KossLab
공개SW개발자Lab

자바스크립트를 이용한 네이티브 모바일 앱 개발

Titanium

공개SW개발자Lab 오픈소스프론티어 2기 이종은

KossLab

타이타늄(Titanium) SDK를 이용하면 자바스크립트로 iOS와 안드로이드 네이티브 앱을 동시에 개발할 수 있다. 자바스크립트로 모바일 앱을 개발한다고 하면 하이브리드로 불리는 웹뷰(WebView)를 통해 UI를 구성하는 폰갭(Phonegap)이나 코도바(Cordova) 프레임워크를 생각하는 경우가 많다. 타이타늄은 웹뷰를 이용하는 것이 아니라 자바스크립트와 네이티브 브리지(Bridge) 기술을 이용한다. 자바스크립트에서 네이티브 코드를 실행(invoke)하여 동작하고 네이티브 UI를 사용하는 진짜 네이티브 앱을 만들게 된다.
이러한 방식이 웹뷰를 이용한 종래의 크로스 플랫폼 모바일 개발 방식이나 타이타늄과 비슷한 접근 방식으로 모바일 앱 개발을 하는 리액트 네이티브(React Native)와 비교하면서 타이타늄을 중심으로 크로스 플랫폼 개발 방법에 대해 설명한다.

[목차]

1. 크로스 플랫폼 모바일 앱 개발

모바일 운영체제 시장은 크게 iOS와 안드로이드로 나뉜다. 이 두 가지 이외에 윈도우와 블랙베리 등이 있지만 그 점유율은 미비하기에 여기서는 iOS와 안드로이드에 대해서만 이야기 하겠다.

그렇다면 iOS를 먼저 대응해야할까? 안드로이드를 먼저 대응해야 할까? 모바일 앱을 만들어 봤거나 만들려고 하는 사람들이라면 한 번쯤 혹은 어쩌면 수도 없이 하는 고민일 것이다. 고민이 되는 이유는 두 OS중에 어느 것이 절대적으로 중요하다고 말할 수 없기 때문이다. 단순히 시장 점유율만 보자면 안드로이드가 월등히 앞서지만 어떤 사람들을 대상으로 어떤 서비스를 제공할지에 따라 분명 그 대상이 되는 사용자가 사용하는 스마트폰의 모바일 OS의 점유율은 판이하게 다르다. 이렇다보니 개발자 개인 혹은 팀에서 어느 하나의 플랫폼을 선택하고 나머지를 완전히 무시할 수는 없다. 결국 현실은 최소한 iOS와 안드로이드 모두를 염두에 두고 있어야 한다.

iOS와 안드로이드를 만든 애플과 구글이 제공하는 이른바 네이티브 개발 도구를 이용하여 앱을 만들게 되면 두 앱은 완전히 다른 언어와 도구를 이용하여 만들게 된다. iOS는 오브젝티브-C(혹은 스위프트)와 Xcode, 안드로이드는 자바(Java)와 안드로이드 스튜디오를 사용해야한다. 다른 언어와 다른 도구를 사용하다 보니 같은 기능이라 할지라도 완전히 다른 코드

로 각각 만들어야 한다. 소프트웨어 개발방법론에서 빠지지 않는 DRY(Don't repeat yourself) 입장에서 보면 이는 바람직하지 않으며 비효율적이라 할 수 있다. 여기에 한 가지 더하자면 스타트업과 같은 작은 규모의 회사나 팀의 입장에서는 양쪽 모두를 개발할 수 있는 개발자를 찾기도 어려우며 각 플랫폼별로 따로 채용하기에도 부담이 된다.

크로스 플랫폼 개발이란 하나의 플랫폼에 종속되지 않는다는 것이다. 여러 플랫폼에서 동작하는 제품을 만드는 것을 의미한다. 모바일에서 이야기 하자면 iOS와 안드로이드 모두에서 동작하는 애플리케이션을 만드는 것이다.

