Mobx

Mobx课程目录

1. Mobx介绍
2. Mobx - 环境搭建
3. Mobx - 第一个store
4. Mobx - computed
5. Mobx - 模块化
6. Mobx + React实战案例

Mobx介绍

1. 什么是Mobx
2. 有什么优势
3. 社区评价

Mobx介绍 - 是什么

一个可以和React良好配合的集中状态管理工具，

mobx和react的关系，相当于vuex和vue

同类工具还有：

1. redux
2. dva
3. recoil

Mobx介绍 - 优势

1. 简单

    编写无模板的极简代码来精准描述你的意图（原生js）

2. 轻松实现最优渲染

    依赖自动追踪最小渲染优化

3. 架构自由

    可移植，可测试

Mobx介绍 - 社区评价

> 兄 dei 们，#mobx 用的并不是发布订阅，也不是你爷爷的观察者模式。非也，它用的是一个经过精心设计的、由宇宙力量驱动的 observable 维度传送门。它并不会进行变更侦测。它其实是个带着一把灵魂刀的 20 级灵能师，连劈带砍让你的视图模型屈服于其刀下。

> 在个人项目里用了几周 #mobx 之后把它介绍给了团队，感觉很棒。一半的时间，双倍的快乐

> 简单地说，使用 #mobx 就是一个持续的循环，我不停地嘀咕"这也太简单了，肯定行不通"，结果不断证明我错了。

> 我已经用 MobX 写过大型的应用了，比起之前那个用 Redux 写的，用 MobX 写的更容易阅读，理解起来也容易得多。

> #mobx 符合我一直以来想要的事物的样子。真是出人意料的简单快速！非常棒！别错过了！

Mobx - 环境配置

Mobx环境配置 - 配置说明

Mobx是一个独立的响应式的库，可以独立于任何UI框架而存在，但是通常人们把它和React来绑定使使用，用Mobx来做响应式数据建模，React作为UI视图框架渲染内容

所以配置方面我们需要三个部分：

1. 一个通过create-react-app 创建好的react项目环境
2. mobx本身
3. 一个链接mobx和react的中间部件

Mobx环境配置 - 如何配置

1. 使用 create-react-app初始化react项目

```
$ npx create-react-app mobx-react-app
```

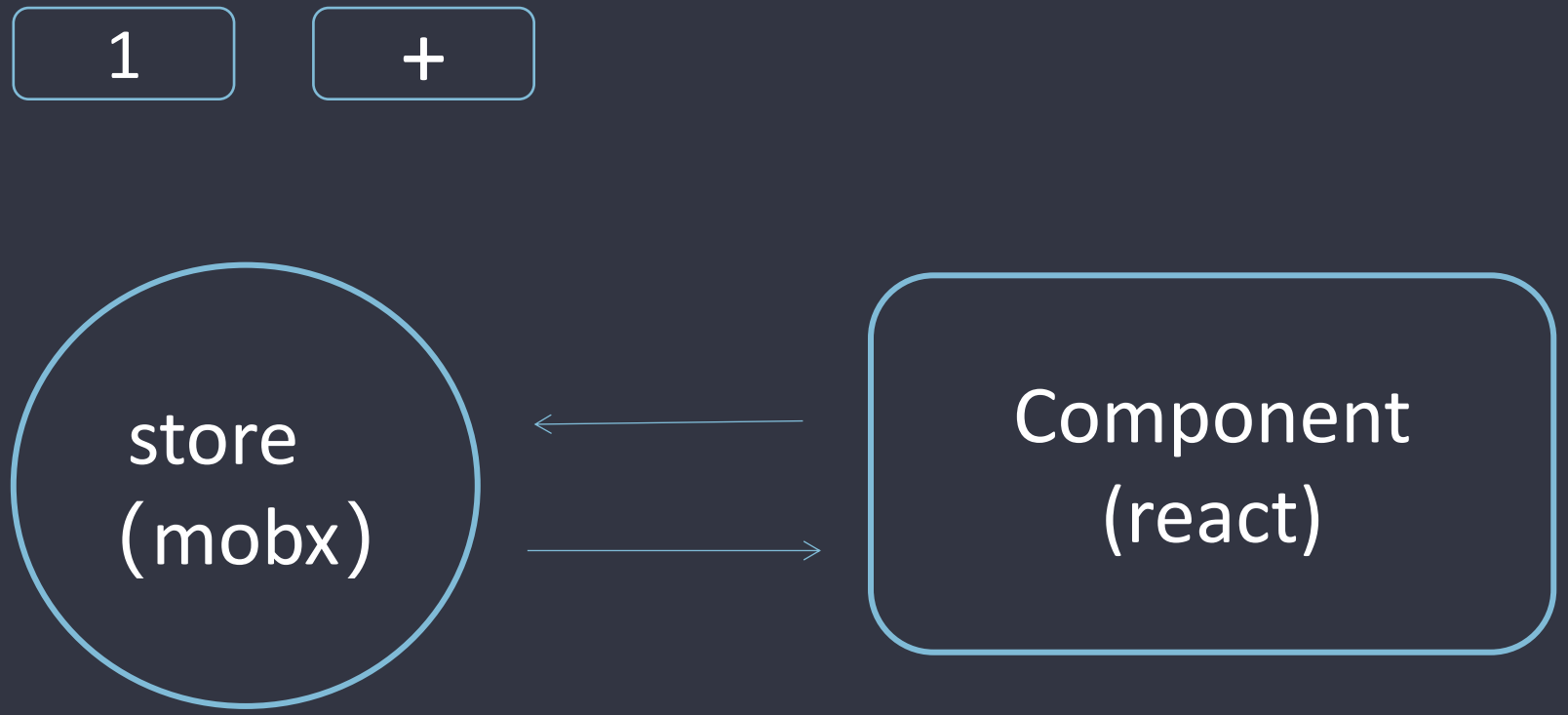
2. 安装mobx和mobx-react-lite

```
$ yarn add mobx mobx-react-lite
```

