Vue 经典面试题

1. 说说什么是 MVVM?

- 这 MVVM 是 Model-View-ViewModel 的缩写。MVVM 是一种设计思想。Model 层代表数据模型，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，View 是一个同步 View 和 Model 的对象
- 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。
- 对 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。

2. 说 mvvm 和 mvc 区别？它和其它框架（jquery）的区别是什么?

- 这 mvc 和 mvvm 其实区别并不大。都是一种设计思想。主要就是 mvc 中 Controller 演变成 mvvm 中的 ViewModel。mvvm 主要解决了 mvc 中大量的 DOM 操作使页面渲染性能降低，加载速度变慢，影响用户体验。
- 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。

3. vue 的优点是什么?

- 低耦合。视图（View）可以独立于 Model 变化和修改，一个 ViewModel 可以绑定到不同的"View "上，当 View 变化的时候 Model 可以不变，当 Model 变化的时候 View 也可以不变。
- 可重用性。你可以把一些视图逻辑放在一个 ViewModel 里面，让很多 View 重用这段视图逻辑。
- 独立开发。开发人员可以专注于业务逻辑和数据的开发（ViewModel），设计人员可以专注于页面设计。
- 可测试。界面素来是比较难于测试的，而现在测试可以针对 ViewModel 来写。

4. 用于构建 vue 的 vue-cli 工程都到了哪些技术，其作用分别是什么?

- vue.js: vue-cli 工程的核心，主要特点是 双向数据绑定 和 组件系统
- vue-router: vue 官方推荐使用的路由框架
- vuex: 专为 vue.js 应用项目开发的状态管理器，主要用于维护 vue 组件间共用的一些变量和方法
- axios（fetch、ajax）:用于发起 GET、 POST 等 http 请求，基于 Promise 设计
- webpack: 模块加载和 vue-cli 工程打包器
- eslint 代码规范工具。

5. 用 vue-cli 工程常用的 npm 命令有哪些?

- `npm install` 下载 node_modules 资源包的命令
- `npm run dev` 启动 vue-cli 开发环境的 npm 命令
- `npm run build` vue-cli 生成 生产环境部署资源 的 npm 命令
- `npm run build--report` 用于查看 vue-cli 生产环境部署资源文件大小的 npm 命令

6. 请说出 vue-cli 工程中每个文件夹和文件的用处?

- `build` 文件夹是保存一些 `webpack` 的初始化配置。
- `config` 文件夹保存一些项目初始化的配置
- `node_modules` 是 npm 加载的项目依赖的模块
- `src` 目录是我们要开发的目录:
  - `assets` 用来放置图片
  - `components` 用来放组件文件
  - `app.vue` 是项目入口文件
  - `main.js` 项目的核心文件

7. 请你详细介绍一些 package.jso 里面的配置?

- 简单关键字说明

```
{
  "name": "demo",
  "version": "1.0.0",
  "description": "这里是介绍",
  "main": "index.js",
  "scripts": {
    "build": "webpack --config webpack.config.js",
    "sync": "gulp sync",
    "syncbuild": "gulp sync && webpack --config webpack.config.js"
  },
  "author": "will",
  "license": "ISC",
  "dependencies": {
    "react": "16.*"
  },
  "devDependencies": {
    "babel-core": "^6.26.0",
    "eslint": "^4.10.0",
  }
}
```

- version 标注了当前项目的版本号，如果是私有项目，不发布到仓库（repository）,此项关键字可以不用变更
- main 我们模块或者是项目的入口文件，且只有一个文件，文件的路径是相对项目的根目录
- license 是标注我们项目的许可证权限，如果是开源项目

- 重点关键字说明

- scripts 是 npm 提供给我们运行 shell 命令的入口
- dependencies 我是项目运行时必要依赖
- devDependencies 是是项目开发时所需依赖

8. 为什么说 vue 是一个渐进式框架?

原首先我们需要理解什么是框架。在最初的前端开发中，为了完成一些功能，我们需要通过原生 js 获取 html 中的 DOM 节点，随后插入内容或添加事件，进行一系列操作。但是，一般需求的业务逻辑都是比较复杂的，这对原生 js 处理起来就很吃力，代码的可维护性也就随时间慢慢降低。这个时候我们需要把视图(View)、数据(Model)、逻辑控制(VM)分离。这就是 MVVM 模式的雏形。

形在核心库的基础上，vue 现在已经有了一套完整的生态系统，我们可以添加 Vuex，可以添加 vue-router,可以添加任何你想添加的第三方库，这些都是互相独立的。可以说 vue 是轻量型的，是渐进型的

- Vue 与 React、Angular 的不同是，但它是渐进的：

- 你可以在原有大系统的上面，把一两个组件改用它实现，当 jQuery 用
- 也可以整个用它全家桶开发，当 Angular 用
- 可以用它的视图，搭配你自己设计的整个下层用
- 可以在底层数据逻辑的地方用 OO 和设计模式的那套理念
- 也可以函数式，都可以，它只是个轻量视图而已，只做了最核心的东西

9. 请问 v-if 和 v-show 有什么区别?

- 共同点：都是动态显示 DOM 元素
- 区别点：

- **手段**
  - v-if 是动态的向 DOM 树内添加或者删除 DOM 元素
  - v-show 是通过设置 DOM 元素的 display 样式属性控制显隐
- **编译过程**
  - v-if 切换有一个局部编译/卸载的过程，切换过程中合适地销毁和重建内部的事件监听和子组件
  - v-show 只是简单的基于 css 切换
- **编译条件**

    - `v-if` 是惰性的，如果初始条件为假，则什么也不做；只有在条件第一次变为真时才开始局部编译（编译被缓存？编译被缓存后，然后再切换的时候进行局部卸载)态的
    - `v-show` 是在任何条件下（首次条件是否为真）都被编译，然后被缓存，而且 `DOM` 元素保留
- **性能消耗**
    - `v-if` 有更高的切换消耗
    - `v-show` 有更高的初始渲染消耗
- **使用场景**
    - `v-if` 适合运营条件不大可能改变
    - `v-show` 适合频繁切换

10. vue 常用的修饰符?

