



# Component trong Jetpack Compose (tiếp theo)

- Card, Chip
- Switch, Snackbar
- Dialog
- Floating Action Button

# Nội dung

- Card, Chip
- Switch, Snackbar
- Dialog
- Floating Action Button

# Card

Trong Jetpack Compose, Card là một container theo thiết kế Material Design, thường dùng để hiển thị nội dung đồng nhất và có liên quan, như sản phẩm trong ứng dụng mua sắm hoặc tin tức trong ứng dụng tin tức. Card khác biệt với các container khác như Column hay Row bởi nó cung cấp một API đơn giản và chung chung hơn.

## Đặc điểm:

- Hiển thị nội dung với bố cục dạng hộp chữ nhật có viền và bóng đổ.
- Có thể chứa nhiều loại nội dung khác nhau như ảnh, văn bản, nút, v.v.
- Dễ dàng tùy chỉnh giao diện với các thuộc tính như màu sắc, độ cao, v.v.

If you are an Android developer, It's a **CardView**.

# Card

**\*Card đơn giản**

```
@Composable
fun CardMinimalExample() {
    Card() {
        Text(text = "Hello, world!")
    }
}
```

# Card

## \*Card với Elevation (shadow)

```
@Composable
fun SimpleCard(){
    val paddingModifier = Modifier.padding(10.dp)
    Card(elevation = CardDefaults.cardElevation(10.dp), modifier = paddingModifier) {
        Text(text = "Simple Card with elevation",
            modifier = paddingModifier)
    }
}
```

GreetingPreview



Simple Card with elevation

# Card

## \*Card với shape

Ta có thể set được shape của Card, nếu không set shape cho Card, mặc định sẽ nhận giá trị là **RoundedCornerShape(4.dp)**

Một số shape có thể sử dụng:

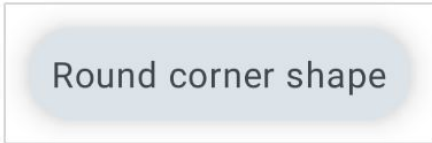
- RectangleShape
- CircleShape
- RoundedCornerShape
- CutCornerShape

# Card

## \*Card với shape

```
@Composable
fun CardWithShape() {
    val paddingModifier = Modifier.padding(10.dp)
    Card(shape = RoundedCornerShape(20.dp), elevation = CardDefaults.cardElevation(10.dp), modifier = paddingModifier) {
        Text(text = "Round corner shape", modifier = paddingModifier)
    }
}
```

GreetingPreview

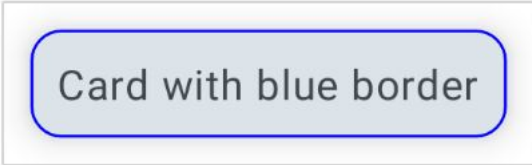


Round corner shape

# Card

## \*Card với border

GreetingPreview



Card with blue border

```
fun CardWithBorder() {  
    val paddingModifier = Modifier.padding(10.dp)  
    Card(  
        elevation = CardDefaults.cardElevation(10.dp),  
        border = BorderStroke(1.dp, Color.Blue),  
        modifier = paddingModifier  
    ) {  
        Text(text = "Card with blue border", modifier = paddingModifier)  
    }  
}
```

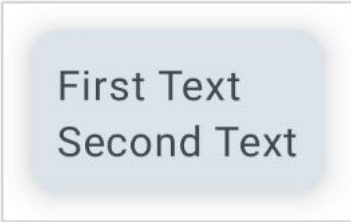


# Card

## \*Card với Multiple views

```
@Composable
fun CardWithMultipleViews() {
    val paddingModifier = Modifier.padding(10.dp)
    Card(
        elevation = CardDefaults.cardElevation(10.dp),
        modifier = paddingModifier
    ) {
        Column(modifier = paddingModifier) {
            Text(text = "First Text")
            Text(text = "Second Text")
        }
    }
}
```

