

Funktionale Reaktive Programmierung (FRP)

mit reactive-banana-0.6.0.0

Heinrich Apfelmus

Funktionale Reaktive Programmierung (FRP)

mit reactive-banana-0.6.0



Heinrich Apfelmus

Warum?

Funktionale Reaktive Programmierung ist eine elegante Methode zur programmierung von interaktiver Software

- Graphische Benutzeroberflächen (GUI)
- Animationen
- Digitale Musik
- Robotik

Wie?

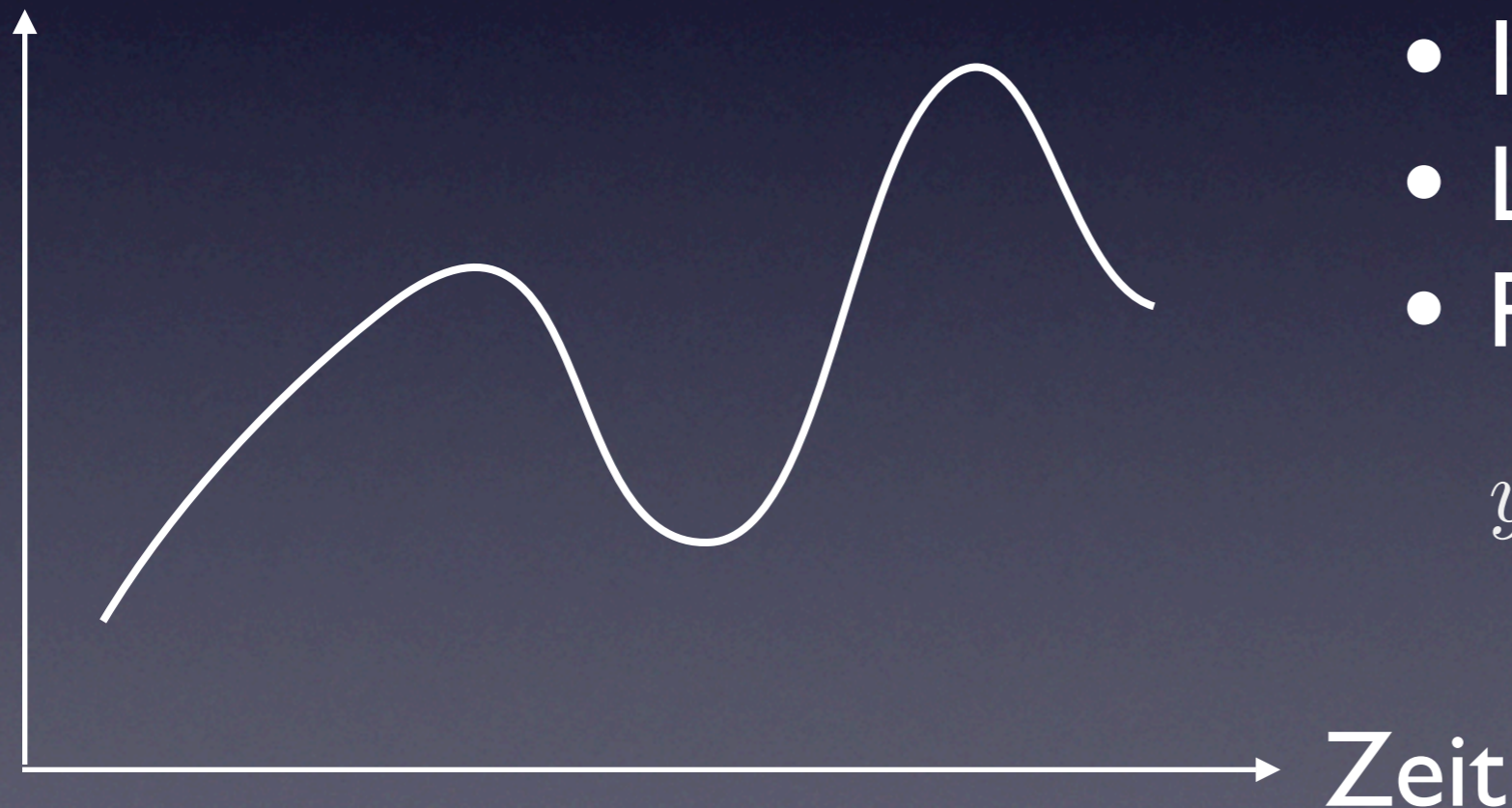
zeitliches Verhalten als
first-class value

```
type Behavior a = Time → a
type Event a    = [(Time, a)]
```

