React BlogApp – Setup & Guide

# 1. Create a New React Application

Open your terminal and run:

npx create-react-app blogapp

# 2. Open the Application in VS Code

In your terminal:

cd blogapp  
code .

Note: Make sure you have VS Code installed and added to your system path.

# 3. Create the Post.js Class

In the src folder, create a new file named Post.js and add the following code:

// src/Post.js  
class Post {  
 constructor(userId, id, title, body) {  
 this.userId = userId;  
 this.id = id;  
 this.title = title;  
 this.body = body;  
 }  
}  
  
export default Post;

# 4. Create the Posts.js Class Component

In the src folder, create a new file named Posts.js and add the following code:

// src/Posts.js  
import React, { Component } from 'react';  
import Post from './Post';  
  
class Posts extends Component {  
 constructor(props) {  
 super(props);  
 this.state = { posts: [] };  
 }  
  
 loadPosts = () => {  
 fetch('https://jsonplaceholder.typicode.com/posts')  
 .then(response => {  
 if (!response.ok) throw new Error('Network error');  
 return response.json();  
 })  
 .then(data => {  
 const posts = data.map(  
 post => new Post(post.userId, post.id, post.title, post.body)  
 );  
 this.setState({ posts });  
 })  
 .catch(error => { throw error; });  
 };  
  
 componentDidMount() {  
 this.loadPosts();  
 }  
  
 componentDidCatch(error, info) {  
 alert('An error occurred: ' + error.message);  
 }  
  
 render() {  
 return (  
 <div>  
 <h1>Blog Posts</h1>  
 {this.state.posts.map(post => (  
 <div key={post.id} style={{ marginBottom: "40px" }}>  
 <h2>{post.title}</h2>  
 <p>{post.body}</p>  
 <hr/>  
 </div>  
 ))}  
 </div>  
 );  
 }  
}  
  
export default Posts;

# 5. Add the Posts Component to App

Open src/App.js and replace its contents with:

import React from 'react';  
import Posts from './Posts';  
  
function App() {  
 return (  
 <div className="App">  
 <Posts />  
 </div>  
 );  
}  
  
export default App;

# 6. Run the Application

Back in your terminal, start the development server:

npm start

Your browser should open (or go to http://localhost:3000) and display the posts fetched from the API.

# 7. Optional: Build for Production

When you're ready to deploy:

npm run build

This creates a production-ready version of your app.

# Summary

You create the app, define a Post class and a Posts class-based component (with data fetching and error handling), integrate them into App.js, and run the project. Follow these steps to have a working blog application using React's class components and best practices.