모바일 크로스 플랫폼 개발 프레임워크는 크게 두 가지로 나눌 수 있다. 첫째가 웹뷰 위에서 UI를 구현하고 각 플랫폼 앱으로 패키징을 하는 경우다. 코도바가 가장 대표적인 예다. 웹뷰가 있는 플랫폼이라면 어디든 동작하게 한다는 측면에서는 매력적이지만 네이티브가 제공하는 강력하고 미려한 다양한 UI 컴포넌트를 사용하지 못한다는 점과 이로 인해 해당 플랫폼에서 제공하는 사용자 경험가 차이가 있을 수 있다는 단점이 있다. 두 번째는 바로 하나의 언어로 여러 플랫폼 앱을 개발할 수 있게 해주는 프레임워크들이다. 자바스크립트를 이용하는 타이타늄과 리액트 네이티브, C#을 이용하는 사마린(Xamarin)등이 대표적이다. 이 방법의 장점은 바로 네이티브 UI 컴포넌트를 사용한다는 점이다. HTML과 CSS로 화면을 구성하는 것이 아니라 해당 플랫폼에서 제공하는 네이티브 UI 컴포넌트를 이용한다. 해당 플랫폼의 사용자 경험(UX)을 그대로 가져가면서도 하나의 언어와 도구를 사용하여 여러 플랫폼에서 동작하는 애플리케이션을 만들 수 있다는 것이 가장 큰 장점이다.

	사용언어	UI	지원플랫폼	라이선스
PhoneGap	HTML, JavaScript, CSS	WebView	iOS, Android, Blackberry, Windows Phone, Ubuntu, Firefox OS	Apache 2 라이선스
Xamarin	C#	Native	iOS, Android, Windows Phone	MS 인수 후 공개SW화됨 (MIT 라이선스)
Titanium Mobile SDK	JavaScript	Native	iOS, Android, and BlackBerry, Mobile Web	Apache 2 라이선스)
React Native	JavaScript	Native	iOS, Android	BSD 라이선스

[표 1] 주요 크로스 플랫폼 개발환경 비교

2015년 페이스북의 F8 컨퍼런스에서 리액트 네이티브가 처음 소개될 때 "한번 배워서 모든 곳에 사용한다(Learn once, write everywhere)"가 리액트 네이티브의 특징이자 지향점이라고 했다. 어디서 많이 들어본 말이지 않은가? 그렇다. 한번 쓰면 모든 곳에서 실행되는(Write once, run everywhere) 폰갭, 한번 쓰면 모든 곳에 적용할 수 있는(Write once, adapt everywhere) 타이타늄(Titanium) 프레임워크와 비슷하면서도 다른 표현이다. 표현이 비슷하지만 다르듯이 모두 여러 플랫폼에서 동작하는 크로스 플랫폼 모바일 앱이지만 분명한 차이점이 있다.

2. 무엇이 네이티브 앱인가?

네이티브 앱의 정의는 무엇일까? 여러가지 정의가 있을 수 있지만 이 정의는 기술적인 접근보다는 사용자에게 다가서는 UI와 UX에 더 관련이 깊다고 할 수 있다. 어떤 언어로 만들었는지와 상관없이 아이폰 앱은 아이폰 앱 다워야하고 안드로이드는 안드로이드 앱 다워야 네이티브 앱이라 할 수 있을 것이다. 아이폰 앱 답고 안드로이드 앱 다우려면 해당 플랫폼에서 제공하는 UI 컴포넌트와 해당 플랫폼의 특징적인 기능을 잘 버무려서 앱을 만들어야 한다.
타이타늄과 리액트 네이티브는 HTML, CSS를 기반으로 이러한 UI컴포넌트와 기능들을 흉내내는 것이 아니라 해당 플랫폼에서 제공하는 UI 컴포넌트와 기능들을 자바스크립트에서 사용할 수 있는 인터페이스를 제공한다.

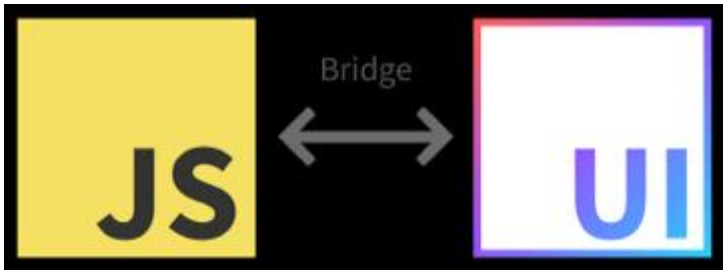
3. 왜 자바스크립트인가?

크로스 플랫폼 모바일 앱 개발에서 빠지지 않는 것이 자바스크립트이다. 타이타늄과 리액트 네이티브만 보더라도 자바스크립트만을 이용해서 앱을 완성할 수 있다. 그렇다면 왜 이러한 플랫폼들이 자바스크립트를 사용하는 것일까?

