模块化语法

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

目录

Contents

模块化基本认知

模块化：把一个大的程序，【拆分】成若干的小的模块，通过【特定的语法】，可以进行任意组合

ArkTS 中每个 ets 文件，都可以看做是一个模块

目录

Contents

默认导出和导入

默认导出：指一个模块，只能默认导出的 一个值 或 对象。使用时，可以 自定义 导入名称。

使用步骤：

1. 当前模块中 导出模块
2. 需要使用的地方 导入模块

```
// 默认导出
export default 需要导出的内容
```

```
// 默认导入
import xxx from '模块路径'
```

目录

Contents

按需导出和导入

按需导出：指一个模块，可以按照需要，导出多个特性。

```
// 逐个导出单个特性
export let name1 = …, name2 = …, …, nameN;
export function FunctionName(){...}
export class ClassName {...}

// 一次性导出
export { name1, name2, …, nameN };


// ------------------------------

// 导入
import { name1, name2, name3 as 别名 } from "module-name";
```

目录

Contents

全部导入

将所有的按需导入，全部导入进来 → 导出部分不需要调整，调整导入的语法即可

```
import * as Utils from './utils'
// 通过 Utils 即可获取 utils模块中导出的所有内容
```

自定义组件 - 基础

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

自定义组件 - 基本使用

概念：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。

```
@Entry
@Component
struct Index {
  build() {
    Column() {
      Text('系统组件')
      // 自定义组件
      HelloComponent()
      HelloComponent()
    }
  }
}
```

```
// 定义
@Component
struct HelloComponent {
  // 状态变量
  @State message:string =''
  build(){
    // .... 描述 UI
  }
}
```

自定义组件 - 通用样式事件

自定义组件可以通过点语法，设置 通用样式, 通用事件

如果想要单独预览组件，可以使用 @Preview 进行装饰

```
@Entry
@Component
struct Index {
  build() {
    Column() {
      HelloComponent()
        .width(200)
        .height(100)
        .backgroundColor(Color.Orange)
        .onClick(() => {
          console.log('外部添加的点击事件')
        })
    }
  }
}
```

```
@Preview
@Component
export struct HelloComponent {
  @State info: string = '默认info'
  build() {
    Row() {
      Text(this.info)
      Button('修改数据')
    }
  }
}
```

自定义组件 – 成员函数变量

除了必须要实现 build() 函数外，还可以定义其他的成员函数，以及成员变量。

成员变量的值 → 外部可传参覆盖


```arkts
@Component
struct HelloComponent {
  // 状态变量
  @State msg:string = ''
  // 成员变量-数据
  info:string = ''
  // 成员变量-函数
  sayHello = () => {}

  // 成员函数
  sayHi(){}
  build(){
    // .... 描述 UI
  }
}
```

```arkts
@Entry
@Component
struct CustomComponentDemo {
  build() {
    Column() {
      // 使用组件内部定义的初始值
      HelloComponent()
      // 使用传入的值，覆盖子组件的默认值
      HelloComponent({ info: '你好', msg: 'ArkTS' })
      // 函数也可以传入
      HelloComponent({ sayHello(){ console.log('传入的逻辑') } })
    }
  }
}
```

@BuilderParam 传递UI

@BuilderParam 传递UI

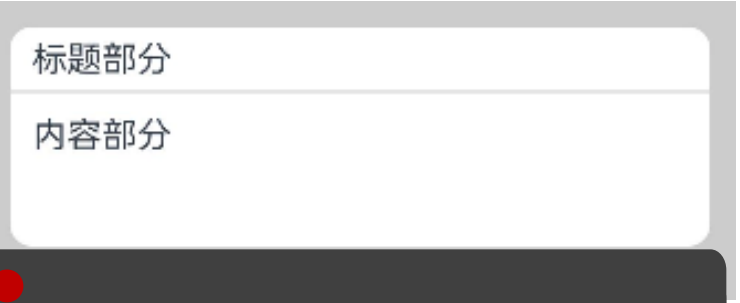
利用 @BuilderParam 构建函数，可以让自定义组件 允许外部传递 UI。


```
@Entry
@Component
struct Index {
  build() {
    Column({ space: 15 }) {
      SonCom() {
        // 直接传递进来(尾随闭包)
        Button('待付款')
      }
    }
  }
}
```

```
@Component
struct SonCom {
  // 1.定义 BuilderParam 接受外部传入的 ui，并设置默认值
  @BuilderParam ContentBuilder: () => void = this.defaultBuilder
  // 默认 的 Builder
  @Builder
  defaultBuilder() {
    Text('默认的内容')
  }
  build() {
    Column() {
      // 2. 使用 @BuilderParam 装饰的成员变量
      this.ContentBuilder()
    }
  }
}
```

