perplexity

Strings in js in detail, representation, template literals , methods, functions, facts, key points

JavaScript Strings: Detailed Guide

1. Representation of Strings in JavaScript

- **Definition:** A string in JavaScript is a sequence of characters, used to represent text such as words, sentences, symbols, or numbers[1] [2] [3].
- **Internal Storage:** JavaScript strings are sequences of UTF-16 code units. Each character is represented either by a single 16-bit unit or a pair for characters outside the basic multilingual plane (surrogate pairs)[4] [5] [6].
- **No Char Type:** Unlike languages like C or Java, JavaScript does not have a separate character type; a single character is just a string of length 1[1] [6].

Creation:

```
let s1 = 'hello';       // Single quotes
let s2 = "world";       // Double quotes
let s3 = `template`;    // Backticks (template literals)
let sObj = new String('object'); // String object (not recommended)
```

- **Note:** Use of `new String()` creates a string object, which behaves differently from a primitive string and may cause unexpected comparison results. It's recommended to create strings using literals[1] [2].

2. Template Literals

- **Syntax:** Enclosed by backticks (`), introduced in ES6[7] [8] [9].
- **Features:**
  - **Multi-line Strings:** Line breaks are preserved; no need for escape characters for new lines.
  - **String Interpolation:** Embed variables and expressions directly using `${...}`.
  - **Tagged Templates:** Apply a function to process the content of the template literal.

```
const user = 'Alex';
const greet = `Hello, ${user}!`; // Interpolates variable
// Multi-line
const msg = `Line 1
Line 2`;
// Tagged template
```

```
function tag(parts, ...subs) { /* custom logic */ }
tag`User: ${user}`;
```

- **Escaping Backticks:** Use ``` inside template literals[7] [8] [9].

3. String Methods and Functions

JavaScript strings are **immutable**. String methods never modify the original string, but return new ones[10] [1] [11].

Common Methods

Method	Description	Example
`length`	Length of the string	`"JS".length // 2`
`charAt(i)`	Char at index	`"JS".charAt(0) // "J"`
`charCodeAt(i)`	UTF-16 code at index	`"JS".charCodeAt(0) // 74`
`codePointAt(i)`	Unicode code point at index	
`concat()`	Concatenate strings	`"A".concat("B") // "AB"`
`slice(start, end)`	Extract section (supports negative indices)	`"hello".slice(1,3) // "el"`
`substring()`	Extract section (no negative indices)	`"hello".substring(1,3) // "el"`
`substr()`	Deprecated: substring by start index & length	
`toUpperCase()`	Uppercase	`"abc".toUpperCase() // "ABC"`
`toLowerCase()`	Lowercase	`"ABC".toLowerCase() // "abc"`
`trim()`	Remove surrounding white space	`" x ".trim() // "x"`
`trimStart()`	Remove leading spaces	`" x".trimStart() // "x"`
`trimEnd()`	Remove trailing spaces	`"x ".trimEnd() // "x"`
`padStart(len,s)`	Pad from the start to given length	`"5".padStart(3,"0") // "005"`
`padEnd(len,s)`	Pad from the end	`"5".padEnd(3,"0") // "500"`
`repeat(n)`	Repeat string n times	`"ha".repeat(3) // "hahaha"`
`replace(a, b)`	Replace first occurrence	`"abc".replace("a","z") // "zbc"`
`replaceAll(a,b)`	Replace all occurrences	`"aa".replaceAll("a","z") // "zz"`
`split(sep)`	Splits string into array using separator	`"a,b,c".split(",") // ["a","b","c"]`
`indexOf(s)`	Index of first occurrence	`"hello".indexOf("e") // 1`
`lastIndexOf(s)`	Index of last occurrence	`"hello".lastIndexOf("l") // 3`

Method	Description	Example
`includes(s)`	Check if substring exists	`"abc".includes("b") // true`
`startsWith(s)`	Starts with substring	`"hello".startsWith("he") // true`
`endsWith(s)`	Ends with substring	`"hello".endsWith("lo") // true`
`match(regex)`	Matches regex	`"abc".match(/b/) // Array|Null`
`search(regex)`	Index of regex match	`"abc".search(/b/) // 1`
`fromCharCode()`	Create string from UTF-16 units	`String.fromCharCode(72,73) // "HI"`
`valueOf()`	Gets primitive string from object	`(new String("abc")).valueOf() // "abc"`
`at(n)`	Returns character at position n (supports negatives)	`"abc".at(-1) // "c"`

- See full list and differences between slice, substring, and substr for edge cases[10] [12] [13] [11] [14] [15].

4. Interesting Facts & Key Points

- **Immutability:** Strings cannot be changed after creation. All string-manipulation methods return new strings[1] [11].
- **UTF-16 Encoding:** All JS strings use UTF-16, which can have surrogate pairs for characters outside the basic multilingual plane[4] [6].
- **Type Coercion:** Many operations auto-convert data types to strings (like using `String(x)` or template literals for interpolation)[4].
- **No Character Type:** Single characters are still strings, not a special type[1] [6].
- **Auto-boxing:** When you use string methods, primitives are temporarily auto-converted to String objects to allow method usage[14].
- **Template Literals:** Preferred for dynamic, multi-line, and readable strings[7] [9] [8].
- **Well-formed Strings:** Use `String.prototype.isWellFormed()` to check if string does not have lone surrogates; `toWellFormed()` to fix them (ES2022+)[4] [10].

5. Essential Examples

```
// Traditional String
let str = "Hello, World!";
console.log(str.length);           // 13
console.log(str.charAt(1));        // "e"

// Template Literal
let name = "Sam";
let msg = `Hi, ${name}!`;          // "Hi, Sam!"

// Methods
```

```
console.log(str.slice(7, 12));     // "World"
console.log(str.toUpperCase());    // "HELLO, WORLD!"
console.log(str.repeat(2));        // "Hello, World!Hello, World!"
console.log(str.split(", "));      // ["Hello", "World!"]
```

In summary:
JavaScript strings are versatile and powerful, supporting Unicode text, template literals for modern syntax, and a comprehensive library of methods for manipulation and inspection. They are always immutable and should be handled with an understanding of their UTF-16 foundation and encoding implications for Unicode and emoji[4] [1] [7] [11] [9] [8].

⁂

1. https://www.geeksforgeeks.org/javascript/javascript-strings/
2. https://www.freecodecamp.org/news/javascript-string-tutorial-string-methods-in-js/
3. https://www.linkedin.com/pulse/explain-strings-javascript-like-i-am-6-luc-constantin
4. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String
5. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Numbers_and_strings
6. https://javascript.info/string
7. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals
8. https://www.freecodecamp.org/news/template-literals-in-javascript/
9. https://www.geeksforgeeks.org/javascript/javascript-template-literals/
10. https://www.w3schools.com/js/js_string_methods.asp
11. https://talent500.com/blog/20-javascript-essential-string-methods-to-ace-your-next-interview/
12. https://www.cybrosys.com/blog/15-javascript-string-methods-with-examples
13. https://dev.to/devsmitra/javascript-string-methods-a-cheat-sheet-for-developer-4kbk
14. https://www.geeksforgeeks.org/javascript/javascript-string-methods/
15. https://www.programiz.com/javascript/library/string