LITHO: BEST PRACTICES FOR BUILDING EFFICIENT UI

Sergey Ryabov
Facebook

AGENDA

- ✦ What is Litho
- ✦ How does it work
- ✦ State management
- ✦ Animations
- ✦ Interop
- ✦ Tools

WHAT IS LITHO

WHAT IS LITHO

WHAT IS LITHO

- UI framework

WHAT IS LITHO

- UI framework
- 4.5 years old

WHAT IS LITHO

- UI framework
- 4.5 years old
- Used in Facebook apps

WHAT IS LITHO

- UI framework
- 4.5 years old
- Used in Facebook apps
- *But not only...*

WHAT IS LITHO

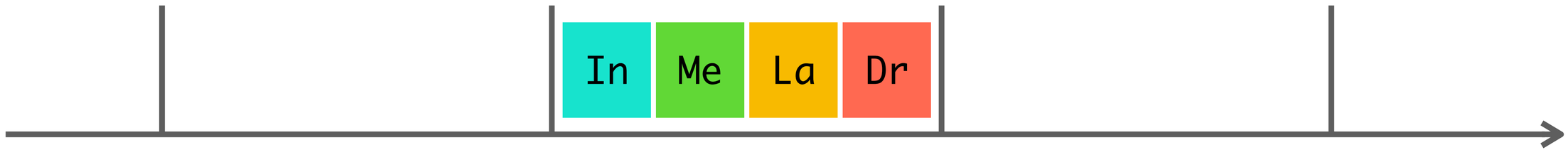
- ✦ UI framework
- ✦ 4.5 years old
- ✦ Used in Facebook apps
- ✦ *But not only...*

appbrain.com/stats/libraries/details/litho/litho

WHAT IS LITHO ABOUT

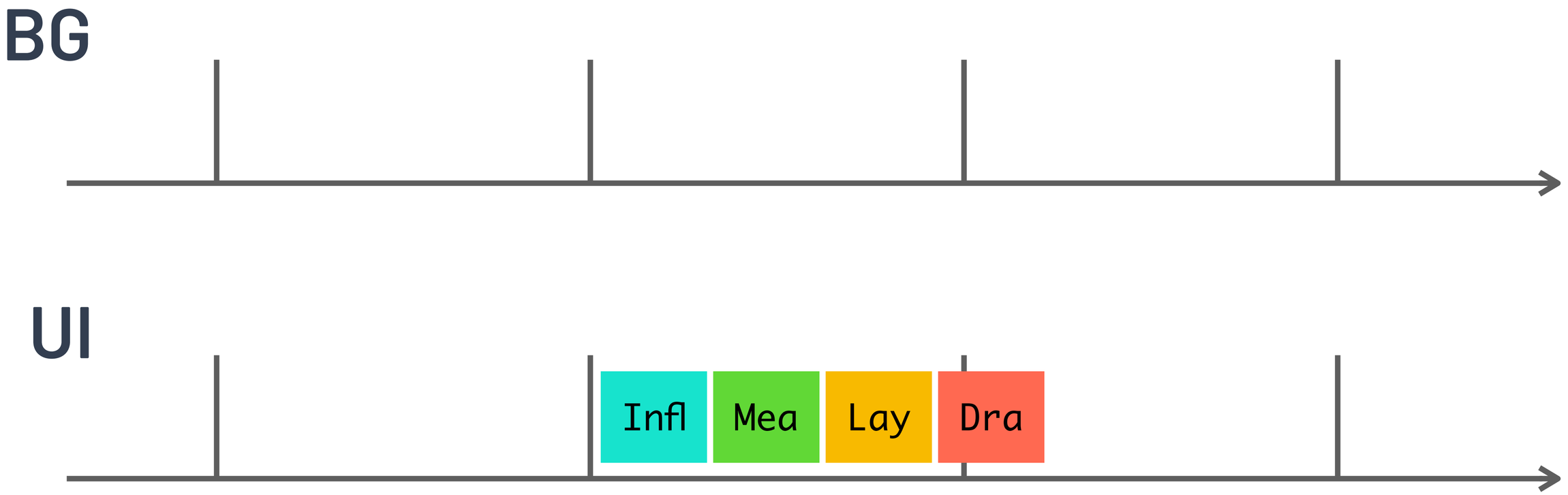
WHAT IS LITHO ABOUT

WHAT IS LITHO ABOUT

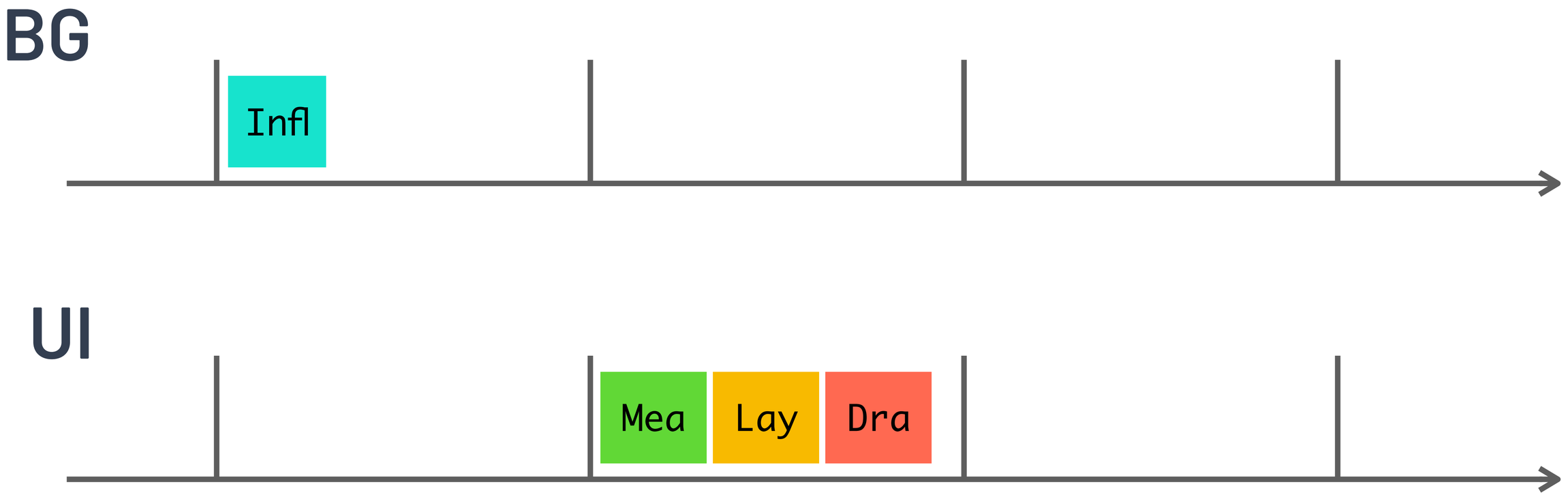


WHAT IS LITHO ABOUT

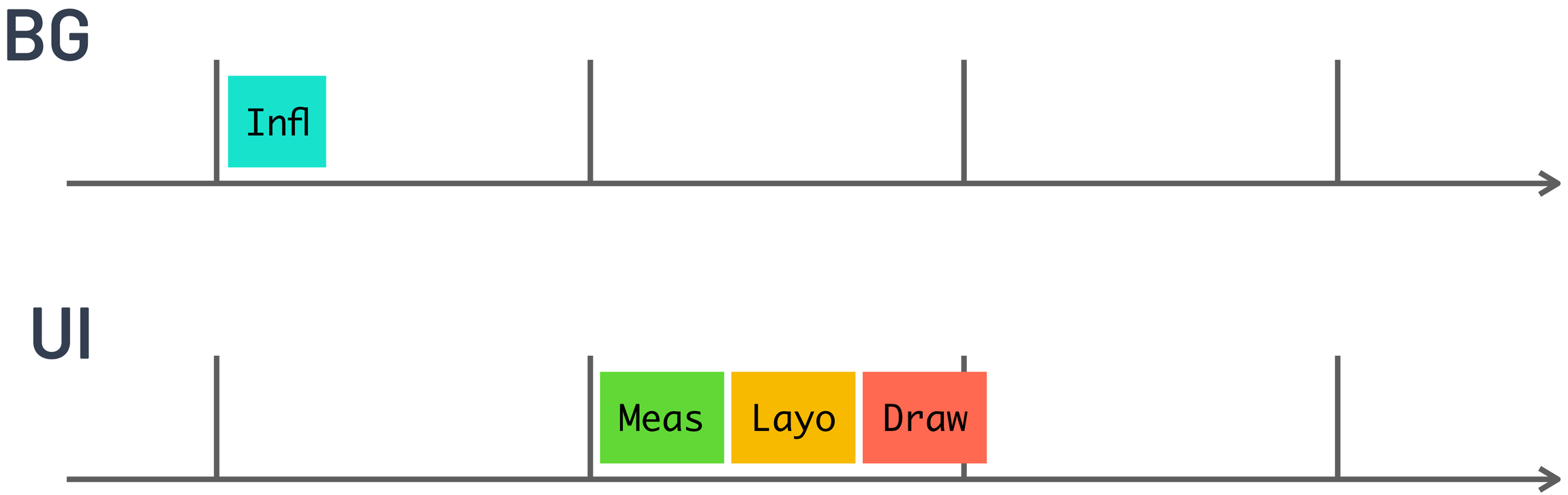
WHAT IS LITHO ABOUT



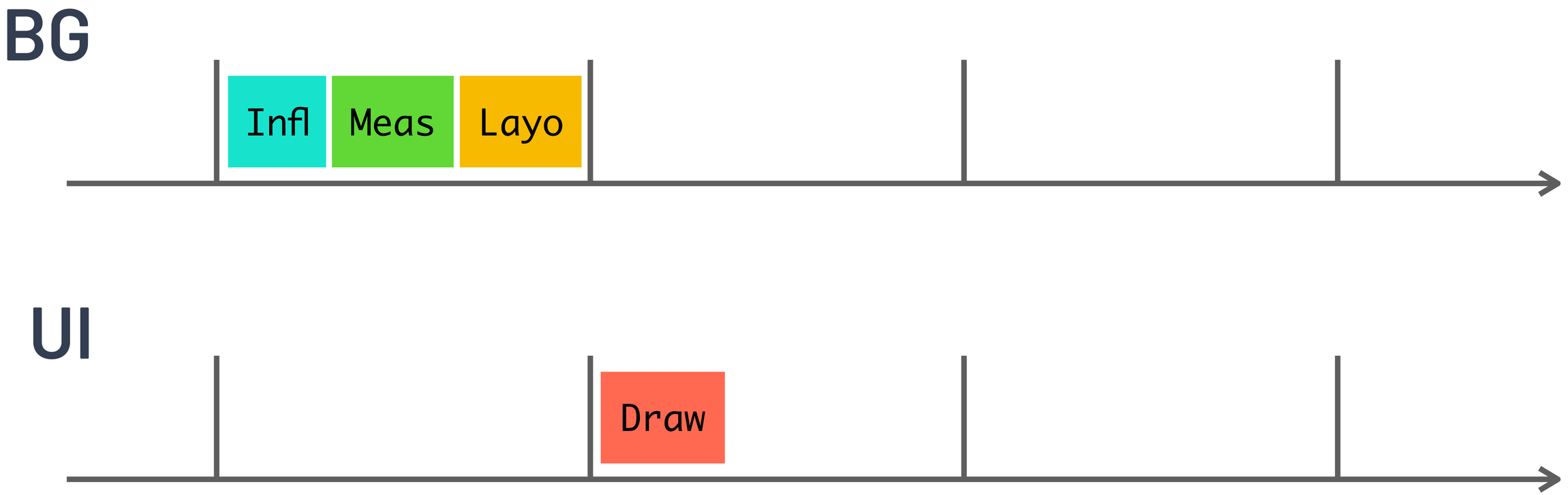
WHAT IS LITHO ABOUT



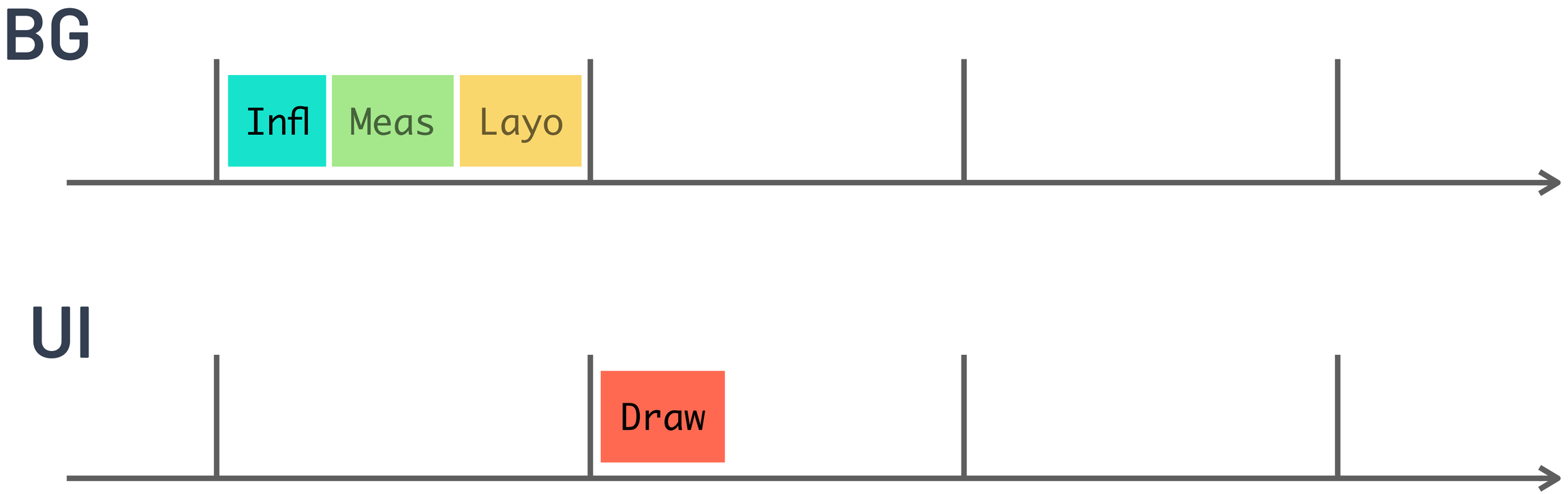
WHAT IS LITHO ABOUT



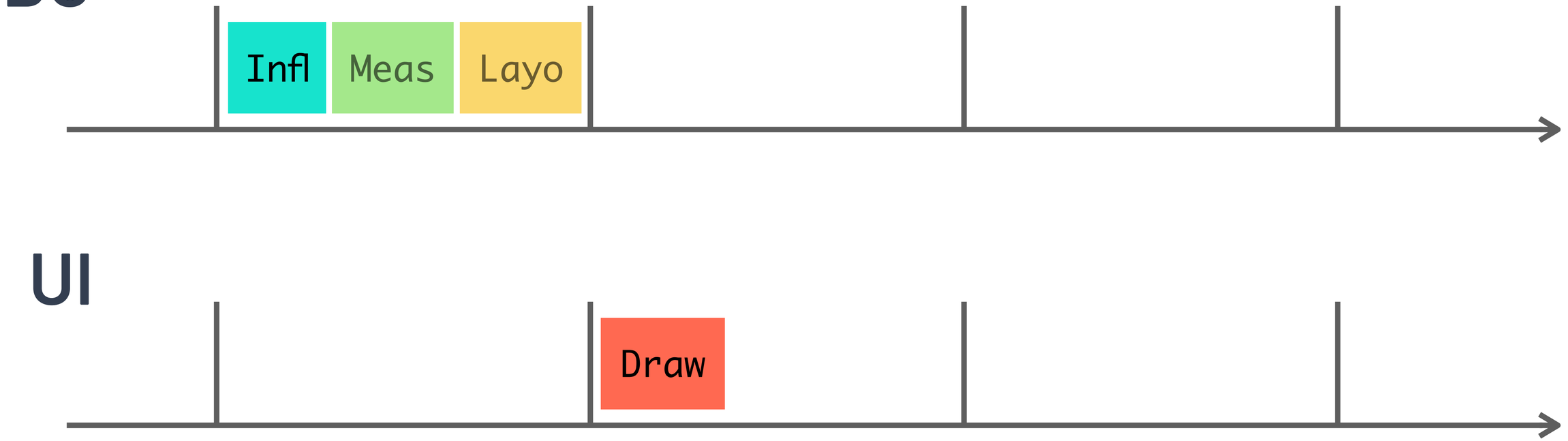
WHAT IS LITHO ABOUT



WHAT IS LITHO ABOUT



WHAT IS LITHO ABOUT

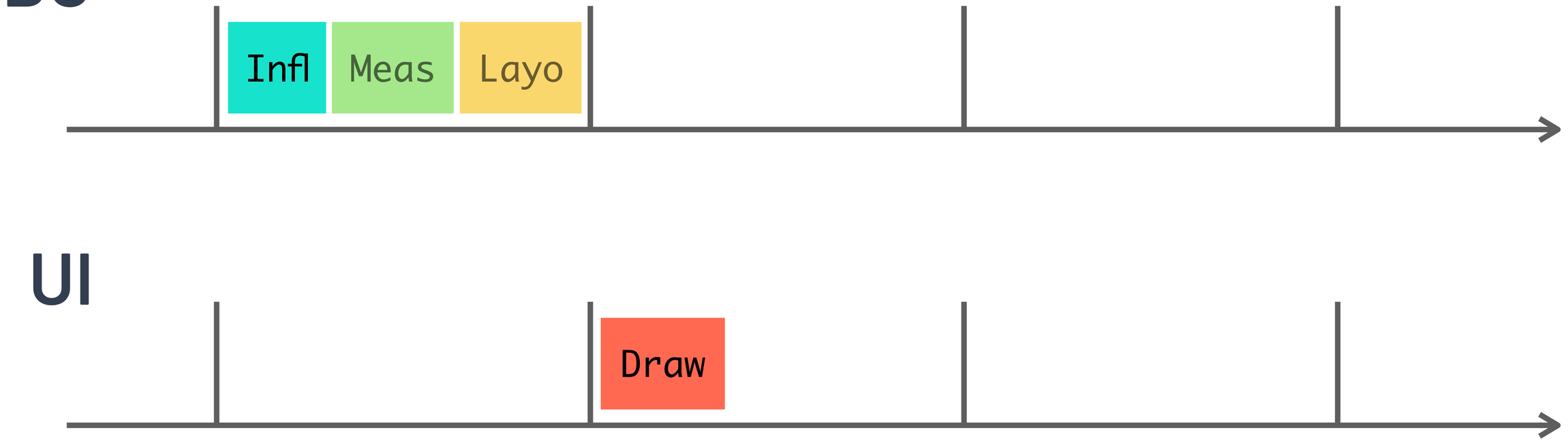


WHAT IS LITHO ABOUT

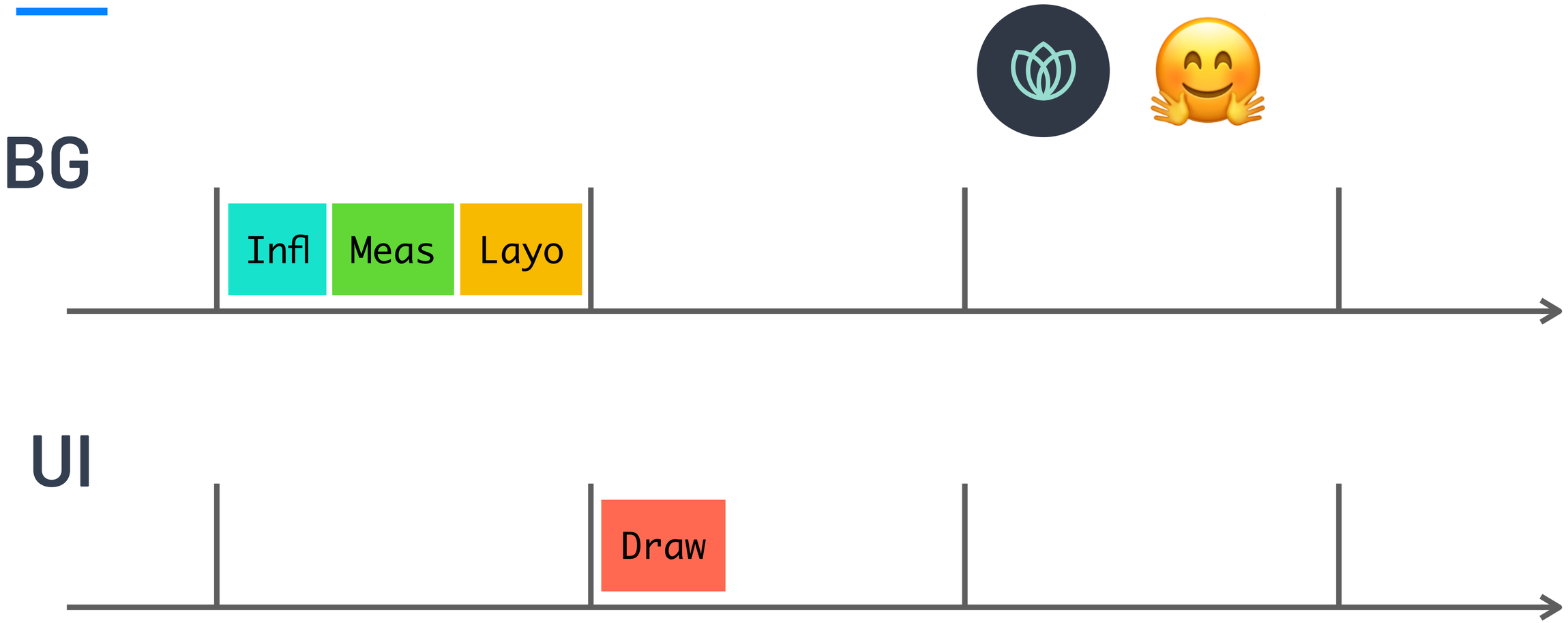
WHAT IS LITHO ABOUT

- Asynchronous layout

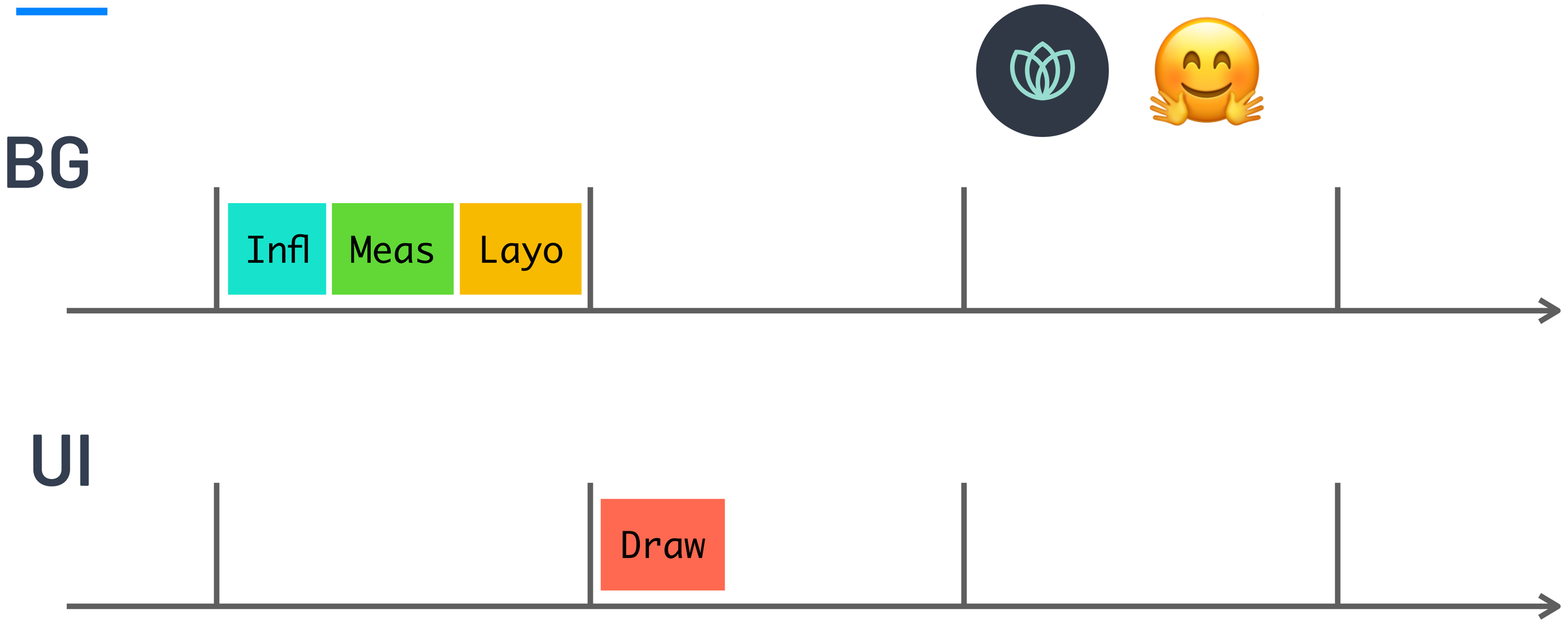
WHAT IS LITHO ABOUT



WHAT IS LITHO ABOUT



WHAT IS LITHO ABOUT



yogalayout.com

WHAT IS LITHO ABOUT

- Asynchronous layout

WHAT IS LITHO ABOUT

- ✦ Asynchronous layout
- ✦ Declarative API

UI = f(properties)

UI = f(props)

```
fun f(
    title: String,
    subtitle: String
): UI {
  return Column.create()
      .child(
          Text.create()
              .text(title))
      .child(
          Text.create()
              .text(subtitle))
      .build()
}
```

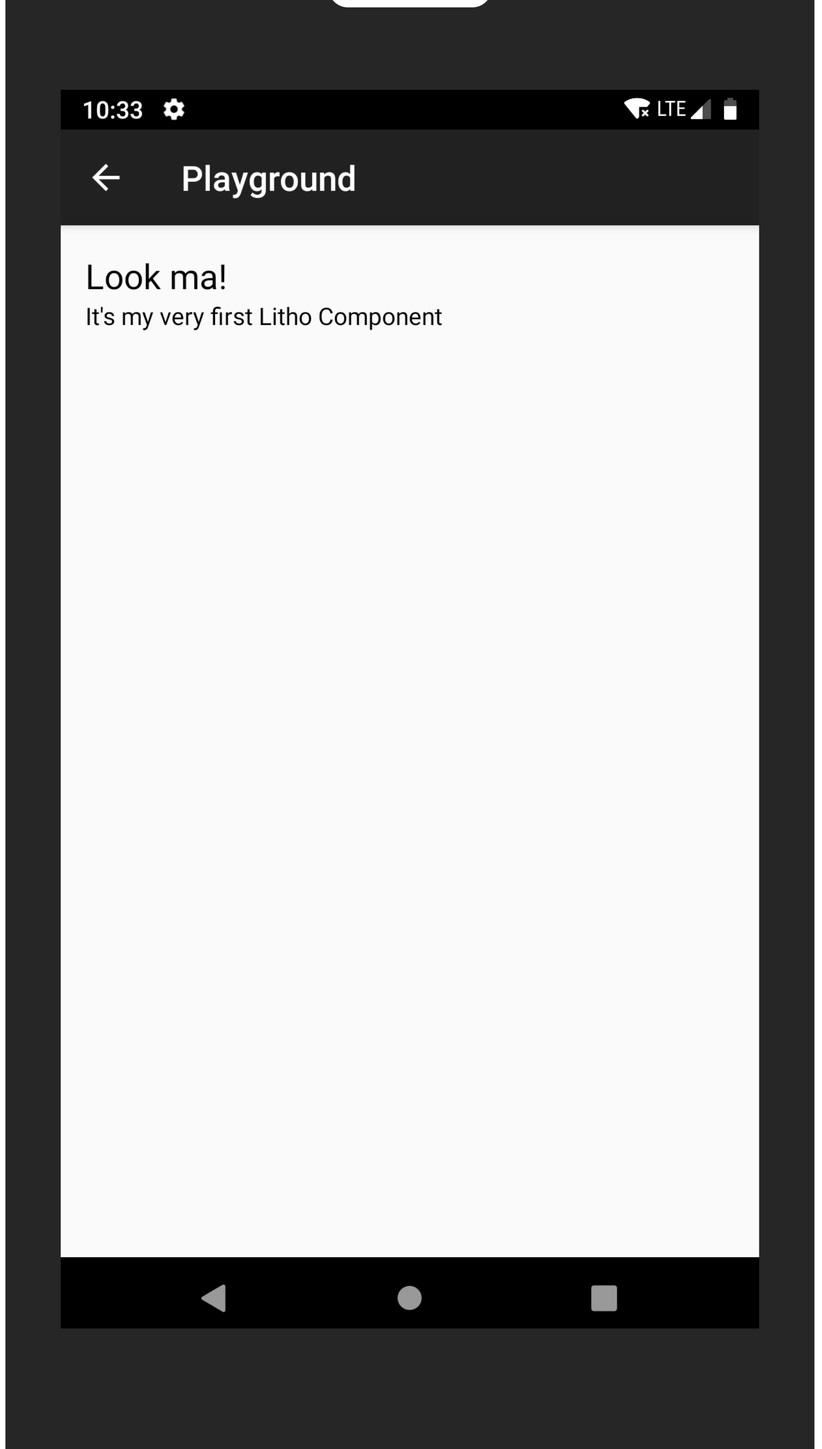
```
@OnCreateLayout
fun onCreateLayout(
    c: ComponentContext,
    @Prop title: String,
    @Prop subtitle: String
): Component {
  return Column.create(c)
      .child(
          Text.create(c)
              .text(title))
      .child(
          Text.create(c)
              .text(subtitle))
      .build()
}
```

```
@OnCreateLayout
fun onCreateLayout(
    c: ComponentContext,
    @Prop title: String,
    @Prop subtitle: String
): Component {
  return Column.create(c)
      .child(
          Text.create(c)
              .text(title))
      .child(
          Text.create(c)
              .text(subtitle))
      .build()
}
```

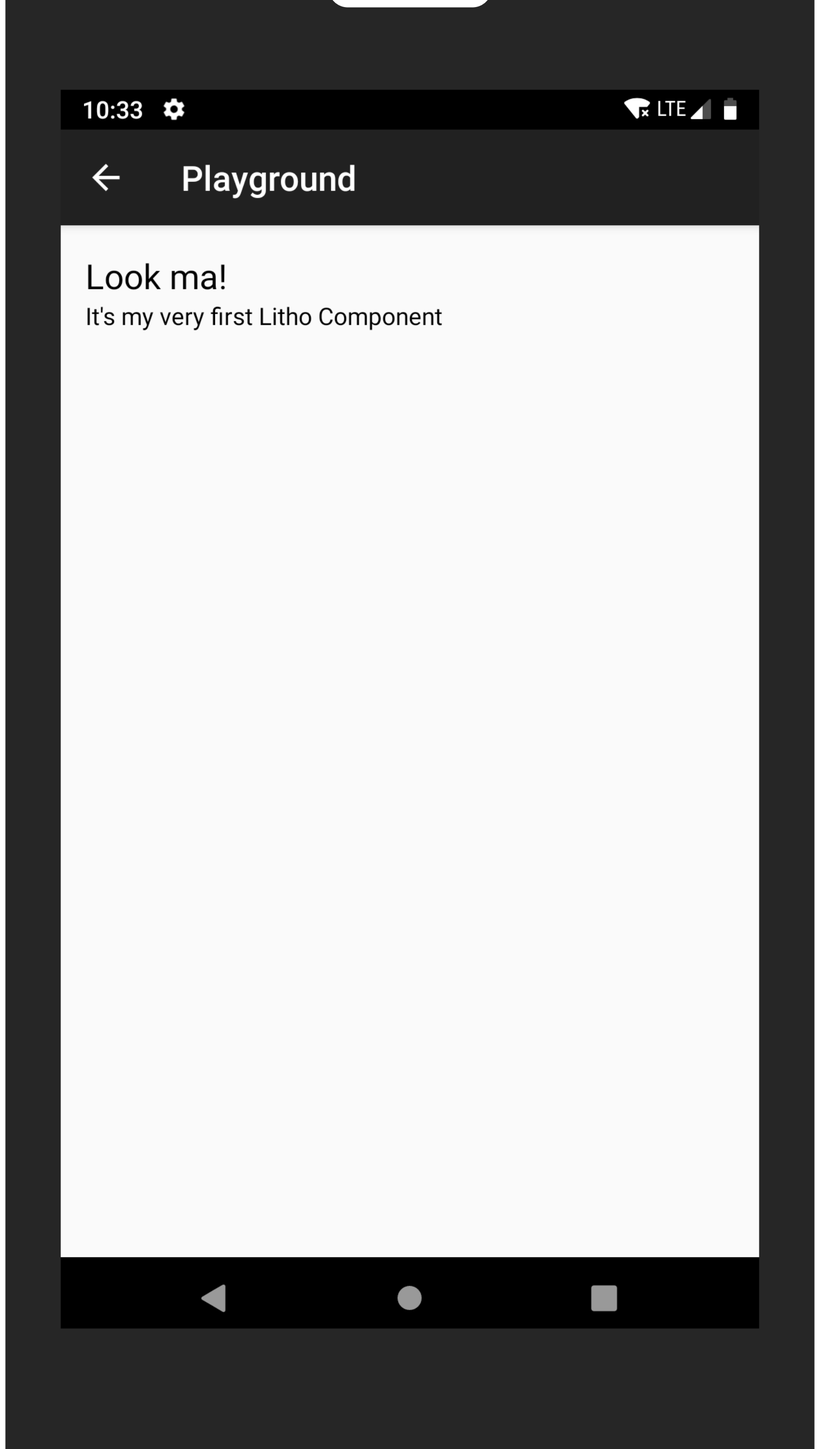
```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```



```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .paddingDip(ALL, 16f)
        .child(
            Text.create(c)
                .text(title)
                .textSizeSp(20f))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```



```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemWithImageSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop image: Drawable,
      @Prop title: String,
      @Prop subtitle: String): Component {
    return Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemWithImageSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop image: Drawable,
      @Prop title: String,
      @Prop subtitle: String): Component {
    return Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemWithImageSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop image: Drawable,
      @Prop title: String,
      @Prop subtitle: String): Component {
    return Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemWithImageSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop image: Drawable,
      @Prop title: String,
      @Prop subtitle: String): Component {
    return Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemWithImageSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop image: Drawable,
      @Prop title: String,
      @Prop subtitle: String): Component {
    return Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
class MainActivity : AppCompatActivity() {
  override fun onCreate(state: Bundle?) {
    super.onCreate(savedInstanceState)

    val c = ComponentContext(this)
    setContentView(LithoView.create(c,
        ListItem.create(c)
            .title("Title")
            .subtitle("Subtitle")
            .build()))
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
class MainActivity : AppCompatActivity() {
  override fun onCreate(state: Bundle?) {
    super.onCreate(savedInstanceState)

    val c = ComponentContext(this)
    setContentView(LithoView.create(c,
        ListItem.create(c)
            .title("Title")
            .subtitle("Subtitle")
            .build()))
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
class MainActivity : AppCompatActivity() {
  override fun onCreate(state: Bundle?) {
    super.onCreate(savedInstanceState)

    val c = ComponentContext(this)
    setContentView(LithoView.create(c,
        ListItem.create(c)
            .title("Title")
            .subtitle("Subtitle")
            .build()))
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
class MainActivity : AppCompatActivity() {
  override fun onCreate(state: Bundle?) {
    super.onCreate(savedInstanceState)

    val c = ComponentContext(this)
    setContentView(LithoView.create(c,
        ListItem.create(c)
            .title("Title")
            .subtitle("Subtitle")
            .build()))
  }
}
```

```
@LayoutSpec
object ListItemSpec {

  @OnCreateLayout
  fun onCreateLayout(
      c: ComponentContext,
      @Prop title: String,
      @Prop subtitle: String
  ): Component {
    return Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

```
class MainActivity : AppCompatActivity() {
  override fun onCreate(state: Bundle?) {
    super.onCreate(savedInstanceState)

    val c = ComponentContext(this)
    setContentView(LithoView.create(c,
        ListItem.create(c)
            .title("Title")
            .subtitle("Subtitle")
            .build()))
  }
}
```

HOW DOES IT ALL WORK?

HOW DOES IT ALL WORK

HOW DOES IT ALL WORK

1. Create an Internal Tree

HOW DOES IT ALL WORK

1. Create an Internal Tree

2. Create LayoutState

HOW DOES IT ALL WORK

1. Create an Internal Tree
2. Create LayoutState
3. Mount LayoutState

```
@LayoutSpec
object ListItemWithImageSpec {

    // ...

    Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemWithImageSpec {

    // ...

    Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```

```
@LayoutSpec
object ListItemWithImageSpec {

    // ...

    Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```

ListItemWithImage

```
@LayoutSpec
object ListItemWithImageSpec {

    // ...

    Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```

ListItemWithImage

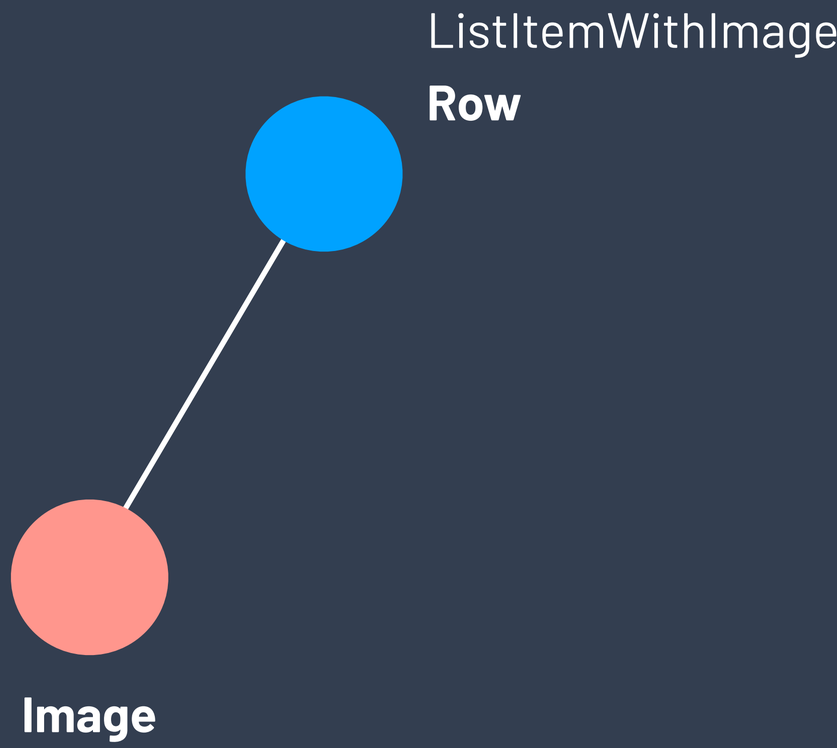
Row

```
@LayoutSpec
object ListItemWithImageSpec {

    // ...

    Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```

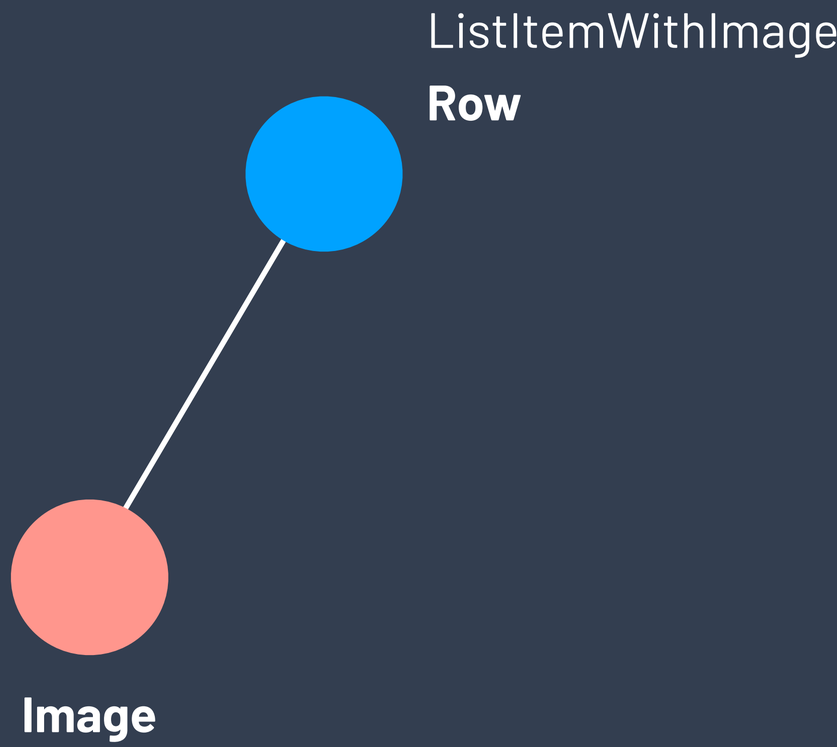
ListItemWithImage

Row

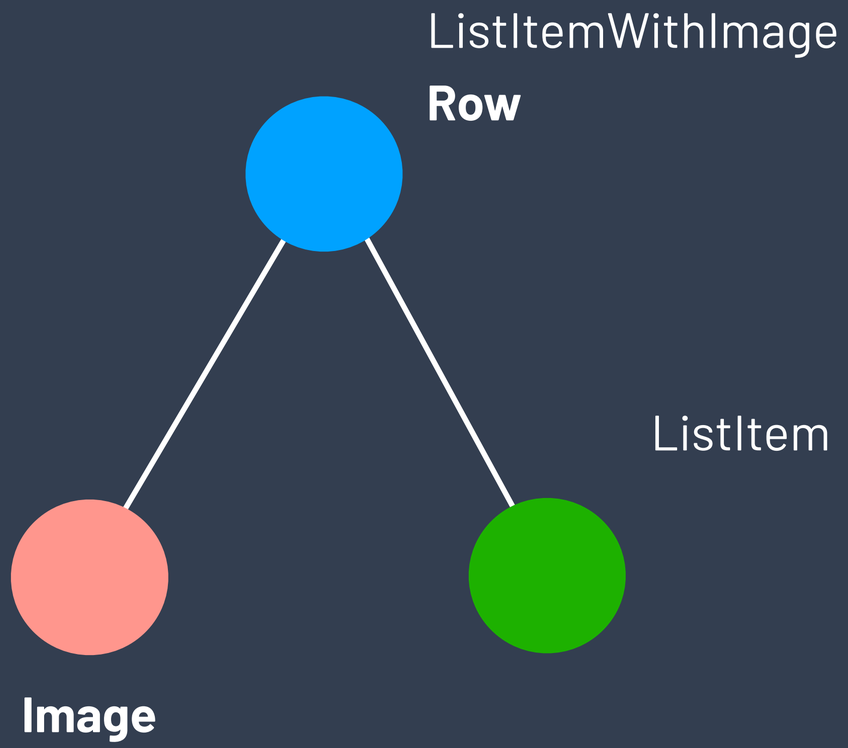
```
@LayoutSpec
object ListItemWithImageSpec {

    // ...

    Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```



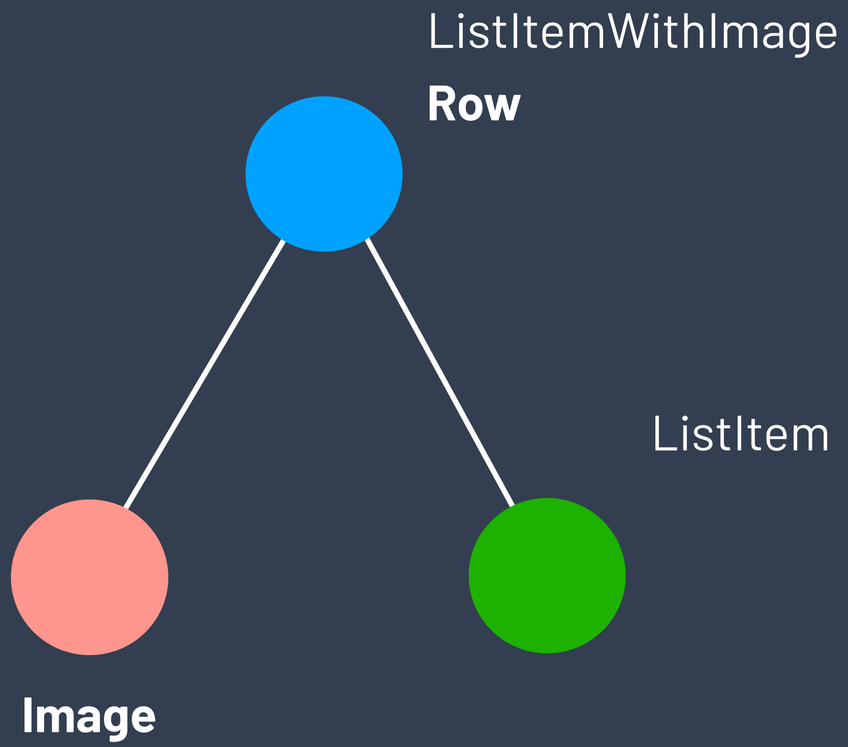
```
@LayoutSpec
object ListItemWithImageSpec {

    // ...

    Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```



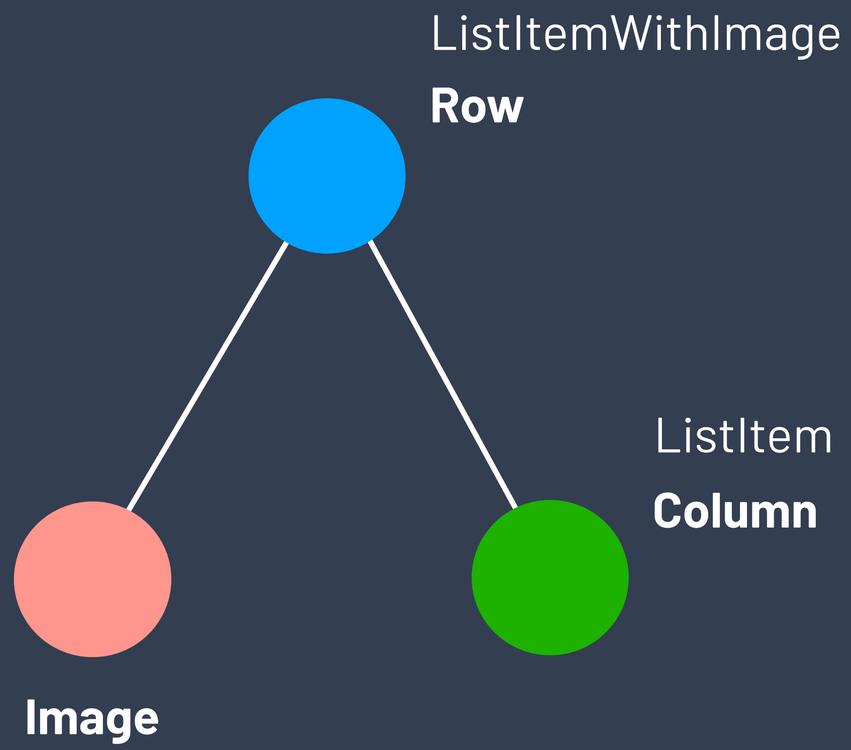
```
@LayoutSpec
object ListItemWithImageSpec {

    // ...

    Row.create(c)
        .child(
            Image.create(c)
                .drawable(image))
        .child(
            ListItem.create(c)
                .title(title)
                .subtitle(subtitle))
        .build()
  }
}
```



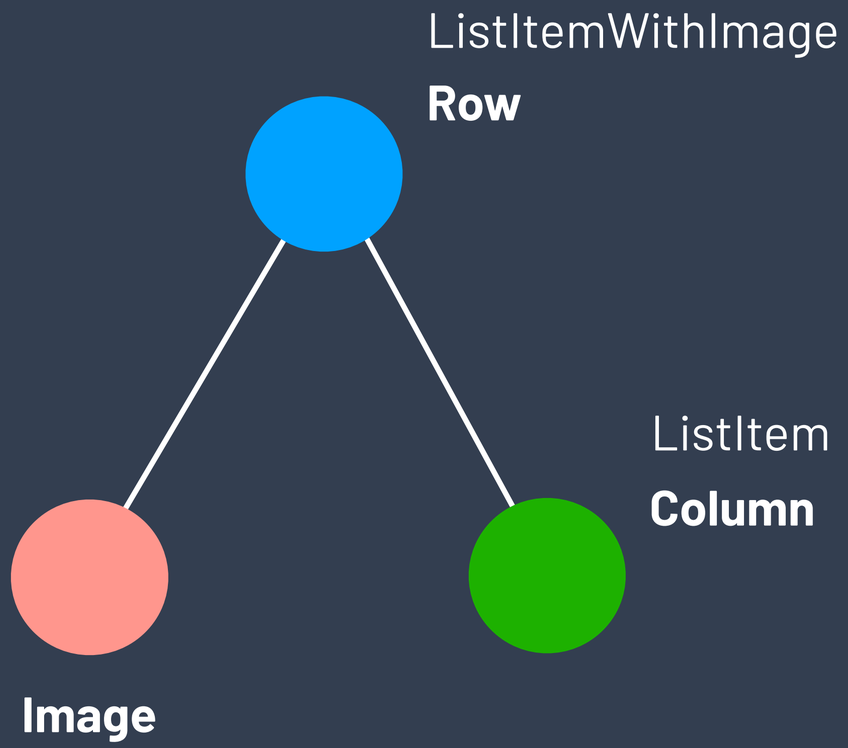
```
@LayoutSpec
object ListItemSpec {

    // ...

    Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```



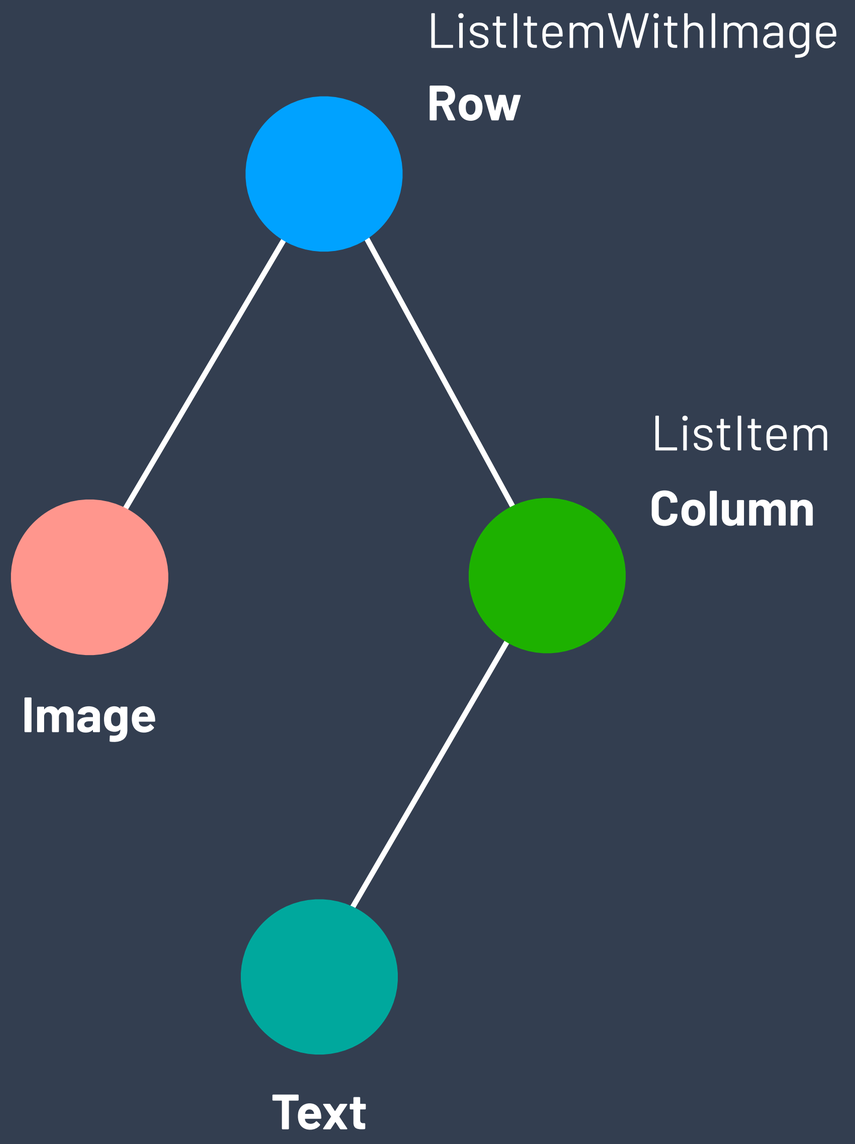
```
@LayoutSpec
object ListItemSpec {

    // ...

    Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```



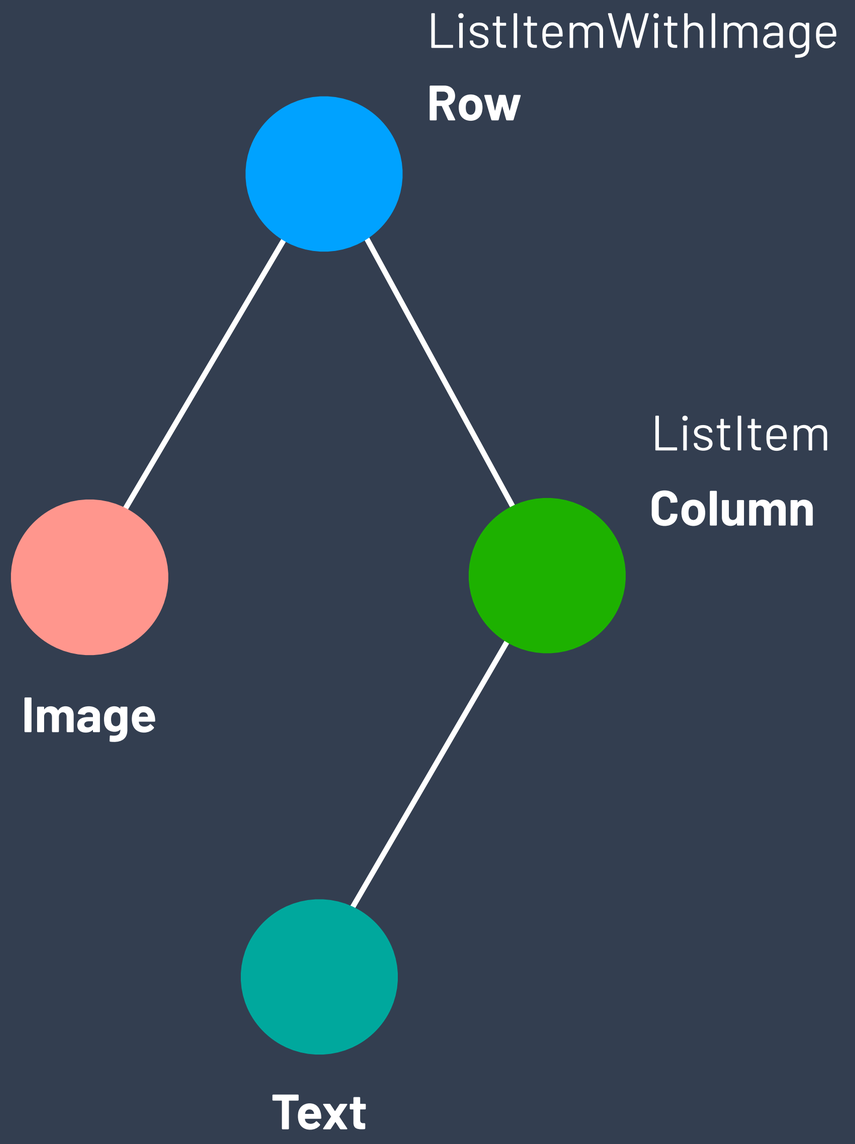
```
@LayoutSpec
object ListItemSpec {

    // ...

    Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```



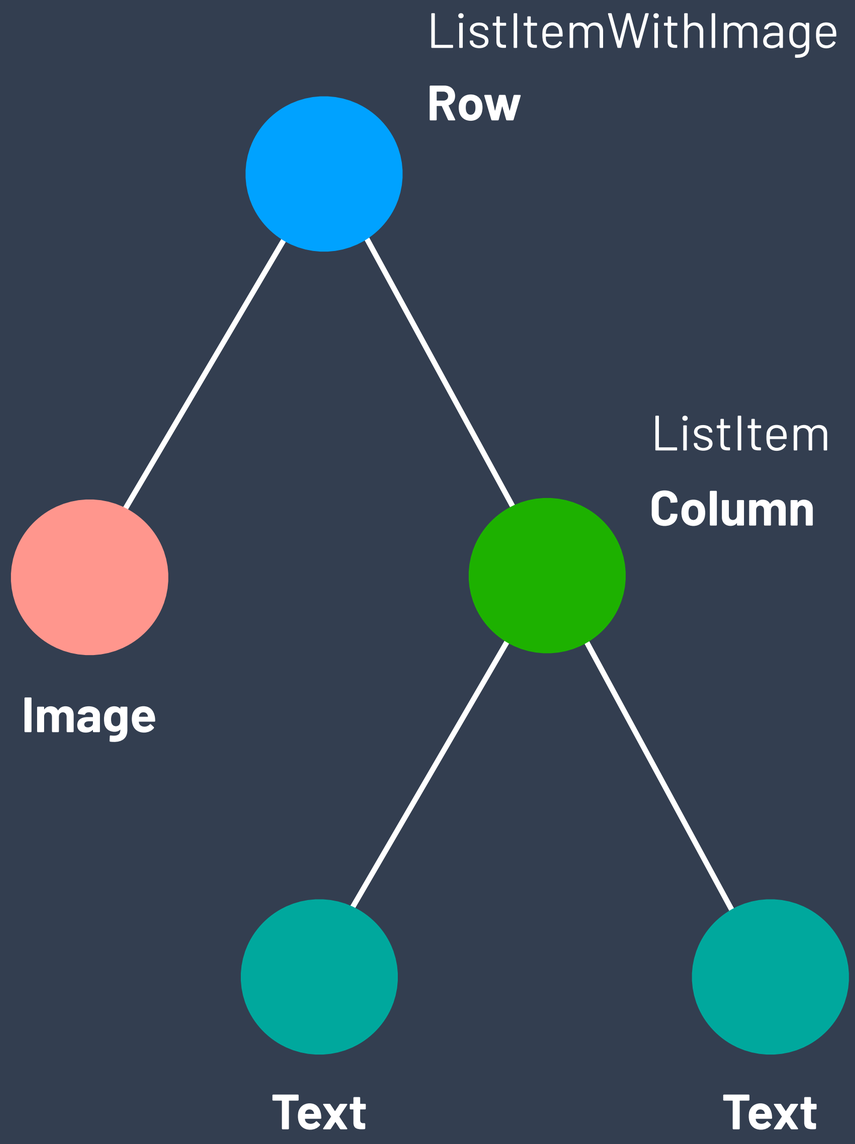
```
@LayoutSpec
object ListItemSpec {

    // ...

    Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```



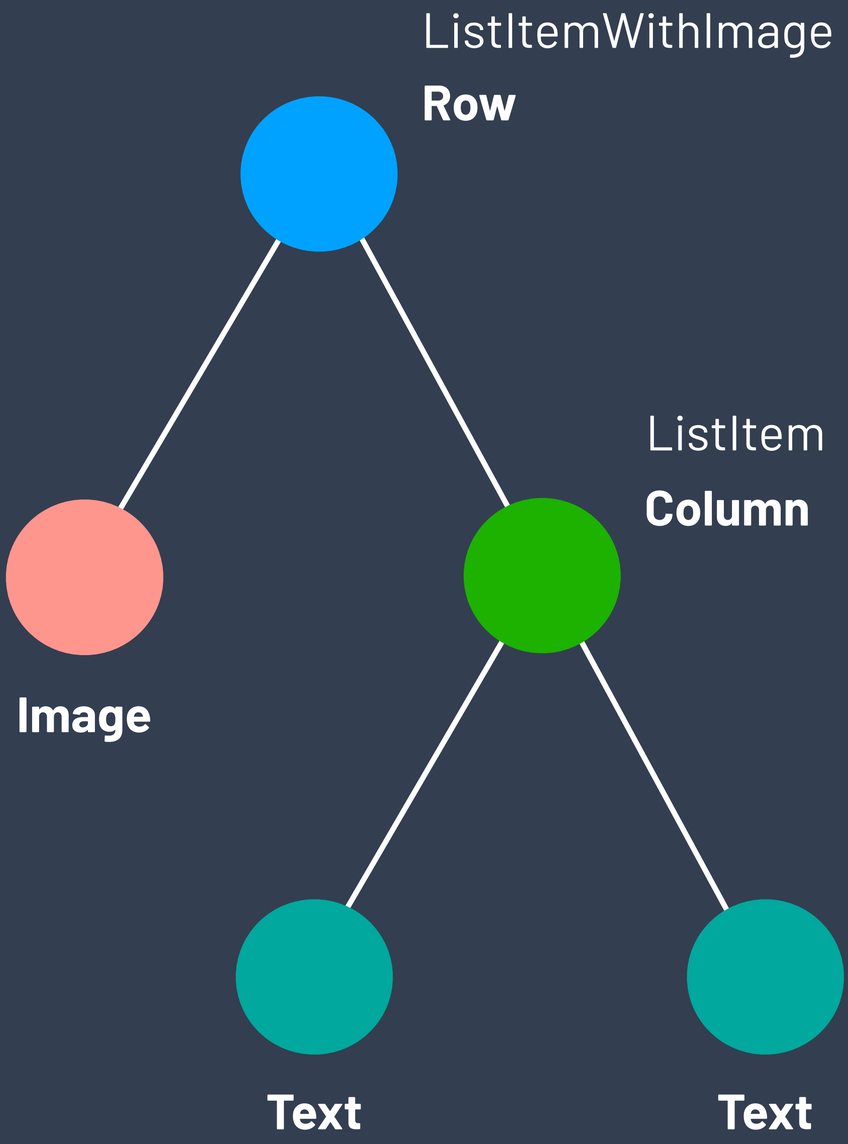
```
@LayoutSpec
object ListItemSpec {

    // ...

    Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```



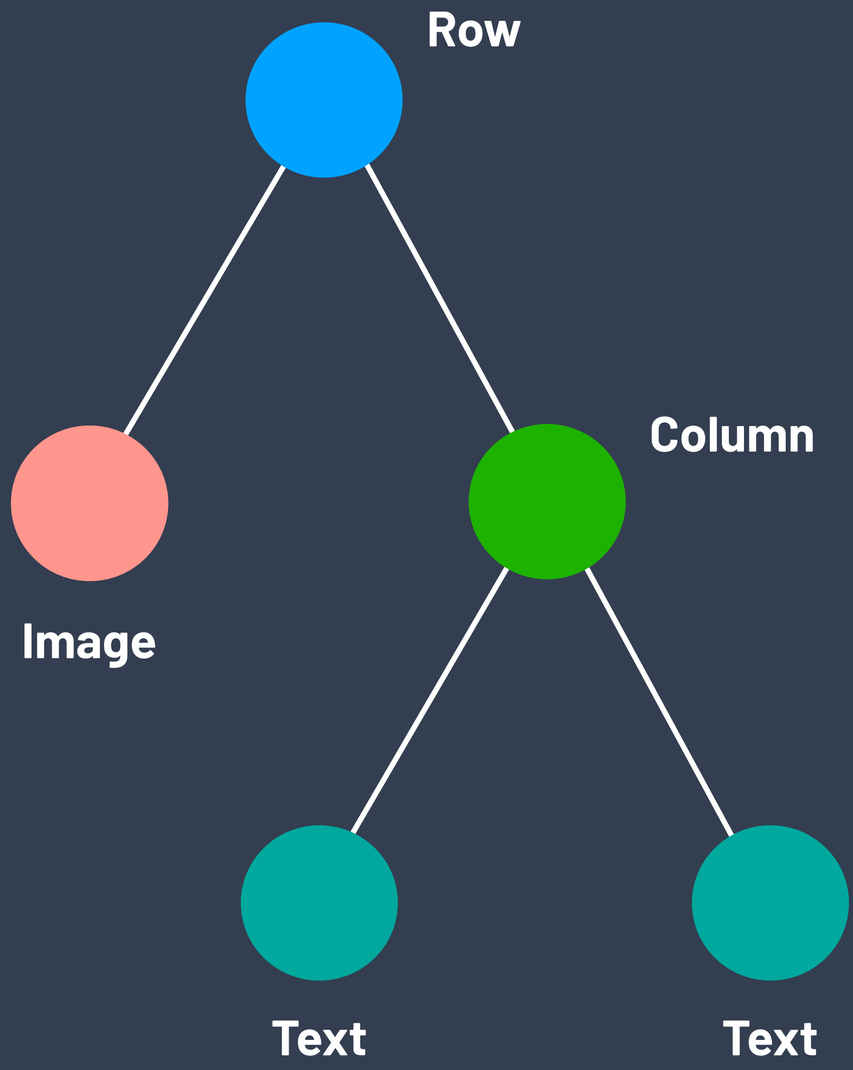
```
@LayoutSpec
object ListItemSpec {

    // ...

    Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```



```
@LayoutSpec
object ListItemSpec {

    // ...

