WIKIPEDIA

Dart (programming language)

Dart is a programming language designed for client development,[9][10] such as for the web and mobile apps. It is developed by Google and can also be used to build server and desktop applications.

Dart is an object-oriented, class-based, garbage-collected language with C-style syntax.[11] Dart can compile to either native code or JavaScript. It supports interfaces, mixins, abstract classes, reified generics, and type inference.[12]

Dart

Paradigm	Multi-paradigm: functional, imperative, object-oriented, reflective[1]
Designed by	Lars Bak and Kasper Lund
Developer	Google
First appeared	October 10, 2011[2]
Stable release	2.14.1[3] / 8 September 2021
Preview release	2.14.1 / September 8, 2021[4]
Typing discipline	1.x: Optional 2.x: Inferred[5] (static, strong)
Platform	Cross-platform
OS	Cross-platform
License	BSD
Filename extensions	.dart
Website	dart.dev (https://dart.dev)
Major implementations	
Dart VM, dart2native, dart2js, DDC, Flutter	
Influenced by	
C, C++, C#, Erlang, Java, JavaScript, Kotlin,[6] Ruby, Smalltalk, Strongtalk,[7] TypeScript[8]	

Contents

History

Dart was unveiled at the GOTO conference in Aarhus, Denmark, October 10–12, 2012.[13] The project was founded by Lars Bak and Kasper Lund.[14] Dart 1.0 was released on November 14, 2013.[15]

Dart initially had a mixed reception and the Dart initiative has been criticized by some for fragmenting the web, due to the original plans to include a Dart VM in Chrome. Those plans were dropped in 2015 with the 1.9 release of Dart to focus instead on compiling Dart to JavaScript.[16]

In August 2018, Dart 2.0 was released, with language changes including a sound type system.[17]

Dart 2.6 introduced a new extension, dart2native. The feature extends native compilation to the Linux, macOS, and Windows desktop platforms. Earlier developers were able to create new tools only using Android or iOS devices. Moreover, with this extension it becomes possible to compose a Dart program into self-contained executables. Thus, according to the company representatives, it's not obligatory now to have Dart SDK installed, the self-contained executables can now start running in a few seconds. The new extension is also integrated with Flutter toolkit, thus making it possible to use the compiler on small services (backend supporting for example).[18][19]

Standardization

Ecma International has formed technical committee TC52[20] to work on standardizing Dart, and inasmuch as Dart can be compiled to standard JavaScript, it works effectively in any modern browser. Ecma International approved the Dart language specification first edition in July 2014, at its 107th General Assembly,[21] and a second edition in December 2014.[22] The latest specification is available at Dart language specification (https://dart.dev/guides/language/spec).

Usage

There are four ways to run Dart code:

Web
: To run in mainstream web browsers, Dart relies on a source-to-source compiler to JavaScript. According to the project site, Dart was "designed to be easy to write development tools for, well-suited to modern app development, and capable of high-performance implementations."[23] When running Dart code in a web browser the code is precompiled into JavaScript using the dart2js compiler. Compiled as JavaScript, Dart code is compatible with all major browsers with no need for browsers to adopt Dart. Through optimizing the compiled JavaScript output to avoid expensive checks and operations, code written in Dart can, in some cases, run faster than equivalent code hand-written using JavaScript idioms.[24]

Stand-alone
: The Dart software development kit (SDK) ships with a stand-alone Dart VM, allowing Dart code to run in a command-line interface environment. As the language tools included in the Dart SDK are written mostly in Dart, the stand-alone Dart VM is a critical part of the SDK. These tools include the dart2js compiler and a package manager called pub. Dart ships with a complete standard library allowing users to write fully working system apps, such as custom web servers.[25]

Ahead-of-time compiled
: Dart code can be AOT-compiled into machine code (native instruction sets). Apps built with Flutter, a mobile app SDK built with Dart, are deployed to app stores as AOT-compiled Dart code.[26]

Native
: Dart 2.6 with dart2native compiler to compile to self-contained, native executables code. Before Dart 2.6, this feature only exposed this capability on iOS and Android mobile devices via Flutter.[27]

Isolates

To achieve concurrency, Dart uses isolates, which are independent workers that do not share memory, but instead use message passing.[28] This is similar to Erlang processes (see also Actor model). Every Dart program uses at least one isolate, which is the main isolate. Since Dart 2 the Dart web platform no longer supports isolates, and suggests developers use Web Workers instead.[29]

Snapshots

Snapshots are a core part of the Dart VM. Snapshots are files which store objects and other runtime data.[28]

Script snapshots
: Dart programs can be compiled into snapshot files. These files contain all of the program code and dependencies preparsed and ready to execute. This allows fast startups.

