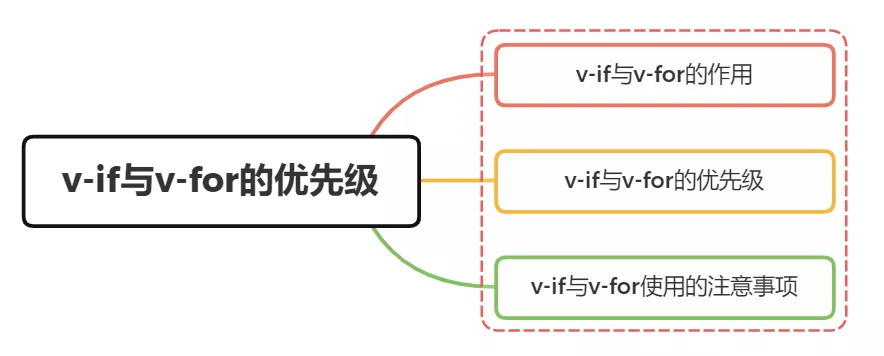
# 面试官：v-if和v-for的优先级是什么？



## 一、作用

v-if 指令用于条件性地渲染一块内容。这块内容只会在指令的表达式返回 true值的时候被渲染

v-for 指令基于一个数组来渲染一个列表。v-for 指令需要使用 item in items 形式的特殊语法，其中 items 是源数据数组或者对象，而 item 则是被迭代的数组元素的别名

在 v-for 的时候，建议设置key值，并且保证每个key值是独一无二的，这便于diff算法进行优化

两者在用法上

<Modal v-if="isShow" />  
  
<li v-for="item in items" :key="item.id">  
 {{ item.label }}  
</li>

## 二、优先级

v-if与v-for都是vue模板系统中的指令

在vue模板编译的时候，会将指令系统转化成可执行的render函数

### 示例

编写一个p标签，同时使用v-if与 v-for

<div id="app">  
 <p v-if="isShow" v-for="item in items">  
 {{ item.title }}  
 </p>  
</div>

创建vue实例，存放isShow与items数据

const app = new Vue({  
 el: "#app",  
 data() {  
 return {  
 items: [  
 { title: "foo" },  
 { title: "baz" }]  
 }  
 },  
 computed: {  
 isShow() {  
 return this.items && this.items.length > 0  
 }  
 }  
})

模板指令的代码都会生成在render函数中，通过app.$options.render就能得到渲染函数

ƒ anonymous() {  
 with (this) { return   
 \_c('div', { attrs: { "id": "app" } },   
 \_l((items), function (item)   
 { return (isShow) ? \_c('p', [\_v("\n" + \_s(item.title) + "\n")]) : \_e() }), 0) }  
}

\_l是vue的列表渲染函数，函数内部都会进行一次if判断

初步得到结论：v-for优先级是比v-if高

再将v-for与v-if置于不同标签

<div id="app">  
 <template v-if="isShow">  
 <p v-for="item in items">{{item.title}}</p>  
 </template>  
</div>

再输出下render函数

ƒ anonymous() {  
 with(this){return   
 \_c('div',{attrs:{"id":"app"}},  
 [(isShow)?[\_v("\n"),  
 \_l((items),function(item){return \_c('p',[\_v(\_s(item.title))])})]:\_e()],2)}  
}

这时候我们可以看到，v-for与v-if作用在不同标签时候，是先进行判断，再进行列表的渲染

我们再在查看下vue源码

源码位置：\vue-dev\src\compiler\codegen\index.js

export function genElement (el: ASTElement, state: CodegenState): string {  
 if (el.parent) {  
 el.pre = el.pre || el.parent.pre  
 }  
 if (el.staticRoot && !el.staticProcessed) {  
 return genStatic(el, state)  
 } else if (el.once && !el.onceProcessed) {  
 return genOnce(el, state)  
 } else if (el.for && !el.forProcessed) {  
 return genFor(el, state)  
 } else if (el.if && !el.ifProcessed) {  
 return genIf(el, state)  
 } else if (el.tag === 'template' && !el.slotTarget && !state.pre) {  
 return genChildren(el, state) || 'void 0'  
 } else if (el.tag === 'slot') {  
 return genSlot(el, state)  
 } else {  
 // component or element  
 ...  
}

在进行if判断的时候，v-for是比v-if先进行判断

最终结论：v-for优先级比v-if高

## 三、注意事项

1. 永远不要把 v-if 和 v-for 同时用在同一个元素上，带来性能方面的浪费（每次渲染都会先循环再进行条件判断）
2. 如果避免出现这种情况，则在外层嵌套template（页面渲染不生成dom节点），在这一层进行v-if判断，然后在内部进行v-for循环

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

1. 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  
 })  
 }  
}