F# Functional Data Structures

Introduction

Kit Eason
www.kiteason.com
@kitlovesfsharp

pluralsight
hardcore dev and IT training

Why Do Functional Data Structures Matter?
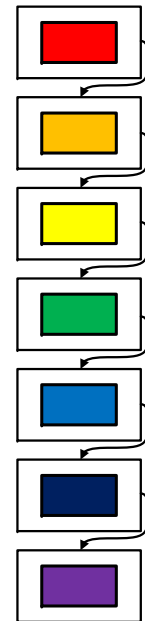
This is where you get the payoff

Essential to idiomatic code

Structures *and* associated operations

What is a Functional Data Structure?

- Any structure which it's convenient to use from F#!

- Arrays

- Sequences

- Lists

- Dictionaries

- Sets

- Maps

- Trees

Prerequisites

The **let** keyword

Returning values

The **forward pipe** operator |>

Tuples ()

Records {}

F# Interactive **FSI**

FSI

- ALT/Enter

- Type directly

let

- Binds an identifier to a value or function

- Declaration and assignment in one

- Types inferred if possible

```
let Area radius =
    System.Math.PI * radius**2.
```

|>

- "Forward pipe"

- Result of previous to argument of next

- Composable

```
let AreaApprox radius =
    Area radius
    |> round
```

Tuples

- Small group of values

- No field naming

- Don't overdo it!

```
let position = (1.0, 3.5, 9.2)
```

Records

- A type with field names and types

- Easy to instantiate

- Structural equality

```
type BankAccount =
    {
        Number : string
        Name : string
        Balance : double
    }

let myAccount =
    {
        Number = "123492A"
        Name = "J Richardson"
        Balance = 1394.98
    }
```

Summary

Functional data structures
- Any structure which it's convenient to use from F#!
- Arrays, Sequences, Lists, Dictionaries, Sets, Maps, Trees

The **let** keyword
- Binds an identifier to a value or function

Tuples
- Small anonymous group of values
- No field names
- () syntax

Records
- A named type
- With field names
- {} syntax

FSI
- Run F# code