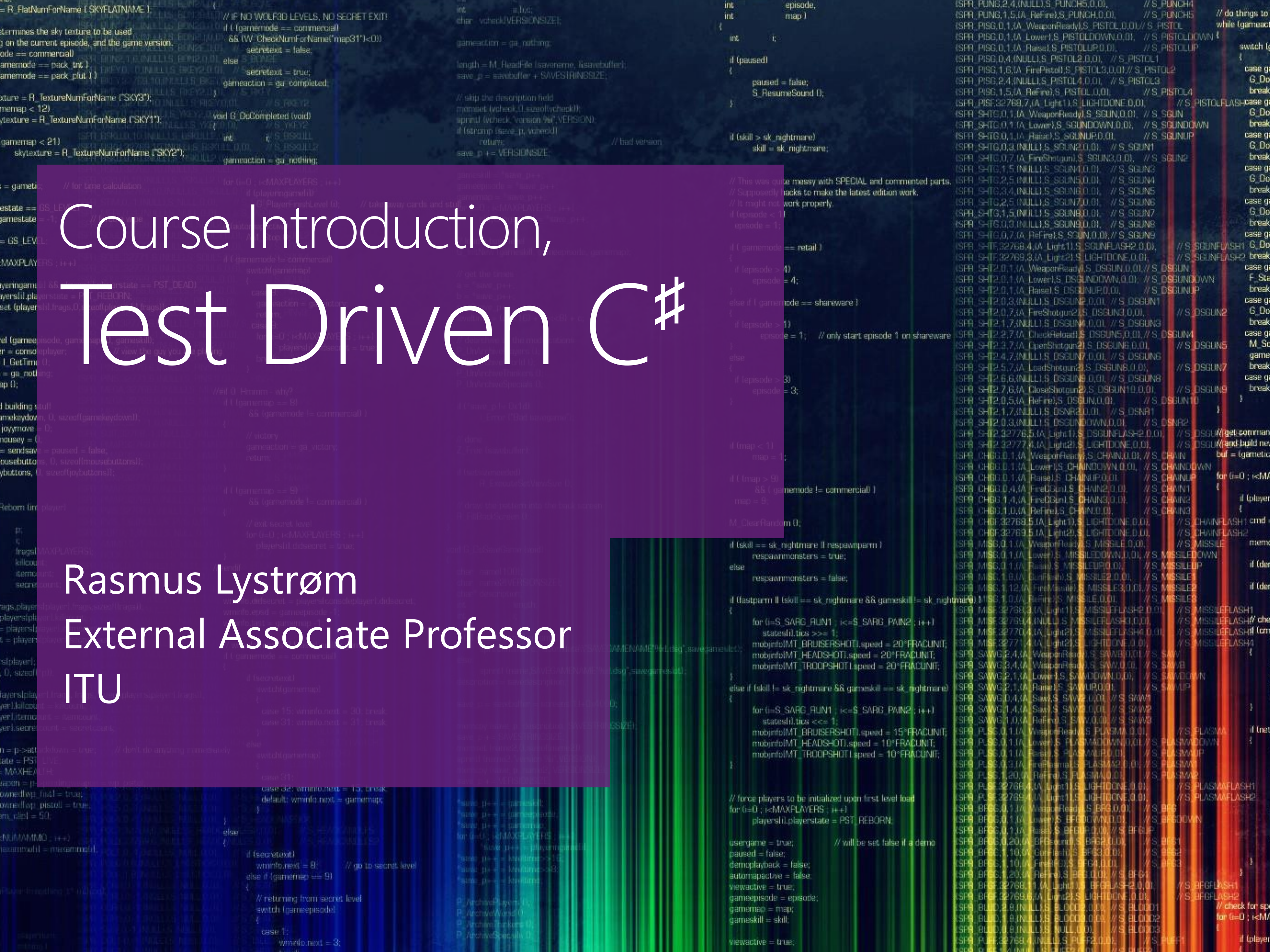


Course Introduction, Test Driven C♯

Rasmus Lystrøm
External Associate Professor
ITU

Me

Rasmus Lystrøm, rasmus@lystroem.dk

37.997 yrs. old

Wife Katrine, married for 1 year

Daughters: Alma Sophia, age 6.83 yrs. Laura, age 7 weeks

2013-

Consultant @ Microsoft

2007-
d&n

2001-
Lystrøm ApS

2011-2013

Teacher

