***Angular 2.0***

**목차**

***2.2 Angular 2.0 과 1.0의 차이점***

- Controller, $scope, jqlite 등의 주요 요소 삭제 --> Component 로 대체

2.0에서 주 내용은, Controller, $scope, j0qlite와 같이 주요 요소는 삭제되고, 타입 스크립트(TypeScript)가 추가된다는 것입니다

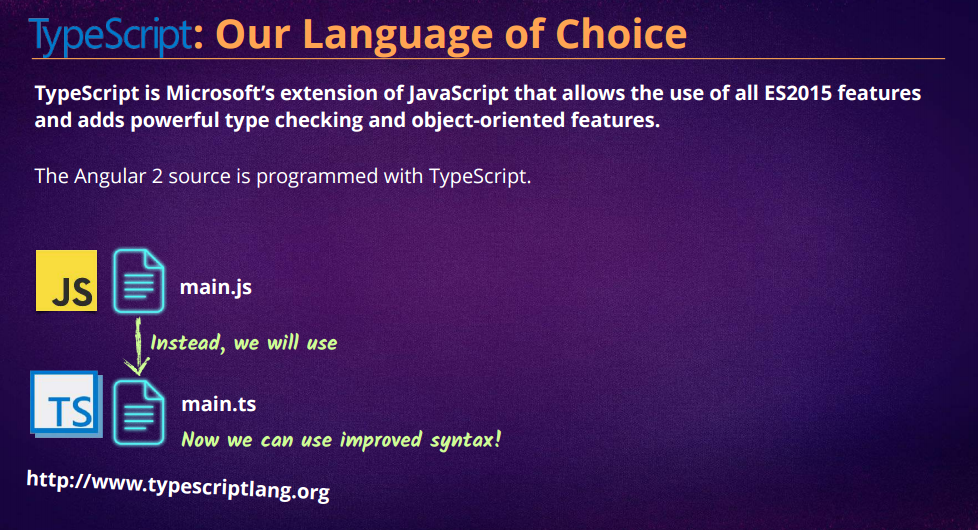
- Web 컴포넌트로서의 Angular 2.0

AngularJS 2.0은 웹을 컴포넌트화를 시도하고 있습니다. controller 기능은 사라지고, 대신 기능을 컴포넌트화 하여 레고블록과 같이 조립하는 구조를 가지고 있습니다..

컴포넌트의 모든 속성과 메소드를 템플릿에서 사용할 수 있습니다. D.I 로 대명사였던 angular.module도 삭제되고 대신 서비스의 개념을 이용 합니다. 또한 jquery를 component로 보고, 컴포넌트로 추가를 하면 Jquery 기능을 모두 사용할 수 있습니다. angular는 jquery와 같은 JS에 종속된 프레임워크로 가져가지 않고 측면에서 기존 directive와 같은 선언형식의 컴포넌트 형식으로 집중하겠다는 의미로 풀이됩니다.

***2.3 Typescript***

타입스크립트(Typescript)는 Angular 2.0에 채택된 언어입니다.



타입스크립트(Typescript)는 MS에서 개발한 자바스크립트 상위 집합 언어로서 Angular 2.0의 주 언어로 채택 될 정도로 기능이 강력하고 ECMA 스크립트 표준을 따르는 차세대 자바스크립트입니다.

Typescript의 특징으로는 큰 규모의 자바스크립트 프로젝트에 적용 가능 합니다. 객체 지향 프로그래밍 지원으로 ES5(브라우저 지원 자바스크립트)에 비해 코드 복잡도가 낮습니다.

또한 명시적인 타입을 제공하여 통해 컴파일 단계에서 발생 될 수 있는 자바스크립트의 결점을 보완했습니다. ex) Boolean형 fales와 숫자 0을 구분할 수 있게 되었습니다.

인텔리센스를 지원하며 자바, C# 등의 객체지향 언어와 비슷한 문법 스타일을 갖추고 있어 학습비용이 낮습니다.

***2.4 Angular 2.0의 장,단점***

***Angular 2.0의 장점***

￮ 깔끔한 MVC 구성으로 튜토리얼 정도만 참조해도 MVC 구조를 파악할 수 있을 정도로 직 관적입니다.

￮ Component는 가독성이 높고 API특징이 명확하여 이해하기 쉽다. 게다가 Angular 2로 전환이 자유롭다.

