STRING		
СВОЙСТВА STRING		
Имя	**Сводка и синтаксис**	**Пример**
String.length	***Сводка*** Свойство length представляет длину строки. ***Синтаксис:*** str.length	var x = 'Mozilla'; var empty = ''; console.log('Слово «Mozilla» занимает ' + x.length + ' кодовых значений'); /* "Слово «Mozilla» занимает 7 кодовых значений" */ console.log('Пустая строка имеет длину, равную ' + empty.length); /* "Пустая строка имеет длину, равную 0" */
МЕТОДЫ STRING		

String.prototype[@@iterator]()	***Сводка***
	Метод [@@iterator]() возвращает новый объект итератора Iterator, который проходит по кодовым точкам строкового значения, возвращая каждую кодовую точку в виде строкового значения.
	
	Синтаксис:
	string[Symbol.iterator]
		var string = 'A\uD835\uDC68';
		var strIter = string[Symbol.iterator]();
		console.log(strIter.next().value); // "A"
		console.log(strIter.next().value); // "\uD835\uDC68"
		
		var string = 'A\uD835\uDC68B\uD835\uDC69C\uD835\uDC6A';
		for (var v of string) {
		  console.log(v);
		}
		// "A"
		// "\uD835\uDC68"
		// "B"
		// "\uD835\uDC69"
		// "C"
		// "\uD835\uDC6A"

String.prototype.toString()	***Сводка*** Метод toString() возвращает строку, представляющую указанный объект. ***Синтаксис:*** str.toString()	var x = new String('Привет, мир'); console.log(x.toString()); // Отобразит 'Привет, мир'
String.prototype.charAt()	***Сводка*** Метод charAt() возвращает указанный символ из строки. ***Синтаксис:*** str.charAt(index) ***Параметры:*** index Целое число от 0 до длины строки минус 1.	var anyString = 'Дивный новый мир'; console.log("Символ по индексу 0 равен '" + anyString.charAt(0) + "'"); Символ по индексу 0 равен 'Д'

String.prototype.charCodeAt()	***Сводка*** Метод charCodeAt() возвращает числовое значение Юникода для символа по указанному индексу (за исключением кодовых точек Юникода, больших 0x10000). ***Синтаксис:*** str.charCodeAt(index) ***Параметры:*** index Целое число больше, либо равное 0 и меньше длины строки; если параметр не является числом, он устанавливается в 0.	' ABC'.charCodeAt(0); // вернёт 65
String.prototype.concat()	***Сводка*** Метод concat() объединяет текст из двух или более строк и возвращает новую строку. ***Синтаксис:*** str.concat(string2, string3[, ..., stringN]) ***Параметры:*** string2...stringN Строки, объединяемые в эту строку.	var hello = 'Привет, '; console.log(hello.concat('Кевин', ', удачного дня.'));

String.prototype.includes()	***Сводка***
	Метод includes() проверяет, содержит ли строка заданную подстроку, и возвращает, соответственно true или false.
	
	Синтаксис:
	str.includes(searchString[, position])
	
	Параметры:
	searchString
	Строка для поиска в данной строке.
	position Необязательный
	Позиция в строке, с которой начинать поиск строки searchString, по умолчанию 0.
	
	Возвращаемое значение
	true, если искомая строка была найдена в данной строке; иначе false.
		var str = 'Быть или не быть вот в чём вопрос.';
		
		console.log(str.includes('Быть')); // true
		
		console.log(str.includes('вопрос')); // true
		
		console.log(str.includes('несуществующий')); // false
		
		console.log(str.includes('Быть', 1)); // false
		
		console.log(str.includes('БЫТЬ')); // false

String.prototype.indexOf()	***Сводка***
	Метод indexOf() возвращает индекс первого вхождения указанного значения в строковый объект String, на котором он был вызван, начиная с индекса fromIndex. Возвращает -1, если значение не найдено.
	