    Column.create(c)
        .child(
            Text.create(c)
                .text(title))
        .child(
            Text.create(c)
                .text(subtitle))
        .build()
  }
}
```

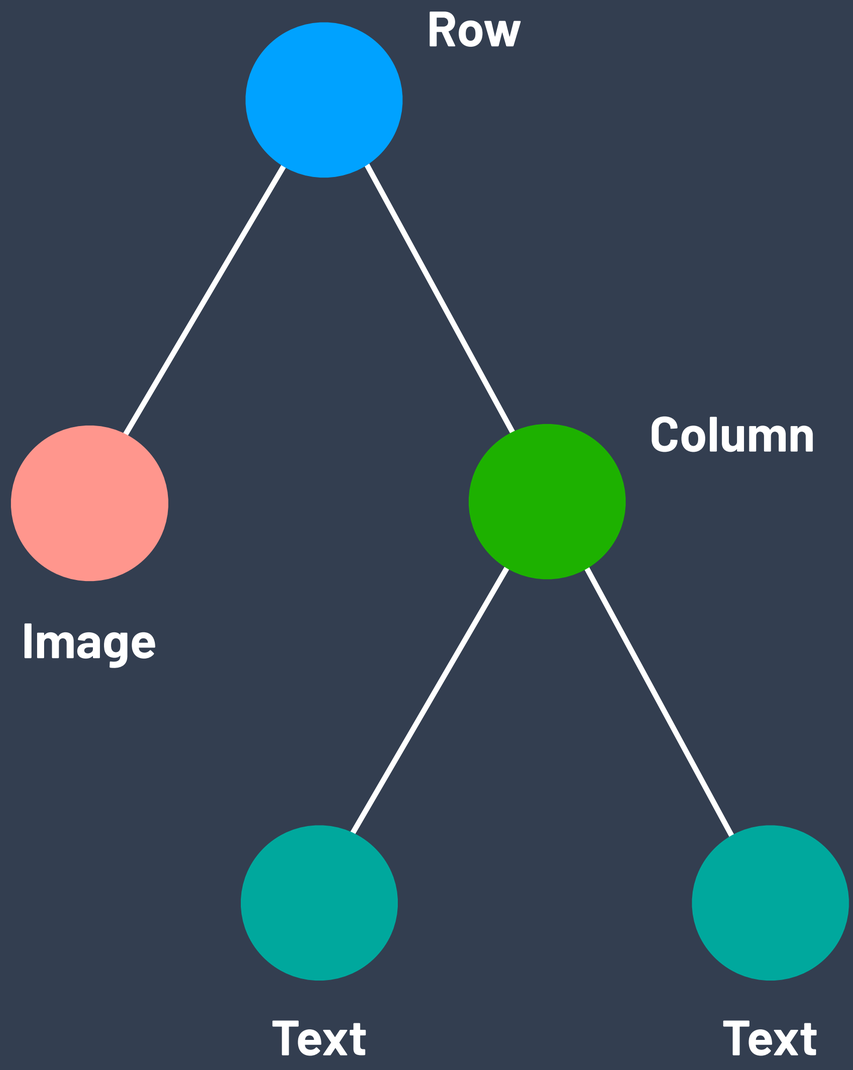


CREATE LAYOUTSTATE



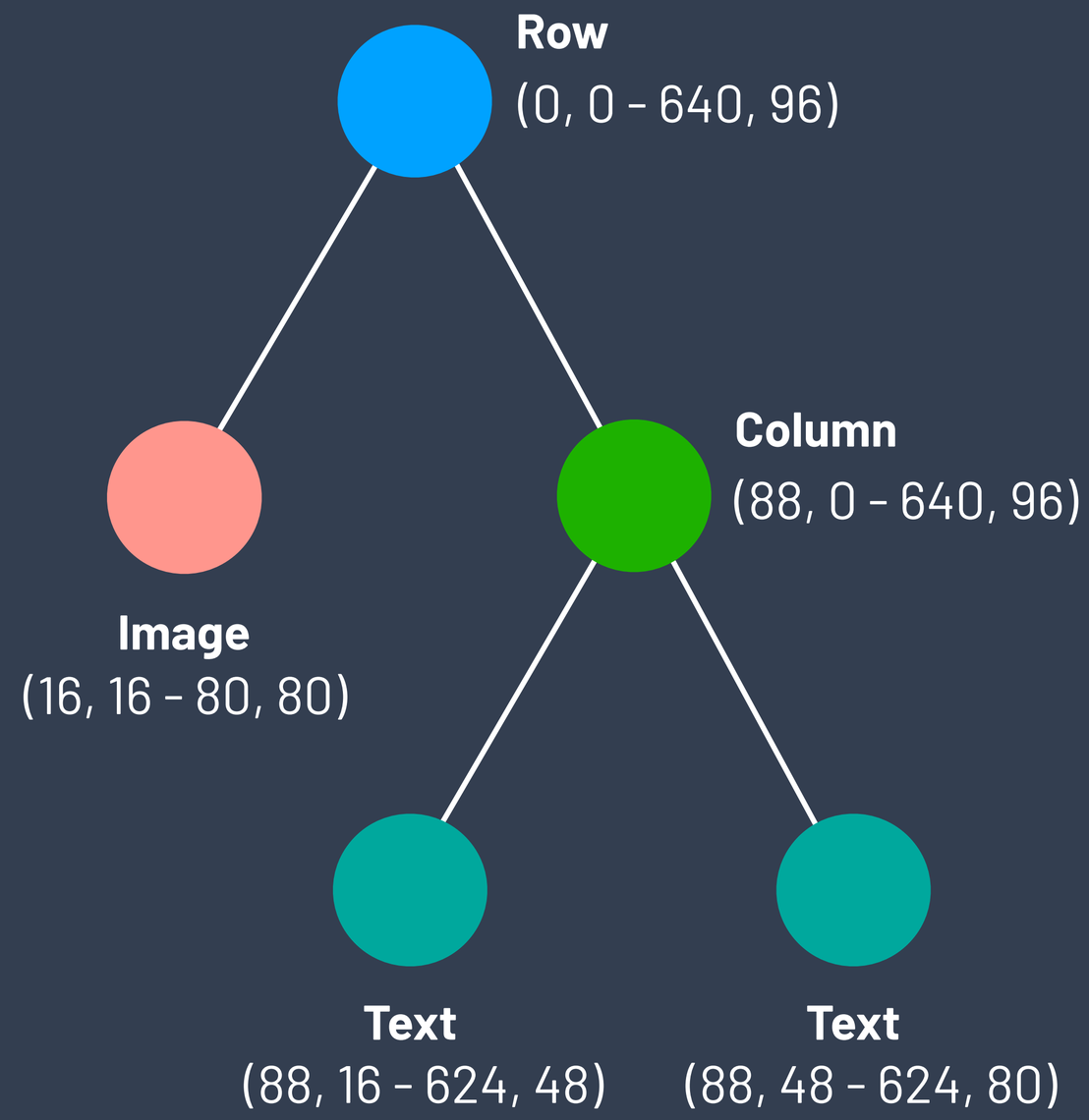
CREATE LAYOUTSTATE

Measure with Yoga



CREATE LAYOUTSTATE

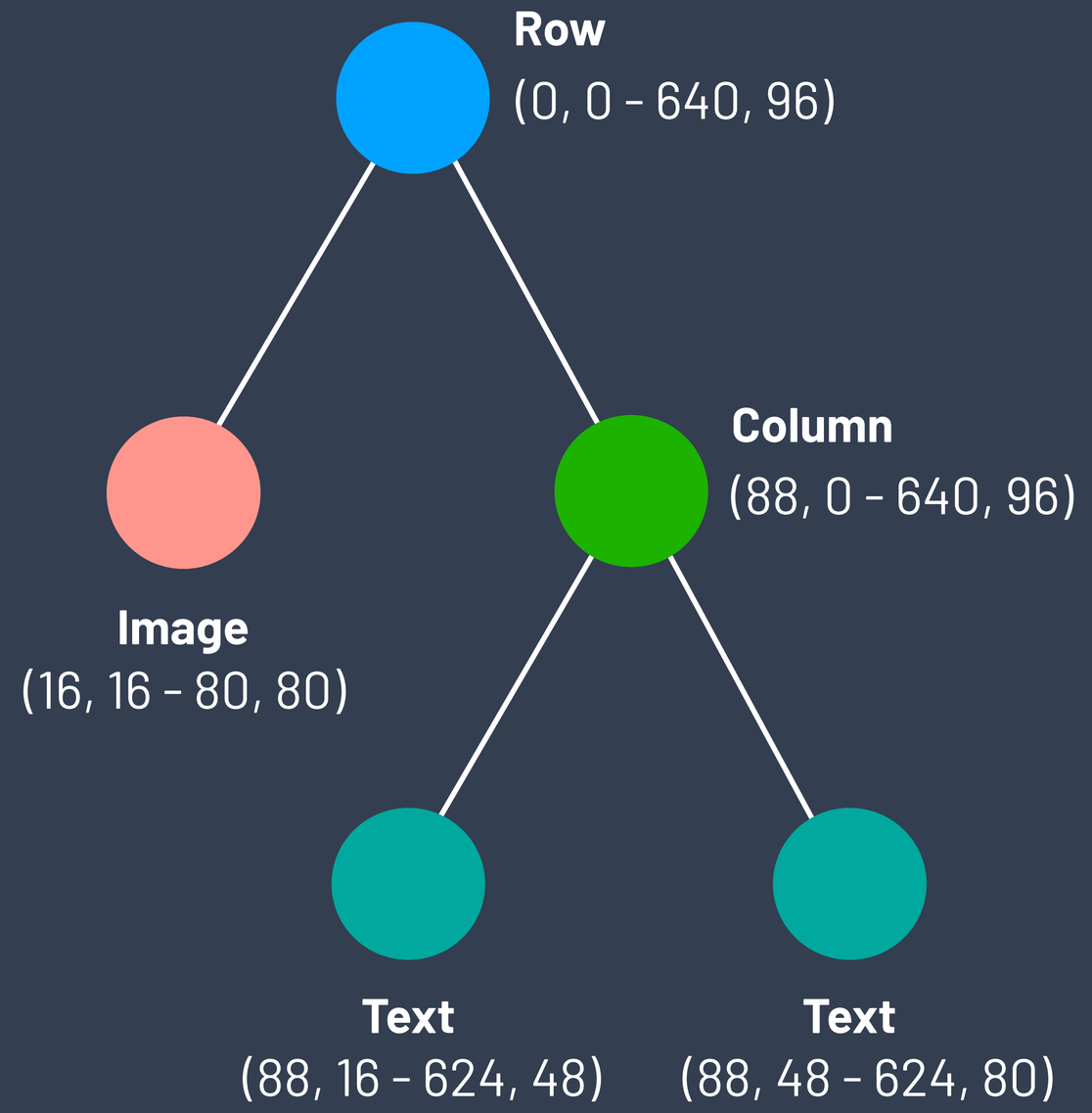
Measure with Yoga

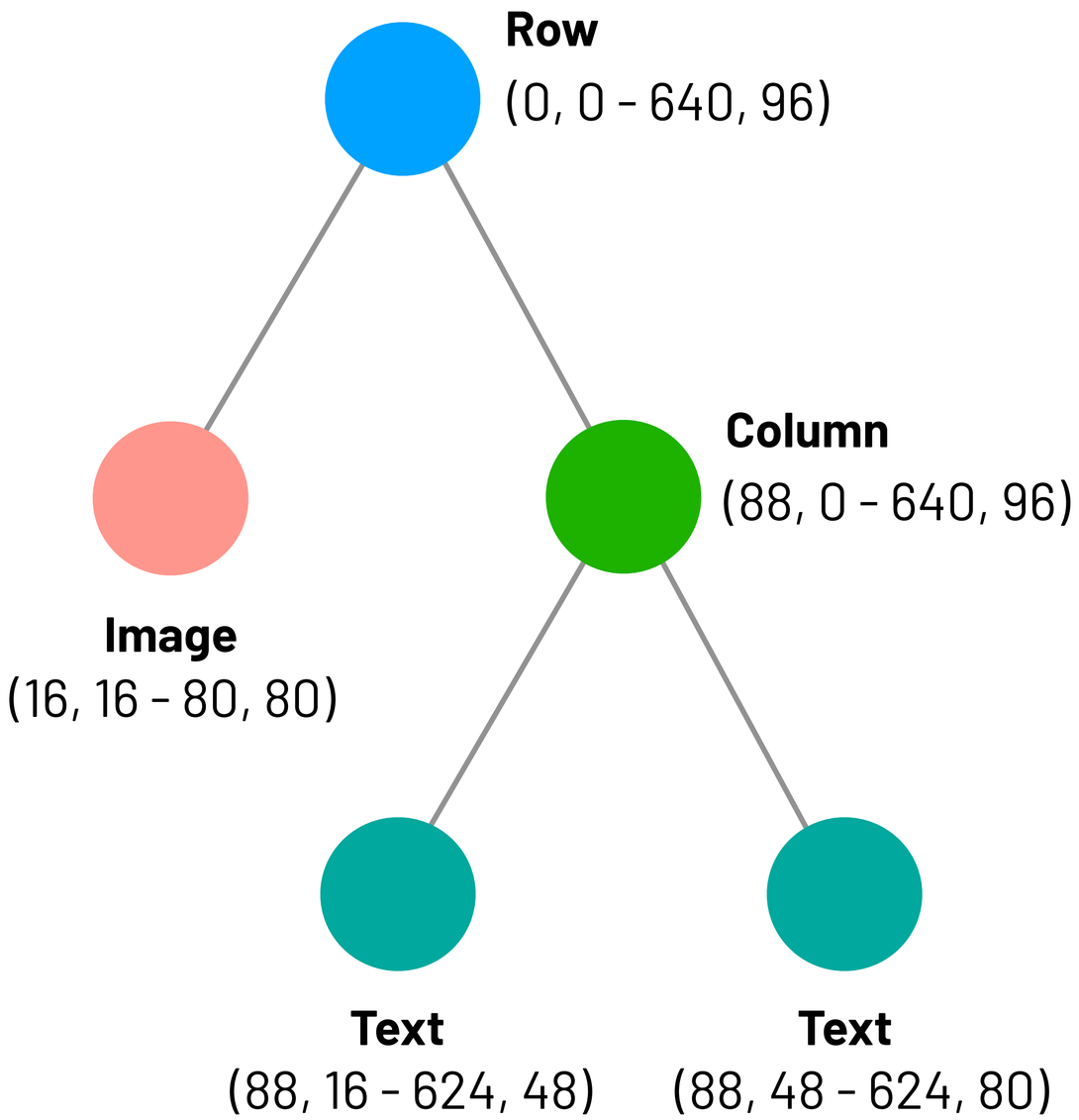


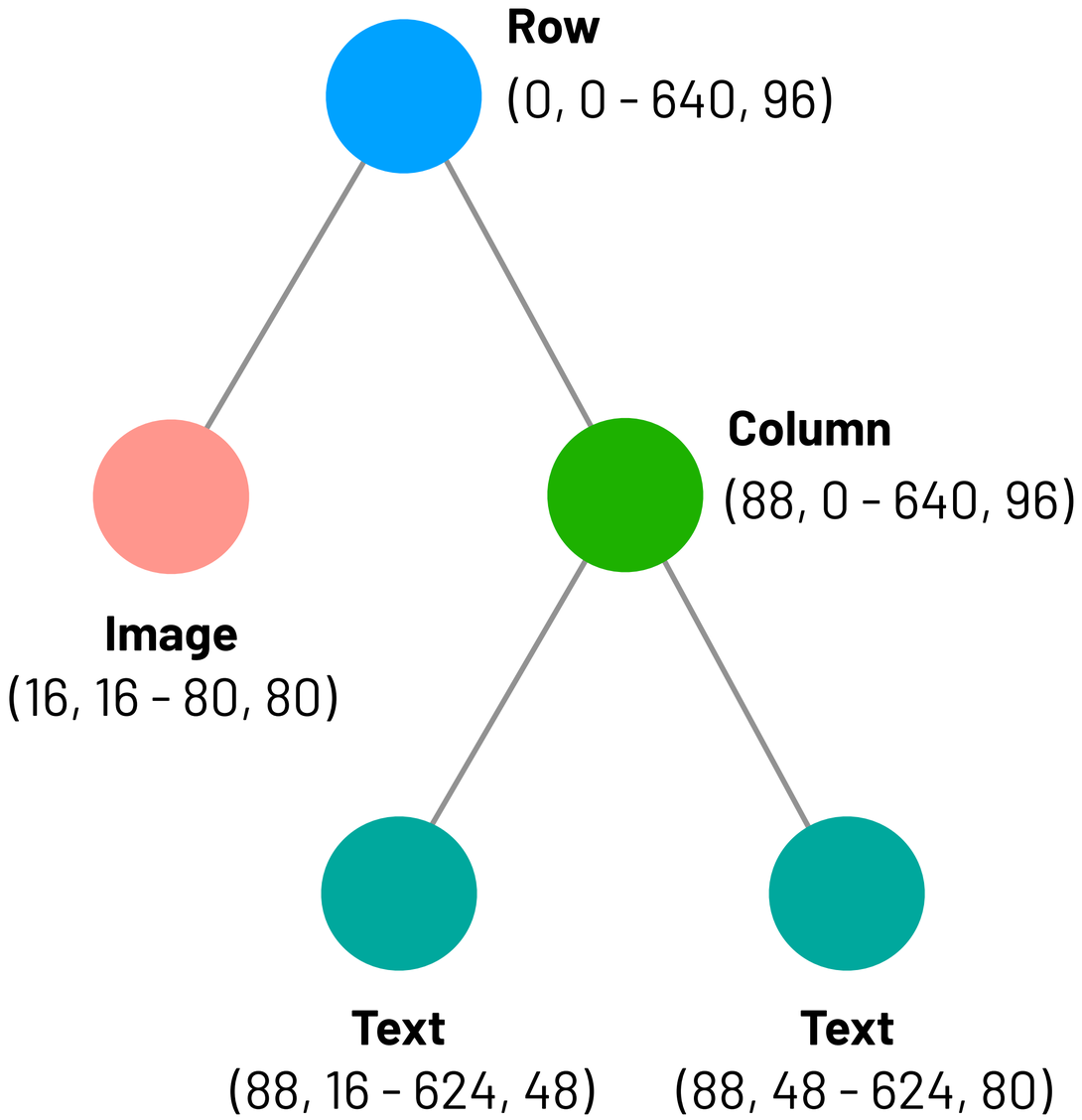
CREATE LAYOUTSTATE

Measure with Yoga

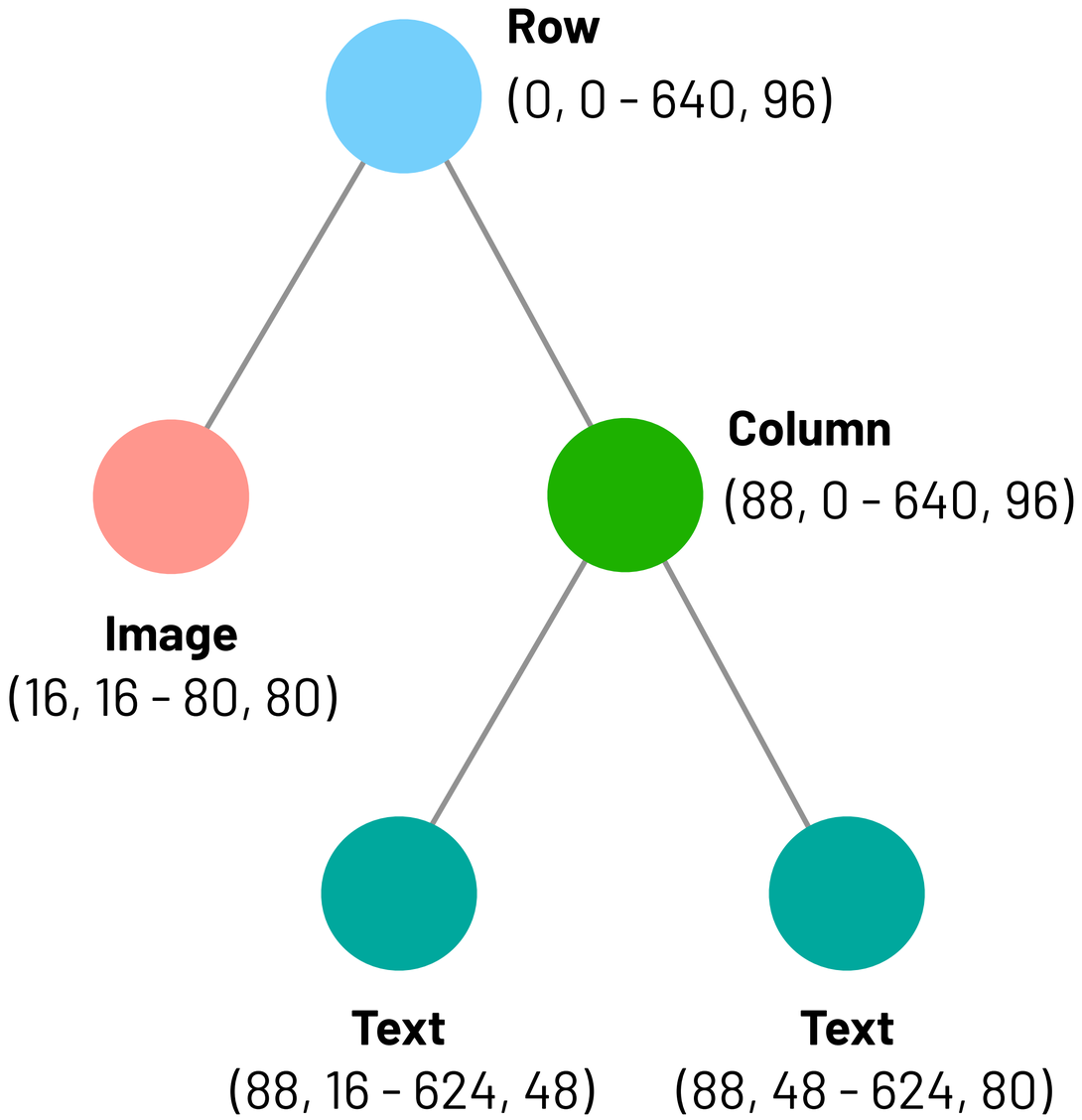
Collect a list of items to be drawn and their positions



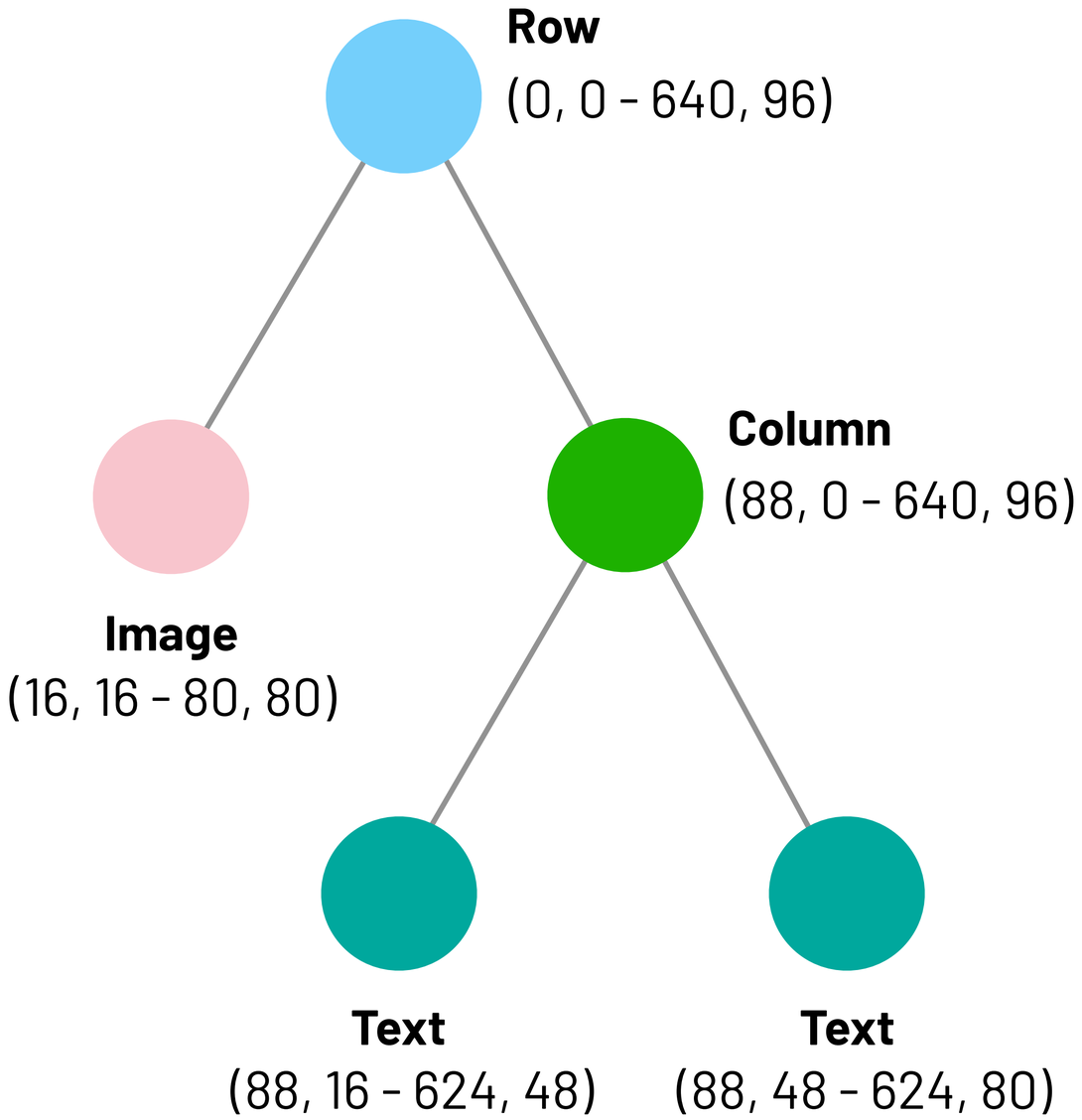

Row
(0, 0 - 640, 96)
Column
(88, 0 - 640, 96)
Image
(16, 16 - 80, 80)
Text
(88, 16 - 624, 48)
Text
(88, 48 - 624, 80)

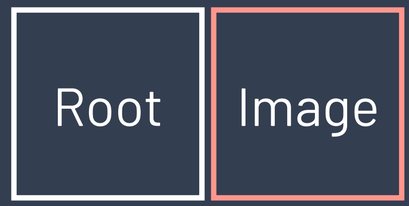

Row
(0, 0 - 640, 96)
Column
(88, 0 - 640, 96)
Image
(16, 16 - 80, 80)
Text
(88, 16 - 624, 48)
Text
(88, 48 - 624, 80)

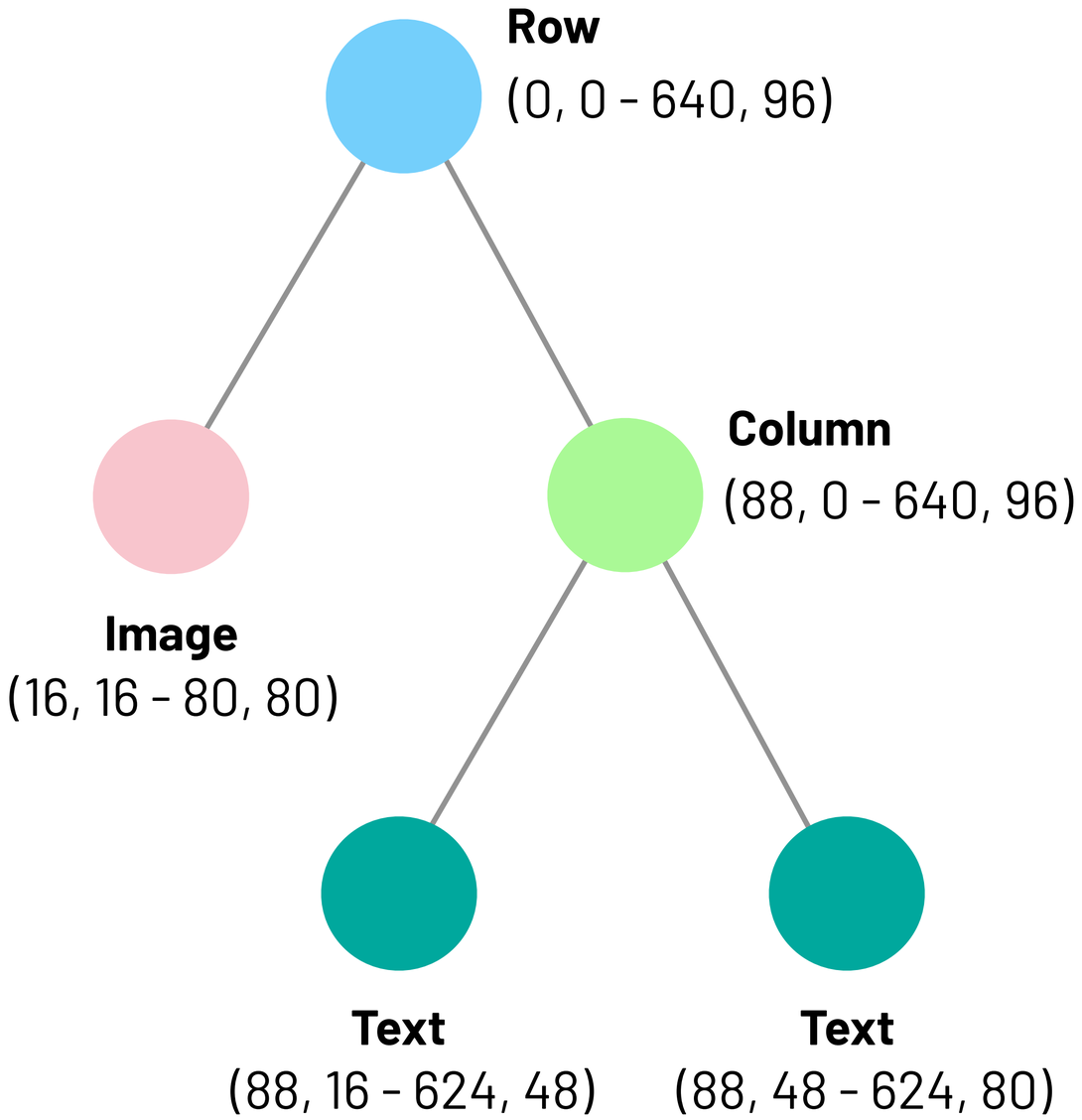
Root

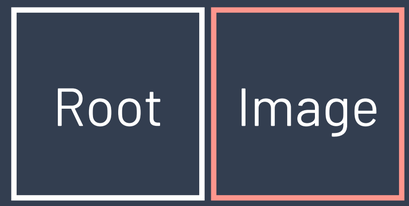

Row
(0, 0 - 640, 96)
Column
(88, 0 - 640, 96)
Image
(16, 16 - 80, 80)
Text
(88, 16 - 624, 48)
Text
(88, 48 - 624, 80)

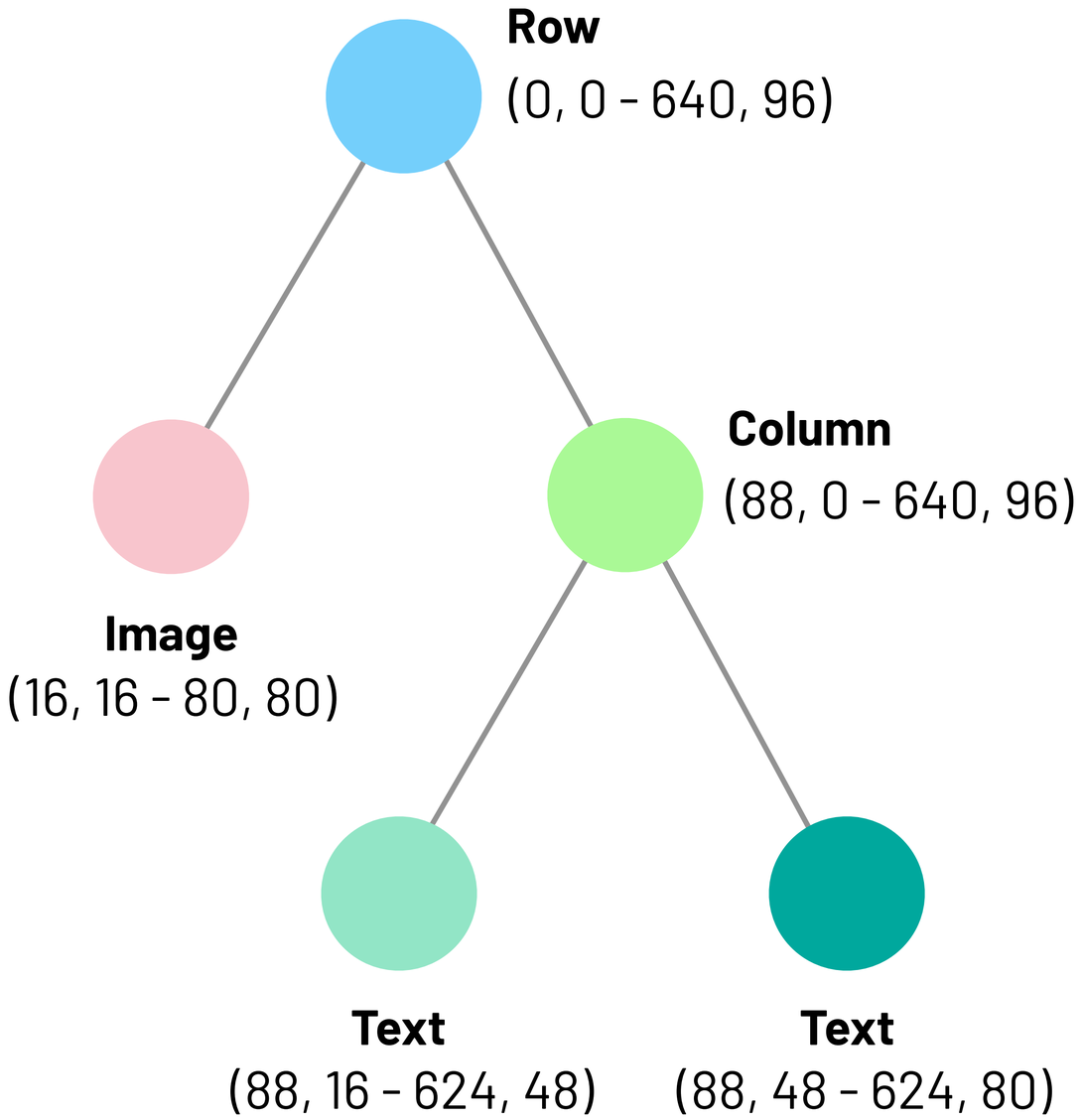
Root


Row
(0, 0 - 640, 96)
Column
(88, 0 - 640, 96)
Image
(16, 16 - 80, 80)
Text
(88, 16 - 624, 48)
Text
(88, 48 - 624, 80)

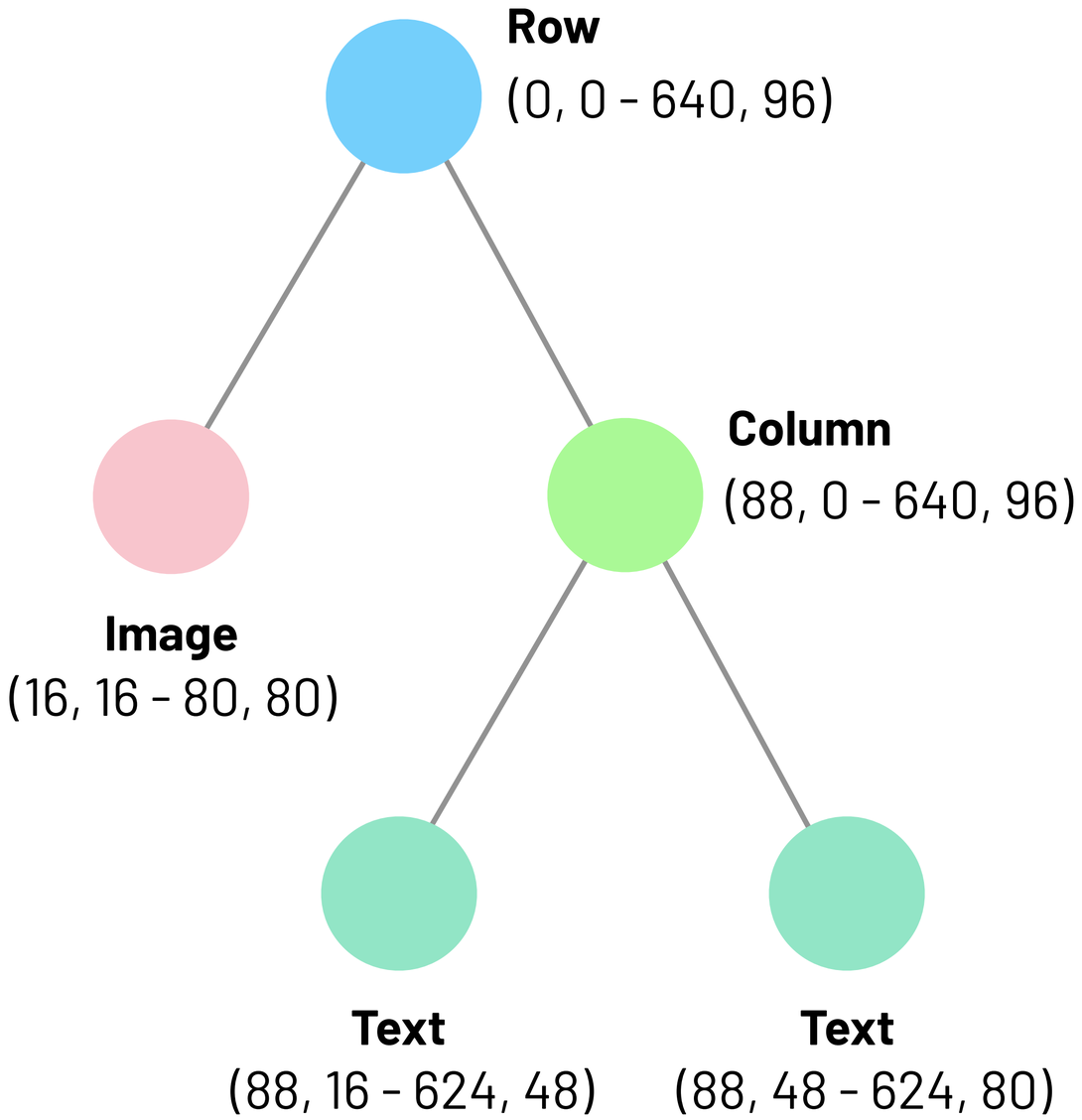

Root
Image


Row
(0, 0 - 640, 96)
Column
(88, 0 - 640, 96)
Image
(16, 16 - 80, 80)
Text
(88, 16 - 624, 48)
Text
(88, 48 - 624, 80)

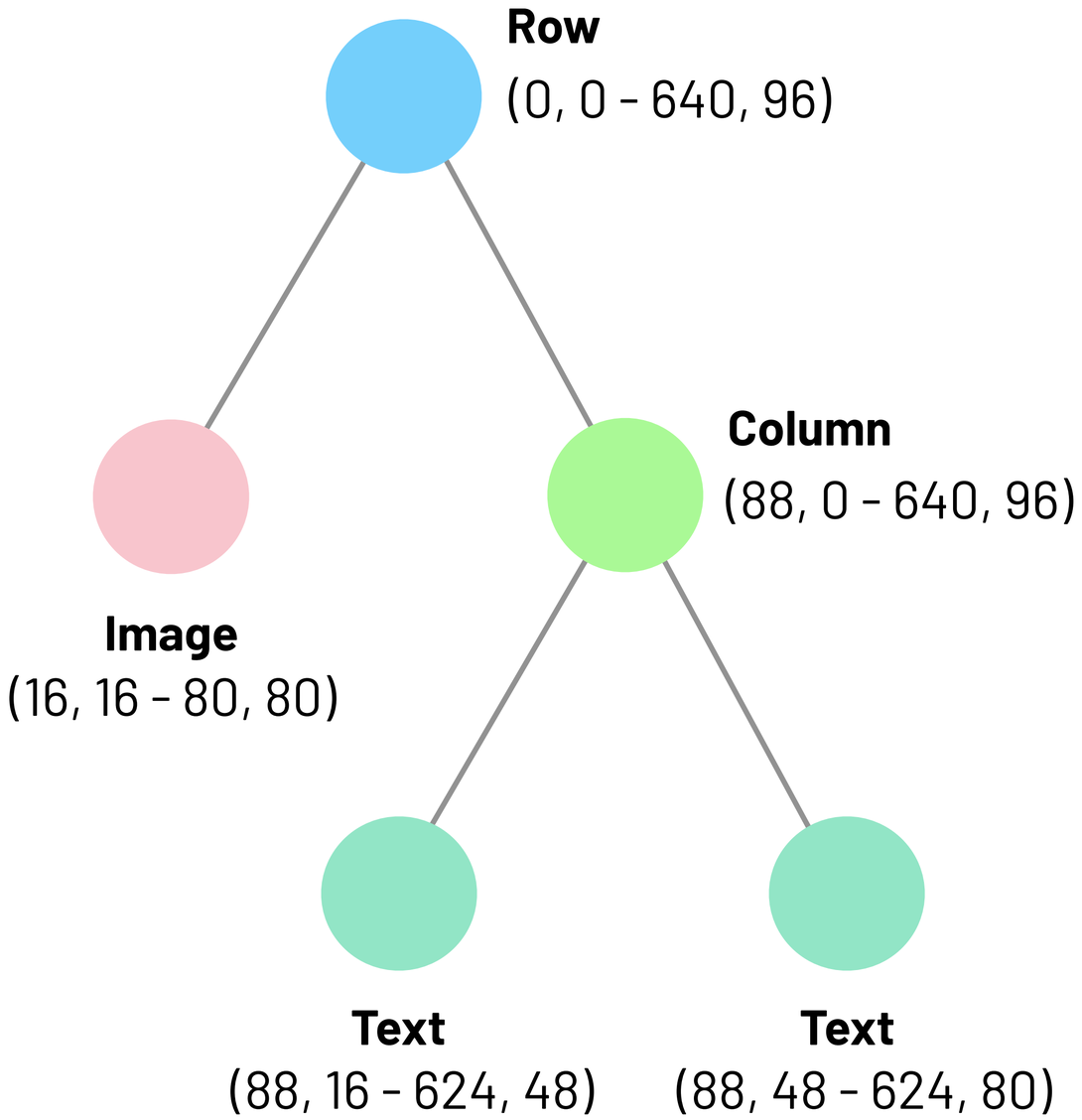

Root
Image

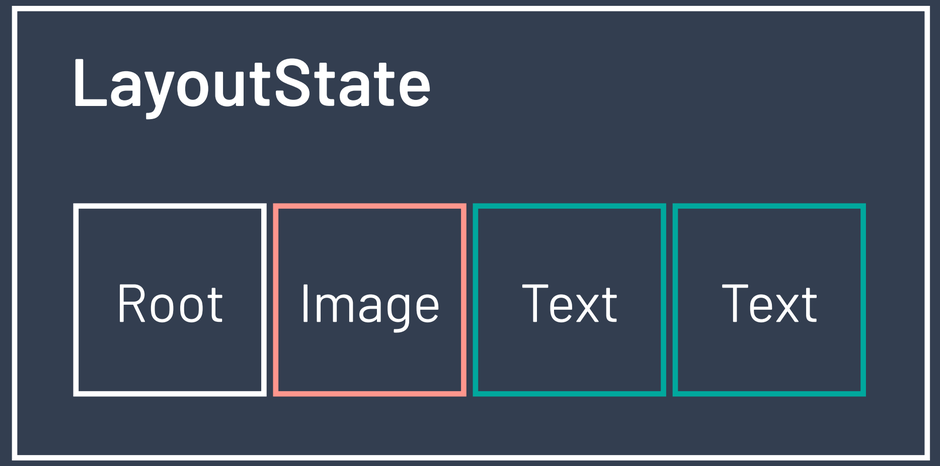

Row
(0, 0 - 640, 96)
Column
(88, 0 - 640, 96)
Image
(16, 16 - 80, 80)
Text
(88, 16 - 624, 48)
Text
