**Visual Studio 2019 IDE Overview**

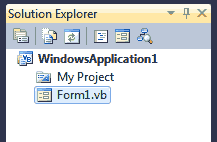
1. Different types of windows (solution exp.,properties etc.)

## **Toolbox Window**

The Toolbox window contains all the controls you can use to build your application’s interface. This window is usually retracted, and you must move the pointer over it to view the Toolbox. The controls in the Toolbox are organized in various tabs, so take a look at them to become familiar with the controls and their functions.

## **Solution Explorer Window**

The Solution Explorer window contains a list of the items in the current solution. A solution can contain multiple projects, and each project can contain multiple items. The Solution Explorer displays a hierarchical list of all the components, organized by project. You can right-click any component of the project and choose Properties in the context menu to see the selected component’s properties in the Properties window. If you select a project, you will see the Project Properties dialog box. You will find more information on project properties in the following chapter.



If the solution contains multiple projects, you can right-click the project you want to become the startup form and select Set As StartUp Project. You can also add items to a project with the Add Item command of the context menu, or remove a component from the project with the Exclude From Project command. This command removes the selected component from the project, but doesn’t affect the component’s file on the disk. The Delete command removes the selected component from the project and also deletes the component’s file from the disk.

## **Properties Window**

This window (also known as the Properties Browser) displays all the properties of the selected component and its settings. Every time you place a control on a form, you switch to this window to adjust the appearance of the control. You have already seen how to manipulate the properties of a control through the Properties window.

Many properties are set to a single value, such as a number or a string. If the possible settings of a property are relatively few, they’re displayed as meaningful constants in a drop-down list. Other properties are set through a more elaborate interface. Color properties, for example, are set from within a Color dialog box that’s displayed right in the Properties window. Font properties are set through the usual Font dialog box. Collections are set in a Collection Editor dialog box, in which you can enter one string for each item of the collection, as you did for the items of the ComboBox control earlier in this chapter.

If the Properties window is hidden, or if you have closed it, you can either choose *View > Properties* Window, or right-click a control on the form and choose Properties. Or you can simply press F4 to bring up this window. There will be times when a control might totally overlap another control, and you won’t be able to select the hidden control and view its properties. In this case, you can select the desired control in the ComboBox at the top of the Properties window. This box contains the names of all the controls on the form, and you can select a control on the form by selecting its name on this box.

1. Solution,Project

## **Projects**

When you create an app or website in Visual Studio, you start with a *project*. In a logical sense, a project contains all files that are compiled into an executable, library, or website. Those files can include source code, icons, images, data files, and so on. A project also contains compiler settings and other configuration files that might be needed by various services or components that your program communicates with.

## **solutions**

A project is contained within a *solution*. Despite its name, a solution is not an "answer". It's simply a container for one or more related projects, along with build information, Visual Studio window settings, and any miscellaneous files that aren't associated with a particular project. A solution is described by a text file (extension *.sln*) with its own unique format; it's not intended to be edited by hand.

1. Code editor features

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Syntax Coloring | Some syntax elements of code and markup files are colored differently to distinguish them. For example, keywords (such as using in C# and Imports in Visual Basic) are one color, but types (such as Console and Uri) are another color. Other syntax elements are also colorized, such as string literals and comments. C++ uses color to differentiate among types, enumerations, and macros, among other tokens. |
| Error and Warning Marks | As you add code and build your solution, you may see (a) different-colored wavy underlines (known as squiggles) or (b) light bulbs appearing in your code. Red squiggles denote syntax errors, blue denotes compiler errors, green denotes warnings, and purple denotes other types of errors. [Quick Actions](https://docs.microsoft.com/en-us/visualstudio/ide/quick-actions?view=vs-2019) suggest fixes for problems and make it easy to apply the fix. |
| Printing | You can use the options in the Print dialog box to include line numbers or hide collapsed regions of code when you print a file. In the Page Setup dialog box, you can also choose to print the full path and the name of the file by choosing Page header.  You can set color printing options in the Tools > Options > Environment > Fonts and Colors dialog box. Choose Printer in the Show settings for list to customize color printing. You can specify different colors for printing a file than for editing a file. |

1. Shortcuts

### **Analyze**

|  |  |
| --- | --- |
| **Commands** | **Keyboard shortcuts** |
| Analyze.NavigateBackward | Shift+Alt+3 |
| Analyze.NavigateForward | Shift+Alt+4 |

