Báo Cáo Khóa học Futter

The Complete 2020 Flutter Development Bootcamp with Dart

Created by Dr. Angela Yu

Mục lục

[I. Dart 2](#_Toc37963097)

[1. Basic: 2](#_Toc37963098)

[2. Variable: 2](#_Toc37963099)

[3. Function: 3](#_Toc37963100)

[4. Access Modifier: 3](#_Toc37963101)

[5. Generic type: 3](#_Toc37963102)

[6. String: 3](#_Toc37963103)

[7. Random: 3](#_Toc37963104)

[8. Khác 4](#_Toc37963105)

[II. Flutter: 4](#_Toc37963106)

[1. Basic 4](#_Toc37963107)

[4. Statefull và Stateless Widgets: 5](#_Toc37963108)

[5. *build* method: 5](#_Toc37963109)

[6. Hot Reload và Hot Restart: 6](#_Toc37963110)

[5. Giúp giảm thời gian phát triển ứng dụng. 6](#_Toc37963111)

[III. Widgets: 6](#_Toc37963112)

[1. Tính chất chung: 6](#_Toc37963113)

[2. MaterialApp: 6](#_Toc37963114)

[3. Scaffold: 6](#_Toc37963115)

[4. Text 7](#_Toc37963116)

[5. Container (standart) 7](#_Toc37963117)

[6. Column/ Row(standart) 7](#_Toc37963118)

[7. Expanded: 7](#_Toc37963119)

[8. Icon 7](#_Toc37963120)

[9. Image: 8](#_Toc37963121)

[10. SizedBox: 8](#_Toc37963122)

[11. Safe Area: 8](#_Toc37963123)

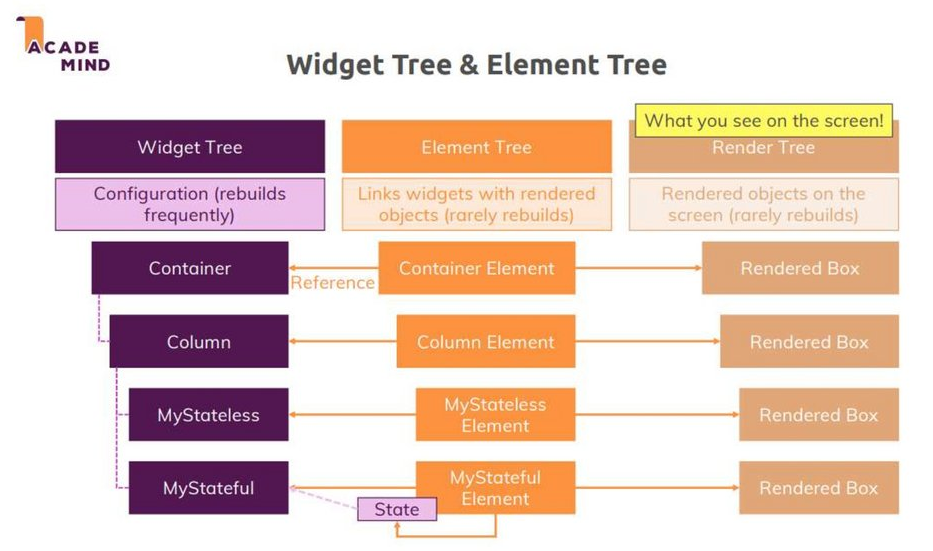
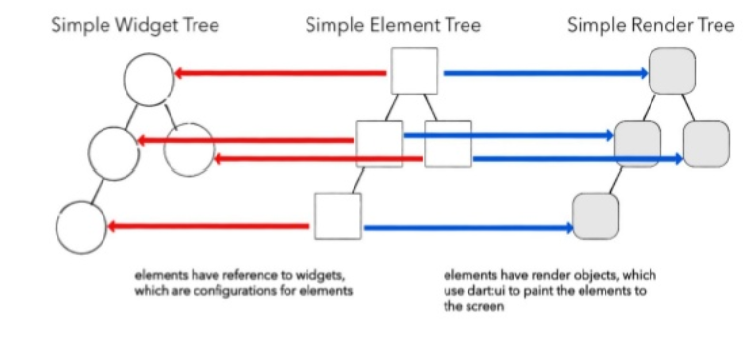
[IV. Tips 8](#_Toc37963124)

