JavaScript Strings

A JavaScript String is a **sequence of characters**. Let a = “abc”

Types:

String Literals

String Constructor

Template Literals (String Interpolation)

Template literals allow you to embed expressions within backticks (`) for dynamic string creation, making it more readable and versatile.(Interpolation)

Template String **provide an easy way to interpolate variables and expressions into strings. ${...}**

```
let s1 = 'abcd'; // recommended

console.log(s1);

let s = new String('abcd');

console.log(s);

let s2 = `You are learning from ${s1}`;


console.log(s2);
```

1. Finding the length of a String

```
let s = 'JavaScript';

let len = s.length;
```

2. String Concatenation

```
let s1 = 'Java';

let s2 = 'Script';

let res = s1 + s2;

 let txt = s1.concat(s2)

console.log(txt)
```

3. Escape Characters

```
\' - Inserts a single quote
\" - Inserts a double quote
\\ - Inserts a backslash
```

```
const s1 = "\'SDLC\' is a learning portal";
const s2 = "\"SDLC\" is a learning portal";

const s3 = "\\SDLC\\ is a learning portal";

console.log(s1);

console.log(s2);

console.log(s3);
```

4. Find Substring of a String

```
let s1 = 'JavaScript Tutorial';

let s2 = s1.substring(0, 10);
```

5. Convert String to Uppercase and Lowercase

```
let uCase = s.toUpperCase();

let lCase = s.toLowerCase();
```

===
=============

The charAt() method returns the character at a specified index (position) in a string: text.charAt(0); name.at(2); text[0];

The charCodeAt() method returns the ASCII code of the character at a specified index in a string: text.charCodeAt(0)

===
===============================

There are 3 methods for extracting a part of a string:

- slice(*start*, *end*)
- substring(*start*, *end*)
- substr(*start*, *length*)

- slice() extracts a part of a string and returns the extracted part in a new string.

- The method takes 2 parameters: start position, and end position (end not included).

let text = "Apple, Banana, Kiwi";
let part = text.slice(7, 13);

let part = text.slice(7);

let part = text.slice(-12);

let part = text.slice(-12, -6);

substring() is similar to slice().

The difference is that start and end values less than 0 are treated as 0 in substring().

let part = str.substring(7, 13);

==

concat() joins two or more strings:

The trim() method removes whitespace from both sides of a string: text1.trim(); trimStart(), trimEnd()

The padStart() method pads a string from the start. let text = "5";
let padded = text.padStart(4,"0"); Ex : 0004

The padEnd() method pads a string from the end.

The repeat() method returns a string with a number of copies of a string.

The repeat() method returns a new string. text.repeat(2);

===

The replace() method replaces a specified value with another value in a string:

 let text = "Please visit Microsoft!";
let newText = text.replace("Microsoft", "AROSPACE");

let newText = text.replace(/MICROSOFT/i, "DEX"); case insensitive

let text = "Please visit Microsoft and Microsoft!";
let newText = text.replace(/Microsoft/g, "HEVEN"); global match

text = text.replaceAll("cats","dogs");

A string can be converted to an array with the split() method: text.split(" ") text.split("|") text.split(",")

String Search Methods

indexOf() => Returns the index of the first occurrence of a given substring, or -1 if not found. Indexing is zero-based.

```
const text = "Hello world, world!";

console.log(text.indexOf("world"));
```

lastIndexOf() => Returns the index of the **last** occurrence of a substring, searching backwards. Returns -1 if not found.

```
const text = "Hello world, world!";

console.log(text.lastIndexOf("world"));
```

search() => Finds and returns the index of the first match of a substring or regular expression. Doesn't support a starting position parameter. Returns -1 if none.

```
const text = "Find the word locate here";

console.log(text.search("locate")); // Outputs: position of "locate"

console.log(text.search(/locate/)); // Same result with regex
```

match() => Executes a search using a string or regex. Returns an array with matched results (or null). Without the g flag, it returns only the first match.

```
const text = "The rain in SPAIN stays mainly in the plain";

console.log(text.match(/ain/g)); // Outputs: [ "ain", "ain", "ain" ]
```

matchAll() => Returns an iterable of all regex matches with detailed info (requires the g flag). Throws an error if g is missing. ES2020+ feature.

```
const text = "Cats and cats and Cats";

const iterator = text.matchAll(/Cats/gi);

for (const m of iterator) {

  console.log(m);

}
```

includes() => Returns true if the substring is found within the string. Case-sensitive. Supports optional start index.

