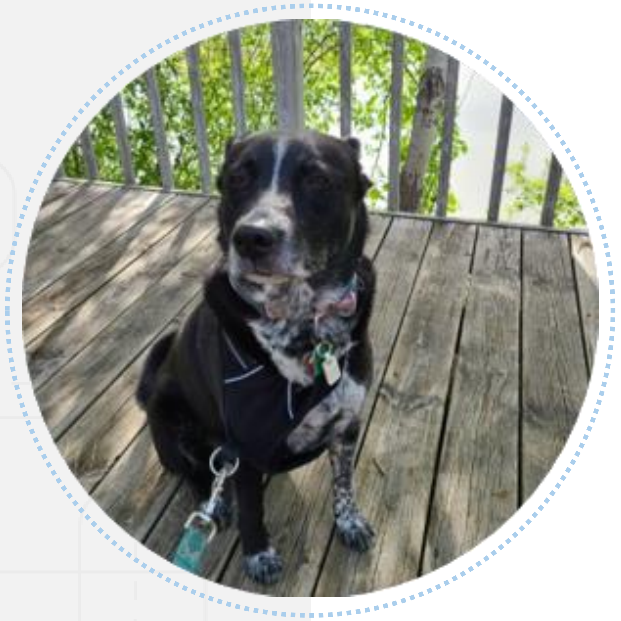
Things I’ve Done Wrong in React.js

Josh Doro

nvisia
connect. build. enable.



Who Am I?

- Senior Technical Architect with nvisia
- nvisia for almost 9 years
- Working with React.js for 7

nvisia
connect. build. enable.

About nvisia

- Software consulting firm
- Offices in Chicago and Milwaukee
- A group of passionate technologists who like to help create cool software.
- Involved in tech community

nvisia
connect. build. enable.

Bad Code

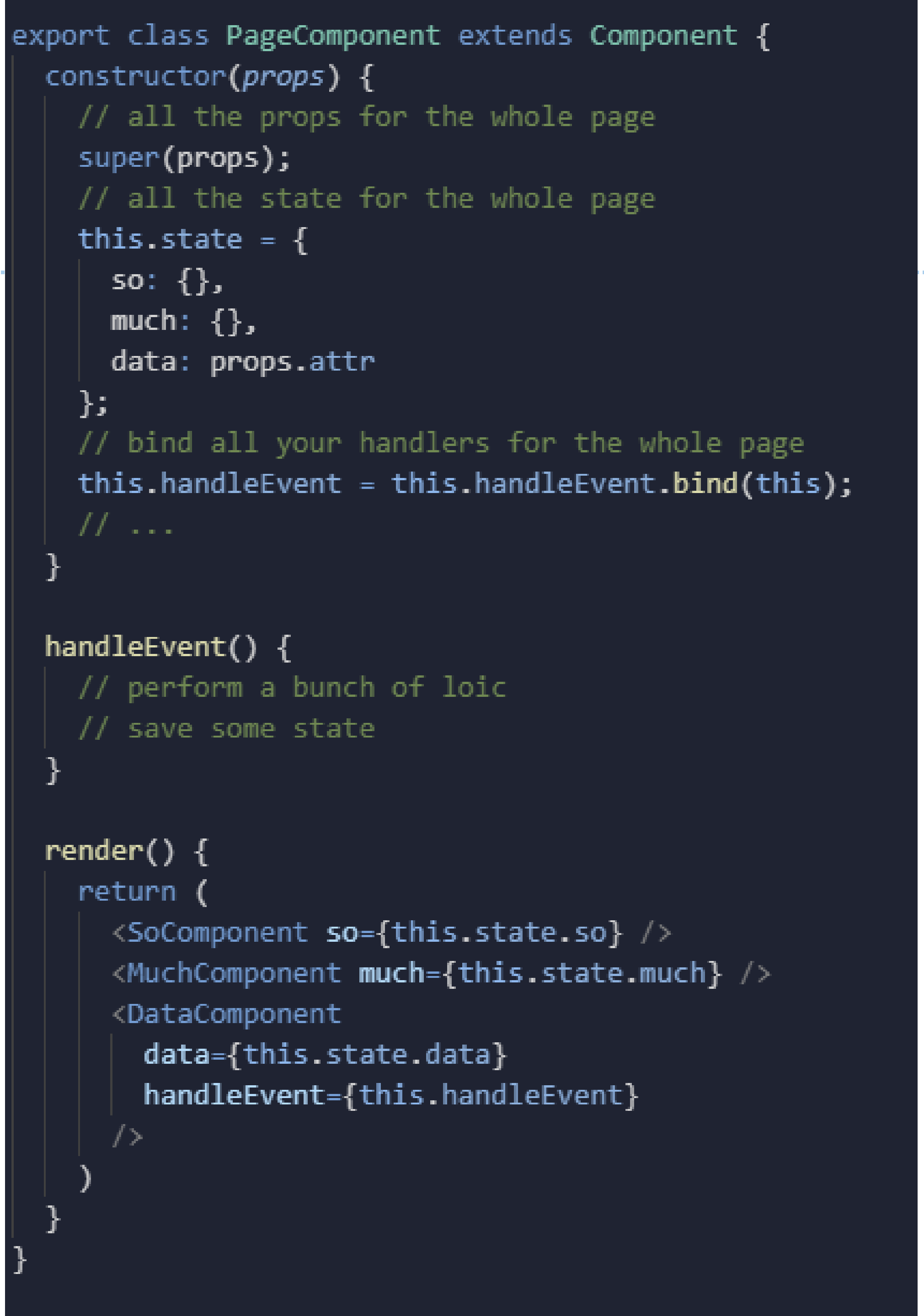
When you finally catch the person that's been writing bad code all the time
It's just me, myself and I

I Made Components Too Big

nvisia
connect. build. enable.

Page Component

- One component with all of the state

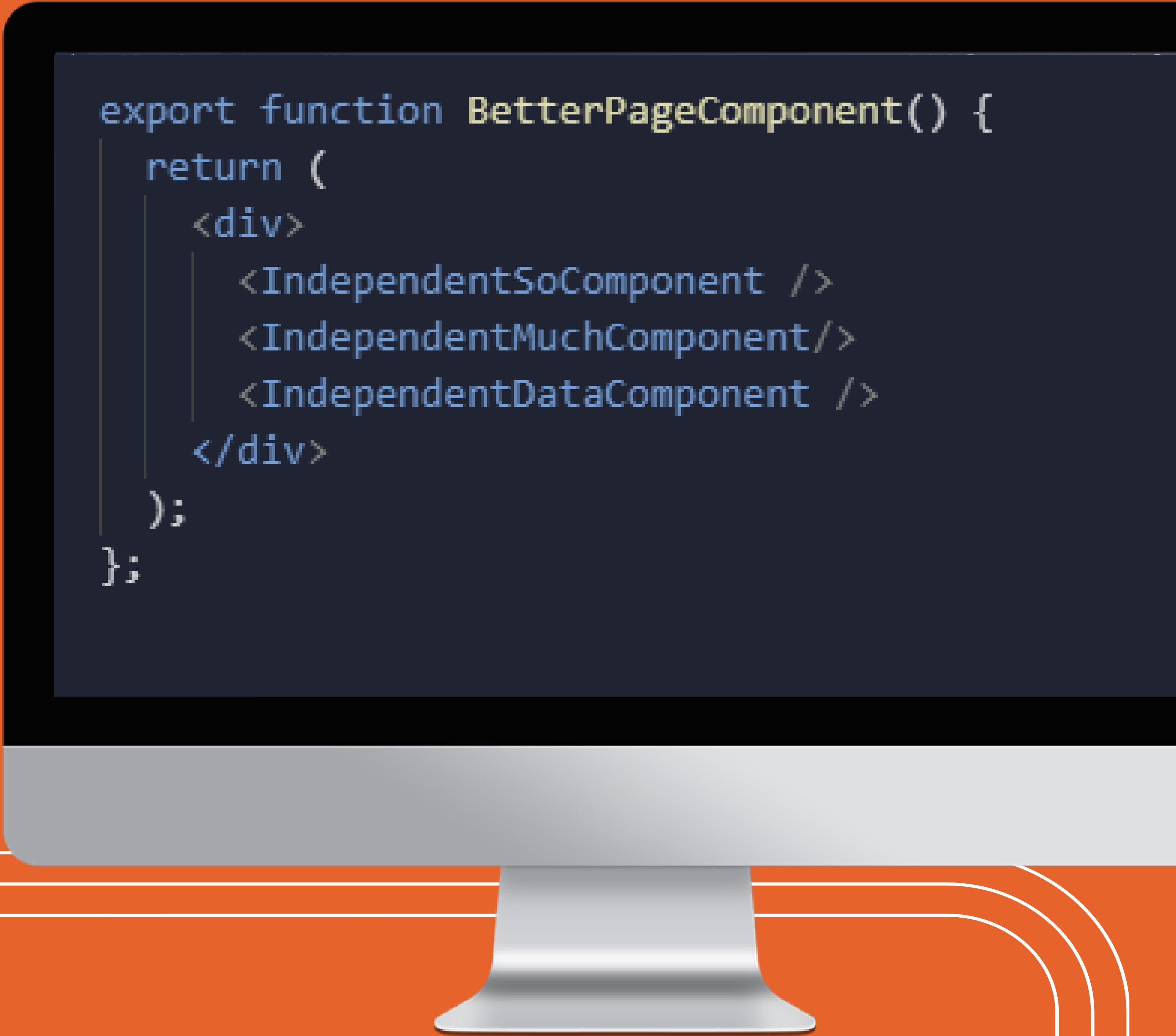


```
export class PageComponent extends Component {
  constructor(props) {
    // all the props for the whole page
    super(props);
    // all the state for the whole page
    this.state = {
      so: {},
      much: {},
      data: props.attr
    };
    // bind all your handlers for the whole page
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

nvisia
connect. build. enable.

Page Component Remedies

- 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..)

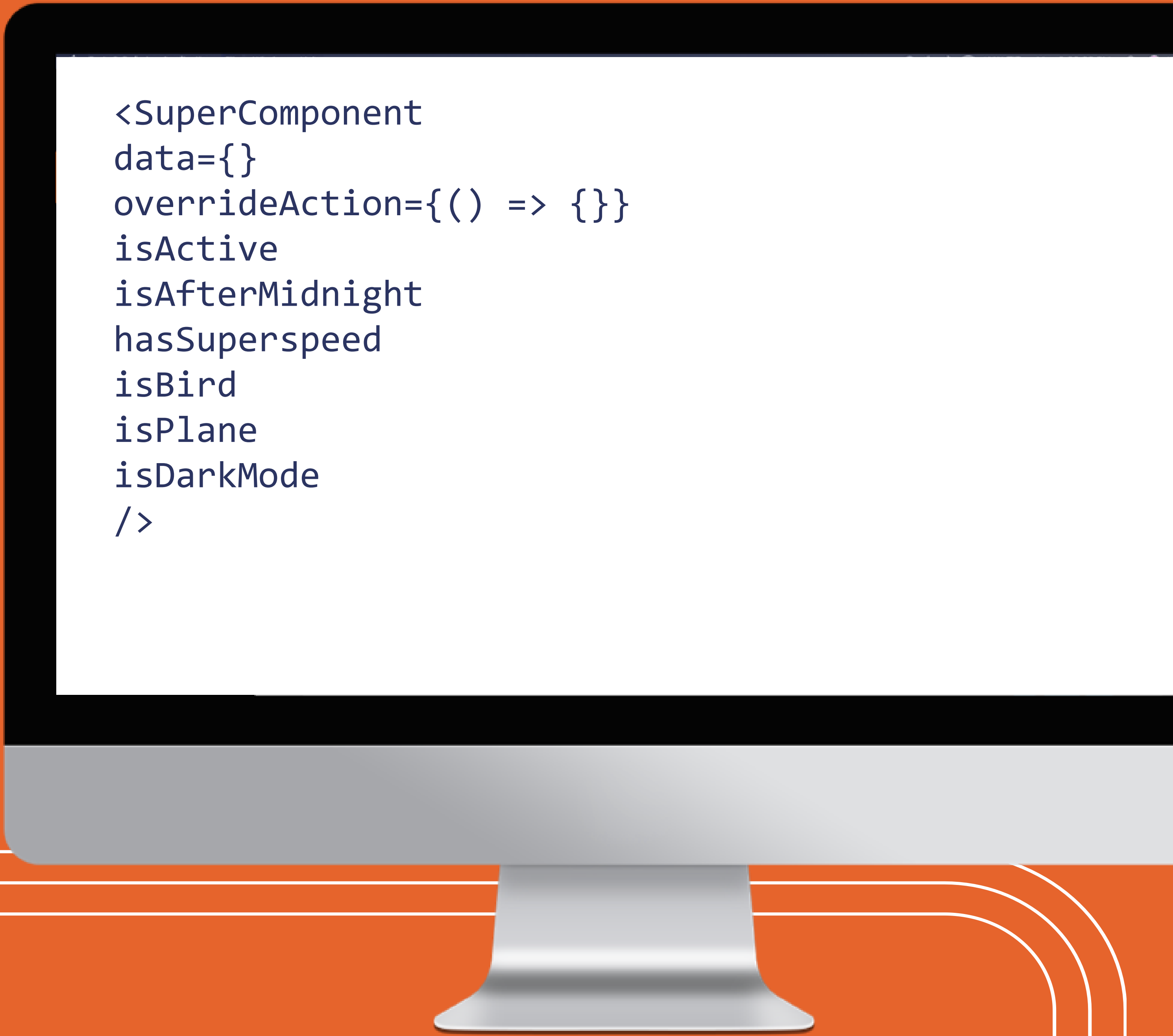


