A BETTER CUSTOMER EXPERIENCE STARTS HERE

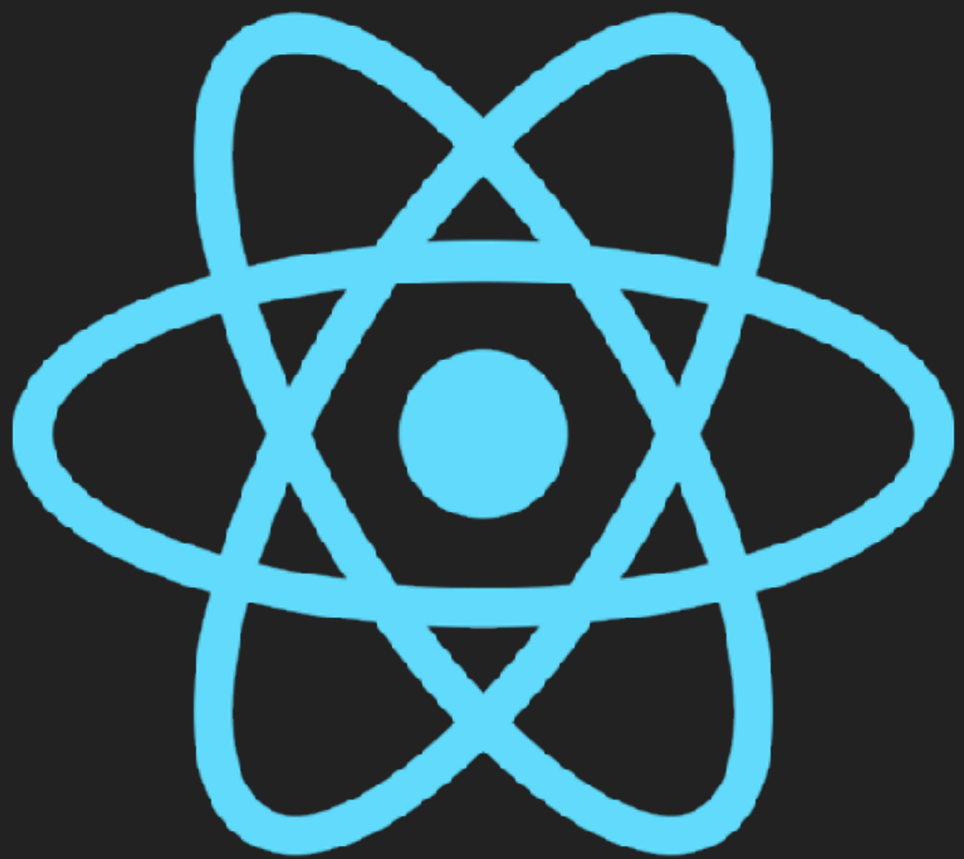
We are an omnichannel customer engagement tool built for the real time web.

Our live chat system is easy to use and requires no training. Your employees will be able to hop right in from any device in the world.

Intelligent customer service powered by your data!

For more information come check us out at help.com/start or shoot me an email, daniel.fernandez@help.com.

Help.com



WHAT'S CHANGED IN REACT

Daniel Fernandez, Twitter @danielHeartsJS

REACT

16.3.0 is going to be packed with new goodies!

New Context API - createContext is going to create a provider and consumer

```
import { createContext } from "react";
const ThemeContext = createContext({
  background: 'yellow',
  color: 'white'
});
```

To use the above provider generated by the above factory you can add in this fashion

```
class Application extends React.Component {

  render() {
    return (
      <ThemeContext.Provider value={{background: 'black', color: 'white'}}>
        <Header />
        <Main />
        <Footer />
      </ThemeContext.Provider>
    )
  }
}
```

REACT

Now to consume the context you would have to use the consumer like so!

This offers a lot more control of what context to expose and allows for deterministic consumption

```
const Header = () => (
  <ThemeContext.Consumer>
    {(context) => {
      return (
        <div style={{background: context.background, color: context.color}}>
          Welcome!
        </div>
      );
    }}
  </ThemeContext.Consumer>
);
```

componentWillMount and componentWillUpdate will be start throwing deprecation warnings. Use componentDidMount and componentDidUpdate instead

REACT

static getDerivedStateFromProps replaces since its getting removed. Since its a static method it does not have access to state or the instance, check out the below example.

```
static getDerivedStateFromProps(nextProps, prevState) {
  if (nextProps.currentRow === prevState.lastRow) {
    return null;
  }
  return {
    lastRow: nextProps.currentRow,
    isScrollingDown: nextProps.currentRow > prevState.lastRow
  };
}
```

Its only a setter not a getter! getDerivedStateFromProps is called on initial rendering and any re-renders

Strict mode is a new way to make sure your code is following the best practices. It's a component available under React.StrictMode and can be added to your application tree or subtree

AsyncMode - Prob should a talk!

Checkout - https://medium.com/@baphemot/whats-new-in-react-16-3-d2c9b7b6193b