Declarative
Reactive
Functional
iOS
Application
Architecture

Declarative?

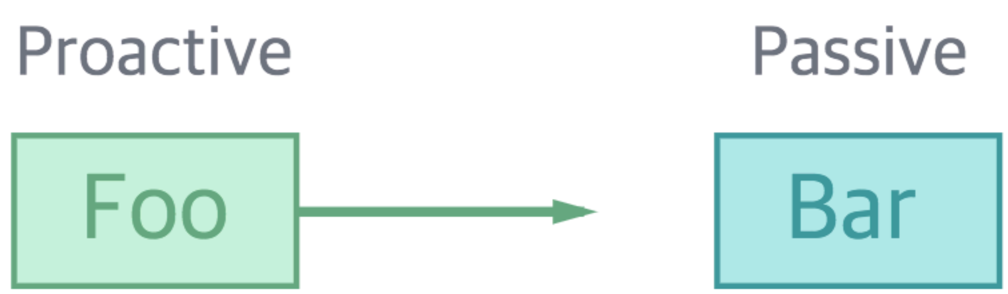
```swift
let label = UILabel()
label.textColor = .red
label.textAlignment = .center
```

Declarative?

```swift
struct LabelStyle {
    let textColor: UIColor
    let textAlignment: NSTextAlignment
}

func labelStyle(with color: UIColor, and alignment: NSTextAlignment) -> LabelStyle {
    return LabelStyle.init(
        textColor: color,
        textAlignment: alignment
    )
}

func run(label: UILabel, with style: LabelStyle) {
    label.textColor = style.textColor
    label.textAlignment = style.textAlignment
}

let label = UILabel()
let style = labelStyle(with: .red, and: .center)

run(label: label, with: style)
```

Reactive?

Reactive?




```
class Bar {
    func incrementCounter() {}
}

class Foo {
    var bar: Bar = Bar()

    func onNetworkRequest() {
        // ....
        bar.incrementCounter()
        // ....
    }
}
```

Reactive?


```
class Bar {
    init() {
        Foo.addOnNetworkRequestListener {
            self.incrementCounter()
        }
    }

    func incrementCounter() {
        // ....
    }
}

class Foo {
    static func addOnNetworkRequestListener(callback: @escaping () -> ()) {
        // ....
    }
}
```

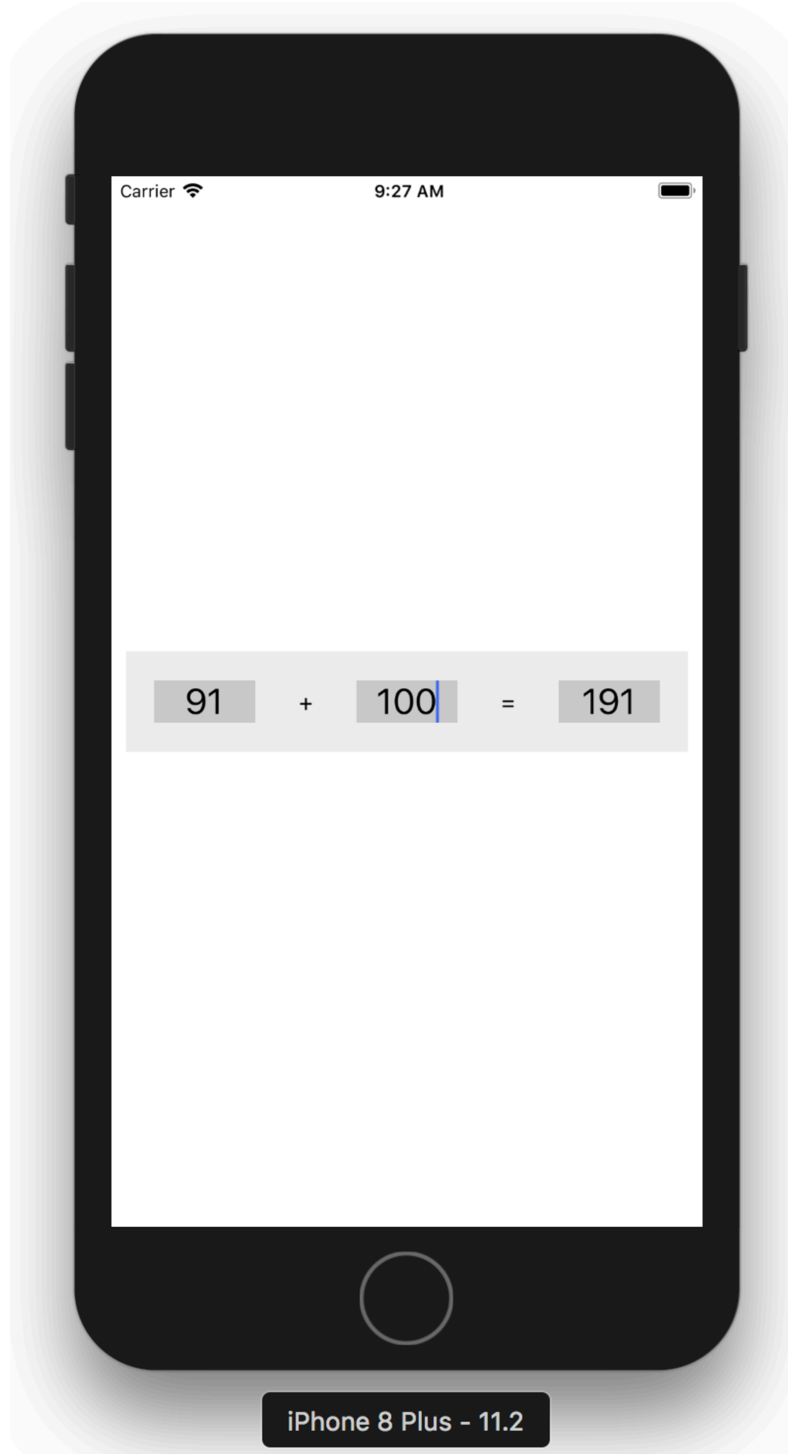
Functional?

```
// f: (A) -> B
// g: (B) -> C
// g . f: (A) -> C

func compose<A, B, C>(_ f: @escaping (A) -> B, _ g: @escaping (B) -> C) -> (A) -> C {
    return { x in
        return g(f(x))
    }
}

func increment(_ x: Int) -> Int {
    return x + 1
}

func square(_ x: Int) -> Int {
    return x * x
}

let incrementAndThenSquare = compose(increment, square)
```

