IONIC 4

WEB NATIVE

I O N I C 4

Web Native applications made right with Ionic 4

Ionic 4 - Web Native

The Web is the new Native

Cristian Olaru

This book is for sale at http://leanpub.com/ionic4

This version was published on 2018-08-23

Also By Cristian Olaru

Grails 3 - Step By Step

Contents

How this book is organized

We have split this book in two parts.

Ionic 4 Essentials

Chapter 1 - Introduction to Ionic 4

Chapter 2 - The app implemented in this book

Chapter 3 - The MVP

Ionic 4 Advanced

Chapter 4 - Advanced mobile development

Chapter 4 - Advanced integration with Firebase

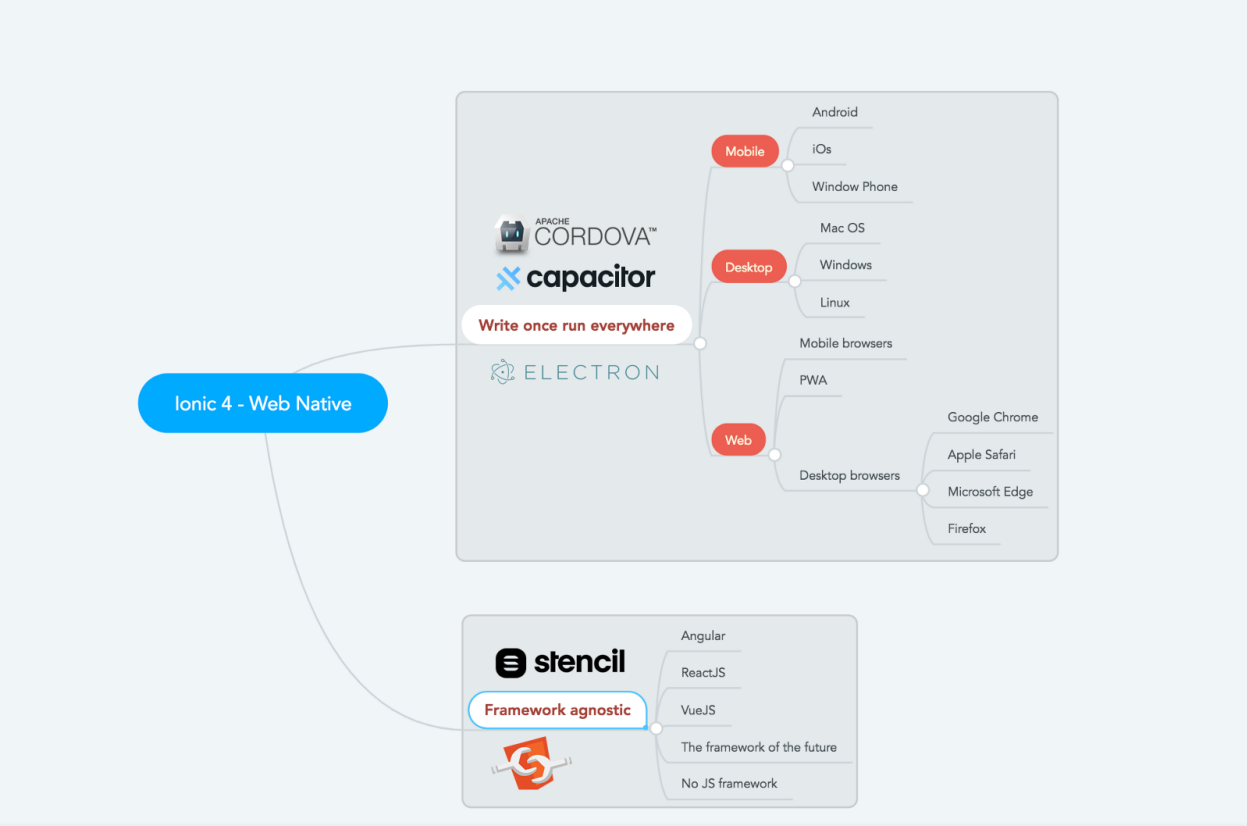
Chapter 5 - Notifications with OneSignal

Introduction to Ionic 4

Introduction to the book

First of all, thank you for buying this book dedicated to Ionic framework[1] at its 4th version.

> This book is a work in progress created using the Lean Pub[2] self-publishing system. You can buy a copy of it at a small price when it is still not finished, and you will get the updated chapters after at the initial price. As the time is passing and new chapters are added, the price can increase for the new buyers.



Ionic 4 Vision

We try in this book to implement an application according to the new Ionic 4 vision to **write once run everywhere**. So we have to write one Web application and we can host it as a static Web app

[1] https://ionicframework.com
[2] https://leanpub.com/ionic4

(PWA), package it and publish it to the mobile app stores, or package it as a desktop application and let the users download and install it.

