AUGUST 10, 2022 / #JAVASCRIPT

JavaScript Multiline String – How to Create Multi Line Strings in JS

Dionysia Lemonaki

In this article, you will learn three different ways to create multiline strings in JavaScript.

I will first explain the basics of strings in JavaScript and go over how to use template literals. Then, you will learn how to create a string that spans multiple lines with the help of code examples along the way.

Here is what we will cover:

What Is A String in JavaScript? An Intro on How to Create A String in JS

Strings are an effective way of communicating through text.

A string is an ordered sequence of character values. Specifically, a string is a sequence of one or more characters that can be either letters, numbers, or symbols (such as punctuation marks).

There are three ways you can create a string in JavaScript:

- By using single quotes.
- By using double quotes.
- By using backticks.

Here is how to create a string using **single quotes**:

```
// string created using single quotes ('')
let favePhrase = 'Hello World!';
```

Here is how to create a string using **double quotes**:

```
// string created using double quotes ("")
let favePhrase = "Hello World!";
```

Here is how to create a string using **backticks**:

```
// string created using backticks (``)
let favePhrase = `Hello World!`;
```

The last way of creating strings in JavaScript is known as a **template literal**.

I created a variable named `favePhrase`.

Inside the variable, I stored the string `Hello World!`, which I created using three different ways.

To view the output of the string in the browser's console, pass the variable name to `console.log();`.

For example, If I wanted to see the output of the string created with double quotes, I would do the following:

```
// string created using double quotes ("")
let favePhrase = "Hello World!";

// print string to the console
console.log(favePhrase);

// output

// Hello World!
```

Creating strings using single or double quotes works the same, so there is no difference between the two.

You can choose to use either or both of them throughout a file. That said, it is a good idea to remain consistent across your file.

When creating a string, make sure that the type of quotes you use is the same on both sides.

```
// Don't do this
let favePhrase = 'Hello World!";

console.log(favePhrase);

// output

// Uncaught SyntaxError: Invalid or unexpected token (at test.js:2:18)
```

Another thing to note is that you can use one type of quote inside another.

For example, you could use double quotes inside single quotes, like so:

```
let favePhrase = 'My fave phrase is "Hello World"!';
```

Make sure that the inside quotes don't match the surrounding ones because doing so would lead to an error:

```
console.log(favePhrase)


// output

//Uncaught SyntaxError: Unexpected identifier (at test.js:2:38)
```

Same thing happens when you try to use an apostrophe inside single quotes:

```
// Don't do this
let favePhrase = 'My fave phrase is "Hello world"! Isn't it awesome?';

console.log(favePhrase);

// output

// Uncaught SyntaxError: Unexpected identifier (at test.js:3:56)
```

I used double quotes inside single quotes, and that worked. However, when I introduced the apostrophe, the code broke.

The way to get this to work is to escape the single quotes by using the `\` escape character:

```
let favePhrase = 'My fave phrase is \'Hello World\'! ';

console.log(favePhrase);

// output

// My fave phrase is 'Hello World'!
```

And to make the apostrophe work, you would have to do the following:

```
let favePhrase = 'My fave phrase is "Hello world"! Isn\'t it awesome?';

console.log(favePhrase);
```

What is A Template Literal in JavaScript? Why and How to Use Template Literals in JavaScript

Earlier, you saw that to create a template literal, you have to use backticks.

Template literals were introduced with ES6, and they allow you to perform more complex operations using strings.

One of those is the ability to embed a variable inline inside a string, like so:

```
let firstName = 'John';
let lastName = 'Doe';

console.log(`Hi! My first name is ${firstName} and my last name is ${lastName}!`);

// output

//Hi! My first name is John and my last name is Doe!
```

In the example above, I created two variables, `firstName` and `lastName`, and stored a person's first and last name, respectively.

Then, using `console.log()`, I printed a string created with backticks, also known as a template literal.

Inside that string, I embedded those two variables.

To do so, I wrapped the variable names in `${}` - this is also known as **string interpolation** which allows you to introduce any variables without having to concatenate them like so:

```
let firstName = 'John';
let lastName = 'Doe';

console.log("Hi! My first name is " + firstName + " and my last name is " + lastName + "!")
```

Another thing that template literals allow you to do is to use single quotes, double quotes, and apostrophes inside them without the need to escape them:

```
let favePhrase = `My fave phrase is "Hello World" ! Isn't it awesome?`

console.log(favePhrase);

// output

// My fave phrase is "Hello World" ! Isn't it awesome?
```

String literals also allow you to create multiline strings, which you will learn how to do in the following section.

How to Create Multiline Strings in JavaScript

There are three ways to create strings that span multiple lines:

- By using template literals.
- By using the `+` operator – the JavaScript concatenation operator.
- By using the `\` operator – the JavaScript backslash operator and escape character.

If you choose to use single or double quotes instead of template literals to create a string that spans multiple lines, you would have to use either the `+` operator or the `\` operator.