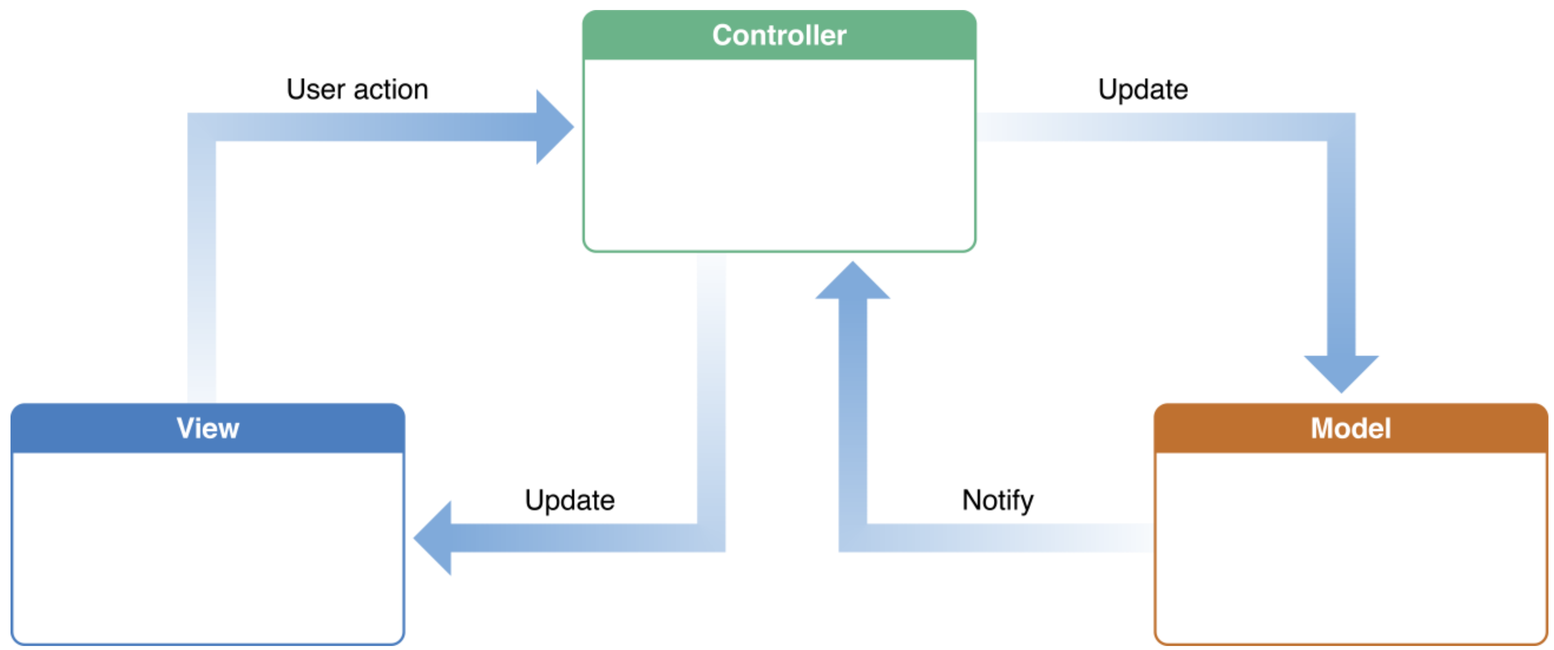
Architecture List

1. MVC
2. (Reactive) MVVM
3. React
4. Redux
5. Cycle.js
6. Elm

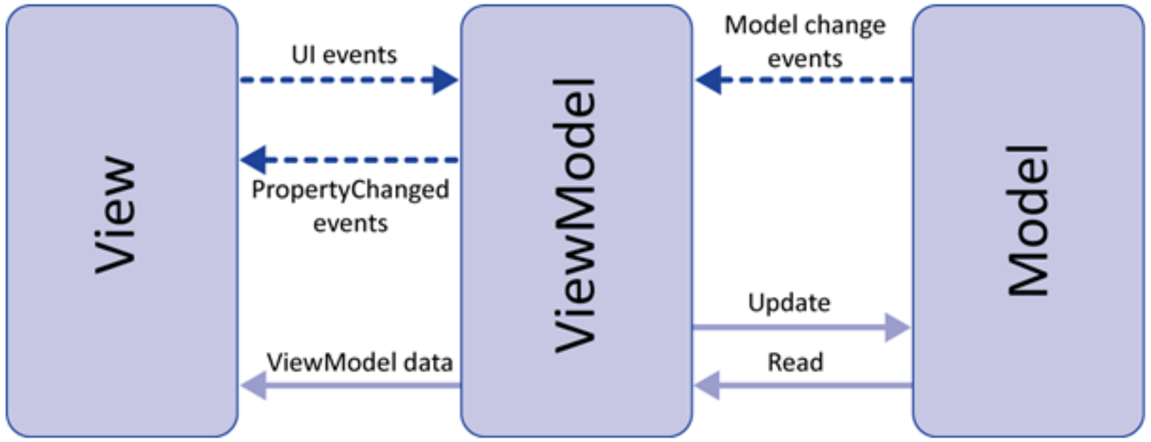
Sample App



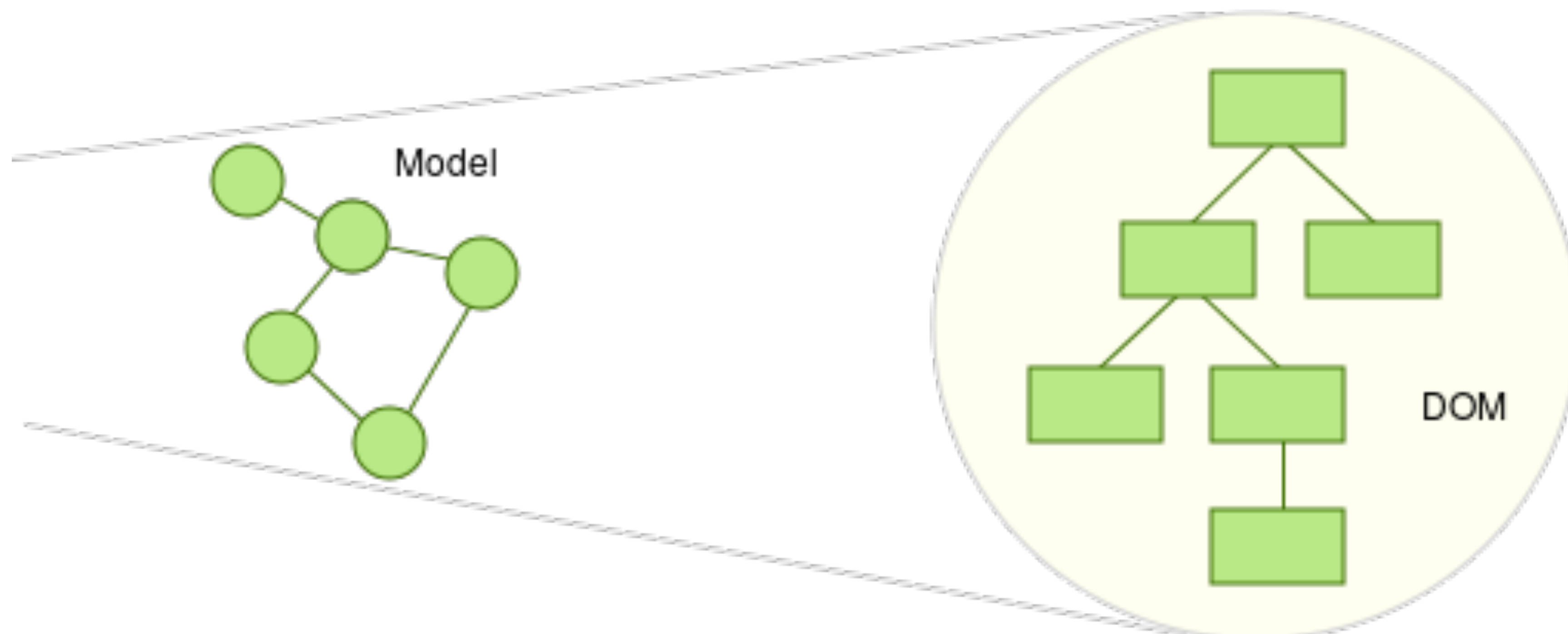
