1-го августа 2019

Начало строки ^ и конец $

Материал на этой странице устарел, поэтому скрыт из оглавления сайта.

Более новая информация по этой теме находится на странице https://learn.javascript.ru/regexp-anchors.

Знак каретки `'^'` и доллара `'$'` имеют в регулярном выражении особый смысл. Их называют «якорями» (anchor – англ.).

Каретка `^` совпадает в начале текста, а доллар `$` – в конце.

Якоря являются не символами, а проверками.

До этого мы говорили о регулярных выражениях, которые ищут один или несколько символов. Если совпадение есть – эти символы включаются в результат.

А якоря – не такие. Когда поиск ходит до якоря – он проверяет, есть ли соответствие, если есть – продолжает идти по шаблону, не прибавляя ничего к результату.

Каретку `^` обычно используют, чтобы указать, что регулярное выражение необходимо проверить именно с начала текста.

Например, без каретки найдёт все числа:

```
var str = '100500 попугаев съели 500100 бананов!';
alert( str.match(/\d+/ig) ); // 100500, 500100 (нашло все числа)
```

А с кареткой – только первое:

```
var str = '100500 попугаев съели 500100 бананов!';
alert( str.match(/^\d+/ig) ); // 100500 (только в начале строки)
```

Знак доллара `$` используют, чтобы указать, что паттерн должен заканчиваться в конце текста.

Аналогичный пример с долларом для поиска числа в конце:

```
var str = '100500 попугаев съели 500100';
alert( str.match(/\d+$/ig) ); // 500100
```

Оба якоря используют одновременно, если требуется, чтобы шаблон охватывал текст с начала и до конца. Обычно это требуется при валидации.

Например, мы хотим проверить, что в переменной `num` хранится именно десятичная дробь.

Ей соответствует регэксп `\d+\.\d+`. Но простой поиск найдёт дробь в любом тексте:

```
var num = "ля-ля 12.34";
alert( num.match(/\d+\.\d+/ig) ); // 12.34
```

Наша же задача – проверить, что `num` *целиком* соответствует паттерну `\d+\.\d+`.

Для этого обернём шаблон в якоря `^...$`:

```
var num = "ля-ля 12.34";
alert( num.match(/^\d+\.\d+$/ig) ); // null, не дробь

var num = "12.34";
alert( num.match(/^\d+\.\d+$/ig) ); // 12.34, дробь!
```

Теперь поиск ищет начало текста, за которым идёт число, затем точка, ещё число и конец текста. Это как раз то, что нужно.

Комментарии

Александр Гусев • 5 лет назад

Смотря по комментариям, основная масса до сюда не доходит:))

9 • Ответить • Поделиться ›

Dmitry • 2 года назад • edited

Пришла мысль сделать так: /^([0-9a-f]{2})(:\1){5}$/i но очухался, что в \1 будет прилетать второе число мак-адреса, а не само выражение в скобках

1 • Ответить • Поделиться ›

Maxim Kaplin • 2 года назад

автор спасибо!

1 • Ответить • Поделиться ›

Азат Сабыров • 7 месяцев назад

Здравствуйте, а можно ли использовать это в **:contains**?

• Ответить • Поделиться ›

Vitaliy • год назад

задача с MAC /([A-F0-9]{2}(:|$)){6}/

• Ответить • Поделиться ›

Костян Ермаков • 2 года назад • edited

Сначала получилось такое выражение:

```
var re = /^((\d|[A-F]){2}:?){6}$/g;
```

Но оно в каких-то случаях (не помню уже в каких), ему начала нравиться вторая проверка, где строка без двоеточий.
Пришлось переделать немного его, но получилось длиннее:

```
var re = /^((\d|[A-F]){2}:){5}(\d|[A-F]){2}$/g;
```

Казалось бы, что можно укоротить данное выражение, дав просто ссылку на второе скобочное выражение:

```
var re = /^((\d|[A-F]){2}:){5}\2{2}$/g;
```

, но такой вариант не работает почему-то (пробовал оборачивать \2 в скобки(), вариант тоже не рабочий). Может ли мне кто-то разъяснить данный момент? (нашел ответ на данный вопрос в комментариях ниже).
А чего так мало решений, обычно есть из чего выбрать и посмотреть, попробовать отследить ход мыслей?

• Ответить • Поделиться ›

Вадим Свиридов ➚ Костян Ермаков • 2 года назад • edited

У тебя изначально (\d|[A-F]) работает как "Если часть шаблона обозначена скобками, то она станет отдельным элементом массива" значит что два элемента пойдут в разные ячейки массива, для этого надо использовать [a-f0-9], а \2 говорит пропустить все (что было взято пять раз) два раза и взять конечное значение, я даже не знаю на каком примере это все будет работать.
Вот сайт для провреки https://regex101.com/