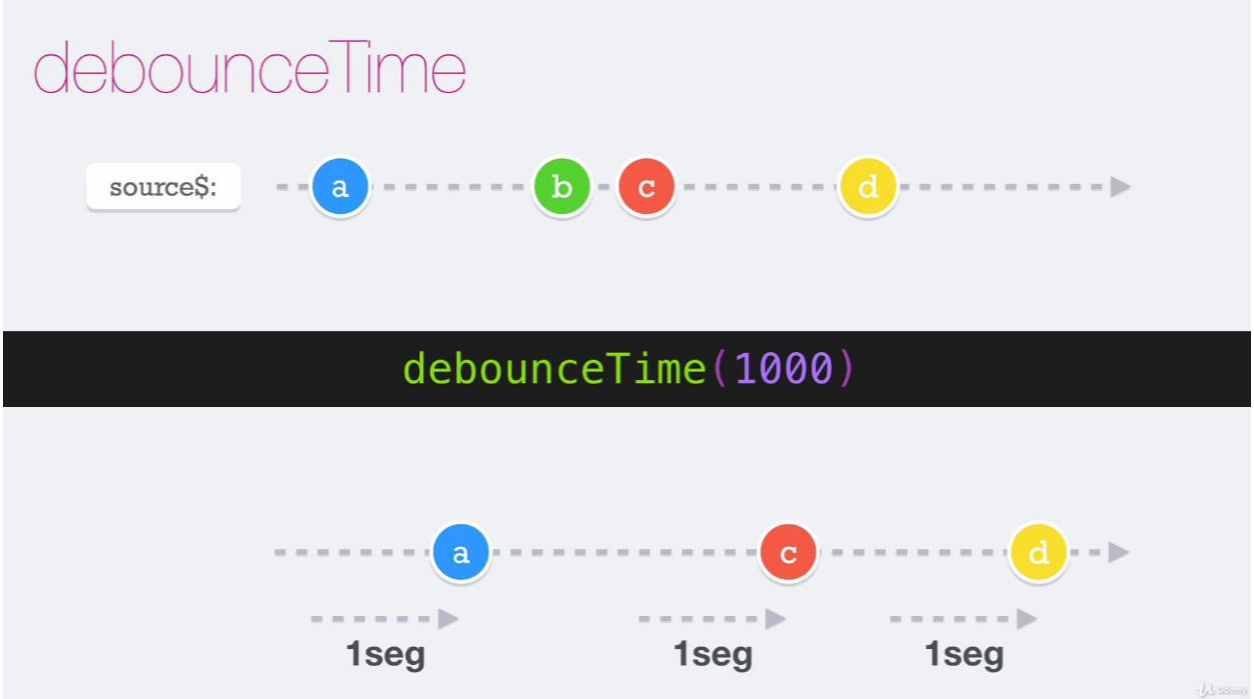
58. debounceTime

El operador debounceTime es un operador que trabaja en base a intervalos de tiempo, y básicamente este nos ayuda a contar cuantas milésimas de segundo han pasado desde la última emisión, y si esas milésimas de segundo sobrepasan el parámetro que tengamos en los paréntesis entonces emitirá dicho valor. También el debounceTime nos va a ayudar a nosotros a poder restringir la cantidad de emisiones que nuestro source o nuestro observable inicial esta emitiendo.

Este es sumamente útil cuando queremos controlar observables que emiten una gran cantidad de mensajes rápidamente.

