Testing Automation

Conditions and Terms of Use

Microsoft Confidential

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

Copyright and Trademarks

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see Use of Microsoft Copyrighted Content at

http://www.microsoft.com/en-us/legal/intellectualproperty/Permissions/default.aspx

DirectX, Hyper-V, Internet Explorer, Microsoft, Outlook, OneDrive, SQL Server, Windows, Microsoft Azure, Windows PowerShell, Windows Server, Windows Vista, and Zune are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Module: Visual Studio Testing

Module Overview

Overview

- Unit Testing Fundamentals
- Unit Tests
- Isolating Code with Fakes
- Selenium
- Appium

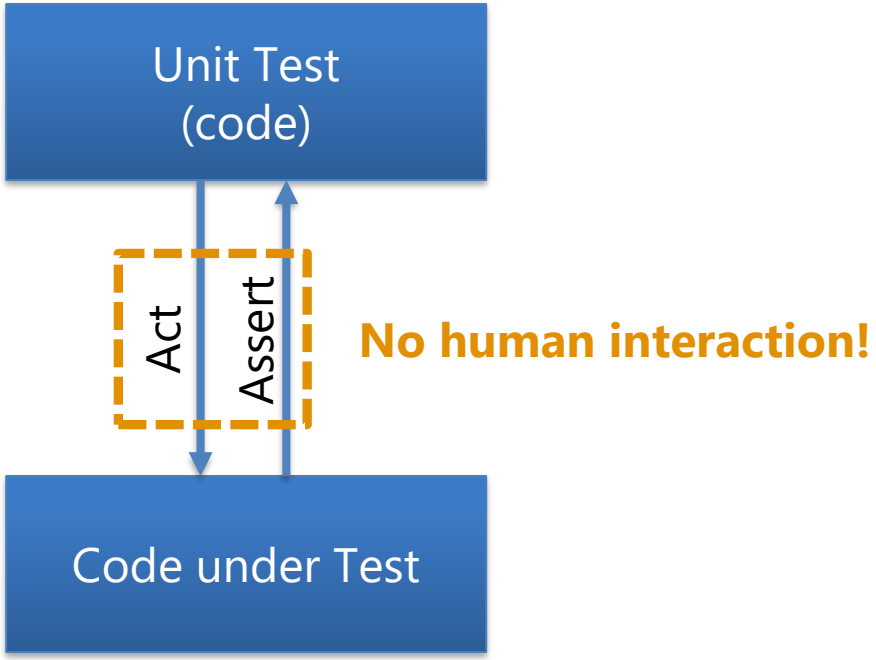
Module: Visual Studio Testing

Lesson 1: Unit Testing Fundamentals

Overview

- What is a Unit Test?
- Benefits of Unit Tests
- Unit Testing Framework
- Unit Tests vs Integration Tests

What is a Unit Test?



What is a Unit Test? (continued)

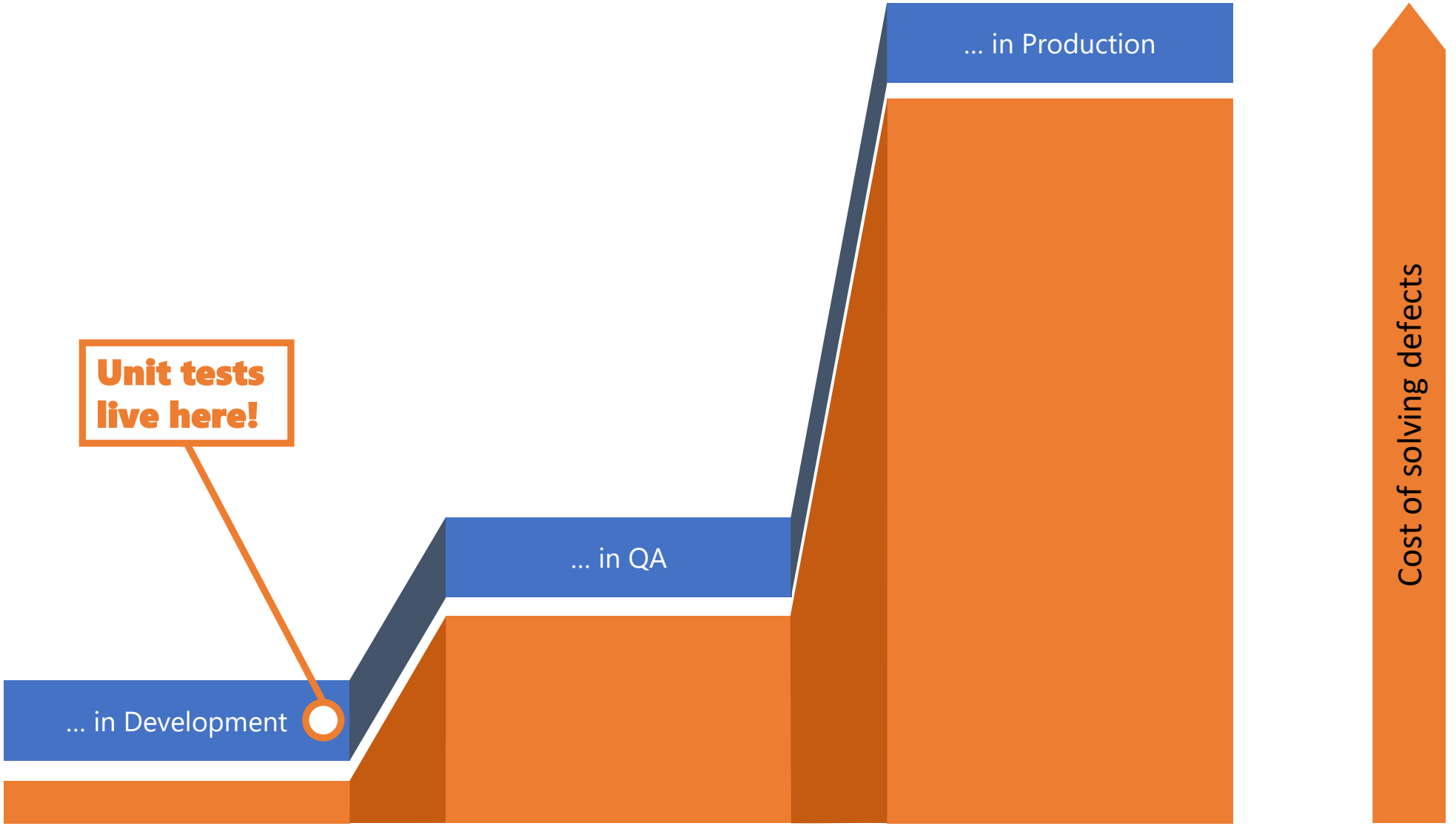
Unit Test

```
[TestMethod]
public void TestGetNameOfNumber()
{
    //Arrange
    var converter = new Converter();
    //Act
    string result = converter.GetNameOfNumber(1);
    //Assert
    Assert.AreEqual("One", result);
}
```

Marks unit test

Asserts result

Benefits of Unit Tests

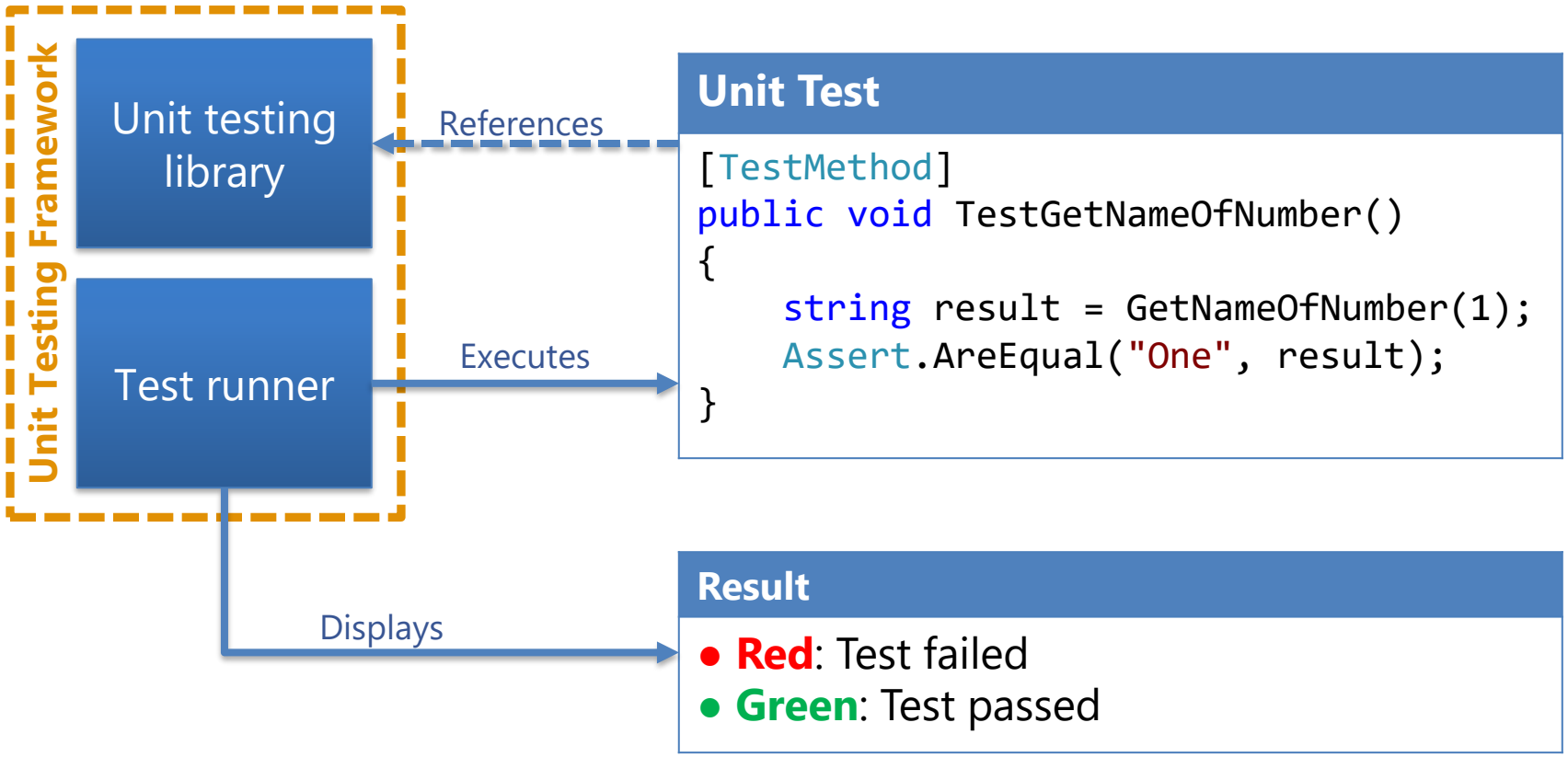


Benefits of Unit Tests (continued)

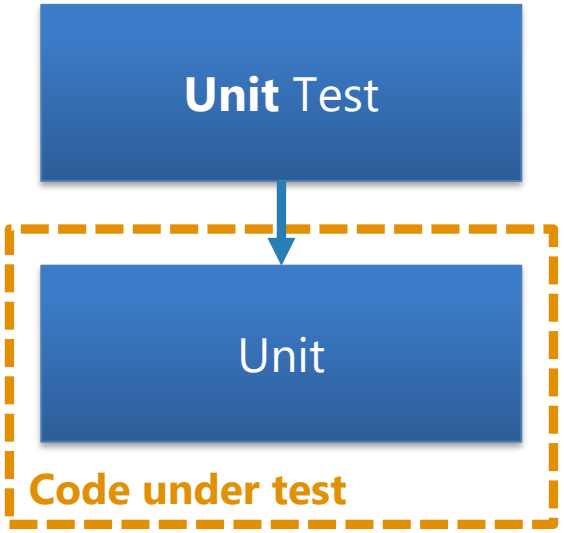
- Find defects early
- Avoid regressions
- Enable refactoring
- Provide living documentation (source code)
- Reduce manual testing effforts

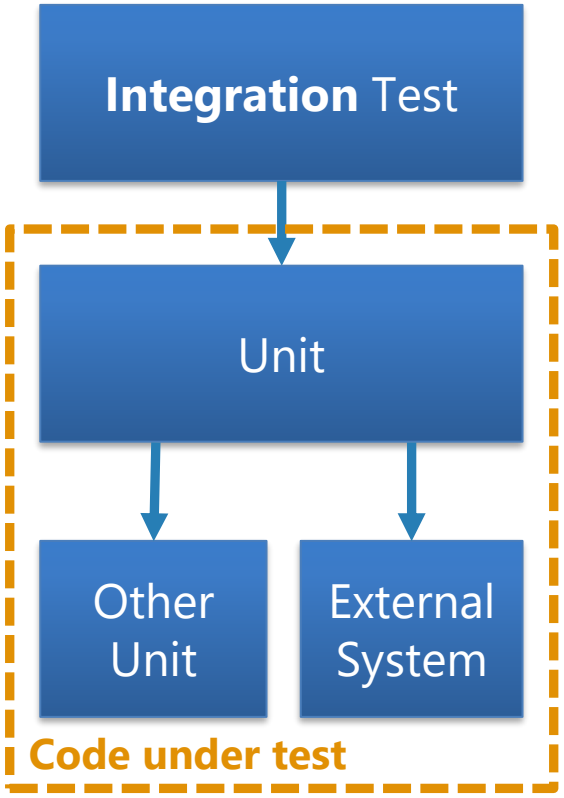
Requires tests to be as **complete** as possible
and to be **run as early and often** as possible

Unit Testing Framework



Unit Tests vs Integration Tests





Lesson Knowledge Check

1. What is a unit test?
2. What are the differences between unit and integration testing?
3. What are some of the benefits of unit tests?

