INTRO INTO REACTIVE PROGRAMMING

https://github.io/przemyslawjanpietrzak
https://twitter.com/przemyslawjanp
https://stackoverflow.com/users/5914352/przemyslaw-pietrzak

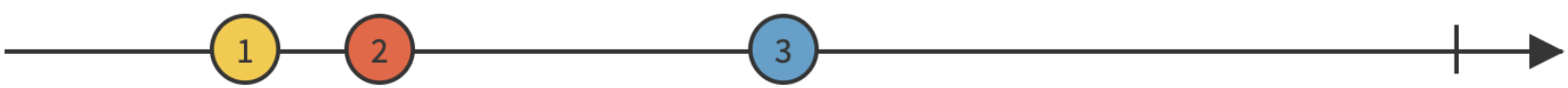
OOP: Alan Key

Objects communicate by asynchronous message passing

Reactive extension

Observable<T>

Flow



Overview

```
import { interval } from 'rxjs';

const timmer$ = interval(1000);

timmer$.subscribe(
  (value) => console.log(value);
  (error) => console.error(error);
  () => console.log('done');
)
```

So what?

```
class Car {
  public setSpeed(speed: int){
    this.speed = speed;
    speedCamera.inform(this, speed);
  }
}

class SpeedCamera {
  public inform(car: Car, speed:
number) {
    ...
  }
}
```

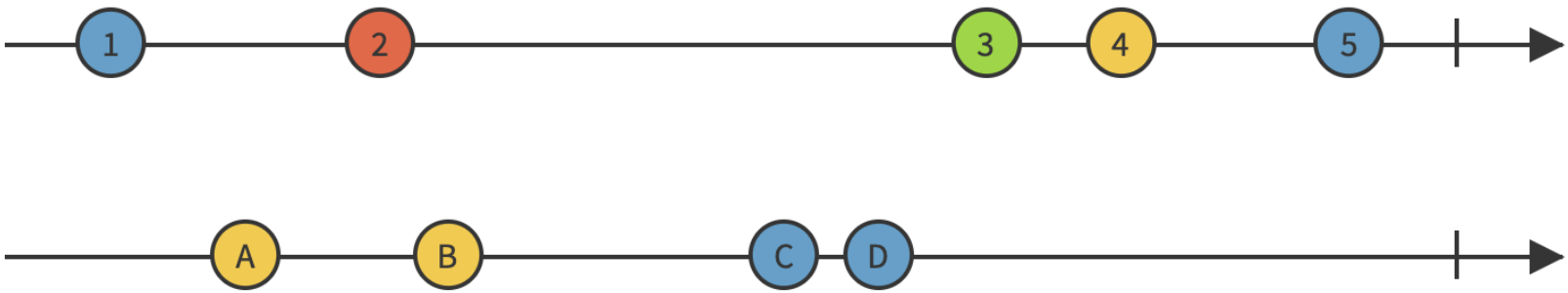
```
class Car {
  public speed$: Observable<int>

  public setSpeed(speed: int){
    this.speed = speed;
    this.speed$.next(speed);
  }
}

class SpeedCamera {
  registerCar(car: Car) {
    car.speed$.subscribe((speed) => {
      ...
    })
  }
}
```

```
import { tempInC$ } from '...';

export const tempInK$ = tempInC$.map(temp => temp + 275)

export const criticalTemp$ = tempInC.filter(temp => temp > 100)
```

