Component trong Jetpack Compose

- Text, Button, Image, Floating Action Button, Icon, Box,...
- Row/Column
- Modifier

Nội dung

- Text, Button, Image, Floating Action Button, Icon, Box,...
- Row/Column
- Modifier

Text

Text là gì?

Nếu bạn là Android developer, thì nó chính là **TextView**.

Nếu bạn là Android programming, thì nó chính là **label** hay **paragraph**.

Text

1. Size chữ

Để thay đổi size chữ ta sử dụng thuộc tính ***fontSize***

```
@Composable
fun TextWithSize(label : String, size : TextUnit) {
    Text(label, fontSize = size)
}

//TextWithSize("Big text",40.sp) -- call this method
```

Text

2. Màu chữ

Để thay đổi màu sắc của chữ ta sử dụng thuộc tính ***color***

```
@Composable
fun ColorText() {
    Text("Color text", color = Color.Blue)
}
```

Text

3. In đậm chữ

Để in đậm chữ ta sử dụng thuộc tính ***fontWeight***

```
@Composable
fun BoldText() {
    Text("Bold text", fontWeight = FontWeight.Bold)
}
```

Text

4. In nghiêng chữ

Để in nghiêng chữ ta sử dụng thuộc tính ***fontStyle***

```
@Composable
fun ItalicText() {
    Text("Italic Text", fontStyle = FontStyle.Italic)
}
```

Text

5. Giới hạn số dòng tối đa

Để giới hạn số dòng tối đã ta sử dụng thuộc tính ***maxLines***

```
@Composable
fun MaxLines() {
    Text("hello ".repeat(50), maxLines = 2)
}
```

Text

6. Hiển thị overflow text

Để hiển thị overflow khi nội dung quá dài ta sử dụng thuộc tính ***overflow***

```
@Composable
fun OverflowedText() {
    Text("Hello Compose ".repeat(50), maxLines = 2, overflow = TextOverflow.Ellipsis)
}
```

Hello Compose Hello Compose Hello Compose
Hello Compose Hello Compose Hello Compose H...

Text

7. Bật chế độ sao chép nội dung

Theo mặc định các nội dung hiển thị trên ứng dụng sẽ không thể copy, để bật tính năng cho phép copy nội dung ta sử dụng thuộc tính ***SelectionContainer***

```
@Composable
fun SelectableText() {
    SelectionContainer {
        Text("This text is selectable")
    }
}
```

Button

1. Button đơn giản

```
@Composable
fun SimpleButton() {
    Button(onClick = {
        //your onclick code here
    }) {
        Text(text = "Simple Button")
    }
}
```

Button

2. Màu sắc button

```
fun ButtonWithColor(){
    Button(onClick = {
        //your onclick code
    },
        colors = ButtonDefaults.buttonColors(containerColor = Color.DarkGray))
    {
        Text(text = "Button with gray background",color = Color.White)
    }
}
```

Button

3. Button với multiple text

```
@Composable
fun ButtonWithTwoTextView() {
    Button(onClick = {
        //your onclick code here
    }) {
        Text(text = "Click ", color = Color.Magenta)
        Text(text = "Here", color = Color.Green)
    }
}
```

Button

4. Button với icon


```
@Composable
fun ButtonWithIcon() {
    Button(onClick = {}) {
        Image(
            painterResource(id = R.drawable.ic_cart),
            contentDescription ="Cart button icon",
            modifier = Modifier.size(20.dp))

        Text(text = "Add to cart",Modifier.padding(start = 10.dp))
    }
}
```

Button

5. Button với shape

*** Rectangle Shape**

```
@Composable
fun ButtonWithRectangleShape() {
    Button(onClick = {}, shape = RectangleShape) {
        Text(text = "Rectangle shape")
    }
}
```

Button

5. Button với shape

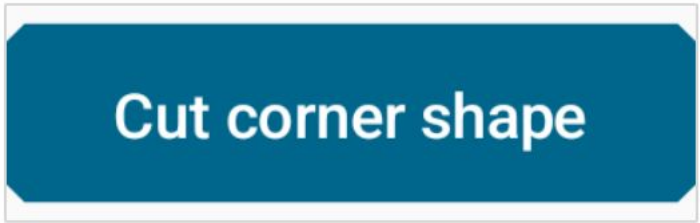
*** Round corner Shape**

```
@Composable
fun ButtonWithRoundCornerShape() {
    Button(onClick = {}, shape = RoundedCornerShape(20.dp)) {
        Text(text = "Round corner shape")
    }
}
```

