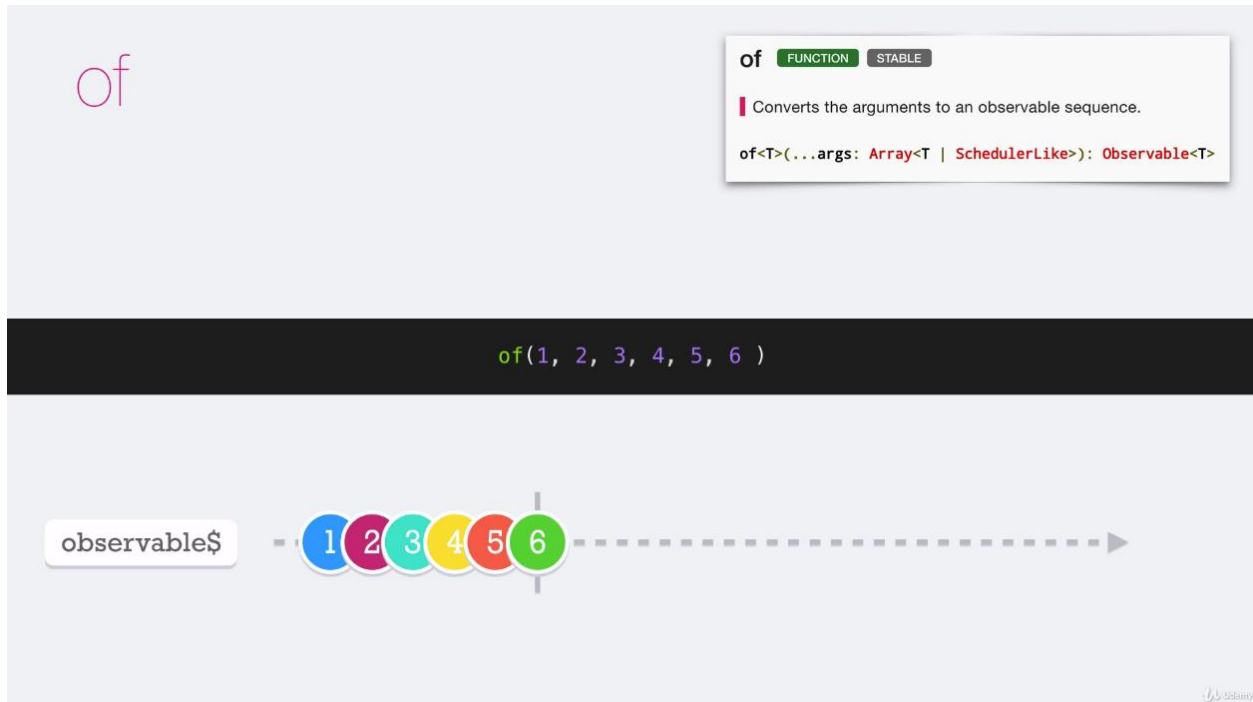


23. Of

El Of es una función que nos permite crear observables en base a un listado de elementos (números y cualquier tipo de objeto). Lo importante es que el operador Of va a emitir los valores en secuencia uno por uno de manera síncrona y cuando emite el último valor automáticamente se completa el observable.



24. fromEvent

Esta función nos permite crear observables en base a un event target, es decir, de cierto tipo que provienen de mi event target, que en el caso del ejemplo sería del document y solo quiero saber los eventos que sean emitidos del document que tienen que ver con el scroll.

fromEvent


fromEvent

FUNCTION

STABLE

Creates an Observable that emits events of a specific type coming from the given event target.

```
fromEvent<T>(target: FromEventTarget<T>, eventName: string, options?: EventListenerOptions | ((...args: any[]) => T), resultSelector?: ((...args: any[]) => T)): Observable<T>
```



```
fromEvent<Event>(document, 'scroll');
```

input\$:

Event


Event


Event

Event

Event

Event





25. range

La función `range` nos crea un observable que emite una secuencia de números en base a un rango. Por defecto son síncronos, pero a su vez se pueden transformar en asíncronos usando un `asyncScheduler`.

