RxJS: How to Observe an Object

Nicholas Jamieson Follow
Jun 14, 2018 · 2 min read

Photo by Dose Media on Unsplash

A while ago, John Lindquist published a package named `rx-handler`. With it, you can create event handler functions that are also observables.

When it was published, I noticed a few queries about whether something similar could be done with Angular’s `Input` properties—so that they, too, could be treated as observables.

I was thinking about this last night and I’ve published a small package that can be used to observe the properties and methods of an arbitrary object: `rxjs-observe`.

How does it work?

Let’s look at an example:

```
import { observe } from "rxjs-observe";

const instance = { name: "Alice" };
const { observables, proxy } = observe(instance);
observables.name.subscribe(value => console.log(name));
```

When an object `instance` is passed to the `observe` function, it returns an `observables` object—containing an observable source for the `name` property—and a `proxy` instance.

When the `name` property of the `proxy` is assigned, the observable source emits the assigned value—which is written to the console.

The TypeScript declaration for `observe` ensures that the `observables` object is strongly-typed—containing appropriately-typed observables for each property and method on the `instance`.

Internally, a `Proxy` is created for the instance. The proxy is used to intercept property assignments and method calls. A proxy is used for `observables`, too, so that an observable source is only created for a property or method if the source is actually used.

How could it be used with Angular?

With Angular's component API, you create a component class and Angular sets its properties and calls its methods. You could use `rxjs-observe` to convert this to an observable API.

Let's look at an example component:

```
import { Component, Input, OnInit, OnDestroy } from "@
import { debounceTime, distinctUntilChanged, switchMap
import { observe } from "rxjs-observe";

@Component({
  selector: "some-component",
  template: "<span>Some useless component that writes
})
class SomeComponent implements OnInit, OnDestroy {
  @Input() public name: string;
  constructor() {
    const { observables, proxy } = observe(this as Som
    observables.ngOnInit.pipe(
      switchMapTo(observables.name),
      debounceTime(400),
      distinctUntilChanged()
```

In the component, `observables` will contain observable sources for:

- calls to the `ngOnInit` method;

- calls to the `ngOnDestroy` method; and
- assignments to the `name` property.

Using those observables, the component: composes another observable that debounces changes to the `name` input; writes said changes to the console; and unsubscribes when the component is destroyed.

Writing the `name` to the console isn’t particularly useful, but it does show how an observable API could be used within an Angular component.

Should it be used with Angular?

`rxjs-observe` was fun to write, but is it something you should really be using in an Angular component?

I dunno; it’s definitely unconventional. And it requires TypeScript 2.8 or later.

So maybe. Or maybe not, but … YOLO.