# 比较一下React与Vue

## 相同点

1. 都有组件化开发和Virtual DOM
2. 都支持props进行父子组件间数据通信
3. 都支持数据驱动视图, 不直接操作真实DOM, 更新状态数据界面就自动更新
4. 都支持服务器端渲染
5. 都有支持native的方案,React的React Native,Vue的Weex

## 不同点

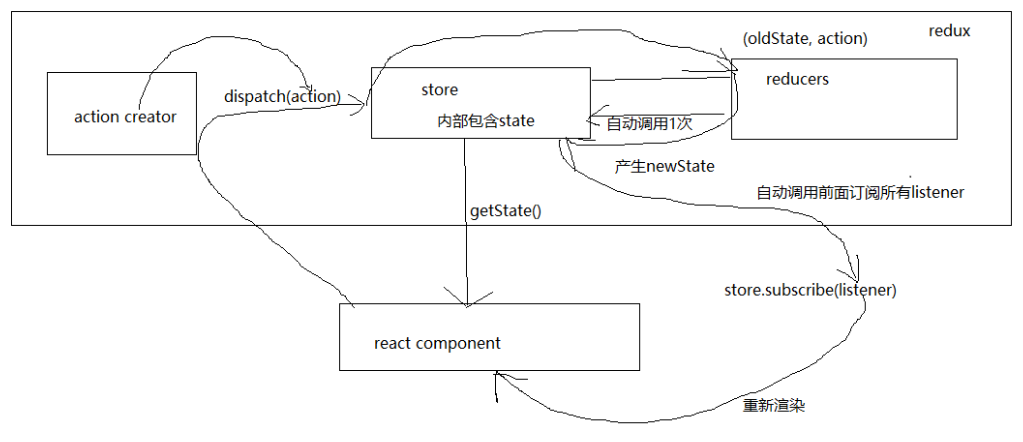
1. 数据绑定: vue实现了数据的双向绑定,react数据流动是单向的
2. 组件写法不一样, React推荐的做法是 JSX , 也就是把HTML和CSS全都写进JavaScript了,即'all in js'; Vue推荐的做法是webpack+vue-loader的单文件组件格式,即html,css,js写在同一个文件
3. state对象在react应用中不可变的,需要使用setState方法更新状态;在vue中,state对象不是必须的,数据由data属性在vue对象中管理
4. virtual DOM不一样,vue会跟踪每一个组件的依赖关系,不需要重新渲染整个组件树.而对于React而言,每当应用的状态被改变时,全部组件都会重新渲染,所以react中会需要shouldComponentUpdate这个生命周期函数方法来进行控制
5. React严格上只针对MVC的view层,Vue则是MVVM模式

# Redux管理状态的机制

## 对Redux基本理解

1. redux是一个独立专门用于做状态管理的JS库, 不是react插件库
2. 它可以用在react, angular, vue等项目中, 但基本与react配合使用
3. 作用: 集中式管理react应用中多个组件共享的状态和从后台获取的数据

## Redux的工作原理



## Redux使用扩展

1. 使用react-redux简化redux的编码
2. 使用redux-thunk实现redux的异步编程
3. 使用Redux DevTools实现chrome中redux的调试

# 说说Vue组件间通信方式

## 通信种类

1. 父组件向子组件通信
2. 子组件向父组件通信
3. 隔代组件间通信
4. 兄弟组件间通信

## 实现通信方式

1. props
2. vue自定义事件
3. 消息订阅与发布
4. vuex
5. slot

## 方式1: props

1. 通过一般属性实现父向子通信
2. 通过函数属性实现子向父通信
3. 缺点: 隔代组件和兄弟组件间通信比较麻烦

## 方式2: vue自定义事件

1. vue内置实现, 可以代替函数类型的props
   1. 绑定监听: <MyComp @eventName="callback"
   2. 触发(分发)事件: this.$emit("eventName", data)
2. 缺点: 只适合于子向父通信

## 方式3: 消息订阅与发布

1. 需要引入消息订阅与发布的实现库, 如: pubsub-js
   1. 订阅消息: PubSub.subscribe('msg', (msg, data)=>{})
   2. 发布消息: PubSub.publish(‘msg’, data)
2. 优点: 此方式可用于任意关系组件间通信

## 方式4: vuex

1. 是什么: vuex是vue官方提供的集中式管理vue多组件共享状态数据的vue插件
2. 优点: 对组件间关系没有限制, 且相比于pubsub库管理更集中, 更方便

## 方式5: slot

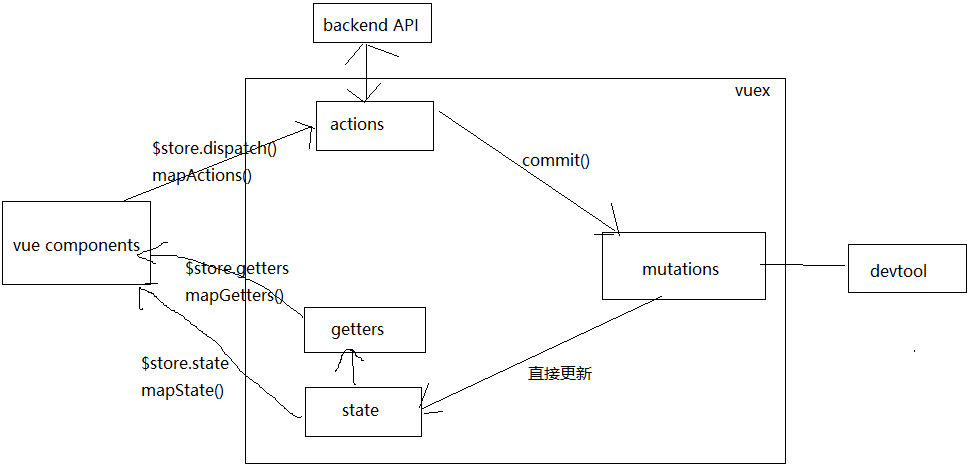
1. 是什么: 专门用来实现父向子传递带数据的标签
   1. 子组件
   2. 父组件
2. 注意: 通信的标签模板是在父组件中解析好后再传递给子组件的

# Vuex管理状态的机制

## 对Vuex基本理解

1. 是什么: Vuex 是一个专为 Vue.js 应用程序开发的状态管理的vue插件
2. 作用: 集中式管理vue多个组件共享的状态和从后台获取的数据

## Vuex的工作原理



# 说说Vue的MVVM实现原理

## 理解

1. Vue作为MVVM模式的实现库的2种技术
   1. 模板解析
   2. 数据绑定
2. 模板解析: 实现初始化显示
   1. 解析大括号表达式
   2. 解析指令
3. 数据绑定: 实现更新显示
   1. 通过数据劫持实现

## 原理结构图

