# Installing Visual Studio with F#

## Lab Overview

In this lab you will configure Visual Studio 2015 for optimal F# development and then write your first F# programs.

## Prerequisites

* Visual Studio 2015 with F#

## Time Estimate

* 20 minutes

## Table of Contents

[Installing Visual Studio with F# 1](#_Toc429672397)

[Lab Overview 1](#_Toc429672398)

[Prerequisites 1](#_Toc429672399)

[Time Estimate 1](#_Toc429672400)

[Table of Contents 1](#_Toc429672401)

[Exercise 1: Configuring F# and F# Power Tools in Visual Studio 1](#_Toc429672402)

[Exercise 2: Hello, World in F#. 3](#_Toc429672403)

[Namespace.Namespace.Class.Method(arg1, arg2) 3](#_Toc429672404)

[printHello(“sally”) 3](#_Toc429672405)

[printfn “test %s” myName 3](#_Toc429672406)

[Exercise 3: Hello, World in an F# Console Application. 3](#_Toc429672407)

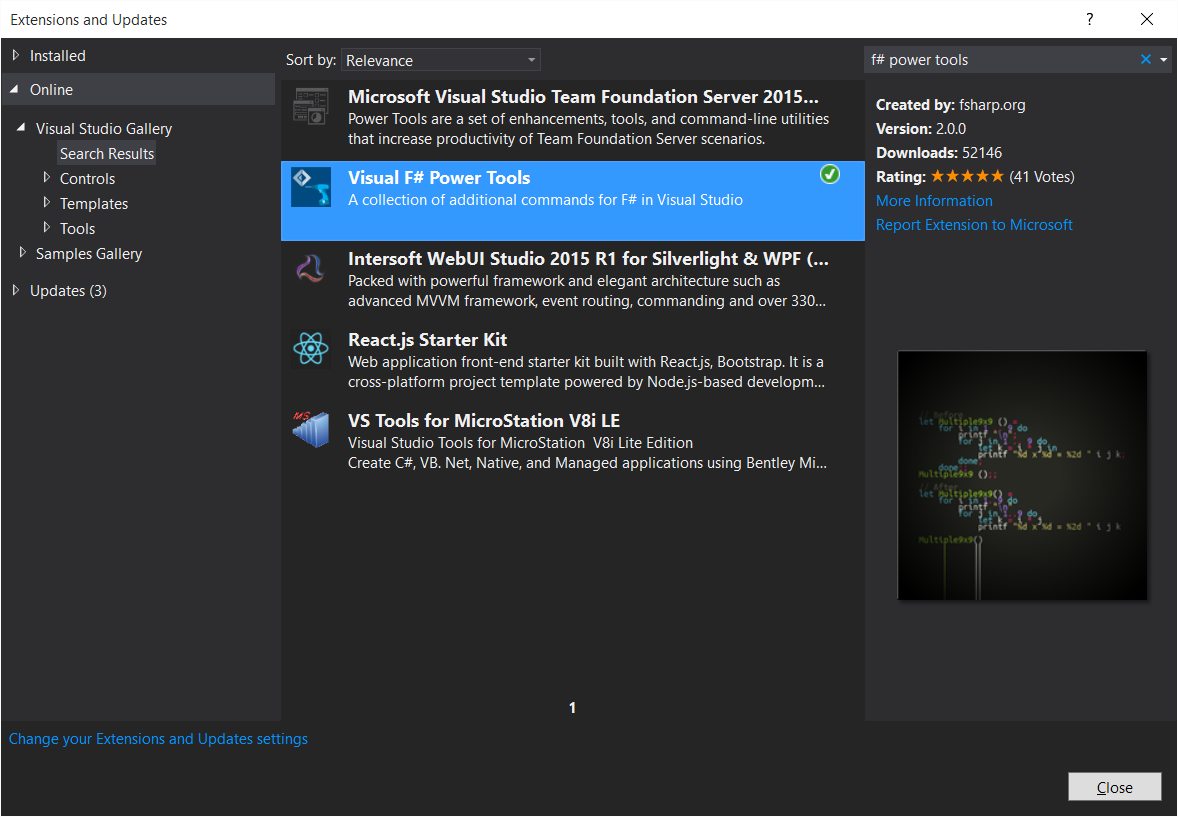
[argv.[0] 4](#_Toc429672408)

