React JS Coding Standards & Best Practices (2025)

1. Project Structure

Maintain a clean, modular folder structure. Below is a recommended standard structure for medium to large React projects:

```
src/
■■■ api/
■■■ assets/
■■■ components/
■■■ hooks/
■■■ context/
■■■ pages/
■■■ utils/
■■■ constants/
■■■ App.jsx
```

2. Component Best Practices

Use functional components, maintain SRP (Single Responsibility Principle), and keep UI clean and reusable.

Example: Clean Functional Component

```
import React from "react";

export const UserCard = ({ user }) => {
  return (
    <div className="user-card">
      <h3>{user.name}</h3>
      <p>Email: {user.email}</p>
    </div>
  );
};
```

3. Custom Hooks Example

Encapsulate reusable logic inside custom hooks:

```
import { useState, useEffect } from "react";

export const useFetch = (url) => {
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetch(url)
      .then((res) => res.json())
      .then((res) => {
```

```
        setData(res);
        setLoading(false);
      });
  }, [url]);

  return { data, loading };
};
```

4. Constants Management

Separate all constant UI labels, API endpoints, and static values into a dedicated constants file.

```
// constants/labels.js
export const LABELS = {
  DASHBOARD: "Dashboard",
  NOTIFICATION: "Notification",
  CHECK_ALL: "Check All",
};
```

```
// constants/api.js
export const API_ENDPOINTS = {
  USERS: "/api/users",
  LOGIN: "/api/login",
};
```

This document contains recommended standards for scalable, maintainable React applications.