CSC 372 PROJECT

Hanwei Li
lihanwei4c@email.arizona.edu

Evan Kuo
evankuo@email.arizona.edu

April 26, 2020

ABSTRACT

In this paper, we will talk about the Dart programming language first showcased in 2011. In particular, we will cover the language on a technical level, covering its control structures, data types, etc, and its uses and how it distinguishes itself from the competition (mainly Java).

1 Introduction

Dart is a programming language made to be heavily client-optimized on multiple different platforms, including mobile, desktop, and web applications. It is a class-based, objected-oriented language that is automatically garbage compiled with Java-esque syntax. It can be compiled to JavaScript or native code and it supports interfaces, mixins, abstract classes, reified graphics and type interface.

2 History

Dart was developed by Google in 2011 by Lars Bak and Kasper Lund and first unveiled at the GOTO conference in Aarhus, Denmark[1]. It is a objected-oriented language with C style syntax that is used largely in mobile development [2].It is used heavily within the Flutter framework to develop mobile applications such as Google Ads on mobile and can develop both IOS and Android apps. One of the primary advantages of Dart is that as of version 2.6 it can be compiled into self contained Javascript executables and doesn't require the Dart SDK to be installed [3]. The language is constantly supported and got its most recent patch on March 24, 2020 [4].

3 Control Structures

Dart has a wide array of control structures that help augment the flow of our code much like many other programming languages. Knowledge of Python and/or Java would be helpful in understanding how the control structures in Dart work.

Control structures in Dart are similar to those found in a language like python but have a syntax more similar to Java. They are higher level control structures that have all of the elements we would expect from a high level language. This includes, if/then/else, while, for, switch, for...In and more.

Let's go over each of these individually and provide examples:

— The For Loop —

The for loop in Dart has a structure very similar to that of C or Java. It starts with a variable declaration, then a conditional statment, and then an iteration. An example follows:

```
for (var i = 1; i <= 20; i++) {
    print(i);
}
```

where this example will print out 1...20.

— The For...In Loop —

Dart has a for...In structure for for loops that is very similar to that found in Python. This is very useful if we are trying to loop through an object or list of some sort. It follows a structure that is practically identical to that found in python and uses a "for var name in structure" type of formatting. An example is as follows:

```
var groceryList = ["bread", "bananas", "dog␣food"];

for (var food in groceryList) {
        print("On␣my␣grocery␣list␣I␣have:␣" + food + ".");
    }
```

— The While Loop —

The while loop in Dart is also similar to C/Java/Python. It has a conditional that must be evaluated and continues until that conditional is false. An example is as follows:

```
var num = 10;
while (num > 0) {
    print(num);
    num = num - 1;
}
```

— The If Statement —

The If statement in Dart is similar to that found in Java/C/Python and other high level languages. It consists of an if statement with a conditional in parenthesis followed by an else if needed. An example is as follows:

```
    // If x is even, print out that x is even
    if (x % 2 == 0) {
            print("x␣is␣currently␣" + xString + "␣and␣it␣is␣an␣even␣number");
    }

    // Else, print out it is odd
    else {
            print("x␣is␣currently␣" + xString + "␣and␣it␣is␣an␣odd␣number");
    }
```

— The Switch Statement —

The switch statement is also very similar to the switch statements found in other languages. It takes a value and then goes into different control patterns based upon whether or not certain conditionals pass. These conditionals are exited out of with a break statement and like in other languages this break statement is not strictly necessary. An example is as follows:

```
var cscClass = "CSC372";

switch(cscClass) {

    case "CSC372": { print("CSC372␣is␣a␣good␣class!"); }
    break;

    case "CSC345": { print("CSC345␣is␣super␣hard!"); }
    break;

    case "CSC445": { print("CSC445␣is␣really␣hard!!!"); }
    break;

    default: { print("I␣haven't␣heard␣of␣that␣class."); }
    break;

}
```

More details can be found in the github page of this project [5].

