FullStack #WebDevelopment Bootcamp

Course Week 7

Summary – Why Angular

Latest web tools and patterns

Super fast

Huge community

Enterprise friendly

Asynchronous Programming

Angular itself is easy

… but it forces you to confront modern web development.

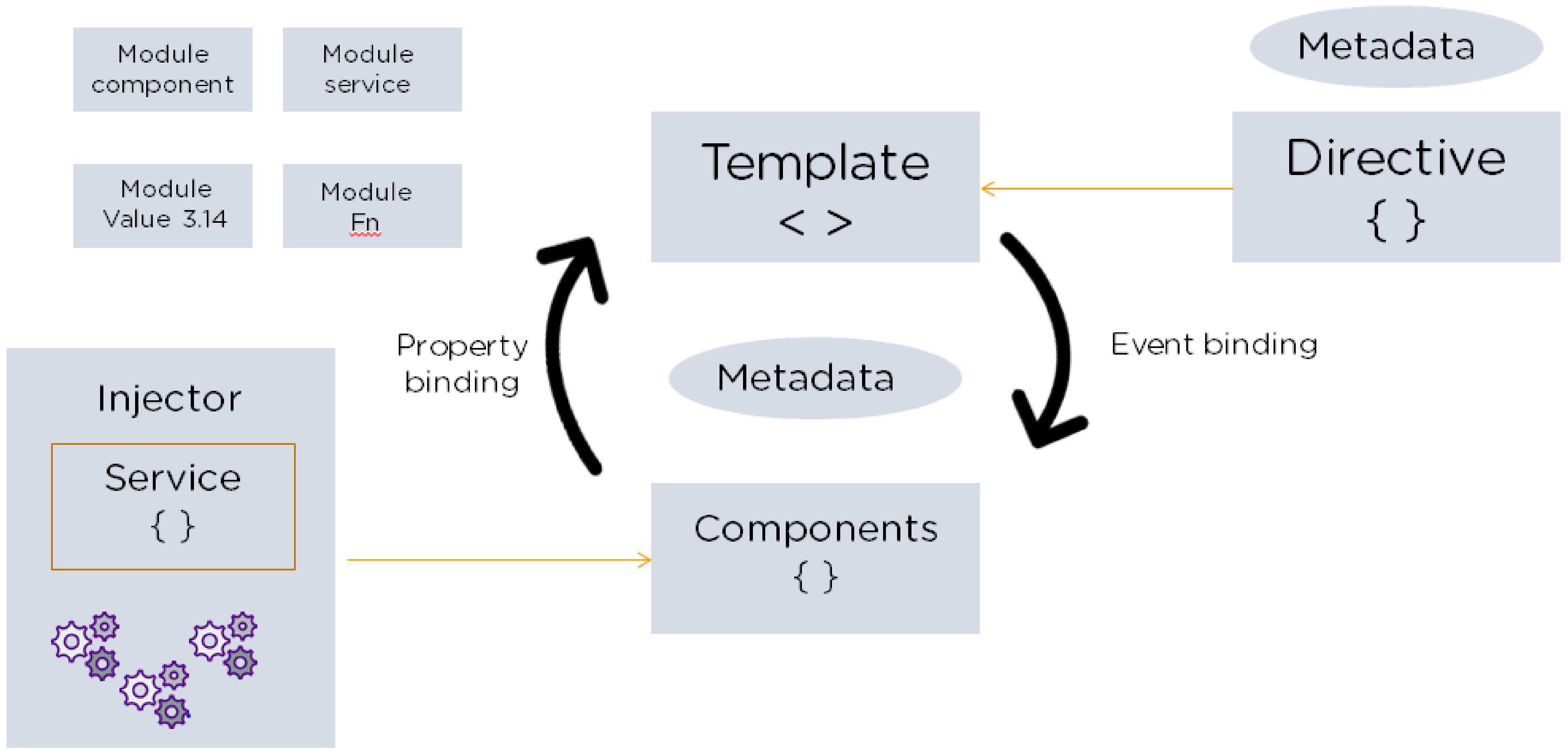
TypeScript

RxJS

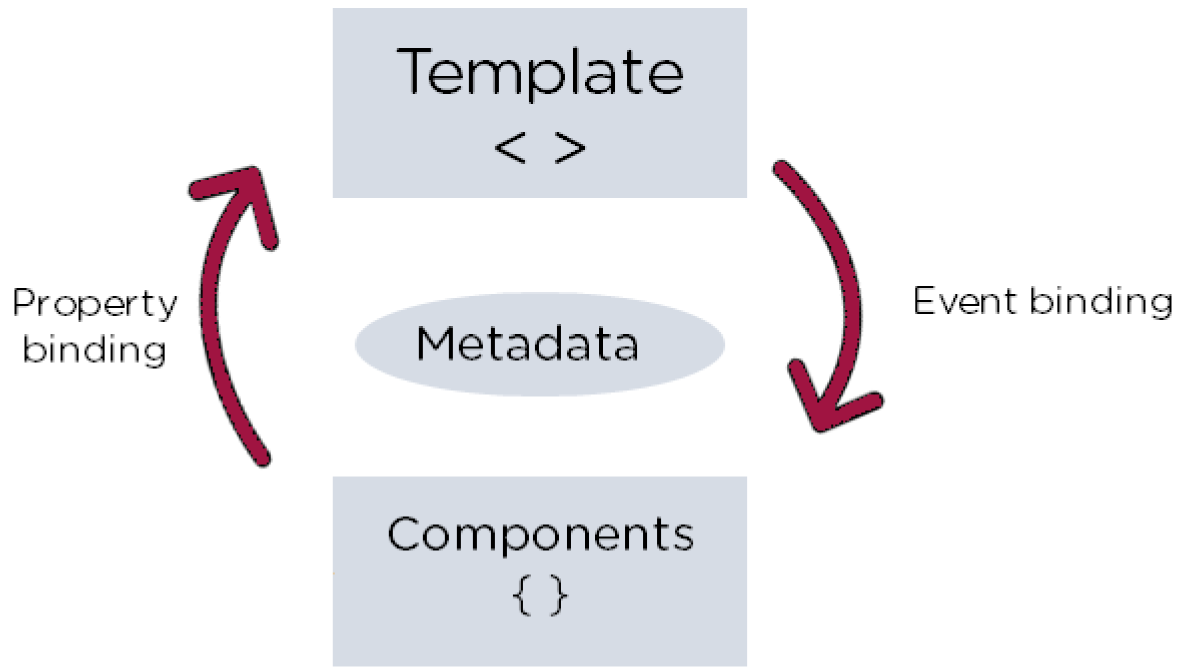
Modules

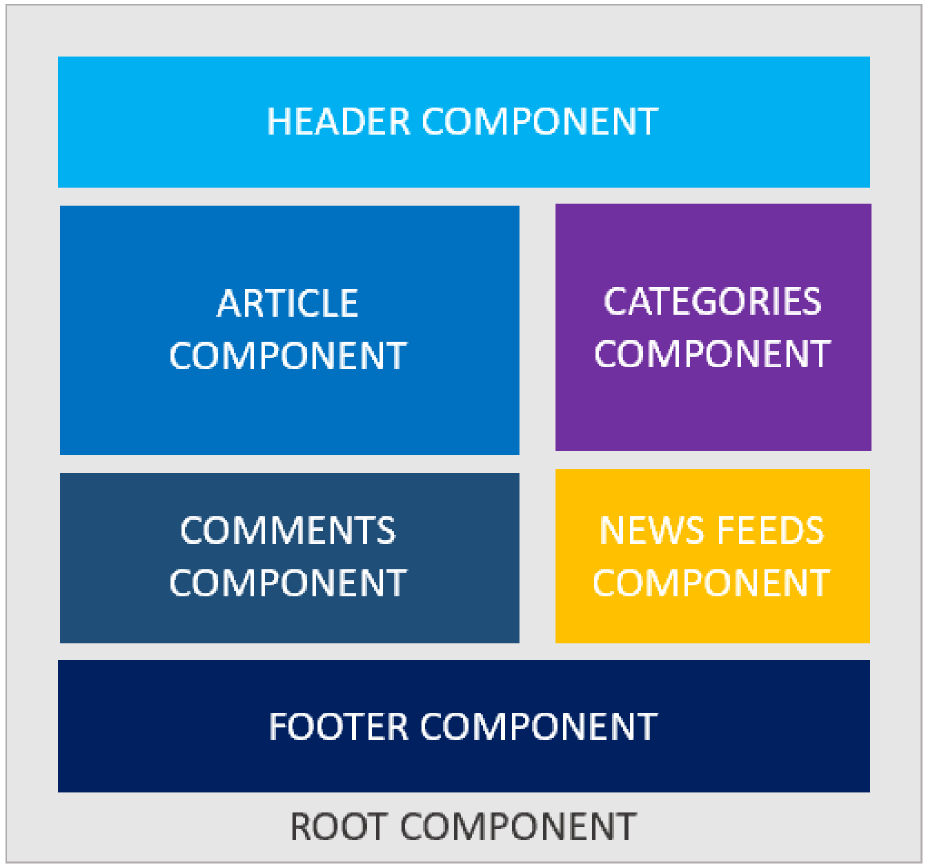
Redux

Angular Architecture Overview

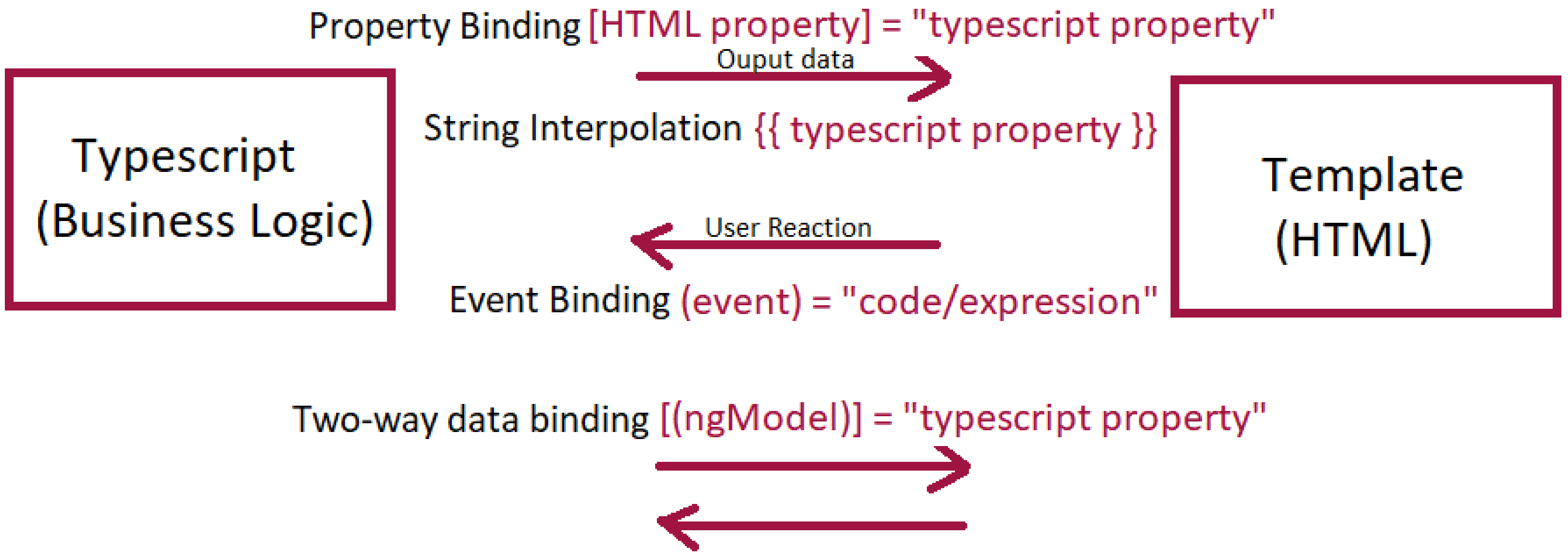


Angular - Component-Based Architecture





Angular - Component-Based Architecture



Property Binding [HTML property] = "typescript property"
Ouput data
Typescript (Business Logic)
String Interpolation {{ typescript property }}
Template (HTML)
User Reaction
Event Binding (event) = "code/expression"
Two-way data binding [(ngModel)] = "typescript property"

Data Binding = **Communication**

Angular - Component-Based Architecture

Import

```
import { Component, OnInit } from '@angular/core';
```

Metadata

```
@Component({
  selector: 'hello',
  templateUrl: `./hello.component.html`,
  styleUrls: [`./hello.component.scss`],
})
export class HelloComponent implements OnInit {
```

Component Class

```
  public title: string;

  public ngOnInit(): void {
    this.title = 'Hello!';
  }

  public changeTheTItle(): void {
    this.title = 'new title';
  }
}
```

```
<div>
  <h1>{{ title }}</h1>
  //...
  <button (click)="changeTheTItle()">Change</button>
</div>
```

Presentational and Container Components

Presentational	Container
Concerned with how things look	Concerned with how things work
HTML markup and CSS styles	Have little to no HTML and CSS styles
No dependencies on the rest of the app	Have injected dependencies
Don’t specify how data is loaded or changed but emit events via @Outputs	Are stateful and specify how data is loaded or changed
Receive data via @Inputs	Top level routes
May contain other components	May contain other components

