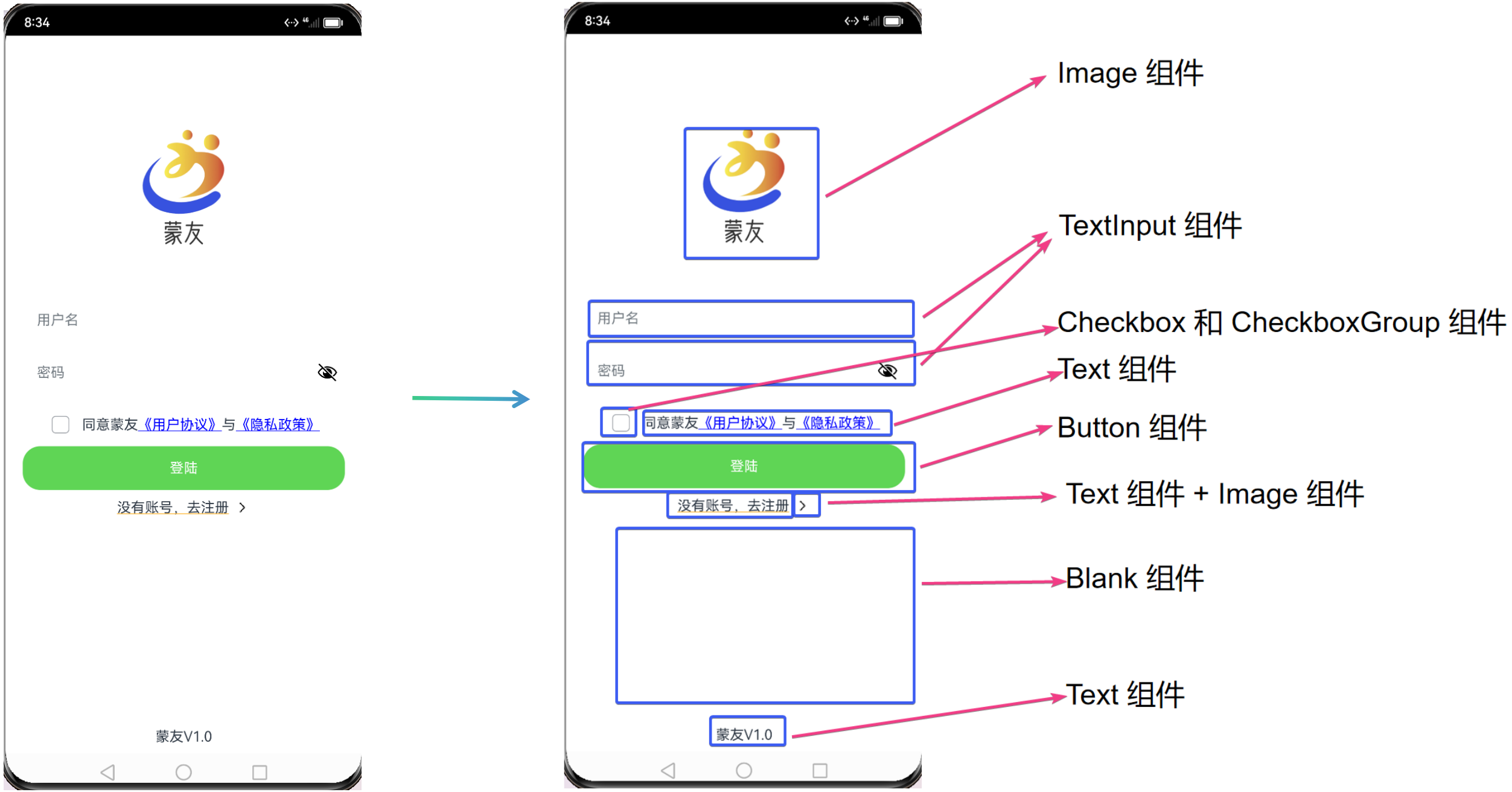
登录页面实现

UI分析


8:34
蒙友
用户名
密码
同意蒙友《用户协议》与《隐私政策》
登陆
没有账号，去注册
蒙友V1.0
Image 组件
TextInput 组件
Checkbox 和 CheckboxGroup 组件
Text 组件
Button 组件
Text 组件 + Image 组件
Blank 组件
Text 组件

线性布局

Row组件-水平布局

接口定义：Row(value?:{space?: number | string })