Behavior

```
type Behavior a = Time → a
```

Wert



- Position – Animation
- Inhalt Textfeld – GUI
- Lautstärke – Musik
- Physikalische Größe

$$y(t) = y_0 + v_0 t - g \frac{t^2}{2}$$

Behavior API

```
instance Functor Behavior
```

Functor

```
instance Applicative Behavior
```

Applicative

Behavior API

```
(<$>) :: (a -> b)  
-> Behavior a -> Behavior b
```

Functor

```
pure :: a -> Behavior a  
<*> :: Behavior (a -> b)  
-> Behavior a -> Behavior b
```

Applicative

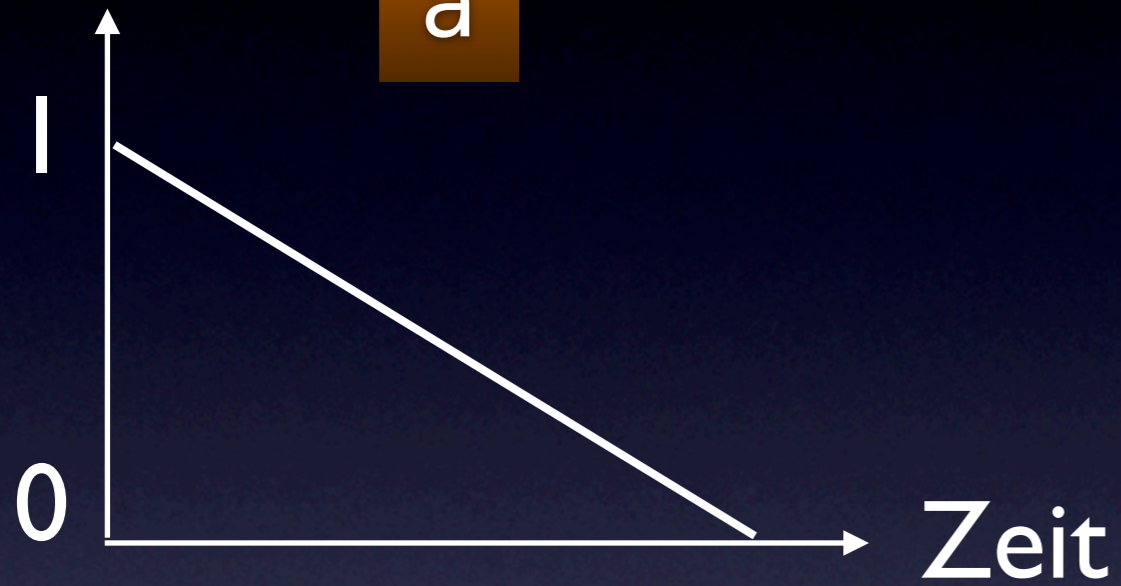
```
bf <*> bx =  
  \time -> bf time $ bx time
```

