C# Equality and Comparisons

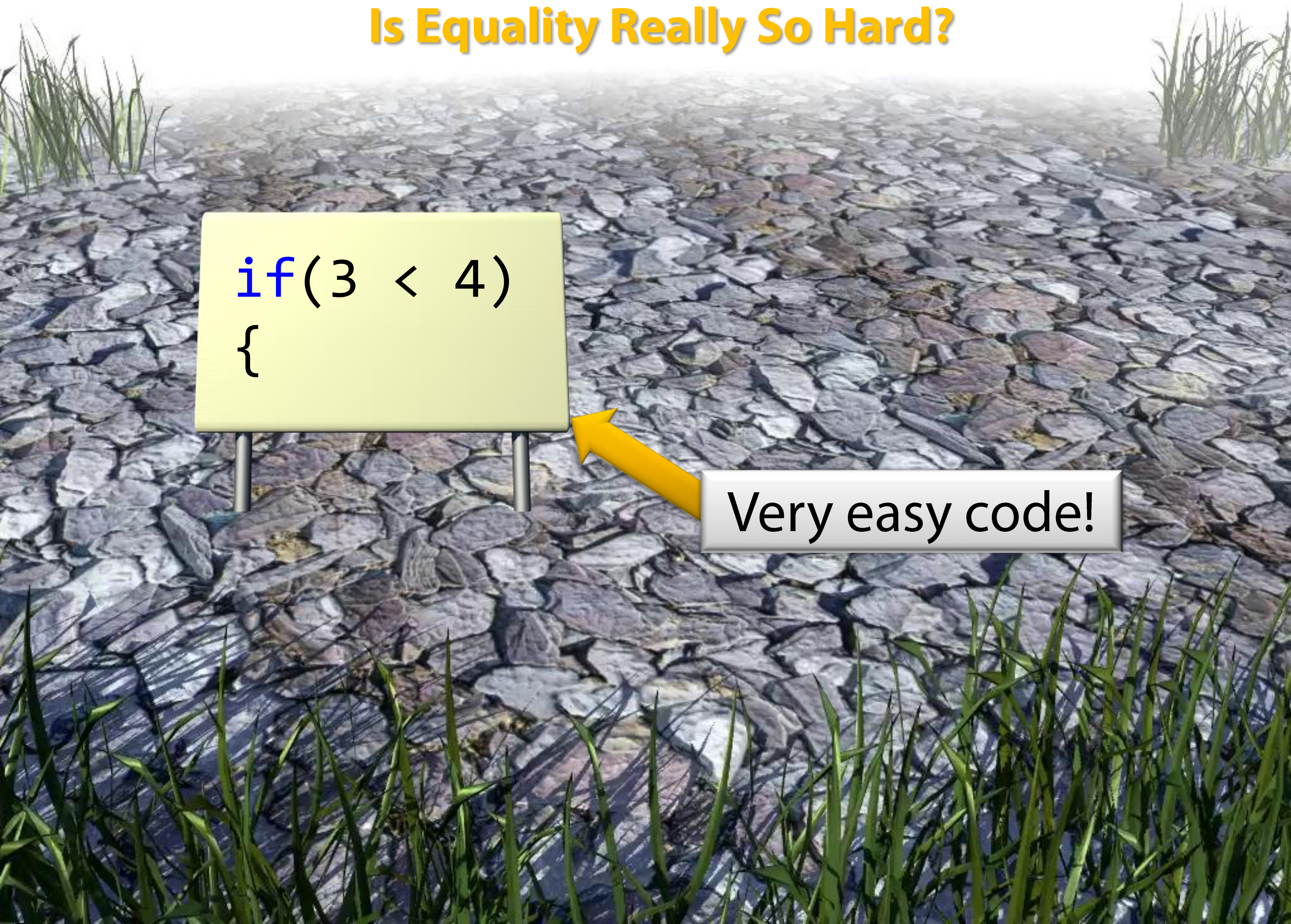
Why Is Equality So Hard?

Simon Robinson
http://TechieSimon.com
@TechieSimon

pluralsight
hardcore dev and IT training

7. Strings
4. Overriding Equality: Value Types
5. Overriding Equality: Reference Types
8. Comparers and Equality Comparers
10. Structural Equality
3. == in C#
6. Comparisons in .NET
9. Hash Codes
2. Equality in .NET
1.Why Is Equality So Hard?

Is Equality Really So Hard?



```
if(3 < 4)
{
```

Very easy code!

.NET Provides…

```
System.Object

static Equals()
virtual Equals()
static ReferenceEquals()
virtual GetHashCode()
```

```
IEquatable<T>
IComparable
IComparable<T>
IComparer
IComparer<T>
IEqualityComparer
IEqualityComparer<T>
IStructuralEquatable
IStructuralComparable
```

Incorrect implementation can
- Cause subtle bugs
- Break collections etc.

