A COMPREHENSIVE GUIDE TO JAVASCRIPT STRING METHODS

Categorized by purpose for easy understanding.

String Access & Search

Methods to find characters, check for substrings, and locate indices within a string (e.g., charAt, includes, indexOf, search, match).

String Extraction & Slicing

Methods to extract parts of a string and return them as new strings (e.g., slice, substring, substr).

String Modification

Methods to transform strings, replace content, change case, and remove whitespace (e.g., concat, replace, toLowerCase, trim, padStart).

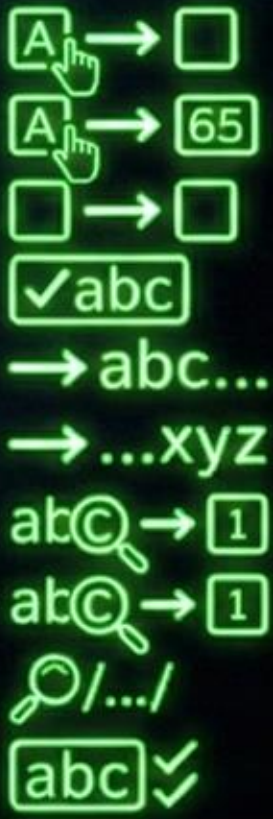
String Splitting

Method to split a string into an array of substrings based on a separator (e.g., split).

Note: In JavaScript, strings are immutable; all string methods are Non-Mutating (they return a new string).

STRING ACCESS & SEARCH

- charAt() (Non-Mutating)
- charCodeAt() (Non-Mutating)
- at() (Non-Mutating)
- includes() (Non-Mutating)
- startsWith() (Non-Mutating)
- endsWith() (Non-Mutating)
- indexOf() (Non-Mutating)
- lastIndexOf() (Non-Mutating)
- search() (Non-Mutating)
- match() (Non-Mutating)



STRING ACCESS & SEARCH: charAt(), at(), & charCodeAt()

• charAt() (Non-Mutating)

Returns the character at the specified index.


```
const str = 'Neon'; str.charAt(1) → 'e'
```

• at() (Non-Mutating)

Returns the character at the specified index, accepting negative integers.


```
const str = 'Neon'; str.at(-1) → 'n'
```

• charCodeAt() (Non-Mutating)

Returns the Unicode of the character at the specified index.

```
const str = 'Neon'
str.charCodeAt(1) → 101
```

STRING ACCESS & SEARCH: includes(), startsWith() & endsWith()

• includes() (Non-Mutating)

Checks if a string contains a specified substring.

```
str.includes('eo') → true
```

• startsWith() (Non-Mutating)

Checks if a string starts with a specified substring.

```
str.startsWith('Ne') → true
```

• endsWith() (Non-Mutating)

Checks if a string ends with a specified substring.

```
const str = 'Neon';
str.endsWith('on') → true
```

STRING ACCESS & SEARCH: indexOf() & lastIndexOf()

• indexOf() (Non-Mutating)

Returns the index of the first occurrence of a specified value.


```
const str = 'banana';

str.indexOf('a') → 1
```

• lastIndexOf() (Non-Mutating)

Returns the index of the last occurrence of a specified value.


```
const str = 'banana';

str.lastIndexOf('a') → 5
```

STRING ACCESS & SEARCH: search() & match()

• search() (Non-Mutating)

Searches for a match against a regular expression and returns the index.


```
const str = 'hello world';

str.search(/world/) → 6
```

• match() (Non-Mutating)

Searches for a match against a regular expression and returns an array of matches.

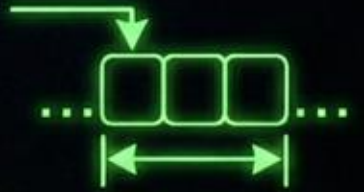
```
const str = 'hello world';

str.match(/l/g) → ['l', 'l', 'l']
```

STRING EXTRACTION & SLICING

- slice() (Non-Mutating)

- substring() (Non-Mutating)

- substr() (Non-Mutating)



STRING EXTRACTION & SLICING: slice(), substring() & substr()

• slice() (Non-Mutating)

Extracts a section of a string and returns it as a new string, accepting negative indices.

```
const str = 'NeonLight';
str.slice(1, 5) → 'eonL'
```

