

Zhi Chen

HTML5 HYBRID MOBILE APPLICATION

Building mobile applications using web technologies with Ionic

Thesis

CENTRIA UNIVERSITY OF APPLIED SCIENCES

Information Technology

May 2018

ABSTRACT

Centria University of Applied Sciences	Date May 2018	Author Zhi Chen
Degree programme Information Technology		
Name of thesis HTML5 HYBRID MOBILE APPLICATION. Building mobile applications using web technologies with Ionic		
Instructor Kauko Kolehmainen		Pages 32 + 7
Supervisor Kauko Kolehmainen		
<p>The aim of the thesis is to provide hybrid mobile application using web technologies like HTML, CSS, JavaScript and Ionic.</p> <p>Before building a mobile application, a company needs to decide what types of mobile application to use: native, web or hybrid. Each type has its own advantages and disadvantages. Hybrid mobile application is a good choice when the company wants to build the product in a fast and inexpensive way.</p> <p>The thesis creates two simple hybrid applications with Ionic and RESTful API. The first one is a language learning application which uses NoSQL database firebase as a back end. The second one is a weather application which uses RESTful API from WunderGround.</p>		
Key words Hybrid application, HTML, CSS, JavaScript, Ionic, Angular, Firebase, RESTful API.		

ABSTRACT
CONCEPT DEFINITIONS
CONTENTS

1 INTRODUCTION	1
2 DIFFERENT TYPES OF MOBILE APPLICATIONS	2
2.1 Native application	3
2.2 Web application.....	3
2.3 Hybrid application	4
3 TECHNOLOGY STACK.....	6
3.1 HTML	6
3.2 CSS	7
3.3 Less	7
3.4 JavaScript	8
3.5 Angular	9
3.6 Ionic.....	10
4 CASE STUDY: CREATE A LANGUAGE LEARNING APPLICATION WITH IONIC	11
4.1 Start an Ionic project.....	12
4.2 Add Firebase Realtime Database.....	14
4.3 Create a login page.....	16
4.4 Create courses page	17
4.5 Create lesson list page.....	20
4.6 Create lesson content page	22
5 CASE STUDY: CREATE A WEATHER APPLICATION WITH IONIC	23
5.1 Weather Underground API.....	23
5.2 Build iWeather application	26
6 CONCLUSION	30
REFERENCES.....	31
APPENDICES	

GRAPHS

GRAPH 1. Comparison of different mobile applications	2
GRAPH 2. YouTube web application vs native application	4
GRAPH 3. MarketWatch hybrid application on Android and iOS (Case Studies, Ionicframework, 2018)	5
GRAPH 4. HTML element (Angular, 2018)	7
GRAPH 5. CSS ruleset (Angular, 2018)	7
GRAPH 6. Nesting in Less and compiled to CSS	8
GRAPH 7. Angular Template.....	10
GRAPH 8. Visual Studio Code in macOS.....	11
GRAPH 9. Cordova and Ionic installation in terminal	12
GRAPH 10. Create an app with Ionic CLI	12
GRAPH 11. Code in package.json.....	13
GRAPH 12. Preview login page in a web browser.....	17

GRAPH 13. Courses page.....	20
GRAPH 14. Lesson list page	21
GRAPH 15. Preview of lesson content page	22
GRAPH 16. Preview the weather application in different platforms	23
GRAPH 17. Weather Underground Weather API	24
GRAPH 18. Before and after applying style rules.....	29

TABLES

TABLE 1. Different mobile platforms for native applications.....	3
--	---

1 INTRODUCTION

Over the last few years, there were some indications that hybrid is becoming popular as the preferred way to create mobile applications, especially in the retail markets and enterprise. Hybrid applications mix the performance and device capabilities of native development with the flexibility and simplicity of web applications.

Here is how Hybrid applications work. Developers start by creating a native iOS or Android application with hybrid framework. Within that application, developers embed a web view, which allows displaying a web page within that native application. Web content is created with well-known technologies like HTML, CSS, and JavaScript, which are much more popular than most native application languages (Salesforce, 2016). This allows to not only use existing web technologies but also leverage a larger set of developers to build the application. By displaying certain content and app features within a web view, hybrid application can be updated across multiple platforms quickly and easily, without having to re-deploy the application to the app store.

This is a study case, in which an Asian market store wants to redesign the user interface to place more products on the page. In traditional development, developers have to update the iOS, Android and website applications separately. This not only takes more time to develop and test but also requires users to update their applications, which is a very slow process. With a hybrid app, it is possible to easily update the website one time and the changes are automatically reflected in the iOS and Android application immediately.

This thesis will compare different types of mobile applications and its advantages and disadvantages. Then it will go through the technology stack which is required to build a hybrid application. Finally, there will be two study cases to build two simple hybrid applications using Ionic.

2 DIFFERENT TYPES OF MOBILE APPLICATIONS

There are mainly three types of mobile applications - the native application, the web application and the hybrid application. GRAPH 1 shows the difference among those three types of mobile applications. Native application is downloaded from the Apple Store or the Play Store written in the native language such as objective-c, or Swift for iOS, Java for Android, C# for Windows (Salesforce, 2016). The software lives on the real device. It has full access to the platforms APIs. Web application is a website designed for mobile and sometimes tries to simulate the native application design but it is written in HTML, CSS and JavaScript. (Salesforce, 2016.)



GRAPH 1. Comparison of different mobile applications

As to hybrid application, it is a combination of native application and web application. It is available in the app stores and runs like a native application on mobile phone, while it is written in HTML, CSS and JavaScript as a web application. It has a native layer with embedded HTML pages with basic access to the native APIs (Salesforce, 2016). Apache Cordova is one of such platforms which provides JavaScript APIs to access device capabilities via plugins.

2.1 Native application

Different platform like iOS, Android and Windows provide a different software development kit to create applications (Differencebetween, 2018). These kits compile binary code which the operating system runs. As a result, different platforms require different programming languages and tools to write. TABLE 2 lists different programming languages and development environment of different platforms. The most popular way to get native applications is to download from an app store such as App Store or Google Play. Mobile operating systems also provide additional methods to get the app onto the device. Once the application has been stored on the device the user can launch it anytime without internet.

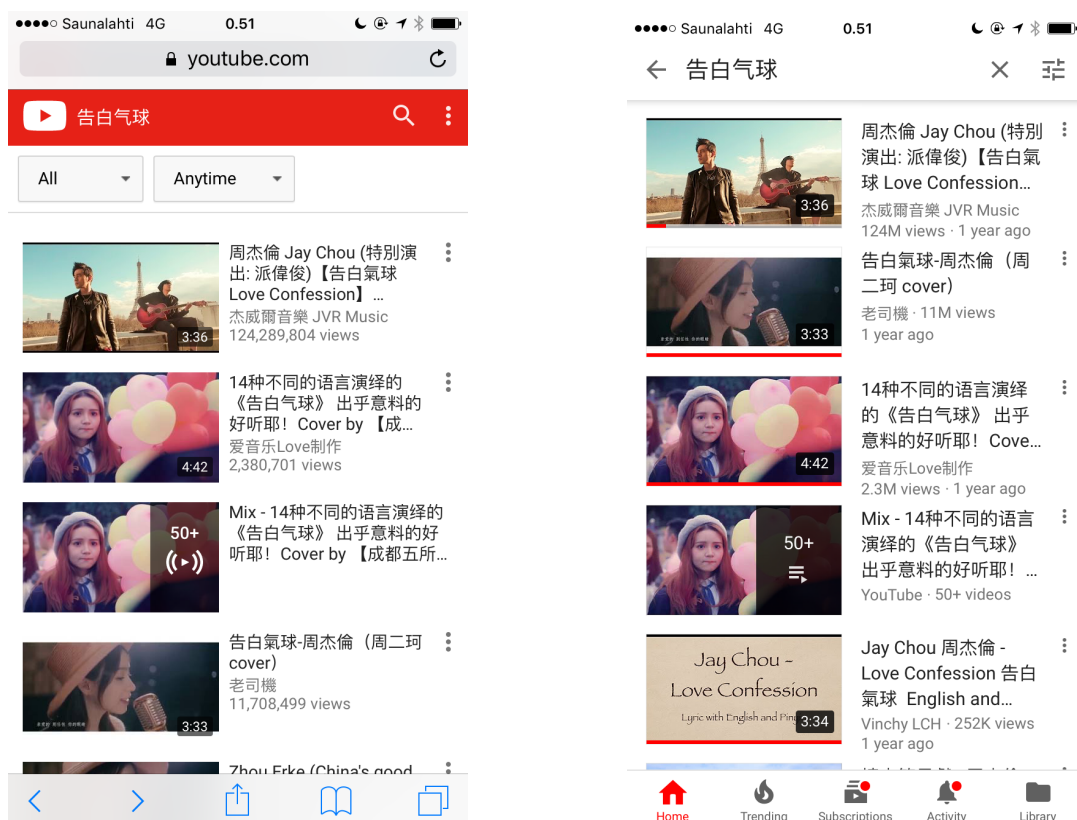
Platform	Language	Development environment
iOS	Object C, Swift	Xcode
Android	Java	Eclipse, Netbeans
Windows	C#	Visual Studio

TABLE 1. Different mobile platforms for native applications

Native apps are very fast and that is because they are built for that specific platform. Native applications are also very easily distributed into app stores whether it is the Apple Store Google Play or the Windows Store because they all have their own language and SDKs. However, they are built for only one platform, which means the company has to hire developers to create iOS, Android or Windows applications separately. As a result, it costs more for the company. It is not a very good idea for a startup company to create native applications separately for each platform because of the high cost. (Salesforce, 2016.)

2.2 Web application

Today's mobile devices come with very powerful browsers with support for many HTML5 features, CSS3 and advanced JavaScript (W3schools, HTML5 Browser Support, 2018). It is possible now to develop very advanced web applications by using them, which brings the next category of mobile web applications. For instance, YouTube web application not only looks very similar to its native application, but also provides very similar functionality. GRAPH 3 is screenshots of YouTube web and native application.

