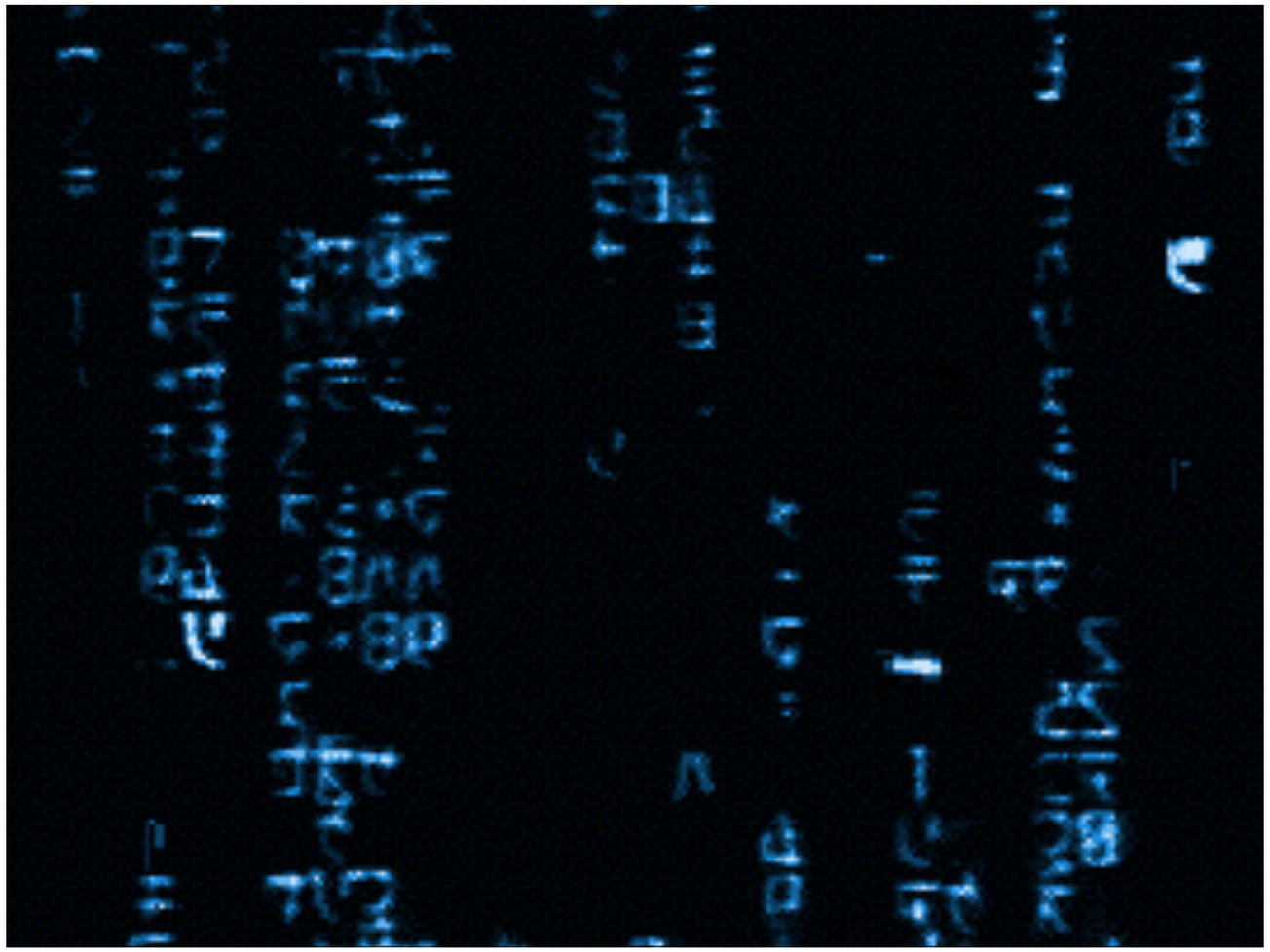
Reaktiv programmering

Sven A Robbestad
Inmeta Consulting AS

