**Tìm hiểu công nghệ Flutter – lập trình di động**

**Flutter là gì?**

Flutter là một framework phát triển ứng dụng di động được Google phát triển. Nó cho phép lập trình viên tạo ra các ứng dụng di động cho cả Android và iOS từ cùng một codebase (chỉ cần viết mã một lần và có thể chạy trên cả hai hệ điều hành). Flutter sử dụng ngôn ngữ lập trình Dart và cung cấp một bộ công cụ giao diện người dùng phong phú, giúp tạo ra các ứng dụng với giao diện đẹp mắt và hiệu suất cao.

Flutter có thể được xây dựng bằng bất cứ trình soạn thảo hay IDE nào. Được cài phổ biến trên Android Studio, Visual Studio Code hoặc IntelliJ.

**Tầm quan trọng của Flutter**

Flutter giúp lập trình viên viết mã một lần và có thể chạy trên nhiều nền tảng khác nhau ví dụ di động (android và IOS), web, desktop. Điều này giảm thời gian, công sức và đảm bảo tính nhất quán của giao diện.

Flutter sử dụng ngôn ngữ lập trình Dart và cơ chế biên dịch Ahead-of-time(AOT) giúp ứng dụng chạy nhanh, mượt mà và tối ưu hiệu suất tốt trên các thiết bị di động

Công cụ giao diện widget phong phú, có khả năng tuỳ biến, giúp cho người lập trình tạo ra giao diện người dừng đẹp mắt hơn, mang lại trải nghiệm người dùng tốt hơn

**Ngôn ngữ Dart (Sang, Hải)**

Dart, là một ngôn ngữ lập trình đa mục đích mã nguồn mở, đã được đặt nền móng bởi Google. Đây là một khía cạnh của ngôn ngữ lập trình hướng đối tượng, đặc trưng bởi cú pháp kiểu C. Tinh thần lập trình hướng đối tượng thể hiện qua việc hỗ trợ giao diện và lớp, mở ra khả năng sáng tạo không giới hạn khi đặt lên bàn cân với những ngôn ngữ khác. Sự đa dạng của Dart thể hiện thông qua khả năng phát triển ứng dụng web, di động, máy chủ và máy tính để bàn.

Những đặc điểm nổi bật của ngôn ngữ Dart:

1. Cú pháp đơn giản và dễ học

Dart có cú pháp tương tự như các ngôn ngữ lập trình C-style khác như Java và JavaScript, giúp lập trình viên dễ dàng tiếp cận và bắt đầu sử dụng Dart nhanh chóng. Điều này giúp giảm thời gian học tập và cho phép các lập trình viên nhanh chóng bắt đầu xây dựng ứng dụng.

2. Hiệu suất cao

Dart được thiết kế để đạt hiệu suất cao khi chạy trên mọi thiết bị và nền tảng. Điều này đạt được nhờ vào khả năng biên dịch Ahead-Of-Time (AOT) của Dart, một phương pháp cho phép biên dịch mã nguồn thành mã máy gốc trước khi ứng dụng được khởi chạy, giúp cải thiện đáng kể tốc độ và hiệu suất của ứng dụng. Điều này đặc biệt quan trọng đối với các ứng dụng di động và ứng dụng yêu cầu hiệu suất cao.

3. Hỗ trợ lập trình hướng đối tượng (OOP)

Dart là một ngôn ngữ lập trình hướng đối tượng (OOP) với đầy đủ các đặc điểm như class-based, tính kế thừa và đa hình. Điều này giúp các lập trình viên tổ chức mã nguồn một cách rõ ràng và dễ quản lý. Lập trình hướng đối tượng cũng giúp tái sử dụng mã và phát triển các ứng dụng phức tạp hơn một cách hiệu quả.

