Homework 6 (Dart) Report
CS 131

Abstract

This paper is a report analyzing the pros and cons of the language Dart and evaluating its features as a potential platform to build GarageGarner. The paper compares Dart with other mainstream programming languages such as OCaml, Java, and Python and highlights the differences between Dart and each of them.

1 Introduction to Dart

Dart is a language that has multiple applications from writing shell scripts to fully built applications. The language is split into Dart Native and Dart Web.

1.1 Dart Native

Dart Native is for programmers writing code with mobile and desktop applications as well as many others. With Dart Native, programmers can choose to compile the code using a just-in-time compiler which is used for iOS applications. This also allows for runtime optimization. The other option for Dart Native programmers is an ahead-of-time compiler that compiles into ARM, both ARM32 and ARM64, or x86-64 machine code. This allows for applications to start instantly and run smoothly on whatever platform the program is deployed to.

1.2 Dart Web

For web development, Dart Web allows Dart code to be compiled into JavaScript so that it can be run on browsers. Dart Web also powers Flutter which is a multi-platform UI toolkit that can be used for web development. Dart Web comes with two compilers, dartdevc and dart2js. dartdevc compiles Dart code into JavaScript rapidly for development purposes. On the other hand, while dart2js also compiles Dart code to JavaScript, it employs optimization techniques to build deployable code.

1.3 Multi-threading and Asynchronous Code

Dart supports asynchronous programming through the await and async keywords. The await keyword is usually used before expressions that return a Future[1]. If the expression doesn't return a Future, it is automatically wrapped inside one.
One interesting feature of Dart is that instead of supporting multi-threading, Dart employs the use of isolates. An isolate is like a thread except that it has its own memory on the heap that no other isolate, even its parent is allowed to access. Each isolate has its own event loop and can communicate with other isolates only through messages that are sent and received by the event loop. This removes the need for locking and unlocking memory because only one isolate is in execution at a time and so the memory isn't being modified anytime any isolate tries to access it.

1.4 Garbage Collection

Garbage collection in Dart is very important especially in Flutter since widgets are created and destroyed constantly on the user's screen. There are two approaches dart has towards garbage collection. The first is used for rapidly created and destroyed objects. Objects are allocated in a nursery that is split in two halves, one half is active, and the other half is inactive. When the active half fills up, all live objects are copied to the other half which gets set as the active half. Live objects are determined by following references from root objects. The other garbage collection approach is the typical Mark and Sweep approach.

2 Comparison of Languages

2.1 OCaml

Dart and OCaml are two different languages with very different applications. Both languages are object oriented but OCaml is more functional while Dart is driven by an event loop. While OCaml employs strict static type checking, Dart uses a combination of both static and dynamic type checking to ensure type-safe programs. Dart also makes concurrent programming easy and straightforward with the await and async

[1] The Future object is a promise to return an object.

keywords while writing asynchronous programs in OCaml are somewhat difficult.

2.2 Java

Dart and Java are both object-oriented languages that support generic typing. Like OCaml, Java is statically typed whereas Dart employs a combination of static and dynamic type checking. While Java allows for multi-threading and parallel computing, Dart is single-threaded, meaning if the current isolate is stuck, the whole program will freeze. Finally, both Java and Dart employ a similar style of garbage collection; they use a generation-based allocator to free space described in Section 1.4.

2.3 Python

Python is an interpreted language while Dart is compiled meaning that Dart is much faster at runtime than Python, making it better for responsive applications like UI development. Both Python and Dart use the Mark and Sweep approach as a secondary garbage collection approach. Additionally, they both have simple ways of writing asynchronous code simply by using the async and await keywords. One other difference between the languages is that Python is much easier to read and much more beginner friendly due to its English-like syntax.

3 GarageGarner

Dart is a good choice to build a less "klunky" application because it allows for ahead-of-time compilation which can optimize the code and decrease the amount of space it takes. This also allows the application to start instantly and run smoothly which is key in user facing applications. Although the Dart compiled code itself would be less bulky than other language choices, the size of the machine learning code, now on the application, would cause the application to be a lot bulkier. However, Dart's asynchronous features, will allow the application to continue to run smoothly for the user while another isolate awaits on the TensorFlow Lite libraries to process the image. Dart's garbage collector will make sure the memory used by the application is minimized so performance is not reduced due to the machine learning models running on the phone.

One issue with Dart as a choice for this application is that it doesn't support multi-threading. Although this results in thread-safe code, a multi-threaded approach such as Java may improve performance. This won't be known without testing because although Java's multi-threading will allow for more computation through parallel processing, Dart code inherently runs much faster than Java. This means without testing, it will be hard to tell which approach will have better performance.

4 Conclusion

Dart is a relatively new language targeted towards fixing problems in many current languages for building applications. It is faster and more efficient than Java, JavaScript, and Python and contains many of the same benefits of each of these languages. Excluding being strictly statically typed and having multi-threading capabilities, Dart supports almost every other feature programmers want in languages today such as easy-to-use asynchronous libraries, garbage collection for efficiency, and quick testing in development.

5 References

1. https://medium.com/dartlang/dart-asynchronous-programming-isolates-and-event-loops-bffc3e296a6a

2. https://dart.dev/

3. https://www.foreach.be/blog/parallel-and-asynchronous-programming-in-java-8?lang=nl