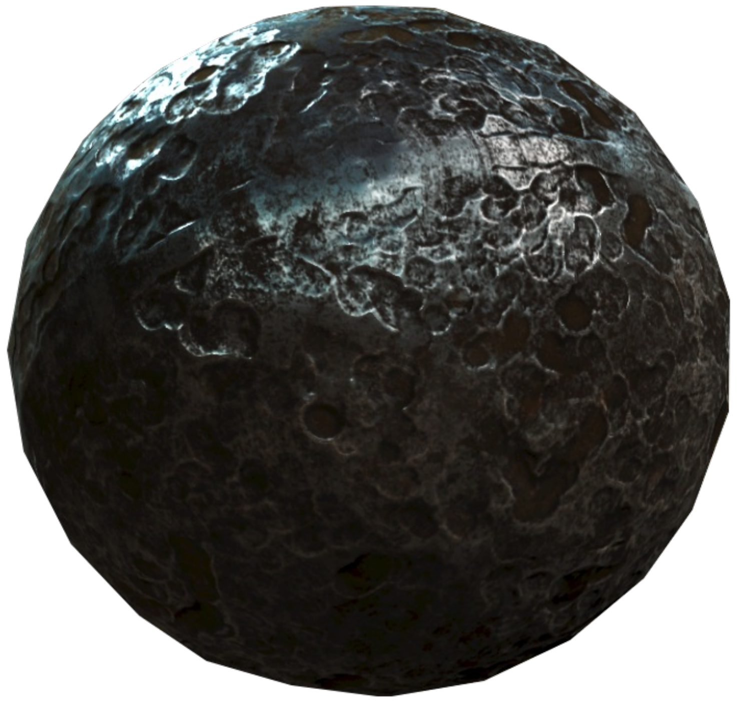
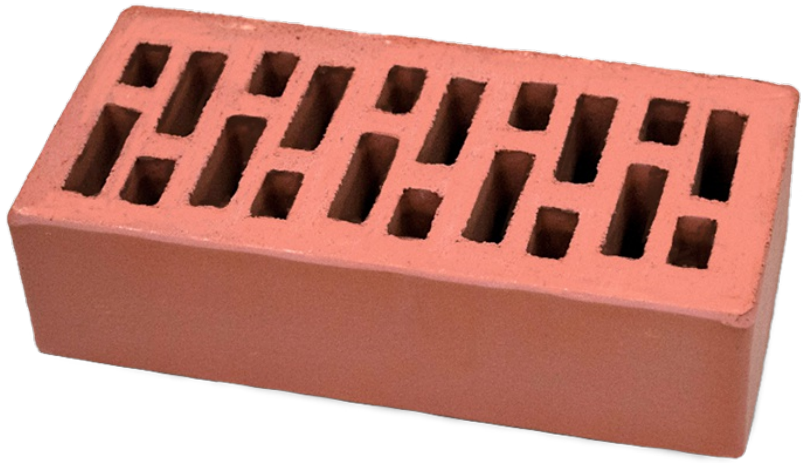
От фреймворков к сверхфреймворкам



