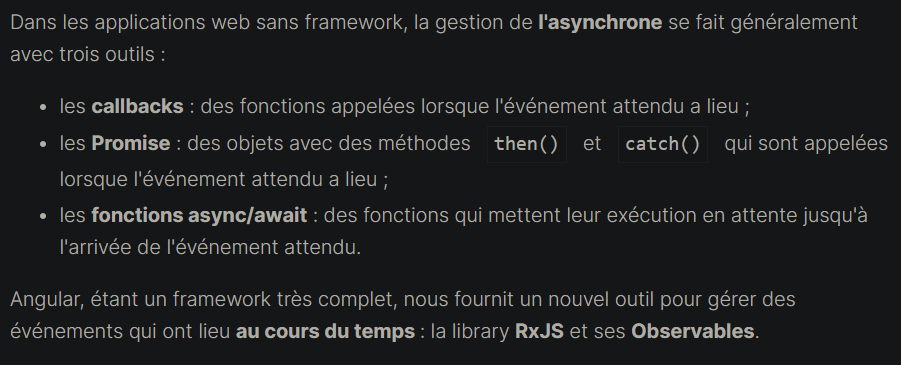
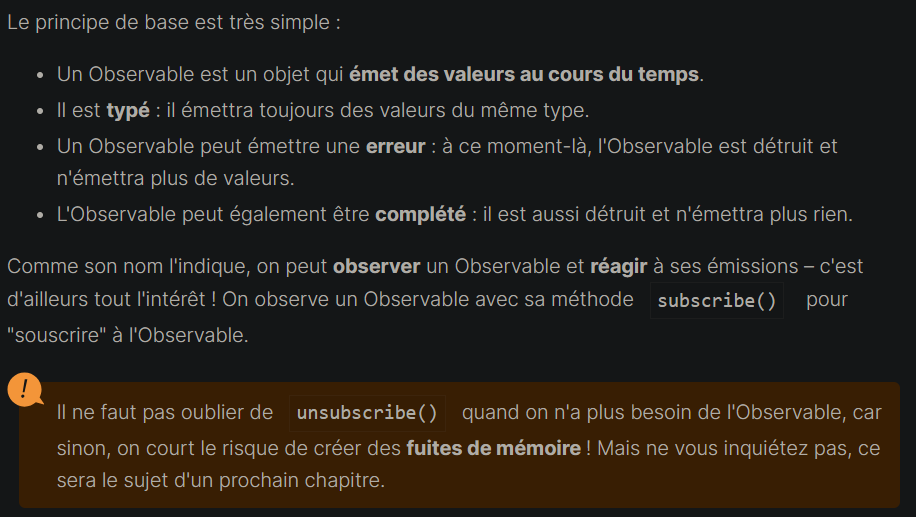
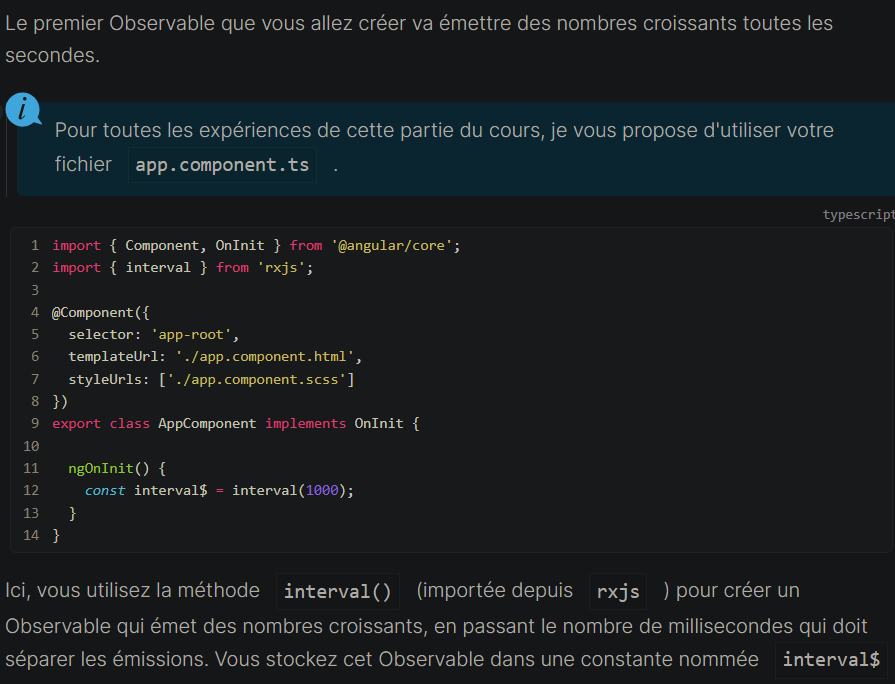
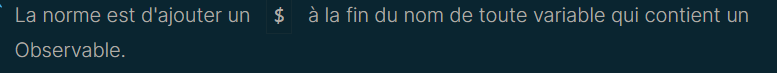
**Les observables RxJS :**