Aim of this course

IEquatable<T>

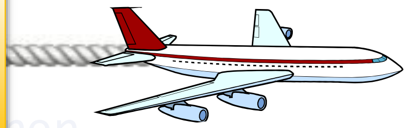
IComparable<T>

IComparable

Know how to use all these methods/interfaces

Key skills:

- Understand equality and comparisions in .NET out of the box
- Know how to customize equality and comparisons correctly



virtual GetHashCode()

IEqualityComparer<T>

IStructuralEquatable

IStructuralComparable

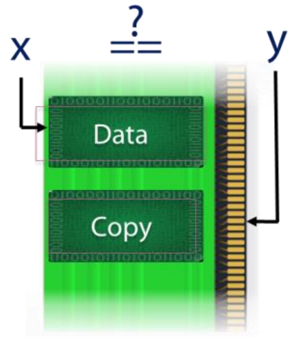
Why is Equality Hard?

4 Reasons

Do Not Place Anything

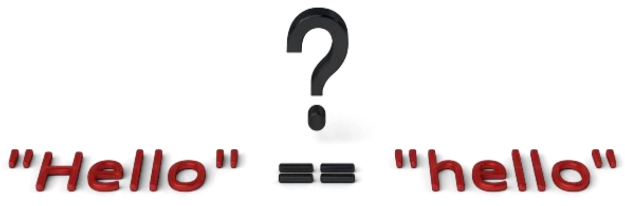
Why is Equality Hard?

1. Reference/Value Equality



(C# syntax doesn't distinguish ref/value)

2. Multiple ways to compare values

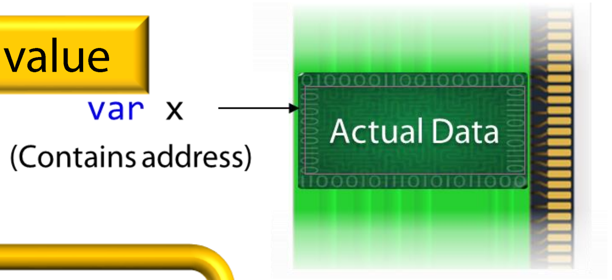


3. Accuracy
(for floating points)

4. Conflict with OOP

Reference vs Value Equality

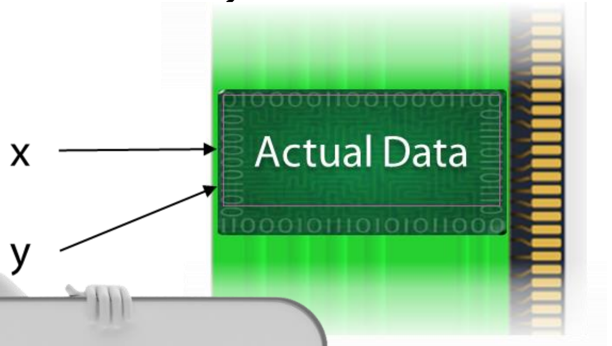
Reference types contain a pointer to the value

