Videregående F#

Programmering og problemløsning

Jon Sporring

Doven evaluering: lazy

- Erklæring: let name = lazy (expr)
- Tvungen beregning: name.Force()
- Beregning gjort en og kun en gang

```
let a = 2;
let b = lazy (printfn "calulating"; a*a);
printfn "Before: %A" b
b.Force()
printfn "After: %A" b
b.Force()
printfn "After repeat: %A" b
```

lazy.fsx

Sekvenser (seq)

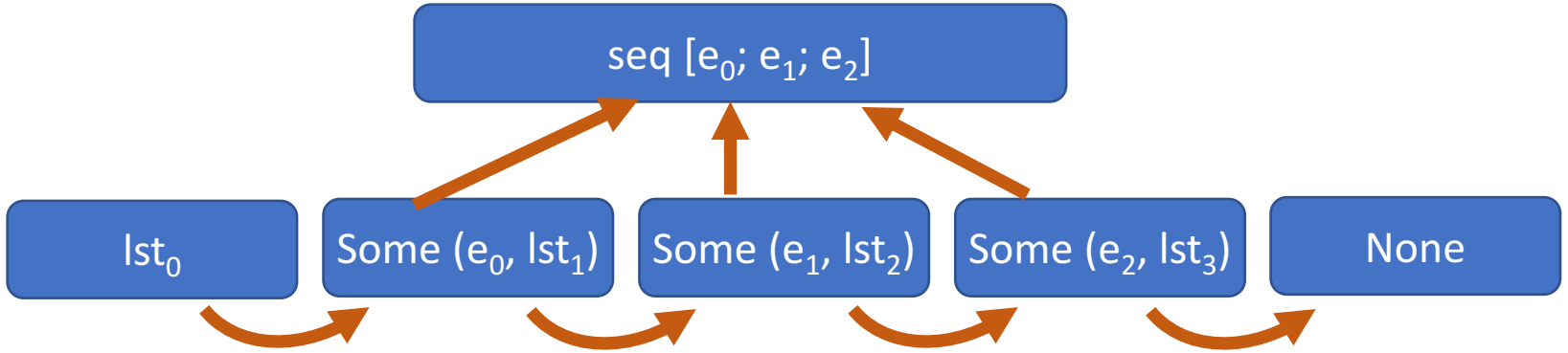
- En sekvens (sequence) er en ordnet liste af elementer af samme type.
- Type: seq<'a> eller System.Collections.Generic.Ienumerable
- Støttefunktioner: Seq modulet
- Lazy evaluering!
- Eksempler:

```
seq { 0 .. 10 .. 100 }
seq { for i in 1 .. 10 do yield (float i) ** 3.0 }
seq { for i in 1 .. 10 -> (float i)**3.0 }
seq [1; 2; 3]
```

seqComputationExpr.fsx

seqDelayedEval.fsx

Seq.unfold : ((‘a -> (‘b * ‘a) option) -> ‘a -> seq<‘b>



- Fibonacci

```
let fib (n,m) = if m < 1000 then Some (n+m, (m, n+m)) else None;;
let a = Seq.unfold fib (1,1);;
Seq.toList a;;
```

- Sekvenser er dovne, så vi beregner kun dem, vi bruger:

```
let fib (n,m) = Some (n+m, (m, n+m));;
let a = Seq.unfold fib (1,1);;
a |> Seq.take 15 |> Seq.toList;;
```

Uendelige sekvenser og yield! (bang)

- Elementer i sekvenser bliver udregnet efter behov (lazy evaluation)

```
let repeat items =  seq { while true do yield items }
printfn "%A" (repeat [1;2;3]);;
```

- Bang:

```
let repeat items =  seq { while true do yield! items }
printfn "%A" (repeat [1;2;3]);;
```

- Function with bang:

```
let rec repeat items = seq { yield! items; yield! repeat items }
printfn "%A" (repeat [1;2;3]);;
```

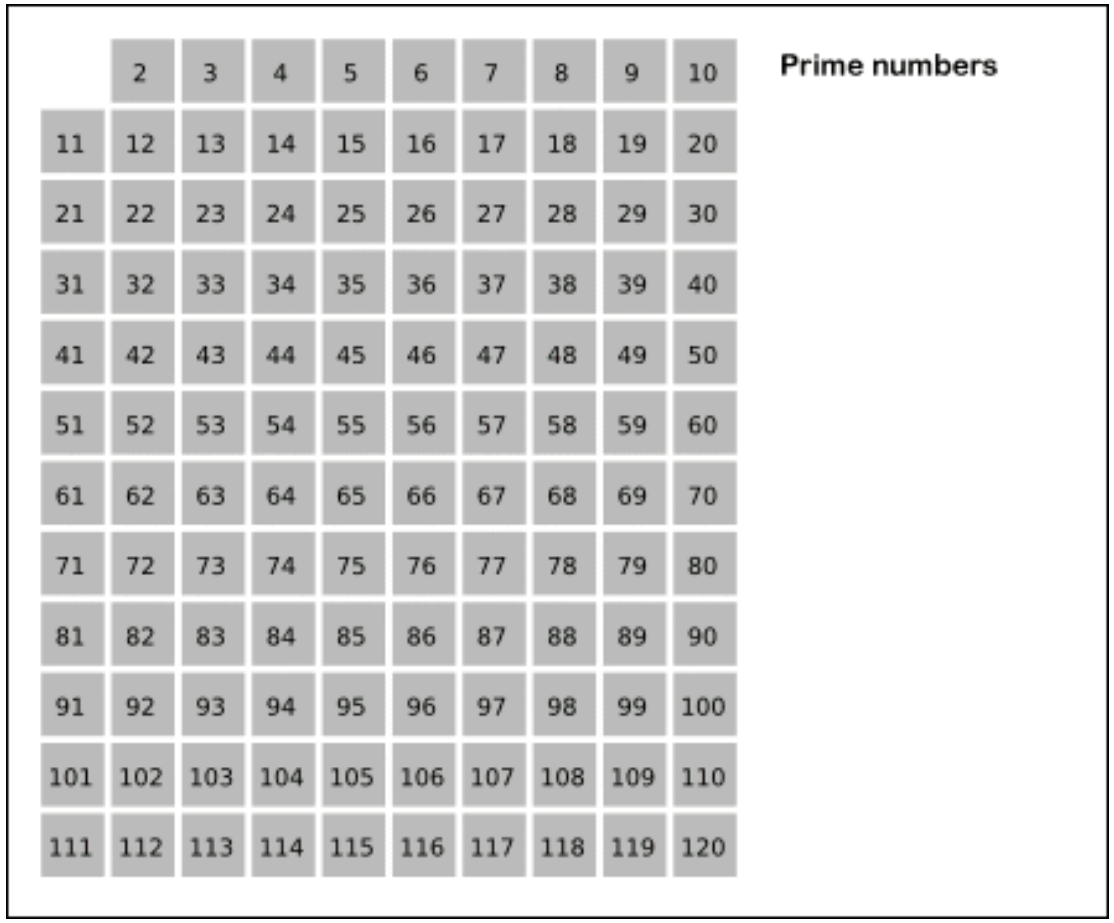
- Seq.initInfinite:

```
let s = Seq.initInfinite (fun i -> (float i)**2.0)
printfn "%A"  s;;
```

