Flux over the Wire

ReactJS Paris Meetup #1

Elie Rotenberg <elie@rotenberg.io>

About me – personal branding™

CTO Webedia Gaming

jeuxvideo.com | millenium.org | fr.ign.com

@elierotenberg 
elierotenberg 
blog.rotenberg.io 
elie@rotenberg.io 

Some repos I'm proud of:
nexus-flux | react-nexus | remutable | react-animate | es6-starterkit

App state

Global state: state updated by DOM/HTML APIs

Window size, mouse/scroll position, location URL, clocks…

```
React.createClass({
  render() {
    return <div>The current scrolling position is {/* ???? */} </div>;
  }
});
```

App state

Shared state: state shared by multiple components

User login, navigation state, non-local business logic

```
React.createClass({
  increaseCounter() {
    /* ??? */
  }
  render() {
    return <button onClick={this.increaseCounter}>Click to +1</button>
  }
});

React.createClass({
  render() {
    return <div>The number of clicks is {/* ???? */}</div>;
  }
})
```

App State in React – using globals

- Use global variables
- Re-render on change events

```
window.clickCounter = 0;
window.clickEvents = new EventEmitter();

React.createClass({
  increaseCounter() {
    window.clickCounter++;
    window.clickEvents.emit('change');
  }
  render() {
    return <button onClick={this.increaseCounter}>Click to +1</button>
  }
});

React.createClass({
  componentDidMount() {
    window.clickEvents.on('change', () => this.forceUpdate());
  }
  render() {
    return <div>The number of clicks is {window.clickCounter}</div>;
  }
});
```

App State in React – passing down props

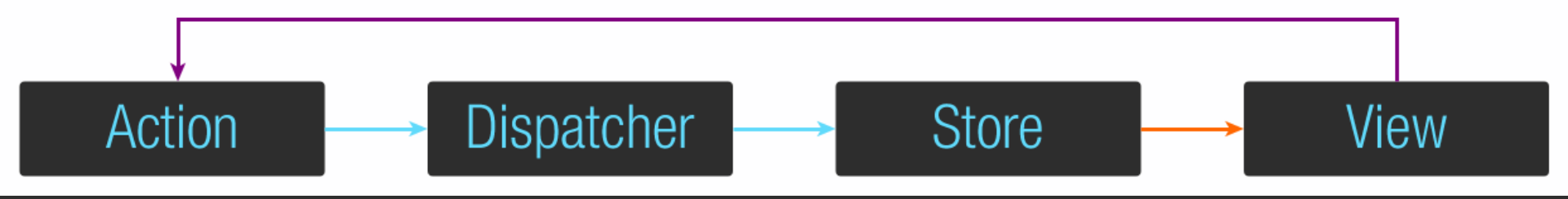
- Pass down getters and setters as props

```
const Clickbutton = React.createClass({
  render() {
    return <button onClick={this.props.increaseCounter}>Click to +1</button>
  }
});

const ClickLabel = React.createClass({
  render() {
    return <div>The number of clicks is {this.props.counter}</div>;
  }
});

React.createClass({
  getInitialState() {
    return { clicks: 0 };
  }
  increaseCounter() {
    this.setstate({ clicks: this.state.clicks + 1 });
  }
  render() {
    return <div>
      <ClickButton increaseCounter={this.increaseCounter} />
      <ClickLabel counter={this.state.clicks}
    </div>;
  }
});
```

App State Management with Flux



- Implicit global
- One-way data flow
- Stores represent read-only values that might be updated
- Actions represent fire-and-forget intents

