TANSTACK QUERY

TANSTACK QUERIES

QUERY

A query is a declarative dependency on an asynchronous source of data that is tied to a unique key. A query can be used with any Promise based method (including GET and POST methods) to fetch data from a server

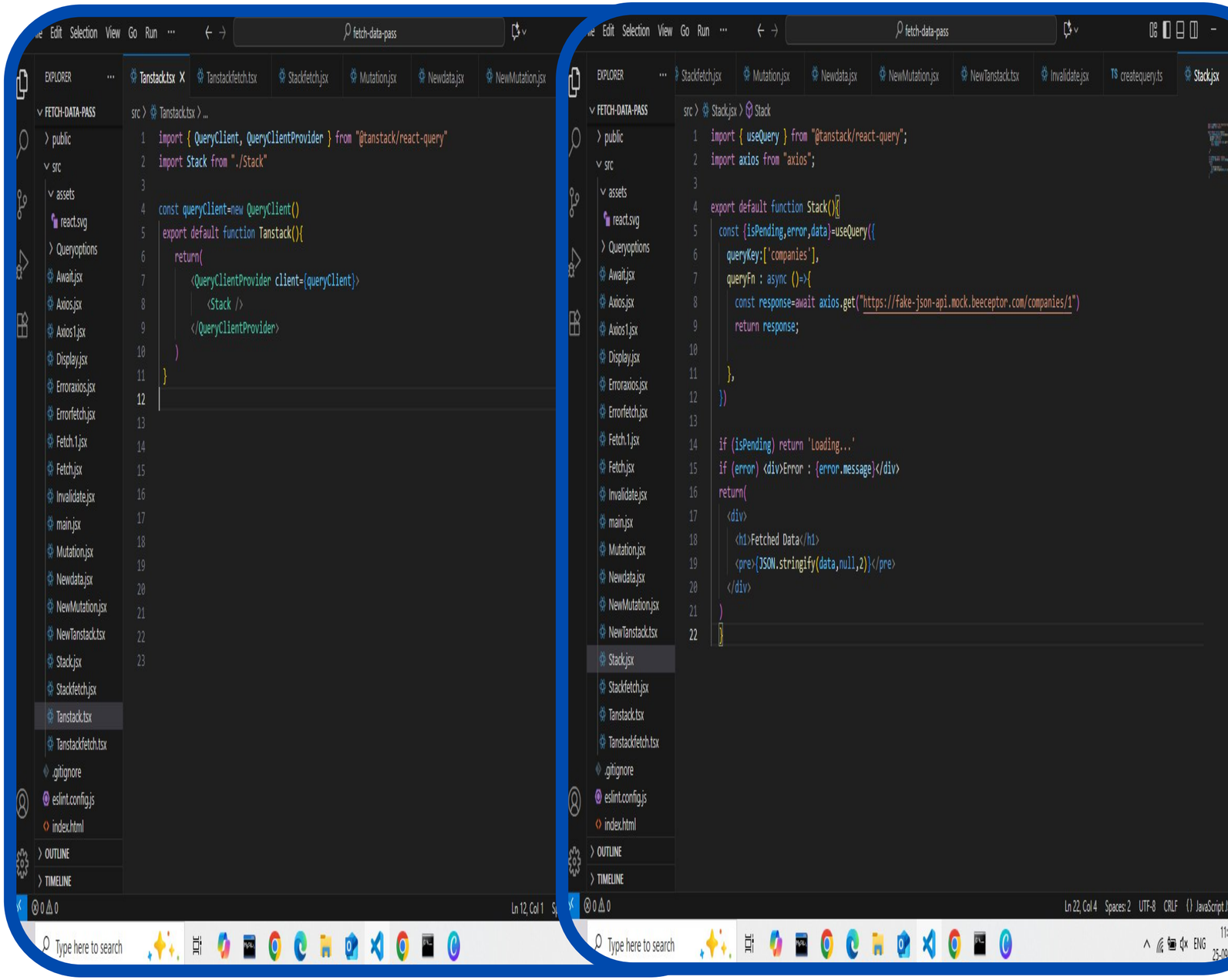
MUTATION

Mutations are typically used to create/update/delete data or perform server side-effects. For this purpose, TanStack Query exports a useMutation hook.

INVALIDATE

In React Query, invalidating a query means marking it as "stale," which tells React Query that the data associated with that query is potentially outdated and should be re-fetched.

