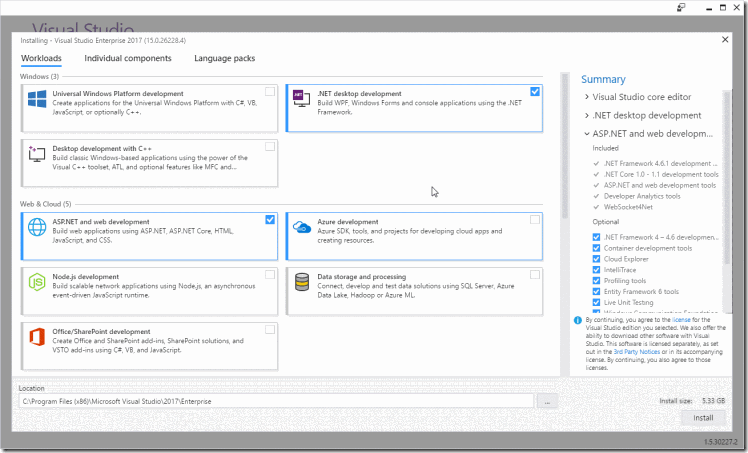
[**VS2017十五项新功能体验**](http://www.cnblogs.com/Leo_wl/p/6581647.html)

[**Visual Studio 2017十五项新功能体验**](http://www.cnblogs.com/JamesLi2015/p/6557717.html)

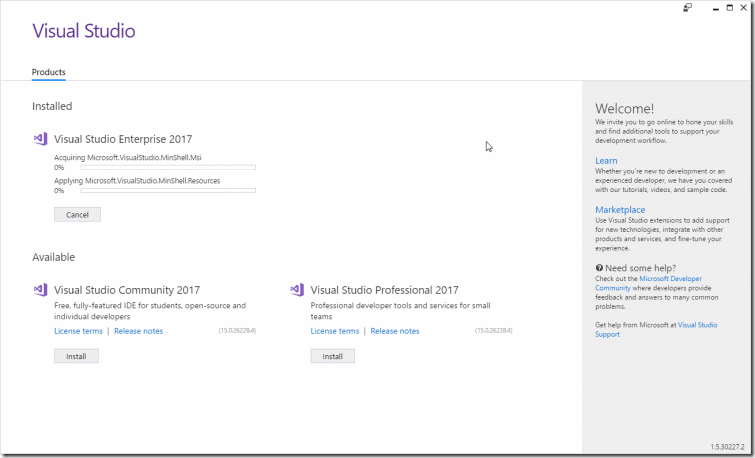
Visual Studio 2017正式已经于2017.3.7号正式发布，选在这一天发布也是为了纪念Visual Studio 二十周年。MVP 2017技术峰会将于这个周末（3.17）在北京举办，由于正在外地出差不能前往参加很是遗憾。我是从Visual Studio 6一路走过来的一线开发人员，亲历.NET （自2002年接触）开发这15年的变化。从Visual Studio 2003的ASP.NET Form到ASP.NET MVC，最后到AngularJS +Bootstrap，从Windows Forms到WPF。走技术的变更与业务领域结合的路线，一直对.NET开发很有信心。

**安装Visual Studio 2017**

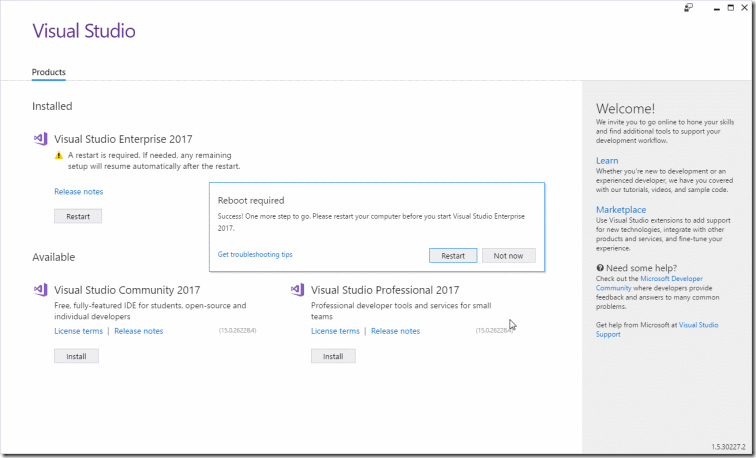
从微软网站下载安装文件visual\_studio\_enterprise\_2017\_x86\_x64.exe，只有不到1MB大小。这只是一个引导程序(Web Installer)，启动之后勾选需要的组件即可进行安装。

