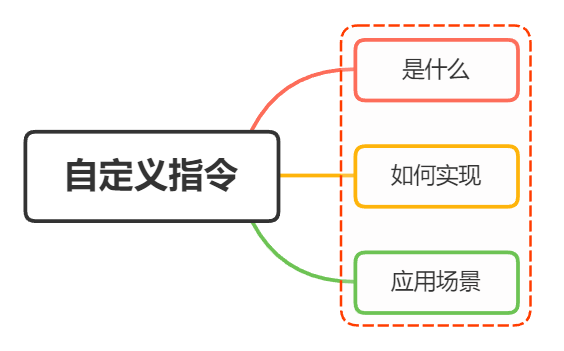
# 面试官：你有写过自定义指令吗？自定义指令的应用场景有哪些？



## 一、什么是指令

开始之前我们先学习一下指令系统这个词

**指令系统**是计算机硬件的语言系统，也叫机器语言，它是系统程序员看到的计算机的主要属性。因此指令系统表征了计算机的基本功能决定了机器所要求的能力

在vue中提供了一套为数据驱动视图更为方便的操作，这些操作被称为指令系统

我们看到的v-开头的行内属性，都是指令，不同的指令可以完成或实现不同的功能

除了核心功能默认内置的指令 (v-model 和 v-show)，Vue 也允许注册自定义指令

指令使用的几种方式：

//会实例化一个指令，但这个指令没有参数   
`v-xxx`  
  
// -- 将值传到指令中  
`v-xxx="value"`   
  
// -- 将字符串传入到指令中，如`v-html="'<p>内容</p>'"`  
`v-xxx="'string'"`   
  
// -- 传参数（`arg`），如`v-bind:class="className"`  
`v-xxx:arg="value"`   
  
// -- 使用修饰符（`modifier`）  
`v-xxx:arg.modifier="value"`

### 二、如何实现

注册一个自定义指令有全局注册与局部注册

全局注册主要是通过Vue.directive方法进行注册

Vue.directive第一个参数是指令的名字（不需要写上v-前缀），第二个参数可以是对象数据，也可以是一个指令函数

// 注册一个全局自定义指令 `v-focus`  
Vue.directive('focus', {  
 // 当被绑定的元素插入到 DOM 中时……  
 inserted: function (el) {  
 // 聚焦元素  
 el.focus() // 页面加载完成之后自动让输入框获取到焦点的小功能  
 }  
})

局部注册通过在组件options选项中设置directive属性

directives: {  
 focus: {  
 // 指令的定义  
 inserted: function (el) {  
 el.focus() // 页面加载完成之后自动让输入框获取到焦点的小功能  
 }  
 }  
}

然后你可以在模板中任何元素上使用新的 v-focus property，如下：

<input v-focus />

自定义指令也像组件那样存在钩子函数：

* bind：只调用一次，指令第一次绑定到元素时调用。在这里可以进行一次性的初始化设置
* inserted：被绑定元素插入父节点时调用 (仅保证父节点存在，但不一定已被插入文档中)
* update：所在组件的 VNode 更新时调用，但是可能发生在其子 VNode 更新之前。指令的值可能发生了改变，也可能没有。但是你可以通过比较更新前后的值来忽略不必要的模板更新
* componentUpdated：指令所在组件的 VNode 及其子 VNode 全部更新后调用
* unbind：只调用一次，指令与元素解绑时调用

所有的钩子函数的参数都有以下：

* 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 property：
  + name：指令名，不包括 v- 前缀。
  + value：指令的绑定值，例如：v-my-directive="1 + 1" 中，绑定值为 2。
  + oldValue：指令绑定的前一个值，仅在 update 和 componentUpdated 钩子中可用。无论值是否改变都可用。
  + expression：字符串形式的指令表达式。例如 v-my-directive="1 + 1" 中，表达式为 "1 + 1"。
  + arg：传给指令的参数，可选。例如 v-my-directive:foo 中，参数为 "foo"。
  + modifiers：一个包含修饰符的对象。例如：v-my-directive.foo.bar 中，修饰符对象为 { foo: true, bar: true }
* vnode：Vue 编译生成的虚拟节点
* oldVnode：上一个虚拟节点，仅在 update 和 componentUpdated 钩子中可用

除了 el 之外，其它参数都应该是只读的，切勿进行修改。如果需要在钩子之间共享数据，建议通过元素的 dataset 来进行

举个例子：

<div v-demo="{ color: 'white', text: 'hello!' }"></div>  
<script>  
 Vue.directive('demo', function (el, binding) {  
 console.log(binding.value.color) // "white"  
 console.log(binding.value.text) // "hello!"  
 })  
</script>

## 三、应用场景

使用自定义指令可以满足我们日常一些场景，这里给出几个自定义指令的案例：

* 表单防止重复提交
* 图片懒加载
* 一键 Copy的功能

### 表单防止重复提交

表单防止重复提交这种情况设置一个v-throttle自定义指令来实现

举个例子：

// 1.设置v-throttle自定义指令  
Vue.directive('throttle', {  
 bind: (el, binding) => {  
 let throttleTime = binding.value; // 节流时间  
 if (!throttleTime) { // 用户若不设置节流时间，则默认2s  
 throttleTime = 2000;  
 }  
 let cbFun;  
 el.addEventListener('click', event => {  
 if (!cbFun) { // 第一次执行  
 cbFun = setTimeout(() => {  
 cbFun = null;  
 }, throttleTime);  
 } else {  
 event && event.stopImmediatePropagation();  
 }  
 }, true);  
 },  
});  
// 2.为button标签设置v-throttle自定义指令  
<button @click="sayHello" v-throttle>提交</button>