Full snapshots
: The Dart core libraries can be compiled into a snapshot file which allows fast loading of the libraries. Most standard distributions of the main Dart VM have a prebuilt snapshot for the core libraries which is loaded at runtime.

Object snapshots
: Dart is a very asynchronous language. With this, it uses isolates for concurrency. Since these are workers which pass messages, it needs a way to *serialize* a message. This is done using a snapshot, which is generated from a given object, and then this is transferred to another isolate for deserializing.

Native mobile apps

Google has introduced Flutter for native mobile app development on both Android and iOS.[30] Flutter is a mobile app SDK, complete with framework, widgets, and tools, that gives developers a way to build and deploy mobile apps, written in Dart.[31] Flutter works with Firebase[32] and other mobile app SDKs, and is open source.

Compiling to JavaScript

The Dart SDK contains two Dart-to-JavaScript compilers. During development, dartdevc (https://webdev.dartlang.org/tools/dartdevc) supports quick refresh cycles. For the final version of an app, dart2js (https://webdev.dartlang.org/tools/dart2js) produces deployable JavaScript.[33]

The first compiler to generate JavaScript from Dart code was dartc, but it was deprecated. The second Dart-to-JavaScript compiler was Frog. It was written in Dart, but never implemented the full semantics of the language. The third Dart-to-JavaScript compiler was dart2js. An evolution of earlier compilers, dart2js is written in Dart and intended to implement the full Dart language specification and semantics.

On March 28, 2013, the Dart team posted an update on their blog addressing Dart code compiled to JavaScript with the dart2js compiler,[34] stating that it now runs faster than handwritten JavaScript on Chrome's V8 JavaScript engine for the DeltaBlue benchmark.[35]

Editors

On November 18, 2011, Google released Dart Editor, an open-source program based on Eclipse components, for macOS, Windows, and Linux-based operating systems.[36] The editor supports syntax highlighting, code completion, JavaScript compiling, running web and server Dart applications, and debugging.

On August 13, 2012, Google announced the release of an Eclipse plugin for Dart development.[37]

On April 18, 2015, Google announced that the Dart Editor would be retired in favor of the JetBrains integrated development environment (IDE),[38] which is now the recommended IDE for the language. The Dart plugin[39] is available for IntelliJ IDEA, PyCharm, PhpStorm and WebStorm. This plugin supports many features such as syntax highlighting, code completion, analysis, refactoring, debugging, and more. Other plugins are available for editors like Sublime Text, Atom, Emacs, Vim and Visual Studio Code.[40]

Chrome Dev Editor

In 2013, the Chromium team began work on an open source, Chrome App-based development environment with a reusable library of GUI widgets, codenamed Spark.[41] The project was later renamed as Chrome Dev Editor.[42] It was built in Dart, and contained Spark which is powered by Polymer.[43]

In June 2015, Google transferred the CDE project to GitHub as a free software project and ceased active investment in CDE.[44] As of April 2019 Chrome Dev Editor is no longer in active development.[45]

DartPad

The Dart team created DartPad at the start of 2015, to provide an easier way to start using Dart. It is a fully online editor from which users can experiment with Dart application programming interfaces (APIs), and run Dart code. It provides syntax highlighting, code analysis, code completion, documentation, and HTML and CSS editing.[46]

SIMD

In 2013, John McCutchan announced[47] that he had created a performant interface to single instruction, multiple data (SIMD) instruction sets for Dart.

The interface consists of two types:

- Float32×4, 4× single precision floating point values
- Uint32×4, 4× 32-bit unsigned integer values

Instances of these types are immutable and in optimized code are mapped directly to SIMD registers. Operations expressed in Dart typically are compiled into one instruction with no overhead. This is similar to C and C++ intrinsics. Benchmarks for 4×4 matrix multiplication, 3D vertex transformation, and Mandelbrot set visualization show near 400% speedup compared to scalar code written in Dart.

