1. **Tìm hiểu về Hero Animations trong Flutter**



* **Giới thiệu về Hero animations**

Hero Animation là một widget giúp chúng ta tạo các hiệu ứng khi chuyển màn. Dưới đây là một ví dụ về Hero Animations.

* **Tạo Hero Animation cơ bản**

Hero Animations là một trong những hiệu ứng dễ làm nhất trong Flutter và không yêu cầu thiết lập nhiều. Nhìn vào ví dụ dưới đây, chúng ta có thể thấy rằng cùng một biểu tượng tồn tại trên cả hai trang. Tất cả những gì chúng ta cần là nói cho Flutter rằng cả hai được liên kết với nhau. Chúng tôi thực hiện việc này bằng cách bao bọc một widget như một biểu tượng trong Hero widget.

Hero(

tag: "DemoTag",

child: Icon(

Icons.add,

size: 70.0,

),

),

Chúng tôi cung cấp cho mỗi Hero widget một tên cụ thể. Điều này là cần thiết vì nếu chúng ta có thể có nhiều Hero trên cùng một màn hình và mỗi Hero wigdet sẽ có những điểm đến khác nhau

Bây giờ ứng dụng biết rằng có một Hero widget muốn chuyển đến trang tiếp theo. Bây giờ tất cả những gì chúng ta cần làm là xác định nơi mà Hero Widget sẽ bay tới.

Tất cả những gì chúng ta cần là tạo một Hero Widget ở màn hình thứ hai với tên giống với tên hero widget ở màn hình thứ nhất.

Hero(

tag: "DemoTag",

child: Icon(

Icons.add,

size: 150.0,

),

),

Hero Animations cho phép chúng ta tùy chỉnh các thông số để tạo ra các hoạt ảnh mà chúng ta mong muốn. Cùng xem cách làm như nào nhé.

Hero(

tag: "DemoTag",

child: Icon(

Icons.add,

size: 150.0,

),

placeholderBuilder: (context, widget) {

return Container(

height: 150.0,

width: 150.0,

child: CircularProgressIndicator(),

);

},

),

Flutter cho phép chúng ta thay đổi widget thực sự di chuyển từ màn hình này sang màn hình khác mà không thay đổi các widget mà chúng ta setup từ đầu trên cả hai màn hình. Dưới đây chúng tôi sẽ dùng biểu tượng tên lửa trong lúc di chuyển từ màn hình này sang màn hình khác thay cho icon "+"

Hero(

tag: "DemoTag",

child: Icon(

Icons.add,

size: 150.0,

),

flightShuttleBuilder: (flightContext, animation, direction,

fromContext, toContext) {

return Icon(FontAwesomeIcons.rocket, size: 150.0,);

},

),

Phương thức flightShuttleBuilder có 5 tham số và cho chúng ta hoạt ảnh cũng như hướng của hoạt ảnh . Hiện tại, kích thước biểu tượng tên lửa vẫn ở mức 150,0 cho cả hai hướng. Chúng ta có thể có các cấu hình khác nhau cho mỗi hướng bằng cách sử dụng tham số **direction** của phương thức.

// push : từ màn hình thứ nhất về màn hình thứ hai.

// pop: từ màn hình thứ hai quay lại màn hình thứ nhất.

if(direction == HeroFlightDirection.push) {

return Icon(

FontAwesomeIcons.rocket,

size: 150.0,

);

} else if (direction == HeroFlightDirection.pop){

return Icon(

FontAwesomeIcons.rocket,

size: 70.0,

);

}

* **Tạo hiệu ứng vuốt ngược giống với IOS bằng Hero Animation**

Mặc định, Hero Animation hoạt động khi người dùng sử dụng phím **back** chứ không hoạt động với hành động back swipe. Tuy nhiên chúng ta có thể tạo ra hiệu ứng này một cách dễ dàng với Hero Animations

1. **Tìm hiểu Flutter Opacity Widget**



* **Ẩn Widget (full opacity)**

Giả sử bạn có 1 list Widget được sắp xếp theo Column như sau:

class SomeWidget extends StatelessWidget {

@override

Widget build(BuildContext context) {

final widgets = [

MyWidget(Colors.green),

MyWidget(Colors.blue),

MyWidget(Colors.yellow),

];

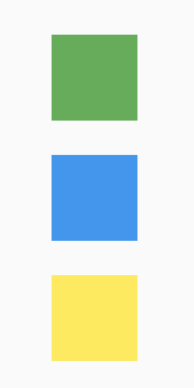
return Column(

children: widgets,

);

}

}



Thật dễ dàng để loại bỏ màu xanh lam chỉ bằng cách rebuild mà không dùng nó (comment code chỗ đó lại như sau):

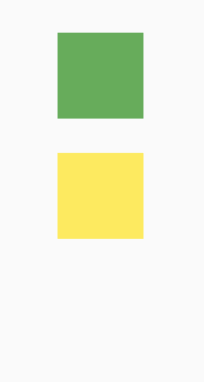
final widgets = [

MyWidget(Colors.green),

// MyWidget(Colors.blue),

MyWidget(Colors.yellow),

];



Nhưng điều đó đã khiến Widget màu vàng di chuyển lên để chiếm vị trí của Widget màu xanh lam. Nếu bạn muốn bố cục giữ nguyên và chỉ làm cho Widget màu xanh lam biến mất, bạn có thể bọc nó trong một **Opacity** widget.

final widgets = [

MyWidget(Colors.green),

Opacity(

opacity: 0.0,

child: MyWidget(Colors.blue),

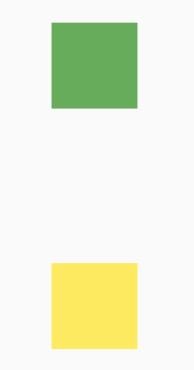
),

MyWidget(Colors.yellow),

];

Opacity bằng 0.0 nghĩa là nó hoàn toàn trong suốt. Nếu bạn muốn làm cho nó hoàn toàn transparent (nghĩa là không còn trong suốt (transparency)), bạn sẽ đặt opacity thành 1.0. Bất kỳ thứ gì trong khoảng từ 0.0 đến 1.0 đều làm cho Widget trở nên trong suốt dần.

Chạy đoạn code trên ta được kết quả sau. Widget màu xanh lam vẫn ở đó, nhưng bạn không thể nhìn thấy nó.



Opacity widget rất hữu ích nếu bạn muốn làm cho toàn bộ Widget trong suốt. Tuy nhiên có lúc bạn chỉ cần tạo một màu trong suốt duy nhất mà thôi. [Đây](https://www.youtube.com/watch?v=_jKL5IY3OOM) là một ví dụ về AppBar trong suốt:

