# 声明式开发范式组件

# 前言

- 组件是UI构建与显示的最小单位，如列表、网格、按钮、单选框、进度条、文本等。开发者通过多种组件的组合，构建出满足自身应用诉求的完整界面。
- 本章主要讲述基于ArkTS的声明式开发范式组件，包括基础组件、容器组件和媒体组件。

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

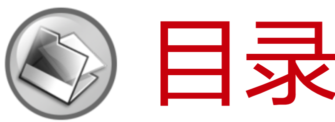
HUAWEI

# 课程目标

- 学完本课程后，您将能够：
  - 区分基础组件和容器组件；
  - 了解各个组件的功能；
  - 使用组件进行UI界面的搭建；
  - 熟悉组件的核心属性和事件等。

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI



# 目录

1. **ArkUI介绍**
2. 布局组件
3. 常用组件介绍

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# ArkUI简介

- ArkUI 是一套声明式开发框架，它具备简洁自然的 UI 信息语法、丰富的 UI 组件、多维状态管理，以及实时多维度预览等能力，帮助开发者提升应用开发效率，并能在多种设备实现生动而流畅的用户体验。

```
build() {
Text('Hello World')
    .fontSize(40)
}
```

Hello
World

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

# ArkUI基本概念

**布局：**布局是UI的必要元素，它定义了组件在界面中的位置，也是装载组件的容器。
例如，此处框出区域为轮播布局。

**页面路由和组件导航：**应用可能包含多个页面，可通过页面路由实现页面间的跳转。
例如，此处展示应用首页以及具体商品页面。

**组件：**组件是UI的必要元素，形成了在界面中的样子。由框架直接提供的称为系统组件，由开发者定义的称为自定义组件。
例如，此处框出搜索框组件与图片组件。

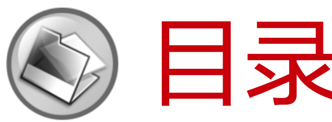
**交互事件：**交互事件是UI和用户交互的必要元素。方舟开发框架提供了多种交互事件。
例如，此处框出点击按钮会触发其绑定的点击事件。

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# ArkUI优势

- 在开发一款新应用时，推荐采用声明式开发范式来构建UI，主要基于以下几点考虑：
  - 开发效率：声明式开发范式更接近自然语义的编程方式，数据驱动UI变化，让开发者更专注自身业务逻辑的处理。开发高效简洁。
  - 性能优越：声明式UI前端和UI后端分层，语言编译器和运行时的优化。
  - 生态容易快速推进：能够借力主流语言生态快速推进，语言相对中立友好，有相应的标准组织可以逐步演进。

Copyright © Huawei Technologies Co., Ltd. All rights reserved.



# 目录


1. ArkUI介绍
2. **布局组件**
3. 常用组件介绍


Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

# 布局说明

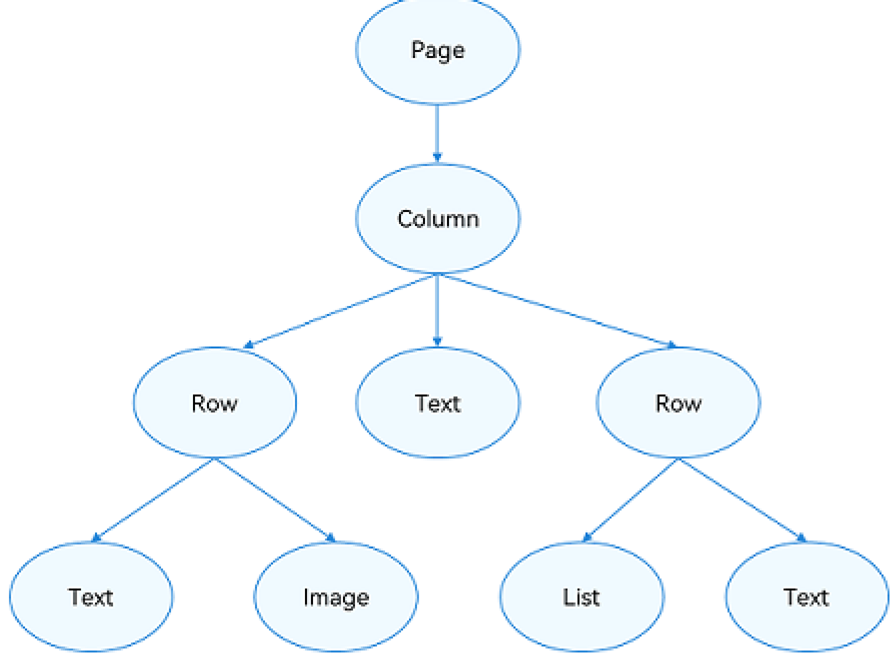
- 一个页面的基本元素包含标题区域、文本区域、图片区域等，每个基本元素内还可以包含多个子元素，开发者根据需求还可以添加按钮、开关、进度条等组件。在构建页面布局时，需要对每个基本元素思考以下几个问题：

| | | |
|---|---|---|
| 该元素的尺寸和排列位置 | 是否有重叠的元素 | 是否需要设置对齐、内间距或者边界 |
| 是否包含子元素及其排列位置 | 是否需要容器组件及其类型 | 将页面中的元素分解之后再对每个基本元素按顺序实现，以减少多层嵌套造成的视觉混乱和逻辑混乱 |

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# 布局结构

- 布局通常为分层结构，一个常见的页面结构如下所示：
- 为实现下图效果，开发者需要在页面中声明对应的元素。其中，Page表示页面的根节点，Column/Row等元素为系统组件。针对不同的页面结构，ArkUI提供了不同的布局组件来帮助开发者实现对应布局的效果，例如Row用于实现线性布局。




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

# 常用布局组件

- ArkUI框架提供了多种布局方式，除了基础的线性布局、层叠布局、弹性布局、相对布局、栅格布局外，也提供了相对复杂的列表、宫格、轮播。

| 布局 | 应用场景 |
| --- | --- |
| 线性布局（Row、Column） | 如果布局内子元素超过一个时，且能够以某种方式线性排列时优先考虑此布局。 |
| 层叠布局（Stack） | 组件需要有堆叠效果时优先考虑此布局。层叠布局的堆叠效果不会占用或影响其他同容器内子组件的布局空间。 |
| 弹性布局（Flex） | 弹性布局是与线性布局类似的布局方式。区别在于弹性布局默认能够使子组件压缩或拉伸。 |
| 相对布局（RelativeContainer） | 相对布局是在二维空间中的布局方式，不需要遵循线性布局的规则，布局方式更为自由。 |
| 栅格布局（GridRow、GridCol） | 栅格是多设备场景下通用的辅助定位工具，可将空间分割为有规律的栅格。 |
| 列表（List） | 使用列表可以高效地显示结构化、可滚动的信息。 |
| 网格（Grid） | 网格布局具有较强的页面均分能力、子元素占比控制能力。 |
| 轮播（Swiper） | 轮播组件通常用于实现广告轮播、图片预览等。 |

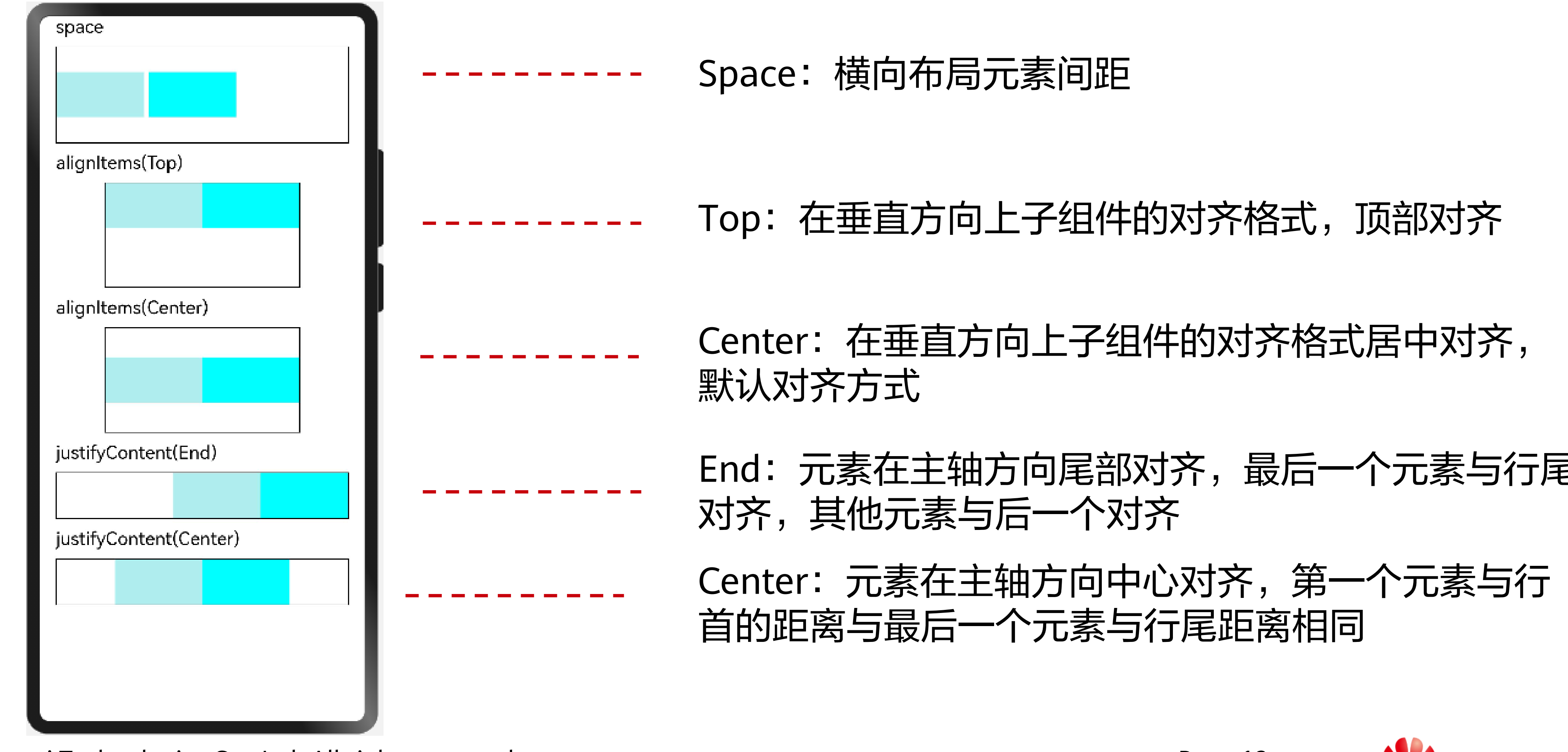
Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

# Row

- 沿水平方向布局容器。




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

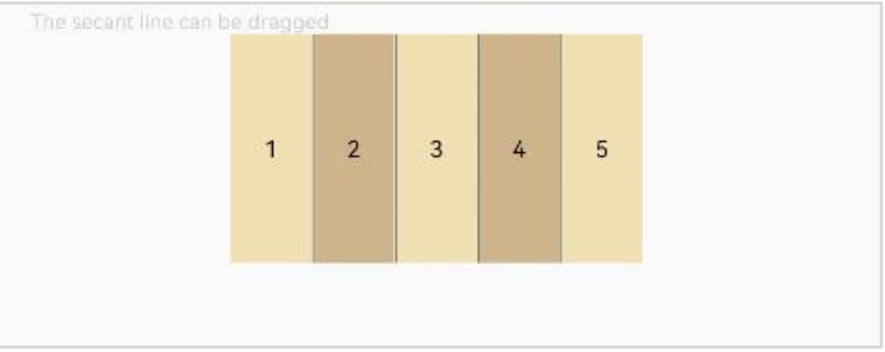


HUAWEI

# RowSplit

- 将子组件横向布局，并在每个子组件之间插入一根纵向的分割线。

```
@Entry
@Component
struct RowSplitExample {
  build() {
    Column() {
      Text('The second line can be dragged').fontSize(9).fontColor(0xCCCCCC).width('90%')
      RowSplit() {
        Text('1')
          .width('10%')
          .height(100)
          .backgroundColor(0xF5DEB3)
          .textAlign(TextAlign.Center)
        Text('2')
          .width('10%')
          .height(100)
          .backgroundColor(0xD2B48C)
          .textAlign(TextAlign.Center)
          ......
      }
      .resizeable(true)        // 分割线是否可拖拽，默认为false
      .width('90%').height(100)
    }.width('100%').margin({ top: 5 })
  }
}
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

# Column：参数space

- 沿垂直方向布局的容器。

```
@Entry
@Component
struct ColumnExample {
  build() {
    Column() {
      Text('space').fontSize(25).fontColor(0x000000).width('90%')
      Column({ space: 30 }) {     //纵向布局元素间距

Column().width('100%').height(30).backgroundColor(0x008000)

Column().width('100%').height(30).backgroundColor(0x0000FF)
      }.width('90%').height(100).border({ width: 1 })

    }
  }
}
```

Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

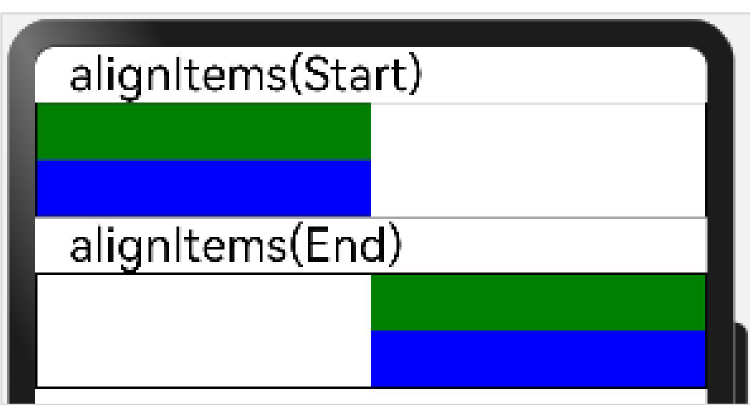
# Column：alignItems

- 设置子组件在水平方向上的对齐格式。

