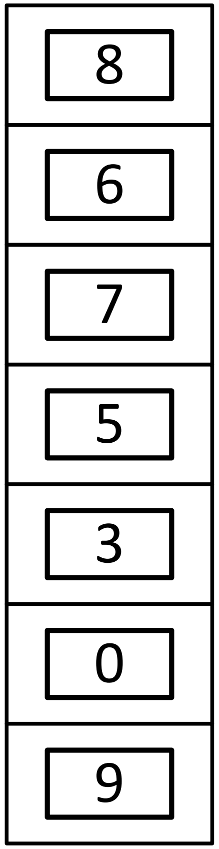
Arrays

Kit Eason
www.kiteason.com
@kitlovesfsharp

pluralsight
hardcore dev and IT training

What is an Array?

- Standard .NET type
- Length fixed on creation
- All elements of same type
- Array as a whole is immutable
  - `let myArray = [|8;6;7;5;3;0;9|]`
  - `myArray <- [|8;7;7;5;3;0;9|]` ✗
- Elements mutable
  - `myArray.[1] <- 7`



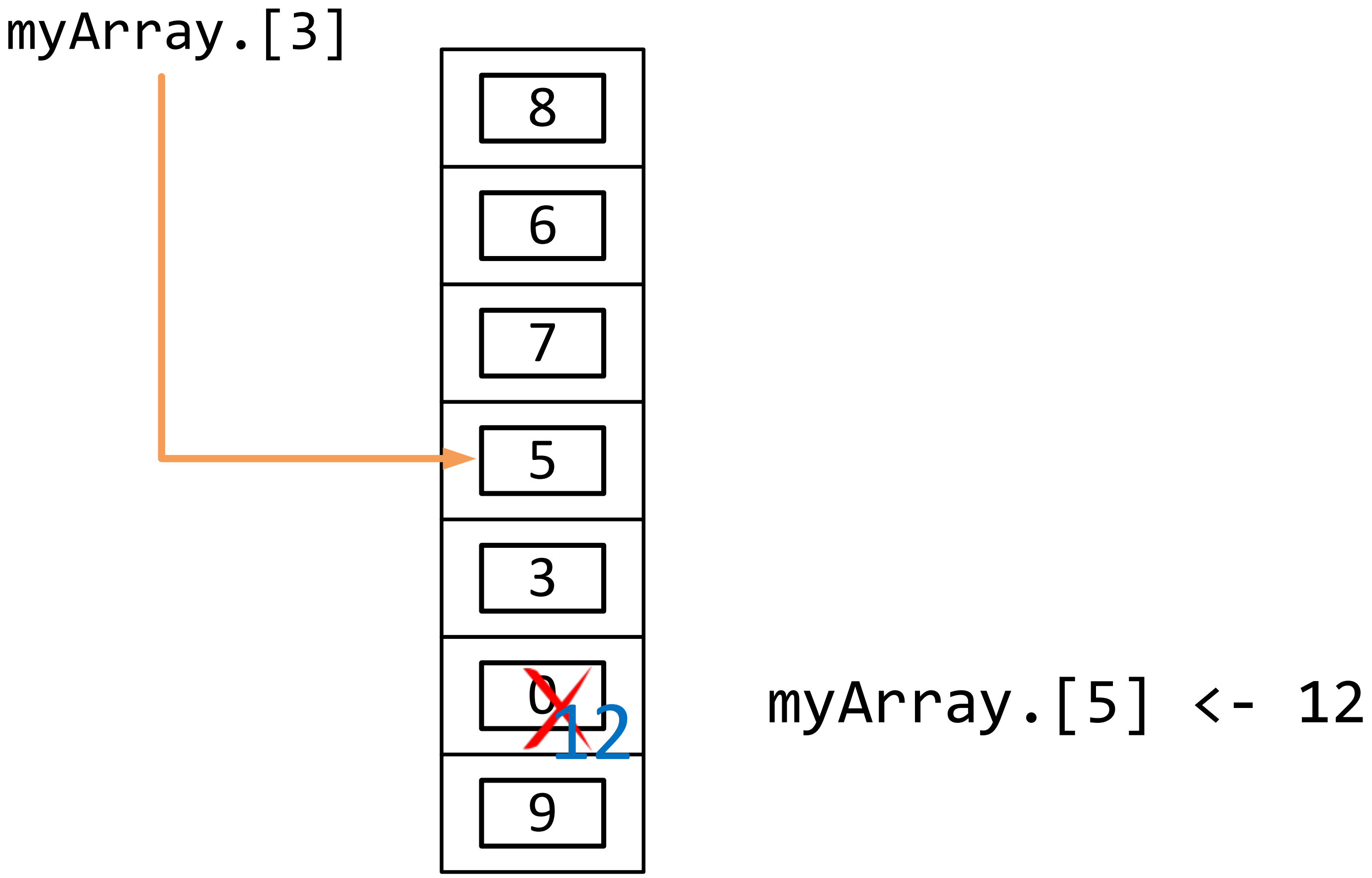
Creating an Array

- **From a literal**
  - `let primes = [|1; 3; 5; 7; 11|]`
- **From a comprehension**
  - `let someNumbers = [|1000..1020|]`

  -
    ```
    let smallEvens = [|for i in 1..100 do
