Ionic 2 Introduction

Lawrence Zhou

Mobile Softsmith Solutions Inc.
www.mobilesoftsmith.com

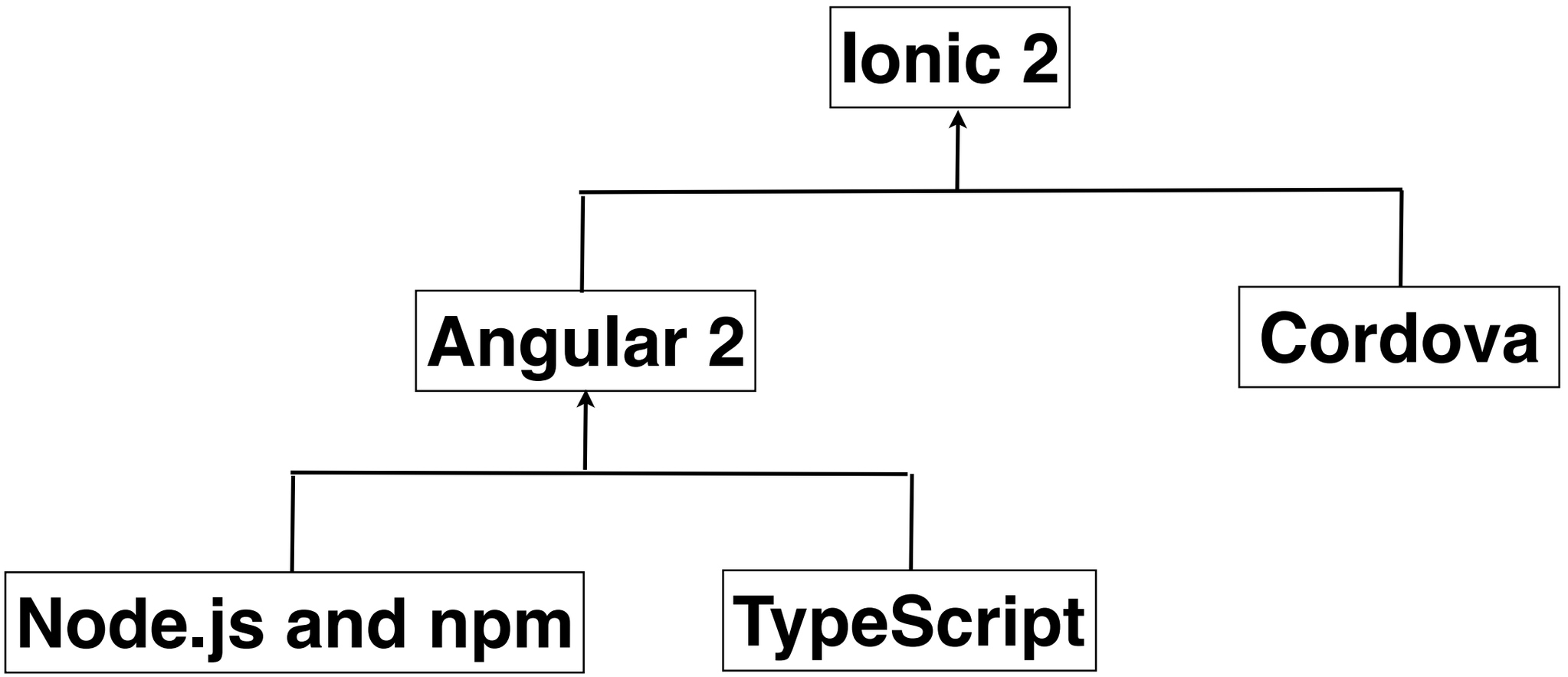
Main Content

- What is ionic 2?
- Setting up ionic 2
- Project structure
- Angular 2 Quick Review
- Rendering content
- Navigation between pages
- Theme an ionic 2 app
- Retrieving JSON data from remote server by HTTP
- Saving data in local storage or sqlite database

What is Ionic?

- Ionic is a complete open-source SDK for **hybrid** mobile app development, built on top of **AngularJS** and **Apache Cordova**.

- History
  1. Created by Drifty Co. in 2013
  2. 1.0 Beta in November 2013
  3. 1.0 final in May 2015
  4. 2.0 releases in 2016


