Things I've Done Wrong in React.js

Josh Doro

Who Am I?

- Technical Architect with nvisia
- Writing code professionally for 10 years
- Working with React.js for the last 4
- Quiet

nvisia
connect. build. enable.

About nvisia

- Software consulting firm
- Offices in Chicago, Milwaukee, and Madison
- We write code for clients

nvisia
connect. build. enable.

Bad Code

When you finally catch the person that's been writing bad code all the time

It's just me, myself and I

nvisia
connect. build. enable.

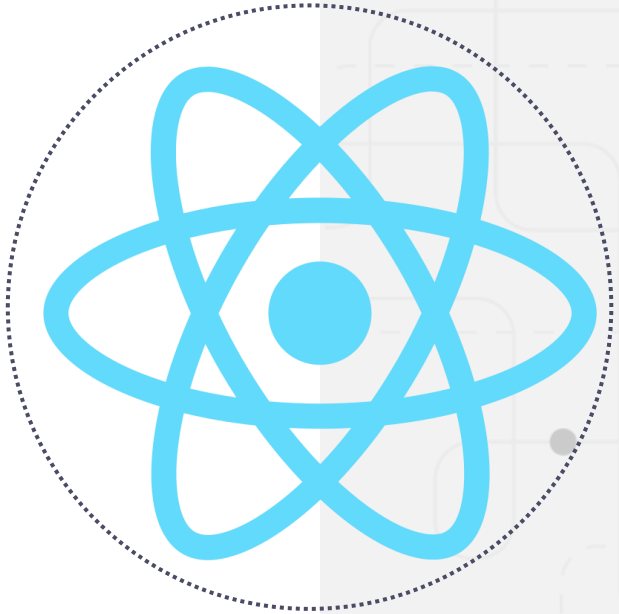
Warning!

I Tried to Learn React and Redux at the Same Time

nvisia
connect. build. enable.

React.js

- Javascript library
- Components
- State and props
- Components update when Data changes
- Immutability



Redux

- State management library
- Centralized
- Predictable
- Not exclusive to React

3 parts

- React - presentation
- Redux - state
- React-Redux (connect)

Giant Page Connects

```
const mapStateToProps = (state, props) => {
    const edmSecurityId = _.get(props, 'params.securityId');
    const pmvInputs = getPmvInputsForEdmSecurityId(state, edmSecurityId);

    return {
        pmvInputs,
        ...state.userAttributesReducer,
        ...state.appReducer,
        ...state.pitchPacketReducer,
        ...state.securitiesReducer,
        ...state.userPermissionsReducer,
        ...state.featureToggleReducer,
        isMobile: getIsMobile(state),
    }
};

export default connect(mapStateToProps,
Object.assign({}, { isFeatureEnabled, createNoteRequest, createNoteSuccess,
getMostRecentAnalysis, getRelatedSecurities, getSecurityAttributes, getSecurityById,
getSecurityMetrics, getSecurityPDF, saveSecurityScenario}, { getMostRecentPitchPacket,
savePitchPacket }, appAlertActions, searchActions, { saveNote }, {goBack, push}))
(withRouter(SecurityView));
```

nvisia
connect. build. enable.

How to improve

- Connected more components
- Utilized redux middleware (thunks or sagas)
- Adopted a selector pattern
- Pulled as much logic out of react as we could

My Confessions

I Tried to Learn React and Redux at the Same Time

I didn't separate presentation and state

State

Presentation

```
export class TogetherComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      value: '',
    };
    this.handleValueChange = this.handleValueChange.bind(this);
  }

  handleValueChange(event) {
    this.setState({value: event.target.value});
  }

  render() {
    const {value} = this.state;
    const style = {color: value.length > 5 ? 'red': 'black'};
    return (
      <div>
        Together: 
        <input
          style={style}
          type="text"
          value={value}
          onChange={this.handleValueChange} />
      </div>
    );
  }
}
```

nvisia
connect. build. enable.