参数说明：

- space：设置子组件之间的间隙。默认值：0，单位vp。可选值为大于等于0的数字，或者可以转换为数字的字符串。

属性方法：

- justifyContent：设置子组件在水平方向的对齐方式，默认值：FlexAlign.Start
- alignItems：设置子组件在垂直方向的对齐方式，默认值：VerticalAlign.Center。

Row组件-水平布局

FlexAlign枚举类型说明：

名称	描述
Start	元素在主轴方向首端对齐，第一个元素与行首对齐，同时后续的元素与前一个对齐。
Center	元素在主轴方向中心对齐，第一个元素与行首的距离与最后一个元素与行尾距离相同。
End	元素在主轴方向尾部对齐，最后一个元素与行尾对齐，其他元素与后一个对齐。
SpaceBetween	Flex主轴方向均匀分配弹性元素，相邻元素之间距离相同。第一个元素与行首对齐，最后一个元素与行尾对齐。
SpaceAround	Flex主轴方向均匀分配弹性元素，相邻元素之间距离相同。第一个元素到行首的距离和最后一个元素到行尾的距离是相邻元素之间距离的一半。
SpaceEvenly	Flex主轴方向均匀分配弹性元素，相邻元素之间的距离、第一个元素与行首的间距、最后一个元素到行尾的间距都完全一样。

VerticalAlign枚举类型说明：

名称	描述
Top	顶部对齐。
Center	居中对齐，默认对齐方式。
Bottom	底部对齐。

Column组件

接口定义：Column(value?: {space?: string | number})

参数说明：

- space：设置子组件之间的间隙。默认值：0，单位vp。可选值为大于等于0的数字，或者可以转换为数字的字符串。

属性方法：

- justifyContent：设置子组件在垂直方向的对齐方式，默认值：FlexAlign.Start
- alignItems：设置子组件在水平方向的对齐方式，默认值：HorizontalAlign.Center。

Column组件

FlexAlign枚举类型说明：

名称	描述
Start	元素在主轴方向首端对齐，第一个元素与行首对齐，同时后续的元素与前一个对齐。
Center	元素在主轴方向中心对齐，第一个元素与行首的距离与最后一个元素与行尾距离相同。
End	元素在主轴方向尾部对齐，最后一个元素与行尾对齐，其他元素与后一个对齐。
SpaceBetween	Flex主轴方向均匀分配弹性元素，相邻元素之间距离相同。第一个元素与行首对齐，最后一个元素与行尾对齐。
SpaceAround	Flex主轴方向均匀分配弹性元素，相邻元素之间距离相同。第一个元素到行首的距离和最后一个元素到行尾的距离是相邻元素之间距离的一半。
SpaceEvenly	Flex主轴方向均匀分配弹性元素，相邻元素之间的距离、第一个元素与行首的间距、最后一个元素到行尾的间距都完全一样。

HorizontalAlign 枚举类型说明：

名称	描述
Start	按照语言方向起始端对齐。
Center	居中对齐，默认对齐方式。
End	按照语言方向末端对齐。

Text 组件

Text 组件-接口说明

Text (content?: string | Resource): TextAttribute;

参数说明：

- content：文本的内容，默认值为“”。

注意，该组件可以包含子组件Span，如果包含了Span子组件，Text 的内容不生效，内容显示为Span 组件的内容，并且给Text组件设置的样式也不生效。

Text 组件-属性方法

fontColor(value: ResourceColor): TextAttribute;
设置字体颜色

fontSize(value: number | string | Resource): TextAttribute;
设置字体大小

fontStyle(value: FontStyle): TextAttribute;
设置字体样式

fontWeight(value: number | FontWeight | string): TextAttribute;
设置字体粗细

```
//字体样式
declare enum FontStyle {
    Normal,
    Italic
}

//字体粗细

 declare enum FontWeight {
     Lighter,
     Normal,  //400
     Regular,
     Medium,
     Bold,  //700
     Bolder  //900
 }
```

Text 组件-属性方法

textAlign(value: TextAlign): TextAttribute;
设置文本对齐，默认值： TextAlign.Start

lineHeight(value: number | string | Resource): TextAttribute;
设置行高

textOverflow(value: {
    overflow: TextOverflow;
}): TextAttribute;
设置文本超长时的显示方式

decoration(value: {
    type: TextDecorationType;
    color?: ResourceColor;
}): TextAttribute;
设置文本装饰线

```
//设置超长文本的显示方式
declare enum TextOverflow {
    Clip,  //截断
    Ellipsis, //省略号
    None //
}
```

```
//设置文本装饰线
declare enum TextDecorationType {
    None,
    Underline,  //下划线
    Overline,  //上划线
    LineThrough // 下划线
}
```

