

FullStack #WebDevelopment Bootcamp

Course Week 7

Summary – Why Angular

Latest web tools and patterns

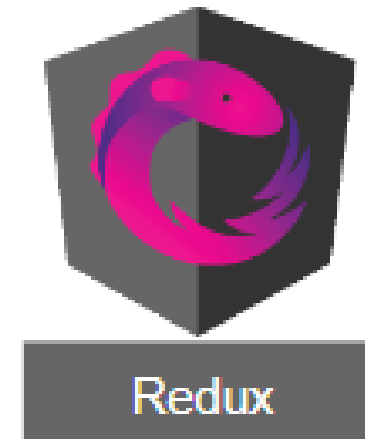
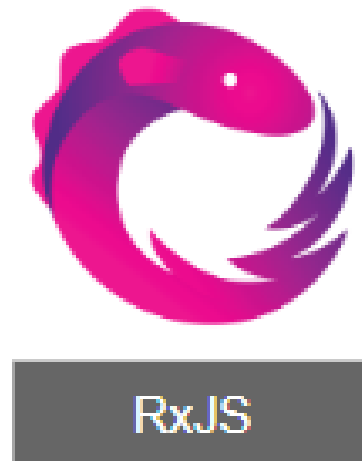
Super fast

Huge community

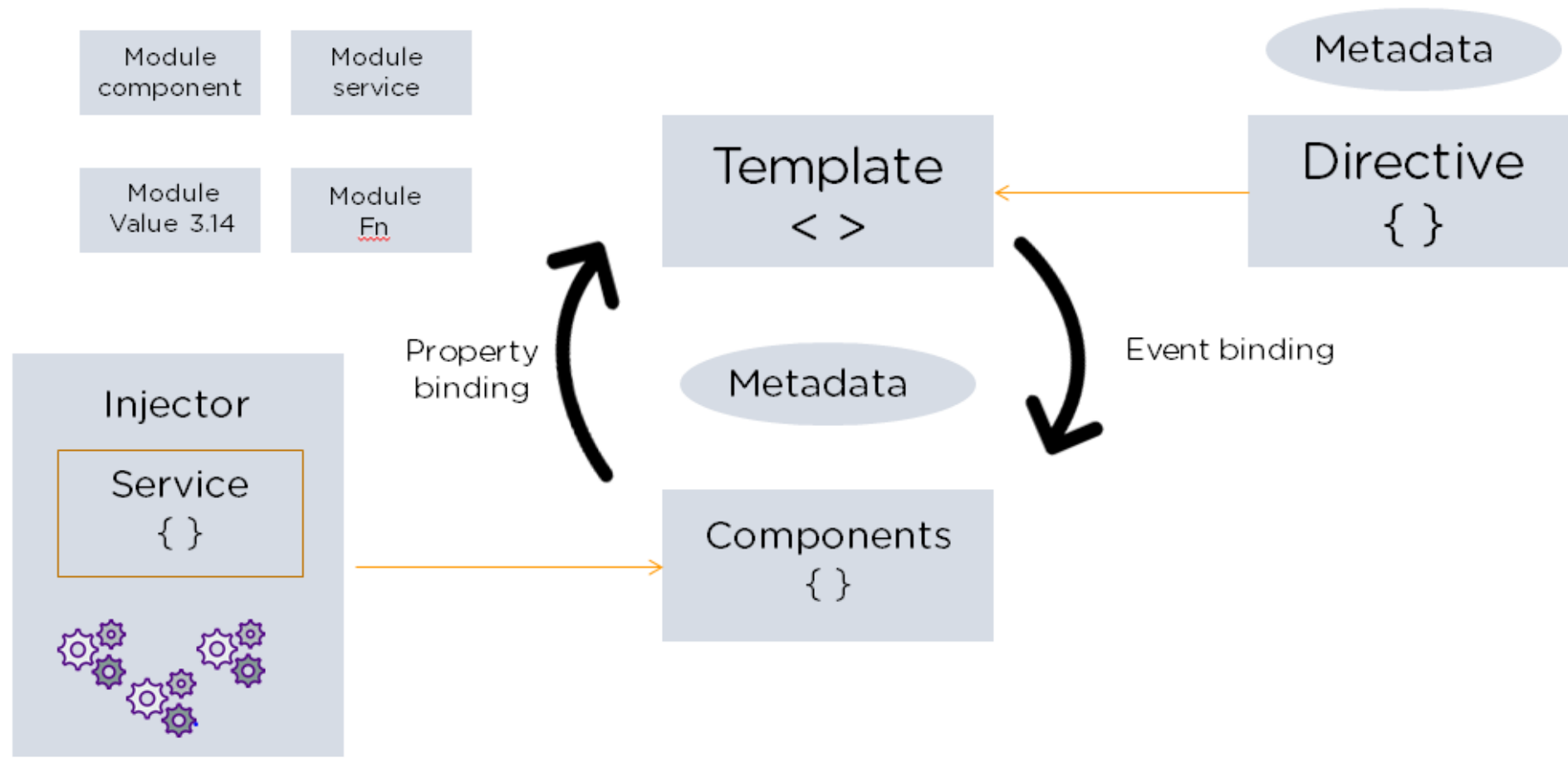
Enterprise friendly

Asynchronous Programming

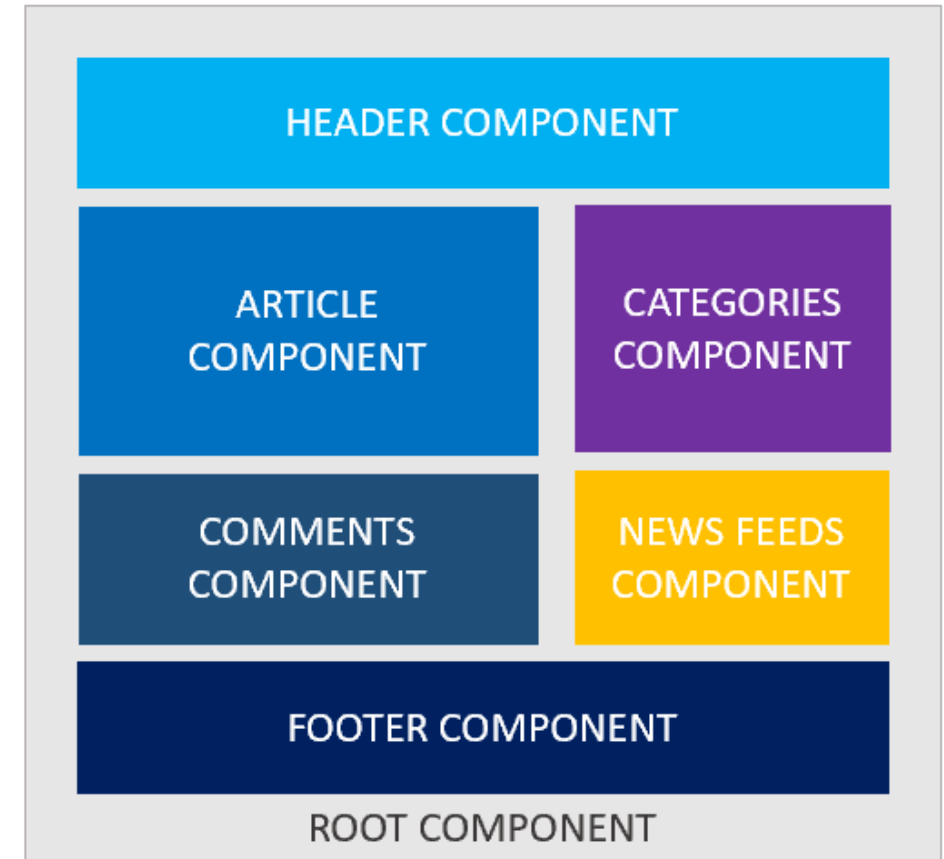
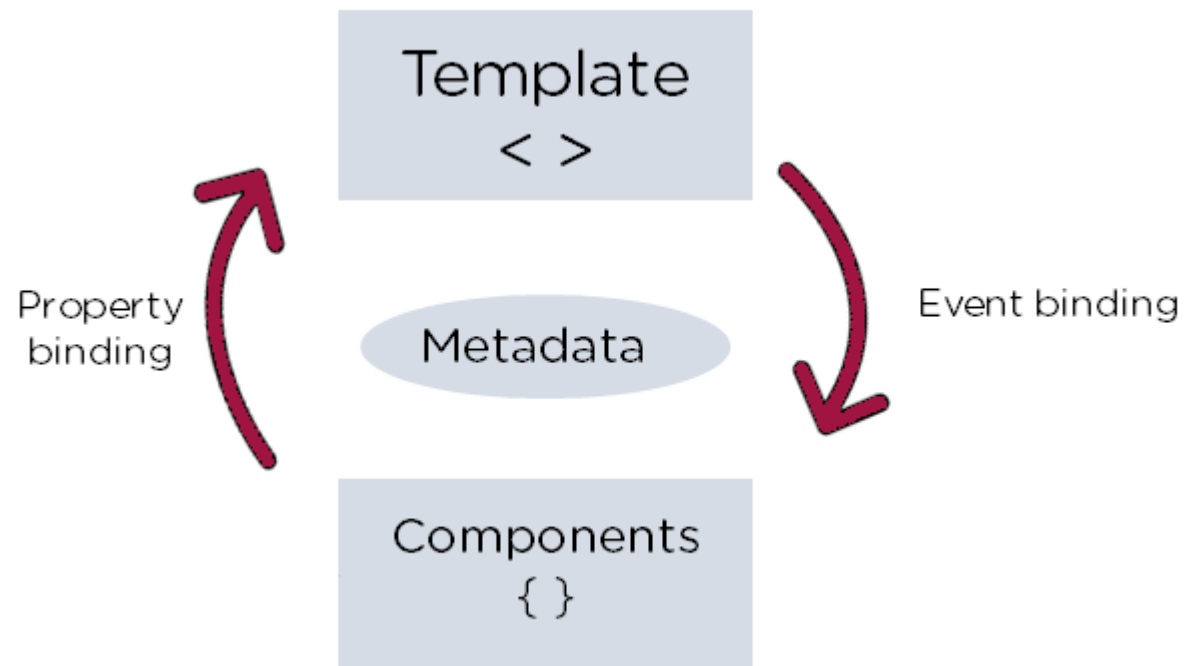
Angular itself is easy
... but it forces you to confront modern web
development.