Für jeden
Zeitpunkt

Behavior API

Double

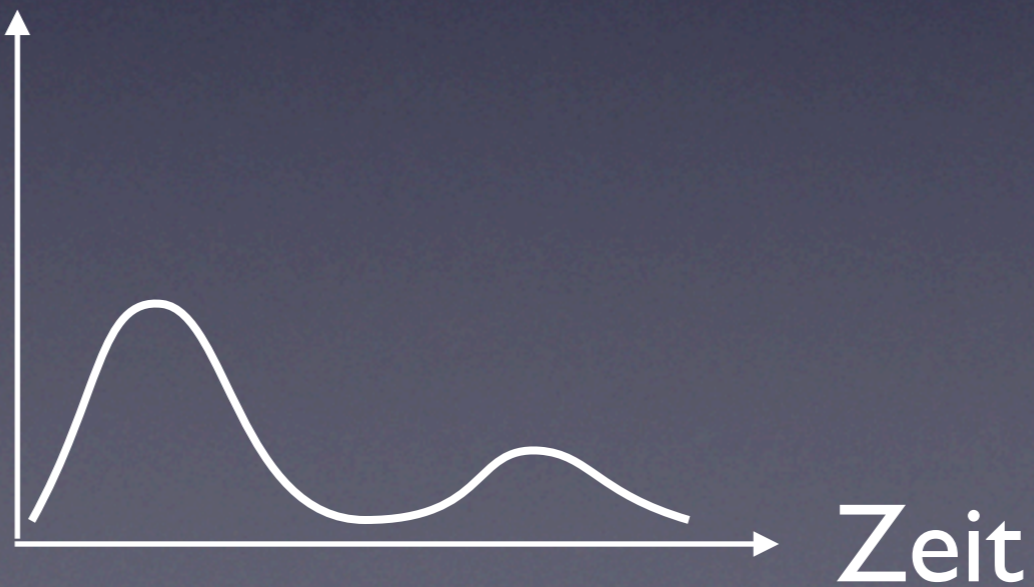
a



b



(*) <\$> a <*> b



Event

```
type Event a = [(Time, a)]
```

Wert



- Mausklicks – GUI
- Noten – Musik
- Stöße – Physik

Event API

```
instance Functor Event
```

Functor

```
never      :: Event a  
unionWith :: (a -> a -> a)  
           -> Event a -> Event a -> Event a
```

List

[]

zipWith

```
filterE   :: (a -> Bool)  
           -> Event a -> Event a
```

filter

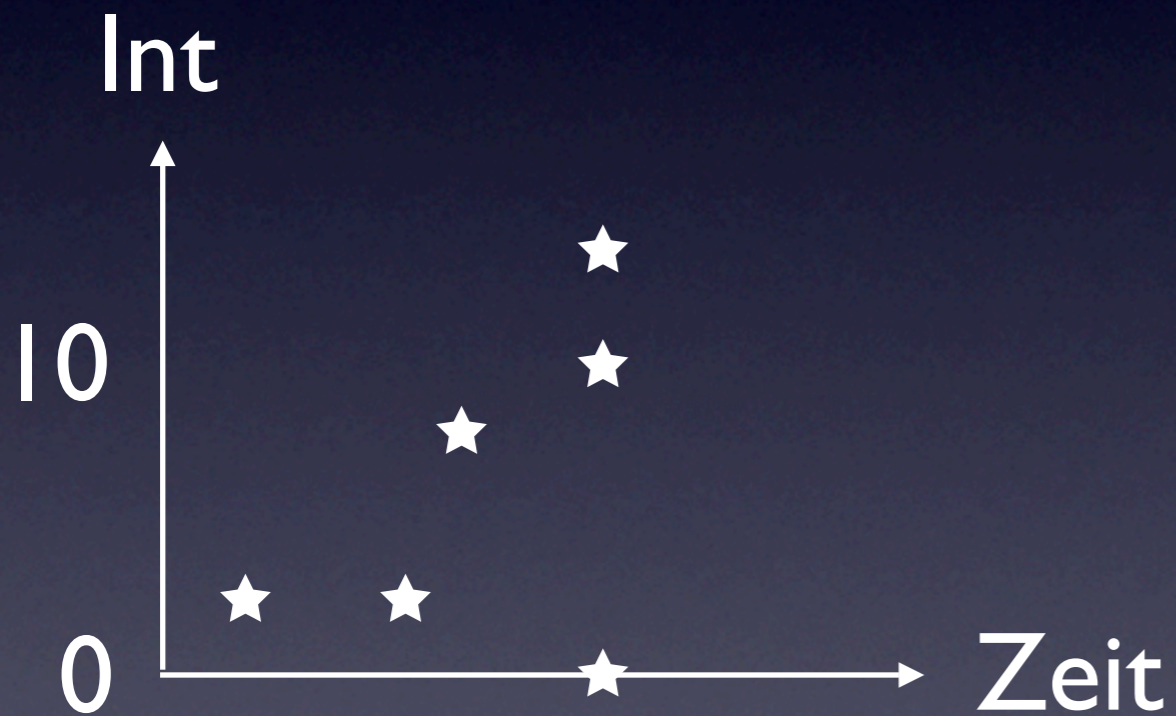
```
accumE    :: a -> Event (a -> a)  
           -> Event a
```