Text 组件-属性方法

```
textAlign(value: TextAlign): TextAttribute;
```
设置文本对齐

```
textOverflow(value: {
        overflow: TextOverflow;
    }): TextAttribute;
```
设置文本超长时的显示方式

```
decoration(value: {
        type: TextDecorationType;
        color?: ResourceColor;
    }): TextAttribute;
```
设置文本装饰线

```
//设置超长文本的显示方式
declare enum TextOverflow {
    Clip,  //截断
    Ellipsis, //省略号
    None //
}
```

```
//设置文本装饰线
declare enum TextDecorationType {
    None,
    Underline,  //下划线
    Overline,  //上划线
    LineThrough // 下划线
}
```

Text 组件-属性方法

maxLines(value: number): TextAttribute;
设置最大行数。说明：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(value: number | string | Resource): TextAttribute;
设置行高。说明：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，Length为number类型时单位为fp。

TextInput 组件

TextInput 组件-接口定义

TextInput(value?:{placeholder?: ResourceStr, text?: ResourceStr, controller?: TextInputController})

参数说明:

- 参数整体为一个对象。
  - placeholder：设置无输入时的提示文本。
  - text：设置输入框当前的文本内容。
  - controller：设置TextInput控制器

TextInputController 类的方法：

- caretPosition(value: number): void：设置输入光标的位置。

TextInput 组件-属性方法

type(value: InputType): TextInputAttribute;

设置输入的数据类型。默认值为，InputType.Normal，更多取值参考其枚举类型。

InputType 枚举类型说明

名称	描述
Normal	基本输入模式。 支持输入数字、字母、下划线、空格、特殊字符。
Password	密码输入模式。支持输入数字、字母、下划线、空格、特殊字符。密码显示小眼睛图标并且默认会将文字变成圆点。
Email	邮箱地址输入模式。支持数字，字母，下划线，以及@字符（只能存在一个@字符）。
Number	纯数字输入模式。
PhoneNumber[9+]	电话号码输入模式。 支持输入数字、+ 、-、*、#，长度不限。

TextInput 组件-属性方法

placeholderColor(value: ResourceColor): TextInputAttribute;

设置placeholder文本颜色

placeholderFont(value?: Font): TextInputAttribute;

设置placeholder文本样式

Font 枚举类型如下：

名称	类型	必填	说明
size	Length	否	设置文本尺寸，Length为number类型时，使用fp单位。不支持设置百分比字符串。
weight	FontWeight | number | string	否	设置文本的字体粗细，number类型取值[100, 900]，取值间隔为100，默认为400，取值越大，字体越粗。
family	string | Resource	否	设置文本的字体列表。使用多个字体，使用','进行分割，优先级按顺序生效。例如：'Arial, HarmonyOS Sans'。当前支持'HarmonyOS Sans'字体和注册自定义字体。
style	FontStyle	否	设置文本的字体样式。

TextInput 组件-属性方法

enterKeyType(value: EnterKeyType): TextInputAttribute;

设置输入法回车键类型

caretColor(value: ResourceColor): TextInputAttribute;

设置光标的颜色

maxLength(value: number): TextInputAttribute;

设置文本的最大输入字符数

EnterKeyType枚举类型如下：

名称	描述
Go	显示为开始样式。
Search	显示为搜索样式。
Send	显示为发送样式。
Next	显示为下一个样式。
Done	显示为换行样式。

TextInput 组件-属性方法

inputFilter(value: ResourceStr, error?: (value: string) => void): TextInputAttribute;

正则输入匹配 。匹配表达式的输入允许显示，不匹配的输入将被过滤。目前仅支持单个字符匹配，不支持字符串匹配。

- value：设置正则表达式。
- error：正则匹配失败时，返回被过滤的内容。

```
//匹配只能输入整数
TextInput()
  .inputFilter("-?[1-9]\d*", (value:string)=>{
  console.info("error:" + value)
})
```

TextInput 组件-属性方法

showPasswordIcon(value: boolean): TextInputAttribute;
显示密码输入框。密码输入模式时，输入框末尾的图标是否显示。默认值：true

style(value: TextInputStyle): TextInputAttribute;
设置输入框的风格。默认值：TextInputStyle.Default

textAlign(value: TextAlign): TextInputAttribute;
设置输入文本在输入框中的对齐方式。默认值：TextAlign.Start