tests

```
it("initial render", () => {
  const container = shallow(<TogetherComponent />);
  expect(container.html()).toBe(`<div>Together:<input type="text" style="color:black"
  value=""/></div>`);
});

it("redText render", () => {
  const container = shallow(<TogetherComponent />);
  container.setState({value: 'tester'});
  expect(container.html()).toBe(`<div>Together:<input type="text" style="color:red"
  value="tester"/></div>`);
});

it("handleValueChange", () => {
  const container = shallow(<TogetherComponent />);
  container.instance().handleValueChange({target:{value:'tester'}});
  expect(container.state('value')).toBe('tester');
});
```

```
export class StateComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      value: '',
      color: 'black'
    };
    this.handleValueChange = this.handleValueChange.bind(this);
  }

  handleValueChange(event) {
    const value = event.target.value;
    this.setState({
      value: value,
      color: value.length > 5 ? 'red': 'black'
    });
  }

  render() {
    const {color, value} = this.state;
    return (
      <ViewComponent
        label="Separated: "
        value={value}
        color={color}
        handleValueChange={this.handleValueChange}
      />
    );
  }
}
```

```
export function ViewComponent({label, value, color, handleValueChange}) {
  const style = {color: color};
  return (
    <div>
      {label}
      <input
        style={style}
        type="text"
        value={value}
        onChange={handleValueChange} />
    </div>
  );
}
```

nvisia
connect. build. enable.

State swap examples

```
const mapStateToProps = (state) => ({
  label: 'Connected: ',
  value: selectValue(state),
  color: selectColor(state)
});

const mapDispatchToProps = {
  handleValueChange
};

export const ConnectedComponent =  connect(mapStateToProps, mapDispatchToProps)(ViewComponent);
```

```
const COLORS = ['black', 'red', 'blue', 'green', 'yellow'];
export class AlternateStateComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      value: '',
      color: 'black'
    };
    this.handleValueChange = this.handleValueChange.bind(this);
  }

  handleValueChange(event) {
    const value = event.target.value;
    this.setState({
      value: value,
      color: COLORS.includes(value) ? value: 'black'
    });
  }

  render() {
    const viewProps = {
      ...this.state,
      label: 'Alternative: ',
      handleValueChange: this.handleValueChange
    }
    return ViewComponent(viewProps);
  }
}
```

nvisia
connect. build. enable.

My Confessions

I Tried to Learn React and Redux at the Same Time

I didn’t Separate Presentation and State

I've Made Components Too Big

Find a common owner component (a single component above all the components that need the state in the hierarchy).