Mobx - 第一个 store

第一个store - 理解需求

需求：使用Mobx实现计数器案例，mobx负责计数逻辑，react负责渲染和事件触发



第一个store - 初始化mobx

实现步骤

1. 定义数据状态（state）
2. 数据响应式处理
3. 定义action函数 （修改数据）
4. 实例化并导出实例

```
import { makeAutoObservable } from 'mobx'

class CounterStore {
  // 定义数据
  count = 0
  constructor(){
    // 响应式处理
    makeAutoObservable(this)
  }
  addCount = ()=>{
    // 定义action函数
    this.count++
  }
}
// 实例化
const counterStore = new CounterStore()
export default counterStore
```

第一个store - 连接react

实现步骤

1. 导入store实例
2. 使用store中的数据
3. 修改store中的数据
4. 让组件视图响应数据变化

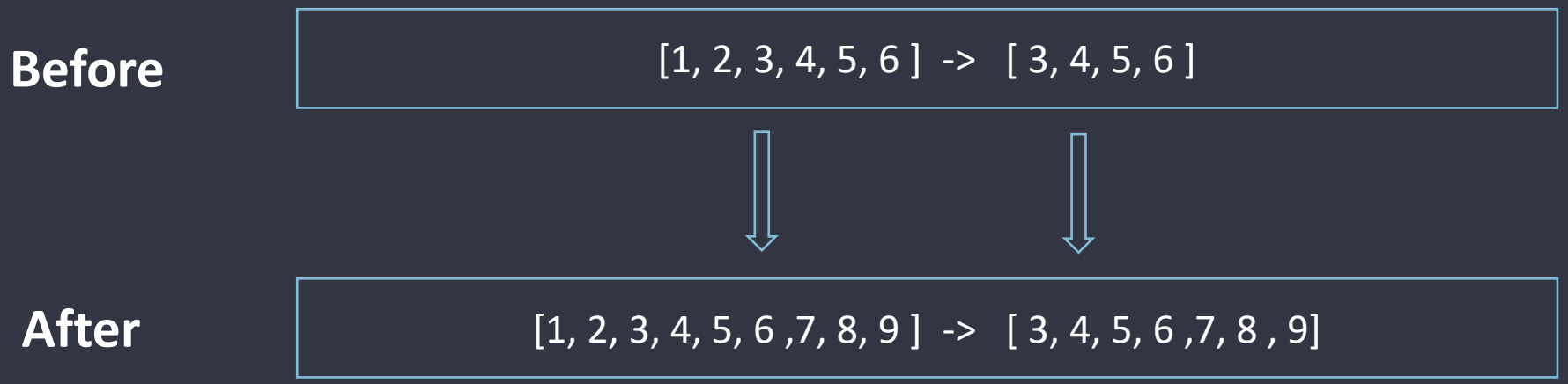
```jsx
// 引入定义好的counterStore
import counterStore from './store'
// 引入更新视图的关键方法
import { observer } from 'mobx-react-lite'

function App(){
  return (
    <div>
      {/*使用数据 并点击修改数据*/}
      <button
        onClick={counterStore.addCount}
      >
          {counterStore.count}
      </button>
    </div>
  )
}
// 使用observer方法包裹组件 使其响应数据变化
export default observer(App)
```