Prime: heltal n > 2 kun deleligt med 1 og n

```
let isPrime n =
  let sqrtInt = float >> sqrt >> int
  List.forall (fun x -> n % x <> 0) [ 2 .. sqrtInt n ]
let s = Seq.initInfinite (fun i -> i+2)
let primes = Seq.filter isPrime s
for i = 0 to 299 do printf "%d " (Seq.item i primes)
printfn "";;
```

Eratosthenes' sigte



https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes

```
let s = Seq.initInfinite (fun i -> i+2)
let sieve n = Seq.filter (fun v -> v%n <> 0)
let step s =
   let h = Seq.head s
   Some(h, sieve h (Seq.tail s))
let primes = Seq.unfold step s
for i = 0 to 299 do printf "%d " (Seq.item i primes)
printfn "";;
```

Asynkrone beregninger: async

```
let comp = async {
  for i = 1 to 100 do
    printfn "%d" i;
    do! Control.Async.Sleep 100;
};;
Control.Async.RunSynchronously comp;; // await
resultControl.Async.Start comp;; // do not await result
```

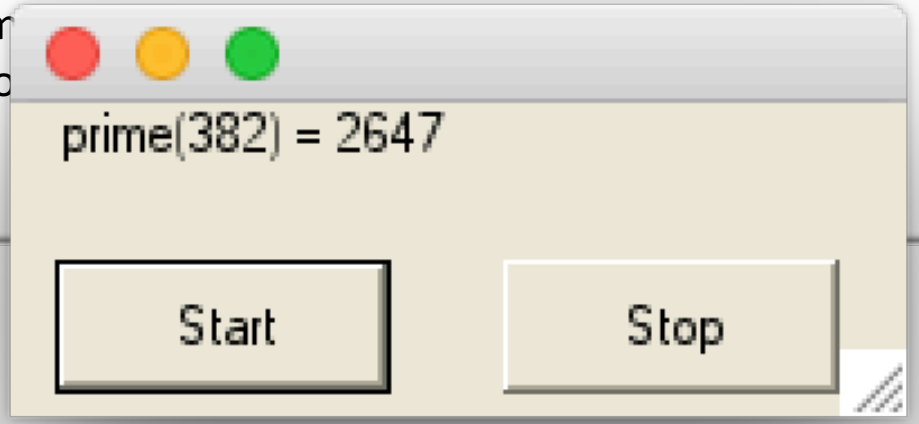
Asynkrone beregninger: Cancel event

```
let cancellationSource = new System.Threading.CancellationTokenSource()
let comp = async {
  for i = 1 to 100 do
    printfn "%d" i;
    do! Control.Async.Sleep 100;
}
Control.Async.Start (comp, cancellationSource.Token) // allow cancellation
System.Threading.Thread.Sleep(200) // do nothing in 200ms
cancellationSource.Cancel();; // cancel comp
```

Parallelbe

```
let fetchUrl url =
  let req = WebR
  use resp = req
  use stream = r
  use reader = n
  let html = reac
  printfn "finishe
```

```
Jons-mac:20180119SeqNAsync sporring$ i=webDownload; fsharpc $i.fsx && mono32 $i.exe
Microsoft (R) F# Compiler version 4.1
Copyright (c) Microsoft Corporation. All Rights Reserved.
finished downloading http://www.bing.com
finished downloading http://www.google.com
finished downloading http://www.microsoft.com
finished downloading http://www.amazon.com
finished downloading http://www.yahoo.com
00:00:03.2283960 msec
Jons-mac:20180119SeqNAsync sporring$ i=asyncWeb; fsharpc $i.fsx && mono32 $i.exe
Microsoft (R) F# Compiler version 4.1
Copyright (c) Microsoft Corporation. All Rights Reserved.
finished downloading http://www.microsoft.com
finished downloading http://www.google.com
finished downloading http://www.bing.com
finished downloading http://www.yahoo.com
finished downloading http://www.amazon.co
00:00:01.5359210 msec
```



webDownload.fsx vs. asyncWeb.fsx

buttonControl.fsx