- Re-render whenever a depended Store is updated

App State Management with Flux

```
const Flux = {
  actions: {
    increaseCounter() {
      Flux.stores.counter._value++;
    }
  },
  stores: {
    counter: {
      _value: 0,
      getValue() {
        return this._value;
      }
    }
  }
}

const Clickbutton = React.createClass({
  render() {
    return <button onClick={() => Flux.actions.increaseCounter()}>Click to +1</button>
  }
});

const ClickLabel = React.createClass({
  componentWillMount() {
    Flux.stores.counter.onChange(this.forceUpdate);
  }
  render() {
    return <div>The number of clicks is {Flux.stores.counter.getValue()}</div>;
  }
});
```

App State Management with Flux

Goal:

- share state between multiple components

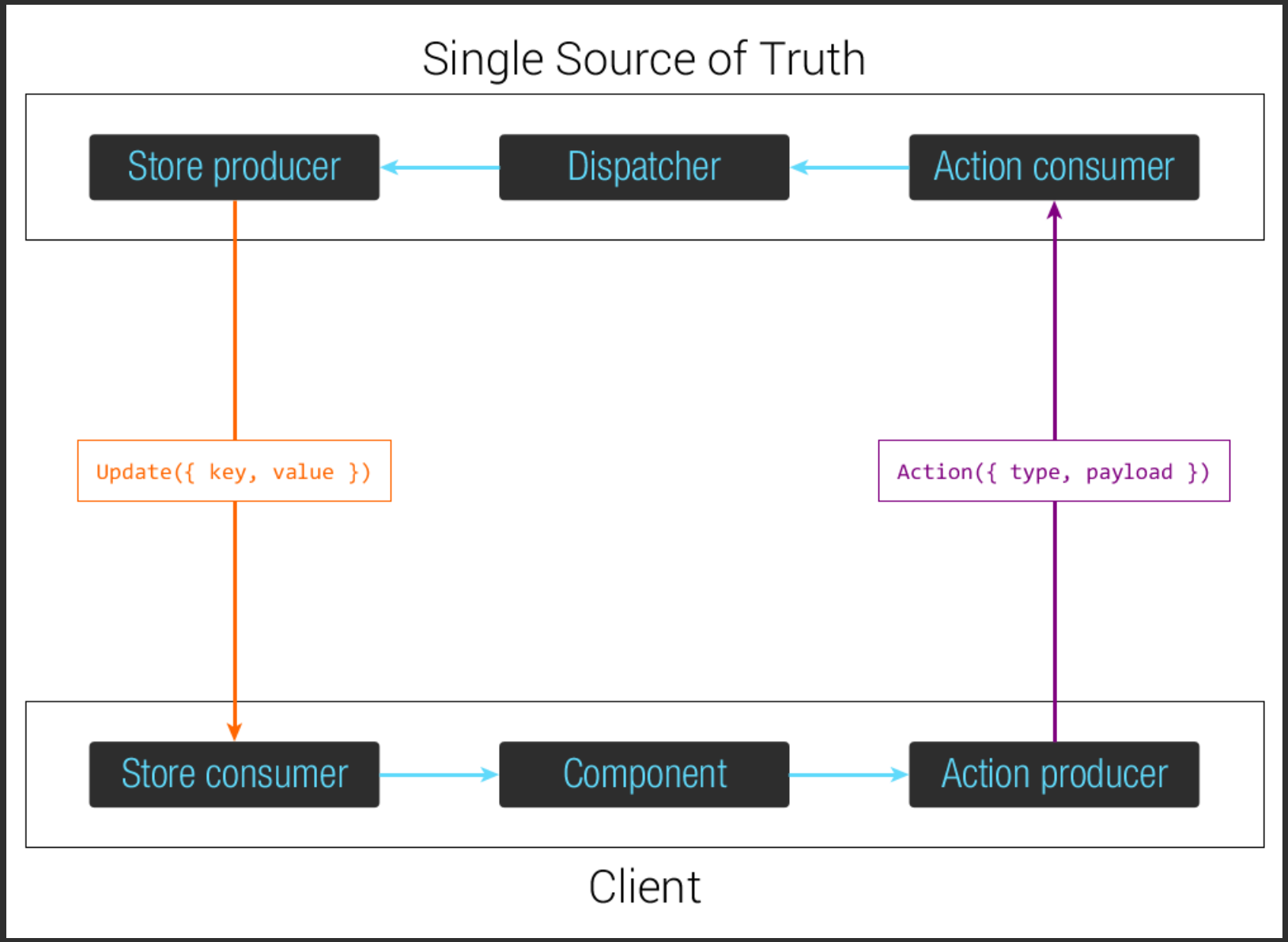
Problems:

