Алгоритмы обработки данных на JavaScript

Работа со строками

Строки

Строка в JS это любые текстовые данные.
Строки полезны для хранения данных, которые можно представить в текстовой форме.

Есть несколько способов создать строку:

- одинарными кавычками ';
- двойными кавычками ";
- шаблонной строкой через обратный апостроф `

Записи одинарными и двойными кавычками **идентичны**

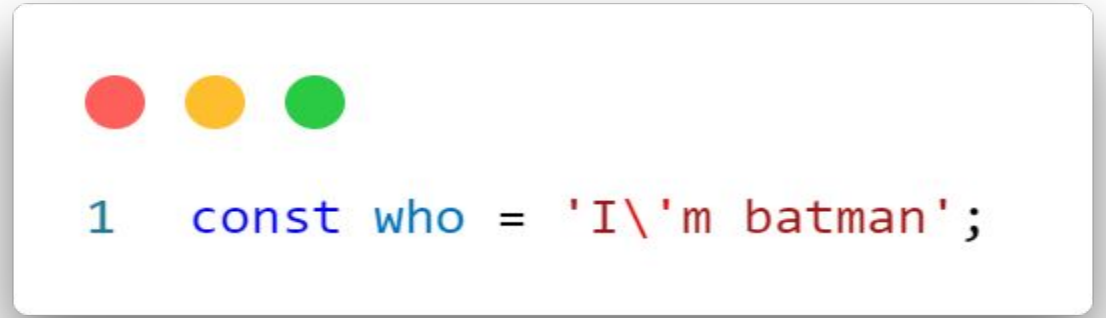

1 const str1 = 'hello';
2 const str2 = "hello";
3 const str3 = `hello`;

Спецсимволы

Спецсимвол	Значение
\n	Начало новой строки
\t	Табуляция, аналогичная нажатию клавиши Tab
\’ \”	Экранированная кавычка

Экранирование

Если в строке, объявленной при помощи одинарных кавычек, нужно поставить апостроф, то символ экранируют обратным слэшем \. Так мы даём JavaScript понять, что это просто символ, а не закрывающая кавычка.



```
1  const poem = "Я вижу - \n\tздесь \n\t\tстоял Маяковский,\nстоял \n\tи стихи слагал по слогам..."
2  console.log(poem);
3  Я вижу -
4      здесь
5          стоял Маяковский,
6  стоял
7      и стихи слагал по слогам...
```

Конкатенация строк

Для строк определена операция сложения, её также называют конкатенацией строк. При сложении двух строк получается новая строка, склеенная из исходных.

```
1  const greeting = 'Hello, ';
2  const userName = 'User';
3
4  console.log(greeting + userName + '!');
```

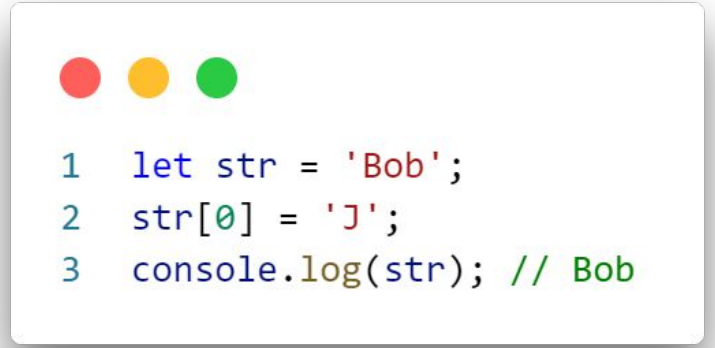
Длина строки

Свойство length представляет длину строки.

```
1  const js = 'JavaScript';
2  const emptyString = '';
3  console.log('Слово ' + js + ' занимает ' + js.length + ' символов');
4  console.log('Пустая строка занимает ' + emptyString.length + ' символов');
```

