LAST UPDATED **26 JULY 2019**
IONIC VERSION CHECK **5.2.3**

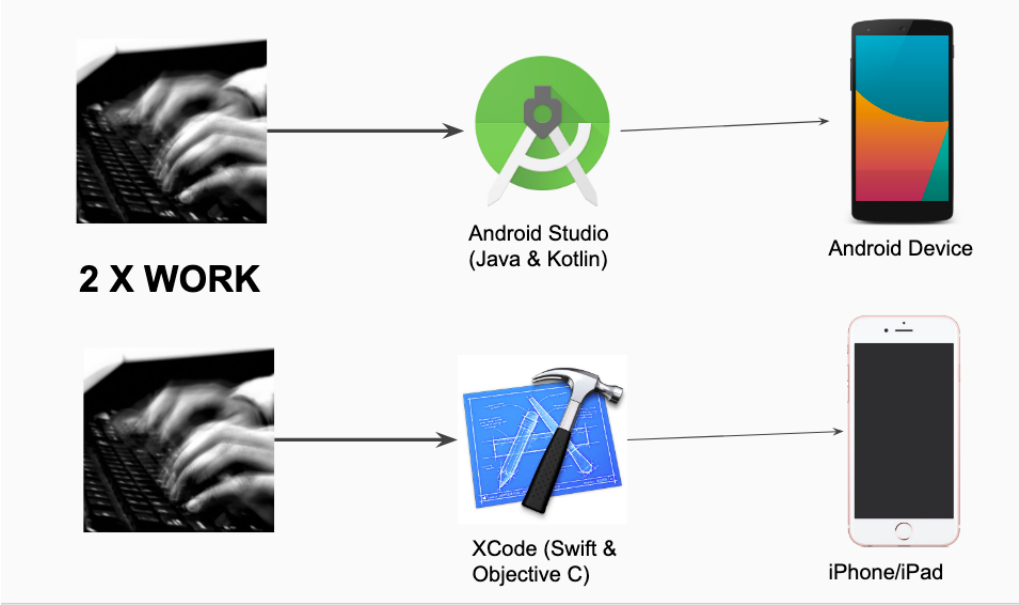
LAST UPDATED **26 JULY 2019**
IONIC VERSION CHECK **5.2.3**

Chapter 1: About Ionic

iOS & Android Mobile Application Development in a nutshell

Native Development

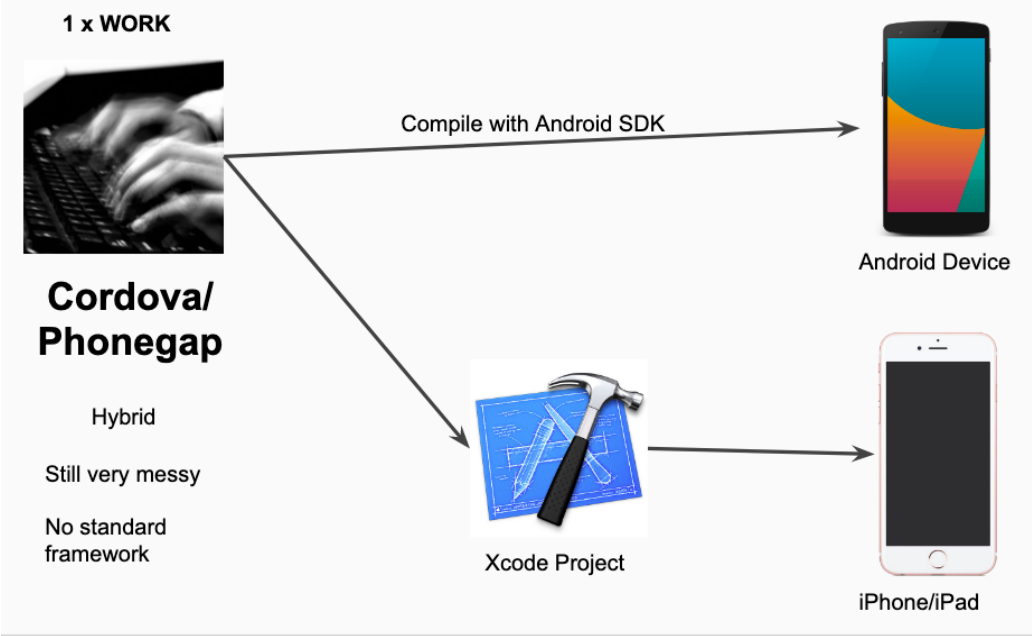
In order to build a native mobile application, it would require 2 different distinct codebase to develop for either Android or XCode:



Hybrid Development

Hybrid development came into the picture to assist mobile app developers to build mobile applications with one single codebase. By creating a web layer on top of a native layer, developers are able to use knowledge in web development to develop mobile applications. However, this was not enough, the codes are very messy because there is no standard framework to control the device capabilities and theme the applications effectively. Early hybrid mobile applications were also ‘slow’ (Please note: differences in micro seconds)

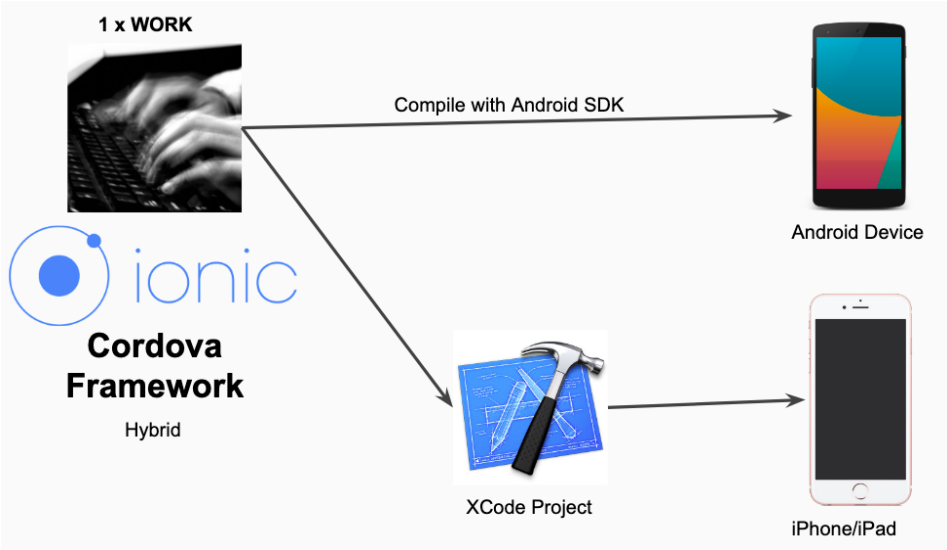
LAST UPDATED **26 JULY 2019**
IONIC VERSION CHECK **5.2.3**



Ionic Framework

Ionic Framework is a hybrid framework that provides a standard framework to assist developers to effectively build mobile applications with a standard set of user interface controls and components and native access to devices' API, such as cameras, vibration and device controls in a single codebase.

Using Angular framework (which has since been expanded to support many JavaScript/TypeScript based UI frameworks) as the basis for the controllers and HTML conventions, developers are able to build fully fledged mobile application with performance on par with natively develop mobile applications



LAST UPDATED **26 JULY 2019**
IONIC VERSION CHECK **5.2.3**

Progressive Web Apps and the future of Mobile Application Development

Being a web based hybrid framework, an Ionic Framework project can easily be converted into a Progressive Web Application (PWA). This would make it possible for web apps to be developed with Ionic Framework and installed directly to user's devices, bypassing the Apple App Store and Google Play Store.

With the addition of the Capacitor framework (a new framework to assist Ionic developers, as an alternative to Cordova, by the Ionic team), users are able to build mobile application not only for iOS and Android, but a proper computer software application as well.