4. Null Safety

Một trong những tính năng quan trọng và tiên tiến của Dart là null safety, giúp ngăn chặn các lỗi runtime liên quan đến null bằng cách đảm bảo rằng các biến không thể chứa giá trị null trừ khi được khai báo rõ ràng. Null safety không chỉ cải thiện độ an toàn của mã nguồn mà còn giúp các nhà phát triển tránh được những lỗi phổ biến và khó phát hiện. Điều này đặc biệt hữu ích trong các dự án lớn và phức tạp, nơi việc kiểm soát các giá trị null là rất quan trọng để đảm bảo chất lượng và hiệu suất của ứng dụng

5. Khả năng biên dịch nhanh

Dart hỗ trợ cả hai phương pháp biên dịch là Ahead-Of-Time (AOT) và Just-In-Time (JIT), giúp tối ưu hóa quy trình phát triển ứng dụng.

6. Phát triển đa nền tảng

Dart, kết hợp với Flutter, cho phép phát triển ứng dụng đa nền tảng từ một cơ sở mã nguồn duy nhất. Các nhà phát triển có thể xây dựng ứng dụng cho iOS, Android, web và các ứng dụng desktop mà không cần viết mã lập trình riêng biệt cho từng nền tảng.

7. Tính năng hot reload

Một trong những tính năng nổi bật và được yêu thích nhất của Dart là hot reload. Khi sử dụng Flutter, Dart hỗ trợ tính năng hot reload, cho phép lập trình viên ngay lập tức thấy các thay đổi trong mã nguồn mà không cần phải khởi động lại ứng dụng, tăng tốc quá trình phát triển và gỡ lỗi. Dart cũng tích hợp tốt với các IDE phổ biến như IntelliJ IDEA, Android Studio và Visual Studio Code, cung cấp các tiện ích như tự động hoàn thành mã, gỡ lỗi và phân tích mã nguồn.

8. Tích hợp tốt với các công cụ phát triển

Dart có khả năng tích hợp tốt với nhiều công cụ phát triển hiện đại ngày này như Visual Studio Code, Android Studio và IntelliJ IDEA. Điều này giúp các lập trình viên dễ dàng viết code, debug và triển khai mã nguồn một cách hiệu quả. Các công cụ hỗ trợ này cũng cung cấp các tính năng như gợi ý câu lệnh code, kiểm tra lỗi và quản lý dự án, giúp tăng cường hiệu suất làm việc

Ứng dụng của ngôn ngữ Dart:

1. Phát triển ứng dụng di động

Dart trở nên đặc biệt nổi tiếng nhờ Flutter, framework UI do Google phát triển. Flutter cho phép các nhà phát triển xây dựng các ứng dụng di động đa nền tảng (cross-platform) với hiệu suất cao, giao diện người dùng đẹp mắt và trải nghiệm mượt mà. Bằng cách sử dụng một cơ sở mã nguồn duy nhất, các nhà phát triển có thể triển khai ứng dụng của mình trên cả Android và iOS, giúp tiết kiệm thời gian và công sức so với việc phải viết mã riêng cho từng nền tảng.

2. Phát triển ứng dụng web

Dart ban đầu được thiết kế để thay thế JavaScript trong phát triển web. Mặc dù không đạt được sự phổ biến như mong đợi ban đầu, Dart vẫn là một lựa chọn mạnh mẽ cho phát triển ứng dụng web. Với bộ công cụ như DartPad và các thư viện như AngularDart, các nhà phát triển có thể xây dựng các ứng dụng web tương tác và hiệu quả.

3. Phát triển ứng dụng desktop

Cùng với sự phát triển của Flutter, Dart không chỉ giới hạn ở các ứng dụng di động và web mà còn mở rộng sang phát triển ứng dụng máy tính để bàn. Flutter for Desktop cho phép các nhà phát triển tạo ra các ứng dụng chạy trên Windows, macOS và Linux từ cùng một cơ sở mã nguồn

4. Phát triển server-side

Dart không chỉ mạnh mẽ trong phát triển giao diện người dùng mà còn tỏ ra rất hiệu quả trong việc xây dựng các ứng dụng phía máy chủ. Thư viện dart:io và các gói mở rộng như Aqueduct giúp Dart xây dựng các ứng dụng máy chủ mạnh mẽ và hiệu quả. Dart có thể xử lý các yêu cầu HTTP, quản lý kết nối cơ sở dữ liệu và thực hiện các tác vụ phức tạp phía máy chủ, tương tự như Node.js nhưng tận dụng được các ưu điểm vượt trội của Dart.