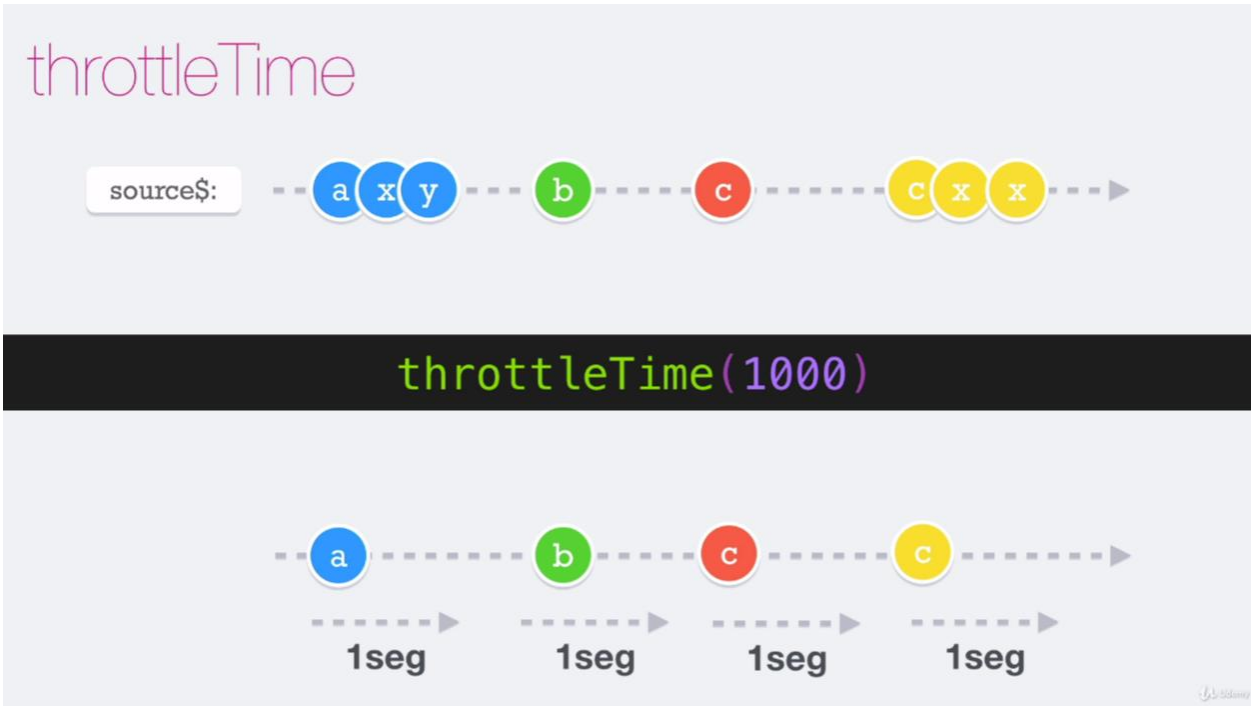


59. throttleTime

El operador throttleTime es un operador muy similar al debounceTime pero funciona un poquito diferente. Ya que si por ejemplo se emite un valor va a empezar a contar 1 segundo y si dentro de ese segundo se emiten varios valores estos van a ser ignorados, pero existe una forma de obtener el primer y último valor.

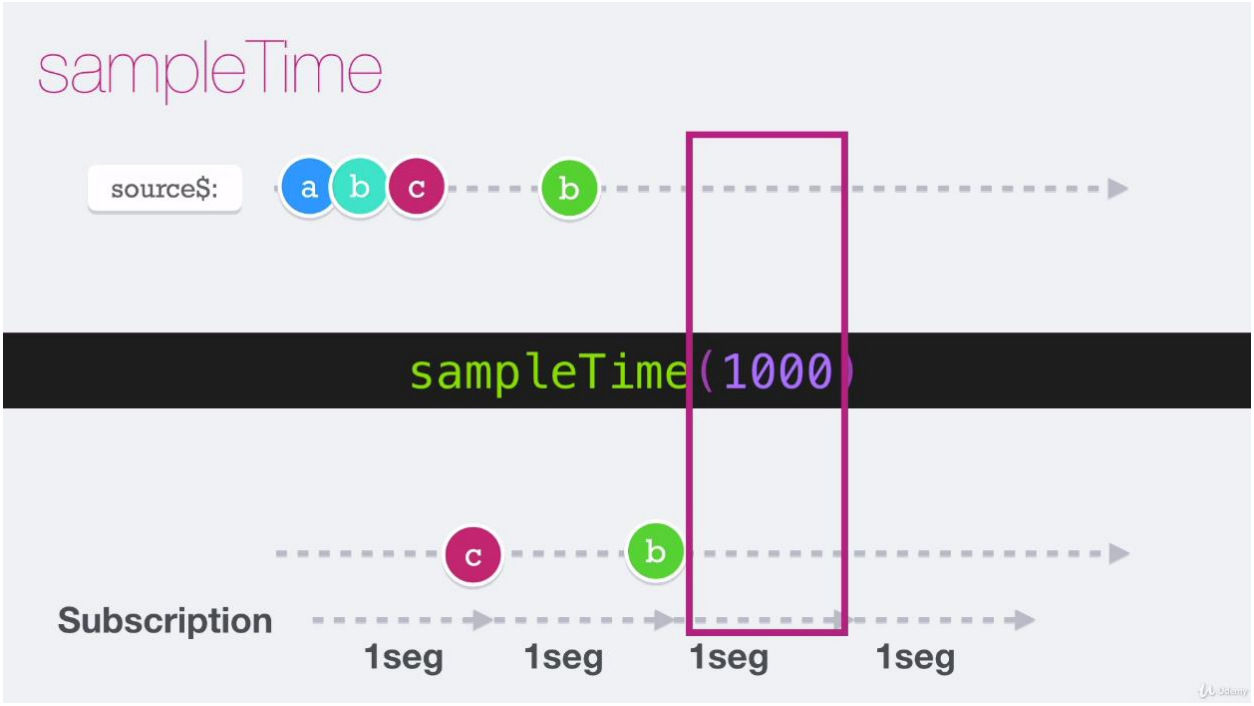
Otra cosa es que podemos considerar que el operador throttleTime es exactamente lo opuesto al debounceTime ya que el debounceTime espera el tiempo especificado para emitir el valor, mientras que el throttleTime emite inmediantamente el valor y luego espera el tiempo especificado.

