.NET Framework 4.7 and C# 8.0

Lesson 04 : Using Microsoft Visual Studio

Capgemini

Lesson Objectives

- After completing this module you will understand:
  - Debugging techniques in .NET
  - Creating multiple projects in one solution

Overview

- Debugging is about running your code line by line to ensure that the execution path and data are both correct
- Debugging is trickier and difficult as compared to writing a code
- Around 30 percent of the developer’s time is spent in debugging
- Visual Studio offers many debugging features

Debugging Tools

- Debugging Tools
  - Integrated Debugger
  - Visual Debugger

Features

- Debugging Offers:
  - Code execution examination. Step line by line
  - Viewing variable values at each step
  - Changing the value of a certain variable
  - Moving the execution point and running application to another point

- These features can be categorized as:
  - Breakpoints
  - Stepping
  - Data Viewing

Overview

- Places in code, where the debugger stops application execution
- To set the breakpoint, click the gray margin to the left of the line
- Point is visible as a red circle in the left gray margin
  - To clear the breakpoint, click the red circle or press F9.

Overview (Contd…)

Example1 - Microsoft Visual Studio

File Edit View Website Build Debug Tools Window Community Help

Debug .NET

Solution Explor…le1' (1 project) | Output | Default.aspx.vb | Default.aspx

Button1 | Click

```
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object, ByVal e
        Dim str As String
        Dim i As Integer

        If TextBox1.Text Is Nothing Then
            Label1.Text = "Please, enter your name. "
            Return
        End If

        str = TextBox1.Text
        i = str.Length
        Label1.Text = "Hello " + str
    End Sub
End Class
```

