**Vue**

#### 1.VueAPP适配是怎么写的，用的什么单位？

页面上的尺寸（高度，宽度，字体）随屏幕宽度进行变化。

使用的是rem，相对于根节点字体的大小。

#### 2.计算属性和监听器有什么区别？

模板里不要有太多的逻辑，所以对于复杂的逻辑要使用计算属性，computed是由缓存的。

侦听属性是一种更通用的方式来观察和相应Vue实例的属性，侦听器适用于数据变化时执行异步或开销较大的操作时更有用。

watch只要被监听的数据发生改变，就会触发相应的逻辑代码，如果被监听的数据是一个对象，对象的属性发生改变时，监听是不会触发的，这时候就需要深度监听，只要obj的任何一个属性发生改变，都会触发相应的代码，但是拿不到oldVal。

deep：true

#### 3.v-for渲染列表时key的作用是什么？

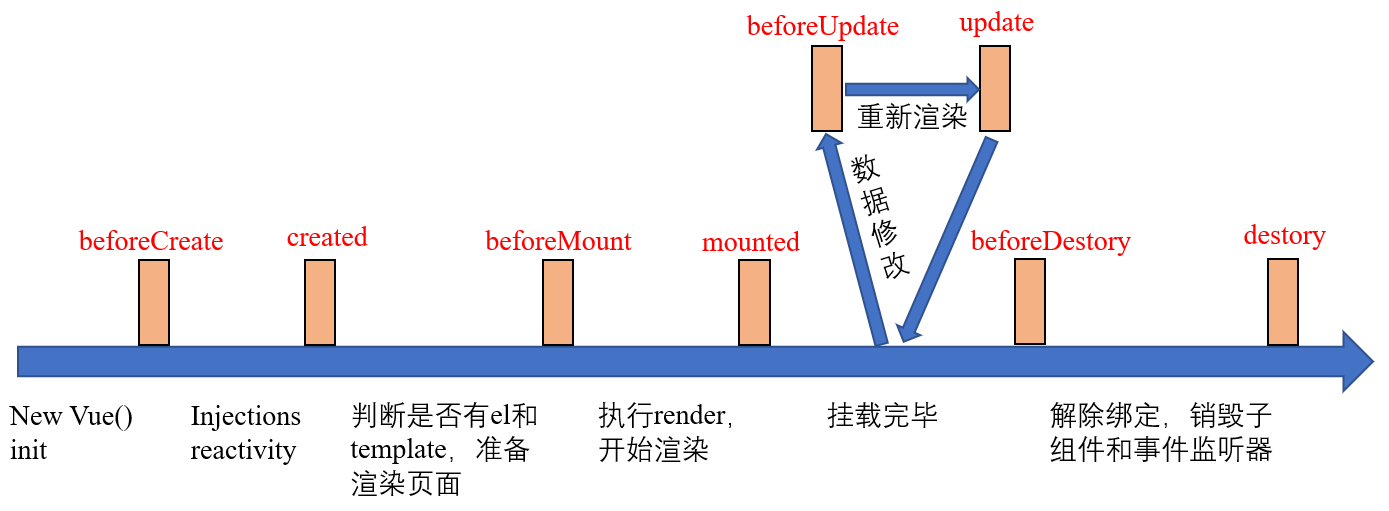
key是给每一个vnode的唯一id，可以通过这个key更准确更迅速的拿到oldVnode中对应的vnode节点。

如果没有key vue就会最大限度减少动态元素，并且尽可能尝试就地修改，带有key就不是就地复用，用于强制替换元素而不是重复使用它。

#### 4.生命周期（记住钩子函数和调用时间，以及一些原理）

Vue实例有一个完整的生命周期，生命周期也就是指一个实例从开始创建到销毁的这个过程。

生命周期钩子就是Vue实例在某一个时间点会自动执行的函数。



#### 5.v-if和v-show的区别和使用场景？

v-if用于条件性的渲染一块内容，决定一个标签是否存在。

v-show也是用于条件性渲染一块内容，只不过元素适中会被渲染并且保留在DOM中，只是简单地切换元素的display。

|  |  |  |
| --- | --- | --- |
|  | v-if | v-show |
| 渲染 | 条件为真时渲染 | 一直渲染 |
| 开销 | 切换开销较高 | 切换开销较少 |
| 使用场景 | 运行时条件很少改变 | 需要频繁切换 |