- v-model 修饰符：
    - `.lazy`：输入框改变，这个数据就会改变，这个修饰符会在光标离开 input 框才会更新数据

```
<input type="text" v-model.lazy="value">
```

    - `.number`：营先输入数字就会限制输入只能是数字，先字符串就相当于没有加 number，注意，不是输入框不能输入字符串，是这个数据是数字

```
<input type="text" v-model.number="value">
```

    - `.trim`：输入框过滤首尾的空格

```
<input type="text" v-model.trim="value">
```

- 事件修饰符：
    - `.stop`：阻止事件冒泡，相当于调用了 event.stopPropagation()方法

```
<button @click.stop="test"></button>
```

    - `.prevent`：阻止默认行为，相当于调用了 event.preventDefault()方法，比如表单的提交、a 标签的跳转就是默认事件

```
<a href="" @click.prevent="test"></a>
```

    - `.self`：只有元素本身触发时才触发方法，就是只有点击元素本身才会触发。比如一个 div 里面有个按钮，div 和按钮都有事件，我们点击按钮，div 绑定的方法也会触发，如果 div 的 click 加上 self，只有点击到 div 的时候才会触发，变相的算是阻止冒泡

```
<div @click.stop="test"></div>
```

- .once：事件只能用一次，无论点击几次，执行一次之后都不会再执行

```
<div @click.once="test"></div>
```

- .capture：事件的完整机制是捕获-目标-冒泡，事件触发是目标往外冒泡
- .sync：对 prop 进行双向绑定
- .keyCode：监听按键的指令，具体可以查看 vue 的键码对应表

11. v-on 可以监听多个方法吗?

- 关于 v-on 监听多个方法的简写：

```
<input type="text" v-on="{ input:onInput, focus:onFocus, blur:onBlur,}"/>

<!-- v-on 在 vue2.x 中测试,以下两种均可-->
<button v-on="{mouseenter: onEnter,mouseleave: onLeave}">鼠标进来 1</button>
<button @mouseenter="onEnter" @mouseleave="onLeave">鼠标进来 2</button>

<!-- 一个事件绑定多个函数，按顺序执行，这里分隔函数可以用逗号也可以用分号-->
<button @click="a(),b()">点我 ab</button>
<button @click="one()">点我 onetwothree</button>

<!-- v-on 修饰符 .stop .prevent .capture .self 以及指定按键.{keyCode|keyAlias} -->
<!-- 这里的.stop 和 .prevent 也可以通过传入&event 进行操作 -->
<!-- 全部按键别名有：enter tab delete esc space up down left right -->
<form @keyup.delete="onKeyup" @submit.prevent="onSubmit">
<input type="text" placeholder="在这里按 delete">
<button type="submit">点我提交</button>
```

12. vue 中 key 值的作用?

- key 值：用于 管理可复用的元素。因为 Vue 会尽可能高效地渲染元素，通常会复用已有元素而不是从头开始渲染。这么做使 Vue 变得非常快，但是这样也不总是符合实际需求

13. vue 事件中如何使用 event 对象?

- 获取事件对象，方法参数传递 $event 。注意在事件中要使用 $ 符号

```
<button @click="Event($event)">事件对象</button>
```

14. $nextTick 的使用?

- this.$nextTick()将回调延迟到下次 DOM 更新循环之后执行。在修改数据之后立即使用它，然后等待 DOM 更新。它跟全局方法 Vue.nextTick 一样，不同的是回调的 this 自动绑定到调用它的实例上
- this.$nextTick()在页面交互，尤其是从后台获取数据后重新生成 DOM 对象之后的操作有很大的优势

15. vue 组件中 data 为什么必须是函数?

- Object 是引用数据类型,如果不用 function 返回,每个组件的 data 都是内存的同一个地址,一个数据改变了其他也改变了
- javascipt 只有函数构成作用域(注意理解作用域,只有函数的{}构成作用域,对象的{}以及 if(){}都不构成作用域)，data 是一个函数时，每个组件实例都有自己的作用域，每个实例相互独立,不会相互影响
- 在创建或注册模板的时候传入一个 data 属性作为用来绑定的数据。但是在组件中，data 必须是一个函数，因为每一个 vue 组件都是一个 vue 实例，通过 new Vue() 实例化，引用同一个对象，如果 data 直接是一个对象的话，那么一旦修改其中一个组件的数据，其他组件相同数据就会被改变，而 data 是函数的话，每个 vue 组件的 data 都因为函数有了自己的作用域，互不干扰

16. v-for 与 v-if 的优先级?

- v-for 和 v-if 同时使用，有一个先后运行的优先级，v-for 比 v-if 优先级更高，这就说明在 v-for 每次的循环赋值中每一次调用 v-if 的判断，所以不推荐 v-if 和 v-for 在同一个标签中同时使用。

17. vue 中子组件调用父组件的方法?

- 直接在子组件中通过 this.$parent.event 来调用父组件的方法。
- 在子组件里用$emit()向父组件触发一个事件，父组件监听这个事件就行了。
- 父组件把方法传入子组件中，在子组件里直接调用这个方法。

18. vue 中组件 keep-alive 的作用?

- keep-alive：主要用于保留组件状态或避免重新渲染。
- 属性：
  - include 字符串或正则表达式。只有匹配的组件会被缓存。
  - exclude 字符串或正则表达式。任何匹配的组件都不会被缓存。

- 用法：包裹动态组件时，会缓存不活动的组件实例，而不是销毁它们。和 `<transition>`相似，`<keep-alive>`是一个抽象组件：它自身不会渲染一个 DOM 元素，也不会出现在父组件链中。

19. 什么是 vue 生命周期和生命周期钩子函数?

- 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。
- 创建前/后： 在 `beforeCreate` 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
- 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
- 更新前/后：当 `data` 变化时，会触发 `beforeUpdate` 和 `updated` 方法
- 属销毁前/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

20. vue 如何监听键盘事件中的按键?

原在一些搜索框中，我们往往需要监听键盘的按下(`onkeydown`)或抬起(`onkeyup`)事件以进行一些操作。在原生 js 或者 jQuery 中，我们需要判断 `e.keyCode` 的值来获取用户所按的键。这样就存在一个问题：我们必须知道某个按键的 `keyCode` 值才能完成匹配，使用起来十分不便。

- 在 Vue 中，已经为常用的按键设置了别名，这样我们就无需再去匹配 keyCode，直接使用别名就能监听按键的事件

别名	实际键值
.delete	delete（删除）/BackSpace（退格）
.tab	Tab
.enter	Enter（回车）
.esc	Esc（退出）
.space	Space（空格键）
.left	Left（左箭头）
.up	Up（上箭头）
.right	Right（右箭头）
.down	Down（下箭头）
.ctrl	Ctrl
.alt	Alt
.shift	Shift
.meta	(window系统下是window键，mac下是command键)

```
<input @keyup.enter="function">
```

21. vue 更新数组时触发视图更新的方法?