MVC

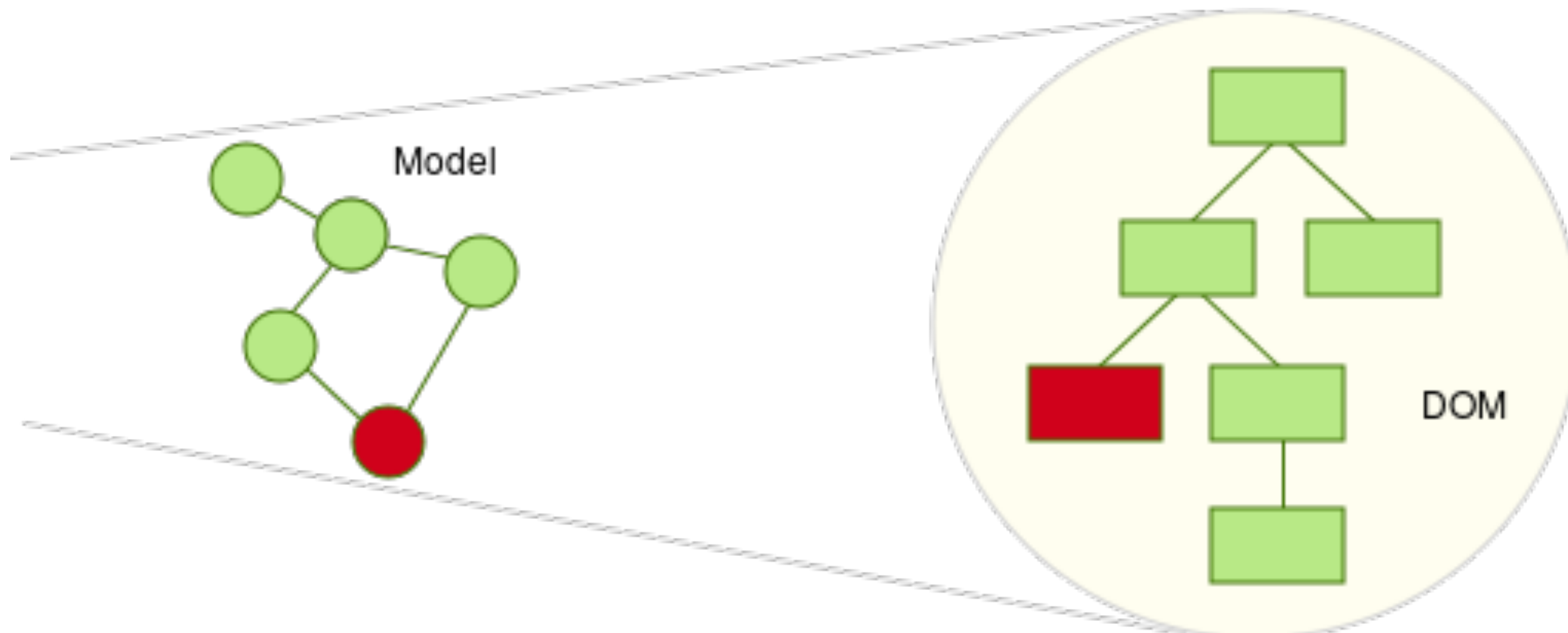


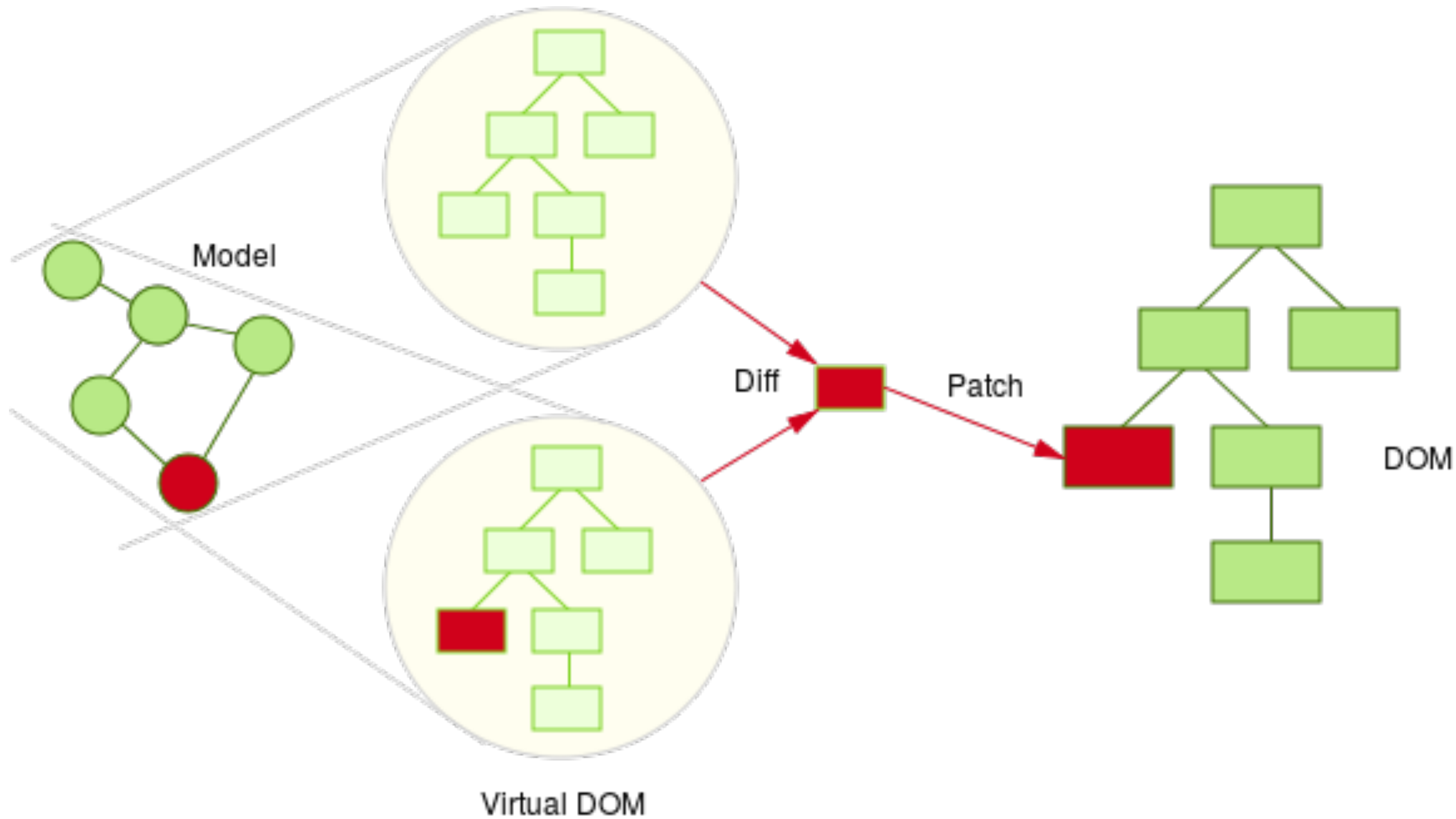
MVVM


View
UI events
PropertyChanged
events
ViewModel data
ViewModel
Model change
events
Update
Read
Model

