Method Category	**Method**	**Description**
Manipulating Strings	concat()	Combines multiple strings and returns a new string.
	slice()	Extracts a section of the string and returns a new string.
	substring()	Extracts a section of the string. Handles negative values differently than `slice()`.
	replace()	Replaces occurrences of a specified substring with another substring.
	toUpperCase()	Converts the string to uppercase.
	toLowerCase()	Converts the string to lowercase.
	trim()	Removes whitespace from both ends of the string.
	padStart()	Pads the beginning of the string with a specified character or string to a desired length.
	padEnd()	Pads the end of the string with a specified character or string to a desired length.
Searching and Checking	includes()	Checks if a string contains a specified substring.
	indexOf()	Returns the index of the first occurrence of a specified substring.
	lastIndexOf()	Returns the index of the last occurrence of a specified substring. [1]
	startsWith()	Checks if a string starts with a specified substring.
	endsWith()	Checks if a string ends with a specified substring.
Splitting and Joining	split()	Splits a string into an array of substrings based on a specified separator.
	join()	Joins the elements of an array into a single string, using a specified separator.

Manipulating Strings

1. concat() - Combine multiple strings.

Description: The `concat()` method is used to join two or more strings into a single string.

Syntax:

```javascript
let newString = string1.concat(string2, string3, ..., stringN);
```

Example:

```javascript
let str1 = 'Hello';
let str2 = 'World';
let combined = str1.concat(', ', str2, '!');
console.log(combined); // 'Hello, World!'
```

2. slice() - Extract a section of the string.

Description: The `slice()` method returns a portion of the string, selected from the start index to the end index (exclusive). It does not modify the original string.

Syntax:

```javascript
let newString = string.slice(beginIndex, endIndex);
```

Example:

```javascript
let str = 'Hello, World!';
let sliced = str.slice(7, 12);
console.log(sliced); // 'World'
```

3. substring() - Extract a section of the string (handles negative values differently than `slice()`).

Description: The `substring()` method is similar to `slice()`, but it does not allow negative values for indexes. If negative values are provided, it treats them as `0`.

Syntax:

javascript Copy code

```javascript
let newString = string.substring(startIndex, endIndex);
```

Example:

javascript Copy code

```javascript
let str = 'Hello, World!';
let substring1 = str.substring(0, 5);
console.log(substring1); // 'Hello'

let substring2 = str.substring(7, 12);
console.log(substring2); // 'World'

// If negative values are used, substring treats them as 0
let substring3 = str.substring(-5, 12);
console.log(substring3); // 'Hello, World!'
```

4. replace() - Replace a substring with another substring.

Description: The `replace()` method searches for a specified substring or pattern and replaces it with another substring. It replaces only the **first occurrence** by default.

Syntax:

javascript Copy code

```javascript
let newString = string.replace(searchValue, newValue);
```

- `searchValue`: A substring or regular expression to search for.
- `newValue`: The string that will replace the `searchValue`.

Example:

javascript Copy code

```javascript
let str = 'Hello, World!';
let newStr = str.replace('World', 'JavaScript');
console.log(newStr); // 'Hello, JavaScript!'

// Using regular expression for case-insensitive replacement
let str2 = 'hello, world!';
let newStr2 = str2.replace(/world/i, 'JavaScript');
console.log(newStr2); // 'hello, JavaScript!'
```

5. toUpperCase() - Convert the string to uppercase.

Description: The `toUpperCase()` method converts all characters in the string to uppercase letters.

Syntax:

javascript Copy code

```javascript
let upperCaseString = string.toUpperCase();
```

Example:

javascript Copy code

```javascript
let str = 'Hello, World!';
let upperStr = str.toUpperCase();
console.log(upperStr); // 'HELLO, WORLD!'
```

6. toLowerCase() - Convert the string to lowercase.

Description: The `toLowerCase()` method converts all characters in the string to lowercase letters.

Syntax:

javascript Copy code

```javascript
let lowerCaseString = string.toLowerCase();
```

Example:

```javascript
let str = 'Hello, World!';
let lowerStr = str.toLowerCase();
console.log(lowerStr); // 'hello, world!'
```

7. trim() - Remove whitespace from both ends of the string.

Description: The `trim()` method removes whitespace characters (spaces, tabs, line breaks) from the beginning and end of the string.

Syntax:

```javascript
let trimmedString = string.trim();
```

Example:

```javascript
let str = '   Hello, World!   ';
let trimmedStr = str.trim();
console.log(trimmedStr); // 'Hello, World!'
```

8. padStart() - Pad the string from the beginning.

Description: The `padStart()` method pads the string with a specified character (or characters) until it reaches the given length. The padding is applied at the start of the string.

Syntax:

```javascript
let paddedString = string.padStart(targetLength, padString);
```

- `targetLength`: The desired length of the resulting string.
- `padString` (optional): The string used for padding (defaults to a space).

Example:

javascript

```javascript
let str = '5';
let paddedStr = str.padStart(3, '0');
console.log(paddedStr); // '005'

let str2 = 'Hello';
let paddedStr2 = str2.padStart(10, '*');
console.log(paddedStr2); // '*****Hello'
```

9. padEnd() - Pad the string from the end.

Description: The `padEnd()` method pads the string with a specified character (or characters) until it reaches the given length. The padding is applied at the end of the string.

Syntax:

javascript

```javascript
let paddedString = string.padEnd(targetLength, padString);
```

- `targetLength`: The desired length of the resulting string.
- `padString` (optional): The string used for padding (defaults to a space).

Example:

javascript

```javascript
let str = '5';
let paddedStr = str.padEnd(3, '0');
console.log(paddedStr); // '500'

let str2 = 'Hello';
let paddedStr2 = str2.padEnd(10, '*');
console.log(paddedStr2); // 'Hello*****'
```

Searching and Checking

1. includes() - Check if a substring exists.

