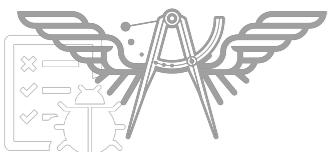


Getting start with Angular 7



Somkiat Puisungnoen

Search

Somkiat | Home

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro

Software Craftsmanship

Software Practitioner at สยามชัมนาภิกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · Public

Java and Bigdata





Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

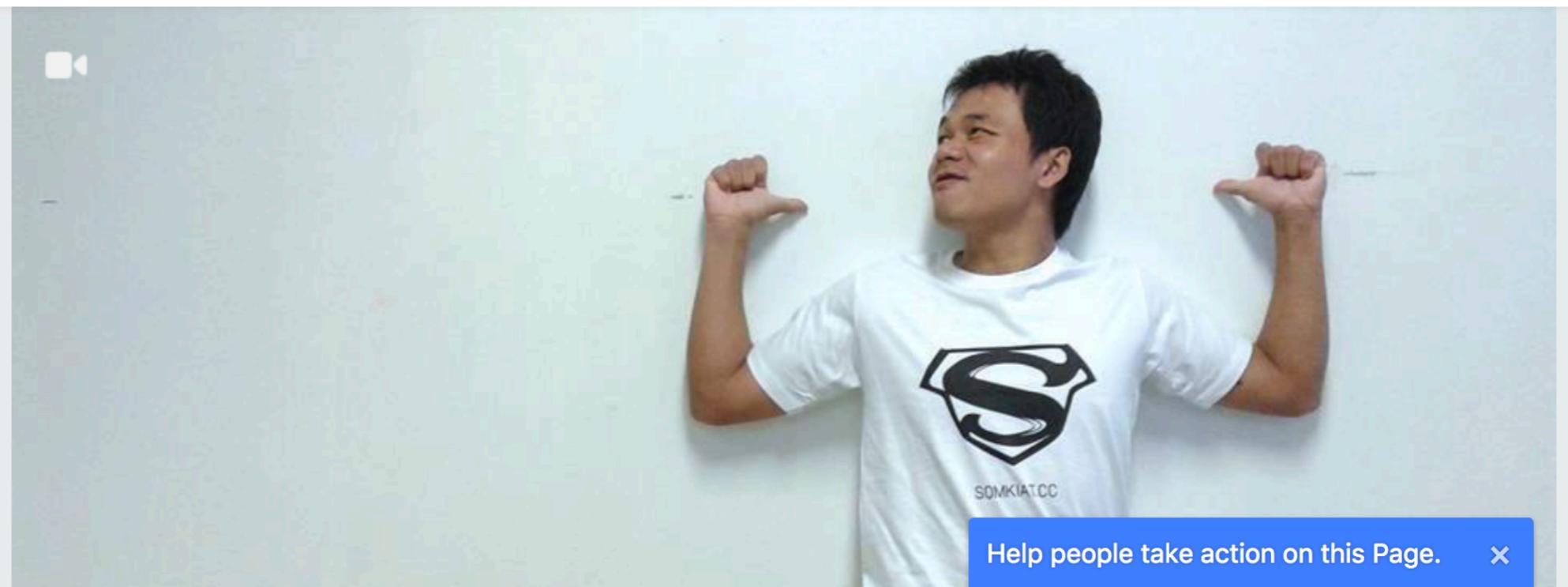
@somkiat.cc

Home

Posts

Videos

Photos



Agenda

- Introduction to Angular 7
- Installation and configuration
- Structure of Angular project
- Introduction to TypeScript
- Design and develop component/service
- Routing management
- Working with RESTful APIs



Angular 7



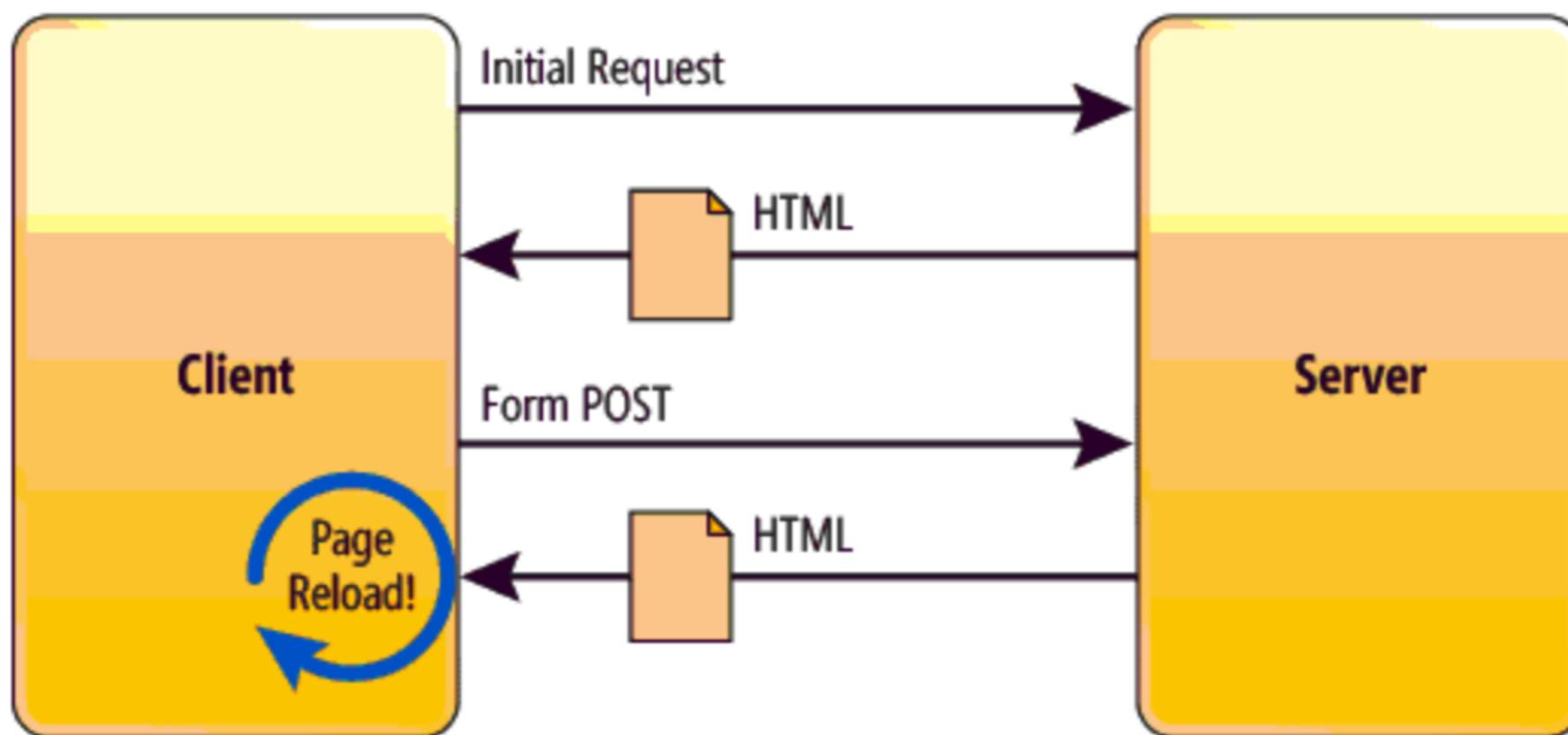
What is Angular ?

JavaScript framework with allows us to create
reactive Single Page Application (SPA)

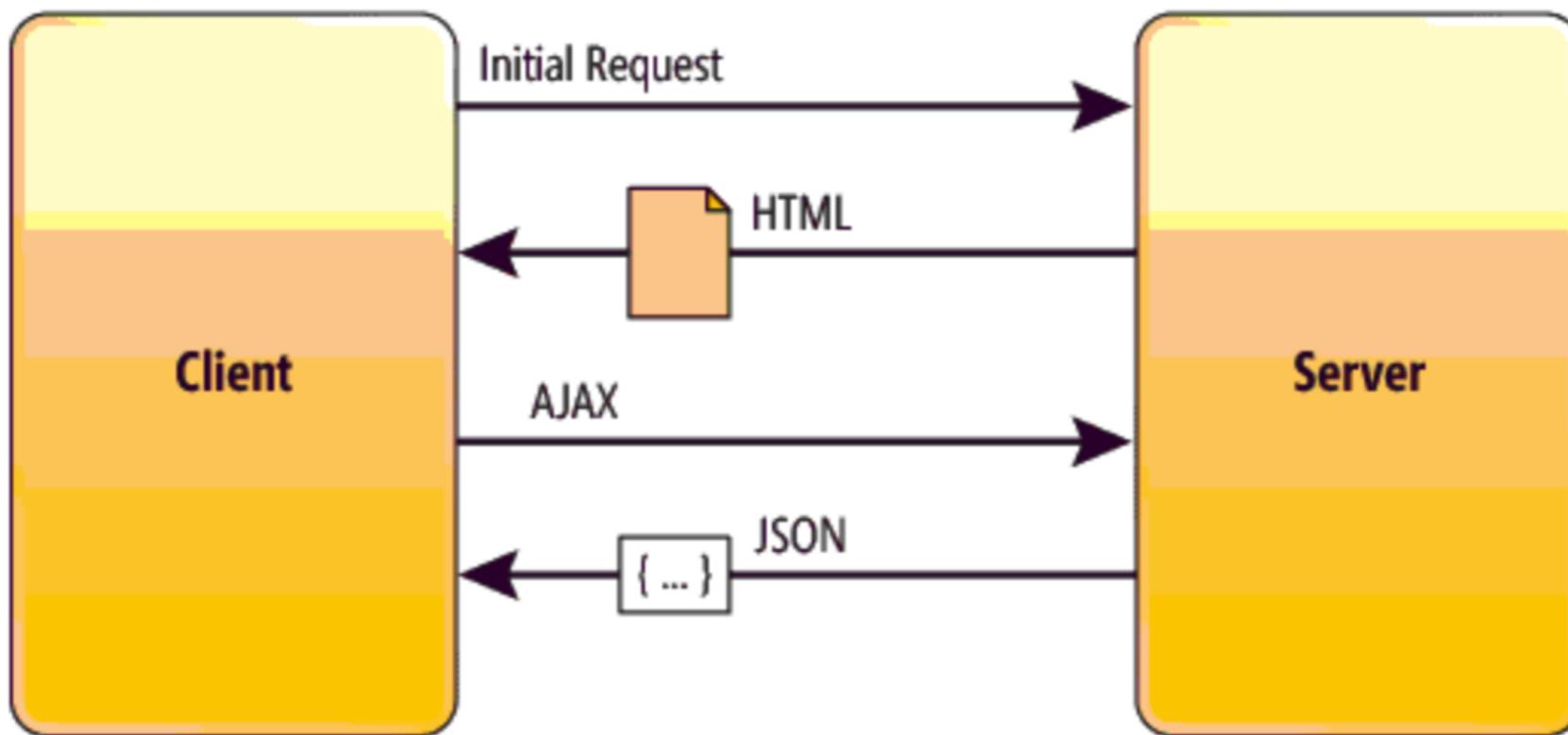
<https://angular.io/>



Traditional



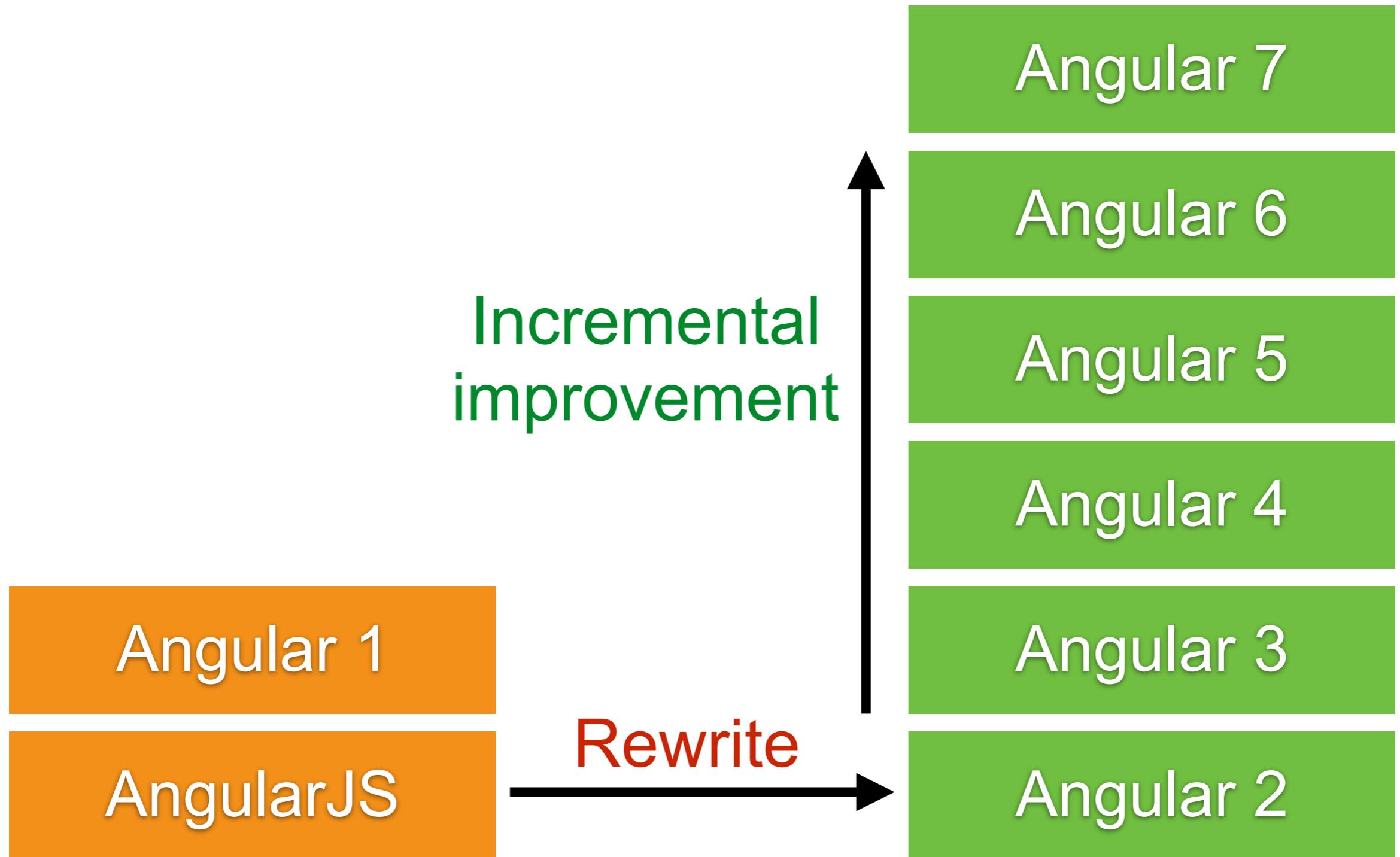
Single Page Application



Angular 7



Angular 7



Software requirement

Install NodeJS

Download for macOS (x64)