```
export function BetterPageComponent() {
  return (
    <div>
      <IndependentSoComponent />
      <IndependentMuchComponent/>
      <IndependentDataComponent />
    </div>
  );
};
```

nvisia
connect. build. enable.

Over-Generalized Components

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



```
<SuperComponent
data={}
overrideAction={() => {}}
isActive
isAfterMidnight
hasSuperspeed
isBird
isPlane
isDarkMode
/>
```

nvisia
connect. build. enable.

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

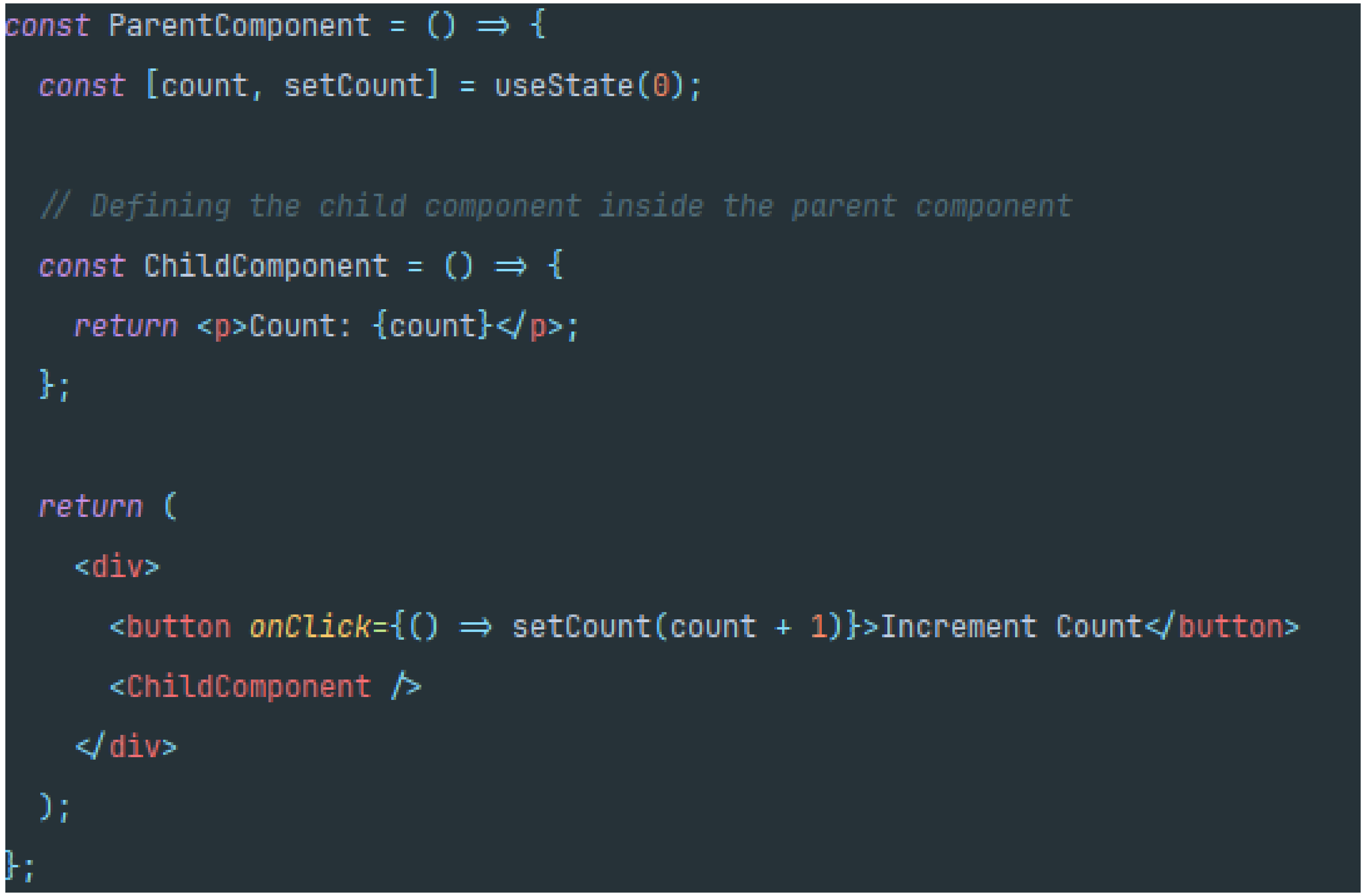
Composability

- Render props
- Higher order Components
- Custom hooks

I Defined Components Inside Components

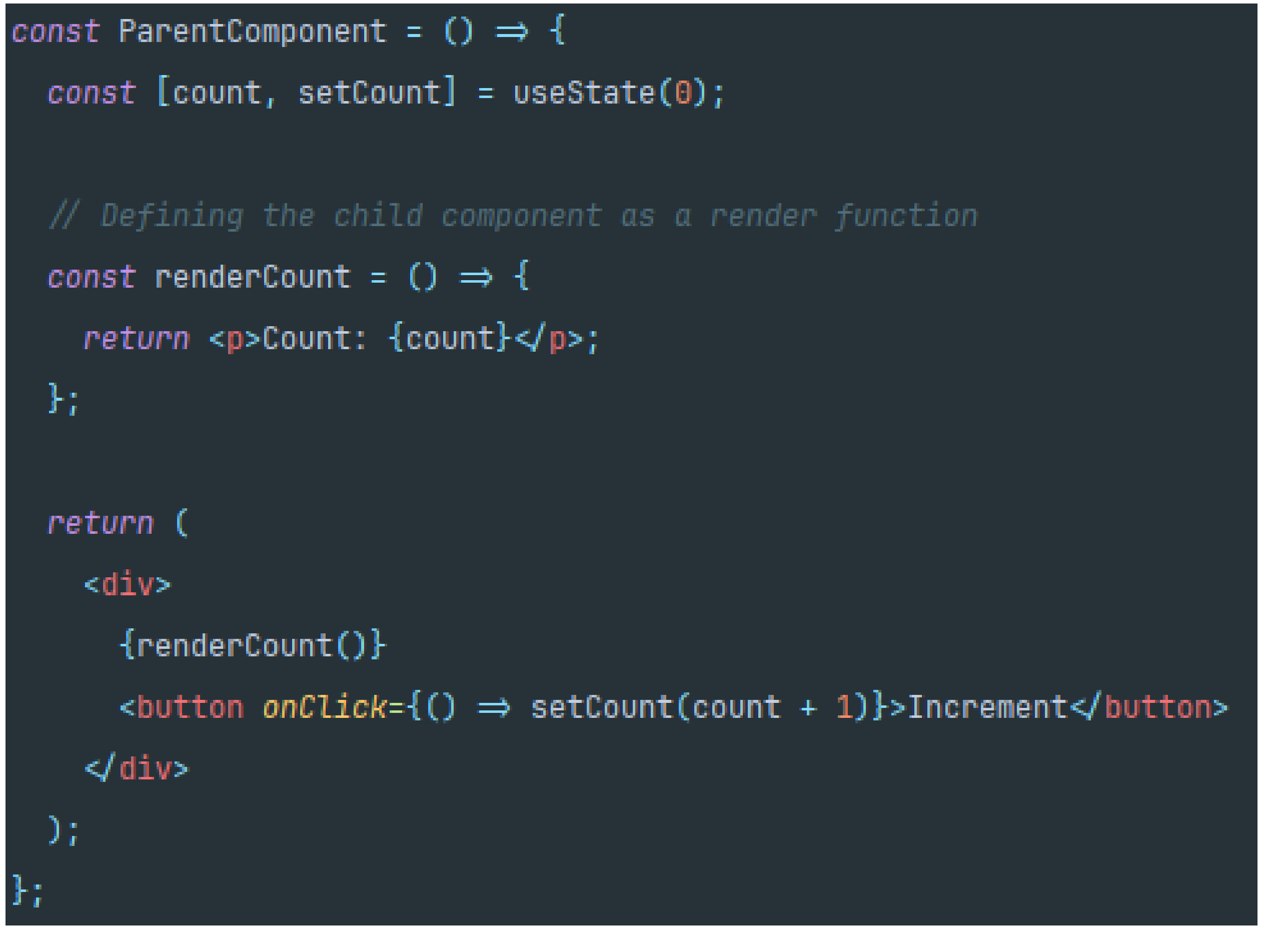
nvisia
connect. build. enable.

Redefined on Every Render



