前端面试指南 — vue 篇面试题

Vue不会出现在笔试中，所以这些需要全部准备好，这一关过就已经七七八八了。不过vue多问的除了基本知识，比较多的是问题解决方案，需要一定的项目经验，对于新人，可以掌握基础点，如 果面试的是初级，那项目问题一般不会很多，可以准备几个到时候用，如果是中级，那就需要多找 项目遇到的难点，到时候可以吹水了，比如webpack等。

考点 1：vue 的理解

问题：什么是 MVVM?

MVVM 是Model-View-ViewModel 的缩写， Model 代表数据模型， 定义数据操作的业务逻辑，View 代表 UI 组件， 它负责将数据模型转化成 UI 展现出来， ViewModel 通过双向绑定把 View 和Model 进行同步交互，不需要手动操作 DOM 的一种设计思想。

问题：Vue 的优点？Vue的缺点?

优点：渐进式，组件化，轻量级，虚拟dom，响应式，单页面路由，数据与视图分开

缺点：单页面不利于seo，不支持IE8以下，首屏加载时间长

问题：Vue跟React的异同点

相同点：

1.都使用了虚拟dom

2.组件化开发

3.都是单向数据流(父子组件之间，不建议子修改父传下来的数据)

4.都支持服务端渲染

不同点：

1.React的JSX，Vue的template

2.数据变化，React手动(setState)，Vue自动(初始化已响应式处理，Object.defineProperty)

3.React单向绑定，Vue双向绑定

4.React的Redux，Vue的Vuex

问题：请说说双向绑定实现的原理?

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。

问题：请说说你对“渐进式框架”的理解?

所说的"渐进式"，其实就是 vue 的使用方式，同时也体现了 vue 的设计的理念，渐进式代表的含义是：主张最少。

我们在使用过程中，可以通过添加组件系统、客户端路由 vue-router、大规模状态管理 vuex 来构建一个完整的框架。

更重要的是，这些功能相互独立，你可以在核心功能的基础上任意选用其他的部件，不一定要全部整合在一起。

问题：Vue和JQuery的区别在哪？为什么放弃JQuery用Vue?

1.jQuery是直接操作DOM，Vue不直接操作DOM，Vue的数据与视图是分开的，Vue只需要操作数据即可
2.jQuery的操作DOM行为是频繁的，而Vue利用虚拟DOM的技术，大大提高了更新DOM时的性能
3.Vue中不倡导直接操作DOM，开发者只需要把大部分精力放在数据层面上
4.Vue集成的一些库，大大提高开发效率，比如Vuex，Router等

问题：请说说什么是虚拟 DOM?

Virtual Dom 是一个 JS 对象，通过对象的方式来表示 DOM 结构。将页面的状态抽象为 JS 对象的形式，配合不同的渲染工具，使跨平台渲染成为可能。

通过事务处理机制，将多次 DOM 修改的结果一次性的更新到页面上，从而有效的减少页面渲染的次数，减少修改 DOM 的重绘重排次数，提高渲染性能。

在代码渲染到页面之前，vue 会把代码转换成一个对象(虚拟 DOM)。以对象的形式来描述真实 dom 结构，最终渲染到页面。在每次数据发生变化前，虚拟 dom 都会缓存一份，变化之时，现在的虚拟 dom 会与缓存的虚拟 dom 进行比较。

在 vue 内部封装了 diff 算法，通过这个算法来进行比较，渲染时修改改变的变化，原先没有发生改变的通过原先的数据进行渲染。 另外现代前端框架的一个基本要求就是无需手动操作 DOM，一方面是因为手动操作 DOM 无法保证程序性能，省略手动 DOM 操作可以大大提升开发效率。

考点 2：生命周期

问题：说说 vue 生命周期的做用是什么?