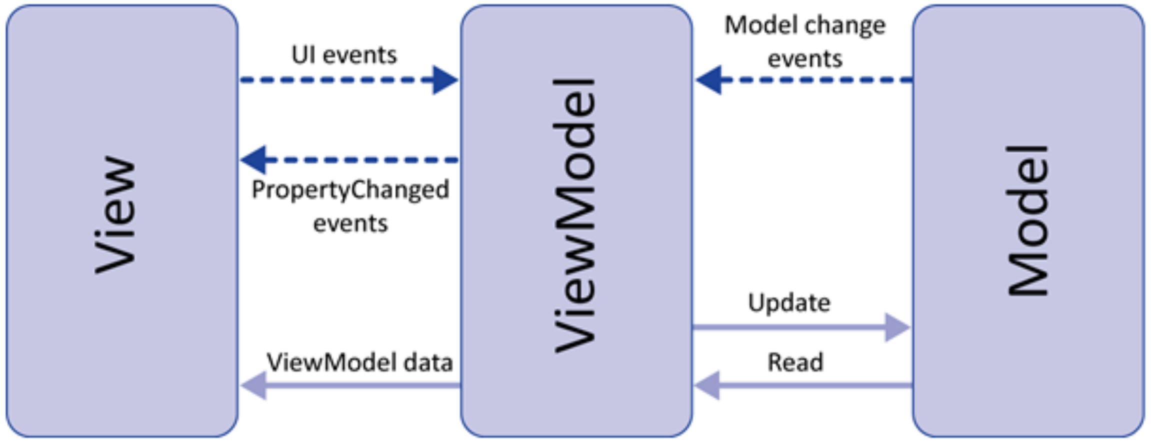
React



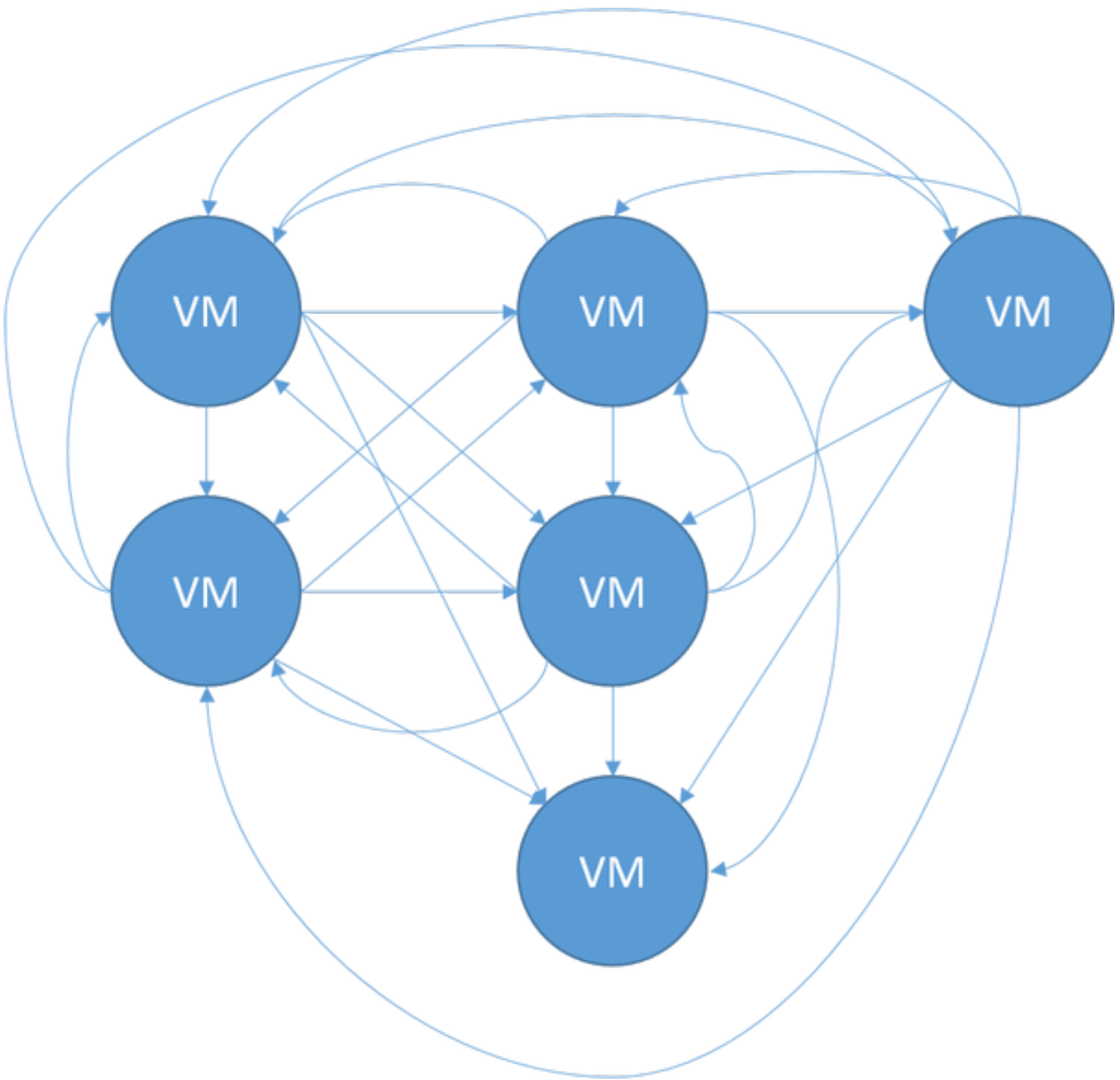

Model
DOM

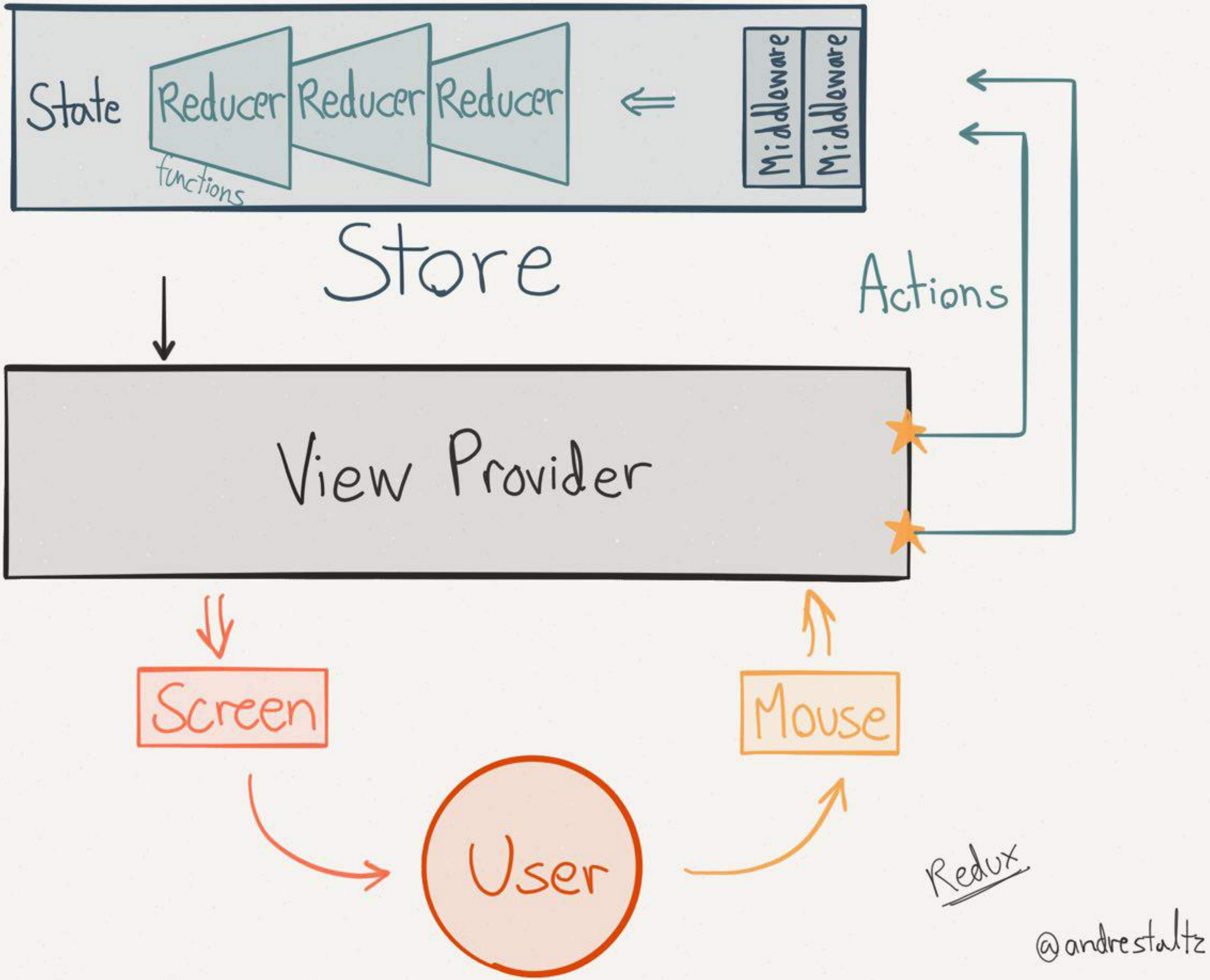

Model
Diff
Patch
DOM
Virtual DOM

Redux

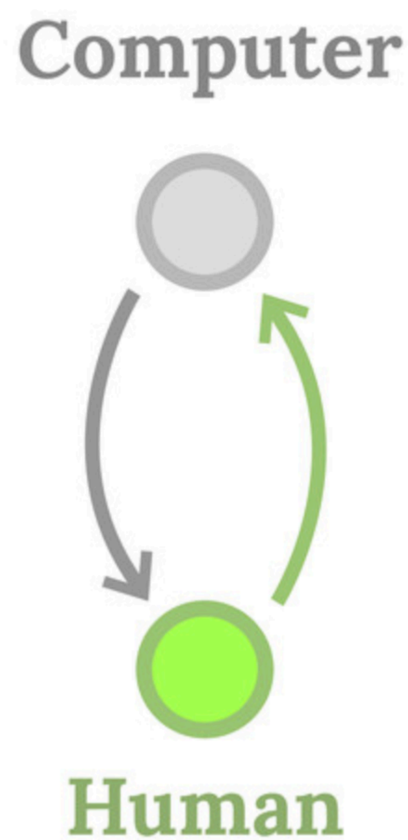


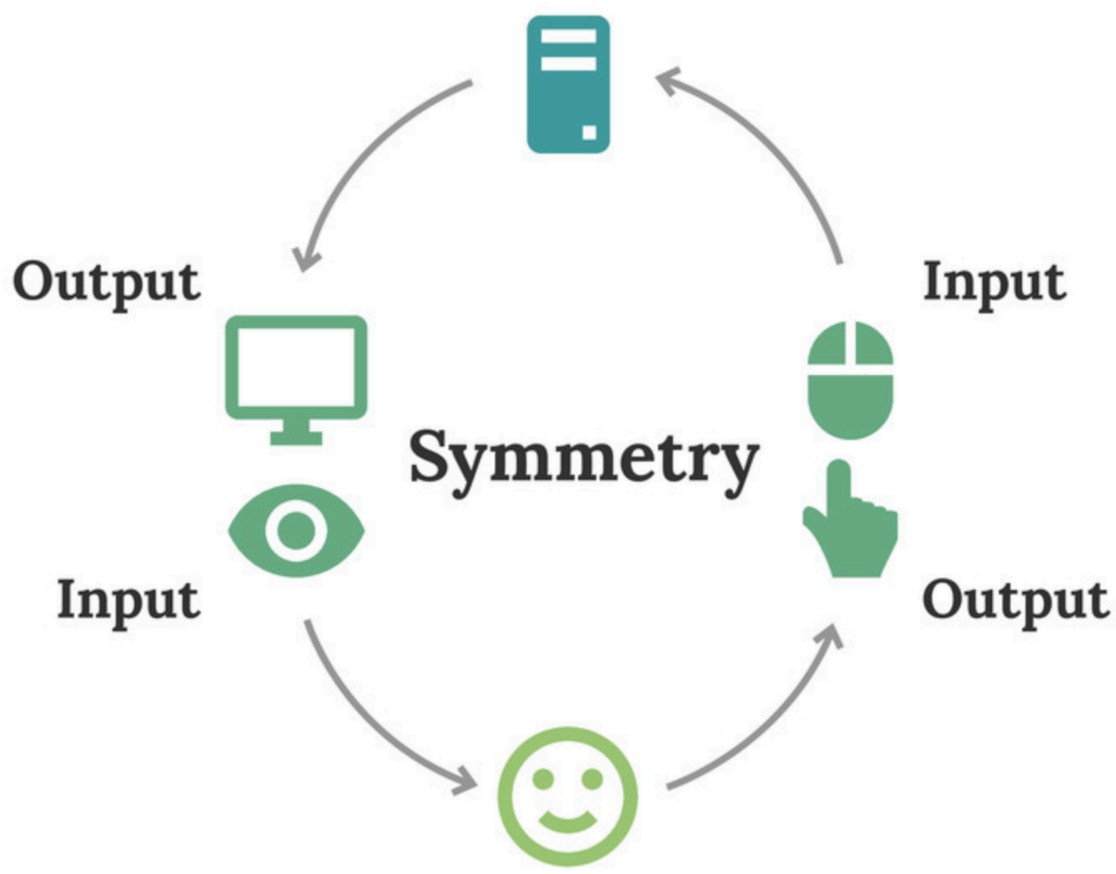
View
ViewModel
Model
UI events
PropertyChanged events
ViewModel data
Model change events
Update
Read