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083316276-870705155.gif)**

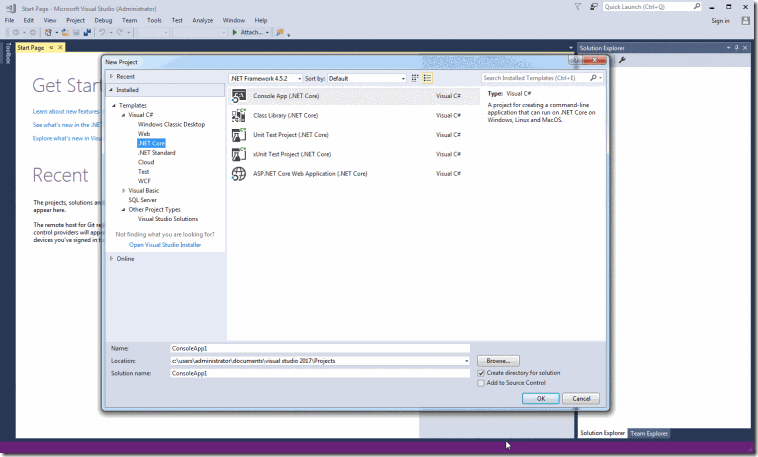
我只勾选了.NET desktop Development和ASP.NET and web development两项，右下角显示需要5.33G磁盘空间。点击Install之后开始安装。

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083317182-791955495.gif)**

我的电脑配置是i5 6400，128G固态硬盘，安装过程耗费时间约10分钟。安装完成之后需要重启一下电脑。

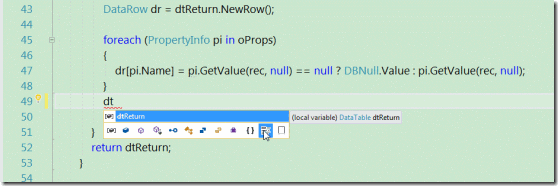
**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083317823-25901200.gif)**

启动之后如下图所示，看到期待的.NET Core和.NET Standard。J

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083318729-1290082094.gif)**

**Visual Studio 2017 新功能**

**1. Filtered IntelliSense**

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083319323-1185167323.gif)**

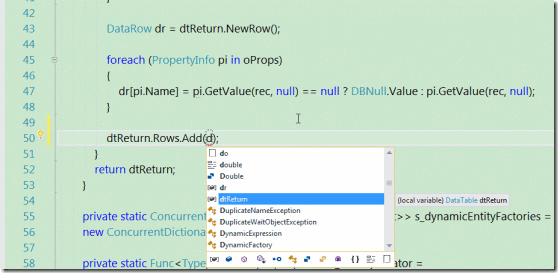
IntelliSense可以大量节省敲击代码的时间。以往的IntelliSense会根据当前字母所在的上下文自动判断，并查找合适的匹配字符供用户挑选。新版本的加入Filtered特性，下面的几点按钮依次是

* Locals and Parameters –  局部变量
* Constants –  常量
* Properties –  属性
* Fields –  字段
* Methods –  方法
* Interfaces – 接口
* Classes –  类型
* Modules –  模块
* Structures – 结构
* Enums – 枚举
* Namespaces – 命名空间
* Keywords – 关键字
* Snippets – 脚本

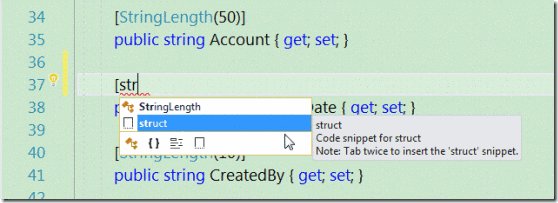
点击相应的按钮会过滤IntelliSense的结果，以便更精确的查找需要的数据。