Mobx - computed

computed - 是什么

概念：基于现有的数据做计算得到新的数据，并且可以在依赖的数据发生变化时立刻进行计算



computed - 如何实现

实现步骤

1. 声明一个存在的数据

2. 定义get 计算属性 （ 定义计算公式 ）

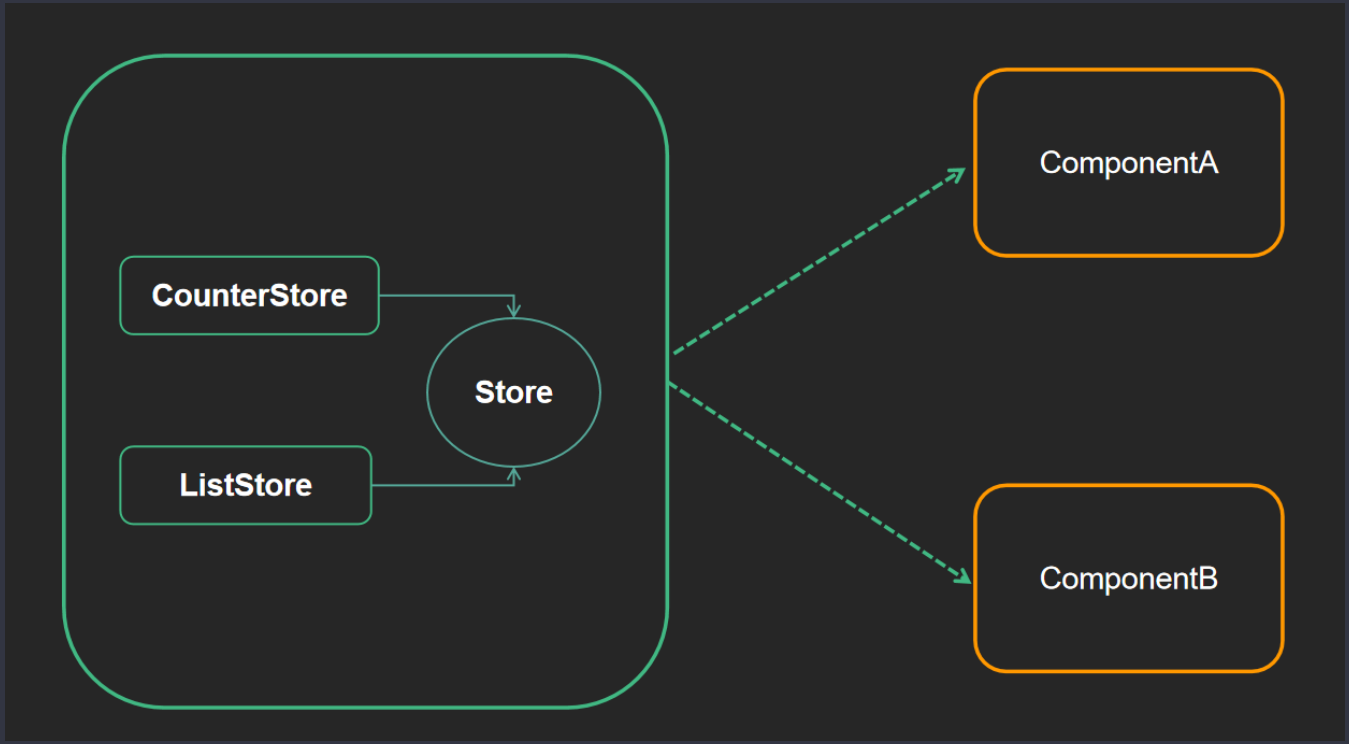
3. 在makeAutoObservable方法中标记

```
// 导入computed
import {computed, makeAutoObservable } from 'mobx'
class CounterStore {
  list = [1,2,3,4,5]
  constructor(){
    makeAutoObservable(this,{
      filterList: computed
    })
  }
  // 计算属性
  get filterList(){
    return this.list.filter(item => item > 2)
  }
  // 修改源数据
  addList = ()=>{
    this.list.push(6,7,8)
  }
}
```

