Enhanced AEK

Tools and recommendations for AEK 2.0

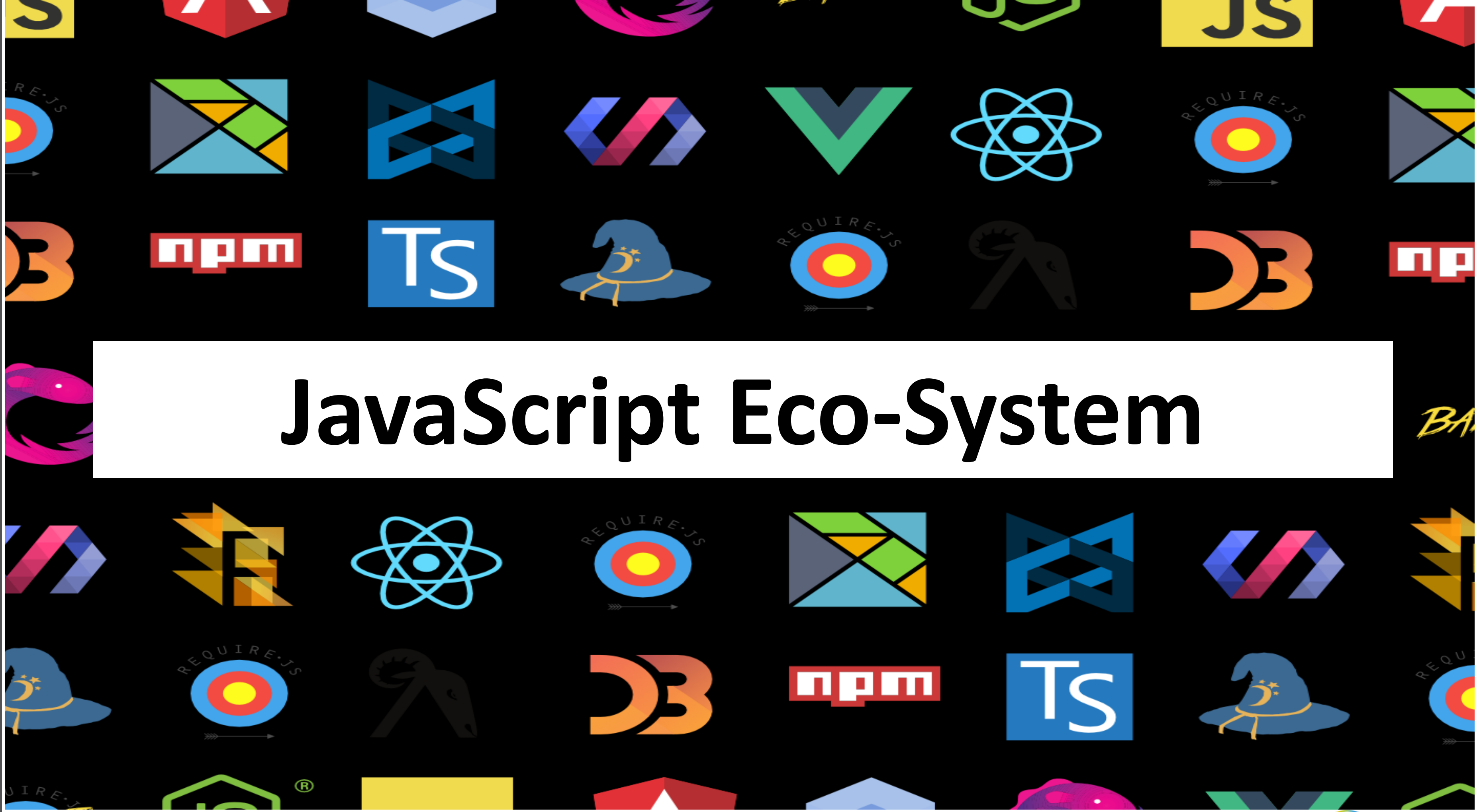
Liam Bennett – Mobile Solutions Architect

