عنوان: **شروع کار با Apache Cordova در ویژوال استودیو #1**
نویسنده: غلامرضا ربال
تاریخ: ۱۳۹۴/۰۱/۰۵ ۱۰:۵۰
آدرس: www.dotnettips.info
گروه‌ها: Visual Studio 2013, Apache Cordova, Mobile Programming

Apache Cordova یک فریمورک سورس باز برای ساخت اپلیکیشن‌های **چند سکویی** موبایل (cross platform) با استفاده از Html5 می‌باشد.

طی چند مقاله، با استفاده کردن از این فریمورک در VS آشنا خوهیم شد.
هدف خالقان Cordova یافتن یک راه ساده برای تولید اپلیکیشن‌های چند سکویی موبایل بود که برای رسیدن به این هدف تصمیم گرفتند از تکنولوژی‌های بومی (native) و تکنولوژی‌های وب استفاده کنند. به این نوع از اپلیکیشن‌های موبایل، Hybrid Application می‌گویند.
Cordova دارای قابلیت‌های بومی بالایی است و مهم‌تر اینکه به طور طبیعی توسط مرورگرها پشتیبانی می‌شود. بعد از تولد Corodva، این فریمورک تبدیل شده است به بهترین روش تولید اپلیکیشن‌هایی که بر روی چند نوع پلتفرم کار می‌کنند.
پیشتر محدودیتی که وجود داشت شامل این بود که اپلیکیشن‌های موبایل، به چیزهایی بیشتر از HTML و مرورگرهای وب، نیاز داشتند. برخی از این نیازها عبارتند از ارتباط متقابل وب اپلیکیشن‌ها با دوربین یا لیست شماره‌های تماس گوشی که برطرف کردن آن هم به راحتی امکان پذیر نبود.
Cordova برای مقابله با این محدودیت، مجموعه‌ای از رابطهای برنامه کاربردی را برای توسعه قابلیت‌های بومی device، مانند لیست مخاطبین، دوربین، تشخیص دهنده‌ی تغییر جهت گوشی (accelerometer) و مانند این موارد، در نظر گرفته است.

Cordova شامل یک سری کامپوننت به شرح زیر است:
سورس کدی برای هر Container و برنامه محلی برای هر یک از سکوهای موبایل که پشتیبانی می‌شوند. container، کدهای Html5 را بر روی دستگاه (Device) رندر می‌کند. (در مطالب بعدی در مورد این مطلب توضیح خواهم داد)

مجموعه‌ای از رابطهای برنامه‌ی کاربردی که امکان دسترسی به قابلیت‌های بومی دستگاه را به برنامه‌ی وبی که درون آن در حال اجرا است، می‌دهند.

مجموعه‌ای از ابزارها برای مدیریت فرآیند ایجاد پروژه، مدیریت پلاگین‌ها، ساخت (با استفاده از SDKهای محلی) برنامه‌های محلی و تست برنامه بر روی دستگاه موبایل یا شبیه ساز .

برای ساخت یک برنامه‌ی Cordova، در واقع شما یک وب اپلیکیشن می‌سازید و آن را داخل Container محلی، بسته بندی می‌کنید. سپس تست کرده و بعد از دیباگ می‌توانید اپلیکیشن را توزیع کنید.

فرآیند بسته بندی :

