100 XP

Declare implicitly typed local variables

6 minutes

The C# compiler works behind the scenes to assist you 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 by using the `var` keyword followed by a variable initialization. For example:

C#

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

In this example, a string variable was created using the `var` keyword instead of the `string` keyword.

The `var` keyword tells the C# compiler that the data type is *implied* by the assigned value. After the type is implied, the variable acts the same as if the actual data type had been used to declare it. The `var` keyword is used to save on keystrokes when types are lengthy or when the type is obvious from the context.

In the example:

C#

```
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. For example, consider the following code:

C#

```
var message = "Hello World!";
message = 10.703m;
```

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

Output

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

Note

Other programming languages use the `var` keyword differently. In C#, variables are assigned a type by the compiler regardless of whether you use the actual data type name or allow the compiler to imply 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.

Variables using the `var` keyword must be 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#

```
var message;
```

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

Output

```
(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. It's likely that 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 it's important to understand its usage.

The `var` keyword has an important use in C#. Many times, the type of a variable is obvious from its initialization. In those cases, it's simpler to use the `var` keyword. The `var` keyword can also be useful when planning the code for an application. When you begin developing code for a task, you may not immediately know what data type to use. Using `var` can help you develop your solution more dynamically.

As you get started, it is recommended that you continue to use the actual data type name when you declare variables until you become more comfortable working with code. Using the data type when you declare variables will help you be purposeful as you write your code.

Recap

Here's what you've learned about the `var` keyword so far:

- 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.

Module complete:

Unlock achievement

.NET Editor

Press CTRL + M, TAB to exit the editor

Clear Run

Output