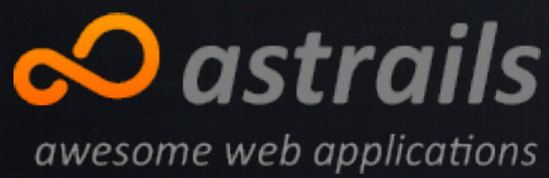
Migrating from Flux to Redux

why and how



Boris Nadion
boris@astrails.com
@borisnadion

astrails

awesome web apps since 2005

flux? redux?

Apr-2015

backbone and marionette

React - breath of fresh air

best thing since jQuery

spaghetti -> MVC

V = React

M+C = ?

obviously Flux

```
npm install flux --save
```

hold on!

is there anything better?

hell yeah!

lummox

Alt

Flexor

Flux This

MartyJS

McFly

Flexible

Delores

Lux

Reflux

OmniscientJS

Fluffy

Material Flux

...

each one of them was better

...then any other

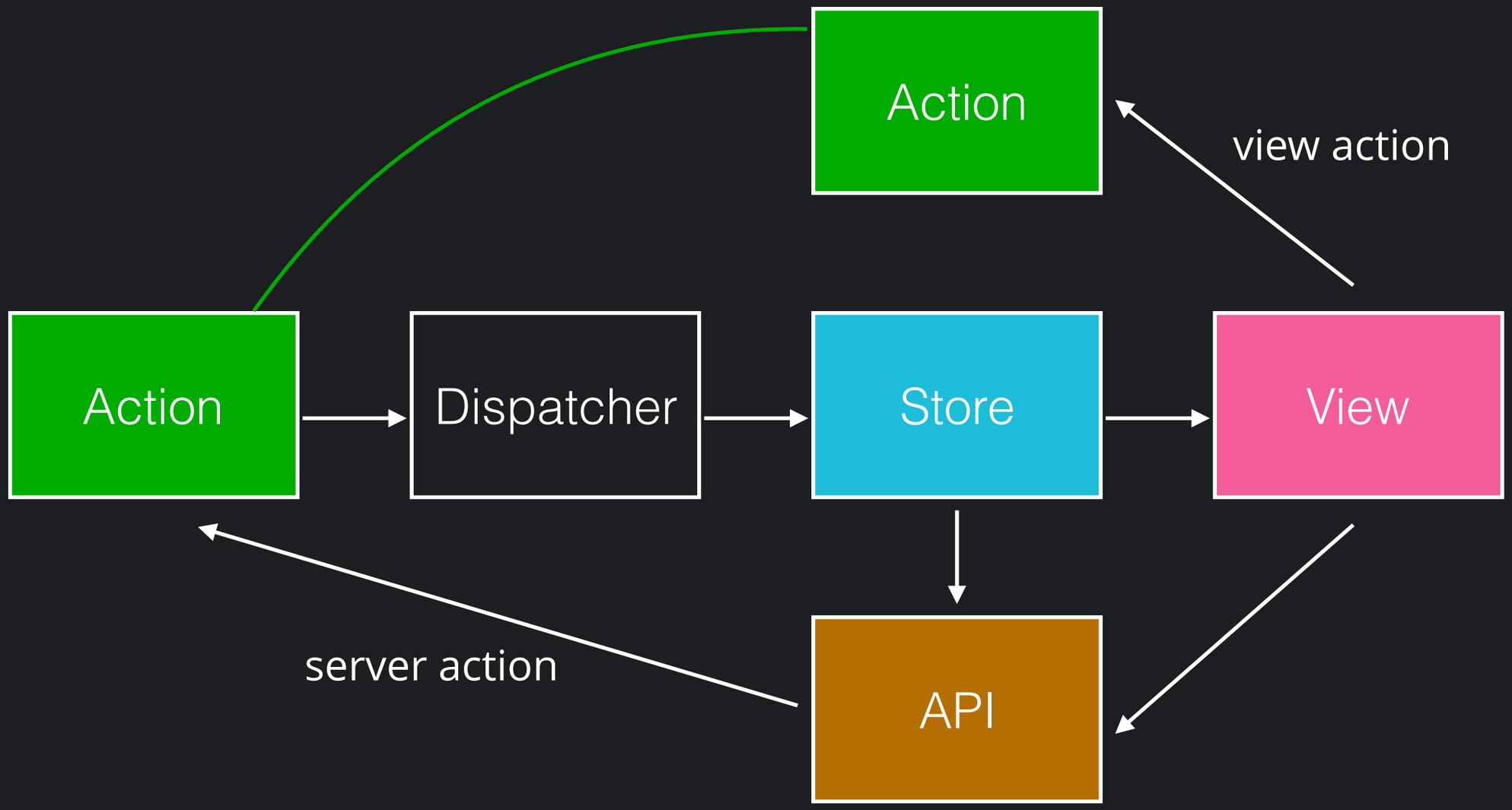
reflux is better than flux because [...]

