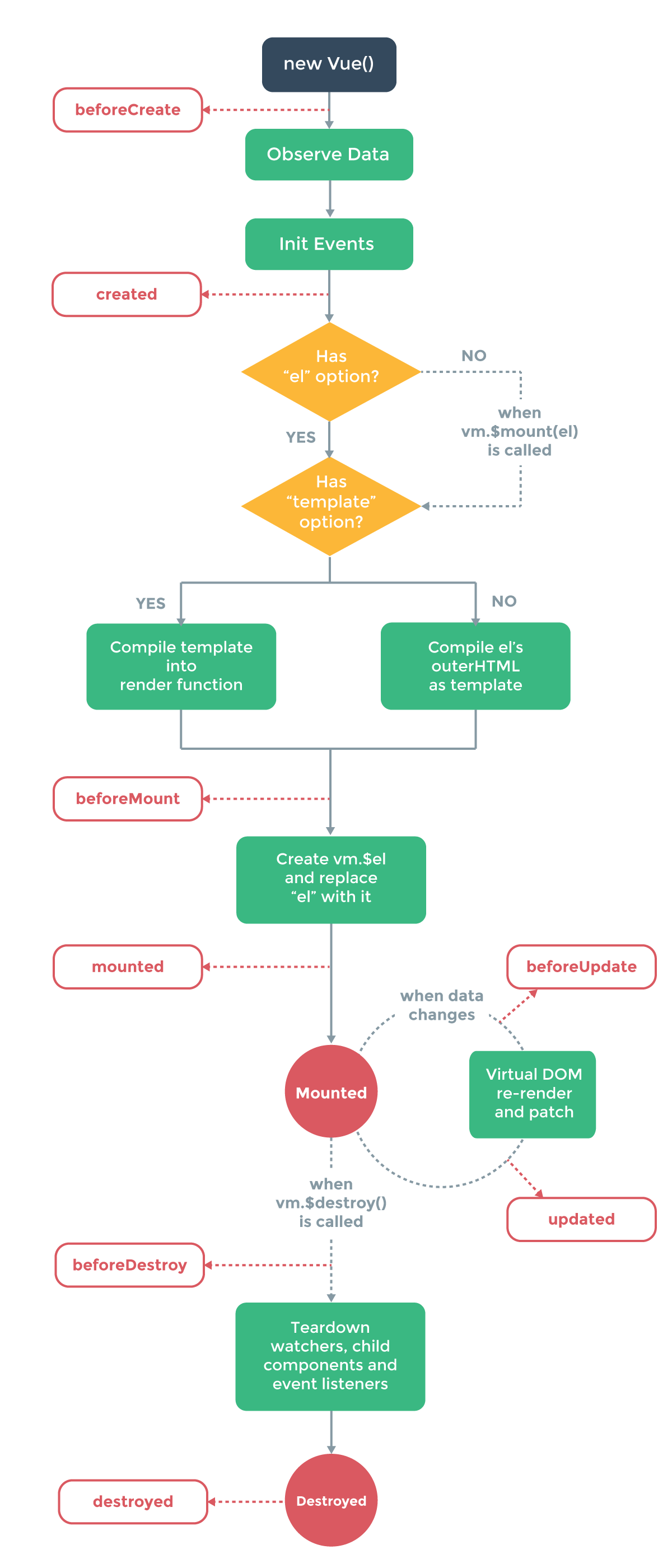
1. vue路由开启keep-alive（标签形式）时，钩子函数的调用情况

第一进入页面：created-> mounted-> activated，退出时触发deactivated。

再次进入页面：只触发activated。

注：activated和deactivated是只有在使用了keep-alive标签时才生效。

1. <span>Message: {{ msg }}</span>  
   绑定的数据对象上 msg 属性发生了改变，插值处的内容都会更新。

<span v-once>Message: {{ msg }}</span>  
 通过使用 v-once 指令，你也能执行一次性地插值，当数据改变时，插值处的内容不会更新。