[1. Dealing with distractions(Sự xao lảng): 8](#_Toc37963125)

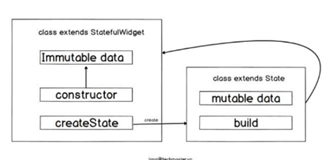
[V. Các website hữu ích và app tham khảo. 8](#_Toc37963126)

1. Dart
   1. Basic:
      * Everything is objects, kể cả function và null.
   2. Variable:
      * Là ngôn ngữ **static typing, x**ác định kiểu dữ liệu tại thời điểm **compile-time** ( Khác với **dynamic typing** (vd: JavaScript) xác định kiểu dữ liệu tại thời **runtime**.
      * So sánh với dynamic typing.
        1. Ngắn gọn: dynamic typing ngắn gọn hơn do không cần khai báo kiểu.
        2. Tính đúng đắn: trong static typing lỗi về kiểu (type errors) có thể được phát hiện tại thời điểm compile vì vậy chương trình sẽ an toàn và ít lỗi hơn.
        3. Hiệu năng: static typing sẽ nhanh hơn vì không mất thêm chi phí để kiểm tra kiểu lúc runtime.

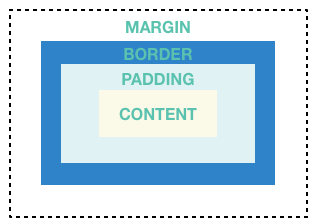
* Nếu muốn là dynamic ta có thể dùng từ khóa *dynamic.*
* Nếu khi khai báo *var* không có kiểu dữ liệu phía sau có nghĩa nó là *dynamic* hay kiểu trả về hoặc parameter của function không định nghĩa kiểu dữ liệu thì nó cũng là *dynamic.*
* final được khởi tạo 1 lần duy nhất (run time).
* const được khởi tạo lúc compile.
* Các property của class chỉ có thể là final mà k thể là const vì class được khởi tạo ở run time.
* Có thể tự suy ra (infer) kiểu dữ liệu từ giá trị nhận được.
* Khi khai báo ta nên khai báo kiểu dữ liệu cho dữ liệu.
* Hổ trợ top-level variable: Sử dụng ở bất kỳ đâu mà không cần thông qua class.
* Int và double đều là subtypes của num.
  1. Function:
     + => (fat arrow): được sử dụng khi thân hàm chỉ có 1 biểu thức (expression).
     + Hỗ trợ top-level function (vd *main()*): Có thể sử dụng ở bất kỳ đâu mà không cần thông qua tên class hay instance của class.
     + Có thể tạo 1 hàm bên trong 1 hàm khác (nested function).
     + *main* function: điểm bắt đầu khi chạy ứng dụng.
     + Optional named parameters:
       1. function({type variable}).
       2. Được dùng khi có nhiều parameter.
     + Optional positional parameters:
       1. Function([type variable]).
       2. Được dùng khi có thể không cần truyền giá trị cho parameter đó.
     + Default parameter values.
  2. Access Modifier:
     + Không hổ trợ public, protected, private.
     + Nếu identifier (variable, function) bắt đầu với dấu gạch dưới ‘\_’ thì nó private trong library của nó. Mỗi file .dart được coi là 1 library.
  3. Generic type:
     + Có thể xây dựng các hàm, lớp mà không làm việc trên 1 kiểu dữ liệu cụ thể.VD: Class<T,E> function<T>(T t).
  4. String:
     + Nên dùng dấu ngoặc đơn.
     + Interpolation: ‘text${variable}text’.
     + Để tạo multi-line:’’’ text /n text ‘’’
  5. Random:
     + Random().nextInt(max): trả ra giá trị từ 0 tới max – 1.
  6. Khác
     + Dùng == để so sánh 2 object.