Description: The `includes()` method checks if a given substring exists within the string. It returns `true` if the substring is found, otherwise `false`.

Syntax:

javascript Copy code

```javascript
let result = string.includes(searchValue, fromIndex);
```

- `searchValue`: The substring you want to check for.
- `fromIndex` (optional): The position to start searching from (default is `0`).

Example:

javascript Copy code

```javascript
let str = 'Hello, World!';
console.log(str.includes('World')); // true
console.log(str.includes('JavaScript')); // false

let str2 = 'JavaScript is great!';
console.log(str2.includes('Script', 4)); // true (search starts from index 4)
```

2. indexOf() - Find the first index of a substring.

Description: The `indexOf()` method returns the **first index** of the substring (or character) within the string. If the substring is not found, it returns `-1`.

Syntax:

javascript Copy code

```javascript
let index = string.indexOf(searchValue, fromIndex);
```

- `searchValue`: The substring or character to search for.
- `fromIndex` (optional): The index to start searching from (default is `0`).

Example:

javascript Copy code

```javascript
let str = 'Hello, World!';
console.log(str.indexOf('World')); // 7
console.log(str.indexOf('JavaScript')); // -1

let str2 = 'JavaScript is great!';
console.log(str2.indexOf('Script')); // 4
```

3. lastIndexOf() - Find the last index of a substring.

Description: The `lastIndexOf()` method works like `indexOf()`, but it searches for the substring from the **end of the string**, returning the last index where the substring appears.

Syntax:

javascript Copy code

```javascript
let index = string.lastIndexOf(searchValue, fromIndex);
```

- `searchValue`: The substring or character to search for.
- `fromIndex` (optional): The position to start searching from, counting from the end of the string.

Example:

javascript Copy code

```javascript
let str = 'Hello, World! Hello!';
console.log(str.lastIndexOf('Hello')); // 14 (last occurrence of 'Hello')
console.log(str.lastIndexOf('World')); // 7

let str2 = 'JavaScript is great, and JavaScript is powerful!';
console.log(str2.lastIndexOf('JavaScript')); // 35
```

4. startsWith() - Check if the string starts with a given substring.

Description: The `startsWith()` method checks if the string begins with a specified substring. It returns `true` if the string starts with the substring, otherwise `false`.

Syntax:

javascript Copy code

```javascript
let result = string.startsWith(searchValue, fromIndex);
```

- `searchValue`: The substring to check at the start.
- `fromIndex` (optional): The index at which to start the search (default is `0`).

Example:

javascript Copy code

```javascript
let str = 'Hello, World!';
console.log(str.startsWith('Hello')); // true
console.log(str.startsWith('World')); // false

let str2 = 'JavaScript is awesome!';
console.log(str2.startsWith('JavaScript')); // true
console.log(str2.startsWith('is', 11)); // true (starts checking from index 11)
```

5. endsWith() - Check if the string ends with a given substring.

Description: The `endsWith()` method checks if the string ends with a specific substring. It returns `true` if the string ends with the substring, otherwise `false`.

Syntax:

javascript Copy code

```javascript
let result = string.endsWith(searchValue, length);
```

- `searchValue`: The substring to check at the end.
- `length` (optional): The length of the string to consider for checking the end (default is the full string length).

Example:

javascript

```javascript
let str = 'Hello, World!';
console.log(str.endsWith('World!')); // true
console.log(str.endsWith('Hello')); // false

let str2 = 'JavaScript is awesome!';
console.log(str2.endsWith('awesome!')); // true
console.log(str2.endsWith('is', 15)); // true (checking first 15 characters)
```

Splitting and Joining

1. split() - Split the string into an array of substrings.

Description: The `split()` method splits a string into an array of substrings, based on a specified delimiter or separator. The delimiter can be a string or a regular expression. If the delimiter is not found, the entire string is returned as a single element in an array.

Syntax:

javascript Copy code

```javascript
let result = string.split(separator, limit);
```

- `separator`: The string or regular expression used to split the string (can be any character or pattern).
- `limit` (optional): The maximum number of elements in the returned array.

Example:

javascript Copy code

```javascript
let str = 'apple,orange,banana';
let result = str.split(','); // Split by comma
console.log(result); // ['apple', 'orange', 'banana']

let str2 = 'Hello World!';
let result2 = str2.split(' '); // Split by space
console.log(result2); // ['Hello', 'World!']

let str3 = 'a,b,c,d,e';
let result3 = str3.split(',', 3); // Split by comma, limit to 3 elements
console.log(result3); // ['a', 'b', 'c']
```

2. join() - Join an array of substrings into a single string.

Description: The `join()` method combines all elements of an array into a single string, with an optional separator between each element. If no separator is provided, the array elements are joined with commas by default.

Syntax:

javascript Copy code

```javascript
let result = array.join(separator);
```

- `separator` (optional): The string that separates each element in the joined string (default is `,`).

Example:

javascript Copy code

```javascript
let arr = ['apple', 'orange', 'banana'];
let result = arr.join(', '); // Join elements with a comma and a space
console.log(result); // 'apple, orange, banana'

let arr2 = ['Hello', 'World!'];
let result2 = arr2.join(' '); // Join elements with a space
console.log(result2); // 'Hello World!'

let arr3 = ['a', 'b', 'c'];
let result3 = arr3.join('-'); // Join elements with a hyphen
console.log(result3); // 'a-b-c'

let arr4 = ['1', '2', '3'];
let result4 = arr4.join(); // Default separator is comma
console.log(result4); // '1,2,3'
```