Copenhagen University, College of Engineering Technical University of Denmark

2008-2012

M.Sc. IT @ ITU

Thesis: Forecalc – Developing a core spreadsheet implementation in F♯

1998-2001

B.Sc. Warfare @ Hærens Officersskole

1996-2008

Danish Army

1st Lieutenant

Agenda

TDD

.NET and the C♯ language

Visual Studio 2015

Test Driven C♯

Test-Driven Development

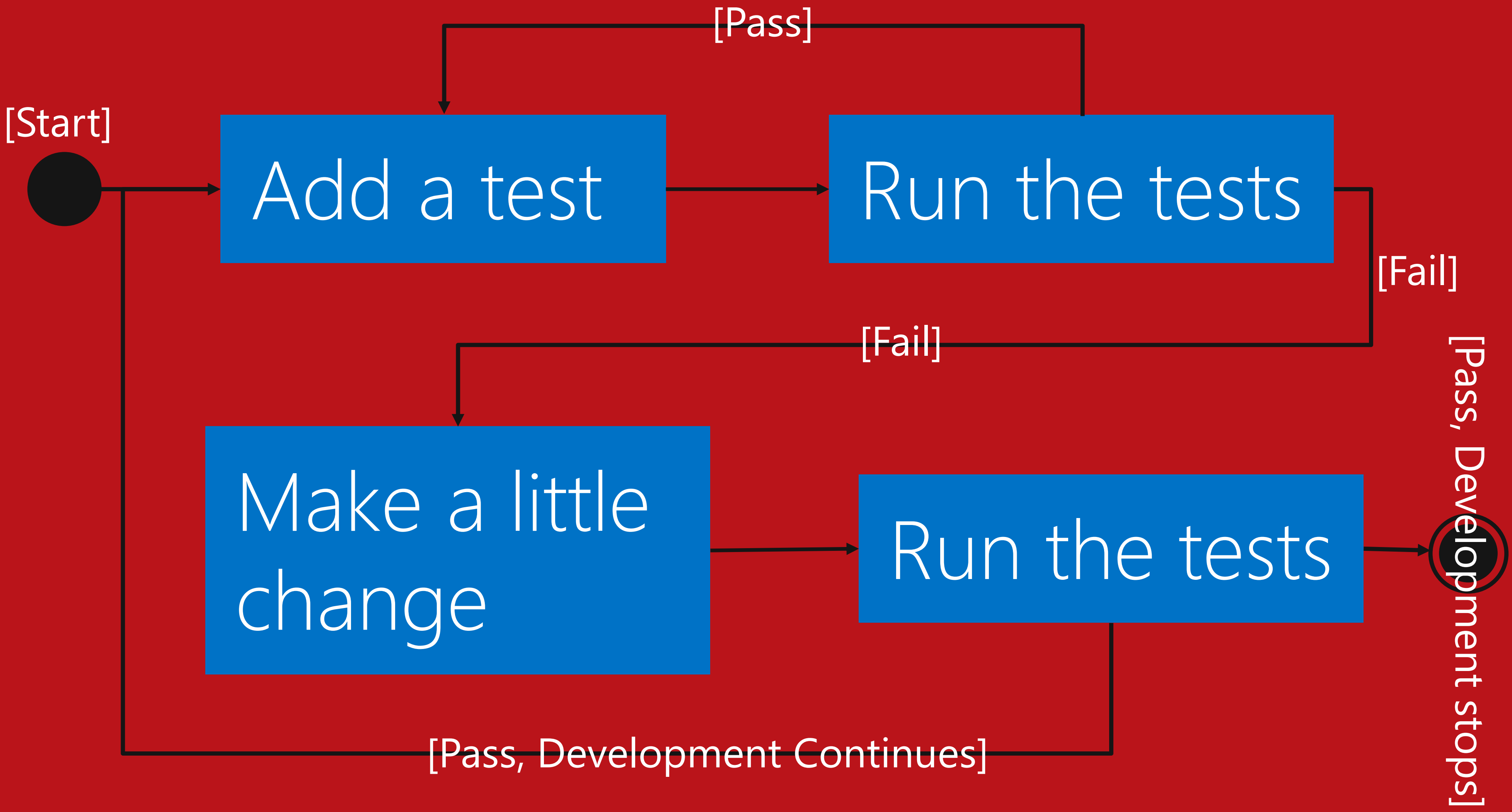
What?

Why?

How?

How?

