**Vue**

1.MVC和MVVM的区别？

MVC：MVC是Model-View-Controller的缩写，是一种软件设计典范。

Model：模型，在应用程序中，负责处理数据逻辑相关的操作，即负责数据的增删改查；

View：视图，在应用程序中，负责Model的显示；

Controller：控制器，在应用程序中，负责处理Model和View之间的交互，即控制Model在View中的显示，处理用户在视图上的交互带来的Model更新等。

MVVM：MVVM是Model-View-VIewModel的缩写，新增了VM，通过数据绑定和DOM事件监听完成了数据的双向绑定，使得Model和View之间实现了自动同步，即Model更新，不再需要手动操作View，而是改变Model后对应的View会自动更新。

整体来看，MVVM相比于MVC，不仅解决了业务和界面的耦合，还解决了数据频繁更新的问题，降低了M和V之间耦合。

但是，Vue在实现时并没有完全遵循MVVM规范，因为严格的MVVM规范要求View和Model不能直接通信，而Vue提供了$refs这个属性，使得Model可以通过$refs获取DOM来直接操作View。，违反了MVVM的规范。

2.为什么data是一个函数？

组件中的data写成一个函数并且将数据以返回值的形式定义，使得组件在实例化每个实例时，都会返回一个新的data，也就是为每个组件实例创建了私有的数据空间，每个组件实例维护各自的数据。反之，将data单纯的写成对象形式，就使得所有组件实例共用同一份数据。

3.Vue组件通讯有哪几种方式？

（1）props/$emit：父组件向子组件传递数据是通过props，子组件向父组件传递数据是通过$emit触发事件来完成；

（2）eventBus：通过创建一个空的Vue实例来管理项目中组件间的通讯，适用于各种层级的组件；

（3）Vuex：Vuex为Vue提供状态管理，当项目规模越来越大，需要管理的公用数据越来越多时，Vuex是不错的选择，它提供以下API供我们使用：

state：需要管理的数据；

mutations：用于更新state的唯一方式，Vuex中更新state中数据的方式只有通过提交mutation；

actions：用于管理异步操作，在action中更新state的方式只能是通过提交mutation，不能直接更改；

getters：相当于Vue中的计算属性，用于做复杂的计算和数据结构拼接使用；

modules：当项目中需要管理的状态越来越多时，store下index.js的内容会越来越多，将项目所有状态管理放入一个文件肯定是不符合项目管理规范的，所以此时将一些独立的功能拆分到对应的module下进行管理变得势在必行。

（4）$parent/$children：可以获取组件的父组件实例和子组件实例，因此可以直接操作父组件和子组件来完成通讯；

（5）$refs：可以获取注册在$refs下的组件实例，因此可以直接操作任何层级的组件来完成通讯；

（6）provide/inject：父组件通过provide提供数据，子组件通过inject来注入数据。（官方不推荐此方式，但是很多组件库很欢迎）

4.Vue的生命周期方法有哪些？一般在哪一步发请求？ （1）beforeCreate：实例初始化之后调用，在此期间，实例尚没有完成数据观测（data observe）、event和watcher的绑定，即此时的操作data、methods和watcher等是没有效果的；

（2）created：实例创建完成之后调用，此时已完成数据观测（data observe）、事件绑定、method、watcher和computed定义，但是真实的DOM还没有挂载，可以请求服务端数据来更新data，如果要操作DOM的话，需要使用$nextTick来完成；

（3）beforeMount：DOM挂载之前调用，此时完成了虚拟DOM的创建，即将挂载DOM，render方法将首次调用；

（4）mounted：DOM挂载完成之后调用，此时已完成真实DOM的挂载，并完成了数据双向绑定，可以获取并操作DOM；

（5）beforeUpdate：实例数据更新之后且引起真实DOM更新之前，此时修改实例数据不会引起不必要的重复渲染，此周期在重新构建虚拟DOM，并与旧的虚拟DOM进行diff，完成patch之前；

（6）updated：实例DOM已完成重新渲染，在此期间尽量不要做更新实例数据的操作，以避免引起不必要的循环更新；

（7）beforeDestroy：实例销毁之前，此时实例的data、method、watcher、computed和真实DOM依然存在且可以调用，可以在这时完成一些善后工作，比如清楚实例使用的定时器等；

