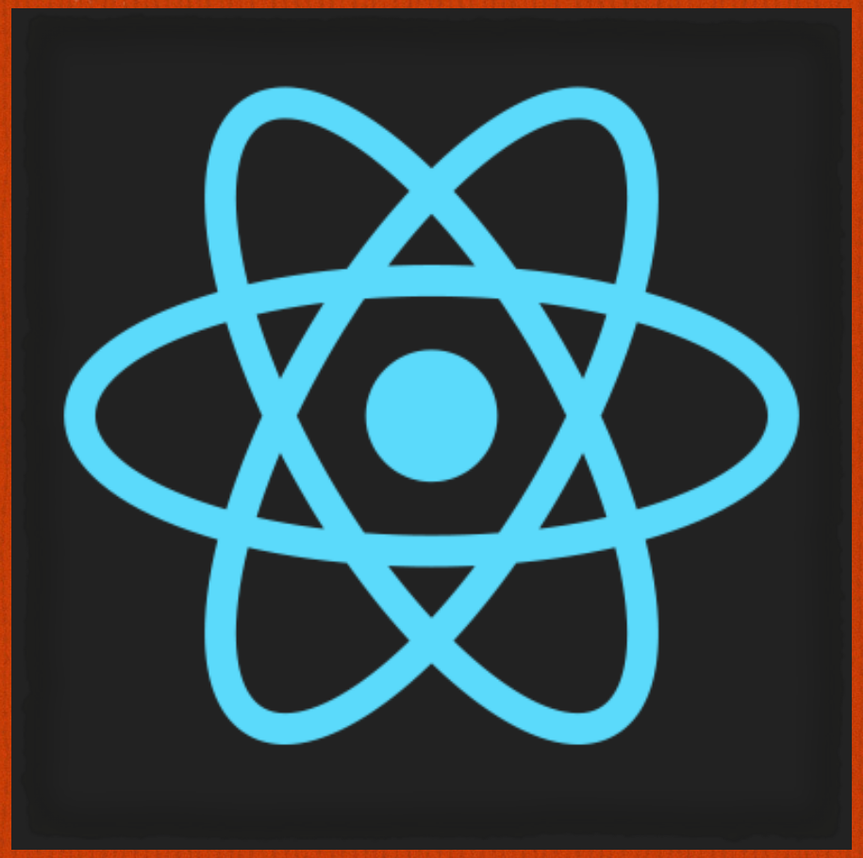
# Advanced React & Redux

Ynon Perek
ynon@ynonperek.com



**bit.ly/react-perion**

Offer a ***consistent*** and ***simple*** way to write web applications

# History & Culture

facebook

Instagram

Flipboard

Dropbox

WIX.com

KHAN ACADEMY

# History & Culture

- Created by Jordan Walke
- 2011 - first deployed in FB
- May 29, 2013 - initial public release (0.3.0)
- April 7, 2016 - React 15

# React Vs. The World

- ☐ Vanilla
- ☐ jQuery
- ☐ Backbone, Angular1 (MVC)
- ☐ React, Angular2, Vue (Component Based)

# React IDE

- ☐ Webstorm
- ☐ vim
- ☐ Anything in between...

# React Build Tools

- ☐ Webpack + Babel
- ☐ npm
- ☐ eslint
- ☐ Demo

# Debugging React

- [ ] Use source maps
- [ ] Use debugger instruction
- [ ] Assign globals to window

# Q & A

# Sharing Code Between Components

# Agenda

- ☐ Class Inheritance
- ☐ Higher Order Components
- ☐ External Code (actions)

# Let’s Implement Render Lock

- ☐ Create a global “lock” that when locked should prevent change in all components
- ☐ Implement common logic in 3 methods: Inheritance, HOC, Actions

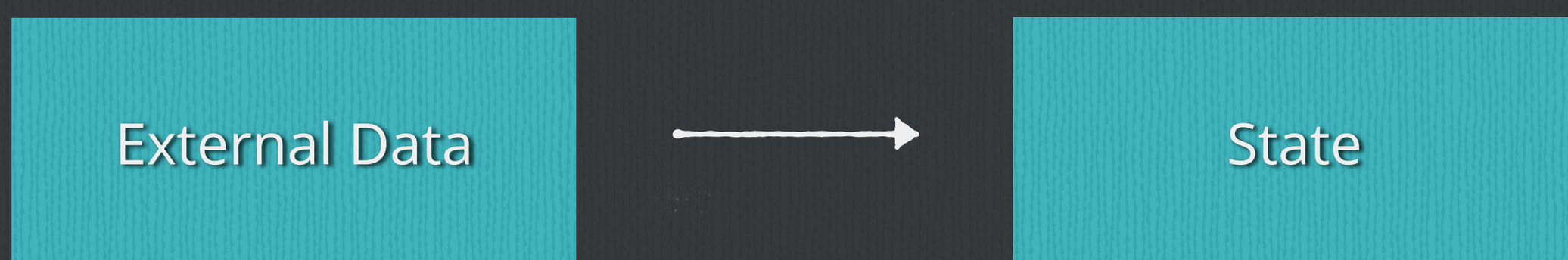
# External Actions

- When code can be written outside a React component, it probably should be
- Example: The lock object

# External Actions

```
class RenderLock {
  constructor() {
    this.locked = false;
  }

  isLocked() {
    return this.locked;
  }

  toggle() {
    this.locked = !this.locked;
    this.trigger('change');
  }
}

Object.assign(RenderLock.prototype, EventEmitter.prototype);
window.lock = new RenderLock();
```

# Connecting External Services



# Connecting External Services

```
class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {};
    this.readLockState();

    this.readLockState = this.readLockState.bind(this);
  }

  readLockState() {
    this.setState({ locked: this.props.globalLock.isLocked() });
  }

  componentWillMount() {
    this.props.globalLock.on('change', this.readLockState);
  }
}
```

# Your Turn #1

- ☐ Write two React components that have visible state that can be modified
- ☐ Implement shouldComponentUpdate to check global lock before rendering
- ☐ Extract common functionality to a common base class

# Your Turn #2

- ☐ Extract lock connection code from App to a base class, so we'll be able to easily create more components that affect the lock

# Discussion

- ☐ What Works?
- ☐ What Doesn’t Work?

# HOC

- A function that takes a component and returns another component is called “higher order component”

# HOC: Hello World

```
function around(WrappedComponent) {
  return function(props) {
    return (
      <div className='wrapper'>
        <WrappedComponent {...props} />
      </div>
    )
  }
}
```

# HOC: Hello World (classes)

```
function around(WrappedComponent) {
  class Wrapper extends React.Component {
    render() {
      return (
        <div className='wrapper'>
          <WrappedComponent {...this.props} />
        </div>
      )
    }
  }
  return Wrapper;
}
```

# HOC: display name

```
function getWrappedDisplayName(wrapped) {
  return wrapped.displayName || wrapped.name || 'Component';
}

function around(WrappedComponent) {
  class Wrapper extends React.Component {  }

  Wrapper.displayName = `RenderCounter(
          ${getWrappedDisplayName(WrappedComponent)})`;

  return Wrapper;
}
```

