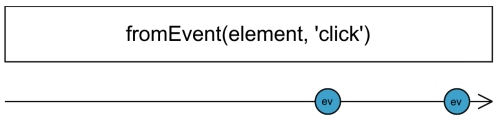
fromEvent

Crea un Observable que emite eventos de un tipo específico, originados en el event target proporcionado

▶ Signatura

Descripción

Crea un Observable a partir de eventos del DOM, de eventos EventEmitter de Node.js u otros.



`fromEvent` acepta un *event target* como primer argumento. Un *event target* es un objeto con métodos para registrar las funciones de manejo de eventos. Como segundo argumento recibe una cadena que indica el tipo de evento al que se quiere escuchar. `fromEvent` es compatible con varios tipos de *event targets*, listados un poco más abajo. Si se desea utilizar *event target* que no sea compatible con `fromEvent`, se debe utilizar `fromEventPattern`, que da soporte a APIs arbitrarias. En el caso de las APIs compatibles con `fromEvent`, los métodos para añadir y elminar funciones de manejo de eventos se llaman de diferente manera, pero todos aceptan una cadena que describe el tipo de evento y la función en sí, que puede llamarse cuando dicho evento se dispare.

Cada vez que se realiza una suscripción al Observable resultante, la función de manejo de eventos se registra al `event target`. Cuando el evento se dispare, el valor que se pase como primer argumento a la función registrada será emitido por el Observable resultante. Cuando se cancele la suscripción al Observable, la función se desvinculará del *event target*.

Se debe tener en cuenta que si las llamadas a la función registrada al *event target* se hacen con más de un argumento, ningún argumento a partir del segundo, este inclusive, aparecerá en el flujo resultante. Para poder acceder a dichos argumentos, se le puede proporcionar una

función de proyección opcional a `fromEvent`, que se llamará con todos los argumentos proporcionados al manejador de eventos. El Observable resultante emitirá los valores retornados por la función de proyección, en lugar del valor habitual.

También debe tenerse en cuenta que los *event targets* listados más adelante se comprueban mediante duck typing, o tipifación dinámica. Esto implica que, independientemente del tipo de objeto y del entorno en el que se trabaje, se puede utilizar `fromEvent` en dicho objeto si se exponen los métodos descritos (siempre y cuando se tengan el comportamiento descrito anteriormente). Por ejemplo, si una biblioteca de Node.js expone un *event target* cuyos métodos se llaman igual que los del `EventTarget` del DOM, el utilizar `fromEvent` es una buena elección.

Si la API que se desea utilizar es más orientada a *callback* que a *event handler* (la función *callback* suscrita se dispara únicamente una vez, por lo que no hay necesidad de desvincularla manualmente), se debe utilizar `bindCallback` o `bindNodeCallback` en lugar de `fromEvent`.

`fromEvent` es compatible con los siguientes tipos de *event targets*:

DOM EventTarget

Es un objeto con los métodos `addEventListener` y `removeEventListener`.

En el navegador, `addEventListener` recibe, además de la cadena indicando el tipo de evento y la función de manejo de eventos, un tercer parámetro opcional, que es o bien un objeto o un booleano, ambos utilizados para agregar configuración adicional de cuándo y cómo se hará la llamada a la función proporcionada.

Node.js EventEmitter

Es un objeto con los métodos `addListener` y `removeListener`.

JQuery-style event target

Es un objeto con los métodos `on` y `off`.

DOM NodeList

Es una lista de Nodos del DOM, como por ejemplo, la que retornan `document.querySelectorAll` o `Node.childNodes`.

Aunque esta colección no es un *event target* propiamente dicho, `fromEvent` iterará a través de todos los Nodos que contenta e instalará la función de manejo de eventos en cada uno de ellos. Cuando se cancele la suscripción al Observable retornado, la función será retirada de todos los Nodos.

DOM HtmlCollection

Al igual que en el caso de un `NodeList`, se trata de una colección de Nodos del DOM. En este caso, la función de manejo de eventos se vinculará y desvinculará de cada uno de los elementos.,

---

Ejemplos

Crear un Observable que emite clicks

StackBlitz

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

const click$ = fromEvent<MouseEvent>(document, "click");

click$.subscribe((click) => console.log(click));
// Salida: (click) MouseEvent {isTrusted: true}
```

Crear un Observable que emite teclas pulsadas

StackBlitz

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

const keyPressed$ = fromEvent<KeyboardEvent>(document, "keydown");

keyPressed$.subscribe(console.log);
// Salida: (pulsar tecla) KeyboardEvent {isTrusted: true}
```

Crear un Observable que emita cambios en el scroll

StackBlitz

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

const scroll$ = fromEvent<UIEvent>(document, "scroll");
```

```
scroll$.subscribe((scroll) => console.log(scroll));
// Salida: (scroll) UIEvent {isTrusted: true}
```

Crear un Observable que emite cuando se copie un texto

StackBlitz

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

const copie$ = fromEvent<ClipboardEvent>(document, "copy");

copie$.subscribe(console.log);
// Salida: (copiar) ClipboardEvent {isTrusted: true}
```

Copy

Ejemplos de la documentación oficial

Emitir los clicks que ocurran en el DOM

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

const clicks = fromEvent(document, "click");
clicks.subscribe((x) => console.log(x));

// Salida:
// (click) MouseEvent{...} (click) MouseEvent{...}
```

Copy

Usar addEventListener con la opción de captura