- Vue 包含一组观察数组的变异方法，所以它们也将会触发视图更新。这些方法如下：
  - push()
  - pop()
  - shift()
  - unshift()
  - splice()
  - sort()
  - reverse()
- filter(), concat()和 slice() 。这些不会改变原始数组，但总是返回一个新数组。当使用这些非变异方法时，可以用新数组替换旧数组
- JavaScript 的限制，Vue 不能检测以下变动的数组：
  - 当你利用索引直接设置一个项时，例如：vm.items[indexOfItem] = newValue
  - 当你修改数组的长度时，例如：vm.items.length = newLength

```
// 举个例子：
var vm = new Vue({
    data: {
        items: ['a', 'b', 'c']
    }
})
vm.items[1] = 'x' // 不是响应性的
vm.items.length = 2 // 不是响应性的
// 为了解决第一类问题，以下两种方式都可以实现和 vm.items[indexOfItem] = newValue 相同的效果，同时也将触发状态更新：
// Vue.set
Vue.set(vm.items, indexOfItem, newValue)
// Array.prototype.splice
vm.items.splice(indexOfItem, 1, newValue)
// 你也可以使用 vm.$set 实例方法，该方法是全局方法 Vue.set 的一个别名：
vm.$set(vm.items, indexOfItem, newValue)
// 为了解决第二类问题，你可以使用 splice：
vm.items.splice(newLength)
```

22. 对象更改注意事项

- Vue 无法检测到对象属性的添加和删除。对于已经创建的实例，Vue 不能动态添加根级别的响应式属性，可以使用 Vue.set(object, key, value) 方法向嵌套对象添加响应式属性。
- Vue 不能动态添加根级别的响应式属性：

```
var vm = new Vue({
    data: {
        a: 1
    }
})
vm.b = 2;// vm.a 现在是响应式的，vm.b 不是响应式的
```

- 向嵌套对象添加响应式属性：

```
var vm = new Vue({
    data: {
        userProfile: {
            name: 'Anika'
        }
    }
})
// 问题
vm.userProfile.age = 27;// 非响应
// 解决方案
Vue.set(vm.userProfile, 'age', 27);// 方案一
vm.$set(vm.userProfile, 'age', 27);// 等同方案一
vm.userProfile = Object.assign({}, vm.userProfile, {
    age: 27,
    favoriteColor: 'Vue Green'
});// 方案二
//方案二是用两个对象的属性创建一个新的对象，注意不要使用以下方式，因为此种方式是与
vm.userProfile.age = 27 的本质是一样的，均是非响应属性。
Object.assign(vm.userProfile, {
    age: 27,
    favoriteColor: 'Vue Green'
})
```

23. 如何解决非工程化项目，网速慢时初始化页面闪动问题？

- 使用 v-cloak 指令，v-cloak 不需要表达式，它会在 Vue 实例结束编译时从绑定的 HTML 元素上移除，经常和 CSS 的 display:none 配合使用。

```
<div id="app" v-cloak>{{message}}</div>
<script>
    var app = new Vue({el:"#app",
        data:{
            message:"这是一段文本"
        }
    })
</script>
```

- 这时虽然已经加了指令 v-cloak，但其实并没有起到任何作用，当网速较慢、Vue.js 文件还没加载完时，在页面上会显示{{ message }}的字样，直到 Vue 创建实例、编译模版时，DOM 才会被替换，所以这个过程屏幕是有闪动的。只要加一句 CSS 就可以解决这个问题了

```
[v-cloak]{
    display:none;
}
```

24. v-for 产生的列表，实现 active 的切换?

- v-for 生成序列。

```
<ul>
    <li v-for="(info,index) in list" :key="info.id" @click="select(index)" v-bind:class="{'active':info.active}">{{info.name}}</li>
</ul>
```

- data 数据

```
list:[
    {name:'a',id:1,active:false},
    {name:'b',id:2,active:false},
]
```

- 点击事件

```
select(d){
    this.list.map(s=>s.active=false);
    this.list[d].active=true;
},
```

25. v-model 语法糖的组件中的使用?

- v-model 主要是用于表单上数据的双向绑定。
- 主要用于 input，select，textarea，component
- 修饰符：
  - .lazy 取代 input 监听 change 事件
  - .number 输入字符串转为数字
  - .trim 输入首尾空格过滤
- vue 里面 v-model 的语法糖：

```
<custom v-model="something"></custom>
```

约等于：

```
<custom :value="something" @input="value => { something = value }"></custom>
```

- 这样有利用我们手动实现双向绑定，在子组件里面，首先在 `props` 里面接收一下 `value` 值，然后初始化到 `newValue` 里面，然后监听 `newValue` 值变化，变化后发射事件到父组件
- 到父组件通过 `v-model` 绑定 `index` 到子组件上，而通过 `v-model` 绑定的数据在子组件中默认的 `key` 是 `value`，所以在 `props` 上用 `value` 取值，最后通过点击事件执行`$emit`,而`$emit` 上触发的事件是 `input`，前面我们说过 `v-model` 绑定的事件是 `input`，从而在子组件上触发了父组件的 `input` 事件，而在触发事件的时候可以传值，所以就实现了父子组件数据双向绑定，如果用的是 `v-bind`，还需要自己去定义事情，所以使用 `v-model` 可以减少代码量。

26. Keep-Alive 的作用是什么?

- `<keep-alive></keep-alive>` 包裹动态组件时，会缓存不活动的组件实例,主要用于保留组件状态或避免重新渲染
- 比如有一个列表和一个详情，那么用户就会经常执行打开详情=>返回列表=>打开详情...这样的话列表和详情都是一个频率很高的页面，那么就可以对列表组件使用`<keep-alive></keep-alive>`进行缓存，这样用户每次返回列表的时候，都能从缓存中快速渲染，而不是重新渲染

27. 什么是 vue 的计算属性?