EXAMPLE OF TANSTACK QUERY



FETCHING API

FETCH

Fetching data in React applications often involves using the fetch API, a native JavaScript interface for making HTTP requests

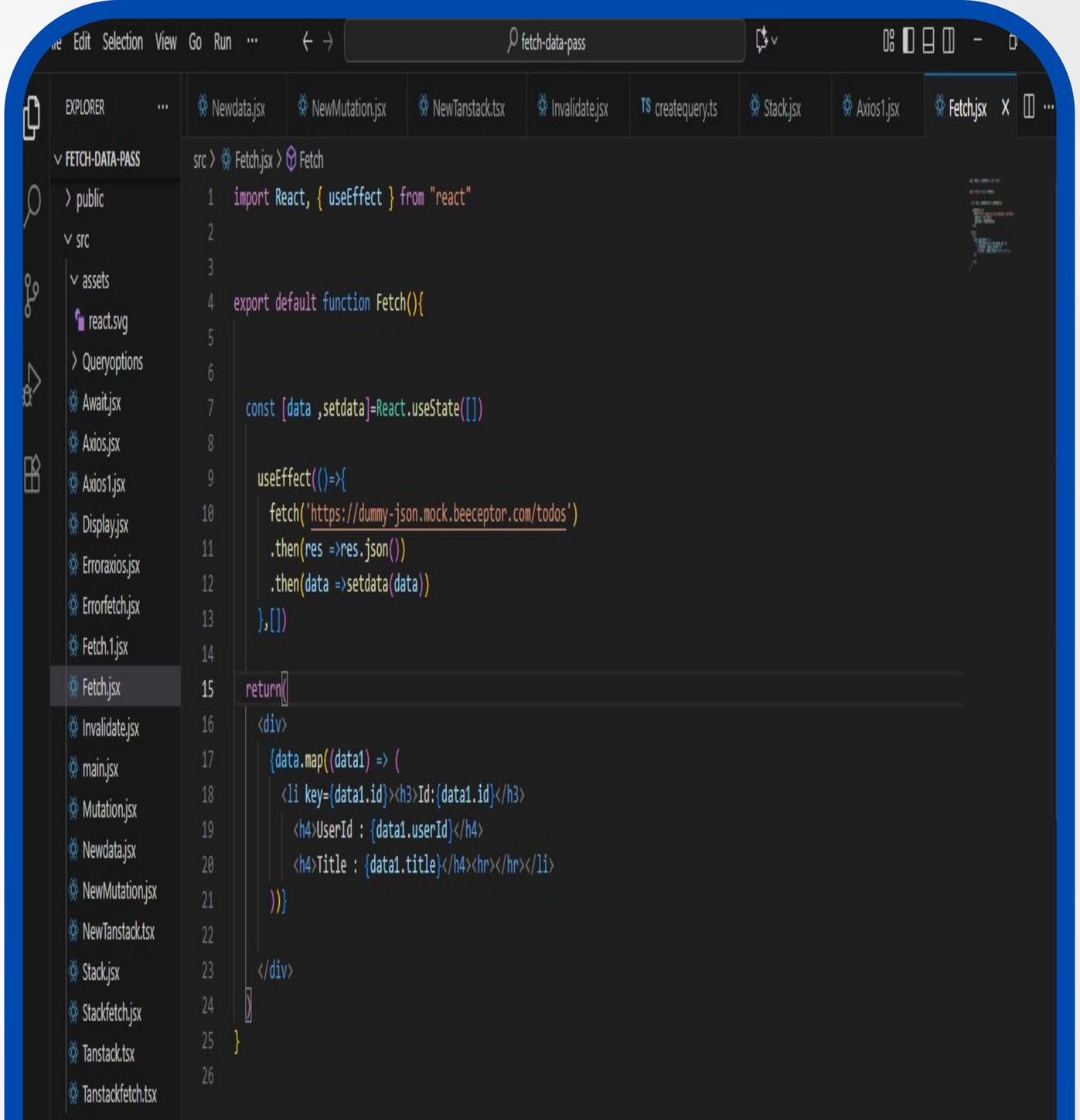
AXIOS

Axios is a popular, promise-based HTTP client used in React applications to make asynchronous requests to APIs or other backend services

