Implementing Functional Reactive Programming in Scala

Remember this?

```
Behaviour[T] = Time => T
```

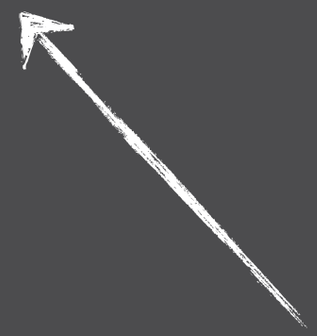
```
Event[T] = Stream[(Time, T)]
```

Lets use them...

```
val beh = new Behaviour(time => Color.red)
```

```
beh.at(now)
```

```
def now = System.currentTimeMillis
```



```
beh.at(now)
```

```
beh.map(val => val.toString)
```

```
val event = new Event[Boolean]
```

```
beh.until(event, beh1)
```

```
"Hi!".toggle(event, "Bye!")
```

“Hi!”.toggle(event, “Bye!”)

?

```
implicit def lift[T](value : T) = {
        new Behaviour(time => value)
}
```

```
implicit def lift[T](value : T) = {
        new Behaviour(time => value)
}
```

So Far:

So Far:

- Core FRP: Behaviour/Event

So Far:

- Core FRP: Behaviour/Event
- UI Framework

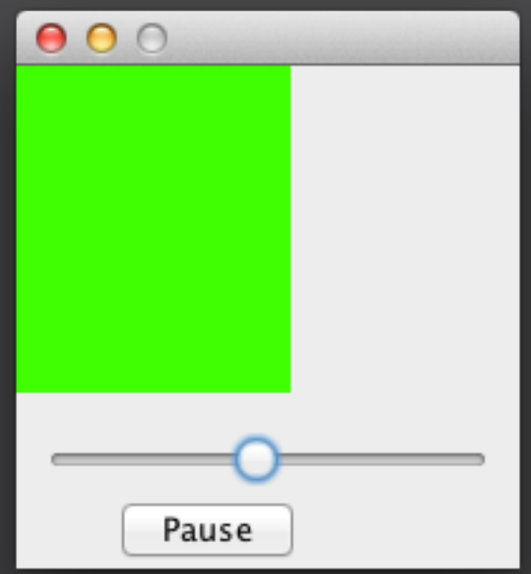
So Far:

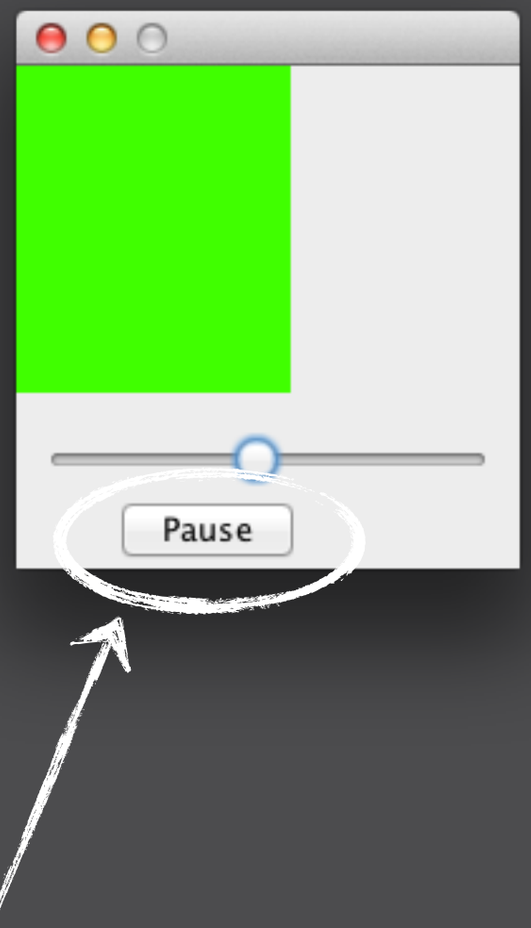
- Core FRP: Behaviour/Event
- UI Framework
- I/O Framework

