Declare variables

3 minutes

A literal is *literally* a hard-coded value. However, most applications will require us to work with values that we don't know much about ahead of time. In other words we'll need to work with data that comes from users, from files, or from across the network.

When we need to work with data from outside of our code, we'll declare a variable.

What is a variable?

A **variable** is a data item that may change its value during its lifetime. You use variables to temporarily store values that you intend to use later in your code. A variable is a friendly label that we can assign to a computer memory address. When we want to temporarily store a value in that memory address, or whenever we want to retrieve the value that is stored in the memory address, we just use the variable name we created.

Declaring a variable

To create a new variable, you must first declare the data type of the variable, then give it a name.

C# Copy

```
string firstName;
```

In this case, we're creating a new variable of type `string` called `firstName`. From now on, this variable can only hold string values.

I can choose any name as long as it adheres to a few C# syntax rules for naming variables.

Variable name rules and conventions

A software developer once famously said "The hardest part of software development is naming things." Not only does the name of a variable have to follow certain syntax rules, it should also be used to make the code more human-readable and understandable. That's a lot to ask of one line of code!

Here's a few important considerations about variable names:

- Variable names can contain alphanumeric characters and the underscore character. Special characters like the hash symbol `#` (also known as the number symbol or pound symbol) or dollar symbol `$` are not allowed.
- Variable names must begin with an alphabetical letter or an underscore, not a number. Developers use the underscore for a special purpose, so try to not use that for now.
- Variable names must **not** be a C# keyword. For example, you cannot use the following variable declarations: `decimal decimal;` or `string string;`.
- Variable names are case-sensitive, meaning that `string Value;` and `string value;` are two different variables.
- Variable names should use **camel case**, which is a style of writing that uses a lower-case letter at the beginning of the first word and an upper-case letter at the beginning of each subsequent word. For example, `string thisIsCamelCase;`.
- Variable names should be descriptive and meaningful in your application. Choose a name for your variable that represents the kind of data it will hold.
- Variable names should be one or more entire words appended together. Don't use contractions because the name of the variable (and therefore, its purpose) may be unclear to others who are reading your code.
- Variable names shouldn't include the data type of the variable. You might see some advice to use a style like `string strValue;`. That advice is no longer current.

The example `string firstName;` follows all of these rules and conventions, assuming I want to use this variable to store data that represents someone's first name.

Variable name examples

Here's a few examples of variable declarations using the data types we learned about previously.

C# Copy

```csharp
char userOption;

int gameScore;

decimal particlesPerMillion;

bool processedCustomer;
```

Recap

Here's the main takeaways you've learned so far about variables:

- Variables are temporary values you store in the computer's memory.
- Before you can use a variable, you have to declare it.
- To declare a variable, you first select a data type for the kind of data you want to store, and then give the variable a name that follows the rules.

Now that we know how to declare a variable, let's learn how to set, retrieve, and initialize the value of a variable.

Next unit: Exercise - Setting and getting values from variables

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆

.NET Editor

Press CTRL + M, TAB to exit the editor

Clear Run

```
```

Output