**Strings in JavaScript | Web Development Tutorials #49**

**Strings In JavaScript**

In this tutorial, we are going to learn about Strings in JavaScript and how to make them, what are the methods to manipulate those Strings, and how to display them in the browser using JavaScript. A JavaScript string is zero or more characters written inside the quotes. The **String** object is used to represent and manipulate a sequence of characters.

Start by making a new file as *tut49.html* and add the boilerplate to get the basic HTML code. Name the file as **JavaScript | String and String Methods** under the <title> tag.

To start, we will add the basic HTML code as shown below-

<div class="container">

<h1>Lorem ipsum dolor sit.</h1>

<div id="content"></div>

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Quam nisi, quaerat corrupti quas, illum nobis tempore sequi cumque laboriosam magni expedita earum? Similique corrupti nam magni reprehenderit quia vero, reiciendis eius officiis doloremque ipsa?</p>

</div>

Copy

To write the JavaScript we will include a <script> tag. For example, if we write-

var string = "this";

Copy

We will see that the “this” word is reflected back on the console tab in the browser. However, apart from the double quotes, we can also write a Sting in a single quote as ‘this’. It is recommended to use double quotes when we are using single quotes between the strings. For example-

var string = 'thi"s';

Copy

If we declare new variables with strings and concatenate them with a ‘+’ sign as shown here-

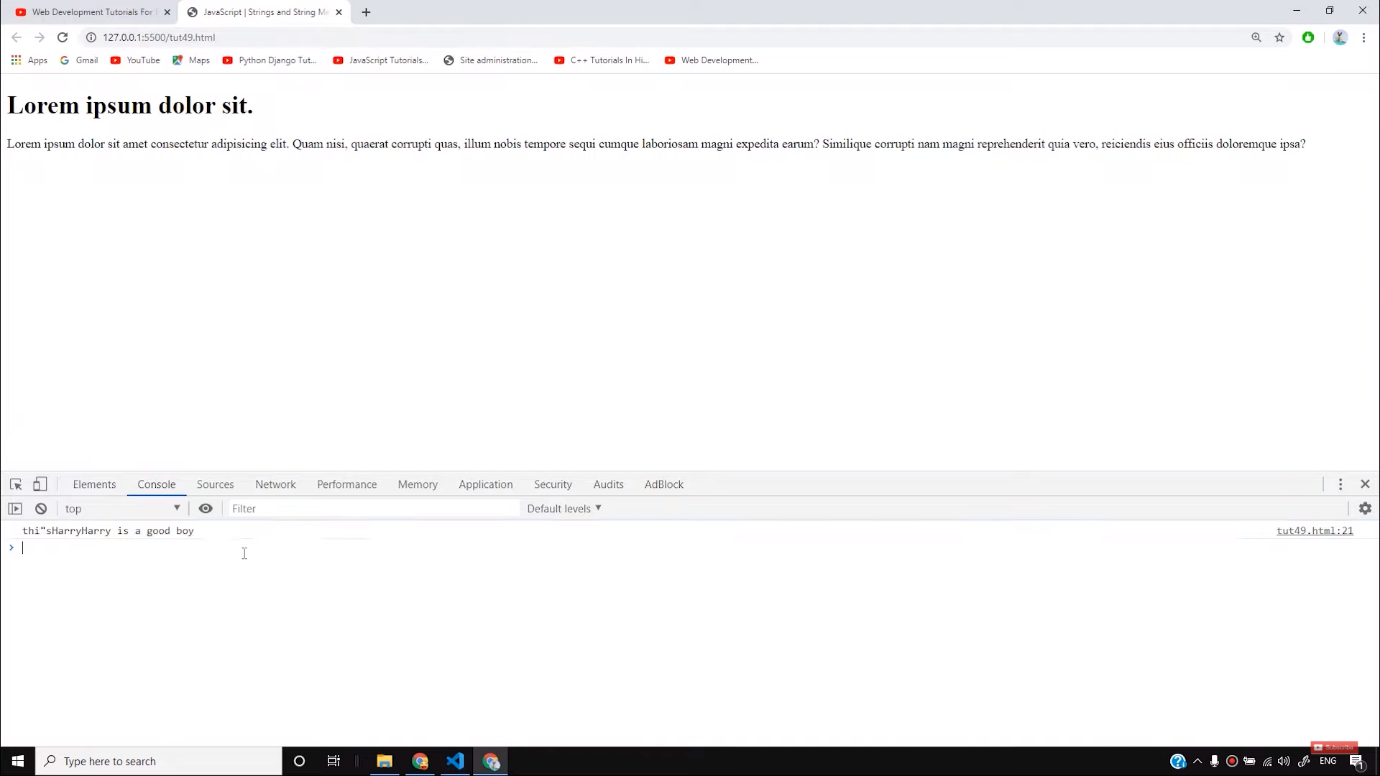
var string = 'thi"s';

var name = 'Harry';

var message = 'Harry is a good boy';