Lesson Summary

- In this lesson, you learned about:
  - Unit Tests
  - Benefits of Unit Tests
  - Unit Testing Framework
  - Unit Tests vs Integration Tests

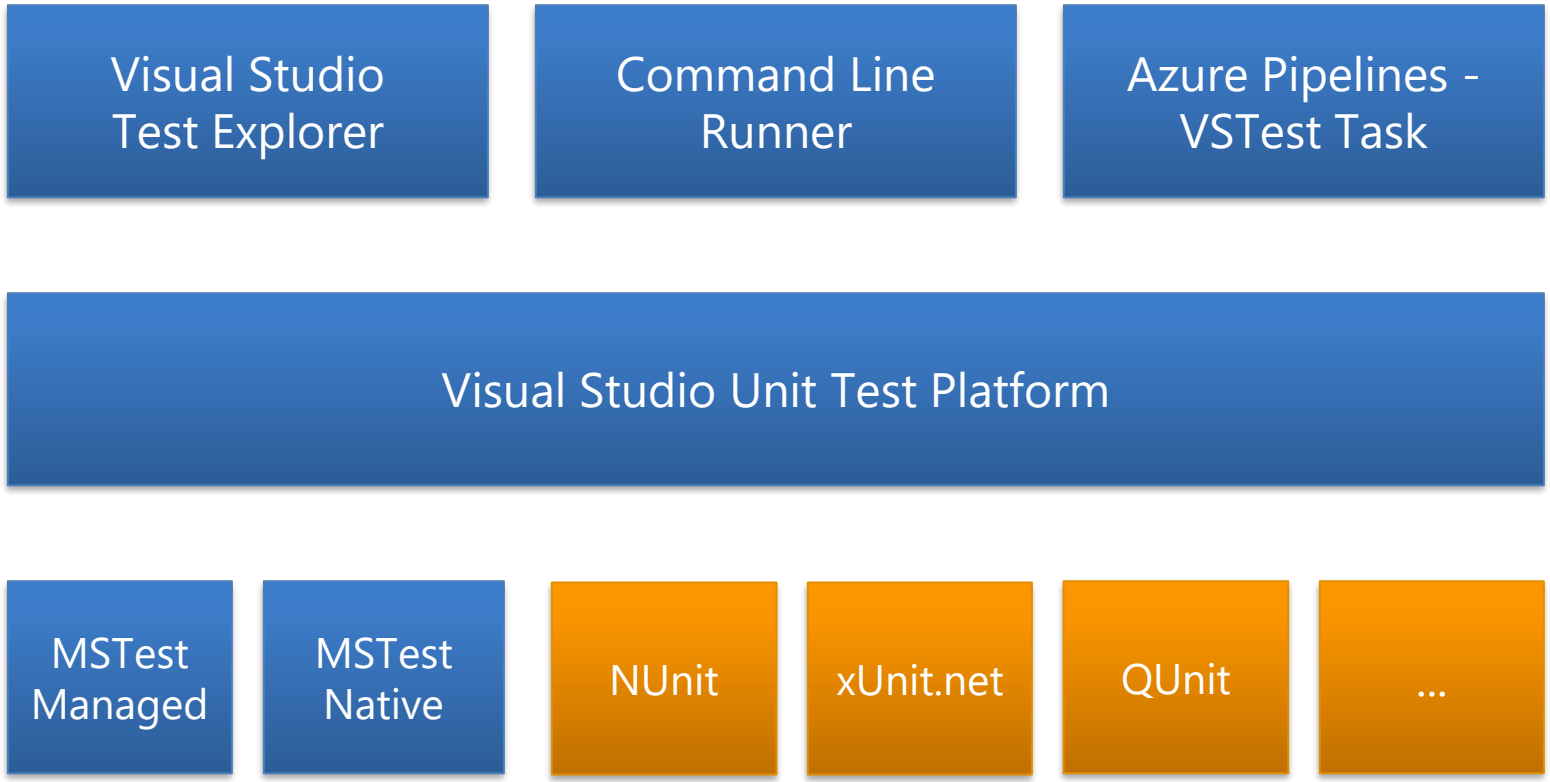
Module 4: Visual Studio Testing

Lesson 2: Unit Tests

Overview

- Visual Studio Unit Test Architecture
- Creating Unit Tests
- Test Explorer
- Test Categories
- Data-driven Unit Tests
- Running Unit Tests
- Live Unit Testing
- Code Coverage
- Run Settings

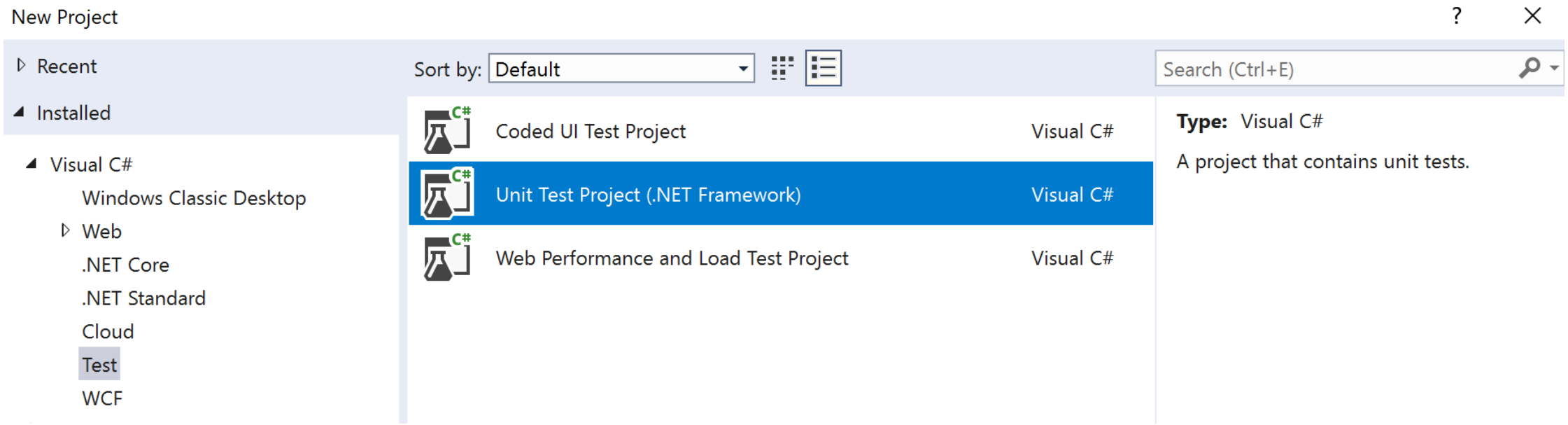
Visual Studio Unit Test Architecture



Creating Unit Test Project

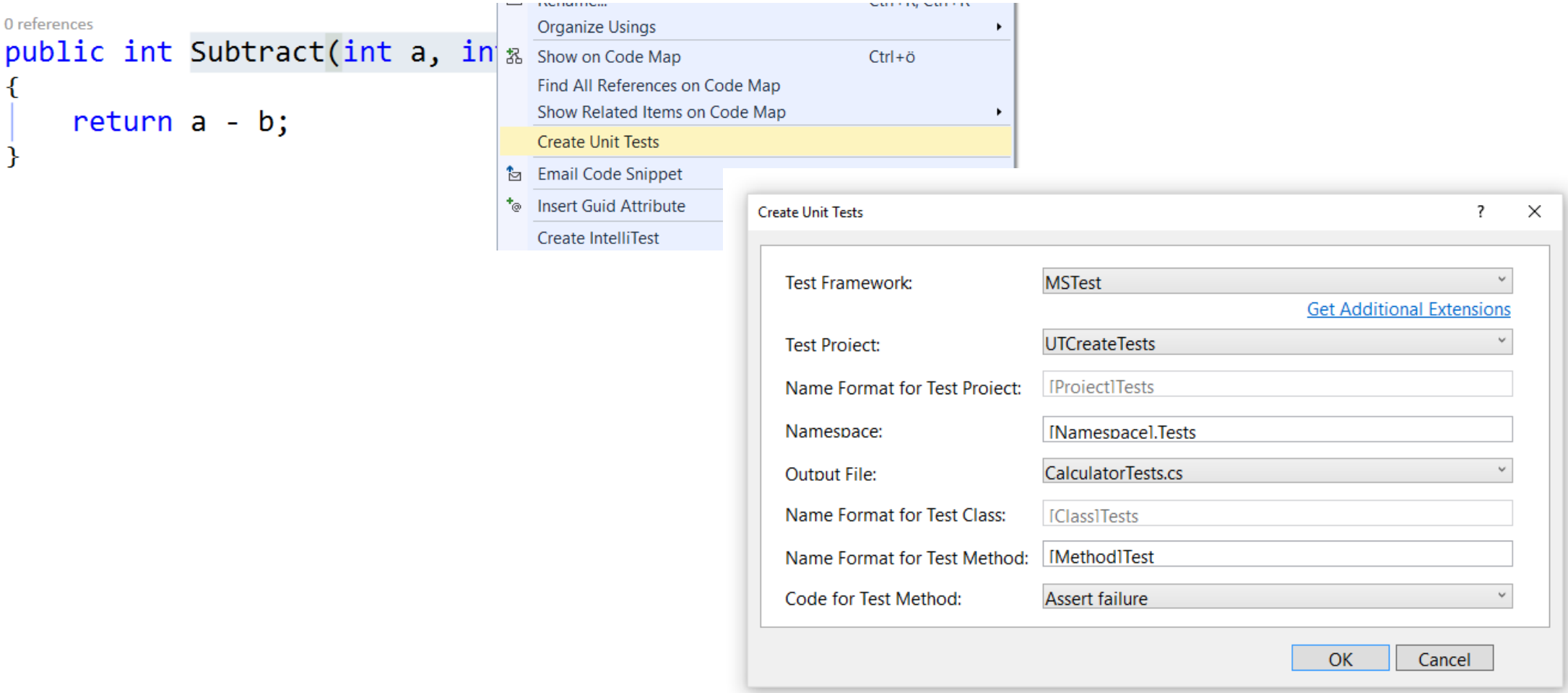
- Use the **Unit Test Project** template in Visual Studio.
- Add additional NuGet packages as needed.
  - Example: NUnit or xUnit
- From the Unit Test Project, add a project reference to the project containing the code to be tested.
- Attributes: TestClass, TestMethod, TestInitialize, TestCleanup

Creating Unit Test Project (continued)