RED-GREEN-REFACTOR

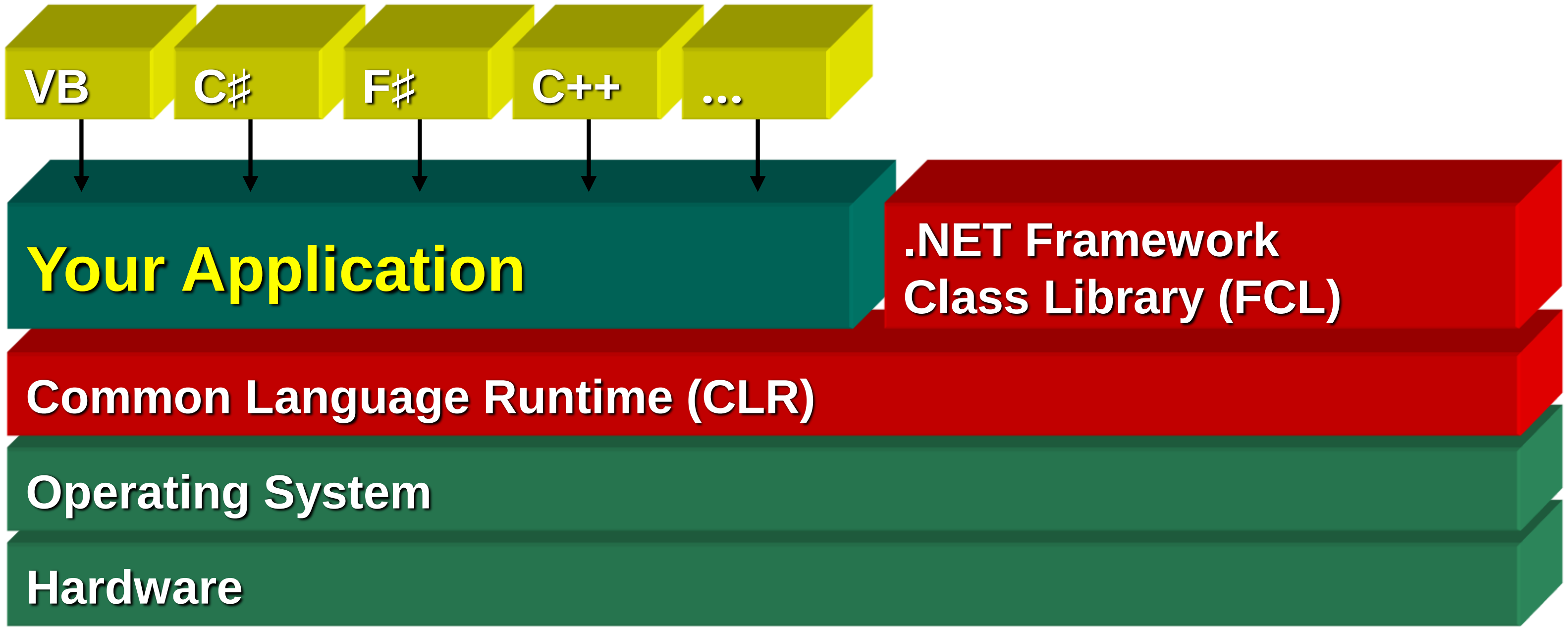


Microsoft®
.NET

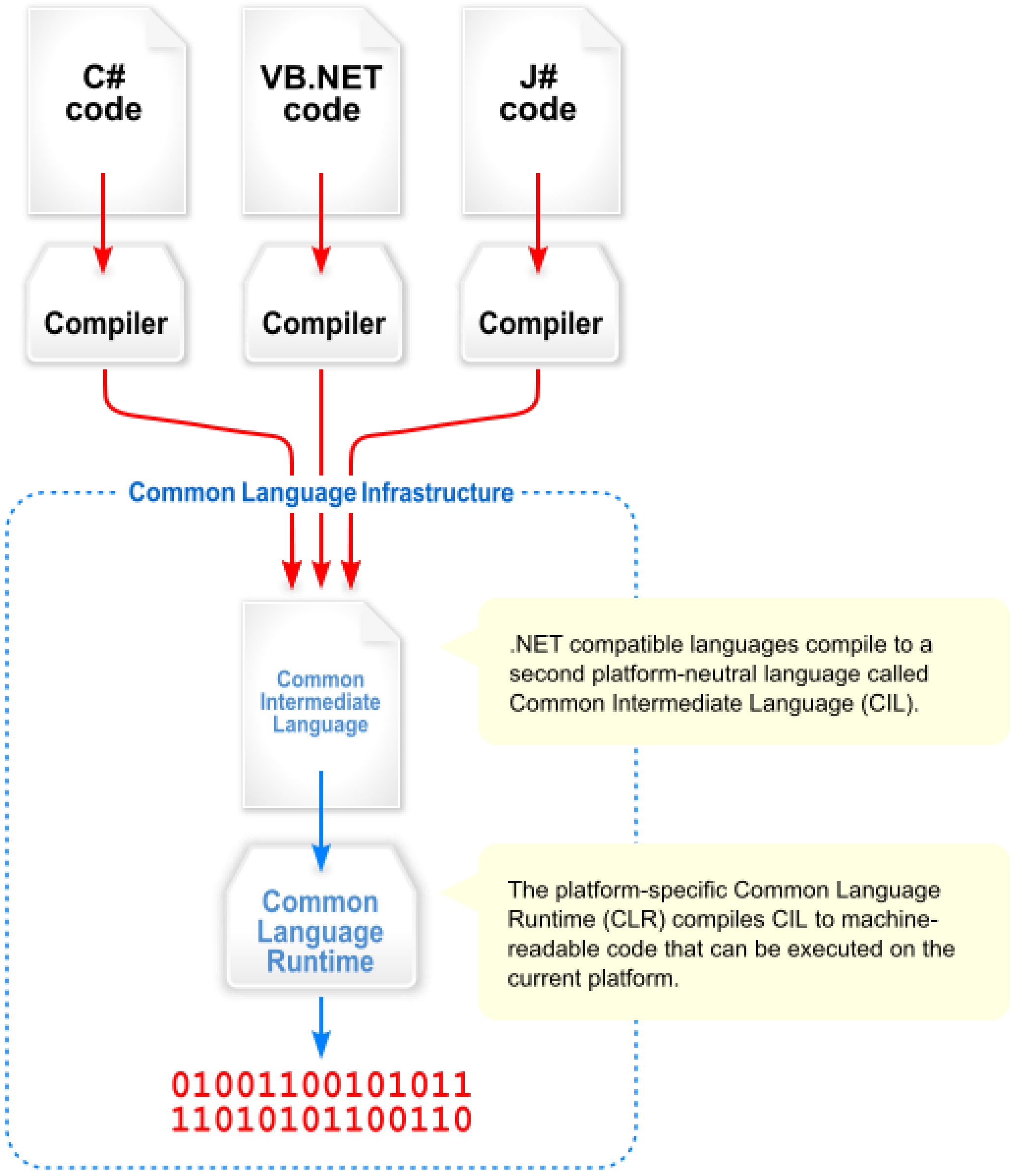
A brief introduction

.NET Framework

.NET offers multiple languages, a large class library, driven by a virtual machine


VB
C♯
F♯
C++
...
Your Application