TextInputStyle枚举类型说明：

名称	描述
Default	默认风格，光标宽1.5vp，光标高度与文本选中底板高度和字体大小相关。
Inline	内联输入风格。文本选中底板高度与输入框高度相同。 内联输入是在有明显的编辑态/非编辑态的区分场景下使用，例如：文件列表视图中的重命名。

TextInput 组件-事件方法

onChange(callback: (value: string) => void)

输入内容发生改变时的事件方法，value就是输入的内容，触发该事件的条件：

1、键盘输入。

2、粘贴、剪切。

3、键盘快捷键Ctrl+v。

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。enterKeyType：输入法回车键类型。具体类型见EnterKeyType枚举说明。

onEditChanged(callback: (isEditing: boolean) => void)【废弃】

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。

TextInput 组件-事件方法

onCopy(callback:(value: string) => void)

onCut(callback:(value: string) => void)

onPaste(callback:(value: string) => void)

在进行复制、剪切、粘贴操作时的回调函数。

Checkbox组件
CheckboxGroup组件

Checkbox组件-接口定义

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

参数说明：

name：多选框的名称

group：多选框的群组名称。未配合使用CheckboxGroup组件时，此值无用。

Checkbox组件-属性方法

select(value: boolean): CheckboxAttribute;

设置多选框是否选中。默认值：false

selectedColor(value: ResourceColor): CheckboxAttribute;

设置多选框选中状态颜色。

Checkbox组件-事件方法

onChange(callback: (value: boolean) => void): CheckboxAttribute;

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。

CheckboxGroup组件-接口定义

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

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。

CheckboxGroup组件-属性方法

selectAll(value: boolean): CheckboxGroupAttribute;

设置是否全选。默认值：false，若同组的Checkbox显式设置select，则Checkbox的优先级高。

selectedColor(value: ResourceColor): CheckboxGroupAttribute;

设置被选中或部分选中状态的颜色。

CheckboxGroup组件-事件方法

onChange(callback: (event: CheckboxGroupResult) => void): CheckboxGroupAttribute;

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。

CheckboxGroupResult类型说明：

```
declare interface CheckboxGroupResult {
      //结果集
     name: Array<string>;
    //选择状态
     status: SelectStatus;
}
```

```
//选择状态
declare enum SelectStatus {
      All,  //全选0
      Part, //部分选择 1
     None //一个也没选2
}
```

Image组件

Image组件-接口定义

Image(src: string | PixelMap | Resource)

参数说明：

- src：图片的数据源。可以加载网络图片和本地图片。PixelMap格式为像素图，常用于图片编辑的场景。

```
"requestPermissions": [
  {
    "name": "ohos.permission.INTERNET"
  }
]
```

Image组件-属性方法

alt(value: string | Resource): ImageAttribute;

加载时显示的占位图，支持本地图片（png、jpg、bmp、svg和gif类型），不支持网络图片。默认值：null

objectFit(value: ImageFit): ImageAttribute;

设置图片的填充效果。默认值：ImageFit.Cover

名称	描述
Contain	保持宽高比进行缩小或者放大，使得图片完全显示在显示边界内。
Cover	保持宽高比进行缩小或者放大，使得图片两边都大于或等于显示边界。
Auto	自适应显示
Fill	不保持宽高比进行放大缩小，使得图片充满显示边界。
ScaleDown	保持宽高比显示，图片缩小或者保持不变。
None	保持原有尺寸显示。

Image组件-属性方法

objectRepeat(value: ImageRepeat): ImageAttribute;

设置图片的重复样式。从中心点向两边重复，剩余空间不足放下一张图片时会截断。

默认值：ImageRepeat.NoRepeat

名称	描述
X	只在水平轴上重复绘制图片。
Y	只在竖直轴上重复绘制图片。
XY	在两个轴上重复绘制图片。
NoRepeat	不重复绘制图片。

Image组件-属性方法

interpolation(value: ImageInterpolation): ImageAttribute;

设置图片的插值效果，即减轻低清晰度图片在放大显示时出现的锯齿问题。

默认值：ImageInterpolation.None

名称	描述
None	不使用图片插值。
High	高图片插值，插值质量最高，可能会影响图片渲染的速度。
Medium	中图片插值。
Low	低图片插值。

Image组件-属性方法

renderMode(value: ImageRenderMode): ImageAttribute;

