VUE

实现一下双向绑定

```
<body>
    <div id="app">
        <input type="text" id="txt">
        <p id="show-txt"></p>
    </div>
    <script>
        var obj = {}
        Object.defineProperty(obj, 'txt', {
            get: function () {
                return obj
            },
            set: function (newValue) {
                document.getElementById('txt').value = newValue
                document.getElementById('show-txt').innerHTML = newValue
            }
        })
        document.addEventListener('keyup', function (e) {
            obj.txt = e.target.value
        })
    </script>
</body>
```

1、scss是什么？在vue.cli中的安装使用步骤是？有哪几大特性？

答：预处理css，把css当前函数编写，定义变量,嵌套。css的预编译。

使用步骤：

第一步：用npm 下三个loader（sass-loader、css-loader、node-sass）

第二步：在build目录找到webpack.base.config.js，在那个extends属性中加一个拓展.scss

第三步：还是在同一个文件，配置一个module属性

第四步：然后在组件的style标签加上lang属性 ，例如：lang=”scss”

有哪几大特性:

1、可以用变量，例如（$变量名称=值）；

2、可以用混合器，例如（）

3、可以嵌套

2、请说出vue.cli项目中src目录每个文件夹和文件的用法？

答：assets文件夹是放静态资源；components是放组件；router是定义路由相关的配置;view视图；app.vue是一个应用主组件；main.js是入口文件

3、说出至少4种vue当中的指令和它的用法？

v-if：判断是否隐藏；v-for：数据循环出来；v-bind:class：绑定一个属性；v-model：实现双向绑定

4、自定义指令（v-check、v-focus）的方法有哪些？它有哪些钩子函数？还有哪些钩子函数参数？

答：全局定义指令：在vue对象的directive方法里面有两个参数，一个是指令名称，另外一个是函数。组件内定义指令：directives

钩子函数：bind（绑定事件触发）、inserted(节点插入的时候触发)、update（组件内相关更新）

钩子函数参数：el、binding

5、mint-ui是什么？怎么使用？说出至少三个组件使用方法？

答：基于vue的前端组件库。npm安装，然后import样式和js，vue.use（mintUi）全局引入。在单个组件局部引入：import {Toast} from 'mint-ui'。组件一：Toast('登录成功')；组件二：mint-header；组件三：mint-swiper

6、v-model是什么？怎么使用？ vue中标签怎么绑定事件？

答：可以实现双向绑定，指令（v-class、v-for、v-if、v-show、v-on）。vue的model层的data属性。绑定事件：<input @click=doLog() />

7、axios是什么？怎么使用？描述使用它实现登录功能的流程？

答：请求后台资源的模块。npm install axios -S装好，然后发送的是跨域，需在配置文件中config/index.js进行设置。后台如果是Tp5则定义一个资源路由。js中使用import进来，然后.get或.post。返回在.then函数中如果成功，失败则是在.catch函数中

8、axios+tp5进阶中，调用axios.post('api/user')是进行的什么操作？axios.put('api/user/8')呢？

答：跨域，添加用户操作，更新操作。

9、什么是RESTful API？怎么使用？

答：是一个api的标准，无状态请求。请求的路由地址是固定的，如果是tp5则先路由配置中把资源路由配置好。标准有：.post .put .delete

10、mvvm框架是什么？它和其它框架（jquery）的区别是什么？哪些场景适合？

答：一个model+view+viewModel框架，数据模型model，viewModel连接两个

区别：vue数据驱动，通过数据来显示视图层而不是节点操作。

场景：数据操作比较多的场景，更加便捷

11、请说下封装 vue 组件的过程？

答：首先，组件可以提升整个项目的开发效率。能够把页面抽象成多个相对独立的模块，解决了我们传统项目开发：效率低、难维护、复用性等问题。

然后，使用Vue.extend方法创建一个组件，然后使用Vue.component方法注册组件。子组件需要数据，可以在props中接受定义。而子组件修改好数据后，想把数据传递给父组件。可以采用emit方法。

12、vue如何实现父子组件通信，以及非父子组件通信？

1、父传子props

- props(v-bind)

