Vue

一、对于**MVVM**的理解？

MVVM 是 Model-View-ViewModel 的缩写。
Model代表数据模型，也可以在Model中定义数据修改和操作的业务逻辑。
View 代表UI 组件，它负责将数据模型转化成UI 展现出来。
ViewModel 监听模型数据的改变和控制视图行为、处理用户交互，简单理解就是一个同步View 和 Model的对象，连接Model和View。
在MVVM架构下，View 和 Model 之间并没有直接的联系，而是通过ViewModel进行交互，Model 和 ViewModel 之间的交互是双向的， 因此View 数据的变化会同步到Model中，而Model 数据的变化也会立即反应到View 上。
ViewModel 通过双向数据绑定把 View 层和 Model 层连接了起来，而View 和 Model 之间的同步工作完全是自动的，无需人为干涉，因此开发者只需关注业务逻辑，不需要手动操作DOM, 不需要关注数据状态的同步问题，复杂的数据状态维护完全由 MVVM 来统一管理。

二、**Vue**的生命周期

beforeCreate（创建前） 在数据观测和初始化事件还未开始
created（创建后） 完成数据观测，属性和方法的运算，初始化事件，$el属性还没有显示出来
beforeMount（载入前） 在挂载开始之前被调用，相关的render函数首次被调用。实例已完成以下的配置：编译模板，把data里面的数据和模板生成html。注意此时还没有挂载html到页面上。
mounted（载入后） 在el 被新创建的 vm.$el 替换，并挂载到实例上去之后调用。实例已完成以下的配置：用上面编译好的html内容替换el属性指向的DOM对象。完成模板中的html渲染到html页面中。此过程中进行ajax交互。
beforeUpdate（更新前） 在数据更新之前调用，发生在虚拟DOM重新渲染和打补丁之前。可以在该钩子中进一步地更改状态，不会触发附加的重渲染过程。
updated（更新后） 在由于数据更改导致的虚拟DOM重新渲染和打补丁之后调用。调用时，组件DOM已经更新，所以可以执行依赖于DOM的操作。然而在大多数情况下，应该避免在此期间更改状态，因为这可能会导致更新无限循环。该钩子在服务器端渲染期间不被调用。
beforeDestroy（销毁前） 在实例销毁之前调用。实例仍然完全可用。
destroyed（销毁后） 在实例销毁之后调用。调用后，所有的事件监听器会被移除，所有的子实例也会被销毁。该钩子在服务器端渲染期间不被调用。
1.什么是vue生命周期？

答： Vue 实例从创建到销毁的过程，就是生命周期。从开始创建、初始化数据、编译模板、挂载Dom→渲染、更新→渲染、销毁等一系列过程，称之为 Vue 的生命周期。

2.vue生命周期的作用是什么？
答：它的生命周期中有多个事件钩子，让我们在控制整个Vue实例的过程时更容易形成好的逻辑。

3.vue生命周期总共有几个阶段？
答：它可以总共分为8个阶段：创建前/后, 载入前/后,更新前/后,销毁前/销毁后。

4.第一次页面加载会触发哪几个钩子？
答：会触发 下面这几个beforeCreate, created, beforeMount, mounted 。

5.DOM 渲染在 哪个周期中就已经完成？
答：DOM 渲染在 mounted 中就已经完成了。

三、 Vue实现数据双向绑定的原理：Object.defineProperty（）

vue实现数据双向绑定主要是：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，触发相应监听回调。当把一个普通 Javascript 对象传给 Vue 实例来作为它的 data 选项时，Vue 将遍历它的属性，用 Object.defineProperty 将它们转为 getter/setter。用户看不到 getter/setter，但是在内部它们让 Vue 追踪依赖，在属性被访问和修改时通知变化。

vue的数据双向绑定 将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的数据变化，通过Compile来解析编译模板指令（vue中是用来解析 {{}}），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变更双向绑定效果。

js实现简单的双向绑定

```
<body>
    <div id="app">
    <input type="text" id="txt">
    <p id="show"></p>
</div>
</body>
<script type="text/javascript">
    var obj = {}
    Object.defineProperty(obj, 'txt', {
        get: function () {
            return obj
```

```
        },
        set: function (newValue) {
            document.getElementById('txt').value = newValue
            document.getElementById('show').innerHTML = newValue
        }
    })
    document.addEventListener('keyup', function (e) {
        obj.txt = e.target.value
    })
</script>
```