                          if i%2 = 0 then yield i|]
    ```
- **Using a function from the Array module**
  - Array.create
  - Array.init
- **With zero-valued elements**
  - Array.zeroCreate
- **From another array or IEnumerable**

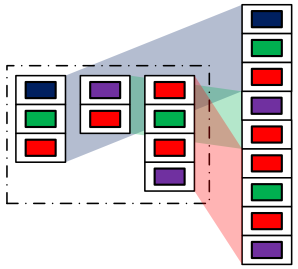
Accessing Array Elements

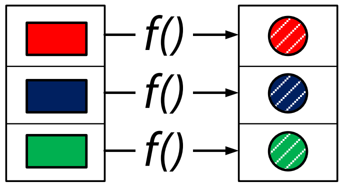
- .[index] notation
  - let myValue = myArray.[3]
- Update elements with <-

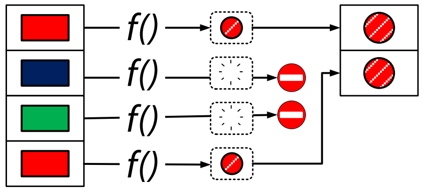


The Array Module

- Provides operations on arrays
- ≈70 different operations!
- No need to open a namespace
- Not implemented as methods/extension methods
- Equivalent operations for other structures





