Using Interfaces to Future-Proof Code

The "Why" (Part 1)

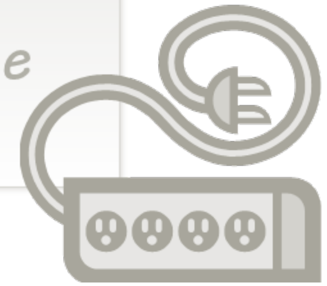
Jeremy Clark
www.jeremybytes.com
jeremy@jeremybytes.com

pluralsight
hardcore developer training

Why Interfaces?

Maintainable

Extensible



Easily Testable

Interfaces help us get there

Best Practice

Program to an abstraction rather than a concrete type

Translation

Contract

Program to an interface rather than a concrete class

Concrete Classes

Collections

List<T>
Array
ArrayList
SortedList<TKey, TValue>
HashTable
Queue / Queue<T>
Stack / Stack<T>
Dictionary<TKey, TValue>
ObservableCollection<T>
+
Custom Types

Interfaces

Collection Interfaces

```
public class List<T> : IList<T>,
    ICollection<T>, IList, ICollection,
    IReadOnlyList<T>, IReadOnlyCollection<T>,
    IEnumerable<T>, IEnumerable
```

IEnumerable

Used with

- **foreach**
- **List Boxes**

Summary

- **Best Practice**

Program to an abstraction rather than a concrete type

or

Program to an interface rather than a concrete class

- **Concrete Class**
  - Brittle / Easily Broken
- **Interface**
  - Resilience in the face of change
  - Insulation from implementation details

- **Next Up: The "How" of Interfaces**