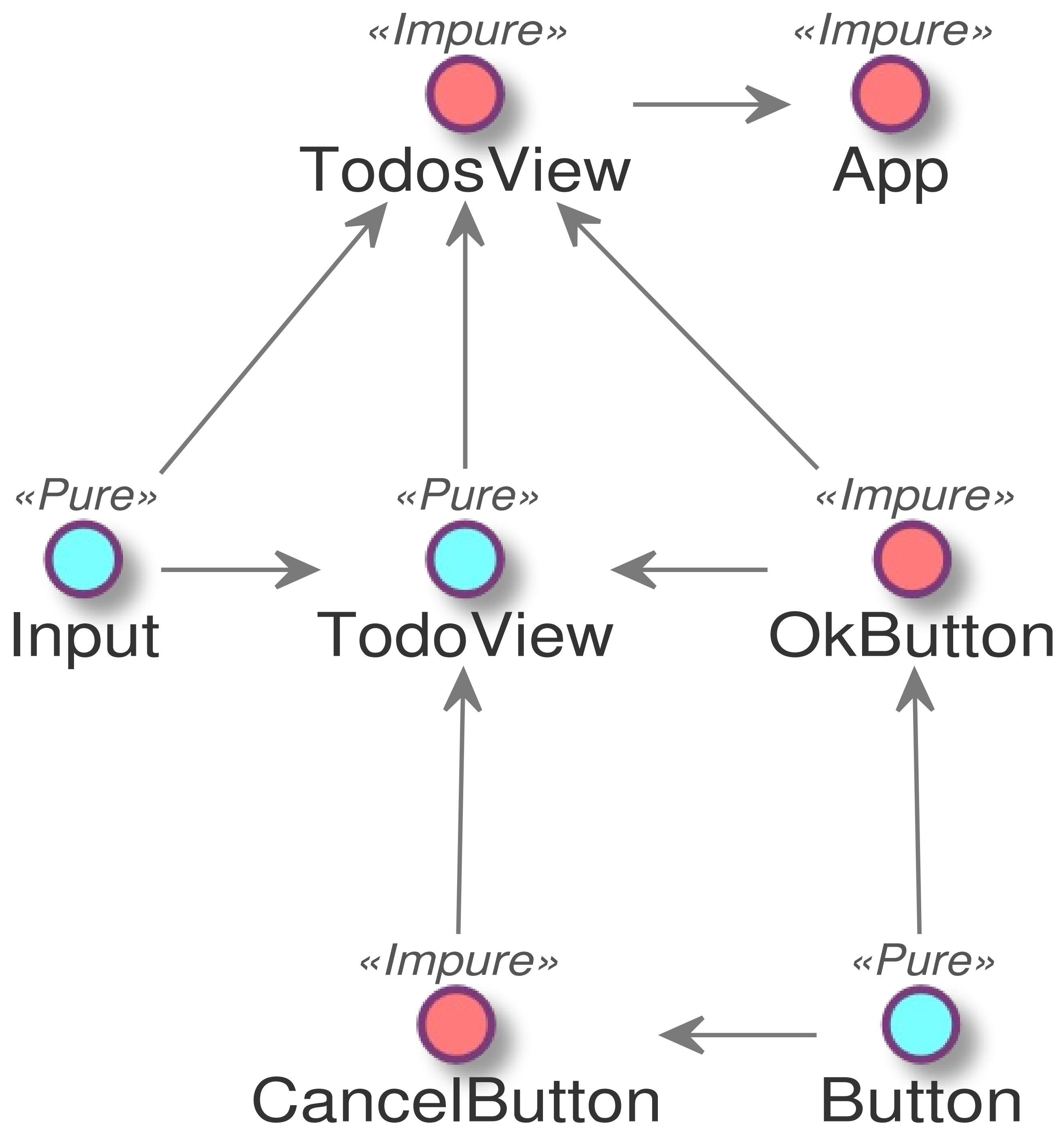
- Component = view + data + logic
- React.setState, redux, rxjs, mobx?
- ts, flow (Angular2 driven)

- PHP - Symfony, silex
- Легкий каркас, библиотека, интеграция
- Микросервисы, микроядерность
- JS - Angular2



- f(props)
- f(context)(props)
- new F(context).method(props)

Компоненты


«Impure»
TodosView
«Impure»
App
«Pure»
Input
«Pure»
TodoView
«Impure»
OkButton
«Impure»
CancelButton
«Pure»
Button

