

ТИНЬКОФФ

Как мы создавали дизайн- систему на Jetpack Compose

Стручков Михаил

09.11.2023

Кому будет полезно?

01

Есть дизайн-система на Compose, но есть сложности с развитием

02

У вас ещё нет дизайн-системы на Jetpack Compose

03

Интересны тонкости реализации дизайна на Compose

Стручков Михаил

Android-разработчик
в команде Core UI



@ m1hastr@yandex.ru



Тинькофф онлайн
банк
4,3 ★



Тинькофф Мобайл:
сотовая связь
4,4 ★



Тинькофф
Инвестиции - ...
4,4 ★



Тинькофф Джуниор



Тинькофф Бизнес
для ИП и ООО
5,0 ★



Кредит онлайн:
кредитная карта



Академия
инвестиций...
4,9 ★



Тинькофф Журнал



Тинькофф Подпись



Тинькофф
Бухгалтерия для ИП
5,0 ★

Сегодня



Tinkoff UI

Сегодня



Tinkoff UI



As-is: Дизайн-система
на Jetpack Compose V1

Сегодня



Tinkoff UI



As-is: Дизайн-система
на Jetpack Compose V1



To-be: Дизайн-система V2

Сегодня



Tinkoff UI



As-is: Дизайн-система
на Jetpack Compose V1



To-be: Дизайн-система V2



Нестандартные UI-кейсы

Сегодня



Tinkoff UI



As-is: Дизайн-система
на Jetpack Compose V1



To-be: Дизайн-система V2



Нестандартные UI-кейсы



Дизайн-ревью

 Tinkoff UI

 Дизайн-система V1

 Дизайн-система V2

 Нестандартные кейсы

 Дизайн-ревью

 Итоги

Tinkoff UI (a.k.a TUI)

Универсальная дизайн-система
для мобильных приложений Тинькофф

Tinkoff UI



Палитра, иконки,
типографика



Более 40 компонентов
разного уровня



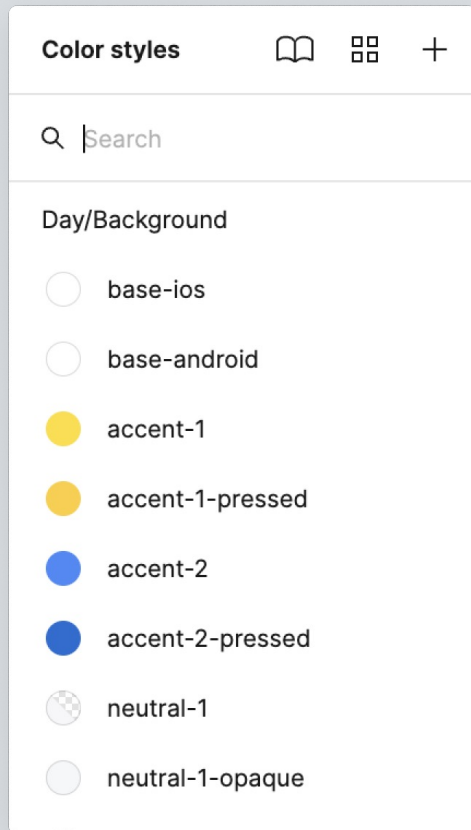
Пресеты



Анимации



Кастомные тени



Tinkoff UI



Палитра, иконки,
типографика



Более 40 компонентов
разного уровня



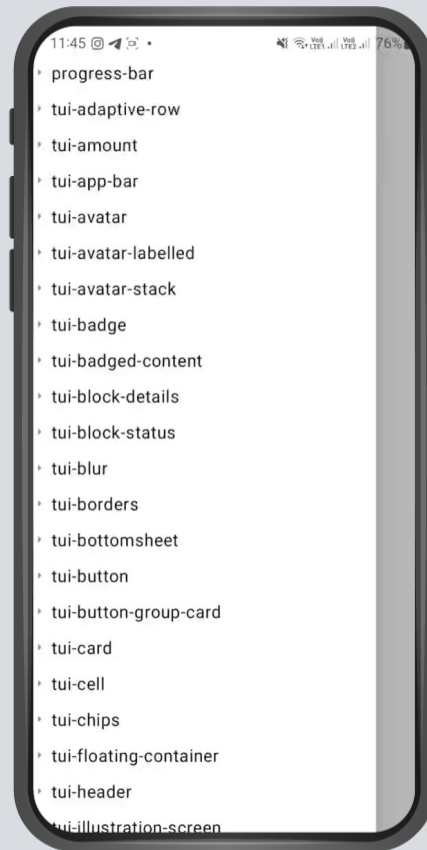
Пресеты



Анимации



Кастомные тени



Tinkoff UI



Палитра, иконки,
типографика



Более 40 компонентов
разного уровня



Пресеты



Анимации



Кастомные тени

★ Primary - tinkoff

★ Primary - opposite

★ Primary - tinkoff

★ Primary - opposite

Tinkoff UI



Палитра, иконки,
типографика



Более 40 компонентов
разного уровня



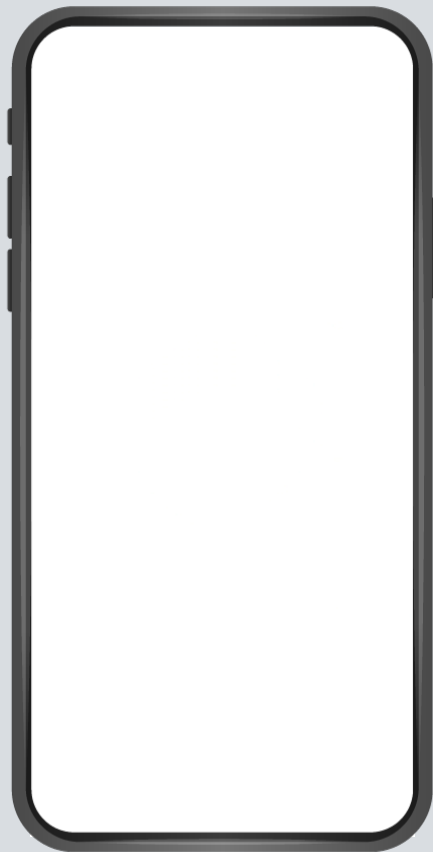
Пресеты



Анимации



Кастомные тени



Tinkoff UI



Палитра, иконки,
типографика



Более 40 компонентов
разного уровня



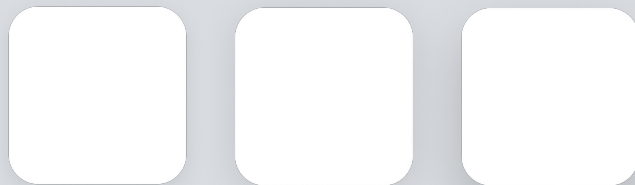
Пресеты



Анимации



Кастомные тени



Tinkoff UI



Уже реализована на View





Compose стал приходить в массы



Дизайн-система V1

- ➔ Tinkoff UI
- ➔ **Дизайн-система V1**
- ➔ Дизайн-система V2
- ➔ Нестандартные кейсы
- ➔ Дизайн-ревью
- ➔ Итоги