Flummox is better than flux because [...]

alt is better than flux because [...]

Flux from FB

```
npm install flux --save
```


Action
view action
Action
Dispatcher
Store
View
server action
API

useless dashboard

code review

https://github.com/astrails/react-presentation-flux-redux

tag: #flux

```
export default class HackerNews exte
  constructor() {
    super(...arguments);
    this.state = HackerNewsStore.get
    this.onNewsUpdated = this.onNews
  }

  componentWillMount() {
    HackerNewsStore.addChangeListene
  }

  componentWillUnmount() {
    HackerNewsStore.removeChangeList
  }

  onNewsUpdated() {
    this.setState(HackerNewsStore.ge
  }
```

```
export default class Locations extends Component
  constructor() {
    super(...arguments);
    this.state = LocationsStore.getLocations();
    this.onLocationsChanged = this.onLocationsCha
  }

  componentWillMount() {
    LocationsStore.addChangeListener(this.onLoca
  }

  componentWillUnmount() {
    LocationsStore.removeChangeListener(this.onL
  }

  onLocationsChanged() {
    this.setState(LocationsStore.getLocations())
  }
```

duplication

sync state <- store

direct access to actions and store

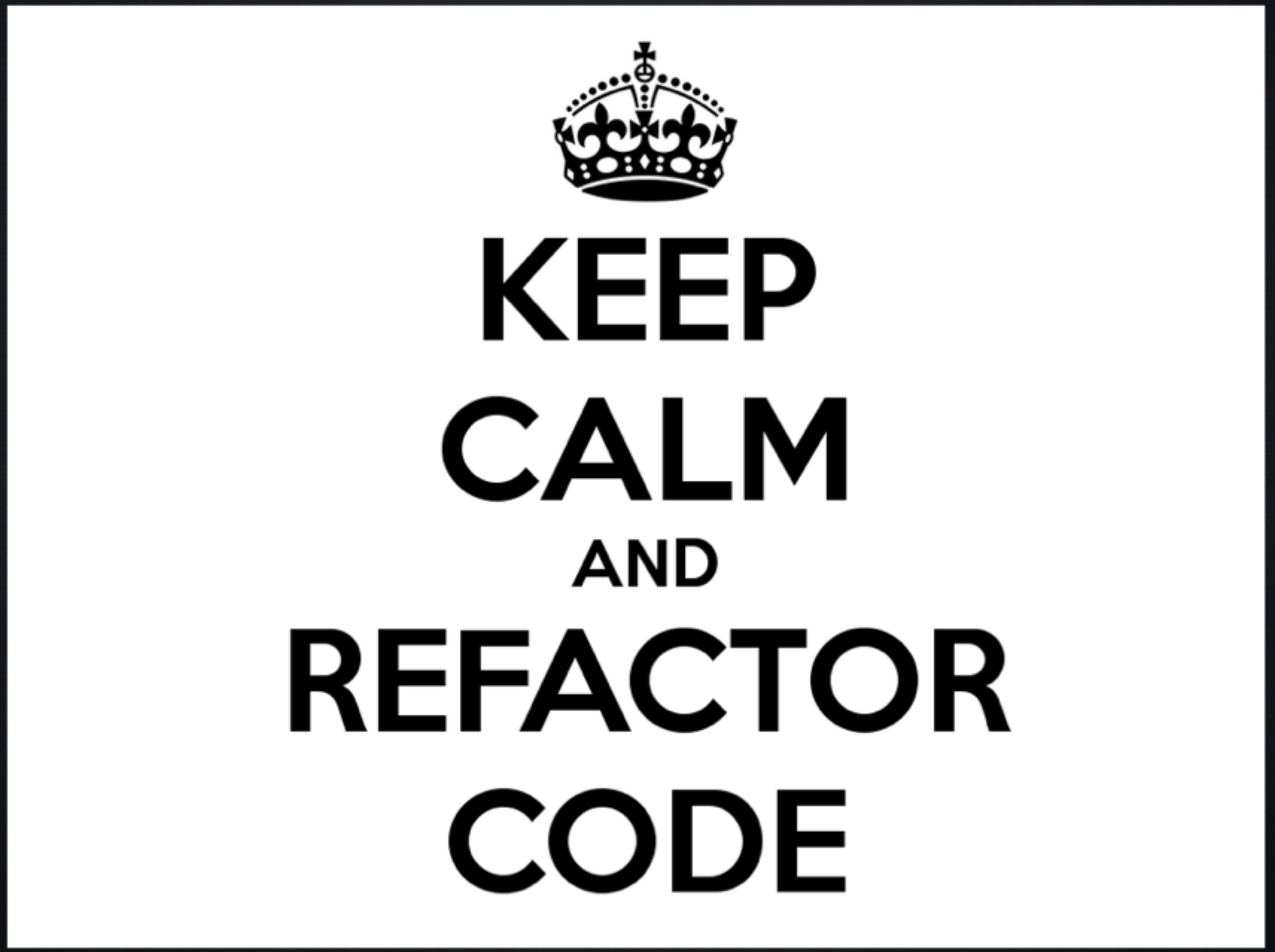


image: http://www.tech-dojo.org/?_escaped_fragment_=/articles/564326afd862ff0b00ef3cc9

smart and dumb

```
render() {
  const allYouNeedIsHere = ...; // functions and data
  return (
    <DumpComponent { ...allYouNeedIsHere } />
  );
}
```

```
// const NewsEntry = (props, context) => {};

const NewsEntry = ({ id, title }) => {
  if (title) {
    return (
      <div>
        <a href={ `...${id}` } target="_blank">{ title }</a>