자바스크립트가 갖고 있는 생태계 때문일 것이다. Node.js로 서버를 만들고 웹페이지를 만들고 타이타늄이나 리액트 네이티브로 앱을 만들게 될 때 생기는 장점은 바로 하나의 언어로 작성한다는 것이다. 언어가 같다는 말은 DOM, Node나 타이타늄 API등과 같이 특정 플랫폼에 종속되지 않은 Uderscores.js, Moment.js 같은 순수 자바스크립트 라이브러리를 모든 플랫폼 개발에서 사용할 수 있다는 말이다. 또한 디펜던시를 관리한다거나 디버깅을 할 때 사용하는 도구가 거의 유사하다는 장점 또한 무시할 수 없다.

4. 브리지! 자바스크립트로 네이티브를 동작시킨다.

몇 년 전만 하더라도 자바스크립트를 사용하는 네이티브 모바일 앱 개발 프레임워크 분야에 타이타늄 이외에 다른 프레임워크를 찾기 어려웠다. 이 분야에서 타이타늄이 거의 유일한 프레임워크였기 때문에, 자바스크립트로 네이티브 앱을 만든다는 접근 방식이 폰갭과 같은 웹뷰 기반의 패키징 기술과 동일한 것으로 오해를 받기도 하고, 타이타늄 이외에 다른 플랫폼이 없는 이유가 이러한 접근방식에 단점이 많아서가 아니냐는 평들도 있었다. 그러나 2015년 초에 페이스북에서 리액트 네이티브를 발표하면서 자바스크립트로 네이티브를 다루는 플랫폼이 재조명 받기 시작했다. 페이스북은 올해 초 리액트 네이티브 iOS 버전을 발표한데 이어 가을에 Android 버전을 발표했다. 리액트 네이티브는 공개된지 얼마 되지 않아 성숙도 면에서 실제 제품을 만드는데 부족함이 있을 수 있지만 웹 사용자 인터페이스를 만드는 자바스크립트 라이브러리인 리액트(React.js)와 동일한 방식으로 네이티브 앱을 제작 할 수 있는 장점이 있어 많은 개발자들의 관심을 받고 있다.



[그림 1] JS Thread와 UI(Main) Thread사이에 존재하는 Bridge

타이타늄과 리액트 네이티브에서 자바스크립트로 네이티브 객체에 접근하고 이를 다룰 수 있는 것은 그 중간에 존재하는 브릿지가 있기 때문이다. 이는 마치 브라우저에서 자바스크립트로 DOM을 다루는 것과 유사하다. 타이타늄과 리엑트 네이티브에서는 자바스크립트로 네이티브 객체를 다루게 된다. 싱글쓰레드 방식으로 DOM과 자바스크립트가 처리되는 웹 브라우저와 달리 타이타늄과 리액트 네이티트의 자바스크립트 코드는 개별 쓰레드에서 자바스크립트 앤진이 구동되어 UI를 담당하는 메인쓰레드와 별개로 동작한다.

타이타늄은 SDK에서 제공하는 자바스크립트 API를 기준으로 개발자가 자바스크립트 코드를 작성하게 되면 자바스크립트 API와 브릿지로 연결되어 있는 타이타늄 네이티브 SDK의 해당 기능이 구동하는 방식이다.

5. 크로스 플랫폼을 위한 도구

iOS는 오브젝티브-C로 안드로이드는 자바로 작성해야하는 것뿐만 아니라 각 플랫폼에서 제공하는 도구를 따로 익혀야하는 것도 두 가지 플랫폼을 대응하는 어려움 중의 하나이다.

하지만 타이타늄과 리액트 네이티브처럼 크로스 플랫폼을 개발 플랫폼으로 사용하는 경우 사용하는 IDE에 종속되지 않고 자신이 원하는 어떤 에디터에서든 쉽게 개발이 가능하다. 특히 깃허브가 "A hackable text editor for the 21st Century"라고 소개하며 공개한 Atom 에디터는 웹 기술에 기반을 두고 있는 개발자들에게 사랑을 받고 있다. 페이스북의 경우에는 리액트 개발 관련된 기능을 포함하는 아톰용 패키지 세트형태로 nuclide(http://nuclide.io)라는 프로젝트를 공개했다. 타이타늄 진영에서도 개인 개발자들을 중심으로 관련된 패키지들이 만들어지고 있다.

