1. Đặt vấn đề

Trong những năm gần đây, xu hướng phát triển ứng dụng đa nền tảng ngày càng phổ biến do nhu cầu triển khai đồng thời trên nhiều hệ điều hành khác nhau như Android, iOS, Windows và Web.

Hình 1.1: Framework Flutter

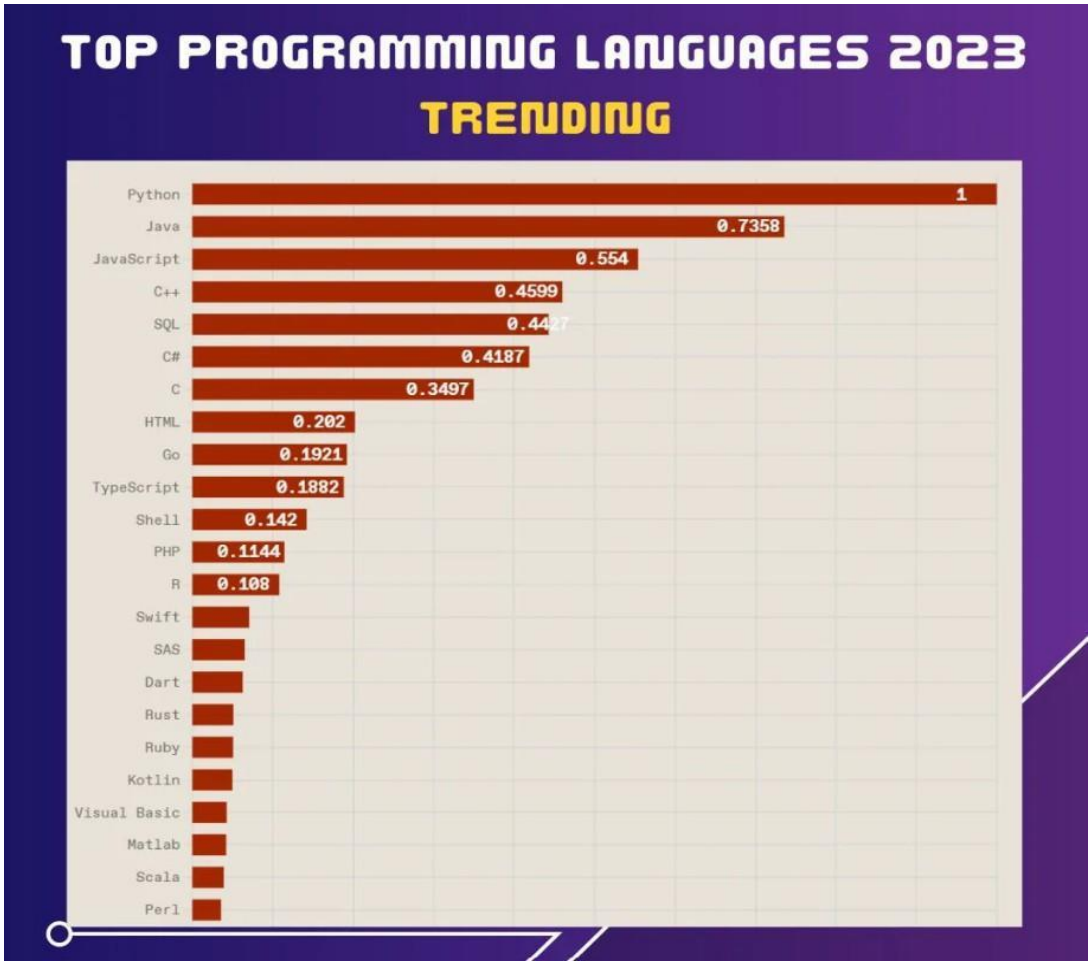
Flutter – một framework được phát triển bởi Google - cùng với ngôn ngữ Dart, đã trở thành một trong những lựa chọn hàng đầu cho phát triển ứng dụng đa nền tảng. Với khả năng hỗ trợ giao diện người dùng phong phú, hiệu suất cao và công nghệ "hot reload", Flutter đã nhanh chóng chiếm lĩnh thị trường và được cộng đồng phát triển ứng dụng đón nhận rộng rãi.