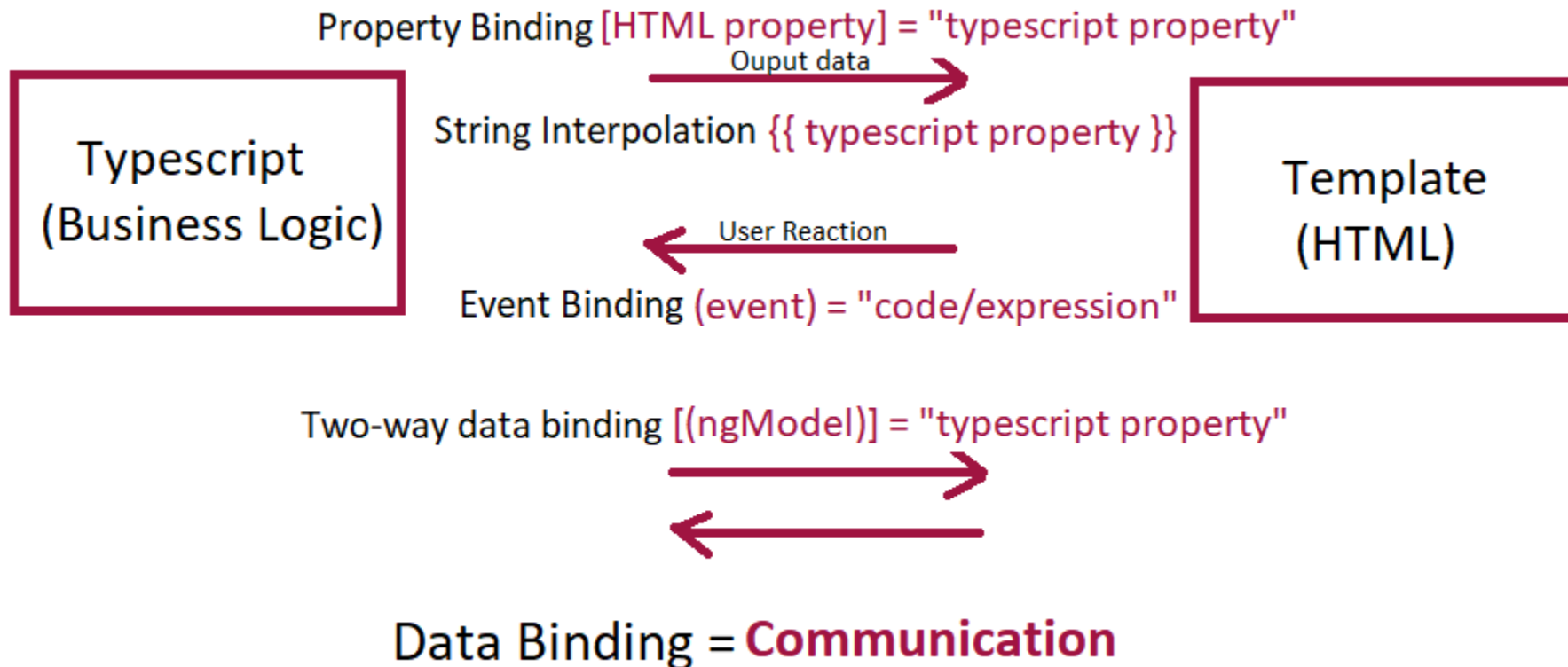
Angular Architecture Overview



Angular - Component-Based Architecture •msg



Angular - Component-Based Architecture •msg



Angular - Component-Based Architecture •msg

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

Metadata

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

Component
Class

```
export class HelloComponent implements OnInit {  
  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>
```

Presentation and Container Components •msg

Presentation

Concerned with how things look

HTML markup and CSS styles

No dependencies on the rest of the app

Don't specify how data is loaded or changed but emit events via @Outputs

Receive data via @Inputs

May contain other components

Container

Concerned with how things work

Have little to no HTML and CSS styles

Have injected dependencies

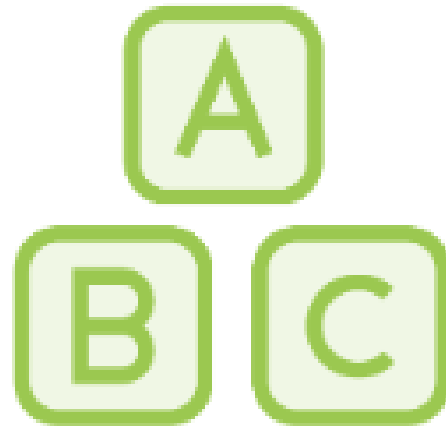
Are stateful and specify how data is loaded or changed