四、Vue组件间的参数传递

1.父组件与子组件传值
父组件传给子组件：子组件通过props方法接受数据;
子组件传给父组件：$emit方法传递参数
2.非父子组件间的数据传递，兄弟组件传值
eventBus，就是创建一个事件中心，相当于中转站，可以用它来传递事件和接收事件。项目比较小时，用这个比较合适。（虽然也有不少人推荐直接用VUEX，具体来说看需求咯。技术只是手段，目的达到才是王道。）

五、Vue的路由实现：hash模式 和 history模式

hash模式：在浏览器中符号“#”，#以及#后面的字符称之为hash，用window.location.hash读取;
特点：hash虽然在URL中，但不被包括在HTTP请求中；用来指导浏览器动作，对服务端安全无用，hash不会重加载页面。
hash 模式下，仅 hash 符号之前的内容会被包含在请求中，如 http://www.xxx.com，因此对于后端来说，即使没有做到对路由的全覆盖，也不会返回 404 错误。

history模式：history采用HTML5的新特性；且提供了两个新方法：pushState（），replaceState（）可以对浏览器历史记录栈进行修改，以及popState事件的监听到状态变更。
history 模式下，前端的 URL 必须和实际向后端发起请求的 URL 一致，如 http://www.xxx.com/items/id。后端如果缺少对 /items/id 的路由处理，将返回 404 错误。**Vue-Router** 官网里如此描述：“不过这种模式要玩好，还需要后台配置支持……所以呢，你要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。”

六、Vue与Angular以及React的区别？

（版本在不断更新，以下的区别有可能不是很正确。我工作中只用到vue，对angular和react不怎么熟）

1.与**AngularJS**的区别

相同点：

都支持指令：内置指令和自定义指令；都支持过滤器：内置过滤器和自定义过滤器；都支持双向数据绑定；都不支持低端浏览器。

不同点：

AngularJS的学习成本高，比如增加了Dependency Injection特性，而Vue.js本身提供的API都比较简单、直观；在性能上，AngularJS依赖对数据做脏检查，所以Watcher越多越慢；Vue.js使用基于依赖追踪的观察并且使用异步队列更新，所有的数据都是独立触发的。

2.与**React**的区别

相同点：

React采用特殊的JSX语法，Vue.js在组件开发中也推崇编写.vue特殊文件格式，对文件内容都有一些约定，两者都需要编译后使用；中心思想相同：一切都是组件，组件实例之间可以嵌套；都提供合理的钩子函数，可以让开发者定制化地去处理需求；都不内置列数AJAX，Route等功能到核心包，而是以插件的方式加载；在组件开发中都支持mixins的特性。