Ionic 2
Angular 2
Cordova
Node.js and npm
TypeScript

Setting up ionic 2

- Install Node.js: https://nodejs.org/en/

- Install Ionic CLI(command line utility) and Cordova

  $ npm install -g ionic cordova

- Uninstall Ionic

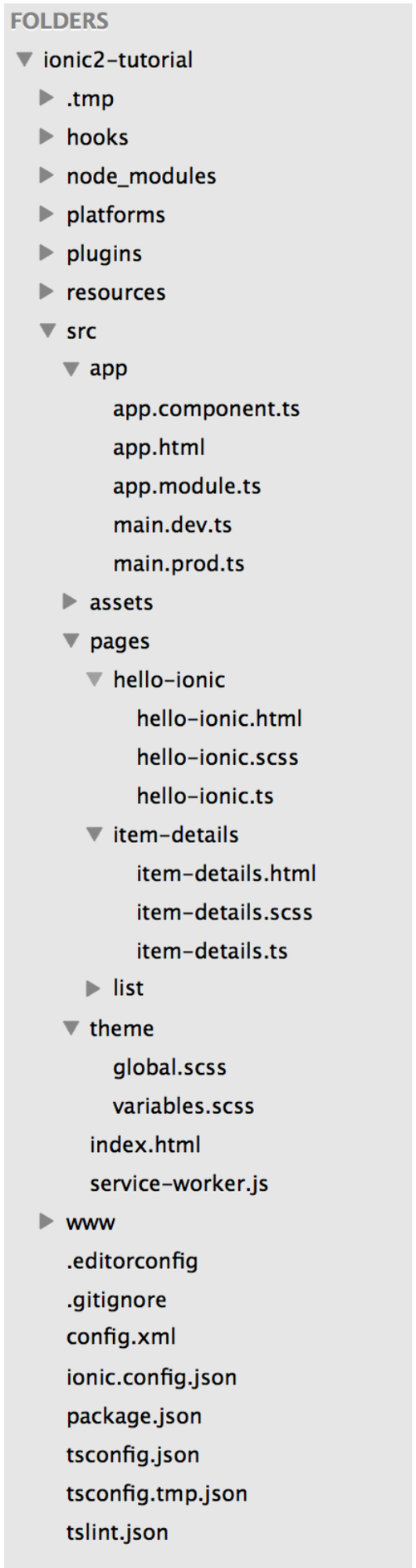
  $ npm uninstall -g cordova

  $ npm uninstall -g ionic

Creating App Project

$ ionic start {your-app} --v2 [tabs | sidemenu | blank | tutorial]

- tabs
- sidemenu
- blank
- tutorial



Project Structure

- Entry point
  src/index.html <ion-app></ion-app>

- Component
  High-level building blocks to make an ionic app

- Root component
  src/app/app.component.ts

- The app module and bootstrap module
  src/app/app.module.ts
  src/app/main.ts

- Pages
  src/pages/...

- Theme
  src/theme/variables.scss

Adding Platform and Run Your App

- Adding a platform

  $ ionic platform add { android | ios }

- Run in browser

  $ ionic serve [--lab]