Benefits

Performance

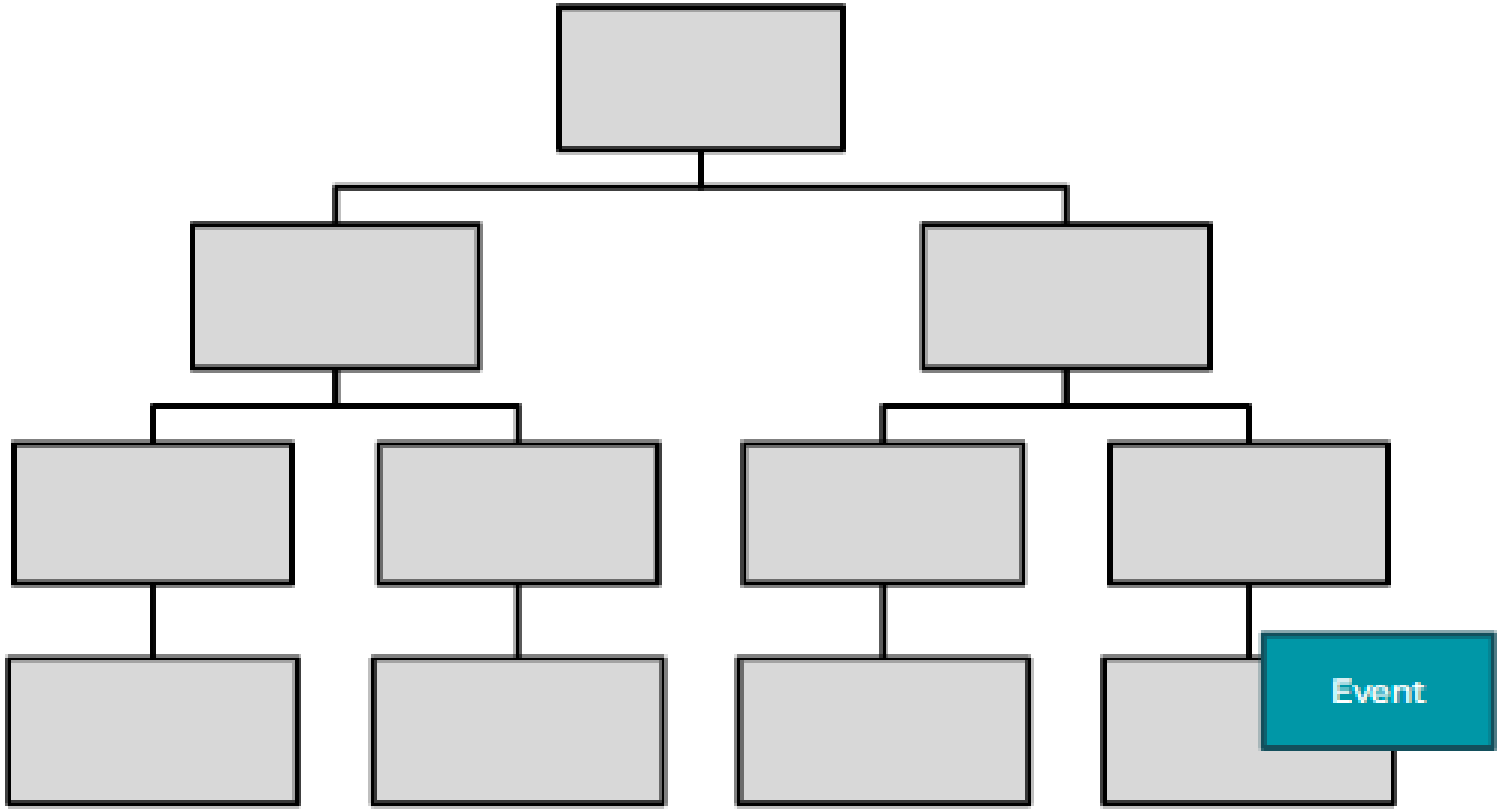
Composability

Easier to test

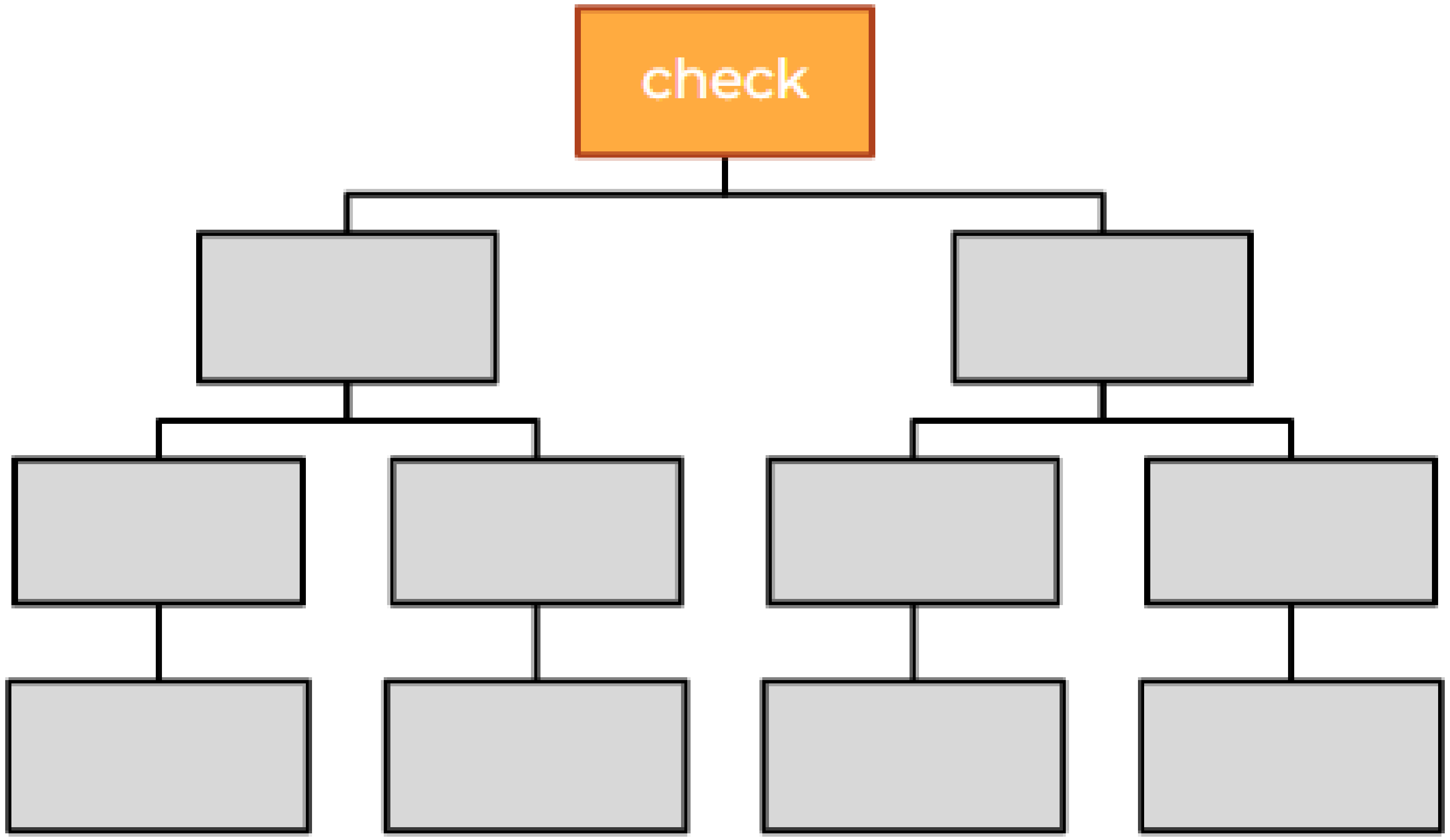
ChangeDetectionStrategy.OnPush

ChangeDetectionStrategy.Default

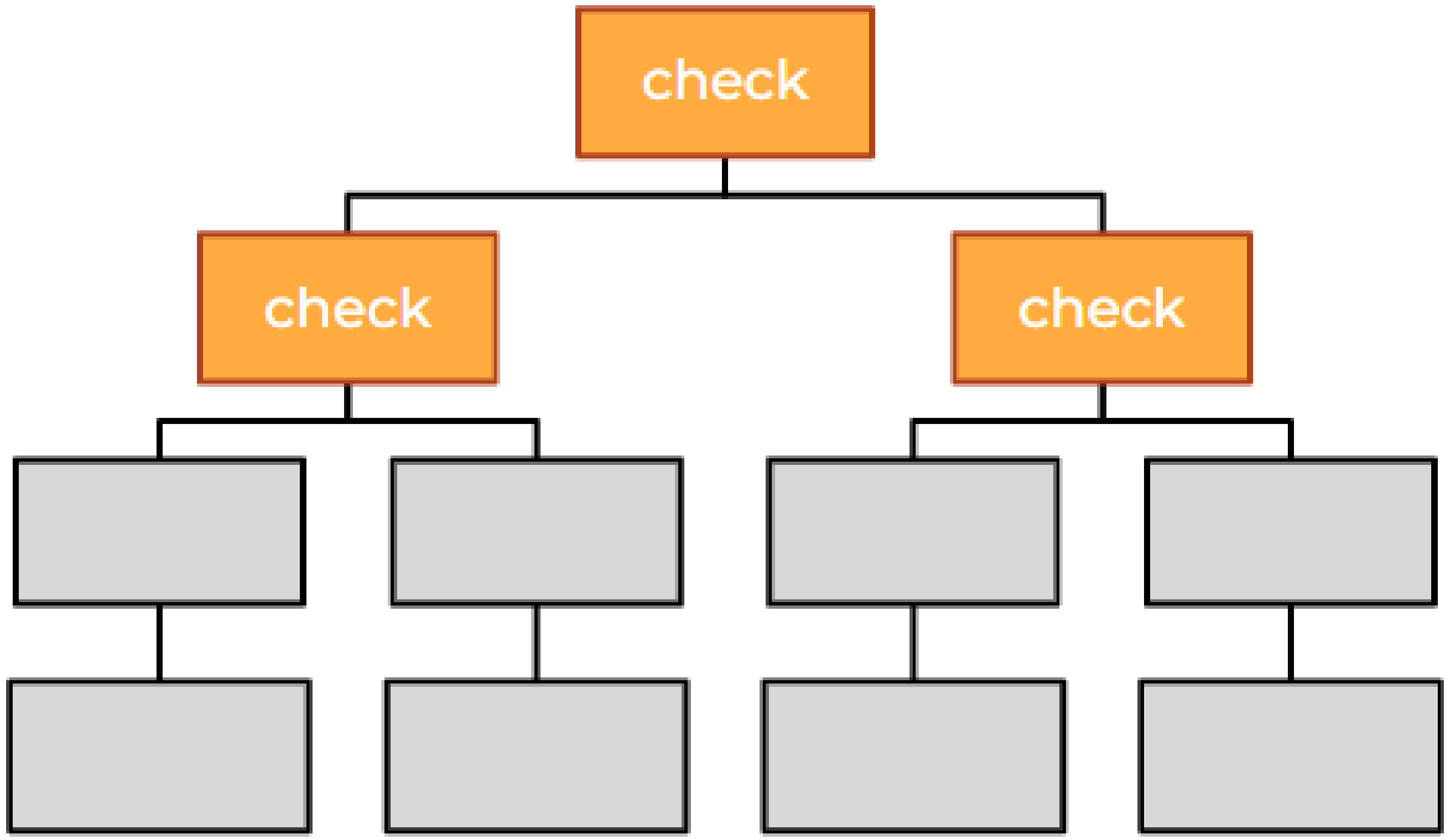
ChangeDetectionStrategy.Default



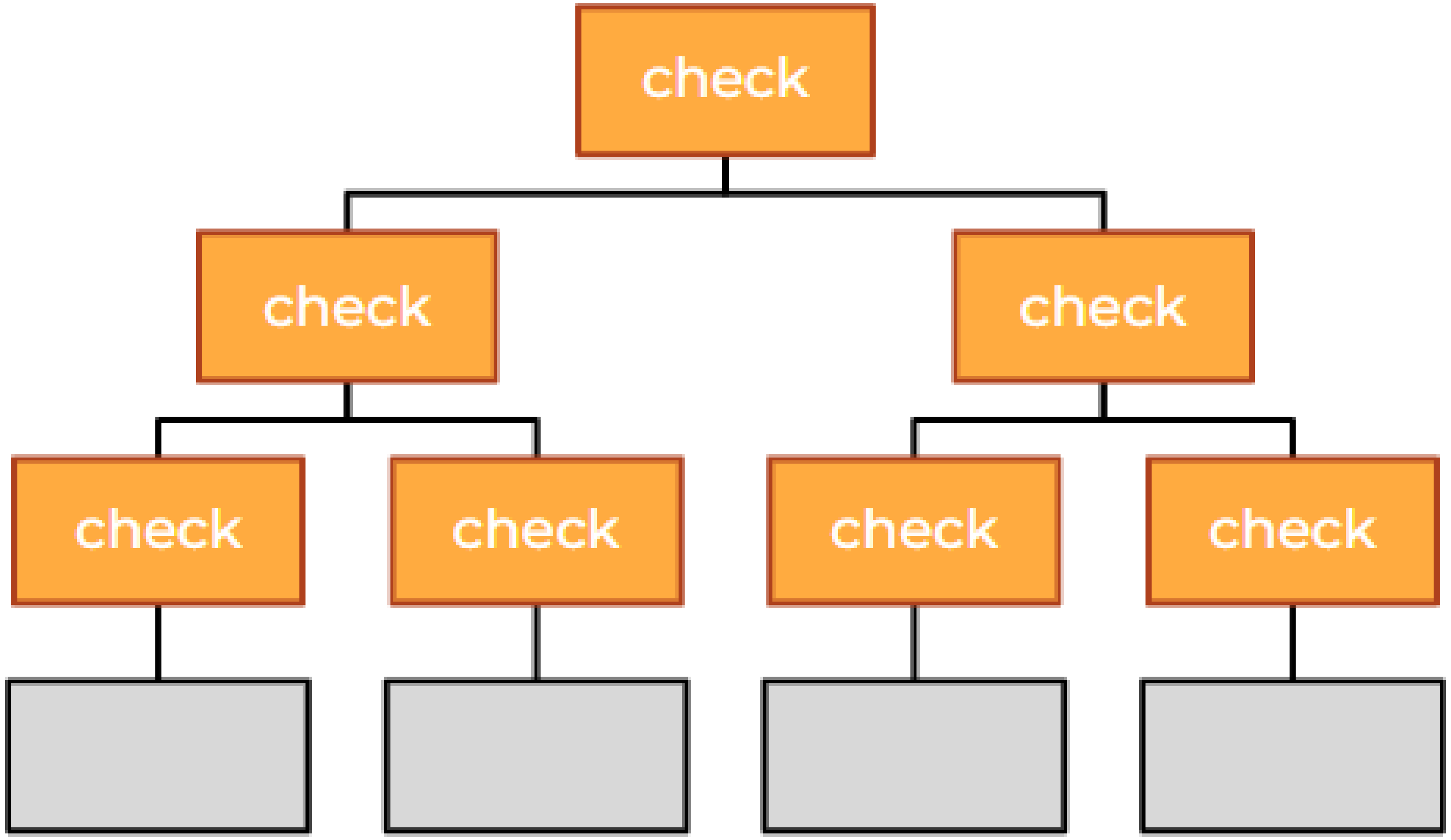
ChangeDetectionStrategy.Default



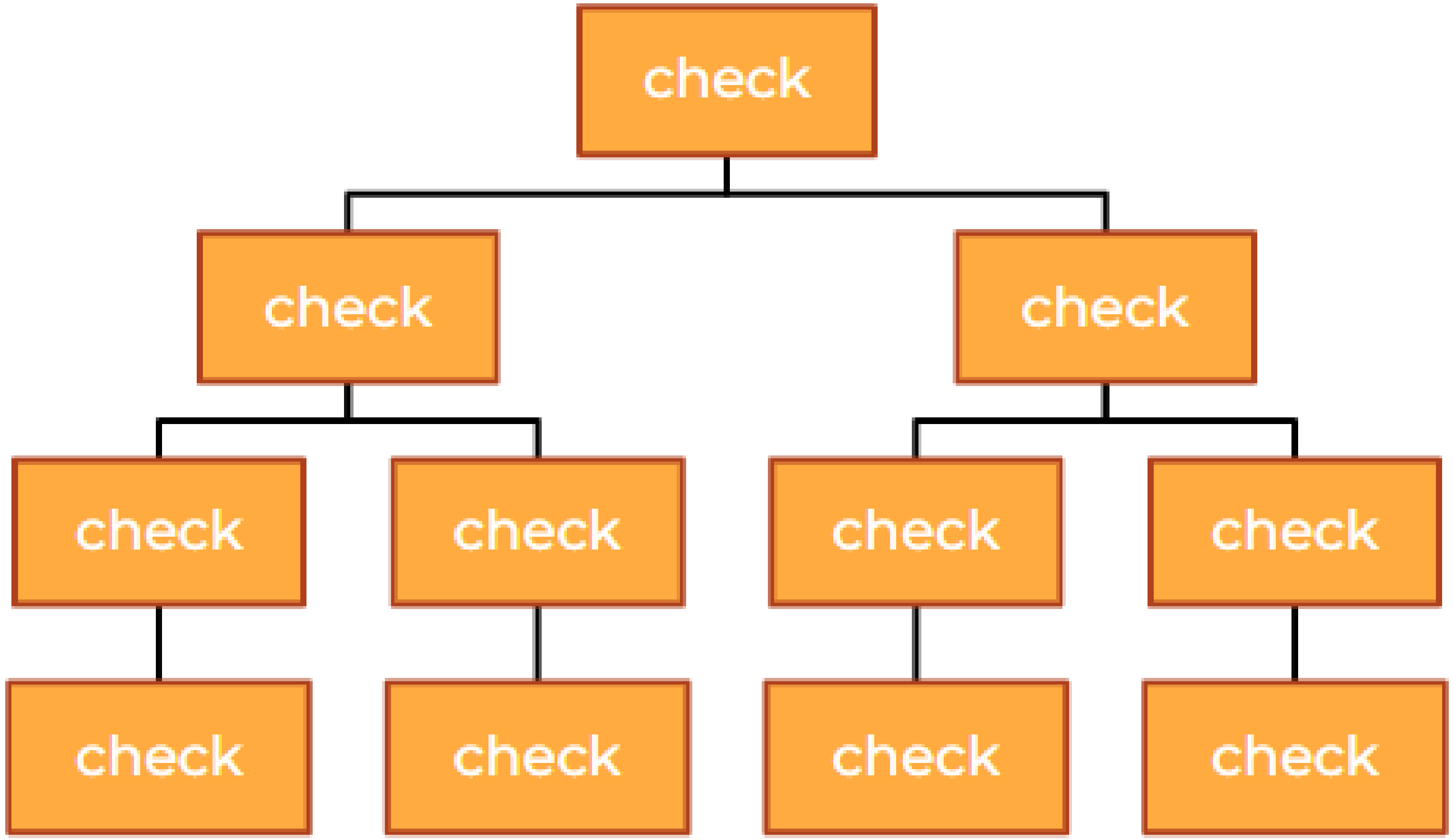