Why Ionic?

- Framework specifically to develop mobile apps across multiple platforms
- 'One codebase to rule them all'
- Apps development using HTML, CSS, TypeScript (a subset of JavaScript)
- Ability to use plugins such as cameras, bluetooth and device controls, all of this available to be referred to at ionicframework.com

LAST UPDATED **26 JULY 2019**
IONIC VERSION CHECK **5.2.3**

Chapter 2: Ionic Framework Installation & Starting A Project

What do you need?

To start/run Ionic ● NodeJS (see website, LTS version) ● Node Package Manager (NPM) ● Ionic (installed via NPM, see below) **To turn Ionic project into Android Application** ● Android Studio ● Java Runtime Environment (JRE) 1.8 ● Java Development Kit (JDK) 1.8 **To turn Ionic project into iOS Application** ● Xcode	**To edit/develop** ● Any Integrated Development Environment (IDE), example: VSCode, Atom, Sublime etc. Please install TypeScript plugin if you are using Atom or Sublime ● Google Chrome Web Browser ● Ionic DevApp ● GitLab (All codes for this course will be made available here)

How to install Ionic?

1. Download NodeJS from http://www.nodejs.org . Get the LTS version of Node. LTS stands for 'Long Term Support'.

2. Open command prompt
    a. Windows: Start button-> Search-> type 'cmd'
    b. Mac: Search for 'terminal'

3. Type the following:
    a. node -v
    b. npm -v

4. If both command responds with version numbers you are all set to install Ionic. To install Ionic, type the following:

   a. npm install -g ionic cordova

5. If this is successful, type the following, there should be a response:
   a. ionic -v

6. If all the above steps are completed, you are ready to start building a mobile app with the Ionic Framework

Starting a new Ionic Project

1. In order to start an Ionic project, navigate to the folder with 'cd' of where you want your Ionic project to be

2. Type the following:
   a. ionic start project_name
   b. *Please note there is no spaces allowed in the project name. Changing the app name will be done later, but this is the project name.

3. You will be presented with several project types. For details of the project types, please refer to the next section.

4. After all the steps above is done, you should have your new project folder project_name

Trying out your app - ionic serve

1. You can try to run your app to see what it looks like

2. 'cd' into your project folder and run the following command:
   a. ionic serve

3. This will automatically launch a web browser with your project application

4. If the web browser is Google Chrome, right click and select 'Inspect'

5. Press this button to view the application in a mobile phone format of your choice

LAST UPDATED **26 JULY 2019**
IONIC VERSION CHECK **5.2.3**

Chapter 3: Creating Pages and Navigation

Referring to the codes from the previous application created in the previous chapter, Ionic looks and feel like a typical web project. It uses pages and each page has its own html and typescript functions.

How to create a page?

1. To create a page, inside the project folder, you will need to type the following:
    a. ionic g page name
    b. *Please note, similar to the app name, the page name must not have any spaces, must not include the word 'page'
2. This will generate a new page folder inside your project.
3. You can 'scaffold' the pages by using '/' operator, example:
    a. ionic g page start/abcd
4. This will generate ‘abcd’ page in the start folder, by ‘scaffolding’ your application, you can organise the pages efficiently

How to navigate to a page?

There are 3 methods to navigate to a page:

a. Via HTML
b. Via TypeScript commands
c. With a Modal Controller

The following are instructions on how to navigate to a page AFTER the page has been created (refer to the previous section)

Navigate via HTML

1. At the button or desired HTML component to be used to navigate to another page, insert the following snippet inside the html, in this example we are going to use a button to move between pages:
    a. <ion-button routerDirection=”forward” routerLink=”/first’>Go to First Page</ion-button>

2. This will bring you to the next page using html. However, since it is done in HTML, you can’t place conditions on the code, or do any processes to ensure the navigation is done correctly. For this, you will need to use the TypeScript method.