#### 6.Vue中插槽的作用？

插槽的作用是父组件向子组件插入一段内容。在子组件中使用<slot>就是引用父组件中使用的子组件里面的子节点，来代替<slot>节点。

匿名插槽（默认插槽）、具名插槽、作用域插槽

通过v-slot : name来使用具名插槽，v-slot只能添加在template标签上。

作用域插槽就是父组件可以拿到子组件中的data，在子组件中定义插槽先拿到data里面的值，然后在父组件中使用具名插槽，再使用具名插槽.子组件中插槽.data值就可以获得子组件中的值。

#### 7.jQuery和Vue的区别？

首先区分library和framework的区别，一个是用来实现特定需求API的集合，另一个是构建应用的主体结构。

jQuery是一个方法库，它是一系列操作DOM和AJAX的API组成的方法库，关注的是。

Vue是一个JavaScript Framework，实现的是MVVM设计模式，Vue关注的是视图层。

#### 8.Vue中父子组件和兄弟组件的传值？

父传子：父组件通过属性的形式向子组件传递，子组件通过props属性接收从父组件传过来的值（父组件传递的值子组件只能够使用，不能够修改，单项数据流）

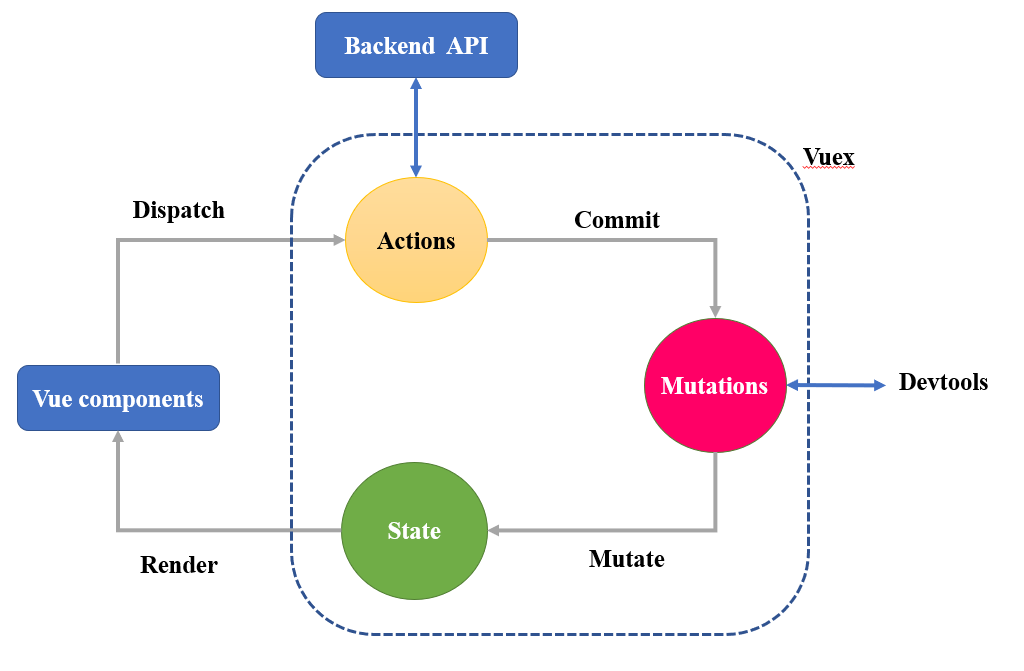
子传父：子组件以某种方式来触发一个自定义事件，需要传的值作为$emit的第二个参数，将该值作为实参传给相应自定义事件的方法，在父组件中注册子组件并在子组件标签上绑定对自定义事件的监听，然后执行方法。

兄弟组件传值：使用发布订阅模式（总线机制），Vue.prototype.bus=new Vue()，将bus属性挂载在Vue实例上面，则所有的Vue实例都会有bus属性，则this.$emit(‘change’，this.content)通过bus触发事件，并返回值；或者使用Vue推出的Vuex可以很方便的实现组件之间的参数传递。

