Create observable from a string, array & object in angular

10 Comments / 5 minutes of reading / August 4, 2023

← Angular Observable Tutorial

Angular Tutorial

Observable from Event →

In this tutorial, we will show you how to create observable using `create`, `of`, `from` operators in Angular. We can use them to create new observable from the array, string, object, collection or any static data. Also learn the difference between the `Of` & `From` operators. If you are new to observable, we recommend you to read the Angular observable before continuing here.

Table of Contents

Observable creation functions

There are many ways to create observable in Angular. You can make use of Observable Constructor as shown in the observable tutorial. There are a number of functions that are available which you can use to create new observables. These operators help us to create observable from an array, string, promise, any iterable, etc. Here are some of the operators

- create
- defer
- empty
- from
- fromEvent
- interval
- of
- range
- throw
- timer

All the creation related operators are part of the RxJs core library. You can import it from the 'rxjs' library

Create

The Create method is one of the easiest. The create method calls the observable constructor behind the scene. Create is a method of the observable object, Hence you do not have to import it.

```

ngOnInit() {

  //Observable from Create Method
  const obsUsingCreate = Observable.create( observer => {
    observer.next( '1' )
    observer.next( '2' )
    observer.next( '3' )

    observer.complete()
  })

   obsUsingCreate
     .subscribe(val => console.log(val),
           error=> console.log("error"),
           () => console.log("complete"))
}



****Output *****
1
2
3
Complete

```

Observable Constructor

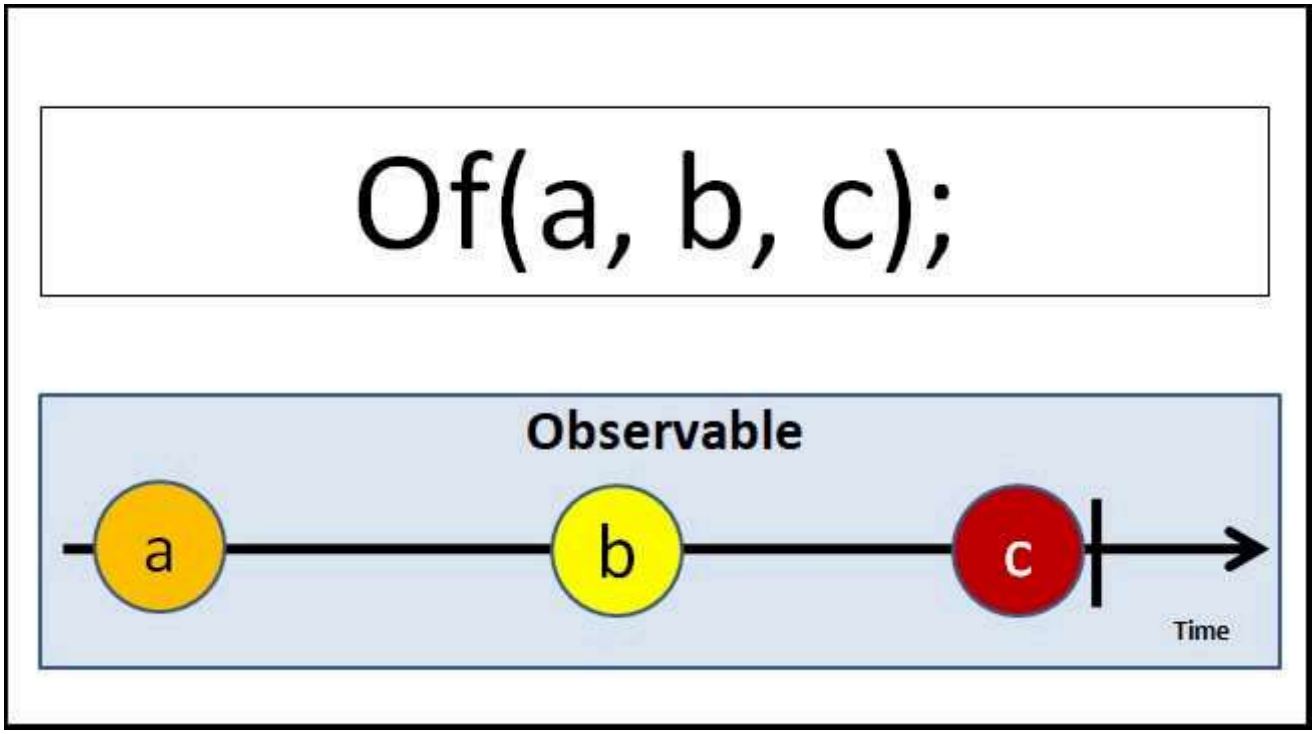
We looked at this in the previous tutorial. There is no difference between the Observable.create method and observable constructor. The Create method calls the constructor behind the scene.

```

ngOnInit() {
  //Observable Using Constructor
  const obsUsingConstructor = new Observable( observer => {
```

```
    observer.next( '1' )
    observer.next( '2' )
    observer.next( '3' )

    observer.complete()
  })

  obsUsingConstructor
      .subscribe(val => console.log(val),
            error=> console.log("error"),
            () => console.log("complete"))
}


****Output *****
1
2
3
complete

```

Of Operator

The `Of` creates the observable from the arguments that you pass into it. You can pass any number of arguments to the `Of`. Each argument emitted separately and one after the other. It sends the `Complete` signal in the end.