Tema

```
@Composable
fun TuiTheme(
    colors: TuiColors = TuiTheme.current,    ...
) {
    CompositionLocalProvider(LocalTuiColors provides colors, ...) {...}
}
```

Tema

```
object TuiTheme {  
    val colors: TuiColors  
        @Composable  
        get() = LocalTuiColors.current  
  
    val typography: TuiTypography  
        @Composable  
        get() = LocalTuiTypography.current  
  
    val shadows: TuiShadows  
        @Composable  
        get() = LocalTuiShadows.current  
    ...  
}
```

Компоненты V1

01

15 по-разному
реализованных
КОМПОНЕНТОВ

Компоненты V1

01

15 по-разному
реализованных
КОМПОНЕНТОВ

02

Сложно сохранять
бинарную совместимость

Компоненты V1

01

15 по-разному
реализованных
КОМПОНЕНТОВ

02

Сложно сохранять
бинарную совместимость

03

Пресеты
выглядят криво

Компоненты V1

01

15 по-разному
реализованных
компонентов

02

Сложно сохранять
бинарную совместимость

03

Пресеты
выглядят криво

04

Нет помощи со стороны
команд мобильных
приложений

Требования



API

Единообразное,
понятное и простое

Требования



API

Единообразное,
понятное и простое



Архитектура

Не должна сказываться
на производительности

Требования



API

Единообразное,
понятное и простое



Архитектура

Не должна сказываться
на производительности



Бинарная совместимость

Дешевые бинарно-
совместимые
изменения

Требования



API

Единообразное,
понятное и простое



Архитектура

Не должна сказываться
на производительности



Бинарная СОВМЕСТИМОСТЬ

Дешевые бинарно-
совместимые
изменения



Пресеты

Возможность легко
их создавать
и использовать



Material 3 API

```
@Composable
fun Button(
    ...,
    shape: Shape = ButtonDefaults.shape,
    colors: ButtonColors = ButtonDefaults.buttonColors(),
    elevation: ButtonElevation? = ButtonDefaults.buttonElevation(),
    border: BorderStroke? = null,
    ...
    content: @Composable RowScope.() -> Unit
) { ... }
```

Material 3 API

```
@Composable
fun Button(
    ...,
    shape: Shape = ButtonDefaults.shape,
    colors: ButtonColors = ButtonDefaults.buttonColors(),
    elevation: ButtonElevation? = ButtonDefaults.buttonElevation(),
    border: BorderStroke? = null,
    ...
    content: @Composable RowScope.() -> Unit
) { ... }
```


Material 3 API

```
@Composable
fun Button(
    ...,
    shape: Shape = ButtonDefaults.shape,
    colors: ButtonColors = ButtonDefaults.buttonColors(),
    elevation: ButtonElevation? = ButtonDefaults.buttonElevation(),
    border: BorderStroke? = null,
    ...
    content: @Composable RowScope.() -> Unit
) { ... }
```


Material 3 API

```
@Composable
fun Button(
    ...,
    shape: Shape = ButtonDefaults.shape,
    colors: ButtonColors = ButtonDefaults.buttonColors(),
    elevation: ButtonElevation? = ButtonDefaults.buttonElevation(),
    border: BorderStroke? = null,
    ...
    content: @Composable RowScope.() -> Unit
) { ... }
```

Как сделать такую кнопку?

 Download

Material 3 API

```
@Composable
fun Button(
    ...,
    shape: Shape = ButtonDefaults.shape,
    colors: ButtonColors = ButtonDefaults.buttonColors(),
    elevation: ButtonElevation? = ButtonDefaults.buttonElevation(),
    border: BorderStroke? = null,
    ...
    content: @Composable RowScope.() -> Unit
) { ... }
```

Material 3 API

```
class ButtonColors internal constructor(...) {
```

Material 3 API

```
@Composable
fun buttonColors(
    containerColor: Color = ...,
    contentColor: Color = ...
): ButtonColors {...}
```

Material: Итоги



Плохо расширяется

Особенно Material 3

Material: Итоги



Плохо расширяется

Особенно Material 3



Версии выходят одна за другой

В итоге рано или поздно понадобится миграция

Material: Итоги



Плохо расширяется

Особенно Material 3



Версии выходят одна за другой

В итоге рано или поздно понадобится миграция



Разработчики могут использовать другой Material

Дизайн-система V2

- ➔ Tinkoff UI
- ➔ Дизайн-система V1
- ➔ **Дизайн-система V2**
- ➔ Нестандартные кейсы
- ➔ Дизайн-ревью
- ➔ Итоги



Material все-таки используется

```
CompositionLocalProvider(LocalContentColor provides someColor) { ... }
```

Интероп с Material

```
build.gradle
```

```
compileOnly("androidx.compose.material:material:XXX")
```

Интероп с Material

- Используем `CompositionLocal` из `Material`, если он подключен

```
val LocalTuiContentColor by lazy(PUBLICATION) {  
    when {  
        isMaterialAvailable -> LocalContentColor  
        else -> compositionLocalOf { Color.Black }  
    }  
}
```

Интероп с Material

- Используем `CompositionLocal` из `Material`, если он подключен

```
val LocalTuiContentColor by lazy(PUBLICATION) {  
    when {  
        isMaterialAvailable -> LocalContentColor  
        else -> compositionLocalOf { Color.Black }  
    }  
}
```

Compose- КОМПОНЕНТЫ на базе View

Сделать первую версию
используя View

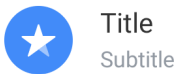


Header



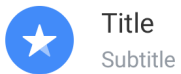
Title
Subtitle

Header



```
@Composable
fun Card(
    header: (@Composable () -> Unit)? = null,
    ...
) {...}
```

Header



```
@Composable
fun Card(
    header: (@Composable () -> Unit)? = null,
    ...
) {
    AndroidView(
        factory = ::TuiLargeCardLayout,
        ...
    )
}
```


Header



Title
Subtitle

```
@Composable
fun Card(
    header: (@Composable () -> Unit)? = null,
    ...
) {
    AndroidView(
        factory = ::TuiLargeCardLayout,
        update = {
            if (header != null) {
                card.addView(
                    ComposeView(it.context).apply {
                        setContent { header() }
                    })
            } else { ... }
        })
}
```

Header



Title
Subtitle

```
@Composable
fun Card(
    header: (@Composable () -> Unit)? = null,
    ...
) {
    AndroidView(
        factory = ::TuiLargeCardLayout,
        update = {
            if (header != null) {
                card.addView(
                    ComposeView(it.context).apply {
                        setContent { header() }
                    })
            } else { ... }
        })
}
```


Минусы



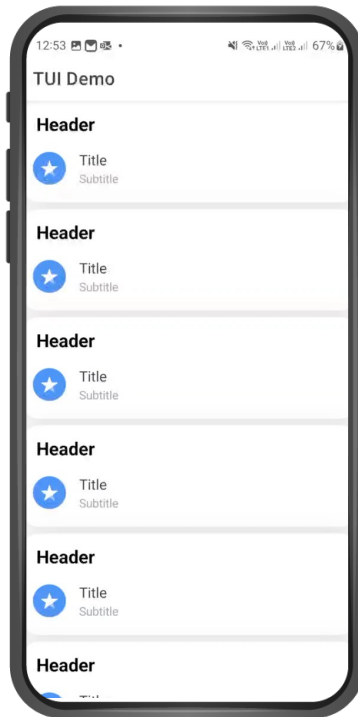
**Большая иерархия вложенных View
(~в 2 раза больше View-варианта)**

