软通教育
ISOFTSTONE EDUCATION

《鸿蒙北向应用开发基础》之

基础组件的使用

软通教育教学教研部

目录

CONTENTS

【课程目标】

- 掌握Text、TextInput、CheckBox、Image、Button等常用组件的使用。

01 UI组件分类

UI组件分类

UI组件是构建页面的核心，每个组件封装了数据和方法，实现独立可视、可交互的功能单元。系统内置的组件从其作用上可分为以下几大类

分类	组件
基础组件	Blank、Button、CheckBox、CheckboxGroup、Divider、Image、Marquee、Menu、MenuItem、MenuItemGroup、Navigation、NavRouter、Progress、Radio、ScrollBar、Search、Select、Slider、Span、Text、TextArea、TextInput、TextTimer、Toggle等
容器组件	Badge、Column、ColumnSplit、Counter、Flex、GridContainer、GridCol、GridRow、Grid、GridItem、List、ListItem、Navigator、Panel、Row、RowSplit、Scroll、Stack、Swiper、Tabs、TabContent等
媒体组件	Video
绘制组件	Circle、Ellipse、Line、Polyline、Polygon、Path、Rect、Shape
画布组件	Canvas、

02 常用基础组件

（1）Text

- Text是文本组件，通常用于展示用户的视图，如显示文章的文字。
- 文本的类型为**string**字符串，文本的创建方式有两种：

直接在组件中添加文本，如：

Text('这是一段文本')

引用Resource资源，通过$r创建Resource类型对象，文件位置为/resources/base/element/string.json。

如：**Text($r('app.string.module_desc'))**

在Text组件内可以添加多个子组件Span来显示文本信息，单独写Span组件不会显示信息，**Text与Span同时配置文本内容内容时，Span内容覆盖Text内容。**

Text组件的属性说明

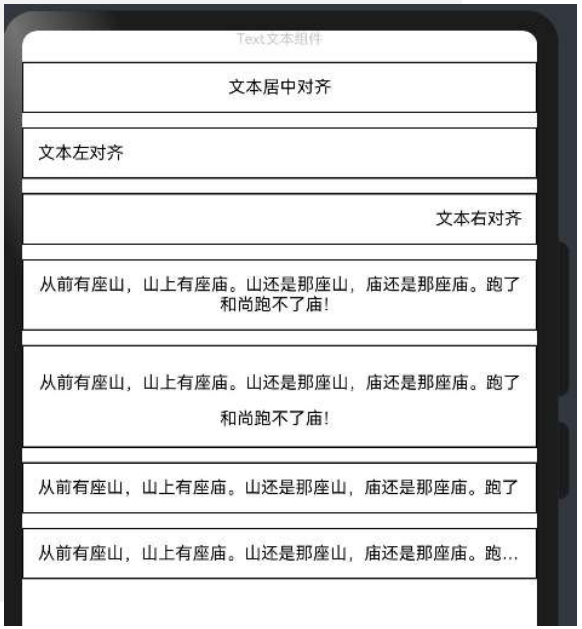
属性	参数类型	描述
textAlign	TextAlign	设置文本在水平方向的对齐方式，文本宽度默认占满Text组件宽度。 TextAlign.Start左对齐（默认值）、TextAlign.Center居中对齐、TextAlign.End右对齐
textOverflow	{overflow: TextOverflow}	• 设置文本超长时的显示方式，需配合maxLines使用，单独设置不生效。 • {overflow: TextOverflow.Clip}文字超长时进行裁剪、 • {overflow: TextOverflow.Ellipsis}文本超长时显示不下的文本用省略号代替。 • {overflow: TextOverflow.None}文本超长时不进行裁剪。 （文本截断是按字截断，英文以单词为最小单位进行截断）
maxLines	number	设置文本的最大行数。默认值：Infinity。默认情况下文本是自动折行的，如果指定此参数，则文本最多不会超过指定的行。如果有多余的文本，可以通过 textOverflow来指定截断方式。
lineHeight	string | number | Resource	设置文本的文本行高，当值不大于0时，不限制文本行高，自适应字体大小。

decoration	{
	type: TextDecorationType,
	color?: ResourceColor
	}
		设置文本装饰线样式及其颜色。其中type用于设置装饰线样式。color为装饰线颜色，默认为black，可选。
		TextDecorationType包含以下几种类型：
		• Underline文字下划线修饰。
		• LineThrough穿过文本的修饰线。
		• Overline文字上划线修饰。
		• None不使用文本装饰线，默认值。

baselineOffset	number | string	设置文本基线的偏移量，默认值0。说明：设置该值为百分比时，按默认值显示。
letterSpacing	number | string	设置文本字符间距，负值表示缩小字符间距，正值表示增大字符间距
textCase	TextCase	设置文本大小写。Normal正常，默认值。LowerCase文本采用全小写。UpperCase文本采用全大写。
minFontSize、 maxFontSize	number | string | Resource	通过minFontSize与maxFontSize自适应字体大小，minFontSize设置文本最小显示字号，maxFontSize设置文本最大显示字号，minFontSize与maxFontSize必须搭配同时使用，以及需配合maxline或布局大小限制一起使用，单独设置不生效。

