



Community
Meetup
Shanghai



跟社区专家一起聊聊微服务、.NET开发、物联网和大数据



Community
Meetup
Shanghai

这还是我认识的Visual C#吗?

“放弃”：C#指南





Community
Meetup
Shanghai



放弃 - C# 指南

从 C# 7 开始，C# 支持放弃，这是一种在应用程序代码中人为取消使用的临时虚拟变量。放弃相当于未赋值的变量；它们没有值。因为只有一个放弃变量，并且甚至不为该变量分配存储空间，所以放弃可减少内存分配。因为它们使代码的意图清楚，增强了其可读性和可维护性。

通过将下划线 (`_`) 赋给一个变量作为其变量名，指示该变量为一个放弃。例如，以下方法调用返回一个 3 元组，其中第一个和第二个值为放弃：

```
(var _, _, area) = city.GetCityInformation(cityName);
```

在 C# 7 中，支持在以下上下文中的分配中使用放弃：

- 元组和对象析构。
- 使用 `is` 和 `switch` 的模式匹配。
- 对具有 `out` 参数的方法的调用。
- 当范围内没有 `_` 时，独立的 `_`。

当 `_` 是有效放弃时，尝试检索其值或在赋值操作中使用它时会生成编译器错误 CS0301：“当前上下文中不存在名称 ‘_’”。这是因为 `_` 未赋值，甚至可能未分配存储位置。如果它是一个实际变量，则不能像之前的示例那样放弃多个值。



Community
Meetup
Shanghai



讲师简介



奖励类别

Visual Studio and Development
Technologies

首次获奖年份:

2012

MVP奖励次数:

6



陈晴阳 (daxnet)

- 系统分析员 (2006)
- 微软最有价值专家 (MVP, 2012-)
- 珀金埃尔默首席软件工程师
- 专注于Visual Studio/.NET与DDD技术
- 关注前沿开发技术
- 爱生活, 爱分享