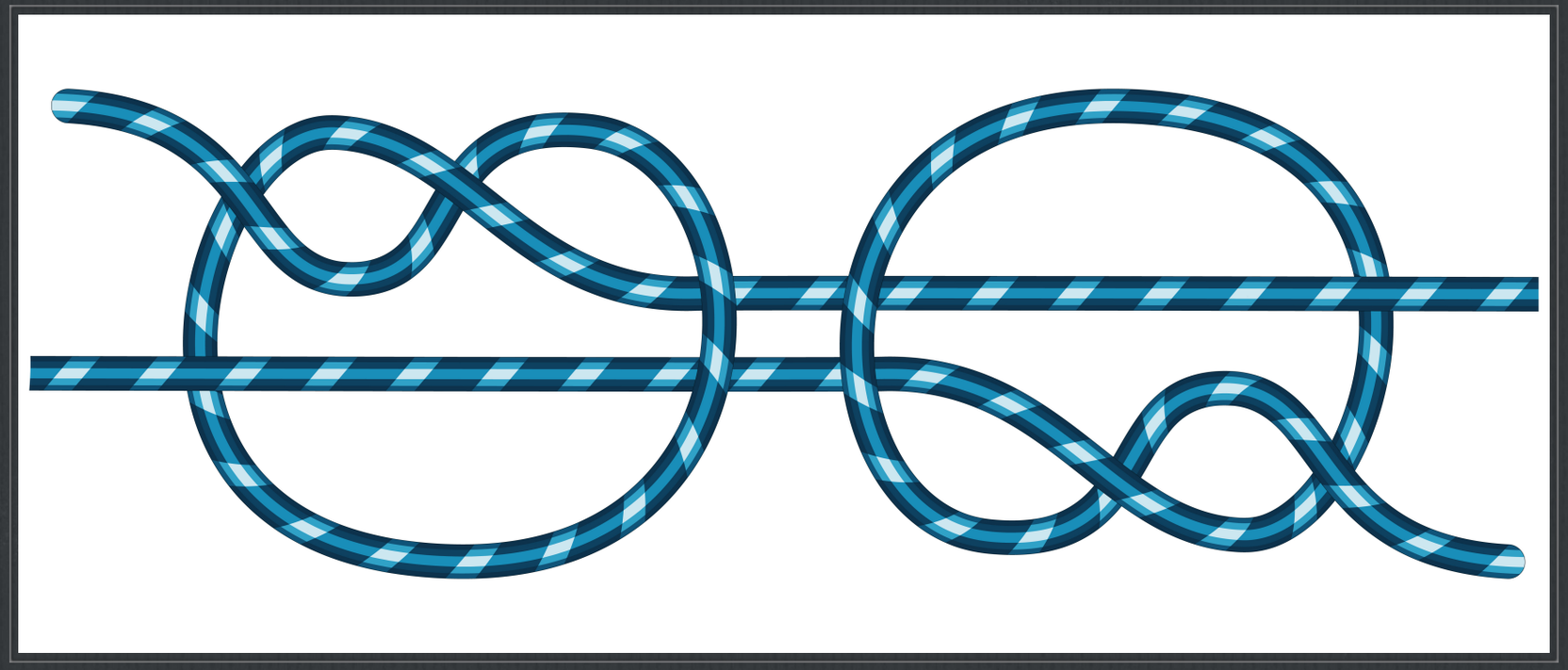
# Using HOC

```
class App extends React.Component {
  render() {
    return (
      <div>
        <p>Hello World</p>
      </div>
    );
  }
}

App = around(App);
```

# Using HOC (ES7)

```
@around
class App extends React.Component {
  render() {
    return (
      <div>
        <p>Hello World</p>
      </div>
    );
  }
}
```

# Discussion:
# HOC Vs. Inheritance



# HOC Lab #1

- [ ] Create an HOC component to handle subscribing <App> to Lock object - from the previous exercise
- [ ] Can you also move each Element's locking mechanism to a HOC?

# HOC Lab #2

- ☐ Create a withAPIData HOC that provides JSON data from an API to a component. Usage:

```
@withAPIData('https://swapi.co/api/people/1/?format=json')
class App extends React.Component {
  render() {
    return (
      <div>
        <p>Data: {JSON.stringify(this.props.data)}</p>
      </div>
    );
  }
}
```

# HOC Lab #3

- What is wrong in the following HOC code?

```
function withSpinner(WrappedComponent) {
  return function(props) {
    if (WrappedComponent.loading) {
      return <p>Loading...</p>
    } else {
      return <WrappedComponent {...props} />
    }
  }
}
```

# Q & A

# Forms & React

- Problem - high coupling:
- https://codepen.io/ynonp/pen/qjMzeO

# Better Way

- Use small, replaceable components:

# ContactForm

```
render() {
  return (
    <Form
      onSubmit={this.printFormData}
      schema={this.formSchema()}
    >

      <TextField name='name' label='User Name' />
      <EmailField name='email' label='Email' />
    </Form>);
}
```

**Code: https://codepen.io/ynonp/pen/NgLQpO**

# Lab

- ☐ Add a <CountryCitySelect /> field that provides two select boxes that depend on each other

```
<TextField name='name' label='User Name' />
<CountryCitySelect
    options={{
      Belgium: ['Brussels', 'Antwerpen'],
      Denmark: ['Copenhagen', 'Arhus'],
      France: ['Paris', 'Bourgogne'],
      Italy: ['Rome', 'Calabria', 'Campania'],
    }}
/>
```

# Lab

- [ ] Result should be 2 select boxes, where selecting a country in one box renders the list of relevant cities in the other
- [ ] The box should add 2 fields to the form: country and city

# Custom Form Controls

- Use el.focus() to assign keyboard focus
- Use onFocus to respond to focus change events
- Demo: https://codepen.io/ynonp/pen/JJmjyq

# Lab: Custom Form Controls

- ☐ Add to the panel (from previous slide):
  - ☐ Ability to take onChange attribute and report value change
  - ☐ Ability for user to use “Backspace” to delete a character

# Q & A

# Styling React Components

- ☐ JS Style
- ☐ CSS file style (CSS modules)
- ☐ Themes (using context)

# JS Style - How

```
const divStyle = {
  color: 'white',
  backgroundImage: 'url(' + imgUrl + ')',
  transition: 'all 0.2s',
};

ReactDOM.render(
  <div style={divStyle}>Hello World!</div>,
  mountNode);
```

# JS Style - How

```
const commonStyle = {
  fontFamily: 'Arial',
  fontSize: '16px',
};

const divStyle = {
  color: 'white',
  fontSize: '24px',
};

const combinedStyle = Object.assign({},
                                    commonStyle,
                                    divStyle);
```

# JS Style - Why

```
class CoolTheme extends BaseTheme {
  button() {
    return Object.assign({},
      this.baseStyle(),
      {
        color: 'blue',
        border: '2px solid lightblue',
      }
    );
  }
}

ReactDOM.render(
  <button style={theme.button()}>Hello World!</button>,
  mountNode);
```

# JS Style - Why Not

- No media queries
- No pseudo classes (:hover, :active)
- Hard to maintain

# JS Style - Radium

- Adds media queries and pseudo classes
- http://formidable.com/open-source/radium/

# Radium - behind the scenes

- Automatically add event handlers and state to components https://github.com/FormidableLabs/radium/blob/777647b2935594d3131233158b4e775b8e50edc2/src/plugins/resolve-interaction-styles-plugin.js

# CSS Modules - BEM

```
.select {}
.select__item {}
.select__item__icon {}
.select--loading {}
```

A CSS Module is a CSS file in which all class names and animation names are scoped locally by default.

# How It Works

```
.container {
  background: orange;
}
```

```
import css from 'app.css';

const App = function (props) {
  return (
    <p className={css.container}>
      Hello World
    </p>
  );
};
```

# Why

- Scoped by default
- Easy to find unused CSS

# Why Not

- Class names are no longer in your control
- Add complexity when working with non-react designers/developers

# Webpack Config

- CSS loader creates the CSS modules
- Style loader injects result into HTML
- ExtractTextPlugin writes a styles.css file
- Demo: 02-cssmodules

# Q & A

# Context

# The Problem

```
function App(props) {
  return <Page appName="demo"></Page>
}

function Page(props) {
  return (
    <List appName={props.appName}>
      <Item appName={props.appName} />
    </List>
  );
}

function Item(props) {
  if ( appName.length > 10 ) { ... }
}
```

# Context

- Send property “down the tree” to deeply nested components

# Context - How

```
class App extends React.Component {

  getChildContext() {
    return { appName: 'demo' };
  }

  render() {
    return <Page />
  }
}

App.childContextTypes = {
  appName: React.PropTypes.string,
}
```

# Context - How

```
function Item(props, context) {
  return <p>name = {context.appName}</p>
}

Item.contextTypes = {
  appName: React.PropTypes.string,
}
```

# Context Limitations

- Updating data in context does not always re-render child components that use it

# Context + Styles = Theme

```
class Button extends React.Component {
  render() {
    style = Object.assign(
      {},
      this.context.theme,
      { background: 'red' },
    );
  }
}

Button.contextTypes = {
  theme: PropTypes.object
};
```

# Q & A

# Lab

- Given the following project, change style to:
  1. Use JS inline style
  2. Use CSS Modules
- Project: https://github.com/garethrees/landing-page/archive/master.zip

# React Lifecycle Methods

# In Brief

constructor()

componentWillMount()

render()

componentDidMount()

componentWillReceiveProps()

shouldComponentUpdate()

componentWillUpdate()

render()

componentDidUpdate()

componentWillUnmount()

# constructor

- Called once - when component is created
- create state
- bind methods

# constructor

```
constructor(props) {
  super(props);
  this.state = {
    color: 'red',
  };
}
```

# componentWillMount

- Called before rendering AND on server
- constructor() is usually the better option

# componentDidMount

- ☐ Trigger network requests
- ☐ Initialize plugins

# componentDidMount

```
componentDidMount() {
  this.player = new YT.Player(this.youtubePlayerEl, {
    height: '390',
    width: '640',
    videoId: 'M7lc1UVf-VE',
  })
}
```

https://codepen.io/ynonp/pen/jLNjVm

# componentWillReceiveProps

- Before *mounted* component receives new props
- Props -> State pattern

# shouldComponentUpdate

- The big question- Did anything change?
- Example implement shouldComponentUpdate to reduce render() actions: https://codepen.io/ynonp/pen/brbPRB

