Chapter 30: Option Infer

Section 30.1: How to enable/disable it

Document level

It is on by default, but you can set it by placing `Option Infer On|Off` at the top of the code file. The option will apply to the whole document.

Project level

You can switch it on/off via the menu in Visual Studio:

> Project > [Project] Properties > Compile Tab > Option infer

Choose `On|Off` in the drop-down menu. The option will apply to the whole document.

All new projects

You can switch it On by default for all new Projects by selecting:

> Tools > Options > Projects and Solutions > VB defaults > Option Infer

Choose `On|Off` in the drop-down menu.

Section 30.2: What is it?

Enables the use of local type inference in declaring variables.

What is type inference?

You can declare local variables without explicitly stating a data type. The compiler infers the data type of a variable from the type of its initialization expression.

Option Infer On:

```
Dim aString  = "1234" '--> Will be treated as String by the compiler
Dim aNumber  = 4711   '--> Will be treated as Integer by the compiler
```

vs. explicit type declaration:

```
'State a type explicitly
Dim aString as String = "1234"
Dim aNumber as Integer = 4711
```

Option Infer Off:
The compiler behavior with `Option Infer Off` depends on the `Option Strict` setting which is already documented here.

- **Option Infer Off - Option Strict Off**
  All variables without explicit type declarations are declared as `Object`.

```
Dim aString  = "1234" '--> Will be treated as Object by the compiler
```

- **Option Infer Off - Option Strict On**
  The compiler won´t let you declare a variable without an explicit type.

```
'Dim aString  = "1234" '--> Will not compile due to missing type in declaration
```

Section 30.3: When to use type inference

Basically you can use type inference whenever it is possible.
However, be careful when combining `Option Infer Off` and `Option Strict Off`, as this can lead to undesired run-time behavior:

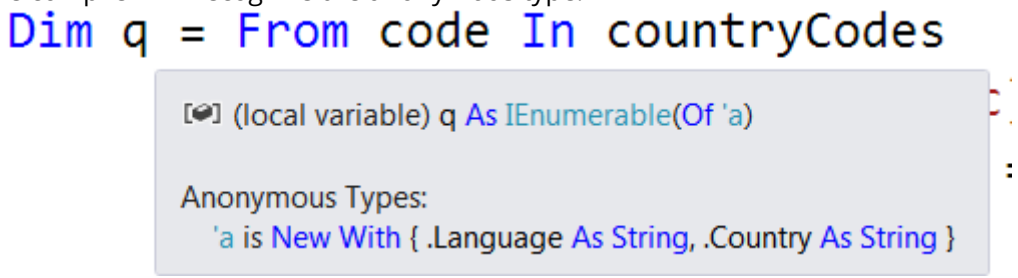
```
Dim someVar = 5
someVar.GetType.ToString() '--> System.Int32
someVar = "abc"
someVar.GetType.ToString() '--> System.String
```

Anonymous Type
Anonymous types can **only** be declared with `Option Infer On`.
They are often used when dealing with LINQ:

```
Dim countryCodes = New List(Of String)
countryCodes.Add("en-US")
countryCodes.Add("en-GB")
countryCodes.Add("de-DE")
countryCodes.Add("de-AT")

Dim q = From code In countryCodes
        Let split = code.Split("-"c)
        Select New With {.Language = split(0), .Country = split(1)}
```

- **Option Infer On**
  The compiler will recognize the anonymous type:



- **Option Infer Off**
  The compiler will either throw an error (with `Option Strict On`)
  or will consider `q` as type `object` (with `Option Strict Off`).
  Both cases will produce the outcome that you cannot use the anonymous type.

Doubles/Decimals
Numeric variables with decimal places will be infered as `Double` by default:

```
Dim aNumber = 44.11 '--> Will be treated as type `Double` by the compiler
```

If another type like `Decimal` is desired the value which initialized the variable needs to be marked:

```vb
Dim mDecimal = 47.11D '--> Will be treated as type `Decimal` by the compiler
```