Lists

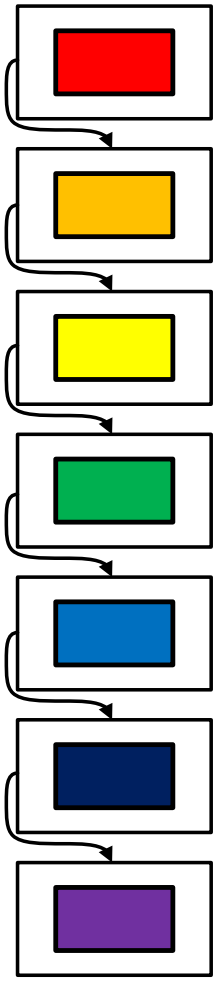
Kit Eason
www.kiteason.com
@kitlovesfsharp

> Lists represent the backbone of functional programming and in order to be an effective F# programmer you must truly master list processing.

— Chris Smith, F# Team

What is a List?

- A list of elements
- Computed on creation
- All elements same type
- Can’t assign to elements
- Implemented as a (singly) linked list



F# List Versus C# List

An F# list isn't the same thing as a C# list!

Full Name	C# Name	F# Name	Mutability
System.Collections.Generic.List<'T>	List<'T>	ResizeArray	**Mutable**
Microsoft.FSharp.Collections.List<'T>	FSharpList<'T>	List	**Immutable**

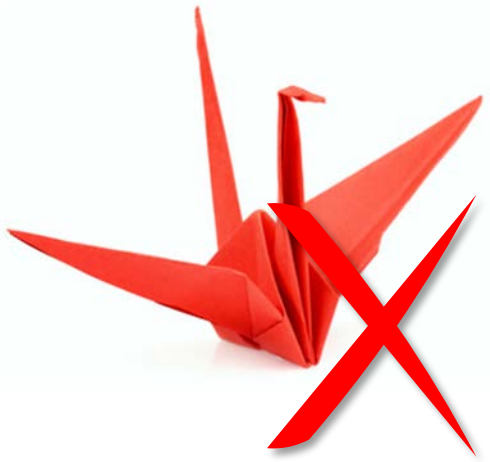
Creating a List

- **From a range expression**
  - `let integers = [1..1000]`
- **From a list expression**
  - `let integers = [for i in 1..1000 do yield i]`
  - `let integers = [for i in 1..1000 -> i]`
- **Using a function in the List module**
  - `let integers = List.init 1000 (fun i -> i+1)`
- **From another other collection**

```
let Files (dir : string) =
   Directory.EnumerateFiles(dir)
   |> List.ofSeq
```

What? No List.unfold?

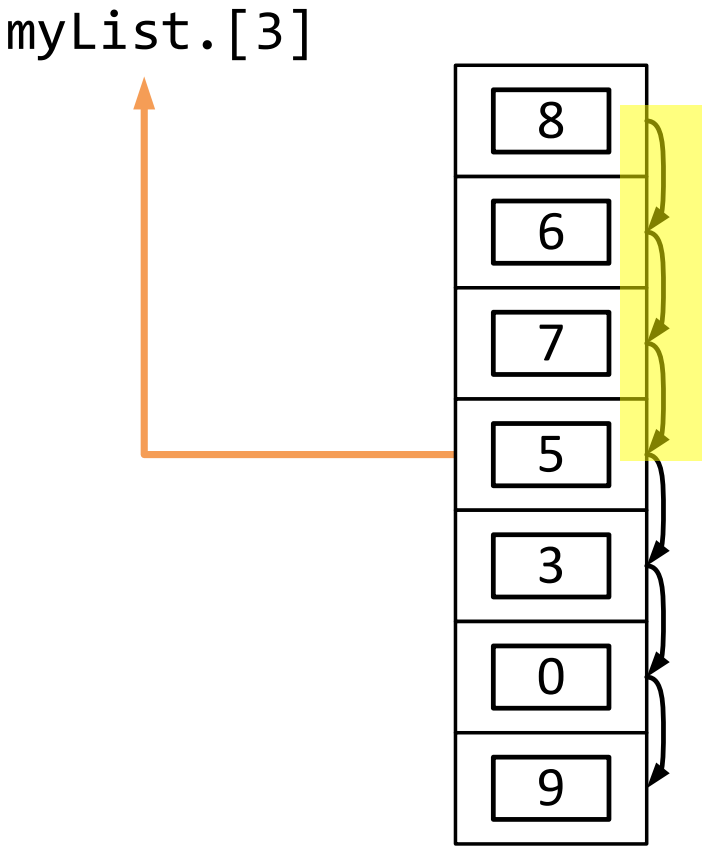
```
let myUnfoldList =
    Seq.unfold (fun state ->
        if state > 100 then None
        else Some(state, state+1)
    ) 0 |> List.ofSeq
```



```
let rec myListRecursive n =
    [
        if n < 100 then
            yield n
            yield! (myListRecursive (n+1))
    ]
```