# shouldComponentUpdate

- Shallow compare props (for immutable data)

```
class App extends React.PureComponent {

}
```

# componentWillUpdate

- Called after setState() or props change
- Best to leave blank
- NOTE: don’t call setState here

# componentDidUpdate

- Called after props/state change
- Demo: youtube API dynamic video
  https://codepen.io/ynonp/pen/EvYBrM

# componentWillUnmount

- Free up resources taken by the component:
  - Invalidate timers
  - Cancel network requests

# async setState

- setState calls are async
- this may cause confusion
- see “bug” here: https://codepen.io/ynonp/pen/GvKbbN

# async setState

- Workaround: use callback

```
inc() {
  this.setState({ clicks: this.state.clicks + 1 }, () => {
    this.setState({ clicks: this.state.clicks + 1 });
  });
}
```

# Q & A

# Lifecycle Lab #1

- Write a component called RateYO that wraps the following jQuery plugin: http://rateyo.fundoocode.ninja/
- Props:
  - ratedFill
  - numStars
  - onChange - called when rating changes

# Lifecycle Lab #2

- ☐ The following code uses jQuery Sortable to allow re-ordering of a list, but it doesn't work: https://codepen.io/ynonp/pen/GpZJpv
- ☐ Find the bug and fix

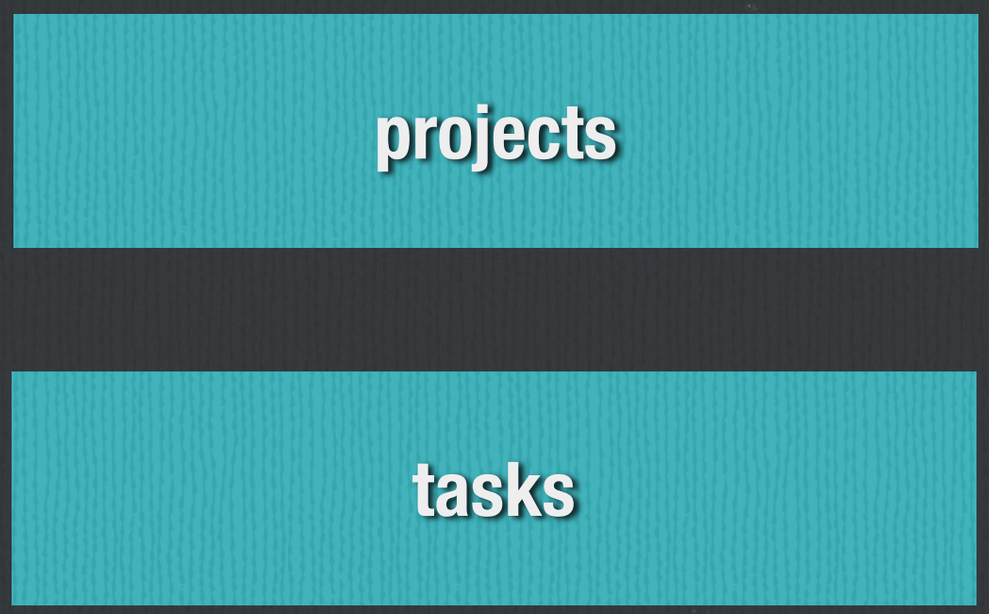
# Immutable Data

# Immutable vs. Mutable data

```
const x = 'hello';
const y = x;

f(y);

console.log(x);
```

```
const x = [];
const y = x;

f(y);

console.log(x);
```

# Immutable vs. Mutable data



# Immutable vs. Mutable data

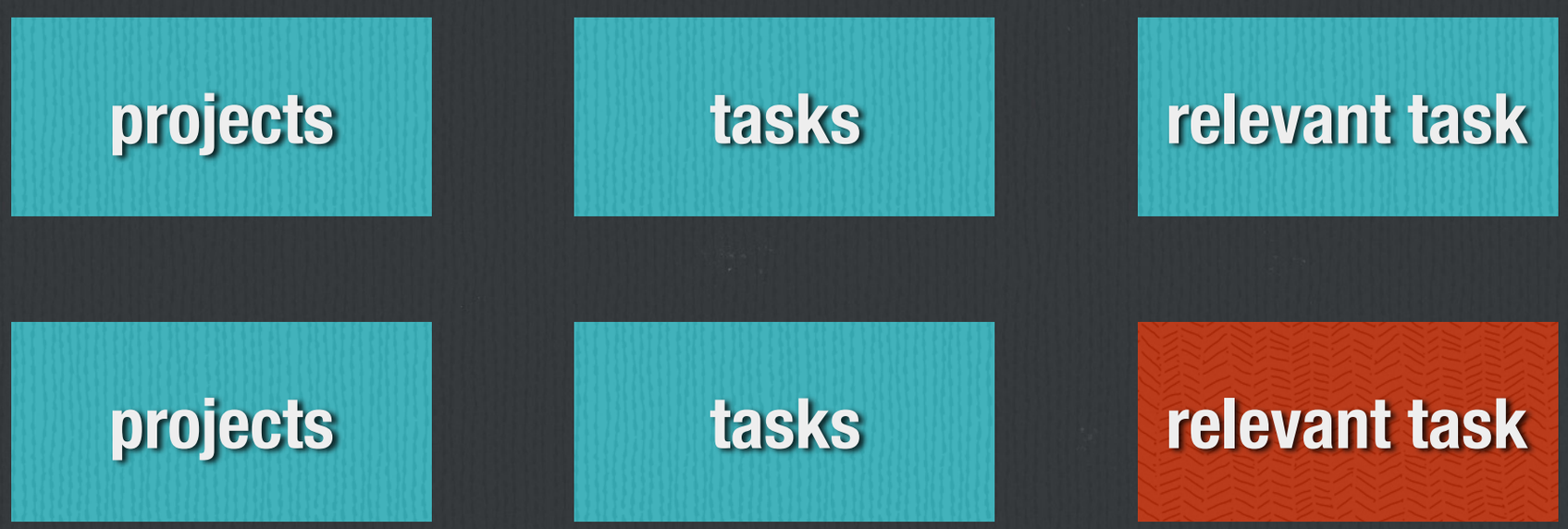
```
const data = {
  projects: [
    {
      name: 'Learn Immutable Data',
      tasks: [
        { id: 1, text: '...', done: false },
        { id: 2, text: '...', done: false },
        { id: 3, text: '...', done: false },
        { id: 4, text: '...', done: false },
      ]
    }
  ]
}
```

# Mutable Change

```
data.projects[0].tasks[2].done = true
```

# Mutable Change Detection

- Deep compare 2 state objects to find the difference

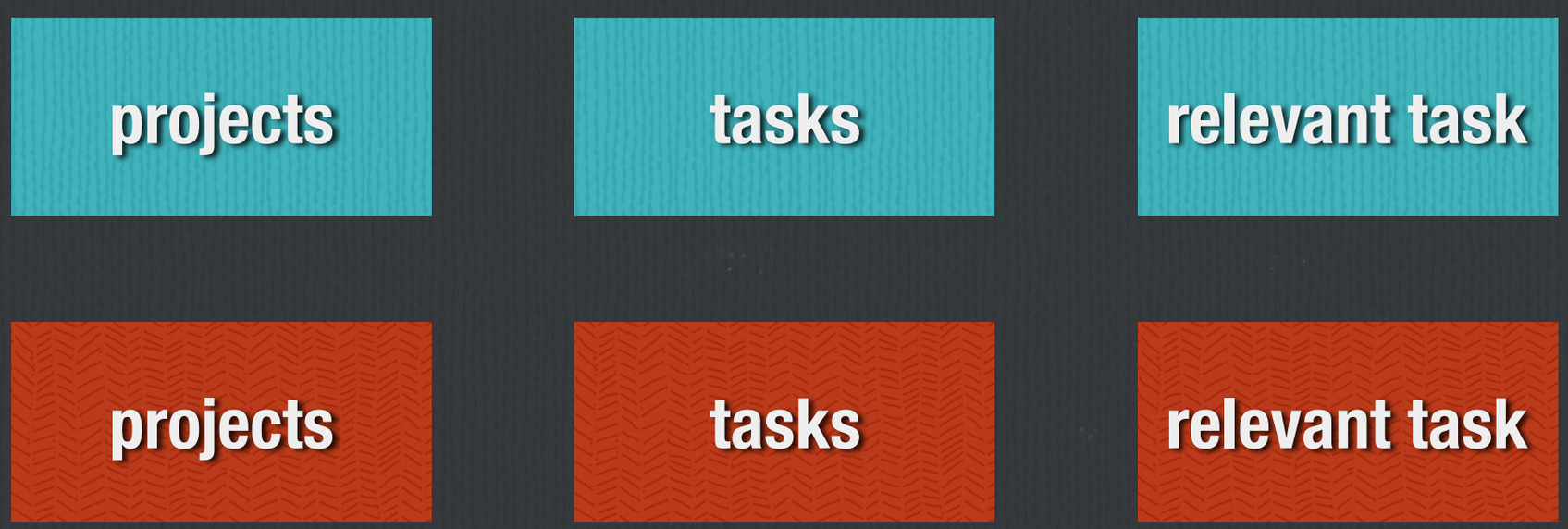


# Immutable Change

```
data = {
  ...data,
  projects: data.projects.map(p, i => i === 0 ?
    {
      ...p,
      tasks: p.tasks.map(t, i => i === 2 ?
        { ...t, done: true }
        : t)
    }
    : p),
}
```

