이 글은 [What’s Revolutionary about Flutter](https://hackernoon.com/whats-revolutionary-about-flutter-946915b09514)를 번역한 글입니다. 조금 더 쉽게 읽히는 글을 위해 약간의 의역이 섞여 있습니다.

**Flutter란 무엇인가?**

Flutter 모바일 앱 SDK는 빠르고, 아름다운 모바일 앱을 만들 수 있는 새로운 방법입니다. Flutter는 과거 판에 박힌 앱만 찍어내던 시절로부터 개발자들을 탈출하도록 도와줍니다. Flutter를 사용해본 개발자들은 이 새로운 개발 방식을 정말로 좋아하고 있습니다; 예를 들어, [이 글](https://medium.com/@aubykhan/how-i-built-a-cross-platform-connected-app-in-7-days-93728a987424), [이 글](https://hackernoon.com/flutter-first-steps-fffd906f093b), 혹은 [이 글](https://code.tutsplus.com/tutorials/developing-an-android-app-with-flutter--cms-28270)을 확인해보세요. 또한 [Awesome-Flutter Git 저장소](https://github.com/Solido/awesome-flutter)도 확인해보세요.

모든 새로운 시스템을 만났을때 그렇지만, 사람들은 어떤점이 특별한지 그리고 “Flutter가 가지는 새롭고 흥미로운 점이 어떤것인지?” 궁금해합니다. 이는 당연한 질문이며, 저는 이 글에서 이에 대한 답을 기술적인 관점에서 해보고자 합니다 — 단순히 무엇이 흥미로운지 뿐아니라 왜 그런지도 포함해서요.

잠깐, 간략한 역사 공부 먼저.

**모바일 앱 개발의 간략한 역사**

모바일 앱 개발 분야는 상대적으로 최신 분야라고 할 수 있습니다. 개발자들이 모바일 앱을 개발한지 미처 10년이 되지 않았고, 이를 생각해보면 개발 도구들이 여전히 발전하고 있다는 것은 그다지 놀라운 사실은 아닙니다.

**OEM SDKs**

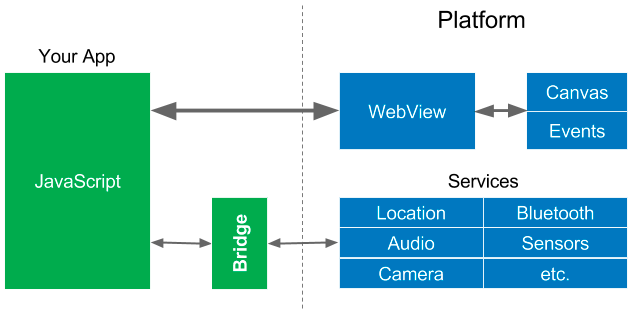
애플은 iOS SDK를 2008년에 릴리즈했고, 구글은 Android SDK를 2009년에 릴리즈 했습니다. 이 두 SDK들은 각기 다른 언어를 기반으로 하고 있죠: Objetive-C, JAVA.



여기서 앱은 위젯을 만들고 카메라와 같은 서비스에 접근하기 위해 플랫폼과 직접 대화를 합니다. 위젯들은 스크린 캔버스에 렌더가 되고, 화면에서 발생된 이벤트들은 다시 해당 위젯으로 보내지게 됩니다. 매우 간단한 구조이죠. 하지만, 각 플랫폼마다 위젯은 다르고 언어 또한 다르기 때문에, 위의 구조에서는 각 플랫폼마다 독립적인 앱을 만들어야합니다.

**WebViews**

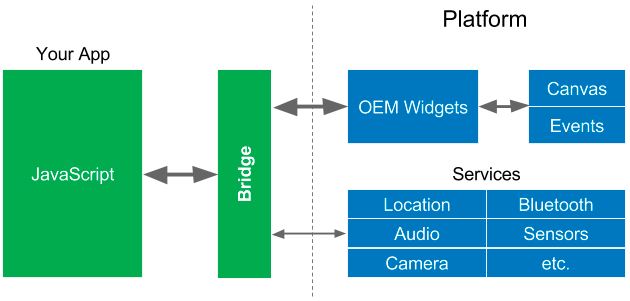
하나의 코드로 여러개의 환경에서 동작하는 크로스 플랫폼 프레임워크를 향한 첫번째 시도는 JavaScript와 WebView를 기반으로 만들어졌습니다. PhoneGap, Apache Cordova, Ionic 등등이 여기에 속합니다. iOS SDK를 릴리즈하기 전, 애플이 아이폰 개발자들에게 웹앱을 만들라고 권장한 사실을 생각해보면, Web 기술을 이용한 크로스 플랫폼의 등장은 어쩌면 당연한 절차였는지 모르겠습니다.



