

第三届 FEDAY(前端开发日)

UCloud

100offer

upyun

Broadview

图灵教育

GenePlan

StuQ

CSDN
不止于代码

W3C
plus

掘金

前端早读课

中生代技术

segmentfault

码云
Gitee.com

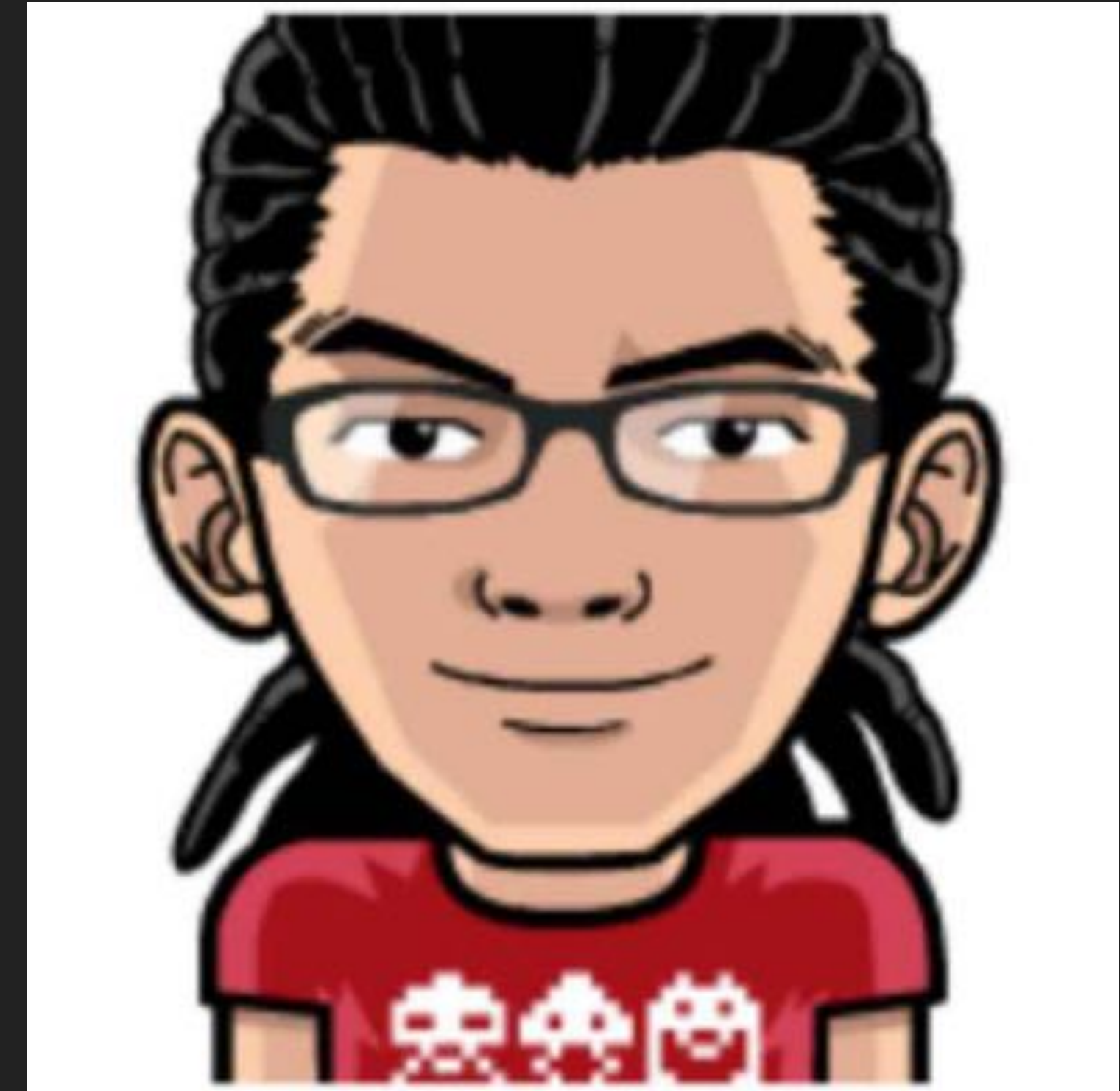


Jasin Yip

REACTIVE PROGRAMMING & STREAM WITH CYCLE.JS

SELF INTRODUCTION

- ▶ 叶俊星 (Jasin Yip)
- ▶ 美团点评 资深前端工程师
- ▶ 知乎前端开发、JavaScript 话题优秀回答者
- ▶ CyclejsCN (Cycle.js 中文社区) 发起者



OVERVIEW

- ▶ Introduction
 - ▶ What is *reactive programming*?
 - ▶ What is *stream*?
- ▶ Cycle.js - What if the user was a function?



WHAT IS

REACTIVE PROGRAMMING?

What is reactive programming?

$$a := b + c$$

What is reactive programming?

$$a := b + c$$

- ▶ Imperative Programming

- ▶ When ***b*** or ***c*** changed, no effect on ***a***.

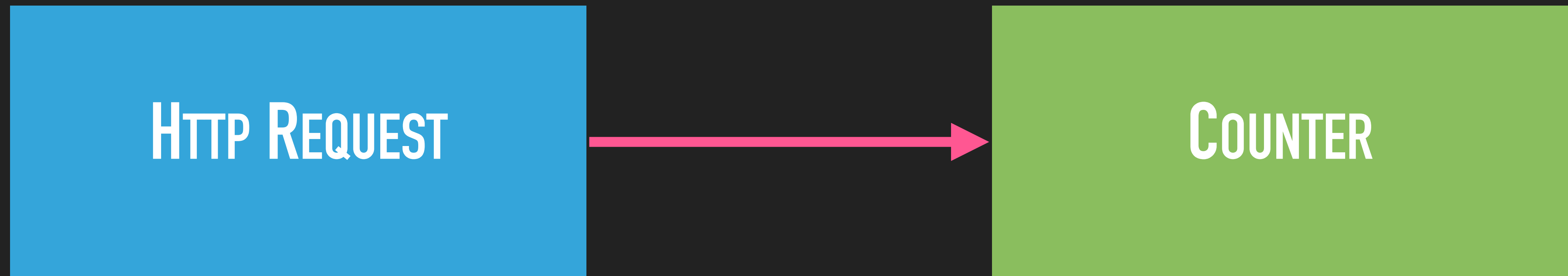
- ▶ Reactive Programming

- ▶ When ***b*** or ***c*** changed, ***a*** automatically updated.

WHY SHOULD WE USE REACTIVE PROGRAMMING?

Why Should we use reactive programming?

For example: count HTTP request times



Why Should we use reactive programming?

PROACTIVE

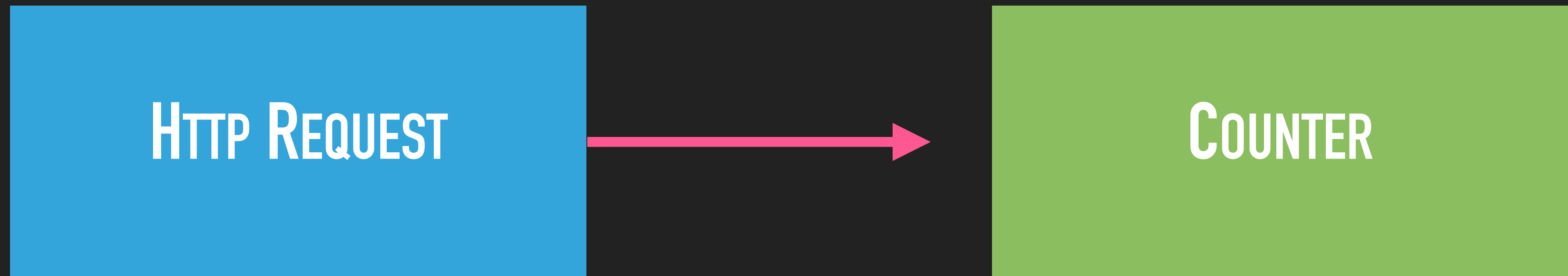
HTTP REQUEST

```
class HttpRequest {  
  // ...  
  send () {  
    // ...  
    counter.increment()  
  }  
}
```

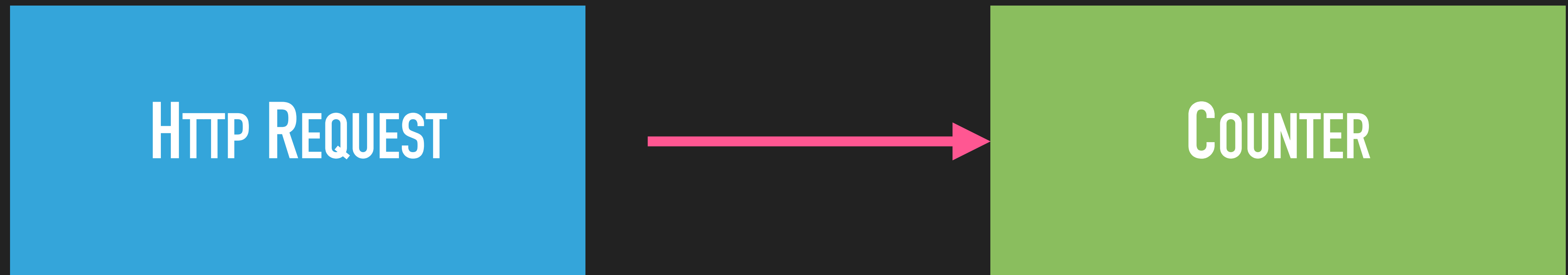
PASSIVE

COUNTER

Why Should we use reactive programming?



Why Should we use reactive programming?



Why Should we use reactive programming?

LISTENABLE

HTTP REQUEST

```
class HttpRequest {  
  // ...  
  send () {  
    // ...  
    EventBus.emit('requestDone')  
  }  
}
```

REACTIVE

COUNTER

```
class Counter {  
  constructor (eventBus) {  
    eventBus.on('requestDone')  
      .then(this.increment)  
  },  
  // ...  
}
```


Why Should we use reactive programming?

	Passive	Reactive
How does Counter work?	<i>Find usages</i>	Look inside

	Proactive	Listenable
Which modules are affected by HttpRequest?	Look inside	<i>Find usages</i>

Separation of Concerns

The Selling Point of Reactive Programming

WAYS TO IMPLEMENT REACTIVE PROGRAMMING

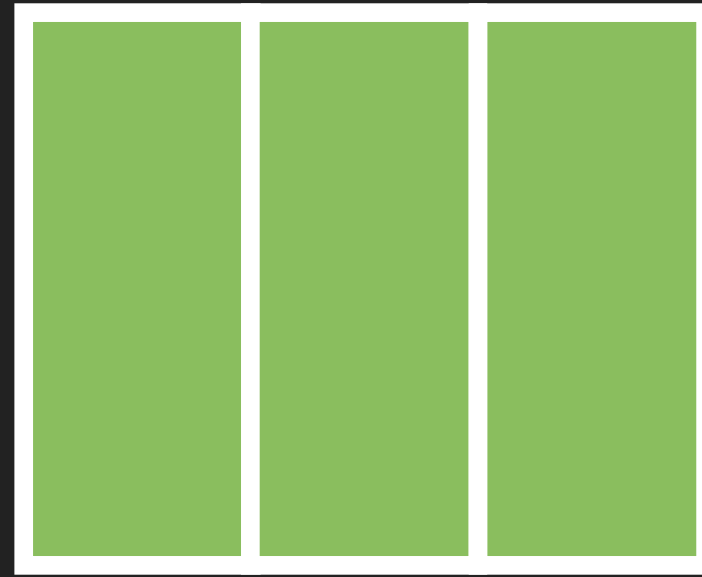
- ▶ EventBus
- ▶ Object.defineProperty
- ▶ ES2015 Proxy
- ▶ Streams (with some libraries like RxJS, xstream)
- ▶ ...

WHAT IS STREAM?

What is Stream?

A typical array:

Space



What is Stream?

A typical stream:



What is Stream?

- ▶ Array: sequence over space



- ▶ Stream: sequence over time



What is Stream?

[1, 2, 3]

```
map(e => e * 2)
```

[2, 4, 6]

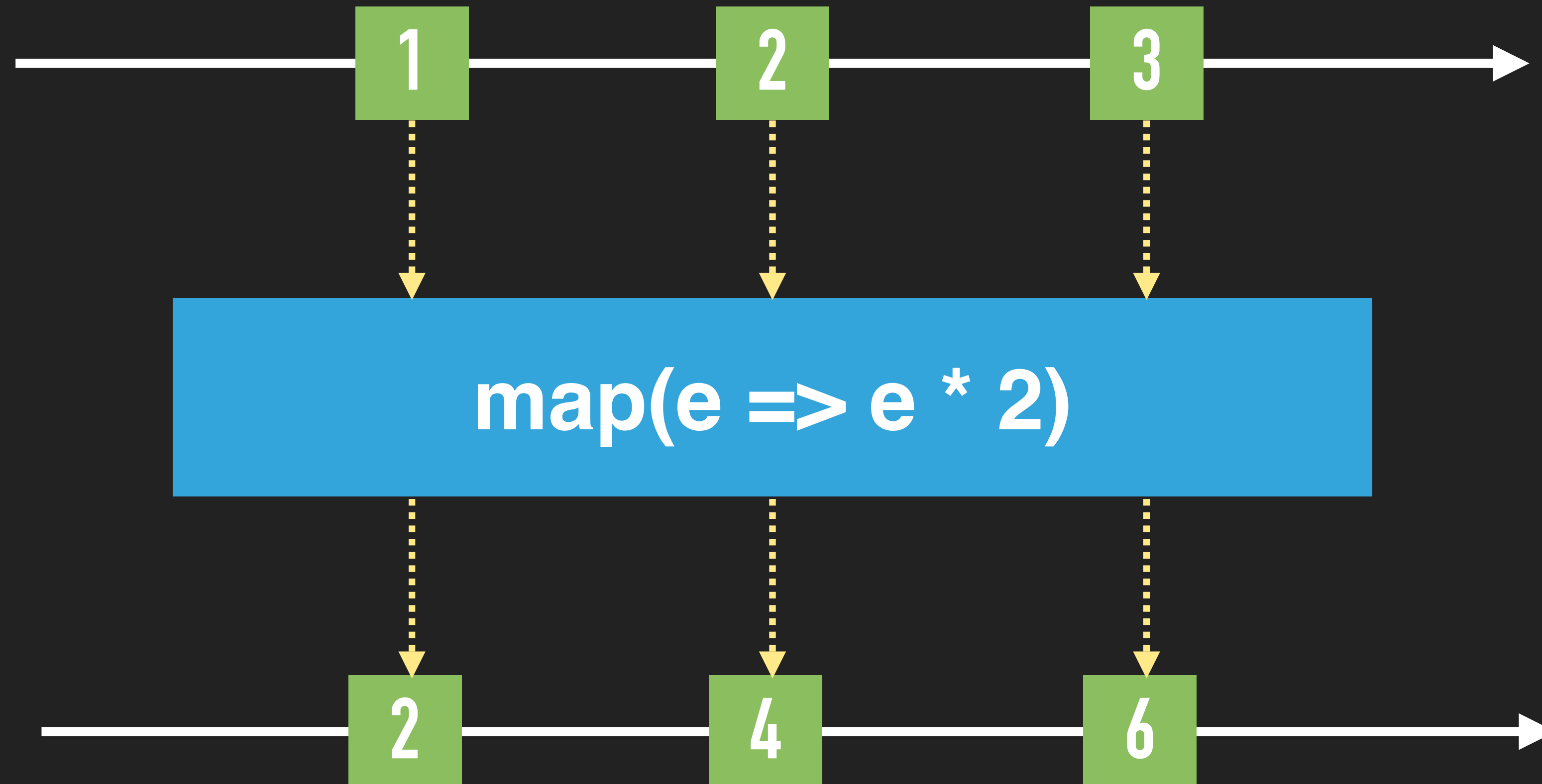
What is Stream?

