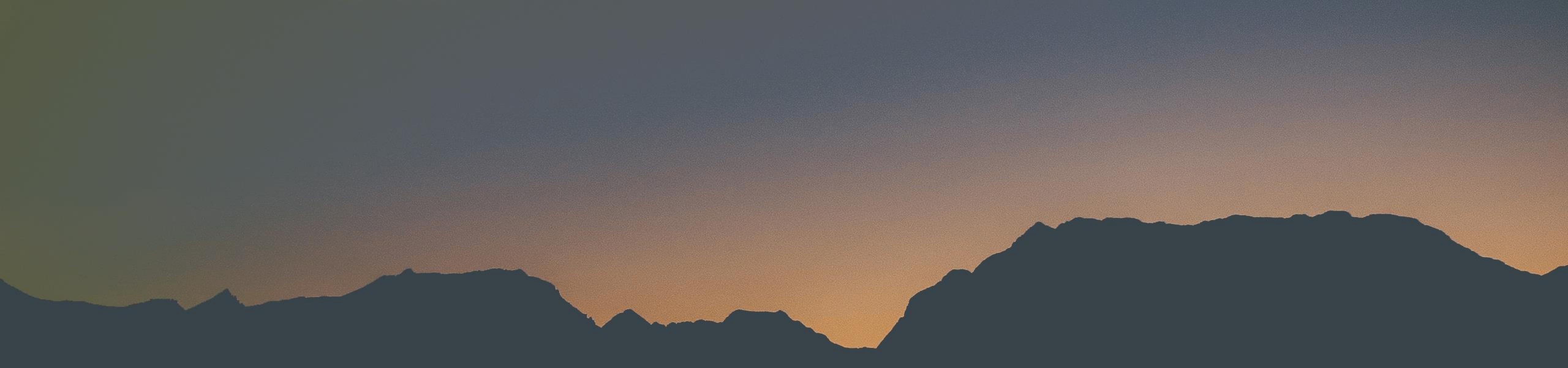


Unit Testing

With a focus on practical topics

Student: Aryan Ebrahimpour

Professor: Saeed Zahedi

Microsoft

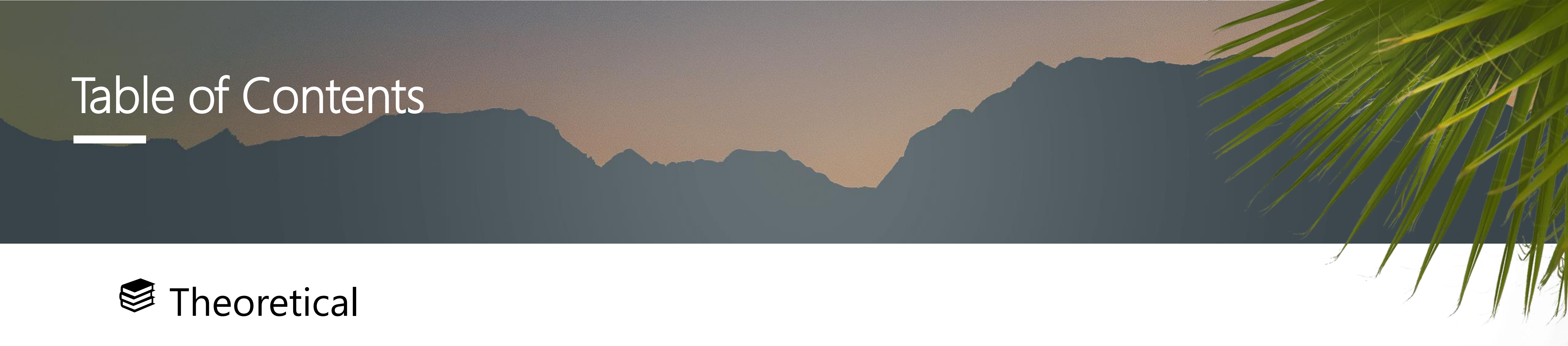
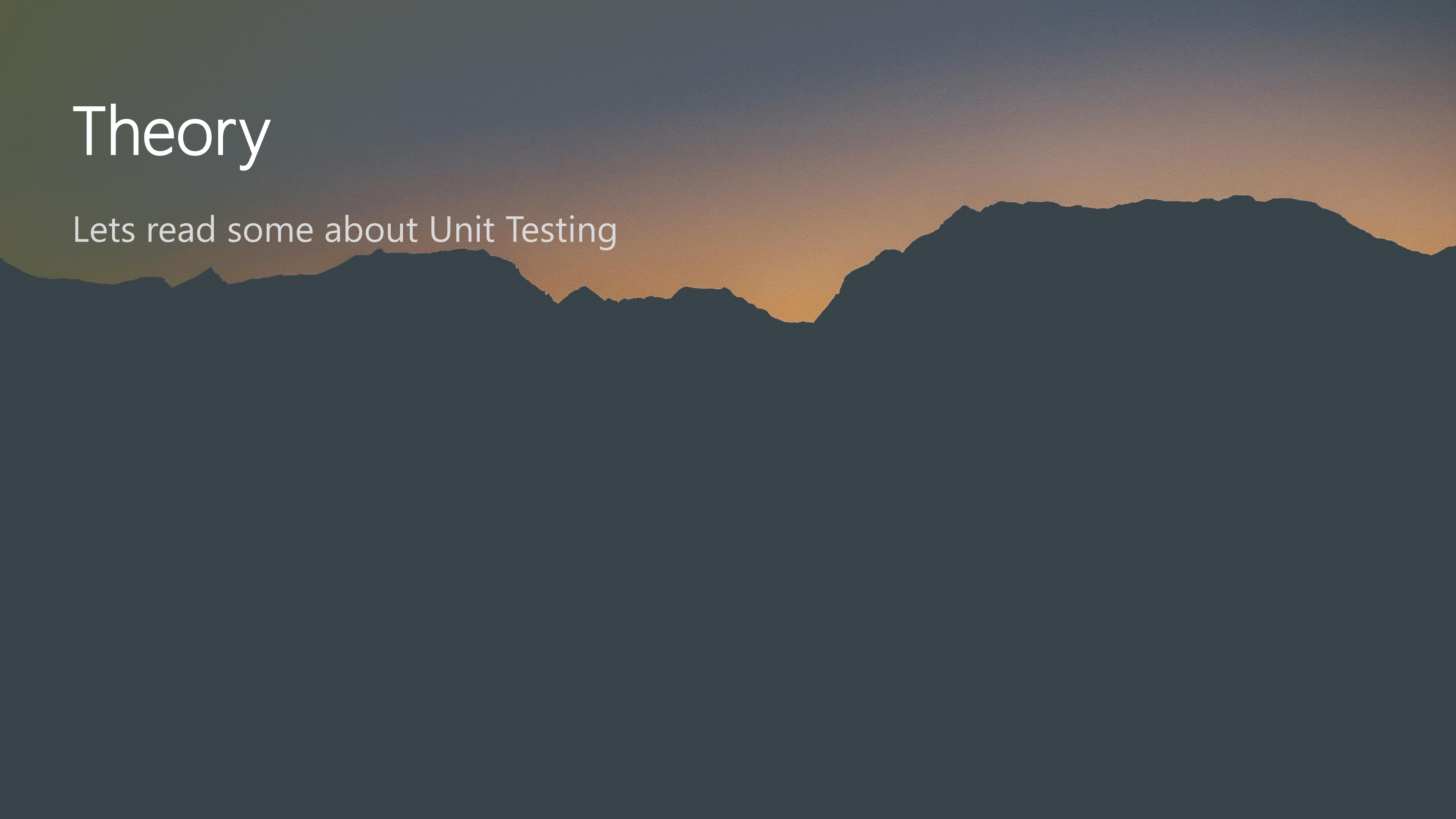


