vue99面试题（https://github.com/minsion）

面试题 1. 请简述Vue ref 的作用是什么?

推荐指数：★★★ 试题难度： 中级 试题类型： 原理题

试题回答参考思路：

ref 的作用是被用来给元素或子组件注册引用信息。引用信息将会注册在父组件的 $refs 对象上。其特点是：

如果在普通的 DOM 元素上使用，引用指向的就是 DOM 元素
如果用在子组件上，引用就指向组件实例
所以常见的使用场景有：

基本用法，本页面获取 DOM 元素
获取子组件中的 data
调用子组件中的方法

面试题 2. Vue.extend 和 Vue.component 的区别是什么?

推荐指数：★★★★★ 试题难度： 中级 试题类型： 原理题

试题回答参考思路：

Vue.extend 用于创建一个基于 Vue 构造函数的“子类”，其参数应为一个包含组件选项的对象。
Vue.component 用来注册全局组件。

面试题 3. 移动端如何实现一个比较友好的 header 组件 ?

推荐指数：★★ 试题难度： 初级 试题类型： 原理题

试题回答参考思路：

Header 一般分为左、中、右三个部分，分为三个区域来设计，中间为主标题，每个页面的标题肯定不同，所以可以通过 vue props的方式做成可配置对外进行暴露，左侧大部分页面可能都是回退按钮，但是样式和内容不尽相同，右侧一般都是具有功能性的操作按钮，所以左右两侧可以通过 vue slot 插槽的方式对外暴露以实现多样化，同时也可以提供 default slot 默认插槽来统一页面风格

面试题 4. Vue 为什么没有类似于 React 中 shouldComponentUpdate 的生命周期?

推荐指数：★★★★ 试题难度： 中级 试题类型： 原理题

试题回答参考思路：

根本原因是 Vue 与 React 的变化侦测方式有所不同

React 是 pull 的方式侦测变化，当 React 知道发生变化后，会使用 Virtual Dom Diff 进行差异检测，但是很多组件实际上是肯定不会发生变化的，这个时候需要用 shouldComponentUpdate 进行手动操作来减少 diff，从而提高程序整体的性能。

Vue 是 pull+push 的方式侦测变化的，在一开始就知道那个组件发生了变化，因此在 push 的阶段并不需要手动控制 diff，而组件内部采用的 diff 方式实际上是可以引入类似于 shouldComponentUpdate 相关生命周期的，但是通常合理大小的组件不会有过量的 diff，手动优化的价值有限，因此目前 Vue 并没有考虑引入 shouldComponentUpdate 这种手动优化的生命周期

面试题 5. 简述接口请求一般放在哪个生命周期中？为什么要这样做?

推荐指数：★★★ 试题难度： 中级 试题类型： 原理题

试题回答参考思路：

接口请求可以放在钩子函数 created、beforeMount、mounted 中进行调用，因为在这三个钩子函数中，data 已经创建，可以将服务端端返回的数据进行赋值。

但是推荐在 created 钩子函数中调用异步请求，因为在 created 钩子函数中调用异步请求有以下优点：

能更快获取到服务端数据，减少页面 loading 时间
SSR 不支持 beforeMount 、mounted 钩子函数，所以放在 created 中有助于代码的一致性
created 是在模板渲染成 html 前调用，即通常初始化某些属性值，然后再渲染成视图。如果在 mounted 钩子函数中请求数据可能导致页面闪屏问题

面试题 6. 请简述Vue事件绑定原理？

推荐指数：★★★★★ 试题难度： 中级 试题类型： 原理题

试题回答参考思路：

一、事件绑定概述
在Vue开发中，我们经常需要给DOM元素绑定事件来响应用户的操作。Vue的事件绑定提供了简洁的语法和灵活的方式，可以轻松地实现各种事件绑定。而Vue将事件处理器绑定在DOM元素上的方式与原生JavaScript有所不同，下面我们来详细了解Vue事件绑定的原理。