多个 @BuilderParam 参数

子组件有多个BuilderParam，必须通过参数的方式来传入




```
@Component
struct SonCom {
  // 由外部传入 UI
  @BuilderParam tBuilder: () => void = this.tDefaultBuilder
  @BuilderParam cBuilder: () => void = this.cDefaultBuilder
  // 设置默认 Builder
  @Builder
  tDefaultBuilder() { ... }
  @Builder
  cDefaultBuilder() { ... }

  build() {
    Column() {
      this.tBuilder()
      this.cBuilder()
    }
  }
}
```

```
@Entry
@Component
struct Index {
  @Builder
  fTBuilder() { ... }
  @Builder
  fCBuilder() { ... }

  build() {
    Column({ space: 15 }) {
      SonCom({
        tBuilder: this.fTBuilder,
        cBuilder: this.fCBuilder
      })
    }
  }
}
```

状态管理

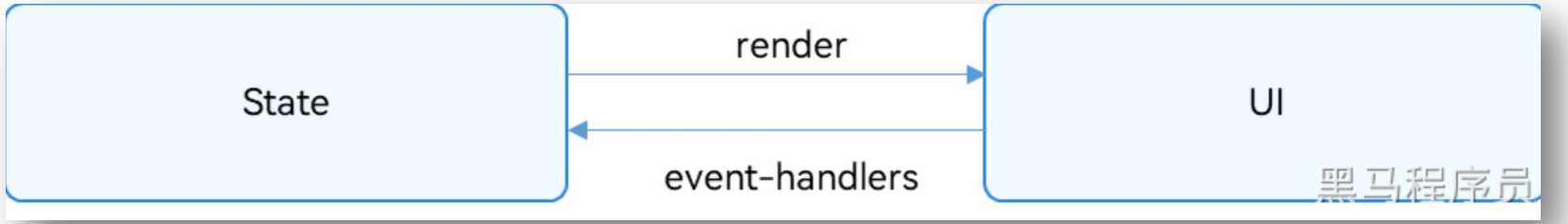
黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

状态管理概述

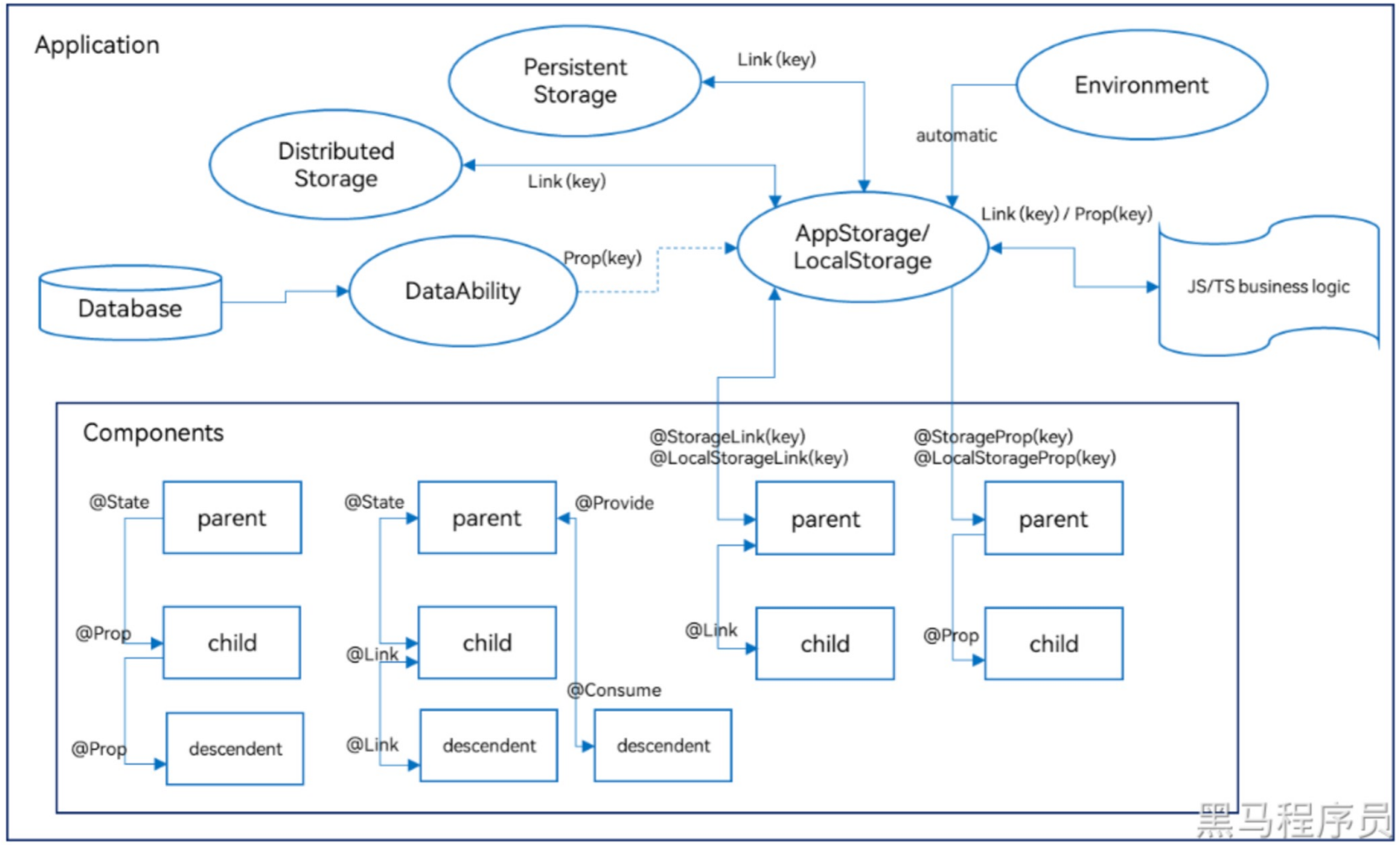
当运行时的 状态变量 变化，带来UI的重新渲染，在ArkUI中统称为 状态管理机制。