ChangeDetectionStrategy.Default



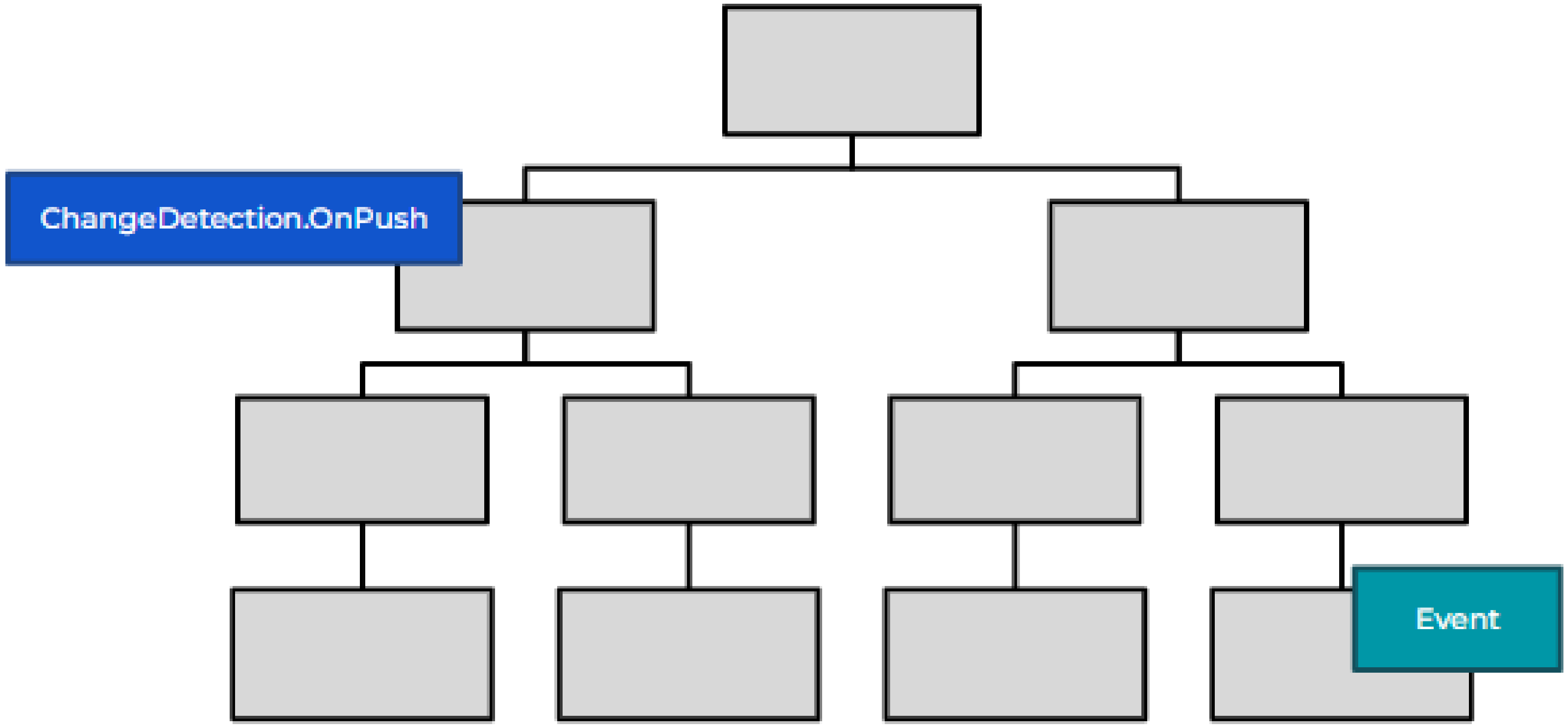
ChangeDetectionStrategy.Default



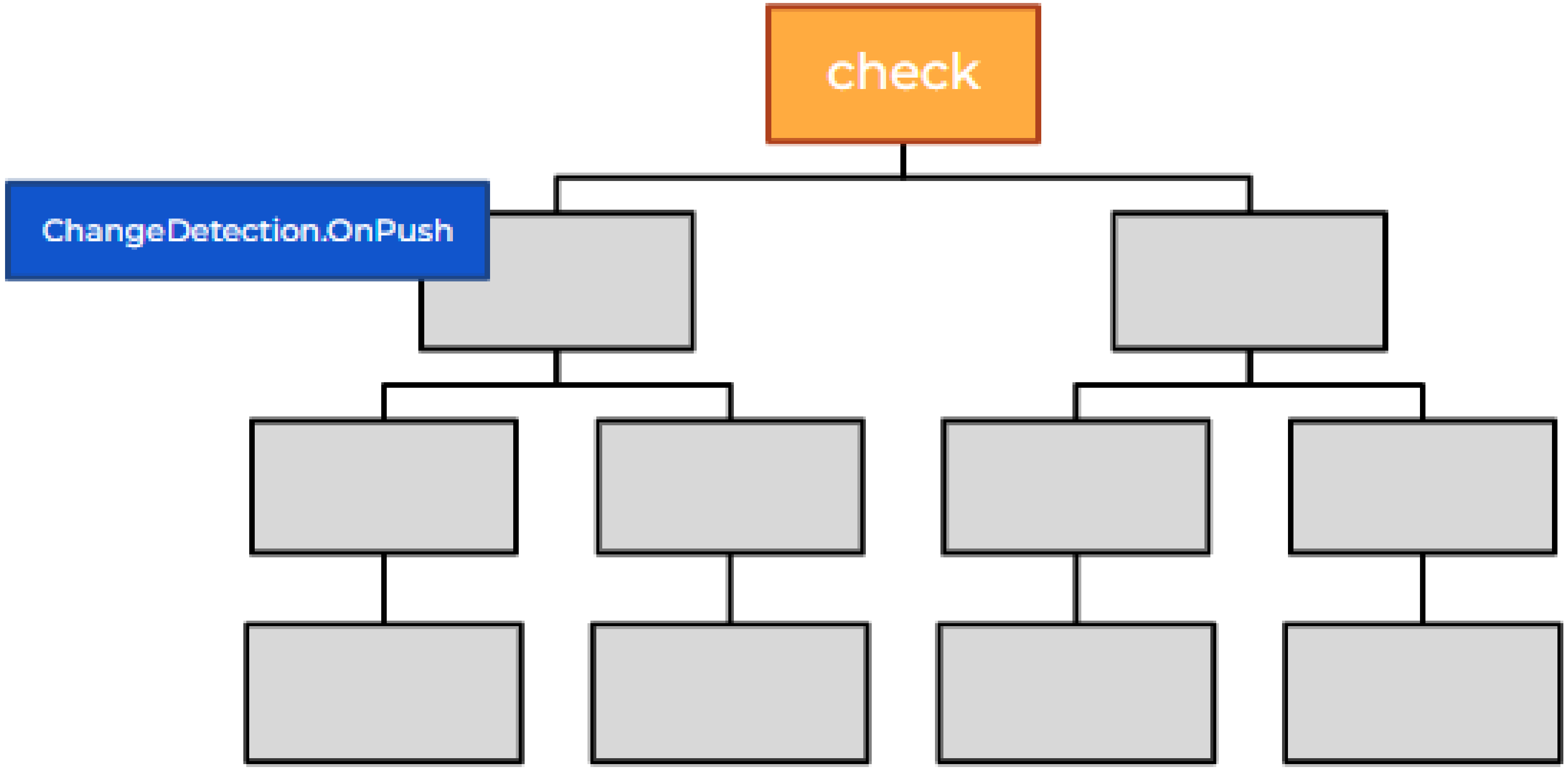
ChangeDetectionStrategy.Default



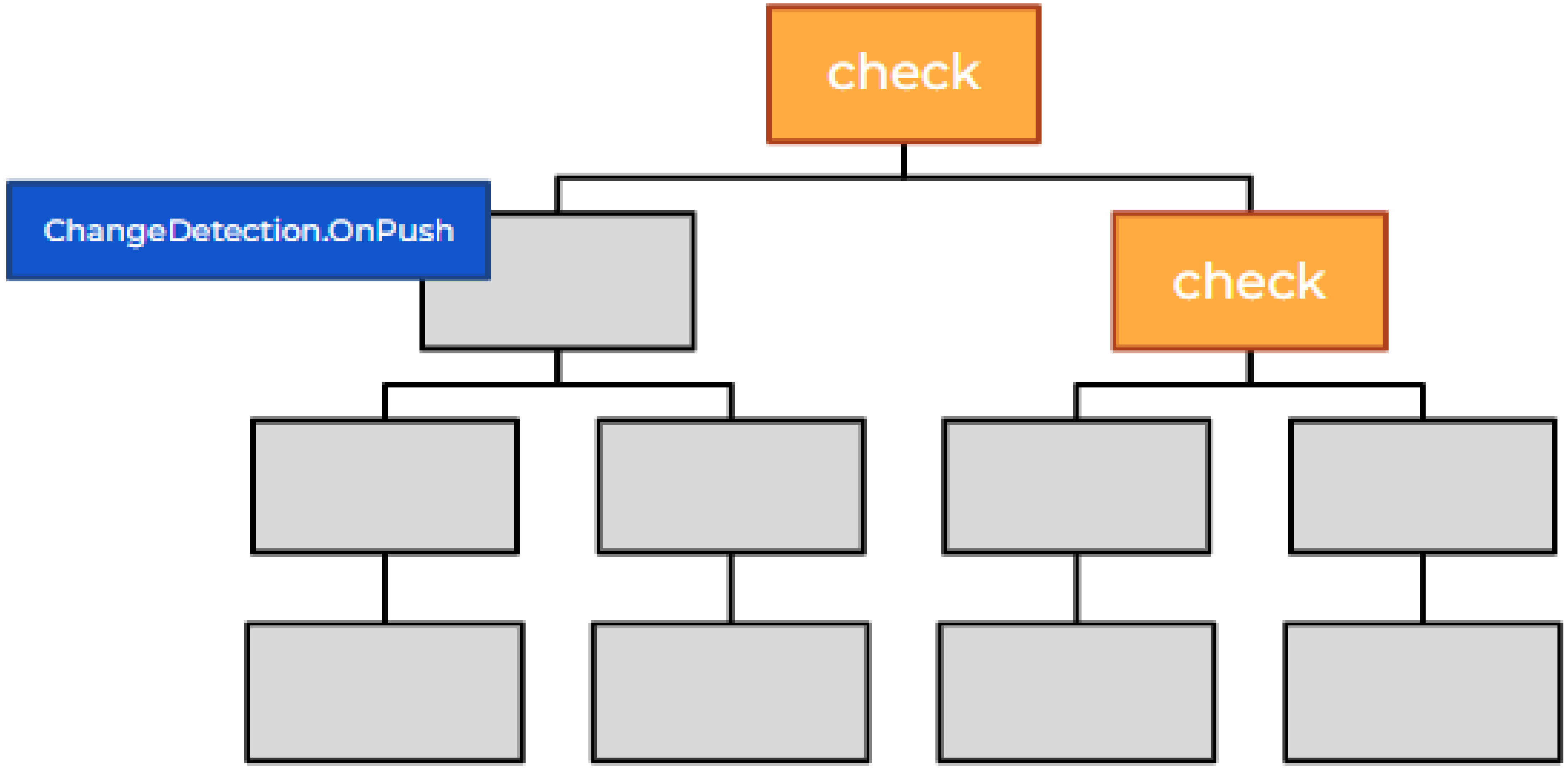
ChangeDetectionStrategy.OnPush



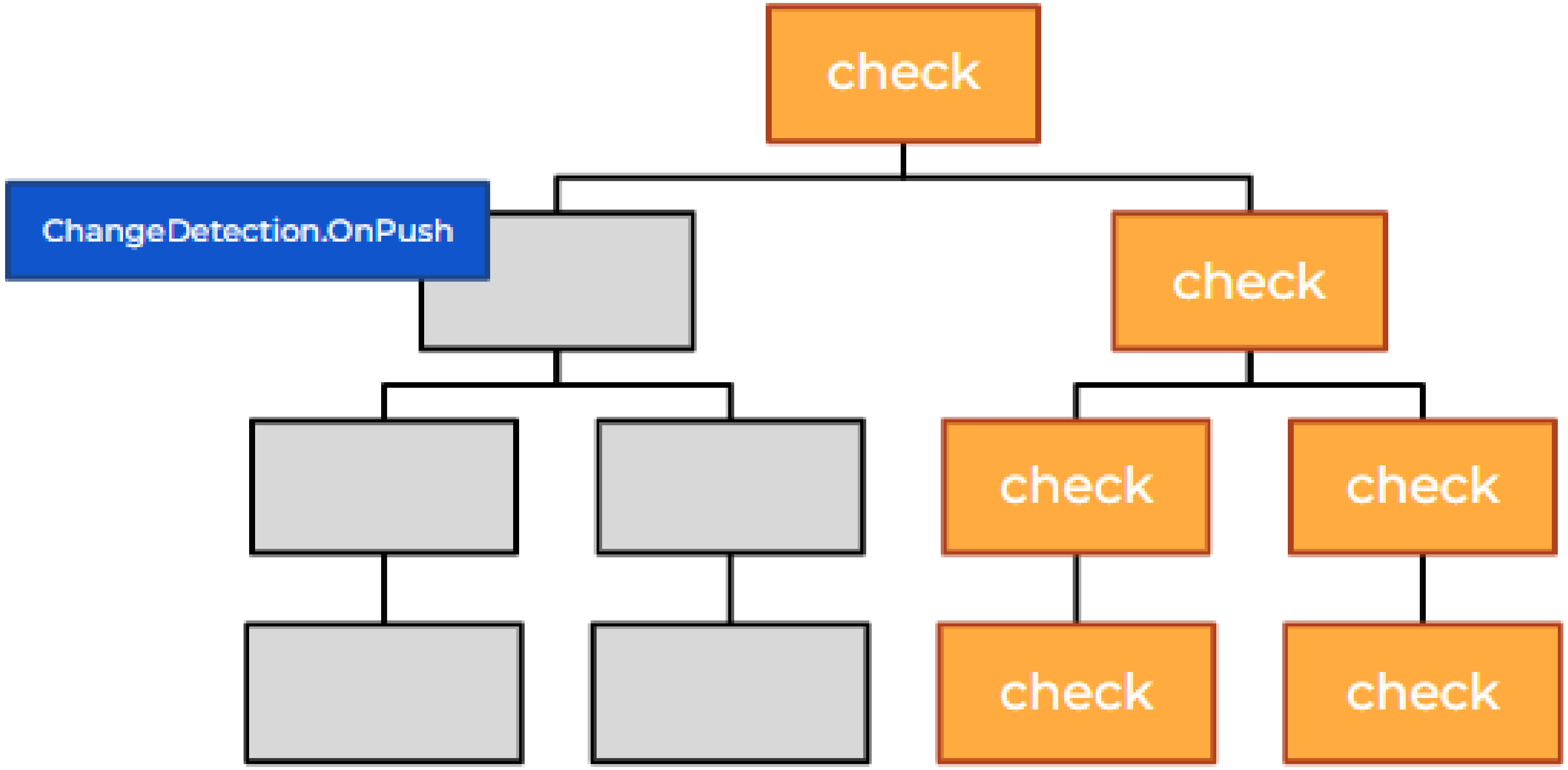
ChangeDetectionStrategy.OnPush

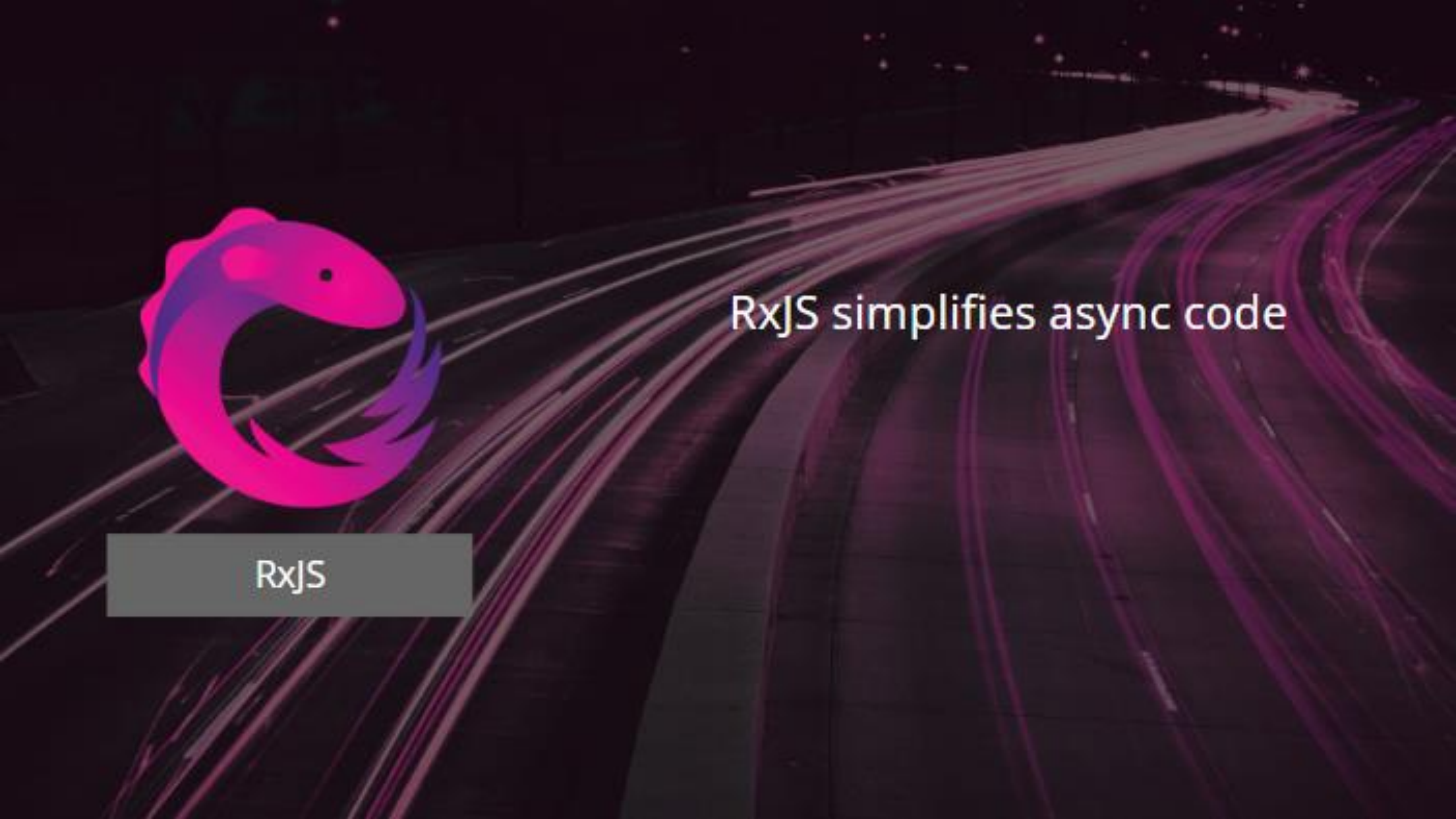