Ready Ln 9 Col 26 Ch 26 INS

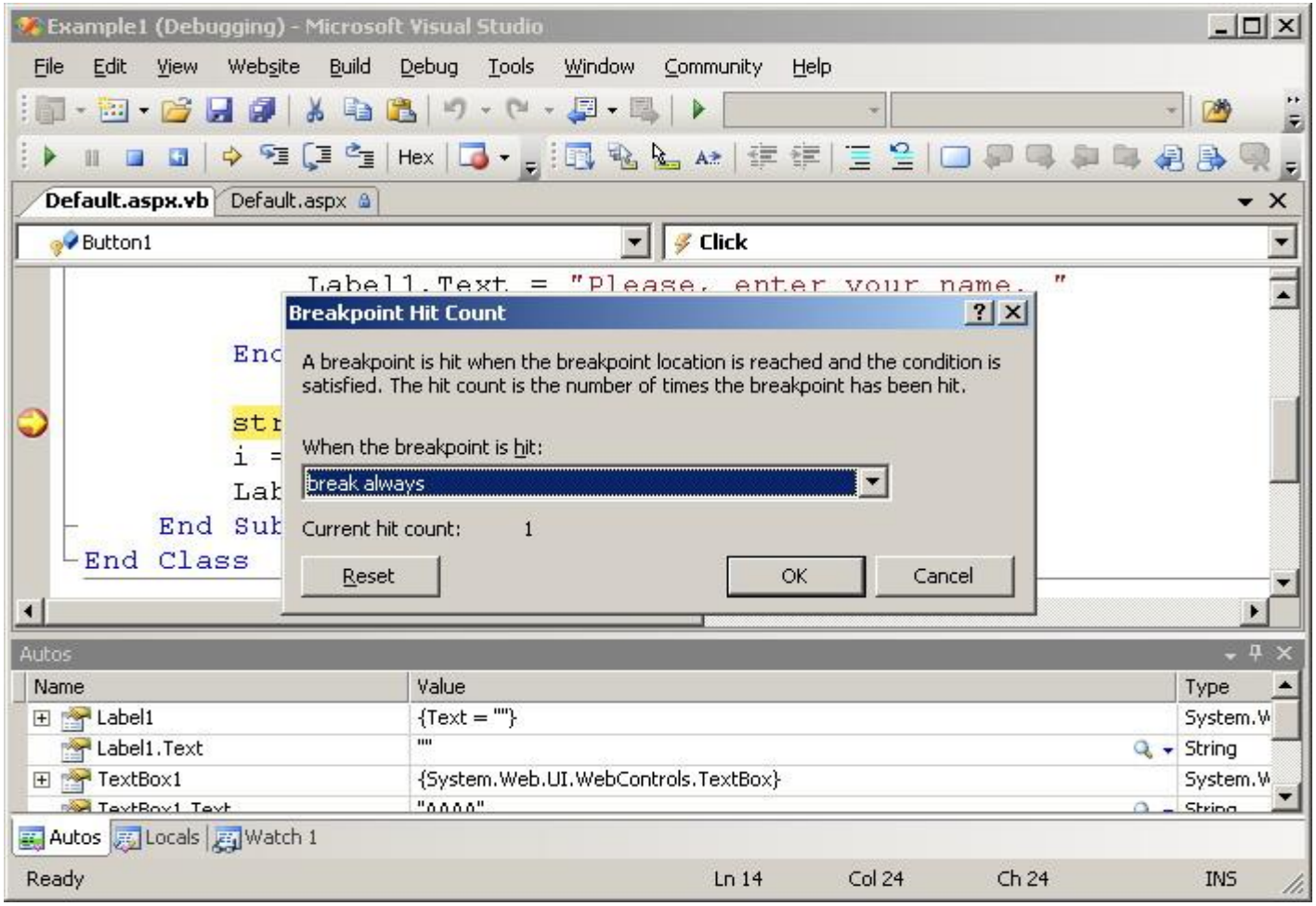
Advantages of Breakpoints

- No code required to be added to your program
- Set or disable breakpoints without changing source codes
- Pause the execution of a program at any point
- Breakpoint management is simple

Important Features of Breakpoints

- Hit Count:
  - Specify how many times you wish to hit your breakpoint before the application breaks.
  - Useful when one has to deal with loops.
- Condition:
  - Set a condition.
    - Breakpoint is hit when this condition evaluates to TRUE.

Setting a Hit Count



Setting a Breakpoint Condition

Example1 - Microsoft Visual Studio

File Edit View Website Build Debug Tools Window Community Help

Debug .NET

Solution Explor...le1' (1 project) | Start Page | Output | Default.aspx.vb | Default.aspx

Button1 | Click

```
    Inheri
    Protec                                          ct, ByVal e
        Di
        Di
        If
        En
        str = TextBox1.Text
        i = str.Length
        Label1.Text = "Hello " + str
    End Sub
End Class
```

Breakpoint Condition

When the breakpoint location is reached, the expression is evaluated and the breakpoint is hit only if the expression is true or has changed.

☑ Condition:

i>5

◉ Is true

○ Has changed

OK | Cancel

Ready | Ln 16 | Col 23 | Ch 23 | INS

Demo: BreakPoints

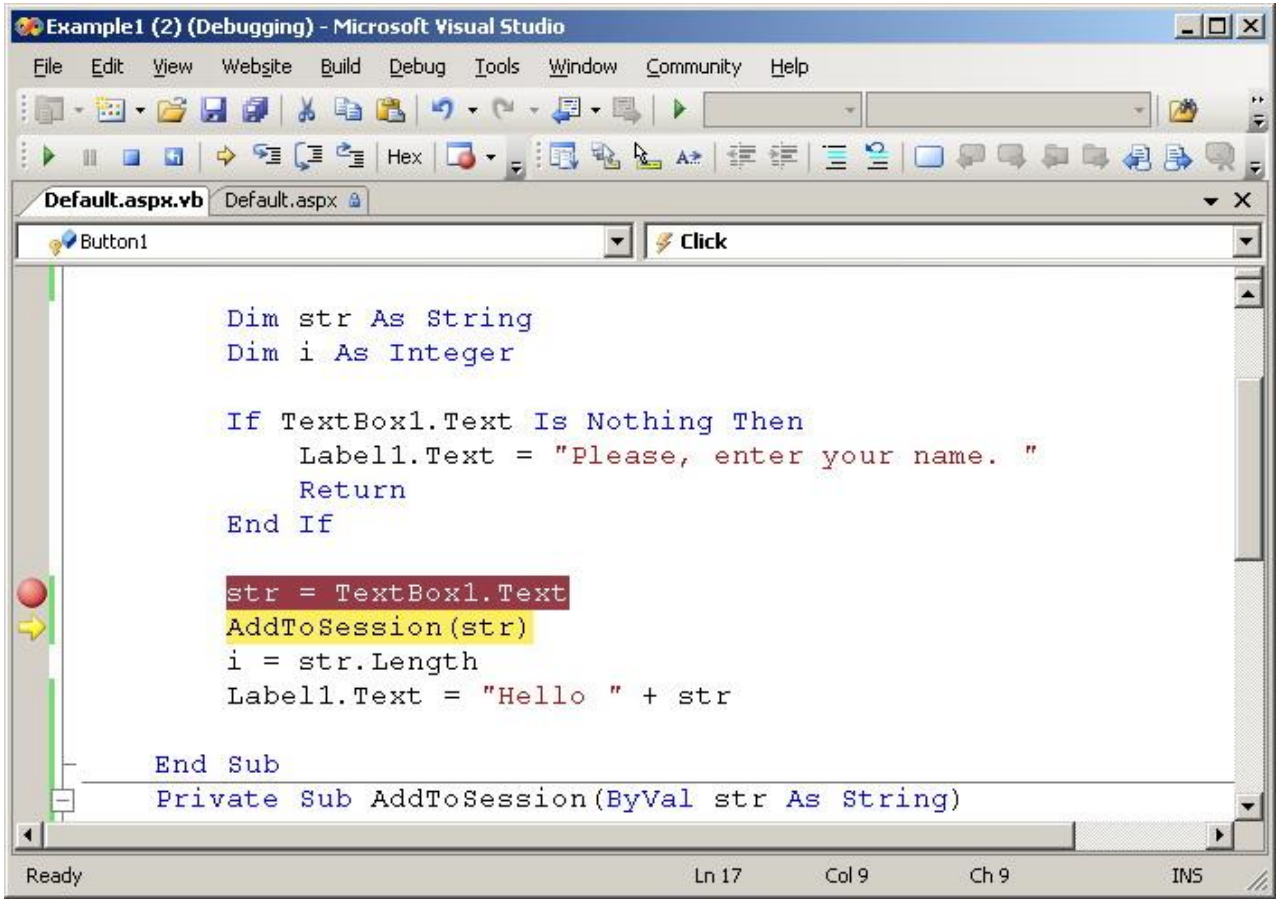
- Adding Breakpoints
- Setting Hit Count
- Setting Breakpoints Condition
- Viewing and Changing Data

Overview

- Following are options to step through code:
  - Step Into
  - Step Over
  - Step Out

Step Into

- Step through the code line by line
  - Press F11 or on the Debug menu, click Step Into.



Step Over

- Step Over differs from Step Into in the way it handles a function call
- When we use Step Over, the function is executed as a whole. No need to step through it line by line
- Press F10 or on the Debug menu, click Step Over

