**Section 13: Understanding Observables**

**Section 13: Lecture 158//Module Introduction**

1. In the last module about routing we had first contact with observables. Now, we will dive deeper into observables in this section. We have a separate course on observables so we will not dive that deep into them here.
2. Here we will understand what observables are and where we can use them for, and why angular uses them.
3. What is an observable?

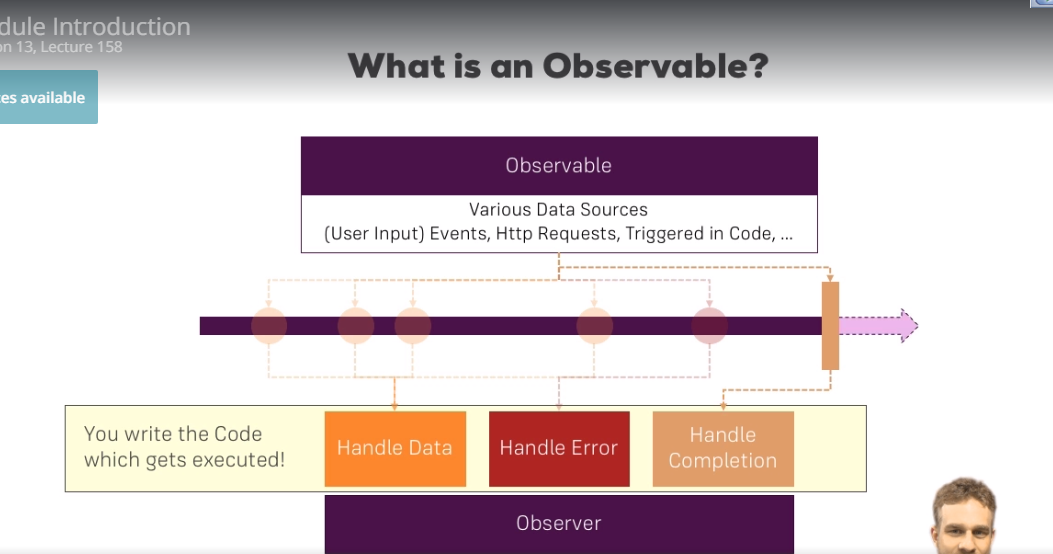
**An observable can be thought of as a data source**, now, in our angular project an observable basically is an object we import from a 3rd party package i.e. RXJS

1. The observable is implemented here such that it follows the observable pattern. So, we have an observable and we have an observer - in between we have a steam i.e. a timeline and on this timeline we have multiple events emitted by the observable or data packages you could say - emitted by observable depending on the data source of that observable.
2. So, an observable could emit data because you triggered it to do so – we can also do that programmatically i.e. it can be linked to a button - therefore when the button is clicked an event or a data package is emitted automatically or as the angular Http service does it - its connected to Http requests. So, when the response returns - the response is emitted as a data package.
3. There are dozens of other data sources too, we will have a look – where to find more soon.
4. So, as we observed - the other part is observer 🡪 this actually is your code, you could say. Its, the subscribe function we saw earlier or at least it has something to do with that.
5. Using observer, we have three ways of handling data packages

Handle Data

Handle error

Handle Completion

1. As shown above – by using observer we can handle normal data, we can handle the error or we can handle the completion of the observable. Because these are the 3 types of the data packages we can receive.
2. In these boxes our code gets executed, so, we can determine what shall we if we receive a new data package, what should happen if we receive an error , what should happen when the observable eventually completes - Note: An Observable doesn’t have to complete.
3. There are observables, for example, hooked up to normal button which neve completes – how would you know, when it completes?
4. Other observables such as http have a clear end and will complete eventually. Because once the response is there, what else should happen? - it’s done
5. This is how the observable pattern generally works and of course we use it to handle asynchronous tasks - because all these data sources here, user event triggered in your code or a HTTP request are asynchronous tasks - you don’t know when they will happen and you don’t know how long they will take. So if you execute your normal application code - you don’t want to wait for such an event and you don’t want to wait for such a HTTP requests, because that would block your program/block your logic.
6. Therefore we need methods for handling such an asynchronous tasks, historically you might have used callbacks and promises - it’s not necessarily bad to use them. Observable is just a different approach of handling that different alternative and angular embraces observables which is why I chose to explain these as angular uses them a lot.
7. Observables have one major advantage i.e. their operators which I will show later in this section too.
8. Back to this slide, we have got our observable and our observer with our free buckets or we can say with our free hooks where our different code is executed depending on the type of package we receive.
9. And then an observable may emit a couple of normal data packages, It might emit an error or it might get completed and the respective code is then executed.
10. 

**Section 13: Lecture 159// Analysing a Built-in Angular Observable**