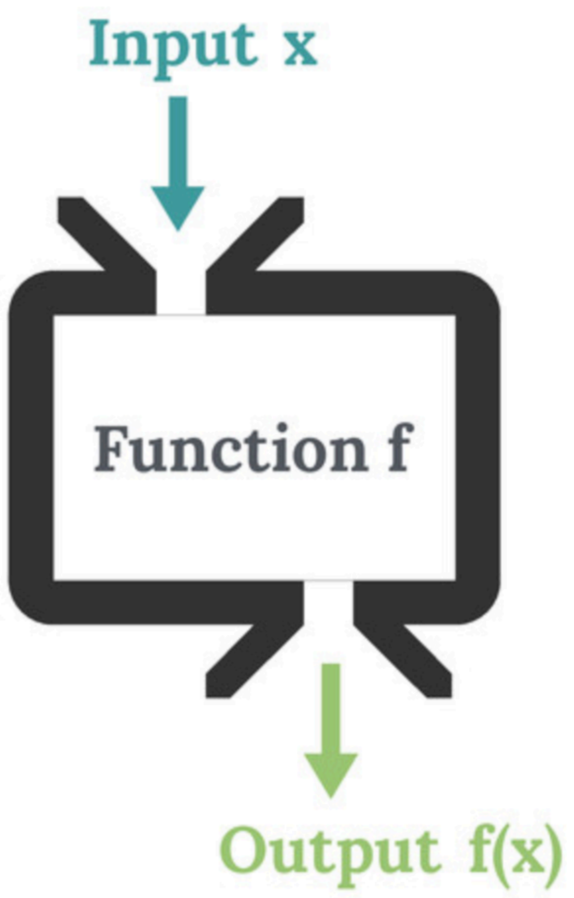

VM
VM
VM
VM
VM
VM


State
Reducer
Reducer
Reducer
functions
Middleware
Middleware
Store
Actions
View Provider
Screen
Mouse
User
Redux
@andrestaltz

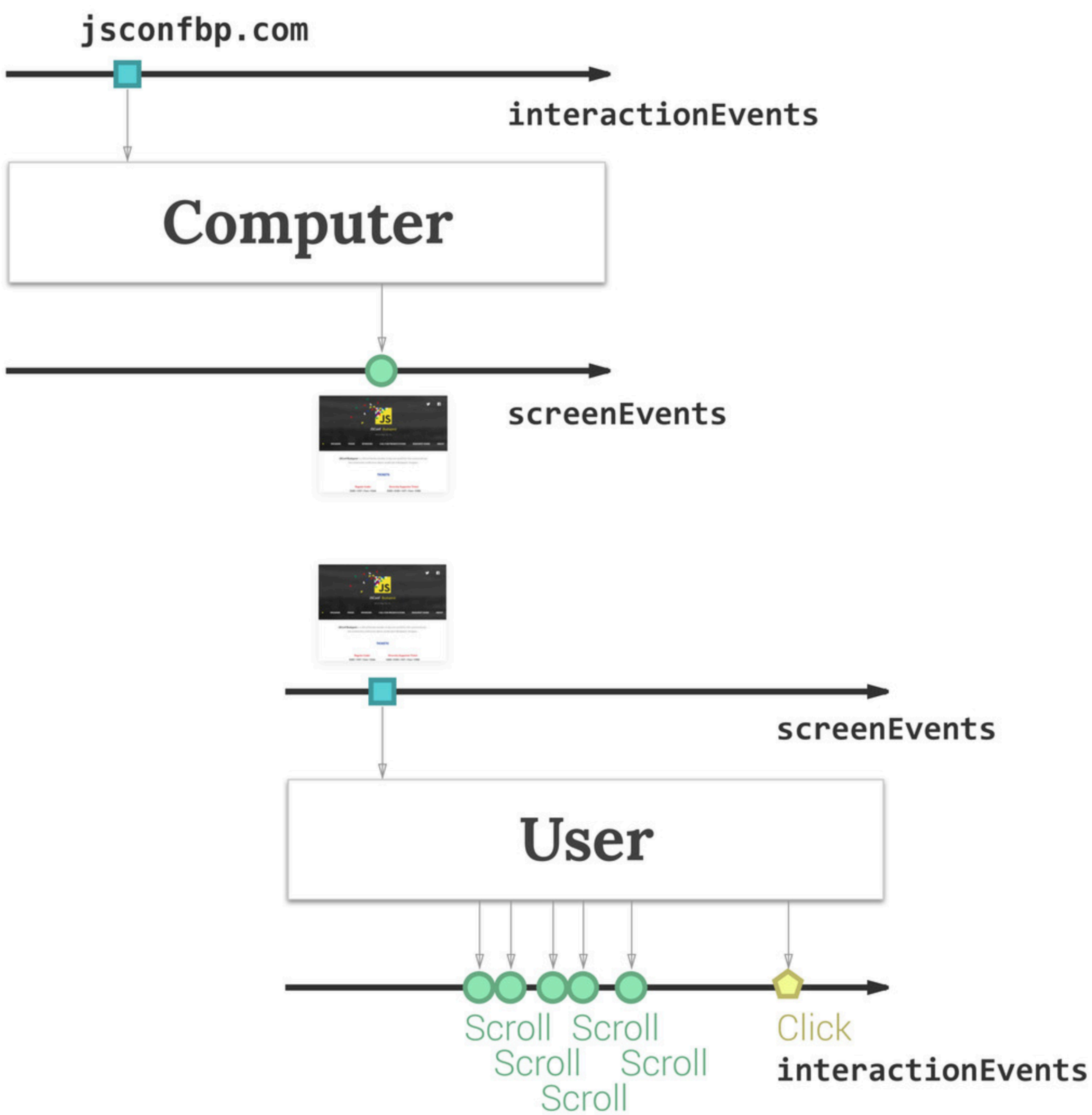
Cycle.js

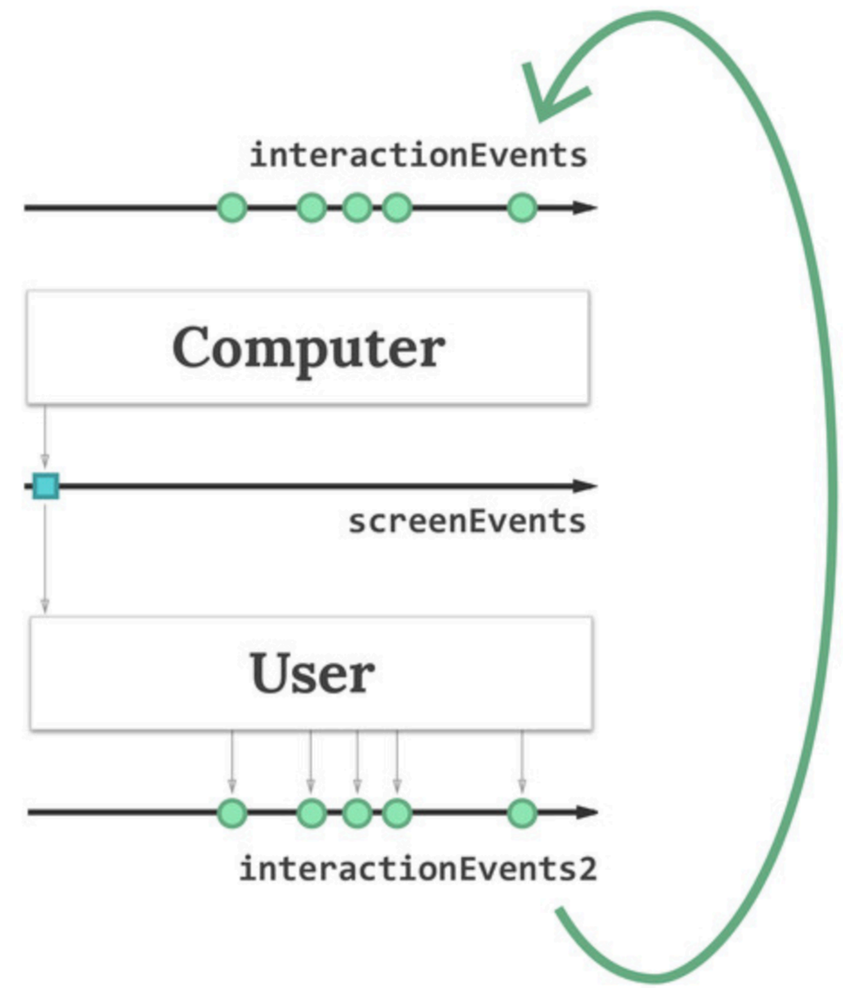



Output
Input
Symmetry
Input
Output


Input x
Function f
Output f(x)