（8）destroyed：实例已被销毁，实例的data、method、watcher、computed和真实DOM已不存在，所有子实例也会被移除；

（9）activated：keep-alive专属，组件被激活时触发；

（10）deactived：keep-alive专属，组件从激活状态转为未激活状态时触发；

请求可以在created、beforeMount和mounted周期发请求，因为此时data已经完成数据监测，并且此时修改data数据不会引起重复渲染，但是一般会把请求放在created生命周期，优点如下：

1）请求放在created中，能更快获取服务端数据，减少页面等待时间；

2）SSR不只支持beforeMount和mounted，有助于代码一致性。

5.v-if和v-show的区别？

v-if：在编译时会转化成三元表达式，不满足条件时不会渲染节点，适用于不频繁改变展示条件的场景；

v-show：在编译时会转化成指令，不满足条件时依然会渲染节点，只是将其不展示，即display: none，适用于频繁改变展示条件的场景;

扩展：display: none; visibility: hidden; 和opacity: 0; 的区别

（1）空间占用：

display: none; 不占用

visibility: hidden; 占用

opacity: 0; 占用

（2）子元素继承：

display: none; 子元素不继承，但是父元素都不展示了，子元素也不会展示；

visibility: hidden; 子元素继承，可以通过改变子元素的visibility: visible；来展示子元素；

opacity: 0; 子元素继承，但是通过改变子元素的opacity无法展示子元素；

（3）触发绑定事件：

display: none; 不会触发；

visibility: hidden; 不会触发；

opacity: 0; 触发；

（4）过渡函数transition：

display: none; 无效

visibility: hidden; 无效

opacity: 0; 有效

6.说说Vue内置指令？

（1）v-once：定义它的元素或组件只渲染一次，包括元素或组件的所有子节点，首次渲染后，不再随数据的变化重新渲染，将被视为静态内容；

（2）v-bind：定义动态属性；

（3）v-show：定义元素或组件的显示和隐藏，不满足显示条件时，元素或组件依然会渲染，只是通过display: none;隐藏，适用于频繁切换元素或组件的显示和隐藏条件的场景；

（4）v-if/v-else-if/v-else：定义元素或组件的显示和隐藏，不满足显示条件时，元素或组件不会渲染，适用于不频繁切换元素或组件的显示和隐藏条件的场景，可以配合template来使用，编译时会转化成三元表达式；

（5）v-html：定义元素或组件的innerHtml属性；

（6）v-text：定义元素或组件的textContent属性；

（7）v-for：适用于渲染列表元素或组件，不能和v-if配合使用，因为v-for的优先级高于v-if，这种需求尽量使用计算属性来处理，并且使用v-for是需要为每一项添加key属性，key属性要具备唯一性，尽量不要使用index；

（8）v-pre：跳过元素或组件的编译过程，包括元素或组件的所有子节点，利用它可以提升页面的编译速度；

（9）v-on：用于监听DOM事件；

（10）v-cloak：在关联实例编译完成前，v-cloak都会保持在元素上，为了解决初始化慢导致的页面闪动；

（11）v-model：input的value和change的语法糖。

7.怎样理解Vue的单向数据流？

数据总是从父组件传到子组件，子组件没有权利修改父组件传过来的数据，只能请求父组件对原始数据进行修改，这样防止子组件意外篡改父组件的数据，从而导致应用的数据流向难以理解。

8.computed和watch的区别以及适用场景？

（1）computed，计算属性，具备缓存特性，返回组件中多个属性或计算属性的计算结果，应用于模版中，只有当计算结果改变时才会返回，适用于组件中某个属性依赖于其他多个属性或计算属性的场景；

（2）watch，侦听属性，不具备缓存特性，当侦听的属性变化时，触发其回调函数，适用于监听某个属性变化并对其变化做相应处理的场景。

9.v-if和v-for为什么不建议一起使用？

因为在编译时v-for的优先级高于v-if，建议使用computed计算属性来完成。

10.Vue2.0响应式数据的原理？

整体思路是数据劫持结合观察者模式。 结合源码分析，通过使用Object.defineProperty对data中的属性进行劫持（只会劫持已存在的属性），而数组则是通过重写数组方法来实现数据劫持。当视图中使用对应属性时，每个属性都拥有属于自己的主题对象dep，dep用于存放依赖该属性的watcher（观察者），即依赖收集；当属性变化后会通过主题对象dep通知对应的watcher进行更新，即派发更新。