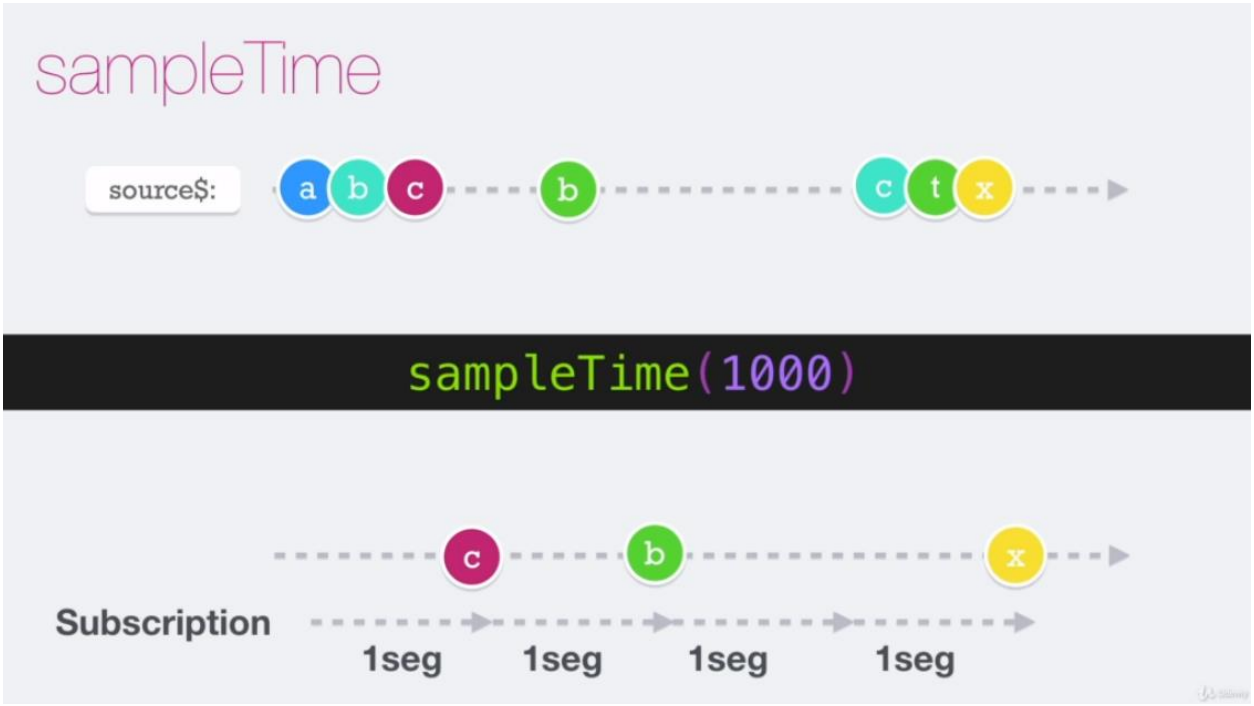
Adicionalmente este es otro operado bastante útil para controlar las emisiones de observables que emiten demasiados valores muy frecuentemente.



