Hybrid apps are essentially small websites running in a browser shell in an app that have access to the native platform layer

# Reactive Native

Install android SDK

(<http://facebook.github.io/react-native/docs/android-setup.html#content>

Android SDK Build-tools version 23.0.1

Android 6.0 (API 23)

Android Support Repository)

Install npm > 4.0

>npm install –g react-native-cli

>react-native init AwesomeProject

(note: fails to connect to git://github.com

>git config –global url.<https://github.com/>.insteadOf [git@github.com](mailto:git@github.com)

>git config –global url.https://.insteadOf git://

Current progress requires several minutes)

# Ionic

Ionic comes with very native-styled mobile UI elements and layouts that you’d get with a native SDK on iOS or Android but didn’t really exist before on the web

Since Ionic is an HTML5 framework, it needs a native wrapper like Cordova or PhoneGap in order to run as a native app

Ionic apps aren’t meant to be run in a mobile browser app like Chrome or Safari, but rather the low-level browser shell like iOS’s UIWebView or Android’s WebView, which are wrapped by tools like Cordova/PhoneGap.

Eager developers might also dig down into the native layer with custom Cordova plugins or native code

Ionic targets iPhone and Android devices (currently). We support iOS 6+, and Android 4.0+ (though 2.3 should work).

//installation

**>npm install –g cordova ionic**

//project template

**Project Parent> ionic start myApp tabs**

(or ionic start myApp blank, ionic start myApp sidemenu)

//build via android or ios

**myApp> ionic platform add android**

**myApp> ionic build android**

//test

**myApp> ionic emulate android // testing in simulator**

(or substitute android with ios to build for ios)

Note: 若emulate失败，可以将myApp\platforms\android\build\outputs\apk\android-debug.apk拷到mobile device安装测试

**myApp> ionic serve //testing in a browser**

//publish

myApp> cordova plugin rm cordova-plugin-console // disenable the debug console plugin

//generate our private key

>keytool -genkey -v -keystore **my-release-key.keystore** -alias alias\_name -keyalg RSA -keysize 2048 -validity 10000

//sign the unsigned APK

>jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore **my-release-key.keystore** **HelloWorld-release-unsigned.apk** alias\_name

//optimize the APK

>zipalign -v 4 HelloWorld-release-unsigned.apk HelloWorld.apk

Yeoman: node必须是最新版

>npm install –g generator-ionic

(如果有错误，类似operaton not permitted, rename ‘…\npm-cache\...’, 要先清空

>npm cache clean)

>mkdir my-ionic-project && cd $\_

my-ionic-project>yo ionic

Ionic

Ionic is a bunch of UI elements made in HTML5 & CSS3 that covers a lot of the mobile interactions

The big advantage of Ionic is all the UI components are AngularJS Directives

• Built with Sass and optimized for AngularJS

• Beautifully designed

• Extends the HTML vocabulary

• UI Components using Directives and Services

• Proven for large-scale app development

• Ionicons (over 700 MIT licensed font-icons)

• Supported by Drifty and has a large community:

• Very active internal forum

导航（路由：AnguarUI Router）

Like Angular’s core $route service, URLs can be used to control the views. However, the AngularUI Router provides a more powerful state manager in that states are bound to named, nested, and parallel views, allowing more than one template to be rendered on the same page. Additionally, **each state is not required to be bound to a URL**, and data can be pushed to each state which allows much flexibility.

By default, views are cached to improve performance. When a view is navigated away from, its element is left in the DOM, and its scope is disconnected from the $watch cycle. When navigating to a view that is already cached, its scope is then reconnected, and the existing element that was left in the DOM becomes the active view.

By default, Ionic will cache a maximum of 10 views, Cached can be disabled, or the maximum number of cached views changed in $ionicConfigProvider

By default, when navigating back in the history, the “forward” views are removed from the cache. If you navigate forward to the same view again, it’ll create a new DOM element and controller instance. Basically, any forward views are reset each time

Directive: **ionNavView**

Ionic is able to keep track of their navigation history. By knowing their history, transitions between views correctly enter and exit using the platform’s transition style.