```
@Entry
@Component
struct ColumnExample {
  build() {
    Column() {
      Text('alignItems(Start)').fontSize(25).fontColor(0x000000).width('90%')
      Column() {
        Column().width('50%').height(30).backgroundColor(0x008000)
        Column().width('50%').height(30).backgroundColor(0x0000FF)
      }.alignItems(HorizontalAlign.Start).width('100%').border({ width: 1 })

      Text('alignItems(End)').fontSize(25).fontColor(0x000000).width('90%')
      Column() {
        Column().width('50%').height(30).backgroundColor(0x008000)
        Column().width('50%').height(30).backgroundColor(0x0000FF)
      }.alignItems(HorizontalAlign.End).width('100%').border({ width: 1 })

    }
  }
}
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

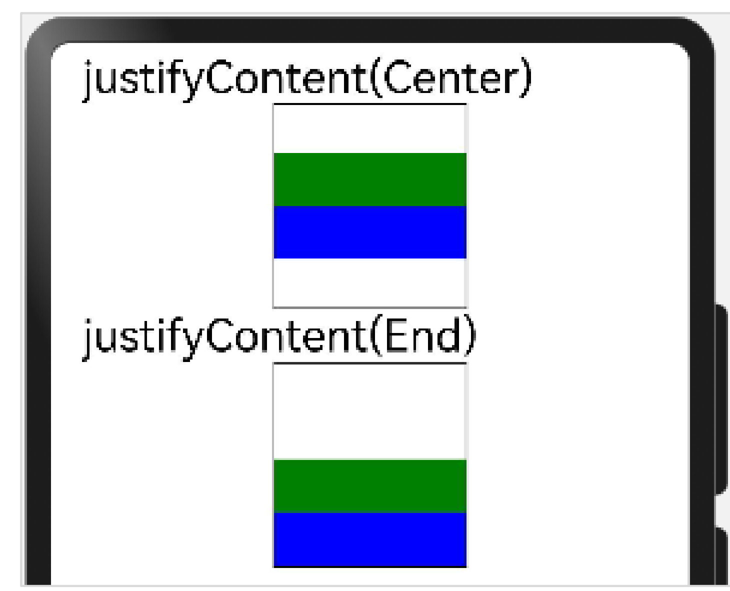
# Column：justifyContent

- 设置子组件在垂直方向上的对齐格式。

```
@Entry
@Component
struct ColumnExample {
  build() {
    Column() {

Text('justifyContent(Center)').fontSize(25).fontColor(0x000000).width('90%')
      Column() {
        Column().width('30%').height(30).backgroundColor(0x008000)
        Column().width('30%').height(30).backgroundColor(0x0000FF)
      }.height('15%').border({ width: 1 }).justifyContent(FlexAlign.Center)

      Text('justifyContent(End)').fontSize(25).fontColor(0x000000).width('90%')
      Column() {
        Column().width('30%').height(30).backgroundColor(0x008000)
        Column().width('30%').height(30).backgroundColor(0x0000FF)
      }.height('15%').border({ width: 1 }).justifyContent(FlexAlign.End)
    }.width('100%').padding({ top: 5 })
  }
}
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

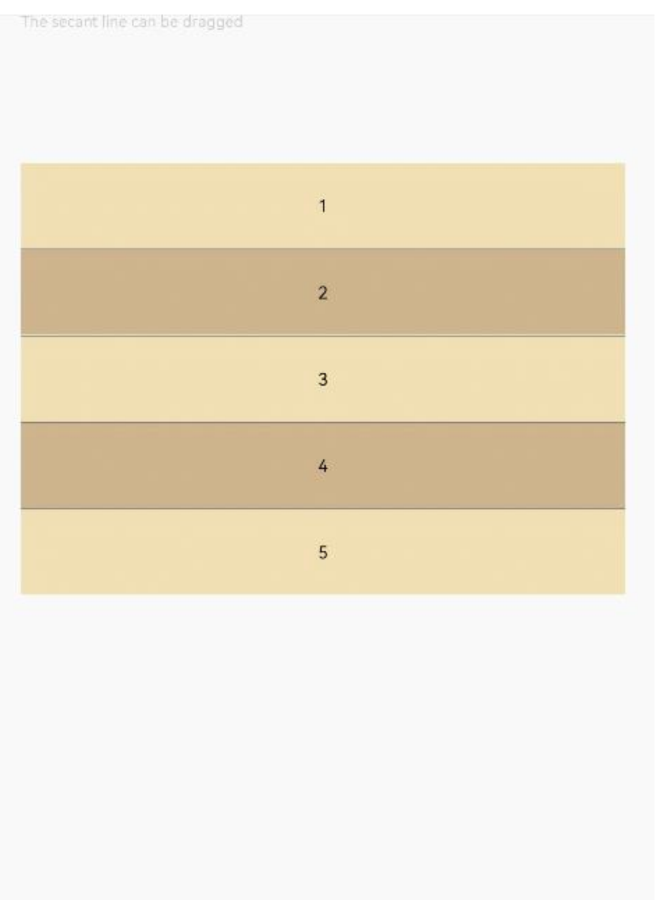


HUAWEI

# ColumnSplit

- 将子组件纵向布局，并在每个子组件之间插入一根横向的分割线。

```
@Entry
@Component
struct ColumnSplitExample {
  build() {
    Column(){
      Text('The secant line can be dragged').fontSize(9).fontColor(0xCCCCCC).width('90%')
      ColumnSplit() {
Text('1').width('100%').height(50).backgroundColor(0xF5DEB3).textAlign(TextAlign.Center)
Text('2').width('100%').height(50).backgroundColor(0xD2B48C).textAlign(TextAlign.Center)
Text('3').width('100%').height(50).backgroundColor(0xF5DEB3).textAlign(TextAlign.Center)
Text('4').width('100%').height(50).backgroundColor(0xD2B48C).textAlign(TextAlign.Center)
Text('5').width('100%').height(50).backgroundColor(0xF5DEB3).textAlign(TextAlign.Center)
      }
      .resizeable(true)        //分割线是否可拖拽，默认为false
      .width('90%').height('60%')
    }.width('100%')
  }
}
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

# Stack

- 堆叠容器，子组件按照顺序依次入栈，后一个子组件覆盖前一个子组件。

```
@Entry
@Component
struct StackExample {
  build() {
    Stack({ alignContent: Alignment.Bottom }) {
    //设置子组件在容器内的对齐方式
      Text('First child, show in bottom').width('90%').height('100%').backgroundColor(0xfcc800).align(Alignment.Top)
      Text('Second child, show in top').width('70%').height('60%').backgroundColor(0x62b230).align(Alignment.Top)
    }.width('100%').height(150).margin({ top: 5 })
  }
}
```

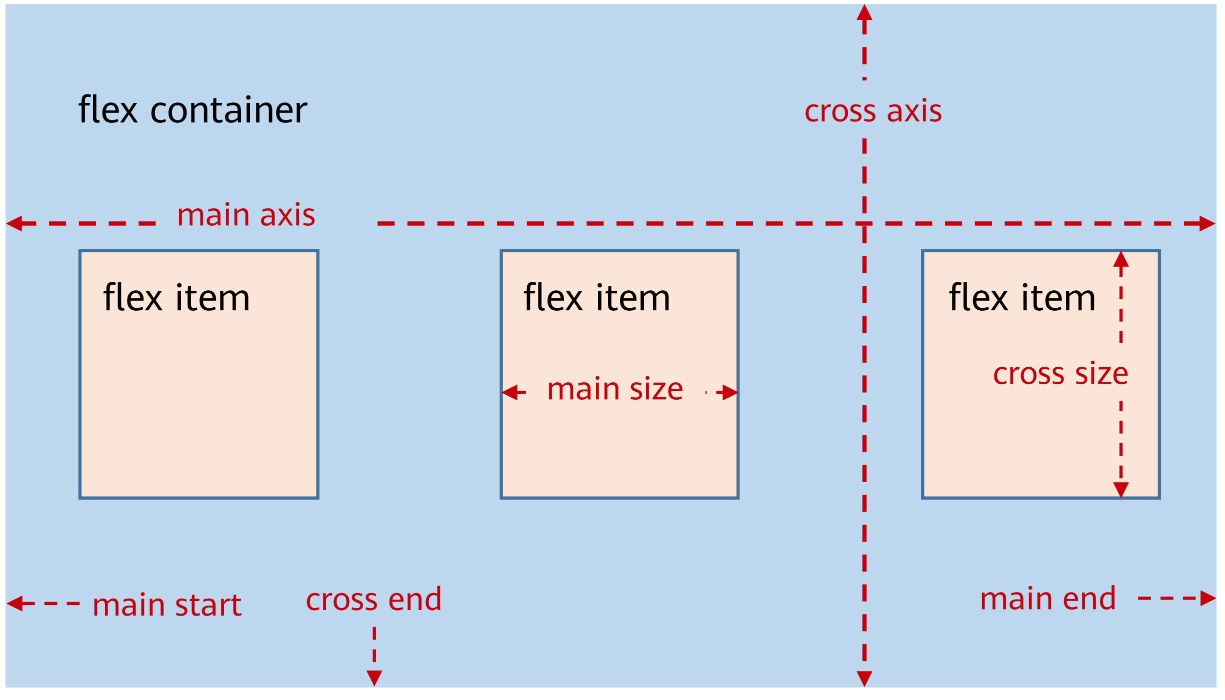
Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

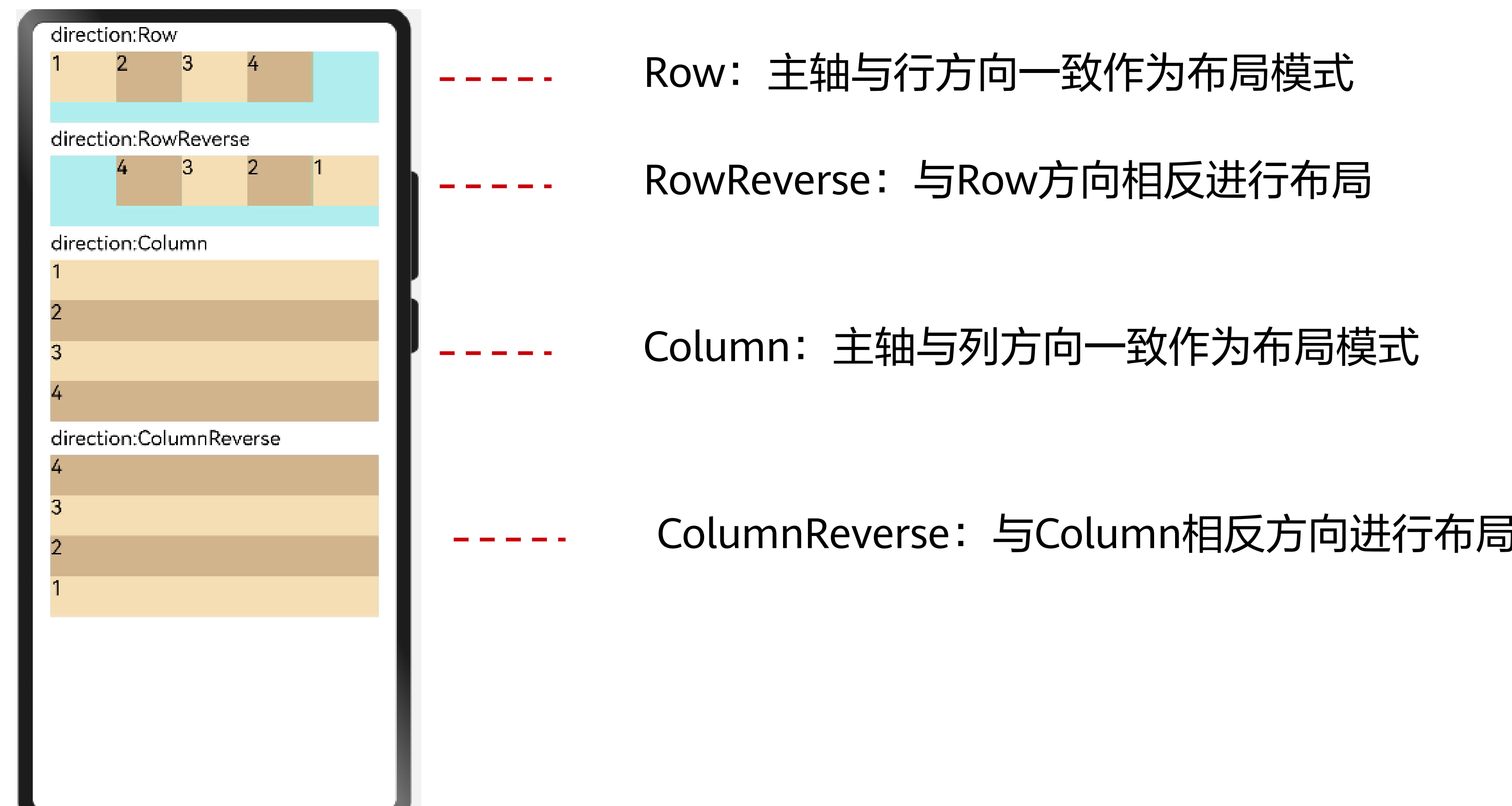
# Flex弹性布局组件

- Flex存在两根轴：默认水平的主轴（main axis）和垂直交叉轴（cross axis）。主轴的开始位置（与边框的交叉位置）叫做main start，结束位置叫做main end；交叉轴的开始位置叫做cross start，结束位置叫做cross end。



