

Angular

<https://angular.io/docs>

Front-end, SPA, Typescript

Package 구조

package.json	Project에 필요한 Package 설치 및 외부 모듈에 대한 의존성 관리를 위한 환경설정
tsconfig.json	Typescript → Javascript 변환을 위한 Typescript Compiler 설정
typings.json	Typescript Compiler가 인식할 수 없는 추가적인 외부 Library 파일 정의
systemjs.config.js	Application Module을 찾기 위한 Module Loader 정보 제공, 필요한 패키지 설치

Package 생성

- <https://github.com/angular/quickstart>

수동 생성

1. package.json 생성

```
npm init
#이후 질문에 따라 입력하면 자동 생성.
#파일 생성 이후 dependencies, devDependencies 정의
```

- dependencies : Angular Project 실행에 직접적으로 필요한 필수적인 의존성 패키지 기술
- devDependencies : Angular Project 개발에 필요한 부가적 패키지 기술

2. 나머지 파일 생성

[Quick Start](#) 참조

3. 의존성 패키지 설치

```
npm install
```

4. Project 실행

```
npm start
```

Angular CLI

설정보다 **개발**에만 집중할 수 있도록 Project 설정 및 관리를 명령어 기반으로 수행할 수 있도록 지원.

<https://github.com/angular/angular-cli>

```
#설치
npm install -g angular-cli@latest

#프로젝트 생성
ng new hello-ng2

#Server 실행 (Default port:4200)
cd hello-ng2

# 추가 인자: --port 4200 / open
ng serve
```

```
#Component 추가
ng g component NAME

#Directive 추가
ng g directive NAME

#Pipe 추가
ng g pipe NAME

#Service 추가
ng g service NAME

#Class 추가
ng g class NAME
```

```
#-prod : 배포 환경을 위한 옵션. 파일 크기 최적화
ng build -prod

#/dist에 결과 생성
```

Architecture

Component	가장 핵심이 되는 구성요소. 화면 구성 담당. HTML, CSS, JavaScript를 하나의 단위로 묶는 기술. 크게 Directive, Template, Class로 나뉜다.
------------------	---

Module	화면의 구성요소 집합. (사용하려는 컴포넌트 명시하는 관리자 역할)
Service	재사용 가능한 Logic/기능 집합. Component 외부에 정의.
Directive	Template의 Element 속성/이름 등을 관리. (공통적인 임의 속성 처리에 유용)
Router	Page 이동/교체

Data Binding

One way

Interpolation binding

삽입식. 문자열로 변환될 수 있는 값을 View에 Binding. Template 표현식에서 사용하는 변수와 함수는 **Component Class**의 **Context**에 포함된 **Attribute** 및 **Method**.

- **하지 말아야 할 것** (☞ Component에 작성. 혹은 Data 가공이 필요한 경우 Pipe 이용)
 - 시간이 오래걸리는 연산 작성
 - 값을 할당하는 표현식 작성

```
<span>{{name}}</span>
```

Property binding

Component에서 DOM의 Property로 값을 Binding.

```
<input type="text" [disabled]="name === '철수'" #nameInput>
```

Event binding

View에서 발생하는 Event를 처리할 Logic binding.

<https://developer.mozilla.org/pl/docs/Web/Events>

```
<input type="button" (click)="setName(nameInput.value)"
```

Two Way

```
<input type="text" [(ngModel)]="userName"> <!-- FormsModule에서 제공 -->
```

Component

기초

명명법

- Component 파일명은 “**component**” 단어를 중간에 넣어 표현
 - ex) myComp.**component**.ts
- Component Tag명은 “-”으로 단어 구분.
 - ex) selector:'my-book'

영역

import, Component, class 영역으로 나눈다.

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

// 화면에 보이는 부분(Component) 정의
// 만약 속성값이 여러줄이면 `(억음부호)`을 사용한다.
@Component({
  selector: 'my-person', // template으로 인식할 HTML Tag 지정
  templateUrl: './app/person.component.html',
  styleUrls:['./assets/stylesheets/person.css']
})

// Logic 정의
// template에서 {{name}} 혹은 {{getName()}}으로 호출
export class PersonComponent {
  private name:string = "Luke";

  public getName():string {
    return this.name;
  }
}
```

Dependency Inject

동적 Dependency Inject

- @Host() : 상위 DI 정보를 찾지 않고, **현재 Component**에서 DI 정보를 찾아서 주입
- @Optional() : DI 정보가 있으면 주입, 없으면 주입하지 않음.