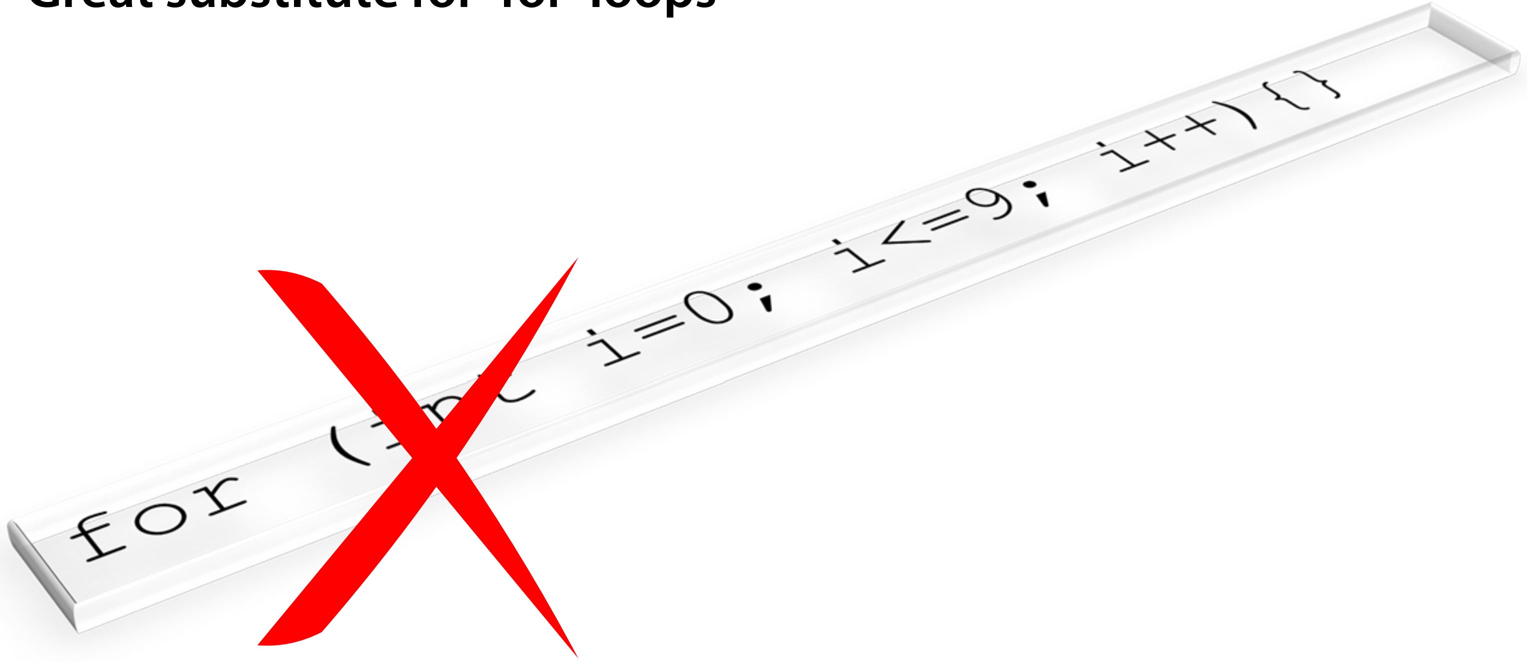


Array.map

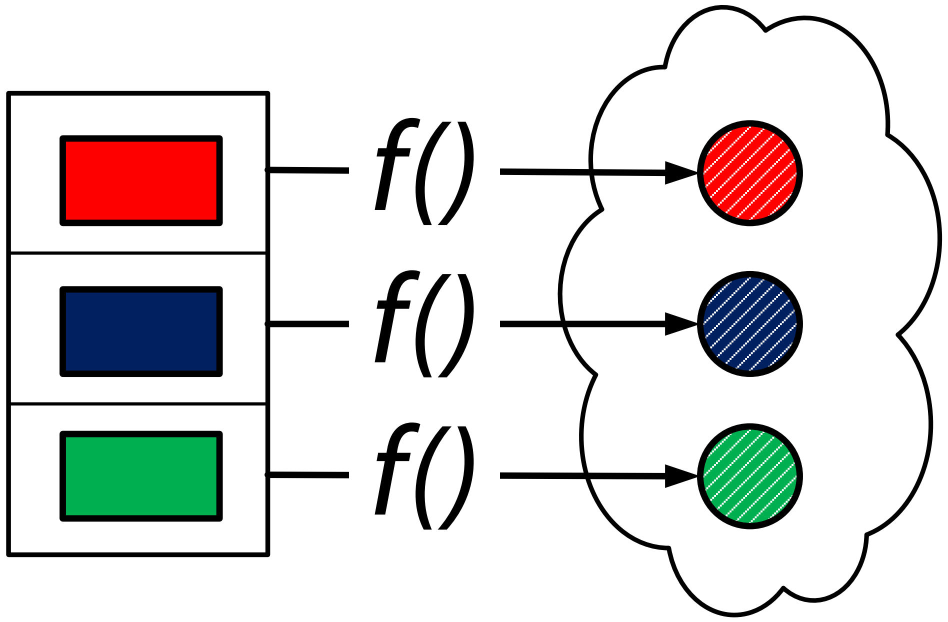
- Takes an array
- Returns another of same length
- Returned array has results of applying function to each element
- Input array unaffected
- Can have side effects too

Array.mapi

- Like array map
- Provides the index of each element
- Great substitute for 'for' loops



Array.iter



- Iterates and calls a function with each element
- Doesn’t return anything!
- Side effects only!
- Array.iteri if you need an index

Array.filter

- Takes an array
- Applies some function to each element
- Returns elements where function returned ‘true’

