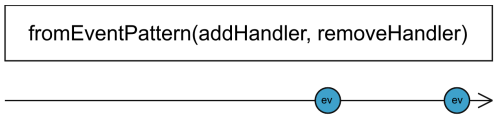
fromEventPattern

Crea un Observable a partir de una API arbitraria de registro de manejadores de eventos

▶ Signatura

Descripción

Se utiliza cuando `fromEvent` no está preparado para manejar un determinado método para añadir manejadores de eventos.



`fromEventPattern` permite convertir cualquier API de registro de funciones manejadores en eventos, en un Observable. Es similar a `fromEvent`, pero mucho más flexible. De hecho, todos los casos de uso de `fromEvent` podrían manejarse con `fromEventPattern` (aunque de forma más verbosa.)

Este operador recibe una función `addHandler` como primer argumento, que se inyecta con un parámetro `handler`. Dicho `handler` es una función de manejo de eventos que se le puede proporcionar a la API que la espera. `addHandler` se llamará cuando se realice alguna suscripción al Observable retornado por `fromEventPattern`, por lo que el registro del manejador en la API no tiene que ocurrir necesariamente cuando se llame a `fromEventPattern`.

Tras el registro, cada vez que se dispare un evento al que se está escuchando, el Observable retornado por `fromEventPattern` emitirá el valor con el que se haya llamado a la función de manejo de eventos. Si el manejador de eventos se ha llamado con más de un argumento, ningún argumento a partir del segundo, este inclusive, aparecerá en el flujo resultante.

Si la API utilizada permite desvincular manejadores de eventos, se le puede pasar un segundo parámetro a `fromEventPattern`: la función `removeHandler`. Se inyectará con la misma función manejadora que antes, pero ahora se puede utilizar para desvincularla de la API. `removeHandler` será llamada cuando el consumidor del Observable resultante cancele la suscripción a dicho Observable.

En algunas APIs, la desvinculación del manejador de eventos se maneja de otra forma. Al vincular un manejador de eventos, se retorna algún tipo de token, que o bien se utiliza después para identificar qué función se debe desvincular, o el propio token contiene un método para desvincular el manejador de eventos. Si ese es el caso, se debe asegurar que el token retornado por el método de registro lo retorna la función `addHandler`. Entonces se le proporcionará a `removeHandler` como segundo argumento, donde se podrá hacer uso de él.

Si se necesita tener acceso a todos los parámetros del manejador de eventos, o se necesita poder transformarlos, se le puede proporcionar un tercer parámetro opcional a `fromEventPattern`: una función de proyección que acepta todos los argumentos pasados al manejador de eventos. El resultado de la función de proyección aparecerá en el flujo resultante en lugar del primer argumento del manejador de eventos.

Ejemplos

Ejemplos de la documentación oficial

Emitir los clicks que ocurran en el DOM

```
import { fromEventPattern } from "rxjs";

function addClickHandler(handler) {
  document.addEventListener("click", handler);
}

function removeClickHandler(handler) {
  document.removeEventListener("click", handler);
}

const clicks = fromEventPattern(addClickHandler, removeClickHandler);
clicks.subscribe((x) => console.log(x));
// (click) MouseEvent {} (click) MouseEvent {}
```

Usar `fromEventPattern` con una API que retorna un token de cancelación

```
import { fromEventPattern } from "rxjs";

const token = someAPI.registerEventHandler(function () {});
someAPI.unregisterEventHandler(token); // El método de cancelación de esta A

const someAPIObservable = fromEventPattern(
  function (handler) {
    return someAPI.registerEventHandler(handler);
  }, // Aquí se retorna el token...
  function (handler, token) {
    someAPI.unregisterEventHandler(token);
  } // ...para utilizarlo aquí
);
```

Usar `fromEventPattern` con función de proyección