**Discussion 4.1 – Observables and Observable Streams**

Patrick Cuauro

Bellevue University

WEB 425-307O Angular with TypeScript

Professor Krasso

August 29, 2023

# Observables and Observable Streams

Observables are software design patterns in which an object called the subject maintains a list of its dependents, called “observers” and notifies them automatically of state changes. This pattern is similar to the “publish/subscribe” design pattern. They provide support for passing messages between parts of your application. They are used frequently in Angular and are a technique for event handling, asynchronous programming, and handling multiple values.

Observables are declarative -that is, you define a function for publishing values, but it is not executed until a consumer subscribes to it. The subscribed consumer then receives notifications until the function completes, until they unsubscribe.

Observables are lazy Push collections of multiple values. They fill the missing spot in the following table:

|  |  |  |
| --- | --- | --- |
|  | Single | Multiple |
| Pull | Function | Iterator |
| Push | Promise | Observable |

Example. The following is an Observable that pushes the values 1, 2, 3 immediately (synchronously) when subscribed, and the value 4 after one second has passed since the subscribed call, then completes:

|  |
| --- |
| Import { Observable } from 'rxjs';  const observable = new Observable((subscriber) => {  subscriber.next(1);  subscriber.next(2);  subscriber.next(3);  setTimeout(() => {  subscriber.next(4);  subscriber.complete();  }, 1000);  }); |

To invoke the Observable and see these values, we need to *subscribe* to it:

|  |
| --- |
| Import { Observable } from 'rxjs';  const observable = new Observable((subscriber) => {  subscriber.next(1);  subscriber.next(2);  subscriber.next(3);  setTimeout(() => {  subscriber.next(4);  subscriber.complete();  }, 1000);  });    console.log('just before subscribe');  observable.subscribe({  next(x) {  console.log('got value ' + x);  },  error(err) {  console.error('something wrong occurred: ' + err);  },  complete() {  console.log('done');  },  });  console.log('just after subscribe'); |

Which executes as such on the console:

|  |
| --- |
| just before subscribe  got value 1  got value 2  got value 3  just after subscribe  got value 4  done |

## Stream

For those who can get at one glance, it is a sequence of values over time. That means instead of variables we are using streams in reactive programming. The value of the stream may change as time passed. Its value is not unique.

Sources:

* Angular official page

<https://angular.io/guide/observables>

* RxJS dev guide Observable

<https://rxjs.dev/guide/observable>

* Streams vs Observables in Angular RXJS

<https://medium.com/@pramodyasachin/streams-vs-observables-in-angular-rxjs-c54f523236e3>