**2. Predictive IntelliSense**

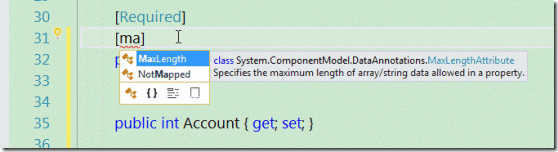
这一项功能也非常实用，参考下图

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083319948-372699145.gif)**

当我们在dtReturns.Rows.Add方法的参数中输入值时，它会智能的匹配当前上下文环境中匹配的数据。因为Rows.Add方法的参数接受数组类型或是DataRow，Predictive IntelliSense会找出当前环境中能够匹配此参数类型是的数据项。再参考一个例子：

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083320651-801768211.gif)**

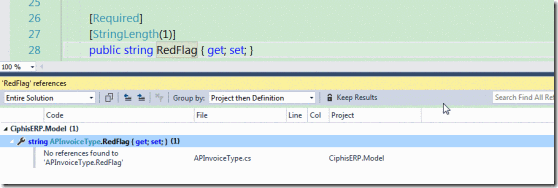
我们在属性上面输入特性(Attribute)，Predictive IntelliSense会根据当前的属性类型提示出StringLength特性。

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083321291-797754411.gif)**

当我们在在int类型的属性上添加特性时，它提示出MaxLength特性，正好可应用于当前的属性。

**3. Find All References**

这个特性在ReSharper中早已经存在，Visual Studio终于将它内置进来。

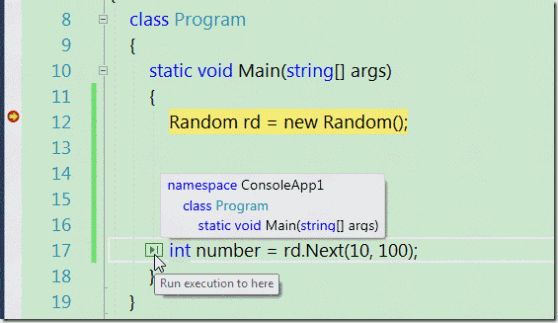
**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083321854-1585603040.gif)**

ReSharper的Find All References可以做的更深一点，对于查找所有字段或属性的引用，它将Set或Get分开显示，也就是可以看出有哪些地方是引用变量的值，

有哪些地方是设置变量的值。在大型系统调试变量的值时，这个特性会非常有用。

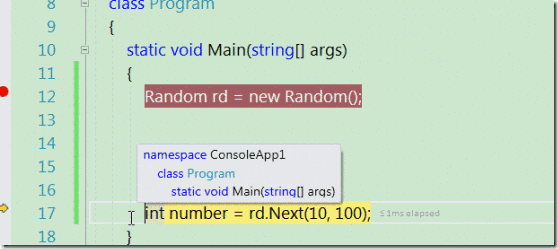
**4. Run-To-Click**

这个功能强化了以前的Run to Cursor。以前我们需要点击鼠标右键，选Run to Cursor，现在只需要在相关的行前面，点击小绿色按钮，将可实现将代码运行到此处。

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083322495-1980161745.gif)**

**5. Run-To-Click Performance Tips**

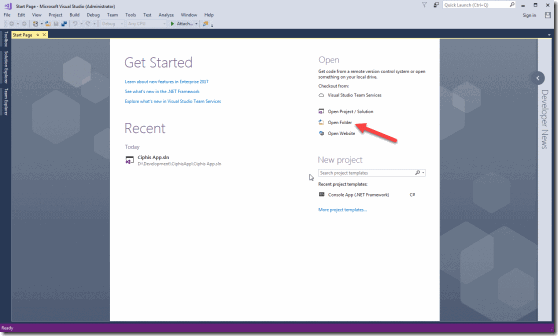
这个功能第4点属于同一组Enhancement，参考下图：

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083323213-1311412792.gif)**

当我们勾选了Run to Click之后，程序运行到当前行，在当前行的代码之后，会显示<1ms elapsed这样的字样，表示运行到这一句耗费的时间。

**6. Open Folder**

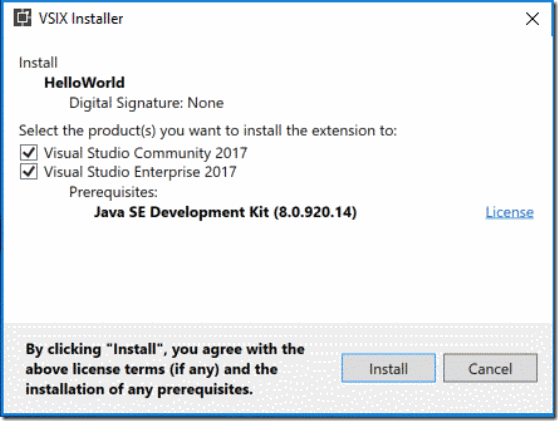
在Start Page中有一项Open Folder，可用于打开Windows Explorer，点击之后打开默认位置C:\Users\Administrator\Documents\Visual Studio 2017\Projects