11.Vue如何监测数组的变化？

考虑性能原因，Vue在对数组进行监测时没有使用Object.defineProperty对数组的每一项进行劫持，而是对7种数组方法进行重写以达到监测数组变化的目的，所以在Vue中修改数组的索引和数组的长度是无法监测到的，需要通过调用7种数组方法才能触发数组对应的watcher进行更新。

push、pop、unshift、shift、splice、reverse、sort。

12.Vue3.0用过吗？了解多少？（需要加强）

Vue3.0没有实际用过，只是之前有关注过，只记得Vue3.0在实现响应式数据原理方面做了改进，用Proxy代替Object.defineProperty来实现数据劫持，能够更有效的解决数组和对象数据的监测问题。

13.Vue3.0和Vue2.0的响应式原理区别？

在响应式数据实现方面，Vue3.0用Proxy代替Object.defineProperty来实现数据劫持。因为Proxy可以直接监听数组和对象的变化，并且有多达13种拦截方法。

14.Vue的父子组件生命周期钩子函数执行顺序？

（1）加载渲染过程

父beforeCreate -> 父created -> 父beforeMount -> 子beforeCreate -> 子created -> 子beforeMount -> 子mounted -> 父mounted

（2）更新过程

子组件更新过程

父beforeUpdate -> 子beforeUpdate -> 子updated -> 父updated

父组件更新过程

父beforeUpdate -> 父updated

（3）销毁过程

父beforeDestroy -> 子beforeDestroy -> 子destroyed -> 父destroyed

（4）keep-alive

父actived -> 子actived -> 子deactived -> 父deactived

15.虚拟DOM是什么？有什么优缺点？

由于在浏览器中操作DOM是很昂贵的。频繁操作DOM，会产生一定的性能问题。这就是虚拟DOM产生的原因。

Vue2.0的虚拟DOM借鉴了开源库snabbdom的实现。虚拟DOM本质就是用一个原生的JS对象去描述一个DOM节点，是对真实DOM的一层抽象。

优点：

（1）保证性能下限：框架的虚拟DOM需要适配任何上层API可能产生的操作，它的一些DOM操作的实现必须是普适的，所以它的性能并不是最优的；但是比起粗暴的DOM操作性能要好很多，因此框架的虚拟DOM至少可以保证在不需要手动优化的情况下，依然可以提供还不错的性能，即保证性能的下限。

（2）无需手动操作DOM：我们不再需要手动去操作DOM，只需要写好VIewModel中的代码逻辑即可，框架会根据虚拟DOM和数据双向绑定，帮我们以可预期的方式更新视图，极大提高我们的开发效率。

（3）跨平台：虚拟DOM本质上是JS对象，而DOM与平台强相关，相比之下虚拟DOM可以进行更方便的跨平台操作，例如服务器渲染、weex开发等。

缺点：

（1）无法进行极致优化：虽然虚拟DOM外加合理的优化，足以应对绝大部分应用的性能需求，但在一些性能要求极高的应用中虚拟DOM无法进行针对性的极致优化。

（2）跨平台： 虚拟DOM 本质上是JS对象，而DOM与平台强相关，相比之下虚拟DOM可以进行更方便地跨平台操作，例如服务器渲染、weex开发等等。

16.v-model

v-model本质上是输入元素值属性和对应事件的语法糖。

在text和textarea标签中使用，使用value属性和input事件；

在radio和checkbox标签中使用，使用checked和change事件；

在select标签中使用，使用value属性和change事件。

 17.v-for为什么要加key？

如果不使用key，Vue会使用一种最大限度减少动态元素并且尽可能尝试就地修改/复用相同类型元素的算法。key是Vue中vnode的唯一标记，通过这个标记，使得diff操作更准确，更快速。

（1）更准确：因为带key就不是就地复用，在sameNode函数a.key === b.key对比中可以避免就地复用的情况。所以会更准确。

（2）更快速：利用key的唯一性生成map对象来获取对应节点，比遍历方式更快。

18.Vue事件绑定原理？

原生事件绑定使用的是addEventListener绑定给真实元素，组件事件绑定是通过Vue自定义的$on实现的。如果要在组件上使用原生事件，需要加.native修饰符，这样就相当于把组件当作普通html标签，然后加上原生事件。

$on和$emit是基于发布订阅模式设计的，维护一个事件中心，通过on将事件按名称存在事件中心，称之为订阅者，然后emit通过事件名称将事件发布出去，去执行事件中心里对应的事件回调。