```
const ParentComponent = () ⇒ {
  const [count, setCount] = useState(0);

  // Defining the child component inside the parent component
  const ChildComponent = () ⇒ {
    return <p>Count: {count}</p>;
  };

  return (
    <div>
      <button onClick={() ⇒ setCount(count + 1)}>Increment Count</button>
      <ChildComponent />
    </div>
  );
};
```

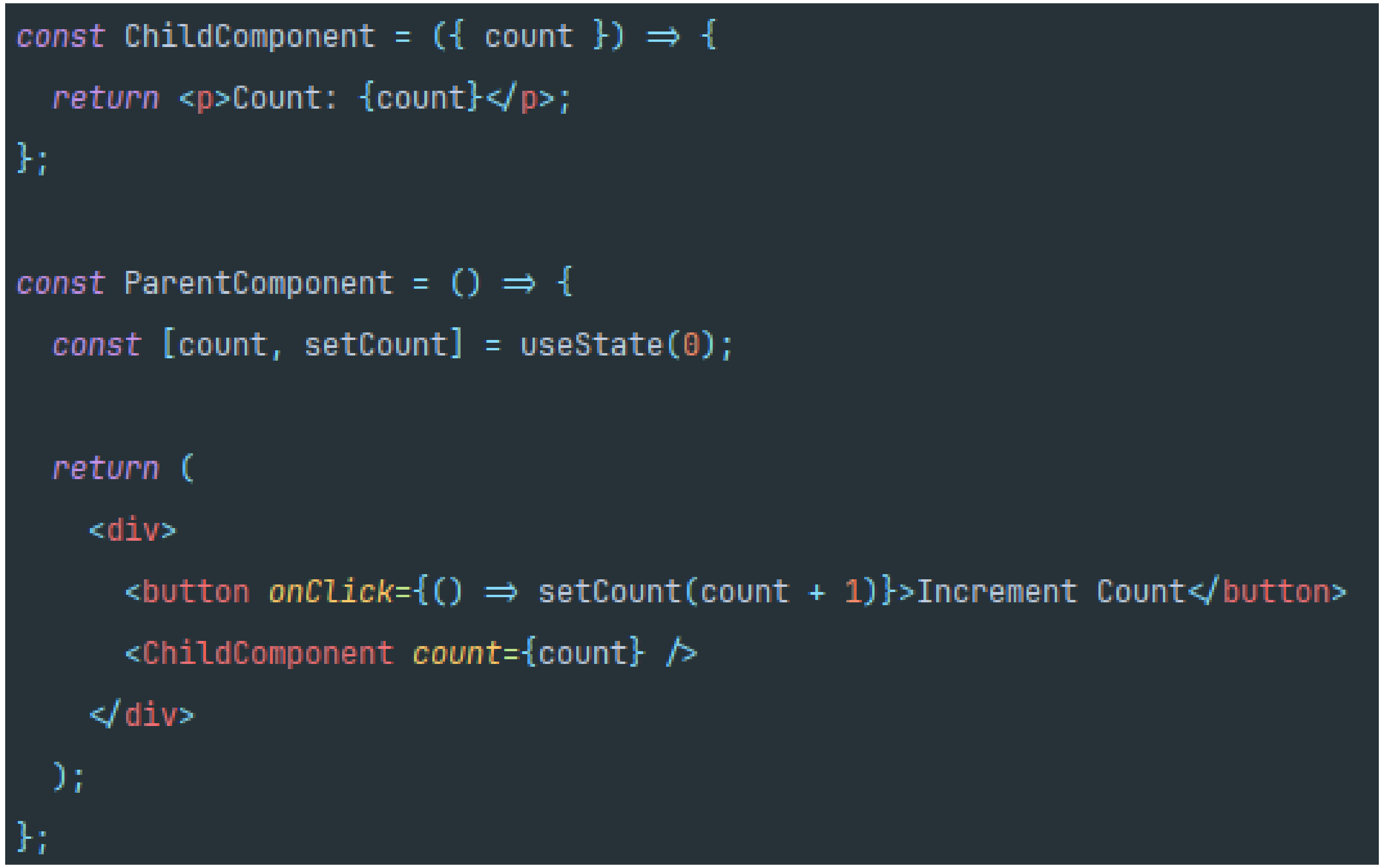
Render function



```
const ParentComponent = () ⇒ {
  const [count, setCount] = useState(0);

  // Defining the child component as a render function
  const renderCount = () ⇒ {
    return <p>Count: {count}</p>;
  };

  return (
    <div>
      {renderCount()}
      <button onClick={() ⇒ setCount(count + 1)}>Increment</button>
    </div>
  );
};
```

nvisia
connect. build. enable.

Defined Outside Life-Cycle



```
const ChildComponent = ({ count }) ⇒ {
  return <p>Count: {count}</p>;
};

const ParentComponent = () ⇒ {
  const [count, setCount] = useState(0);

  return (
    <div>
      <button onClick={() ⇒ setCount(count + 1)}>Increment Count</button>
      <ChildComponent count={count} />
    </div>
  );
};
```

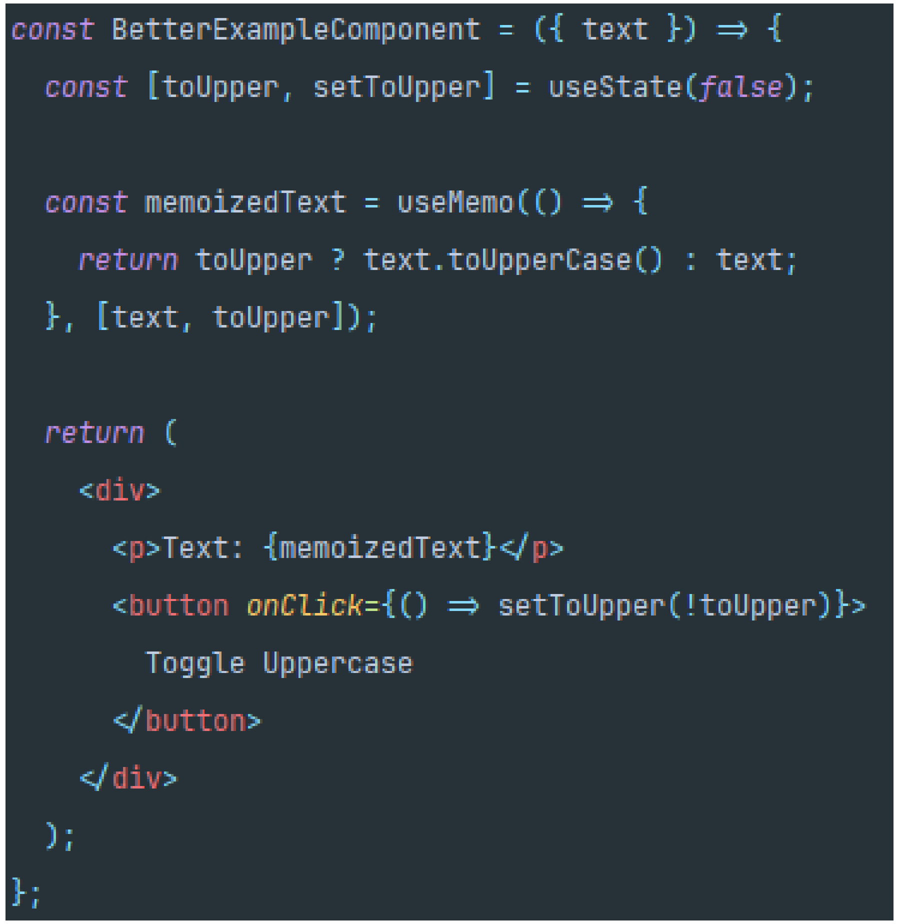
I Made Copies of State

Keep it In-Sync