#### 9.Vuex

Vuex是专门为Vue.js开发的状态管理模式。帮助我们管理共享状态（data里面的属性），适合中大型单页应用。

把共享的数据都保存在Vuex里面，单项数据改变流程。



根据这个图看出Actions作为数据的输入，State作为数据的输出，只能在mutations里面修改state，这个过程只能是同步的，如果是异步就先把操作放到actions里面，拿到数据以后在通过mutations同步处理。

Vuex的核心是store对象，通过Vue.Store这个构造函数new出来的实例，包括以下五个对象：

1. state—存放状态
2. getters—state的计算属性
3. mutations—更改状态的逻辑，同步操作
4. actions—提交mutation，异步操作
5. mudules—将store模块化

当一个组件获取多种状态的时候，则在计算属性中要写多个函数，为了方便，可以使用mapState辅助函数来帮我们生成计算属性。

#### 10.axios

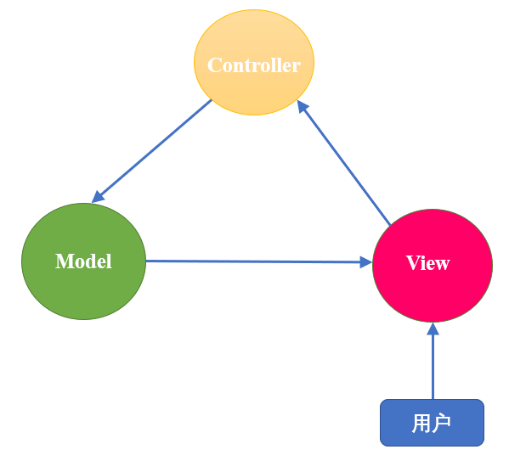
axios是一个基于Promise用于浏览器和nodejs的HTTP客户端，就是ajax的封装。

它本身具有以下特征：

1. 从浏览器中创建 XMLHttpRequest
2. 从 node.js 发出 http 请求
3. 支持 Promise API
4. 拦截请求和响应
5. 转换请求和响应数据
6. 取消请求
7. 自动转换JSON数据

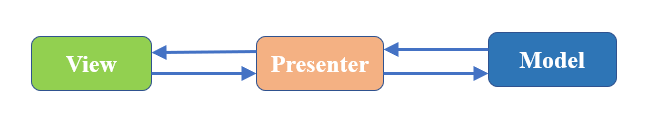
#### 11.对MVC、MVP和MVVM的理解？

MVC模式（Model View Controller）



用户操作界面，传递给controller，然后对模型进行修改或者查找底层数据，最后把改动渲染到视图上面。

MVP模式（Model View Present）



是MVC的改良模式，双向通信，View与Model之间不同信，通过Presenter进行传递，主要逻辑都在Presenter里面实现。

MVVM模式（Model View ViewModel）



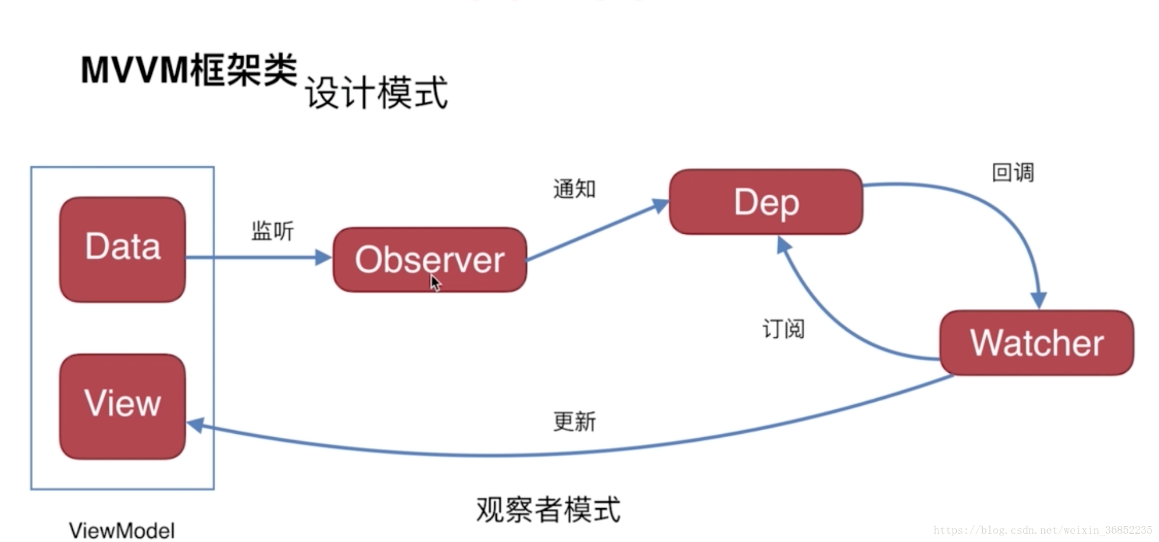
和MVC一样，主要是分离视图和模型，只需要关注数据即可，当数据变化的时候，页面也在变化，面向数据开发。