[2, 6, 9]

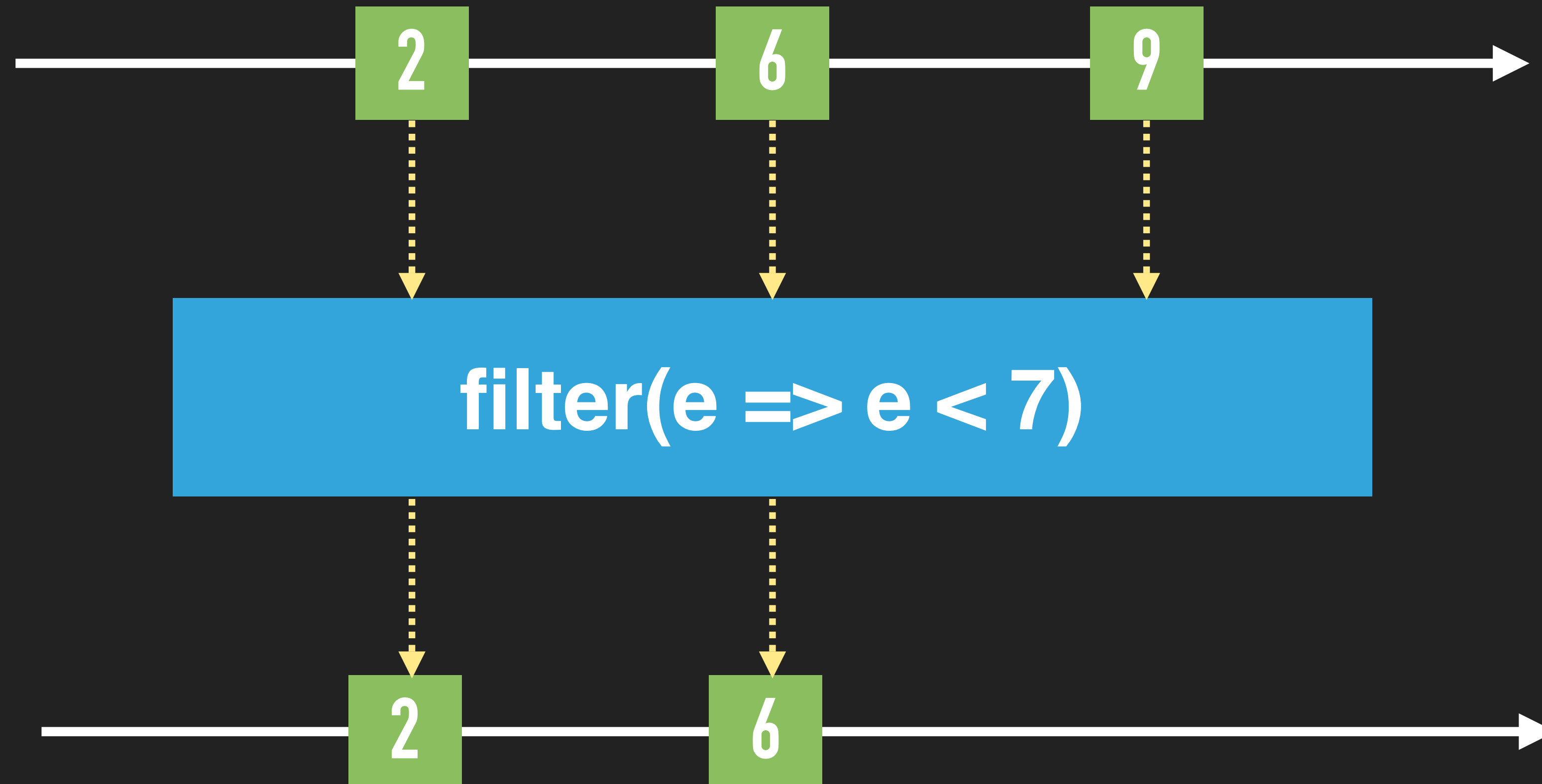
```
filter(e => e < 7)
```

[2, 6]

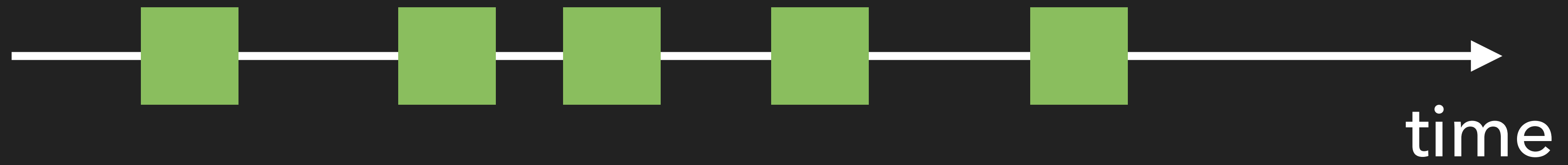
What is Stream?



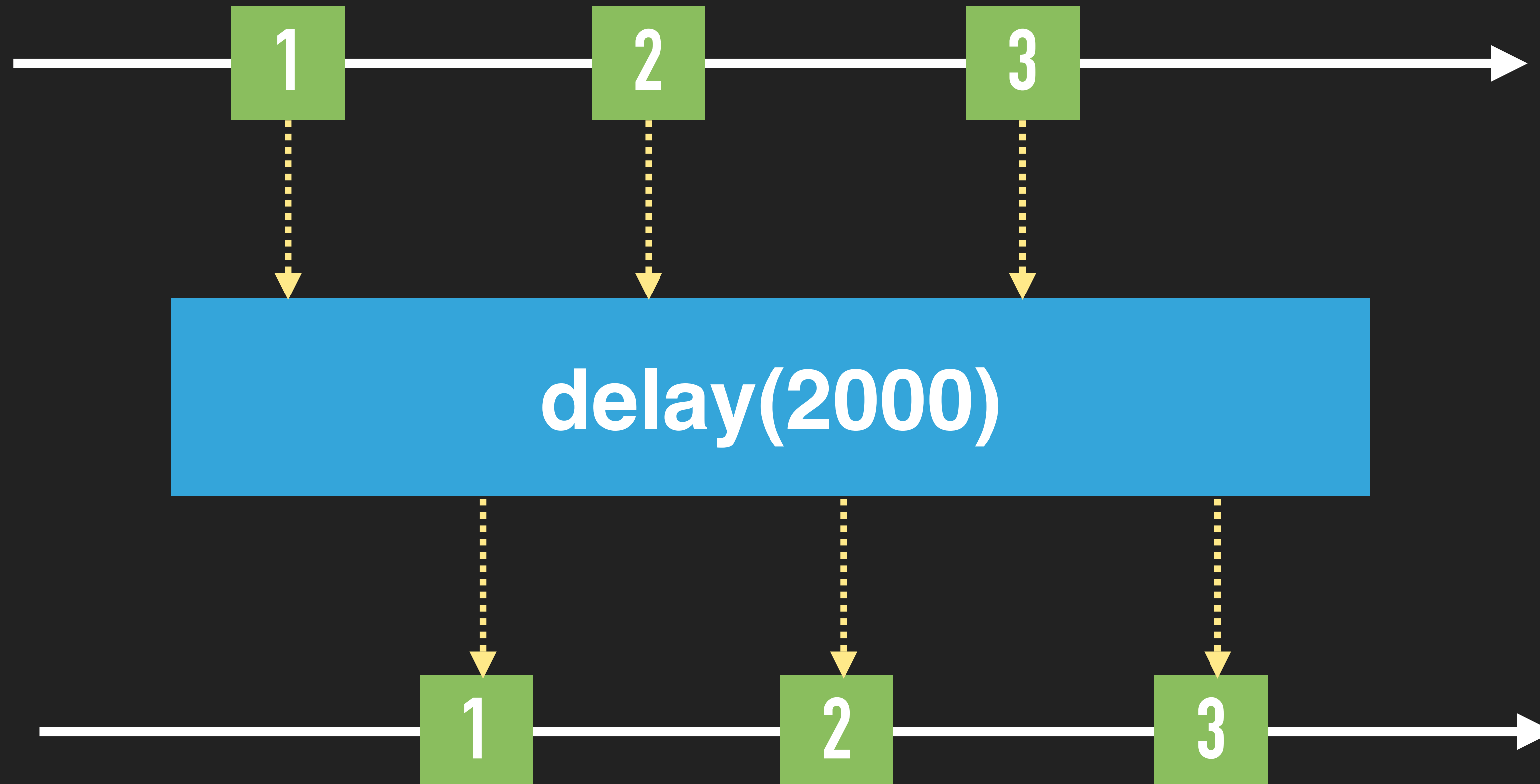
What is Stream?



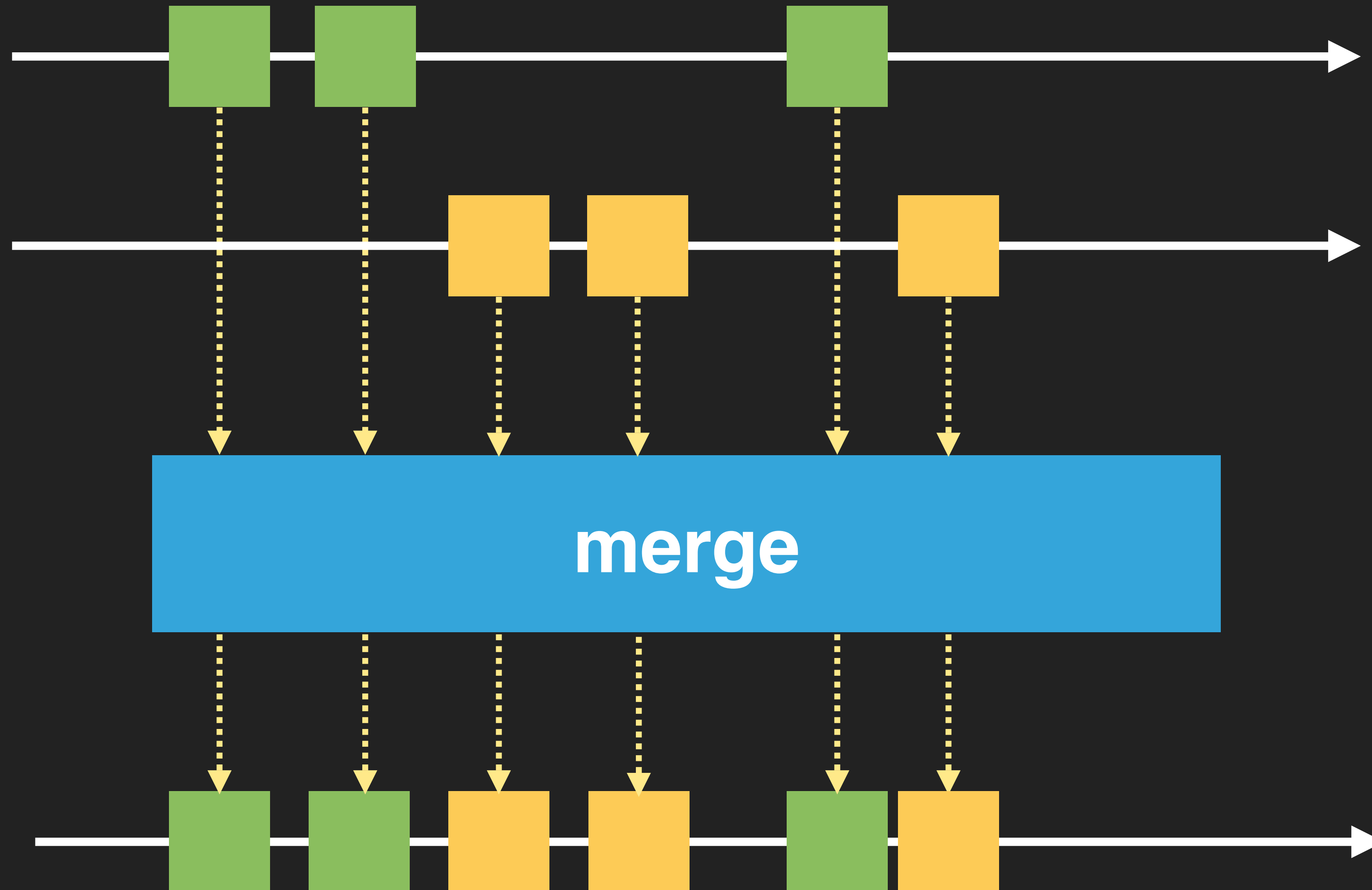
What is Stream?



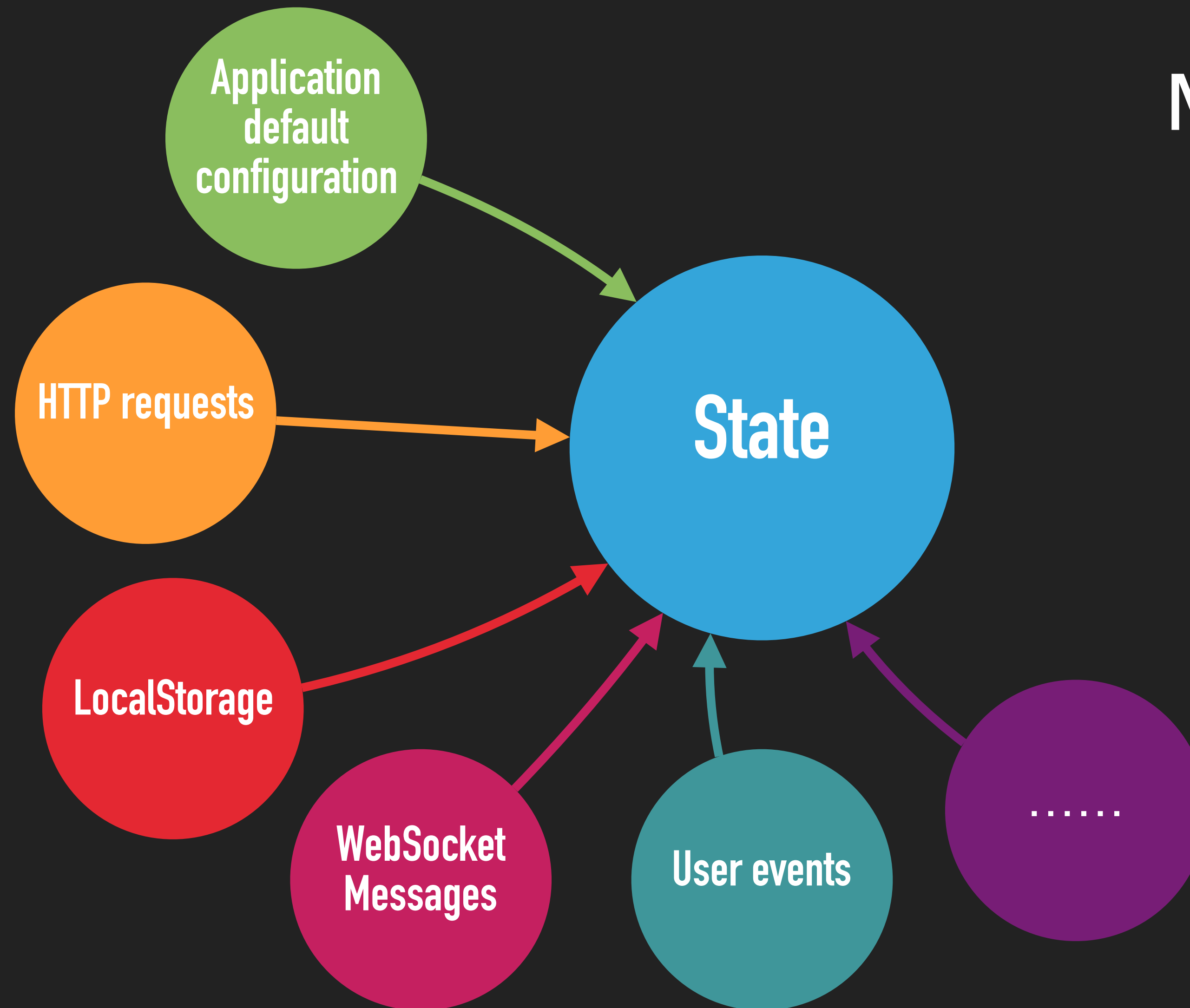
What is Stream?