Top level routes

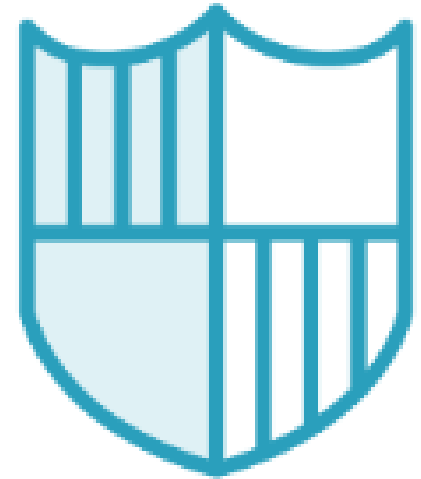
May contain other components



Performance



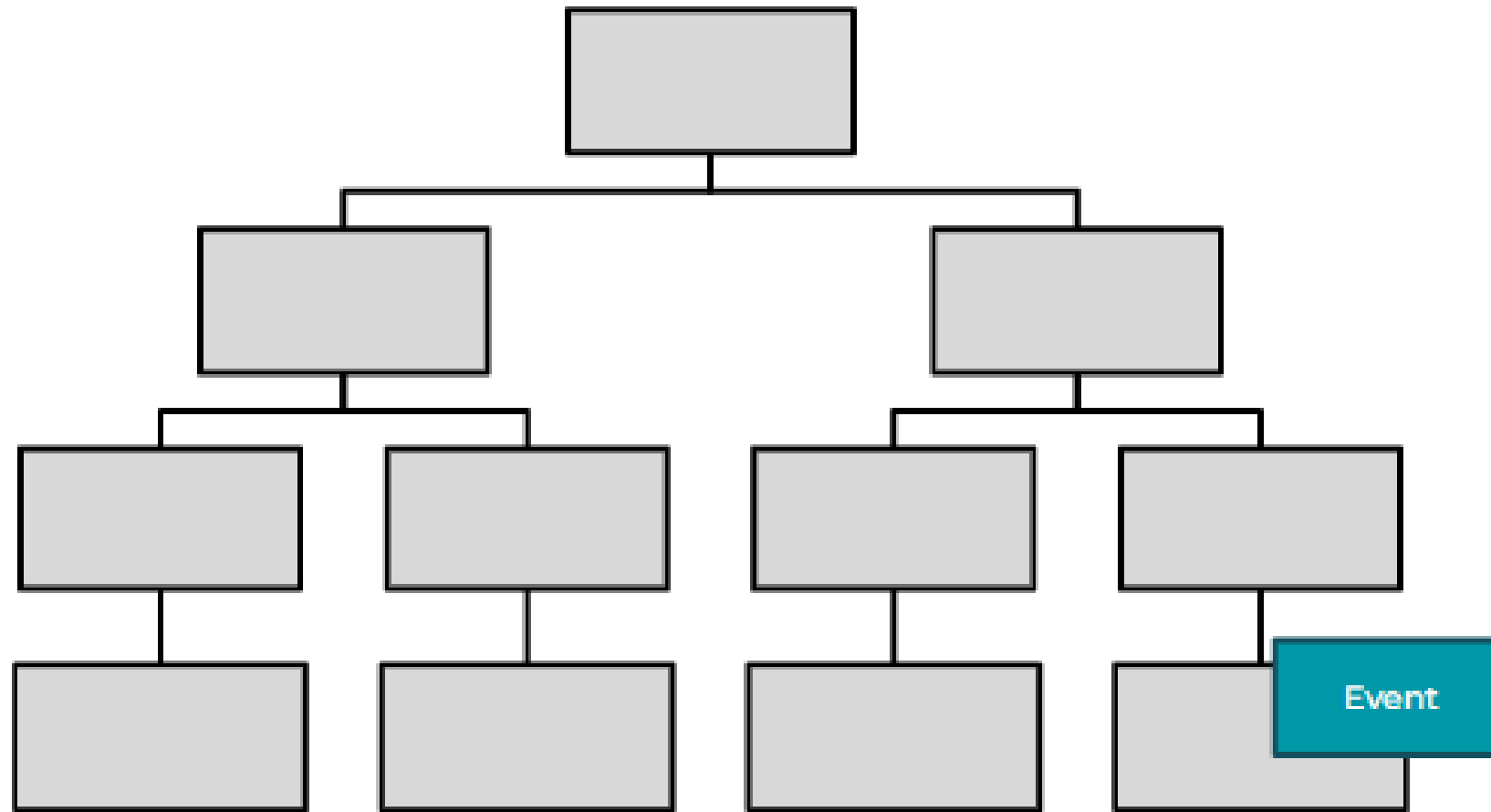
Composability

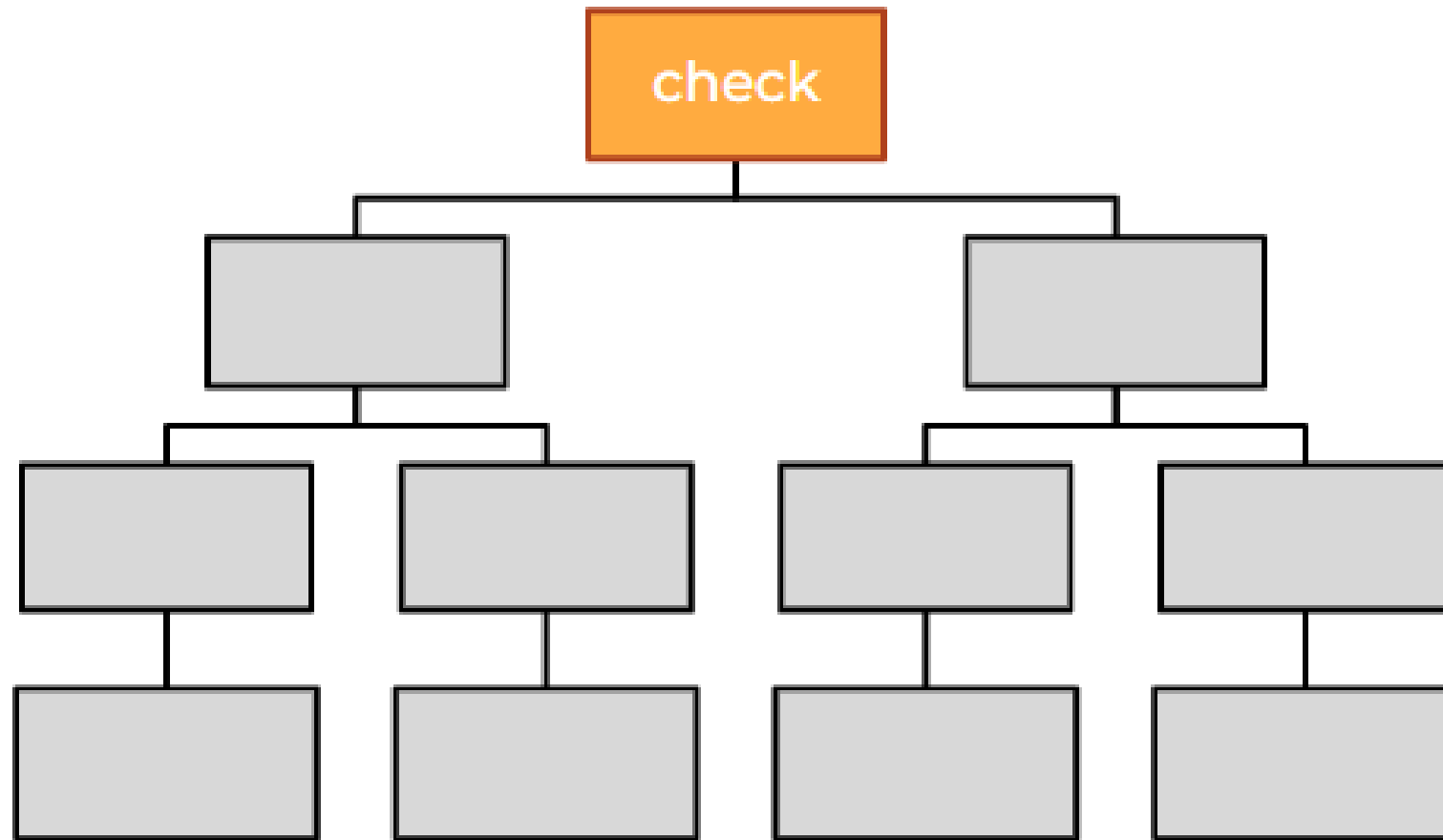


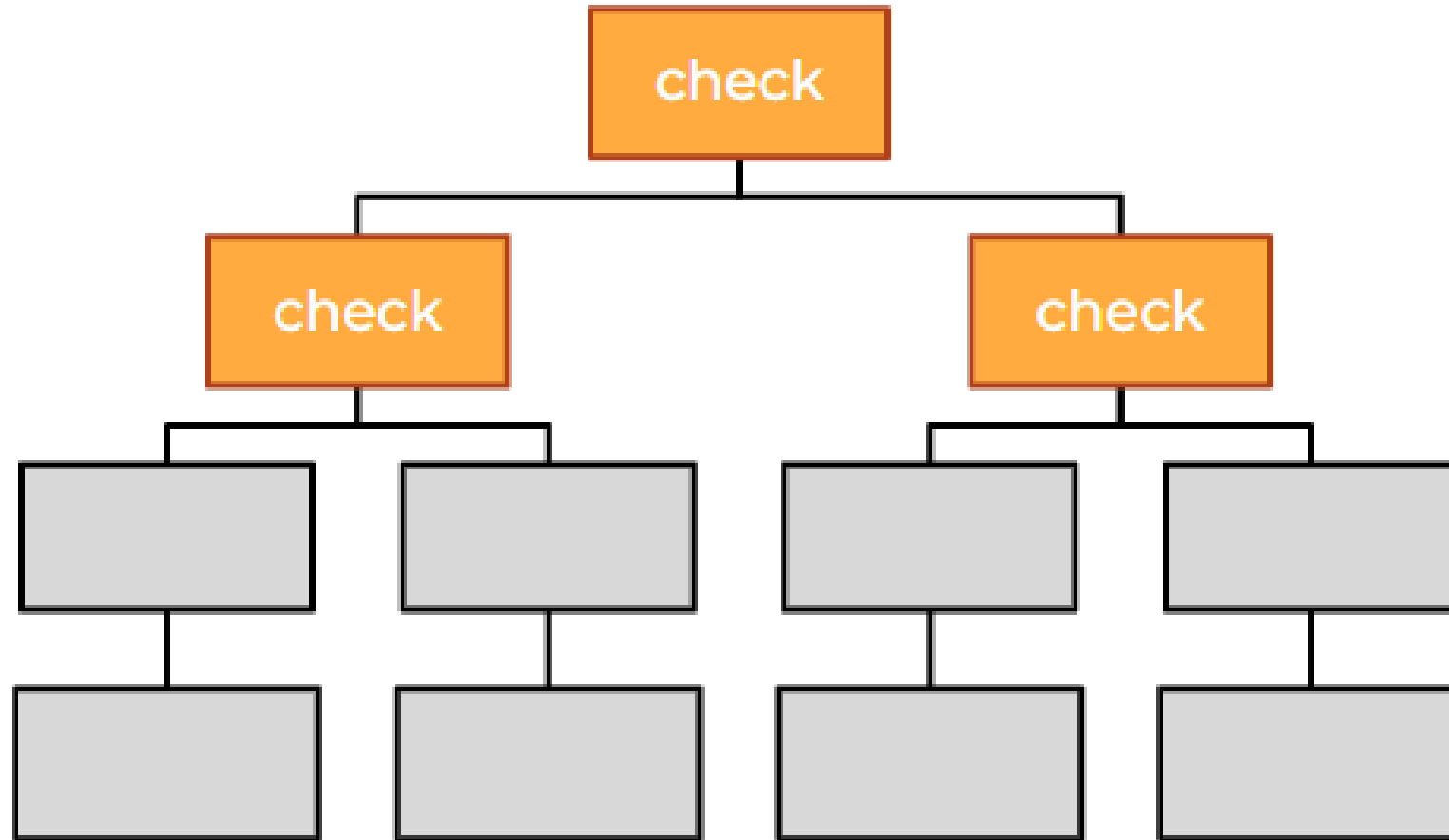
Easier to test

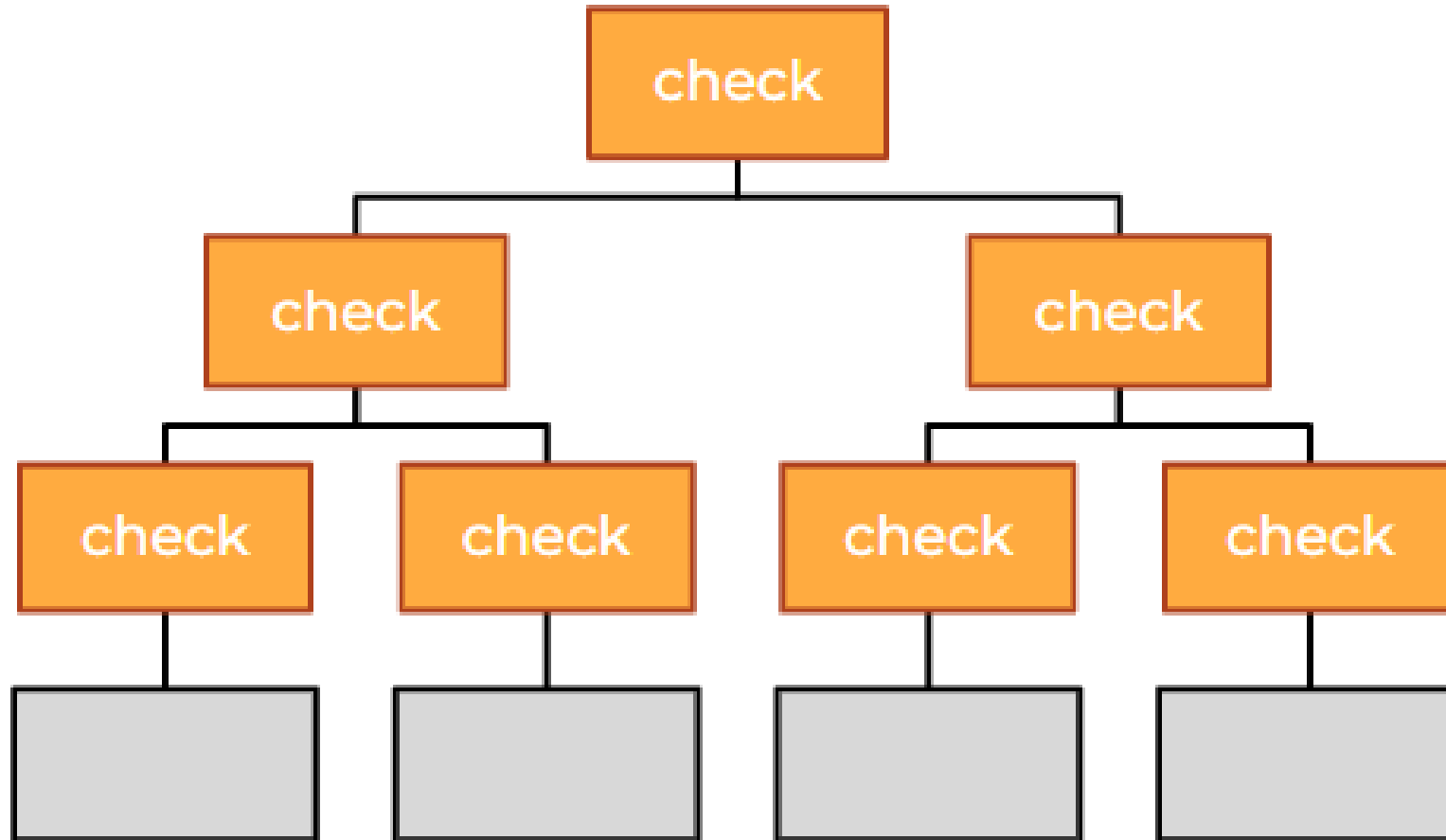
ChangeDetectionStrategy.OnPush