5. Phát triển các công cụ

Dart còn được sử dụng để phát triển các công cụ và tiện ích hỗ trợ quá trình phát triển phần mềm. Ví dụ, các công cụ dòng lệnh (CLI) được viết bằng Dart có thể giúp tự động hóa các tác vụ, quản lý dự án và thực hiện các công việc khác một cách hiệu quả.

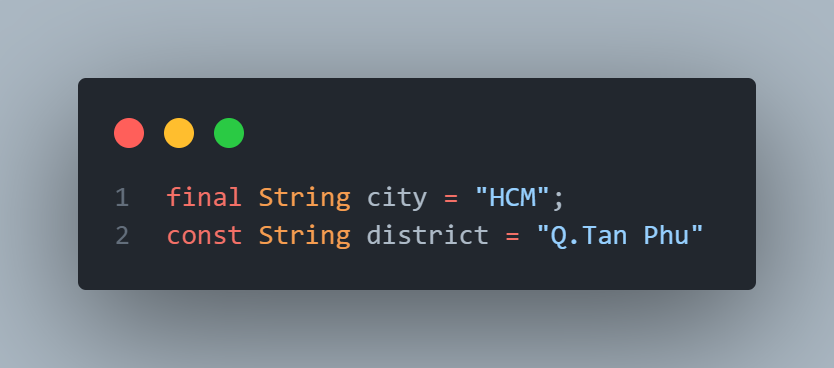
Cú pháp cơ bản của ngôn ngữ Dart

Khai báo biến bằng *var* và *dynamic*

*var* thường dùng khi không rõ kiểu dữ liệu, trình biên dịch sẽ tự động xác định dựa vào biến khởi tạo.

*Dynamic* kiểu dữ liệu này chấp nhận mọi kiểu dữ liệu được truyền vào.

Ngoài ra Dart vẫn có những kiểu dữ liệu như *String, int, double, bool, List, Set, Map, …*

Khai báo hằng số bằng *final* và *const*

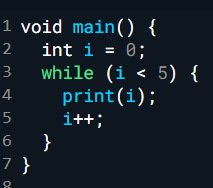
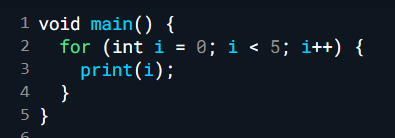
*final*: giá trị của biến chỉ được khởi tạo một lần.

*const*: biến phải được khởi tạo tại thời điểm compile và không thể thay đổi.

Câu lệnh điều kiện if else và switch-case cũng sẽ tương tự như ngôn ngữ lập trình khác.

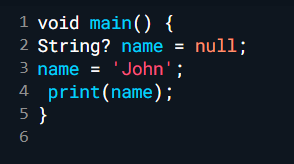


Và vòng lập for và while cũng sẽ tương tự như ngôn ngữ lập trình khác.



Null Safety

Dart hỗ trợ tính năng an toàn với null. Bạn cần khai báo kiểu dữ liệu nullable với dấu ? nếu biến có thể chứa giá trị null.



**Widget trong Flutter (Quí, Thịnh)**

* Widget là gì, cách hoạt động
* Widget cơ bản (động, tĩnh)
* Cách tạo và tuỳ chỉnh widget

**Quản lý trạng thái trong Flutter**

Quản lý trạng thái giúp kiểm soát và cập nhật lại giao diện thi thay đổi dữ liệu. Trong Flutter có những cách quản lý cơ bản là setState, Provider,…

setState

Là một phương thức được sử dụng để xây lại giao diện người dùng khi trạng thái của một StatefulWidget thay đổi.