What is Stream?



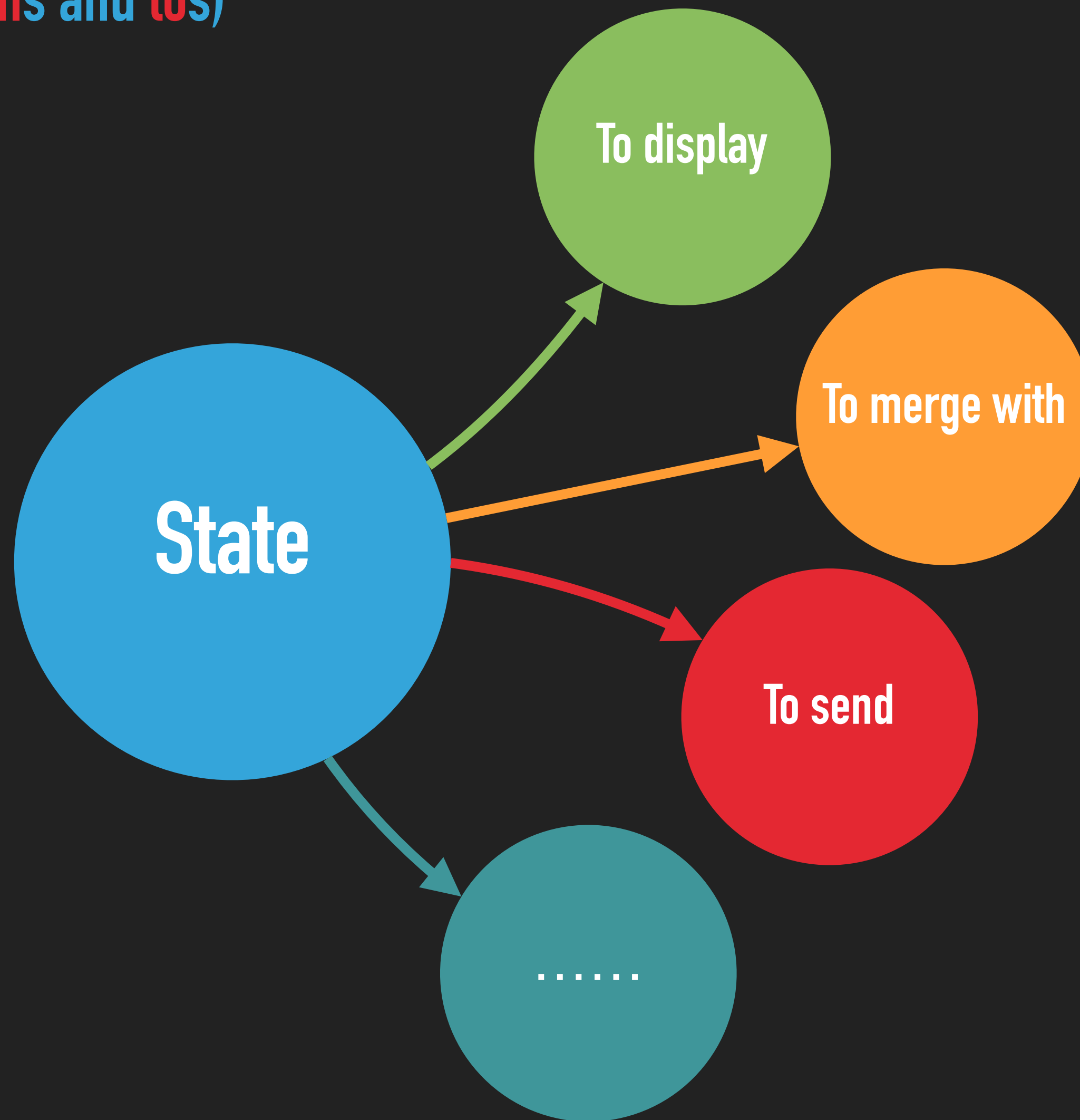
USAGE SCENARIOS (abstract **froms** and **tos**)



Multiple sources
mutate
the same state

USAGE SCENARIOS (abstract **from**s and **to**s)

Multiple places
use
the same state



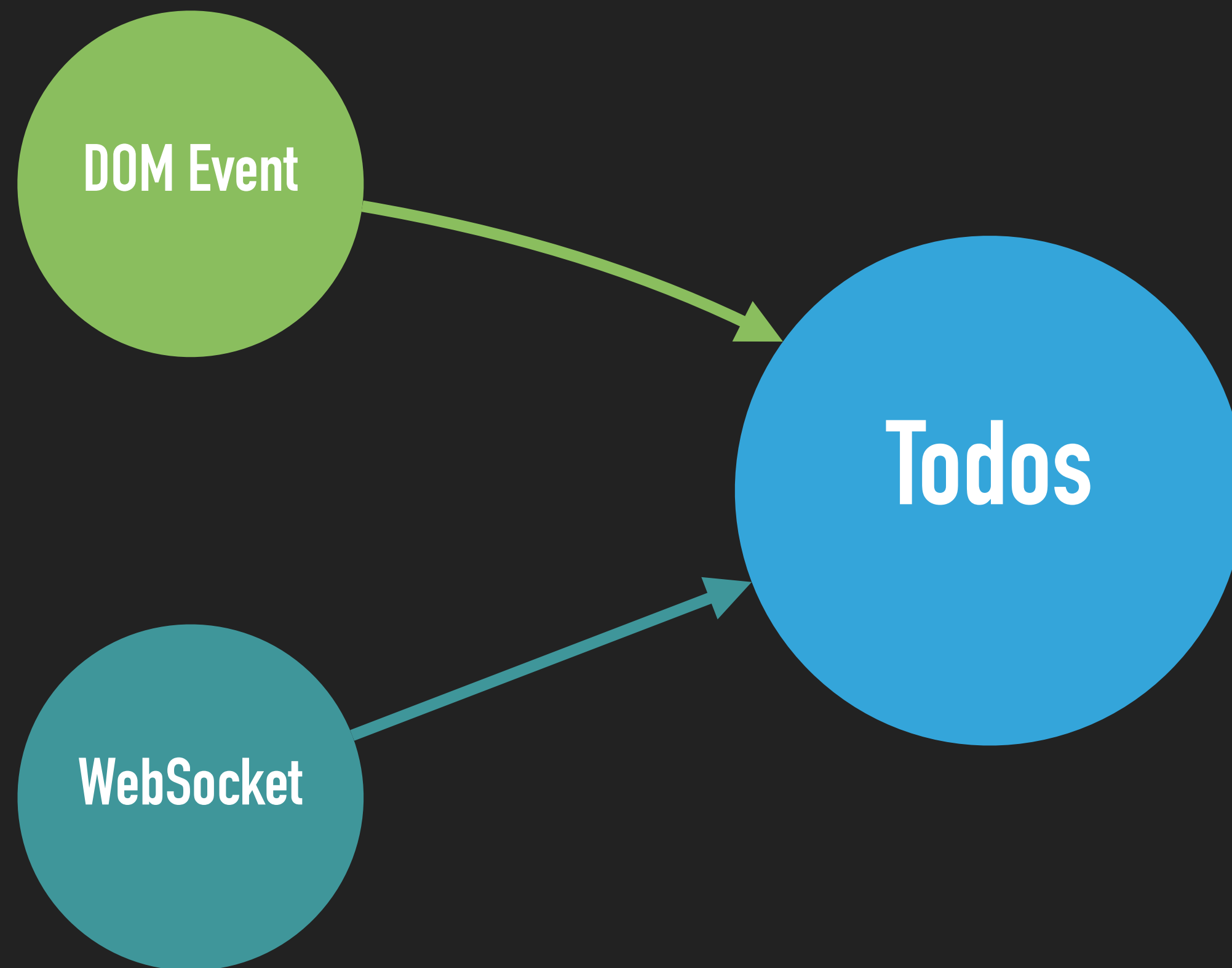
Normally using Redux:

```
const changeTodo = todo => {  
  dispatch({ type: 'updateTodo', payload: todo })  
}
```

```
const changefromDOMEvent = () => {  
  const todo = formState  
  changeTodo(todo)  
}
```

```
const changefromWebSocket = () => {  
  const todo = fromWS  
  changeTodo(todo)  
}
```


USAGE SCENARIOS (abstract **from**s and **to**s)



Using stream(with RxJS):

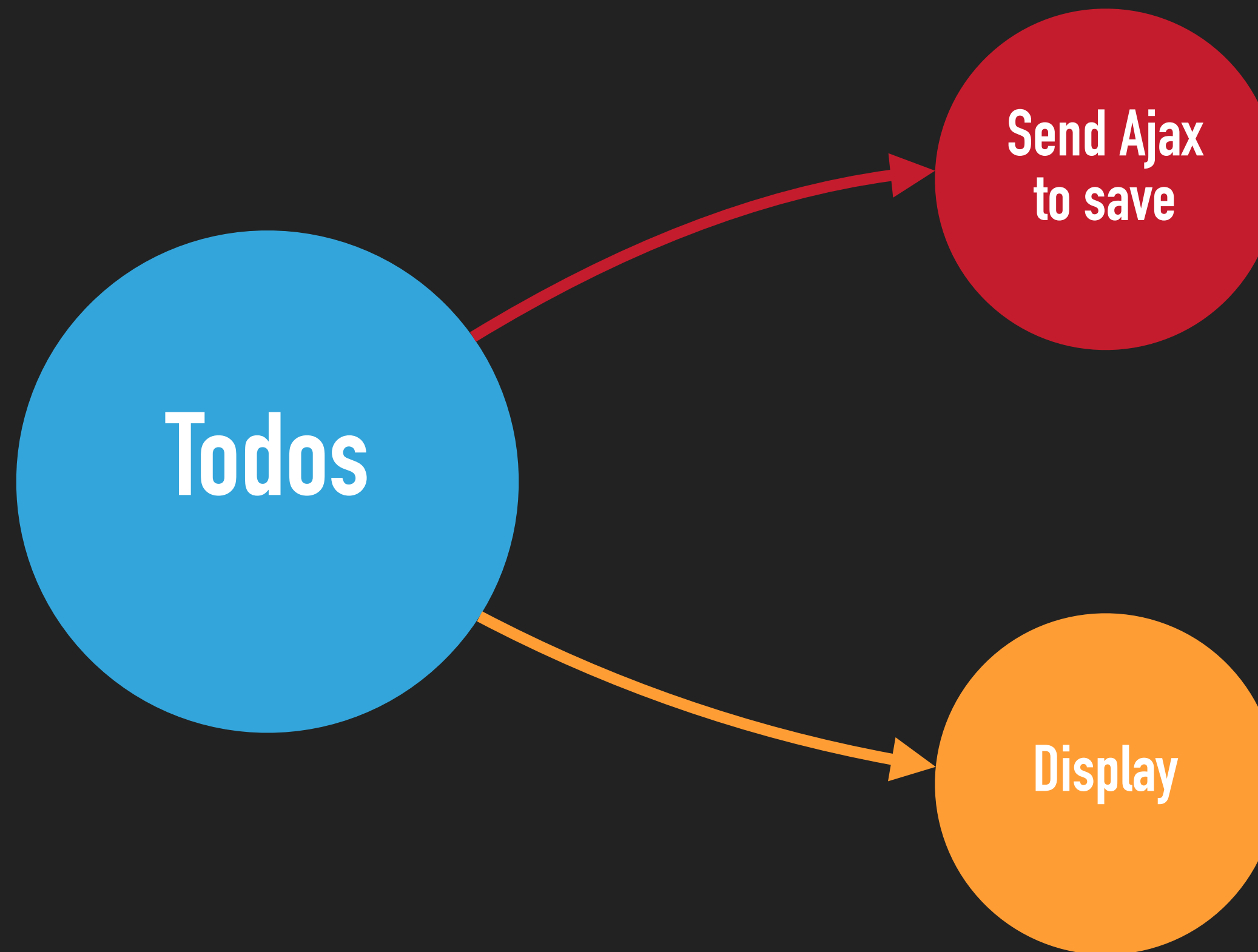
```
const changeFromDOMEvent$ = Rx.Observable  
  .fromEvent($('.btn', 'click'))  
  .map(evt => evt.data)
```

```
const changeFromWebSocket$ = Rx.Observable  
  .fromEvent(ws, 'message')  
  .map(evt => evt.data)
```