# Flex容器组件属性（1）

- FlexDirection：子组件在Flex容器上排列方向，即主轴的方向，默认值Row。

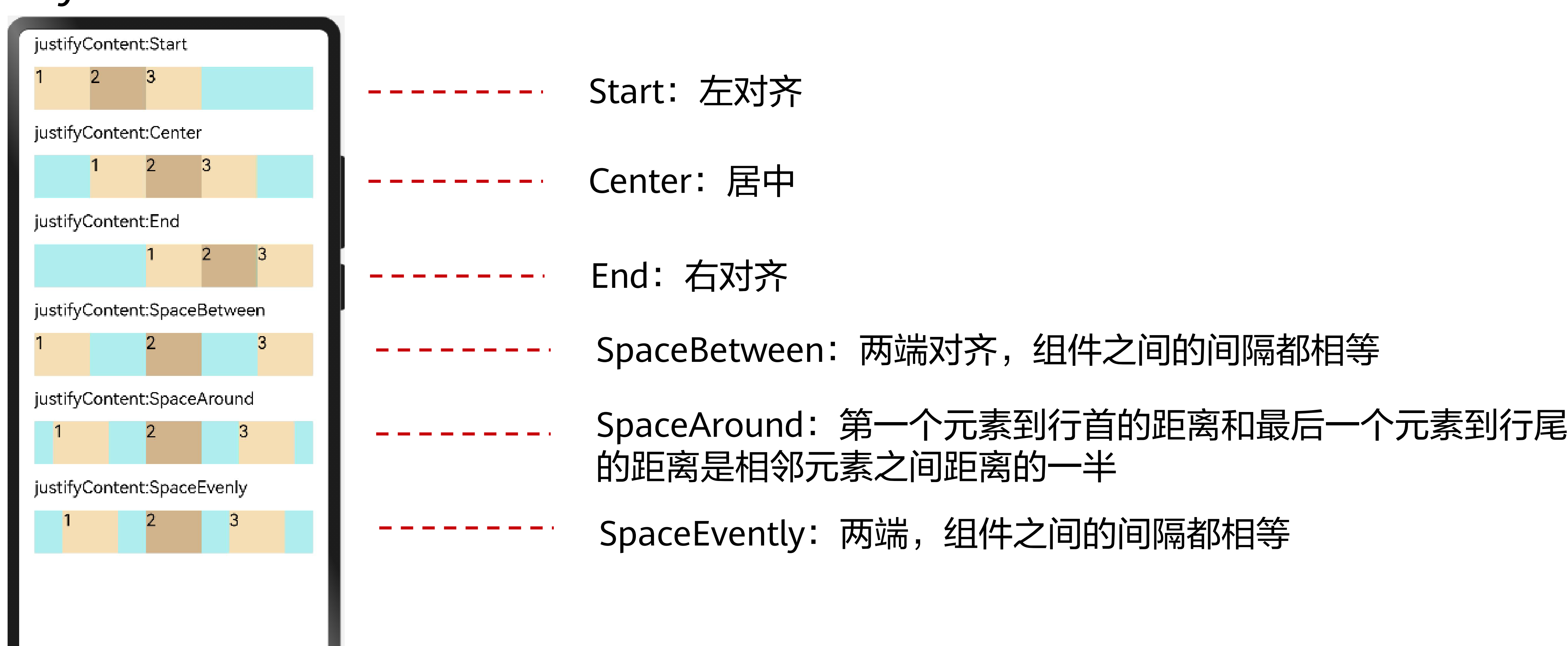



Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

# Flex容器组件属性（2）

- justifyContent：子组件在Flex容器主轴上的对齐格式。



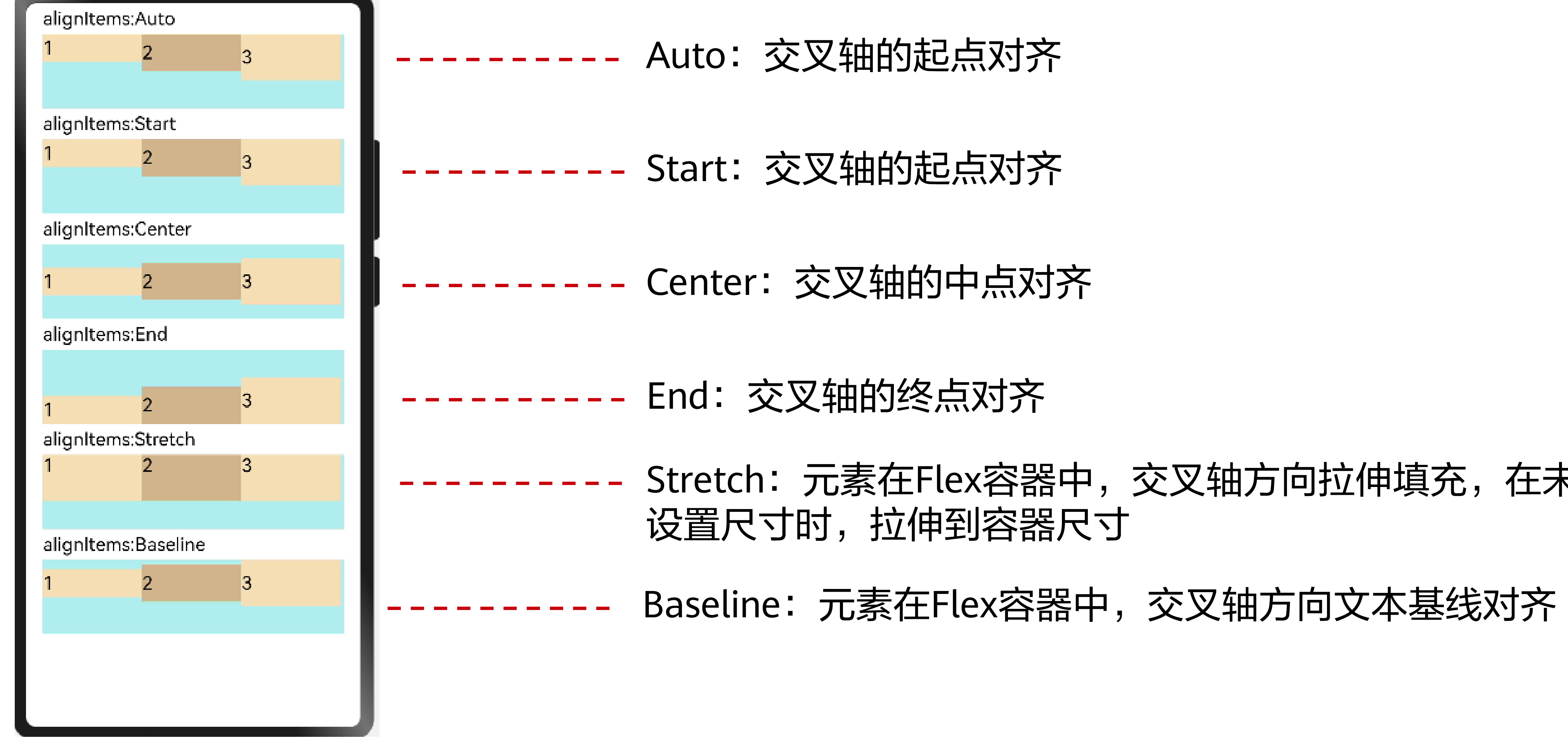

Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

# Flex容器组件属性（3）

- alignItems：子组件在Flex容器交叉轴上的对齐格式。




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

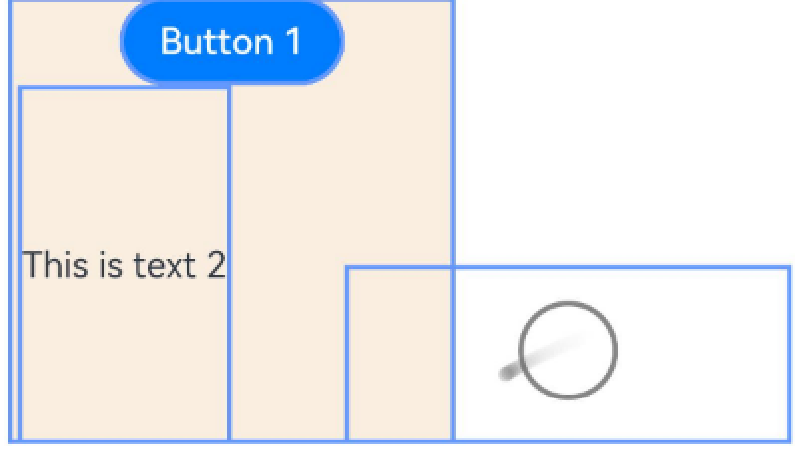


HUAWEI

# RelativeContainer

- 相对布局组件，用于复杂场景中元素对齐的布局。

```
// 外层容器的id默认为为'__container__'，右图中填充为黄色。
RelativeContainer() {
  Button("Button 1")
    .alignRules({
      middle: { anchor: "__container__", align: HorizontalAlign.Center },
      // 水平方向上，组件中部与容器中间对齐，即组件在容器中水平居中
    })
    .id("bt1") // id设置为bt1
  Text("This is text 2")
    .id("tx2") // id设置为tx2
    .alignRules({
      bottom: { anchor: "__container__", align: VerticalAlign.Bottom }, // 组件下边与容器下边对齐
      top: { anchor: "bt1", align: VerticalAlign.Bottom }, // 组件上边与button1底部对齐
      right: { anchor: "bt1", align: HorizontalAlign.Center } //  组件右侧与button1中间点对齐
    })
  LoadingProgress( )
    .id("lp3") // id设置为lp3
    .alignRules({
      left: { anchor: "bt1", align: HorizontalAlign.End }, // 组件左边对齐容器bt1的右边
      top: { anchor: "tx2", align: VerticalAlign.Center }, // 组件上边对齐容器tx2的中间
      bottom: { anchor: "__container__", align: VerticalAlign.Bottom } // 组件下边对齐最外层容器的
底边
    })
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# GridCol和GridRow

- 栅格布局是一种通用的辅助定位工具，可以帮助开发人员解决多尺寸多设备的动态布局问题。通过将页面划分为等宽的列数和行数，可以灵活调整间距，自动换行和自适应功能可以自动适应不同设备上的排版。
- GridRow为栅格容器组件，需与栅格子组件GridCol联合使用。

```
struct GridRowExample {
build() {
    Column() {
      GridRow({
        columns: 5,       //设置布局列数
        gutter: {x:5, y:10},       //栅格布局间距，x代表水平方向
        breakpoints: {value:["400vp", "600vp", "800vp"],   //设置断点值的断点数列以及
基于窗口或容器尺寸的相应参照
          reference: BreakpointsReference.WindowSize},
        direction: GridRowDirection.Row     //栅格布局排列方向
      }) {
        ForEach(this.bgColors, (color)=>{
          GridCol({ span: {xs:1, sm:2, md:3, lg:4}}) {
                ......
       .onBreakpointChange((breakpoint) => {
        this.currentBp = breakpoint
```

Copyright © Huawei Technologies Co., Ltd. All rights reserved.



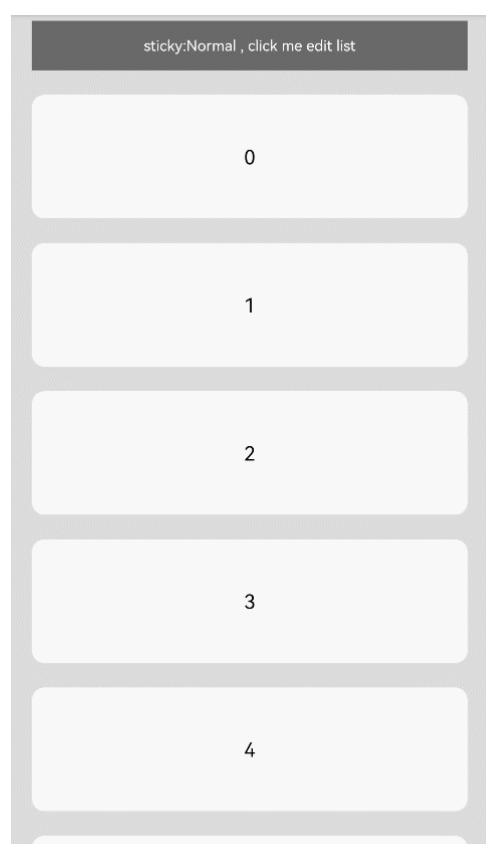
HUAWEI

# List与ListItem

- List：列表包含一系列相同宽度的列表项。适合连续、多行呈现同类数据，例如图片文本。
- ListItem：用来展示列表具体item，宽度默认充满List组件，必须配合List来使用。

```
// xxx.ets
@Entry
@Component
struct ListItemExample {
  private arr: number[] = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  @State editFlag: boolean = false

  build() {
    Column() {
      List({ space: 20, initialIndex: 0 }) {      //设置当前List初次加载时视口起始位置显示的item的索引值。如果设置的值超过了当前List最后一个item的索引值，则设置不生效
        ListItem() {
          Text('sticky:Normal , click me edit list')
            .width('100%').height(40).fontSize(12).fontColor(0xFFFFFF)
            .textAlign(TextAlign.Center).backgroundColor(0x696969)
            .onClick(() => {
              this.editFlag = !this.editFlag    //当前ListItem元素是否可编辑，进入编辑模式后可删除或移动列表项
            })
        }.sticky(Sticky.Normal)      //设置ListItem吸顶效果
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