￮ HTML5 웹 컴포넌트의 확장성의 용이함을 갖추고 있습니다.

￮ Typescript 채택으로 IDE를 사용하는 프로그래머는 환상적인 자동 완성 기능과 type 제 안을 제공 받을 수 있다.

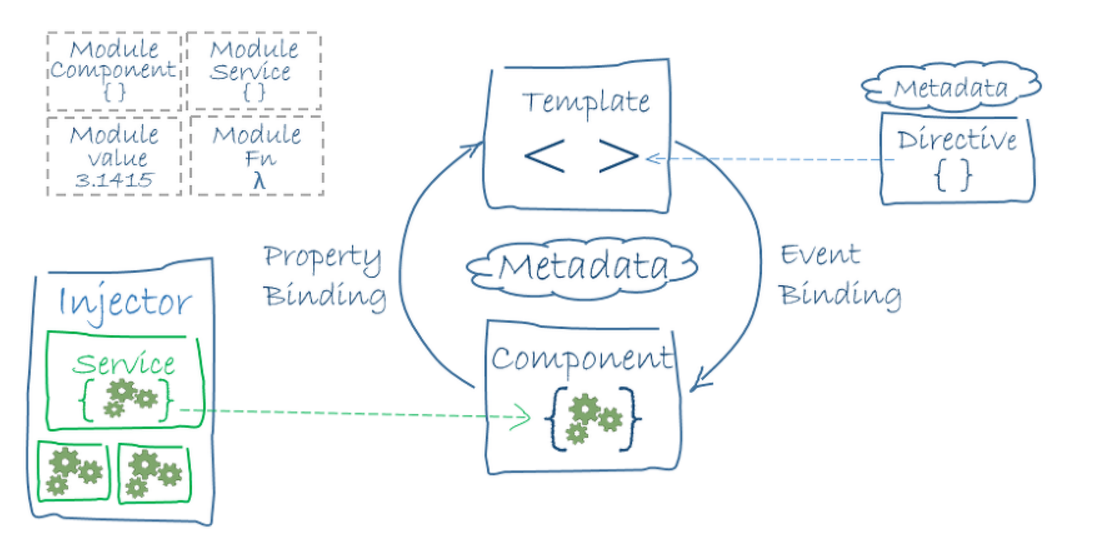
***Angular 2.0의 단점***

￮ Typescript의 채용은 맞은 이점은 주지만 이전의 javascript 보단 높은 진입장벽과 systemjs와 webpack 등의 프레임워크 사용을 강제 합니다.

￮ 국내에 부족한 정보, 대부분의 사이트에서 튜토리얼 단계의 정보만 제공되며 그 이상의 정 보는 부족합니다.

**3. Angular 2.0 의 구조**

Angular에서 가장 중요한 구조는 바로 컴포넌트 입니다. 컴포넌트는 화면을 구성하기 위한 하나의 단위이며, 기본적으로 템플릿 + 메타데이터 + (컴포넌트)클래스 조합으로 구성하고 애플리케이션 로직을 정의한다.



**- Angular의 Architecture diagram**

***3.1 Modules***

컨트롤러, 서비스, 필터, 지시어 등을 포함하여 응용프로그램의 서로 다른 기능을 구성하는 컨테이너

모든 자바스크립트 기능들이 ng-app"첫 모듈명"을 시작으로 모듈 단위로 관리됨

Angular에서 Module은 Application을 조작하거나 외부 라이브러리를 통해 기능을 확장하는 기능을 갖고 있습니다. 모든 Angular app은 최소한 하나의 모듈을 가지고 있습니다. 이는 Angular app이 root module 하나는 필수적으로 포함해야하기 때문입니다. 이 root modul의 관념적인 이름이 AppModule 입니다.

root module은 작은 application의 유일한 하나의 모듈일 수도 있지만, 대부분의 application에서는 더 많은 수의 feature module을 가지고 있습니다. feature module 들은 각각 application 도메인, work flow 등등의 다양한 용도로 쓰입니다.

Angular에서의 Module은 root module이거나 feature Module인데, @NgModule이라는 Decorator가 있는 클래스입니다. FormsModule, HttpModule, RouterModule ... 등 많은 Angular library는 모듈로 제작 되었으며, 많은 Third-party 라이브러리도 Angular Module로 활용할 수 있다.