- $refs

```
<parent>
    <child :child-msg="msg"></child>//这里必须要用 - 代替驼峰
</parent>

data(){
    return {
        msg: [1,2,3]
    };
}
```

```
props: {
    childMsg: {
        type: Array,
        default: [0,0,0] //这样可以指定默认的值
    }
}
```

2、子传父this.$emit（）

- events(v-on)

- $parent $root

```
子组件:
<template>
    <div @click="up"></div>
</template>

methods: {
    up() {
        this.$emit('upup','hehe'); //主动触发upup方法，'hehe'为向父组件传递的数据
    }
}
```

```
<div>
    <child @upup="change" :msg="msg"></child> //监听子组件触发的upup事件,然后调用change方法
</div>
methods: {
    change(msg) {
        this.msg = msg;
    }
}
```

3、非父子eventHub()

- event Hub

- vuex

```
let Hub = new Vue(); //创建事件中心
```

```
<div @click="eve"></div>
methods: {
    eve() {
        Hub.$emit('change','hehe'); //Hub触发事件
    }
}
```

```
<div></div>
created() {
    Hub.$on('change', () => { //Hub接收事件
        this.msg = 'hehe';
    });
}
```

## 13、Vue-router原理：

单页面应用(SPA)的核心之一是: 更新视图而不重新请求页面;vue-rouetr在实现单页面前端路由时，提供了两种方式：Hash模式和History模式；根据mode参数来决定采用哪一种方式。

1、Hash模式：

hash（#）是URL 的锚点，代表的是网页中的一个位置，单单改变#后的部分，浏览器只会滚动到相应位置，不会重新加载网页，也就是说 #是用来指导浏览器动作的，对服务器端完全无用，HTTP请求中也不会不包括#；同时每一

次改变#后的部分，都会在浏览器的访问历史中增加一个记录，使用”后退”按钮，就可以回到上一个位置；
2、History模式：
HTML5 History API提供了一种功能，能让开发人员在不刷新整个页面的情况下修改站点的URL，就是利用 history.pushState API 来完成 URL 跳转而无须重新加载页面；
通常情况下，我们会选择使用History模式，原因就是Hash模式下URL带着‘#’会显得不美观；但实际上，这样选择一不小心也会出问题；比如：
但当用户直接在用户栏输入地址并带有参数时：
Hash模式：xxx.com/#/id=5 请求地址为 xxx.com,没有问题;
History模式: xxx.com/id=5 请求地址为 xxx.com/id=5，如果后端没有对应的路由处理，就会返回404错误；
为解决这一问题，vue-router提供的方法是：
在服务端增加一个覆盖所有情况的候选资源：如果 URL 匹配不到任何静态资源，则应该返回同一个 index.html 页面，这个页面就是你 app 依赖的页面。
14、active-class是哪个组件的属性？嵌套路由怎么定义？
答：vue-router模块的router-link组件。（router-link、router-view）
15、怎么定义vue-router的动态路由？怎么获取传过来的动态参数？
答：在router目录下的index.js文件中，对path属性加上/:id。 使用router对象的params.id
路由导航钩子（导航守卫）（考察频率：中）

- 全局钩子
- 路由独享钩子
- 组件内钩子

16、vue-router有哪几种导航钩子？（导航钩子有哪些？它们有哪些参数？）
答：三种，一种是全局导航钩子：router.beforeEach(to,from,next)，作用：跳转前进行判断拦截。第二种：组件内的钩子；第三种：单独路由独享组件
导航钩子有：a/全局钩子和组件内独享的钩子。b/beforeRouteEnter、afterEnter、beforeRouterUpdate、beforeRouteLeave
参数：有to（去的那个路由）、from（离开的路由）、next（一定要用这个函数才能去到下一个路由，如果不用就拦截）最常用就这几种
17、vue生命周期
beforeCreate（创建前），
created（创建后），
创建前/后： 在beforeCreated阶段，vue实例的挂载元素$el和数据对象data都为undefined，还未初始化。在created阶段，vue实例的数据对象data有了，$el还没有。
beforeMount(载入前),