Example

A Hello, World! example:

```
void main() {
  print('Hello, World!');
}
```

A function to calculate the nth Fibonacci number:

```
int fib(int n) => (n > 2) ? (fib(n - 1) + fib(n - 2)) : 1;
// A Fibonacci function implementation with a conditional operator in Dart
// This code is read as:
//  given an integer n,
//  if n > 2, return fib(n - 1) + fib(n - 2);
//  otherwise, return the integer 1 as result

void main() {
  print('fib(20) = ${fib(20)}');
}
```

A simple class:

```
// Import the math library to get access to the sqrt function.
// Imported with `math` as name, so accesses need to use `math.` as prefix.
import 'dart:math' as math;

// Create a class for Point.
class Point {

  // Final variables cannot be changed once they are assigned.
  // Declare two instance variables.
  final num x, y;

  // A constructor, with syntactic sugar for setting instance variables.
  // The constructor has two mandatory parameters.
  Point(this.x, this.y);

  // A named constructor with an initializer list.
  Point.origin()
      : x = 0,
        y = 0;

  // A method.
  num distanceTo(Point other) {
    var dx = x - other.x;
    var dy = y - other.y;
    return math.sqrt(dx * dx + dy * dy);
  }

  // Example of a "getter".
  // Acts the same as a final variable, but is computed on each access.
  num get magnitude => math.sqrt(x * x + y * y);

  // Example of operator overloading
  Point operator +(Point other) => Point(x + other.x, y + other.y);
  // When you instantiate a class such as Point in Dart 2+, new is
  // an optional word
}

// All Dart programs start with main().
void main() {
  // Instantiate point objects.
  var p1 = Point(10, 10);
  print(p1.magnitude);
  var p2 = Point.origin();
  var distance = p1.distanceTo(p2);
  print(distance);
}
```

Influences from other languages

Dart is a descendant of the ALGOL language family,[48] alongside C, Java, C#, JavaScript, and others.

The method cascade syntax, which provides a syntactic shortcut for invoking several methods one after another on the same object, is adopted from Smalltalk.

Dart's mixins were influenced by Strongtalk[49] and Ruby.

Dart makes use of isolates as a concurrency and security unit when structuring applications.[50] The Isolate concept builds upon the Actor model, which is most famously implemented in Erlang.

The Mirror API for performing controlled and secure reflection was first proposed in a paper[51] by Gilad Bracha (who was a member of the Dart team) and David Ungar and originally implemented in Self.

See also

- Google Web Toolkit
- TypeScript, a strongly-typed programming language that transpiles to JavaScript

References

1. Kopec, David (30 June 2014). *Dart for Absolute Beginners* (https://books.google.com/books?id=EcvjAwAAQBAJ&q=dart%20multi-paradigm&pg=PA56). p. 56. ISBN 9781430264828. Retrieved 24 November 2015.
2. Bak, Lars (10 October 2011). "Dart: a language for structured web programming" (http://googlecode.blogspot.com/2011/10/dart-language-for-structured-web.html). *Google Code Blog*. Retrieved 31 January 2016.
3. https://github.com/dart-lang/sdk/releases/tag/2.14.1.
4. "Dart SDK archive" (https://dart.dev/tools/sdk/archive).
5. "The Dart type system" (https://dart.dev/guides/language/sound-dart). *dart.dev*.
6. "Announcing Dart 2.7: A safer, more expressive Dart - Dart - Medium" (https://medium.com/dartlang/dart-2-7-a3710ec54e97). Michael Thomsen. 12 December 2019. Retrieved 24 January 2020.
7. "Web Languages and VMs: Fast Code is Always in Fashion. (V8, Dart) - Google I/O 2013" (https://www.youtube.com/watch?v=huawCRlo9H4&t=30m10s). *YouTube*. Retrieved 22 December 2013.
8. "The Dart Team Welcomes TypeScript" (https://news.dartlang.org/2012/10/the-dart-team-welcomes-typescript.html). Retrieved 22 February 2020.
9. "Dart overview" (https://dart.dev/overview.html). *dart.dev*. Retrieved 2021-04-06.
10. "Dart programming language" (https://dart.dev/). *dart.dev*. Retrieved 2021-04-06. "A programming language optimized for building user interfaces with features such as the spread operator for expanding collections, and collection if for customizing UI for each platform"
11. "A Tour of the Dart Language" (https://dart.dev/guides/language/language-tour#important-concepts). *dart.dev*. Retrieved 2018-08-09.
12. "The Dart type system" (https://dart.dev/guides/language/sound-dart). *dart.dev*.
13. "Dart, a new programming language for structured web programming" (http://gotocon.com/aarhus-2011/presentation/Opening%20Keynote:%20Dart,%20a%20new%20programming%20language%20for%20structured%20web%20programming), *GOTO conference* (http://gotocon.com/aarhus-2011/) (presentation) (opening keynote), Århus conference, 2011-10-10
14. Ladd, Seth. "What is Dart" (http://radar.oreilly.com/2012/03/what-is-dart.html). *What is Dart?*. O'Reilly. Retrieved August 16, 2014.