设置图片的渲染模式为原色或黑白。

默认值：ImageRenderMode.Original

名称	描述
Original	原色渲染模式。
Template	黑白渲染模式。

Image组件-属性方法

```
sourceSize(value: {
        width: number;
        height: number;
    }): ImageAttribute;
```

设置图片解码尺寸，降低图片的分辨率，常用于需要让图片显示尺寸比组件尺寸更小的场景。和ImageFit.None配合使用时可在组件内显示小图。

单位：px

```
autoResize(value: boolean): ImageAttribute;
```

设置图片解码过程中是否对图源自动缩放。设置为true时，组件会根据显示区域的尺寸决定用于绘制的图源尺寸，有利于减少内存占用。如原图大小为1920x1080，而显示区域大小为200x200，则图片会自动解码到200x200的尺寸，大幅度节省图片占用的内存。

默认值：true

Image组件-属性方法

syncLoad(value: boolean): ImageAttribute;

设置是否同步加载图片，默认是异步加载。同步加载时阻塞UI线程，不会显示占位图。

默认值：false

说明：建议加载尺寸较小的本地图片时将syncLoad设为true，因为耗时较短，在主线程上执行即可。

Image组件-属性方法

copyOption(value: CopyOptions): ImageAttribute;

设置图片是否可复制。

当copyOption设置为非CopyOptions.None时，支持使用长按、鼠标右击、快捷组合键'CTRL+C'等方式进行复制。

默认值：CopyOptions.None

名称	描述
None	不支持复制。
InApp	支持应用内复制。
LocalDevice	支持设备内复制。

Image组件-事件方法

onComplete(callback: (event?: { width: number, height: number, componentWidth: number,

componentHeight: number, loadingStatus: number }) => void)

图片数据加载成功和解码成功时均触发该回调，返回成功加载的图片尺寸。

onError(callback: (event?: { componentWidth: number, componentHeight: number , message: string }) => void)

图片加载异常时触发该回调。

onFinish(event: () => void)

当加载的源文件为带动效的svg格式图片时，svg动效播放完成时会触发这个回调。如果动效为无限循环动效，则不会触发这个回调。

仅支持svg格式的图片。

Button组件

Button组件-接口定义

接口定义1

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

接口定义2

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

参数说明：

- type:按钮的类型，是一个枚举类
- stateEffect，按钮按下时是否开启按压显示效果。
- label：按钮的文本内容

ButtonType 枚举类说明如下：

名称	描述
Capsule	胶囊型按钮（圆角默认为高度的一半）。
Circle	圆形按钮。
Normal	普通按钮（默认不带圆角）。

Button组件-属性方法

type(value: ButtonType): ButtonAttribute;

设置按钮的类型

stateEffect(value: boolean): ButtonAttribute;

按钮按下时是否开启按压态显示效果，默认true。

fontColor(value: ResourceColor): ButtonAttribute;

fontSize(value: Length): ButtonAttribute;

fontWeight(value: number | FontWeight | string): ButtonAttribute;

fontStyle(value: FontStyle): ButtonAttribute;

fontFamily(value: string | Resource): ButtonAttribute;

设置按钮字体的一些属性方法

Button组件-事件方法

onClick(event: (event?: ClickEvent) => void): T;

点击事件

onTouch(event: (event?: TouchEvent) => void): T;

触摸事件

```
declare enum TouchType {
    Down, //0
     Up,  //1
    Move, //2
    Cancel
}
```

自定义组件

自定义组件

方式1：组件内

使用 @Builder 装饰器

```
@Builder Header(){
  Row(){
    Image($r("app.media.icon_arrow_left"))
      .width(30)
      .height(30)
    Text("标题")
      .width("90%")

      .textAlign(TextAlign.Center)
  }.width("100%")
}
```

```
Column(){
  this.Header()
}
```

自定义组件

方式2：page 内

使用@Component 装饰器

```
@Component
struct  Header{
  build(){
    Row(){
      Image($r("app.media.icon_arrow_left"))
        .width(30)
        .height(30)
      Text("标题")
        .width("90%")

        .textAlign(TextAlign.Center)
    }.width("100%")
  }
}
```

```
Column(){
  Header()
}
```