二、事件绑定实现方式
Vue的事件绑定主要有以下几种方式：

v-on: 绑定DOM事件，简写为@
v-bind: 绑定DOM属性/插值表达式，简写为:

v-model: 双向数据绑定
自定义事件：父组件向子组件传递数据，简化父子之间事件的使用
其中，v-on绑定DOM事件是最常用的方式，下面我们来详细介绍v-on的实现方式。

三、事件绑定原理解析
1. 事件绑定本质
事件绑定本质上是将DOM事件与Vue实例中的方法进行绑定，当DOM事件被触发时，相应的Vue方法会被调用。

2. 事件绑定的实现方式
Vue的事件绑定是通过DOM事件监听器实现的。当用户触发某个DOM事件时，Vue将会执行Vue实例中相应的方法。Vue实现了一种特殊的指令v-on，通过它可以指定事件的类型和触发该事件的处理函数。

3. 事件绑定的事件处理函数
事件处理函数就是在相应的DOM事件被触发时所要执行的JavaScript函数。Vue事件处理函数一般有两种形式：

在methods中定义处理函数：

内联JavaScript语句：

4. 事件绑定的底层实现
Vue事件绑定的底层实现是通过addEventListener方法实现的。在Vue初始化时，会进行事件监听器的绑定操作，为DOM元素添加相应的监听器函数。当DOM事件被触发时，Vue会调用事件监听器函数，然后执行相应的Vue方法。

四、事件绑定的优点
Vue事件绑定具有如下优点：

Vue事件绑定实现了DOM事件与Vue实例的解耦，使得代码更加灵活和可维护。
Vue事件绑定提供了简洁的语法，方便开发者进行快速开发。
Vue事件绑定提供了丰富的事件类型，能够满足各种场景的需求。

面试题 7. v-on 可以实现监听多个方法么?

推荐指数：★★★★★ 试题难度： 中级 试题类型： 原理题

试题回答参考思路：

可以监听多个方法。关于监听多个方法提供了几种不同的写法：

写法一：

```
<div v-on="{ 事件类型: 事件处理函数, 事件类型: 事件处理函数 }"></div>
```

写法二：

```
<div @事件类型=“事件处理函数” @事件类型=“事件处理函数”></div>
```

写法三：在一个事件里面书写多个事件处理函数

写法四：在事件处理函数内部调用其他的函数
示例代码如下：

面试题 8. Vue 的数据为什么频繁变化但只会更新一次?

推荐指数：★★★★ 试题难度：中级 试题类型：原理题

试题回答参考思路：

这是因为 vue 的 DOM 更新是一个异步操作，在数据更新后会首先被 set 钩子监听到，但是不会马上执行 DOM 更新，而是在下一轮循环中执行更新。

具体实现是 vue 中实现了一个 queue 队列用于存放本次事件循环中的所有 watcher 更新，并且同一个 watcher 的更新只会被推入队列一次，并在本轮事件循环的微任务执行结束后执行此更新(UI Render 阶段)，这就是 DOM 只会更新一次的原因。

这种在缓冲时去除重复数据对于避免不必要的计算和 DOM 操作是非常重要的。然后，在下一个的事件循环“tick”中，vue 刷新队列并执行实际（已去重的）工作。vue 在内部对异步队列尝试使用原生的 Promise.then、MutationObserver 和 setImmediate，如果执行环境不支持，则会采用 setTimeout(fn, 0) 代替

面试题 9. 阐述Vue 中 computed 和 methods 的区别 ?