Trong bối cảnh nhiều công nghệ như React Native, Xamarin hay NativeScript liên tục được phát triển để hỗ trợ xây dựng ứng dụng đa nền tảng, việc lựa chọn một framework đáp ứng đồng thời các tiêu chí về hiệu suất, khả năng tuỳ biến giao diện và tương thích hệ điều hành vẫn là thách thức lớn. Flutter nổi lên như một giải pháp hiện đại, khắc phục nhiều hạn chế của các nền tảng trước nhờ kiến trúc widget thống nhất, khả năng kết xuất (rendering) mạnh mẽ và hiệu năng tiệm cận native.

2. GIỚI THIỆU VỀ NGÔN NGỮ LẬP TRÌNH DART

Dart, một ngôn ngữ lập trình đa năng do Google phát triển, đã chuyển mình từ một ứng cử viên thay thế JavaScript thành một trụ cột quan trọng trong phát triển ứng dụng đa nền tảng hiện đại. Được công bố rộng rãi vào năm 2011, Dart giờ đây hỗ trợ các ứng dụng trên web, di động, máy tính để bàn và phía máy chủ, nhờ vào sự kết hợp với Flutter cùng bộ tính năng mạnh mẽ. Phần này sẽ giới thiệu nguồn gốc, các mốc phát triển quan trọng và vai trò của Dart trong việc giúp các nhà phát triển tạo ra ứng dụng đa nền tảng hiệu suất cao từ một mã nguồn duy nhất.

Hình 2.1: Ngôn ngữ lập trình Dart

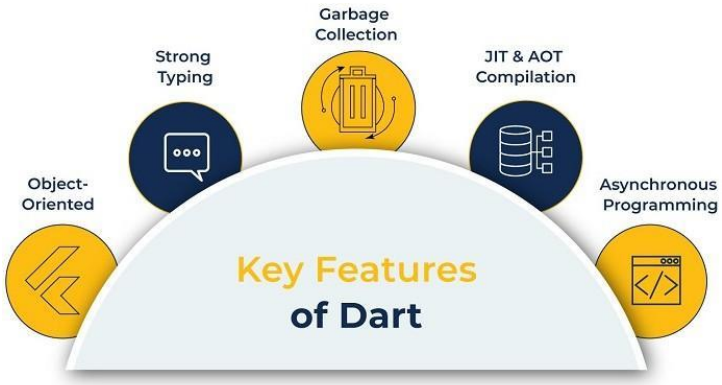


Hình 2.3: Thống kê các ngôn ngữ lập trình phổ biến năm 2023

Với vai trò mở rộng trong nhiều lĩnh vực và sự hỗ trợ từ cộng đồng, Dart hứa hẹn sẽ tiếp tục là một công cụ không thể thiếu cho các nhà phát triển trong tương lai.

2.1 Đặc điểm của ngôn ngữ lập trình Dart

Dart là một ngôn ngữ lập trình hiện đại. Ngôn ngữ này chứng tỏ vị thế nổi bật trong phát triển ứng dụng đa nền tảng nhờ khả năng tích hợp chặt chẽ với *Flutter* – framework giao diện người dùng (UI) do Google phát triển. Sự tích hợp đó cho phép **Dart** tạo ra giao diện mượt mà và hiệu năng cao trên thiết bị di động, trình duyệt web và máy tính để bàn. Ngoài ưu thế tích hợp, cú pháp gần gũi với C, Java và JavaScript giúp lập trình viên đã quen các ngôn ngữ này nhanh chóng nắm bắt cú pháp **Dart**.



Hình 2.4: Các đặc điểm chính của ngôn ngữ lập trình Dart

Khả năng thích ứng đa nền tảng tiếp tục được củng cố bởi hệ thống biên dịch linh hoạt của **Dart**. Cơ chế *ahead-of-time* (AOT) tối ưu hiệu suất thực thi, trong khi *just-in-time* (JIT) hỗ trợ tính năng *hot reload* giúp rút ngắn vòng lặp chỉnh sửa–kiểm thử. Kết hợp với kiểu tĩnh an toàn và bộ thư viện phong phú, tập đặc điểm đó định vị **Dart** thành lựa chọn lý tưởng cho nhà phát triển đang tìm kiếm một ngôn ngữ duy nhất nhưng đủ mạnh để triển khai ứng dụng trên mọi nền tảng. Những đặc điểm nổi bật dưới đây đã giúp Dart trở thành một lựa chọn lý tưởng cho các nhà phát triển ứng dụng:

- **Lập trình bất đồng bộ (Asynchronous Programming):** Dart hỗ trợ lập trình

bất đồng bộ thông qua các từ khóa `async` và `await`, cho phép xử lý các tác vụ không đồng bộ như gọi API, tải dữ liệu, hoặc xử lý sự kiện mà không làm gián đoạn giao diện người dùng. Tính năng này rất quan trọng khi phát triển ứng dụng đa nền tảng, vì nó giúp xây dựng các giao diện động và phản hồi nhanh.Tính năng này giúp Dart trở thành lựa chọn lý tưởng cho các ứng dụng đòi hỏi hiệu suất cao và giao diện mượt mà.

Ví dụ, một hàm bất đồng bộ trong Dart có thể được viết như sau:

```
Future<String> fetchData() async {
    await Future.delayed(Duration(seconds: 2));
    return "Loading data successfully!";
}
```

- **Hot Reload:** Hot Reload là một trong những tính năng nổi bật nhất khi Dart kết hợp với Flutter, mang lại các tiện ích như :
- **Mã nguồn mở (Open Source):** Dart được phát hành dưới giấy phép BSD-style, nhờ đó cộng đồng lập trình viên toàn cầu có thể trực tiếp tham gia phát triển, báo lỗi và đề xuất cải tiến.
- **Kiểu tĩnh (Statically Typing):** Dart sử dụng hệ thống kiểu dữ liệu tĩnh với thiết kế âm thanh (sound type system)
- **Tính di động (Portability):** Dart được thiết kế để trở thành ngôn ngữ **"write once, run anywhere"**, cho phép biên dịch mã nguồn sang nhiều nền tảng khác nhau mà không cần thay đổi logic chính.
- **Công cụ hỗ trợ mạnh mẽ (Productive Tooling):** Dart đi kèm với các công cụ phát triển mạnh mẽ, bao gồm DartPad (một trình biên tập trực tuyến cho phép viết và chạy mã Dart mà không cần cài đặt), và tích hợp tốt với các IDE như Visual Studio Code, IntelliJ IDEA, và Android Studio. Các công cụ này cung cấp tính năng gợi ý mã, kiểm tra lỗi thời gian thực, và gỡ lỗi, giúp

tăng năng suất lập trình. Ngoài ra, Dart còn có hệ thống quản lý gói (package manager) gọi là `pub`, giúp dễ dàng quản lý và tích hợp các thư viện bên ngoài.

- **Thu gom rác (Garbage Collection):** Dart có hệ thống quản lý bộ nhớ tự động thông qua thu gom rác, giúp giảm thiểu lỗi liên quan đến bộ nhớ như rò rỉ bộ nhớ (memory leak). Hệ thống này tự động giải phóng bộ nhớ không còn sử dụng, đảm bảo hiệu suất ổn định cho ứng dụng, đặc biệt là các ứng dụng lớn và phức tạp. Theo bài kiểm tra của Google, Dart GC giảm 50% thời gian tạm

- **Nền tảng cho Flutter:** Dart là ngôn ngữ chính thức của Flutter, một framework của Google để xây dựng ứng dụng giao diện người dùng đa nền tảng. Sự kết hợp giữa Dart và Flutter mang lại hiệu suất cao nhờ biên dịch AOT, cùng với khả năng tạo giao diện đẹp mắt và đồng nhất trên các nền tảng. Tính năng Hot Reload và lập trình bất đồng bộ của Dart đặc biệt hữu ích trong Flutter, giúp lập trình viên phát triển ứng dụng nhanh chóng và hiệu quả.
- **Hỗ trợ lập trình hướng đối tượng (Object-Oriented Programming):** Dart là một ngôn ngữ hướng đối tượng hoàn chỉnh, hỗ trợ các khái niệm như lớp, giao diện, mixin, và lớp trừu tượng. Điều này cho phép lập trình viên xây dựng các ứng dụng có cấu trúc rõ ràng và dễ bảo trì.