mounted（载入后），

载入前/后：在beforeMount阶段，vue实例的$el和data都初始化了，但还是挂载之前为虚拟的dom节点，data.message还未替换。在mounted阶段，vue实例挂载完成，data.message成功渲染。

beforeUpdate（更新前），

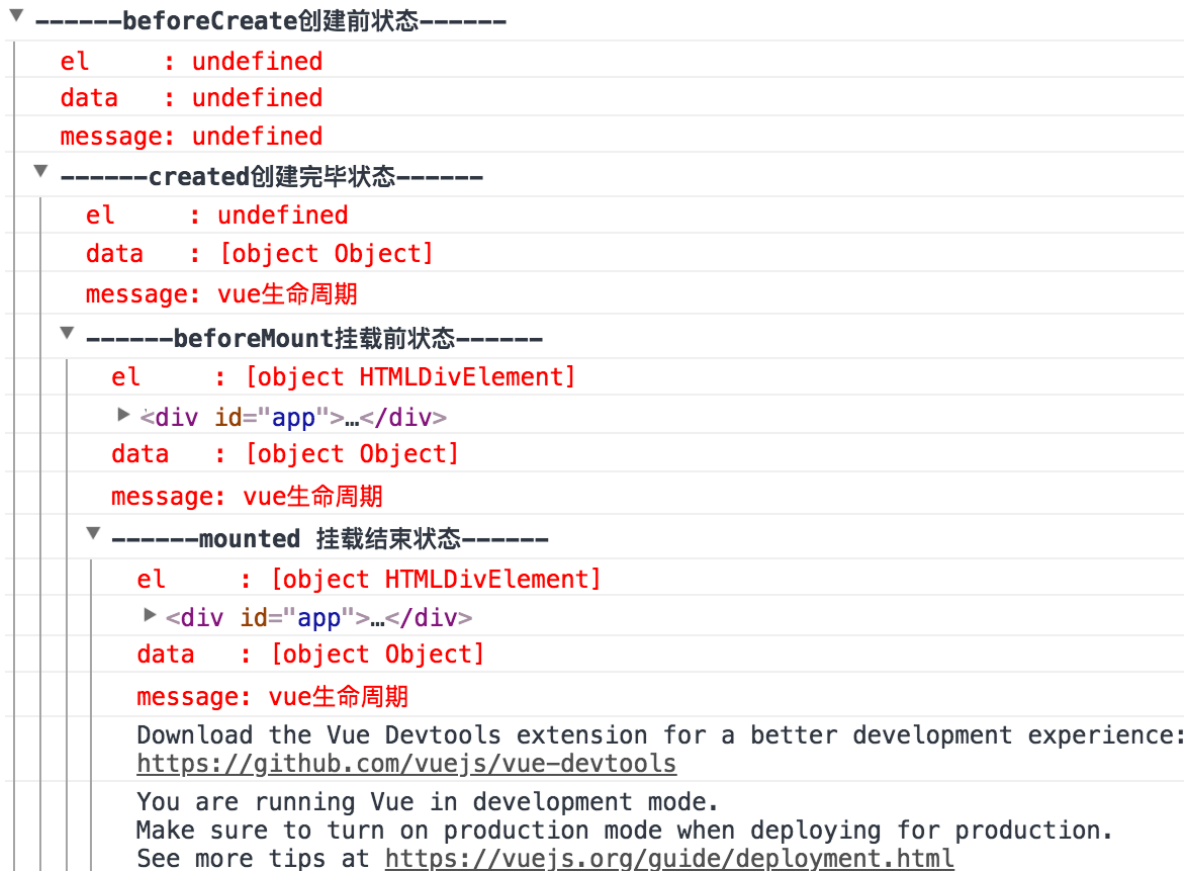
updated（更新后），

更新前/后：当data变化时，会触发beforeUpdate和updated方法。

beforeDestroy（销毁前），

destroyed（销毁后）

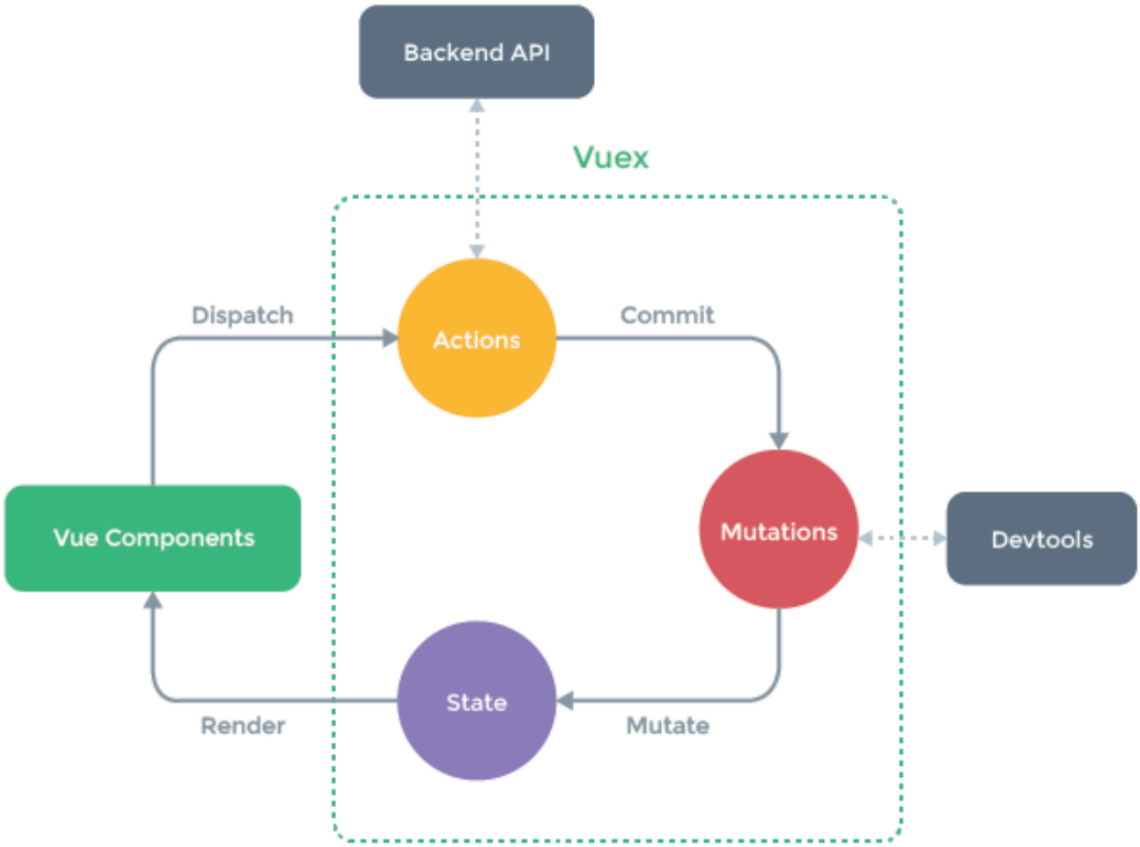
销毁前/后：在执行destroy方法后，对data的改变不会再触发周期函数，说明此时vue实例已经解除了事件监听以及和dom的绑定，但是dom结构依然存在




## 18、vuex工作流程

vue框架中状态管理。在main.js引入store，注入。新建了一个目录store，…..

export 。应用级的状态集中放在store中； 改变状态的方式是提交mutations，这是个同步的事物； 异步逻辑应该封装在action中。




- Vue Components：Vue组件。HTML页面上，负责接收用户操作等交互行为，执行dispatch方法触发对应action进行回应。
- dispatch：操作行为触发方法，是唯一能执行action的方法。
- actions：操作行为处理模块。负责处理Vue Components接收到的所有交互行为。包含同步/异步操作，支持多个同名方法，按照注册的顺序依次触发。向后台API请求的操作就在这个模块中进行，包括触发其他action以及提交mutation的操作。该模块提供了Promise的封装，以支持action的链式触发。
- commit：状态改变提交操作方法。对mutation进行提交，是唯一能执行mutation的方法。
- mutations：状态改变操作方法。是Vuex修改state的唯一推荐方法，其他修改方式在严格模式下将会报错。该方法只能进行同步操作，且方法名只能全局唯一。操作之中会有一些hook暴露出来，以进行state的监控等。
- state：页面状态管理容器对象。集中存储Vue components中data对象的零散数据，全局唯一，以进行统一的状态管理。页面显示所需的数据从该对象中进行读取，利用Vue的细粒度数据响应机制来进行高效的状态更新。
- getters：state对象读取方法。图中没有单独列出该模块，应该被包含在了render中，Vue Components通过该方法读取全局state对象。