range

range

FUNCTION

STABLE

Creates an Observable that emits a sequence of numbers within a specified range.

```
range(start: number = 0, count?: number, scheduler?: SchedulerLike): Observable<number>
```

Parameters

start	Optional. Default is 0. The value of the first integer in the sequence.
count	Optional. Default is undefined. The number of sequential integers to generate.
scheduler	Optional. Default is undefined. A <code>SchedulerLike</code> to use for scheduling the emissions of the notifications.

```
range(1,5);
```

salida:

1

2

3

4

5

range

range

FUNCTION

STABLE

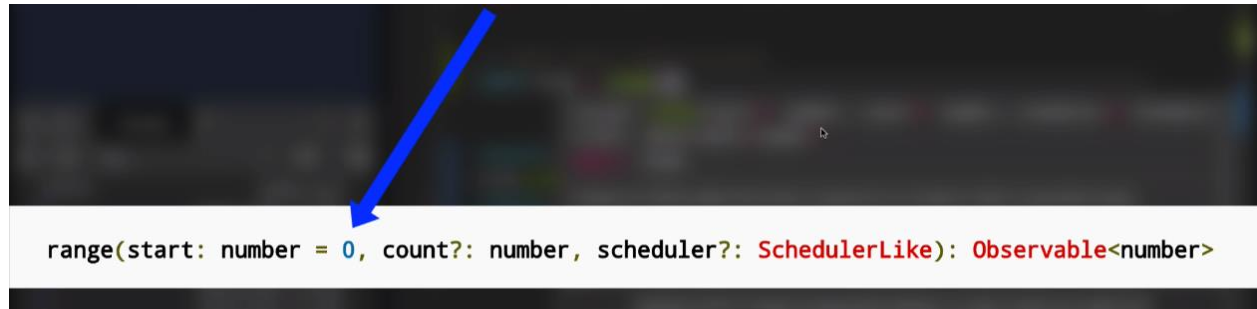
Creates an Observable that emits a sequence of numbers within a specified range.

```
range(start: number = 0, count?: number, scheduler?: SchedulerLike): Observable<number>
```

Parameters

start	Optional. Default is 0. The value of the first integer in the sequence.
count	Optional. Default is undefined. The number of sequential integers to generate.
scheduler	Optional. Default is undefined. A <code>SchedulerLike</code> to use for scheduling the emissions of the notifications.

Adicionalmente hay que saber que el range recibe una posición inicial como primer argumento el cual por defecto es cero, como segundo va la cantidad de emisiones que se desean y al final va un scheduler.



