# 1. 컴포넌트

목차



## Text

```
@Composable
fun TextExample(name: String, modifier: Modifier = Modifier) {
    Text(
        color = Color.Red, // Color(0xFFFF9944),
        text = "Hello $name!",
        modifier = modifier,
        fontSize = 30.sp,
        fontWeight = FontWeight.Bold,
        fontFamily = FontFamily.SansSerif,
        letterSpacing = 2.sp, //
        maxLines = 2,
        textDecoration = TextDecoration.Underline,
        textAlign = TextAlign.Center
    )
}
```


## Button

```
@Composable
fun ButtonExample(onButtonClicked: () -> Unit) {
    Button(
        onClick = onButtonClicked,
                  modifier = Modifier.wrapContentSize(),
        enabled = true,
        border = BorderStroke(10.dp, Color.Magenta),
        shape = CircleShape,
        contentPadding = PaddingValues(20.dp)
    ) {
        Icon(
            imageVector = Icons.Filled.Star,
            contentDescription = "Send"
        )
        Spacer(modifier = Modifier.size(ButtonDefaults.IconSpacing))
        Text(text = "Send")
    }
}
```


## Modifier

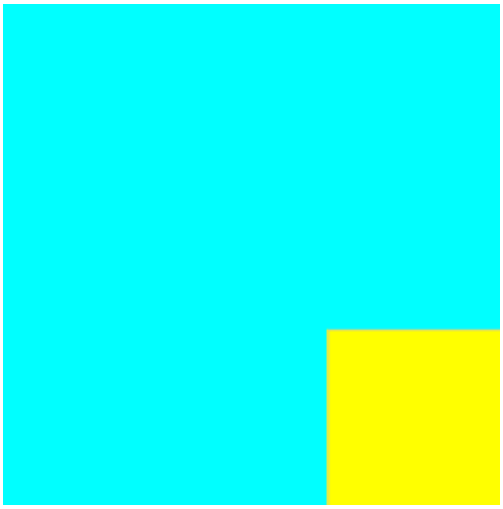
```
@Composable
fun ButtonExample(onButtonClicked: () -> Unit) {
        // fillMaxSize()
    // height(dp)
    // width(dp)
    // size(dp, dp)
    // background(color)
    // colors = ButtonDefaults.buttonColors(containerColor, contentColor)
    // padding(dp)
    // enabled, clickable
    // offset(dp, dp)
    Button(
        onClick = {},
        modifier = Modifier
            .size(200.dp)
            .padding(10.dp),
        colors = ButtonDefaults.buttonColors(
            containerColor = Color.Magenta,
            contentColor = Color.Cyan
        )
    ) {
        Icon(
            imageVector = Icons.Filled.Search,
            contentDescription = null,
            modifier = Modifier.background(Color.Blue)
        )
        Spacer(
            modifier = Modifier
                .size(ButtonDefaults.IconSpacing)
                .background(Color.Blue)
        )
        Text(
            "Search",
            modifier = Modifier
                .offset(x = 10.dp, y = (-10).dp)
                .background(Color.Blue)
        )
    }
}
```

## Surface

```
@Composable
fun SurfaceExample() {
    // Surface  View    .
    Surface(
        modifier = Modifier.padding(5.dp),
        shadowElevation = 10.dp,
        border = BorderStroke(
            width = 2.dp,
            color = Color.Magenta
        ),
        shape = CircleShape,
        color = MaterialTheme.colorScheme.error
    ) {
        Text(
            text = "Hello World!",
            modifier = Modifier.padding(8.dp)
        )
    }
}
```

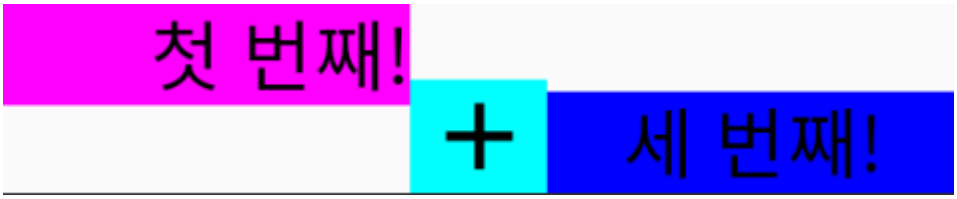
## Box

```
@Composable
fun BoxExample() {
    // Box   FrameLayout     .
    // Modifier.align(Alignment)
    // matchParentSize()
    // BoxScope    align, matchParentSize
    // fillMaxSize()
    Box(modifier = Modifier.size(200.dp)) {
        Box(
            modifier = Modifier
                .matchParentSize()
                .background(Color.Cyan)
                .align(Alignment.TopStart)
        )
        Box(
            modifier = Modifier
                .size(70.dp)
                .background(Color.Yellow)
                .align(Alignment.BottomEnd)
        )
    }
}
```



