# 1.computed原理

watcher dirty属性，dirty为true重新计算，计算属性依赖的属性发生变化，dirty置为true，计算一次计算属性的值，如果发生变化就主动通知组件watcher进行重新渲染

# 2. watch原理

new Watcher 全局存储watcher，get读取数据的时候收集依赖，watch第一个参数可以是字符串也可以是函数，函数作为get函数，watch的数据变化时，重新执行操作 deep:true循环依赖

# 3. nextTick原理 为什么异步更新

vue中数据更新会通知watcher，触发虚拟dom更新渲染，vue中组件的所有状态发送到同一个Watcher，然后虚拟DOM会对整个组件进行比对，更改DOM。也就是说同一轮事件循环中两个数据发生了变化组件Watcher会收到两份通知，从而进行两次渲染，这是不需要的。虚拟DOM会对整个组件进行渲染，所以只需要等待所有的状态都修改完毕，一次性将整个组件的DOM渲染到最新即可。

vue内部对异步队列尝试使用Promise.then MutationObserver和setImmediate,如果执行环境不支持，则采用setTimeout(fn,0)支持

vue的实现方式是将收到通知的Watcher添加到队列缓存，并且在添加前检查是否已经存在，不存在添加，然后下一次事件循环，让队列中的Wacher触发渲染流程并清空队列

// this.$nextTick(function() { // DOM更新 }) // 然后修改数据 this.message = 'changed'

先调nextTick然后更改数据获取不到最新的DOM，因为更新DOM操作也是使用nextTick注册到微任务中的，会先执行自己的nextTick回调然后执行DOM更新的回调

# 4.$on $emit 注册事件的时候将回调函数收集起来，触发事件的时候将收集起来的回调函数依次调用

$on存取，$emit触发

# 5. $mount

参数是element or selector 手动挂载一个未挂载的一个实例，如果没有参数，则创建一个元素，必须使用原生DOM api插入到文档中，这个方法返回实例本身，所以可以链式调用

完整版本：会检查template或el选项提供的模板是否已经转换成了render函数，如果没有，进入编译过程，将其编译成渲染函数

运行时版本： 默认已存在渲染函数，不存在就创建一个空函数

挂载之前触发beforeMounted，挂载之后每当数据变化都会进行渲染

vm._watcher = new Watcher(vm, () => { vm._update(vm._render()) // **先调用渲染函数得到一份最新的VNode节点树，然后通过_update方法dom diff，更新DOM节点。简单来说就是执行了渲染操作。** },noop) callhook(vm,'mounted') return vm

**-update是调用虚拟DOM中的patch方法来执行节点的比对与渲染操作， _render执行渲染函数，得到一份最新的 VNode节点树**

挂载是持续性的，持续性的关键在于new Watcher第二个参数是一个函数，当是函数时候会同时观察函数中所读取的所有响应式数据

# 6. $set(target,key,value)原理

数组： $set(arr, 0,3) 先设置数组长度，如果数组长度小于传递的索引值，则数组长度等于索引值加1，否则不变 然后通过splice方法将value设置到指定索引 调用splice会收集依赖 返回value

对象： 判断可以在target中吗，存在即返回 判断target是否响应式，不是直接返回 否则是新增，调用defineReactive将新增属性转换getter，setter，然后向target的依赖触发变化通知，并返回val

# 7.为什么data是一个函数

数据以函数形式返回，这样每次复用组件都会返回一份新的data，每个组件实例维护各自的数据，否则组件之前的数据会相互影响

# 8. vue组件通讯

1. props 和on 父到子 子到父是$emit
2. $parent $children 获取当前组件的父子组件
3. $atters $children
4. provide inject
5. $refs获取组件实例
6. eventBus
7. vuex

# 9.生命周期

（服务端渲染不可用）： beforeMounted，mounted，beforeUpdate，updated，beforeDestroy destroy 都可用：beforeCreated created

activated keep-alive 专属，组件被激活时调用

deactivated keep-alive 专属，组件被销毁时调用

## 初始化顺序

props methods data computed watch

# 10.v-if和v-show

v-if会转换成三元表达式，不满足条件不渲染 v-show编译成指令，不满足条件则隐藏

使用场景： v-if适合改动不频繁 v-show适合频繁切换

# 11.vue内置指令

v-model v-cloak 页面闪烁问题 v-bind v-on v-html v-text v-if v-for v-show v-once

# 12如何理解vue单向数据流

数组总是从父到子，父无权直接修改父传递过来的数据，只能青青父对该数据修改，这样可防止子意外修改父导致应用数据流向难以理解 如果要改父传过来的prop： 子data定义一个变量，prop初始化它，然后$emit通知父改

# 13. computed和watch区别

computed 是计算属性，依赖其他属性计算值，并且 computed 的值有缓存，只有当计算值变化才会返回内容，它可以设置 getter 和 setter。 watch 监听到值的变化就会执行回调，在回调中可以进行一些逻辑操作。 计算属性一般用在模板渲染中，某个值是依赖了其它的响应式对象甚至是计算属性计算而来；而侦听属性适用于观测某个值的变化去完成一段复杂的业务逻辑