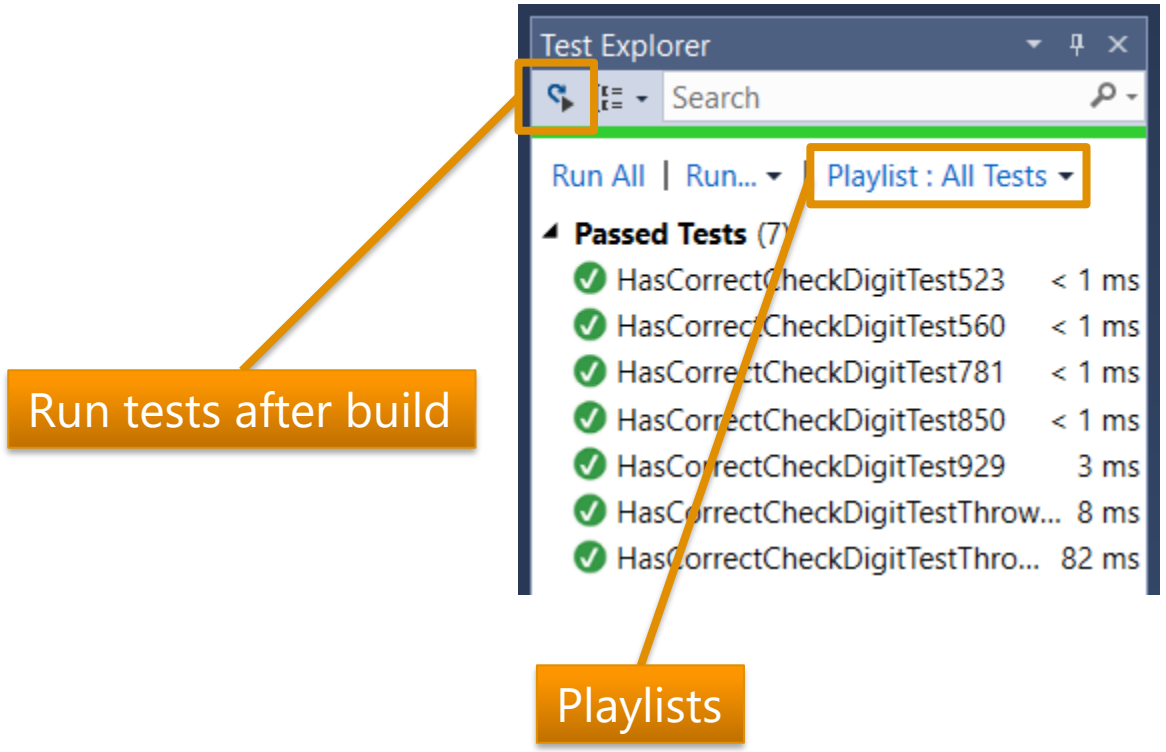
Creating Unit Tests from Existing Code

- Right click on a method to create a new test method.



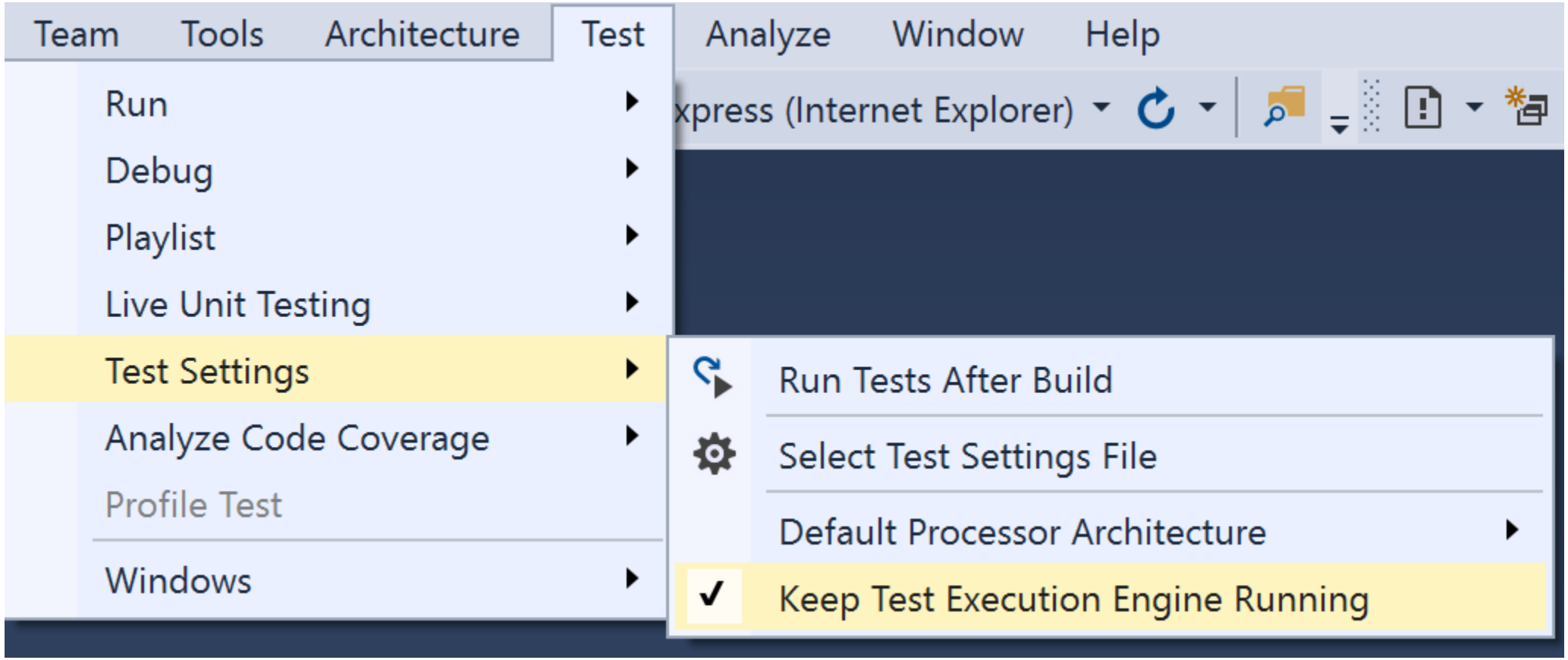
Test Explorer

- Use **Test Explorer** to run unit tests from Visual Studio or third-party unit test projects.
- Group tests into categories, filter the test list, and create, save, and run playlists of tests.
- Debug tests and analyze test performance and code coverage.



Keep Test Execution Engine Running

- Improve performance by keeping the engine running.

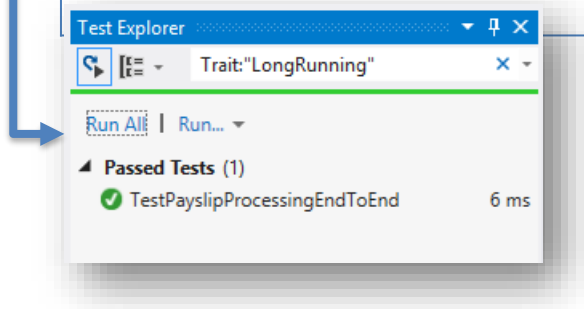


Test Categories

- Organize and filter tests using Test Categories.
- Attribute: **TestCategory**
- Multiple test categories can be applied to one test method.

Unit Tests

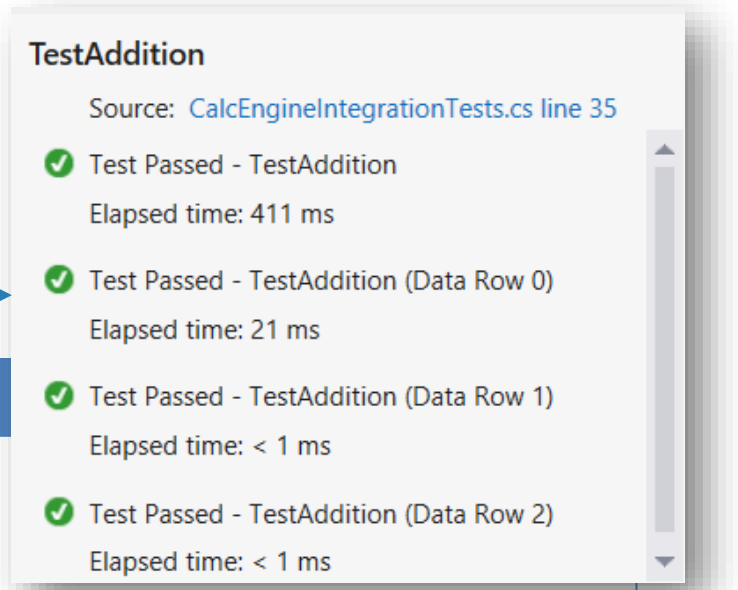
```
[TestMethod]
[TestCategory("LongRunning")]
[TestCategory("Integration")]
public void TestPayslipProcessingEndToEnd()
{
    ...
}
```



Filters in
Test Explorer

Data-driven Unit Tests

Value1	Value2	ExpectedResult
1	2	3
0	0	0
-5	10	5



Parameterized Unit Test

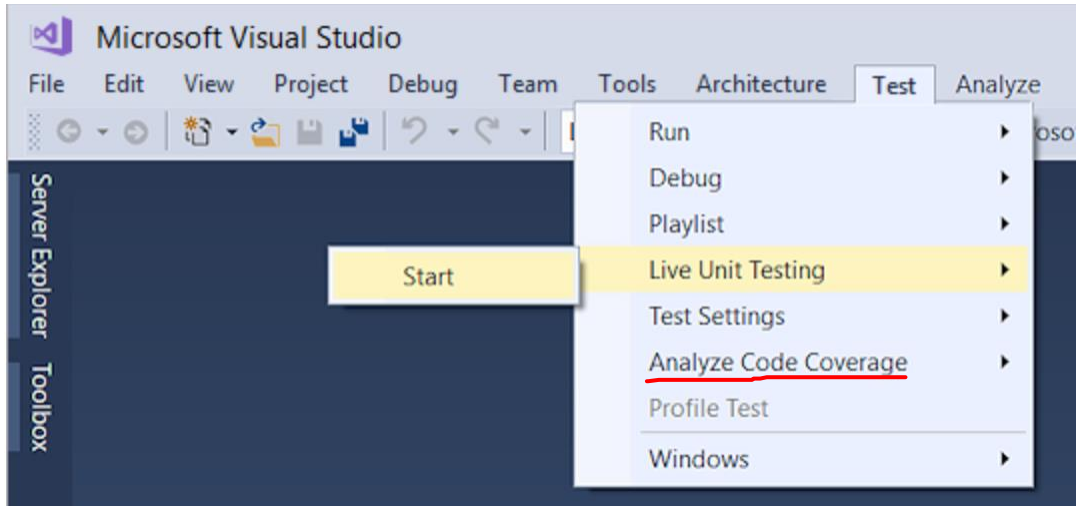
```
[TestMethod]
[DataSource(...)]
[DeploymentItem("Values.xml")]
public void TestAddition()
{
  var actual = Convert.ToInt32(TestContext.DataRow["Value1"])
             + Convert.ToInt32(TestContext.DataRow["Value2"]);
  var expected = Convert.ToInt32(TestContext.DataRow["ExpectedResult"]);
  Assert.AreEqual(expected, actual);
}
```

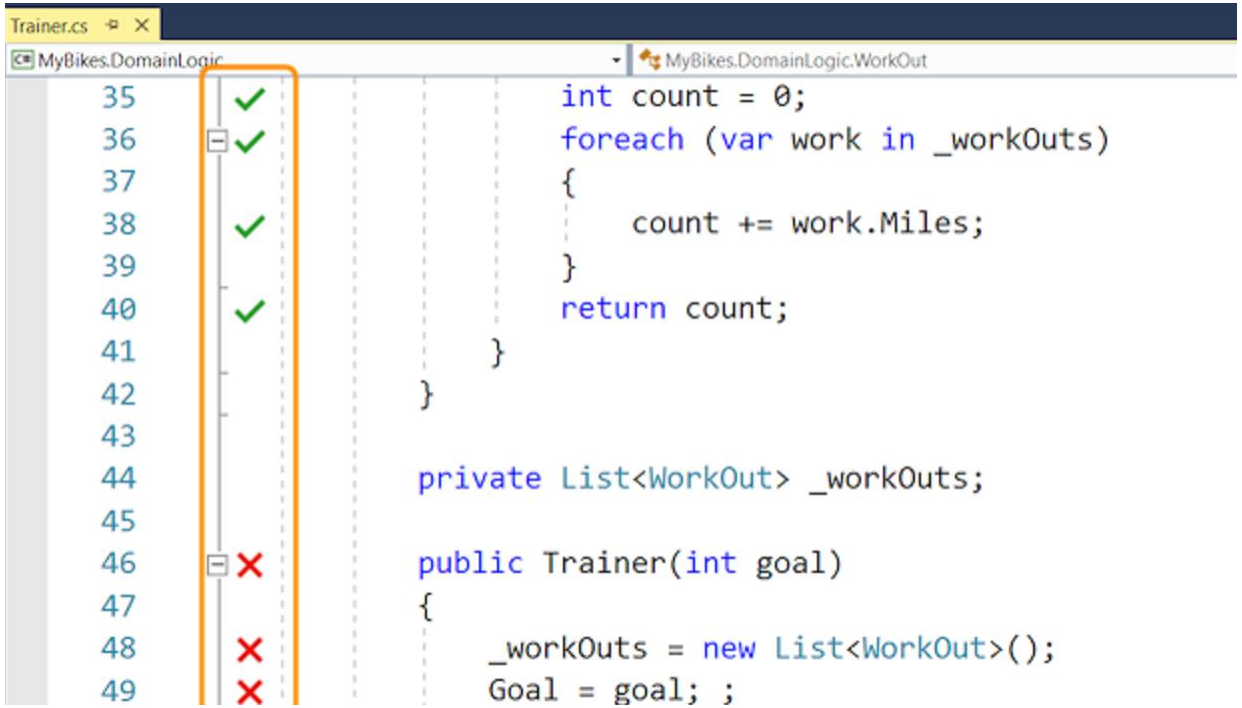
Running Unit Tests

- Using Test Explorer
  - Run all tests or selected test(s).
- Right click on the test method and click **Run Tests**
- Command line
- Azure Pipelines - Visual Studio Test task

Live Unit Testing

- If you are using the MSTest, xUnit, or NUnit testing framework in Visual Studio 2017 or later, you can see live results of your unit tests.
- Live unit testing is available in Enterprise edition only.