vue 的生命周期中有多个钩子函数，让咱们在控制整个 vue 实例的过程当中，更容易造成良好的逻辑。

问题：请详细说下你对 vue 生命周期的理解，分别有哪些阶段？（会读最好，不会读用中文说）

总共分为 8 个阶段：创建前/后，载入前/后，更新前/后，销毁前/后。

1. 创建前

beforeCreated 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。

2. 创建后

created 阶段，vue 实例的数据对象 data 有了，$el 还没有。

3. 载入前

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。

4. 载入后

在 mounted 阶段，vue 实例挂载完成，data.message 成功渲染。

5. 更新前

当 data 变化时，会触发 beforeUpdate 方法。

6. 更新后

当 data 变化后，会触发 updated 方法。

7. 销毁前

beforeDestory 阶段，在执行 destroy 方法时会触发。

8. 销毁后

destoryed 阶段，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。

9. dctivated阶段，被 keep-alive 缓存的组件激活时调用。

10. deactivated阶段，被 keep-alive 缓存的组件失活时调用。

11. errorCaptured阶段，在捕获一个来自后代组件的错误时被调用。官网地址：https://cn.vuejs.org/v2/api/#选项-生命周期钩子

问题：说说 vue 实例生命周期钩子函数具体适用的场景?

beforecreate：可以在这加个 loading 事件，在加载实例时触发。

created：初始化完成时的事件写在这里，如在这结束 loading 事件，异步请求也适宜在这里调用。

mounted：挂载元素，获取到 DOM 节点。

updated：如果对数据统一处理，在这里写上相应函数。

beforeDestroy：可以做一个确认停止事件的确认框。

问题：第一次加载会触发哪几个钩子?

会触发 beforeCreate ， created , beforeMount , mounted。

问题：子组件与父组件生命周期发生的先后顺序

加载渲染过程：beforeCreate->父created->父beforeMount->子beforeCreate->子created->子beforeMount->子mounted->父mounted

子组件更新过程：父beforeUpdate->子beforeUpdate->子updated- >父updated

销毁过程：父beforeDestroy->子beforeDestroy- >子destroyed->父destroyed

考点 3：计算属性

问题：请说说 computed 和 methods、watch 的区别。

computed：是计算属性，依赖其他属性的值。具有缓存，只有他依赖的值发生变化，下一次取当前属性时才会重新计算，这样的好处是避免了每次取值时都重新计算。

watch：用作侦听器，当侦听的值发生改变时，其他变化会跟着改变或者有些操作会被触发，当需要在数据变化时执行异步或开销较大的操作时，使用 watch 是最合理的。

methods：方法页面刚加载时调用一次，结果不会缓存。methods 里面是用来定义函数的，它需要手动调用才能执行。而不像 watch 和 computed 那样，"自动执行"预先定义的函数。

考点 4：vuex

问题：Vuex 是什么？怎么使用？哪种功能场景使用它?

Vuex 是一个专为 Vue.js 应用程序开发的状态管理器， 采用集中式存储管理应用的所有组件的状态，主要是为了多页面、多组件之间的通信。

Vuex 有 5 个重要的属性， 分别是 state、getter、mutation、action、module， 由 view 层发起一个 Action 给 Mutation， 在 Mutation 中修改状态， 返回新的状态， 通过 Getter 暴露给 view 层的组件或者页面，页面监测到状态改变于是更新页面。

如果你的项目很简单，最好不要使用 Vuex，对于大型项目，Vuex 能够更好的帮助我们管理组件外部的状态，一般可以运用在购物车、登录状态、播放等场景中。

问题：Vuex 的五个属性分别如何使用

Vuex的State存储数据

Mutation是更改 Vuex 的 store 中的状态的唯一方法。在action中调用store.commit来触发Mutation

Action 类似于 mutation，不同在于：Action 提交的是 mutation，而不是直接变更状态。Action 可以包含任意异步操作。通过store.dispatch来触发Action

问题：请描述一下 Vuex 和 localStorage 的区别是什么?

1. vuex 存储在内存，而 localstorage 以文件的方式存储在本地

2. localstorage 只能存储字符串类型的数据，存储对象需要 JSON 的 stringify 和 parse 方法进行处理。读取内存比读取硬盘速度要快

3. vuex 是一个转为为 Vue.js 应用程序开发的状态管理模式。它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化。

4. vuex 用域组件之间的传值，而 localstorage 是本地存储，是将数据存储到浏览器的方法，一般是在跨页面传递数据时使用。

5. vuex 能做到数据的响应式，localstorage 不能做到

6. 刷新页面时 vuex 存储的值会丢失， localstorage 是永久性，不会丢失。很多人觉得用 localstorage 可以代替 vuex，对于不变的数据确实可以，但是当两个组件公用一个数据源时，如果其中一个组件改变了该数据源，希望另一个组件响应该变化时，localstorage 无法做到。

考点 5：vue-router

问题：vue-router 是什么？它有哪些组件?

