Adding Variables

In this lesson, you'll create a simple Hello World program, and then you'll learn how you can customize the behavior of the program by adding variables to it.

This simple program just prints "Hello World!" to the console:

```
Console.WriteLine("Hello World!");
```

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Hello World!");
    }
}
```

Tip {.tip .newLanguage }
It's programming tradition that the first program one writes in a new language print out the phrase, "Hello World".

You can change this greeting to be more personalized by using a variable. On the line above this one, you can add a variable that holds your name, like this:

```
var name = "Steve"; // use your name here
```

There are a few new elements to this line of code. First, you're using a C# keyword, `var`, which you can think of as *variable*. The `var` keyword is shorthand for whatever the type on the other side of the assignment operator (`=`) might be. In this case, the value in double quotes (`"Steve"` in the example above) is a *string*. Strings are one of the built-in types in C#, and are used to represent text values. You can also declare a variable by specifying its type explicitly. In this example, the equivalent statement would be `string name = "Steve";`.

Tip {.tip .javascript}
Although Javascript also uses `var` for variable declaration, don't let that confuse you. The C# `var` is strongly typed, meaning that unlike Javascript's dynamic type system, the variable being declared will be of a specific type, just as if that type's name had been used to declare it.

The `//` on the line represents a single-line comment. Everything on the line that follows these two characters is ignored by the compiler. You can use these comments on a line all by themselves, or following other code as in this case. Comments are useful for explaining why you're doing something a certain way in your application, but avoid the temptation to overuse them or to use them to explain complicated code. A better solution is to make the code less complicated.

Now that you have a variable representing your name, you can use it in the next line so that the program greets *you*, rather than the world. To do that, remove the word *World*and replace it with `{name}`. Note that these are curly braces around the name of the variable. By using this convention, you're letting C# know that you want it to substitute the value of the variable `name` in that location. The last thing you need to do for this convention to work is prefix the string with a `$` sign. When completed, the two lines of code should look like this:

```
var name = "Steve"; // use your name here
Console.WriteLine($"Hello {name}!");
```

```
using System;

class Program
{
    static void Main()
    {
        var name = "Steve"; // use your name here
        Console.WriteLine($"Hello {name}!");
    }
}
```

Run the program.

Tip {.tip .CLI}
From a command prompt in the project folder, you can run the program by typing `dotnet run`.

Tip {.tip .visualstudio}
Within Visual Studio, ctrl+F5 will run the console application, launching a new console window.

You should see the output that includes your name (or "Steve" if you decided to just use the code above).