ExLibris a ProQuest Company

AEK is built on JavaScript

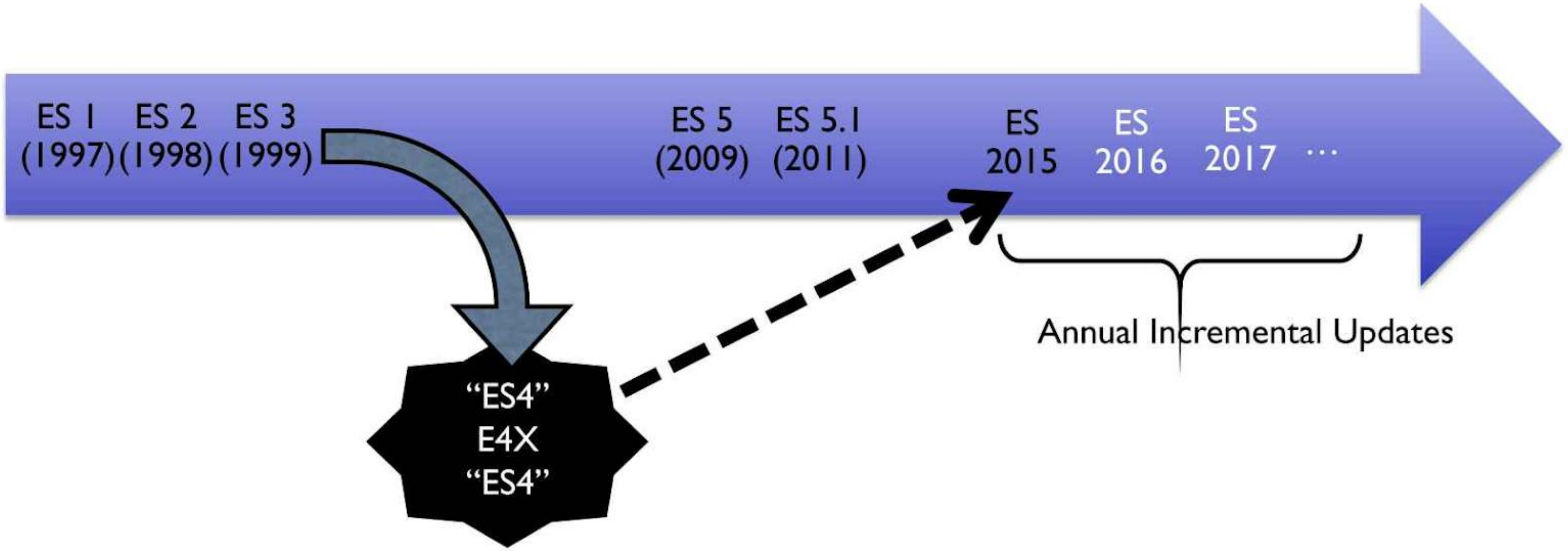
Learning JavaScript in 2018

https://hackernoon.com/how-it-feels-to-learn-javascript-in-2016-d3a717dd577f



JavaScript Eco-System

The ECMAScript Standard Timeline

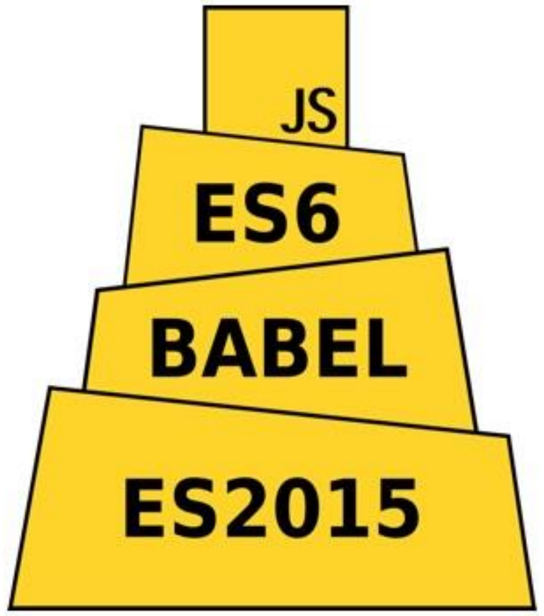


Ignore vs Embrace

ES6, ES7, ES8 and beyond

ES201x Features

Let/Const	let a = 1, const b = 2
Arrow functions	=>
Sets	new Set()
Maps	new Map()
Classes	Class HelloWorld {}
Destructuring	const [a, b] = [10, 20]
Promises	new Promise()
Modules	import/export
Object Rest / Spread Properties	…
Async functions	async/await
Template strings	\`Hello ${name}\`
Many more...	



AEK + Babel = no worries

Linting

Why?

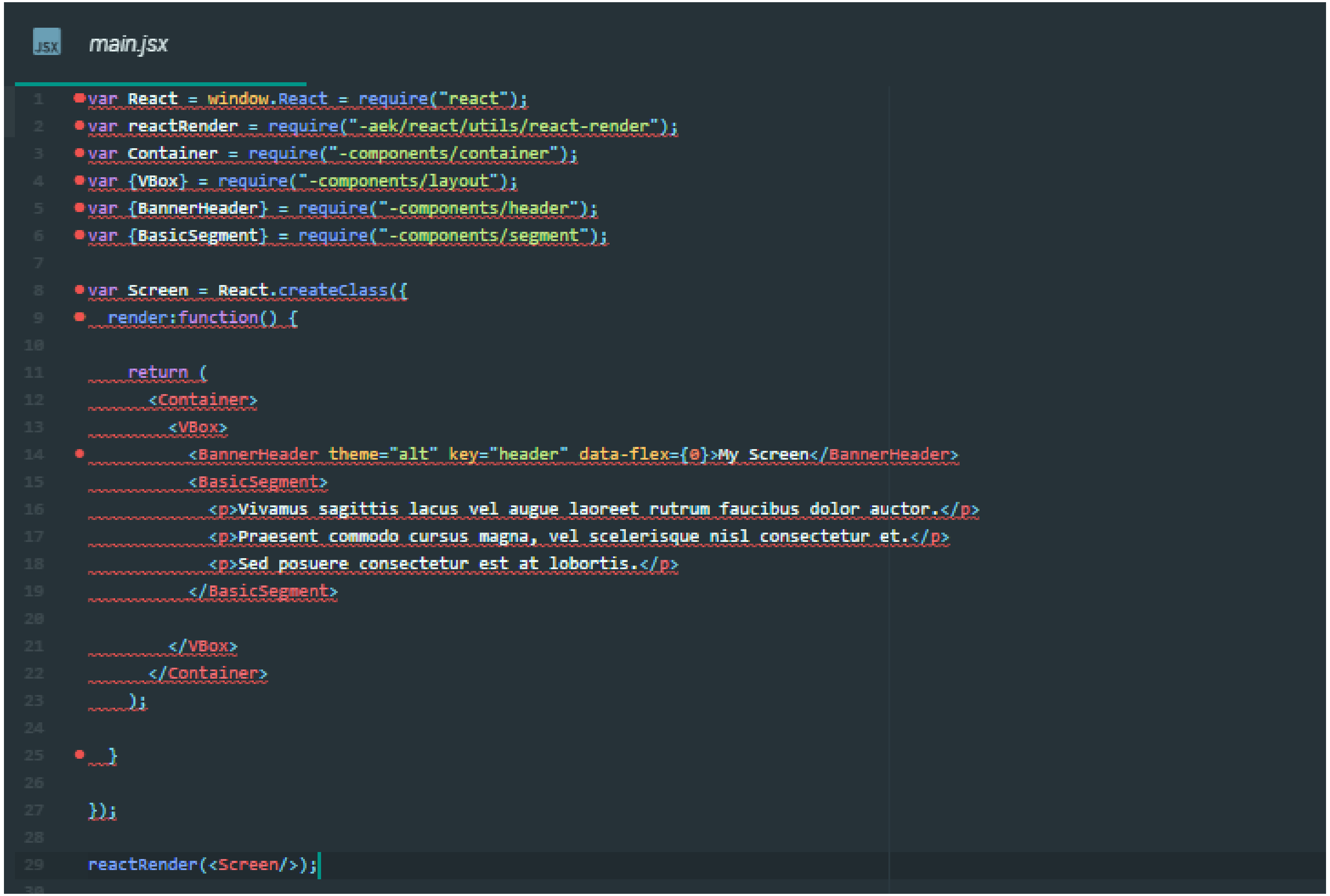
Analyse your code and warn you of potential errors. In order for it to work, you need to configure it with specific rules.

ESLint + AirBnB Config = :)

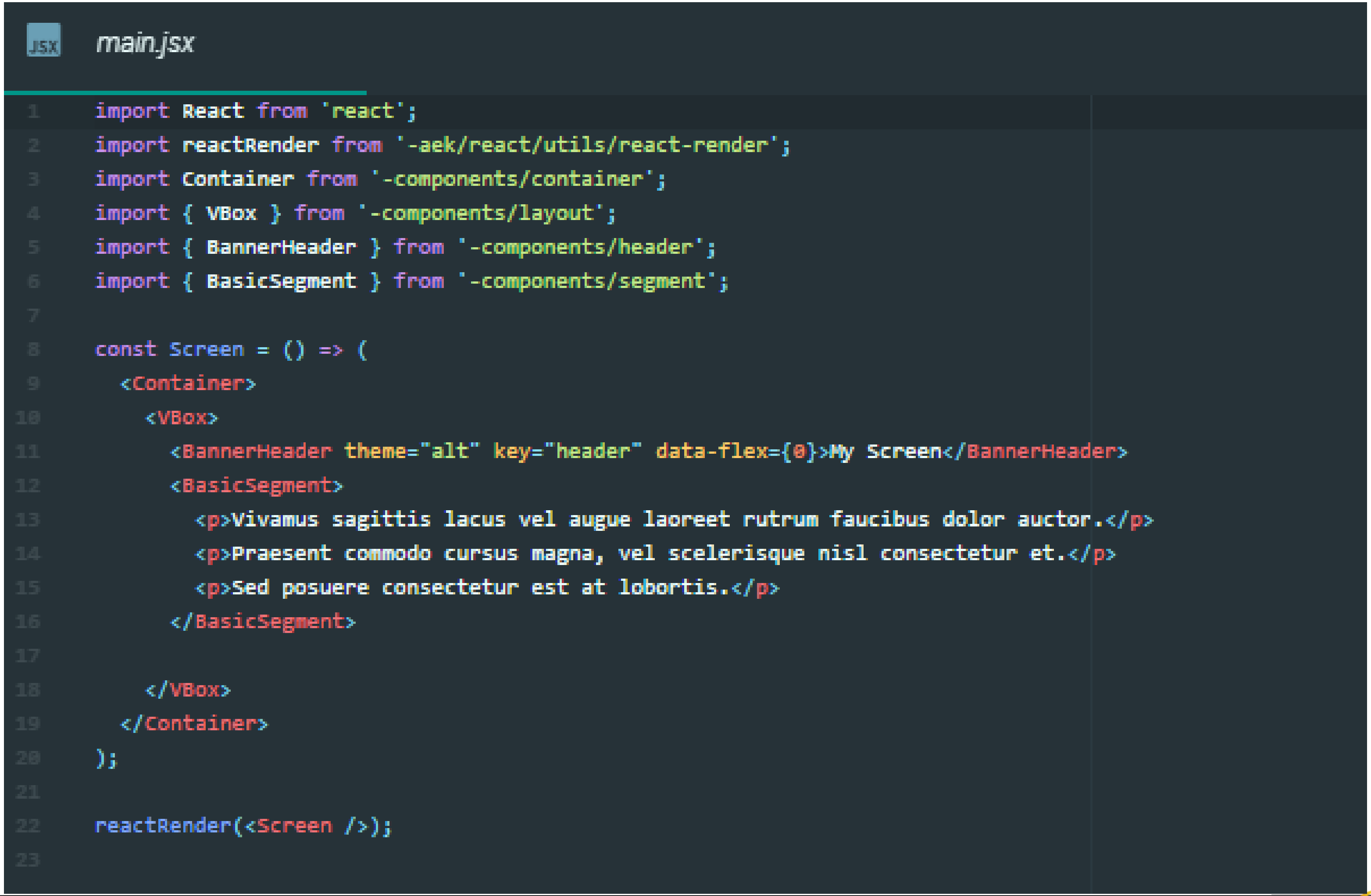
```
npm install --save-dev eslint-config-airbnb@^16.1.0 eslint@^4.15.0 eslint-plugin-jsx-a11y@^6.0.3 eslint-plugin-import@^2.8.0 eslint-plugin-react@^7.5.1 babel-eslint@^7.2.3
```