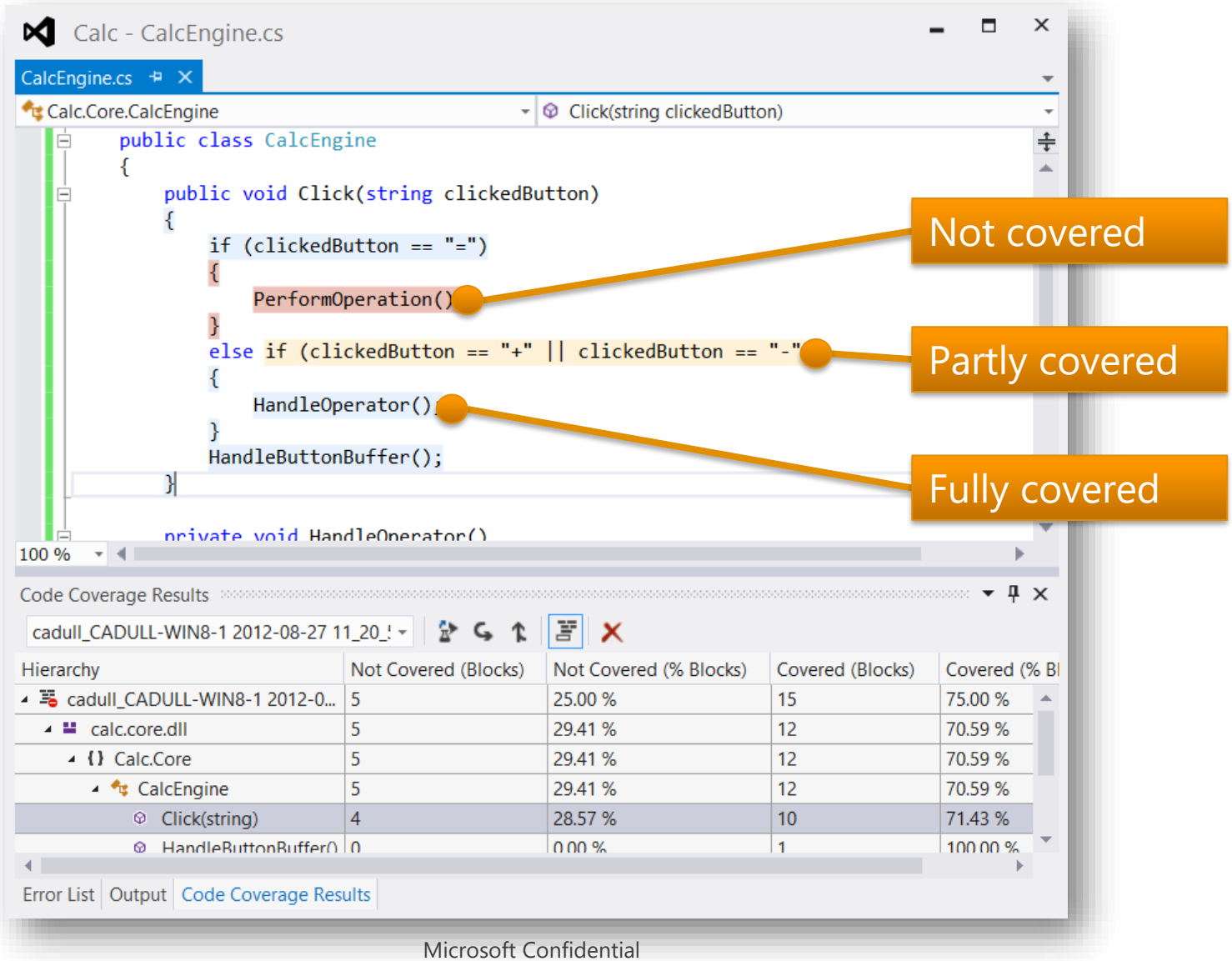




Code Coverage

- To determine what proportion of your project's code is actually being tested by coded tests such as unit tests, you can use the code coverage feature of Visual Studio.
- Code coverage is an option when you run test methods using Test Explorer.
- To guard effectively against bugs, your tests should exercise or 'cover' a large proportion of your code.
- Code coverage analysis can be applied to both managed (CLI) and unmanaged (native) code.
- The results table shows the percentage of the code that was run in each assembly, class, and method. In addition, the source editor shows you which code has been tested.

Code Coverage



Run Settings

- Run settings files can be used to configure tests that are run from the command line, in the IDE, or in a build workflow using Azure Test Plans.
- For example, you can change the .NET Framework version on which the tests are run, the directory for the test results, or the data that's collected during a test run.
- Run settings files are optional. If you don't require any special configuration, you don't need a *.runsettings* file. The most common use of a *.runsettings* file is to customize code coverage analysis.

Demo 1: Unit Tests

Lesson Knowledge Check

1. Name two ways to run a unit test.
2. What is Code Coverage?

Lesson Summary

- In this lesson, you learned about:
  - Visual Studio Unit Test Architecture
  - Creating Unit Tests
  - Test Explorer
  - Test Categories
  - Data-driven Unit Tests
  - Running Unit Tests
  - Live Unit Testing
  - Code Coverage
  - Run Settings

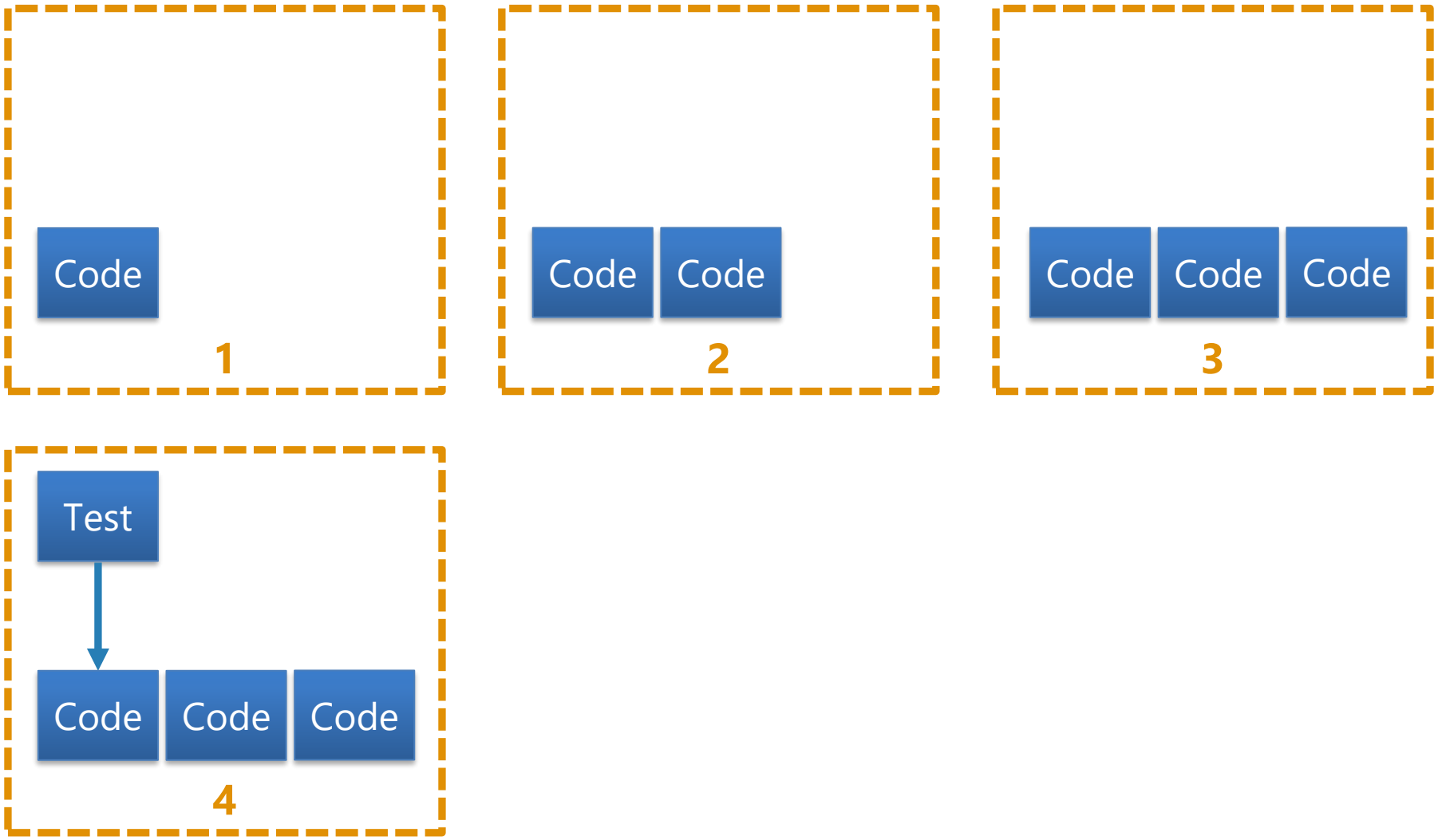
Module: Unit Testing

Lesson 3: TDD

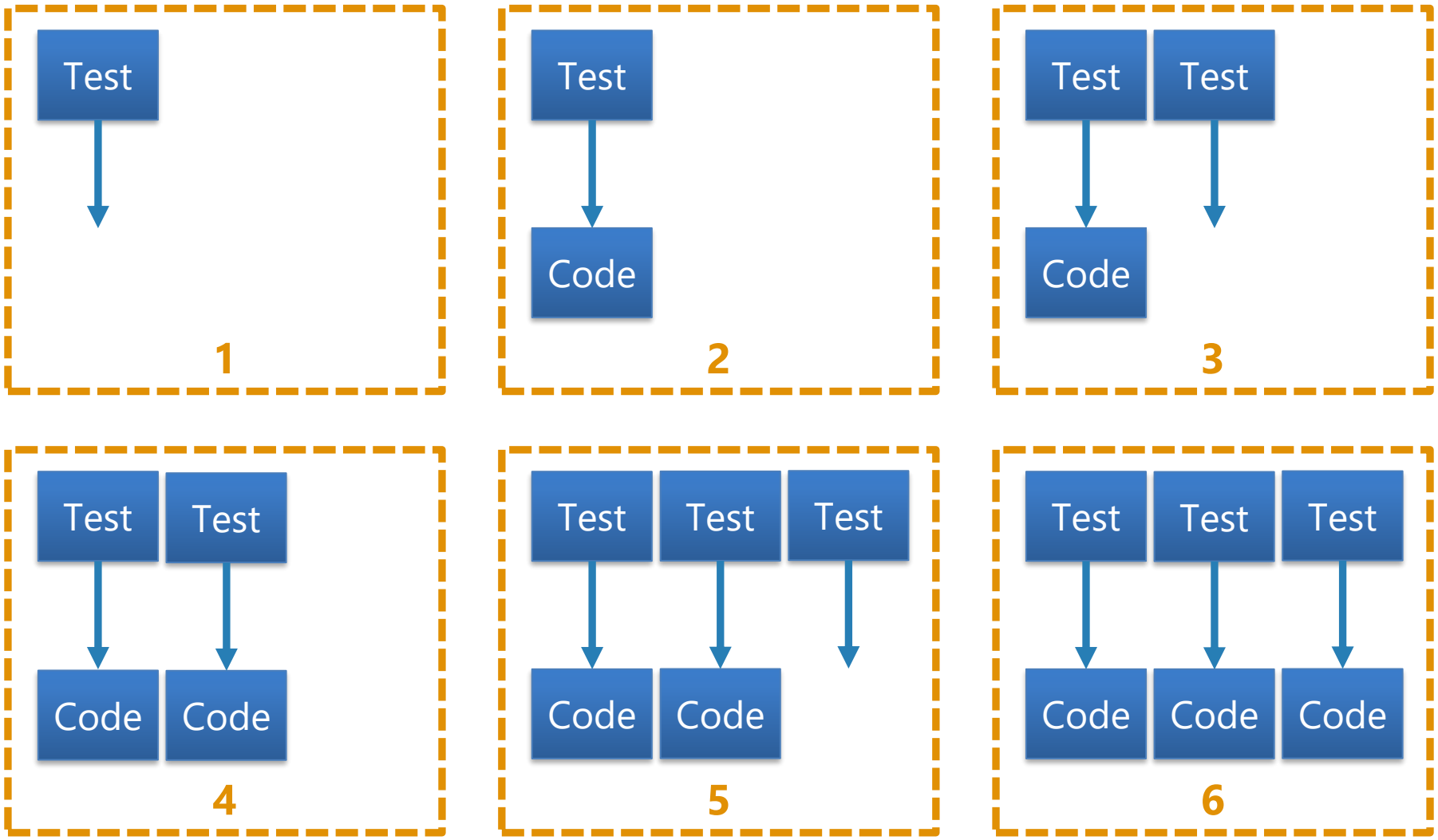
Overview

- Traditional Unit Testing Approach
- The TDD Approach
- TDD Benefits
- TDD Drawbacks
- The Red-Green-Refactor Cycle

