# Text

A Simple Text

| *@Composable*  *fun HelloWorldText() {*  *Text(*  *text = "Hello World",*  *)*  *}* |
| --- |

Result



This is how to set font size .

| *@Composable*  *fun HelloWorldText() {*  *Text(*  *text = "Hello World",*  *fontSize = 100.sp*  *)*  *}* |
| --- |

This is how to set line height between texts line

| *@Composable*  *fun HelloWorldText() {*  *Text(*  *text = "Hello World",*  *fontSize = 24.sp,*  *lineHeight = 200.sp*  *)*  *}* |
| --- |

This is how to set text Alignment

| *@Composable*  *fun HelloWorldText() {*  *Text(*  *text = "Hello World",*  *fontSize = 24.sp,*  *lineHeight = 200.sp,*  *textAlign = TextAlign.Center*  *)*  *}* |
| --- |

How to read text from resource

| *GreetingImage(*  *message = getString(R.string.happy\_birthday\_text),*  *from = "From Emma",*  *modifier = Modifier.padding(8.dp)*  *)* |
| --- |

| **Note:** Some Android Studio versions replace the hardcoded string with the getString() function. Please manually change the function to stringResource() in such cases.  If needed, add import androidx.compose.ui.res.stringResource to the imports section. |
| --- |

# 

# 

# Row

Simple Example

| *@Composable*  *fun RowExample() {*  *//without trailing lambda*  *Row(*  *content =* ***{***  *Text("Some text ")*  *Text("Some more text ")*  *Text("Last text ")*  ***}***  *)*  *}* |
| --- |



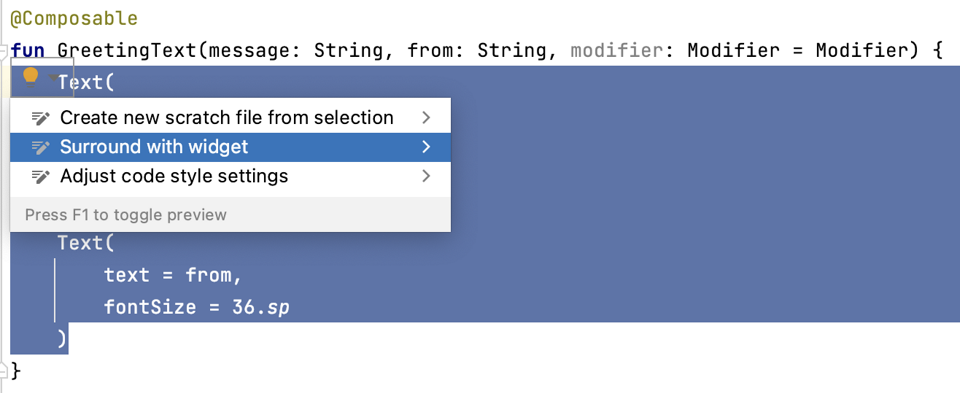
# 

Simple Example

| *@Composable*  *fun RowExample() {*  *//with trailing lambda*  *Row* ***{***  *Text("Some text ")*  *Text("Some more text ")*  *Text("Last text ")*  ***}***  *}* |
| --- |



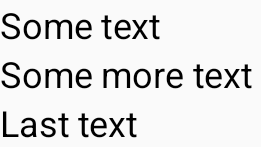
Shortcut to add Composable Function into Row



# Column

Simple Example

| *@Composable*  *fun ColumnExample() {*  *//without trailing lambda*  *Column(*  *content =* ***{***  *Text("Some text ")*  *Text("Some more text ")*  *Text("Last text ")*  ***}***  *)*    *}* |
| --- |



Simple Example

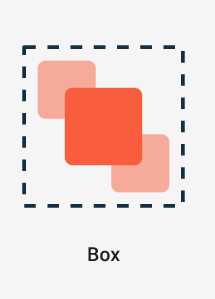
| *@Composable*  *fun ColumnExample() {*  *//with trailing lambda*  *Column* ***{***  *Text("Some text ")*  *Text("Some more text ")*  *Text("Last text ")*  ***}***  *}* |
| --- |