不同点：

React采用的Virtual DOM会对渲染出来的结果做脏检查；Vue.js在模板中提供了指令，过滤器等，可以非常方便，快捷地操作Virtual DOM。

七、vue路由的钩子函数

首页可以控制导航跳转，beforeEach，afterEach等，一般用于页面title的修改。一些需要登录才能调整页面的重定向功能。

beforeEach主要有3个参数to，from，next：

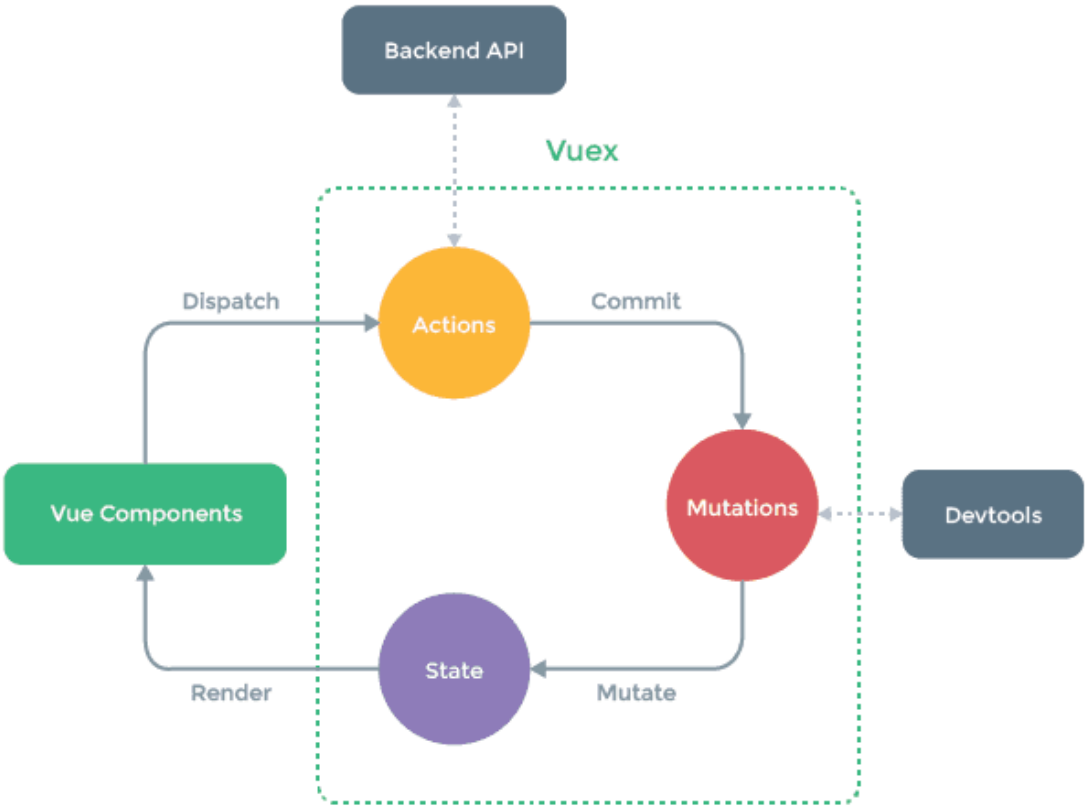
to：route即将进入的目标路由对象，

from：route当前导航正要离开的路由

next：function一定要调用该方法resolve这个钩子。执行效果依赖next方法的调用参数。可以控制网页的跳转。

八、vuex是什么？怎么使用？哪种功能场景使用它？

只用来读取的状态集中放在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。
在main.js引入store，注入。新建了一个目录store，..... export 。
场景有：单页应用中，组件之间的状态、音乐播放、登录状态、加入购物车



state
Vuex 使用单一状态树,即每个应用将仅仅包含一个store 实例，但单一状态树和模块化并不冲突。存放的数据状态，不可以直接修改里面的数据。
mutations
mutations定义的方法动态修改Vuex 的 store 中的状态或数据。
getters
类似vue的计算属性，主要用来过滤一些数据。
action
actions可以理解为通过将mutations里面处里数据的方法变成可异步的处理数据的方法，简单的说就是异步操作数据。view 层通过 store.dispath 来分发 action。

```
const store = new Vuex.Store({ //store实例
      state: {
         count: 0
             },
```

```
      mutations: {
         increment (state) {
          state.count++
         }
          },
      actions: {
         increment (context) {
          context.commit('increment')
   }
 }
})
```

modules

项目特别复杂的时候，可以让每一个模块拥有自己的state、mutation、action、getters,使得结构非常清晰，方便管理。

```
const moduleA = {
  state: { ... },
  mutations: { ... },
  actions: { ... },
  getters: { ... }
 }
const moduleB = {
  state: { ... },
  mutations: { ... },
  actions: { ... }
 }

const store = new Vuex.Store({
  modules: {
    a: moduleA,
    b: moduleB
})
```

九、**vue-cli**如何新增自定义指令？

1.创建局部指令

```
var app = new Vue({
    el: '#app',
    data: {
```

```
    },
    // 创建指令(可以多个)
    directives: {
        // 指令名称
        dir1: {
            inserted(el) {
                // 指令中第一个参数是当前使用指令的DOM
                console.log(el);
                console.log(arguments);
                // 对DOM进行操作
                el.style.width = '200px';
                el.style.height = '200px';
                el.style.background = '#000';
            }
        }
    }
})
```

2.全局指令

```
Vue.directive('dir2', {
    inserted(el) {
        console.log(el);
    }
})
```

3.指令的使用

```
<div id="app">
    <div v-dir1></div>
    <div v-dir2></div>
</div>
```

十、**vue**如何自定义一个过滤器？

html代码：

```
<div id="app">
     <input type="text" v-model="msg" />
     {{msg| capitalize }}
</div>
```

JS代码：

```
var vm=new Vue({
    el:"#app",
    data:{
        msg:''
    },
    filters: {
      capitalize: function (value) {
        if (!value) return ''
        value = value.toString()
        return value.charAt(0).toUpperCase() + value.slice(1)
      }
    }
})
```