[Lab Summary 5](#_Toc429672409)

## Exercise 1: Configuring F# and F# Power Tools in Visual Studio

In this exercise you will install and configure F# Power Tools.

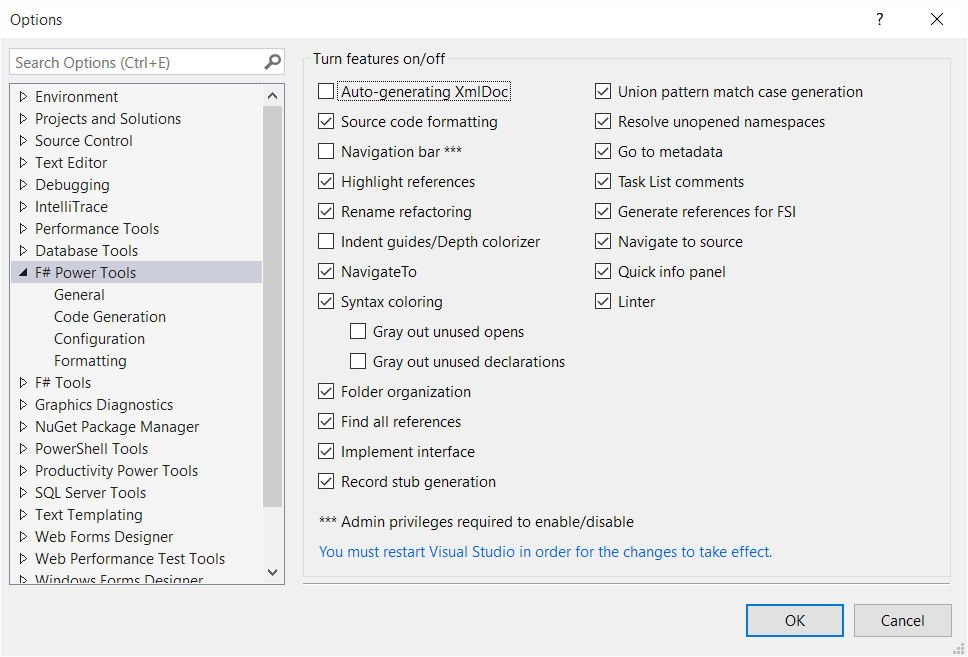
1. Open Visual Studio.
2. In the TOOLS menu, open up EXTENSIONS AND UPDATES.
3. Search for F# POWER TOOLS and install the VISUAL F# POWER TOOLS extension.