Community
Meetup
Shanghai

议程

- Visual C#的发展史
- Visual C# 7.x的新特性
- 在开发环境中使用C# 7.x
- 让CI环境支持C# 7.x新语法
- 案例演示



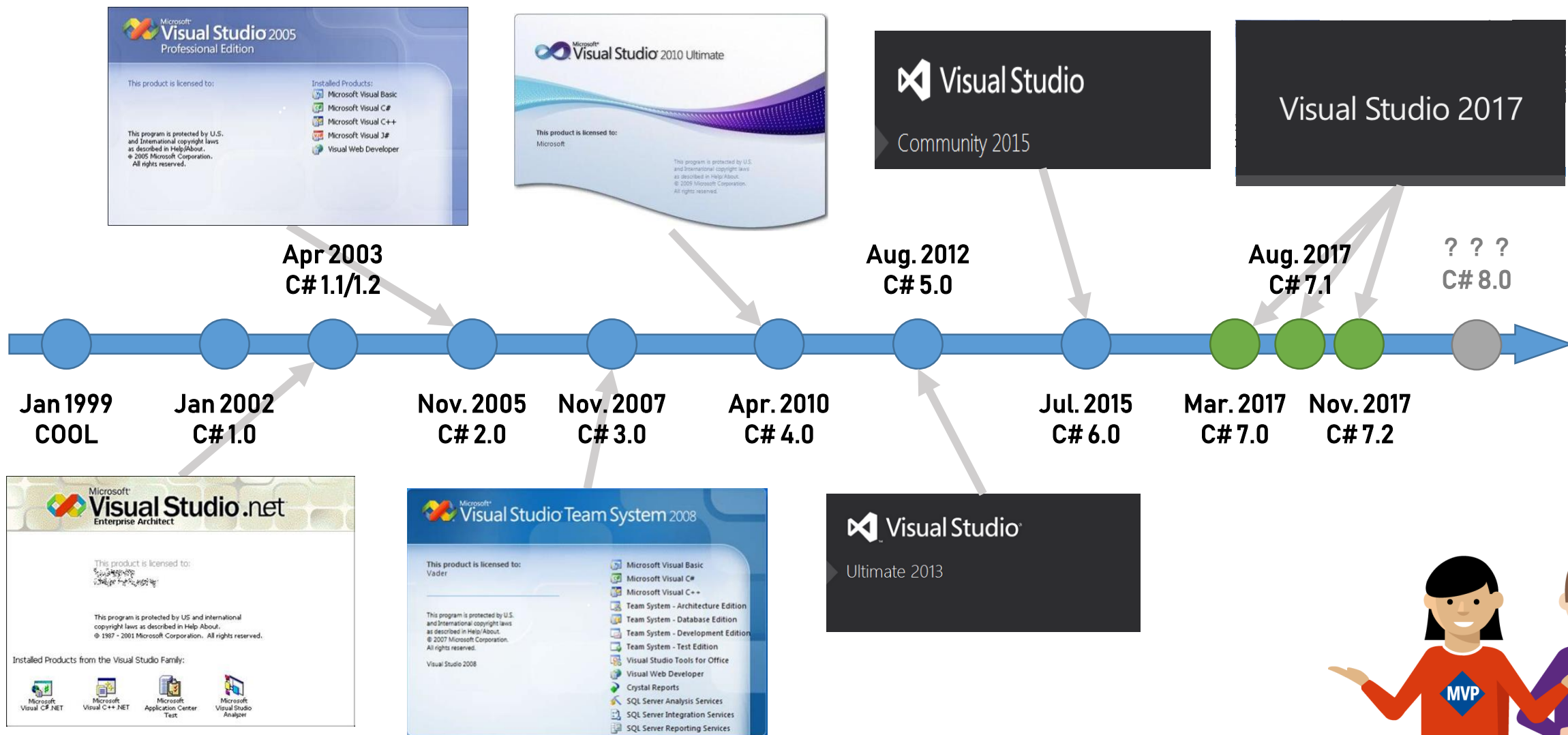


Community
Meetup
Shanghai

Visual C#的发展史



C#的发展史



C#语言新特性 (一)

C# 版本	新特性	C# 版本	新特性
C# 2.0	<ul style="list-style-type: none">泛型 (Generics)部分类 (Partial Classes)匿名方法 (Anonymous Methods)迭代器 (Iterator) yield return可空类型 (Nullable Types)Getter/Setter访问级别的分离 (Getter/setter separate accessibility)委托的协变与逆变 (Co- and Contra-variance for delegates)方法组的转换 (Method Group Conversions)静态类 (Static classes)委托类型推断 (Delegate Type Inference)	C# 3.0	<ul style="list-style-type: none">隐式类型推断 (Implicitly typed local variables)对象与集合的初始化器 (Object and collection initializers)属性的自动实现 (Auto-implemented properties)匿名类型 (Anonymous types)扩展方法 (Extension methods)查询表达式 (Query expressions)Lambda表达式 (Lambda expressions)表达式树 (Expression trees)部分方法 (Partial methods)
C# 4.0	<ul style="list-style-type: none">动态绑定 (Dynamic binding)命名参数与可选参数 (Named and optional arguments)泛型的协变与逆变 (Generic co- and contra-variance)类型等效性和嵌入的互操作类型 (Embedded interop types ("NoPIA"))	C# 5.0	<ul style="list-style-type: none">异步方法async/await (Asynchronous methods)调用者信息特性 (Caller info attributes)



C#语言新特性 (二)

