10. React Project Structure and Best Practices

Last updated by | Subramanya Dixit | May 5, 2025 at 8:57 PM GMT+5:30

📘 Why Structure Matters

A well-organized project structure improves code readability, scalability, and team collaboration, especially as applications grow larger.

🧠 Key Concepts

✅ Suggested Folder Structure

```
my-app/
├── public/
├── src/
│   ├── assets/
│   ├── components/
│   ├── pages/
│   ├── hooks/
│   ├── contexts/
│   ├── services/
│   ├── utils/
│   ├── App.jsx
│   └── main.jsx
├── .gitignore
├── package.json
└── README.md
```

✅ Separation of Concerns

Keep logic, presentation, and data-fetching responsibilities in separate files and folders.

✅ Naming Conventions

- Files and folders: `camelCase` or `PascalCase`
- Components: `PascalCase`
- Hooks: start with `use`

✅ Custom Hooks

Encapsulate reusable logic into the `hooks/` directory.

```
// hooks/useToggle.js
import { useState } from 'react';
export default function useToggle(initial = false) {
  const [state, setState] = useState(initial);
  const toggle = () => setState(s => !s);
  return [state, toggle];
}
```

✅ Environment Variables

Place API keys or config in `.env` files.

```
VITE_API_URL=https://api.example.com
```

Access with `import.meta.env.VITE_API_URL`

📊 Visual Overview

```
flowchart TD
  A[Components] -->|Used in| B[Pages]
  B -->|Rendered in| C[App.jsx]
  D[Hooks] -->|Imported by| A
  E[Contexts] -->|Provide state| A
  F[Services] -->|Fetch data| A
```

💡 Best Practices

- Keep components small and reusable.
- Group related files together (Component, style, test).
- Use meaningful names.
- Avoid deeply nested folders.
- Use `index.js` for barrel exports in folders.
- Leverage lazy loading for routes.

📝 Practice Exercises

1. Create a new React project with a clean folder structure.
2. Create a custom hook and use it in a component.
3. Refactor a file into its appropriate directory.
4. Setup an `.env` file and read an environment variable.

❓ Quiz Questions

1. Why is a good folder structure important?
a) Helps with animations
b) Reduces app size
✅ **c) Improves maintainability and scalability**
d) Required by React

---

2. What should a custom hook file start with?
a) `custom_`
b) `newHook_`

✅ **c)** `use`
d) `React_`

---

3. Where should global state providers like context go?

a) Inside `App.jsx` only
b) Under `services/`
✅ **c) Under `contexts/`**
d) Inside `utils/`

---

4. How do you access a variable in a `.env` file (Vite)?

a) `process.env.API_URL`
b) `env.API_URL`
✅ **c)** `import.meta.env.VITE_API_URL`
d) `getEnv('API_URL')`
