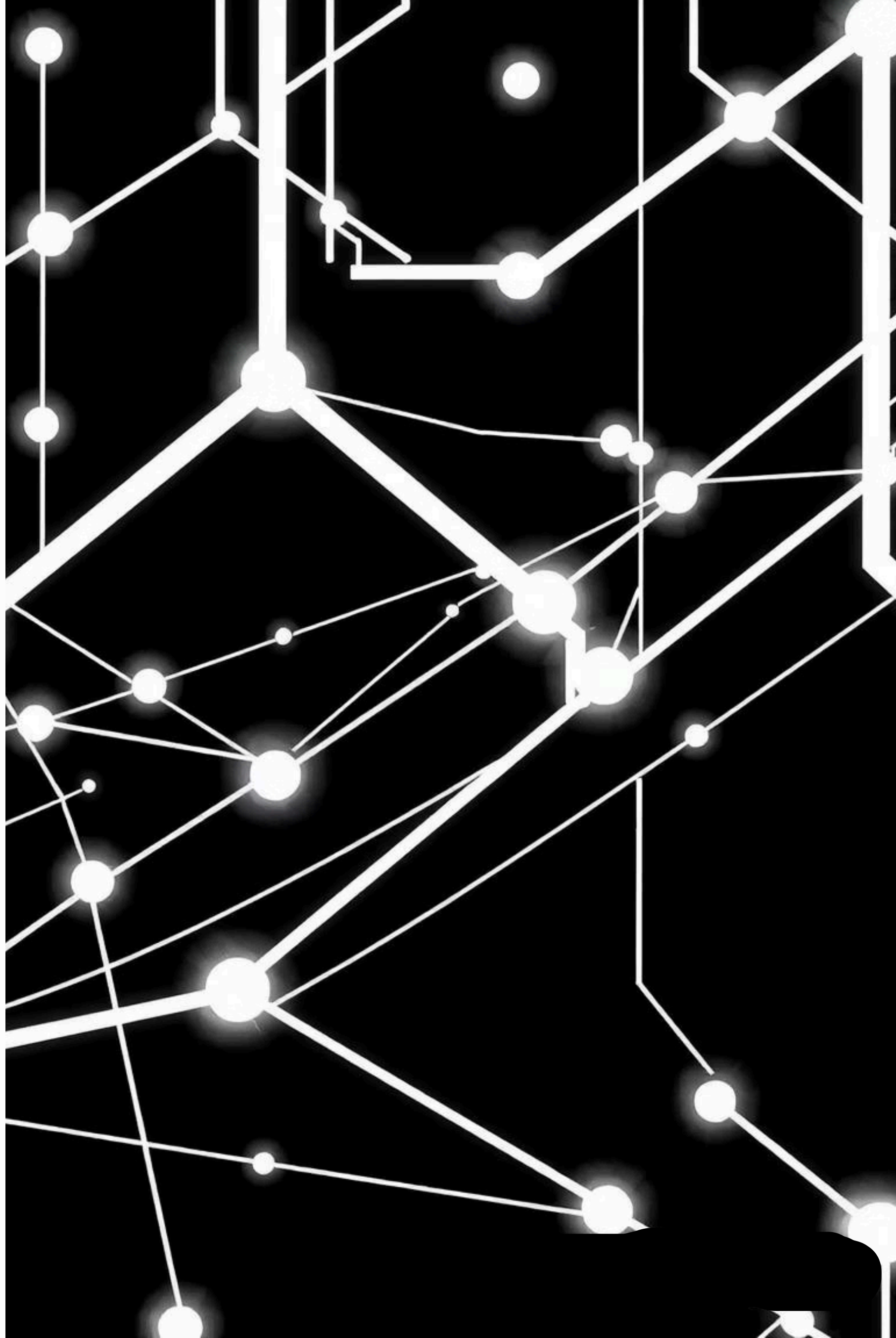
Mastering React Data Fetching & Animation



Agenda

1 **React Query: Data Management**

Streamlining data fetching and state management.

2 **Framer Motion: Animation Power**

Adding smooth, declarative animations to your React apps.

3 **Comparison & Best Practices**

Choosing the right tools for your project needs.

Installing React Query

Use the following command to install React Query:

```
npm install @tanstack/react-query
```

or

```
yarn add @tanstack/react-query
```

Setup example:

```
;'import { QueryClient, QueryClientProvider } from '@tanstack/react-query

;()const queryClient = new QueryClient
```

React Query Overview

React Query (now TanStack Query) is a powerful library for managing server state in React applications, handling data fetching, caching, synchronization, and updates efficiently.