ChangeDetectionStrategy.OnPush



ChangeDetectionStrategy.OnPush



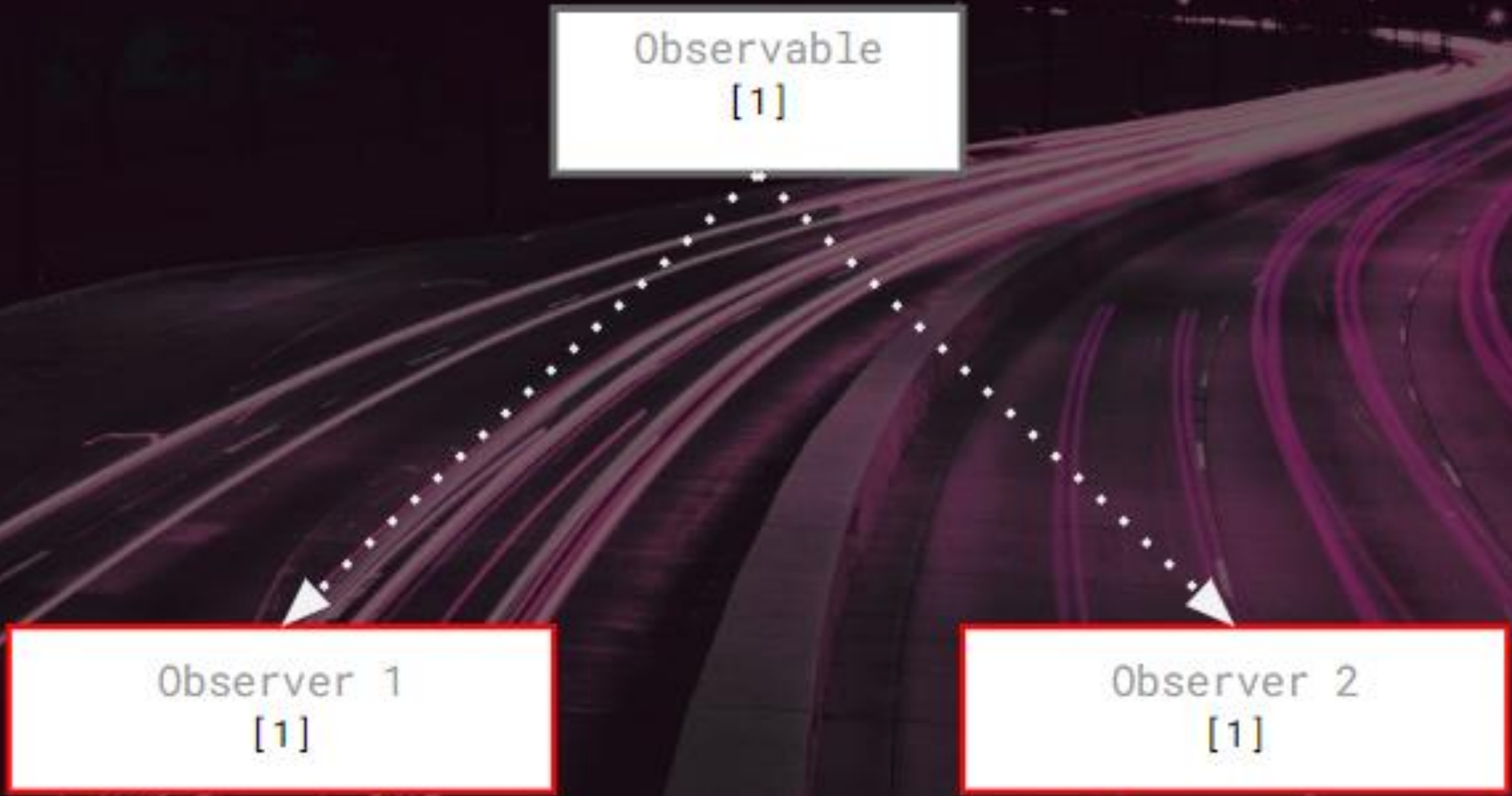


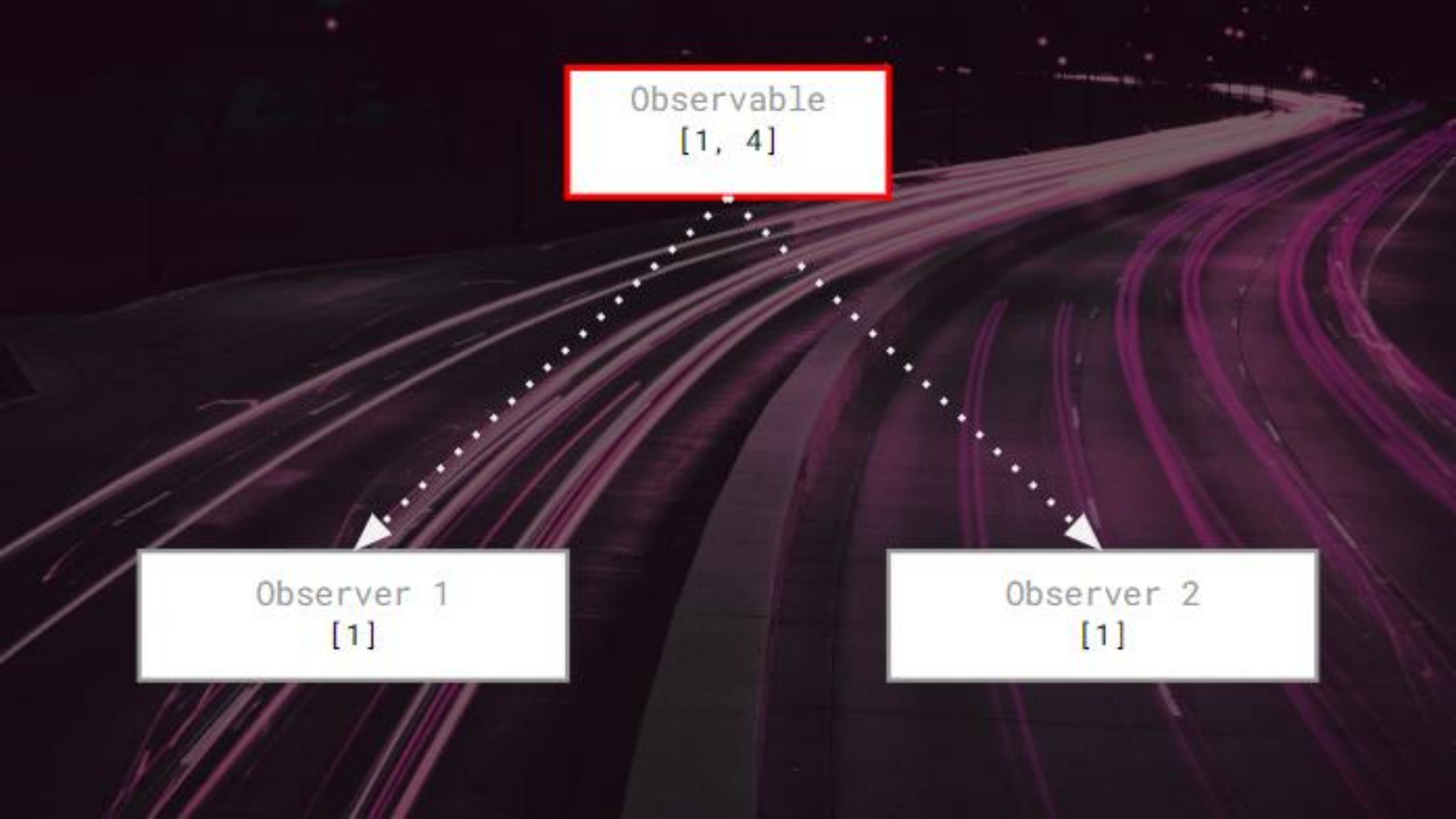
RxJS simplifies async code

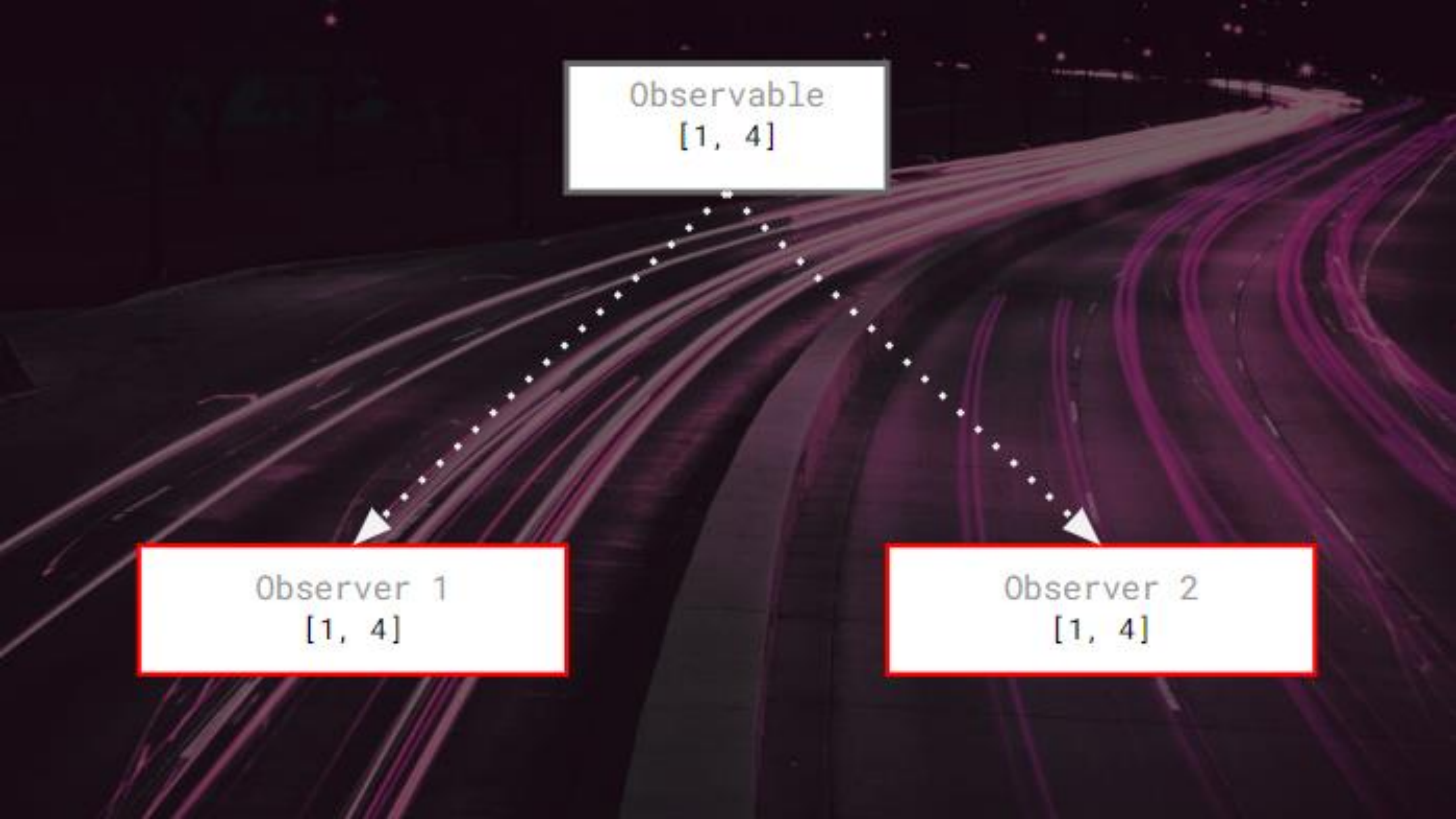
RxJS

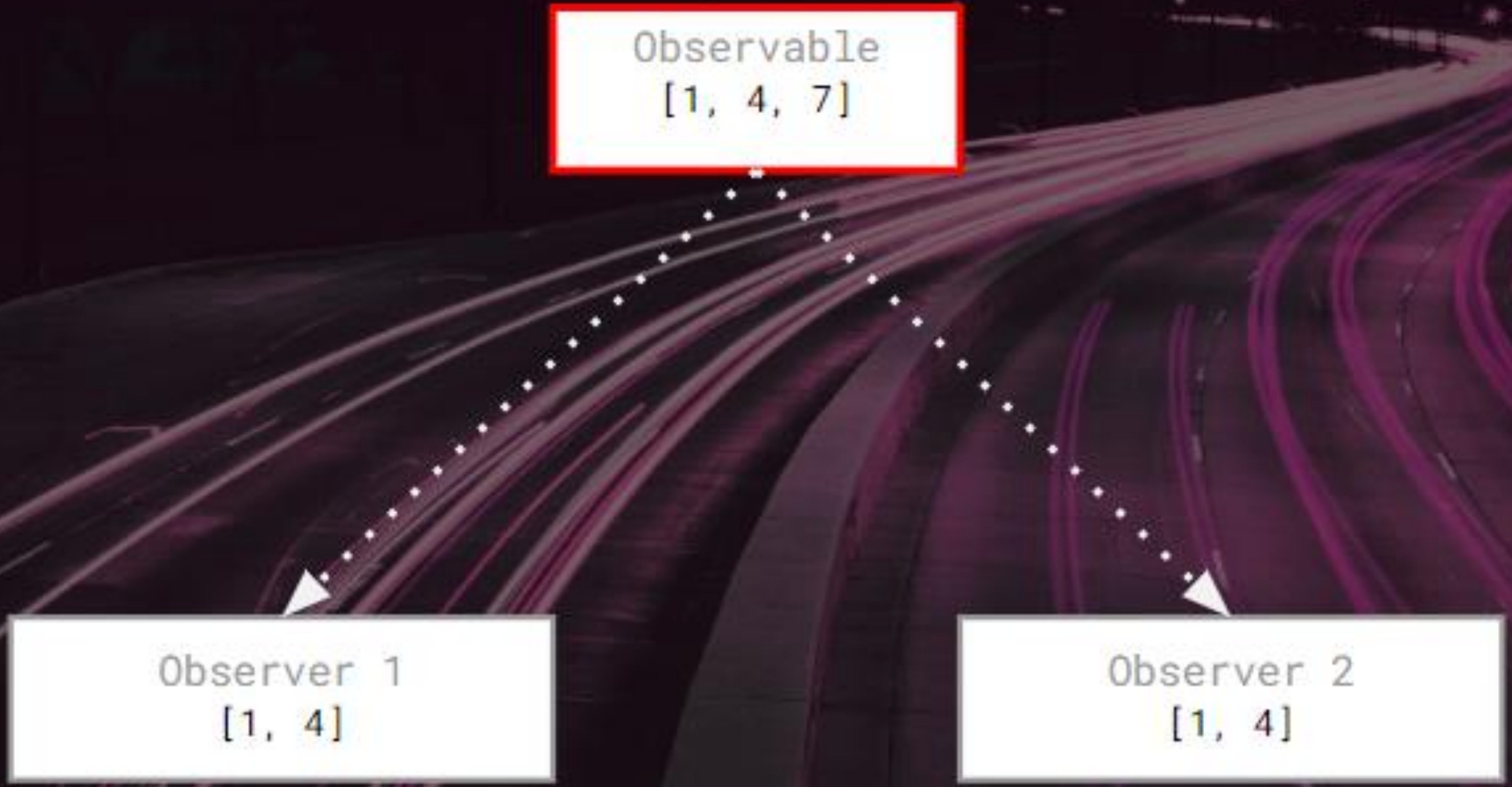
RxJS

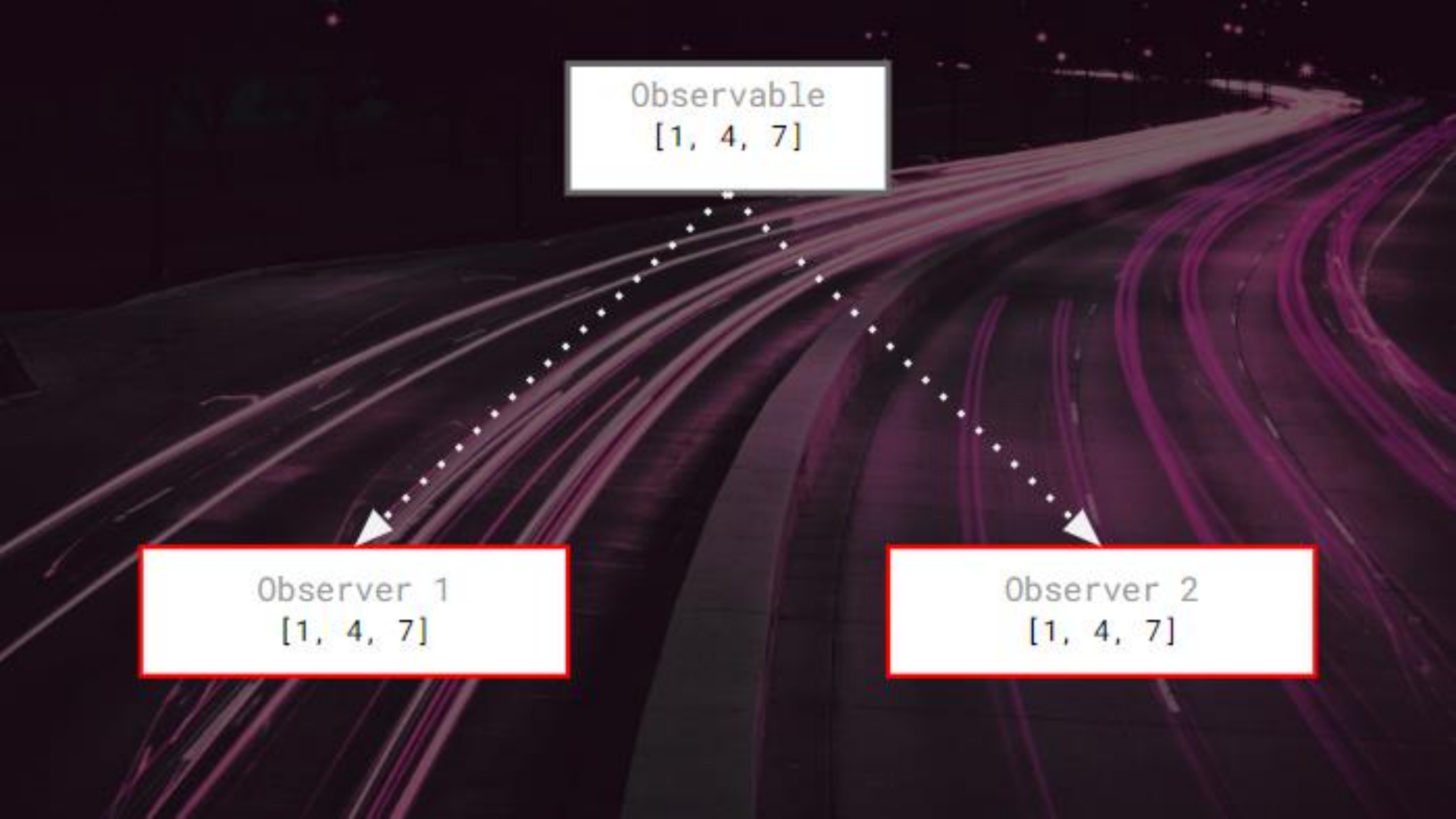
An API for asynchronous programming with observable streams