8.11.3 LTS

Recommended For Most Users

10.7.0 Current

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

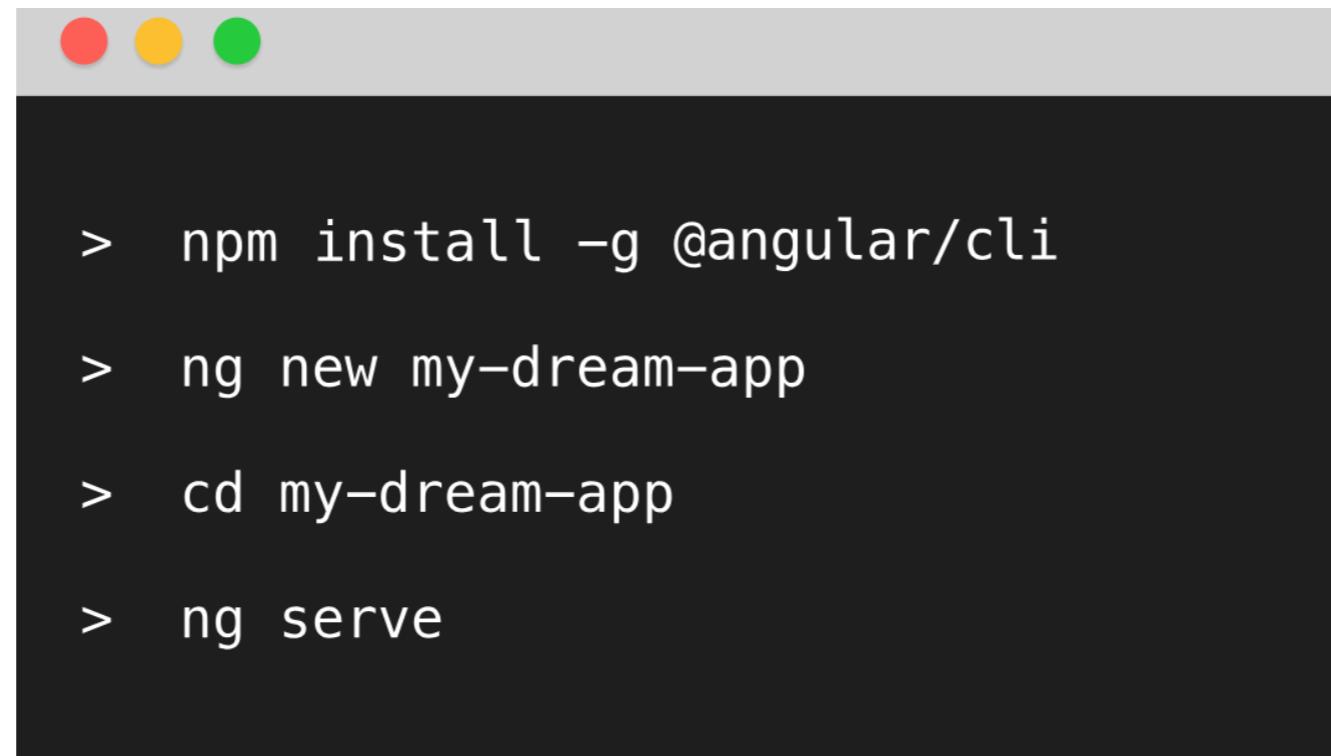
[Other Downloads](#) | [Changelog](#) | [API Docs](#)

<https://nodejs.org/en/>



Angular CLI

A tool to initialise, develop, scaffold and maintain
Angular application

A screenshot of a macOS terminal window. The title bar shows three colored window control buttons (red, yellow, green). The main window area is black with white text, displaying the following command sequence:

```
> npm install -g @angular/cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
```

<https://cli.angular.io/>



Install Angular CLI

```
$npm install -g @angular/cli@latest
```



Try after installed

\$ng version

```
Angular CLI: 7.3.5
Node: 11.11.0
OS: darwin x64
Angular:
...
Package          Version
-----
@angular-devkit/architect    0.13.5
@angular-devkit/core         7.3.5
@angular-devkit/schematics   7.3.5
@schematics/angular          7.3.5
@schematics/update           0.13.5
rxjs                         6.3.3
typescript                   3.2.4
```



Try to create first project

\$ng new hello-app

```
CREATE hello-app/README.md (1025 bytes)
CREATE hello-app/angular.json (3575 bytes)
CREATE hello-app/package.json (1313 bytes)
CREATE hello-app/tsconfig.json (384 bytes)
CREATE hello-app/tslint.json (2805 bytes)
CREATE hello-app/.editorconfig (245 bytes)
CREATE hello-app/.gitignore (503 bytes)
CREATE hello-app/src/environments/environment.prod.ts (51 bytes)
CREATE hello-app/src/environments/environment.ts (631 bytes)
CREATE hello-app/src/favicon.ico (5430 bytes)
CREATE hello-app/src/index.html (295 bytes)
```



Run your app

```
$cd hello-app  
$ng serve
```

** Angular Live Development Server is listening on localhost:4200, open your browser to <http://localhost:4200>

Date: 2018-07-19T17:42:59.617Z

Hash: cf107798cf25722bc556

Time: 22285ms

```
chunk {main} main.js, main.js.map (main) 10.7 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 227 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 5.22 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 15.6 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.06 MB [initial] [rendered]
i 「wdm」: Compiled successfully.
```



Open in browser

`http://localhost:4200/`

Welcome to app!

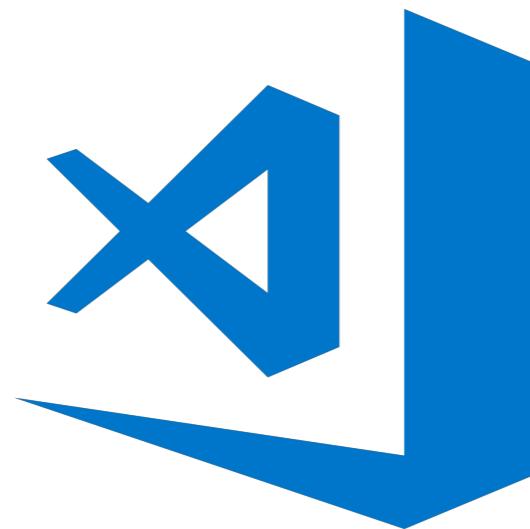


Here are some links to help you start:

- [Tour of Heroes](#)
- [CLI Documentation](#)
- [Angular blog](#)



Install Text Editor



What's new in Angular 7 ?



New in Angular 7

Step in Angular CLI

Bundle Budget features (limit size of app bundle)

Component Dev Kit (CDK)

Support TypeScript 3, RxJs 6.3



Bundle Budget features

Default value of budgets in `angular.json`

```
"budgets": [  
  {  
    "type": "initial",  
    "maximumWarning": "2mb",  
    "maximumError": "5mb"  
  }  
]
```



Angular Material Design

<https://material.angular.io/>

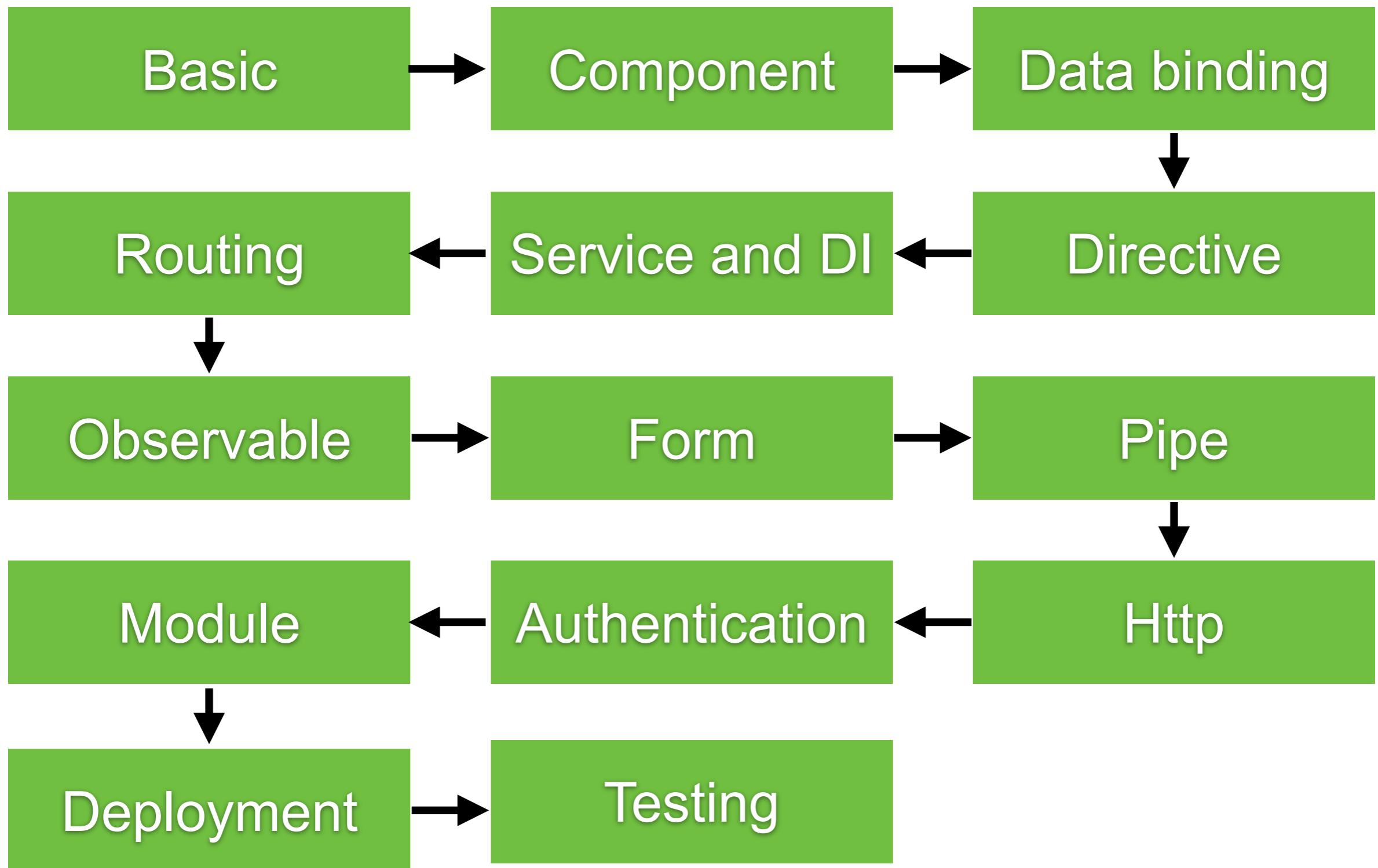
The screenshot shows the official Angular Material website. The header is blue and contains the 'Material' logo, navigation links for 'Components', 'CDK', and 'Guides', a version dropdown set to '7.3.4', and social links for 'Slack' and 'GitHub'. The main content area has a dark blue background. The title 'Angular Material' is displayed in large white font, followed by the subtitle 'Material Design components for Angular' in smaller white font. A prominent white button with the text 'Get started' is centered at the bottom of this section.