	Синтаксис:
	str.indexOf(searchValue, [fromIndex])
	
	Параметры:
	searchValue
	Строка, представляющая искомое значение.
	fromIndex
	Необязательный параметр. Местоположение внутри строки, откуда начинать поиск. Может быть любым целым числом. Значение по умолчанию установлено в 0. Если fromIndex < 0, поиск ведётся по всей строке (так же, как если бы был передан 0). Если fromIndex >= str.length, метод вернёт -1, но только в том случае, если searchValue не равен пустой строке, в этом случае он вернёт str.length.
		var anyString = 'Дивный новый мир';
		
		console.log('Индекс первого вхождения «й» с начала строки равен ' + anyString.indexOf('й'));
		// Отобразит 5
		
		console.log('Индекс первого вхождения «й» с конца строки равен ' + anyString.lastIndexOf('й'));
		// Отобразит 11

String.prototype.lastIndexOf()	***Сводка***
	Метод lastIndexOf() возвращает индекс последнего вхождения указанного значения в строковый объект String, на котором он был вызван, или -1, если ничего не было найдено. Поиск по строке ведётся от конца к началу, начиная с индекса fromIndex.
	
	Синтаксис:
	str.lastIndexOf(searchValue[, fromIndex])
	
	Параметры:
	searchValue
	Строка, представляющая искомое значение.
	
	fromIndex
	Необязательный параметр. Местоположение внутри строки, откуда начинать поиск, нумерация индексов идёт слева направо. Может быть любым целым числом. Значение по умолчанию установлено в str.length. Если оно отрицательно, трактуется как 0. Если fromIndex > str.length, параметр fromIndex будет трактоваться как str.length.
		var anyString = 'Дивный новый мир';
		
		console.log('Индекс первого вхождения «й» с начала строки равен ' + anyString.indexOf('й'));
		// Отобразит 5
		console.log('Индекс первого вхождения «й» с конца строки равен ' + anyString.lastIndexOf('й'));
		// Отобразит 11

String.prototype.match()	***Сводка***
	Метод match() возвращает получившиеся совпадения при сопоставлении строки с регулярным выражением.
	
	Синтаксис:
	str.match(regexp)
	
	Параметры:
	regexp
	Объект регулярного выражения. Если будет передан объект obj, не являющийся регулярным выражением, он будет неявно преобразован в объект RegExp через вызов конструктора new RegExp(obj).
	array
	Объект Array, содержащий результаты сопоставления, или null, если ничего не было сопоставлено.
		var str = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЬЫЪЭЮЯ абвгдеёжзийклмнопрстуфхцчшщьыъэюя';
		var regexp = /[А-Д]/gi;
		var matches_array = str.match(regexp);
		
		console.log(matches_array);
		// ['А', 'Б', 'В', 'Г', 'Д', 'а', 'б', 'в', 'г', 'д']

String.prototype.repeat()	***Сводка***
	Метод repeat() конструирует и возвращает новую строку, содержащую указанное количество соединённых вместе копий строки, на которой он был вызван.
	
	Синтаксис:
	str.repeat(count)
	
	Параметры:
	count
	Целое число от 0 до +∞: [0, +∞), определяющее число повторений строки во вновь создаваемой и возвращаемой строке.
	Новая строка, содержащая указанное количество копий строки, для которой был вызван метод.
		'абв'.repeat(-1); // RangeError
		'абв'.repeat(0); // ''
		'абв'.repeat(1); // 'абв'
		'абв'.repeat(2); // 'абвабв'
		'абв'.repeat(3.5); // 'абвабвабв' (количество будет преобразовано в целое число)
		'абв'.repeat(1/0); // RangeError
		
		({ toString: () => 'абв', repeat: String.prototype.repeat }).repeat(2);
		// 'абвабв' (метод repeat() является обобщённым методом)

String.prototype.search()	***Сводка***
	Метод search() выполняет поиск сопоставления между регулярным выражением и этим объектом String.
	