WebView 기반의 앱은 HTML을 만들고 이를 플랫폼 내부의 WebView를 통해 사용자에게 보여주는 방식입니다. 하지만, JavaScript와 같은 언어가 직접적으로 네이티브 코드(위의 서비스와 같은)와 소통하는것은 그리 쉬운 일이아닙니다. 이를 위해 JavaScript 영역과 Native 영역 사이에서 문맥 전환을 해주는 **브릿지**를 사용하게 됩니다. 플랫폼 내부의 서비스들은 일반적으로 자주 불리지는 않기 때문에, 이런 방식의 구현이 엄청난 퍼포먼스 문제로 이어지지는 않았습니다.

**리액티브 뷰**

[리액티브 프로그래밍](https://en.wikipedia.org/wiki/Reactive_programming) 패턴을 이용한 [ReactJS](https://facebook.github.io/react/" \t "_blank)와 같은 리액티브 웹 프레임워크는 이전보다 웹뷰 생성을 매우 쉽게 만들어주었다는 점에서 큰 인기를 얻게 되었습니다. 2015년에 공개된 React Native는 리액티브 스타일의 뷰들이 가지고 있는 다양한 장점들을 모바일 앱에 적용하기 위해 만들어졌습니다.



React Native는 현재 엄청난 인기를 누리고 있지만, JavaScript 영역의 코드가 네이티브 영역에 있는 OEM 위젯에 접속을 하기위해서 [결국 **브릿지**를 거치게 됩니다](https://medium.com/@talkol/performance-limitations-of-react-native-and-how-to-overcome-them-947630d7f440). 리액티브 뷰에서는 OEM 위젯에 굉장히 자주 접근하게 되고 (사용자가 화면에서 스와이프 한번을 했을때 최대 초당 60번까지도 접근), 이는 결국 퍼포먼스 이슈를 야기하게 됩니다. React Native에 대한 [한 글](https://medium.com/@talkol/performance-limitations-of-react-native-and-how-to-overcome-them-947630d7f440)에서는 이에 대해 다음과 같이 이야기하고 있습니다.

*React Native의 성능을 이해하는 주된 열쇠가 여기에 있다. 각 영역 그 자체는 엄청나게 빠르다. 성능의 병목은 주로 하나의 영역에서 다른 영역으로 넘어갈때 발생한다. 고성능의 React Native 앱을 구축하기 위해서는, 이 브릿지를 건너는 횟수를 최소화해야한다.*

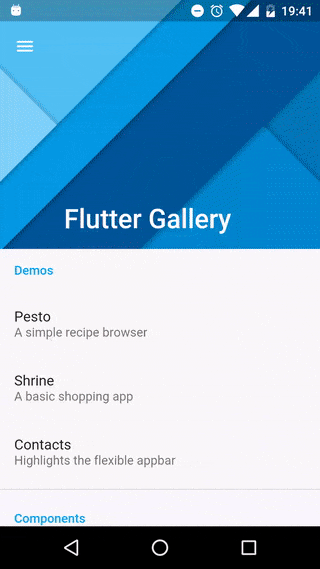
**Flutter**

Flutter는 React Native와 마찬가지로 리액티브 스타일 뷰를 제공하고 있습니다. 하지만, Flutter는 [Dart](https://www.dartlang.org/)라는 컴파일 프로그래밍 언어를 이용해 이 브릿지로 인해 발생하는 성능 문제를 피하고 있습니다. Dart는 Ahead of time (AOT)에 여러 플랫폼에 상응하는 네이티브 코드로 컴파일 됩니다. 다시말해, Flutter는 문맥 교환을 하는 JavaScript 브릿지를 거치지 않고 플랫폼과 직접적으로 커뮤니케이션하는 것입니다. 이처럼 네이티브 코드로 컴파일이 되면 앱의 스타트업 시간 또한 빨라지게 됩니다.

Flutter가 JavaScript 브릿지를 필요로 하지 않으면서, 리액티브 뷰를 제공하는 유일한 모바일 SDK라는 사실만으로도 Flutter는 이미 충분히 흥미롭고 시도해볼만한 가치를 가집니다. 하지만, 여기에 그치지 않고 Flutter는 훨씬 더 혁신적인 부분들을 제공하고 있습니다.