# Immutable Change Detection

- Safe to shallow compare objects. Change is propagated



# Why should you care

- ☐ render is slow
- ☐ maybe safe to skip

# Immutable JS

- A library from FB making it easy to work with immutable data

# Immutable Data

```
import Immutable from 'immutable';
```

```
<script src="
https://cdnjs.cloudflare.com/ajax/libs/
immutable/3.8.1/immutable.js"></script>
```

```
const data = Immutable.List([10, 20, 30]);
```

# Immutable List

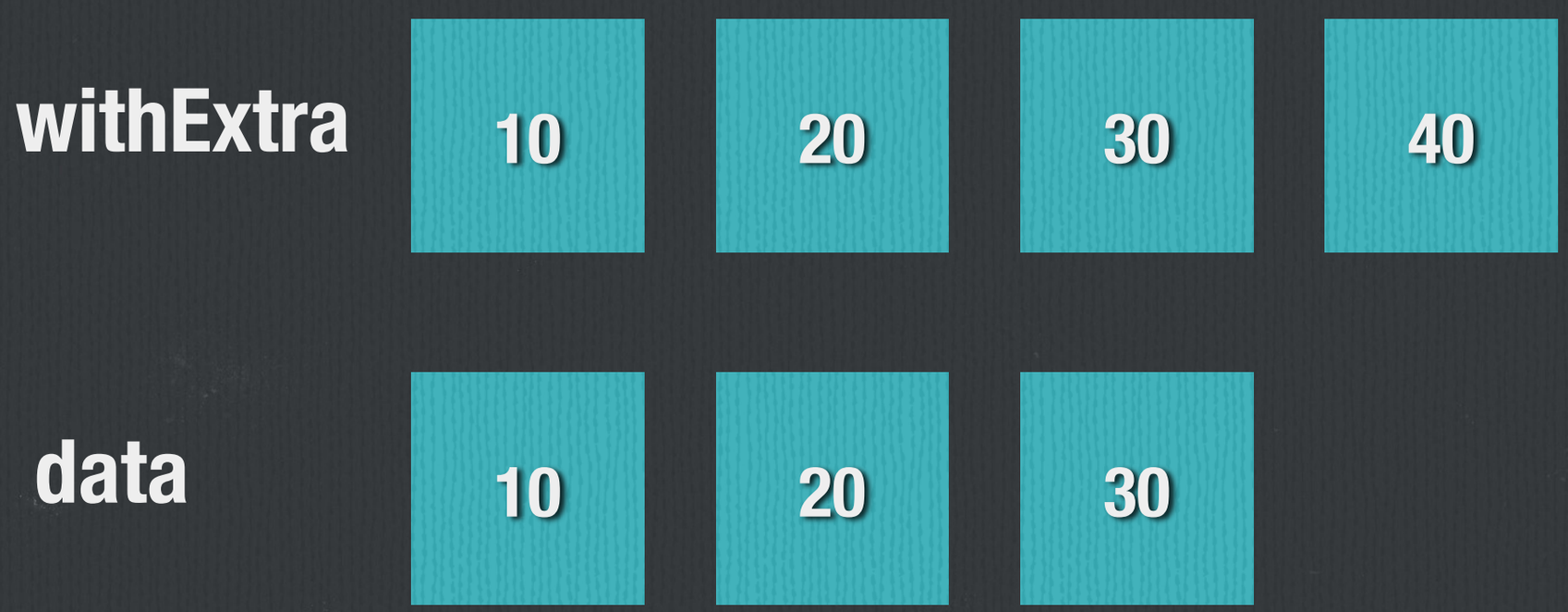
```
const data = Immutable.List([10, 20, 30]);

const withExtra = data.push(40);
```

# Immutable List

```
const data = Immutable.List([10, 20, 30]);

const withExtra = data.push(40);
```

# Immutable List

```
const data = Immutable.List([10, 20, 30]);

const withExtra = data.push(40);
```



```
const data = Immutable.List([10, 20, 30]);

data.size === 3;

// returns the list: [10, 20]
// does not change data
data.delete(2)

// returns the list [10, 'new item', 20, 30]
data.insert(1, 'new item')

// returns the list [10, 20, 30, 50]
data.push(50)
```

```
const data = Immutable.List([10, 20, 30]);

// returns the list [50, 20, 30]
data.set(0, 50);

// returns the list [20, 20, 30]
data.update(0, (val) => val * 2);
```

```
const data = Immutable.List([10, 20, 30]);

const sum = data.reduce((acc, val) => acc + val);

const squares = data.map((x) => x * x);

const even = data.filter((i) => i % 2 == 0);
```

```
const data = Immutable.List([10, 20, 30]);

const sum = data.reduce((acc, val) => acc + val);

const squares = data.map((x) => x * x);

const even = data.filter((i) => i % 2 == 0);
```

```
const data = Immutable.List([10, 20, 30]);

const sum = data.reduce((acc, val) => acc + val);

const squares = data.map((x) => x * x);

const even = data.filter((i) => i % 2 == 0);
```

```
const data = Immutable.List([10, 20, 30]);

const sum = data.reduce((acc, val) => acc + val);

const squares = data.map((x) => x * x);

const even = data.filter((i) => i % 2 == 0);
```

```
const data = Immutable.Map({ a: 10, b: 20 })

data.size === 2

data.get('a') === 10
```

```
const data = Immutable.Map({ a: 10, b: 20 });
data = data.delete('a');

data = data.update('b', (val) => val * 10);

data.get('b') === 200
data.size === 1
```

```
const data = {
    name: 'joe',
    likes: ['skiing', 'dancing'],
    livesIn: {
        country: 'USA',
        city: 'Gotham',
    },
};

data.livesIn.country === 'USA'
```

```
const data = Immutable.fromJS({
    name: 'joe',
    likes: ['skiing', 'dancing'],
    livesIn: {
        country: 'USA',
        city: 'Gotham',
    },
});
```

```
// change joe's city to Dallas
data = data.setIn(['livesIn', 'city'],
                  'Dallas');

// read joe's first hobby
data = data.getIn(['likes', 0]);

// update joe's second hobby to upper case
data = data.updateIn(['likes', 1],
                     (val) => val.toUpperCase());
```

