**BÁO CÁO TUẦN 4**

**Họ và tên:** Nguyễn Kỳ Anh  
**MSSV:** 20225793  
 **GR1**  
**Giáo viên hướng dẫn:** Nguyễn Đức Toàn

1. **Widget trong Flutter: Text Style**

* Tetx(): Là một loại Widget con trong Flutter cho phép hiển thị một chuỗi Text với một dòng duy nhất trong ứng dụng. Tùy thuộc vào các ràng buộc về bố cục, có thể ngắt chuỗi trên nhiều dòng hoặc tất cả có thể được hiển thị trên một dòng. Nếu không chỉ định bất kỳ kiểu nào cho Widget Text, nó sẽ sử dụng kiểu lớp DefaultTextStyle gần nhất. Lớp này không có bất kỳ kiểu rõ ràng nào.

VD:

import 'package:flutter/material.dart';

void main() { runApp(MyApp()); }

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

theme: ThemeData(

primarySwatch: Colors.green,

),

home: MyTextPage()

);

}

}

class MyTextPage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title:Text("Text Widget Example")

),

body: Center(

child:Text("Welcome to Cafedev")

),

);

}

}

* Các thuộc tính tất yếu của Widget Text
* TextAlign: Được sử dụng để chỉ định cách Text được căn chỉnh theo chiều ngang.

Vd:

Text(

‘Xin chào’,

textAlign: TextAlign.center,// căn giữa văn bản

)

Các giá trị của TextAlign:

+ TextAlign.left - Căn tría

+ TextAlign.right – Căn phải

+ TextAlign.center - Căn giữa

+ TextAlign.justify – Căn đều 2 bên

+ TextAlign.start – căn theo hướng bắt đầu

+ TextAlign.end - căn theo hướng kết thúc

* TextDirection: Được sử dụng để xác định khi nào Text sẽ không vừa với không gian sẵn.

+ TextDirection.ltr – Hướng của văn bản sẽ từ trái sang phải

+ TextDirection.rtl – Hướng của văn bản sẽ từ phải sang trái

Vd:

Text(

‘Hello Work’,

textDirection: TextDirection.ltr,

)

* Overflow : Được sử dụng để xác định khi nào Text sẽ không vừa với không gian sẵn có. Khi hết không gian nó sẽ có “…” ở cuối đoạn Text.

Các giá trị của TextOverflow:

+ TextOverflow.clip - Cắt văn bản tại ranh giới mà không hiển thị dấu hiệu cắt.

+ TextOverflow.fade - Làm mờ dần văn bản tại ranh giới.

+ TextOverflow.ellipsis - Hiển thị dấu ... khi văn bản bị cắt.

+ TextOverflow.visible - Văn bản có thể tràn ra ngoài giới hạn của widget.

VD:

Text(

'Đây là một đoạn văn bản dài có thể bị cắt...',

overflow: TextOverflow.ellipsis, // Hiển thị dấu "..." khi bị cắt

)

* TextScaleFactor: được sử dụng để xác định tỷ lệ của Text được hiển thị bởi widget Text. Giả sử đã chỉ định hệ số tỷ lệ Text là 1,5, thì Text của chúng ta sẽ lớn hơn 50 phần trăm so với kích thước phông chữ được chỉ định.

Vd:

Text(

‘Hello Work’,

TextScaleFactor: 1.5,

)

* SoftWrap: cho phép xuống dòng nếu không đủ không gian của Text nếu nó trả về true. Còn nếu trả về false thì không cho phép.

VD:

Text(

'Đây là một đoạn văn bản rất dài và có thể cần xuống dòng nếu không đủ không gian hiển thị.',

softWrap: true, // Cho phép tự động xuống dòng

)

* MaxLines: được sử dụng để giới hạn số dòng của Text.

Vd:

Text(

‘Hello Work’,

maxlines: 3,// giới hạn số dòng tối đa của Text là 3

)

* TextWidthBasis: được sử dụng để xác định cách tính toán chiều rộng của văn bản, đặc biệt hữu ích khi làm việc với văn bản có khoảng trắng hoặc ký tự rộng.