示例（TextDemo.ets）


```
@Entry
@Component
struct TextDemo {
  build() {
    Column( { space: 10 }) {
      Text('Text文本组件').fontSize(10).fontColor(0xCCCCCC)
      Text('文本居中对齐')
        .textAlign(TextAlign.Center)
        .fontSize(12)
        .border({ width: 1 })
        .padding(10)
        .width('100%')
      Text('文本左对齐')
        .textAlign(TextAlign.Start)
        .fontSize(12)
        .border({ width: 1 })
        .padding(10)
        .width('100%')
      Text('文本右对齐')
        .textAlign(TextAlign.End)
        .fontSize(12)
        .border({ width: 1 })
        .padding(10)
        .width('100%')
```




```
      //多行文本自动换行
      Text('从前有座山，山上有座庙。山还是那座山，庙还是那座庙。跑了和尚跑不了庙！')
        .textAlign(TextAlign.Center)
        .fontSize(12)
        .border({ width: 1 })
        .padding(10)
        .width('100%')
      //行高
      Text('从前有座山，山上有座庙。山还是那座山，庙还是那座庙。跑了和尚跑不了庙！')
        .textAlign(TextAlign.Center)
        .fontSize(12)
        .border({ width: 1 })
        .padding(10)
        .width('100%')
        .lineHeight(25)
    //摘要显示
      //最大行数，超出maxLines截断处理
      Text('从前有座山，山上有座庙。山还是那座山，庙还是那座庙。跑了和尚跑不了庙！')
        .maxLines(1)
        .fontSize(12)
        .border({ width: 1 })
        .padding(10)
        .width('100%')
      //超出maxLines显示省略号
      Text('从前有座山，山上有座庙。山还是那座山，庙还是那座庙。跑了和尚跑不了庙！')
        .maxLines(1)
        .textOverflow({ overflow:TextOverflow.Ellipsis })
        .fontSize(12)
        .border({ width: 1 })
        .padding(10)
        .width('100%')
  }}}
```

（2）TextInput

- **作用：**单行文本输入框
- **接口：**TextInput(value?:{placeholder?: ResourceStr, text?: ResourceStr, controller?: TextInputController})

参数	类型	必填	描述
placeholder	ResourceStr	否	设置无输入时的提示文本
text	ResourceStr	否	设置输入框当前的文本内容
controller	TextInputController	否	设置TextInput控制器

- TextInputController：TextInput组件控制器
- controller：TextInputController = new TextInputController()
- caretPosition：光标输入位置，caretPosition(value: number): void

TextInput组件的属性说明

属性	参数类型	描述
type	InputType	设置输入框类型。Normal基本输入模式，默认值、Password密码输入模式、Email邮箱地址输入模式、Number纯数字输入模式、PhoneNumber电话号码输入模式
placeholderColor	ResourceColor	设置placeholder文本颜色。
placeholderFont	Font	设置placeholder文本样式。
caretColor	ResourceColor	设置输入框光标颜色。
maxLength	number	设置文本的最大输入字符数。
copyOption	CopyOptions	设置输入的文本是否可复制。 None不支持复制、InApp支持应用内复制、LocalDevice支持设备内复制。
showPasswordIcon	boolean	密码输入模式时，输入框末尾的图标是否显示。默认值：true
textAlign	TextAlign	设置文本在水平方向的对齐方式，文本宽度默认占满Text组件宽度。 TextAlign.Start左对齐（默认值）、TextAlign.Center居中对齐、TextAlign.End右对齐

常用事件

名称	描述
onChange(callback: (value: string) => void)	输入内容发生变化时，触发该回调
onSubmit(callback: (enterKey: EnterKeyType) => void)	按下输入法回车键触发该回调，返回值为当前输入法回车键的类型
onEditChange(callback: (isEditing: boolean) => void)	输入状态变化时，触发该回调。isEditing为true表示正在输入
onCopy(callback:(value: string) => void)	长按输入框内部区域弹出剪贴板后，点击剪切板复制按钮，触发该回调。 value：复制的文本内容
onCut(callback:(value: string) => void)	长按输入框内部区域弹出剪贴板后，点击剪切板剪切按钮，触发该回调。 value：剪切的文本内容
onPaste(callback:(value: string) => void)	长按输入框内部区域弹出剪贴板后，点击剪切板粘贴按钮，触发该回调。 value：粘贴的文本内容

示例（TextInputDemo.ets）


```
@Entry
@Component
struct TextInputDemo {
  @State text: string = ''
  controller: TextInputController = new TextInputController()

  build() {
    Column({ space: 10 }){
      Text('TextInput案例演示')
        .fontSize(15)
        .fontColor(0xcccccc)
      TextInput({ placeholder: '请输入你的名字...',controller: this.controller })
        .placeholderColor(Color.Grey)
        .placeholderFont({ size: 14, weight: 400 })
        .caretColor(Color.Blue)
        .width(400)
        .height(40)
        .margin(20)
        .fontSize(14)
        .fontColor(Color.Black)
        .onChange((value: string) => {
          this.text = value
        })
      Text(`Hello ${this.text}`)
        Button('set caretPosition 1')
          .margin(15)
          .onClick(() => {
            //将光标移动到第一个字符后
            this.controller.caretPosition(1)
          })
      // 密码输入框
        TextInput({ placeholder: '请输入你的密码...'})
          .width(400)
          .height(40)
          .margin(20)
          .type(InputType.Password)
          .maxLength(9)
          .showPasswordIcon(true)
        //内联风格输入框
        TextInput({ placeholder: '内联风格' })
          .width(400)
          .height(50)
          .borderRadius(0)
          .style(TextInputStyle.Inline)
      }.width('100%')
    }
}
```

（3）TextArea

- **作用：**可以输入多行文本
- **接口：**TextArea(value?:{placeholder?: ResourceStr, text?: ResourceStr, controller?: TextAreaController}

参数	类型	必填	描述
placeholder	ResourceStr	否	设置无输入时的提示文本
text	ResourceStr	否	设置输入框当前的文本内容
controller	TextAreaController	否	设置TextArea控制器

- **属性：**placeholderColor、placeholderFont、textAlign、caretColor、copyOption等属性和TextInput类似
- 事件：
  - onChange、onCopy、onCut、onPaste同样可以参考TextInput
  - TextAreaController：TextArea组件的控制器，通过它操作TextArea组件
  - controller: TextAreaController = new TextAreaController()
  - caretPosition：光标位置，caretPosition(value: number): void

示例（TextAreaDemo.ets）


```
@Entry
@Component
struct TextAreaDemo {
  controller: TextAreaController = new TextAreaController()
  @State text: string = ''

  build() {
    Column() {
      TextArea({ placeholder:'请吟诗一首...', controller: this.controller})
        .placeholderColor(Color.Grey)
        .textAlign(TextAlign.Center)
        .caretColor(Color.Yellow)
        .fontSize(20)
        .fontWeight(FontWeight.Bolder)
        .fontColor(Color.Red)
    }
  }
}
```

(4) TextPicker

- **作用：**滑动选择文本内容
- **接口：**TextPicker(options?: {range: string[]|Resource, selected?: number, value?: string})
- 根据range指定的选择范围创建文本选择器

参数	类型	必填	描述
range	string[] | Resource	是	选择器的数据选择列表
selected	number	否	设置默认选中项在数组中的索引值。默认值：0
value	string	否	设置默认选中项的值，优先级低于selected，默认值：第一个元素值

- **事件：**defaultPickerItemHeight：number | string类型，设置Picker各选择项的高度
- **属性：**onChange(callback: (value: string, index: number) => void)：滑动选中TextPicker文本内容后，触发该回调。value: 当前选中项的文本， index: 当前选中项的索引值

示例（TextPickerDemo.ets）


```
@Entry
@Component
struct TextPickerDemo {
  private students: string[] = ['刘备','关羽','张飞','赵云']
  private select: number = 1

  build() {
    Column() {
      TextPicker({ range: this.students,selected: this.select })
        .onChange((value: string, index: number) => {
          console.info('选项:value: ' + value + ', index: ' + index)
        })
    }
  }
}
```


```
[phone]01-07 08:40:08.243 33880 42080 I 03b00/JSApp: app Log: 选项:value: 张飞, index: 2
[phone]01-07 08:40:08.475 33880 42080 I 03b00/JSApp: app Log: 选项:value: 赵云, index: 3
```

（5）Checkbox

作用： 提供多选框组件，通常用于某选项的打开或关闭

接口： Checkbox(options?: {name?: string, group?: string })

参数	类型	必填	描述
name	string	否	多选框名称
group	string	否	多选框的群组名称

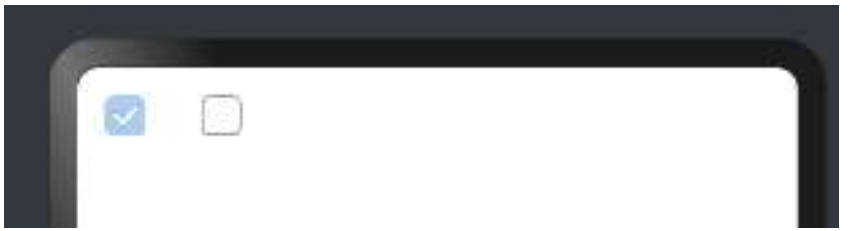
属性：

名称	类型	描述
select	boolean	设置多选框是否选中，默认值：false
selectedColor	ResourceColor	设置多选框选中状态颜色

事件： onChange(callback: (value: boolean) => void)：当选中状态发生变化时，触发该回调，value为true时，表示已选中，value为false时，表示未选中。

示例（CheckBoxDemo.ets）


```
@Entry
@Component
struct CheckBoxDemo {
  build() {
    Row() {
      Checkbox({name: 'box1', group: 'ckboxgroup'})
        .select(true)
        .selectedColor(0xabcdef)
        .onChange((value:boolean) => {
          console.info('box1:'+value)
        })
      Checkbox({name: 'box2', group: 'ckboxgroup'})
        .selectedColor(Color.Green)
        .onChange((value:boolean) => {
          console.info('box2:'+value)
        })
    }
  }
}
```



```
[phone]01-07 09:20:05.702 37296  3152 I 03b00/JSApp: app Log: box2:true
[phone]01-07 09:20:06.767 37296  3152 I 03b00/JSApp: app Log: box1:true
[phone]01-07 09:20:49.272 37296  3152 I 03b00/JSApp: app Log: box2:false
```

（6）CheckboxGroup

作用：多选框群组，用于控制多选框全选或者不全选状态

接口：CheckboxGroup(options?: { group?: string })

group：群组名称，string类型

创建多选框群组，可以控制群组内的Checkbox全选或者不全选，group值相同的Checkbox和CheckboxGroup为同一群组

属性：

名称	类型	描述
selectAll	boolean	设置是否全选，默认值：false
selectedColor	ResourceColor	设置被选中或部分选中状态的颜色

事件：onChange (callback: (event: CheckboxGroupResult) => void)：CheckboxGroup的选中状态或群组内的Checkbox的选中状态发生变化时，触发回调

CheckboxGroup

- **CheckboxGroupResult对象**

名称	类型	描述
name	Array<string>	群组内所有被选中的多选框名称
status	SelectStatus	选中状态

- **SelectStatus枚举说明**

名称	描述
All	全部选择
Part	部分选择
None	没有选择

示例（CheckboxGroupDemo.ets）


```
@Entry
@Component
struct CheckboxGroupDemo {

  build() {
    Column() {
      CheckboxGroup({ group:'chbxGroup'})
        .selectedColor(Color.Blue)
        .onChange((itemName: CheckboxGroupResult) => {
          console.info('checkbox:'+ JSON.stringify(itemName) )
        })
      Text('全选').fontSize(20)
      Checkbox({name:'chkbox1', group:'chbxGroup'})
        .selectedColor(0xabcdef)
        .onChange((value: boolean) => {
          console.info('chkbox1:'+value)
        })
      Text('西瓜').fontSize(20)
      Checkbox({name:'chkbox2', group:'chbxGroup'})
        .selectedColor(0xabcdef)
        .onChange((value: boolean) => {
          console.info('chkbox2:'+value)
        })
      Text('车厘子').fontSize(20)
      Checkbox({name:'chkbox3', group:'chbxGroup'})
        .selectedColor(0xabcdef)
        .onChange((value: boolean) => {
          console.info('chkbox3:'+value)
        })
      Text('番茄').fontSize(20)}}}
```


```
03b00/JSApp: app Log: checkbox:{"name":[],"status":2}
[phone]01-07 14:38:46.738 25964 39516 I 03b00/JSApp: app Log: chkbox1:false
[phone]01-07 14:38:57.902 25964 39516 I 03b00/JSApp: app Log: checkbox:{"name":["chkbox1"],"status":1}
[phone]01-07 14:39:00.947 25964 39516 I 03b00/JSApp: app Log: chkbox2:true
[phone]01-07 14:39:06.976 25964 39516 I 03b00/JSApp: app Log: checkbox:{"name":["chkbox1","chkbox2","chkbox3"],"status":0}
[phone]01-07 14:39:07.145 25964 39516 I 03b00/JSApp: app Log: chkbox3:true
```

