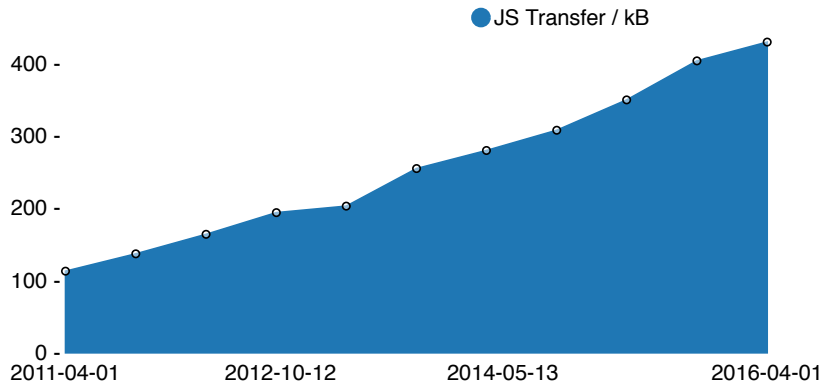
BEYOND FLUX

SCALABLE FRONTEND ARCHITECTURES USING PUBLISH/SUBSCRIBE

Michael Kurze @ goto Amsterdam 2016

WHY ARCHITECTURE?

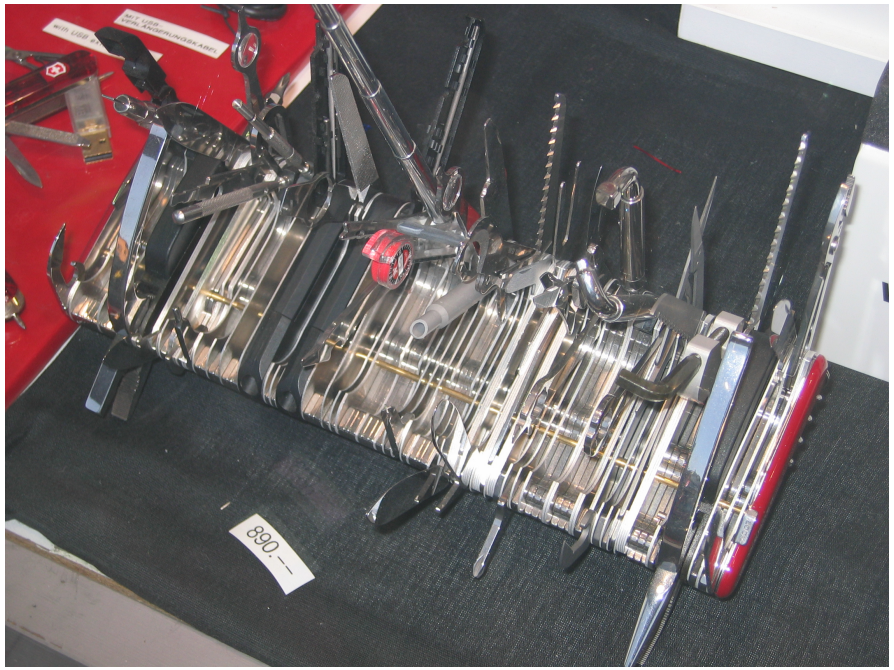
IT'S ABOUT TIME!



(Source: httparchive.org)

- We are *building* complex software *(requires tools)*,
- and we are *talking* about them *(requires a language)*.