### **Architecture**

|  |  |
| --- | --- |
| **Commands** | **Keyboard shortcuts** |
| Architecture.NewDiagram | Ctrl+\, Ctrl+N |

### **Build**

|  |  |
| --- | --- |
| **Commands** | **Keyboard shortcuts** |
| Build.BuildSelection | Ctrl+B (Visual Studio 2019) |
| Build.BuildSolution | Ctrl+Shift+B |
| Build.Cancel | Ctrl+Break |
| Build.Compile | Ctrl+F7 |
| Build.RunCodeAnalysis Solution | Alt+F11 |

### **Class View context menus**

|  |  |
| --- | --- |
| **Commands** | **Keyboard shortcuts** |
| ClassViewContextMenus.ClassViewMultiselectProjectreferencesItems.Properties | Alt+Enter |

### **Debug**

|  |  |
| --- | --- |
| **Commands** | **Keyboard shortcuts** |
| Debug.ApplyCodeChanges | Alt+F10 |
| Debug.Autos | Ctrl+Alt+V, A |
| Debug.BreakAll | Ctrl+Alt+Break |
| Debug.Breakpoints | Ctrl+Alt+B |
| Debug.CallStack | Ctrl+Alt+C |
| Debug.DeleteAllBreakpoints | Ctrl+Shift+F9 |
| Debug.DiagnosticsHub.Launch | Alt+F2 |
| Debug.Disassembly | Ctrl+Alt+D |
| Debug.DOMExplorer | Ctrl+Alt+V, D |
| Debug.EnableBreakpoint | Ctrl+F9 |
| Debug.Exceptions | Ctrl+Alt+E |
| Debug.FunctionBreakpoint | Ctrl+K, B (Visual Studio 2019)  Ctrl+B (Visual Studio 2017) |
| Debug.GoToPreviousCallorIntelliTraceEvent | Ctrl+Shift+F11 |
| Debug.Graphics.StartDiagnostics | Alt+F5 |
| Debug.Immediate | Ctrl+Alt+I |
| Debug.IntelliTraceCalls | Ctrl+Alt+Y, T |
| Debug.IntelliTraceEvents | Ctrl+Alt+Y, F |
| Debug.JavaScriptConsole | Ctrl+Alt+V, C |
| Debug.Locals | Ctrl+Alt+V, L |
| Debug.LocationToolbar.ProcessCombo | Ctrl+5 |
| Debug.LocationToolbar.StackFrameCombo | Ctrl+7 |
| Debug.LocationToolbar.ThreadCombo | Ctrl+6 |
| Debug.LocationToolbar.ToggleCurrentThreadFlaggedState | Ctrl+8 |
| Debug.LocationToolbar.ToggleFlaggedThreads | Ctrl+9 |
| Debug.Memory1 | Ctrl+Alt+M, 1 |
| Debug.Memory2 | Ctrl+Alt+M, 2 |
| Debug.Memory3 | Ctrl+Alt+M, 3 |
| Debug.Memory4 | Ctrl+Alt+M, 4 |
| Debug.Modules | Ctrl+Alt+U |
| Debug.ParallelStacks | Ctrl+Shift+D, S |
| Debug.ParallelWatch1 | Ctrl+Shift+D, 1 |
| Debug.ParallelWatch2 | Ctrl+Shift+D, 2 |
| Debug.ParallelWatch3 | Ctrl+Shift+D, 3 |
| Debug.ParallelWatch4 | Ctrl+Shift+D, 4 |
| Debug.Processes | Ctrl+Alt+Z |
| Debug.QuickWatch | Shift+F9 or Ctrl+Alt+Q |
| Debug.RefreshWindowsapp | Ctrl+Shift+R |
| Debug.Registers | Ctrl+Alt+G |
| Debug.Restart | Ctrl+Shift+F5 |
| Debug.RunToCursor | Ctrl+F10 |
| Debug.SetNextStatement | Ctrl+Shift+F10 |
| Debug.ShowCallStackonCodeMap | Ctrl+Shift+` |
| Debug.ShowNextStatement | Alt+Num \* |
| Debug.Start | F5 |
| Debug.StartWindowsPhoneApplicationAnalysis | Alt+F1 |
| Debug.StartWithoutDebugging | Ctrl+F5 |
| Debug.StepInto | F11 |
| Debug.StepIntoCurrentProcess | Ctrl+Alt+F11 |
| Debug.StepIntoSpecific | Shift+Alt+F11 |
| Debug.StepOut | Shift+F11 |
| Debug.StepOutCurrentProcess | Ctrl+Shift+Alt+F11 |
| Debug.StepOver | F10 (When debugging: Performs a step over action) |
| Debug.StepOver | F10 (When not debugging: Starts debugging and stops on the first line of user code) |
| Debug.StepOverCurrentProcess | Ctrl+Alt+F10 |
| Debug.StopDebugging | Shift+F5 |
| Debug.StopPerformanceAnalysis | Shift+Alt+F2 |
| Debug.Tasks | Ctrl+Shift+D, K |
| Debug.Threads | Ctrl+Alt+H |
| Debug.ToggleBreakpoint | F9 |
| Debug.ToggleDisassembly | Ctrl+F11 |
| Debug.Watch1 | Ctrl+Alt+W, 1 |
| Debug.Watch2 | Ctrl+Alt+W, 2 |
| Debug.Watch3 | Ctrl+Alt+W, 3 |
| Debug.Watch4 | Ctrl+Alt+W, 4 |