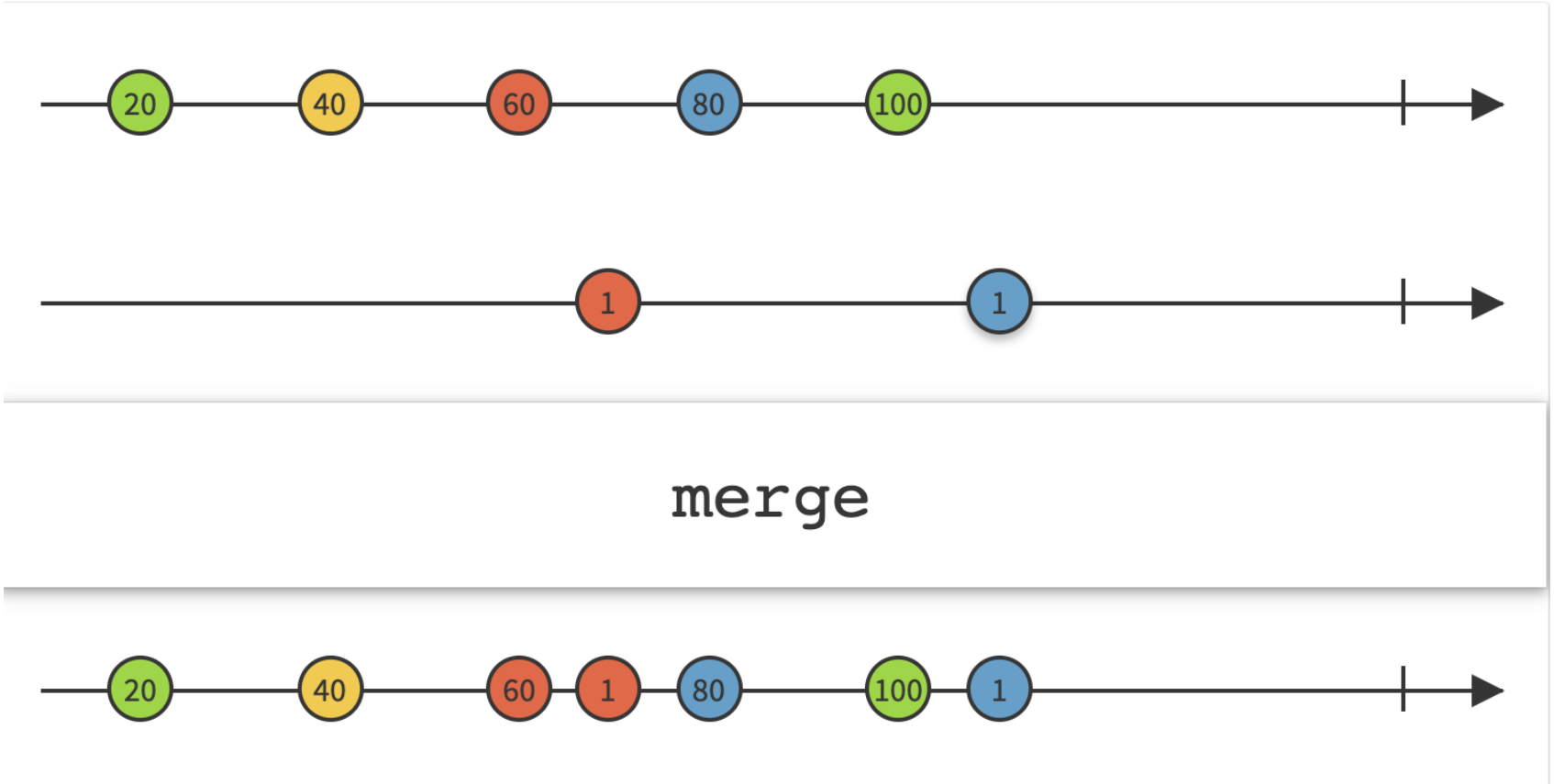

```
map(x => 10 * x)
```


```
filter(x => x > 10)
```

debounce




```
combineLatest((x, y) => "" + x + y)
```


20
40
60
80
100
1
1
merge
20
40
60
1
80
100
1

Problem

Unsubscribe

Memory leak

```
class MyComponent {
  onInit() {
    globalStream$.subscribe(...)
  }
}
```

Solution 1

```
class MyComponent {
  onInit() {
    this.subscription = globalStream$.subscribe(...)
  }

  onDestroy() {
    this.subscription.unsubscribe();
  }
}
```

Solution 2

```
class MyComponent {
  onInit() {
    globalStream$.take(1).subscribe(...)
  }
}
```

Solution 3

```
class MyComponent {

  private unsubscribe$ = new Observable<void>();

  onInit() {
    globalStream$.takeUntil(this.unsubscribe$).subscribe(...)
  }

  onDestroy() {
    this.unsubscribe$.done();
  }
}
```

Solution 4

```
class MyComponent {
  onInit() {
    this.data$ = globalStream$.map(...)
  }

  render() {
    return <div>{{ data$ | async }}</div>
  }

}
```

Problem

Nested subscriptions

Problem

```
this.router.params.subscribe(params => {
  this.http.get(`url/${params}`).subscribe(data => {
    this.data = data;
  })
});
```

Solution

```
this.router.params
  .switchMap(params => this.http.get(`url/${params}`))
  .subscribe(data => {
    this.data = data;
  })
});
```

switchMap

```
import { jsFramework$ } from 'twitter';
import { getDev, recruit } form 'hr';

jsFramework$
  .switchMap(framework => getDev(framework))
  .subscribe(dev => {
    recruit(dev);
  })
```

Problem

Unsubscribe

TOWER DEFENCE

Timmer

```
const ticker$ = interval(17);
```

Enemy factory

```
const enemy$ = ticker$
  .filter(counter => counter % 60 === 0)
  .map(() => enemyFactory())
  .subscribe(enemy => {
    globalEnemies.push(enemy);
  })
```

Enemy move

```
ticker$.subscribe(() => {
  globalEnemies.forEach(enemy => {
    const newDirections = getMove(enemy.position, nextStep, enemy.speed);
    enemy.position.x = newDirections.x;
    enemy.position.y = newDirections.y;
    enemyMove$.next(enemy);
  });
})
```

Tower & Bullet

```
const bulletCreate = enemyMove$
  .filter(enemy => isInRange(this, enemy))
  .map(enemy => BulletFactory(enemy))
  .subscribe(bullet => {
    globalBullets.push(bullet);
});

ticker$.subscribe(() => {
  globalBullets.forEach(bullet => {
    const newDirections = getMove(bullet.position, nextStep,
bullet.speed);
    enemy.position.x = newDirections.x;
    enemy.position.y = newDirections.y;
    bulletMove$.next(bullet)
  });
});
```

Death of enemy

```
const enemyDie$ = bulletMove$
  .filter(bullet => isInRange(bullet, globalEnemies)
  .map(bullet => getFirstEnemy(bullet, globalEnemies));

enemyDie.subscribe(enemy => {
  enemy.die();
  globalEnemies.remove(enemy);
});
```

Why?

Immutable

Modular

Lazy

Declarative

Lazy

Time agnostic

Where?

Angular

Redux-Observable

react-recompose

Vue-rx

Cycle.js

Flutter

Thank you :*