COMPLEXITY



PLATFORM COMPLEXITY

DRIVING

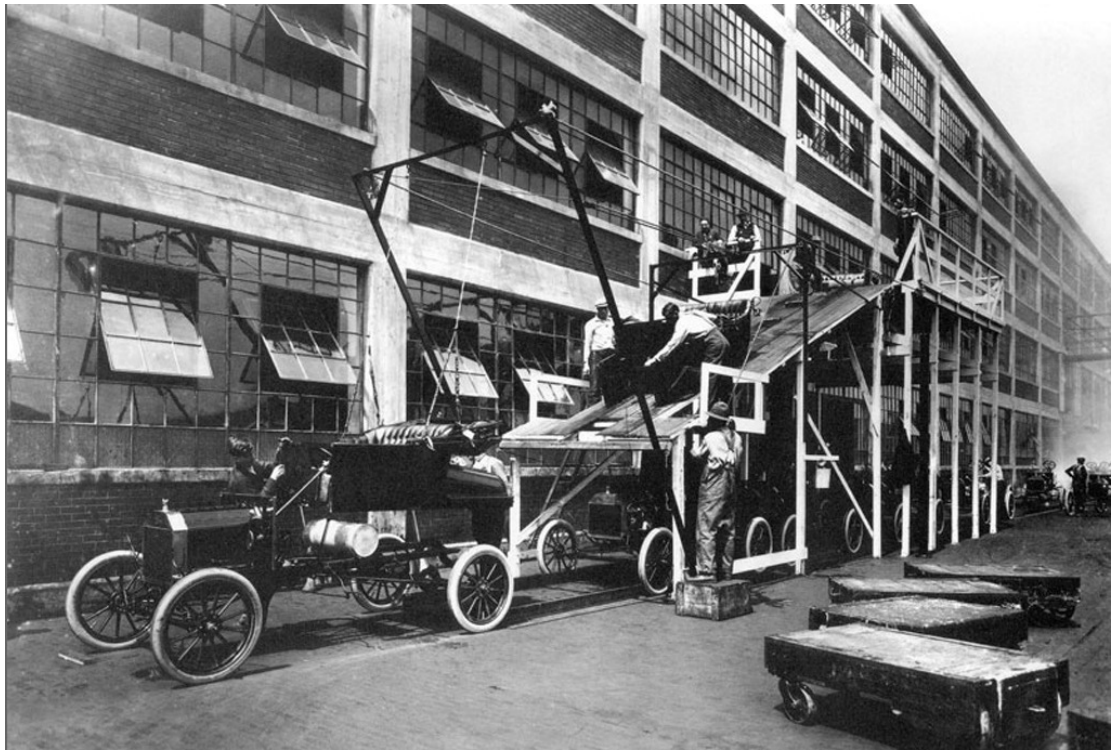
APPLICATION COMPLEXITY

REACTIVITY & RESPONSIVENESS

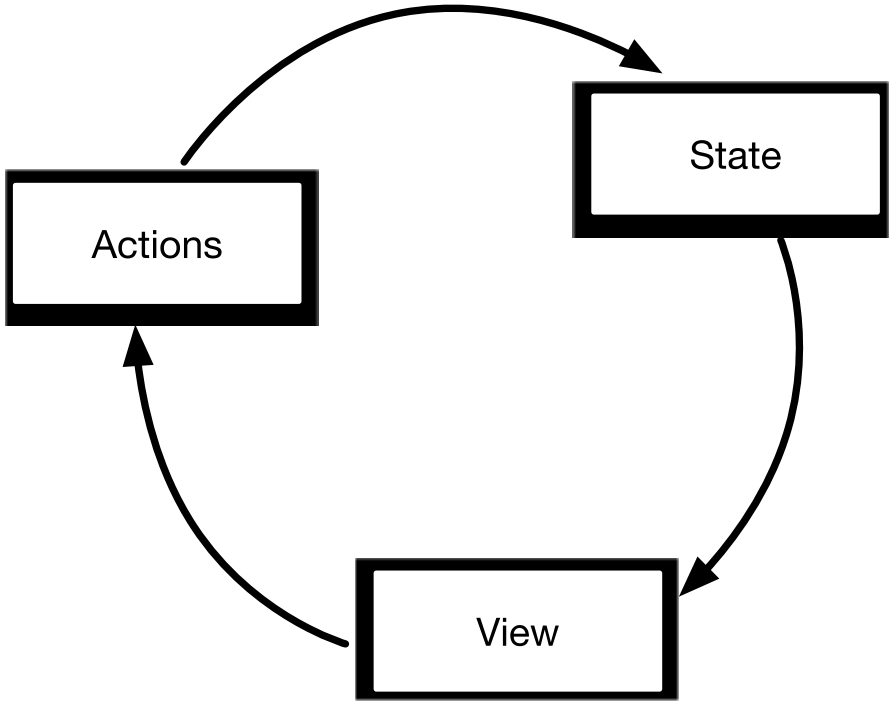
- synchronize state (across devices)
- provide immediate feedback
- continuous bi-directional communication (client/server)