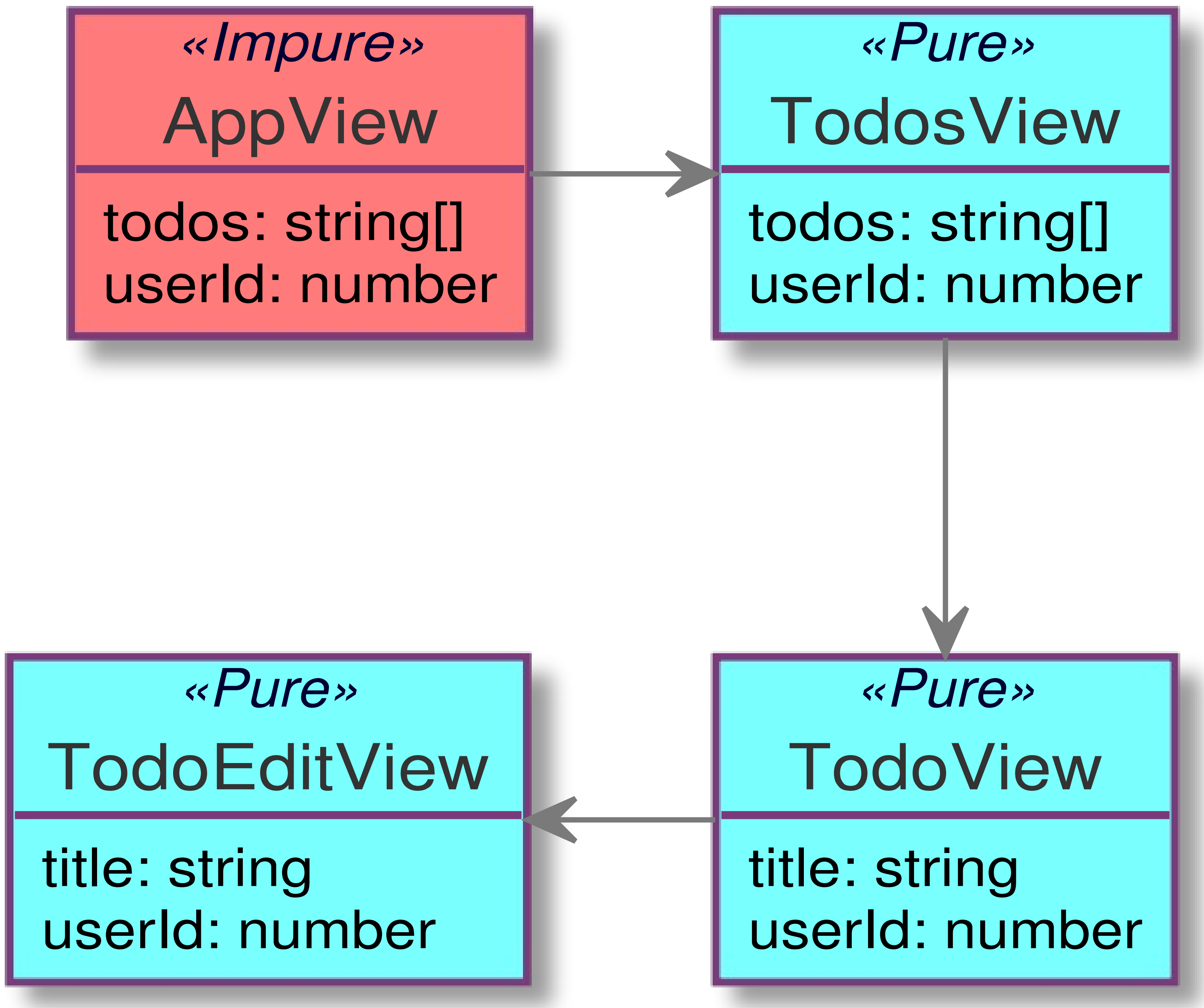
React

- Presentional (view)
- No framework reuse
- Container (injector, view)
- No app reuse

Чистый компонент

```
function CounterView(props: {count: number}) {
  return <div>  Count: {props.count} </div>
}
```

- JSX + flow = контракт к шаблонам
- Кастомизируемость
- Рефакторинг: O(depth * props)


«Impure»
AppView
todos: string[]
userId: number
«Pure»
TodosView
todos: string[]
userId: number
«Pure»
TodoEditView
title: string
userId: number
«Pure»
TodoView
title: string
userId: number

```
function CounterView({count}) {
  return React.createElement('div', null, 'Count: ', count)
}
```

- чистый компонент != чистая функция
- ослабить связь

vue-jsx

```
Vue.component('jsx-example', {
  render (h) { // <-- h must be in scope
    return <div id="foo">bar</div>
  }
})
```

h auto-injection

```
Vue.component('jsx-example', {
  render () {
    // const h = this.$createElement
    return <div id="foo">bar</div>
  }
})
```