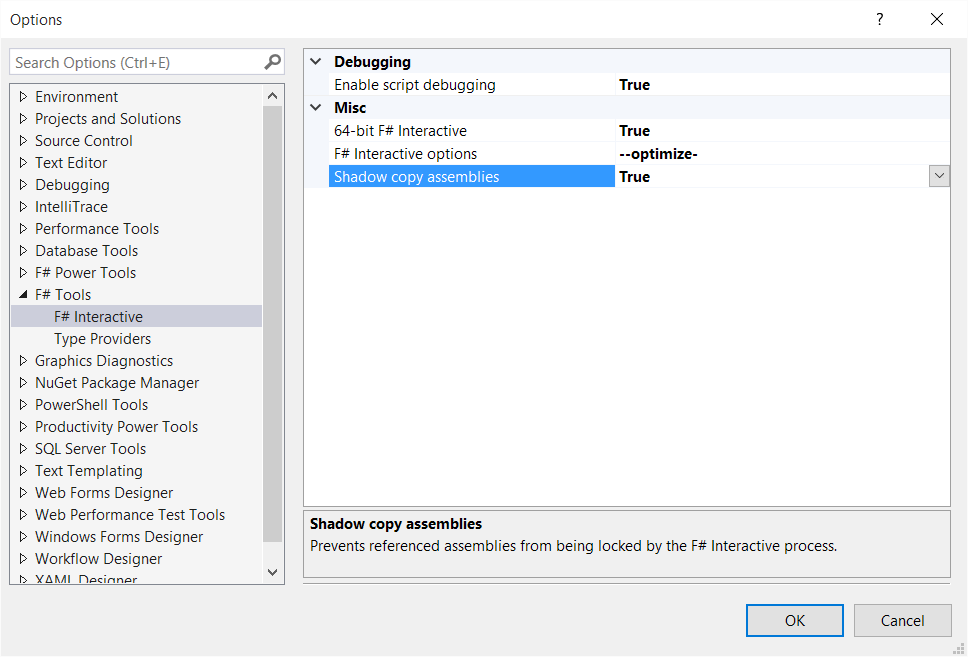
1. Restart Visual Studio.
2. Go to TOOLS -> OPTIONS -> ENVIRONMENT -> FONTS AND COLOURS. Configure the settings for F#. You can pick whichever colours you want, however if you wish to use the settings that will be used throughout the course, the table below shows the colours for all the “F#” entries in the list. Where no entry is shown in the list, simply use the default value.

|  |  |  |  |
| --- | --- | --- | --- |
| Display Item | R | G | B |
| Functions / Methods | 063 | 213 | 250 |
| Modules | 255 | 128 | 064 |
| Mutable vars / ref cells | 255 | 128 | 128 |
| Operators | 255 | 128 | 255 |
| Patterns | 255 | 255 | 128 |
| Printf Format | 128 | 128 | 192 |

1. Go to OPTIONS -> F# POWER TOOLS -> GENERAL. Ensure that the following all options are set as below (correct at the time of writing). If your machine is relatively fast, also turn on the “Gray out unused…” options.



1. Go to OPTIONS -> F# TOOLS, and set the options as follows: -



## Exercise 2: Hello, World in F#.

In this task you will create the obligatory Hello World application through a script in F#.

1. Open the HelloWorld.fsx file from within the Lab folder. You do not need a project or solution to do this.
2. Modify the “printHello” function to print out “Hello, World {your name}”. You can call standard BCL functions in F# very similarly to C#, although you do not need to worry about semi colons at the end of a line. Don’t worry too much about the function signature at this point – we’ll come onto that later.

Namespace.Namespace.Class.Method(arg1, arg2)

Once you have made the appropriate changes, highlight both lines of the function and send them to FSI using ALT + ENTER.

1. Call the function as per the example below, again using ALT + ENTER to execute the code.

printHello(“sally”)