# 14.v-if和v-for为何不一起用

解析时会先解析v-for然后v-if，可以使用计算属性解决这个问题

# 15. vue2响应式原理

数据劫持+观察者模式

对象内部通过 defineReactive 方法，使用 Object.defineProperty 将属性进行劫持（只会劫持已经存在的属性），数组则是通过重写数组方法来实现。当页面使用对应属性时，每个属性都拥有自己的 dep 属性，存放他所依赖的 watcher（依赖收集），当属性变化后会通知自己对应的 watcher 去更新(派发更新)。

# 16 vue2检测数组变化

数组考虑性能原因没有用 defineProperty 对数组的每一项进行拦截，而是选择对 7 种数组（push,shift,pop,splice,unshift,sort,reverse）方法进行重写(AOP 切片思想) 所以在 Vue 中修改数组的索引和长度是无法监控到的。需要通过以上 7 种变异方法修改数组才会触发数组对应的 watcher 进行更新

# 17.vue父子组件生命周期钩子

加载渲染过程：

父beforeCreated - 父created - 父beforeMounted - 子beforeCreated -子created - 子beforeMounted - 子mounted - 父mounted

子更新：

父beforeUpdate - 子beforeUpdate - 子updated - 父updated

父更新

父beforeUpdate - 父updated

销毁：

父beforeDestroy - 子beforeDestroy - 子destroyed - 父destroyed

# 18 虚拟DOM是什么 有什么优缺点

操作dom代价比较高性能有问题。vnode是用一个js对象描述DOM节点，是对真实dom的抽象映射

优点： 避免频繁操作dom，进行了一些优化，比如dom diff，自动批量更新dom

缺点： 无法极致优化，首次渲染大量dom多了一层虚拟DOM计算，会比innnerHTML慢

# 19. v-model原理

语法糖 普通标签： 绑定value，监听input value值= e.target.value

组件： props:value $emit $event.target.value

如果是其他类型input Vue.component('base-checkbox', { model: { prop: 'checked', event: 'change' }, props: { checked: Boolean }, template: `<input type="checkbox" v-bind:checked="checked" v-on:change="$emit('change', $event.target.checked)" >` })

# 20.事件绑定原理

原生事件绑定addEventListener绑定给真实元素 组件事件绑定，vue的$on实现

# 21vue中使用了哪些设计模式

1. 工厂模式：传入参数创建实例 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 vuex 和 vue-router 的插件注册方法 install 判断如果系统存在实例就直接返回掉
3. 发布订阅模式 事件机制 发布者 事件中心 订阅者
4. 观察者模式 响应式原理 目标和观察者（watcher）

# 22 Vue.mixin使用场景和原理

抽离公共逻辑 原理：组件初始化的时候会调用mergeOptions方法进行合并，对不同的属性进行合并

# 23. nextTick

下次DOM更新循环结束之后执行对应回调

# 24 keep-alive使用场景和原理

实现组件缓存，组件切换时候不会进行卸载 常用两个属性：include/exclude 允许组件有条件的进行缓存 两个生命周期 activated/deactivated，用来得知当前组件是否处于活跃状态。 keep-alive 的中还运用了 LRU(最近最少使用) 算法，选择最近最久未使用的组件予以淘汰。

# 25 Vue.extend作用和原理

使用基础Vue构造器，创建一个子类，参数是一个包含组件选项的对象 data选项是特例，在Vue.extend()中，它必须是函数

Vue.extend()方法内增加了缓存策略，反复调用Vue.extend其实应该返回同一个结果，只要返回结果是固定的，就可以将计算结果缓存，再次调用extend方法时，只需要从缓存中取出结果即可。 本质是创建一个函数并继承父级

使用场景：

比如全局提示组件，可以挂载到vue原型上，如果需要包含操作DOM就需要使用Vue.extend了

# 26 自定义指令及原理

自定义指令有五个生命周期：bind，inserted，update，componentUpdate，unbind 原理 1.在生成 ast 语法树时，遇到指令会给当前元素添加 directives 属性 2.通过 genDirectives 生成指令代码 3.在 patch 前将指令的钩子提取到 cbs 中,在 patch 过程中调用对应的钩子 4.当执行指令对应钩子函数时，调用对应指令定义的方法

# 27 vue修饰符

事件: stop 阻止传播 prevent capture 自身先处理然后内部 self 只自身 once 只一次 passive 不阻止事件默认行为

v-model： lazy 转换为change事件 number 数值类型 trim

键盘： enter tab delete...

系统: ctrl alt

鼠标 left right

# 28.vue模板编译原理

template转化为render函数

1. 转化为ast
2. 静态标记
3. 生成render函数

# 29.生命周期钩子实现原理

核心是 利用发布订阅模式先把用户传入的钩子函数订阅好，存储在数组里，然后在创建组件实例的过程中会依次执行对应的钩子方法（发布）

# 30 如何让CSS只在当前组件中起作用?

答：在组件中的 style 前面加上 scoped

# 31 vue-loader

Vue Loader 还提供了很多酷炫的特性：

允许为 Vue 组件的每个部分使用其它的 webpack loader，例如在