全局定义过滤器

```
Vue.filter('capitalize', function (value) {
  if (!value) return ''
  value = value.toString()
  return value.charAt(0).toUpperCase() + value.slice(1)
})
```

过滤器接收表达式的值 (msg) 作为第一个参数。capitalize 过滤器将会收到 msg的值作为第一个参数。

十一、对**keep-alive** 的了解？

keep-alive是 Vue 内置的一个组件，可以使被包含的组件保留状态，或避免重新渲染。在vue 2.1.0 版本之后，keep-alive新加入了两个属性: include(包含的组件缓存) 与 exclude(排除的组件不缓存，优先级大于include) 。

使用方法

```
<keep-alive include='include_components'
exclude='exclude_components'>
  <component>
    <!-- 该组件是否缓存取决于include和exclude属性 -->
  </component>
</keep-alive>
```

参数解释
include - 字符串或正则表达式，只有名称匹配的组件会被缓存
exclude - 字符串或正则表达式，任何名称匹配的组件都不会被缓存
include 和 exclude 的属性允许组件有条件地缓存。二者都可以用“，”分隔字符串、正则表达式、数组。当使用正则或者是数组时，要记得使用v-bind 。

使用示例

```
<!-- 逗号分隔字符串，只有组件a与b被缓存。 -->
<keep-alive include="a,b">
  <component></component>
</keep-alive>

<!-- 正则表达式 (需要使用 v-bind，符合匹配规则的都会被缓存) -->
<keep-alive :include="/a|b/">
  <component></component>
</keep-alive>

<!-- Array (需要使用 v-bind，被包含的都会被缓存) -->
<keep-alive :include="['a', 'b']">
  <component></component>
</keep-alive>
```

十二、一句话就能回答的面试题

1.css只在当前组件起作用
答：在style标签中写入**scoped**即可 例如：

2.v-if 和 **v-show** 区别
答：v-if按照条件是否渲染，v-show是display的block或none；

3.$route**和**$router的区别
答：$route是“路由信息对象”，包括path，params，hash，query，fullPath，matched，name等路由信息参数。而$router是“路由实例”对象包括了路由的跳转方法，钩子函数等。

4.vue.js的两个核心是什么？
答：数据驱动、组件系统

5.vue几种常用的指令
答：v-for 、 v-if 、v-bind、v-on、v-show、v-else

6.vue常用的修饰符？
答：.prevent: 提交事件不再重载页面；.stop: 阻止单击事件冒泡；.self: 当事件发生在该元素本身而不是子元素的时候会触发；.capture: 事件侦听，事件发生的时候会调用

7.v-on 可以绑定多个方法吗？
答：可以

8.vue中 **key** 值的作用？
答：当 Vue.js 用 v-for 正在更新已渲染过的元素列表时，它默认用“就地复用”策略。如果数据项的顺序被改变，Vue 将不会移动 DOM 元素来匹配数据项的顺序， 而是简单复用此处每个元素，并且确保它在特定索引下显示已被渲染过的每个元素。key的作用主要是为了高效的更新虚拟DOM。

9.什么是**vue**的计算属性？
答：在模板中放入太多的逻辑会让模板过重且难以维护，在需要对数据进行复杂处理，且可能多次使用的情况下，尽量采取计算属性的方式。好处：①使得数据处理结构清晰；②依赖于数据，数据更新，处理结果自动更新；③计算属性内部this指向vm实例；④在template调用时，直接写计算属性名即可；⑤常用的是getter方法，获取数据，也可以使用set方法改变数据；⑥相较于methods，不管依赖的数据变不变，methods都会重新计算，但是依赖数据不变的时候computed从缓存中获取，不会重新计算。

10.vue等单页面应用及其优缺点
答：优点：Vue 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件，核心是一个响应的数据绑定系统。MVVM、数据驱动、组件化、轻量、简洁、高效、快速、模块友好。
缺点：不支持低版本的浏览器，最低只支持到IE9；不利于SEO的优化（如果要支持SEO，建议通过服务端来进行渲染组件）；第一次加载首页耗时相对长一些；不可以使用浏览器的导航按钮需要自行实现前进、后退。

11.怎么定义 **vue-router** 的动态路由**?** 怎么获取传过来的值
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 获取。