Option Data Type

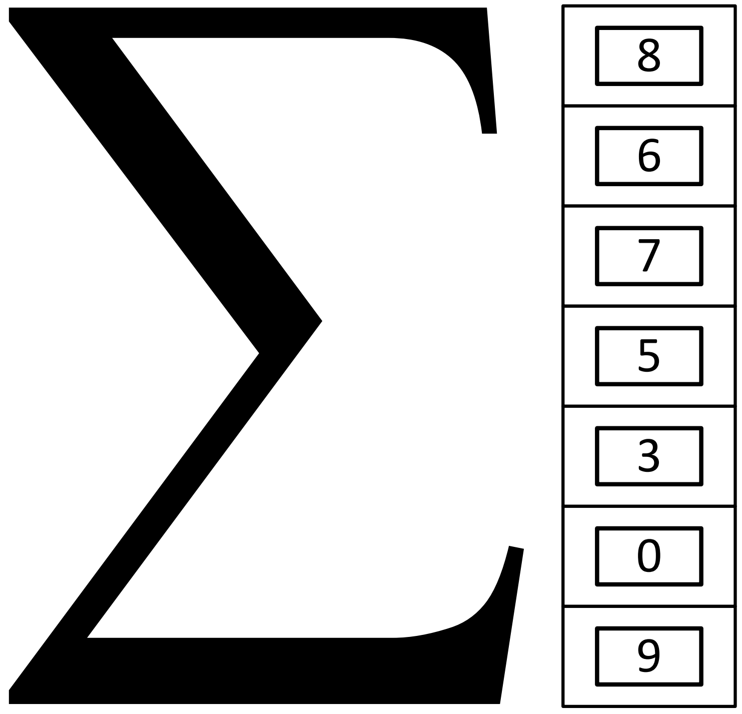
- **None means ‘no useful value’**
- **Some *value* means ‘we have a value and this is it’**

Array.choose

- **Takes an array of values**
- **Applies supplied function to each element**
  - Function must return an option type
- **Filters for ‘Some’ results**
- **Returns an array of the actual values**

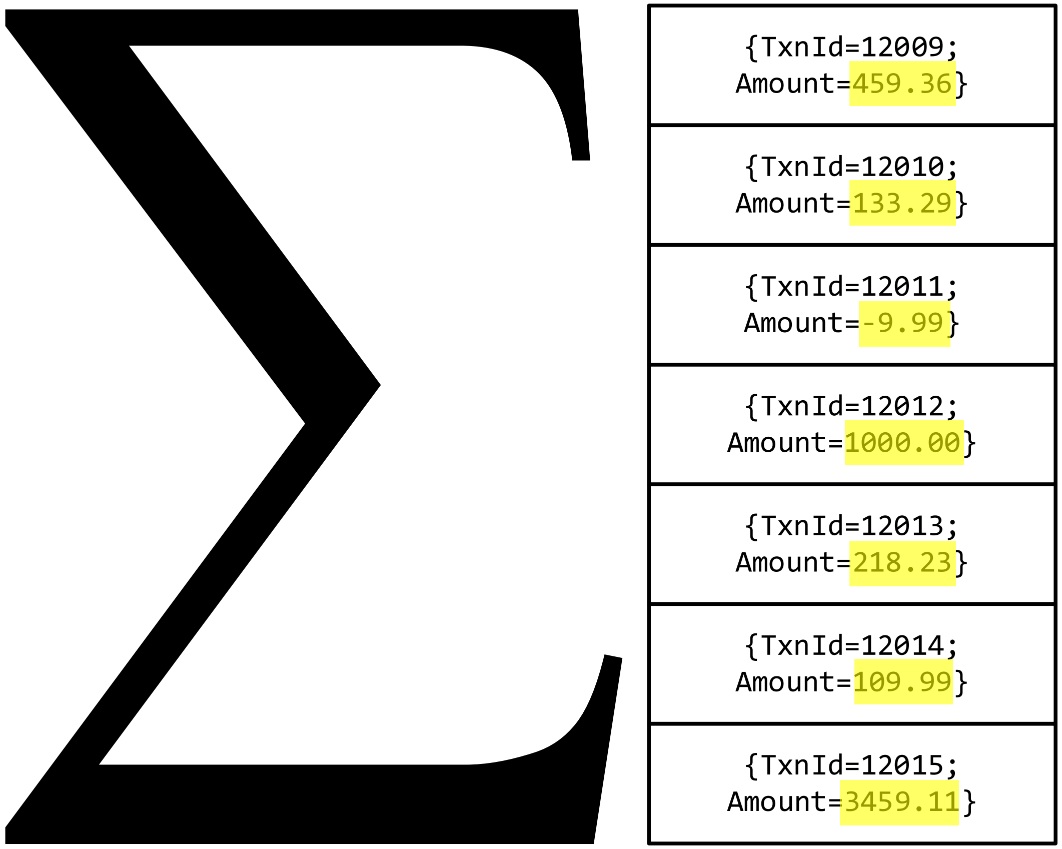
Array.sum

- **Returns the total value**
- **Type must support addition**
  - Uses the applicable + operator
- **Elements must have a Zero member**



Array.sumBy

- Takes a function
- …which returns some aspect of the elements to total