Learning path



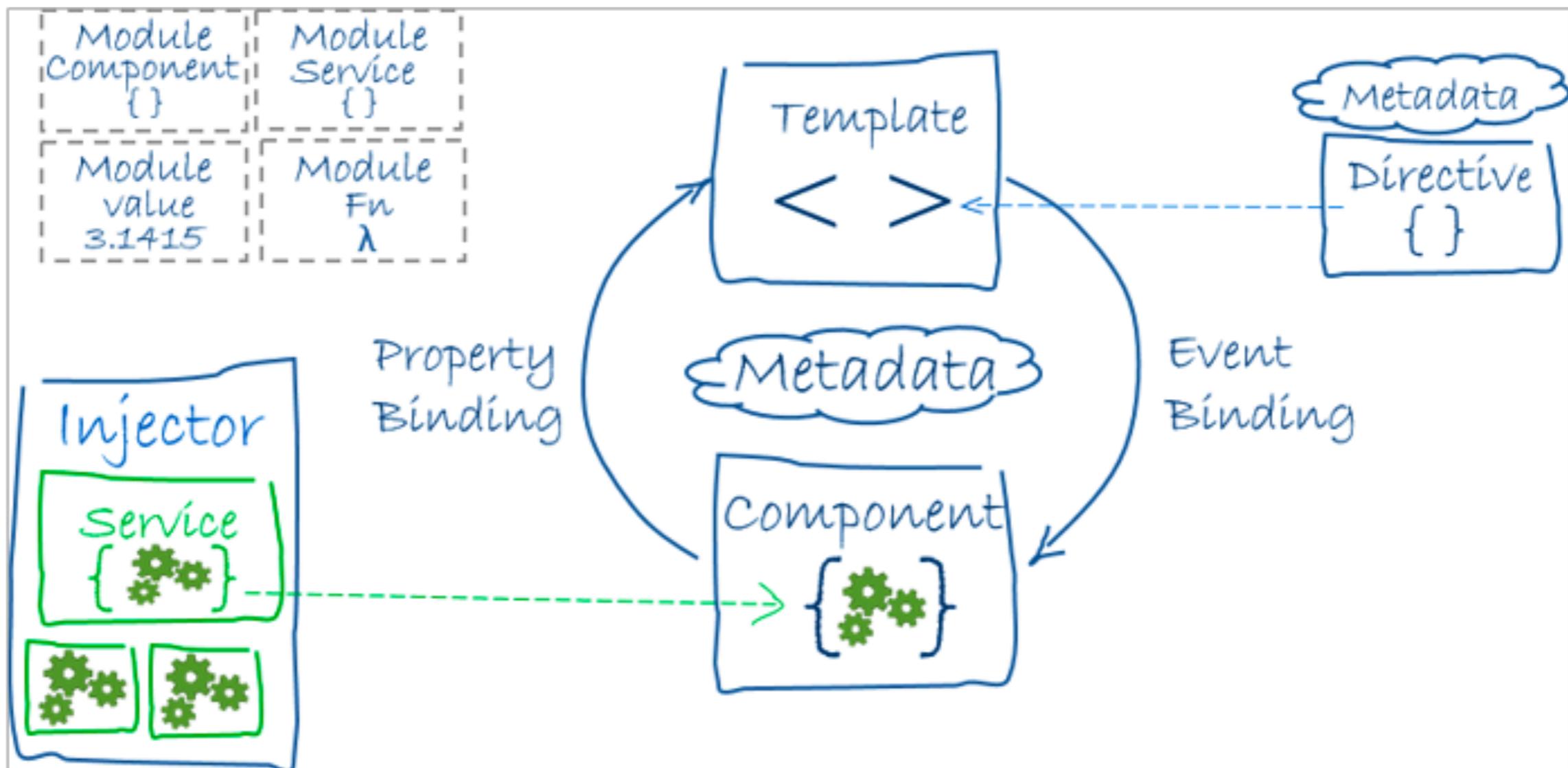
Learning path



Basic of Angular



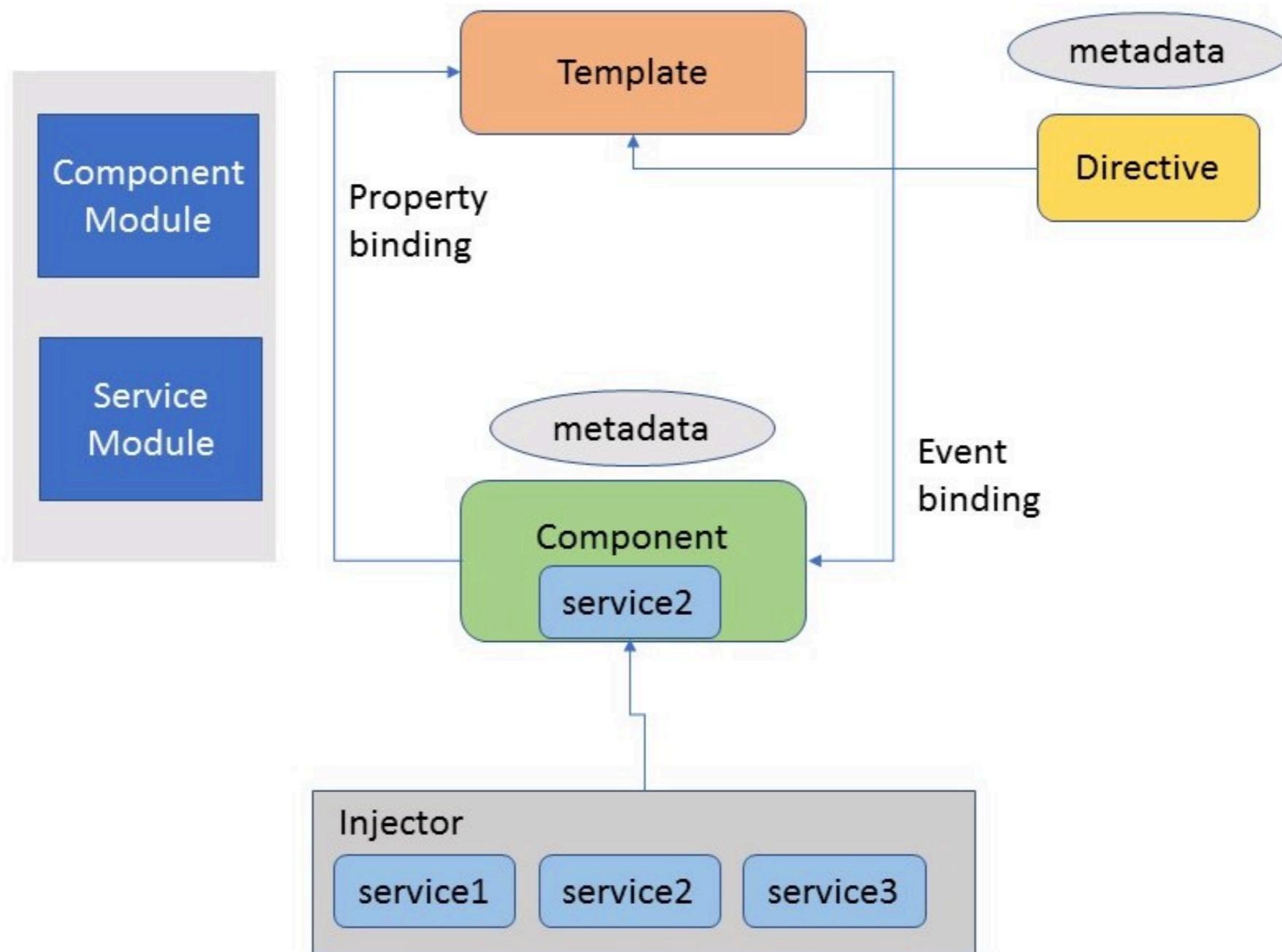
Angular Architecture



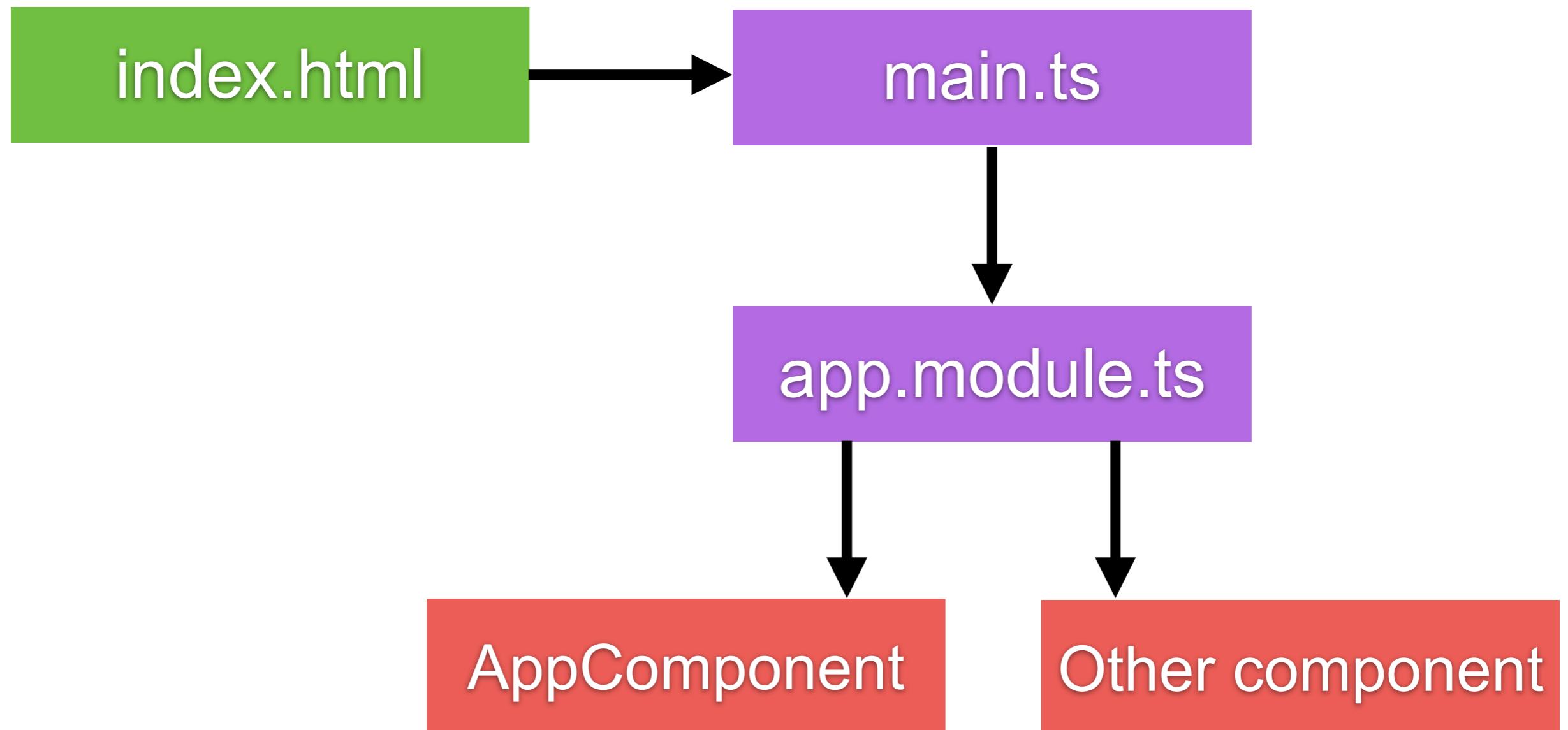
<https://angular.io/guide/architecture>



Angular Architecture



How Angular working ?

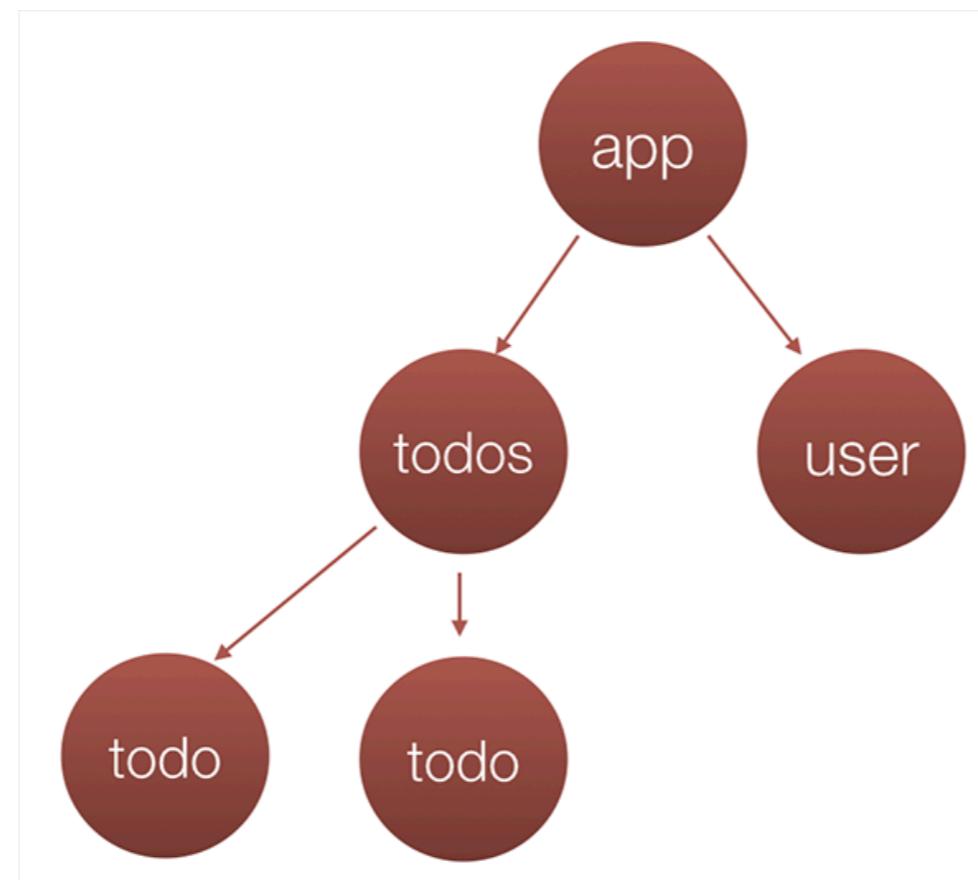


Components

Angular application is a tree of **Components**

Top level component is the application itself

Component is rendered by the browser



Components

Own template

Own style

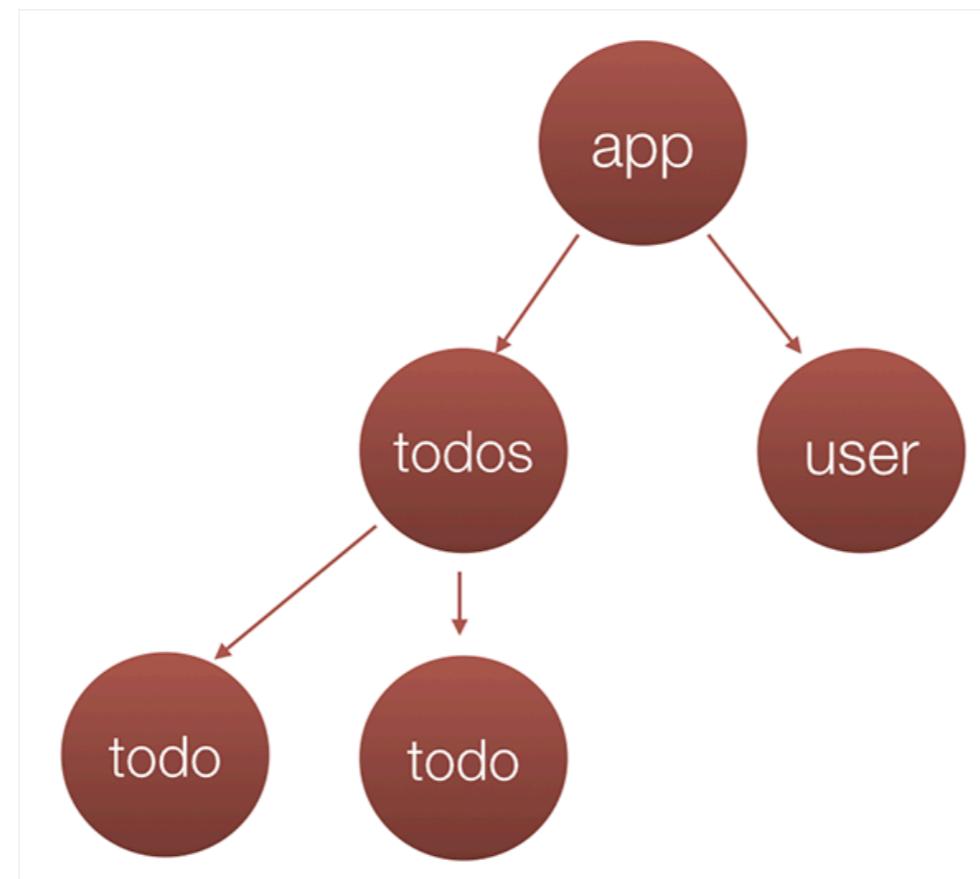
Own business logic

Split complex app to small component !!