（7）Radio

作用： 单选框

接口： Radio(options: {value: string, group: string})

参数	类型	必填	描述
value	string	是	当前单选框的值
group	string	是	组名，相同组名的Radio只能有一个被选中

属性： checked：选择状态，boolean类型，默认值false

事件： onChange(callback: (isChecked: boolean) => void)：单选框选中状态改变时触发回调，isChecked为true时，选中，isChecked为false时，未选中

示例（RadioDemo.ets）


```
@Entry
@Component
struct RadioDemo {
  build() {
    Column() {
      Text('西瓜')
        .fontSize(20)
      Radio({value:'xiagua', group: 'rdGroup'})
        .checked(true)
        .height(50)
        .width(50)
        .onChange((isChecked: boolean) => {
          console.log('xigua is:'+ isChecked)
        })
      Text('车厘子').fontSize(20)
      Radio({value:'chelizi', group: 'rdGroup'})
        .height(50)
        .width(50)
        .onChange((isChecked: boolean) => {
          console.log('chelizi is:'+ isChecked)
        })
      Text('番茄').fontSize(20)
      Radio({value:'fanqie', group: 'rdGroup'})
        .height(50)
        .width(50)
        .onChange((isChecked: boolean) => {
          console.log('fanqie is:'+ isChecked)
        })
    }
  }
}
```


```
[phone]01-07 14:54:16.750 41776 11080 I 03b00/JSApp: app Log: chelizi is:true
[phone]01-07 14:54:17.763 41776 11080 I 03b00/JSApp: app Log: fanqie is:true
[phone]01-07 14:54:30.496 41776 11080 I 03b00/JSApp: app Log: fanqie is:true
[phone]01-07 14:54:32.865 41776 11080 I 03b00/JSApp: app Log: chelizi is:true
[phone]01-07 14:54:35.705 41776 11080 I 03b00/JSApp: app Log: xigua is:true
```

（8）Select

作用： 下拉菜单

接口： Select(options: Array<SelectOption>)

SelectOption对象	参数	类型	必填	描述
	value	ResourceStr	是	下拉选项内容
	icon	ResourceStr	否	下拉选项图标

属性：

名称	类型	描述
selected	number	下拉菜单初始选项的索引，从0开始
value	string	下拉按钮文本
font	Font	下拉按钮文本样式
fontColor	ResourceColor	下拉按钮文本颜色

名称	类型	描述
selectedOptionBgColor	ResourceColor	选中项背景色
selectedOptionFont	Font	选中项文本样式
optionBgColor	ResourceColor	下拉菜单选项背景色
optionFont	Font	下拉菜单选项文本样式

事件： onSelect(callback: (index: number, value?: string) => void)：下拉菜单选中某项时回调，index：选中索引，value：选中值

示例（SelectDemo.ets）


```
@Entry
@Component
struct SelectDemo {
  build() {
    Column() {
      Select([
        {value: '恐怖南瓜', icon: '/path/image0.png'},
        {value: '毒苹果', icon: '/path/image1.png'},
        {value: '阴森大树', icon: '/path/image2.png'},
        {value: '快乐椰子', icon: '/path/image3.png'},
      ]).selected(1)
        .value('魔法植物')
        .onSelect((index: number) => {
          console.info('select:'+index)
        })
    }
  }
}
```


```
[phone]01-07 17:02:33.401 46444 49104 I 03b00/JSApp: app Log: select:2
[phone]01-07 17:02:36.748 46444 49104 I 03b00/JSApp: app Log: select:3
```

（9）Toggle

作用： 组件提供勾选框、状态按钮及开关样式，仅当ToggleType为Button时，可以包含子组件：

接口： Toggle(options: { type: ToggleType, isOn?: boolean })

参数	类型	描述
Type	ToggleType	开关类型
isOn	Boolean	开关是否打开，打开：true；关闭：false，默认为false

ToggleType枚举：

名称	描述
Checkbox	单选框样式
Button	按钮样式，子组件使用Text会显示相应文本
Switch	开关样式

周围间隙可以通过通用属性padding调整

（以Switch默认值为例）

```
{
top: 12 vp,
right: 12 vp,
bottom: 12 vp,
left: 12 vp
}
```

示例（ToggleDemo.ets）

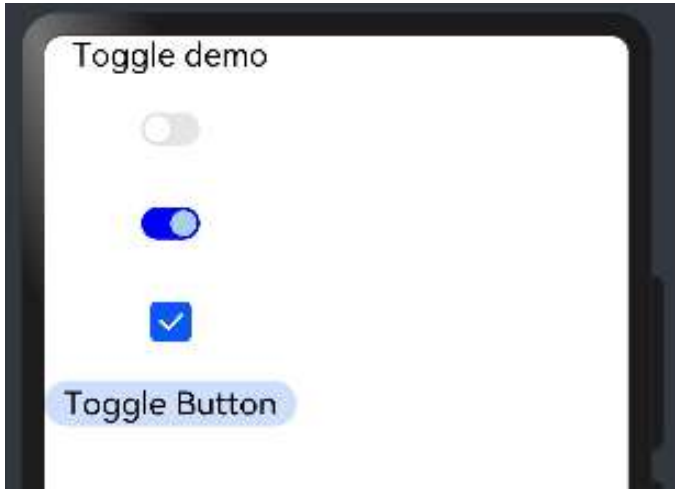
属性：

名称	参数	描述
selectedColor	ResourceColor	打开状态背景色
switchPointColor	ResourceColor	指定type为Toggle.Switch时原型滑块颜色