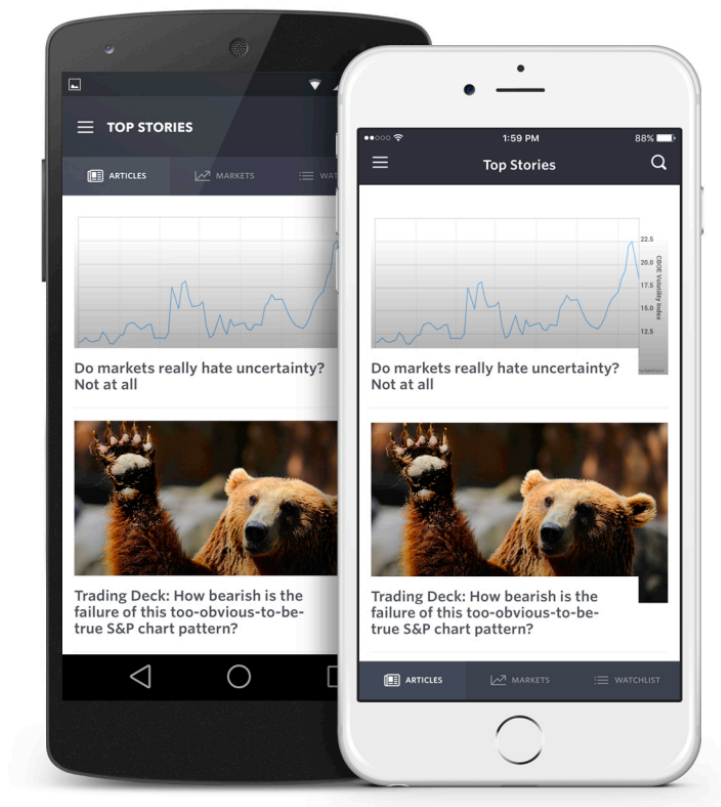


GRAPH 2. YouTube web application vs native application

Web applications are the cheapest option because it is much cheaper for employers to hire one web team instead of iOS and Android teams together. Furthermore, web applications can be displayed on desktop devices. However, as web applications live on web browsers, network connection is a requirement for it to work. Also, users have to remember the URL to access web applications. Username and password are required every time when the user opens the web application. This is not user-friendly and also brings some security problems. (Salesforce, 2016.)

2.3 Hybrid application

Hybrid applications are in between native application and web application. They use web technologies such as HTML5, CSS and JavaScript to write, but are almost indistinguishable to the user from native apps. They are installed on the home page and launched like a native app. MarketWatch is such a Hybrid application (shown in GRAPH 4). MarketWatch delivers the latest financial news and market data. It is now one of the most successful financial application in the market with over 300K users and 16 million visitors per month. (Case Studies, Ionicframework, 2018.)



GRAPH 3. MarketWatch hybrid application on Android and iOS (Case Studies, Ionicframework, 2018.)

Since hybrid applications use well-known HTML5, CSS and JavaScript, company can build the mobile application very fast and deploy in different mobile platforms. In addition, due to the popularity of hybrid applications, there is an increasing amount of free open-source hybrid frameworks to help developers to make the development easier and quicker. Some popular hybrid frameworks include: PhoneGap, Ionic framework, Sencha Touch 2 (Upwork, 2018). However, hybrid applications run slower and less powerful than native applications. They also have more bugs comparing to native applications.

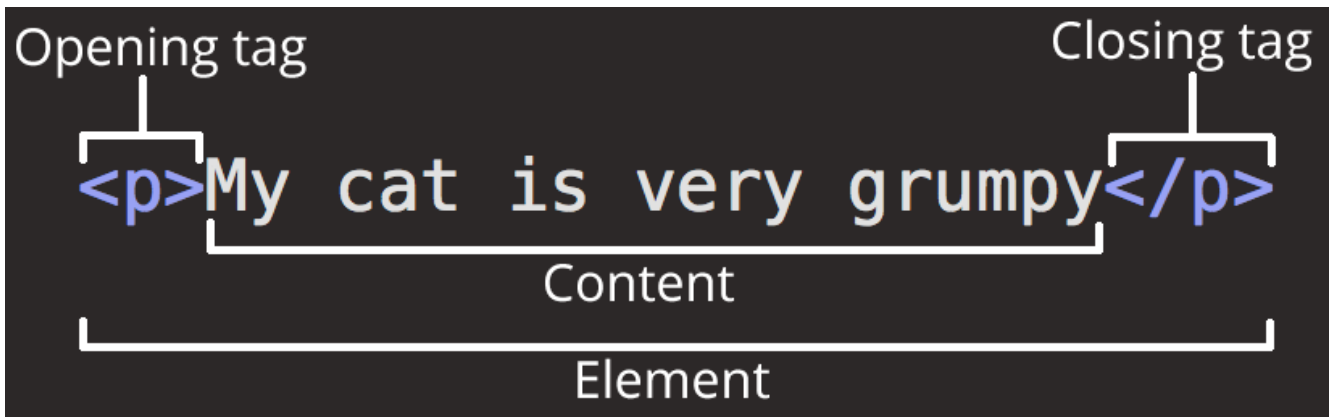
3 TECHNOLOGY STACK

When a company wants to build a website from scratch, they have to think what technology stack to use. The technology stack consists of all the programming languages, frameworks and tools used for the development. There are two main parts of any application, which are the client end and back end. The front end is what the user sees in the browser, while the back end focuses on servers and databases. Most of complex websites require both front and back end development to provide full features. (WebpageFX, 2018.)

This section explains the technology stack needed to build a hybrid application with web technologies. HTML, CSS and JavaScript are the three core web technologies. HTML structures the web, CSS styles the web and JavaScript makes the web interactive. In modern web development, there are many frameworks which make web development more efficient. Angular by Google and React by Facebook are the most popular JavaScript framework for front end development in the market today. Ionic is based on React, so it is easy for an Angular developer to start with it. (Medium, 2017.)

3.1 HTML

HTML stands for Hypertext Markup Language, which is the standard markup language to create websites. Web browsers use HTML to interpret text, images, videos and other content to web pages. HTML elements are the most basic building blocks of the web, which are used as HTML tags written using angle brackets. There are 3 main parts for element, which are opening tag, content and closing tag. The opening tag has the name of the element wrapped with angle brackets. Besides this, it can include attributes in the angle brackets. One of the most common attribute is class. For example, `<p class="comment">` is an opening tag. The content is between an opening tag and a closing tag. The closing tag is similar as the opening tag but it has a slash before the element name. Also, the closing tag does not have attributes. GRAPH 5 explains one simple HTML p element. (MDN, 2018.)

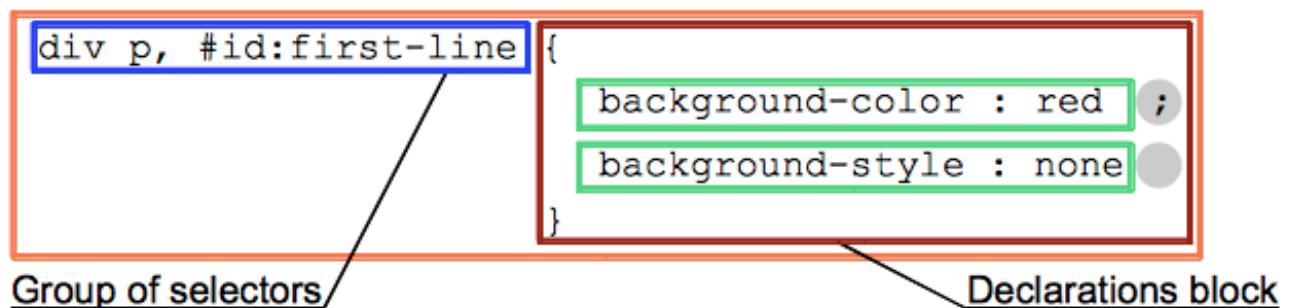


GRAPH 4. HTML element (MDN, 2018)

3.2 CSS

CSS stands for Cascading Style Sheets (CSS). It is a language to add style for HTML and XHTML page. CSS can alter font, colour, displace position, font size and picture style for the HTML elements. Browser turns HTML into a DOM when it opens an HTML file. CSS use selector to select HTML elements. CSS properties set styles to the selected HTML elements (MDN 2018). GRAPH 5 is a simple CSS ruleset example.

A CSS ruleset (or rule):



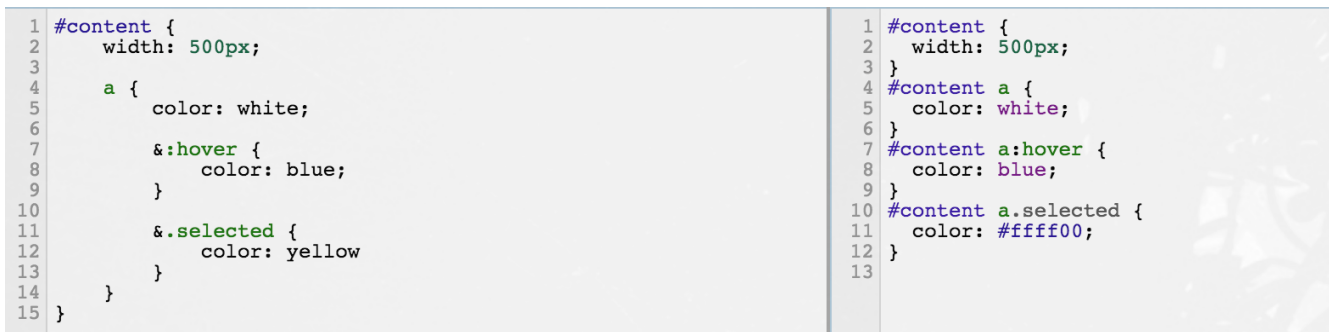
GRAPH 5. CSS ruleset (MDN 2018)

3.3 Less

Less stands for Leaner Style Sheets. It is designed by Alexis Sellier and first appeared in 2009. Less is a CSS pre-processor which extends the functionality of CSS. Less is open-source and written in JavaScript. Less extends CSS with variables, nesting, mixins, operators and functions. Because Less looks just

like normal CSS and valid CSS is also valid Less code, the learning curve is very low for a web developer. What makes Less different and powerful between other CSS pre-processors is that Less can be compiled via less.js by web browsers and run on both client and server side. (Lesscss, 2018.)