Page Component

- One component with all of the state

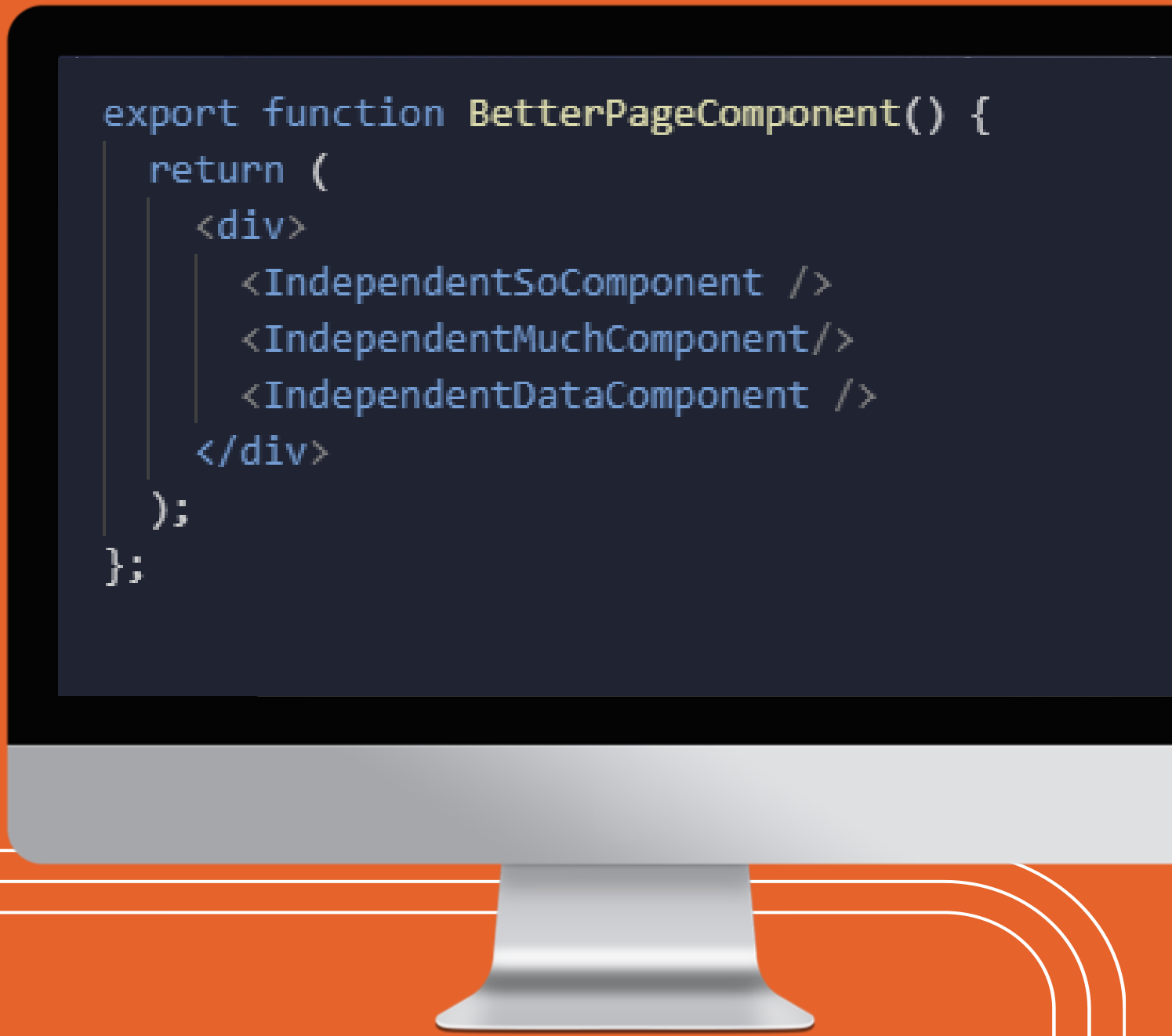
```
export class PageComponent extends Component {
  constructor(props) {
    // all the props for the whole page
    super(props);
    // all the state for the whole page
    this.state = {
      so: {},
      much: {},
      data: props.attr
    };
    // bind all your handlers for the whole page
    this.handleEvent = this.handleEvent.bind(this);
    // ...
  }

  handleEvent() {
    // perform a bunch of loic
    // save some state
  }

  render() {
    return (
      <SoComponent so={this.state.so} />
      <MuchComponent much={this.state.much} />
      <DataComponent
        data={this.state.data}
        handleEvent={this.handleEvent}
      />
    )
  }
}
```

Page Component Remedies

- Consider state earlier in the design process
- Think about the render cycle – What all needs to render when this value changes?
- Components without state are ok
- Centralized State management(Context, Redux, Apollo, etc..)



Over-Generalized Components

- This component Can do everything!
- We'll never need to make another like it!

- What I thought I made:
  - The perfect abstraction that will last forever.
- What I really made:
  - tests

Composability

- Render props
- Higher order Components

My Confessions

- I Tried to Learn React and Redux at the Same Time
- I didn’t Separate Presentation and State
- I've Made Components Too Big

nvisia
connect. build. enable.