Add ‘extends: airbnb’ to your .eslintrc

Remove ‘rules’ from your .eslintrc


main.jsx

```
var React = window.React = require("react");
var reactRender = require("-aek/react/utils/react-render");
var Container = require("-components/container");
var {VBox} = require("-components/layout");
var {BannerHeader} = require("-components/header");
var {BasicSegment} = require("-components/segment");

var Screen = React.createClass({
  render:function() {

    return (
      <Container>
        <VBox>
          <BannerHeader theme="alt" key="header" data-flex={0}>My Screen</BannerHeader>
          <BasicSegment>
            <p>Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor.</p>
            <p>Praesent commodo cursus magna, vel scelerisque nisl consectetur et.</p>
            <p>Sed posuere consectetur est at lobortis.</p>
          </BasicSegment>

        </VBox>
      </Container>
    );

  }

});

reactRender(<Screen/>);
```



main.jsx

```
import React from 'react';
import reactRender from '-aek/react/utils/react-render';
import Container from '-components/container';
import { VBox } from '-components/layout';
import { BannerHeader } from '-components/header';
import { BasicSegment } from '-components/segment';

const Screen = () => (
  <Container>
    <VBox>
      <BannerHeader theme="alt" key="header" data-flex={0}>My Screen</BannerHeader>
      <BasicSegment>
        <p>Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor.</p>
        <p>Praesent commodo cursus magna, vel scelerisque nisl consectetur et.</p>
        <p>Sed posuere consectetur est at lobortis.</p>
      </BasicSegment>

    </VBox>
  </Container>
);

reactRender(<Screen />);

```

Upgrade and go

Lebab - https://lebab.io/try-it

ESlint - eslint --fix

React 15/16/17

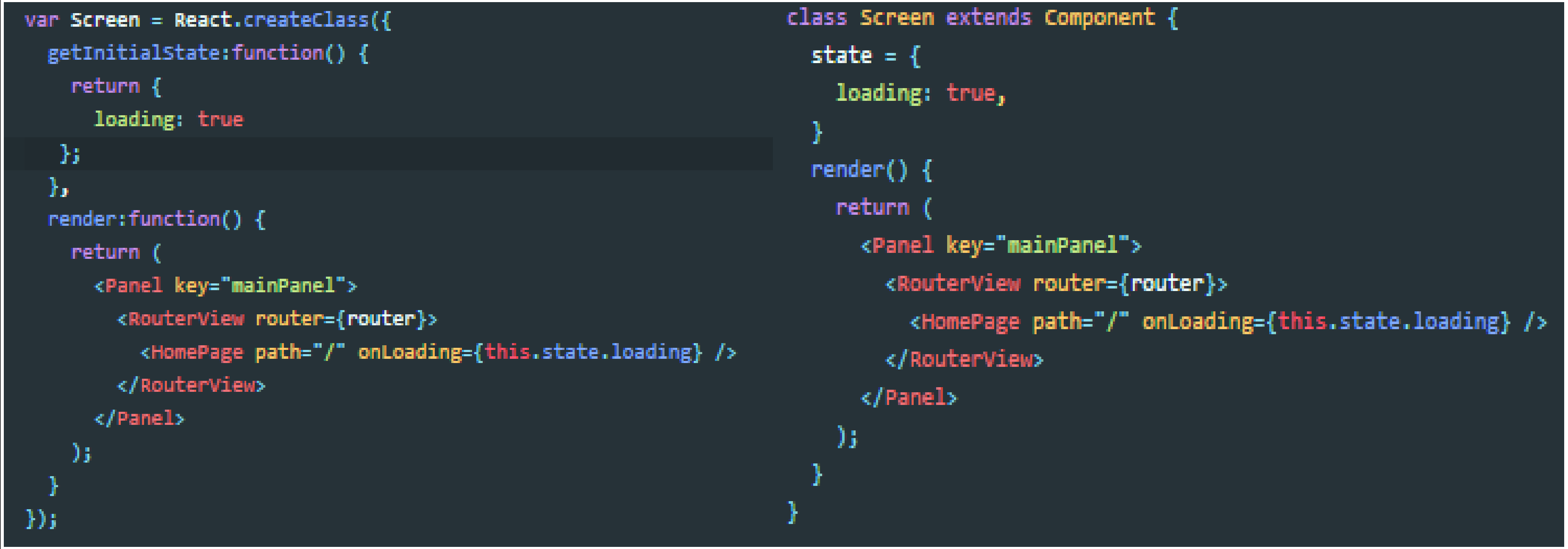
Why upgrade?

Peformance, reduce noise/bloat, enforce best practise, easy to test...

Why? - Continual deprecation.

React.createClass and React.PropTypes removed in React 16.

