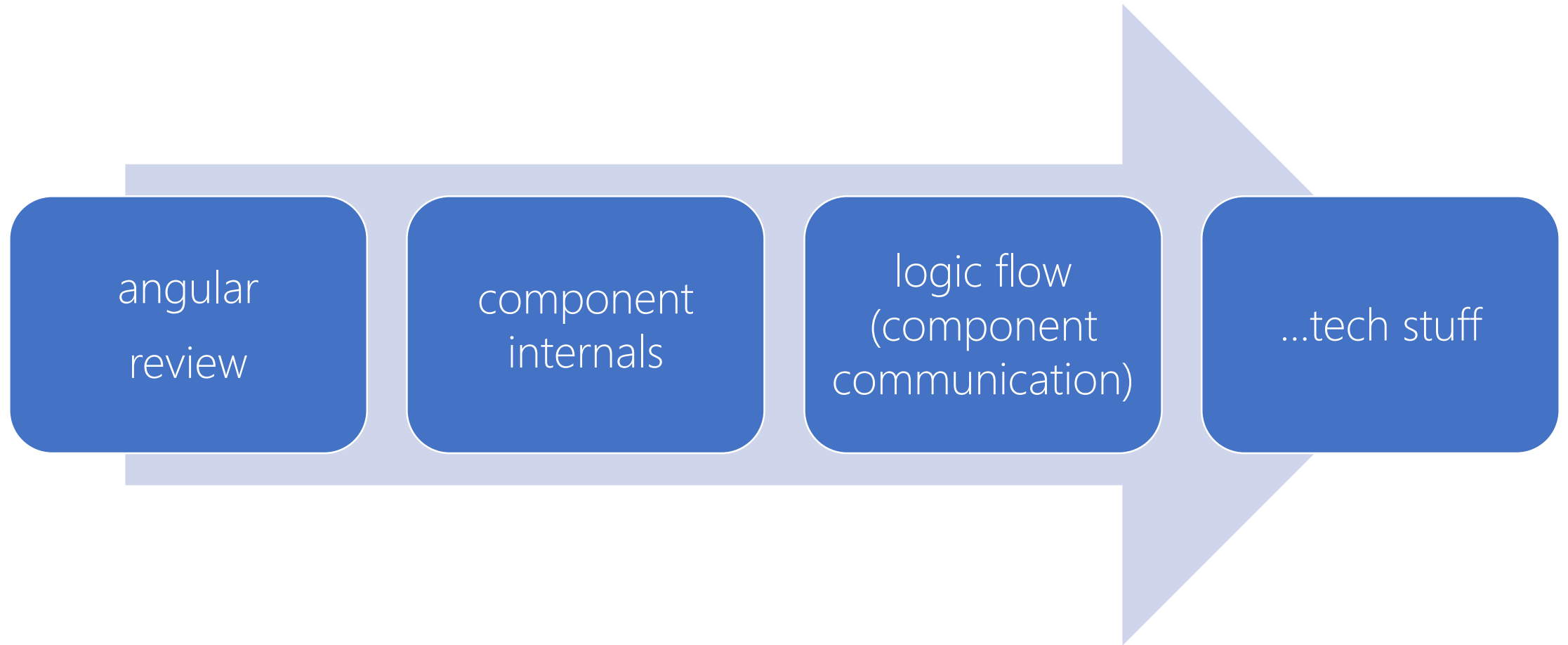




part 2

# plan



angular



# component(s)

everything is component...