I Used component life-cycle to control logic

componentWillReceiveProps

- This is not a replacement for promises
- This triggers on every render!
- This has since been deprecated

```
componentWillReceiveProps(nextProps) {
  if(nextProps.value !== this.props.value) {
    // do something...
  }
}
```


```
componentWillReceiveProps(nextProps) {
  const { cleanUpdate, formState, params, selectedIdentifiers } = this.props;
  const { identifierToRemove } = this.state;
  if (this.userCanCreateNotes() && nextProps.userAttributes.guid !== '') {
    if (!nextProps.formState.noteAuthor || !nextProps.formState.noteAuthor.guid) {
      cleanUpdate({ noteAuthor: nextProps.userAttributes });
    }
  }
  if (nextProps.params.noteId !== params.noteId) {
    this.triggerDelayedSummaryAutoSave.cancel();
    this.updateNote(nextProps.params.noteId);
  }
  const identifierDiff = _.differenceWith(selectedIdentifiers,
  formState.selectedIdentifiers, (prop, form) => {
    return (
      prop.id === form.id &&
      prop.name === form.name &&
      prop.type === form.type &&
      prop.private === form.private
    );
  });

  const newIdentifier = identifierDiff && identifierDiff.length > 0 ? identifierDiff
  [0] : {};
  const isSecurityRequest = newIdentifier.type === IDENTIFIER_TYPE.SECURITY_NEW;
  if (isSecurityRequest && !_.isEqual(newIdentifier, identifierToRemove)) {
    this.handleAddSelectedIdentifier(identifierDiff[0]);
  }
}
```

nvisia
connect. build. enable.

My Confessions

- I Tried to Learn React and Redux at the Same Time
- I didn’t Separate Presentation and State
- I've Made Components Too Big
- I Used Component Life-Cycle to Control Logic

nvisia
connect. build. enable.

I stored deep objects in the state

Deep object in React state

```
this.state = {
  modal:{
    isActive: false,
    data: {}
  },
  textValue: '',
  currentUserInfo: {
    name: {
      first: '',
      last: '',
    },
    id: null,
  }
};
```

nvisia
connect. build. enable.

Immutability

- Updates
- Unnecessary renders

nvisia
connect. build. enable.

Flatten it!

```
this.state = {
  modalIsActive: false,
  modalData: {},
  textValue: '',
  currentUserInfoFirstName: '',
  currentUserInfoLastName: '',
  currentUserInfoId: null
};
```

nvisia
connect. build. enable.

My Confessions

- I Tried to Learn React and Redux at the Same Time
- I didn’t Separate Presentation and State
- I've Made Components Too Big
- I Used Component Life-Cycle to Control Logic
- I Stored Deep Objects in the State

nvisia
connect. build. enable.

I didn't recognize the different kinds of state

App state - state that only exists in the app to change component presentation

Api Data - The results for api calls. Your domain data

Form data - Data that changes based on user input

nvisia
connect. build. enable.

App State

```
export class AppStateComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      isOpen: '',
    };
    this.handleOpenToggle = this.handleOpenToggle.bind(this);
  }

  handleOpenToggle() {
    this.setState({
      isOpen: !this.state.isOpen,
    });
  }

  render() {
    const {isOpen} = this.state;
    const {children} = this.props;
    return (
      <React.Fragment>
        <button onClick={this.handleOpenToggle} >Click Me!</button>
        {isOpen ? children : null}
      </React.Fragment>
    );
  }
}
```

Api Data

```
ComponentDidMount() {
  fetchCurrentUserInfo().then(storeInCentralizedState)
}
```

Form Data

```
export class FormDataComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      value: props.initialValue,
    };
    this.handleValueChange = this.handleValueChange.bind(this);
    this.handleValueSave = this.handleValueSave.bind(this);
  }

  handleValueChange(event) {
    this.setState({value: event.target.value});
  }

  handleValueSave() {
    apiSave(this.state.value);
  }

  render() {
    return (
      <ViewComponent
        value={value}
        handleValueChange={this.handleValueChange}
        handleValueSave={this.handleValueSave}
      />
    );
  }
}
```

nvisia
connect. build. enable.