React.createClass vs Class



```
var Screen = React.createClass({
  getInitialState:function() {
    return {
      loading: true
    };
  },
  render:function() {
    return (
      <Panel key="mainPanel">
        <RouterView router={router}>
          <HomePage path="/" onLoading={this.state.loading} />
        </RouterView>
      </Panel>
    );
  }
});
```

```
class Screen extends Component {
  state = {
    loading: true,
  }
  render() {
    return (
      <Panel key="mainPanel">
        <RouterView router={router}>
          <HomePage path="/" onLoading={this.state.loading} />
        </RouterView>
      </Panel>
    );
  }
}
```

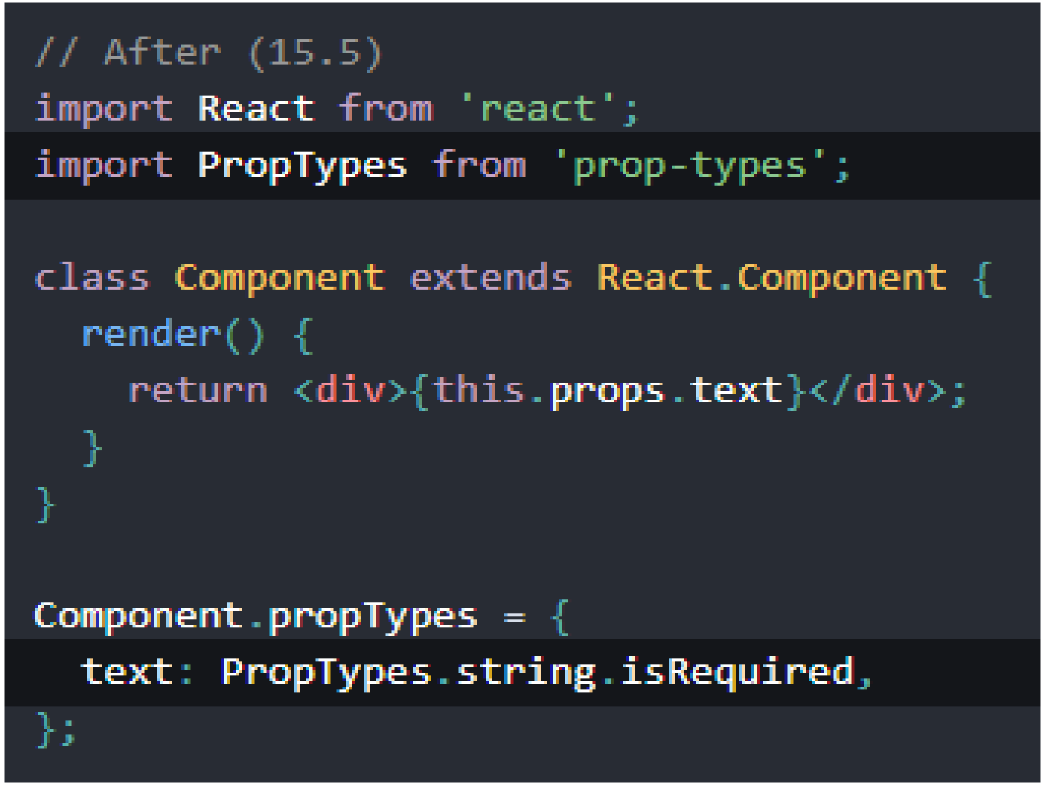
Always use PropTypes - Why?

Catch bugs by validating data types.

Flag props as mandatory or set default values.

They provide a great benefit with little effort.

React.PropTypes



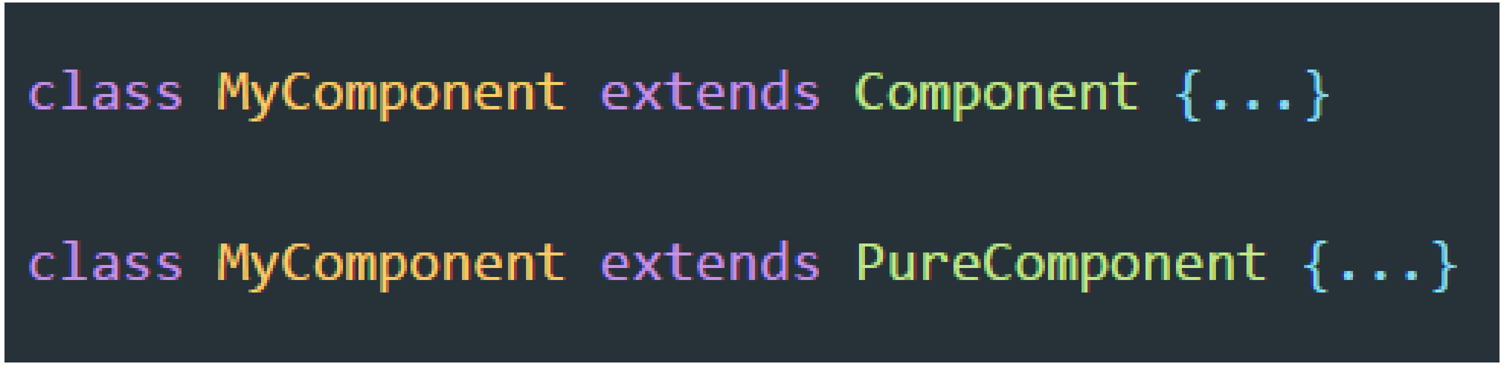
```
// After (15.5)
import React from 'react';
import PropTypes from 'prop-types';

class Component extends React.Component {
  render() {
    return <div>{this.props.text}</div>;
  }
}

Component.propTypes = {
  text: PropTypes.string.isRequired,
};
```

React.PureComponent?

React.PureComponent is one of the most significant ways to optimize React applications that is easy and fast to implement.

PureComponent changes the life-cycle method shouldComponentUpdate to automatically check whether a re-render is required for the component.

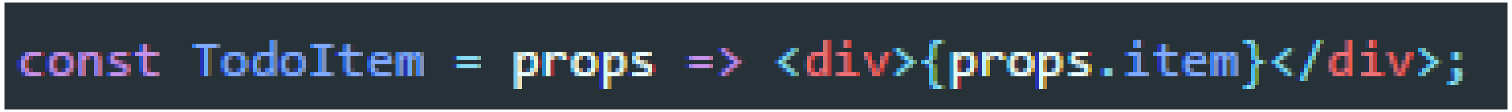
This allows a PureComponent to call render only if it detects changes in state or props.



```
class MyComponent extends Component {...}

class MyComponent extends PureComponent {...}
```

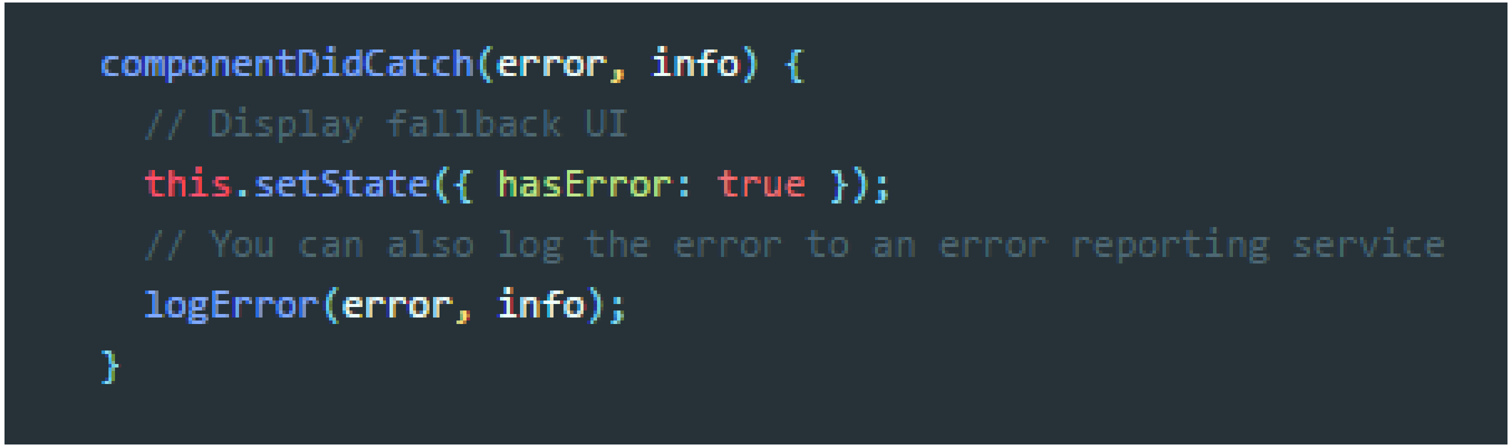
Additional

Write Stateless Functional Components



```
const TodoItem = props => <div>{props.item}</div>;
```