Button

5. Button với shape

*** Cut corner Shape**

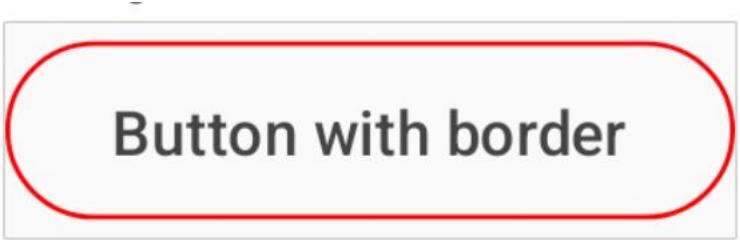



```
@Composable
fun ButtonWithCutCornerShape() {
    //CutCornerShape(percent: Int)- it will consider as percentage
    //CutCornerShape(size: Dp)- you can pass Dp also.
    //Here we use Int, so it will take percentage.
    Button(onClick = {}, shape = CutCornerShape(10)) {
        Text(text = "Cut corner shape")
    }
}
```

Button

6. Button với border

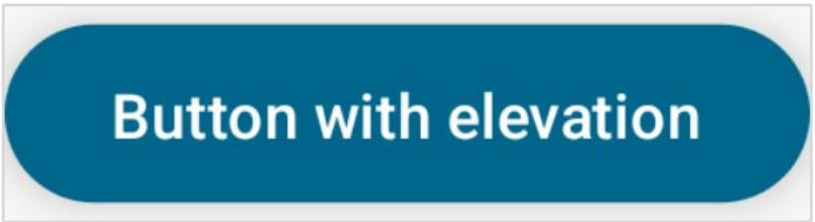
```
@Composable
fun ButtonWithBorder() {
    Button(
        onClick = {
            //your onclick code
        },
        border = BorderStroke(1.dp, Color.Red),
        colors = ButtonDefaults.outlinedButtonColors(contentColor = Color.Red)
    ) {
        Text(text = "Button with border", color = Color.DarkGray)
    }
}
```



Button

7. Button với Elevation

```
@Composable
fun ButtonWithElevation() {
    Button(
        onClick = {
            //your onclick code here
        }, elevation = ButtonDefaults.buttonElevation(
            defaultElevation = 10.dp,
            pressedElevation = 15.dp,
            disabledElevation = 0.dp
        )
    ) {
        Text(text = "Button with elevation")
    }
}
```



Image

Các thuộc tính có trong component Image

```
@Composable
fun Image(
    painter: Painter,
    contentDescription: String?,
    modifier: Modifier = Modifier,
    alignment: Alignment = Alignment.Center,
    contentScale: ContentScale = ContentScale.Fit,
    alpha: Float = DefaultAlpha,
    colorFilter: ColorFilter? = null
)
```

Image

Để tạo ra một ảnh, cần phải khai báo những thuộc tính sau:

- **Painter** – Để load drawable từ resources bạn cần sử dụng painterResource.

  fun painterResource(@DrawableRes id: Int): Painter

- **ContentDescription** – Mô tả hình ảnh. Nếu không có, có thể set giá trị null.
- **Modifier (Optional)** – Nếu bạn không sử dụng modifier, Image sẽ lấy kích thước ban đầu của file. Vì vậy, hãy sử dụng modifier để thay đổi kích thước mặc định, tránh các vấn đề liên quan đến thiết kế

Image

1. Image đơn giản

```
@Composable
fun SimpleImage() {
    Image(
        painter = painterResource(id = R.mipmap.logo),
        contentDescription = "Flower",
        modifier = Modifier.fillMaxWidth()
    )
}
```

Image

2. Image có dạng hình tròn

```
fun CircleImageView() {
    Image(
        painter = painterResource(R.mipmap.logo),
        contentDescription = "Circle Image",
        contentScale = ContentScale.Crop,
        modifier = Modifier
            .size(128.dp)
            .clip(CircleShape) // clip to the circle shape
            .border(5.dp, Color.Gray, CircleShape)//optional
    )
}
```

Image

3. Image bo góc

```
@Composable
fun RoundCornerImageView() {
    Image(
        painter = painterResource(R.mipmap.logo),
        contentDescription = "Round corner image",
        contentScale = ContentScale.Crop,
        modifier = Modifier
            .size(128.dp)
            .clip(RoundedCornerShape(10))
            .border(5.dp, Color.Gray, RoundedCornerShape(10))
    )
}
```