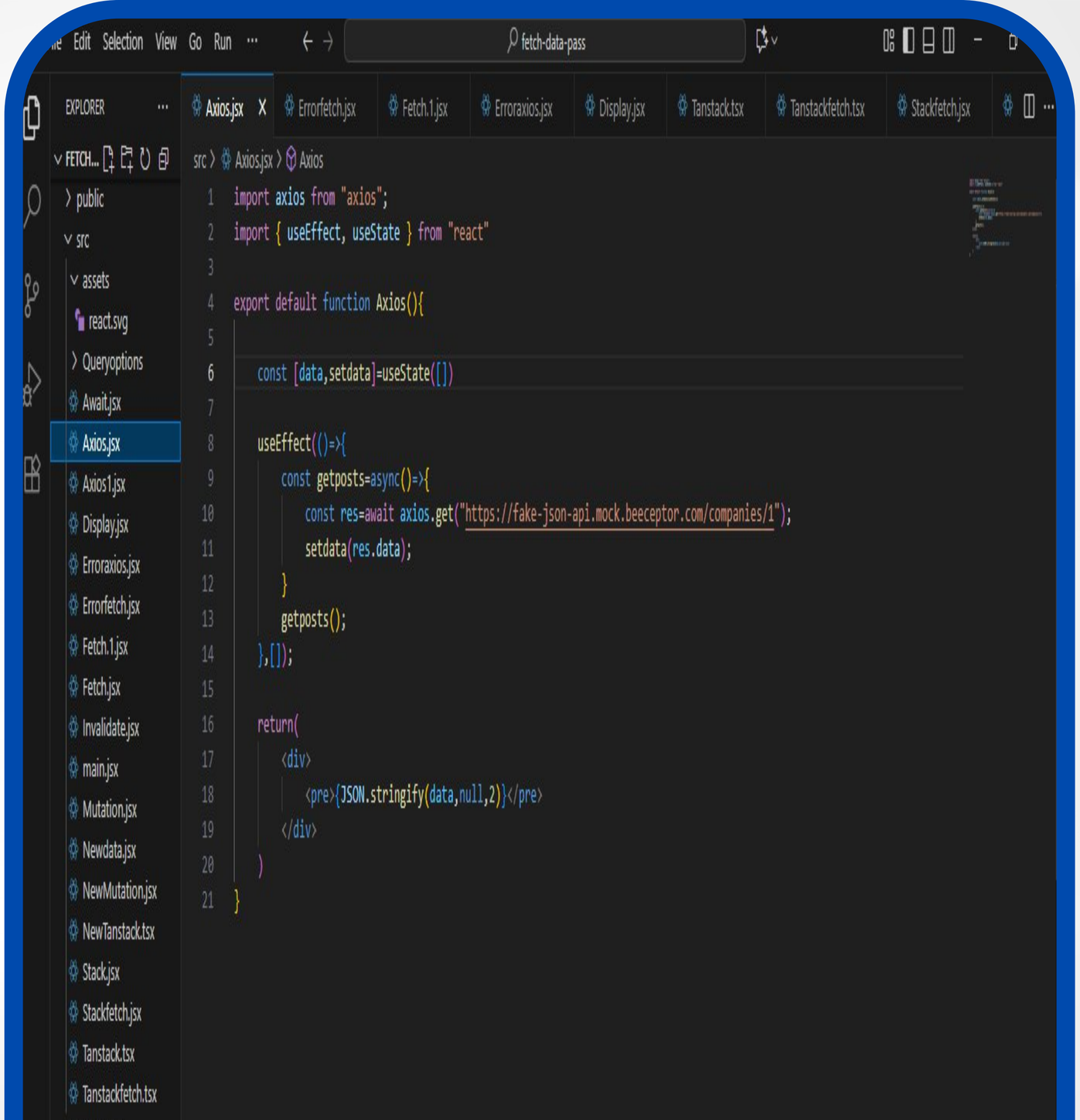
TANSTACK QUERY

TanStack Query, formerly known as React Query, is a powerful data-fetching library for React applications.
The process of fetch ,caching and updating

EXAMPLE OF FETCH DATA



EXAMPLE OF AXIOS



Background sync in React, particularly within Progressive Web Apps (PWAs), enables web applications to defer actions, such as sending data to a server, until the device has a stable network connection. This ensures a more robust and offline-capable user experience.

Background Sync

Handling loading, error, and success states in a React component using useQuery from React Query involves destructuring the relevant properties returned by the hook.

The most common ones for handling loading, error, and success are:

1. **isLoading** or **isPending** : A boolean indicating if the query is currently fetching data for the first time. isPending is the newer, preferred term.
2. **isError** : A boolean indicating if the query encountered an error.
3. **Error** : Contains the error object if isError is true.