(88, 48 - 624, 80)

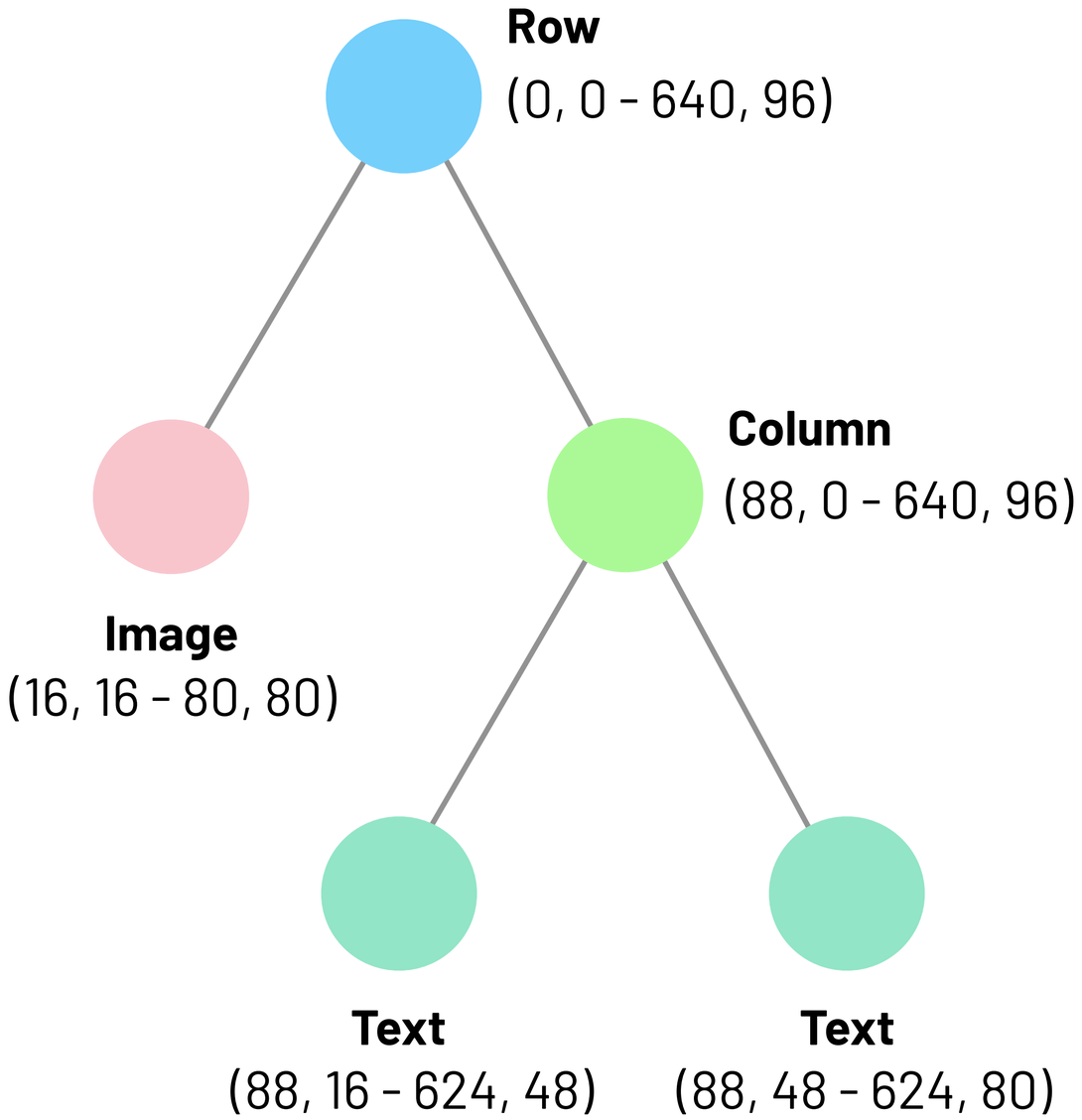

Root
Image
Text

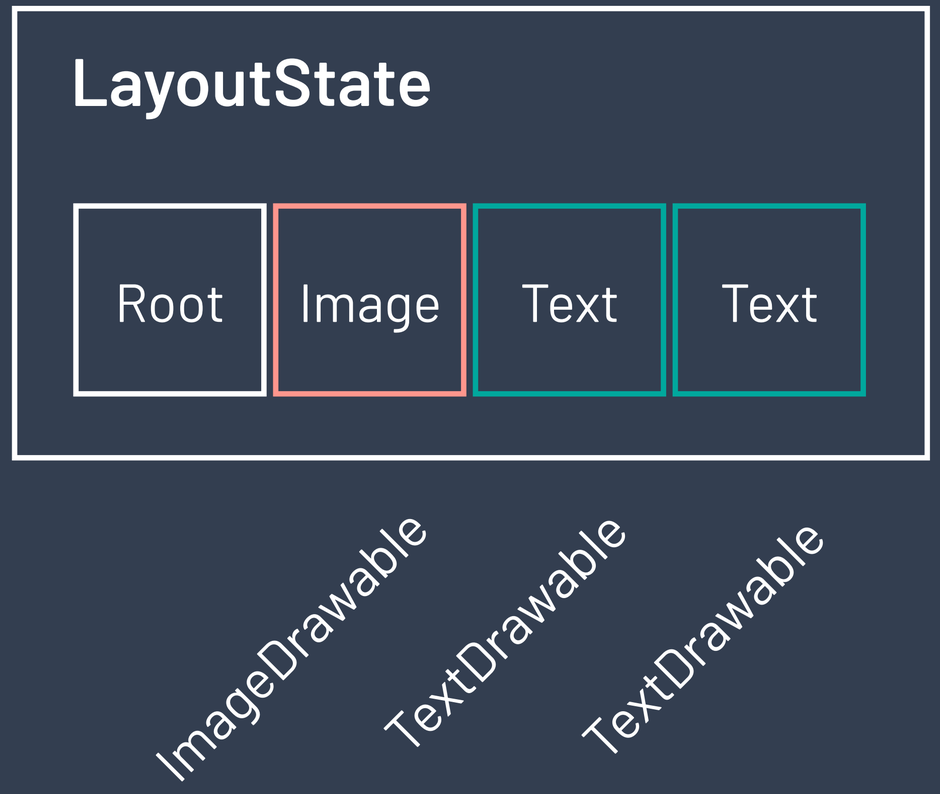

Row
(0, 0 - 640, 96)
Column
(88, 0 - 640, 96)
Image
(16, 16 - 80, 80)
Text
(88, 16 - 624, 48)
Text
(88, 48 - 624, 80)


Root
Image
Text
Text

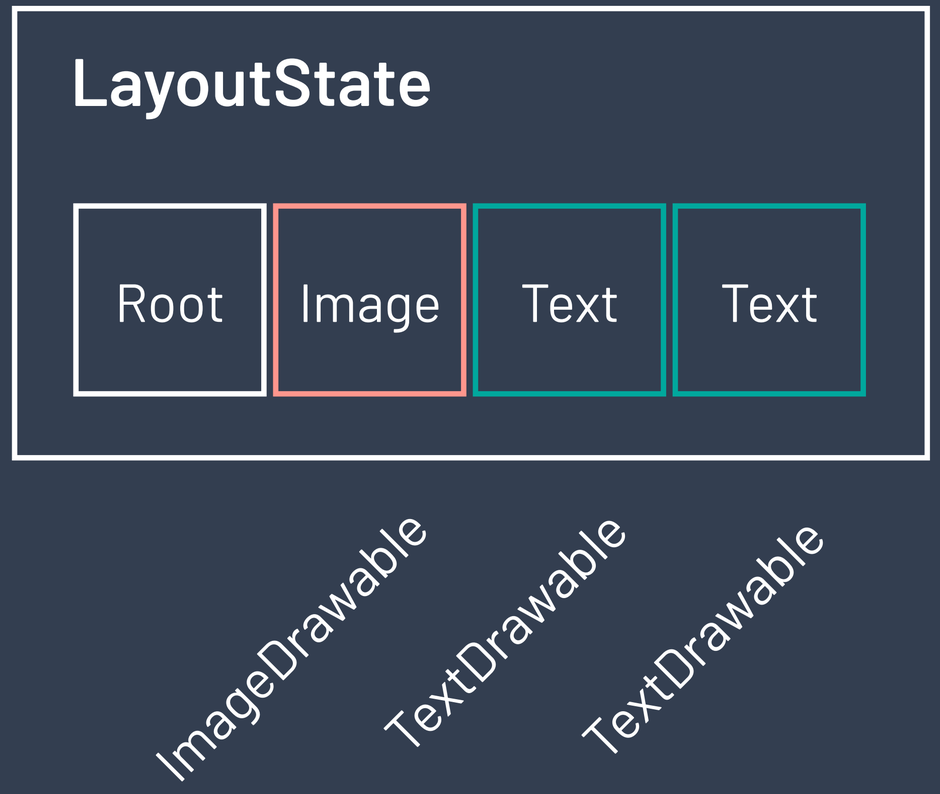

Row
(0, 0 - 640, 96)
Column
(88, 0 - 640, 96)
Image
(16, 16 - 80, 80)
Text
(88, 16 - 624, 48)
Text
(88, 48 - 624, 80)


LayoutState
Root
Image
Text
Text


Row
(0, 0 - 640, 96)
Column
(88, 0 - 640, 96)
Image
(16, 16 - 80, 80)
Text
(88, 16 - 624, 48)
Text
(88, 48 - 624, 80)

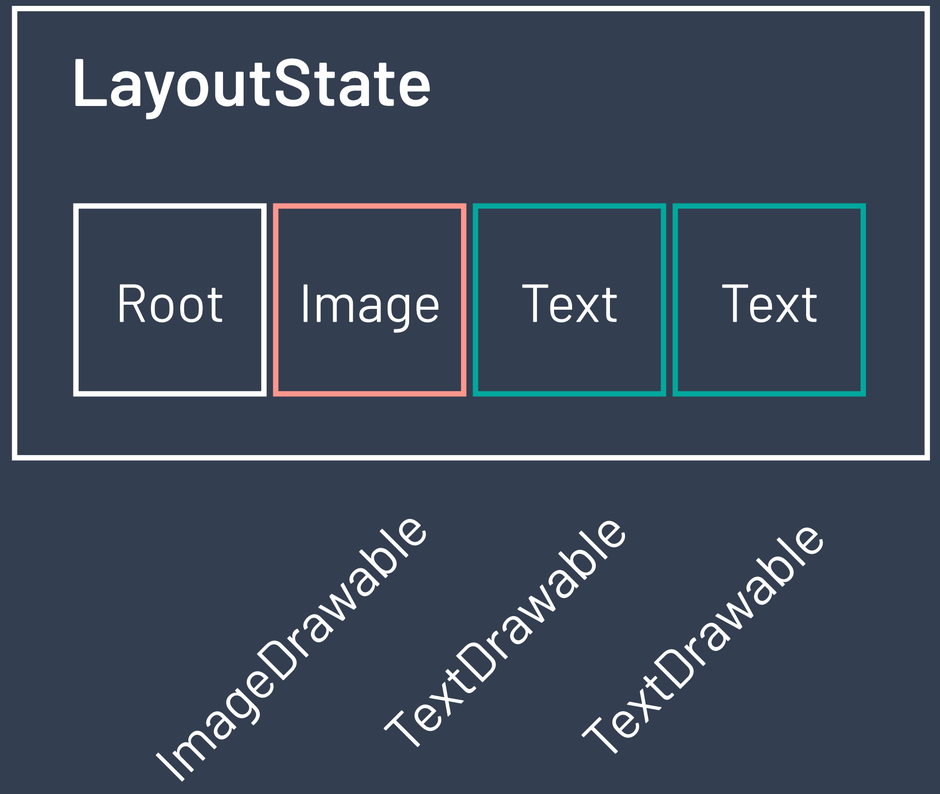

LayoutState
Root
Image
Text
Text
ImageDrawable
TextDrawable
TextDrawable

MOUNT LAYOUTSTATE



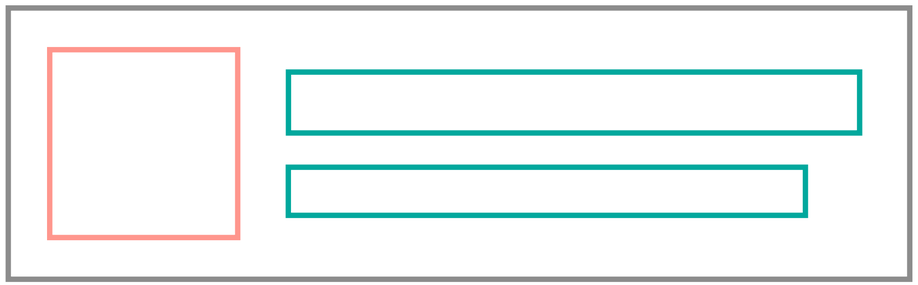
MOUNT LAYOUTSTATE

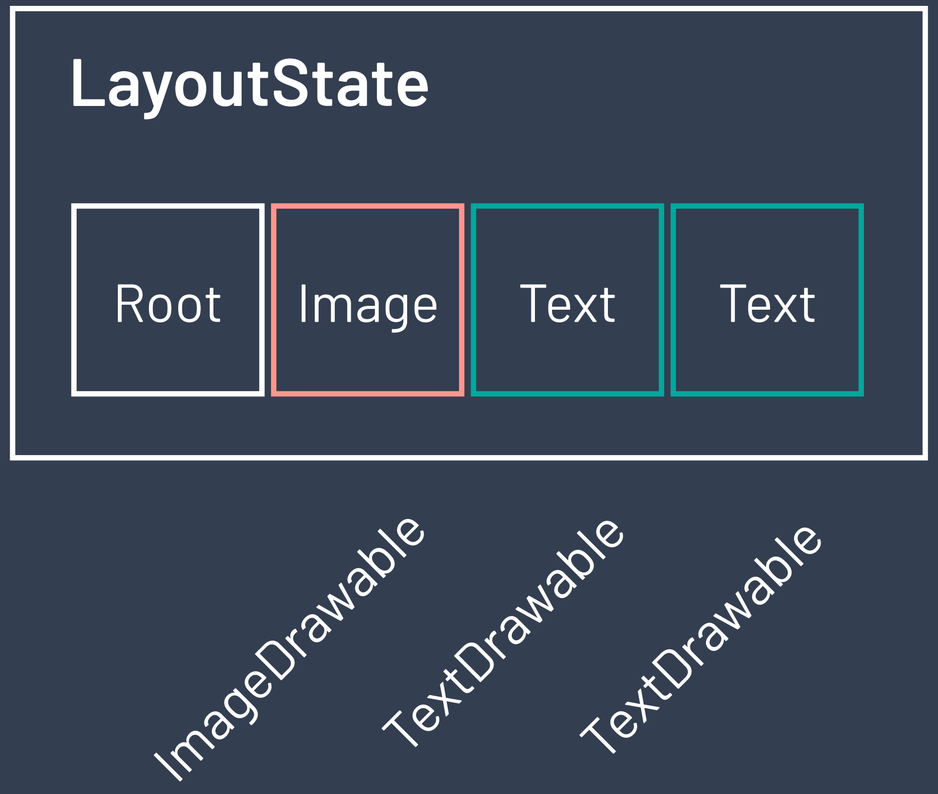
The only step that always happens on UI thread



MOUNT LAYOUTSTATE

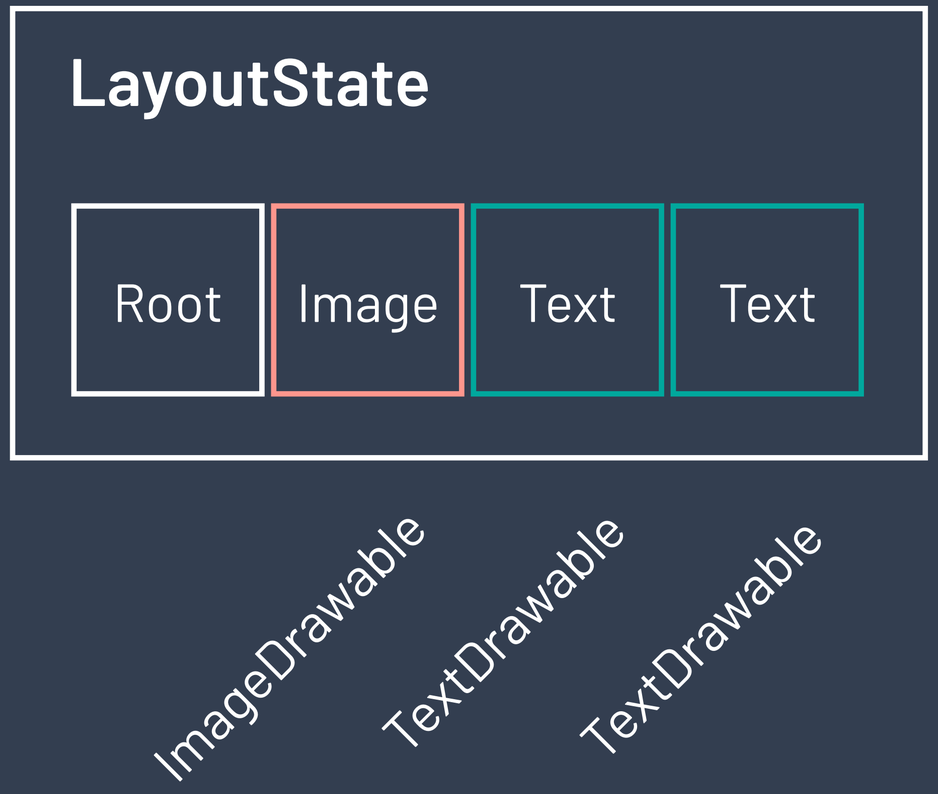
The only step that always happens on UI thread



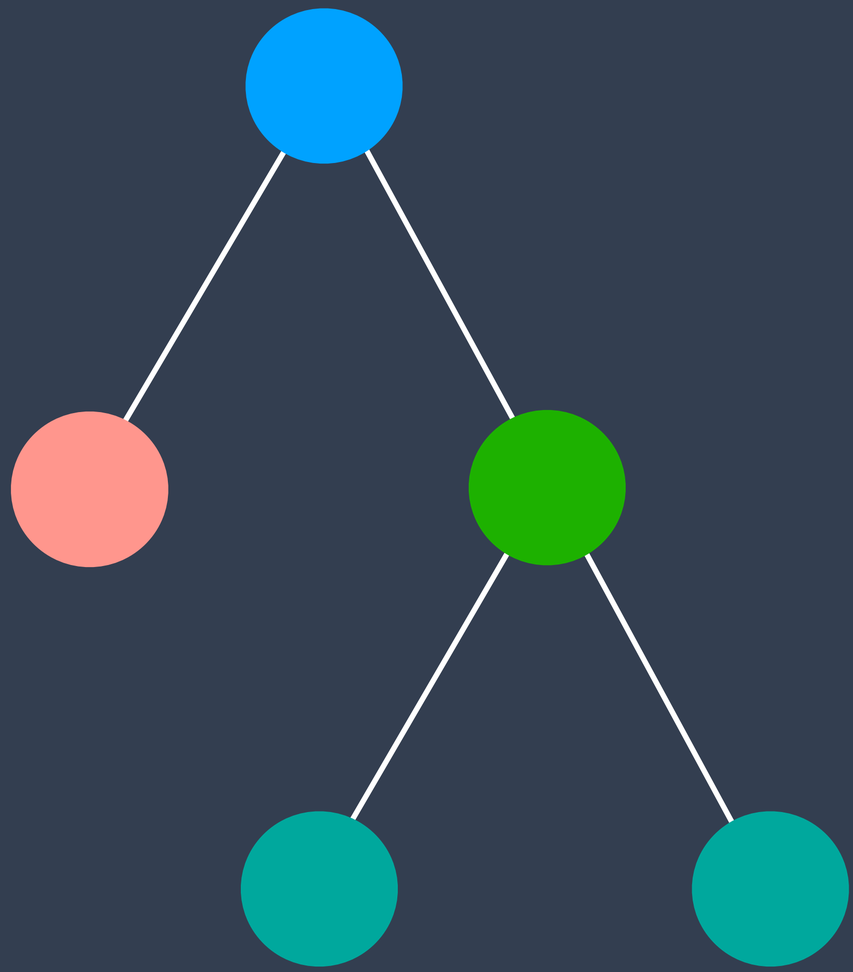


MOUNT LAYOUTSTATE

The only step that always happens on UI thread

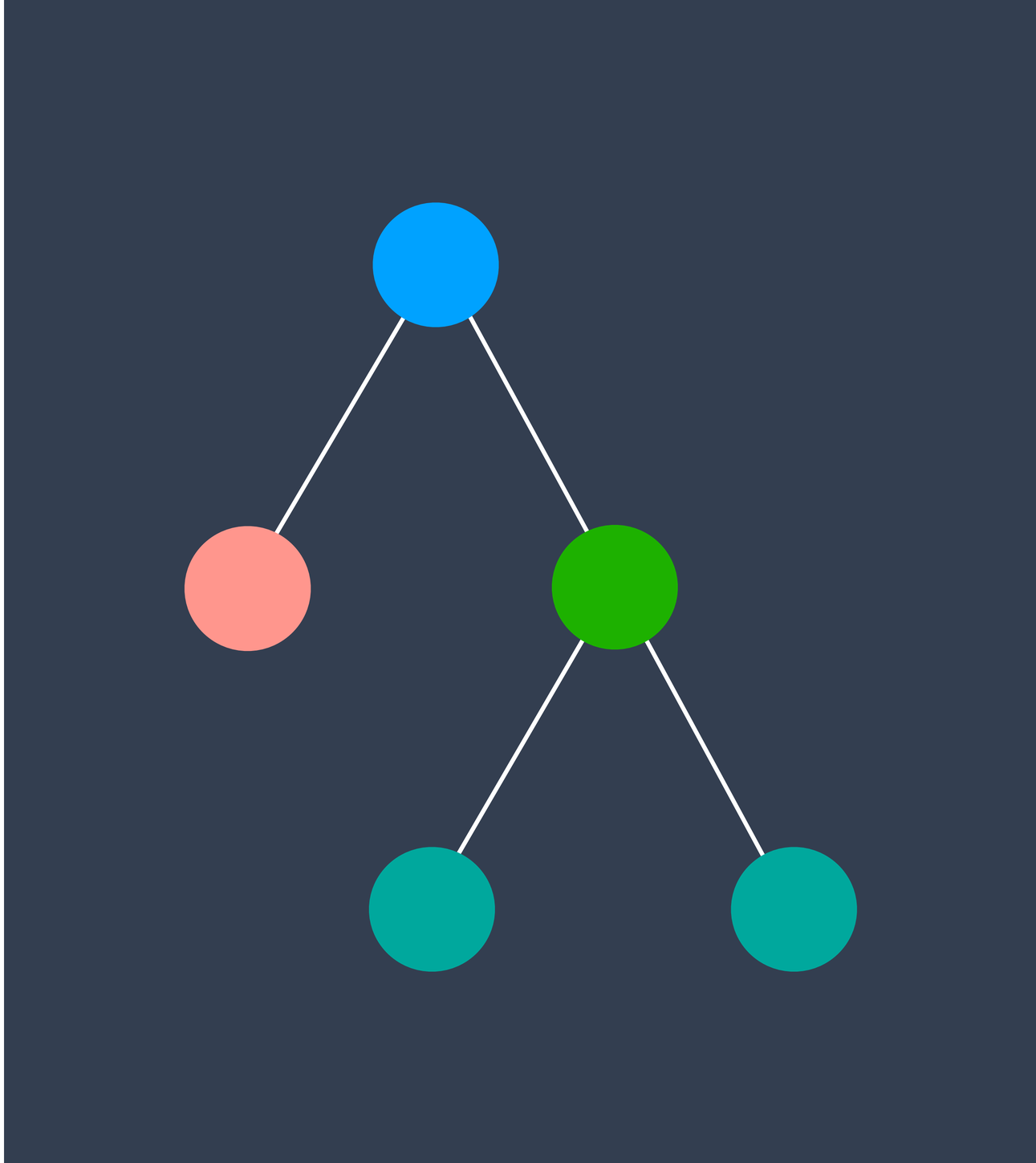


BUILDING BLOCKS



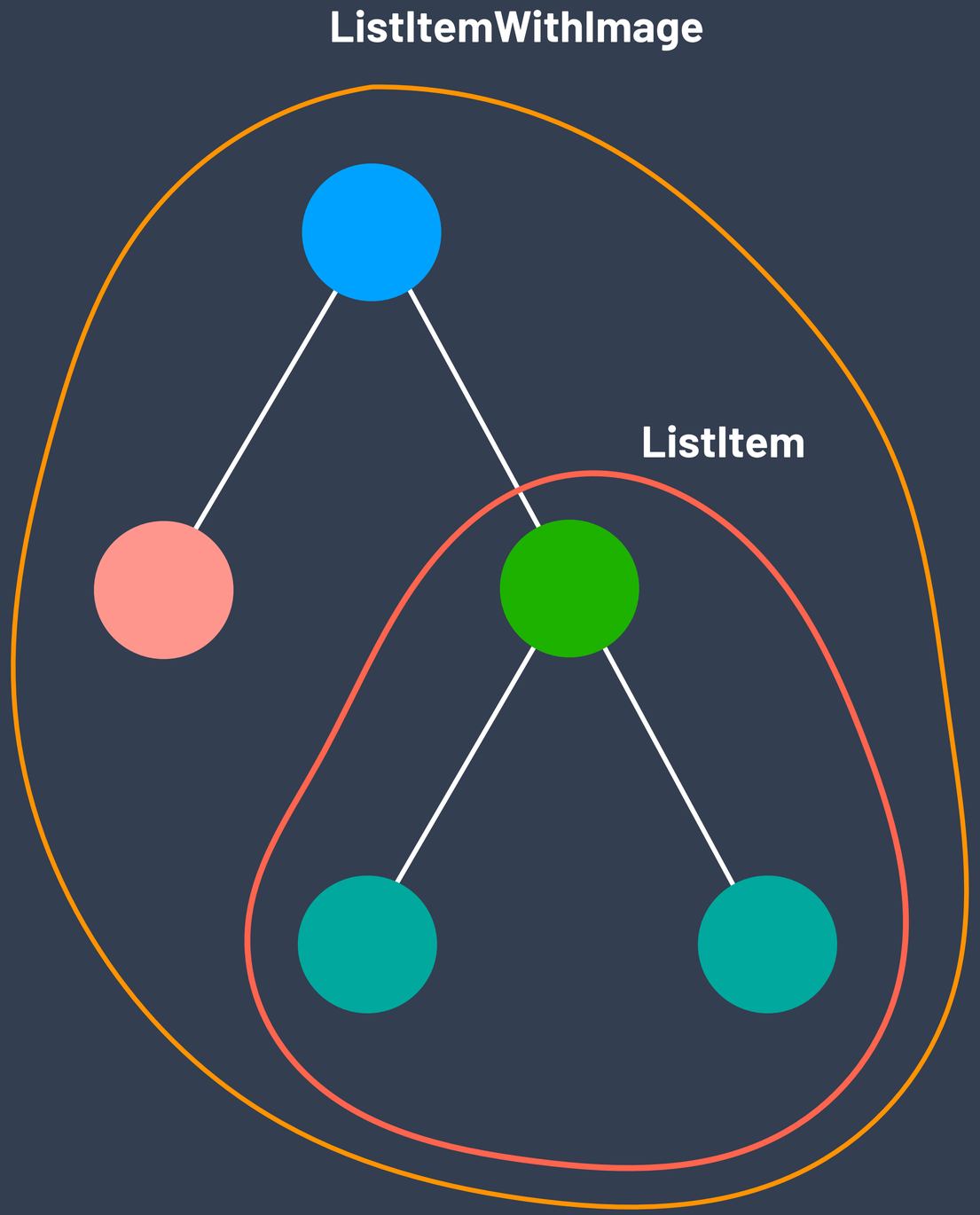
BUILDING BLOCKS

- ✦ @LayoutSpec



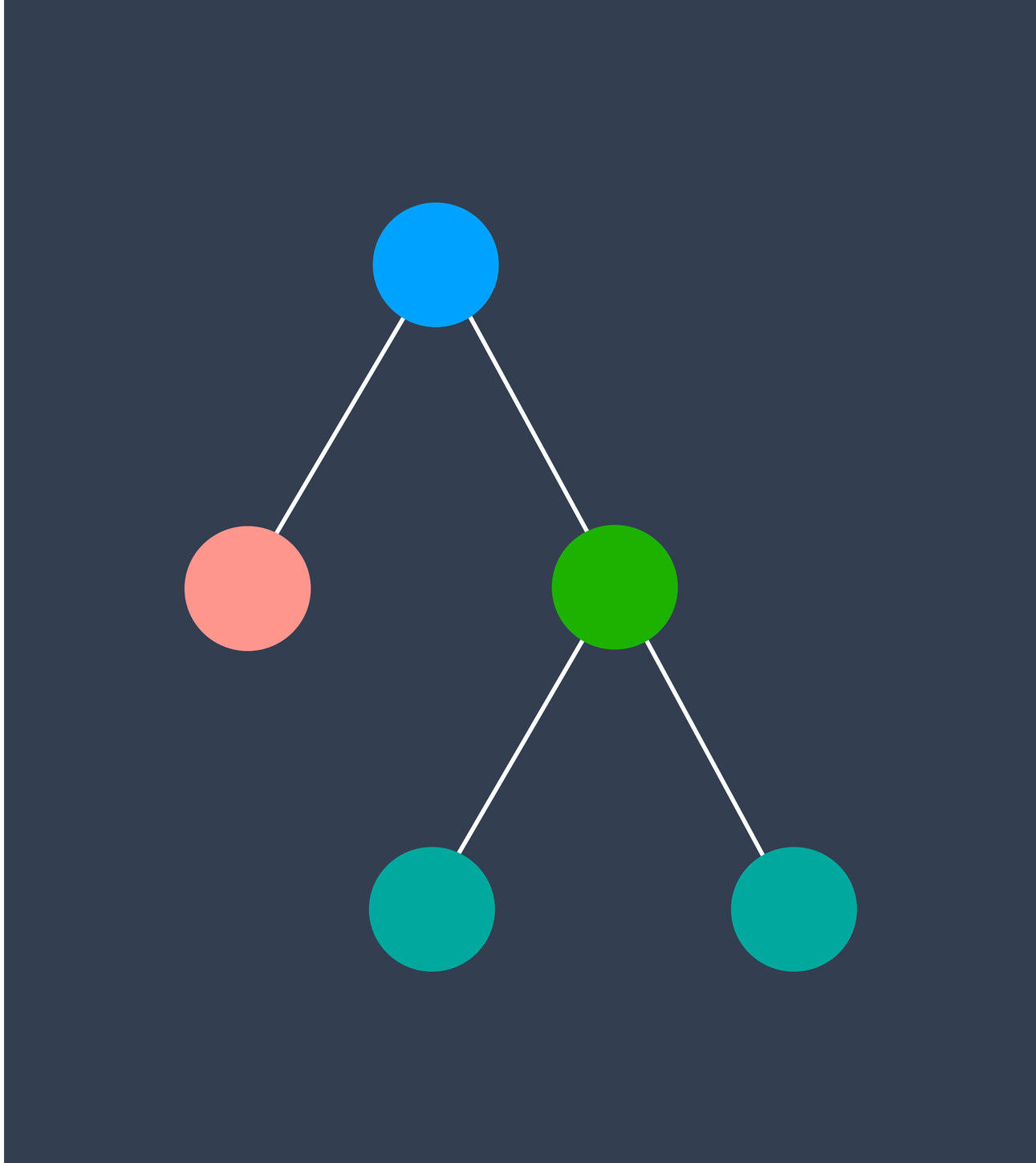
BUILDING BLOCKS

- `@LayoutSpec`



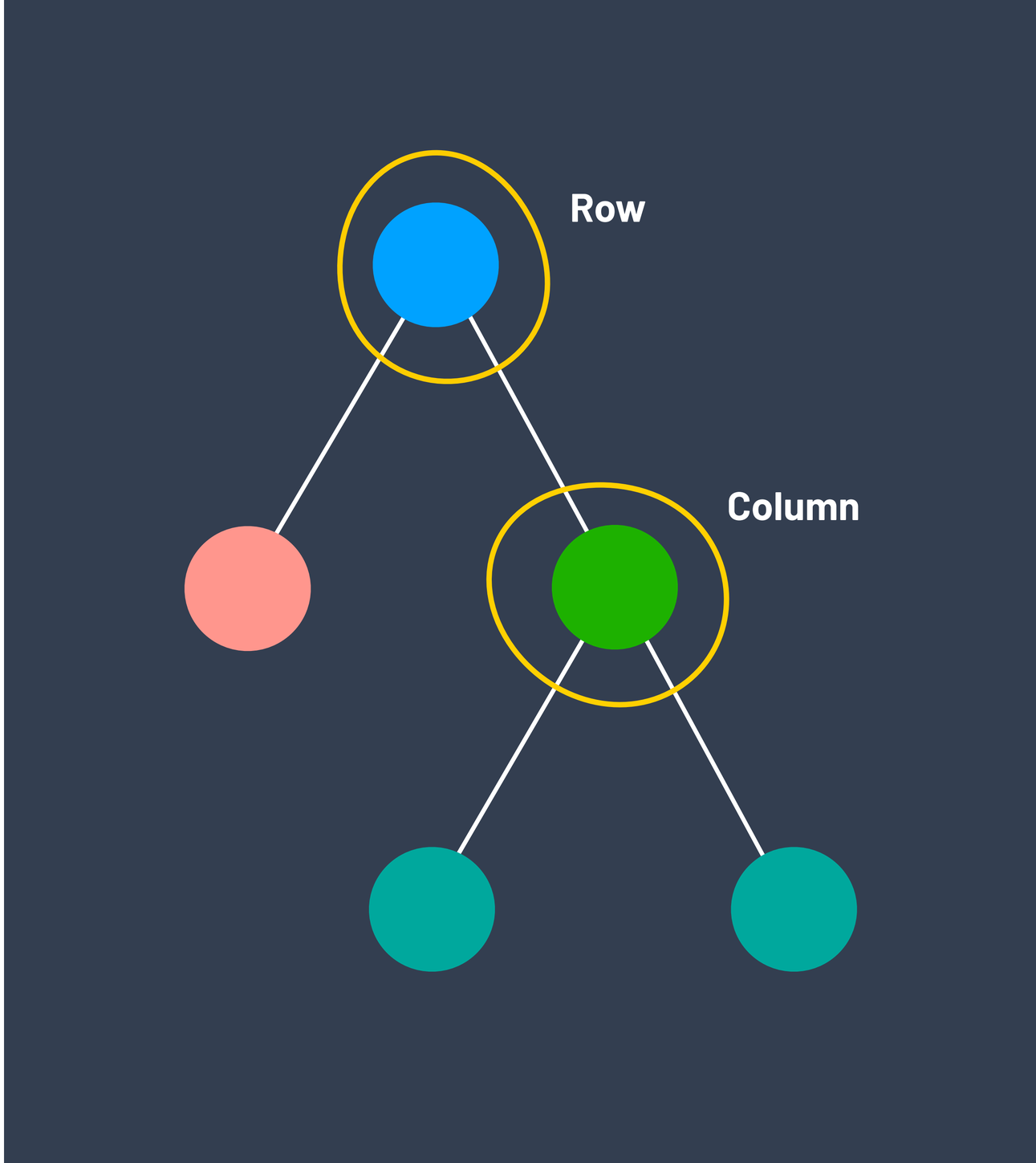
BUILDING BLOCKS

- @LayoutSpec
- Row / Column



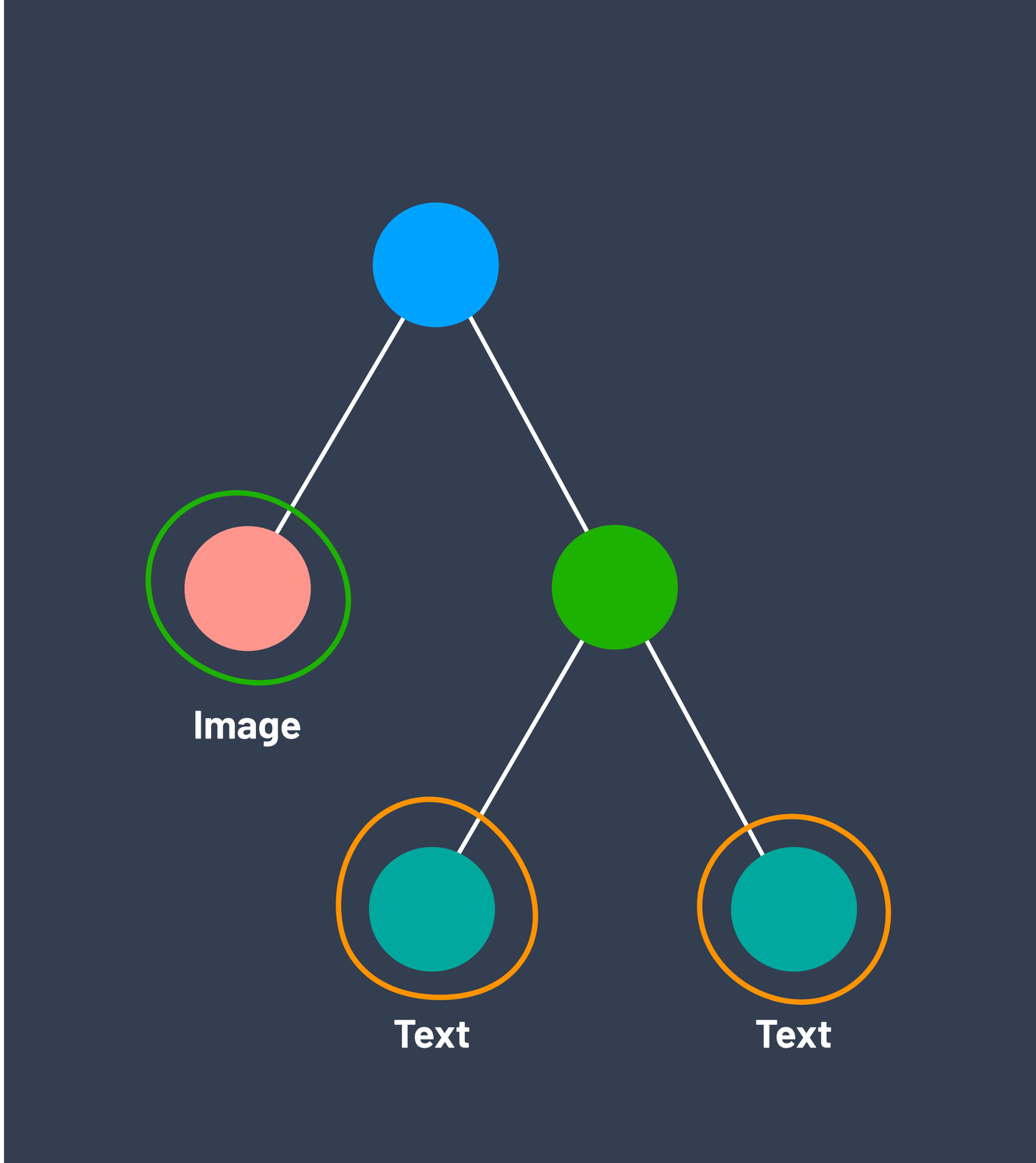
BUILDING BLOCKS

- @LayoutSpec
- Row / Column



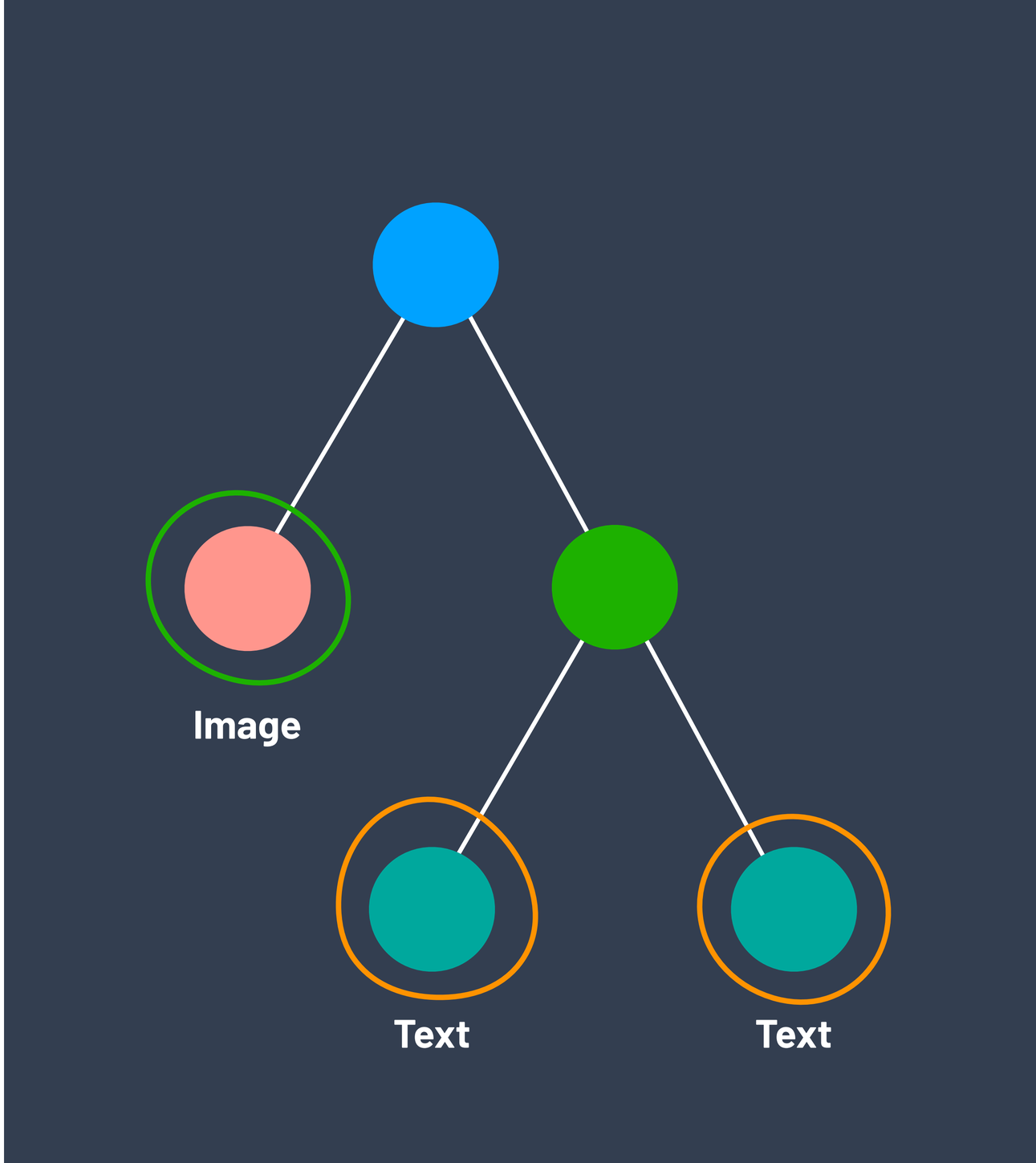
BUILDING BLOCKS

- @LayoutSpec
- Row / Column



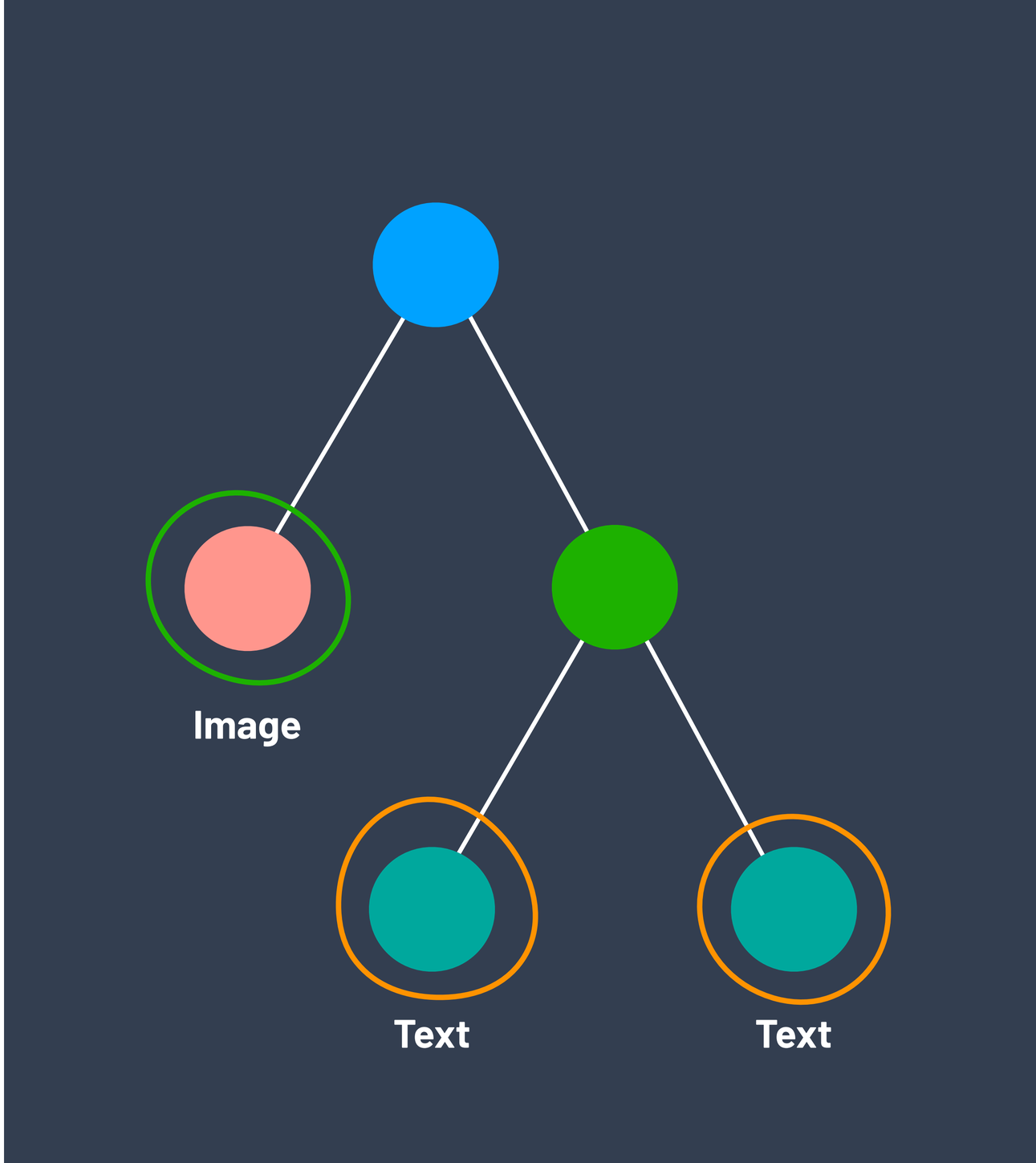
BUILDING BLOCKS

- @LayoutSpec
- Row / Column
- ...?



BUILDING BLOCKS

- @LayoutSpec
- Row / Column
- @MountSpec