### 图片懒加载

设置一个v-lazy自定义指令完成图片懒加载

const LazyLoad = {  
 // install方法  
 install(Vue,options){  
 // 代替图片的loading图  
 let defaultSrc = options.default;  
 Vue.directive('lazy',{  
 bind(el,binding){  
 LazyLoad.init(el,binding.value,defaultSrc);  
 },  
 inserted(el){  
 // 兼容处理  
 if('IntersectionObserver' in window){  
 LazyLoad.observe(el);  
 }else{  
 LazyLoad.listenerScroll(el);  
 }  
   
 },  
 })  
 },  
 // 初始化  
 init(el,val,def){  
 // data-src 储存真实src  
 el.setAttribute('data-src',val);  
 // 设置src为loading图  
 el.setAttribute('src',def);  
 },  
 // 利用IntersectionObserver监听el  
 observe(el){  
 let io = new IntersectionObserver(entries => {  
 let realSrc = el.dataset.src;  
 if(entries[0].isIntersecting){  
 if(realSrc){  
 el.src = realSrc;  
 el.removeAttribute('data-src');  
 }  
 }  
 });  
 io.observe(el);  
 },  
 // 监听scroll事件  
 listenerScroll(el){  
 let handler = LazyLoad.throttle(LazyLoad.load,300);  
 LazyLoad.load(el);  
 window.addEventListener('scroll',() => {  
 handler(el);  
 });  
 },  
 // 加载真实图片  
 load(el){  
 let windowHeight = document.documentElement.clientHeight  
 let elTop = el.getBoundingClientRect().top;  
 let elBtm = el.getBoundingClientRect().bottom;  
 let realSrc = el.dataset.src;  
 if(elTop - windowHeight<0&&elBtm > 0){  
 if(realSrc){  
 el.src = realSrc;  
 el.removeAttribute('data-src');  
 }  
 }  
 },  
 // 节流  
 throttle(fn,delay){  
 let timer;   
 let prevTime;  
 return function(...args){  
 let currTime = Date.now();  
 let context = this;  
 if(!prevTime) prevTime = currTime;  
 clearTimeout(timer);  
   
 if(currTime - prevTime > delay){  
 prevTime = currTime;  
 fn.apply(context,args);  
 clearTimeout(timer);  
 return;  
 }  
  
 timer = setTimeout(function(){  
 prevTime = Date.now();  
 timer = null;  
 fn.apply(context,args);  
 },delay);  
 }  
 }  
  
}  
export default LazyLoad;

### 一键 Copy的功能

import { Message } from 'ant-design-vue';  
  
const vCopy = { //  
 /\*  
 bind 钩子函数，第一次绑定时调用，可以在这里做初始化设置  
 el: 作用的 dom 对象  
 value: 传给指令的值，也就是我们要 copy 的值  
 \*/  
 bind(el, { value }) {  
 el.$value = value; // 用一个全局属性来存传进来的值，因为这个值在别的钩子函数里还会用到  
 el.handler = () => {  
 if (!el.$value) {  
 // 值为空的时候，给出提示，我这里的提示是用的 ant-design-vue 的提示，你们随意  
 Message.warning('无复制内容');  
 return;  
 }  
 // 动态创建 textarea 标签  
 const textarea = document.createElement('textarea');  
 // 将该 textarea 设为 readonly 防止 iOS 下自动唤起键盘，同时将 textarea 移出可视区域  
 textarea.readOnly = 'readonly';  
 textarea.style.position = 'absolute';  
 textarea.style.left = '-9999px';  
 // 将要 copy 的值赋给 textarea 标签的 value 属性  
 textarea.value = el.$value;  
 // 将 textarea 插入到 body 中  
 document.body.appendChild(textarea);  
 // 选中值并复制  
 textarea.select();  
 // textarea.setSelectionRange(0, textarea.value.length);  
 const result = document.execCommand('Copy');  
 if (result) {  
 Message.success('复制成功');  
 }  
 document.body.removeChild(textarea);  
 };  
 // 绑定点击事件，就是所谓的一键 copy 啦  
 el.addEventListener('click', el.handler);  
 },  
 // 当传进来的值更新的时候触发  
 componentUpdated(el, { value }) {  
 el.$value = value;  
 },  
 // 指令与元素解绑的时候，移除事件绑定  
 unbind(el) {  
 el.removeEventListener('click', el.handler);  
 },  
};  
  
export default vCopy;

关于自定义指令还有很多应用场景，如：拖拽指令、页面水印、权限校验等等应用场景

## 参考文献

* https://vue3js.cn/docs/zh
* https://juejin.cn/post/6844904197448531975#heading-5
* https://www.cnblogs.com/chenwenhao/p/11924161.html#\_label2