Components

Composable
Reusable
Hierarchical



Design components

Home

About

Contact

Menu

Item 1

Item 2

Item 3



Header component

Home

About

Contact

Menu

Item 1

Item 2

Item 3



Sidebar/Menu component

Home

About

Contact

Menu

Item 1

Item 2

Item 3



Item component

Home

About

Contact

Menu

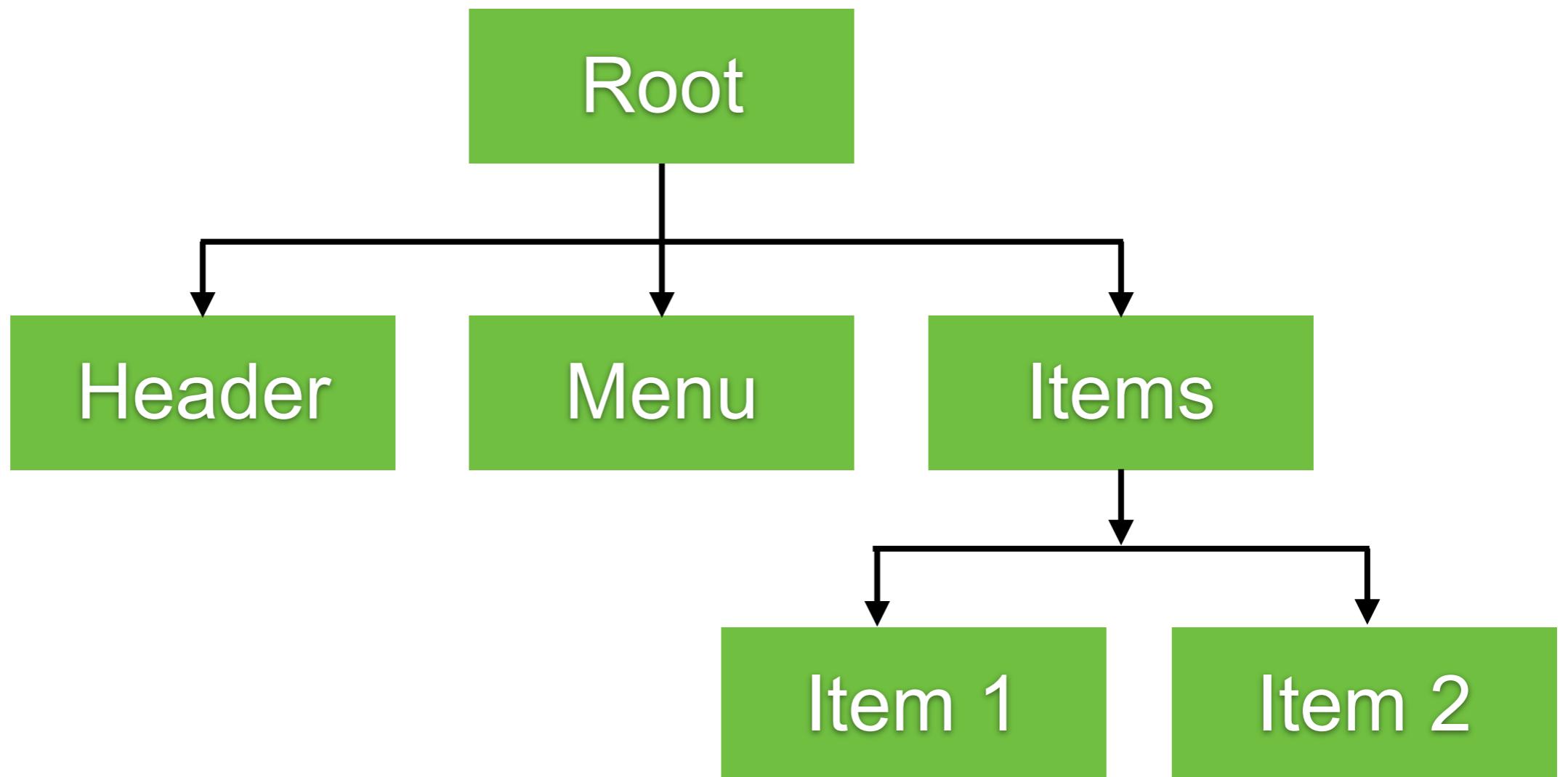
Item 1

Item 2

Item 3



Design components



Component

Template

Class

Metadata

View layout

Code support view

Extra data

Create with HTML

Create with TypeScript

Define decorator

Binding and directives

Data and logic

@Component



Component

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

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

Metadata

Template

```
export class AppComponent {  
  title = 'demo01';  
}
```



Template

Building template

Using component as a **directive**

Binding with **Interpolation**

Add logic with directive



Define template in component

Inline template
Link template

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

@Component({
  selector: 'app-root',
  template: `<h1>Welcome to {{ title }}!</h1>`,
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'hello';
}
```

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'hello';
}
```



Metadata

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  title = 'demo01';
}
```

Component decorator

Directive name used in HTML



Metadata

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

@Component({← Component decorator
  selector: 'app-root',← Directive name used in HTML
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})View layout , template and style
```

```
export class AppComponent {
  title = 'demo01';
}
```



Component

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

```
export class AppComponent {
  title = 'demo01';
}
```

Class



Component

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

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

Import module/library

```
export class AppComponent {  
  title = 'demo01';  
}
```



Import ?

3-party libraries

ES modules

Angular modules



Angular is Modular

@angular/
core

@angular/
http

@angular/
router

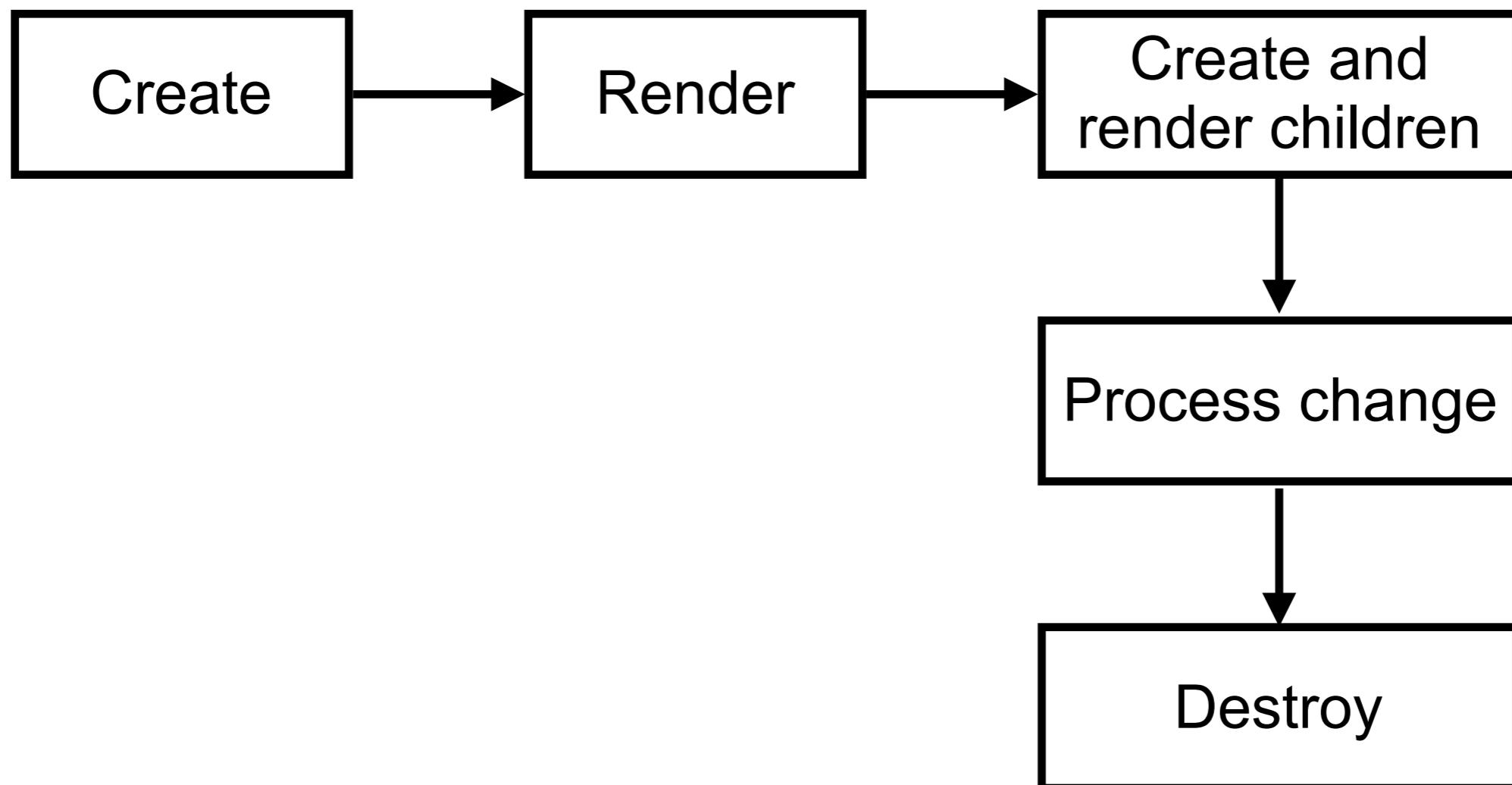


Create new component with CLI

```
$ng generate component <name>
```



Component life cycle



Component life cycle hooks

Method	Description
OnInit()	Perform component initialization Retrive data
OnChanges()	Perform action after properties are changed
OnDestroy()	Perform cleanup

<https://angular.io/guide/lifecycle-hooks>



Using life cycle hook

1. Import hook from @angular core

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

```
export class ProductListComponent implements OnInit {
```

```
  constructor() {  
    console.log(this.products);  
  }
```

2. Implement Hooks interface

```
  ngOnInit(): void {  
    console.log('Called ngOnInit');  
  }
```

```
}
```

3. Implement methods from hook



Workshop

<https://github.com/up1/workshop-angular-note>



1. Create project

\$ng new note



Using global style

Edit file style.css

<http://bit.ly/2CjEHnc>



2. Create Note component

\$ng generate component note

```
CREATE  src/app/note/note.component.css (0 bytes)
CREATE  src/app/note/note.component.html (23 bytes)
CREATE  src/app/note/note.component.spec.ts (614 bytes)
CREATE  src/app/note/note.component.ts (261 bytes)
UPDATE  src/app/app.module.ts (388 bytes)
```



Create Note component in main

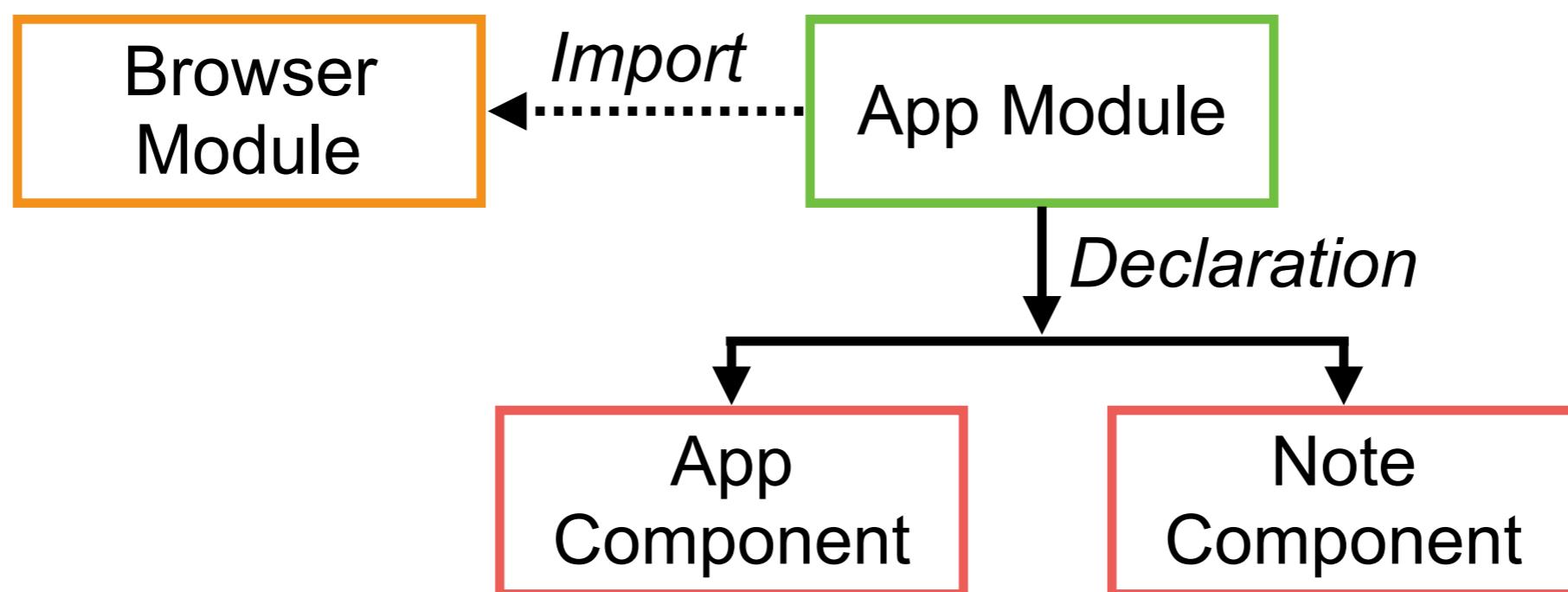
```
$ng generate component note --flat
```

```
CREATE  src/app/note.component.css (0 bytes)
CREATE  src/app/note.component.html (23 bytes)
CREATE  src/app/note.component.spec.ts (614 bytes)
CREATE  src/app/note.component.ts (261 bytes)
UPDATE  src/app/app.module.ts (457 bytes)
```



Add Note component

Edit file /app/app.module.ts



Note component

/app/note/note.component.ts

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

@Component({
  selector: 'app-note',
  templateUrl: './note.component.html',
  styleUrls: ['./note.component.css']
})
export class NoteComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

