MVVM分为Model、View、ViewModel三者。
Model 代表数据模型，数据和业务逻辑都在Model层中定义；
View 代表UI视图，负责数据的展示；
ViewModel 负责监听 Model 中数据的改变并且控制视图的更新，处理用户交互操作；
Model 和 View 并无直接关联，而是通过 ViewModel 来进行联系的，Model 和 ViewModel 之间有着双向数据绑定的联系。因此当 Model 中的数据改变时会触发 View 层的刷新，View 中由于用户交互操作而改变的数据也会在 Model 中同步。
这种模式实现了 Model 和 View 的数据自动同步，因此开发者只需要专注对数据的维护操作即可，而不需要自己操作 dom。

vue

谈谈你对MVVM开发模式的理解

Vue 有哪些指令

- 指令：v-if v-else v-for v-text v-show v-htm v-once v-bind v-model v-on
- 注意：当v-for和v-if同处于一个节点时，v-for的优先级比v-if更高。这意味着v-if将运行在每个v-for循环中

v-model修饰符

- .lazy
  默认情况下，v-model同步输入框的值和数据。可以通过这个修饰符，转变为在change事件再同步。
- .number
  自动将用户的输入值转化为数值类型
- .trim
  自动过滤用户输入的首尾空格

v-on修饰符

- .stop 阻止事件继续传播
- .prevent 事件不再重载页面
- .capture 使用事件捕获模式,即元素自身触发的事件先在此处处理，然后才交由内部元素进行处理
- .self 只当在 event.target 是当前元素自身时触发处理函数
- .once 事件将只会触发一次
- .passive 告诉浏览器你不想阻止事件的默认行为

props属性默认值

```
 props: {
// 基础的类型检查 (`null` 匹配任何类型)
propA: Number,
// 多个可能的类型
propB: [String, Number],
// 必填的字符串
propC: {
type: String,
required: true
},
// 带有默认值的数字
propD: {
type: Number,
default: 100
},
// 带有默认值的对象
propE: {
type: Object,
// 对象或数组且一定会从一个工厂函数返回默认值
default: function () {
return { message: 'hello' }
}
},
// 自定义验证函数
propF: {
validator: function (value) {
// 这个值必须匹配下列字符串中的一个
return ['success', 'warning', 'danger'].indexOf(value) !== -1
}
}
}
```

vuex

- 作用：vuex是基于vue框架的一个状态管理库。可以管理复杂应用的数据状态，比如兄弟组件的通信、多层嵌套的组件的传值等等。
- vuex核心模块

state => 基本数据
getters => 从基本数据派生的数据，相当于state的计算属性
mutations => 提交更改数据的方法，同步，与this.$store.state.变量名 = XXX的区别是：
1) 共同点： 能够修改state里的变量，并且是响应式的 (能触发视图更新)
2) 不同点：若将vue创建 store 的时候传入 strict: true, 开启严格模式，那么任何修改state的操作，只要不经过 mutation的函数就会报错

actions => 像一个装饰器，包裹mutations，使之可以异步。
包裹mutations，使之可以异步
action的功能和mutation是类似的，都是去变更store里的state，不过action和
mutation有两点不同：
1. action主要处理的是异步的操作，mutation必须同步执行，而action就不受这样的限制，也就是说action中我们既可以处理同步，也可以处理异步的操作
2. action改变状态，最后是通过提交mutation
注意：Action 函数接受一个与 store 实例具有相同方法和属性的 context 对象，因此你可以调用 context.commit 提交一个 mutation，或者通过 context.state 和 context.getters 来获取 state 和 getters。Action 通过 store.dispatch 方法触发

modules => 模块化Vuex
1) 背景：在Vue中State使用是单一状态树结构，应该的所有的状态都放在state里面，如果项目比较复杂，那state是一个很大的对象，store对象也将对变得非常大，难于管理。
2) module：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

生命周期

- beforeCreate：在实例初始化之前调用，数据观察和event/watch事件配置之前被调用
- created：在实例创建完成之后立即调用，在这一步实例已完成 数据观测 (data observe) 属性和算法的运算watch event事件回调，挂载阶段还没开始，$el 属性目前尚不可用
- beforeMount：在挂载之前被调用：相关的render函数首次被调用.
- mounted：在挂载之后被调用：注意该钩子在服务器渲染期间不会被调用：mounted不会保证所有的子组件也都一起被挂载。如果希望等到整个视图被渲染完毕后，可以在mounted内部使用vm.$nextTick;
- beforeUpdate：数据更新时调用，发生在虚拟 DOM 打补丁之前。这里适合在更新之前访问现有的 DOM，比如手动移除已添加的事件监听器数据更新时调用，发生在虚拟 DOM 打补丁之前。这里适合在更新之前访问现有的 DOM，比如手动移除已添加的事件监听器。
- update：数据更改导致的虚拟 DOM 重新渲染和打补丁后会调用该钩子。
- beforeDestory：解除绑定，销毁子组件以及事件监听
- destory：销毁完毕
- activated：被 keep-alive 缓存的组件激活时调用。

原理

当一个Vue实例创建时，vue会遍历data选项的属性，用 Object.defineProperty 将它们转为 getter/setter并且在内部追踪相关依赖，在属性被访问和修改时通知变化。