Navigate via TypeScript

1. The first thing you need to do is add the following code inside the desired page.ts:

```
import { Component, OnInit } from '@angular/core';
import {NavController} from '@ionic/angular'

@Component({
```

2. Next you will need to add into the constructor brackets the following line of code:

```
  constructor(
    public nav:NavController
  ) { }
```

3. After this is done, you can use the nav command with the following line of code in a function that you want to use to navigate back, in this example, to navigate to a page called home:

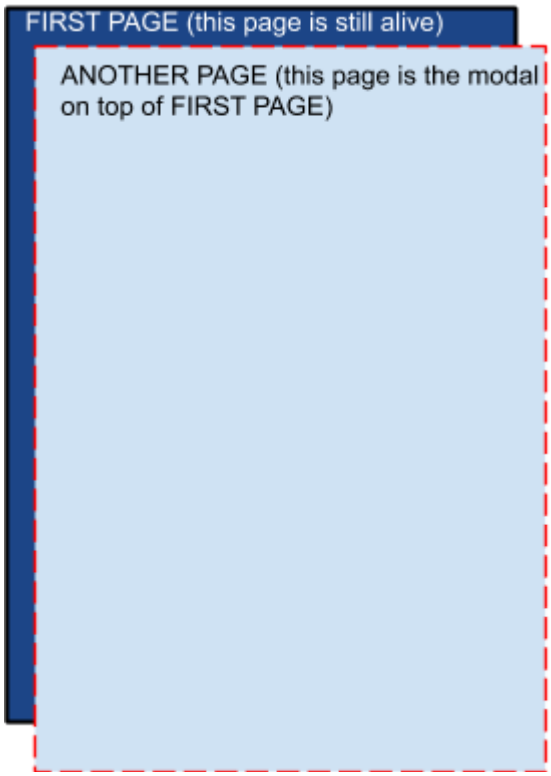
```
        this.nav.navigateForward('home');
```

4. With the above code you are able to navigate forward to a page called home. You can also use other commands such as navigateBack and navigateRoot

Navigate via Modal

In some cases you will want to create a modal. A modal is a page that is opened above an existing page that is still ‘alive’. This can be done in TypeScript part of any page.

LAST UPDATED **26 JULY 2019**
IONIC VERSION CHECK **5.2.3**



1. In order to navigate to a page via Modal, you will need to import the modal page module into the app module. For this example, I will be calling my page new page modal. Open app.module.ts and add the following code at the top of the page:

```
import {ModalPageModule} from './modal/modal.module';
```

2. You will then need to add the page together with the rest of the modules inside the app.module.ts page:

```
18    imports: [BrowserModule, IonicModule.forRoot(), AppRoutingModule, ModalPageModule],
```

3. Next, you will need to import Modal Page inside the page where you will call this particular modal page. In this example, I will add it in a page called home.ts. I will also need to import ModalController, similar to the way how we imported the NavController

```
3  import {NavController, ModalController} from '@ionic/angular';
4
5  import {ModalPage} from '../modal/modal.page';
```

4. I will need to declare the ModalController in the constructor, similar on how we declared this with our NavController

```
14    constructor(
15      public nav:NavController,
16      public mc:ModalController
17    ) {}
```

5. And then, we can use it in any of our functions. In this case, I am going to use it in a function called open_modal():

```
async open_modal()
{
  const modal = await this.mc.create({
    component : ModalPage
  });

  return await modal.present();
}
```