传统页面切换是用超链接 a 标签进行切换，vue-router 是 vue.js 官方路由管理器。vue 的单页应用是基于路由和组件的，路由用于设定访问路径，并将路径和组件映射起来，有 router-link、router-view 组件。

问题：请说说 vue-router 的实现原理?

单一页面应用程序，只有一个完整的页面；它在加载页面时，不会加载整个页面，而是只更新某个指定的容器中内容。

单页面应用(SPA)的核心之一是: 更新视图而不重新请求页面。vue-router 在实现单页面前端路由时， 提供了两种方式：Hash 模式和 History 模式；根据 mode 参数来决定采用哪一种方式 。

Hash 模式：vue-router 默认 hash 模式，使用 URL 的 hash 来模拟一个完整的 URL，于是当 URL 改变时，页面不会重新加载。 hash（#）是 URL 的锚点，代表的是网页中的一个位置，单单改变# 后的部分，浏览器只会滚动到相应位置，不会重新加载网页，也就是说 #是用来指导浏览器动作的，同时每一次改变#后的部分，都会在浏览器的访问历史中增加一个记录，使用"后退"按钮，就可以回到上一 个位置；所以说 Hash 模式通过锚点值的改变，根据不同的值，渲染指定 DOM 位置的不同数据。History 模式：由于 hash 模式会在 url 中自带#，如果不想要很丑的 hash，我们可以用路由的 history 模式，只需要在配置路由规则时，加入"mode: ' history'"，这种模式充分利用 history.pushState API来完成 URL 跳转而无须重新加载页面。

问题：请说出使用路由模块来实现页面跳转的三种方式。

1：直接修改地址栏；

2：this.$router.push('路由地址')；

3：<router-link to="路由地址"></router-link>

问题：$route和$router的区别是什么

router为VueRouter的实例，是一个全局路由对象，包含了路由跳转的方法、钩子函数等。

route 是路由信息对象||跳转的路由对象，每一个路由都会有一个route对象，是一个局部对象，包含path,params,hash,query,fullPath,matched,name等路由信息参数。

问题：Router的传参方式有哪些，有什么区别

1.params 2.query 3.hash 4.history

Params只能使用name，不能使用path，参数不会显示在路径上，浏览器强制刷新参数会被清空Query:参数会显示在路径上，刷新不会被清空，name 可以使用path路径

Hash:原理是onhashchage事件，可以在window对象上监听这个事件

History:利用了HTML5 History Interface 中新增的pushState()和replaceState()方法。需要后台配置支持。如果刷新时，服务器没有响应的资源，会刷出404

问题：Router的懒加载是如何实现的

把不同路由对应的组件分割成不同的代码块，然后当路由被访问时才加载对应的组件即为路由的懒加载component:() = import('../views/home')

问题：Router的导航钩子有哪些？说说导航守卫

1.全局前置守卫

router.beforeEach((to, from, next) = {})

to:Route,代表要进入的目标，它是一个路由对象。

from:Route,代表当前正要离开的路由，也是一个路由对象

next:Function,必须需要调用的方法，具体的执行效果则依赖next方法调用的参数

next():进入管道中的下一个钩子，如果全部的钩子执行完了，则导航的状态就是comfirmed（确认的）next(false):终端当前的导航。如浏览器URL改变，那么URL会充值到from路由对应的地址。next('/')||next({path:'/'}):跳转到一个不同的地址。当前导航终端，执行新的导航。

next 方法必须调用，否则钩子函数无法

resolved2.全局后置钩子

router.afterEach((to, from) = {})

后置钩子并没有next函数，也不会改变导航本身。3.路由独享的守卫beforeEnter

4.组件内的守卫

1.beforeRouteEnter

2.beforeRouteUpdate

3.beforeRouteLeave

官方文档：https://router.vuejs.org/zh/guide/advanced/navigation-guards.html#%E8%B7%AF%E7%94%B1%E7%8B%AC%E4%BA%AB%E7%9A%84%E5%AE%88%E5%8D%AB

考点 6：组件通信

问题：常用的 Vue 组件之间的通信方式有哪些?

