Using React Context and the useContext React Hook for Component Data Sharing

Peter Kellner
DEVELOPER, CONSULTANT AND AUTHOR

ReactAtScale.com @pkellner linkedin.com/in/peterkellner99

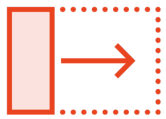
React Context Introduced

Introduced in React version 16.3 in 2018

Context API Benefits

Easier to design

Easier to reason about

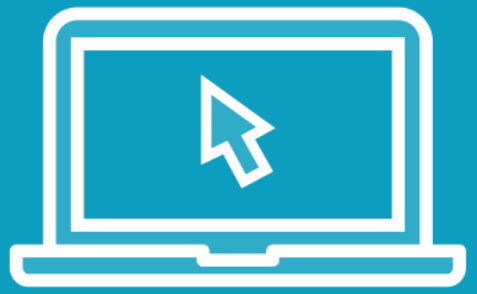


Easier to maintain and extend

Using React Context allows for easily consuming React Hooks for state management.

Using React Context makes it easier to enhance other components.

Demo



Build a new React Context

Replace HOC in our conference app with our new Context

Custom React Hooks

A custom Hook is a JavaScript function whose name starts with ”use” and that may call other Hooks.

React Context with Custom React Hooks

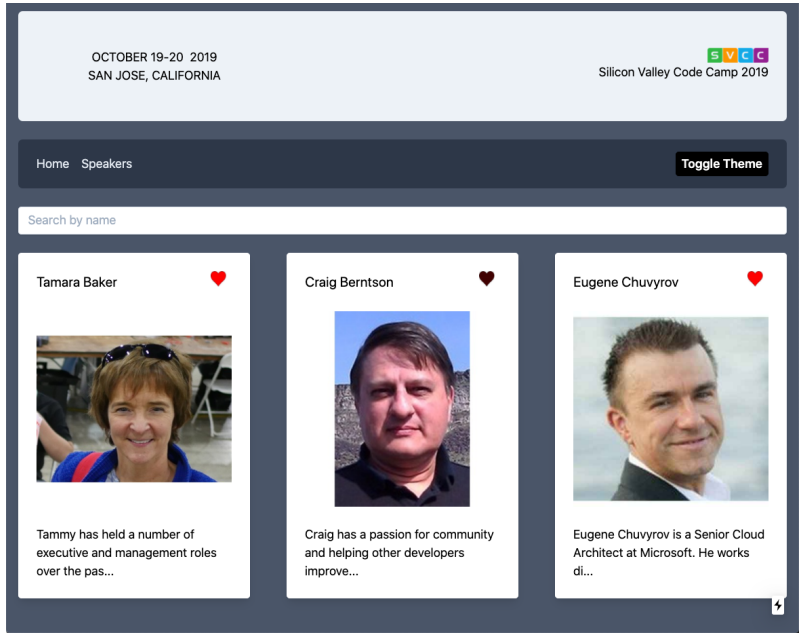
(DataContext internally links to useRequest)

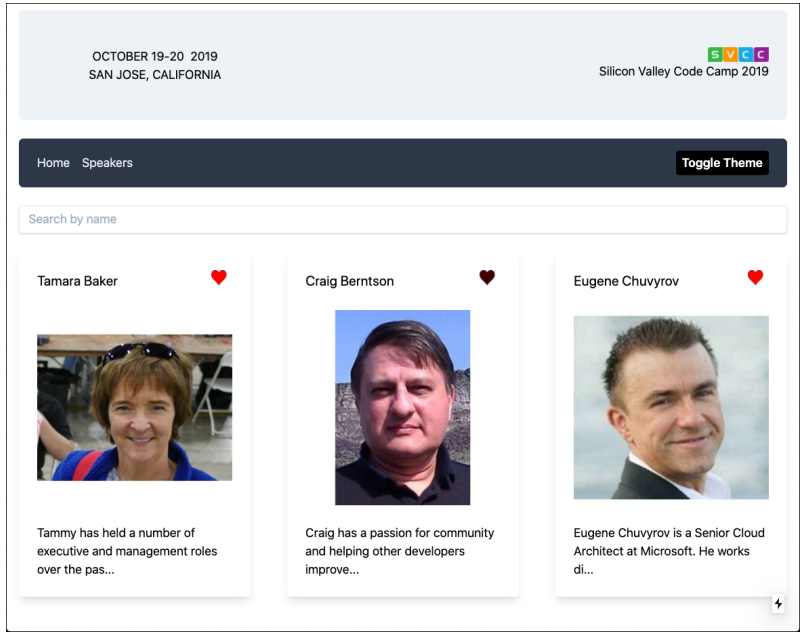
DataContext.js

```
<DataContext url="http://...">
  <Speakers>
       {Rendering UI Happens with
        useRequest generated state
        updates}
  </Speakers>
</DataContext>
```

useRequest.js

```
const useRequest = () => {
  // generating state and updates
  useEffect ( () => {…});
}
```





Light-dark theme

- Theme Context
- useTheme Custom React Hook

Our Layout Component with Context

Layout.js

```
const Layout = ({ children }) => (
  <div className="mx-4 my-3">
    <Header />
    <Menu />
    {children}
    <Footer />
  </div>
);
```

Our Layout Component with Context

Layout.js

```
const Layout = ({ children }) => (
  <div className="mx-4 my-3">
    <Header />
    <Menu />
    {children}
    <Footer />
  </div>
);
```

Our Layout Component With Context

Layout.js

```
const Layout = ({ children }) => (
  <div className="mx-4 my-3">
    <Header />
    <Menu />
    {children}
    <Footer />
  </div>
);
```

```
export default function Page() {
  return (
    <Layout>
      <Home />
    </Layout>
  );
}
```

Our Layout Component With Context

Layout.js

```
const Layout = ({ children }) => (
  <div className="mx-4 my-3">
    <Header />
    <Menu />
    {children}
    <Footer />
  </div>
);
```

```
export default function Page() {
  return (
    <Layout>
      <Home />
    </Layout>
  );
}
```

```
export default function Page() {
  return (
    <Layout>
      <Speakers />
    </Layout>
  );
}
```

Our Layout Component With Context

Layout.js

```
const Layout = ({ children }) => (
  <div className="mx-4 my-3">
    <Header />
    <Menu />
    {children}
    <Footer />
  </div>
);
```

```
export default function Page() {
  return (
    <Layout>
      <Home />
    </Layout>
  );
}
```

```
export default function Page() {
  return (
    <Layout>
      <Speakers />
    </Layout>
  );
}
```

Our Layout Component With Context

Layout.js

```
const Layout = ({ children }) => (
  <ThemeProvider>
    <div className="mx-4 my-3">
      <Header />
      <Menu />
      {children}
      <Footer />
    </div>
</ThemeProvider> );
```

useTheme Custom React Hook

useTheme.js

```
const useTheme = () => {
  const [theme, setTheme] = useState();
  return {
    theme,
    toggleTheme: () => {
      if (theme === THEMELIST.DARK) { setTheme(THEMELIST.LIGHT); }
      else { setTheme(THEMELIST.DARK);}
    },
  };
};
```

Takeaways and Lookaheads

React shares data between components

- Context Object
- Context Provider

Context Provider wraps full or partial app

Context value can contain state

Context gives some state management

Coming up

- React Debugger Extension
- React Error Boundaries