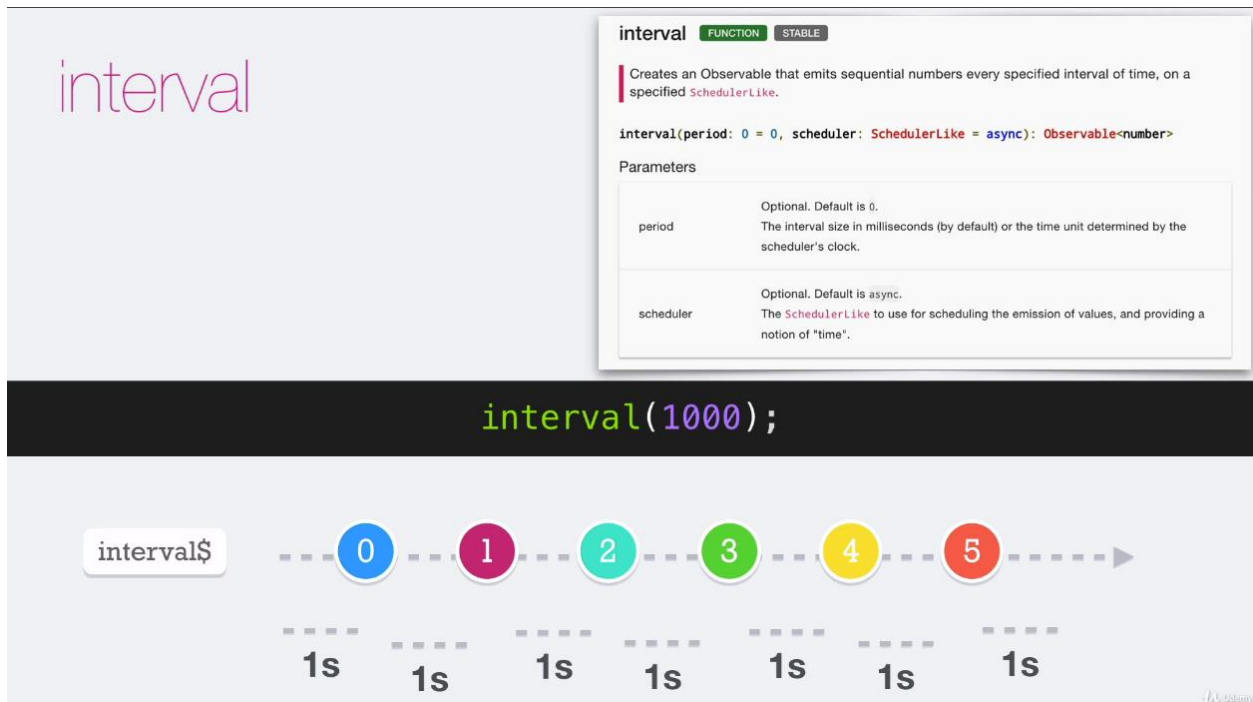
```
range(start: number = 0, count?: number, scheduler?: SchedulerLike): Observable<number>
```

26. interval y timer

El interval y el timer son funciones que permiten crear observables que trabajan con intervalos de tiempo y son dos observables asíncronos por naturaleza.

También hay algo especial es que aunque cancelemos la suscripción del interval este va a seguir corriendo.

Entonces al suscribirnos a la función interval este va a empezar a emitir valores que inician en cero e incrementan dependiendo del parámetro que se indica en milisegundos y este va a emitirlos hasta el fin de los tiempos ya que como tal no tiene un complete a no ser que lo indiquemos. En el caso del siguiente ejemplo vemos que se están emitido valores cada segundo.



Ahora en cuanto a la función timer esta es especial y es muy parecido al interval aunque su comportamiento por defecto podrá ser lo opuesto y de igual manera el timer recibe un parámetro que se indica en milisegundos. Entonces en el siguiente ejemplo vemos que se va a emitir un valor a los 2 segundos y se va a completar y no se siguen emitiendo valores y es bastante útil si sabemos en qué momento es necesario ejecutar dicha tarea aunque hay que aclarar que el timer es más poderoso que solo hacer eso que acabamos de mencionar.

timer

timer FUNCTION STABLE

Creates an Observable that starts emitting after an `dueTime` and emits ever increasing numbers after each `period` of time thereafter.

```
timer(dueTime: number | Date = 0, periodOrScheduler?: number | SchedulerLike, scheduler?: SchedulerLike): Observable<number>
```

Parameters

<code>dueTime</code>	Optional. Default is 0. The initial delay time specified as a Date object or as an integer denoting milliseconds to wait before emitting the first value of 0.
<code>periodOrScheduler</code>	Optional. Default is undefined. The period of time between emissions of the subsequent numbers.
<code>scheduler</code>	Optional. Default is undefined. The <code>SchedulerLike</code> to use for scheduling the emission of values, and providing a notion of "time".

```
timer(2000);
```

interval\$

2s

0

The diagram illustrates the behavior of the `timer(2000)` function. It features a horizontal timeline with a dashed line and an arrow pointing to the right. A vertical tick mark at the start of the timeline is labeled with a blue circle containing the number '0'. A dashed line segment below the timeline, starting from the '0' mark and extending to the left, is labeled '2s', indicating a 2-second delay before the first emission. A box labeled 'interval\$' is positioned to the left of the timeline. The diagram shows that only one value (0) is emitted at the start, and no subsequent values are emitted, which is the default behavior for the `timer` function when no period is specified.