      </div>
    );
  }

  return (
    <div>Loading entry data...</div>
  );
};
```

pure function

maintain its own state

but don’t access actions and store

indeed reusable

less talk, more work

refactor actions

```
const FormActions = {
  addLocation: (location) => {
    handleViewAction({
      type: Constants.Form.ADD_LOCATION,
      location
    });
    RemoteApi.queryTheWeather(location);
  }
};
```

```
export const addLocation = (location) => {
  handleViewAction({
    type: Constants.Form.ADD_LOCATION,
    location
  });
  RemoteApi.queryTheWeather(location);
};
```

```
const LocationActions = {
  refresh: (location) => {
    handleViewAction({ type:
Constants.Location.REFRESHING, location });
    RemoteApi.queryTheWeather(location);
  },

  querySuccess: (data, location) => {
    handleServerAction({
      type:
Constants.Location.QUERY_SUCCESS,
      data,
      location
    });
  },

  queryFailed: (error) => {
    handleServerAction({
      type: Constants.Location.QUERY_FAILED,
      error
    });
  },

  remove: (locationID) => {
    handleViewAction({ type:
Constants.Location.REMOVE, locationID });
  }
};
```

```
export const refresh = (location) => {
  handleViewAction({ type: Constants.Location.RE
  RemoteApi.queryTheWeather(location);
};

export const remove = (locationID) => {
  handleViewAction({ type: Constants.Location.RE
};

export const querySuccess = (data, location) =>
  handleServerAction({
    type: Constants.Location.QUERY_SUCCESS,
    data,
    location
  });
};

export const queryFailed = (error) => {
  handleServerAction({
    type: Constants.Location.QUERY_FAILED,
    error
  });
};
```