# 

# 

# Box

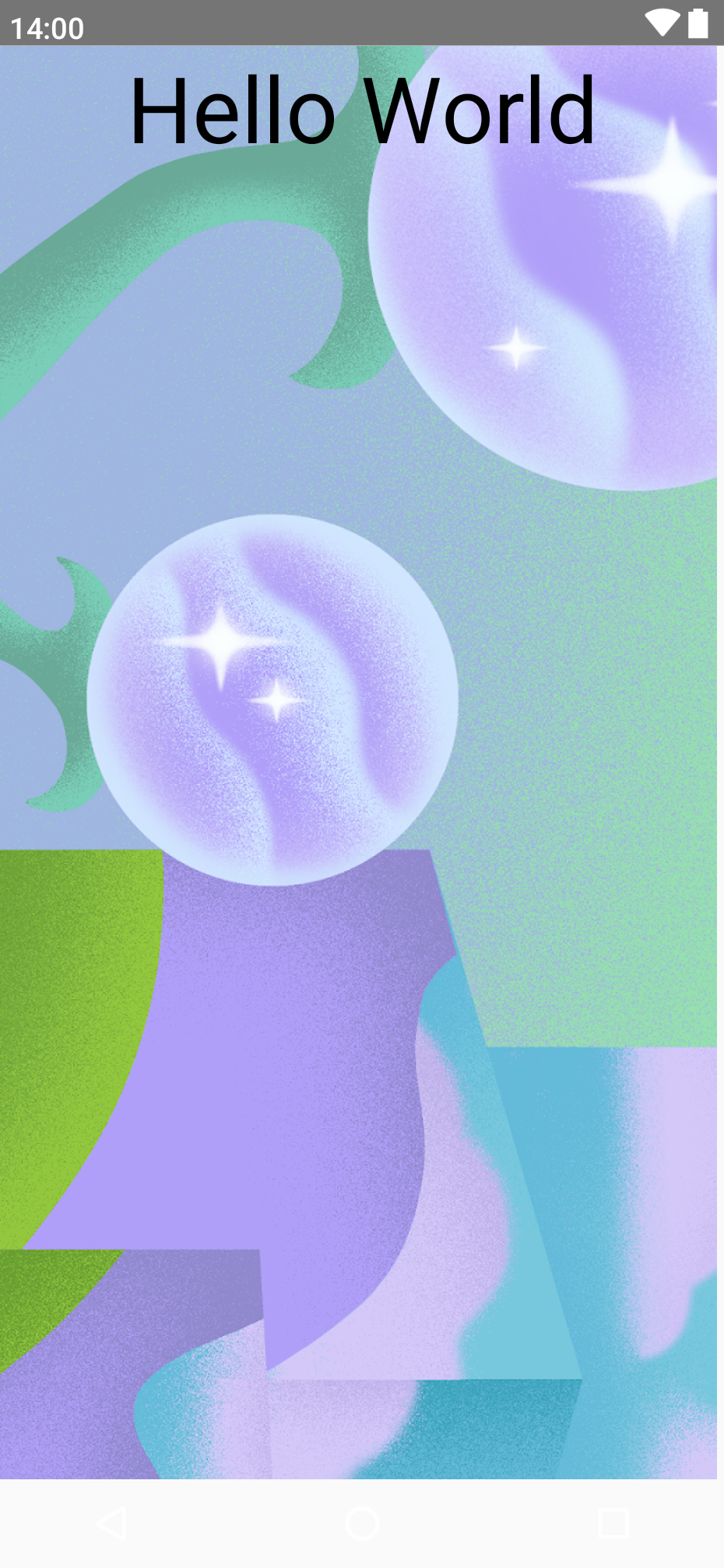
Box layout is one of the standard layout elements in Compose. Use Box layout to stack elements on top of one another. Box layout also lets you configure the specific alignment of the elements that it contains.



| *@Composable*  *fun BoxExample() {*  *Box* ***{***  *//Composable function 1*  *//Composable function 2*  ***}***  *}* |
| --- |

Example

| *@Composable*  *fun GreetingImage() {*  *val image = painterResource(R.drawable.androidparty)*  *Box* ***{***  *Image(*  *painter = image,*  *contentDescription = null*  *)*  *Text(*  *text = "Hello World",*  *fontSize = 50.sp,*  *modifier = Modifier.fillMaxSize(),*  *textAlign = TextAlign.Center*  *)*  ***}***  *}* |
| --- |



# Surface

Surface is a basic building block for displaying content and can be used to wrap other composable to provide a background color, elevation, padding, and other layout properties.