Write once run everywhere

This book isn't a tutorial of Ionic 4 - you have the Ionic 4 documentation site[3] for this.

Web will be the new native

- There is still a big investment made by companies in Web applications compared to desktop/-mobile applications
- The browsers tend to offer more and more APIs: PWA (Web/Service Workers, Web App Manifest), WebRTC, Web Assembly, Web storage, Web sockets, Web components, Canvas
- The modern browsers are more compatible: Chrome, Safari, Firefox, Edge
- The Web JS frameworks are involving and, is simple now to have just static sites created with Angular, React or Vue JS using web services to communicate with the backends
- The Web services APIs can be easily be created with REST/JSON and well-defined with Swagger and OpenAPI and secured with OAuth
- More serverless cloud solutions for backends, cheaper and easy to be scaled like Firebase, AWS
- Now we can write once a Web application and package it to different platforms with solutions like Cordova, Capacitator, Electron

[3] https://beta.ionicframework.com/docs/

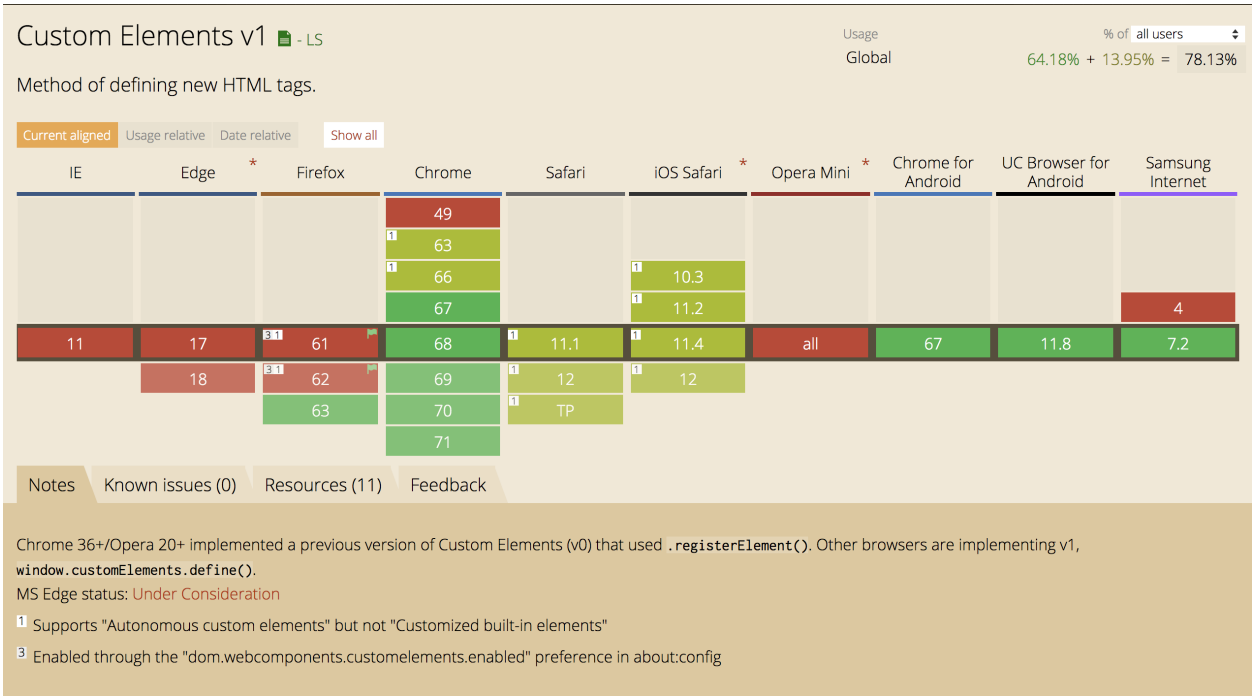
What is Ionic 4

Ionic 4 in its essence is a set of standard Web Components that are rendered according to the design guidelines Materila Design[4] for Web and Android devices and Human Interface[5] for iOs devices. These web components are created by Ionic team using Stencil JS.