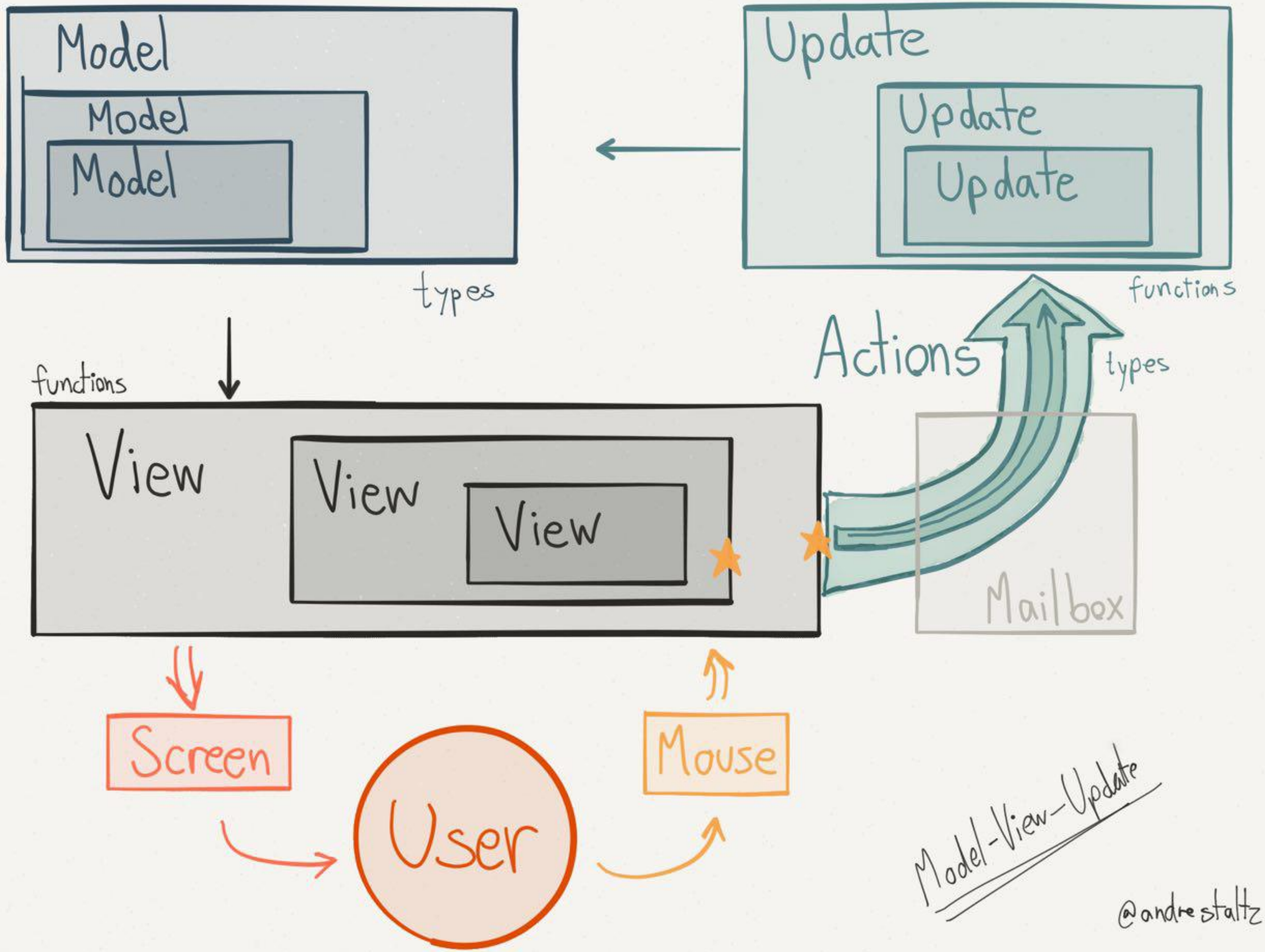
```kotlin
typealias Program = (Observable<Event>) -> Observable<Effect>
typealias User = (Observable<Effect>) -> Observable<Event>
```


jsconfbp.com
interactionEvents
Computer
screenEvents
screenEvents
User
Scroll
Scroll
Scroll
Scroll
Scroll
Click
interactionEvents


interactionEvents
Computer
screenEvents
User
interactionEvents2

Elm

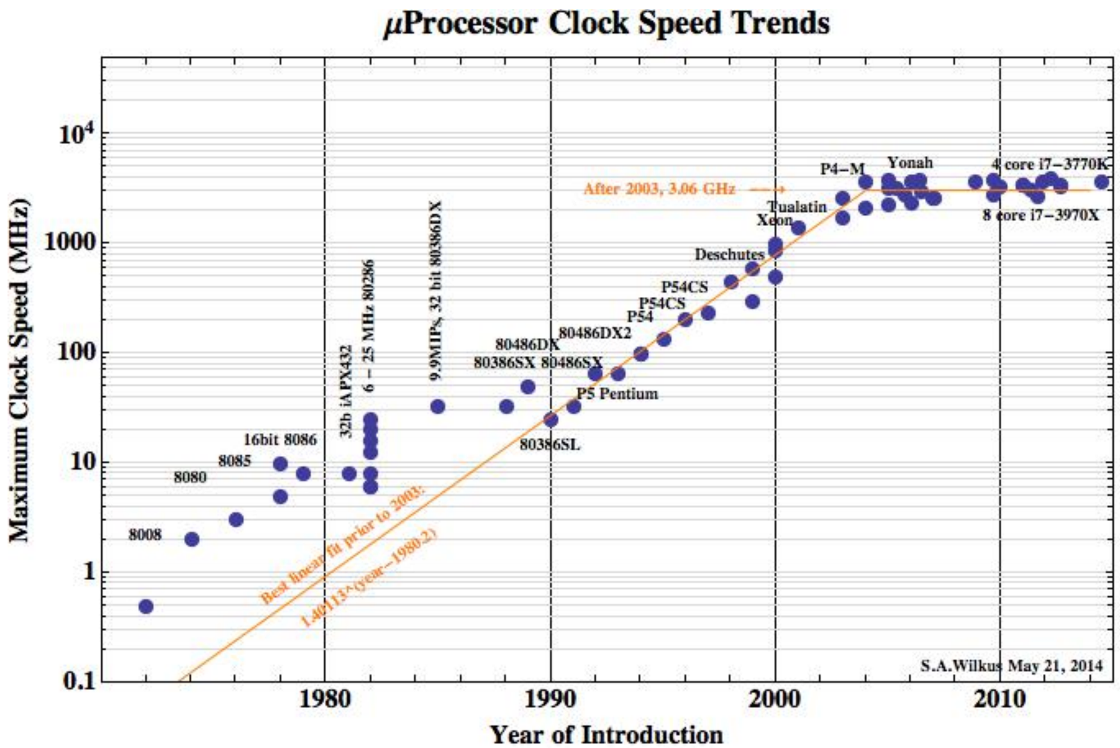

```
typealias Program<Message> = (Message) -> Command<Message>
```

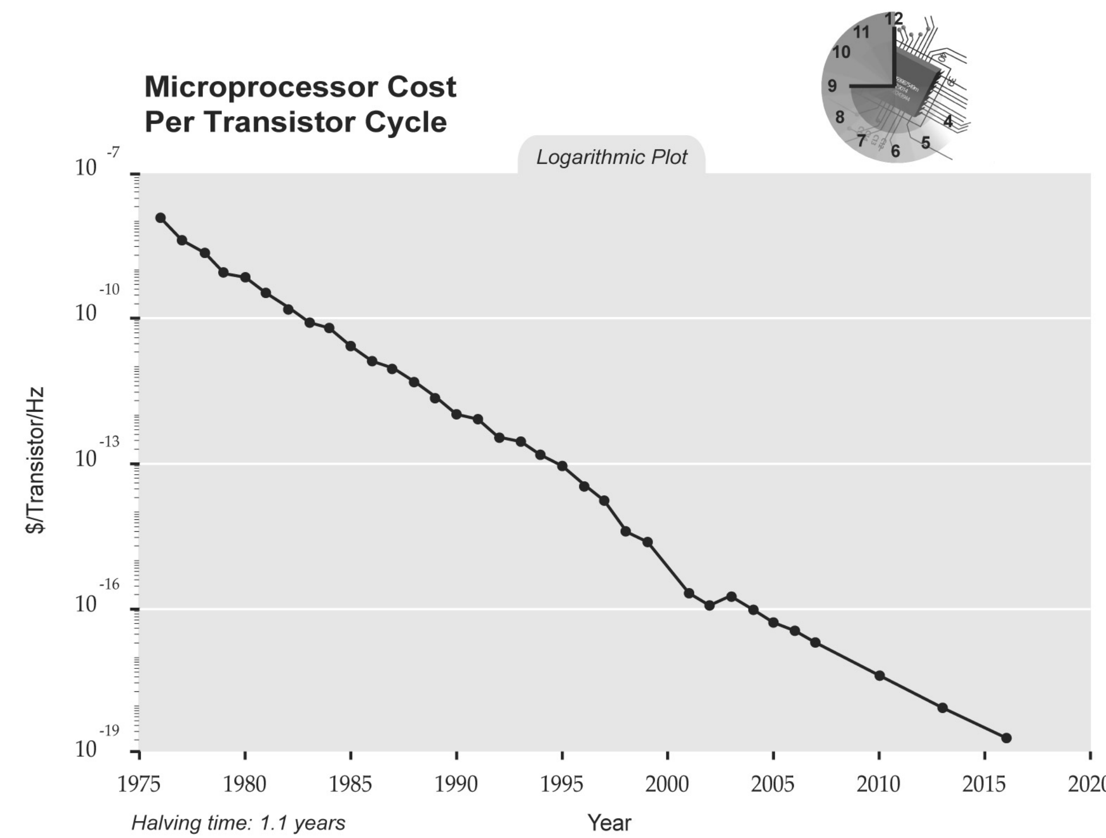

Model
Model
Model
types
Update
Update
Update
functions
Actions
types
functions
View
View
View
Mailbox
Screen
User
Mouse
Model-View-Update
@andrestaltz

Why Functional?