Step Out

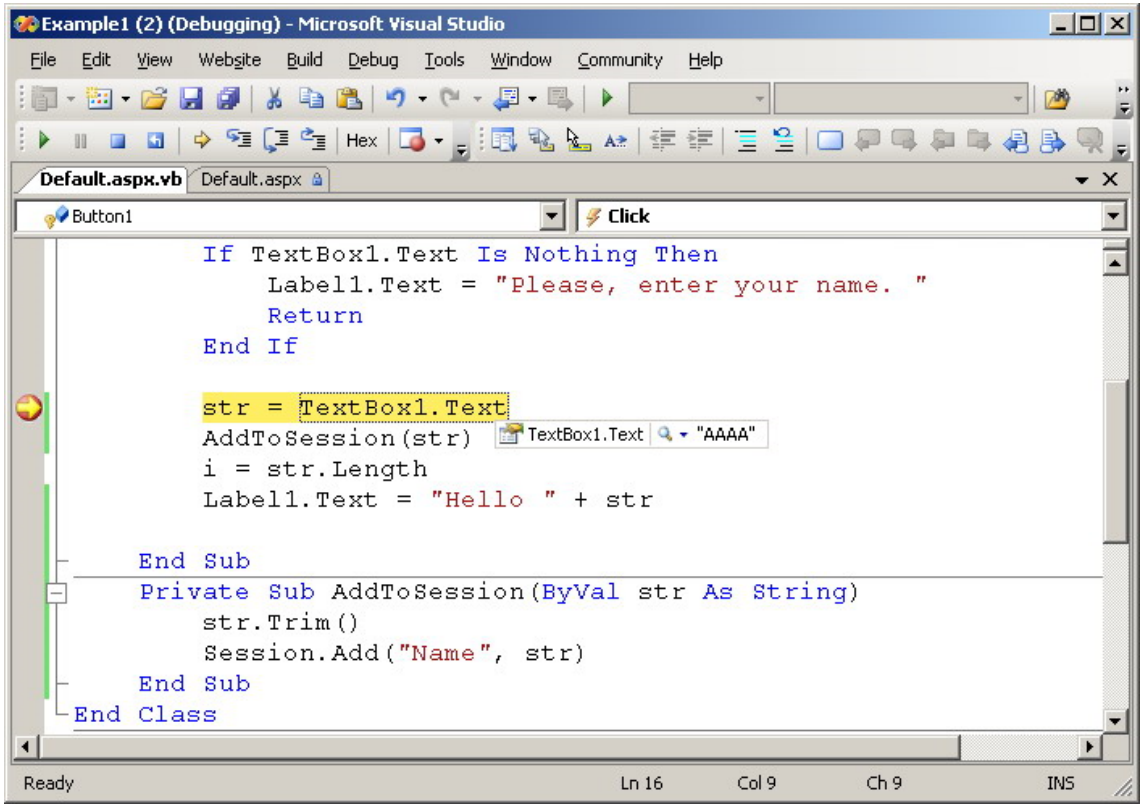
- Use Step Out when inside a function and you wish to go directly to the point from where the function is called without stepping to the end of the function
- If the control is at the top level of a program, choosing the Step Out option causes the program to resume running as normal.
- Press shifted+F11 or on the Debug menu, click Step Out

Demo: Stepping

- Demonstrating Step Into, Step Over and Step Out

Overview

- View and Track data while debugging
- Edit values in break mode



Types of Windows

- Integrated Developer provides following Windows to display and edit application's variables and expressions:
  - Locals Window:
    - Watch variables inside your current scope
    - On the Debug menu, navigate click Windows from the Locals menu to view this window
  - Autos Window:
    - View variables in the current line of code and above it.
    - On the Debug menu, navigate click Windows from the Autos menu to view this window
  - Watch Window:
    - Watch certain variables or expressions