变量必须被 装饰器 装饰才可以成为状态变量。



状态管理概述

一张完整的装饰器说明图：



@State 自己的状态

注意：不是状态变量的所有更改都会引起刷新。只有可以被框架观察到的修改才会引起UI刷新。

1. boolean、string、number类型时，可以观察到数值的变化
2. class或者Object时，可观察 自身的赋值 的变化， 第一层属性赋值的变化，即Object.keys(observedObject) 返回的属性。

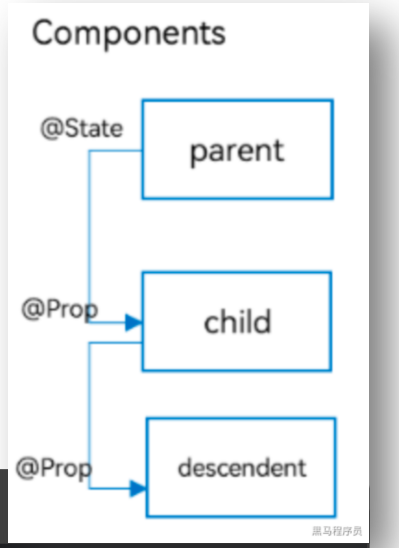
```
// 状态变量
@State message: string = 'Hello, World!';
@State person: Person = {
  name: 'jack',
  dog: {
    name: '柯基'
  }
}
```

```
Button('修改title外层属性')
  .onClick(() => {
    this.person.name = '666'
  })
Button('修改title嵌套属性')
  .onClick(() => {
    // 无法触发更新
    // this.person.dog.name = '内部的 666'
    this.person.dog = {
      name: '阿拉斯加'
    }
  })
```

@Prop - 父子单向

@Prop 装饰的变量可以和父组件建立单向的同步关系。

@Prop 装饰的变量是可变的，但是变化不会同步回其父组件




```
@Entry
@Component
struct FatherCom {
  @State info: string = '么么哒'

  build() {
    Column() {
      Text(this.info)
      SonCom({
        info: this.info,
        changeInfo: (newInfo: string) => {
          this.info = newInfo
        }
      })
    }
  }
}
```

```
@Component
struct SonCom {
  @Prop info: string
  changeInfo = (newInfo: string) => {
  }

  build() {
    Button('info:' + this.info)
      .onClick(() => {
        this.changeInfo('改啦')
      })
  }
}
```

传智教育旗下高端IT教育品牌