

Раздел

Регулярные выражения

Навигация по уроку

Экранирование символов

Косая черта

new RegExp


Итого

Комментарии

Поделиться



Редактировать на GitHub

 → Регулярные выражения 6-го сентября 2019

Экранирование, специальные символы

Как мы уже видели, обратная косая черта `\` используется для обозначения классов символов, например `\d`. Это специальный символ в регулярных выражениях (как и в обычных строках).

Есть и другие специальные символы, которые имеют особое значение в регулярном выражении. Они используются для более сложных поисковых конструкций. Вот полный перечень этих символов: `[\ ^ $. | ? * + ()]`.

Не надо пытаться запомнить этот список: мы разберёмся с каждым из них по отдельности, и таким образом вы выучите их «автоматически».

Экранирование символов

Допустим, мы хотим найти буквально точку. Не «любой символ», а именно точку.

Чтобы использовать специальный символ как обычный, добавьте к нему обратную косую черту: `\.`

Это называется «экранирование символа».

К примеру:

```
1 alert( "Глава 5.1".match(/\\d\\.\\d/) ); // 5.1 (совпадение)
2 alert( "Глава 511".match(/\\d\\.\\d/) ); // null ("\" - и
```

Круглые скобки также являются специальными символами, поэтому, если нам нужно использовать именно их, нужно указать `\\`. В приведённом ниже примере ищется строка `"g()"`:

```
1 alert( "function g()".match(/g\\(\\)/) ); // "g()"
```

Если мы ищем обратную косую черту `\\`, это специальный символ как в обычных строках, так и в регулярных выражениях, поэтому мы должны удвоить её.

```
1 alert( "1\\2".match(/\\\\/) ); // '\\'
```

Косая черта

Символ косой черты `'/'`, так называемый «слэш», не является специальным символом, но в JavaScript он используется для открытия и закрытия регулярного выражения: `/...шаблон.../`, поэтому мы должны экранировать его.

Вот как выглядит поиск самой косой черты `'/'`:

```
1 alert( "/" .match(/\\//) ); // '/'
```

С другой стороны, если мы не используем короткую запись `/.../`, а создаём регулярное выражение, используя `new RegExp`, тогда нам не нужно экранировать косую черту:

```
1 alert( "/" .match(new RegExp("/")) ); // находит /
```

Раздел

Регулярные выражения

Навигация по уроку

Экранирование символов

Косая черта

new RegExp

Итого

Комментарии

Поделиться



Редактировать на GitHub

new RegExp

Если мы создаём регулярное выражение с помощью `new RegExp`, то нам не нужно учитывать `/`, но нужно другое экранирование.

Например, такой поиск не работает:

```
1 let regexp = new RegExp("\\d\\.\\d");
2
3 alert( "Глава 5.1".match(regexp) ); // null
```

Аналогичный поиск в примере выше с `/\\d\\.\\d/` вполне работал, почему же не работает `new RegExp("\\d\\.\\d")`?

Причина в том, что символы обратной косой черты «съедаются» строкой. Как вы помните, что обычные строки имеют свои специальные символы, такие как `\n`, и для экранирования используется обратная косая черта.

Вот как воспринимается строка «`\\d\\.\\d`»:

```
1 alert("\\d\\.\\d"); // d.d
```

Строковые кавычки «съедают» символы обратной косой черты для себя, например:

- `\n` — становится символом перевода строки,
- `\u1234` — становится символом Юникода с указанным номером,
- ...А когда нет особого значения: как например для `\d` или `\z`, обратная косая черта просто удаляется.

Таким образом, `new RegExp` получает строку без обратной косой черты. Вот почему поиск не работает!

Чтобы исправить это, нам нужно удвоить обратную косую черту, потому что строковые кавычки превращают `\\` в `\`:

```
1 let regStr = "\\d\\.\\d";
2 alert(regStr); // \\d\\.\\d (теперь правильно)
3
4 let regexp = new RegExp(regStr);
5
6 alert( "Глава 5.1".match(regexp) ); // 5.1
```

Итого

- Для поиска специальных символов `[\ ^ $. | ? * + ()]`, нам нужно добавить перед ними `\` («экранировать их»).
- Нам также нужно экранировать `/`, если мы используем `/.../` (но не `new RegExp`).
- При передаче строки в `new RegExp` нужно удваивать обратную косую черту: `\\` для экранирования специальных символов, потому что строковые кавычки «съедают» одну черту.

Проводим курсы по JavaScript и фреймворкам.



Комментарии

перед тем как писать...