Demo: Variables Window

- Trying to view values in Variables Window

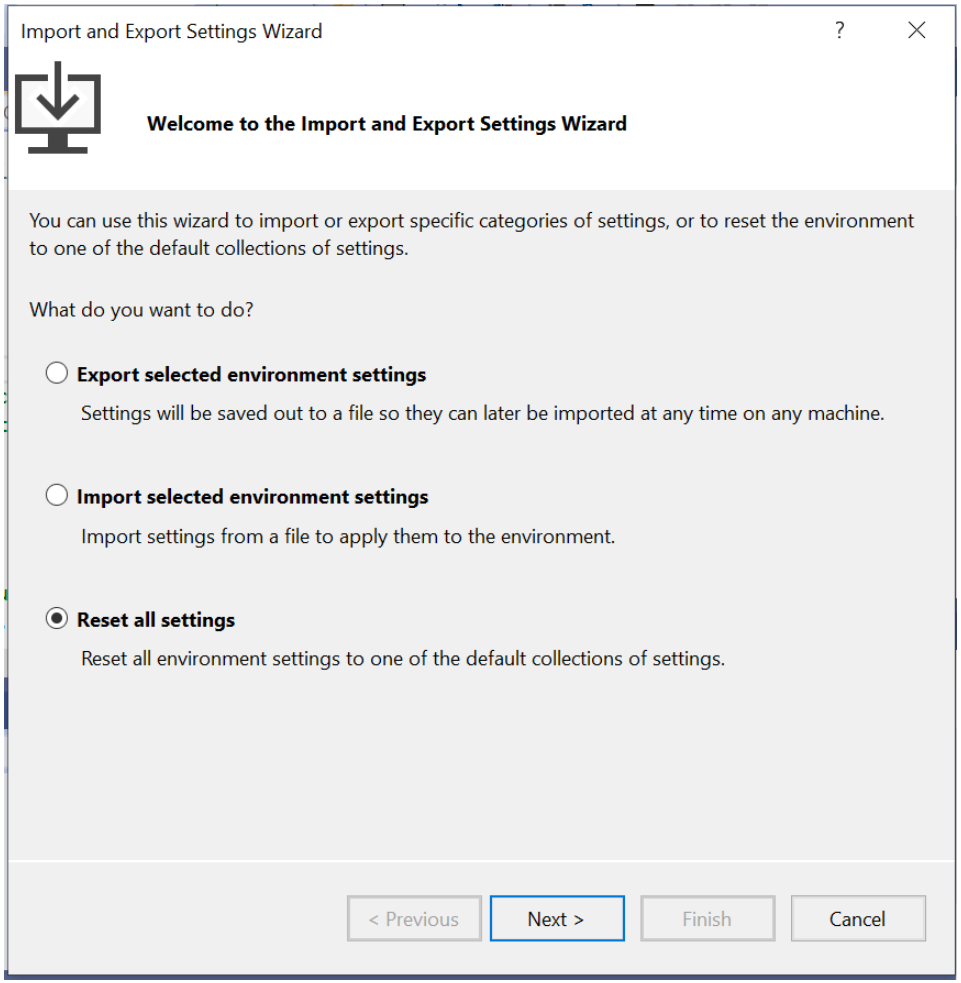
Customizing Visual Studio – Environment Settings

- When you open Visual Studio for the first time, you can optimize the development environment for the type of development that you do the most by choosing a collection of settings.
- Each collection optimizes elements such as keyboard shortcuts, window layouts, project and item templates, and command visibility.