- 我们可以将同一函数定义为一个方法而不是一个计算属性。两种方式的最终结果确实是完全相同的。然而，不同的是计算属性是基于它们的依赖进行缓存的。只在相关依赖发生改变时它们才会重新求值。这就意味着只要 `message` 还没有发生改变，多次访问 `reversedMessage` 计算属性会立即返回之前的计算结果，而不必再次执行函数。

```
// 在组件中
computed: {
    reversedMessage: function () {
        return this.message.split('').reverse().join('')
    }
}
```

- 计算属性用法
  - 在一个计算属性里可以完成各种复杂的逻辑，包括运算、函数调用等，只要最终返回一个结果就可以。除了上例简单的用法，计算属性还可以依赖多个 Vue 实例的数据，只要其中任一数据变化，计算属性就会重新执行，视图也会更新
  - `getter` 和 `setter` 方法: 每一个计算属性都包含一个 `getter` 方法和一个 `setter` 方法
- 如果计算属性后面直接跟一个 `function`，使用的就是计算属性的默认方法 `getter` 来读取
- 我们也可以在需要时使用 `setter` 函数 ， 当手动修改计算属性的值就像修改一个普通数据那样时，就会触发 `setter` 函数，执行一些自定义的操作

```
computed: {
    fullName: {
        get: function(){
            return this.firstName + ' ' + this.lastName
        },
        set: function(newValue){
            // 传进来的值用逗号分隔，如'Liu,Bei'
            var names = newValue.split(',') // 分隔为数组
            this.firstName = names[0]
            this.lastName = names[1]
        }
    }
},
```

- 计算属性可以依赖其他计算属性
- 计算属性不仅可以依赖当前 Vue 实例的数据，还可以依赖其他实例的数据
- 计算属性缓存是定义在计算属性 `computed` 里的
- 计当需要数据实时发生变化时，适合用 `methods`

28. vue 父组件如何向子组件中传递数据?

- 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
- 计算属性不仅可以依赖当前 Vue 实例的数据，还可以依赖其他实例的数据
  - `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）

29. vue 弹窗后如何禁止滚动条滚动?

```
methods: {
    //禁止滚动
    stop() {
      var mo = function(e) {
        e.preventDefault();
      };
      document.body.style.overflow = "hidden";
      document.addEventListener("touchmove", mo, false); //禁止页面滑动
    },
```

```
      /***取消滑动限制***/
      move() {
        var mo = function(e) {
          e.preventDefault();
        };
        document.body.style.overflow = ""; //出现滚动条
        document.removeEventListener("touchmove", mo, false);
      }
    }

```

30. 计算属性的缓存和方法调用的区别?

- 我们可以将同一函数定义为一个方法或是一个计算属性。两种方式的最终结果确实是完全相同的。不同的是计算属性是基于它们的依赖进行缓存的。只在相关依赖发生改变时它们才会重新求值。相比之下，每当触发重新渲染时，调用方法将总会再次执行函数
- 使用计算属性还是 `methods` 取决于是否需要缓存，当遍历大数组和做大量计算时，应当使用计算属性，除非你不希望得到缓存。
- 计算属性是根据依赖自动执行的，`methods` 需要事件调用

31. vue-cli 如何新增自定义指令?

- 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';
            }
        }
    }
})
```

- 全局指令

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

- 指令的使用

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

32. 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

33. vuex 有哪几种属性?

- 分别是 State、 Getter、Mutation 、Action、 Module
- `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
- `Module`：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

34. vuex 的核心概念和使用方法?

- `state`：数据源，载体
- `getters`：用于改变 state 的值，派生出多个数据源
- `mutation`：唯一可以提交可以改变 state 的状态，也就是数据的属性值
- `actions`：提交的是 mutation,用 commit 提交 而不是直接变更状态，可以包含任意异步出操作

- `modules`：拆分成多个模块

```
// vuex 核心概念
// state
// 通过计算属性改变值
computed:{
    count(){
        //定义语法
        return this.$store.state.count
    }
}
// 通过 getters 可以派生出一些新的状态
const store =new Vuex.Store({
    state:{
        todos:[
            {id:1,text;'...',done:true}
        ]
    },
    //通过此方法对 todos 数据源进行过滤，拿到新的数据源赋值给 doneTodos
    getters:{
        doneTodos:state=>{
            return state.todos.filter(tod0=>todo.done)
        }
    },
})
// mutations
// 更改 Vuex 的 store 中的状态的唯一方法，也只能通过 mutations 去改变值
const store=new Vux.Store({
    state:{
        count:1
    },
    mutations:{
        //随便自定义一个函数，参数是默认 state
        add(state){
        //通过 state.count 拿到 count 属性值在进行自增
            state.count++
        }
    },
    actions:{
        add(context){
            //通过 commit 的方法提交一个 add,这时就会调用 mutation 中的 add 方法就会加加 1
            context.commit("add")
        }
    }
})
```

35. 在 vuex 中使用异步修改?

- 在调用 vuex 中的方法 action 的时候，用 promise 实现异步修改

```
const actions = {
  asyncInCrement({ commit }, n) {
    return new Promise(resolve => {
      setTimeout(() => {
        commit(types.TEST_INCREMENT, n);
        resolve();
      }, 3000);
    });
  }
};
```

36. pc 端页面刷新时实现 vuex 缓存?

- Localstorage 这是很容易想到的方法。将 this.$store.state.vueX 存入到缓存里之后，然后监听页面重载事件，如果页面重新载入了，那就从缓存里读取数据，然后赋值，这样我们的 dom 元素就又可爱的回到了文档中间
- 如果有缓存的情况下 computed 会优先使用缓存，在 activity.vue 里写下来 computed