```jsx
const BadExampleComponent = ({ text }) => {
  const [textState, setTextState] = useState(text);

  useEffect(() => {
    setTextState(text);
  }, [text]);

  return (
    <div>
      <p>Text: {textState}</p>
      <button onClick={() => setTextState(textState.toUpperCase())}>Uppercase Text</button>
    </div>
  );
};
```

nvisia
connect. build. enable.

Derive it!



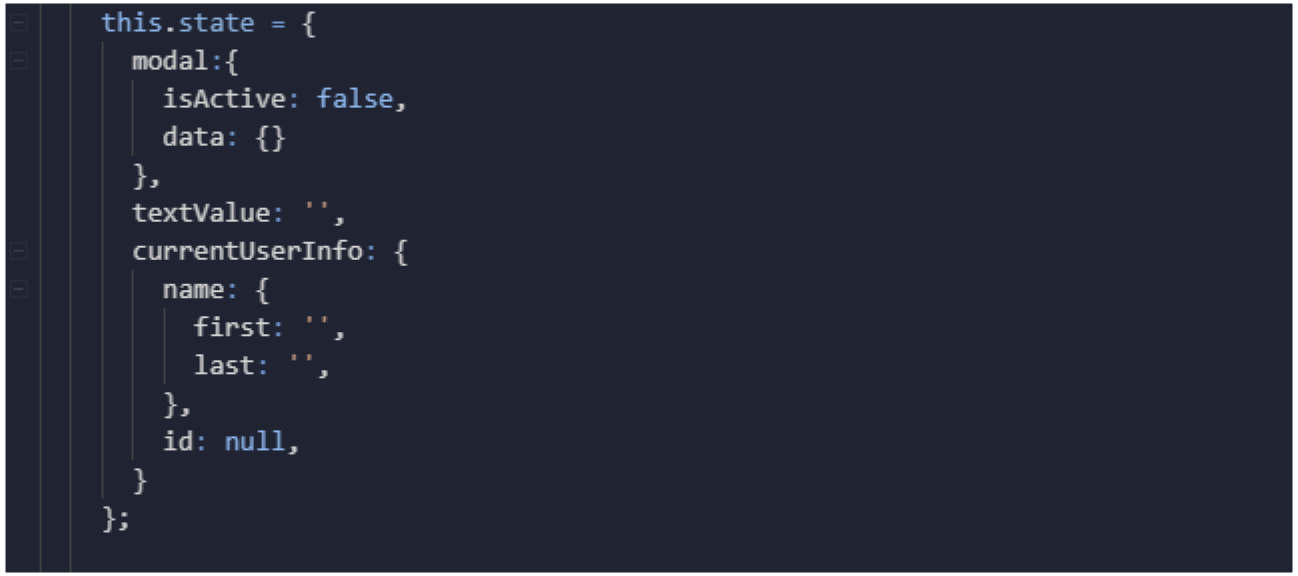
```
const BetterExampleComponent = ({ text }) ⇒ {
  const [toUpper, setToUpper] = useState(false);

  const memoizedText = useMemo(() ⇒ {
    return toUpper ? text.toUpperCase() : text;
  }, [text, toUpper]);

  return (
    <div>
      <p>Text: {memoizedText}</p>
      <button onClick={() ⇒ setToUpper(!toUpper)}>
        Toggle Uppercase
      </button>
    </div>
  );
};
```

nvisia
connect. build. enable.

I Stored Deep Objects in State

nvisia
connect. build. enable.

Deep object in React state

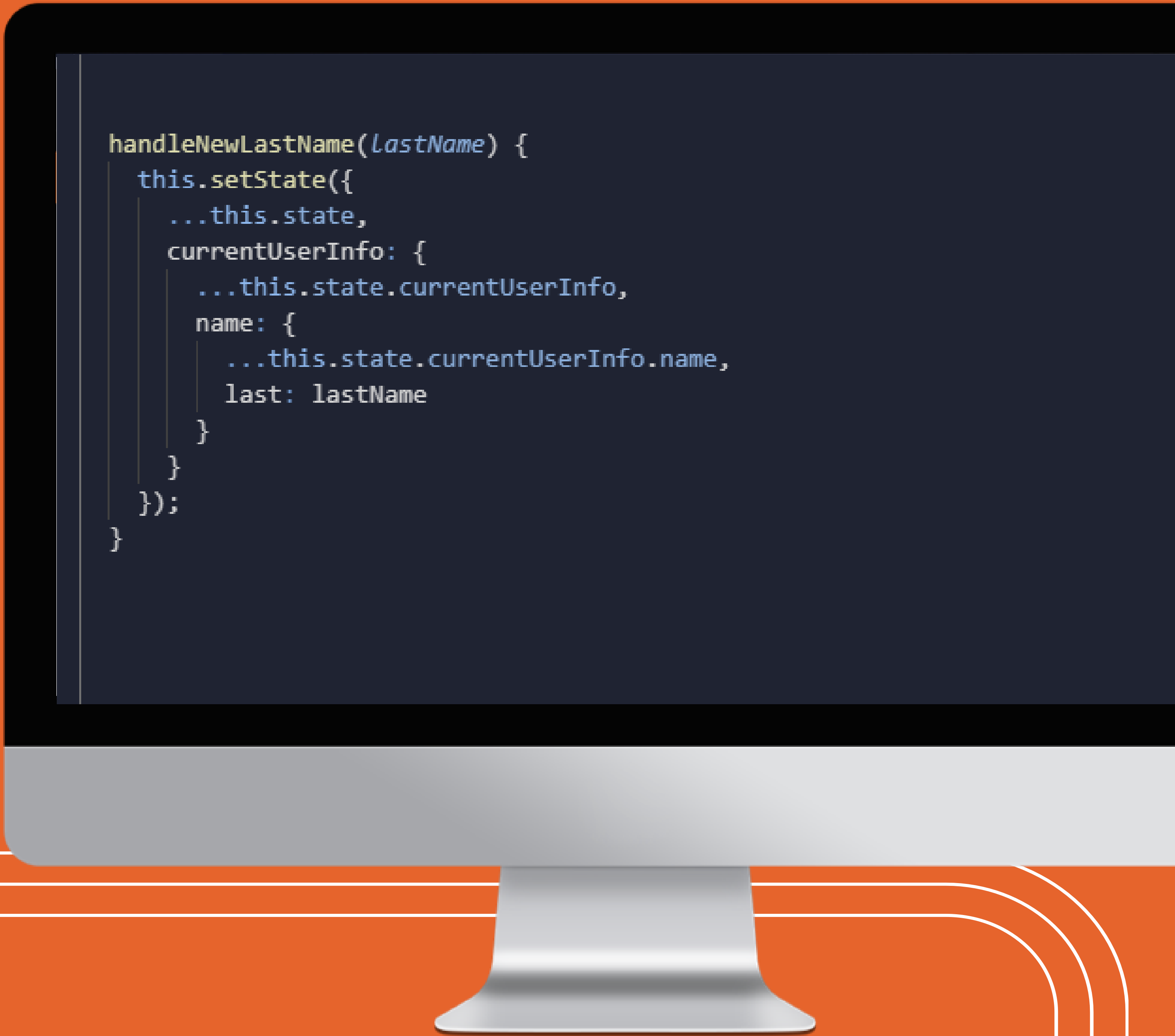


```
this.state = {
  modal:{
    isActive: false,
    data: {}
  },
  textValue: '',
  currentUserInfo: {
    name: {
      first: '',
      last: '',
    },
    id: null,
  }
};
```

nvisia
connect. build. enable.

Immutability

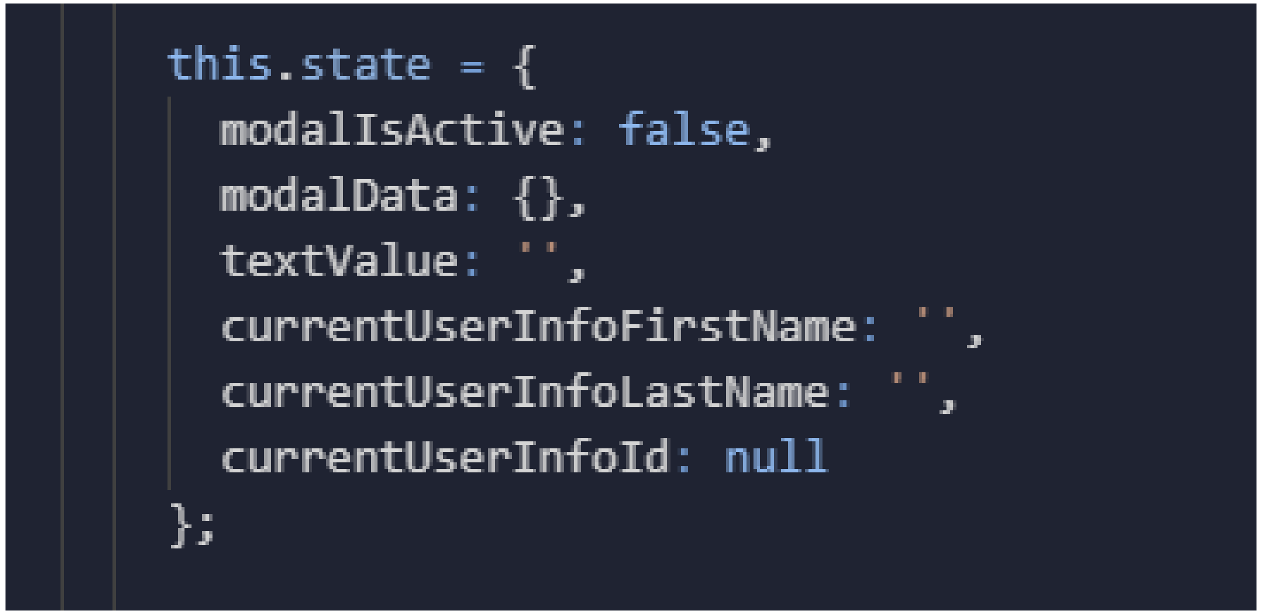
- Updates
- Unnecessary renders