Image

4. Image với background

```
@Composable
fun ImageWithBackgroundColor() {
    Image(
        painter = painterResource(id = R.drawable.ic_android),
        contentDescription = "",
        modifier = Modifier
            .size( 200.dp)
            .background(Color.DarkGray)
            .padding(20.dp)
    )
}
```

Image

5. Image ColorFilter (tint) – Thay đổi màu ảnh

```
@Composable
fun ImageWithTint() {
    Image(
        painter = painterResource(id = R.drawable.ic_android),
        contentDescription = "",
        colorFilter = ColorFilter.tint(Color.Red),
        modifier = Modifier
            .size( 200.dp)
    )
}
```

Image

6. ContentScale

XML Image		Jetpack Compose Image	
ScaleType	**Note**	**ContentScale**	**Alignment**
Matrix	Default with no value	None	Top Start
Center		None	Center
Center Inside		Inside	Center
Center Crop		Crop	Center
Fit Center		Fit	Center
Fit Start		Fit	Top Start
Fit End		Fit	Bottom End
Fit XY		FillBounce	<Not Applicable>

Image

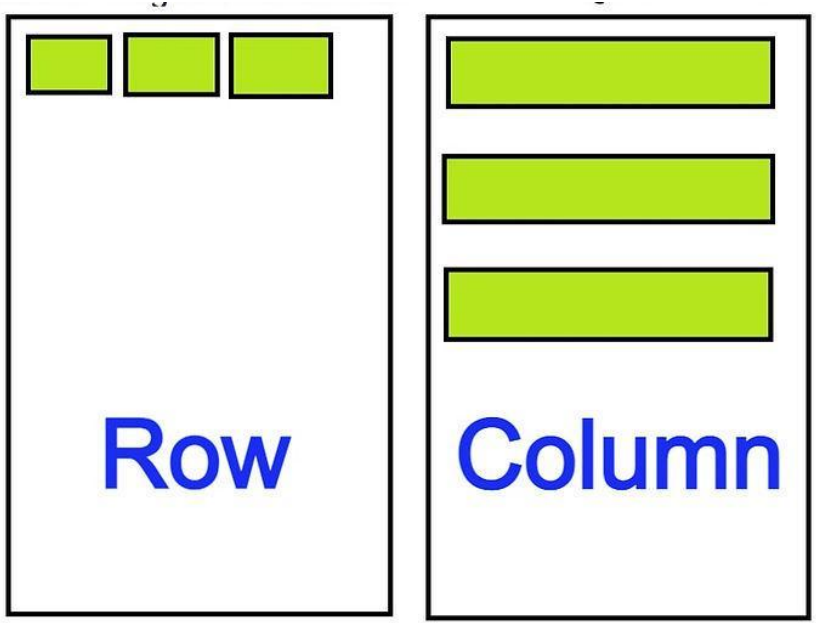
6. ContentScale

```
@Composable
fun InsideFit() {
    Image(
        painter = painterResource(id = R.mipmap.logo),
        contentDescription = "",
        modifier = Modifier
            .size(150.dp)
            .background(Color.LightGray),
        contentScale = ContentScale.Inside
    )
}
```

Row/Column

Row – Sắp xếp các component theo chiều ngang (horizontal).

Column – Sắp xếp các component theo chiều dọc (vertical).



Row/Column

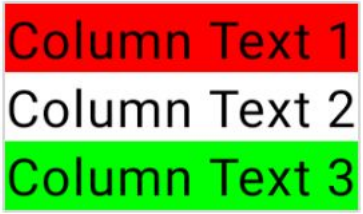
***Row**

```
@Composable
fun SimpleRow() {
    Row {
        Text(text = "Row Text 1", Modifier.background(Color.Red))
        Text(text = "Row Text 2", Modifier.background(Color.White))
        Text(text = "Row Text 3", Modifier.background(Color.Green))
    }
}
```

Row Text 1 Row Text 2 Row Text 3

Row/Column

***Column**




```
@Composable
fun SimpleColumn() {
    Column {
        Text(text = "Column Text 1", Modifier.background(Color.Red))
        Text(text = "Column Text 2", Modifier.background(Color.White))
        Text(text = "Column Text 3", Modifier.background(Color.Green))
    }
}
```

Row/Column

***Alignment**

Có 9 tùy chọn alignment có thể áp dụng cho UI elements

TopStart	**TopCenter**	**TopEnd**
CenterStart	**Center**	**CenterEnd**
BottomStart	**BottomCenter**	**BottomEnd**