# Q & A

# Lab: Immutable Data

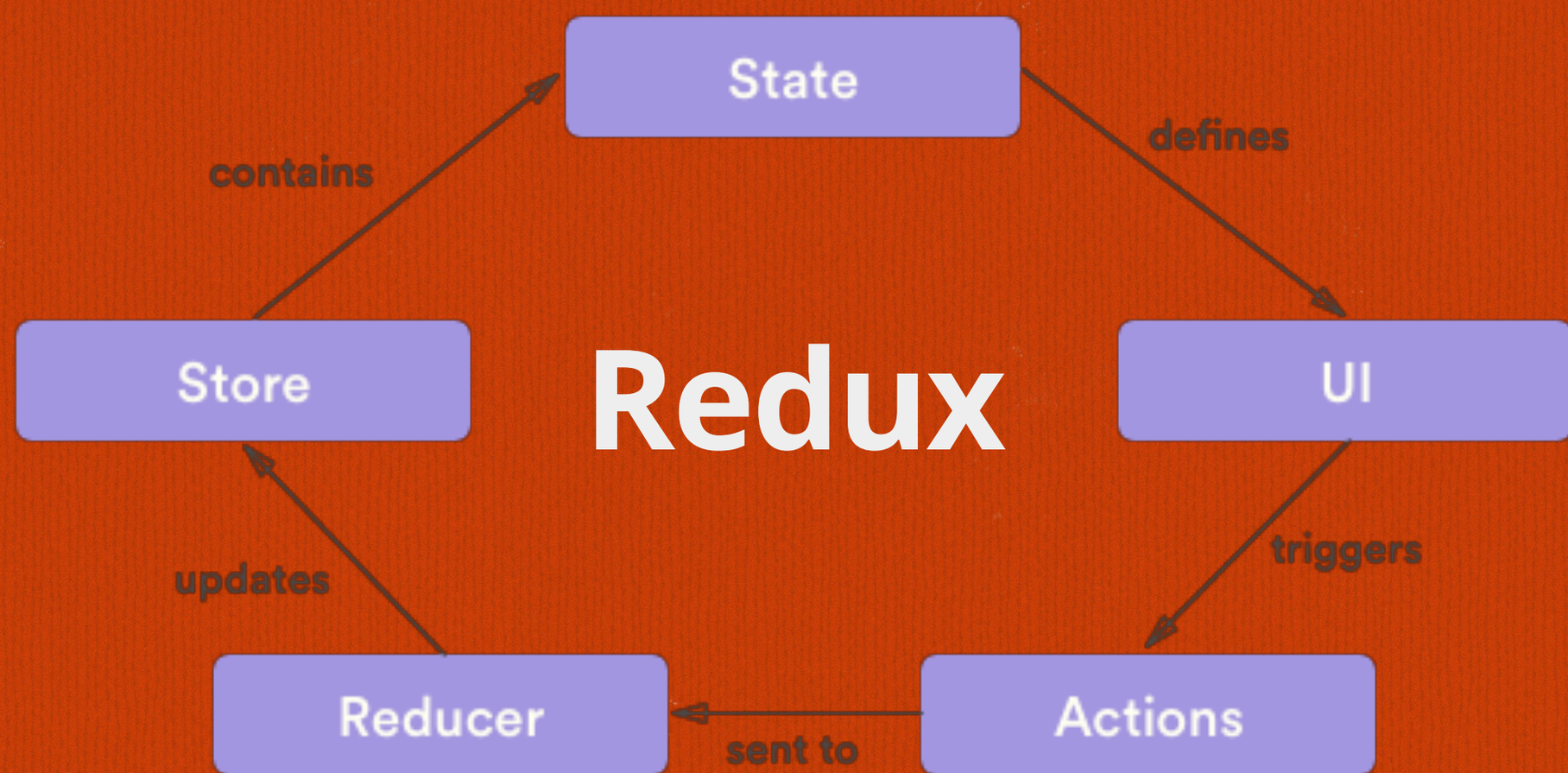
- Given the following immutable object:

```
var state = Immutable.fromJS({
  rooms: [
    { id: 0, name: 'Lobby', current: true },
    { id: 1, name: 'The-Club' },
    { id: 2, name: 'Food and Drinks' },
    { id: 3, name: 'Life etc.' }
  ],
  messages: [
    { from: 'joe', text: 'Morning all' },
    { from: 'jane', text: 'Hi joe' }
  ],
  username: 'jack',
  otherUsers: [ 'joe', 'jane' ]
});
```

# Lab: Immutable Data

- [ ] Change username to “carla”
- [ ] Add 2 new messages from carla
- [ ] Remove all messages from joe
- [ ] Show only messages that have more than 7 characters in their text
- [ ] Move current room to room number 2

# Redux


State
contains
defines
Store
UI
updates
triggers
Reducer
sent to
Actions

# History & Culture

- 2015 - Created by Dan Abramov
- “Ties your hands,
  to free your mind”

# What To Like

- ☐ Tiny API
- ☐ Easy to debug
- ☐ Nicer than flux

# What Not To Like

- ☐ Tiny API
- ☐ Lot of boilerplate code
- ☐ Hard to understand

# Redux Devtools

- [ ] Demo: 03-redux-devtools

# Redux Concepts In Brief

- ☐ Reducer
- ☐ Action
- ☐ Store

# Live Demo: Tic-Tac-Toe Game

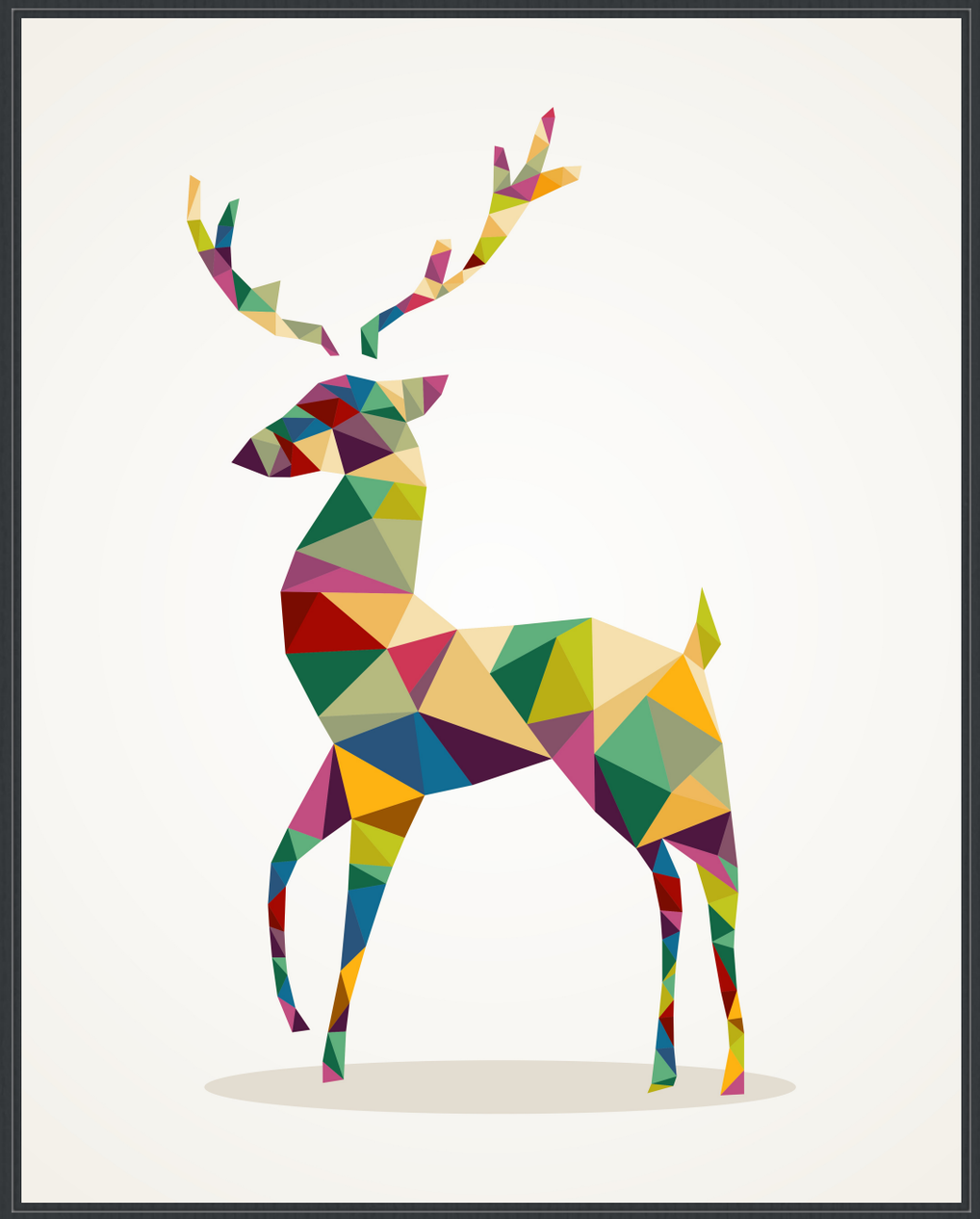
- ☐ Start with state object
- ☐ Add gameReducer
- ☐ add play() action

# Lab

- actions:
  - join(userId, userName)
  - say(userId, messageText)

# Redux 3 Principles

- ☐ Single source of truth
- ☐ State is read-only
- ☐ Changes are made with pure functions



# Multiple Reducers

# The Gist

```
function counter(state = initialState, action) {
  switch (action.type) {
    case 'INCREMENT':
      return inc(state, action);

    default:
      return state
  }
}
```

# Multiple Reducers Pattern

```
function mainReducer(state = initialState, action) {
  return {
    counter: counter(state, action),
    chat: chat(state, action),
    notifications: notifications(state, action),
  }
}
```

# Testing Reducers

- Reducers are very easy to test since they are pure functions
- Demo + Lab: 04-testing-react-redux