推荐指数：★★★★★ 试题难度：高难 试题类型：原理题

试题回答参考思路：

首先从表现形式上面来看， computed 和 methods 的区别大致有下面 4 点：

在使用时，computed 当做属性使用，而 methods 则当做方法调用

computed 可以具有 getter 和 setter，因此可以赋值，而 methods 不行
computed 无法接收多个参数，而 methods 可以
computed 具有缓存，而 methods 没有
而如果从底层来看的话， computed 和 methods 在底层实现上面还有很大的区别。

vue 对 methods 的处理比较简单，只需要遍历 methods 配置中的每个属性，将其对应的函数使用 bind 绑定当前组件实例后复制其引用到组件实例中即可

而 vue 对 computed 的处理会稍微复杂一些。

面试题 10. 请说明给 vue 中的元素设置 key 值时可以使用 Math 的 random 方法么?

推荐指数： ★★★ 试题难度： 初级 试题类型： 原理题

试题回答参考思路：

random 是生成随机数，有一定概率多个 item 会生成相同的值，不能保证唯一。
如果是根据数据来生成 item，数据具有 id 属性，那么就可以使用 id 来作为 key。
如果不是根据数据生成 item，那么最好的方式就是使用时间戳来作为 key。或者使用诸如 uuid 之类的库来生成唯一的 id

面试题 11. 解释Vue 插槽与作用域插槽的区别是什么?

推荐指数： ★★★★★ 试题难度： 初级 试题类型： 原理题

试题回答参考思路：

插槽
创建组件虚拟节点时，会将组件儿子的虚拟节点保存起来。当初始化组件时，通过插槽属性将儿子进行分类 {a:[vnode],b[vnode]}
渲染组件时会拿对应的 slot 属性的节点进行替换操作。（插槽的作用域为父组件）

作用域插槽
作用域插槽在解析的时候不会作为组件的孩子节点。会解析成函数，当子组件渲染时，会调用此函数进行渲染。
普通插槽渲染的作用域是父组件，作用域插槽的渲染作用域是当前子组件。

面试题 12. 请简述Vue 中相同逻辑如何进行抽离?

推荐指数： ★★★★ 试题难度： 中级 试题类型： 原理题

试题回答参考思路：

可以使用 vue 里面的混入（mixin）技术。混入（mixin）提供了一种非常灵活的方式，来将 vue 中相同的业务逻辑进行抽离。

例如：

在 data 中有很多是公用数据
引用封装好的组件也都是一样的
methods、watch、computed 中也都有大量的重复代码
当然这个时候可以将所有的代码重复去写来实现功能，但是我们并不不推荐使用这种方式，无论是工作量、工作效率和后期维护来说都是不建议的，这个时候 mixin 就可以大展身手了。

一个混入对象可以包含任意组件选项。当组件使用混入对象时，所有混入对象的选项将被“混合”进入该组件本身的选项。说白了就是给每个生命周期，函数等等中间加入一些公共逻辑。

混入技术特点

当组件和混入对象含有同名选项时，这些选项将以恰当的方式进行“合并”。比如，数据对象在内部会进行递归合并，并在发生冲突时以组件数据优先。
同名钩子函数将合并为一个数组，因此都将被调用。另外，混入对象的钩子将在组件自身钩子之前调用。
值为对象的选项，例如 methods、components 和 directives，将被合并为同一个对象。两个对象键名冲突时，取组件对象的键值对

面试题 13. 如何监听 pushstate 和 replacestate 的变化呢?

推荐指数： ★★★★ 试题难度： 高难 试题类型： 原理题

试题回答参考思路：

History.replaceState 和 pushState 不会触发 popstate 事件，所以我们可以通过在方法中创建一个新的全局事件来实现 pushstate 和 replacestate 变化的监听。

具体做法为：

```
var _wr = function(type) {
var orig = history[type];
return function() {
var rv = orig.apply(this, arguments);
var e = new Event(type);
e.arguments = arguments;
window.dispatchEvent(e);
return rv;
};
};
history.pushState = _wr('pushState');
history.replaceState = _wr('replaceState');
```

这样就创建了 2 个全新的事件，事件名为 pushState 和 replaceState，我们就可以在全局监听

```
window.addEventListener('replaceState', function(e) {
console.log('THEY DID IT AGAIN! replaceState 111111');
});
window.addEventListener('pushState', function(e) {
console.log('THEY DID IT AGAIN! pushState 2222222');
});
```

