* [首页](https://km.woa.com/?kmref=km_header)
* [发现](https://km.woa.com/discovery?kmref=km_header)
* [悦读](https://km.woa.com/read?kmref=km_header)
* [乐问](https://km.woa.com/q?kmref=km_header)
* [直播](https://tencent.lexiangla.com/lives?company_from=tencent&kmref=km_header)
* [应用](javascript:void(0))
* [我的K吧](javascript:void(0))
* [[](https://km.woa.com/user/rickyrqzhao)](https://km.woa.com/user/rickyrqzhao)

窗体顶端



窗体底端

[](https://km.woa.com/group/vasknowlege)[互娱增值服务部 - 技术&服务藏经阁](https://km.woa.com/group/34294)

[加入K吧](javascript:void(0);)

* [首页](https://km.woa.com/group/34294)
* [文章](https://km.woa.com/group/34294/articles)
* [文集](https://km.woa.com/group/34294/knowledges)
* [期刊](https://km.woa.com/group/34294/journals)
* [潘多拉](http://km.oa.com/group/34294/knowledge/4447)
* [DataMore](http://km.oa.com/group/34294/knowledge/4453)
* [挖掘推荐](http://km.oa.com/group/34294/knowledge/4663)
* [奇点ODP](http://km.oa.com/group/34294/knowledge/4988)
* [学术论文](http://km.oa.com/group/34294/knowledge/5165)

目录

简介

1. v-for 遍历必须为 item 添加 key，且避免同时使用 v-if

2. 长列表性能优化

3. vue组件中的data是函数而不是对象

4. Vue钩子函数之钩子事件hookEvent,监听组件简化代码

5. 组件懒加载

6. 非响应式数据

7. 不要将所有的数据都放到data中

8. v-for元素绑定事件代理

9. 函数式组件

10. provide和inject组件通信

[[](https://km.woa.com/base/headlines)](https://km.woa.com/base/headlines" \t "_blank)

【道聚城】Vue跨平台性能优化十法

* [bartonwang](https://km.woa.com/user/bartonwang)

* 2021年08月31日 10:37

* 浏览(1399)

* [收藏(80)](javascript:void(0);)

* [评论(21)](javascript:void(0);)

* [分享](javascript:void(0);)

| 导语 Vue是一套用于构建用户界面的渐进式的JavaScript框架。具有体积小，更高的运行效率，双向数据绑定，生态丰富、学习成本低等优点，所以Vue也被广泛用在移动端跨平台框架上。

IEG增值服务部 - 技术藏经阁：秉承IEG增值服务部核心的创新向上的理念；利用KM知识分享开放平台来沉淀和输出部门内的核心技术相关能力。构建与公司团队共同交流、共同成长的开放性知识K吧。 更多文章请点击：<http://km.oa.com/group/vasknowlege>

**简介**

Vue 框架通过数据双向绑定和虚拟 DOM 技术，帮我们处理了前端开发中最脏最累的 DOM 操作部分， 我们不再需要去考虑如何操作 DOM 以及如何最高效地操作 DOM，但是我们仍然需要去关注 Vue在跨平台项目性能方面的优化，使项目具有更高效的性能、更好的用户体验。

**1. v-for 遍历必须为 item 添加 key，且避免同时使用 v-if**

在列表数据进行遍历渲染时，需要为每一项 item 设置唯一 key 值，方便 Vue.js 内部机制精准找到该条列表数据。当 state 更新时，新的状态值和旧的状态值对比，较快地定位到 diff 。

我们在使用的使用经常会使用index(即数组的下标)来作为key,但其实这是不推荐的一种使用方法;

举个例子：

var list = [

{

id: 1,

name: 'test1',

},

{

id: 2,

name: 'test2',

},

{

id: 3,

name: 'test3',

},

]

<div v-for="(item, index) in list" :key="index" >{{item.name}}</div>

在最后一条数据后再加一条数据

var list = [

{

id: 1,

name: 'test1',

},

{

id: 2,

name: 'test2',

},

{

id: 3,

name: 'test3',

},

{

id: 4,

name: '我是在最后添加的一条数据',

},

]

此时前三条数据直接复用之前的,新渲染最后一条数据,此时用index作为key,没有任何问题;

在中间插入一条数据

var list = [

{

id: 1,

name: 'test1',

},

{

id: 4,

name: '我是插队的那条数据',

},

{

id: 2,

name: 'test2',

},

{

id: 3,

name: 'test3',

},

]

此时更新渲染数据,通过index定义的key去进行前后数据的对比,发现

之前的数据 之后的数据

key: 0 index: 0 name: test1 key: 0 index: 0 name: test1

key: 1 index: 1 name: test2 key: 1 index: 1 name: 我是插队的那条数据

key: 2 index: 2 name: test3 key: 2 index: 2 name: test2

key: 3 index: 3 name: test3

通过上面清晰的对比,发现除了第一个数据可以复用之前的之外,另外三条数据都需要重新渲染;