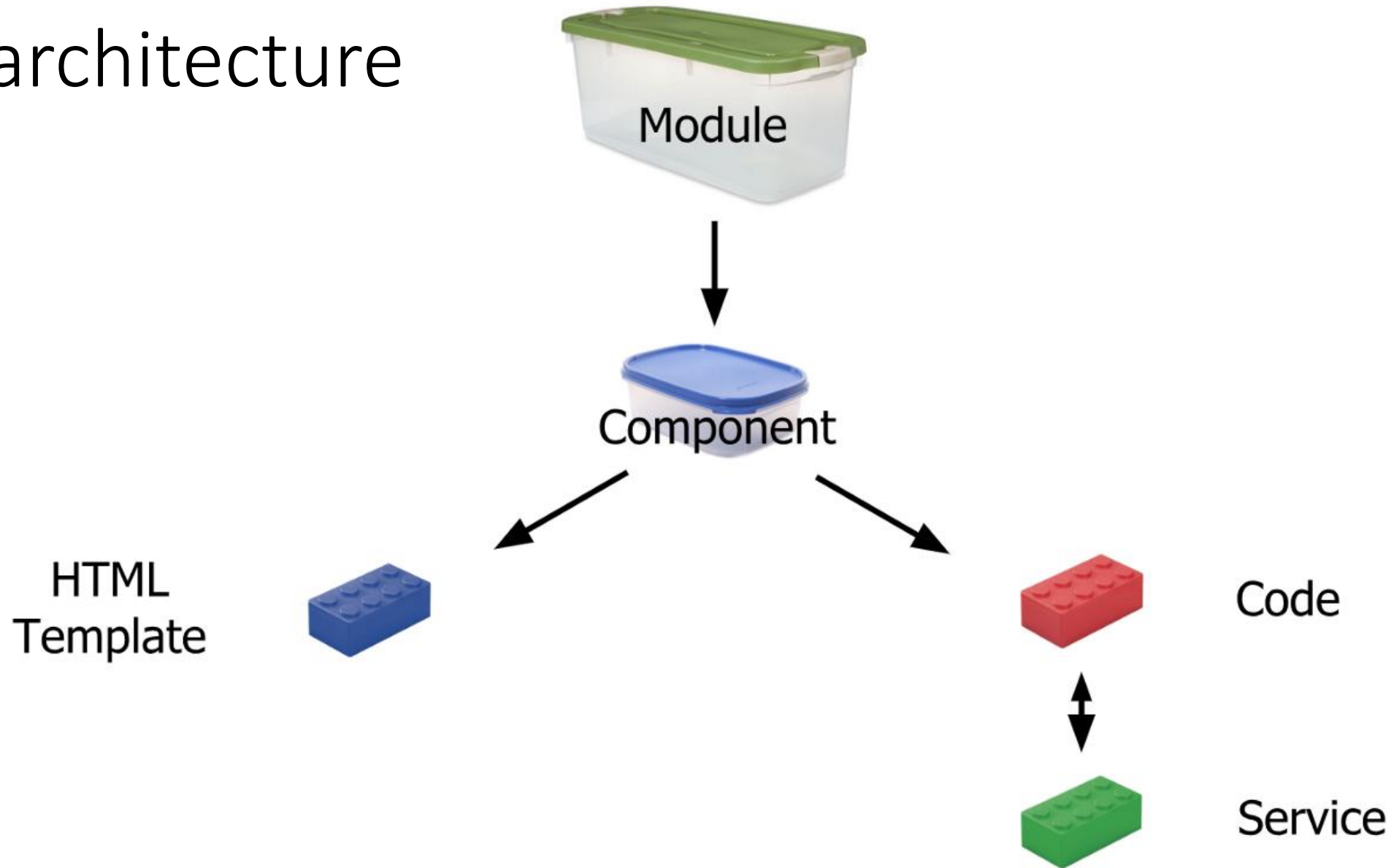
...web component

has structure, state & behaviour

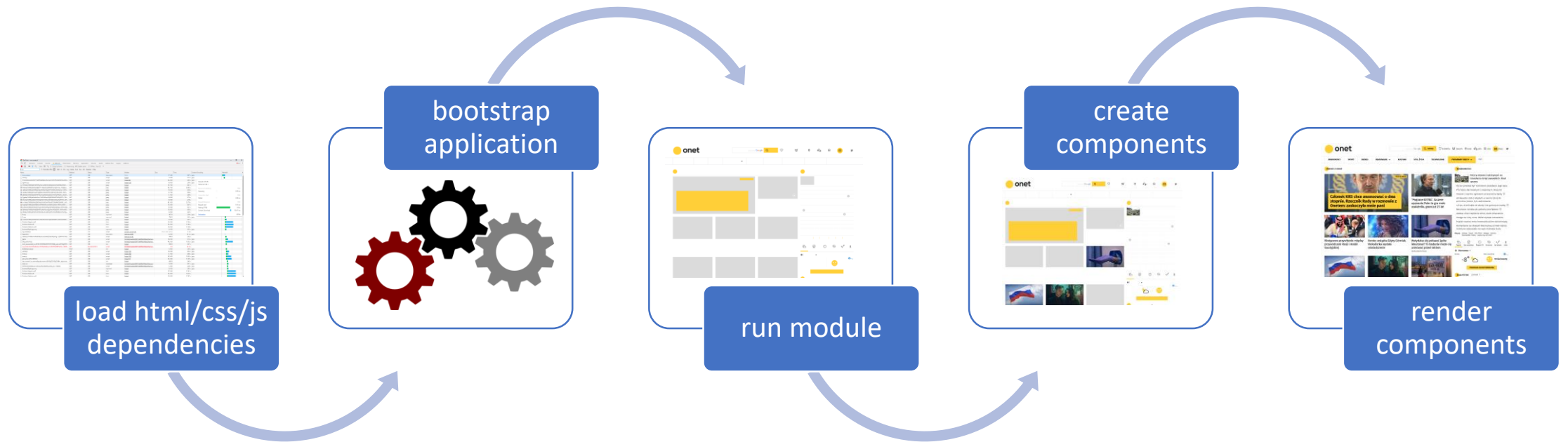
user can interact with...

it's rendered (visible?)

# architecture



# angular app start



structure

root HTML (host)