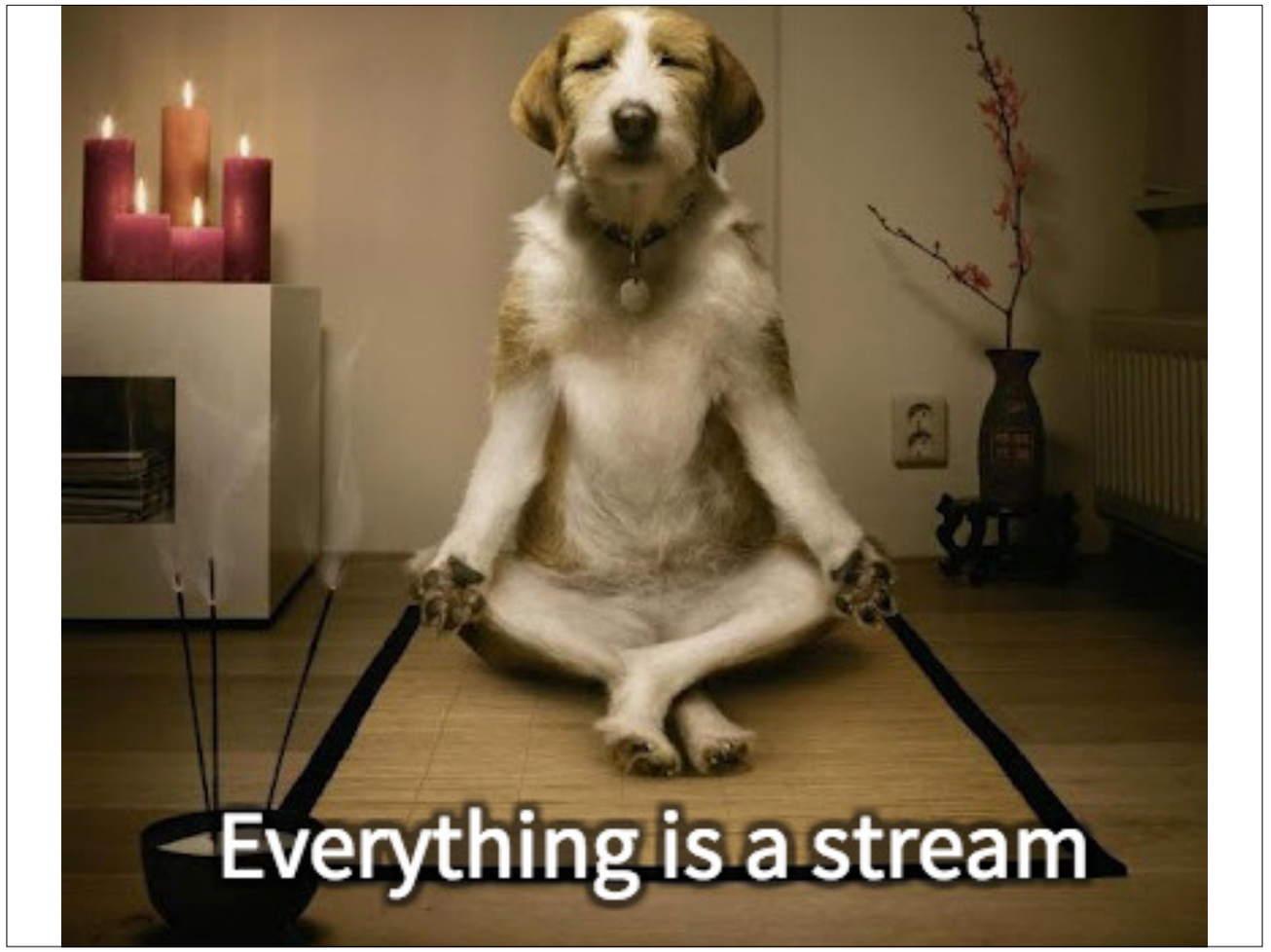
Twitter: @svenardocom

Når noen forsøker å fortelle deg hva
reaktiv programmering er....