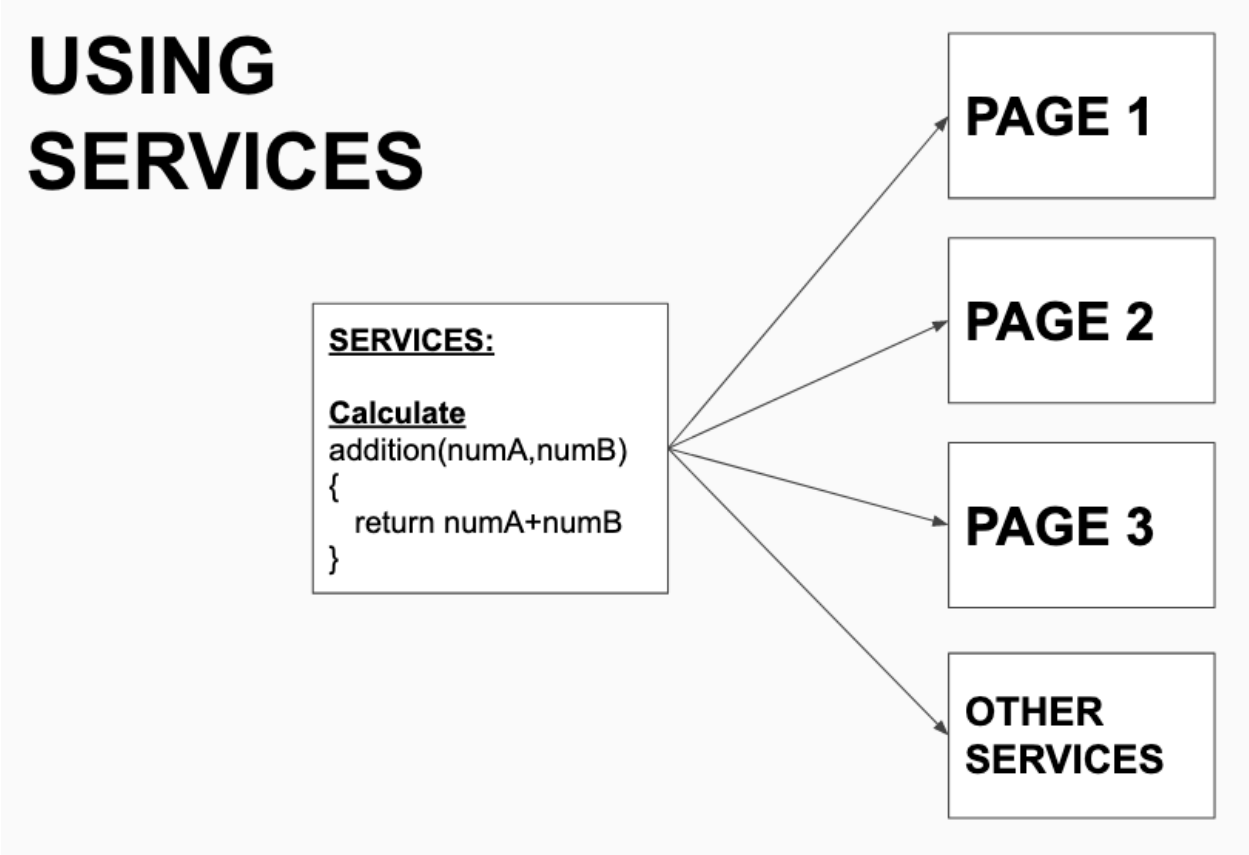
LAST UPDATED **26 JULY 2019**
IONIC VERSION CHECK **5.2.3**

Chapter 4: Service, Components and Pipes

In addition to navigation, it is useful to learn about Services, Components and Pipes and how to use them when building an Ionic project.

What is Services?

Services is particularly useful when creating a mobile application because it allows the developer to place commonly used functions in one single place.



In this example, the function addition is in a service called calculate. It is being used by Page 1, Page 2, Page 3 as well as other services.

After identifying a particular, commonly used, function, the function can be placed in a service and be used anywhere in the application. The advantage here is that any changes can be done at the service itself, rather than on the individual page where the same function is written multiple times.