In Less, variables are defined with @ at the beginning of assignment. Variables work like constants and can only be defined once. For example, a developer needs to change all the red color to green. In the traditional way, the developer has to find all the red color and set them to green one by one. With Less, it is possible to create a variable and only one-time update is needed (Lesscss, 2018). Another useful feature of Less is nesting. It can make the code more concise and readable. GRAPH 7 is an example how nesting works.



<pre> 1 #content { 2 width: 500px; 3 4 a { 5 color: white; 6 7 &:hover { 8 color: blue; 9 } 10 11 &.selected { 12 color: yellow; 13 } 14 } 15 }</pre>	<pre> 1 #content { 2 width: 500px; 3 } 4 #content a { 5 color: white; 6 } 7 #content a:hover { 8 color: blue; 9 } 10 #content a.selected { 11 color: #ffff00; 12 } 13</pre>
---	---

GRAPH 6. Nesting in Less (left) and compiled to CSS (right).

3.4 JavaScript

JavaScript (abbreviated as JS) is the most-known scripting language for web pages. HTML, CSS and JavaScript are three core technologies of the web. JavaScript makes website interactive. JavaScript is not Java. It has nothing to do with Java, but like Java, JavaScript is a programming language and probably one of the most popular and widely used in the world. JavaScript works inside another web browser whether that is IE, Chrome, Safari, Firefox or Opera which has a JavaScript engine inside them. The operating system runs the web browser. The web browser contains pages and pages contain JavaScript. JavaScript does not have access to the file system of a computer. (MDN, 2018.)

JavaScript was only implemented the client side in the web browsers before the JavaScript run-time environment came out. Nowadays JavaScript engines are embedded in many other types of software, as a result JavaScript can be used on server side and even desktop and mobile applications. Node.js is such

an engine written by Ryan Dahl in 2009. It supports Linux, macOS and Windows. A package manager called npm was introduced for Node.js platform in 2010. It makes Node.js developers life much easier to publish and share source code of Node.js libraries. (Node.js foundation, 2018.)

3.5 Angular

Angular is an open-source popular JavaScript framework created by Google, which helps developers build modern applications quickly. Angular 2 was rewritten from AngularJS by the same team. Angular offers faster initial loads. Angular works with TypeScript and ES6. More than just a framework, Angular is actually a whole platform which comes with a collection of tools like Angular CLI, debugging and testing tools. (Angular, 2018.)

Angular apps are modular. Modules are the way how Angular organizes the application and works with other third-party libraries. Angular uses both JavaScript modules and Ngmodules. In JavaScript, modules are separate JavaScript files. The “import” statement is used to import JavaScript code from other module, and the “export” statement is used to export JavaScript code to be available in other JavaScript files. In Angular, NgModules are classed with decorator “@NgModule”. Every Angular application has at least root NgModule. The most common properties of a NgModule are declarations, exports, imports and providers. Declarations declare the classes belongs to the module. Exports are declarations that can be accessed in other modules. Imports includes other modules which are imported to this module. Providers are available in the whole app. (Angular, 2018.)

Angular component controls the logic on the page and the view of the application and on click event execution. It is a fundamental part of the application and it belongs to the controller class. Angular creates, updates and destroys components when the user browses through the application. @Component is used to register a component. @Component is a decorator function which contains component metadata. CSS styles can also be connected with a component with inline styles. The template is defined along with component, which is a HTML file that tells Angular where and how to render the component. Template not only uses standard HTML elements, but also Angular’s template syntax like *ngFor, (click). GRAPH 8 is the code snippet for the HTML template. (Angular, 2018.)

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>

```

GRAPH 7. Angular Template (Angular, 2018)

Metadata is mainly created to extend the class's functionality. In Typescript, this purpose is defined by using a decorator. Selector, templateUrl and providers are the most basic properties for a component decorator. Data binding is the connection between the model and the view. Angular supports four types of data binding - Property Binding, Event Binding, Interpolation and Two-Way Binding. (Angular, 2018.)

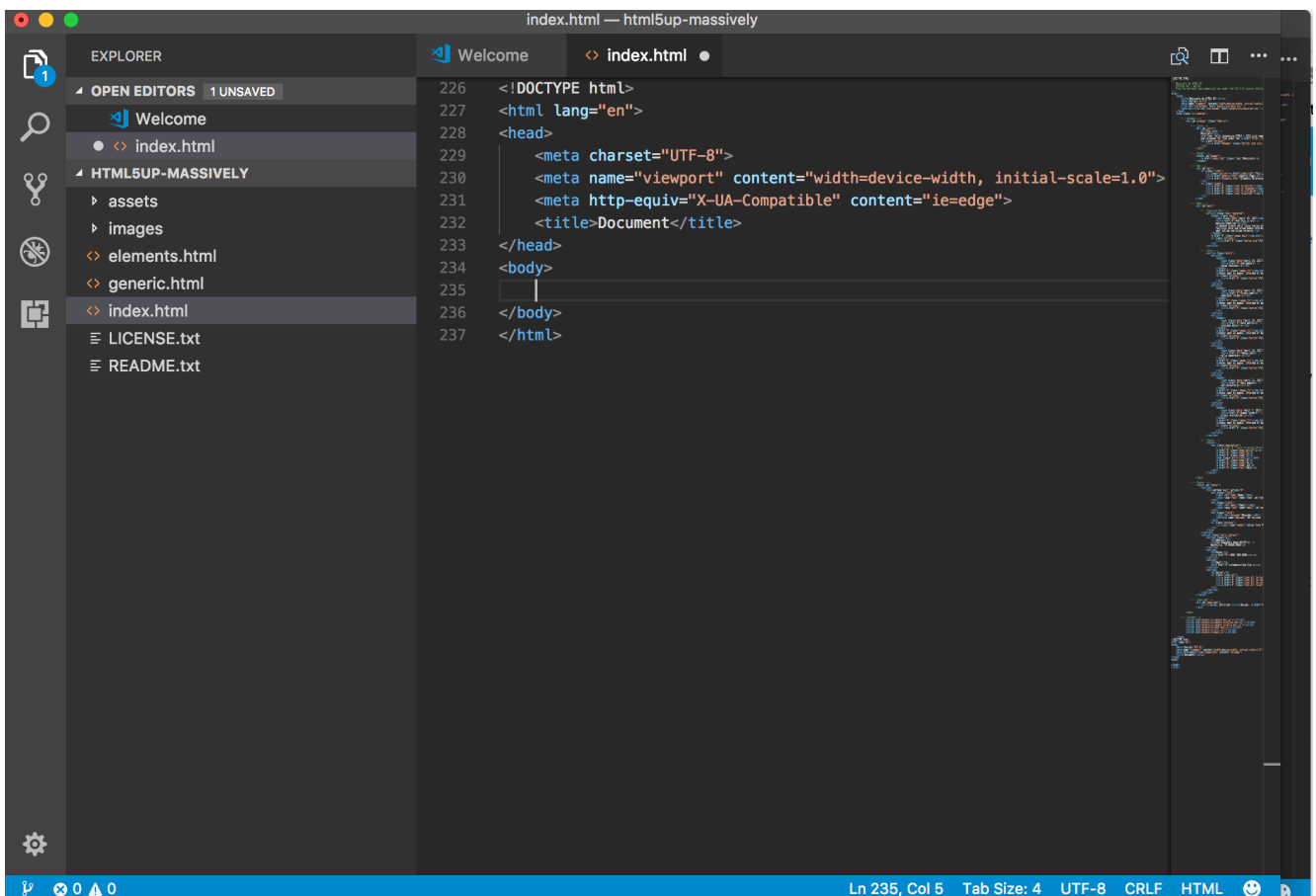
3.6 Ionic

Ionic is an open-source JavaScript framework to create hybrid mobile applications. It was released based on AngularJS and Apache Cordova. The newer version has migrated from AngularJS to Angular. Ionic app is created mainly through the ionic CLI (command line utility). CLI makes the development very fast and easy. Ionic CLI tool can install and update Ionic, generate a new page and run server. For example, simply typing "ionic start" will create a new project. (Ionic, 2018.)

Ionic components are a collection of UI elements that mimic the native look. Unlike in native application, Ionic components are built with HTML, CSS and JavaScript. Ionic components have different styles for different mobile platforms. With Ionic components, developer can quickly create the interface of the application. Ionic has a set of components, including action sheets, alerts, badges, buttons, cards and checkbox. (Ionic, 2018.)

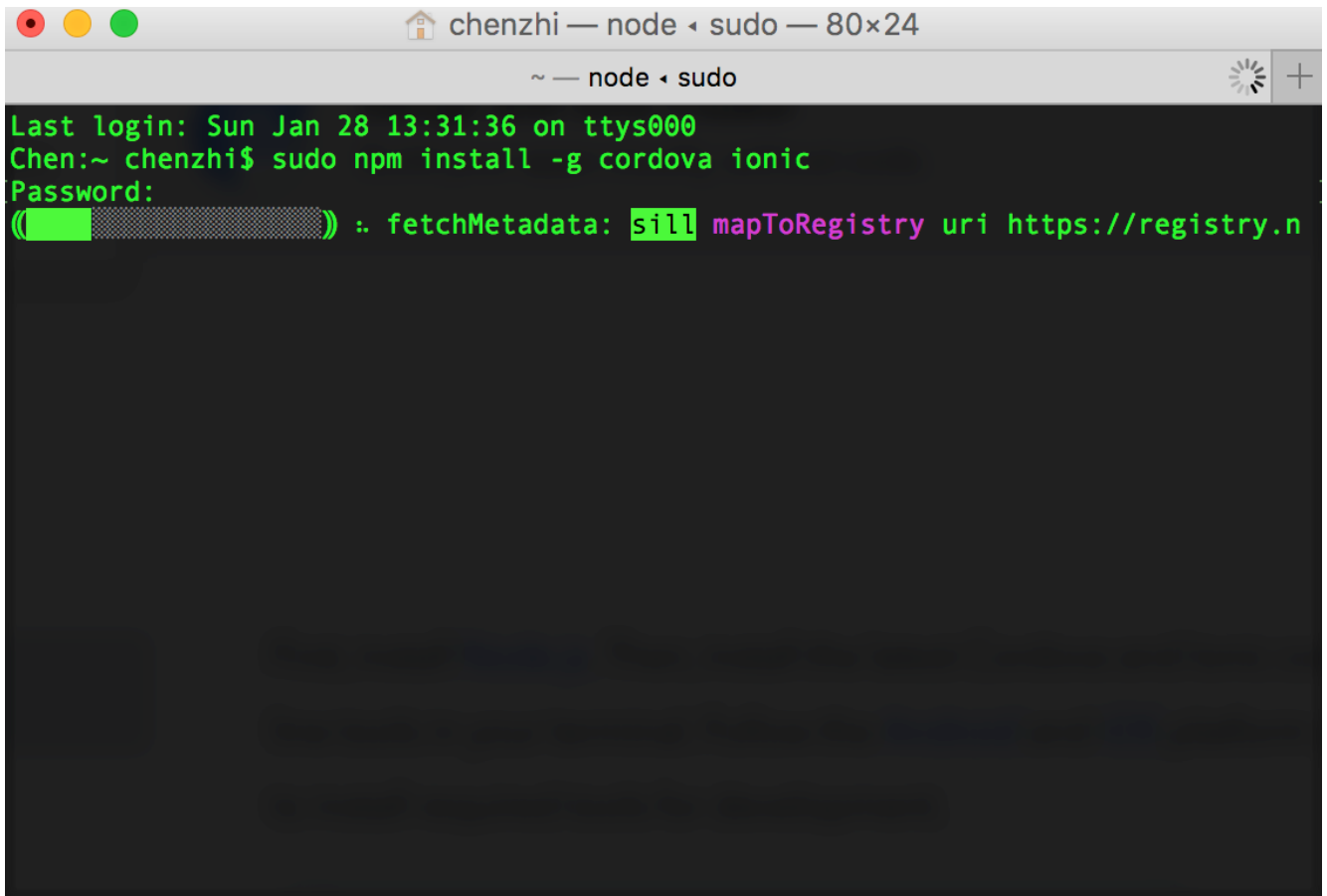
4 CASE STUDY: CREATE A LANGUAGE LEARNING APPLICATION WITH IONIC

Ionic was selected for the case study, because developers can build a hybrid application very easy and fast with it. Before coding, there are several tools which need to be installed. First tool is IDE. Visual Studio Code is an open-source code editor owned by Microsoft. It is available in Windows, Linux and macOS. It is lightweight but also powerful. It supports HTML, CSS, JavaScript, TypeScript and Python. Users can install additional extensions to add more languages, themes and debuggers. GRAPH 9 is the interface of Visual Studio.



GRAPH 8. Visual Studio Code in macOS

Then Node.js is needed since Ionic is based on Node.js run-time environment. NPM is a package manager for Node.js, and it is always downloaded with Node.js. First, download and install Node.js. Once it has been installed, run “sudo npm install -g cordova ionic ” in terminal to install the latest Cordova and Ionic CLI. GRAPH 10 is a screenshot of installation in Terminal.