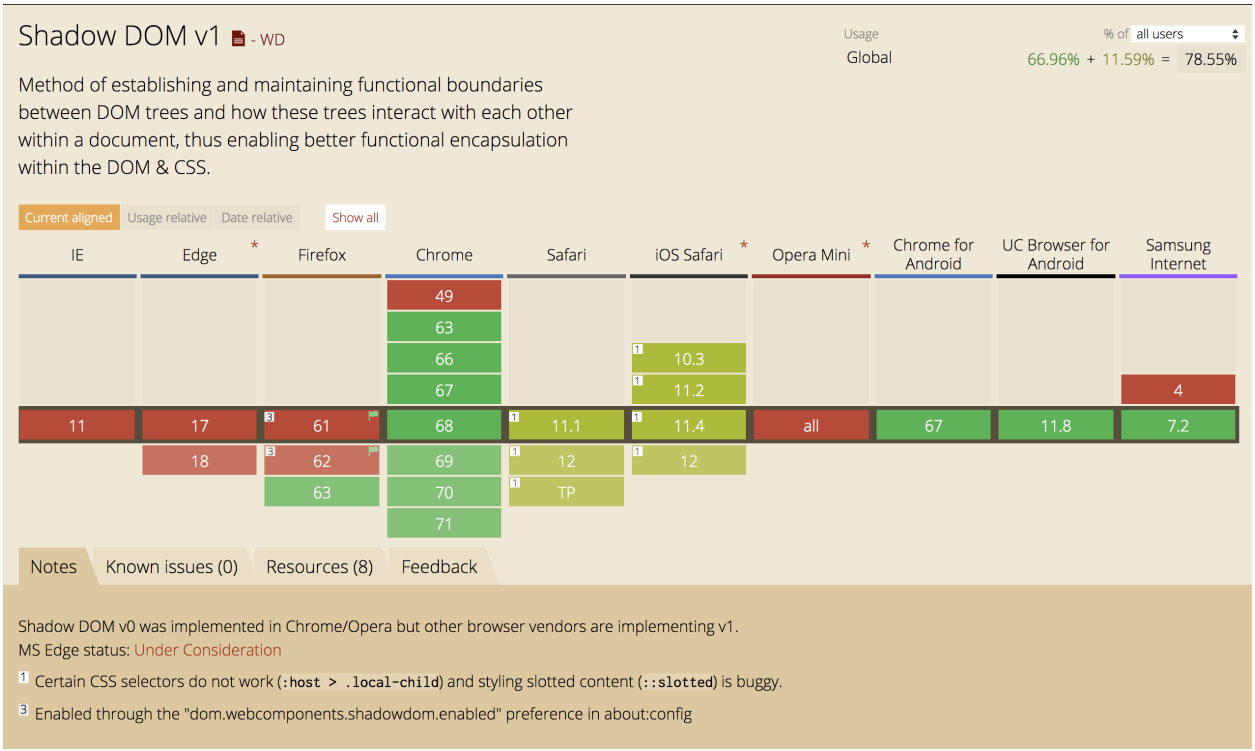
What are Web Components

- WC is an open standard on W3C[6] already implemented in modern browsers
- Extensions of the popular HTML tags
- Can be reused across sites and applications
- Are a combination of both state and view
- Can be grouped in collections of components
- One standard multiple implementations: Angular Elements[7], Stencil[8], Polymer 3[9]
- Natively supported by modern browsers or via polyfills in the old ones WCs are fast because are browsers native elements
- Independent of a JS framework - can be used with Angular, Vue, Preact or the framework of the future, or no framework at all

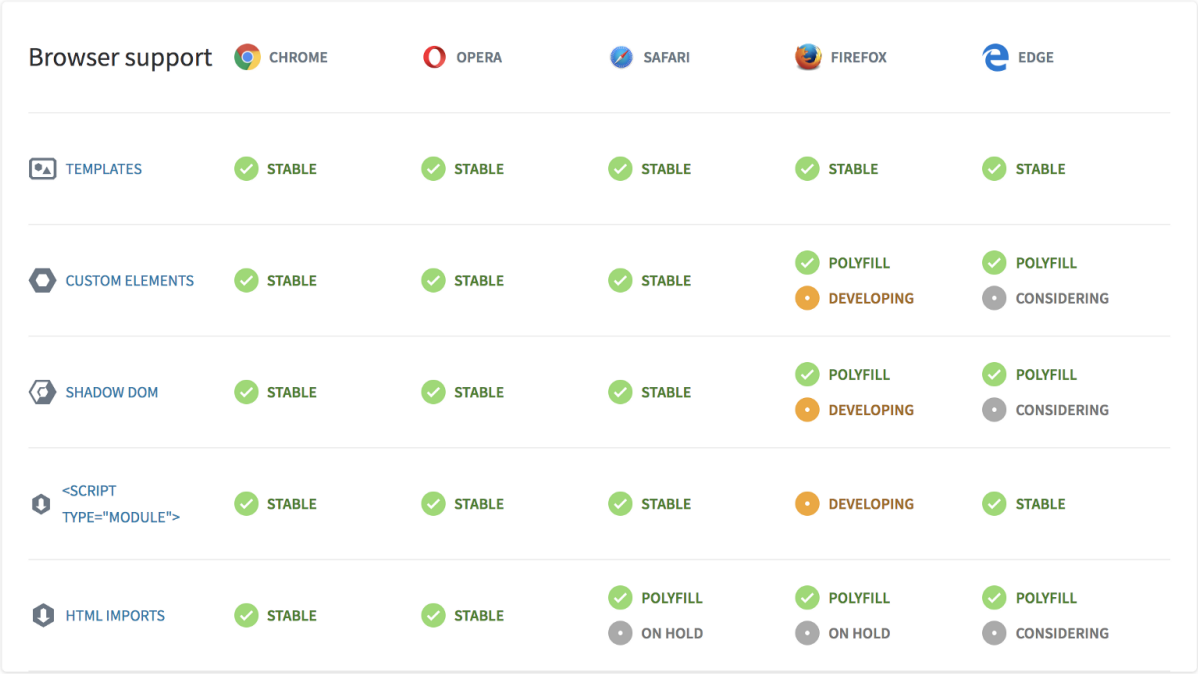
[4] https://material.io/
[5] https://developer.apple.com/design/human-interface-guidelines/
[6] https://www.w3.org/
[7] https://angular.io/guide/elements
[8] https://stenciljs.com/
[9] https://www.polymer-project.org/



