var this = Object

ElementUI,iviewUI,antdesign,mitiUI,

**你对MVVM的理解是什么?**

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。

我们可以不关心DOM的结构，而是关心数据如何存储。

我们把数据看作Model，把DOM节点看作View，并假定它们之间通过ViewModel关联起来了。我们并不操作DOM，而是直接修改JavaScript对象,通过改变JavaScript对象的状态，会导致DOM结构作出对应的变化, 这就是MVVM的设计思想：关注Model的变化，让MVVM框架去自动更新DOM的状态，从而把开发者从操作DOM的繁琐步骤中解脱出来.

**数据双向绑定实现的原理(Vue的响应式原理)?**

vue数据双向绑定是通过数据劫持结合发布者-订阅者模式的方式来实现的.

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。

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（用来解析双大括号{{}}），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。

**如何理解Vue的渐进式框架?**

使用vue，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；

还可以用它的视图，搭配你自己设计的整个下层用它只是个轻量视图而已，只做了自己该做的事，没有做不该做的事，仅此而已。你不必一开始就用Vue所有的全家桶，根据场景，官方提供了方便的框架供你使用。比如:

你只想用Vue的声明式渲染,组件系统和数据双向绑定来管理你整个页面的DOM;当你觉得不够的时候,可以继续引入Vue客户端路由控制路由,引入axios来对接后台数据,或者继续引入vuex来实现大规模状态管理.

根据场景，官方都提供了方便的框架供我们使用.

**你是怎么使用Vue的?**

一般在我之前的项目中,都是使用vue全家桶来开发项目,通过vuecli脚手架工具搭建 vue（整体架构）;

使用vue-router路由模块来管理页面路由调转;

使用axios（ajax请求）发送请求,对接后台数据,实现数据拦截.

使用vuex进行大型数据状态管理.

使用mint-UI(移动端UI框架库) 或antd-vue(PC端UI框架库)来构建UI组件.

**Vue的特性有哪些?(简单介绍一下Vue)**

(1)数据双向绑定

vue.js最大的优点，通过MVVM思想实现数据的双向绑定，让开发者不用再操作dom对象，有更多的时间去思考业务逻辑。

(2) 组件化开发

提高开发效率,方便重复使用.

(3) Virtual DOM

虚拟DOM属于预处理操作，并没有真实的操作DOM，所以叫做虚拟DOM。最后在计算完毕才真正将DOM操作提交，将DOM操作变化反映到DOM树上。

(4) 动画系统

**Vue 有哪些指令?指令有哪些好处?封装过自己的指令吗?**

1. v-text: 更新元素文本

2. v-html: 等同于JS的innerHtml属性。

3. v-pre: 已知页面内部有大段内容无需编译，使用v-pre指令阻止编译可以提高性能.

4. v-cloak:这防止页面加载时出现 vuejs 的变量名而设计的.

5. v-once: Vue性能优化:vue-once关联的实例，只会渲染一次。之后的重新渲染，所有的子节点将被视为静态内容跳过，这可以用于优化更新性能。

6. v-if:实现条件渲染，Vue会根据表达式的值的真假条件来渲染元素。

7. v-else:v-else是搭配v-if使用的，它必须紧跟在v-if或者v-else-if后面，否则不起作用。

8. v-show: 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为none。

Ps: 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.

9. v-for: 用v-for指令根据遍历数组来进行渲染.

10. v-bind:属性绑定, 常用于动态绑定class和style.

11. v-model:用于表单元素的双向绑定, v-model会忽略所有表单元素的value、checked、selected特性的初始值。因为它选择Vue实例数据做为具体的值。相当于react的受控表单.

Ps: v-model修饰符:

(1): .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。

(2): .number自动将用户的输入值转化为数值类型.

(3) .trim自动过滤用户输入的首尾空格

12. v-on:主要用来监听dom事件.

Ps: v-on修饰符:

(1) .stop 阻止事件继续传播

(2) .prevent 事件不再重载页面

(3) .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

(4) .self 只当在 event.target 是当前元素自身时触发处理函数

(5) .once 事件将只会触发一次

(6) .passive 告诉浏览器你不想阻止事件的默认行为

自定义指令好就好在：原先的那些通用组件，无论是纯js的也好，基于jQuery的也好，都可以拿来主义直接吸收，而不需要改造或重构。

第三方插件如何与Vue.js集成的问题，都可以尝试用自定义指令实现。

我写文档通常会用到highlight.js，我们可以直接将其封装为一个自定义指令，这样highlight.js就变成了Vue的一个新功能。

我用自定义指令实现过一个常见的 tip 提示弹框的功能，鼠标移入元素，可以在元素的右方显示 tip，鼠标移出则隐藏 tip。

文本框限制只能输入指定位数的小数，写正则每次去匹配很麻烦，所以我写了一个自定义指令,使用起来更方便.

**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。

**vuex是什么？怎么使用？哪种功能场景使用它？**

vuex其实是集中的数据管理仓库，任何组件都可以存取仓库中的数据。vuex中有四个部分：state 、 mutations 、 actions 、getters:

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computed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里面处理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。

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。

**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) 。

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 。

**$route和$router的区别?**

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

**vue.js的两个核心是什么？**

答：数据驱动、组件系统

**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.

**v-on 可以绑定多个方法吗？**

答：可以,对象,事件作为键,执行函数作为值.例如:

v-on='{click:DoSomething,mouseleave:MouseLeave}'

**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。

**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视图使用数据时调，直接写计算的属性名即可；⑤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。

**vue等单页面应用及其优缺点**

答：优点：Vue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。

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(首屏加载耗时).

**怎么实现首屏加载速度优化?**

将各模块拆分为组件粒度

将组件依赖的资源全部封装在组件内部进行调用.

组件化的拆分，确保模块之间不会互相影响和产生耦合之后，我们可以方面地调整加载策略。加载的策略是根据可见性来处理优先级问题。

优先加载首屏可见模块.

其余不可见模块懒加载，待可见或即将可见时加载

将公用的JS库通过script标签外部引入，减小app.bundel的大小，让浏览器并行下载资源文件，提高下载速度；

在配置 路由时，页面和组件使用懒加载的方式引入，进一步缩小 app.bundel 的体积，在调用某个组件时再加载对应的js文件；

加一个首屏 loading 图，提升用户体验；

**Vue实现前进刷新,后退不刷新.**

**例举Vue组件通信的几种方式.**