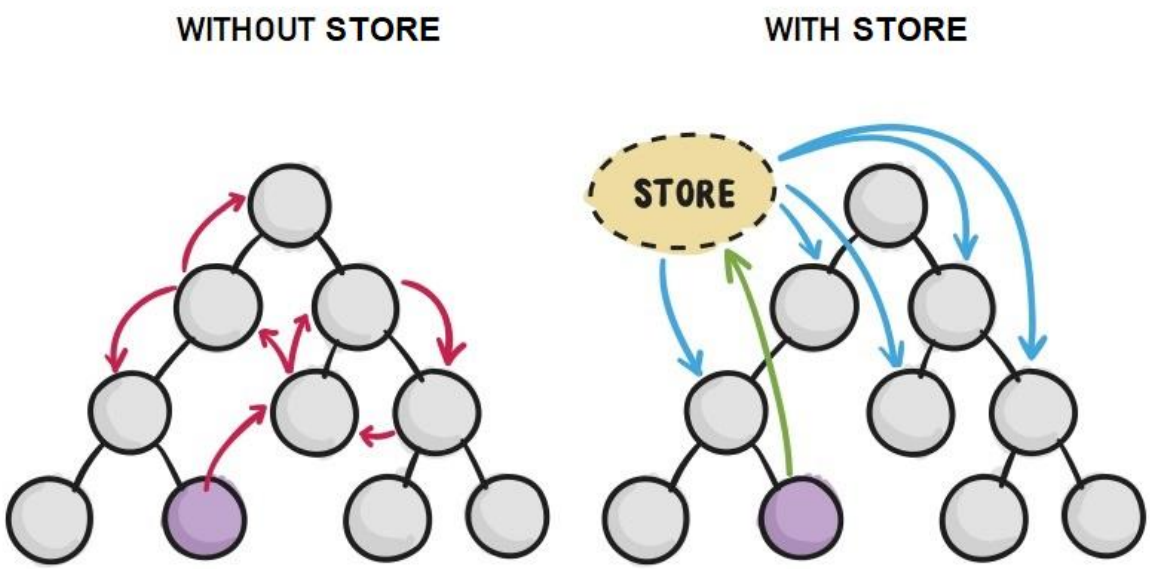
# 全局数据共享



## 1、什么是全局数据共享

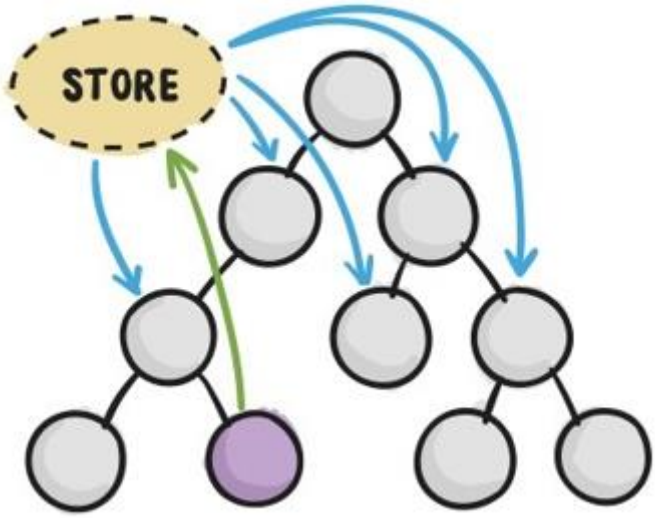
全局数据共享（又叫做：状态管理）是为了解决组件之间数据共享的问题。
开发中常用的全局数据共享方案有：Vuex、Redux、MobX 等。




## 2、小程序中的全局数据共享方案

在小程序中，可使用 mobx-miniprogram 配合 mobx-miniprogram-bindings 实现全局数据共享。其中：

1. mobx-miniprogram 用来创建 Store 实例对象
2. mobx-miniprogram-bindings 用来把 Store 中的共享数据或方法，绑定到组件或页面中使用




# 3、 MobX

## 3.1、安装 MobX 相关的包

在项目中运行如下的命令，安装 MobX 相关的包：

```
npm install --save mobx-miniprogram@4.13.2 mobx-miniprogram-bindings@1.2.1
```

注意：MobX 相关的包安装完毕之后，记得删除 miniprogram_npm 目录后，重新构建 npm。

## 3.2、创建 MobX 的 Store 实例

```js
import { observable, action } from 'mobx-miniprogram'

export const store = observable({
  // 数据字段
  numA: 1,
  numB: 2,
  // 计算属性
  get sum() {
    return this.numA + this.numB
  },
  // actions 方法，用来修改 store 中的数据
  updateNum1: action(function (step) {
    this.numA += step
  }),
  updateNum2: action(function (step) {
    this.numB += step
  }),
})
```

## 3.3、将 Store 中的成员绑定到页面中

```js
// 页面的 .js 文件
import { createStoreBindings } from 'mobx-miniprogram-bindings'
import { store } from '../../store/store'

Page({
  onLoad: function () { // 生命周期函数--监听页面加载
    this.storeBindings = createStoreBindings(this, {
      store,
      fields: ['numA', 'numB', 'sum'],
      actions: ['updateNum1']
    })
  },
  onUnload: function () { // 生命周期函数--监听页面卸载
    this.storeBindings.destroyStoreBindings()
  }
})
```

## 3.4、在页面上使用 Store 中的成员

```
// 页面的 .wxml 结构
<view>{{numA}} + {{numB}} = {{sum}}</view>
<van-button type="primary" bindtap="btnHandler1" data-step="{{1}}">
  numA + 1
</van-button>
<van-button type="danger" bindtap="btnHandler1" data-step="{{-1}}">
  numA - 1
</van-button>

// 按钮 tap 事件的处理函数
btnHandler1(e) {
  this.updateNum1(e.target.dataset.step)
}
```

## 3.5、将 Store 中的成员绑定到组件中

```
import { storeBindingsBehavior } from 'mobx-miniprogram-bindings'
import { store } from '../../store/store'

Component({
  behaviors: [storeBindingsBehavior], // 通过 storeBindingsBehavior 来实现自动绑定

  storeBindings: {
    store,    // 指定要绑定的 Store
    fields: { // 指定要绑定的字段数据
      numA: () => store.numA,      // 绑定字段的第 1 种方式
      numB: (store) => store.numB, // 绑定字段的第 2 种方式
      sum: 'sum'                   // 绑定字段的第 3 种方式
    },
    actions: { // 指定要绑定的方法
      updateNum2: 'updateNum2'
    }
  },
})
```

### 3.6、在组件中使用 Store 中的成员

```
// 组件的 .wxml 结构
<view>{{numA}} + {{numB}} = {{sum}}</view>
<van-button type="primary" bindtap="btnHandler2" data-step="{{1}}">
  numB + 1
</van-button>
<van-button type="danger" bindtap="btnHandler2" data-step="{{-1}}">
  numB - 1
</van-button>

// 组件的方法列表
methods: {
  btnHandler2(e) {
    this.updateNum2(e.target.dataset.step)
  }
}
```

## 4、总结

① 能够知道如何使用 MobX 实现全局数据共享
安装包、创建 Store、参考官方文档进行使用