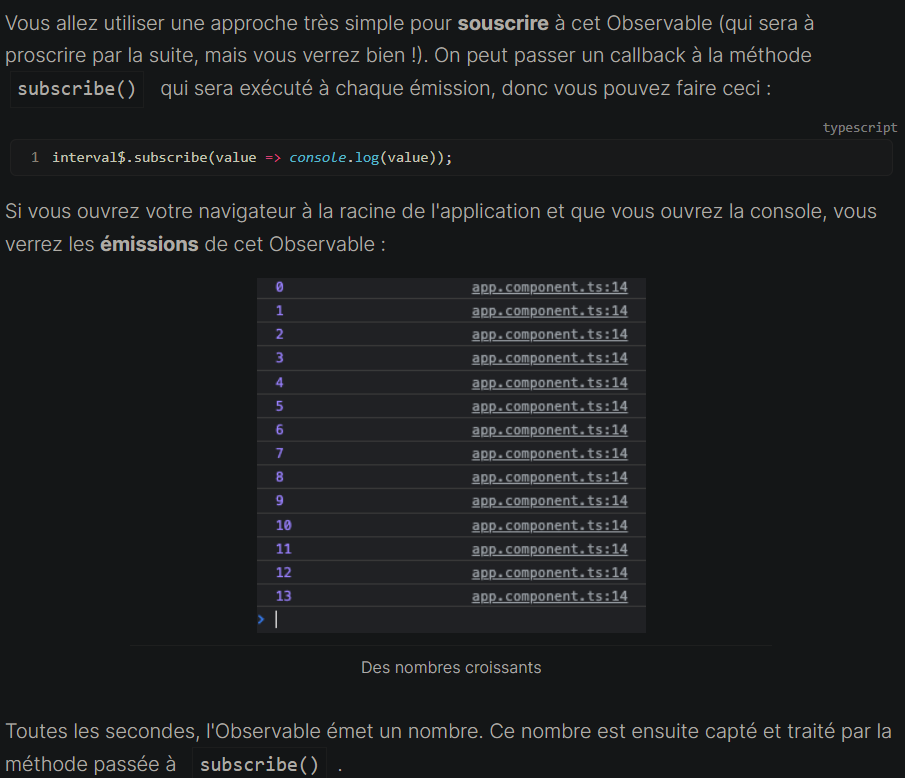
****

****

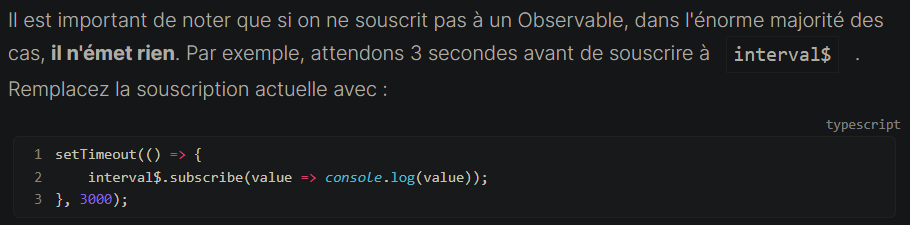
Create an observable

****

****

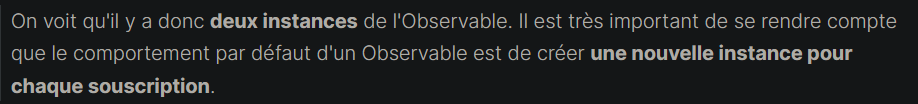
****

If we don’t subscribe to an observable, il n’émet rien !