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083323807-372872069.gif)**

**7. Installation Nickname**

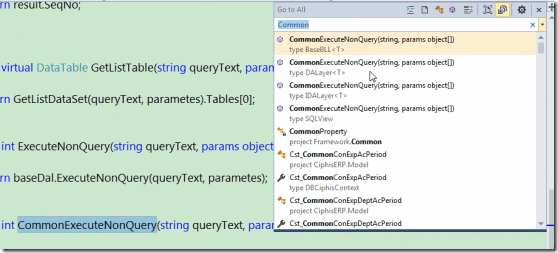
Visual Studio 2017有三个版本Enterprise,Professiona, Community，我们可以在同一台机器中同时安装这三个版本的Visual Studio。在安装时只需要指定不同的别名(Nickname)即可。

这会导致一个扩展插件，需要分别勾选要安装到的Visual Studio版本，如下图所示。

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083324713-202165417.gif)**

**8. Improved Code Navigation**

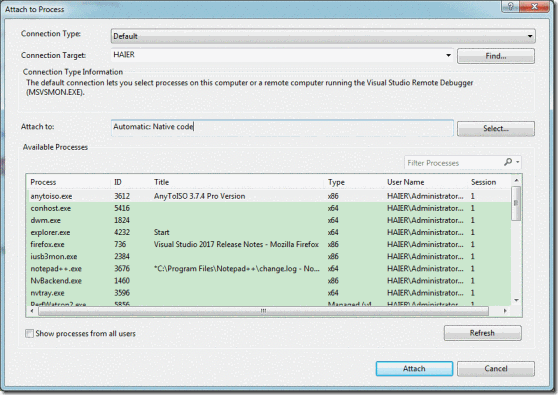
Visual Studio改善Go to All，相比之前的Go To，可以更有效率的找到需要的数据。我们在任意地方敲击Ctrl + ,

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083325432-2113883529.gif)**

Visual Studio 可以帮忙找到文件，类型，符号等匹配项，快速导航到需要的地方。

**9. Attach to Process Filter**

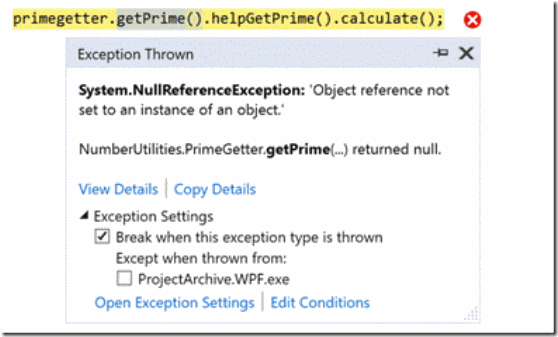
附加到进程现在支持按进程名称过滤，参考下图。

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083326213-501272098.gif)**

在可用进程(Available Processes)列表的右上角，有一个带Filter Processes水印的文本框，可以对进程名称过滤。

**10. The New Exception Helper**

经过改善的异常辅助器如下图所示。

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083327057-1868814236.gif)**

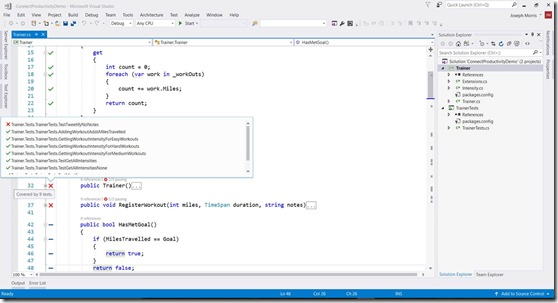
我们点击Edit Conditions可编辑中断条件。

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083327651-1122738350.jpg)**

其实还有做的更好的OzCode的Exception Helper，它可以快速帮忙定位到哪里抛出异常(Throw Exception)，哪里处理异常(Handle Exception)。在大量系统中查找数据异常非常方便。