Everything:

http://github.com/oetzi/echo

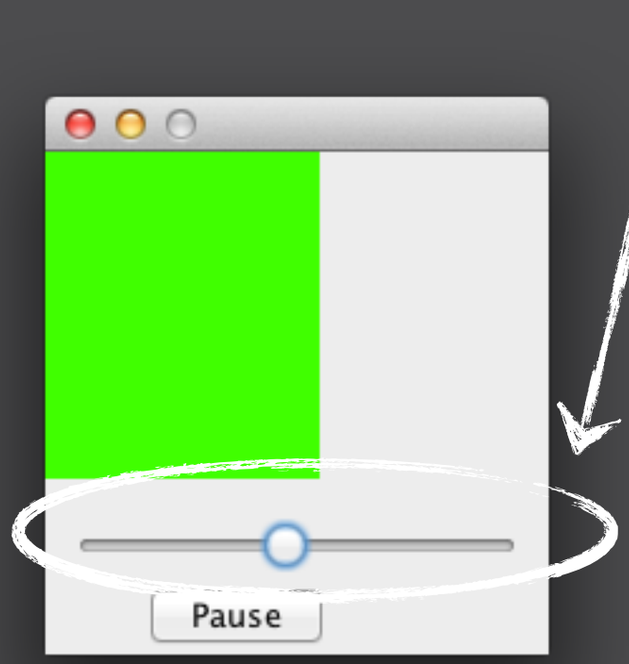
An example...


Pause

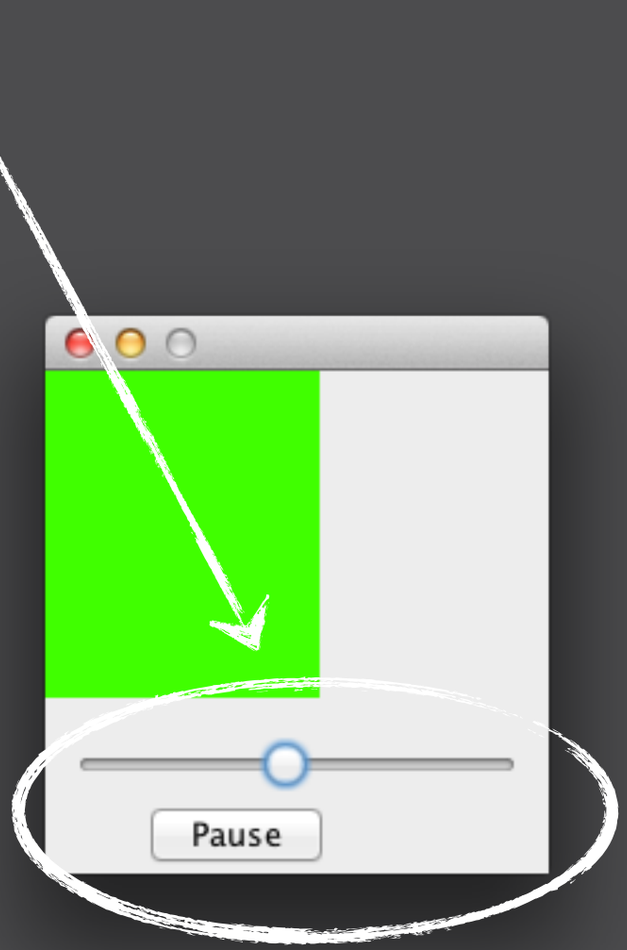


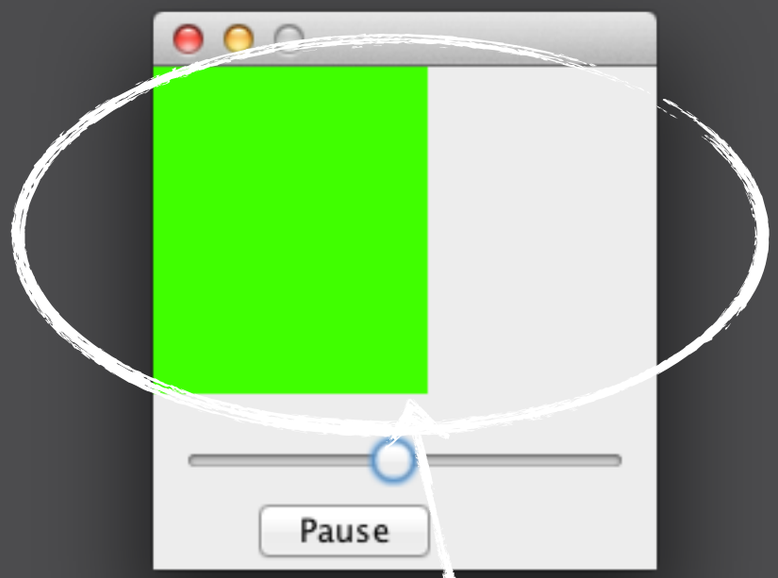

```
val playButton = Button(button => "Play".toggle(button.click, "Pause"))
```

```
val slider = Slider()
```

