Angular RXJS Functions:

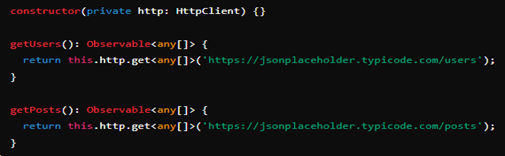
How to Handle 2 subscribes asynchronous operations avoid null or undefined in further subscribes

Using **forkJoin** to wait for both HTTP requests to complete before processing the data:

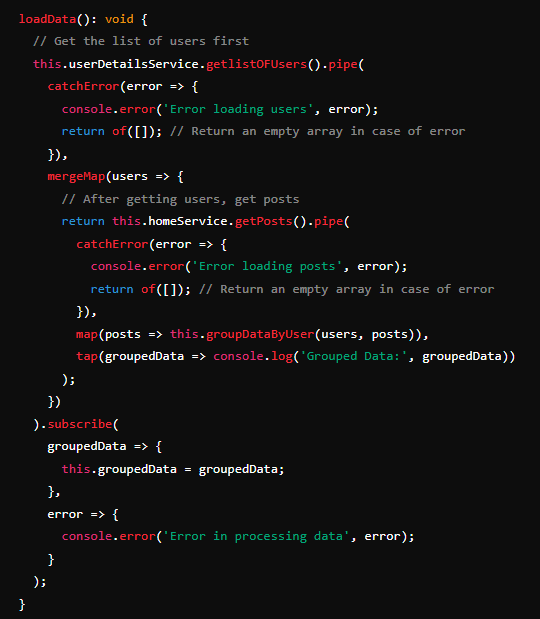
**forkJoin**: This operator is used to wait for all provided observables to complete and then combine their last emitted values into a single observable. In this case, it waits for both getUsers and getPosts to complete.

**catchError**: In case any of the requests fail, catchError is used to handle errors gracefully by returning an empty array.

**map**: Once both requests complete, map is used to process the combined data. The groupDataByUser function is called to group posts by their associated user.



**use mergeMap (also known as flatMap). This operator is useful when you want to run multiple asynchronous tasks in parallel and handle each result as it arrives.**

**** **getlistOFUsers**: Fetches the list of users.

 **catchError**: Handles any errors during the fetching of users.

 **mergeMap**: After successfully fetching the users, it fetches the posts. The mergeMap operator ensures that the fetching of posts starts as soon as the fetching of users is completed, without waiting for all posts to be fetched before proceeding.

 **map**: Once posts are fetched, map is used to group the posts by users.

 **tap**: Logs the grouped data for debugging purposes.

 **subscribe**: Assigns the grouped data to the groupedData property.

**Advantages of Using mergeMap:**

* **Concurrency**: mergeMap allows concurrent execution of multiple inner observables, which means it fetches posts immediately after fetching users.
* **Flexibility**: Suitable for handling independent operations where the completion of one does not necessarily depend on the other.
* **Efficiency**: Avoids nested subscriptions and keeps the code clean and readable

the pipe method is used to compose and chain multiple operators to transform, filter, and manage the values emitted by an observable. It allows you to build complex data processing flows in a readable and declarative manner.

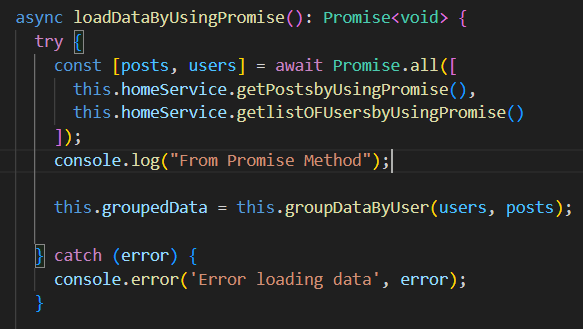
Here's a breakdown of the pipe method and its usage in the provided code:

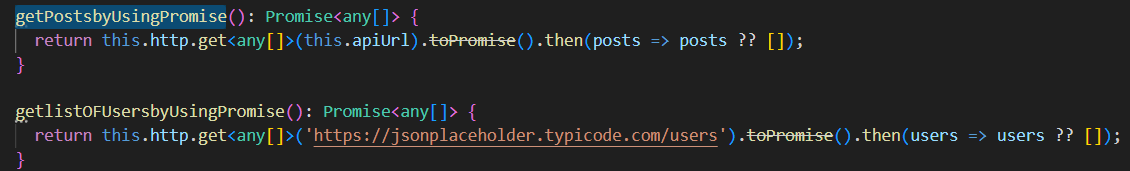
**What is pipe?**

* pipe is a method on observables that allows you to combine multiple operators into a single, reusable pipeline.
* Operators are functions that take an observable as input and return a transformed observable as output.

**Usage of pipe in the Example**

In the loadData method, pipe is used to chain several operators together for the observables returned by getlistOFUsers and getPosts.

**Using Promise**

**** **Fallback Value**: In the service methods, posts ?? [] ensures that if posts is undefined, it will fallback to an empty array.

 **Error Handling**: The try-catch block in loadData handles any errors that might occur during the fetching of data.

*  Promise.all is used to run both HTTP requests concurrently.
* await is used to wait for both requests to complete.
* The results are processed once both Promises resolve, and groupDataByUser is called to group the data.

 **Error Handling**:

* try-catch block is used to handle any errors that might occur during the fetching of data.

