Cordova - Guide - App Architecture

- Dr Nick Hayward

A brief overview and introduction to Apache Cordova application architecture.

Contents

- intro
- JS & Web plugins
- Web container
- SDKs and OSs

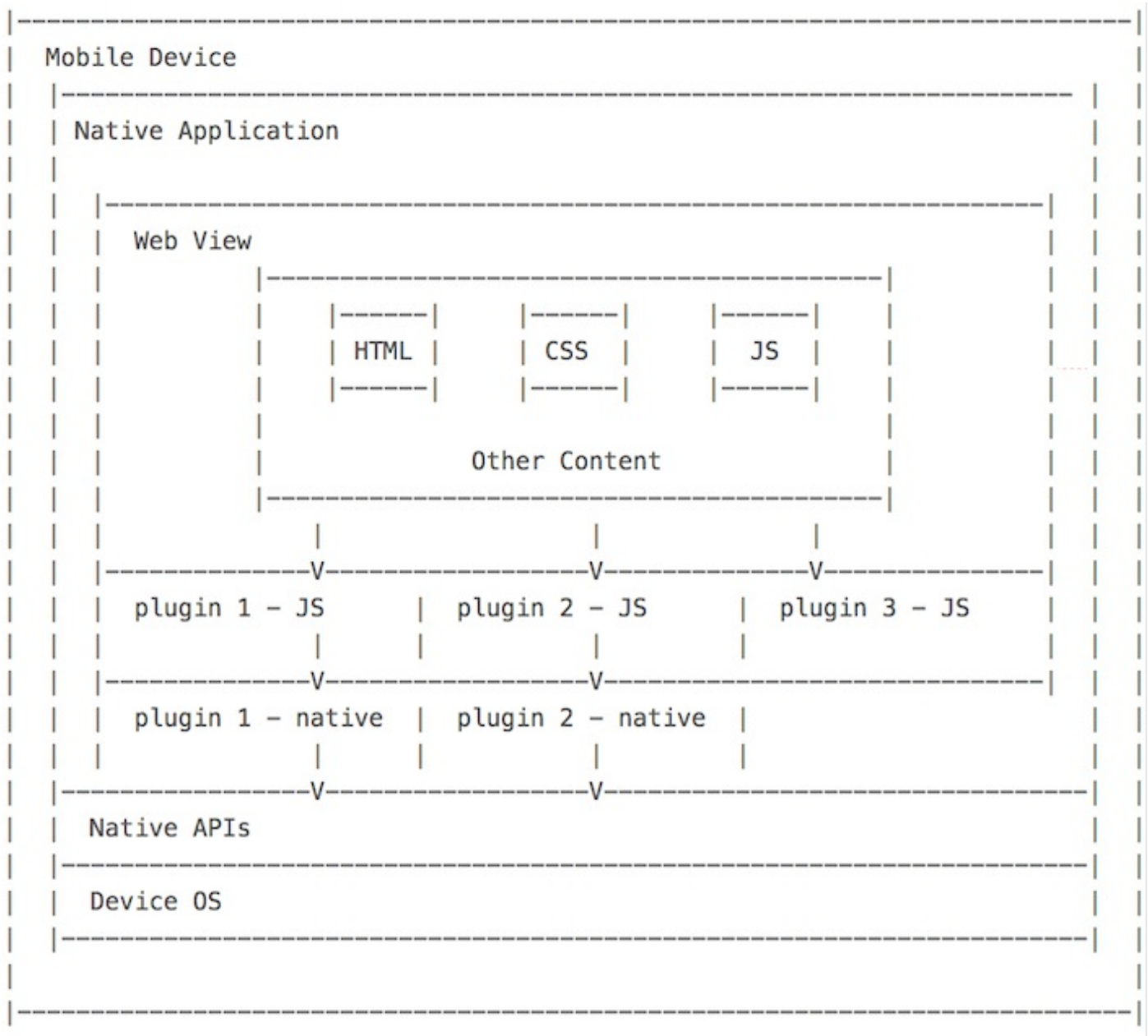
Intro

Let's start working our way through Cordova architecture, and then app design and development.

In essence, Cordova can be thought of as consisting of the following components

- source code to allow us to build a native application container. This will be specific to the mobile platforms we choose to add to our project, for example Android, iOS. This container allows us to render the web application on the native device. If the platform, such as Firefox OS, natively supports a web application, then there is no need for this container.
- then, we have a collection of various APIs, implemented by Cordova as plugins, which provide a web application running within the container access to native device functionality, APIs, and applications.
- Cordova also provides a useful set of tools that help us manage the underlying process of creating an application, and its project files, managing any required plugins, building our native applications, using the native SDK of course, and testing our application with emulators and associated simulators.

Cordova Design - architecture - diagram



JS & Web plugins

In the previous diagram, the outline architecture includes the option for JavaScript only plugins. Whilst we normally consider JS plugins in Cordova as simply a bridge or go-between from our web container to the native APIs, a useful way to expose native device functionality to the web application. However, we can also use and develop plugins purely in JS. We can add an existing library, to help with data visualisations, graphics, and so on, and we can create our own focused plugins for any abstraction of application features, or other specific requirements.

We've also noticed in recent years greater support for native functionality at the web application level. The simple support for touch enabled input and interaction has obviously helped, but we have many mobile UI libraries natively developed in JS. We also have access to a growing number of HTML5 APIs, including a useful implementation of service workers for easier development of offline apps.

We'll look at creating a JS only plugin as we consider plugin development later in the semester.

Web container

Whilst Cordova development uses many of the same underlying technologies as standard web application development, there are a few limitations relative to network access that we need to consider.

To allow us to develop a hybrid mobile application with Cordova, a web application needs to be written as a self-contained application. In effect, it needs to be able to run within this container as a self-contained application. A

reliance on the fetching of external resources for a complete app is not considered particularly good practice, and will naturally be an issue if we lose a network connection.

So, our `index.html` file will normally be the only HTML file we use, for most apps at least, and, as we've seen, our separate pages will be containers within this file.

As developers, we need to rethink our approach to building such mobile web stack applications to help us leverage the inherent capabilities of Cordova instead of falling back on old web development habits.

With self-contained applications, we need to ensure any application files and data are at least available to allow the application to launch and load on the native device. In effect, a basic minimum necessary to load the application and render the initial UI. If necessary, the application can then optionally fetch data from, and generally communicate, with a remote server.

Remember, whilst it's possible to build a minimal shell for our application, and then fetch data as required from a server, it's best to consider this an option when we are unable or unwilling to store application data and functionality within the container or on the native device.

A banking app is a good example of this type of limited scope app. The app itself is able to load, render, and show options to the user without a network connection. However, if the user wishes to query their account, make a payment &c., they will need to authenticate their session with a remote server.

In effect, we need to think about the various stages of the design of the container for our applications, and tailor accordingly.

SDKs and OSs

As we build our Cordova applications, regardless of whether we use just the default Cordova APIs or many additional options, each app still has to be packaged into a native application to allow it to run on the host native device. Each native SDK has its own set of custom or proprietary tools for building and packaging their native applications.

Therefore, to be able to build our Cordova applications for a native device, the web content portion of the app is added to a project applicable to the chosen mobile platforms, such as Android, iOS, and Windows 10 Universal Platform. This project is then built for each required platform, via Cordova CLI, using each of the applicable platform's specific set of tools wrapped in a Cordova build command.