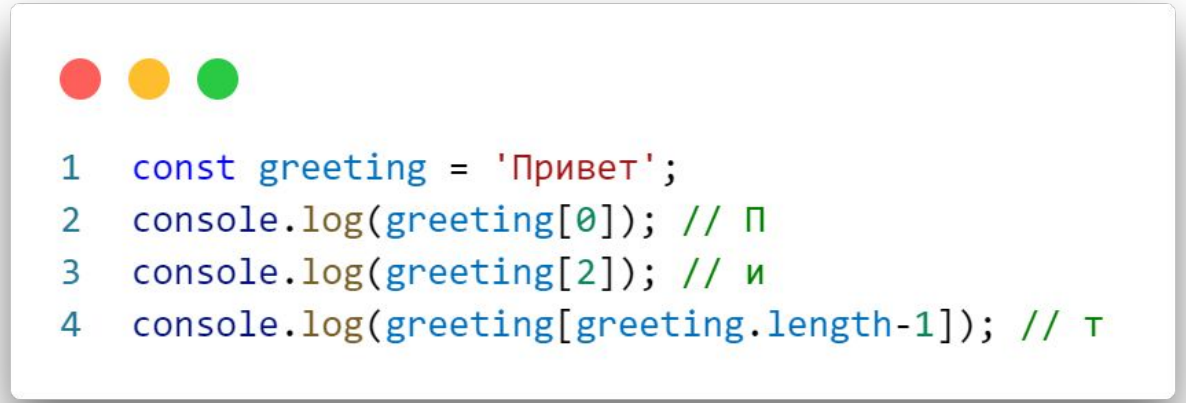
Неизменяемость

В JavaScript значения String неизменяемы, что означает, что они не могут быть изменены после создания.



Получение отдельного символа

Получить символ строки можно по порядковому номеру символа. Конструкция аналогична получению элемента массива по индексу.



```
const greeting = 'Привет';
console.log(greeting[0]); // П
console.log(greeting[2]); // и
console.log(greeting[greeting.length-1]); // т
```

Шаблонные строки

Шаблонными строками называются строковые литералы, допускающие использование выражений внутри. С ними вы можете использовать многострочные литералы и строковую интерполяцию

Интерполяция выражений

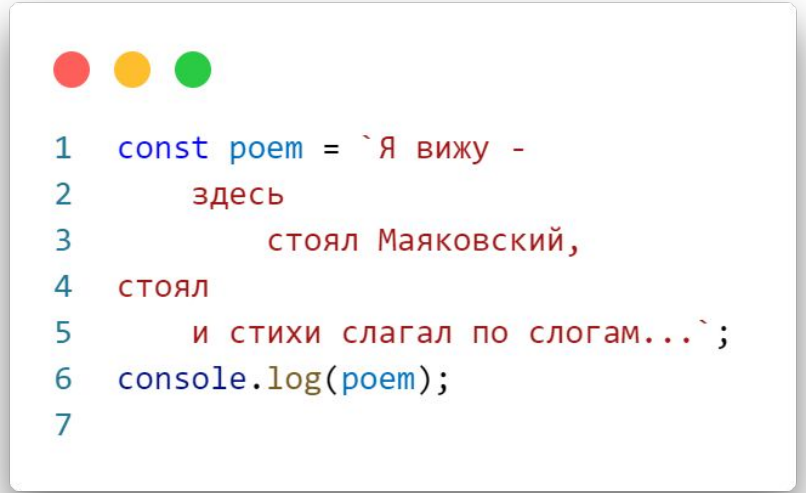
Шаблонные строки могут содержать подстановки, обозначаемые знаком доллара и фигурными скобками ${выражение}. Значение выражение будет вставлено в строку.

```
1  const a = 20;
2  const b = 30;
3
4  const result = `${a}x${b}=${a * b}`;
5  console.log(result); //20x30=600
```

```
1  const a = 20;
2  const b = 30;
3
4  const result = a + 'x' + b + '=' + a * b;
5  console.log(result);
6
```

