

WEB APP DEVELOPMENT WITH REACTJS (INT252)

Lecture 17: Context API – Global State Management

Unit V – State Management & Advanced React

Syllabus Mapping:

- Context API
- Global state management
- Avoiding props drilling

Target Learner: Beginner → Intermediate

Why Global State Is Needed

So far we passed data using props:

App → Header → Navbar → Button

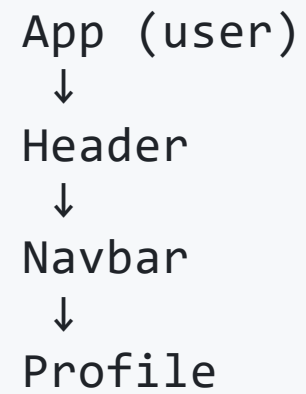
This becomes painful.

What is Props Drilling?

Props drilling means:

- Passing props through many levels
- Even when middle components don't need it

Props Drilling Visual



```
graph TD; A[App (user)] --> B[Header]; B --> C[Navbar]; C --> D[Profile];
```

App (user)
↓
Header
↓
Navbar
↓
Profile

Messy and hard to maintain ❌

Solution: Context API

Context API allows:

- Sharing data globally
- Without passing props manually

What is Context?

Context is:

- A global store
- Accessible by any component

Real-World Use Cases

- User authentication
- Theme (dark/light)
- Language selection
- App settings

Context API Components

1. createContext
2. Provider
3. Consumer (useContext)

Step 1: Create Context

```
import { createContext } from 'react';  
  
export const UserContext = createContext();
```


Step 2: Provide Context

```
<UserContext.Provider value="Aman">  
  <App />  
</UserContext.Provider>
```


Step 3: Consume Context (useContext)

```
import { useContext } from 'react';  
  
const user = useContext(UserContext);
```


Complete Example

```
// useContext.js  
export const useContext = createContext();
```

```
// main.jsx  
<useContext.Provider value="Aman">  
  <App />  
</useContext.Provider>
```

```
// Profile.jsx  
const user = useContext(useContext);
```


Context Data Flow

Provider



Any Child Component

Using State with Context

```
const [user, setUser] = useState(null);  
  
<UserContext.Provider value={{ user, setUser }}>
```


Consuming Object Context

```
const { user, setUser } = useContext(UserContext);
```


Updating Context Data

```
setUser('Admin');
```

Updates everywhere

Common Beginner Mistakes

- ✗ Forgetting Provider
- ✗ Wrong context import
- ✗ Overusing context

When NOT to Use Context

- Frequently changing state
- Very local component state

Practice Exercises

1. Create ThemeContext
2. Toggle dark/light mode
3. Consume context in two components

✓ Answers – Practice Exercises

```
const ThemeContext = createContext();
```

```
<ThemeContext.Provider value={{ theme, setTheme }}>
```

```
const { theme } = useContext(ThemeContext);
```


Key Takeaways

- Context solves props drilling
- Provider supplies global data
- useContext consumes data

Next Lecture

Lecture 18: Performance Optimization & Memoization

Unit V – State Management & Advanced React