Table of Contents

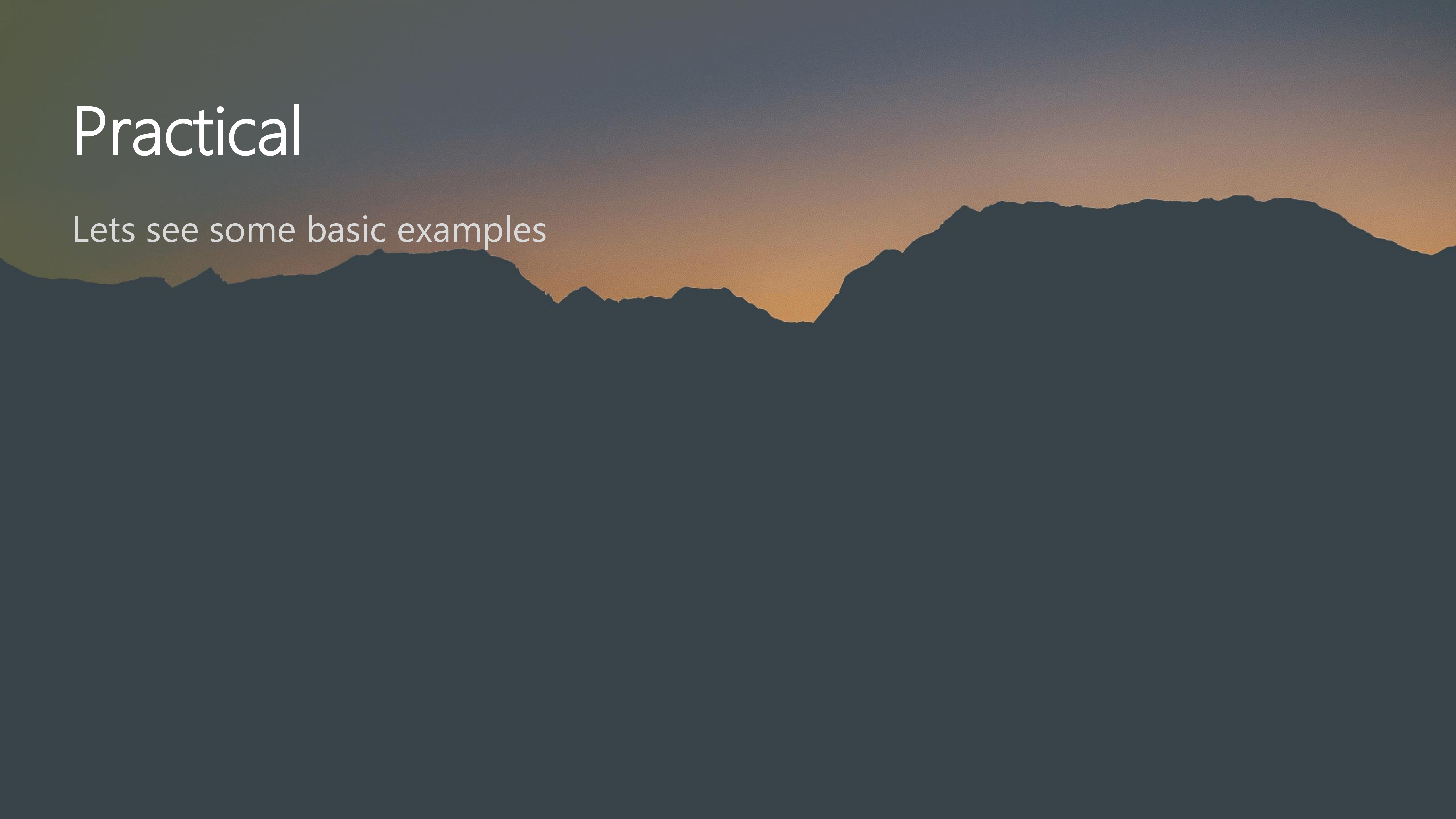


Theory

Lets read some about Unit Testing

A Basic Intro to Unit Testing

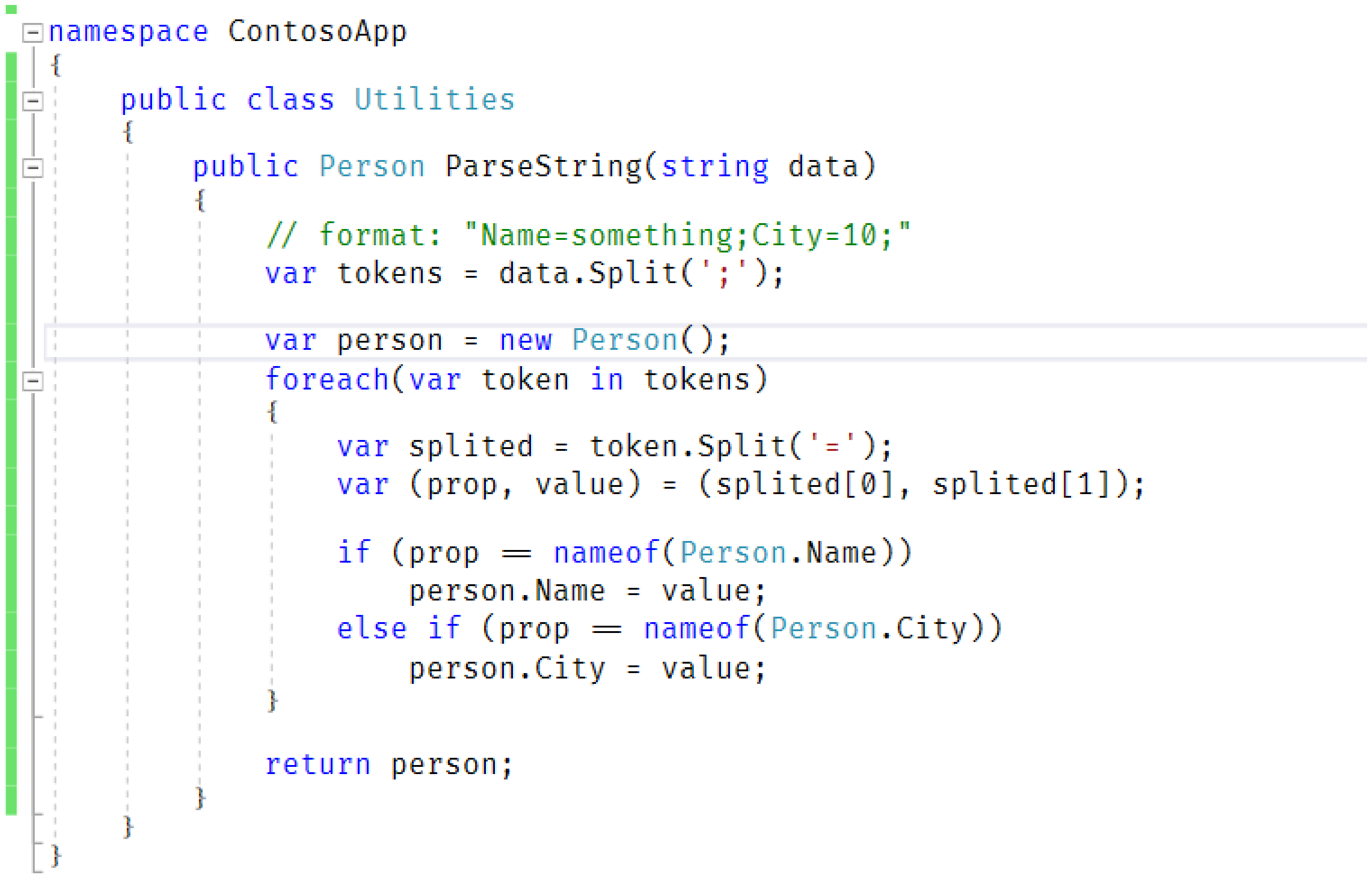
- White box
- Individual modules are tested
- Checks if there are any issues by the **Developer** himself
- Isolate each unit of System
- Purpose: Identify, Analyze and Fix the defects



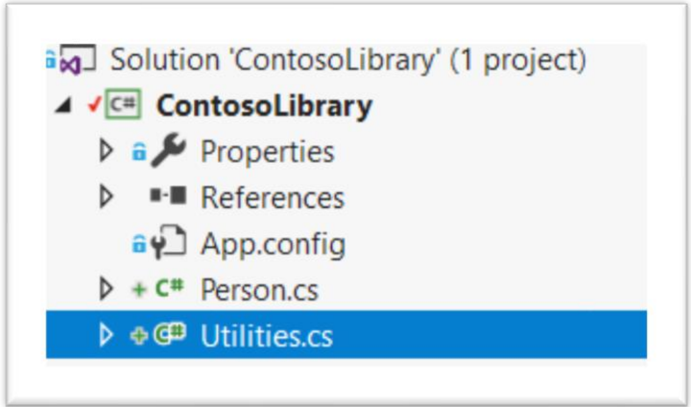
Practical

Lets see some basic examples

Basic Examples (1), The Code

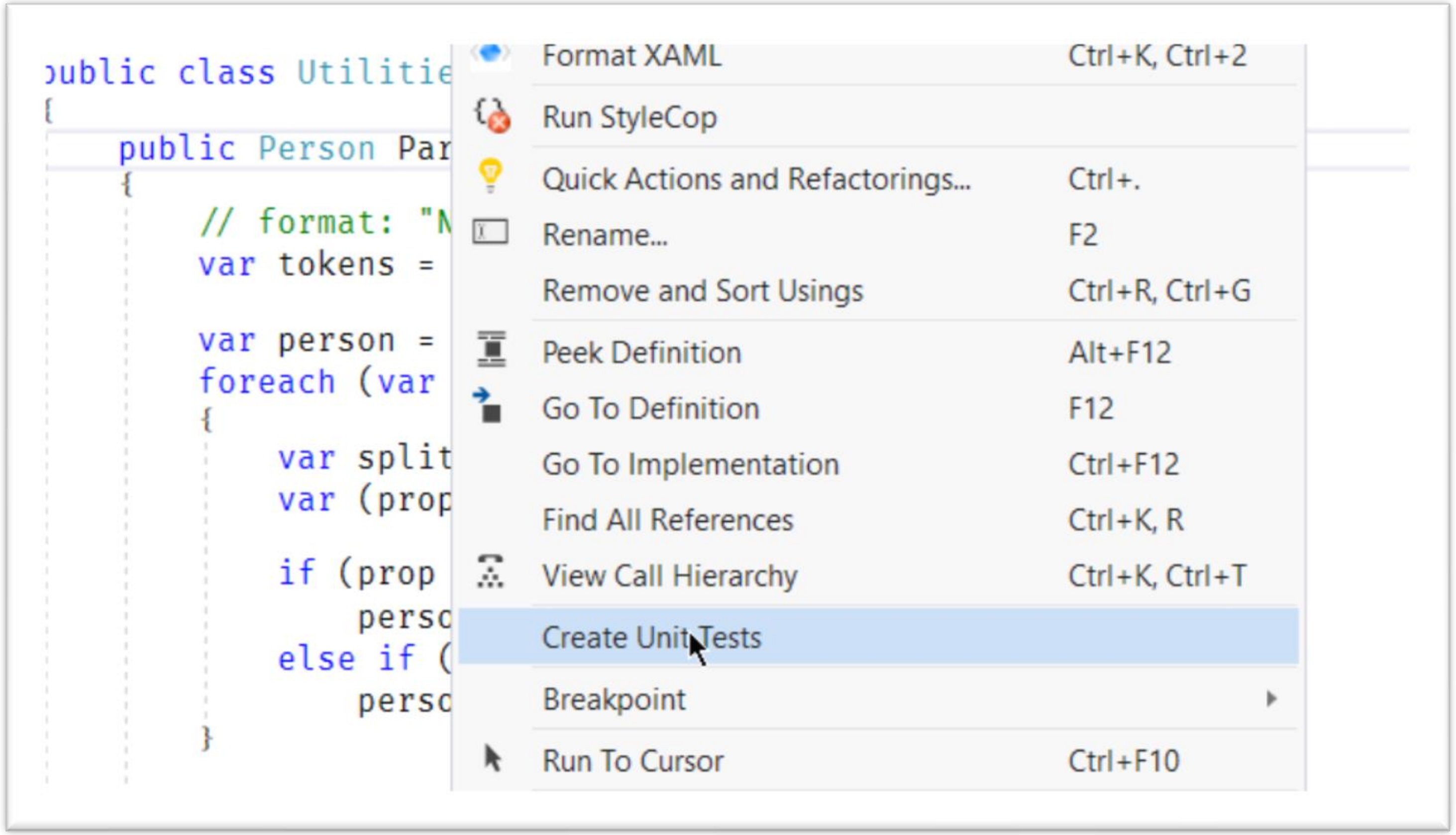


```csharp
namespace ContosoApp
{
    public class Utilities
    {
        public Person ParseString(string data)
        {
            // format: "Name=something;City=10;"
            var tokens = data.Split(';');

            var person = new Person();
            foreach(var token in tokens)
            {
                var splited = token.Split('=');
                var (prop, value) = (splited[0], splited[1]);

                if (prop == nameof(Person.Name))
                    person.Name = value;
                else if (prop == nameof(Person.City))
                    person.City = value;
            }

            return person;
        }
    }
}
```



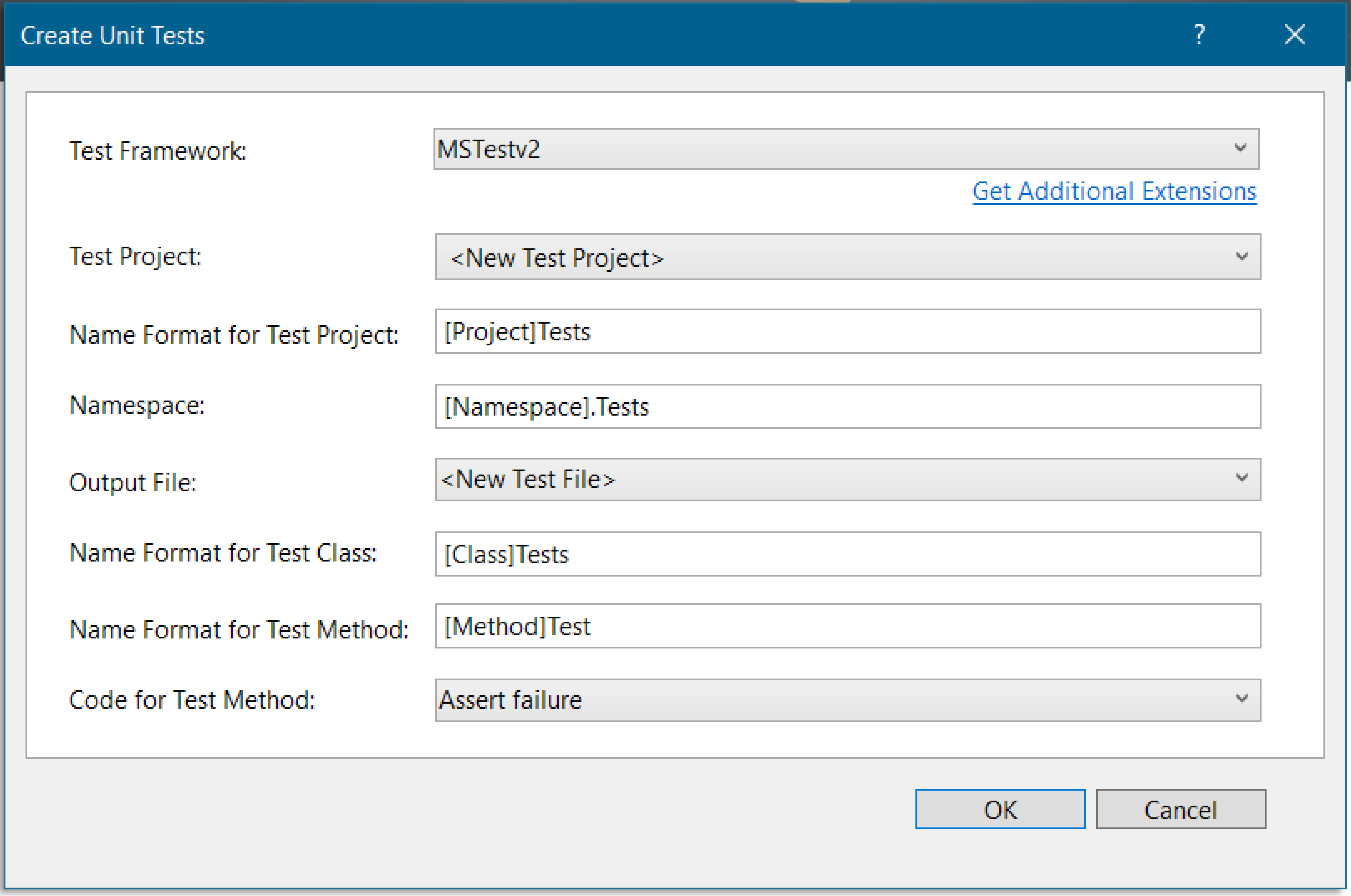
Solution 'ContosoLibrary' (1 project)
- ContosoLibrary
  - Properties
  - References
  - App.config
  - Person.cs
  - Utilities.cs

