Unit Testing

1. Setting up a Test Project
2. How to run
    a. Shortcut Keys
        i. Run All Tests - Control + R, A
        ii. Repeat Last Run - Control + R, L
        iii. Debug Last Run - Control + R, D
3. What are assertions
    a. Verifies a condition in your test code
    b. Types in MSTest
        i. Assert.AreEqual (Test the equivalence method =)
        ii. Assert.AreNotEqual
        iii. Assert.AreSame (Test if references the same instance)
        iv. Assert.AreNotSame
4. General Structure of a Unit Test
    a. Given, When, Then (Alternate: Arrange, Act, Assert)
    b. Given - Set up the test
    c. When - Test the condition
    d. Then - Assert
5. Sample unit test
    a. Sum
    b. Formica String function
6. Unit Test Naming
    a. Anytime you're using the word "And", it could indicate that you should split up the test.
    b. UnitOfWork_StateUnderTest_ExpectedBehavior
    c. Test [MethodOrClass]_When [Condition]_Should [ExpectedBehavior]
    d. https://osherove.com/blog/2005/4/3/naming-standards-for-unit-tests.html
7. Setup(), TearDown()
8. SUT - Subject Under Test
9. Code Coverage
    a. The percentage of your code that's covered by unit test.
        i. How it works: when unit tests run, it collects the lines of code that gets executed.
    b. Code coverage requires Visual Studio Enterprise
    c. There's also an open source project called OpenCover.

Unit Test Tips

1. A test should be small and focused. Test only one element of behavior.
    a. Some say to have only one assertion in a test, but I don't want to make that a hard and fast rule.

2. Treat the subject under test as a black box.
    a. Be sure to verify behavior, not implementation.
    b. Don't test private methods
    c. Test should not know any implementation detail.
3. Assertion Framework
4. Treat unit test code like production code.
    a. Keep it clean
    b. It serves also as documentation
5. Tests should be fast!
    a. Shouldn't be relying on anything external to the test
        i. making database calls
        ii. making network requests
    b. Don't test code you don't own
        i. Don't test Microsoft's frameworks
        ii. Don't test third party libraries

Dependencies

1. What is a dependency
2. Dependency injection
    a. Inversion of Control
3. Types of Dependency Injection
    a. Setter
    b. Constructor
    c. Method
4. Frameworks
    a. Unity
    b. Ninject
5. Dsfsdf

Dependency Injection Tips

1. Inheritance is frown upon
    a. If you're class is dependent on external framework, it's hard to test
    b. Composition over inheritance
        i. Idea that classes should be composed of instances of other classes rather than inheriting from a base class.
    c. Instances are injected into the class (sut)
2. Inject Interfaces
    a. Defines a contract, not an implementation
3.

Test Driven Development

1. Red, Green, Refactor
2. Red - Write a failing test

3. Green - Write the simplest possible
4. Refactoring - MOST IMPORTANT STEP in TDD
5. Kata
    a. PrimeFactors.generate() -> takes in an integer and returns a list of prime factors
    b. http://butunclebob.com/ArticleS.UncleBob.ThePrimeFactorsKata

Test Doubles Vocab

1. Two types of doubles
    1. Mocks
        a. Verify a method gets called
    2. Stubs
        a. Returns dummy data
    3. Mocks vs Stubs
        a. A mock verifies that a method (or some other event occurs in a dependency)
        b. A stub returns canned answers
        c. A test double can be both a mock and a stub.

Mocking

1. Create your own
2. Mocking framework
    a. Moq

Advanced Mocking / Stubbing

1. Http Mocking
2. Database Mocking
    a. ADO.NET - Sqlite in memory
    b. bluecloudsoft - unit tests
3. Testing Controllers

Unit Testing with JavaScript

1. Jasmine
2.