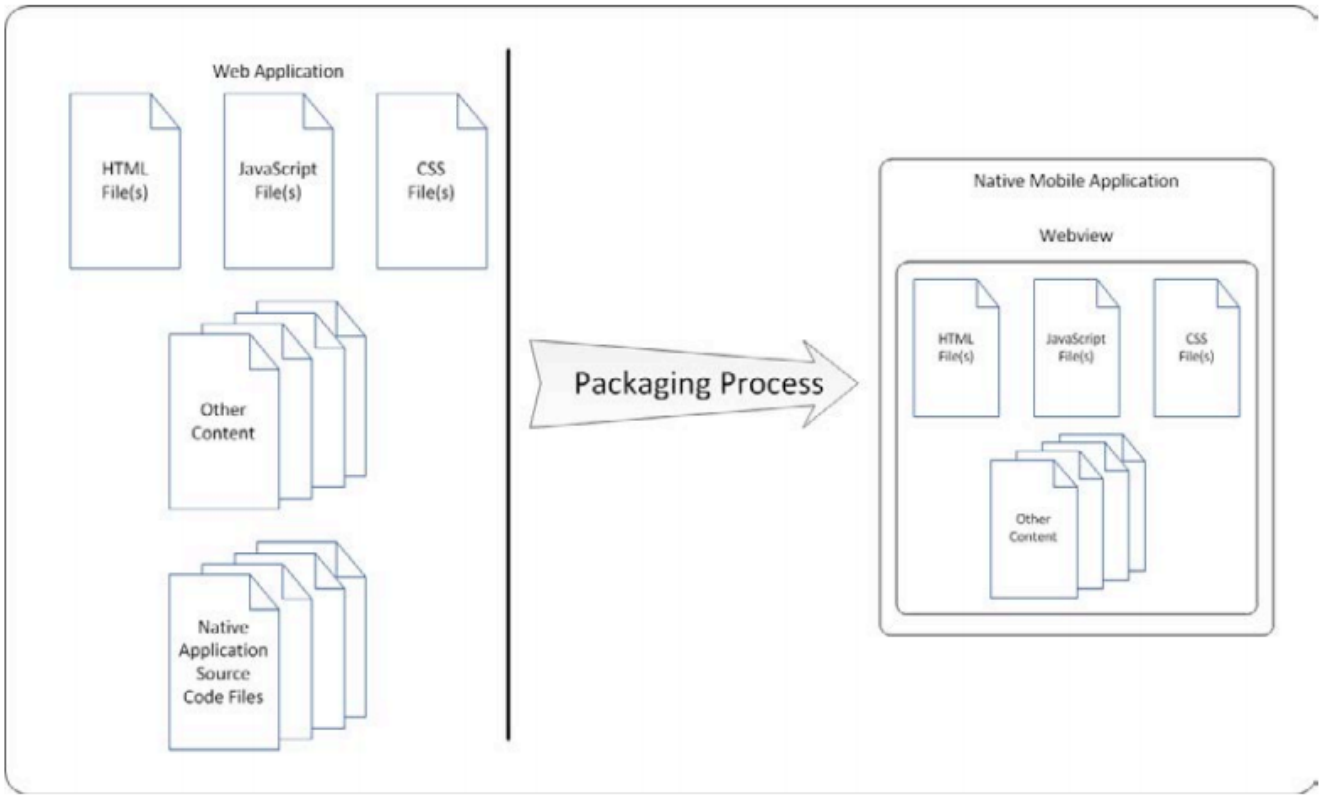


Figure 1.1 Apache Cordova Application Packaging Process

داخل اپلیکیشن محلی، رابط کاربری اپلیکیشن شامل یک صفحه‌ی نمایش که خود آن چیزی نیست به غیر از یک Web View که از فضای نمایش دستگاه استفاده می‌کند. زمانی که برنامه آغاز به کار می‌کند، برنامه‌ی وب نوشته شده، درون این web view لود میشود و کنترل‌های موجود، برای تعامل کاربر با برنامه‌ی وب، در اختیار آن قرار می‌گیرند. مانند تعامل کاربر با محتوا، در برنامه‌ها ی تحت وب، لینک‌ها، کدهای نوشته شده‌ی js در فایل‌ها و یا حتی می‌تواند به اینترنت دسترسی داشته باشد و محتوا را از یک وب سرور تغذیه کند.

درباره Web Views

Web View جزء برنامه‌های بومی است که برای رندر کردن محتوای وب (به عنوان نمونه صفحه HTML) درون اپلیکیشن بومی یا صفحه نمایش استفاده می‌شود. در اصل Web View یک Wrapper برنامه نویسی شده قابل دسترس برای نمایش محتوای صفحات وب توکار است.