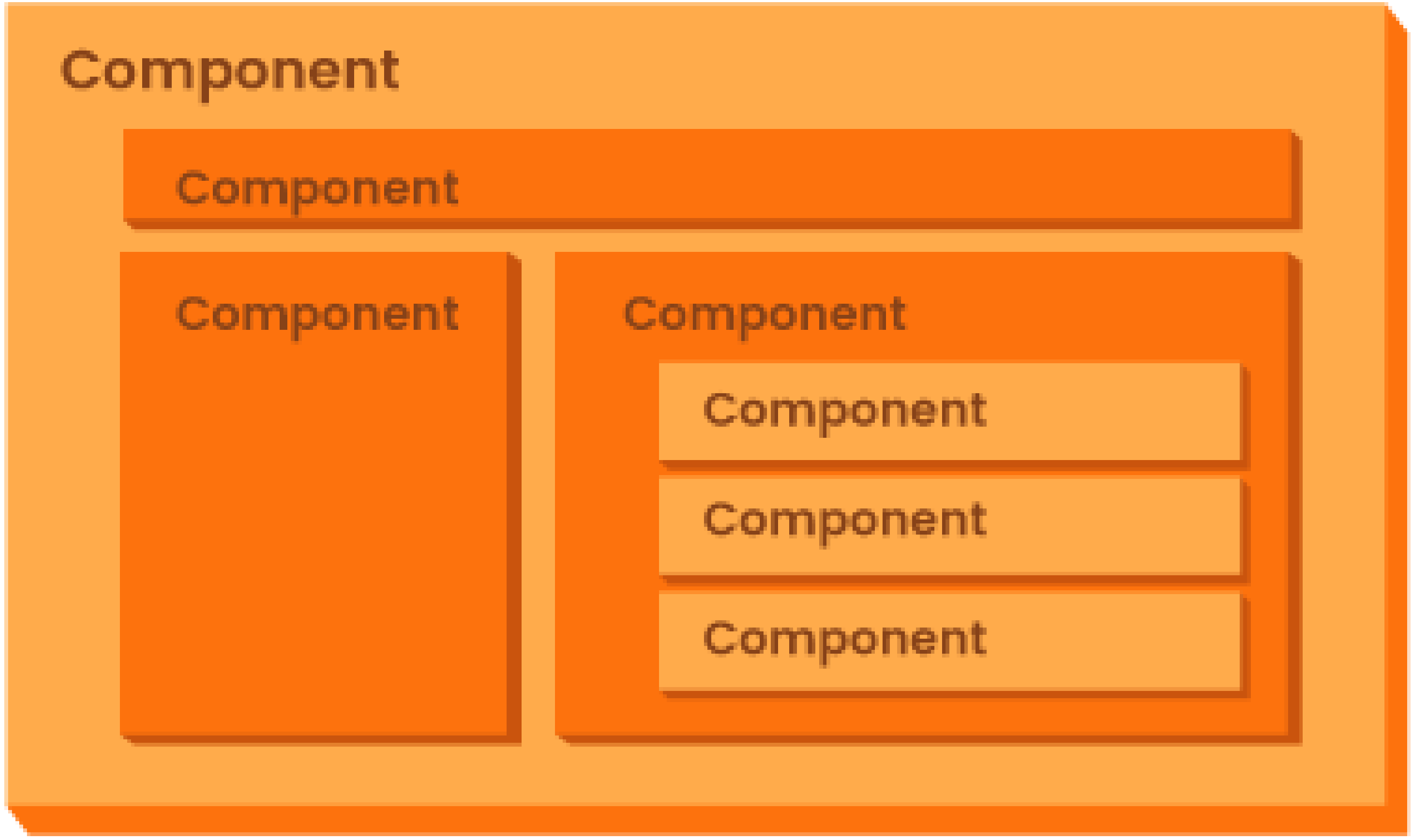
Error Boundaries

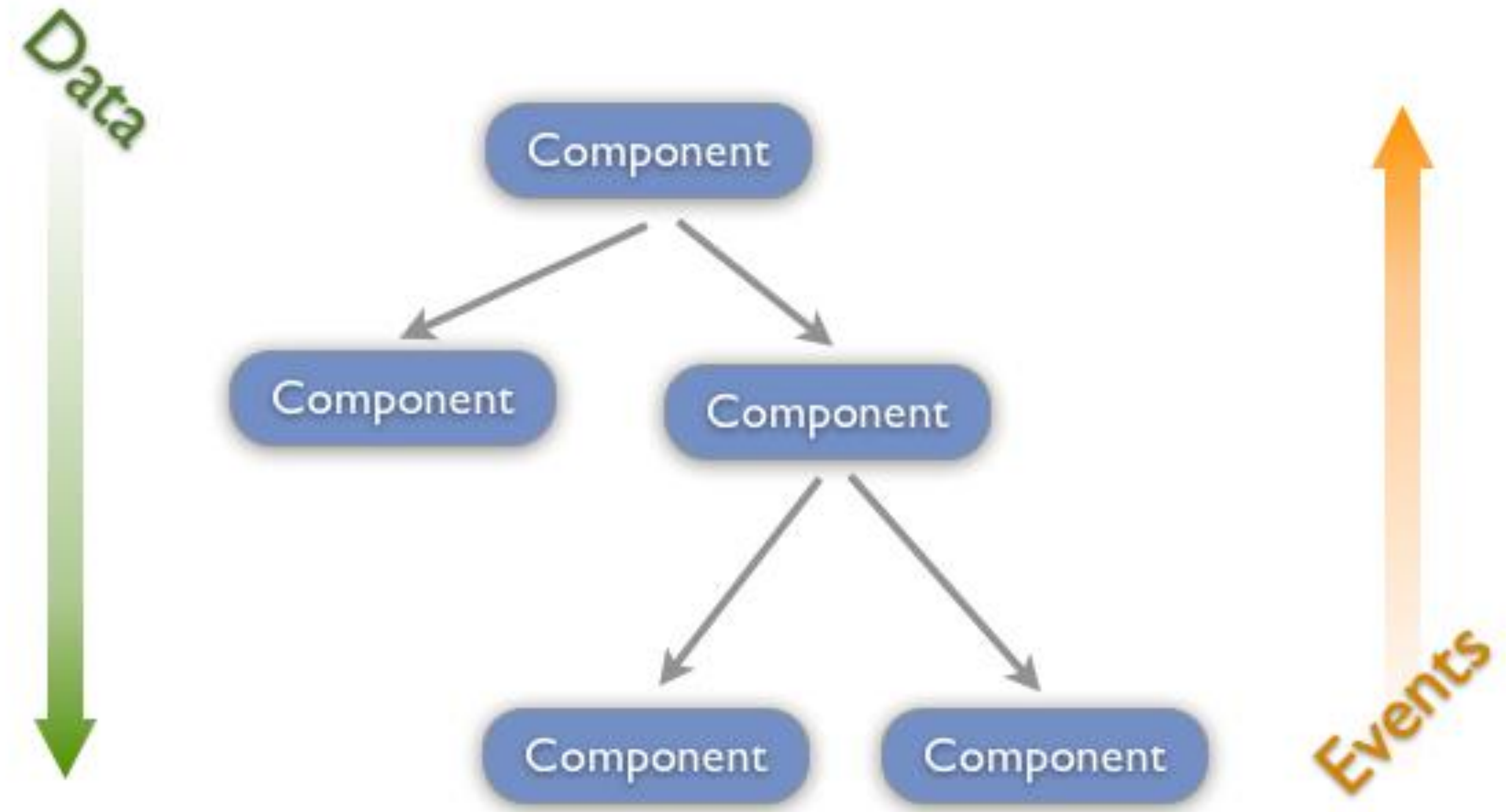


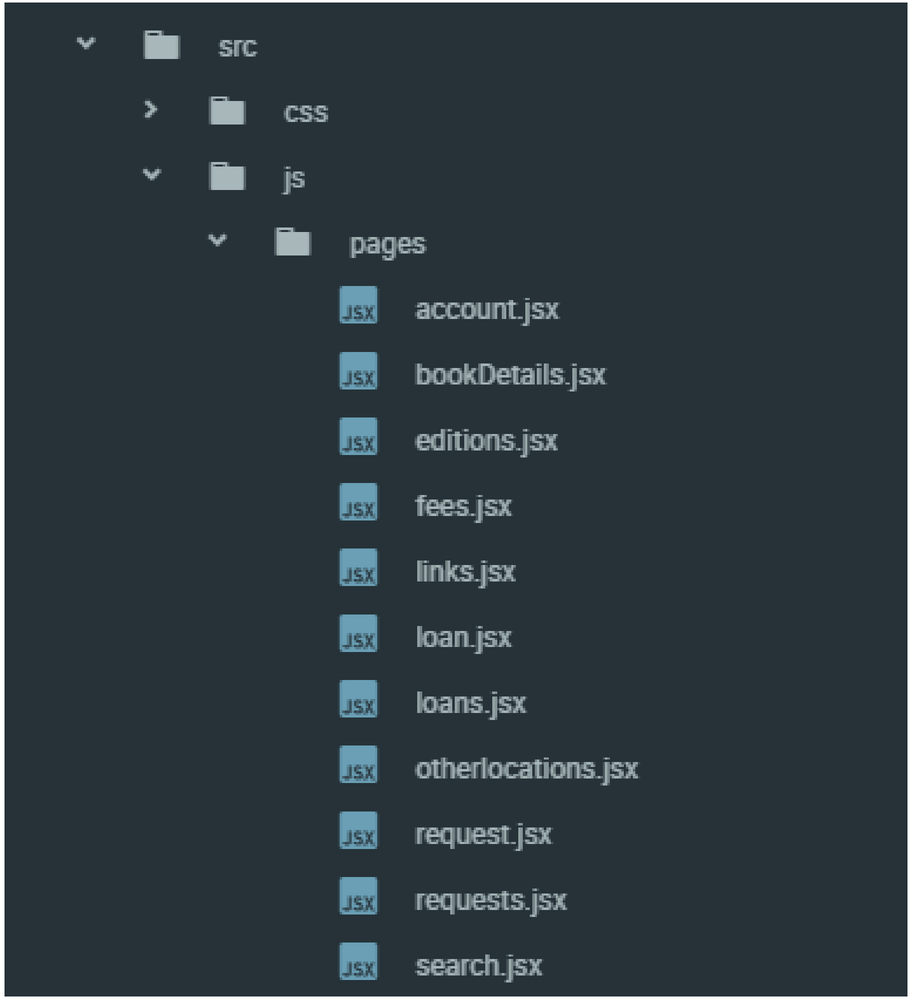
```
  componentDidCatch(error, info) {
    // Display fallback UI
    this.setState({ hasError: true });
    // You can also log the error to an error reporting service
    logError(error, info);
  }
```

App Structure

Component Based Architecture


Component
Component
Component
Component
Component
Component
Component


Data
Component
Component
Component
Component
Component
Events



```
src
  css
  js
    pages
      account.jsx
      bookDetails.jsx
      editions.jsx
      fees.jsx
      links.jsx
      loan.jsx
      loans.jsx
      otherlocations.jsx
      request.jsx
      requests.jsx
      search.jsx
```

Presentational and Container Components

Separate the behaviour from the presentation

Presentational Components