Custom Elements support in modern browsers



Shadow DOM support in modern browsers



Web components support in browsers

Web components history

- The history of Web components is starting with Facebook React - a first non-standard way of building Web components for Facebook apps (https://www.quora.com/Is-the-Facebook-app-built-with-react-js)
- Angular at the beginning it was an MVC request/response framework (AngularJS = 1) - Angular >=2 is a component-based system (it chooses the same React path)
- The Polymer 3 project[10] is the Google way to create a WCs framework in a standard way (WCs) - unfortunately using Bower and not NPM and ES6 modules
- Polymer 2 is fixing these problems - ES6 Modules and NPM for better integration with tools
- Angular 6 is introducing Angular Elements - a way to convert classical Angular 2 components to the WC3 standard
- Stencil JS[11] by Ionic is a way to create standard WCs - the entire Ionic 4 components ecosystem is rebuilt with standard components
- Google Amp, Ionic 4 components are examples of standard WCs

JavaScript is evolving

JavaScript is the language used by Ionic.

[10] https://www.polymer-project.org/3.0/docs/about_30
[11] https://stenciljs.com/

JavaScript

JavaScript is the language used by de developers for giving dynamism to the Web applications that are running in the browser (CSS is used for styling them). In general, there is a compatibility between the JavaScript implementations in the modern browsers (Google Chrome, Firefox, Apple Safari, Microsoft Edge), mainly because all are implementing a specification named ECMAScript which is responsible for regimenting the language syntax an the way this syntax is interpreted by the browsers. Throughout the developers, there is a known exception Microsoft Internet Explorer which it was always out of the standards in the past - this problem is fixed by Edge that is was released starting with Windows 10.

JavaScript is a language build on the 'single threaded' model - because it was created for the GUI purposes where the user interaction with the app interface is generating a big number of events that have to be processed rapidly, one after one. This is contrary to the 'multithreading' concurrency, specific to the server applications. This concurrency model is eliminating many possible concurrency problems like 'race conditions'. And because is based on an 'event loop' that is consuming events, it is very simple to create new events - this is an explanation for the asynchronous nature of the language.

In the last time, there is a tendency of using JavaScript not just on the client side but also on the server side using technologies like Node JS (which is also used by Ionic). This is a reason for the explosion of the JavaScript community. Also, JSON - which is, in fact, the data model from the JavaScript language is a preferred choice in client-server communication (a replacement for XML) - mainly when the client and the server are 'speaking' Javascript.

There are a series of popular JavaScript frameworks:

- JQuery - initially created for reducing the differences between the JavaScript implementations on various Web browsers
- BlackboneJS, React.js, Vue.js, Angular 1/2 - MVC frameworks, letting us implement an application completely and in a clean manner on the client side in an MVC style (taking with the server side using REST API calls)
- UnderscoreJS, Lodash - simplify the work with JSON data
- RxJs - is short for Reactive Extensions for Javascript which brings the concept of Observables, which we know and love a lot of a lot of server-side technologies, to Javascript world.
- TypeScript, Coffe Script - programming languages 'transpiled' to JavaScript

EcmaScript

The JavaScript language is specified by a standard named ECMAScript (European Computer Manufacturers Association). The language used by all of our day's browsers, named also *regular JavaScript* is the version ES5 of this standard. Angular is based on TypeScript which has many elements from the version ES6 of the standard (also named ES2015) which is coming with a set of new concepts, not existing in ES5:

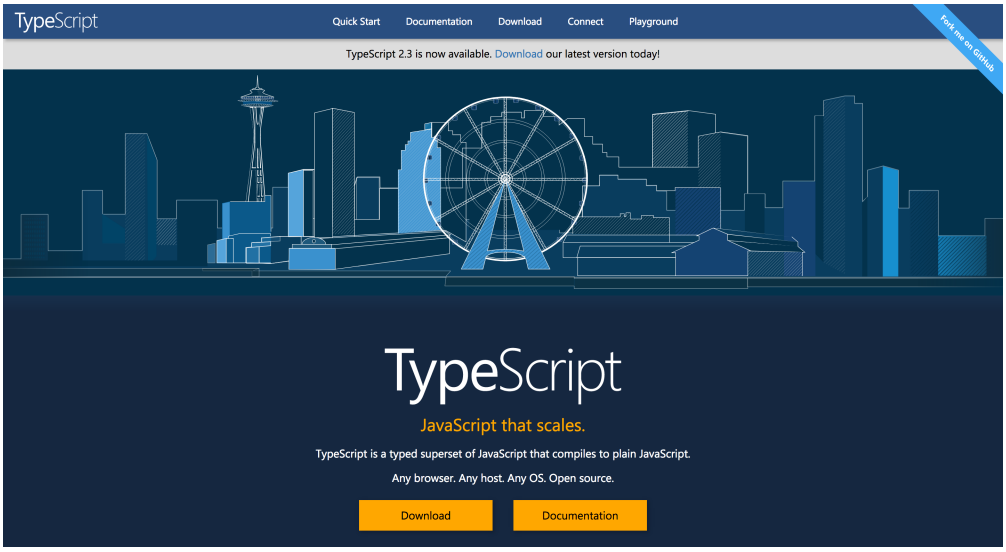
- Classes – ES6 classes provide a way to encapsulate and extend the code.
- Modules – Provides a modular way of organizing and loading code.
- Arrow functions – A short-hand version of an anonymous function.
- Block-level scope – ES6 now supports scoping variables to blocks (if, for, while, etc.) using the let keyword.
- Constants – We can now define constants in ES6 code using the const keyword.
- Destructuring – A succinct and flexible way to assign values from arrays or objects into variables.
- Generators – Specialized functions that create iterators using function* and the yield keyword.
- Promises – Used with async operations.
- Rest parameters – Replaces the need for using arguments to access functions arguments. Allows us to get to an array representing "the rest of the parameters"
- Default parameters – Ever wished that a function parameter could be assigned a default value? We can do that now in ES6.
- Map – Dictionary type object that can be used to store key/value pairs.
- Set – A collection object that can be used to store a list of data values.
- Template Strings – Clean way to build up string values.

In the modern browsers, the support for JavaScript ES5[12] is excellent. JavaScript ES6[13] is also well supported in the modern browsers.

TypeScript

The Angular framework starting with version 2 is coming with the TypeScript language[14] which is adding data types to the classes and the new constructs taken from ES6[15]. TypeScript is very similar with an object-oriented compiled language like Java - now when the source is edited (before execution), type checks and static checks, in general, can be done - this is a significant advantage because the number of possible runtime errors is decreased. There is excellent support in IDEs for the Typescript language.

[12] http://kangax.github.io/compat-table/es5
[13] http://kangax.github.io/compat-table/es6
[14] https://www.infoq.com/articles/Angular2-TypeScript-High-Level-Overview
[15] http://kangax.github.io/compat-table/es6/#typescript



The official Typescript Web page

Because TypeScript is not understood by the browsers, there is a process of **transpiling**. This is not a compilation to an intermediate bytecode like in Java language, but a transformation of sources from TypeScript to ES5, which is understood by all the browsers. This transformation it is done automatically by Ionic in a dynamic way using the transpiler that is coming with the TypeScript language (there are transpiler for the conversion from ES6 to ES5: Traceur[16] by Google and Babel[17] created by the JavaScript community)

Mobile development with Ionic 4

The most interesting think about Ionic is that we can develop one application that can be published in any of the major existing application stores and can run on any existing mobile device. This can be a significant advantage because we will write the app once and package it to different platforms.

We say that Ionic is a framework for developing *hybrid mobile applications* in contrast with the classical mobile development which is named *native mobile development*. There is a third style named *compiled app development*. What do all these mean?

Mobile Native language development

To develop natively, it means to develop for devices of a specific mobile platform, using the programming language, the SDK, and the tools that are specific for that platform, to build a specific platform binary and to publish it in that corresponding store. For example, Android devices are running the Android operating system which is based on Java programming language (a subset of

[16] https://github.com/google/traceur-compiler
[17] https://babeljs.io/

it) or Kotlin, and we can develop using the Android SDK and the Android Studio IDE (based on IntelliJ Idea). After building we will get a **.apk** file that can be published in Google Play store.

But if we want the same application in another app store, Apple App Store because we want to be available on iPhones and iPads devices running the iOs operating system, we have to rewrite the app in Objective-C or Swift and to use the Xcode mobile SDK and IDE. The same for Windows Mobile which is based on C# and Visual Studio for development.

