Thinking Reactive with RxJS

Exercise

Starting Point

Here is our starting point:

```
const Observable = Rx.Observable;

const startButton   = document.querySelector('#start');
const stopButton    = document.querySelector('#stop');
const resetButton   = document.querySelector('#reset');

const start$     = Observable.fromEvent(startButton, 'click');
const stop$      = Observable.fromEvent(stopButton, 'click');
const reset$     = Observable.fromEvent(resetButton, 'click');
const interval$  = Observable.interval(1000);


const data  = {count:0};
const inc   = (acc)=> ({count: acc.count + 1});
const reset = (acc)=> data;

const intervalThatStops$ = interval$
    .takeUntil(stop$);

const incOrReset$ = Observable.merge(
    intervalThatStops$.mapTo(inc),
    reset$.mapTo(reset)
);

start$
    .switchMapTo(incOrReset$)
    .startWith(data)
    .scan((acc, curr)=> curr(acc))
    .subscribe((x)=>
document.querySelector('#output').innerHTML=x.count);
```

Your Challenge – Faster Timers

We would like to have 2 more buttons that make our timer go faster (Half a second or Quarter of a second).

Step 1 – Getting the millisecond value onto the stream

1. Add buttons: Half and Quarter
2. Use Observable.merge to create a starters$ stream which merges the 3 starter streams
    a. Test yourself, do they all start the timer?
3. Use mapTo to map each of these clicks' streams to a millisecond value
4. Use do() to console.log those values

Step 2 – Refactoring to interval by the value on the stream

We cannot configure the streams as it's right now, as the interval is statically set to 1000ms .
We are going to refactor the code in strategy that is very common when composing streams, hang on to your sit now.

a. Inline the streams (remove those variables completely)
    i. interval$
    ii. intervalThatStops$
    iii. incOrReset$
b. Write a function: *intervalActions* that gets a time param and return that observable
c. Change from switchMapTo to switchMap so you get the time from the stream
d. You should have a working solution.