事件： onChange(callback: (isOn: boolean) => void)：开关状态切换时触发该事件

示例（ToggleDemo.ets）


```
@Entry
@Component
struct ToggleDemo {
  build() {
    Column({ space: 10 }) {
      Text('Toggle demo').fontSize(20)
      Toggle({ type: ToggleType.Switch, isOn: false })
      Toggle({ type: ToggleType.Switch, isOn: true })
        .selectedColor(Color.Blue)
        .switchPointColor(0xabcdef)
        .onChange((isOn: boolean) => {
          console.info('switch status:' + isOn)
        })
      Toggle({ type: ToggleType.Checkbox, isOn: true })
        .width(25)
        .height(25)
      Toggle({ type: ToggleType.Button, isOn: true }) {
        Text('Toggle Button').fontSize(20).padding({ left: 12, right: 12 })
      } }}}
```




```
[phone]01-08 14:20:47.882 42760 30500 I 03b00/JSApp: app Log: switch status:false
[phone]01-08 14:20:50.817 42760 30500 I 03b00/JSApp: app Log: switch status:true
```

（10）Slider

作用： 滑动条，可快速调节设置值，如音量、亮度调节等

接口： Slider(options?: {value?: number, min?: number, max?: number, step?: number, style?: SliderStyle, direction?: Axis, reverse?: boolean})

参数	类型	必填	描述
value	number	否	当前值，默认0
min	number	否	最小值，默认0
max	number	否	最大值，默认100
step	number	否	滑动步长，默认1
style	SliderStyle	否	滑块和滑轨样式，默认值：SliderStyle.OutSet
direction	Axis	否	滑动条方向，水平或者竖直
reverse	boolean	否	滑动条取值范围是否反向，默认从左到右，从上到下

OutSet：滑块在滑轨上
InSet：滑块在滑轨内

Slider

属性：

名称	类型	描述
showSteps	boolean	显示步长刻度
showTips	boolean	显示百分比提示
blockColor	ResourceColor	滑块颜色
trackColor	ResourceColor	滑轨颜色
selectedColor	ResourceColor	已滑动部分颜色

事件： onChange(callback: (value: number, mode: SliderChangeMode) => void)：滑动时触发回调，value，滑动进度值，mode：拖到状态

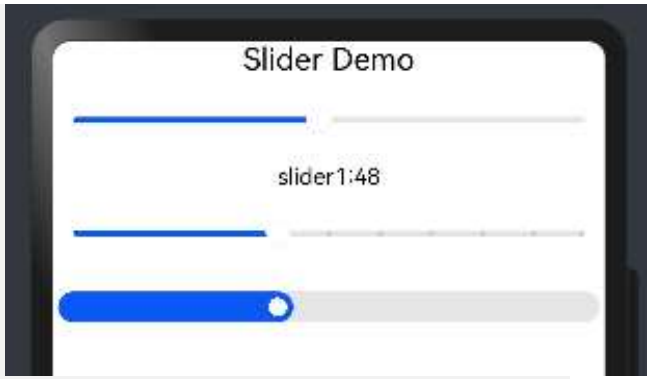
SliderChangeMode枚举：

名称	值	描述
Begin	0	开始拖动
Moving	1	正在拖动
End	2	结束拖动
Click	3	点击定位到某位置

示例（SliderDemo.ets）


```
@Entry
@Component
struct SliderDemo {
  @State slider1value: number = 40
  @State slider2value: number = 40

  build() {
    Column({ space: 8 }) {
      Text('Slider Demo').fontSize(20)
      Slider({}) //都使用默认值，但{}不可省略
        .onChange((value: number, mode: SliderChangeMode) => {
          this.slider1value = value
          console.info('value:' + value + 'mode:' + mode.toString())
        })
      //处理成整数后显示
      Text(`slider1:${this.slider1value.toFixed(0)}`).fontSize(15)
```




```
      Slider({
        value: this.slider2value,
        step: 10,
        min: 0,
        max: 100,
        style: SliderStyle.OutSet
      })
        .showSteps(true)
        .showTips(true)

      Slider({
        value: 40,
        step: 10,
        min: 0,
        max: 100,
        style: SliderStyle.InSet
      })

    }
  }
```

```
[phone]01-08 15:45:20.705 46992 52216 I 03b00/JSApp: app Log: value:76mode:3
[phone]01-08 15:47:14.644 45964 50700 I 03b00/JSApp: app Log: value:48mode:0
[phone]01-08 15:47:14.725 45964 50700 I 03b00/JSApp: app Log: value:48mode:3
```

（11）Progress

作用： 进度条，用于显示内容加载或操作处理的进度

接口： Progress(options: {value: number, total?: number, type?: ProgressType})

参数	类型	描述
value	number	当前进度值
total	number	进度总长，默认100
type	ProgressType	进度条类型，默认值：ProgressType.Linear