Application이 로드될 때 Module을 로드할 수 있으며, NgModule은 속성 모듈을 설명하는 하나의 메타데이터 객체를 취한다. 이와 관련하여 가장 중요한 properties는 다음과 같습니다.

**exports** - declaration의 부분집합.

이것은 다른 모듈의 component template안에서 표시되고, 사용됩니다.

export할 때 여러 자료형을 가진다. Classes, function, values

**imports** - 모듈 내의 component template에서 필요에 의해 선언 된 class가 exported 된 다른 모듈과 의존관계 모듈.

- import 시 핵심 모듈은 @를 붙인다. @angular/core, @angular/common

**providers** - service의 creator.

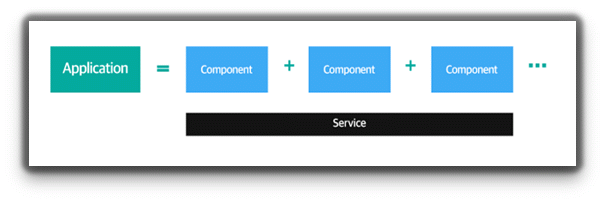
이 모듈은 service 중에서도 global collection 에 기여한다.

이것은 app의 모든 부분에 접근한다.

**bootstraping** : (Application Level에서) 최초 수행, 가동할 Component입니다. root component라고 불리는 main application view 이다. 다른 모든 app views를 host 하며

오직 root module만 이 bootstrap property로 설정 됩니다.

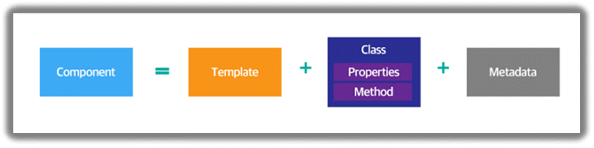
***3.2 Component***



Angular.js 2.0 의 Application을 이루고 있는 것은 컴포넌트(Component)입니다.

Component는 'view'라고 불리는 화면 중 일부분을 컨트롤하는 역할을 하고 있습니다.

컴포넌트들이 기능과 목적에 맞게 하나씩 작성되고, 결할 될 수 있으며, 컴포넌트 단위로 Customising을 하거나 별도의 기능을 넣고 테스트 할 수 있습니다.



컴포넌트는 Template + Class(Properites + Method) + Metadata로 구성되고 있습니다.

- 컴포넌트를 이용하는 이유

컴포넌트는 탬플릿을 위한 실행 문맥으로 이해할 수 있습니다. 컴포넌트를 이용하는 이유는 많은 이유가 있지만, 대략적으로 생각해보면 다음과 같습니다.

1. 코드량을 줄이는 목적

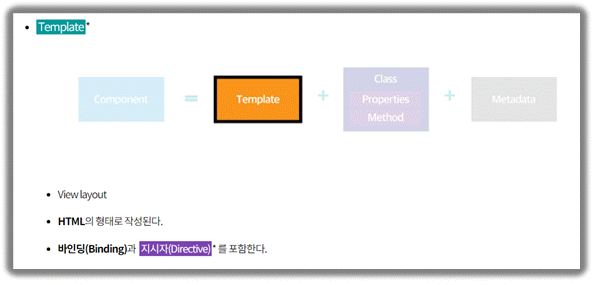
2. 코드의 기능을 표현하기 위한 목적

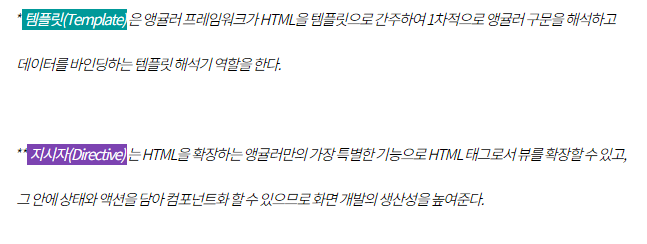
3. 내부적인 자세함을 추상화(Encapsulate) 하려는 목적

4. 엘리먼트를 재사용함으로서 생산성을 높이기 위한 측면

***3.3 Template***

Component의 view를 정의할 수 있습니다.





template은 우리가 알고 있는 HTML의 형태와 같고, Angular에게 어떻게 component를 렌더링 할지에 대해 알려주는 역할을 수행 합니다.

Template는 기본적으로 HTML처럼 보이지만 몇 가지 다른 점이 있습니다.

