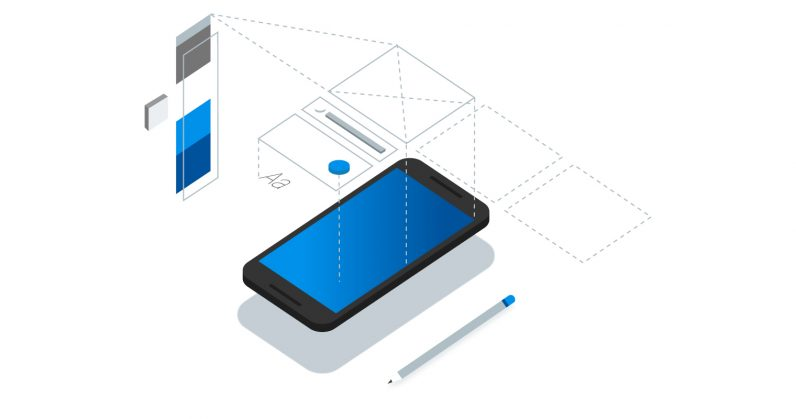
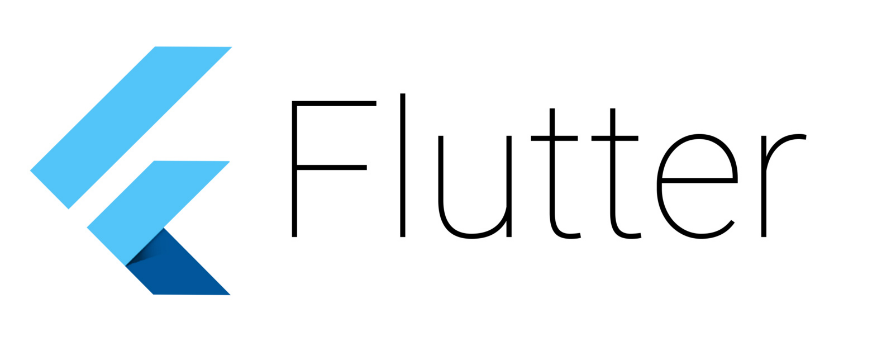
**Flutter and Dart: Changing How Native Mobile Applications Are Built**

**Cameron Stark**

CS 332 – Report 1

Embry-Riddle Aeronautical University

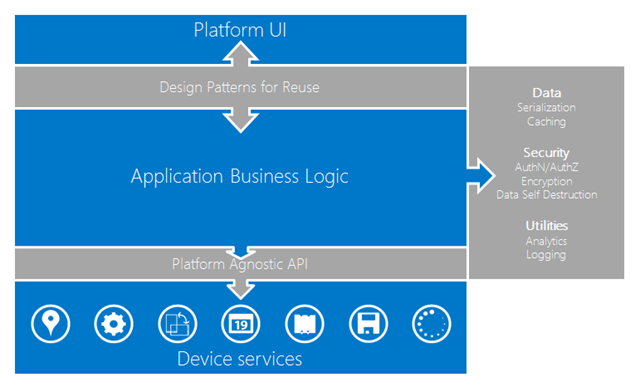
December 2018



*Figure 1 [1] Flutter Logo*

*Figure 2 [2] Flutter Mobile Design Image*

**Introduction:**

 Ever since the beginning of the mobile device market, those who have wanted to get their product onto the various different devices ranging from iPhones, Android phones, blackberries, palm pilots, and many others, if they wanted to gain access to the internal components of the device such as camera, location, microphone and other components, would have to build their app in each platforms native language in order to get the full usability out of their app. For example, iPhones use the language swift and Android uses Java, just to list to the two major markets in the mobile industry. iOS is the operating system by Apple for only the iPhone and Android is the operating by Google for a multitude of phones by Google, Samsung, Essential, OnePlus, and many more. Companies would need to have to hire teams to develop a Swift application and a Java application, in order to reach both markets, or a single team make a progressive web application (which is a website designed to work on desktop and mobile devices), but not be able make full use of the power of the mobile device. This problem prompted solutions like Kotlin and Xamarin, who were both among the first major attempts at cross-platform development, where they both followed the same format of a native UI and a shared business logic in a specific language that would be compiled to the respected platform.

*Figure 3 [6] Breakdown of a Xamarin App*

This report will briefly discuss the cross-platform native compiler and then delve into Dart and Flutter, describing the language and framework respectively by breaking down the paradigm and how the language makes use of it, how the language and framework perform common functions such as declaration just to name one, and why this language and thusly the framework was created and developed.