```

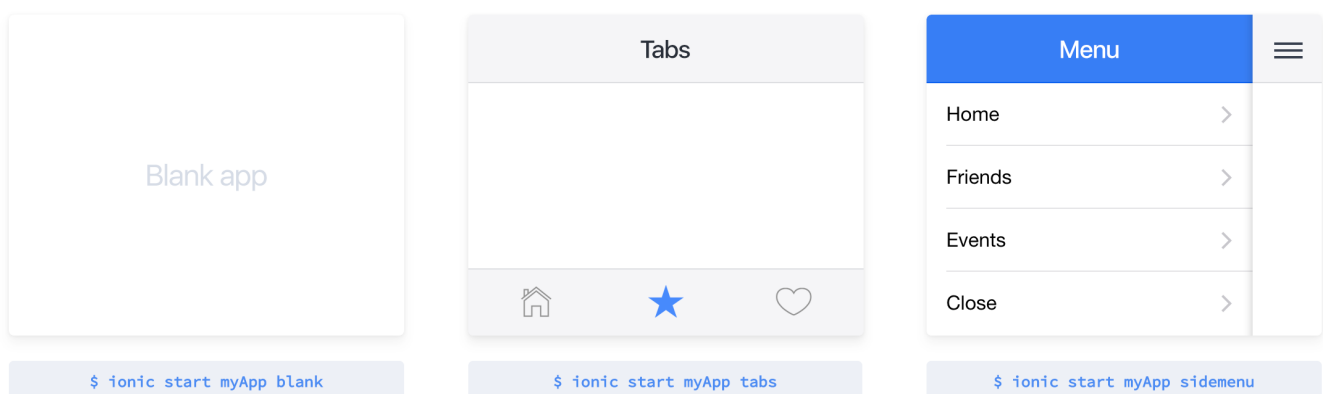
chenzhi — node sudo — 80x24
~ — node sudo
Last login: Sun Jan 28 13:31:36 on ttys000
Chen:~ chenzhi$ sudo npm install -g cordova ionic
Password:
:: fetchMetadata: sill mapToRegistry uri https://registry.n

```

GRAPH 9. Cordova and Ionic installation in terminal

4.1 Start an Ionic project

With Ionic CLI, developer can start a new application by running one command line in terminal. Use “ionic start myApp tabs” to create an app with tabs design (remember to replace myApp to your project name). GRAPH 11 illustrates three basic Ionic start templates by use the command “ionic start”.



GRAPH 10. Create an app with Ionic CLI

Also, there are some additional dependencies in the application. In general, there are two ways to install dependencies. One is to run “npm install dependency-name” in CMD, another is to edit the package.json file. The package.json file is core in Node.js environment. It must be valid JSON, not only JavaScript Object literal. The scripts property is npm scripts which can run in CMD. For example, if a developer has "clean": "ionic-app-scripts clean" in script field, type “npm run clean” in CMD will run “ionic-app-scripts clean” script automatically. The dependencies property is where other dependencies are defined. After updating package.json, run “npm install” to install all the dependencies for the project. The devDependencies are dependencies installed only in development environment, so they will not be installed in the production server. GRAPH 12 is the code snippet for dependencies and devdependencies in package.json file.

```
"dependencies": {
  "@angular/common": "4.1.2",
  "@angular/compiler": "4.1.2",
  "@angular/compiler-cli": "4.1.2",
  "@angular/core": "4.1.2",
  "@angular/forms": "4.1.2",
  "@angular/http": "4.1.2",
  "@angular/platform-browser": "4.1.2",
  "@angular/platform-browser-dynamic": "4.1.2",
  "@ionic-native/core": "3.10.2",
  "@ionic-native/http": "^4.3.0",
  "@ionic-native/splash-screen": "3.10.2",
  "@ionic-native/status-bar": "3.10.2",
  "@ionic/storage": "2.0.1",
  "angularfire2": "^4.0.0-rc.0",
  "cordova-plugin-advanced-http": "^1.5.10",
  "firebase": "^4.0.0",
  "ionic-angular": "3.3.0",
  "ionicons": "3.0.0",
  "rxjs": "5.1.1",
  "sw-toolbox": "3.6.0",
  "zone.js": "0.8.11"
},
"devDependencies": {
  "@ionic/app-scripts": "1.3.7",
  "@ionic/cli-plugin-ionic-angular": "1.3.1",
  "typescript": "2.3.3"
}
```

GRAPH 11. Code in package.json

4.2 Add Firebase Realtime Database

The Firebase Realtime Database is hosted in cloud. Data is stored in JSON format so data is synchronized in real time. Every time when user updates database, it is automatically updated in cloud. In this case, clients can access the same data no matter if they use iOS, Android or other JavaScript SDKs. To have connection of firebase database, authentication must be added in the project. Add the following code snippet to `src/app/app.module.ts` file:

```
const firebaseConfig = {
  apiKey: "AIzaSyDNVca_1bhyHL_-nvEs61eoTXchlSLnPPI",
  authDomain: "japanesecourse.firebaseio.com",
  databaseURL: "https://japanesecourse.firebaseio.com",
  projectId: "japanesecourse",
  storageBucket: "japanesecourse.appspot.com",
  messagingSenderId: "400567341682"
};
```

For good practice, Ionic Components should not fetch or save data directly. Ionic provider is a good way to share data. Ionic CLI provides a shortcut to generate a provider. Run “ionic generate provider firebase” to generate a provider. This command imports Injectable and Http by default. In order to connect with firebase, import AngularFireModule, AngularFireDatabase, FirebaseListObservable and FirebaseObjectObservable at the top of the code in `src/providers/firebase/firebase.ts` as shown below.

```
import { AngularFireModule } from 'angularfire2';
import { AngularFireDatabase, FirebaseListObservable, FirebaseObjectObservable } from 'angularfire2/database';
```

In the same file, inject AngularFireDatabase and AngularFireModule as parameters into the constructor of the class `FirebaseProvider`. Create `getcourseList`, `getLessonListByCourseId` and `getWholeDb` methods to get data from database. The code is shown below.