- Зависимость от Vue.component

Нуль-компонент

```
function CounterView({count}, h: CreateElement) {
  return h('div', null, 'Count: ', count)
}
```

h auto-injection

```
function CounterView({count} /* ,h */) {
  return <div>Count: count</div>
}
```

Переиспользовать

Компонент с состоянием

- view = component(state)(props)
- state - труднее кастомизировать
- O((depth * subProps) + state)
- props = subProps + state

```
class CounterView
  extends React.Component<void, {name: string}, {count: number}> {

  state = {count: 1}

  constructor(props: Props) { super(props) }

  add() {
    this.setState({ count: this.count++ })
  }

  render() { /* ... */ }
```

- Конструктор занят под props
- setState

```
import Component from 'my-react-like'
```

```
class CounterView
  extends Component<{name: string}, {count: number}> {

  some: Some

  constructor(some: Some) { super(); this.some = some }

  render() { /* ... */ }
}
// ...
```

```
<CounterView name={123} /> // 0 errors
```

Типы и JSX в Vue, Deku?

```
function CounterView(props: {count: number, add: () => void}) {
  return <div>
    {props.count}: <button onclick="{add}">Add</button>
  </div>
}
```

```
function mapStateToProps(store) {
  return { count: store.counter.count }
}
const CounterContainer = connect(mapStateToProps)(CounterView)
```

```
<Provider store={'XYZ'}> // unsafe
  <CounterContainer/>
</Provider>
```

```
class App extends React.Component {
  static childContextTypes = {
    store: PropTypes.object
  }

  getChildContext() {
    return { store: this.props.store }
  }

  render = () => <CounterContainer/>
}
```

```
class CounterContainer extends React.Component {
  static contextTypes = {
    store: PropTypes.object
  }

  render = () => CounterView({ count: this.context.store.count })
}
```

```
@Component({
  selector: 'my-counter',
  templateUrl: './counter.component.html'
})
class CounterView {
  counter: number = 0
  @Input name: string

  constructor(private counterService: CounterService) {}

  addCounter() {
    this.counter = this.counterService.add(this.counter)
  }
}
```

- Component = template + view model + logic
- PropTypes на constructor

```
const Injectable = 0 as any

interface ITest {}
class CounterService {}

@Injectable()
class CounterView {
  constructor(private cs: CounterService, test: ITest) {}
}
```

tsc --emitDecoratorMetadata test.ts

```
Reflect.metadata(CounterView, "design:paramtypes", [
  CounterService,
  Object
])
```

ITest -> Object, WAT?

`map[ITest] = SomeClass`

Angular2 templates

```
@Component({
  selector: 'app',
  template: `{{cnt}} <button (click)="addSome()">Add</button>`
})
export class CounterView {
  counter: number = 0
  add(){
    this.counter += 1
  }
}
```