My Confessions

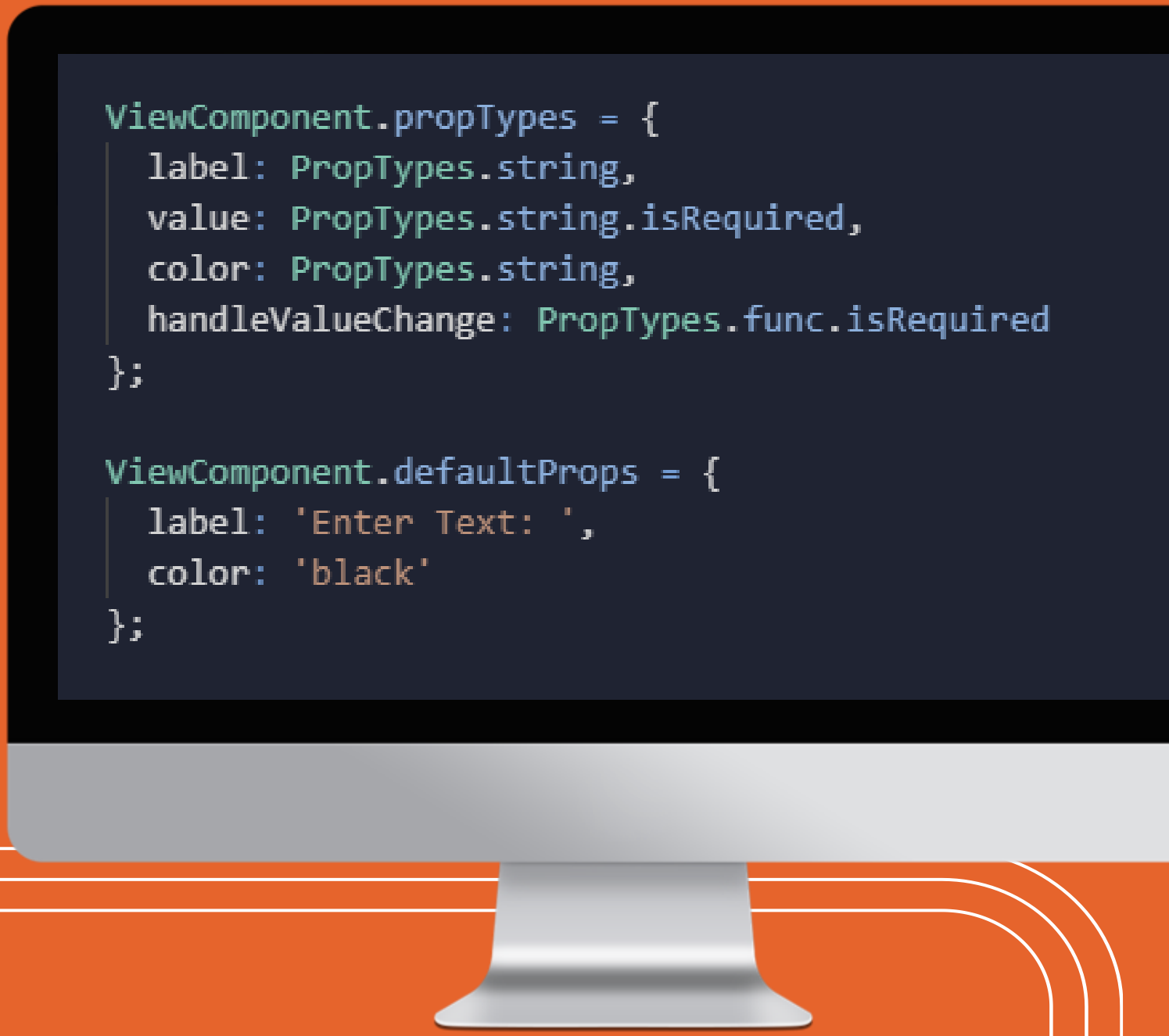
- I Tried to Learn React and Redux at the Same Time
- I didn’t Separate Presentation and State
- I've Made Components Too Big
- I Used Component Life-Cycle to Control Logic
- I Stored Deep Objects in the State
- I Didn’t Recognize the Different Kinds of State

nvisia
connect. build. enable.

