед ним нужно поставить `\`.

Например, регулярное выражение `.` используется для нахождения любого единичного символа. Регулярное выражение `(f|c|m)at\.?` означает строчную букву `f`, `c` или `m`, а затем `a`, за ней `t` с последующим дополнительным символом `.`.

`"(f|c|m)at\." => The fat cat sat on the mat.`

[Тестировать выражение](#)

2.8 Анкеры - Привязки

В regex мы используем привязки, чтобы проверить, является ли соответствующий символ первым или последним символом входной строки. Привязка бывает двух типов: первый - это `^`, который проверяет является ли соответствующий символ первым введенным, а второй - знак доллара, который проверяет, является ли соответствующий символ последним символом введенной строки.

2.8.1. Caret

Символ `^` используется в regex, чтобы проверить, является ли соответствующий символ

Наш сайт использует файлы cookie для вашего максимального удобства.

Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

ничего не вернет, потому что во входной строке `abc` символ «b» не является первым. Давайте посмотрим на другое регулярное выражение `^(T|t)he`, которое означает: `T` или `t` - это символ начала входной строки, за которым следует строчный символ `h`, а затем `e`.

`"(T|t)he"` => **The** car is parked in **the** garage.

[Тестировать выражение](#)

`"^(T|t)he"` => **The** car is parked in the garage.

[Тестировать выражение](#)

2.8.2 Доллар

Знак доллара используется для проверки, является ли символ в выражении последним в введенной строке. Например `(at\\..)$` означает строчную `a`, за которой следует `t`, за которой следует `a .`, которые должны заканчивать строку.

`"(at\\..)"` => The fat **cat. sat. on the mat.**

[Тестировать выражение](#)

`"(at\\..)$"` => The fat cat. sat. on the **mat.**

[Тестировать выражение](#)

3. Сокращения для обозначения символов

Регекс позволяет использовать сокращения для некоторых наборов символов, что делает работу с ними более комфортной. Таким образом, здесь используются следующие сокращения:

Сокращение	Описание
<code>.</code>	Любой символ, кроме новой строки
<code>\w</code>	Соответствует буквенно-цифровым символам: <code>[a-zA-Z0-9_]</code>

Наш сайт использует файлы cookie для вашего максимального удобства. Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

Сокращение	Описание
\d	Соответствует нецифровым знакам: <code>[^\d]</code>
\s	Соответствует знаку пробела: <code>[\t\n\f\r\p{Z}]</code>
\S	Соответствует символам без пробела: <code>[^\s]</code>

4. Lookaround Позиционная проверка

Lookbehind и lookahead (также называемые lookaround) - это определенные типы non-capturing групп (Они используются для поиска, но сами в него не входят). Lookaheads используются, когда у нас есть условие, что этому шаблону предшествует или следует другой шаблон. Например, мы хотим получить все числа, которым предшествует символ \$ из входной строки \$4.44 and \$10.88 . Мы будем использовать регулярное выражение `(?<=\$)[0-9\.]*` , которое означает: получить все числа, содержащие . и которым предшествует символ \$. Ниже приведены lookarounds, что используются в регулярных выражениях:

Символ	Описание
?=	Положительный Lookahead
?!	Отрицательный Lookahead
?<=	Положительный Lookbehind
?<!	Отрицательный Lookbehind

4.1 Положительный Lookahead

Положительный lookahead означает, что эта часть выражения должна следовать за впереди идущим выражением. Возвращаемое значение содержит текст, который совпадает с первой частью выражения. Чтобы определить позитивный lookahead, используют скобки. Внутри них размещают знак вопроса и знак равенства: `(?=...)` . Само же выражение пишется после = . Например, выражение `(T|t)he(=?\sfat)` - это T в верхнем или

Наш сайт использует файлы cookie для вашего максимального удобства.
Пользуясь сайтом, вы даете свое согласие с [условиями пользования](#)
[cookie](#)

```
"(T|t)he(=?\sfat)" => The fat cat sat on the mat.
```