app entry point

angular cli config


NG-PB-APP

environments


 browserslist


 favicon.ico


 index.html


 karma.conf.js


 main.ts

 polyfills.ts

 styles.css

 test.ts


 tsconfig.app.json


 tsconfig.spec.json


 tslint.json

 .editorconfig


 .gitignore

 angular.json

 package-lock.json

 package.json

 README.md

 tsconfig.json

 tslint.json

# structure

## NG-PB-APP

- e2e
- node\_modules
- ▾ src

### app

app-routing.module.ts

app.component.css

app.component.html

app.component.spec.ts

app.component.ts

app.module.ts

▸ assets

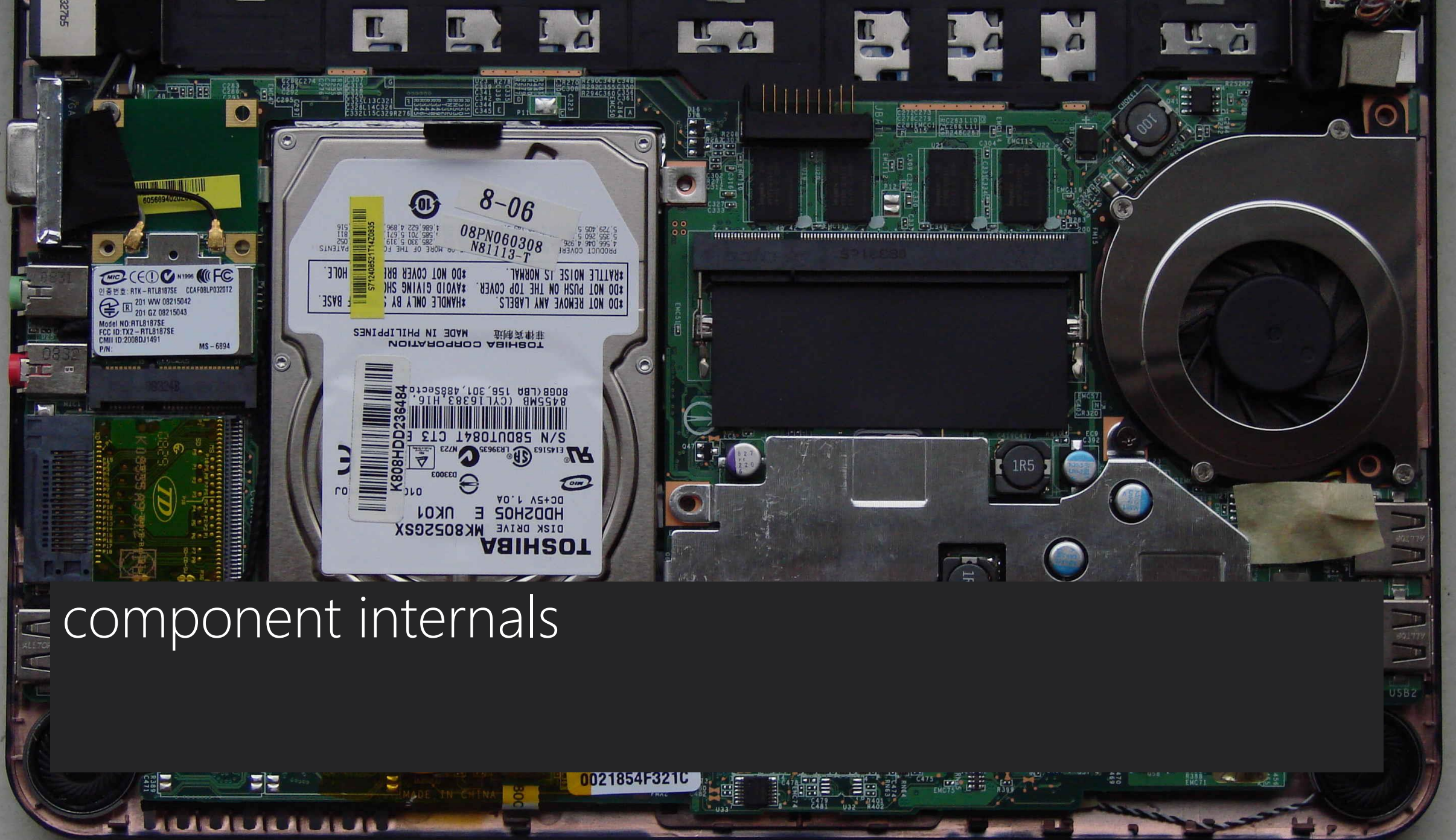
▸ environments

root component

main module

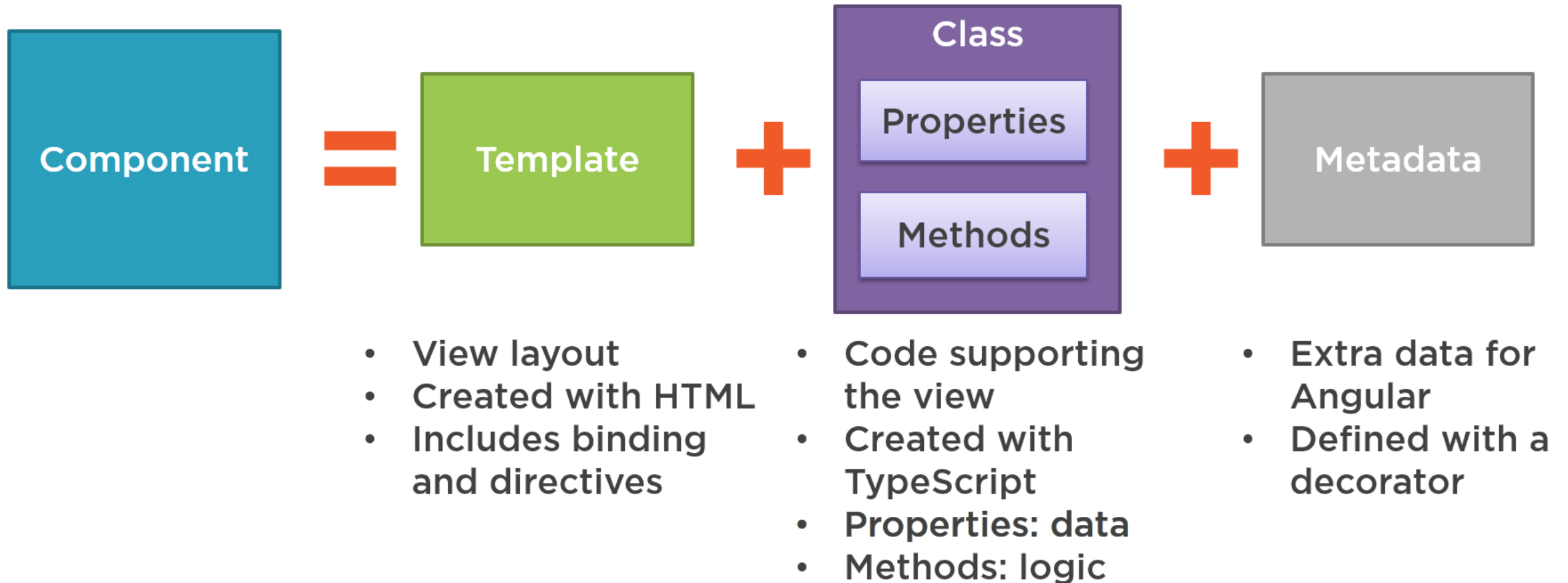


?



