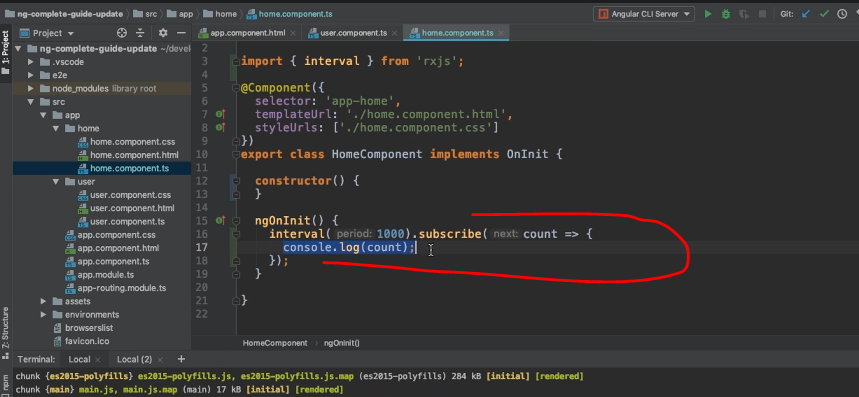
**172. Getting Closer to the Core of Observables:**

* We are going to build our own observables here. But before that lets about built in observable – interval
* We will go to home component, and there in ngOnInit , I want to build a new observable.
* Observable feature is not present in JS or TS. Instead they are added by a package called RxJs

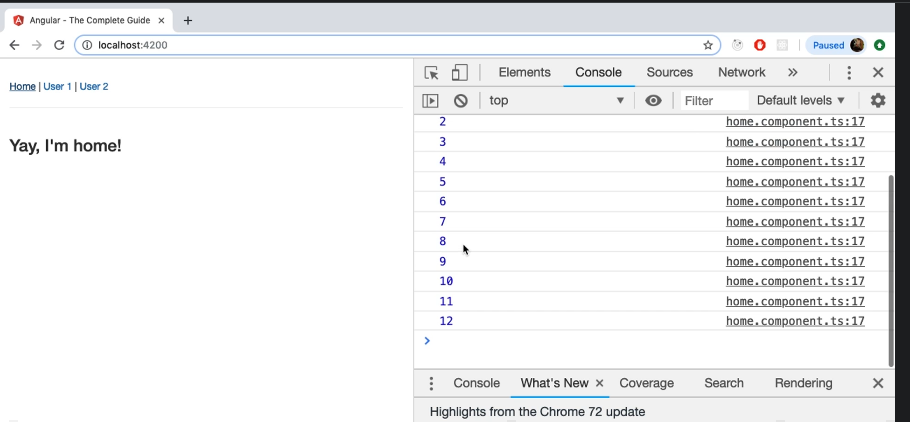
Text

Description automatically generated

* To create a new Observable we need certain features from RxJs package. RxJs gives us different ways of creating a new observable.
* **Interval Method:**
* One of the easy ways: Interval function
* We can call interval and pass a number , which Is interval number in milliseconds
* This interval will fire an event every X milliseconds . So if we pass 1000 to this function – for every second a new event will be emitted.
* Thus this gives an observable and we can subscribe to the observable here.
* There in the method I can get my count (can name whatever we want) and use it inside our function. Thus we are passing an anonymous function here to subscribe.
* This function is the handler for all the data values that are emitted.
* Inside subscribe function , it gets the value that is emitted, where we have named as count
* As part of this subscription – a new value will be fired every second and in this function – we will simply console log count.

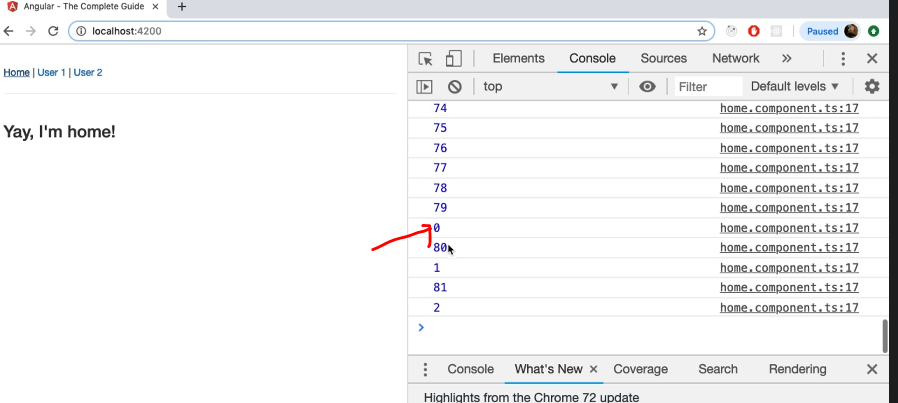


* So we will get a new value emitted every second and logged to the console.
* We can see the same in browser