Copy

Then the result will be that all the three strings will be merged together as shown below-



Let us now understand what are **template literals**. These are written under **`** sign. If we write the code as below-

var name = 'Harry';

var channel = 'CodeWithHarry';

var message = 'Harry is a good boy';

var temp = `${name} is a 'nice' person "and" he has a channel called ${channel}`;

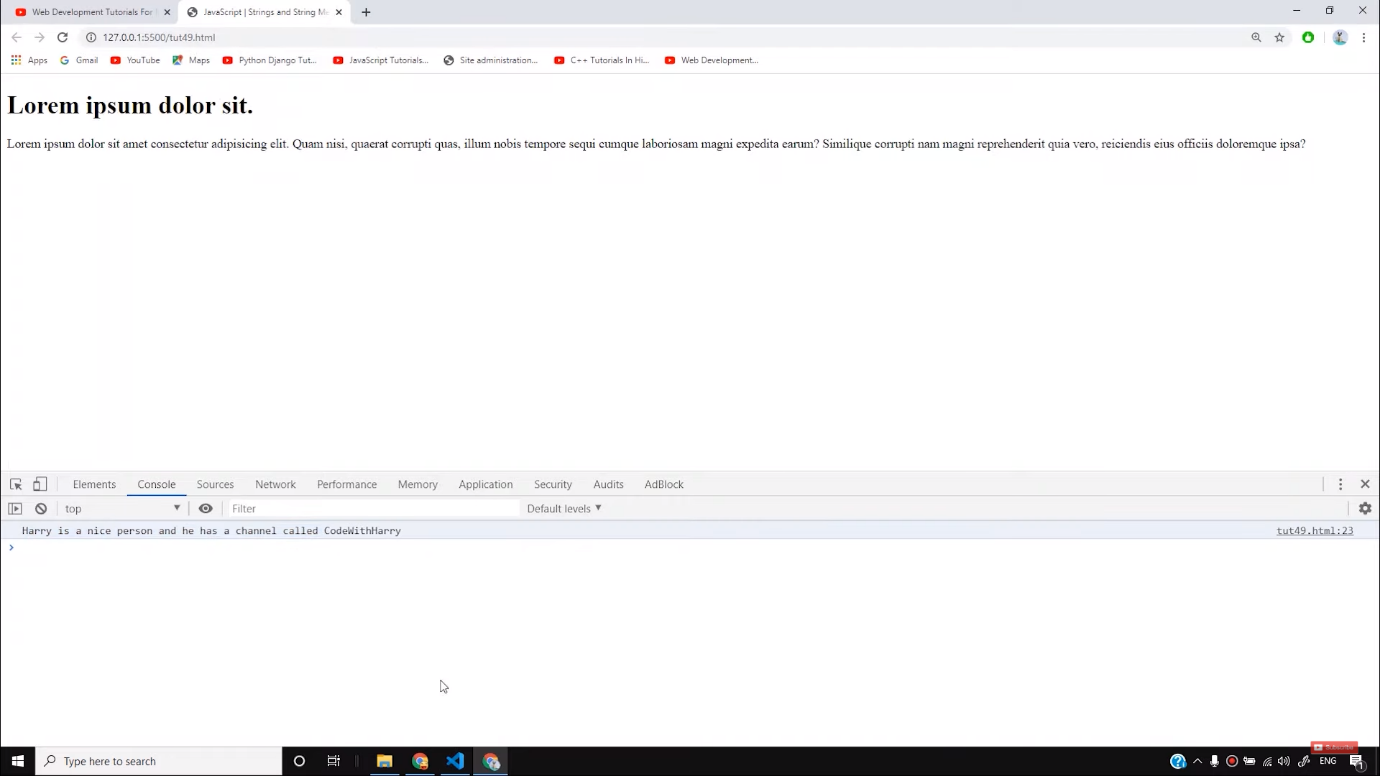
Copy

**$**symbol is used here to pick that particular string from the variable. And if we do,

console.log(temp);

Copy

Then the result will be as follows-



The main benefit of using this method is, we can use both double quotes or single quotes to identify the strings.