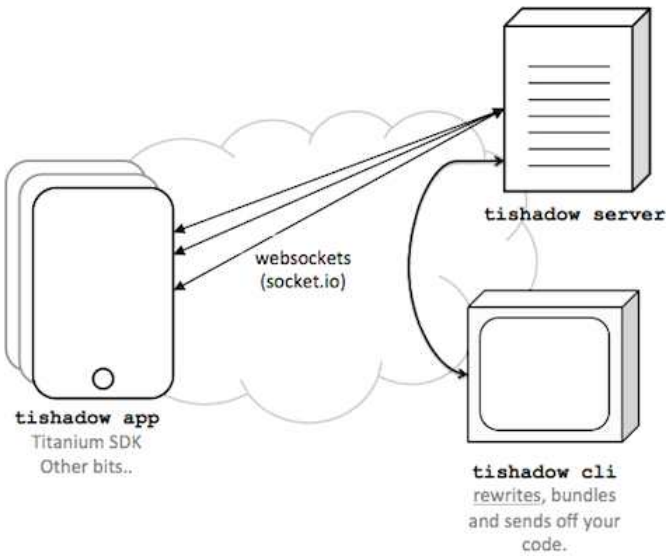
"A hackable text editor"라고 소개된 100% 공개SW인 아톰은 기존과는 완전히 다른 라이프사이클을 갖는다. 기존에는 개발자가 에디터를 사용하다 불편한 점이 있다면 불편한 점을 참고 사용하거나 불편한 점을 해결한 다른 에디터로 갈아타야했다. 적극적인 개발자는 에디터 제작사나 개발자에게 제품 개선을 요구할 수는 있었지만 실제 제품에 반영 될지는 생산자의 몫이였다. 해커블한 제품의 차이는 바로 이 적극적인 개발자에게 에디터를 변화 시킬 수 있는 열쇠를 준다는 것이다. 불편한 사항이 있다면 직접 수정해서 사용가능한 것이다.

스마트폰은 개인 개발자들에게 주어진 최초의 Hackable 폰이라 할 수 있다. 생산자중심의

기획하에만 만들어졌던 스마트폰의 기능이 개인 개발자들이 자신의 구미에 따라 새로운 기능을 앱으로 만들어 내고 있다. Atom은 Hackable 개발 도구이다. 공개SW이면서 개발자들이 많은 부분을 보다 쉽게 손댈 수 있도록 Atom의 주요 기능을 다룰 수 있는 다양한 API를 제공하고 있다. Atom 자체가 개발자들 사이에서 대중적인 자바스크립트로 만들어졌고 API 또한 자바스크립트로 제공되다 보니 아톰을 사용하는 개발자라면 선뜻 해킹해볼 생각이 들 가능성이 높다. 실제로 많은 개발자들이 Atom을 자신들의 구미에 맞게 바꾸는 패키지를 만들고 있다. 그 중에 Autocomplete-plus이라는 패키지는 정식으로 Atom 제품에 포함될 만큼 Atom의 장점주의 하나로 자리 잡았다.

5-1 타이타늄 개발 도구 1 : TiShadow

타이쉐도우(TiShadow)는 타이타늄의 서드파티 개발자 도구이다. 이 도구의 장점은 앱 개발 시에 코드 변경사항을 실제 애플리케이션에 적용하기 위해 다시 빌드할 필요 없이 즉각적으로 앱에 적용하여 확인할 수 있다는 점이다. 네이티브 앱 개발은 웹 개발과 달리 빌드과정을 거쳐야 한다. 하지만 브리지를 이용하여 네이티브 코드를 구동하는 방식인 타이타늄의 경우 변경된 자바스크립트 코드를 앱에게 보내고 해당 코드를 자바스크립트 엔진이 다시 실행해줌으로써 앱의 빌드나 앱의 재시작 없이도 변경한 코드를 반영할 수 있다.



[그림 2] 타이쉐도우 구동 방식.

그림 2는 타이쉐도우의 구동 방식에 나타낸다. 타이쉐도우가 파일 변경을 감지하고 변경된 파일을 Node기반의 서버로 보내고 해당 서버는 소켓으로 연결된 애플리케이션에 변경된 자바스크립트 코드를 전달한다. 타이쉐도우는 2013년 공개되어 타이타늄 개발시 필수 도구로 여겨질 만큼 많은 사랑을 받고 있으며 이와 유사한 타이타늄 서드파티 도구들이 나오기도 했다. 또한 리액트 네이티브 패키저(Packager) 또한 타이쉐도우와 기본 동작 방식이 유사하다.

5-2 타이타늄 개발 도구 2 : 타이타늄 Alloy를 위한 Atom 패키지

앱셀러레이터는 타이타늄 전용 IDE인 Appcelerator Studio를 제공하고 있다. 전용 IDE인 만큼 많은 기능을 갖고 있으나 이클립스 기반이라는 점 때문에 실행속도가 느리고 변경하기 어려운 단점이 있다. 따라서 2015년부터 타이타늄을 위한 아톰 패키지를 개발하여 현재는 v0.11.4 버전이 최신 버전이다 (https://atom.io/packages/titanium-alloy)

이 패키기로 해결하고자 했던 문제는 다음과 같다.

