JAVA and DART programming languages: Conceptual comparison

Afaf Mirghani Hassan
Computer Department, Tabuk University (Branch Duba), Saudi Arabia

ABSTRACT

This paper elaborates on the concepts of a new programming language “Dart”, which has been developed by Google and considered for future use. Here, we compare it to the most famous, real time, and updated language “Java”. This is to define similarities and differences between the two important languages, explain programs’ behavior, with a focus on investigating alternative implementation strategies and problem definitions. We used programming languages’ concepts and terminologies to compare between the main characteristics of the two languages, Dart & Java.

1. INTRODUCTION

Google has released a new language aimed at developing complex, Google-scale web applications in October 2011. The aim was to develop a language that is better language for the web than JavaScript. That was because of the frustration with the slow progress in evolving JavaScript, partly caused by the so many interested parties. The main goal was to sustain the dynamic nature of JavaScript, but have a better performance and is extendable to tooling for large projects. It would also be able to cross-compile to JavaScript. This language was given the name Dart [1].

Dart is a general purpose programming language. It is a new language in the C tradition, designed with ease of use, familiarity to the vast majority of programmers, and scalability in mind. It is purely object-oriented, class-based, programming language. Dart is intended to provide a platform that is specifically crafted to support future needs and emerging software/hardware platforms. As such it hides low level details of the underlying platform, while enabling programmers to use the powerful facilities new platforms have to offer [2].

It is an open source, structured language to create complex, browser-based web applications. Applications usually run in Dart either by the browser directly, which supports Dart code, or by compiling code to JavaScript. Dart has a familiar syntax, and it’s class-based. It has a concurrency model called isolates that allows parallel execution. In addition to running code in web browsers and converting it to JavaScript, it can also run code on the command line [3-10]. For client side web app development, Dart has many advantages over JavaScript. These include but are not limited to improved speed, enforcement of programmatic structure, and improved facilities for software reuse. Best of all, Dart is automatically converted to JavaScript so that it works with all web browsers, Dart is a fresh start, without the baggage of
the last two decades of the webDart language Designer has pragmatic choice to make smooth experience coding [3-10].

Java is an efficient programming language likable by developers and so is Dart. Both languages have powerful concepts such as object creation, concurrency, serialization, reflection, and many more, all in real time [3, 4, 11-17]. Java has evolved over time; newer versions of Java increase the need for specific best-practices advice for multiple paradigms, functional interfaces, lambda expressions, method references, and streams. Default and static methods in interfaces, resources’ statement, New library features such as the Optional interface, java.time, and factory methods for collections. All of that, so developers can convert to dart with relative ease [11-17]. One example of a Dart implementation is Flutter, a mobile app SDK from Google, which has Java integrity. The app Create a simple Dart class, Use optional parameters (overloading). Create a factory. Implement an interface. Use Dart for functional programming [3-12]. The usability and familiarity of the language makes it a good candidate to implement complex engineered systems such as those in [18-21].

We would like to find a geometric recognition language, a graphics interpreter, a rule-based control interpreter, and an object-oriented language interpreter to work together at once [22]. One good practice is to structure a complex program as a collection of languages, each of which provides a different viewpoint, different way for different program elements [22]. It might be this is the reason why our programs are becoming increasingly complex thinking more explicitly about languages might be the best way to deal with this complexity. The basic idea is that the interpreter itself is just a program that is written in some language, whose interpreter is another program, which is written in some other language etc.

One main objective or strategy concept of a programming language is to distinguish itself from other languages based on the characteristics and usages or utilities of the language. In this study, we try to investigate the programming language Dart, by comparing to the important programming language (JAVA), concentrating on the similarities and differences of the two. Section 2 details this comparison and present it in an easy readable table format. The paper is concluded in Section 3.

2. DART AND JAVA: COMPARISON OF CONCEPTS

Google is a real time interactive system application dealing with search, electronic mail, translation, play, images, drive and many other applications. Hence, the company is in a constant search to develop programming languages that connect all of these, and also future applications. Dart programming language has come to meet this need [23].

Java is considered a general-purpose programming language while Dart is a client-optimized programming language. There are many similarities and differences between the two programming languages. They are similar in criteria such as readability, reliability, cost, portability, and generality [4]. Both languages are writable and well-defined or precise languages. The two languages are roughly OOP languages (Object Oriented Programming), they are classes’ structure, and both based on C structure i.e. similar software syntax in C. Also, the two of them are web software languages, and both are lovely languages for developers [24].