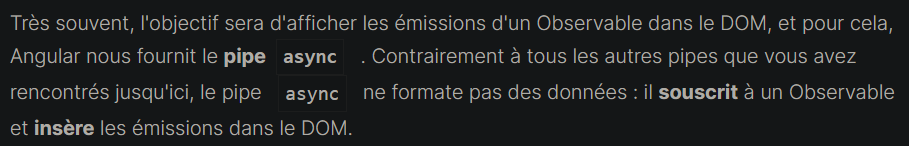
****

Si on souscrit 2 fois à un observable, on crée 2 instances de l’objet qui sont indépendants !

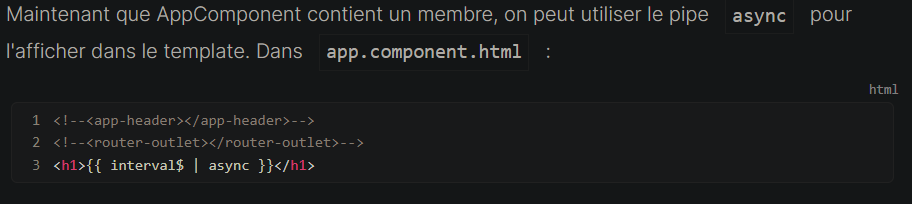
****

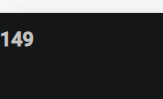
****

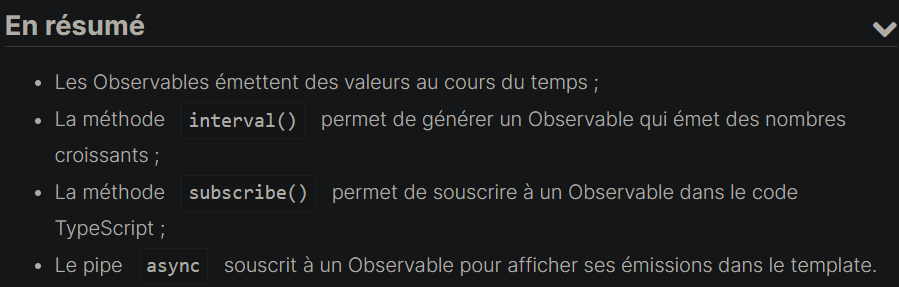
**Pipe async =>** permet d’inscrire à un observable et émettre ses valeurs dans le dom.