- actually expose the state
- avoid shared mutable state
- re-render when and only when required

Solution: Flux

- expose state to all components
- encapsulate state in a one-way flow
- stores are read-only observers
- actions are fire & forget intents

Nexus Flux


Single Source of Truth
Store producer
Dispatcher
Action consumer
Update({ key, value })
Action({ type, payload })
Store consumer
Component
Action producer
Client

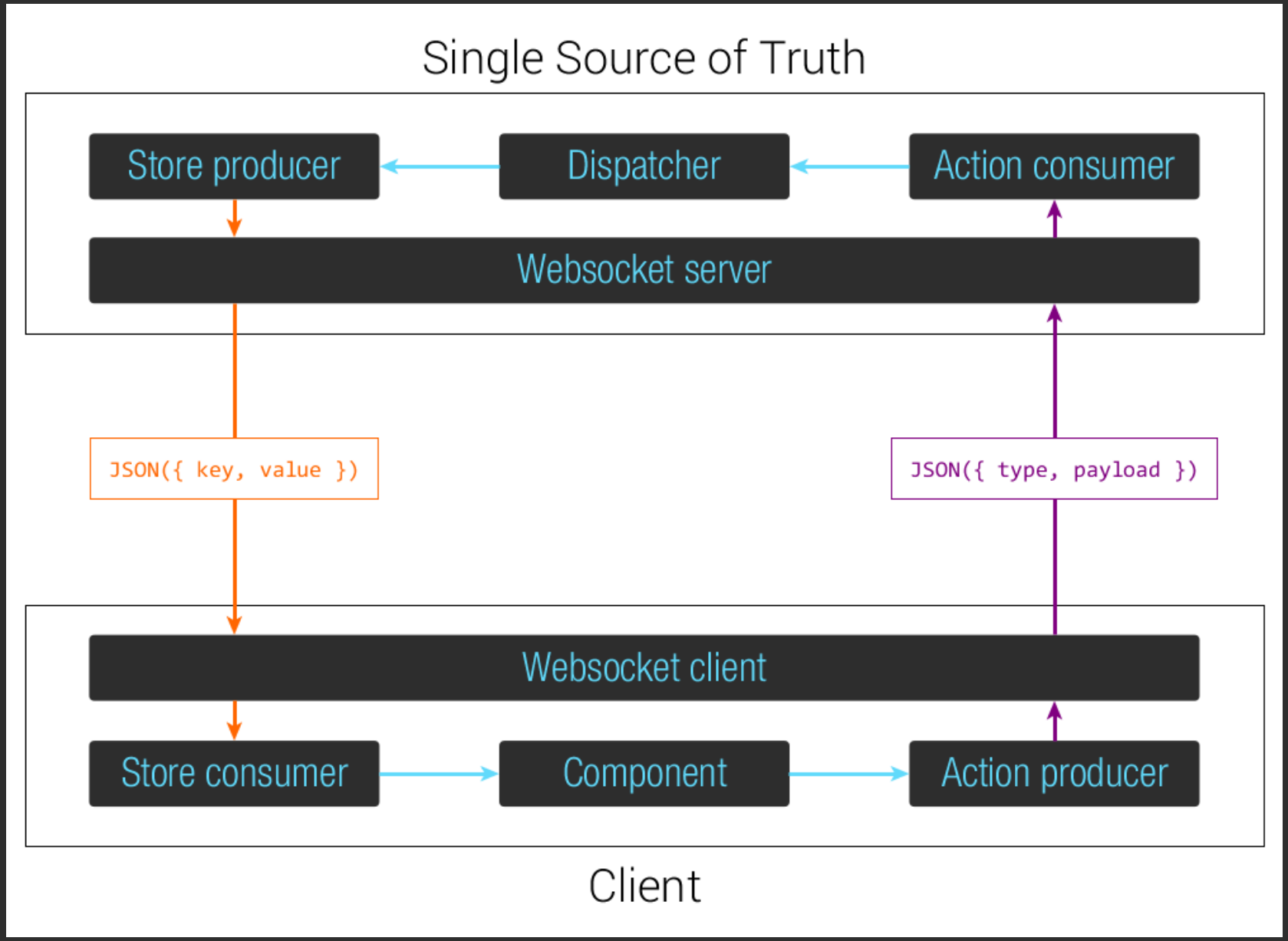
Nexus Flux

```
const Flux = new Nexus.Server(); // Single Source of Truth
flux.registerStore('clicks', { value: 0 });
flux.registerAction('increaseCounter', () => flux.setStore('clicks', { value: clicks.value + 1 }));

const Clickbutton = React.createClass({
  mixins: [Nexus.Mixin(flux)],
  render() {
    // dispatch action on click
    return <button onClick={() => this.dispatch('increaseCounter')}>
      Click to +1
    </button>
  }
});

const ClickLabel = React.createClass({
  mixins: [Nexus.Mixin(flux)],
  getFluxBindings() {
    return ['counter']; // one-way bind from Flux store to this.state
  }
  render() {
    // counter is injected to state and auto-updated
    return <div>The number of clicks is {this.state.counter}</div>;
  }
});
```

