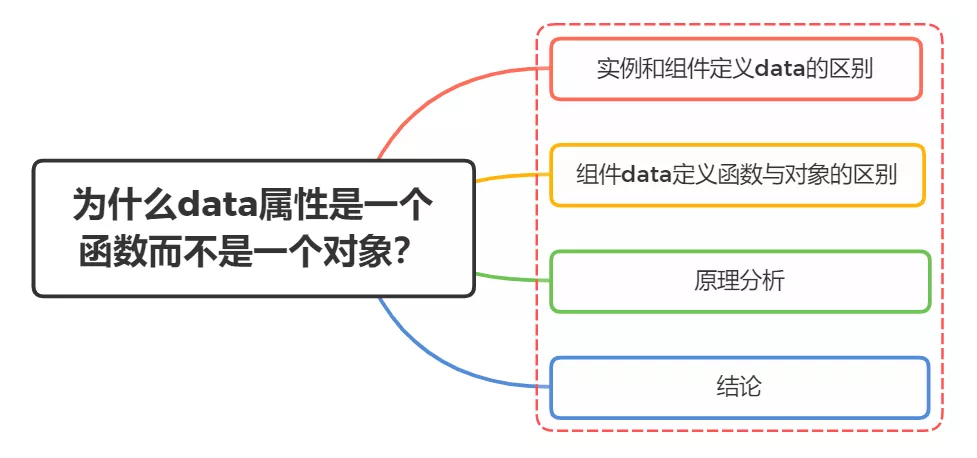
# 面试官：为什么data属性是一个函数而不是一个对象？



## 一、实例和组件定义data的区别

vue实例的时候定义data属性既可以是一个对象，也可以是一个函数

const app = new Vue({  
 el:"#app",  
 // 对象格式  
 data:{  
 foo:"foo"  
 },  
 // 函数格式  
 data(){  
 return {  
 foo:"foo"  
 }  
 }  
})

组件中定义data属性，只能是一个函数

如果为组件data直接定义为一个对象

Vue.component('component1',{  
 template:`<div>组件</div>`,  
 data:{  
 foo:"foo"  
 }  
})

则会得到警告信息



警告说明：返回的data应该是一个函数在每一个组件实例中

## 二、组件data定义函数与对象的区别

上面讲到组件data必须是一个函数，不知道大家有没有思考过这是为什么呢？

在我们定义好一个组件的时候，vue最终都会通过Vue.extend()构成组件实例

这里我们模仿组件构造函数，定义data属性，采用对象的形式

function Component(){  
   
}  
Component.prototype.data = {  
 count : 0  
}

创建两个组件实例

const componentA = new Component()  
const componentB = new Component()

修改componentA组件data属性的值，componentB中的值也发生了改变

console.log(componentB.data.count) // 0  
componentA.data.count = 1  
console.log(componentB.data.count) // 1

产生这样的原因这是两者共用了同一个内存地址，componentA修改的内容，同样对componentB产生了影响

如果我们采用函数的形式，则不会出现这种情况（函数返回的对象内存地址并不相同）

function Component(){  
 this.data = this.data()  
}  
Component.prototype.data = function (){  
 return {  
 count : 0  
 }  
}

修改componentA组件data属性的值，componentB中的值不受影响

console.log(componentB.data.count) // 0  
componentA.data.count = 1  
console.log(componentB.data.count) // 0

vue组件可能会有很多个实例，采用函数返回一个全新data形式，使每个实例对象的数据不会受到其他实例对象数据的污染

## 三、原理分析

首先可以看看vue初始化data的代码，data的定义可以是函数也可以是对象

源码位置：/vue-dev/src/core/instance/state.js

function initData (vm: Component) {  
 let data = vm.$options.data  
 data = vm.\_data = typeof data === 'function'  
 ? getData(data, vm)  
 : data || {}  
 ...  
}

data既能是object也能是function，那为什么还会出现上文警告呢？

别急，继续看下文

组件在创建的时候，会进行选项的合并

源码位置：/vue-dev/src/core/util/options.js

自定义组件会进入mergeOptions进行选项合并

Vue.prototype.\_init = function (options?: Object) {  
 ...  
 // merge options  
 if (options && options.\_isComponent) {  
 // optimize internal component instantiation  
 // since dynamic options merging is pretty slow, and none of the  
 // internal component options needs special treatment.  
 initInternalComponent(vm, options)  
 } else {  
 vm.$options = mergeOptions(  
 resolveConstructorOptions(vm.constructor),  
 options || {},  
 vm  
 )  
 }  
 ...  
 }

定义data会进行数据校验

源码位置：/vue-dev/src/core/instance/init.js

这时候vm实例为undefined，进入if判断，若data类型不是function，则出现警告提示

strats.data = function (  
 parentVal: any,  
 childVal: any,  
 vm?: Component  
): ?Function {  
 if (!vm) {  
 if (childVal && typeof childVal !== "function") {  
 process.env.NODE\_ENV !== "production" &&  
 warn(  
 'The "data" option should be a function ' +  
 "that returns a per-instance value in component " +  
 "definitions.",  
 vm  
 );  
  
 return parentVal;  
 }  
 return mergeDataOrFn(parentVal, childVal);  
 }  
 return mergeDataOrFn(parentVal, childVal, vm);  
};

### 四、结论

* 根实例对象data可以是对象也可以是函数（根实例是单例），不会产生数据污染情况
* 组件实例对象data必须为函数，目的是为了防止多个组件实例对象之间共用一个data，产生数据污染。采用函数的形式，initData时会将其作为工厂函数都会返回全新data对象