.NET Framework Class Library (FCL)
Common Language Runtime (CLR)
Operating System
Hardware

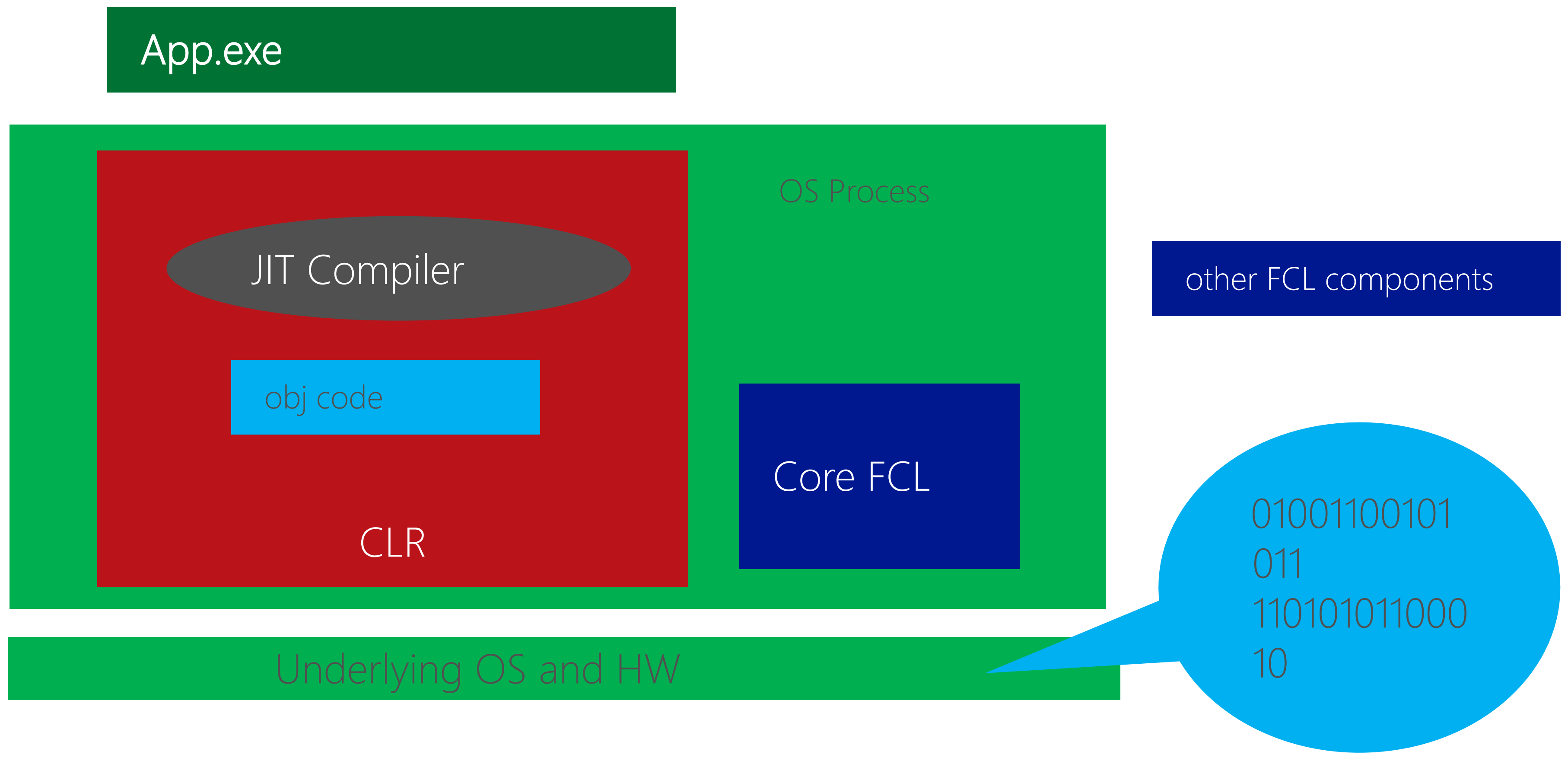
.NET Framework Compilation



.NET compatible languages compile to a second platform-neutral language called Common Intermediate Language (CIL).