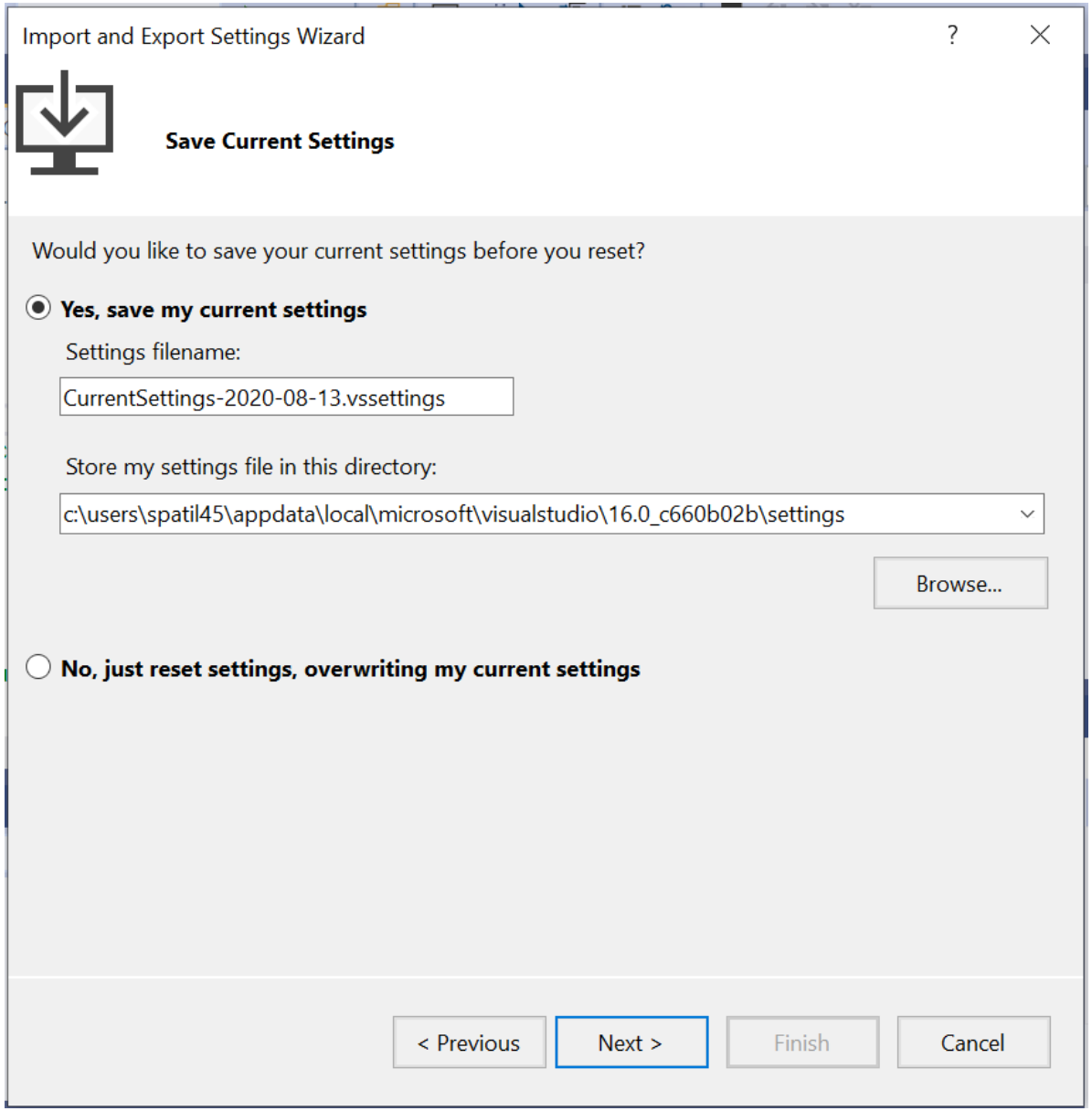
- The following settings collections are available:
  - General
  - JavaScript
  - Visual Basic
  - Visual C#
  - Visual C++
  - Web Development
  - Web Development (Code Only)