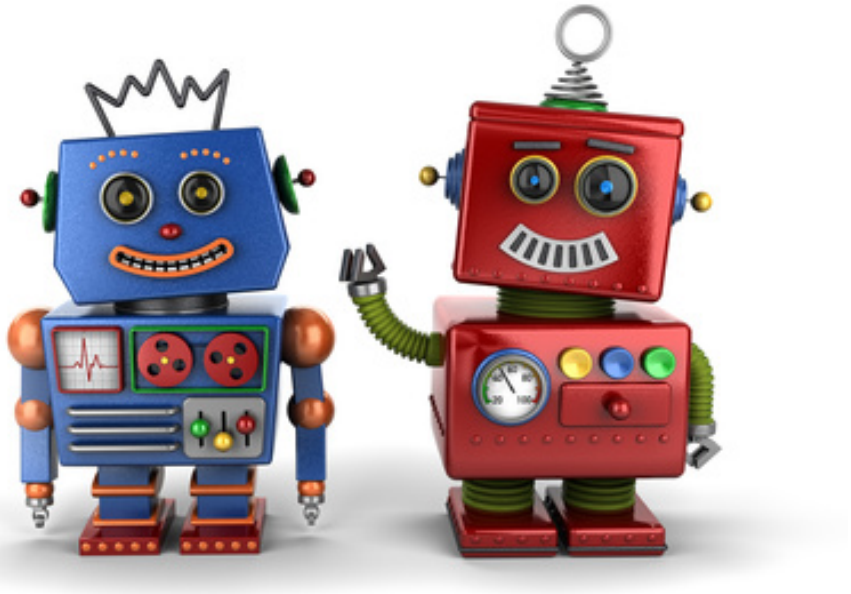
Accessing List Elements

- **Array-like access…**
  - …superficially!
- **Array access is *O(1)***
- **List access is *O(n)***
- **Watch out!**



List Module

- Functions as in Array and Seq modules
- List.map, List.iter, List.filter



List Mutability

- List as a whole is immutable
  - `let myList = [|8;6;7;5;3;0;9|]`
  - `myList <- [|8;7;7;5;3;0;9|]`
  - …unless you bind it as a mutable
- There are no Add… or Remove… methods
- Elements also immutable
  - `myList.[1] <- 7`
  - …but they might have mutable properties!

