Create Observable from Event using FromEvent in Angular

Leave a Comment / 3 minutes of reading / March 9, 2023

← Create observable from array, string, number, object & static data

Angular Tutorial

Angular Observable pipe →

Angular provides `FromEvent` method to create an observable from DOM events directly. In this article let us learn how to do that by creating an observable from the button click event, keyup even & scroll events.

Table of Contents

Syntax
Example of fromEvent
How it works
fromevent from button click
fromevent from scroll
fromevent from keyup
Reference

Syntax

```
1
2 fromEvent<T>(target: FromEventTarget<T>,
3           eventName: string,
```

```
        options: EventListenerOptions,
        resultSelector: (...args: any[]) => T): Observable<T>

```

FromEventTarget is the first argument to fromevent. It can be a DOM EventTarget, Node.js EventEmitter, JQuery-like event target, NodeList or HTMLCollection. The target must have a method to register/unregister the event handler. (addEventListener / removeEventListener in case of DOM Event target)

eventName is the second argument, which is a type of event we want to listen to.

Options are the additional argument that we want to pass to , when registering the event handler i.e addEventListener

resultSelector is optional and will be deprecated in future versions.

Example of fromEvent

To create an observable from any event, first, we need to get the reference to DOM element using the viewchild & ElementRef. For example the following code gets the reference to the button element with the id #btn

```

//Template
<button #btn>Button</button>

```

```

//Component

@ViewChild('btn', { static: true }) button: ElementRef;

```

The code `this.button.nativeElement` returns the native DOM element. We pass this as the first argument to the `fromEvent` to create an observable to the `click` event.

```

buttonClick() {
  this.buttonSubscription =  fromEvent(this.button.nativeElement, 'click')
      .subscribe(res => console.log(res));
}

```

We can invoke the above method from the `ngAfterViewInit` method. Note that the `@ViewChild` will not initialize the `btn` element until the `ngOnInit` Hence we are using the `ngAfterViewInit` here.

```

ngAfterViewInit() {
  this.buttonClick();
}


```

How it works

When we subscribe to an observable, which we created using the `fromEvent` method, it registers the event handler using the `addEventListener` in the DOM element. Whenever the user clicks on the button, `fromevent` captures the value and emits it to the subscriber as the first argument. When we unsubscribe, it unregisters the event handler using the `removeEventListener`.

`fromevent` from button click

The following is the complete code of `fromevent` from a button click.

```

import { Component, Input, ViewChild, ElementRef, AfterViewInit, OnInit, OnDestroy } fron
```

```
import { Observable, of, from, fromEvent } from 'rxjs';
import { debounceTime } from 'rxjs/operators';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements AfterViewInit , OnInit, OnDestroy {

  title = 'Angular fromEvent Example';

  @ViewChild('btn', { static: true }) button: ElementRef;

  buttonSubscription

  constructor(private elm: ElementRef) {
  }

  ngOnInit() {
  }


  ngAfterViewInit() {
    this.buttonClick();
  }


  buttonClick() {
    this.buttonSubscription =  fromEvent(this.button.nativeElement, 'click')
        .pipe(debounceTime(300))
        .subscribe(res => console.log(res));
  }


  ngOnDestroy() {
    this.buttonSubscription.unsubscribe()
  }

}

```

fromevent from scroll

The following code shows how to create observable from the window scroll event

```
scroll() {
  const source = fromEvent(window, 'scroll');
  source.subscribe(val => console.log(val));
}
```

fromevent from keyup

The following code shows how to create observable from a keyUp event.

```
//Component

@ViewChild('name', { static: true }) name: ElementRef;

ngAfterViewInit() {
   fromEvent(this.name.nativeElement, 'keyup')
     .subscribe(res => console.log(res));
 }
```

Name

Reference

1. RxJs FromEvent API

Read More

1. Angular Observable
2. Create Observable from string, array etc

← Create observable from array, string, number, object & static data

Angular Tutorial

Angular Observable pipe →