这样就可以监听到 pushState 和 replaceState 行为。

面试题 14. 简述 Vue3.0 为什么速度更快？

推荐指数：★★★★★ 试题难度：中级 试题类型：原理题

试题回答参考思路：

优化 Diff 算法

相比 Vue 2，Vue 3 采用了更加优化的渲染策略。去掉不必要的虚拟 DOM 树遍历和属性比较，因为这在更新期间往往会产生最大的性能开销。

这里有三个主要的优化：

首先，在 DOM 树级别。
在没有动态改变节点结构的模板指令（例如 v-if 和 v-for）的情况下，节点结构保持完全静态。

当更新节点时，不再需要递归遍历 DOM 树。所有的动态绑定部分将在一个平面数组中跟踪。这种优化通过将需要执行的树遍历量减少一个数量级来规避虚拟 DOM 的大部分开销。

其次，编译器积极地检测模板中的静态节点、子树甚至数据对象，并在生成的代码中将它们提升到渲染函数之外。这样可以避免在每次渲染时重新创建这些对象，从而大大提高内存使用率并减少垃圾回收的频率。

第三，在元素级别。

编译器还根据需要执行的更新类型，为每个具有动态绑定的元素生成一个优化标志。

例如，具有动态类绑定和许多静态属性的元素将收到一个标志，提示只需要进行类检查。运行时将获取这些提示并采用专用的快速路径。

综合起来，这些技术大大改进了渲染更新基准，Vue 3.0 有时占用的 CPU 时间不到 Vue 2 的十分之一。

体积变小

重写后的 Vue 支持了 tree-shaking，像修剪树叶一样把不需要的东西给修剪掉，使 Vue 3.0 的体积更小。

需要的模块才会打入到包里，优化后的 Vue 3.0 的打包体积只有原来的一半（13kb）。哪怕把所有的功能都引入进来也只有 23kb，依然比 Vue 2.x 更小。像 keep-alive、transition 甚至 v-for 等功能都可以按需引入。

并且 Vue 3.0 优化了打包方法，使得打包后的 bundle 的体积也更小。

官方所给出的一份惊艳的数据：打包大小减少 41%，初次渲染快 55%，更新快 133%，内存使用减少 54%。

面试题 15. 简述Vue自定义指令有哪些生命周期?

推荐指数：★★★★★ 试题难度：中级 试题类型：原理题

试题回答参考思路：

自定义指令的生命周期，有 5 个事件钩子，可以设置指令在某一个事件发生时的具体行为：

bind: 只调用一次，指令第一次绑定到元素时调用，用这个钩子函数可以定义一个在绑定时执行一次的初始化动作。
inserted: 被绑定元素插入父节点时调用（父节点存在即可调用，不必存在于 document 中）。
update: 被绑定元素所在的模板更新时调用，而不论绑定值是否变化。通过比较更新前后的绑定值，可以忽略不必要的模板更新（详细的钩子函数参数见下）。
componentUpdated: 被绑定元素所在模板完成一次更新周期时调用。
unbind: 只调用一次， 指令与元素解绑时调用。
钩子函数的参数（包括 el，binding，vnode，oldVnode）

el: 指令所绑定的元素，可以用来直接操作 DOM 。
binding: 一个对象，包含以下属性：name: 指令名、value: 指令的绑定值、oldValue: 指令绑定的前一个值、expression: 绑定值的字符串形式、arg: 传给指令的参数、modifiers: 一个包含修饰符的对象。
vnode: Vue 编译生成的虚拟节点。
oldVnode: 上一个虚拟节点，仅在 update 和 componentUpdated 钩子中可