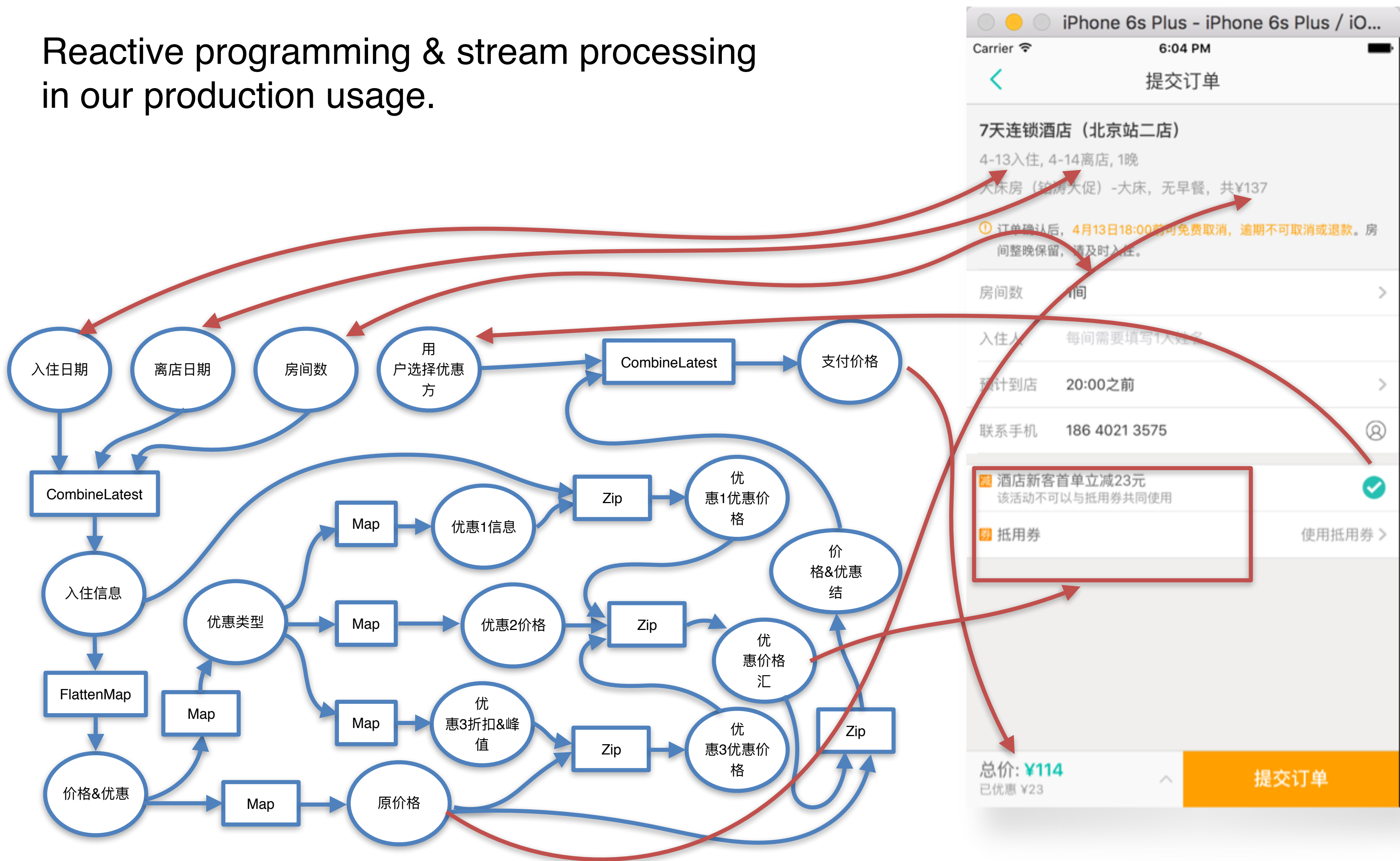
```
// Merge all data source
```

```
const changes$ = Rx.Observable.merge(  
  changeFromDOMEvent$,  
  changeFromWebSocket$  
)  
changes$.subscribe(todo => dispatch({ type: 'updateTodo', payload: todo })))
```


USAGE SCENARIOS (abstract **froms** and **tos**)



Reactive programming & stream processing
in our production usage.



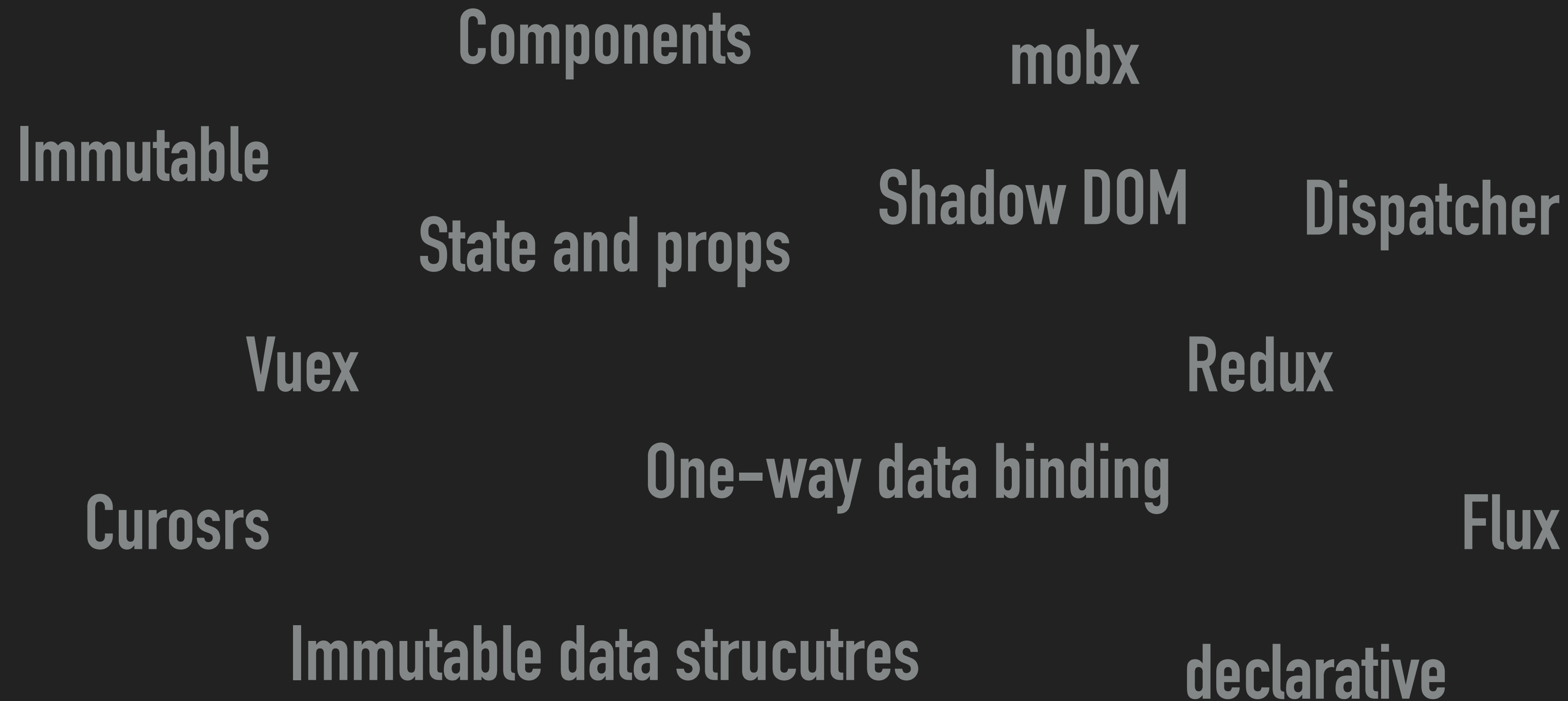
Just use stream!

The solution for the complicate situation below

**WHAT IF THE USER WAS
A FUNCTION?**

What if the user was a function?

JAVASCRIPT IN 2016/2017

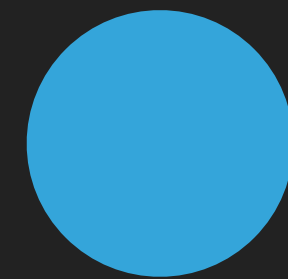
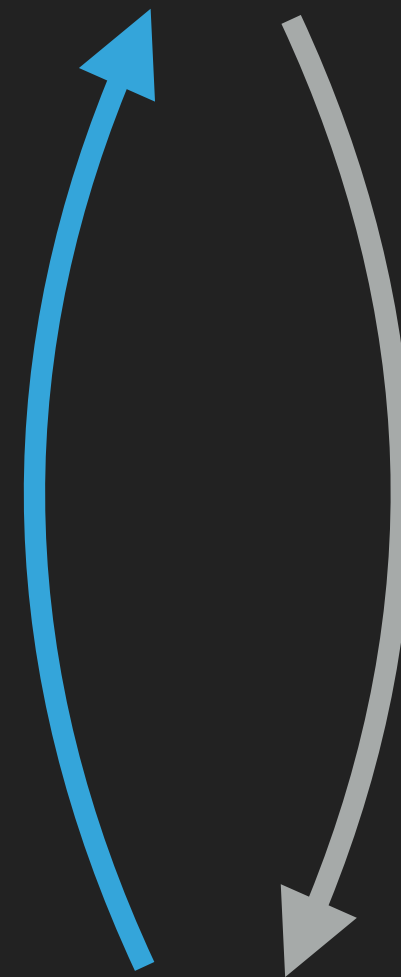
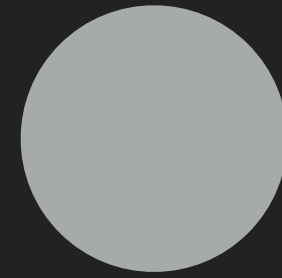


A person with long brown hair is wearing a black Oculus VR headset and a white t-shirt with a colorful graphic. They are sitting at a desk, looking upwards with an open mouth, appearing to be in a state of immersion or surprise. In the background, another person is also wearing a VR headset and using a controller. The setting appears to be a gaming or tech event with other people and bright lights in the distance.

NATURAL

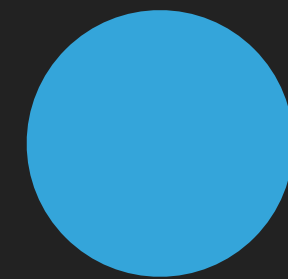
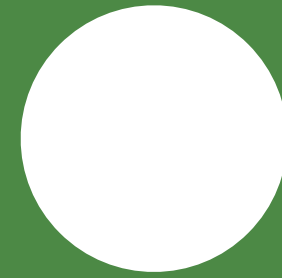
What if the user was a function?

Computer



Human

Computer



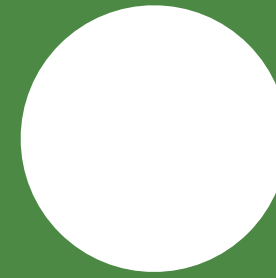
Human



What if the user was a function?

► Insight #1: UIs are cycles

Computer



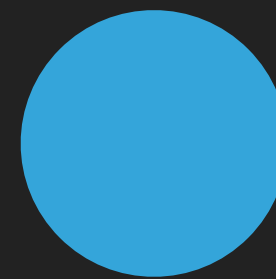
Input Devices



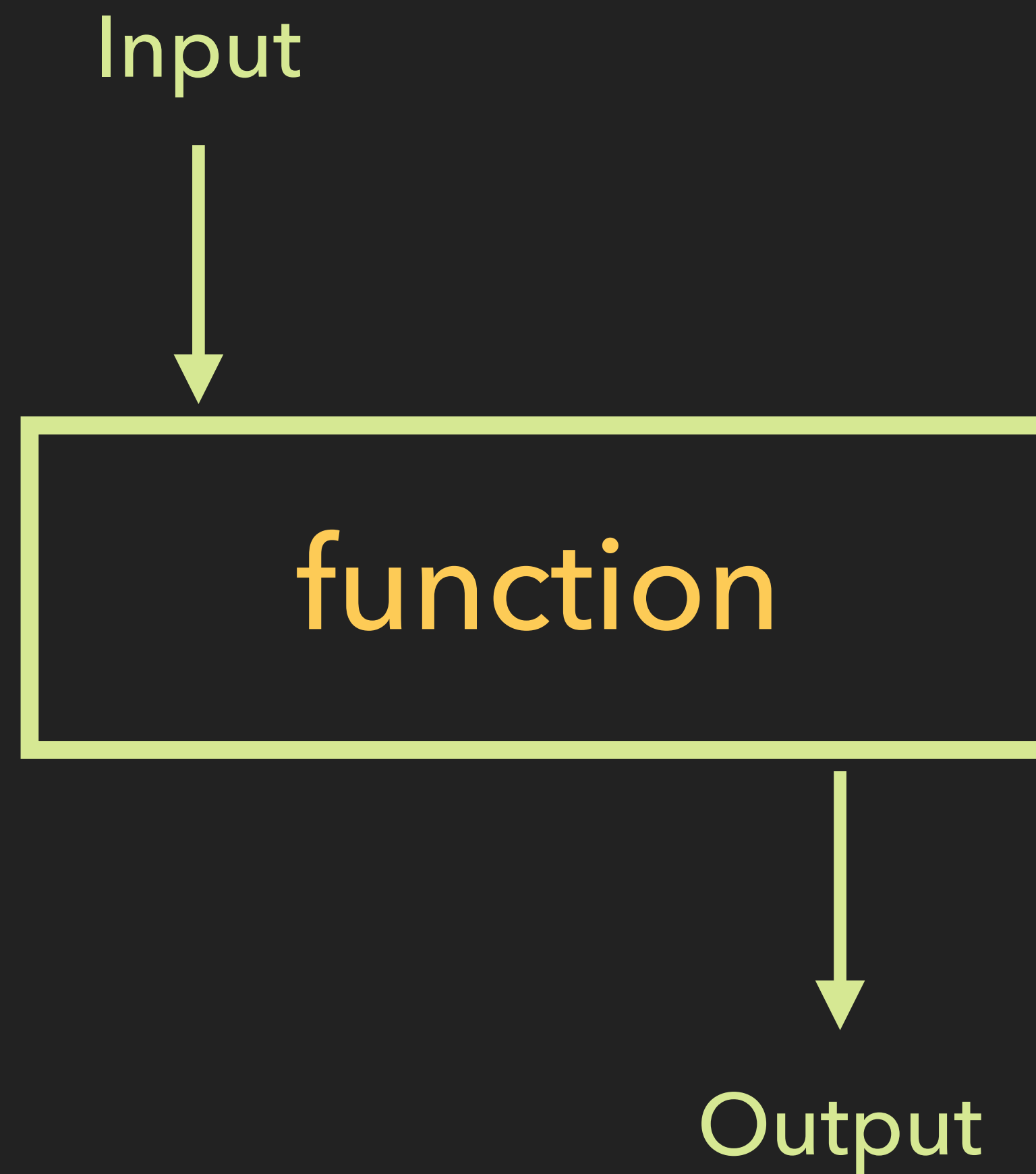
Output Devices



Human



What if the user was a function?



What if the user was a function?

- ▶ Insight #1: UIs are cycles
- ▶ Insight #2: UIs are functions



JavaSc|



javascript
javascript 教程
javascript array
javascript高级程序设计
javascript map
javascript date
javascript foreach
javascript 正则表达式
javascript 闭包
javascript promise

Google 搜索

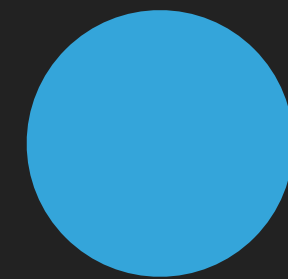
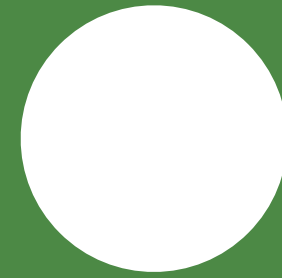
手气不错

[举报不当的联想查询](#)

What if the user was a function?