**11. Live Unit Testing**

当跑一次单元测试之后，Visual Studio可以显示代码覆盖率，通过代码行前面的绿色勾或红色叉，可以区别分运行过的代码路径与没有跑的代码路径。

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083328870-481498830.jpg)**

这个功能也不算新鲜，dotCover多年以前就具备这个功能。

**12. Redgate Data Tools Included with Visual Studio**

Redgate的SQL 工具直接集成到Visual Studio中以辅助快速编写SQL语句和管理与部署数据库。它们分别是

· [**Redgate ReadyRoll Core**](http://www.red-gate.com/products/sql-development/readyroll/entrypage/microsoft-and-readyroll?utm_source=microsoft&utm_medium=link&utm_campaign=readyroll&utm_term=relnotes0317), 数据库升级与自动化部署

· [**Redgate SQL Prompt Core**](http://www.red-gate.com/products/sql-development/sql-prompt/entrypage/microsoft-and-sql-prompt?utm_source=microsoft&utm_medium=link&utm_campaign=sqlprompt&utm_term=relnotes0317), SQL语句智能提示

· [**Redgate SQL Search**](http://www.red-gate.com/products/sql-development/sql-search/?utm_source=microsoft&utm_medium=link&utm_campaign=sqlsearch&utm_term=relnotes0317), 数据库搜索

Redgate 开发了大量的.NET开发人员必备的工具，如.NET Reflector,SQL Prompt,dotPerformance。如今直接集成到Visual Studio中变成官方支持版本，可见其相当完善，市场占有率高。

当年Visual Studio集成了水晶报表，让Crystal Report成为报表制作的工业标准。微软之后开发出自己的SQL Server Reporting Services，Crystal Report便不再直接集成在Visual Studio安装包中。

**13. TypeScript 2.1**

项目中用到AngularJS，JavaScript也是由TypeScript生成。TypeScript是一种设计时语言，最终是为了生成JavaScript，它有强类型语言的特点，配合Visual Studio开发工具，项目中JavaScript代码的可维护性和质量大大提升。

Array 数组类型

|  |  |
| --- | --- |
| TypeScript | JavaScript |
| var array: string[] = ['test', 'dummy'];  var first: string = array[0]; | var array = ['test', 'dummy'];  var first = array[0]; |

Primitive 基本类型

|  |  |
| --- | --- |
| **TypeScript** | var name = { firstName: 'Homer', lastName: 'Simpson' };  name.firstName = 2; //This gives compile time error |
| **JavaScript** | var name = { firstName: 'Homer', lastName: 'Simpson' };  name.firstName = 2; //No Error in JavaScript |

Type Inference类型决断

|  |  |
| --- | --- |
| **TypeScript** | var dummy; //any type  var num = 10; //number  var str = 'Hello TypeScript'; //string  var bln = true; //boolean  var stringArray = ['Homer', 'Simpson']; //string[] |
| **JavaScript** | var dummy;  var num = 10;  var str = 'Hello TypeScript';  var bln = true;  var stringArray = ['Homer', 'Simpson']; |

|  |  |
| --- | --- |
| **TypeScript** | var addFunction = function (n1: number, n2: number, n3: number) {  var sum = n1 + n2 + n3;  return sum;  };  var str1: string = addFunction(10, 20, 30); //Gives compile time error as return type of a function is number and is being assigned to a string  var sum: number = addFunction(10, 20, 30); // This works  var result = addFunction(10, 20, 30); // This also works |
| **JavaScript** | var addFunction = function (n1, n2, n3) {  var sum = n1 + n2 + n3;  return sum;  };  var str1 = addFunction(10, 20, 30);  var sum = addFunction(10, 20, 30);  var result = addFunction(10, 20, 30); |

Optional Type 可选参数

|  |  |
| --- | --- |
| **TypeScript** | var addFunction = function (n1: number, n2: number, n3?: number) : number {  var sum = n1 + n2 + n3;  return sum;  };  var sum: number = addFunction(10, 20); |
| **JavaScript** | var addFunction = function (n1, n2, n3) {  var sum = n1 + n2 + n3;  return sum;  };  var sum = addFunction(10, 20); |