Lựa chọn ngôn ngữ lập trình phù hợp là yếu tố then chốt bảo đảm hiệu suất và khả năng mở rộng của ứng dụng. Sự lựa chọn này thường được cân nhắc giữa hai ngôn ngữ phổ biến là Dart và JavaScript, mỗi ngôn ngữ sở hữu những đặc điểm riêng. Các đặc điểm ấy sẽ được trình bày trong Bảng so sánh các tiêu chí giữa Dart và JavaScript như sau:

Tiêu chí	Dart	JavaScript
Kiểu dữ liệu	Kiểu tĩnh (Statically Typed), hỗ trợ Null Safety	Kiểu động (Dynamically Typed)
Thu gom	Bộ thu gom rác thế hệ	Bộ thu gom rác incremental

rác	(Generational GC)	truyền thống
Biên dịch	AOT và JIT compilation, hỗ trợ native	Chạy trực tiếp trên trình duyệt (interpreted)
Công cụ hỗ trợ	DartPad, Flutter DevTools, Analyzer	Chrome DevTools, VSCode, Node.js Tools
Tính di động	Web, mobile, desktop, server, IoT, Fuchsia OS	Web, server (Node.js)
Hiệu suất	Native tốc độ cao với AOT, nhanh hơn trong mobile apps	Tối ưu tốt trên trình duyệt, nhưng mobile native chậm hơn Dart+Flutter

Bảng 2.1: Bảng so sánh các đặc điểm giữa Dart và JavaScript

2.2 Các kiểu dữ liệu

Dart hỗ trợ các kiểu dữ liệu cơ bản như **bool, int, double, String, List, Set** và **Map**. Mỗi kiểu dữ liệu biểu diễn một loại thông tin khác nhau, có miền giá trị và đặc điểm riêng.

2.2.1 Kiểu bool

Dart cung cấp hỗ trợ sẵn có cho kiểu dữ liệu Boolean. Kiểu dữ liệu Boolean trong DART chỉ hỗ trợ hai giá trị true và false. Từ khóa bool được sử dụng để biểu diễn một ký tự Boolean trong DART.

2.2.2 Kiểu int

int (viết tắt của integer) là kiểu dữ liệu số nguyên trong Dart, dùng để biểu diễn các số nguyên (không có phần thập phân).

2.2.3 Kiểu double

double là kiểu dữ liệu số thực dấu phẩy động trong Dart, dùng để biểu diễn các số có phần thập phân (số thực). Kiểu `double` chiếm 64-bit và tuân theo tiêu chuẩn IEEE 754 (độ chính xác kép).

Kiểu dữ liệu	Mô tả
bool	Kiểu logic (Boolean), chỉ có hai giá trị: `true` (đúng) hoặc `false` (sai).
int	Kiểu số nguyên (integer), biểu diễn các số nguyên (có dấu) trong khoảng 64-bit.
double	Kiểu số thực dấu phẩy động (double precision), độ chính xác kép 64-bit, biểu diễn số thực (số có phần thập phân).
String	Kiểu chuỗi ký tự (string), biểu diễn một dãy các ký tự Unicode (UTF-16). Khai báo bằng cặp dấu nháy đơn hoặc nháy kép.
List (Danh sách)	Cấu trúc danh sách (list) các phần tử, tương tự mảng một chiều. Các phần tử có thứ tự tuyến tính và được truy cập bằng chỉ số (index).
Set (Tập hợp)	Cấu trúc tập hợp (set) các phần tử không trùng lặp. Không duy trì thứ tự cố định của phần tử, dùng để lưu các giá trị duy nhất.
Map (Ánh xạ)	Cấu trúc ánh xạ (map) gồm các cặp khóa - giá trị (key - value). Mỗi khóa ánh xạ tới một giá trị, cho phép tra cứu giá trị thông qua khóa.