Basic Examples (2), Create Unit Test Project


```
ɔublic class Utilitie
{
    public Person Par
    {
        // format: "N
        var tokens =

        var person =
        foreach (var
        {
            var split
            var (prop

            if (prop
                perso
            else if (
                perso
        }
```

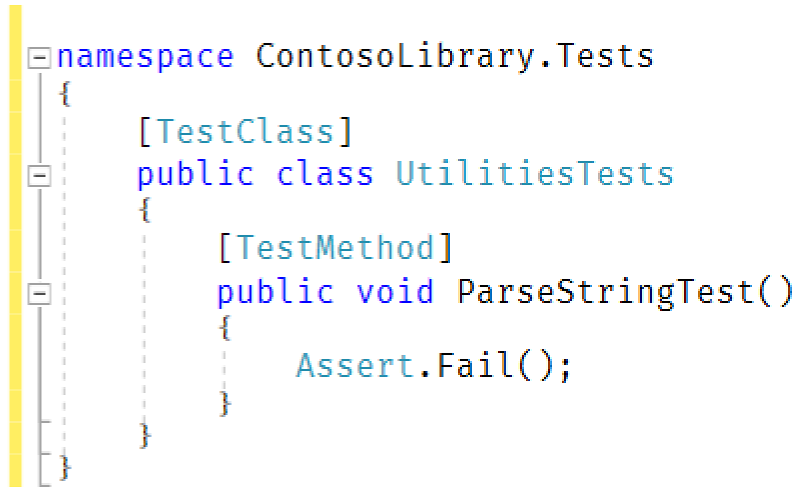
Format XAML Ctrl+K, Ctrl+2
Run StyleCop
Quick Actions and Refactorings... Ctrl+.
Rename... F2
Remove and Sort Usings Ctrl+R, Ctrl+G
Peek Definition Alt+F12
Go To Definition F12
Go To Implementation Ctrl+F12
Find All References Ctrl+K, R
View Call Hierarchy Ctrl+K, Ctrl+T
Create Unit Tests
Breakpoint
Run To Cursor Ctrl+F10

Basic Examples (3), Select Framework and Template

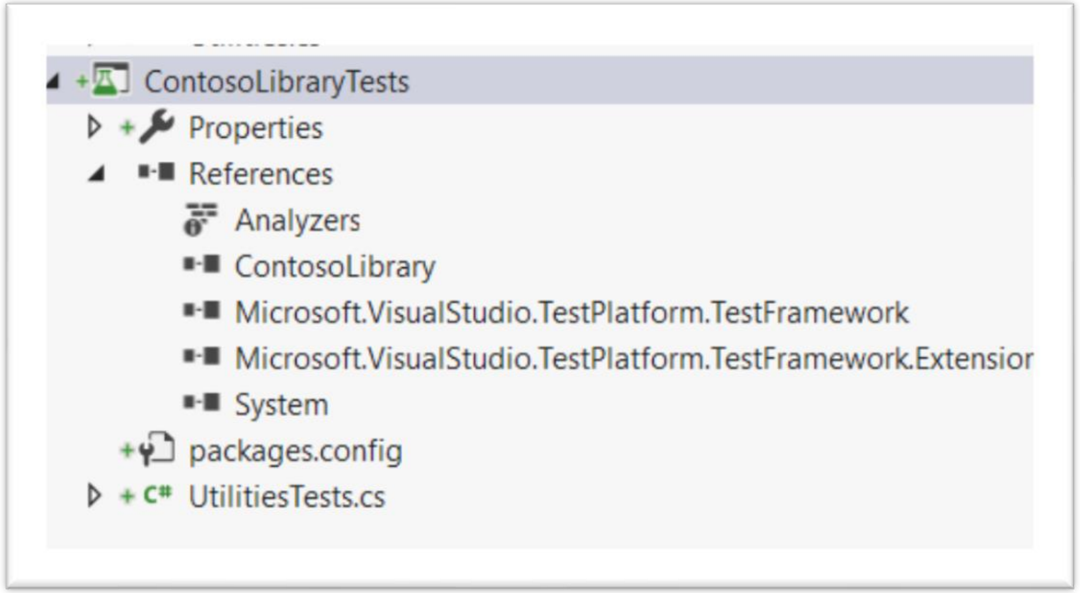


Test Framework:	MSTestv2
Test Project:	<New Test Project>
Name Format for Test Project:	[Project]Tests
Namespace:	[Namespace].Tests
Output File:	<New Test File>
Name Format for Test Class:	[Class]Tests
Name Format for Test Method:	[Method]Test
Code for Test Method:	Assert failure

Basic Examples (4), Generated Test Project

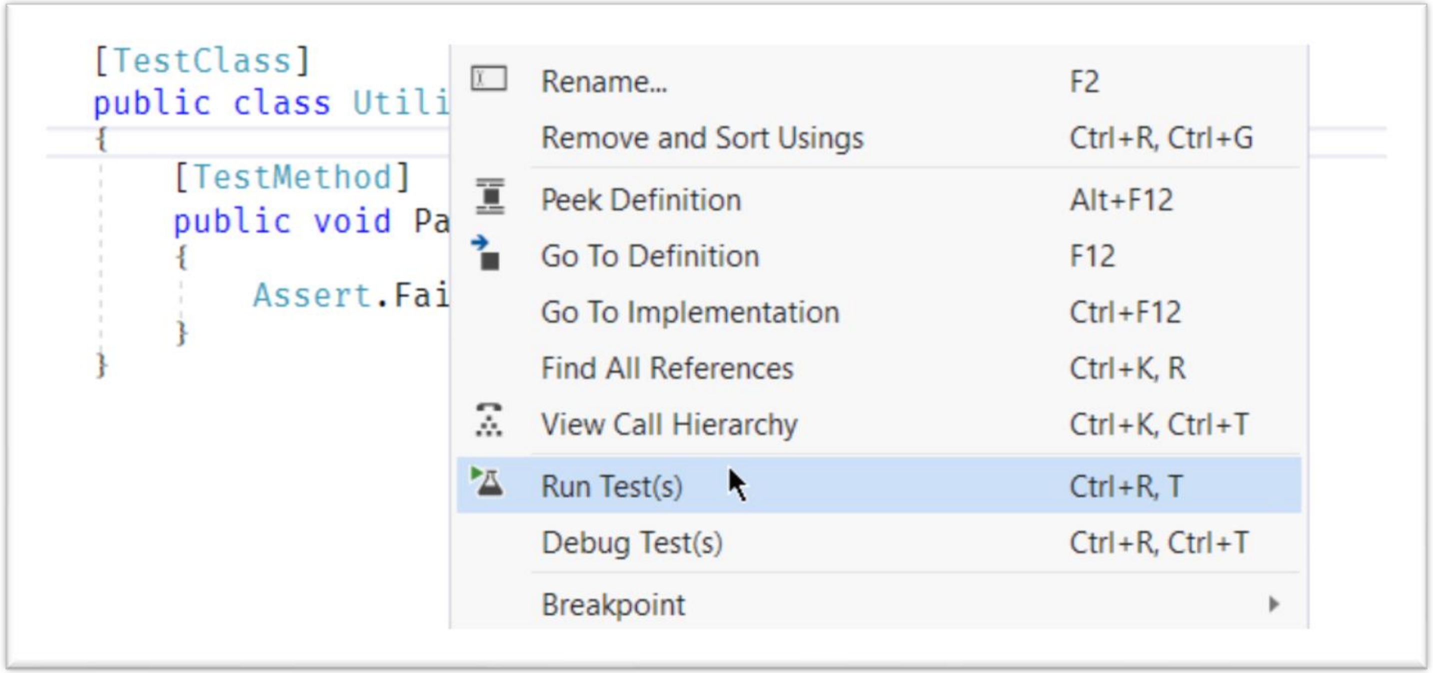


```
namespace ContosoLibrary.Tests
{
    [TestClass]
    public class UtilitiesTests
    {
        [TestMethod]
        public void ParseStringTest()
        {
            Assert.Fail();
        }
    }
}
```



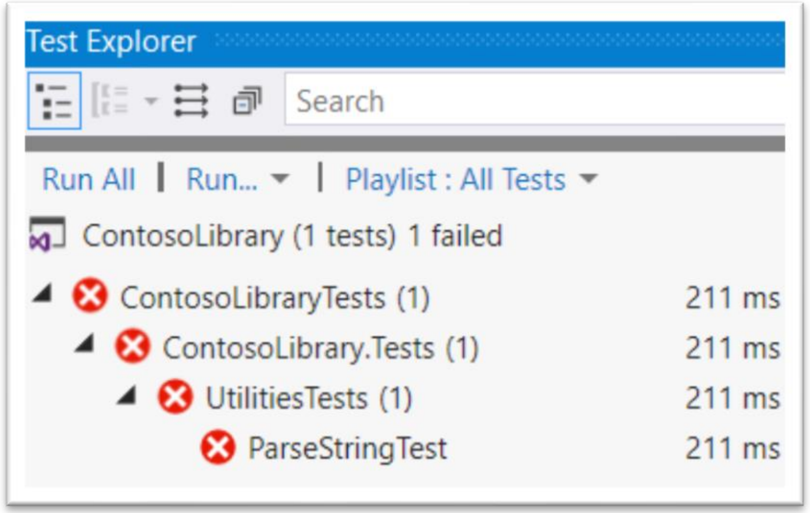
```
◢ ContosoLibraryTests
    ▷ Properties
    ◢ References
        Analyzers
        ContosoLibrary
        Microsoft.VisualStudio.TestPlatform.TestFramework
        Microsoft.VisualStudio.TestPlatform.TestFramework.Extensior
        System
    packages.config
    ▷ UtilitiesTests.cs
```

Basic Examples (5), Run Empty Test



```
[TestClass]
public class Utili
{
    [TestMethod]
    public void Pa
    {
        Assert.Fai
    }
}
```

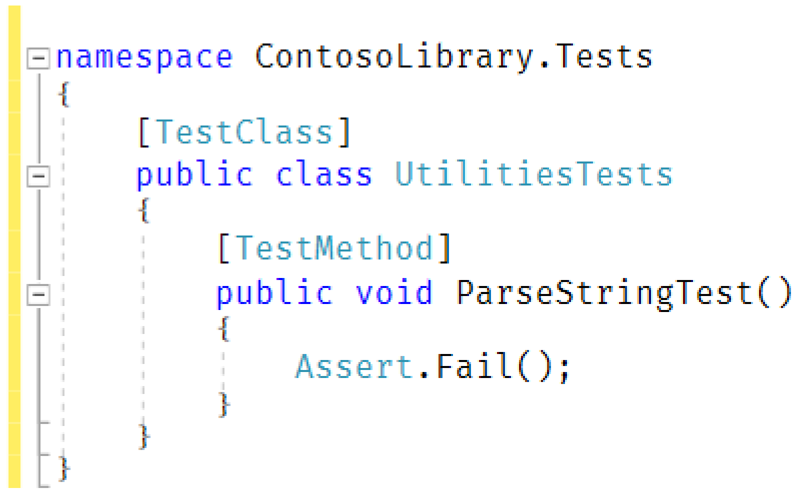
Rename... F2
Remove and Sort Usings Ctrl+R, Ctrl+G
Peek Definition Alt+F12
Go To Definition F12
Go To Implementation Ctrl+F12
Find All References Ctrl+K, R
View Call Hierarchy Ctrl+K, Ctrl+T
Run Test(s) Ctrl+R, T
Debug Test(s) Ctrl+R, Ctrl+T
Breakpoint



```
ContosoLibrary (1 tests) 1 failed
  ContosoLibraryTests (1)          211 ms
    ContosoLibrary.Tests (1)       211 ms
      UtilitiesTests (1)           211 ms
        ParseStringTest            211 ms
```

Basic Examples (6), What to write?

What should we write in this method?



```
namespace ContosoLibrary.Tests
{
    [TestClass]
    public class UtilitiesTests
    {
        [TestMethod]
        public void ParseStringTest()
        {
            Assert.Fail();
        }
    }
}
```

Basic Examples (7), What to write?