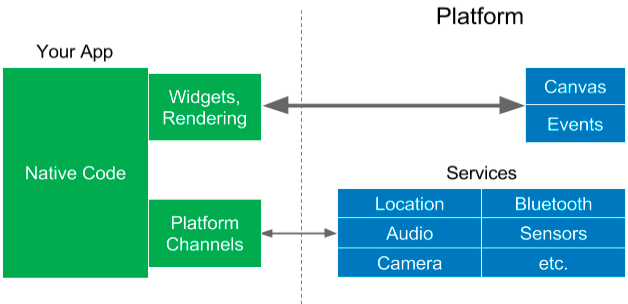
**위젯**

위젯은 앱의 뷰와 인터페이스를 컨트롤하는 엘리먼트라고 할 수 있습니다. 위젯은 모바일 앱에서 가장 중요한 요소라고 해도 과언이 아닐 겁니다. 사실상, 위젯만으로 앱의 성패가 갈릴 수 있습니다.



* 위젯에서 룩앤필은 다른 어떠한 것보다도 중요합니다. 위젯은 다양한 스크린 사이즈에서 보기 좋아야하고 또 자연스러워 보여야 합니다.
* 위젯은 반드시 빠르게 동작해야합니다: 위젯 트리를 만들때, 위젯들을 인플레이트할때, 화면에 배치시킬때, 렌더링할때, (특히) 애니메이션 효과를 줄때 등을 포함해서 말이죠.
* 요즘 앱에서 위젯은 확장가능해야하며 커스터마이징이 가능해야합니다. 개발자들은 마음에 드는 새로운 위젯들을 추가할 수 있기를 바라고, 앱의 브랜드와 매칭되도록 이들을 커스터마이징 하기를 바랍니다.

Flutter는 위에서 언급한 요구사항들 즉, 좋은 룩앤필, 빠른 속도, 커스터마이즈, 확장가능성을 지원하는 위젯을 제공하는 새로운 아키텍쳐를 가지고 있습니다. 맞습니다! 바로 Flutter가 OEM 위젯들 (또는 DOM 웹뷰)을 사용하지 않고, 자체적인 위젯을 사용하고 있기 때문입니다.



Flutter는 위젯들과 렌더러들을 플랫폼 영역에서 앱의 영역으로 옮겨왔습니다. 이것은 이제 위젯들이 커스터마이징 가능해졌고 확장가능해졌다는 것을 의미합니다. Flutter가 플랫폼으로부터 요구하는 것은 위젯을 렌더할 수 있는 캔버스뿐 입니다. 이 캔버스를 통해 디바이스 스크린에 위젯을 그리고, 이벤트들(터치, 타이머 등)과 서비스들(위치, 카메라 등)에 접근하게 됩니다.

Dart 프로그램 (초록색) 영역과 네이티브 플랫폼 코드 (파란색) 사이에는 인코딩과 디코딩을 담당하는 인터페이스가 여전히 존재하고 있습니다. 그러나, JavaScript 브릿지와 비교했을때, 비교할 수 없을 정도로 빠른 성능을 가지고 있습니다.

위젯들과 렌더러를 앱으로 옮겨온것은 앱 사이즈에도 영향을 미칩니다. Flutter 앱의 최소 사이즈는 약 6.7MB이며, 이것은 비슷한 개발 도구들을 이용해서 만든 최소 크기의 앱들과 비교했을 때 비슷한 사이즈입니다. Flutter의 장점들이 다른 개발 환경에 비해 가치가 있는지 없는지 판단하는 것은 여러분들의 몫입니다. 이 글의 남은 부분에서는 Flutter가 가지고 있는 더 많은 장점들에 대해서 다뤄보고자 합니다.

**레이아웃**

Flutter에서 가장 큰 개선점중 하나는 레이아웃을 하는 방식에 있습니다. 여기서 레이아웃이란 정해진 규칙(혹은 제한들)에 따라서 위젯의 크기와 위치를 정하는 것을 의미합니다.

전통적으로, 레이아웃은 어떠한 위젯에도 적용할 수 있는 규칙들의 거대한 집합을 사용해왔습니다. 여기서 규칙들은 다양한 레이아웃 함수들을 구현합니다. 가장 잘 알려져있는 CSS 레이아웃을 예로 들어 봅시다. CSS는 속성들(규칙들)을 가지고 있고, 이 속성들은 HTML 엘리먼트들 (위젯들)에 적용이 됩니다 (CSS3는 375개의 속성을 정의하고 있습니다).

CSS는 박스 모델, 플로팅 엘리먼트, 테이블, 텍스트 컬럼, 페이지 미디어 등등을 포함하는 수 많은 레이아웃 모델을 포함하고 있습니다. 시간이 지나서 개발자들과 디자이너들은 레이아웃에 대해 더 많은 부분을 컨트롤 하고 싶어했고, 이 필요에 따라서 Flex Box와 Grid Layout 모델들을 추가해야만 했습니다. 전통적인 레이아웃에서 새로운 모델들을 추가할때, 개발자들이 자체적으로 추가하는 것은 불가능합니다. 이 때문에, Flex Box와 Grid Layout은 우리가 알고 있듯이 CSS 형태로 추가되어야만했습니다. 다시 말해 모든 브라우저들은 이에 대한 구현을 다시 해야만 했습니다.