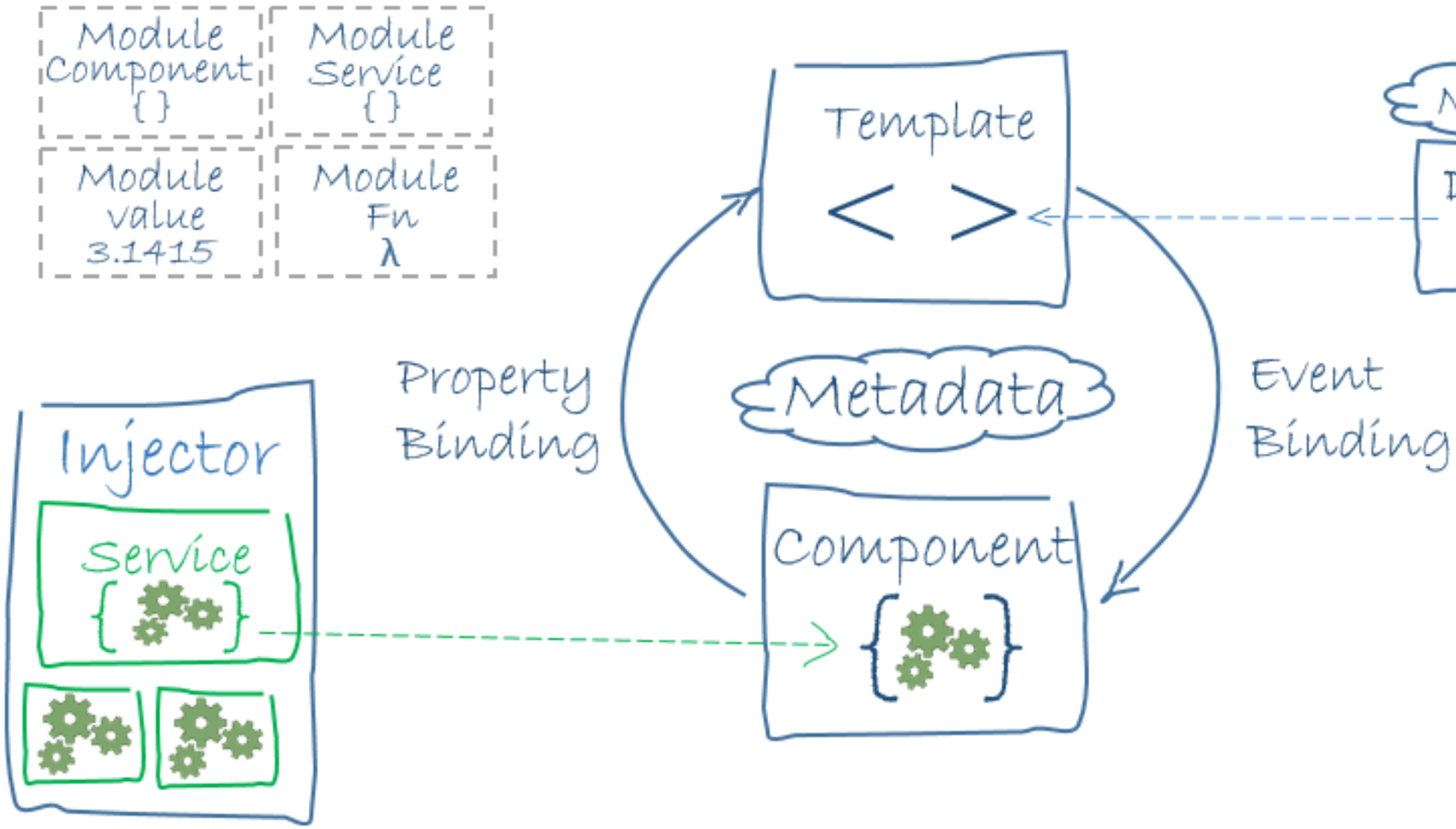
- Run in simulator

  $ ionic emulate { android | ios }

- Run in real device

  $ ionic run { android | ios --device }

Angular 2 Quick Review



- Templates, Components, and Modules (DI)

The Root Module

- Every application must have a root module to bootstrap to launch the application.

- Important @NgModule metadata properties:
    1. imports: Angualr Modules (only NgModule classes)
    2. declarations: declarables (components, directives and pipes)
    3. bootstrap: the component(s) will be created and insert into the DOM
    4. providers: global services
    5. entryComponents: specifies a list of components that should be compiled when this module is defined.
- Bootstrap in main.ts

- Binding a Property to a Value (Property Binding)

<input **[value]**="firstName">

- Calling a Function on an Event (Event Binding)

<button **(click)**="someFunction($event)">

- Rendering Expressions

Hi, **{{name}}**

- Two Way Data Binding

<input **[(ngmodel)]**="name">

- Creating a Variable to Access an Element

<p **#myParagraph**></p>

...

<button (click)="**myParagraph**.innerHTML = 'Once upon a time...'">

- Annotations

```
@Component({
  selector: 'my-component',
  services: [MyService]
})
```

Directives

- A directive is a view class with directive metadata.
- A component is a directive-with-a-template
- Structural directives (*ngFor, *ngIf)

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

- Attribute directives ([(ngModel)])

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

- Custom directives

Pipes

- Pipe is a view class changes displayed value in a template

- Pipes is usually used for formatting:

```
<p>The hero's birthday is {{ birthday | date }}</p>
```

Displaying content

- Simple UI topics
  1. Bindings
  2. Calling functions from view
  3. Alerts or popups
  4. Images

- Advanced UI topics
    1. Lists
    2. Sliders
    3. Custom Pipes (filter data format)
    4. Custom Directives (extra style decoration)
    5. Navigation between pages

    tip: add page command:

    $ ionic g page {page_name}

    A. Push and popup
    B. Set root
    C. Passing data between pages

Custom Pipes

- Create pipe class and register in src/app/app.module.ts

```
...
@NgModule({
  declarations: [
        ...
        CurrencyAnnotation,
        ...
```

- Import Pipe in pipe.ts

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

- @Pipe decorator

```
@Pipe ({
    name: 'currencyAnnotation'
})
export class CurrencyAnnotation {...}
```

- Override transform() function

```
...
transform(value, args) {
      if (args == 'cdn') {
          return 'CD $' + value;
       }else{
          return '$' + value;
       }
}
```

- Use the pipe in any page

The price is: {{price | **currencyAnnotation**: currency}}

price -> value

currency -> args

For example, if price = 20 and currency = 'cdn', then result will be:

The price is: **CD $20**

Custom Directives

- Create a new directive with CLI

$ ionic g directive HighLighter

- Import ElementRef service and inject it in constructor

```
import { Directive, ElementRef, Input } from '@angular/core';
@Directive({
  selector: '[high-lighter]' // Attribute selector
})
...
constructor(public el: ElementRef) {}
```

- Override 'ngOnInit()' as entrance to modify the element style

```
ngOnInit(){
    this.el.nativeElement.style.color = 'red';
    this.el.nativeElement.style.backgroundColor = 'yellow';
}
```