1. Flutter:
   1. Basic
      * Everything is widgets.
   2. Widget tree
      * Là cấu trúc đại diện cho toàn bộ ứng dụng( giống DOM).
      * Các bước chạy chương trình.
      * 
        1. Gọi hàm runApp(), Flutter build một Widget tree chưa các **widget**.
        2. Các **Element** được tạo bởi các widget. Sauk hi widget được build, framework gọi hàm Widget.createElement(this).
        3. Element gọi phương thức createRenderObject() để tạo ra các **RenderObject**, các **RenderObject** này cấu thành Render Tree.
        4. Mỗi một **Element** tham chiếu đến một **Widget** và một **RenderObject**

* Lưu ý: StateObject thực sự được quản lý bởi Element, không phải widget. Flutter render dựa trên Elements và State Object, chứ nó không liên quan gì đến widget.
  1. Vì Sao cần đến 3 tree (Widget, Element, Render):
     + **Element** không giữ cấu hình widget, nhưng có thể nhìn cấu hình chi tiết thông qua việc tham chiếu đến widget tương ứng.
     + **RenderObject** chưa toàn bộ logic để **rendering** widget, nên nó khá nặng để khởi tạo.
     + Khi widget trong tree thay đổi, Flutter sử dụng Elements để so sánh với widget mới với RenderObject đã tồn tại, nếu type của widget không đổi, Flutter sẽ không tạo lại các RenderObject nặng nề đó, thay vào đó chỉ cập nhật mutable configuration.
     + Widget rất nhẹ để khởi tạo, nên nó được dùng để mô tả state hiện tại.
  2. Statefull và Stateless Widgets:
     + Stateless widgets:
       1. Không có state.
       2. Không chấp nhận sự thay đổi bên trong nó.
       3. Khi widget cha thay đổi nó sẽ thụ động thay đổi theo.
     + Stateful widgets:
       1. Mối quan hệ giữa state và statefulWidget



* + - 1. Được sử dụng khi cần có sự thay đổi giao diện linh hoạt.
      2. Bản thân statefulWidget không chứa dữ liệu mutable mà State mới là nơi chưa dữ liệu mutable.
      3. State:
         1. 
         2. Trạng thái: Đại diện cho dữ liệu thay đổi trong vòng đời ứng ưng.
         3. Khi dữ liệu của stateful widget thay đổi, UI vẽ lại widget để phản ánh state mới.
         4. Phần giao diện chỉ đại diện cho state tại 1 thời điểm, thay đổi state -> thay đổi giao diện.
      4. Tạo ra UI mà state có thể thay đổi.
      5. setState():
         1. Khích hoạt hàm build.
         2. Khi không có hàm này biến vẫn sẽ được thay đổi nhưng không hiển thị lên màng hình.
         3. Khi ta gọi setState, ta sẽ cập nhật biến, và những nơi sử dụng biến đó sẽ bị đánh dấu là dirty. Và khi build nó sẽ tìm vẽ lại những nơi bị đánh dấu là dirty.
  1. *build* method:
     + Trả về một widget object chứa thông tin cấu hình để flutter render.
     + Được gọi khi:
       1. StatelessWidget
          1. Lần đầu tiên được thêm vào tree.
          2. Khi widget cha thay đổi.
          3. Khi các giá trị từ nguồn khác thay đổi. (Ví dụ 1 widget kế thừa sẽ phụ thuộc vào các thay đổi từ nơi nó kết thừa). (?? Chưa hiểu).
       2. StatefulWidget
       3. Sau khi gọi initState.
       4. Sau khi gọi didUpdateWidget ( Được gọi nếu widget cha thay đổi và phải xây dựng lại widget).
  2. Hot Reload và Hot Restart:
     + Hot Reload
       1. Chỉ hoạt động trên Stateless hoặc statefull widgets.
       2. Chỉ gọi lại phương thức *build* của widgets.
       3. Sử dụng: Ctrl + S.
       4. Không mất dữ liệu (state) của app. (ví dụ: form, text, …).
       5. Giúp giảm thời gian phát triển ứng dụng.
     + Hot Restart
       1. Reset app (reset data (state) của app).
  3. Cách custom font:
     + 1. Import the font files.
       2. Định nghĩa font trong pubspec.
       3. Set font mặc định.
       4. Sử dụng font trong widget.
  4. Khác
     + Nên thêm dấu phẩy sau mỗi dấu đóng ngoặc đơn để dễ format code.
     + YAML (Ain’t Markup Language): Cẩn thận khi chỉnh sửa, đặc biệt là sử dụng dấu cách.
     + Assets: Thay vì để đường dẫn chính xác cả tên file ta có thể chỉ cần dùng tên folder.
     + Folder Icon:
       1. Android: app/src/main/res/…
       2. IOS: Runner/Assets.xcassets/Appicon.appiconset/…