```
handleNewLastName(lastName) {
  this.setState({
    ...this.state,
    currentUserInfo: {
      ...this.state.currentUserInfo,
      name: {
        ...this.state.currentUserInfo.name,
        last: lastName
      }
    }
  });
}
```

nvisia
connect. build. enable.

Flatten it!



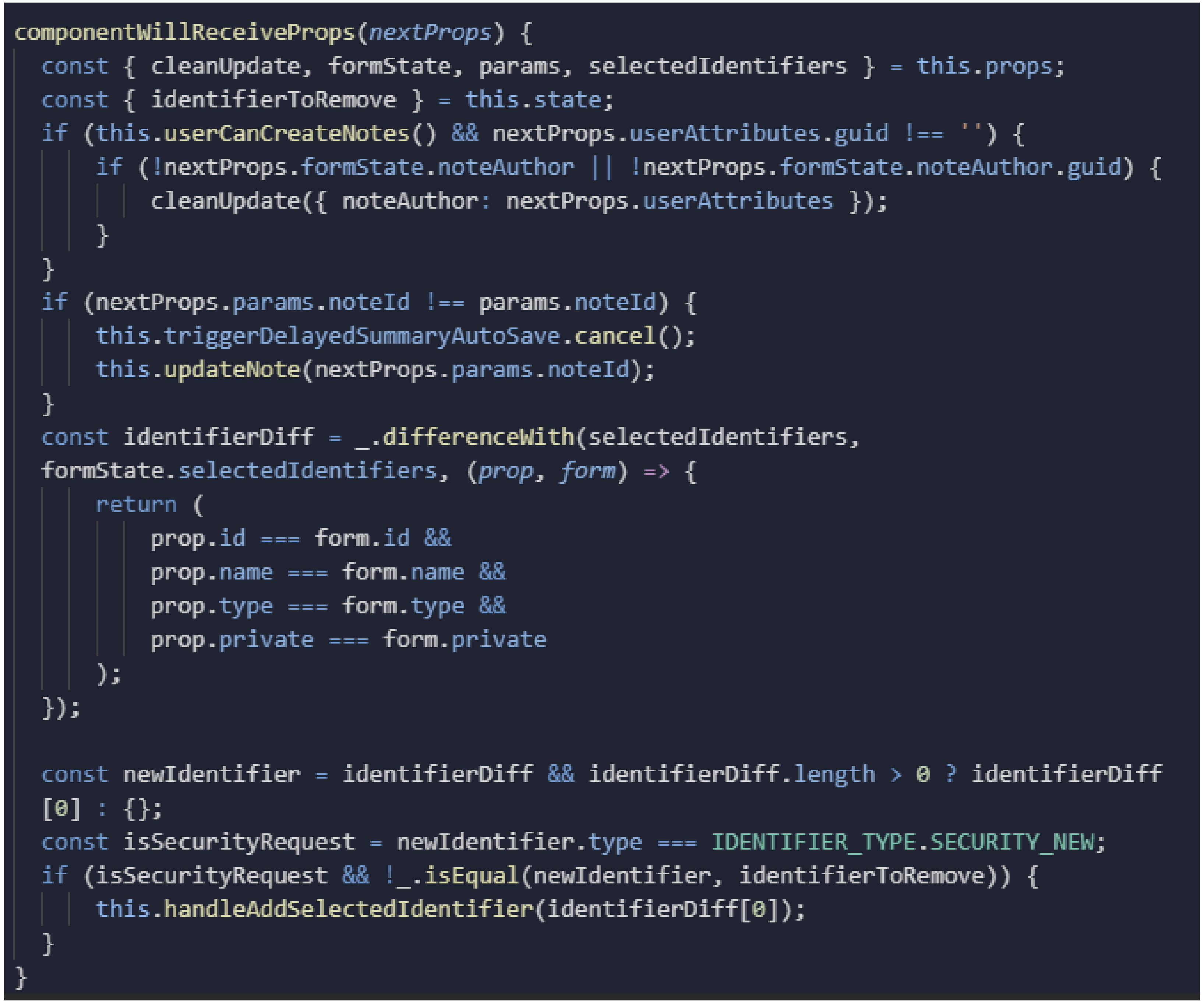
```
this.state = {
  modalIsActive: false,
  modalData: {},
  textValue: '',
  currentUserInfoFirstName: '',
  currentUserInfoLastName: '',
  currentUserInfoId: null
};
```

I Abused React's Life-Cycle

nvisia
connect. build. enable.

Pre-hooks

- componentWillMount(removed)
- componentDidMount
- componentWillReceiveProps(removed)
- componentWillUpdate(removed)
- shouldComponentUpdate
- componentDidUpdate
- componentWillUnmount

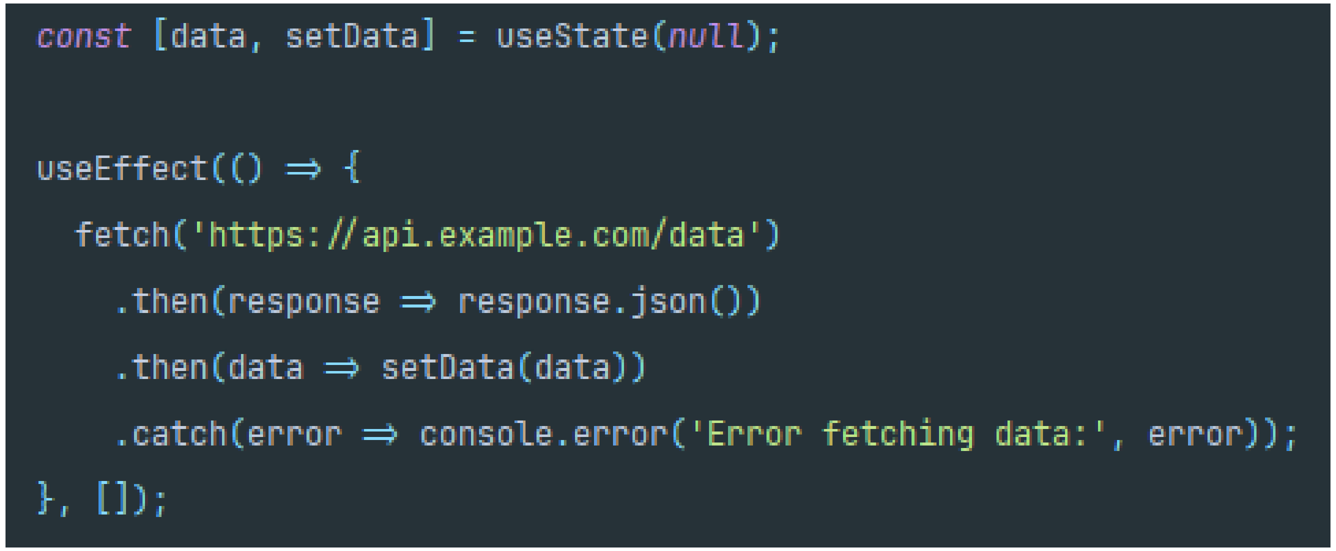


```
componentWillReceiveProps(nextProps) {
  const { cleanUpdate, formState, params, selectedIdentifiers } = this.props;
  const { identifierToRemove } = this.state;
  if (this.userCanCreateNotes() && nextProps.userAttributes.guid !== '') {
    if (!nextProps.formState.noteAuthor || !nextProps.formState.noteAuthor.guid) {
      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.

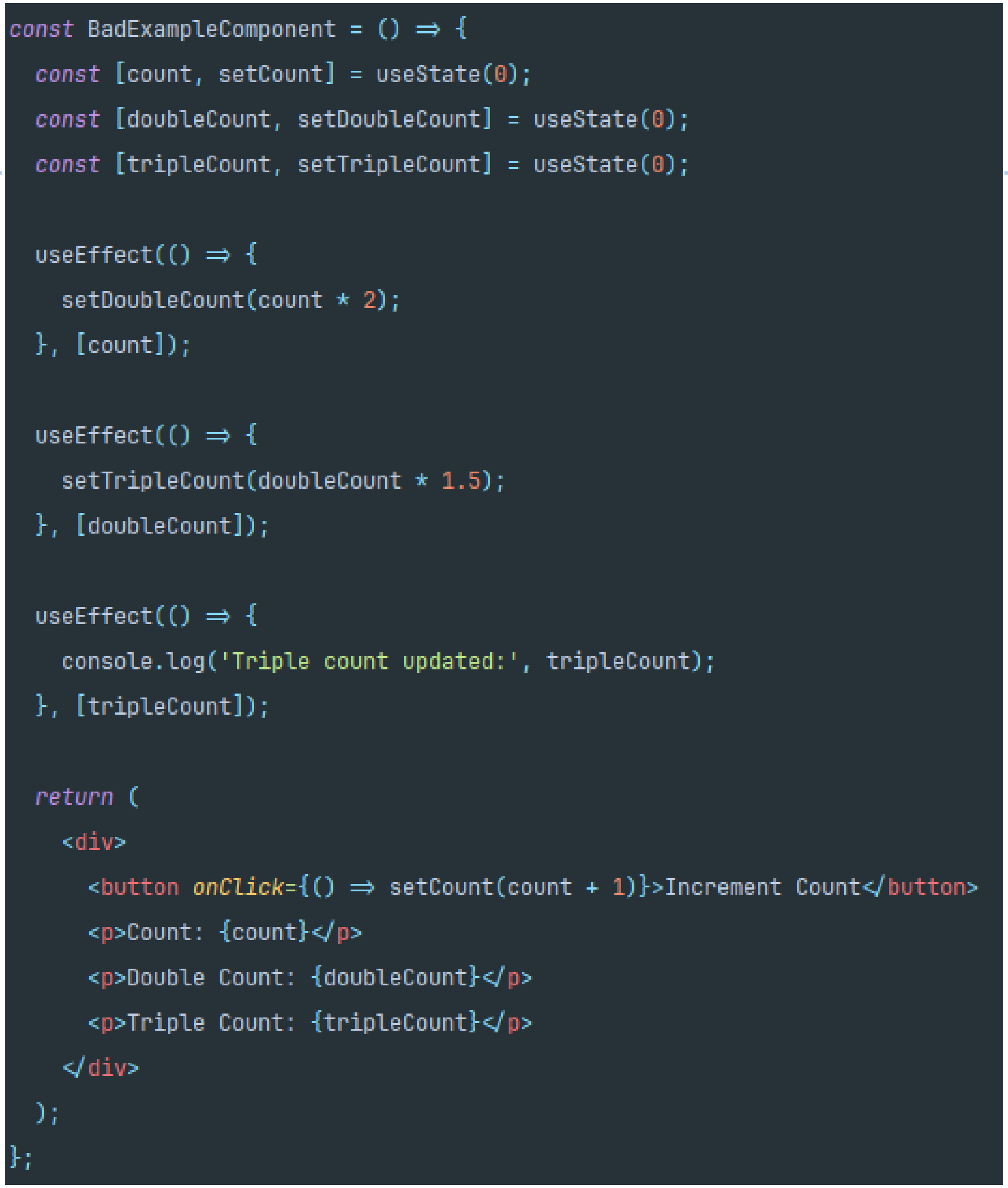
A new way to abuse lifecycle - useEffect