Traditional Unit Testing Approach



The TDD Approach



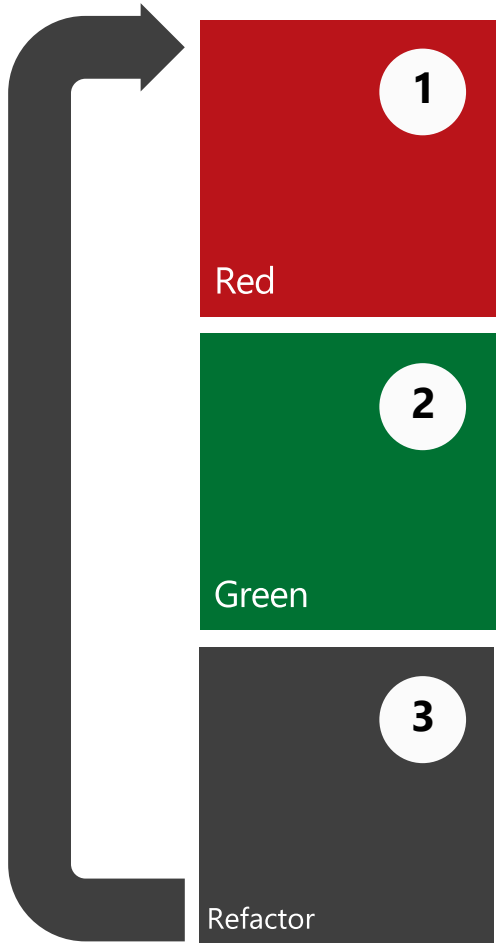
TDD Benefits

- Improves testability and design of production code
- Avoids untested code
- Prevents tests from being postponed or missed
- All production code has tests
- You write less code, only the code needed to make the tests pass

TDD Drawbacks

- May be seen as counterintuitive by developers
- May be hard to get management buy-in, takes time
- Problematic with legacy code

The Red-Green-Refactor Cycle



Create a test and make it fail

Make the best pass by simplest means necessary

Improve the code, improve the design

Demo 3: Creating a Method with TDD

Lesson Knowledge Check

- Describe TDD
- Does TDD have an influence on the design of production code?
- What is Red–Green–Refactor?

Lesson Summary

- In this lesson, you learned about:
  - Traditional Unit Testing Approach
  - The TDD Approach
  - TDD Benefits and Drawbacks
  - The Red-Green-Refactor Cycle

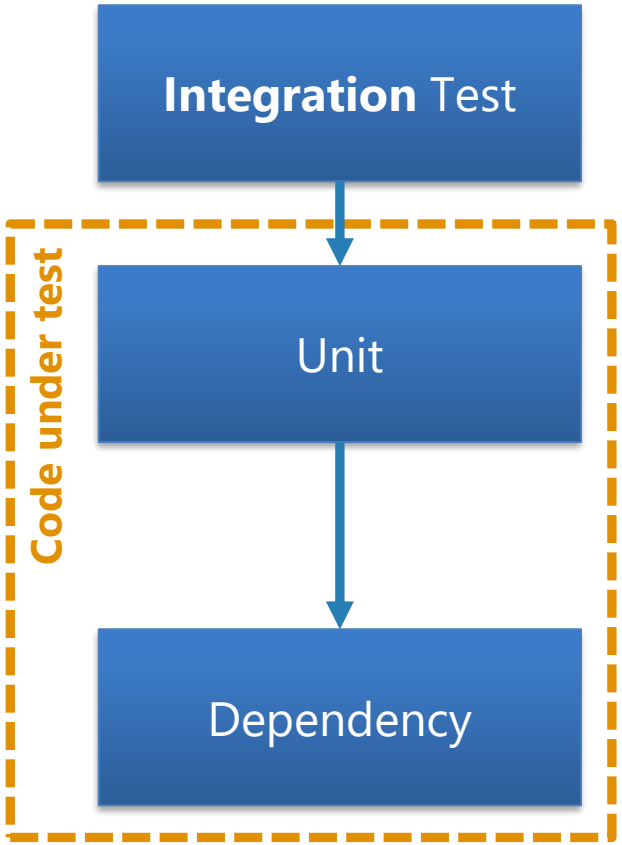
Module: Visual Studio Testing

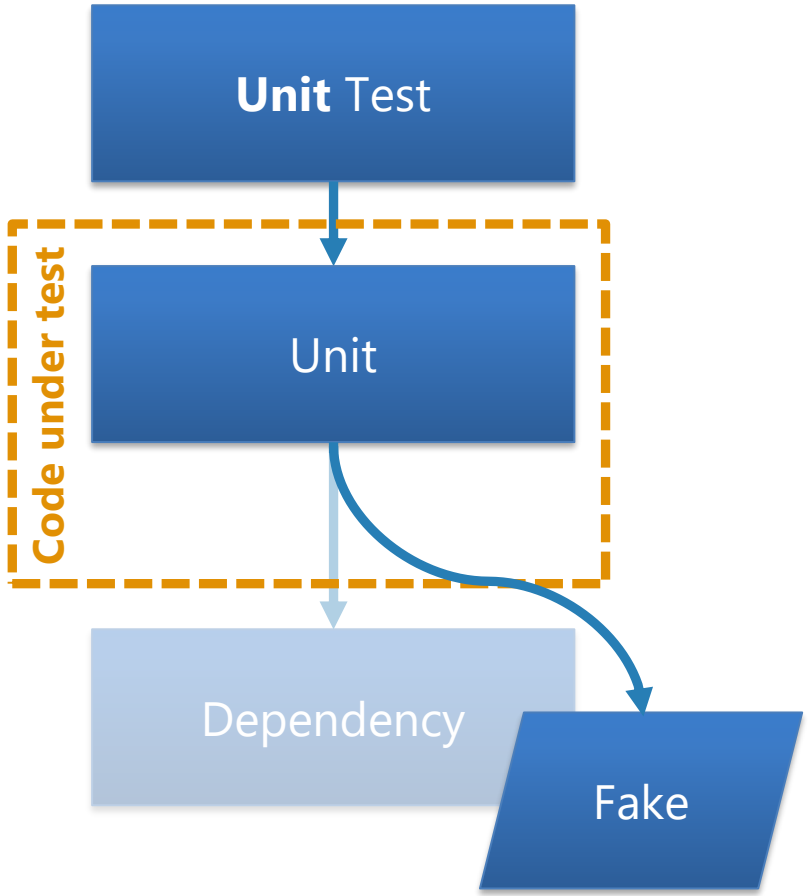
Lesson 4: Isolating Code with Fakes

Overview

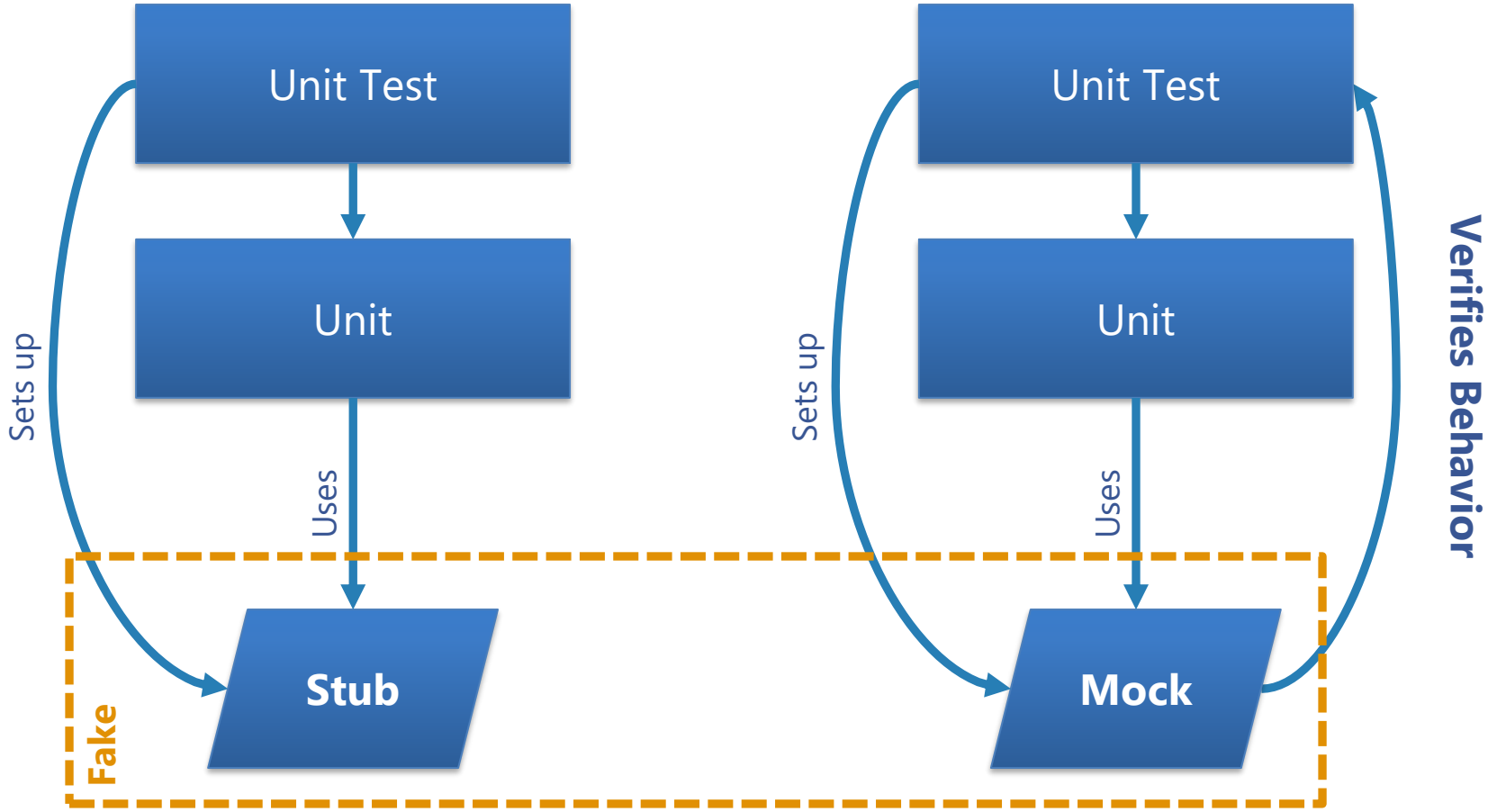
- Breaking Dependencies
- Stubs Vs. Mocks
- Using Microsoft Fakes to Create Stubs
- Using Microsoft Fakes to Create Shims