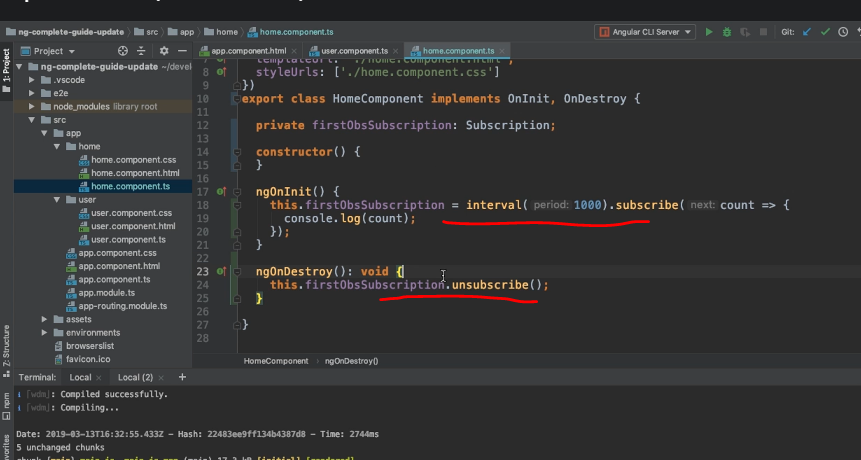


**Something to be aware of:**

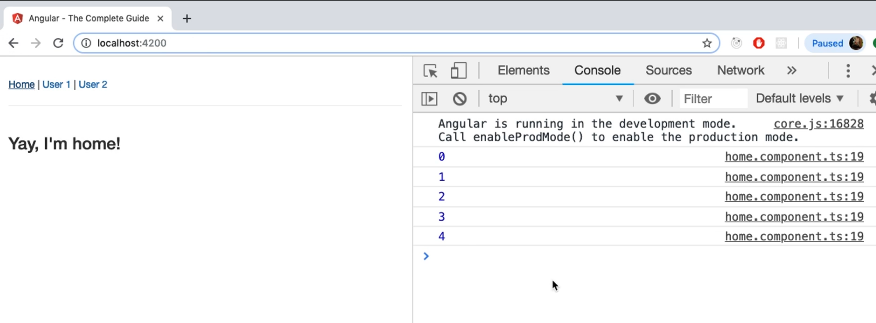
* If we navigate to different tab, that is user1, we can see that it still keeps on counting
* *Observables don’t stop emitting values, just because we are not interested in that anymore.*
* *There are certain observables, that emit values and then they are done . Eg HTTPRequest/HTTPResponse*
* But there are other observables that keep on emitting values
* To stop It and to prevent memory leaks,  ***we should unsubscribe from any observable in which values you are no longer interested.***
* *Another thing is that if we go to home and comeback to User1, a new observable gets created.*



* *Old goes keeps going on and the new one gets started.*
* On repeating the same, we can see more and more observables gets started and that is really bad.
* So behind the scenes, we quickly run out of resources and it slows down our app, which ultimately leads to memory leak.
* **Solution:**
* To unsubscribe, we should store our subscriptions.
* So we initialize firstObsSubscription which is of type :Subscription and we store our subscription in this variable.
* Idea behind storing the subscription simply is : Now we can implement onDestroy here
* Inside ngOnDestroy(), we can use our subscription and call unsubscribe
* That means that whenever we leave that component, **we clear that subscription**
* Thus we prevent memory leaks



* Now if we save and reload the app, and move between components , we don’t get new values on navigating away



**Why we don’t unsubscribe Angular Observables:**

* Because all angular Observables are managed by Angular. Hence we don’t need to unsubscribe manually