What Is Prop Drilling?

freecodecamp.org/learn/front-end-development-libraries-v9/lecture-react-strategies-and-debugging/what-is-prop-drilling

Prop drilling is the most basic approach to state management in React applications. It looks simple, but can get messy quickly, and is very hard to scale.

Let's look at what prop drilling is, why it's a problem, and a good replacement for it as an application grows.

Prop drilling is the process of passing props from a parent component to deeply nested child components, even when some of the child components don't need the props.

For example, say you have three components named `Parent`, `Child`, and `Grandchild`. If you want to use some data in the `Grandchild` component, but it's in the `Parent` component, you'd need to pass it from the `Parent` to the `Child` component, then from the `Child` to the `Grandchild` component.

Or if the data is even further up the chain, the data might have to be passed to the `Parent` component, too.

Here, the data I want to display is the string `Hello, Prop Drilling!`. It's assigned to the `greeting` variable in the root `App` component:

```
import "./App.css";
import Parent from "./Parent";

function App() {
  const greeting = "Hello, Prop Drilling!";

  return <Parent greeting={greeting} />;
}

export default App;
```

You can see the `Parent` component is also receiving the `greeting` variable as the value of a `greeting` prop. Here's the `Parent` component passing it into the `Child` component as the value of another `greeting` prop in the `Child`:

```
import Child from "./Child";

const Parent = ({ greeting }) => {
  return <Child greeting={greeting} />;
};

export default Parent;
```

And here's the `Child` component that passes it to the `Grandchild` component:

```
import Grandchild from "./Grandchild";

const Child = ({ greeting }) => {
  return <Grandchild greeting={greeting} />;
};

export default Child;
```

And finally the `Grandchild` component receives the greeting and uses it as the content of an `h1` element:

```
const Grandchild = ({ greeting }) => {
  return <h1>{greeting}</h1>;
};

export default Grandchild;
```

In the browser, you'll see a page with a single `h1` element that has the text `Hello, Prop Drilling!`.

At first, prop drilling might not seem like such a big deal. But as your app grows, it gets harder to understand, debug, and maintain.

If you need to pass props around, try to keep them all in a single parent component. This approach of centralizing all necessary data is called the "single source of truth".

For instance, say you want to add a new `response` to go with your `greeting`, and that you want to use both of them in the `Grandchild` component. Since `greeting` is already in the `App` component, it makes sense to put `response` there, too, and pass both of them down the chain:

```
function App() {
  const greeting = "Hello, Prop Drilling!";
  const response = "I'm not here to play!";

  return <Parent greeting={greeting} response={response} />;
}

const Parent = ({ greeting, response }) => {
  return <Child greeting={greeting} response={response} />;
};

const Child = ({ greeting, response }) => {
  return <Grandchild greeting={greeting} response={response} />;
};

const Grandchild = ({ greeting, response }) => {
  return (
    <>
      <h1>{greeting}</h1>
      <h2>{response}</h2>
    </>
  );
};

export default App;
```

In the browser, you'll see a page with an `h1` element that has the text `Hello, Prop Drilling!` and an `h2` element that has the text `I'm not here to play!`.

To avoid prop drilling, especially in large, complex applications, consider using the Context API or state management libraries like Redux and Redux Toolkit, Zustand, Recoil, and others.

You'll learn more about these in the coming lessons.

Questions

How would a prop flow from a parent to a grandchild component?

By defining the prop inside the grandchild component.

By passing it from parent to child, then from child to grandchild.

Correct!

By using the `useEffect` hook to fetch the prop dynamically.

By using the `useState` hook in the grandchild.

What is prop drilling in React?

Passing props directly to only the components that need them.

Using context to share state between components.

Passing props from a parent to deeply nested child components.

Correct!

Drilling down into component state using hooks.

Why is prop drilling considered a problem in larger applications?

It makes it easier to manage state.

It improves performance by reducing re-renders.

It makes the code harder to read, debug, and maintain.

Correct!

It eliminates the need for state management libraries.