전통적인 레이아웃이 가지고 있는 또 다른 문제로는 규칙들이 서로 상호작용 할 수 있으며 (혹은 속성들간의 Conflict가 발생), 보통 하나의 엘리먼트에 수십개의 규칙들이 한꺼번에 적용된다는 것입니다. 이러한 환경은 레이아웃을 시간을 느리게 만듭니다. 레이아웃 성능은 일반적으로 N 제곱의 복잡도를 따릅니다. 다시 말해, 엘리먼트의 숫자가 증가함에따라, 레이아웃 속도는 훨씬 더 빠르게 느려지게 됩니다.

이를 극복하기 위해, 구글 크롬 브라우저 팀 멤버들은 새로운 실험을 시작했습니다. 그들은 만약 그들이 전통적인 레이아웃 모델을 머릿속에 완전히 지워버린다면, 훨씬 더 빠른 렌더러를 만들 수 있는지 궁금했습니다. 몇주가 지나고 그들은 눈에 띄눈 성능 향상을 얻을 수 있었습니다. 이들이 발견한것은:

* 대부분의 레이아웃은 상대적으로 간단하다는 점. 스크롤 페이지에 있는 텍스트, 디스플레이 사이즈에 따라 위치와 크기가 변하는 직사각형, 그리고 테이블들, 플로팅 엘리먼트들 정도만 필요하다는 점.
* 대부분의 레이아웃은 위젯의 서브트리에 속해있으며, 이 서브트리는 보통 하나의 레이아웃 모델만을 사용한다는 점. 다시 말해, 하나의 특정 위젯을 지원하기 위해 아주 적은 숫자의 규칙들만이 필요하다는 점.

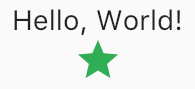
그들은 관점을 완전히 바꾸면 레이아웃을 훨씬 더 간단하게 만들 수 있다는 것을 깨달았습니다.

* 어떠한 위젯에도 적용될 수 있는 엄청나게 큰 규모의 레이아웃 규칙들 대신, 각 위젯이 본인만이 가지고 있는 고유의 간단한 레이아웃 모델을 가지면 된다.
* 이제 각 위젯은 고려해야하는 레이아웃 규칙이 굉장히 작아지기 때문에, 각 위젯을 위한 고도의 최적화를 할 수 있게 된다.
* 모든 것들을 위젯화 시키면, 더욱 간단한 레이아웃을 구조를 만들 수 있다.

이제 Flutter를 이용해 간단한 위젯 트리를 만드는 코드를 살펴봅시다:

new Center(  
 child: new Column(  
 children: [  
 new Text('Hello, World!')),  
 new Icon(Icons.star, color: Colors.green)  
 ]  
 )  
)

이 코드의 구문에 이미 어떤 결과물이 만들어질지 충분한 의미를 가지고 있습니다. 이 코드를 통해 얻어지는 결과는 아래와 같습니다.



이 코드에서 레이아웃을 포함한 [모든것은 위젯입니다](https://flutter.io/widgets/). **Center 위젯**은 위젯의 부모(위의 예제에서는 스크린)의 중앙에 위젯의 자식 위젯들을 위치시킵니다. **Column 위젯**은 이 위젯의 자식 위젯들을 수직으로 나열 시킵니다. 이 Column은 **Text 위젯**과 **Icon 위젯**(색 속성을 가지는) 을 가지고 있습니다.

Flutter에서 가운데 정렬이나 패딩 역시도 위젯입니다. 심지어 테마들까지도 자식 위젯들에게 적용되는 또다른 위젯이며, 어플리케이션과 네비게이션까지도 위젯입니다.

Flutter는 Column뿐 아니라 Row, Grid, List 등 꽤나 많은 위젯을 포함하고 있습니다. 게다가, Flutter는 Silver Layout Model이라고 칭하는 스크롤링을 위해 사용되는 고유한 레이아웃 모델을 가지고 있습니다. 스크롤링은 반드시 즉각적이면서도 부드럽게 반응해야만 사용자들이 물리적인 화면을 드래그했을 때, 손끝에 이미지가 붙어있는 듯한 느낌을 줍니다. Flutter의 레이아웃은 굉장히 빨라서 바로 이것을 해낼 수 있습니다.