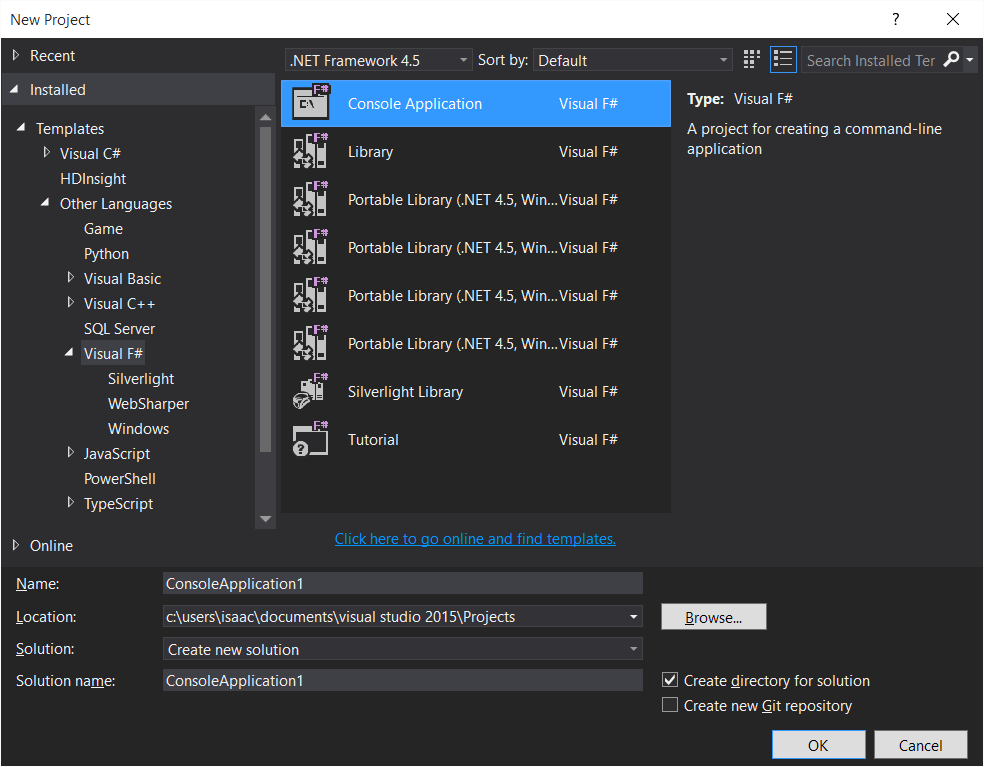
1. You should see the results output in F# Interactive.
2. **Optional!** Try removing the explicit qualified namespace to the call to Console.WriteLine and replace it with an open statement. See if you can get F# Power Tools to add it for you using the Visual Studio lightbulb helper.
3. **Optional!** Try replacing the call to **Console.WriteLine** with a call to the F#-specific **printfn** function. You can read up more on **printfn** at <https://msdn.microsoft.com/en-us/library/ee370560.aspx>, but essentially you supply placeholders to the string with type markers, such as **%s** for a string, or **%d** for a number. Unlike Console.WriteLine, this adds some type safety in that you cannot supply a number where a string is expected etc. Also, when you call printfn, you should omit any brackets e.g.

