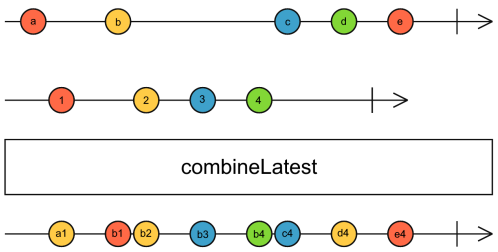
combineLatest

Combina varios Observables para crear otro Observable cuyos valores se calculan a partir de las emisiones más recientes de cada uno de sus Observables de entrada

▶ Signatura

Descripción

Cuando uno de los Observables de entrada emite un valor, utiliza las últimas emisiones de todos los Observables de entrada para computar el valor que se emite en el Observable resultante.



`combineLatest` combina los valores de todos los Observables de entrada. Para ello, se suscribe a cada uno de los Observables en orden, y cuando alguno de los Observables emite, recoge las emisiones más recientes de cada uno en un array. Por tanto, si se le proporcionan *n* Observables al operador, el Observable retornado siempre emitirá un array de *n* valores, en el orden en el que los Observables se hayan pasado como parámetros (el valor del primer Observable estará en la primera posición del array etc.)

La versión estática de `combineLatest` acepta un array de Observables o varios Observables pasados directamente como argumentos individuales. Se debe tener en cuenta que el array de Observables es una buena opción, si no se sabe de antemano cuántos Observables se van a

combinar. Proporcionarle un array vacío a `combineLatest` resulta en un Observable que se completa inmediatamente.

Para asegurar que el array de salida siempre tenga la misma longitud, `combineLatest` espera a que todos los Observables de entrada emitan al menos una vez, antes de empezar a emitir. Esto quiere decir que si algún Observable emite varios valores antes de que los demás Observables emitan su primer valor, todos los valores que emita, excepto el último, se perderán. Por otra parte, si algún Observable se completa sin emitir ningún valor, el Observable resultante se completará en ese mismo momento sin emitir nada, ya que sería imposible incluir el valor del Observable completado en el array resultante. Además, si alguno de los Observables de entrada no emite ningún valor, ni llega a completarse nunca, `combineLatest` nunca emitirá ningún valor, ni llegará a completarse, dado que tiene que esperar a que todos los Observables emitan algún valor antes de poder emitir.

Si se le proporciona al menos un Observable a `combineLatest` y todos los Observables proporcionados han emitido un valor, el Observable resultante se completará cuando todos los Observables se completen. Por tanto, aunque alguno de los Observables de entrada se complete, `combineLatest` seguirá emitiendo valores mientras los demás Observables sigan haciéndolo. En el caso del Observable completado, su valor siempre será el último valor emitido. Por otra parte, si alguno de los Observables lanza un error, `combineLatest` también lanzará un error inmediatamente, y se cancelará la suscripción a todos los Observables restantes.

`combineLatest` acepta una función de proyección como parámetro opcional, que recibe como argumento todos los valores que se emitirían en el Observable resultante. La función `project` puede retornar cualquier tipo de valor, que será emitido en el Observable resultante en lugar del array por defecto. Se debe tener en cuenta que `project` no recibe como argumento una array de valores, sino los valores en sí mismos. Por tanto, la función `project`, por defecto, puede considerarse como una función que recoge en un array todos los argumentos que recibe.

Ejemplos

Ejemplos de la documentación oficial

Combinar dos Observables `timer`

```
import { combineLatest, timer } from "rxjs";

const firstTimer = timer(0, 1000); // emit 0, 1, 2... after every second, st
const secondTimer = timer(500, 1000); // emit 0, 1, 2... after every second,
```

Copy

```
const combinedTimers = combineLatest(firstTimer, secondTimer);
combinedTimers.subscribe((value) => console.log(value));
// Salida:
// [0, 0] tras 0.5s
// [1, 0] tras 1s
// [1, 1] tras 1.5s
// [2, 1] tras 2s
```

Combinar un array de Observables

Copy

```
import { combineLatest, of } from "rxjs";
import { delay, startWith } from "rxjs/operators";

const observables = [1, 5, 10].map((n) =>
  of(n).pipe(
    delay(n * 1000), // emite 0 y después emite n tras n seconds
    startWith(0)
  )
);
const combined = combineLatest(observables);
combined.subscribe((value) => console.log(value));
// Salida:
// [0, 0, 0] inmediatamente
// [1, 0, 0] tras 1s
// [1, 5, 0] tras 5s
// [1, 5, 10] tras 10s
```

Usar la función de proyección para calcular el índice de masa corporal dinámicamente