스크롤링을 위해 레이아웃을 사용함으로써 Flutter는 아래 이미지와 같은 발전된 형태의 스크롤링을 가능하게 합니다. 아래 이미지가 GIF 이미지라는 점을 잊지 마세요. Flutter는 이보다 더 부드럽습니다! 이 글의 마지막에 있는 리소스에서 해당 앱의 코드를 찾아 돌려볼 수 있습니다. 꼭 한번 돌려보세요!

대부분의 경우 Flutter는 한번의 패스로 레이아웃을 구축 할 수 있습니다. 다시 말해, 리니어 타임안에 끝낼 수 있는거죠. 이 때문에 Flutter는 많은 수의 위젯들을 한꺼번에 다루는 것도 가능합니다. 여기에 추가로 Flutter는 매번 레이아웃 작업을 하지 않기위해, 캐쉬를 비롯한 다양한 작업들을 수행합니다.

**커스텀 디자인**

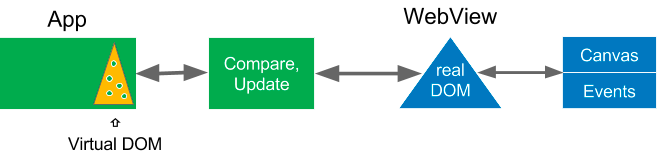
Flutter는 위젯을 앱의 일부로 옮겨왔기 때문에, 새로운 위젯을 추가하는 것 혹은 기존의 위젯들에 다른 룩앤필을 주거나 회사 브랜드에 맞도록 커스터마이징하는 것이 가능합니다. 현대의 모바일 디자인 트렌드는 판에 박힌 앱을 만드는 것에서 완전히 멀어져 있으며, 사용자에게 기쁨을 주는 커스텀 디자인을 향해 점점 더 발전해 나아가고 있습니다.

Flutter는 풍부하고, 커스터마이즈 가능한 Android, [iOS](https://flutter.io/widgets/cupertino/)그리고 [Material 디자인](https://material.io/)을 가지고 있습니다 (사실, Flutter가 가장 높은 수준의 Material 디자인을 제공하고 있다는 이야기가 종종 들립니다). Flutter는 커스터마이즈 능력을 활용해서 멀티 플랫폼에 있는 네이티브 위젯들의 룩앤필과 매칭되는 위젯들을 지원하고 있습니다. 앱 개발자들 역시 이 커스터마이즈 능력을 사용해서 자신들의 요구와 필요에 맞도록 Flutter에서 제공하는 위젯을 변형시키는 것이 가능합니다.

**리액티트 뷰에 관한 더 많은 이야기**

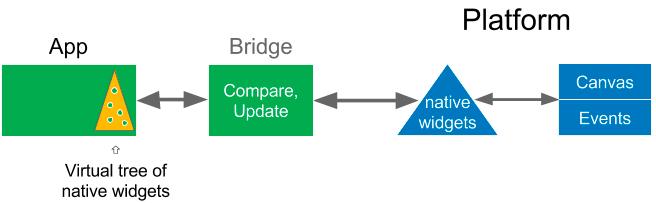
리액티브 웹 뷰를 위한 라이브러리들은 **Virtual DOM** 개념을 새롭게 소개했습니다. DOM은 HTML Document Object Model을 의미하며, DOM의 API는 JavaScript를 이용해 엘리먼트의 트리로 표현되는 HTML document들을 다루는데 사용됩니다. Virtual DOM은 DOM의 추상화된 버전으로 프로그래밍 언어 (여기에서는 JavaScript) 내부의 객체들을 이용해서 DOM을 생성하는 방식을 의미합니다.

리액티브 웹 뷰(ReatJS와 같은 시스템에 의해서 구축된)에서 이 Virtual DOM은 변경이 불가능(immutable)하며 어떤 변화를 위해서도 매번 처음부터 다시 구축이 됩니다. Virtual DOM은 실제 DOM과 비교를 통해, 최소한의 변경사항들을 찾아내고 이를 기반으로 실제 DOM을 업데이트합니다. 실제 DOM이 업데이트되고 나서야, 플랫폼은 실제 DOM을 다시 렌더링하고 캔버스에 그리게 됩니다.



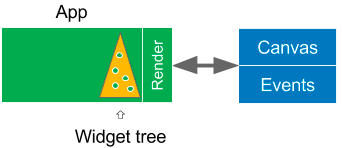
이러한 방식은 듣기에 과도한 추가 작업이 있는것처럼 들리지만, [HTML DOM을 수정하는것이 매우 비싸다는 것](https://www.codecademy.com/articles/react-virtual-dom)을 생각한다면 충분한 가치를 지니고 있습니다.