面试题 16. Vue 组件中写 name 选项有哪些好处？

推荐指数：★★★★ 试题难度：初级 试题类型：原理题

试题回答参考思路：

可以通过名字找到对应的组件（ 递归组件：组件自身调用自身 ）
可以通过 name 属性实现缓存功能（keep-alive）
可以通过 name 来识别组件（跨级组件通信时非常重要）
使用 vue-devtools 调试工具里显示的组见名称是由 vue 中组件 name 决定的

面试题 17. 简述Vue中如何扩展一个组件？

推荐指数： ★★★★★ 试题难度： 中级 试题类型： 原理题

试题回答参考思路：

常见的组件扩展方法有：mixins，slots，extends等

混入mixins是分发 Vue组件中可复用功能的非常灵活的方式。混入对象可以包含任意组件选项。当组件使用混入对象时，所有混入对象的选项将被混入该组件本身的选项。

插槽主要用于vue组件中的内容分发，也可以用于组件扩展。
子组件Child

这个内容会被父组件传递的内容替换
3
父组件Parent

来自老爹的内容

如果要精确分发到不同位置可以使用具名插槽，如果要使用子组件中的数据可以使用作用域插槽。

混入的数据和方法不能明确判断来源且可能和当前组件内变量产生命名冲突，vue3中引入的composition api，可以很好解决这些问题，利用独立出来的响应式模块可以很方便的编写独立逻辑并提供响应式的数据，然后在setup选项中组合使用，增强代码的可读性和维护性。例如：

```
// 复用逻辑1
function useXX() {}
// 复用逻辑2
function useYY() {}
// 逻辑组合
const Comp = {
setup() {
const {xx} = useXX()
const {yy} = useYY()
return {xx, yy}
}
}
```

面试题 18. Vue中子组件可以直接改变父组件的数据么，说明原因？

推荐指数： ★★★★★ 试题难度： 中级 试题类型： 原理题

试题回答参考思路：

所有的 prop 都使得其父子之间形成了一个单向下行绑定：父级 prop 的更新会向下流动到子组件中，但是反过来则不行。这样会防止从子组件意外变更父级组件的状态，从而导致你的应用的数据流向难以理解。另外，每次父级组件发生变更时，子组件中所有的 prop 都将会刷新为最新的值。这意味着你不应该在一个子组件内部改变 prop。如果你这样做了，Vue 会在浏览器控制台中发出警告。

```
const props = defineProps(['foo'])
// ❌ 下面行为会被警告，props是只读的!
props.foo = 'bar'
```

实际开发过程中有两个场景会想要修改一个属性：

这个 prop 用来传递一个初始值；这个子组件接下来希望将其作为一个本地的 prop 数据来使用。在这种情况下，最好定义一个本地的 data，并将这个 prop 用作其初始值：

```
const props = defineProps(['initialCounter'])
const counter = ref(props.initialCounter)
```

这个 prop 以一种原始的值传入且需要进行转换。在这种情况下，最好使用这个 prop 的值来定义一个计算属性：

```
const props = defineProps(['size'])
// prop变化，计算属性自动更新
const normalizedSize = computed(() => props.size.trim().toLowerCase())
```

实践中如果确实想要改变父组件属性应该emit一个事件让父组件去做这个变更。注意虽然我们不能直接修改一个传入的对象或者数组类型的prop，但是我们还是能够直接改内嵌的对象或属性。

面试题 19. Vue过渡动画实现的方式有哪些?

推荐指数：★★★ 试题难度：高难 试题类型：原理题

试题回答参考思路：

1.使用vue的transition标签结合css样式完成动画
2.利用animate.css结合transition实现动画
3.利用 vue中的钩子函数实现动画

面试题 20. Vue watch怎么深度监听对象变化 ？

推荐指数：★★★★★ 试题难度：高难 试题类型：原理题

试题回答参考思路：

deep设置为true 就可以监听到对象的变化let vm=new Vue({el:"#first",data:{msg:{name:' 北 京 '}},watch:{msg:{handler (newMsg,oldMsg) {console.log(newMsg);},immediate:true,deep:true}}})

面试题 21. 动态给vue的data添加一个新的属性时会发生什么？怎样解决?