To extract the number of characters in a particular string we can take the help of length function as shown below-

var len = name.length;

console.log(`Length of name is ${len}`)

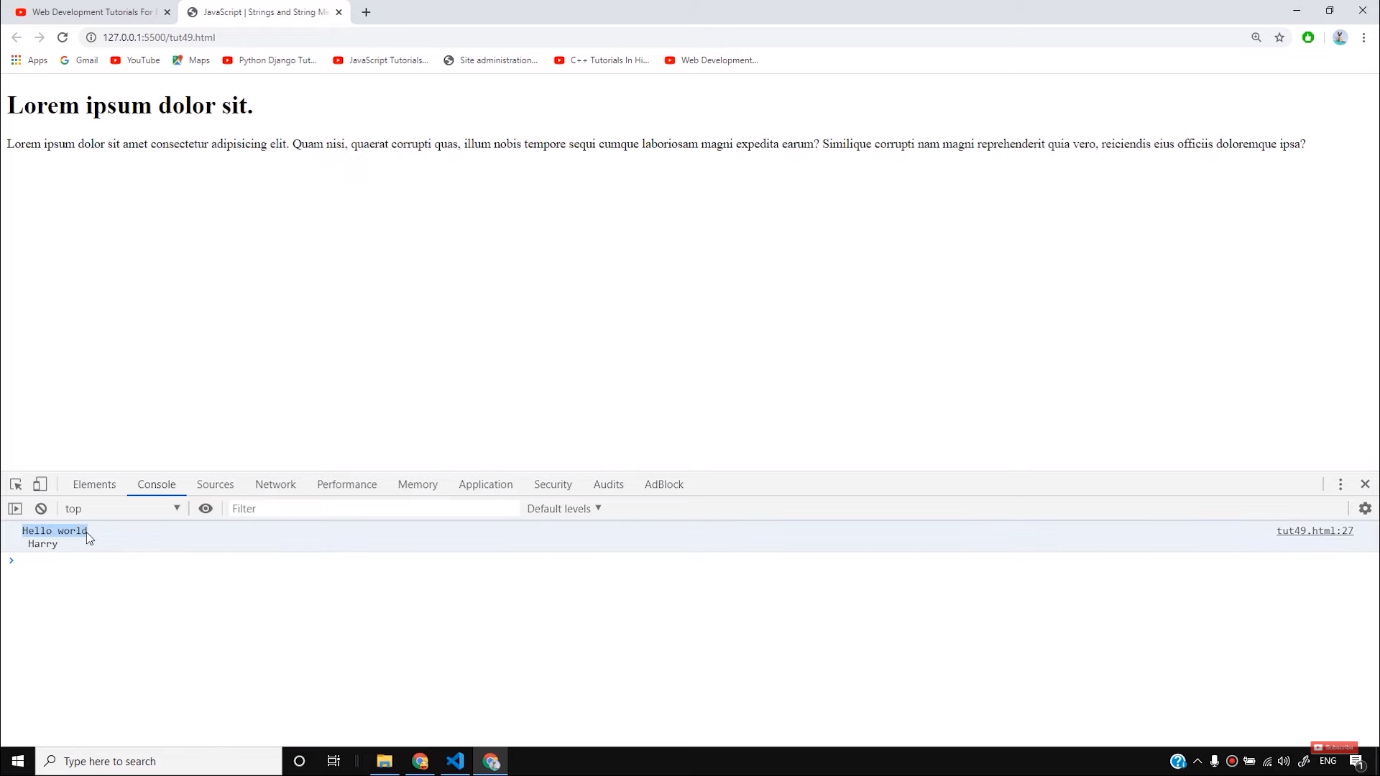
Copy

It will return the number of characters in the variable **name.**We also have **escape sequence**characters in the Strings. For example, if we write-

console.log("Hello world\nHarry");

Copy

The result will be as follows-



“\n” is an escape sequence that is used to take the string after it in the next line. It’s a **new line**character.

So I hope you must have got a basic understanding of Strings of JavaScript. In the next tutorials, we will learn the different methods of creating strings. Therefore, stay with tutorials to learn more.