Presentational components are coupled with the view or how things look.

These components accept props from their **container** counterpart and render them.

Everything that has to do with describing the UI should go here.

Container Components

A container component tells the **presentational** component what should be rendered using props.

It shouldn't contain limited DOM mark-ups and styles. This is the place where you should place your API calls and store the result into the component's state.

Additional

Container vs Presentational Components

https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0

Folder Organization

https://medium.com/styled-components/component-folder-pattern-ee42df37ec68

Naming Conventions

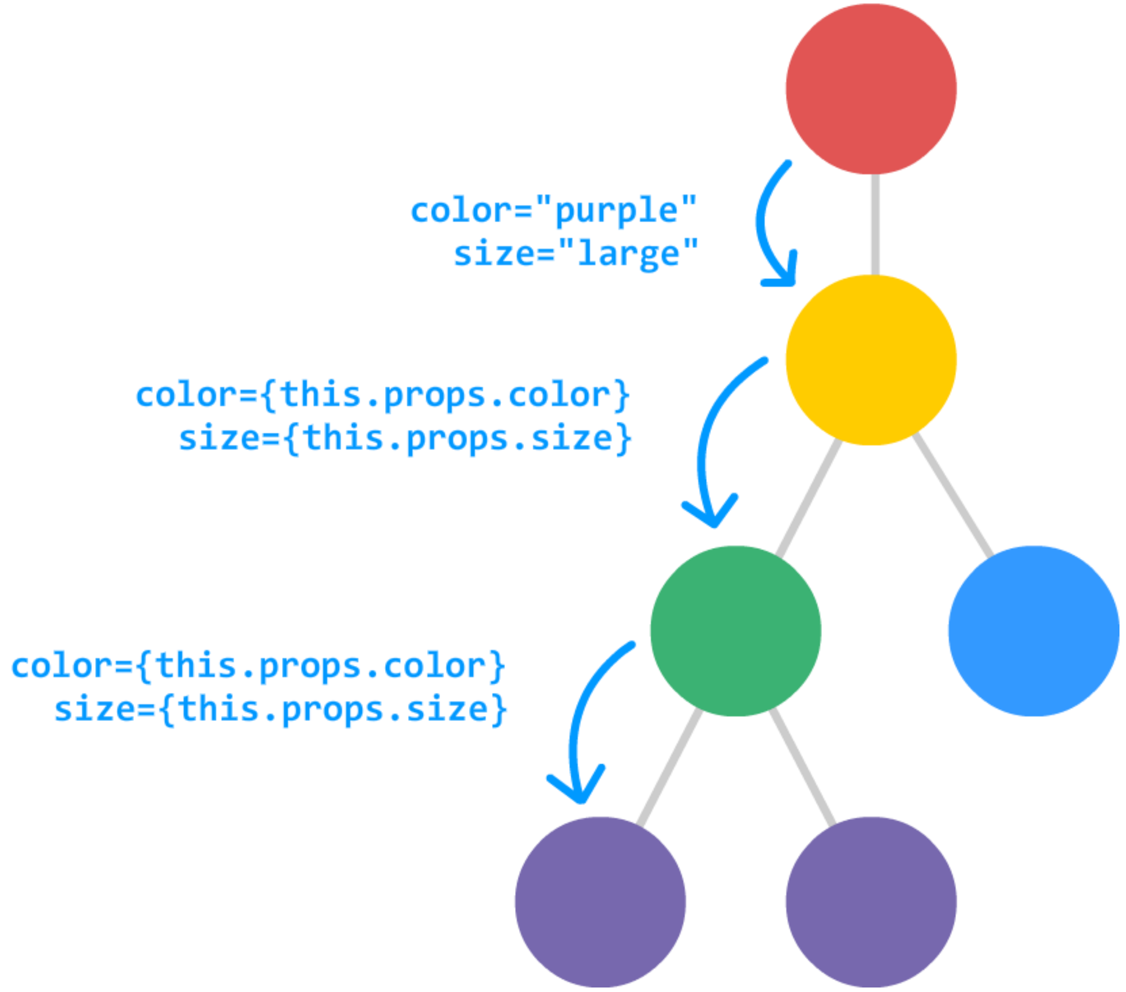
https://hackernoon.com/the-100-correct-way-to-structure-a-react-app-or-why-theres-no-such-thing-3ede534ef1ed