****

****

****

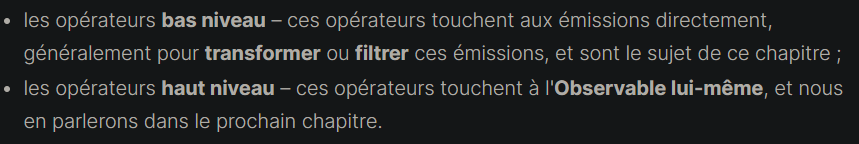
**Au bout de 149s.**

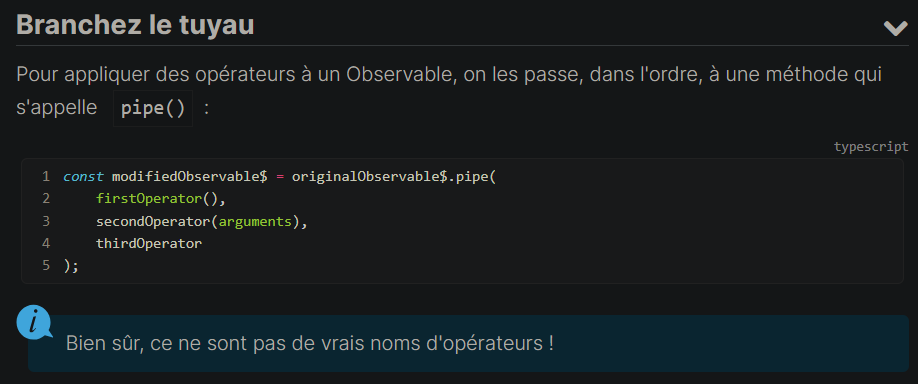
****

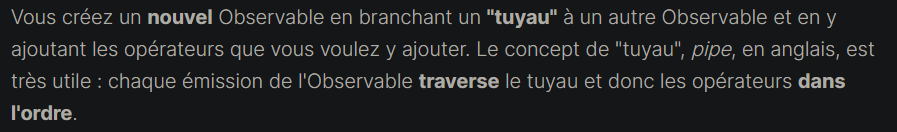
Les observables occupent une place très importante dans le framework d’Angular, et nous permettent de créer des fonctionnalités très avancées et complexes beaucoup plus facilement.

Il est donc essentiel de maîtriser la manipulation de ces observables. Cette manipulation se fait avec les opérateurs.

Il existe deux types principaux d’opérateurs :



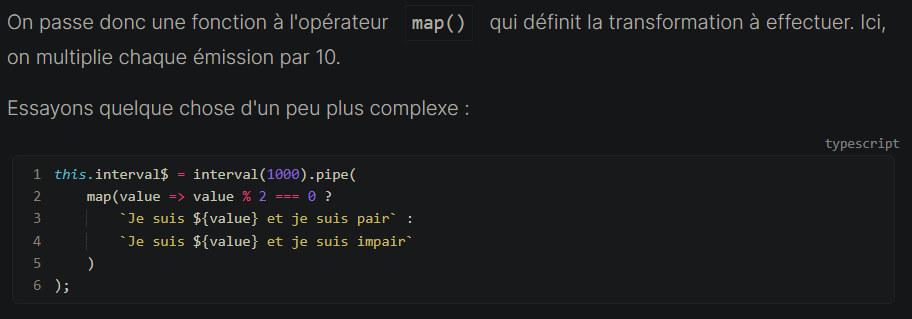




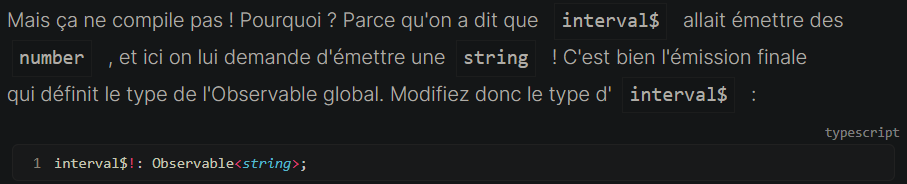
Obsevable.pipe(

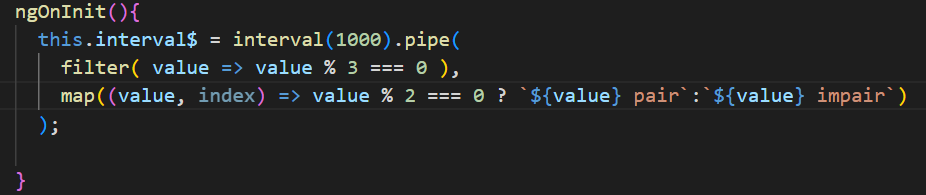
Map(element => element\*5

) ;



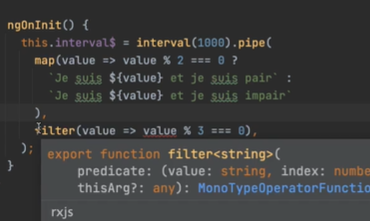
Là on a changé le type de retour en string, il faut donc changer le type de l’observable.