The platform-specific Common Language Runtime (CLR) compiles CIL to machine-readable code that can be executed on the current platform.

CLR Execution



.NET Framework

Versions

1.0 Visual Studio .NET (2002)

1.1 Visual Studio .NET 2003

2.0 Visual Studio 2005

3.0 (2006)

3.5 Visual Studio 2008 (2007)

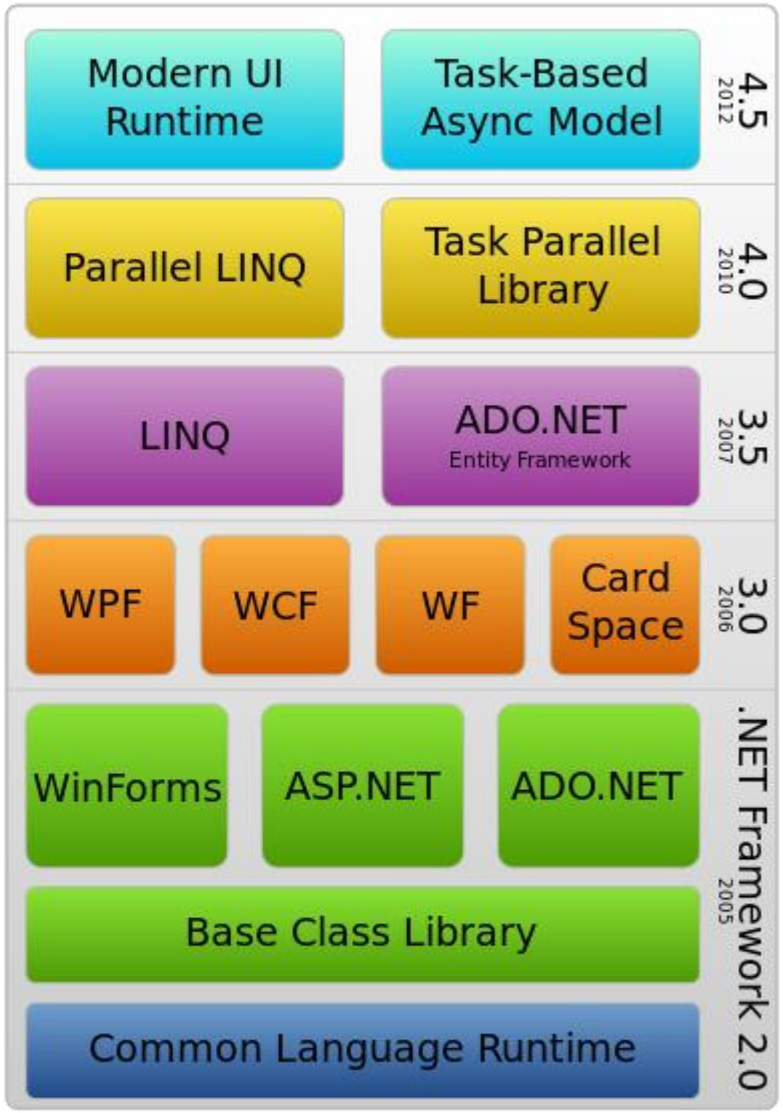
4.0 Visual Studio 2010

4.5 Visual Studio 2012

4.5.1 Visual Studio 2013

4.5.2 (2014)

4.6 Visual Studio 2015



The .NET Framework Stack

C♯

C♯ is intended to be a simple, modern, general-purpose, object-oriented programming language.

Ecma International (2006)

C#

Language Basics

Show me the CODE!!!