1、Vue组件接收交互行为，调用dispatch方法触发action相关处理，
2、若页面状态需要改变，则调用commit方法提交mutation修改state，
3、通过getters获取到state新值，
4、重新渲染Vue Components，界面随之更新。

## 19、Vue的数据绑定的原理：

（vue.js 是采用数据劫持结合发布者-订阅者模式的方式，通过Object.defineProperty()来劫持各个属性的setter，getter，在数据变动时发布消息给订阅者，触发相应的监听回调。）

具体步骤：

第一步：需要observe的数据对象进行递归遍历，包括子属性对象的属性，都加上 setter和getter

这样的话，给这个对象的某个值赋值，就会触发setter，那么就能监听到了数据变化

第二步：compile解析模板指令，将模板中的变量替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，更新视图

第三步：Watcher订阅者是Observer和Compile之间通信的桥梁，主要做的事情是:

1、在自身实例化时往属性订阅器(dep)里面添加自己
2、自身必须有一个update()方法
3、待属性变动dep.notice()通知时，能调用自身的update()方法，并触发Compile中绑定的回调，则功成身退。

第四步：MVVM作为数据绑定的入口，整合Observer、Compile和Watcher三者，通过Observer来监听自己的model数据变化，通过Compile来解析编译模板指令，最终利用Watcher搭起Observer和Compile之间的通信桥梁，达到数据变化 -> 视图更新；视图交互变化(input) -> 数据model变更的双向绑定效果。

## 20、vue和react的数据绑定有什么区别

vue的数据劫持相当于一个观察监听的过程，而React的数据绑定是通过每次渲染时生成虚拟DOM ，手动通知更新数据动态

Vue在渲染过程中，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。

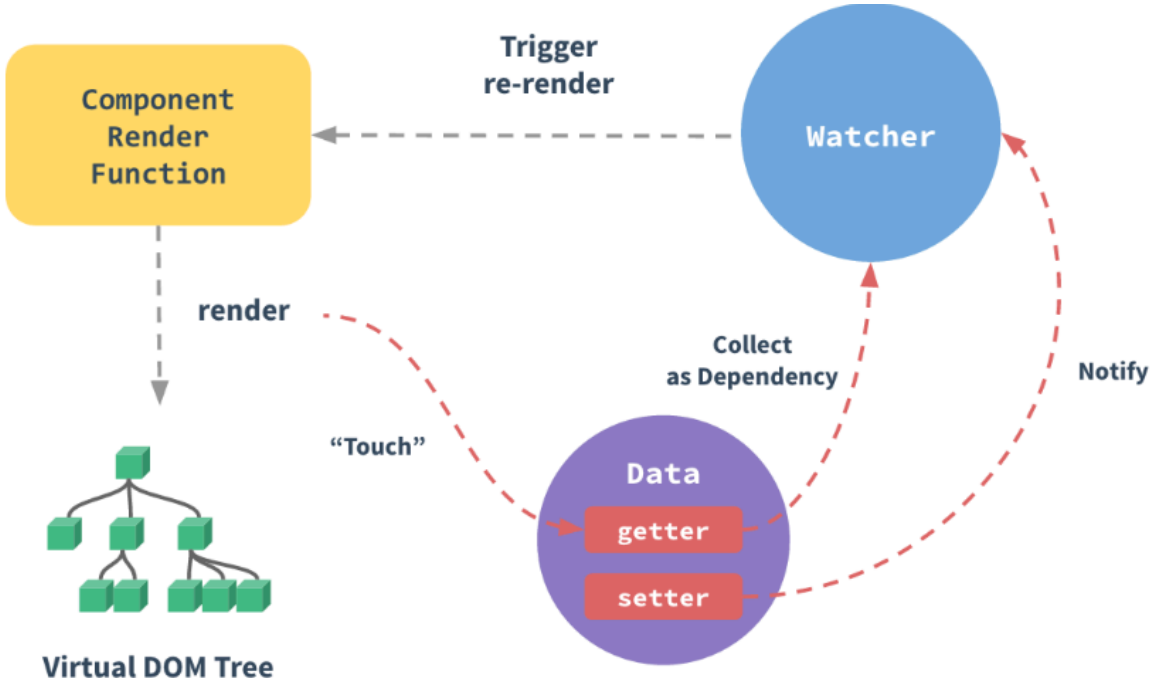
React每当应用的状态被改变时，全部子组件都会重新渲染。这可以通过shouldComponentUpdate这个生命周期方法来进行控制。

