Strings: Properties, Methods & Template Literals in JavaScript | JavaScript Tutorial In Hindi #6

In today's tutorial, we will study about ***string properties and methods in JavaScript.*** As we know, strings are useful for holding data that can be represented in text form. One of the most popular operations on strings are to check their length, to build and concatenate them using the operator (+), checking for the existence or location of substrings with the indexOf() method, or extracting substrings with the substring() method. In this tutorial, we will also learn about ***JavaScript template literals*** that makes it easier to work with the string template.

Note that: JavaScript counts positions from zero. 0 is the first position in a string, 1 is the second, and so on.

JavaScript String Methods:-

Let's see the JavaScript string methods with examples.

- **charAt(c):** It returns the character at the “c” position within the string.

```
var myString = 'JavaScript!!!';
console.log(myString.charAt(7));
//output: i
```

- **concat(p1, p2):** Combines one or more strings and returns the concatenated string. Remember that the original string is not modified.

```
var str1="JavaScript"
var str2=str1.concat(" is","awesome")
console.log(str2);
//Output: JavaScript is awesome
```

- **indexOf(substr, [start_from]):** This method Searches and returns the index number of the searched character within the string. If not found, it will return -1 . “Start_from” is an optional argument specifying the position within string to begin the search. Default is 0.

```
//indexOf(char/substring)
var str1="Hi, my name is Sam!";
var str2 = str1.indexOf("locate");
//Output: -1
```

- **lastIndexOf(substr, [start_from]):** This method searches and returns the index number of the searched character within the string. This method will return the index of the last occurrence of a specified text in a string.If the character or string is not found, -1 is returned. "Start_from" is an optional argument specifying the position within string to begin the search. Default is string.length-1

```
var myString = 'javascript Node.js';
console.log(myString.lastIndexOf('N'));
//output: 11
```

- **slice(start, [end]):** This method returns a substring of the string based on the "start" and "end" index , it will not include the "end" index itself. "End" argument is optional, and if none is specified, the slice includes all characters from "start" to end of string.

```
var text="programming"
var mystr= text.slice(0,4)
console.log(mystr)
//Output:- "prog"
```

- **split(delimiter, limit):** This method Splits a string into substring to the specified delimiter, and returns an array containing each element. The argument “limit” is an integer that lets you specify the maximum number of elements to return.

```
var txt = "a,b,c,d,e";   // String
txt.split(",");          // Split on commas
txt.split(" ");          // Split on spaces
```

- **substring(from, [to]):** This method will returns the characters in a string between “from” and “to” indexes. It will not include “to” inself. “To” argument is optional, and if omitted, up to the end of the string is assumed.

```
//substring(from, [to])
var myString = 'javascript Programming';
myString = myString.substring(0,10);
```

```
console.log(myString)
//output: javascript
```

- **toLowerCase():** This method will Returns the string with all of its characters converted to lowercase.

```
//toLowerCase()
var myString = 'JAVASCRIPT';
myString = myString.toLowerCase();
console.log(myString)
//output: javascript
```

- **toUpperCase():** This method will Returns the string with all of its characters converted to uppercase.

```
//toUpperCase()
var myString = 'javascript';
myString = myString.toUpperCase();
console.log(myString)
//output: JAVASCRIPT
```

- **search("str"):** The method searches a string for a specified value and returns the position of the match:

```
var str = "Welcome to programming World!!";
var pos = str.search("programming");
```

- **substr() Method:** The method substr() is similar to slice(). The only difference is that the second parameter specifies the length of the extracted part.

```
var str = "Apple, Banana, Kiwi";
var res = str.substr(7, 6);
// Output: Banana
```

JavaScript template literals:-

Prior to ES6, we use single quotes (') or double quotes (") to wrap a string literal. At that time, the strings have very limited functionality. To help us in solving more complex problems, ES6 template literals provide the syntax that allows you to work with strings in a much cleaner way. In ES6, we can create a template literal by wrapping the string in backticks as follows:

```
let simple = `This is an example of
                    Template literal`;
```

Following are the features of javascript template literals:

1. We can write a Multiline string
2. It provides the ability to substitute part of the string for the values of variables or expressions. This feature is also called string interpolation.
3. It provides the ability to transform a string so that it is safe to include in HTML.

```
let firstName = 'Code With',
lastName = 'Harry';
let greeting = `Welcome to ${firstName} ${lastName}`;
console.log(greeting);
//Welcome to Code With Harry
```

Code index.html as described/written in the video

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Tutorial on Js</title>
</head>
<body>
    <h1>This is Js tutorial by Harry</h1>
</body>

<!-- <script src="js/tut2.js"></script> -->
<!-- <script src="js/tut3.js"></script> -->
<!-- <script src="js/tut4.js"></script> -->
<!-- <script src="js/tut5.js"></script> -->
<!-- <script src="js/tut6.js"></script> -->
<!-- <script src="js/tut7.js"></script> -->
<!-- <script src="js/tut8.js"></script> -->
<!-- <script src="js/tut9.js"></script> -->
<!-- <script src="js/tut10.js"></script> -->
```

```
<script src="js/tut11.js"></script>
</html>
```

Js code as described/written in the video

```
console.log('We are at tut 6');
const name = 'Harry';
const greeting = 'Good Morning';
// console.log(greeting + ' ' + name);

let html;
html  = "<h1> this is heading</h1>"+
        "<p> this is My para</p>";

html = html.concat('this', ' str2');
console.log(html);
// console.log(html.length);
// console.log(html.toLowerCase());
// console.log(html.toUpperCase());
// console.log(html);


// console.log(html[1]);
// console.log(html.indexOf('<'));
// console.log(html.lastIndexOf('<'));
// console.log(html.charAt(3));
// console.log(html.endsWith('dsfsdfd'));
// console.log(html.includes(' fg'));
// console.log(html.substring(1,6));
// console.log(html.slice(0, 4))
// console.log(html.split('>'));
// console.log(html.replace('this', 'it'));

let fruit1 = 'Orange\'';
let fruit2 = 'Apple';
let myHtml = `Hello ${name}
            <h1> This is "my" heading </h1>
            <p> You like ${fruit1} and ${fruit2}`;
```

```
document.body.innerHTML = myHtml;
```

Previous

Next

CodeWithHarry