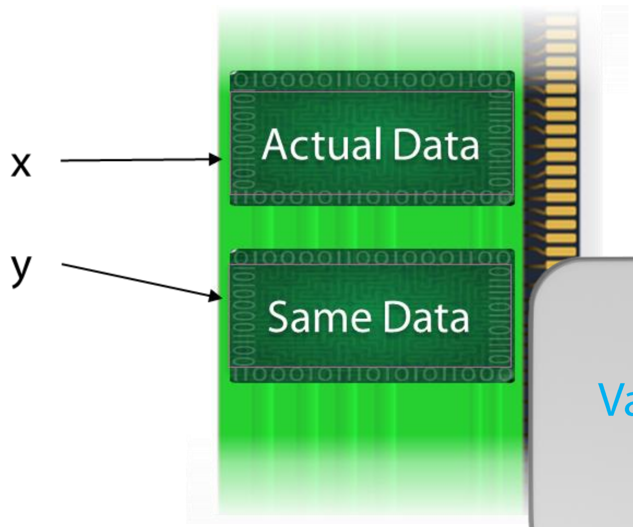


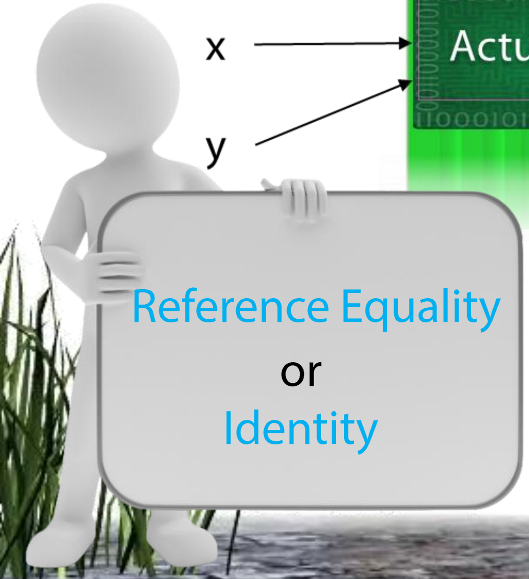
x == y

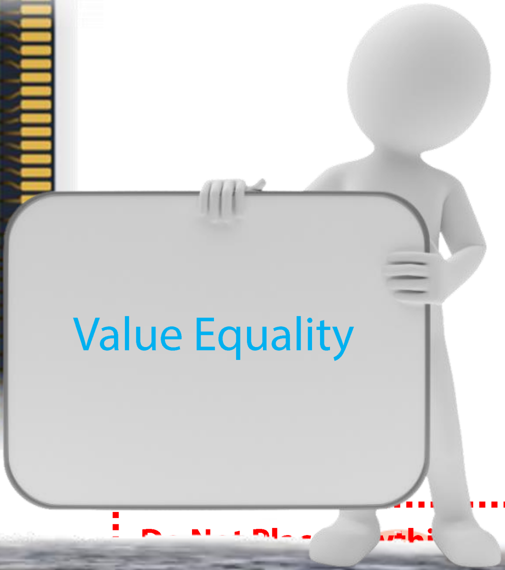
Same memory location?

Same value?





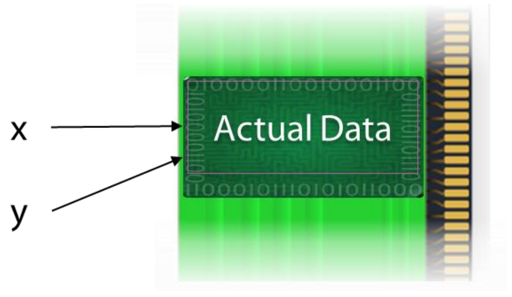




Code Demo

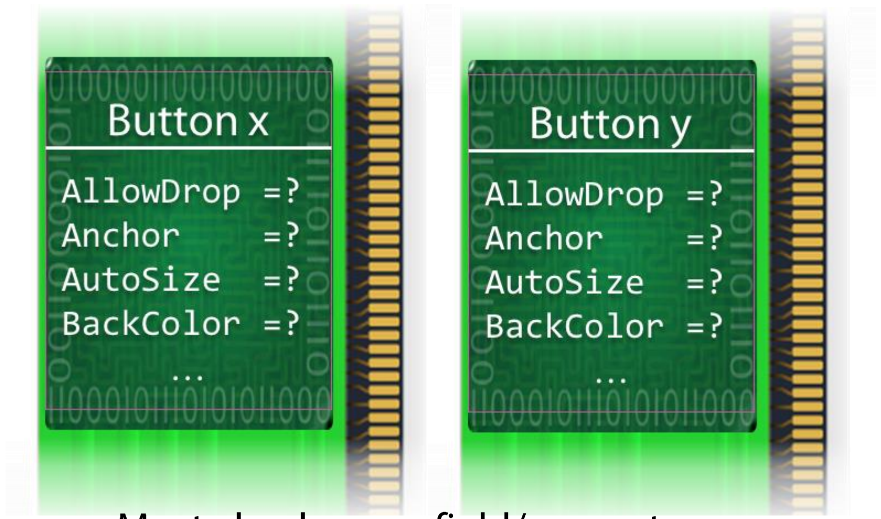
Performance

Reference Equality: Quick



Do x and y contain the same address?