Row/Column

***Arrangement**

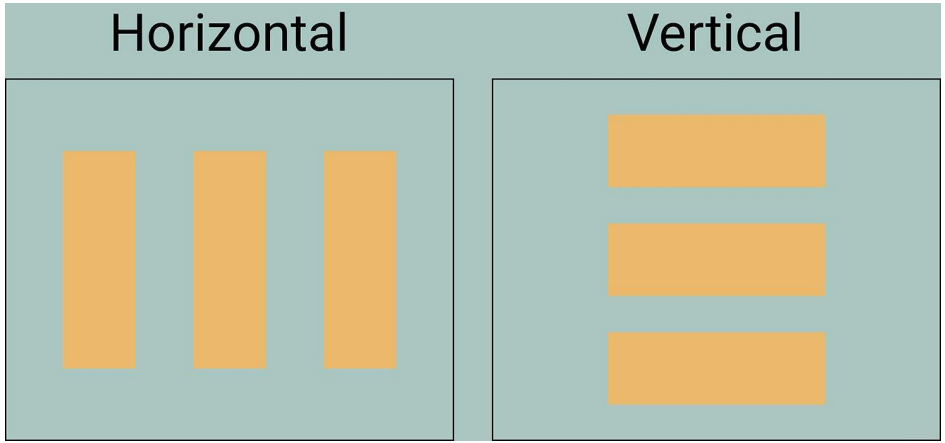
Có 3 cách sắp xếp các component có thể áp dụng:

- **SpaceEvenly**
- **SpaceBetween**
- **SpaceAround**

Row/Column

***Arrangement**

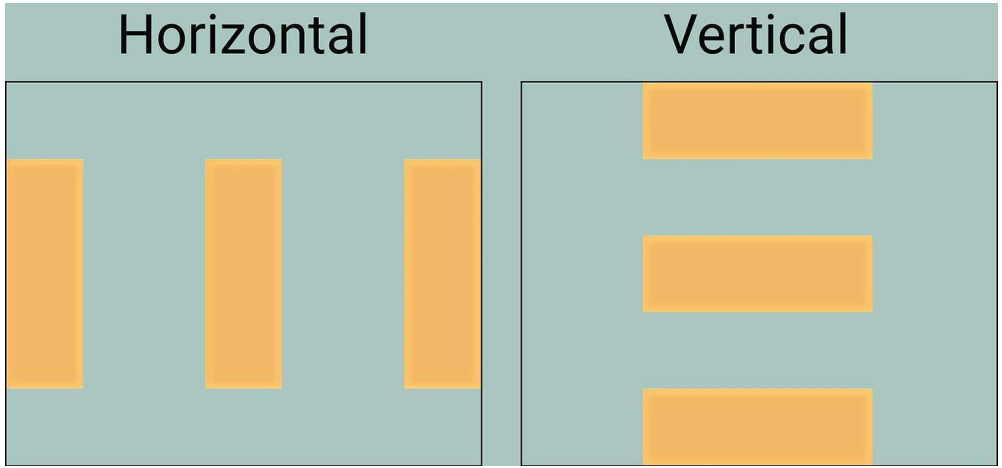
- **SpaceEvenly:** các item được phân phối sao cho khoảng cách giữa hai item bất kỳ, giữa item và các lề là bằng nhau