[Тестировать выражение](#)

4.2 Отрицательный Lookahead

Негативный lookahead используется, когда нам нужно получить все совпадения в строке, за которой не следует определенный шаблон. Негативный lookahead определяется так же, как и позитивный, с той лишь разницей, что вместо знака равенства мы используем знак отрицания `!`. Таким образом, наше выражение приобретает следующий вид: `(?!...)`. Теперь рассмотрим `(T|t)he(?!\sfat)`, что означает: получить все `The` или `the` в введенной строке, за которыми не следует слово `fat`, предшествующее знаку пробела.

```
"(T|t)he(?!\sfat)" => The fat cat sat on the mat.
```

[Тестировать выражение](#)

4.3 Положительный Lookbehind

Положительный lookbehind используется для получения всех совпадений, которым предшествует определенный шаблон. Положительный lookbehind обозначается так: `(?<=...)`. Например, регулярное выражение `(?<=(T|t)he\s)(fat|mat)` означает получить все `fat` или `mat` из строки ввода, которые идут после слова `The` или `the`.

```
"(? The fat cat sat on the mat."
```

[Тестировать выражение](#)

4.4 Отрицательный Lookbehind

Отрицательный lookbehind используется для получения всех совпадений, которым не предшествует определенный шаблон. Отрицательный lookbehind обозначается выражением `(?<!...)`. Например, регулярное выражение `(?<!(T|t)he\s)(cat)` означает: получить все `cat` слова из строки ввода, которые не идут после `The` или `the`.

```
"(? The cat sat on cat."
```

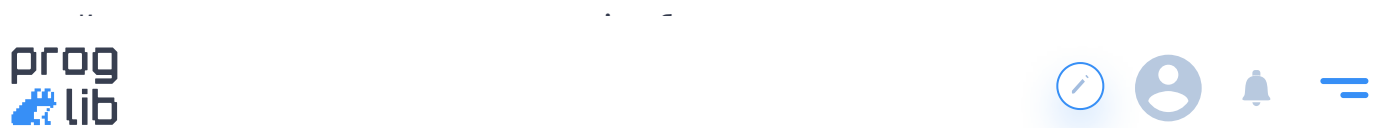
Наш сайт использует файлы cookie для вашего максимального удобства.
Пользуясь сайтом, вы даете свое согласие с [условиями пользования](#)
[cookie](#)

Флаги также часто называют модификаторами, так как они могут изменять вывод регехр. Флаги, приведенные ниже являются неотъемлемой частью и могут быть использованы в любом порядке или сочетании регехр.

Флаг	Описание
i	Нечувствительность к регистру: делает выражение нечувствительным к регистру.
g	Глобальный поиск: поиск шаблона во всей строке ввода.
m	Многострочность: анкер метасимвола работает в каждой строке.

5.1 Нечувствительные к регистру

Модификатор `i` используется для поиска совпадений, нечувствительных к регистру. Например, выражение `/The/gi` означает прописную букву `T`, за которой следуют `h` и



Тестировать выражение

`"/The/gi" => The fat cat sat on the mat.`

Тестировать выражение

5.2 Глобальный поиск

Модификатор используется для выполнения глобального поиска шаблона(поиск будет продолжен после первого совпадения). Например, регулярное выражение `/(at)/g` означает любой символ, кроме новой строки, за которым следует строчный символ `a`, а затем `t`. Поскольку мы использовали флаг `g` в конце регулярного выражения, теперь он найдет все совпадения в вводимой строке, а не только в первой (что является стандартом).

