목차

typescript로 시작하는 helloworld

> index.html

```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://unpkg.com/core-js@2.4.1/client/shim.min.js"></script>
        <script src="https://unpkg.com/zone.js@0.8.5"></script>
        <script src="https://unpkg.com/systemjs@0.19.47/dist/system.src.js"></scri
pt>
        <script src="https://unpkg.com/typescript@2.3.3"></script>

        <script>
            System.config({
                transpiler : 'typescript',
                map : {
                    'rxjs' : 'https://unpkg.com/rxjs@5.3.0',
                    '@angular/core' : 'https://unpkg.com/@angular/core@4.1.0',
                    '@angular/common' : 'https://unpkg.com/@angular/common@4.1.0',
                    '@angular/compiler' : 'https://unpkg.com/@angular/compiler@4.1
.0',
                    '@angular/platform-browser' : 'https://unpkg.com/@angular/plat
form-browser@4.1.0',
                    '@angular/platform-browser-dynamic' : 'https://unpkg.com/@angu
lar/platform-browser-dynamic@4.1.0'
                }
            });
            System.import('main.ts');
        </script>
    </head>
    <body>
        <hello-world></hello-world>
    </body>
</html>
```

main.ts

```
import { Component } from '@angular/core';
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

// 컴포넌트 선언
@Component({
    selector : 'hello-world',
    template : '<h1>Hello {{ name }}!</h1>'
})
class HelloWorldComponent {
    name : string;

    constructor () {
        this.name = 'Angular';
    }
}

// 모듈 선언
@NgModule({
    imports : [BrowserModule],
    declarations : [HelloWorldComponent],
    bootstrap : [HelloWorldComponent]
})
export class AppModule {}

// 부트스트랩
platformBrowserDynamic().bootstrapModule(AppModule)
```

es5,6

https://github.com/han41858/Angular-Development-with-TypeScript/tree/master/chapter02/2.hello-world-es5

https://github.com/han41858/Angular-Development-with-TypeScript/tree/master/chapter02/3.hello-world-es6

애플리케이션 실행하기

nodejs설치(npm 설치)

Angular CLI설치

`npm install -g @angular/cli`

my-app 프로젝트 생성

`ng new my-app`

server 실행

```
cd my-app
ng serve --open
```

프로젝트 구조

```
src
    app
        app.component.css
        app.component.html
        app.component.spec.ts
        app.component.ts
        app.module.ts
    assets
        .gitkeep
    environments
        environment.prod.ts
        environment.ts
    favicon.ico
    index.html
    main.ts
    polyfills.ts
    styles.css
    test.ts
    tsconfig.app.json
    tsconfig.spec.json
```

모듈

컴포넌트, 서비스, 디렉티브를 하나로 모은 것

> app.module.ts

```
@NgModule({
//이곳에 선언되는 항목은 애플리케이션 전역에서 사용가능(사용자 컴포넌트)
  declarations: [
    AppComponent,
    HeroesComponent,
    HeroDetailComponent,
    MessagesComponent
  ],
  //브라우저를 사용하는 모든 앱은 BrowserModele을 불러와야 함(기본 모듈)
  imports: [
    BrowserModule,
    FormsModule,
    AppRoutingModule
  ],
  providers: [ HeroService, MessageService ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

모듈을 추가하면 포함된 컴포넌트들을 사용가능.

컴포넌트

뷰+클래스

> heroes.component.ts

```
// 뷰를 정의하는 부분
@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.css'],
  encapsulation: ViewEncapsulation.None
})
// 로직을 구현하는 부분
export class HeroesComponent implements OnInit {

  heroes: Hero[];

  selectedHero: Hero;
  ngOnInit() {
    this.getHeroes();
  }
  onSelect(hero: Hero): void {
    this.selectedHero = hero;
  }
  getHeroes(): void {
    this.heroService.getHeroes()
      .subscribe(heroes => this.heroes = heroes);
  }

  constructor(private heroService: HeroService) { }


}
```

디렉티브

컴포넌트와 비슷하지만 뷰를 갖지 않는 경우도 있으므로 컴포넌트라고 할 수 없다.

```
@Directive({
    selector : 'input[log-directive]',
    host : {
        '(input)' : 'onInput($event)'
    }
})
class LogDirective {
    onInput (event) {
        console.log(event.target.value);
    }
}
```

선언한 디렉티브는 다음과 같이 사용

`<input log-directive>`

```
@Component({
    selector : 'hello-world',
    template : '<h1>Hello {{ name }}!</h1>' +
    '<input log-directive/>'
})
class HelloWorldComponent {
    name : string;

    constructor () {
        this.name = 'Angular';
    }
}
```

데이터바인딩 기초

컴포넌트의 프로퍼티값을 템플릿에 표시

> app.component.html

```
<h1>{{title}}</h1>
<app-heroes></app-heroes>
```

> app.component.ts

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

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

HTML 엘리먼트값을 컴포넌트 프로퍼티로 바인딩

> heroes.component.html

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

> heroe-detail.component.ts

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

@Component({
  selector: 'app-hero-detail',
  templateUrl: './hero-detail.component.html',
  styleUrls: ['./hero-detail.component.css'],
  encapsulation: ViewEncapsulation.None
})
export class HeroDetailComponent implements OnInit {

  @Input() hero: Hero;
  constructor() { }

  ngOnInit() {
  }

}
```

> hero.ts

```
export class Hero {
  id: number;
  name: string;
}
```

엘리먼트에서 발생한 이벤트를 이벤트핸들러에 연결

> heroes.component.html

```
 <li *ngFor="let hero of heroes"
      [class.selected] = "hero === selectedHero"
      (click)="onSelect(hero)">
```

> heroes.component.ts

```
@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.css'],
  encapsulation: ViewEncapsulation.None
})

export class HeroesComponent implements OnInit {

  heroes: Hero[];
  selectedHero: Hero;
  ngOnInit() {
    this.getHeroes();
  }
  onSelect(hero: Hero): void {
    this.selectedHero = hero;
  }
  getHeroes(): voi...
```

SystmeJS 모듈로더

`<script>` 문제점

1. 개발자 직접관리 힘듬(시간이 지나 필요없는 부분이 생길 수도 있다.)
2. 태그의 순서가 문제가 될 수 있다.

```
<script>
        // SystemJS 설정
        System.config({
            transpiler : 'traceur',
            map : {
                'traceur' : 'https://unpkg.com/traceur@0.0.111/bin/traceur.js'
            }
        });
        // 모듈 로드
        Promise.all([
            System.import('./es6module.js'),
            System.import('./es5module.js')
        ]).then(function (modules) {
            var moduleNames = modules
                .map(function (m) { return m.name; })// 각각의 모듈에서 name프로퍼티 추출
                .join(', '); // 불러운 모듈의 name프로퍼티를 문자열 하나로 연결
            console.log('The following modules are loaded: ' + moduleNames);
        });
</script>
```

Promise를 이용해서 모듈로드에 성공하면 then()을 실행한다.

- baseURL(자동으로 /app가 기본 경로가 된다.) `System.config({baseURL : '/app'});`
- defaultHSExtension (자동으로 .js가 붙는다.)
  `System.config({ defaultJSExtensions: true });`
- map (미리 지정한 이름으로 모듈을 참조할 수 있는 맵을 만든다.)
  `System.config({map: {'es6module.js: 'esSixModule.js' } });`
- packages (packages 단위로 설정)
- paths (map과 비슷한데 추가로 특수문자 매칭을 지원)
- transpiler (어떤 코드변환기를 사용할 지)
- typescriptOprions (typescript 컴파일러 옵션 지정하고 싶을 때)