- ▶ Insight #1: UIs are cycles
- ▶ Insight #2: UIs are functions
- ▶ Insight #3: UIs are async

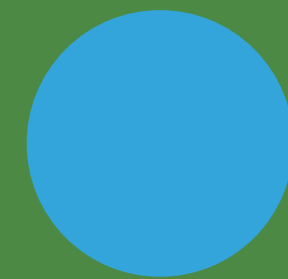
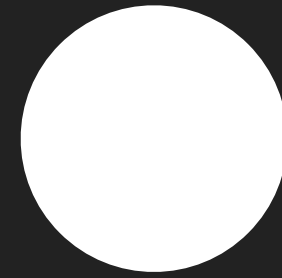
Computer



Human



Computer

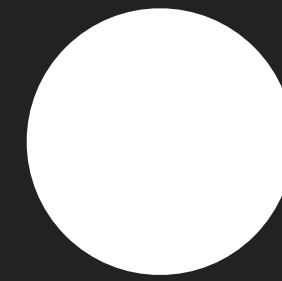


Human



- ▶ Insight #1: UIs are cycles
- ▶ Insight #2: UIs are functions
- ▶ Insight #3: UIs are async
- ▶ Insight #4: UIs are symmetric

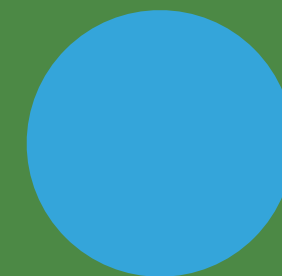
Computer



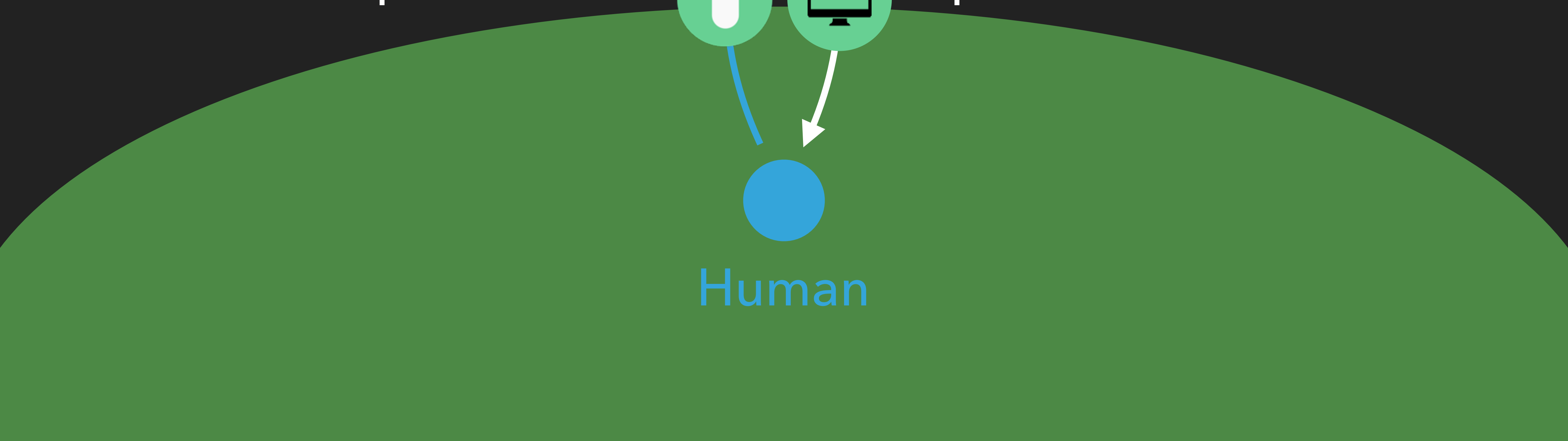
Output Devices



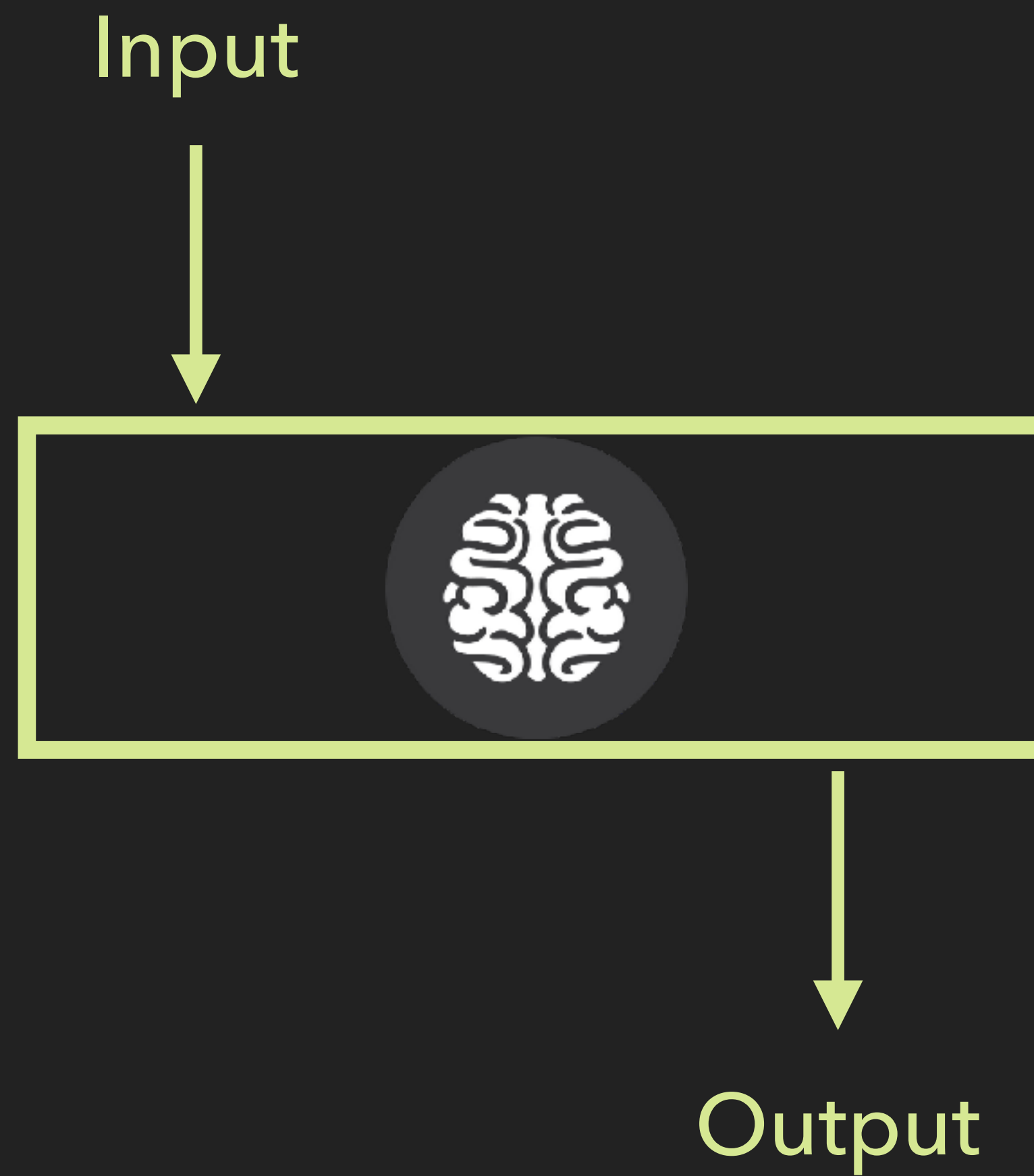
Input Devices



Human



What if the user was a function?

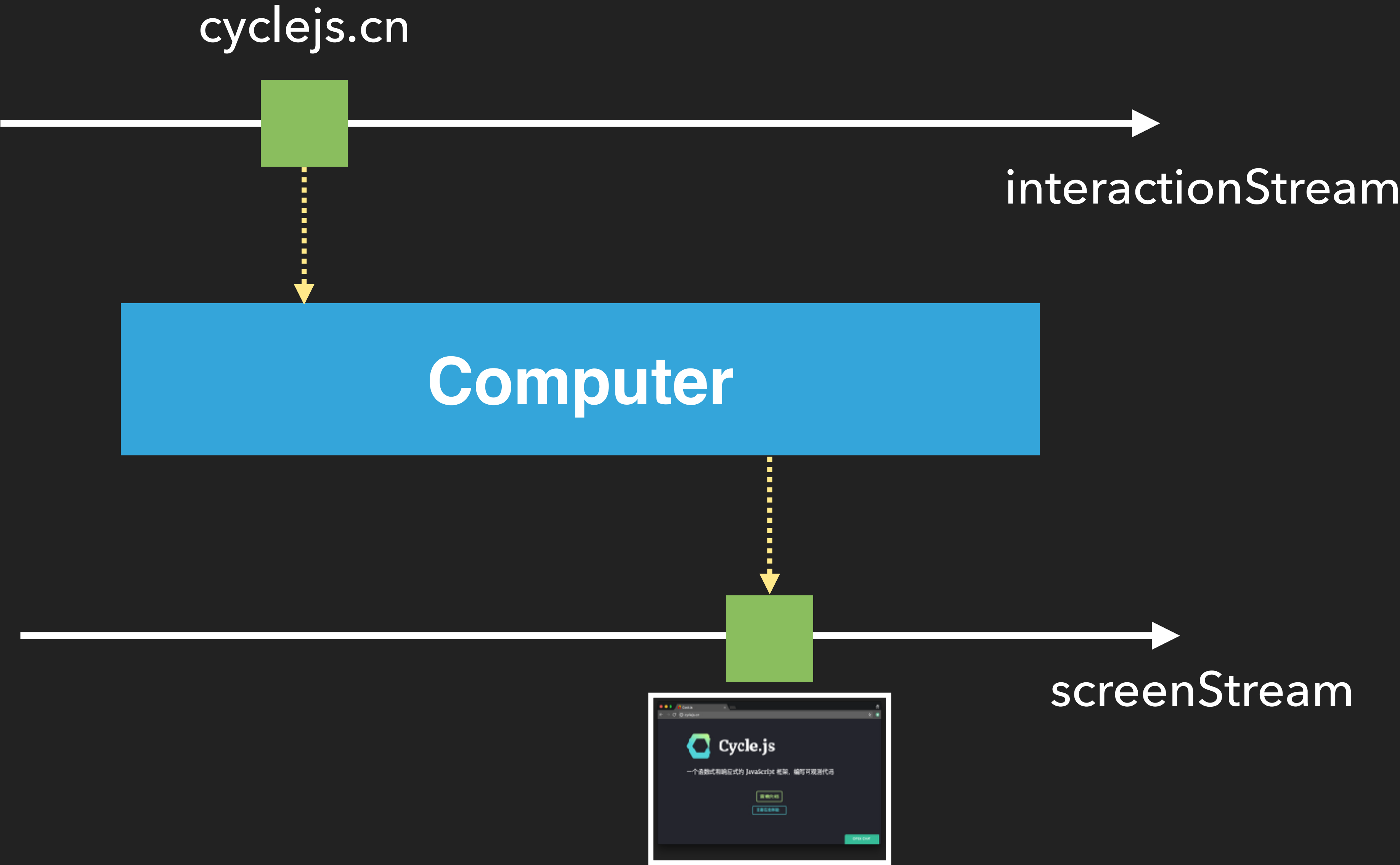


What if the user was a function?

- ▶ Insight #1: UIs are cycles
- ▶ Insight #2: UIs are functions
- ▶ Insight #3: UIs are async
- ▶ Insight #4: UIs are symmetric
- ▶ Insight #5: The user is a function

HOW TO CODE IT?

What if the user was a function?

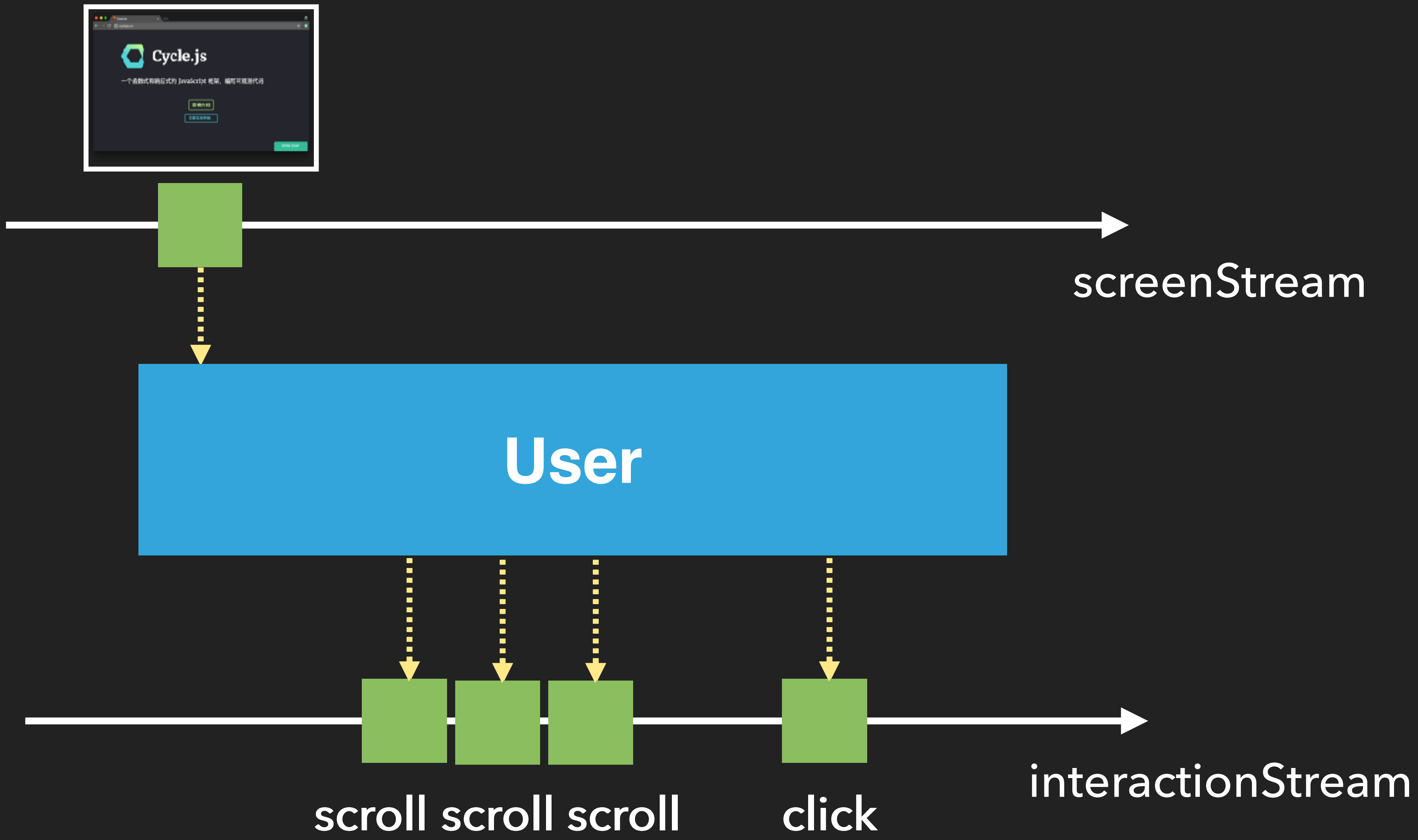


What if the user was a function?

```
function computer (url: EventStream<String>): EventStream<screen> {  
    // ...  
}
```

```
let screenStream = computer(interactionStream)  
screenStream.listen(function (ev) { ... })
```