**The Flutter Cross-Platform Native Complier**

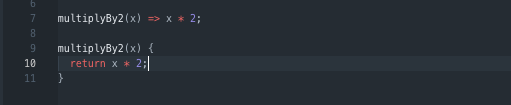
To begin with the idea of Cross-Platform is writing the application in a single code base which is the compiled into the native code for each of the targeted platforms, being iOS and Android [3]. Other methods of being able to produce visual components on multiple devices is making use of web views, which is essentially a website handled inside of a window inside the application, or using the OEM’s widgets, which refers to the original manufacturer. Whereas the Flutter Native Compiler uses its own visual components and rendering engine, which allows the framework to create applications that are as powerful as pure native applications. This is made possible with Ahead of Time compilation (AOT) which takes any of the C/C++ code and compiles it into Native Development Kit (NDK) for Android and a Low-Level Virtual Machine (LLVM) for iOS and any of the Dart code is compiled into the respected native code for each platform [3]. The NDK is the base for the compilation process of Java into android applications [4]. The LLVM provides a middle layer for the compiler, in the case of Apple and iOS works to support Objective-C and Swift in the project life cycle [5]. The creation of compilers to take code of one language and create code of another language allows a single team to create applications for two different platforms in record time, thus allowing the company to get their product on the market at a much lower cost than a company that had two separate teams working on each platform. Flutter and Dart also has its own Dart VM (Virtual Machine) which allows for hot reloading, which is the live reloading and recompiling of the application without losing the current state and other values that be present, thusly increasing the efficiency of the development process.

**The Dart Programming Language:**

Dart was developed and is maintained by Google, originally as a successor and replacement to ECMAScript 7 also known as JavaScript, thus it implements many of the important characteristics such as async and await, when dealing with multiple or stringed processes [3]. Dart is however structured and built as an object-oriented language, designed to have a similar syntax to that of Java, as to attract more developers [3]. Flutter which is built on top of Dart, makes use of reactive views which with every view (where a view is the current visible user interface, or in the case of flutter a series of widgets) refreshes the view tree, which in some cases can create objects that are only used in a single case, where Dart is able to optimize the memory usage of “Generational Garbage Collector”, which deletes objects or items that are no longer used or referenced [3]. While Dart may be an Object-Oriented language, it is not in the conventional sense of the meaning, as in Java or C# where classes and inheritance are a major factor of them, Dart is in the sense that for an object the only thing that matters are the observable behaviors [7]. Dart unlike its fellow OO (Object-Oriented) languages, is optionally typed, which means that it is not a syntactic requirement for the program to have defined types, whereas Java every variable must be defined as a type, where a type is a String, Int (integer number), Boolean (true, false), Char (single character) [7]. Dart at runtime treats everything from elementary data such as numbers to the most complex things as objects and tries to adhere to some of the most basic principles of OO programming, where the language treats types as interfaces not classes and can thusly mean that any class can be used as interface into another class [7]. Dart declaration is similar to that of other contemporary languages where the variable name is to the left of the assignment operator “=” and the value is to the right.

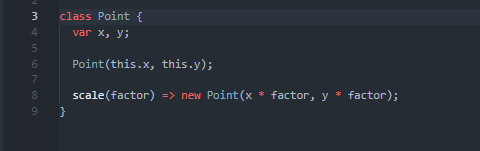


**1** Shows simple variable declaration

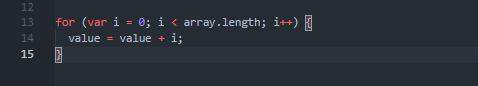
Since the language is syntactically type independent, the variable can simply be defined as a var, which tells the compiler that it is a variable and at runtime will be determined the type based on what is assigned to the input. Similar to Java the function definition the Dart function definition follows the form of function name, and parameters followed by block of code to run when the function is called [7].

2 Shows two types of function creation

Class creation and defining is similar to that of java with attributes and functions associated with the class [7].

****

3 Sample of a basic class declaration with a constructor

Dart again deals with looping with the same format as most other languages and continues to follow the formatting and syntax of Java with the for loops, where the starting condition, ending condition and iterator are required in the definition and the code that will be ran during the loop put in the following lines.

4 Sample code showing a simple for loop

**Flutter and Dart in Conclusion**

Flutter and Dart were created in response to the mobile and web environment needing or wanting alternative ways of performing tasks, that is efficient, powerful and fast. Flutter answers those in the native cross platform experience and dart answers that in the web, object-oriented and mobile experience.

**References:**

[1] https://cdn0.tnwcdn.com/wp-content/blogs.dir/1/files/2018/02/Flutter-hed-796x419.jpg

[2] https://cdn.arstechnica.net/wp-content/uploads/2018/02/7-2.jpg

[3] Wu, Wenhao, “React Native vs Flutter, cross-platform mobile application frameworks” (01 March 2018) Retrieved from URL

[4] Ratabouil, Sylvain “Android NDK: Beginner’s Guide – Second Edition” (April 2018)

[5] Henno, J, Kiyoki, Y, Tokuda, T “Using LLVM for a Functional Programming Language Implementation” (2012) Retrieved from URL

[6] https://blogs.infosupport.com/wp-content/uploads/2013/05/image4.png

[7] Bracha, Gilad “The Dart Programming Language” (2016) Retrieved from URL