JavaScript - Strings

The **String** object in JavaScript lets you work with a series of characters; it wraps JavaScript's string primitive data type with a number of helper methods.

As JavaScript automatically converts between string primitives and String objects, you can call any of the helper methods of the String object on a string primitive.

The string is a sequence of characters containing 0 or more characters. For example, 'Hello' is a string.

Syntax

JavaScript strings can be created as objects using the String() constructor or as primitives using string literals.

Use the following syntax to create a String object –

```
var val = new String(value);
```

The String parameter, value is a series of characters that has been properly encoded.

We can create string primitives using string literals and the String() function as follows –

```
str1 = 'Hello World!'; // using single quote
str2 = "Hello World!"; // using double quote
str3 = 'Hello World'; // using back ticks
str4 = String('Hello World!'); // using String() function
```

JavaScript String Object Properties

Here is a list of the properties of String object and their description.

Sr.No.	Property	Description
1	**constructor**	Returns a reference to the String function that created the object.
2	**length**	Returns the length of the string.
3	**prototype**	The prototype property allows you to add properties and methods to an object.

JavaScript String Object Methods

Here is a list of the methods available in String object along with their description.

Static Methods

The static methods are invoked using the 'String' class itself.

Sr.No.	Property	Description
1	**fromCharCode()**	Converts the sequence of UTF-16 code units into the string.
2	**fromCodePoint()**	Creates a string from the given sequence of ASCII values.

Instance Methods

The instance methods are invoked using the instance of the String class.

Sr.No.	Method	Description
1	**at()**	Returns the character from the specified index.
2	**charAt()**	Returns the character at the specified index.
3	**charCodeAt()**	Returns a number indicating the Unicode value of the character at the given index.
4	**codePointAt()**	Returns a number indicating the Unicode value of the character at the given index.

5	**concat()**	Combines the text of two strings and returns a new string.
6	**endsWith()**	Checks whether the string ends with a specific character or substring.
7	**includes()**	To check whether one string exists in another string.
8	**indexOf()**	Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
9	**lastIndexOf()**	Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.
10	**localeCompare()**	Returns a number indicating whether a reference string comes before or after or is the same as the given string in sort order.
11	**match()**	Used to match a regular expression against a string.
12	**matchAll()**	Used to match all occurrences of regular expression patterns in the string.
13	**normalize()**	To get the Unicode normalization of the string.
14	**padEnd()**	To add padding to the current string with different strings at the end.
15	**padStart()**	To add padding to the current string with different strings at the start.
16	**raw()**	Returns a raw string form of a given template literal.
17	**repeat()**	To get a new string containing the N number of copies of the current string.
18	**replace()**	Used to find a match between a regular expression and a string and replace the matched substring with a new one.
19	**replaceAll()**	Used to find a match between a regular expression and a string and replace all the matched substring with a new one.

20	**search()**	Executes the search for a match between a regular expression and a specified string.
21	**slice()**	Extracts a section of a string and returns a new string.
22	**split()**	Splits a String object into an array of strings by separating the string into substrings.
23	**substr()**	Returns the characters in a string beginning at the specified location through the specified number of characters.
24	**substring()**	Returns the characters in a string between two indexes into the string.
25	**toLocaleLowerCase()**	The characters within a string are converted to lowercase while respecting the current locale.
26	**toLocaleUpperCase()**	The characters within a string are converted to the upper case while respecting the current locale.
27	**toLowerCase()**	Returns the calling string value converted to lowercase.
28	**toString()**	Returns a string representing the specified object.
29	**toUpperCase()**	Returns the calling string value converted to uppercase.
30	toWellFormed()	Returns a new string that is a copy of this string.
31	**trim()**	It removes white spaces from both ends.
32	**trimEnd()**	It removes white spaces from the start.
33	**trimStart()**	It removes white spaces from the end.
34	**valueOf()**	Returns the primitive value of the specified object.

String constructor

Sr.No.	Constructor	Description
1	String()	Creates a string object and initializes it with the provided value.

Examples

Let's take understand the JavaScript String object and string primitives with the help of some examples.

Example: Creating JavaScript String Objects

In the example below, we used the string() constructor with the 'new' keyword to create a string object.