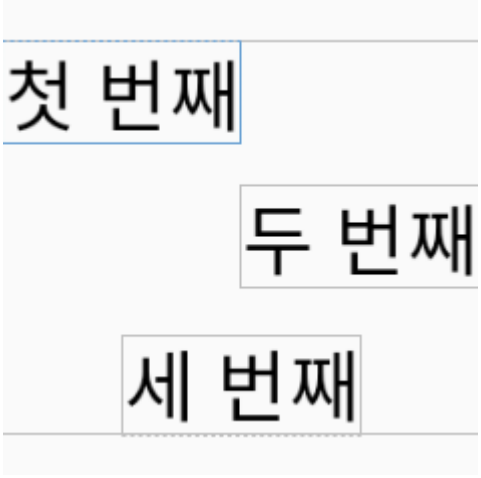
## Row

```
@Composable
fun RowExample() {
    // Row    alignment   .
    // Modifier.align(Alignment)
    // verticalAlignment(Alignment)
    // horizontalArrangement
    // weight
    // textAlign(TextAlign)
    Row(
        modifier = Modifier.size(200.dp, 40.dp),
        verticalAlignment = Alignment.Bottom
    ) {
        Text(
            text = " !",
            textAlign = TextAlign.End,
            modifier = Modifier
                .align(Alignment.Top)
                .weight(3f)
                .background(Color.Magenta)
        )
        Icon(
            imageVector = Icons.Filled.Add,
            contentDescription = "",
            modifier = Modifier
                .weight(1f)
                .background(Color.Cyan)
        )
        Text(
            text = " !",
            textAlign = TextAlign.Center,
            modifier = Modifier
                .weight(3f)
                .background(Color.Blue)
        )
    }
}
```




## Column

```
@Composable
fun ColumnExample() {
    Column(
        modifier = Modifier.size(100.dp),
        horizontalAlignment = Alignment.End,
        verticalArrangement = Arrangement.SpaceEvenly
    ) {
        Text(text = " ", modifier = Modifier.align(Alignment.Start))
        Text(text = " ")
        Text(text = " ", modifier = Modifier.align(Alignment.CenterHorizontally))
    }
}
```




## BoxWithConstraints

```
@Composable
fun Outer() {
    //  size       size .
    Column(Modifier.width(150.dp)) {
        Inner(
            Modifier
                .width(200.dp)
                .height(160.dp)
        )
        Inner(
            Modifier
                .widthIn(min = 200.dp)
                .heightIn(min = 50.dp, max = 100.dp)
        )
    }
}

@Composable
fun Inner(modifier: Modifier = Modifier) {
    BoxWithConstraints(modifier) {
        if (maxHeight > 150.dp) {
            Text(
                text = "  !",
                modifier = Modifier.align(Alignment.BottomCenter)
            )
        }
        Text(text = "maxW:$maxWidth maxH:$maxHeight minW:$minWidth minH:$minHeight")
    }
}
```

maxW:150.18182.dp maxH:160.0.dp minW:150.18182.dp minH:160.0.dp

여기 꽤 길군요!

maxW:150.18182.dp maxH:100.0.dp minW:150.18182.dp

## Image

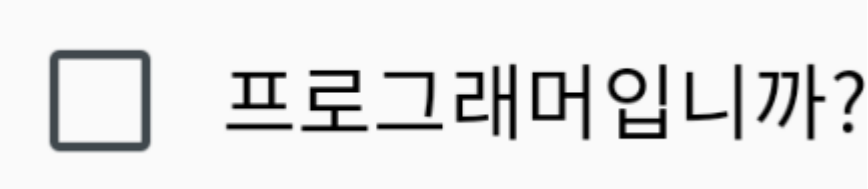
```
@Composable
fun ImageExample() {
    Column {
        Image(
            painter = painterResource(id = R.drawable.wall),
            contentDescription = " "
        )
        Image(
            imageVector = Icons.Filled.Settings,
            contentDescription = ""
        )
    }
}
```

## Coil 라이브러리

```
implementation 'io.coil-kt:coil:2.4.0'
implementation 'io.coil-kt:coil-compose:2.4.0'

---

@Composable
fun CoilExample() {
    Column {
        val imageUrl = "https://picsum.photos/1920/1080"
        // rememberImagePainter [Deprecated]
        val painter = rememberImagePainter(data = imageUrl)
        Image(
            painter = painter,
            contentDescription = ""
        )
        // Coil Image [Recommended]
        AsyncImage(
            model = imageUrl,
            contentDescription = ""
        )
    }
}
```