A simple surface without any attribute

| *Surface( )* ***{***    ***}*** |
| --- |

A surface which cover full screen

| *Surface(modifier = Modifier.fillMaxSize())* ***{***    ***}*** |
| --- |

# Image

A simple Image class

| *@Composable*  *fun GreetingImage() {*  *Image(painter = painterResource(id = R.drawable.androidparty), contentDescription = null)*  *}* |
| --- |

| **Note:** The painterResource() function loads a drawable image resource,and takes resource ID (R.drawable.androidparty in this case) as an argument. |
| --- |

Scale content

| Image(  painter = image,  contentDescription = null,  contentScale = ContentScale.Crop  ) |
| --- |

Change opacity

| *Image(*  *painter = image,*  *contentDescription = null,*  *contentScale = ContentScale.Crop,*  *alpha = 0.5F*  *)* |
| --- |

# Modifiers

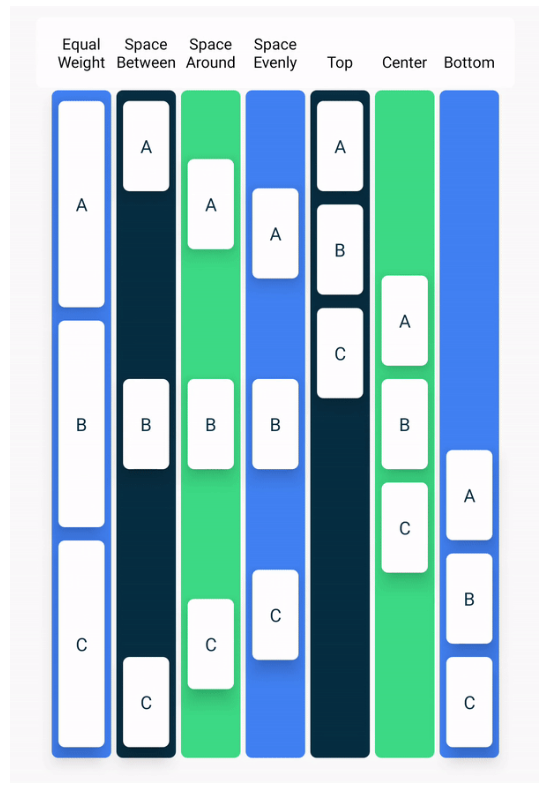
Modifiers are used to decorate or add behavior to Jetpack Compose UI elements. For example, you can add backgrounds, padding or behavior to rows, text, or buttons. To set them, a composable or a layout needs to accept a modifier as a parameter.

| *// Example*  *Text(*  *text = "Hello, World!",*  *// Solid element background color*  *modifier = Modifier.background(color = Color.Green)*  *)* |
| --- |

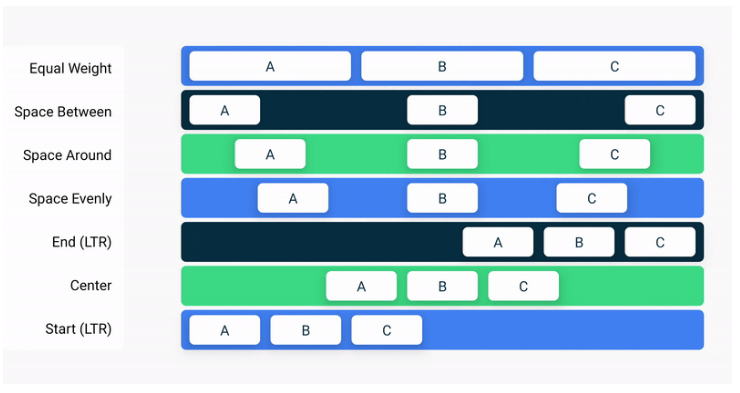
Similar to the above example, you can add Modifiers to layouts to position the child elements using arrangement and alignment properties.

To set children's position within a Row, set the ***horizontalArrangement*** and ***verticalAlignment arguments***. For a Column, set the ***verticalArrangement*** and ***horizontalAlignment*** arguments.

For example: when the size of the ***Column*** is larger than the sum of its children sizes, a ***verticalArrangement*** can be specified to define the positioning of the children inside the Column. Below is an illustration of different vertical arrangements:



Similarly, when the size of the Row is larger than the sum of its children sizes, a ***horizontalArrangement*** can be specified to define the positioning of the children inside the Row. Below is an illustration of different horizontal arrangements:



Padding

| // This is an example.  *Modifier.padding(*  *start = 16.dp,*  *top = 16.dp,*  *end = 16.dp,*  *bottom = 16.dp*  *)* |
| --- |

How to apply

| *modifier = Modifier*  *.fillMaxSize()*  *.padding(8.dp)* |
| --- |