**Project Types**

1. Windows App, Class Library

**Github folder inside Module 1 WindowsFormApplication1**

1. Web Application

**Github folder inside Module 1 WebApplication1**

**Create First C# Program "Hello World"**

1. What is namespace?

A namespace is designed for providing a way to keep one set of names separate from another. The class names declared in one namespace does not conflict with the same class names declared in another.

1. What is class?

A class is a user-defined blueprint or prototype from which objects are created. Basically, a class combines the fields and methods(member function which defines actions) into a single unit. In C#, classes support polymorphism, inheritance and also provide the concept of derived classes and base classes.

1. Variable & Method Declaration

variable is a named location in a memory where a program can manipulate the data. This location is used to hold the value of the variable.

A method declaration can provide more information about the method, including the return type of the method, the number and type of the arguments required by the method, and which other classes and objects can call the method. The next table shows all possible elements of a method declaration

**Understanding C# Program**

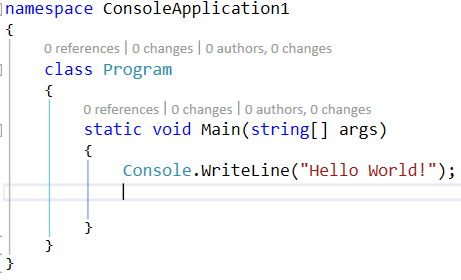
1. Program Flow

**github/module1/ConsoleApplication1**

In C# language there are several keywords that are used to alter the flow of the program. When the program is run, the statements are executed from the top of the source file to the bottom. One by one. This flow can be altered by specific keywords. Statements can be executed multiple times. Some statements are called conditional statements. They are executed only if a specific condition is met. --

1. Understanding Syntax

**github/module1/ConsoleApplication1**



Line 1: namespace is a used to organize your code, and it is a container for classes and other namespaces.

Line 2: A blank line. C# ignores white space. However, multiple lines makes the code more readable.

Line 4: The curly braces {} marks the beginning and the end of a block of code.

Line 5: Class is a container for data and methods, which brings functionality to your program. Every line of code that runs in C# must be inside a class. In our example, we named the class Program.

Line 7: Another thing that always appear in a C# program, is the main method. Any code inside its curly brackets {} will be executed. You don't have to understand the keywords before and after Main. You will get to know them bit by bit while reading this tutorial.

Line 9: Console is a class of the System namespace, which has a WriteLine() method that is used to output/print text. In our example it will output "Hello World!".

**Working with code files, projects & solutions**

1. Understanding structure of solution

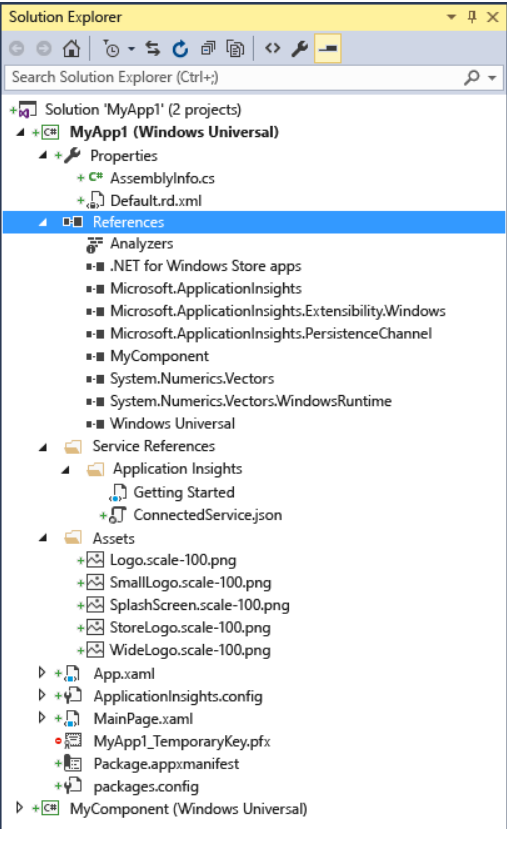
A project is contained within a *solution*. Despite its name, a solution is not an "answer". It's simply a container for one or more related projects, along with build information, Visual Studio window settings, and any miscellaneous files that aren't associated with a particular project. A solution is described by a text file (extension *.sln*) with its own unique format; it's not intended to be edited by hand.