推荐指数：★★ 试题难度：初级 试题类型：原理题

试题回答参考思路：

如果在实例创建之后添加新的属性到实例上，它不会触发视图更新。如果想要使添加的值做到响应式，应当使用$set()来添加对象。

面试题 22. 简述active-class是哪个组件的属性？嵌套路由怎么定义？

推荐指数：★★★★★ 试题难度：初级 试题类型：原理题

试题回答参考思路：

vue-router模块的router-link组件。

面试题 23. 简述scss是什么？在vue.cli中的安装使用步骤是？有哪几大特性？

推荐指数：★★ 试题难度：初级 试题类型：原理题

试题回答参考思路：

css的预编译。
使用步骤：
第一步：用npm 下三个loader（sass-loader、css-loader、node-sass）
第二步：在build目录找到webpack.base.config.js，在那个extends属性中加一个拓展.scss
第三步：还是在同一个文件，配置一个module属性
第四步：然后在组件的style标签加上lang属性，例如：lang=”scss”
有哪几大特性:
1、可以用变量，例如（$变量名称=值）；
2、可以用混合器，例如（）
3、可以嵌套

面试题 24. 简述mint-ui是什么？怎么使用？说出至少三个组件使用方法？

推荐指数：★★★★★ 试题难度：初级 试题类型：原理题

试题回答参考思路：

基于vue的前端组件库。npm安装，然后import样式和js，vue.use（mintUi）全局引入。在单个组件局
部引入：import {Toast} from ‘mint-ui’。组件一：Toast(‘登录成功’)；组件二：mint-header；组件三：
mint-swiper

面试题 25. 简述Vue.js的template编译的理解？

推荐指数：★★★★ 试题难度：中级 试题类型：原理题

试题回答参考思路：

简而言之，就是先转化成AST树，再得到的render函数返回VNode（Vue的虚拟DOM节点）
详情步骤：
首先，通过compile编译器把template编译成AST语法树（abstract syntax tree 即 源代码的抽象语法结构的树状表现形式），compile是createCompiler的返回值，createCompiler是用以创建编译器的。另外compile还负责合并option。
然后，AST会经过generate（将AST语法树转化成render funtion字符串的过程）得到render函数，render的返回值是VNode，VNode是Vue的虚拟DOM节点，里面有（标签名、子节点、文本等等）

面试题 26. 简述Vue声明组件的state是用data方法，那为什么data是通过一个function来返 回一个对象，而不是直接写一个对象呢?

推荐指数：★★★★ 试题难度：中级 试题类型：原理题

试题回答参考思路：

从语法上说，如果不用function返回就会出现语法错误导致编译不通过。从原理上的话，大概就是组件可以被多次创建，如果不使用function就会使所有调用该组件的页面公用同一个数据域，这样就失去了组件的概念了

面试题 27. 简述Vue中mixin与extend区别？

推荐指数：★★★ 试题难度：中级 试题类型：原理题

试题回答参考思路：

全局注册混合对象，会影响到所有之后创建的vue实例，而Vue.extend是对单个实例进行扩展。
mixin 混合对象（组件复用）
同名钩子函数（bind，inserted，update，componentUpdate，unbind）将混合为一个数组，因此都将被调用，混合对象的钩子将在组件自身钩子之前调用methods，components，directives将被混为同一个对象。两个对象的键名（方法名，属性名）冲突时，取组件（而非mixin）对象的键值对。

面试题 28. 简述prop 如何指定其类型要求？

推荐指数：★★★ 试题难度：初级 试题类型：原理题

试题回答参考思路：

通过实现 prop 验证选项，可以单个 prop 指定类型要求。这对生产没有影响，但是会在开发段发出警告，从而帮助开发人员识
别传人数据和 prop 的特定类型要求的潜在问题。
配置三个 prop 的例子 ：
props：{
accountNumber:{
type：Number,

required : true
},
name :{
type : String,
required : true
},
favoriteColors : Array
}

