React Context Pattern - MEMORIZE THIS

Step 1: Create Context File (FormContext.jsx)

jsx

```jsx
// ALWAYS import both createContext and useState
import { createContext, useState } from 'react';

// Create the context - this is your data highway
const FormContext = createContext();

// ALWAYS export default the context
export default FormContext;

// Provider component - this wraps your app and provides data
export function FormProvider({ children }) {
  // State that will be shared across components
  const [formData, setFormData] = useState({
    name: '',
    email: ''
  });

  // Return the Provider with value prop containing your data/functions
  return (
    <FormContext.Provider value={{ formData, setFormData }}>
      {children}
    </FormContext.Provider>
  );
}
```

Step 2: Wrap Your App (main.jsx)

```jsx
// Import your Provider (not the Context itself)
import { FormProvider } from './FormContext';
import App from './App';

// Wrap your entire app with the Provider
<FormProvider>
  <App />
</FormProvider>
```

Step 3: Consume Context (Any Component)

```jsx
// ALWAYS import useContext hook and your Context
import { useContext } from 'react';
import FormContext from './FormContext';

function MyComponent() {
  // Destructure what you need from context
  const { formData, setFormData } = useContext(FormContext);

  // To update data - ALWAYS spread existing data first
  const handleChange = (e) => {
    setFormData({
      ...formData,           // Keep existing data
      name: e.target.value   // Update only what changed
    });
  };

  // Use the data
  return (
    <input
      value={formData.name}
      onChange={handleChange}
    />
  );
}
```

MEMORIZE THESE EXACT PATTERNS:

1. Context Creation Pattern:

```jsx
const ContextName = createContext();
export default ContextName;
```

2. Provider Pattern:

```jsx
export function ContextNameProvider({ children }) {
  const [state, setState] = useState(initialValue);
  return (
    <ContextName.Provider value={{ state, setState }}>
      {children}
    </ContextName.Provider>
  );
}
```

3. Consumer Pattern:

```jsx
const { state, setState } = useContext(ContextName);
```

4. Update Pattern:

```jsx
setState({ ...state, propertyName: newValue });
```

CRITICAL RULES TO MEMORIZE:

1. **ALWAYS** import `useState` with `createContext`
2. **ALWAYS** export default the Context
3. **ALWAYS** use `{ children }` in Provider
4. **ALWAYS** spread existing state when updating: `{ ...state, newProp: value }`
5. **ALWAYS** import `useContext` when consuming

6. **ALWAYS** destructure what you need from context

COMMON MISTAKES TO AVOID:

- ❌ `{chidren}` → ✅ `{children}`
- ❌ `setFormData({name: value})` → ✅ `setFormData({...formData, name: value})`
- ❌ `<formprovider>` → ✅ `<FormProvider>`
- ❌ Forgetting to import `useContext`
- ❌ Forgetting to import `useState` in context file