refactor components

```jsx
const Home = () => (
  <div className={ classes.home }>
    <h2>Useless dashboard, flux version</h2>
    <Form />
    <Locations />
    <HackerNews />
  </div>
);
```

```jsx
const Home = () => (
  <div className={ classes.home }>
    <h2>Useless dashboard, flux version</h2>
    <Weather />
    <HackerNews />
  </div>
);
```

```
import { addLocation } from 'fluxx/actions/form';
import { refresh, remove } from 'fluxx/actions/location';

const Weather = () => (
  <div>
    <Form addLocation={ addLocation } />
    <Locations remove={ remove } refresh={ refresh } />
  </div>
);
```

```
export default class Locations extends Component {
  static propTypes = {
    remove: PropTypes.func.isRequired,
    refresh: PropTypes.func.isRequired
  };


//…

  render() {
    return (
      <div className={ classes.locations }>
        { this.state.locations.map(location => <Location { ...this.props } { ...location } key={ location.uniqId } />) }
      </div>
    );
  }
}
```

```jsx
import React from 'react';
import classes from './location.sass';

const Location = ({ remove, refresh, status, temperature, icon_url: iconUrl, text, location, uniqId }) => {
  const removeLocation = (e) => {
    e.preventDefault();
    remove(uniqId);
  };

  const refreshLocation = (e) => { /* … */ };

  const renderState = () => { /* … */ };

  return (
    <div className={ classes.location }>
      <a className={ classes.remove } href="#" onClick={ removeLocation }>&times;</a>
      <div className={ classes.name }>
        { location }
      </div>
      { renderState() }
    </div>
  );
};

export default Location;
```

```
export default class HackerNews exte
  constructor() {
    super(...arguments);
    this.state = HackerNewsStore.get
    this.onNewsUpdated = this.onNews
  }

  componentWillMount() {
    HackerNewsStore.addChangeListene
  }

  componentWillUnmount() {
    HackerNewsStore.removeChangeList
  }

  onNewsUpdated() {
    this.setState(HackerNewsStore.ge
  }
```

```
export default class Locations extends Component
  constructor() {
    super(...arguments);
    this.state = LocationsStore.getLocations();
    this.onLocationsChanged = this.onLocationsCha
  }

  componentWillMount() {
    LocationsStore.addChangeListener(this.onLoca
  }

  componentWillUnmount() {
    LocationsStore.removeChangeListener(this.onL
  }

  onLocationsChanged() {
    this.setState(LocationsStore.getLocations())
  }
```

this shit

options?

mix-ins

aren't they dead?

Mixins Are Dead. Long Live Composition

https://medium.com/@dan_abramov/mixins-are-dead-long-live-higher-order-components-94a0d2f9e750

