



고용노동부



한국산업인력공단



국가인적자원개발컨소시엄



# Hybrid App 개발 with IONIC

2018.5.28 ~ 6.1

백명숙



## 과정개요

■ IONIC Framework의 개념을 알아보고, IONIC CLI을 사용하여 IONIC Project 생성 방법을 살펴본다. 단말 디바이스 전용 에뮬레이터인 IONIC View를 사용하여 단말에서 IONIC App을 테스트 해 볼 수 있다. IONIC UI 컴포넌트들의 사용법을 이해하고, HttpClient를 사용한 RESTful App도 작성 해보는 과정입니다.

## 과정목표

■ IONIC Framework의 다양한 UI 컴포넌트를 사용한 IONIC App를 작성할 수 있는 능력을 키우는 것이 이 과정의 목표입니다.

# 러닝 맵



## 강사 프로필

성명	백명숙
소속 및 직함	휴클라우드 전임강사
주요 경력	일은시스템(제일은행 IT자회사), Sun MicroSystems 교육서비스, 인터넷커머스코리아
강의/관심 분야	Java, Framework, Python, Data Science, Machine Learning, Deep Learning
자격/저서/대외활동	Java기반 오픈소스 프레임워크/NCS 개발위원

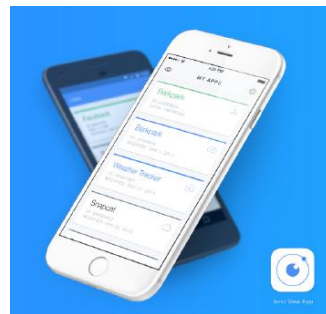


LO	커리큘럼
1. IONIC Framwork 소개	<ul style="list-style-type: none"> <li>- IONIC Framework 개요 및 History</li> <li>- 개발환경 설정</li> <li>- IONIC 프로젝트 생성</li> <li>- IONIC View App (디바이스 전용 에뮬레이터)</li> </ul>
2. IONIC 주요 요소	<ul style="list-style-type: none"> <li>- IONIC UI 컴포넌트의 사용</li> <li>- SQLite Device DB의 사용</li> <li>- Email Composer / SMS / Call Number의 사용</li> </ul>
3. IONIC RESTful App	<ul style="list-style-type: none"> <li>- HttpClient를 사용한 IONIC RESTful App 작성</li> </ul>

# IONIC 프레임워크 소개

# IONIC Framework

- I Drifty에서 개발한 HTML5 기반 하이브리드 앱 개발 프레임워크  
모바일 하이브리드 앱 개발을 위해 많이 사용함  
클라우드 개발환경, UI 도구 등을 다양하게 지원함
- IONIC은 Native, Progressive 웹 앱을 손쉽게 개발할 수 있는 오픈 소스 모바일 SDK 입니다.





# 모바일 **App** 종류 및 **App** 프레임워크

---

	Native App	Web App	Hybrid App
디바이스 접근	가능	부분적 가능	가능
앱 속도 (성능)	매우 빠름	보통	빠름(Native에 근접)
개발비용	높음	낮음	보통
앱 스토어 이용	가능	불가능	가능

- jQuery Mobile (<http://jquerymobile.com>)
- Sencha Touch (<http://www.sencha.com/products/touch>)
- Kendo UI (<http://www.telerik.com/kendo-ui>)
- DHTMLX Touch (<http://dhtmlx.com/touch>)
- famo.us (<http://famous.co>)

# IONIC Framework

---

- Hybrid App을 위한 Front-End 프레임워크
- 모바일 최적화된 HTML, CSS, JS 컴포넌트 지원
- Angular를 이용하여 편리한 개발과 빠른 성능 제공
- Cordova를 이용한 다양한 네이티브 기능 지원
- Angular + JavaScript + TypeScript + HTML5 + Cordova
- 무료 & 오픈 소스 ( MIT License )
- 크로스 플랫폼 지원 ( Android, iOS,, Windows 등)
- 다양한 네이티브 플러그인 지원 ( Cordova / PhoneGap )
- 멀티 디바이스의 다양한 해상도 대응



# IONIC History

---

- 2013년 11월 알파버전 발표
- 2014년 3월 1.0 베타버전 발표
- 2015년 5월 1.0 정식버전 발표
- 2016년 11월 2.0 정식버전 발표
- 2017년 5월 3.0 정식버전 발표
- 2018년 현재 3.9.2 버전
- 3백만 개의 앱이 Ionic으로 개발
- 5백만명의 Ionic 개발자 활동

# 개발환경 설정

# 개발 환경 설정 : **nodejs**와 **ionic** 설치

---

- CLI의 도구는 대부분 Node를 기반으로 하며 npm을 통해 관리됩니다.  
노드와 NPM을 컴퓨터에 설치하는 방법은 NodeJS 프로그램을 설치함
- 1.노드의 LTS 버전을 설치  
<https://nodejs.org/ko/download/>
- 설치 확인하려면 `npm --version` 및 `node --version`을 실행하십시오.
- 2.Ionic CLI and Cordova 설치  

```
$ npm install -g ionic cordova
```

  
설치 확인하려면 `ionic info` 를 실행하십시오

# 개발 환경 설정 : **git / webstorm** 설치

---

- 3. Git Client 설치

<https://git-scm.com/downloads>

- 4. WebStorm 설치

<https://www.jetbrains.com/webstorm/download/download-thanks.html>

- 5. Git과 Ionic Pro에 각자의 id 만들어서 로그인 가능하게 하기.

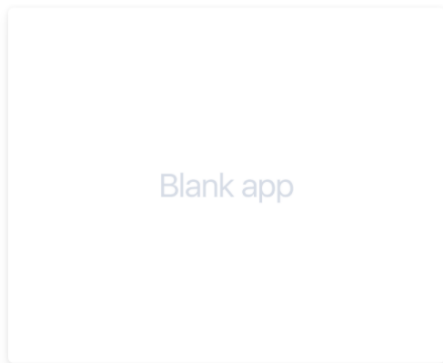
# IONIC 프로젝트 생성

# Ionic 프로젝트 생성

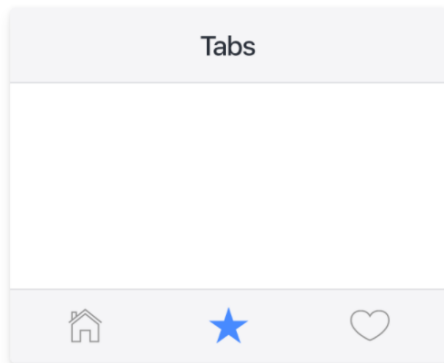
- 1. Ionic App 프로젝트 생성하기

```
$ ionic start ionic-myApp tabs
```

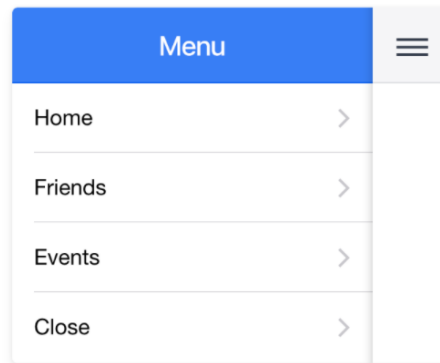
[template option]



```
$ ionic start myApp blank
```



```
$ ionic start myApp tabs
```



```
$ ionic start myApp sidemenu
```



# Ionic 프로젝트 생성

---

- 1-1.Ionic App 생성

1) Would you like to integrate your new app with Cordova to target native iOS and Android? No

2) Install the free Ionic Pro SDK and connect your app? No

1-2.Ionic App 생성 폴더로 이동, 개발 서버 start

```
cd ./ionic-myApp
```

```
$ ionic serve
```

# Ionic 프로젝트 실행

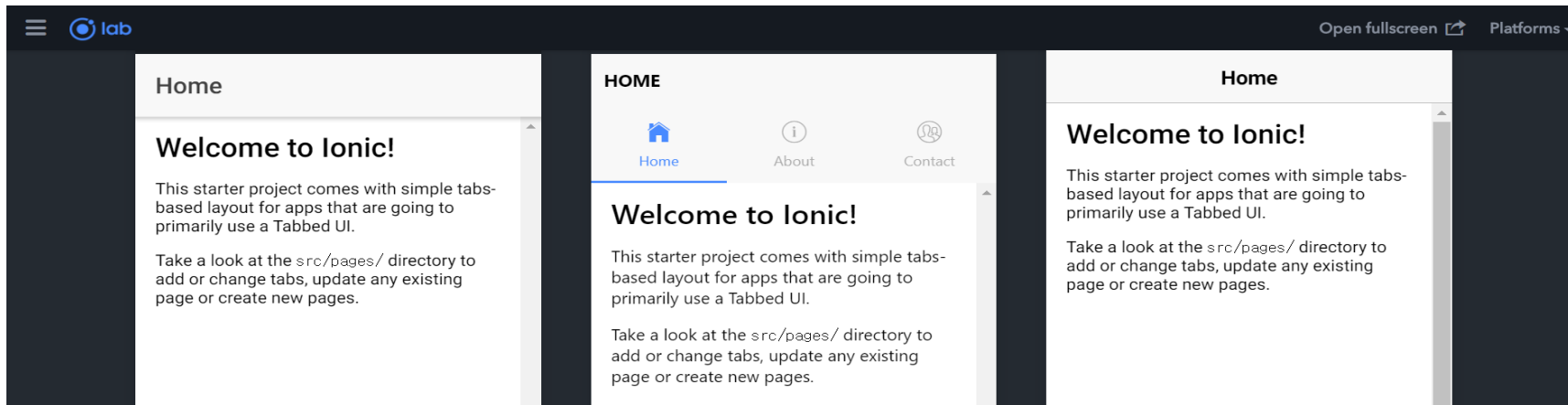
- 2.Ionic App 실행하기

```
$ cd ionic-myApp
```

```
$ ionic serve
```

- 2.1 플랫폼 별 Look and feel 확인

```
$ ionic serve -lab (or -l )
```



# IONIC View 사용하기

## : Device 전용 에뮬레이터

# Git repository 생성

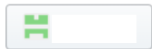
- Git에 Repository 생성

<https://github.com/각자의ID?tab=repositories>

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



Repository name

/ ionic-myApp ✓

Great repository names are short and memorable. Need inspiration? How about [special-octo-barnacle](#).

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

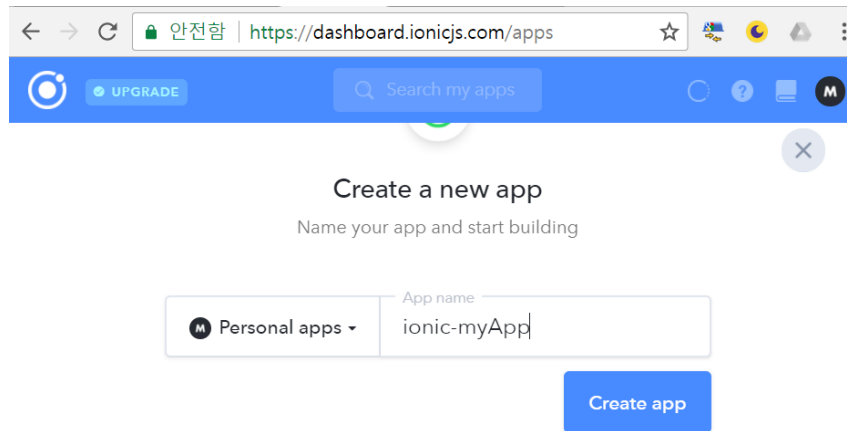
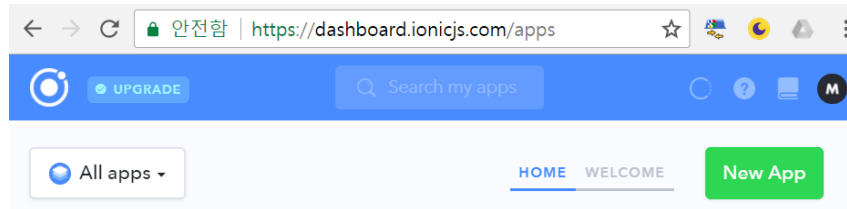
☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Create repository

# Ionic App 생성

[Ionic Dashboard\(https://dashboard.ionicjs.com/apps\)](https://dashboard.ionicjs.com/apps)에서 새로운 App 생성



# Ionic link

---

\$ ionic login (email 주소와 password 입력)

Ionic ssh setup

> Skip for now

\$ ionic link --pro-id 43311e0b

Which git host would you like to use? GitHub

Does the repository exist on GitHub? Yes

Which GitHub repository would you like to link? vega2k / ionic-myApp

Which would you like to do? Link to master branch only

cd ./ionic-myApp

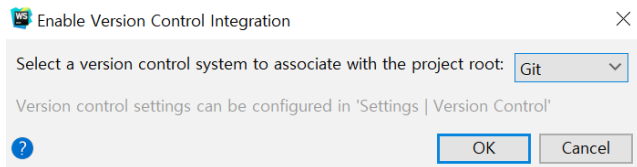
# Ionic App commit & push

---

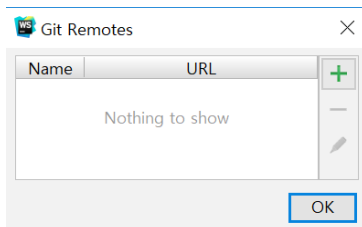
- > git add .
- > git status
- > git config --global user.email "you@example.com"
- > git config --global user.name "your.name"
- > git commit -m "first upload"
- > git remote add origin <https://github.com/vega2k/ionic-myApp.git>
- > git push --set-upstream origin master

# WebStorm에 git 설정

VCS -> Enable Version Control Integration



프로젝트 우클릭 메뉴 Git -> Repository -> Remotes  
생성한 Github Repository 주소를 입력해 준다.  
예를 들면) <https://github.com/각자의Id/ionic-myApp.git>