4 Data Types

Dart includes the same data types as many other high level programming languages. It has: numbers, strings, booleans, lists, maps, and sets. We will go over each individually and then we will go over the operations that can be performed on these data types. We will include examples of the discussed data type or operation where appropriate. And at the very end we include some useful resources for learning about Dart's data types.

— Numbers —

The number data type is... numbers! It has integers and doubles. Integers are represented with the int keyword and are used for any non-fractional value such as 15. Integers are also represented in 32 bits so they cannot exceed 4.3 billion. There are also doubles. Doubles are used to represented fractional or floating point numbers such as 15.5. They are represented in 64 bits so they can go very high.

— Strings —

Strings are text sequences in Dart. They are written as a sequence of UTF-16 characters and are represented with the keyword String. Like in Python, they can be represented with single and double quotes. Also like in Python, there are no characters. Every text sequences, even those of a single letter, are strings. An example of a string would be "I am a string" or "b". Strings in Dart support all of the usual UTF characters and can include special characters as well which are encapsulated in runes.

— Booleans —

Booleans are either true or false. They are represented by the bool keyword and act just like booleans in other high level languages such as Java or Python. Just like in other langauges, booleans can be used in loops, can be checked with ==, etc. Just like in other languages like C, booleans are not strictly necessary to get most things done but are nevertheless useful.

— List —

A List is an ordered sequence of objects. In Dart, the List replaces what is normally the array in most languages. Dart does not have any arrays and uses only the list to for an ordered collection. Dart lists, unlike those in Java, can have multiple different data types in them. So a list in Dart can have both strings and ints. Lists in Dart are growable, we can insert at the beginning, end, or some point inbetween, and we can also remove elements from them. We can also index into the list. Like in basically every other programming language, the list's indexing starts at 0. A list is created with the bracket syntax. For example, [1, 2, 3].

— Map —

A map works much like a dictionary in Python. It is a collection of key/value pairs where we access the values with map[key] = value. By using the Keys() and the Values() method we can also get an iterable object of the keys and the values in the map. A map is created with the curly braces syntax such as "username": "Evan", "password": 1234. The keys can then later be accessed with the bracket syntax such as mapName["username"], which would then return the string "Evan" for the dictionary we defined above.

— Sets —

And finally we have the set. The set is an unordered collection of objects in which each object can appear only once, just like in most other programming languages. Unlike a list, a set can only contain one sort of data type. This is so the set can be ordered. Sets can have elements removed from them but we cannot index into them as they have a random ordering. We can however find elements.

— Operations —

And now we are at the operations we can perform on the data types. In Dart, we have the standard arithmetic operators such as +, -, *, /, ++, –, modulo, and etc. These are operators that are used primarly on arithmetic but some can be used on strings as well. Strings can be concatenated with the '+' operator. We also have the equality and relational operators such as ==, >=, <, !=, etc. These equality operators can be used on any data type, such as comparing strings, lists, etc. Dart also has the "is" operator. This checks if a given object is of the given datatype and returns true or false based on the result. So for example, "I'm a string" is String would return true and "I'm a string" is int would return false. This is very similar to the typeof operator in Java. We also have the bitwise operators such as , |, «, etc. These can of course be used for bitwise calculations. Dart also supports logical operators such as , ||, and !. Noticeably, it does not use the Python syntax of and, or, not and instead uses a more Java centered syntax. There are also assignment operators similar to those found in other languages such as =, +=, -=, *= and more. There is ??= which only assigns the value if the variable is null. There are also conditional expression operators such as expr1 ?? expr2, which returns expr1 if it is non-null and returns expr 2 elsewise.

— Further Reading —

More information about data types and operators in Dart can be found at https://dart.dev/guides. While writing this, the dart tutorial on the website tutorialspoint.com was also very useful.

5 Subprograms

Dart supports many of the same subprograms as other high level programming languages. These subprograms include classes, packages, and functions. Let's go over each one of these and look at each in some depth.

