

# 1 实验一 UI与交互

---

## 1.1 实验目的与要求

---

- 熟悉鸿蒙应用基础的代码框架并学习使用调试工具。
- 掌握ArkTS中声明式UI的编写方法与渲染逻辑。
- 掌握常用组件的使用方法，学习如何添加样式。

## 1.2 实验内容

---

1. 完成单页page中声明式UI的编写。
2. 使用样式API修改文本颜色与按钮间距，并使用响应式变量使得文本颜色可以在完成目标时变化。
3. 添加“New Game”按钮并完成组件事件的处理逻辑，同时使用条件渲染切换按钮的显示状态。

## 1.3 实验指南

---

1. 完成单页page中声明式UI的编写。

从这里你可以了解ArkTS的基础概念：[基本语法概述](#)。

从这里你可以了解声明式UI的基础概念：[声明式UI描述](#)。

只有 push 和 pop 你是无法获得 [2, 2] 来赢得游戏的，你需要添加 reverse 按钮并通过list的 reverse方法来操作你的List。

阅读已有的代码会很有帮助，参考给出的代码与 Button 组件的文档。

2. 使用样式API修改文本颜色与按钮间距，并使用响应式变量使得文本颜色可以在完成目标时变化。

为 win?/win! 文本添加颜色，并使得其可以在赢得游戏时变为金色。

为几个按钮设置合适的间距。

阅读已有的代码会很有帮助，参考声明式UI的编写方法及相关文档。

3. 添加 New Game 按钮并完成组件事件的处理逻辑，同时使用条件渲染切换按钮的显示状态。

使用条件渲染，当赢得游戏时隐藏 push, pop, reverse 按钮，转而显示 New Game 按钮。

处理 New Game 按钮的点击事件，生成一个新的可玩儿的Goal，你可以通过随机进行 push, pop, reverse 三个操作来生成合法可玩儿的Goal。

参考条件渲染相关内容：[if/else: 条件渲染-渲染控制](#)，参考事件处理相关文档。

## 2 实验二 应用Ability模型


### 2.1 实验目的与要求

- 学习如何响应来自组件的事件。
- 学习ArkTS路由与页面的运作方法。
- 学习UIAbility模型的运行与数据交互。

### 2.2 实验内容

1. 显示并通过存储API保存最高分。
2. 为应用添加其他两个页面：欢迎页面与关于页面，在关于页面添加作者信息。
3. 使用Dialog组件实现退出时进行弹窗提醒。

### 2.3 实验指南

 无法完成实验二任务，你需要使用模拟器进行验证。

1. 显示并通过存储API保存最高分。

每次点击 `New Game` 即获得一分。你需要新建一个变量存储最高分，要求在程序退出并重新启动后正常保存最高分（重新安装后会消失）。

参考

- [PersistentStorage: 持久化存储UI状态-管理应用拥有的状态-状态管理](#)
- [布局概述-开发布局-基于ArkTS的声明式开发范式](#)

2. 为应用添加其他两个页面：欢迎页面与关于页面，在关于页面添加作者信息。

你需要添加页面 `welcome` 与 `About`。

你需要右键点击“**pages**”文件夹，选择“**New > Page**”，此时无需手动配置相关页面路由。否则请参考手动配置路由的方法。

`welcome`：从 `welcome` 页面可以进入到主游戏界面 `Index` 与 `About`。你需要使用按钮以及路由API完成这个功能。

`About`：在该页面添加你的作者信息，使用图像 `Image` 标签引入你喜欢的头像。

由于默认的启动页面是 `Index`，你需要修改入口 `Ability` 的代码使其启动时进入 `welcome` 页面。

参考：

- [页面路由 \(router\) -设置页面路由和组件导航](#)
- [显示图片 \(Image\) -显示图形](#)

