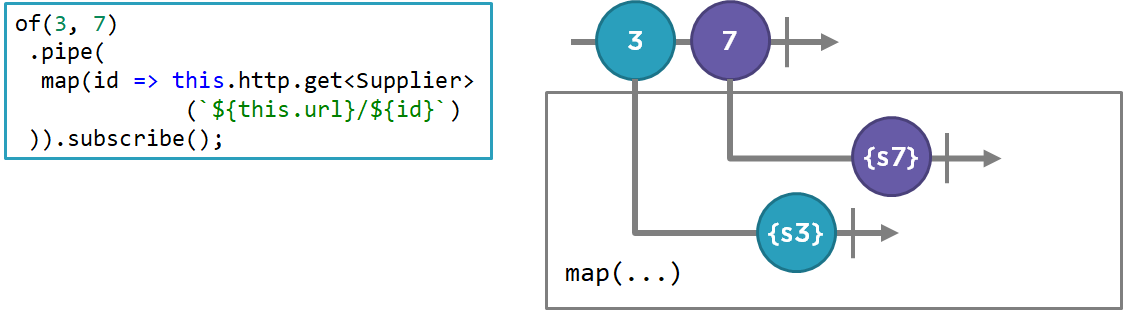
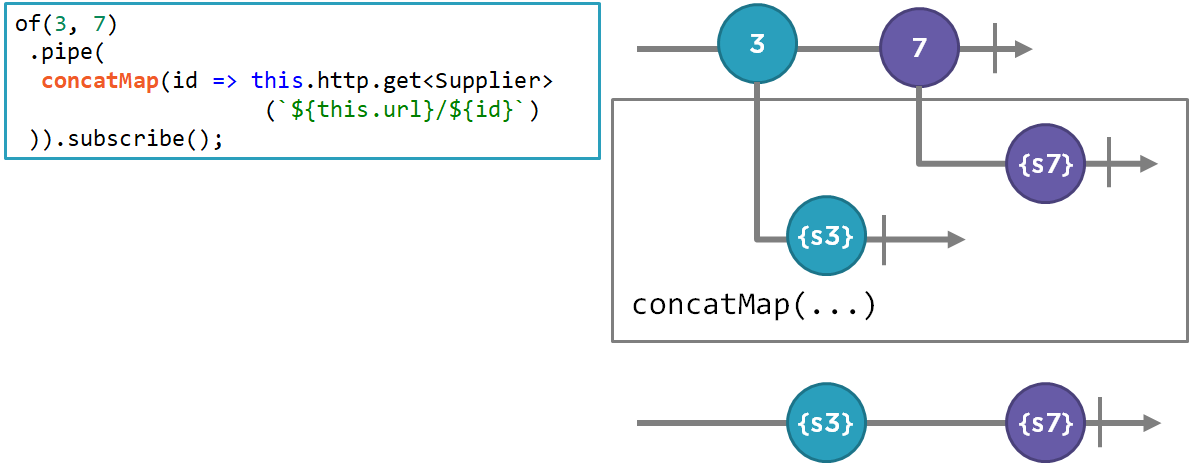
**Observables de orden superior**

Un Observable de orden superior es un Observable que emite otros Observables. "¿Por qué?" Tu puedes preguntar.

Respondamos eso con un ejemplo. Supongamos que recuperamos datos de productos y como parte de esos datos obtenemos un conjunto de identificadores de proveedores. Los usuarios desean ver la lista de proveedores junto con el producto para poder reordenar rápidamente cuando el inventario es bajo. Entonces, dados los identificadores de proveedor, necesitamos recuperar los datos de cada proveedor**.**



**concatMap**

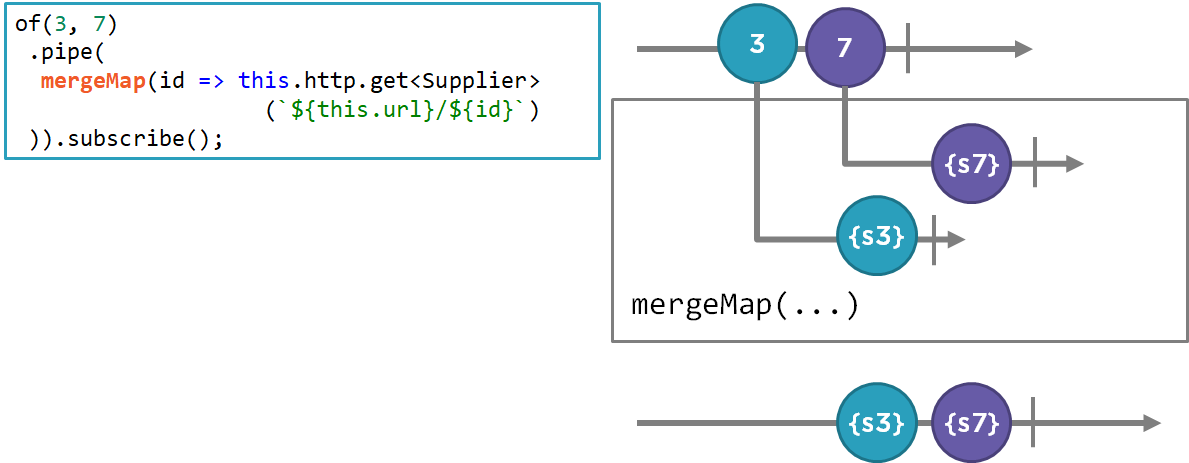
El operador de mapeo de orden superior concatMap \* espera \* a que se complete cada Observable interno antes de procesar el siguiente. Concatena los resultados de los Observables internos en secuencia.

