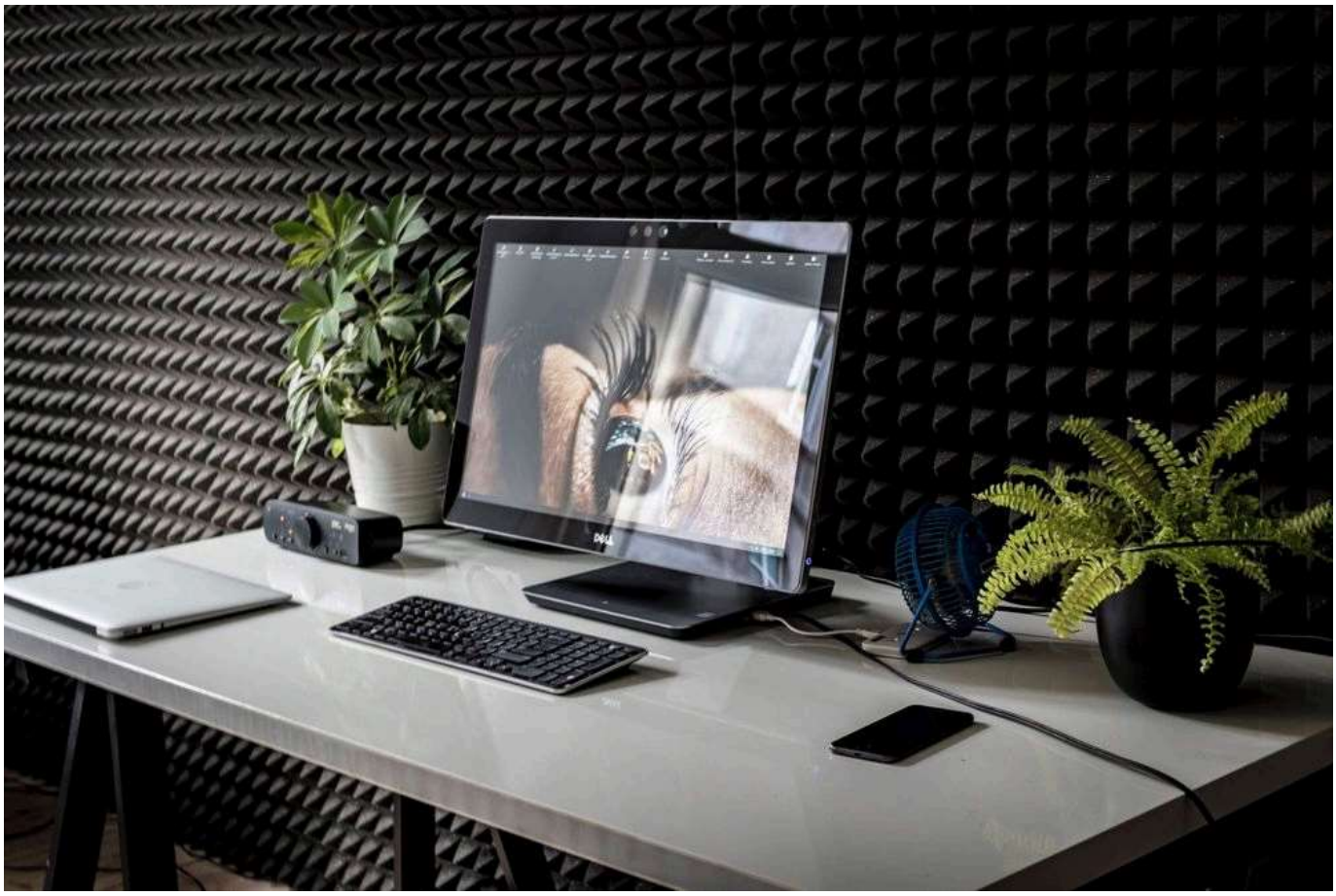
AUGUST 10, 2022 / #JAVASCRIPT

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

Dionysia Lemonaki



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).

- 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`.

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);
```

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:

```
// Don't do this
let favePhrase = 'My fave phrase is 'Hello World'! ';

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?';
```

```
// 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);

// output

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

## 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 ${last

// 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

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

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

// output

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

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:

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:

```
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 freeCodeCamp to learn all the above and much, much more !
```

# 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 - Lea
```

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' +
' - Learn CSS\n' +
' - Learn JavaScript\n' +
' Use freeCodeCamp to learn all the above and much, much more!'


console.log(learnCoding);

// output
```

```
// - 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.