String.prototype[@@iterator]()

Метод [@@iterator]() возвращает новый объект итератора Iterator, который проходит по кодовым точкам строкового значения, возвращая каждую кодовую точку в виде строкового значения.

string[Symbol.iterator]

```
let string = 'Возвращает';
let strIter = string[Symbol.iterator]();

console.log(strIter.next().value); // В
console.log(strIter.next().value); // о
console.log(strIter.next().value); // з
```

String.prototype.at()

Возвращает символ по входному числу (индекс строки)

```
let str = 'Some text';
let index = 2;
let tindex = -2;
console.log(str.at(index)); // m
console.log(str.at(tindex)); // x
```

---

str.toLowerCase()

Преобразует символы в строке в нижний регистр.

```
let str = 'SoMe teXT';
console.log(str.toLowerCase()); // some text
```

str.toUpperCase()

Преобразует символы в строке в верхний регистр.

```
let str = 'some text';
console.log(str.toUpperCase()); // SOME TEXT
```

---

str1.concat(str2);

Объединяет две или более строки и возвращает одну строку.

```
let str1 = 'London ';
let str2 = 'is the ';
let str3 = 'capital of New York.'
console.log(str1.concat(str2)); // London is the
console.log(str1.concat(str2, str3)); // London is the capital of New York.
```

---

str.repeat(k);

Принимает в качестве параметра число (k) и повторяет строку указанное количество раз.

```
let str = 'cat';
console.log(str.repeat(4)); // catcatcatcat
```

---

mainStr.includes(includStr, indexFind);

Проверяет, содержит ли строка указанную подстроку. Возвращает значение true или false. Вторым параметром можно указать позицию в строке, с которой следует начать поиск.

```
let str = 'cat and dog';
console.log(str.includes('and')); // true
console.log(str.includes('and', 7)); // false
```

str.indexOf(symbol);

Возвращает индекс первого найденного вхождения указанного значения. Поиск ведётся от начала до конца строки. Если совпадений нет, возвращает -1. Вторым параметром можно передать позицию, с которой следует начать поиск.

```
let str = 'cat and dog';
console.log(str.indexOf('t')); // 2
console.log(str.indexOf('t', 3)); // -1
```

---

str.lastIndexOf(symbol);

Возвращает индекс последнего найденного вхождения указанного значения. Поиск ведётся от конца к началу строки. Если совпадений нет, возвращает -1. Вторым параметром можно передать позицию, с которой следует начать поиск.

```
let str = 'cat and dog';
console.log(str.lastIndexOf('d')); // 8
console.log(str.lastIndexOf('a', 3)); // 1
```

---

str1.endsWith(str2);

Проверяет, заканчивается ли строка символами, заданными первым параметром. Возвращает true или false. Есть второй необязательный параметр — ограничитель по диапазону поиска. По умолчанию он равен длине строки.

```
let str = 'cat and dog';
console.log(str.endsWith('og')); // true
console.log(str.endsWith('cat', 3)); // true
```

str1.startsWith(str2);

Проверяет, начинается ли строка с указанных символов. Возвращает true или false. Вторым параметром можно указать индекс, с которого следует начать проверку.

```
let str = 'cat and dog';
console.log(str.startsWith('cat')); // true
console.log(str.startsWith('and', 4)); // true
```

---

mainStr.search(subStr);

Проверяет, есть ли в строке указанное значение или регулярное выражение и возвращает индекс начала совпадения.

```
let str = 'cat and dog';
console.log(str.search('and')); // 4
console.log(str.search('gas')); // -1
```

---

str.slice(begin, end);

Извлекает часть строки и возвращает новую строку. **Обязательный параметр** — начало извлечения. Вторым параметром можно установить границу (по умолчанию — до конца строки). **Отрицательные значения тоже работают**

```
let str = 'cat and dog';
console.log(str.slice(5)); // nd dog
console.log(str.slice(0, 9)); // cat and d
```

str.substring(begin, end);

Извлекает символы из строки между двумя указанными индексами. Второй индекс указывать не обязательно. В таком случае будут извлечены все символы от начала до конца строки. В отличие от slice, можно задавать start больше, чем end. Отрицательные значения не поддерживаются, они интерпретируются как 0.

```
let str = 'cat and dog';
console.log(str.substring(5)); // nd dog
console.log(str.substring(9, 0)); // cat and d
```

---

str.replace(subStrOld, subStrNew);

Ищет в строке указанное значение или регулярное выражение и возвращает новую строку, в которой выполнена замена на второй параметр. Можно заменить найденные значения другой строкой или передать функцию для работы над совпадениями.

```
let str = 'cat and dog';
console.log(str.replace('and', 'not')); // cat not dog
```

str.replaceAll(subStrOld, subStrNew);

Даёт такой же результат, как метод replace() с глобальным флагом g. Заменяет все найденные совпадения другой строкой или переданной функцией.

---

str.padEnd(k, symbol);

Добавляет в конце отступы, пока строка не достигнет длины, заданной первым параметром. Вторым параметром можно указать другой символ вместо пробела.

```
let str = 'cat and dog';
console.log(str.padEnd(25, 'o_O')); // cat and dogo_Oo_Oo_Oo_Oo_
```

```
str.padStart(num, substr);
```

Добавляет в начале отступы, пока строка не достигнет длины, заданной первым параметром. Вторым параметром можно указать другой символ вместо пробела.

```
let str = 'cat and dog';
console.log(str.padStart(25, 'o_O')); // o_Oo_Oo_Oo_Oo_cat and dog
```

```
str.trim();
```

Обрезает пробелы с обоих концов строки.

```
str.trimEnd();
```

Обрезает пробелы в конце строки

```
str.trimStart()
```

Обрезает пробелы в начале строки