C# 版本	新特性	C# 版本	新特性
C# 6.0	<ul style="list-style-type: none">• 编译器即服务 (Compiler-as-a-service (Roslyn))• 静态类型成员的命名空间导入 (Import of static type members into namespace)• 异常筛选器 (Exception filters)• catch/finally中对Await的支持 (Await in catch/finally blocks)• 自动属性初始化器 (Auto property initializers)• 只读属性的默认值支持 (Default values for getter-only properties)• 表达式主体定义成员的实现 (Expression-bodied members)• Null条件运算符 (null-conditional operator, succinct null checking)• 字符串内插 (String interpolation)• nameof操作符 (nameof operator)• 集合初始值设定项的扩展方法 (Dictionary initializer)	C# 7.0	<ul style="list-style-type: none">• Out变量 (Out variables)• 模式匹配 (Pattern matching)• 元组 (Tuples)• 解构 (Deconstruction)• 本地函数 (Local functions)• 数字分隔符 (Digit separators)• 二进制数 (Binary literals)• Ref局部变量与返回结果 (Ref returns and locals)• 通用的异步返回类型 (Generalized async return types)• 表达式主体定义构造函数与析构函数 (Expression bodied constructors and finalizers)• 表达式主体定义属性的Getter与Setter (Expression bodied getters and setters)• Throw表达式 (Throw can also be used as expression)
C# 7.1	<ul style="list-style-type: none">• 支持Async的main函数 (Async main)• default文本表达式 (Default literal expressions)• 推断元组元素名称 (Inferred tuple element names)	C# 7.2	<ul style="list-style-type: none">• 引用语义结合值类型 (Reference semantics with value types)• 非尾随命名参数 (Non-trailing named arguments)• 数值文字中的前导下划线 (Leading underscores in numeric literals)• private protected访问修饰符 (private protected access modifier)



C#与Visual Studio的关系

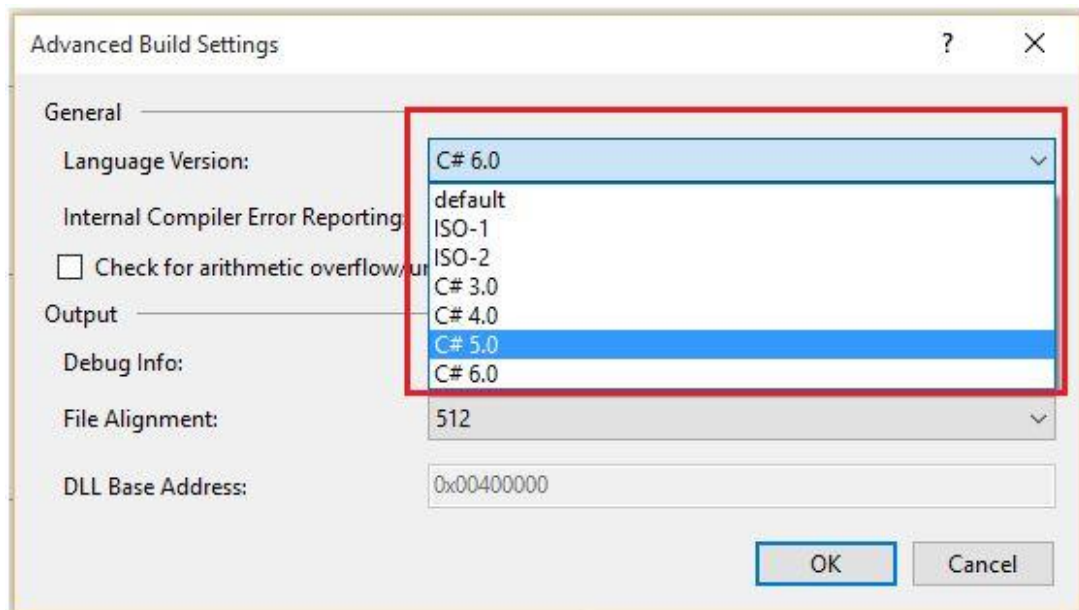


- Visual Studio 2015及以前
 - C#新版本的发布与Visual Studio同步
- Visual Studio 2017开始
 - C#新版本的发布与Visual Studio分离
 - 通过Visual Studio更新使得VS能够支持新版本的C#