Замерим производительность

```
LazyColumn {  
    repeat(500) {  
        item {  
            Card(...)  
        }  
    }  
}
```

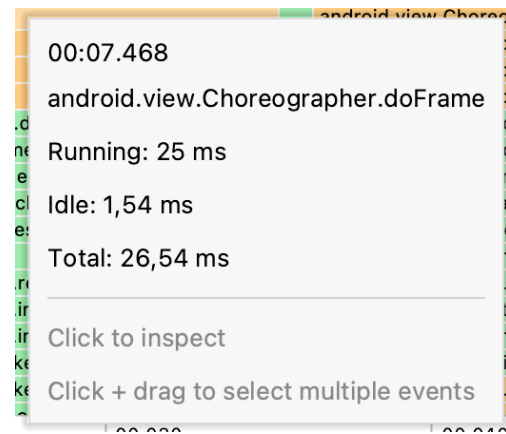
Замерим производительность

```
LazyColumn {  
    repeat(500) {  
        item {  
            Card(...)  
        }  
    }  
}
```



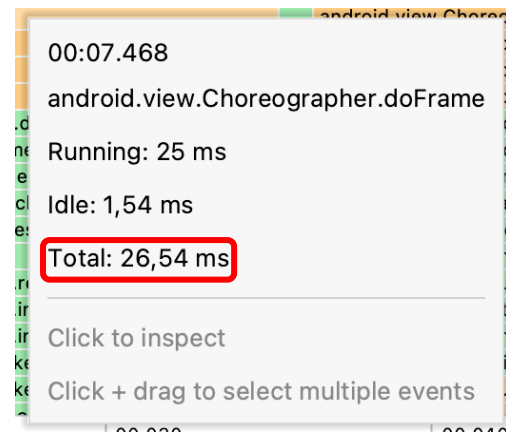
Замерим производительность

```
LazyColumn {  
    repeat(500) {  
        item {  
            Card(...)  
        }  
    }  
}
```



Замерим производительность

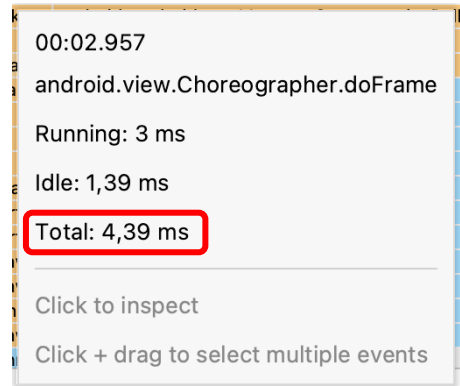
```
LazyColumn {  
    repeat(500) {  
        item {  
            Card(...)  
        }  
    }  
}
```



Замерим производительность

```
LazyColumn {  
    repeat(500) {  
        item {  
            Card(...)  
        }  
    }  
}
```

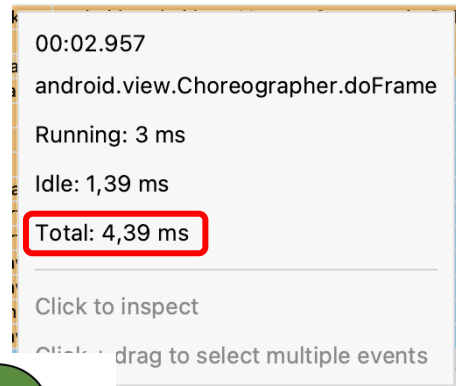
Compose Only:



Замерим производительность

```
LazyColumn {  
    repeat(500) {  
        item {  
            Card(...)  
        }  
    }  
}
```

Compose Only:



Минусы



**Большая иерархия вложенных View
(~в 2 раза больше View-варианта)**



**Низкая производительность решения
Compose – View – Compose**



Appearance + Size

Appearance

- Всё, что касается внешнего вида компонента

```
interface TuiButtonAppearance {  
    @Composable  
    fun backgroundBrush(enabled: Boolean): Brush  
  
    @Composable  
    fun fadeAnimationSpec(): AnimationSpec<Float>  
  
    ...  
}
```

Appearance

- Всё, что касается внешнего вида компонента

```
interface TuiButtonAppearance {  
    @Composable  
    fun backgroundBrush(enabled: Boolean): Brush  
  
    @Composable  
    fun fadeAnimationSpec(): AnimationSpec<Float>  
  
    ...  
}
```

Appearance

- Всё, что касается внешнего вида компонента

```
interface TuiButtonAppearance {  
    @Composable  
    fun backgroundBrush(enabled: Boolean): Brush  
  
    @Composable  
    fun fadeAnimationSpec(): AnimationSpec<Float>  
  
    ...  
}
```

★ Primary - tinkoff

★ Primary - opposite

Size

- Всё, что касается размеров

```
interface TuiButtonSize {  
    @Composable  
    fun spaceBetweenIconAndText(): Dp  
  
    @Composable  
    fun horizontalTextPadding(): Dp  
  
    ...  
}
```

Size

- Всё, что касается размеров

```
interface TuiButtonSize {  
    @Composable  
    fun spaceBetweenIconAndText(): Dp  
  
    @Composable  
    fun horizontalTextPadding(): Dp  
  
    ...  
}
```

★ Primary - tinkoff

★ Primary - tinkoff

Size

- Всё, что касается размеров

```
interface TuiButtonSize {  
    @Composable  
    fun spaceBetweenIconAndText () : Dp  Может, стейт?  
  
    @Composable  
    fun horizontalTextPadding () : Dp  
  
    ...  
}
```

State удобен для анимаций

```
@Composable
fun containerColor(enabled: Boolean): State<Color> {
    return animateColorAsState(if (enabled) color else disabledColor)
}
```

State удобен для анимаций

```
@Composable
fun containerColor(enabled: Boolean): State<Color> {
    return animateColorAsState(if (enabled) color else disabledColor)
}
```

Animated Button

Не используем State

- Чтобы выдавать только нужное API для ан

```
interface TuiNotificationMiddleAppearance {  
  
    @Composable  
    fun dialogOpenAnimationSpec(): AnimationSpec<Float>  
  
    @Composable  
    fun dialogCloseAnimationSpec(): AnimationSpec<Float>  
}
```

State на стороне пользователей

```
val appearance = remember {  
    TuiButtonAppearanceTinkoff.primaryTinkoff.copy(  
        backgroundBrush = {  
            rememberUpdatedState(newValue = ...) )  
    }  
}
```

State на стороне пользователей

```
val appearance = remember {  
    TuiButtonAppearanceTinkoff.primaryTinkoff.copy(  
        backgroundBrush = {  
            rememberUpdatedState(newValue = ...) )  
    }  
}
```