# commit & push

---

- 생성된 프로젝트 Git에 commit & push

## 1) Git -> Commit Directory

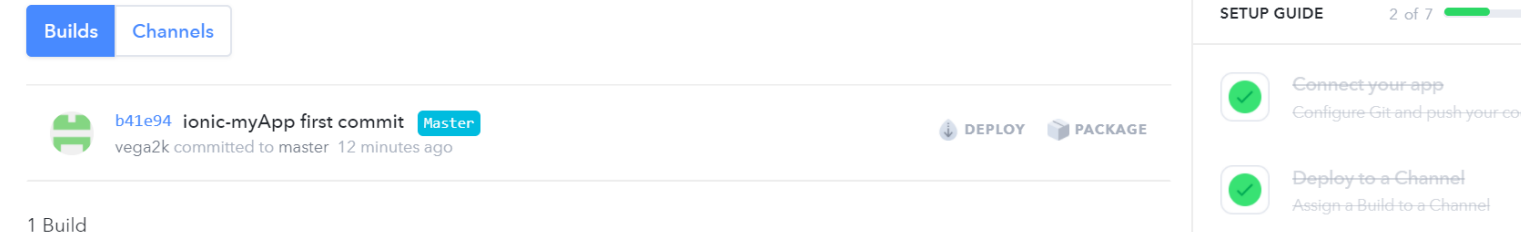
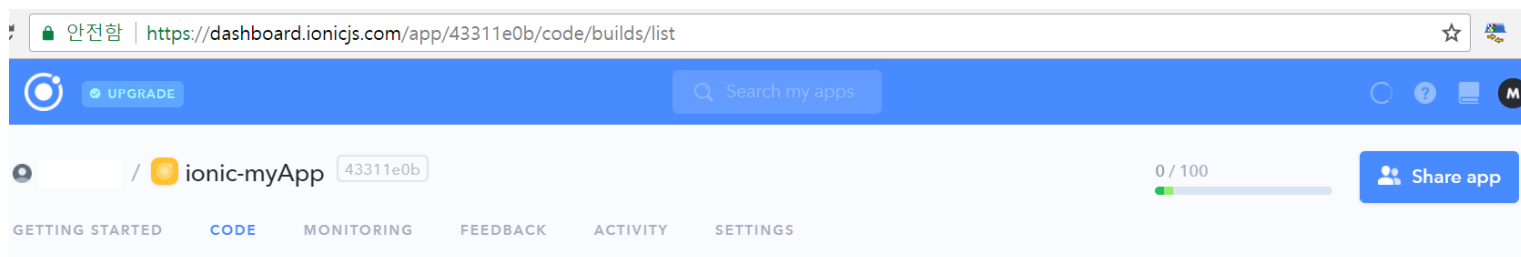
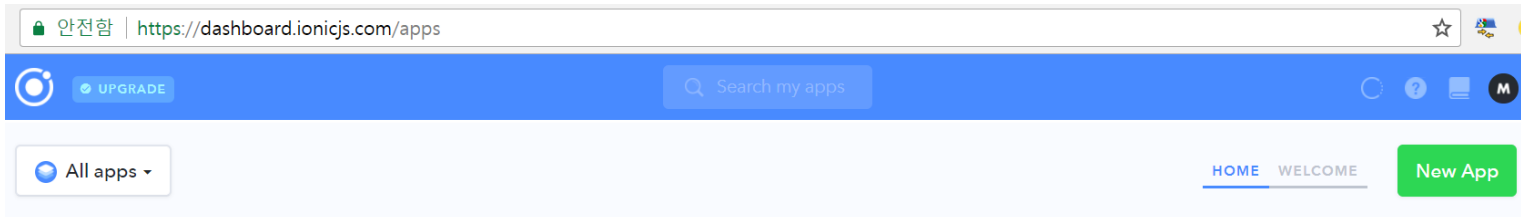
Commit Message 입력 후에 Commit and Push 선택

2) push 할 때 Remote URL에 git의 repository url를 지정해야 함

3) push 한 후에 <https://github.com/각자의ID/ionic-myApp> 에서 source 확인해야 한다.


# Ionic App Deploy & Packae 확인

- ionic dashboard 에서 등록한 App을 확인하고 Deploy와 PACKAGE 확인



# Ionic App 을 public 으로 변경

- App을 Deploypoy 수행하기 전에 channel을 public으로 변경하기
  - 1) Activity -> View Master -> SETTINGS -> **Make public** -> **Save channel**
  - 2) **stop sharing publicly** 로 변경된 것을 확인해야 함


 Master  
active commit b41e94

HISTORY SETTINGS

**Auto-deploys**  
Commits pushed to this branch will be automatically deployed to this channel.

**Edit channel**  
Change the channel name or color  
  
**Save channel**

**Administrative**  
Update the privacy and existantial settings.  
**Make this channel public**  
When public, anyone with the link or App ID can preview the contents of this channel  
**Make public**

 Master  
active commit b41e94

HISTORY SETTINGS

**Auto-deploys**  
Commits pushed to this branch will be automatically deployed to this channel.

**Edit channel**  
Change the channel name or color  
  
**Save channel**

**Administrative**  
Update the privacy and existantial settings.  
**Make this channel public**  
When public, anyone with the link or App ID can preview the contents of this channel  
**Stop sharing publicly**

# Ionic App 을 Deploy 하기

- App을 Deplopy 수행하기

1) CODE 탭에서 해당 App을 DEPLOY 하고 Snapshot deployed 메시지 확인

Deploy Build



**b41e94** ionic-myApp first commit  
vega2k committed to master 38 minutes ago

Channel

**Master**

active commit b41e94

[Edit Native Versioning](#)

Deploy

**Live Update** your apps on demand for any changes that do not require binary modifications. Roll back to previous versions, automatically apply updates, and control every aspect of the upgrade.

[Learn more](#)



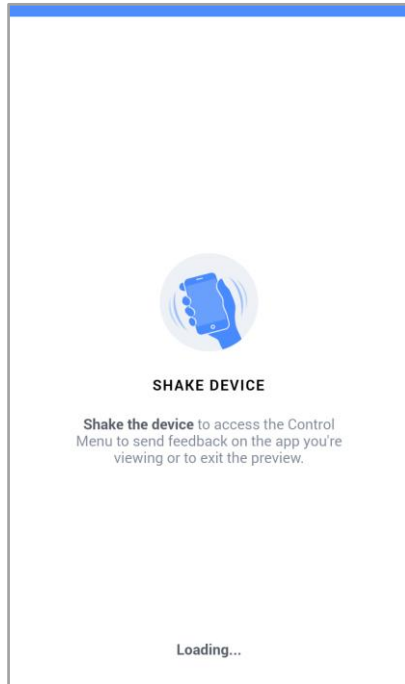
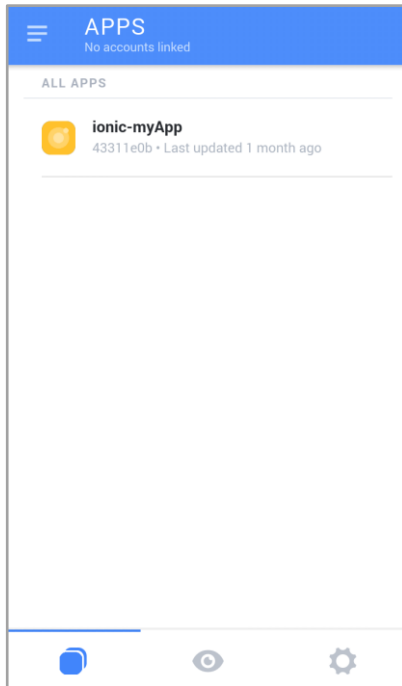
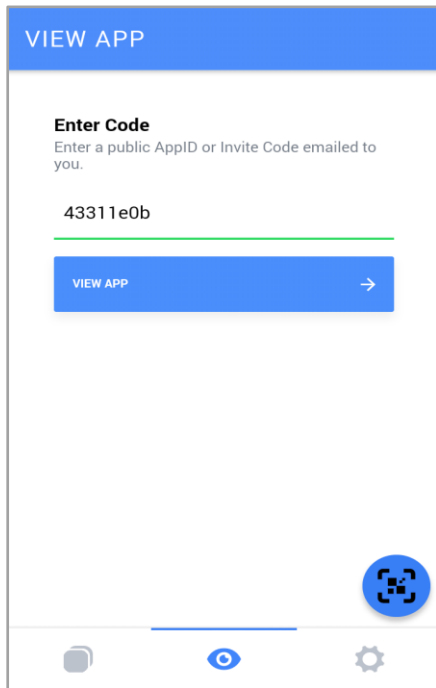
**Snapshot deployed**

Snapshot (commit **b41e94**) successfully deployed to **Master**

# Ionic View App 설치 및 확인

- 모바일 폰에서 Ionic View App 설치 및 확인

1) Ionic View 앱을 설치 후에 AppID를 입력 후에 모바일에서 App 실행



# IONIC UI 컴포넌트

UI컴포넌트를 사용한 앱 작성

# Ionic UI Components

- Lists UI Component

```
<ion-list>
```

```
  <button ion-item *ngFor="let item of items"
    (click)="itemSelected(item)">
    {{ item }}
```

```
  </button>
```

```
</ion-list>
```

## Lists

Pokémon Yellow

Super Metroid

Mega Man X

The Legend of Zelda

Pac-Man

Super Mario World

Street Fighter II

Half Life

Final Fantasy VII

Star Fox

Tetris



# Ionic UI Components

---

- More Components (<https://ionicframework.com/docs/components/>)

Action Sheets

Alerts

Badges

Buttons

Cards

Checkbox

DateTime

FABs

Gestures

Grid

Icons

Inputs

Lists

Loading

Menus

Modals

Navigation

Popover

Radio

Range

Searchbar

Segment

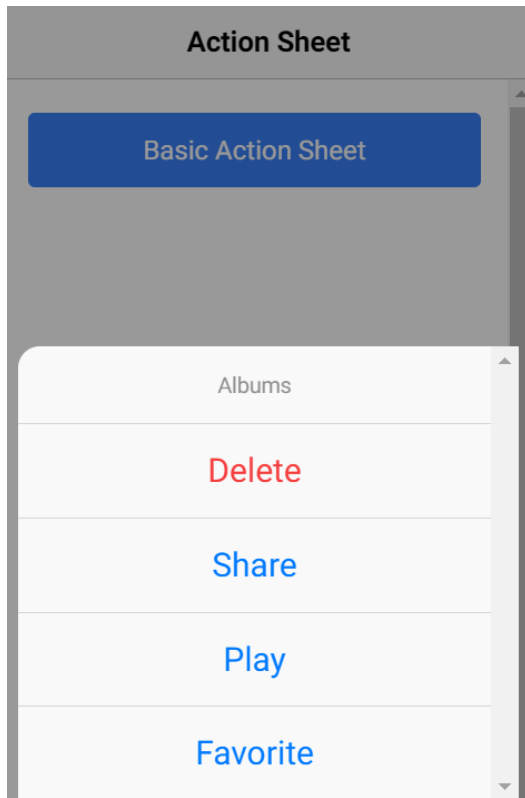
Select



# Ionic API

- Ionic API (<https://ionicframework.com/docs/api/>)  
: `ActionSheetController`

```
import { ActionSheetController } from 'ionic-angular'
export class MyClass{
  constructor(public actionSheetCtrl: ActionSheetController) {}
  presentActionSheet() {
    let actionSheet = this.actionSheetCtrl.create({
      title: 'Modify your album',
      buttons: [ {
        text: 'Destructive',
        role: 'destructive',
        handler: () => { console.log('Destructive clicked');}
      }, {
        text: 'Archive',
        handler: () => { console.log('Archive clicked');}
      }, {
        text: 'Cancel',
        role: 'cancel',
        handler: () => { console.log('Cancel clicked');}
      } ]
    });
    actionSheet.present();
  }
}
```



# Ionic CLI

---

- Ionic Command Line Interface(CLI)는 Ionic App를 개발을 위한 go-to tool이다.
- Ionic CLI Command List (<https://ionicframework.com/docs/cli/commands.html>)
- `ionic -help`
  
- 개발서버 실행 (`ionic serve`)  
    `cd [프로젝트명]`  
    `ionic serve`

: Webpack 빌드 과정이 끝나면 <http://localhost:8100> 접속하면 개발 서버를 확인할 수 있습니다

# Ionic CLI

---

- **\$ ionic generate**

**ionic generate [<type>] [<name>] 또는 ionic g [<type>] [<name>]**

: type은 generator의 타입 ( component, directive, page, pipe, provider, tabs)

: name은 generated 될 component의 이름

option

--no-module은 컴포넌트를 위한 NgModule을 생성하지 않는다.

--constants는 lazy-loaded page를 위한 page constant file를 생성한다.

예시)

\$ ionic generate component

\$ ionic generate directive

\$ ionic generate page

\$ ionic generate pipe

\$ ionic generate provider

\$ ionic generate tabs

\$ ionic generate component foo

\$ ionic generate page Login

\$ ionic generate page Detail --no-module

\$ ionic generate page About --constants

\$ ionic generate pipe MyFilterPipe

# Lazy Loading in ionic 3

- Ionic 2에서는 모든 구성 요소와 페이지가 app.module에서 로드 되므로 lazy load가 발생하고 앱을 열면 앱을 로드 하는 데 보통 15-20 초가 소요 되었음.
- Ionic3에서는 lazy loading과 boot time loading의 주된 결점을 해결함.
- Lazy Loading은 Encapsulated Module 방식으로 제어 할 수 있습니다.
- Encapsulated Module  
: 모든 페이지는 별도의 module을 가지며, 이 module은 각 페이지 구성 요소들을 묶어 준다.



# Lazy Loading in ionic3

---

- `pages/home` 디렉토리를 삭제한다.
- `app.module.ts`와 `app.component.ts` 에서 `HomePage`가 없어서 에러가 발생하는 코드를 수정한다.
- `$ ionic g page home`
  - : `pages/home` 디렉토리를 새로 생성해 준다.
- 생성된 `home` 디렉토리 내에 새로 생성된 `home.module.ts` 를 확인한다.
- `app.component.ts` 코드를 아래와 같이 수정한다.  
(`HomePage`에 `single quotation` 추가하여 `String` 문자열로 처리한다.)

```
rootPage:any = 'HomePage';
```

# Home page에서 **List** 컴포넌트 사용하기

---

(1) List 태그 안에 Item 태그를 여러 개 추가한다.

src/pages/home/home.html

```
<ion-content padding>
  <ion-list>
    <ion-item>
      <div>Item1</div>
    </ion-item>
    <ion-item>
      <div>Item2</div>
    </ion-item>
    <ion-item>
      <div>Item3</div>
    </ion-item>
    <ion-item>
      <div>Item4</div>
    </ion-item>
  </ion-list>
</ion-content>
```

# Data Binding 적용하기 : One-way Binding

---

(1) ngFor Directive와 Data Binding 적용

1) ngFor Directive

2) Data Binding (One-way Binding)

2.1) interpolation : {{item.name}}

2.2) event binding : (click)="itemSelected(item)"

2.3) property binding : [disabled]="disabledSwitch"

src/pages/home/home.html

```
<ion-content padding>
  <ion-list>
    <ion-item *ngFor="let item of items" (click)="itemSelected(item)">
      {{ item.name }}
    </ion-item>
  </ion-list>
  <button ion-button full [disabled]="disabledSwitch" (click)="clickButton($event)">Button</button>
</ion-content>
```

# Data Binding 적용하기 : One-way Binding

## (2) 객체 배열 변수와 이벤트를 처리하는 메서드 추가하기

src/pages/home/home.ts

```
export class HomePage {
  disabledSwitch = false

  items = [
    {"id":1, name:"첫번째 item"},
    {"id":2, name:"두번째 item"},
    {"id":3, name:"세번째 item"},
    {"id":4, name:"네번째 item"},
    {"id":5, name:"다섯번째 item"},
  ]
  itemSelected(item){
    alert(item.id);
  }
  clickButton(event) {
    alert(event.target.textContent)
    console.log(event.clientX + ", " + event.clientY);
  }
}
```



# Data Binding 적용하기 : Two-way Binding

---

## (1) ngModel Directive 적용

### 1) ngModel Directive

`[(ngModel)] = "username"`

**src/pages/home/home.html**

```
<ion-item>
  <ion-label fixed>사용자 이름</ion-label>
  <ion-input type="text" value="" [(ngModel)]="userName"> </ion-input>
</ion-item>

{{userName}}
```

**src/pages/home/home.ts**

```
//변수 선언
userName : any;
```

# Page Navigation : NavController

---

(1) API에서 제공되는 NavController 란?