Compiled app development

In this category are React Native[18], Native Script[19], and Flutter[20] which will compile your XML/HTML/CSS code components to native code. We name these UI frameworks, and they don't use a WebView to render a page but a native application is generated from your code.

Excepting Flutter which is based on Dart language[21], the rest of the UI frameworks are based on JavaScript. The React Native is the most popular UI framework and is used by big companies, first of them being Facebook, the sponsor of the framework. Airbnb is sunsetting React Native[22]

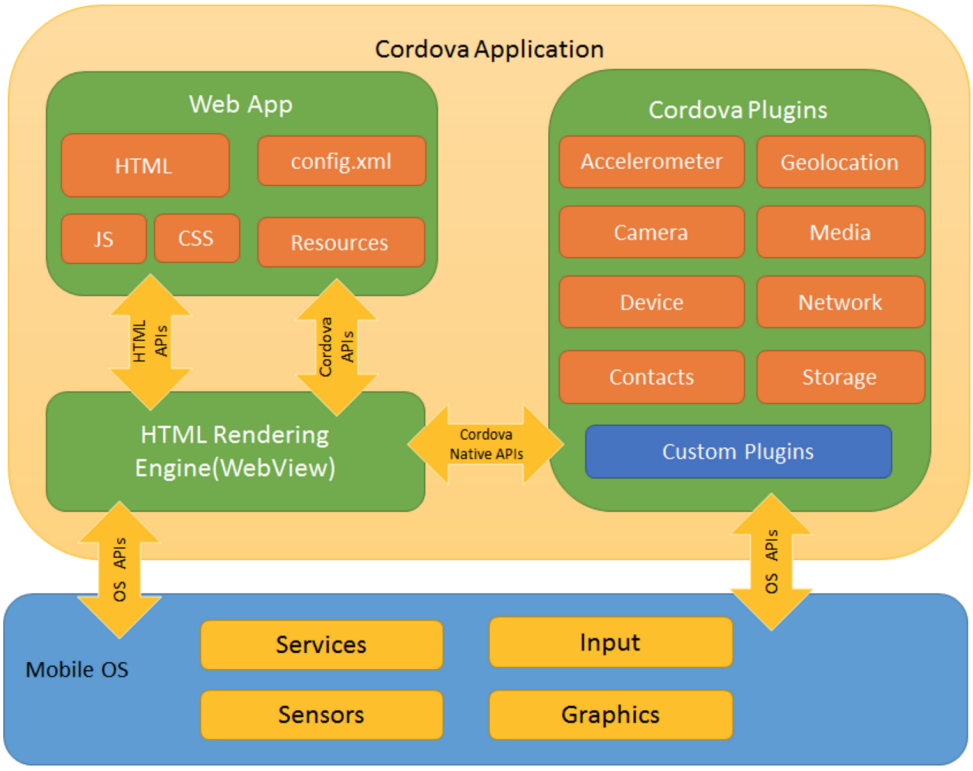
- Native Script Site: https://www.nativescript.org/
- Native Script Resources: https://market.nativescript.org/
- React Native (Facebook) Site: https://facebook.github.io/react-native/
- React Native Resources: http://www.awesome-react-native.com/
- Flutter: https://flutter.io/
- Flutter Resources: https://github.com/Solido/awesome-flutter

Mobile Hybrid development

The hybrid applications development is based on the possibility to develop a single HTML 5 application that can run in any mobile browser (any mobile device has a browser), and we can use JavaScript for accessing native capabilities of that device. More precisely the app will run in the WebView (a special browser window) and the Cordova framework I used for making native calls. In the end, the web application will be packaged for each platform in a native app (.apk files for Android and .ipa for iOs) that can be published in stores.

The architecture of a Cordova application[23] is ilustrated on the below image:

[18] https://facebook.github.io/react-native/
[19] https://www.nativescript.org/
[20] https://flutter.io/
[21] https://www.dartlang.org/
[22] https://medium.com/airbnb-engineering/sunsetting-react-native-1868ba28e30a
[23] https://cordova.apache.org/docs/en/latest/guide/overview/#architecture



Cordova architecture

Advantages:

- cross-platform mobile development (we are in the same situation like Java: Write once, run anywhere) we will develop an HTML5 application with very well known technologies like JS (and JSON), CSS for powerful styling, HTML
- the developed app can also run in a desktop browser so we can deploy it as a classical Web application and we can use PWA for encouraging mobile installs
- we can rely on the ability of the Web apps to be responsive[24] and to auto-adapt to the devices resolutions
- because we have a web application, we can use the browser from to development environment to test and debug it

Disadvantages:

- some very particular native functionalities are hard to be implemented compared with a native application

[24] https://developers.google.com/web/fundamentals/design-and-ui/responsive

Mobile devices and mobile operating systems

There are a significant number of **smartphones** and **tablets** on the market, each having a given operating system, and it is produced by a given manufacturer. We live in the years when the number of mobile phones overcomes the number of desktop and laptop computers, and the difference is still growing (Google is reporting that there are more searches from mobile now than from desktops).