每个组件实例都有相应的 watcher 程序实例，它会在组件渲染的过程中把属性记录为依赖，之后当依赖项的 setter 被调用时，会通知 watcher 重新计算，从而致使它关联的组件得以更新。

v-if v-show的区别

- 用于根据条件展示元素。和v-if不同的是，如果v-if的值是false，则这个元素被销毁，不在dom中。但是v-show的元素会始终被渲染并保存在dom中，它只是简单的切换css的dispaly属性。
- v-if有更高的切换开销
  v-show有更高的初始渲染开销。
  因此，如果要非常频繁的切换，则使用v-show较好；如果在运行时条件不太可能改变，则v-if较好

如何在组件内部实现一个双向数据绑定

父组件通过 props 传值给子组件，子组件通过 $emit 来通知父组件修改相应的props值

vue父子组件传值方式有哪些

- 父传子
  - 子组件在props中创建一个属性，用以接收父组件传过来的值
  - 父组件中注册子组件
  - 在子组件标签中添加子组件props中创建的属性
  - 把需要传给子组件的值赋给该属性
- 子传父
  - 子组件 (备注：在响应该点击事件的函数中使用$emit来触发一个自定义事件，并传递一个参数)
  - 父组件 (备注：在父组件中的子标签中监听该自定义事件并添加一个响应该事件的处理方法)
- 兄弟传值：数据输入方使用$emit()，收据接受方.$on()

computed和watch的区别

- 计算属性computed：
  1. 支持缓存，只有依赖数据发生改变，才会重新进行计算
  2. 不支持异步，当computed内有异步操作时无效，无法监听数据的变化
  3.computed 属性值会默认走缓存，计算属性是基于它们的响应式依赖进行缓存的，也就是基于data中声明过或者父组件传递的props中的数据通过计算得到的值
  4. 如果一个属性是由其他属性计算而来的，这个属性依赖其他属性，是一个多对一或者一对一，一般用computed
  5.如果computed属性属性值是函数，那么默认会走get方法；函数的返回值就是属性的属性值；在computed中的，属性都有一个get和一个set方法，当数据变化时，调用set方法。
- watch：
  1. 不支持缓存，数据变，直接会触发相应的操作；
  2.watch支持异步；
  3.监听的函数接收两个参数，第一个参数是最新的值；第二个参数是输入之前的值；
  4. 当一个属性发生变化时，需要执行对应的操作；一对多；
  5. 监听数据必须是data中声明过或者父组件传递过来的props中的数据，当数据变化时，触发其他操作，函数有两个参数，
     immediate：组件加载立即触发回调函数执行，
     deep: 深度监听，为了发现对象内部值的变化，复杂类型的数据时使用，例如数组中的对象内容的改变，注意监听数组的变动不需要这么做。注意：deep无法监听到数组的变动和对象的新增，参考vue数组变异,只有以响应式的方式触发才会被监听到。

webpack

简介

- Webpack是前端资源打包工具，从入口文件出发，依次加载并解析依赖的 js、css 、图片等资源模块，在内部生成一个依赖图，最终根据不同优化规则，打包生成一个或多个bundle。
- 使用webpack管理模块依赖，并编译输出她们所需要的静态文件，它能够很好地管理、打包web开发中所用到的html、css、js及各种静态文件，让开发过程更加高效。

Webpack实现原理

把所有依赖打包成一个 bundle.js 文件，通过代码分割成单元片段并按需加载。

插件

- CommonsChunkPlugin：CommonsChunkPlugin插件用于提取入口文件引用的公共模块，并打包到一个独立的文件。
- HtmlWebpackPlugin：这个模块是为了简化html文件的创建，特别是当html引用的js或css文件跟webpack编译结果有关时。输出的html就会在body的最后添加script标签，来引用所有的入口js文件的编译chunk：
- 作用：用于扩展webpack的功能。不同于loader，plugin的功能更加丰富，比如压缩打包，优化，不只局限于资源的加载。

loader加载器

- 定义：loader用于加载某些资源文件。因为webpack本身只能打包common.js规范的js文件，对于其他资源如css，img等，是没有办法加载的，这时就需要对应的loader将资源转化，从而进行加载。
- babel-loader cssloader style-loader lessl-loader url-loader
- loader的作用
  1、实现对不同格式的文件的处理，比如说将scss转换为css，或者typescript转化为js
  2、转换这些文件，从而使其能够被添加到依赖图中

webpack的优势

- (2) 能被模块化的不仅仅是 JS 了。
- (3) 开发便捷，能替代部分 grunt/gulp 的工作，比如打包、压缩混淆、图片转base64等。
- (4) 扩展性强，插件机制完善
- Set数据结构，类似数组。所有的数据都是唯一的，没有重复的值。它本身是一个构造函数
- loader加载器：babel-loader cssloader style-loader lessl-loader url-loader

loader和插件的区别

loaders负责的是处理源文件的如css、jsx，一次处理一个文件。而plugins并不是直接操作单个文件，它直接对整个构建过程起作用

核心模块

- entry：入口模块
- output：出口
- loader:加载器
- plugins:插件