- Side Effects
- Runs whenever one of it's dependencies changes
- An empty dependency Array means it only runs on mount
- Returning a function will call that function on unmount

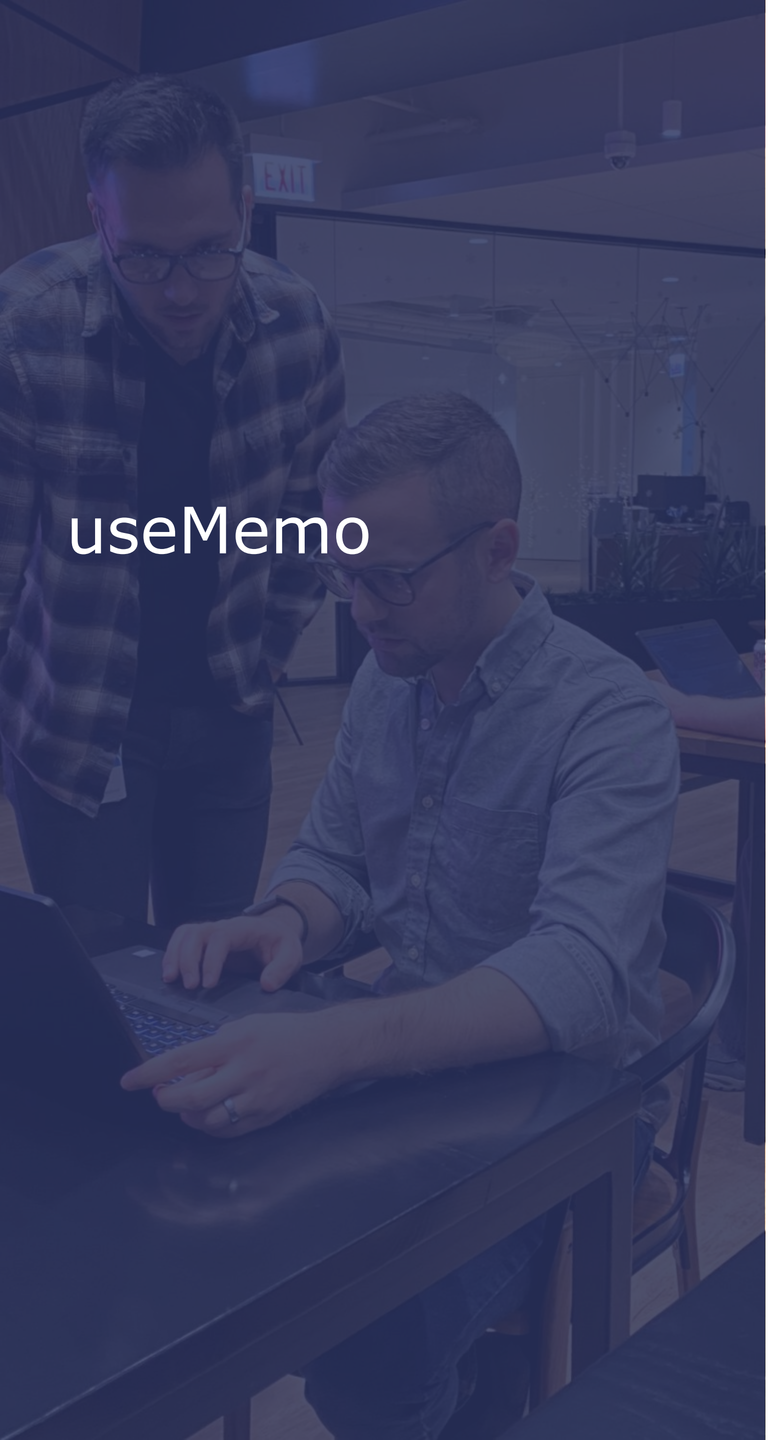


```
const [data, setData] = useState(null);

useEffect(() => {
  fetch('https://api.example.com/data')
    .then(response => response.json())
    .then(data => setData(data))
    .catch(error => console.error('Error fetching data:', error));
}, []);
```

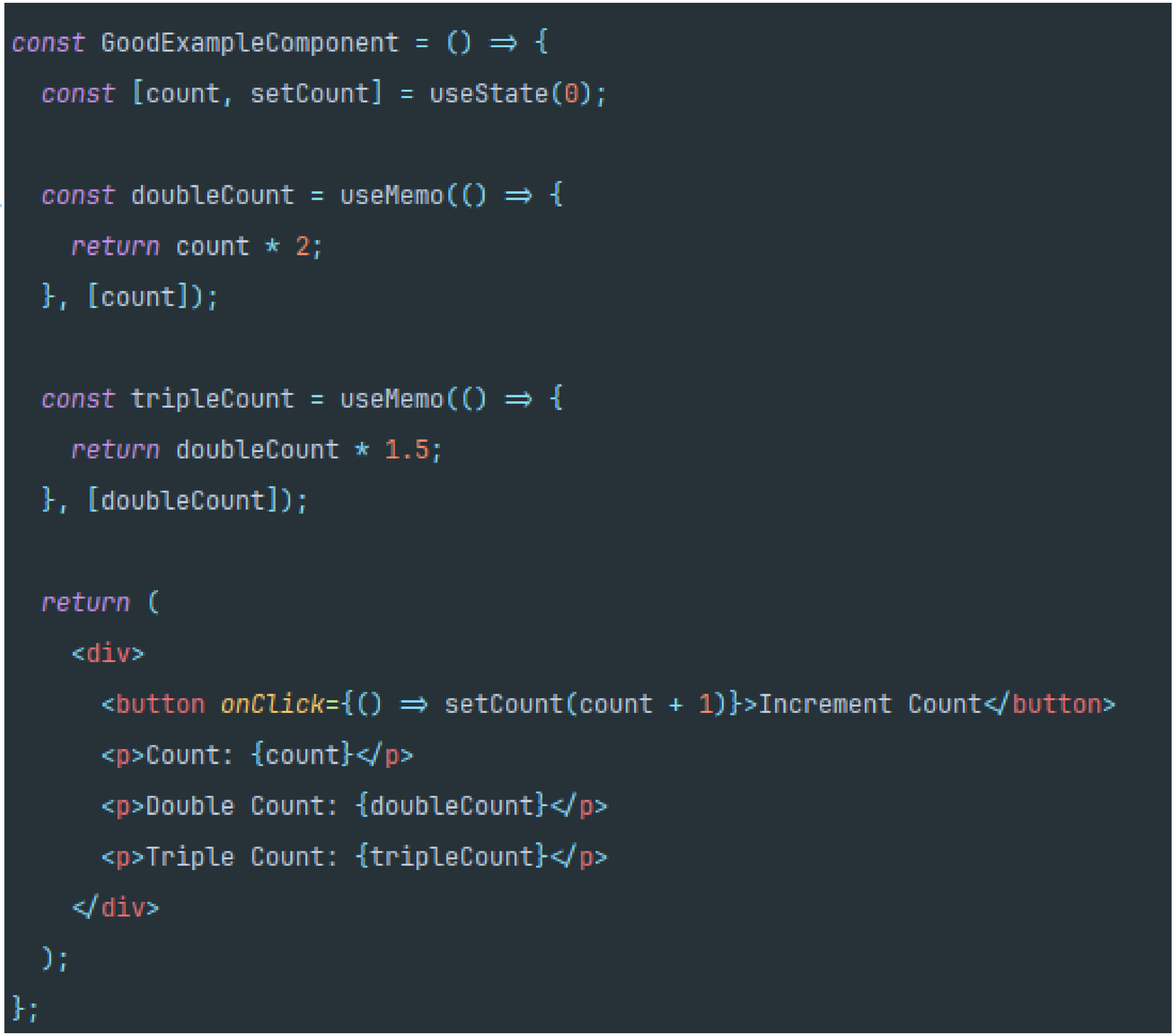
useEffect



```jsx
const BadExampleComponent = () => {
  const [count, setCount] = useState(0);
  const [doubleCount, setDoubleCount] = useState(0);
  const [tripleCount, setTripleCount] = useState(0);

  useEffect(() => {
    setDoubleCount(count * 2);
  }, [count]);

  useEffect(() => {
    setTripleCount(doubleCount * 1.5);
  }, [doubleCount]);

  useEffect(() => {
    console.log('Triple count updated:', tripleCount);
  }, [tripleCount]);

  return (
    <div>
      <button onClick={() => setCount(count + 1)}>Increment Count</button>
      <p>Count: {count}</p>
      <p>Double Count: {doubleCount}</p>
      <p>Triple Count: {tripleCount}</p>
    </div>
  );
};
```



useMemo



```
const GoodExampleComponent = () => {
  const [count, setCount] = useState(0);

  const doubleCount = useMemo(() => {
    return count * 2;
  }, [count]);

  const tripleCount = useMemo(() => {
    return doubleCount * 1.5;
  }, [doubleCount]);

  return (
    <div>
      <button onClick={() => setCount(count + 1)}>Increment Count</button>
      <p>Count: {count}</p>
      <p>Double Count: {doubleCount}</p>
      <p>Triple Count: {tripleCount}</p>
    </div>
  );
};
```