Let's use **the** AAA **Pattern**,

Which basically means:

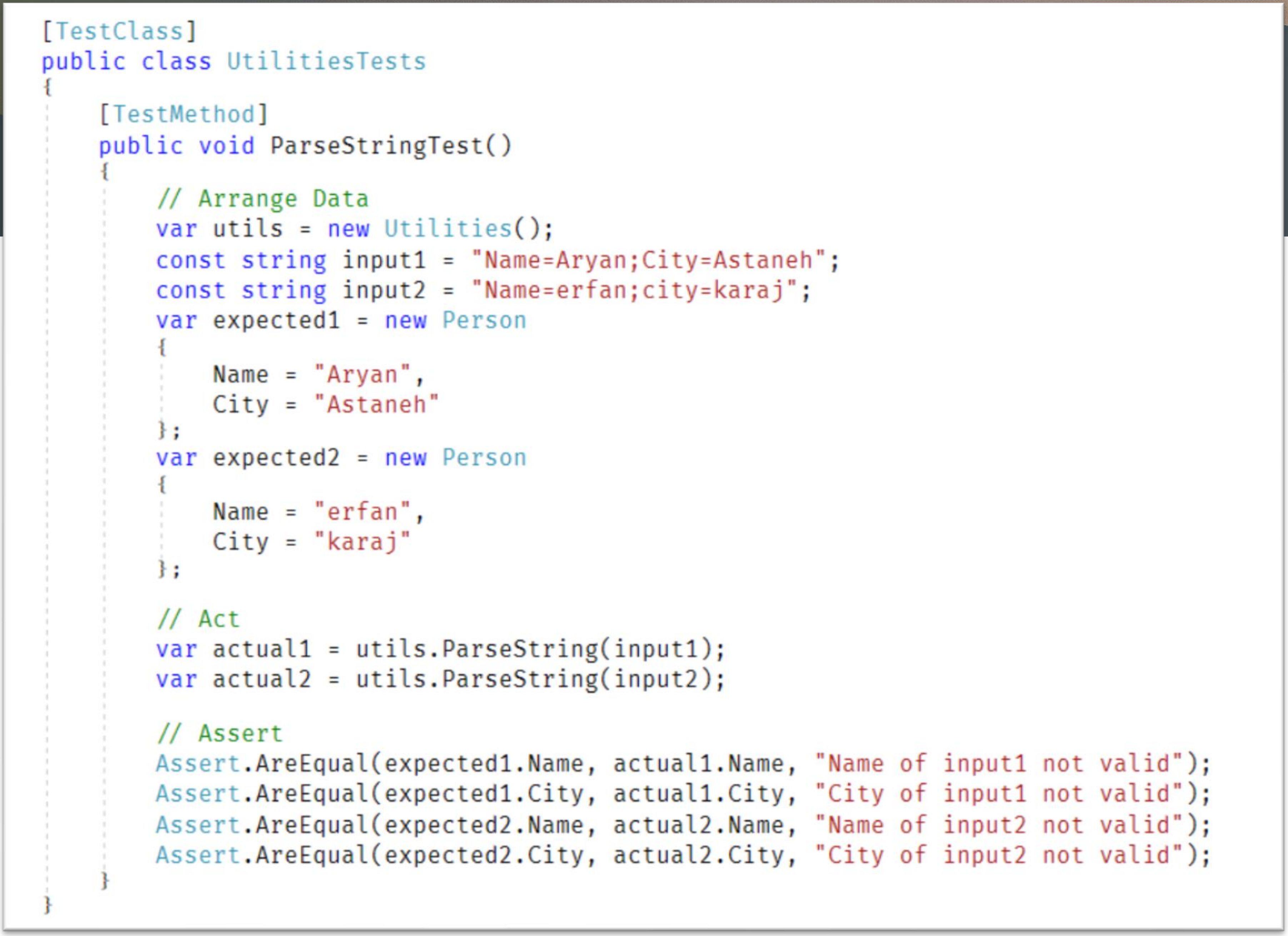
- Arrange
- Act
- Assert

Basic Examples (8), What to write?

Arrange: initializes objects and sets the value of the data that is passed to the method under test.

Act: invokes the method under test with the arranged parameters

Assert: verifies that the action of the method under test behaves as expected

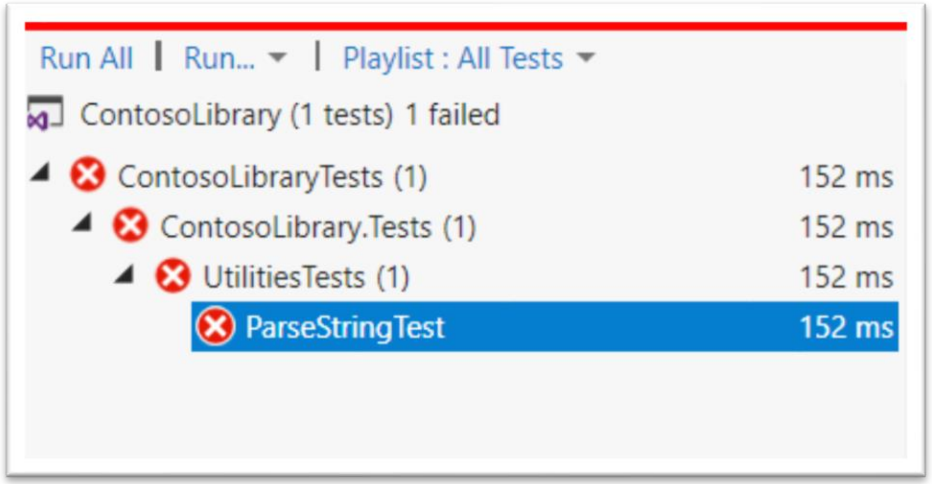


```
[TestClass]
public class UtilitiesTests
{
    [TestMethod]
    public void ParseStringTest()
    {
        // Arrange Data
        var utils = new Utilities();
        const string input1 = "Name=Aryan;City=Astaneh";
        const string input2 = "Name=erfan;city=karaj";
        var expected1 = new Person
        {
            Name = "Aryan",
            City = "Astaneh"
        };
        var expected2 = new Person
        {
            Name = "erfan",
            City = "karaj"
        };

        // Act
        var actual1 = utils.ParseString(input1);
        var actual2 = utils.ParseString(input2);

        // Assert
        Assert.AreEqual(expected1.Name, actual1.Name, "Name of input1 not valid");
        Assert.AreEqual(expected1.City, actual1.City, "City of input1 not valid");
        Assert.AreEqual(expected2.Name, actual2.Name, "Name of input2 not valid");
        Assert.AreEqual(expected2.City, actual2.City, "City of input2 not valid");
    }
}
```

Basic Example (10)

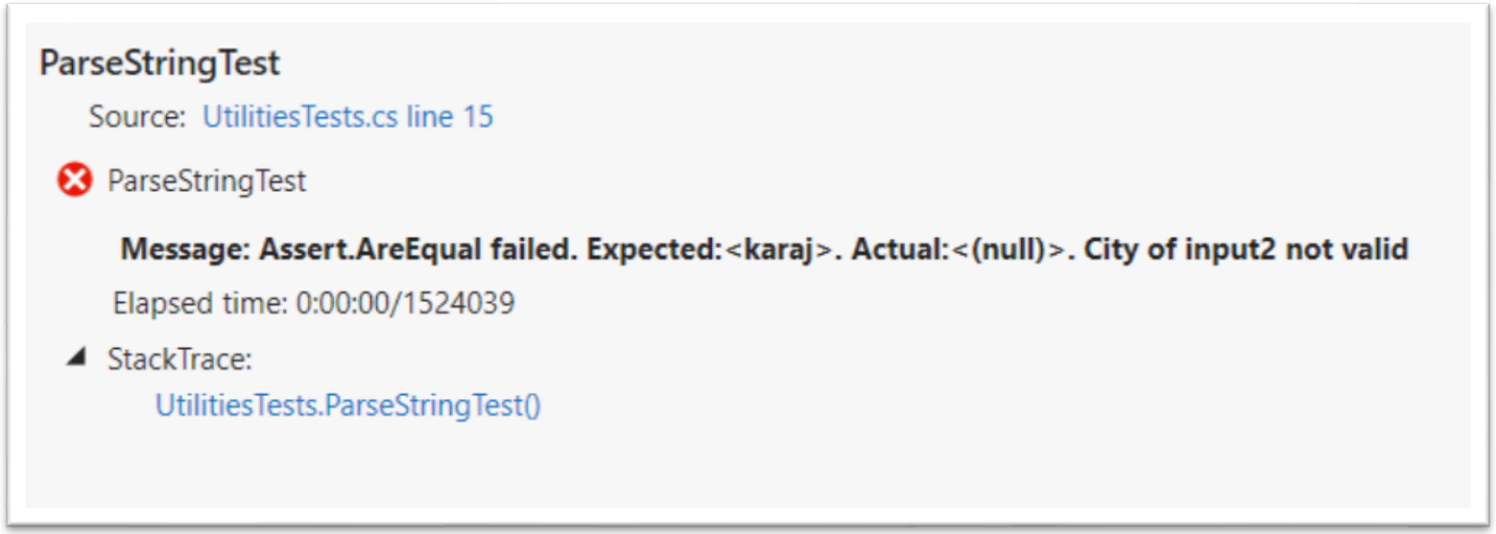
Let's check the result:

Again no! But why?



```
ContosoLibrary (1 tests) 1 failed
ContosoLibraryTests (1)          152 ms
  ContosoLibrary.Tests (1)       152 ms
    UtilitiesTests (1)           152 ms
      ParseStringTest            152 ms
```

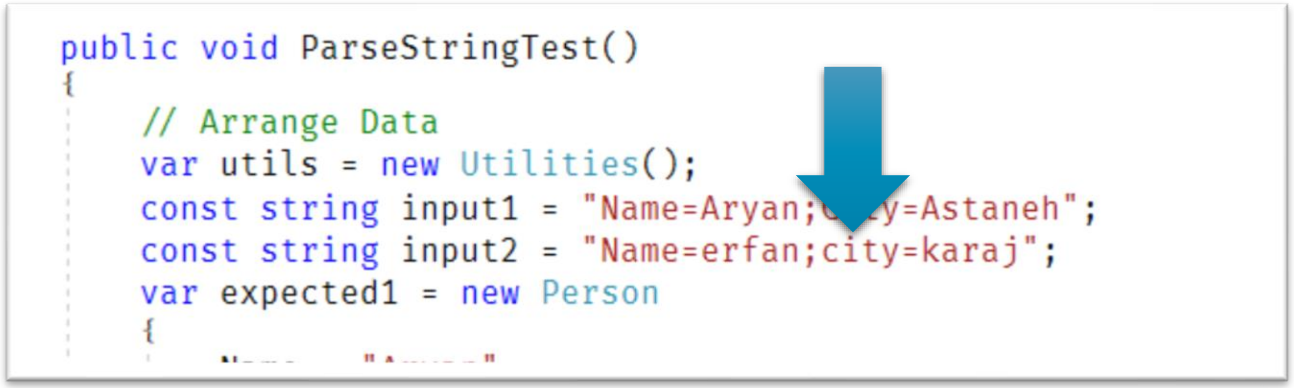
Hmmm... Makes sense!



```
ParseStringTest
   Source: UtilitiesTests.cs line 15

ParseStringTest

   Message: Assert.AreEqual failed. Expected:<karaj>. Actual:<(null)>. City of input2 not valid
   Elapsed time: 0:00:00/1524039
 StackTrace:
    UtilitiesTests.ParseStringTest()
```

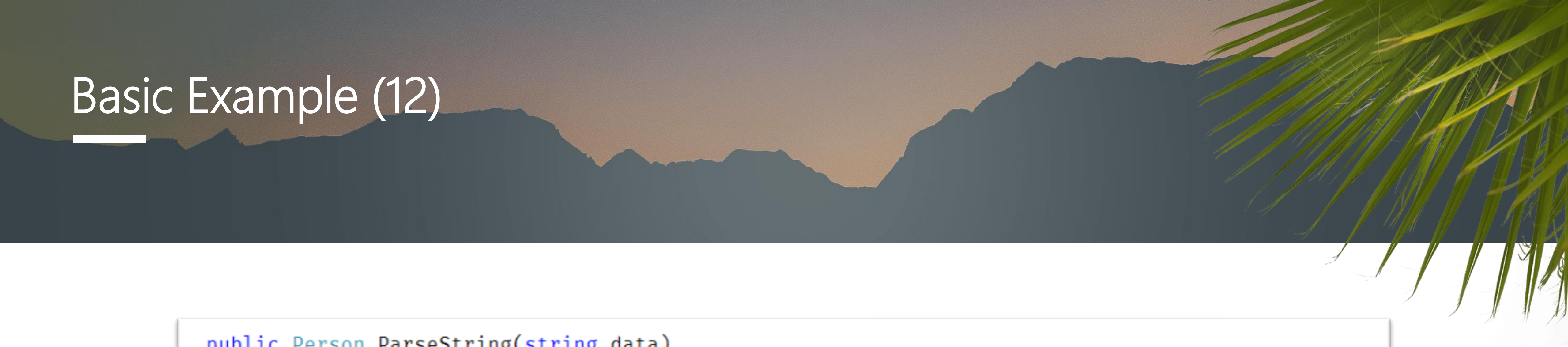
Basic Example (11)