React Native는 위와 비슷한 작업을 모바일 앱을 대상으로 수행합니다. DOM 대신 모바일 플랫폼에 있는 네이티브 위젯들을 조종하는겁니다. React Native는 Virtual DOM 대신 위젯들의 Virtual tree를 만들고 이를 네이티브 위젯과 비교해서 필요한 부분에 대해서만 네이티브 위젯에 변경을 가하는 방식을 사용합니다.



React Native가 네이티브 위젯과 소통하기 위해서는 반드시 브릿지를 거쳐야한다는 점을 기억해야합니다. 다시 말해, React Native가 만드는 Virtual tree는 네이티브 위젯의 사용을 가능하게함과 동시에, 브릿지를 건너는 횟수를 최소화하는 역할을 수행하는 것입니다. 최종적으로 네이티브 위젯이 업데이트 되고나면, 플랫폼은 캔버스에 변경된 위젯을 렌더링하게 됩니다.

모바일 개발에 있어서 React Native는 큰 성취를 이루어냈고 이는 Flutter에게 영감을 주었습니다. 하지만, Flutter는 여기에 그치지 않고 한 스텝 더 나아갑니다.



Flutter에서는 위젯과 렌더러 모두가 플랫폼에서 사용자의 앱상으로 옮겨졌다는 점을 기억하세요. 이제 더 이상 변경 사항을 전달 해야하는 네이티브 위젯은 존재하지 않습니다. Virtual 위젯 트리였던 부분이 이제 진짜 위젯 트리가 되는거죠. Flutter가 위젯트리를 렌더하고 플랫폼의 캔버스에 페인팅을 합니다. 굉장히 좋고 간단하죠 (그리고 빠르고요). 추가로, 애니메이션이 사용자 영역에서 발생하기 때문에, 앱 (그러니까 개발자들)은 애니메이션에 대한 더 다양한 컨트롤을 가지고 있습니다.

Flutter 렌더러는 그 자체만으로도 흥미롭습니다: 이 렌더러는 스크린상에서 업데이트되어야하는 위젯들만을 렌더하기 위해서 내부적으로 몇개의 트리 구조들을 가지고 있습니다. 예를 들어, 렌더러는 “[structural repainting using compositing](https://youtu.be/UUfXWzp0-DU)” 이라는 방법을 사용하고 있습니다 (여기에서 structural는 스크린상의 직사각형 영역을 사용하는 것보다 훨씬 더 효과적인 위젯을 의미합니다). 변경사항이 없는 위젯들에 대해서는, 심지어 위치가 이동했다고 하더라도, 캐쉬로부터 “[bit blit](https://en.wikipedia.org/wiki/Bit_blit)”이 되기 때문에 굉장히 빠릅니다. 이것이 Flutter에서 스크롤링이 엄청난 퍼포먼스를 보이는 이유입니다.

Flutter 렌더러에 대해서 더 자세히 살펴보고 싶다면 [이 비디오](https://youtu.be/UUfXWzp0-DU)를 추천합니다. 더불어, [Flutter는 오픈소스](https://github.com/flutter)이기 때문에 코드를 직접 살펴보는 것도 가능합니다. 이는 렌더러, 컴포지터, 애니메이션, 제스처 인식 등 모든 위젯을 포함해서 전체 스택을 마음대로 커스터마이즈하는것이 가능하다는 의미입니다.

**Dart 프로그래밍 언어**

리액티브 뷰를 사용하는 다른 시스템들처럼 Flutter는 새로운 프레임마다 새로운 뷰 트리를 구축합니다. 다시말해, 하나의 프레임을 위해 많은 수의 객체들을 생성하게 됩니다. 다행히도, Dart는 이러한 시스템에서 굉장히 효율적인 “[generational garbage collection](https://en.wikipedia.org/wiki/Tracing_garbage_collection#Generational_GC_.28ephemeral_GC.29)” 을 사용하고 있습니다. 이 정책에서는 객체들 (특히 짧은 시간동안 살아있는 객체들)의 비용이 매우 쌉니다. 추가적으로, 객체 할당이 빠르며 락을 요구하지도 않는 싱글 포인터 범프로 가능합니다. 이는 UI [jank](https://en.wiktionary.org/wiki/jank" \t "_blank)와 stutter을 피하는것을 도와줍니다.

또한 Dart는 앱에서 필요한 코드만을 포함시키는 “[tree shaking](https://medium.com/@Rich_Harris/tree-shaking-versus-dead-code-elimination-d3765df85c80)”라는 컴파일러를 가지고 있습니다. 아주 큰 위젯 라이브러리에서 한개 혹은 두개만을 사용하더라도 부담없이 사용하면 됩니다.