1. Widgets:
   1. Tính chất chung:
      * Size của widget con sẽ phụ thuộc vào widget cha.
   2. MaterialApp:
      * Dùng phong cách thiết kế Material Design của Google.
      * Cấu hình top-level Navigator.
   3. Scaffold:
      * Implement các thiết kế cơ bản của Material Design.
      * Cung cấp API để hiển thị Drawer, Snack Bar, Bottom Sheet.
      * Chỉnh sửa theme.
      * Thuộc tính:
        1. *appBar*: Hiển thị appBar của app.
        2. *body*: The primary content của scaffold.
   4. Text
      * Mặc định align là góc trái trên cùng.
   5. Container (standart)
      * 
      * Nếu dùng decoration- (ví dụ: shape là circle) thì sẽ dùng min(height,width) của child.
      * Không có child: as big as possible.
      * Có child: điều chỉnh để vừa với child nếu widget cha không dùng constrain.
      * Khi dùng alignment hoặc dùng constraint expand: sẽ mở rộng hết cở so với max size của widget cha.
      * Vì container phước tạp nên tốn chi phí tính toán nếu chỉ dùng padding ta có thể dùng Widget Padding để giảm chi phí.
   6. Column/ Row(standart)
      * Mở rộng hết cở theo chiều dọc/ngang để đủ khoảng trống cho widget con. Nếu gặp giới hạn sẽ bị overflowed.
      * mainAxisAlignment: Sắp xếp children theo trục chính.
      * crossAxisAlignment:
        1. Sắp xếp children theo trục phụ.
        2. stretch: childer sẽ max size trục phụ.
        3. Để sử dụng crossAxisAlignment.end ta thêm Container với width max.
        4. baseLine: Các dòng chữ bằng nhau dù cho fontSize chênh lệch. (Phải có textBaseLine).
   7. Expanded:
      * Làm cho child của Row, Column, Flex chiếm khoảng trống nhiều nhất có thể ở trục chính.
      * Nếu nhiều child được expanded thì khoảng trống mỗi child chiếm được phụ thuộc vào flex.
   8. Icon
      * Là 1 vector.
      * Linh hoạt (flexible) hơn khi sử dụng Image.
        1. Có thể thay color, size mà không bị pixelated(thấy từng pixel riêng lẻ của hình khi zoom hình chất lượng thấp).
   9. Image:
      * AssetImage: là Image Provider khác với Image.asset là widget.
   10. SizedBox:
       * Thêm khoảng trống giữa 2 widget.
   11. Safe Area:
       * Đặt giao diện trong safe area ( không bao gồm notch(phần trên cùng của IOS và status bar của Android), interactive area(phần cuối cùng của IOS)).
2. Tips
   1. Dealing with distractions(Sự xao lảng):
      * Tìm khoảng thời gian rảnh và không gian yên tĩnh.
      * Đặt chế độ máy bay và đặt điện thoại ở nơi không nhìn thấy.
3. Các website hữu ích và app tham khảo.
   1. Appicon.co
   2. Icons8.com
   3. Canva.com
   4. Vecteezy.com
   5. Layout Cheat Sheet: [https://medium.com/flutter-community/flutter-layout-cheat-sheet-5363348d037ehttps://medium.com/flutter-community/flutter-layout-cheat-sheet-5363348d037e](https://medium.com/flutter-community/flutter-layout-cheat-sheet-5363348d037e)
   6. App History of EveryThing.