Visual Studio uses two file types (*.sln* and *.suo*) to store settings for solutions:

|  |  |  |
| --- | --- | --- |
| **Extension** | **Name** | **Description** |
| .sln | Visual Studio Solution | Organizes projects, project items, and solution items in the solution. |
| .suo | Solution User Options | Stores user-level settings and customizations, such as breakpoints. |

1. Understanding structure of project (Win app, web app,web api, class library).

After you create a new project, you can use Solution Explorer to view and manage the project and solution and their associated items. The following illustration shows Solution Explorer with a C# solution that contains two projects:



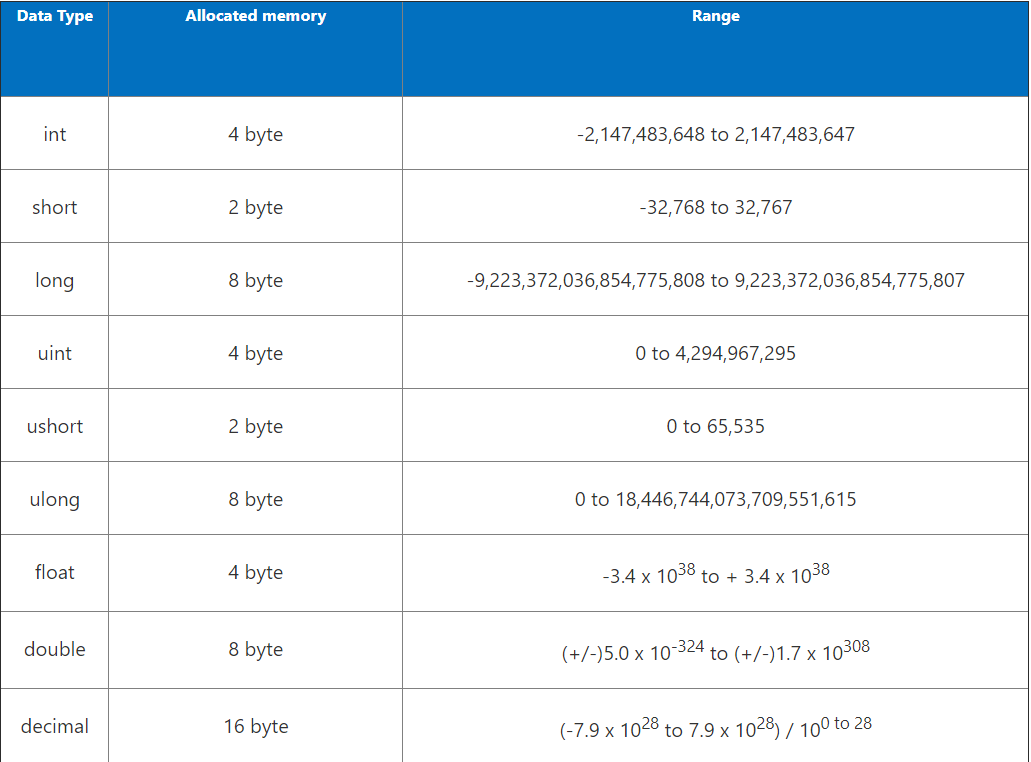
1. Familiar with different type of file extensions