: NavController는 Nav 및 Tab과 같은 Navigation Controller Component의 기본 클래스이며, 이 Controller를 사용하여 App의 다른 페이지로 이동합니다.

: Pushing은 탐색 스택 맨 위에 새 페이지가 생기고, 새 페이지가 로드 되어 집니다.  
Popping은 현재 페이지가 스택의 이전 페이지로 이동합니다.

(2) Injecting NavController

: Ionic은 Dependency Injection을 사용하여, 새로운 NavController를 인스턴스화 해서 개발자가 작성한 컴포넌트의 생성자를 통해 주입(Injection) 시켜 준다.

# Page Navigation

---

(1) 새로운 bind Page 컴포넌트 생성

```
$ ionic g page bind
```

(2) NavController 사용하여 home Page에서 bind Page로 이동하기

src/pages/home/home.ts

```
constructor(public navCtrl: NavController, public navParams: NavParams) {  
}  
clickButton(event) {  
  this.navCtrl.push('bindPage');  
}
```

src/pages/bind/bind.ts

```
@IonicPage({ name:'bindPage', })  
@Component({  
  selector: 'page-bind',  
  templateUrl: 'bind.html',  
})  
export class BindPage {  
  constructor(public navCtrl: NavController, public navParams: NavParams) { }  
  goBack() { this.navCtrl.pop(); }  
}
```

# Page Navigation

---

(3) NavController 사용하여 bind Page에서 home Page로 돌아가기

src/pages/bind/bind.html

```
<ion-header>
  <ion-navbar>
    <ion-title>bind</ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  <button ion-button full (click)="goBack()">Back</button>
</ion-content>
```

# Page Navigation : Life Cycle Event

---

## (1) Navigation의 Life Cycle Event 리스트

Page Event	Returns	Description
ionViewDidLoad	void	Runs when the page has loaded. This event only happens once per page being created. If a page leaves but is cached, then this event will not fire again on a subsequent viewing. The ionViewDidLoad event is good place to put your setup code for the page.
ionViewWillEnter	void	Runs when the page is about to enter and become the active page.
ionViewDidEnter	void	Runs when the page has fully entered and is now the active page. This event will fire, whether it was the first load or a cached page.
ionViewWillLeave	void	Runs when the page is about to leave and no longer be the active page.
ionViewDidLeave	void	Runs when the page has finished leaving and is no longer the active page.
ionViewWillUnload	void	Runs when the page is about to be destroyed and have its elements removed.
ionViewCanEnter	boolean/ Promise<void>	Runs before the view can enter. This can be used as a sort of "guard" in authenticated views where you need to check permissions before the view can enter
ionViewCanLeave	boolean/ Promise<void>	Runs before the view can leave. This can be used as a sort of "guard" in authenticated views where you need to check permissions before the view can leave

# UI Component : ActionSheet 사용

---

(1) 새로운 component Page 컴포넌트 생성

\$ ionic g page component

(2) NavController 사용하여 home Page에서 component Page로 이동하기

src/pages/home/home.ts

```
itemSelected(item) {  
  if(item.id === 1) {  
    this.navCtrl.push('componentPage');  
  }  
}
```

src/pages/component/component.html

```
<ion-content padding>  
  <button ion-button full (click)="actionSheet()">Action Sheet</button>  
</ion-content>
```

# UI Component : ActionSheet 사용

## (3) ActionSheet를 생성하는 메서드 추가하기

src/pages/component/component.ts

```
actionSheet() {  
  let actionSheet = this.actionSheetCtrl.create({  
    title: 'Choose Menu',  cssClass: 'action-sheets-basic-page',  
    buttons: [ {  
      text: 'Share',  
      icon: !this.platform.is('ios') ? 'share' : null,  
      handler: () => { console.log('Share clicked'); }  
    },  
    {  
      text: 'Search',  
      icon: !this.platform.is('ios') ? 'search' : null,  
      handler: () => { console.log('Search clicked'); }  
    },  
    {  
      text: 'Cancel',  
      role: 'cancel', // will always sort to be on the bottom  
      icon: !this.platform.is('ios') ? 'close' : null,  
      handler: () => { console.log('Cancel clicked'); }  
    }  
  ] });  
  actionSheet.present();  
}
```

# UI Component : ActionSheet 사용

---

## (4) ActionSheet의 Style 설정하기

src/pages/component/component.scss

```
.action-sheets-basic-page {  
  .ion-md-share {  
    color: #ED4248;  
  }  
  .ion-md-search{  
    color: #31D55F;  
  }  
  .action-sheet-cancel ion-icon,  
  .action-sheet-destructive ion-icon {  
    color: #757575;  
  }  
}
```



# UI Component : ModalController

---

(1) API에서 제공되는 ModalController 란?

: Modal은 사용자의 현재 페이지를 이동하는 Content Pane이며, item을 선택하거나 편집할 때 사용됩니다.

Modal이 표시된 후 ViewController의 dismiss() 메서드를 사용하여 닫을 수 있다.

(2) Injecting ModalController

: Ionic은 Dependency Injection을 사용하여, 새로운 ModalController를 인스턴스화 해서 개발자가 작성한 컴포넌트의 생성자를 통해 주입(Injection) 시켜 준다.

# UI Component : Modal 사용

(1) 새로운 modal Page 컴포넌트 생성

```
$ ionic g page modal
```

(2) ModalController 사용하여 component Page에서 modal Page로 생성하기

src/pages/component/component.html

```
<ion-content padding>
  <button ion-button full (click)="actionSheet()">Action Sheet</button>
  <div class="divider" style="width:100%;height:20px"> </div>
  <button ion-button full color="secondary" (click)="modal()">Modal</button>
</ion-content>
```

src/pages/component/component.ts

```
modal() {
  let modal = this.modalCtrl.create('modalPage');
  modal.present();
}
```

src/pages/modal/modal.ts

```
@IonicPage({
  name:'modalPage',
})
```

# UI Component : Modal 사용

(3) 생성된 modal Page에서 ViewController 의 dismiss() 메서드 사용하기

src/pages/modal/modal.html

```
<ion-navbar color="primary">
  <ion-title>modal</ion-title>
  <ion-buttons end>
    <button ion-button icon-only (click)="close()">
      <ion-icon name="close"> </ion-icon>
    </button>
  </ion-buttons>
</ion-navbar>
```

src/pages/modal/modal.ts

```
constructor(public navCtrl: NavController,
             public viewCtrl : ViewController,
             public navParams: NavParams) {
}
close() {
  this.viewCtrl.dismiss();
}
```

# UI Component : ViewController

---

(1) API에서 제공되는 ViewController 란?

: 현재 View에 대한 다양한 기능 및 정보를 제공한다.

(2) Injecting ViewController


: Ionic은 Dependency Injection을 사용하여, 새로운 ViewController를 인스턴스 화 해서 개발자가 작성한 컴포넌트의 생성자를 통해 주입(Injection) 시켜 준다.

# UI Component : Modal Page내에 Input Forms 사용

(1) Modal 컴포넌트 내에 입력 정보를 저장, ViewController의 dismiss() 메서드

src/pages/modal/modal.ts

```
private profile = {  
  actionSwitch : false, name:"", gender:"", domestic:"", startDate:"  
}  
save() {  
  this.viewCtrl.dismiss(this.profile);  
}
```

 dismiss(data, role , navOptions)

Dismiss the current viewController

Param	Type	Details
data	any	Data that you want to return when the viewController is dismissed. <b>OPTIONAL</b>
role	any	<b>OPTIONAL</b>
navOptions	NavOptions	Options for the dismiss navigation.

↳ Returns: any

data Returns the data passed in, if any.

# UI Component : Modal Page내에 Input Forms 사용

## (2)-1 Input Forms 컴포넌트들의 사용

src/pages/modal/modal.html

```
<ion-list>
  <ion-item>
    <ion-label>Action Switch</ion-label>
    <ion-toggle [(ngModel)]="profile.actionSwitch"> </ion-toggle>
  </ion-item>

  <ion-item>
    <ion-label fixed>이름</ion-label>
    <ion-input type="text" placeholder="이름을 입력하세요" [(ngModel)]="profile.name"> </ion-input>
  </ion-item>

  <ion-item>
    <ion-label>Gender</ion-label>
    <ion-select [(ngModel)]="profile.gender">
      <ion-option value="여자"> 여자</ion-option>
      <ion-option value="남자"> 남자</ion-option>
    </ion-select>
  </ion-item>
```

# UI Component : Modal Page내에 Input Forms 사용

## (2)-2 Input Forms 컴포넌트들 사용

src/pages/modal/modal.html

```
<ion-item>
  <ion-label>국내거주</ion-label>
  <ion-checkbox [(ngModel)]="profile.domestic"> </ion-checkbox>
</ion-item>

<ion-item>
  <ion-label>Start Date</ion-label>
  <ion-datetime displayFormat="YYYY-MM-DD" [(ngModel)]="profile.startDate"> </ion-datetime>
</ion-item>
<button ion-button full (click)="save()">Save</button>
</ion-list>
```

## (3) Modal Page에서 입력한 Data를 Component Page에서 전달 받기

src/pages/component/component.ts

```
modal() {
  let modal = this.modalCtrl.create('modalPage');
  modal.onDidDismiss(data => {
    console.log(data);
  });
  modal.present();
}
```

# UI Component : Slides 사용

(1) 새로운 slide Page 컴포넌트 생성

```
$ ionic g page slide
```

(2) NavController 사용하여 component Page에서 slide Page로 생성하기

src/pages/component/component.html

```
<ion-content padding>  
  <div class="divider" style="width:100%;height:20px"> </div>  
  <button ion-button full color="secondary" (click)="slide()">Slide</button>  
</ion-content>
```

src/pages/component/component.ts

```
slide() {  
  this.navCtrl.push('slidePage');  
}
```

src/pages/slide/slide.ts

```
@IonicPage({  
  name:'slidePage',  
})
```



## UI Component : Slides 사용

(3) 생성된 slide Page에서 slides 컴포넌트 사용하기

: src/assets/imgs/ 디렉토리 아래에 image 파일들을 준비해 놓는다.

src/pages/slide/slide.ts

```
private images = [  
  {title : 'assets/imgs/1.jpg'},  
  {title : 'assets/imgs/2.jpg'},  
  {title : 'assets/imgs/3.jpg'},  
  {title : 'assets/imgs/4.jpg'},  
  {title : 'assets/imgs/5.jpg'},  
]
```

src/pages/slide/slide.html

```
<ion-slides pager autoplay="5000" loop="true" speed="2000">  
  
  <ion-slide *ngFor="let image of images">  
    <img [src]="image.title" />  
  </ion-slide>  
  
</ion-slides>
```

# UI Component : AlertController

---

(1) API에서 제공되는 AlertController 란?

: Alert는 사용자에게 입력을 사용하여 정보를 제공하거나 사용자로부터 정보를 수집하는 대화 상자입니다.

```
constructor(private alertCtrl: AlertController) { }

presentAlert() {
  let alert = this.alertCtrl.create({
    title: 'Low battery',
    subTitle: '10% of battery remaining',
    buttons: ['Dismiss']
  });
  alert.present();
}
```

# UI Component : Prompt Alerts 사용

## (1)-1 alertController를 통해 Prompt Alerts 사용하기

src/pages/component/component.html

```
<ion-content padding>  
  <button ion-button full color="danger" (click)="promptAlert()">Alert</button>  
</ion-content>
```

src/pages/component/component.ts

```
private accountData = {  
  name : "",  
  email : ""  
}  
  
constructor(public alertCtrl: AlertController) {  
}  
  
promptAlert() {  
  let prompt = this.alertCtrl.create();  
  prompt.present();  
}
```

# UI Component : Prompt Alerts 사용

## (1)-2 alertController를 통해 Prompt Alerts 사용하기

src/pages/component/component.ts

```
promptAlert() {  
  let prompt = this.alertCtrl.create({  
    title: 'Login',  
    message: "이름과 E-Mail를 입력하세요",  
    inputs: [  
      { name: 'name', placeholder: 'Name 입력' },  
      { name: 'email', placeholder: 'Email 입력' },  
    ],  
    buttons: [  
      { text: '취소', handler: data => { console.log('Cancel clicked'); } },  
      {  
        text: '저장',  
        handler: data => {  
          this.accountData = { name : data.name, email : data.email }  
          this.navCtrl.push('navPage',{account:this.accountData});  
        }  
      }  
    ]  
  });  
  prompt.present();  
  console.log(this.accountData);  
}
```

# UI Component : Prompt Alerts 사용

(2) 새로운 nav Page 컴포넌트 생성

```
$ ionic g page nav
```

(3) AlertController가 전달해 준 data를 NavParams에서 꺼내서 출력하기

src/pages/nav/nav.ts

```
private accountData = { name : "", email : "" }  
constructor(public navCtrl: NavController, public NavParams: NavParams) { }  
ionViewDidLoad() {  
  this.accountData = this.NavParams.get('account');  
}
```

src/pages/nav/nav.html

```
<ion-item>  
  <ion-label fixed>이름</ion-label>  
  <ion-input type="text" [value]="accountData.name"> </ion-input>  
</ion-item>  
  
<ion-item>  
  <ion-label fixed>E메일</ion-label>  
  <ion-input type="text" readonly [value]="accountData.email"> </ion-input>  
</ion-item>
```

# UI Component : ToastController

---

(1) API에서 제공되는 ToastController 란?