به عنوان مثال:

در اندروید با استفاده از WebView موجود در (Using andoid.webkit.WebView) , در iOS با UIWebView موجود در (Using System/Library/Framworks/UIKit.framewor) به این هدف دست پیدا می‌کنند. وب اپلیکیشن ما درون این Container مانند سایر وب اپلیکیشن‌هایی است که هر روز با آنها سرو کار دارید و آنها را در مرورگر موبایل خود اجرا می‌کنید و می‌توانید بین صفحات Navigation داشته باشید. وب اپلیکیشن‌های معمول باید روی یک سرور هاست شوند. در برنامه نویسی چند سکویی با Cordova، این کار می‌تواند درون Cordova Application انجام گیرد.

شاید سؤالی در ذهن شما وجود داشته باشد که مرورگر معمولا به اپلیکیشن‌های موجود در دستگاه، سخت افزار و یا APIهای بومی دستگاه، دسترسی ندارد. برای مثال شاید بگویید که یک وب اپلیکیشن معمولا به لیست مخاطبین با دوربین دستگاه و ... دسترسی ندارد.

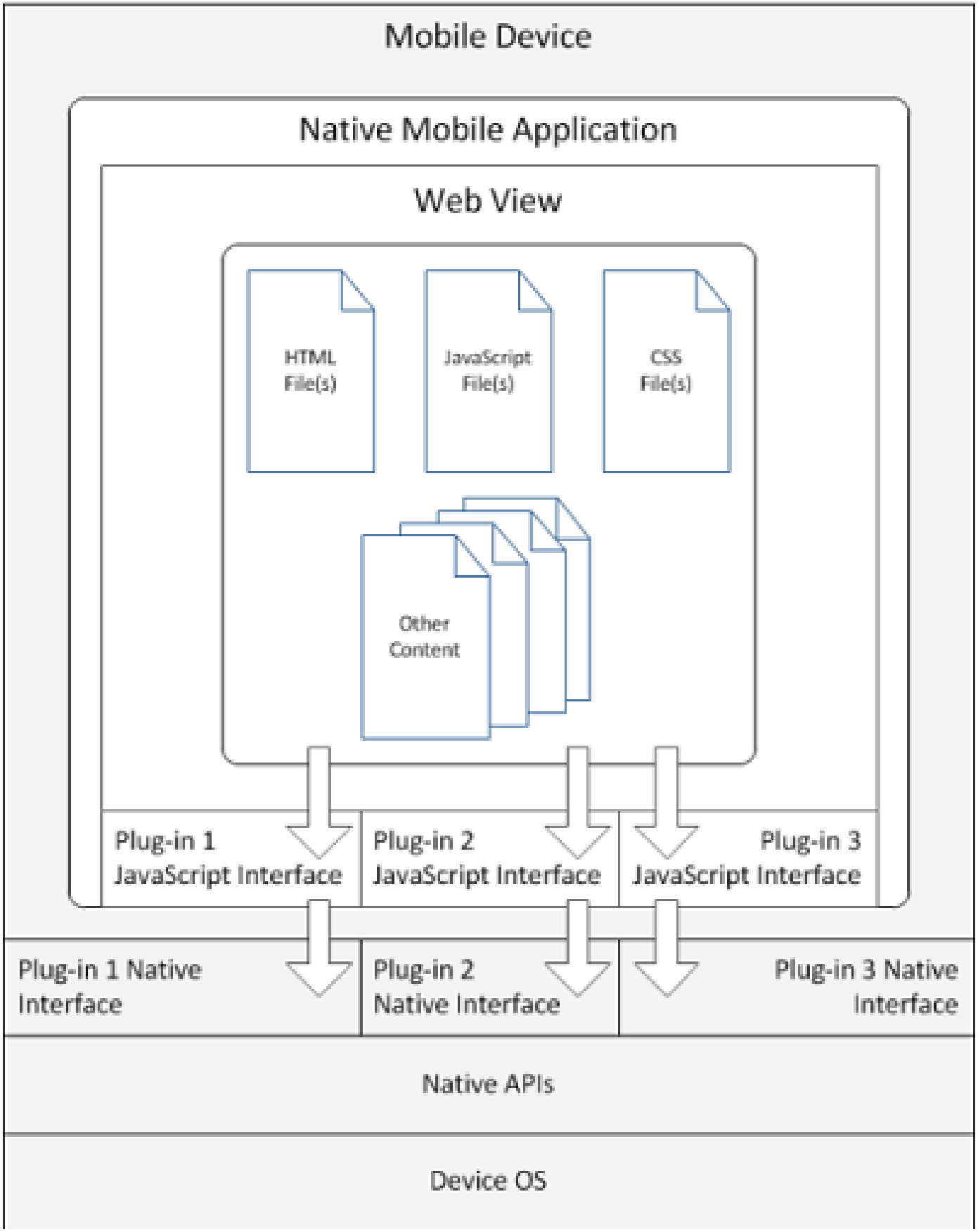
جواب : در واقع امکان دسترسی به این قابلیت‌ها توسط اپلیکیشن بومی (native mobile application) ایجاد می‌شود.

