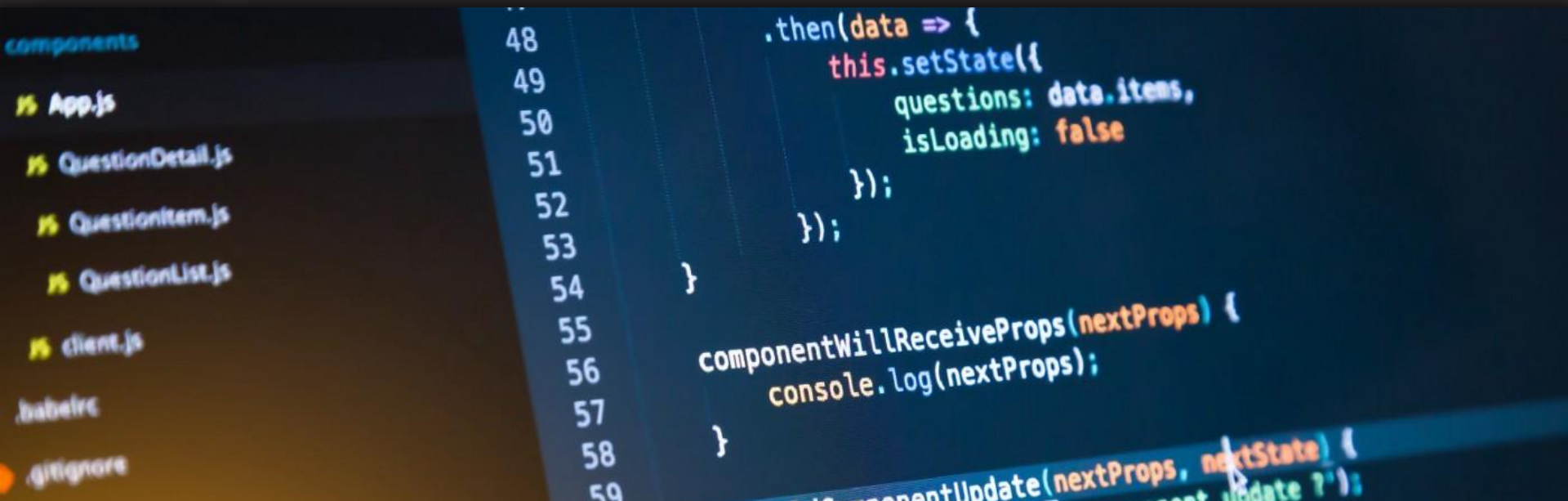




Demystifying React **Context**





Craig Newton

@craignewtondev

Connection Telecom (Pty) Ltd.



Hosted Contact Centre

Need ultimate flexibility and control of your company's internal, external, landline and mobile phone calls – without vendor lock-ins?



VOIPconnect

Want to reap the benefits of cost-efficient call routing with simple technology that integrates PSTN and VoIP?



Omnichannel Inbound Solution

Telviva Touchpoint provides a real-time instant messaging service for new and existing customers to your company.



Professional Services

A smooth migration from existing platforms is a top priority



Cloud PBX

Initially cloud computing was limited to data applications, but with the evolution of VoIP voice services are now also available via the cloud.



Cloud Video Conferencing

Cloud Video Conferencing brings you face to face with your clients

The problem?



We have some data sitting on a component that we need access to far down the component tree.

A solution?

We can pass the piece of data from one component to the next as a prop.

What is prop-drilling?

The pattern of passing props down levels and levels of the component tree when not all of those components necessarily need those props.

A solution?

We can use a library like redux or mobX to manage our application state.



"If you're only using Redux to avoid passing down props, Context could replace Redux - but then you probably didn't need Redux in the first place!"

- Hacker News

DEMO TIME!

The Solution: Context

Context provides a way to pass data through the component tree without having to pass props down manually at every level.

How to use Context?

1. define and initialize a **Context**.
2. setup a **Context Provider** and define the data you want to store;
3. use a **Context Consumer** wherever you need the data from the store.

Create an instance of Context

```
const AppContext = React.createContext();
```

Setup a Context Provider

```
class AppProvider extends Component {  
  state = {  
    number: 1337  
  };  
  
  render() {  
    return (  
      <AppContext.Provider value={this.state}>  
        {this.props.children}  
      </AppContext.Provider>  
    );  
  }  
}
```


Wrap our Container component

```
class Red extends Component {  
  render() {  
    return (  
      <AppProvider>  
        <div className="red-component">  
          <Blue />  
        </div>  
      </AppProvider>  
    );  
  }  
}
```

Setup our Context Consumer and get some state

```
<AppContext.Consumer>  
  {(context) => context.number}  
</AppContext.Consumer>
```

Using Consumer in Green component

```
const Green = () => (  
  <div className="green-component">  
    <AppContext.Consumer>  
      {(context) => context.number}  
    </AppContext.Consumer>  
  </div>  
) ;
```

DEMO TIME!

Using actions to modify data

Generally you will need a way to update/change the data that comes from the React Context.

