INTRODUCTION TO C#

Lesson 1.01 – C# Syntax

https://alextushinsky.com

C# 101

Version 1.0 of C# was first introduced in 2002, as a modern, object-oriented programming language. Today, we are on C# version 9. C# can be used with .NET Framework and .NET 5 (.NET Core)

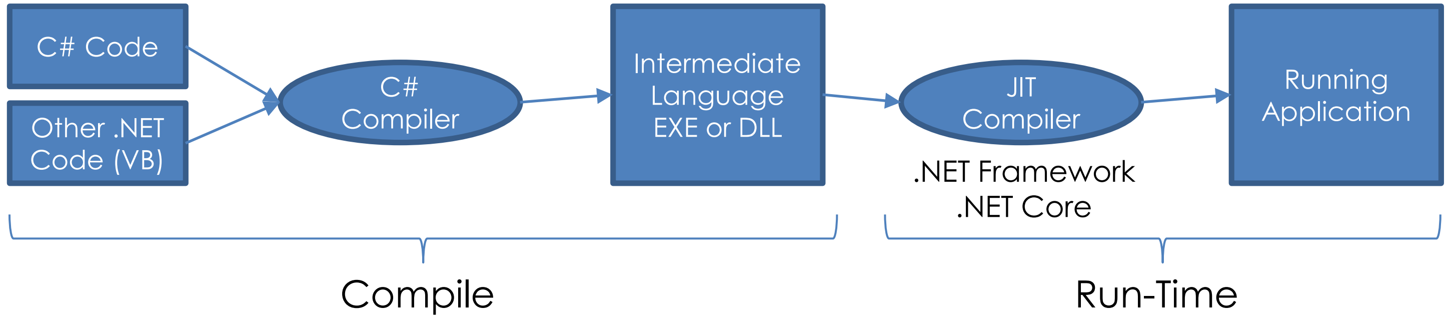
The language is designed to be straight-forward, and can be used for general purpose programming, such as web or desktop applications, as well as mobile, and embedded systems applications.

C# follows basic principals of object-oriented programming, and is very similar in nature to other languages, such as C++ and Java.

C# 101

C# is a high-level programming language, meaning that we write code similarly to how we speak and understand written languages. For it to be read and understood by machines, we need to translate that high-level code into low-level or machine-readable code. That process is known as "compiling." This is the process of transforming your source code from one language to another. This is necessary because computers only understand machine language, as such C#, Visual Basic, or Java aren't understood. The compiler translates those languages to something the machine can understand and work with.

In C#, there are two different types of compiling methods that take place. The first is a CLR or common-language-runtime compile, and the second is a JIT or just-in-time compilation.



Naming Standards

Here are some definitions that you'll find useful as you write C# code, as well as in the next section where we talk about various C# conventions.

- **PascalCase** – Capitalize each word, removing any spaces.
  Ex: UserAccount, AdminControlPanel
  Used for: Namespace, Class, Struct, Interface, Method, Property

- **camelCase** - The first word is not capitalized, but the rest are, all spaces removed.
  Ex: userAccount, adminControlPanel
  Used for: Parameters, Variables

- **_underScore** – Start with an underscore "_" symbol, then follow camelCase.
  Ex: _userAccount, _adminControlPanel
  Used for: Private class members.

Syntax

Each C# statement needs to be ended with a semi-colon.

Statements can be broken down into multiple lines, with the semi-colon added only to the last line of the statement.

Example:

```
for (int i = DateTime.Now.Year + 7; i > DateTime.Now.Year - 10; i--)            {
      lstNewFacilityYear.Items.Add(new
                                   Telerik.Web.UI.RadComboBoxItem(i.ToString(),
                                   i.ToString()));
}
lstNewFacilityYear.SelectedValue = DateTime.Now.Year.ToString();
GenUtils.WriteEx(ex);
lblSourceFac.Text = SV.facility_name;
```

Syntax

Groups of statements, known as "**code blocks**", are always surrounded by the **{** and **}** braces. A code block can define an action for a condition, the body of a method, contents of a loop, or anywhere a group of statements make up a cohesive function.

Example:

```
for (int i = DateTime.Now.Year + 7; i > DateTime.Now.Year - 10; i--)
{
      lstNewFacilityYear.Items.Add(new
                  Telerik.Web.UI.RadComboBoxItem(i.ToString(),
                  i.ToString()));
}
```

Comments

C# lets you create several different types of comments within your code. Comments are meant to relate information regarding the component in question.

XML Comments (see https://docs.microsoft.com/en-us/dotnet/csharp/codedoc for more detail):

```
/// <summary>
/// This class does something important.
/// </summary>
/// <param name="args"></param>
```

Single line comment:

```
// This is a comment
```

Multi-line comment:

```
/* This is a comment
   This is line 2 of that comment.  */
```