react-redux

aware of components and flow control

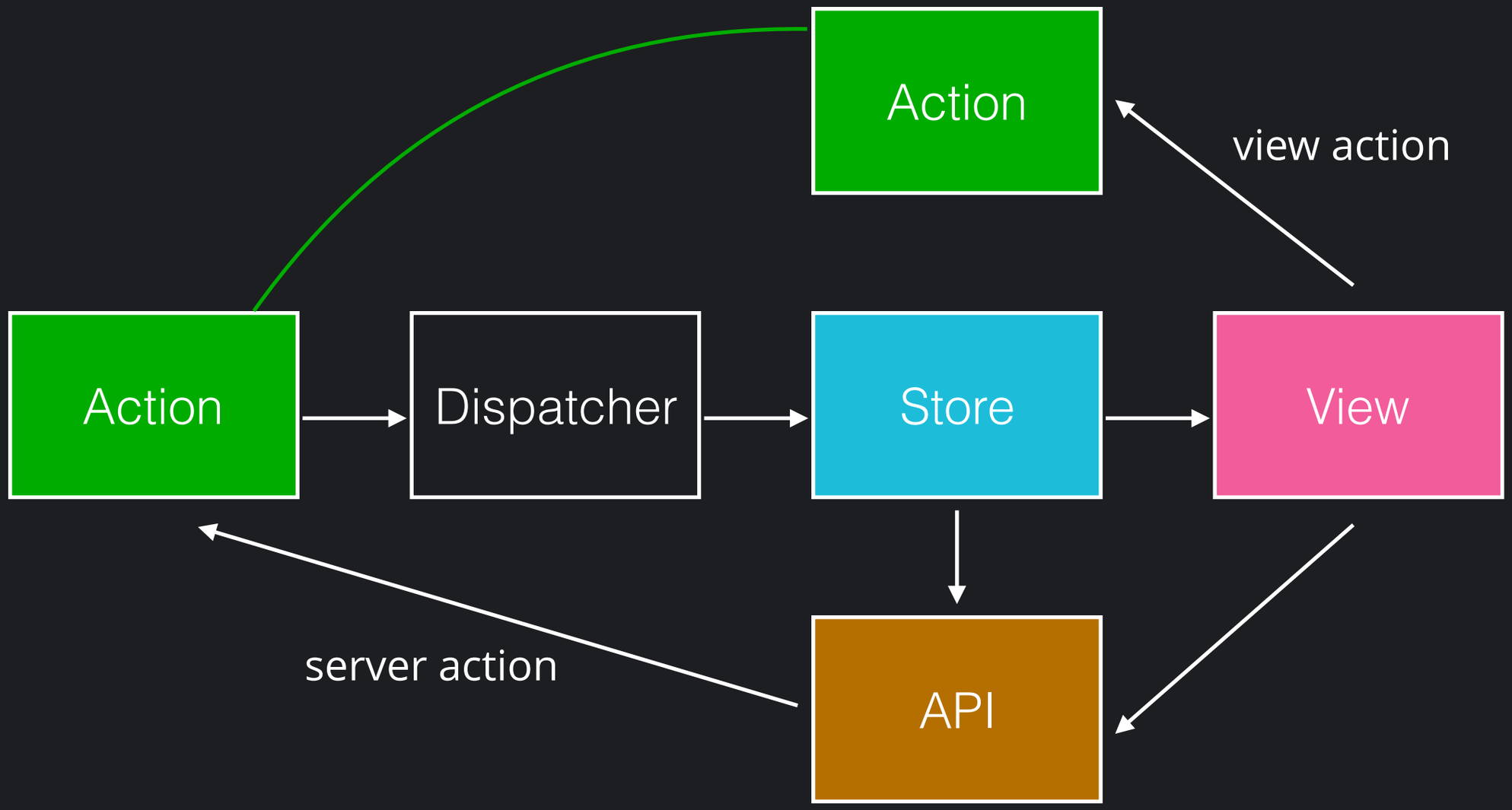
redux

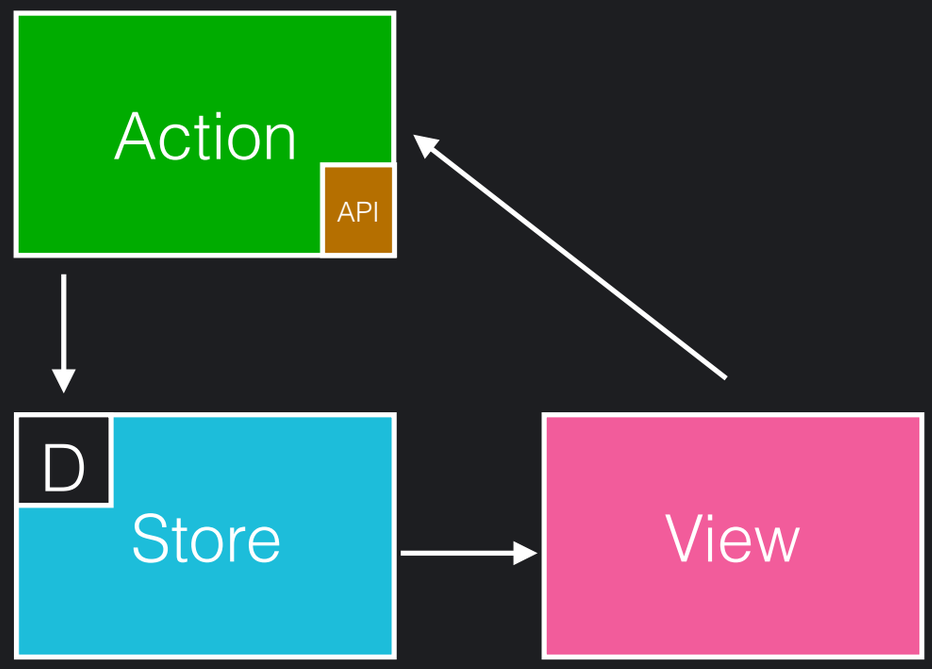
single source of truth

state is read-only

changes are made with pure functions

x!