I didn't utilize PropTypes

What are PropTypes?

- Types checking in React
- Split into its own library in v15.5
- Not necessary if you're using Typescript



nvisia
connect. build. enable.

The benefits of PropTypes

- Obviously, type checking
- Optional vs. Required props
- Communicates a contract
- Custom PropTypes!

My Confessions

- I Tried to Learn React and Redux at the Same Time
- I didn’t Separate Presentation and State
- I've Made Components Too Big
- I Used Component Life-Cycle to Control Logic
- I Stored Deep Objects in the State
- I Didn’t Recognize the Different Kinds of State
- I didn't utilize PropTypes

nvisia
connect. build. enable.

I didn't Minimize/Isolate DOM Interactions

Refs

Reacts built in way to manipulate the DOM

```
handleSave(saveCompletefn) {
    let author = this.props.pitchAuthor ? this.props.pitchAuthor : undefined;
    let securityId = this.props.selectedSecurity ? this.props.selectedSecurity.id : null;
    let scenarioId = this.props.pitch.securityScenarioAnalysis?this.props.pitch.securityScenarioAnalysis.id:null;
    let pitchData =
    {
        id: this.props.pitch.id,
        author: author,
        pitchDate: moment().format("YYYY-MM-DD"),
        packetLink: "TODO",
        franchise: this.refs.franchise.value,
        exposures: this.refs.exposures.value,
        management: this.refs.management.value,
        competition: this.refs.competition.value,
        marginProfile: this.refs.marginProfile.value,
        balanceSheet: this.refs.balanceSheet.value,
        conclusion: this.refs.conclusion.value,
        securityScenarioAnalysis:
        {
            id: scenarioId,
            edmSecurityId: securityId,
            statusId: SCENARIO_STATUS.DRAFT.id,
            scenarioAnalysisDate: moment().format("YYYY-MM-DD"),
            thesis: this.refs.thesis.value,
            bearCasePercent: this.refs.bearCasePercent.value,
            bearCase: this.refs.bearCase.value,
            bullCasePercent: this.refs.bullCasePercent.value,
            bullcase: this.refs.bullcase.value,
            baseCasePercent: this.refs.baseCasePercent.value,
            baseCase: this.refs.baseCase.value
        }
    };
    this.props.savePitchPacket(pitchData, saveCompletefn);
}
```

When to manipulate the DOM?

- Libraries that require DOM interactions should be isolated in components that wrap all this logic and prevent rerenders.
- When you need to handle focus or like operations

My Confessions

- I Tried to Learn React and Redux at the Same Time
- I didn’t Separate Presentation and State
- I've Made Components Too Big
- I Used Component Life-Cycle to Control Logic
- I Stored Deep Objects in the State
- I Didn’t Recognize the Different Kinds of State
- I didn't utilize PropTypes
- I didn't Minimize/Isolate DOM Interactions

I didn't Define a Style Guide

Without a style guide

- Every file feels like it was written by someone else
- Individual styles deviate over time