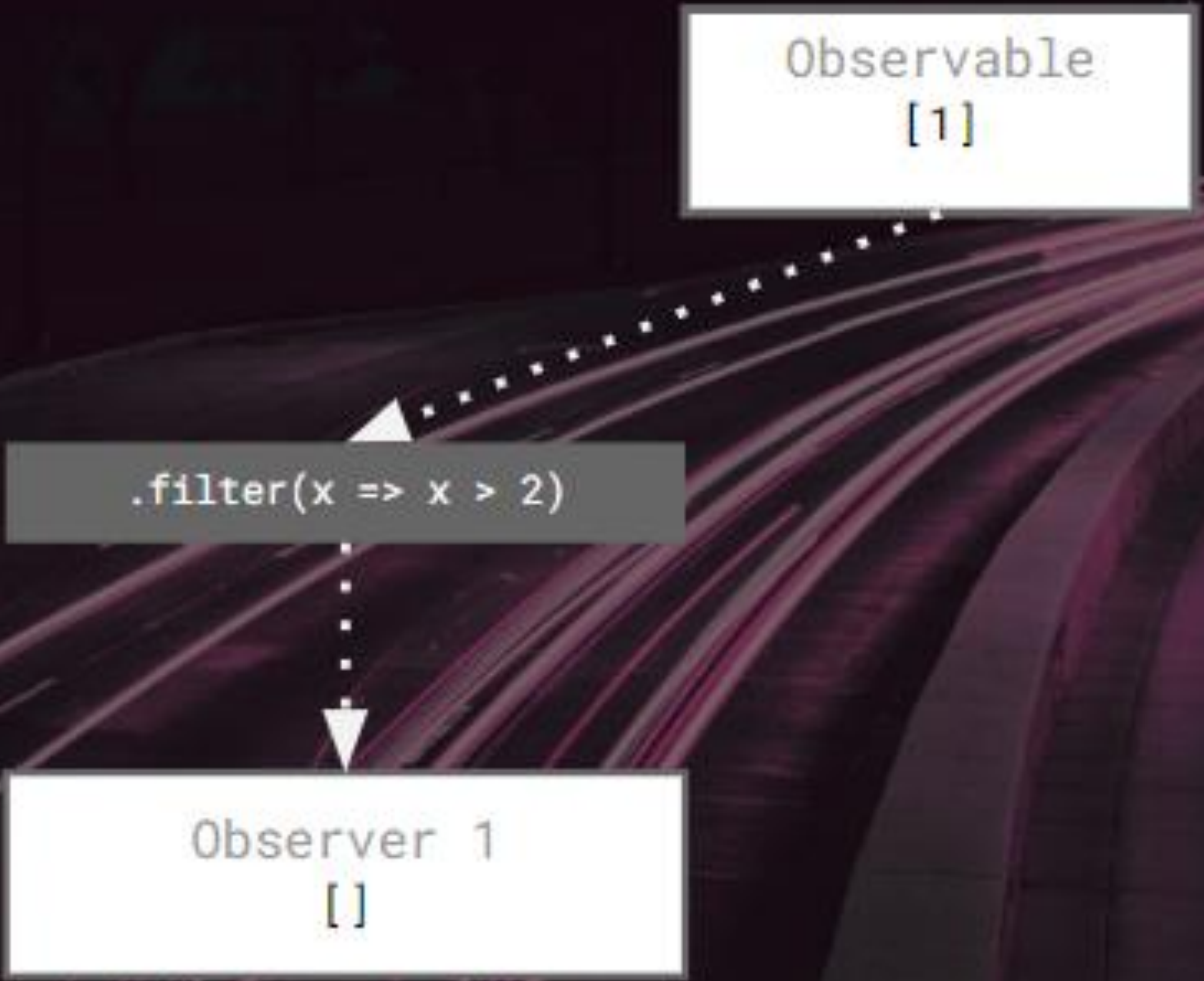

Observable
[1]
Observer 1
[1]
Observer 2
[1]


Observable
[1, 4]
Observer 1
[1]
Observer 2
[1]

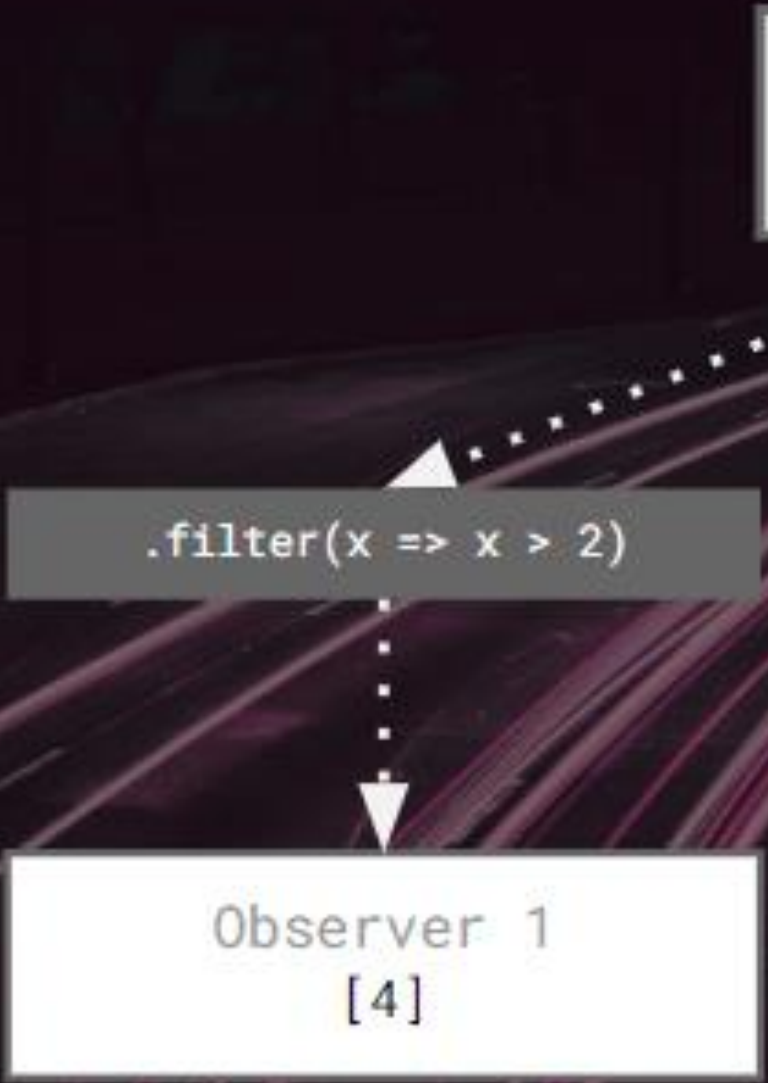

Observable
[1, 4]
Observer 1
[1, 4]
Observer 2
[1, 4]


Observable
[1, 4, 7]
Observer 1
[1, 4]
Observer 2
[1, 4]

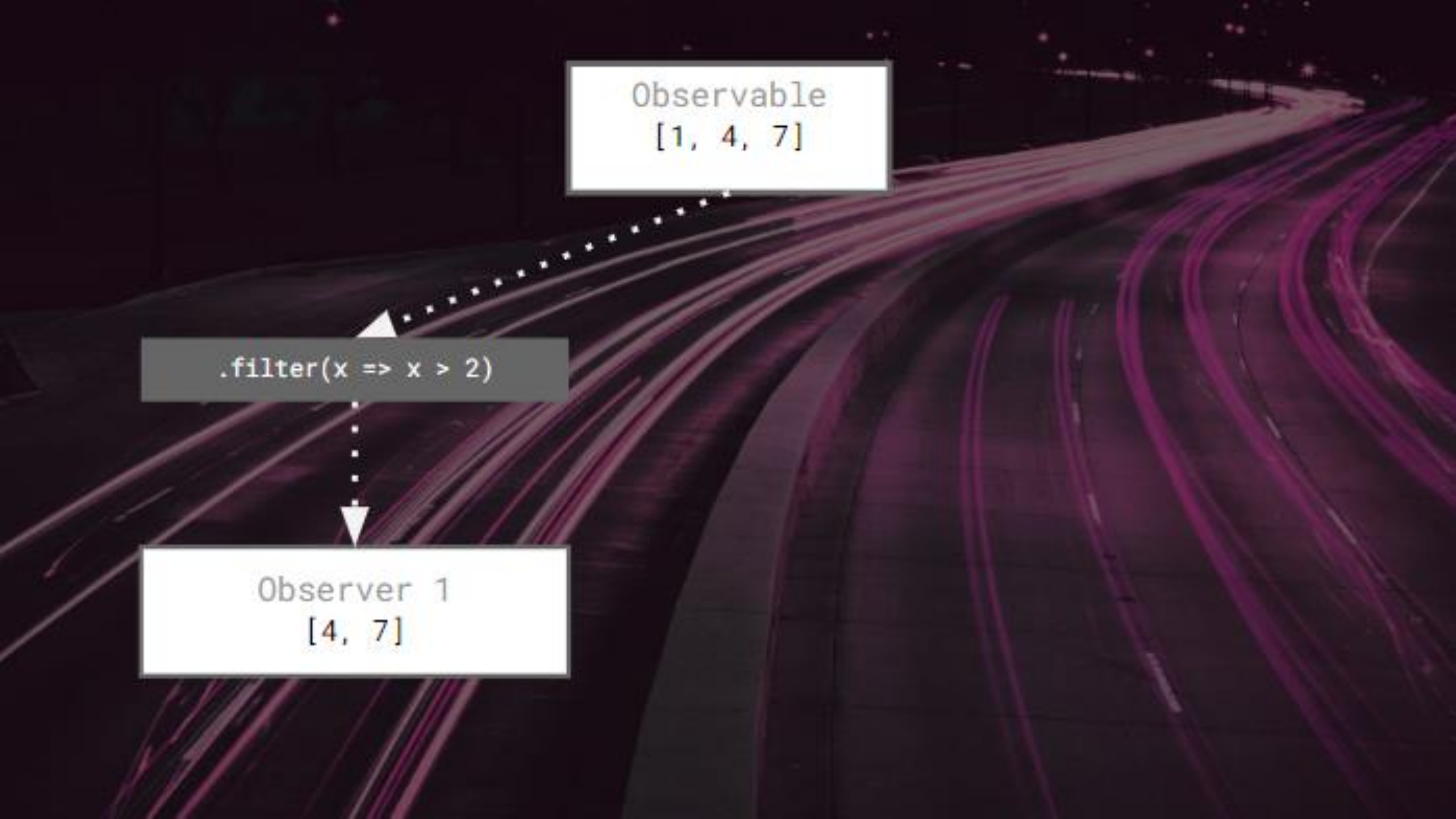

Observable
[1, 4, 7]
Observer 1
[1, 4, 7]
Observer 2
[1, 4, 7]



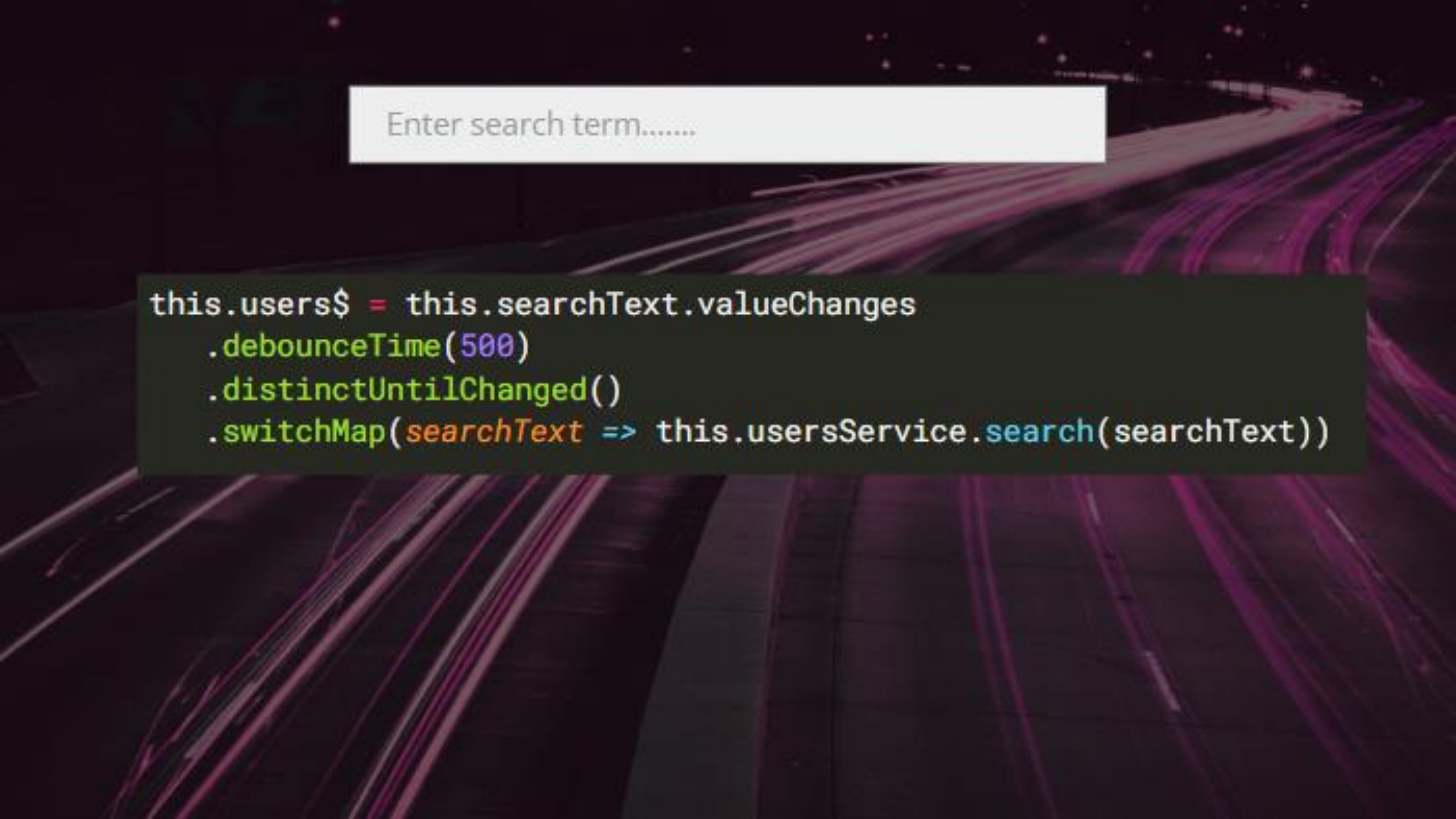
```
.filter(x => x > 2)
```



```
.filter(x => x > 2)
```


Observable
[1, 4, 7]
.filter(x => x > 2)
Observer 1
[4, 7]

RxJS is a journey…



Enter search term.......

```
this.users$ = this.searchText.valueChanges
   .debounceTime(500)
   .distinctUntilChanged()
   .switchMap(searchText => this.usersService.search(searchText))
```

```
@Injectable()
export class PeopleService {
  constructor(private httpClient: HttpClient) { }

   getPeople() {
      return this.httpClient.get(`http://swapi.co/api/people/`)
       .subscribe(people => console.log(people));
   }
}
```

```
@Component()
export class SearchComponent implements OnInit {
  searchControl = new FormControl();
  ngOnInit() {
    this.searchControl.valueChanges.subscribe(value => {
      // do something with value here
    });
  }
}
```

```
ngOnInit() {
   this.route.params
    .subscribe(params => this.id = params['id'])
  }
```

```
ngOnInit() {
    this.route.params.pipe(
      map(params => params['id']),
      switchMap(id => this.contactsService.getContact(id)),
     )
     .subscribe(contact => this.contact = contact);
  }
```