# Takeaways

- ☐ Reducers are pure sync functions
- ☐ helper functions -> easier maintenance

# Adding React UI

# Tools

- store.subscribe()
- store state -> component state -> child props
- https://codepen.io/ynonp/pen/JyjJOw

# Q & A

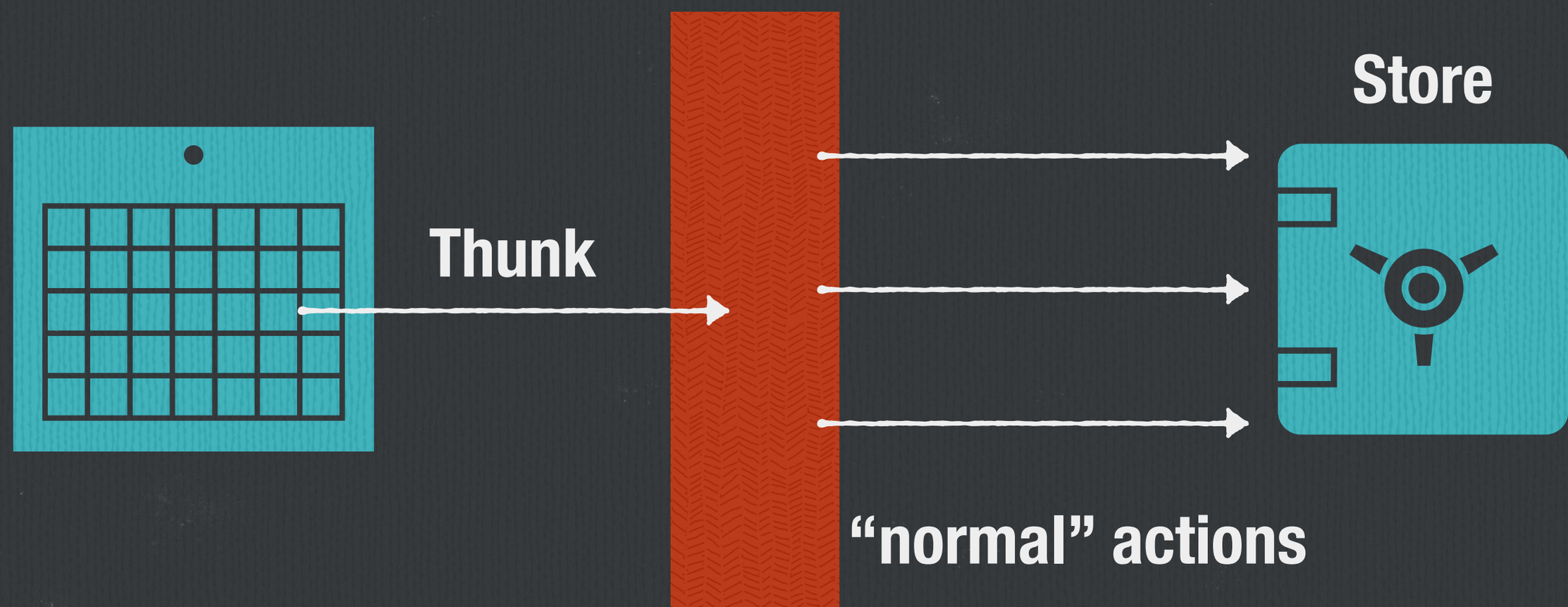
# Your Turn

- [ ] Convert above code into HOC
- [ ] Implement shouldComponentUpdate

# “Async” Actions

- All code in reducer is synchronized
- Async code has to go elsewhere

# Thunk Middleware



# thunk middleware

- dispatch(code)
- code performs async logic and dispatches normal actions

# Why

- ☐ Reducers are still sync
- ☐ Reducers are still easy to test
- ☐ Application logic is still easy to debug

# Why Not

- Actions are getting more complex
- Hard to reuse actions code

# Live Demo: Ajax Actions

- Using swapi.co
- We'll build an interface to select a person and get the list of films that person appeared in

# Takeaways

- ☐ Actions can use async/await
- ☐ Actions call other actions
- ☐ Actions represent sequences of events

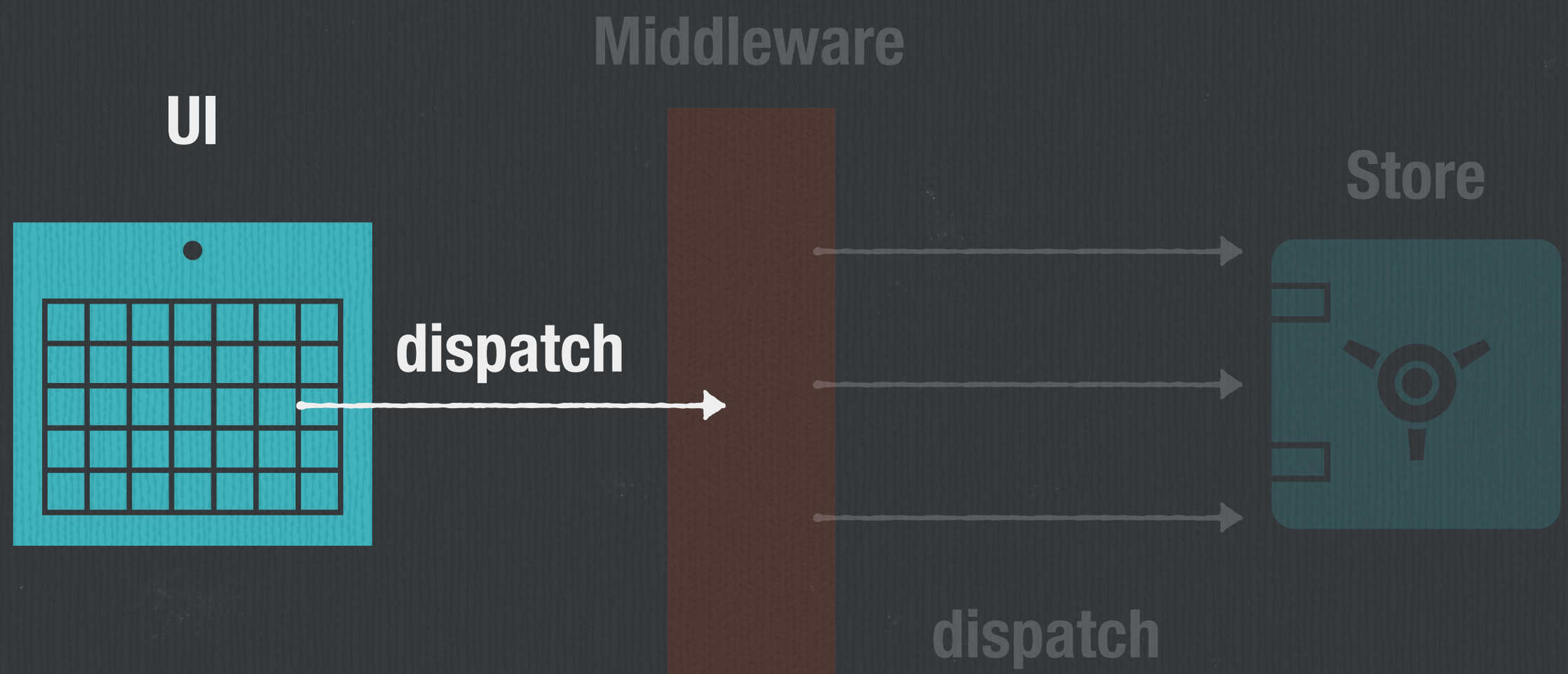
# React/Redux Tips

- ☐ Track render count
- ☐ Decide what should cause render
- ☐ Create components according to render scope