Naming Conventions

Composed names

`currentLayout`, `CurrentLayout`

Variables and fields

`vehicle`, `leftElement`

Private fields

`_vehicle`, `_leftElement`

Methods

`CurrentVehicle()`, `Size()`

Properties

`Pi`, `Name`, `Size`

Classes

`MyClass`, `List<T>`

Interfaces

`IException`, `IObserver`

http://msdn.microsoft.com/en-us/library/ms229002(v=vs.110).aspx

`Value Types can never be null!`

Value Types

Holds a value – assignment copies the value

Struct

- Numeric types
    - Integral types
    - Floating point types
    - Decimal
- bool

Integral	Floating point
`byte, sbyte, short, ushort, Char, int, uint, long, ulong`	`float Double`
	Decimal `decimal`

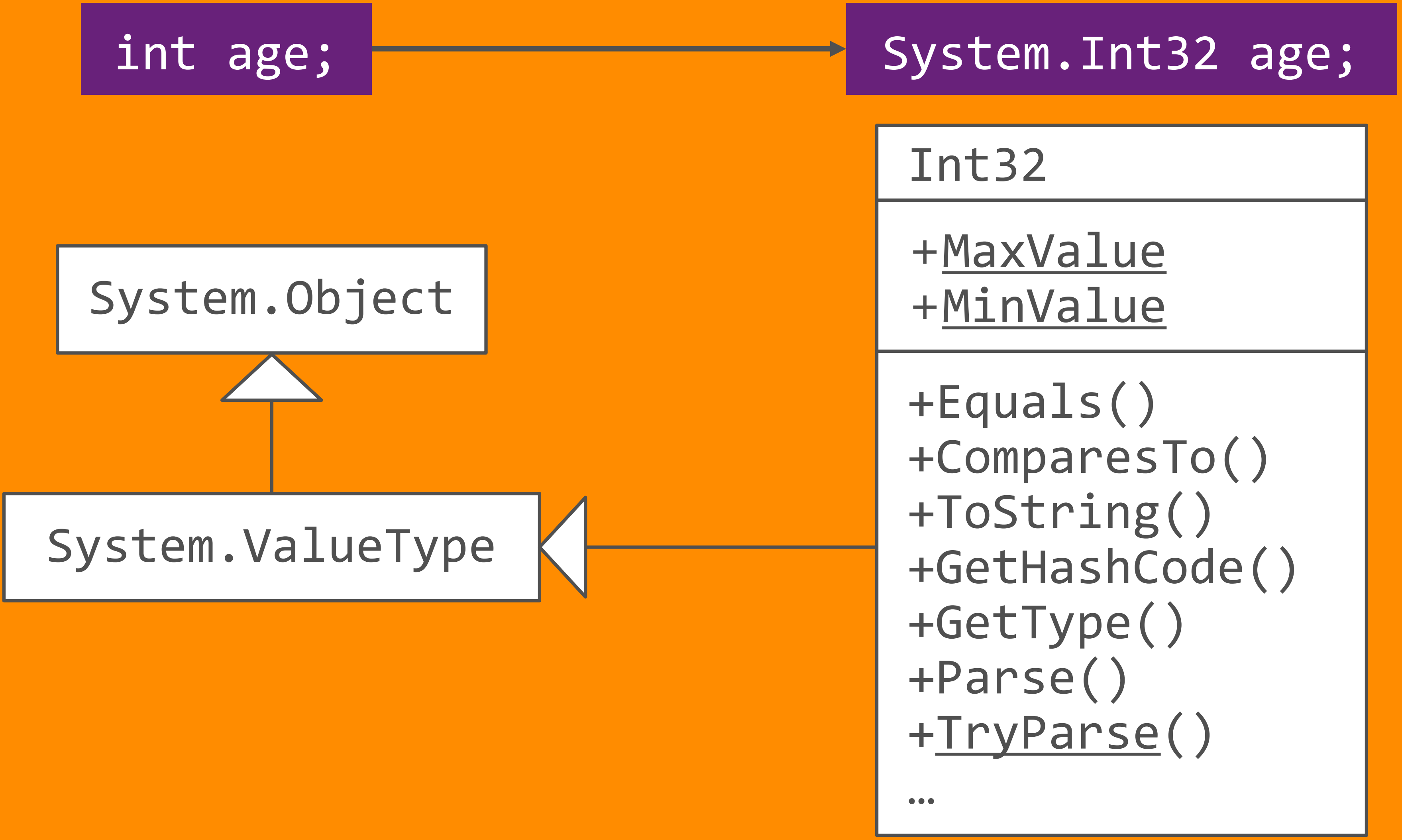
Enumeration

```
enum Days {Sat, Sun, Mon, Tue, Wed, Thu, Fri};
```

