concat

Concatena varios Observables de entrada, uno tras otro, emitiendo secuencialmente todos los valores de cada uno de ellos

💡 Para emitir valores de varios Observables a la vez (concurrentemente), se puede utilizar merge

Firma

```
concat<O extends ObservableInput<any>, R>(...observables: (SchedulerLike | O)[]): Observable<ObservedValueOf<O> | R>
```

Parámetros

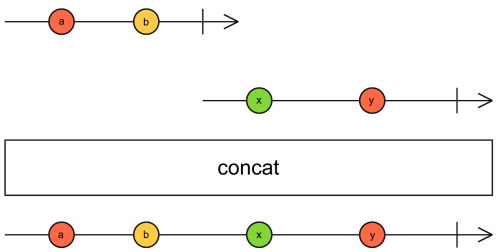
observables	Tipo: `(SchedulerLike | O)[]`.

Retorna

`Observable<ObservedValueOf<O> | R>`: Todos los valores de cada Observable de entrada fusionados en un solo Observable, por orden.

Descripción

Concatena varios Observables, uno tras otro, emitiendo secuencialmente sus valores.



`concat` une varios Observables, suscribiéndose a ellos de uno en uno y fusionando los valores que emitan en el Observable resultante. Los Observables de entrada se pueden proporcionar en un array, o directamente como argumentos. Proporcionarle un array vacío a `concat` resulta en un Observable que se completa inmediatamente.

`concat` se suscribe al primer Observable de entrada y emite todos sus valores intactos, sin cambiarlos ni transformarlos. Cuando ese primer Observable se completa, se suscribe al siguiente Observable y, de nuevo, emite todos sus valores. Este proceso se repite hasta que el operador agote todos los Observables de entrada. Cuando el último Observable de entrada se complete, `concat` también se completará. `concat` emite los valores de un solo Observable cada vez.

`concat` es el equivalente a utilizar el operador `merge` con el parámetro de concurrencia `1`.

Si alguno de los Observables de entrada nunca llega a completarse, `concat` tampoco se completará y los demás Observables de entrada nunca llegarán a ser suscritos. Por otra parte, si alguno de los Observables de entrada se completa inmediatamente después de ser suscrito, será invisible para `concat`, que se suscribirá al siguiente Observable.

Si alguno de los Observables de entrada lanza un error, en lugar de suscribirse al siguiente Observable, `concat` también lanzará un error inmediatamente, y no llegará a suscribirse a los Observables de entrada siguientes al que haya lanzado el error.

Si se le pasa el mismo Observable a `concat` varias veces, su flujo de emisiones se repetirá en cada suscripción. Se puede repetir un Observable tantas veces como se quiera. Sin embargo, si pasarle el mismo Observable a `concat` 1000 veces resulta demasiado tedioso, siempre se puede utilizar repeat.

Ejemplos

Concatenar varios Observables distintos

StackBlitz

```
import { range, from, concat } from "rxjs";
import { ajax } from "rxjs/ajax";

const number$ = range(1, 4);

const fruit$ = from(["Fresa", "Cereza", "Arándano"]);

const totoroFilmData$ = ajax.getJSON(
  "https://ghibliapi.herokuapp.com/films/58611129-2dbc-4a81-a72f-77ddfc1b1b4
```

Copy

```
);

concat(number$, fruit$, totoroFilmData$).subscribe(console.log);
// Salida: 1, 2, 3, 4, 'Fresa', 'Cereza', 'Arándano', { ..., title: 'My Nei
```

Concatenar el mismo Observable para repetirlo

StackBlitz

```
import { from, concat } from "rxjs";

const message$ = from(["RxJS mola"]);

concat(message$, message$, message$).subscribe(console.log);
// Salida: 'RxJS mola', 'RxJS mola', 'RxJS mola'
```

Si uno de los Observables de entrada nunca llega a completarse, concat nunca se suscribirá a los siguientes Observables de entrada

StackBlitz

```
import { interval, from, concat } from "rxjs";

const infinite$ = interval(1000);
const message$ = from(["Nunca", "se", "emitirá"]);

concat(infinite$, message$).subscribe(console.log);
// Salida: 0, 1, 2, 3...
```

Si alguno de los Observables de entrada lanza un error, el Observable resultante lanzará un error inmediatamente, y el flujo se terminará

StackBlitz

```
import { throwError, from, concat } from "rxjs";

const message$ = from(["Este mensaje se emitirá"]);
const error$ = throwError("Oh no");
```

```
const sadMessage$ = from(["No se llega a emitir :("]);

concat(message$, error$, sadMessage$).subscribe(console.log, console.error);
// Salida: 'Este mensaje se emitirá', (error) Oh no
```

Ejemplos de la documentación oficial

Concatenar un temporizador que cuente del 0 al 3 con una secuencia síncrona de los números del 1 al 10

```
import { concat, interval, range } from "rxjs";
import { take } from "rxjs/operators";

const timer = interval(1000).pipe(take(4));
const sequence = range(1, 10);
const result = concat(timer, sequence);
result.subscribe((x) => console.log(x));

// Salida:
// 0 -1000ms-> 1 -1000ms-> 2 -1000ms-> 3 -inmediatamente-> 1 ... 10
```

Concatenar 3 Observables

```
import { concat, interval } from "rxjs";
import { take } from "rxjs/operators";

const timer1 = interval(1000).pipe(take(10));
const timer2 = interval(2000).pipe(take(6));
const timer3 = interval(500).pipe(take(10));

const result = concat(timer1, timer2, timer3);
result.subscribe((x) => console.log(x));

// Salida:
// (Los valores se imprimen por consola secuencialmente)
// -1000ms-> 0 -1000ms-> 1 -1000ms-> ... 9
// -2000ms-> 0 -2000ms-> 1 -2000ms-> ... 5
// -500ms-> 0 -500ms-> 1 -500ms-> ... 9
```

Concatenar el mismo Observable para repetirlo