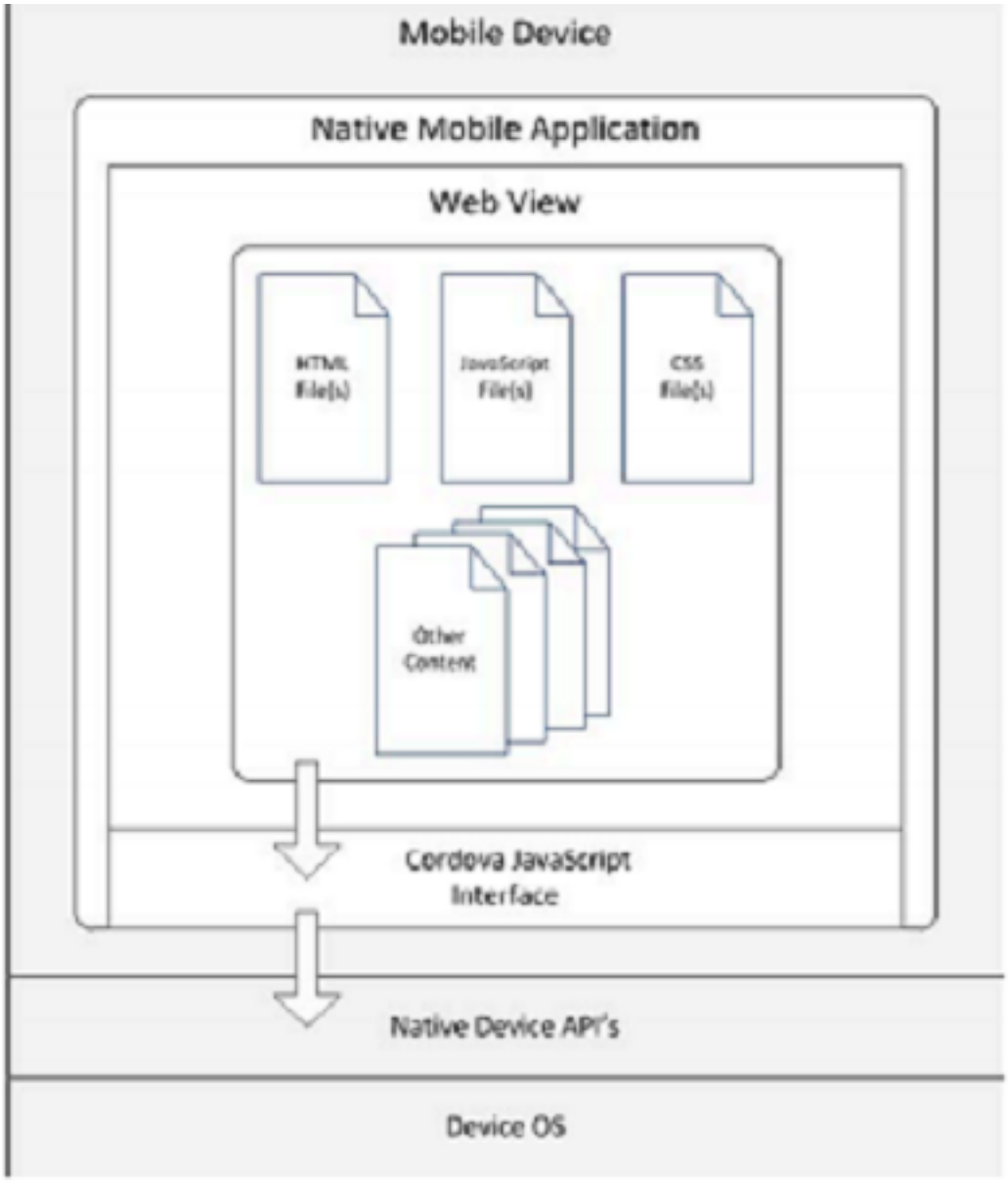
Cordova مجموعه ای از APIهای جاوااسکریپت را به عنوان اهرم اجازه برای دسترسی برنامه وب درون cordova container به قابلیت‌های بومی دستگاه، در اختیار توسعه دهندگان قرار داده است.