Value Equality: Slow



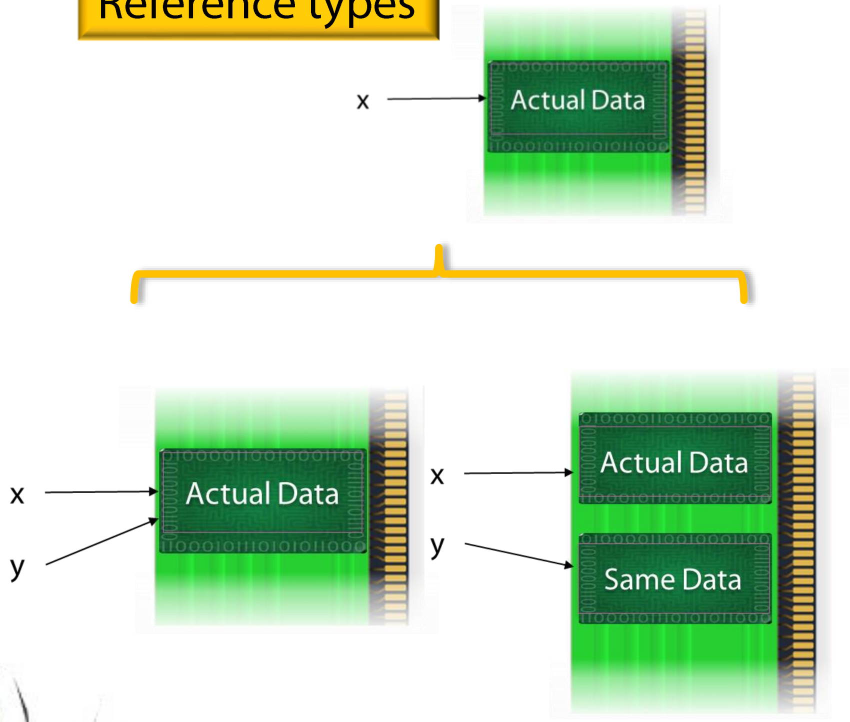
Must check every field/property

No operator does this for buttons in C#

Code Demo

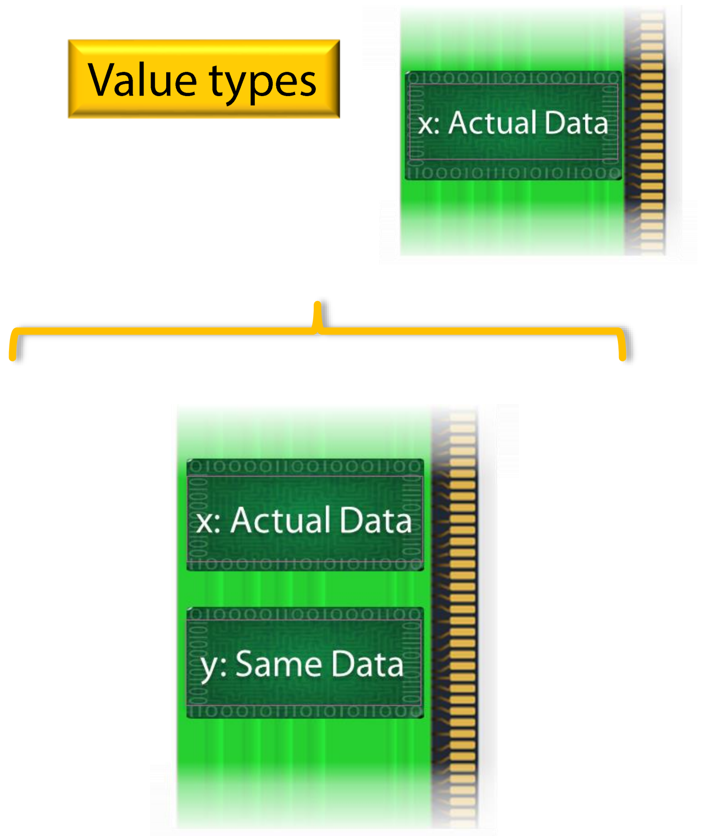
Reference vs Value Types

Reference types



Reference Equality Value equality

Value types



Value equality only

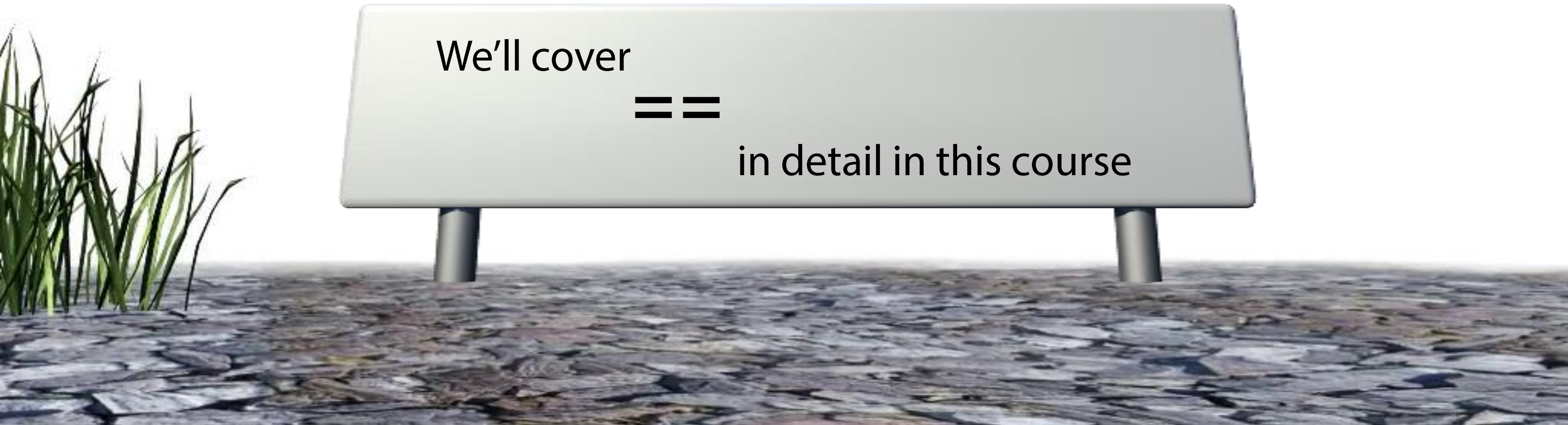
Code Demo

C# Has Only One == Operator

```
if (a == b) {
```

What will this do?

