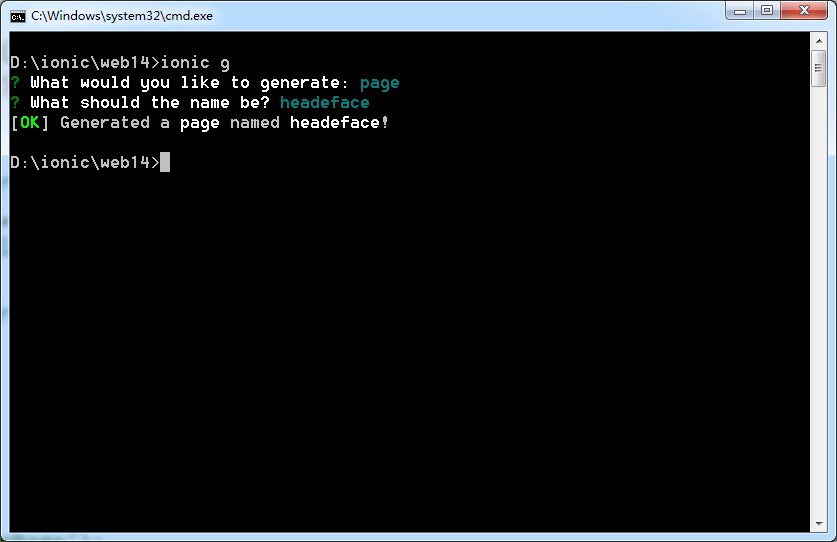
跨平台开发Ionic4 第八节----个人中心=>底层camera

1. 需要有一个headeface的page



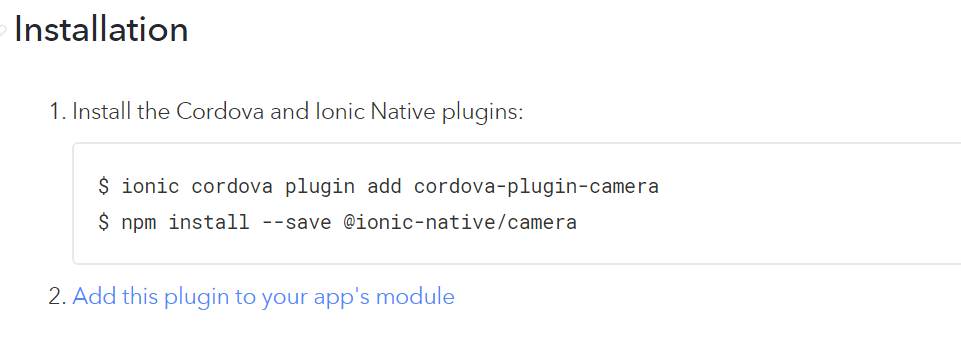
1. App.moudle.ts引入我们的headefacepage
2. Userpage中引入我们的headefacepage
3. Headefacepage需要先写好html，
4. Headeface.ts中需要做好准备：
5. 点击选择图库还是照相机：

<https://ionicframework.com/docs/components/#action-sheets；>

1. Camera底层组件需要进入api----并且进行命令行安装

<https://ionicframework.com/docs/native/camera/>

如图所示两步骤：



1. 首先进入到app.module.ts页面中,需要如图所示：



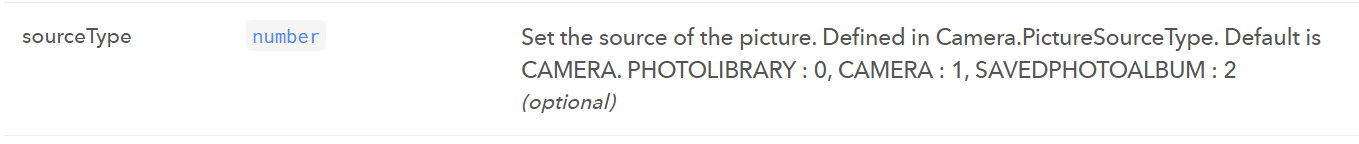
教程地址：

<https://ionicframework.com/docs/native/#Add_Plugins_to_Your_App_Module>

1. Camera用法：

<https://ionicframework.com/docs/native/camera/>

1. sourceType====调用底层摄像头还是图库：



sourceType: this.camera.PictureSourceType.CAMERA => 调用照相机

sourceType: this.camera.PictureSourceType.PHOTOLIBRARY => 调用图库

1. 写一个函数,封装底层的图片获取方法：

text(sourceType){

const options: CameraOptions = {

quality: 100,

sourceType: sourceType,

destinationType: this.camera.DestinationType.FILE\_URI,

encodingType: this.camera.EncodingType.JPEG,

mediaType: this.camera.MediaType.PICTURE

}

this.camera.getPicture(options).then((imageData) => {

// imageData is either a base64 encoded string or a file URI

// If it's base64 (DATA\_URL):

let base64Image = 'data:image/jpeg;base64,' + imageData;

this.lastImage=base64Image;

}, (err) => {

// Handle error

});

}

1. 注意事项：我们引入camera需要这么引入：

import { Camera, CameraOptions, PictureSourceType } from '@ionic-native/camera';

1. 选择返回值的格式。Camera.DestinationType中定义。默认是FILE\_URI。DATA\_URL: 0，以base64编码的字符串返回图像(DATA\_URL可能占用大量内存，并导致应用程序崩溃或内存不足。如果可能，使用FILE\_URI或NATIVE\_URI)， FILE\_URI: 1，返回图像文件URI, NATIVE\_URI: 2返回图像原生URI(例如，在iOS上的资产库://或在Android上的内容://)
2. 如图所示函数嵌套：