Row/Column

***Arrangement**

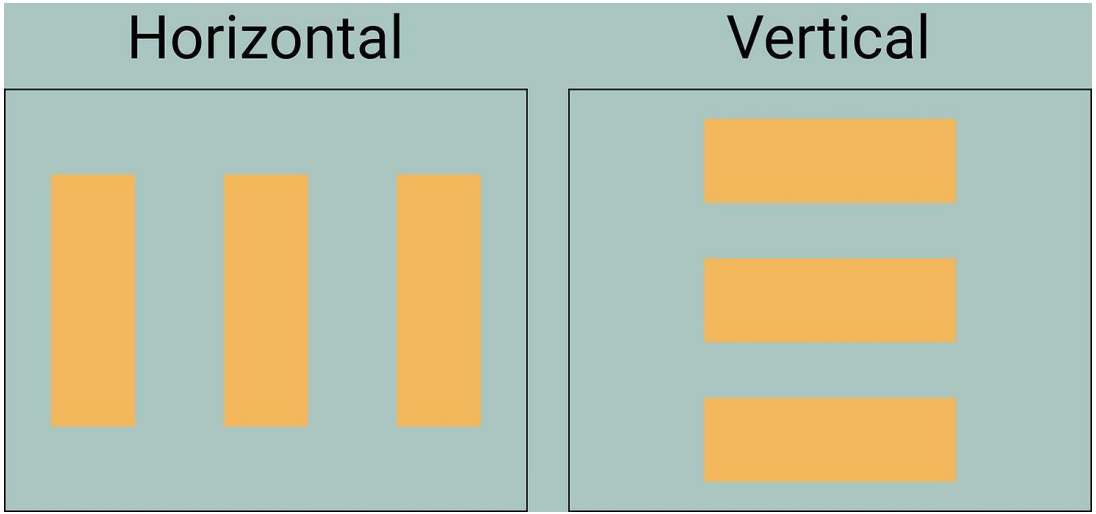
- **SpaceBetween:** các item sẽ có khoảng cách giữa các phần tử bằng nhau do container sẽ tự động căn khoảng cách, item đầu tiên sát lề chứa điểm main-start, item cuối cùng sát lề chứa điểm main-end



Row/Column

***Arrangement**

- **SpaceAround:** tương tự space-between, nhưng khác ở chỗ là mỗi item có khoảng cách 2 bên cạnh và những khoảng cách này bằng nhau



Row/Column

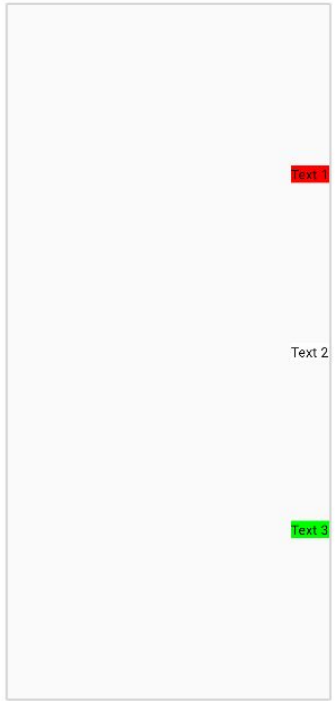
Ví dụ:

```
@Composable
fun RowArrangement(){
    Row(modifier = Modifier.fillMaxWidth(),
        verticalAlignment = Alignment.Top,
        horizontalArrangement  =  Arrangement.SpaceEvenly) {
        Text(text = " Text 1")
        Text(text = " Text 2")
        Text(text = " Text 3")
    }
}
```

Row/Column

Ví dụ:

```
@Composable
fun ColumnArrangement(){
    Column(modifier = Modifier.fillMaxHeight().fillMaxWidth(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment  =  Alignment.End
    ) {
        Text(text = "Text 1",Modifier.background(Color.Red))
        Text(text = "Text 2",Modifier.background(Color.White))
        Text(text = "Text 3",Modifier.background(Color.Green))
    }
}
```



Modifiers

Đối tượng sửa đổi (modifier) cho phép bạn trang trí hoặc tăng cường hoạt động của thành phần kết hợp. Đối tượng sửa đổi cho phép bạn thực hiện những việc sau:

- Thay đổi kích thước, bố cục, hành vi và giao diện của thành phần kết hợp
- Thêm thông tin, như nhãn hỗ trợ tiếp cận
- Xử lý dữ liệu do người dùng nhập
- Thêm các lượt tương tác cấp cao, như làm cho một thành phần có thể nhấp vào, cuộn được, có thể kéo hoặc thu phóng

Nếu bạn là Android Developer, hầu hết các thuộc tính xml (id, padding, margin, color, alpha, ratio, elevation...) được sử dụng với sự trợ giúp của modifiers.

Modifiers

***Background Color**

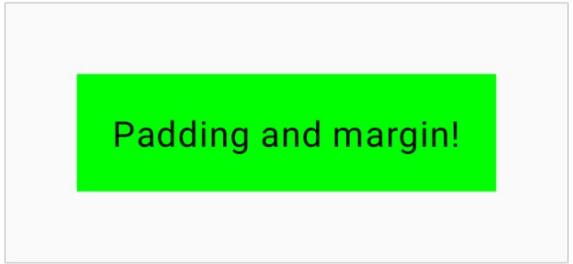
```
Text("Text with green background color",
    Modifier.background(color = Color.Green))
```