<ion-nav-view></ion-nav-view>

Directive: **ionView**

A container for view content and any navigational and header bar information. Views can be cached, which means controllers normally only load once, which may affect your controller logic. To know when a view has entered or left, events have been added that are emitted from the view’s scope. These events also contain data about the view, such as the title and whether the back button should show. Also contained is transition data, such as the transition type and direction that will be or was used.

Directive: **ionNavBar**

If we have an ionNavView directive, we can also create an <ion-nav-bar>, which will create a topbar that updates as the application state changes.

Note that the ion-nav-bar element will only work correctly if your content has an ionView around it.

Directive: **ionNavBackButton**

The back button will appear when the user is able to go back in the current navigation stack Additionally, the button is automatically set to $ionicGoBack() on click/tap. By default, the app will navigate back one view when the back button is clicked

Directive: **ionNavButtons**

Any buttons you declare will be positioned on the navbar’s corresponding side. Primary buttons generally map to the left side of the header, and secondary buttons are generally on the right side. However, their exact locations are platform-specific. For example, in iOS, the primary buttons are on the far left of the header, and secondary buttons are on the far right, with the header title centered between them. For Android, however, both groups of buttons are on the far right of the header, with the header title aligned left.

Note that ion-nav-buttons must be immediate descendants of the ion-view or ion-nav-bar element (basically, don’t wrap it in another div).

Directive: **ionNavTitle**

The nav title directive replaces an ionNavBar title text with custom HTML from within an ionView template. This gives each view the ability to specify its own custom title element, such as an image or any HTML, rather than being text-only. Alternatively, text-only titles can be updated using the

view-title ionView attribute.

Note that ion-nav-title must be an immediate descendant of the ion-view or

ion-nav-bar element (basically don’t wrap it in another div).

**$ionicNavBarDelegate**

Delegate for controlling the ionNavBar directive.

$ionicNavBarDelegate.align([direction]): The direction to the align the title text towards. Available: 'left', 'right', 'center'. Default: 'center'.

$ionicNavBarDelegate.showBackButton([show]): Set/get whether the ionNavBackButton is shown (if it exists and there is a previous view that can be navigated to).

$ionicNavBarDelegate.showBar(show): Set/get whether the ionNavBar is shown.

$ionicNavBarDelegate.title(title): Set the title for the ionNavBar.

**$ionicHistory**

$ionicHistory keeps track of views as the user navigates through an app. Similar to the way a browser behaves, an Ionic app is able to keep track of the previous view, the current view, and the forward view (if there is one). However, a typical web browser only keeps track of one history stack in a linear fashion.

Unlike a traditional browser environment, apps and webapps have parallel independent histories,

such as with tabs. Should a user navigate few pages deep on one tab, and then switch to a new

tab and back, the back button relates not to the previous tab, but to the previous pages

visited within that tab.

$ionicHistory facilitates this parallel history architecture.

$ionicHistory.viewHistory()

$ionicHistory.currentView()

$ionicHistory.currentHistoryId()

$ionicHistory.currentTitle([val])

$ionicHistory.backView()

$ionicHistory.backTitle()

$ionicHistory.forwardView()

$ionicHistory.currentStateName()

$ionicHistory.goBack([backCount])

$ionicHistory.clearHistory()

$ionicHistory.clearCache()

$ionicHistory.nextViewOptions()

Example:

<body ng-app="starter">

<!-- The nav bar that will be updated as we navigate -->

<ion-nav-bar class="bar-positive">

<ion-nav-back-button>

</ion-nav-back-button>

</ion-nav-bar>

<!-- where the initial view template will be rendered -->

<ion-nav-view>

<ion-view>

<ion-nav-buttons side="primary">

<button class="button" ng-click="doSomething()">primary</button>

</ion-nav-buttons>

<ion-nav-title><img src="logo.svg"></ion-nav-title>

<ion-content>

Hello!