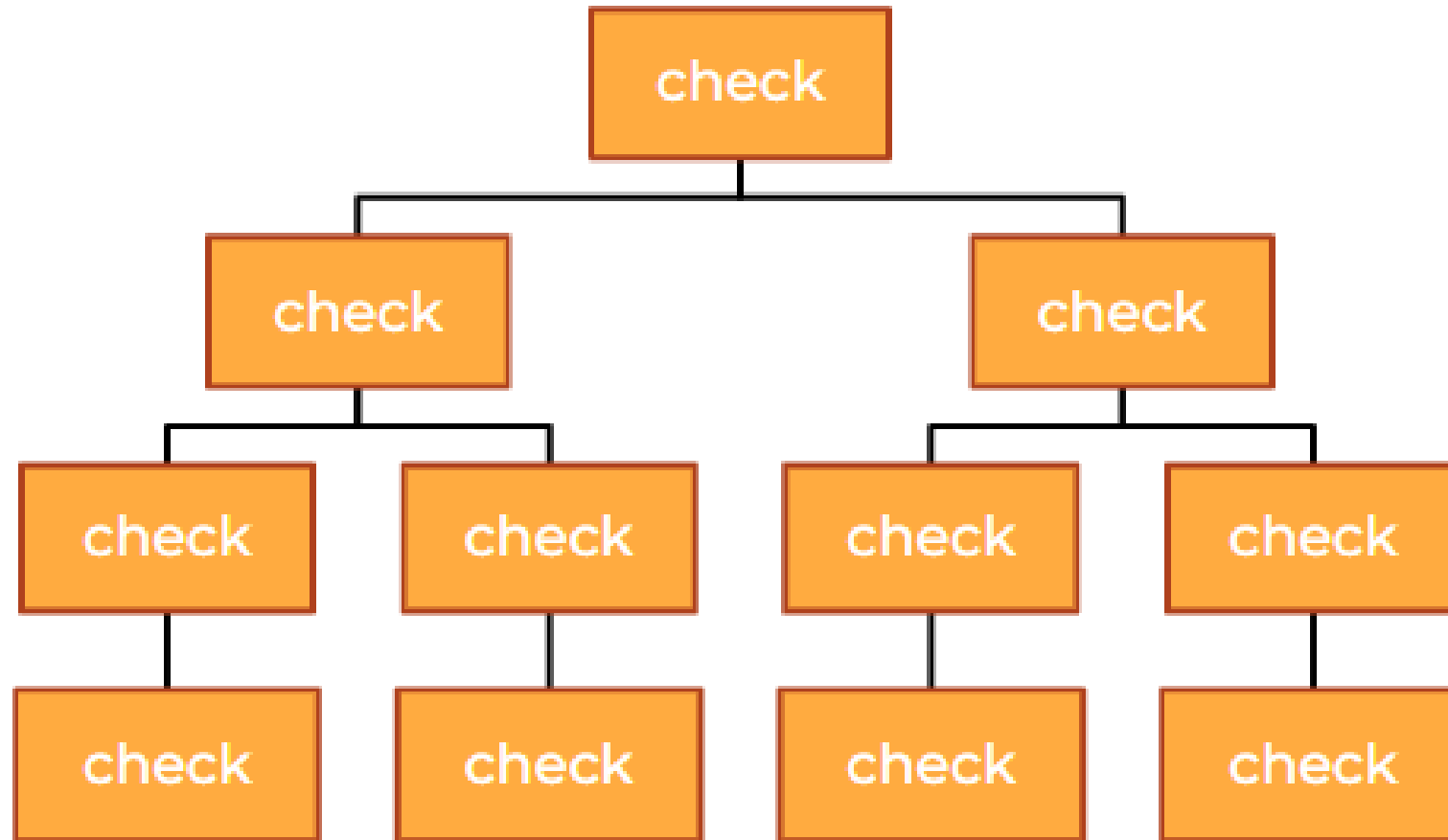
ChangeDetectionStrategy.Default



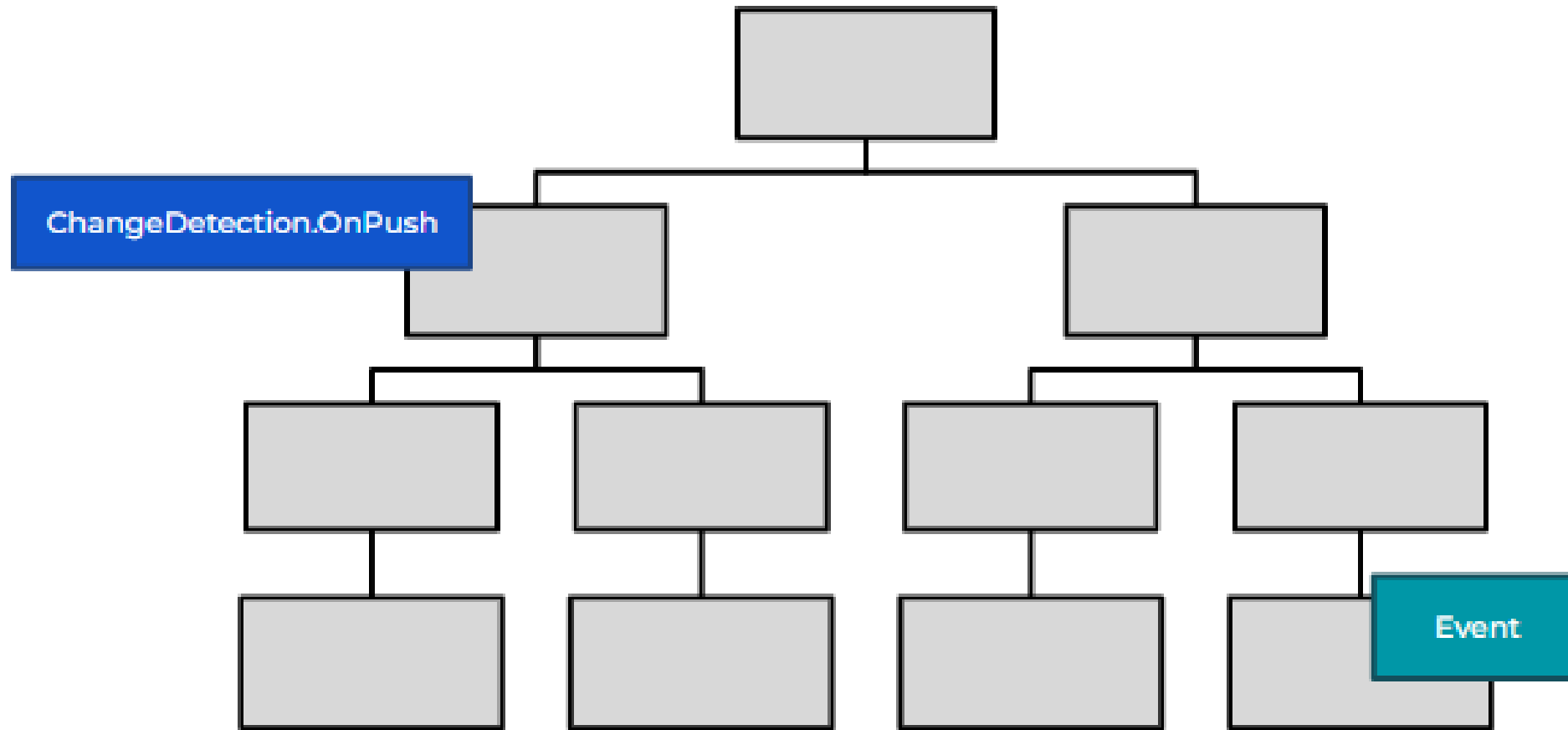




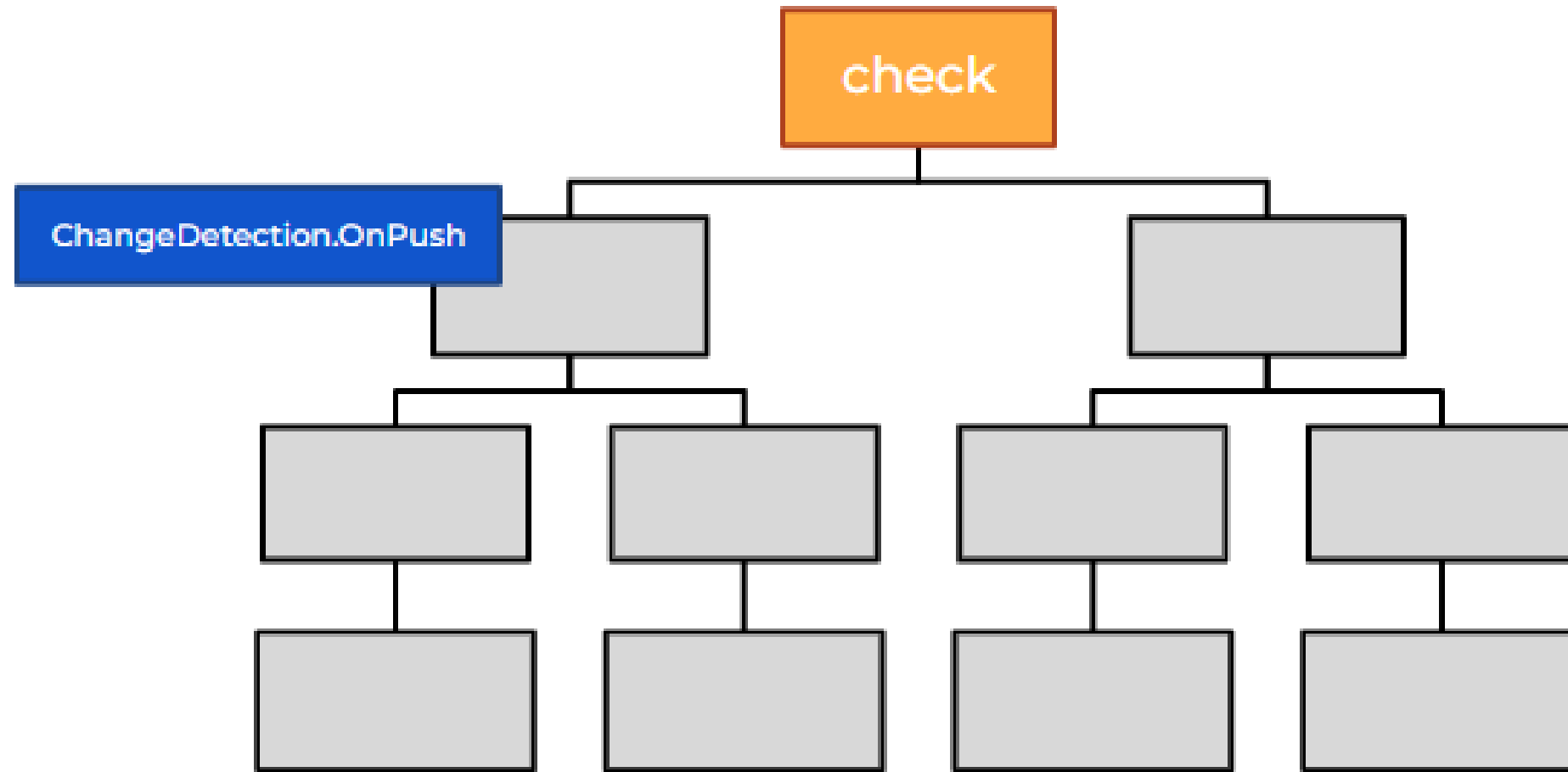




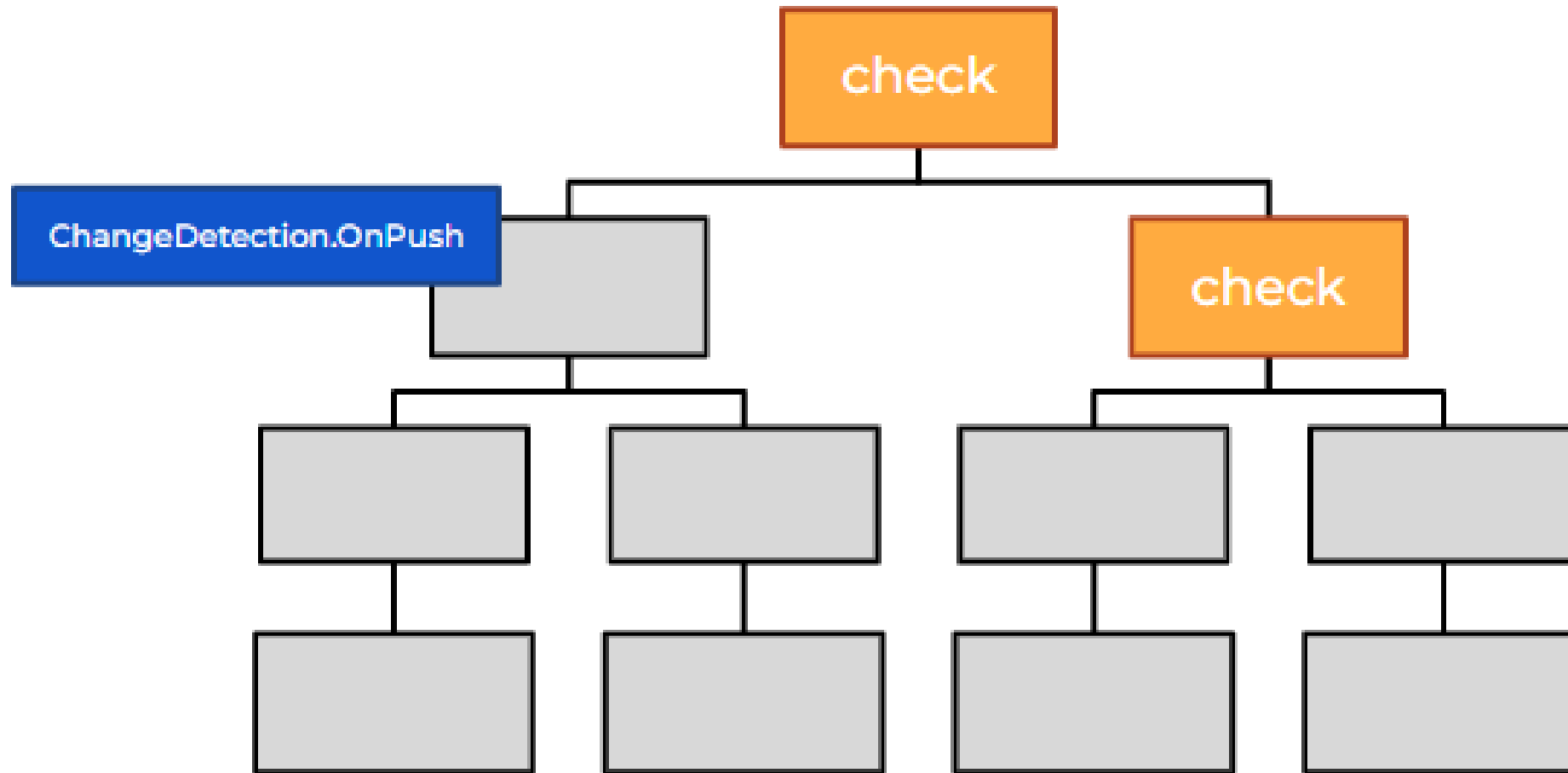
ChangeDetectionStrategy.OnPush



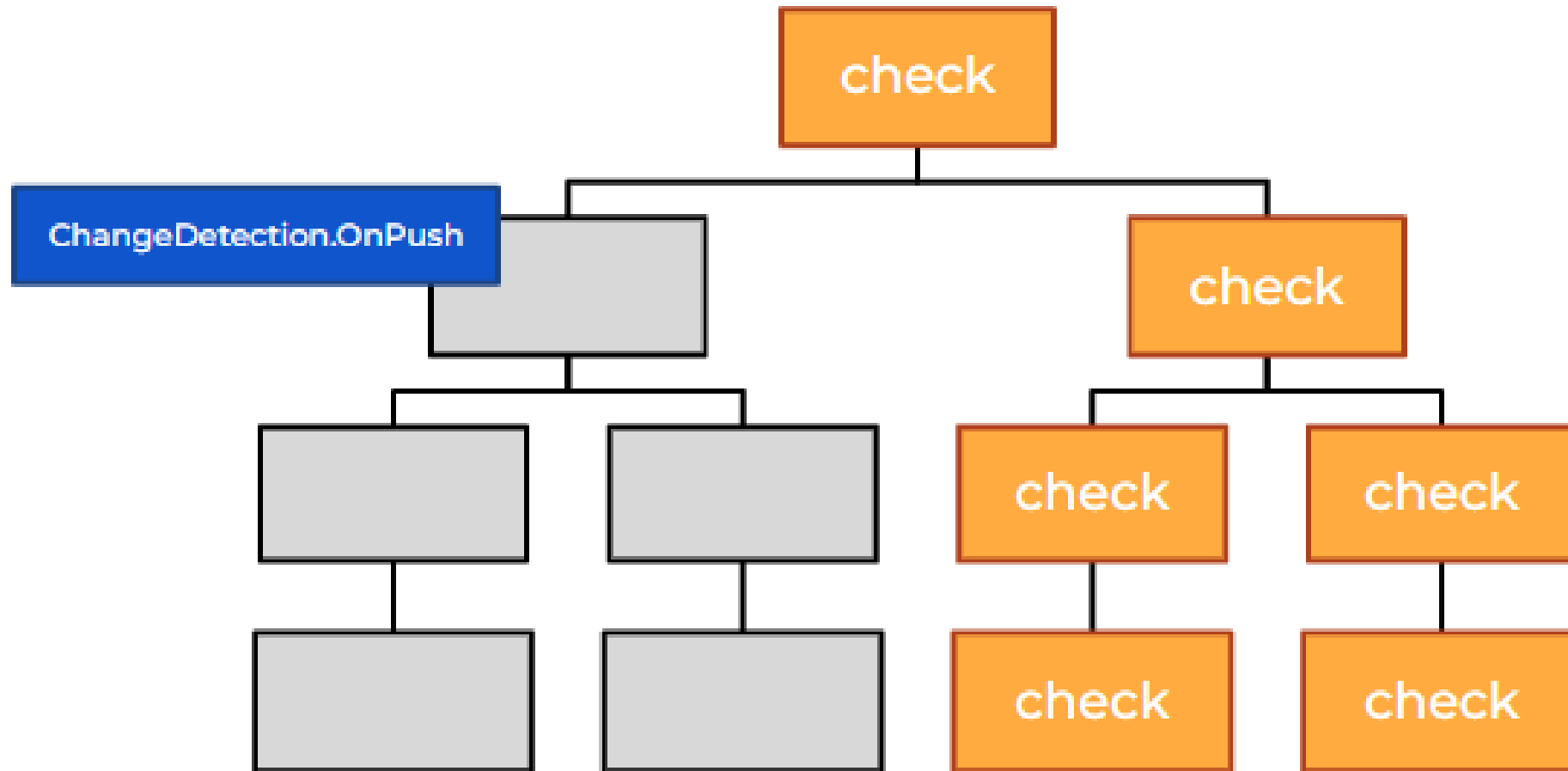
ChangeDetectionStrategy.OnPush



ChangeDetectionStrategy.OnPush



ChangeDetectionStrategy.OnPush





RxJS

RxJS simplifies async code



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]

The diagram illustrates a curved path of light rays, represented by a series of white dots, as they travel through a gravitational well. The path starts from the bottom left and curves upwards and to the right. At the top of the curve, a box labeled 'Observable [1, 4]' is positioned. Two observers, 'Observer 1 [1, 4]' and 'Observer 2 [1, 4]', are located at the bottom of the curve, with arrows pointing towards the path of the light rays. The background is a dark, curved surface representing the gravitational well, with several purple and blue light trails following the curve of the well.