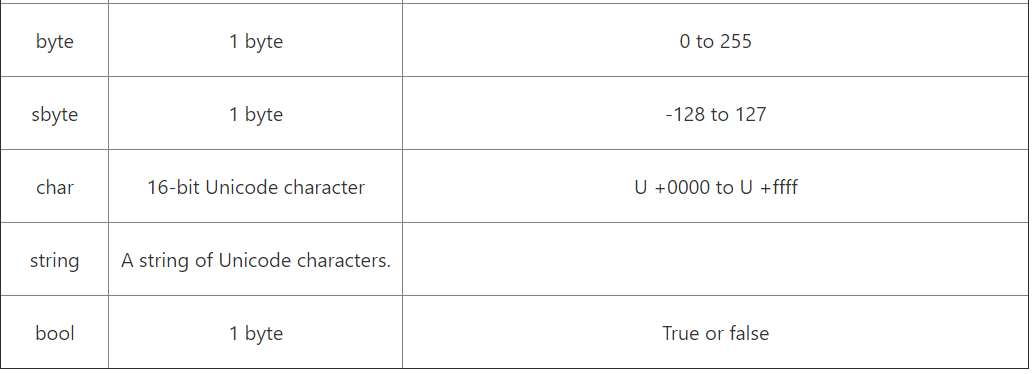
The following table shows common files in a Visual Studio project, and identifies them with their file extension.

|  |  |  |
| --- | --- | --- |
| **File extension** | **Type** | **Contents** |
| .asmx | Source | Deployment file. |
| .asp | Source | Active Server Page file. |
| .atp | Project | Application template project file. |
| .bmp, .dib, .gif, .jpg, .jpe, .png | Resource | General image files. |
| .bsc | Compiling | The browser code file. |
| .cpp, .c | Source | Main source code files for your application. |
| .cur | Resource | Cursor bitmap graphic file. |
| .dbp | Project | Database project file. |
| .disco | Source | The dynamic discovery document file. Handles XML Web service discovery. |
| .exe, .dll | Project | Executable or dynamic-link library files. |
| .h | Source | A header (include) file. |
| .htm, .html, .xsp, .asp, .htc, .hta, .xml | Resource | Common Web files. |
| .HxC | Project | Help project file. |
| .ico | Resource | Icon bitmap graphic file. |
| .idb | Compiling | The state file, containing dependency information between source files and class definitions. It can be used by the compiler during incremental compilation. Use the [/Fd](https://docs.microsoft.com/en-us/cpp/build/reference/fd-program-database-file-name?view=vs-2019) compiler option to specify the name of the .idb file. |
| .idl | Compiling | An interface definition language file. For more information, see [Interface Definition (IDL) File](https://docs.microsoft.com/en-us/windows/win32/Rpc/the-interface-definition-language-idl-file) in the Windows SDK. |
| .ilk | Linking | Incremental link file. For more information, see [/INCREMENTAL](https://docs.microsoft.com/en-us/cpp/build/reference/incremental-link-incrementally?view=vs-2019). |
| .map | Linking | A text file containing linker information. Use the [/Fm](https://docs.microsoft.com/en-us/cpp/build/reference/fm-name-mapfile?view=vs-2019) compiler option to name the map file. For more information, see [/MAP](https://docs.microsoft.com/en-us/cpp/build/reference/map-generate-mapfile?view=vs-2019). |
| .mfcribbon-ms | Resource | A resource file that contains the XML code that defines the MFC buttons, controls, and attributes in the ribbon. For more information, see [Ribbon Designer](https://docs.microsoft.com/en-us/cpp/mfc/ribbon-designer-mfc?view=vs-2019). |
| .obj, .o |  | Object files, compiled but not linked. |
| .pch | Debug | Precompiled header file. |
| .rc, .rc2 | Resource | [Resource script files](https://docs.microsoft.com/en-us/cpp/windows/working-with-resource-files?view=vs-2019) to generate resources. |
| .sbr | Compiling | Source browser intermediate file. The input file for [BSCMAKE](https://docs.microsoft.com/en-us/cpp/build/reference/bscmake-options?view=vs-2019). |
| .sln | Solution | The [solution](https://docs.microsoft.com/en-us/visualstudio/ide/solutions-and-projects-in-visual-studio) file. |
| .suo | Solution | The solution options file. |
| .txt | Resource | A text file, usually the "readme" file. |