Data Fetching

Fetches data seamlessly from APIs, integrating it with existing data sources.

State Management

Manages loading and error states automatically, reducing boilerplate code.

Data Caching

Caches fetched data for optimal performance and improved user experience.

Background Data Refetching

Refetches data to keep the UI updated with the latest information.

React Hooks: Understanding useEffect

```
// useEffect( () => { console.log("") }, [] )
```

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

} ()export default function App
;([])const [posts, setPosts] = useState
;const [loading, setLoading] = useState(true)

<= ())useEffect

fetch("https://jsonplaceholder.typicode.com/posts")
then(res => res.json()).
} <= then(data.
;setPosts(data)
;setLoading(false)
({

} <= catch(err.
;console.error("Error fetching data:", err)
;setLoading(false)
;({
;([] ,{

<if (loading) return <p>Loading...</p>

) return
<div>

<h1>Posts</h1>
) <= mposts.slice(0, 5).map(post}
<div key={post.id} style={{ marginBottom: "10px" }}>
<h3>{post.title}</h3>
<p>{post.body}</p>
</div>
{((

</div>
;(
{
```

React Query: Simple Code Examples

1. Setting up QueryClientProvider (e.g., in `main.jsx`)

```
;"import React from "react
;"import ReactDOM from "react-dom/client
;"import App from "./App
;"import { QueryClient, QueryClientProvider } from "@tanstack/react-query

;()const queryClient = new QueryClient

) ReactDOM.createRoot(document.getElementById("root")).render
<React.StrictMode>
<QueryClientProvider client={queryClient}>
<App/>
<QueryClientProvider/>
<React.StrictMode/>
;(
```

2. Fetching Data with useQuery (e.g., in `App.jsx`)

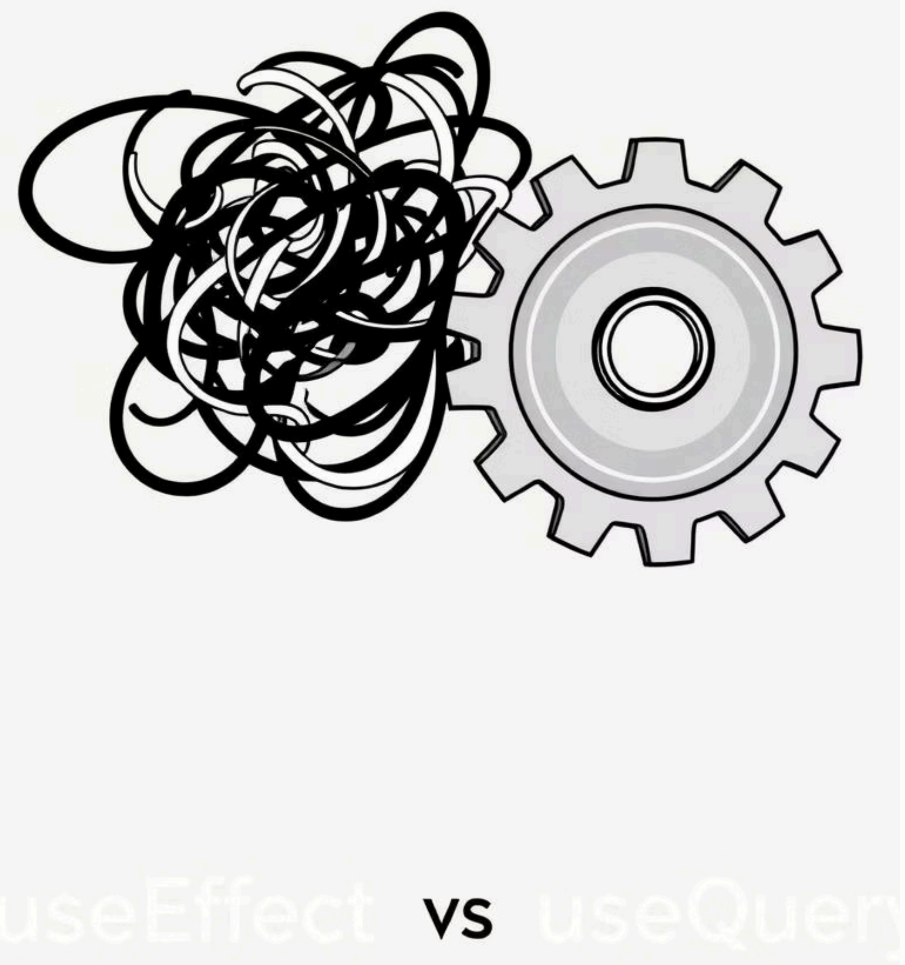
```
;"import React from "react
;"import { useQuery } from "@tanstack/react-query
;"import axios from "axios