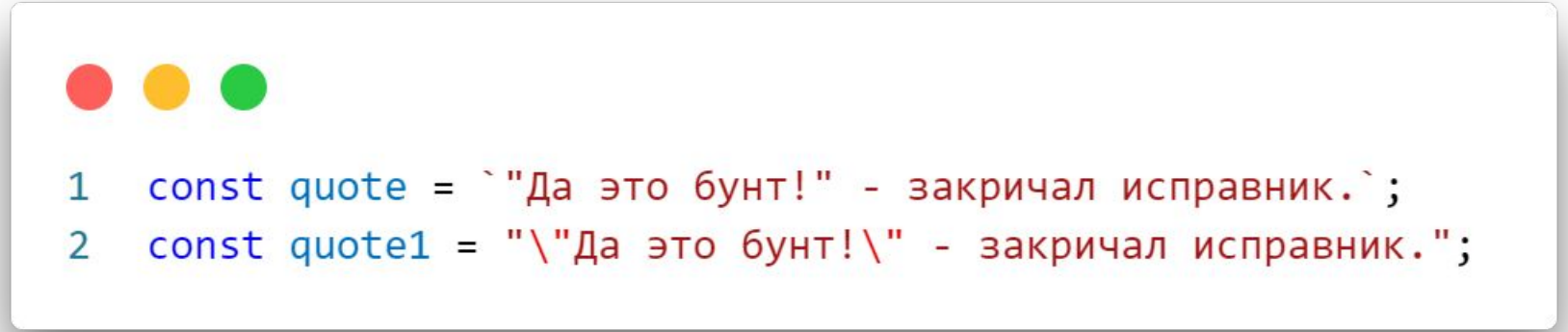
Многострочные литералы

Символы новой строки являются частью шаблонных строк.



```
1  const poem = `Я вижу -
2      здесь
3          стоял Маяковский,
4  стоял
5      и стихи слагал по слогам...`;
6  console.log(poem);
7
```

Экранирование символов



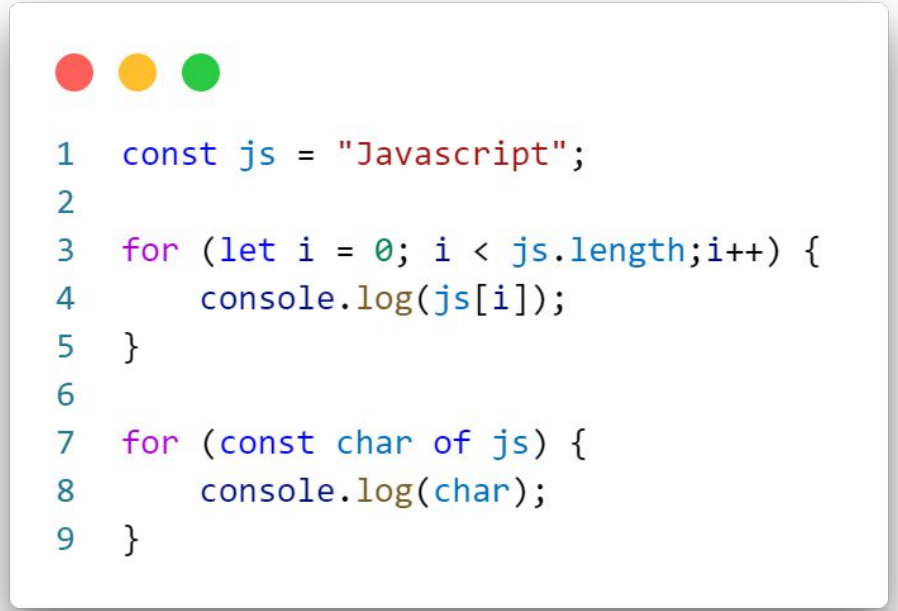
```
1   const quote = `"Да это бунт!" - закричал исправник.`;
2   const quote1 = "\"Да это бунт!\" - закричал исправник.";
```

Изменение регистра

Методы toLowerCase и toUpperCase позволяют изменить регистр символов

```
1  'UPPER'.toLowerCase(); // upper
2  'lower'.toUpperCase(); // LOWER
```

Итерация по строке



```
1  const js = "Javascript";
2
3  for (let i = 0; i < js.length;i++) {
4      console.log(js[i]);
5  }
6
7  for (const char of js) {
8      console.log(char);
9  }
```

Поиск подстроки

Метод str.includes(substr) строки str возвращает true, если в строке str есть подстрока substr. В противном случае false.

Метод startsWith(substr) строки str возвращает true, если строка str начинается с подстроки substr. В противном случае false

Метод endsWith(substr) строки str возвращает true, если строка str заканчивается подстрокой substr. В противном случае false

```
const text = 'JSON is a lightweight data-interchange format';

text.contains('data'); // true
text.startsWith('JS'); // true
text.endsWith('at'); // true

text.contains('JSon'); // false

```

Получение подстроки

Метод slice(start, end) извлекает часть строки от start до (не включая) end и возвращает новую строку без изменения оригинальной строки. Если end не указан, то возвращается подстрока до конца строки

```
let str = 'Bob';
str = 'J' + str.slice(1);
console.log(str); // Job
```

```
console.log("Javascript".slice(3, 5)); //as
```

Преобразование строки в массив

Метод str.split(delim) разбивает строку str на массив по заданному разделителю delim

```
1  const fruits = 'Яблоко,груша,банан';
2  const list = fruits.split(',');
3  // ['Яблоко','груша','банан']
```