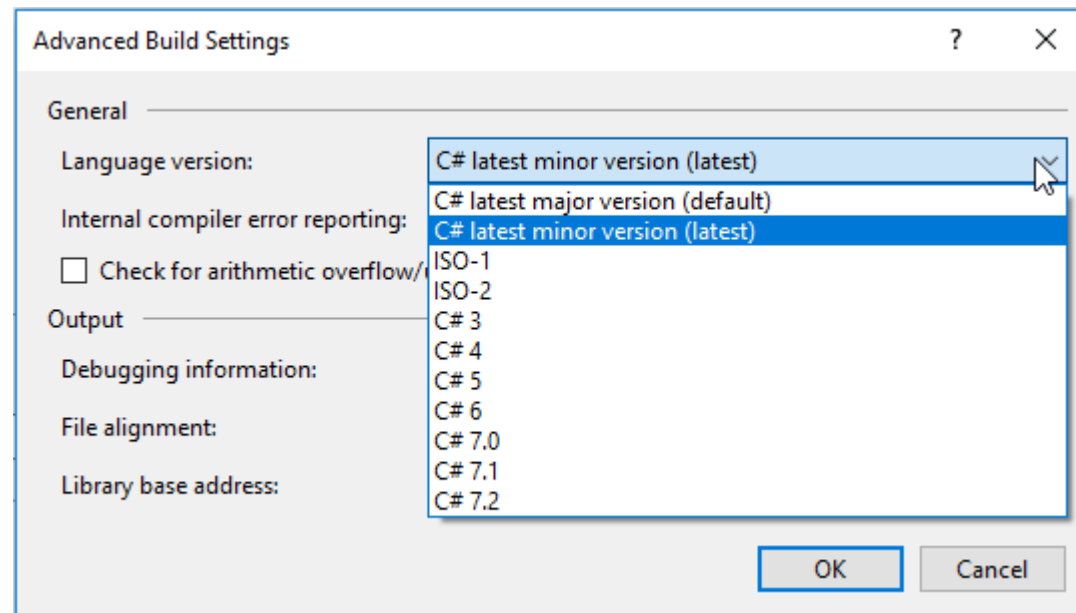


C#与Visual Studio的关系

Visual Studio 2015



Visual Studio 2017



C#与.NET Framework的关系



- 基于Roslyn的C#编译器工具集已经有了自己的发布周期，不再需要与.NET Framework同步
- 通常情况下，开发者并不会注意到这个变化
- 如需在旧版本的.NET Framework中使用新的C#语法，只需通过NuGet补装遗漏的库即可
- .NET Framework 4.0是一个分水岭





Community
Meetup
Shanghai

游览Visual C# 7.x的新特性



Visual C# 7.x的新特性



- out 变量
- 元组
- 弃元
- 模式匹配
- ref 局部变量和返回结果
- 本地函数
- 更多的 expression-bodied 成员
- throw 表达式
- 通用的异步返回类型
- 数字文本语法改进
- async Main方法
- default文本表达式
- 推断元组元素名称
- 引用语义结合值类型
- 非尾随命名参数
- 数值文字中的前导下划线
- private protected 访问修饰符



元组 (Tuples)

- .NET Framework 4.0: `Tuple<T...>`
 - 便于函数返回多个值
 - 由于是引用类型，需要在堆上分配，由GC回收
 - `Item1, Item2...`不具备可读性，让人无法理解
- C# 7.0与`System.ValueTuple`
 - 元组的表示更为简单直观
 - 语言级别直接支持多值返回函数
 - 更加轻便高效
 - 可用于泛型参数，但不能作为泛型约束类型



多值返回函数与解构函数 (Deconstruct)



```
0 references
public class Rectangle
{
    0 references
    public static class Helper
    {
        0 references
        public static void Deconstruct(this Point p,
            out int x, out int y)
        {
            x = p.X;
            y = p.Y;
        }
    }
}
```

```
0 references
static void Main(string[] args)
{
    0 references
    static void Main(string[] args)
    {
        var point = new Point(10, 20);
        var (x, y) = point;
        Console.WriteLine(x);
    }
}
```

height



弃元 (Discards)