Customizing Visual Studio – Environment Settings

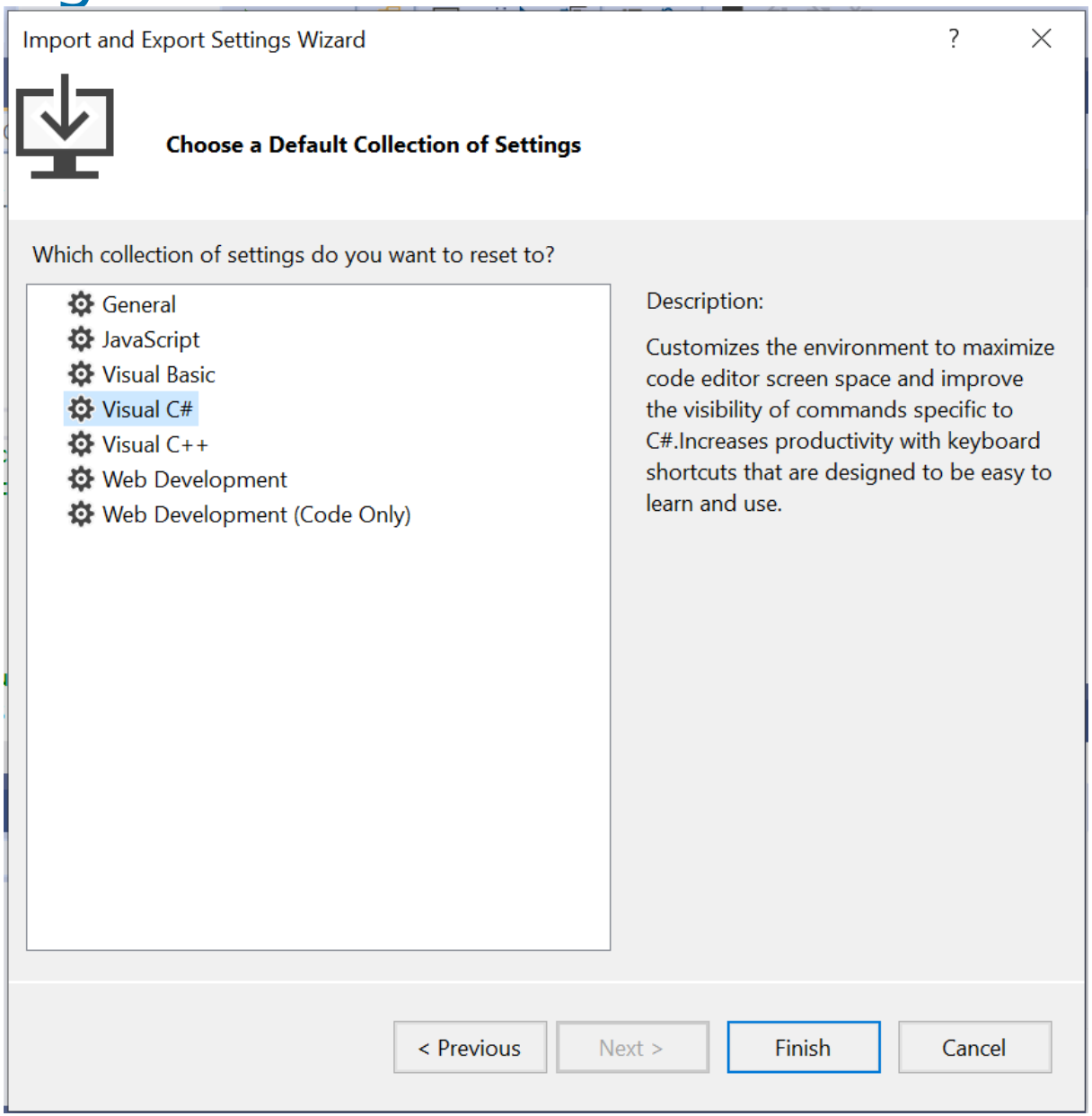
- To change your development settings after you open Visual Studio for the first time, follow these steps:
  - Select Tools > Import and Export Settings from the menu bar to open the Import and Export Settings Wizard.
  - In the Import and Export Settings Wizard, select Reset all settings, and then select Next.
  - On the Save Current Settings page, select either Yes or No, and then select Next.
  - On the Choose a Default Collection of Settings page, choose a collection, and then select Finish.