Кодген для реализации интерфейсов

```
// generated code
inline fun TuiButtonSize(
    crossinline spaceBetweenIconAndText: @Composable TuiButtonSize.() -> Dp
): TuiButtonSize = object : TuiButtonSize {

    @Composable
    override fun spaceBetweenIconAndText(): Dp = spaceBetweenIconAndText(this)

}
```

Кодген для реализации интерфейсов

```
// generated code
inline fun TuiButtonSize(
    crossinline spaceBetweenIconAndText: @Composable TuiButtonSize.() -> Dp
): TuiButtonSize = object : TuiButtonSize {

    @Composable
    override fun spaceBetweenIconAndText(): Dp = spaceBetweenIconAndText(this)

}
```

Кодген для реализации интерфейсов

```
// generated code
inline fun TuiButtonSize(
    crossinline spaceBetweenIconAndText: @Composable TuiButtonSize.() -> Dp
): TuiButtonSize = object : TuiButtonSize {

    @Composable
    override fun spaceBetweenIconAndText(): Dp = spaceBetweenIconAndText(this)

}
```


Не забыли про копирование

- Для использования стиля как базы для другого

```
val primaryAccent = default.copy(  
    backgroundBrush = { ... }  
)
```

Теперь легко собрать компонент

```
object TuiButtonSizeTinkoff {  
  ...  
}
```



```
object TuiButtonAppearanceTinkoff {  
  ...  
}
```



```
TuiButton(  
  onClick = { },  
  size = TuiButtonSizeTinkoff.small,  
  appearance = TuiButtonAppearanceTinkoff.primaryTinkoff  
)
```

Для упрощения API делаем перегрузки

```
@Composable
fun TuiCell(
    title: @Composable () -> Unit,
) {}
```

Для упрощения API делаем перегрузки

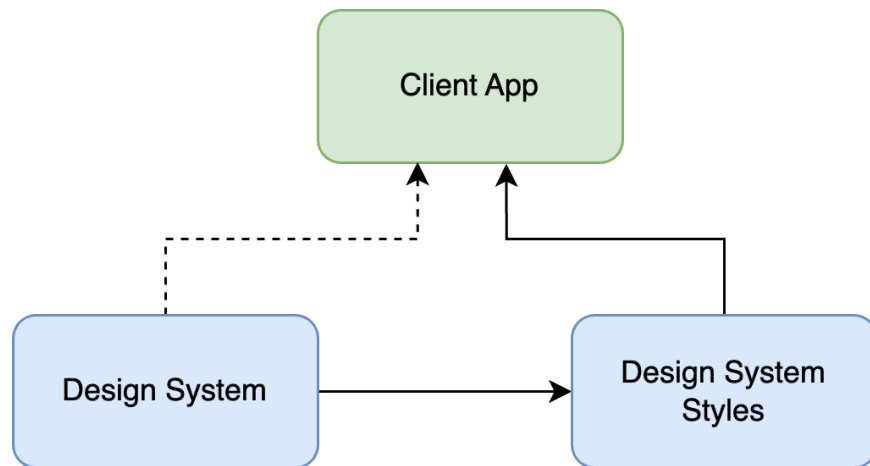
```
@Composable
fun TuiCell(
    title: String
) {
    TuiCell(
        title = {
            ProvideTextStyle(...) {
                Text(text = title)
            }
        })
}
```

Для упрощения API делаем перегрузки

```
@Composable
fun TuiCell(
    title: String
) {
    TuiCell(
        title = {
            ProvideTextStyle(...) {
                Text(text = title)
            }
        })
}
```

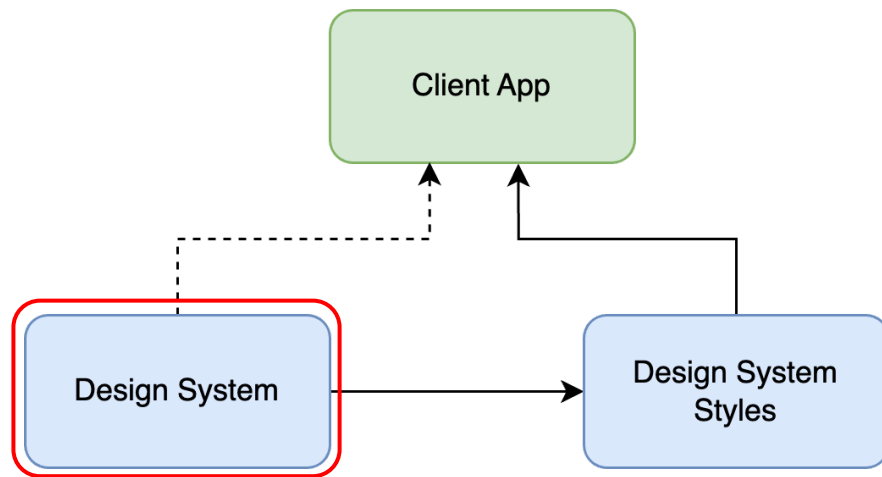

Многомодульная структура

Разделили компоненты и их стили по разным модулям



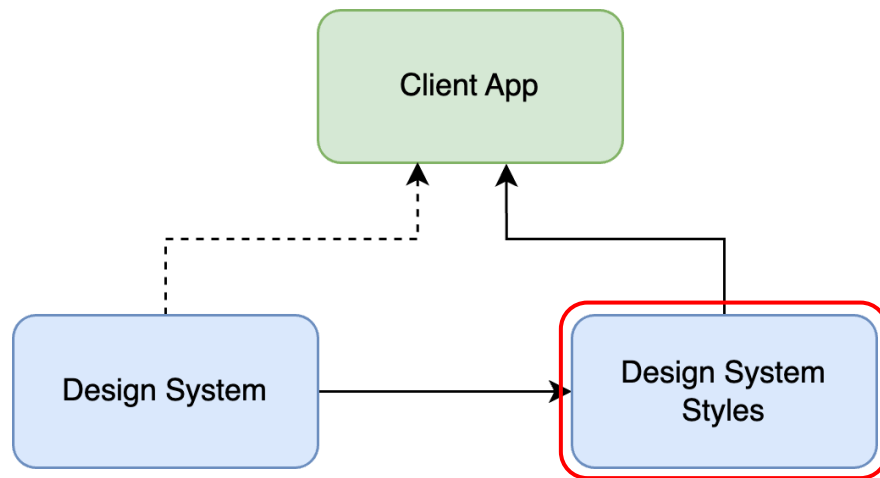
Многомодульная структура

Разделили компоненты и их стили по разным модулям



Многомодульная структура

Разделили компоненты и их стили по разным модулям



Промежуточные результаты



Не стали использовать View



Реализовали интероп с Material



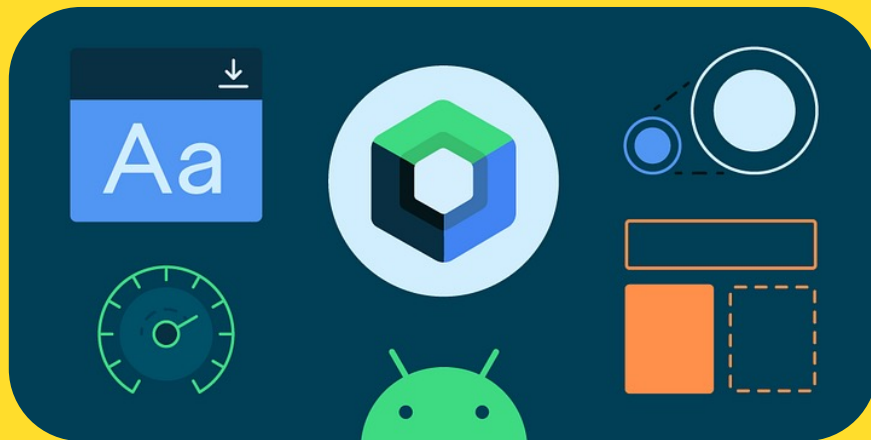
Выбрали архитектуру для API
компонентов



Отделили реализацию
компонентов от их стилизации

Нестандартные кейсы

- ➔ Tinkoff UI
- ➔ Дизайн-система V1
- ➔ Дизайн-система V2
- ➔ **Нестандартные кейсы**
- ➔ Дизайн-ревью
- ➔ Итоги