این APIها در دو بخش پیاده سازی می‌شوند:

-1کتابخانه‌ی جاوااسکریپت که اجازه‌ی استفاده از قابلیت‌های بومی را به وب اپلیکیشن می‌دهد و کد بومی مشابه در Container اجرا می‌شود که مربوط است به بخش بومی این API ها. در اصل یک کتابخانه‌ی جاوا اسکریپت وجود دارد، اما بخش بومی APIها وابسته به سکوی (platform) انتخاب شده پیاده سازی می‌شود.

اگر شما از APIهای موجود استفاده نکنید، می‌توانید آنها را از کتابخانه جاوااسکریپت و native container حذف کنید. این کار به صورت دستی شاید خوشایند نباشد ولی چون در Cordova 3.0 همه‌ی API ها از بیرون وارد می‌شوند، می‌توانید با استفاده از بحث مدیریت پلاگین آن، پلاگین‌ها را اضافه یا حذف کنید. در بخش‌های بعد با مثال‌هایی عملی این مباحث را کار خواهیم کرد. **ادامه دارد..**


Mobile Device
Native Mobile Application
Web View
HTML File(s)
JavaScript File(s)
CSS File(s)
Other Content
Plug-in 1 JavaScript Interface
Plug-in 2 JavaScript Interface
Plug-in 3 JavaScript Interface
Plug-in 1 Native Interface
Plug-in 2 Native Interface
Plug-in 3 Native Interface
Native APIs
Device OS


Mobile Device
Native Mobile Application
Web View
HTML File(s)
JavaScript File(s)
CSS File(s)
Other Content
Cordova JavaScript Interface
Native Device API's
Device OS

نظرات خوانندگان

نویسنده: افشین عباسپور
تاریخ: ۱۳۹۴/۰۱/۰۸ ۱۲:۰

خیلی ساده و روان توضیح دادید . متشکرم .
-1 در مورد محدودیت‌ها هم توضیح بدید لطفا ...
-2 اینکه Performance این برنامه‌ها چطور است ؟
-3 امنیت برنامه‌های تولید شده چگونه است ؟ آیا سورس برنامه رو میتوان غیر قابل دسترس کرد ؟
باز هم تشکر از این آموزش . منتظر آموزشهای بعدی و تکمیل این بحث هستم .

نویسنده: غلامرضا ربال
تاریخ: ۱۳۹۴/۰۱/۰۸ ۱۲:۴۵

حتما مقاله ای را برای این در ادامه تهیه خواهم کرد تا به طور کامل این مباحث رو پوشش دهد. فعلا شاید این بتواند کمک کند.در ضمن سرعت این برنامه‌های نوشته شده با Cordova نسبت به برنامه‌های بومی اندکی کم است (مزایا و معایب خود را دارد) و برای اینکه سورس در دسترس نباشد روش هایی برای آن در نظر گرفته شده. این مقاله هم مفید است.