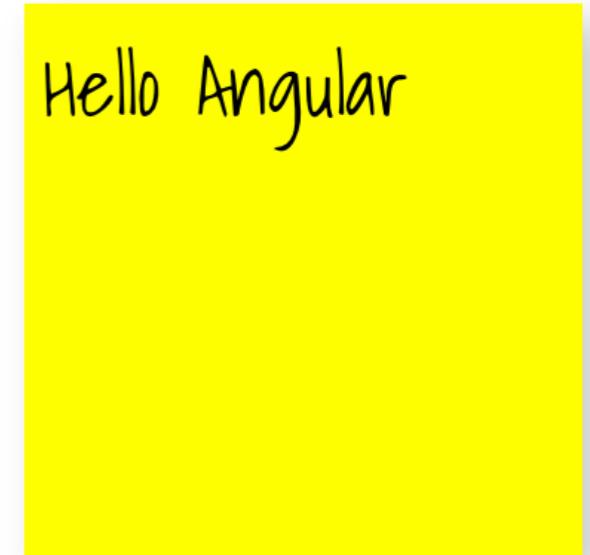
}
```



Note component template

/app/note/note.component.html

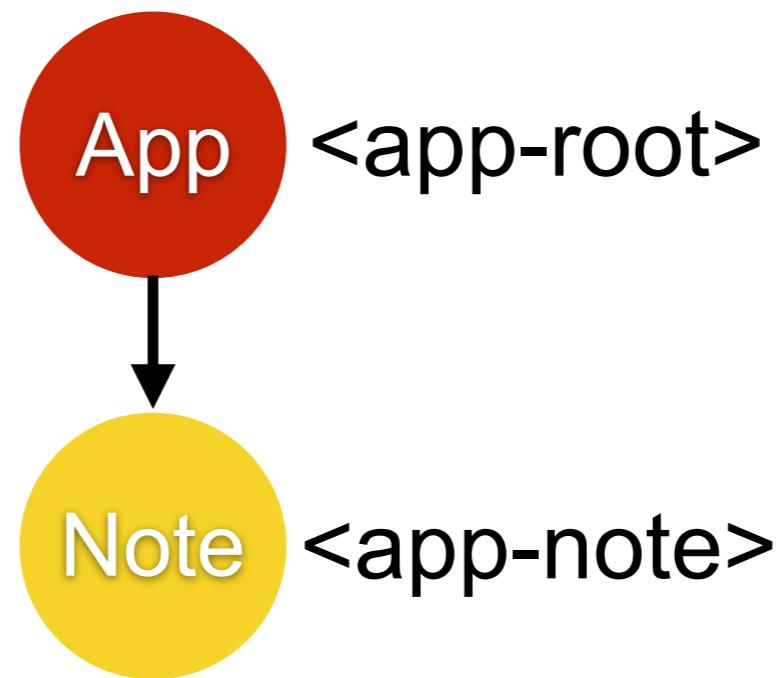
```
<div class="note">
  <p>Hello Angular</p>
  <span>
    <button id="edit">Edit</button>
    <button id="remove">Remove</button>
  </span>
</div>
```



Use note component in App

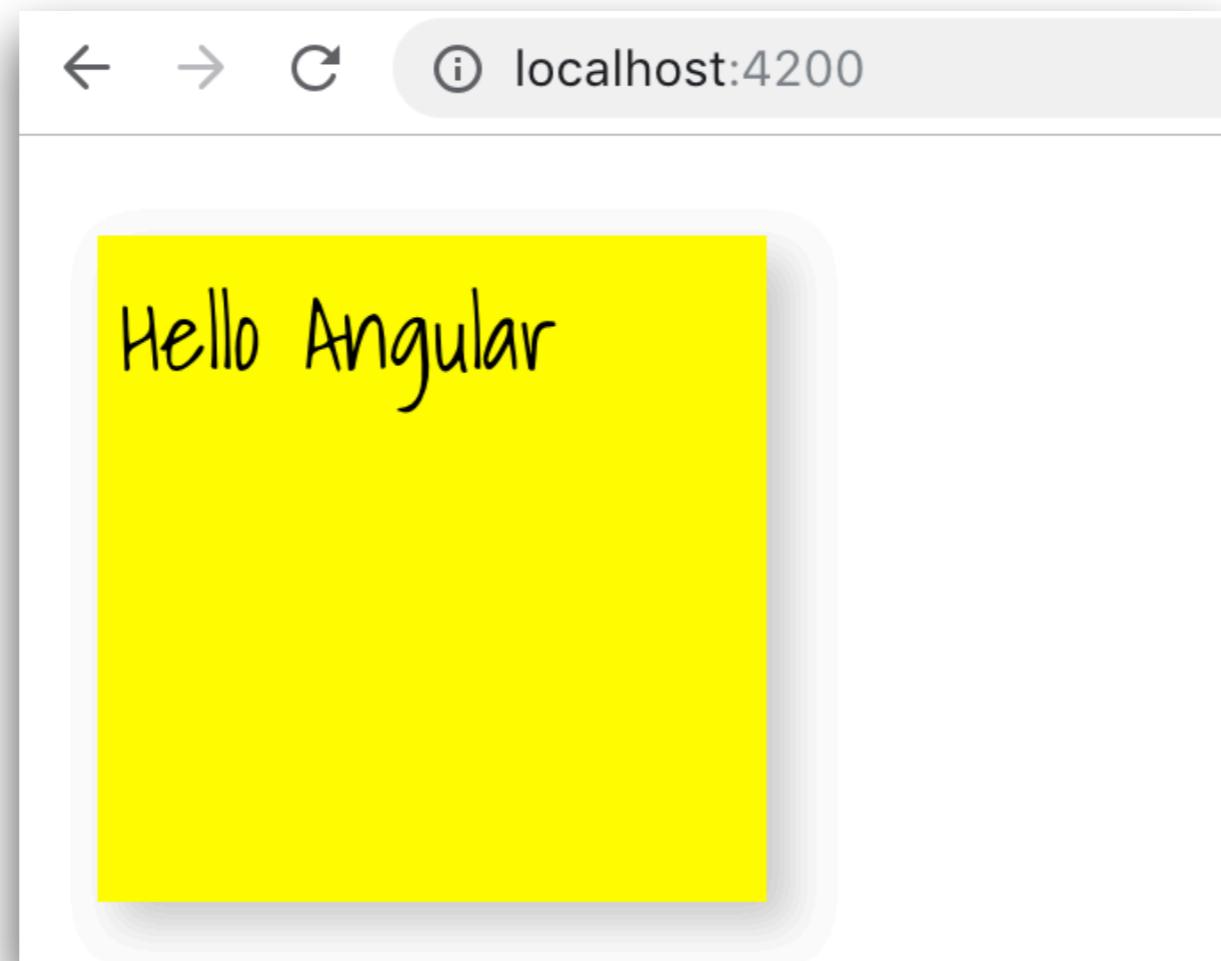
/app/app.component.html

```
<div>
  <app-note></app-note>
</div>
```

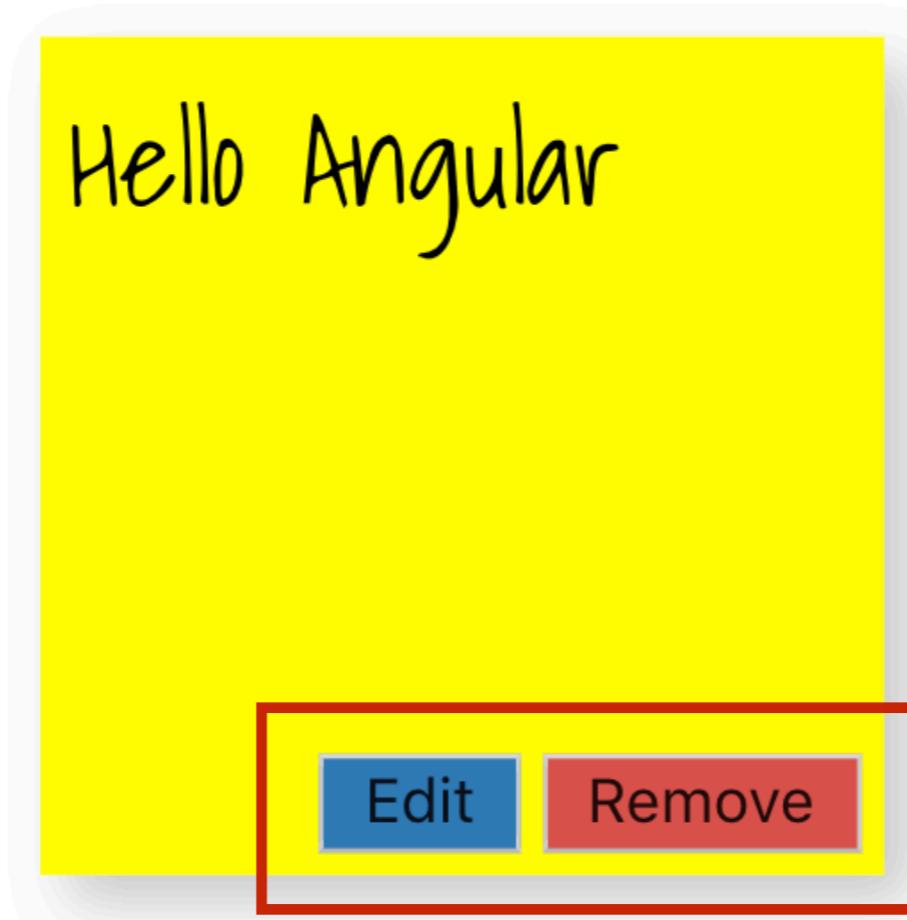


3. Run project

\$ng serve



4. Handling events



4. Handling events (Event binding)

Click on Edit and Remove button

```
<div class="note">
  <p>Hello Angular</p>
  <span>
    <button id="edit" (click)="edit()">
      Edit
    </button>
    <button id="remove" (click)="remove()">
      Remove
    </button>
  </span>
</div>
```

<https://angular.io/guide/user-input>



4. Handling events (Event binding)

Call function in NoteComponent

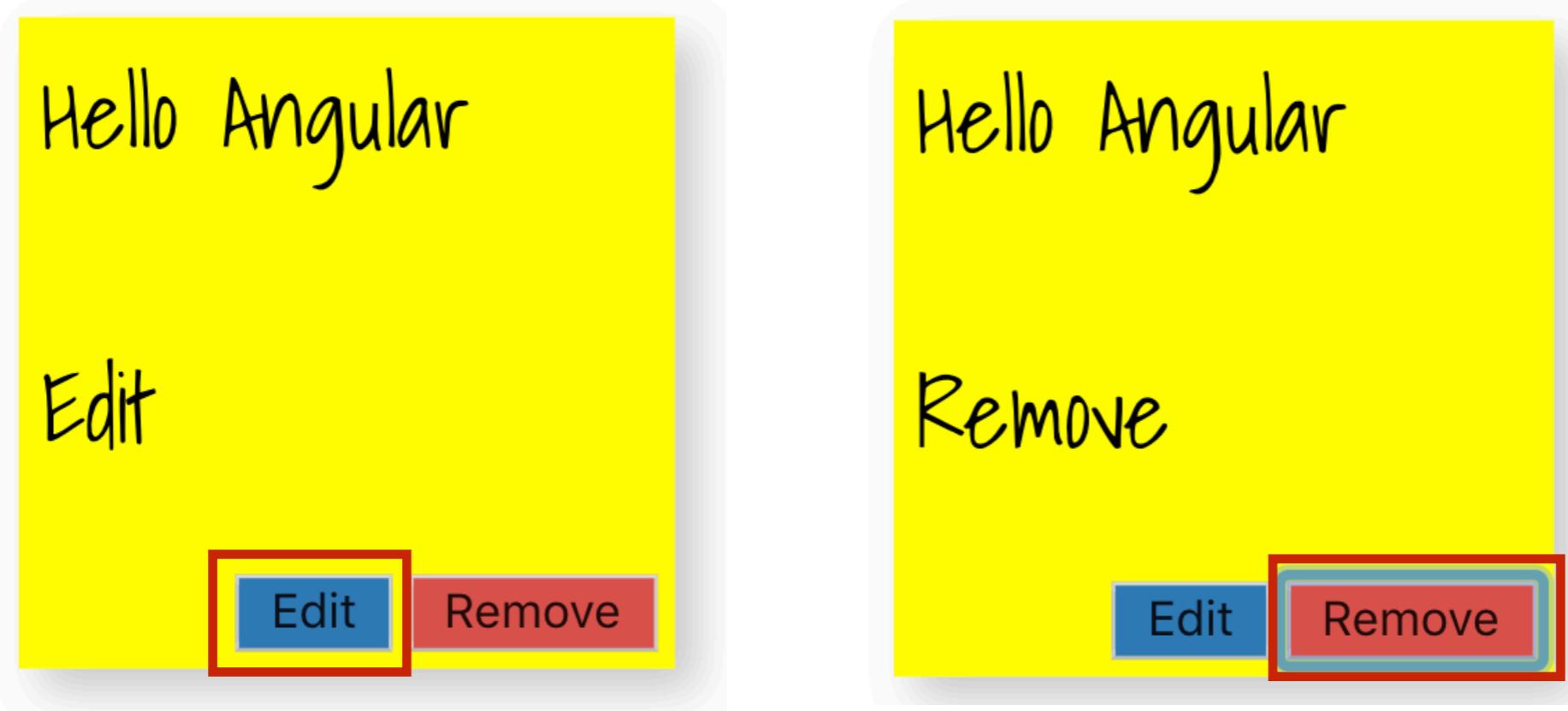
```
export class NoteComponent implements OnInit {  
  constructor() { }  
  
  ngOnInit() {  
    }  
  
    edit() {  
    }  
  
    remove() {  
    }  
}
```

<https://angular.io/guide/user-input>



Show message when click button

Call function in NoteComponent



Add property into NoteComponent

Property name = userEvent : string

```
export class NoteComponent implements OnInit {  
  userEvent: string;  
  
  edit() {  
    this.userEvent = 'Edit';  
  }  
  
  remove() {  
    this.userEvent = 'Remove';  
  }  
}
```



Show data in template

Edit file note.component.html

```
<div class="note">
  <p>Hello Angular</p>

  <p>{{userEvent}}</p>

<span>
  <button id="edit" (click)="edit()">Edit</button>
  <button id="remove" (click)="remove()">Remove</button>
</span>
</div>
```



5. Add state to component

isEdit = true | false

isEdit = false



isEdit = true



Add state to component

Create state in class

```
export class NoteComponent implements OnInit {  
  isEdit = false;  
}
```



State change when click Edit

Implement in edit()

```
export class NoteComponent implements OnInit {  
  isEdit = false;  
  
  edit() {  
    this.isEdit = true;  
  }  
}
```



Display UI in each state

Edit file note.component.html by using *ngIf

```
<div class="note" *ngIf="!isEdit"> isEdit = false
  <p>Hello Angular</p>
  <span>
    <button id="edit" (click)="edit()">Edit</button>
    <button id="remove" (click)="remove()">Remove</button>
  </span>
</div>
```

```
<div class="note" *ngIf="isEdit"> isEdit = true
  <p><textarea></textarea></p>
  <span>
    <button id="save" (click)="save()">Save</button>
  </span>
</div>
```

<https://angular.io/api/common/NgIf>



State change when click Save

Implement in save()

