[JavaScript String Methods] (CheatSheet)

1. Basic String Properties

- **length**: `str.length`
- **charAt**: `str.charAt(index)`
- **charCodeAt**: `str.charCodeAt(index)`
- **fromCharCode (static)**: `String.fromCharCode(asciiCode)`

2. String Searching

- **indexOf**: `str.indexOf(substring)`
- **lastIndexOf**: `str.lastIndexOf(substring)`
- **includes**: `str.includes(substring)`
- **startsWith**: `str.startsWith(substring)`
- **endsWith**: `str.endsWith(substring)`
- **search**: `str.search(regex)`
- **match**: `str.match(regex)`
- **localeCompare**: `str1.localeCompare(str2)`

3. String Manipulation

- **concat**: `str1.concat(str2)`
- **slice**: `str.slice(start, end)`
- **substring**: `str.substring(start, end)`
- **substr**: `str.substr(start, length)`
- **replace**: `str.replace(searchValue, newValue)`
- **replaceAll**: `str.replaceAll(searchValue, newValue)`
- **trim**: `str.trim()`
- **trimStart** or **trimLeft**: `str.trimStart()` or `str.trimLeft()`
- **trimEnd** or **trimRight**: `str.trimEnd()` or `str.trimRight()`
- **padStart**: `str.padStart(targetLength, padString)`
- **padEnd**: `str.padEnd(targetLength, padString)`

4. Case Conversion

- **toLowerCase**: `str.toLowerCase()`

- **toUpperCase**: `str.toUpperCase()`

5. Regular Expressions

- **split (with regex)**: `str.split(/regex/)`
- **replace (with regex)**: `str.replace(/regex/, newSubstr)`
- **search (with regex)**: `str.search(/regex/)`
- **match (with regex)**: `str.match(/regex/)`

6. Working with Templates

- **Template Literals**: `` `Hello, ${name}!` ``
- **Multi-Line Strings with Template Literals**: `` `Line 1\nLine 2` ``

7. Escape Sequences

- **New Line**: `'Line 1\\nLine 2'`
- **Tab**: `'Column 1\\tColumn 2'`
- **Single Quote**: `'It\\'s a quote'`
- **Double Quote**: `"He said, \\"Hello\\""`
- **Backslash**: `'Use \\\\ to represent a backslash'`

8. Unicode and Internationalization

- **fromCodePoint (static)**: `String.fromCodePoint(codePoint)`
- **codePointAt**: `str.codePointAt(pos)`
- **normalize**: `str.normalize(form)`

9. String Iteration

- **Iterate over Characters**: `[...str]`
- **Spread Operator to Array**: `[...str]`

10. Advanced String Manipulation

- **repeat**: `str.repeat(count)`
- **String Interpolation**: `` `String with ${expression}` ``
- **String Raw (Static Method)**: `` String.raw `string` ``

By: Waleed Mousa

11. String Checking

- **isString (Type Checking)**: `typeof str === 'string'`

12. String Encoding and Decoding

- **encodeURI**: `encodeURI(str)`
- **encodeURIComponent**: `encodeURIComponent(str)`
- **decodeURI**: `decodeURI(str)`
- **decodeURIComponent**: `decodeURIComponent(str)`

13. Advanced Searching

- **Regular Expression Test**: `/regex/.test(str)`
- **Regular Expression Exec**: `/regex/.exec(str)`

14. String to Primitive Conversion

- **String to Integer (parseInt)**: `parseInt(str, radix)`
- **String to Float (parseFloat)**: `parseFloat(str)`
- **String to Number**: `Number(str)`

15. Advanced Template Literal Features

- **Tagged Templates**: ``tagFunction `Template ${expression}` ``
- **Raw String Access in Tagged Templates**: `function tag(strings) { strings.raw[0]; }`

16. String as an Iterable

- **Using for...of Loop**: `for (const char of str) { }`
- **Array Destructuring**: `[firstChar, secondChar] = str`

17. String Parsing

- **JSON.parse with Strings**: `JSON.parse(jsonString)`

18. Advanced Regular Expressions

- **Flags in Regex**: `/pattern/gim`
- **Named Groups in Regex**: `/(?<name>pattern)/`

19. String Generation

- **Generate Random String (Example using Math.random)**: `Math.random().toString(36).substring(2)`

20. String and Data Structures

- **String to Array**: `'string'.split('')`
- **Array to String**: `['s', 't', 'r'].join('')`

21. String and URL Manipulation

- **BTOA (Base64 Encoding)**: `btoa('string')`
- **ATOB (Base64 Decoding)**: `atob('encodedString')`

22. String as Object

- **String Constructor**: `new String('string')`
- **Checking String Object Type**: `(new String('str')) instanceof String`

23. String Escape Characters (Advanced)

- **Unicode Escape Sequence**: `'\\u00A9'`
- **Hexadecimal Escape Sequence**: `'\\xA9'`

24. String Methods and Performance

- **Performance Considerations**: `Comparing 'str' + 'ing' vs` \str${'ing'}``

25. String Data Handling

- **Handling Multiline Strings**: Using backticks or concatenation for multiline strings
- **String and Byte Conversion (Example)**: `new TextEncoder().encode(str); new TextDecoder().decode(bytes)`

26. String and HTML

- **Escape HTML**: `str.replace(/&/g, '&amp;').replace(/</g, '&lt;').replace(/>/g, '&gt;')`

27. String and Arrays

- **Converting Comma-Separated String to Array**: `'1,2,3'.split(',')`
- **Converting Array to Comma-Separated String**: `[1, 2, 3].join(',')`

28. Advanced Case Conversion

- **Locale-Specific Uppercase**: `str.toLocaleUpperCase('tr-TR')`
- **Locale-Specific Lowercase**: `str.toLocaleLowerCase('tr-TR')`

29. String Mutation (Note: Strings in JavaScript are Immutable)

- **Creating a New Modified String**: `const newStr = oldStr.replace('old', 'new')`

30. Debugging and Inspection

- **Console Logging a String**: `console.log('Debug: ' + str)`
- **String Length for Debugging**: `console.log('Length: ' + str.length)`

31. String and Regular Expressions (Advanced)

- **Replacing with Function**: `str.replace(/regex/, (match) => match.toUpperCase())`
- **Split with Limit**: `'a,b,c,d'.split(',', 2)`

32. String and Character Access

- **Accessing Characters using Bracket Notation**: `'string'[0]`
- **Using charAt for Character Access**: `'string'.charAt(0)`

33. String and Memory

- **String Interning**: Understanding how identical strings are stored

34. String and Comparisons

- **Locale Compare for Sorting**: `['ä', 'a', 'z'].sort((a, b) => a.localeCompare(b))`

35. String Literal vs String Object

- **Literal vs Object**: `'literal' vs new String('object')`

36. String and Browser API

- **Clipboard Access with Strings**: `navigator.clipboard.writeText('string')`
- **Reading Clipboard as String**: `navigator.clipboard.readText()`

37. String Best Practices

- **Immutable String Practices**: Working effectively with immutable string nature
- **Minimizing String Concatenation**: Using arrays or template literals for large concatenations
- **Using Template Literals for Readability**: `` `Hello, ${name}!` ``

38. String Libraries and Frameworks

- **Lodash for String Manipulation**: `_.camelCase('string')`
- **Using String.js**: `S('string').left(4).s`