面试题 29. 简述什么是 mixin ？

推荐指数： ★★★★★ 试题难度： 初级 试题类型： 原理题

试题回答参考思路：

Mixin 使我们能够为 Vue 组件编写可插拔和可重用的功能。 如果你希望再多个组件之间重用一组组件选项，例如生命周期hook、 方法等，则可以将其编写为 mixin，并在组件中简单的引用它。然后将 mixin 的内容合并到组件中。如果你要在 mixin中定义生命周期 hook，那么它在执行时将优化于组件自己的 hook。

面试题 30. 简述什么是Vue渲染函数？举个例子？

推荐指数： ★★★★ 试题难度： 初级 试题类型： 原理题

试题回答参考思路：

Vue 允许我们以多种方式构建模板，其中最常见的方式是只把 HTML 与特殊指令和 mustache 标签一起用于相响应功能。但是
你也可以通过 JavaScript 使用特殊的函数类 （称为渲染函数） 来构建模板。这些函数与编译器非常接近，这意味它们比其他
模板类型更高效、快捷。由于你使用 JavaScript 编写渲染函数，因此可以在需要的地方自由使用该语言直接添加自定义函
数。
对于标准 HTML 模板的高级方案非常有用。
这里是用 HTML 作为模板 Vue 程序
new Vue ({
el: '#app',
data:{
fruits: ['Apples','Oranges','Kiwi'] },
template:
`

Fruit Basket

1. {{ fruit }}

`

});

这里是用渲染函数开发的同一个程序：

new Vue({

el: '#app',

data: {

fruits: ['Apples', 'Oranges', 'Kiwi'] },

render: function(createElement) {

return createElement('div', [

createElement('h1', 'Fruit Basket'),

createElement('ol', this.fruits.map(function(fruit) {

return createElement('li', fruit);

}))

]);

}

});

输出如下：

Fruit Basket

1、Apples 2、Oranges 3、Kiwi

在上面的例子中，我们用了一个函数，它返回一系列 createElement() 调用，每个调用负责生成一个元素。尽管 v-for 指令在

基于 HTML 的模板中起作用，但是当时用渲染函数时，可以简单的用标准的 .map() 函数遍历 fruits 数据数组。

面试题 31. 简述什么时候调用“updated”生命周期 hook？ ？

推荐指数： ★★★★ 试题难度： 初级 试题类型： 原理题

试题回答参考思路：

在更新响应性数据并重新渲染虚拟 DOM 之后，将调用更新的 hook。它可以用于执行与 DOM 相关的操作，但是（默认情况下）不能保证子组件会被渲染，尽管也可以通过在更新函数中使用 this.$nextTick 来确保。

面试题 32. 简述在 Vue 实例中编写生命周期 hook 或其他 option/propertie 时，为什么不使用箭头函数？

推荐指数： ★★★ 试题难度： 初级 试题类型： 原理题

试题回答参考思路：

箭头函数自已没有定义 this 上下文中。当你在 Vue 程序中使用箭头函数（ => ）时，this 关键字病不会绑定到 Vue 实例，因此会引发错误。所以强烈建议改用标准函数声明

面试题 33. 简述Vue中引入组件的步骤？

推荐指数： ★★ 试题难度： 初级 试题类型： 原理题

试题回答参考思路：

1.采用ES6的import ... from ...语法或CommonJS的require()方法引入组件
2.对组件进行注册,代码如下
// 注册
Vue.component('my-component', {
template: '

A custom component!
'
})
3.使用组件
子组件需要数据，可以在props中接受定义。而子组件修改好数据后，想把数据传递给父组件。可以采用emit方法。

面试题 34. 简述如何让CSS只在当前组件中起作用？

推荐指数： ★★★★★ 试题难度： 初级 试题类型： 原理题

试题回答参考思路：

将当前组件的