- 函数参数或者返回值需要，但调用程序本身并不关心的临时变量
- 使用下划线作为占位符
- 弃元不具备访问属性
- 可在以下场景中使用：
 - 元组以及对象的解构
 - is和switch的模式匹配
 - 函数的out参数
 - 当_未被定义时，单独的_

```
0 references
static void Main(string[] args)
{
    var x = int.TryParse("a", out var _);
}
```

The name '_' does not exist in the current context
Show potential fixes (Ctrl+.)

```
0 references
static void Main(string[] args)
{
    _ = 10;
}
```



模式匹配 (Pattern Matching)

三种主要的模式类型:

- 常量模式 `if (p is 10) { }`
- 类型模式 `if (p is int t) { }`
- Var模式 `if (p is var t) { }`

增强的语言特性包括:

- `is` 表达式
- `switch`的`case`子句

```
switch(s)
{
    case MemoryStream ms: break;
    case FileStream fs: break;
}
```



本地函数与委托 (Local Functions)

- 本地函数是真实的函数，更为直观，也没有运行时性能开销
- 本地函数因为有名，很容易实现递归调用
- 本地函数的参数可以使用ref和out关键字

```
1 reference
private static void LocalDelegate()
{
    Func<int, int, bool> func = (x, y)
        => x >= y;

    Console.WriteLine(func(1, 2));
    // Output: False
}
```

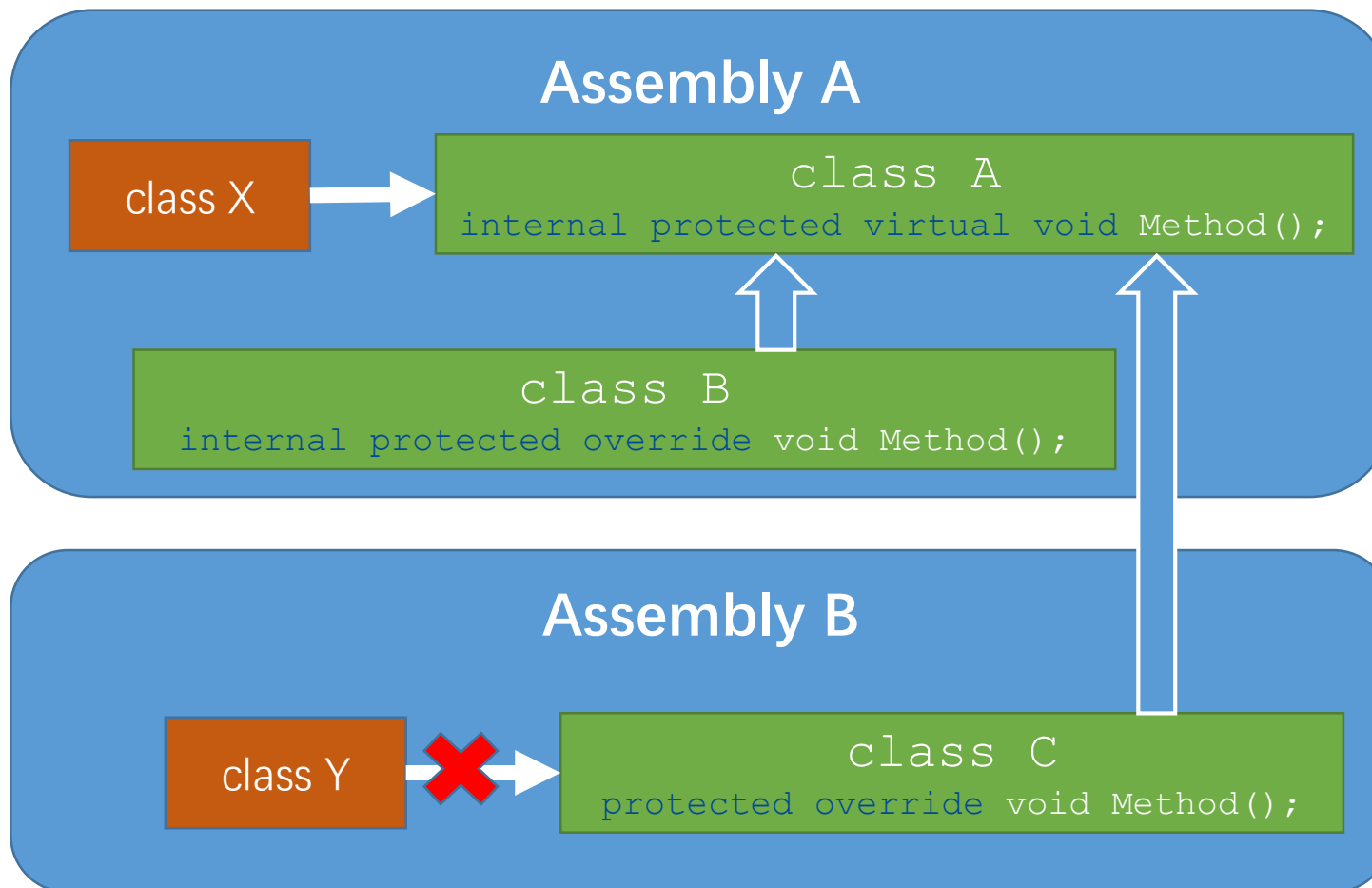
```
1 reference
private static void LocalFunction()
{
    bool Func(int x, int y, out int z)
    {
        z = x >= y ? x : y;
        return z == x;
    }

    Console.WriteLine(Func(1, 2, out var max));
    // Output: False
}
```



