Prior to ES6, the old way to concatenate strings together was by using the string concatenation operator ( + ).

**const** student = {

name: 'Richard Kalehoff',

guardian: 'Mr. Kalehoff'

};

**const** teacher = {

name: 'Mrs. Wilson',

room: 'N231'

}

**let** message = student.name + ' please see ' + teacher.name + ' in ' + teacher.room + ' to pick up your report card.';

***Returns:****Richard Kalehoff please see Mrs. Wilson in N231 to pick up your report card.*

This works alright, but it gets more complicated when you need to build multi-line strings.

**let** note = teacher.name + ',\n\n' +

'Please excuse ' + student.name + '.\n' +

'He is recovering from the flu.\n\n' +

'Thank you,\n' +

student.guardian;

***Returns:*** *Mrs. Wilson,*

*Please excuse Richard Kalehoff.  
He is recovering from the flu.*

*Thank you,  
Mr. Kalehoff*

However, that’s changed with the introduction of *template literals* (previously referred to as "template strings" in development releases of ES6).

***NOTE:****As an alternative to using the string concatenation operator ( + ), you can use the String's concat() method, but both options are rather clunky for simulating true*[*string interpolation*](https://en.wikipedia.org/wiki/String_interpolation)*.*

**Template Literals**

**Template literals** are essentially string literals that include embedded expressions.

Denoted with backticks (` `) instead of single quotes ( '' ) or double quotes ( "" ), template literals can contain placeholders which are represented using ${expression}. This makes it *much easier* to build strings.

Here's the previous examples using template literals.

**let** message = `${student.name} please see ${teacher.name} in ${teacher.room} to pick up your report card.`;

***Returns:****Richard Kalehoff please see Mrs. Wilson in N231 to pick up your report card.*

By using template literals, you can drop the quotes along with the string concatenation operator. Also, you can reference the object's properties inside expressions.

Here, you try. Change the greeting string below to use a template literal. Also, feel free to swap in your name for the placeholder.





/\*

\* Instructions: Change the `greeting` string to use a template literal.

\*/

const myName = ['souvik'];

const greeting = `Hello, my name is ${myName}`;

console.log(greeting);

* RESET QUIZ
* TEST RUN
* SUBMIT ANSWER

...but what about the multi-line example from before?

This silent animation shows how template literals make multi-line strings easier to read and more concise.  
  
This video does not have audio. It was created as a visual to aid learning.

Here’s where template literals really shine. In the code animation above, the quotes and string concatenation operator have been dropped, as well as the newline characters ( \n ). That's because template literals also preserve newlines as part of the string!

***TIP:****Embedded expressions inside template literals can do more than just reference variables. You can perform operations, call functions and use loops inside embedded expressions!*