NeoSOFT®

React Props

Introduction

Props (short for "properties") are used to pass data from a parent component to a child component in React. They are read-only and help make components reusable and dynamic.

Example of Using Props

Parent Component (Passing Props)

```jsx
function Parent() {
  return <Child name="Alice" age={25} />;
}
```

Child Component (Receiving Props)

```jsx
function Child({ name, age }) {
  return <h2>Hello, my name is {name} and I am {age} years old.</h2>;
}
```

Key Points about Props

- Props allow data flow from parent to child.
- Props are immutable (cannot be modified by child components).
- Props help make components reusable.
- Props can be strings, numbers, arrays, objects, or functions

Props Drilling

Props are passed through multiple levels in a component hierarchy. Each component receives and passes props further. This concept is known as Props Drilling.

This parent component needs to pass data deeply to nested child component through multiple intermediary components, even if some of those intermediary components don’t actually need the data.

Problems with Props Drilling

- **Unnecessary Passing**: Components that don't need the data must still accept and pass it.
- **Reduced Maintainability**: If you need to add or remove props, you have to update multiple components.
- **Harder Refactoring**: Deeply nested components are harder to restructure.

```
Parent Component (App.js)
  ├── Passes props → <Parent name="Alice" />
  ↓
Parent Component (Parent.js)
  ├── Receives props → function Parent({ name })
  ├── Passes props → <Child name={name} />
  ↓
Child Component (Child.js)
  ├── Receives props → function Child({ name })
  ├── Passes props → <GrandChild name={name} />
  ↓
GrandChild Component (GrandChild.js)
  ├── Receives props → function GrandChild({ name })
  ├── Uses props → return <h2>Hello, {name}!</h2>
```

Solutions to Avoid Props Drilling

Context API (Built-in React)

Instead of passing props manually, React's Context API allows you to share values directly.

State Management Libraries (Redux, Zustand, Recoil)

For larger applications, using a global state management solution like **Redux** or **Zustand** can help.