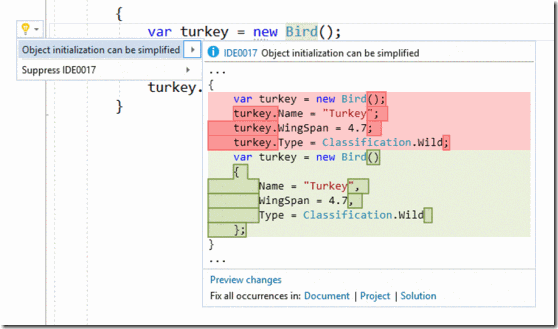
Classes and Interface 类和接口

|  |  |
| --- | --- |
| **TypeScript** | class Student {  private firstName: string;  private lastName: string;  yearOfBirth: number; //Public scope by default  schoolName: string;  city: string;  //Constructor  constructor(firstName: string, lastName: string, schoolName: string, city: string, yearOfBirth: number) {  this.firstName = firstName;  this.lastName = lastName;  this.yearOfBirth = yearOfBirth;  this.city = city;  this.schoolName = schoolName;  }  //Function  age() {  return 2014 - this.yearOfBirth;  }  //Function  printStudentFullName(): void {  alert(this.lastName + ',' + this.firstName);  }  } |
| **JavaScript** | var Student = (function () {  //Constructor  function Student(firstName, lastName, schoolName, city, yearOfBirth) {  this.firstName = firstName;  this.lastName = lastName;  this.yearOfBirth = yearOfBirth;  this.city = city;  this.schoolName = schoolName;  }  //Function  Student.prototype.age = function () {  return 2014 - this.yearOfBirth;  };  //Function  Student.prototype.printStudentFullName = function () {  alert(this.lastName + ',' + this.firstName);  };  return Student;  })(); |

代码来自于TypeScript - 101 : The Basics，地址是[**https://www.codeproject.com/Articles/802722/TypeScript-The-Basics**](https://www.codeproject.com/Articles/802722/TypeScript-The-Basics)

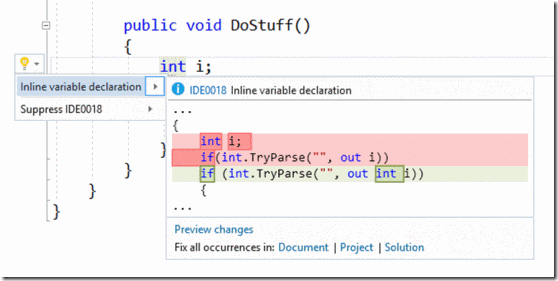
**14. Quick Actions and Refactorings**

代码重构。这项功能早就集成在Resharper中。

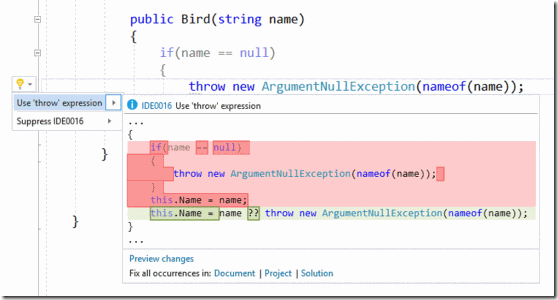
**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083329807-1395144984.gif)**

这一功能可以将对象的赋值语句块转化为对象初始化写法。

out内联

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083330948-1072086803.gif)**

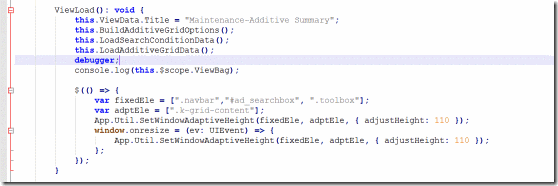
空值检查

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083332245-2064972988.gif)**

**15. Chrome Debugging Support**

Visual Studio可以调试运行在Chrome中的JavaScript脚本。也可以调试由TypeScript生成的JavaScript脚本。在Chrome中启动调试时，只需要在相应的脚本文件中放入

Debugger中断，再启动Chrome的调试模式，Chrome会在代码执行到debugger时中断，此时可单步调用TypeScript文件。

**[](http://images2015.cnblogs.com/blog/17042/201703/17042-20170316083332963-2020574608.gif)**