Problem lies here:

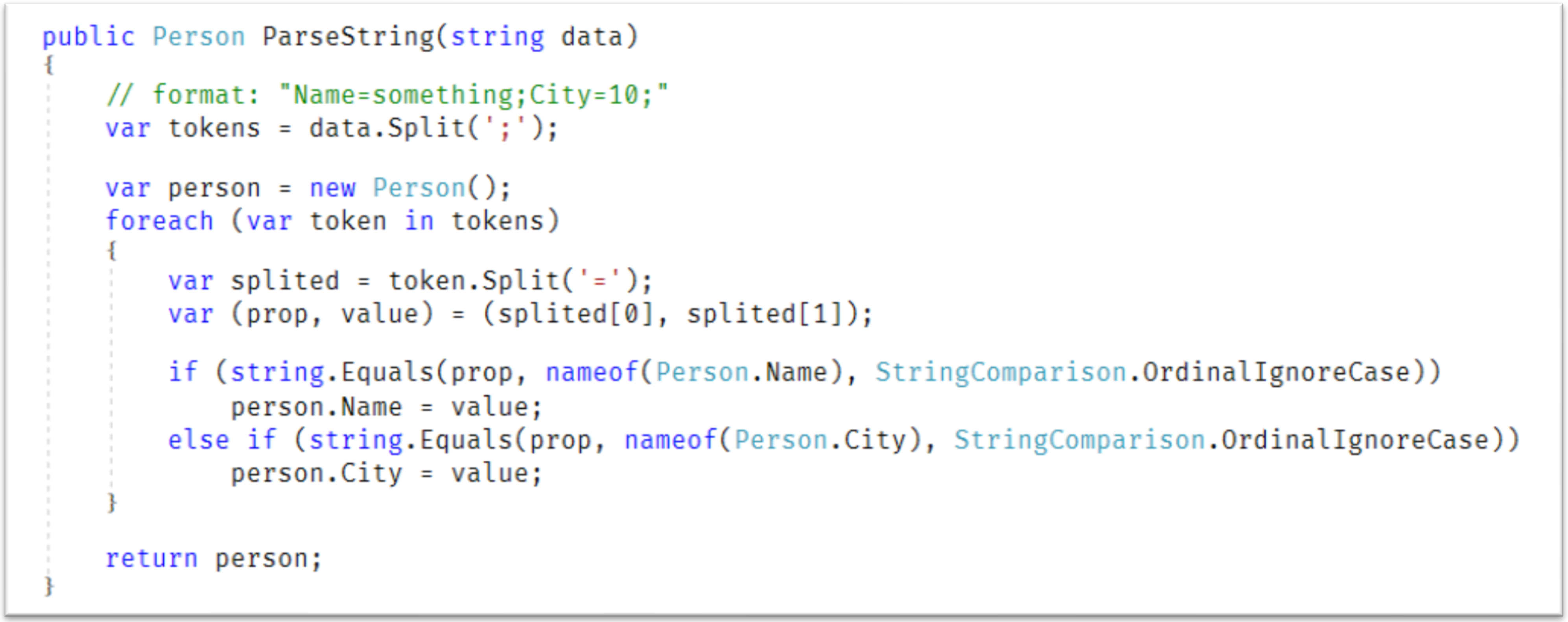


```
public void ParseStringTest()
{
    // Arrange Data
    var utils = new Utilities();
    const string input1 = "Name=Aryan;City=Astaneh";
    const string input2 = "Name=erfan;city=karaj";
    var expected1 = new Person
    {
```

Let’s fix it in the code



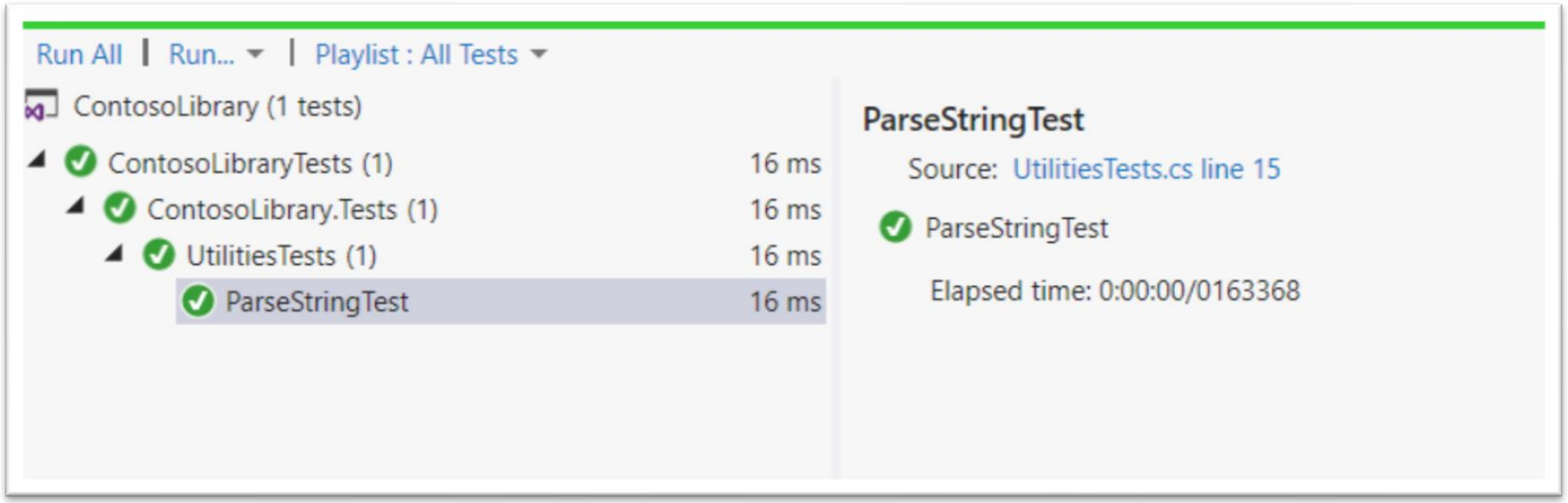
Basic Example (12)



```
public Person ParseString(string data)
{
    // format: "Name=something;City=10;"
    var tokens = data.Split(';');

    var person = new Person();
    foreach (var token in tokens)
    {
        var splited = token.Split('=');
        var (prop, value) = (splited[0], splited[1]);

        if (string.Equals(prop, nameof(Person.Name), StringComparison.OrdinalIgnoreCase))
            person.Name = value;
        else if (string.Equals(prop, nameof(Person.City), StringComparison.OrdinalIgnoreCase))
            person.City = value;
    }

    return person;
}
```

Basic Example (13)

Let's check the result one more time:


Run All | Run... | Playlist : All Tests

ContosoLibrary (1 tests)

```
ContosoLibraryTests (1)            16 ms
  ContosoLibrary.Tests (1)         16 ms
    UtilitiesTests (1)             16 ms
      ParseStringTest              16 ms
```