自定义组件

方式3：page外

使用@Component 装饰器自定义组件，使用 export/import进行导出导入

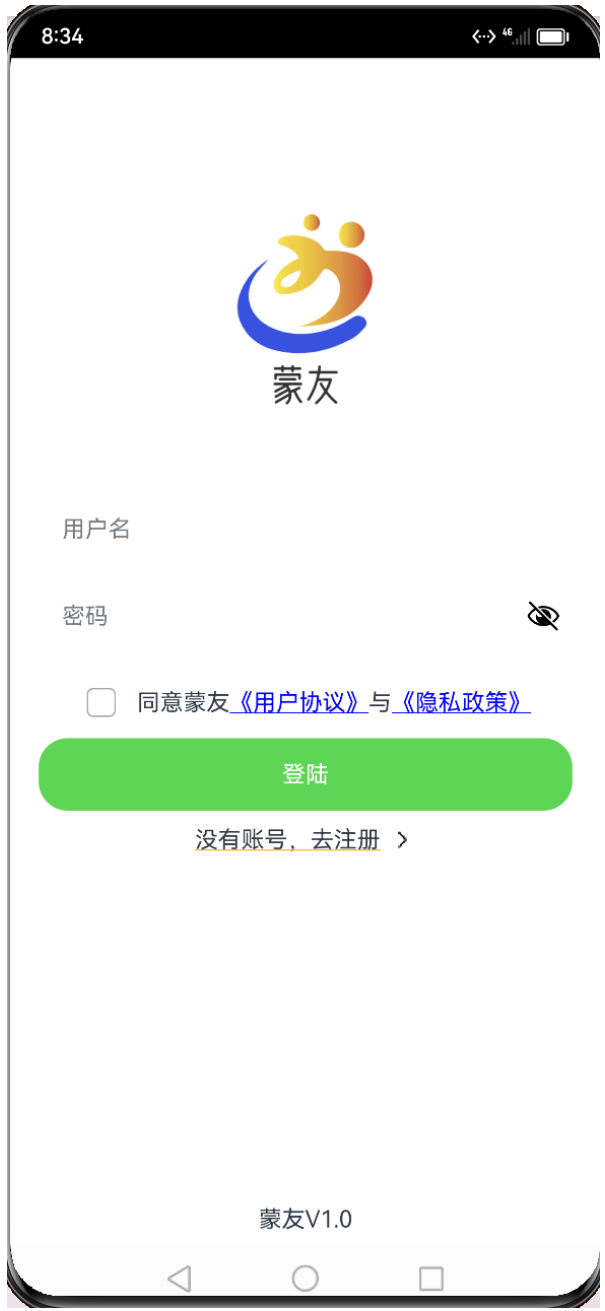
```
@Component
export struct  Header{
  build(){
    Row(){
      Image($r("app.media.icon_arrow_left"))
        .width(30)
        .height(30)
      Text("标题")
        .width("90%")

        .textAlign(TextAlign.Center)
    }.width("100%")
  }
}
```

```
import {Header} from "./components/Header"


Column(){
  Header()
}
```

登录页面UI实现


8:34
蒙友
用户名
密码
同意蒙友《用户协议》与《隐私政策》
登陆
没有账号，去注册
蒙友V1.0

属性抽离

@Styles

设置在组件外

```
@Styles function mysize(){
  .width(300)
  .height(300)
}
```

设置在组件内

```
@Styles mysize(){
  .width(300)
  .height(300)
}
```

调用

```
Image($r("app.media.icon"))
  .mysize()
```

不支持传参数

```
@Styles function mysize(size:number){
  .width(size)
  .height(size)
}
```

@Extend

不传递参数

```
@Extend(Text) function myfont(){
  .fontSize(30)
  .fontColor(Color.Brown)
  .fontWeight(FontWeight.Bold)
  .decoration({
    type:TextDecorationType.Underline
  })
}
```

传递参数，也可以给参数设置 默认值

```
@Extend(Text) function myfont(size:number=30){
  .fontSize(size)
  .fontColor(Color.Brown)
  .fontWeight(FontWeight.Bold)
  .decoration({
    type:TextDecorationType.Underline
  })
}
```

支持彼此调用

```
@Extend(Text) function myColor(){
  .fontColor(Color.Red)
}

@Extend(Text) function myfont(size:number=30){
  .fontSize(size)
  .fontWeight(FontWeight.Bold)
  .myColor()
  .decoration({
    type:TextDecorationType.Underline
  })
}
```