scanl

Event API

x

`filterE (> 5) x`



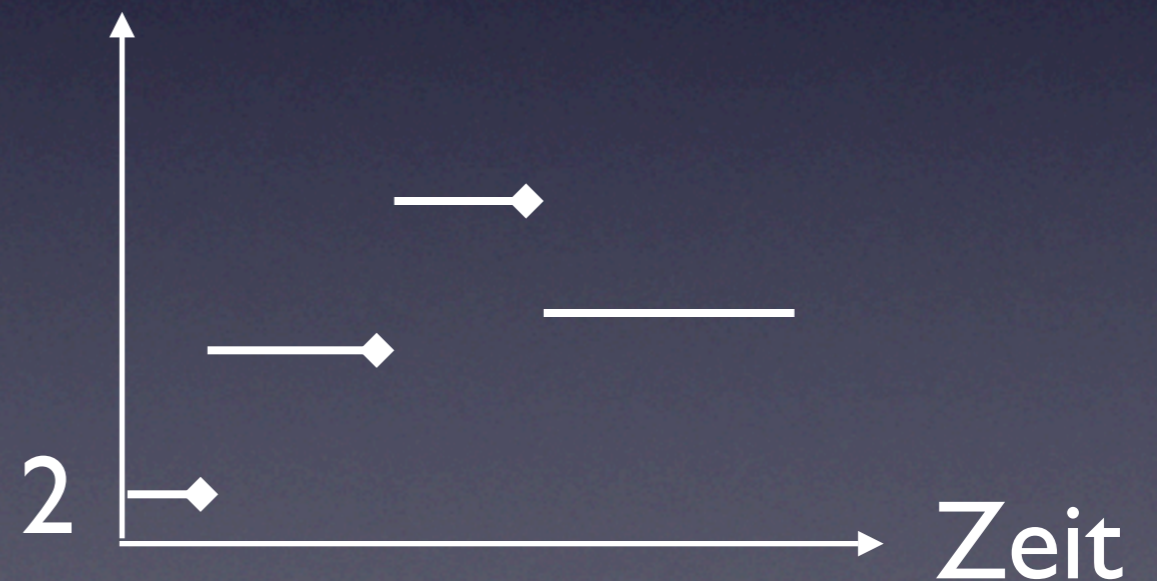
Event & Behavior API

stepper :: a -> Event a -> Behavior a

x



stepper 2 x



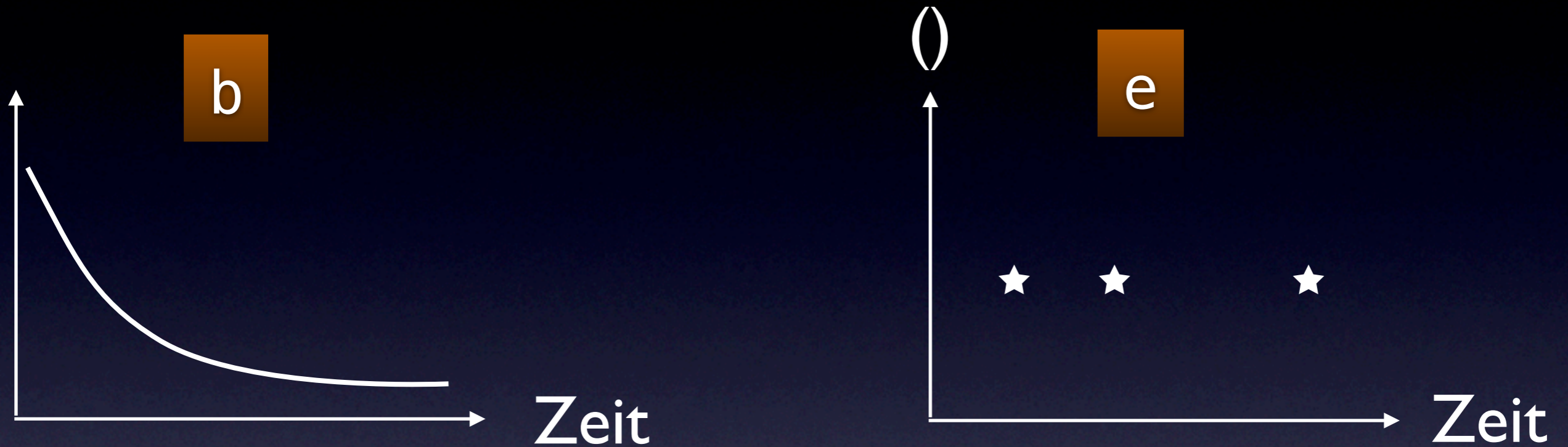
Event & Behavior API

```
(<@>) :: Behavior (a -> b)  
      -> Event a -> Event b
```

```
(<@)  :: Behavior b  
      -> Event a -> Event b
```

„apply“

Event & Behavior API



b <@ e



Frameworks (GUI, ...)

data NetworkDescription t a

fromAddHandler

Events importieren

fromPoll

Behaviors importieren

reactimate

Event exportieren

changes

Event aus Behavior erhalten