60. sampleTime

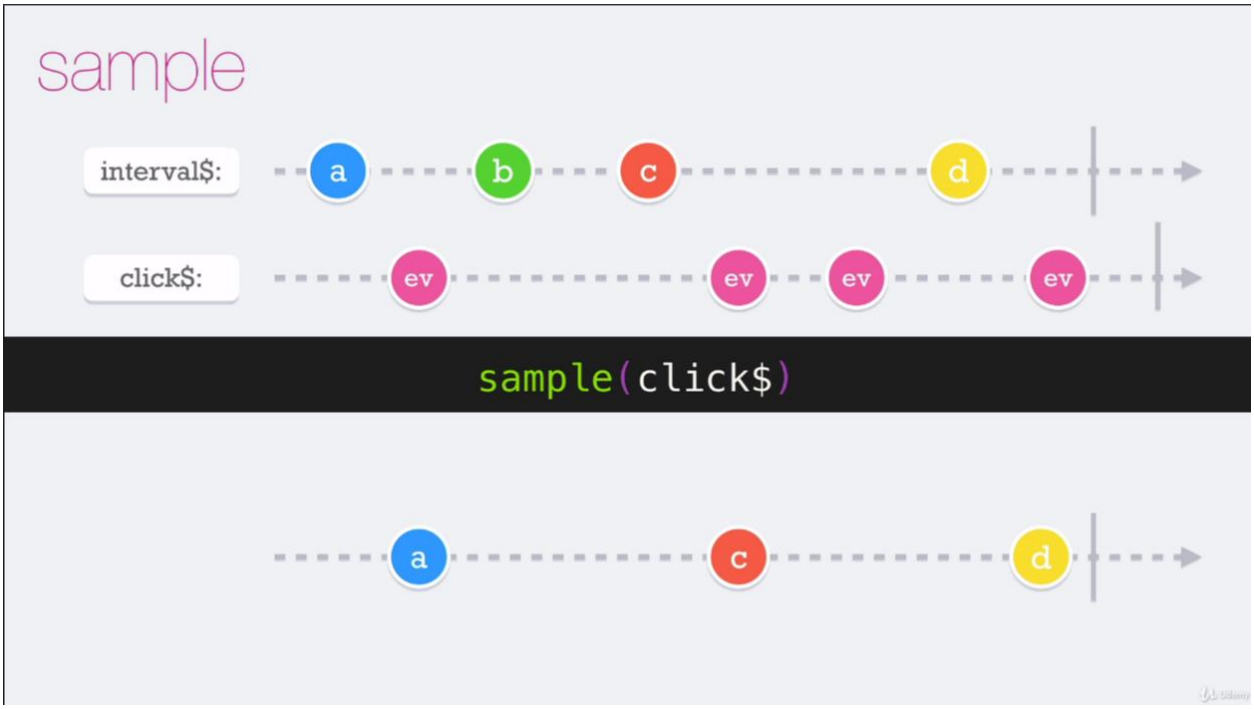
El operador sampleTime es un operador que nos permite obtener el último valor emitido en un intervalo de tiempo y adicionalmente si en ese periodo de tiempo no se emite ningún valor entonces no se emite nada. En pocas palabras este operador nos permite tener una suscripción que esta pendiente de cada una de las emisiones en periodos de tiempo.





61. sample

El operador sample es un operador que emite el último valor emitido por el observable hasta que el otro observable que tenemos dentro del operador sample emita un valor.



62. auditTime

El operador auditTime lo que hace es emitir el último valor que ha sido emitido por el observable en un periodo de tiempo determinado, es decir, inicia a contar y escuchar cuando se emite un valor y mira si hay más valores en dicho tiempo, en caso de ser así emite el último valor. Adicionalmente si el observable se completa antes del tiempo determinado este no va a emitir nada.