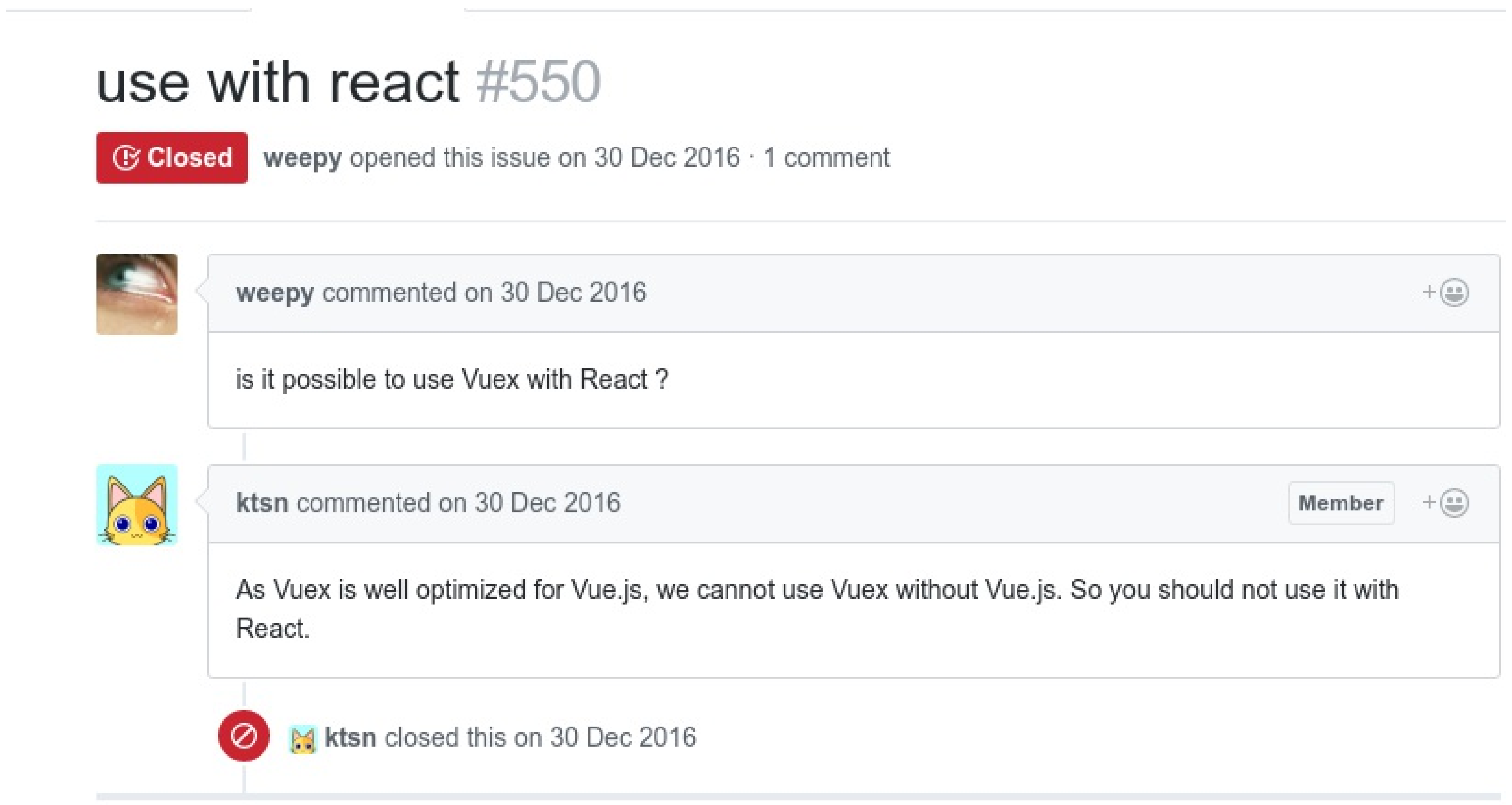
- Типы в шаблонах
- typescript проигнорирует addSome

Vue

```
var app5 = new Vue({
  el: '#app-5',
  data: {
    message: 'Hello Vue.js!'
  },
  mixins: [myMixin],
  methods: {
    reverseMessage: function () {
      this.message = this.message.split('').reverse().join('')
    }
  }
})
```

- К React.createClass, опять?
- fuck the flow
- Быть всем
- Быть всем в монолите

vuex - vue only


use with react #550

Closed weepy opened this issue on 30 Dec 2016 · 1 comment

weepy commented on 30 Dec 2016

is it possible to use Vuex with React ?

ktsn commented on 30 Dec 2016 Member

As Vuex is well optimized for Vue.js, we cannot use Vuex without Vue.js. So you should not use it with React.

ktsn closed this on 30 Dec 2016

- react-router
- react-router-redux
- mobx-react-router
- inferno-router
- vue-router
- vuex-router-sync

```
function CaseComponent({history}) {

  return <Router history={history}>
    <Route path="/" component={App}>
      <Route path="foo" component={Foo}/>
      <Route path="bar" component={Bar}/>
    </Route>
  </Router>
}
```

- ReactRouter, ReactSideEffect, ReactHelmet
- Контроллер
- Смешение слоев

```
function CaseComponent({ path }) {
  switch (path) {
    case '/': return App
    case 'foo': return Foo
    default: return App
  }
}
```