Контент с бейджем



Реализация с костылем

```
@Composable
fun TuiBadgedContent() {
    Box {
        Avatar()
        Badge(...)
    }
}
```

Реализация с костылем

```
@Composable
fun TuiBadgedContent() {
    Box {
        Avatar()
        Badge(
            modifier = Modifier
                .align(Alignment.BottomEnd)
                .offset(x = (-3).dp, y = (-3).dp)
        )
    }
}
```

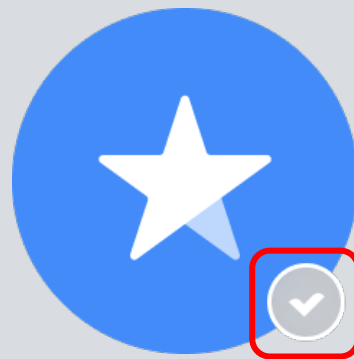

Контент с бейджем



Бейдж может быть любого размера и формы



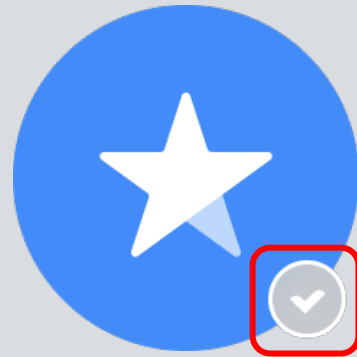
Располагается строго в углу, обязательно под 45 градусов



Контент с бейджем




Должны быть якорные точки



Alignment lines

The screenshot shows the Android Developers website interface. At the top, there's a navigation bar with 'developers' logo, 'Essentials', 'Design & Plan', 'Develop', and 'Google Play'. A search bar and 'Language' dropdown are on the right. Below this is a breadcrumb trail: 'MODERN ANDROID > COMPOSE'. A secondary navigation bar includes 'Overview', 'Tutorial', 'Samples', 'Guides' (which is active), and 'For teams'. A left sidebar contains a 'Filter' box and a list of categories: 'UI architecture', 'Develop your app's layout' (with sub-items like 'Overview', 'Layout basics', 'Modifiers', etc.), 'Components', 'Theming', 'Text and typography', 'Images and graphics', and 'Animation'. The main content area is titled 'Alignment lines in Jetpack Compose' and includes an introductory paragraph, a paragraph about the Compose layout model, a paragraph about alignment lines in Compose, and a paragraph about a custom `LayoutModifier`. Below the text are two diagrams: one showing 'Text with padding to first baseline' with a 24.dp padding arrow and a red line for the 'First baseline', and another showing 'Text with normal padding' with a 24.dp padding arrow.

developers  Essentials ▾ Design & Plan ▾ Develop Google Play

🔍 Search 🌐 Language ▾ [Android Studio](#) [Войти](#)

MODERN ANDROID > COMPOSE

Overview Tutorial Samples **Guides** For teams

Filter

UI architecture ▾

Develop your app's layout ▾

- Overview
- Layout basics
- Modifiers
- List of modifiers
- Pager
- Flow layouts
- Custom layouts
- Build adaptive layouts
- Alignment lines**
- Intrinsic measurements
- ConstraintLayout

Components ▾

Theming ▾

Text and typography ▾

Images and graphics ▾

Animation ▾

Android Developers > Modern Android > Compose > Guides Эта информация оказалась полезной? 🍏 🗨

Alignment lines in Jetpack Compose


The Compose layout model lets you use `AlignmentLine` to create custom alignment lines that can be used by parent layouts to align and position their children. For example, `Row` can use its children's custom alignment lines to align them.

When a layout provides a value for a particular `AlignmentLine`, the layout's parents can read this value after measuring, using the `Placeable.get` operator on the corresponding `Placeable` instance. Based on the position of the `AlignmentLine`, the parents can then decide the positioning of the children.

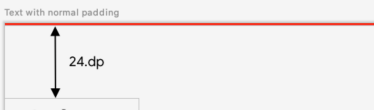
Some composables in Compose already come with alignment lines. For example, the `BasicText` composable exposes the `FirstBaseline` and `LastBaseline` alignment lines.

In the example below, a custom `LayoutModifier` called `firstBaselineToTop` reads the `FirstBaseline` to add padding to the `Text` starting from its first baseline.

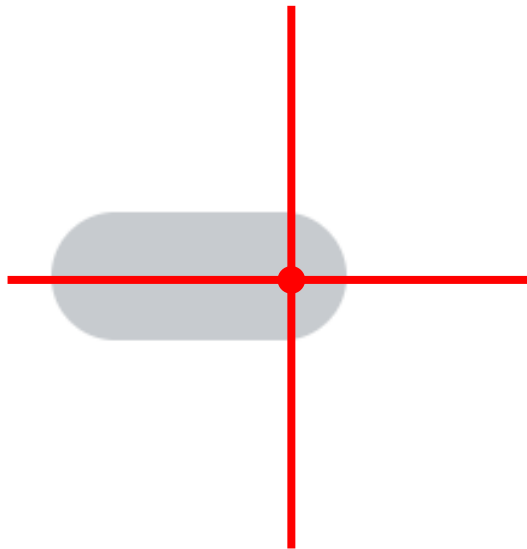
Text with padding to first baseline



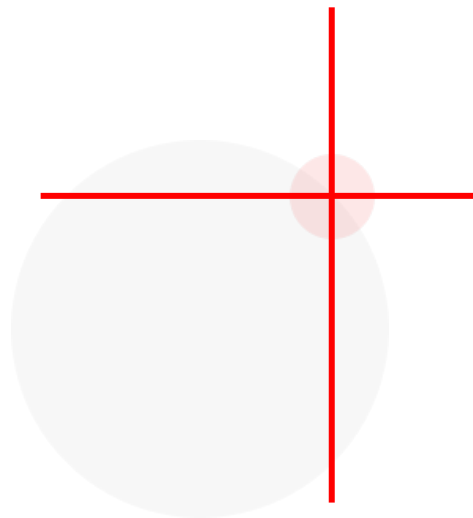
Text with normal padding



Alignment Lines для бейджа



Alignment Lines для контейнера



Создаем Alignment Lines

- Создадим Alignment Line-ы для бейджей и контента

```
val BadgeEndAnchorX = VerticalAlignmentLine (::maxOf)
val BadgeEndAnchorY = HorizontalAlignmentLine (::maxOf)
```