Dart에 대한 더 많은 정보를 얻고 싶다면 “[Why Flutter uses Dart](https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf)”를 읽어보세요 (다음 번역글로 계획하고 있습니다).

**Hot reload**

Flutter에서 가장 유명한 특징은 빠르고, stateful한 Hot reload 기능입니다. 앱이 동작하고 있는 동안 Flutter 앱에 변경을 줄 수 있습니다. Flutter는 보통 일초안에 이전 코드 상태를 유지하면서 변경된 부분을 적용합니다. 만약 에러가 발생하면, 개발자는 해당 에러를 수정하고 Flutter는 마치 에러가 없었던것처럼 그 이전 상태를 유지하며 동작합니다. 심지어 앱 전체를 다시 로딩할 때조차도 굉장히 빠릅니다.

이 과정을 개발자들은 “앱을 페인팅한다”고 표현합니다. 한 시점에 하나의 변화를 만들고 앱을 재시작할 필요없이 그 결과를 바로 바로 확인하기 때문입니다.

**Compatibility**

위젯들과 렌더러들은 플랫폼의 일부가 아니고 앱의 일부이기 때문에 더 이상 “[compat librar](https://developer.android.com/topic/libraries/support-library/index.html" \t "_blank)y”들이 필요하지 않습니다. Flutter로 만든 앱은 오래된 OS 뿐만 아니라 최신 OS에서도 동일하게 동작합니다 (Android Jelly Bean부터 최신, iOS 8.0부터 최신). 이것은 예전 OS 버전을 테스트해야하는 공수를 엄청나게 줄여줍니다. 더불어, 미래의 OS 버전에서도 동일하게 동작할것입니다.

우리가 사람들로부터 자주 듣는 걱정이 하나 있습니다. Flutter가 OEM 네이티브 위젯들을 사용하지 않기 때문에, 새로운 종류의 위젯을 가지고 있거나 기존의 위젯의 모양과 행동의 변화를 포함하는 새로운 iOS 혹은 Android 버전이 나오면, 이를 서포트하는데까지 시간이 너무 오래걸리지 않겠느냐하는 걱정입니다.

* 먼저, 구글은 Flutter의 큰 내부 사용자입니다. 그렇기에, 현재의 OEM 위젯들과 가능하면 가깝게 위젯의 셋들을 유지하는데 큰 이점을 가지고 있습니다.
* 위젯 업데이트가 너무 느린 경우에도 구글만이 Flutter의 위젯을 업데이트하는 권한을 가지고 있지 않다는 점을 기억하세요. Flutter의 위젯은 굉장히 확장가능하고 커스텀이 가능하기 때문에, 당신을 포함한 누구든지 업데이트를 할 수 있습니다. Pull Request를 쌓을 필요도 없으며, Flutter의 자체적인 업데이트를 기다릴 필요도 없습니다.
* 또한 위의 이야기들은 여러분의 앱에 새로운 변화를 주었을때에만 해당됩니다. 만약, 여러분의 앱의 디자인이나 행동에 아무런 변화도 주지 않았다면, 전혀 문제가 없습니다. 위젯은 앱의 일부이기 때문에, 플랫폼의 업데이트로 인해 앱이 갑자기 이상하게 보이거나 하는 문제가 절대 발생하지 않습니다.
* 추가적인 장점으로 여러분이 새롭게 만든 위젯 역시 예전 OS 버전에서도 잘 동작한다는 점이 있습니다.

**또 다른 이점들**

Flutter의 간결성은 앱을 빠르게 만들지만, 앱 전반에 걸쳐 존재하는 커스터마이징 능력과 확장가능성은 Flutter를 더욱 강력하게 만듭니다.

Dart의 [소프트웨어 패키지 저장소](https://pub.dartlang.org/flutter/packages/)를 이용해 여러분의 앱이 가지는 능력을 확장시킬 수 있습니다. 예를 들어, 다수 패키지들이[Firebase에 접근](https://github.com/flutter/plugins/blob/master/FlutterFire.md)하는 것을 쉽게 만들어주기 때문에 Flutter를 이용해 Serverless앱을 만드는 것은 매우 간단합니다. 또한 어떤 개발자는 [Redux data store에 접근](https://pub.dartlang.org/packages/flutter_built_redux)할 수 있는 패키지도 만들어 두었습니다. [플러그인](https://flutter.io/developing-packages/)이라고 불리는 패키지들이 존재하는데, 이들은 OS와 상관없이 앱에서 가속도센서 카메라등 플랫폼의 서비스와 하드웨어에 접근할 수 있는 방법을 제공하고 있습니다.