component internals

# What Is a Component?



# component fundamentals

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

component decorator

```
@Component({
```

```
  selector: 'app-test-item',
```

selector name

```
  templateUrl: './test-item.component.html',
```

template & style

```
  styleUrls: ['./test-item.component.css']
```

```
})
```

```
export class TestItemComponent implements OnInit {
```

```
  constructor() {}
```

```
  ngOnInit() {
```

component internals

```
  }
```

```
}
```

# component internals

model definition

template html

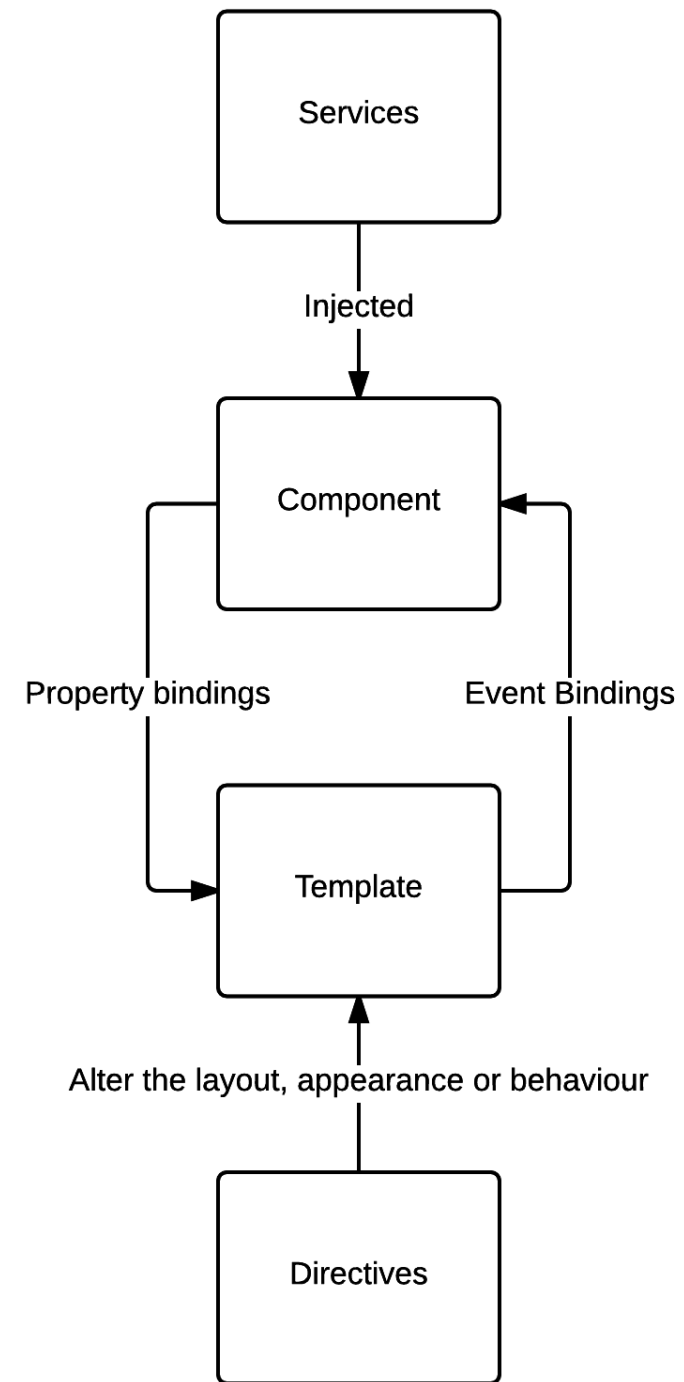
depending classes injection

expression interpolation {{}}

(input) property binding [\*]

output event binding (\*), \$event

build-in directives



# angular binding infrastructure

{{interpolation}}

[property  
/attribute  
binding]

class binding  
ngClass  
[class.name]

style binding  
ngStyle  
[style.name]

(event binding)