```
computed: {
    posVerson() {
        return this.$store.state.vueX
    }
}
```

  - 纯响应式，computed 里面所用到的 data 一旦改变，整个 computed 的方法就回重新计算这个属性值
  - 计算结果会被缓存起来，方便下次使用，如果下次调用的时候，其中的数据没有发生变化，则不会重新计算。

37. 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
- history 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

38. 实现 Vue SSR?

- app.js 作为客户端与服务端的公用入口，导出 Vue 根实例，供客户端 entry 与服务端 entry 使用。客户端 entry 主要作用挂载到 DOM 上，服务端 entry 除了创建和返回实例，还进行路由匹配与数据预获取。
- webpack 为客服端打包一个 Client Bundle ，为服务端打包一个 Server Bundle 。
- 服务器接收请求时，会根据 url，加载相应组件，获取和解析异步数据，创建一个读取 Server Bundle 的 BundleRenderer，然后生成 html 发送给客户端。
- 客户端混合，客户端收到从服务端传来的 DOM 与自己的生成的 DOM 进行对比，把不相同的 DOM 激活，使其可以能够响应后续变化，这个过程称为客户端激活 。为确保混合成功，客户端与服务器端需要共享同一套数据。在服务端，可以在渲染之前获取数据，填充到 store 里，这样，在客户端挂载到 DOM 之前，可以直接从 store 里取数据。首屏的动态数据通过 window.__INITIAL_STATE__ 发送到客户端
- Vue SSR 的实现，主要就是把 Vue 的组件输出成一个完整 HTML, vue-server-renderer 就是干这事的
- Vue SSR 需要做的事多点（输出完整 HTML），除了 complier -> vnode，还需如数据获取填充至 HTML、客户端混合（hydration）、缓存等等。 相比于其他模板引擎（ejs, jade 等），最终要实现的目的是一样的，性能上可能要差点

39. vue-router 如何响应路由参数的变化?

- 当使用路由参数时，例如从 /user/foo 导航到 /user/bar，原来的组件实例会被复用。因为两个路由都渲染同个组件，比起销毁再创建，复用则显得更加高效。不过，这也意味着组件的生命周期钩子不会再被调用。
- 监测路由参数变化的方法

✧ watch 监听：

```
✧  watch: { // watch 的第一种写法
✧      $route (to, from) {
✧          console.log(to)
✧          console.log(from)
✧      }
✧  },
✧  // ====
✧  watch: { // watch 的第二种写法
✧      $route: {
✧          handler (to, from){
✧              console.log(to)
✧              console.log(from)
✧          },
✧          // 深度观察监听
✧          deep: true
✧      }
✧  }
```

- 导航守卫

```
beforeRouteEnter (to, from, next) {
    console.log('beforeRouteEnter 被调用：在渲染该组件的对应路由被 confirm 前调用')
    // 在渲染该组件的对应路由被 confirm 前调用
    // 不！能！获取组件实例 `this` 因为当守卫执行前，组件实例还没被创建
    // 可以通过传一个回调给 next 来访问组件实例。在导航被确认的时候执行回调，并且把组件实例作为回调方法的参数。
    next(vm => {
        // 通过 `vm` 访问组件实例
        console.log(vm)
    })
},
// beforeRouteEnter 是支持给 next 传递回调的唯一守卫。
// 对于 beforeRouteUpdate 和 beforeRouteLeave 来说，this 已经可用了，所以不支持传递回调，因为没有必要了。
beforeRouteUpdate (to, from, next) {
    // 在当前路由改变，但是该组件被复用时调用
    // 举例来说，对于一个带有动态参数的路径 /foo/:id，在 /foo/1 和 /foo/2 之间跳转的时候，
    // 由于会渲染同样的 Foo 组件，因此组件实例会被复用。而这个钩子就会在这个情况下被调用。
    // 可以访问组件实例 `this`
    console.log('beforeRouteUpdate 被调用：在当前路由改变，但是该组件被复用时调用')
    next()
},
beforeRouteLeave (to, from, next) {
    // 导航离开该组件的对应路由时调用
    // 可以访问组件实例 `this`
    const answer = window.confirm('是否确认离开当前页面')
    if (answer) {
        console.log('beforeRouteLeave 被调用：导航离开该组件的对应路由时调用')
        next()
    } else {
        next(false)
    }
},
```

40. vue-router 有哪几种导航钩子?

- 全局钩子
- 单个路由独享的
- 由组件级的

41. 完整的 vue-router 导航解析流程?

- 1.导航被触发。
- 2.在失活的组件里调用离开守卫。
- 3.调用全局的 `beforeEach` 守卫。
- 4.在重用的组件里调用 `beforeRouteUpdate` 守卫（2.2+）。
- 5.在路由配置里调用 `beforeEnter`。
- 6.解析异步路由组件。
- 7.在被激活的组件里面调用 `beforeRouterEnter`。
- 8.调用全局的 `beforeResolve` 守卫（2.5+）。
- 9.导航被确认。
- 10.调用全局的 `afterEach` 钩子。
- 11.触发 `DOM` 更新。
- 12.用创建好的实例调用 beforeRouteEnter 守卫中传给 next 的回调函数。

42. vue-router 实现路由懒加载（动态加载路由）?

- 也叫延迟加载，即在需要的时候进行加载，随用随载。
- 如何实现：
  - 第一种写法：使用 `AMD` 风格的 `require`，于是就更简单了

```
// 引入
const Foo = resolve => require(['./Foo.vue'], resolve)
// 使用
const routers = [
    {
        path: '/',
        name: 'index',
        component: (resolve) => require(['./views/index.vue'], resolve)
    }
]
```

  - 第二种写法：(使用 `import`)

```
const Index = () => import(/* webpackChunkName: "group-home" */  '@/views/index')
const routers = [
    {
        path: '/',
        name: 'index',
        component: Index
    }
]
```

- 第三种写法：使用 webpack 特有的 require.ensure()。注：require.ensure 是 webpack 的特殊语法，用来设置 code-split point

43. 指令 V-EI 的作用是什么?

- 提供一个在页面上已存在的 DOM 元素作为 Vue 实例的挂载目标.可以是 CSS 选择器，也可以是一个 HTMLElement 实例。

44. 在 vue 中使用插件的步骤?

- 提采用 ES6 的 import ... from ...语法或 CommonJS 的 require()方法引入插件使用全局方法 Vue.use(plugin)使用插件,可以传入一个选项对象 Vue.use(MyPlugin, { someOption: true })

45. 说说 ajax、fetch、axios 之间的区别?

- Ajax 请求

```
$.ajax({
    type: 'POST',
    url: url,
    data: data,
    dataType: dataType,
    success: function () { },
    error: function () { }
});
```

  - 本身是针对 MVC 的编程,不符合现在前端 MVVM 的浪潮
  - 基于原生的 XHR 开发， XHR 本身的架构不清晰，已经有了 fetch 的替代方案
  - JQuery 整个项目太大，单纯使用 ajax 却要引入整个 JQuery 非常的不合理（采取个性化打包的方案又不能享受 CDN 服务）

- fetch 请求

  - fetcht 只对网络请求报错，对 400 ， 500 都当做成功的请求，需要封装去处理
  - fetch 默认不会带 cookie ，需要添加配置项
  - fetch 不支持 abort ，不支持超时控制，使用 setTimeout 及 Promise.reject 的实的超时控制并不能阻止请求过程继续在后台运行，造成了量的浪费
  - fetch 没有办法原生监测请求的进度，而 XHR 可以