- Put the attribute selector to the tag you want to decorate

```
<h2 high-lighter>This a highlighted title</h2>
```

Theme

- Sass variables
- src/app/app.scss - main global scss
- src/pages/{page_name}/{page_name}.scss - component scss
- src/theme/variables.scss - override ionic variables here

- Ionic built-in SCSS/CSS utility attributes (e.g. text-left)

<ion-content **padding**>...</ion-content>

- Platform Specific Styles

customize your app for individual platform to fit the platform style

- Overriding Ionic Sass Variables

Ionic Sass variables can be overrided from src/theme/variables.scss file:

```
$text-color: #000099;

$colors(

  ...

)
```

Retrieving JSON data from remote server by HTTP

- Import Http object and JSON tool

```
import { Http } from '@angular/http';
import 'rxjs/add/operator/map';
```

- Call HTTP web service

```
...
 this.http.get('https://api.randomuser.me/?results=20')
    .map(res => res.json())
    .subscribe(data => {
        this.users = data.results;
    }, error => {
      console.log(error);
    });
...
```

- Display result

```
<ion-content padding>
  <ion-list>
    <ion-item *ngFor="let user of users" (click)="itemClicked($event,
user)">
        {{user.name.first}} {{user.name.last}}
    </ion-item>
  </ion-list>
</ion-content>
```

Saving data

- Local data storage
  1. Easy to use
  2. For simple data storage
  3. Size is up to 5M
  4. Not backup by cloud service

- SQLite storage
  1. Rely on Cordova plugin
  2. For complex data storage
  3. Size is unlimited
  4. Backup by cloud service

- Use local data storage
    1. Importing Storage service

        import { **Storage** } from '@ionic/storage';

    2. Registering in src/app/app.module.ts

        @NgModule({

        ...

            providers: [**Storage**]

        ...

        }

    3. Saving data

        constructor(public navCtrl: NavController, private localStorage: Storage) {

            this.localStorage.set('name', 'John Smith');

        }

    4. Retrieving data

        this.localStorage.get('name').then((data) => {

            this.name = data;

        });

- Use SQLite Database
  1. **Install SQLite Cordova plugin**

     $ ionic plugin add cordova-sqlite-storage
  2. **Import SQLite class**

     import { SQLite } from 'ionic-native';
  3. **Initialize the database**

```
this.sqliteStorage = new SQLite();
this.sqliteStorage.openDatabase({
  name: 'data.db',
  location: 'default'
}).then(() => {
  this.sqliteStorage.executeSql('CREATE TABLE IF NOT EXISTS users(name
VARCHAR(32))', {}).then(() => {
    console.log('Execute sql successfully');
  }, (err) => {
    console.error('Unable to execute sql: ', err);
  });
}, (err) => {
  console.error('Unable to open database: ', err);
});
```

4. Write data into database

```
this.sqliteStorage.openDatabase({
      name: 'data.db',
      location: 'default'
    }).then(() => {
      let query = "INSERT OR REPLACE INTO users VALUES (?)";
      this.sqliteStorage.executeSql(query, [name]).then(() => {
        console.log('Save data successfully');
    },(err) => {
      console.error('Unable to save data: ', err);
    });
}, (err) => {
    console.error('Unable to open database: ', err);
});
```

5. Retrieve data from database

```
this.sqliteStorage.openDatabase({
      name: 'data.db',
      location: 'default'
    }).then(() => {
      this.sqliteStorage.executeSql("SELECT * FROM users", []).then((data) => {
          console.log('search data...');
          this.sqliteData = [];
          if(data.rows.length > 0) {
              for(var i = 0; i < data.rows.length; i++) {
                  this.sqliteData.push({name: data.rows.item(i).name});
              }
          }
      }, (error) => {
          console.log("ERROR: " + JSON.stringify(error));
      });
}, (err) => {
    console.error('Unable to open database: ', err);
});
```

Summary and Conclusion

- What is ionic 2?
- Setting up ionic 2
- Project structure
- Angular 2 Quick Review
- Rendering content
- Navigation between pages
- Theme an ionic 2 app
- Retrieving JSON data from remote server by HTTP
- Saving data in local storage or sqlite database

- Advantages
  1. Support multiple platform with same code set
  2. Easy to create customized UI
  3. Easy to learn for web developers

- Disadvantages
  1. Hard to debug for complex app
  2. Heavily rely on plugin to access native resources
  3. Development environment are more complicated than native development. Some issues could delay project progress significantly.

- Conclusion

It's good for web developers to create simple logic app or prototype. It's ideal for the project with low budget and tight time frame to deploy on multiple platforms.

Resources

- Official docs: http://ionicframework.com/docs/
- Course github https://github.com/mobilesoftsmith/ionic-course