# Data Binding

DOM

Interpolation: `{{pageTitle}}`

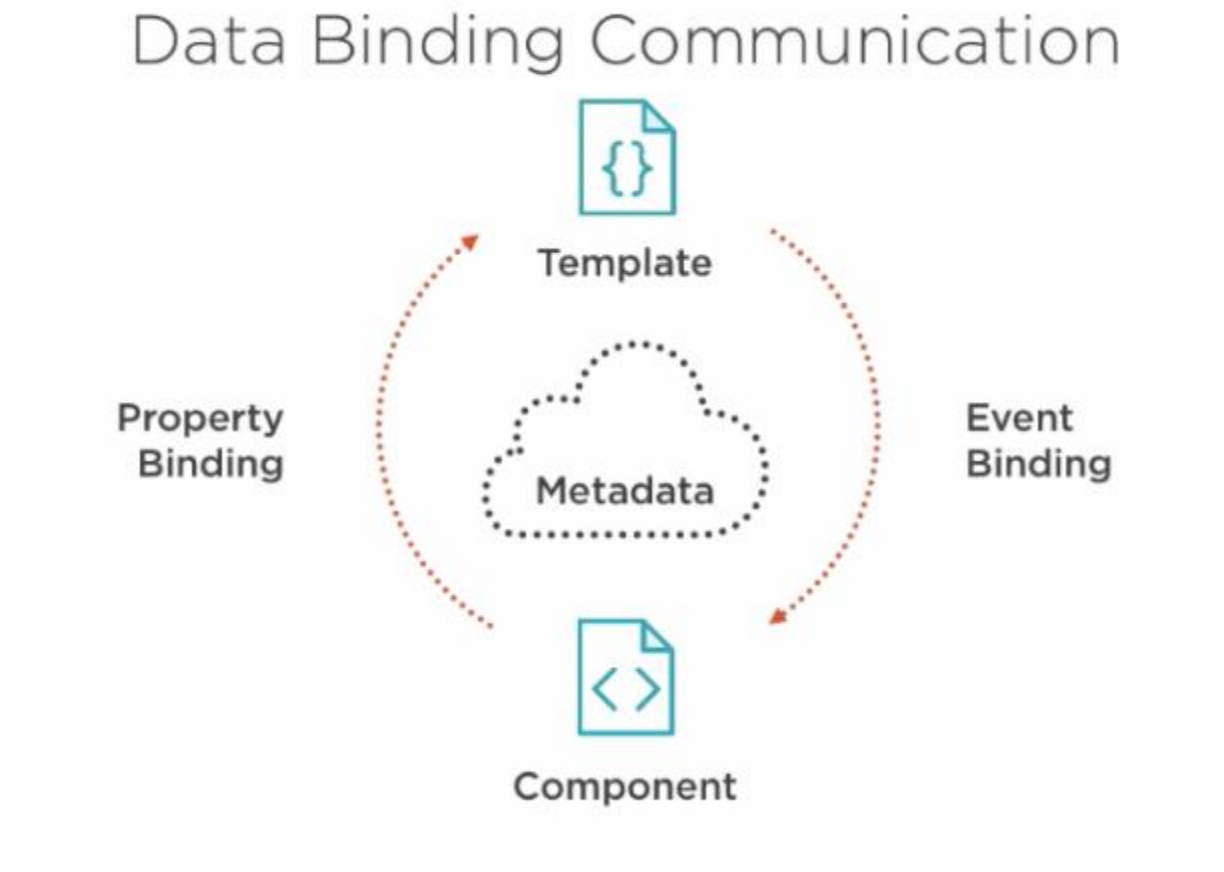
Property Binding: `<img [src]='product.imageUrl'>`