**Code 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>JavaScript | Strings and String Methods</title>

</head>

<body>

<div class="container">

<h1>Lorem ipsum dolor sit.</h1>

<div id="content"></div>

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Quam nisi, quaerat corrupti quas, illum nobis tempore sequi cumque laboriosam magni expedita earum? Similique corrupti nam magni reprehenderit quia vero, reiciendis eius officiis doloremque ipsa?</p>

</div>

<script>

// var string = "this";

var string = 'thi"s';

var name = 'Harry';

var channel = 'CodeWithHarry';

var message = 'Harry is a good boy';

var temp = `${name} is a 'nice' person "and" he has a channel called ${channel}`;

// console.log(string + name + message);

// console.log(temp);

// var len = name.length;

// console.log(`Length of name is ${len}`)

// console.log("Hello \\nworld\nHarry\tand");

var y = new String("this");

console.log(y);

document.getElementById('content').innerHTML = '<h3>this is an h3 heading</h3>'

</script>

</body>

</html>

# JavaScript Strings

[❮ Previous](https://www.w3schools.com/js/js_events.asp)[Next ❯](https://www.w3schools.com/js/js_string_methods.asp)

JavaScript strings are for storing and manipulating text.

A JavaScript string is zero or more characters written inside quotes.

### **Example**

let text = "John Doe";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string)

You can use single or double quotes:

### **Example**

let carName1 = "Volvo XC60";  // Double quotes  
let carName2 = 'Volvo XC60';  // Single quotes

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_quotes)

You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

### **Example**

let answer1 = "It's alright";  
let answer2 = "He is called 'Johnny'";  
let answer3 = 'He is called "Johnny"';

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_quotes_mixed)

## **String Length**

To find the length of a string, use the built-in length property:

### **Example**

let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
let length = text.length;

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_length)

ADVERTISEMENT

## **Escape Character**

Because strings must be written within quotes, JavaScript will misunderstand this string:

let text = "We are the so-called "Vikings" from the north.";

The string will be chopped to "We are the so-called ".

The solution to avoid this problem, is to use the **backslash escape character**.

The backslash (\) escape character turns special characters into string characters:

|  |  |  |
| --- | --- | --- |
| **Code** | **Result** | **Description** |
| \' | ' | Single quote |
| \" | " | Double quote |
| \\ | \ | Backslash |

The sequence \"  inserts a double quote in a string:

### **Example**

let text = "We are the so-called \"Vikings\" from the north.";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_escape_quotes2)

The sequence \'  inserts a single quote in a string:

### **Example**

let text= 'It\'s alright.';

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_escape_quotes1)

The sequence \\  inserts a backslash in a string:

### **Example**

let text = "The character \\ is called backslash.";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_escape_backslash)

Six other escape sequences are valid in JavaScript:

|  |  |
| --- | --- |
| **Code** | **Result** |
| \b | Backspace |
| \f | Form Feed |
| \n | New Line |
| \r | Carriage Return |
| \t | Horizontal Tabulator |
| \v | Vertical Tabulator |

The 6 escape characters above were originally designed to control typewriters, teletypes, and fax machines. They do not make any sense in HTML.

## **Breaking Long Code Lines**

For best readability, programmers often like to avoid code lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it is after an operator:

### **Example**

document.getElementById("demo").innerHTML =  
"Hello Dolly!";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_statements_linebreak)

You can also break up a code line **within a text string** with a single backslash:

### **Example**

document.getElementById("demo").innerHTML = "Hello \  
Dolly!";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_break)

The \ method is not the preferred method. It might not have universal support.  
Some browsers do not allow spaces behind the \ character.

A safer way to break up a string, is to use string addition:

### **Example**

document.getElementById("demo").innerHTML = "Hello " +  
"Dolly!";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_break_ok)

You cannot break up a code line with a backslash:

### **Example**

document.getElementById("demo").innerHTML = \  
"Hello Dolly!";

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_strings_codebreak)

## **JavaScript Strings as Objects**

Normally, JavaScript strings are primitive values, created from literals:

let x = "John";

But strings can also be defined as objects with the keyword new:

let y = new String("John");

### **Example**

let x = "John";  
let y = new String("John");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_object)

Do not create Strings objects.

The new keyword complicates the code and slows down execution speed.

String objects can produce unexpected results:

When using the == operator, x and y are **equal**:

let x = "John";  
let y = new String("John");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_object1)

When using the === operator, x and y are **not equal**:

let x = "John";  
let y = new String("John");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_object2)

Note the difference between (x==y) and (x===y).

(x == y) true or false?

let x = new String("John");  
let y = new String("John");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_string_object3)

(x === y) true or false?

let x = new String("John");  
let y = new String("John");