# Q & A

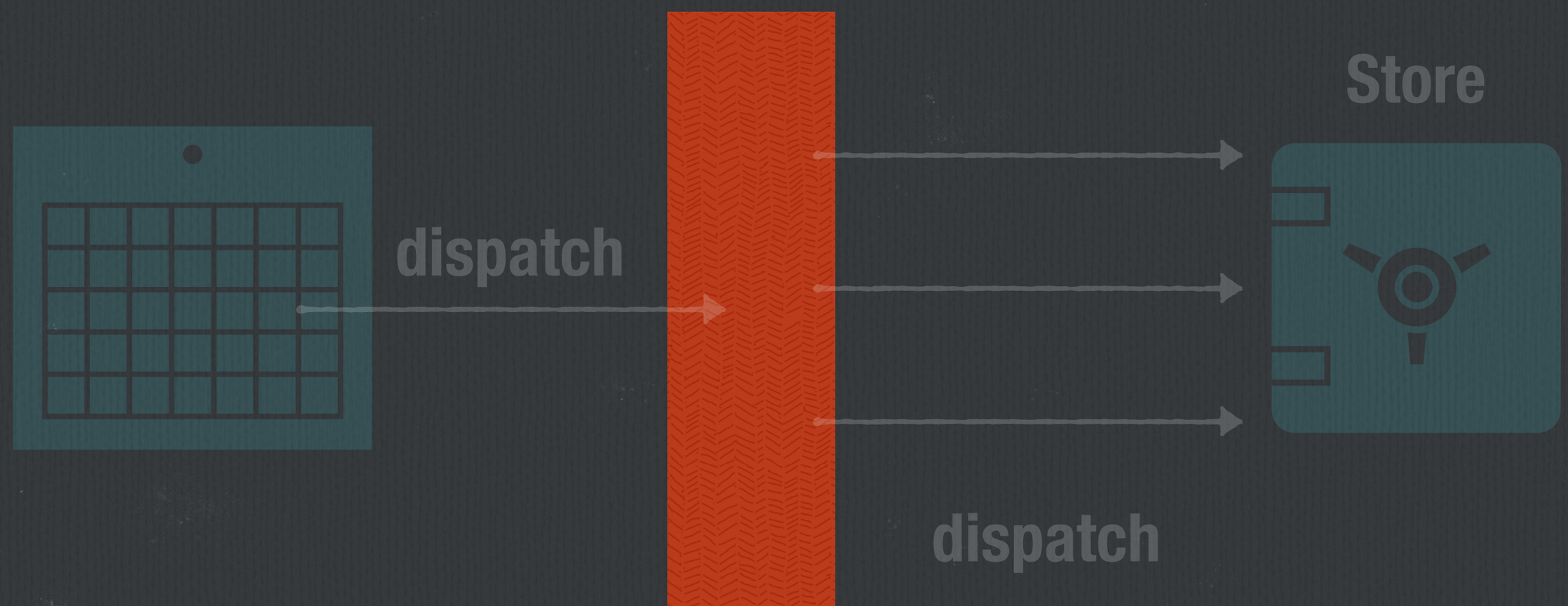
# Redux Middlewares

# #1

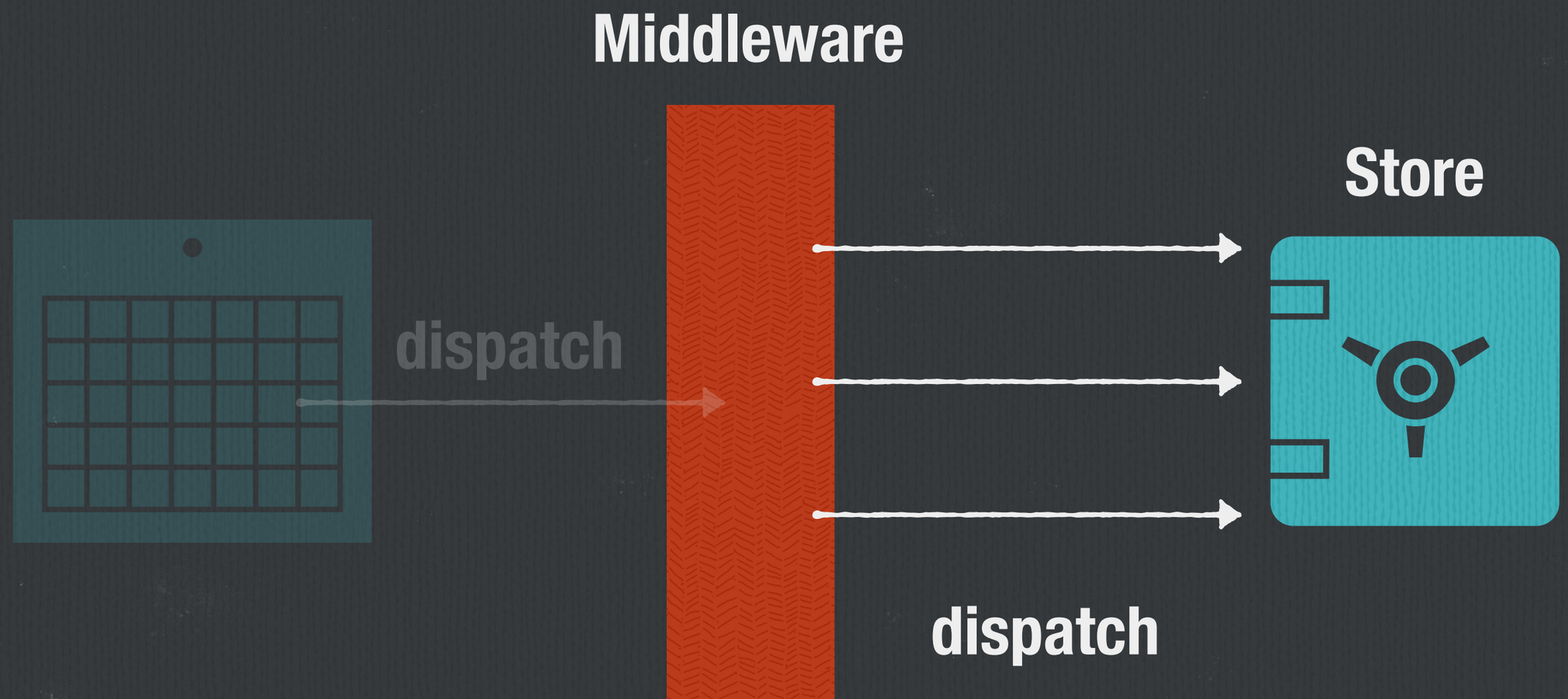


# #2

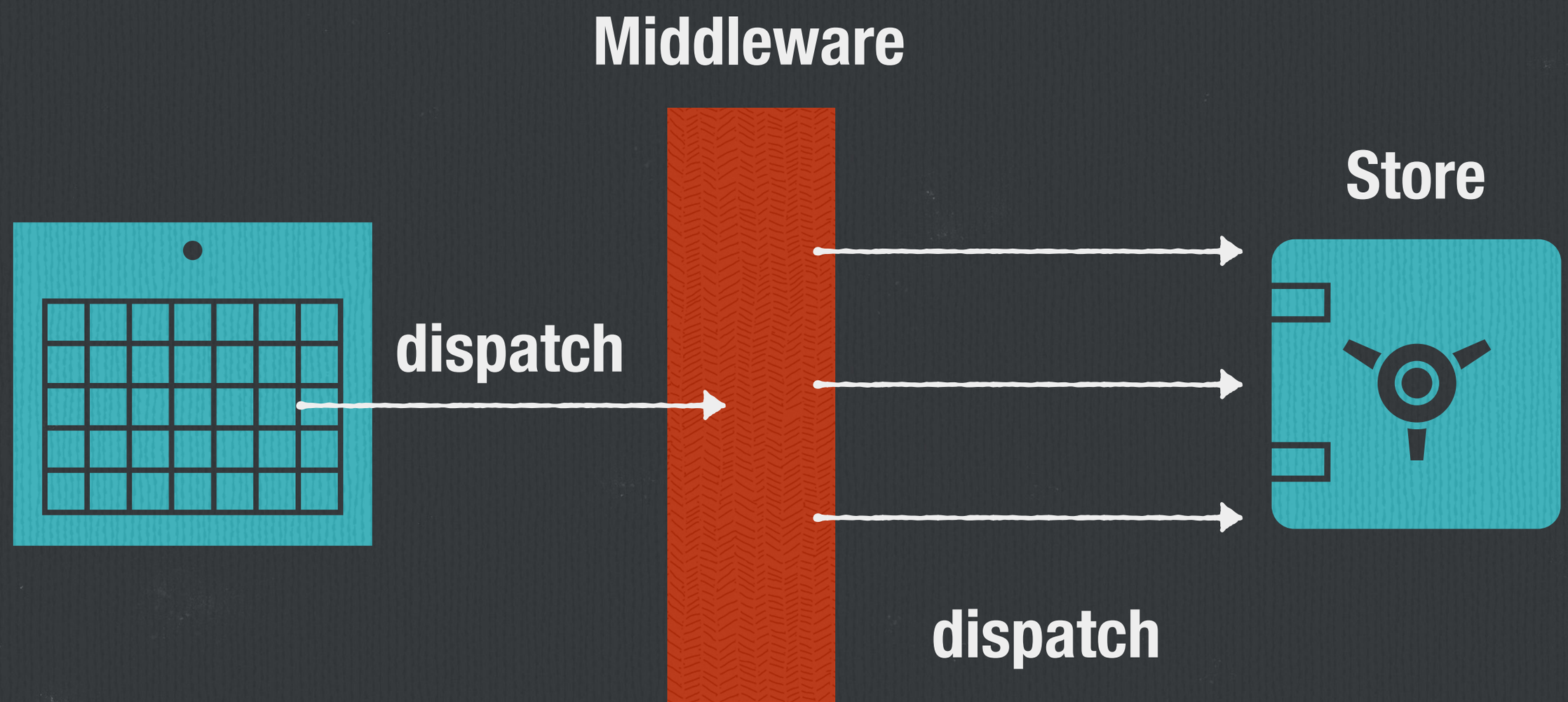
## Middleware

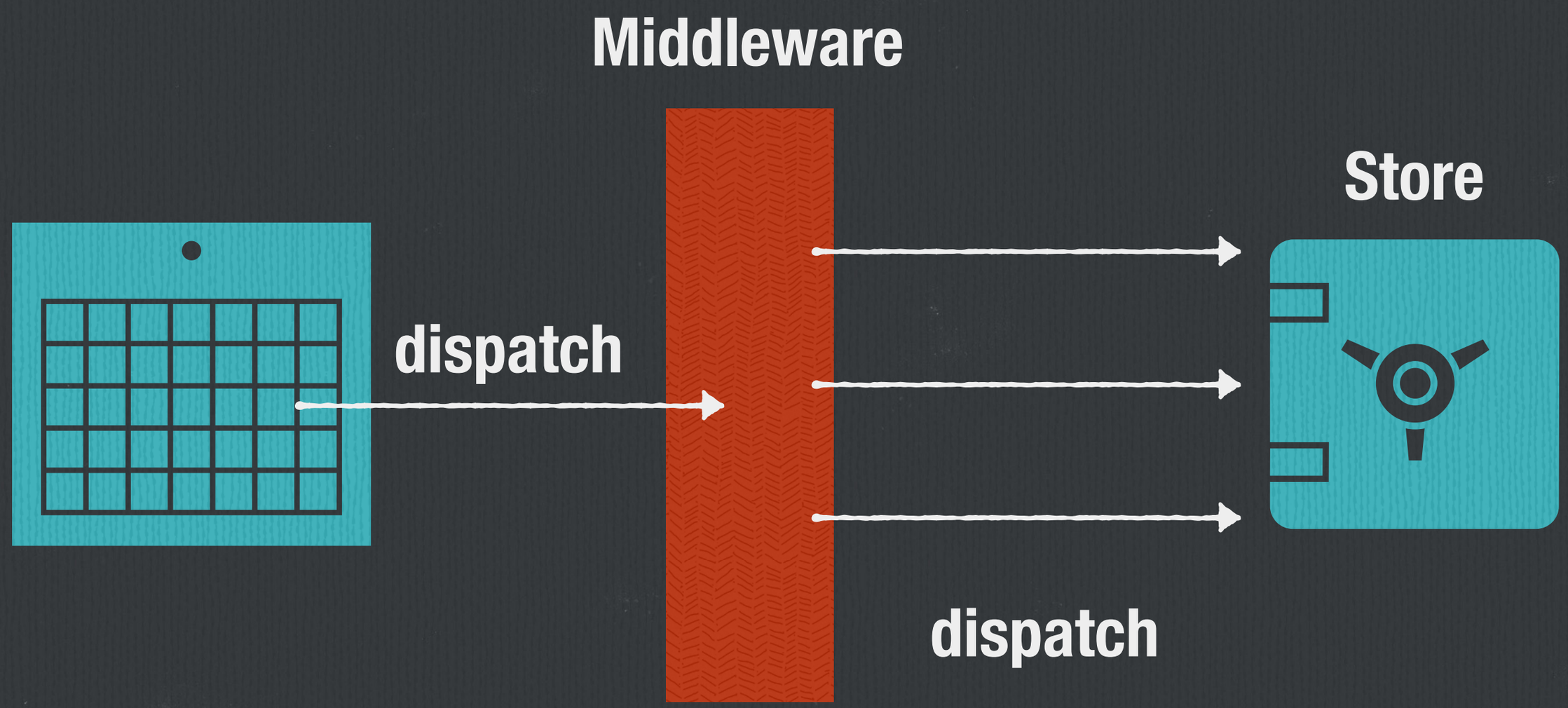


# #3



# What




Middleware
Store
dispatch
dispatch

# Why

- ☐ Thunks, Promises
- ☐ Logging, Debugging
- ☐ Undo/Redo
- ☐ Transactions

Middlewares provide “global” action handlers

# When

- “Share” common logic between thunks

# Hello World

```
function noop({ dispatch, getState }){
  return function(next){
    return function(action){
      // code to run BEFORE action is dispatched
      const res = next(action);

      // code to run AFTER action is dispatched
      // or alter return value
      return res;
    }
  }
}
```

# Installing Our Middleware

```
const store = createStore(reducer,
                          applyMiddleware(noop));
```

# Installing multiple middleware

```
createStore(reducer,
            applyMiddleware(
              noop,
              thunk,
              log
            ));
```

- Actions are handled from bottom to top
- log -> thunk -> noop -> real dispatch

# Live Demo: Log

- ☐ Let’s write a middleware that logs all actions
- ☐ Add that middleware to any existing redux project

# Our Own Thunk

```
export default function mythunk({ dispatch, getState }){
  return function(next){
    return function(action){

      if (typeof action === 'function') {
        return action(dispatch, getState);
      }
      return next(action);
    }
  }
}
```

# Testing Middlewares - setup

```
beforeEach(function() {
  this.fakeNext = jasmine.createSpy();
  this.mockStoreApi = jasmine.createSpyObj(
     'mockStoreApi', ['dispatch', 'getState']);
  const makeThunk = mythunk(this.mockStoreApi);
  this.thunk = makeThunk(this.fakeNext);
});
```

# Testing Middlewares

```
it('should call dispatched functions', function() {
  const spy = jasmine.createSpy();
  this.thunk(spy);


  expect(spy).toHaveBeenCalledOnce;
  expect(this.fakeNext).notToHaveBeenCalled;
});
```

# Undo middleware

```
export function undo({ dispatch, getState }){
  return function(next){
    const undoQueue = [];

    return function(action){
      if (action.type === '@@undo' && undoQueue.length > 0) {
        dispatch({ type: '@@setstate', payload: undoQueue.pop() });
        return action;
      } else if (action.type !== '@@setstate') {