```
@MountSpec
object GradientSpec {

  @OnCreateMountContent
  fun onCreateMountContent(context: Context): GradientDrawable {
    return GradientDrawable()
  }

  @OnMount
  fun onMount(
      c: ComponentContext,
      drawable: GradientDrawable,
      @Prop @ColorInt colors: IntArray) {
    drawable.colors = colors
  }

}
```

```
@MountSpec
object GradientSpec {

  @OnCreateMountContent
  fun onCreateMountContent(context: Context): GradientDrawable {
    return GradientDrawable()
  }

  @OnMount
  fun onMount(
      c: ComponentContext,
      drawable: GradientDrawable,
      @Prop @ColorInt colors: IntArray) {
    drawable.colors = colors
  }

}
```

```
@MountSpec
object GradientSpec {

  @OnCreateMountContent
  fun onCreateMountContent(context: Context): GradientDrawable {
    return GradientDrawable()
  }

  @OnMount
  fun onMount(
      c: ComponentContext,
      drawable: GradientDrawable,
      @Prop @ColorInt colors: IntArray) {
    drawable.colors = colors
  }

}
```

```
@MountSpec
object GradientSpec {

  @OnCreateMountContent
  fun onCreateMountContent(context: Context): GradientDrawable {
    return GradientDrawable()
  }

  @OnMount
  fun onMount(
      c: ComponentContext,
      drawable: GradientDrawable,
      @Prop @ColorInt colors: IntArray) {
    drawable.colors = colors
  }

}
```

```
public @interface MountSpec {
  int poolSize() default 3;
  boolean isPureRender() default false;
}
```

```
public @interface MountSpec {
  int poolSize() default 3;
  boolean isPureRender() default false;
}
```

```
public @interface MountSpec {
  int poolSize() default 3;
  boolean isPureRender() default false;
}
```

```
public @interface MountSpec {

  int poolSize() default 3;

  boolean isPureRender() default false;

}
```

```
@MountSpec(poolSize = 30, isPureRender = true)
class ImageSpec {

  @ShouldUpdate
  static boolean shouldUpdate(...) {}
}
```

```
public @interface MountSpec {

  int poolSize() default 3;

  boolean isPureRender() default false;