ProgressType枚举：

名称	描述
Linear	线性样式
Ring	环形
Eclipse	圆形
ScaleRing	环形有刻度
Capsule	胶囊样式

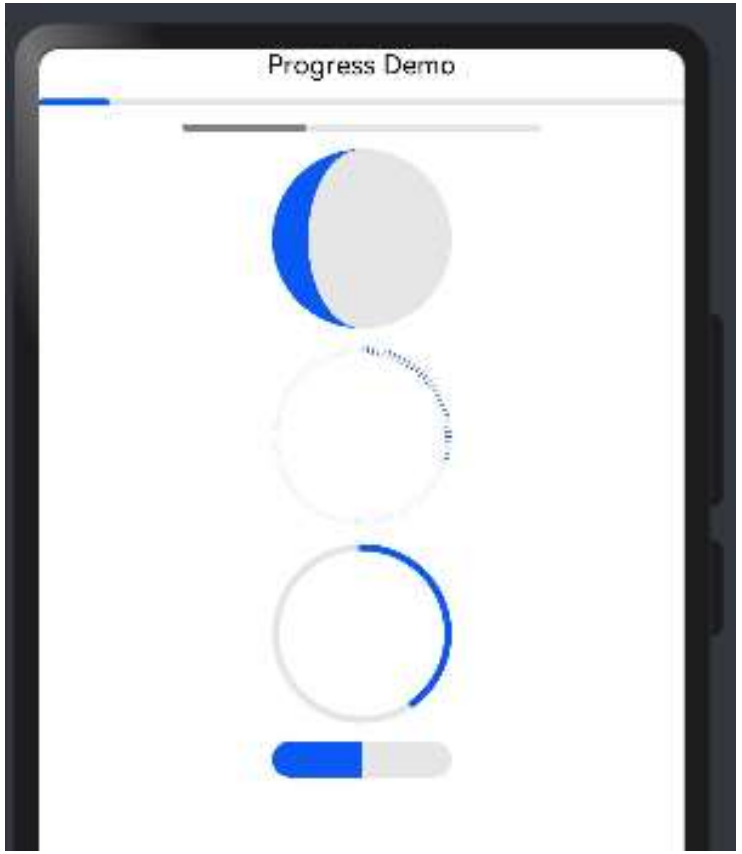
属性：

名称	类型	描述
value	number	设置当前进度值
color	ResourceColor	进度条前景色

示例（ProgressDemo.ets）


```
@Entry
@Component
struct ProgressDemo {
  build() {
    Column({ space: 10 }) {
      Text('Progress Demo').fontSize(15)
      Progress({ value: 10 })
      Progress({ value: 20, total: 150, type: ProgressType.Linear })
        .color(Color.Grey)
        .value(50)
        .width(200)
      Progress({ value: 20, total: 100, type: ProgressType.Eclipse }).width(100)
      Progress({ value: 30, total: 100, type: ProgressType.ScaleRing }).width(100)
      Progress({ value: 40, total: 100, type: ProgressType.Ring }).width(100)
      Progress({ value: 50, total: 100, type: ProgressType.Capsule }).width(100).height(20)

    }
  }
}
```



（12）TimePicker

作用： 滑动选择时间

接口： TimePicker(options?: {selected?: Date})，默认使用12小时制

参数： selected：Date类型，选中项时间，默认为当前系统时间

属性： useMilitaryTime：boolean类型，展示时间是否是24小时制，默认值：false

事件： onChange(callback: (value: TimePickerResult) => void)：选择时间时触发

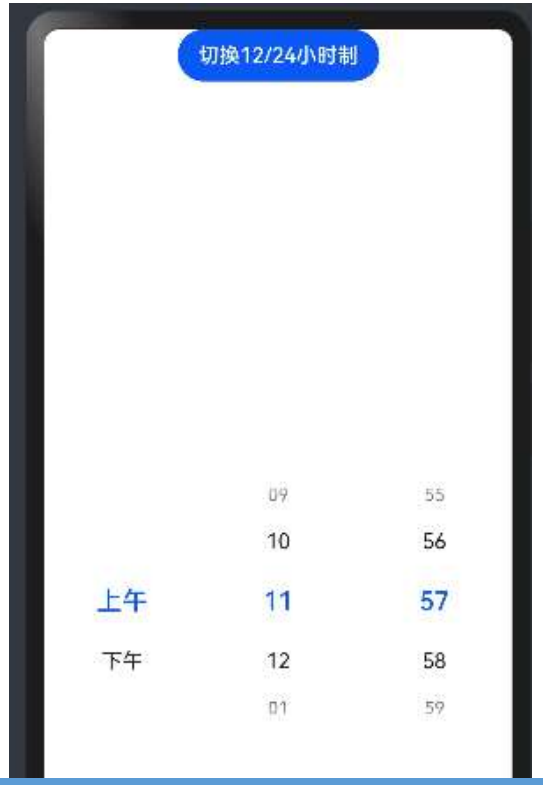
TimePickerResult对象：

名称	类型	描述
hour	number	选中的小时
minute	number	选中的分钟

示例（TimePickerDemo.ets）


```
@Entry
@Component
struct TimePickerDemo {
  @State is24Hour: boolean = false
  private pickerTime: Date = new Date()

  build() {
    Column({ space: 10 }) {
      Button('切换12/24小时制')
        .onClick(() => {
          this.is24Hour = !this.is24Hour
        })
      TimePicker({})
        .useMilitaryTime(this.is24Hour)
        .onChange((value: TimePickerResult) => {
          this.pickerTime.setHours(value.hour,value.minute)
          console.info('selected hour:' + value.hour + ',minute:' + value.minute)
        })
    }
  }
}
```




```
[phone]01-09 08:58:04.446 16656 24264 I 03b00/JSApp: app Log: selected hour:9,minute:57
[phone]01-09 08:58:04.584 16656 24264 I 03b00/JSApp: app Log: selected hour:10,minute:57
[phone]01-09 08:58:04.772 16656 24264 I 03b00/JSApp: app Log: selected hour:11,minute:57
```

（13）DatePicker

作用： 日期滑动选择器

接口： DatePicker(options?: {start?: Date, end?: Date, selected?: Date})