Aquí, el Observable externo emite dos números y por cada número emitido llama http get. Cuando se emite el primer número, el concatMap se suscribe automáticamente al primer Observable interno y se emite la solicitud de obtención de http. Luego \* espera \* la respuesta devuelta. Una vez que se completa el primer Observable interno, emite el resultado a la secuencia de resultados. Solo entonces se suscribe al siguiente Observable interno, emitiendo esa solicitud de obtención. Nuevamente espera la respuesta devuelta. Cuando todos los Observables internos se completan, la secuencia de resultados se completa.

Use concatMap si desea asegurarse de que cada Observable interno se procese uno a la vez y en orden. Esta es una gran técnica para usar al actualizar o eliminar datos para garantizar que cada operación se procese en secuencia.

**mergeMap**

El operador de mapeo de orden superior mergeMap procesa cada Observable interno en paralelo. Fusiona los resultados de los Observables internos a medida que se completan.



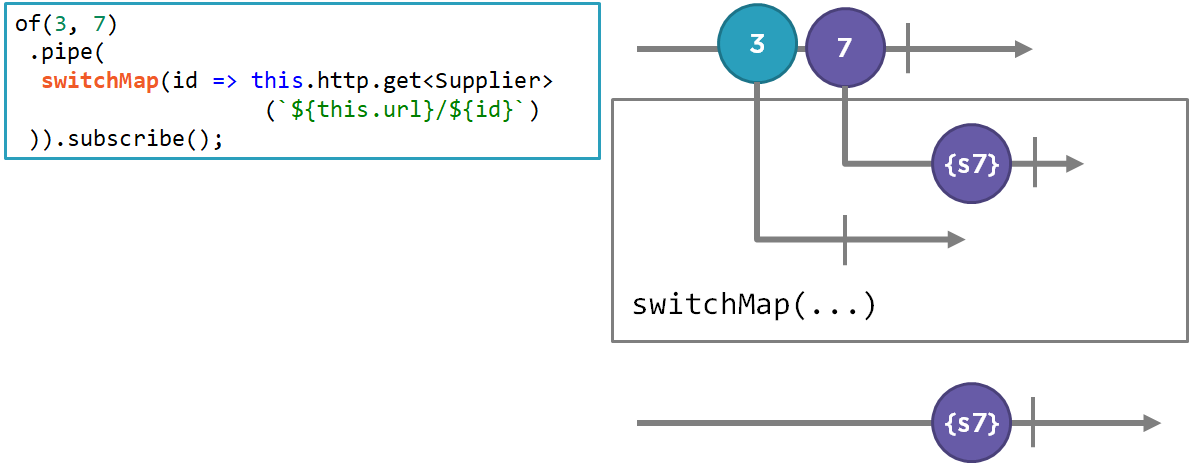
Este es el mismo ejemplo, pero con mergeMap. El Observable externo emite dos números y para cada número emitido llama http get. Cuando se emite cada número, mergeMap se suscribe automáticamente a su Observable interno y emite cada solicitud de obtención de http. A medida que se recibe cada respuesta, se fusiona con la secuencia de resultados. Cuando todos los Observables internos se completan, la secuencia de resultados se completa.

Debido a que los Observables internos se procesan en paralelo, la secuencia de resultados puede no estar en orden secuencial. Si, en el ejemplo anterior, la solicitud de obtención de http para el proveedor s3 tardó más en ejecutarse, su resultado podría emitirse a la secuencia de resultados después del proveedor s7.

Use mergeMap para un mejor rendimiento, ya que cada Observable interno se ejecuta simultáneamente, pero solo si el orden resultante no importa.

switchMap

El operador de mapeo de orden superior switchMap se da de baja de cualquier Observable interno anterior y cambia a cualquier Observable interno nuevo.



Nuevamente, usando el mismo ejemplo pero con switchMap. El Observable externo emite dos números y para cada número emitido llama http get. Cuando se emite el primer número, el switchMap se suscribe automáticamente al primer Observable interno y se emite la solicitud de obtención de http. Cuando se emite el siguiente elemento, switchMap cancela la suscripción del Observable interno anterior y cambia al nuevo Observable interno, suscribiéndose a él. Si se completa un Observable interno, emite el resultado a la secuencia de resultados. Cuando todos los Observables internos se completan, la secuencia de resultados se completa.

Use switchMap para detener cualquier Observable interno anterior antes de procesar el siguiente. Esto es útil para funciones como escribir con anticipación o completar automáticamente cuando desea detener el procesamiento anterior cuando el usuario escribe el siguiente carácter.

NOTA:

No más suscriptores anidados

Cada vez que usamos datos emitidos desde un Observable para emitir otro Observable, necesitamos suscribirnos tanto al Observable externo fuente como al Observable interno emitido. En lugar de suscripciones anidadas, utilizamos operadores de mapeo de orden superior. Los operadores de mapeo de orden superior mapean Observables de orden superior y se encargan de suscribirse y cancelar la suscripción a los Observables internos para que no tengamos que hacerlo.