Reactive Programming in Angular 2

Muhammad Kamran

What is Reactive Programming?

Reactive Programming is a Programming Paradigm like Object Oriented Programing and It’s a combination of Observer design pattern and the Iterator design pattern to solve the asynchronous programming with observable streams.

What is Rxjs?

RxJS or Reactive Extensions is implementation of Reactive Programming Paradigm for Javascript.

How it solves the asynchronous issues?

In Javascript when we are programming in asynchronously then we stuck in callback hell like.

```
getData((x) => {
    getMoreData(x, (y) => {
        getMoreData(y, (z) => { ... });
    });
});
```

How it solves the asynchronous issues?

So deal with callback hell the promises was introduced but again we stuck in doom of pyramid like below and we were not able to handle data stream and we can’t cancel the promise.

```
getData().then((x) => {
    getMoreData(x).then((y) => {
        getMoreData(y).then((z) => { ... });
    });
});
```

How it solves the asynchronous issues?

This is how rxjs solves the problem.

- Everything is Steams like Dom Event, Literals, Http Request.
- Observables are cancellable.
- Observables are not executed without subscribtion.
- Observables have operators.

```
const click = Rx.Observable.fromEvent(document, 'click');
clicks.subscribe(() => console.log('double click'));
```

Angular 2 and Rxjs

Rxjs is a dependency of Angular 2 and uses Rxjs 5 which is in beta version.

Below are the sections where Rxjs will be used in angular 2.

- Dom Query
- Http Request
- Data Services
- Application Architecture

Dom Query

```
var searchbox = document.getElementById(‘searchbox’);

Rx.Observable.fromEvent(searchbox, ‘change’)
    .map(event => event.target.value)
    .subscribe(value => console.log(value))
```

Dom Query

```
var searchbox = document.getElementById(‘searchbox’);

Rx.Observable.fromEvent(searchbox, ‘change’)
    .map(event => event.target.value)
    .debounceTime(400)
    .map(keyword => keyword.trim())
    .filter(keyword => keyword.length > 0)
    .distinctUntilChanged()
    .subscribe(value => console.log(value))
```

Http Request

```
import {Http} from "angular2/http";

@Injectable()
export class GithubSearch {
  private baseUrl = "https://api.github.com";

  constructor(private http: Http) {}

  searchUsers(term: string): Observable<string[]> {
    return this.http.get(this.baseUrl + "/search/users?q=" + term)
      .map(response => response.json())
      .map(result => result.items.map(item => item.login))
      .share();
  }
}
```

Data Service (Redux Like)

```
Import { Injectable } from ‘@angular/core’;

@Injectable()
export class DataService {
  todos$ : BehaviorSubject<Array<string>> = new BehaviorSubject([])
  constructor() {}

  newTodo(todo: string) {
    let todos = this.todos$.getValue();
    todos.push(todo);
    this.todos$.next(todos);
  }


}
```

Application Architecture

ngrx

ngRedux

STACK INNOVATIONS

Muhmmad Kamran
Founder & CEO at Stack Innovations

mkamran@stackinnovations.io
facebook.com/devkamran

+923008203588

Thanks You