• substring() (Non-Mutating)

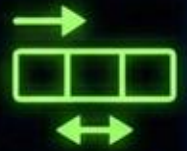
Similar to slice(), but does not accept negative indices.

```
const str = 'NeonLight';
str.substring(1, 5) → 'eonL'
```

• substr() (Non-Mutating)

Extracts a part of a string, starting at a specified index and extending for a given number of characters.

```
const str = 'NeonLight';
str.substr(1, 3) → 'eon'
```



STRING MODIFICATION (NON-MUTATING)

- concat() (Non-Mutating)
- replace() (Non-Mutating)
- replaceAll() (Non-Mutating)
- toLowerCase() (Non-Mutating)
- toUpperCase() (Non-Mutating)
- trim() (Non-Mutating)
- trimStart() (Non-Mutating)
- trimEnd() (Non-Mutating)
- padStart() (Non-Mutating)
- padEnd() (Non-Mutating)
- repeat() (Non-Mutating)

STRING MODIFICATION: concat()

- `concat()` (Non-Mutating)

(Non-Mutating)

Joins two or more strings and returns a new combined string.

```
const str1 = 'Hello';
const str2 = 'World';

str1.concat(' ', str2) → 'Hello World'
```

STRING MODIFICATION: replace() & replaceAll()

• replace() (Non-Mutating)

Replaces the first occurrence of a specified value or regular expression with a new value.


```
const str = 'banana';
str.replace('a', 'o') →
  'bonana'
```

• replaceAll() (Non-Mutating)

Replaces all occurrences of a specified value or regular expression with a new value.


```
const str = 'banana';
str.replaceAll('a', 'o') →
  'bonono'
```

STRING MODIFICATION: toLowerCase() & toUpperCase()

- `toLowerCase()` (Non-Mutating)

Converts all characters in a string to lowercase.

ABC → abc

```
const str = 'HELLO';
str.toLowerCase() → 'hello'
```

- `toUpperCase()` (Non-Mutating)

Converts all characters in a string to uppercase.

abc → ABC

```
const str = 'hello';
str.toUpperCase() → 'HELLO'
```

STRING MODIFICATION: trim(), trimStart() & trimEnd()

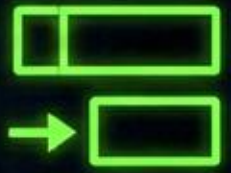
• trim() (Non-Mutating)

Removes whitespace from both the start and end of a string.

```
const str = '  hi  ';
str.trim() → 'hi'
```

• trimStart() (Non-Mutating)

Removes whitespace from the start of a string.

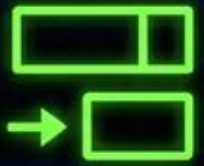


```
const str = '  hi';
str.trimStart() → 'hi'
```

• trimEnd() (Non-Mutating)

Removes whitespace from the end of a string.

```
const str = 'hi  ';
str.trimEnd() → 'hi'
```



STRING MODIFICATION: padStart() & padEnd()

• padStart() (Non-Mutating)

Pads the start of a string with another string until it reaches a given length.

```
const str = '5';
str.padStart(2, '0') → '05'
```

• padEnd() (Non-Mutating)

Pads the end of a string with another string until it reaches a given length.

```
const str = '5';
str.padEnd(2, '0') → '50'
```

STRING MODIFICATION: repeat()

- `repeat()` (Non-Mutating)

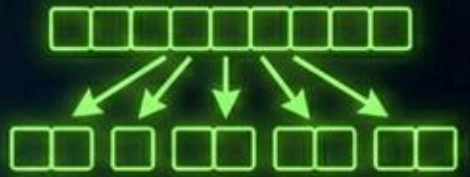
(Non-Mutating)

Returns a new string with a specified number of copies of the original string concatenated together.

```
const str = 'Na';

str.repeat(4) → 'NaNaNaNa'
```

STRING SPLITTING & JOINING

- split() (Non-Mutating)



STRING SPLITTING & JOINING: split()

- `split()` (Non-Mutating)

  Splits a string into an array of substrings based on a specified separator.

```
const str = 'a,b,c';

str.split(',') → ['a', 'b', 'c']
```