— Functions —

Since packages and classes make use of functions, let's go over functions first. Function in Dart work the same way as they do in almost every other statically typed programming language. They have a type declaration, a function name, accept parameters in parenthesis, and have curly brackets for the function body. Here is an example of a function:

```
void sayHello(String firstName, String secondName) {
    print("Hello␣" + firstName);
    print("Hey␣" + secondName);
}
```

This example function has the type declaration of void. Functions in Dart can have any data type or structure as their type declaration. To learn morea bout these, please see our section on data types in Dart. The function also has a name, which in this case is helloWorld. This function also has parenthesis for parameters and accepts two parameters: the String firstName and another String secondName. The parameters are separated by commas and also have type declarations. Lastly, the function has a function body that is enclosed in brackets. The body of a function can contain other functions such as print. Dart also supports recursion within functions, anonymous functions, etc.

— Classes —

Just like in Java, Dart supports object-oriented programming through the use of classes. Classes in Dart are declared the same way as they are in Java with the syntax:

```
class ClassName {

    String name = "Fred";

    // Here is a function for the class
    void sayMyName() {
        print("I'm␣" + name);
    }

}
```

A class is declared with the keyword class. It has a name, which in this case is ClassName and its body is enclosed within curly brackets. A class can have its own local variables. In this case, our class has a local String called name. It then has a function which prints out that name.

— Packages —

Dart also supports many different packages. Some of the most popular packages can be found at https://pub.dev/ and these packages support a wide range of features, mainly support for mobile phone app development. Many of the packages support Flutter, which is Dart's primary mobile app development framework. To use a package in our dart programs, we use the keyword import:

```
import "packagename";
```

One important thing to note about Dart's import is that it requires quotations around the name of the package, unlike in other languages like Java where we simply name the package without the imports. Once we import the package we are then free to use the functions and other items from the package. It is also noteworthy that this import syntax is how we import functions from other dart files. So if we write an object-oriented dart program and want to use a class/function from another part of the program, then we can use the import syntax listed above. We can also create our own packages in Dart. The process is rather complex, so instead of listing it all here I believe it would be easier and more succinct to include a link to the page: https://www.tutorialspoint.com/dart$_{p}rogramming/dart_{p}rogramming_{p}ackages.htm$

— Summary of Subprograms —

In summary, dart has many of the same subprograms that other languages have. If Java or Python has a certain subprogram feature, then Dart probably has something similar. What's also noteworthy is that many of the subprograms have a syntax similar or even identical to that found in Java, so any Java programmers should have an easy time transitioning from Java's subprograms to Dart's.

6 Summary

In summary, Dart is a language very similar to Java in its broad steps. Syntactically, it borrows the vast majority of its syntax from Java and this makes it really easy to go from one to the other. However, Dart does have things that make it distinct. Most notably, it can be compiled as JavaScript, compiled as snapshots, and it can be used to develop Flutter mobile apps, which is what is best known for. Dart is a great language for anyone who already knows Java since the barrier-for-entry will be very low and the benefits (especially being able to develop mobile apps) are numerous.

References

[1] Gilad Bracha and Lars Bak. Goto conference. In *Opening Keynote: Dart, a new programming language for structured web programming*. Arhus, 2011.

[2] Gilad Bracha. *The Dart Programming Language*. Addison-Wesley Professional, 2015.

[3] Jakub Lewkowicz. Dart 2.6 release with dart2native. *SD Times: Software Development*, 11 2019. https://sdtimes.com/goog/dart-2-6-released-with-dart2native/.

[4] Alexandar Thomas. 2.8.0-dev.20.0. Github, 3 2020. https://github.com/dart-lang/sdk/releases/tag/2.8.0-dev.20.0.

[5] Evan Kuo and Hanwei Li. 372 group project. Github. https://github.com/372groupproject/milestones-team32-evankuo-lihanwei4c.