Mobx - 模块化

Mobx模块化 - 是什么

一个项目有很多业务模块，我们不能把所有的代码都写到一起，这样很难维护，为了提供可维护性，需要引入模块化

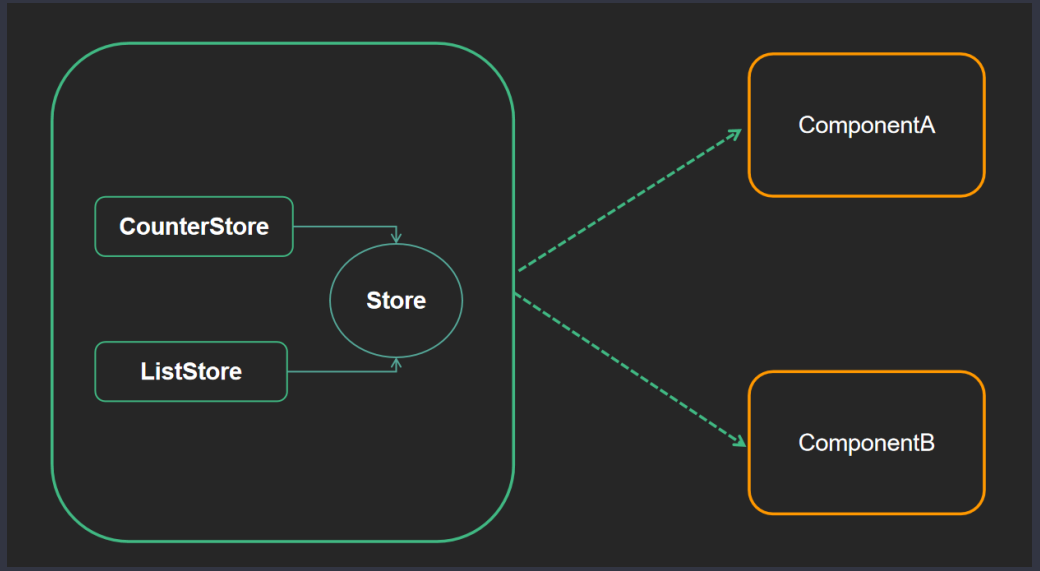


Mobx模块化 - 怎么做

1. 拆分Count和List模块，每个模块定义自己独立的state/actions

2. 在store/index.js中导入拆分之后的模块，进行模块组合

3. 使用React的 useContext机制导出统一的useStore方法，供业务组件使用



Mobx模块化 - 代码实现

```mobx
import { makeAutoObservable } from 'mobx'

class CounterStore {
  // 定义数据
  // 响应式处理
  // 定义action函数
}
// 实例化并导出
const counterStore = new CounterStore()
export default counterStore
```

store/CounterStore

```mobx
import { makeAutoObservable } from 'mobx'

class ListStore {
  // 定义数据
  // 响应式处理
  // 定义action函数
}
// 实例化并导出
const listStore = new ListStore()
export default listStore
```

store/ListStore

```mobx
import  counterStore  from './countStore'
import  listStore from './listStore'

class RootStore {
  // 组合store
  constructor(){
    this.counterStore = counterStore
    this.listStore = listStore
  }
}
// 实例化根store注入context
const rootStore = new RootStore()
const context = React.createContext(rootStore)

// 导出useStore方法 供组件通过调用该方法使用根实例
const useStore = () => React.useContext(context)

export { useStore }
```