: ToastController는 작업에 대한 피드백을 제공하거나 시스템 메시지를 표시하는 데 사용할 수 있습니다.

```
constructor(private toastCtrl: ToastController) { }  
let toast = this.toastCtrl.create({  
  message: 'User was added successfully',  
  duration: 3000,  
  position: 'top'  
});  
toast.onDidDismiss(() => { console.log('Dismissed toast'); });  
toast.present();
```

# UI Component : Toast 사용

## (1) ToastController를 통해 toast 생성하기

src/pages/component/component.html

```
<ion-content padding>  
  <button ion-button full color="dark" (click)="toast()">Toast</button>  
</ion-content>
```

src/pages/component/component.ts

```
constructor(public toastCtrl: ToastController) {  
  }  
  
  toast() {  
    let toast = this.toastCtrl.create({  
      message: '3초 동안 보였다가 사라집니다.',  
      duration: 3000,  
      position : 'top',  
    });  
    toast.present();  
  }  
}
```

# UI Component : LoadingController 사용

## (1) LoadingController를 통해 Loader 생성하기

src/pages/component/component.html

```
<ion-content padding>  
  <button ion-button full color="secondary" (click)="loading()">Loading</button>  
</ion-content>
```

src/pages/component/component.ts

```
constructor(public loadingCtrl: LoadingController) {  
  }  
  
  loading() {  
    let loading = this.loadingCtrl.create({  
      content: '잠시만 기다려주세요...'  
    });  
  
    loading.present();  
  
    setTimeout(() => {  
      loading.dismiss();  
    }, 3000);  
  }  
}
```



# UI Component : Loading을 처리하는 Provider 생성하기

## (1) Provider 컴포넌트 생성하기

\$ ionic g provider loading

src/providers/loading/loading.ts

```
@Injectable()
export class LoadingProvider {

  constructor(public http: HttpClient) {
    console.log('Hello LoadingProvider Provider');
  }
}
```

src/app/app.module.ts

```
@NgModule({
  providers: [
    StatusBar,
    SplashScreen,
    {provide: ErrorHandler, useClass: IonicErrorHandler},
    LoadingProvider
  ]
})
export class AppModule {}
```

# UI Component : Loading을 처리하는 Provider 생성하기

## (2) LoadingProvider 구현하기

src/providers/loading/loading.ts

```
@Injectable()
export class LoadingProvider {
  private loading : any;

  constructor(public loadingCtrl:LoadingController) {
  }

  show() {
    this.loading = this.loadingCtrl.create({
      content: '잠시만 기다려주세요...'
    });
    this.loading.present();
  }

  hide() {
    this.loading.dismiss();
  }
}
```

# UI Component : Loading을 처리하는 Provider 생성하기

---

## (3) LoadingProvider 사용하기

src/pages/component/component.ts

```
import {LoadingProvider} from "../../providers/loading/loading";

constructor(public loadingProvider : LoadingProvider) { }

loading() {
  this.loadingProvider.show()

  setTimeout(() => {
    this.loadingProvider.hide();
  }, 3000);
}
```

# UI Component : Interface 선언하여 사용하기

---

## (1) Interface 선언하기

: src/interfaces/account.ts 파일 생성하기

src/interfaces/account.ts

```
export interface AccountInterface {  
  name : string,  
  email : string,  
}
```

## (2) Interface 사용하기

src/pages/component/component.ts

```
import {AccountInterface} from "../../interfaces/account";  
  
private accountData = {} as AccountInterface;
```

src/pages/nav/nav.ts

```
import {AccountInterface} from "../../interfaces/account";  
  
private accountData = {} as AccountInterface;
```

# SQLite Device DB : SQLite 설치 및 설정하기

(1) SQLite (<https://ionicframework.com/docs/native/sqlite/>) 설치 ( 단말기에서만 테스트 가능)

```
$ ionic cordova plugin add cordova-sqlite-storage
```

```
$ npm install --save @ionic-native/sqlite
```

(2) SQLite를 Root Module에 등록하기

src/app/app.module.ts

```
import { SQLite } from "@ionic-native/sqlite";

@NgModule({
  providers: [
    SQLite,
  ]
})
export class AppModule {}
```

# SQLite Device DB : 데이터 등록

(1) 새로운 database Page 컴포넌트 생성

```
$ ionic g page database
```

(2) NavController 사용하여 home Page에서 database Page로 이동하기

src/pages/home/home.ts

```
items = [ {"id":1, name:"UI Component"},  
          {"id":2, name:"SQLite"}, ];  
itemSelected(item){  
  if (item.id === 1) {  
    this.navCtrl.push('componentPage');  
  }else if ( item.id === 2) {  
    this.navCtrl.push('databasePage');  
  } }
```

src/pages/database/database.ts

```
@IonicPage({ name:'databasePage', })  
  
constructor(private sqlite : SQLite) { }  
  
ionViewDidLoad() {  
  this.getData();  
}
```

# SQLite Device DB : 데이터 등록

## (3) Table 생성 및 Data 등록하는 화면 호출

src/pages/database/database.ts

```
getData() {  
  this.sqlite.create({  
    name : 'mydb.db', location : "default"  
  }).then((db:SQLiteObject) => {  
    let sql = "create table if not exists account(id INTEGER PRIMARY KEY,name TEXT,email TEXT,phone TEXT,gender TEXT)";  
    db.executeSql(sql,{}).then(result => { console.log(result);}).catch((error) => {console.log(error);});  
  }).catch((error) => { console.log(error);});  
}  
addData() {  
  this.navCtrl.push('addDataPage',{ mode : "add", profile : " " });  
}
```

src/pages/database/database.html

```
<ion-navbar>  
  <ion-title color="primary">database</ion-title>  
  <ion-buttons end>  
    <button ion-button (click)="addData()" icon-only>  
      <ion-icon name="add"></ion-icon>  
    </button>  
  </ion-buttons>  
</ion-navbar>
```

# SQLite Device DB : 데이터 등록

## (4) 새로운 add-data Page 컴포넌트 생성 및 Interface 작성

```
$ ionic g page addData
```

src/pages/add-data/add-data.ts

```
@IonicPage({ name:'addDataPage', })  
export class AddDataPage {  
  private userData = {} as UserProfileInterface;  
}
```

src/interfaces/user-profile.ts

```
export interface UserProfileInterface {  
  name : string,  
  email : string,  
  phone : string,  
  gender : string,  
  id : string  
}
```



# SQLite Device DB : 데이터 등록

## (5) Data를 등록 하는 화면 작성

src/pages/add-data/add-data.html

```
<ion-list>
  <ion-item>
    <ion-label fixed>이름 </ion-label>
    <ion-input type="text" [(ngModel)]="userData.name"> </ion-input>
  </ion-item>
  <ion-item>
    <ion-label fixed>E메일 </ion-label>
    <ion-input type="email" [(ngModel)]="userData.email"> </ion-input>
  </ion-item>
  <ion-item>
    <ion-label fixed>전화번호 </ion-label>
    <ion-input type="phone" [(ngModel)]="userData.phone"> </ion-input>
  </ion-item>
  <ion-item>
    <ion-label>성별 </ion-label>
    <ion-select [(ngModel)]="userData.gender">
      <ion-option value="남자">남자 </ion-option>
      <ion-option value="여자">여자 </ion-option>
    </ion-select>
  </ion-item>
</ion-list>
<div padding>
  <button ion-button color="primary" block (click)="save()">등록 </button>
</div>
```

# SQLite Device DB : 데이터 등록

## (6) Data를 등록하는 SQL문 작성하기

src/pages/add-data/add-data.ts

```
private userData = {} as UserProfileInterface;
private mode : any;

constructor(public navCtrl: NavController, public sqlite:SQLite, public navParams: NavParams) {
  this.mode = this.navParams.get('mode');
}
save() {
  this.sqlite.create({
    name : 'mydb.db',
    location : 'default'
  }).then((db : SQLiteObject) => {
    if (this.mode === "add") {
      let sql = "insert into account values(NULL,?,?,?,?,?)";
      db.executeSql(sql,[
        this.userData.name,
        this.userData.email,
        this.userData.phone,
        this.userData.gender
      ]).then((result) => {
        console.log(result);
        this.navCtrl.pop();
      }).catch((error) => {console.log(error);})
    }
  }).catch((error) => {console.log(error);})
}
```

# SQLite Device DB : 데이터 등록

(7) Data를 등록 한 후에 Data를 보여주는 화면과 SQL 작성하기

src/pages/database/database.html

```
<ion-list>
  <ion-item *ngFor="let user of users">
    <div>{{user.name}}</div>
    <div>{{user.email}}</div>
    <div>{{user.phone}}</div>
    <div>{{user.gender}}</div>
  </ion-item>
</ion-list>
```

src/pages/database/database.ts

```
getData() {
  let sql2 = "select * from account order by id DESC";
  db.executeSql(sql2, {}).then((result) => {
    this.users = [];
    for(let i=0; i < result.rows.length; i++){
      this.users.push({
        id: result.rows.item(i).id,
        name: result.rows.item(i).name,
        email: result.rows.item(i).email,
        phone: result.rows.item(i).phone,
        gender: result.rows.item(i).gender,
      });
    }
  })
}
```

# SQLite Device DB : 데이터 수정

## (1) Data를 수정 하는 화면 작성 ( item-sliding 사용)

src/pages/database/database.html

```
<ion-list>
  <ion-item-sliding #item *ngFor="let user of users">
    <ion-item>
      <div>{{user.name}}</div>
      <div>{{user.email}}</div>
      <div>{{user.phone}}</div>
      <div>{{user.gender}}</div>
    </ion-item>

    <ion-item-options side="right">
      <button ion-button (click)="editData(item,user)">수정</button>
      <button ion-button color="danger" (click)="deleteData(item,user)">삭제</button>
    </ion-item-options>

  </ion-item-sliding>
</ion-list>
```

# SQLite Device DB : 데이터 수정

## (2) Data를 수정 하는 SQL문 작성하기

src/pages/database/database.ts

```
editData(item:ItemSliding,user) {  
  item.close();  
  this.navCtrl.push('addDataPage',{  
    mode : "edit",  
    user : user  
  }); }  
}
```

src/pages/add-data/add-data.ts

```
constructor(public navCtrl: NavController, public sqlite:SQLite, public navParams: NavParams) {  
  this.mode = this.navParams.get('mode');  
  if (this.mode === "edit") {  
    this.userData = this.navParams.get('user');  
  }  
}  
  
if(this.mode === "edit") {  
  let sql2 = "update account set name=?,email=?,phone=?,gender=? where id=?";  
  db.executeSql(sql2,[  
    this.userData.name, this.userData.email, this.userData.phone, this.userData.gender,this.userData.id  
  ]).then((result) => { this.navCtrl.pop();})  
    .catch((error) => { console.log(error);})  
}  
}
```

# SQLite Device DB : 데이터 삭제

(1) Data를 삭제 하는 화면 작성 ( item-sliding 사용)

src/pages/database/database.html

```
<ion-list>
  <ion-item-sliding #item *ngFor="let user of users">
    <ion-item>
      <div>{{user.name}}</div>
      <div>{{user.email}}</div>
      <div>{{user.phone}}</div>
      <div>{{user.gender}}</div>
    </ion-item>

    <ion-item-options side="right">
      <button ion-button color="danger" (click)="deleteData(item,user)">삭제 </button>
    </ion-item-options>

  </ion-item-sliding>
</ion-list>
```

# SQLite Device DB : 데이터 삭제

## (2) Data를 삭제 하는 SQL문 작성하기

src/pages/database/database.ts

```
deleteData(item:ItemSliding,user) {  
  item.close();  
  let confirm = this.alertCtrl.create({  
    title: '삭제',  
    message: user.name + '을 삭제하시겠습니까?',  
    buttons: [  
      { text: '아니요',  
        handler: () => { console.log('아니요 clicked'); } },  
      {  
        text: '예',  
        handler: () => {  
          console.log('예 clicked');  
          this.sqlite.create({  
            name:'mydb.db', location : 'default'  
          }).then((db:SQLiteObject) => {  
            db.executeSql("delete from account where id=?", [user.id])  
              .then((result) => { this.getData(); })  
              .catch((error) => { console.log(error); });  
          }).catch((error) => { console.log(error); });  
        }  
      }  
    ]  
  });  
  confirm.present();  
}
```

# Email Composer : Email Composer 설치 및 설정하기

---

(1) EmailComposer (<https://ionicframework.com/docs/native/email-composer/>) 설치

```
$ ionic cordova plugin add cordova-plugin-email-composer
```

```
$ npm install --save @ionic-native/email-composer
```

(2) EmailComposer를 Root Module에 등록하기

src/app/app.module.ts

```
import { EmailComposer } from "@ionic-native/email-composer";

@NgModule({
  providers: [
    EmailComposer,
  ]
})
export class AppModule {}
```



# Email Composer : Email 보내기

---

## (3) Email를 보내는 화면 작성 ( item-sliding 사용)

src/pages/database/database.html

```
<ion-list>
  <ion-item-sliding #item *ngFor="let user of users">
    <ion-item>
      <div>{{user.name}}</div>
      <div>{{user.email}}</div>
      <div>{{user.phone}}</div>
      <div>{{user.gender}}</div>
    </ion-item>

    <ion-item-options side="left">
      <button ion-button (click)="sendEmail(item,user)">Email</button>
    </ion-item-options>

  </ion-item-sliding>
</ion-list>
```

# Email Composer : Email 보내기

---

## (4) Email를 보내는 처리하기

src/pages/database/database.ts

```
constructor(private emailComposer : EmailComposer) {  
}  
sendEmail(item:ItemSliding,user) {  
  item.close();  
  let email = {  
    to: user.email,  
    subject: 'To :' + user.name,  
    body: "",  
    isHtml: true  
  };  
  
  // Send a text message using default options  
  this.emailComposer.open(email);  
}
```

# SMS : SMS 설치 및 설정하기

---

(1) SMS (<https://ionicframework.com/docs/native/sms/>) 설치

```
$ ionic cordova plugin add cordova-sms-plugin
```

```
$ npm install --save @ionic-native/sms
```

(2) SMS를 Root Module에 등록하기

src/app/app.module.ts

```
import { SMS } from "@ionic-native/sms";

@NgModule({
  providers: [
    SMS,
  ]
})
export class AppModule {}
```

# SMS : SMS 보내기

## (3) SMS를 보내는 화면 작성 ( item-sliding 사용)

src/pages/database/database.html

```
<ion-list>
  <ion-item-sliding #item *ngFor="let user of users">
    <ion-item>
      <div>{{user.name}}</div>
      <div>{{user.email}}</div>
      <div>{{user.phone}}</div>
      <div>{{user.gender}}</div>
    </ion-item>

    <ion-item-options side="left">
      <button ion-button color="danger" (click)="sendSMS(item,user)">SMS</button>
    </ion-item-options>

  </ion-item-sliding>
</ion-list>
```

# SMS : SMS 보내기

## (4) SMS를 보내는 처리하기

src/pages/database/database.ts

```
constructor(private sms : SMS) { }  
sendSMS(item:ItemSliding,user) {  
  item.close();  
  let prompt = this.alertCtrl.create({  
    title: 'SMS',  
    message: "메세지를 작성하여 주시기 바랍니다.",  
    inputs: [  
      {  
        name: 'message',  
        placeholder: 'Message Here....'  
      },  
    ],  
    buttons: [  
      {  
        text: '취소',  
        handler: data => { console.log('취소 clicked'); }  
      },  
      {  
        text: '보내기',  
        handler: data => { this.sms.send(user.phone, data.message);}  
      }  
    ]  
  });  
  prompt.present();  
}
```

## Call Number : Call Number 설치 및 설정하기

---

(1) Call number (<https://ionicframework.com/docs/native/call-number/>) 설치

```
$ ionic cordova plugin add call-number
```

```
$ npm install --save @ionic-native/call-number
```

(2) CallNumber를 Root Module에 등록하기

src/app/app.module.ts

```
import { CallNumber } from "@ionic-native/call-number";

@NgModule({
  providers: [
    CallNumber,
  ]
})
export class AppModule {}
```

# Call Number : Call Number 보내기

## (3) CallNumber를 보내는 화면 작성 ( item-sliding 사용)

src/pages/database/database.html

```
<ion-list>
  <ion-item-sliding #item *ngFor="let user of users">
    <ion-item>
      <div>{{user.name}}</div>
      <div>{{user.email}}</div>
      <div>{{user.phone}}</div>
      <div>{{user.gender}}</div>
    </ion-item>

    <ion-item-options side="left">
      <button ion-button color="secondary" (click)="sendCall(item,user)">Call</button>
    </ion-item-options>

  </ion-item-sliding>
</ion-list>
```

# Call Number : Call Number 보내기

---

## (4) Call Number를 보내는 처리하기

src/pages/database/database.ts

```
constructor(private callNumber : CallNumber) { }  
sendCall(item:ItemSliding,user) {  
  item.close();  
  alert(user.phone);  
  this.callNumber.callNumber(user.phone, true)  
    .then(res => console.log('Launched dialer!', res))  
    .catch(err => console.log('Error launching dialer', err));  
}
```



# Ionic3 REST CRUD App 구현

# Ionic3 REST CRUD App : 개요와 json-server 설치

---

## (1) Ionic3 REST CRUD App

- : a) 간단한 CRUD Ionic3 어플리케이션 입니다.
- b) Angular의 HttpClient를 사용하여 REST Server와 Http 통신을 합니다.
- c) 백엔드 개발자가 서버 프로그램을 다 완성할 때 까지 기다릴 필요없이 Fake REST API Server( json-server )를 사용하여 프론트 엔드 응용 프로그램을 테스트 할 수 있도록 해줍니다.