무어의 법칙

18개월마다 칩에 집적할 수 있는 트랜지스터 수가 2배씩 증가할 것





기종	코어 수
~iPhone4	1
iPhone4s ~ 6s	2
iPhone 7	4
iPhone 8, X	6
iPhone 20	??

The free lunch is over

이제 공짜 점심을 즐기기 위해서는 Concurrency, Parallelism을 극대화할 수 있는 방식으로 소프트웨어를 작성해야 한다

```
protocol ComplexObject {
    func doSomething()
    var getValue: Int { get }
}

class Foo: ComplexObject {
    private var privateValue: Double = 0

    func doSomething() {
        // ...
        privateField = 1
        // ...
    }

    var getValue: Int {
        return Int(privateValue)
    }
}

let foo = Foo()
let bar = foo

bar.doSomething()

print(bar.getValue) // prints "1"
print(foo.getValue) // prints "1"
```

C#	Async await
Go	Goroutines
Scala(Akka)	Actor
FP	불변 데이터, 순수함수, 함수합성

Why Reactive?

```
@objc func leftOperandFieldDidChange(_ sender: UITextField) {
        guard let number = sender.text.map(Int.init) else { return }
        model.leftOperand = number
        resultLabel.text = model.result?.description
}

@objc func rightOperandFieldDidChange(_ sender: UITextField) {
        guard let number = sender.text.map(Int.init) else { return }
        model.rightOperand = number
        resultLabel.text = model.result?.description
}
```

```
let leftOperand$ = leftOperandSubject
            .asObservable()
            .map { $0.flatMap(Int.init) }
            .filterNil()

let rightOperand$ = rightOperandSubject
            .asObservable()
            .map { $0.flatMap(Int.init) }
            .filterNil()

result$ = Observable.combineLatest(leftOperand$, rightOperand$)
            .map(+)
            .map { $0.description }
```

불변식(invariant)

Why Declarative?