There are some stores today where we can publish our mobile application based on its operating system. There are some signs[25] that the number of devices Android is the winner in the market (~70%), followed by iOS (~20%). Windows Phone has just a few percents but they choose to invest in mobile in the last time, and our application can also be available to other Windows 10 devices like PC, tablet, Xbox due to UWP[26] (Windows 10 is around 25% from desktop OS market). Here is the list of major mobile operating systems on the market:

Devices	OS	Store Language	
Android phones and tablets	Google Android	PlayStore	Android(Java)
iPhone and iPad	Apple iOs	AppStore	Objective-C or Swift

Android OS

Google is the owner of Android OS[27] and the devices are manufactured by many others like Samsung, Motorola, etc. There is a reference device Google Pixel[28] series produced by Google. Till now it was Nexus, but the manufacturer was not Google. The programming language is Java (Android is derived from Java), and the development tool is Android Studio (based on IntelliJ Idea now, in the past was Eclipse) which can run on Linux, Windows or Mac operation systems.

iOs OS

Apple is the owner of the iOs operating system[29] and is also the manufacturer of the iPhones[30] and iPads[31]. The programming language in Objective C[32] with the addition of a more scripting language named Swift[33]. The development tool is named XCode[34], and we need a Mac OS system to develop our application.

[25] http://www.idc.com/promo/smartphone-market-share/os
[26] https://docs.microsoft.com/en-us/windows/uwp/get-started/whats-a-uwp
[27] https://www.android.com
[28] https://madeby.google.com/phone
[29] http://www.apple.com/ios
[30] http://www.apple.com/iphone
[31] http://www.apple.com/ipad
[32] https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html
[33] https://developer.apple.com/swift/
[34] https://developer.apple.com/xcode

Mobile App stores

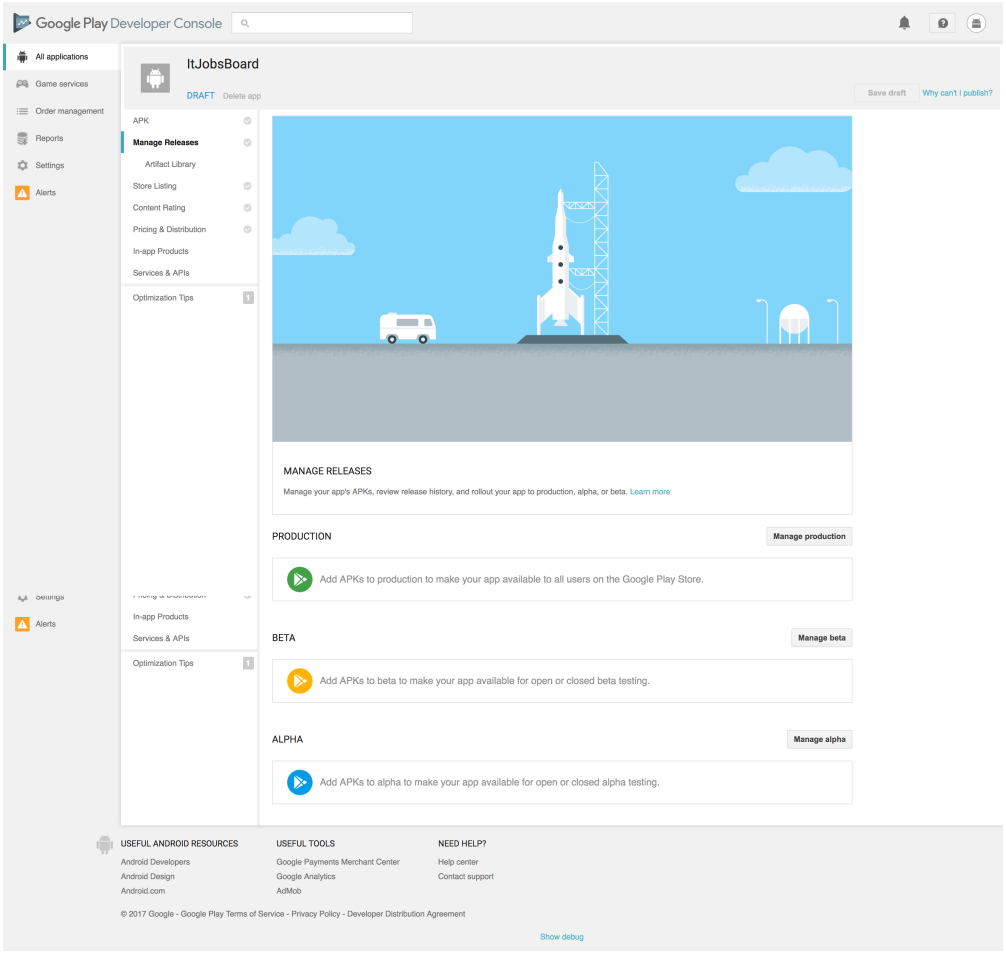
Publishing an application to the app store is simple in the case of Google Play store because there are not so many reviews steps. But will be more difficult with the Apple AppStore and Windows Store. We have to improve our application usability to be accepted in the store. We can use a professional template from *Ionic Market*[35] an official *Ionic market* for starters, plugins, and themes. Now we will describe the steps that should de follow to publish the app in all three stores.