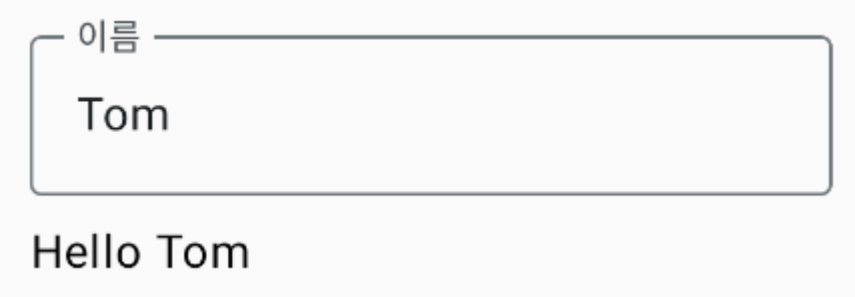
## CheckBox

```
@Composable
fun CheckBoxExample() {
    //    Recomposition(redraw)  .
    Row(verticalAlignment = Alignment.CenterVertically) {
        val (getter, setter) = remember { mutableStateOf(false) }
        Checkbox(
            checked = getter,
            onCheckedChange = setter
        )

        Text(
            text = "?",
            modifier = Modifier.clickable {
                setter(getter.not())
            }
        )
    }
}
```



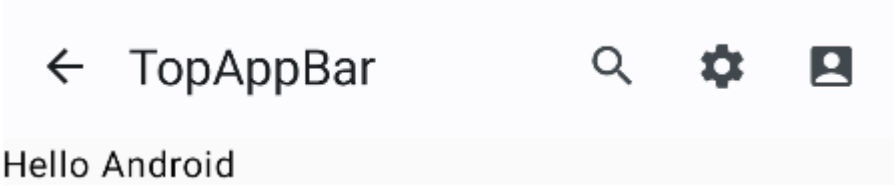

## TextField

```
@Composable
fun TextFieldExample() {
    var name by remember { mutableStateOf("Tom") }
    Column(modifier = Modifier.padding(16.dp)) {
        OutlinedTextField(
            value = name,
            onValueChange = { name = it },
            label = {
                Text("")
            }
        )
        Spacer(modifier = Modifier.size(8.dp))
        Text(text = "Hello $name")
    }
}
```




## TopAppBar

```
@Composable
fun TopAppBarExample() {
    Column {
        TopAppBar(
            title = { Text("TopAppBar") },
            navigationIcon = {
                IconButton(onClick = {}) {
                    Icon(
                        imageVector = Icons.Filled.ArrowBack,
                        contentDescription = " "
                    )
                }
            },
            actions = {
                IconButton(onClick = {}) {
                    Icon(
                        imageVector = Icons.Filled.Search,
                        contentDescription = ""
                    )
                }
                IconButton(onClick = {}) {
                    Icon(
                        imageVector = Icons.Filled.Settings,
                        contentDescription = ""
                    )
                }
                IconButton(onClick = {}) {
                    Icon(
                        imageVector = Icons.Filled.AccountBox,
                        contentDescription = ""
                    )
                }
            }
        )
        Text(text = "Hello Android")
    }
}
```




## Slot API

```
@Composable
fun CheckBoxWithSlot(
    checked: Boolean,
    onCheckedChanged: () -> Unit,
    content: @Composable RowScope.() -> Unit
) {
    //
    Row(
        verticalAlignment = Alignment.CenterVertically,
        modifier = Modifier.clickable {
            onCheckedChanged()
        }
    ) {
        Checkbox(
            checked = checked,
            onCheckedChange = { onCheckedChanged() }
        )
        content()
    }
}

@Composable
fun SlotExample() {
    var checked1 by remember { mutableStateOf(false) }
    var checked2 by remember { mutableStateOf(false) }

    Column {
        CheckBoxWithSlot(
            checked = checked1,
            onCheckedChanged = {
                checked1 = checked1.not()
            }
        ) {
            Text(text = " 1")
        }
        CheckBoxWithSlot(
            checked = checked2,
            onCheckedChanged = {
                checked2 = checked2.not()
            }
        ) {
            Text(text = " 2")
        }
    }
}
```

## Scaffold

```
@Composable
fun ScaffoldExample() {
    var checked by remember {
        mutableStateOf(false)
    }

    Scaffold(
        topBar = {
            TopAppBar(
                navigationIcon = {
                    IconButton(onClick = {}) {
                        Image(
                            imageVector = Icons.Filled.ArrowBack,
                            contentDescription = ""
                        )
                    }
                },
                title = {
                    Text(text = "Scaffold App")
                }
            )
        }
    ) { paddingValues ->
        Surface(modifier = Modifier.padding(paddingValues)) {
            CheckBoxWithSlot(
                checked = checked,
                onCheckedChanged = { checked = checked.not() },
            ) {
                Text(text = "Hello World!")
            }
        }
    }
}
```