Breaking Dependencies

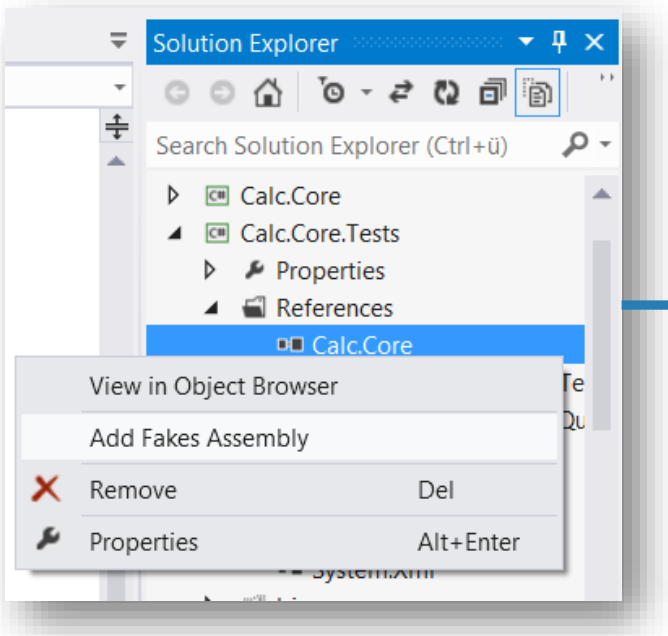




Stubs vs. Mocks



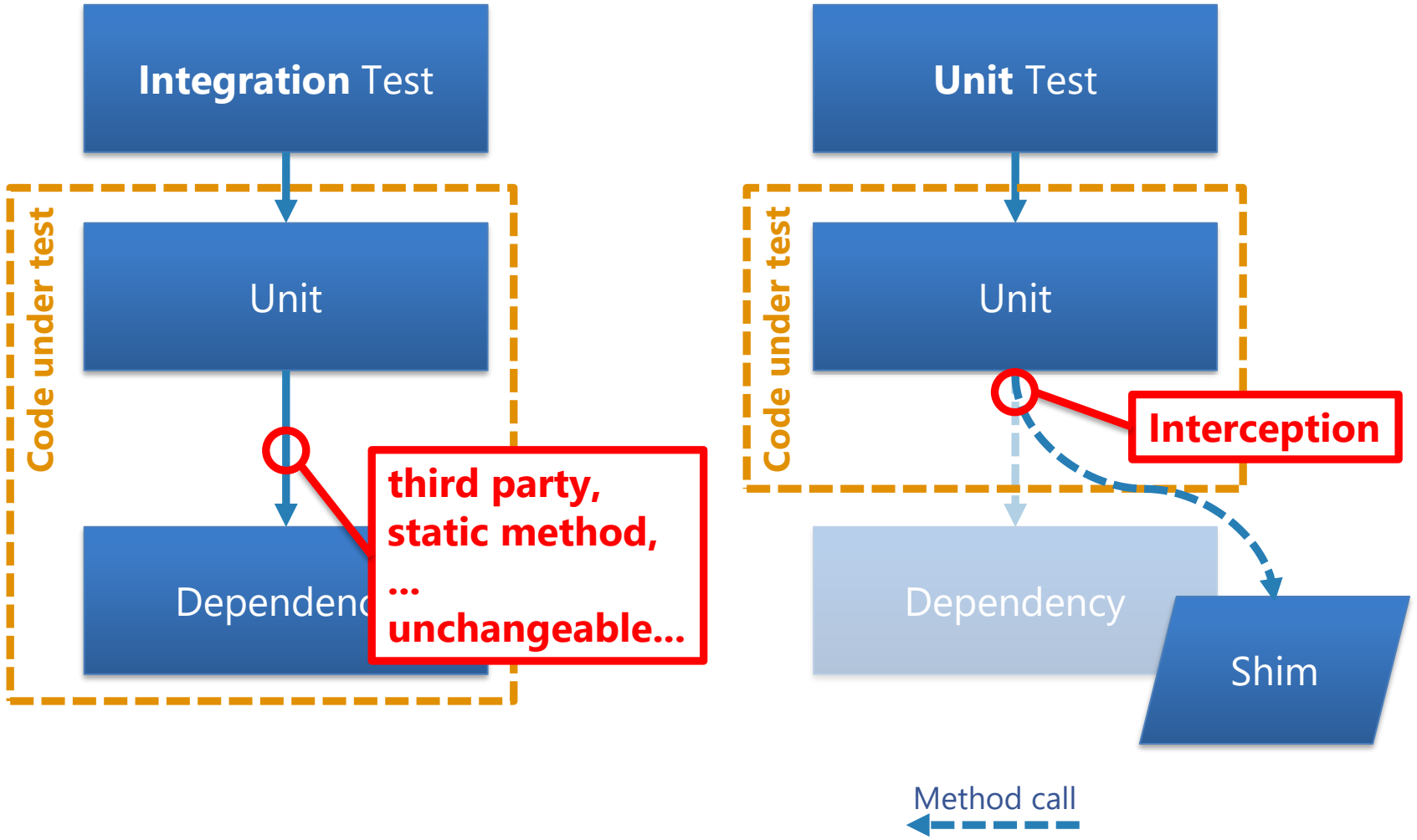
Using Microsoft Fakes to Create Stubs



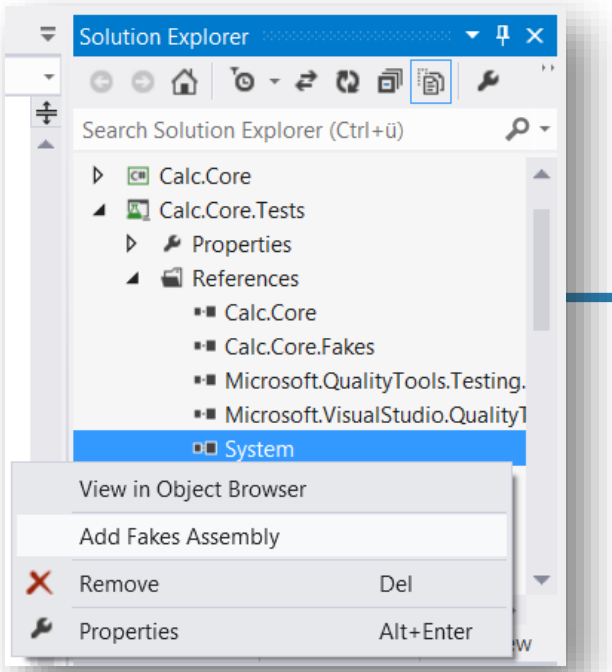
Unit Test with Stub

```
[TestMethod]
public void Click_RWithValue42InStorage_TextIs42()
{
    var storageStub = new StubIStorage();
    storageStub.ReadValue = () => { return 42; };
    var calcEngine = new CalcEngine();
    calcEngine.Storage = storageStub;
    TestUtilities.ClickSequence(calcEngine, "R");
    Assert.AreEqual("42", calcEngine.Text);
}
```

What If Dependencies Cannot be Replaced?



Using Microsoft Fakes to Create Shims



Unit Test with Shim

```
[TestMethod]
public void Click_OnSundays_DoNotWork()
{
    using (ShimsContext.Create())
    {
        ShimDateTime.NowGet =
            () => new DateTime(2012, 09, 16);
        var calcEngine = new CalcEngine();
        calcEngine.Click("1");
        var expected = "Service unavailable on Sundays.";
        Assert.AreEqual(expected, calcEngine.Text);
    }
}
```

Demo 2: Using Microsoft Fakes

Lesson Knowledge Check

1. What is Microsoft Fakes?
2. Does a stub verify behavior of the code under test?
3. True/False: Stubs can be injected into any software system

Lesson Summary

- In this lesson, you learned about:
  - Breaking Dependencies
  - Stubs vs. Mocks
  - Using Microsoft Fakes to Create Stubs
  - Using Microsoft Fakes to Create Shims

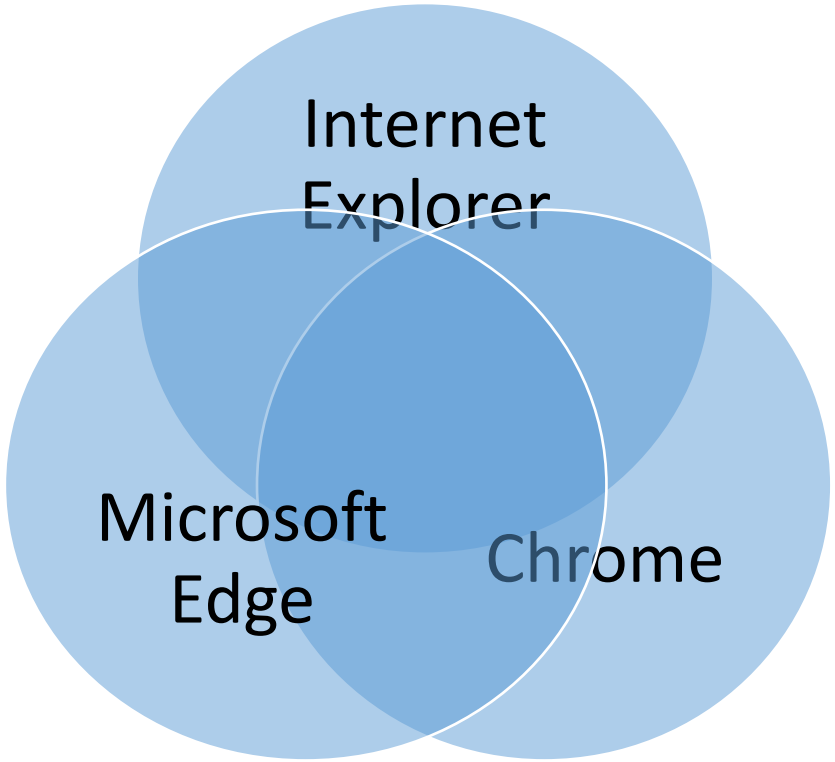
Module: Visual Studio Testing

Lesson 5: Selenium

Overview

- Why Cross-Browser Testing?
- What is Selenium?
- Selenium Project
  - Selenium Server
  - Selenium Integrated Development Environment (IDE)
  - Selenium WebDriver
- Perform UI Tests With Selenium

Why Cross-Browser Testing ?



Web Standard Implementations

What is Selenium?

Selenium automates browsers. That's it!

Selenium Project

- Selenium Server
- Selenium WebDriver
- Selenium Integrated Development Environment (IDE)

Selenium Server

- Selenium-Grid
- Enables tests to be distributed across machines
- Allows you to connect to a remote machine that has a specific browser version not on your current machine.
- May, or may not, be required, depending on how Selenium WebDriver will be used.
    - If your browser and tests will all run on the same machine, and your tests only use the WebDriver API, then you do not need to run the Selenium-Server; WebDriver will run the browser directly.