# Grid和GridItem

- Grid：网格容器，由“行”和“列”分割的单元格所组成，通过指定“项目”所在的单元格做出各种各样的布局。
- GridItem：网格容器中单项内容容器。

```
@Entry
@Component
struct GridItemExample {
  @State numbers: string[] = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15"]
  build() {
    Column() {
      Grid() {
        GridItem() {Text('4')}.rowStart(0).rowEnd(3) //设置当前元素起始行号，设置当前元素终点行号
        ForEach(this.numbers, (item: string) => { GridItem() {Text(item)}
        }, (item: string) => item)

        GridItem() {Text('5')} . columnStart(0).columnEnd(4) //设置当前元素起始列号，当前元素终点列号

      }
      .columnsTemplate('1fr 1fr 1fr 1fr 1fr') // 设置当前网格布局列的数量或最小列宽值，不设置时默认1列
      .rowsTemplate('1fr 1fr 1fr 1fr 1fr') // 设置当前网格布局行的数量或最小行高值，不设置时默认1行
    }
  }
}
```

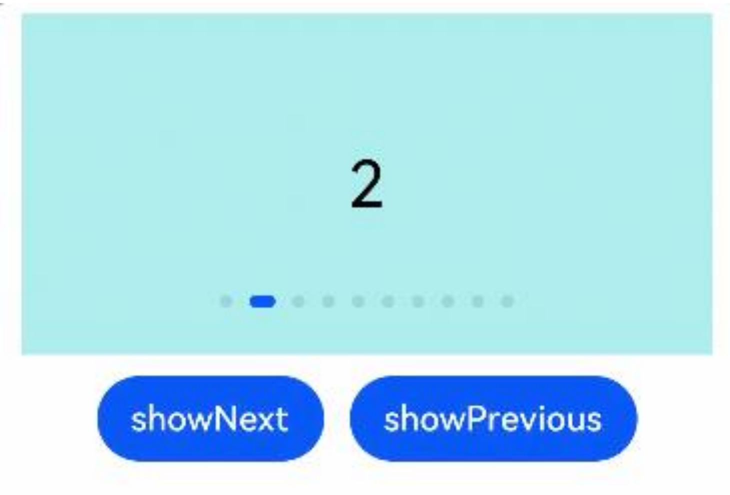
| 4 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| | 4 | 5 | 6 | 7 |
| | 8 | 9 | 10 | 11 |
| | 12 | 13 | 14 | 15 |
| 5 | | | | |

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# Swiper

- 滑块视图容器，提供子组件滑动轮播显示的能力。

```
@Entry
@Component
struct SwiperExample {
  private swiperController: SwiperController = new SwiperController()    //给组件绑定一个控制器，用来控制组件翻页
  build() {
    Column({ space: 5 }) {
      Swiper(this.swiperController) {
      .cachedCount(2)   //设置预加载子组件个数
      .index(1)        //设置当前在容器中显示的子组件的索引值
      .autoPlay(true)   //子组件是否自动播放，自动播放状态下，导航点不可操作
      .interval(4000)   //使用自动播放时播放的时间间隔，单位为毫秒
      .indicator(true)    //是否启用导航点指示器
      .loop(true)      //是否开启循环
      .duration(1000)    //子组件切换的动画时长，单位为毫秒
      .itemSpace(0)    //设置子组件与子组件之间间隙
      .curve(Curve.Linear)    //设置Swiper的动画曲线，默认为淡入淡出曲线
      .onChange((index: number) => {
        console.info(index.toString())
      })
......
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# Refresh

- 可以进行页面下拉操作并显示刷新动效的容器组件。

```
// xxx.ets
@Entry
@Component
struct RefreshExample {
  @State isRefreshing: boolean = false
  @State counter: number = 0

  build() {
    Column() {
      Refresh({ refreshing: $$this.isRefreshing, offset: 120, friction: 100 }) {   //当前组件是否正在刷新,该参数支持$$双向绑定变量；刷新组件静止时距离父组件顶部的距离；下拉摩擦系数，取值范围为0到100。
        Text('Pull Down and refresh: ' + this.counter)
          .fontSize(30)
          .margin(10)
      }
      .onStateChange((refreshStatus: RefreshStatus) => {     //当前刷新状态变更时，触发回调
        console.info('Refresh onStatueChange state is ' + refreshStatus)
      })
      .onRefreshing(() => {     //进入刷新状态时触发回调
        setTimeout(() => {
          this.counter++
          this.isRefreshing = false
        }, 1000)
        console.log('onRefreshing test')
```

Pull Down and refresh: 0

Copyright © Huawei Technologies Co., Ltd. All rights reserved.



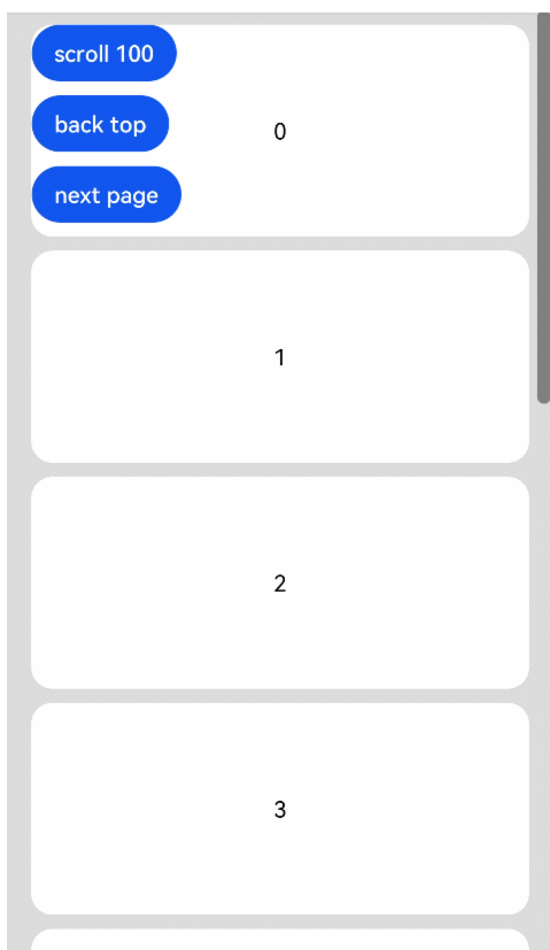
HUAWEI

# Scroll

- 可滑动的容器组件，当子组件的布局尺寸超过父组件的视口时，内容可以滑动。

```
// xxx.ets
@Entry
@Component
struct ScrollExample {
  scroller: Scroller = new Scroller()
  private arr: number[] = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  build() {
    Stack({ alignContent: Alignment.TopStart }) {
      Scroll(this.scroller) {
        Column() {
          ForEach(this.arr, (item) => {
            Text(item.toString())
              ......
          }, item => item)
        }.width('100%')
      }
```

```
      .scrollable(ScrollDirection.Vertical)   //设置滑动方向
      .scrollBar(BarState.On)   //设置滑动条状态
      .scrollBarColor(Color.Gray)   //设置滑动条颜色
      .scrollBarWidth(30)   //设置滑动条的宽度
      .onScroll((xOffset: number, yOffset: number) => {   //滚动事件回调，返回滚动时水平、竖直方向偏移量
        console.info(xOffset + ' ' + yOffset) })
      .onScrollEdge((side: Edge) => {   //滚动到边缘事件回调
        console.info('To the edge')})
      .onScrollEnd(() => {   //滚动停止事件回调
        console.info('Scroll Stop') })
      Button('scroll 100')
        .onClick(() => {   // 点击后下滑100.0距离
          this.scroller.scrollTo({ xOffset: 0, yOffset: this.scroller.currentOffset().yOffset + 100 })
              .......
      Button('back top')
        .onClick(() => {   // 点击后回到顶部
          this.scroller.scrollEdge(Edge.Top) })
        .margin({ top: 60, left: 20 })
      Button('next page')
        .onClick(() => {   // 点击后下滑到底部
          this.scroller.scrollPage({ next: true }) })
        .margin({ top: 110, left: 20 })......
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

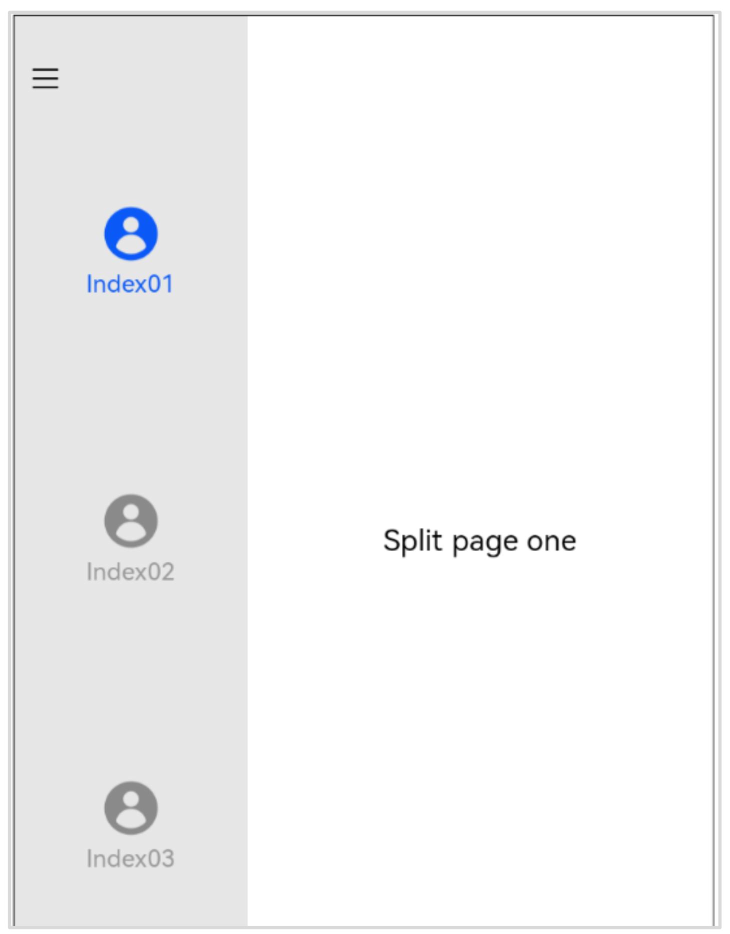
HUAWEI

# SideBarContainer

- 提供侧边栏可以显示和隐藏的侧边栏容器，通过子组件定义侧边栏和内容区，第一个子组件表示侧边栏，第二个子组件表示内容区。

```
// xxx.ets
@Entry
@Component
struct SideBarContainerExample {
  normalIcon : Resource = $r("app.media.user")
  selectedIcon: Resource = $r("app.media.userFull")
  @State arr: number[] = [1, 2, 3]
  @State current: number = 1

  build() {
    SideBarContainer(SideBarContainerType.Embed)    //设置侧边栏的显示类型。默认值：SideBarContainerType.Embed
     ........
    .sideBarWidth(240)    //设置侧边栏的宽度
    .minSideBarWidth(210)       //设置侧边栏最小宽度
    .maxSideBarWidth(260)       //设置侧边栏最大宽度
    .onChange((value: boolean) => {
      console.info('status:' + value)
       })
    }
}
```



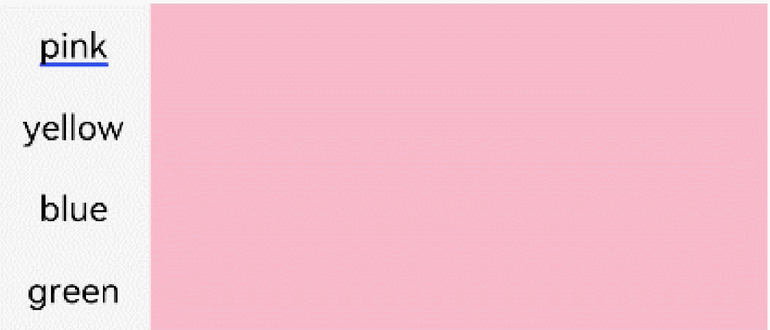

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# Tabs和TabContent

- Tabs：通过页签进行内容视图切换的容器组件，每个页签对应一个内容视图。
- TabContent：仅在Tabs中使用，对应一个切换页签的内容视图。

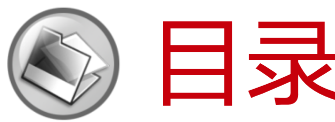
```
@Entry
@Component
struct TabsExample {
  private controller: TabsController = new TabsController()    //设置Tabs控制器

  build() {
    Column() {
      Tabs({ barPosition: BarPosition.Start, controller: this.controller }) {     //设置Tabs的页签位置
        TabContent() {
          Column().width('100%').height('100%').backgroundColor(Color.Pink)
        }.tabBar('pink')
      .vertical(true)    //设置为false是为横向Tabs，设置为true时为纵向Tabs
      .scrollable(true)     //设置为true时可以通过滑动页面进行页面切换，为false时不可滑动切换页面
      .barMode(BarMode.Fixed)    //所有TabBar平均分配barWidth宽度（纵向时平均分配barHeight高度）
      .barWidth(70)       //TabBar的宽度值
      .barHeight(150)      //TabBar的高度值
      .animationDuration(400)      //TabContent滑动动画时长
      .onChange((index: number) => {     //Tab页签切换后触发的事件
        console.info(index.toString())})
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI



# 目录


1. ArkUI介绍
2. 布局组件
3. **常用组件介绍**


Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

# Text

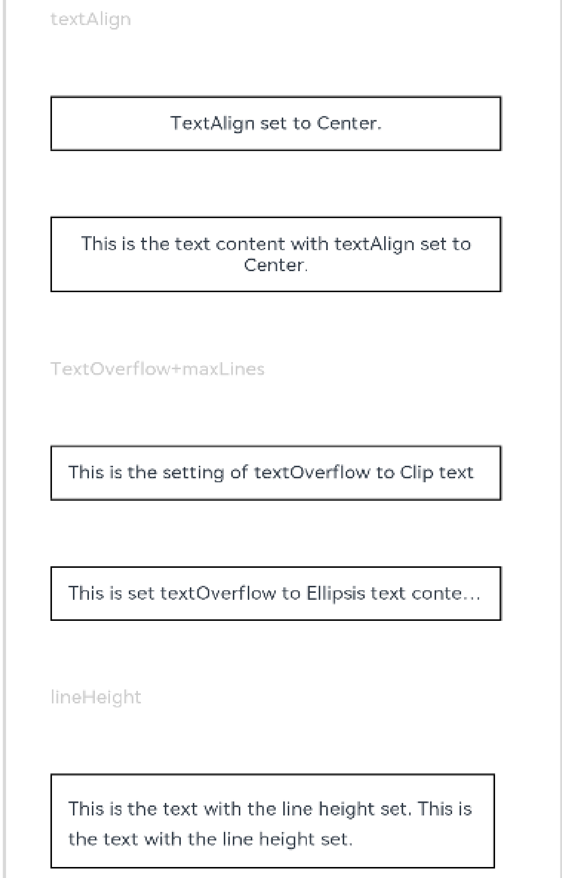
- 显示一段文本的组件。