}
```

```
@MountSpec(poolSize = 30, isPureRender = true)
class ImageSpec {

  @ShouldUpdate
  static boolean shouldUpdate(...) {}
}
```

OPTIMIZATIONS

OPTIMIZATIONS

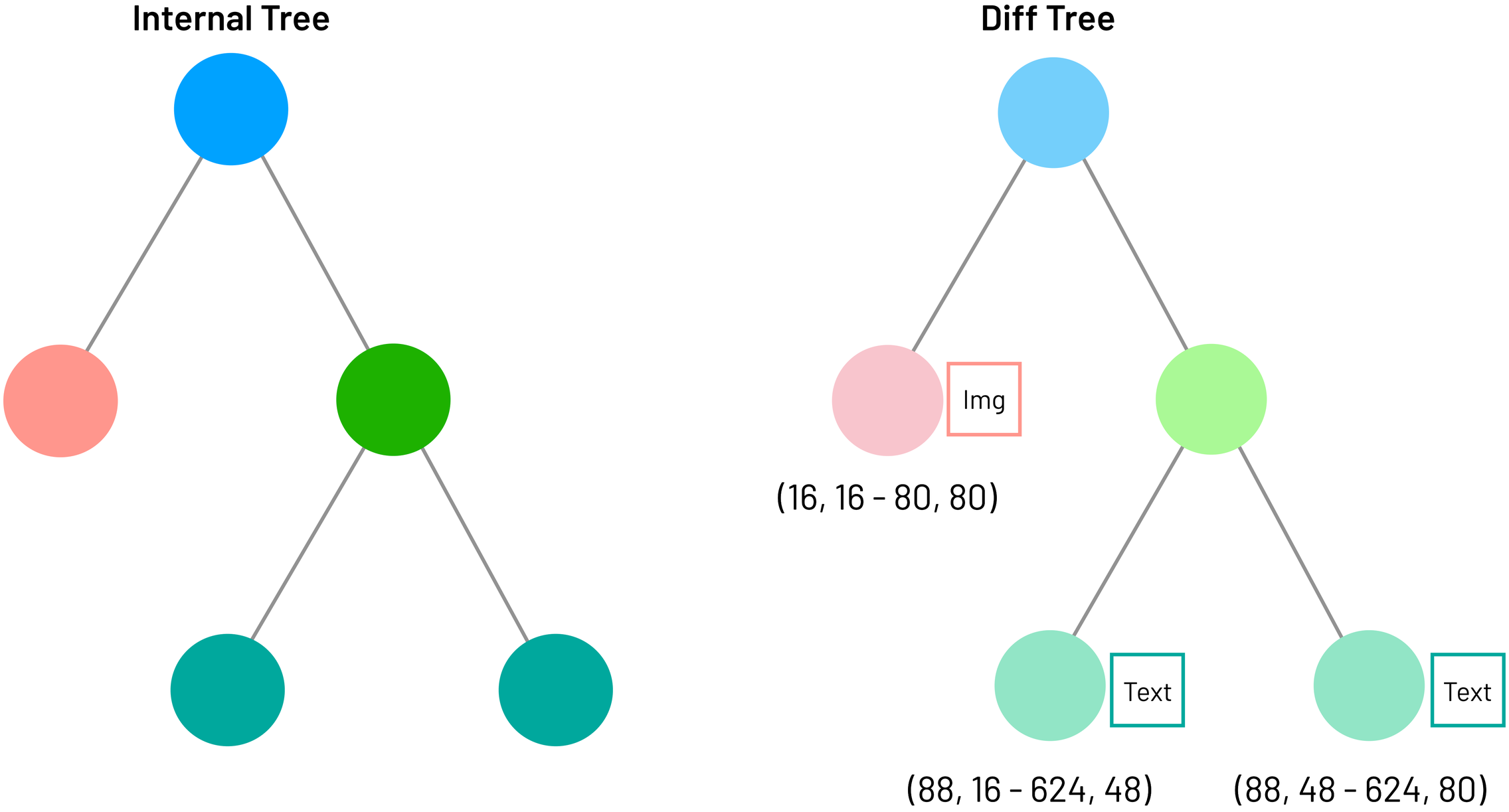
- Layout/Mount Diffing

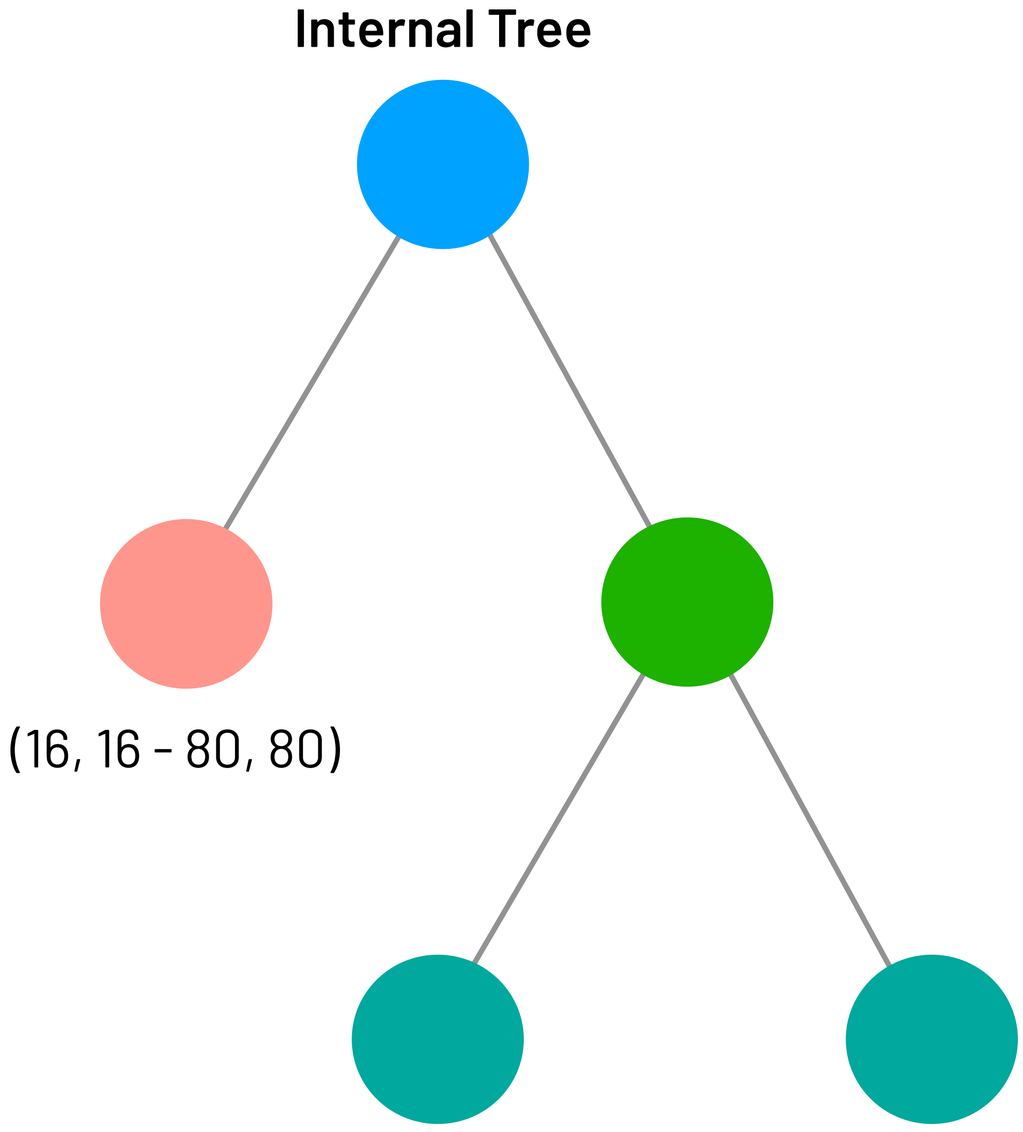
OPTIMIZATIONS

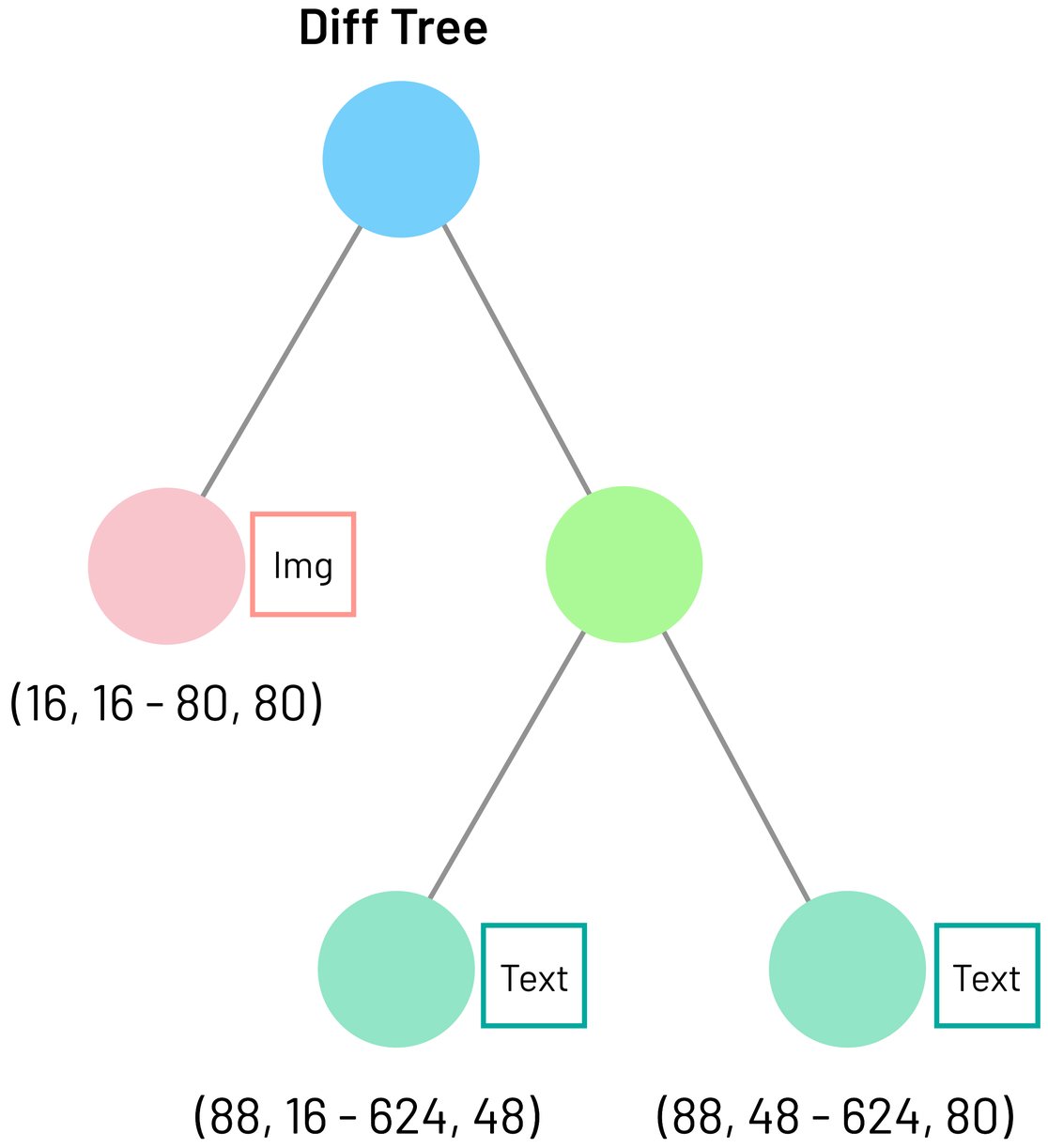
- ✦ Layout/Mount Diffing
  - Reuse Measurements

OPTIMIZATIONS

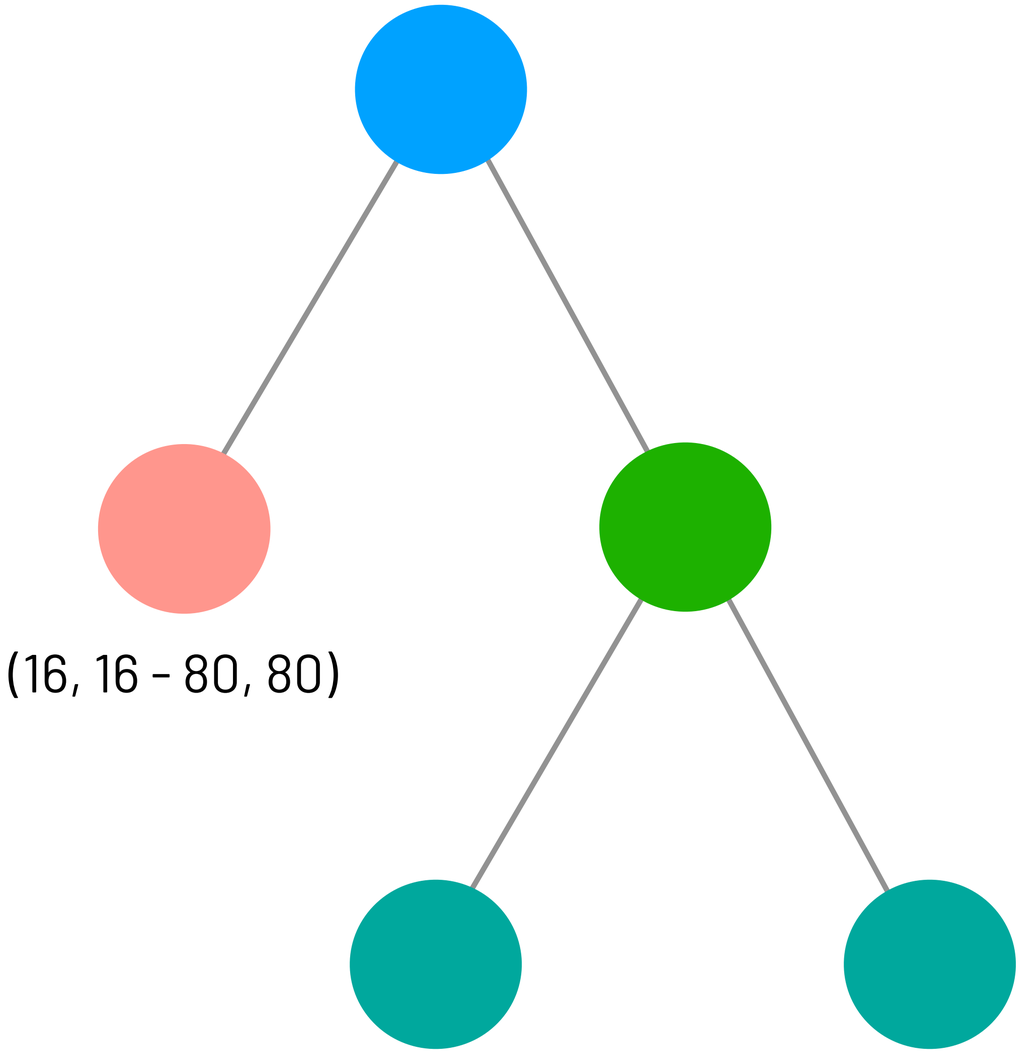
- ✦ Layout/Mount Diffing
  - Reuse Measurements
  - Reuse LayoutOutputs


Internal Tree
Diff Tree
Img
(16, 16 - 80, 80)
Text
Text
(88, 16 - 624, 48)
(88, 48 - 624, 80)


Internal Tree
(16, 16 - 80, 80)


Diff Tree
Img
(16, 16 - 80, 80)
Text
Text
(88, 16 - 624, 48)
(88, 48 - 624, 80)

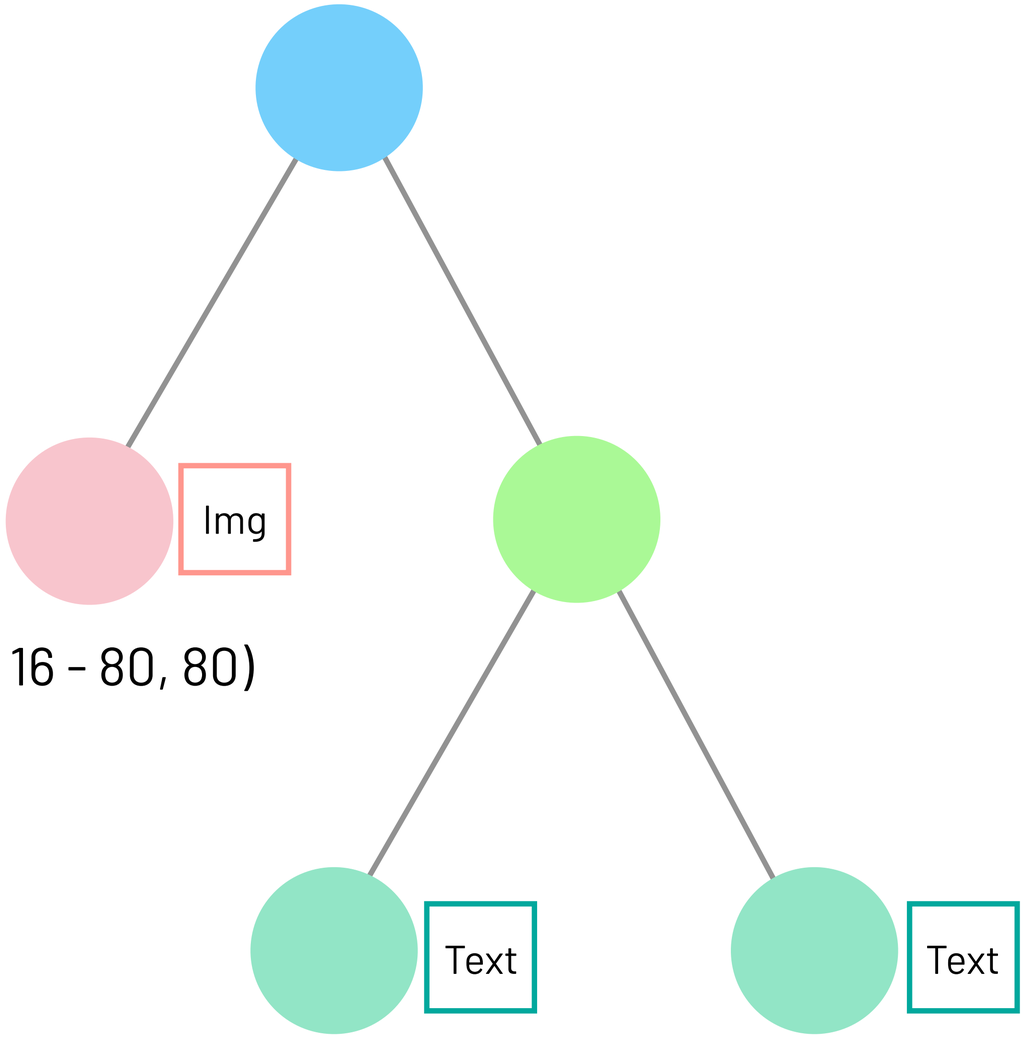
Internal Tree



(16, 16 - 80, 80)

(88, 16 - 624, 48)

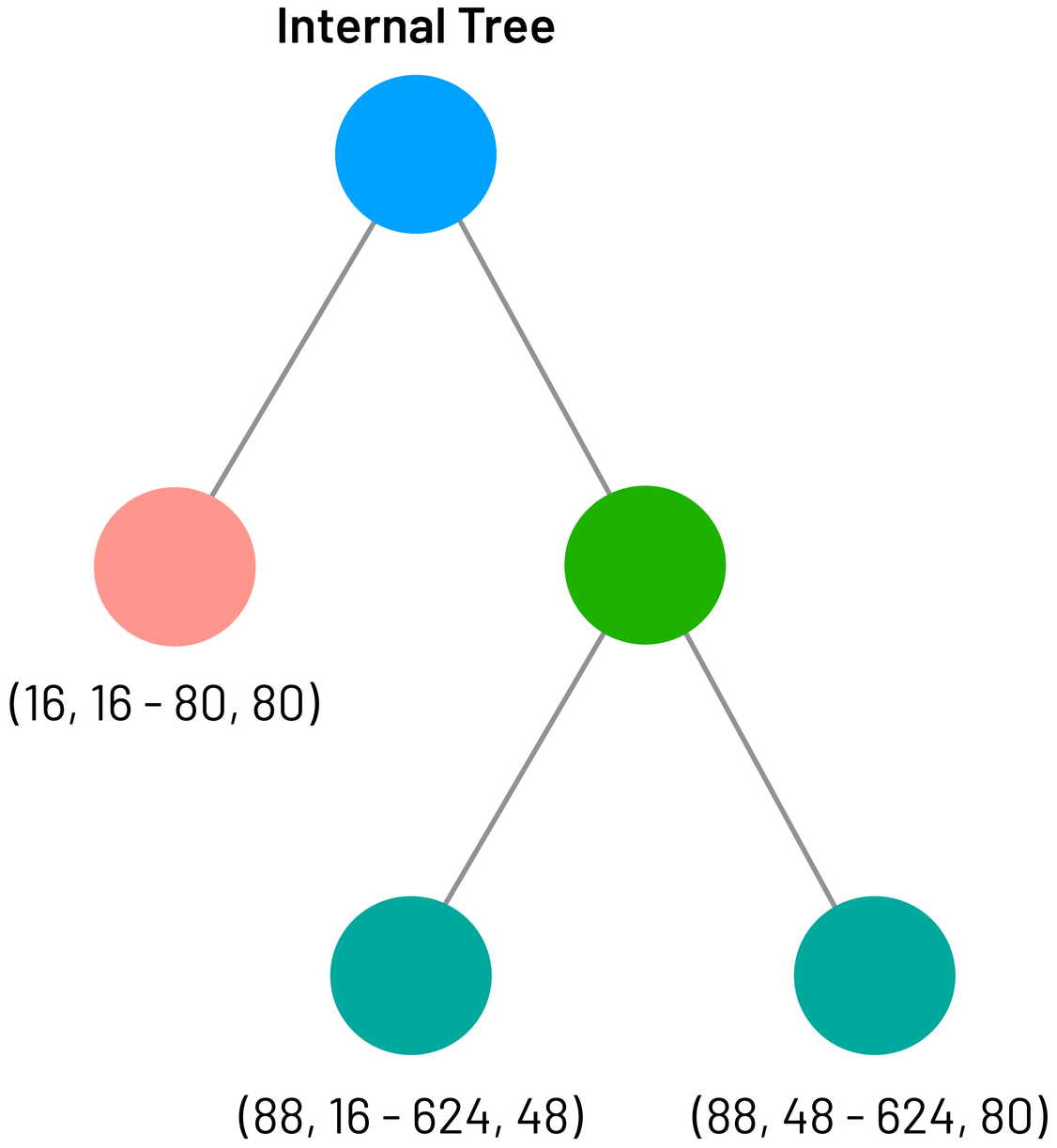
Diff Tree

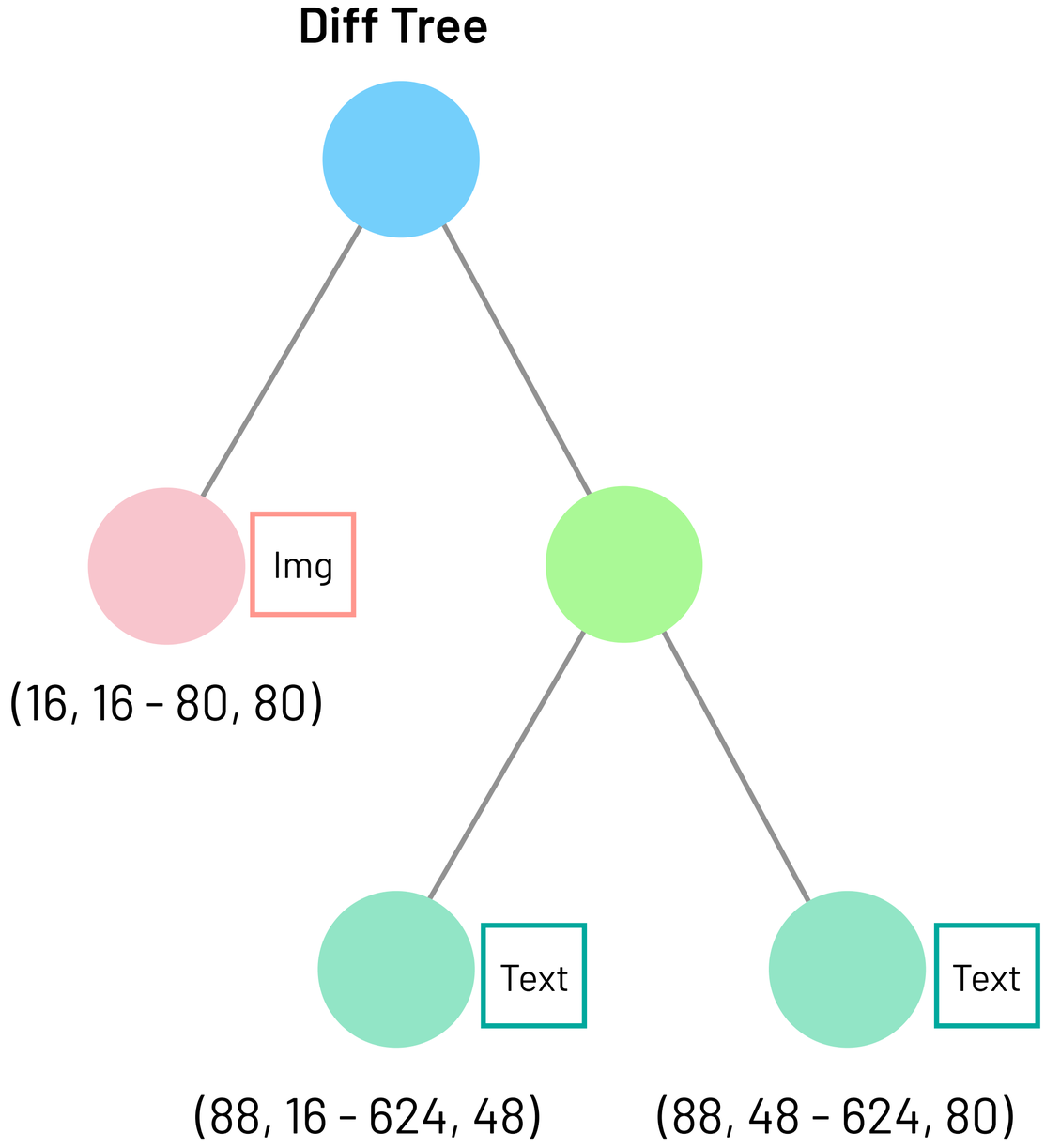


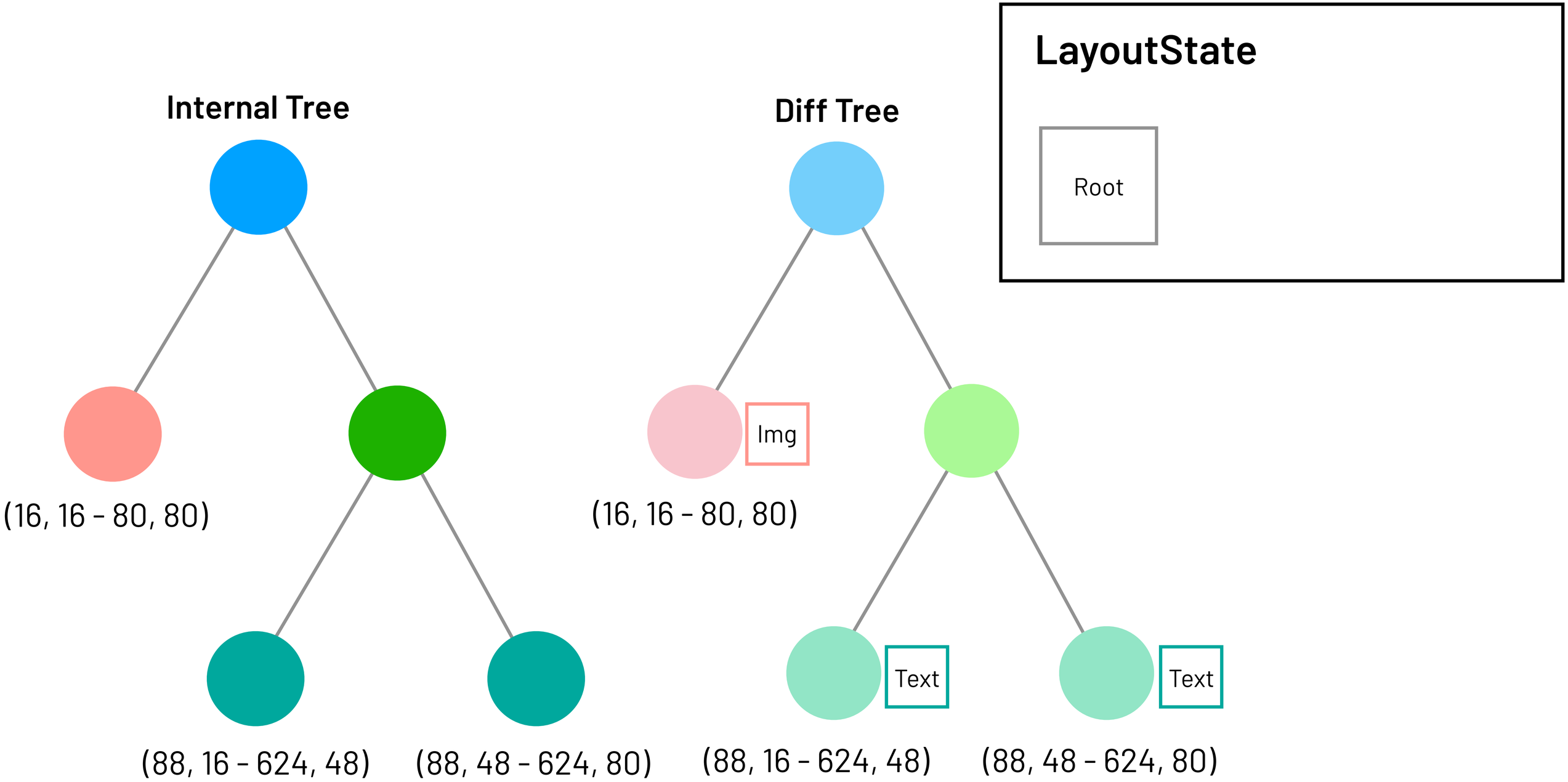
(16, 16 - 80, 80)

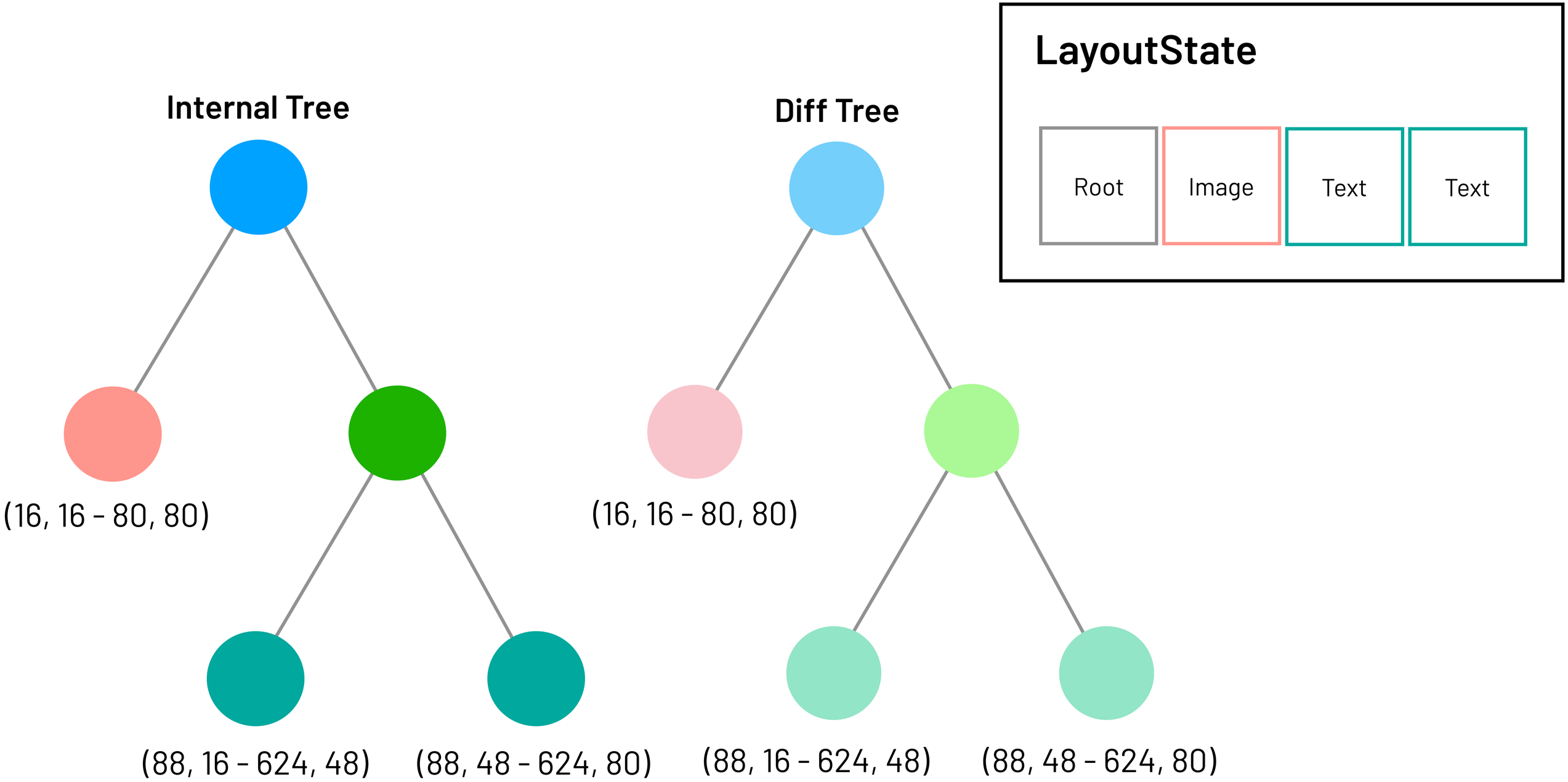
(88, 16 - 624, 48)

(88, 48 - 624, 80)


Internal Tree
(16, 16 - 80, 80)
(88, 16 - 624, 48)
(88, 48 - 624, 80)


Diff Tree
Img
(16, 16 - 80, 80)
Text
Text
(88, 16 - 624, 48)
(88, 48 - 624, 80)


LayoutState
Root
Internal Tree
(16, 16 - 80, 80)
(88, 16 - 624, 48)
(88, 48 - 624, 80)
Diff Tree
Img
(16, 16 - 80, 80)
Text
(88, 16 - 624, 48)
Text
(88, 48 - 624, 80)


Internal Tree
(16, 16 - 80, 80)
(88, 16 - 624, 48)
(88, 48 - 624, 80)
Diff Tree
(16, 16 - 80, 80)
(88, 16 - 624, 48)
(88, 48 - 624, 80)
LayoutState
Root
Image
Text
Text

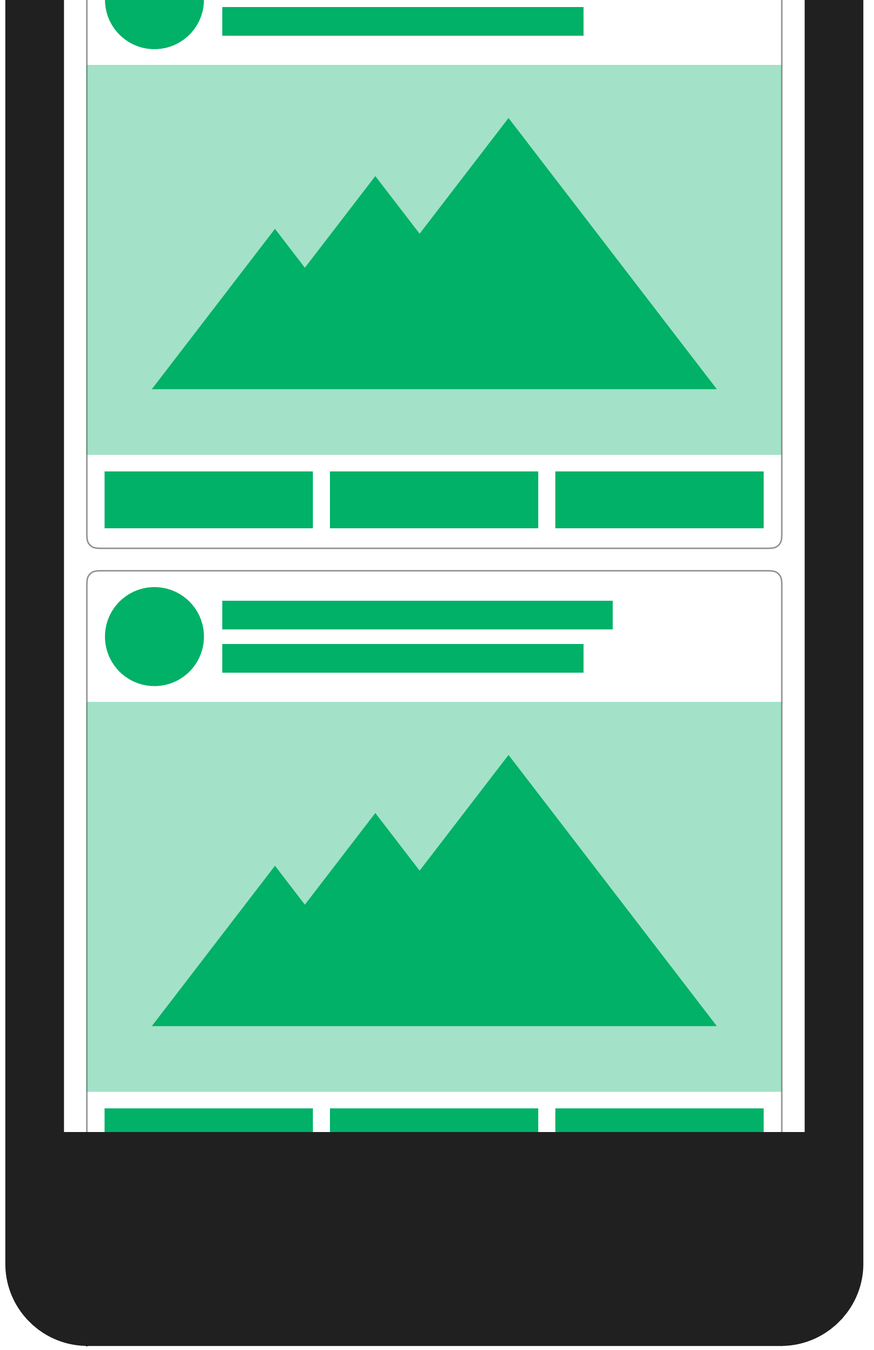
OPTIMIZATIONS

- ✦ Layout/Mount Diffing
  - Reuse Measurements
  - Reuse LayoutOutputs

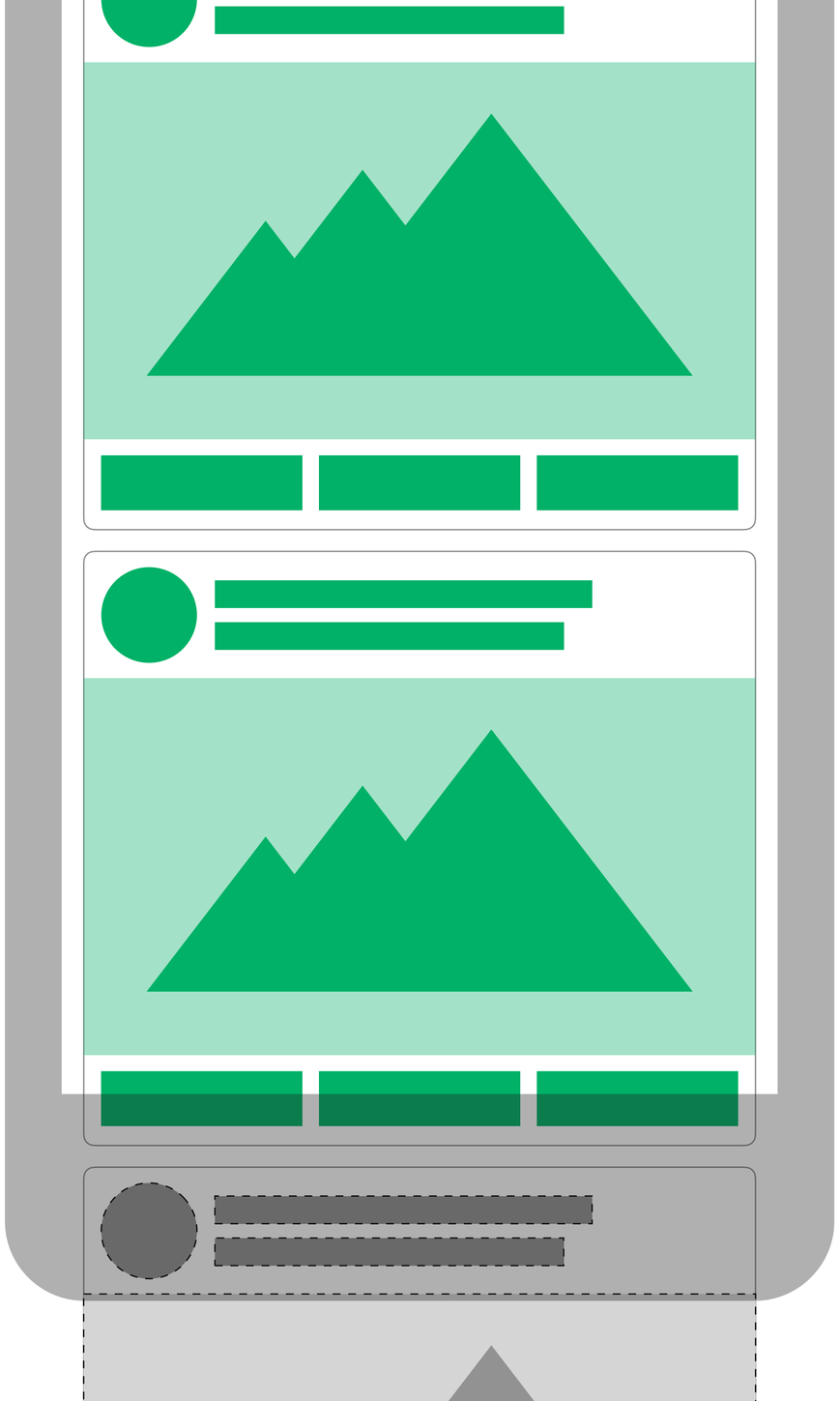
OPTIMIZATIONS

- Layout/Mount Diffing
  - Reuse Measurements
  - Reuse LayoutOutputs
- IncrementalMount

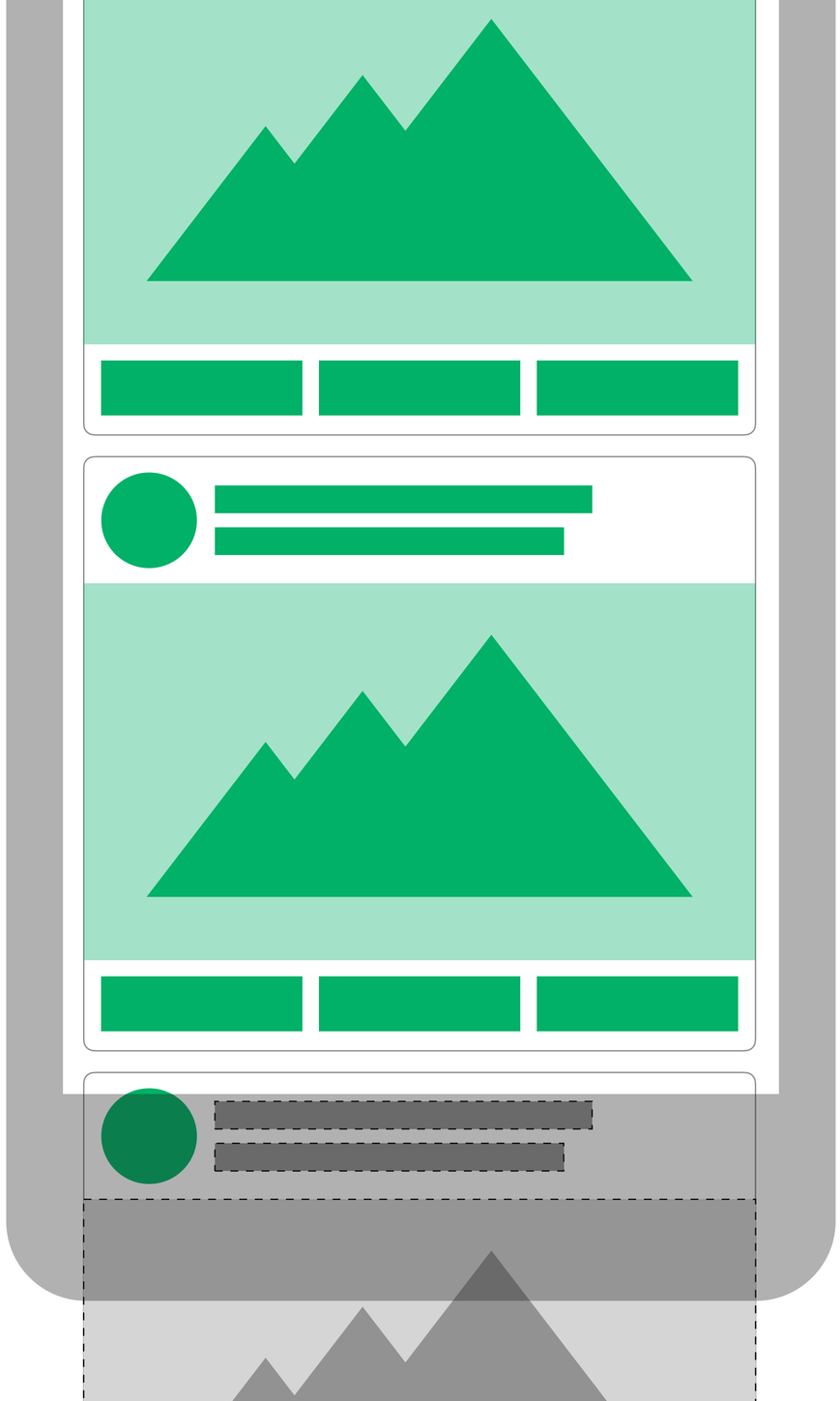
INCREMENTAL MOUNT



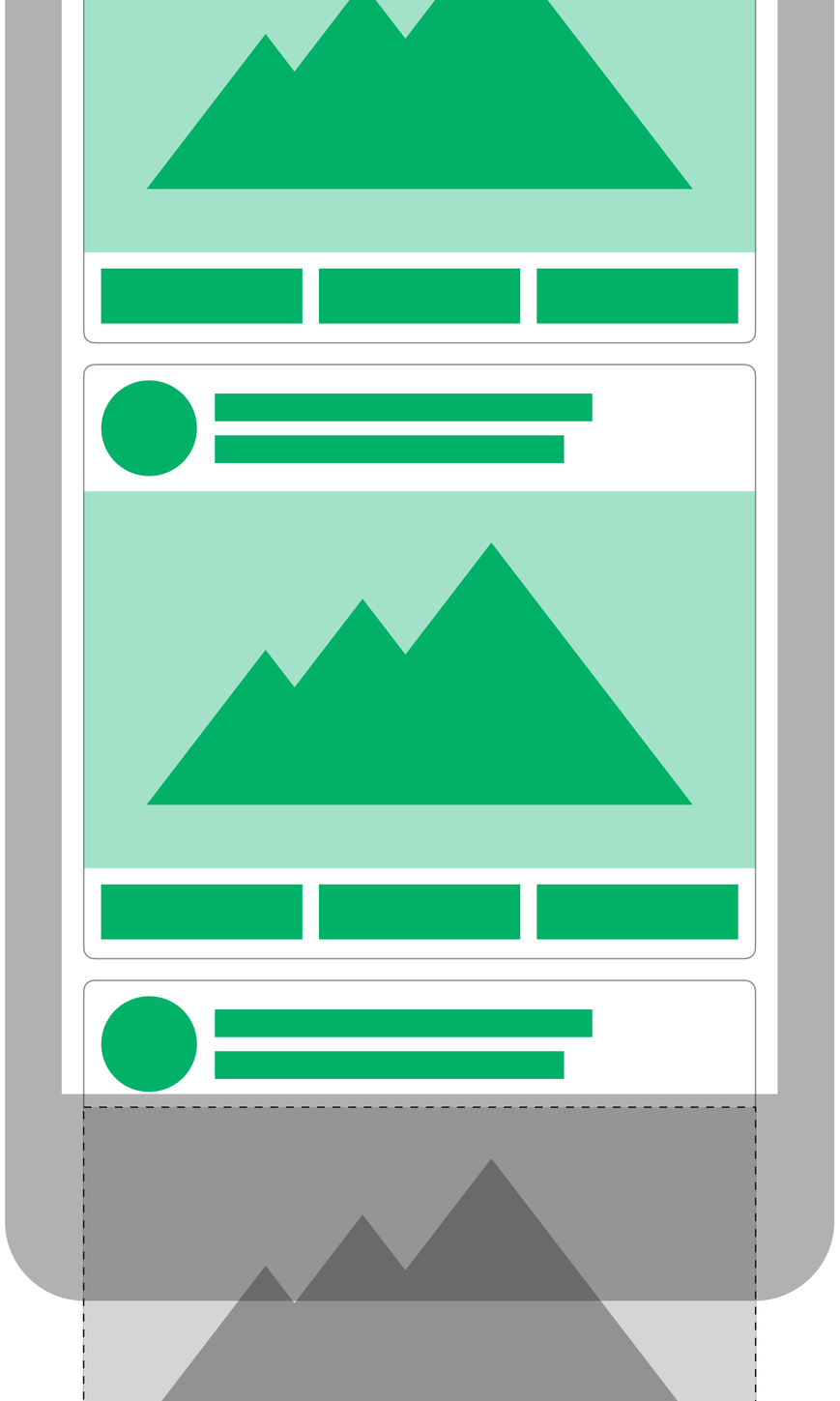
INCREMENTAL MOUNT



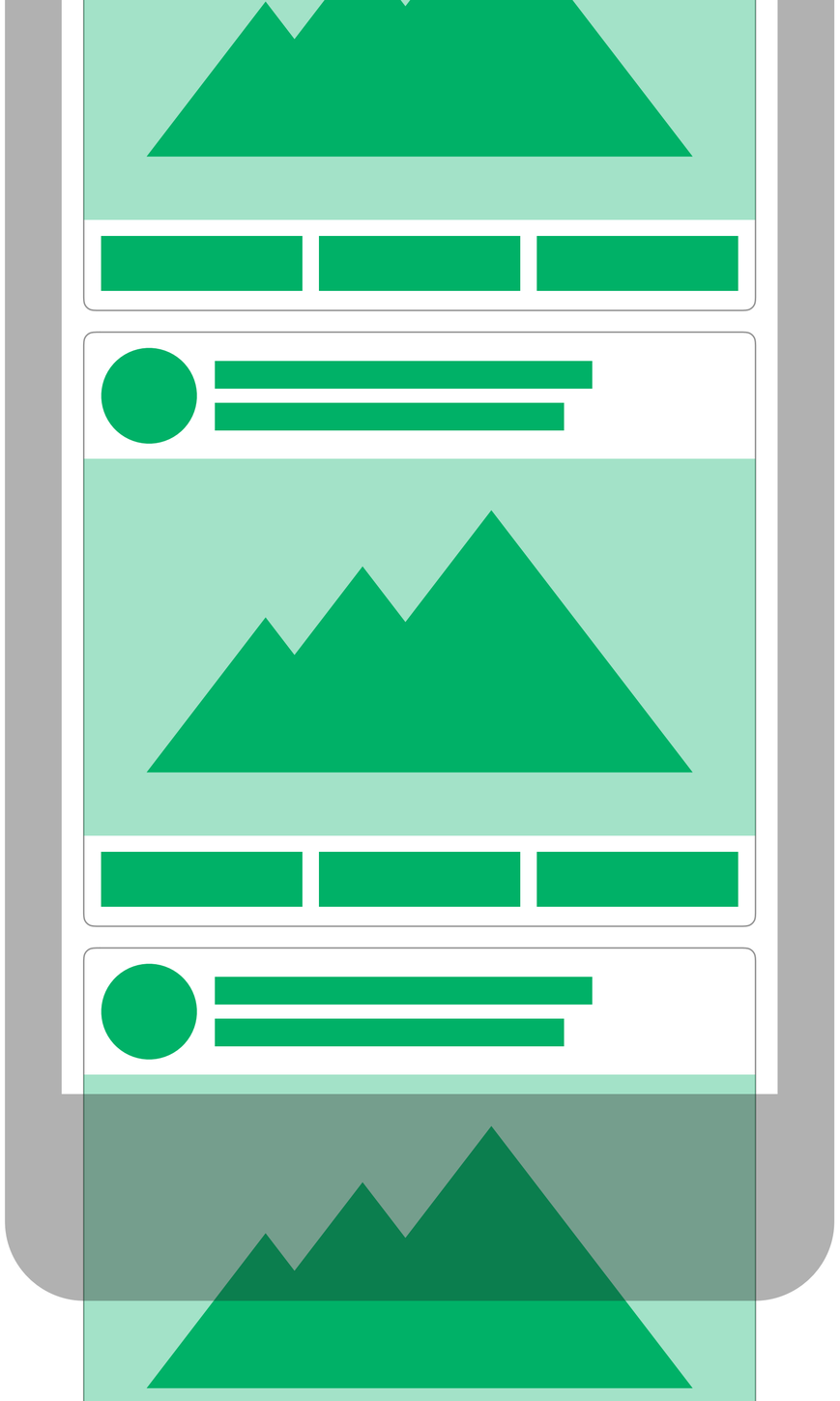
INCREMENTAL MOUNT



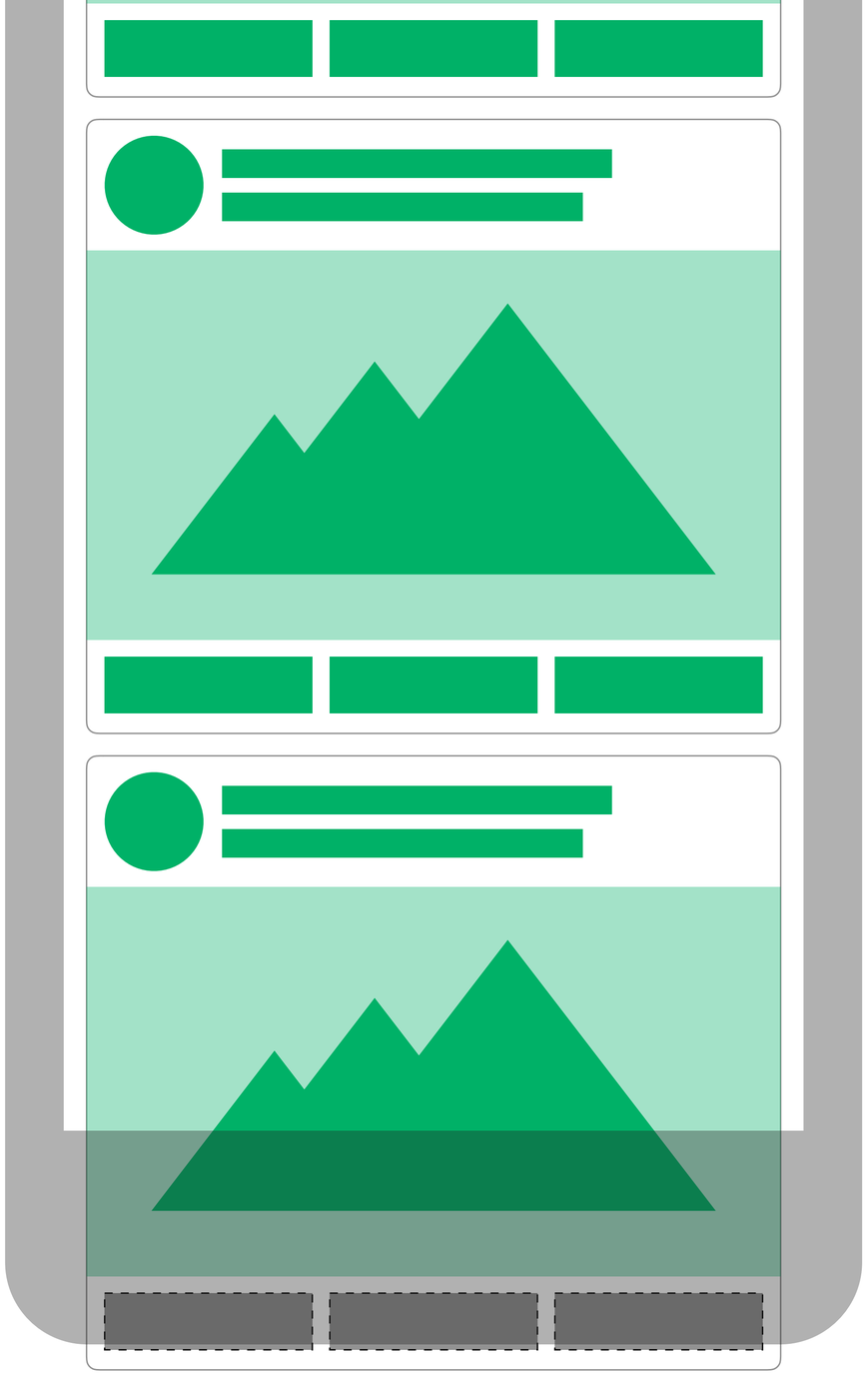
INCREMENTAL MOUNT



INCREMENTAL MOUNT



INCREMENTAL MOUNT



GOTCHAS

GOTCHAS

- Use `isPureRender` / `@ShouldUpdate` to reuse previous layout results for `@MountSpecs`

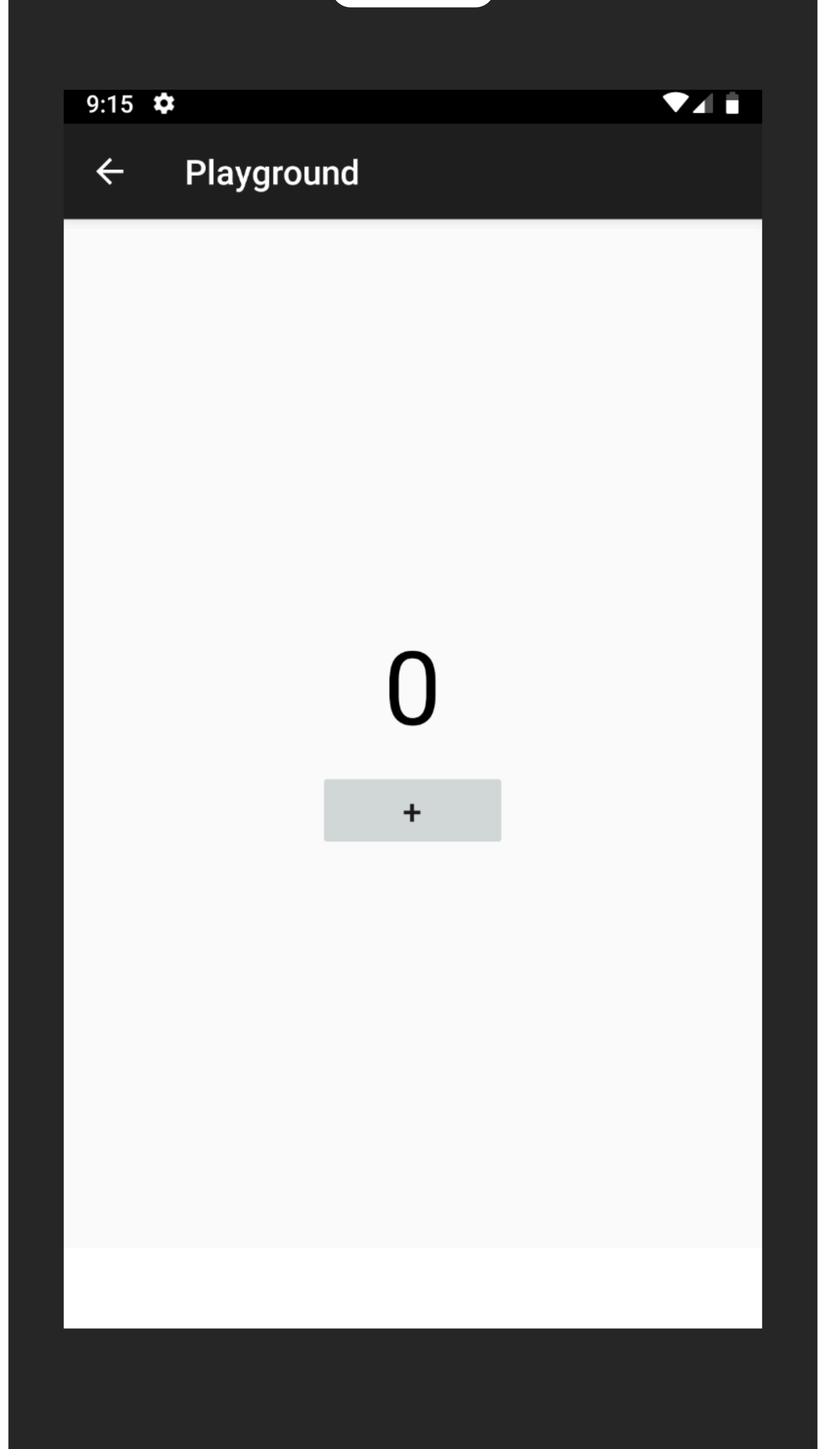
GOTCHAS

- Use `isPureRender` / `@ShouldUpdate` to reuse previous layout results for `@MountSpecs`
- Tweak MountPools according to you requirements

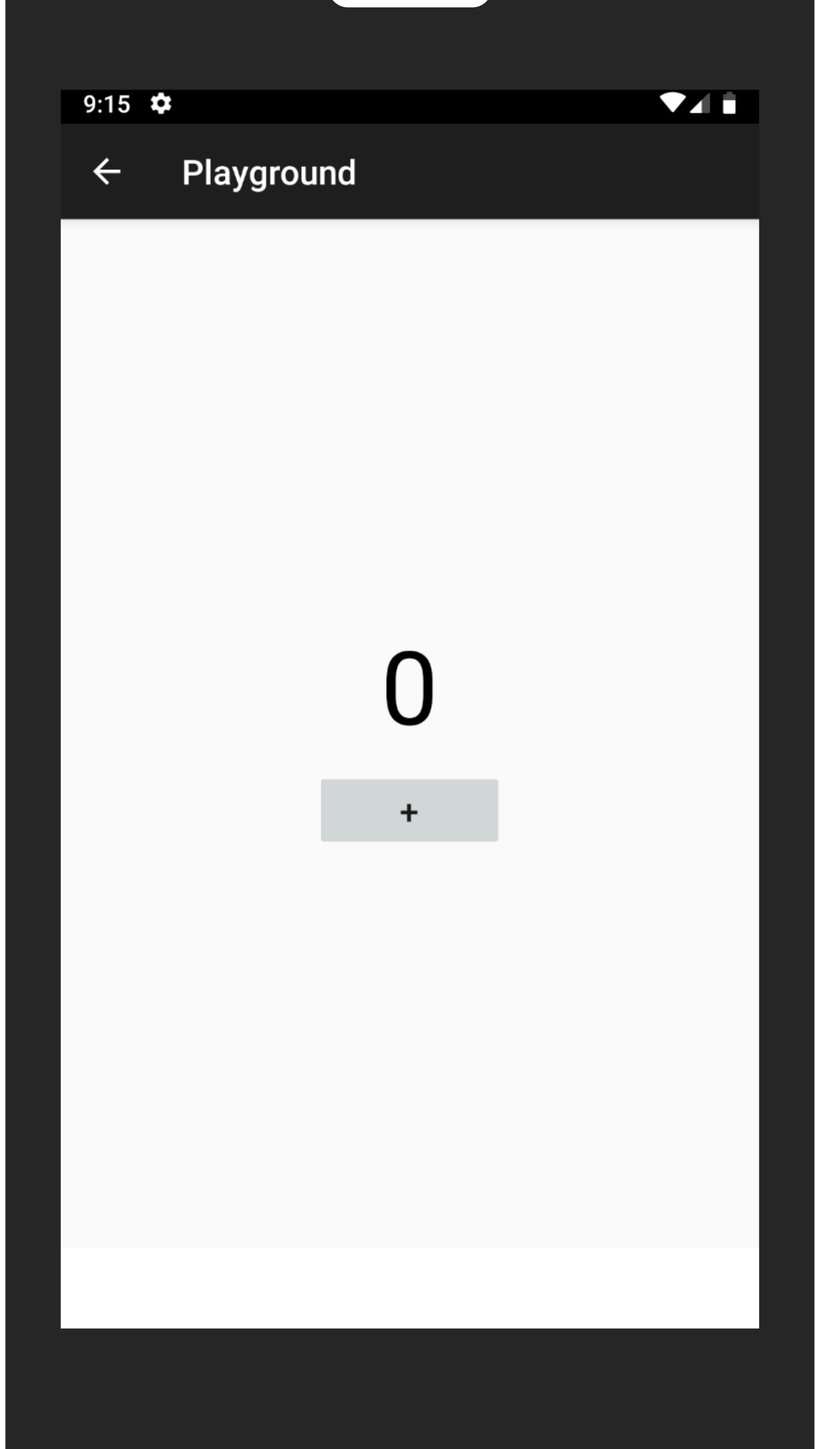
MANAGING STATE