http://www.reactivemanifesto.org

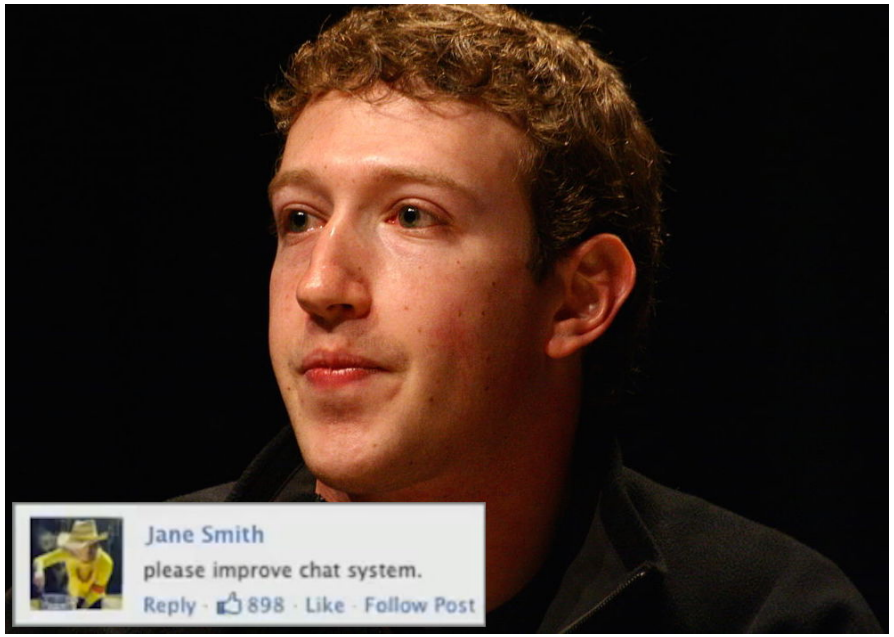
SCALEABILITY



UNIDIRECTIONAL FLOW

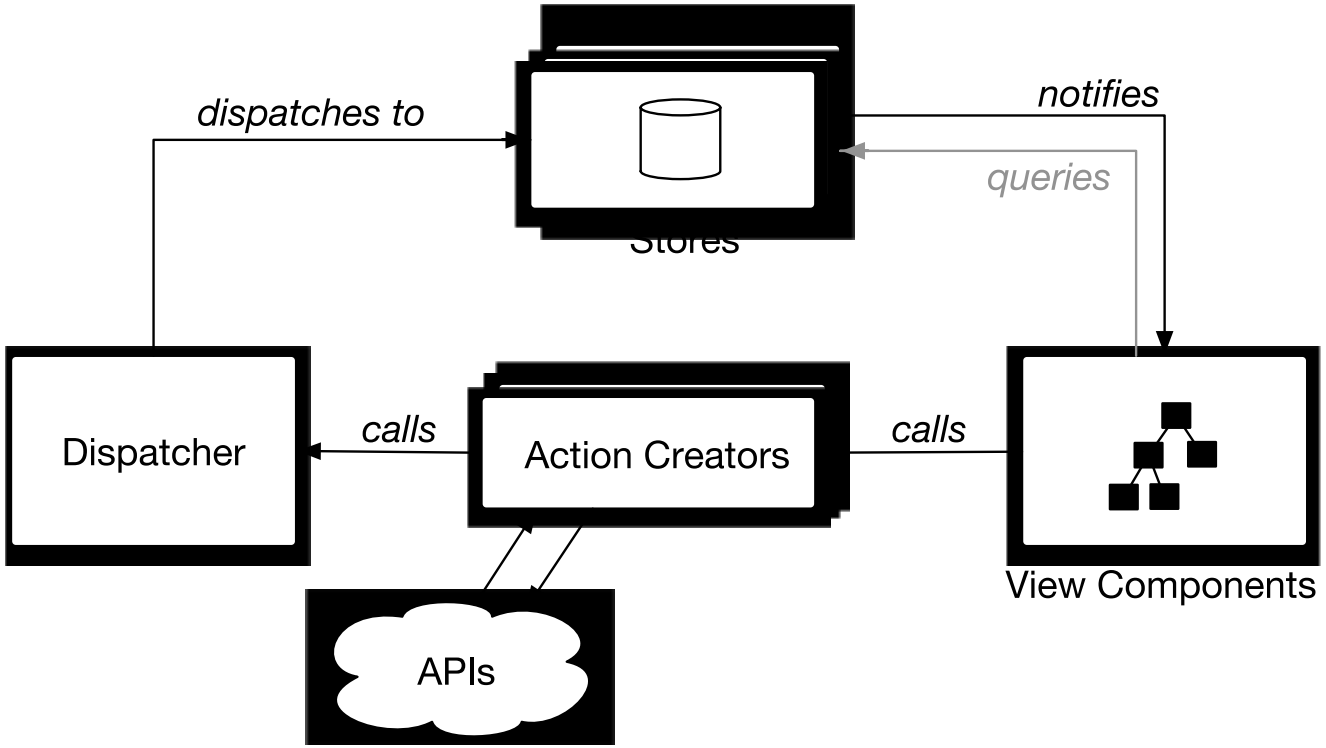


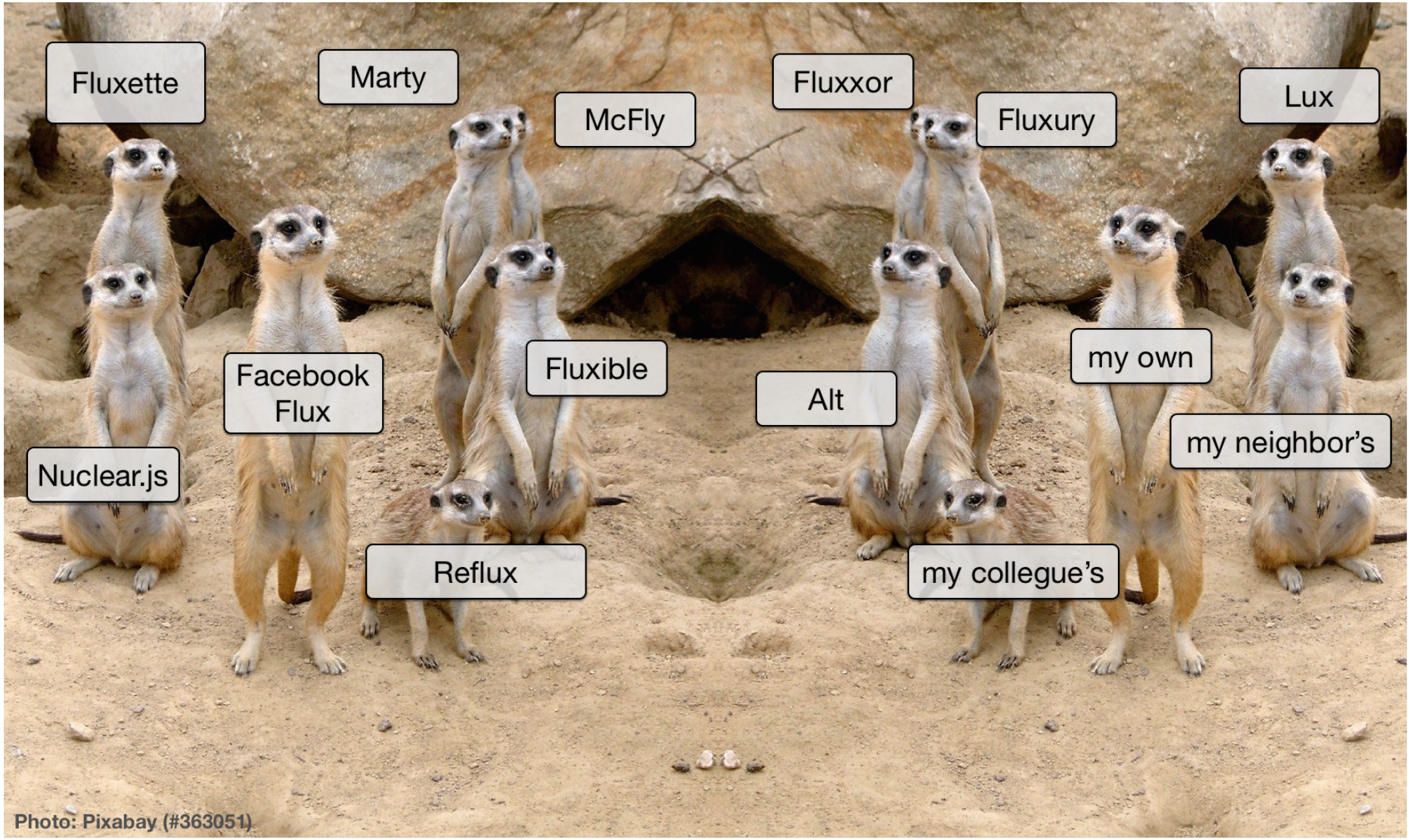
ONCE UPON A TIME ...



- UI inconsistencies in Facebook chat

FLUX


dispatches to
notifies
queries
Stores
Dispatcher
calls
Action Creators
calls
View Components
APIs


Fluxette
Marty
McFly
Fluxxor
Fluxury
Lux
Facebook Flux
Fluxible
Alt
my own
Nuclear.js
my neighbor's
Reflux
my collegue's
Photo: Pixabay (#363051)

FLUX - EXAMPLE

Flux Shop Demo

iPad 4 Mini - €500.01

Add to cart

H&M T-Shirt White - €10.99

Add to cart

Charli XCX - Sucker CD - €19.99

Add to cart

Your Cart

iPad 4 Mini - €500.01 x 1
H&M T-Shirt White - €10.99 x 3
Total: €532.98

Checkout

- https://github.com/voronianski/flux-comparison

FLUX - EXAMPLE - VIEW (1/3)

```
var React = require('react');
var FluxibleMixin = require('fluxible').FluxibleMixin;
var ProductStore = require('./stores/ProductStore');
var addToCart = require('./actions/addToCart');

var Products = React.createClass({
    mixins: [FluxibleMixin],
    render: function() {
        return <ul>
            {this.state.products.map(product =>
                <li>
                    <img src={product.image}/>
                    {product.title} - {product.price}€
                    <button onClick={() => this.addToCart(product)}
                            disabled={product.inventory === 0}>add</button>
                </li>
            )}
        </ul>;
    },

    addToCart: function(product) {
        this.executeAction(addToCart, { product: product });
    },

    // ...
});
```