Noen klassiske bilder som forsøker å forklare reaktive programmering kommer



I reaktiv programmering er all data en strøm

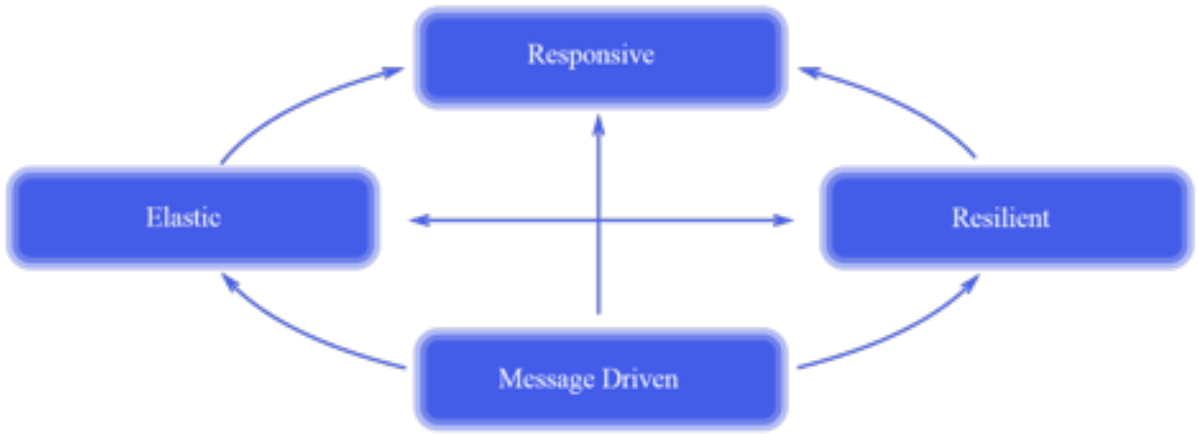


Joa, alt er en strøm

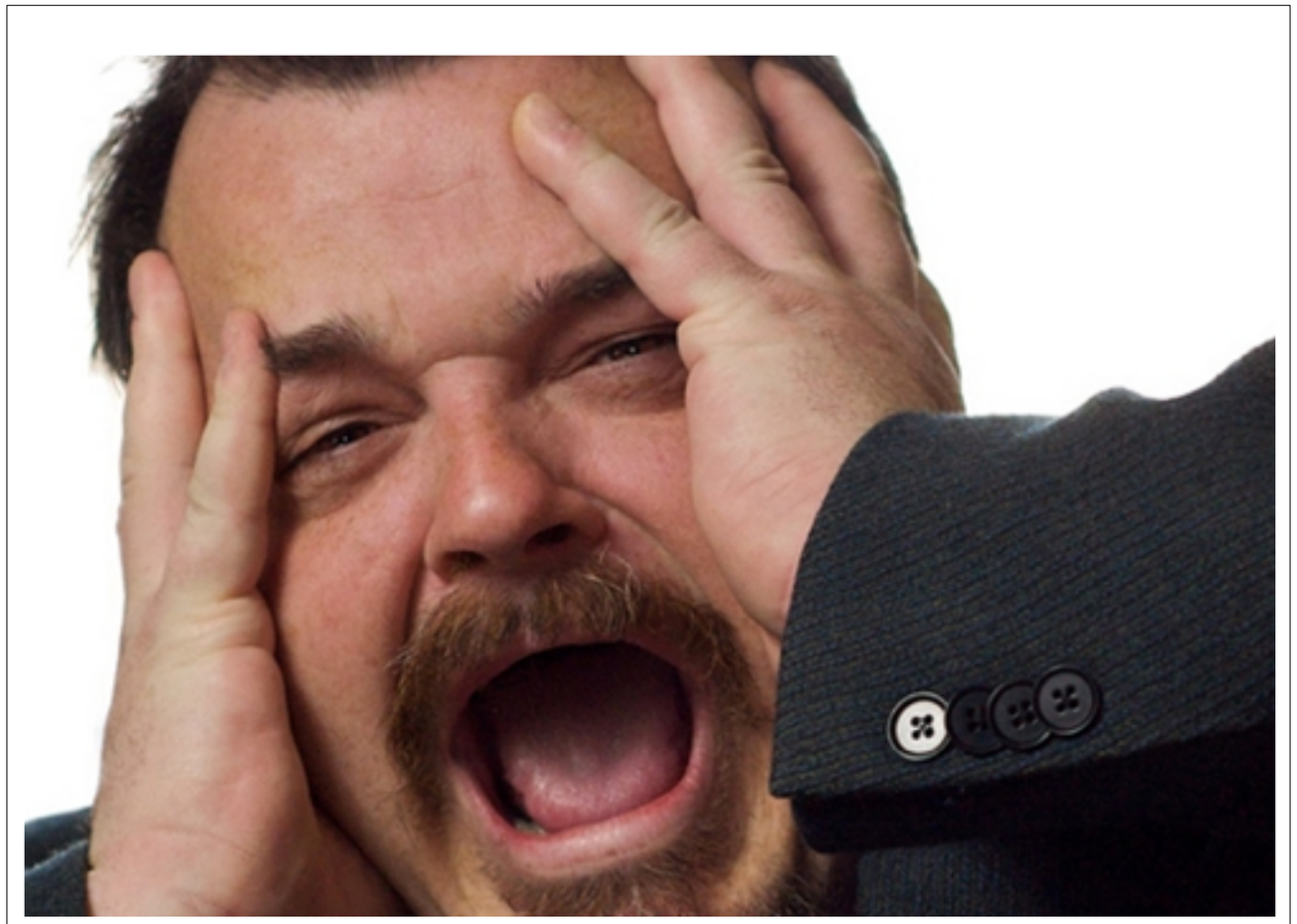
$$
\begin{aligned}
x_1(t_{j+1}) &= f_1(t_j, x_1, \ldots, x_n) \\
&\vdots \\
x_n(t_{j+1}) &= f_n(t_j, x_1, \ldots, x_n)
\end{aligned}
$$