```
@OnCreateLayout
fun onCreateLayout(
    c: ComponentContext,
    @State count: Int): Component {
  return Column.create(c)
      .child(
          Text.create(c)
              .text("$count")
      )
      .child(
          Text.create(c)
              .text("+")
      )
      .build()
}
```

```
@OnCreateLayout
fun onCreateLayout(
    c: ComponentContext,
    @State count: Int): Component {
  return Column.create(c)
      .child(
          Text.create(c)
              .text("$count")
      )
      .child(
          Text.create(c)
              .text("+")
      )
      .build()
}
```



```
@OnCreateLayout
fun onCreateLayout(
    c: ComponentContext,
    @State count: Int): Component {
  return Column.create(c)
      .child(
          Text.create(c)
              .text("$count")
      )
      .child(
          Text.create(c)
              .text("+")
      )
      .build()
}
```



```
@OnUpdateState
fun increment(count: StateValue<Int>) {
  count.set(count.get() + 1)
}
```

```
@OnUpdateState
fun increment(count: StateValue<Int>) {
  count.set(count.get() + 1)
}

@OnEvent(ClickEvent::class)
fun onIncrement(c: ComponentContext) {
  RootComponent.increment(c)
}
```

```
@OnUpdateState
fun increment(count: StateValue<Int>) {
  count.set(count.get() + 1)
}

@OnEvent(ClickEvent::class)
fun onIncrement(c: ComponentContext) {
  RootComponent.increment(c)
}

// ...

      .child(
          Text.create(c)
              .text("+")
              .clickHandler(RootComponent.onIncrement(c))
      )
```

```
@OnUpdateState
fun increment(count: StateValue<Int>) {
  count.set(count.get() + 1)
}

@OnEvent(ClickEvent::class)
fun onIncrement(c: ComponentContext) {
  RootComponent.increment(c)
}
```

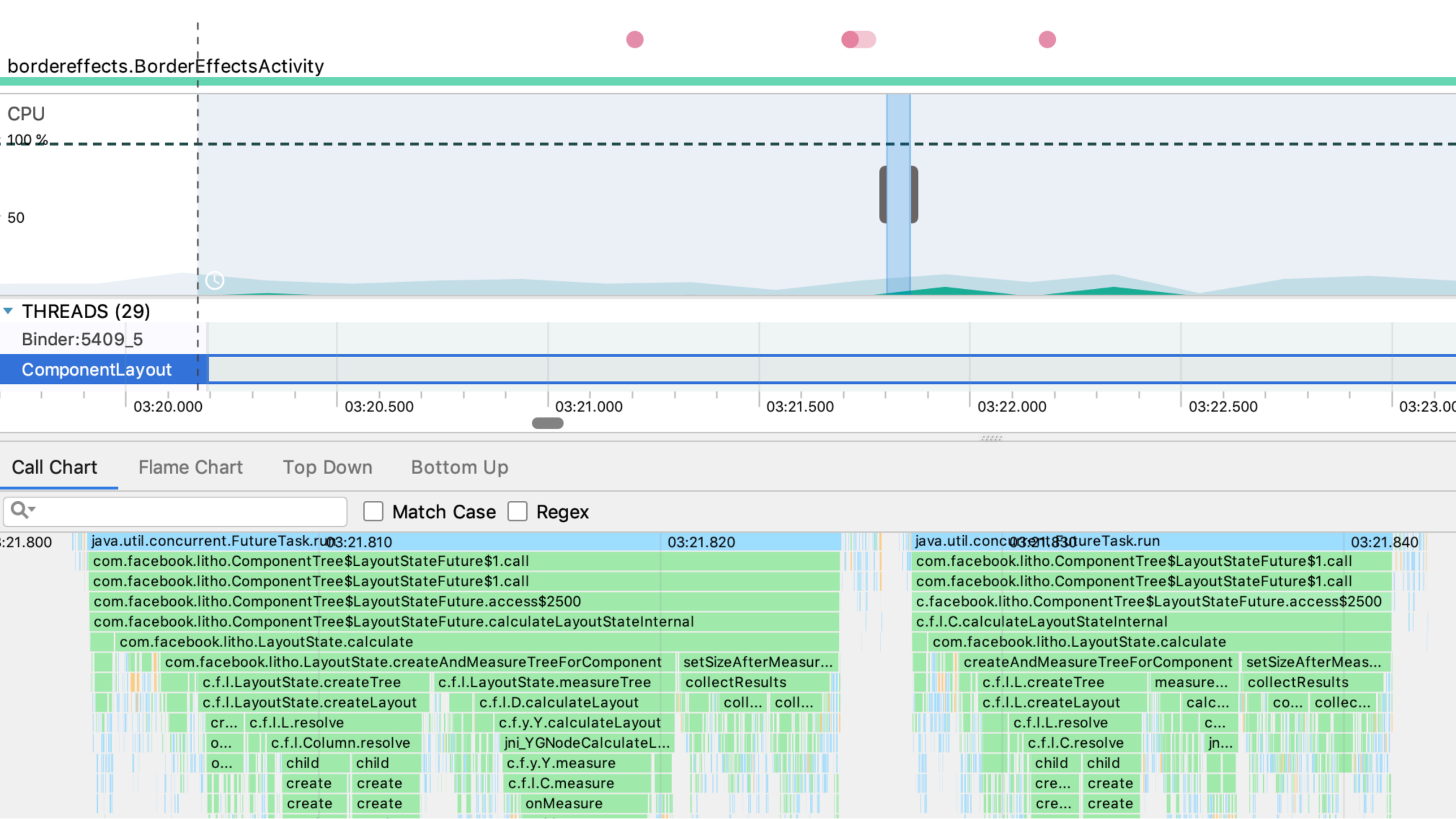
```
@OnUpdateState
fun increment(count: StateValue<Int>) {
  count.set(count.get() + 1)
}
```

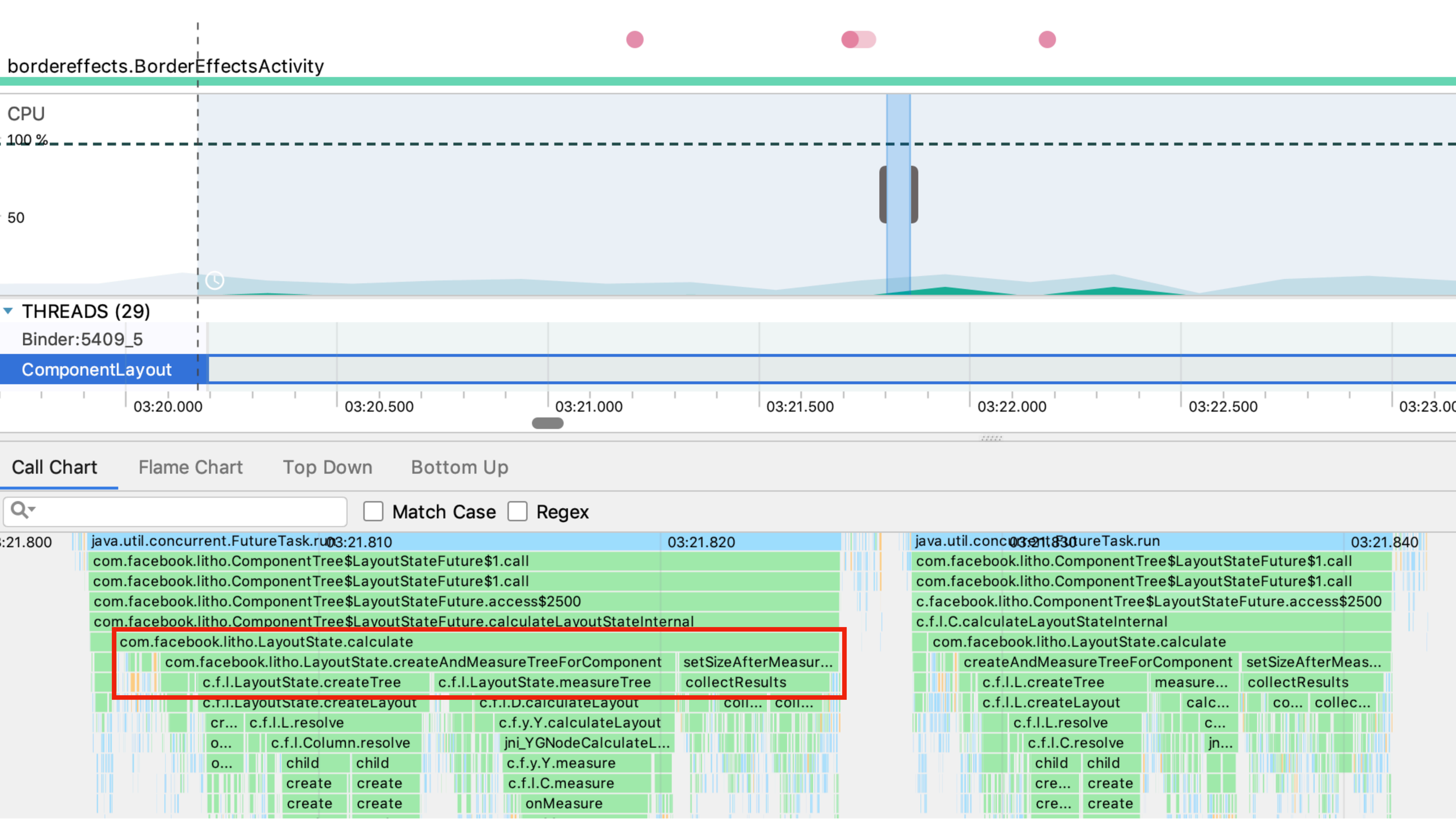
```
  RootComponent.increment(c)
```

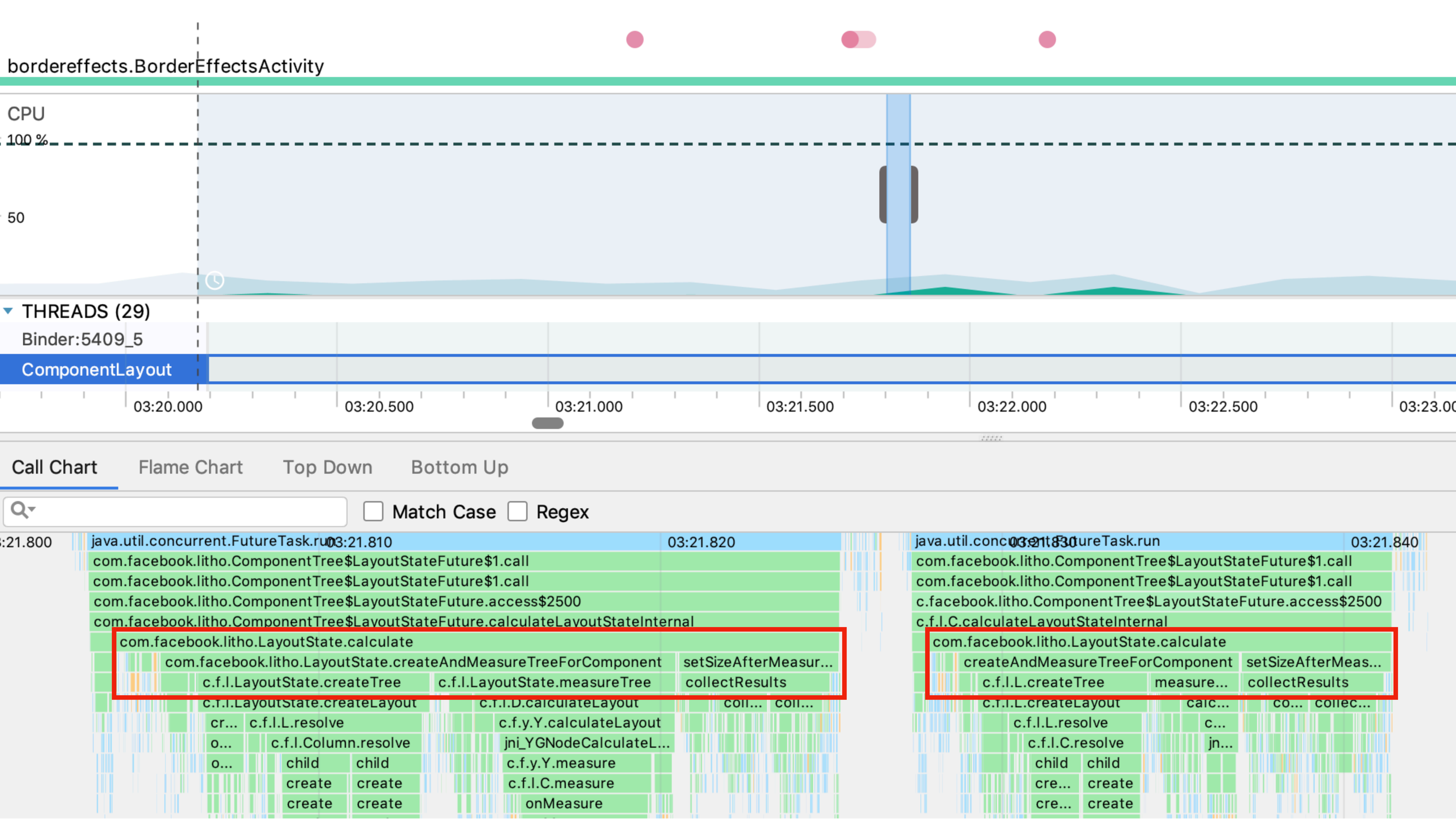
State updates? I'll take two!

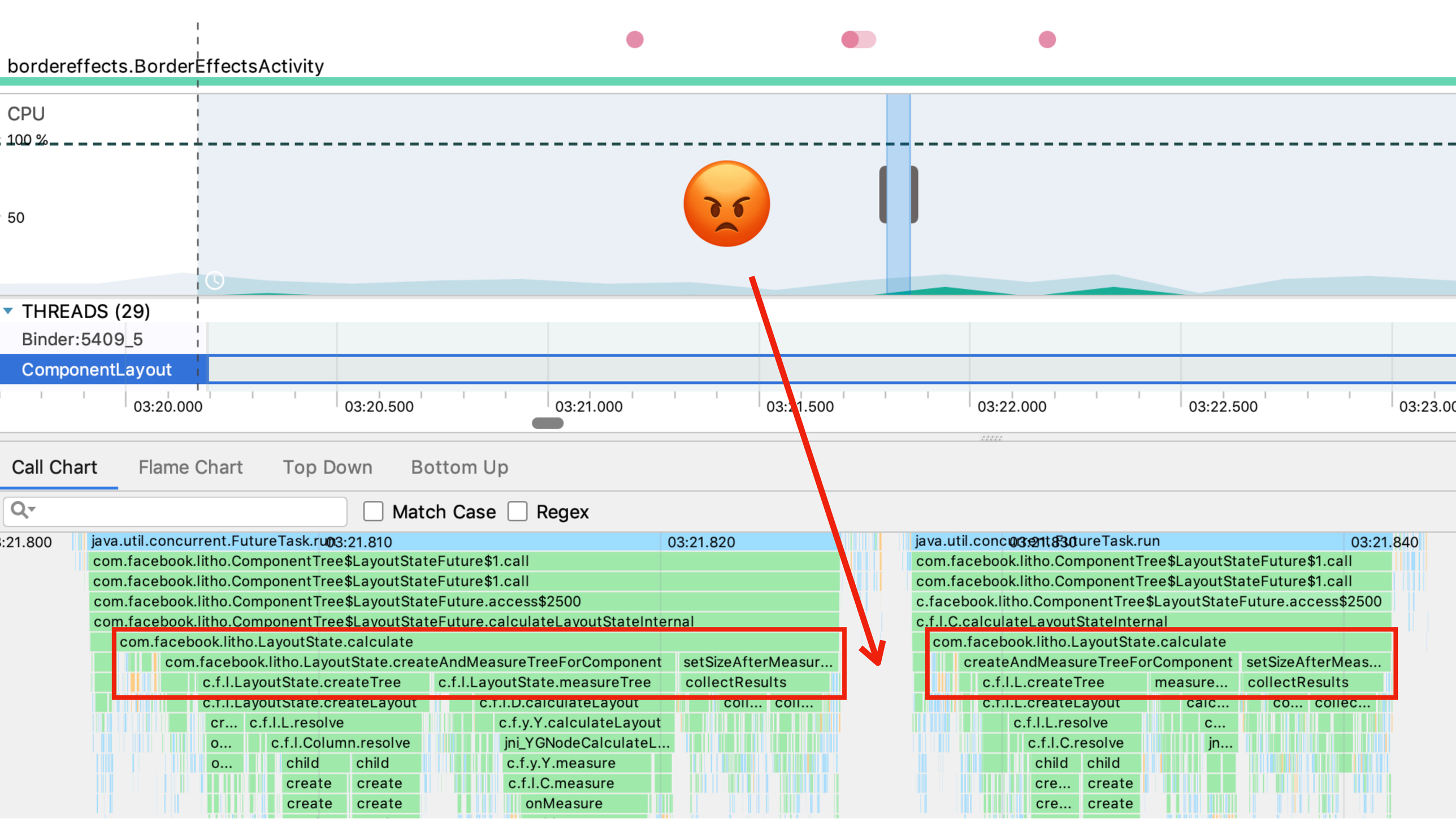
```
@OnUpdateState
fun changeColor(color: StateValue<Color>, @Param newColor: Color) {
  color.set(newColor)
}

@OnUpdateState
fun changeName(name: StateValue<String>, @Param newName: String) {
  name.set(newName)
}
```

```
@OnUpdateState
fun changeColor(color: StateValue<Color>, @Param newColor: Color) {
  color.set(newColor)
}

@OnUpdateState
fun changeName(name: StateValue<String>, @Param newName: String) {
  name.set(newName)
}

...

RootComponent.changeColor(c, Color.RED)
RootComponent.changeName(c, "IronMan")
```


bordereffects.BorderEffectsActivity
CPU
100 %
50
THREADS (29)
Binder:5409_5
ComponentLayout
03:20.000
03:20.500
03:21.000
03:21.500
03:22.000
03:22.500
Call Chart
Flame Chart
Top Down
Bottom Up
Match Case
Regex
03:21.800
java.util.concurrent.FutureTask.run
03:21.810
03:21.820
03:21.830
java.util.concurrent.FutureTask.run
03:21.840
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture.access$2500
com.facebook.litho.ComponentTree$LayoutStateFuture.calculateLayoutStateInternal
com.facebook.litho.LayoutState.calculate
com.facebook.litho.LayoutState.createAndMeasureTreeForComponent
setSizeAfterMeasur...
c.f.l.LayoutState.createTree
c.f.l.LayoutState.measureTree
collectResults
c.f.l.LayoutState.createLayout
c.f.l.D.calculateLayout
coll...
coll...
cr...
c.f.l.L.resolve
c.f.y.Y.calculateLayout
o...
c.f.l.Column.resolve
jni_YGNodeCalculateL...
o...
child
child
c.f.y.Y.measure
create
create
c.f.l.C.measure
create
create
onMeasure
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
c.facebook.litho.ComponentTree$LayoutStateFuture.access$2500
c.f.l.C.calculateLayoutStateInternal
com.facebook.litho.LayoutState.calculate
createAndMeasureTreeForComponent
setSizeAfterMeas...
c.f.l.L.createTree
measure...
collectResults
c.f.l.L.createLayout
calc...
co...
collec...
c.f.l.L.resolve
c...
c.f.l.C.resolve
jn...
child
child
cre...
create
cre...
create

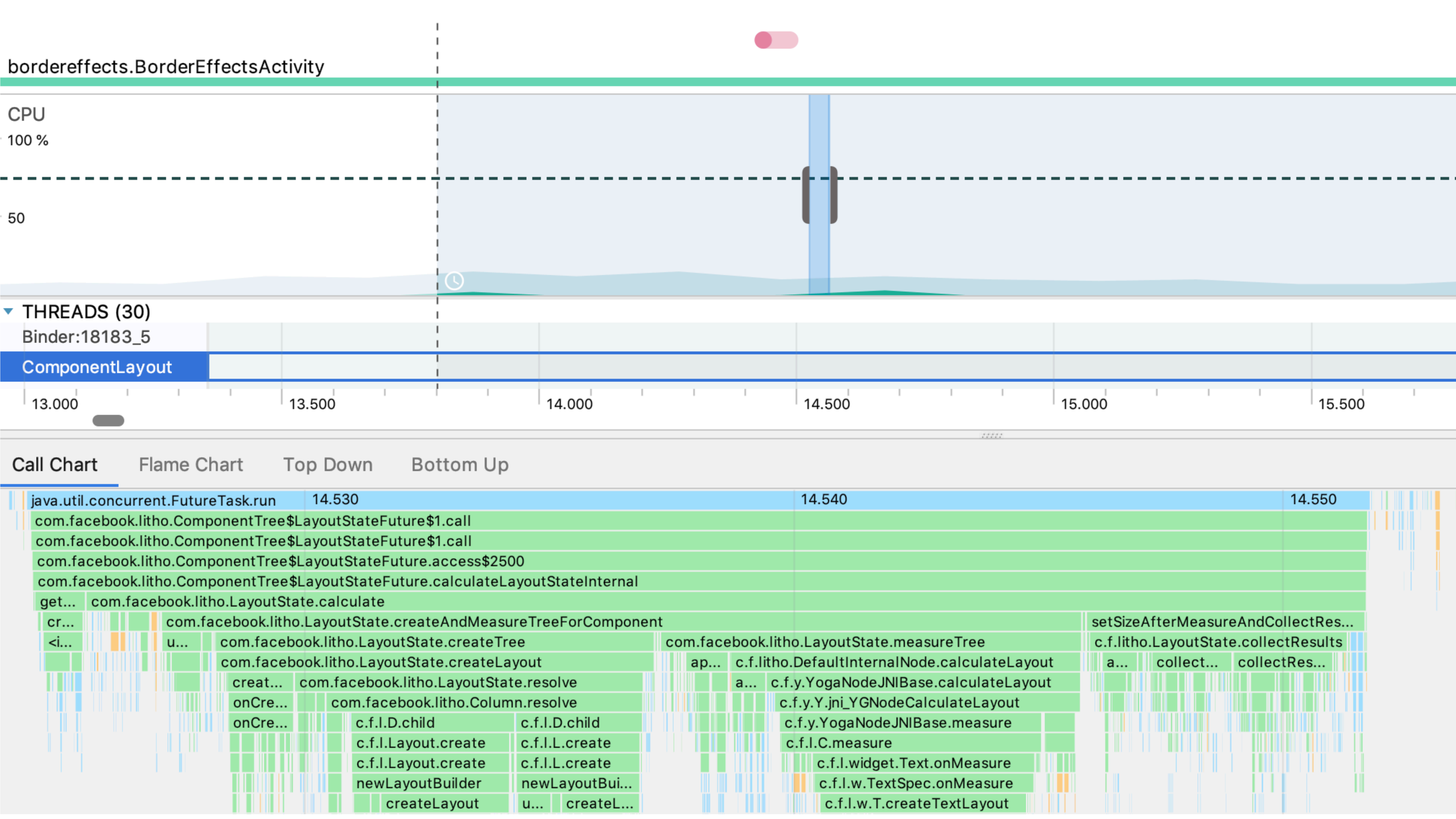

bordereffects.BorderEffectsActivity
CPU
100 %
50
THREADS (29)
Binder:5409_5
ComponentLayout
03:20.000
03:20.500
03:21.000
03:21.500
03:22.000
03:22.500
Call Chart
Flame Chart
Top Down
Bottom Up
Match Case
Regex
03:21.800
03:21.810
03:21.820
03:21.830
03:21.840
java.util.concurrent.FutureTask.run
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture.access$2500
com.facebook.litho.ComponentTree$LayoutStateFuture.calculateLayoutStateInternal
com.facebook.litho.LayoutState.calculate
com.facebook.litho.LayoutState.createAndMeasureTreeForComponent
setSizeAfterMeasur...
c.f.l.LayoutState.createTree
c.f.l.LayoutState.measureTree
collectResults
c.f.l.LayoutState.createLayout
c.f.l.D.calculateLayout
cr...
c.f.l.L.resolve
c.f.y.Y.calculateLayout
o...
c.f.l.Column.resolve
jni_YGNodeCalculateL...
child
child
c.f.y.Y.measure
create
create
c.f.l.C.measure
onMeasure
java.util.concurrent.FutureTask.run
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
c.facebook.litho.ComponentTree$LayoutStateFuture.access$2500
c.f.l.C.calculateLayoutStateInternal
com.facebook.litho.LayoutState.calculate
createAndMeasureTreeForComponent
setSizeAfterMeas...
c.f.l.L.createTree
measure...
collectResults
c.f.l.L.createLayout
calc...
co...
collec...
c.f.l.L.resolve
c...
c.f.l.C.resolve
jn...
child
child
cre...
create
cre...
create

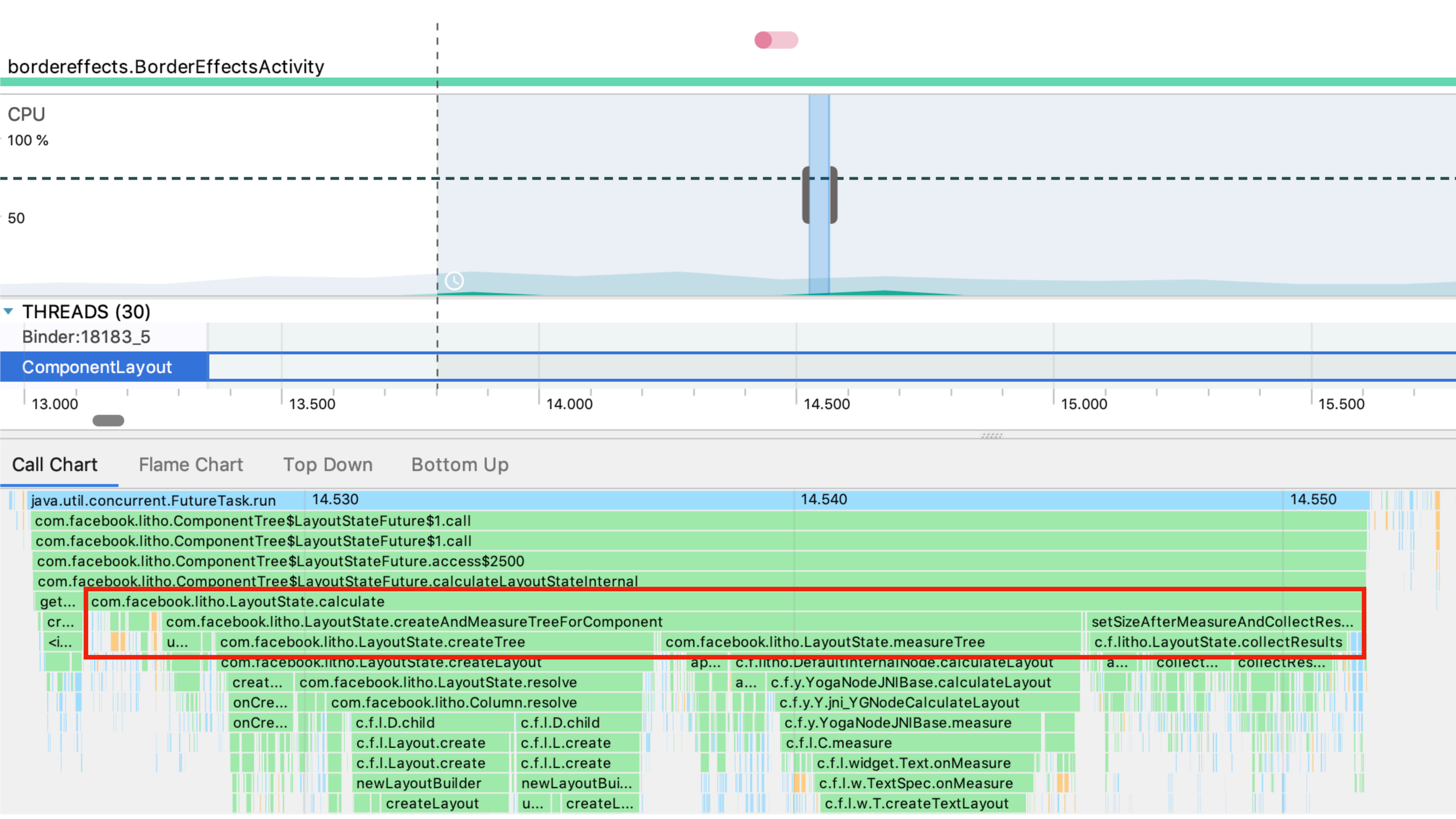

bordereffects.BorderEffectsActivity
CPU
100 %
50
THREADS (29)
Binder:5409_5
ComponentLayout
03:20.000
03:20.500
03:21.000
03:21.500
03:22.000
03:22.500
Call Chart
Flame Chart
Top Down
Bottom Up
Match Case
Regex
03:21.800
java.util.concurrent.FutureTask.run
03:21.810
03:21.820
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture.access$2500
com.facebook.litho.ComponentTree$LayoutStateFuture.calculateLayoutStateInternal
com.facebook.litho.LayoutState.calculate
com.facebook.litho.LayoutState.createAndMeasureTreeForComponent
setSizeAfterMeasur...
c.f.l.LayoutState.createTree
c.f.l.LayoutState.measureTree
collectResults
c.f.l.LayoutState.createLayout
c.f.l.D.calculateLayout
cr...
c.f.l.L.resolve
c.f.y.Y.calculateLayout
o...
c.f.l.Column.resolve
jni_YGNodeCalculateL...
child
child
c.f.y.Y.measure
create
create
c.f.l.C.measure
onMeasure
java.util.concurrent.FutureTask.run
03:21.830
03:21.840
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
c.facebook.litho.ComponentTree$LayoutStateFuture.access$2500
c.f.l.C.calculateLayoutStateInternal
com.facebook.litho.LayoutState.calculate
createAndMeasureTreeForComponent
setSizeAfterMeas...
c.f.l.L.createTree
measure...
collectResults
c.f.l.L.createLayout
c.f.l.L.resolve
c.f.l.C.resolve
child
child
cre...
create
cre...
create


bordereffects.BorderEffectsActivity
CPU
100 %
50
THREADS (29)
Binder:5409_5
ComponentLayout
03:20.000
03:20.500
03:21.000
03:22.000
03:22.500
Call Chart
Flame Chart
Top Down
Bottom Up
Match Case
Regex
java.util.concurrent.FutureTask.run
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture.access$2500
com.facebook.litho.ComponentTree$LayoutStateFuture.calculateLayoutStateInternal
com.facebook.litho.LayoutState.calculate
com.facebook.litho.LayoutState.createAndMeasureTreeForComponent
c.f.l.LayoutState.createTree
c.f.l.LayoutState.measureTree
collectResults
c.f.l.L.createTree
measure...
createAndMeasureTreeForComponent
setSizeAfterMeasur...
setSizeAfterMeas...
c.f.l.L.resolve
c.f.l.Column.resolve
c.f.l.C.resolve
c.f.y.Y.calculateLayout
jni_YGNodeCalculateL...
c.f.y.Y.measure
c.f.l.C.measure
onMeasure
child
create

```
@OnUpdateState
fun changeColor(color: StateValue<Color>, @Param newColor: Color) {
  color.set(newColor)
}

@OnUpdateState
fun changeName(name: StateValue<String>, @Param newName: String) {
  name.set(newName)
}

...