```
{TxnId=12009;
Amount=459.36}
```

```
{TxnId=12010;
Amount=133.29}
```

```
{TxnId=12011;
Amount=-9.99}
```

```
{TxnId=12012;
Amount=1000.00}
```

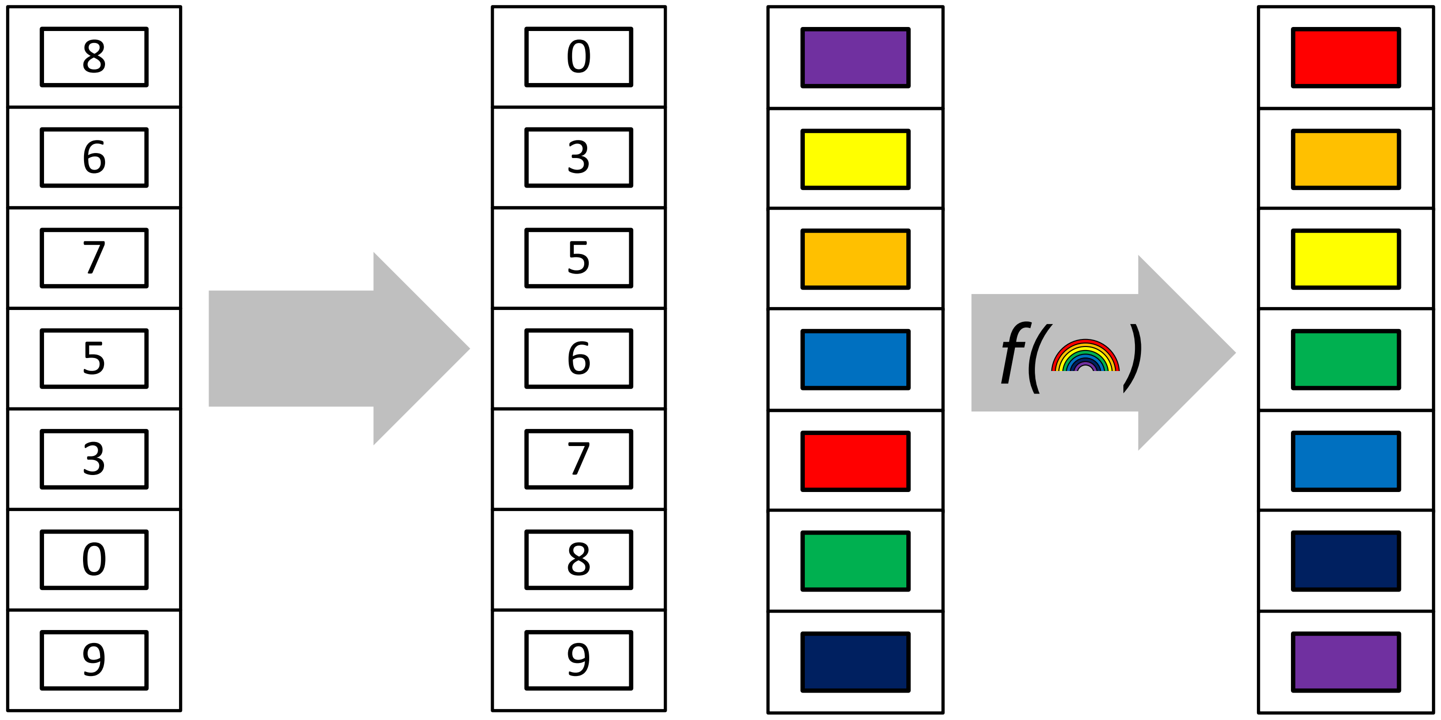
```
{TxnId=12013;
Amount=218.23}
```