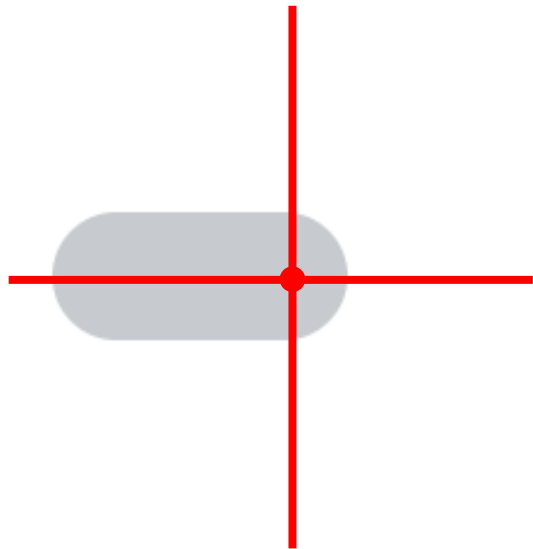
Создаем Alignment Lines

- Создадим Alignment Line-ы для бейджей и контента

```
val BadgeEndAnchorX = VerticalAlignmentLine (::maxOf)
val BadgeEndAnchorY = HorizontalAlignmentLine (::maxOf)

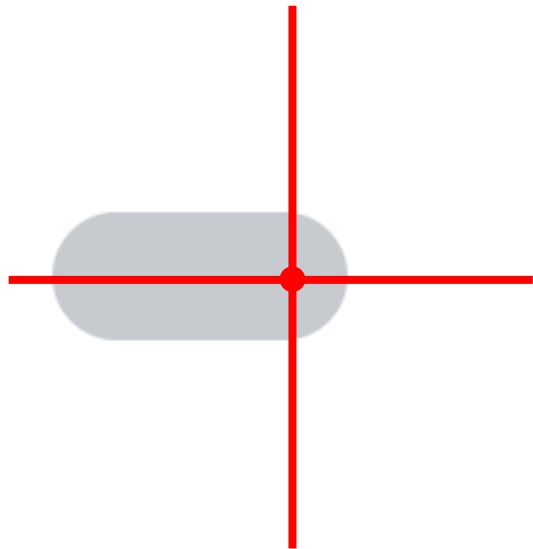
val ContentTopEndCornerX = VerticalAlignmentLine (::maxOf)
val ContentTopEndCornerY = HorizontalAlignmentLine (::maxOf)
```

Передаем Alignment Lines



```
layout {  
  ...  
  layout(  
    alignmentLines = mapOf(  
      BadgeEndAnchorX to ...,  
      BadgeEndAnchorY to ...  
    )  
  ) {...}  
}
```


Передаем Alignment Lines



```
layout {  
  ...  
  layout(  
    alignmentLines = mapOf(  
      BadgeEndAnchorX to ...,  
      BadgeEndAnchorY to ...  
    )  
  ) {...}  
}
```

Передаем Alignment Lines



```
layout {  
  ...  
  layout(  
    alignmentLines = mapOf(  
      ContentTopEndCornerX to ...,  
      ContentTopEndCornerY to ...  
    )  
  ) {...}  
}
```

API компонента

```
@Composable
fun TuiBadgedContent(
    ...
    bottomEndBadge: @Composable (() -> Unit)? = null,
    content: @Composable () -> Unit
) {
    Layout(measurePolicy = remember { MeasurePolicy { ...} })
}
```

API компонента

```
@Composable
fun TuiBadgedContent(
    ...
    bottomEndBadge: @Composable (() -> Unit)? = null,
    content: @Composable () -> Unit
) {
    Layout(measurePolicy = remember { MeasurePolicy { ...} })
}
```

Располагаем

```
MeasurePolicy {  
    ...  
    val badgeOffset = when (alignment) {  
        BadgeAlignment.TopEnd -> IntOffset(  
            contentPlaceable[ContentTopRightCornerX] - badgePlaceable[BadgeEndAnchorX],  
            contentPlaceable[ContentTopRightCornerY] - badgePlaceable[BadgeEndAnchorY]  
        )  
        ...  
    }  
  
    ...  
    badgePlaceable.placeRelative(badgeOffset)  
}
```

Располагаем

```
MeasurePolicy {  
    ...  
    val badgeOffset = when (alignment) {  
        BadgeAlignment.TopEnd -> IntOffset(  
            contentPlaceable[ContentTopRightCornerX] - badgePlaceable[BadgeEndAnchorX],  
            contentPlaceable[ContentTopRightCornerY] - badgePlaceable[BadgeEndAnchorY]  
        )  
        ...  
    }  
  
    ...  
    badgePlaceable.placeRelative(badgeOffset)
```

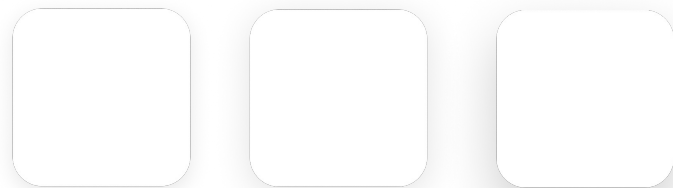
Располагаем

```
MeasurePolicy {  
    ...  
    val badgeOffset = when (alignment) {  
        BadgeAlignment.TopEnd -> IntOffset(  
            contentPlaceable[ContentTopRightCornerX] - badgePlaceable[BadgeEndAnchorX],  
            contentPlaceable[ContentTopRightCornerY] - badgePlaceable[BadgeEndAnchorY]  
        )  
        ...  
    }  
  
    ...  
    badgePlaceable.placeRelative(badgeOffset)  
}
```

Результат



Кастомные тени в Compose



Простой способ

```
modifier = Modifier.graphicsLayer {  
    shadowElevation = ...,  
    spotShadowColor = ...,  
    ambientShadowColor = ...  
}
```

Простой способ

```
modifier = Modifier.graphicsLayer {  
    shadowElevation = ...,  
    spotShadowColor = ...,  
    ambientShadowColor = ...  
}
```

Проблема теней



Title

Subtitle



Почему обрезается тень при $\alpha < 1$?

Offscreen Layer

Как решить проблему во View?

- Самим применить альфу!

```
override fun onSetAlpha(alpha: Int) = true
```

Как решить проблему во View?

- Самим применить альфу!

```
override fun onSetAlpha(alpha: Int) = true

override fun draw(canvas: Canvas) {
    val alpha = (alpha * 255).toInt()
    drawShadow(alpha)
    canvas.saveLayerAlpha(bounds, alpha)
    super.draw(canvas)
    canvas.restore()
}
```

Как решить проблему во View?

- Самим применить альфу!

```
override fun onSetAlpha(alpha: Int) = true

override fun draw(canvas: Canvas) {
    val alpha = (alpha * 255).toInt()
    drawShadow(alpha)
    canvas.saveLayerAlpha(bounds, alpha)
    super.draw(canvas)
    canvas.restore()
}
```


Как решить проблему в Compose?

