Angular 2

Angular2란?

1. Angular2는 JavaScript Framework다.
2. Client Side Application을 빌드하는데 사용한다.
3. HTML, CSS, JavaScript and TypeScript를 사용한다.

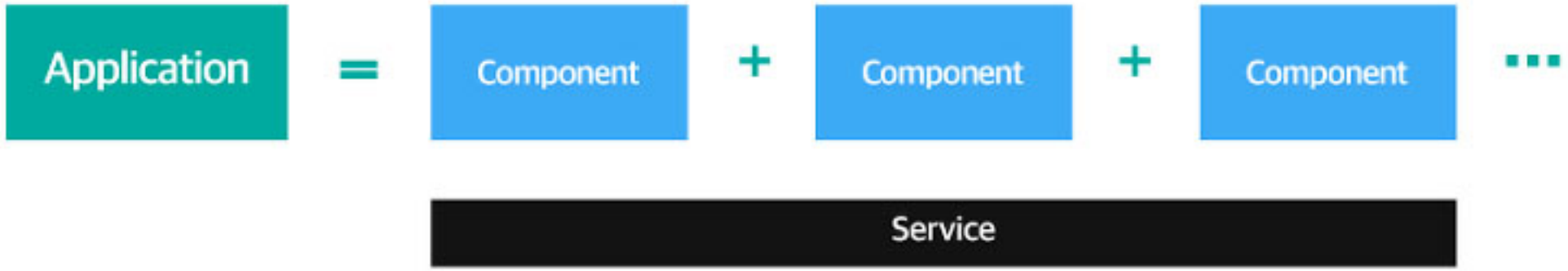
왜 Angular2 인가?

1. Expressive HTML: ngIf, ngFor, ngSwitch 등
2. 강력한 데이터 바인딩(Data Binding)
3. Support Module By Design
4. Back-End 통합 서비스 내장

왜 사용해야 하는가?

1. 빠른 구축이 가능하다.
2. 새로운 기능을 사용할 수 있다.
   (using new features – classes, object)
3. 간결해진 API.
4. 개선된 생산성.

Angular2의 구조



위 사진과 같이 Angualr2의 Application을 이루는 것은 각 컴포넌트(Component)다.
이 컴포넌트들이 기능과 목적에 맞게 하나씩 작성되고, 결합될 수 있으며,
컴포넌트 단위로 customizing을 하거나 별도의 기능을 넣고 빼며 테스트할 수 있다.

Component라는 용어의 사전적 정의를 살펴보면, **'구성요소, 부품'** 이라는 뜻으로 정의 되어있다.

Angular2 Component

Component는
Template + Class (Properties와Method를 포함할 수 있음) + Metadata
로 이루어 진다.

Component Template

1. View layout
2. HTML, CSS의 형태로 작성된다.
3. 바인딩(Binding)과 지시자, Pipe를 포함한다.

Component Class

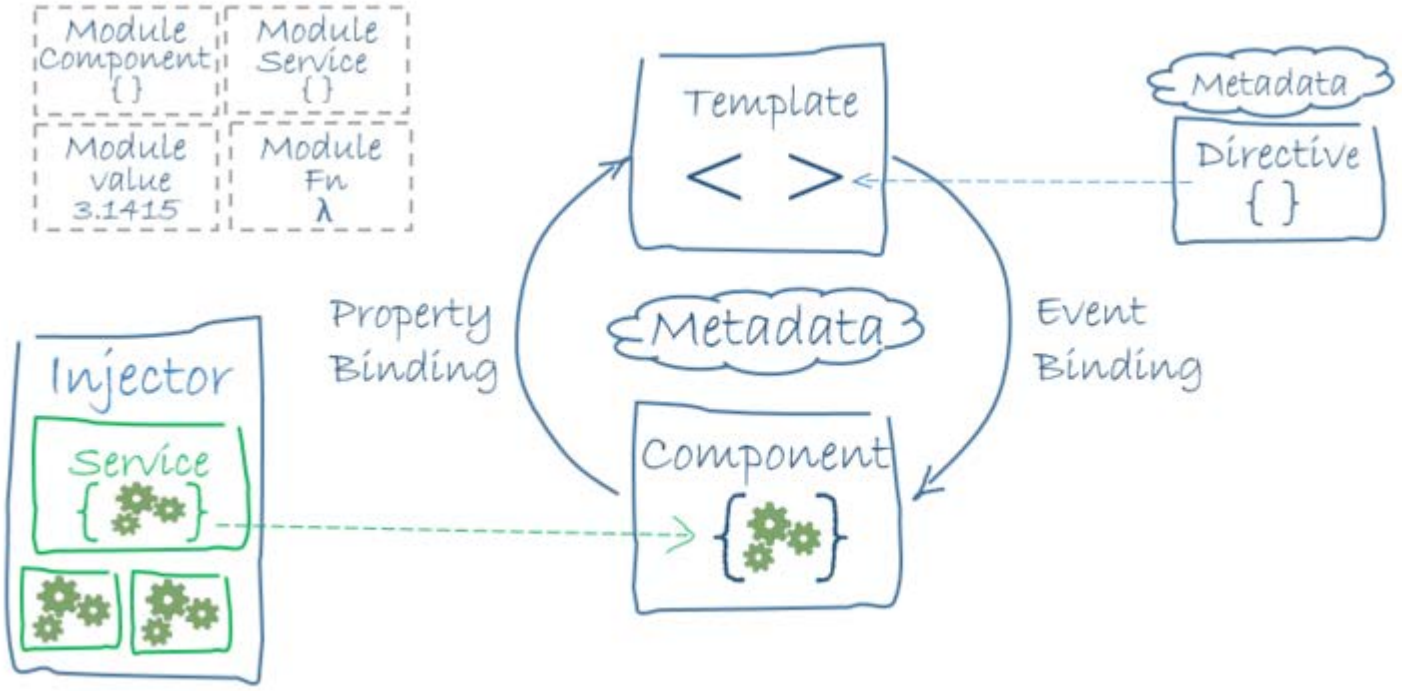
property와 constructor, method를 포함할 수 있다.