store/index.js

Mobx基础使用 - 总结

1. 初始化mobx的过程是怎样的 ？

   声明数据 -> 响应式处理 -> 定义action函数 -> 实例化导出

2. mobx如何配合react，需要依赖什么包 ？

   mobx-react-lite作为链接包，导出observer方法，包裹组件（只能和函数组件配合）

3. 模块化解决了什么问题 ？

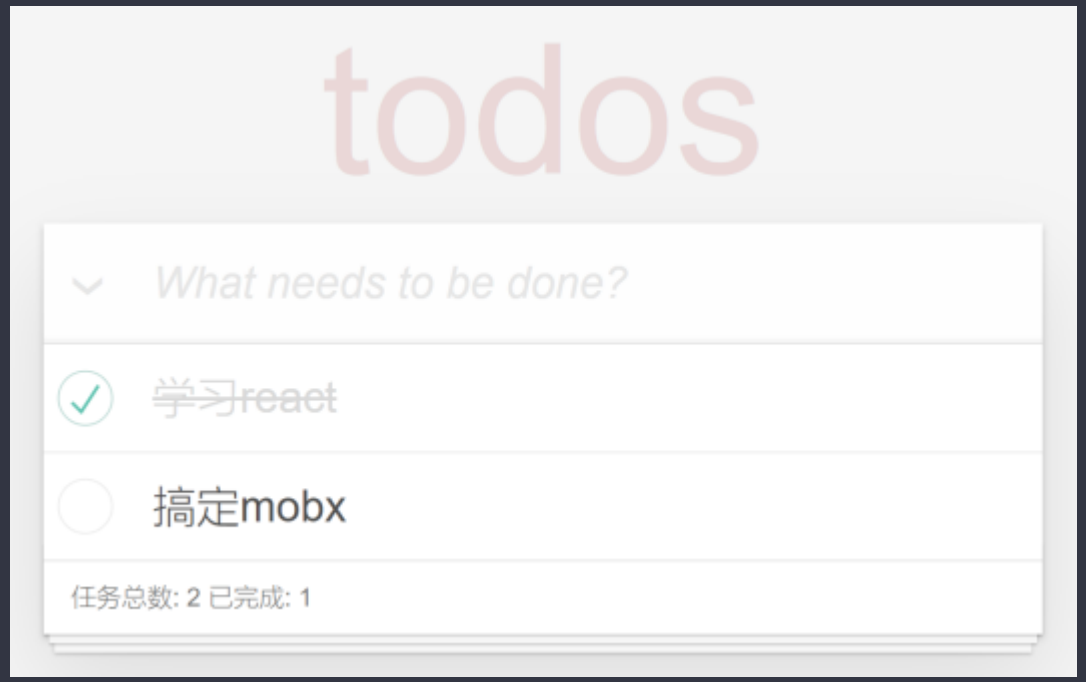
   维护性问题

4. 如何实现mobx的模块化 ？

   按照功能拆分store模块，根模块中组合子模块，利用context机制依赖注入

Mobx实战案例-Todos

Mobx实战案例 - Todos效果演示



1. 渲染列表数据
2. 单选功能
3. 全选功能
4. 删除功能
5. 回车新增功能
6. 统计计数功能

http://react-course-series.gitee.io/mobx_react/ (打开链接预览所有功能)

Mobx实战案例 - Todos需求实现

实现步骤

1. 打开项目地址，克隆项目到本地

```
git clone https://gitee.com/react-course-series/mobx_react.git
```

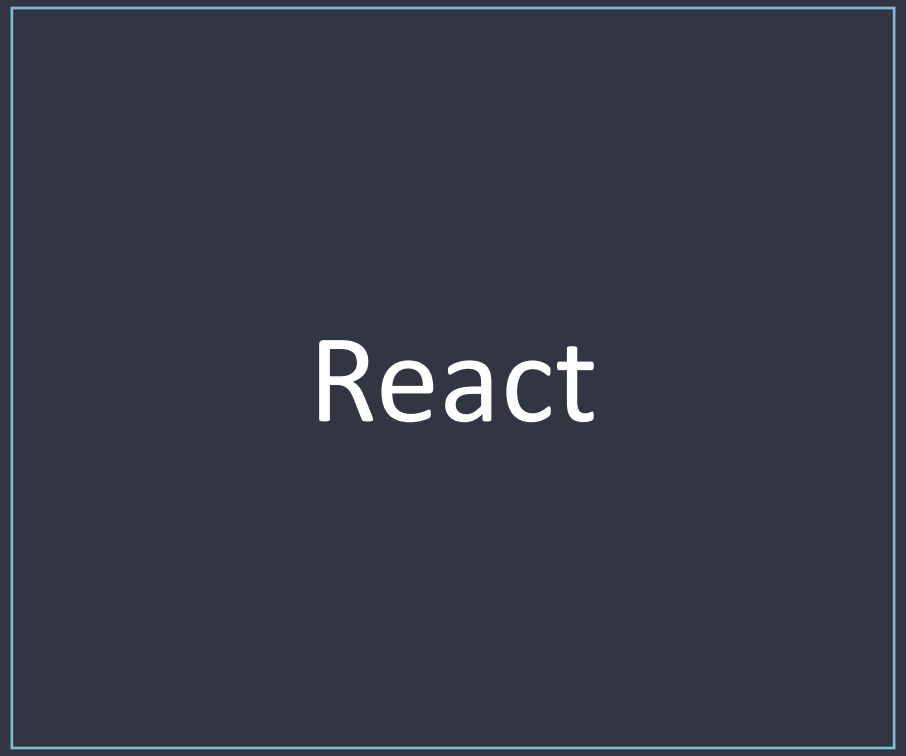
2. 安装所有依赖

```
yarn  install
```

说明： master分支为纯模板，mvc-finished分支为完整版本，如需要可参考

Mobx 和 React 职责划分

1. 业务状态数据
2. 业务状态操作逻辑



1. 渲染业务数据
2. UI临时状态维护
3. 事件触发，调用Mobx