FLUX - EXAMPLE - VIEW (2/3)

```
var React = require('react');
var FluxibleMixin = require('fluxible').FluxibleMixin;
var ProductStore = require('./stores/ProductStore');
var addToCart = require('./actions/addToCart');

var Products = React.createClass({

    // ...render, addToCart...

    getInitialState: function() {
        return this._getStateFromStores();
    },

    _getStateFromStores: function() {
        return {
           products: this.getStore(ProductStore).getAllProducts()
        };
    },

    statics: {
        storeListeners: {
            _onChange: [ProductStore]
        }
    },

    _onChange: function() {
        this.setState(this._getStateFromStores());
    }
});
```

FLUX - EXAMPLE - VIEW (3/3)

```
var React = require('react');
var FluxibleMixin = require('fluxible').FluxibleMixin;
var ProductStore = require('./stores/ProductStore');
var addToCart = require('./actions/addToCart');

var Products = React.createClass({ /* ... */ });
var ShoppingCart = require('./components/CartContainer.jsx');

var App = React.createClass({
    mixins: [FluxibleMixin],
    render: function() {
        return <div>
                    <Products />
                    <ShoppingCart />
                </div>;
    }
});

export function render(context) {
    React.withContext(
        context.getComponentContext(),
        function() {
            React.render(
                React.createElement(App),
                document.getElementById('fluxible-app')
            );
        }
    );
}
```

FLUX - EXAMPLE - ACTION

```
var shop = require('../../../api/shop');

// actions/addToCart.js
module.exports = function (context, payload, done) {
    context.dispatch('ADD_TO_CART', { product: payload.product });
    done();
};

// actions/cartCheckout.js
module.exports = function (context, payload, done) {
    var products = payload.products;

    context.dispatch('CART_CHECKOUT');

    shop.buyProducts(products, function () {
        context.dispatch('SUCCESS_CHECKOUT', { products: products });
        done();
    });
};
```

FLUX - EXAMPLE - DISPATCHER

```
var Fluxible = require('fluxible');
var CartStore = require('./stores/CartStore');
var ProductStore = require('./stores/ProductStore');

var app = new Fluxible();
app.registerStore(CartStore);
app.registerStore(ProductStore);


var view = require('./view')
var receiveProducts = require('./actions/receiveProducts');
var context = app.createContext();
context.executeAction(receiveProducts, {}, function (err) {
    if (err) {
        throw err;
    }
    return view.render(context);
});
```

FLUX - EXAMPLE - STORES

```
var createStore = require('fluxible/addons/createStore');
var ProductStore = createStore({
    storeName: 'ProductStore',
    initialize: function () {
        this._products = [];
    },

    handlers: {
        'ADD_TO_CART': 'decreaseInventory',
        'RECEIVE_PRODUCTS': 'handleReceive'
    },

    handleReceive: function (payload) {
        this._products = payload.products;
        this.emitChange();
    },

    decreaseInventory: function (payload) {
        this.dispatcher.waitFor('CartStore', function() {
            var product = payload.product;
            product.inventory = Math.max(product.inventory-1, 0);
            this.emitChange();
        });
    },

    getAllProducts: function () { return this._products; }
});
module.exports = ProductStore;
```

ADVANTAGES OF FLUX

compared to "classic" MVC (AngularJS, Backbone)

- strict separation: State / UI / Behavior
- each value stored exactly once
- clear data flow, good testability
- server-side rendering relatively simple

DISADVANTAGES OF FLUX

compared to "classic" MVC (AngularJS, Backbone)

- each store *sees all actions*
- hard-wired *store-dependencies*
- stores are (conceptually) *singletons*
- *anti-pattern:* transporting *mutable state* in actions