- Способ 1: Composition Strategy

The screenshot shows the Android Developers website for the `CompositingStrategy` class. The page is in Russian and includes a navigation menu, a search bar, and a sidebar with a list of classes. The main content area displays the class name, a "Select a platform" dropdown, and the class signature: `value class CompositingStrategy`. Below this, there is a description: "Determines when to render the contents of a layer into an offscreen buffer before being drawn to the destination." The "Summary" section contains a table of public companion properties.

| Public companion properties | | |
|-----------------------------------------------------------------------------------------------------------------------------|----------------------------|-------------------|
| <code>CompositingStrategy</code> | <code>Auto</code> | <code>Comn</code> |
| Rendering to an offscreen buffer will be determined automatically by the rest of the <code>graphicsLayer</code> parameters. | | |
| <code>CompositingStrategy</code> | <code>ModulateAlpha</code> | <code>Comn</code> |
| Modulates alpha for each of the drawing instructions recorded within the <code>graphicsLayer</code> . | | |
| <code>CompositingStrategy</code> | <code>Offscreen</code> | <code>Comn</code> |

Composition Strategy + Alpha



Offscreen



ModulateAlpha

CompositionStrategy в деле

```
Modifier.graphicsLayer {  
    compositingStrategy =  
    ModulateAlpha  
    alpha = 0.5F  
}
```



Chip with shadow

```
Modifier.graphicsLayer {  
  compositingStrategy = ModulateAlpha  
  alpha = 0.5F  
}
```

Хардкорное решение

```
layout {...} then layout {...}
```



Chip with shadow

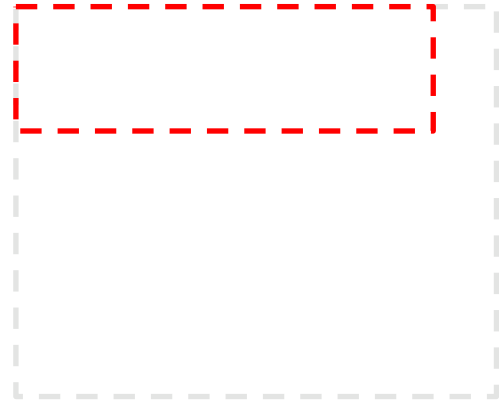
Хардкорное решение

```
layout {...} then layout {  
  // Увеличить размер лейаута на размер  
  тени  
}
```



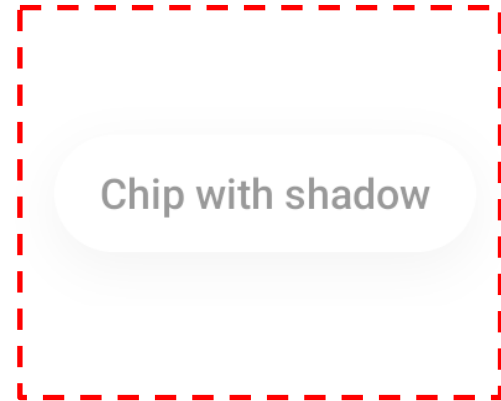
Хардкорное решение

```
layout {  
  // Вернуть размер как он был  
  
} then layout {  
}
```



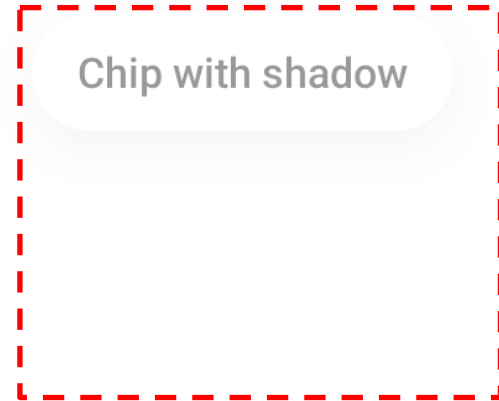
Хардкорное решение

```
layout {  
  layout(...) {  
  
    // разместиться в увеличенный лейаут  
    // и применить ему layer  
  
    placeable.placeWithLayer(  
      shadowWidth,  
      shadowLength,  
      layerBlock = layerBlock  
    )  
  }  
} then layout {  
}
```



Хардкорное решение

```
layout {...} then layout {  
    layout(...) {  
  
        // переместиться на начальное  
        положение  
  
        placeable.place(  
            x = -shadowWidth,  
            y = -shadowLength  
        )  
    }  
}
```



Что получилось

Было:



Chip with shadow

Что получилось

Было:



Chip with shadow

Стало:



Chip with shadow

Выводы



В Compose много интересных инструментов для UI



Инструменты легко использовать не так, как надо



Из коробки нет решения проблемы кастомных теней

- ➔ Tinkoff UI
- ➔ Дизайн-система V1
- ➔ Дизайн-система V2
- ➔ Нестандартные кейсы
- ➔ **Дизайн-ревью**
- ➔ Итоги

Дизайн-ревью



★ Primary - tinkoff

Дизайн-ревью



Title

Description

Text >

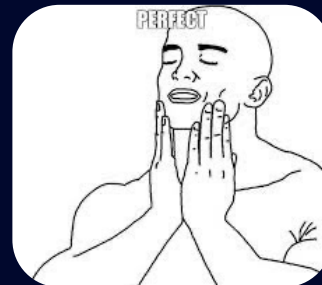
Дизайн-ревью



Title

Description

Text >



Дизайн-ревью

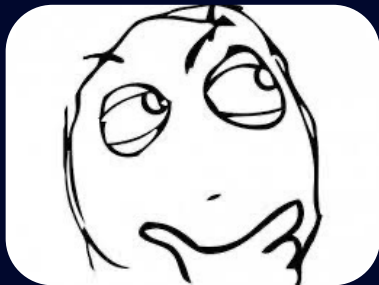


Title

Description

Text >

Почему поехал текст??



Заглянем в Material

```
@Composable  
fun MaterialTheme(...) {  
  
}
```

Заглянем в Material

```
@Composable
fun MaterialTheme(...) {
    ProvideTextStyle(value = typography.body1) {
        ...
    }
}
```

Заглянем в Material

```
@Composable
fun MaterialTheme(...) {
    ProvideTextStyle(value = typography.body1) {
        PlatformMaterialTheme(content)
    }
}
```

Всё ясно

```
body1: TextStyle = TextStyle(  
    fontWeight = FontWeight.Normal,  
    fontSize = 16.sp,  
    letterSpacing = 0.5.sp  
)
```

Всё ясно

```
body1: TextStyle = TextStyle(  
  fontWeight = FontWeight.Normal,  
  fontSize = 16.sp,  
  letterSpacing = 0.5.sp  
)
```