How to enforce a Style guide

- Linting

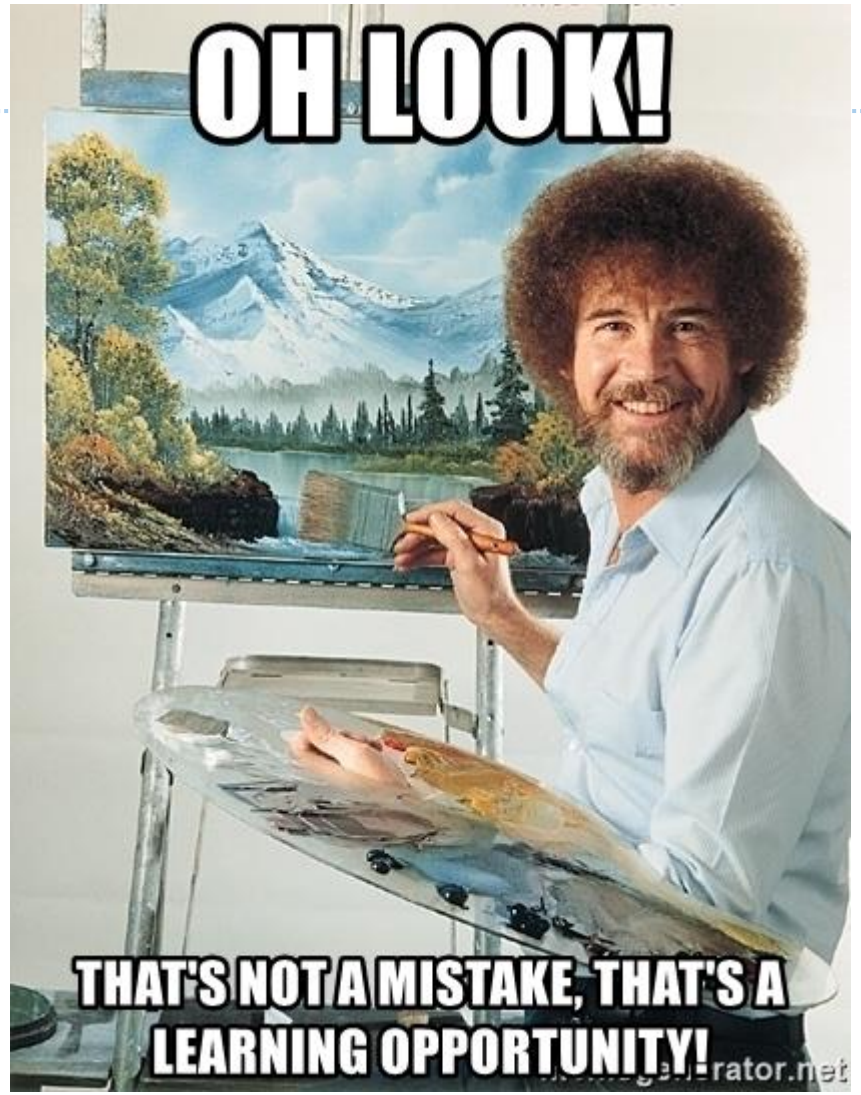
My Confessions

- I Tried to Learn React and Redux at the Same Time
- I didn’t Separate Presentation and State
- I've Made Components Too Big
- I Used Component Life-Cycle to Control Logic
- I Stored Deep Objects in the State
- I Didn’t Recognize the Different Kinds of State
- I Didn't utilize PropTypes
- I Didn't Minimize/Isolate DOM Interactions
- I Didn’t Define a Style Guide

nvisia
connect. build. enable.

Summary

- Try to learn 1 thing at a time
- Separate presentation and state
- Break up your components/ think composability
- Minimize life-cycle interactions
- Flatten your objects
- Treat different types of state differently
- Utilize your tools/PropTypes
- Minimize/isolate DOM interaction
- Set a style guide with your team



My Confessions

- I Tried to Learn React and Redux at the Same Time
- I didn’t Separate Presentation and State
- I've Made Components Too Big
- I Used Component Life-Cycle to Control Logic
- I Stored Deep Objects in the State
- I Didn’t Recognize the Different Kinds of State
- I Didn't utilize PropTypes
- I Didn't Minimize/Isolate DOM Interactions
- I Didn’t Define a Style Guide
- I Gave This Presentation?

nvisia
connect. build. enable.

Questions?