Layout Basics in Jetpack Compose

Layout @Composable

Standard layout components

- `Row { }`: arranges items horizontally along the X-axis
- `Column { }`: arranges items vertically along the Y-axis
- `Box { }`: stack items along the Z-axis (out of the screen towards the viewer)

Compose Row/Column ⇔ CSS Flexbox

If you happen to know CSS FlexBox…

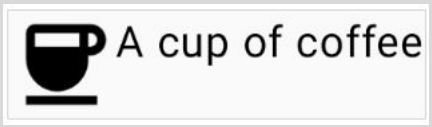
	Row	Column	CSS Flexbox
View Equivalent	`<LinearLayout orientation="horizontal">`	`<LinearLayout orientation="vertical">`	flex-direction: row flex-direction: column
Main Axis Placement	horizontalArrangement	verticalArrangement	justify
Cross Axis Placement	verticalAlignment	horizontalAlignment	align

Common Import for Compose

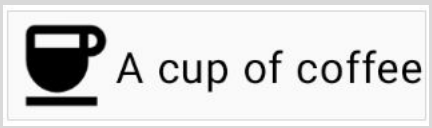
```
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout._______        // containers
import androidx.compose.material3._____                  // widgets
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
```


Row & Vertical Alignment

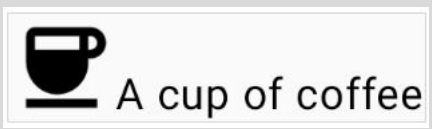
```
Row /* (verticalAlignment = Alignment.Top) */ {
   Image(painterResource(id = R.drawable.coffee),
       contentDescription = null, modifier = Modifier.size(40.dp)
   Text("A cup of coffee")
}
```




```
Row(verticalAlignment = Alignment.CenterVertically) {
   Image(painterResource(id = R.drawable.coffee),
       contentDescription = null, modifier = Modifier.size(40.dp)
   Text("A cup of coffee")
}
```




```
Row(verticalAlignment = Alignment.Bottom) {
   Image(painterResource(id = R.drawable.coffee),
       contentDescription = null, modifier = Modifier.size(40.dp)
   Text("A cup of coffee")
}
```



Row & Horizontal Arrangement

```
Row (horizontalArrangement = Arrangement.Start) {
   Image()
   Text()
}
```


```
Row (horizontalArrangement = Arrangement.Center) {
   Image()
   Text()
}
```

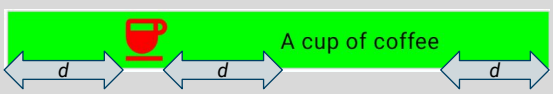

```
Row (horizontalArrangement = Arrangement.End) {
   Image()
   Text()
}
```

Row & Horizontal Arrangement

```
Row (horizontalArrangement = Arrangement.SpaceBetween) {
   Image()
   Text()
}
```

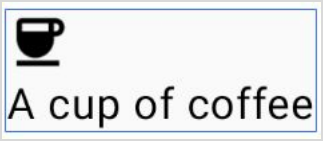

```
Row (horizontalArrangement = Arrangement.SpaceAround) {
   Image()
   Text()
}
```


```
Row (horizontalArrangement = Arrangement.SpaceEvenly) {
   Image()
   Text()
}
```

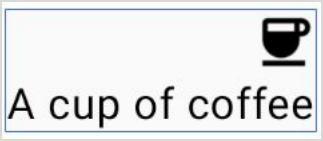


Column & Horizontal Alignment

```
Column /* (horizontalAlignment = Alignment.Start) */ {
   Image(painterResource(id = R.drawable.coffee),
       contentDescription = null)
   Text("A cup of coffee")
}
```




```
Column(horizontalAlignment = Alignment.End) {
   Image(painterResource(id = R.drawable.coffee),
       contentDescription = null)
   Text("A cup of coffee")
}
```




```
Column(horizontalAlignment = Alignment.CenterHorizontally) {
   Image(painterResource(id = R.drawable.coffee),
       contentDescription = null)
   Text("A cup of coffee")
}
```

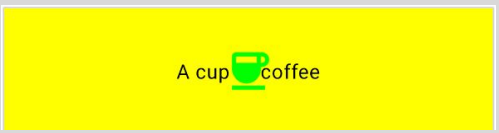
Box & Content Alignment

```
Box (contentAlignment = Alignment.TopStart) {
   Image()
   Text()
}
```


```
Box (contentAlignment = Alignment.BottomEnd) {
   Image()
   Text()
}
```


```
Box (contentAlignment = Alignment.Center) {
   Image()  // Image behind the text
   Text()
}
```


```
Box (contentAlignment = Alignment.Center) {
   Text()  // Text behind the image
   Image()
}
```



Best Practices for Designing Composables

- All composables should take an optional `modifier` parameter and pass it to the first child that emits UI
- Modifier should be the first optional parameter in the param list (after all the required parameters)

(Widget) Modifiers

Modifiers

Category	Examples
Actions	`clickable(), draggable(), selectable(),`
Drawing	`background(), clip(), shadow(),`
Border	`border(width =___, color = ___)`
Animation	`animateItemPlacement(), animateEnterExit()`
Padding	`padding(all = ___), safeConntentPadding ()`
Position	`offset()`
Size	`fillMaxWidth(), wrapContentWidth()`
Many more	

Order of Modifiers Matters!

```
Row (modifiers = Modifier
        .padding(16.dp)
        .background(Color.Green)
{
    Image()
    Text()
}
```


```
Row (modifiers = Modifier
        .background(Color.Green)
        .padding(16.dp)
{
    Image()
    Text()
}
```

Modifiers vs. Parameters

- Parameters are customization properties *specific* to the component
  - `horizontalArrangement` and `contentAlignment` in the snippet below
- Modifiers are *general* customization properties applicable to most widgets
  - `background` color in the snippet below

```
Row (horizontalArrangement = Arrangement.SpaceAround,
     modifier = Modidifier.background(Color.Yellow) {
   Image()
   Text()
}
Box (contentAlignment = Alignment.Center,
     modifier = Modidifier.background(Color.Yellow) {
   Image()
   Text()
}
```


Summary

- Standard Layout Containers
  - Row: arrange widgets along the X-axis
  - Column: arrange widgets along the Y-axis
  - Box: stack widgets along the Z-axis
- Widget Placement
  - Arrangement: placement along main axis
  - Alignment: placement along cross-axis
- Layout parameters
- Widget Modifiers