ParseStringTest

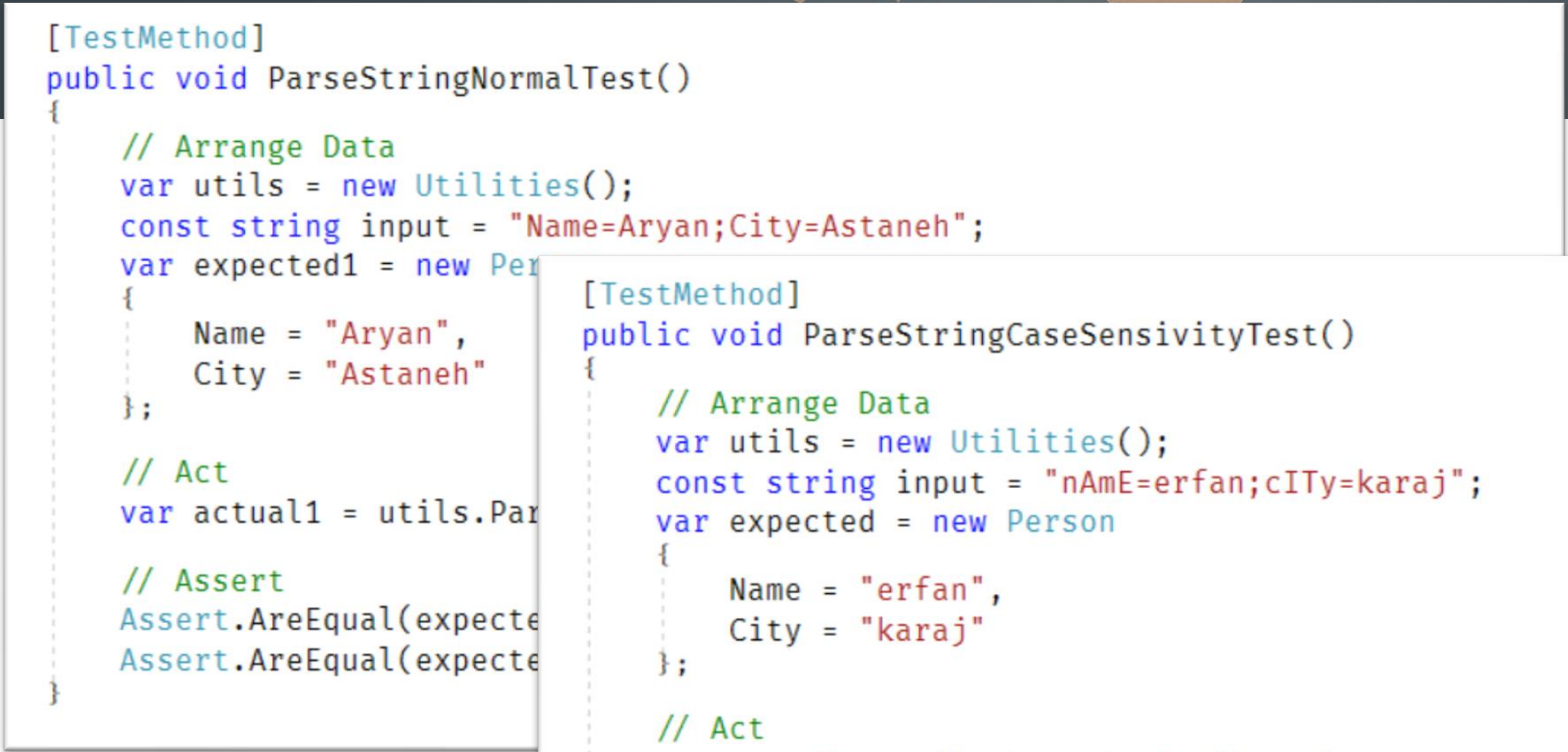
Source: UtilitiesTests.cs line 15

ParseStringTest

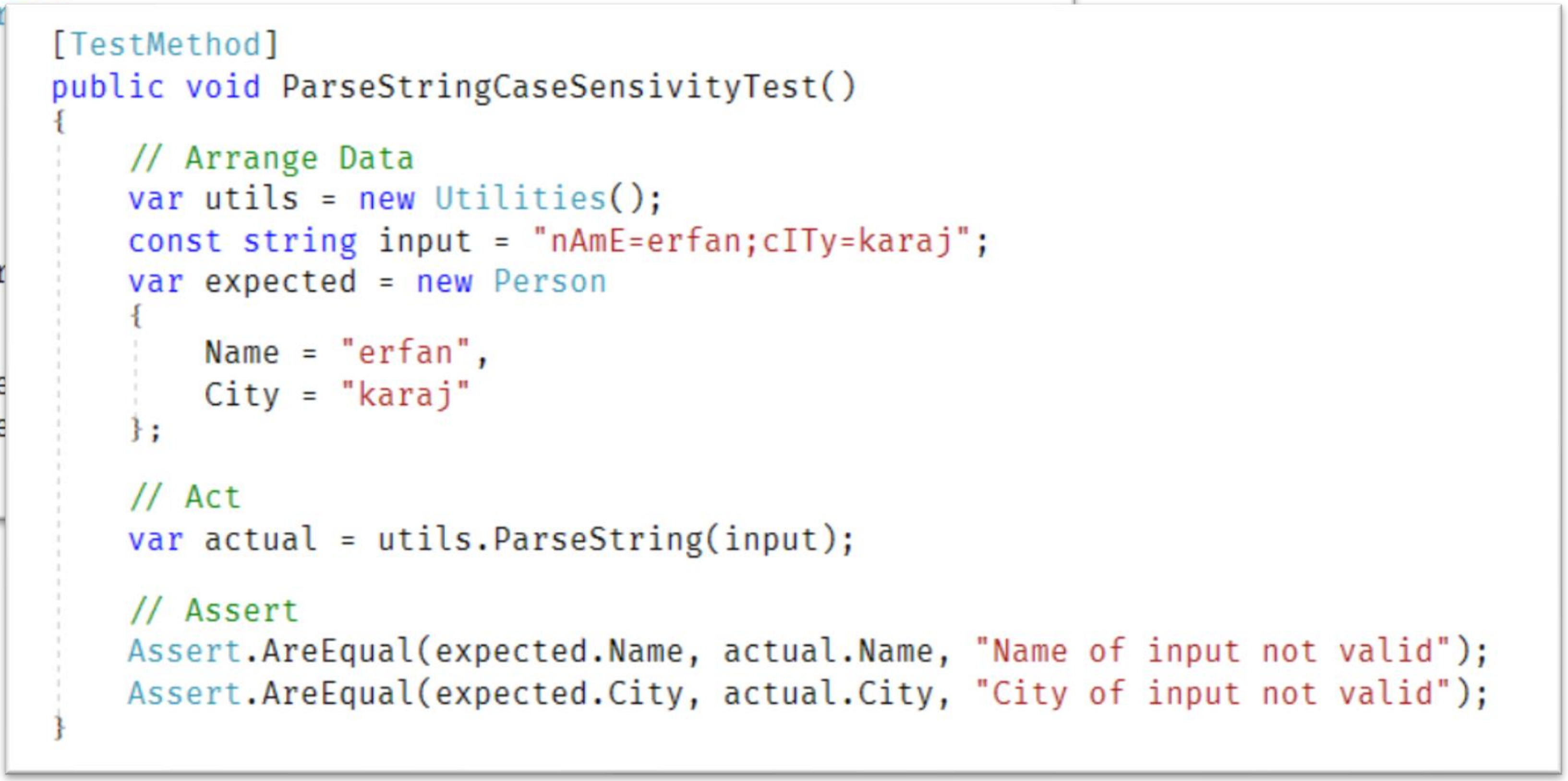
Elapsed time: 0:00:00/0163368

The result is: A happy programmer

Basic Example (14), Split Test Methods



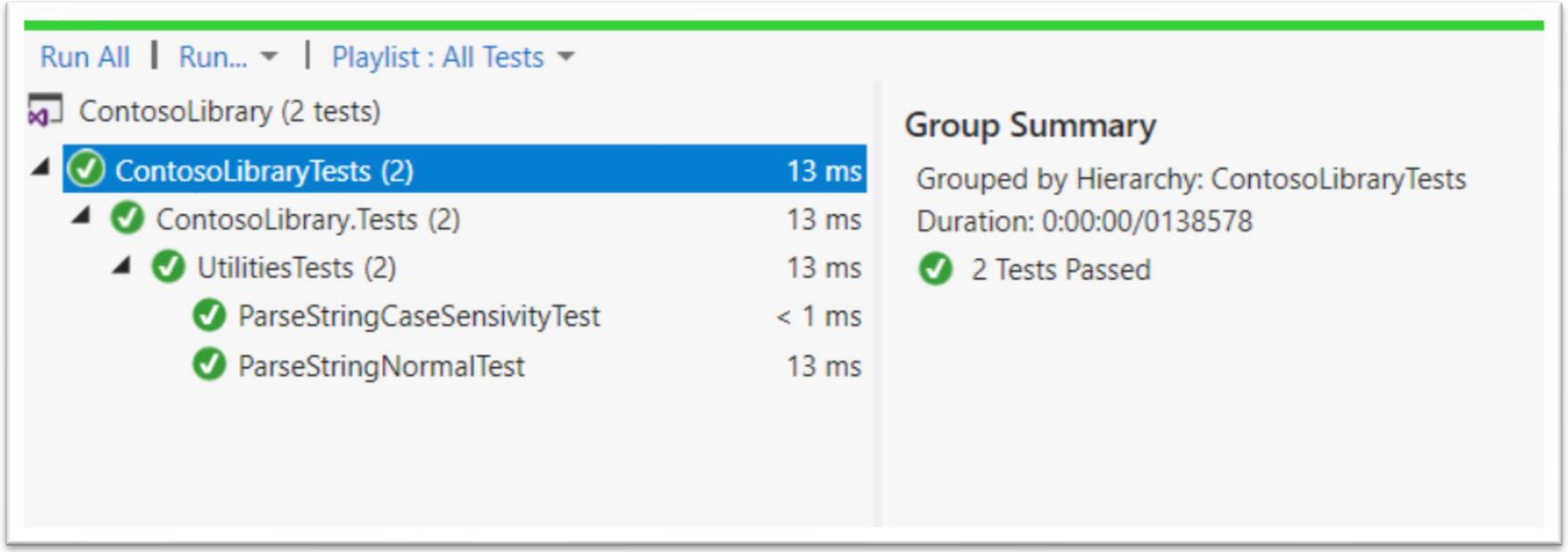
```
[TestMethod]
public void ParseStringNormalTest()
{
    // Arrange Data
    var utils = new Utilities();
    const string input = "Name=Aryan;City=Astaneh";
    var expected1 = new Per
    {
        Name = "Aryan",
        City = "Astaneh"
    };

    // Act
    var actual1 = utils.Par

    // Assert
    Assert.AreEqual(expecte
    Assert.AreEqual(expecte
}
```



```
[TestMethod]
public void ParseStringCaseSensivityTest()
{
    // Arrange Data
    var utils = new Utilities();
    const string input = "nAmE=erfan;cITy=karaj";
    var expected = new Person
    {
        Name = "erfan",
        City = "karaj"
    };

    // Act
    var actual = utils.ParseString(input);

    // Assert
    Assert.AreEqual(expected.Name, actual.Name, "Name of input not valid");
    Assert.AreEqual(expected.City, actual.City, "City of input not valid");
}
```

Basic Example (15), Split Test Methods

And there will be 2 tests in the Test Explorer



```
ContosoLibrary (2 tests)
◢ ✓ ContosoLibraryTests (2)                    13 ms
  ◢ ✓ ContosoLibrary.Tests (2)                 13 ms
    ◢ ✓ UtilitiesTests (2)                     13 ms
        ✓ ParseStringCaseSensivityTest        < 1 ms
        ✓ ParseStringNormalTest                13 ms
```

Group Summary

Grouped by Hierarchy: ContosoLibraryTests
Duration: 0:00:00/0138578
✓ 2 Tests Passed

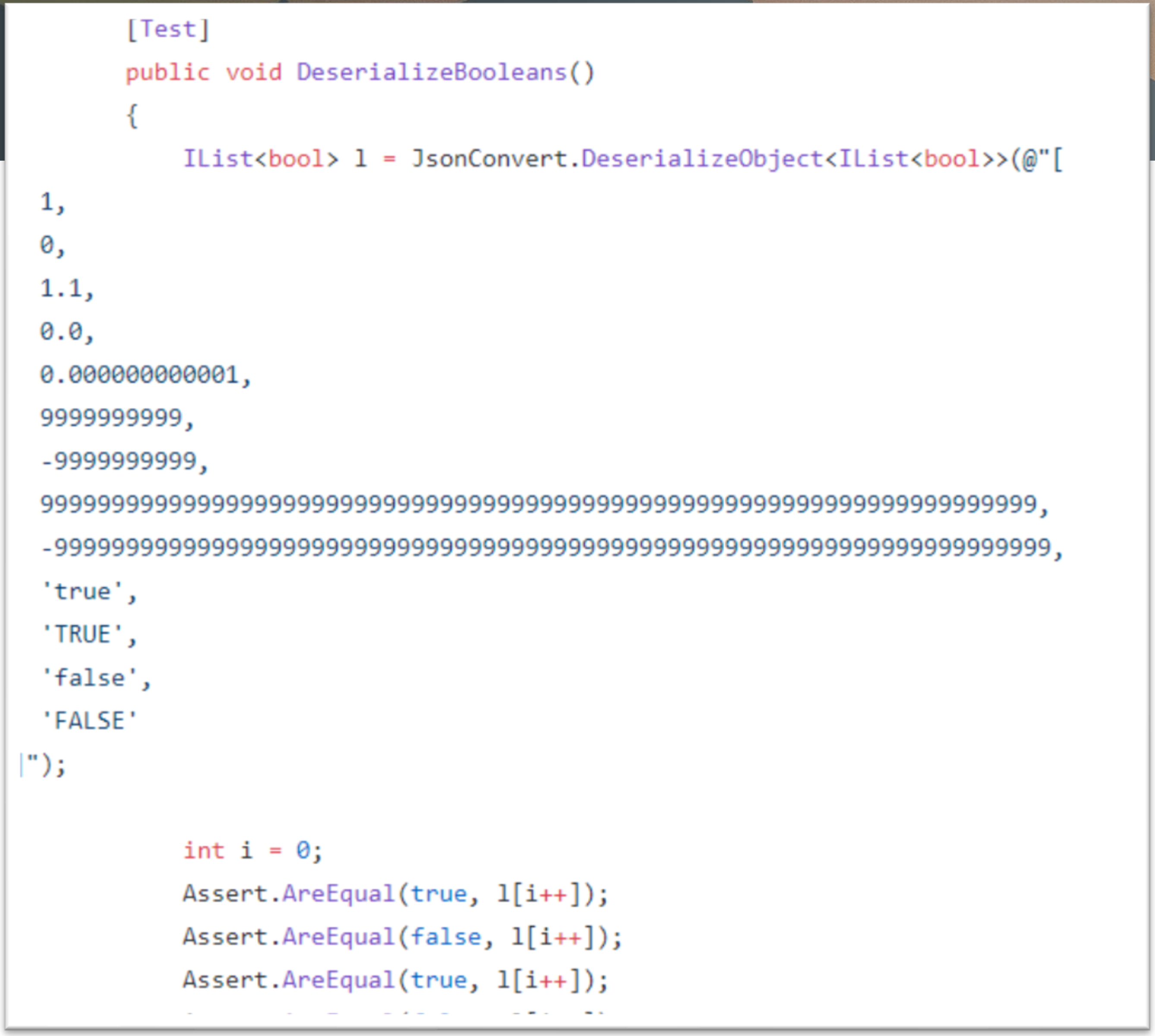
Real World Examples

Let’s check out GitHub for some popular projects

Newtonsoft Json .NET

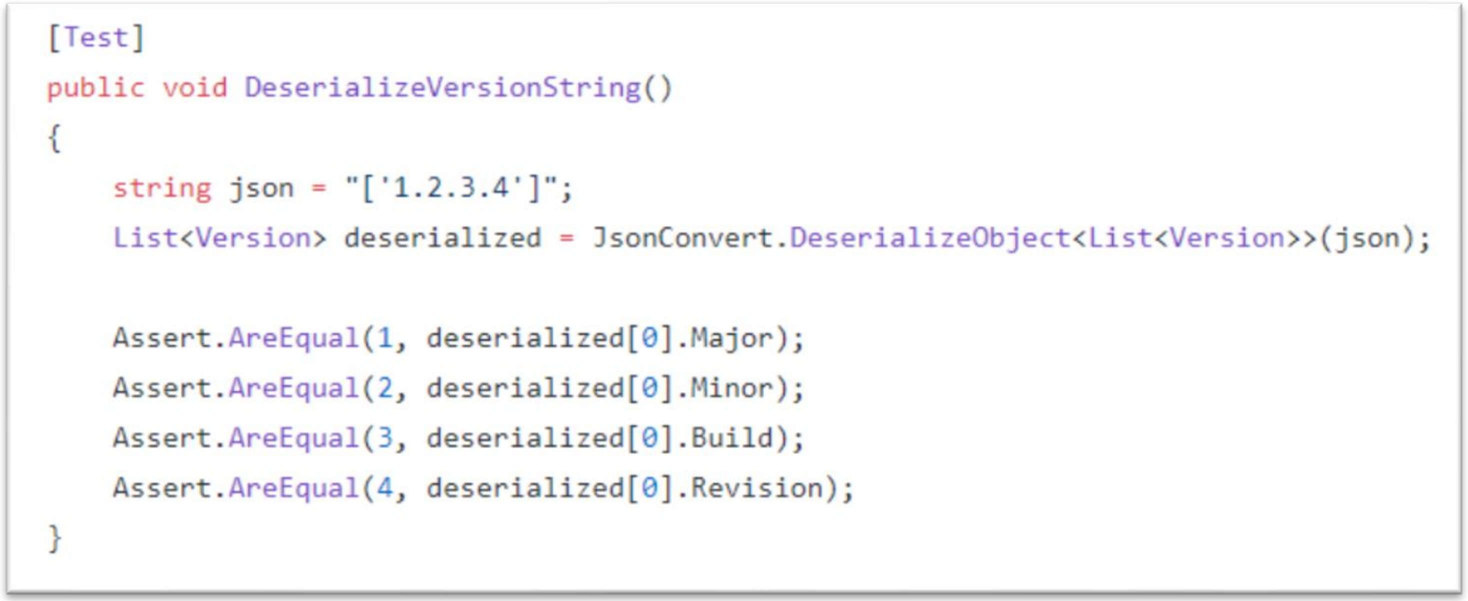
Most popular Nuget package in .NET ecosystem