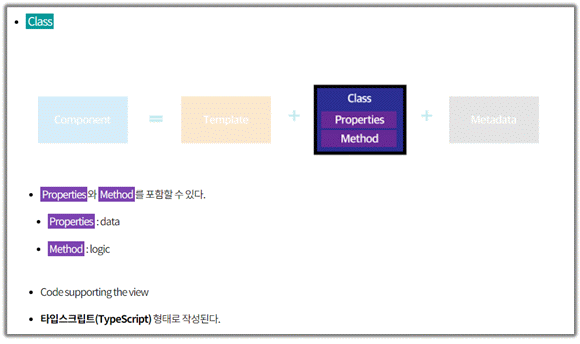
다음 코드를 살펴보면



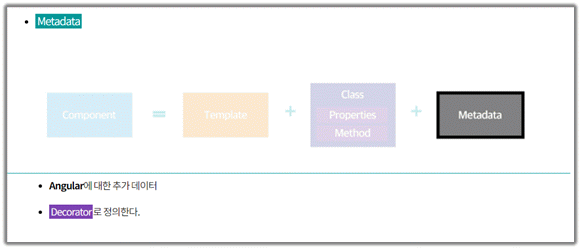
template의 코드는 전형적인 HTML 요소(<li>, <p>) 등을 사용하고 있지만, 이 외에 약간 다른 부분을 요소도 첨가 되어 있습니다. \*ngFor , {{hero.name}} , (click) , [hero] 등의 부분이 바로 전형적인 HTML과 다른 angular의 syntax 입니다.

그리고 가장 하단에 보이는 <hero-detail> 부분은 Angular의 template syntax를 활용하여 작성된 것이다. 이 <hero-detail> 태그는 (예제에서) HeroDetailComponent라는 새로운 component를 표현하기 위한 custom element 입니다.

***3.4 Class***



***3.5 Metadata***

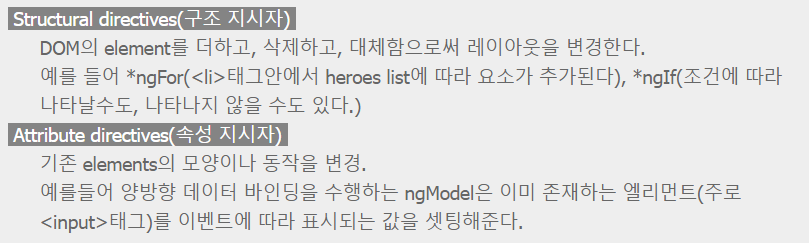


Metadata는 컴포넌트를 구성하는 정보를 JSON 객체로 정의한 것이며, Angular에게 클래스를 처리하는 방법에 대해 말해준다.

클래스에 metadata를 붙임으로 컴포넌트가 된다. TypeScript에서는 decorator(코드 내에서 필요에 따라 부가적인 정보나 속성을 표현하기 위한 언어적 장치)를 사용하여 Metadata를 붙입니다.

***3.6 Directives***

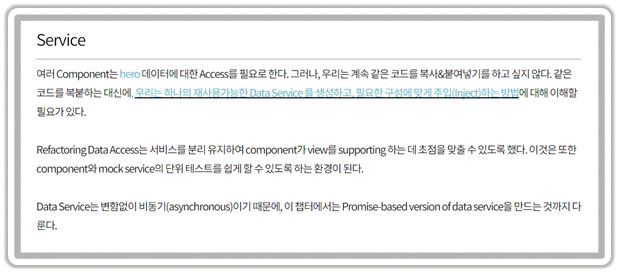
Directives는 directive metadata의 클래스입니다.



Angular 2.0 template은 동적인 형태입니다. Angular가 랜더링 할 때, 주어진 directives 지시에 따라 DOM을 변환 합니다. Angular 2.0에 채택된 TypeScript에서는 metedata를 클래스에 붙이기 위하여 @Directive 데코레이터를 사용한다. 컴포넌트의 부분과 템플릿의 일부를 조정하기 위한 방법으로 데이터 바인딩을 지원한다. 개발자는 어떻게 양쪽을 연결하는지 Angular에게 알리기 위하여 HTML 템플릿에 바인딩 마크업을 추가 합니다.