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]

Observable
[1]


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

Observer 1
[]

Observable
[1, 4]

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

Observer 1
[4]




The diagram is set against a background of purple and blue light trails, suggesting motion. It illustrates the concept of filtering in a reactive programming context. At the top, a box labeled 'Observable' contains the array '[1, 4, 7]'. A dotted line with an arrowhead points from this box down to a grey box containing the code '.filter(x => x > 2)'. From this grey box, another dotted line with an arrowhead points down to a box labeled 'Observer 1' which contains the array '[4, 7]'. This visualizes how the filter operation transforms the original sequence of values into a new sequence based on a predicate.

Observable
[1, 4, 7]

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

Observer 1
[4, 7]

A long-exposure photograph of a multi-lane highway at night. The image shows multiple curved lanes of the road, with light trails from cars appearing as bright, streaky lines of white and yellow. The background is dark, with some distant lights visible on the horizon.

RxJS is a journey...

Enter search term.....

```
this.users$ = this.searchText.valueChanges
    .debounceTime(500)
    .distinctUntilChanged()
    .switchMap(searchText => this.userService.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);  
}
```

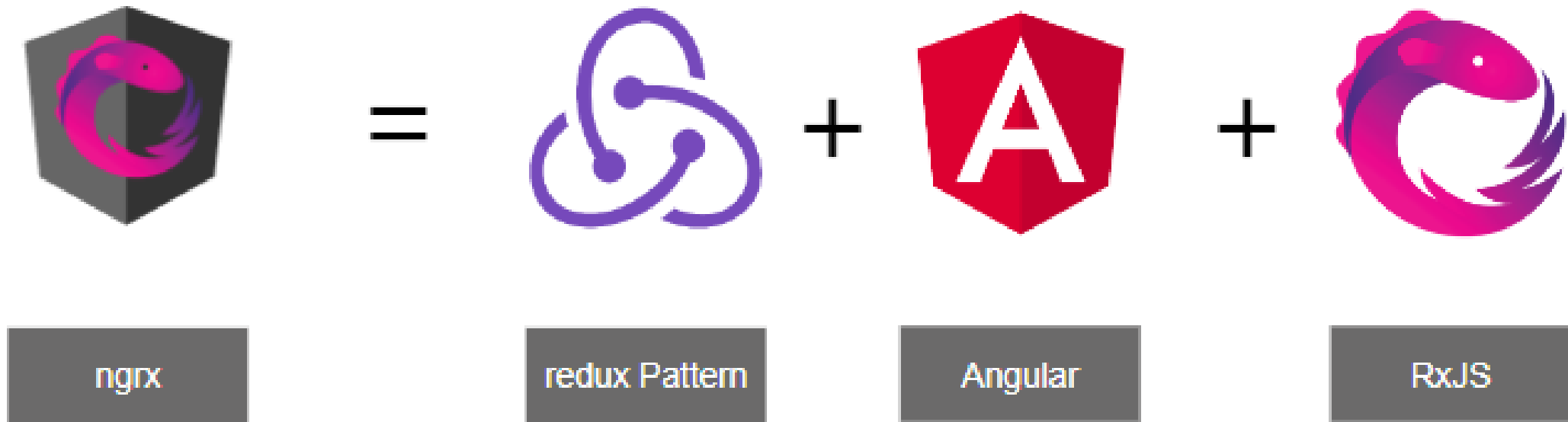
Modern web development is asynchronous

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

RxJS is baked into Angular



NgRX – is the redux pattern for Angular



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

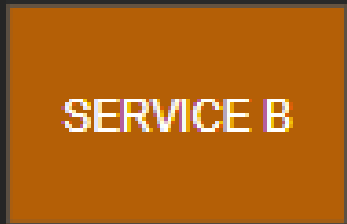
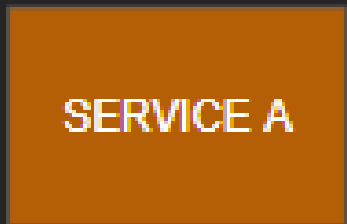
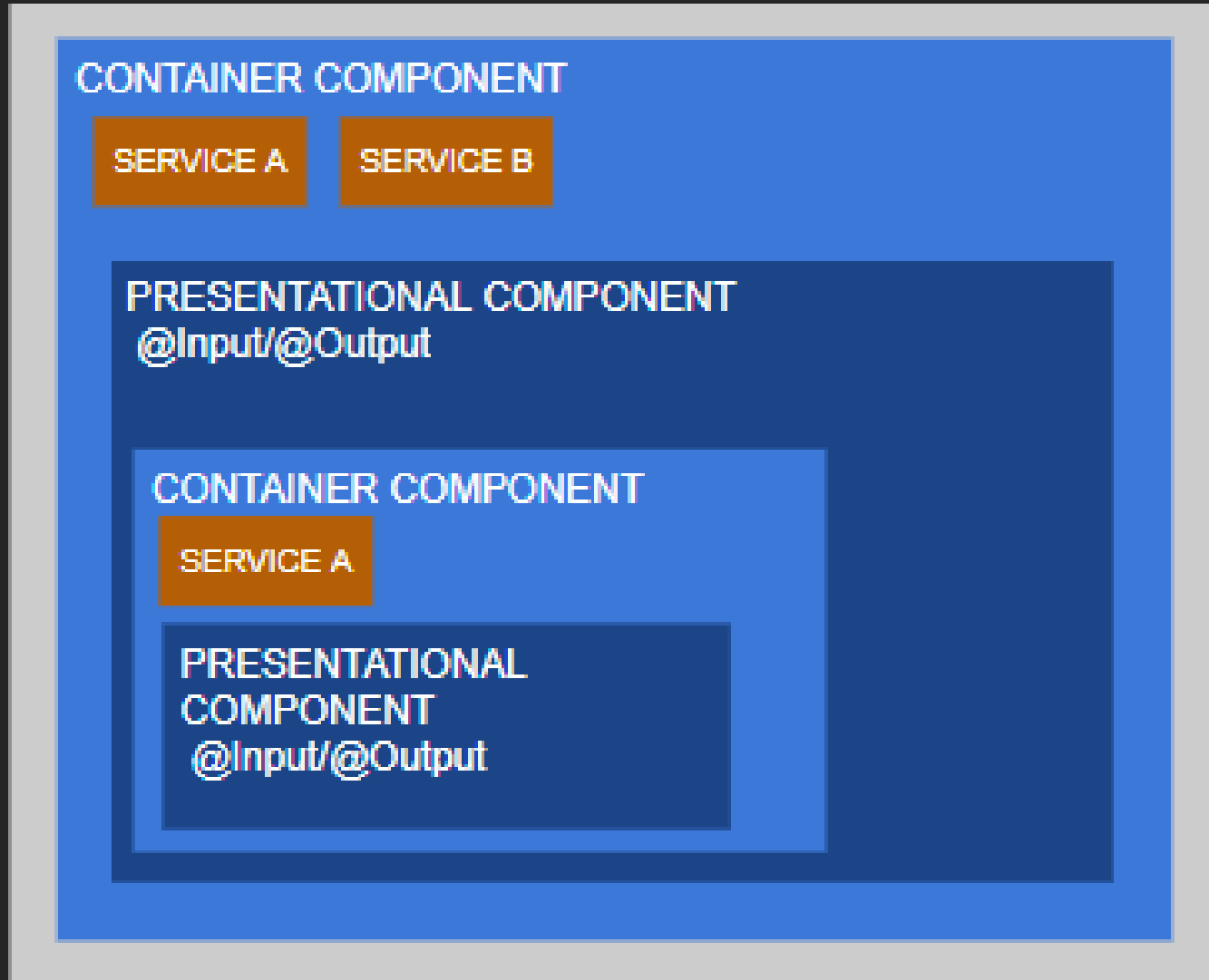
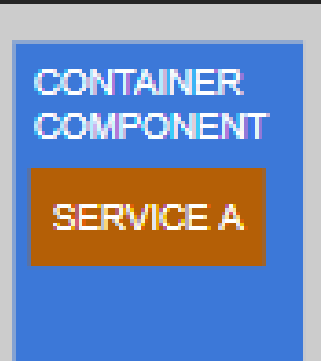
CONTAINER COMPONENT