Action
view action
Action
Dispatcher
Store
View
server action
API


Action
API
D
Store
View

http://www.keepcalm-o-matic.co.uk/p/keep-calm-and-let-the-fun-begin-4/

```
npm i redux --save
npm i react-redux —save
npm i redux-thunk —save
npm i redux-logger --save
```

containers and components

define some terms

components

(presentational components), have no idea about redux

containers

aware of redux, have no own React code

container = smart

component = dumb (less smart)

refactoring plan

api -> better api ;-)

actions -> actions

stores -> reducers

components & containers

API

```
import { querySuccess, queryFailed } from 'fluxx/actions/location';

queryTheWeather: (location) => {
  SuperAgent.get(`…${location}&APPID=${APIKEY}`).set(‘Accept’, 'application/json').
    end((error, res) => {
      if (isSuccess(res)) {
        querySuccess(res.body, location);
     } else {
        queryFailed(res);
      }
    });
},


queryTheWeather: ({ {location}, onSuccess, onFailure }) => {
  SuperAgent.get(`…${location}&APPID=${APIKEY}`).set(‘Accept’, 'application/json').
    end((error, res) => {
      if (isSuccess(res)) {
        onSuccess(res.body);
     } else {
        onFailure(res);
      }
    });
},
```

not aware of actions

actions

```
const querySuccess = (data, location) => ({ type: Constants.Location.QUERY_SUCCESS, data, loca

const queryFailed = (error) => ({ type: Constants.Location.QUERY_FAILED, error });

const startAddingLocation = (location) => ({ type: Constants.Form.ADD_LOCATION, location });

export const addLocation = (location) => (dispatch) => {
  dispatch(startAddingLocation(location));
  RemoteApi.queryTheWeather({
    payload: { location },
    onSuccess: (data) => {
      dispatch(querySuccess(data, location));
    },
    onFailure: (error) => {
      dispatch(queryFailed(error));
    }
  });
};
```

for each async action, usually there are 3 sync actions:

starting the request (update the UI)

request succeed

request failed

reducers

```
✗ cat src/reduxx/actions/location.js| grep type:
const querySuccess = (data, location) => ({ type: Constants.Location.QUERY_SUCCESS, data, location });
const queryFailed = (error) => ({ type: Constants.Location.QUERY_FAILED, error });
const startAddingLocation = (location) => ({ type: Constants.Form.ADD_LOCATION, location });
const startRefershing = (location) => ({ type: Constants.Location.REFRESHING, location });
export const remove = (locationID) => ({ type: Constants.Location.REMOVE, locationID });
```

```
switch (action.type) {
  case Constants.Form.ADD_LOCATION:
    // FIXME: yeah, need a better uniq id
    locations.push({
      location: action.location,
      status: "Looking outside...",
      uniqId: action.location
    });
    return LocationsStore.emitChange();

  case Constants.Location.QUERY_SUCCESS:
    updateLocation(action.location, action.data);
    return saveChanges();

  case Constants.Location.REMOVE:
    removeLocation(action.locationID);
    return saveChanges();

  case Constants.Location.REFRESHING:
    updateState(action.location, "Refreshing data...");
    return saveChanges();

  default:
    return true;
}
```

```
const locations = createReducers(DEFAULT_STATE, {
  [Constants.Form.ADD_LOCATION]: (state, action) => {
    const { location } = action;
    // FIXME: yeah, need a better uniq id
    return [...state, { location, uniqId: location, status: LOOKING_OUTSIDE }];
  },

  [Constants.Location.QUERY_SUCCESS]: (state, action) => updateState(state, acti

  [Constants.Location.REFRESHING]: (state, action) => updateState(state, action.

  [Constants.Location.REMOVE]: (state, action) => removeLocation(state, action.l

  // so what, do your own error handling ;-)
  [Constants.Location.QUERY_FAILED]: (state) => state
});
```

```
 switch (action.type) {
   case Constants.HackerNews.TOP_IDS_LOADED:
     newsIds = action.ids;
     return HackerNewsStore.emitChange();

   case Constants.HackerNews.STORY_FETCHED:
     news[action.id] = action.data;
     return HackerNewsStore.emitChange();

   default:
     return true;
 }



const hackerNews = createReducers(DEFAULT_STATE, {
  [Constants.HackerNews.TOP_IDS_LOADED]: (state, action) => ({ ...state, newsIds: action.ids }),
  [Constants.HackerNews.STORY_FETCHED]: (state, action) => ({ ...state, news: { ...state.news, [action.id]:
action.data } })
});
```