        undoQueue.push(getState());
      }
      return next(action);
    }
  }
}
```

# Your Turn

- [ ] Finish undo middleware
- [ ] Write tests to verify it works

# Batches - redux-batch-middleware

- Middleware

```
export const type = '@@redux-batch-middleware/BATCH';

export const batch = ({ dispatch }) => {
    return (next) => (action) => {
        return Array.isArray(action)
            ? dispatch({ type: type, payload: action })
            : next(action);
    };
};
```

# Batches - redux-batch-middleware

- Reducer Enhancer

```
export const batching = (reducer) => {
    return function batcher(state, action) {
        return action.type === type
            ? action.payload.reduce(batcher, state)
            : reducer(state, action);
    };
};
```

# Q & A

# Lab #1

- ☐ Write a middleware called “persistent” that keeps state in local storage

# Lab #2

- Write a middleware called “lockable” that takes 2 actions:
  - lock - enter locked mode, in which no actions are processed, so UI is frozen
  - release - back to normal

# Lab #3

- ☐ Modify lockable middleware so it “plays back” all actions that happened while store was locked

# More Middleware Examples

- https://github.com/agraboso/redux-api-middleware
- https://github.com/mironov/react-redux-loading-bar
- https://github.com/joshwcomeau/redux-favicon
- https://github.com/AvraamMavridis/redux-perf-middleware

# Store Enhancer

# Why

- Modify “normal” dispatch() behavior
- Mainly - change notifications policy

# Redux Dispatch

```
currentState = currentReducer(currentState, action)

const listeners = currentListeners = nextListeners
for (let i = 0; i < listeners.length; i++) {
  const listener = listeners[i]
  listener()
}

return action
```

# What

- A store enhancer is a function that changes redux “createStore”

# How

```
export default function noopStoreEnhancer(next) {
  return (...args) => {
    const store = next(...args);

    return {
      ...store,
      store.dispatch,
      store.subscribe,
    };
  }
}
```

# Store Enhancers

- https://github.com/tappleby/redux-batched-subscribe
- https://github.com/toplan/redux-on
- https://github.com/tshelburne/redux-skip-by-action
-

# When

- ☐ Use store enhancer to modify notification policy
- ☐ Should not happen often

# Q & A

# Thanks For Listening


- Ynon Perek
- ynon@tocode.co.il