15. "Dart 1.0: A stable SDK for structured web apps" (https://news.dartlang.org/2013/11/dart-10-stable-sdk-for-structured-web.html). *news.dartlang.org*. Retrieved 2018-08-08.
16. Seth Ladd. "Dart News & Updates" (http://news.dartlang.org/2015/03/dart-for-entire-web.html). *dartlang.org*.
17. Moore, Kevin (2018-08-07). "Announcing Dart 2 Stable and the Dart Web Platform" (https://medium.com/dartlang/dart-2-stable-and-the-dart-web-platform-3775d5f8eac7). *Dart*. Retrieved 2018-08-08.
18. "Dart 2.5 brings native compilation to the desktop" (https://www.infoworld.com/article/3454623/dart-26-brings-native-compilation-to-the-desktop.html). *Infoworld*. 20 November 2019. Retrieved 2019-11-28.
19. "Dart 2.6 released with dart2native" (https://sdtimes.com/goog/dart-2-6-released-with-dart2native/). *SDtimes*. 7 November 2019. Retrieved 2019-11-28.
20. "TC52 - Dart" (https://web.archive.org/web/20160802100651/http://www.ecma-international.org/memento/TC52.htm). Archived from the original (http://www.ecma-international.org/memento/TC52.htm) on 2016-08-02. Retrieved 2013-12-16.
21. Anders Thorhauge Sandholm. "Dart News & Updates" (http://news.dartlang.org/2014/07/ecma-approves-1st-edition-of-dart.html). *dartlang.org*.
22. Anders Thorhauge Sandholm. "Dart News & Updates" (http://news.dartlang.org/2014/12/enums-and-async-primitives-in-dart.html). *dartlang.org*.
23. "Why?", *Dart lang* (http://www.dartlang.org/support/faq.html#why-dart) (FAQ), "We designed Dart to be easy to write development tools for, well-suited to modern app development, and capable of high-performance implementations."
24. "JavaScript as a compilation target: Making it fast" (https://web.archive.org/web/20160702204820/http://www.dartlang.org/slides/2012/10/jsconfeu/javascript-as-compilation-target-florian-loitsch.pdf) (PDF). Dartlang.org. Archived from the original (http://www.dartlang.org/slides/2012/10/jsconfeu/javascript-as-compilation-target-florian-loitsch.pdf) (PDF) on 2016-07-02. Retrieved 2013-08-18.
25. "An Introduction to the dart:io Library" (http://www.dartlang.org/articles/io/). *Dartlang.org*. Retrieved 2013-07-21.
26. "Flutter FAQ" (https://flutter.io/faq/#how-does-flutter-run-my-code-on-ios). *flutter.io*. How does Flutter run my code on iOS?. Retrieved 2016-10-02.
27. "Announcing Dart 2.6 with dart2native: Compile Dart to self-contained, native executables" (https://medium.com/dartlang/dart2native-a76c815e6baf). 8 November 2019. Retrieved 2019-12-06.
28. "The Essence of Google Dart: Building Applications, Snapshots, Isolates" (https://www.infoq.com/articles/google-dart/). *InfoQ*. Retrieved 2021-08-29.
29. Moore, Kevin (February 23, 2018). "Dart2 Breaking Change: Removing web support for dart:mirrors and dart:isolate" (https://groups.google.com/a/dartlang.org/d/msg/misc/djfFMNCWmkE/F7WE8a0JAwAJ). *Google Groups*.
30. "Flutter - Beautiful native apps in record time" (https://flutter.dev/). *flutter.dev*.
31. "FAQ" (https://flutter.dev/docs/resources/faq). *flutter.dev*. Retrieved 2021-08-29.
32. "Firebase" (https://flutter.dev/docs/development/data-and-backend/firebase). *flutter.dev*. Retrieved 2021-08-29.
33. "Deployment" (https://angulardart.dev/guide/deployment). *angulardart.dev*.
34. Ladd, Seth (2013-03-28). "Dart News & Updates: Why dart2js produces faster JavaScript code from Dart" (http://news.dartlang.org/2013/03/why-dart2js-produces-faster-javascript.html). *News.dartlang.org*. Retrieved 2013-07-21.
35. "Dart Performance" (https://web.archive.org/web/20170103041945/http://www.dartlang.org/performance/). *Dartlang.org*. Archived from the original (http://www.dartlang.org/performance/) on 2017-01-03. Retrieved 2013-07-21.