## (2) REST CRUD App Project 생성

`$ ionic start ionic-rest blank`

- \* Would you like to integrate your new app with Cordova to target native iOS and Android? No
- \* Install the free Ionic Pro SDK and connect your app? No

## Ionic3 REST CRUD App : 개요와 json-server 설치

---

(3) Run Fake REST API Server 설치 및 json 서버 실행하는 방법

```
$ npm install -g json-server
```

```
$ json-server --watch db.json
```

watch option은 (-- 2개임, db.json 파일이 있는 곳에서 실행)

<http://localhost:3000/products> 로 rest 요청을 보낼 수 있다.

ionic-lab/db.json

```
{
  "products": [
    {
      "id": 1, "name": "computer", "cost": 200, "quantity": 20
    },
  ]
}
```

## Ionic3 REST CRUD App : 프로젝트 생성과 **module** 등록

---

(4) Product(상품) 목록을 화면에 출력할 때 상품명 순으로 정렬(ordering) 기능을 사용하기 위해서 ngx-order-pipe를 설치한다.

(node\_modules 폴더가 있는 곳에서 install 해야함)

```
$ npm install ngx-order-pipe --save
```

(5) HttpClientModule, OrderModule, FormsModule을 Root Module(app.module.ts)에 등록한다.

a. HttpClientModule은 HttpClient를 사용하기 위해 필요한 모듈이다.

b. OrderModule은 정렬(ordering) 기능을 사용하기 위해 필요한 모듈이다.

c. FormsModule은 ngModel Directive를 사용하기 위해 필요한 모듈이다.

# Ionic3 REST CRUD App : 필요한 module 추가

(5-1) app.module.ts의 imports 속성에 필요한 modules 추가

src/app/app.module.ts

```
import { HttpClientModule } from '@angular/common/http'
import { FormsModule } from '@angular/forms';
import { OrderModule } from "ngx-order-pipe";
@NgModule({
  declarations: [
    MyApp, HomePage, ],
  imports: [
    BrowserModule,
    HttpClientModule, OrderModule, FormsModule,
    IonicModule.forRoot(MyApp)
  ],
  bootstrap: [IonicApp],
  entryComponents: [
    MyApp, HomePage, ],
  providers: [
    StatusBar,
    SplashScreen,
    {provide: ErrorHandler, useClass: IonicErrorHandler}, ]
})
export class AppModule {}
```

# Ionic3 REST CRUD App : Model 클래스 작성

---

(1) Model Class : product.ts 클래스

src/model/product.ts

```
export class Product {  
  id:number;  
  name:string;  
  cost:number;  
  quantity:number;  
  
  constructor(values: Object= {}){  
    Object.assign(this, values);  
  }  
}
```

# Ionic3 REST CRUD App : Product List 기능 구현

---

(1) Rest Provider 컴포넌트 생성하기

```
$ ionic g provider rest
```

RestProvider를 생성하면서, app.module.ts의 providers 속성에 자동으로 추가 된다.

src/app/app.module.ts

```
@NgModule({
  declarations: [
    MyApp, HomePage, ],
  ....
  providers: [
    StatusBar,
    SplashScreen,
    {provide: ErrorHandler, useClass: IonicErrorHandler},
    RestProvider
  ]
})
export class AppModule {}
```

# Ionic3 REST CRUD App : Product List 기능 구현

## (2) RestProvider 클래스 : getProducts() 메서드 구현

src/providers/rest/rest.ts

```
@Injectable()
export class RestProvider {

  baseUrl:string = "http://localhost:3000";

  constructor(public http: HttpClient) {
    console.log('RestProvider constructor');
  }

  public getProducts():Observable<Product[]>{
    return this.http.get(this.baseUrl + "/products")
      .map((products:Product[]) => {
        console.log(products);
        return products.map(product=>{ return new Product(product)});
      }).catch((err)=>{
        console.error(err);
        return Observable.empty<Product[]>();
      });
  }
}
```



# Ionic3 REST CRUD App : Product List 기능 구현

## (3) HomePage 클래스 : ionViewDidLoad() life cycle 메서드

src/pages/home/home.ts

```
import { Observable } from 'rxjs/Observable';
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import { RestProvider } from '../providers/rest/rest';
import { Product } from '../model/product';

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {

  productsObservable:Observable<Product[]>;

  constructor(public navCtrl: NavController, private rest:RestProvider) { }

  ionViewDidLoad(){
    this.productsObservable = this.rest.getProducts();
  }
}
```

# Ionic3 REST CRUD App : Product List 기능 구현

(4) HomePage 화면에 name 별로 정렬된 Product List 출력하기

src/pages/home/home.html

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Json-Server Crud Example
    </ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  <ion-list>
    <ion-list-header>Products sorted by name</ion-list-header>
    <ion-item *ngFor="let product of productsObservable | async | orderBy:'name'">
      <h2>{{product.name}}</h2>
      <p>{{product.cost | currency : 'USD' }}</p>
    </ion-item>
  </ion-list>
</ion-content>
```

## **Ionic3 REST CRUD App : Product** 상세보기 구현

---

(1) 새로운 Product Page 생성

```
$ ionic g page product
```

(2) src/pages/product 디렉토리에 아래와 같은 파일들이 생성된다.

product.html (화면)

product.module.ts (모듈, Lazy loading을 할 수 있도록 해줌)

product.scss (스타일)

product.ts (컴포넌트 클래스)

# Ionic3 REST CRUD App : Product 상세보기 구현

## (3) HomePage 화면에 Product 상세보기 함수 호출

src/pages/home/home.html

```
<ion-content padding>
  <ion-list>
    <ion-list-header>Products sorted by name</ion-list-header>
    <ion-item *ngFor="let product of productsObservable | async | orderBy:'name'" (click)="navToDetail(product)">
      <h2>{{product.name}}</h2>
      <p>{{product.cost | currency : 'USD' }}</p>
    </ion-item>
  </ion-list>
</ion-content>
```

## (4) HomePage 클래스에 navToDetail() 메서드 구현 : Product page 호출

src/pages/home/home.ts

```
navToDetail(product:Product){
  this.navCtrl.push("ProductPage", {"product": product});
}
```

## Ionic3 REST CRUD App : Product 상세보기 구현

(5) ProductPage 클래스 : 생성자 구현 - NavParams에서 Product 객체 가져오기

src/pages/product/product.ts

```
import { Product } from '../model/product';

@IonicPage()
@Component({
  selector: 'page-product',
  templateUrl: 'product.html',
})
export class ProductPage {

  product:Product;

  constructor(public navCtrl: NavController, public NavParams: NavParams) {
    this.product = new Product(this.navCtrl.get('product'));
  }
}
```

# ionic3 REST CRUD App : Product 상세보기 구현

## (6) ProductPage 화면 : ngModel Directive 사용

src/pages/product/product.html

```
<ion-header>
  <ion-navbar>
    <ion-title>{{product?.name || 'Product'}}</ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <ion-item>
    <ion-label>Id: {{product.id}}</ion-label>
  </ion-item>
  <ion-item>
    <ion-label>Name:</ion-label>
    <ion-input type="text" [(ngModel)]="product.name"></ion-input>
  </ion-item>
  <ion-item>
    <ion-label>Cost:</ion-label>
    <ion-input type="number" [(ngModel)]="product.cost"></ion-input>
  </ion-item>
  <ion-item>
    <ion-label>Quantity:</ion-label>
    <ion-input type="number" [(ngModel)]="product.quantity"></ion-input>
  </ion-item>
</ion-content>
```

# Ionic3 REST CRUD App : Product 등록 기능 구현

## (1) HomePage 화면에 등록 아이콘 추가하기

src/pages/home/home.html

```
<ion-content padding>
  <button ion-button full icon-right (click)="createProduct()">
    New Product
    <ion-icon name="create"> </ion-icon>
  </button>

  <ion-list>
    <ion-list-header>Products sorted by name</ion-list-header>
    <ion-item *ngFor="let product of productsObservable | async | orderBy:'name'" (click)="navToDetail(product)>
      <h2>{{product.name}}</h2>
      <p>{{product.cost | currency : 'USD' }}</p>
    </ion-item>
  </ion-list>
</ion-content>
```

# ionic3 REST CRUD App : Product 등록 기능 구현

## (2) HomePage 클래스 : createProduct() 메서드 구현 - ProductPage 호출

src/pages/home/home.ts

```
export class HomePage {  
  productsObservable:Observable<Product[]>;  
  
  constructor(public navCtrl: NavController, private rest:RestService) { }  
  
  ionViewDidLoad(){  
    this.productsObservable = this.rest.getProducts();  
  }  
  
  navToDetail(product:Product){  
    this.navCtrl.push("ProductPage", {"product": product});  
  }  
  
  createProduct(){  
    this.navCtrl.push("ProductPage", {"product": {}});  
  }  
}
```



# Ionic3 REST CRUD App : Product 등록 기능 구현

## (3) RestProvider 클래스 : createProduct() 메서드 구현

src/providers/rest/rest.ts

```
@Injectable()
export class RestProvider {

  baseUrl:string = "http://localhost:3000";

  constructor(public http: HttpClient) {
    console.log('RestProduct constructor');
  }

  public createProduct(product:Product): Observable<Product>{
    return this.http.post(this.baseUrl + "/products", product)
      .map(response => {
        return new Product(response)
      }).catch((err)=>{
        console.error(err);
        return Observable.empty<Product>();
      })
  }
}
```

# Ionic3 REST CRUD App : Product 등록 기능 구현

## (4) ProductPage 화면 : Product Save 버튼 추가

src/pages/product/product.html

```
<ion-content padding>
  <ion-item>
    <ion-label>Id: {{product.id}}</ion-label>
  </ion-item>
  <ion-item>
    <ion-label>Name:</ion-label>
    <ion-input type="text" [(ngModel)]="product.name"> </ion-input>
  </ion-item>
  <ion-item>
    <ion-label>Cost:</ion-label>
    <ion-input type="number" [(ngModel)]="product.cost"> </ion-input>
  </ion-item>
  <ion-item>
    <ion-label>Quantity: </ion-label>
    <ion-input type="number" [(ngModel)]="product.quantity"> </ion-input>
  </ion-item>
  <br />
  <button ion-button full color="secondary" icon-right (click)="saveProduct(product)"> Save
    <ion-icon name="md-checkmark" item-end> </ion-icon>
  </button>
</ion-content>
```

# ionic3 REST CRUD App : Product 등록 기능 구현

## (5) ProductPage 클래스 : saveProduct() 메서드 구현

src/pages/product/product.ts

```
export class ProductPage {
  product:Product;

  constructor(public navCtrl: NavController, public navParams: NavParams,
    private rest:RestProvider) {
    this.product = new Product(this.navParams.get('product'));
  }

  saveProduct(product:Product){
    if (product.id){ //수정

    }else{          //등록
      this.rest.createProduct(product).subscribe((product)=>{
        this.product = product;
        this.navCtrl.setRoot(HomePage);
      })
    }
  }
}
//saveProduct
```

# Ionic3 REST CRUD App : Product 등록 기능 구현

## (6) ProductPage 클래스 : showSuccessMessage 메서드 구현

src/pages/product/product.ts

```
export class ProductPage {
  constructor(public navCtrl: NavController, public navParams: NavParams,
    private rest: RestProvider, private toastCtrl: ToastController) {
    this.product = new Product(this.navParams.get('product'));
  }
  saveProduct(product: Product){
    ....
    this.rest.createProduct(product).subscribe((product)=>{
      this.product = product;
      this.showSuccessMessage("Product " + product.id + " - " + product.name + " created")
      this.navCtrl.setRoot(HomePage);
    })
  } //saveProduct

  showSuccessMessage(message: string){
    this.toastCtrl.create({
      message: message, showCloseButton: true,
      duration: 3000, position: 'middle'
    }).present();
  } //showSuccessMessage
}
```

# Ionic3 REST CRUD App : Product 수정 기능 구현

## (1) RestProvider 클래스 : updateProduct() 메서드 구현

src/providers/rest/rest.ts

```
@Injectable()
export class RestProvider {

  baseUrl:string = "http://localhost:3000";

  constructor(public http: HttpClient) {
    console.log('RestProvider constructor');
  }

  public updateProduct(product:Product): Observable<Product>{
    return this.http.put(this.baseUrl + "/products/"+product.id, product)
      .map(resp=>{
        return new Product(resp);
      }).catch((err)=>{
        console.error(err);
        return Observable.empty<Product>();
      });
  }
}
```

# **ionic3 REST CRUD App : Product** 수정 기능 구현

## (2) ProductPage 클래스 : saveProduct() 메서드 구현

src/pages/product/product.ts