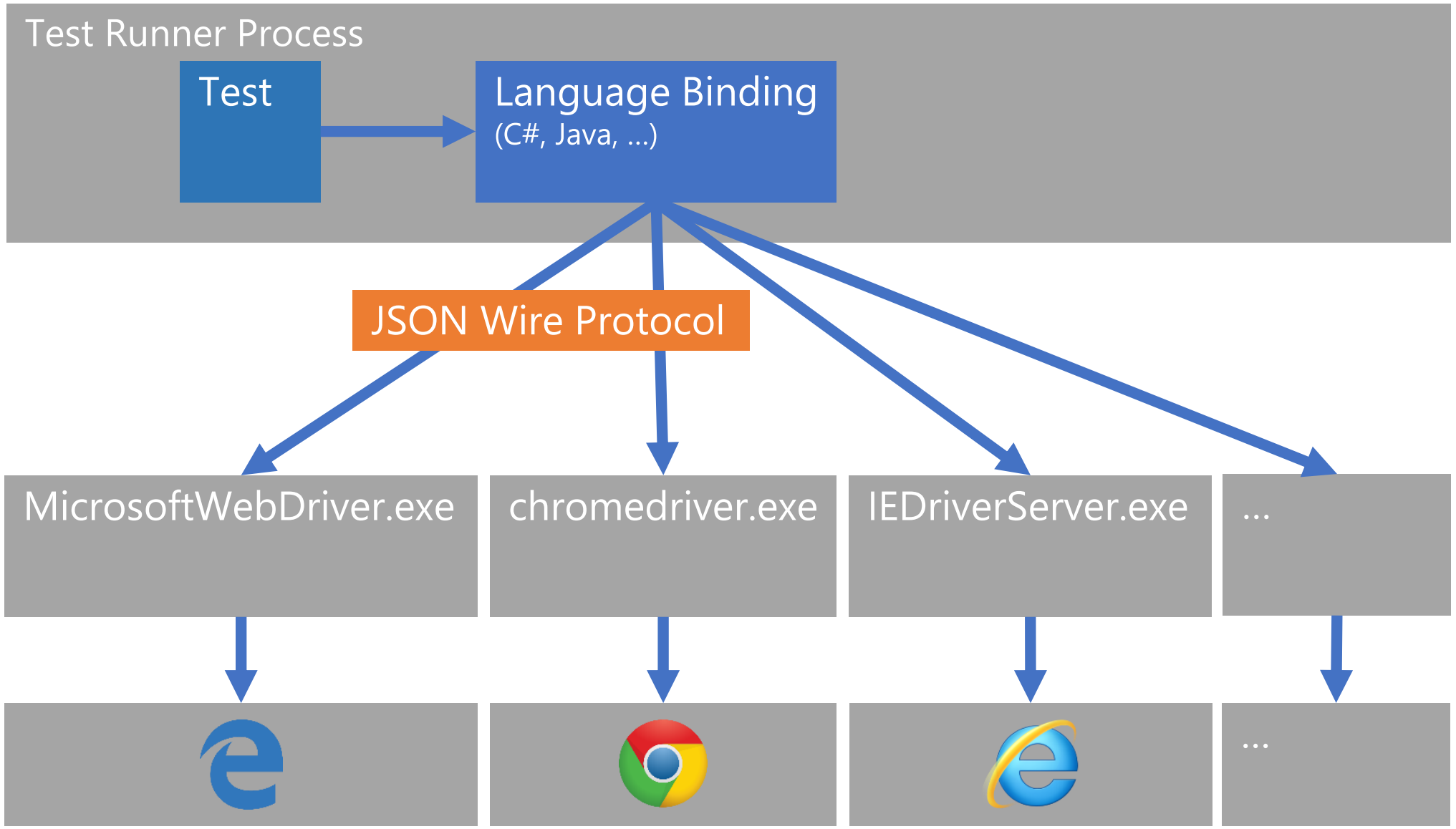
Selenium WebDriver

- Collection of APIs used to automate testing across different browsers.
- Makes direct calls to the browser using each browser's native support for automation.
- Allows you to create robust, *browser-based* regression automation suites and tests.
- Browser implementation specific, same API may behave differently for different browsers
- Works for both Java and .NET (API may be slightly different)

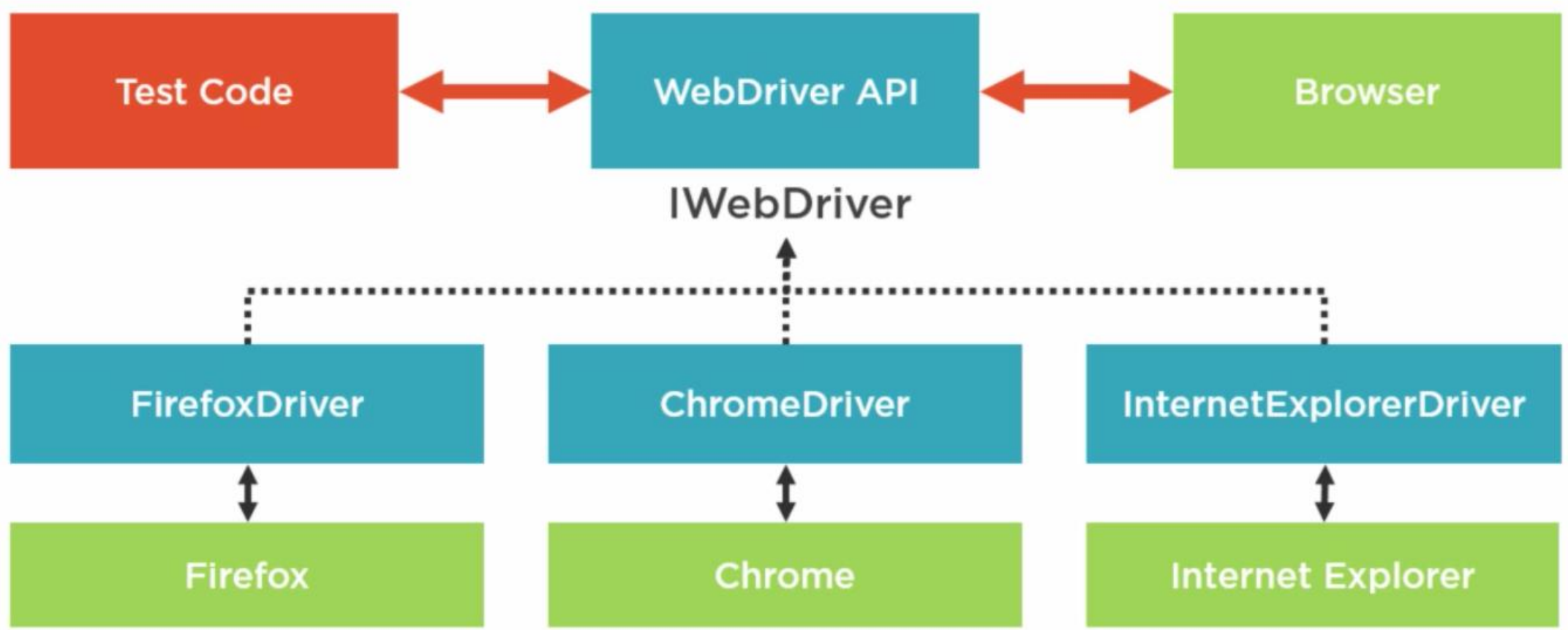
Selenium IDE

- Google Chrome and Mozilla Firefox add-on that will perform simple record-and-playback of interactions with the browser.
- Provides easy-to-use interface for developing automated tests.

Selenium WebDriver Architecture



Selenium WebDriver



Perform UI Tests With Selenium

- Create a Unit Test Project
- Add Selenium NuGet packages:
  - Selenium.WebDriver
  - Selenium.Support
  - Selenium.WebDriver.ChromeDriver
  - Selenium.Firefox.WebDriver
  - Selenium.WebDriver.IEDriver

Sample Selenium Code

```
using (IWebDriver driver = new FirefoxDriver())
{
    //Notice navigation is slightly different than the Java version
    //This is because 'get' is a keyword in C#
    driver.Navigate().GoToUrl("http://www.google.com/");

    // Find the text input element by its name
    IWebElement query = driver.FindElement(By.Name("q"));

    // Enter something to search for
    query.SendKeys("Cheese");

    // Now submit the form. WebDriver will find the form for us from the element
    query.Submit();

    // Google's search is rendered dynamically with JavaScript.
    // Wait for the page to load, timeout after 10 seconds
    var wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
    wait.Until(d => d.Title.StartsWith("cheese", StringComparison.OrdinalIgnoreCase));

    // Should see: "Cheese - Google Search" (for an English locale)
    Console.WriteLine("Page title is: " + driver.Title);
}
```

Demo 3: Selenium

Lesson Knowledge Check

1. What is Selenium?
2. What is Selenium WebDriver?
3. How would you add Selenium support to a test project?

Lesson Summary

- In this lesson, you learned about:
  - Why cross-browser testing is important.
  - Selenium
  - Selenium Project
    - Selenium Server
    - Selenium Integrated Development Environment (IDE)
    - Selenium WebDriver
  - Performing UI Tests With Selenium