```
export class NoteComponent implements OnInit {  
  isEdit = false;  
  
  save() {  
    this.isEdit = false;  
  }  
}
```

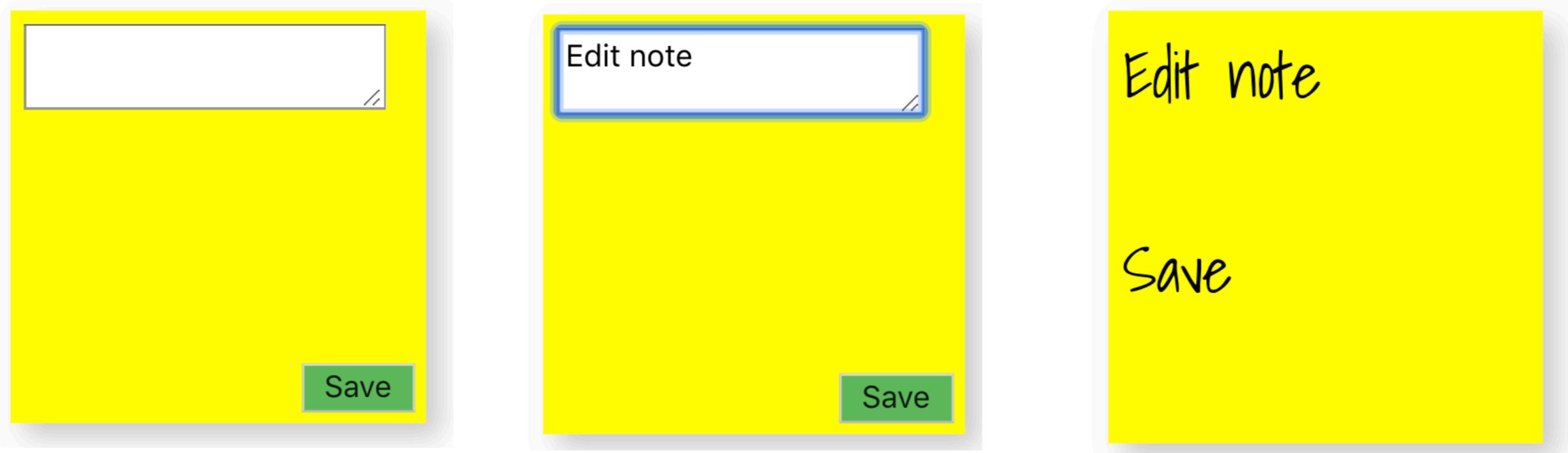


6. Save data from textarea



6. Save data from textarea

Working with Angular Forms



<https://angular.io/guide/forms>



Binding data from textarea

Edit file note.component.html by using template variables (#)

```
<div class="note" *ngIf="isEdit">
```

Template variable

```
    <p><textarea #newTitle></textarea></p>
```

```
<span>
```

```
    <button id="save"
```

Save

```
    </button>
```

```
</span>
```

```
</div>
```

```
(click)="save(newTitle.value)">
```

Get value from variable



Create property and function

Edit in save() and keep data of new title

```
export class NoteComponent implements OnInit {  
  
    isEdit = false;  
    title = 'Hello Angular';  
  
    save(newTitle) {  
        this.title = newTitle;  
        this.isEdit = false;  
    }  
}
```



Display new title

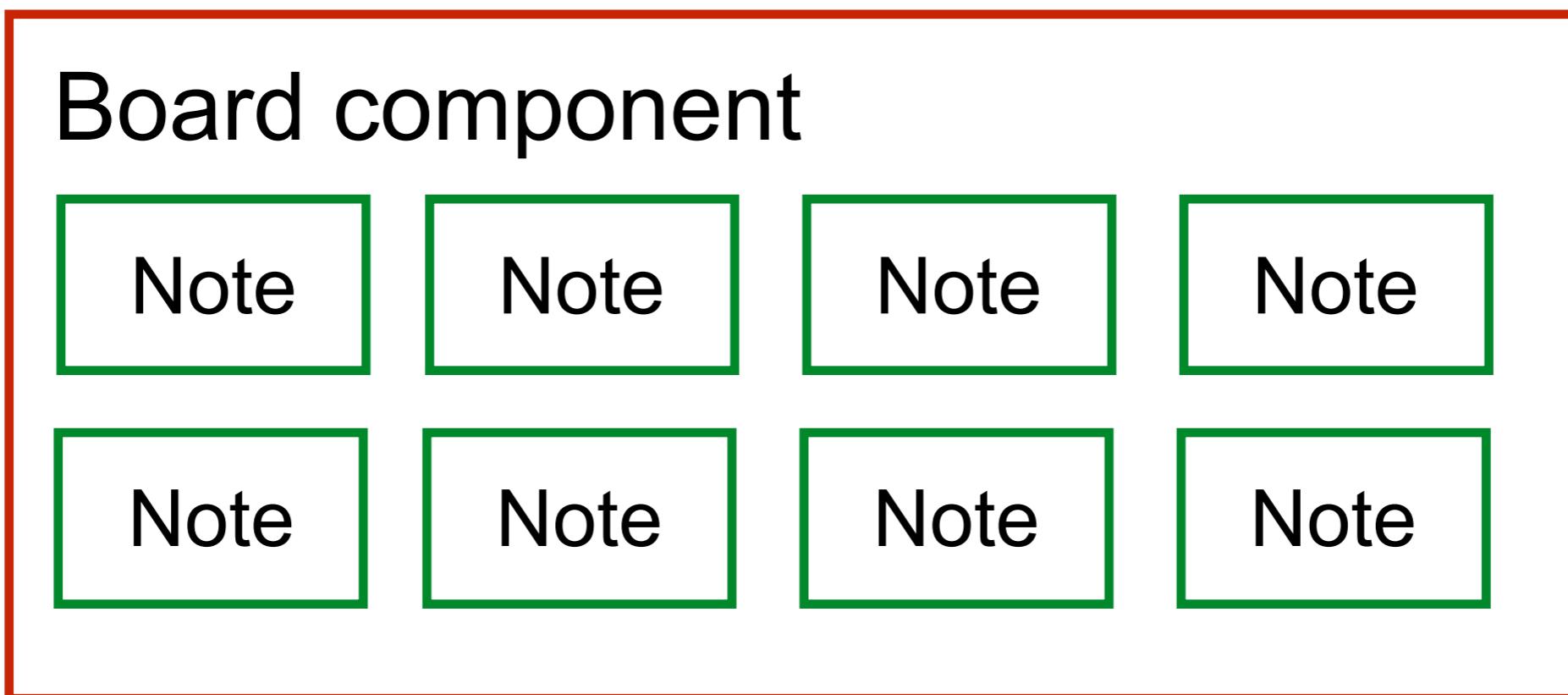
Edit file note.component.html

```
<div class="note" *ngIf="!isEdit">  
  <p>{{title}}</p>  
  <span>  
    <button id="edit" (click)="edit()">Edit</button>  
    <button id="remove" (click)="remove()">Remove</button>  
  </span>  
</div>
```



7. Parent and child component

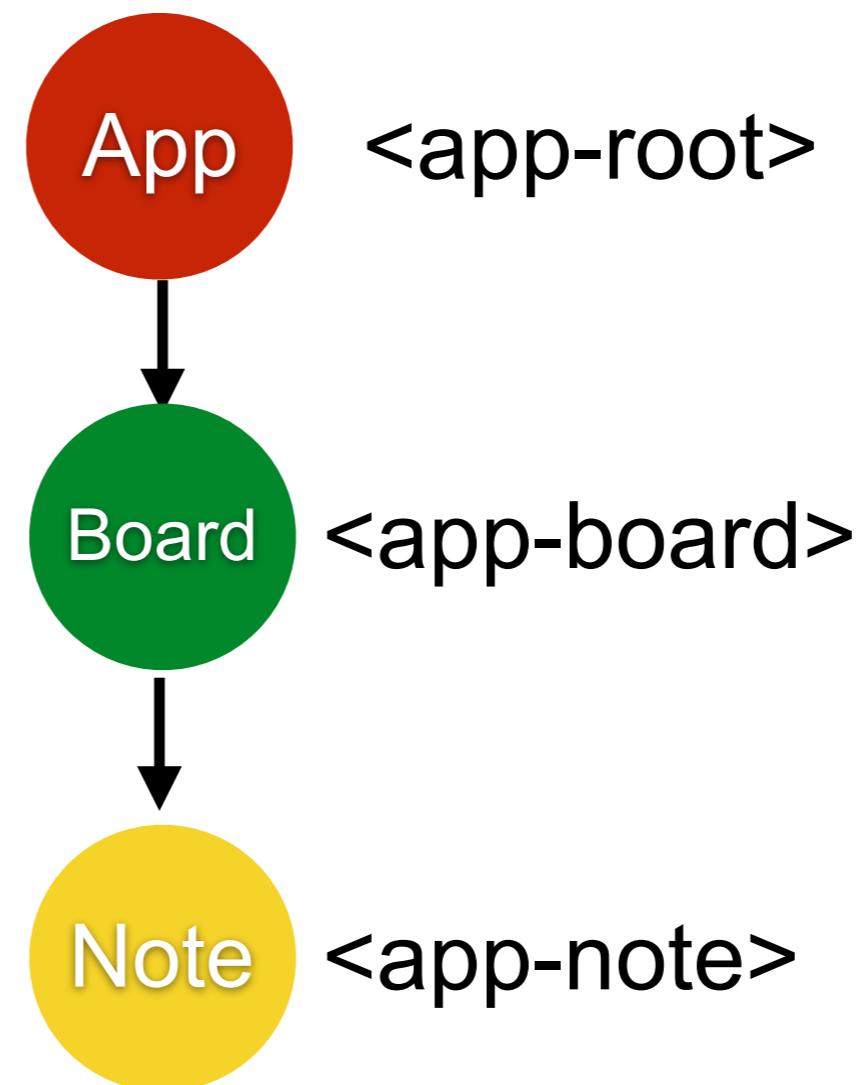
Parent = Board
Child = Note



7. Parent and child component

Parent = Board

Child = Note



Create Board component

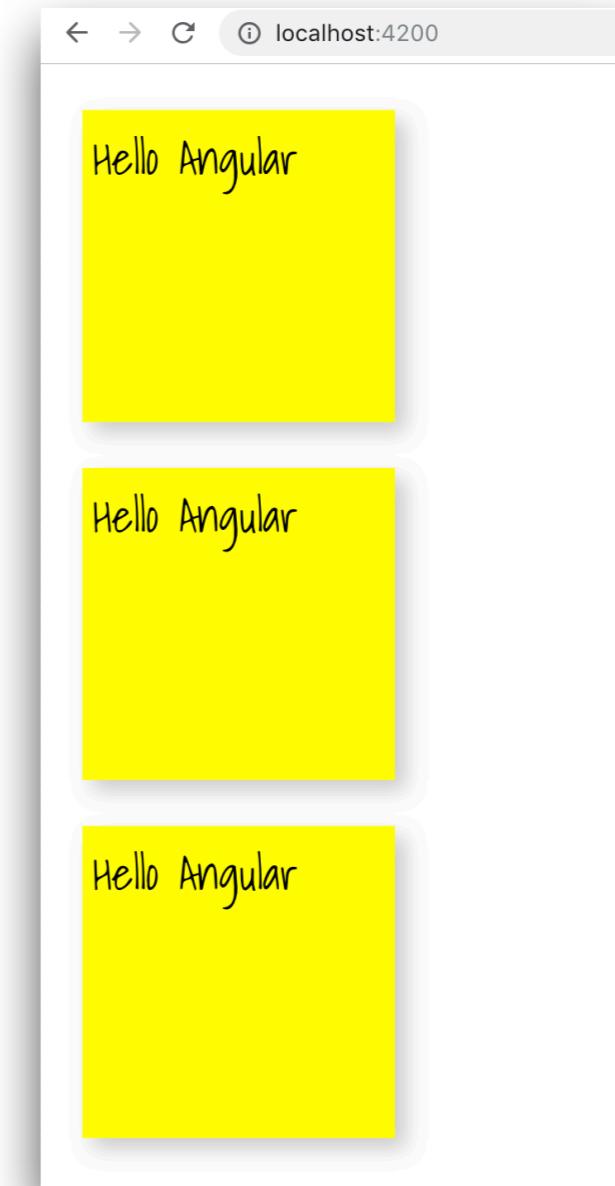
\$ng generate component board



Add note to board component

Edit file board.component.html

```
<div>
  <app-note></app-note>
  <app-note></app-note>
  <app-note></app-note>
</div>
```



8. Send data from parent to child

Board Component

Stateful component

Hold data of all notes

Board component

Note

Note

Note

Note

Note

Note

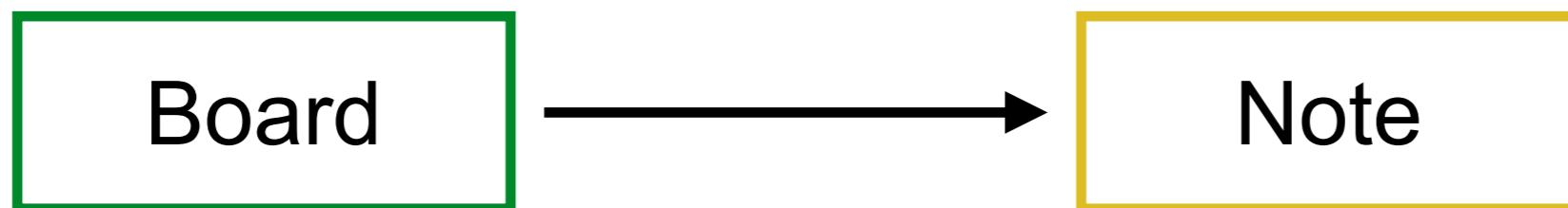
Note

Note



8. Send data from parent to child

Passing data using @Input



Passing data to component

board.component.html

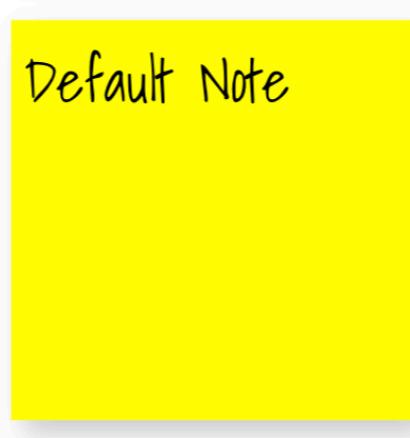
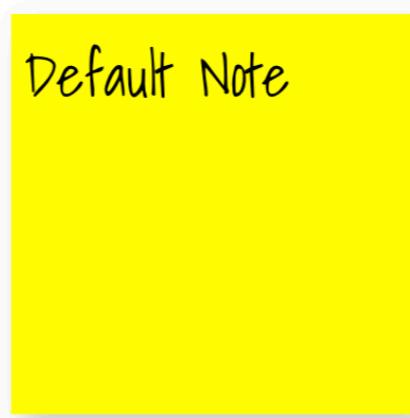
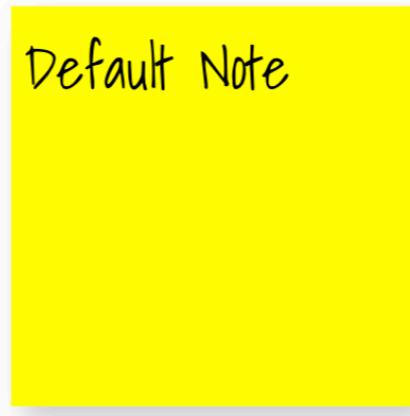
```
<app-note [title] = "title"></app-note>
<app-note [title] = "title"></app-note>
<app-note [title] = "title"></app-note>
```

note.component.ts

```
export class NoteComponent implements OnInit {
  @Input() title: string;
}
```



Result



Send more data

Passing data using @Input