```
export class DynamicDependencyInjectTestComponent {

  constructor(
    // 이 때, doctor은 상위 Component에서 이미 주입을 받았다고 가정.
    // 결과적으로 @Host() 특성상 현재 Component만 기준으로 하므로 null 반환, @Optional로
    인해 person에 programmer를 할당.
    @Host()
    @Optional()
    doctor: Person,
    programmer: Person
  ) {
    this.person = doctor ? doctor : programmer;
  }
}
```

Directive

Template을 동적으로 만들어주는 요소.

Structural directive

DOM 요소 동적 처리 (ngIf, ngFor, ngSwitch, ...)

```
<button type="button" (click)="toggle()">Toggle</button>

<span *ngIf="isShow; else hiding">보인다!</span> <!-- else 구문은 4부터 지원 -->
<ng-template #hiding>
  <span>안보인다!</span>
</ng-template>
```

Attribute directive

DOM 속성 동적 처리 (ngClass, ngStyle). Property Binding을 이용하여 변수값 하나에 대한 바인딩이 아닌 여러 값(객체)으로 Binding이 필요한 경우 사용.

```
<div [ngClass]="myObj">Test</div> <!-- myObj의 true인 아무 Key를 반영.-->
```

```
<div [ngStyle]="myStyle"></div>
```

```
// component
myStyle = {
```

```
background-color: this.isActive? 'white' : "green",
visibility: this.isDisabled? 'hidden' : 'visible'
}
```

영역

```
import {Directive, ElementRef, Renderer, HostListener} from '@angular/core'

@Directive({
  selector: '[my-color]',
  host: {
    // Event 정의
  }
})

export class MyColorDirective {

  //속성 구현...
  @Input('my-color') color;

  constructor(private el:ElementRef, private renderer:Renderer){
  }
  // Event Listener 구현
  @HostListener('focus') onFocus(){
    this.renderer.setStyle(
      this.el.nativeElement,
      'background',
      this.color); //template의 속성에서 my-color="색상"으로 적용.
  }
}
```

Pipe

Template에 Data를 보여줄 때 가공이 필요한 경우 사용.

```
{{ person | personPipe }}
```

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

@Pipe({
  name: 'personPipe'
})
export class PersonPipe implements PipeTransform {
```

```
transform(person:any): any {  
    return `${person.name} (${person.age})`;  
}  
  
}
```

Module

Angular 안의 관련된 요소를 하나로 묶어 Application을 구성하는 단위.

```
import { NgModule }      from '@angular/core';  
import { BrowserModule } from '@angular/platform-browser';  
import { AppComponent }  from './app.component';  
  
@NgModule({  
  imports:      [ BrowserModule ], // 다른 모듈에서 exports로 선언한 Subset을 사용  
  // 하기 위해 정의  
  providers: [Logger], // Service 등록  
  declarations: [ AppComponent ], // Component, Directive, Pipe 등 정의  
  exports: [AppComponent], // 다른 Module에서 사용할 수 있도록 정의한 Subset  
  bootstrap:    [ AppComponent ]  
})  
  
export class AppModule { }
```

Bootstrapping

Application을 최초 실행할 때 진행되는 과정. (Compile 목적)

Life Cycle

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

수동으로 강제 변화 감지

- **ApplicationRef.tick()** : 전체 Component 검사
- **NgZone.run(callback)** : 전체 Component 검사 이후 callback 실행
- **ChangeDetectorRef.detectChanges()** : 자신과 자식 component만 검사.

Trouble Shooting

6+에서 **polyfills.ts**에 **rx** 연산자가 인식 안되는 경우

cli에서 설치하는 rxjs가 5.x이기 때문.

```
npm install rxjs@6 rxjs-compat@6 --save
```

Build 이후 **Path**로 인한 **SecurityError**

<https://stackoverflow.com/questions/49622076/securityerror-failed-to-execute-replacestate-on-history>

- RouterModule에서 useHash 설정

```
RouterModule.forRoot(routes, {useHash: true});
```

- index.html 수정

```
<!-- <base href="/"> -->  
<script>document.write('<base href="' + document.location + '"  
>');</script>
```

- And then build with -base-href

```
ng build --prod --base-href ./
```

Creating libraries

- <https://angular.io/guide/creating-libraries>

Local link

배포하지 않고 실시간으로 개발 하는 방법

1. Library

```
cd ./dist/[Library name]  
npm link
```

```
# library root  
ng build my-lib --watch
```

2. App (Library Module을 import 하는 쪽)

```
npm link
```

```
ng serve
```

From:

<http://ledyx.xyz/> - **Ledyx WIKI**

Permanent link:

<http://ledyx.xyz/doku.php?id=front-end:angular>

Last update: **2023/05/01 14:40**