```
try {
    js
    let response = await fetch(url);
    let data = response.json();
    console.log(data);
} catch (e) {
    console.log("Oops, error", e);
}

```

- axios 请求
  - 从浏览器中创建 XMLHttpRequest
  - 从 node.js 发出 http 请求
  - 支持 Promise API
  - 拦截请求和响应
  - 转换请求和响应数据
  - 取消请求
  - 自动转换 JSON 数据
  - 客户端支持防止 CSRF/XSRF

```

axios({
    method: 'post',
    url: '/user/12345',
    data: {
        firstName: 'Fred',
        lastName: 'Flintstone'
    }
}).then(function (response) {
    console.log(response);
}).catch(function (error) {

    console.log(error);

});
```

46. vuex 的 action 和 mutation 的特性是什么？有什么区别?

- 一些对 State 的异步操作可放在 Action 中，并通过在 Action 中 commit Mutation 变更状态
- Action 可通过 store.dispatch() 方法触发，或者通过 mapActions 辅助函数将 vue 组件的 methods 映射成 store.dispatch() 调用

Mutation

- 在 vuex 的严格模式下，Mutaion 是 vuex 中改变 State 的唯一途径
- Mutation 中只能是同步操作
- 通过 store.commit() 调用 Mutation

47. vue 中父组件触发子组件的函数，子组件触发父组件的函数?

- 父组件中获取子组件函数：`$children`、`$refs`
- 子组件中获取父组件函数：`$on`

48. 对于 vue 了解多少，什么是 vuex，VueRouter?

- `vuex` 为状态管理，它集中存储管理应用的所有组件的状态，可以理解成一个全局仓库
- `VueRouter` 是路由（spa）单页面应用的方式
- `VueRouter` 是路由（spa）单页面应用的方式 `VueRouter` 是路由（spa）单页面应用的方
- `vue2` 取消冒泡全部`$emit` 用 `vuex` 定义全局 `state` 组件修改 `state` 父组件访问 `state` 值

49. 第一次页面加载会触发哪几个钩子?

- `beforeCreate`， `created`， `beforeMount`， `mounted`

50. created 和 mounted 的区别?

- `created`: 在模板渲染成 html 前调用，即通常初始化某些属性值，然后再渲染成视图。
- `mounted`: 在模板渲染成 html 后调用，通常是初始化页面完成后，再对 html 的 dom 节点进行一些需要的操作。

51. $route 和$router 的区别?

- `$router` 是 VueRouter 的实例，在 script 标签中想要导航到不同的 URL,使用$router.push 方法。返回上一个历史 history 用$router.to(-1)
- `$route` 当前 `route` 跳转对象。里面可以获取当前路由的 name,path,query,parmas 等。

vue

1. 从 0 到 1 自己构架一个 vue 项目，说说有哪些步骤、哪些重要插件、目录结构你会怎么组织
2. 你知道 vue 的模板语法用的是哪个 web 模板引擎的吗？说说你对这模板引擎的理解
3. 你知道 v-model 的原理吗？说说看
4. 你有使用过 vue 开发多语言项目吗？说说你的做法？
5. 在使用计算属性的时，函数名和 data 数据源中的数据可以同名吗？
6. vue 中 data 的属性可以和 methods 中的方法同名吗？为什么？
7. 怎么给 vue 定义全局的方法？
8. vue2.0 不再支持 v-html 中使用过滤器了怎么办？
9. 怎么解决 vue 打包后静态资源图片失效的问题？
10. 怎么解决 vue 动态设置 img 的 src 不生效的问题？
11. 使用 vue 后怎么针对搜索引擎做 SEO 优化？
12. 跟 keep-alive 有关的生命周期是哪些？描述下这些生命周期
13. 如果现在让你从 vue/react/angularjs 三个中选择一个，你会选哪个？说说你的理由
14. 你知道 vue2.0 兼容 IE 哪个版本以上吗？
15. 使用 vue 开发一个 todo 小应用，谈下你的思路
16. 你有看过 vue 推荐的风格指南吗？列举出你知道的几条
17. 你是从 vue 哪个版本开始用的？你知道 1.x 和 2.x 有什么区别吗？
18. 你知道 vue 中 key 的原理吗？说说你对它的理解
19. vue 中怎么重置 data？
20. vue 渲染模板时怎么保留模板中的 HTML 注释呢？
21. Vue.observable 你有了解过吗？说说看
22. 你知道 style 加 scoped 属性的用途和原理吗？
23. 你期待 vue3.0 有什么功能或者改进的地方？
24. vue 边界情况有哪些？
25. 如何在子组件中访问父组件的实例？
26. watch 的属性用箭头函数定义结果会怎么样？
27. 在 vue 项目中如果 methods 的方法用箭头函数定义结果会怎么样？
28. 在 vue 项目中如何配置 favicon？
29. 你有使用过 babel-polyfill 模块吗？主要是用来做什么的？
30. 说说你对 vue 的错误处理的了解？
31. 在 vue 事件中传入$event，使用 e.target 和 e.currentTarget 有什么区别？
32. 在.vue 文件中 style 是必须的吗？那 script 是必须的吗？为什么？
33. vue 怎么实现强制刷新组件？
34. vue 自定义事件中父组件怎么接收子组件的多个参数？

35. 实际工作中，你总结的 vue 最佳实践有哪些？
36. vue 给组件绑定自定义事件无效怎么解决？
37. vue 的属性名称与 method 的方法名称一样时会发生什么问题？
38. vue 变量名如果以_、$开头的属性会发生什么问题？怎么访问到它们的值？
39. vue 使用 v-for 遍历对象时，是按什么顺序遍历的？如何保证顺序？
40. vue 如果想扩展某个现有的组件时，怎么做呢？
41. 说下$attrs 和$listeners 的使用场景
42. 分析下 vue 项目本地开发完成后部署到服务器后报 404 是什么原因呢？