***3.6 Service***

* 특정 기능을 담당하는 객체들
* 공통된 특정 작업을 수행하는 Singleton객체
* Singleton은 인스턴스가 1개만 존재하는 디자인 패턴



서비스는 컴포넌트 내에서 재사용이 빈번한 기능을 서비스로 정의 합니다.

서비스는 응용 프로그램이 요구하는 값, 함수, 기능을 포괄하는 광범위한 범주이다. 거의 모든 것이 서비스가 될 수 있습니다. 개발자는 컴포넌트 클래스가 가볍기를 원하기에 컴포넌트를 서버로부터 데이터를 가져오거나, 사용자 입력을 검증하거나, 콘솔에 출력하지 않습니다. 대신 이러한 일들은 서비스에게 위임한다. 서비스에 로직을 반영하고 컴포넌트에 해당 서비스를 사용할 수 있도록 하는 것이 **의존성 주입(Dependency Injection)** 입니다.

***3.7 의존성 주입(Dependency Injection)***

* Dependency injection을 담는 컨테이너 역할

의존성 주입(Dependency Injection, D.I)는 코드를 조직화하고 단위별로 테스트 가능하게 합니다. 완전히 형성된 종속 클래스의 새로운 인스턴스를 공급하는 방법 입니다. 종속성의 대부분이 서비스이다. Angular는 필요한 서비스와 컴포넌트를 제공하기 위하여 D.I를 사용 합니다. 생성자 매개 변수의 유형을 보고 필요한 컴포넌트를 서비스에게 알려 줍니다.

Angular가 컴포넌트를 만들 때, 먼저 Injector에게 컴포넌트가 필요하다고 서비스에 대한 요청을 한다. Injector는 이전에 생성한 서비스 인스턴스의 컨테이너를 유지 합니다. 요청 된 서비스 인스턴스가 컨테이너에 없으면 Injector를 만들고 Angular에게 서비스를 반환하기 전에 컨테이너에 추가한다. 요청 된 모든 서비스가 사용되고 반환 된 경우, Angular는 인수로부터 서비스와 컴포넌트의 생성자를 호출 할 수 있다. Injector가 없는 경우 사전에 서비스의 Providers에 등록해야한다. provider는 서비스를 생성하거나 반환 할 수 있다.

어플리케이션의 모든 곳에서 동일한 서비스의 인스턴스를 사용 할 수 있도록 루트 수준에서 부트스트랩을 추가할 수 있으며 또, 컴포넌트 수준에서 속성으로 Provider를 등록할 수 있습니다.

**4. Angular 2.0 참고 사이트**

Angular 2.0은 최근에 정식 버전이 출시가 된 관계로 아직은 국내에선 많은 정보가 있지 않으며 주로 해외 공식 사이트에서의 정보를 번역한 내용이 대부분을 이루고 있습니다.

1. <https://angular.io/docs/ts/latest/quickstart.html>

- 앵귤러 2.0 튜토리얼인 quickstart 학습할 수 있는 공식 사이트 입니다.

2. <http://courseware.codeschool.com.s3.amazonaws.com/accelerating-through-angular-2/slides/CodeSchool-AcceleratingThroughAngular2.pdf>

- 앵귤러 2.0 학습 내용이 담긴 PDF 뷰어 홈페이지 입니다.

3.

http://courseware.codeschool.com.s3.amazonaws.com/accelerating-through-angular-2/slides/CodeSchool-AcceleratingThroughAngular2.pdf

- Angular 2.0에 대한 각종 정보를 볼 수 있는 Slide Share 홈페이지 입니다.

4. http://cusonar.tistory.com/6

- Angular 2.0 기초적인 시작을 제공해주는 개인 블로그

**5. 출처**

<https://www.slideshare.net/netil/angularjs-43578411>

- 사이드쉐어

http://visualize.tistory.com/458 [시각화를 배우고 정리합니다]

- ‘비주얼라이즈’ 개인 블로그

- Angular 2.0 구조 참고

http://k9e4h.tistory.com/114 [Kim's :D]

* Kim’s 개인 블로그
* Angular 2.0 구조 참고

<http://courseware.codeschool.com.s3.amazonaws.com/accelerating-through-angular-2/slides/CodeSchool-AcceleratingThroughAngular2.pdf>

- PDF 학습 홈페이지

<http://www.notforme.kr/archives/tag/typescript>

- 타입 스크립트 자료 참고

<http://k9e4h.tistory.com/92>

- Angular 2.0 의 개요