GreetingPreview



First Text  
Second Text

# Chip

Trong Jetpack Compose, Chip là một thành phần giao diện người dùng nhỏ gọn và tương tác, thường được sử dụng để biểu diễn các thực thể phức tạp như liên hệ hoặc thẻ tag, thường đi kèm với biểu tượng và nhãn. Chip có thể được cấu hình để có thể kiểm tra (checkable), có thể loại bỏ (dismissible), hoặc có thể nhấp vào (clickable)



Assist chip



Filter chip



Input chip

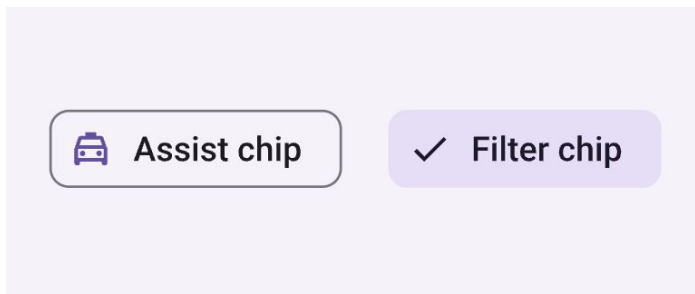


Suggestion chip

# Chip

Có bốn loại Chip chính mà bạn có thể sử dụng trong các tình huống khác nhau:

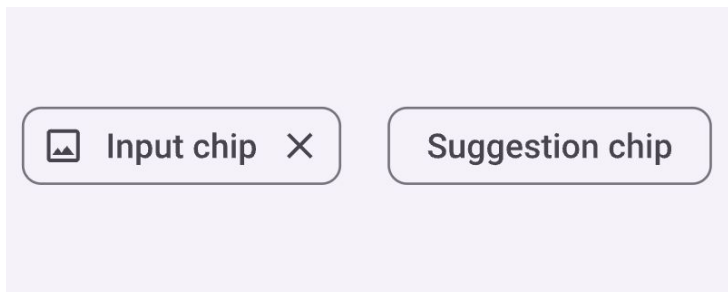
- **Assist Chip:** Hỗ trợ người dùng trong quá trình thực hiện một nhiệm vụ. Thường xuất hiện như một phần tử giao diện người dùng tạm thời phản hồi theo đầu vào của người dùng.
- **Filter Chip:** Cho phép người dùng lọc nội dung từ một tập hợp các tùy chọn. Chúng có thể được chọn hoặc bỏ chọn và có thể bao gồm biểu tượng kiểm tra khi được chọn.



# Chip

Có bốn loại Chip chính mà bạn có thể sử dụng trong các tình huống khác nhau:

- **Input Chip:** Đại diện cho thông tin do người dùng cung cấp, như lựa chọn trong một menu. Chúng có thể chứa biểu tượng và văn bản, và cung cấp một biểu tượng 'X' để loại bỏ.
- **Suggestion Chip:** Cung cấp các đề xuất cho người dùng dựa trên hoạt động gần đây hoặc đầu vào của họ. Thường xuất hiện dưới một trường nhập liệu để thúc đẩy hành động của người dùng.



# Chip

## \*Assist Chip

GreetingPreview



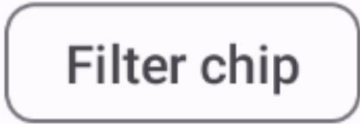
```
@Composable
fun AssistChipExample() {
    AssistChip(
        onClick = { Log.d("Assist chip", "hello world") },
        label = { Text("Assist chip") },
        leadingIcon = {
            Icon(
                Icons.Filled.Settings,
                contentDescription = "Localized
description",
                Modifier.size(AssistChipDefaults.IconSize)
            )
        }
    )
}
```

# Chip

## \*Filter Chip

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun FilterChipExample() {
    var selected by remember { mutableStateOf(false) }