```
//水平方向的对齐方式
Text('TextAlign set to Center.').textAlign(TextAlign.Center)  .fontSize(12)
Text('This is the text content with textAlign set to Center.')
      .textAlign(TextAlign.Center).fontSize(12)

//文本超长时的显示方式1:超出maxLines截断内容展示
 Text('This is the setting of textOverflow to Clip text content.')
      .textOverflow({ overflow: TextOverflow.None })   .maxLines(1)

//文本超长时的显示方式2:超出maxLines展示省略号
Text('This is the setting of textOverflow to Clip text content.')
      .textOverflow({ overflow: TextOverflow. Ellipsis }) .maxLines(1)

//设置文本的文本行高，设置值不大于0时，不限制文本行高，自适应字体大小
Text('This is the text with the line height set. This is the text with the line height set.')
      .fontSize(12).border({ width: 1 }).padding(10) .lineHeight(20)
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

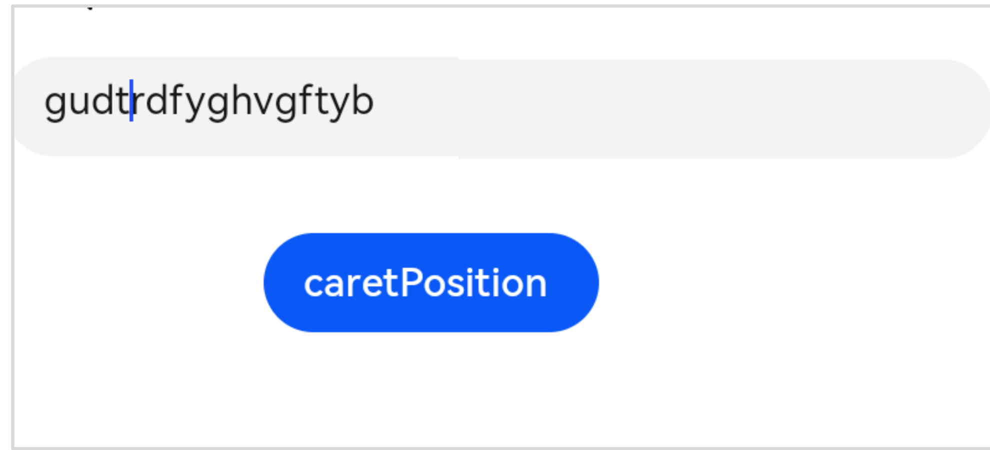


HUAWEI

# TextArea

- 多行文本输入框组件，当输入的文本内容超过组件宽度时会自动换行显示。

```
@Entry
@Component
struct TextAreaExample2 {
    controller: TextAreaController = new TextAreaController()
    build() {
        Column() {
            TextArea({ placeholder: 'input your word',controller:this.controller })
            Button('caretPosition')
                .onClick(() => {
                    this.controller.caretPosition(4)
                })
        }
    }
}
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.



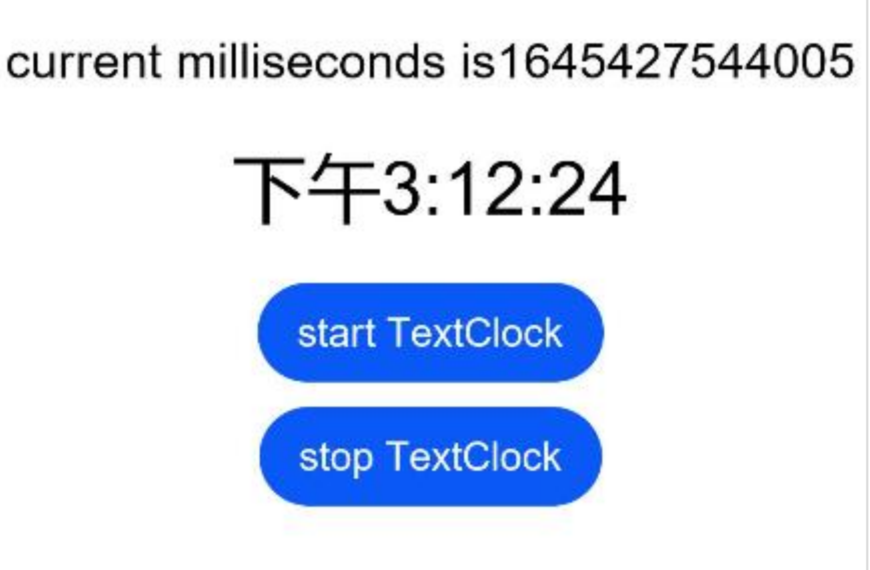
HUAWEI

# TextClock

- TextClock通过文本显示当前系统时间，支持不同时区的时间显示，时间显示最高精度到秒级。

```
// xxx.ets
@Entry
@Component
struct Second {
  @State accumulateTime: number = 0
  controller: TextClockController = new TextClockController()

  build() {
      Flex({ direction: FlexDirection.Column, alignItems: ItemAlign.Center, justifyContent:
FlexAlign.Center}) {
         Text('current milliseconds is' + this.accumulateTime)
           .fontSize(20)
         TextClock({timeZoneOffset: -8, controller: this.controller})      //设置时区偏移量，绑定一个
控制器
           .format('hhmmss')      //设置显示时间格式
           .onDateChange((value: number) => {      //提供时间变化回调，该事件最小回调间隔为秒
             this.accumulateTime = value
           })
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

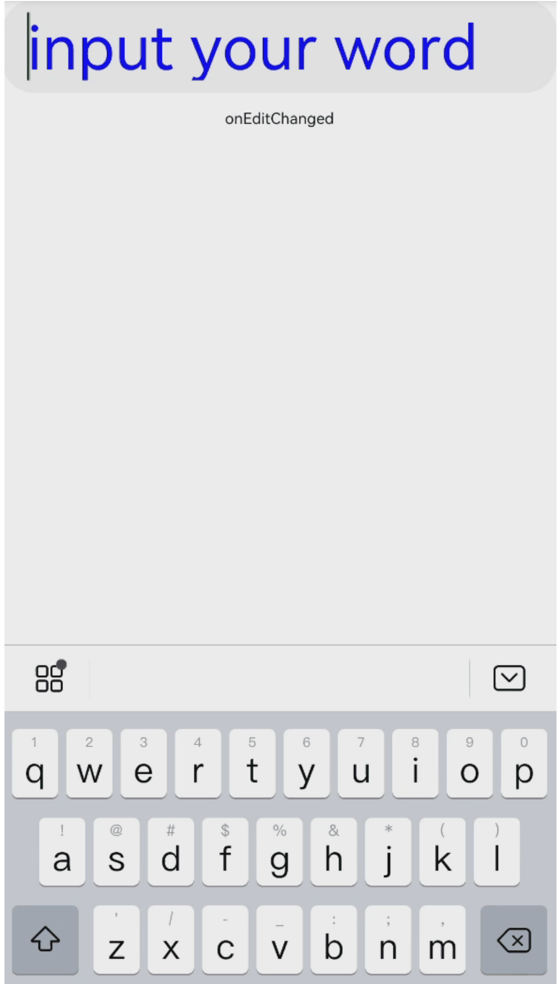


HUAWEI

# TextInput

- 提供单行文本输入组件。

```
@Entry
@Component
struct TextInputExample1 {
  @State text: string = ''
  build() {
    Column() {
      TextInput({ placeholder: ' input your word ' })      //无输入时的提示文本
        .placeholderColor("rgb(0,0,225)")        //设置placeholder颜色
        .placeholderFont({ size: 30, weight: 100, family: 'cursive', style: FontStyle.Italic })      //设置
placeholder文本样式
        .caretColor(Color.Blue)
        .height(50)
        .fontSize(30)
        .fontWeight(FontWeight.Bold)
        .fontFamily("sans-serif")
        .fontStyle(FontStyle.Normal)
        .fontColor(Color.Red)
        .onChange((value: string) => {
          this.text = value
        })
      Text(this.text).width('90%')
    }
  }
}
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# TextPicker

- 文本类滑动选择器组件。

```
@Entry
@Component
struct TextPickerExample {
  private select: number = 1
  private fruits: string[] = ['apple1', 'orange2', 'peach3', 'grape4']

  build() {
    Column() {
      TextPicker({range: this.fruits, selected: this.select})   //选择器的数据选择范围，选中项在数组中的index值
        .onChange((value: string, index: number) => {       //选中项的值，优先级低于selected
          console.info('Picker item changed, value: ' + value + ', index: ' + index)
        })
    }
  }
}
```

grape4
apple1
orange2
peach3
grape4

Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

# TextTimer

- 文本计时器组件，支持自定义时间格式。

```
@Entry
@Component
struct TextTimerExample {
  textTimerController: TextTimerController = new TextTimerController()
  @State format: string = 'hh:mm:ss.ms'
  build() {
    Column() {
      TextTimer({controller: this.textTimerController})
        .format(this.format)
        .fontColor(Color.Black)
        .fontSize(50)
        .onTimer((utc: number, elapsedTime: number) => {      //时间文本发生变化时触发
          console.info('textTimer notCountDown utc is：' + utc + ', elapsedTime: ' + elapsedTime)
        })        //utc:当前显示的时间，单位为毫秒；elapsedTime：计时器经过的时间，单位为毫秒
      Row() {
        Button("start").onClick(() => {
          this.textTimerController.start();
        });
        Button("pause").onClick(() => {
          this.textTimerController.pause();
        });
        Button("reset").onClick(() => {
          this.textTimerController.reset();
        });...
```

Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

# TimePicker

- 选择时间的滑动选择器组件。

```
@Entry
@Component
struct TimePickerExample {
  private selectedTime: Date = new Date('7/22/2022 8:00:00')

  build() {
    Column() {
      TimePicker({
        selected: this.selectedTime,      //设置选中项的时间，默认当前系统时间
      })
      .useMilitaryTime(true)      //展示时间是否为24小时制，不支持动态修改
      .onChange((date: TimePickerResult) => {
        console.info('select current date is: ' + JSON.stringify(date))
      })
    }.width('100%')
  }
}
```

| 06 | 58 |
|---|---|
| 07 | 59 |
| 08 | 00 |
| 09 | 01 |
| 10 | 02 |

Copyright © Huawei Technologies Co., Ltd. All rights reserved.



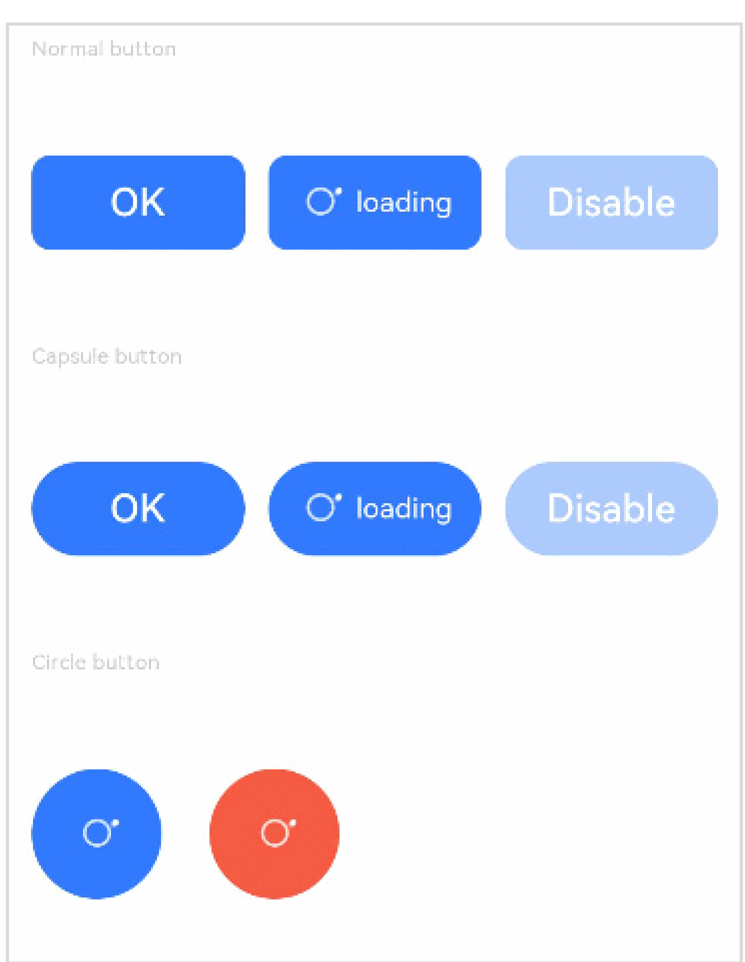
HUAWEI

# Button

- 按钮组件，可快速创建不同样式的按钮。
- 通过type属性的设置，使用不同样式按钮。
- 通过stateEffect属性的设置，决定是否开启按压态显示效果。