You just have to know what == does for each type


We'll cover == in detail in this course

Multiple Ways to Compare Values

?
"Hello" == "hello"

```
string str1 = "apple pie";
string str2 = "apple pie";
if (str1 == str2) {
```

Should we say these are equal?

Clearly, Yes! ✓

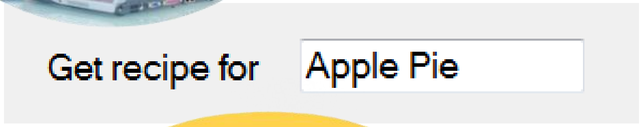
```
string str1 = "apple pie";
string str2 = "Apple Pie";
if (str1 == str2) {
```

Should we say these are equal?

C# says they are not:
`(str1 == str2)` evaluates to `false`

But more generally:
It depends on the context

Should You Ignore Case?

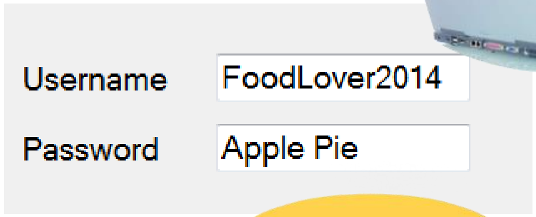


Get recipe for	Apple Pie

Recipes
apple pie
...

Case doesn't matter

We want
"Apple Pie" = "apple pie"



Username	FoodLover2014
Password	Apple Pie

Passwords
FoodLover2014:
apple pie

Case does matter

We want
"Apple Pie" != "apple pie"

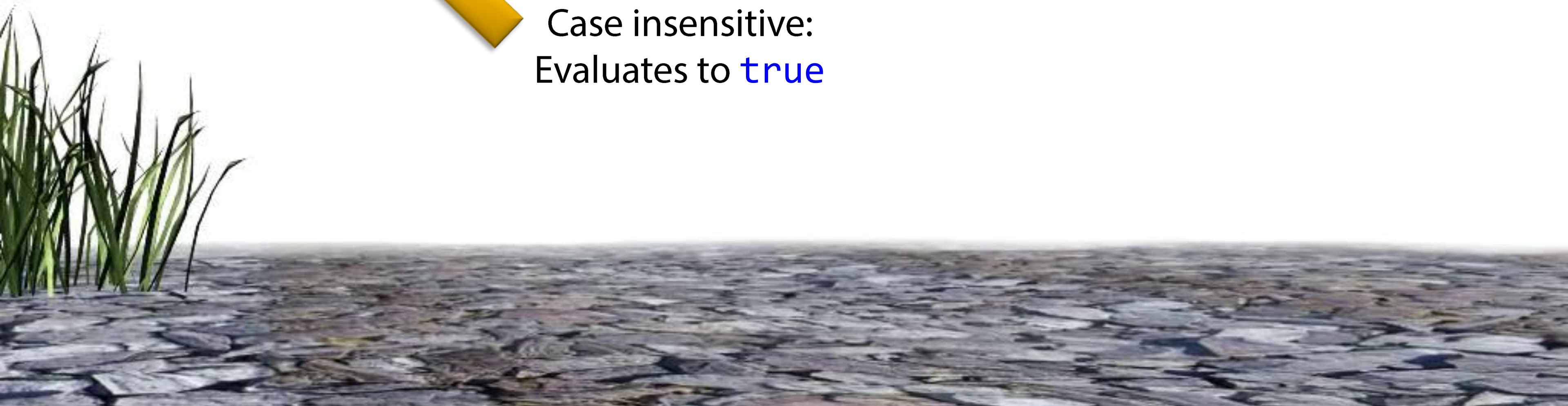
String Comparisons

```
string str1 = "apple pie";
string str2 = "Apple Pie";
if (str1 == str2) {
```

Case sensitive:
Evaluates to `false`

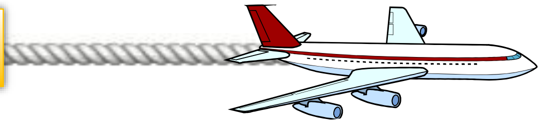
```
string str1 = "apple pie";
string str2 = "Apple Pie";
if (str1.Equals(str2, StringComparison.OrdinalIgnoreCase)) {
```