VD:

Text(

'Hello Flutter!',

textWidthBasis: TextWidthBasis.longestLine, // Cơ sở tính toán dựa trên dòng dài nhất

)

* TextHeightBehavior: giúp kiểm soát cách chiều cao của văn bản được tính toán và áp dụng trong layout. Nó ảnh hưởng đến cách văn bản căn chỉnh theo chiều dọc trong một dòng hoặc trong nhiều dòng.
* Các thuộc tính của TextHightBehavior:

+ applyHeightToFirstAscent(bool) - Nếu true, chiều cao (height) sẽ ảnh hưởng đến phần trên của chữ (ascent).

+ applyHeightToLastDescent (bool) - Nếu true, chiều cao sẽ ảnh hưởng đến phần dưới của chữ (descent).

+ leadingDistribution (TextLeadingDistribution) - Kiểm soát cách phân bố khoảng cách giữa các dòng.

VD:

TextHeightBehavior({

bool applyHeightToFirstAscent = true,

bool applyHeightToLastDescent = true,

TextLeadingDistribution leadingDistribution = TextLeadingDistribution.proportional,

})

* Style: Đây là thuộc tính phổ biến nhất của widget cho phép tạo kiểu dáng cho Text. Nó có thể tạo kiểu dáng bằng cách chỉ định màu nền và nền trước, cỡ chứ, độ đậm của phông chữ, khoảng cách giữa các chữ và từ, ngôn ngữ, bóng.

Bảng các thuộc tính:

|  |  |
| --- | --- |
| **Thuộc tính** | **Mô tả** |
| foreground | Nó xác định màu nền trước cho Text. |
| background | Nó xác định sơn làm nền cho Text. |
| fontWeight | Nó quyết định độ dày của Text. |
| fontSize | Nó xác định kích thước của Text. |
| fontFamily | Nó được sử dụng để chỉ định kiểu chữ cho phông chữ. Đối với điều này, chúng ta cần tải xuống file kiểu chữ trong dự án của mình, sau đó giữ tệp này vào thư mục assets/font. Cuối cùng, cấu hình file pubspec.yaml để sử dụng nó trong dự án. |
| fontStyle | Nó được sử dụng để tạo kiểu cho phông chữ ở dạng in đậm hoặc nghiêng. |
| Color | Nó được sử dụng để xác định màu sắc của Text. |
| letterSpacing | Nó được sử dụng để xác định khoảng cách giữa các ký tự của Text. |
| wordSpacing | Nó được sử dụng để xác định khoảng cách giữa hai từ của Text. |
| shadows | Nó được sử dụng để vẽ bên dưới Text. |
| decoration | Chúng ta sử dụng điều này để trang trí Text bằng cách sử dụng ba tham số: decoration, decorationColor, decorationStyle. The decoration determines the location, decorationColor specify the color, decorationStyle determine xác định hình dạng. |

VD:

Text(

'Hello, Flutter!',

style: TextStyle(

fontSize: 24,

backgroundColor: Colors.white,

foregroundColor; Colors.red,

fontWeight: FontWeight.bold,

fontStyle: FontStyle.italic,

color: Colors.blue,

letterSpacing: 2.0,

wordSpacing: 4.0,

height: 1.5,

decoration: TextDecoration.underline,

decorationColor: Colors.red,

decorationStyle: TextDecorationStyle.dotted,

),

)

1. **RichText**

RichText sử dụng TextSpan để xác định các đoạn văn bản và phong cách riêng biệt cho từng đoạn.