SERVICE A

PRESENTATIONAL
COMPONENT
@Input/@Output

SERVICE A

SERVICE B



CONTAINER COMPONENT

SERVICE A

CONTAINER COMPONENT

SERVICE A

SERVICE B

CONTAINER COMPONENT

SERVICE A

SERVICE B

SERVICE C

SERVICE D

PRESENTATIONAL COMPONENT
@Input/@Output

CONTAINER COMPONENT

SERVICE A

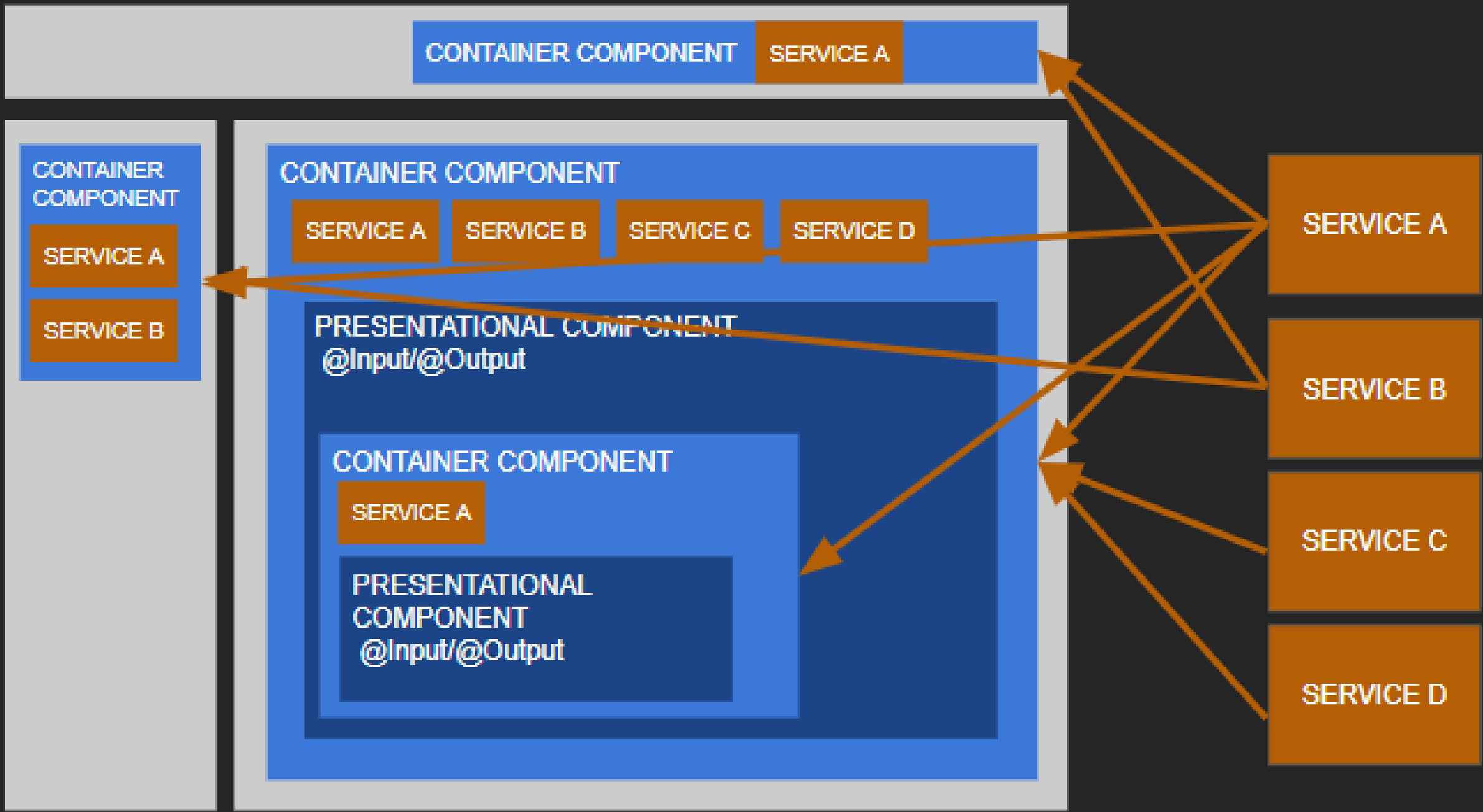
PRESENTATIONAL COMPONENT
@Input/@Output

SERVICE A

SERVICE B

SERVICE C

SERVICE D





action

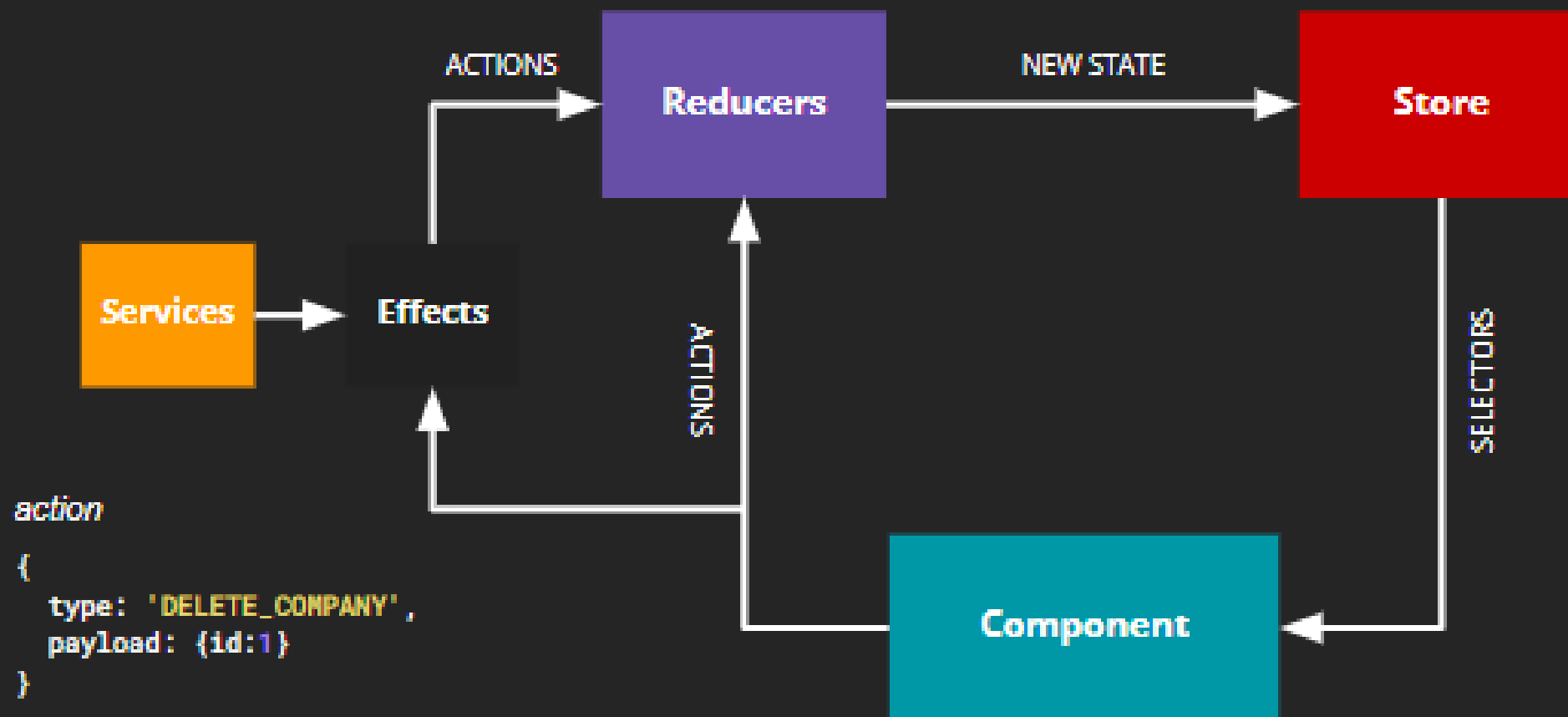
```
{  
  type: 'DELETE_COMPANY_SUCCESS',  
  payload: {id:1}  
}
```

new state

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

store state

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



action

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

Let's 
CODE It.