Summary - RxJS

Modern web development is asynchronous

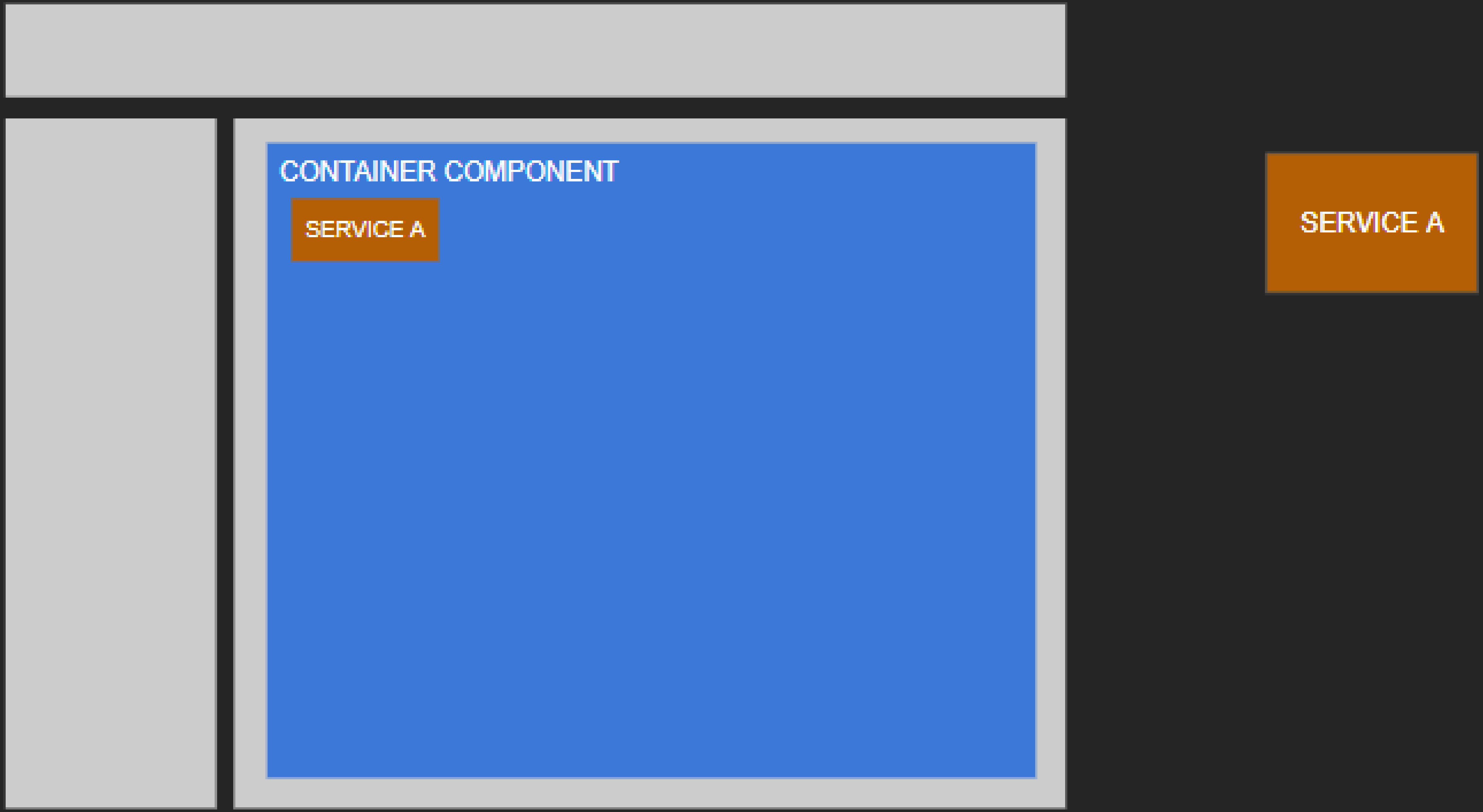
RxJS takes some effort to fully understand, but it simplifies writing async code