Các thành phần chính:

* text: Nội dung văn bản
* style: Kiểu chữ (màu sắc, font, độ đậm, nghiêng, v.v.)
* children: Danh sách các TextSpan con

VD:

RichText(

text: TextSpan(

text: 'Xin chào, ',

style: TextStyle(fontSize: 24, color: Colors.black),

children: <TextSpan>[

TextSpan(

text: 'Flutter',

style: TextStyle(fontWeight: FontWeight.bold, color: Colors.blue),

),

TextSpan(

text: ' là tuyệt vời!',

style: TextStyle(fontStyle: FontStyle.italic, color: Colors.green),

),

],

),

)

* lưu ý khi dùng RichText:

+ Khi cần hiển thị nhiều phong cách khác nhau trong một đoạn văn bản.

+ Khi cần định dạng văn bản một cách linh hoạt.

+ Khi cần hiển thị văn bản với nhiều hiệu ứng như **bài viết, trang tin tức**, hoặc **mô tả sản phẩm.**

1. **Padding and Margin**

* Padding là khoảng cách bên trong giữa nội dung của widget và viền của chính nó.

Vd:

Padding(

padding: EdgeInsets.all(10.0),

child: RichText(

text: TextSpan(

text: 'Hello, ',

style: TextStyle(fontSize: 24, color: Colors.black),

children: <TextSpan>[

TextSpan(

text: 'Flutter!',

style: TextStyle(fontWeight: FontWeight.bold, color: Colors.blue),

),

],

),

),

)

Trong Flutter, Margin được thiết lập thông qua EdgeInsets trong Container. Một số cách phổ biến để khai báo Margin bao gồm:

* EdgeInsets.all(value): Thiết lập khoảng cách đều ở tất cả các cạnh.
* EdgeInsets.symmetric(horizontal: x, vertical: y): Thiết lập khoảng cách theo chiều ngang và dọc.
* EdgeInsets.only(left: x, right: y, top: z, bottom: w): Thiết lập khoảng cách riêng cho từng cạnh.

VD:

Container(

margin: EdgeInsets.all(16.0),

child: Text(

'Xin chào, Flutter!',

style: TextStyle(fontSize: 24, color: Colors.black),

),

)

Lưu ý khi dung Margin:

* Khi muốn tạo khoảng cách giữa các widget để cải thiện bố cục.
* Khi cần căn chỉnh một widget theo ý muốn trong giao diện người dùng.
* Khi cần kiểm soát không gian xung quanh một thành phần cụ thể

1. **TextButton , ElevatedButton và OutlineButton**

* TextButton: là loại nút tối giản, chỉ hiển thị văn bản mà không có viền hoặc nền.

Lưu ý khi sử dụng:

* Khi cần một nút đơn giản, không làm nổi bật so với giao diện xung quanh.
* Khi muốn tạo liên kết hoặc hành động nhẹ nhàng mà không thu hút quá nhiều sự chú ý.

Vd:

TextButton(

onPressed: () {

print('TextButton được nhấn');

},

child: Text('Nhấn vào đây'),

)

* ElevatedButton: là loại nút có nền và hiệu ứng nâng, giúp tạo sự nổi bật rõ ràng trên giao diện.

Lưu ý khi sử dụng:

* Khi cần một nút quan trọng, thu hút sự chú ý của người dùng.
* Khi muốn tạo sự nổi bật cho một hành động chính trong giao diện.

Vd:

ElevatedButton(

onPressed: () {

print('ElevatedButton được nhấn');

},

child: Text('Nhấn vào đây'),

)

* OutlineButton: là nút có viền bao quanh, giúp làm nổi bật hơn so với TextButton, nhưng vẫn giữ phong cách nhẹ nhàng.

Lưu ý khi sử dụng:

* Khi cần một nút có độ nổi bật trung bình giữa TextButton và ElevatedButton.
* Khi muốn tạo một phong cách nút với viền nhưng không làm nổi bật quá mức.

VD:

OutlinedButton(

onPressed: () {

print('OutlinedButton được nhấn');

},

child: Text('Nhấn vào đây'),

)

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **TextButton** | **OutlinedButton** | **ElevatedButton** |
| Viền | Không có | Có viền | Không có |
| Nền | Không có | Không có | Có nền |
| Hiệu ứng nâng | Không có | Không có | Có |
| Độ nổi bật | Thấp | Trung bình | Cao |