```
notes = [
  {
    id: 1,
    title: 'Note 1'
  },
  {
    id: 2,
    title: 'Note 2'
  },
  {
    id: 3,
    title: 'Note 3'
  }
];
```



Send more data

board.component.html

```
<div *ngFor="let note of notes">  
  <app-note [title]="note.title"></app-note>  
</div>
```

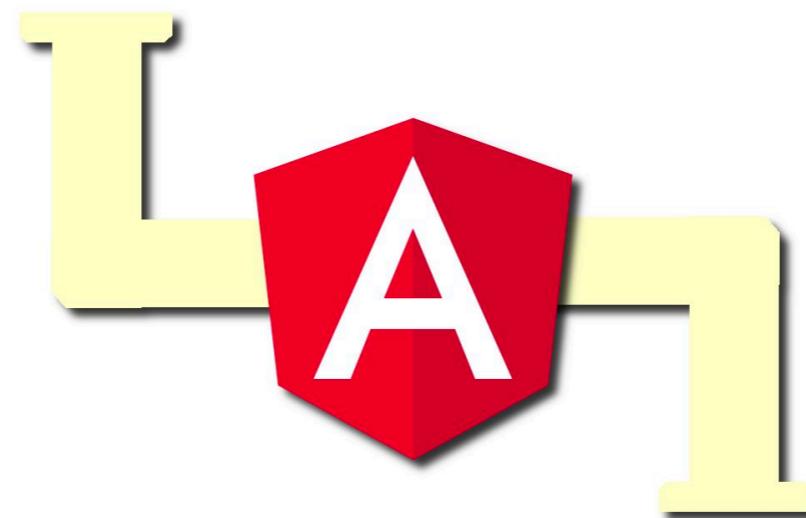
Note 1

Note 2

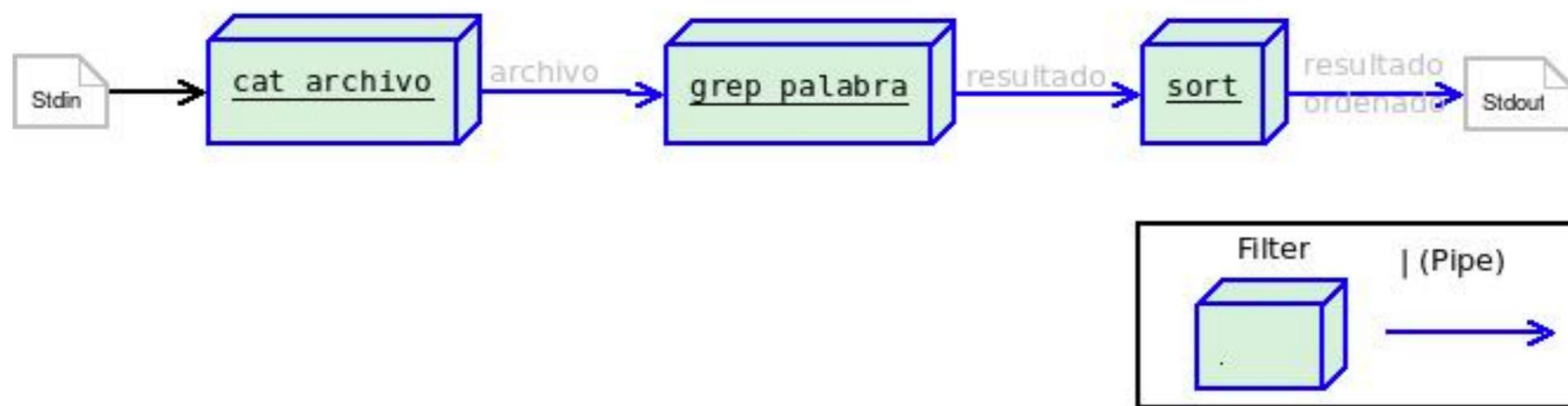
Note 3



Transform data with Pipe



```
$cat Arquivo | grep palabra | sort
```



Angular Pipe

Pipes allow us to change the way to show data and transform data in our template



Angular Pipe

Build-in pipe

Parameterize pipe

Chaining pipe

Custom pipe

Filter



Build-in pipes

Date

Lowercase

Uppercase

Currency

Percent

<https://angular.io/api?type=pipe>



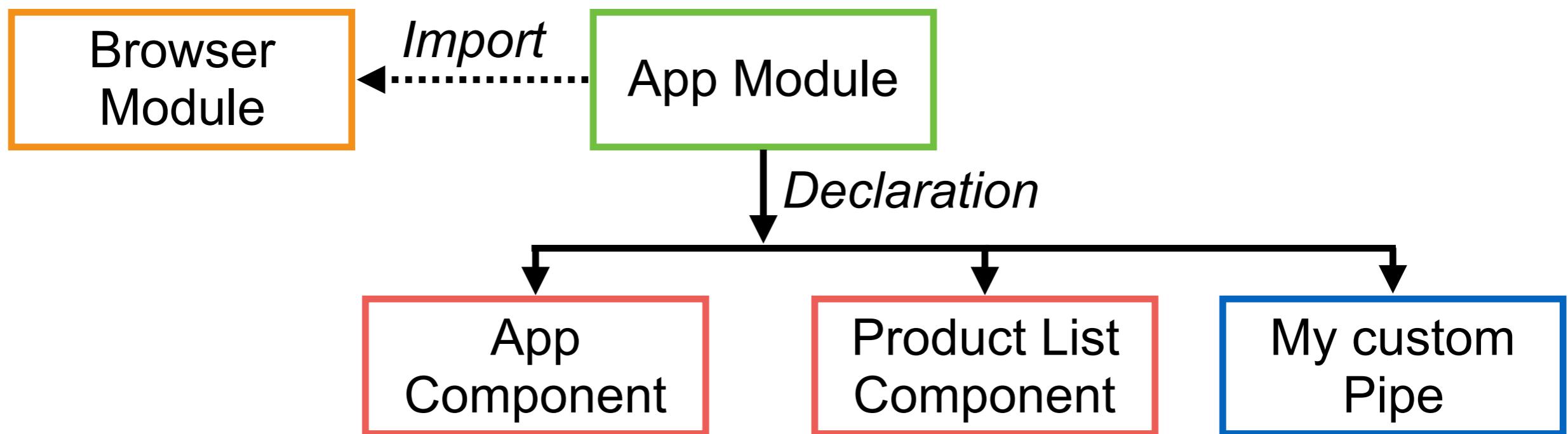
Build-in pipes

```
<th>{{product.code}}</th>
<th>{{product.name | uppercase}}</th>
<th>{{product.price | currency: 'THB ' : 'symbol': '1.2'}}</th>
<th>Yes</th>
<th>****</th>
```



Custom pipes

\$ng generate pipe <pipe name>



Create custom pipes

\$ng generate pipe ReplaceWithDash

replace-with-dash.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'replaceWithDash'
})
export class ReplaceWithDashPipe implements PipeTransform {

  transform(value: string, character: string): any {
    return value.replace(character, '-');
  }
}
```



Add pipe in app module

\$ng generate pipe ReplacewithDash

app.module.ts

```
import { ReplaceWithDashPipe } from './replace-with-dash.pipe';

@NgModule({
  declarations: [
    AppComponent,
    ProductListComponent,
    ReplaceWithDashPipe,
  ])
})
```



Using pipe in template

product-list.component.html

```
<th>{{product.code | replaceWithDash: '-'}}</th>
```

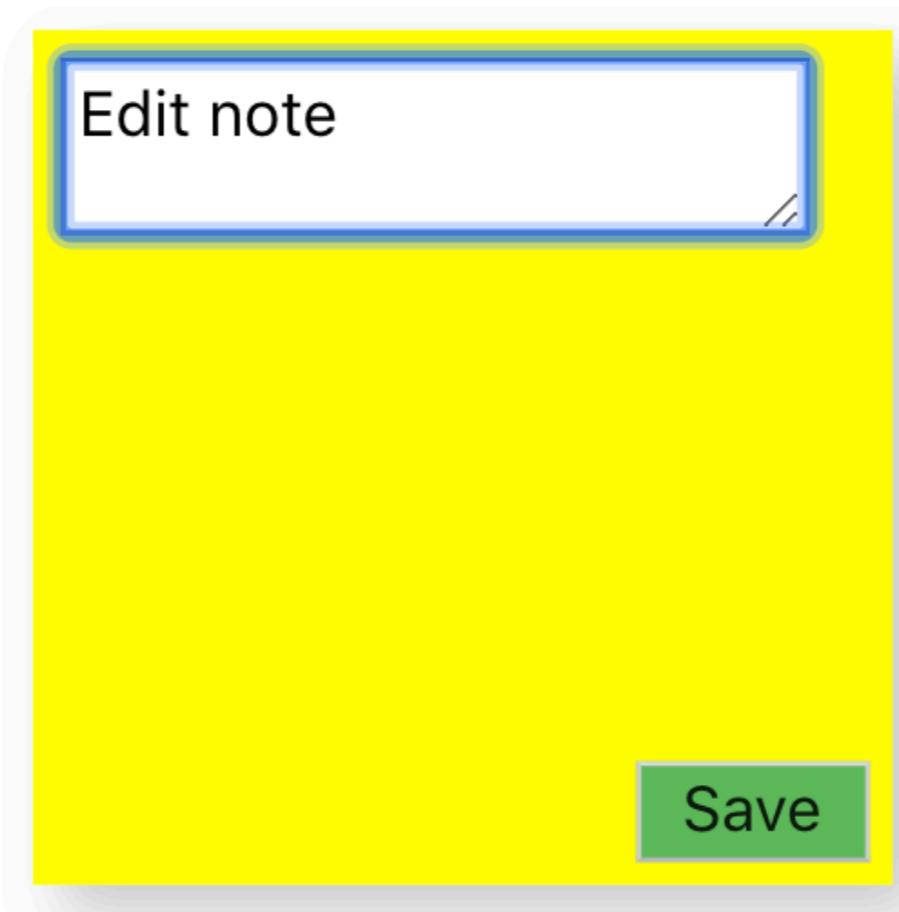
List of Product Page		
Filter by:		
Filtered by		
Hide Image	Product Code	Product Name
	AA-0001	PRODUCT NAME 1
	BB-0002	PRODUCT NAME 2



Edit/Update Note



Edit Note



Edit Note

Board component

Logic to edit note

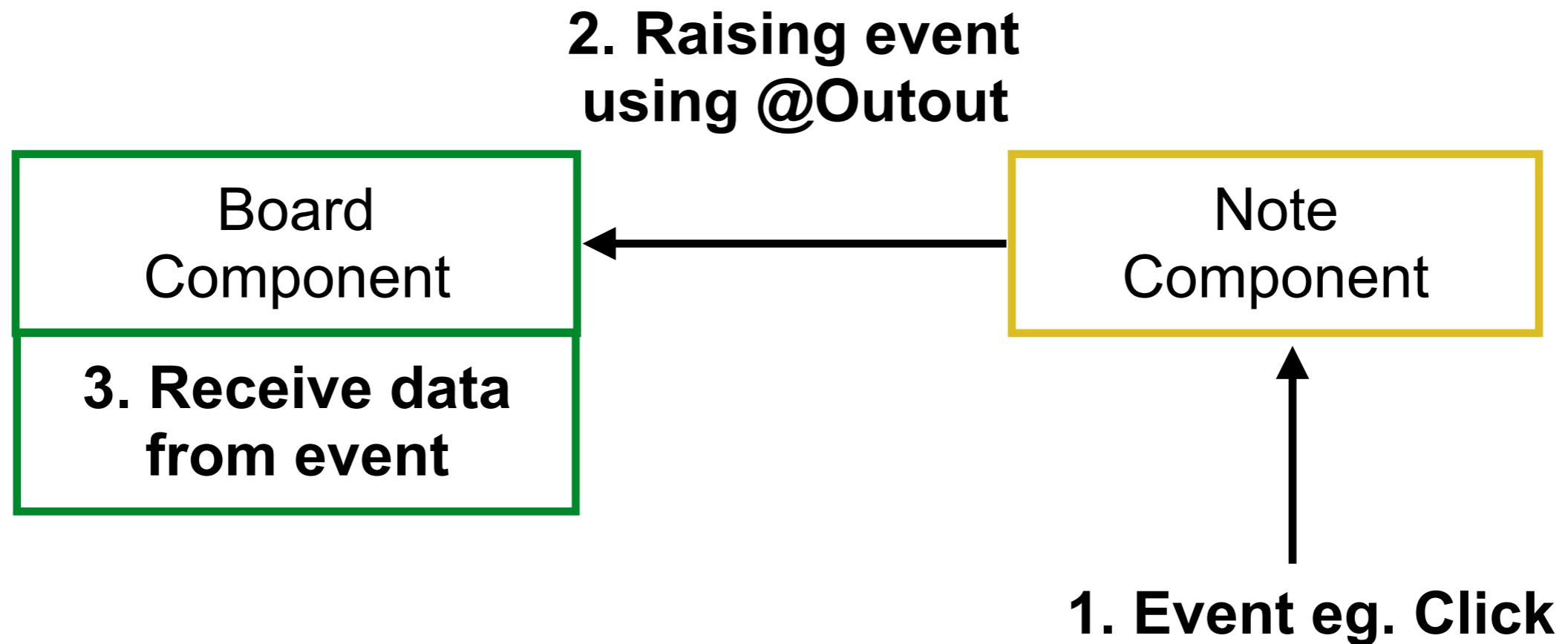
@Input

Note component

Handling when data changed



Raising event from component



1. Event handling on rating

note.component.html

```
<button id="save" (click)="save(newTitle.value)">Save</button>
```

note.component.ts

```
export class NoteComponent implements OnInit {  
  
  @Input() note: Note;  
  
  save(newTitle: string) {  
    ...  
  }  
}
```



2. Raising event with @Output

note.component.ts

```
export class NoteComponent implements OnInit {  
  
  @Input() note: Note;  
  @Output() savedNote: EventEmitter<Note> = new  
    EventEmitter<Note>();  
  
  save(newTitle: string) {  
    const newNote = new Note(this.note.id, newTitle);  
    this.savedNote.emit(newNote);  
  }  
}
```

board.component.html

```
<div *ngFor="let note of notes">  
  <app-note  
    [note]="note"  
    (savedNote)="onSavedNote($event)"></app-note>  
</div>
```



3. Receive data from event

board.component.html

```
<div *ngFor="let note of notes">
  <app-note
    [note]="note"
    (savedNote)="onSavedNote($event)"
  ></app-note>
</div>
```

board.component.ts

```
export class BoardComponent {
  notes = []

  onSavedNote(savednote: Note) {
    this.notes = this.notes.map( note =>
      note.id === savednote.id ? savednote : note );
  }
}
```