User defined structs

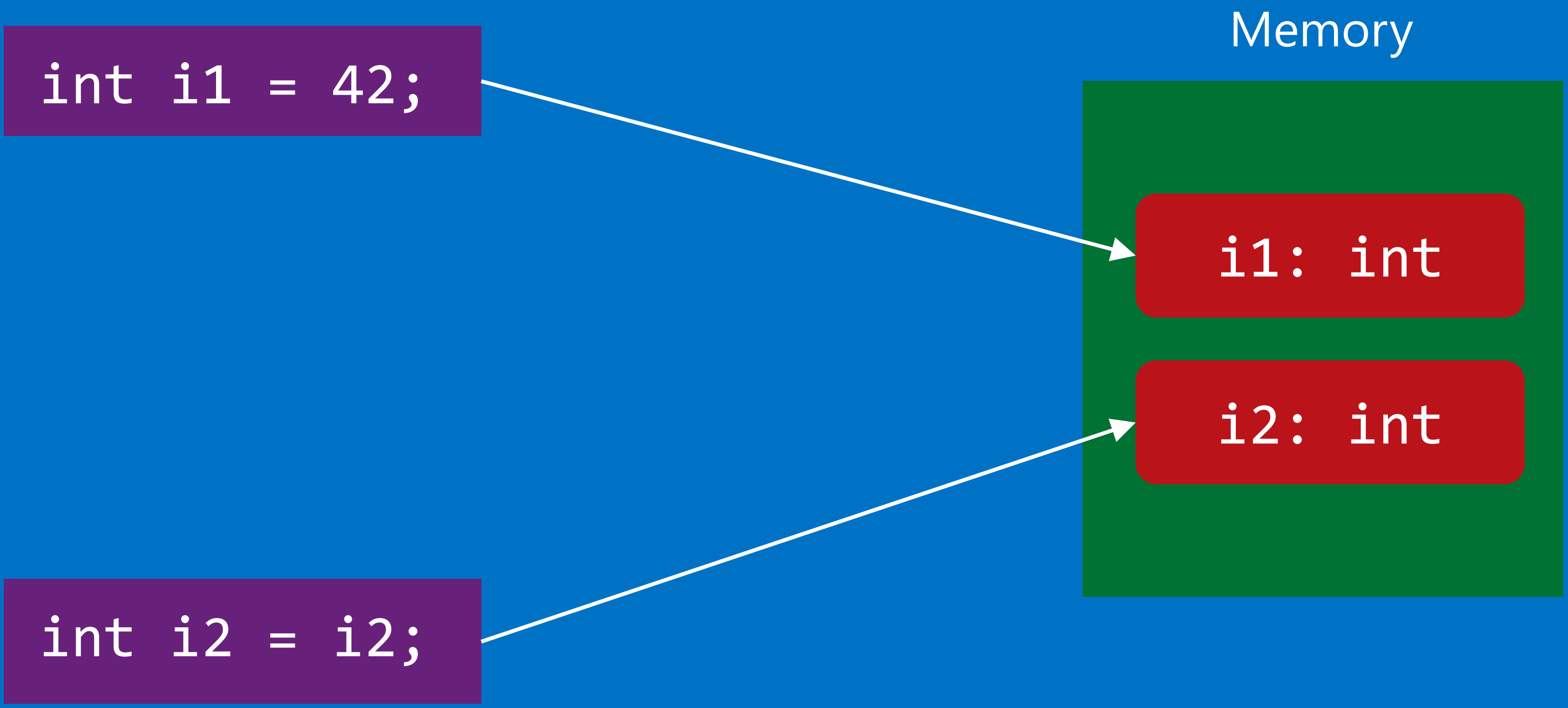
```
System.Guid
System.Drawing.Point
System.DateTime
System.Numerics.BigInteger
System.Numerics.Complex
```

Value Types



int age;
System.Int32 age;
Int32
+MaxValue
+MinValue
+Equals()
+ComparesTo()
+ToString()
+GetHashCode()
+GetType()
+Parse()
+TryParse()
…
System.Object
System.ValueType