#### 12.Vue实现双向绑定的原理？

双向绑定就是Vue实例中的data与其渲染的DOM元素内容保持一致，无论谁被改变，另一方会相应的更新为相同的数据。

采用数据劫持结合发布者-订阅模式的方式，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。

就是通过Object.defineProperty(obj，prop，descriptor)，调用的时候使用get方法，赋值的时候使用set方法。

1. 实现一个监听者Oberver来劫持并监听所有的属性，一旦有属性发生变化就通知订阅者。
2. 实现一个订阅者watcher来接受属性变化的通知并执行相应的方法，从而更新视图。
3. 实现一个解析器compile，可以扫描和解析每个节点的相关指令，并根据初始化模板数据以及初始化相对应的订阅者

#### 13.数据劫持

数据劫持也叫数据代理，指的是在访问或者修改对象的某个属性时，通过一段代码拦截这个行为，进行额外的操作或者修改返回结果。

Vue使用的是Object.definProperty()（Vue3.X改用Proxy来实现）。

Object.definProperty()的缺点，深度监听需要递归到底，一次性计算量大，并且无法监听新增属性、删除属性，可以通过Vue.set和Vue.delete来设置。

#### 14VueRouter

路由就是指向，路由有三个概念：route（单个路由），routes（路由组），router（管理机制）。

通过在js文件配置路由确定正确的跳转，只要把路径和组件对应起来即可。

<touter-link>定义页面中的点击部分，to表示去到哪里。

<router-view>定义页面中的显示部分。

配置完成以后，把router实例注入到Vue根实例中就可以使用路由了。

动态路由就是把路由中的path选项添加上id即 { path:"/user/:id", component: user }，就可以根据不同的id跳转不同的页面。当整个Vue-router注入到根实例后，在组件内部，可以通过this.$route来获取到router实例，他有以一个params属性就是来获取这个动态部分。

#### 15.Vue router的钩子函数有哪些，哪一个是全局使用的钩子？

Vue router总共有六个钩子函数

全局的路由钩子函数：beforeEach、afterEach

单个路由钩子函数：beforeEnter

组件内的路由钩子函数：beforeRouteEnter、beforeRouteLeave、beforeRouteUpdate。

全局钩子函数每次路由跳转都会执行。

#### 16.Vue如何去除url中的#？

vue-router默认使用hash模式，所以在路由加载的时候，项目中的url会自带#，如果不想使用#可以使用vue-router的另外一种模式history。

new Router({

mode: 'history',

routes: [ ]

})

路由常用的两种模式：

1. 哈希模式：url尾巴后的#和字符。
2. history模式：不仅可以在url里面放参数，还可以将数据存放在一个特定的对象中，需要server端支持，无特殊需求可选前者。