```
...
//常规按钮
Button('OK', { type: ButtonType.Normal, stateEffect: true })
.borderRadius(8).backgroundColor(0x317aff).width(90)

 Button('Disable', { type: ButtonType.Normal, stateEffect: false }).opacity(0.4)
.borderRadius(8).backgroundColor(0x317aff).width(90)
...
//胶囊按钮
 Button('OK', { type: ButtonType.Capsule, stateEffect: true })
.backgroundColor(0x317aff).width(90)
...
//圆形按钮
 Button({ type: ButtonType.Circle, stateEffect: true }) {
        LoadingProgress().width(20).height(20).color(0xFFFFFF)
  }.width(55).height(55).backgroundColor(0x317aff)
...
```



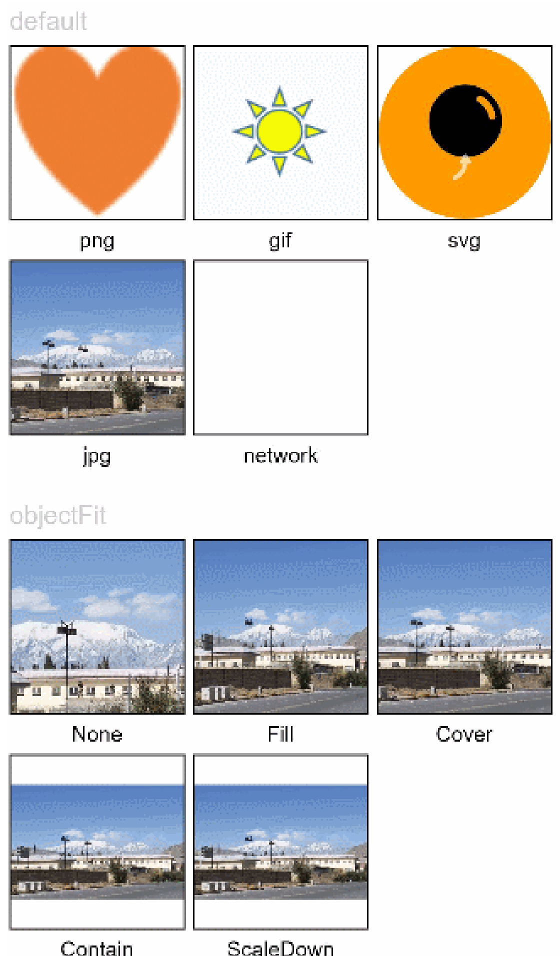

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# Image

- 图片组件，支持本地图片和网络图片的渲染展示。
- 加载显示不同类型的图片，并设置图片的缩放类型。

```
@Entry
@Component
@State src: string = this.on

  build() {
    Column() {
              ......
        Image($r('app.media.ic_png'))
          .width(110).height(110).border({ width: 1 }).borderStyle(BorderStyle.Dashed)
          .overlay('png', { align: Alignment.Bottom, offset: { x: 0, y: 20 } })
         Image($r('app.media.ic_gif'))
          .width(110).height(110).border({ width: 1 }).borderStyle(BorderStyle.Dashed)
          .overlay('gif', { align: Alignment.Bottom, offset: { x: 0, y: 20 } })
         Image($r('app.media.ic_svg'))
          .width(110).height(110).border({ width: 1 }).borderStyle(BorderStyle.Dashed)
          .overlay('svg', { align: Alignment.Bottom, offset: { x: 0, y: 20 } })
        }......
         Image($r('app.media.img_example'))
          .border({ width: 1 }).borderStyle(BorderStyle.Dashed)
          .objectFit(ImageFit.Auto).width(110).height(110)   //设置图片的缩放类型为自适应显示
          .overlay('Auto', { align: Alignment.Bottom, offset: { x: 0, y: 20 } })
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

# Radio

- 单选框，提供相应的用户交互选择项。

```
// xxx.ets
@Entry
@Component
struct RadioExample {
  build() {
    Flex({ direction: FlexDirection.Row, justifyContent: FlexAlign.Center, alignItems: ItemAlign.Center }) {
      Column() {
        Text('Radio1')
        Radio({ value: 'Radio1', group: 'radioGroup'}).checked(true)
        //当前单选框的值Radio1，当前单选框的所属群组名称，相同group的Radio只能有一个被选中
          .height(50)
          .width(50)
          .onChange((isChecked: boolean) => {
            console.log('Radio1 status is' + isChecked)
          })
        ......
      }
```

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

# Checkbox

- 多选框组件，通常用于某选项的打开或关闭。

```
@Entry
@Component
struct index {

  build() {
    Row() {
      Checkbox({name:'checkbox1',  group: 'checkboxGroup'})   //多选框名称，多选框的群组名称
        .select(true)        //设置多选框是否选中
        .selectedColor(0xed6f21)       //设置多选框选中状态颜色
        .onChange((value: boolean) => {          //当选中状态发生变化时，触发该回调
          console.info('Checkbox1 change is'+ value)
        })
      Checkbox({name: 'checkbox2',  group: 'checkboxGroup'})
        .select(false)
        .selectedColor(0x39a2db)
        .onChange((value: boolean) => {
          console.info('Checkbox2 change is'+ value)
        })
    }
  }
}
```

Copyright © Huawei Technologies Co., Ltd. All rights reserved.



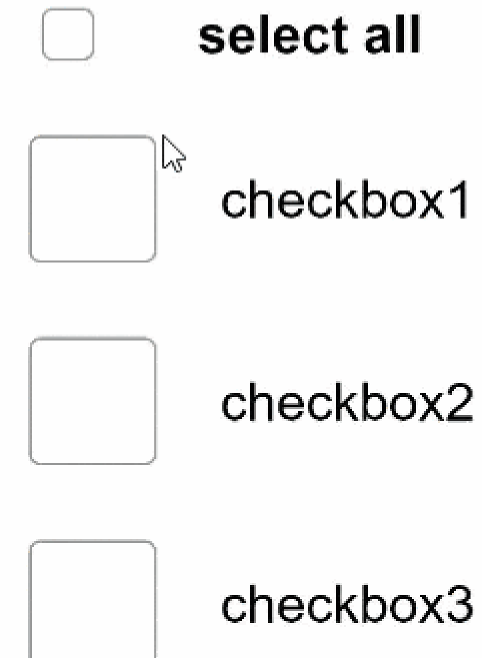
HUAWEI

# CheckboxGroup

- 多选框群组，用于控制多选框全选或者不全选状态。

```
@Entry
@Component
struct CheckboxExample {

  build() {
    Scroll() {
      Column() {
        CheckboxGroup({group : 'checkboxGroup'})    //群组名称
          .selectedColor(0xed6f21)      //设置被选中或部分选中状态的颜色
          .onChange((itemName:CheckboxGroupResult) => {
            console.info("TextPicker::dialogResult is" + JSON.stringify(itemName))
          })
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.



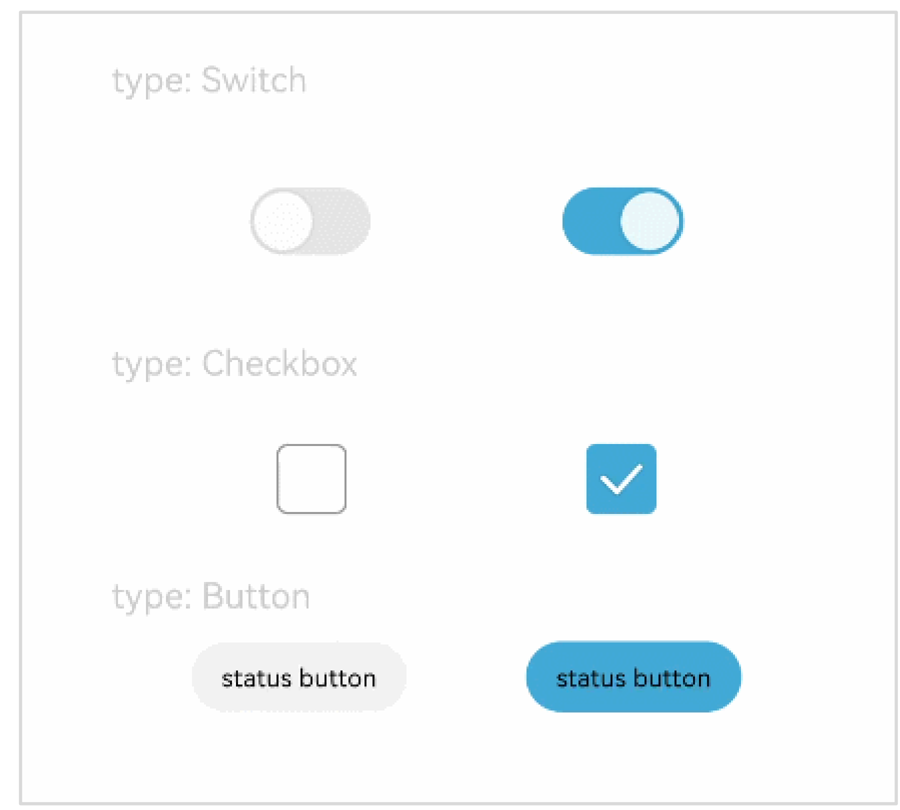
HUAWEI

# Toogle

- 组件提供勾选框样式、状态按钮样式及开关样式。

```
// xxx.ets
@Entry
@Component
struct ToggleExample {

  build() {
    Column({ space: 10 }) {
      Text('type: Switch').fontSize(12).fontColor(0xcccccc).width('90%')
      Flex({ justifyContent: FlexAlign.SpaceEvenly, alignItems:
ItemAlign.Center }) {
        Toggle({ type: ToggleType.Switch, isOn: false })    //开关是否打开
          .selectedColor(0xed6f21)
          .switchPointColor(0xe5ffffff)
          .onChange((isOn: boolean) => {
            console.info('Component status:' + isOn)
          })
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# AlphabetIndexer

- 字母索引条。

```
@Entry
@Component
struct AlphabetIndexerSample {
  private value: string[] = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S',
'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

  build() {
    AlphabetIndexer({ arrayValue: this.value, selected: 0 })
      .selectedColor(0xffffff) // 选中颜色
      .popupColor(0xFFFAF0) // 弹出框颜色
      .selectedBackgroundColor(0xCCCCCC) // 选中背景颜色
      .popupBackground(0xD2B48C) // 弹出框背景颜色
      .usingPopup(true) // 是否显示弹出框
      .selectedFont({ size: 16, weight: FontWeight.Bolder }) // 选中的样式
      .popupFont({ size: 30, weight: FontWeight.Bolder }) // 弹出框的演示
      .itemSize(28) // 每一项的大小正方形
      .alignStyle(IndexerAlign.Left) // 左对齐
      .onSelect((index: number) => {//索引条选中回调，返回值为当前选中索引
        console.info(this.value[index] + '被选中了') })
       .onRequestPopupData((index: number) => {...}) //字母索引提示弹窗字符串列表选中回调
      .margin({ left: 50 })
  }
}
```



Copyright © Huawei Technologies Co., Ltd. All rights reserved.