36. "Google Releases Dart Editor for Windows, Mac OS X, and Linux" (https://web.archive.org/web/20131203024218/http://dartr.com/google-releases-dart-editor/). Archived from the original (http://dartr.com/google-releases-dart-editor/) on 2013-12-03. Retrieved 2011-11-29.
37. "Dart plugin for Eclipse is Ready for Preview" (https://news.dartlang.org/2012/08/dart-plugin-for-eclipse-is-ready-for.html).
38. Ladd, Seth (2015-04-30). "The present and future of editors and IDEs for Dart" (http://news.dartlang.org/2015/04/the-present-and-future-of-editors-and.html). *Dart News & Updates*. Retrieved 2015-05-18.
39. "JetBrains Plugin Repository : Dart" (http://plugins.intellij.net/plugin/?idea&id=6351). *Plugins.intellij.net*. Retrieved 2013-07-21.
40. "Dart Tools" (https://dart.dev/tools). *dart.dev*. Retrieved 2016-11-15.
41. Beaufort, François. "The chromium team is currently actively working" (https://plus.google.com/+FrancoisBeaufort/posts/giSLiGvA4ye).
42. "A Chrome app based development environment" (https://github.com/dart-lang/spark). *GitHub*. 26 October 2021.
43. "Spark, A Chrome App from Google is an IDE for Your Chromebook" (https://www.chromestory.com/2013/11/spark-chrome-app-google-ide-chromebook/). November 22, 2013.
44. Saroop, Sri. "Chrome Dev Editor: Announcements" (https://plus.google.com/+SriSaroop/posts/6EwgknKpesS).
45. "Chrome Dev Editor is a developer tool for building apps on the Chrome platform: Chrome Apps and Web Apps, in JavaScript or Dart. (NO LONGER IN ACTIVE DEVELOPMENT) - googlearchive/chromedeveditor" (https://github.com/googlearchive/chromedeveditor). July 29, 2019 – via GitHub.
46. Ladd, Seth (2015-05-06). "Announcing DartPad: A friction-free way to explore Dart code" (http://news.dartlang.org/2015/05/announcing-dartpad-friction-free-way-to.html). *Dart News & Updates*. Retrieved 2015-05-18.
47. "Bringing SIMD to the web via Dart" (https://web.archive.org/web/20160702204805/https://www.dartlang.org/slides/2013/02/Bringing-SIMD-to-the-Web-via-Dart.pdf) (PDF). Archived from the original (https://www.dartlang.org/slides/2013/02/Bringing-SIMD-to-the-Web-via-Dart.pdf) (PDF) on 2016-07-02.
48. "Algol Family" (http://c2.com/cgi/wiki?AlgolFamily). *c2.com*.
49. Bracha, Gilad; Griswold, David (September 1996). "Extending the Smalltalk Language with Mixins" (http://ftp.linux62.org/~glibersat/publis/p331-bracha.pdf) (PDF). *OOPSLA Workshop*. OOPSLA.
50. "The Essence of Google Dart: Building Applications, Snapshots, Isolates" (http://www.infoq.com/articles/google-dart/). *InfoQ*.
51. Bracha, Gilad; Ungar, David (2004). "Mirrors: design principles for meta-level facilities of object-oriented programming languages" (http://ftp.linux62.org/~glibersat/publis/p331-bracha.pdf) (PDF). *ACM SIGPLAN Notices*. ACM. **39** (10): 331–344. doi:10.1145/1035292.1029004 (https://doi.org/10.1145%2F1035292.1029004). Retrieved 15 February 2014.

Bibliography

- Walrath, Kathy; Ladd, Seth (March 7, 2012). *What is Dart?* (http://shop.oreilly.com/product/0636920025887.do) (1st ed.). O'Reilly Media. p. 20. ISBN 978-14493-32327.
- Walrath, Kathy; Ladd, Seth (November 7, 2012). *Dart: Up and Running* (http://shop.oreilly.com/product/0636920025719.do) (1st ed.). O'Reilly Media. p. 144. ISBN 978-1449330897.
- Buckett, Chris (December 28, 2012). *Dart in Action* (1st ed.). Manning Publications. p. 475. ISBN 978-1617290862.

External links

- Official website (https://dart.dev)
- DartPad (https://dartpad.dev)