	Синтаксис:
	str.search([regexp])
	
	Параметры:
	regexp
	Необязательный параметр. Объект регулярного выражения. Если будет передан не объект регулярного выражения, он будет неявно преобразован в объект RegExp через вызов конструктора new RegExp(regexp).
		function testInput(re, str) {
		var midstring;
		if (str.search(re) != -1) {
		midstring = ' содержит ';
		} else {
		midstring = ' не содержит ';
		}
		console.log(str + midstring + re);
		}
		
		var testString = 'hey JuDe';
		var re = /[A-Z]/g;
		
		testInput(re, testString); // выведет: hey Jude содержит /[A-Z]/g

String.prototype.replace()

Сводка
Метод replace() возвращает новую строку с некоторыми или всеми сопоставлениями с шаблоном, заменёнными на заменитель. Шаблон может быть строкой или регулярным выражением, а заменитель может быть строкой или функцией, вызываемой при каждом сопоставлении.

Синтаксис:
str.replace(regexp|substr, newSubStr|function[, flags])

Параметры:
regexp
Объект регулярного выражения RegExp. Сопоставление заменяется возвращаемым значением второго параметра.
substr
Строка, заменяемая на newSubStr. Обратите внимание, будет заменено только первое вхождение искомой строки.
newSubStr
Строка, заменяющая подстроку из первого параметра. Поддерживает несколько специальных шаблонов замены; смотрите ниже раздел Передача строки в качестве второго параметра.
function
Функция, вызываемая для создания новой подстроки (помещаемой вместо подстроки из первого параметра). Аргументы, передаваемые функции, описаны ниже в разделе Передача функции в качестве второго параметра.
flags Non-standard
Обратите внимание: аргумент flags не работает в ядре v8 (движок JavaScript в Chrome и NodeJs). Строка, задающая комбинацию флагов регулярного выражения. Параметр flags в методе String.prototype.replace() является нестандартным расширением. Вместо использования этого параметра используйте объект RegExp с соответствующими флагами. Значение этого параметра, если он используется, должно быть строкой, состоящей из одного или более следующих символов, следующим образом влияющих на обработку регулярного выражения:
g
глобальное сопоставление
i
игнорировать регистр
m
сопоставление по нескольким строкам

```
var re = /яблоки/gi;
var str = 'Яблоки круглые и яблоки сочные.';
var newstr = str.replace(re, 'апельсины');
console.log(newstr);
// апельсины круглые и апельсины сочные.
```

```
// Ночь перед Рождеством, Xmas - сокращение для Christmas
var str = 'Twas the night before Xmas...';
var newstr = str.replace(/xmas/i, 'Christmas');
console.log(newstr); // Twas the night before Christmas...
```

```
var re = /([А-ЯЁа-яё]+)\s([А-ЯЁа-яё]+)/;
var str = 'Джон Смит';
var newstr = str.replace(re, '$2, $1');
console.log(newstr); // Смит, Джон
```

```
function styleHyphenFormat(propertyName) {
 function upperToHyphenLower(match) {
  return '-' + match.toLowerCase();
 }
 return propertyName.replace(/[A-Z]/g, upperToHyphenLower);
}

var newString = propertyName.replace(/[A-Z]/g, '-' + '$&'.toLowerCase());  // не работает
```

```
function f2c(x) {
 function convert(str, p1, offset, s) {
  return ((p1 - 32) * 5/9) + 'C';
 }
 var s = String(x);
 var test = /(\d+(?:\.\d*)?)F\b/g;
 return s.replace(test, convert);
}

var str = 'x-x_';
var retArr = [];
str.replace(/(x_*)|(-)/g, function(match, p1, p2) {
 if (p1) { retArr.push({ on: true, length: p1.length }); }
 if (p2) { retArr.push({ on: false, length: 1 }); }
});
console.log(retArr);
```

String.prototype.split()	***Сводка***
	Метод split() разбивает объект String на массив строк путём разделения строки указанной подстрокой.
	