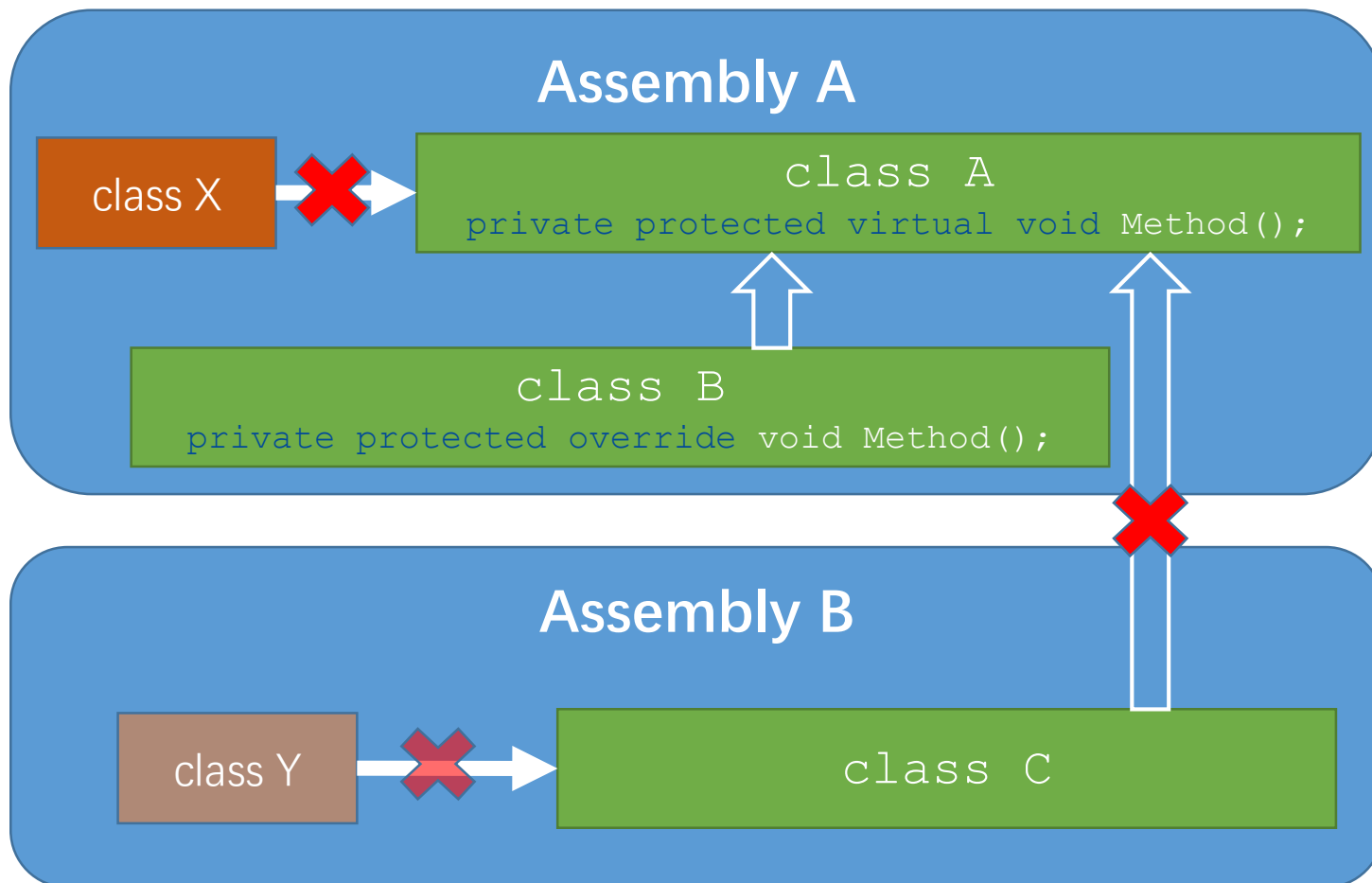
private protected访问修饰符

internal protected: internal 或者 protected



private protected访问修饰符

private protected: internal 并且 protected





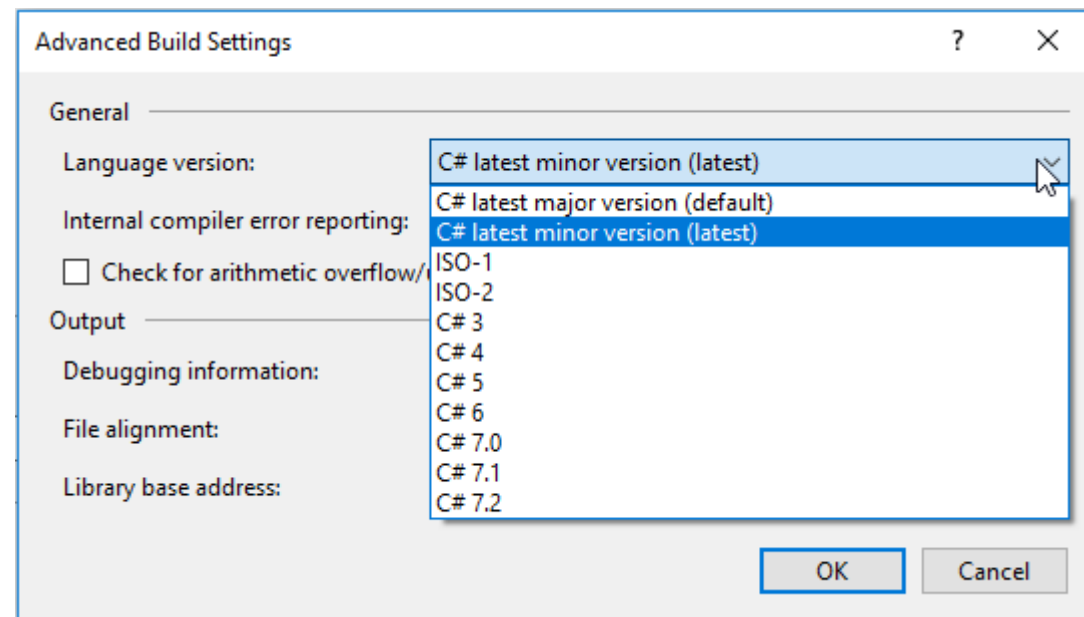
Community
Meetup
Shanghai

在开发环境中使用C# 7.x



在开发环境中使用C# 7.x

- 安装Visual Studio 2017
- 安装相应版本的VS更新
 - VS 2017: C# 7.0 (Mar. 2017)
 - Update 15.3: C# 7.1 (Aug. 2017)
 - Update 15.5: C# 7.2 (Dec. 2017)
- 在编译器选项中启用