State Management

ExLibris
a ProQuest Company

Managing state is <u>hard</u>

Avoid prop hell


color="purple"
size="large"
color={this.props.color}
size={this.props.size}
color={this.props.color}
size={this.props.size}

Need a simple way of managing state

Redux

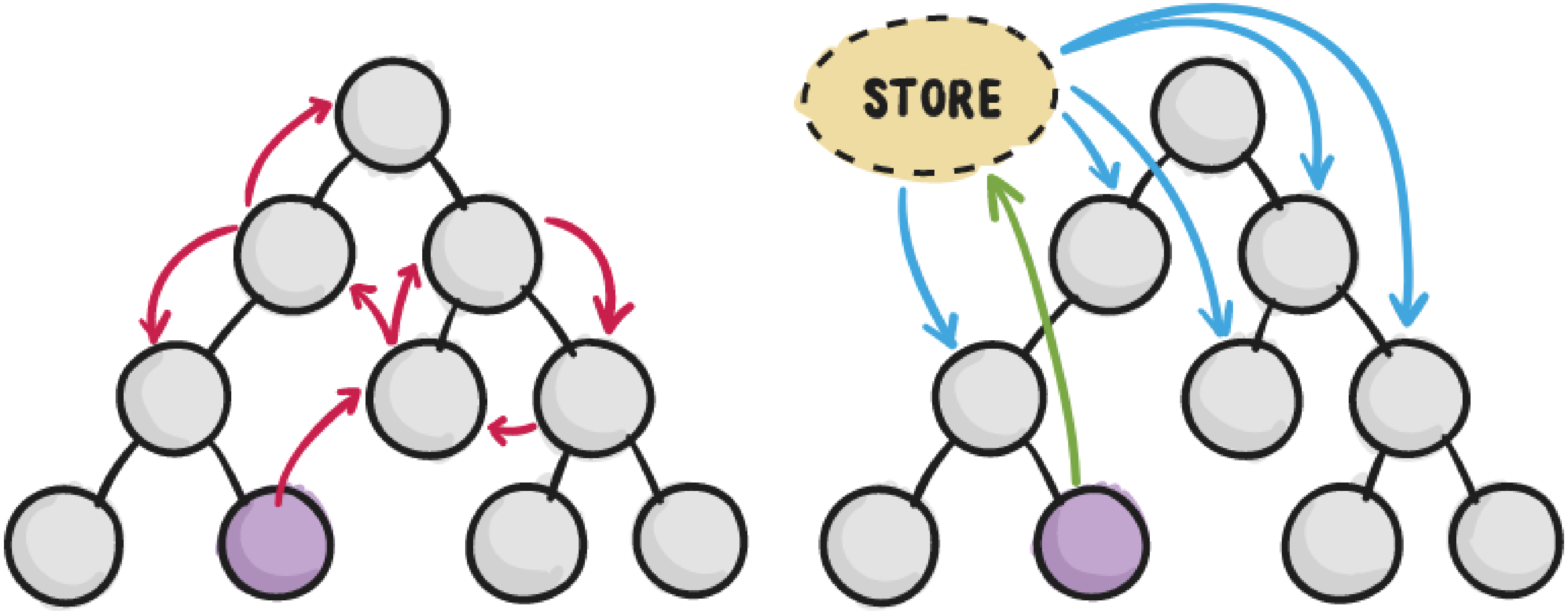
a predictable state container for JavaScript apps

What is Redux?

Redux introduces a <u>central data store </u>in an application.

The store contains the state of the application and is the <u>source of truth for components</u>.

By using the store concept you <u>do not need to synchronize</u> state between components manually.