Open Compiler

```
<html>
<head>
   <title> JavaScript - String Object </title>
</head>
<body>
   <p id = "output"> </p>
   <script>
      const output = document.getElementById("output");
      const str = new String("Hello World!");
      output.innerHTML += "str == " + str + "<br>";
      output.innerHTML += "typeof str == " + typeof str;
   </script>
</body>
</html>
```

Output

```
str == Hello World!
typeof str == object
```

Accessing a string

You can access the string characters using its index. The string index starts from 0.

Example

In the example below, we access the character from the 0th and 4th index of the string.

Open Compiler

```
<html>
<body>
   <p id = "output"> </p>
   <script>
      const output = document.getElementById("output");
      let str1 = new String("Welcome!");
      let str2 = "Welcome!";
      output.innerHTML += "0th character is - " + str1[0] + "<br>";
      output.innerHTML += "4th character is - " + str2[4] + "<br>";
   </script>
</body>
</html>
```

Output

```
0th character is - W
4th character is - o
```

JavaScript string is case-sensitive

In JavaScript, strings are case-sensitive. It means lowercase and uppercase characters are different.

Example

In the example below, char1 contains the uppercase 'S', and char2 contains the lowercase 's' characters. When you compare char1 and char2, it returns false as strings are case-sensitive.

Open Compiler

```
<html>
<head>
   <title> JavaScript - String case-sensitivity </title>
</head>
<body>
```

```
    <p id = "output">
    <script>
       let char1 = 'S';
       let char2 = 's'
       let ans = char1 == char2;
       document.getElementById("output").innerHTML += "s == S
    </script>
</body>
</html>
```

Output

s == S false

JavaScript Strings Are Immutable

In JavaScript, you can't change the characters of the string. However, you can update the whole string.

Example

In the example below, we try to update the first character of the string, but it doesn't get updated, which you can see in the output.

After that, we update the whole string, and you can observe the changes in the string.

Open Compiler

```
<html>
<head>
   <title> JavaScript – Immutable String </title>
</head>
<body>
   <p id = "output"> </p>
   <script>
      const output = document.getElementById("output");
      let str = "Animal";
      str[0] = 'j';
      output.innerHTML += "The string is: " + str + "<br>";
      str = "Hi!";
      output.innerHTML += "The updated string is: " + str;
```

```
    </script>
</body>
</html>
```

Output

```
The string is: Animal
The updated string is: Hi!
```

Escape Characters

You can use the special characters with the string using the backslash (\) characters. Here is the list of special characters.

Escape character	**Description**
\"	Double quote
\'	Single quote
\\	Backslash
\n	New line
\t	Tab
\b	backspace
\f	Form feed
\v	Verticle tab
\r	Carriage return
\uXXXX	Unicode escape

Example

In the example below, we added a single quote between the characters of the str1 string and a backslash between the characters of the str2 string.

Open Compiler

```
<html>
```

```
<head>
   <title> JavaScript - Escape Characters </title>
</head>
<body>
   <p id = "output"> </p>
   <script>
      const output = document.getElementById("output");
      let str1 = "Your\'s welcome!";
      let str2 = "Backslash \\";
      output.innerHTML += "str1 == " + str1 + "<br>";
      output.innerHTML += "str2 == " + str2 + "<br>";
   </script>
</body>
</html>
```

Output

```
str1 == Your's welcome!
str2 == Backslash \
```

String HTML Wrappers

Here is a list of the methods that return a copy of the string wrapped inside an appropriate HTML tag.

Sr.No.	Method & Description
1	anchor() Creates an HTML anchor that is used as a hypertext target.
2	big() Creates a string to be displayed in a big font as if it were in a <big> tag.
3	blink() Creates a string to blink as if it were in a <blink> tag.
4	bold() Creates a string to be displayed as bold as if it were in a <b> tag.
5	fixed() Causes a string to be displayed in fixed-pitch font as if it were in a <tt> tag.
6	fontcolor() Causes a string to be displayed in the specified color as if it were in a <font

	color="color"> tag.
7	fontsize() Causes a string to be displayed in the specified font size as if it were in a <font size="size"> tag.
8	italics() Causes a string to be italic, as if it were in an <i> tag.
9	link() Creates an HTML hypertext link that requests another URL.
10	small() Causes a string to be displayed in a small font, as if it were in a <small> tag.
11	strike() Causes a string to be displayed as struck-out text, as if it were in a <strike> tag.
12	sub() Causes a string to be displayed as a subscript, as if it were in a <sub> tag.
13	sup() Causes a string to be displayed as a superscript, as if it were in a <sup> tag.

In the following sections, we will have a few examples to demonstrate the usage of String methods.