RootComponent.changeColor(c, Color.RED)
RootComponent.changeName(c, "IronMan")
```

```
@OnUpdateState
fun changeHero(
    color: StateValue<Color>, name: StateValue<String>,
    @Param newColor: Color, @Param newName: String) {
  color.set(newColor)
  name.set(newName)
}



...

RootComponent.changeHero(c, Color.RED, "IronMan")
```


bordereffects.BorderEffectsActivity
CPU
100 %
50
THREADS (30)
Binder:18183_5
ComponentLayout
13.000
13.500
14.000
14.500
15.000
15.500
Call Chart
Flame Chart
Top Down
Bottom Up
java.util.concurrent.FutureTask.run
14.530
14.540
14.550
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture.access$2500
com.facebook.litho.ComponentTree$LayoutStateFuture.calculateLayoutStateInternal
get...
com.facebook.litho.LayoutState.calculate
cr...
com.facebook.litho.LayoutState.createAndMeasureTreeForComponent
setSizeAfterMeasureAndCollectRes...
<i...
u...
com.facebook.litho.LayoutState.createTree
com.facebook.litho.LayoutState.measureTree
c.f.litho.LayoutState.collectResults
com.facebook.litho.LayoutState.createLayout
ap...
c.f.litho.DefaultInternalNode.calculateLayout
a...
collect...
collectRes...
creat...
com.facebook.litho.LayoutState.resolve
a...
c.f.y.YogaNodeJNIBase.calculateLayout
onCre...
com.facebook.litho.Column.resolve
c.f.y.Y.jni_YGNodeCalculateLayout
onCre...
c.f.l.D.child
c.f.l.D.child
c.f.y.YogaNodeJNIBase.measure
c.f.l.Layout.create
c.f.l.L.create
c.f.l.C.measure
c.f.l.Layout.create
c.f.l.L.create
c.f.l.widget.Text.onMeasure
newLayoutBuilder
newLayoutBui...
c.f.l.w.TextSpec.onMeasure
createLayout
u...
createL...
c.f.l.w.T.createTextLayout


bordereffects.BorderEffectsActivity
CPU
100 %
50
THREADS (30)
Binder:18183_5
ComponentLayout
13.000
13.500
14.000
14.500
15.000
15.500
Call Chart
Flame Chart
Top Down
Bottom Up
java.util.concurrent.FutureTask.run
14.530
14.540
14.550
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture$1.call
com.facebook.litho.ComponentTree$LayoutStateFuture.access$2500
com.facebook.litho.ComponentTree$LayoutStateFuture.calculateLayoutStateInternal
get...
com.facebook.litho.LayoutState.calculate
cr...
com.facebook.litho.LayoutState.createAndMeasureTreeForComponent
setSizeAfterMeasureAndCollectRes...
<i...
u...
com.facebook.litho.LayoutState.createTree
com.facebook.litho.LayoutState.measureTree
c.f.litho.LayoutState.collectResults
com.facebook.litho.LayoutState.createLayout
ap...
c.f.litho.DefaultInternalNode.calculateLayout
a...
collect...
collectRes...
creat...
com.facebook.litho.LayoutState.resolve
a...
c.f.y.YogaNodeJNIBase.calculateLayout
onCre...
com.facebook.litho.Column.resolve
c.f.y.Y.jni_YGNodeCalculateLayout
onCre...
c.f.l.D.child
c.f.l.D.child
c.f.y.YogaNodeJNIBase.measure
c.f.l.Layout.create
c.f.l.L.create
c.f.l.C.measure
c.f.l.Layout.create
c.f.l.L.create
c.f.l.widget.Text.onMeasure
newLayoutBuilder
newLayoutBui...
c.f.l.w.TextSpec.onMeasure
createLayout
u...
createL...
c.f.l.w.T.createTextLayout

What if State is not tied to rendering?

```
@OnCreateLayout
fun onCreateLayout(
    c: ComponentContext,
    @State count: Int
): Component {
  return Column.create(c)
      // ...
      .build()
}
```

```
@OnCreateLayout
fun onCreateLayout(
    c: ComponentContext,
    @State count: Int,
    @State(canUpdateLazily = true) step: Int
): Component {
  return Column.create(c)
      // ...
      .child(
          TextInput.create(c).initialText("$step"))
      .build()
}
```

```
@State(canUpdateLazily = true) step: Int


RootComponent.lazyUpdateStep(c, value)
```

```
@State(canUpdateLazily = true) step: Int


RootComponent.lazyUpdateStep(c, value)
```

GOTCHAS

GOTCHAS

- Batch your State updates

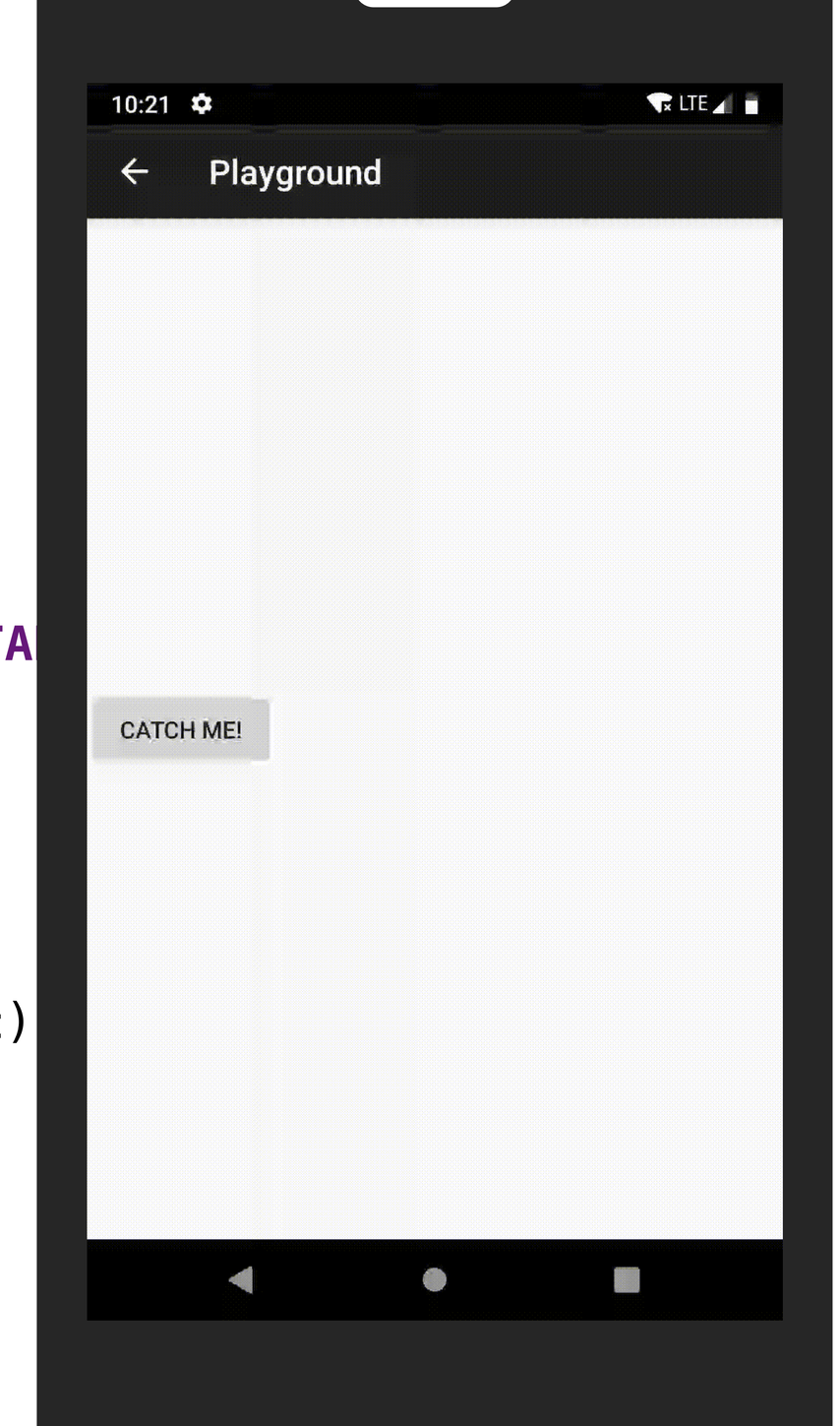
GOTCHAS

- ✦ Batch your State updates
- ✦ Use lazy State updates where possible

ANIMATE ALL THE THINGS

```
@OnCreateLayout
fun onCreateLayout(
    c: ComponentContext,
    @State alignToEnd: Boolean): Component {

  val align = if (alignToEnd) FLEX_END else FLEX_START
  return Column.create(c)
      .justifyContent(CENTER)
      .child(
          Button.create(c)
              .text("Catch me!")
              .alignSelf(align)
              .clickHandler(RootComponent.onClick(c))
      )
      .build()
}
```

```
@OnCreateLayout
fun onCreateLayout(
    c: ComponentContext,
    @State alignToEnd: Boolean): Component {

  val align = if (alignToEnd) FLEX_END else FLEX_STA
  return Column.create(c)
      .justifyContent(CENTER)
      .child(
          Button.create(c)
              .text("Catch me!")
              .alignSelf(align)
              .clickHandler(RootComponent.onClick(c)
      )
      .build()
}
```



```
@OnCreateLayout
fun onCreateLayout(
    c: ComponentContext,
    @State alignToEnd: Boolean): Component {

  val align = if (alignToEnd) FLEX_END else FLEX_START
  return Column.create(c)
      .justifyContent(CENTER)
      .child(
          Button.create(c)
              .text("Catch me!")
              .transitionKey("button")
              .alignSelf(align)
              .clickHandler(RootComponent.onClick(c))
      )
      .build()
}
```

```
@OnCreateLayout
fun onCreateLayout(
    c: ComponentContext,
    @State alignToEnd: Boolean): Component {

  val align = if (alignToEnd) FLEX_END else FLEX_START
  return Column.create(c)
      .justifyContent(CENTER)
      .child(
          Button.create(c)
              .text("Catch me!")
              .transitionKey("button")
              .alignSelf(align)
              .clickHandler(RootComponent.onClick(c))
      )
      .build()
}

@OnCreateTransition
fun onCreateTransition(c: ComponentContext): Transition {
  return Transition.create("button")
      .animate(AnimatedProperties.X)
}
```

```
@OnCreateLayout
fun onCreateLayout(
    c: ComponentContext,
    @State alignToEnd: Boolean): Component {

  val align = if (alignToEnd) FLEX_END else FLEX_START
  return Column.create(c)
      .justifyContent(CENTER)
      .child(
          Button.create(c)
              .text("Catch me!")
              .transitionKey("button")
              .alignSelf(align)
              .clickHandler(RootComponent.onClick(c))
      )
      .build()
}

@OnCreateTransition
fun onCreateTransition(c: ComponentContext): Transition {
  return Transition.create("button")
      .animate(AnimatedProperties.X)
}
```

```
@OnCreateTransition
fun onCreateTransition(c: ComponentContext): Transition {
  return Transition.create("button")
      .animate(AnimatedProperties.X)
}
```

```
@OnCreateTransition
fun onCreateTransition(c: ComponentContext): Transition {
  return Transition.allLayout()
}
```

ANIMATIONS

ANIMATIONS

`onCreateTransition` is called on every `onCreateLayout`

ANIMATIONS

`onCreateTransition` is called on every `onCreateLayout`

Even if you don't need to animate that specific UI update

ANIMATIONS

`onCreateTransition` is called on every `onCreateLayout`

Even if you don't need to animate that specific UI update

More granular control?

```
@OnCreateTransition
fun onCreateTransition(c: ComponentContext): Transition {
  return Transition.allLayout()
}
```

```
@OnCreateTransition
fun onCreateTransition(c: ComponentContext,
    @Prop prop: Diff<String>,
    @State state: Diff<Boolean>): Transition? {

  return if (canAnimate(prop, state)) Transition.allLayout() else null
}
```

```
@OnCreateTransition
fun onCreateTransition(c: ComponentContext,
    @Prop prop: Diff<String>,
    @State state: Diff<Boolean>): Transition? {

  return if (canAnimate(prop, state)) Transition.allLayout() else null
}
```

```java
public final class Diff<T> {
  public T getPrevious()
  public T getNext()
}
```

```kotlin
@OnCreateTransition
fun onCreateTransition(c: ComponentContext,
    @Prop prop: Diff<String>,
    @State state: Diff<Boolean>): Transition? {

  return if (canAnimate(prop, state)) Transition.allLayout() else null
}
```

GOTCHAS

GOTCHAS

Use `Diff<T>` for more granular control over transitions

GRADUAL ADOPTION

GRADUAL ADOPTION

GRADUAL ADOPTION

Can't migrate the whole screen at once? Do step by step:

GRADUAL ADOPTION

Can't migrate the whole screen at once? Do step by step:

- Complex UI parts can be replaced with a LithoView and underlying Component structure

GRADUAL ADOPTION

Can't migrate the whole screen at once? Do step by step:

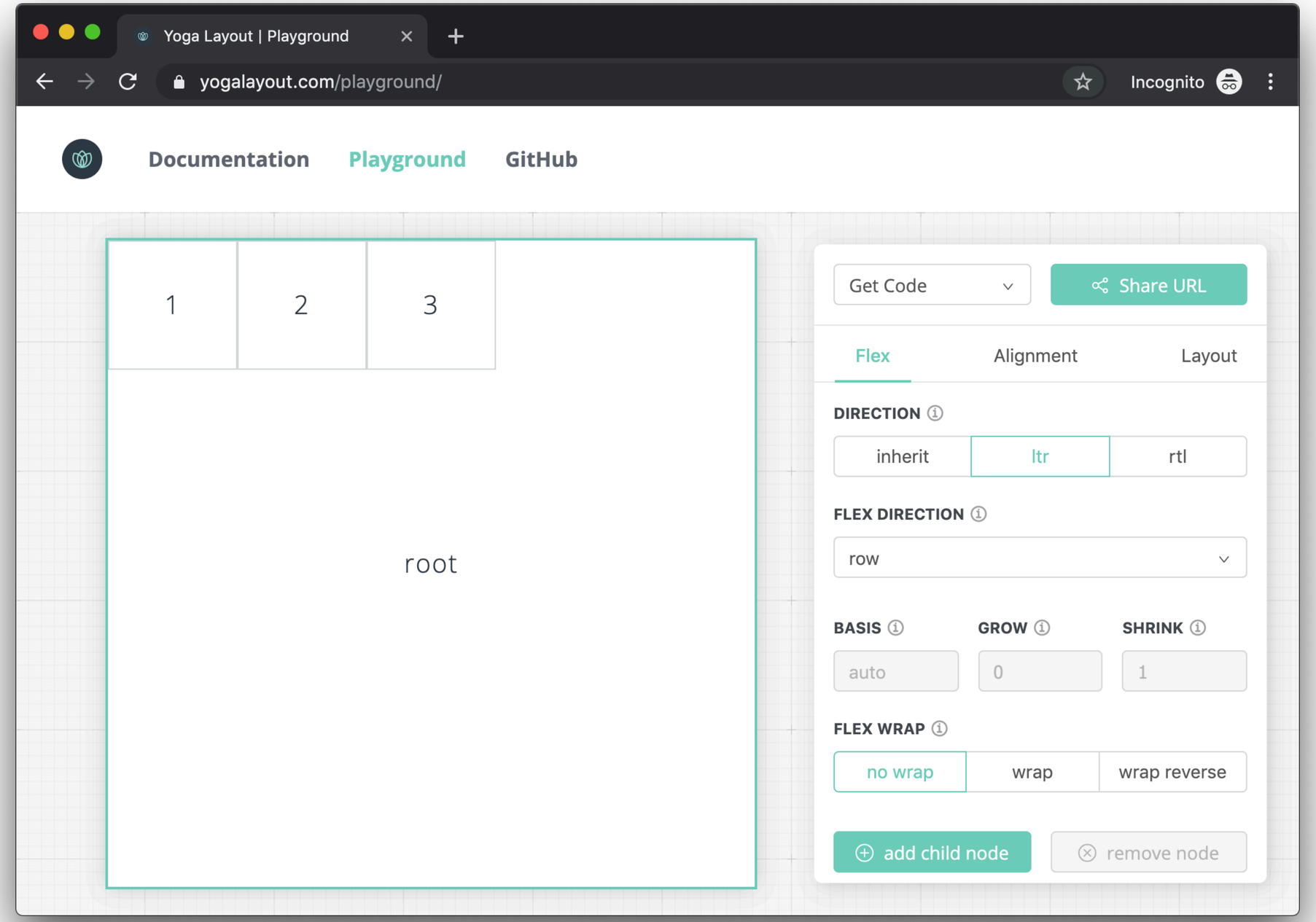
- Complex UI parts can be replaced with a LithoView and underlying Component structure
- Custom Views can be wrapped in `MountSpec`s

SOMETHING DOESN'T WORK!

TOOLS

TOOLS

Yoga Playground



yogalayout.com/playground

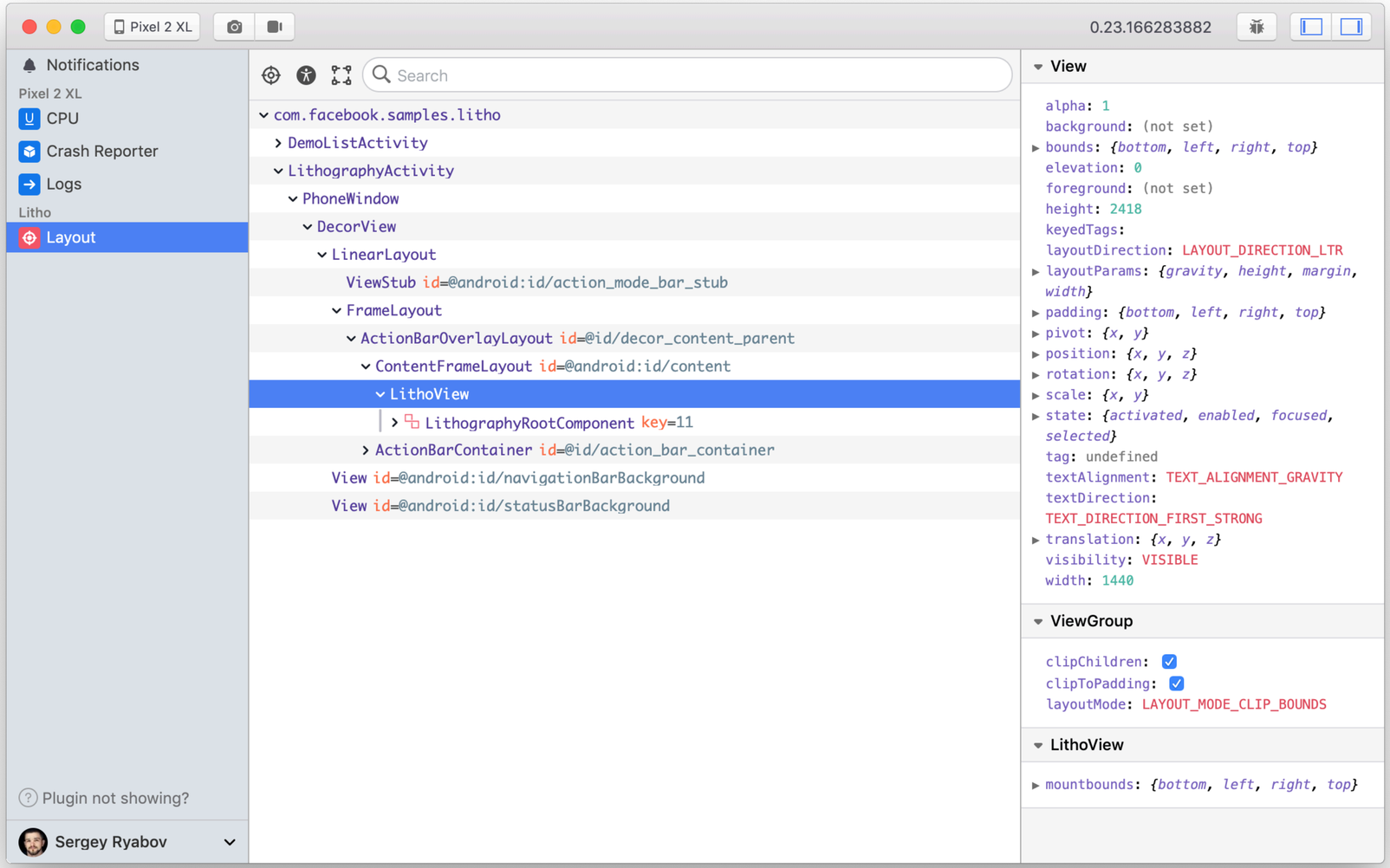
TOOLS

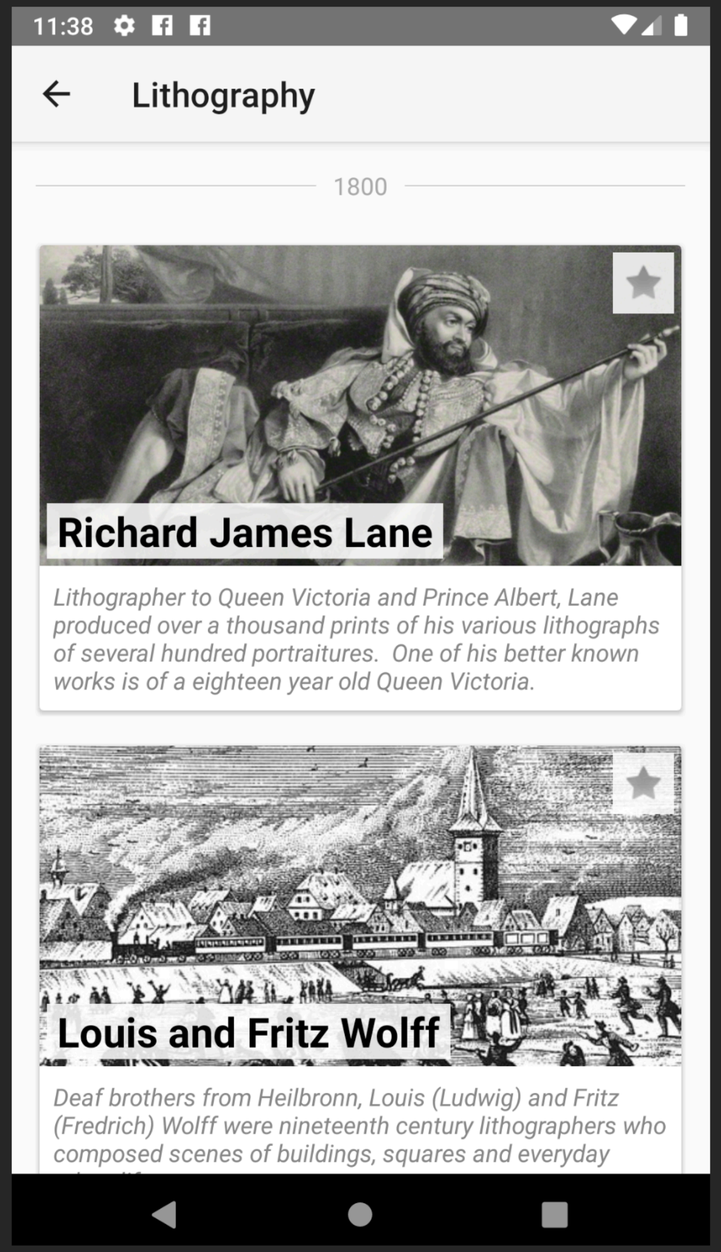
Yoga Playground

TOOLS

Yoga Playground

Flipper + Layout plugin





fbflipper.com

TOOLS

Yoga Playground

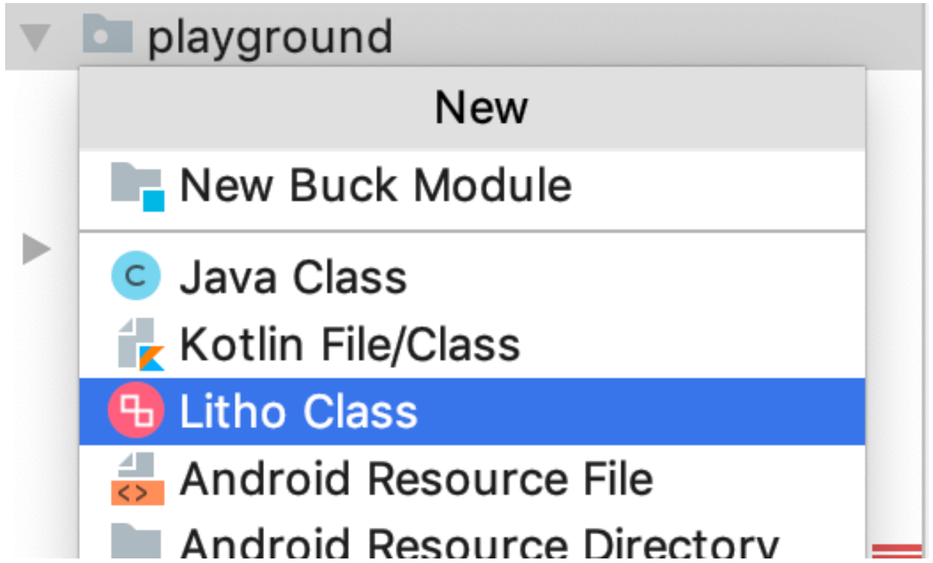
Flipper + Layout plugin

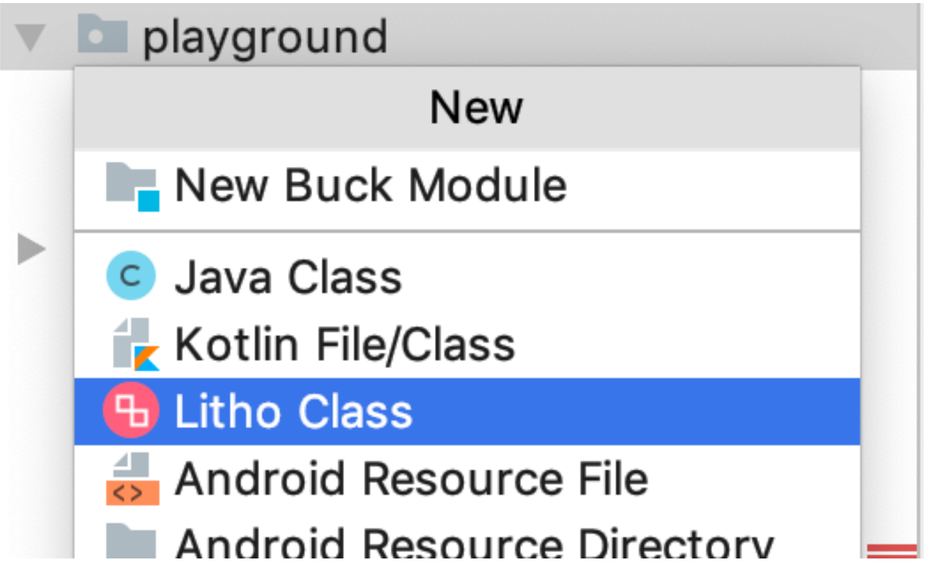
TOOLS

Yoga Playground

Flipper + Layout plugin

Litho IntelliJ Plugin (Beta)


playground
New
New Buck Module
Java Class
Kotlin File/Class
Litho Class
Android Resource File
Android Resource Directory




```
@LayoutSpec
class ExampleSpec {

  @OnCreateInitialState
  static void onCreateInitialState(
      ComponentContext c
  ) {
    // TODO: remove this method if not needed. Set an initial value for a state https://fblitho.com
  }

  @OnCreateLayout
  static Component onCreateLayout(
      ComponentContext c
  ) {
    // TODO: describe your UI with existing components https://fblitho.com/docs/layout-specs
    return Column.create(c)
        .build();
  }
}
```

github.com/facebook/litho/tree/master/litho-intellij-plugin

TOOLS

Yoga Playground

Flipper + Layout plugin

Litho IntelliJ Plugin (Beta)

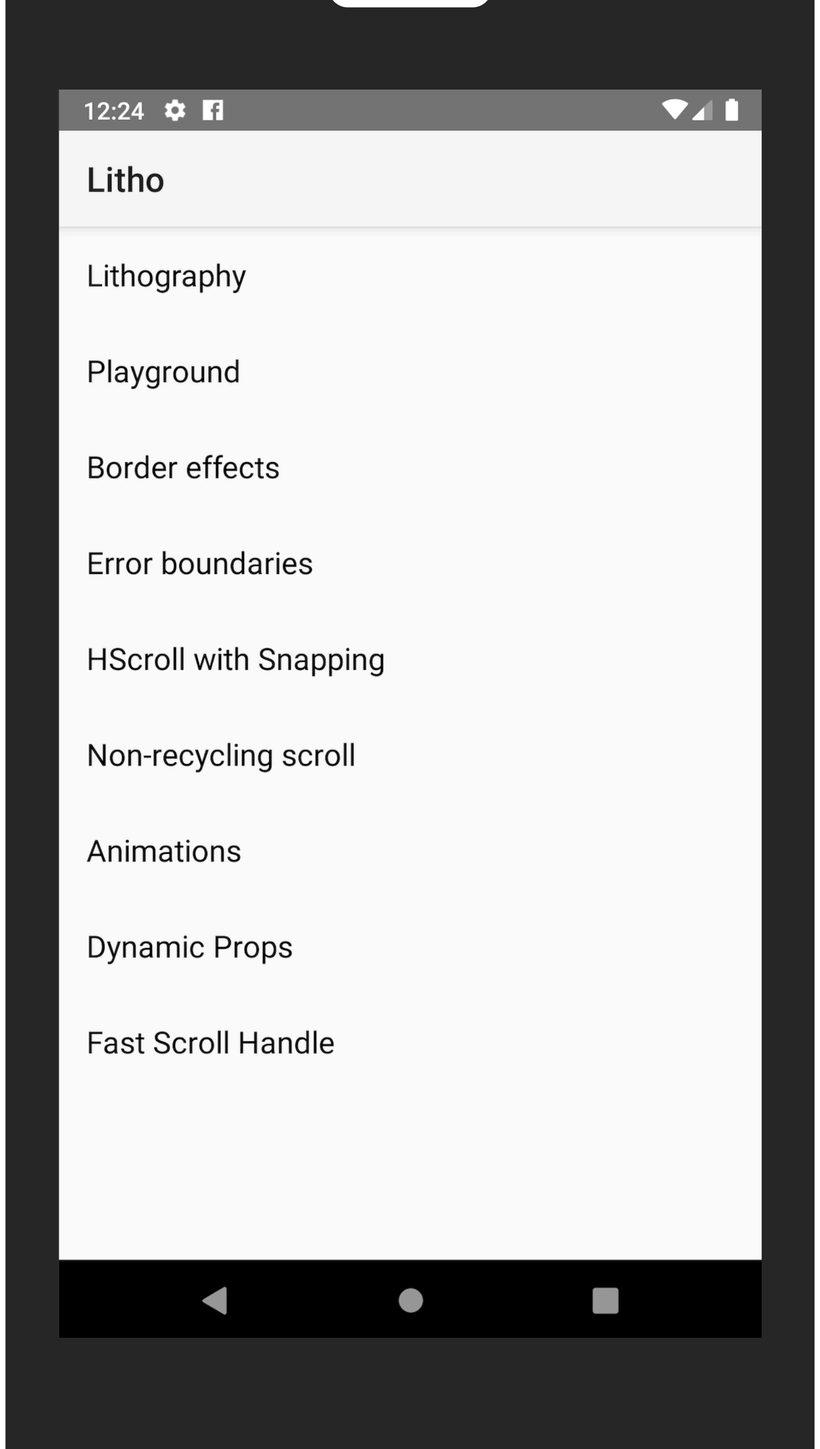
TOOLS

Yoga Playground

Flipper + Layout plugin

Litho IntelliJ Plugin (Beta)

Lithography Sample app


12:24
Litho
Lithography
Playground
Border effects
Error boundaries
HScroll with Snapping
Non-recycling scroll
Animations
Dynamic Props
Fast Scroll Handle

WHAT IS LITHO

- ✦ Asynchronous layout
- ✦ Declarative API
- ✦ Flatter view hierarchies
- ✦ Fine-grained recycling

RESOURCES

- Official Docs: fblitho.com
- Lithography Sample app: github.com/facebook/litho/tree/master/sample
- Litho Codelabs (preview): github.com/facebook/litho/tree/master/codelabs
- Dive into Litho Threading Model: youtu.be/78BUH1LZaZ4

THAT'S ALL FOLKS

THAT'S NOT ALL FOLKS

THAT'S NOT ALL FOLKS

```
Text(text = "Hello, Kotlin World!", textSize = 20.sp)
```

```
Column {
  +Text(text = "Hello, Kotlin World!", textSize = 20.sp)
  +Text(
      text = "with ❤️ from London",
      textStyle = Typeface.ITALIC)
}
```

```
Padding(all = 16.dp) {
  Column {
    +Text(text = "Hello, Kotlin World!", textSize = 20.sp)
    +Text(
        text = "with ❤️ from London",
        textStyle = Typeface.ITALIC)
  }
}
```

```
Clickable(onClick = { }) {
  Padding(all = 16.dp) {
    Column {
      +Text(text = "Hello, Kotlin World!", textSize = 20.sp)
      +Text(
          text = "with ❤️ from London",
          textStyle = Typeface.ITALIC)
    }
  }
}
```

```
val counter by useState { 1 }

Clickable(onClick = { updateState { counter.value = counter.value + 1 } }) {
  Padding(all = 16.dp) {
    Column {
      +Text(text = "Hello, Kotlin World!", textSize = 20.sp)
      +Text(
          text = "with ${"❤️".repeat(counter.value)} from London",
          textStyle = Typeface.ITALIC)
    }
  }
}
```

```
setContent {
  val counter by useState { 1 }

  Clickable(onClick = { updateState { counter.value = counter.value + 1 } }) {
    Padding(all = 16.dp) {
      Column {
        +Text(text = "Hello, Kotlin World!", textSize = 20.sp)
        +Text(
            text = "with ${"❤️".repeat(counter.value)} from London",
            textStyle = Typeface.ITALIC)
      }
    }
  }
}
```

```
class PlaygroundActivity : Activity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    setContent {
      val counter by useState { 1 }

      Clickable(onClick = { updateState { counter.value = counter.value + 1 } }) {
        Padding(all = 16.dp) {
          Column {
            +Text(text = "Hello, Kotlin World!", textSize = 20.sp)
            +Text(
                text = "with ${"❤️".repeat(counter.value)} from London",
                textStyle = Typeface.ITALIC)
          }
        }
      }
    }
  }
}
```

THANK YOU

Sergey Ryabov
@colriot

```
Text.create(
        c,
        android.R.attr.buttonStyle,
        0
)
```

```
Text.create(
        c,
        0,
        R.style.Widget_AppCompat_Button_Colored
)
```