Event Binding: `<button (click)='toggleImage()'>`

Two-Way Binding: `<input [(ngModel)]='listFilter' />`

Component

# property binding





# component data flows



DOM

`{{expression}}`

Interpolation

`[property] = "expression"`

One Way Binding

`(event) = "statement"`

Event Binding

`[(ngModel)] = "property"`

Two Way Binding



Component

# component lifecycle

**ngOnChanges**      input property value changes

**ngOnInit**                      Initialization step

**ngDoCheck**                  every change detection cycle

**ngOnDestroy**                  before destruction

constructor

ngOnChanges

ngOnInit

ngDoCheck

ngAfterContentInit

ngAfterContentChecked

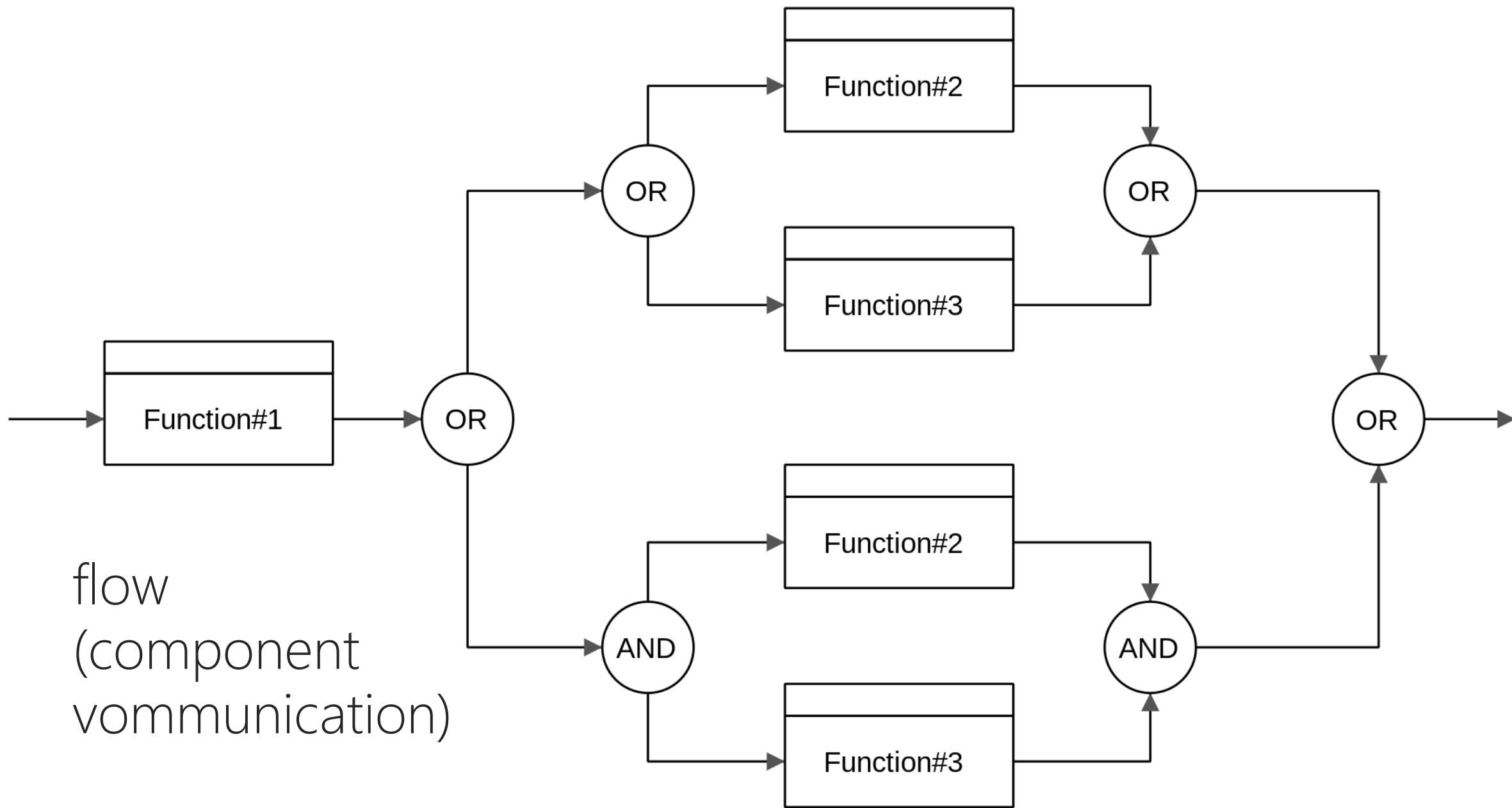
ngAfterViewInit

ngAfterViewChecked

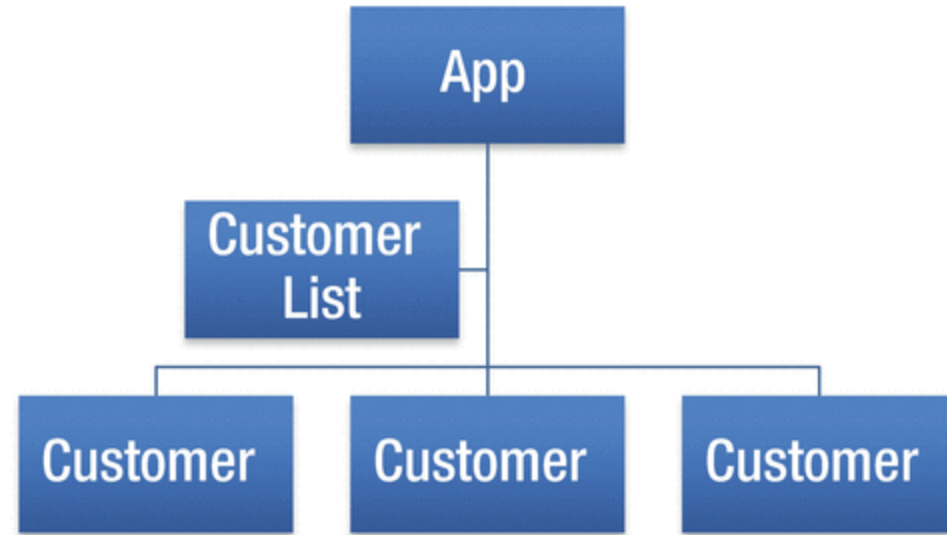
ngOnDestroy

# interfaces and methods

interface	method	description
OnChanges	ngOnChanges	Called when an input or output binding value changes
OnInit	ngOnInit	After the first ngOnChanges
DoCheck	ngDoCheck	Developer's custom change detection
AfterContentInit	ngAfterContentInit	After component content initialized
AfterContentChecked	ngAfterContentChecked	After every check of component content
AfterViewInit	ngAfterViewInit	After component's view(s) are initialized
AfterViewChecked	ngAfterViewChecked	After every check of a component's view(s)
OnDestroy	ngOnDestroy	Just before the directive is destroyed