    FilterChip(
        onClick = { selected = !selected },
        label = {
            Text("Filter chip")
        },
        selected = selected,
        leadingIcon = if (selected) {
            {
                Icon(
                    imageVector = Icons.Filled.Done,
                    contentDescription = "Done icon",
                    modifier = Modifier.size(FilterChipDefaults.IconSize)
                )
            }
        } else {
            null
        },
    )
}
```



Filter chip



✓ Filter chip

# Chip

## \*Input Chip

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun InputChipExample(
    text: String,
    onDismiss: () → Unit,
) {
    var enabled by remember { mutableStateOf(true) }
    if (!enabled) return

    InputChip(
        onClick = {
            onDismiss()
            enabled = !enabled
        },
        label = { Text(text) },
        selected = enabled,
        avatar = {
            Icon(
                Icons.Filled.Person,
                contentDescription = "Localized description",
                Modifier.size(InputChipDefaults.AvatarSize)
            )
        },
        trailingIcon = {
            Icon(
                Icons.Default.Close,
                contentDescription = "Localized description",
                Modifier.size(InputChipDefaults.AvatarSize)
            )
        },
    )
}
```

GreetingPreview



# Chip

## \*Suggestion Chip

GreetingPreview

Suggestion chip

```
@Composable
fun SuggestionChipExample() {
    SuggestionChip(
        onClick = { Log.d("Suggestion chip", "hello world") },
        label = { Text("Suggestion chip") }
    )
}
```



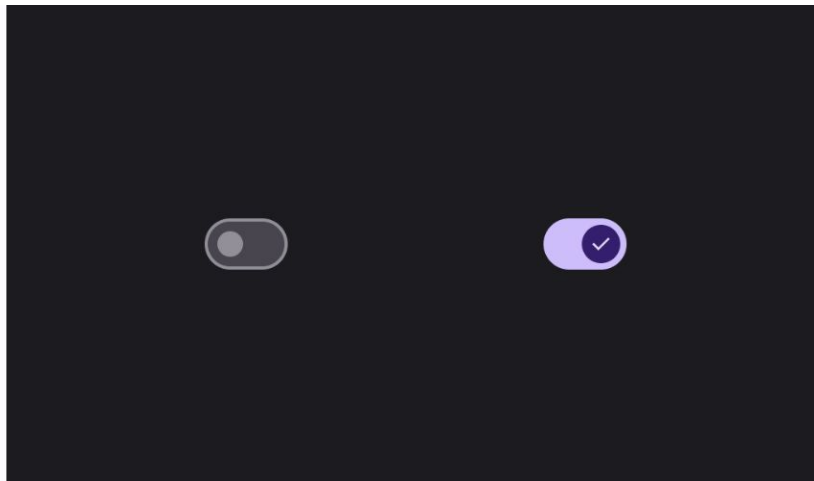
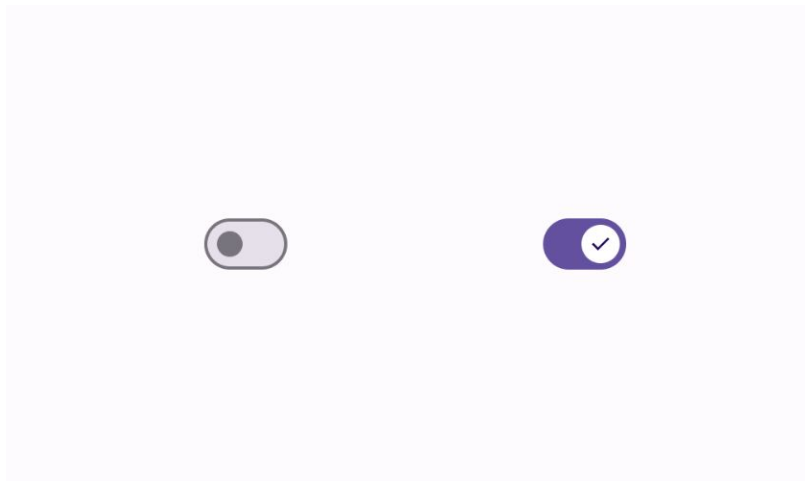
# Switch

Trong Jetpack Compose, Switch là một thành phần giao diện người dùng cho phép người dùng chuyển đổi giữa hai trạng thái: đã chọn (checked) và chưa chọn (unchecked). Bạn có thể sử dụng Switch để cho phép người dùng:

- Bật hoặc tắt một cài đặt.
- Kích hoạt hoặc vô hiệu hóa một tính năng.
- Chọn một tùy chọn.

