React&Redux

React介绍

1. 简述
    - React（有时叫React.js或ReactJS）是一个为数据提供渲染为HTML视图的开源JavaScript 库
    - React视图通常采用包含以自定义HTML标记规定的其他组件的组件渲染
    - React为程序员提供了一种子组件不能直接影响外层组件（"data flows down"）的模型，数据改变时对HTML文档的有效更新，和现代单页应用中组件之间干净的分离
2. React优势
    - 传统开发的缺点
      数据需要实时反应到UI上，导致每次改动都需要操作dom
      复杂或频繁的DOM操作通常是性能瓶颈产生的原因
    - 目标
      解决其它JavaScript框架所面对的一大常见难题，即对大规模数据集的处理
    - 替换原有Angular的开发模块
      成本高、维护复杂、扩展等不如react清晰等原因
3. React特点
    - SPA单页应用
    - 不关注架构
    - 不是一个完整Facebook的单页应用的MVC框架
    - REACT是一套工可比性具，并不是“框架”，人称框架是因为做到了很多框架想做但没做到事

组件化开发

1. 组件化概念
    - 模块
      在react前端开发中，把封装起来具有特定功能的独立UI部件或功能函数都叫做模块
    - 模块化开发
      在react中模块化开发贯穿其中，我们可以把一些相似功能的部分抽

象为独立的组件。组件之间可以彼此组合、嵌套，从而形成一个更大规模的模块

- 组件化的目标
  清晰的职责，松耦合，便于单元测试和重复利用。这里的松耦合不仅体现在js代码之间，也体现在js跟DOM之间的关系
- 一切皆模块
  整个开发过程中所有的设计都要尽可能的遵照把每一个业务功能上相对独立的模块定义成组件，然后将小的组件通过组合或者嵌套的方式构成大的组件，最终完成整体UI的构建

2. 组件化开发的特点
   - 可组合
     拆分简便，组件间可以互相独立又可以互相嵌套包容
   - 可重用
     功能独立，每个组件可以被使用在多个UI场景
   - 可维护
     每个组件仅仅包含自身的逻辑，更容易被理解和维护

React

1. JSX的概念
   JSX 乍看起来可能比较像是模版语言，但事实上它完全是在 JavaScript 内部实现的

2. Virtual Dom

   - 发起
     数据（state & props）变化时，React都会重新构建整个DOM树
     是否成批更新？有的话合并
   - 对比
     React将当前整个DOM树和上一次的DOM树进行对比，得到双方的diff（此过程不可见）
   - 渲染
     然后仅仅将需要变化的部分进行实际的浏览器DOM更新（页面更新）

   例如你连续的先将节点内容从A变成B，然后又从B变成A，React会认为UI不发生任何监听下次更新变化，而如果通过手动控制，这种逻辑通常是极其复杂的

3. React核心概念

    - 状态
      由用户输入（不限于键盘）改变，而维护的组件数据
    - 属性
      主要用于父层数据（状态或属性）向组件方向传递
    - 生命周期
      主要有三大类型：mount update unmount
    - render()
      数据改变后对比virtual dom，调用此渲染函数，更新相关页面

    React应用当中的绝大多数数据都是prop，只有当用户输入内容时才会使用state来处理

4. Virtual Dom的好处

    - 快，因为Javascript 快，Dom 慢
    - 过程简洁：初始状态开始→用户操作导致状态的变化→状态导致页面渲染更新
    - 开发人员不再过多关注dom，从而把精力投入到数据管理、页面状态和业务逻辑上
    - 浏览器对DOM操作批处理的主动权不在前端人员手中，React将这种批处理的时机选择交到了我们手中，看我们什么时候想render页面，我们只需要更新对应的数据即可
    - React 的设计非常简洁，没有规定架构的写法，使用方式非常灵活

Redux

单一Reactc程序缺点：随着前后端分离应用变得越来越复杂，那么对数据流动的控制就显得越来越重要。React 作为一个web数据状态机，由状态驱动页面的render()。
但是state在各个组件之间传递只能单向向子组件传递
不可避免的出现，在子组件向父组件或兄弟组件之间传递数据时，无法完成的尴尬情况

1. Redux
   Redux 是在flux的基础上产生的，用于管理一个SPA中的状态数据，基本思想是保证数据的单向流动，同时便于控制、使用、测试。

Redux不依赖于任意框架(库)，只要subscribe相应框架(库)的内部方法，就可以使用该应用框架保证数据流动的一致性

- 你的 Web 应用是一个数据状态机，视图与状态是一一对应的，视图是状态的反馈
- 所有的状态，保存在store中，这是redux维护的唯一对象
- Redux只是对一个方法的驱动和调用（ reducer ）

2. Redux运行过程