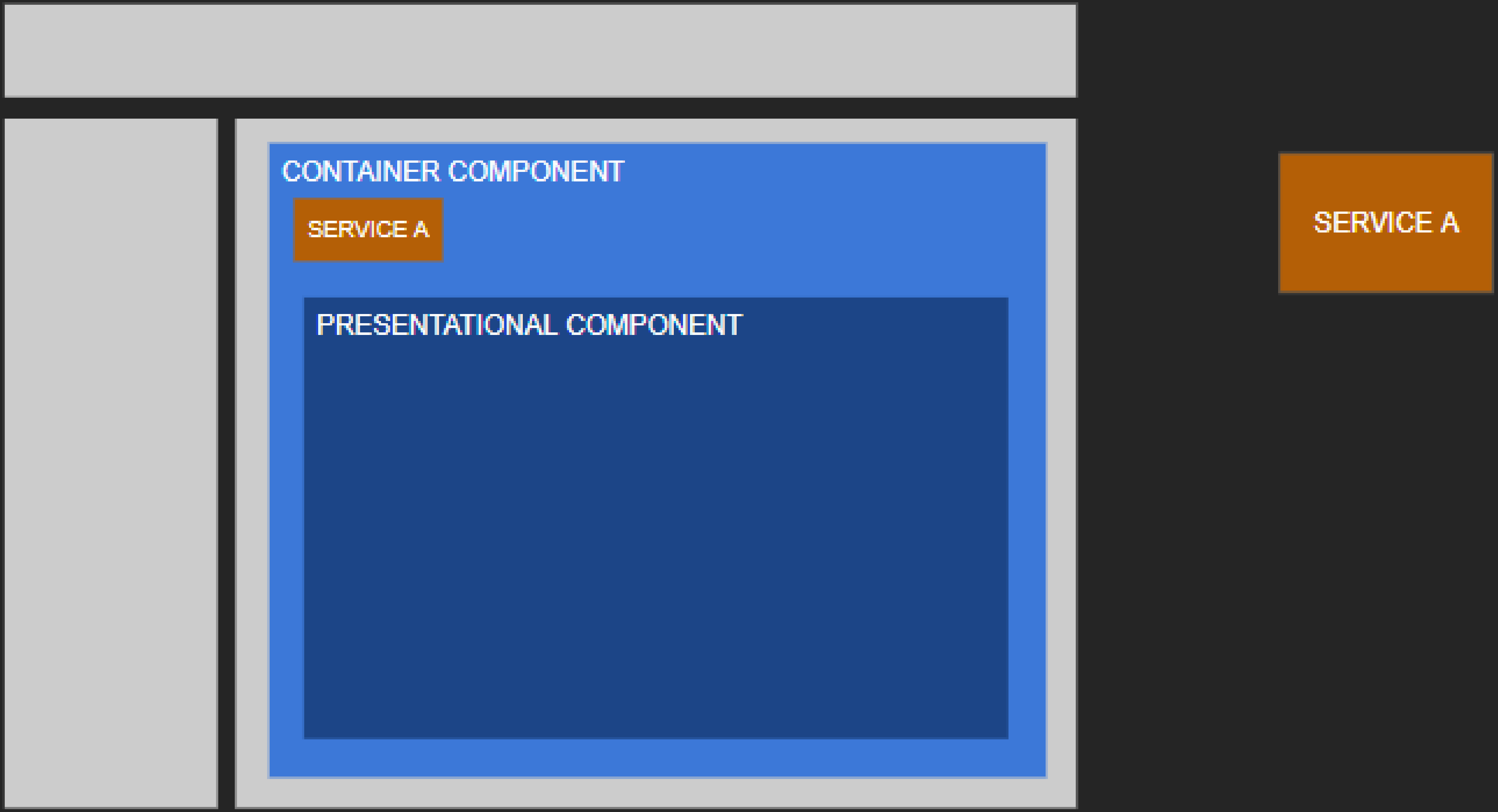
RxJS is baked into Angular

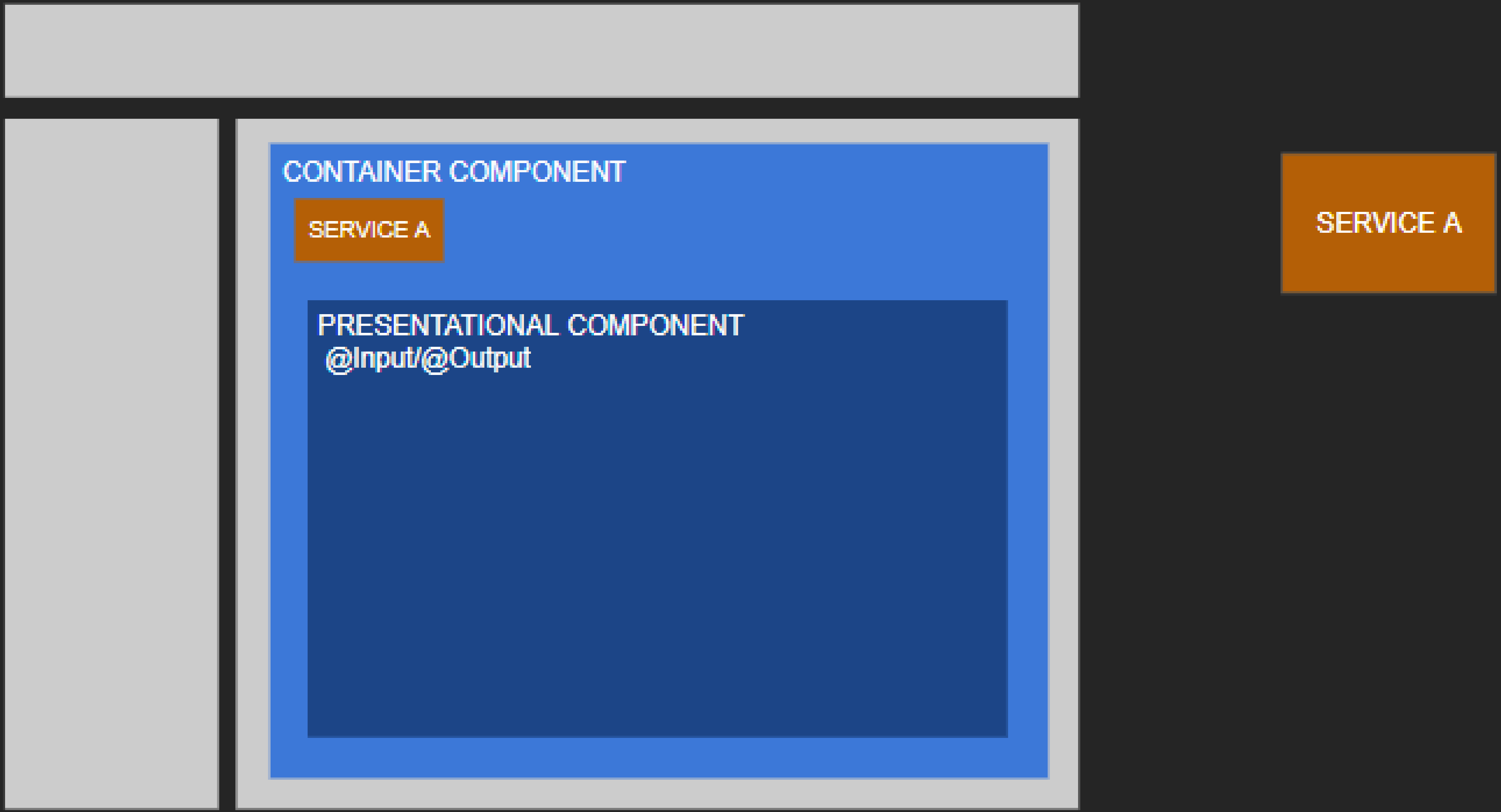
NgRX

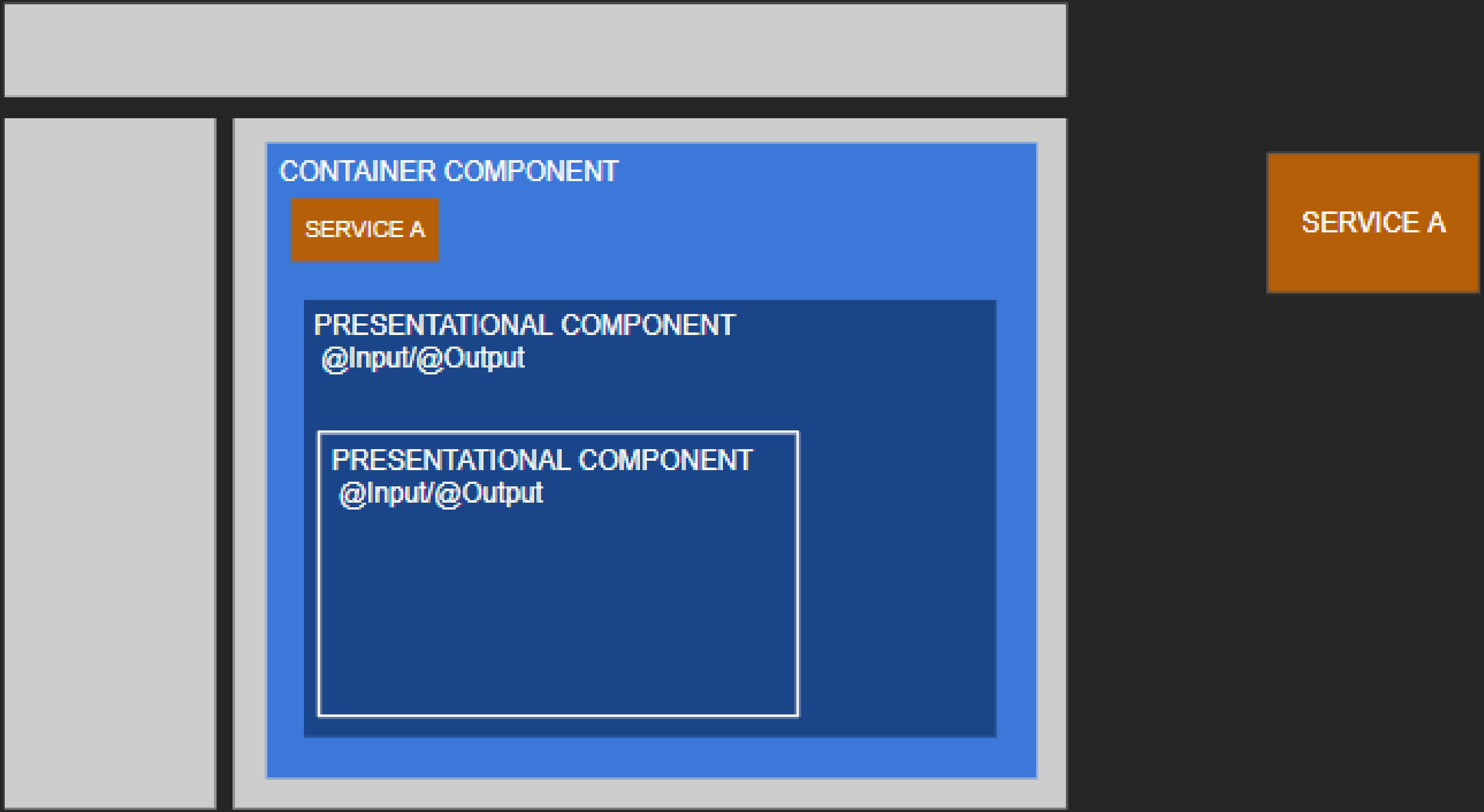
NgRX – is the redux pattern for Angular

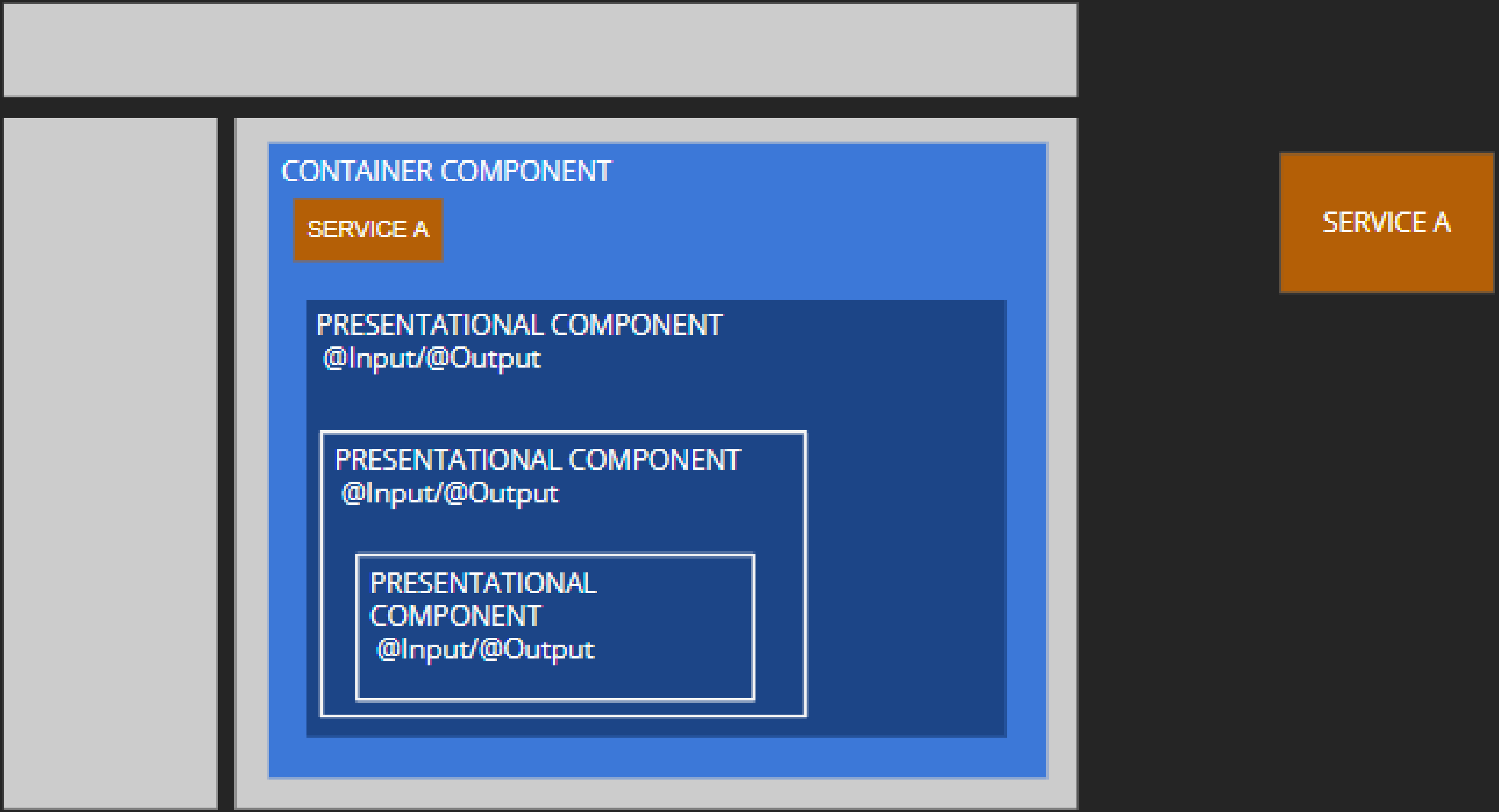
ngrx = redux Pattern + Angular + RxJS

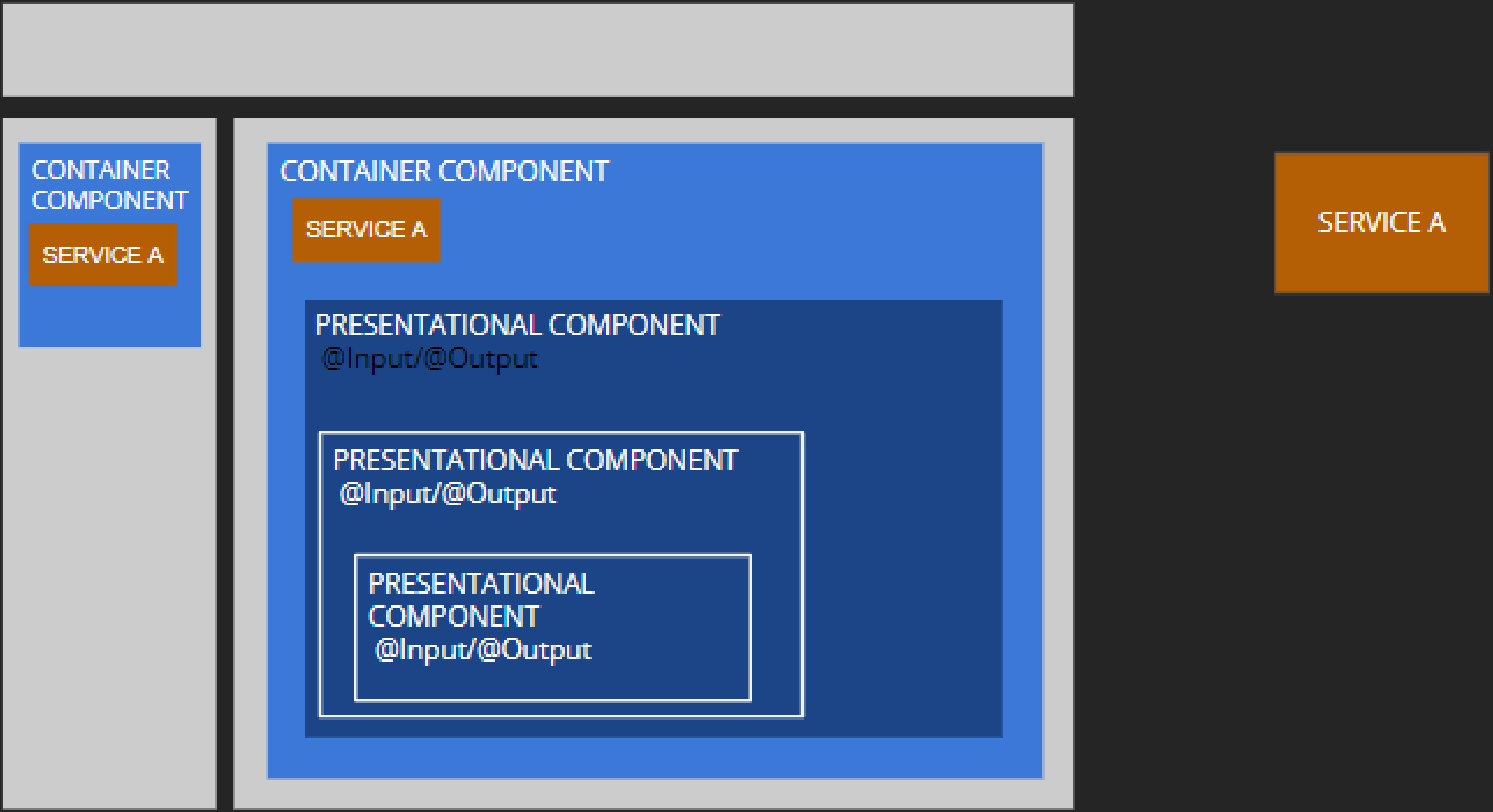

CONTAINER COMPONENT
SERVICE A
SERVICE A

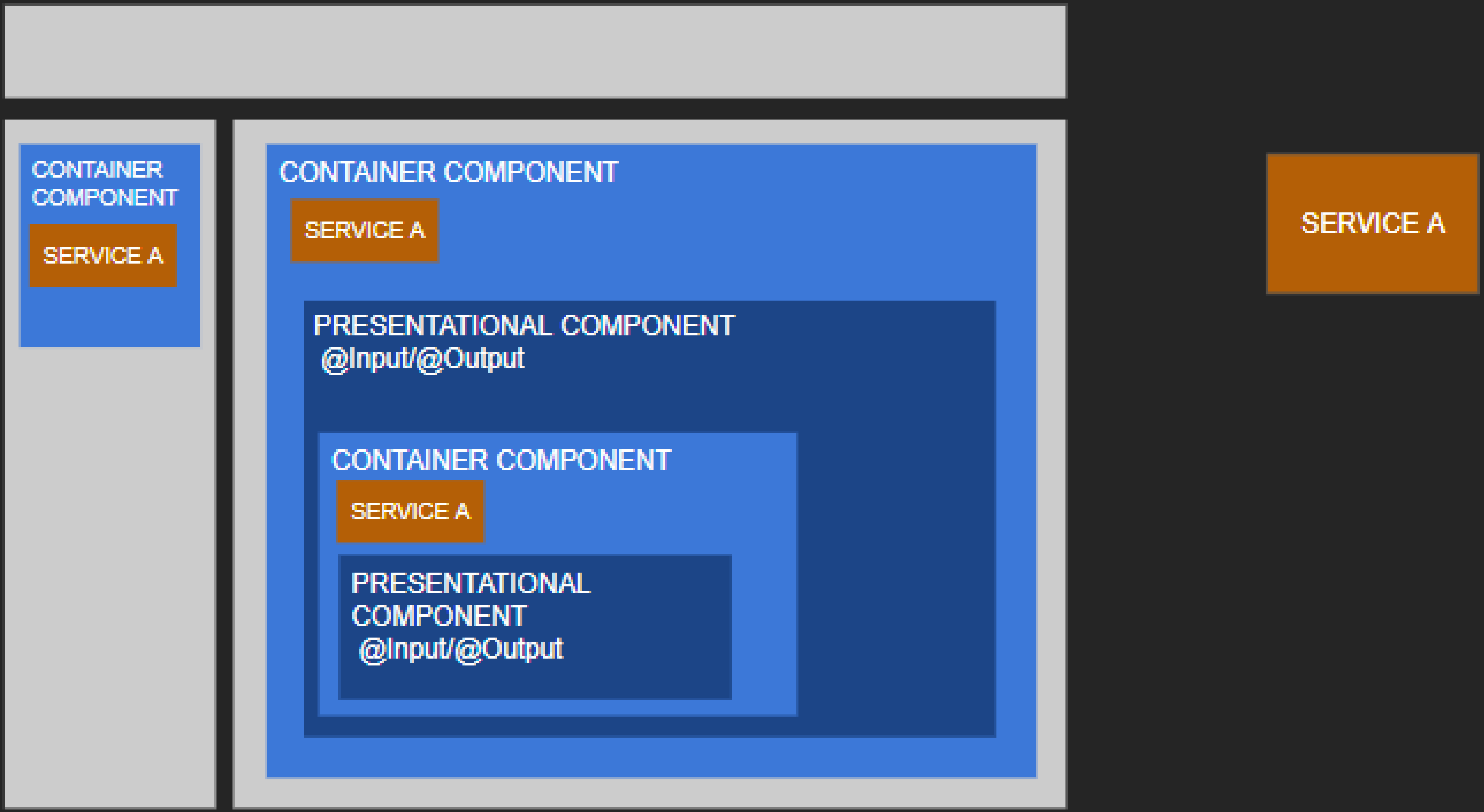

CONTAINER COMPONENT
SERVICE A
PRESENTATIONAL COMPONENT
SERVICE A

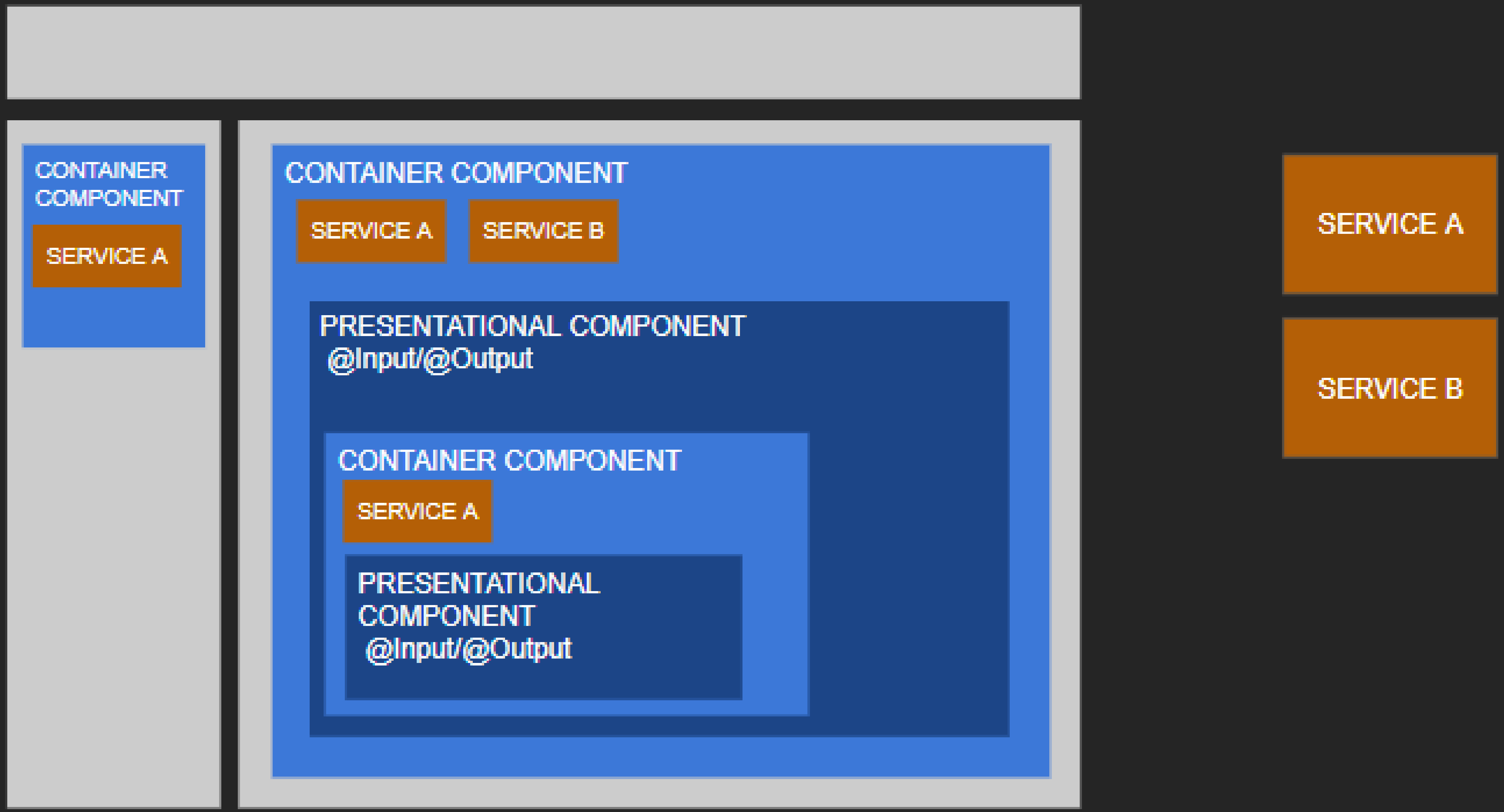

CONTAINER COMPONENT
SERVICE A
PRESENTATIONAL COMPONENT
@Input/@Output
SERVICE A

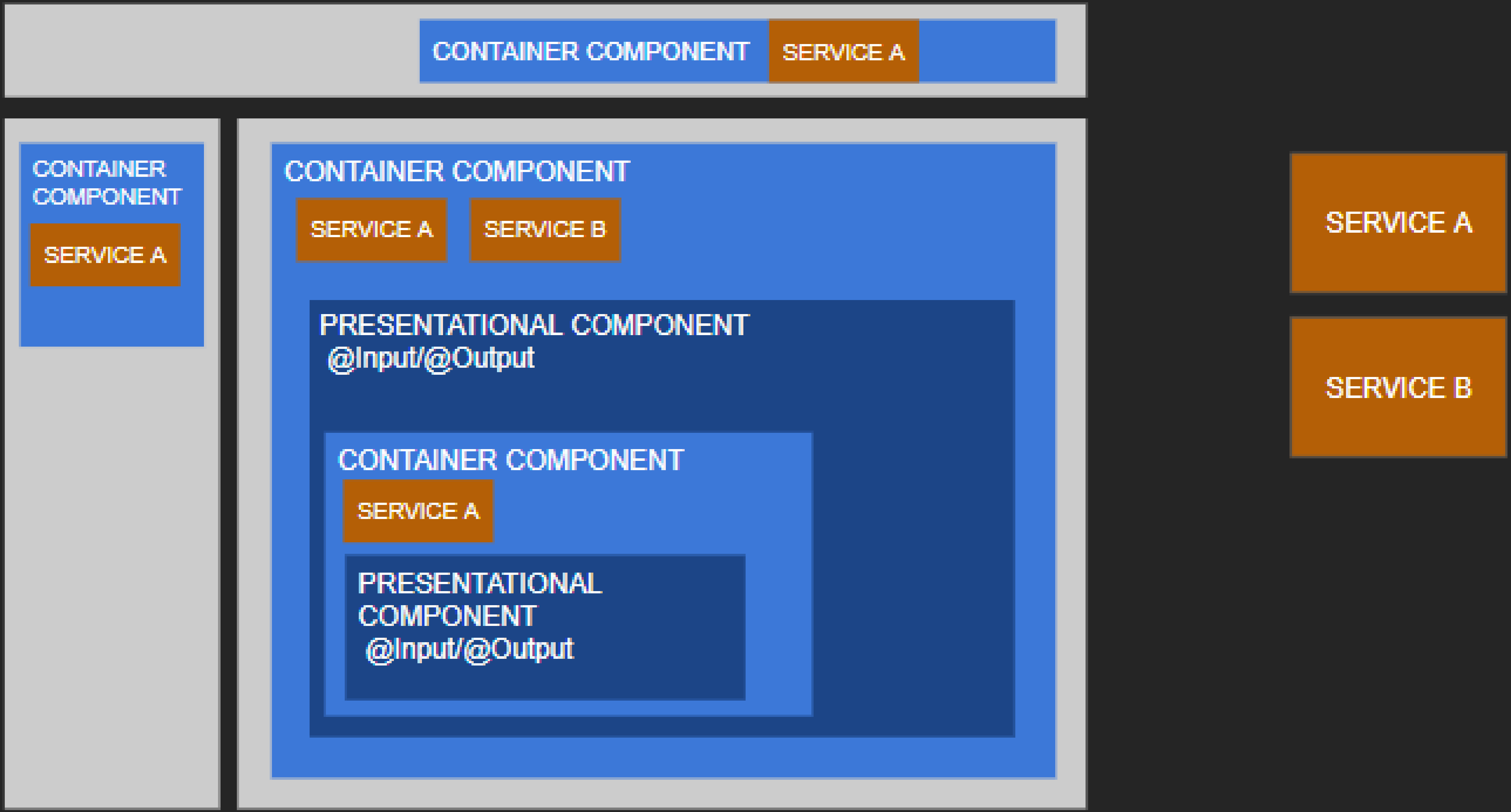

CONTAINER COMPONENT
SERVICE A
PRESENTATIONAL COMPONENT
@Input/@Output
PRESENTATIONAL COMPONENT
@Input/@Output
SERVICE A

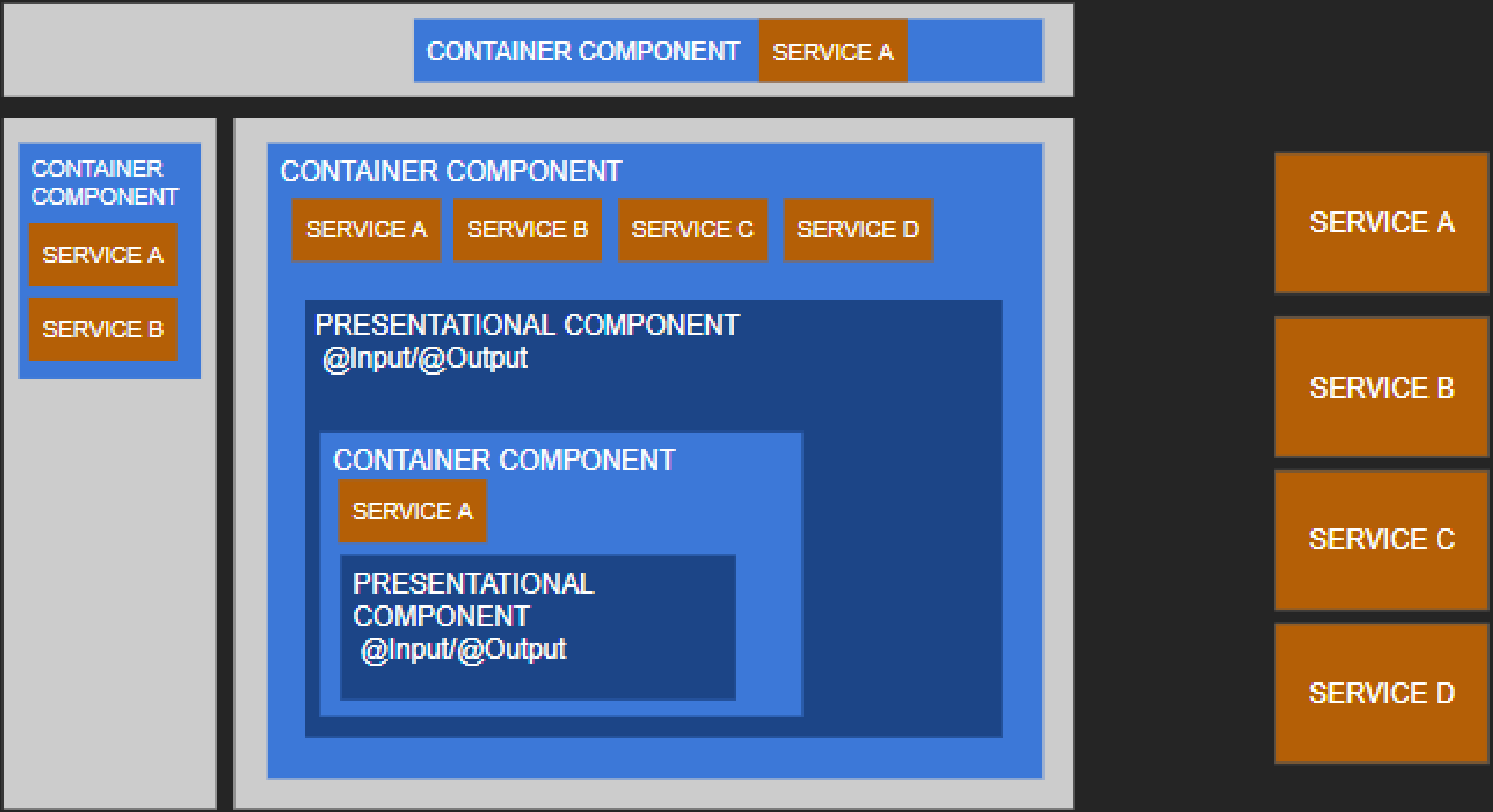

CONTAINER COMPONENT
SERVICE A
PRESENTATIONAL COMPONENT
@Input/@Output
PRESENTATIONAL COMPONENT
@Input/@Output
PRESENTATIONAL COMPONENT
@Input/@Output
SERVICE A

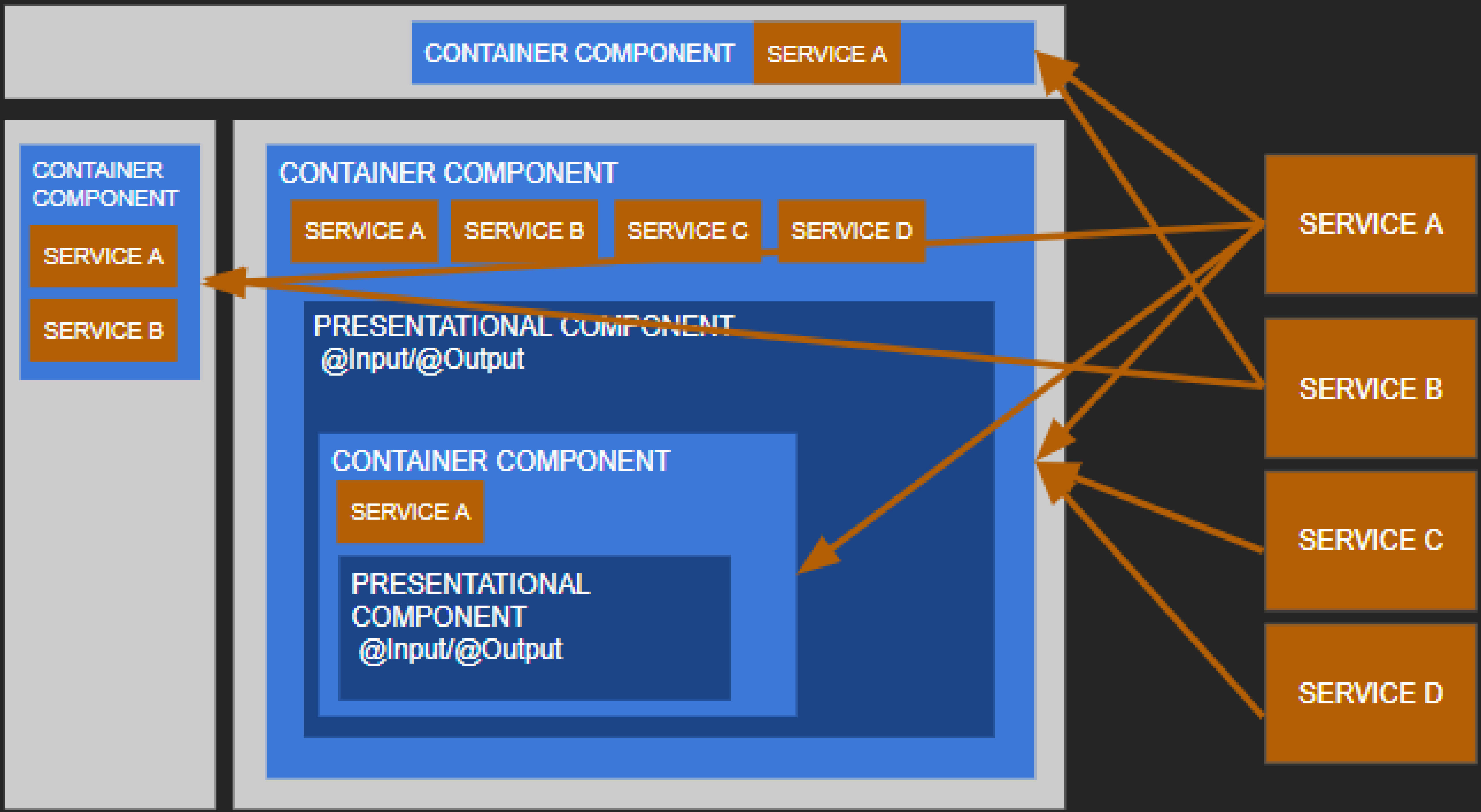

CONTAINER COMPONENT
SERVICE A
CONTAINER COMPONENT
SERVICE A
PRESENTATIONAL COMPONENT
@Input/@Output
PRESENTATIONAL COMPONENT
@Input/@Output
PRESENTATIONAL COMPONENT
@Input/@Output
SERVICE A

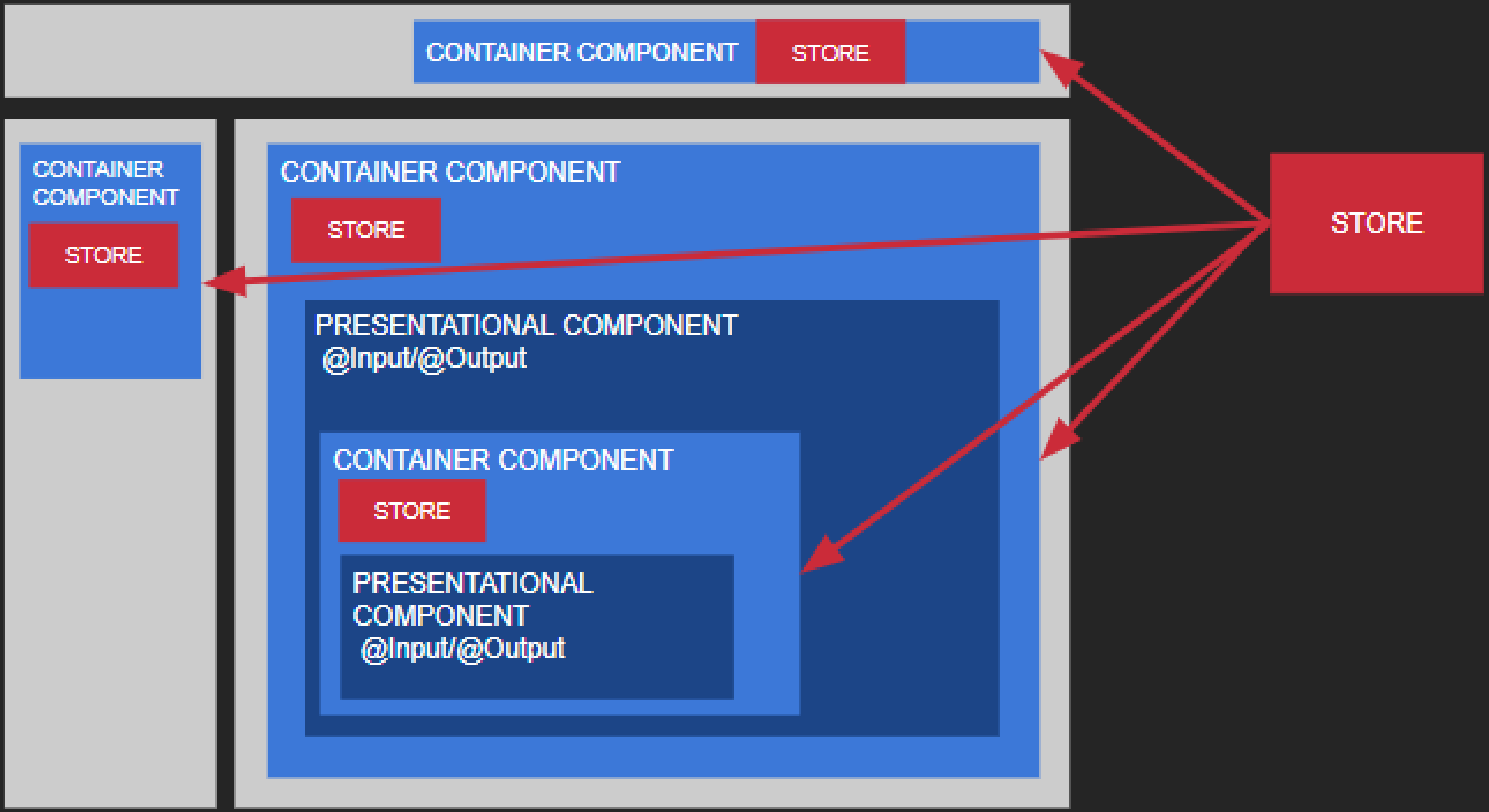

CONTAINER COMPONENT
SERVICE A
CONTAINER COMPONENT
SERVICE A
PRESENTATIONAL COMPONENT @Input/@Output
CONTAINER COMPONENT
SERVICE A
PRESENTATIONAL COMPONENT @Input/@Output
SERVICE A


CONTAINER COMPONENT
SERVICE A
CONTAINER COMPONENT
SERVICE A
SERVICE B