How to Create Multiline Strings with Template Literals in JavaScript

Template literals allow you to create a string that spans multiple lines:

```
- Learn JavaScript
Use freeCodeCamp to learn all the above and much, much more !
`

console.log(learnCoding);


// output

// How to start learning web development?
// - Learn HTML
// - Learn CSS
// - Learn JavaScript
// Use freeCodeCamp to learn all the above and much, much more !
```

Using template literals is the most straightforward way of creating multiline strings.

How to Create Multiline Strings Using the `+` Operator in JavaScript

Taking the same example from the previous section, here is how you would re-write it using the `+` operator:

```
let learnCoding = 'How to start learning web development?' +
' - Learn HTML' +
' - Learn CSS' +
' - Learn JavaScript' +
' Use freeCodeCamp to learn all the above and much, much more!'


console.log(learnCoding);

// output

// How to start learning web development?  - Learn HTML - Learn CSS - Learn JavaScript Use
```

You would also need to include the `\n` newline character to make sentences appear on a new line:

```
let learnCoding = 'How to start learning web development?\n' +
' - Learn HTML\n' +
console.log(learnCoding);

// output

//How to start learning web development?
// - Learn HTML
// - Learn CSS
// - Learn JavaScript
// Use freeCodeCamp to learn all the above and much, much more!
```

How to Create Multiline Strings Using the `\` Operator in JavaScript

If you wanted to use the `\` operator, here is how you would re-write the example from the previous section:

```
let learnCoding = 'How to start learning web development? \n \
 - Learn HTML \n \
 - Learn CSS\n  \
 - Learn JavaScript \n \
Use freeCodeCamp to learn all the above and much, much more!'


console.log(learnCoding);

// output

// let learnCoding = 'How to start learning web development? \n \
// - Learn HTML \n \
// - Learn CSS\n  \
// - Learn JavaScript \n \
//Use freeCodeCamp to learn all the above and much, much more!'


console.log(learnCoding);
```

In this example, I created a multiline string using single quotes.

I first had to use the `\n` newline character followed by the `\` operator to make the string span multiple lines.

Make sure you place the `\` operator after the `\n` newline character.

And there you have it! You now know how to create multiline strings in JavaScript.

To learn more about JavaScript, head to freeCodeCamp's JavaScript Algorithms and Data Structures Certification.

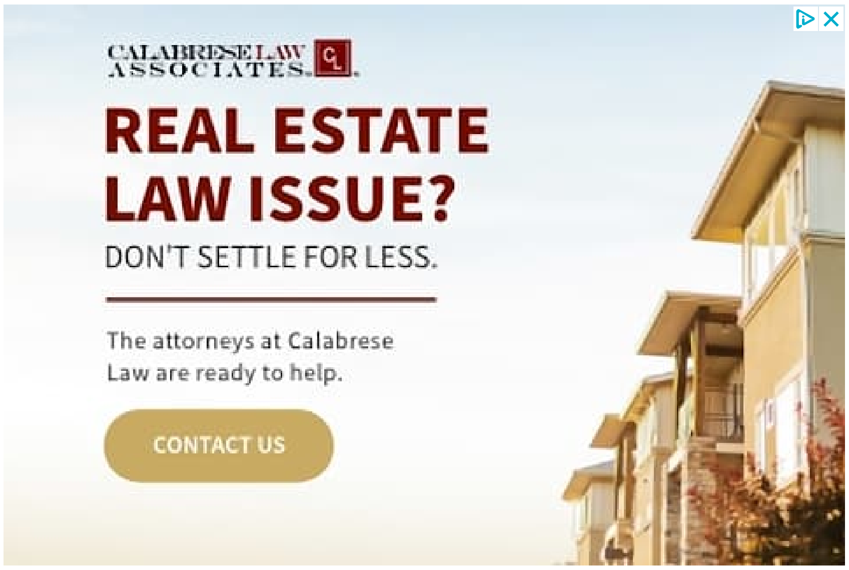
It's a free, well-thought-out, and structured curriculum where you will learn interactively. In the end, you will also build 5 projects to claim your certification and solidify your knowledge.

Thanks for reading!

Dionysia Lemonaki

Read more posts.

If this article was helpful, tweet it.



Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

You can make a tax-deductible donation here.

Trending Guides

What is a Framework?	SQL HAVING
What Do CS Majors Do?	What is OOP?
Discord Update Failed	HTML textarea
Center an Image in CSS	NVM for Windows
What is the MVC Model?	Git Revert File
JavaScript replaceAll()	GROUP BY in SQL
Python Switch Statement	2D Array in Java
Python string.replace()	How to Install NVM
What is a Relational DB?	Percentages in Excel
Split a String in Python	JavaScript Timestamp
Git List Remote Branches	Remove Item from Array JS
Git Delete Remote Branch	Dual Boot Windows + Ubuntu
Software Developer Career	Python Round to 2 Decimals
Three Dots Operator in JS	String to Int in JavaScript
How to Format Dates in JS	What's the .gitignore File?

Our Charity

About Alumni Network Open Source Shop Support Sponsors Academic Honesty Code of Conduct

Privacy Policy Terms of Service Copyright Policy