응용 소프트웨어

현실의 문제를 컴퓨터를 이용해 해결하는 것

무엇을 해결할 것인가

어떻게 해결할 것인가

기계어 - 프로그래밍 언어

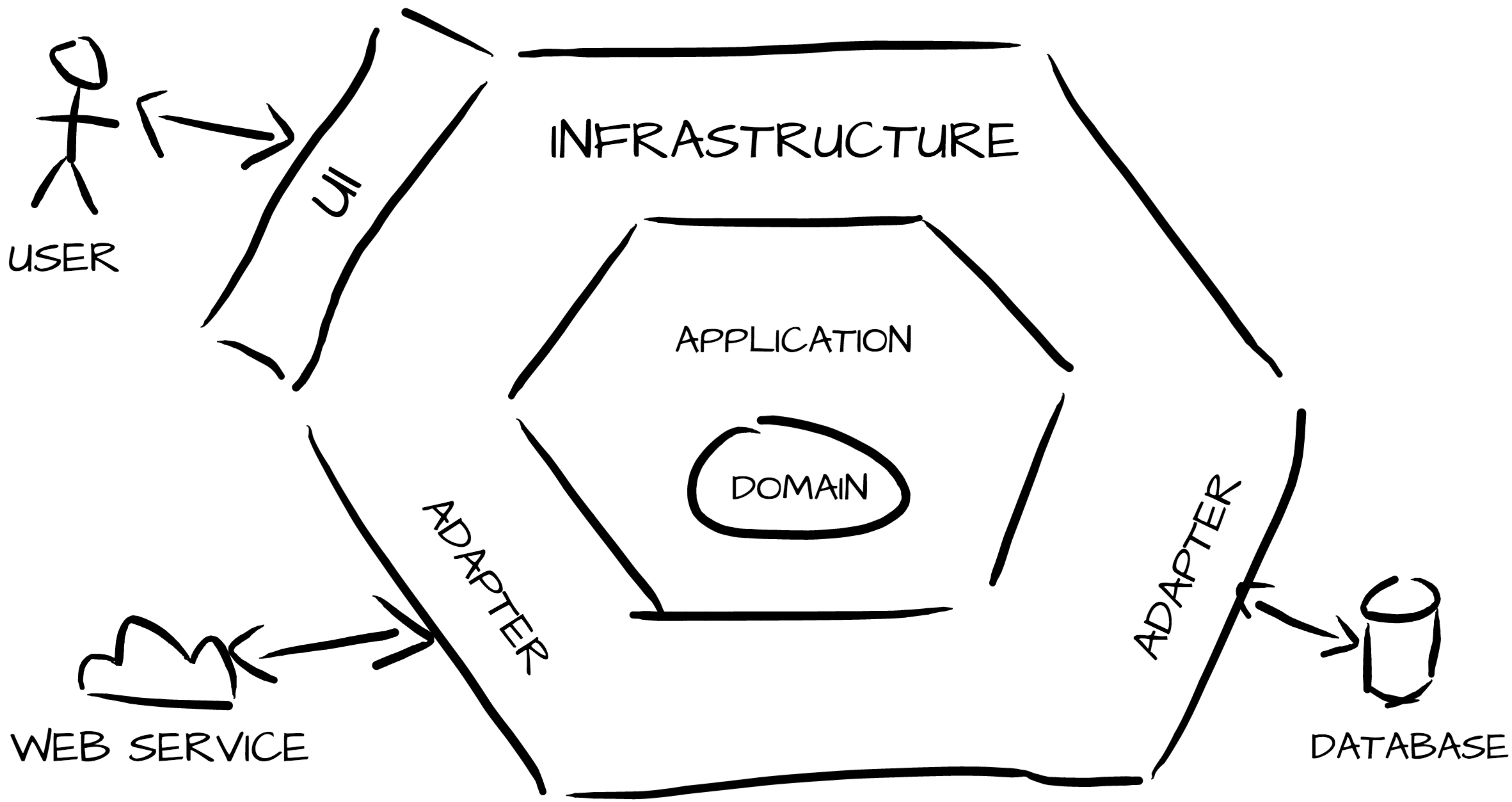
데이터센터 - AWS - Docker

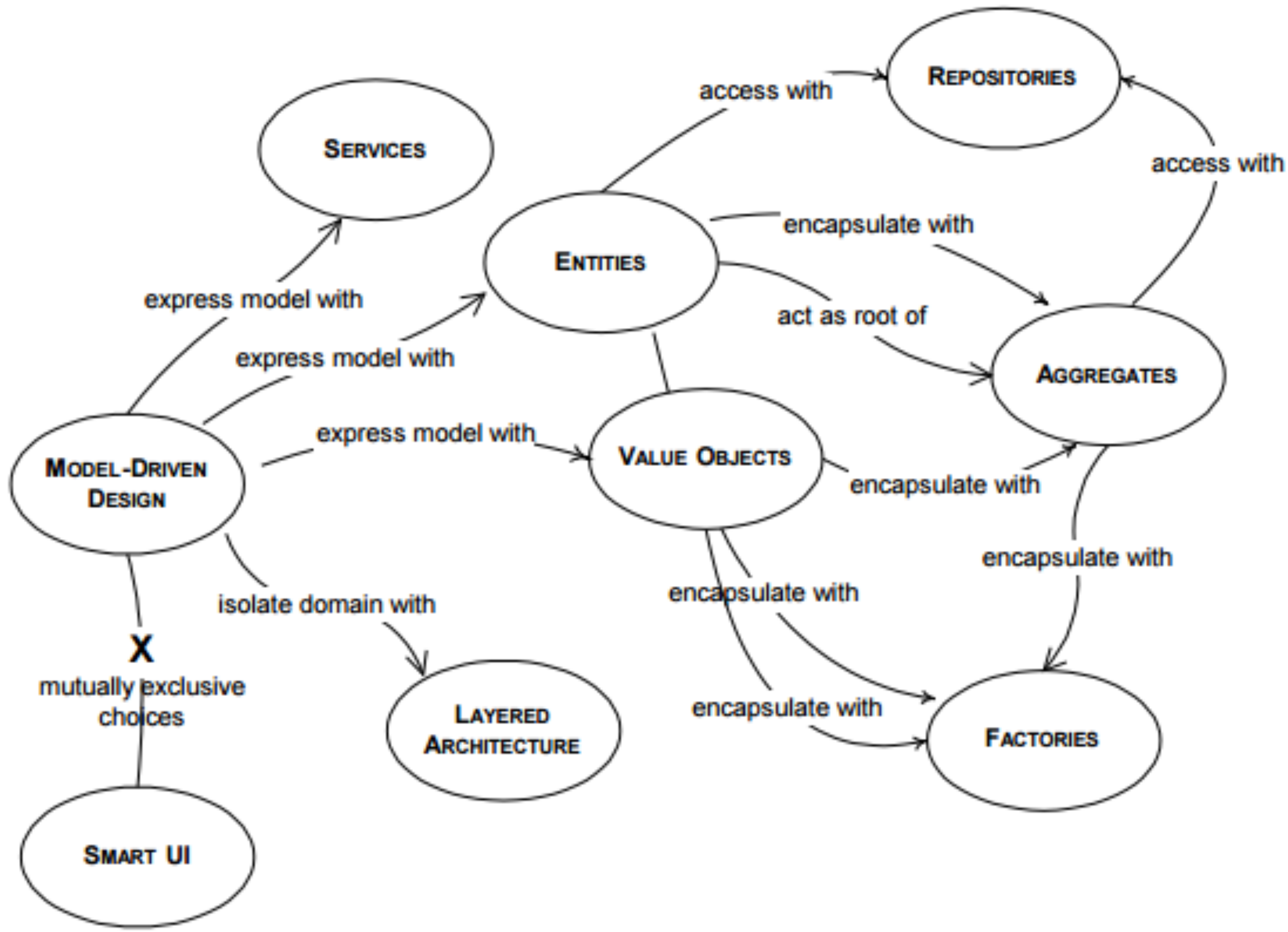
추상화

How → What

DDD(Domain Driven Design)

소프트웨어의 복잡성은 도메인에서 기인하고, 그러한 복잡성을 어떻게 다루느냐가 프로젝트의 성패를 좌우한다

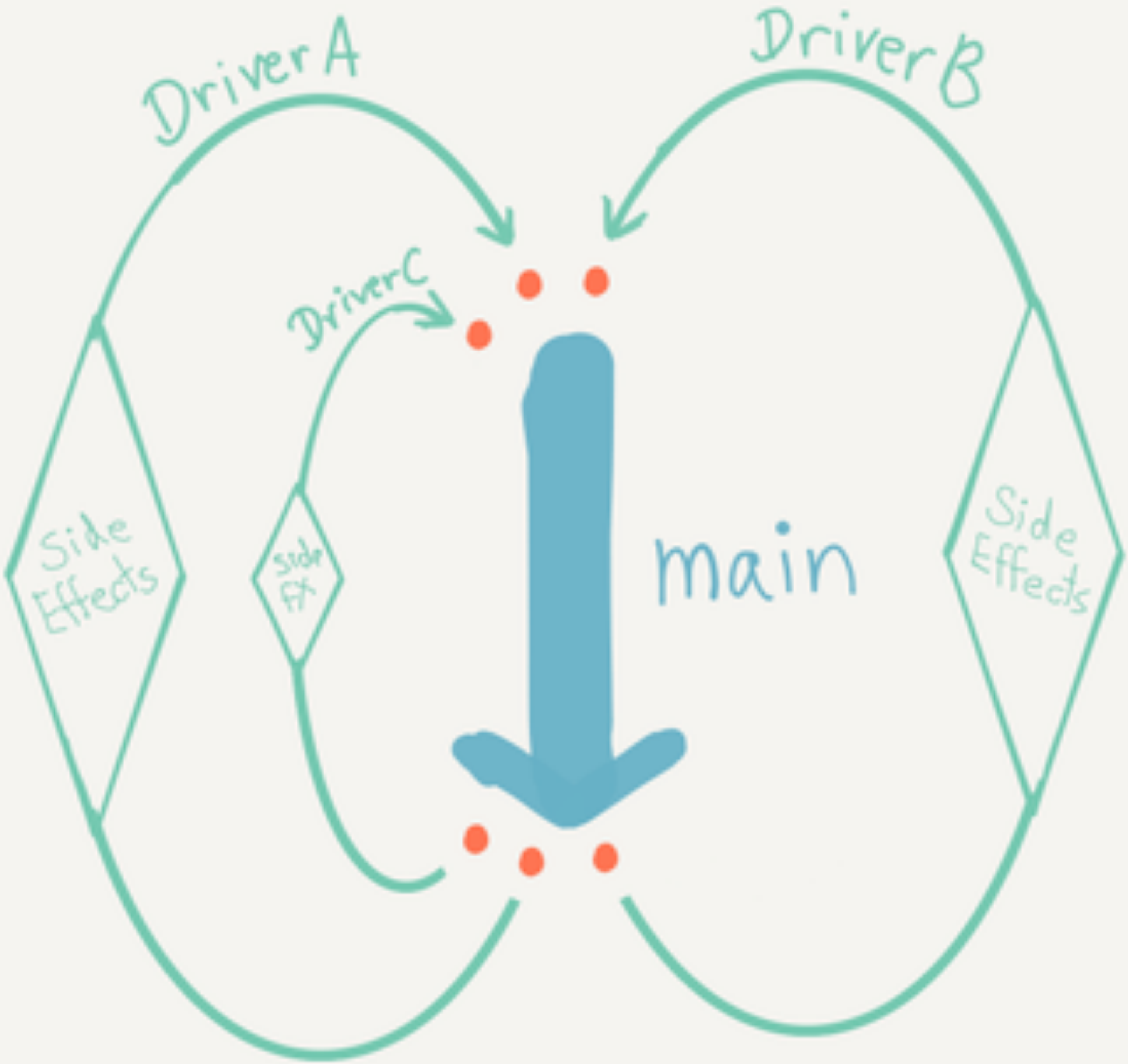

USER
UI
INFRASTRUCTURE
APPLICATION
DOMAIN
ADAPTER
ADAPTER
WEB SERVICE
DATABASE



REPOSITORIES
access with
access with
SERVICES
ENTITIES
encapsulate with
express model with
act as root of
express model with
AGGREGATES
express model with
VALUE OBJECTS
MODEL-DRIVEN DESIGN
encapsulate with
encapsulate with
isolate domain with
encapsulate with
X
mutually exclusive choices
encapsulate with
LAYERED ARCHITECTURE
FACTORIES
SMART UI

Drivers in Cycle.js

Arrows are functions
Dots are Observables
Main is your app function



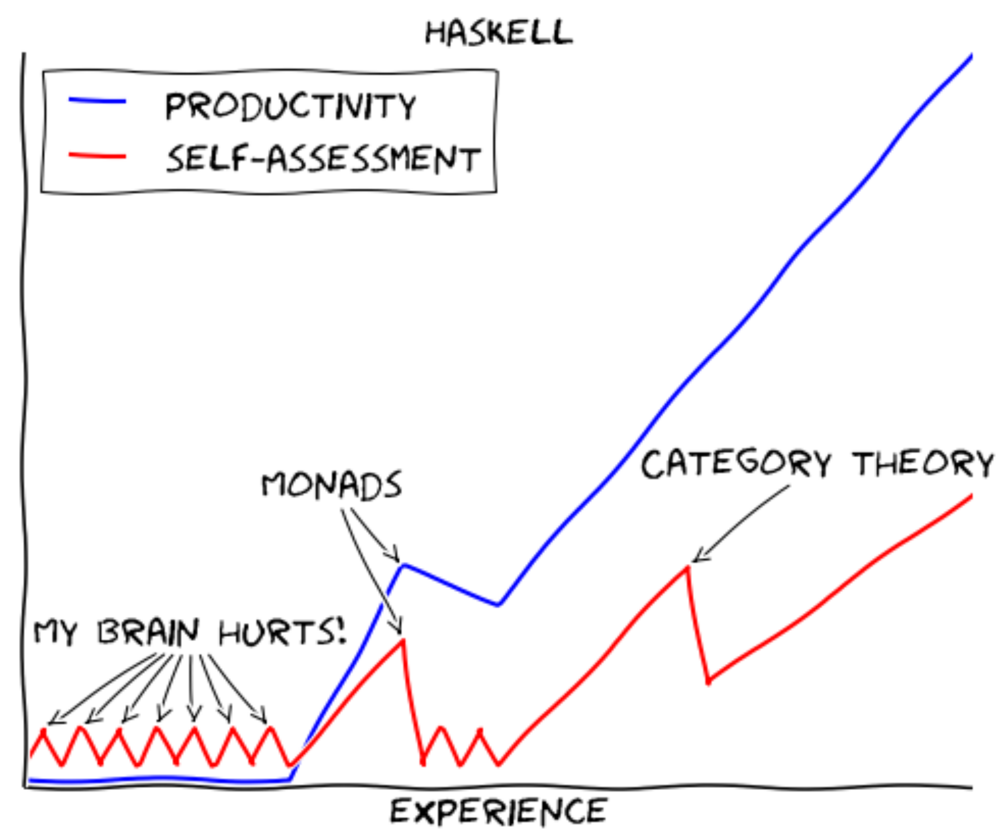
@andrestaltz

More declarative
More reactive
More functional

Part One

Part Two

Part Three



implement once, declare everywhere