```
@Injectable()
export class FirebaseProvider {
  constructor(public db: AngularFireDatabase, public af: AngularFireModule) { }

  getCourseList() {
    return this.db.list('/courses/');
  }

  getLessonListByCourseId(courseId:number){
    var query = this.db.database.ref('courses').equalTo(1,'id').orderByChild('lessons');
    return this.db.list(query);
  }

  getWholeDb() {
    let result = this.db.object('/courses/').subscribe(result => {
      console.log('result');
      console.log(result);
    });
    return this.db.list('/');
  }
}
```

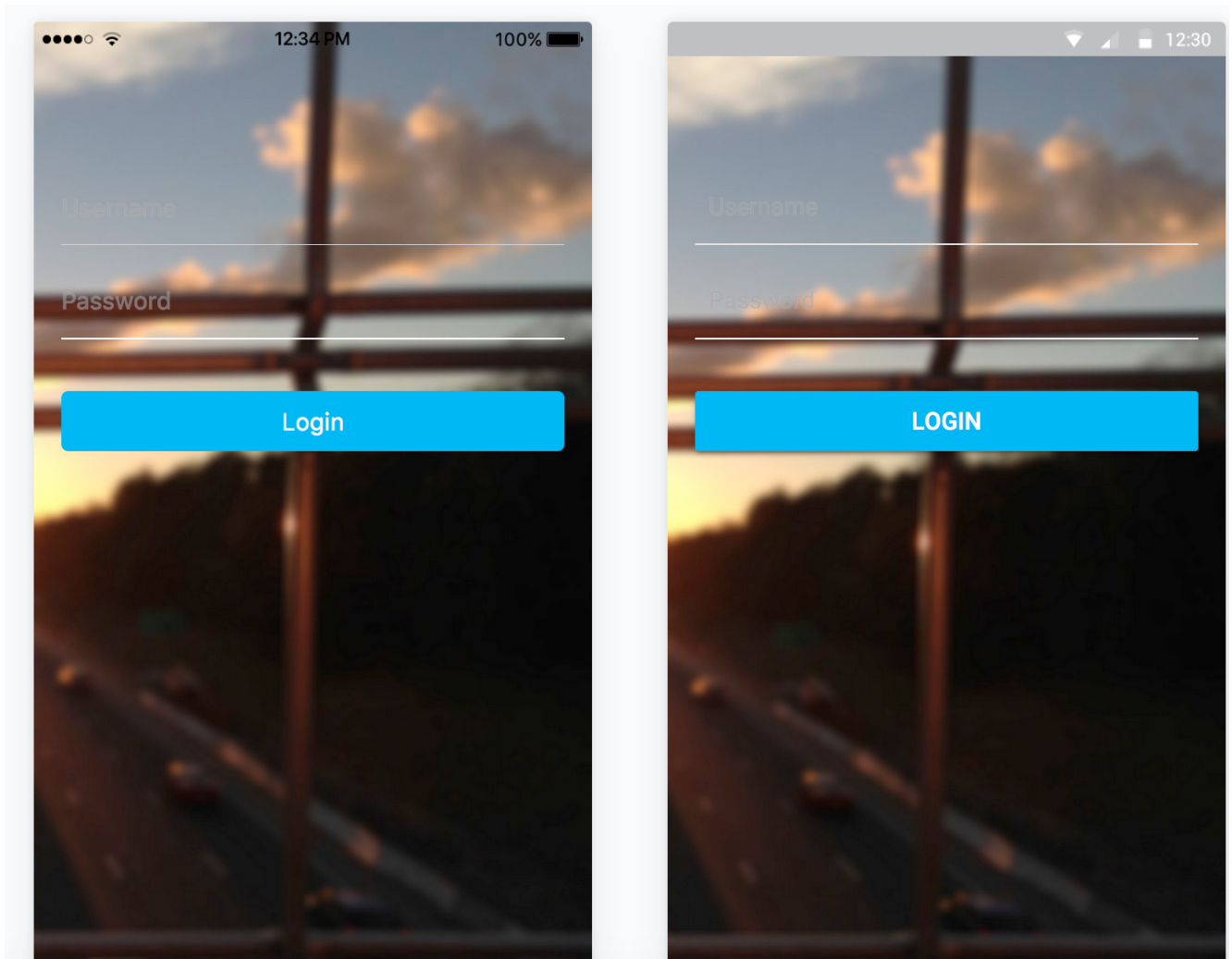
Firebase provider connects the app with some basic CRUD operations. However, to have connection with `courses.json` in Firebase database, a course provider is needed. Run “ionic generate provider course” to generate course provider. Replace all the code in `src/providers/course/course.ts` file as the following:

```
import { Injectable } from '@angular/core';
import { Http } from '@angular/http';
import 'rxjs/add/operator/map';
@Injectable()
export class CourseProvider {
  url;
  constructor(public http: Http) {
    console.log('Hello CourseProvider Provider');
    this.url = 'https://japanesecourse.firebaseio.com/courses.json';
  }
  getCourse() {
    return this.http.get(this.url)
      .map(res => res.json());
  }
}
```

4.3 Create a login page

Login page shows up when user opens the application. Ionic CLI can generate login page by running “ionic generate page login”. This command will create login folder inside of pages folder. Inside of courses folder, `login.html`, `login.scss`, `login.module.ts` and `login.ts` are created. Replace all the code in `login.html` file as in APPENDIX 1.

Ionic CLI provides a useful feature to review the app. Run “ionic serve” to see the application in a web browser. GRAPH 13 shows the login page in FireFox. Left side shows how the application looks like in iPhone, and right side shows how it looks like in Android phone.



GRAPH 12. Preview login page in a web browser

4.4 Create courses page

In the language course application, there is a page to show the list of courses. Ionic cards are used to display the list. In Ionic, cards are good way to display important information and design of cards is cross-platform. Update `src/pages/courses/courses.html` to display the list with Ionic cards components as the following code shown below.

```

<ion-header>
  <ion-navbar>
    <ion-title>
      Courses
    </ion-title>
  </ion-navbar>
</ion-header>
<ion-content>
  <ion-card (click)="navigate(course.id)" *ngFor="let course of courses">
    <!--  -->
    
    <ion-card-content>
      <ion-card-title>
        {{ course.name }}
      </ion-card-title>
      <p>
        {{ course.description }}
      </p>
    </ion-card-content>
  </ion-card>
</ion-content>

```

In the above code, {{ course.name }} and {{ course.description }} are Angular expressions written inside double braces. Like JavaScript expressions, Angular expression can contain literals, operators, and variables. Angular executes expressions every time when change is detected.

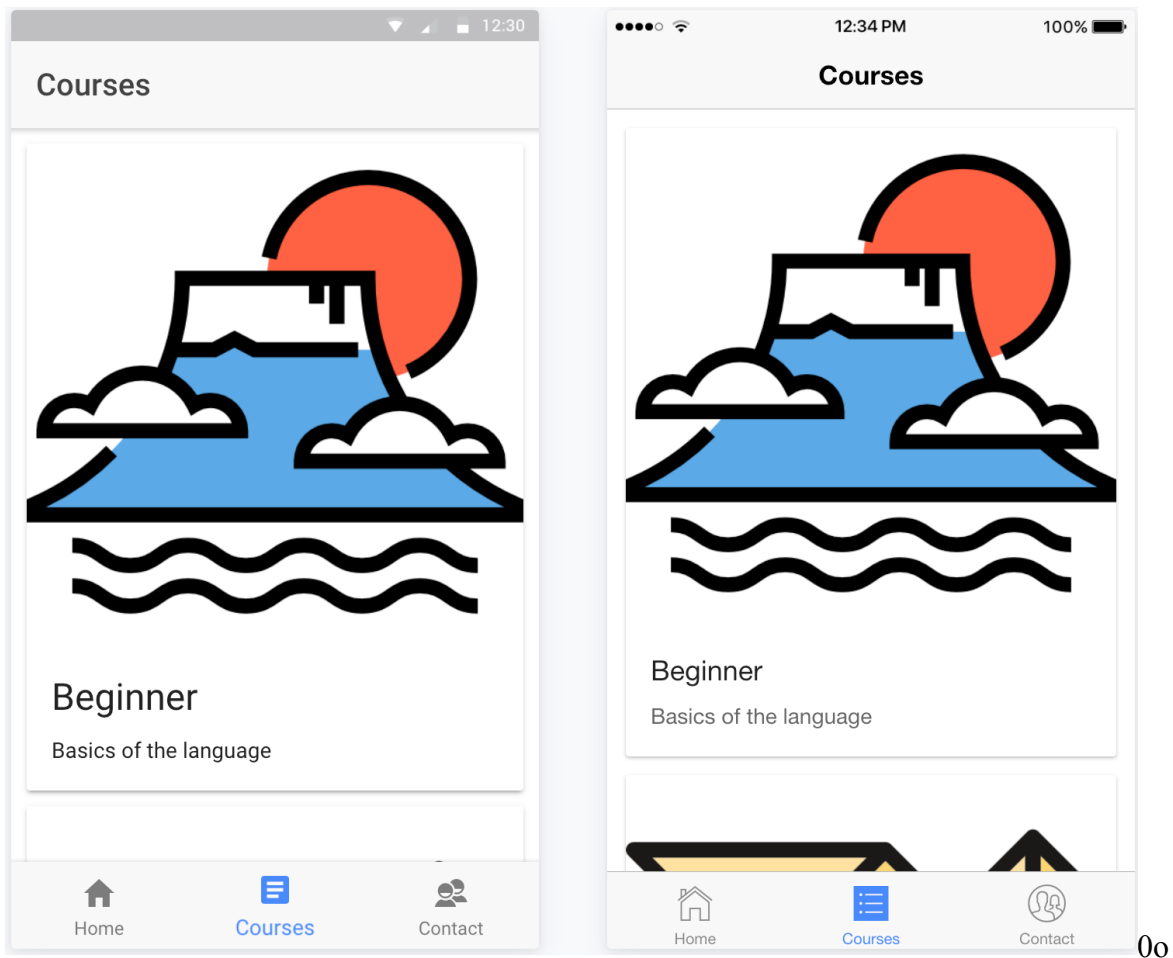
Controller of courses page controls how data is displayed in template. The courseProvider is injected in CoursePage class's constructor function to get the courses data from Firebase database. Then pass the courses data as class CoursePage's property, so template can have access to courses data. Update the code in courses.ts as the following shown below:

```

import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import { Lessons } from '../lessons/lessons';
import { CourseProvider } from '../../providers/course/course';
@Component({
  selector: 'page-course',
  templateUrl: 'courses.html'
})
export class CoursePage {
  courses;
  constructor(public navCtrl: NavController, public courseProvider: CourseProvider) {
  }
  ionViewWillEnter() {
    this.courseProvider.getCourse()
      .subscribe((courses => {
        this.courses = courses;
        console.log('this.courseList',this.courses);
      }));
  }
  public navigate(id: number) {
    this.navCtrl.push(Lessons);
  }
}

```

After updating the courses page, run “ionic serve” to see the courses page (shown in GRAPH 14). Now courses page has two cards on it. User can choose different level to start the course. The beginner card has a picture of Japanese famous Fuji mountain, and the intermediate card has a picture of origami. Both are the symbol of Japan, which is a good design for this Japanese language learning application.



GRAPH 13. Courses page

4.5 Create lesson list page

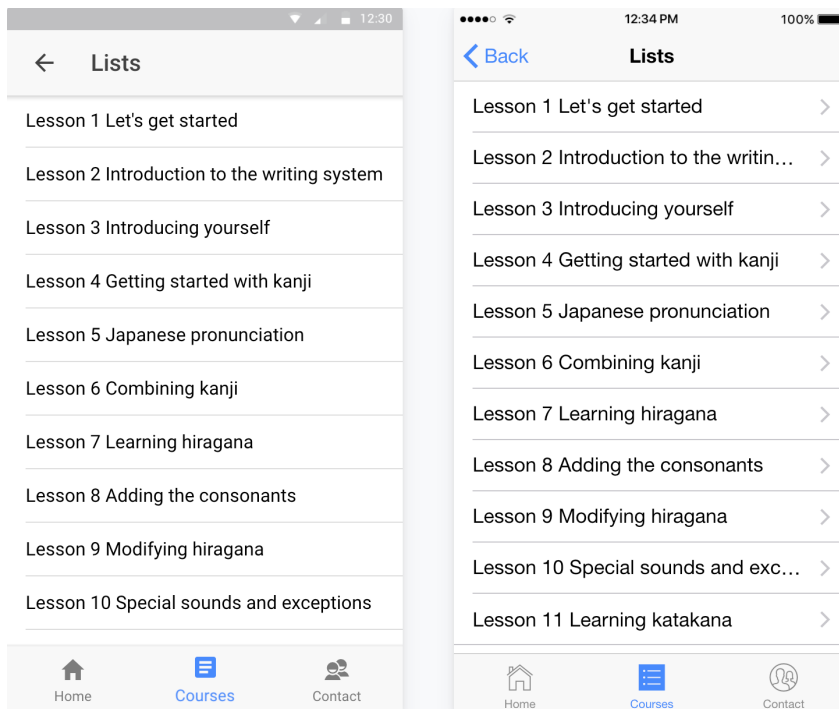
After user clicks a card in courses page, the application navigates to lesson list page. In the application, Ionic lists component is used to display the list of lesson. Update `src/pages/lessons/lessons.html` to display the list with Ionic list components as the following code:


```

<ion-header>
  <ion-navbar>
    <ion-title>Lists</ion-title>
  </ion-navbar>
</ion-header>
<ion-content>
  <ion-list>
    <button ion-item *ngFor="let item of items" (click)="itemSelected(item)">
      {{ item }}
    </button>
  </ion-list>
</ion-content>

```

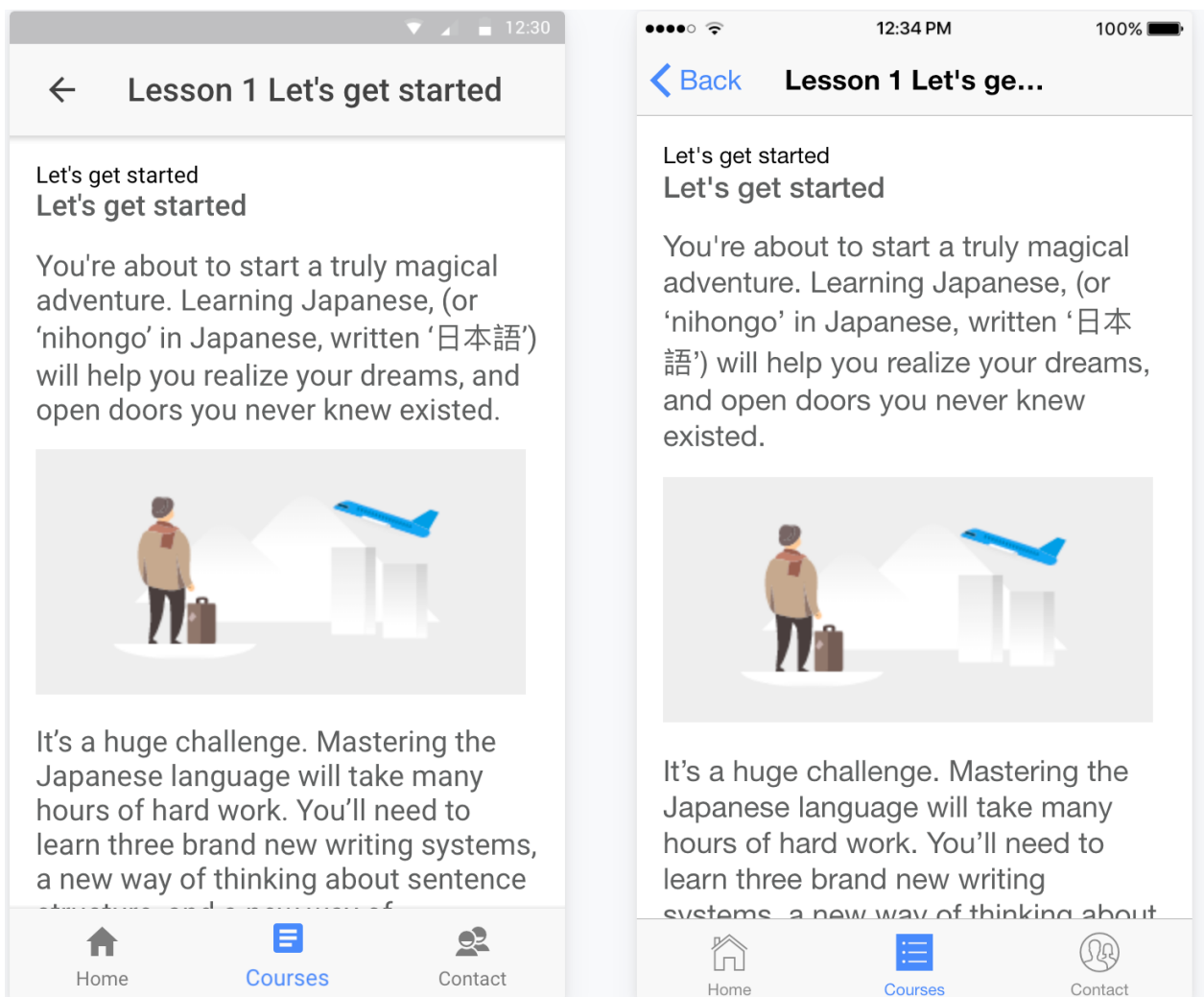
Angular ngFor directive is used to display each item in the lessons list in the template. In button element, (click)="itemSelected(item)" is to call itemSelected() function when the button is clicked. Now run “ionic serve” again to run the application in the web browser. Then go to courses page, and choose a course to see the lesson list page shown in GRAPH 15.



GRAPH 14. Lesson list page

4.6 Create lesson content page

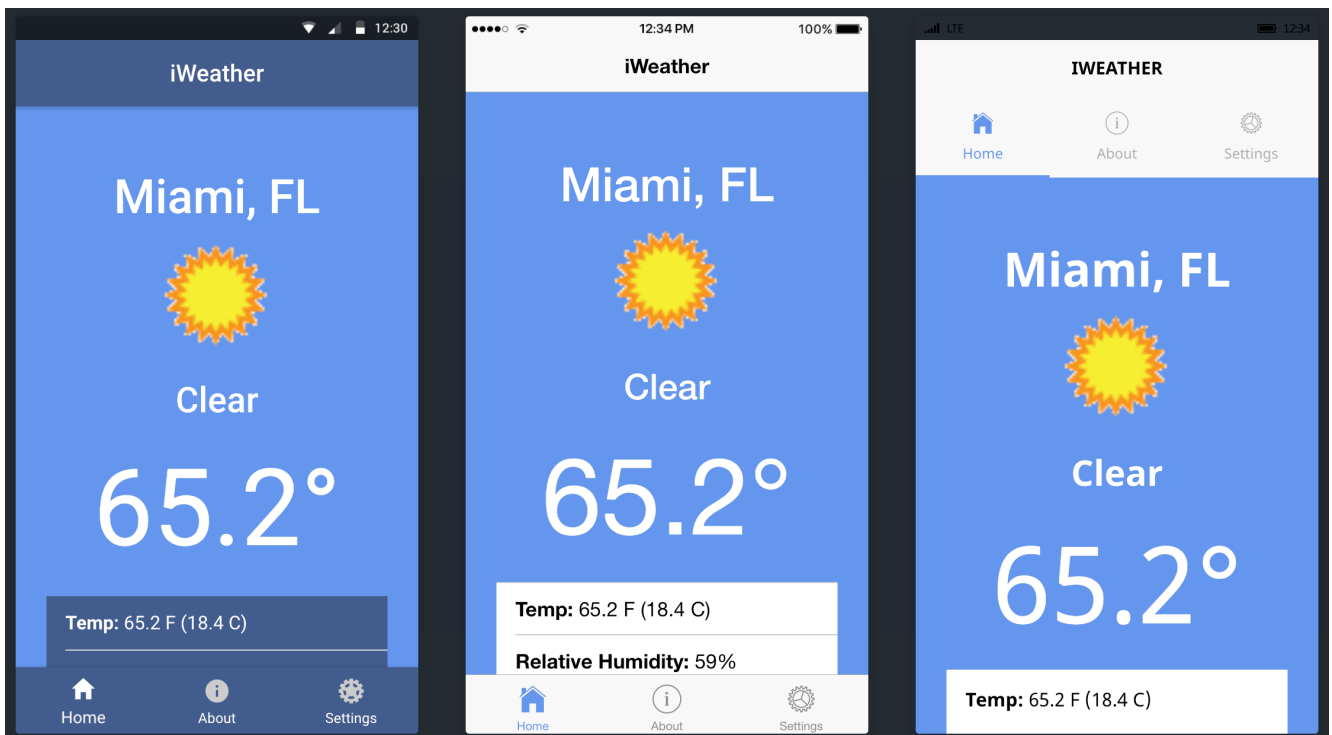
The content of a lesson is the core for the language learning application. Update `src/pages/lesson/lesson.html` to display lesson content. Then run “ionic serve” to preview lesson content page. GRAPH 16 shows the lesson content page.



GRAPH 15. Preview of lesson content page

5 CASE STUDY: CREATE A WEATHER APPLICATION WITH IONIC

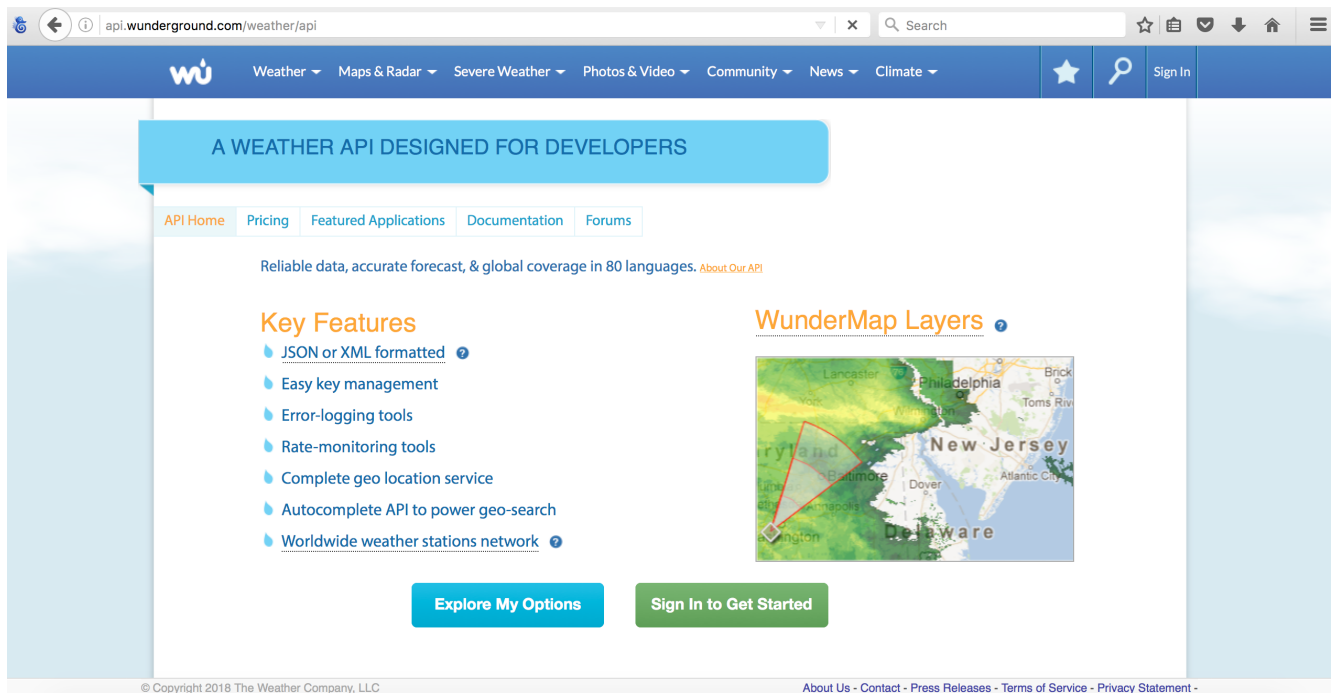
Thanks to WunderGround free weather API, making a weather application became very easy. With the same technology stack as the language application, developing an application demo is fast (WunderGround, 2018). GRAPH 17 shows how the weather application looks like in Android, iOS and Windows platforms (from left to right).



GRAPH 16. Preview at the weather application in different platforms

5.1 Weather Underground API

Weather Underground provides reliable data, accurate forecast and global coverage weather API in 80 languages for free (Weather Underground, 2018). Documentation of Weather Underground API is available at <http://api.wunderground.com/weather/api/d/docs>. GRAPH 18 shows the website of Weather Underground.



GRAPH 17. Weather Underground Weather API (Weather Underground, 2018)

Weather Underground API requires an API key which is free to get after signing up. API requests are made via HTTP. HTTP messages exchange data between a server and a client. In order to get the data, client sends a HTTP request to the server, then the server send back the answer in HTTP response. Here is an example HTTP request URL: <http://api.wunderground.com/api/9cc8101e058bc2fa/conditions/q/FL/Miami.json>. The following code uses Node.js to make HTTP request.

