Pluck

Pluck : Like map, but meant only for picking one of the nested properties of every emitted object.

Consider you have a user details like the following and you want to get the city of user. You can get it using map or pluck operator.

src > app > pluck > TS pluck.component.ts > PluckComponent > getNames

```
30    ngAfterViewInit(): void {
31      fromEvent<any>(this.txt.nativeElement, 'input')
32      .pipe(
33        // map(event => event.target.value),
34        pluck('target', 'value')
35      )
36      .subscribe((val) => console.log(val));
37    }
38
39    getNames(users) {
40      //using map operator
41      // from(users).pipe(
42      //   map(user => user['name']),
43      //   toArray()
44      // ).subscribe(names => {
45      //   console.log('names => ', names);
46      // });
47      //using pluck operator
48      from(users).pipe(
49        pluck('name'),
50        toArray()
51      ).subscribe(names => {
52        console.log('names => ', names);
53      });
54    }
55
56    getCities(users) {
57      from(users).pipe(
58        pluck('address', 'city'),
59        // map(val => val['address']['city']),
60        toArray()
61      ).subscribe(cities => {
```

```
this.http.get('https://jsonplaceholder.typicode.com/users').pipe(
).subscribe((users) => {
console.log('users => ', users);
//users[0].address = null;
this.getNames(users);
this.getCities(users);
});
```

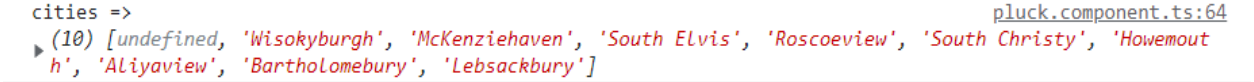
Using map

```
getCities(users) {
from(users).pipe(
map(val => val['address']['city']),
toArray()
).subscribe(cities => {
console.log('cities => ', cities);
});
}
```

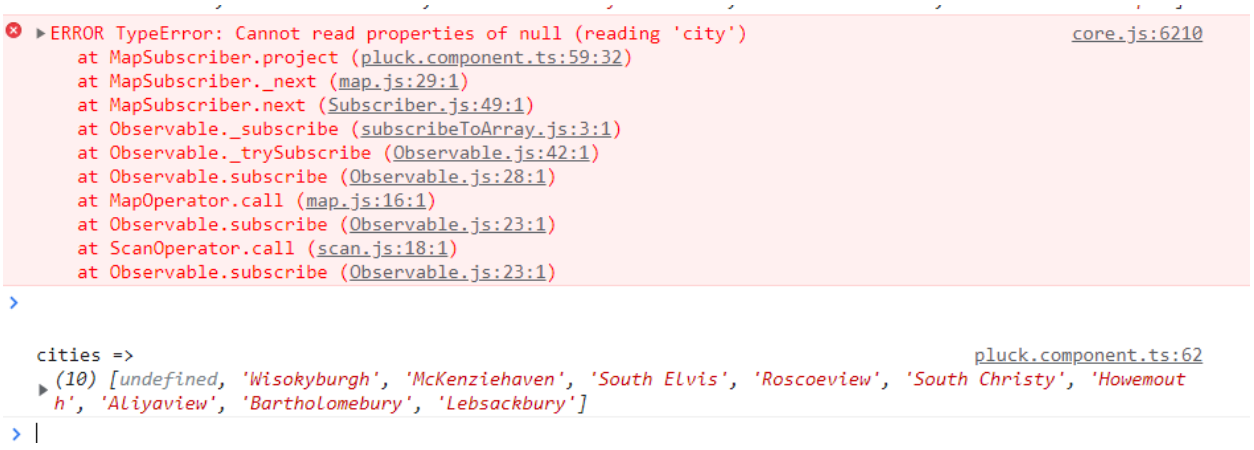
Using pluck

```
getCities(users) {
from(users).pipe(
pluck('address', 'city'),
toArray()
).subscribe(cities => {
console.log('cities => ', cities);
});
}
```

Both will give same result like the following image



Now if, user's address value will be null, then map operator will give error but pluck will not give any error, just print "undefined" in subscription value.



Another example of pluck operator

```
fromEvent<any>(this.txt.nativeElement, 'input')
.pipe(
map(event => event.target.value)
)
.subscribe((val) => console.log(val));
```

Instead of map operator, you can use pluck operator

```
fromEvent<any>(this.txt.nativeElement, 'input')
.pipe(
pluck('target', 'value')
)
.subscribe((val) => console.log(val));
```