# Các framework🡪 mục đích cuối cùng:

## Kỹ năng phải đạt/học 🡪 vừa làm vừa nghĩ

* Typescript
* Các khái niệm :
  + Component, service, databind, routing,…
  + Khái niệm riêng của từng thèn: DOM(render của rectjs, ,,, của vuejs)
  + Nodejs với angular( webpack,…)
* Dùng để xây dựng các mảng chi tiết( table, chart, datepicker,…): search awesome,,,

🡺xây dựng được 1 app bên frontend, chú ý: code framework lien hệ code thuần

## Nó là gì??

* Angularjs: tích hợp mọi thứ (với nodejs)
* Reactjs🡪 giao diện
* Vuejs tương tự reacjs ở mức khái niệm, giống databinding của angularjs
  + Tại sao ổng chọn vuejs🡪 code gọn ở databind,…(Đỡ đào tạo) khi kết hợp với django
  + Trang chủ: guide + style code + các trang …
  + ĐƯợc nhiều nguowfiq quan tâm(github vote cao) 🡪 xây dựng các trang Dùng để xây dựng các mảng chi tiết( table, chart, datepicker,…)🡺 search awesome,,, : họ xây dựng rất nhiều vì đơn giản

# Chi tiết angular js :

## Tổng quan guide :

### Tương tự bên django , phải đọc 2 guide :

* Turiol(First step)(bắt buột đọc, ví dụ đơn giản nhưng ko đọc ko hiểu lun)
* Chi tiết
  + Fundamentals(nguyên tắc cơ bản)🡪 code,kỹ thuật ,..
  + Techniques(kỹ thuật ngoài lề tương tự bên django: i18,security, setup and development, service worker(django: tối ưu cache),.. tối ưu)
  + Api
* Guide: hình thức giống django

### Chi tiết trong guide:

* + Introduction
  + **summary:**
  + **Final code reviewer**

## Chi tiết guide- guide1-turiol:

<https://stackblitz.com/edit/angular-fqaoeb>

+ review từng bài:

### The Application Shell: + cấu trúc project + kiến thức cơ bản typescipt:

<https://angular.io/tutorial/toh-pt0>

* Tạo project, component, module, serive, --🡪 run ,build(1 số file cần thiết bundejs)…

#### Cấu trúc project

* + Component:

app.component.ts— the component class code, written in TypeScript.

app.component.html— the component template, written in HTML.

app.component.css— the component's private CSS styles.

* + App.modules.ts 🡪 import các khai báo vào (thường tự động): component, serive,module,..
  + Các file ngoài thư mục App

#### Typescript

* + Export class/…. Name{

}

* File khác có thể import được, chú ý 3 file component ko cần import, ko cần khai báo lẫn nhau, ko cần khởi tạo, tự nó hiểu rồi

#### Es6

<https://angular.io/tutorial/toh-pt1>

#### Two-way binding:

* + Input form có thể bị lỗi do thiếu khai báo ở file app.modules.ts:
    - import { [FormsModule](https://angular.io/api/forms/FormsModule) } from '@angular/forms'; // <-- [NgModel](https://angular.io/api/forms/NgModel) lives here
    - imports: [ [BrowserModule](https://angular.io/api/platform-browser/BrowserModule), [FormsModule](https://angular.io/api/forms/FormsModule) ],
  + **[(ngModel)]** is Angular's two-way data binding syntax.

<https://angular.io/tutorial/toh-pt2>

#### Export được const

import { Hero } from './hero';

export const HEROES: Hero[] = [

{ id: 11, name: 'Mr. Nice' },

{ id: 12, name: 'Narco' },

{ id: 13, name: 'Bombasto' },

{ id: 14, name: 'Celeritas' },

{ id: 15, name: 'Magneta' },

{ id: 16, name: 'RubberMan' },

{ id: 17, name: 'Dynama' },

{ id: 18, name: 'Dr IQ' },

{ id: 19, name: 'Magma' },

{ id: 20, name: 'Tornado' }

];

* Sử dụng

import { HEROES } from '../mock-heroes';

heroes = HEROES;

#### Add the click event handler:

* + Component template html:

<li \*ngFor="let hero of heroes**" (click)="onSelect(hero)"**>

* + Component ts:

//sự kiện click

selectedHero: Hero;

onSelect(hero: Hero): void {

this.selectedHero = hero;

alert(hero.name);

}

#### Style in template: set class selected

<li \*[ngFor](https://angular.io/api/common/NgForOf)="let hero of heroes" [class.selected]="hero === **selectedHero**"

1. .selected {
2. background-color: #CFD8DC !important;
3. color: white;
4. }

### gọi component từ 1 component khác:

<https://angular.io/tutorial/toh-pt3>

Summary

* You created a separate, reusable HeroDetailComponent.
* You used a [property binding](https://angular.io/guide/template-syntax#property-binding) to give the parent HeroesComponent control over the child HeroDetailComponent. 🡺 **cha +c0n**
* You used the [@Input decorator](https://angular.io/guide/template-syntax#inputs-outputs) to make the hero property available for binding by the external HeroesComponent.
* Cách làm:
  + Tách 1 phần component heroes(phần detail)(cha) 🡪 bỏ vào component hero-datail (component con)

<div \*ngIf="hero">

<h2>{{ hero.name | uppercase }} Details</h2>

<div><span>id: </span>{{hero.id}}</div>

<div>

<label>name:

<input [(ngModel)]="hero.name" placeholder="name"/>

</label>

</div>

</div>

+ Add a hero property, preceded by the @[Input](https://angular.io/api/core/Input)() decorator.

export class HeroDetailComponent implements OnInit {

@Input() hero: Hero;

* + Component cha(heroes) template html:

<app-hero-detail [hero]="selectedHero"></app-hero-detail>

[hero]="selectedHero" is an Angular [property binding](https://angular.io/guide/template-syntax#property-binding).

The hero property [must be an Input property](https://angular.io/guide/template-syntax#inputs-outputs), annotated with the @[Input](https://angular.io/api/core/Input)() decorator, because the external HeroesComponent [will bind to it](https://angular.io/tutorial/toh-pt3#heroes-component-template)

* + Chú ý khai báo bên component con.ts:

import { Hero } from '../hero';

import { [Component](https://angular.io/api/core/Component), [OnInit](https://angular.io/api/core/OnInit), [Input](https://angular.io/api/core/Input) } from '@angular/core';

### Service: + sử dụng injection of dependences:

* 2 ví dụ:

Service inject vào component.ts

Serivice inject serive(you inject the MessageService into the HeroServicewhich is injected into the HeroesComponent.)

* Xem them: <https://angular.io/guide/dependency-injection-pattern>
* <https://angular.io/guide/dependency-injection>
  + Service:
* import { Hero } from './hero';
* import { HEROES } from './mock-heroes';

@Injectable()

export class HeroService {

//ví dụ cơ bản ko cần injection messageSerice, đây là ta đang you inject the MessageService into the HeroServicewhich is injected into the HeroesComponent

constructor(private messageService: MessageService) { }

ngOnInit()

// getHeroes(): Hero[] {

// // return HEROES;

// }

//Observable: xử lý bất đồng bộ với RxJS --> ngOnInit() bên component được injection phải gọi nó mới được, hàm bình thường gọi ko được

// //(2)do xử lý bất đồng bộ : bên service : hàm getHeroes(): Observable<Hero[]>: trả về kiểu Observable

//https://kipalog.com/posts/Cung-Hoc-Angular-2--Can-ban-ve-RxJS-trong-Angular-2

getHeroes(): Observable<Hero[]> {

// TODO: send the message \_after\_ fetching the heroes

this.messageService.add('HeroService: fetched heroes');

return of(HEROES);

}

* + component

//qua service

heroes: Hero[];

//injection

constructor(private heroService: HeroService) {

}

// set biến của lớp(heroes) = method getHeroes bên serivce thông qua injection

getHeroes(): void {

// this.heroes = this.heroService.getHeroes();

this.heroService.getHeroes()

.subscribe(heroes => this.heroes = heroes);

}

//(1)phương thức :vòng đời angular???

//Lifecycle Hooks : https://angular.io/guide/lifecycle-hooks

ngOnInit() {

this.getHeroes();

}

//end service

### Routing:

Xem cách **service** messge component hoạt động

### Http(s):

Get,put,delete,..

## Build:

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

# Tool:

### Editor online

### Debug:

Vuejs tương tự với reactjs: DOM (render)🡪 có app debug riêng