</ion-content>

</ion-view>

</ion-nav-view>

</body>

Hybrid Apps

• HTML 5 that acts like native

• Web wrapper in native layer

• Direct access to native APIs

• A single code base

• Familiar web development environment

A lot of components

Swipeable List Options

• Side menus

• Actionsheets

• Tabs

• Pull to Refresh

• Slidebox

• Infinite Scroll

• Popup

• Popover

• Loading Overlay

• Inputs

• Buttons

Cached Views

View elements left in DOM

• $scope disconnected from cache

• State maintained

• Scroll position maintained

• Life Cycle events

• Highly configurable

Collection-Repeat

Replacement for ng-repeat

• Scroll through thousands of items

• Only renders the viewable items

• Smooth scrolling

Ionic-Cli

• Testing in a browser

• Live Reload App During Development

• Emulating your app

• Running your app on device

• Building your app (with or without SDK)

• Icon and Splash Screen Image Generation

• Crosswalk for Android

ngCordova: CORDOVA WITH THE POWER OF ANGULARJS

ngCordova is a collection of 63+ AngularJS extensions on

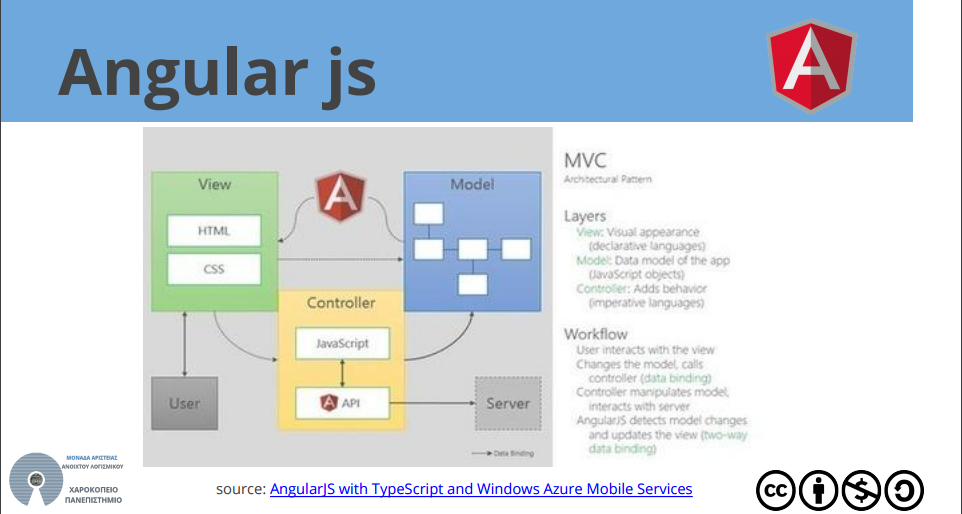
top of the Cordova API that make it easy to build, test, and

deploy Cordova mobile apps with AngularJS.

Apache Cordova

● Set of device APIs that allow a mobile app

developer to access native device function such as the camera or accelerometer from JavaScript.



# PhoneGap

Apache Cordova is the engine that powers Adobe PhoneGap™, similar to how WebKit powers Chrome or Safari. However, Adobe PhoneGap™ provides additional tools that tie into other Adobe services, including tools like the PhoneGap Developer App, the PhoneGap Desktop App, PhoneGap Build and PhoneGap Enterprise.

Install PhoneGap desktop app ->computer

Install PhoneGap Developer App ->mobile device

是在移动设备上调试 PhoneGap 应用的工具，不需要每次打包、安装就可以直接在手机上调试 PhoneGap 应用

PhoneGap Desktop starts a small web server to host your project and returns a server address you can then enter into the PhoneGap Developer app running on your mobile device.