```
export class ProductPage {
  product:Product;

  constructor(public navCtrl: NavController, public navParams: NavParams,
    private rest:RestProvider) {
    this.product = new Product(this.navParams.get('product'));
  }
  saveProduct(product:Product){
    if (product.id){ //수정
      this.rest.updateProduct(product).subscribe((product)=>{
        this.product = product;
        this.showSuccessMessage("Product " + product.name + " updated");
        this.navCtrl.setRoot(HomePage);
      })
    }else{ //등록
      this.rest.createProduct(product).subscribe((product)=>{
        this.product = product;
        this.navCtrl.setRoot(HomePage);
      })
    }
  }
}
//saveProduct
}
```

# Ionic3 REST CRUD App : Product 삭제 기능 구현

## (1) RestProvider 클래스 : deleteProductById() 메서드 구현

src/providers/rest/rest.ts

```
@Injectable()
export class RestProvider {

  baseUrl:string = "http://localhost:3000";

  constructor(public http: HttpClient) {
    console.log('RestProvider constructor');
  }

  public deleteProductById(productId:number): Observable<Product>{
    return this.http.delete(this.baseUrl + "/products/" + productId)
      .map(resp=>{
        return new Product(resp)
      }).catch((err)=>{
        console.error(err);
        return Observable.empty<Product>();
      })
  }
}
```

# Ionic3 REST CRUD App : Product 삭제 기능 구현

## (2) ProductPage 클래스 작성 : deleteProduct() 메서드 구현

src/pages/product/product.ts

```
export class ProductPage {
  product:Product;

  constructor(public navCtrl: NavController, public navParams: NavParams,
    private rest:RestProvider) {
    this.product = new Product(this.navParams.get('product'));
  }

  deleteProduct(productId:number){
    this.rest.deleteProductById(productId).subscribe((product)=>{
      console.log(product)
      this.showSuccessMessage("Product Id "+ productId + " has been removed!");
      this.navCtrl.setRoot(HomePage);
    })
  }
}
```



# Ionic3 REST CRUD App : Product 삭제 기능 구현

## (3) ProductPage 화면에 삭제 버튼 추가하기

src/pages/product/product.html

```
<ion-content padding>
  .....
  <ion-item>
    <ion-label>Cost:</ion-label>
    <ion-input type="number" [(ngModel)]="product.cost"> </ion-input>
  </ion-item>
  <ion-item>
    <ion-label>Quantity: </ion-label>
    <ion-input type="number" [(ngModel)]="product.quantity"> </ion-input>
  </ion-item>
  <br />
  <button ion-button full color="secondary" icon-right (click)="saveProduct(product)"> Save
    <ion-icon name="md-checkmark" item-end> </ion-icon>
  </button>
  <button *ngIf="product.id" ion-button full color="danger" item-end icon-right
    (click)="deleteProduct(product.id)" > Remove
    <ion-icon name="trash"> </ion-icon>
  </button>
</ion-content>
```

# Ionic 프로젝트를 Andorid에 배포하기

---

- 1. Android studio 설치하기 (<https://developer.android.com/studio/index.html?hl=ko>)
- 2. Andorid SDK(Software Development Kit) 설치하기  
: API Level 26
- 3. `$ npm install ionic-native`
- 4. `$ npm install gradle`
  
- `$ ionic cordova platform add android`
- `$ ionic cordova build android`
- `$ ionic cordova run android`

# TypeScript



<http://www.typescriptlang.org>

# ECMA 스크립트에 대하여



- ECMA 스크립트는 ECMA International의 표준

최초 ECMA 스크립트는 브라우저 언어인 Javascript와 Jscript간 차이를 줄이기 위한 공통 스펙 제안으로 출발 (1997, ECMA-262)

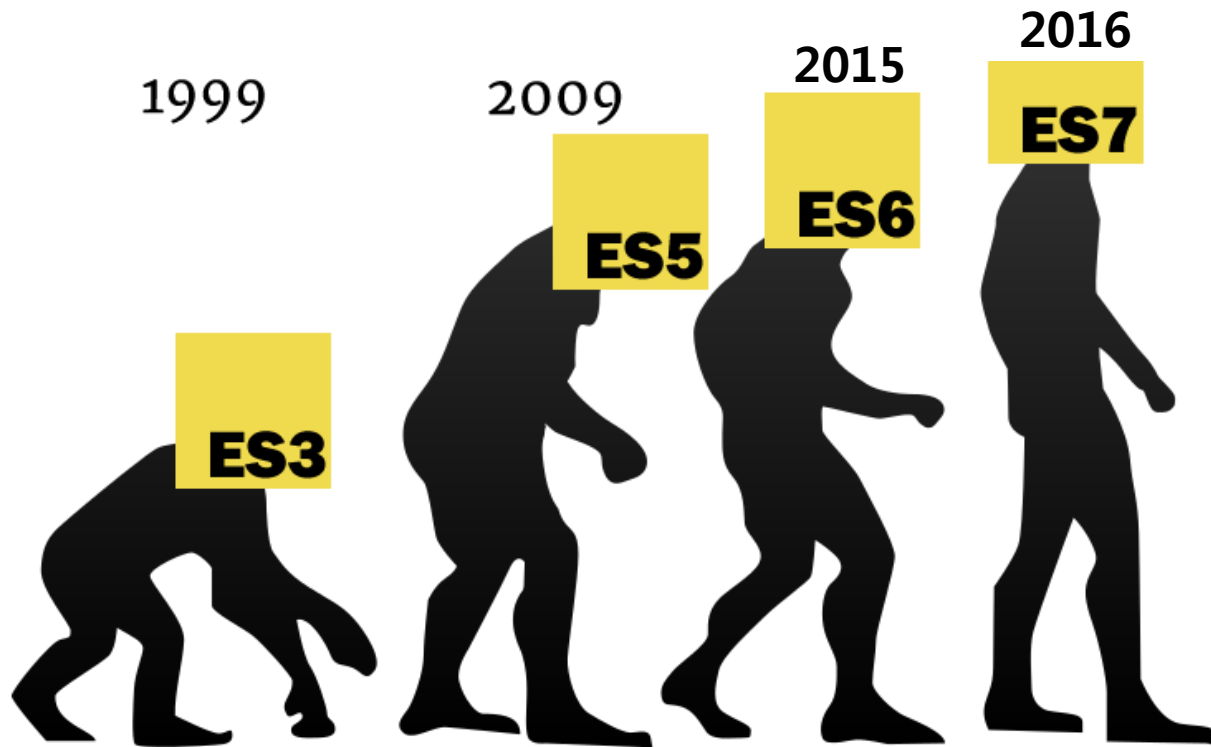
- ECMA International은 전세계적인 표준기관

유럽 컴퓨터 제조회사로부터 기원함

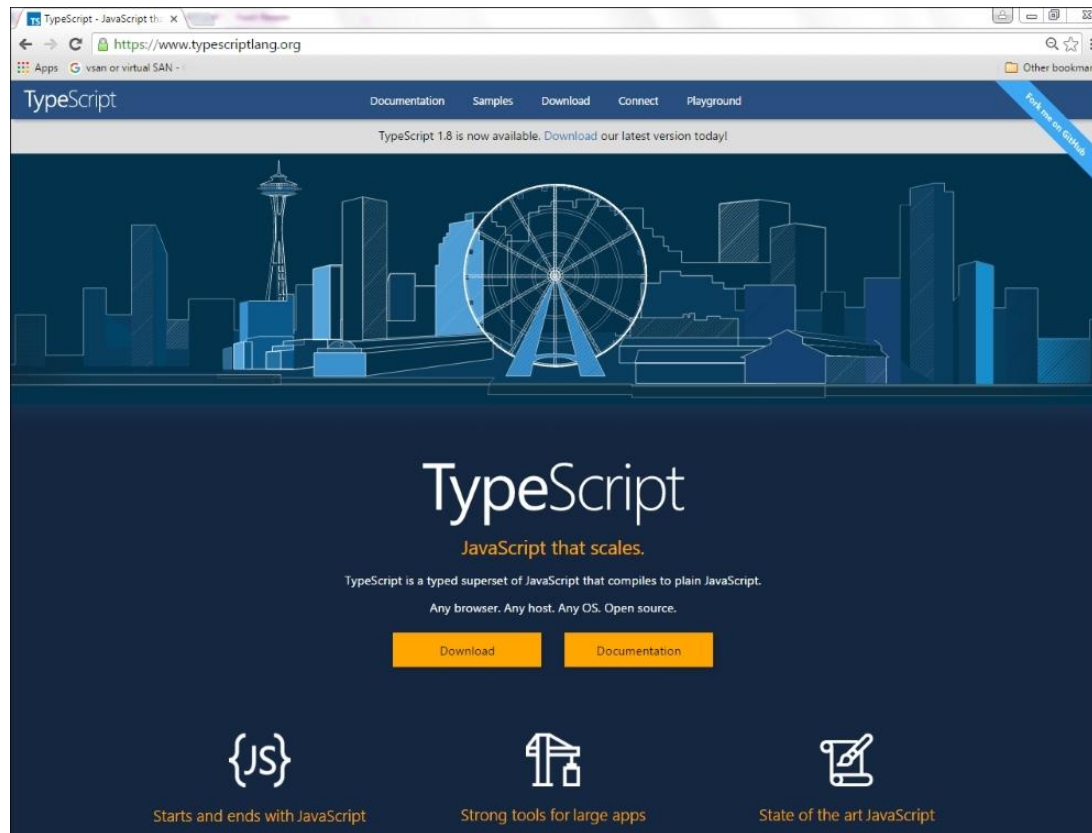
*ECMA(European Computer Manufacturers Association)*

C#, JSON, Dart을 포함한 많은 언어 표준을 관리함

# ECMA Script(ES) 히스토리



# TypeScript ■ [www.typescriptlang.org](http://www.typescriptlang.org)



# TypeScript의 역사

---



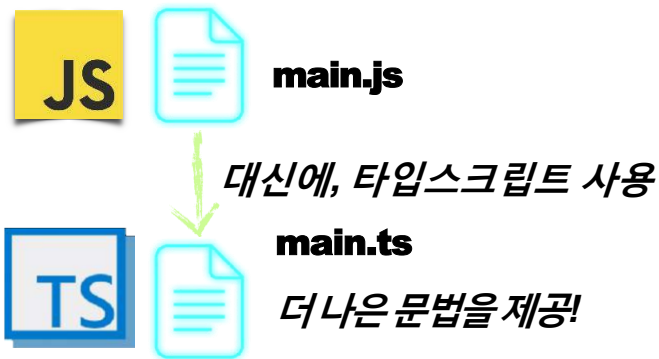
- 2012년 10월 첫 타입스크립트 버전 0.8 발표
- 2013년 6월 18일 타입스크립트 버전 0.9 발표
- 2014년 2월 25일 Visual Studio 2013 빌트인 지원
- 2014년 4월 2일 타입스크립트 1.0 발표
- 2014년 7월 타입스크립트 컴파일러 발표, GitHub 이전
- 2016년 5월 타입스크립트 2.0 발표
- 2017년 6월 타입스크립트 2.5
- 2018년 1월 현재 타입스크립트 2.6

# TypeScript의 개요

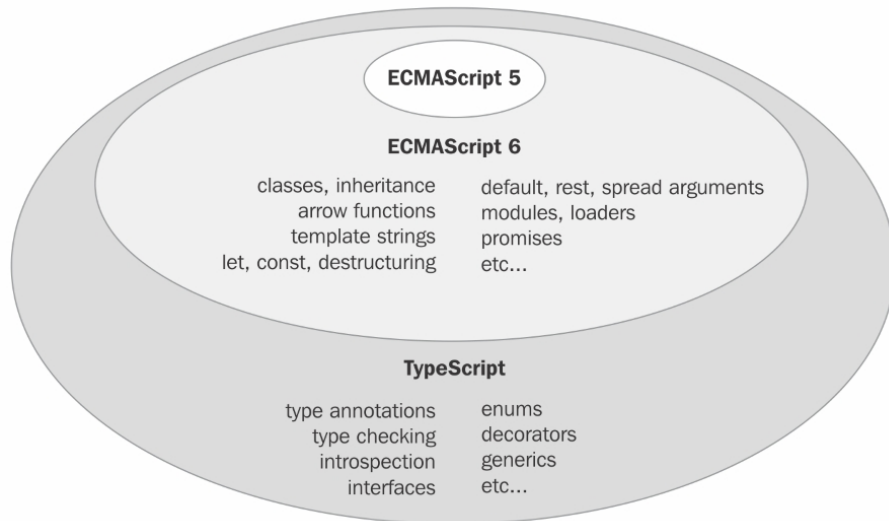


**TypeScript** 는 마이크로소프트( **Anders Hejlsberg** ) 에서 개발한 자바스크립트의 확장된 언어이며, **ES2015** 의 기능에 추가적으로 엄격한 타입체크와 객체지향적 기능 등을 추가한 자바스크립트의 **Superset**이다.

Angular 2 소스는 TypeScript로 개발되었다.



<http://www.typescriptlang.org>

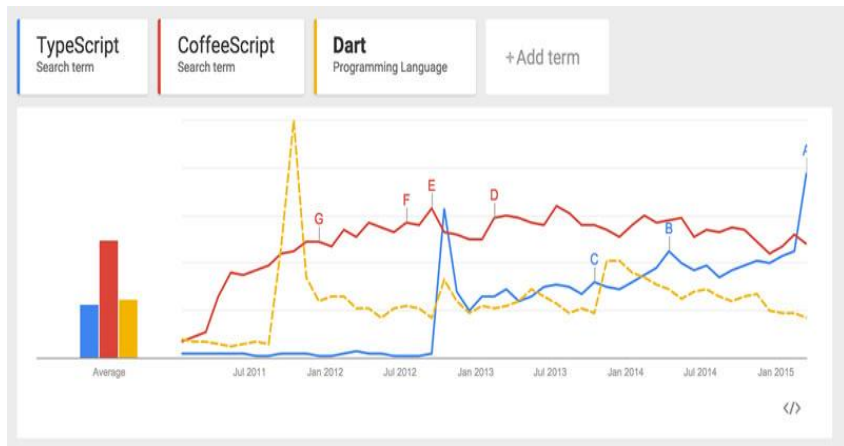




# TypeScript의 개요



- 자바스크립트 → **ES2015(ES6)** → **ES2016(ES7)** → **TypeScript**
- 타입스크립트는 자바스크립트의 미래(?)이다.
- 자바스크립트(**ES6, ES7**) 기능에 타입스크립트의 필요한 기능을 추가하면 된다. (새로운 언어가 아님)
- 클래스 관련기능
- 정적 타이핑 (**static typing**)



# TypeScript의 차별점



- 명시적인 자료형 선언가능

JS

```
var a="10";  
var b=10;  
var sum=a+b;  
console.log(sum);
```

**결과 : 1010**

해석: 20이 출력되길 기대했지만, 자바스크립트의 암묵적(implicit) 형변환은 예측할 수 없는 오류를 만들어 냄



TS