组件之间通信包括三种：父组件传子组件，子组件传父组件，兄弟组件之间。父传子：

主要是通过 props 来实现的 ，父组件需要通过 import 引入子组件，并注册， 在子组件里面添加要传递的属性，子组件用 props 来接收，接收的方式有两种，一种是用对象的形式{}来接收，对象的形式可以传递数据的类型，和必填，另一种是用数组的形式，数组只是简单的接收值。

子传父：

主要是通过$on、$emit (发布订阅)来实现的 ，在子组件中使用$emit，定义事件名和传递的数据；

在父组件中，监听指定事件名的事件触发，并接收传过来的数据。

兄弟组件：

1. 可以使用 EventBus 来作为沟通桥梁，就像是所有组件共用相同的事件中心，可以向该中心注册发送事件或接收事件，所以组件都可以上下平行地通知其他组件。

2. 也可以使用更完善的 Vuex 作为状态管理中心，将通知的概念上升到共享状态层次。通讯方式还可以通过$parent和$children，provide/inject
$parent和$children官网地址 : https://cn.vuejs.org/v2/api/#parent
provide/inject官网地址: https://cn.vuejs.org/v2/api/#provide-inject

考点 7：vue-cli

问题：构建的 vue-cli 工程都到了哪些技术，它们的作用分别是什么?

1. 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
2. Sass，Less
3. vue-router：vue 官方推荐使用的路由框架。
4. 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。
5. axios（或者 fetch、ajax）：用于发起 GET、或 POST 等 http 请求，基于 Promise 设计。
6. elementui 等：一个专为 vue 设计的移动端 UI 组件库。
7. 创建一个 emit.js 文件，用于 vue 事件机制的管理。
8. webpack：模块加载和 vue-cli 工程打包器。

考点 8：常用指令

问题：说说你在日常开发中 vue 常用的指令有哪些?

v-text：更新元素的 textContent，将数据解析为纯文本。

v-html：更新元素的 innerHTML，将数据解析成 html 标签。

v-once：只渲染元素和组件一次。随后的重新渲染，元素/组件及其所有的子节点将被视为静态内容并跳过。可以用于优化更新性能。

v-show：条件渲染指令，与 v-if 不同的是，无论 v-show 的值为 true 或 false，元素都会存在于 HTML 代码中； 而只有当 v-if 的值为 true，元素才会存在于 HTML 代码中。v-show 指令只是设置了元素 CSS 的 style 值。

v-if：条件判断指令，根据表达式值的真假来插入或删除元素。

v-for：循环指令，基于一个数组渲染一个列表，与 JavaScript 的 for 循环类似。

v-bind：给 DOM 绑定元素属性。

v-on：用于监听 DOM 事件。

v-model：实现数据的双向绑定，如果用于表单控件以外的标签是没有用的。

问题：v-for 中为什么使用 key?

在使用 vue 列表渲染 v-for 时，我们最好在后面加上: key，它的作用是当 v-for 所绑定的数据发声变化时只重新渲染变化的项，而不是重新渲染整个列表。

这除了可以节约资源以外更重要的是可以避免一些 bug，比如使用列表渲染生成的一个多选项，我们勾选了其中的第一项，勾选后我们在向数据的开头加入一个数据，如果我们没有使用 key 的话，就会发现勾选的变成我们新加的项，而之前的第一项变成了第二项，且没有被勾选。

加了 key 之后就可以避免这个现象， key 的值只要是一个唯一的数据就可以，一般情况我们使用列表渲染中的 index。v-if 指令是直接销毁和重建 DOM 达到让元素显示和隐藏的效果。

问题：说说 v-show 和 v-if 指令的共同点和不同点。

v-show 指令是通过修改元素的 display 的 CSS 属性让其显示或者隐藏。

v-if 指令是直接销毁和重建 DOM 达到让元素显示和隐藏的效果。

如果需要非常频繁地切换，则使用 v-show 较好；如果在运行时条件很少改变，则使用 v-if 较好。

问题：为什么 Vue 中的 v-if 和 v-for 不建议一起用?