Text with green background color

Modifiers

***Padding**

Trong Jetpack Compose **KHÔNG** có modifier cho **margin**. Chúng ta sẽ sử dụng modifier của padding **cho cả padding và margin**




```
@Composable
fun TextWidthPadding() {
    Text(
        "Padding and margin!",
        Modifier.padding(32.dp) // Outer padding (margin)
            .background(color = Color.Green) //background color
            .padding(16.dp) // Inner padding
    )
}
```

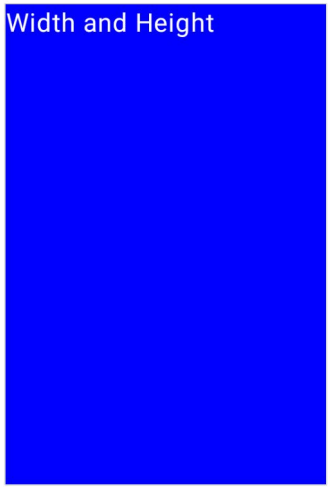
Modifiers

Width and Height

- Đối với chiều rộng: width(value : Dp)

- Đối với chiều cao: height(value: Dp)

```
@Composable
fun WidthAndHeightModifier() {
    Text(
        text = "Width and Height",
        color = Color.White,
        modifier = Modifier
            .background(Color.Blue)
            .width(200.dp)
            .height(300.dp)
    )
}
```



Modifiers

***Size**

- Nếu cần cả chiều rộng và chiều cao trong cùng một modifier, hãy sử dụng **Modifier.size().**
- Nếu cả chiều rộng và chiều cao đều giống nhau, hãy sử dụng **Modifier.size(size: Dp)**

  Ví dụ: Modifier.size(200.dp)
- Nếu muốn chiều rộng và chiều cao khác nhau, hãy sử dụng

  Modifier.size(width= x.dp, height = y.dp)

  Ví dụ: Modifier.size(width=200.dp,height=100.dp)

Modifiers

Ví dụ:

```
@Composable
fun SizeModifier() {
    Text(
        text = "Text with Size",
        color = Color.White,
        modifier = Modifier
            .background(Color.Cyan)
            .size(width = 250.dp, height = 100.dp)
    )
}
```

Modifiers

***Fill Max Width**

- Giá trị từ **0,0** đến **1,0**
- Nếu bạn muốn set chiều rộng là **match_parent**, bạn có thể sử dụng **1,0**

```
0.0 means 0%   0.1 means 10%   1.0 means 100%
```

- Trường hợp chưa có view:

  Nếu fillMaxWidth = 1,0 □ chiếm toàn bộ chiều rộng (match_parent)

  Nếu fillMaxWidth mang các giá trị còn lại □ chiếm theo tỷ lệ tương ứng với giá trị

  Ví dụ: fillMaxWidth = 0,75 □ chiếm 75% chiều rộng

Modifiers

***Fill Max Width**

- Trường hợp đã tồn tại view trước đó:

    Nếu fillMaxWidth = 1,0 □ chiếm toàn bộ khoảng không gian còn lại

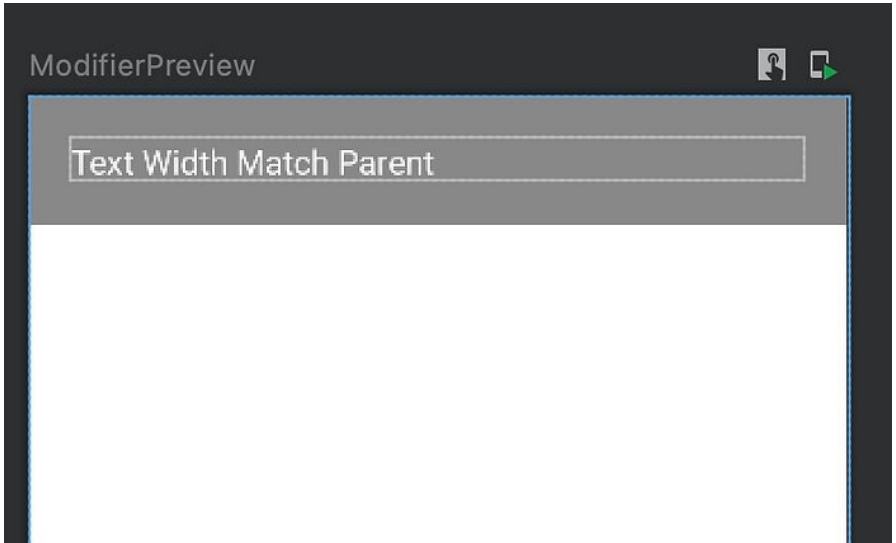
    Nếu fillMaxWidth mang các giá trị còn lại □ chiếm khoảng không gian còn lại theo tỷ lệ tương ứng với giá trị