Some of the differences between the two are: while Java is general purpose language, Dart is considered Google specific language. Dart is a class structure same as Java, however in Dart class code cannot be written. This is in contrast to Java, a class-based object-oriented programming language not a pure object-oriented one. That is, Java has a second scoping mechanism (Package Scope) that can be used in place in all classes in a package, in case there are no access control modifiers that are visible throughout the package [6].

Table 1 shows a Comparison between Dart and Java in terms of behavior, syntax, semantic, value, environment, expression, procedure and conditional clauses. With Dart we can create applications on the web, smart phones and servers [22-26].

<table>
<thead>
<tr>
<th>Comparison points</th>
<th>Java</th>
<th>Dart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authority</td>
<td>Sun company, now Oracle</td>
<td>Google company</td>
</tr>
<tr>
<td>Generation</td>
<td>Updated interactive language</td>
<td>Future language</td>
</tr>
<tr>
<td>Script language</td>
<td>Can be Web programming</td>
<td>Interactive web language</td>
</tr>
<tr>
<td>Syntax</td>
<td>-many similar structure as in Dart</td>
<td>• with Construct class as in java</td>
</tr>
<tr>
<td></td>
<td>- contain keywords</td>
<td>• not contain keywords</td>
</tr>
<tr>
<td></td>
<td>Example is Java’s static semantics rule: else matches with the nearest if.</td>
<td>• depends on has letters (_) that is enters to special library words or commands.</td>
</tr>
<tr>
<td>Semantic</td>
<td>Example is the Flutter semantics package.</td>
<td>Example is the Flutter semantics package.</td>
</tr>
</tbody>
</table>

Table 1. Comparing Dart and Java

### Comparison points

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>Dart</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Portability</td>
<td>(Java Virtual Machine) JVM concept, JIT (Just In Time ) compilers</td>
</tr>
<tr>
<td>7</td>
<td>C language architecture</td>
<td>C, java are imperative language (same categories)</td>
</tr>
<tr>
<td>8</td>
<td>Java Script language</td>
<td>similar syntax</td>
</tr>
<tr>
<td>9</td>
<td>false result</td>
<td>more than one false result (null, false, 0)</td>
</tr>
<tr>
<td>10</td>
<td>Cascade Notation</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>Comment</td>
<td>//</td>
</tr>
<tr>
<td>12</td>
<td>Run</td>
<td>UNICODE, ASCII</td>
</tr>
<tr>
<td>13</td>
<td>Asynchrony support</td>
<td>code run line by line</td>
</tr>
<tr>
<td>14</td>
<td>Exception Handling and Event Handling</td>
<td>-All exceptions are objects of classes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Throw able (Error- exception).</td>
</tr>
<tr>
<td>15</td>
<td>Data Types</td>
<td>Defined all:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Primitive Data Types</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Character String (String class)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Array (index integer types) -Array Initialization string object- support juggled arrays</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-not support unions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-allows replace pointers</td>
</tr>
<tr>
<td>16</td>
<td>Expressions and Assignment Statements</td>
<td>-assignment statement produces a result and can be used as operands</td>
</tr>
<tr>
<td>17</td>
<td>Control Structures</td>
<td>-Mixe Mode (widening assignment)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Java's static semantics rule: else matches with the nearest if.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Multiple- Way Selection (Switch).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unconditional labeled e (break),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-labeled versions of continue.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-do not support goto statement.</td>
</tr>
<tr>
<td>18</td>
<td>Libraries</td>
<td>Use import to specify how a namespace from one</td>
</tr>
<tr>
<td>19</td>
<td>Lexical scope</td>
<td>inherent class</td>
</tr>
<tr>
<td>20</td>
<td>Subprograms</td>
<td>-All parameters are passed by value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Object parameters are passed by reference.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Require Type Checking Parameters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Array inherits a named constant length.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-allow programmers to write multiple versions of subprograms with the same name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-predefined overloaded subprograms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-allow Generic Subprograms</td>
</tr>
<tr>
<td>21</td>
<td>An Example structure</td>
<td>class StackClass {</td>
</tr>
<tr>
<td></td>
<td></td>
<td>private:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>private int *stackRef;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>private int [] maxLen, tophIndex;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>public StackClass() { // a constructor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stackRef = new int [100];</td>
</tr>
<tr>
<td></td>
<td></td>
<td>maxLen = 99;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tophPr = -1;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>public void push (int num) {...};</td>
</tr>
<tr>
<td></td>
<td></td>
<td>public void pop () {...};</td>
</tr>
<tr>
<td></td>
<td></td>
<td>public int top () {...};</td>
</tr>
<tr>
<td></td>
<td></td>
<td>public boolean empty () {...};</td>
</tr>
</tbody>
</table>