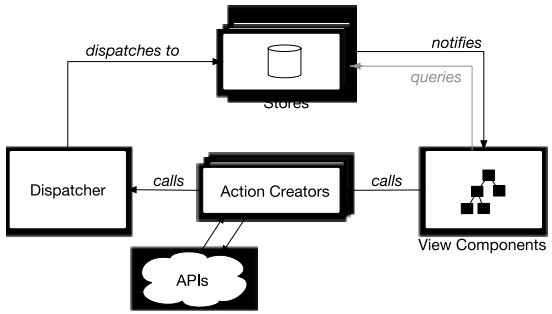
REDUX

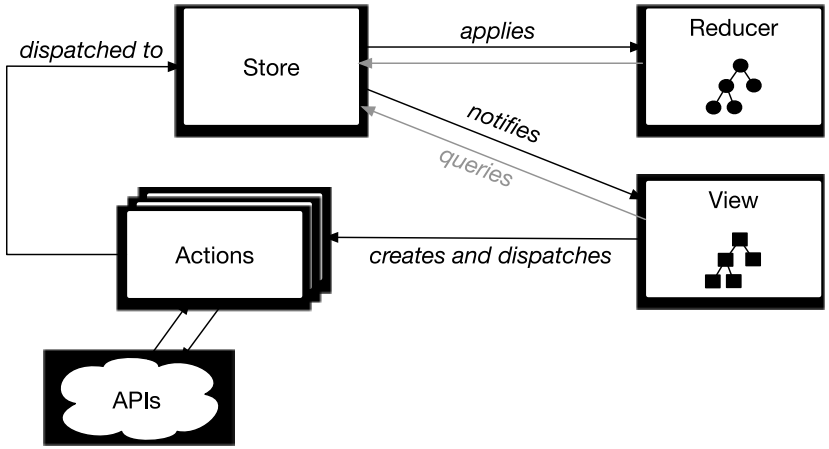
THE EVOLUTION

(STATE, ACTION) → STATE

REDUX

Flux:


dispatches to
notifies
queries
Stores
Dispatcher
calls
Action Creators
calls
View Components
APIs


dispatched to
Store
applies
Reducer
notifies
queries
View
Actions
creates and dispatches
APIs

ADVANTAGES OF REDUX

vs. "vanilla" Flux

- functional composition
- snapshot/replay capabilities
- can yield performance boost (free dirty-checking)

DISADVANTAGES OF REDUX

vs. "vanilla" Flux

- must protect against mutating state, using e.g.
  - spread constructor `{...oldState, prop: newVal}` (ES201?)
  - (test with) recursive `Object.freeze()`
  - Immutable.js
- potentially steep learning curve (functional paradigm)
- container components know the whole state tree

THE REVOLUTION

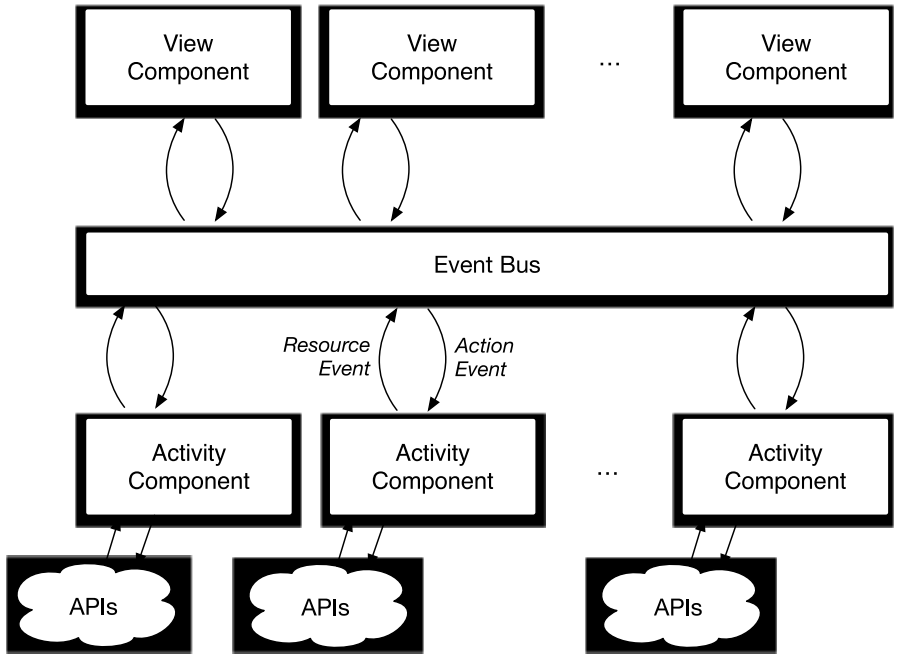
PUBLISH-SUBSCRIBE

an old pattern from the 80ies

DECOUPLING

- full *decoupling* of communicating components:
  - sender does not know about recipient(s)
  - recipient does not know about sender
- How? Broadcast of events through central *bus*
- selection of receivers through *topics*

