Implicitly typed local variables

2 minutes

The C# compiler affords many conveniences as you write your code. It can infer your variable's data type by its initialized value. In this unit, you'll learn about this feature, called implicitly typed local variables.

What are implicitly typed local variables?

An implicitly typed local variable is created using the `var` keyword, which instructs the C# compiler to infer the type. After the type is inferred, it's the same as if the actual data type had been used to declare the variable.

In the following example, we'll declare a variable using the `var` keyword instead of the `string` keyword.

C# Copy

```
var message = "Hello world!";
```

Because the variable `message` is immediately set to the string value `"Hello World!"`, the C# compiler understands the intent and treats every instance of message as an instance of type `string`.

In fact, the `message` variable is typed to be a string and can never be changed. Here, we'll attempt to set `message` to the literal decimal value of `10.0m`.

C# Copy Run

```
var message = "Hello World!";
message = 10.0m;
```

If you run this code, you'll see the following error message.

Output Copy

```
(2,11): error CS0029: Cannot implicitly convert type 'decimal' to 'string'
```

> **Note**
>
> Other programming languages use the `var` keyword differently. In C#, the variable is statically typed by the compiler regardless of whether you use the actual data type or allow the compiler to infer the data type. In other words, the type is locked in at the time of declaration and therefore will never be able to hold values of a different data type.

You can only use the `var` keyword if the variable is initialized

It's important to understand that the `var` keyword is dependent on the value you use to initialize the variable. If you try to use the `var` keyword without initializing the variable, you'll receive an error when you attempt to compile your code.

C# Copy Run

```
var message;
```

If you attempt to run this code, as it compiles you'll see the following output:

Output Copy

```
(1,5): error CS0818: Implicitly-typed variables must be initialized
```

Why use the `var` keyword?

The `var` keyword has been widely adopted in the C# community, so it's likely if you look at a code example in a book or online, you'll see the `var` keyword used instead of the actual data type name. So, we wanted to make sure to introduce it in this module.

However, the `var` keyword has an important use in C#. For reasons that may not be clear to you until you write advanced code, there are situations where the data type may not be obvious at the time you initialize the variable. In fact, in some cases, C# may invent a new data type just for your code and may not be able to give it a predictable name ahead of time. Again, this is an advanced feature of C# that will be covered in other modules.

As you get started, we recommend you continue to use the actual data type name when you declare variables. Using the data type when you declare variables will help you be purposeful as you write your code.

Recap

The most important takeaways from this unit about the `var` keyword and implicitly typed local variables:

- The `var` keyword tells the compiler to infer the data type of the variable based on the value it is initialized to.
- You'll likely see the `var` keyword as you read other people's code; however, you should use the data type when possible.

Next unit: Challenge

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

English (United States) Theme

.NET Editor

Press CTRL+M, TAB to exit the editor

Clear Run

```
1
```

Output