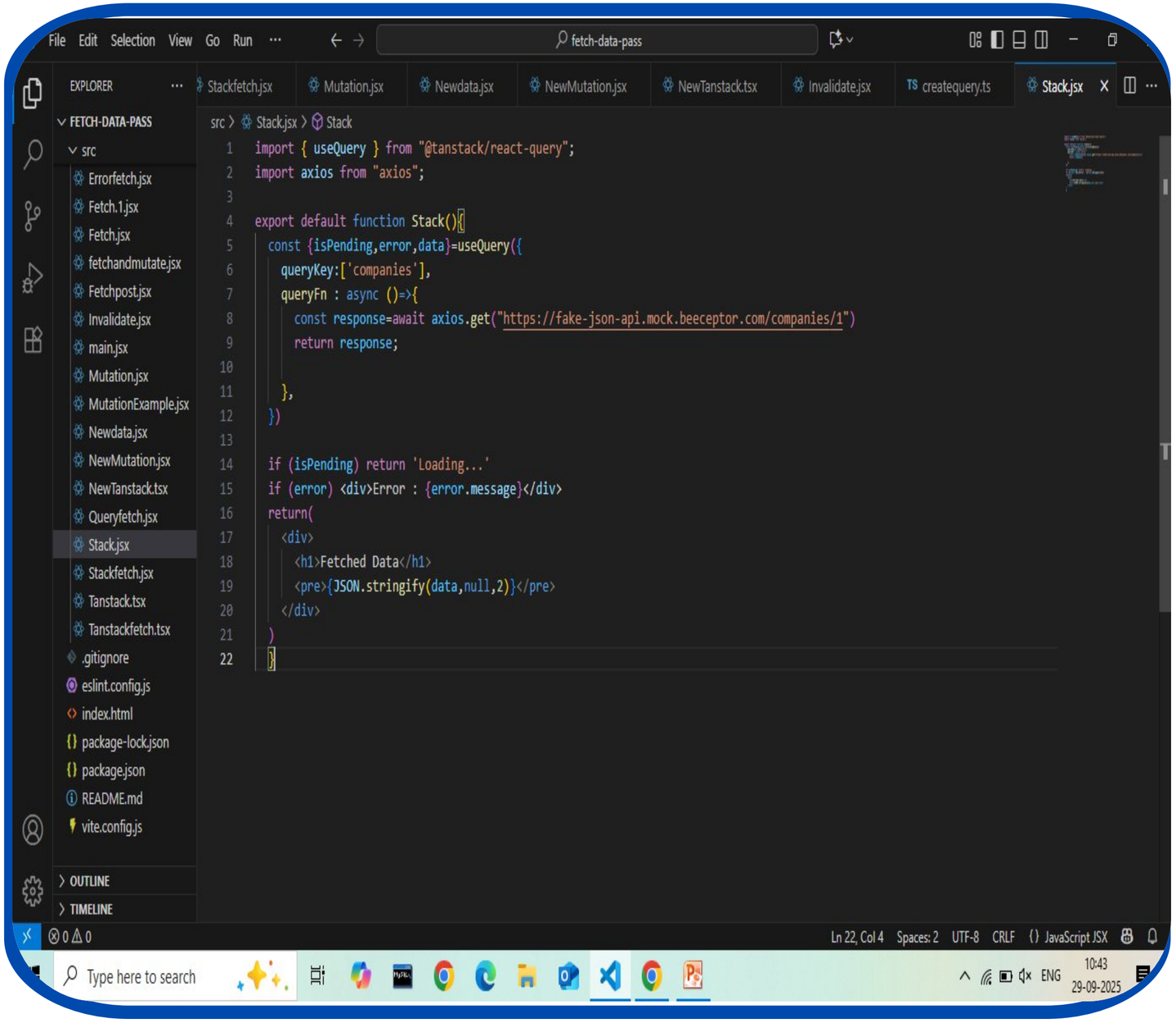
Handling Error, Loading,

Success

EXAMPLE OF

Handling Error, Loading

Success



Stale Time	Cache Time
Definition: Stale time defines how long data is considered "fresh" or valid before it needs to be re-fetched or revalidated from its source.	**Definition:** Cache time (or gcTime) determines how long data remains in the cache after it becomes unused (i.e., no longer actively observed or referenced by any part of the application).
Purpose: It dictates when a cached item should be considered potentially outdated. Once the stale time expires, the data is marked as stale, and subsequent requests for that data will trigger a re-fetch or revalidation attempt to ensure data currency.	**Purpose:** It manages the memory footprint of the cache by specifying when inactive data should be eligible for garbage collection and removal from memory
Impact: A shorter stale time means data is refreshed more frequently, ensuring higher data accuracy but potentially leading to more network requests. A longer stale time reduces network requests but increases the risk of serving outdated information.	**Impact:** A shorter cache time frees up memory more quickly but might require re-fetching data if it's needed again shortly after becoming inactive. A longer cache time keeps data readily available for longer but consumes more memory.
Behavior: Serves cached data, and if stale, triggers an immediate background refetch upon a new query.	**Behavior:** Deletes inactive data completely from the cache after the time limit expires.

THANK YOU