Но есть еще кое-что



Title

Description

Text >

VS

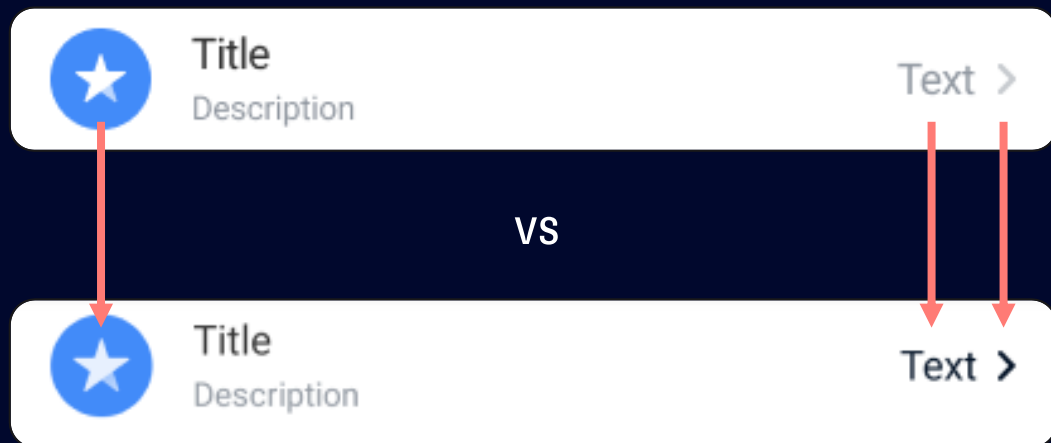


Title

Description

Text >

Но есть еще кое-что



Догадываемся, куда копать

```
@Composable  
fun MaterialTheme(...) {  
  
}
```

Догадываемся, куда копать

```
@Composable
fun MaterialTheme(...) {
    CompositionLocalProvider(
        LocalContentAlpha provides ContentAlpha.high
    ) {...}
}
```

Догадываемся, куда копать

```
@Composable
fun MaterialTheme(...) {
    CompositionLocalProvider(
        LocalContentAlpha provides ContentAlpha.high // 0.87
    ) {...}
}
```

Всё ясно

```
@Composable
fun Text(...) {
    val textColor = color.takeOrElse {
        style.color.takeOrElse {
            LocalContentColor.current.copy(alpha = LocalContentAlpha.current)
        }
    }
}
```

Всё ясно

```
@Composable
fun Text(...) {
    val textColor = color.takeOrElse {
        style.color.takeOrElse {
            LocalContentColor.current.copy(alpha = LocalContentAlpha.current)
        }
    }
}
```

Исправляем проблемы

```
@Composable
fun FixMaterialTheme(...) {
    CompositionLocalProvider(
        LocalTuiContentAlpha provides 1F,
        LocalTuiTextStyle provides TuiTheme.font.body.medium
    ) {...}
}
```

Исправляем проблемы

```
@Composable
fun FixMaterialTheme(...) {
    CompositionLocalProvider(
        LocalTuiContentAlpha provides 1F,
        LocalTuiTextStyle provides TuiTheme.font.body.medium
    ) {...}
}
```

Исправляем проблемы

```
@Composable
fun FixMaterialTheme(...) {
    CompositionLocalProvider(
        LocalTuiContentAlpha provides 1F,
        LocalTuiTextStyle provides TuiTheme.font.body.medium
    ) {...}
}
```


Исправляем проблемы

```
MaterialTheme {  
    FixMaterialTheme {  
        ...  
    }  
}
```

Результаты дизайн-ревью



Избегаем Material-компонентов



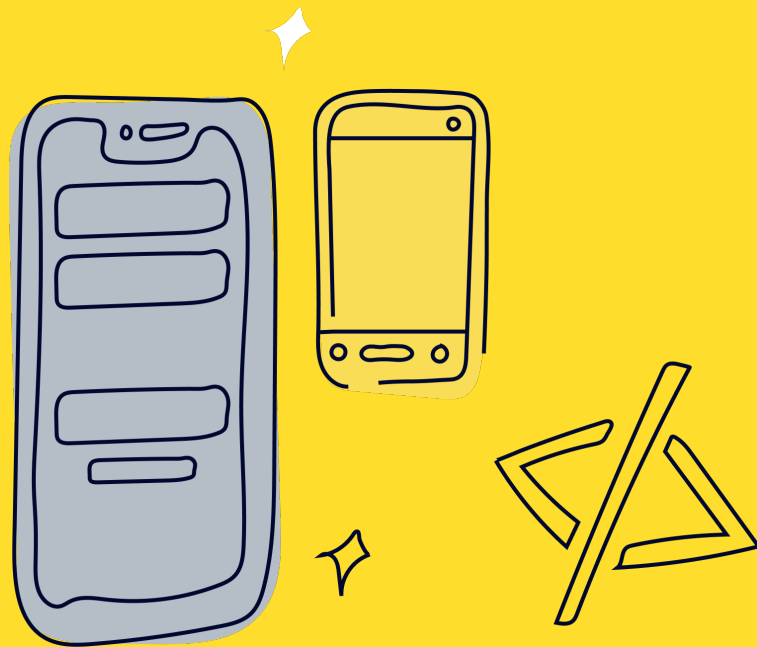
Избегаем использование
публичных CompositionLocal



Вносим правки в тему, чтобы
победить проблемы Material

- ➔ Tinkoff UI
- ➔ Дизайн-система V1
- ➔ Дизайн-система V2
- ➔ Нестандартные кейсы
- ➔ Дизайн-ревью
- ➔ **Итоги**

ИТОГИ



ИТОГИ



Выработали подход к реализации компонентов на Compose



Реализовали 15+ компонентов



Получили InnerSource



Перестали бояться Material

Про организацию дизайн-системы



Как создавали универсальную дизайн-систему для приложений Тинькофф – Михайил Стручков, Тинькофф



Посмотреть аналитику

Изменить видео

Поделиться

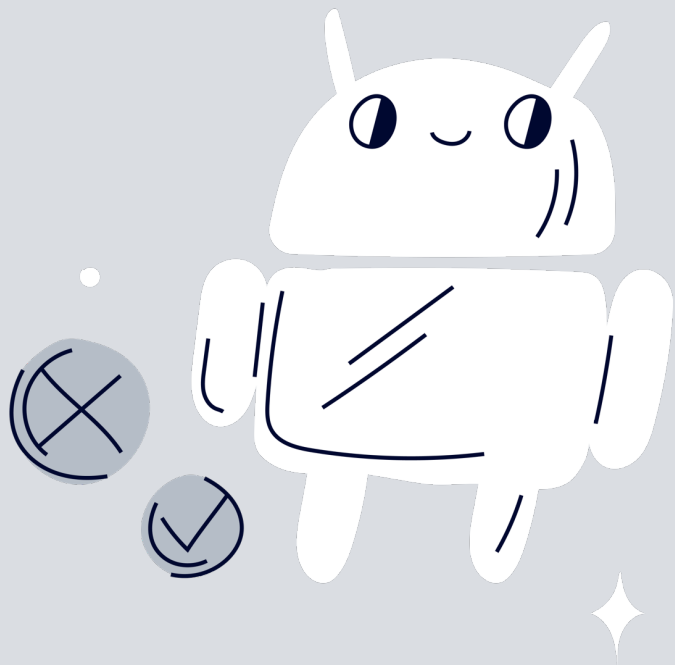
Сохранить



Смотреть
видео



[https://www.youtube.com/
watch?v=-PZeTbh1ywQ](https://www.youtube.com/watch?v=-PZeTbh1ywQ)



@m1hastr



m1hastr@yandex.ru



@valya1



ТИНЬКОФФ