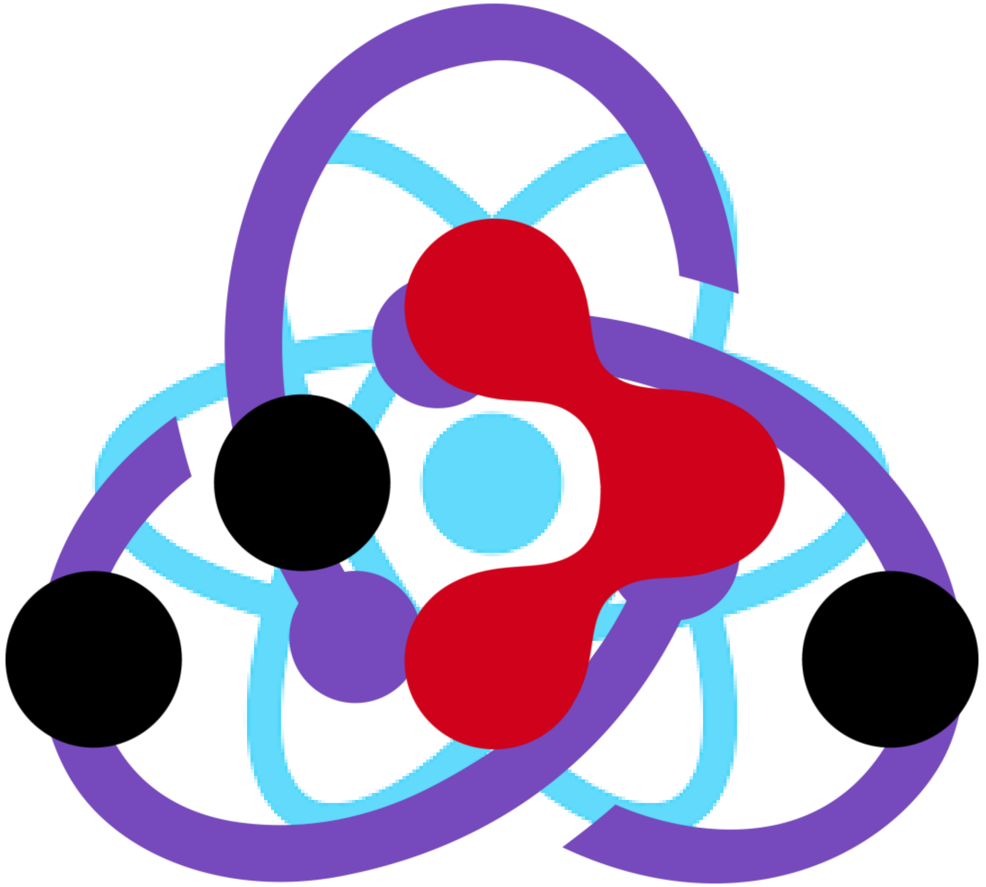
I tried to Learn Too Many Things at Once

Library! Not Framework!

- React, on its own, is a library not a framework
- Pick your router
- Pick your state management
- Pick your build tool
- Pick everything
- Angular was just Angular

nvisia
connect. build. enable.

A La Carte Environment



Break Things Down

React

Redux

Router

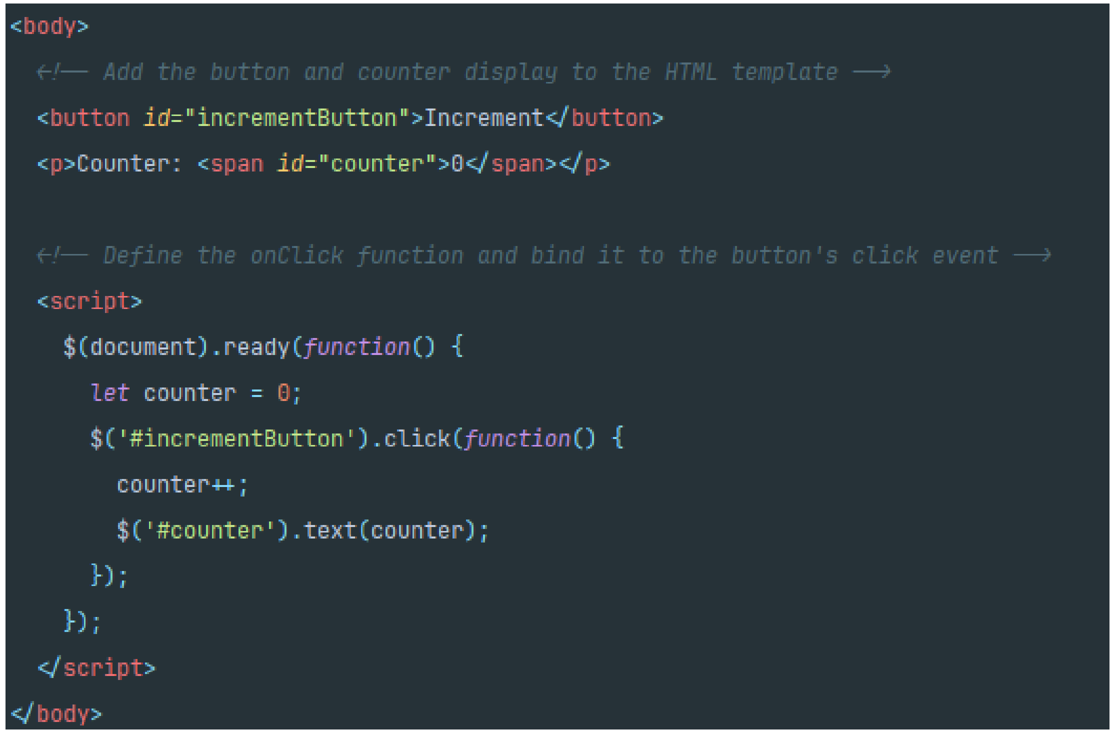
nvisia
connect. build. enable.

I Didn't Think in React

nvisia
connect. build. enable.

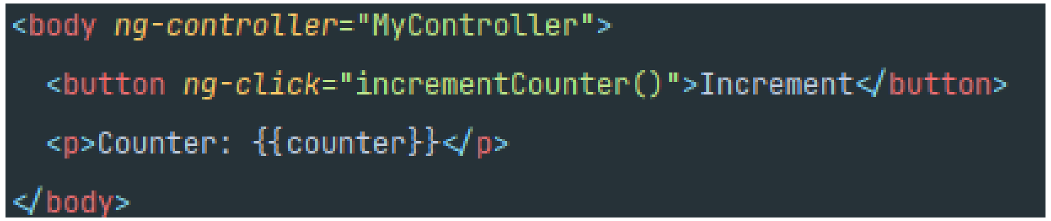
React is Different

JQUERY

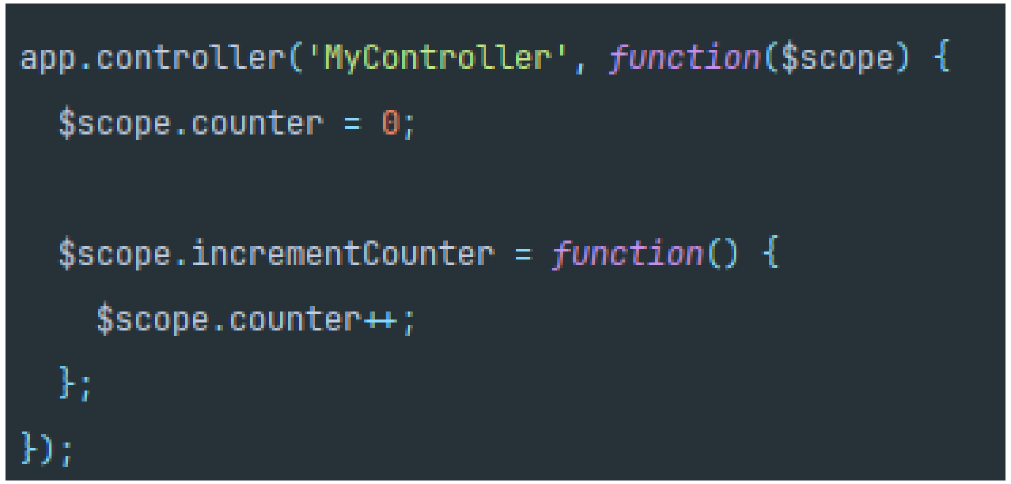


```
<body>
  <!-- Add the button and counter display to the HTML template -->
  <button id="incrementButton">Increment</button>
  <p>Counter: <span id="counter">0</span></p>

  <!-- Define the onClick function and bind it to the button's click event -->
  <script>
    $(document).ready(function() {
      let counter = 0;
      $('#incrementButton').click(function() {
        counter++;
        $('#counter').text(counter);
      });
    });
  </script>
</body>
```

Angular1(js)