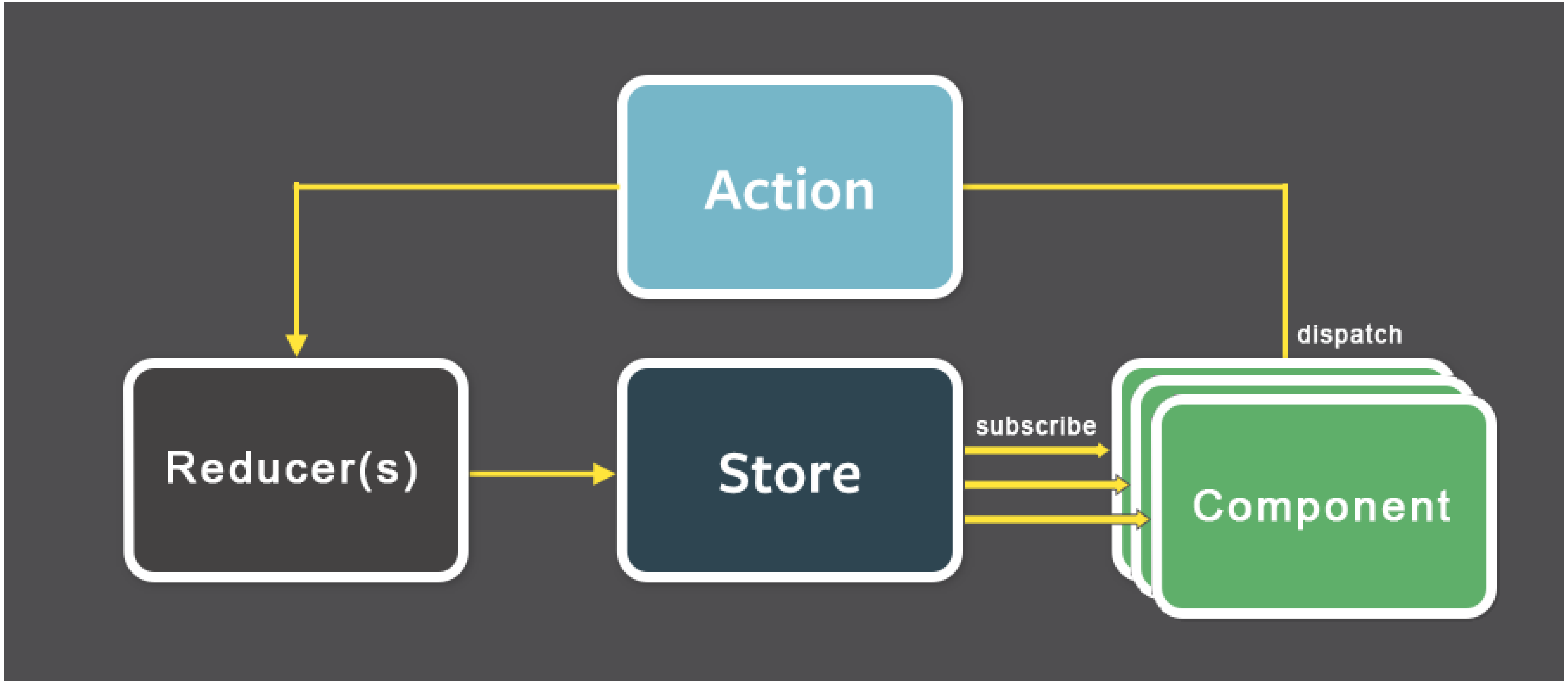
● COMPONENT INITIATING CHANGE

Redux has three main parts:

Actions: Send information from the application to the store.

Reducers: Take the current application state and an action object and return a new application state.

Store: Is the central objects that holds the state of the application.


Action
dispatch
Reducer(s)
Store
subscribe
Component

Why Redux?

Predictable State: Same actions, same order, same state

Seperation of Concerns: Seperate View component from Logic/Async flows

Decoupling: Avoid prop hell, props through store not from parent

Testability: Easier to test actions and reducers seperarely

Additional

Handle Side Effects (Async)

- Redux Thunk - (Delayed) Functions
- Redux Promise - Promises
- Redux Saga - Generators
- Redux Observable - Observables
- Redux Loop - Elm Effects

Component Library

”As the number of apps grew, we found ourselves writing the same code and components for each app. This is not ideal because we had to spend time writing similar code, and it was not centralized, so each variation might have slight differences.”

-Some AEK Developer

More code reuse

Less duplication

Creating a NPM module?

Turn your React components into modules to achieve code reusability.

Where to publish?

Public npm registry?

v

Ex Libris private registry?

Great!

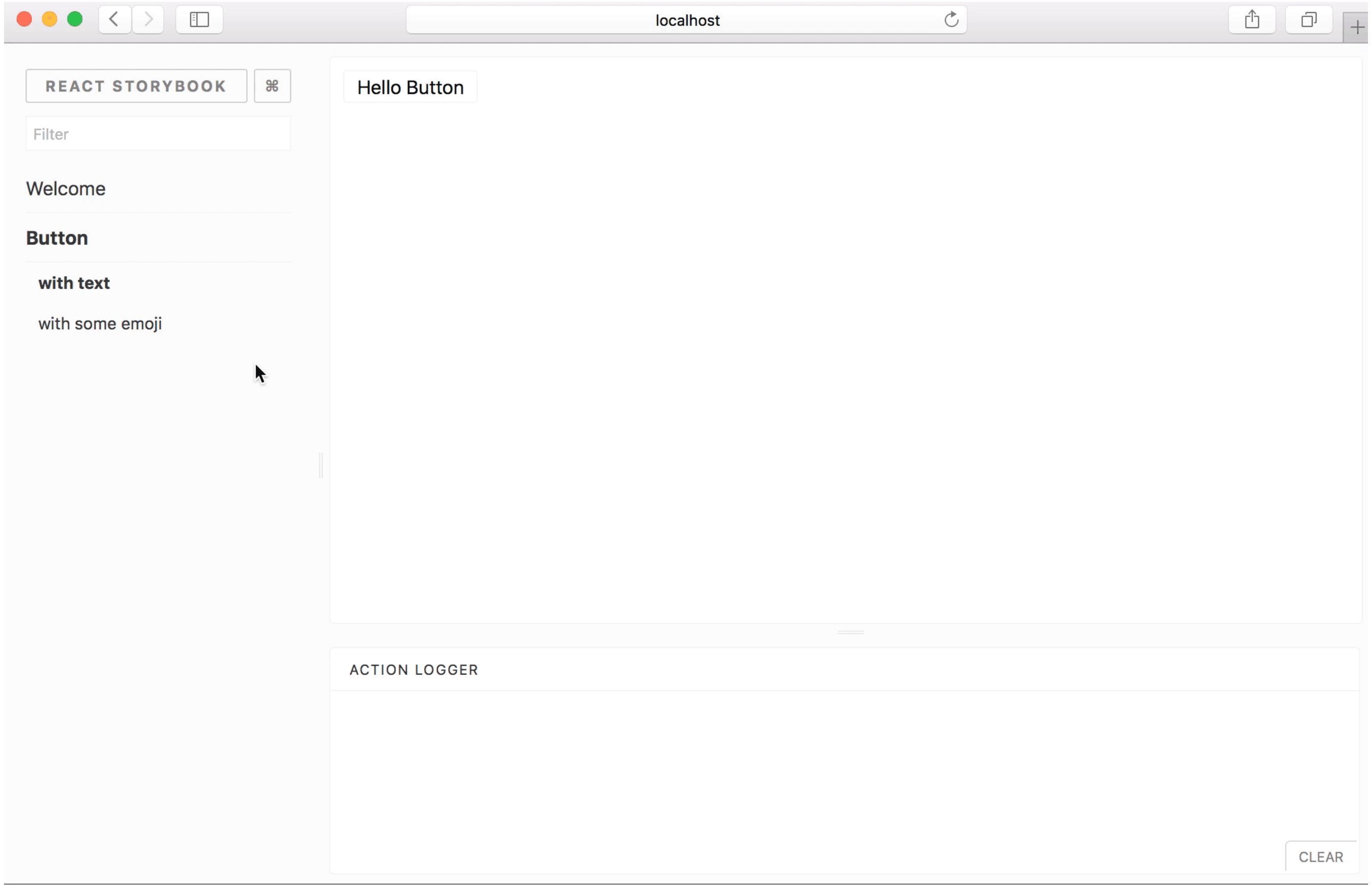
…but, publishing our modules to an NPM registry is a slow process. Not very productive. We need some way to be able to showcase our components in real time…

Storybook to the rescue?

https://storybook.js.org/

Storybook allows us to speed up the development of our components by adding an UI environment.

Storybook provides a fancy web UI, with hot-reloading and several other plugins that help us build faster better components.


localhost
REACT STORYBOOK
⌘
Filter
Welcome
Button
with text
with some emoji
Hello Button
ACTION LOGGER
CLEAR

Great, we have our library..

Monorepo vs Manyrepo?

How do you effectively share front-end components across various AEK projects within an institution?

Lerna to the rescue?

https://lernajs.io/

Lerna is a tool that optimizes the workflow around managing multi-package repositories with git and npm

Lerna lets you construct your application into a very large repository of packages and apps (can all be renamed and configured); these apps are the consumers and the packages are the granular dependencies

Thank you

ExLibris a ProQuest Company