```

var http = require("http");
var options = {
  "method": "GET",
  "hostname": "api.wunderground.com",
  "port": null,
  "path": "/api/9cc8101e058bc2fa/conditions/q/FL/Miami.json",
  "headers": {
    "content-length": "0"
  }
};
var req = http.request(options, function (res) {
  var chunks = [];
  res.on("data", function (chunk) {
    chunks.push(chunk);
  });
  res.on("end", function () {
    var body = Buffer.concat(chunks);
    console.log(body.toString());
  });
});
req.end();

```

AJAX stands for Asynchronous JavaScript And XML, which is used in JavaScript to make HTTP. In order to make a HTTP request, an instance of XMLHttpRequest object needs to be created. After the ready state has been changed, process the data get from the server. Use following JavaScript code to make HTTP request using AJAX. APPENDIX 2 shows the HTTP response example from Wunderground. (MDN, AJAX, 2018.)

```

var xhr = new XMLHttpRequest();
xhr.addEventListener("readystatechange", function () {
  if (this.readyState === this.DONE) {
    console.log(this.responseText);
  }
});
xhr.open("GET", "http://api.wunderground.com/api/9cc8101e058bc2fa/conditions/q/FL/Miami.json");
xhr.send();

```

5.2 Build iWeather application

There are three pages in the application – Home, About and Setting. Home page is the main page to show the weather forecast. In the about page, there is basic information about the application. Type “ionic start iWeather tabs” to create iWeather project with built-in tabs template.

Provider is the place where to fetch data from Weather Underground API and connect with iWeather application. First, use “ionic generate provider weather” to generate a weather provider. Then copy and paste the following code shown below to weather.ts:

```

import { Injectable } from '@angular/core';
import { Http } from '@angular/http';
import 'rxjs/add/operator/map';
@Injectable()
export class WeatherProvider {
  apiKey = '9cc8101e058bc2fa';
  constructor(public http: Http) {
    this.url = 'http://api.wunderground.com/api/'+this.apiKey+'/conditions/q';
  }
  getWeather(city, state) {
    return this.http.get(this.url+'/'+state+'/'+city+'.json')
      .map(res => res.json());
  }
}

```

Home page is the most important part in the application because it is where to place the main content. Use “ionic generate page home” to generate a home page. There are two parts: the header and the content. In the header, iWeather is displayed as a title. In the content, the city, weather image, weather, temperature and other weather data are displayed. Copy and paste the code in APPENDIX 3 into home.html.

The home.html gives template layer. The file home.ts controls how data shown shows in template. As it is an Angular component, component needs to be imported from Angular core library. Use `@Component` decorator before the `HomePage` class and pass an object with selector and templateUrl information. In the `HomePage` class, give a type for weather, city and state. Then pass `NavController` and `WeatherProvider` as constructor’s arguments. Replace the following code at the top of the file home.ts:

```
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';
import { WeatherProvider } from '../providers/weather/weather';
@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {
  weather: any;
  location: {
    city: string,
    state: string
  }
  constructor(public navCtrl: NavController, private weatherProvider: WeatherProvider) {
  }
}
```

Every Ionic component can have lifecycle events, which are triggered during various stages of navigation (Ionic, Lifecycle events, 2018). Weather data is needed before the home page is loaded, so call the method `getWeather` from `weatherProvider` inside `ionViewWillEnter` event.

Finally, home page is ready in the application, but it still lacks some CSS styles. To make the application look more beautiful, put style rules in `app.scss` to apply globally. In `app.scss` file, style rules are not just

for one single component but the entire application. Also, it can be used to import other Sass files as an entry point in the project. Replace the code shown below to `app.scss` to give the application some CSS styles. Then run “ionic serve” to preview the application (shown in GRAPH 19).