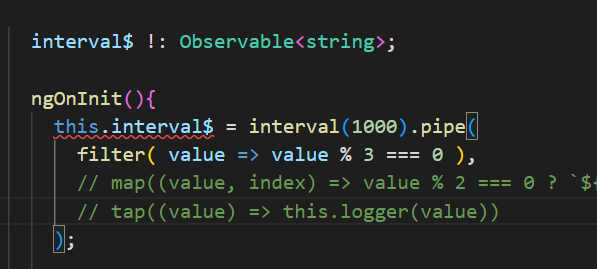
On peut ajouter un 2eme opérateur : filter.

L’ordre est très important, si on met le filter après map :

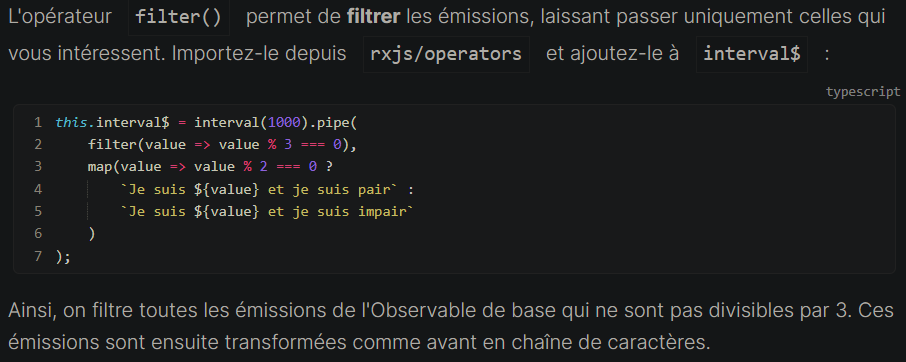


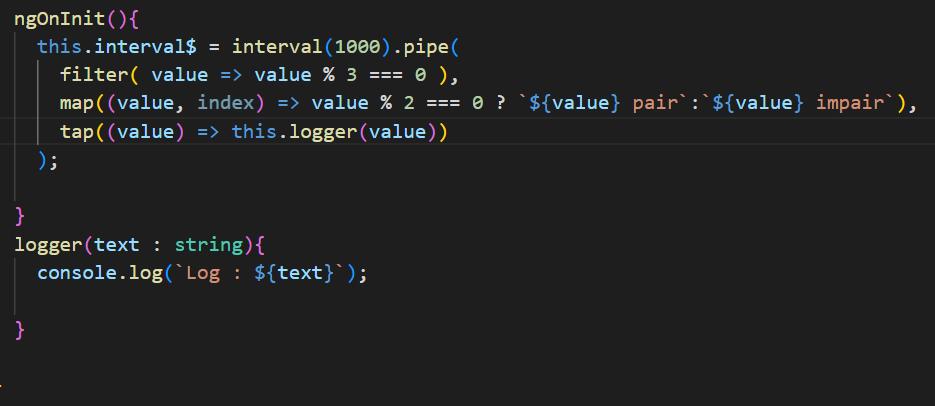
Ça ne marche car le type de retour de map est string, et string divisé par number est impossible !

Il faut bien comprendre que observable<string> veut dire que le renvoie final est string, l’observable peut contenir des numbers mais à la fin il doit à tout coup printer des strings



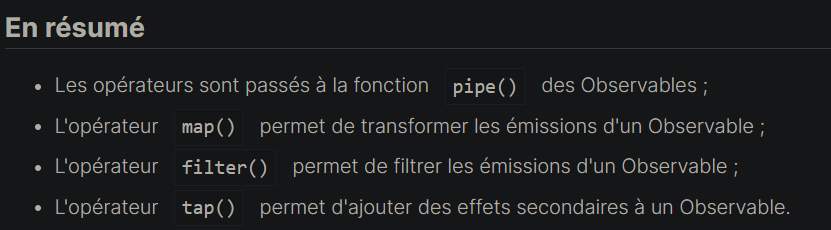
Si on enlève le renvoie final qui retourne des strings, l’intervalle contient des numbers et non pas des strings !





L’opérateur tap permet de réagir aux différentes émissions de l’observable sans toucher à l’observable lui-même.





Observable de haut niveau !

C’est un observable qui souscrit à d’autres observables !