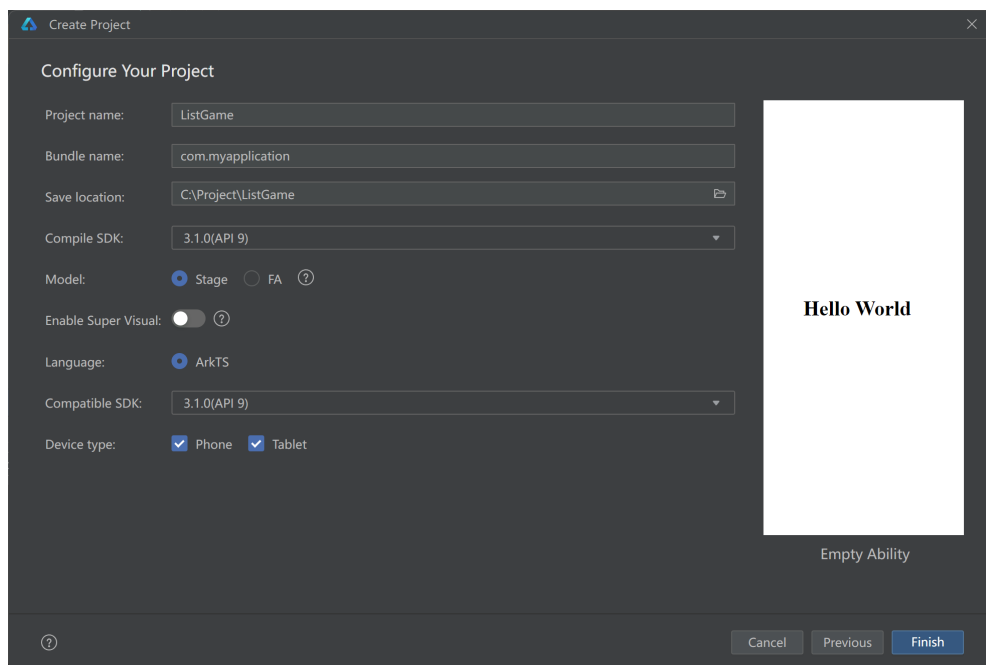
3. 使用 `Dialog` 组件实现退出时进行弹窗提醒。

参考：

- `Dialog` 组件的API
- [UIAbility组件与UI的数据同步-UIAbility组件](#)
- [UIAbilityContext-application-terminateSelf](#)

## 3 帮助

### 实验开始



要开始实验，你需要新建空白项目，模板选择EmptyAbility，注意保持 `Enable Super Visual` 选项关闭。接着将给出的代码粘贴到你的 `Index.ets` 文件中。

实验任务主要涉及index页面的编写，位于

`YourProjectFolder\entry\src\main\ets\pages\Index.ets`，这也是空白应用默认的启动页面。

### 预览器

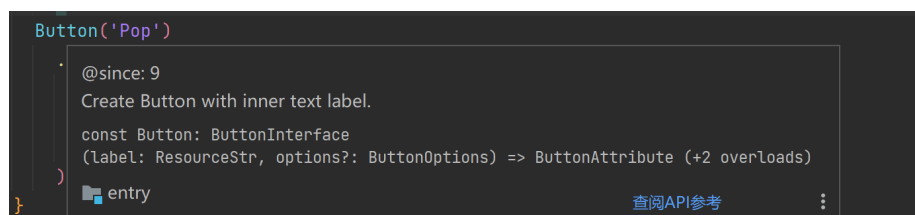
你可以使用 `预览器` 完成该部分实验内容，而不需要打开模拟器。预览器可以实时显示 `页面` 的编译结果，当你保存文件的修改后预览器会自动刷新，你可以把预览器当作一个轻量级的模拟器使用。

### ListGame

ListGame是通过数组的三个操作（pop, push, reverse）来达成目标数组内容的小游戏。在预览器中点击按钮应当会使得你的List发生变化，当你的List与Goal一致时获得游戏的胜利。你可以扩展数组操作的方法，或者将其包装为更有趣的样子。

### 文档

你可以在官方网站上搜索文档与API参考获得帮助，也可以在 `DevEco Studio` 中通过鼠标悬停在函数上并点击 `查看API参考` 来获得帮助。



官方文档包含两种：

- 指南：[快速入门-指南](#)
- API参考：[ArkTS API参考](#)

在你不知道要怎么实现一个功能时查阅指南，里面包含许多例子供你参考。在你遇到函数不知道如何使用时查阅API参考。在右上角搜索任何你需要的内容。



## 初始代码

Index.ets 页面给出初始代码如下（也可以在实验材料中里找到）。

```
@Entry
@Component
struct Index {
  @State gameList: number[] = [1, 2, 3]
  @State gameGoal: number[] = [2, 2]

  iswin() {
    return this.gameGoal.toString() == this.gameList.toString()
  }

  newGame() {}

  quitGame() {}

  build() {
    Row() {
      Column() {
        Text(this.iswin() ? 'Win!' : 'Win?')
          .fontSize(50)
          .fontWeight(FontWeight.Bold)

        Text('Goal: ' + this.gameGoal.toString())
          .fontSize(50)
          .fontWeight(FontWeight.Bold)
          .fontColor(Color.Orange)

        Text(this.gameList.toString())
          .fontSize(50)
          .fontWeight(FontWeight.Bold)
      }
    }
  }
}
```

```
Row() {  
  Button('Push')  
    .onClick(  
      () => {  
        this.gameList.push(this.gameList.length + 1)  
      }  
    )  
  Button('Pop')  
    .onClick(  
      () => {  
        this.gameList.pop()  
      }  
    )  
}  
}  
.width('100%')  
}  
.height('100%')  
}  
}
```

# 4 参考效果

效果仅供参考，使用样式API和更多的界面元素实现更漂亮、个性化的内容。

更多组件参考：[添加常用组件-添加组件-基于ArkTS的声明式开发范式-UI开发-开发](#)