Value Types

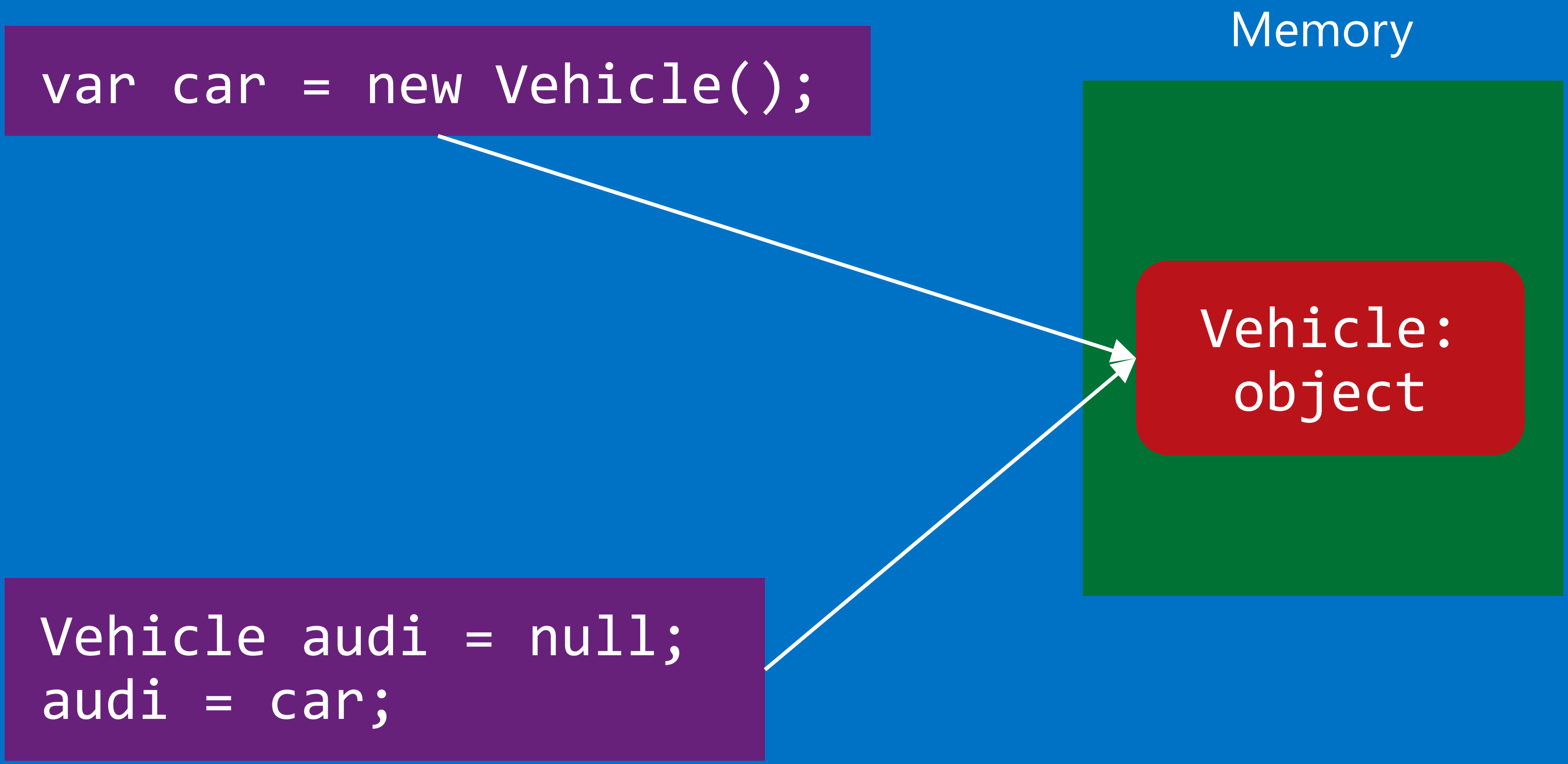


```
int i1 = 42;
```

```
int i2 = i2;
```

`ReferenceEquals` is always *`false`*

Reference Types



```
var car = new Vehicle();
```

```
Vehicle audi = null;
audi = car;
```

Reference Type Equality

ReferenceEquality:
Person p1 = new Person(“Joe”);
Person p2 = new Person(“Joe”);
Person p3 = p2;
ReferenceEquality(p1, p2) = false
ReferenceEquality(p2, p3) = true;

In C# the == operator is “equal” to reference equality. (Can be overridden)

Value Equality (for reference types)
p1.Equals(p2) = true; (Can be overridden)

Value Type Equality

Equals the same as for reference types

object.ReferenceEquality will always return false for value types

== operator is overridden so it does value equality

String Interning

```
string a = "Peter";
string b = "Peter";

a.Equals(b);     ==> true

a == b;          ==> true

object.ReferenceEquals(a, b); ==> true
```

String Interning

The String Type is Immutable – assigning creates a new value…

Local Variable Type Inference

var *identifier* = *expression*;

var is a keyword, not a type

Enumeration

```
public enum Day { Mon, Tue, Wed, Thu, Fri, Sat, Sun }

public enum Month : uint { Jan = 1, Feb, Mar, }

public enum Color : uint { Red = 0xFF0000,
                           Green = 0x00FF00,
                           Blue = 0x0000FF }

Vehicle car = new Vehicle();
car.Color = Color.Red;

Console.Write(Day.Mon);
```

Array stuff

```
int[] intArray = new int[4];

double[,] doubleArray = new double[4, 5];

int[,] array1 = {{1,2],{3,4]};

int value1 = intArray[0];

double value2 = doubleArray[0,1];

Console.WriteLine(array1[1,1]);
```

Arrays are 0-based

String stuff

```
public static void Main(string[] args)
{
    var name = "Anders";
    var argument = args[0];

    Console.WriteLine(10 + " hello " + name + argument);
}
```

Automatic conversion

Compile from Command Line

Developer Command Prompt for VS2015:

```
C:\[program_dir]> csc
```

Command Line Options

Compiler Option	Short Form	Description
`/target:exe` `/target:library`	`/t:exe` `/t:library`	Compile to an executable (.exe file) (default) Compile to a library (.dll file)
`/reference:Lib.dll`	`/r:Lib.dll`	Include reference to library **`Lib.dll`**
`/main:MyClass`	`/m:MyClass`	Method **`Main`** in **`MyClass`** is the entry point
`/debug`	`/d`	Add debugging information

```
C:\Program Files (x86)\Microsoft Visual Studio 14.0>csc /?
```