Case insensitive:
Evaluates to `true`

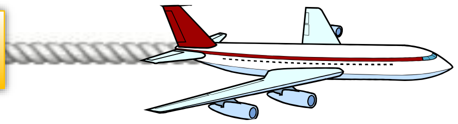


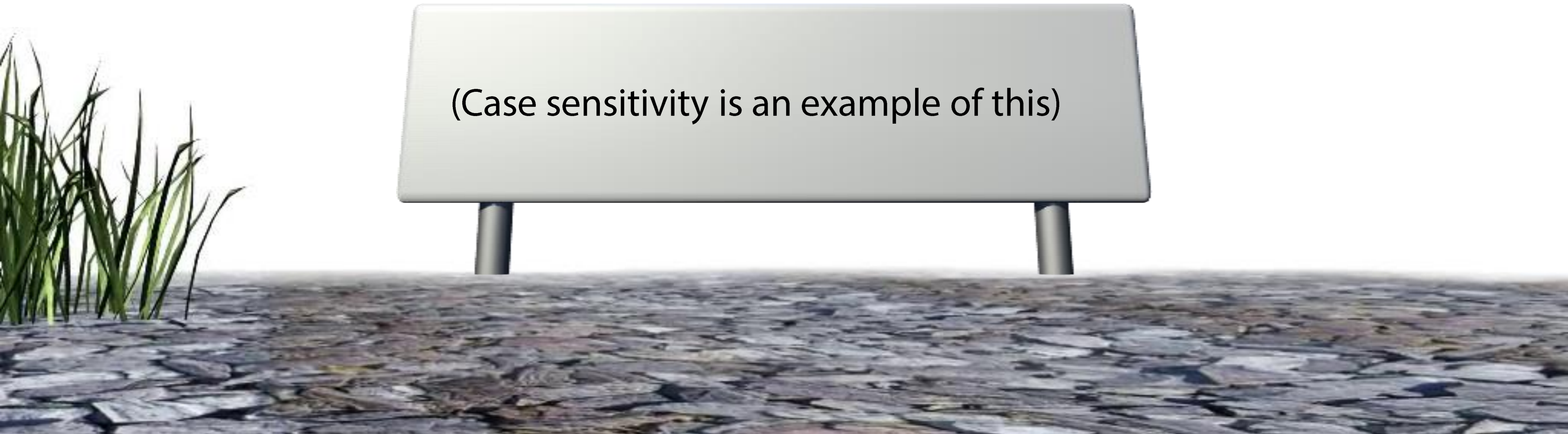
More generally…

Equality is not an absolute…



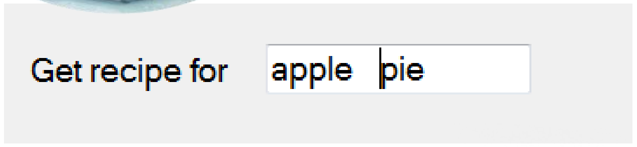
… equality is often context-sensitive





(Case sensitivity is an example of this)

Example: Whitespace

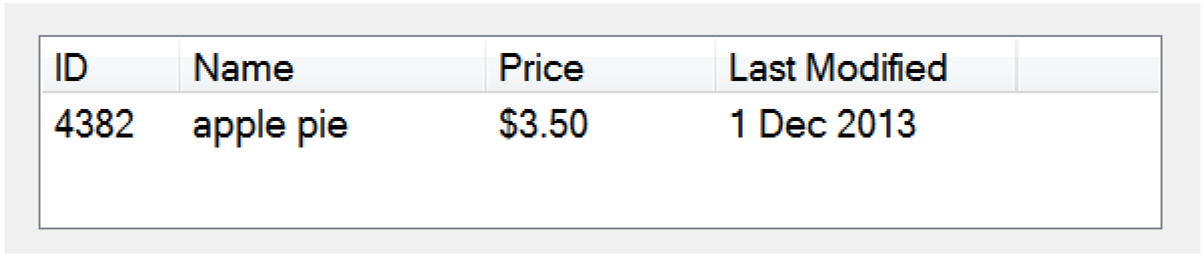

Get recipe for apple   pie

Recipes
apple pie
...

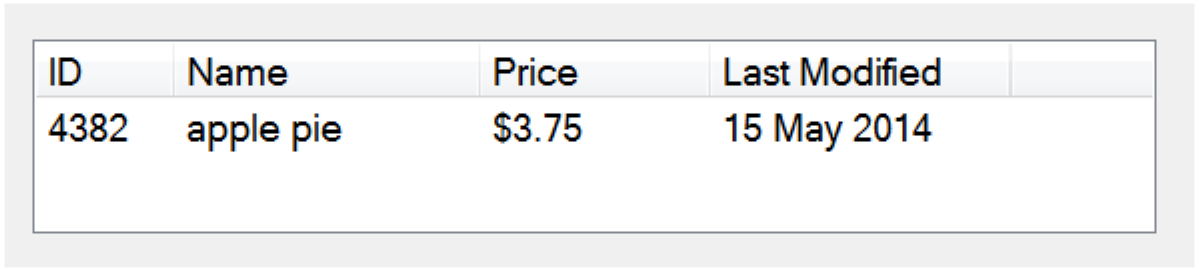
Excess whitespace doesn't matter here

Example: Database Records

Are these equal…?