**Understanding datatypes & variables with conversion**

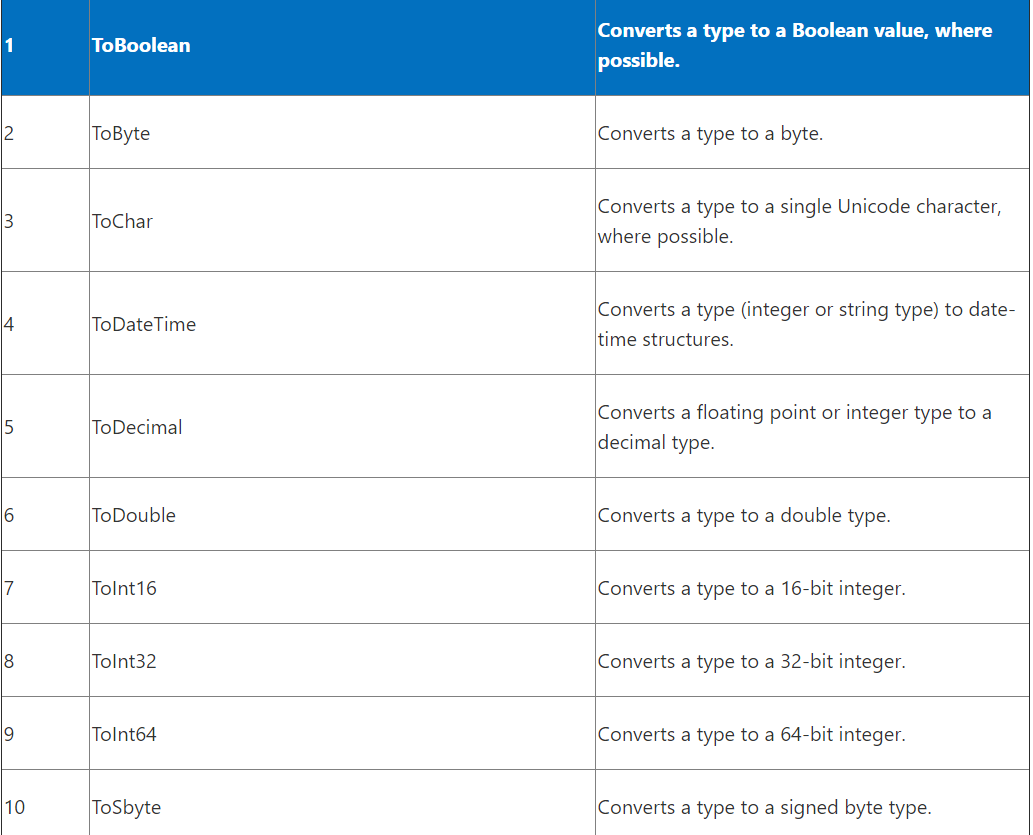
1. Base datatype

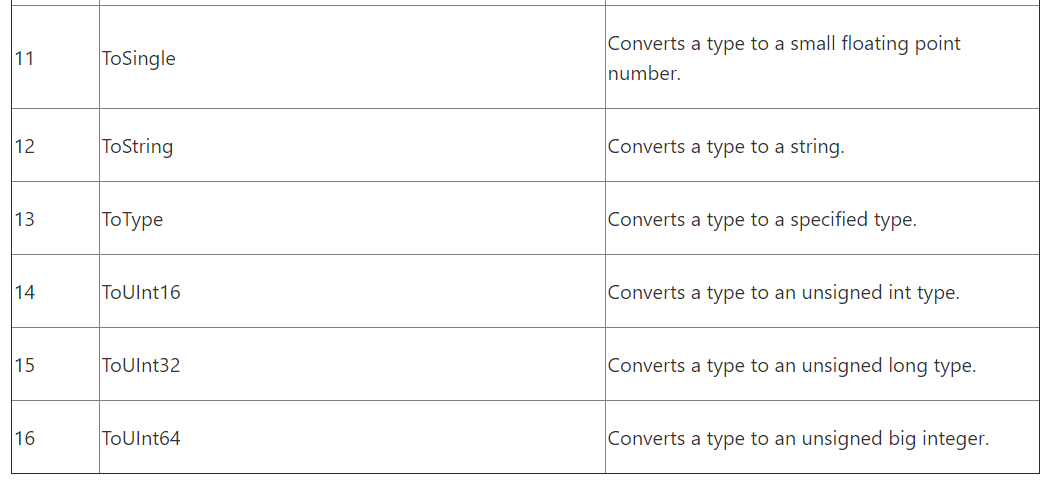




1. Datatype Conversion

**github/module1/basicdatatypeconversion**





1. Boxing/Unboxing

**github/module1/BoxingUnboxingexample**

**Understanding Decision making & statements**

1. if else,switch

**github/module1/DecisionMaking\_switch**