Bảng 3.1: Mô tả các kiểu dữ liệu cơ bản trong Dart

2.3 Một số toán tử trên lớp, đối tượng

Toán tử	Ý nghĩa
`[]`	Truy cập phần tử trong danh sách hoặc map (theo chỉ số hoặc khóa)
`.`	Truy cập thuộc tính hoặc phương thức của đối tượng
`?.`	Truy cập thành viên nếu đối tượng không null; nếu đối tượng null thì trả về null
`as`	Ép kiểu đối tượng (cú pháp: `obj as MyClass`)

`is`	Kiểm tra kiểu (đúng nếu đối tượng là kiểu chỉ định)
`is!`	Kiểm tra phủ định kiểu (đúng nếu đối tượng không phải kiểu chỉ định)

Bảng 2.3: Các toán tử truy cập lớp/đối tượng trong Dart.

2.4 Một số widgets phổ biến trong Flutter

Đây là các thành phần cơ bản giúp tạo cấu trúc ứng dụng như nút, biểu tượng, trường nhập liệu, menu...

- **Button:** Buttons cung cấp cho người dùng khả năng thực hiện các hành động, đưa ra lựa chọn, gửi biểu mẫu, lưu dữ liệu, mở trang mới,... bằng cách click vào nó. Nút được phân loại thành nhiều loại: **Elevated Button**, **Floating Action Button**, **Outlined Button**, **Icon Button**, **Text Button**,...

  Ví dụ:

```
ElevatedButton(
    onPressed: () {
        print('Clicked');
    },
    style:             ElevatedButton.styleFrom(
        backgroundColor: Colors.blue,
    ),
    child: Text('Login'),
),
```

  Trong mỗi Widget lại có các properties riêng biệt, giúp cho bản thân nó trở nên sinh động hơn khi hiển thị ra giao diện.

- **TextField:** Cho phép người dùng nhập văn bản. Được sử dụng rộng rãi trong các biểu mẫu đăng nhập, đăng ký, tìm kiếm, bình luận,...

  Ví dụ:

```
TextField(
  decoration:    InputDecoration(    border:
    OutlineInputBorder(), labelText: 'Họ tên',
    hintText: 'Nhập họ và tên',
```

```
        prefixIcon: Icon(Icons.person),
      ),
    ),
```

Thuộc tính	Mô tả thuộc tính
autofocus	Nhận vào giá trị boolean xác định nút có được focus mặc định khi hiển thị hay không
	Xác định nội dung của nút có bị cắt (clip) nếu vượt quá kích thước không
focusNode	Đại diện cho node focus của widget
ButtonStyle	Xác định kiểu hiển thị (style) của nút
onLongPress	Hành động sẽ thực hiện khi người dùng nhấn giữ nút
enabled	Nhận vào giá trị boolean xác định nút có hoạt động hay không
hashcode	Xác định mã băm (hashcode) của nút
Key	Điều khiển cách một widget thay thế widget khác trong cây widget
onFocusChanged	Hàm sẽ được gọi khi focus của nút thay đổi
onHover	Hành động được thực hiện khi người dùng di chuột qua nút

Bảng 2.4: Các thuộc tính của ElevatedButton trong Flutter

3. TÀI LIỆU THAM KHẢO

[1] Wikipedia. Dart (programming language). https://en.wikipedia.org/wiki/Dart_(programming_language)

[2] A. W. Junaid, "History and Evolution of Dart Programming Language," *awjunaid.com*, Sep. 13, 2023. [Online]. https://awjunaid.com/dar t/history-and-evolution-of-dart-programming-language/.

[3] DartPad. dart.dev. https://dart.dev/tools/dartpad

[4] GeeksforGeeks (2025). Flutter Tutorial. https://www.geeksforgeeks.org/flutter-tutorial/

[5] Rahul, I. *All In One Flutter Book*. GitHub repository. https://github.com/irahulcse/All_In_One_Flutter_Book

[6] Dart Tutorial - Learn Dart Programming. OOP in Dart. https://dart-tutorial.com/object-oriented-programming/oop-in-dart/ https://github.com/irahulcse/All In One Flutter Book

[7] blup.in (2023). A Brief History of Flutter. https://www.blup.in/blog/a-brief-history-of-flutter

[8] Flutter Docs (2025). Flutter architectural overview.

https://docs.flutter.dev/resources/architectural-overview

[9] Bailey, T., & Biessek, A. (2021). *Flutter for Beginners – Second Edition: An introductory guide to building cross-platform mobile applications with Flutter 2.5 and Dart*.