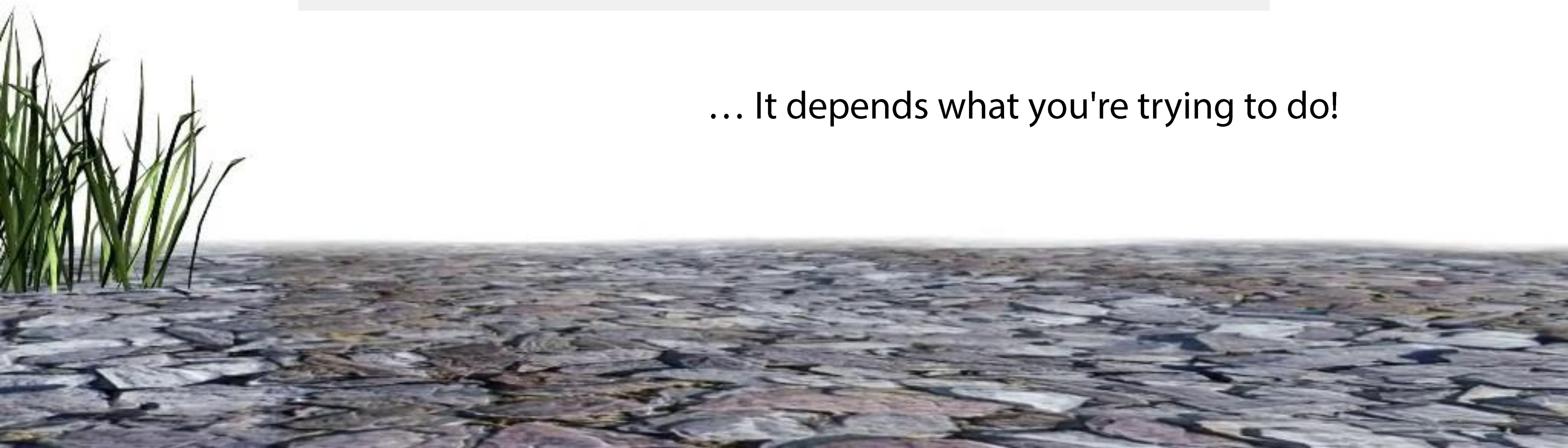


ID	Name	Price	Last Modified
4382	apple pie	$3.50	1 Dec 2013



ID	Name	Price	Last Modified
4382	apple pie	$3.75	15 May 2014

… It depends what you're trying to do!



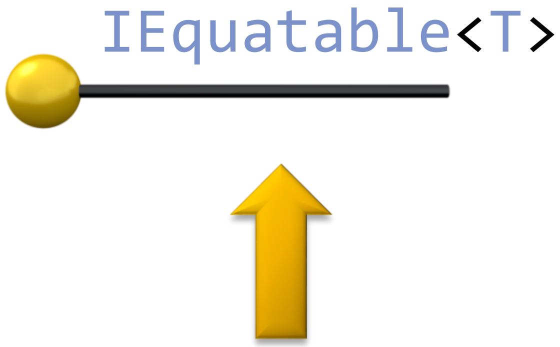
Natural vs Plugged-in Equality

In .NET: Each type can define its own 'natural' equality

Eg. string: Equal if they contain exactly the same sequence of characters

```
string str1 = "apple pie";
string str2 = "Apple Pie";
if (str1 == str2) {
```

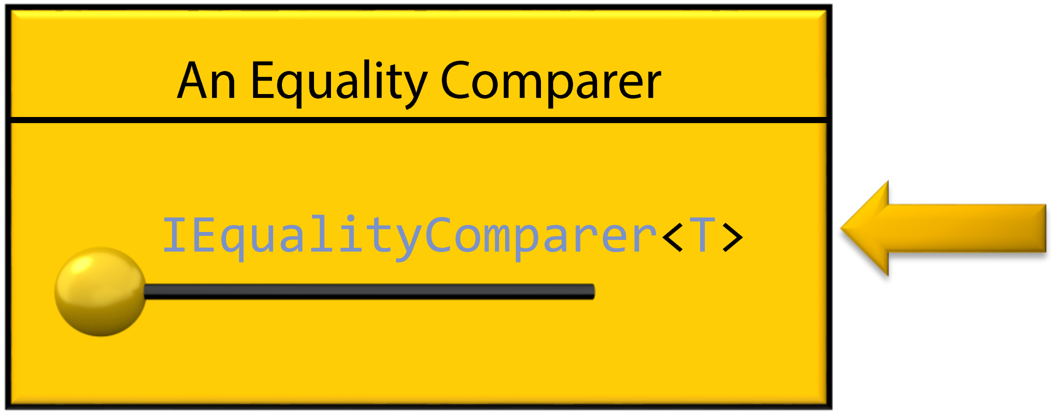
Not equal because – different characters!



Interface for 'natural' equality

Natural vs Plugged-in Equality

But you can plug in alternative equality definitions, using…



Interface for 'plugged in' equality

Example:
You could write an equality comparer to compare strings – ignoring excess whitespace

Equality vs Comparisons

	Equality	Comparisons
"Natural"	`IEquatable<T>` ==, != operators	`IComparable<T>` >, < etc. operators
"Plugged-in"	`IEqualityComparer<T>`	`IComparer<T>`

Accuracy of Data

Some data types are inherently approximate:

```
float      double      decimal
```

Code Demo

A Simple Calculation

$$\begin{array}{r} 5.05 \\ +\,0.95 \\ \hline =\,6.00 \end{array}$$

Code Demo

Floating Point Rounding Errors

```
  5.05
+ 0.05
------
= 6.00
```
✓

```
(5.05f + 0.95f == 6f)
```
`== false` ✗

`Equals(), ==,` etc.