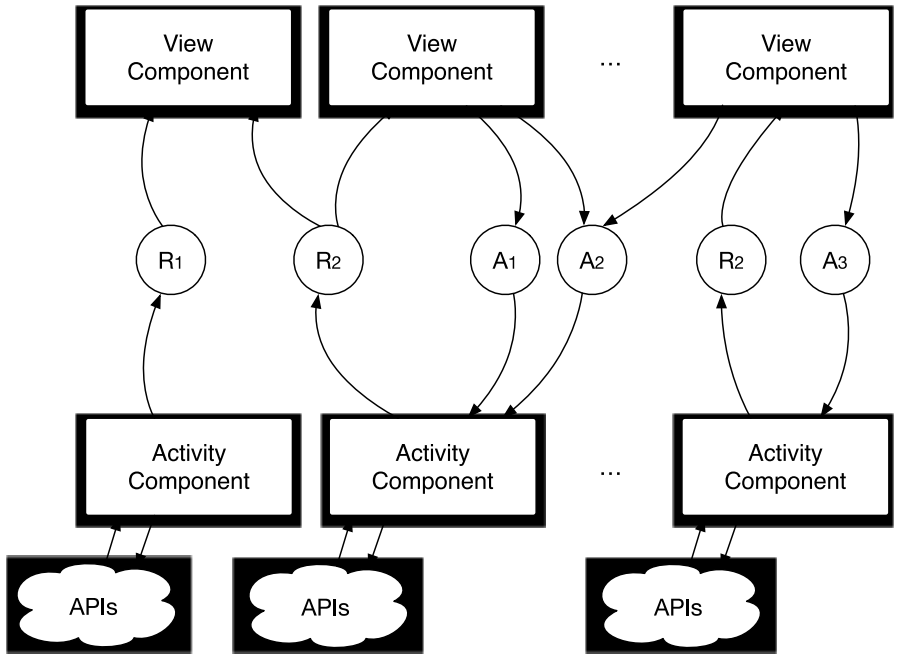
PUBSUB ARCHITECTURE



- event bus: connects senders/receivers (asynchronously)
- *view components,* like in Flux/Redux (+ create action events)
- *activity components,* like Flux stores (+ completion of async actions)
- unidirectional flow using event patterns

PUBSUB ARCHITECTURE

TOPICS FOR SCALABILITY



- topics for *resources* (state slices) and *actions* (intents to modify)
- topics connect components in a *configurable* way

PUBSUB (LAXARJS) - EXAMPLE - VIEW COMPONENT (1/2)

```
import React from 'react';
import patterns from 'laxar-patterns';

const injections = ['axContext', 'axReactRender'];
function create(context, reactRender) {

    patterns.resources.handlerFor(context)
        .registerResourceFromFeature('products', render);

    function render() {
        reactRender(<ul>
            {(context.resources.products || []).map(product =>
                <li>
                    <img src={product.image}/>
                    {product.title} - {product.price}€
                    <button onClick={() => addToCart(product)}
                            disabled={product.inventory === 0}>add</button>
                </li>
            )}
        </ul>);
    }

    // ... addToCart ...
```

PUBSUB (LAXARJS) - EXAMPLE - VIEW COMPONENT (2/2)

```
import React from 'react';
import patterns from 'laxar-patterns';

const injections = ['axContext', 'axReactRender'];
function create(context, reactRender) {

    // ... render ...

    const addToCartPublisher =
        patterns.actions.publisherForFeature(context, 'addToCart')

    function addToCart(product) {
        if(product.inventory > 0) {
            addToCartPublisher({ product: product });
        }
    }

    return { onDomAvailable: render };
}

export default {
    name: 'product-list-view',
    injections: injections,
    create: create
};
```

PUBSUB (LAXARJS) - EXAMPLE - EVENTS

```
{
     "widget": "product-list-view",
     "features": {
         "addToCart": { "action": "add-to-cart" },
         "products": { "resource": "product-list" }
     }
 },
 {
     "widget": "shopping-cart-view",
     "features": { ... }
 },
 {
     "widget": "products-activity",
     "features": {
         "products": { "resource": "product-list" },
         "decrementInventory": { "onActions": ["add-to-cart"] }
     }
 },
 {
     "widget": "shopping-cart-activity",
     "features": { ... }
 }
```

PUBSUB (LAXARJS) - EXAMPLE - ACTIVITY COMPONENT (1/2)

```
import ax from 'laxar';
import patterns from 'laxar-patterns';
import shop from '../../../api';

const injections = ['axContext'];
function create(context) {
    let products = [];
    const replaceProducts =
        patterns.resources.replacePublisherForFeature(context, 'products');

    context.eventBus.subscribe('beginLifecycleRequest', () => {
        shop.getProducts(list => {
            products = list;
            replaceProducts(products);
        });
    });

    // ... action handling (next) ...
}

export default {
    name: 'products-activity',
    injections: injections,
    create: create
};
```

PUBSUB (LAXARJS) - EXAMPLE - ACTIVITY COMPONENT (2/2)