	Синтаксис:
	str.split([separator[, limit]])
	
	Параметры:
	separator
	Необязательный параметр. Указывает символы, используемые в качестве разделителя внутри строки. Параметр separator может быть как строкой, так и регулярным выражением. Если параметр опущен, возвращённый массив будет содержать один элемент со всей строкой. Если параметр равен пустой строке, строка str будет преобразована в массив символов.
	limit
	Необязательный параметр. Целое число, определяющее ограничение на количество найденных подстрок. Метод split() всё равно разделяет строку на каждом сопоставлении с разделителем separator, но обрезает возвращаемый массив так, чтобы он содержал не более limit элементов.
		function splitString(stringToSplit, separator) {
		var arrayOfStrings = stringToSplit.split(separator);
		
		console.log('Оригинальная строка: "' + stringToSplit + '"');
		console.log('Разделитель: "' + separator + '"');
		console.log('Массив содержит ' + arrayOfStrings.length + ' элементов: ' + arrayOfStrings.join(' / '));
		}
		
		// Строчка из «Бури» Шекспира. Перевод Михаила Донского.
		var tempestString = 'И как хорош тот новый мир, где есть такие люди!';
		var monthString = 'Янв,Фев,Мар,Апр,Май,Июн,Июл,Авг,Сен,Окт,Ноя,Дек';
		
		var space = ' ';
		var comma = ',';
		
		splitString(tempestString, space);
		splitString(tempestString);
		splitString(monthString, comma);

String.prototype.toLowerCase()	***Сводка*** Метод toLowerCase() возвращает значение строки, на которой он был вызван, преобразованное в нижний регистр. ***Синтаксис:*** str.toLowerCase()	console.log('АЛФАВИТ'.toLowerCase()); // 'алфавит'
String.prototype.toUpperCase()	***Сводка*** Метод toUpperCase() возвращает значение строки, на которой он был вызван, преобразованное в верхний регистр. ***Синтаксис:*** str.toUpperCase()	console.log('алфавит'.toUpperCase()); // 'АЛФАВИТ'

String.prototype.trim()	***Сводка***
	Метод trim() удаляет пробельные символы с начала и конца строки. Пробельными символами в этом контексте считаются все собственно пробельные символы (пробел, табуляция, неразрывный пробел и прочие) и все символы конца строки (LF, CR и прочие).
	
	Синтаксис:
	str.trim()
		var orig = ' foo ';
		console.log(orig.trim()); // 'foo'
		
		// Другой пример, в котором .trim() убирает пробельные символы только с одной стороны.
		
		var orig = 'foo ';
		console.log(orig.trim()); // 'foo'

String.prototype.valueOf()	***Сводка*** Метод valueOf() возвращает примитивное значение объекта String. ***Синтаксис****:* str.valueOf()	var x = new String('Привет, мир'); console.log(x.valueOf()); // Отобразит 'Привет, мир'

String.prototype.slice()	***Сводка***
	Метод slice() извлекает часть строки и возвращает новую строку без изменения оригинальной строки.
	
	Синтаксис*:*
	str.slice(beginIndex[, endIndex])
	
	Параметры*:*
	beginIndex
	Индекс, с которого начинать извлечение (нумерация начинается с нуля). Если аргумент отрицателен, то трактуется как str.length + beginIndex (например, если beginIndex равен -3, то он трактуется как str.length - 3).
	Если beginIndex не является числом при проверке Number(beginIndex), он трактуется как 0.
	Если beginIndex больше или равен str.length, возвращается пустая строка.
	
	endIndex
	Индекс, перед которым заканчивать извлечение (нумерация начинается с нуля). Символ по этому индексу не будет включён.
	Если *endIndex *опущен или является undefined или больше чем str.length, slice() извлечёт всё до конца строки. Если аргумент отрицателен, то трактуется как str.length + endIndex (например, если endIndex равен -3, то он трактуется как str.length - 3). Если аргумент не undefined и не является числом при проверке Number(endIndex), возвращается пустая строка.
	Если endIndex указан и меньше startIndex, то возвращается пустая строка (например, slice(-1, -3) или slice(3, 1) вернут "").
	
	Возвращаемое значение
	Новая строка, содержащая извлечённую часть строки.
		let str1 = 'Приближается утро.';
		let str2 = str1.slice(1, 8);
		let str3 = str1.slice(4, -2);
		let str4 = str1.slice(12);
		let str5 = str1.slice(30);
		
		console.log(str2); // ВЫВОД: риближа
		console.log(str3); // ВЫВОД: лижается утр
		console.log(str4); // ВЫВОД: утро.
		console.log(str5); // ВЫВОД: ""