Преобразование массива в строку

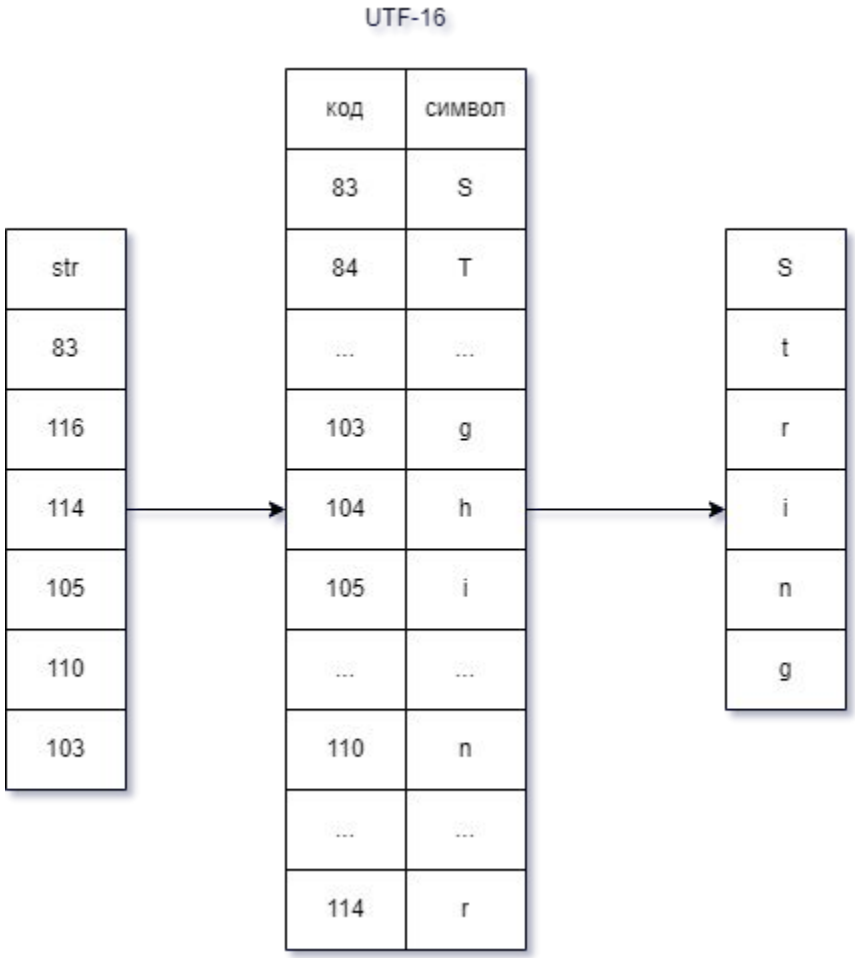
Метод arr.join(glue) создаёт строку из элементов массива arr, используя как разделитель строку glue.

```
1  const names = ['Яблоко', 'Груша', 'Банан'];
2  const str = names.join('/');
3  console.log(str);
4  // Яблоко/Груша/Банан
```

Сравнение строк

Символ, который видно на экране хранится в компьютере как одно или несколько чисел, каждое такое число называют юнитом. Компьютер хранит таблицу в которой числу соответствует символ. Такие таблицы называют кодировкой. В JS используются кодировка UTF-16