当 v-if 与 v-for 一起使用时， v-for 具有比 v-if 更高的优先级。这意味着 v-if 将分别重复运行于每个 v-for 循环中。即先运行 v-for 的循环，然后在每一个 v-for 的循环中， 再进行 v-if 的条件对比。会造成性能问题，影响速度。

为了避免出现这种情况，乐意在外层嵌套 template（页面渲染不生成 dom 节点），在这一层进行

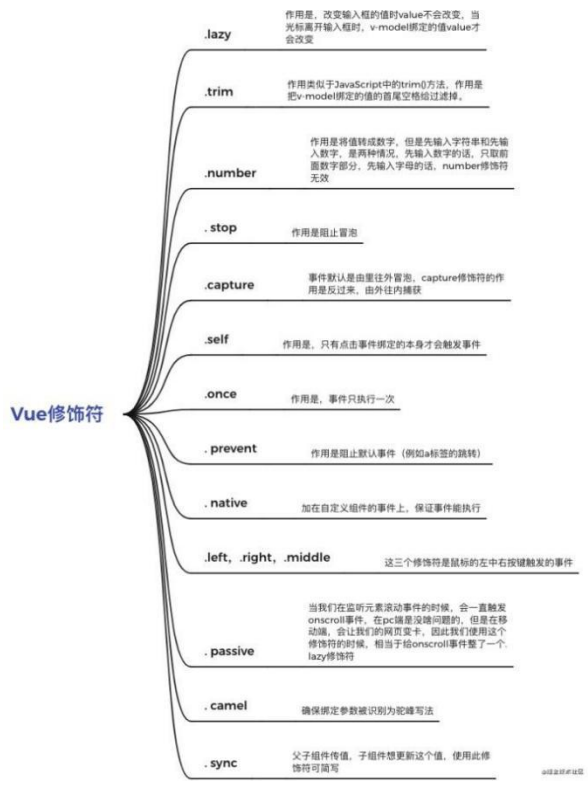
```
<template v-if="isShow">
    <p v-for="item in items">
</template>
```

v-if 判断，然后在内部进行 v-for 循环。

如果条件出现在循环内部，可通过计算属性 computed 提前过滤掉那些不需要显示的项

```
computed: {
  items: function() {
    return this.list.filter(function (item)
      {return item.isShow
    })
  }
}
```

问题：使用过哪些Vue的修饰符呢?



问题

1. 使用过哪些Vue的内部指令呢?

答案

考点 9：其他问题

问题：说computed和watch有何区别?

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

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

问题：为什么v-if和v-for不建议用在同一标签?

v-for优先级是高于v-if，v-for和v-if同时存在，会先把元素都遍历出来，然后再一个个判断，并隐藏掉，这样的坏处就是，渲染了无用的节点，增加无用的dom操作

问题：直接arr[index] = xxx无法更新视图怎么办？为什么?

Vue没有对数组进行Object.defineProperty的属性劫持，所以直接arr[index] = xxx是无法更新视图的
使用数组的splice方法，arr.splice(index, 1, item)
使用Vue.$set(arr, index, value)

问题：为什么data是个函数并且返回一个对象呢?

data之所以只一个函数，是因为一个组件可能会多处调用，而每一次调用就会执行data函数并返回新的数据对象，这样，可以避免多处调用之间的数据污染。

问题：计算变量时，methods和computed哪个好?

computed会好一些，因为computed会有缓存。例如index由0变成1，那么会触发视图更新，这时候methods会重新执行一次，而computed不会，因为computed依赖的两个变量num和price都没变

问题：相同的路由组件如何重新渲染?

开发人员经常遇到的情况是，多个路由解析为同一个Vue组件。问题是，Vue出于性能原因，默认情况下共享组件将不会重新渲染，如果你尝试在使用相同组件的路由之间进行切换，则不会发生任何变化

```
const routes =
    [{path: "/a",
    component: MyComponent
  },
  {
    path: "/b",
    component: MyComponent
  }];
```

如果依然想重新渲染，怎么办呢？可以使用key

```
<template>
    <router-view :key="$route.path"></router-view>
</template>
```

问题：Vue的SSR是什么？有什么好处

SSR就是服务端渲染
基于nodejs serve服务环境开发，所有html代码在服务端渲染
数据返回给前端，然后前端进行"激活"，即可成为浏览器识别的html代码
SSR首次加载更快，有更好的用户体验，有更好的seo优化，因为爬虫能看到整个页面的内容，如果是vue项目，由于数据还要经过解析，这就造成爬虫并不会等待你的数据加载完成，所以其实Vue项目的seo体验并不是很好