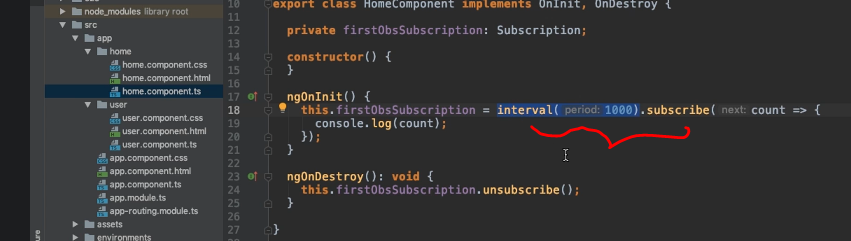
**173. Building a Custom Observable**

* Speaker 1: So I promised you a custom observable to understand its internals and what I gave you was interval .



* Now it's closer to a custom observable because it's not coming from some Angular feature but still this of course is kind of a utility function that gives us a predefined observable that fires a number every second .
* Well actually we can rebuild this manually .

1. **Custom Observable:**

* Let's comment out this code and let's build a real custom observable now .

1. I'll store that in an observable, in a constant and I'll name it customObservable or customIntervalObservable .

* You can of course pick any name you want .

Graphical user interface, text, application

Description automatically generated

b) And to create a new observable we can simply import Observable, the type itself, from rxjs .

* Now you can use observable and call a create method on that .
* And this will do what it sounds like, it creates a new observable .

c) Now, create actually takes a function and I'll pass in an anonymous arrow function here, like this, which will get an argument automatically .

* Rxjs will pass in that argument for us .
* And that argument is a so-called observer .
* Now what's an observer? Now you heard about the observer in the first video of this module .
* *The observer in the end is the part that is interested in being informed about new data, about errors, or about the observable being completed .*
* Now our job here is to tell the observer about new data, about an error, or about the observable being completed .

1. Here, we're not responsible for listening because the observer is the listener .

* *Here, we get that listening part as an argument and we need to tell it once we're done, once new data is there, and so on .*
* So in this *anonymous function* we can now use the *regular set interval method* and, uh, set an interval of one second, let's say, as before .
* Now this again receives an anonymous function .

e) It's the normal set interval method, nothing special about that .

* An insight of said interval here .
* We can now call console log, or do whatever we want, or simply use our observer .

f) And there we now got methods like next .

* *We can call next here to emit a new value,* and that is important .

1. The ***observer has a couple of important methods .***

* **Next** is one of them, **error** is another of them, so that would be the one you use to throw an error, and **complete** is one to let the observer know that you're done .
* Here, however we use next, and to also have an incrementing number here we can simply introduce a variable count, which starts at zero .

Text

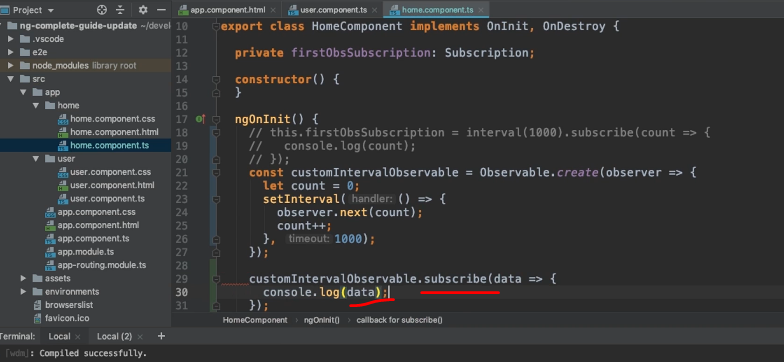
Description automatically generated

h) Inside of our setInterval, we then simply pass this on to next and we then increment count by one .

* So now here our count starts at zero, we incremented by one whenever it setInterval fires, and we let our observer know about that new data here .

i) Now we can also subscribe to our custom observable, and I'll do this in the same place here, in ngOnInit .

* Of course, you could do this in other places too if this were stored in a service or in a property of that component simply .
* So here, I can now call subscribe, and to subscribe you pass the same thing as before a function that simply accepts the data we're emitting .
* So in this case we could name it "count again" or, to avoid naming confusion, simply name it "data", whatever you want .
* And here I will now console-log that data, just as before .



* Now, let's save that all and let's wait for this page to reload .

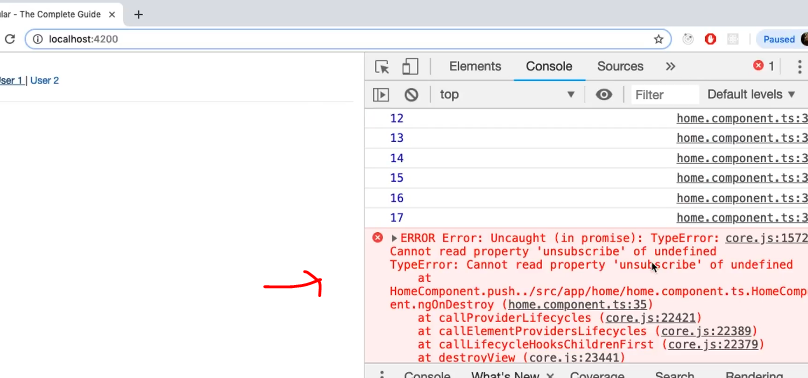
Graphical user interface, application

Description automatically generated

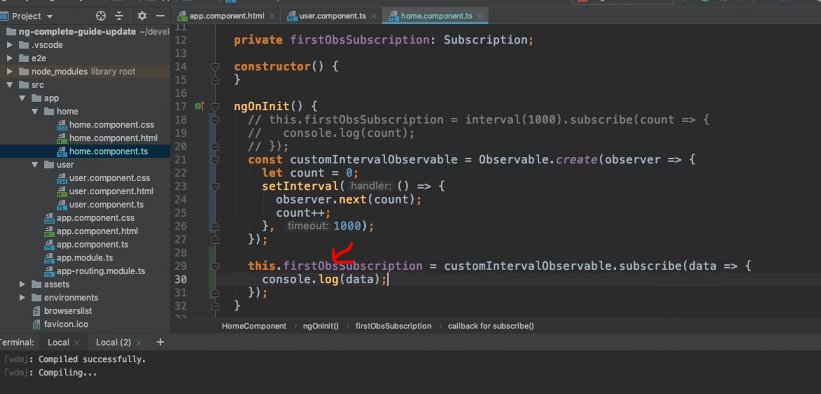
* ***And again, you see the same result as before, but now we built our own interval observable .***

**What happens under the hood:**

* And this is roughly what's happening under the hood here for the built-in interval observable too .
* A)It's giving us an observable, which will essentially fire um, for every X-milliseconds we defined, and B) it lets our observer know about that new data .
* That's what's happening here too .
* And if I navigate away, well then we get a problem here because I try to call unsubscribe of undefined because I don't store my subscription .



* Of course, that should be done here .
* So in my first Obs subscription, I'll now store that subscription again simply by storing the result of subscribe and the subscription again .



* So now with that, that reloads and I navigate away that works without issues .
* Again, if I go back, a new observable starts .
* So this is working and this is a real custom observable and this shows us how to emit new data .
* But what about errors and completing the observable? I mentioned this as well .