Google Play Store

The Android store is Google Play[36] - our application is easily accepted (in hours) on this store. There are many free applications in this store, and their users are in general not paying for apps. Will cost us $25 per life for publishing in this store and Google takes 30% of revenue. The user receives 70% of revenue.

First, we have to create a Play Developer account[37], and there we have to create an application and edit our *Store Listing*. *Title* and *Full Description* are essential for ASO[38] - be aware that around 60% of users are coming from store searches. In *Manage Releases* we will upload our *.apk* files.

[35] https://market.ionic.io
[36] https://play.google.com/store
[37] https://developer.android.com/distribute/googleplay/developer-console.html
[38] https://blog.kissmetrics.com/app-store-optimization/



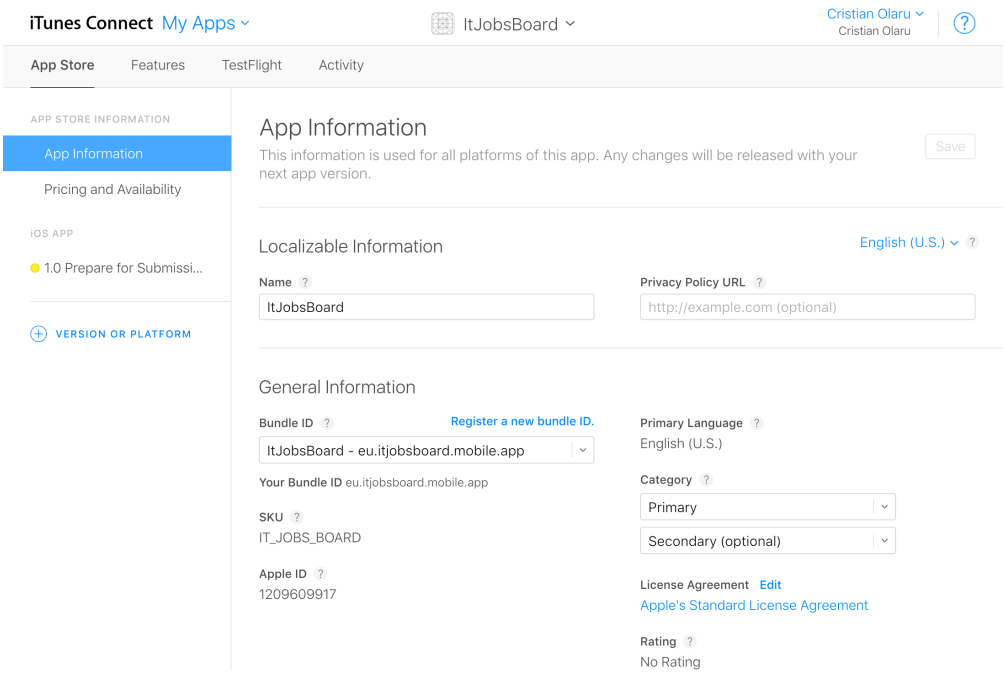
Google Play Store

Apple App Store

The store is Apple Store[39] - our application is verified in detail and takes around one week to be published. We can make more money from this store, and in general, there are more payable applications than free. We pay $99 per year for publishing[40] on this store.

For publishing, we need to have an Apple Id[41]. Then we have to go to our Apple developer account in the area named Certificates, Identifiers & Profiles[42]. There we have to create an application id. Then, we have to go to our iTunes Connect account[43] where we have to request a new application using *New App* using the application identifier created in the previews step. There we have to complete *App Information* and other application data before submitting it for approval.

[39] http://www.apple.com/itunes/charts/free-apps
[40] https://developer.apple.com/
[41] https://appleid.apple.com
[42] https://developer.apple.com/account/ios/certificate
[43] https://itunesconnect.apple.com



Apple AppStore

Also, we have to upload the binary file, and this can be done in two ways. The first option is to use *Product/Archive* option from *Xcode*. Second is using a distinct application named *Application Loader*.