不能定义在组件内部！！！

路由

入口路由

项目运行的入口路由

```
onWindowStageCreate(windowStage: window.WindowStage) {
  // Main window is created, set main page for this ability
  hilog.info(0x0000, 'testTag', '%{public}s', 'Ability onWindowStageCreate');

  windowStage.loadContent('pages/Index', (err, data) => {
    if (err.code) {
      hilog.error(0x0000, 'testTag', 'Failed to load the content. Cause: %{publi
      return;
    }
    hilog.info(0x0000, 'testTag', 'Succeeded in loading the content. Data: %{pub
  });
}
```

路由的基本使用

1、导包

```
import router from '@ohos.router'
```

2、路由的配置

src/main/resources/base/profile/main_pages.json

3、路由的几个函数

```
function pushUrl(options: RouterOptions): Promise<void>;
function replaceUrl(options: RouterOptions): Promise<void>;
function clear(): void;
function getLength(): string;
function getState(): RouterState;
```

```
interface RouterOptions {
        url: string;
        params?: Object;
}
```

```
interface RouterState {
    index: number;
    name: string;
    path: string;
}
```

Web组件

Web组件-接口定义

Web(options: { src: ResourceStr, controller: WebviewController | WebController})

参数：

- option，参数类型为object。有2个参数，src表示加载的url，controller是 Web 的控制器

从API Version 9开始，WebController不再维护，建议使用WebviewController替代。

WebviewController 的方法

forward(): void;

backward(): void;

clearHistory(): void; //清除历史

onActive(): void; //激活

onInactive(): void; //未激活

refresh(): void;

```
import webview from '@ohos.web.webview';
```

loadUrl(url: string | Resource, headers?: Array<WebHeader>): void;//加载页面

zoom(factor: number): void;

zoomIn(): void;

zoomOut(): void;

getTitle(): string;

getPageHeight(): number;

getUrl(): string;

Web组件-属性方法

javaScriptAccess(javaScriptAccess: boolean): WebAttribute;

是否允许js

fileAccess(fileAccess: boolean): WebAttribute;

设置是否启用Web中的本地文件系统访问。

onlineImageAccess(onlineImageAccess: boolean): WebAttribute;

设置是否允许从网络加载图片

darkMode(mode: WebDarkMode): WebAttribute;

设置深色模式

forceDarkAccess(access: boolean): WebAttribute;

强制设置为黑暗模式

WebDarkMode枚举说明：

```
declare enum WebDarkMode {
    Off,
    On,
    Auto
}
```

Web组件-事件方法

```
onPageBegin(callback: (event?: {
        url: string;
    }) => void): WebAttribute;
```

页面开始加载

```
 onPageEnd(callback: (event?: {
        url: string;
    }) => void): WebAttribute;
```

页面加载完成

```
 onProgressChange(callback: (event?: {
        newProgress: number;
    }) => void): WebAttribute;
```

页面加载进度发生变化

动画

转场动画的实现

实现如下代码

```
pageTransition(){
  PageTransitionEnter({
      type?: RouteType,   //
      duration?: number,
      curve?: Curve | string,
      delay?: number
      })
    PageTransitionExit({
      type?: RouteType,
      duration?: number,
      curve?: Curve | string,
      delay?: number
      })
}
```

属性动画

定义：组件的某些通用属性变化时，通过属性动画实现渐变过渡效果。属性动画支持的属性包括：width，height，backgroundColor，opacity，scale，rotate，translate等

动画接口：
animation(value: {duration?: number, tempo?: number, curve?: string | Curve | ICurve,delay?:number, iterations: number, playMode?: PlayMode,onFinish?: () => void})

名称	参数类型	必填	描述
duration	number	否	设置动画时长。单位为毫秒，默认动画时长为1000毫秒。默认值：1000
tempo	number	否	动画播放速度。数值越大，动画播放速度越快，数值越小，播放速度越慢值为0时，表示不存在动画。默认值：1
curve	string | Curve | ICurve	否	设置动画曲线。默认曲线为线性。默认值：Curve.Linear
delay	number	否	设置动画延迟执行的时长。单位为毫秒，默认不延时播放。默认值：0
iterations	number	否	设置播放次数。默认播放一次，设置为-1时表示无限次播放。默认值：1
playMode	PlayMode	否	设置动画播放模式，默认播放完成后重头开始播放。默认值：PlayMode.Normal
onFinish	() => void	否	状态回调，动画播放完成时触发。