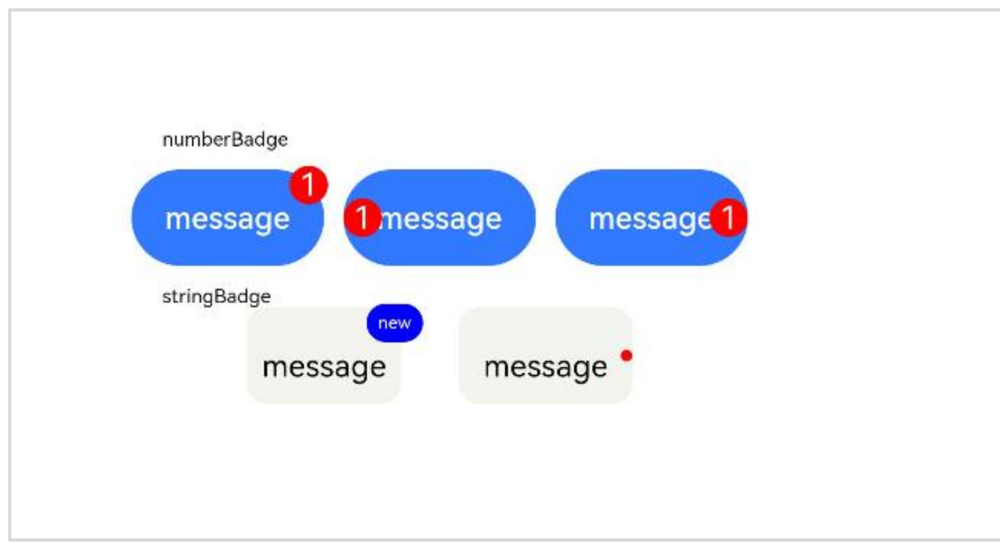
HUAWEI

# Badge

- 可以附加在单个组件上用于信息标记的容器组件。

```
// xxx.ets
@Entry
@Component
struct BadgeExample {
  @State counts: number = 1      //设置消息提醒数
  @State message: string = 'new'

  build() {
    Column() {
      Text('numberBadge').width('80%')
      Row({ space: 10 }) {
        // 数字上标，maxCount默认99,超过99展示99+
        Badge({
          count: this.counts,
          maxCount: 99,     //最大消息数，超过最大消息时仅显示maxCount+
          position: BadgePosition.RightTop,       //设置提示点显示位置
          style: { color: 0xFFFFFF, fontSize: 16, badgeSize: 20, badgeColor: Color.Red }
        }) {
          Button('message')
            .onClick(() => {
              this.counts++
            })
            .width(100).height(50).backgroundColor(0x317aff)
        }.width(100).height(50)
```



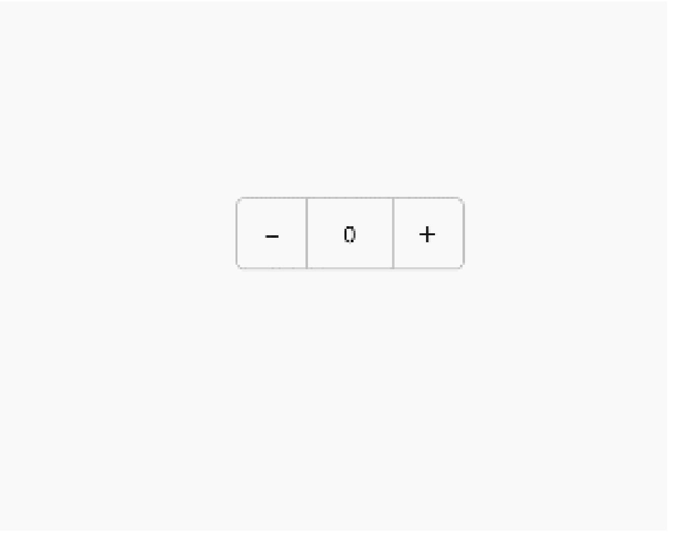

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# Counter

- 计数器组件，提供相应的增加或者减少的计数操作。

```
@Entry
@Component
struct CounterExample {
  @State value: number = 0

  build() {
    Column() {
      Counter() {
        Text(this.value.toString())
      }.margin(100)
      .onInc(() => {     //监听数值增加事件
        this.value++
      })
      .onDec(() => {     //监听数值减少事件
        this.value--
      })
    }.width("100%")
  }
}
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

# DatePicker

- 选择日期的滑动选择器组件：

```
// xxx.ets
@Entry
@Component
struct DatePickerExample {
  @State isLunar: boolean = false
  private selectedDate: Date = new Date('2021-08-08')
  build() {
    Column() {
      Button('切换公历农历')
        .margin({ top: 30 })
        .onClick(() => {
          this.isLunar = !this.isLunar
        })
      DatePicker({
        start: new Date('1970-1-1'),       //指定选择器的起始时间。默认值：Data('1970-1-1')
        end: new Date('2100-1-1'),         //指定选择器的结束日期。默认值：Data('2100-12-31')
        selected: this.selectedDate,       //设置选中项的日期。默认值：当前系统日期
      })
        .lunar(this.isLunar)     //设置弹窗的日期是否显示农历
        .onChange((value: DatePickerResult) => {
          this.selectedDate.setFullYear(value.year, value.month, value.day)
          console.info('select current date is: ' + JSON.stringify(value))
        })
```

切换公历农历

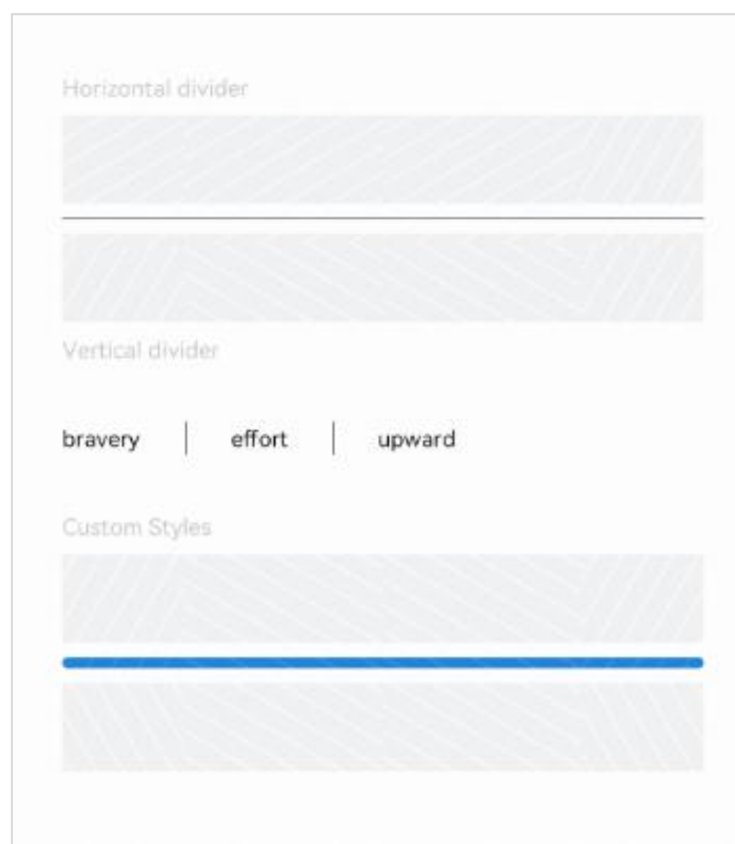
| | | |
|---|---|---|
| 2019年 | 6月 | 6日 |
| 2020年 | 7月 | 7日 |
| 2021年 | 8月 | 8日 |
| 2022年 | 9月 | 9日 |
| 2023年 | 10月 | 10日 |

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# Divider

- 提供分隔器组件，分隔不同内容块/内容元素。

```
...
//普通分割线
Divider()
...
//纵向分割线，vertical属性值为false时是水平，true时是垂直
Text('bravery')
Divider().vertical(true).margin(20).height(15)
Text('effort')
Divider().vertical(true).margin(20).height(15)
Text('upward’)
...
//设置分割线宽度和端点样式
 Divider().vertical(false)
.strokeWidth(5)//分割线宽度
.color(0x2788D9)//分割线颜色
.lineCap(LineCapStyle.Round)//分割线的端点样式
```



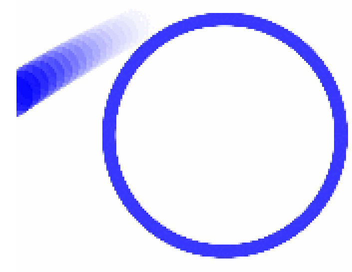

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

# LoadingProgress

- 用于显示加载动效的组件。

```
@Entry
@Component
struct LoadingProgressExample {
  build() {
    Column({ space: 5 }) {
      Text('Orbital LoadingProgress
').fontSize(9).fontColor(0xCCCCCC).width('90%')
      LoadingProgress()
        .color(Color.Blue)    //设置加载进度条颜色
    }.width('100%').margin({ top: 5 })
  }
}
```



Copyright © Huawei Technologies Co., Ltd. All rights reserved.

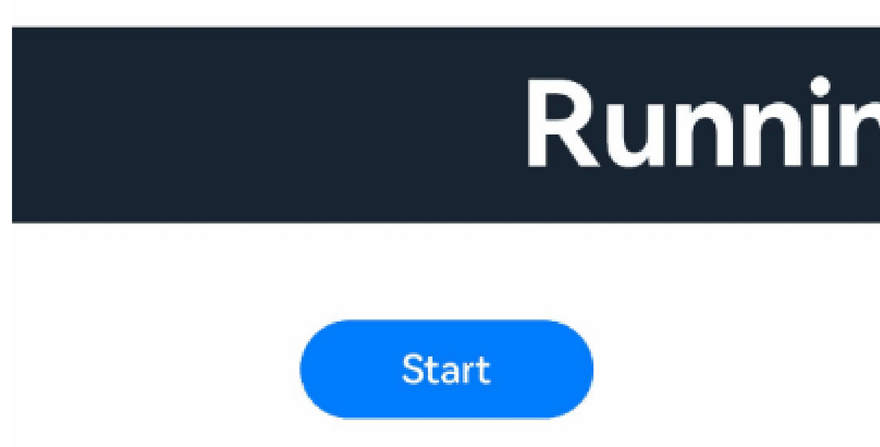


HUAWEI

# Marquee

- 跑马灯组件，用于滚动展示一段单行文本，仅当文本内容宽度超过跑马灯组件宽度时滚动。

```
@Entry
@Component
struct MarqueeExample {
  @State start: boolean = false          //控制跑马灯是否进入播放状态。
  @State fromStart: boolean = true     //设置文本从头开始滚动或反向滚动。
  @State step: number = 50    //滚动动画文本滚动步长。默认值：6，单位vp
  @State loop: number = 3     //设置重复滚动的次数，小于等于零时无限循环。默认值：-1
  @State src: string = "Running Marquee starts rolling“    //需要滚动的文本。
  build() {
      Marquee({
      start: this.start,
      step: this.step,
      loop: this.loop,
      fromStart: this.fromStart,
      src: this.src })
      .onStart(() => {          //开始滚动时触发回调。
       ...})
      .onBounce(() => {   //完成一次滚动时触发，若循环次数不为1，则该事件会多次触发。
        ... })
      .onFinish(() => {       //滚动全部循环次数完成时触发回调。
        ...  })
      Button(‘start’).onClick(() => {...})  //开始
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

# Progress

- 进度条组件，用于显示内容加载或操作处理等进度。

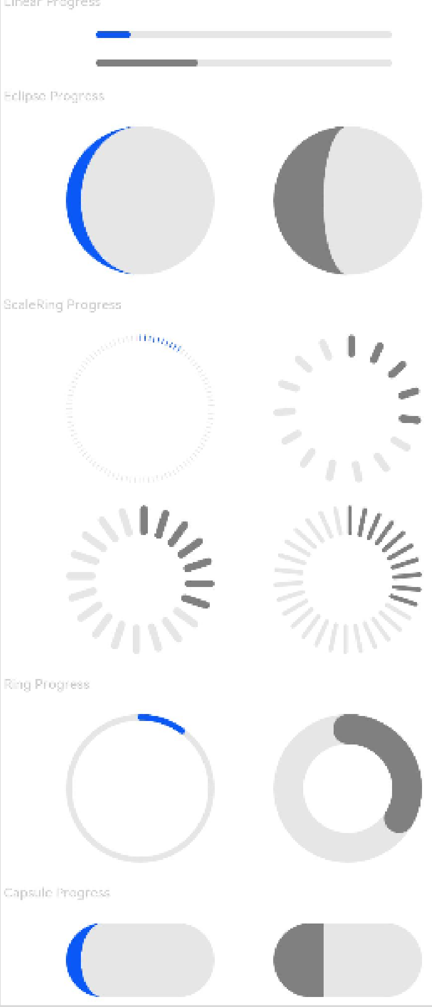
```
//线性样式进度条
Progress({ value: 20, total: 150, type: ProgressType.Linear })
.color(Color.Grey).value(50).width(200)

//圆形样式进度条
Progress({ value: 20, total: 150, type: ProgressType.Eclipse })
.color(Color.Grey).value(50).width(100)

//环形有刻度样式进度条
 Progress({ value: 20, total: 150, type: ProgressType.ScaleRing })
.color(Color.Grey).value(50).width(100)
.style({ strokeWidth: 20, scaleCount: 20, scaleWidth: 5 })

//环形无刻度样式进度条
 Progress({ value: 20, total: 150, type: ProgressType.Ring })
.color(Color.Grey).value(50).width(100)
.style({ strokeWidth: 20, scaleCount: 30, scaleWidth: 20 })

//胶囊样式进度条
 Progress({ value: 20, total: 150, type: ProgressType.Capsule })
.color(Color.Grey) .value(50) .width(100)   .height(50)
```