根据指定范围的日期创建日期滑动选择器

参数	类型	必填	描述
start	Date	否	起始值：默认：Date('1970-1-1')
end	Date	否	结束日期，默认：Date('2100-12-31')
selected	Date	否	选中日期，默认为当前日期

属性： lunar：boolean类型，是否为农历，true为农历，false为公历，默认为false

事件： onChange(callback: (value: DatePickerResult) => void)：选择日期时触发

- DatePickerResult对象：
- year：选中年，number类型
- month：选中月，number类型，0~11，0表示1月，实际月份注意+1
- day：选中日，number类型

示例（DatePickerDemo.ets）


```
@Entry
@Component
struct DatePickerDemo {
  @State isLunar: boolean = false
  private selectDate: Date = new Date()
  build() {
    Column() {
      Button('切换公历/农历')
        .margin({ top: 20 })
        .onClick(() => {
          this.isLunar = !this.isLunar
        })
      DatePicker({
        start: new Date('1970-1-1'),
        end: new Date('2100-1-1'),
        selected: this.selectDate
      })
        .lunar(this.isLunar)
        .onChange((value: DatePickerResult) => {
          this.selectDate.setFullYear(value.year, value.month, value.day)
          console.info('select date:' + JSON.stringify(value))
        })
    }
  }
}
```

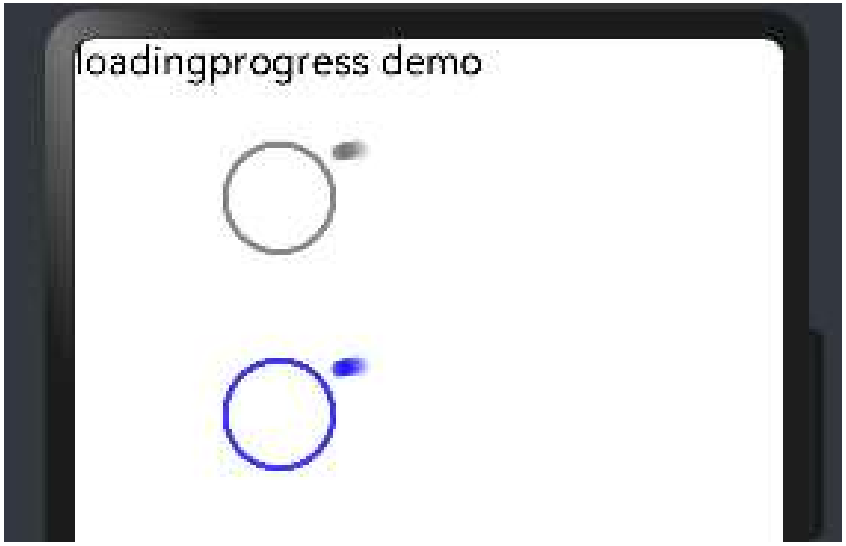

```
[phone]01-09 10:41:37.266 56848 58096 I 03b00/JSApp: app Log: select date:{"year":2023,"month":1,"day":10}
[phone]01-09 10:41:37.624 56848 58096 I 03b00/JSApp: app Log: select date:{"year":2023,"month":1,"day":11}
[phone]01-09 10:41:39.124 56848 58096 I 03b00/JSApp: app Log: select date:{"year":2023,"month":2,"day":11}
```

(14) LoadingProgress

作用： 表示加载中的动画效果

接口： LoadingProgress()

属性： color：ResourceColor类型，加载进度条前景色



示例（LoadingProgressDemo.ets）

```
@Entry
@Component
struct LoadingProgressDemo {
  build() {
    Column({ space: 10 }) {
      Text('loadingprogress demo').fontSize(20)
      LoadingProgress()
        .width(100)
        .height(100)
      LoadingProgress()
        .color(Color.Blue)
        .width(100)
        .height(100)
    }}}
```

(15) Search

作用： 搜索框

接口： Search(options?: { value?: string; placeholder?: string; icon?: string; controller?: SearchController })

参数	类型	必填	描述
value	string	否	搜索框显示内容
placeholder	string	否	提示文本
icon	string	否	搜索图标
controller	SearchController	否	搜索组件控制器

属性：

名称	类型	描述
searchButton	string	搜索按钮文本，默认无搜索按钮
placeholderColor	ResourceColor	占位文本颜色
placeholderFont	Font	占位文本样式
textFont	Font	输入文本样式
textAlign	TextAlign	文本对齐

事件：

名称	描述
onSubmit(callback: (value: string) => void)	提交搜索时触发，value：搜索框中文本
onChange(callback: (value: string) => void)	输入发生变化时触发，value：搜索框中文本
onCopy(callback: (value: string) => void)	拷贝时触发，value：复制文本
onCut(callback: (value: string) => void)	剪切时触发，value：剪切文本
onPaste(callback: (value: string) => void)	粘贴时触发，value：粘贴文本

- SearchController：搜索组件控制器，当前仅可控制光标位置
- controller: SearchController = new SearchController()
- caretPosition：caretPosition(value: number): void,光标位置

示例（SearchDemo.ets）


```
@Entry
@Component
struct SearchDemo {
  @State changeVal: string = ''
  @State submitVal: string = ''
  controller: SearchController = new SearchController()

  build() {

    Column({ space: 10 }) {
      Text(`提交内容:${this.submitVal}`).fontSize(20)
      Text(`正在输入:${this.changeVal}`).fontSize(20)
      Search({ placeholder: '请输入搜索内容...', controller: this.controller })
        .searchButton('搜索')
        .placeholderFont({ size: 15 })
        .textFont({ size: 15 })
        .width(300)
        .height(30)
```

```
        .onSubmit((value: string) => {
          this.submitVal = value
        })
        .onChange((value: string) => {
          this.changeVal = value
        })
      Button('设置光标位置到开头')
        .onClick(() => {
          this.controller.caretPosition(0)
        })
    }
  }
}
```

