

Раздел

[Регулярные выражения](#)

Навигация по уроку

Количество {n}


Короткие обозначения

Ещё примеры

Задачи (2)

Комментарии

Поделиться

[Редактировать на GitHub](#) → [Регулярные выражения](#) 6-го сентября 2019

## Квантификаторы +, \*, ? и {n}

Давайте возьмём строку вида +7(903)-123-45-67 и найдём все числа в ней. Но теперь нас интересуют не цифры по отдельности, а именно числа: 7, 903, 123, 45, 67.

Число — это последовательность из 1 или более цифр `\d`. Чтобы указать количество повторений, нам нужно добавить *квантификатор*.

### Количество {n}

Самый простой квантификатор — это число в фигурных скобках: `{n}`.


Он добавляется к символу (или символному классу, или набору `[...]` и т.д.) и указывает, сколько их нам нужно.

Можно по-разному указать количество, например:

**Точное количество: `{5}`**

Шаблон `\d{5}` обозначает ровно 5 цифр, он эквивалентен `\d\d\d\d\d`.

Следующий пример находит пятизначное число:

```
1 alert( "Мне 12345 лет".match(/\d{5}/) ); // "12345" 
```

Мы можем добавить `\b`, чтобы исключить числа длиннее: `\b\d{5}\b`.

**Диапазон: `{3,5}`, от 3 до 5**

Для того, чтобы найти числа от 3 до 5 цифр, мы можем указать границы в фигурных скобках: `\d{3,5}`

```
1 alert( "Мне не 12, а 1234 года".match(/\d{3,5}/) ); 
```


Верхнюю границу можно не указывать.

Тогда шаблон `\d{3,}` найдёт последовательность чисел длиной 3 и более цифр:

```
1 alert( "Мне не 12, а 345678 лет".match(/\d{3,}/) ); 
```

Давайте вернёмся к строке +7(903)-123-45-67.

Число — это последовательность из одной или более цифр. Поэтому шаблон будет `\d{1,}`:

```
1 let str = "+7(903)-123-45-67"; 
2
3 let numbers = str.match(/\d{1,}/g);
4
5 alert(numbers); // 7,903,123,45,67
```

### Короткие обозначения

Для самых востребованных квантификаторов есть сокращённые формы записи:

+

Означает «один или более». То же самое, что и `{1,}`.

Например, `\d+` находит числа (из одной или более цифр):

Раздел

Регулярные выражения

Навигация по уроку

Количество {n}

Короткие обозначения

Ещё примеры

Задачи (2)

Комментарии

Поделиться



Редактировать на GitHub

```
1 let str = "+7(903)-123-45-67";
2
3 alert( str.match(/\d+/g) ); // 7,903,123,45,67
```

?

Означает «ноль или один». То же самое, что и {0,1}. По сути, делает символ необязательным.

Например, шаблон ou?r найдёт o после которого, возможно, следует u, а затем r.

Поэтому шаблон colou?r найдёт два варианта: color и colour:

```
1 let str = "Следует писать color или colour?";
2
3 alert( str.match(/colou?r/g) ); // color, colour
```

\*

Означает «ноль или более». То же самое, что и {0,}. То есть символ может повторяться много раз или вообще отсутствовать.

Например, шаблон \d0\* находит цифру и все нули за ней (их может быть много или ни одного):

```
1 alert( "100 10 1".match(/\d0*/g) ); // 100, 10, 1
```

Сравните это с + (один или более):

```
1 alert( "100 10 1".match(/\d0+/g) ); // 100, 10
2 // 1 не подходит, т.к 0+ требует как минимум один ноль
```

## Ещё примеры

Квантификаторы используются очень часто. Они служат основными «строительными блоками» сложных регулярных выражений, поэтому давайте рассмотрим ещё примеры.

**Регулярное выражение для десятичных дробей (чисел с плавающей точкой):** \d+\.\d+

В действии:

```
1 alert( "0 1 12.345 7890".match(/\d+\.\d+/g) ); // 12.34
```

**Регулярное выражение для «открывающего HTML-тега без атрибутов», например, <span> или <p>.**

1. Самое простое: /[a-z]+>/i

```
1 alert( "<body> ... </body>".match(/[a-z]+>/gi) ); //
```

Это регулярное выражение ищет символ '<', за которым идут одна или более букв латинского алфавита, а затем '>'.

2. Улучшенное: /[a-z][a-z0-9]\*>/i

Здесь регулярное выражение расширено: в соответствии со стандартом, в названии HTML-тега цифра может быть на любой позиции, кроме первой, например <h1>.

```
1 alert( "<h1>Привет!</h1>".match(/[a-z][a-z0-9]*>/gi) );
```

Раздел

Регулярные выражения

Навигация по уроку

Количество {n}

Короткие обозначения

Ещё примеры

Задачи (2)

Комментарии

Поделиться



Редактировать на GitHub



Регулярное выражение для «открывающего или закрывающего HTML-тега без атрибутов»: `/<\/?[a-z][a-z0-9]*>/i`

В начало предыдущего шаблона мы добавили необязательный слеш `/?`. Этот символ понадобилось заэкранировать, чтобы JavaScript не принял его за конец шаблона.

```
1 alert( "<h1>Привет!</h1>".match(/<\/?[a-z][a-z0-9]*>/gi)
```

**!** Чтобы регулярное выражение было точнее, нам часто приходится делать его сложнее

В этих примерах мы видим общее правило: чем точнее регулярное выражение – тем оно длиннее и сложнее.

Например, для HTML-тегов без атрибутов, скорее всего, подошло бы и более простое регулярное выражение: `<\w+>`. Но стандарт HTML накладывает более жёсткие ограничения на имя тега, так что более точным будет шаблон `<[a-z][a-z0-9]*>`.

Подойдёт ли нам `<\w+>` или нужно использовать `<[a-z][a-z0-9]*>`? А, может быть, нужно ещё его усложнить, добавить атрибуты?

В реальной жизни допустимы разные варианты. Ответ на подобные вопросы зависит от того, насколько реально важна точность и насколько потом будет сложно или несложно отфильтровать лишние совпадения.

## ✓ Задачи

### Как найти многоточие "..."?

важность: 5

Напишите регулярное выражение, которое ищет многоточие (3 и более точек подряд).

Проверьте его:

```
1 let regexp = /ваше выражение/g;
2 alert( "Привет!... Как дела?....".match(regexp) ); //
```

решение

### Регулярное выражение для HTML-цветов

Напишите регулярное выражение, которое ищет HTML-цвета в формате #ABCDEF: первым идёт символ #, и потом – 6 шестнадцатеричных символов.

Пример использования:

```
1 let regexp = /...ваше выражение.../
2
3 let str = "color:#121212; background-color:#AA00ef bad-
4
5 alert( str.match(regexp) ) // #121212,#AA00ef
```

P.S. В рамках этого задания не нужно искать цвета, записанные в иных форматах типа #123 или rgb(1,2,3).

решение

Раздел

[Регулярные выражения](#)

Навигация по уроку

Количество {n}

Короткие обозначения

Ещё примеры

Задачи (2)

Комментарии

Поделиться



[Редактировать на GitHub](#)



## Комментарии

перед тем как писать...



© 2007–2020 Илья Кантор | [о проекте](#) | [связаться с нами](#) | [пользовательское соглашение](#) | [политика конфи](#)