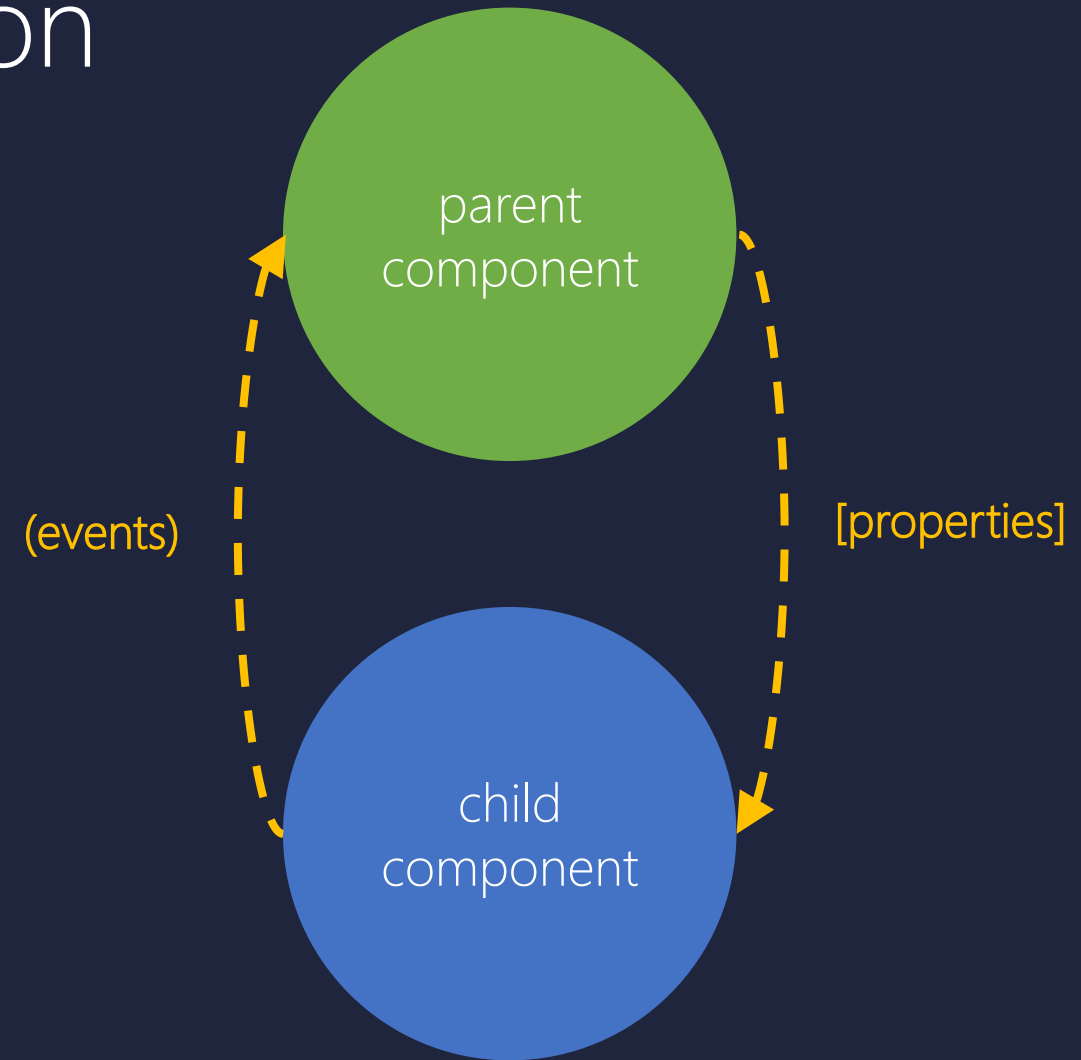


# hierarchy of components



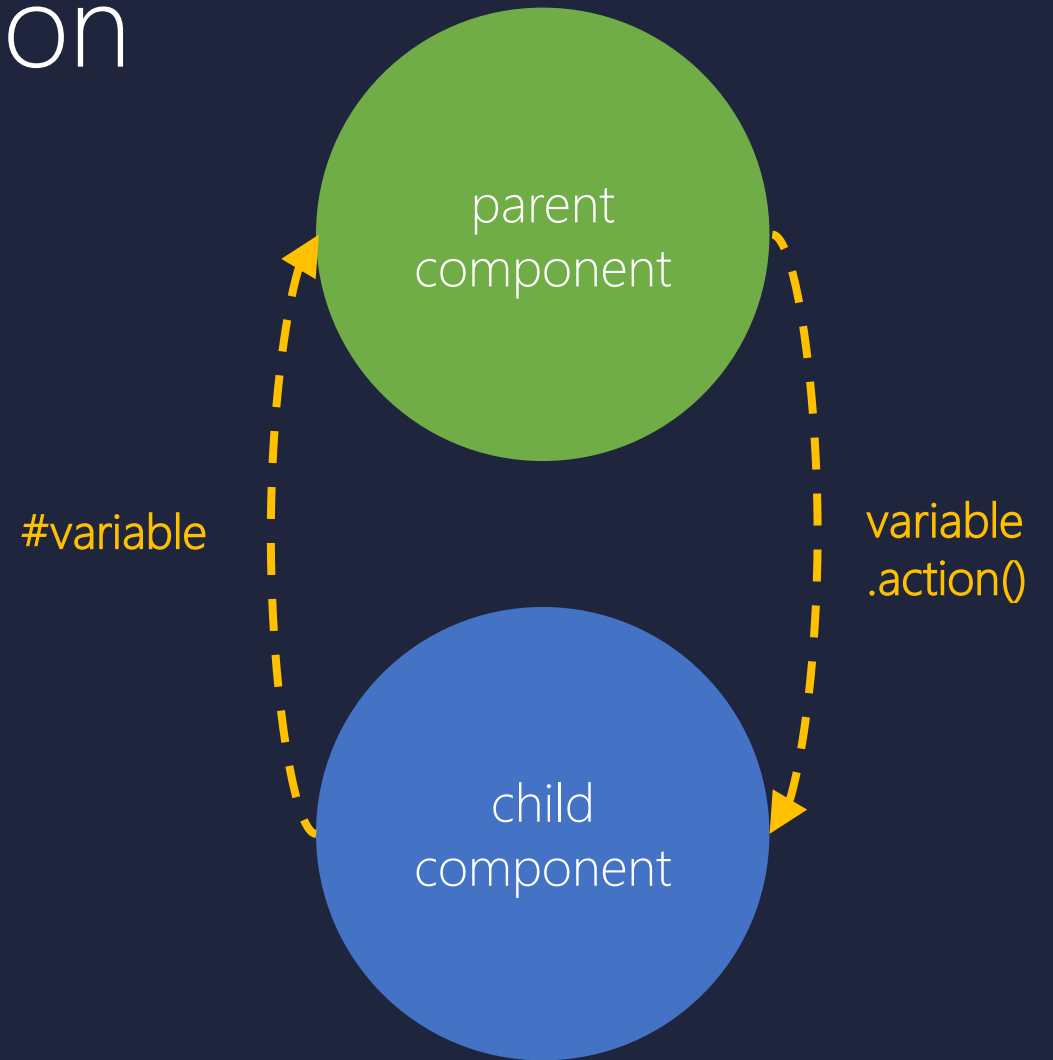
# component communication

- passing data via **properties**
- emitting custom **events**



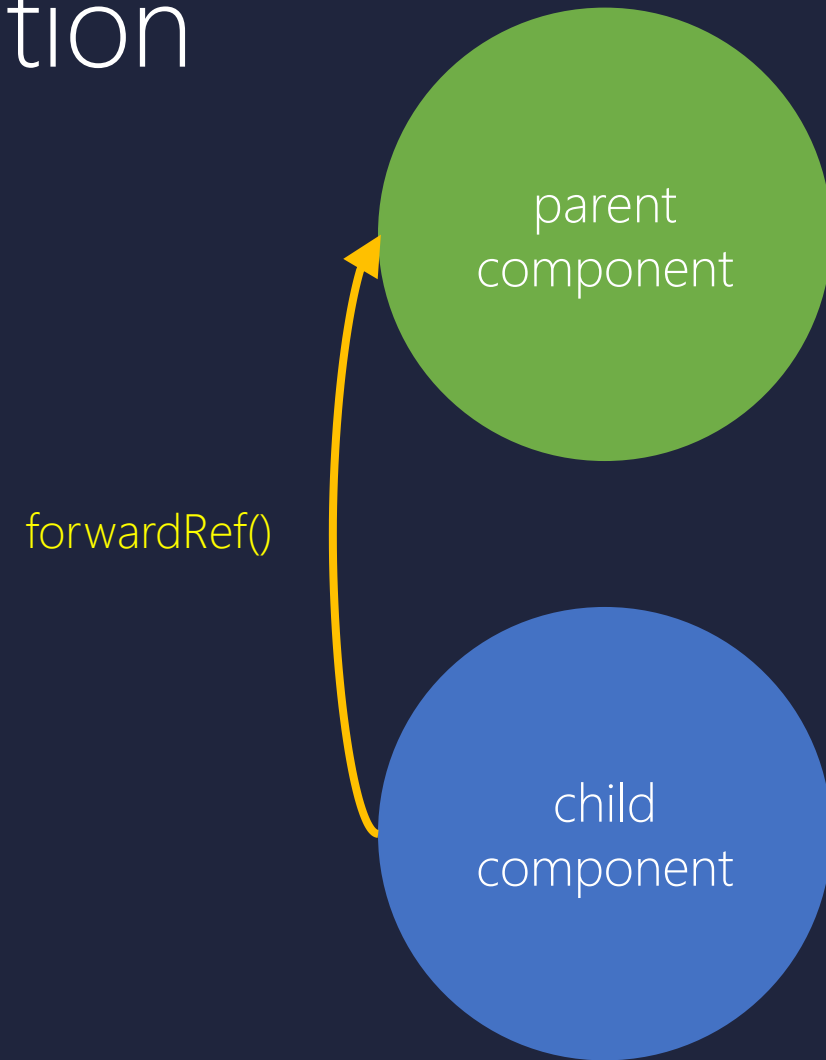
# component communication

- referencing with a local **variable**