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

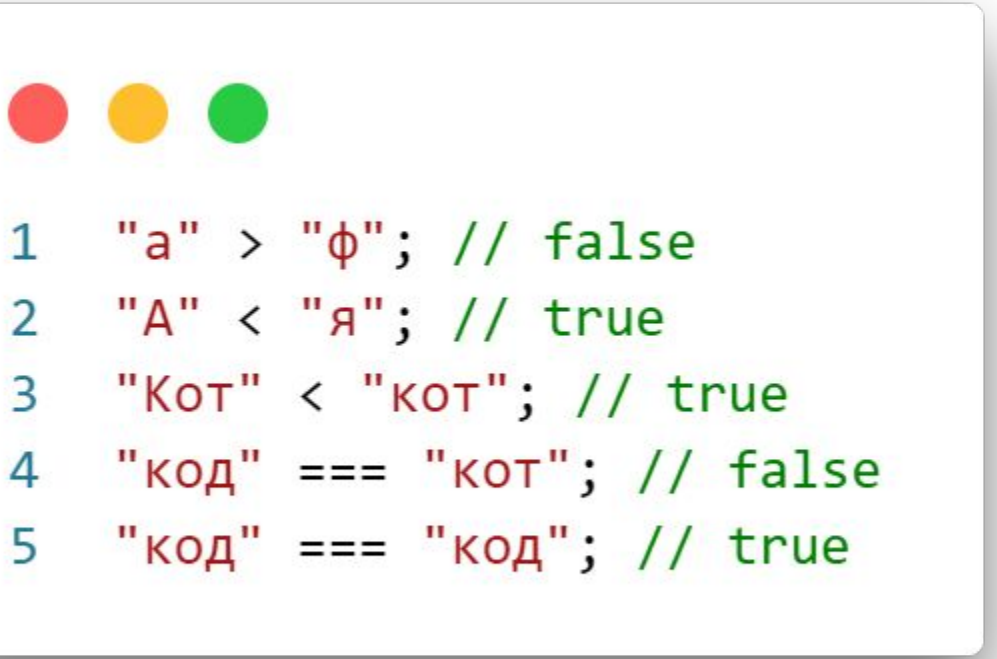

UTF-16
код
символ
83
S
84
T
...
...
103
g
104
h
105
i
...
...
110
n
...
...
114
r
str
83
116
114
105
110
103
S
t
r
i
n
g

Сравнение строк

Строки сравниваются в лексикографическом порядке. (Первые буквы алфавита меньше последних). **Сравнение осуществляется с учётом регистра**

Алгоритм сравнения строк:

1. Сравниваются первые символы строк. Если символы не равны, то большей будет та строка, в которой больше первый символ. Сравнение завершено.
2. Иначе если первые символы совпали, аналогично проверяем вторые символы. Продолжаем, пока не найдём несовпадение или не закончится одна из строк.
3. Если строки закончились одновременно, то они равны. Если закончилась одна из строк, то большей строкой считается строка с большим количеством символов.


1 "а" > "ф"; // false
2 "А" < "я"; // true
3 "Кот" < "кот"; // true
4 "код" === "кот"; // false
5 "код" === "код"; // true

German Alphabet					
Aa ah	Ää ah Umlaut	Bb beh	ß ess-testt	Cc tseh	Dd deh
Ee eh	Ff eff	Gg geh	Hh ha	Ii ee	Jj yot
Kk kah	Ll ell	Mm emm	Nn enn	Oo oh	Öö oh umlaut
Pp peh	Qq kuh	Rr err	Ss ess	Tt teh	Uu uh
Üü uh Umlaut	Vv fow	Ww veh	Xx iks	Yy upsilon	Zz tsett

```
1  "Äpfel" > "Zucker"; // true
```

Правильное сравнение строк

Для правильного сравнения строк следует использовать метод str1.localeCompare(str2). Данный метод сравнивает язык с учётом языка пользователя. По умолчанию берётся язык браузера.
Возвращается:

- -1, если str1 меньше str2
- +1, если str1 больше str2
- 0, если строки равны

```
> "Äpfel".localeCompare("Zucker")
< -1
```

Unicode

В JavaScript строки поддерживают символы, заданные в формате unicode.
Данные символы начинаются с \u
С таблицей юникод символов можно ознакомиться, например, здесь

```
alert('\u{1F916}') 🤖
alert('\u00e4') ä
```

Регулярные выражения

Регулярные выражения - это шаблоны, используемые для сопоставления последовательностей символов в строках

Регулярные выражения создаются при помощи литерала либо с использованием функции конструктора

```
1  const reg = /ab+c/;
2  const regexp = new RegExp("ab+c");
```

Простые шаблоны

Простые шаблоны предназначены для поиска прямого соответствия в тексте

```
1  /абыр/.test('абырвалг'); // true
2  /абыр/.test('рыба'); // false
```

Работа с регулярными выражениями

Метод str.match(regexp) для строки str возвращает совпадения с шаблоном regexp

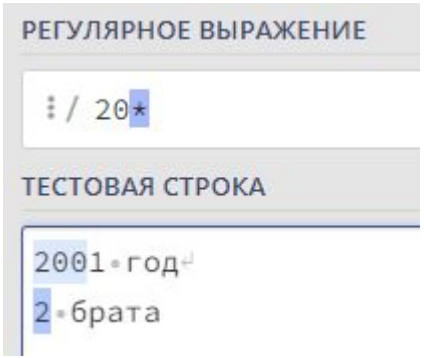
Метод str.replace(regexp, replacement) заменяет совпадения regexp в строке str на replacement

Метод regexp.test(str) проверяет, есть ли хоть одно совпадение. Если да, то true. Иначе false