Flux over the Wire

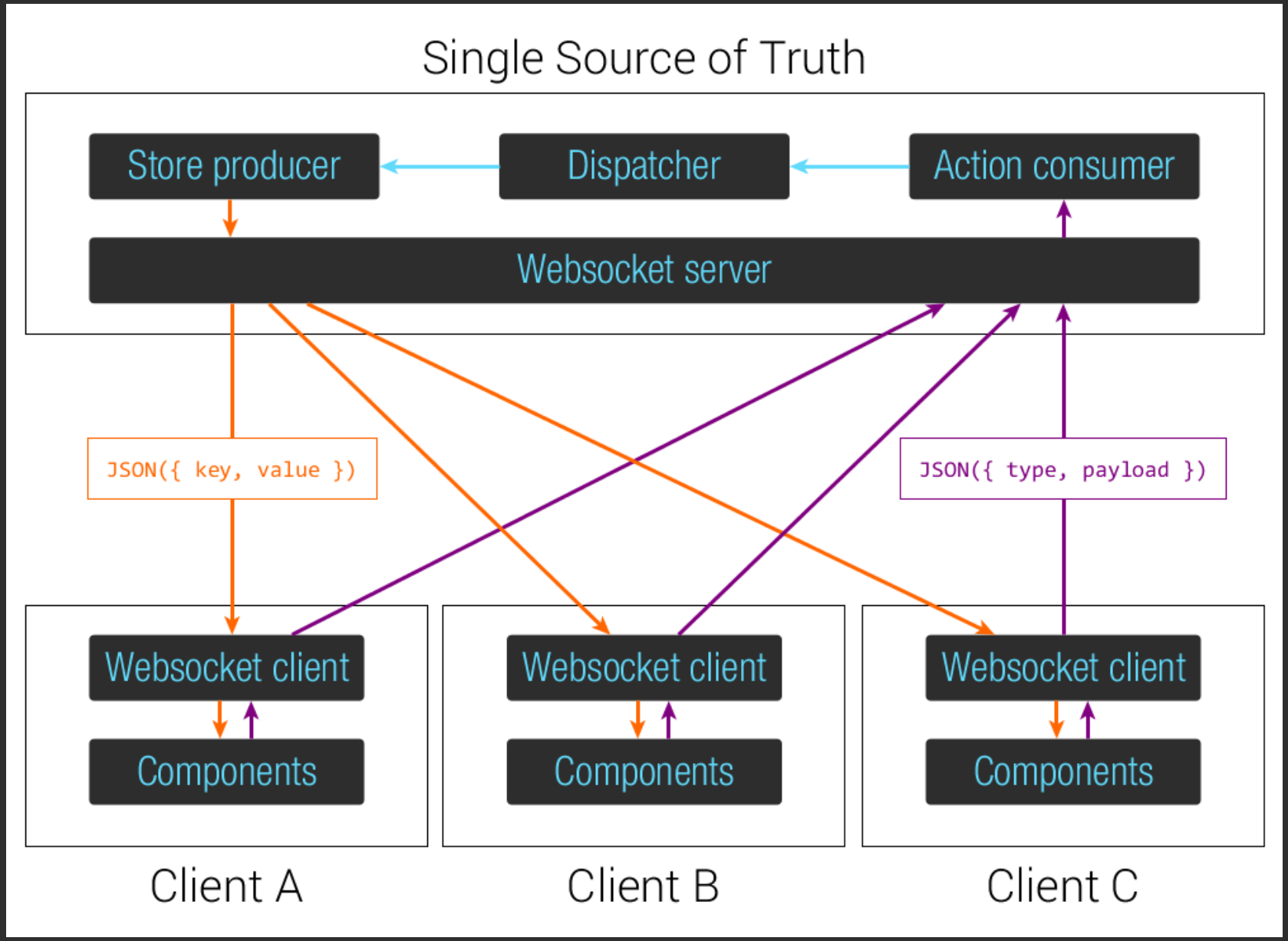


Flux over the Wire

- Implements the Nexus Flux API
- The Single Source of Truth is in the datacenter
- The Client is in the browser

- Actions are encoded in C2S Websocket frames/POST requests
- Stores are updated in S2C Websocket frames/exposed as GET

Flux over the Wire



Flux over the Wire

- Shared state between the server and the client…
- And every other connected client.