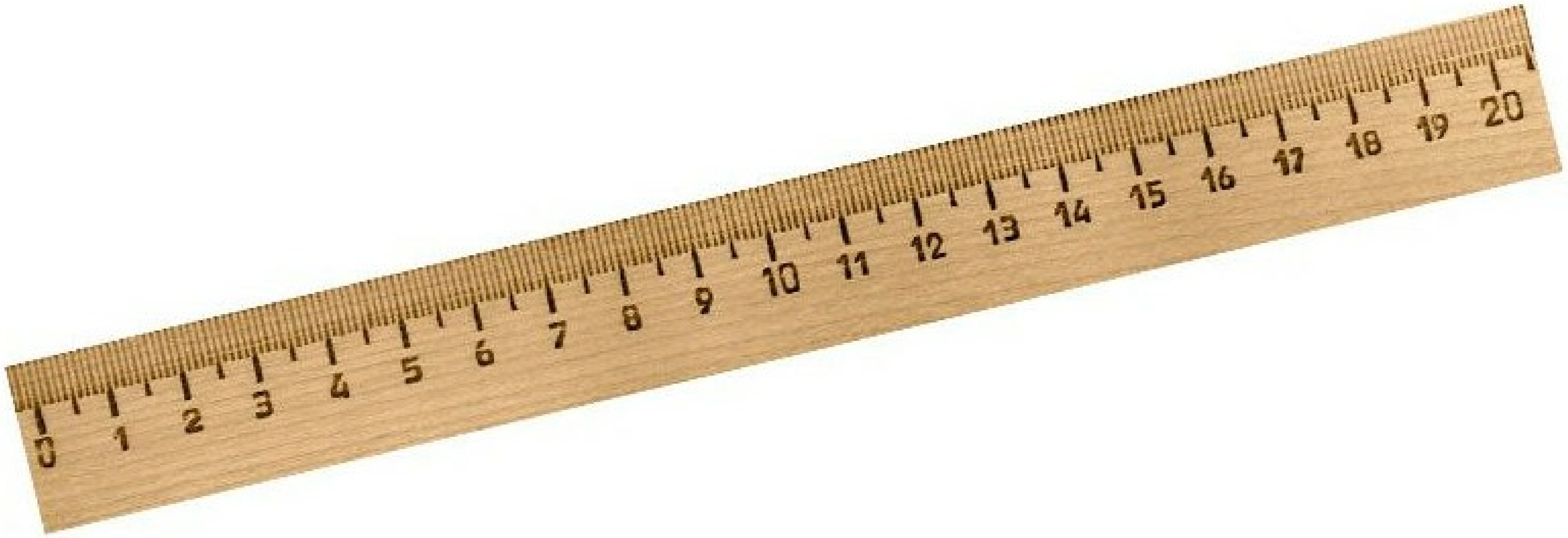
```
class Router {
  @observable path = ''
}
const router = new Router()
location.onChange((path: string) => {
  router.path = path
})
```

Vendor lock-in everywhere

Конкуренция

- Типовой код (angular - 15K, inferno - 5K)
- Монолитный код
- Подсадить на фреймворк
- Одиночки в худшем положении

Оптимизация фреймворков = хайп



- Хайп 5 > 3
- ~~Связанность, сцепленность~~
- react fiber, vdom, prepack, inferno
- Не имеет отношения к решению

Оптимизации в приложении = костыли

```
class CounterView extends React.Component {
  state = {count: 0}

  shouldComponentUpdate(nextProps, nextState) {
    return nextState.count === this.state.count
  }

  _add = () => this.setState({ count: this.state.count++ })

  render() {
    return <div>{this.props.name}: {this.state.count}
      <button onClick={this._add}>Add</button>
    </div>
  }
}
```

Angular

```
@Component({
  selector: 'app',
  changeDetection: ChangeDetectionStrategy.OnPush,
  template: `{{counter}} <button (click)="add()">Add</button>`
})
export class CounterView {
  public counter : number = 0;
  constructor(private cd: ChangeDetectorRef) {}

  add() {
    this.counter += 1
    this.cd.markForCheck()
  }
}
```