} () export default function App

}) const { data, isLoading, error } = useQuery
queryKey: ["posts"]
queryFn: () => axios.get("https://jsonplaceholder.typicode.com/posts")
then(res => res.data).
;{(

<if (isLoading) return <p>Loading...</p
<if (error) return <p>Error loading data: {error.message}</p

)return
<div>
<h1>Posts</h1>
) <= data.slice(0, 5).map(post
<div key={post.id} style={{ marginBottom: "10px" }}>
<h3>{post.title}</h3>
<p>{post.body}</p>
</div>
{((
</div>
;(
{
```



useQuery vs useEffect

useEffect + fetch

- Manual handling of state, loading, and error.
- Full control over logic.

useQuery (React Query)

- Simplifies data fetching with a single hook.
- Includes caching, auto refetching, and error handling out-of-the-box.
- Productivity and clean code.

Comparison Table

Purpose	Manual data fetching	Declarative & automatic
Code Complexity	Manual state & errors	Simplified API
Caching	Not available	Built-in
Auto Refetch	Manual setup	Built-in
Error Handling	Extra logic required	Built-in
Loading States	Manual control	Built-in
Best For	Full control	Fast development

When to Use Which?

Use useEffect when:

- You need low-level control over fetching and side effects.
- You want to combine multiple fetches in complex logic.

Use useQuery when:

- You want automatic caching, refetching, and clean structure.
- You need scalable data-fetching for larger projects.

Installing Framer Motion

Use the following command to install Framer Motion:

```
npm install framer-motion
```

or

```
yarn add framer-motion
```

Example usage in your component:

```
;'import { motion } from 'framer-motion

<motion.div initial={{ opacity: 0 }} animate={{ opacity: 1 }} />
```

Framer Motion Overview

Framer Motion is a powerful animation library for React.

Simple Syntax

Declarative and easy to use.

Smooth Animations

Fluid transitions and effects.

Layout Animations

Shared layout transitions.

Gesture Support

Drag, hover, tap interactions.

Css Vs framer motion for code

1. Code Css Animation

```
                              } keyframes fadeInUp@
                                   } 0%
                               ;opacity: 0
                     ;transform: translateY(-50px)
                                   {

                                  } 100%
                               ;opacity: 1
                       ;transform: translateY(0)
                                    {
                                       {

                                } fade-in.
              ;animation: fadeInUp 1s ease-in-out forwards
                                     {
```

2. code Framer-motion

```
                                 ;"import React from "react
                          ;"import { motion } from "framer-motion

                             }  () export default function App


                                   ) return                          <div style={{textAlign: "center", marginTop: "50px" }}
 <motion.h1 initial={{ opacity: 0, y: -50 }}                               animate={{ opacity: 1, y: 0 }}
 <transition={{ duration: 1 }}


              <motion.h1>
      Hello Framer Motion


              </motion.h1>
                      </div>


                              ;(
                                 {
```

Choosing the Right Web Animation Tool

Here's a detailed comparison of popular web animation tools to help you decide which one best suits your project needs:

Feature / Tool	**CSS Animations**	**GSAP** (GreenSock)	**Framer Motion**
Type	Declarative (in CSS)	Imperative (JavaScript API)	Declarative (React-based)
Ease of Use	Very simple	Advanced, more control	Easy for React developers
React Integration	Not native	Possible, but manual	Built for React
Control	Limited	Full timeline & fine control	Moderate – good control via props
Performance	Hardware-accelerated	Highly optimized	Great performance
Animations	Basic transitions only	Complex timelines, SVG, canvas	Layout, gestures, transitions
Code Style	CSS only	JS-driven (imperative)	JSX + props (declarative)
Interactivity	Limited (hover/focus etc.)	Full gesture & scroll animations	Built-in gestures (drag, tap, etc.)
Learning Curve	Very low	Medium to high	Low for React devs
Community/Support	Built-in browser tech	Large & mature	Growing, supported by Framer

🧠 Summary (What to Use When):

CSS Animations

- Simple transitions (hover, fade, etc.)
- No JavaScript setup needed

GSAP

- Advanced timelines, scroll-based animations
- SVG or canvas animations
- Full control over sequencing

Framer Motion

- React-based projects
- Layout and component transitions
- Easy gesture support (drag, tap, etc.)

✓ thanks