```
function add_error(){  
  let a: string = "10";  
  let b: number = 10;  
  
  let sum: number = a + b;  
  //error!  
  console.log(sum);  
}  
add_error();
```

결과 : 타입이 다른 경우 더하기  
에러가 발생함! (오류가능성 사전  
제거)

# TypeScript의 차별점



- 명시적인 자료형 선언가능

```
function add(){  
  let a: number = 10;  
  let b: number = 10;  
  let sum: number = a + b;  
  console.log(sum);  
}  
  
add();
```

TS

결과 : 20

- 명시적인 자료형 선언으로 가독성이 향상됨
  - 자료형을 명시적으로 정의함으로써 오류를 사전에 감지함
- 예 : 자료형이 다르면 비교나 할당이 불가능함

# TypeScript의 차별점



- 객체지향 프로그래밍 지원

JS

```
var car = (function(){  
    function car(){};  
    car.prototype.getNumTier=function(){};  
    ...  
})();
```



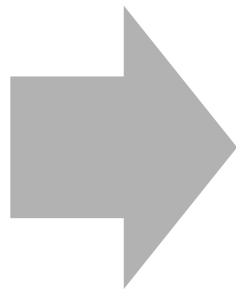
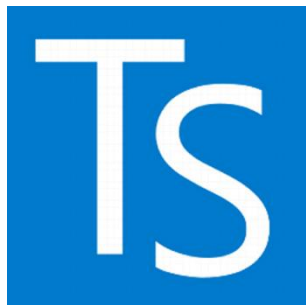
TS

```
class car {  
    numTier: number;  
    constructor(){}  
    getNumTier(){}  
    ...  
}
```

# 트랜스파일러 : tsc



- TSC는 타입스크립트를 자바스크립트로 변환(transpiling)해주는 도구이다.



트랜스파일링

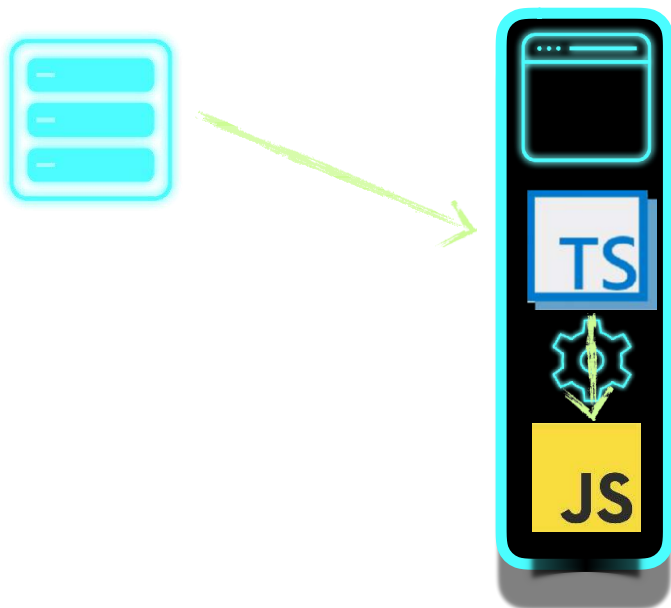
# TypeScript : 트랜스파일 위치



브라우저는 타입스크립트를 해석할 수 없으며, 자바스크립트로 변환하여 브라우저에서 처리되어야 한다. 다음 두 가지 방식이 사용되고 있다.

브라우저에서 자바스크립트로 변환

자바스크립트로 변환 후 브라우저로 로딩



*이 방식이 더 빠르며, 주로 이렇게 개발하게 된다.*

# TypeScript : Playground



TS TypeScript - JavaScript x TS Playground - TypeScript x

← → ↻ ⓘ www.typescriptlang.org/play/index.html ☆

TypeScript Documentation Samples Download Connect Playground

TypeScript 2.2 is now available. [Download](#) our latest version today!

Fork me on GitHub

Select... TypeScript Share Options Run JavaScript

```
1 let myAdd2 = function(x: number, y: number)
2
3 let myAdd3: (baseValue:number, increment:number) => number
4   function(x: number, y: number): number
5
6 let myAdd: (x: number, y: number) => number
7   function(x: number, y: number): number
8 console.log(myAdd(10,20));
9
```

```
1 var myAdd2 = function (x, y) { return x + y; }
2 var myAdd3 = function (x, y) { return x + y; }
3 var myAdd = function (x, y) { return x + y; }
4 console.log(myAdd(10, 20));
5
```

# TypeScript : Download



## Get TypeScript

### Node.js

The command-line TypeScript compiler can be installed as a Node.js package.

#### INSTALL

```
npm install -g typescript
```

#### COMPILE

```
tsc helloworld.ts
```

### Visual Studio



Visual Studio 2017



Visual Studio 2015



Visual Studio Code

### And More...



Sublime Text



Atom



Eclipse



Emacs



WebStorm



Vim



# TypeScript 구성요소



Strongly  
Typed

Classes

Interfaces

Generics

Modules

Type  
Definitions

Compiles to  
JavaScript

EcmaScript 6  
Features

# TypeScript : 타입 시스템

---



- **String**
- **Number**
- **Boolean**
- **Array**
- **Dynamic Typing**
- **Enum**
- **Void**

# TypeScript : 클래스



- class 키워드를 이용하여, 클래스 정의

```
class car {  
  numTier: number;  
  carName: string;  
  
  constructor(  
    carName: string, numTier: number){  
    this.carName = carName;  
    this.numTier = numTier;  
  }  
  getNumTier(){  
    return this.numTier;  
  }  
  getCarName(){  
    return this.carName;  
  }  
}
```

```
let myCar = new car("해피카",4);  
console.log(myCar.getCarName()+"의 타이어  
개수는 "+myCar.getNumTier()+"개 입니다.");
```

[결과]  
해피카의 타이어 개수는 4개 입니다.

# TypeScript : 상속



- extends 키워드를 이용하여 부모 클래스를 상속

```
class HappyCar {
  numTier: number;
  carName: string;
  speed: number;

  constructor(carName: string, numTier: number){
    this.carName = carName;
    this.numTier = numTier;
  }

  setSpeed(speed: number){
    this.speed=speed;
  }

  getSpeed(){
    return this.speed;
  }
}
```

```
class bus extends HappyCar {
  constructor(carName: string, numTier: number) {
    super(carName,numTier);
  }
  setSpeed(speed = 0) {
    super.setSpeed(speed);
  }
}

class truck extends HappyCar {
  constructor(carName: string, numTier: number) {
    super(carName,numTier);
  }
  setSpeed(speed = 0) {
    super.setSpeed(speed);
  }
}
```



- 인스턴스 생성 후 테스트

```
let myBus = new bus("myBus",6);  
let myTruck: HappyCar = new truck("myTruck",10);  
  
myBus.setSpeed(100);  
myTruck.setSpeed(120);  
console.log("현재 버스속도 : "+myBus.getSpeed());  
console.log("현재 트럭속도 : "+myTruck.getSpeed());
```

[결과]

현재 버스속도 : 100

현재 트럭속도 : 120

# TypeScript : 인터페이스



- Interface에 선언된 변수나 메서드에 대한 사용을 강제함

```
interface AddressInterface {  
    addressBookName:string;  
    setBookName(addressBookName: string);  
    getBookName();  
}  
  
class AddressBook implements AddressInterface {  
  
    addressBookName:string;  
    setBookName(addressBookName: string) {  
        this.addressBookName = addressBookName;  
    }  
    getBookName(){  
        return this.addressBookName;  
    }  
    constructor() { }  
}
```

```
let myAddressBook = new AddressBook();  
myAddressBook.setBookName("나의 주소록");  
console.log(myAddressBook.getBookName());
```

[결과]  
나의 주소록

# TypeScript : module



- ECMAScript 2015에서의 module 개념, export 와 import

*Validation.ts*

```
export interface StringValidator {  
  isAcceptable(s: string): boolean;  
}
```

*ZipCodeValidator.ts*

```
import { StringValidator } from "./Validation";  
  
export const numberRegex = /^[0-9]+$/;  
export class ZipCodeValidator implements StringValidator  
{  
  isAcceptable(s: string) {  
    return s.length === 5 && numberRegex.test(s);  
  }  
}
```

*Test.ts*

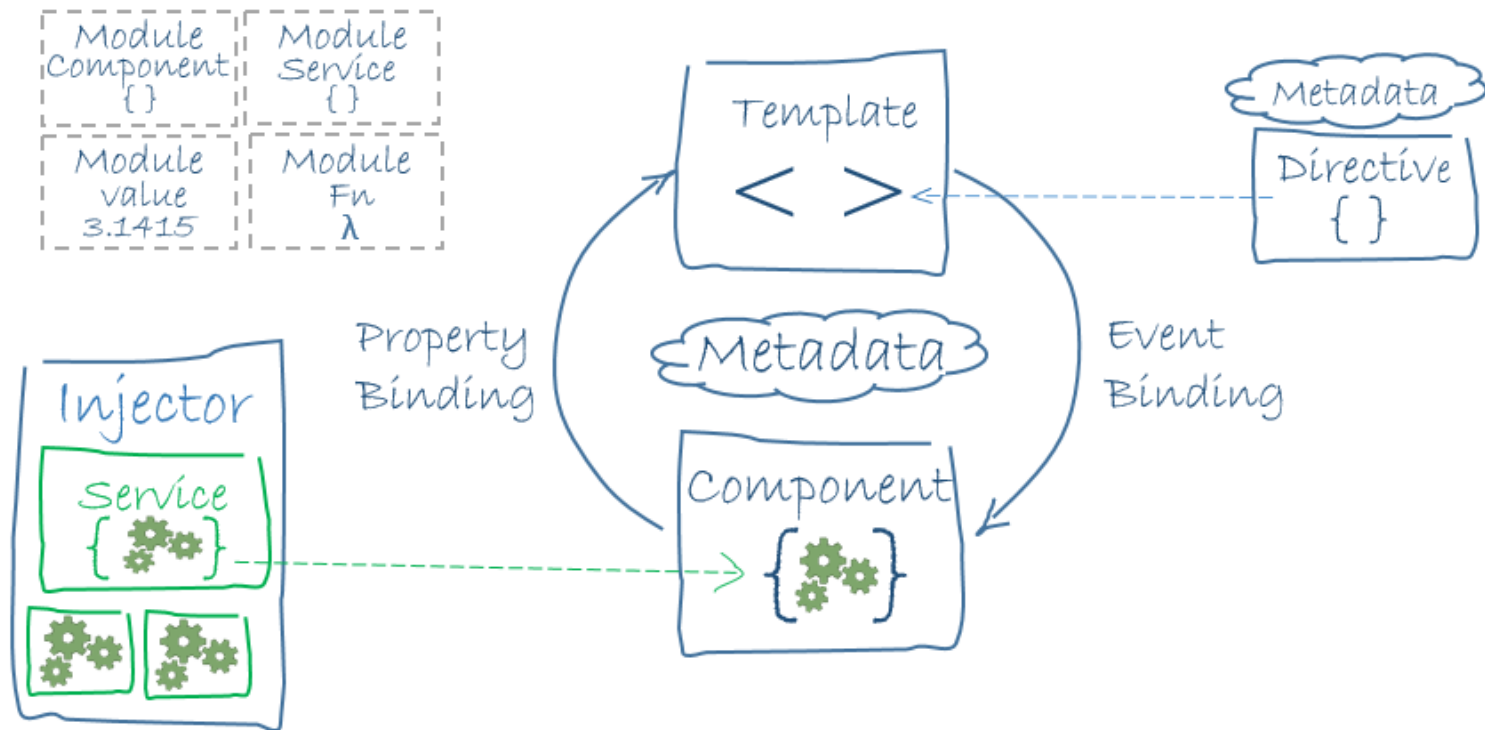
```
import { ZipCodeValidator } from "./ZipCodeValidator";  
let myValidator = new ZipCodeValidator();
```

# Angular 아키텍처와 주요 컴포넌트

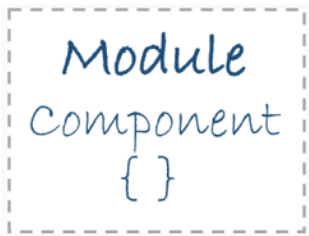


# Angular Architecture

- Component를 이용한 Template과 Service 관리



# 1. Modules



- 모든 Angular App은 1개의 NgModule(root module) 을 가지고 있다.
- Root module의 이름은 일반적으로 AppModule 이다.
- Module은 @NgModule Decorator를 선언한다.

## @NgModule Decorator의 속성들.

declarations	module에 속한 view 클래스들(component, directive, pip)을 선언한다.
exports	declaration에서 정의한 클래스들이 다른 module에서 사용되는 경우에 export 해야 한다.
imports	필요해서 import 하는 다른 module들을 선언한다.
providers	모든 App에 접근 가능한 Service를 선언한다.
bootstrap	root component 역할을 하는 main component view를 선언한다.

# 1. Modules

## src/app/app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports: [ BrowserModule ],
  providers: [ Logger ],
  declarations: [ AppComponent ],
  exports: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

## src/app/main.ts

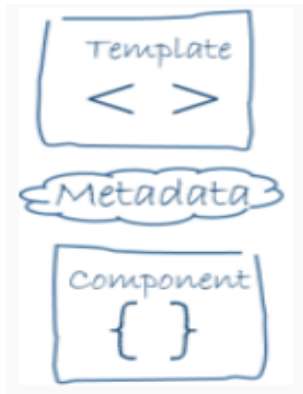
```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule);
```

## 2. Component와 Template



- Component는 view(Template)을 제어하는 역할을 한다.
- View를 support 하기 위해 Component내에 application logic을 정의할 수 있다.
- Component는 @Component Decorator를 선언한다.

@Component Decorator의 속성들.

selector	View에서 사용되는 Tag를 설정한다. 이 Tag를 만나면 Angular component가 생성되어진다.
template	Component 내에 View역할을 하는 HTML Code를 포함한다.
templateUrl	View 역할을 하는 HTML Code를 별도의 html 파일로 정의하고, 그 파일명을 지정한다.
styleUrls	Style을 별도의 CSS 파일로 정의하고, 그 파일명을 지정한다.
providers	특정 component에서만 접근 가능한 Service를 선언한다

## 2. Component & Template

src/app/hero-list.component.ts

```
import { Component } from '@angular/core';
import { Hero } from './hero';
import { HeroService } from './hero.service';