# Switch

Switch bao gồm hai phần: phần di chuyển (thumb) và phần nền (track). Phần di chuyển là phần có thể kéo của switch, và phần nền là nền đằng sau. Người dùng có thể kéo phần di chuyển sang trái hoặc phải để thay đổi trạng thái của switch, hoặc cũng có thể chạm vào switch để chọn hoặc bỏ chọn nó



# Switch

## Ví dụ:

```
@Composable
fun SwitchMinimalExample() {
    var checked by remember { mutableStateOf(true) }

    Switch(
        checked = checked,
        onCheckedChange = {
            checked = it
        }
    )
}
```



# Switch

## \*Custom Thumb

```
@Composable
fun SwitchWithIconExample() {
    var checked by remember { mutableStateOf(true) }

    Switch(checked = checked, onCheckedChange = {
        checked = it
    }, thumbContent = if (checked) {
        {
            Icon(
                imageVector = Icons.Filled.Check,
                contentDescription = null,
                modifier = Modifier.size(SwitchDefaults.IconSize),
            )
        }
    } else {
        null
    })
}
```



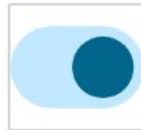
# Switch

## \*Custom Colors

```
@Composable
fun SwitchWithCustomColors() {
    var checked by remember { mutableStateOf(true) }

    Switch(
        checked = checked, onCheckedChange = {
            checked = it
        }, colors = SwitchDefaults.colors(
            checkedThumbColor = MaterialTheme.colorScheme.primary,
            checkedTrackColor = MaterialTheme.colorScheme.primaryContainer,
            uncheckedThumbColor = MaterialTheme.colorScheme.secondary,
            uncheckedTrackColor = MaterialTheme.colorScheme.secondaryContainer,
        )
    )
}
```

GreetingPrevi...



# Snackbar

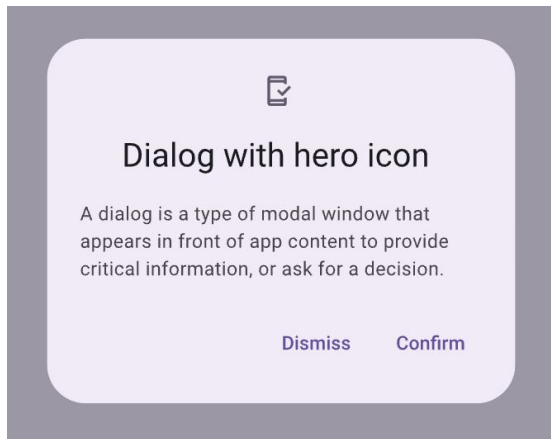
Snackbar trong Jetpack Compose là một thành phần giao diện người dùng cung cấp thông báo ngắn gọn xuất hiện ở phía dưới màn hình. Nó cung cấp phản hồi về một hoạt động hoặc hành động mà không làm gián đoạn trải nghiệm người dùng. Snackbar tự động biến mất sau vài giây, hoặc người dùng có thể loại bỏ nó bằng một hành động, chẳng hạn như nhấn vào một nút.

# Snackbar

```
val scope = rememberCoroutineScope()
val snackbarHostState = remember { SnackbarHostState() }
Scaffold(
    snackbarHost = {
        SnackbarHost(hostState = snackbarHostState)
    },
    floatingActionButton = {
        ExtendedFloatingActionButton(
            text = { Text("Show snackbar") },
            icon = { Icon(Icons.Filled.Info, contentDescription = "") },
            onClick = {
                scope.launch {
                    snackbarHostState.showSnackbar("Snackbar")
                }
            }
        )
    }
) { contentPadding →
    // Screen content
}
```