网络请求及登录实现

网络请求-使用步骤

第一步：导包
import http from '@ohos.net.http'

第二步：创建一个HttpRequest对象

第三步：使用HttpRequest对象发起网络请求

第四步：使用destroy()销毁请求

接口名	功能描述
createHttp()	创建一个http请求。
request()	根据URL地址，发起HTTP网络请求。
destroy()	中断请求任务。
on(type: 'headersReceive')	订阅HTTP Response Header 事件。
off(type: 'headersReceive')	取消订阅HTTP Response Header 事件。
once('headersReceive')$^{8+}$	订阅HTTP Response Header 事件，但是只触发一次。

第三方库 axios网络请求

axios-使用步骤

第一步：安装 ohpm 并进行初始化

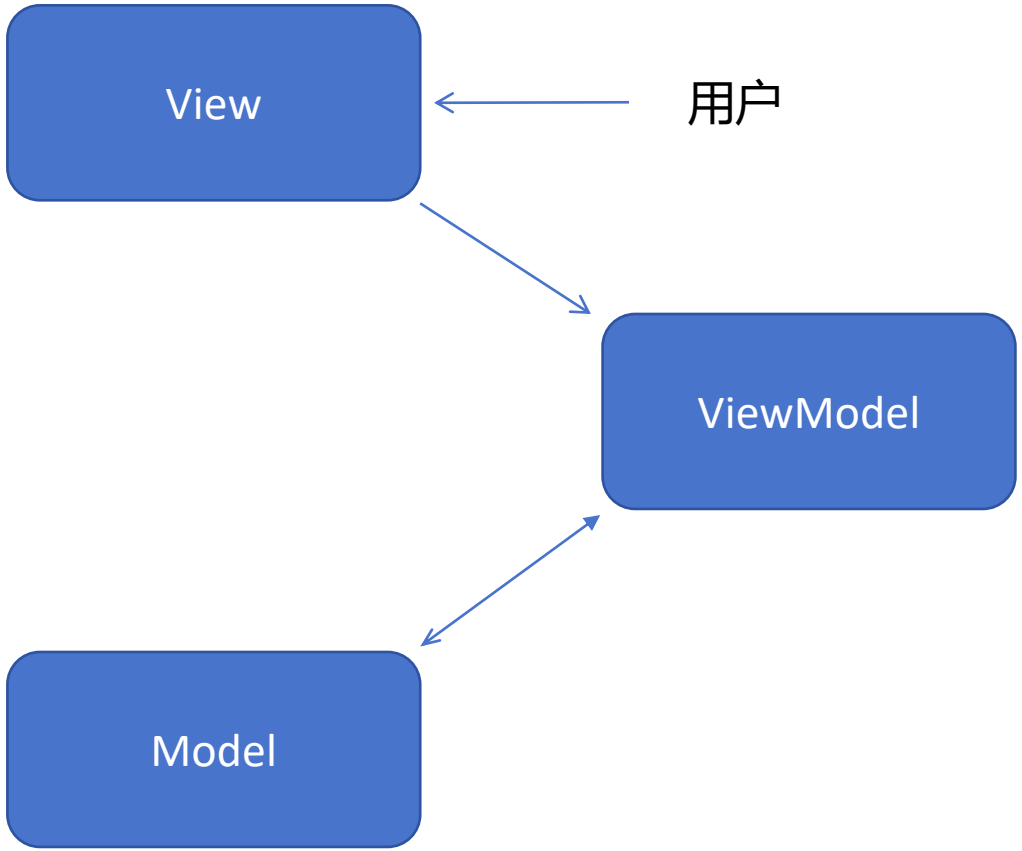
第二步：配置ohpm

第三步：参考第三方库 axios 进行安装
https://ohpm.openharmony.cn/#/cn/home

第四步：使用axios进行网络请求

MVVM及JSON解析

什么是MVVM？



JSON解析和封装

parse(text: string, reviver?: (this: any, key: string, value: any) => any): any;
解析JSON字符串

stringify(value: any, replacer?: (this: any, key: string, value: any) => any, space?: string | number): string;
封装成JSON字符串

应用内数据存储

应用内数据存储

AppStorage 类

static Set<T>(propName: string, newValue: T): boolean;
保存数据

static SetOrCreate<T>(propName: string, newValue: T): void;
保存数据，如果key不存在，创建key

static Get<T>(propName: string): T | undefined;
取出数据