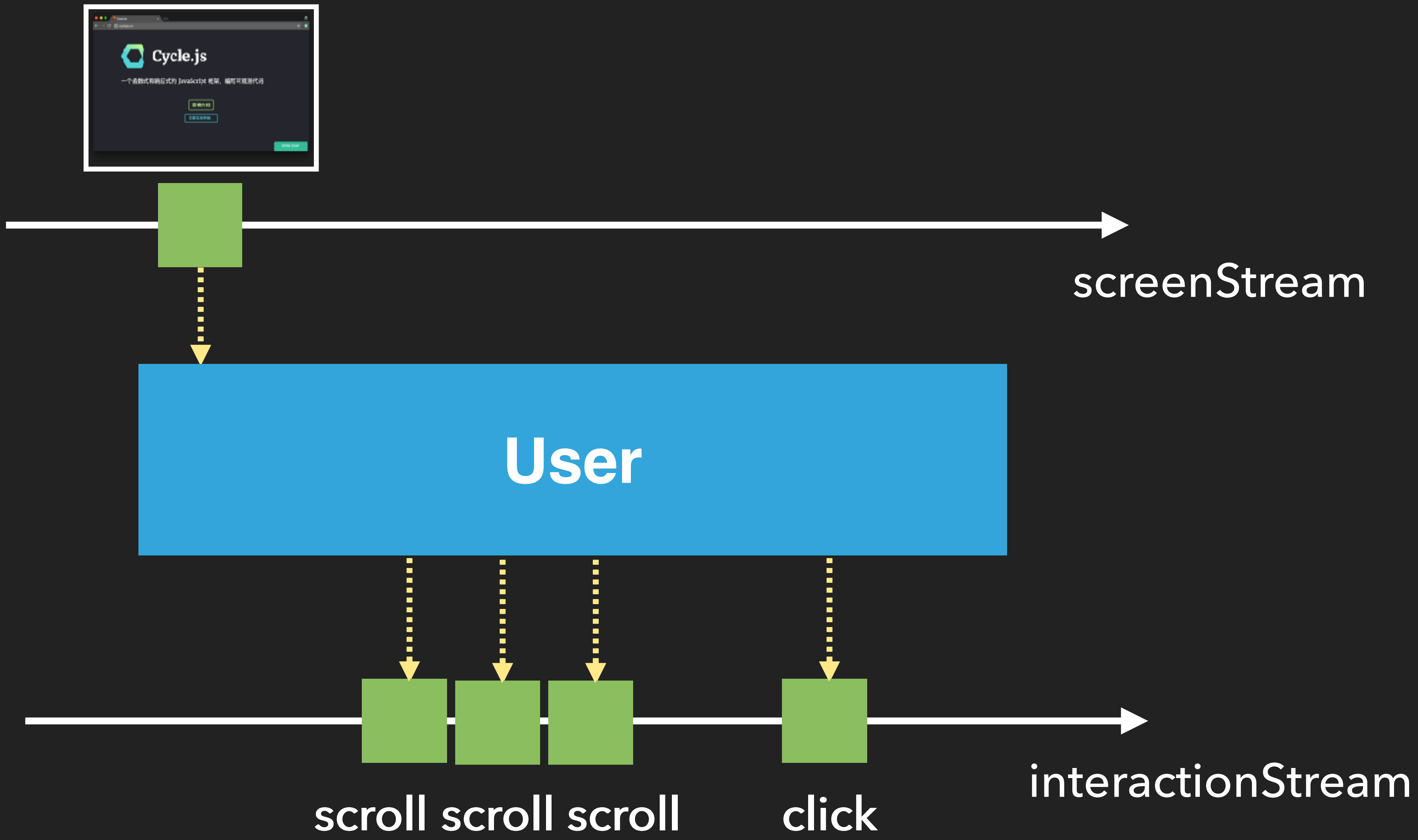

What if the user was a function?



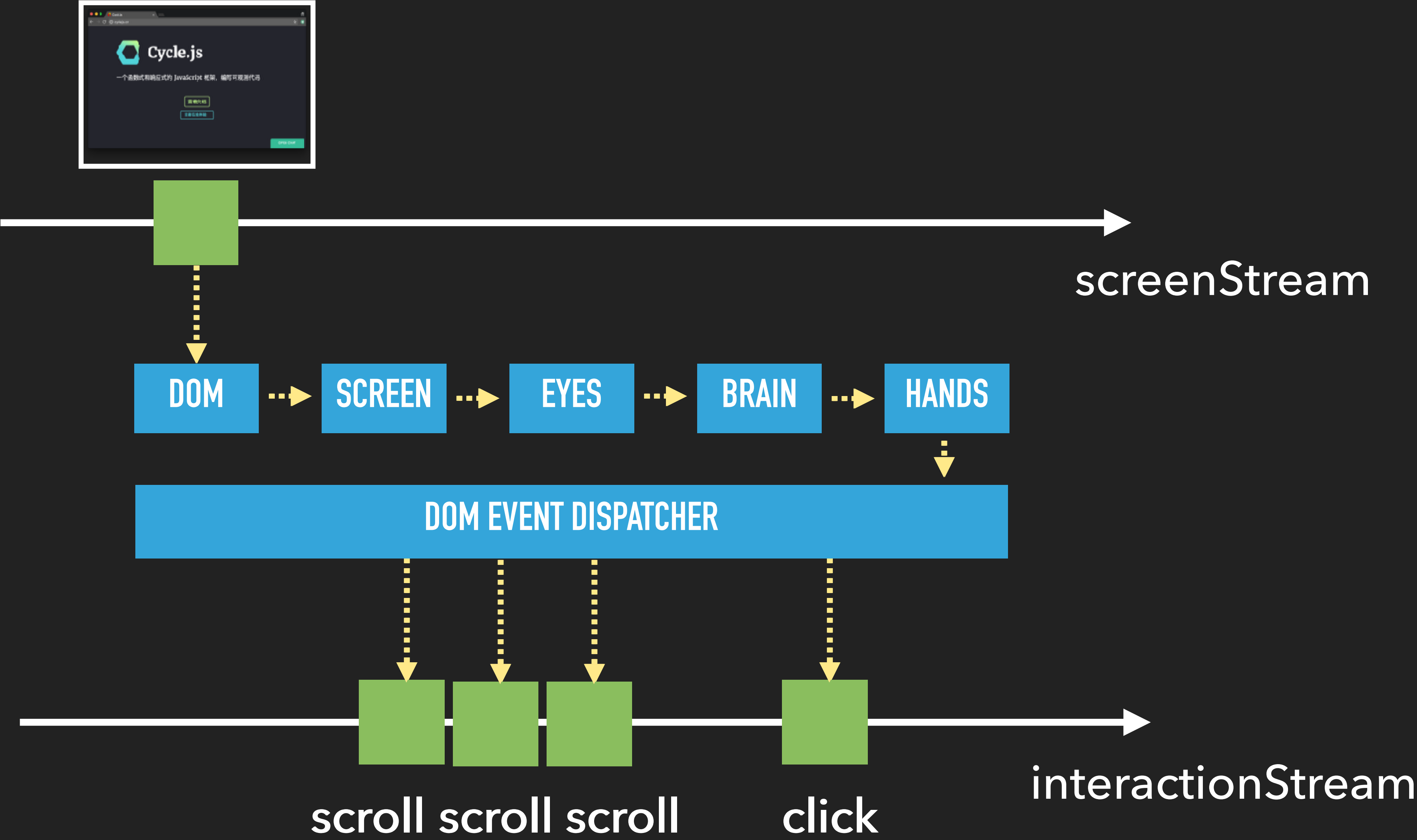
What if the user was a function?

```
function user (screenStream: EventStream): EventStream {  
  // Need your brain here...  
}
```

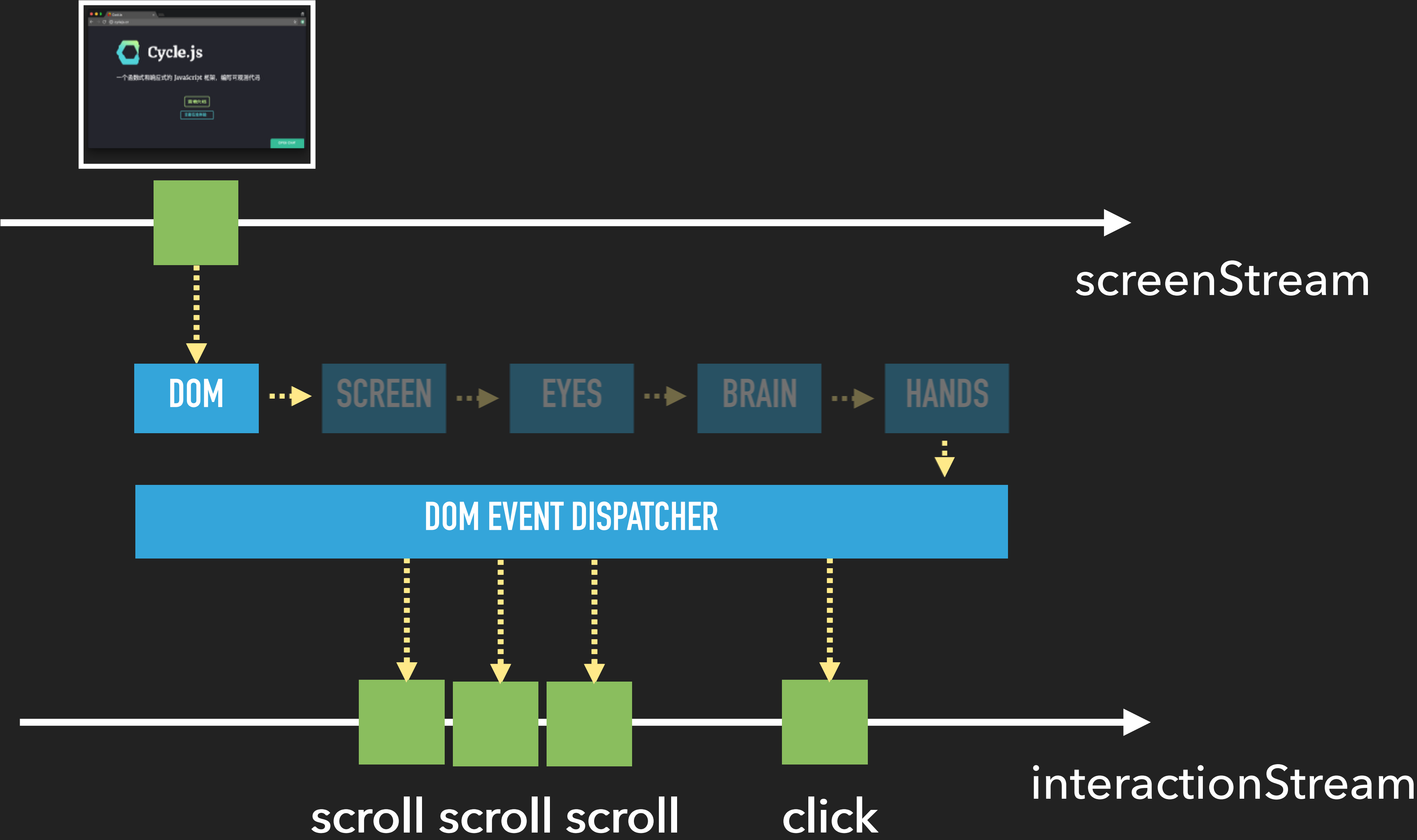

What if the user was a function?



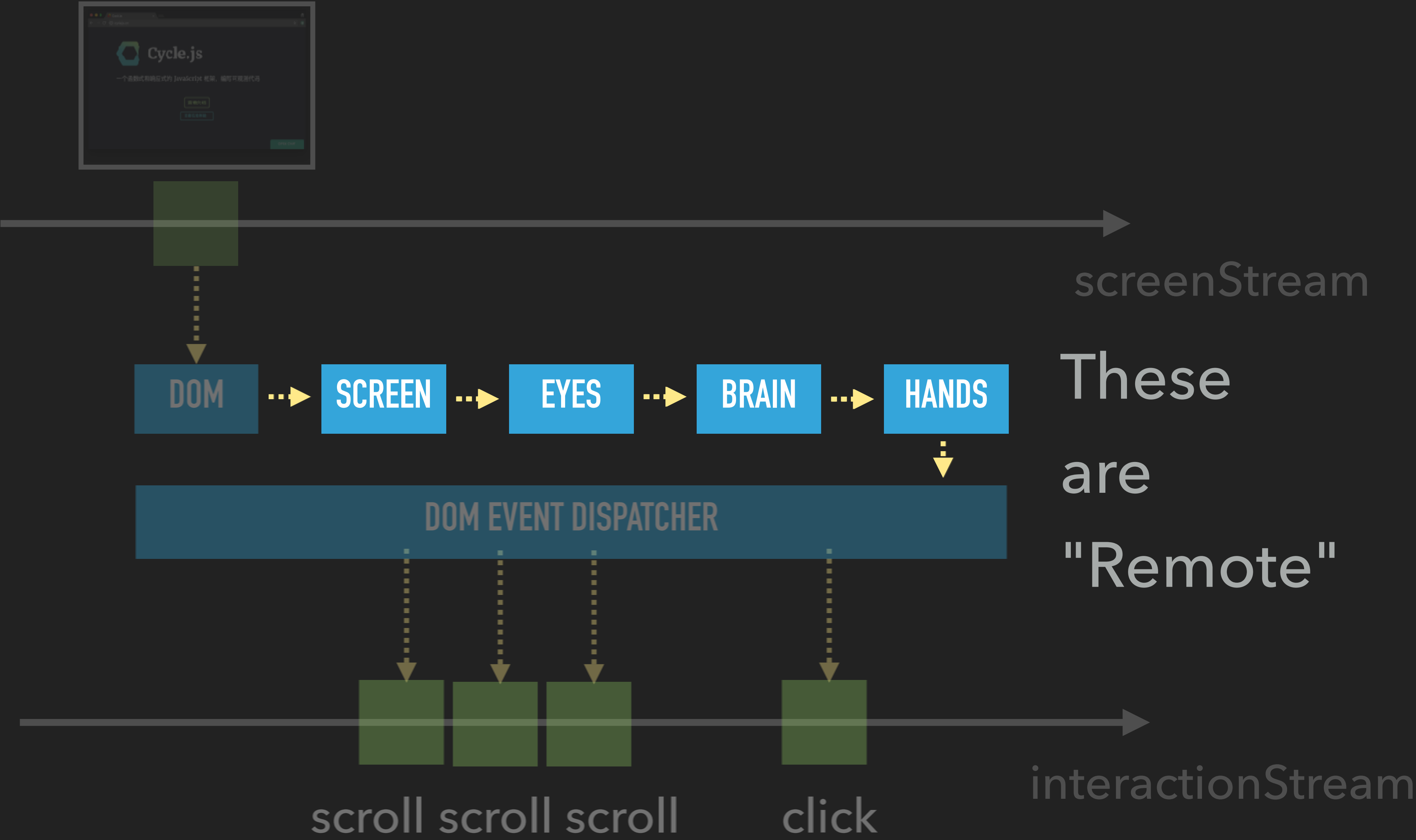
What if the user was a function?



What if the user was a function?



What if the user was a function?



