concatMap

Map values to inner observable, subscribe and emit in order.

concatMap(project: function, resultSelector: function): Observable

Note the difference between concatMap and mergeMap. Because concatMap does not subscribe to the next observable until the previous completes.

Use of concatMap operator:

1. Reduce subscribes of multiple observables to single observable
2. When 2nd API has dependency on 1st API's response
3. Want to call APIs order wise

Real time Example 1

Consider you have an application with 3 scenarios

1. Fetch data of User details
2. Based on the location of user, application's content will change to specific language of that location.

   If user is from India, it will show Hindi

   If user is from USA/UK, it will show English

   If user is from Germany, it will show German

3. Based on user's location you need to show Google map of that specific location
4. Based on user's details you need to integrate the theme of the application

Consider you have 3 APIs:

1. getUserDetails()
2. getLanguage(userCountry)
3. getThemeDetails(userTheme)

Without using concatMap

```
this.apiService.get(`http://localhost:4200/user/getUserInfo`).subsribe((user) => {
        if(user) {
                this.setUser(user);
                this.apiService.get(`http://localhost:4200/user/getLanguage/${
user['userCountry']}`)).subscribe((language) => {
                        this.setLanguageTranslator(language);
                });
                this.apiService.get(`http://localhost:4200/user/getThemeDetails/${
                user['userTheme']}`)).subscribe((themeDetails) => {
                        this.changeTheme(themeDetails);
```

```
            });

        }
});
```

Using concatMap

```
this.apiService.get(`http://localhost:4200/user/getUserInfo`).pipe(
        tap(user => this.setUser(user); ),
        concatMap(res => this.apiService.get(`http://localhost:4200/user/getLanguage/${
res['userCountry']}`))),
        tap(language => this.setLanguageTranslator(language); ),
        concatMap(res => this.apiService.get(`http://localhost:4200/user/getThemeDetails/${
res['userTheme']}`))),
        tap(themeDetails => this.changeTheme (themeDetails); ),
) .subscribe();
```

concatMap *will not merge the response, it will print response one by one by order.* ***concatMap*** *never hits the next http request until the previous one completes.*

Example Using Mockdata

userInfo.json

```
[
  {
    "userId": "1",
    "userName": "Piyali",
    "userCountry": "india",
    "userTheme": "black"
  }
]
```

themes.json

```
[
  {
    "themeId": "1",
    "themeName": "black",
    "bgcolor": "black",
    "fontcolor": "white"
  }
```

```
]
```

Language.json

```
[
  {
    "id": "1",
    "language": "English",
    "country": "usa"
  }
]
```

```
callToAPI(): void {
    this.http.get('http://localhost:4200/assets/userInfo.json').pipe(
      tap(user => {
        console.log('1st res => ', user);
        this.setUser(user);
      }),
      concatMap(res => this.http.get('http://localhost:4200/assets/language.json'
)),
      tap(language => {
        console.log('2nd res => ', language);
        this.setLanguageTranslator(language);
      }),
      concatMap(res => this.http.get('http://localhost:4200/assets/theme.json')),
      tap(themeDetails => {
        console.log('3rd res => ', themeDetails);
        this.changeTheme(themeDetails);
      })
    ).subscribe((res) => {
      console.log('Final res => ', res);
    });
  }

  setUser(user): void {
  }

  setLanguageTranslator(language): void {

  }

  changeTheme(themeDetails): void {

  }
```

If 1st API gets fail, then other apis will not be called. You have to handle error using rxjs error handler.