(16) Image

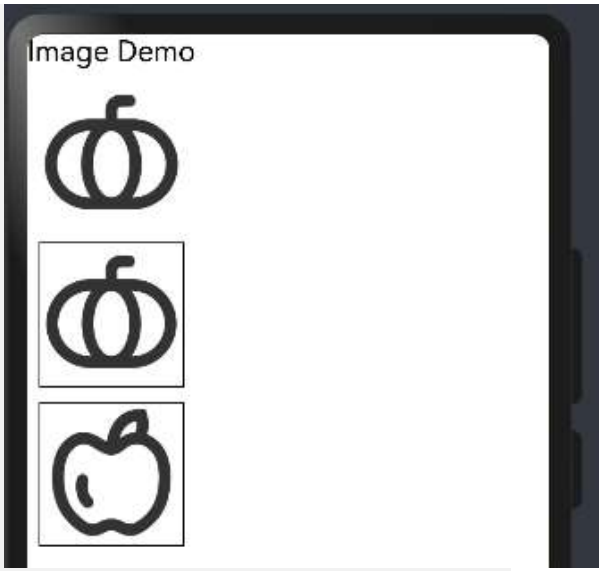
作用： 展示图片，可以展示本地或网络图片

接口： Image(src: string | PixelMap | Resource)

- **src**：必填参数，使用相对路径加载图片时不支持跨包或者模块调用，建议将图标拷贝到resource->base->media目录下并使用$r来引用
- **Resource**：资源引用类型，可以通过$r或者$rawfile创建Resource对象
  - **$r('belonging.type.name')**
    - belonging：系统资源或者应用资源，相应的取值为'sys'和'app'；
    - type：资源类型，支持'color'、'float'、'string'、'media'等；
    - name：资源名称，在资源定义时确定。
  - **$rawfile('filename')**
    - filename：工程中resources/rawfile目录下的文件名称

示例（ImageDemo.ets）


```
@Entry
@Component
struct ImageDemo {
  build() {
    Column({ space: 10 }){
      Text('Image Demo').fontSize(20)
      //使用绝对或相对路径加载图片根目录在ets下
      Image('/path/image0.png')
        .width(100)
        .height(100)
      Image('/path/image0.png')
        .width(100)
        .height(100)
        .border({ width: 1 })
      //建议方式，使用$r方式加载图片，图片需放置路径：src>main>resources>base>media
      Image($r('app.media.image1'))
        .width(100)
        .height(100)
        .border({ width: 1 })
    } }}
```



（17）Button

Button按钮组件，通常用于响应用户的点击操作，其类型包括胶囊按钮、圆形按钮、普通按钮。

Button当做为容器使用时可以通过添加子组件实现包含文字、图片等元素的按钮。其语法如下：

Button(label?: string, options?: { type?: ButtonType, stateEffect?: boolean })

1. 参数label，非必填项，用来用来设置按钮显示的文字。
2. 参数type，非必填项，用来设置按钮显示的样式
3. 参数stateEffect，非必填项，用于设置按钮是否开启点击效果，false关闭，true开启（默认值）。

Button按钮类型

类型说明	示例
• Capsule：胶囊型按钮，默认样式；此类型按钮的圆角自动设置为高度的一半，不支持通过borderRadius属性重新设置圆角。	Button('Capsule',{type:ButtonType.Capsule,stateEffect:false}) .width(300) .height(50) .backgroundColor(Color.Green) 
• Circle：圆形按钮，不支持通过borderRadius属性重新设置圆角。	Button('Circle',{type:ButtonType.Circle,stateEffect:false}) .width(100) .height(100) .backgroundColor(0xFA5C15) 
• Normal：普通按钮（默认不带圆角），此类型的按钮默认圆角为0，支持通过borderRadius属性重新设置圆角。	Button('Normal',{type:ButtonType.Normal,stateEffect:false}) .borderRadius(10) .width(300) .height(50) .backgroundColor(0x0496DB) 

示例代码（ButtonDemo.ets）


```
@Entry
@Component
struct ButtonDemo {

  build() {
    Column({space:10}) {
      Text('Button Demo').fontSize(20)
      Button('默认风格')
      Button('普通按钮', { type: ButtonType.Normal, stateEffect: true})
      Button('禁用', {type: ButtonType.Normal, stateEffect: false})
        .opacity(0.4)
        .borderRadius(8)
        .backgroundColor(Color.Blue)
        .width(100)
```


```
      Button('胶囊按钮', {type: ButtonType.Capsule, stateEffect: true})
        .backgroundColor(Color.Blue)
        .width(100)
      Button('C', {type: ButtonType.Circle, stateEffect: true})
        .width(55)
        .height(55)
      Button({type: ButtonType.Circle, stateEffect: true}){
        LoadingProgress().width(25).height(25).color(0xffffff)
      }.width(55).height(55)
      Button({type: ButtonType.Capsule, stateEffect: true}){
        Row(){
          LoadingProgress().width(20).height(20).color(0xffffff)
          Text('加载中...').fontSize(15).fontColor(0xffffff)
        }.width(90).height(40)
      }.backgroundColor(Color.Blue)
    }
  }
}
```

【课程小结】

- Image组件：通过通过$r资源接口读取到并转换到Resource格式。可以跨包/跨模块引入图片。
- Text组件用来设置文本，字符类型为string，可以包含子组件Span（会覆盖Text组件中设置的文本），属性有textAlign（对齐方式）、textOverflow（超长时的显示方式）、maxLines（最多显示的行数）、decoration（修饰线）、lineHeight（行高）等
- TextInput组件用于输入单行文本，参数placeholder设置无输入时的提示文本，参数text设置输入框当前的文本内容。输入框类型有Normal（文本框）、Password（密码框）、Email（邮箱）、Number（纯数字）、PhoneNumber（电话号码）
- Button按钮组件，type类型有Capsule（胶囊型）、Circle（圆形）、Normal（普通按钮，支持通过borderRadius属性）。