- 반복적 타이핑 : Alloy MVC 프레임워크의 특성상 View에 해당하는 XML에서는 Tag를 사용하고 Style에 해당하는 TSS파일에서는 JSON형태로 스타일을 정의하게 된다. 태그 열고 닫기, JSON Object의 정의형태는 지속적으로 타이핑을 반복하게 된다.
- 수많은 API : 타이타늄은 iOS에서부터 Android, 윈도우폰까지 대응하는 API를 갖고 있다. 공통적인 API는 물로 각 OS의 특장점을 그대로 활용할 수 있는 해당 플랫폼에 제한적인 API들을 제공한다. 이 모든 API를 모두 합치면 그 수는 수천가지에 이른다. 이 모든 API를 외우는 것은 불가능하며 매번 문서 검색을 통해 확인하며 작업하는 것은 쉽지 않은 일이다.
- 오타로 인한 오작동 : 자바스크립트의 단점으로 지적되는 것 중의 하나는 오타를 사전에 검출하지 못하고 런타임에 에러가 발생하거나 심지어 에러가 없더라도 원하는 결과가 나오지 않는 경우가 많다는 것이다. 사전에 컴파일 되지 않고 인터프리터 방식으로 실행되는 구조에서 오는 문제이지만 빌드를 통해 결과를 확인해야하는 모바일 앱개발에서는 오타로 인한 오작동이 발생했을 때 오작동의 원인을 찾는 것에서부터 수정한 후 확인하는 과정이 간단치 않다.
- 여러 파일을 동시에 작업: Alloy MVC의 구조상 하나의 View를 만들 때 View의 구조를 나타내는 XML파일과 스타일에 해당하는 TSS파일, Controller에 해당하는 JS 파일을 번갈아 가며 교차 작업해야 한다.

위에서 언급한 이러한 문제를 아래와 같은 기능으로 해결하려 했다.

- 코드 자동 완성
    - 이 기능은 Atom 기본 패키지인 Autocomplete-plus 에 의해 동작하는 Provider 형태로 구현
- XML 자동완성 : tag 이름 , 현재 태그의 속성 이름, 현재 속성의 값에 대한 자동완성
    - Style 셀렉터에 해당하는 id와 class의 경우 현재 View의Style 파일은 물론 글로벌 스타일인 app.tss 파일까지 참고하여 자동 완성
- 클릭하여 정의부로 이동하기
    - 이 기능은 Facebook이 만든 Hyperclick 패키지에서 동작하는 Provider 형태로 구현
    - XML의 id와 class 지정 이름을 클릭하여 TSS의 스타일 정의부로 이동
    - XML에 등록한 이벤트 핸들러 이름을 클락하여 JS파일에 정의된 핸들러 함수 정의부로 이동
- 관련 파일 열기 및 닫기

- 현재 파일과 관련된 파일 열기 (View, Controller, Style)
- TSS Syntax Highlight

6. 크로스 플랫폼 개발 프레임워크의 개발자 경험

사용자에게 사랑받는 애플리케이션을 만드는 방법을 이야기 할 때 빠지지 않는 것 중에 하나가 바로 사용자 경험(UX)이다. 해당 앱이 어떤 기능을 가지고 있고 얼마나 예쁘게 디자인 되었느냐가 아니라 사용자가 그 앱을 사용하는 경험 자체가 잘 디자인 되었는가가 중요하다. 개발자가 개발 시 사용하는 프레임워크의 성공 여부도 이와 동일하다. 해당 프레임워크가 어떤 기능을 제공하고 얼마나 빠른 성능을 가지고 있는가가 아니라 해당 프레임워크를 사용하여 제품을 만드는 개발 경험이 더욱 중요한 것 같다.
모바일 분야에서 크로스 플랫폼 개발 프레임워크들이 처음 공개되었을 때는 해당 프레임워크를 사용하는 개발 경험이 썩 좋지 않았다. 하지만 벌써 타이타늄 1.0이 나온지 7년이라는 시간이 흘렀고 최근에는 타이타늄과 유사한 접근 방식을 갖고 있는 프레임워크들이 공개될 만큼 이 분야는 많은 발전을 거듭해왔다. 오랜 기간 성숙되어 완성도가 높아진 타이타늄, 아직 성숙하지 않았지만 프레임워크에 개발자 경험을 향상시켜주는 개발 도구들이 내장되어 있는 리액트 네이티브, 이 두 프레임워크가 향후 어떻게 더 발전해나갈지 기대가 된다.