@Component({
  selector: 'app-hero-list',
  templateUrl: './hero-list.component.html',
  providers: [ HeroService ]
})
export class HeroListComponent implements OnInit {
  heroes: Hero[];
  selectedHero: Hero;

  constructor(private service: HeroService) { }

  ngOnInit() {
    this.heroes = this.service.getHeroes();
  }

  selectHero(hero: Hero) { this.selectedHero = hero; }
}
```

## 2. Component와 Template

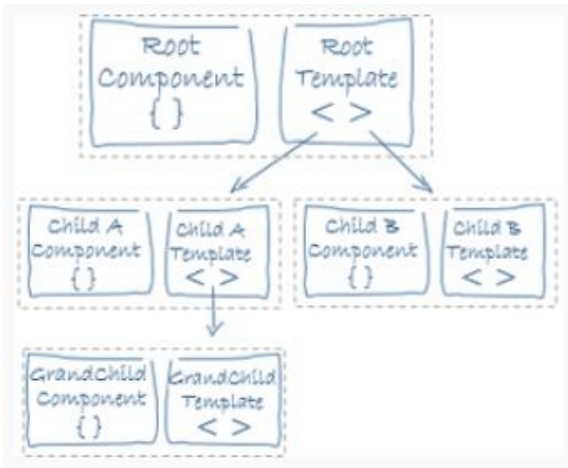
src/app/hero-list.component.html

```
<h2>Hero List</h2>

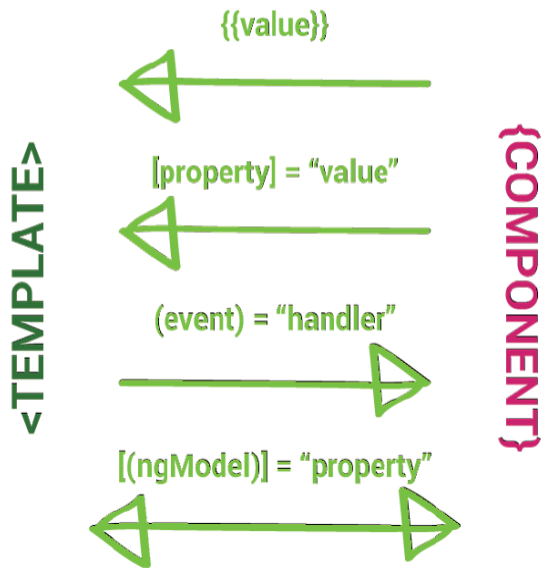
<p><i>Pick a hero from the list</i></p>
<ul>
  <li *ngFor="let hero of heroes" (click)="selectHero(hero)">
    {{hero.name}}
  </li>
</ul>

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

- <app-hero-detail> tag 는 새로운 HeroDetailComponent 를 나타낸다.
- HeroDetailComponent 는 HeroListComponent의 child 이다.
- \*ngFor, {{hero.name}}, (click), [hero], <app-hero-detail> 는 Angular의 Template Syntax 이다.



# 3. Data Binding



Data direction	Syntax	Type
One-way from data source to view target	<code>{{expression}}</code> <code>[target]="expression"</code> <code>bind-target="expression"</code>	Interpolation Property Attribute Class Style
One-way from view target to data source	<code>(target)="statement"</code> <code>on-target="statement"</code>	Event
Two-way	<code>[(target)]="expression"</code> <code>bindon-target="expression"</code>	Two-way

### 3. Data Binding 문법

Type	Target	Examples
Property	Element property Component property Directive property	<div>src/app/app.component.html</div> <pre>&lt;img [src]="heroImageUrl"&gt; &lt;app-hero-detail [hero]="currentHero"&gt;&lt;/app-hero-detail&gt; &lt;div [ngClass]="{'special': isSpecial}"&gt;&lt;/div&gt;</pre>
Event	Element event Component event Directive event	<div>src/app/app.component.html</div> <pre>&lt;button (click)="onSave()"&gt;Save&lt;/button&gt; &lt;app-hero-detail (deleteRequest)="deleteHero()"&gt;&lt;/app-hero-detail&gt; &lt;div (myClick)="clicked=\$event" clickable&gt;Click me&lt;/div&gt;</pre>
Two-way	Event and property	<div>src/app/app.component.html</div> <pre>&lt;input [(ngModel)]="name"&gt;</pre>



# 3. Data Binding

---

Type	Target	Examples
Attribute	Attribute (the exception)	<div>src/app/app.component.html</div> <div>&lt;button [attr.aria-label]="help"&gt;help&lt;/button&gt;</div>
Class	Class Property	<div>src/app/app.component.html</div> <div>&lt;div [class.special]="isSpecial"&gt;Special&lt;/div&gt;</div>
Style	Style property	<div>src/app/app.component.html</div> <div>&lt;button [style.color]="isSpecial ? 'red' : 'green'"&gt;</div>

# 3. Data Binding

src/app/hero-list.component.html (binding)

```
<li>{{hero.name}}</li>
<app-hero-detail [hero]="selectedHero"></app-hero-detail>
<li (click)="selectHero(hero)"></li>
<input [(ngModel)]="hero.name">
```

- {{hero.name}} interpolation 은 component의 hero.name property 값을 출력한다.
- [hero] property binding 은 parent HeroListComponent 가 child HeroDetailComponent 에  
게 selectedHero 의 값을 전달한다.
- (click) event binding 은 hero's name 을 클릭했을 때 component의 selectHero method 을 호  
출한다.
- ngModel Directive를 사용해서 property binding과 event binding을 한번에 동시에 처리한다.

## 4. Directive



- Directive는 @Directive Decorator를 선언한다.
- Structural Directive와 Attribute Directive 두 가지 종류가 있다.
- Structural Directive는 DOM 엘리먼트를 추가,삭제 , 갱신할 수 있다.

Structural Directive : ngFor , ngIf

src/app/hero-list.component.html (structural) <input [(ngModel)]="hero.name">

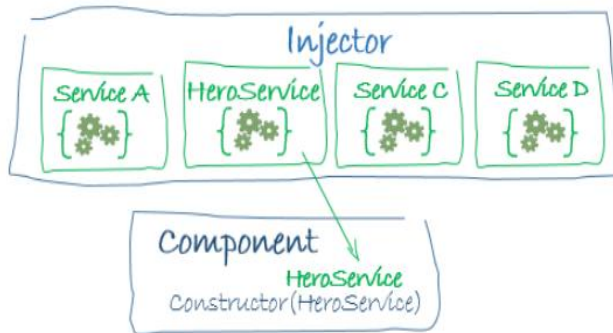
```
<li *ngFor="let hero of heroes"></li>
<app-hero-detail *ngIf="selectedHero"></app-hero-detail>
```

Attribute Directive : ngModel

src/app/hero-detail.component.html (attribute)

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

## 5. Service



- Service는 **@Injectable** Decorator를 선언한다.
- Service에는 다양한 기능들을 정의할 수 있다.
- 예) logging service, data service, message bus, tax calculator
- Component는 Service의 big consumer 이다.
- Angular는 Service를 Component에게 제공하기 위해 DI(dependency Injection) 사용한다.
- Dependency Injection은 의존관계에 있는 다른 객체에게 새로운 객체를 제공하기 위한 방법이다.

## 5. Service

src/app/hero.service.ts

```
@Injectable()
export class HeroService {
  private heroes: Hero[] = [];

  constructor(
    private backend: BackendService, private logger: Logger) { }

  getHeroes() {
    this.backend.getAll(Hero).then( (heroes: Hero[]) => {
      this.logger.log(`Fetched ${heroes.length} heroes.`);
      this.heroes.push(...heroes); // fill cache
    });
    return this.heroes;
  }
}
```

src/app/hero-list.component.ts

```
constructor(private service: HeroService) { }
```

## 5. Service

- Service 객체를 Root module의 providers로 등록하면 , 모든 Component 들이 접근할 수 있다.

src/app/app.module.ts

```
providers: [  
  BackendService,  
  Logger  
],
```

- Service 객체를 Component의 providers로 등록하면 , 특정 Component 에서만 접근할 수 있다.

src/app/hero-list.component.ts

```
constructor(private service: HeroService) { }  
  
@Component({  
  selector: 'app-hero-list',  
  templateUrl: './hero-list.component.html',  
  providers: [ HeroService ]  
})
```

# Dependency Injection

---

## 의존성 주입(Dependency Injection, DI)

■ 프로그래밍에서 구성요소간의 의존 관계가 소스코드 내부가 아닌 외부의 설정파일 등을 통해 정의되게 하는 디자인 패턴 중의 하나이다.

## 의존성 주입(Dependency Injection, DI)의 장점

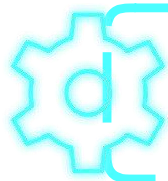
- ① 의존 관계 설정이 컴파일시점이 아닌 실행시점에 이루어져 모듈들간의 결합도를 낮출 수 있다.
- ② 코드 재사용을 높여서 작성된 모듈을 여러 곳에서 소스코드의 수정 없이 사용할 수 있다.
- ③ **mock** 객체 등을 이용한 단위 테스트의 편의성을 높여준다.

# Dependency Injection

컴포넌트 “**providers**” 에 등록되어 사용.

서비스 객체 — 주로 *Injector*로 사용된다.

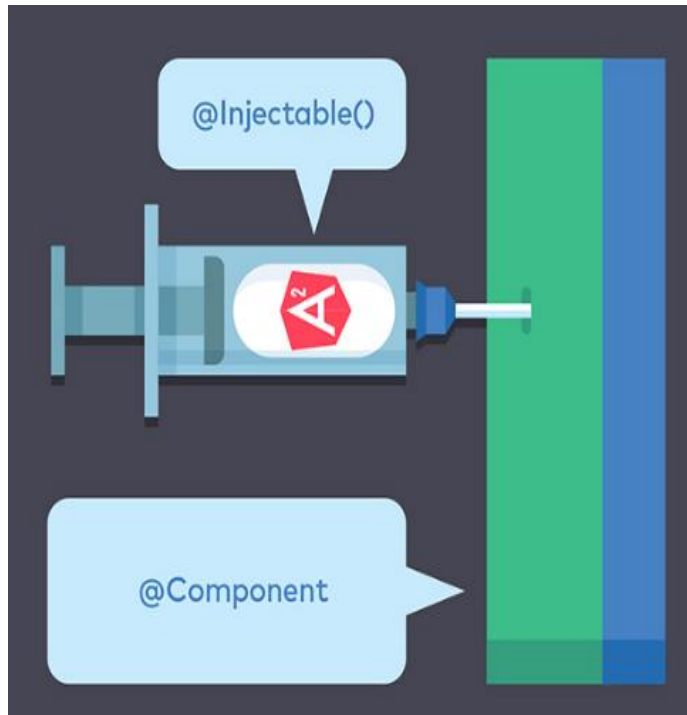
## Dependency Injector



`racing-data.service.ts`  
`another.service.ts`  
`api.service.ts`

HeroService 사용하기 위한 세 가지 절차 :

1. HeroService에 **@Injectable** 데코레이터를 추가.
2. app 모듈에 providers 로 등록.
3. hero-list.component.ts에 의존성을 주입.





# http 모듈

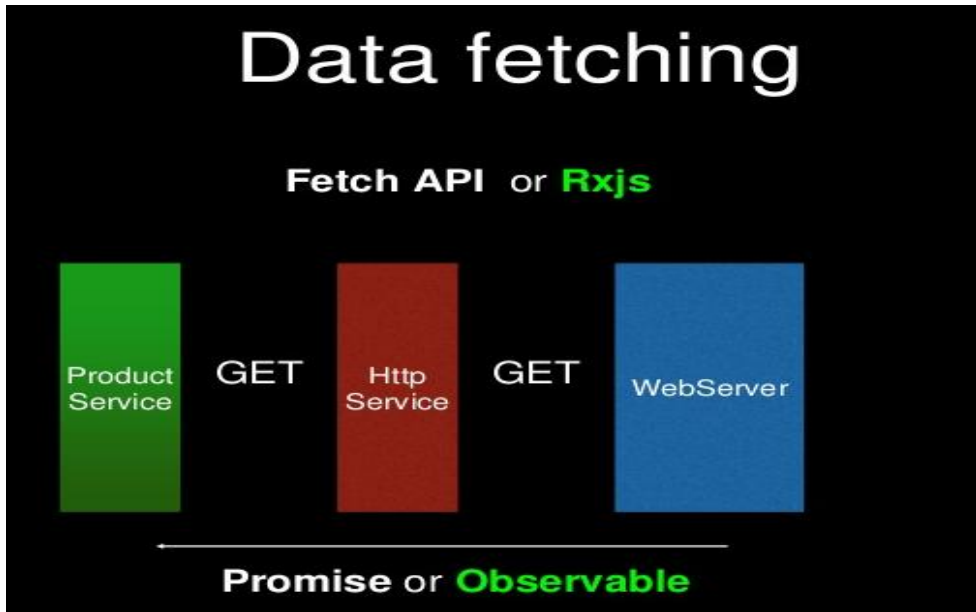
웹에 연결하여 데이터를 가져오자!

# Observable과 Promise 객체

**HTTP** 라이브러리는 원격서버에 접속하여 데이터를 주고 받기위한 라이브러리이다.

**RxJS** 라이브러리는 **Reactive Extensions** 의 약자이고, **http** 요청에 대한 여러가지 기능을 제공한다.

**Angular http**는 **RxJS**의 **Observable**과 **ES6**의 **Promise** 객체를 사용한다.



# Angular의 Http Module

---

Angular에서 제공하는 HTTP 모듈은 RxJS Observable 객체를 기반으로 만들어졌다.

: Angular에서 실행하는 HTTP 관련 함수는 Observable을 반환하며, 이 객체를 subscribe(구독)해서 사용할 수 있다는 뜻이다.

HTTP 요청 함수를 사용할 때 아래의 내용을 염두에 두어야 한다.

1) Observable을 subscribe(구독)하지 않으면 어떠한 요청도 발생하지 않는다. 실제 HTTP 요청은 subscribe()

함수를 실행할 때 발생한다.

2) 하나의 Observable 객체를 여러 번 구독하면, HTTP 요청이 여러 번 발생한다.

3) Observable은 하나의 데이터를 전달하는 스트림이기 때문에, HTTP 요청이 한 번 발생 할 때마다 하나의 데이터를 반환한다.

4) HTTP 요청이 실패하면 Observable 이 에러를 반환한다.

# Angular의 Http Module

---

HTTP Client 모듈을 설치하려면 app.module.ts 에 다음과 같이 HttpClientModule을 추가해야 한다.

src/app/app.module.ts

```
import {HttpClientModule} from '@angular/common/http';

@NgModule({
  declarations: [ AppComponent ],
  imports: [
    BrowserModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

# Angular의 Http Module

---

HTTP Client 을 사용하여 데이터를 요청하기

src/app/app.module.ts

```
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  title = 'app';
  results = '';
  constructor(private http: HttpClient){
  }
  ngOnInit(): void {
    this.http.get('https://api.github.com/users').subscribe(data => {
      console.log(data);
    });
  }
}
```



판교 | 경기도 성남시 분당구 삼평동 대왕판교로 670길 유스페이스2 B동 8층 T. 070-5039-5805  
가산 | 서울시 금천구 가산동 37-47 이노올렉스 1차 2층 T. 070-5039-5815  
웹사이트 | <http://edu.kosta.or.kr> 팩스 | 070-7614-3450