# Dialog

Dialog trong Jetpack Compose là một thành phần giao diện người dùng hiển thị thông điệp hoặc yêu cầu nhập liệu từ người dùng trên một lớp phủ lên nội dung chính của ứng dụng. Nó tạo ra một trải nghiệm giao diện người dùng gián đoạn để thu hút sự chú ý của người dùng. Dialog có thể được sử dụng để xác nhận hành động của người dùng, yêu cầu nhập liệu, hoặc trình bày một danh sách các lựa chọn cho người dùng





# Dialog

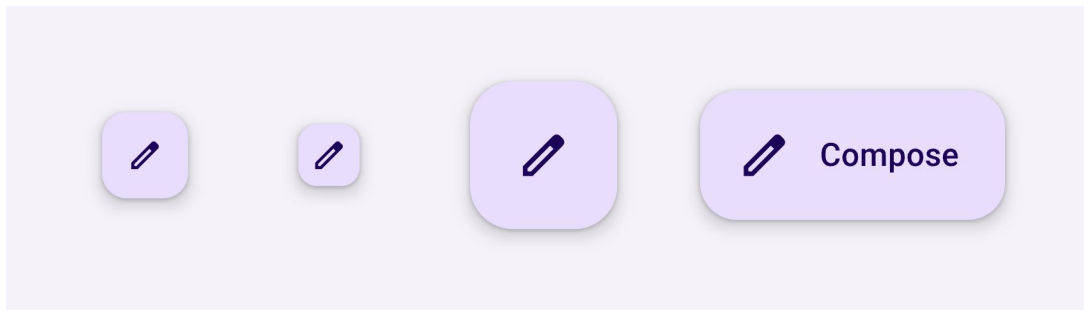
```
@Composable
fun MinimalDialog(onDismissRequest: () → Unit) {
    Dialog(onDismissRequest = { onDismissRequest() }) {
        Card(
            modifier = Modifier
                .fillMaxWidth()
                .height(200.dp)
                .padding(16.dp),
            shape = RoundedCornerShape(16.dp),
        ) {
            Text(
                text = "This is a minimal dialog",
                modifier = Modifier
                    .fillMaxSize()
                    .wrapContentSize(Alignment.Center),
                textAlign = TextAlign.Center,
            )
        }
    }
}
```

GreetingPreview



# Floating Action Button

Floating Action Button (FAB) trong Jetpack Compose là một nút hành động nổi bật, thường được sử dụng để thúc đẩy hành động chính trong ứng dụng của bạn. Nó thường được đặt ở góc dưới bên phải của màn hình và thúc đẩy một hành động đơn lẻ, tập trung, là con đường phổ biến nhất mà người dùng có thể thực hiện



# Floating Action Button

**\*FAB đơn giản**

```
@Composable
fun Example(onClick: () → Unit) {
    FloatingActionButton(
        onClick = { onClick() },
    ) {
        Icon(Icons.Filled.Add, "Floating action button.")
    }
}
```

GreetingPrevi...



# Floating Action Button

**\*FAB nhỏ**

```
@Composable
fun SmallExample(onClick: () → Unit) {
    SmallFloatingActionButton(
        onClick = { onClick() },
        containerColor = MaterialTheme.colorScheme.secondaryContainer,
        contentColor = MaterialTheme.colorScheme.secondary
    ) {
        Icon(Icons.Filled.Add, "Small floating action button.")
    }
}
```

# Floating Action Button

**\*FAB lớn**

```
@Composable
fun LargeExample(onClick: () → Unit) {
    LargeFloatingActionButton(
        onClick = { onClick() },
        shape = CircleShape,
    ) {
        Icon(Icons.Filled.Add, "Large floating action button")
    }
}
```

# Floating Action Button

## \*Extended button

@Composable

```
fun ExtendedExample(onClick: () → Unit) {  
    ExtendedFloatingActionButton(  
        onClick = { onClick() },  
        icon = { Icon(Icons.Filled.Edit, "Extended floating action button.") },  
        text = { Text(text = "Extended FAB") },  
    )  
}
```

GreetingPreview



# Thanks!