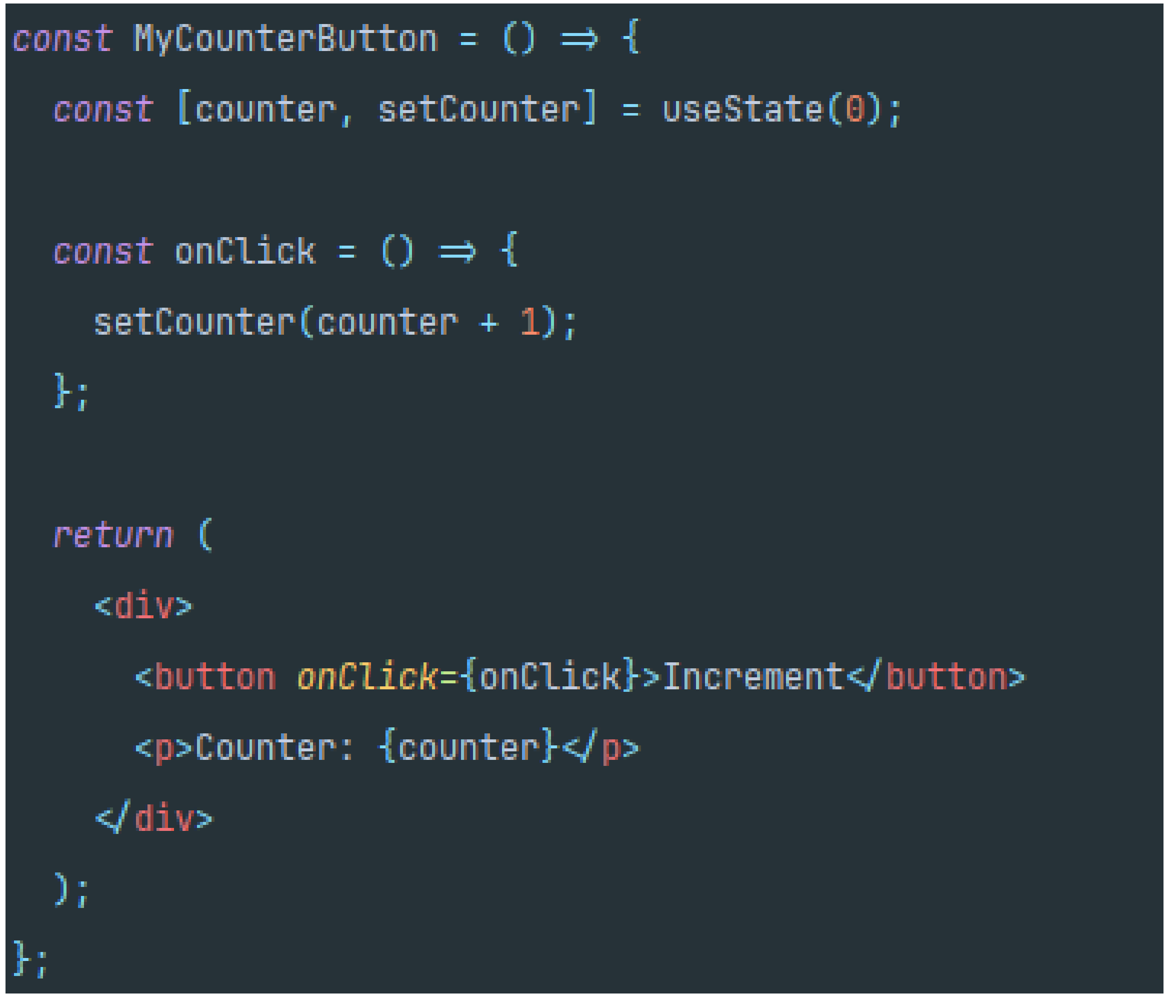
```
<body ng-controller="MyController">
  <button ng-click="incrementCounter()">Increment</button>
  <p>Counter: {{counter}}</p>
</body>
```



```
app.controller('MyController', function($scope) {
  $scope.counter = 0;

  $scope.incrementCounter = function() {
    $scope.counter++;
  };
});
```

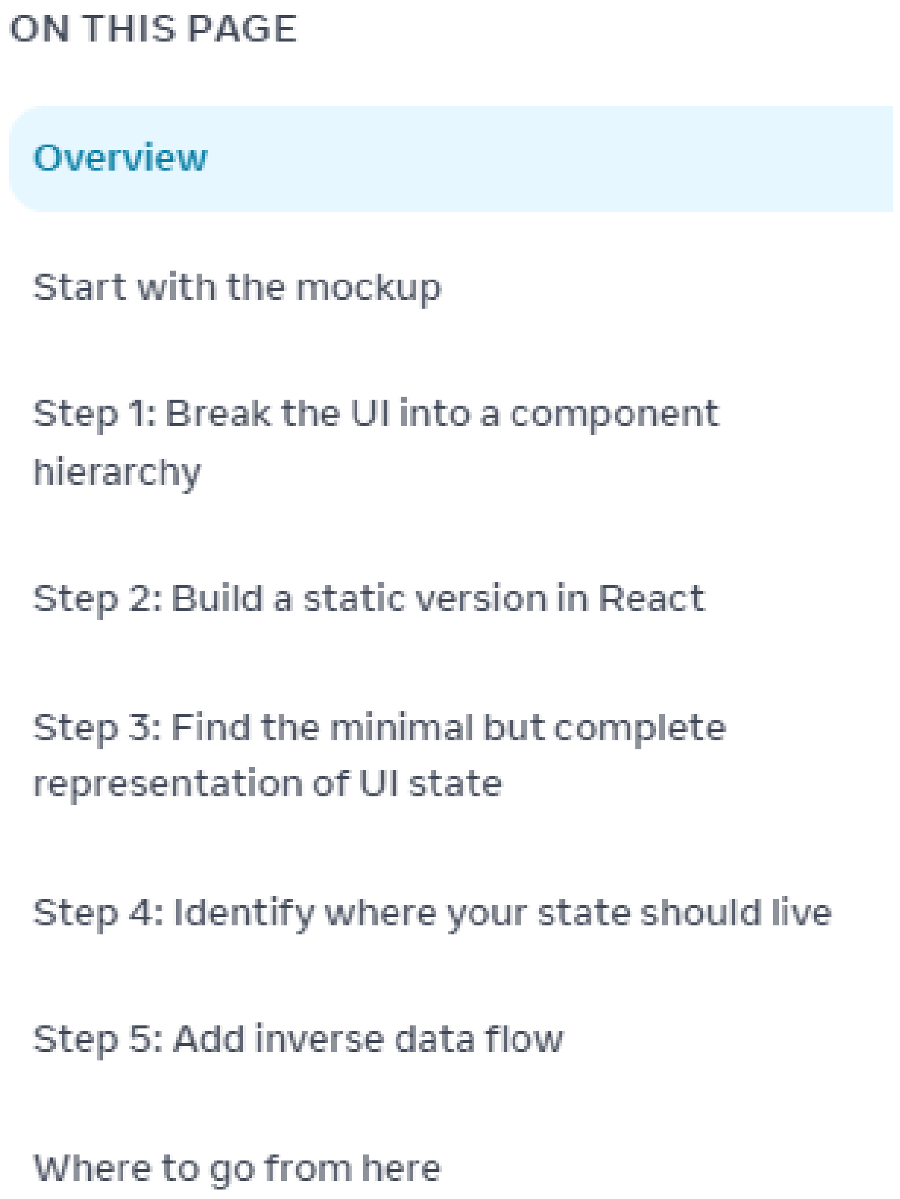
React is Different

- HTML* is the product of the js functions.
- Breaks the UI into components
- Components contain state
- Updates to state trigger a render
- Builds a "virtual DOM" snapshot on render
- Compares new snapshot with previous snapshot
- Paints differences into the "real DOM"



```jsx
const MyCounterButton = () => {
  const [counter, setCounter] = useState(0);

  const onClick = () => {
    setCounter(counter + 1);
  };

  return (
    <div>
      <button onClick={onClick}>Increment</button>
      <p>Counter: {counter}</p>
    </div>
  );
};
```

Thinking In React



ON THIS PAGE

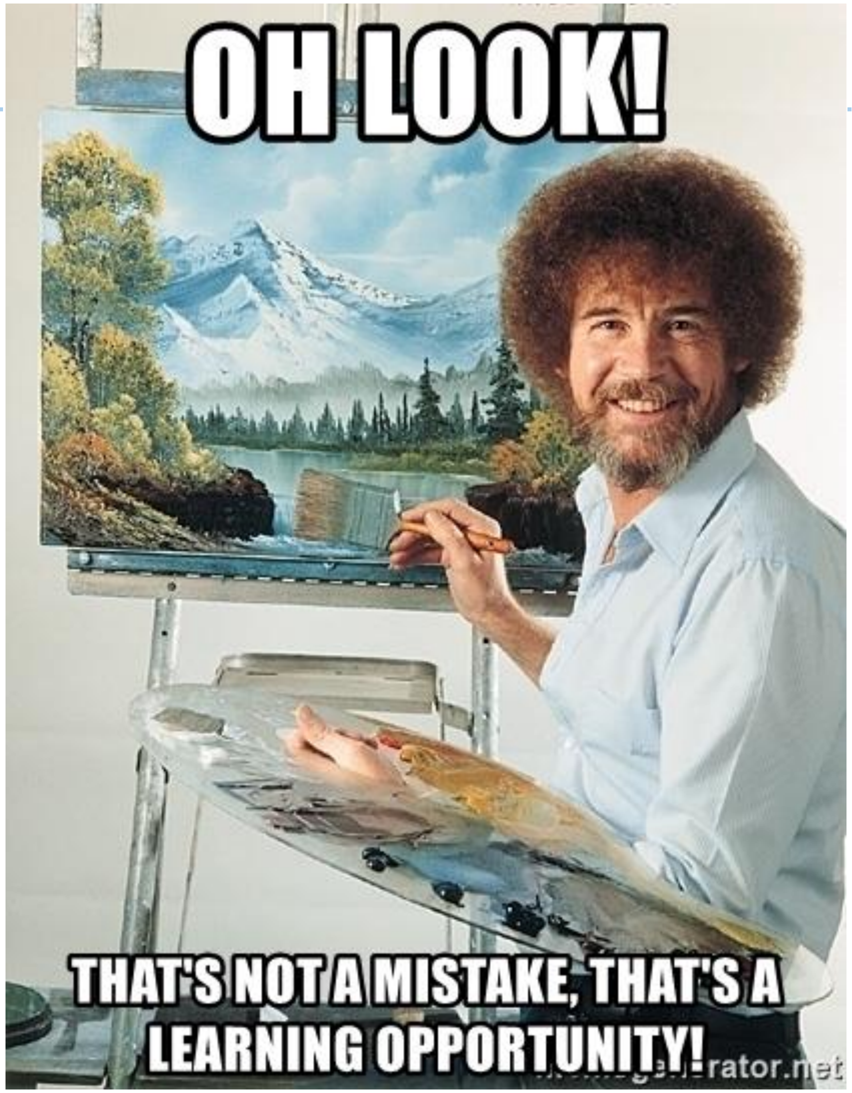
Understand what you're using

If it feels hard to work with you might be doing it wrong

nvisia
connect. build. enable.

Summary

- Break up your components/ think composability
- NEVER define components inside components
- Derive. Don't copy
- Flatten your objects
- Minimize life-cycle logic
- Try to learn one thing at a time
- Think in react



nvisia
connect. build. enable.

Thank You!

nvisia
connect. build. enable.