19.vue-router路由钩子函数是什么？执行顺序是什么？（需要加强）

钩子函数种类有：全局守卫、路由守卫、组件守卫，守卫也就是钩子函数。

完整的导航解析流程：

（1）导航被触发；

（2）在失活的组件里调用beforeRouteLeave守卫；

（3）调用全局的beforeEach守卫；

（4）在重用的组件里调用beforeRouteUpdate守卫；

（5）在路由配置里调用beforeEnter；

（6）解析异步路由组件；

（7）在被激活的组件里调用beforeRouteEnter；

（8）调用全局的beforeResolve守卫；

（9）导航被确认；

（10）调用全局的afterEach钩子；

（11）触发DOM更新；

（12）调用beforeRouteEnter守卫传给next的回调函数，创建好的组件实例会作为回调函数的参数传入。

20.vue-router动态路由是什么？有什么问题？

在需要把某种模式匹配到的所有路由，映射到同一个组件时，可以尝试将路由的路径设置为动态路径，即/User/:id的形式，例如，项目中的所有用户的用户个人中心使用的是同一个组件，只是参数不一样，此时可以用动态路由匹配。

但是，动态路由存在组件复用导致路由参数失效的问题，我们可以通过watch监听路由对象参数的变化来在回调函数中重新请求数据来更新组件，也可以通过给组件加key的方式解决该问题。

21.谈一下对Vuex的个人理解？

Vuex是Vue的全局状态管理系统。用于多个组件的数据共享和数据缓存等。（无法持久化，内部核心原理是通过创造一个全局Vue实例）

主要包括以下几个模块：

state：定义了应用状态的数据结构，可以在这里设置默认的初始状态；

mutation：是唯一更改 store 中状态的方法，且必须是同步函数；

action：用于提交 mutation，而不是直接变更状态，可以包含任意异步操作；

getter：允许组件从 Store 中获取数据，mapGetters 辅助函数仅仅是将 store 中的 getter 映射到局部计算属性；

module：允许将单一的 Store 拆分为多个 store 且同时保存在单一的状态树中；

22.Vuex刷新页面数据丢失怎么解决？

需要做Vuex数据持久化，一般使用本地存储的方案来保存数据，将需要缓存的数据存储到sessionStorage或localStorage中，刷新页面后从本地存储中依然可以获取存储的数据。

第三方插件vuex-persist也可以来做数据持久化，我们不需要操作本地存储，插件自动帮我们把store中的状态放入本地存储。

23.Vuex为什么要分模块并且加命名空间？

模块：由于使用单一状态树，应用的所有状态会集中到一个比较大的对象中。当应用变得非常复杂时，store对象就有可能变得相当臃肿。为了解决这个问题，Vuex允许我们将store拆分成模块。每个模块拥有自己的state、mutation、action、getter，甚至是嵌套子模块。

命名空间：默认情况下，模块内部的action、mutation和getter是注册在全局命名空间的，这样使得多个模块能够对同一mutation或action做出响应。如果希望模块具有更高的封装度和复用性，可以添加namespaced: true的方式使其成为带命名空间的模块。当模块被注册后，它的所有getter、action和mutation都会自动根据模块注册的路径调整命名。

24.使用过Vue的SSR吗？说说SSR？

SSR是服务端渲染，本质上，将Vue在客户端把渲染成html的工作交给服务端完成，服务端将标签转化成html后返回给客户端。

优点：

（1）更好的SEO；

（2）首屏加载速度更快；

缺点：

（1）开发条件会受限制，服务端渲染只支持beforeCreate和created两个生命周期；

（2）需要运行在node.js环境，增加了服务器端的负载。

25.Vue中使用了哪些设计模式？

（1）工厂模式-传入参数即可创建实例

虚拟DOM根据参数的不同返回基础标签的VNode和组件VNode；

（2）单例模式-整个程序有且仅有一个实例

Vuex和vue-router的插件注册方法会判断系统是否存在实例，如果存在直接返回；

（3）发布/订阅模式

Vue事件机制；

（4）观察者模式

响应式数据原理；

（5）装饰模式：

@装饰器的用法

26.你都做过哪些Vue的性能优化？

（1）对象层级不要过深，否则性能会变差；

（2）不需要响应式的数据不要放到data中（可以用Object.freeze()冻结数据）；

（3）v-if和v-show区分使用场景；

（4）computed和watch区分使用场景；

（5）v-for遍历必须加key，key最好不用index索引，且避免同时使用v-if；

（6）防止内部泄漏，组件在销毁后把全局变量和事件销毁；

（7）图片懒加载；

（8）第三方插件按需引入；

（9）适当采用keep-alive缓存组件；

（10）防抖/截流区分场景使用；

（11）服务端渲染SSR或预渲染。