En matematisk tilnærming

Reactive manifesto



Buzzwords...



Kan ingen bare forklare det på en forståelig måte?

Den enkle sannheten

Observert over tid

I reaktiv programmering observerer du data over tid.
Du observerer et datasett gjennom å abonnere på det.
Når som helst kan datasettet gi fra seg en hendelse. Det kan være et datapunkt, en feilmelding eller en fullført-melding.
Det er alt. I boken fra Gang of four om patterns er reaktiv programmering Observer pattern.

Er ikke dette det samme som hendelsesdrevet programmering?

Det er mange likhetstrekk. Når du kjenner til prinsippene som driver reaktiv programmering kan du med relativ enkelhet lage et reaktivt funksjons-sett med hendelsesstyrt programmering.

Forskjellen ligger i tankesettet. Med hendelsesdrevet programmering tenker du gjerne at du skal utføre en enkelt handling steg for steg basert på en enkelthendelse. Du tenker imperativt.
Med reaktiv programmering tenker du i stedet på hvordan du skal utføre oppgaven du har fore basert på en innkommende strøm av data. Du bryr deg ikke om hvor disse kommer fra. Du tenker deklarativt.

Reactive rammeverk

RxJS

Bacon.js

Kefir.js

osv.

RXJS finnes også for Java:

RxJava

Enkel imperativ loop

```
var res;
for(var i = 0; i<8; i++){
  if(i % 2 === 0)
    res = i;
};
console.log(`Last even value: ${i}`);
// -> Last even value: 8
```

Samme kode med reaktiv programmering

```
const range = Rx.Observable.range(1,8)
 .filter((val) => val % 2 === 0)
 .takeLast(1)

range.subscribe(
  (val) => { console.log(`Last even value: ${val}`) }
);
// -> Last even value: 8
```

Forskjellen er her at du frikobler hendelsen med abonnementet på dataene

Eksempel 2
Data fetching med Promises og imperativ loop

```
fetch("http://www.reddit.com/r/puppies.json?limit=5")
.then((response) => response.json())
.then((data) => {
  $('#results')
  .empty()
  var i=0;
  while(i < data.data.children.length) {
    $('#results')
    .append('<div>')
    .append($("<h3></h3>").text(data.data.children[i].data.title))
    .append($('<img>',{src:data.data.children[i].data.thumbnail}))
    .append('</div>')
    i++;
  }
});
```

Promises

Imperativ kode

Eksempel 2
Data fetching med streams

```
const puppiesStream = Rx.Observable
.fromPromise(fetch("http://www.reddit.com/r/puppies.json?limit=5")
.then((response) => response.json()));

puppiesStream.subscribe(
  function(posts) {
    posts.data.children.map((post)=>{
      $('#results')
      .append('<div>')
      .append($("<h3></h3>").text(post.data.title))
      .append($('<img>',{src:post.data.thumbnail}))
      .append('</div>')
    })
  }
);
```

Promise

Reaktiv

Funksjonell programmering

UI-kode (imperativ)

Qwerty the Maltipoo

My little girl turned eight weeks today!

It's Monday!!

Slutt-resultat

Takk!

Sven A Robbestad
Inmeta Consulting AS

Frontendere ønskes. Ta kontakt med meg om du er interessert på **sverobbe@inmeta.no**

Bok om ReactJS og utvikling av ReactJS apps:
https://www.packtpub.com/web-development/reactjs-blueprints