Community
Meetup
Shanghai

让持续集成环境支持C# 7.x新语法



MSBuild发展史



- 2005年与Visual Studio 2005和.NET Framework 2.0一同上市，之后与.NET Framework一同发布
- 2013年开始，MSBuild发布跟随Visual Studio 2013，不再与.NET Framework同步，并可作为独立的安装包分发，谓之Microsoft Build Tools
- 2017年开始，改名为Visual Studio Build Tools，其中还包含了VC++ Build Tools



Visual Studio Build Tools





Community


Installing — Visual Studio Build Tools 2017 — 15.4.1

Workloads Individual components Language packs

Uncategorized (3)

 **Visual C++ build tools**
Build classic Windows-based applications using the power of the Visual C++ toolset, ATL, and optional features like... ☒

 **Web development build tools**
MSBuild tasks and targets for building web applications. ☐

 **.NET Core build tools**
Tools for building .NET Core applications. ☐

Summary

> MSBuild Tools

✓ **Visual C++ build tools**

Included

- ✓ Visual C++ Build Tools core features
- ✓ VC++ 2017 v141 toolset (x86,x64)
- ✓ Visual C++ 2017 Redistributable Update

Optional

- ☒ Windows 10 SDK (10.0.16299.0) for Desktop C++ [x...
- ☒ Visual C++ tools for CMake
- ☐ Windows 8.1 SDK and UCRT SDK
- ☐ Visual C++ ATL support
- ☐ MFC and ATL support (x86 and x64)
- ☐ C++/CLI support
- ☐ Clang/C2 (experimental)
- ☐ Modules for Standard Library (experimental)
- ☐ Windows 10 SDK (10.0.16299.0) for Desktop C++ [...]
- ☐ Windows 10 SDK (10.0.15063.0) for Desktop C++ [x...
- ☐ Windows 10 SDK (10.0.14393.0)
- ☐ Windows 10 SDK (10.0.10586.0)
- ☐ Windows 10 SDK (10.0.10240.0)
- ☒ VC++ 2015.3 v140 toolset for desktop (x86,x64)

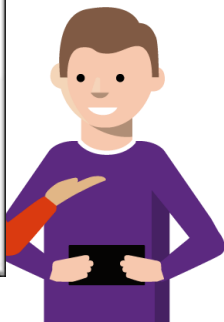
Location: C:\Program Files (x86)\Microsoft Visual Studio\2017\BuildTools

Installation nickname:

Total install size: 6.02 GB

By continuing, you agree to the [license](#) for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the [3rd Party Notices](#) or in its accompanying license. By continuing, you also agree to those licenses.

Install



持续集成环境的配置



- 编译C# 7.0以前的项目
 - 安装.NET Framework
 - 安装Microsoft Build Tools
 - 安装.NET Framework Multi-Targeting Pack
- 编译C# 7.x的项目
 - 安装Visual Studio Build Tools
 - 安装特定版本的MSBuild (15.3,15.5, 包含对应版本的C#编译器)
- 编译.NET Core项目
 - 安装.NET Core SDK



演示

集合计算器



Community
Meetup
Shanghai





Community
Meetup
Shanghai

vă mulțumesc
Salamat sa iyo
Terima kasih Cảm ơn bạn תודה עמרם
Vinaka vakalevu
Níib óolal 多謝
Dankie Gràcies
hvala Dank u धन्यवाद

Grazie Спасибо
Jamädi Gracías
Tak

Teşekkür ederiz
Merci
Danke Obrigado **diolch**
Благодаря köszönöm
Ευχαριστούμε Hvala tí
ขอบคุณ 謝謝 Dėkuju
Дякую

شكرا
ありがとう

Thank you 谢谢