Customizing Visual Studio – Environment Settings



Customizing Visual Studio – Environment Settings

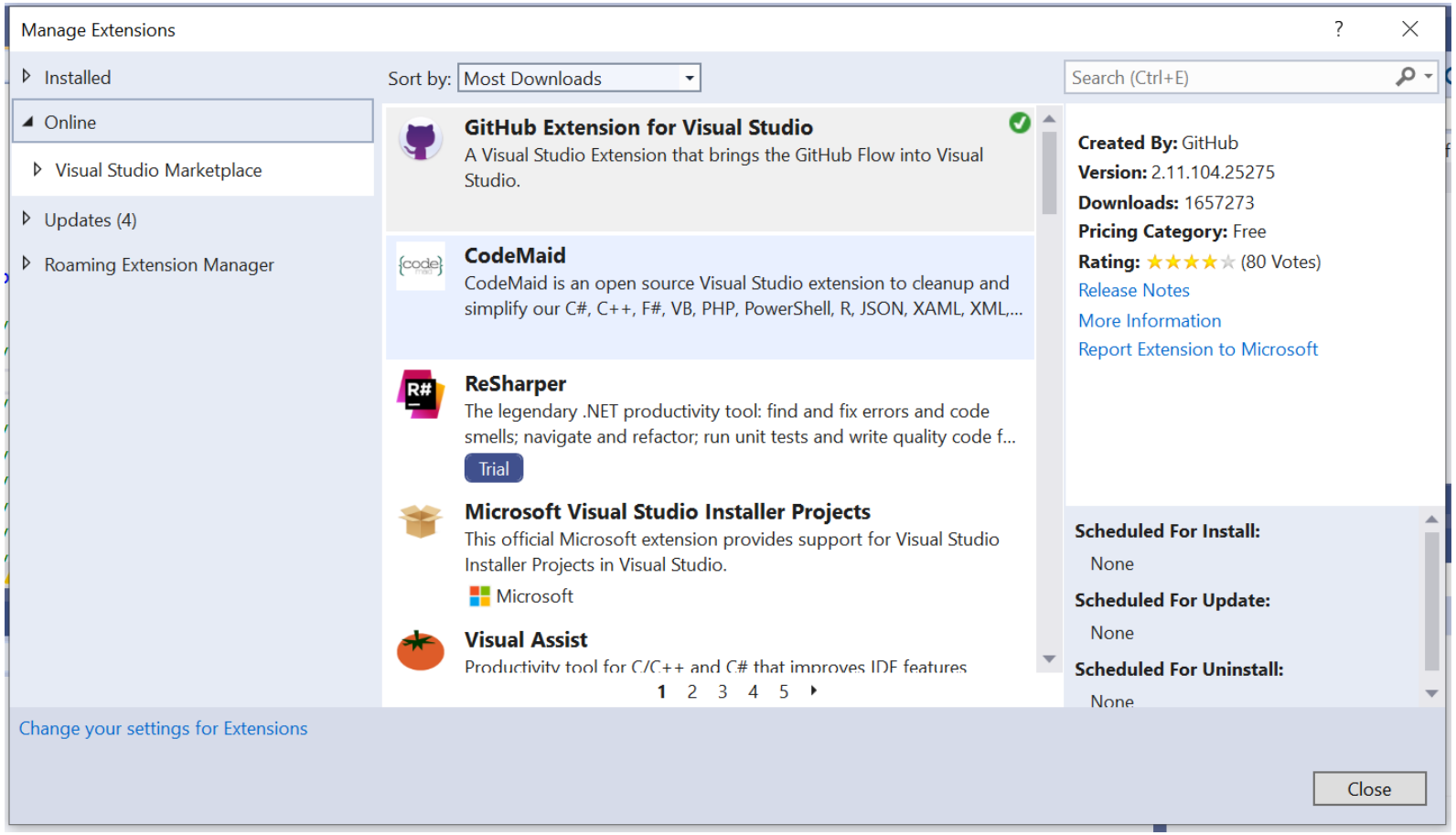


Manage Extensions for Visual Studio

- Extensions are code packages that run inside Visual Studio and provide new or improved features.
- Extensions may be controls, samples, templates, tools, or other components that add functionality to Visual Studio, for example, Live Share or Visual Studio IntelliCode
- Use the Manage Extensions dialog box to install and manage Visual Studio extensions

Manage Extensions for Visual Studio

- To open the Manage Extensions dialog, choose Extensions > Manage Extensions. Or, type Extensions in the search box and choose Manage Extensions.



Summary

- In this lesson you have learnt:
  - Debugging techniques

Review Question

- Question 1: _____ window shows the variables in the active statement and the previous code line
  - Autos
  - Locals
  - Watch
  - Immediate