#### 17.diff算法？

把DOM通过JS来模拟，进行计算对比，找出最小的更新范围，然后去更新，对比的过程就是diff算法。

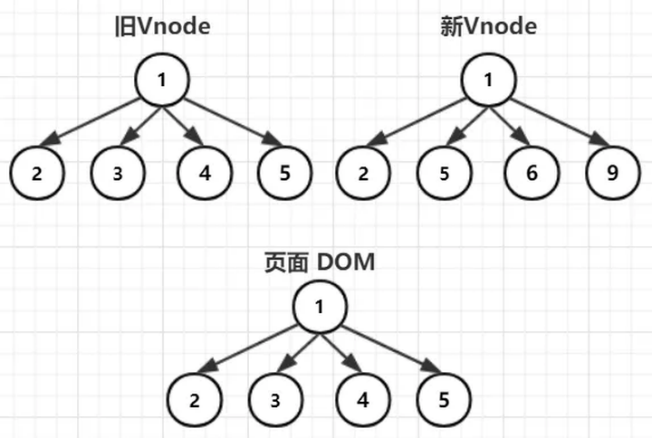
diff即对比，是一个广泛的概念，如linux diff、git diff等。

将时间复杂度从O(n^3)优化到O(n)。

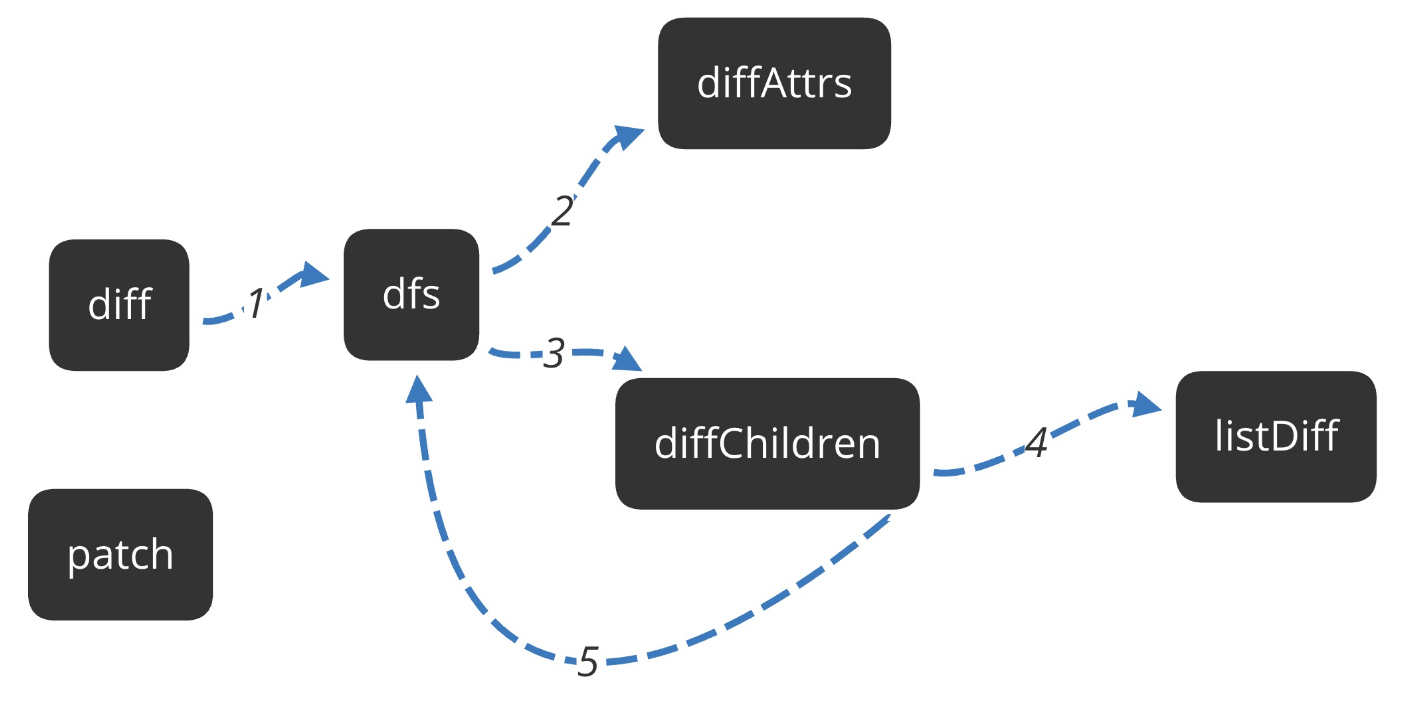
原来是遍历tree1；在遍历tree2；在进行排序；

现在是只比较同一层级（父节点相同的节点），tag不相同，则直接删掉重建，不在深度比较；tag和key两者都相同，则认为是相同节点，不在深度比较。

源码宏观理解：

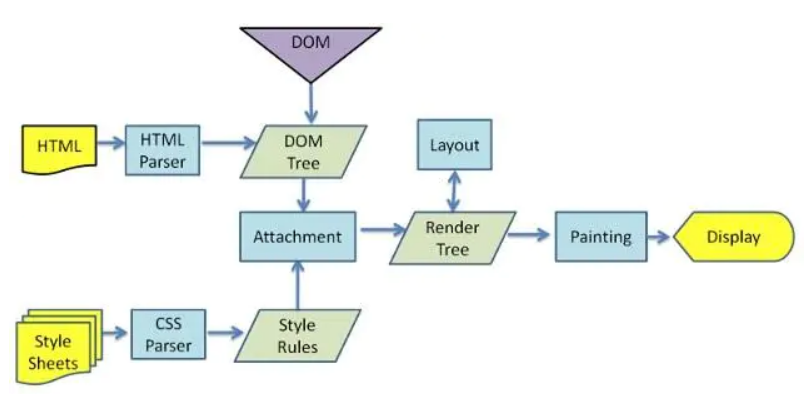


页面树和旧VNode节点是一一对应的，在新旧VNode在进行比较的时候不会立即对页面DOM进行修改，而是把比较的最终结果对页面DOM进行修改。



#### 18.介绍虚拟DOM？

所有浏览器的引擎工作流程都差不多，如上图大致分5步：创建DOM tree –> 创建Style Rules -> 构建Render tree -> 布局Layout –> 绘制Painting



原生api或jQuery操作DOM时，每个dom节点，浏览器会从头到尾执行上面的流程。操作DOM的代价是非常昂贵的。

虚拟DOM就是为了解决这个问题而设计的，虚拟DOM接收到更新DOM的动作不会立即操作DOM，而是将这些更新的diff内容保存到本地的一个js对象中，最终将这个js对象一次性attach到DOM树上，避免大量的无谓的计算量。

用JS模拟DOM结构：（三要素tag、props、children）