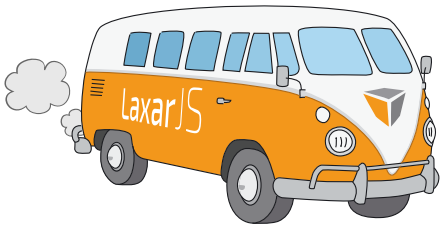
```
import ax from 'laxar';
import patterns from 'laxar-patterns';
import shop from '../../../api';

const injections = ['axContext'];
function create(context) {
    // ... products initialization (previous) ...

    patterns.actions.handlerFor(context)
        .registerActionsFromFeature('decrementInventory', ({ product }) => {
            products
                .filter(current => current.id == product.id)
                .forEach(current => {
                    current.inventory = Math.max(0, current.inventory - 1);
                });
            replaceProducts(products);
        });
}

export default {
    name: 'products-activity',
    injections: injections,
    create: create
};
```

PUBSUB IN PRACTICE



FROM EVENT BUS TO FRAMEWORK

http://www.laxarjs.org

- component configuration
- lifecycle: *when are event collaborators available?*
- services for view components: *where do I render my HTML to?*
- development tools (event log, visual composition structure)

ADVANTAGES OF PUB/SUB

vs. Flux/Redux

- *decoupling* of components
- *visible* architecture
- *flexibility* of composition *(patterns: resources, actions, flags, streams...)*
- *freedom* of implementation *(functional vs. imperative, React vs. AngularJS...)*

DISADVANTAGES OF PUB/SUB

vs. Flux/Redux

- *ecosystem* small
- *freedom* to make mistakes
- server-side rendering *not there yet*

CONCLUSION

IS THIS EVEN FLUX?!

> *"I define Flux as 'unidirectional data flow with changes described as plain objects'."*
>
> *(Dan Abramov, Author of Redux)*

> *"It's cool that you are inventing a better Flux by not doing Flux at all."*
>
> *(André Staltz, Author of Cycle.js)*

CONCLUSION*

	Flux	Redux	PubSub
unidirectional flow	+	+	+
complexity handling	o	+	+
reactivity	o	+	+
scalability (component isolation)	-	o	+
ecosystem	o	+	-
learning curve	+	o	+
snapshot/record/replay	o	+	o
longevity	o	?	+

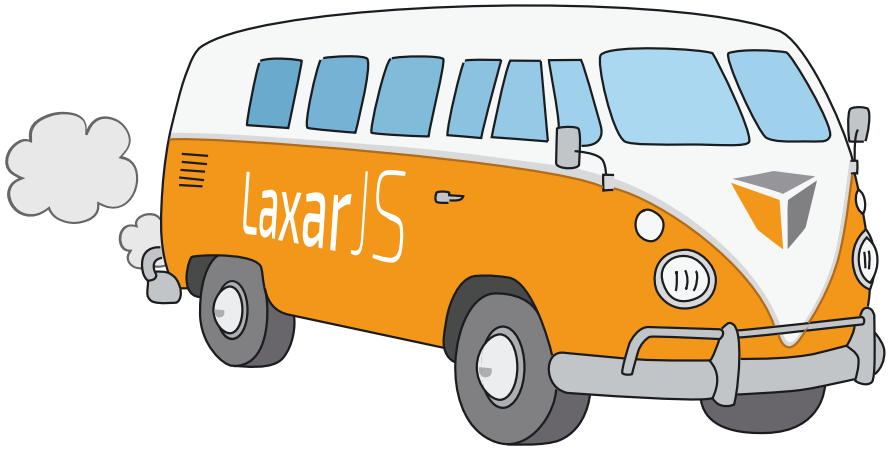
*** YOUR MILEAGE MAY VARY**

Redux  **PubSub**

BEYOND FLUX

Slides: github.com/x1b/beyond-flux-goto2016

Michael Kurze (@0b11011)



laxarjs.org

aixigo
einfach. besser. beraten.

IMAGES

- "The ultimate Swiss Army Knife for sale in Interlaken",
  https://www.flickr.com/photos/redjar/113974357,
  Lizenz: CC BY-SA 2.0, use without modification
- "1913 photograph Ford company, USA",
  https://en.wikipedia.org/wiki/File:A-line1913.jpg,
  Lizenz: Public Domain
- "Mark Zuckerberg Facebook SXSWi 2008 Keynote",
  https://www.flickr.com/photos/99565773@N00/2323729121,
  Lizenz: CC BY 2.0, use without modification
- "Erdmännchen, Zoo, Tier, Sand, Wüste", Pixabay / Didgeman
  https://pixabay.com/de/erdm%C3%A4nnchen-zoo-tier-sand-w%C3%BCste-363051,
  Lizenz: CC0 1.0 Public Domain
- "Magic Cube, Patience Games, Puzzle", Pixabay / Hans
  https://pixabay.com/en/magic-cube-patience-games-puzzle-232282,
  Lizenz: CC0 1.0 Public Domain
- "Two wedding rings lying on a white surface.",
  http://www.picserver.org/w/wedding-rings01.html,
  Lizenz: CC BY-SA 3.0, use without modification