    Ví dụ: fillMaxWidth = 0,75 □ chiếm 75% chiều rộng của khoảng không gian còn lại

Modifiers

***Fill Max Width**

```
@Composable
fun FillWidthModifier() {
    Text(
        text = "Text Width Match Parent",
        color = Color.White,
        modifier = Modifier
            .background(Color.Gray)
            .padding(10.dp)
            .fillMaxWidth(1.0f)
    )
}
```



Modifiers

***Fill Max Height**

- Giá trị từ **0,0** đến **1,0**
- Nếu bạn muốn set chiều cao là **match_parent**, bạn có thể sử dụng **1,0**

```
0.0 means 0%   0.1 means 10%   1.0 means 100%
```

- Trường hợp chưa có view:

  Nếu fillMaxHeight = 1,0 □ chiếm toàn bộ chiều cao (match_parent)

  Nếu fillMaxHeight mang các giá trị còn lại □ chiếm theo tỷ lệ tương ứng với giá trị

  Ví dụ: fillMaxHeight = 0,75 □ chiếm 75% chiều cao

Modifiers

***Fill Max Height**

- Trường hợp đã tồn tại view trước đó:

Nếu fillMaxHeight = 1,0 □ chiếm toàn bộ khoảng không gian còn lại

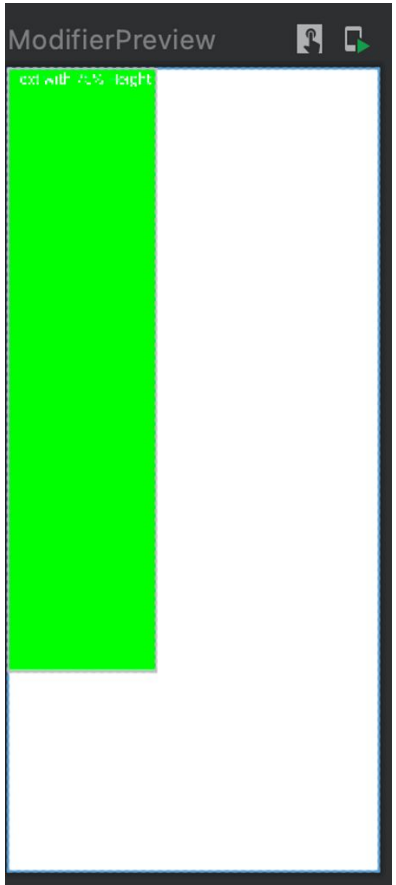
Nếu fillMaxHeight mang các giá trị còn lại □ chiếm khoảng không gian còn lại theo tỷ lệ tương ứng với giá trị

Ví dụ: fillMaxHeight = 0,75 □ chiếm 75% chiều cao của khoảng không gian còn lại

Modifiers

***Fill Max Height**

```
@Composable
fun FillHeightModifier() {
    Text(
        text = " Text with 75% Height ",
        color = Color.White,
        modifier = Modifier
            .background(Color.Green)
            .fillMaxHeight(0.75f) //75% area fill
    )
}
```



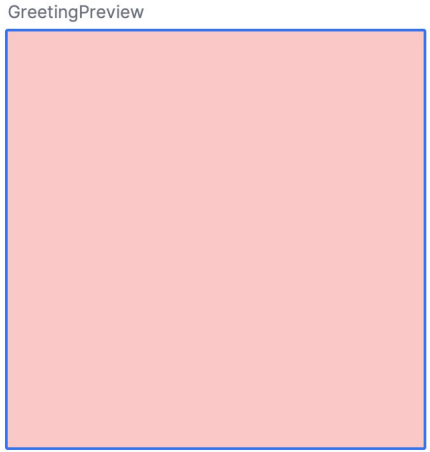
Alpha (Opacitiy)