43. v-once 的使用场景有哪些？
44. 说说你对 vue 的表单修饰符.lazy 的理解
45. vue 为什么要求组件模板只能有一个根元素？
46. EventBus 注册在全局上时，路由切换时会重复触发事件，如何解决呢？
47. 怎么修改 vue 打包后生成文件路径？
48. 你有使用做过 vue 与原生 app 交互吗？说说 vue 与 ap 交互的方法
49. 使用 vue 写一个 tab 切换
50. vue 中什么是递归组件？举个例子说明下？
51. 怎么访问到子组件的实例或者子元素？
52. 在子组件中怎么访问到父组件的实例？
53. 在组件中怎么访问到根实例？
54. 说说你对 Object.defineProperty 的理解
55. vue 组件里写的原生 addEventListeners 监听事件，要手动去销毁吗？为什么？
56. vue 组件里的定时器要怎么销毁？
57. vue 组件会在什么时候下被销毁？
58. 使用 vue 渲染大量数据时应该怎么优化？说下你的思路！
59. 在 vue 中使用 this 应该注意哪些问题？
60. 你有使用过 JSX 吗？说说你对 JSX 的理解
61. 说说组件的命名规范
62. 怎么配置使 vue2.0+支持 TypeScript 写法？
63. `<template></template>`有什么用？
64. vue 的 is 这个特性你有用过吗？主要用在哪些方面？
65. vue 的:class 和:style 有几种表示方式？
66. 你了解什么是函数式组件吗？
67. vue 怎么改变插入模板的分隔符？
68. 组件中写 name 选项有什么作用？
69. 说说你对 provide 和 inject 的理解
70. 开发过程中有使用过 devtools 吗？
71. 说说你对 slot 的理解有多少？slot 使用场景有哪些？

72. 你有使用过动态组件吗？说说你对它的理解
73. prop 验证的 type 类型有哪几种？
74. prop 是怎么做验证的？可以设置默认值吗？
75. 怎么缓存当前打开的路由组件，缓存后想更新当前组件怎么办呢？
76. 说说你对 vue 组件的设计原则的理解
77. 你了解 vue 的 diff 算法吗？
78. vue 如何优化首页的加载速度？
79. vue 打包成最终的文件有哪些？
80. ajax、fetch、axios 这三都有什么区别？
81. vue 能监听到数组变化的方法有哪些？为什么这些方法能监听到呢？
82. vue 中是如何使用 event 对象的？
83. vue 首页白屏是什么问题引起的？如何解决呢？
84. 说说你对单向数据流和双向数据流的理解
85. 移动端 ui 你用的是哪个 ui 库？有遇到过什么问题吗？
86. 你知道 nextTick 的原理吗？
87. 说说你对 v-clock 和 v-pre 指令的理解
88. 写出你知道的表单修饰符和事件修饰符
89. 说说你对 proxy 的理解
90. 你有自己用 vue 写过 UI 组件库吗？
91. 用 vue 怎么实现一个换肤的功能？
92. 有在 vue 中使用过 echarts 吗？踩过哪些坑？如何解决的？
93. 如果让你教一个 2-3 年经验前端经验的同事使用 vue，你该怎么教？
94. vue 性能的优化的方法有哪些？
95. SSR 解决了什么问题？有做过 SSR 吗？你是怎么做的？
96. 说说你觉得认为的 vue 开发规范有哪些？
97. vue 部署上线前需要做哪些准备工作？
98. vue 过渡动画实现的方式有哪些？
99. vue 在 created 和 mounted 这两个生命周期中请求数据有什么区别呢？
100. vue 父子组件双向绑定的方法有哪些？
101. vue 怎么获取 DOM 节点？
102. vue 项目有做过单元测试吗？
103. vue 项目有使用过 npm run build --report 吗？
104. 如何解决 vue 打包 vendor 过大的问题？
105. webpack 打包 vue 速度太慢怎么办？
106. vue 在开发过程中要同时跟 N 个不同的后端人员联调接口（请求的 url 不一样）时你该怎么办？
107. vue 要做权限管理该怎么做？如果控制到按钮级别的权限怎么做？

108. 说下你的 vue 项目的目录结构，如果是大型项目你该怎么划分结构和划分组件呢？
109. 在移动端使用 vue，你觉得最佳实践有哪些？
110. 你们项目为什么会选 vue 而不选择其它的框架呢？
111. 对于即将到来的 vue3.0 特性你有什么了解的吗？
112. vue 开发过程中你有使用什么辅助工具吗？
113. vue 和微信小程序写法上有什么区别？
114. 怎么缓存当前的组件？缓存后怎么更新？
115. 你了解什么是高阶组件吗？可否举个例子说明下？
116. 为什么我们写组件的时候可以写在.vue 里呢？可以是别的文件名后缀吗？
117. vue-loader 是什么？它有什么作用？
118. 说说你对 vue 的 extend（构造器）的理解，它主要是用来做什么的？
119. 如果将 axios 异步请求同步化处理？
120. 怎么捕获组件 vue 的错误信息？
121. 为什么 vue 使用异步更新组件？
122. 如何实现一个虚拟 DOM？说说你的思路
123. 写出多种定义组件模板的方法
124. SPA 单页面的实现方式有哪些？
125. 说说你对 SPA 单页面的理解，它的优缺点分别是什么？
126. 说说你都用 vue 做过哪些类型的项目？
127. 在 vue 项目中如何引入第三方库（比如 jQuery）？有哪些方法可以做到？
128. 使用 vue 手写一个过滤器
129. 你有使用过 render 函数吗？有什么好处？
130. 写出你常用的指令有哪些？
131. 手写一个自定义指令及写出如何调用
132. 组件进来请求接口时你是放在哪个生命周期？为什么？
133. 你有用过事件总线(EventBus)吗？说说你的理解
134. 说说 vue 的优缺点分别是什么？
135. DOM 渲染在哪个周期中就已经完成了？
136. 第一次加载页面时会触发哪几个钩子？
137. vue 生命周期总共有几个阶段？
138. vue 生命周期的作用是什么？
139. vue 和 angular 有什么区别呢？
140. 如何引入 scss？引入后如何使用？
141. 使用 vue 开发过程你是怎么做接口管理的？
142. 为何官方推荐使用 axios 而不用 vue-resource？
143. 你了解 axios 的原理吗？有看过它的源码吗？
144. 你有封装过 axios 吗？主要是封装哪方面的？

145. 如何中断 axios 的请求？
146. axios 是什么？怎样使用它？怎么解决跨域的问题？
147. 说说你对 vue 的 template 编译的理解？
148. v-on 可以绑定多个方法吗？
149. vue 常用的修饰符有哪些？列举并说明
150. 你认为 vue 的核心是什么？