Специальные символы в регулярных выражениях

Специальный символ * соответствует предыдущему символу, повторённому 0 или более раз

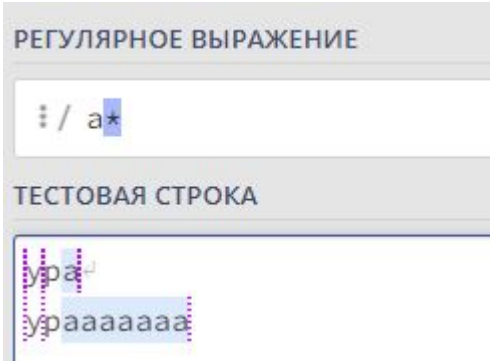
Например, /20*/ соответствует ‘200’ в строке ‘2001 год’ и ‘2’ в строке ‘2 брата’



Специальные символы в регулярных выражениях

Специальный символ + соответствует предыдущему символу, повторенному 1 или более раз

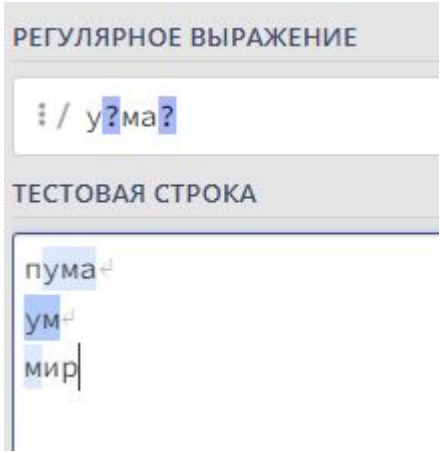
Например, /a+/ соответствует ‘a’ в слове ‘ура’ и всем буквам ‘a’ в слове ‘ураааааа’



Специальные символы в регулярных выражениях

Специальный символ ? соответствует предыдущему символу, повторенному 0 или 1 раз

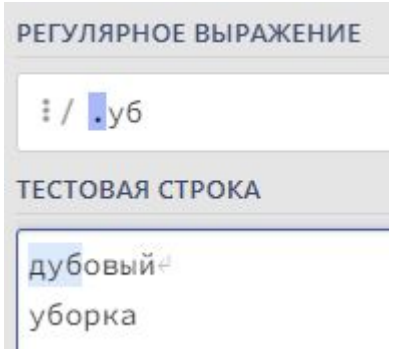
Например /у?ма?/ соответствует ‘ум’ в слове ‘ум’, ‘ума’ в слове ‘пума’, а также ‘м’ в слове мир



Специальные символы в регулярных выражениях

Специальный символ . соответветствует любому символу кроме перевода строки

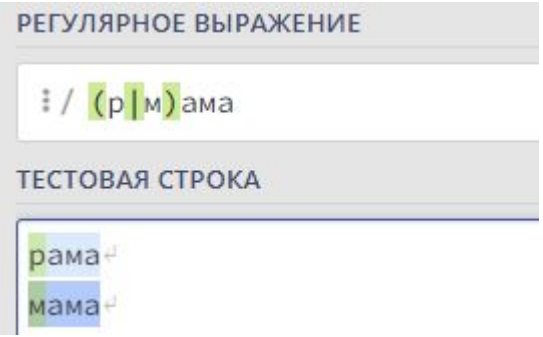
Например /.уб/ соответствует ‘дуб’ в слове ‘дубовый’, но не соответствует ‘уб’ в слове ‘уборка'



Специальные символы в регулярных выражениях

Специальный символ | в выражении x|y соответствует x или y

Например /(р|м)ама/ соответствует словам ‘рама’ и ‘мама’



Флаги регулярных выражений

- i - включает игнорирование регистра
- g - с этим флагом ищет все совпадения. Иначе только первое

Символьные выражения

\d - соответствует любой одной цифре

\s - соответствует любому пробельному символу

\w - соответствует любому символу, записанному латиницей

Например /\d+\s\w/ соответствует строке ‘31 may’

/\d/g в строке ‘+7(233)-233-23-45’ соответствует
‘7’,’2’,’3’,’3’,’3’,’2’,’3’,’3’,’2’,’3’,’4’,’5’

Работа с регулярными выражениями

```
'+7(233)-233-23-45'.replace(/\d/g,'#'); // '+#(###)-###-##-##'
'string'.match(/ri/); // ['ri', index: 2, input: 'string', groups: undefined]

function isDigit(c) {
    return /\d/.test(c)
}

```