### 3. CONCLUSION

Dart is a powerful, interactive language that is expected to get widely adopted by developers the same way Java is adopted today. Dart code can be reused for either smat phones (clients) or servers; however, it still lacks the general-purpose property of Java. In domain-specific (scientific, business, artificial intelligence, web and system) applications, The programming language’s domain is extended to the special objectives of that domain; In this capacity Dart can be considered as a domain-specific programming language. Dart as a domain-specific programming language will be one of the Web Software with three branches of web or Eclectic collection of languages software, which are markup (HTML), script (PHP) and general purpose (JAVA).
Below example code that can be compiled with DART, HTML, CSS on the same time; with HTML output:

```dart
Stream<double> computePi({int batch: 100000}) async* {
  /// Generates a stream of increasingly accurate estimates of π.
  while (true) {
    var count = 0;
    var total = 0;
    var inside = points.where((p) => p.isInsideUnitCircle);
    while (true) {
      var inside = points.where((p) => p.isInsideUnitCircle);
      var count += inside.length;
      var total += Random.nextDouble() * 2 + 1;
      var estimate = count / total;
      await for (var estimate in computePi().take(500)) {
        output.text = estimate.toStringAsFixed(5);
        await window.animationFrame;
        print('π ≅ $estimate');
      }
    }
  }
}
import 'dart:html';
import 'dart:math' show Random;

main() async {
  // after 500 iteration
  // the number gets updated in the DOM, and the program ends
  // the web (so that the execution waits for animation frame,
  // We changed 5 lines of code to make this sample nicer on
  import 'dart:html';

  // perform the calculation
  var output = querySelector('#output');
  await for (var estimate in computePi().take(500)) {
    // after 500 iterations).
    // generate N random numbers within the unit circle.
    var points = generateRandom().take(batch);
    var inside = points.where((p) => p.isInsideUnitCircle);
    var total += inside.length;
    var estimate = inside.length / total;
    // update the output
    output.text = estimate.toString();
  }
}
```

REFERENCES


APPENDIX

Below example code that can be compiled with DART, HTML, CSS on the same time; with HTML output and CONSOLE [26].
var ratio = count / total;
// Area of a circle is \( A = \pi r^2 \), therefore \( \pi = \frac{A}{r^2} \).
// So, when given random points with \( x \in [0,1) \),
// \( y \in [0,1) \), the ratio of those inside a unit circle
// should approach \( \pi / 4 \). Therefore, the value of \( \pi 
// \) should be:
yield ratio * 4;
}
}
Iterables<Point> generateRandom([int seed])
final random = Random(seed);
while (true) {
yield Point(random.nextDouble(), random.nextDouble());
}
class Point {
final double x, y;
const Point(this.x, this.y);
}

main() async {
print('Compute \( \pi \) using the Monte Carlo method.');
var output = querySelector('#output');
await for (var estimate in computePi().take(500)) {
print('\( \pi \equiv \) 
\[ \text{estimate} \]');
output.text = estimate.toStringAsFixed(5);
await window.animationFrame;
}

Iterables<Point> generateRandom([int seed])
final random = Random(seed);
while (true) {
yield Point(random.nextDouble(), random.nextDouble());
}
class Point {
final double x, y;
const Point(this.x, this.y);
bool get isInsideUnitCircle => x * x + y * y <= 1;
import 'dart:html';
import 'dart:math' show Random;
// We changed 5 lines of code to make this sample nicer on
// the web (so that the execution waits for animation frame,
// the number gets updated in the DOM, and the program ends
// after 500 iterations).
Stream<double> computePi({int batch: 100000})
final total = 0;
final count = 0;
while (true) {
var points = generateRandom().take(batch);
var inside = points.where((p) => p.isInsideUnitCircle);
total += batch;
count += inside.length;
var ratio = count / total;
// Area of a circle is \( A = \pi r^2 \), therefore \( \pi = \frac{A}{r^2} \).
// So, when given random points with \( x \in [0,1) \),
// \( y \in [0,1) \), the ratio of those inside a unit circle
// should approach \( \pi / 4 \). Therefore, the value of \( \pi 
// \) should be:
yield ratio * 4;
}

JAVA and DART programming languages: conceptual comparison (Afaf Mirghani Hassan)