```
{TxnId=12014;
Amount=109.99}
```

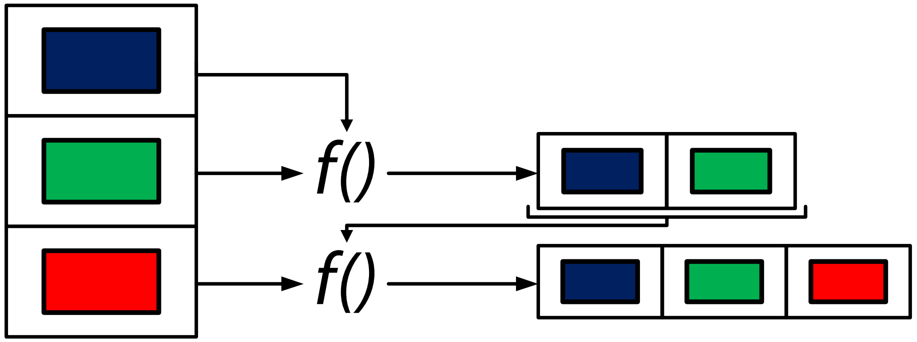
```
{TxnId=12015;
Amount=3459.11}
```

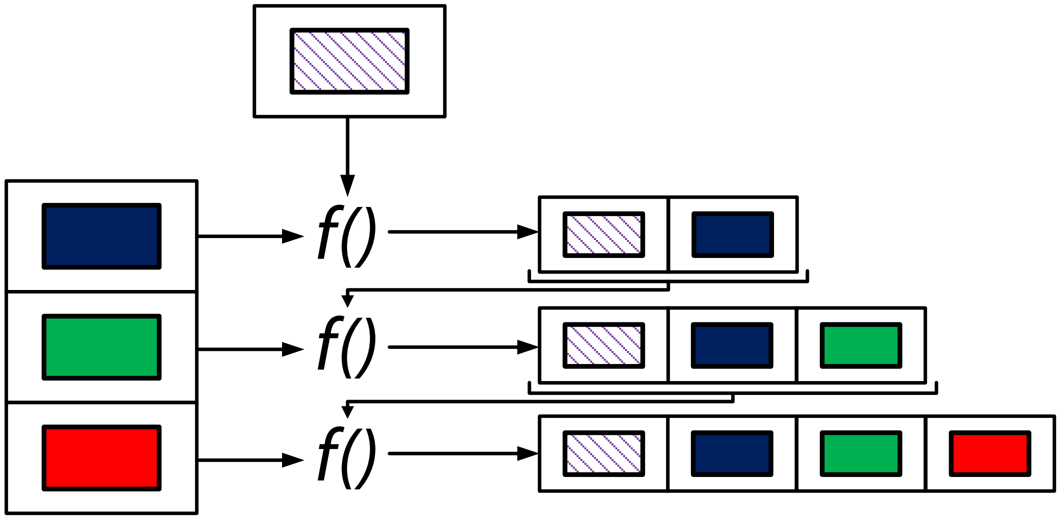
Array.sort/Array.sortBy

- Array.sort sorts by the whole element
- Array.sortBy - specify own function to get something to sort by



Array.reduce and Array.fold





Array.reduce

- **Takes an array**
- **Creates a 'state' based on first element**
- **For each following element**
  - Combines the state and the current element in some way
  - Returns another state which is passed into the next iteration
- **Returns the final accumulated state**

Array.fold

- **Starts with**
  - Array
  - Some initial state e.g. a string prefix
- **Applies function to each element**
  - Combines state and first element in some way
  - Returns another state which is passed into the next iteration
- **Returns final accumulated state**

Interlude: Parameter Naming

- Array.fold and reduce - simpler if you use ‘acc elem’ naming
- Stick to i for mapi and iteri
- Keep lambda arg names short

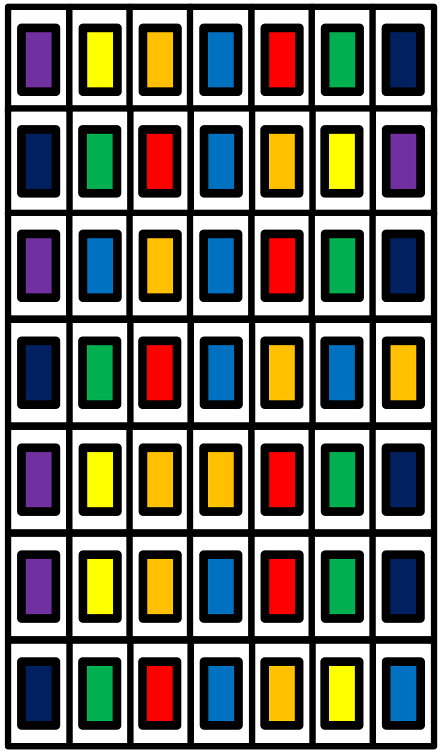
Array.zip

- Takes two equal length arrays
- Produces an array of the same length
- Tuples of elements from each input array
- Nothing to do with compression!
- Array.zip3
- Array.unzip

Multi-Dimensional Arrays

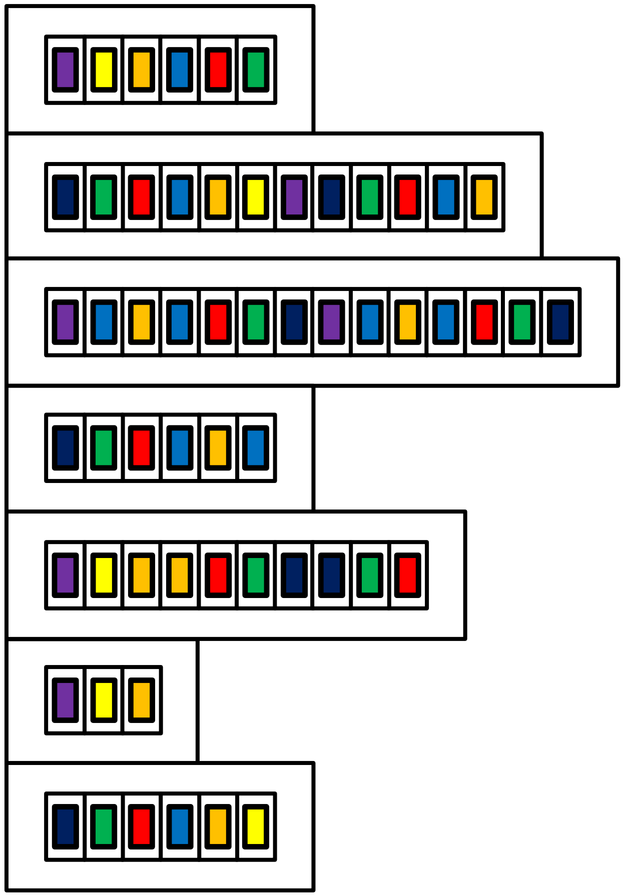
- **Rectangular array**
  - Rows all same length, columns all same length
- **Array2D module**