- **querying** child components
  - @ViewChildren(...)
  - @ContentChildren(...)



# component communication

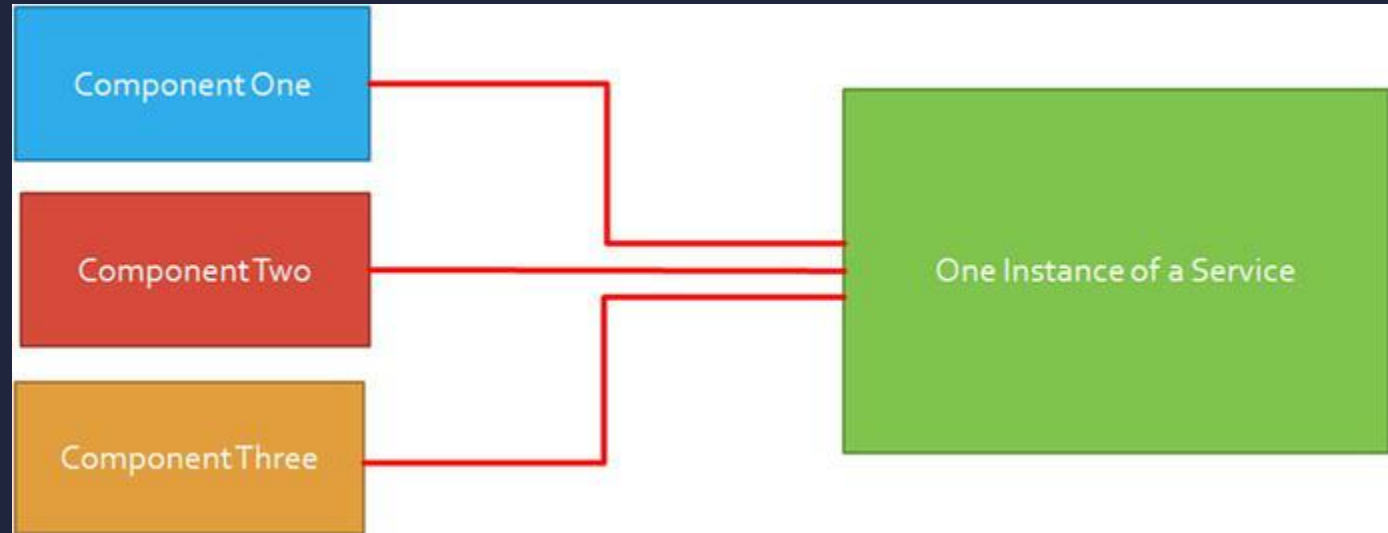
- dependency cycle with `forwardRef`





# component communication

- via **service**





<https://github.com/Banndzior>

#slack

kamil.mijacz@gmail.com

kamil.mijacz@softwarehut.com