## Xx. react和vue比较来说有什么区别

1 component层面，web component和virtual dom
2 数据绑定（vue双向，react的单向）等好多
3 计算属性 vue 有，提供方便；而 react 不行
4 vue 可以 watch 一个数据项；而 react 不行
5 vue 由于提供的 direct 特别是预置的 directive 因为场景场景开发更容易；react

没有
6 生命周期函数名太长 directive

## 21、vue响应式原理?




1.把一个普通 JavaScript 对象传给 Vue 实例的 data 选项，Vue 将遍历此对象所有的属性，并使用 Object.defineProperty 把这些属性全部转为 getter/setter。
2.组件实例的 watcher 实例对象，
二：创建的Vue组件实例上添加响应式属性。 解决办法：将响应属性添加到嵌套的对象上
1.Vue.set(object, key, value) 还可以使用 vm.$set 实例方法，这也是全局 Vue.set 方法的别名。
2.this.someObject = Object.assign({}, this.someObject, { a: 1, b: 2 }) // 来代替 `Object.assign(this.someObject, { a: 1, b: 2 })`
三：异步更新队列
1.数据变化，添加到DOM更新队列。
异步组件

## # 异步组件

在大型应用中，我们可能需要将应用拆分为多个小模块，按需从服务器下载。为了进一步简化，Vue.js 允许将组件定义为一个工厂函数，异步地解析组件的定义。Vue.js 只在组件需要渲染时触发工厂函数，并且把结果缓存起来，用于后面的再次渲染。例如：

```JS
Vue.component('async-example', function (resolve, reject) {
  setTimeout(function () {
    // 将组件定义传入 resolve 回调函数
    resolve({
      template: '<div>I am async!</div>'
    })
  }, 1000)
})
```

工厂函数接收一个 `resolve` 回调，在收到从服务器下载的组件定义时调用。也可以调用 `reject(reason)` 指示加载失败。这里使用 `setTimeout` 只是为了演示，实际上如何获取组件完全由你决定。推荐配合 webpack 的代码分割功能 来使用：

```JS
Vue.component('async-webpack-example', function (resolve) {
  // 这个特殊的 require 语法告诉 webpack
  // 自动将编译后的代码分割成不同的块，
  // 这些块将通过 Ajax 请求自动下载。
  require(['./my-async-component'], resolve)
})
```

假设已经存在later1和later2组件

```vue
<template>
    <div>
        异步组件测试
        <template v-if='show'>
            <later></later>
            <later2></later2>
        </template>
        <button @click='toggle'>加载</button>
    </div>
</template>
<script>
    import Vue from 'vue';
    const later=Vue.compnent('later',function(resolve){
        setTimeout(function(){
            require(['./later.vue'],resolve)
        },3000)
    });
    const later2=Vue.compnent('later2',function(resolve){
        require(['./later2.vue'],resolve)
    });
    export default{
        data:function(){
            return{
                show:false
            }
        },
        components:{
            later,
            later2
        },
        created:function(){},
        mounted:function(){},
        computed:{},
        methods:{
            toggle:function(){
                this.show=!this.show
            }
        }
    }
</script>
```