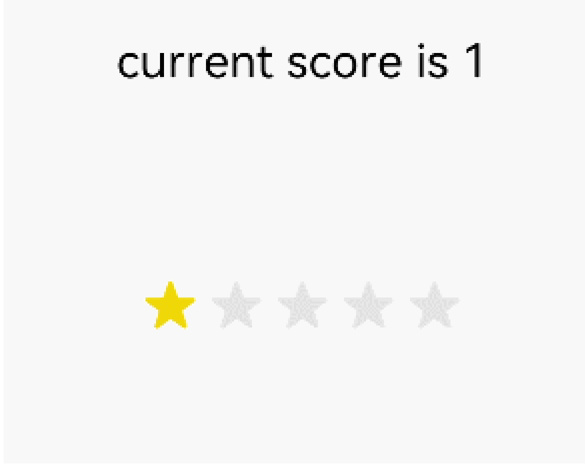

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# Rating

- 提供在给定范围内选择评分的组件。

```
// xxx.ets
@Entry
@Component
struct RatingExample {
  @State rating: number = 1     //设置并接收评分值。
  @State indicator: boolean = false     //仅作为指示器使用，不可操作。

  build() {
    Flex({ direction: FlexDirection.Column, alignItems: ItemAlign.Center,
justifyContent: FlexAlign.SpaceBetween }) {
      Text('current score is ' + this.rating).fontSize(20)
      Rating({ rating: this.rating, indicator: this.indicator })
        .stars(5)        //设置评星总数。
        .stepSize(0.5)        //操作评级的步长。
        .onChange((value: number) => {
          this.rating = value
        })
    }.width(350).height(200).padding(35)
  }
}
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

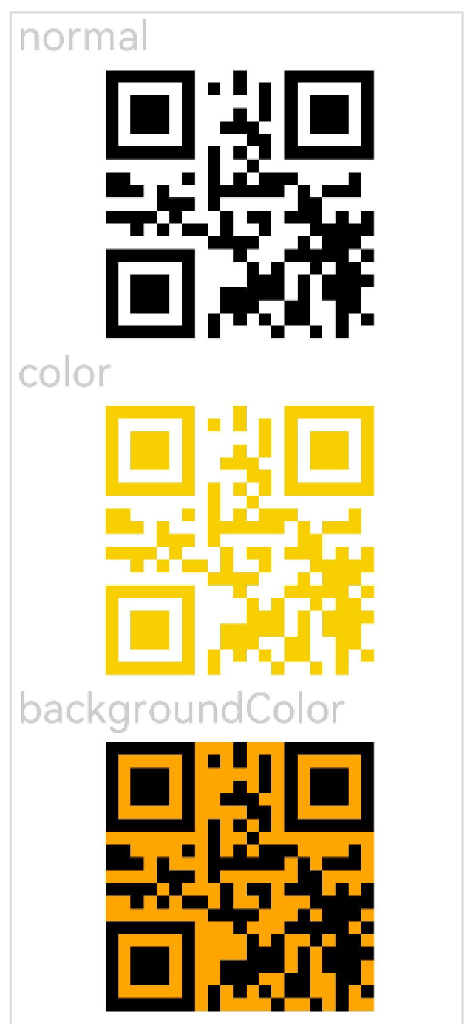
# QRCode

- 用于显示单个二维码的组件。

```
// xxx.ets
@Entry
@Component
struct QRCodeExample {
  private value: string ='hello world'    //二维码内容字符串
  build() {
    Column({ space: 5 }) {
      Text('normal').fontSize(9).width('90%').fontColor(0xCCCCCC).fontSize(30)
      QRCode(this.value).width(200).height(200)

      // 设置二维码颜色
      Text('color').fontSize(9).width('90%').fontColor(0xCCCCCC).fontSize(30)
      QRCode(this.value).color(0xF7CE00).width(200).height(200)

      // 设置二维码背景色
      Text('backgroundColor').fontSize(9).width('90%').fontColor(0xCCCCCC).fontSize(30)
      QRCode(this.value).width(200).height(200).backgroundColor(Color.Orange)
    }.width('100%').margin({ top: 5 })
  }
}
```

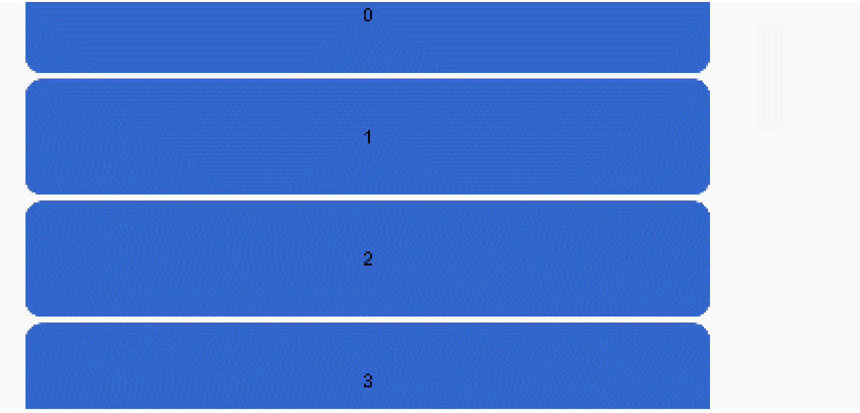



Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# ScrollBar

- 滚动条组件ScrollBar，用于配合可滚动组件使用，如List、Grid、Scroll。

```
@Entry
@Component
struct ScrollBarExample {
  private scroller: Scroller = new Scroller()    //可滚动组件的控制器，用于与可滚动组件进行绑定
  private arr: number[] = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  build() {
    Column() {
      Stack({ alignContent: Alignment.End }) {
        Scroll(this.scroller) {
          Flex({ direction: FlexDirection.Column }) {
            ForEach(this.arr, (item) => {
              Row() {
                 .......
                 }
            }, item => item)
          }.margin({ left: 52 })
        }
        .scrollBar(BarState.Off)
        .scrollable(ScrollDirection.Vertical)
   //纵向滚动条，滚动条状态按需显示
    ScrollBar({ scroller: this.scroller, direction: ScrollBarDirection.Vertical,state: BarState.Auto }){
         ......
    }.width(30).backgroundColor('#ededed')
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

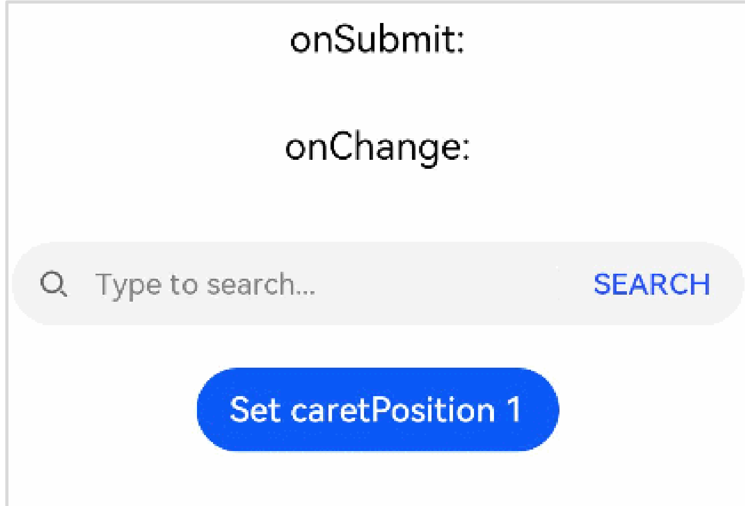


HUAWEI

# Search

- 搜索框组件，适用于浏览器的搜索内容输入框等应用场景。

```
// xxx.ets
@Entry
@Component
struct SearchExample {
  @State changeValue: string = '';
  @State submitValue: string = '';
  controller: SearchController = new SearchController();
  build() {
    Column() {
      Text('onSubmit:' + this.submitValue).fontSize(18).margin(15)
      Text('onChange:' + this.changeValue).fontSize(18).margin(15)
      Search({ value: this.changeValue, placeholder: ' Type to search... ' , controller: this.controller })   //
设置当前显示的搜索文本内容、无输入时的提示文本、Search组件控制器
        .searchButton('SEARCH')      //搜索框末尾搜索按钮文本内容，默认无搜索按钮。
            ......
        .onSubmit((value: string) => {   //点击搜索图标、搜索按钮或者按下软键盘搜索按钮时触发该回调。
          this.submitValue = value; })
        .onChange((value: string) => {     //输入内容发生变化时，触发该回调。
          this.changeValue = value; })
       Button('Set caretPosition 1')
        .onClick(() => {
          // 设置光标位置到输入的第一个字符后
          this.controller.caretPosition(1);
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.

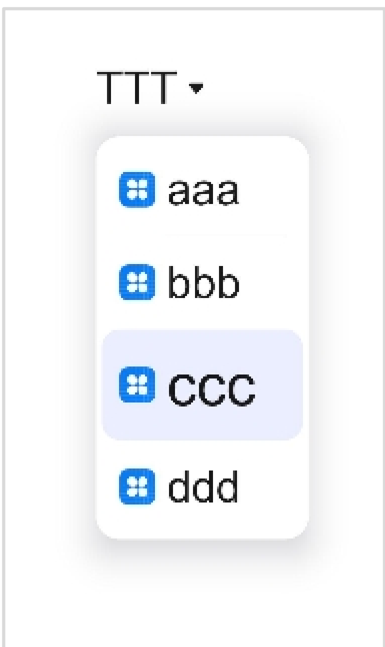


HUAWEI

# Select

- 提供下拉选择菜单，可以让用户在多个选项之间选择。

```
// xxx.ets
@Entry
@Component
struct SelectExample {
  build() {
    Column() {
      Select([{ value: 'aaa', icon: "/common/1.png" }, //下拉选项内容,下拉选项图片
              { value: 'bbb', icon: "/common/2.png" },
              { value: 'ccc', icon: "/common/3.png" },
              { value: 'ddd', icon: "/common/4.png" }])
        .selected(2)  //设置下拉菜单初始选项的索引
        .value('TTT') //设置下拉按钮本身的文本内容
        .font({ size: 30, weight: 400, family: 'serif', style: FontStyle.Normal })
        .selectedOptionFont({ size: 40, weight: 500, family: 'serif', style: FontStyle.Normal })//设置下拉菜单选中项的文本样式
        .optionFont({ size: 30, weight: 400, family: 'serif', style: FontStyle.Normal })//设置下拉菜单项的文本样式
        .onSelect((index: number) => {  //下拉菜单选中某一项的回调
          console.info("Select:" + index)
        })
    }
  }
}
```




Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

# Slider

- 滑动条组件，通常用于快速调节设置值，如音量调节、亮度调节等应用场景。

```
@Entry
@Component
struct SliderExample {
  @State outSetValueOne: number = 40;//滑动条初始值
...
build() {
    Column({ space: 8 }) {
      Row() {
        Slider({
          value: this.outSetValueOne,//当前进度值
          min: 0,//设置最小值
          max: 100,//设置最大值
          style: SliderStyle.OutSet//设置Slider的滑块与滑轨显示样式
        })
          .showTips(true)//设置滑动时是否显示百分比气泡提示
          .onChange((value: number, mode: SliderChangeMode) => {
            this.outSetValueOne = value;
            console.info('value:' + value + 'mode:' + mode.toString());
          })
Text(this.outSetValueOne.toFixed(0)).fontSize(12) // toFixed(0)将滑动条返回值处理为整数精度
...
      }   .width('80%') }.width('100%')}}
```

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

# Stepper、StepperItem

- Stepper：步骤导航器，适用于引导用户按照步骤完成任务的导航场景。
- StepperItem：用作Stepper组件的页面子组件。

```
@Entry
@Component
struct StepperExample {
  @State currentIndex: number = 0
  @State firstState: ItemState = ItemState.Normal
  build() {
    Stepper({
      index: this.currentIndex //设置步骤导航器显示第几个StepperItem
    }) {
      //第一个步骤页
      StepperItem() {
        Text('Page One')
      }
      .nextLabel('Next')        //设置右侧文本按钮内容，最后一页默认值为“开始”，其余页默认值为“下一步”
      .prevLabel('Previous') //设置左侧文本按钮内容，第一页没有左侧文本按钮，当步骤导航器大于一页时，除第一页外默认值都为“返回
      .status(this.firstState)  //设置步骤导航器nextLabel的显示状态
    }
  }
}
```

Page One

下一步 >

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# Video

- 视频播放组件。使用网络视频时，需要申请权限ohos.permission.INTERNET。

```
@Entry
@Component
struct VideoCreateComponent {
  @State srcs: Resource = $rawfile('video1');
 @State previewUri: Resource = $r('app.media.icon')
controller: VideoController = new VideoController();
  build() {
    Column() {
      Video({
        src: this.srcs, //视频的数据源，支持本地视频和网络视频
        previewUri: this.previewUris, //视频未播放时的预览图片路径，默认不显示图片
        currentProgressRate:PlaybackSpeed.Speed_Forward_1_00_X,   //视频播放倍速
        controller: this.controller       //控制器
      })
        .autoPlay(true)     //设置是否自动播放
        .controls(true)      //设置控制视频播放的控制栏是否显示
        .onStart(() => {         //播放时触发该事件
                  console.error('onStart');
                })
        .onPause(() => {          //暂停时触发该事件
                  console.error('onPause');
                })
        .onFinish(() => {          //播放结束时触发该事件
})
```

Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

# 本章小结

- 本章主要介绍了ArkTS的相关组件,介绍了组件的功能以及核心属性，通过实际的组件应用案例，帮助开发者理解和区分ArkTS中的各类组件，具备在实际应用开发中熟练使用ArkTS组件的能力。

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

# 思考题

1. （单选题）某开发者在开发通讯录应用时，需要通过首字母来索引联系人的名字，可以使用以下哪项组件实现该功能？（ ）

   A. AlphabetIndexer

   B. Badge

   C. Cloumn

   D. Grid

Copyright © Huawei Technologies Co., Ltd. All rights reserved.



HUAWEI

# 思考题

2. （单选题）某开发者在开发一款新闻软件时，需要在首页实现一个轮播图的效果，可以使用以下哪一项组件？（ ）

   A. List

   B. Stack

   C. Row

   D. Swpier



Copyright © Huawei Technologies Co., Ltd. All rights reserved.

HUAWEI

# 学习推荐

- 官方学习网站
  - HarmonyOS官网：https://developer.harmonyos.com/
  - HarmonyOS应用开发文档：https://developer.huawei.com/consumer/cn/
  - OpenHarmony官网：https://edu.huaweicloud.com/
  - 华为开发者论坛：https://developer.huawei.com/consumer/cn/forum/

华为云开发者学堂

Copyright © Huawei Technologies Co., Ltd. All rights reserved.

# 感谢

版权所有©2024，华为技术有限公司，保留所有权利。
本资料是华为的保密信息，所有内容仅供华为授权的培训客户内部使用，禁止用于任何其他用途。未经许可，任何人不得对本资料进行复制、修改、改编、也不得将本资料或其任何部分或基于本资料的衍生作品提供给他人。

www.huaweicloud.com