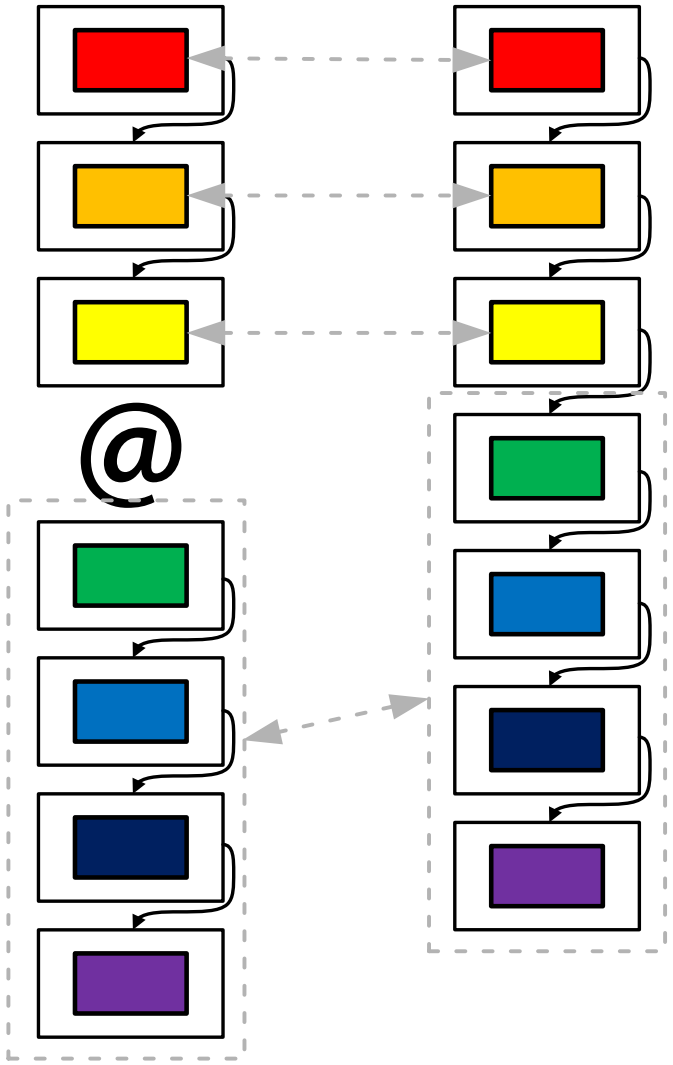
```
type MutableThing = { mutable Name : string }

let thingList =
    [ { Name = "Thing 1"}; { Name = "Thing 2"} ]

thingList.[1].Name <- "Thing 2 was changed"
```

Combining Lists

- Combine two lists with the @ operator
- Creates a new list
- Underlying elements the same in old and new lists

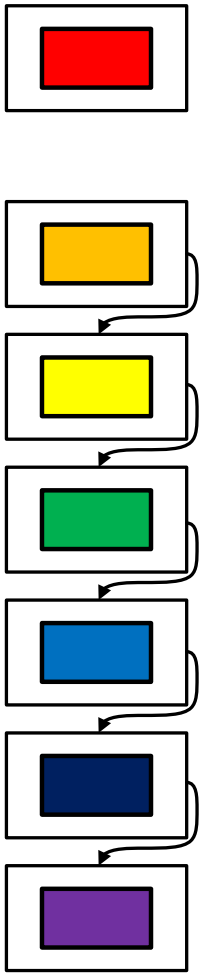
Combining Lists – What Really Happens

- **New list for first ½**
  - Same underlying elements
- **Last element of new first ½ points to first element of original second ½**
  - Same underlying elements



Prepending One Element

- Use the ‘cons’ operator :: to add element to the start
- Cons operations fundamental to List processing



What Does it Cost?

- **Indexed access is *O(n)***
  - Arrays are *O(1)*
- **Joining lists with @**
  …fairly expensive
  - New first list
- **Consing on a new element with ::**
  …very cheap
- **Consing off the first element with ::**
  …very cheap

Summary

- **An F# list is an immutable linked list**
- **Create like Arrays and Sequences**
- **Array-like access**
  - …but watch out for performance
- **Similar functions in List module as in Array and Seq modules**
- **Join two lists with @**
  - …watch performance
- **Adding/taking first element with ::**
  - …very fast