#### 19.NextTick是做什么的？

Vue是异步渲染，data改变以后，DOM不会立刻渲染，$nextTick会在DOM渲染之后被触发，用来获取最新的DOM节点。

只需要把对DOM进行操作的js代码放到this.$nextTick()中就可以。

什么时候需要使用Vue.nextTick():

1. 生命周期的created()钩子函数一定要放在Vue.nextTick()的回调函数中。
2. 当项目中需要在改变DOM后，基于新的DOM做些东西

Vue是异步执行dom更新的，一旦观察到数据变化，Vue就会开启一个队列，然后把在同一个事件循环 (event loop) 当中观察到数据变化的 watcher 推送进这个队列。如果这个watcher被触发多次，只会被推送到队列一次。这种缓冲行为可以有效的去掉重复数据造成的不必要的计算和DOm操作。而在下一个事件循环时，Vue会清空队列，并进行必要的DOM更新。

当你设置 vm.someData = 'new value'，DOM 并不会马上更新，而是在异步队列被清除，也就是下一个事件循环开始时执行更新时才会进行必要的DOM更新。如果此时你想要根据更新的 DOM 状态去做某些事情，就会出现问题。。为了在数据变化之后等待 Vue 完成更新 DOM ，可以在数据变化之后立即使用 Vue.nextTick(callback) 。这样回调函数在 DOM 更新完成后就会调用。

#### 20.Vue组件data为什么必须是函数？

在组件中data必须声明为返回一个初始数据对象的函数，因为组件可能被用来创建多个实例，通过提供data函数，每次创建一个新的实例后，我们能够调用data函数，从而返回初始数据的一个全新副本对象。