Module: Visual Studio Testing

Lesson 6: Page Object Pattern

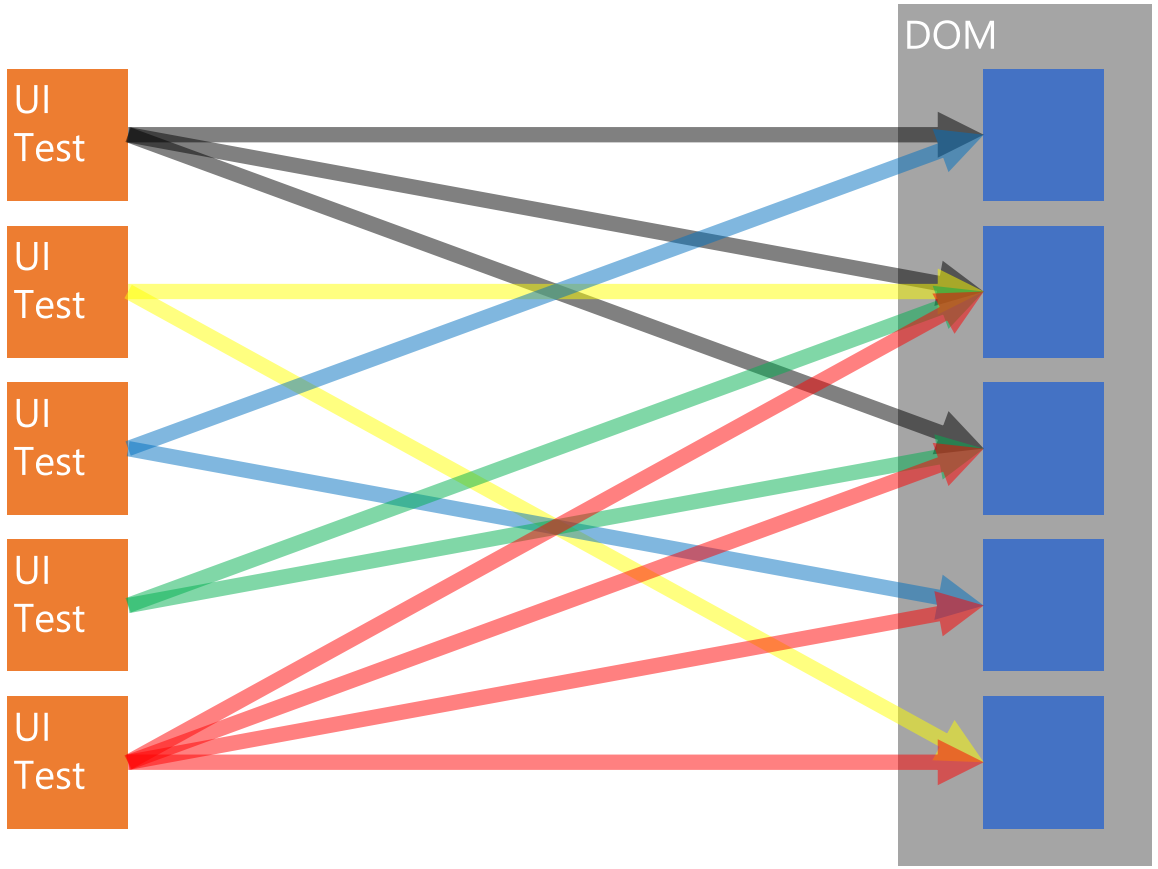
Overview

- Page Object Pattern

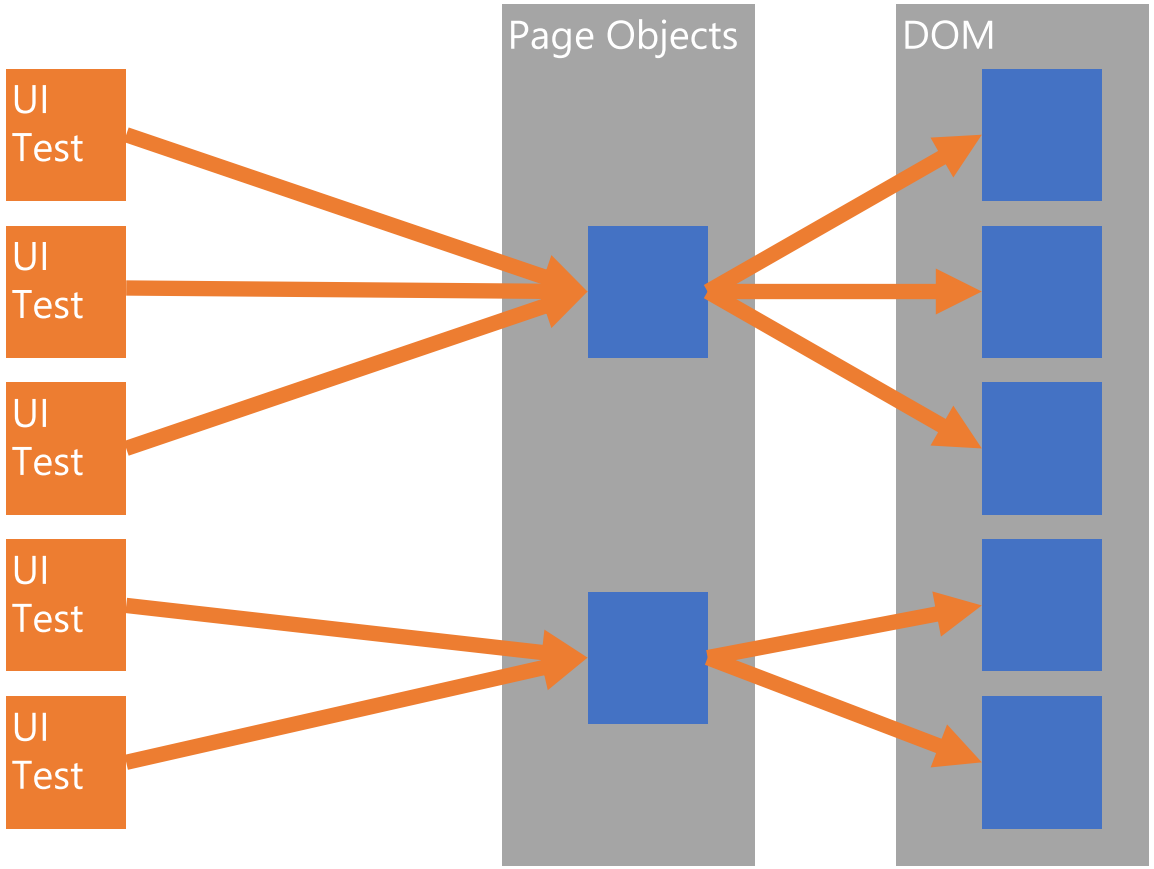
Page Object Pattern

- A page object wraps an HTML page, or fragment, with an application-specific API
- Manipulates page elements without digging around in the HTML.
  - Header object
  - Footer object
  - Login
  - …
- Allows a software client to do anything and see anything that a human can
- Encapsulates the mechanics required to find and manipulate the data in the GUI control itself
- If you navigate to another page, the initial page object should return another page object for the new page

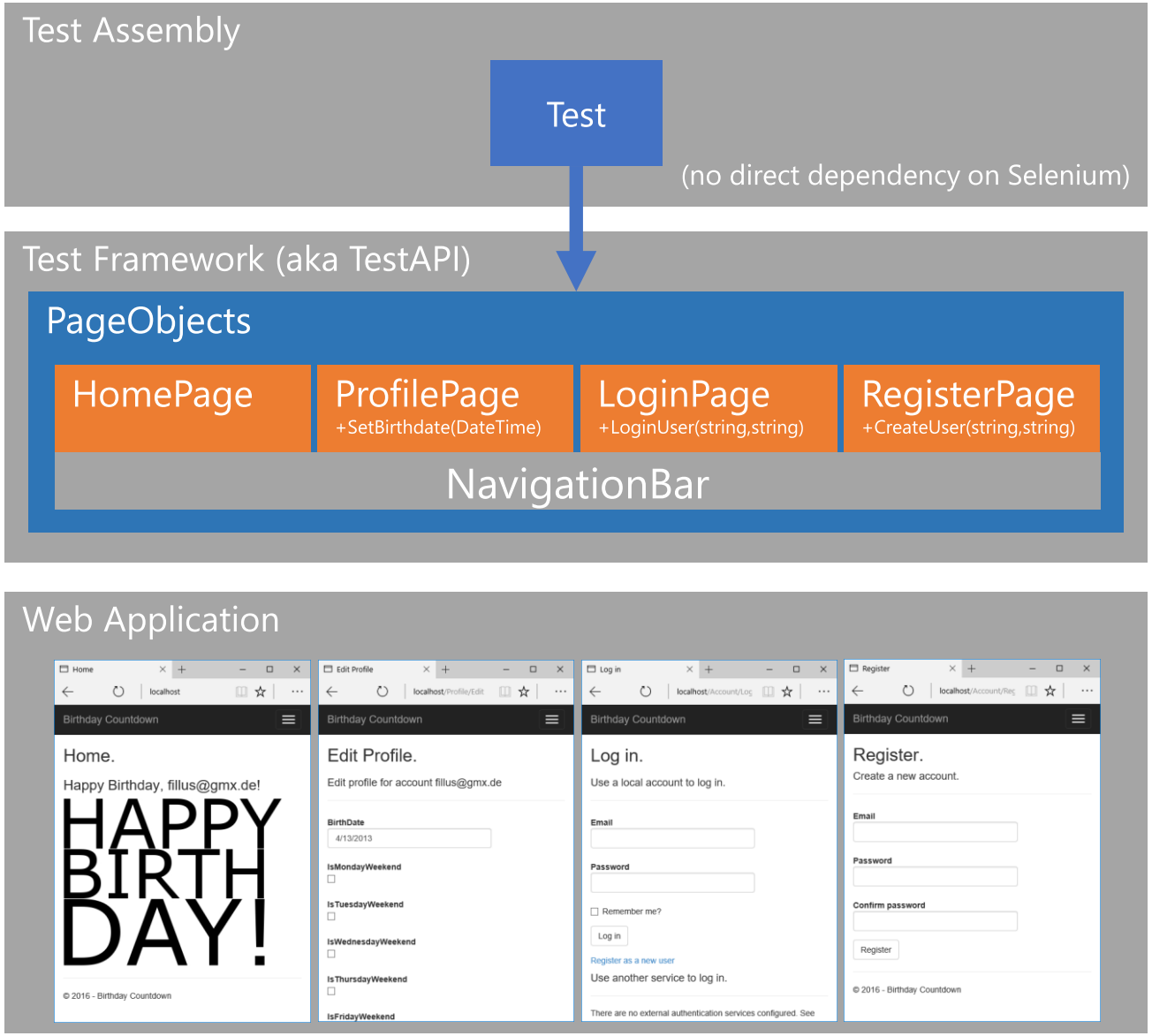
Page Object Pattern



Page Object Pattern



Page Object Pattern



Lesson Knowledge Check

1. Advantages of Page Object Pattern?

2. Could Page Objects share elements?

3. Should I use asserts with Page Object Pattern?

Module 4: Visual Studio Testing

Lesson 5: Appium

Overview

- What is Appium?
- Appium Concepts
- WebDriver
- Driver-Specific Setup
- WinAppDriver (WAD)

What is Appium?

- Open-source tool for automating native, mobile web, and hybrid applications on iOS mobile, Android mobile, and Windows desktop platforms.
- Appium is "cross-platform": it allows you to write tests against multiple platforms (iOS, Android, Windows), using the same API.
- Enables code reuse between iOS, Android, and Windows test suites.

Appium Concepts

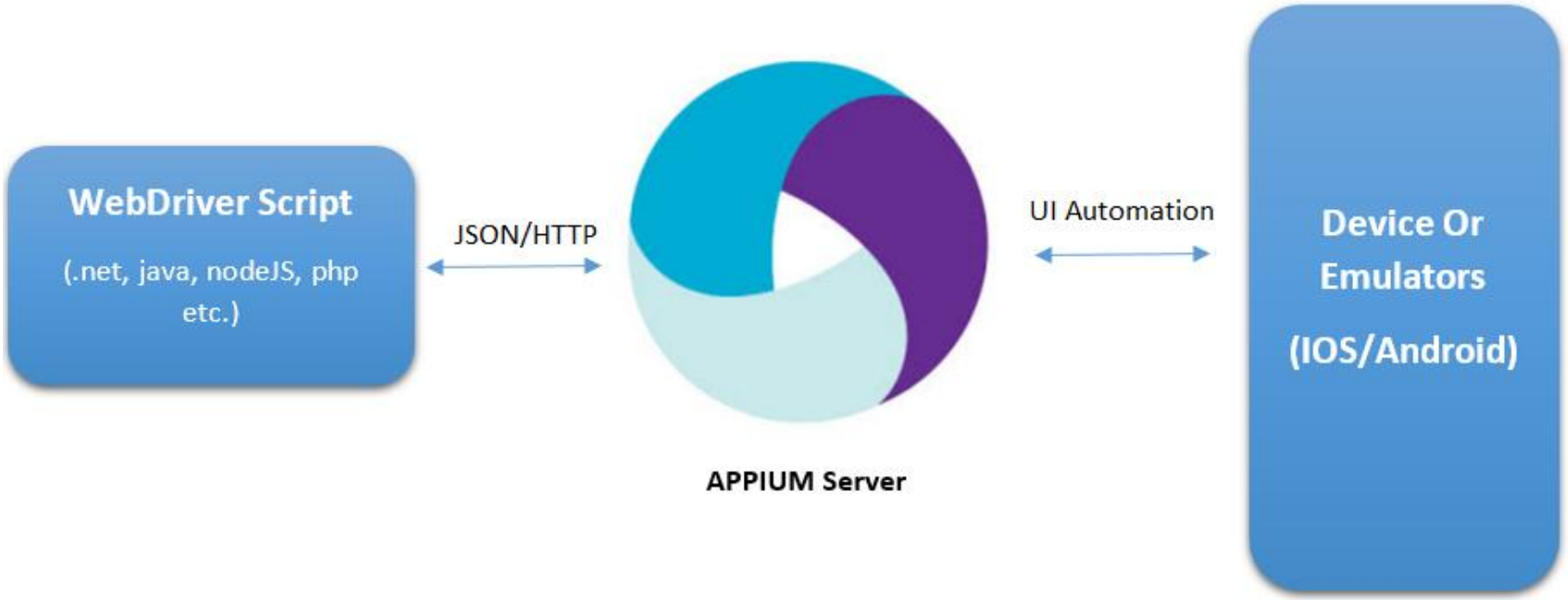
- Client/Server Architecture
  - Appium is a webserver that exposes a REST API.
  - It receives connections from a client, listens for commands, executes those commands on a mobile device, and responds with an HTTP response representing the result of the command execution.
- Session
  - Automation is always performed in the context of a session.
  - Clients initiate a session with a server in ways specific to each library, but they all end up sending a POST session request to the server, with a JSON object called the 'desired capabilities' object. At this point the server will start up the automation session and respond with a session ID which is used for sending further commands.
- Desired Capabilities
  - Desired capabilities are a set of keys and values (i.e., a map or hash) sent to the Appium server to tell the server what kind of automation session is being set up.

Appium Concepts (continued)

- Appium Server
  - Appium is a server written in Node.js.
  - It can be built and installed from source or installed directly from NPM.
- Appium Clients
  - There are client libraries (in Java, Ruby, Python, PHP, JavaScript, and C#) which support Appium's extensions to the WebDriver protocol.
  - When using Appium, use these client libraries instead of the regular WebDriver client.
- Appium Desktop
  - There is a GUI wrapper around the Appium server that can be downloaded for any platform. It comes bundled with everything required to run the Appium server.
- More info can be found here: http://appium.io/docs/en/about-appium/intro/

WebDriver

- Concept similar to Selenium.
- Test app uses WebDriver API to communicate with Appium Server using HTTP/JSON which in turn sends/receives commands to/from devices/emulators.
- For a .NET test project, add Appium support using NuGet.



Driver-Specific Setup

- Support for the automation of a particular platform is provided by an Appium "driver".
  - The XCUITest Driver (for iOS apps)
  - The UiAutomator2 Driver (for Android apps)
  - The Windows Driver - WinAppDriver (for Windows Desktop apps)
  - The Mac Driver (for Mac Desktop apps)
  - (BETA) The Espresso Driver (for Android apps)

WinAppDriver (WAD)

- Appium has the ability to automate Windows PC Desktop apps.
- WinAppDriver is an Appium-compatible WebDriver server for Windows Desktop apps (and more in the future).
- WinAppDriver is often abbreviated "WAD". WAD is bundled with Appium and does not need to be installed separately.
- Supports testing of **Universal Windows Platform (UWP)** and **Classic Windows (Win32)** applications.
- Works only on Windows 10 or up.
- To test a Windows app, make sure you have turned developer mode on.

Lesson Knowledge Check

1. What is Appium?
2. How does the test script communicate with Appium Server?

Lesson Summary

- In this lesson, you learned about:
  - What is Appium?
  - Appium Concepts
  - WebDriver
  - Driver-Specific Setup
  - WinAppDriver (WAD)

Demo 4: Appium

References

- SeleniumHQ - https://www.seleniumhq.org
- Appium - http://appium.io

Module Summary

- In this module, you learned about:
  - Unit Testing Fundamentals
  - Unit Tests
  - Isolating Code with Fakes
  - Selenium
  - Appium

Lab: Visual Studio Testing