151. v-model 是什么？有什么用呢？
152. 说说你对 vue 的 mixin 的理解，有什么应用场景？
153. SPA 首屏加载速度慢的怎么解决？
154. 删除数组用 delete 和 Vue.delete 有什么区别？
155. 动态给 vue 的 data 添加一个新的属性时会发生什么？怎样解决？
156. 组件和插件有什么区别？
157. 说说你使用 vue 过程中遇到的问题（坑）有哪些，你是怎么解决的？
158. 说说你对选项 el,template,render 的理解
159. vue 实例挂载的过程是什么？
160. vue 在组件中引入插件的方法有哪些？
161. v-if 和 v-for 的优先级是什么？如果这两个同时出现时，那应该怎么优化才能得到更好的性能？
162. 分别说说 vue 能监听到数组或对象变化的场景，还有哪些场景是监听不到的？无法监听时有什么解决方案？
163. $nextTick 有什么作用？
164. 为什么 data 属性必须声明为返回一个初始数据对应的函数呢？
165. 怎么在 watch 监听开始之后立即被调用？
166. watch 怎么深度监听对象变化？
167. watch 和计算属性有什么区别？
168. vue 如何监听键盘事件？
169. v-for 循环中 key 有什么作用？
170. 怎么在 vue 中使用插件？
171. 你有写过自定义组件吗？
172. 说说你对 keep-alive 的理解是什么？
173. 怎么使 css 样式只在当前组件中生效？
174. 你有看过 vue 的源码吗？如果有那就说说看
175. 你有写过自定义指令吗？自定义指令的生命周期（钩子函数）有哪些？
176. v-show 和 v-if 有什么区别？使用场景分别是什么？
177. 说说你对 MVC、MVP、MVVM 模式的理解
178. 说下你对指令的理解？
179. 请描述下 vue 的生命周期是什么？
180. vue 组件之间的通信都有哪些？

181. 什么是虚拟 DOM？
182. 什么是双向绑定？原理是什么？
183. vue 和 react 有什么不同？使用场景是什么？
184. 说说 vue 的优缺点
185. 有使用过 vue 吗？说说你对 vue 的理解

186. vue-cli

187. vue-cli 提供了的哪几种脚手架模板？
188. vue-cli 工程中常用的 npm 命令有哪些？
189. 在使用 vue-cli 开发 vue 项目时，自动刷新页面的原理你了解吗？
190. vue-cli3 插件有写过吗？怎么写一个代码生成插件？
191. vue-cli 生成的项目可以使用 es6、es7 的语法吗？为什么？
192. vue-cli 怎么解决跨域的问题？
193. vue-cli 中你经常的加载器有哪些？
194. 你知道什么是脚手架吗？
195. 说下你了解的 vue-cli 原理？你可以自己实现个类 vue-cli 吗？
196. 怎么使用 vue-cli3 创建一个项目？
197. vue-cli3 你有使用过吗？它和 2.x 版本有什么区别？
198. vue-cli 默认是单页面的，那要弄成多页面该怎么办呢？
199. 不用 vue-cli，你自己有搭建过 vue 的开发环境吗？流程是什么？

200. vue-router

201. vue-router 怎么重定向页面？
202. vue-router 怎么配置 404 页面？
203. 切换路由时，需要保存草稿的功能，怎么实现呢？
204. vue-router 路由有几种模式？说说它们的区别？
205. vue-router 有哪几种导航钩子（ 导航守卫 ）？
206. 说说你对 router-link 的了解
207. vue-router 如何响应路由参数的变化？
208. 你有看过 vue-router 的源码吗？说说看
209. 切换到新路由时，页面要滚动到顶部或保持原先的滚动位置怎么做呢？
210. 在什么场景下会用到嵌套路由？
211. 如何获取路由传过来的参数？
212. 说说 active-class 是哪个组件的属性？
213. 在 vue 组件中怎么获取到当前的路由信息？
214. vur-router 怎么重定向？
215. 怎样动态加载路由？

216. 怎么实现路由懒加载呢？
217. 如果让你从零开始写一个 vue 路由，说说你的思路
218. 说说 vue-router 完整的导航解析流程是什么？
219. 路由之间是怎么跳转的？有哪些方式？
220. 如果 vue-router 使用 history 模式，部署时要注意什么？
221. route 和 router 有什么区别？
222. vue-router 钩子函数有哪些？都有哪些参数？
223. vue-router 是用来做什么的？它有哪些组件？

vuex

1. 你有写过 vuex 中 store 的插件吗？
2. 你有使用过 vuex 的 module 吗？主要是在什么场景下使用？
3. vuex 中 actions 和 mutations 有什么区别？
4. vuex 使用 actions 时不支持多参数传递怎么办？
5. 你觉得 vuex 有什么缺点？
6. 你觉得要是不用 vuex 的话会带来哪些问题？
7. vuex 怎么知道 state 是通过 mutation 修改还是外部直接修改的？
8. 请求数据是写在组件的 methods 中还是在 vuex 的 action 中？
9. 怎么监听 vuex 数据的变化？
10. vuex 的 action 和 mutation 的特性是什么？有什么区别？
11. 页面刷新后 vuex 的 state 数据丢失怎么解决？
12. vuex 的 state、getter、mutation、action、module 特性分别是什么？
13. vuex 的 store 有几个属性值？分别讲讲它们的作用是什么？
14. 你理解的 vuex 是什么呢？哪些场景会用到？不用会有问题吗？有哪些特性？
15. 使用 vuex 的优势是什么？
16. 有用过 vuex 吗？它主要解决的是什么问题？推荐在哪些场景用？

ElementUI

1. ElementUI 是怎么做表单验证的？在循环里对每个 input 验证怎么做呢？
2. 你有二次封装过 ElementUI 组件吗？
3. ElementUI 怎么修改组件的默认样式？
4. ElementUI 的穿梭组件如果数据量大会变卡怎么解决不卡的问题呢？
5. ElementUI 表格组件如何实现动态表头？
6. ElementUI 使用表格组件时有遇到过问题吗？
7. 有阅读过 ElementUI 的源码吗？

8. 项目中有使用过 ElementUI 吗？有遇到过哪些问题？它的使用场景主要是哪些？
9. 有用过哪些 vue 的 ui？说说它们的优缺点？

mint-ui

1. mint-ui 使用过程中有没有遇到什么坑？怎么解决的？
2. 说出几个 mint-ui 常用的组件
3. mint-ui 是什么？你有使用过吗？