printfn “test %s” myName

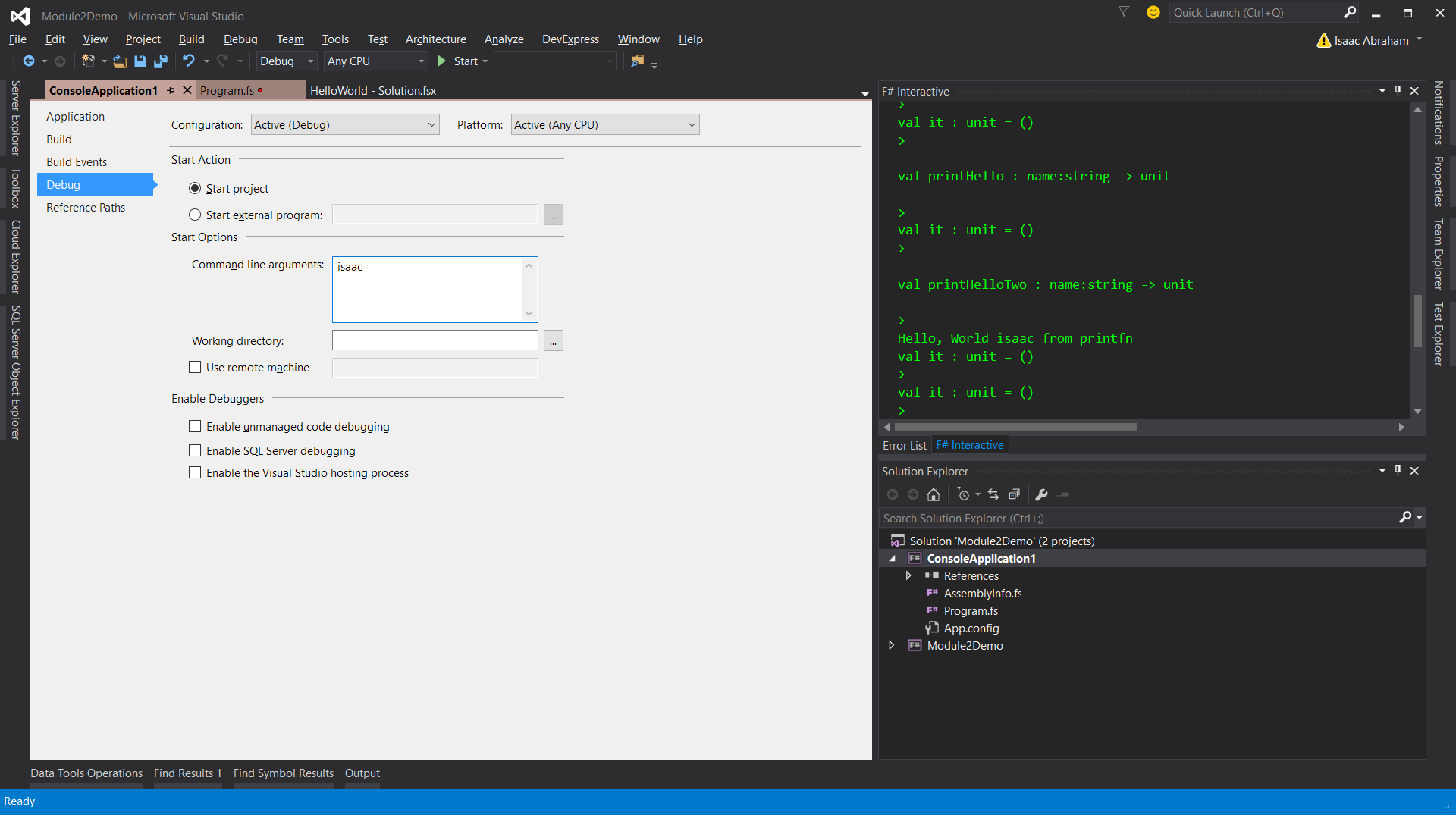
## Exercise 3: Hello, World in an F# Console Application.

Although Console Applications are not typically used as frequently as in other .NET languages due to the F# REPL, they can still be useful. This exercise will show you how to move code from a prototype script into a standalone .NET application.

1. Start by creating a new F# Console Application. If you have selected F# as the default VS Environment, this will show up underneath the Templates node. Otherwise, you’ll need to look further underneath Other Languages.



1. If the script you created in the previous lab is not open, either open it as per before, or Add it to the project.
2. In the **Program.fs** file, copy your **printHello** function above the **main** function. Ensure that both functions start on the first column of the line.
3. Modify the **main** function so that it calls your **printHello** function instead of the call to printfn, with your name supplied to the **printHello** function. Ensure that you leave the 0 at the end of the function.
4. Run the application. You should see your name printed in the Console.
5. We now will modify our application so that it passes in something on the command line. Go to the Properties page of your Console Application and navigate to the DEBUG pane. In the command line arguments, enter your name as a single word, e.g.



1. Update your code so that instead of supplying a hard coded name to your printHello function, it uses the arguments supplied to the application. This is what the **argv** variable contains – an array of strings. You can index individual items of an array in F# as follows: -

argv.[0]

Notice the “**.**” before the **[ ]**.

1. Run the application. You will see your name printed out on the Console.

## Lab Summary

In this lab you have configured F# in Visual Studio, and then created a simple script. You then created a standalone Console application, and saw how to migrate code from a prototype script to a fully compiled .NET executable.