React Context API — Beginner-Friendly Tutorial (Step-by-Step)

This version is written **for freshers**, with **diagrams, full file structure, and complete component code** for both:

- ❌ Props Drilling version
- ✅ Context API version

PART 1 — UNDERSTANDING THE PROBLEM (PROPS DRILLING)

1. What is Props Drilling? (Simple Explanation)

Props drilling refers to the process of passing data from a parent component to deeply nested child components through multiple layers of intermediate components—even when those intermediate components do not need the data themselves.

Simple analogy:

```
App (has water tank)
  |
  v
Floor 1 (doesn't need water)
  |
  v
Floor 2 (doesn't need water)
  |
  v
Floor 3 (needs water)
```

Water = data (username). Even though only Floor 3 needs water, Floor 1 and Floor 2 must still carry it. This is exactly what props drilling is in React.

---

2. Component Tree Diagram

```
App
 └── Profile
     └── UserCard
         └── Username
```

Only **Username** needs the name, but we will pass it through all components.

PART 2 — PROJECT 1: PROPS DRILLING VERSION (FULL CODE)

📁 File Structure

```
src/
  App.js
  components/
    Profile.js
    UserCard.js
    Username.js
```

📄 src/App.js

```
import React, { useState } from "react";
import Profile from "./components/Profile";

function App() {
  const [username, setUsername] = useState("John Doe");

  return (
    <div style={{ padding: "20px", border: "2px solid black" }}>
      <h2>App Component (Parent)</h2>
      <Profile username={username} />
    </div>
  );
}

export default App;
```

📄 src/components/Profile.js

```
import React from "react";
import UserCard from "./UserCard";

function Profile({ username }) {
  return (
    <div style={{ padding: "20px", border: "2px solid blue" }}>
      <h3>Profile Component</h3>
```

```
      <UserCard username={username} />
    </div>
  );
}

export default Profile;
```

📄 src/components/UserCard.js

```
import React from "react";
import Username from "./Username";

function UserCard({ username }) {
  return (
    <div style={{ padding: "20px", border: "2px solid green" }}>
      <h4>UserCard Component</h4>
      <Username username={username} />
    </div>
  );
}

export default UserCard;
```

📄 src/components/Username.js

```
import React from "react";

function Username({ username }) {
  return (
    <div style={{ padding: "20px", border: "2px solid red" }}>
      <h5>Username Component</h5>
      <p>Welcome, {username}</p>
    </div>
  );
}

export default Username;
```

Why is this bad? (For students)

- Profile and UserCard **do not use username**.
- They are just passing it down.
- If we add more layers, it becomes messy.

PART 3 — SOLUTION: CONTEXT API VERSION

📁 New File Structure

```
src/
  App.js
  context/
    UserContext.js
  components/
    Profile.js
    UserCard.js
    Username.js
```

📄 src/context/UserContext.js

```
import React from "react";

export const UserContext = React.createContext();
```

📄 src/App.js (Using Context)

```
import React, { useState } from "react";
import { UserContext } from "./context/UserContext";
import Profile from "./components/Profile";

function App() {
  const [username, setUsername] = useState("John Doe");

  return (
    <UserContext.Provider value={{ username, setUsername }}>
      <div style={{ padding: "20px", border: "2px solid black" }}>
        <h2>App Component (Provider)</h2>
        <Profile />
      </div>
    </UserContext.Provider>
  );
}

export default App;
```

📄 src/components/Profile.js (No Props)

```
import React from "react";
import UserCard from "./UserCard";

function Profile() {
  return (
    <div style={{ padding: "20px", border: "2px solid blue" }}>
      <h3>Profile Component</h3>
      <UserCard />
    </div>
  );
}

export default Profile;
```

📄 src/components/UserCard.js (No Props)

```
import React from "react";
import Username from "./Username";

function UserCard() {
  return (
    <div style={{ padding: "20px", border: "2px solid green" }}>
      <h4>UserCard Component</h4>
      <Username />
    </div>
  );
}

export default UserCard;
```

📄 src/components/Username.js (Consumes Context)

```
import React, { useContext } from "react";
import { UserContext } from "../context/UserContext";

function Username() {
  const { username } = useContext(UserContext);

  return (
    <div style={{ padding: "20px", border: "2px solid red" }}>
      <h5>Username Component</h5>
      <p>Welcome, {username}</p>
```

```
    </div>
  );
}

export default Username;
```

Diagram — Before vs After

🔩 Before (Props Drilling)

```
App -> Profile -> UserCard -> Username
  (username passed through all)
```

🔗 After (Context API)

```
        App (Provider)
            |
   -------------------------
   |        |        |
Profile   UserCard  Username (Consumer)
```

Username directly reads data from Context.

PART 4 — REAL PROJECT EXAMPLE: THEME SWITCHER

📁 File Structure

```
src/
  App.js
  context/
    ThemeContext.js
  components/
    Navbar.js
    Card.js
    Footer.js
```

📄 src/context/ThemeContext.js

```
import React from "react";
export const ThemeContext = React.createContext();
```

📄 src/App.js

```
import React, { useState } from "react";
import { ThemeContext } from "./context/ThemeContext";
import Navbar from "./components/Navbar";
import Card from "./components/Card";
import Footer from "./components/Footer";

function App() {
  const [theme, setTheme] = useState("light");

  return (
    <ThemeContext.Provider value={{ theme, setTheme }}>
      <div style={{ padding: "20px" }}>
        <h2>Theme App</h2>
        <Navbar />
        <Card />
        <Footer />
      </div>
    </ThemeContext.Provider>
  );
}

export default App;
```

📄 src/components/Navbar.js

```
import React, { useContext } from "react";
import { ThemeContext } from "../context/ThemeContext";

function Navbar() {
  const { theme, setTheme } = useContext(ThemeContext);

  return (
    <div style={{ padding: "10px", border: "1px solid gray" }}>
      <h3>Navbar</h3>
      <button onClick={() => setTheme(theme === "light" ? "dark" : "light")}>
        Toggle Theme
      </button>
    </div>
  );
}
```

```
export default Navbar;
```

📄 src/components/Card.js

```
import React, { useContext } from "react";
import { ThemeContext } from "../context/ThemeContext";

function Card() {
  const { theme } = useContext(ThemeContext);

  return (
    <div style={{
      padding: "20px",
      margin: "10px",
      background: theme === "light" ? "white" : "black",
      color: theme === "light" ? "black" : "white",
      border: "1px solid gray",
    }}>
      <h3>Card</h3>
      <p>Current Theme: {theme}</p>
    </div>
  );
}

export default Card;
```

📄 src/components/Footer.js

```
import React, { useContext } from "react";
import { ThemeContext } from "../context/ThemeContext";

function Footer() {
  const { theme } = useContext(ThemeContext);

  return (
    <div style={{ padding: "10px", border: "1px solid gray" }}>
      <h4>Footer - Theme: {theme}</h4>
    </div>
  );
}

export default Footer;
```

Comparing Props Drilling vs Context API

Feature	Props Drilling	Context API
Code readability	Poor	Good
Scalability	Difficult	Easier
Component coupling	High	Low
Data access	Must pass manually	Direct access

PART 5 — STUDENT EXERCISES

Exercise 1 — Practice Props Drilling

Create: App → Dashboard → Sidebar → UserName Pass name through all components.

Exercise 2 — Convert to Context API

Use Context to remove props drilling.

Exercise 3 — Counter App with Context

- Create CounterContext
- Store count in App
- Create IncrementButton, DecrementButton, DisplayCount
- All components use Context