To use `of` you need to import it from `rxjs` library as shown below.

```
1
2 import { of } from 'rxjs';
3
```

observable from an array

Example of sending an array. Note that the entire array is emitted at once.

```
1
2 ngOnInit() {
3   const array=[1,2,3,4,5,6,7]
4   const obsof1=of(array);
5   obsof1.subscribe(val => console.log(val),
6           error=> console.log("error"),
7          () => console.log("complete"))
8
9 }
10
11
12 **** Output ***
13 [1, 2, 3, 4, 5, 6, 7]
14 complete
15
```

You can pass more than one array

```

ngOnInit() {
  const array1=[1,2,3,4,5,6,7]
  const array2=['a','b','c','d','e','f','g']
  const obsof2=of(array1,array2 );
  obsof2.subscribe(val => console.log(val),
          error=> console.log("error"),
         () => console.log("complete"))

}


**** Output ***
[1, 2, 3, 4, 5, 6, 7]
['a','b','c','d','e','f','g']
complete

```

observable from a sequence of numbers

In the following example, we pass 1,2 & 3 as the argument to the from. Each emitted separately.

```

ngOnInit() {
    const obsof3 = of(1, 2, 3);
    obsof3.subscribe(val => console.log(val),
      error => console.log("error"),
      () => console.log("complete"))

}



**** Output ***
1
2
3
complete

```

observable from string

We pass two strings to the `of` method. Each argument is emitted as it is.

```
 ngOnInit() {
    const obsof4 = of('Hello', 'World');
    obsof4.subscribe(val => console.log(val),
      error => console.log("error"),
      () => console.log("complete"))
}


**** Output ***
Hello
World
complete
```

observable from a value, array & string

We can pass anything to the `Of` operator. It justs emits it back one after the other.

```
 ngOnInit() {
    const obsof5 = of(100, [1, 2, 3, 4, 5, 6, 7],"Hello World");
    obsof5.subscribe(val => console.log(val),
      error => console.log("error"),
      () => console.log("complete"))
}

**** Output ***
100
[1, 2, 3, 4, 5, 6, 7]
Hello World
complete
```

From Operator

From Operator takes only one argument that can be iterated and converts it into an observable.

You can use it to convert

- an Array,
- anything that behaves like an array
- Promise
- any iterable object
- collections
- any observable like object

It converts almost anything that can be iterated to an Observable.

To use `from` you need to import it from `rxjs` library as shown below.

```
1
2 import { from } from 'rxjs';
3
```

observable from an array

The following example converts an array into an observable. Note that each element of the array is iterated and emitted separately.

```
ngOnInit() {

   const array3 = [1, 2, 3, 4, 5, 6, 7]
   const obsfrom1 = from(array3);
   obsfrom1.subscribe(val => console.log(val),
     error => console.log("error"),
     () => console.log("complete"))

}

*** Output ****
1
2
3
4
5
6
7
complete
```

Observable from string

The `from` operator iterates over each character of the string and then emits it. The example is as shown below.

```
ngOnInit() {
  const obsfrom2 = from('Hello World');
   obsfrom2.subscribe(val => console.log(val),
     error => console.log("error"),
     () => console.log("complete"))
}


*** Output ****
```

```
H
e
l
l
o

W
o
r
l
d
complete

```

Observable from collection

Anything that can be iterated can be converted to observable. Here is an example using a collection.

```

ngOnInit() {
    let myMap = new Map()
    myMap.set(0, 'Hello')
    myMap.set(1, 'World')
    const obsFrom3 = from(myMap);
    obsFrom3.subscribe(val => console.log(val),
      error => console.log("error"),
      () => console.log("complete"))
)

*** output ***
[0, "Hello"]
[1, "World"]
complete

```

Observable from iterable

Any Iterable types like Generator functions can be converted into an observable using *from* the operator.

```

ngOnInit() {
    const obsFrom4 = from(this.generateNos())
     obsFrom4.subscribe(val => console.log(val),
     error => console.log("error"),
     () => console.log("complete"))
}

*generateNos() {
   var i = 0;
   while (i < 5) {
     i = i + 1;
     yield i;
  }


*** Output ***
1
2
3
4
5

```

Observable from promise

Use it to convert a Promise to an observable

```

ngOnInit() {
    const promiseSource = from(new Promise(resolve => resolve('Hello World!')));
    const obsFrom5 = from(promiseSource);
    obsFrom5.subscribe(val => console.log(val),
      error => console.log("error"),
      () => console.log("complete"))
}

*** Output ****
Hello World
complete

```

Of Vs From