  -
    ```
    let my2D =
        Array2D.init 12 12
          (fun x y -> (x+1) * (y+1))
    ```
- **Accessing elements**
  - `printfn "%i" my2D.[3, 4]`
- **Array3D, Array4D modules**



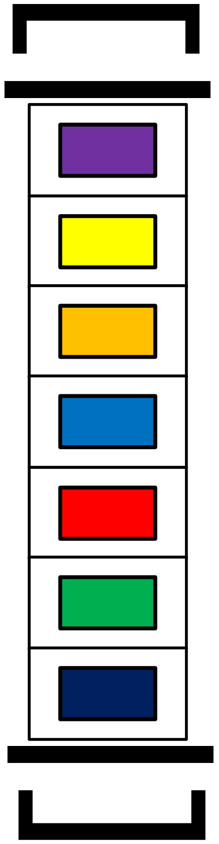
Jagged Arrays

- An array of arrays
  - Not all inner arrays same length
- Creating a jagged array
  - `let jagged1 = [| [|1; 3; 9|];  [|4; 6|] |]`
  - `let jagged2 = Array.init 3 (fun x -> [|0..x|])`
- Accessing elements
  - `printfn "%i" jagged1.[0].[2]`
- Type annotations
  - 2D array: int [,]
  - Jagged array: int [] []



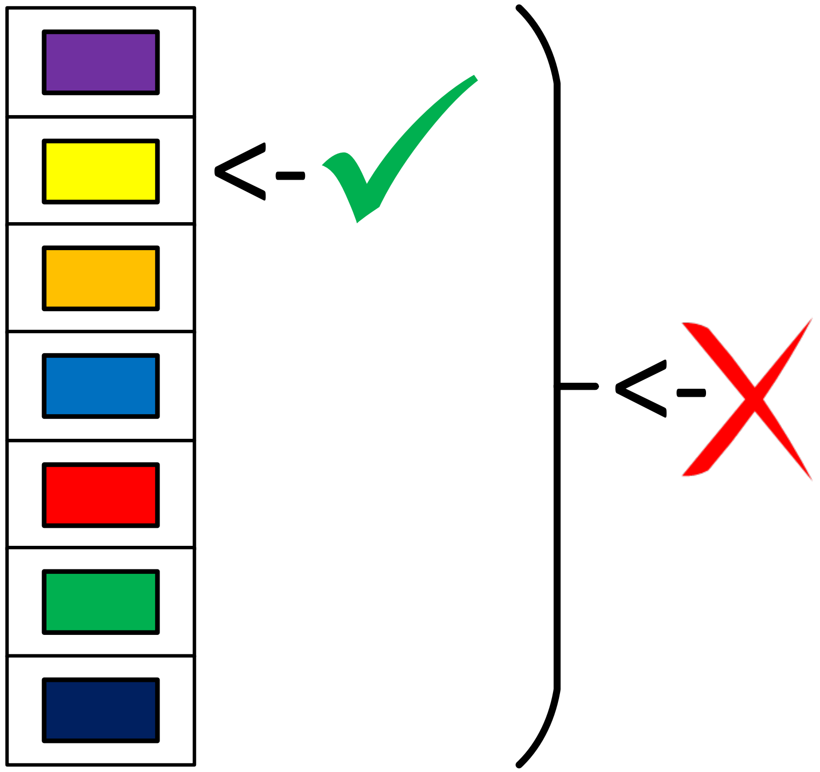
Summary

- Fixed length (except ResizeArray)