ElevatedButton(

child: Text("Đổi màu"),

onPressed: (){

// Thay đổi trạng thái UI với các thành phần liên quan đến changedColor

setState(() {

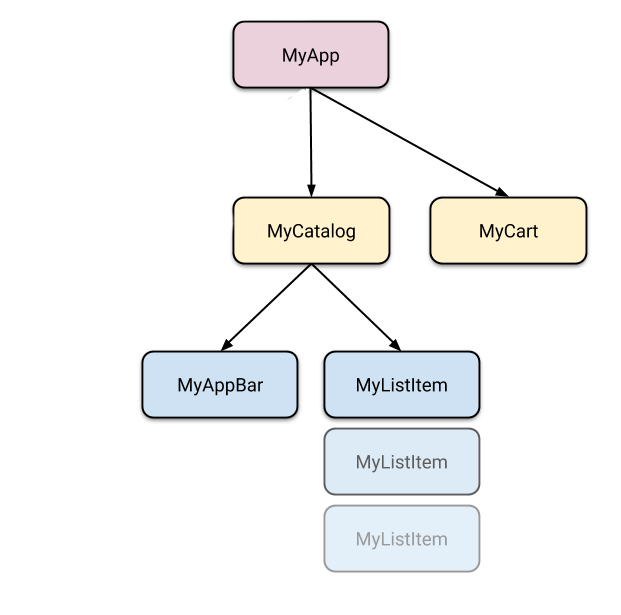
changedColor = ! changedColor;

});

},

),

Theo như đoạn code demo, ElevatedButton là một nút bấm với chức năng đổi màu, sau khi lắng nghe được sự kiện onPress hàm setState sẽ được gọi và thực thi cập nhật lại toàn bộ giao diện.

Provider

Cho phép các widget chia sẻ và cập nhật trạng thái mà không cần phải truyền dữ liệu qua nhiều cấp độ trong cây widget.

Ví dụ như hình bên khi chúng ta muốn cập nhật lại trạng thái của MyCart, các widget con (MyAppBar, MyListItem) sẽ tự động cập nhật lại để phản ánh trạng thái mới.

A screen shot of a phone

Description automatically generatedA screen shot of a phone

Description automatically generated

A diagram of a company

Description automatically generated

Provider tương tự như một chuỗi cung ứng:

Nhà cung cấp sẽ kế thừa lớp ChangeNotifier, mỗi khi có sự thay đổi thì nhà cung cấp sẽ gọi hàm notifyListeners() để chuyển xuống người dùng.

Nhà phân phối sẽ phân phối đến cho người thông qua ChangeNotifierProvider (chuyển 1 kênh phân phối) và MultiProvider (chuyển nhiều kênh phân phối).

Người sử dụng sẽ dùng context.watch<>() và context.read<>()

* **read():** chỉ truy cập trạng thái mà không theo dõi sự thay đổi, giúp tránh việc xây dựng lại không cần thiết.
* **watch():** theo dõi sự thay đổi của trạng thái và xây dựng lại widget khi trạng thái thay đổi.

Tóm lại:

**setState:** Đơn giản, nhanh chóng và thích hợp cho các ứng dụng nhỏ hoặc các trường hợp thay đổi trạng thái cục bộ trong một widget.

**Provider:** Mạnh mẽ hơn, phù hợp cho các ứng dụng phức tạp hoặc khi cần chia sẻ trạng thái giữa nhiều phần của ứng dụng.

Ngoài ra cũng sẽ có những phương pháp cải thiện từ 2 phương pháp cơ bản trên như Riverpod, Bloc,… những phương pháp cũng sẽ có ưu và nhược điểm riêng.

**Điều hướng trong Flutter**

Điều hướng là một phần quan trọng để chuyển đổi giữa các trang trong ứng dụng, sử dụng *Navigator* và *MaterialPageRoute* để thực hiện chuyển trang*.*

Navigator và MaterialPageRoute

Cú pháp: chuyển hướng trang từ FirstPage sang SecondPage.

A computer screen shot of a black rectangle with yellow and orange text

Description automatically generated

Sử dụng phương thức *push* trên đối tượng Navigator để đẩy widget mới của SecondPage vào stack, nó sẽ hiện đè lên trên trang trước đó.

*MaterialPageRoute* định nghĩa cách chuyển cảnh hiển thị trang mới.

Cú pháp: quay lại trang FirstPage

A black rectangular with yellow green and orange text

Description automatically generated

Sử dụng phương thức *pop* để xoá đi widget của SecondPage => Hiện lại FirstPage.