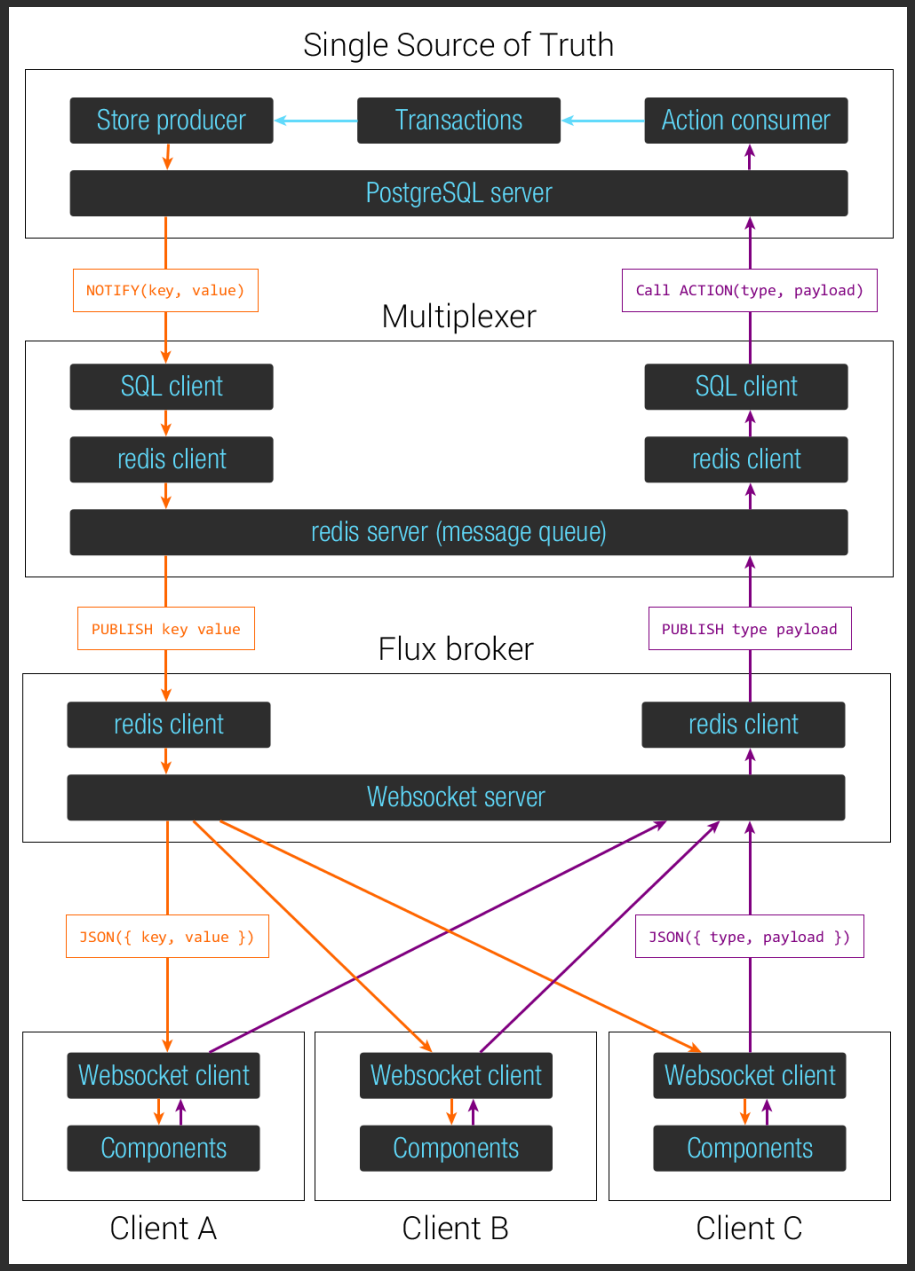
- Enables real-time first app design
- Gracefully degrade to non-real-time

remutable

- Record & serialize mutations on the server
- Transmis the update diff over the wire
- Unserialize & apply mutations on the client

```
single source of truth (server)        consumer (client)
v1 = Immutable.Map()
send(serialize(v1))        ----------> v1 = unserialize(receive())
(v2, diff1) = v1.set(...)
send(diff1)                ----------> v2 = v1.patch(unserialize(receive()))
(v3, diff2) = 2.set(...)
send(diff2)                ----------> v3 = v2.patch(unserialize(receive()))
```

Million user webchat with Nexus Flux



Server side rendering with React Nexus

```
import App from './components/App';

if(__CLIENT__) {
  Nexus.mountApp(<App nexus={App.createNexus({ window })} >);
}
if(__SERVER__) {
  express().get('*', (req, res) => Nexus.prerenderApp(<App nexus={App.createNexus({ req })} >, (err, [html]) => {
      if(err) { res.status(500).json(err);  }
      else    { res.status(200).send(html); }
    })).listen(80);
}
```

Demo!

Conclusion

- React = ❤
- React + Flux = ❤❤
- React + Flux + Nexus = ❤❤❤
- React + Flux + Nexus + Flux over the Wire = ❤❤❤❤
- React + Flux + Nexus + Flux over the Wire + React Nexus = ❤❤❤❤❤

About me – personal branding™

CTO Webedia Gaming

jeuxvideo.com | millenium.org | fr.ign.com

@elierotenberg 
elierotenberg 
blog.rotenberg.io 
elie@rotenberg.io 

Some repos I'm proud of:
nexus-flux | react-nexus | remutable | react-animate | es6-starterkit