Summary

- Immutable by default *as a whole*
- Mutable *elements* by default



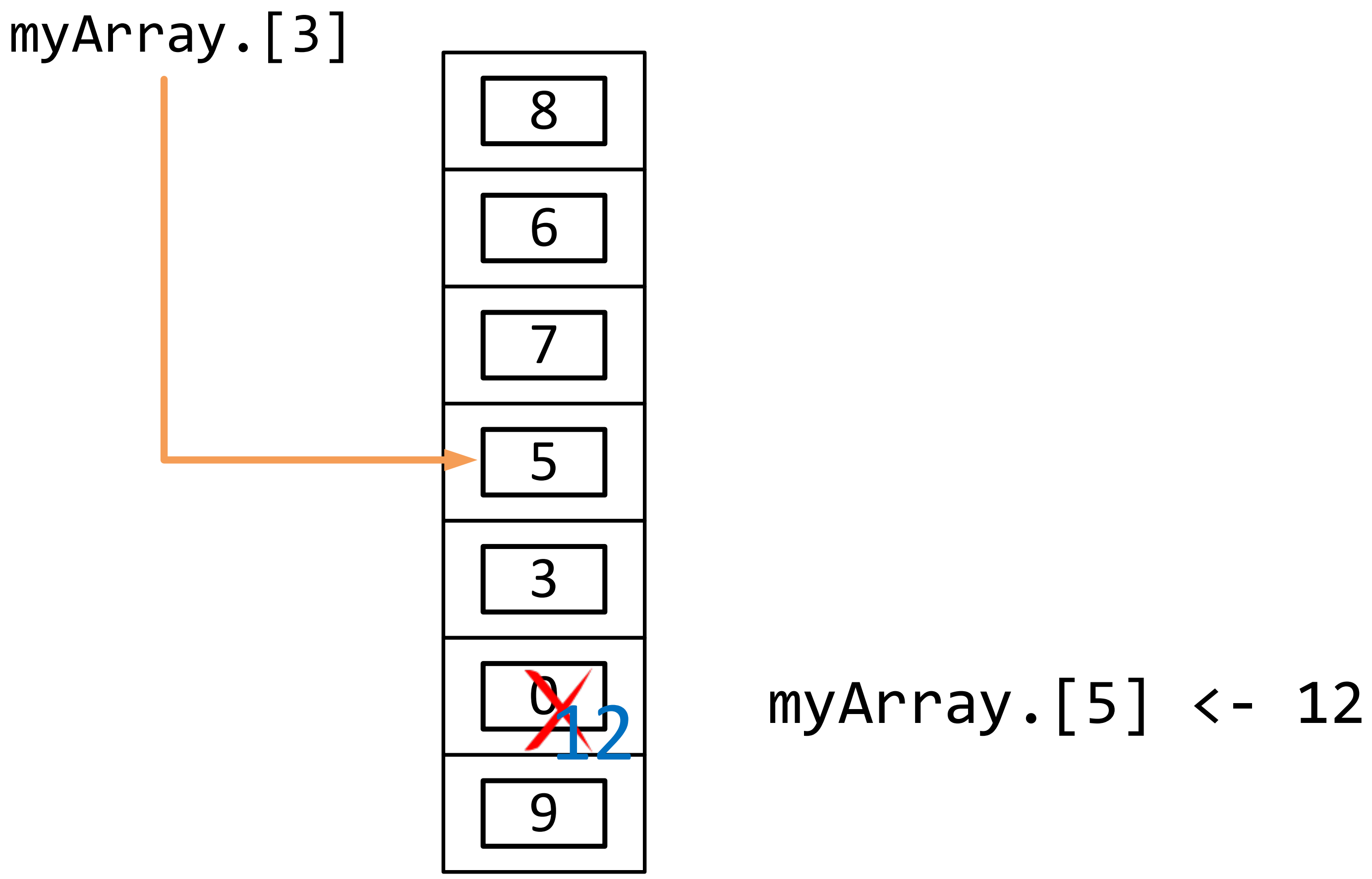
Summary

- **Create with… literal**
  - `let arr = [|1; 2; 5|]`
- **…comprehension**
  - `let someNumbers = [|1000..1020|]`

  -
    ```
    let smallEvens =
        [|for i in 1..100 do if i%2 = 0 then yield i|]
    ```
- **…Array.create, Array.zeroCreate, Array.init**
  - `let squares = Array.init 10 (fun x -> x * x)`
- **…from another collection**
  - `let arr = myList |> Array.ofList`

Summary

- **Access elements with .[index] notation**



Array Module

- **Contains about 70 useful functions**
- **Must understand at least:**
  - Array.map
  - Array.iter
  - Array.filter
- **Doing something to an array? Look at the Array module first!**