1. property: 속성
2. constructor: 생성자
3. method: 기능, Logic

Component Metadata

1. Angular의 추가 데이터
2. Decorator로 정의

Angular2


Module
Component
{}
Module
Service
{}
Module
value
3.1415
Module
Fn
λ
Template
< >
Metadata
Directive
{}
Property
Binding
Metadata
Event
Binding
Injector
Service
{}
Component
{}

프로젝트 사전 설정

프로젝트 폴더 만들기 및 구성

1. 프로젝트 폴더 만들기
2. 설정 파일 만들기
3. 패키지 설치

프로젝트 사전 설정

1. 프로젝트 폴더 만들기

mkdir angular-quickstart

cd angular-quickstart

프로젝트 사전 설정

2. 설정파일 작성

전형적인 Angular 프로젝트는 몇 가지 설정 파일이 필요합니다.

1. **package.json**: 프로젝트 npm 패키지 종속성을 확인합니다.
2. **tsconfig.json**: TypeScript 컴파일러가 프로젝트 파일에서 JavaScript를 생성하는 방법을 정의합니다.
3. **typings.json**: TypeScript 컴파일러가 기본적으로 인식하지 않는 라이브러리의 추가 정의 파일을 제공합니다.
4. **systemjs.config.js**: 응용 프로그램 모듈의 위치를 모듈 로더에 정보를 제공, 필요한 모든 패키지를 등록합니다. 또한 이후의 문서의 예에서 필요한 다른 패키지가 포함되어 있습니다

프로젝트 사전 설정

3. 설정파일 작성

명령 줄에서 npm을 사용하여 package.json에 기재되어 있는 패키지를 설치합니다.

```
npm install
```

설치 중에 npm WARN 메시지가 나타날 수 있습니다.

빨강색으로 표시되는 npm ERR!이 존재하지 않으면 설치 성공입니다.

```
angular-quickstart
 ├ node_modules ...
 ├ typings ...
 ├ package.json
 ├ systemjs.config.js
 ├ tsconfig.json
 └ typings.json
```

프로젝트 사전 설정
4. 최종 구조

angular2-quickstart

```
├ app
|    ├ app.component.ts
|    ├ app.module.ts
|    └ main.ts
├ html
|    ├ sub.html
├ node_modules ...
├ typings ... 이하는 전 페이지와 동일
```

Angular2 Polyfill

core-js / shim.js

```
<script src="node_modules/core-js/client/shim.min.js"></script>
```

API를 가로채어, Parameter를 변경하고, 동작 자체를 변경하는 것을 말한다. 웹에 적용해 본다면 구 버전 혹은 표준 스펙과 다르게 구현된 Browser, 대표적으로 IE라 할 수 있다)의 API를 가로채서 표준에 맞게 변경하는 작업을 한다.

Angular2 Polyfill

zone.js

```
<script src="node_modules/zone.js/dist/zone.js"></script>
```

Zone.js는 브라우저의 addEventListener나 setTimeout, XMLHttpRequest, 등등을 몽키 패치를 해둔다.

그리고 어플리케이션 로직에서 이것들을 호출하게 되면 Zone.js이 설치해둔 패치를 호출하게 되고, 이를 통해 이런 호출들을 hooking할 수 있도록 해 주는 것이다.

이런 호출을 hooking할 수 있다는 것은, 이런 호출이 일어나는 시점을 감지할 수 있다는 의미를 포함하므로, 이 hooking 부분에서 Change Detection이 시작되도록 하면 되는 것이다.

Angular2 Polyfill

reflect-metadata.js

<script src="node_modules/reflect-metadata/Reflect.js"></script>

ECMAScript 7에서 제안한 데코레이터 문법을 사용하기 위한 라이브러리이다.

데코레이터 문법은 AngularJS2에서 다음과 같이 사용하는데 @Component가 데코레이션이고 일종의 Syntax sugar라 한다.

```
@Component({
    selector: ‘app-com’,
    template: ``
  …..
```

Angular2 Polyfill

system.js

<script src="node_modules/systemjs/dist/system.src.js"></script>

아래 코드가 이 라이브러리를 사용하는 부분인데 첫번째 라인에서 systemjs를 위한 환경 설정 파일을 로딩하고 그 다음에 프로젝트에서 구현한 자바스크립트를 로딩하는 과정이다.

```
<script src="systemjs.config.js"></script>

<script>

  System.import('app/chap04/A01.component').catch(function(err){ console.error(err); });

</script>
```

전체 파일에 대해선 systemjs.config.js 에서 설정하고 그 다음 코드에서는 위에서 설정한 정보를 바탕으로 모든 자바스크립트 코드를 한번에 로딩한다.