RxJS tap对源可观察到的每个值执行副作用，并返回一个与源可观察到的相同的可观察对象，直到没有错误。tap是可伸缩操作符，它是RxJS do操作符的更改后的名称。Angular 6使用RxJS 6，它自带了可伸缩操作符，可以独立于可观察对象使用。为了避免与JavaScript关键字的冲突，一些RxJS操作符的名称进行了更改，如将do更改为tap, catch更改为catchError, switch更改为switchAll，最后更改为finalize。这些操作符是从rxjs/操作符导入的。例如，按以下方式导入tap。

可管道操作符如tap，用于可观察的管道功能。tap只在订阅了由tap返回的观察对象时执行副作用。tap可用于调试由可观察对象发出的值或执行任何副作用

tap是一个RxJS可伸缩操作符，它返回与源可观察值相同的可观察值，并可用于执行副作用，如记录源可观察值发出的每个值。tap的声明如下。

public tap(nextOrObserver: Observer | function, error: function, complete: function): Observable

tap有三个可选参数。

**nextOrObserver**: 一个正常的可观察对象来执行副作用。

**error**: 回调可观察源中的错误。

**complete:** 回调以完成源。

示例。

of(1, 2, 3, 4).pipe(

tap(el => console.log("Process "+ el),

err => console.error(err),

() => console.log("Complete")

),

filter(n => n % 2 === 0)

).subscribe(el => console.log("Even number: "+ el));

我们将得到以下输出。

Process 1

Process 2

Even number: 2

Process 3

Process 4

Even number: 4

Complete

在上面的例子中，我们可以看到tap的第一个回调记录了从可观察源发出的每个值。第二个回调是记录错误，但是源中没有可观察到的错误，因此错误不被记录。第三个回调是记录完成消息，即所有值都已从可观察源发出。

由于tap的参数是可选的，我们也可以使用tap，如下所示。

of(1, 2, 3, 4).pipe(

tap(el => console.log("Process "+ el)),

filter(n => n % 2 === 0)

).subscribe(el => console.log("Even number: "+ el));

我们还可以多次使用tap来调试不同阶段的值。找到示例代码片段。

of(1, 2, 3, 4).pipe(

tap(el => console.log(el)),

filter(n => n % 2 === 0),

tap(el => console.log(el)),

map(n => n + 10),

tap(el => console.log(el)),

scan((sum, n) => sum + n),

tap(el => console.log(el))

).subscribe(result => console.log("Result: "+ result));