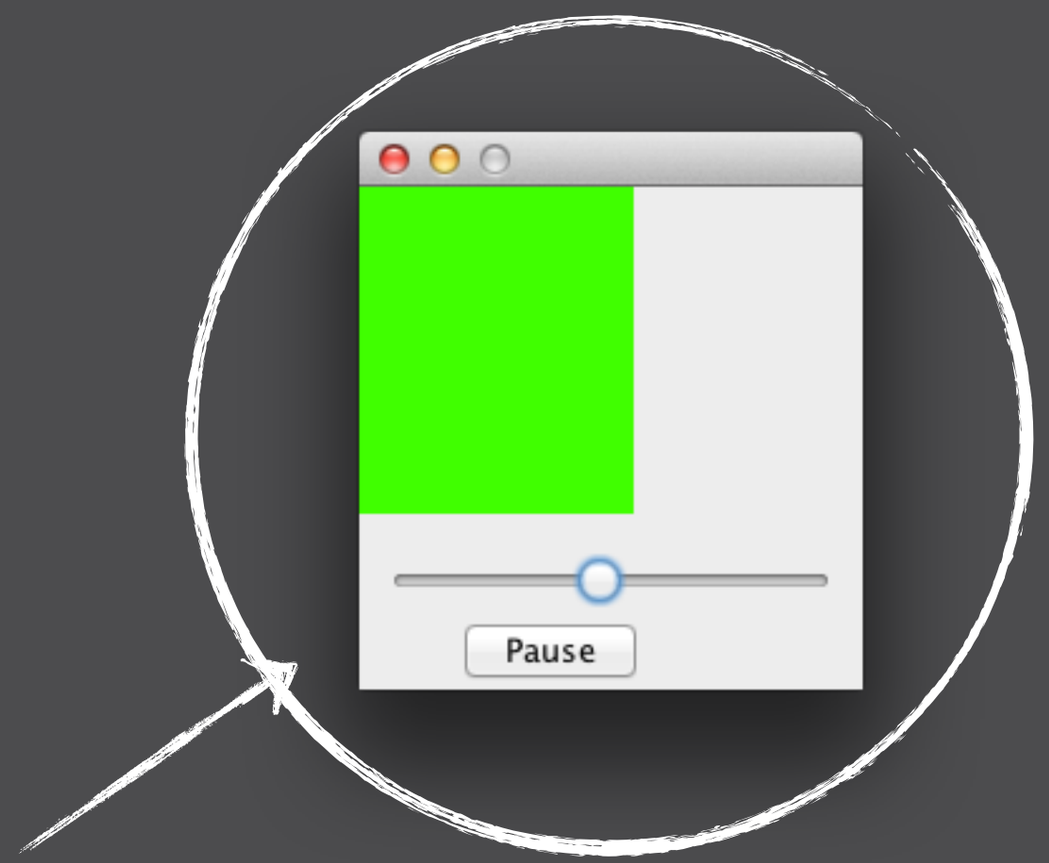


```
val playing = false.toggle(playButton.click, true)
val song = new Song(file, playing, slider.value)
```






```
val anim = song.volume.map1(width) {
  (v, w) => math.min(v * (w / 50F), w)
}
val bouncer = Block(anim, height - 70, Color.green)
```




```
val frame = Frame(width, height, List(
  bouncer,
  slider,
  playButton
))
```

What if something goes wrong?

```
class Dangerous extends Breakable {
    dangerous { () =>
        break_everything()
    }
}
```

```
class Dangerous extends Breakable {
    dangerous { () =>
        break_everything()
    }
}

val danger = new Dangerous()
val err : Event[Exception] = danger.errors
```


INSIDE
DOING IT LIVE

Thank You!