1. <div v-html=“rawHtml"></div>  
   被插入的内容都会被当做 HTML —— 数据绑定会被忽略。只对可信内容使用 HTML 插值，绝不要对用户提供的内容插值，否则很容易导致 XSS 攻击。
2. v-on指令用于给监听DOM事件  
   <a v-on:click="doSomething"></a>  
   <!-- 缩写 -->  
   <a @click="doSomething"></a>
3. v-bind指令作用于元素的特性（attribute），例如：v-bind:class  
   <a v-bind:href="url"></a>  
   <!-- 缩写 -->  
   <a :href="url"></a>
4. v-model指令在表单元素上创建双向数据绑定  
   <input type="text" v-model="message"/>
5. 条件渲染  
   <h1 v-if=“type === 1">yesterday</h1>  
   <h1 v-else-if=“type === 2">today</h1>  
   <h1 v-else=“type === 3">tomorrow</h1>  
   条件不满足时，带有 v-if 的元素不会被渲染并保留在 DOM 中
6. 条件展示  
   <h1 v-show=“type === 1">yesterday</h1>  
   带有 v-show 的元素始终会被渲染并保留在 DOM 中
7. v-if & v-show  
   v-show 就简单得多——不管初始条件是什么，元素总是会被渲染，并且只是简单地基于 CSS 进行切换。  
   v-if 是“真正的”条件渲染，它会确保在切换过程中条件块内的事件监听器和子组件适当地被销毁和重建; 它也是惰性的：如果在初始渲染时条件为假，则什么也不做——直到条件第一次变为真时，才会开始渲染条件块。
8. v-for指令基于一个数组渲染一个列表  
   <div v-for="item of items"></div>  
   当 v-if 与 v-for 一起使用时，v-for 具有比 v-if 更高的优先级。
9. <p>Computed reversed message: "{{ reversedMessage }}"</p>  
   var vm = new Vue({  
    el: ‘#example',  
    data: {  
    message: 'Hello'  
    },  
   computed: {  
    reversedMessage: function () {  
    return this.message.split('').reverse().join('')  
    }  
    }  
   })  
   计算属性是基于它们的依赖进行缓存的。只有在它的相关依赖发生改变时才会重新求值。这就意味着只要 message 还没有发生改变，多次访问 reversedMessage 计算属性会立即返回之前的计算结果，而不必再次执行函数。  
   注：如果你不希望有缓存，请用 method 替代computed。
10. watch：在数据变化响应时，执行异步操作或开销较大的操作。计算属性可以当做值直接使用，watch只能用于执行某些操作，而不能当做值使用。
11. 因为 JavaScript 的限制，Vue.js 不能检测到数组变化。为了解决这个问题 ，Vue.js 扩展了观察数组，为它添加了一个 $set() 方法，通过索引设置数组元素并触发视图更新。
12. Vue.js 不能检测到对象属性的添加或删除。因为 Vue.js 在初始化实例时将属性转为 getter/setter，所以属性必须在 data 对象上才能让 Vue.js 转换它，才能让它是响应的。
13. Vue.nextTick：在 DOM 更新循环结束之后执行延迟回调，获取更新后的 DOM。
14. 自定义组件中的监听函数，需要用自定义属性包一层，不然会没有效果。
15. Vue 异步执行 DOM 更新。只要观察到数据变化，Vue 将开启一个队列，并缓冲在同一事件循环中发生的所有数据改变。如果同一个 watcher 被多次触发，只会一次推入到队列中。这种在缓冲时去除重复数据可以避免不必要的计算和 DOM 操作。
16. vue双向数据绑定实现原理：<https://segmentfault.com/a/1190000006599500>  
    Observer：将data属性加上getter/setter  
    Object.defineProperty()：劫持getter/setter，并在get()中注册订阅者（即watcher对象），在set()中将数据变化通知订阅者，订阅者调用update()方法。  
    Compile：解析编译DOM节点，监听数据并绑定更新函数。更新函数通过watcher获取数据的更新通知，从而更新视图。  
    Watcher：实例化时触发data的getter，将自己添加到data的订阅器中，在数据变化时获得dep的通知，调用update方法，触发Compile更新视图。