Delete Note



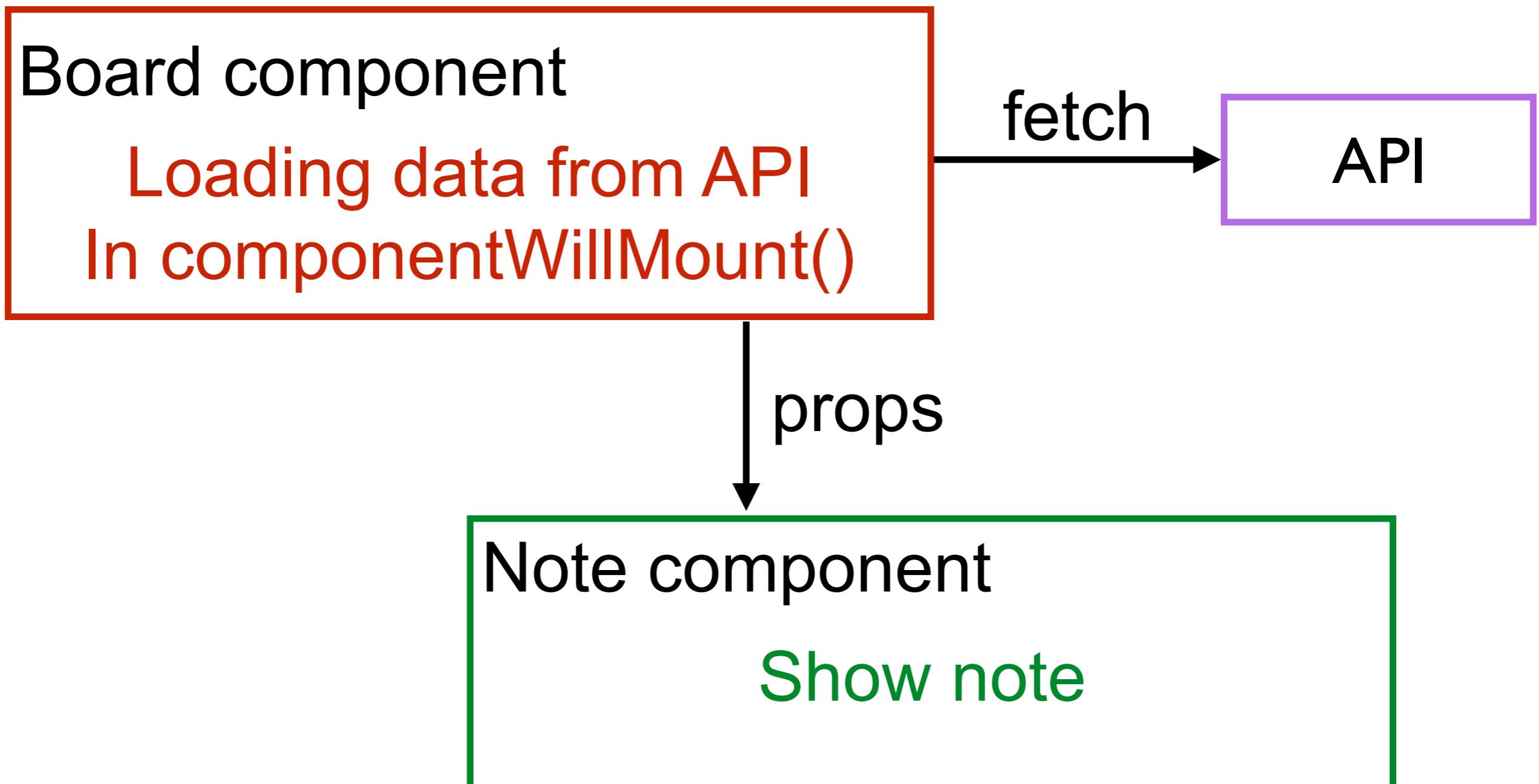
Add new Note



Loading data from API



Loading data from API



<https://baconipsum.com/json-api/>



Services



Service

Class with specified purpose

Independent from any component

Provide shared data and logic across components

Encapsulate external interactions

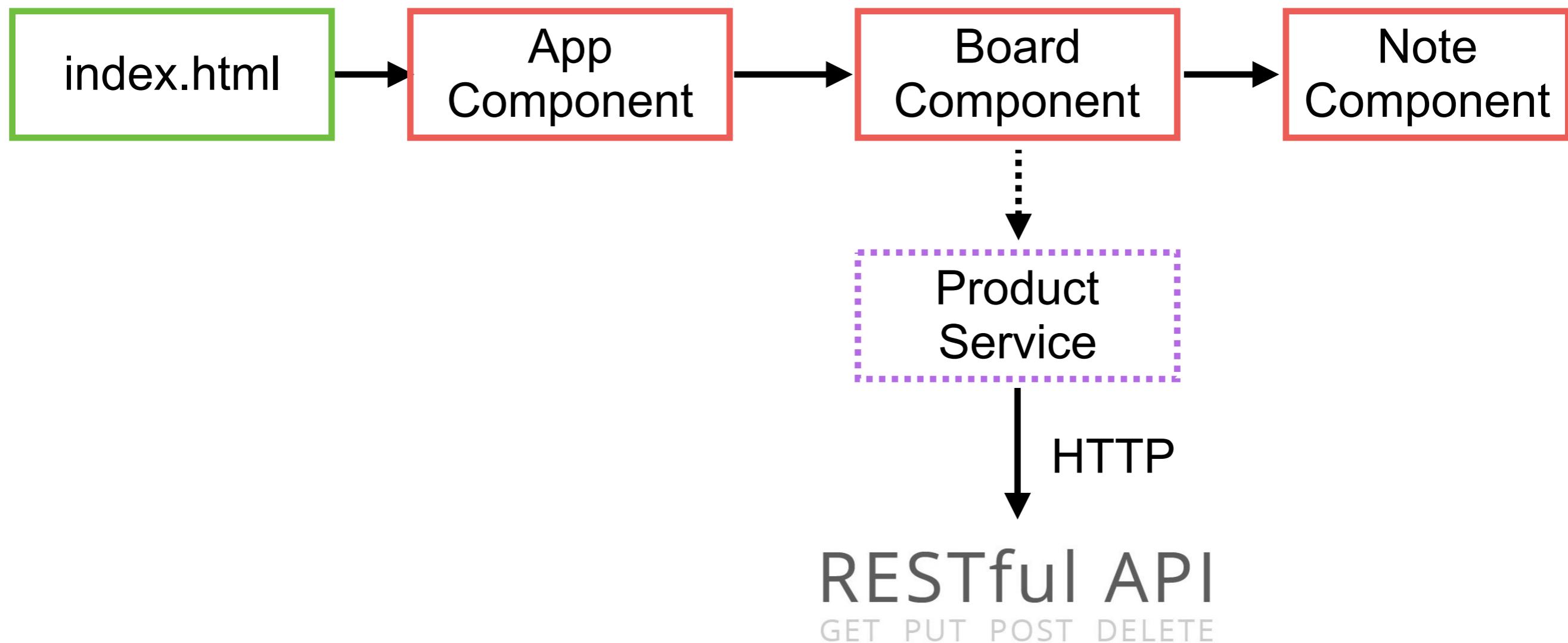


Service

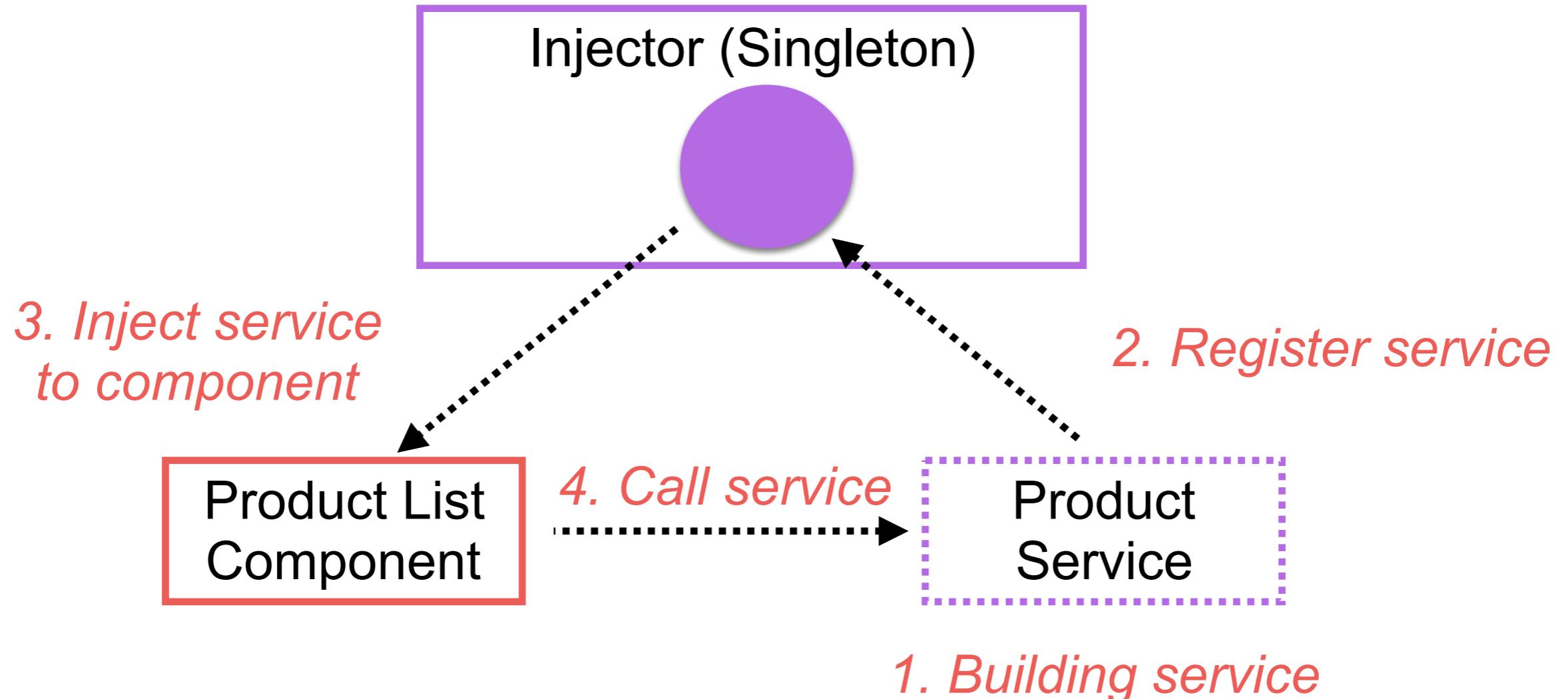
How service working ?
Building service
Register service
Inject service



Application Architecture



How service working ?



Building service

\$ng generate service api/my

my.service.ts

```
import { Injectable } from '@angular/core';

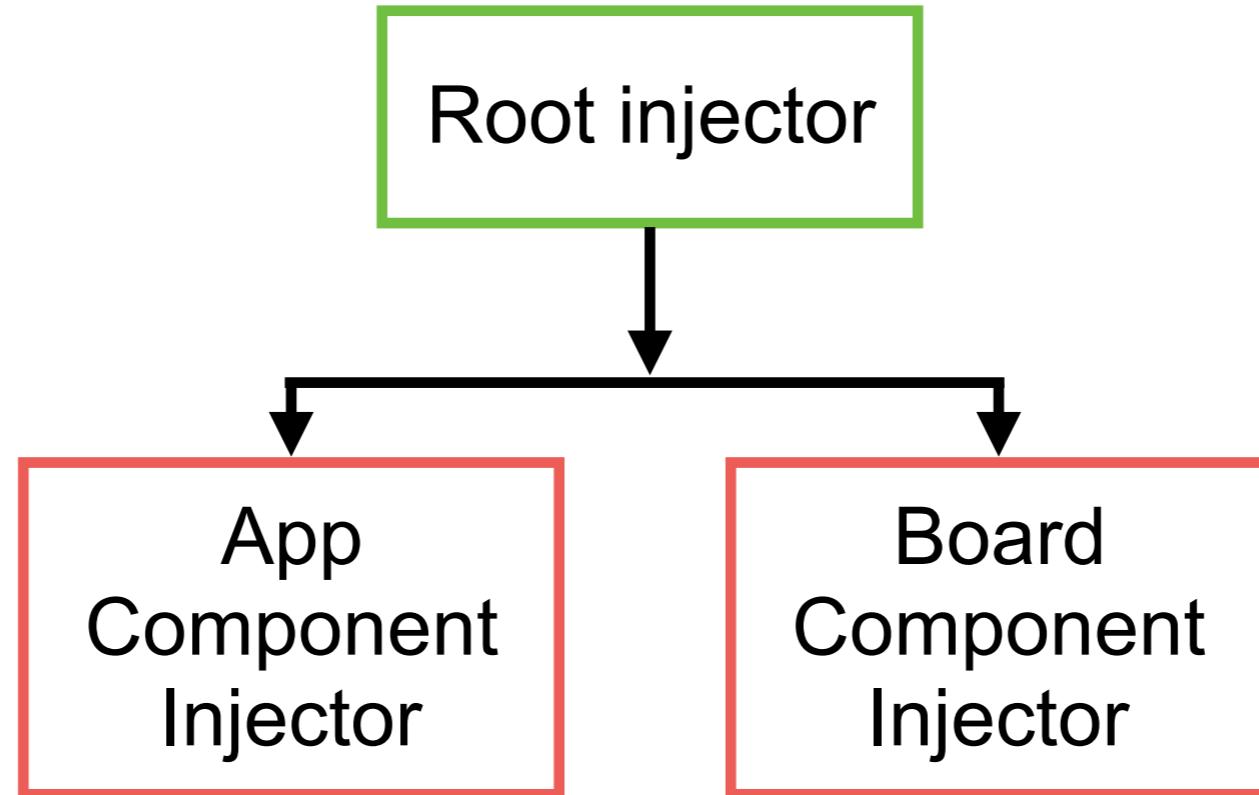
@Injectable({
  providedIn: 'root'
})
export class MyService {

  constructor() { }

}
```



Register service



Register service

Root injector
Component injector

product.service.ts

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class ProductService {

  constructor() { }

}
```

→ *Root injector*



Add service in component

board.component.ts

```
import {MyService} from './my.service';

@Component({
  selector: 'app-board',
  providers: [MyService],
})
export class BoardComponent implements OnInit {
```

Add service providers



Inject service to component

Using constructor

board.component.ts

```
export class BoardComponent implements OnInit {  
    Constructor injection  
    constructor(private myService: MyService) {  
    }  
  
    ngOnInit(): void {  
        this.datas = this.myService.getDatas();  
    }  
}
```



Working with HTTP