如果data是一个对象，则所有的实例将共享引用一个数据对象；写成函数就会有函数作用域的概念，是私有函数，只会用到当前组件中。

Vue实例中是对象，组件中是函数。

#### 21.Vue中的事件？

Vue的事件把他放在哪个元素下面，他就挂载在某个元素下面。

事件中如果不传入参数，直接通过event获取，这里event是原生的event对象。

传入自定义参数，$event

#### 22.keep-alive的理解？

Vue常见的性能优化，他能够缓存不活动的组件，频繁切换，不需要重复渲染。比较简单地使用v-show控制，复杂的使用keep-alive来做缓存。

首先在路由的meta属性中记录keepAlive，需要缓存则设置为true。

利用<keep-alive>组件进行包裹router-view组件，就可以实现缓存，切换的时候就不会被销毁，而是保存之前的状态，这时候所有的组件都会被缓存，可以根据需求缓存需要的组件。

1. include值只缓存需要的组件。
2. exclude值不缓存匹配的组件。

#### 23.单页面与多页面的比较？

单页面应用：刷新页面会请求一个HTML文件，切换页面的时候并不会发起新的HTML请求，只是页面内容发生了变化。页面跳转》》JS渲染。

优点：页面切换快（不需要去做HTML文件的请求，节约HTTP请求发送），

缺点：首屏时间稍慢，SEO差（搜索引擎只认识HTML内容，不认识JS内容）。

多页面应用：每次页面跳转，后台都会返回一个新的HTML文档。页面跳转》》返回HTML。

优点：首屏时间快，SEO效果好。

缺点：页面切换慢。

#### 25.异步组件

如果是大组件或者需要的时候才加载就可以设置为异步加载。

通过impor函数来进行异步加载。

components: {

BaseLevel: () => import ('url');

}

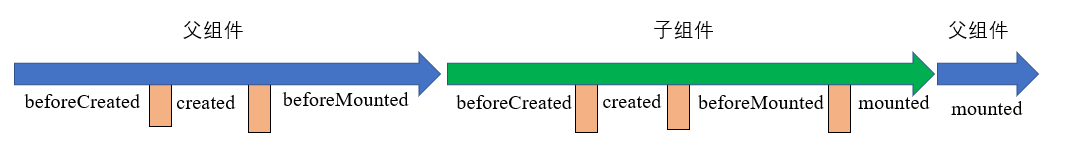
#### 26.单个组件和父子组件的生命周期

单页面生命周期过程，四个阶段，八个钩子：

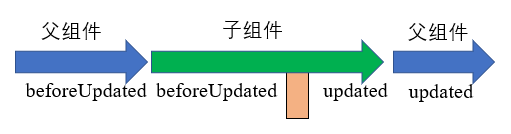
1. 创建（created）
2. 挂载（mounted）
3. 更新（updated）
4. 销毁（destroyed）

父子组件生命周期：

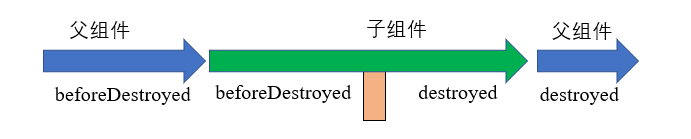
父组件先按顺序执行到beforeMounted，然后子组件开始执行beforeCreated到mounted。



创建初始化从父组件到子组件（先初始化父组件再初始化子组件）。



渲染是从内到外（子组件渲染完才能渲染父组件）。



销毁是从内到外（子组件销毁完才能销毁父组件）。

#### 27mixin的问题

mixin是把多个组件中有相同逻辑的给抽离出来，能高效的实现组件内容的复用。

在组件引入mixin后，mixin中的方法和属性也就并入到该组件中，可以直接使用，钩子函数会两个都被调用，mixin中的钩子函数首先执行，一旦出现冲突方法以组件为主。

问题：多mixin可能会造成命名冲突，mixin和组件可能会出现多对多的关系，复杂度较高。

#### 28.Vue中的this从哪里开始调用？

在Vue所有的声明周期钩子方法里使用的this，都指向调用他的Vue实例。