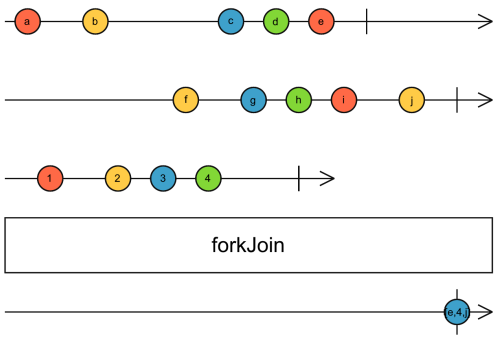
forkJoin

Acepta un Array de Observables o un diccionario de Observables, y retorna otro Observable que emite o bien un array de valores en el mismo orden que el array proporcionado, o un diccionario de valores con la misma forma que el diccionario proporcionado

▶ Signatura

Descripción

Espera a que todos los Observables se completen, y combina sus últimas emisiones.



`forkJoin` es un operador que recibe un array de Observables o un diccionario de Observables como parámetro de entrada. Si no se proporciona ningún Observable de entrada, el Observable resultante se completa inmediatamente.

`forkJoin` espera a que todos los Observables de entrada se completen, y entonces emite un array u objeto con la última emisión de cada uno de estos Observables.

Si se le proporciona un array de *n* Observables a `forkJoin`, el array resultante contendrá *n* valores, donde el primer valor es la última emisión del primer Observable, el segundo valor es la última emisión del segundo Observable, y así sucesivamente.

Si se le proporciona un diccionario de Observables a `forkJoin` el objeto resultante tendrá las mismas claves que el diccionario. Los últimos valores que se hayan emitido por cada Observable de entrada estarán situados bajo la clave correspondiente.

`forkJoin` emite una única vez, y se completará justo después. Si se necesita emitir valores combinados durante el ciclo de vida de los Observables de entrada, se recomienza utilizar combineLatest o zip.

Para que el array resultante tenga la misma longitud que el número de Observables de entrada, cuando alguno de dichos Observables se complete sin emitir ningún valor, `forkJoin` también se completará y no emitirá ningún valor, aunque ya tenga recogidos algunos valores de los demás Observables. Además, si hay algún Observable que nunca llegue a completarse, `forkJoin` tampoco se completará, a no ser que, en cualquier momento, alguno de los demás Observables de entrada se complete sin emitir ningún valor, lo que nos trae de vuelta al caso anterior. Como norma general, para que `forkJoin` pueda emitir un valor, todos los Observables de entrada tienen que emitir como mínimo un valor, y completarse.

Si alguno de los Observables de entrada lanza un error, `forkJoin` también lo hará, y se cancelará la suscripción a todos los demás Observables de entrada.

Opcionalmente, `forkJoin` recibe una función de proyección, que se llamará con los valores que normalmente se emitirían en el array resultante. El resultado de la función de proyección, sea cual sea, se emitirá en el Observable resultante. Debido a esto, se puede considerar a la función de proyección como una función que recoge todos los argumentos que recibe en un array. La función de proyección se llamará solo cuando el Observable resultante tenga que emitir un valor.

Ejemplos

Combinar la última emisión de dos Observables distintos

StackBlitz

```
import { forkJoin, from, of } from "rxjs";

const language$ = forkJoin([
  of("Java", "Ruby", "Haskell"),
  from(["Orientado a objetos", "Multiparadigma", "Funcional"]),
]);

// Combinar la última emisión de dos Observables distintos
```

```
language$.subscribe(console.log);
// Salida: ["Haskell", "Funcional"]
```

Combinar la última emisión de dos Observables distintos, contenidos en un diccionario de datos

StackBlitz

```
import { forkJoin, from, of } from "rxjs";

const languageDictionary$ = forkJoin({
  language: of("Java", "Ruby", "Haskell"),
  type: from(["Orientado a objetos", "Multiparadigma", "Funcional"]),
});

languageDictionary$.subscribe(console.log);
// Salida: { language: Haskell, type: Funcional }
```

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, forkJoin } from "rxjs";

const message$ = from(["Este mensaje se emitirá"]);
const error$ = throwError("Oh no");
const sadMessage$ = from(["No se llega a emitir :("]);

forkJoin([message$, error$, sadMessage$]).subscribe(console.log, console.erro
// Salida: 'Este mensaje se emitirá', (error) Oh no
```

Si se utiliza el operador `catchError` en el Observable de entrada que lanza el error, el Observable resultante se completará sin problemas

StackBlitz

```
import { from, forkJoin, of, throwError } from "rxjs";
import { catchError } from "rxjs/operators";

const message$ = from(["Este mensaje se emitirá"]);

// Capturando el error con catchError
const error$ = throwError("Oh no").pipe(catchError((err) => of(err)));

const happyMessage$ = from(["Ahora sí se emite :)"]);

forkJoin([message$, error$, happyMessage$]).subscribe(console.log);
// Salida: ['Este mensaje se emitirá', 'Oh no', 'Ahora sí se emite :)']
```

Copy

Ejemplos de la documentación oficial

Usar forkJoin con un diccionario de Observables de entrada

```
import { forkJoin, of, timer } from "rxjs";

const observable = forkJoin({
  foo: of(1, 2, 3, 4),
  bar: Promise.resolve(8),
  baz: timer(4000),
});
observable.subscribe({
  next: (value) => console.log(value),
  complete: () => console.log("¡Y así es como acaba!"),
});

// Salida:
// { foo: 4, bar: 8, baz: 0 } tras 4 segundos
// "¡Y así es como acaba!" inmediatamente después
```

Copy

Usar forkJoin con un array de Observables de entrada