PRESENTATIONAL COMPONENT
@Input/@Output
CONTAINER COMPONENT
SERVICE A
PRESENTATIONAL COMPONENT
@Input/@Output
SERVICE A
SERVICE B


CONTAINER COMPONENT
SERVICE A
CONTAINER COMPONENT
SERVICE A
CONTAINER COMPONENT
SERVICE A
SERVICE B
PRESENTATIONAL COMPONENT
@Input/@Output
CONTAINER COMPONENT
SERVICE A
PRESENTATIONAL COMPONENT
@Input/@Output
SERVICE A
SERVICE B


CONTAINER COMPONENT
SERVICE A
CONTAINER COMPONENT
SERVICE A
SERVICE B
CONTAINER COMPONENT
SERVICE A
SERVICE B
SERVICE C
SERVICE D
PRESENTATIONAL COMPONENT @Input/@Output
CONTAINER COMPONENT
SERVICE A
PRESENTATIONAL COMPONENT @Input/@Output
SERVICE A
SERVICE B
SERVICE C
SERVICE D


CONTAINER COMPONENT
SERVICE A
CONTAINER COMPONENT
SERVICE A
SERVICE B
CONTAINER COMPONENT
SERVICE A
SERVICE B
SERVICE C
SERVICE D
PRESENTATIONAL COMPONENT
@Input/@Output
CONTAINER COMPONENT
SERVICE A
PRESENTATIONAL COMPONENT
@Input/@Output
SERVICE A
SERVICE B
SERVICE C
SERVICE D


CONTAINER COMPONENT
STORE
CONTAINER COMPONENT
STORE
CONTAINER COMPONENT
STORE
STORE
PRESENTATIONAL COMPONENT
@Input/@Output
CONTAINER COMPONENT
STORE
PRESENTATIONAL COMPONENT
@Input/@Output

new state

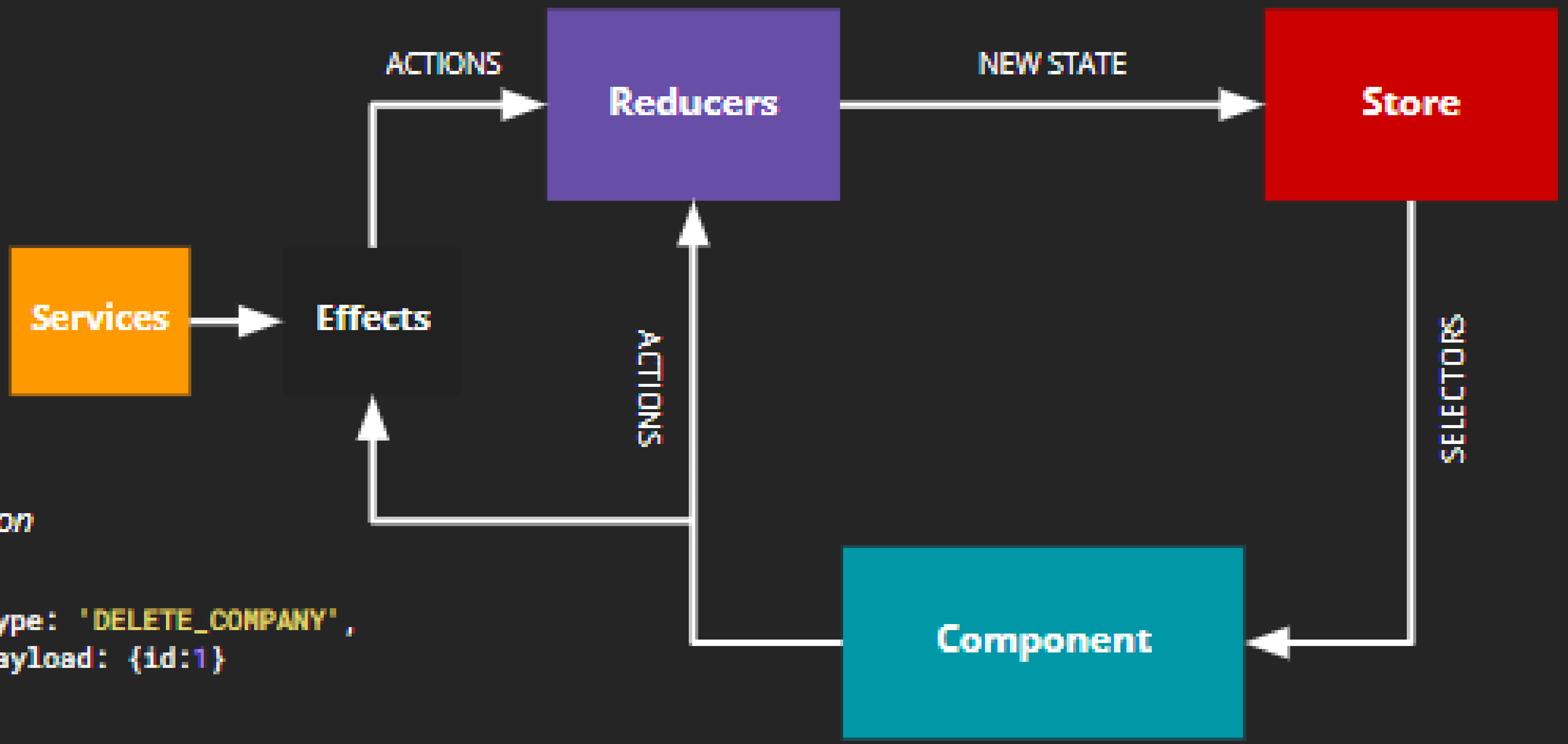
```
companies: [
    {id:2, name:'Microsoft'}
];
```

store state

```
{
  companies: [
   {id:2, name:'Microsoft'}
  ],
  contacts: [
   {id:1, name:Duncan}
  ]
};
```

action

```
{
  type: 'DELETE_COMPANY_SUCCESS',
  payload: {id:1}
}
```



action

```
{
  type: 'DELETE_COMPANY',
  payload: {id:1}
}
```

Let's
</>
CODE It.