creating store

```
const allowDebugging = (process.env.NODE_ENV !== 'production') ||
  (localStorage && localStorage.getItem('reactDebug') === 'yes');

export const buildStore = () => {
  const rootReducer = combineReducers({ locations, hackerNews });

  const devToolsExt = (allowDebugging && window.devToolsExtension) ?
    window.devToolsExtension() :
    f => f;

  const middleWare = allowDebugging ?
    applyMiddleware(thunkMiddleware, createLogger()) :
    applyMiddleware(thunkMiddleware);

  const store = createStore(
    rootReducer,
    undefined,
    compose(middleWare, devToolsExt)
  );

  return store;
};
```

components & containers

```
import HackerNews from 'containers/hacker_news';
import Weather from 'containers/weather';

const store = buildStore();

const Home = () => (
  <Provider store={ store }>
    <div className={ classes.home }>
      <h2>Useless dashboard, redux version</h2>
      <Weather />
      <HackerNews />
    </div>
  </Provider>
);
```

container

```
import { addLocation, refresh, remove } from 'reduxx/actions/locations';
import Weather from 'components/weather';
import { connect } from 'react-redux';

const mapDispatchToProps = (dispatch) => ({
  addLocation: (location) => dispatch(addLocation(location)),
  refresh:     (location) => dispatch(refresh(location)),
  remove:      (location) => dispatch(remove(location))
});

const mapStateToProps = (state) => ({ locations: state.locations });

export default connect(mapStateToProps, mapDispatchToProps)(Weather);
```

component

```
import React, { PropTypes } from 'react';
import Form from 'components/form';
import Locations from 'components/locations';

const Weather = ({ addLocation, refresh, remove, locations }) => (
  <div>
    <Form addLocation={ addLocation } />
    <Locations remove={ remove } refresh={ refresh } locations={ locations } />
  </div>
);
```

https://github.com/astrails/react-presentation-flux-redux

tag: #redux

move files to folders - less mess

```
▾ src/
  ▾ components/
      form.js
      hacker_news.js
      hacker_news.sass
      home.js
      home.sass
      location.js
      location.sass
      locations.js
      locations.sass
      news_entry.js
```

```
▾ src/
  ▾ components/
    ▾ hacker_news/
        hacker_news.sass
        index.js
    ▾ home/
        home.sass
        index.js
    ▾ location/
        index.js
        location.sass
    ▾ locations/
        index.js
        locations.sass
      form.js
      news_entry.js
      weather.js
  ▾ containers/
      hacker_news.js
      weather.js
```

adding propTypes

kinda documentation you'll use

```
"react/prop-types": 2,
```

```
/Users/boris/a/astrails/react-presentation/src/components/locations/index.js
  5:22  error  'locations' is missing in props validation  react/prop-types
  5:33  error  'refresh' is missing in props validation    react/prop-types
  5:42  error  'remove' is missing in props validation     react/prop-types

✖ 3 problems (3 errors, 0 warnings)
```

```
const Location = ({ remove, refresh, status, temperature, icon_url: iconUrl, text, location, uniqId }) => {
  //…
};

Location.propTypes = {
  remove: PropTypes.func.isRequired,
  refresh: PropTypes.func.isRequired,
  status: PropTypes.string.isRequired,
  temperature: PropTypes.number,
  icon_url: PropTypes.string,
  text: PropTypes.string,
  location: PropTypes.string.isRequired,
  uniqId: PropTypes.string.isRequired
};



export default class HackerNews extends Component {
  static propTypes = {
    hackerNews: PropTypes.arrayOf(PropTypes.shape({
      id: PropTypes.number.isRequired,
      title: PropTypes.string
    })),
    fetch: PropTypes.func.isRequired
  };

  //…
}
```

https://flic.kr/p/7EsetZ

```
newsItem: PropTypes.shape({…})
```

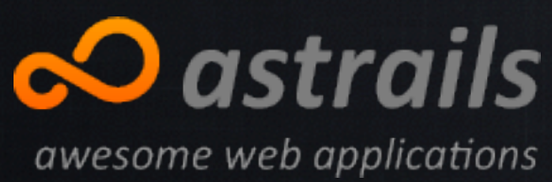
https://github.com/astrails/react-presentation-flux-redux

branch: master

thanks!

http://io9.gizmodo.com/5794957/see-darth-vader-riding-a-unicorn-it-is-your-destiny/

Q&A



Boris Nadion
http://astrails.com