Defining actions to modify data

```
class AppProvider extends Component {  
  state = {  
    number: 1337,  
    increment: () => {  
      this.setState({number: this.state.number + 1});  
    }  
  };  
  // ...  
}
```


Using actions to modify data

```
const Blue = () => (  
  <div className="blue-component-np">  
    <AppContext.Consumer>  
      {(context) => <button  
onClick={context.increment}>INCREMENT</button>}  
    </AppContext.Consumer>  
    <Green />  
  </div>  
) ;
```

DEMO TIME!

Polyfill for React < 16.3

<https://github.com/jamiebuilds/create-react-context>

```
import createReactContext from 'create-react-context';  
const AppContext = createReactContext();
```

Real world use cases

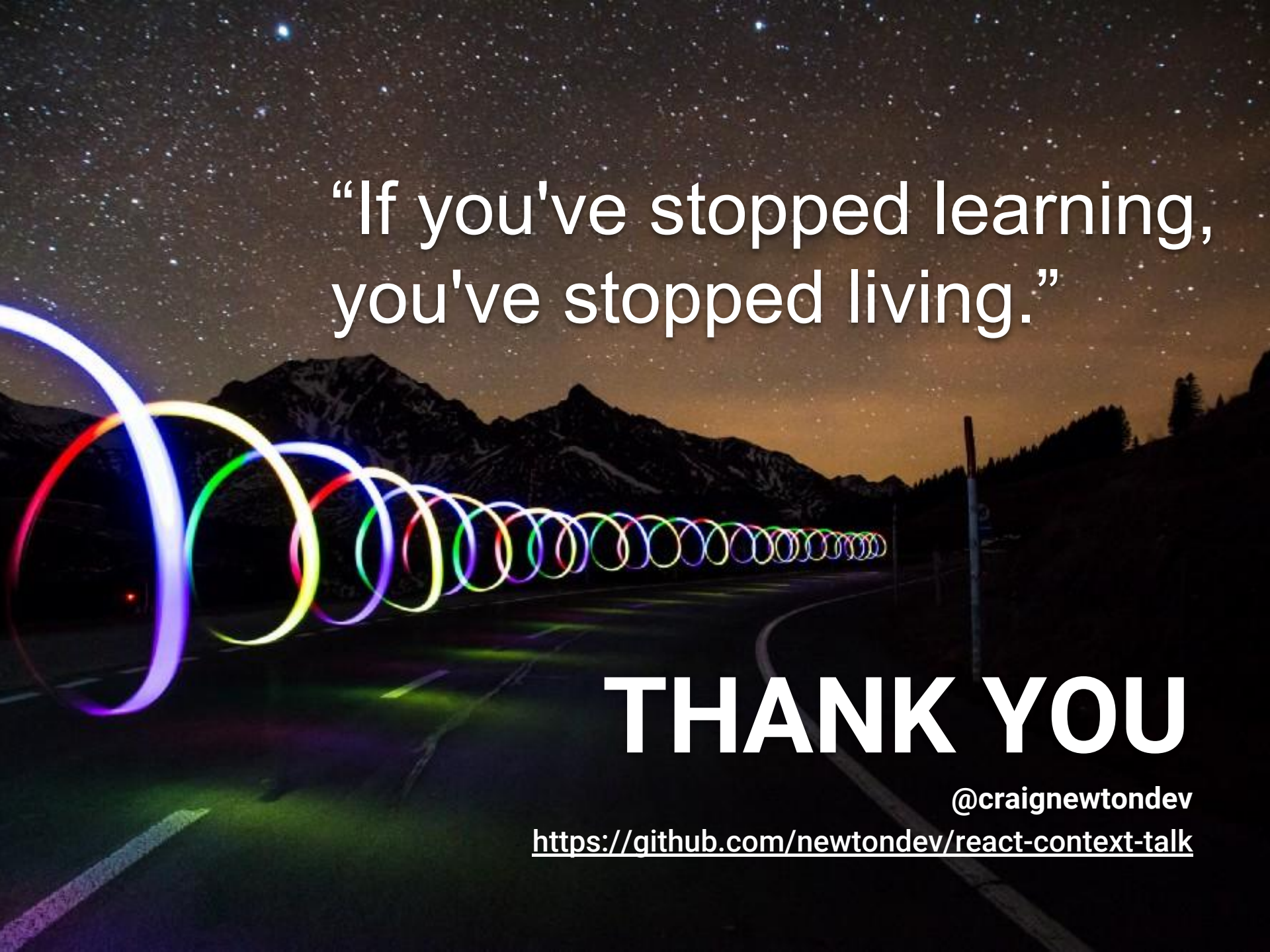
- State of currently authenticated user
- Localization (i18n, language)
- Theming (colours, background images)
- Global application state

Reference Links

<https://github.com/newtondev/react-context-talk>

<https://reactjs.org/docs/context.html>

<https://www.udemy.com/user/sgslo/>



“If you've stopped learning,
you've stopped living.”

THANK YOU

@craignewtondev

<https://github.com/newtondev/react-context-talk>