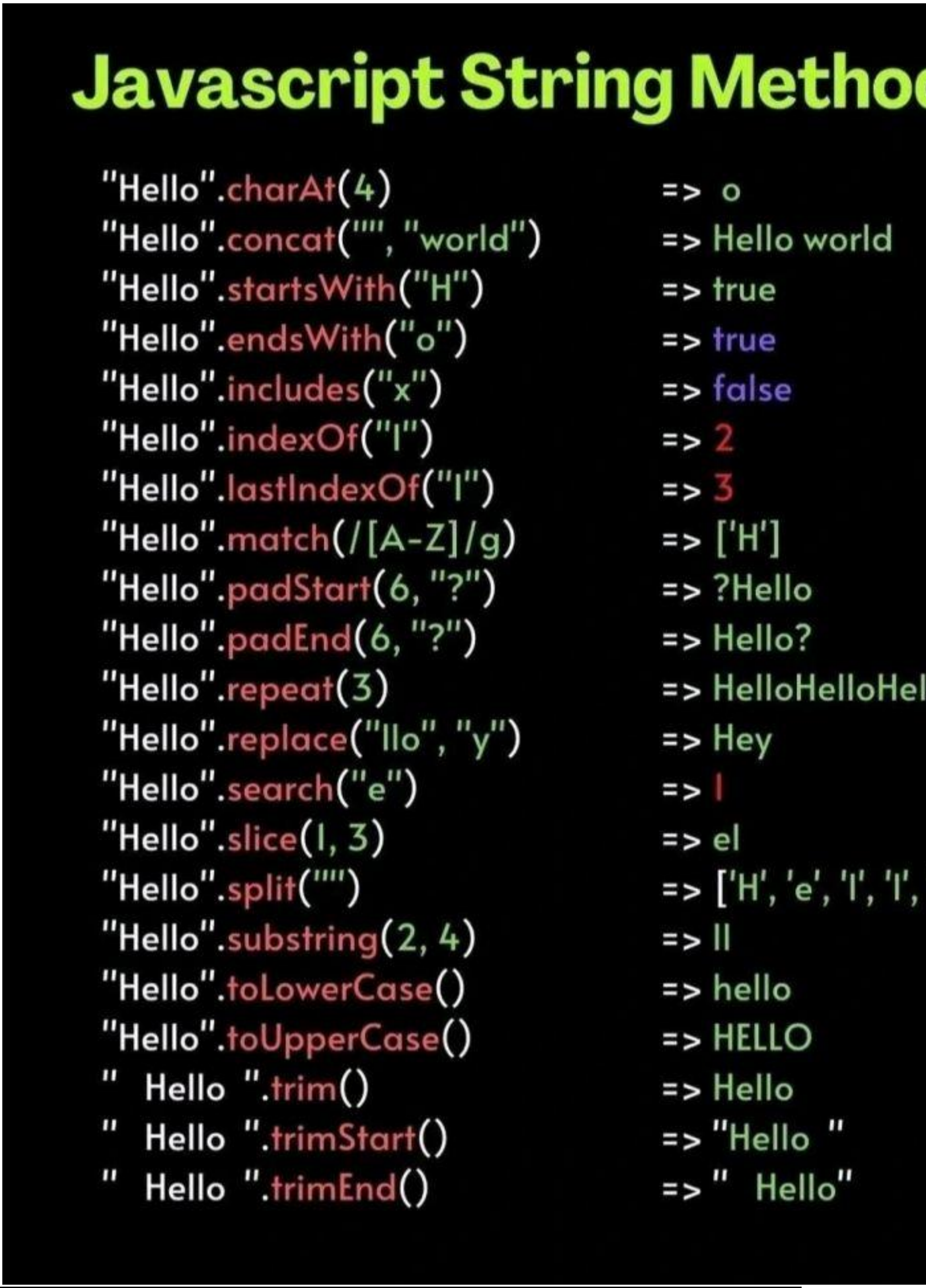
```
const text = "Hello world";

console.log(text.includes("world")); // true

console.log(text.includes("World")); // false
```

startsWith() => Checks if the string begins with a given substring. Returns Boolean. Case-sensitive. Supports optional start index

```
const text = "Saturday night";

console.log(text.startsWith("Sat")); // true

console.log(text.startsWith("night", 9)); // true
```

endsWith() => Checks if the string ends with a specified substring. Returns Boolean. Case-sensitive. Supports checking up to a given ending index.

```
const text = "To be, or not to be, that is the question.";

console.log(text.endsWith("question.")); // true

console.log(text.endsWith("to be", 19));   // true
```



Javascript String Metho

```
"Hello".charAt(4)                  => o
"Hello".concat("", "world")        => Hello world
"Hello".startsWith("H")            => true
"Hello".endsWith("o")              => true
"Hello".includes("x")              => false
"Hello".indexOf("l")               => 2
"Hello".lastIndexOf("l")           => 3
"Hello".match(/[A-Z]/g)            => ['H']
"Hello".padStart(6, "?")           => ?Hello
"Hello".padEnd(6, "?")             => Hello?
"Hello".repeat(3)                  => HelloHelloHel
"Hello".replace("llo", "y")        => Hey
"Hello".search("e")                => 1
"Hello".slice(1, 3)                => el
"Hello".split("")                  => ['H', 'e', 'l', 'l',
"Hello".substring(2, 4)            => ll
"Hello".toLowerCase()              => hello
"Hello".toUpperCase()              => HELLO
"  Hello  ".trim()                 => Hello
"  Hello  ".trimStart()            => "Hello  "
"  Hello  ".trimEnd()              => "  Hello"
```