What if the user was a function?

```
function user (screenStream: EventStream): EventStream {  
  // Need your brain here...  
}
```


What if the user was a function?

```
function user (screenStream: EventStream): EventStream {  
  screenStream.listen(function (screen) {  
    renderToDOM(screen)  
  })  
  let interactionEvents = new EventStream()  
  document.addEventListener('*', function (ev) {  
    interactionEvents.emit(ev)  
  })  
  return interactionEvents  
}
```


What if the user was a function?

```
function user (screenStream: EventStream): EventStream {  
  screenStream.listen(function (screen) {  
    renderToDOM(screen)  
  })  
  let interactionEvents = new EventStream()  
  document.addEventListener('*', function (ev) {  
    interactionEvents.emit(ev)  
  })  
  return interactionEvents  
}
```


What if the user was a function?

```
function user (screenStream: EventStream): EventStream {  
  screenStream.listen(function (screen) {  
    renderToDOM(screen)  
  })  
  let interactionEvents = new EventStream()  
  document.addEventListener('*', function (ev) {  
    interactionEvents.emit(ev)  
  })  
  return interactionEvents  
}
```


What if the user was a function?

```
let interactionStream = user(screenStream.listen)  
interactionStream.listen(function (ev) { ... })
```


What if the user was a function?

```
let screenStream = computer(interactionStream)  
let interactionStream = user(screenStream)
```


What if the user was a function?

let screenStream = computer(interactionStream)

let interactionStream = user(screenStream)



let a = f(b)

let b = g(a)

What if the user was a function?

let screenStream = computer(interactionStream)

let interactionStream = user(screenStream)



let a = f(b)

let b = g(a)



let b = g(f(b))



What if the user was a function?

```
let interactionStream = makeEmptyEventStream()
```

```
let screenStream = computer(interactionStream)
```

```
let interactionStream2 = user(screenStream)
```

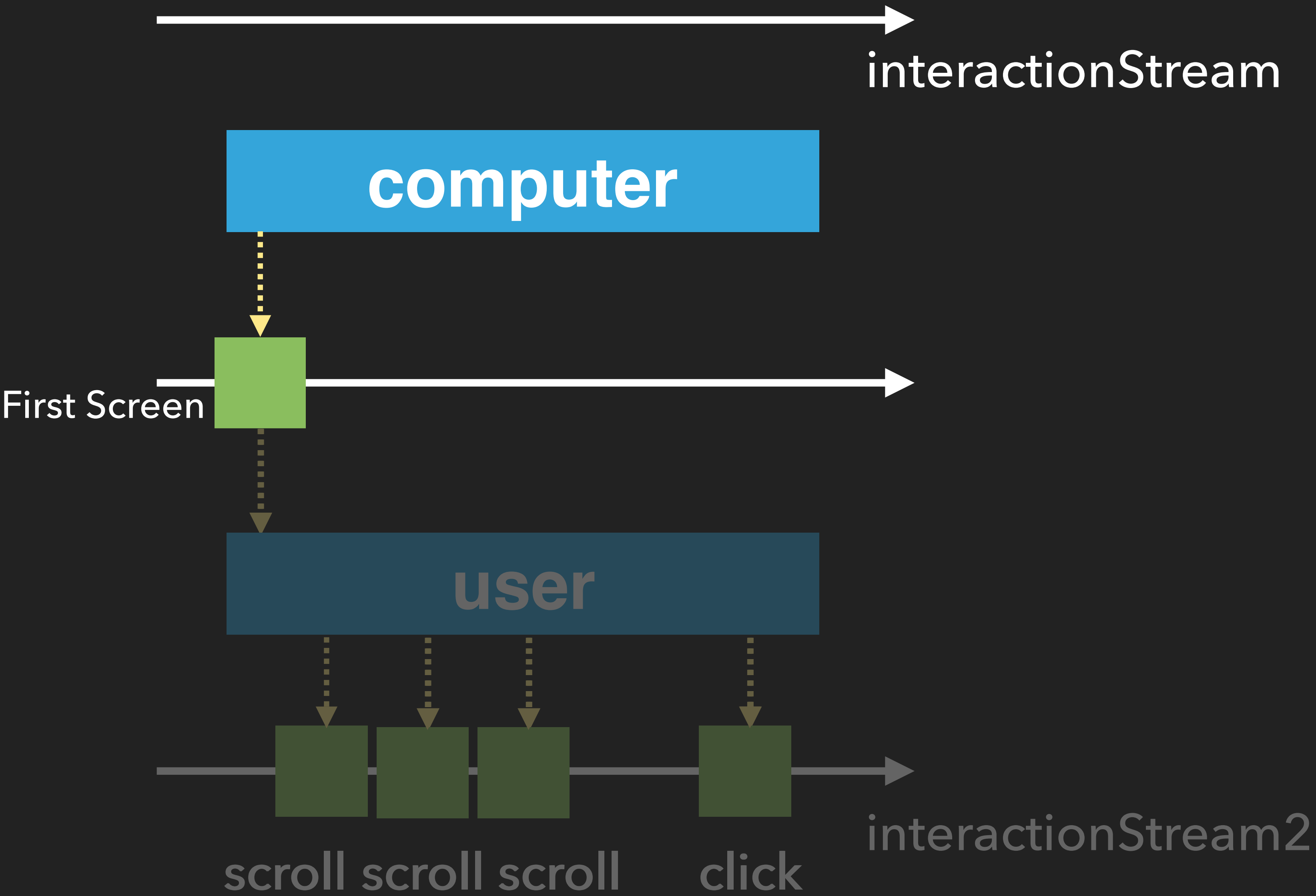

What if the user was a function?

```
let interactionStream = makeEmptyEventStream()
```

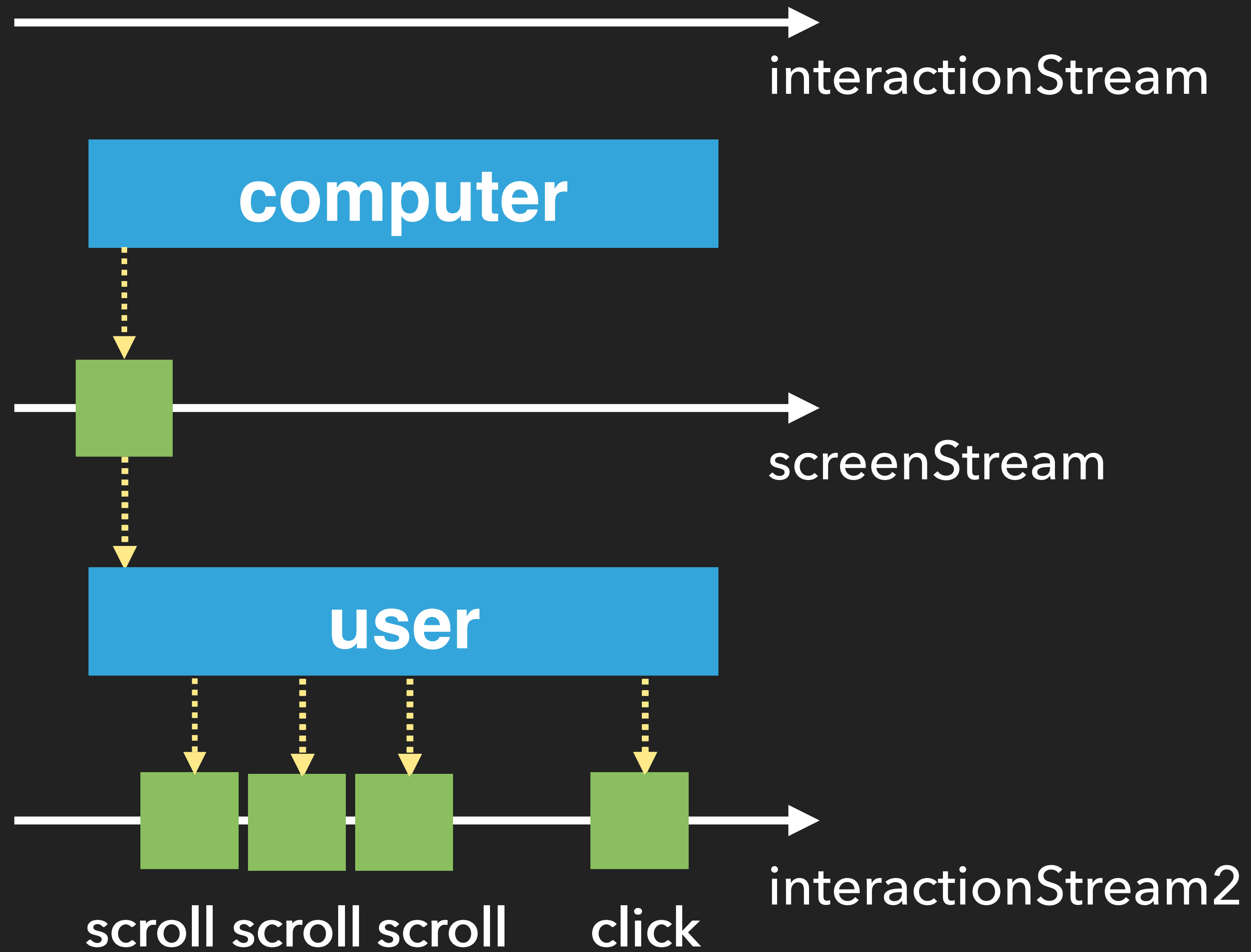
```
let screenStream = computer(interactionStream)
```

```
let interactionStream2 = user(screenStream)
```

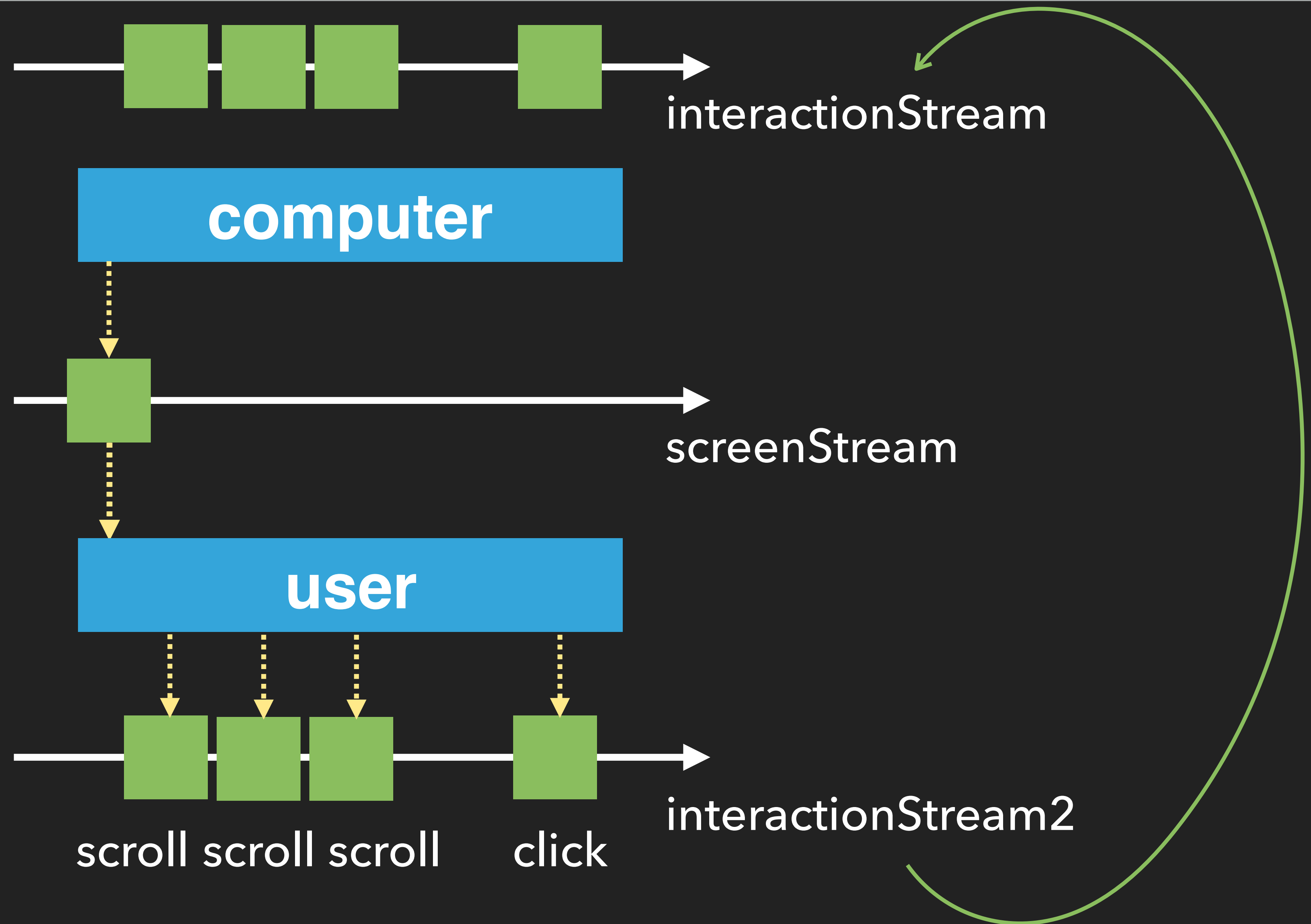

What if the user was a function?



What if the user was a function?



What if the user was a function?



What if the user was a function?

```
let interactionStream = makeEmptyEventStream()
```

```
let screenStream = computer(interactionStream)
```

```
let interactionStream2 = user(screenStream)
```

```
interactionStream2.listen(function (ev) {  
  interactionStream.emit(ev)  
})
```


What if the user was a function?

```
let interactionStream = makeEmptyEventStream()
```

```
let screenStream = computer(interactionStream)
```

```
let interactionStream2 = user(screenStream)
```

```
interactionStream2.listen(function (ev) {  
  interactionStream.emit(ev)  
})
```


What if the user was a function?

```
let interactionStream = makeEmptyEventStream()
```

```
let screenStream = computer(interactionStream)
```

```
let interactionStream2 = user(screenStream)
```

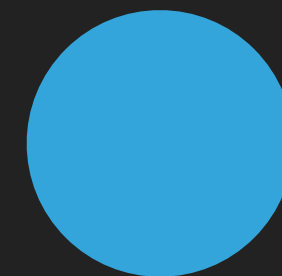
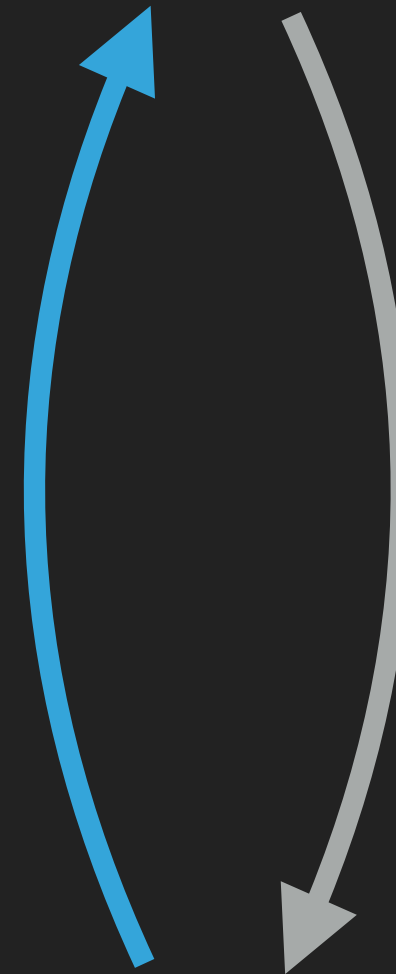
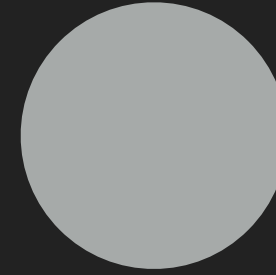
```
interactionStream2.listen(function (ev) {  
  interactionStream.emit(ev)  
})
```