Newtonsoft Json .NET



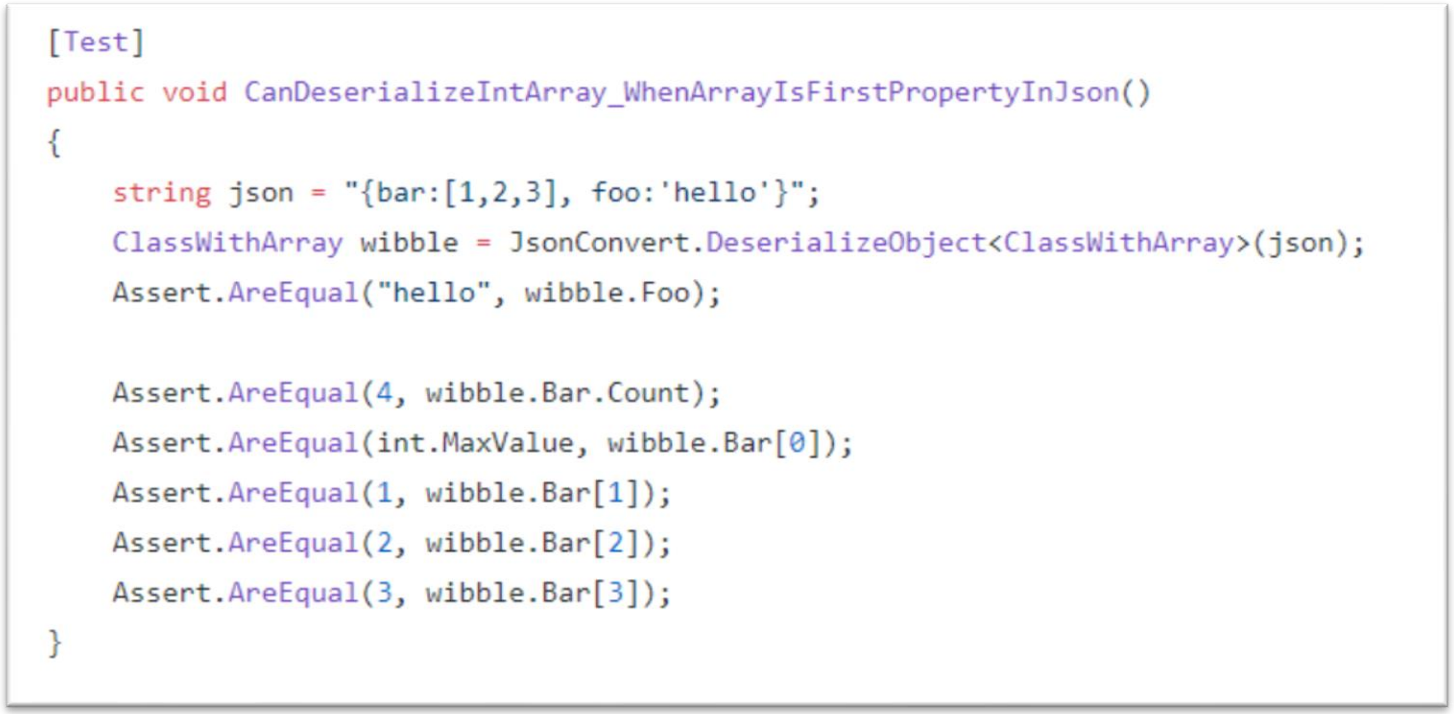
```
      [Test]
      public void DeserializeBooleans()
      {
          IList<bool> l = JsonConvert.DeserializeObject<IList<bool>>(@"[
1,
0,
1.1,
0.0,
0.000000000001,
9999999999,
-9999999999,
9999999999999999999999999999999999999999999999999999999999999999999999,
-9999999999999999999999999999999999999999999999999999999999999999999999,
'true',
'TRUE',
'false',
'FALSE'
]");


          int i = 0;
          Assert.AreEqual(true, l[i++]);
          Assert.AreEqual(false, l[i++]);
          Assert.AreEqual(true, l[i++]);
```

Newtonsoft Json .NET



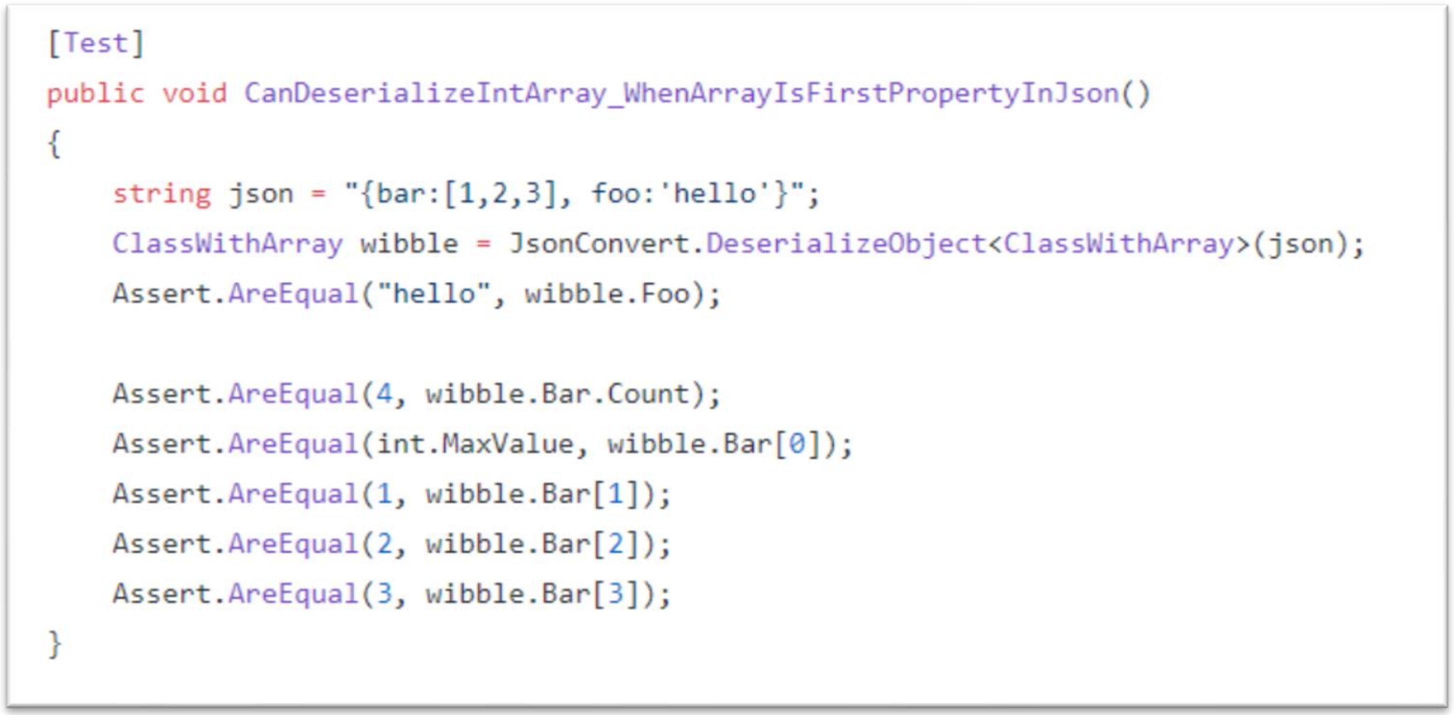
```
[Test]
public void DeserializeVersionString()
{
    string json = "['1.2.3.4']";
    List<Version> deserialized = JsonConvert.DeserializeObject<List<Version>>(json);

    Assert.AreEqual(1, deserialized[0].Major);
    Assert.AreEqual(2, deserialized[0].Minor);
    Assert.AreEqual(3, deserialized[0].Build);
    Assert.AreEqual(4, deserialized[0].Revision);
}
```

Newtonsoft Json .NET



```
[Test]
public void CanDeserializeIntArray_WhenArrayIsFirstPropertyInJson()
{
    string json = "{bar:[1,2,3], foo:'hello'}";
    ClassWithArray wibble = JsonConvert.DeserializeObject<ClassWithArray>(json);
    Assert.AreEqual("hello", wibble.Foo);

    Assert.AreEqual(4, wibble.Bar.Count);
    Assert.AreEqual(int.MaxValue, wibble.Bar[0]);
    Assert.AreEqual(1, wibble.Bar[1]);
    Assert.AreEqual(2, wibble.Bar[2]);
    Assert.AreEqual(3, wibble.Bar[3]);
}
```