Services can be used globally, repeatedly and by different pages as well as other services

LAST UPDATED **26 JULY 2019**
IONIC VERSION CHECK **5.2.3**

Using Services

1. Generate a service inside your application by using the following command. In this example, I will be creating a service for calculate:
    a. ionic g service service/calculate

2. This will create a service called calculate in the folder service.

3. You can then add a function into the service. In this example, lets create a service called subtraction to take two values and return the result of subtraction of the two values.

```
19    subtraction(numA,numB)
20    {
21      return Number(numA)-Number(numB);
22    }
```

4. After this is done, we can use this in any page to call the subtraction function, for example to call it in a home page, we will need to first declare the calculate service:

```
3  import { CalculateService } from '../service/calculate.service';
```

5. Declare calculate service in the constructor

```
constructor(
  public calc:CalculateService
) {}
```

6. Finally after this is done, you will be able to use it in any function in your application

```
27    subtract()
28    {
29      this.total = this.calc.subtraction(this.numA, this.numB);
30      this.calc.page1_result=this.total;
31    }
32
```

LAST UPDATED **26 JULY 2019**
IONIC VERSION CHECK **5.2.3**

What is Component?

A component is an independent construct which encapsulates the controller code, the view and styles,and has inputs and outputs.

Using Component

1. Generate a component with the following command. In this example, we will generate a component called Title component:
    a. ionic g component component/title

2. This will generate a component called title in the component folder. You can then edit the component HTML and ts files.
    a. Please take note of the ts title in the app inside the selector ts file

3. In order to use the title component, we will need to create a component module. Create a new file called component.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { IonicModule } from '@ionic/angular';

import {TitleComponent} from './title/title.component';

@NgModule({
  declarations: [TitleComponent],
  imports: [
    CommonModule,
    IonicModule,
  ],
  exports: [TitleComponent]
})
export class ComponentModule { }
```

4. You will then need to add the file into a local module file. In this example, we will add the component inside the home.module.ts file. We will import it and add inside the list of imports inside the home.module.ts page

```
8  import { ComponentModule } from '../component/component.module';
```

```
@NgModule({
  imports: [
    IonicModule,
    CommonModule,
    FormsModule,
    ComponentModule,
    RouterModule.forChild([{ path: '', component: Tab1Page }])
  ],
```

5. After adding the file into the home.module.ts page, it is then possible to use the component inside of your app with the following command inside the home.page.html

```
    <app-title></app-title>
```

What is Pipes?
Similar to components, pipes is an independent construct to filter variables when it is presented inside the app's html files using the curly braces.

How to use Pipes?

1. Similar to components, we will need to generate a new pipe for the project

LAST UPDATED **26 JULY 2019**
IONIC VERSION CHECK **5.2.3**

Chapter 5: Theming, Assets & LocalStorage

Theming your app

Ionic Framework is built to be a blank slate that can easily be customized and modified to fit a brand, while still following the standards of the different platforms.

Theming Ionic apps is now easier than ever. Because the framework is built with CSS, it comes with pre-baked default styles which are extremely easy to change and modify.

There are 9 colours already pre baked inside Ionic that can be used straight away. The 9 colours can also be changed by accessing the ionicframework.com website under the colour generator part of the website https://ionicframework.com/docs/theming/basics

How to use the colours in your app?

The colours can be used in the app through the "color" operator.

1. For example, if you would like to change the colour of a button, this can be done with the following line of code:
    a. <ion-button color="success"> Button with different colour </ion-button>
2. This can also be applied to many other ionic properties throughout your app, such as the ion-toolbar etc

Assets

In certain cases, where you will need to place local assets such as pictures and videos into your applications, you can place these local files inside the assets folder. The assets folder can be located under your src/app folder

How to use locally based assets in your application

1. After placing the file inside the assets folder, the file can be used in your app with the following command, in this example, an image with the name abc.png
    a. <img src="assets/abc.png">

2. Files in assets can be placed into folders to organise and arrange them effectively.