- Event -> viewRef.detectChanges
- Minesweeper
- OnPush = shouldComponentUpdate

Mobx

- cellx, derivablejs, glimmer, mol
- Обратился к свойству - подписался
- Ранняя точная оптимизация без VDOM

```
const CounterView = observer(store => <div>{store.count}</div>)

const AppView = observer(store => <div>
  <CounterView count={store}/>
</div>)

class Store {
  @observable count: number = 0
}

const store = new Store()
React.render(<AppView store={store} />, document.body)

store.count = 1 // rerender
```

```
const CounterView = /*observer*/(store => <div>{store.count}</div>)

const AppView = /*observer*/(store => <div>
  <CounterView count={store}/>
</div>)

class Store {
  /*@observable*/ count: number = 0
}

const store = new Store()
React.render(<AppView store={store} />, document.body)

store.count = 1 // rerender
```

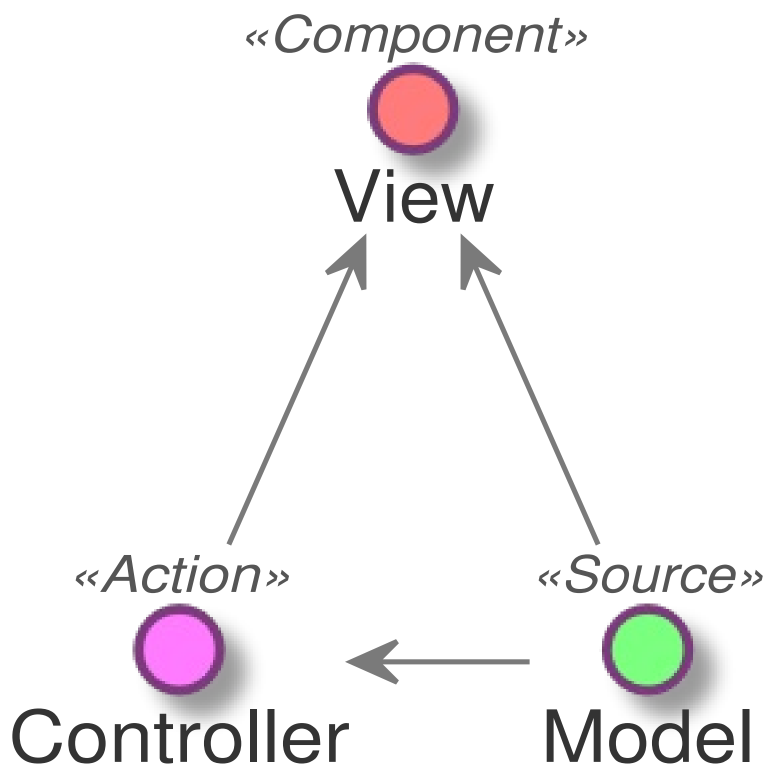
```
class Counter { count = 0 }

function Hello(
    // public
    {text}: { text: string; },

    // private
    {counter}: { counter: Counter; }
) {
    return <div>
        <h1>{text} {counter.count}</h1>
    </div>
}
```

Reactive-di view

```
function Counter() { this.count = 0 }

function Hello(_ref, _ref2, _t) {
    var text = _ref.text;
    var counter = _ref2.counter;

    return _t.h(2, 'div', null, [
      _t.h(2, 'h1', null, ['count ', counter.count])
    ]);
}

Hello._isComponent = true;
Hello._dependencies = [{ counter: Counter }];
```

context = DI + metadata

- 15й стандарт
- Совместим с 14м (React)
- Поддерживается в flow
- Работает legacy
- Interoperability
- Ъ-Чистые
- ~~Smart, dumb~~



- React - View
- Mobx - Model
- Reactive-di - Окружение, все внутри стримов

!!!

- Экосистема вокруг типов
- Слои: data - ui - business logic
- Ненавязчивость (mobx)
- KISS
- КПД: 3-4 (angular - 15K, inferno - 5K)

- github.com/zerkalica/reactive-di
- medium.com/@sergey_yuferev
- nexor@ya.ru