Newtonsoft Json .NET

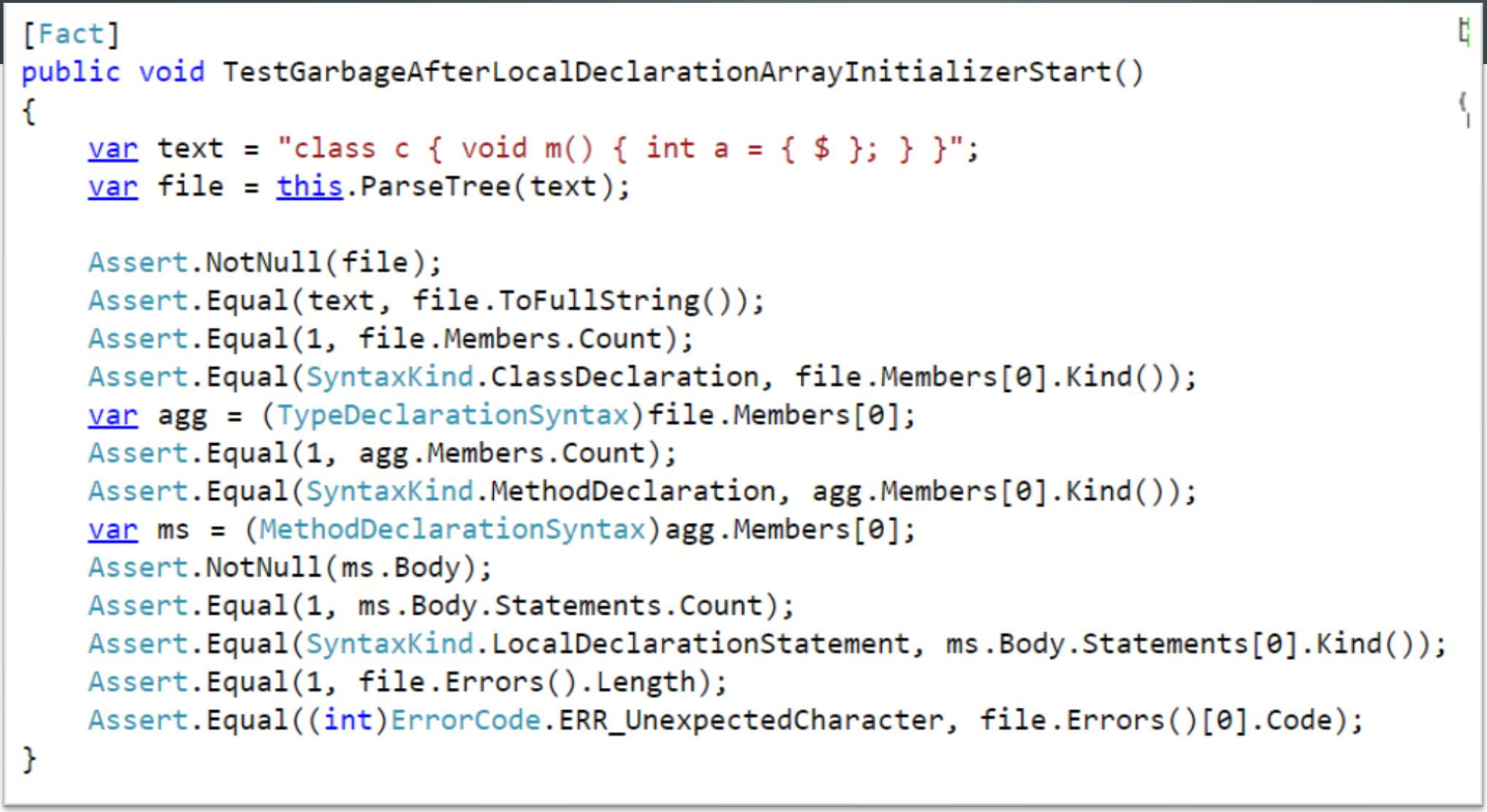


```
[Test]
public void CanDeserializeIntArray_WhenArrayIsFirstPropertyInJson()
{
    string json = "{bar:[1,2,3], foo:'hello'}";
    ClassWithArray wibble = JsonConvert.DeserializeObject<ClassWithArray>(json);
    Assert.AreEqual("hello", wibble.Foo);

    Assert.AreEqual(4, wibble.Bar.Count);
    Assert.AreEqual(int.MaxValue, wibble.Bar[0]);
    Assert.AreEqual(1, wibble.Bar[1]);
    Assert.AreEqual(2, wibble.Bar[2]);
    Assert.AreEqual(3, wibble.Bar[3]);
}
```

Microsoft C# Language

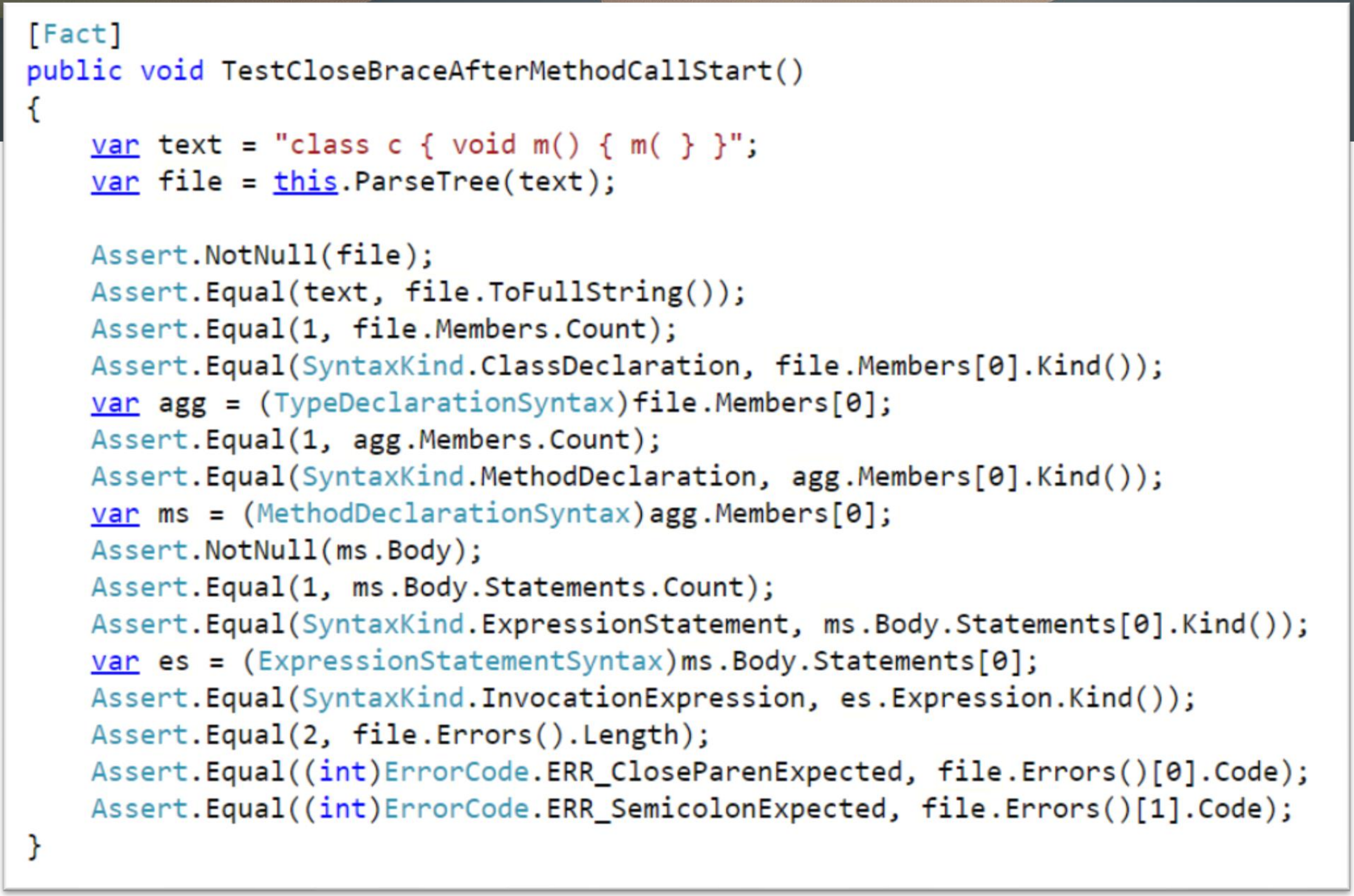
Roslyn tests for C# Language

Microsoft C# Language



```
[Fact]
public void TestGarbageAfterLocalDeclarationArrayInitializerStart()
{
    var text = "class c { void m() { int a = { $ }; } }";
    var file = this.ParseTree(text);

    Assert.NotNull(file);
    Assert.Equal(text, file.ToFullString());
    Assert.Equal(1, file.Members.Count);
    Assert.Equal(SyntaxKind.ClassDeclaration, file.Members[0].Kind());
    var agg = (TypeDeclarationSyntax)file.Members[0];
    Assert.Equal(1, agg.Members.Count);
    Assert.Equal(SyntaxKind.MethodDeclaration, agg.Members[0].Kind());
    var ms = (MethodDeclarationSyntax)agg.Members[0];
    Assert.NotNull(ms.Body);
    Assert.Equal(1, ms.Body.Statements.Count);
    Assert.Equal(SyntaxKind.LocalDeclarationStatement, ms.Body.Statements[0].Kind());
    Assert.Equal(1, file.Errors().Length);
    Assert.Equal((int)ErrorCode.ERR_UnexpectedCharacter, file.Errors()[0].Code);
}
```

Microsoft C# Language



```
[Fact]
public void TestCloseBraceAfterMethodCallStart()
{
    var text = "class c { void m() { m( } }";
    var file = this.ParseTree(text);

    Assert.NotNull(file);
    Assert.Equal(text, file.ToFullString());
    Assert.Equal(1, file.Members.Count);
    Assert.Equal(SyntaxKind.ClassDeclaration, file.Members[0].Kind());
    var agg = (TypeDeclarationSyntax)file.Members[0];
    Assert.Equal(1, agg.Members.Count);
    Assert.Equal(SyntaxKind.MethodDeclaration, agg.Members[0].Kind());
    var ms = (MethodDeclarationSyntax)agg.Members[0];
    Assert.NotNull(ms.Body);
    Assert.Equal(1, ms.Body.Statements.Count);
    Assert.Equal(SyntaxKind.ExpressionStatement, ms.Body.Statements[0].Kind());
    var es = (ExpressionStatementSyntax)ms.Body.Statements[0];
    Assert.Equal(SyntaxKind.InvocationExpression, es.Expression.Kind());
    Assert.Equal(2, file.Errors().Length);
    Assert.Equal((int)ErrorCode.ERR_CloseParenExpected, file.Errors()[0].Code);
    Assert.Equal((int)ErrorCode.ERR_SemicolonExpected, file.Errors()[1].Code);
}
```

Behavior-driven Development

Or simply BDD

BDD

- Emerged from TDD (or Test-driven Development)
- Express the behavior and the expected outcomes

BDD

Story: Returns go to stock

As a store owner
In order to keep track of stock
I want to add items back to stock when they're returned.

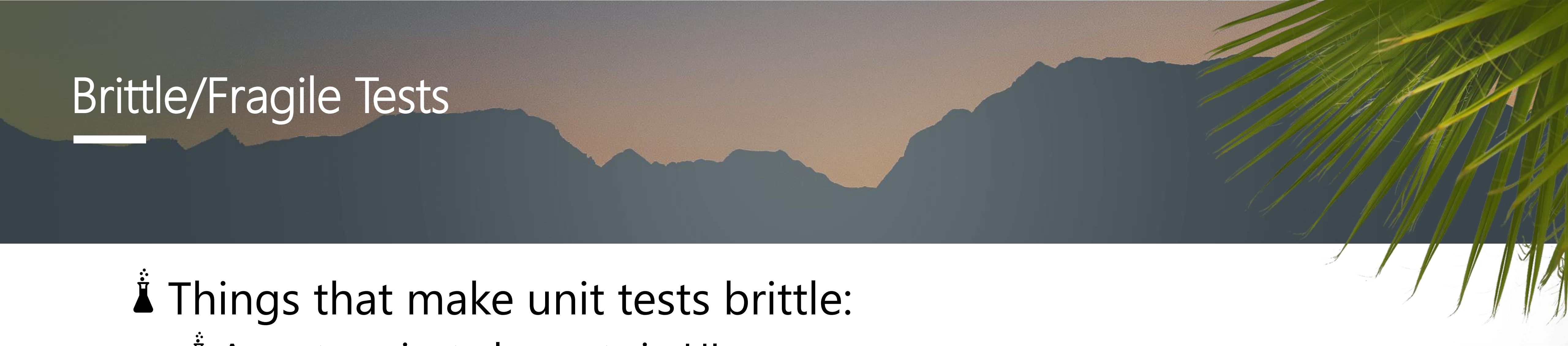
Scenario 1: Refunded items should be returned to stock

Given that a customer previously bought a black sweater from me
And I have three black sweaters in stock.
When they return the black sweater for a refund
Then I should have four black sweaters in stock.

Scenario 2: Replaced items should be returned to stock
Given that a customer previously bought a blue garment from me
And I have two blue garments in stock
And three black garments in stock.
When they return the blue garment for a replacement in black
Then I should have three blue garments in stock
And two black garments in stock.

Brittle/Fragile Tests

Tests that are easy to break



Brittle/Fragile Tests

- Things that make unit tests brittle:
  - Assert against elements in UI
  - Asserting against large result strings instead of small ones
  - Static states shared between threads
  - Unrealistic test data
  - Brittle code under test
  - Too many responsibilities
  - Tests whose results can vary based on the environment, such as performance timings.
  - Tests that can't run independently of each other.

References

Microsoft

References

Robert E. Filman; Tzilla Elrad; Siobhán Clarke; Mehmet Aksit (2004). Aspect-Oriented Dependency Management

Anastasios Manessis, Adrian Hilton, Phil McLauchlan and Phil Palmer (2000). "A Statistical Geometric Framework for Reconstruction of Scene Models"

Virginia Postrel (1999). "Power fantasies: the strange appeal of the Y2K bug – Year 2000 transition problem"

Roger S. Pressman, Software Engineering: A Practitioner's Approach, 7/e

https://github.com/JamesNK/Newtonsoft.Json

http://source.roslyn.io

http://url.aryan.software/TestRef_BDD

Thank you!