물론, Flutter 자체도 오픈소스입니다. Flutter의 렌더링 스택도 앱의 일부라는 의미이고, 각 앱에서 여러분이 원하는 거의 모든 부분을 커스터마이즈 할 수 있다는 것을 의미합니다. 아래 그림에서 초록색으로 표시된 모든 부분이 전부 커스터마이징 가능합니다.

https://miro.medium.com/max/60/1*AMb5JzTXKsR8_PKdEtDfLQ.png?q=20

**그래서, Flutter는 무엇이 새롭고 흥미로운가?**

이제 누군가가 이런 질문을 여러분에게 한다면 어떻게 대답해야하는지 아실 겁니다.

* JavaScript 브릿지 없는 리액티브 뷰의 장점들
* 빠르고, 부드럽고, 예측가능한 AOT 에서 Native 코드로 컴파일되는 언어
* 위젯과 레이아웃에 모든 접근이 가능한 점
* 아름답고 커스터마이징이 가능한 위젯들
* Hot reload를 포함한 최고의 개발 환경
* 성능이 좋고, 호환성이 뛰어나며, 재미있음!

제가 위의 리스트에서 어떤것을 빼먹었는지 눈치채셨나요? 이 부분은 사람들이 Flutter에 대해서 이야기할때 가장 먼저 이야기하는 장점이지만, 저에게는 가장 덜 흥미로운 점입니다.

바로 Flutter가 하나의 코드로 멀티 플랫폼에서 동작하는 앱을 만들 수 있다는 점입니다. 물론, 이 부분도 응답에 포함시켜야겠죠! 하지만, 성능을 포기하지 않고 멀티 플랫폼에서 Flutter가 동작할 수 있도록 해준점은 강력한 커스터마이징 능력과 확장성이라는 점을 잊지 마세요.

**혁명적인**

저는 아직도 Flutter가 왜 혁명적인지에 대해서 완전히 설명하지는 못했습니다. [Flutter를 이용해 외부에서 개발된 첫번째 앱](https://blog.goposse.com/a-revolutionary-app-for-a-revolutionary-show-1b687f24023b)이 미국 독립 전쟁 (American Revolutionary War)를 배경으로한 “[Hamilton: An American Musical](http://www.hamiltonbroadway.com/)” 공식 앱이라는 점을 생각해보면, 어느 정도 맞아 떨어져가는것처럼 보입니다. Hamilton은 역사상 가장 유명한 브로드웨이 뮤지컬 중 하나입니다.

https://miro.medium.com/max/60/1*51ayq5t6n5Ea3bpPJS-XvA.png?q=20

개발자 [Posse](http://goposse.com/)는 앱을 단지 세달만에 만들어야했기 때문에 Flutter를 선택했다고 이야기합니다. 그들은 이를 두고 “혁명적인 쇼를 위한 혁명적인 앱”이라고 이야기하고 “Flutter는 아름답고, 높은 성능을 가지는, 브랜드 지향 모바일 경험을 제공하기 위한 최고의 선택이다”고 이야기합니다. 이들은 Google Developer Days 컨퍼런스에서 Flutter를 이용해 그들이 [앱을 만들었던 경험을 공유](https://youtu.be/prlK_QL_qOA)했습니다. 이 앱은 [Android](https://play.google.com/store/apps/details?id=com.hamilton.app), [iOS](https://itunes.apple.com/us/app/hamilton-the-official-app/id1255231054)에서 사용 가능하며 사용자들로부터 [좋은 리뷰들](http://www.cnn.com/2017/08/14/entertainment/hamilton-app/index.html)을 받았습니다.

**혁명에 동참하세요!**

Flutter는 현재 베타 릴리즈 상태입니다. 우리는 계속해서 새로운 기능들을 추가하고 있고 더 많은 최적화 계획을 가지고 있습니다. 하지만, 구글의 내부와 외부에 존재하는 그룹들에서 이미 미션 크리티컬한 앱을 만드는데 Flutter를 사용하고 있습니다.

만약 여러분이 Flutter에 관심이 있다면, [지금 바로 설치](https://flutter.io/setup/)해보시고 같이 설치된 [앱을 가지고 즐겨보세요](https://flutter.io/getting-started/). 특히 stateful hot reload를 확인하는것을 잊지 마세요.

만약 여러분이 개발자가 아니거나 단지 몇몇개의 앱을 찾고 있다면, Flutter를 이용해 개발된 앱들을 설치할 수도 있고 어떻게 그들이 생겼는지 그리고 동작하는지 확인할 수도 있습니다. 저는 Hamilton app을 추천하지만, 이 외에도 다른 앱이 많이 존재합니다. Google I/O에서 Flutter 앱을 [라이브 코딩하는 이 영상](https://goo.gl/csEr1f)도 꼭 살펴보세요.