Alpha được sử dụng để set độ mờ của một view

Giá trị của Alpha từ **0,0** đến **1,0**

> 0.0 means 0% 0.1 means 10% 1.0 means 100%

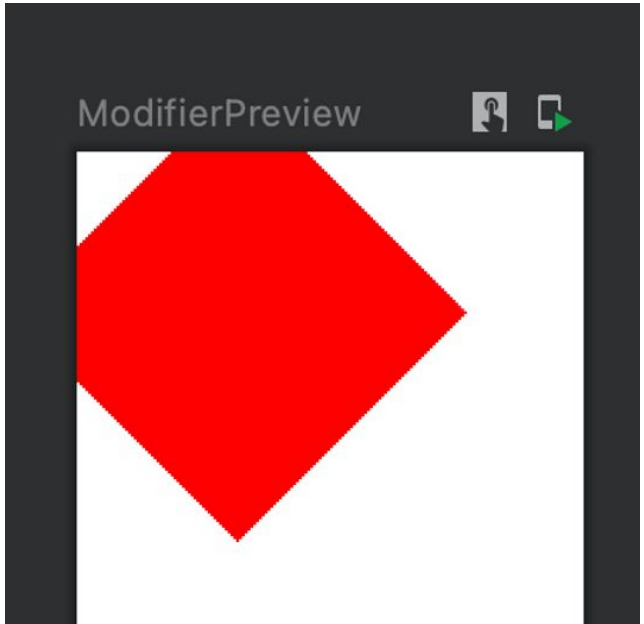
```
@Composable
fun AlphaModifier() {
    Box(
        Modifier
            .size(250.dp)
            .alpha(0.2f)//20% opacity
            .background(Color.Red)
    )
}
```



Rotate

Alpha được sử dụng để xoay một view

```
@Composable
fun RotateModifier() {
    Box(
        Modifier
            .rotate(45f)
            .size(250.dp)
            .background(Color.Red)
    )
}
```



Weight

Tương tự như layout_weight (Android), Weight dùng để chia tỷ lệ hiển thị giữa có component

```
@Composable
fun WeightModifier(){
    Row() {
        Column(
            Modifier.weight(1f).background(Color.Red)){
            Text(text = "Weight = 1", color = Color.White)
        }
        Column(
            Modifier.weight(1f).background(Color.Blue)){
            Text(text = "Weight = 1", color = Color.White)
        }
        Column(
            Modifier.weight(2f).background(Color.Green)
        ) {
            Text(text = "Weight = 2")
        }

    }
}
```

Border

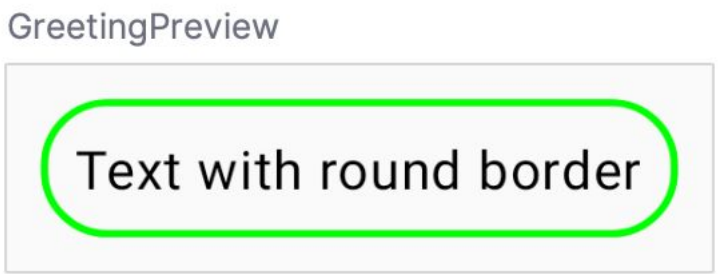
Sử dụng border để tạo viền cho component

```
@Composable
fun BorderModifier() {
    Text(
        text = "Text with Red Border",
        modifier = Modifier
            .padding(10.dp)
            .background(Color.Yellow)
            .border(2.dp,Color.Red)
            .padding(10.dp)
    )
}
```

Border

Sử dụng border để tạo viền cho component

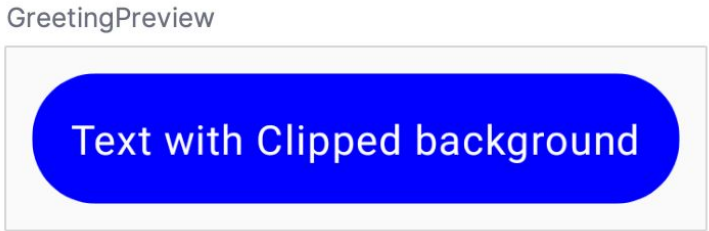
```
@Composable
fun BorderWithShape() {
    Text(
        text = "Text with round border",
        modifier = Modifier
            .padding(10.dp)
            .border(2.dp, SolidColor(Color.Green), RoundedCornerShape(20.dp))
            .padding(10.dp)
    )
}
```



Clip

Sử dụng clip để cắt view của component

```
@Composable
fun ClipModifier() {
    Text(
        text = "Text with Clipped background",
        color = Color.White,
        modifier = Modifier
            .padding(Dp(10f))
            .clip(RoundedCornerShape(25.dp))
            .background(Color.Blue)
            .padding(Dp(15f))
    )
}
```



Tài liệu tham khảo

- kotlinlang.org
- kotlinlang.org/docs
- play.kotlinlang.org/byExample

Thanks!