Often give the 'wrong' result for floating point numbers

(But less-than/greater-than comparisons are normally OK)

The Equality / Type Safety / OOP Conflict

Code Demo

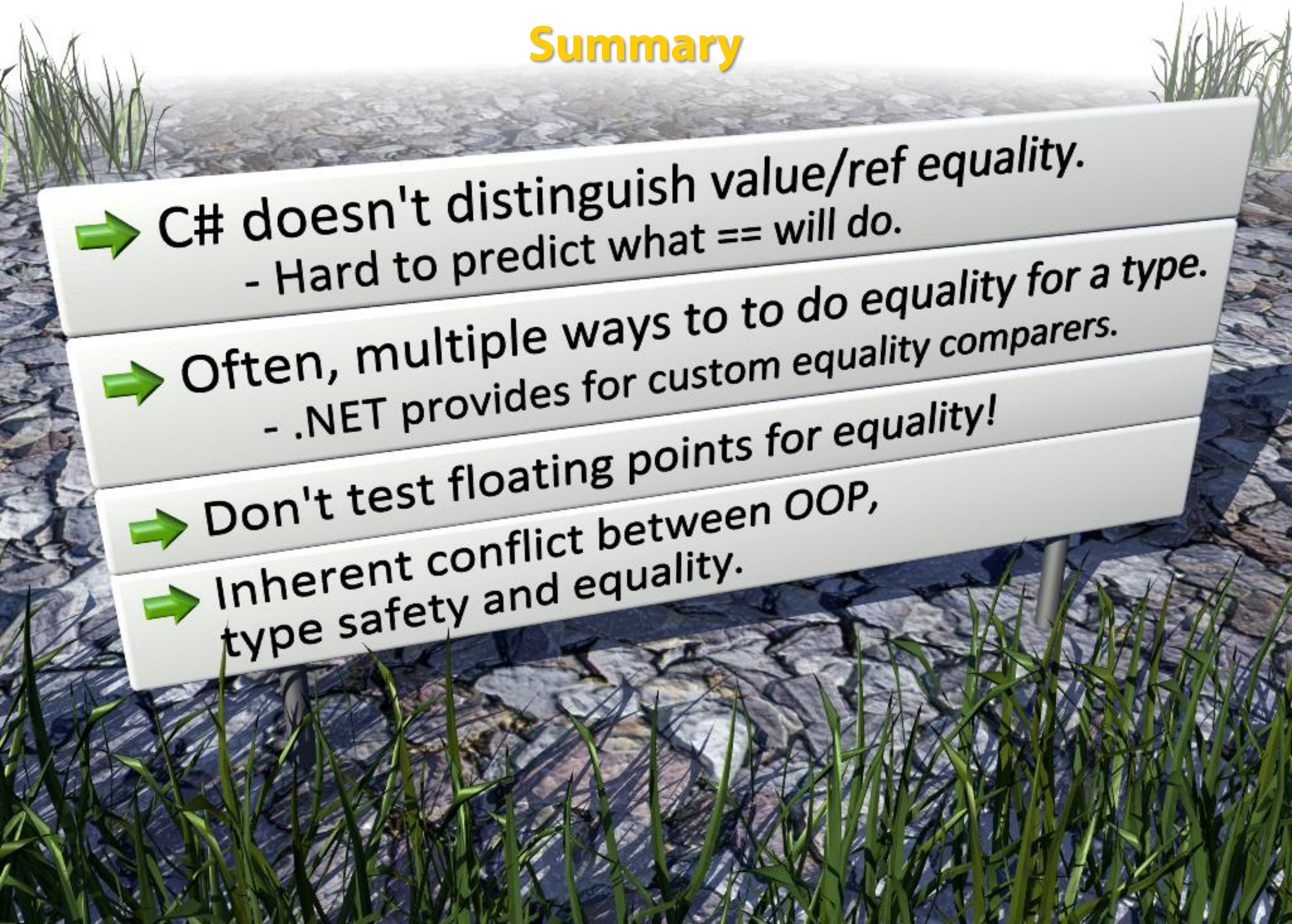
The Equality / Type Safety / OOP Conflict

The Workaround: Lose type safety!

Not type safe – but inheritance works!

```
class Object
{
        virtual bool Equals(object other) {
```

Summary



- C# doesn't distinguish value/ref *equality.*
  - Hard to predict what == *will do.*
- Often, multiple ways to to do *equality for a type.*
  - .NET provides for custom *equality comparers.*
- Don't test floating points for *equality!*
- Inherent conflict between *OOP,* type safety and equality.