CYCLE.JS

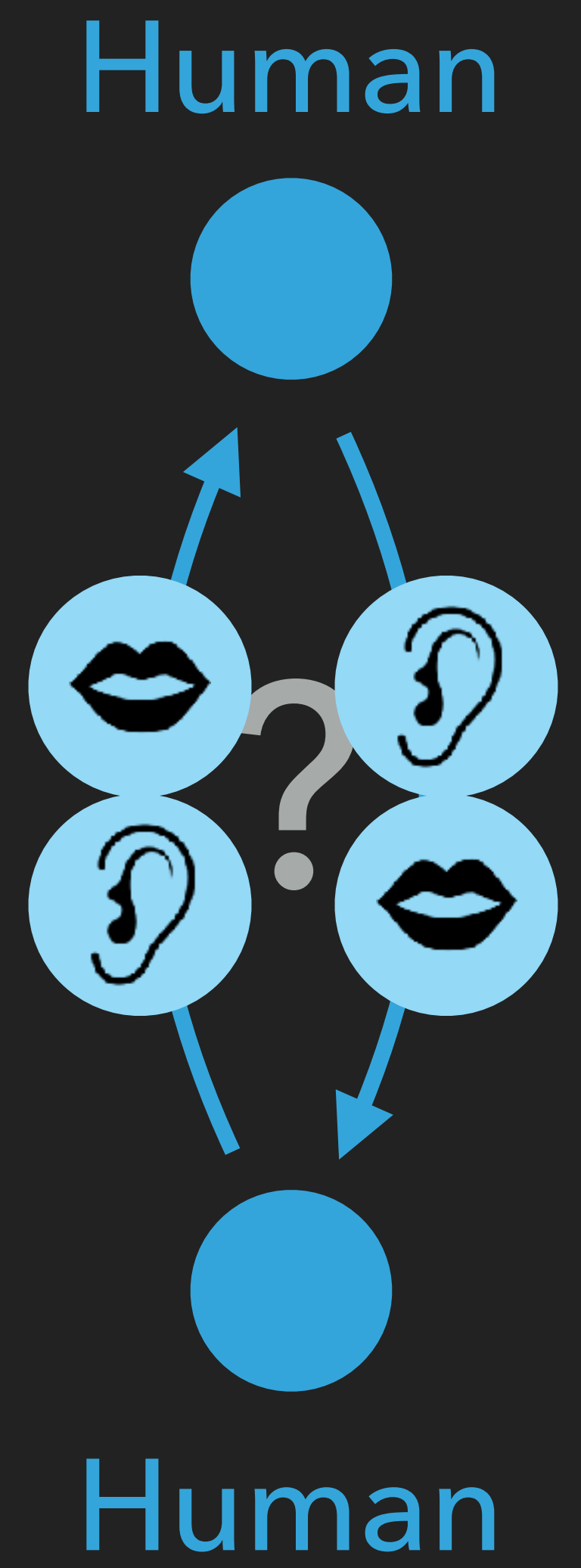
What if the user was a function?

Computer

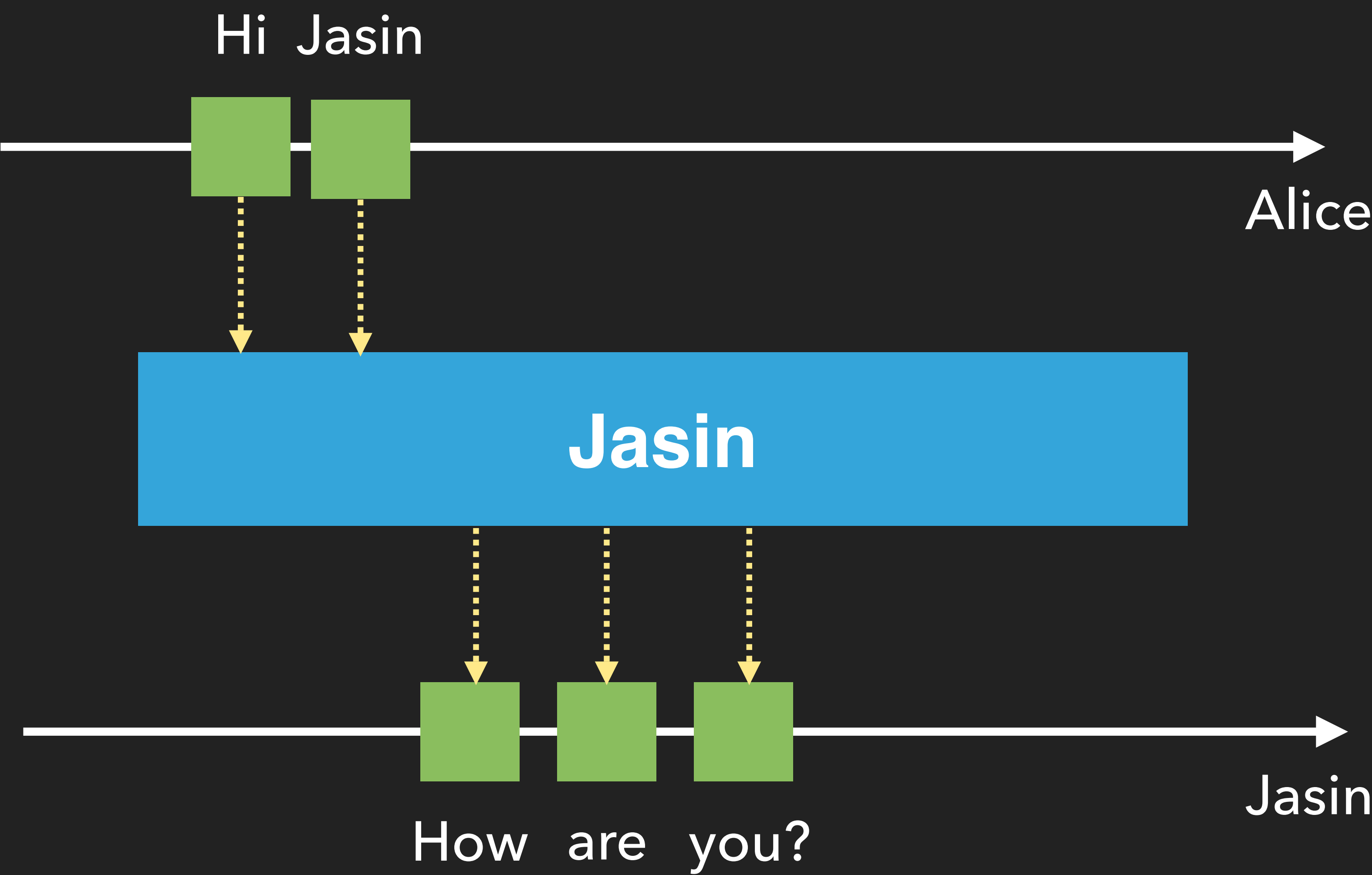


Human

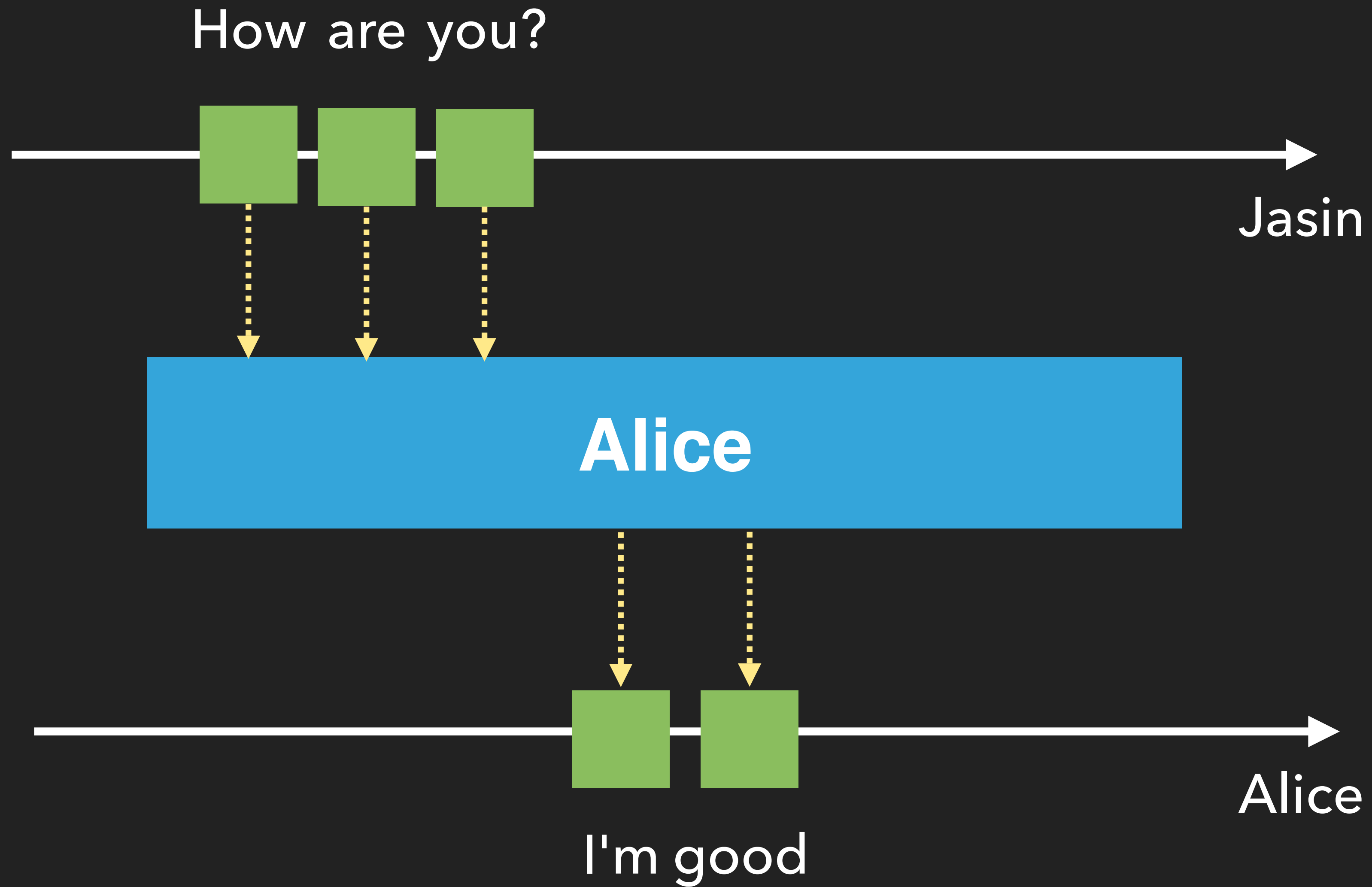
What if the user was a function?



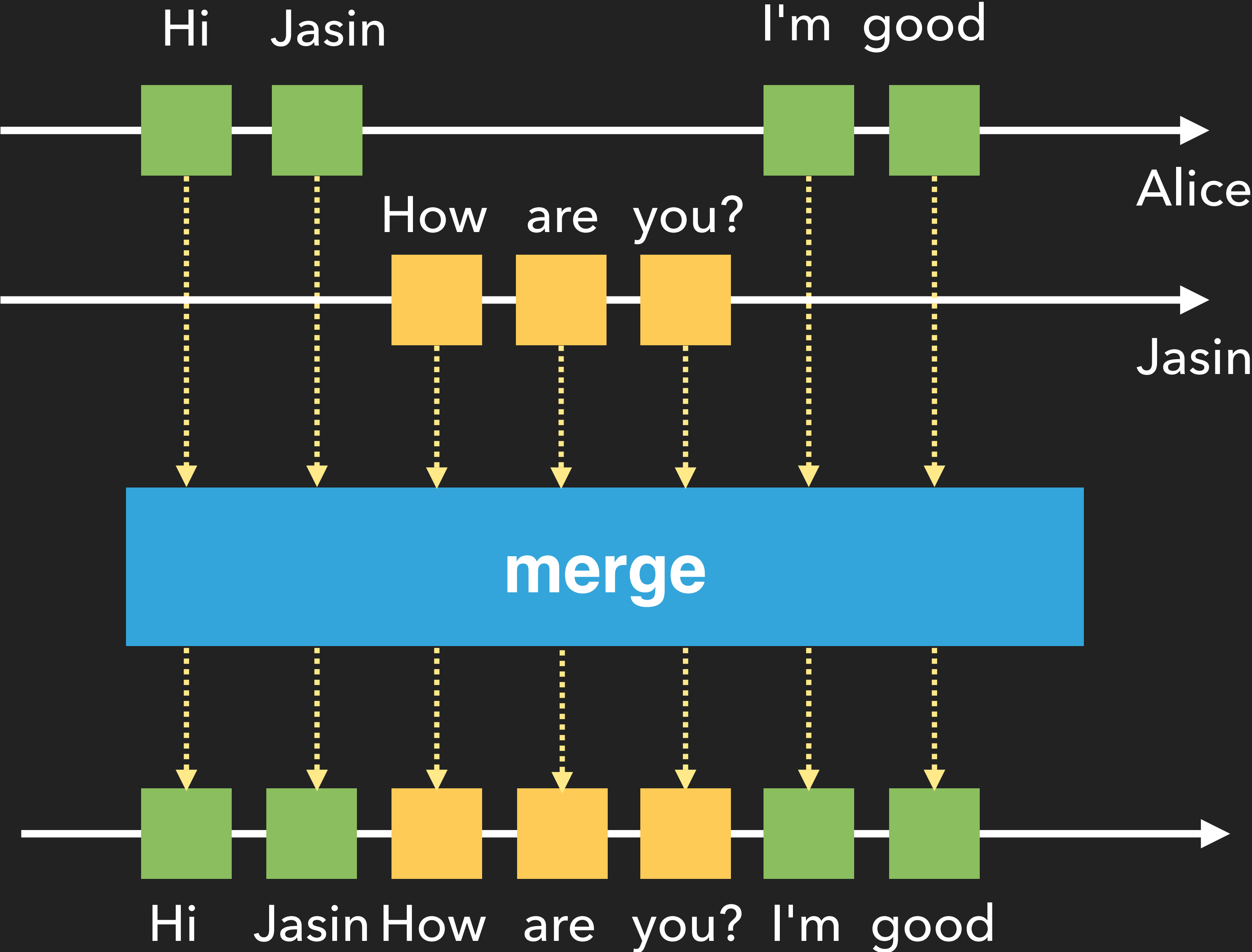
What if the user was a function?



What if the user was a function?



What if the user was a function?



What if the user was a function?



ONE MORE THING...



Cyclejs.cn

THANKS!

REFERENCE

- ▶ [Cycle.js](#)
- ▶ [RxJS](#)
- ▶ [Functional Programming - 6.1 Streams](#)
- ▶ [What if the user was a function? —Andre staltz](#)
- ▶ [流动的数据——使用 RxJS 构造复杂单页应用的数据逻辑 ——徐飞](#)
- ▶ [单页应用的数据流方案探索 ——徐飞](#)