`"/.(at)/" => The fat cat sat on the mat.`

Наш сайт использует файлы cookie для вашего максимального удобства.
Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

[Тестировать выражение](#)

5.3 Многострочный поиск

Модификатор `m` нужен для выполнения многострочного поиска. Как было сказано ранее, привязки (`^`, `$`) используются для проверки, является ли шаблон началом или концом строки. Но если мы хотим, чтобы привязки работали в каждой строке, нужно использовать флаг `m`. Например, регулярное выражение `/at(.*?)$/gm` означает: строчный символ `a`, за которым следует `t` и что угодно, только не новая строка. А благодаря флагу `m` этот механизм регулярных выражений соответствует шаблону в конце каждой строки строки.

```
"/.at(.*?)$/gm" => The fat  
cat sat  
on the mat.
```

[Тестировать выражение](#)

```
"/.at(.*?)$/gm" => The fat  
cat sat  
on the mat.
```

[Тестировать выражение](#)

6. Жадные vs. ленивые выражения

По умолчанию регулярные выражения выполняются благодаря "жадным" квантификаторам, им соответствует максимально длинная строка из всех возможных.

```
"/(. *at)/" => The fat cat sat on the mat.
```

[Тестировать выражение](#)

Чтобы получить "ленивое" выражение, нужно использовать `?`. Так будет получена максимально короткая строка.

```
"/(. *?at)/" => The fat cat sat on the mat.
```

Наш сайт использует файлы cookie для вашего максимального удобства.
Пользуясь сайтом, вы даете свое согласие с [условиями пользования](#)
[cookie](#)

- Практическое введение в регулярные выражения для новичков
- 5 практических примеров использования регулярных выражений на JavaScript
- 11 материалов по регулярным выражениям

Источник

♥ 15 💬 3 📌 8 🔥 1 💧 0 💩 0

Учиться



Реклама. ООО «Озон Технологии»

ROUTE
256

OZON

Бесплатные курсы
для middle-разработчиков



МЕРОПРИЯТИЯ

Cinimex DATA meetup

📅 15 февраля [Санкт-Петербург](#) [Онлайн](#) [Бесплатно](#)

+ Показать еще

Комментарии

Оставьте свой комментарий (можно использовать markdown)




Отправить



Подписывайтесь на рассылку

Наш сайт использует файлы cookie для вашего максимального удобства.
Пользуясь сайтом, вы даете свое согласие с [условиями пользования](#)
[cookie](#)

а есть регулярное выражение для любого кол-ва символов с пробелами и без? То есть вообще любые символы в определённом диапазоне? В моём случае этот диапазон между `div` и `/div`

Ответить ...


 masigo egi 10 декабря 2021

 0  0

Спасибо , коротко без воды и с примерами в песочнице!!!

Ответить ...

 Kolyus 05 марта 2020

 0  0

Опечатка - "Например, выражение `\s*cat\s` будет означать 0 или более пробелов, за которыми следует символ `s`, за ним `a` и `t`, а за ними снова 0 либо больше пробелов". В данном случае выражение должно быть - `\s*cat\s*`

Ответить ...

ЛУЧШИЕ СТАТЬИ ПО ТЕМЕ

Помнить все: делимся лучшей шпаргалкой по Python

Мы подготовили очень занимательную коллекцию, которая по праву может называться лучшей шпаргалкой по Python благодаря ее простоте использования.

Английский язык для IT-специалистов

Всем людям, так или иначе связанным с IT сферой, прекрасно известно, что рано или поздно вопрос о владении английским языком становится ребром.

Изучаем алгоритмы: полезные книги. веб-сайты. онлайн-

0 проекте

Реклама

Пользовательское соглашение

Наш сайт использует файлы cookie для вашего максимального удобства.

Пользуясь сайтом, вы даете свое согласие с [условиями пользования cookie](#)

Контакты

☐ Push-уведомления

☐ Темная тема



© 2024, Proglib. При копировании материала ссылка на источник обязательна.

Наш сайт использует файлы cookie для вашего максимального удобства.
Пользуясь сайтом, вы даете свое согласие с [условиями пользования](#)
[cookie](#)