```
.home{
  background:color($colors, primary);
  color:color($colors, light);
  margin-top:30px;
}
.toolbar-title{
  text-align: center;
}
.tabs-md .tab-button-icon{
  color:#ccc;
}
.temp{
  font-size:90px;
  text-align: center;
  font-weight: normal;
}
.location,.icon, .desc{
  text-align: center;
}
.icon img{
  width:100px;
}
.location{
  font-size:40px;
}
.desc{
  font-size:30px;
}
```




GRAPH 18. Before and after applying style rules

6 CONCLUSION

It is difficult to create a mobile application from scratch. Choosing the right technologies is always important and crucial at the stage of the planning a project. Companies need to choose which programming languages, frameworks, development tools, version control tools to work with at the beginning.

There are three types of mobile applications – native, web and hybrid. Native application has the best performance but it requires to create different application for different platforms. For example, and iOS apps are written in Objective-C or Swift and Android applications are written in Java. As to web application, it uses pure web technologies like HTML, CSS and JavaScript. User can access web application by opening a web browser and entering a URL. Hybrid application is in between the native and the web application. It uses the same web technologies as web application then wrapped into native application. However, users can download the hybrid application from an App Store and run it offline like a native application.

This thesis provides two study cases of developing hybrid application with Ionic. Ionic is easy to get started with. It only requires developers to have basic knowledge of HTML, CSS and JavaScript. With Ionic CLI, developer can easily start a project with built-in templates and components. It is a good option to choose hybrid application if the company is short of budget because the company only needs to build one code base and run it on every platform. Also, it is faster to build a hybrid application with modern JavaScript framework like Ionic.

As the web technologies are developing so fast nowadays, new frameworks come out every year and gain popularity quickly. Ionic with Angular, React Native with Reactjs and Week with Vuejs are now popular in the market. All of them can create hybrid applications with web technologies like HTML, CSS and JavaScript. Angular is considered to be easy to start with for developers with object-oriented programming experience while Reactjs uses both object-oriented programming functional programming. Vuejs is considered the lightest library among those three. However, it is very difficult for developers to fully understand all the new technologies when they just came out. Companies need to not only choose the new technologies but also consider if they are stable.

REFERENCES

Salesforce, 2016. Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options. Available at:

https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

Differencebetween, 2018. Difference between Native and Hybrid App. Available at:

<http://www.differencebetween.info/difference-between-native-and-hybrid-app>

W3schools 2018. HTML5 Browser Support. 2018. Available at:

https://www.w3schools.com/html/html5_browsers.asp

Upwork, 2018. Should You Build a Hybrid Mobile App? Available at:

<https://www.upwork.com/hiring/mobile/should-you-build-a-hybrid-mobile-app/>

Case Studies, Ionicframework, 2018. Available at:

<https://ionicframework.com/case-studies/MarketWatch.pdf>

WebpageFX 2018, A Guide to Technology Stacks. Available at:

<https://www.webpagefx.com/blog/web-design/technology-stacks-infographic/>

Medium 2018, Modern Web Development & Practices. Available at:

<https://medium.com/paramsingh-66174/modern-web-development-practices-64194d9a48>

Getting started with HTML, MDN, 2018. Available at:

https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started

W3schools, 2018. HTML Elements. Available at:

https://www.w3schools.com/html/html_elements.asp

MDN, 2018. CSS syntax. Available at:

https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Syntax

Lesscss, 2018. Less overview. Available at:

<http://lesscss.org/>

Node.js foundation, 2018. About The Node.js Foundation. Available at:

<https://foundation.nodejs.org/about>

Angular, 2018. Angular tutorial. Available at:

<https://angular.io/tutorial/toh-pt1>

Ionicframework 2018. The Top Open Source Framework for Building Amazing Mobile Apps. Available at:

<https://ionicframework.com/framework>

MDN, 2018. AJAX. Available at:

https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started

HTML file for login page

```

<style type="text/css">
  ::-webkit-input-placeholder {
    color: white !important;
  }
  :-moz-placeholder {
    color: white !important;
  }
  :-ms-input-placeholder {
    /* IE10+ */
    color: white !important;
  }
</style>

<ion-content padding class="login" style="background-image:url('https://unsplash.it/1242/2208/?blur&gravity=east');background-repeat: no-repeat;background-size: 100% 100%">
  <ion-grid>
    <ion-row>
      <ion-col col-4></ion-col>
      <ion-col col-4 style="margin-left: auto;margin-right: auto;display: block;">
        <!--  -->
      </ion-col>
    <ion-col col-4></ion-col>
  </ion-row>
</ion-grid>

<ion-list class="list-form" style="margin-top:30px;margin-bottom: 30px">
  <ion-item style="padding:0px !important; border-bottom:none !important;border-top:none;font-size: 14px;margin-top:10px;background:none;border-bottom:1px solid white;box-shadow:none;">

```

```

    <ion-input style="color:white;border-bottom-color:white;box-shadow:none;" type="text" place-
holder="Username" [(ngModel)]="username"></ion-input>
  </ion-item>
  <ion-item style="padding:0px !important; font-size: 14px;margin-top:10px;background:none;border-
der-bottom:1px solid white;box-shadow:none;">
    <ion-input style="color:white;border-bottom-color:white;box-shadow:none;" type="password"
placeholder="Password" [(ngModel)]="password"></ion-input>
  </ion-item>
</ion-list>
<button style="margin-top: 15px;height: 35px;font-size: 14px;background: #10ABF4;" ion-button
block (click)="login()">Login</button>
<p style="color: red;" *ngIf="!validateUser">Wrong username or password</p>
</ion-content>

```

HTTP response example

```

{
  "response": {
    "version": "0.1",
    "termsOfService": "http://www.wunderground.com/weather/api/d/terms.html",
    "features": {
    "conditions": 1
    }
  },
  "current_observation": {
    "image": {
      "url": "http://icons.wxug.com/graphics/wu2/logo_130x80.png",
      "title": "Weather Underground",
      "link": "http://www.wunderground.com"
    },
    "display_location": {
      "full": "Miami, FL",
      "city": "Miami",
      "state": "FL",
      "state_name": "Florida",
      "country": "US",
      "country_iso3166": "US",
      "zip": "33101",
      "magic": "1",
      "wmo": "99999",
      "latitude": "25.78000069",
      "longitude": "-80.19999695",
      "elevation": "10.1"
    },
    "observation_location": {
      "full": "Frost Museum of Science WeatherSTEM, Miami, Florida",
      "city": "Frost Museum of Science WeatherSTEM, Miami",
      "state": "Florida",

```

```

"country":"US",
"country_iso3166":"US",
"latitude":"25.785189",
"longitude":"-80.187874",
"elevation":"32 ft"
},
"estimated": {
},
"station_id":"KFLMIAMI361",
"observation_time":"Last Updated on January 30, 5:29 PM EST",
"observation_time_rfc822":"Tue, 30 Jan 2018 17:29:27 -0500",
"observation_epoch":"1517351367",
"local_time_rfc822":"Tue, 30 Jan 2018 17:30:28 -0500",
"local_epoch":"1517351428",
"local_tz_short":"EST",
"local_tz_long":"America/New_York",
"local_tz_offset":"-0500",
"weather":"Clear",
"temperature_string":"65.1 F (18.4 C)",
"temp_f":65.1,
"temp_c":18.4,
"relative_humidity":"60%",
"wind_string":"From the NNE at 5.0 MPH",
"wind_dir":"NNE",
"wind_degrees":31,
"wind_mph":5.0,
"wind_gust_mph":0,
"wind_kph":8.0,
"wind_gust_kph":0,
"pressure_mb":"1020",
"pressure_in":"30.14",
"pressure_trend":"+",
"dewpoint_string":"51 F (11 C)",

```



```

"dewpoint_f":51,
  "dewpoint_c":11,
  "heat_index_string":"NA",
  "heat_index_f":"NA",
  "heat_index_c":"NA",
  "windchill_string":"NA",
  "windchill_f":"NA",
  "windchill_c":"NA",
  "feelslike_string":"65.1 F (18.4 C)",
  "feelslike_f":"65.1",
  "feelslike_c":"18.4",
  "visibility_mi":"10.0",
  "visibility_km":"16.1",
  "solarradiation":"25",
  "UV":"0.0","precip_1hr_string":"0.00 in ( 0 mm)",
  "precip_1hr_in":"0.00",
  "precip_1hr_metric":" 0",
  "precip_today_string":"0.00 in (0 mm)",
  "precip_today_in":"0.00",
  "precip_today_metric":"0",
  "soil_temp_f": "73.0",
  "soil_moisture": "5.0",
  "leaf_wetness": "0.0",
  "icon":"clear",
  "icon_url":"http://icons.wxug.com/i/c/k/clear.gif",
  "forecast_url":"http://www.wunderground.com/US/FL/Miami.html",
  "history_url":"http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=KFLMI-
AMI361",
  "ob_url":"http://www.wunderground.com/cgi-bin/findweather/getForecast?query=25.785189,-
80.187874",
  "nowcast":""
}
}

```

HTML file for home page

```

<ion-header>
  <ion-navbar>
    <ion-title>iWeather</ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding class="home">
  <ion-grid *ngIf="weather">
    <ion-row>
      <ion-col width-50 offset-25>
        <h2 class="location">{{ weather.display_location.full }}</h2>
        <div class="icon"></div>
        <h3 class="desc">{{ weather.weather }}</h3>
        <h1 class="temp">{{ weather.temp_f }}&deg;</h1>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col width-100>
        <ion-list>
          <ion-item>
            <strong>Temp: </strong> {{ weather.temperature_string }}
          </ion-item>
          <ion-item>
            <strong>Relative Humidity: </strong> {{ weather.relative_humidity }}
          </ion-item>
          <ion-item>
            <strong>Dewpoint: </strong> {{ weather.dewpoint_string }}
          </ion-item>
          <ion-item>
            <strong>Visibility: </strong> {{ weather.visibility_mi }}
          </ion-item>
        </ion-list>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>

```

```
</ion-item>
<ion-item>
  <strong>Heat Index: </strong> {{weather.heat_index_string}}
</ion-item>
</ion-list>
</ion-col>
</ion-row>
</ion-grid>
</ion-content>
```