是不是很惊奇,我明明只是插入了一条数据,怎么三条数据都要重新渲染?而我想要的只是新增的那一条数据新渲染出来就行了

最好的办法是使用数组中不会变化的那一项作为key值,对应到项目中,即每条数据都有一个唯一的id,来标识这条数据的唯一性;使用id作为key值,我们再来对比一下向中间插入一条数据,此时会怎么去渲染

之前的数据 之后的数据

key: 1 id: 1 index: 0 name: test1 key: 1 id: 1 index: 0 name: test1

key: 2 id: 2 index: 1 name: test2 key: 4 id: 4 index: 1 name: 我是插队的那条数据

key: 3 id: 3 index: 2 name: test3 key: 2 id: 2 index: 2 name: test2

key: 3 id: 3 index: 3 name: test3

现在对比发现只有一条数据变化了,就是id为4的那条数据,因此只要新渲染这一条数据就可以了,其他都是就复用之前的;

总结 ：所以一句话，key的作用主要是为了高效的更新虚拟DOM。另外vue中在使用相同标签名元素的过渡切换时，也会使用到key属性，其目的也是为了让vue可以区分它们，否则vue只会替换其内部属性而不会触发过渡效果。

v-for 遍历避免同时使用 v-if

v-for 比 v-if 优先级高，如果每一次都需要遍历整个数组，将会影响速度，尤其是当之需要渲染很小一部分的时候，必要情况下应该替换成 computed 属性。

**2. 长列表性能优化**

Vue 会通过 Object.defineProperty 对数据进行劫持，来实现视图响应数据的变化，然而有些时候我们的组件就是纯粹的数据展示，不会有任何改变，我们就不需要 Vue 来劫持我们的数据，在大量数据展示的情况下，这能够很明显的减少组件初始化的时间，那如何禁止 Vue 劫持我们的数据呢？可以通过 Object.freeze 方法来冻结一个对象，一旦被冻结的对象就再也不能被修改了。

export default {

data: () => ({

users: {}

}),

async created() {

const users = await axios.get("/api/users");

this.users = Object.freeze(users);

}

};

**3. vue组件中的data是函数而不是对象**

export default {

data() {

// data是一个函数,data: function() {}的简写

return {

// 页面要初始化的数据

name: 'bartonwang',

};

},

};

而非:如下所示

export default {

data: {

// data是一个对象

name: 'bartonwang',

},

};

当一个组件被定义, 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,复用在多个页面

如果 data是一个纯碎的对象,则所有的实例将共享引用同一份 data 数据对象,无论在哪个组件实例中修改 data,都会影响到所有的组件实例

如果 data是函数,每次创建一个新实例后,调用 data函数,从而返回初始数据的一个全新副本数据对象

这样每复用一次组件,会返回一份新的 data数据,类似于给每个组件实例创建一个私有的数据空间,让各个组件的实例各自独立,互不影响,保持低耦合

**4. Vue钩子函数之钩子事件hookEvent,监听组件简化代码**

用法：

1. 通过 $on(eventName, eventHandler) 侦听一个事件
2. 通过 $once(eventName,eventHandler) 一次性侦听一个事件
3. 通过 $off(eventName, eventHandler) 停止侦听一个事件

通常实现一个定时器的调用与销毁我可能会以以下方式实现：

export default{

data(){

timer:null // 需要创建实例

},

mounted(){

this.timer = setInterval(()=>{

//具体执行内容

console.log('1');

},1000);

}

beforeDestory(){

clearInterval(this.timer);

this.timer = null;

}

}

这种方法存在的问题是：

vue实例中需要有这个定时器的实例，感觉有点多余。创建的定时器代码和销毁定时器的代码没有放在一起，不容易维护，通常很容易忘记去清理这个定时器。

使用$on(‘hook:’)监听beforeDestory生命周期可以避免该问题，

并且因为只需要监听一次，所以使用$once进行注册监听。

export default{

methods:{

fn(){

const timer = setInterval(()=>{

console.log('1');

},1000);

this.$once('hook:beforeDestory',()=>{ // 监听一次即可

clearInterval(timer);

timer = null;

})

}

}

}

**5. 组件懒加载**

在单页应用中，如果没有应用懒加载，运用webpack打包后的文件将会异常的大，造成进入首页时，需要加载的内容过多，延时过长，不利于用户体验，而运用懒加载则可以将页面进行划分，需要的时候加载页面，可以有效的分担首页所承担的加载压力，减少首页加载用时。

Vue.js 2.0 组件级懒加载方案

* 支持 组件可见或即将可见时懒加载
* 支持 组件延时加载
* 支持 加载真实组件前展示骨架组件，提高用户体验
* 支持 真实组件代码分包异步加载

安装

npm install @xunlei/vue-lazy-component

在组件中实现局部注册组件

import { component as VueLazyComponent } from '@xunlei/vue-lazy-component'

export default {

components: {

'vue-lazy-component': VueLazyComponent

}

}

需要懒加载的组件将其包裹在vue-lazy-component中，slot值为 skeleton 指的是在懒加载过程中显示的加载状态组件

<vue-lazy-component :timeout="5000" tagName="div">

<child1 slot="skeleton" />

<child2 />

<child3 />

<child4 />

<child5 />

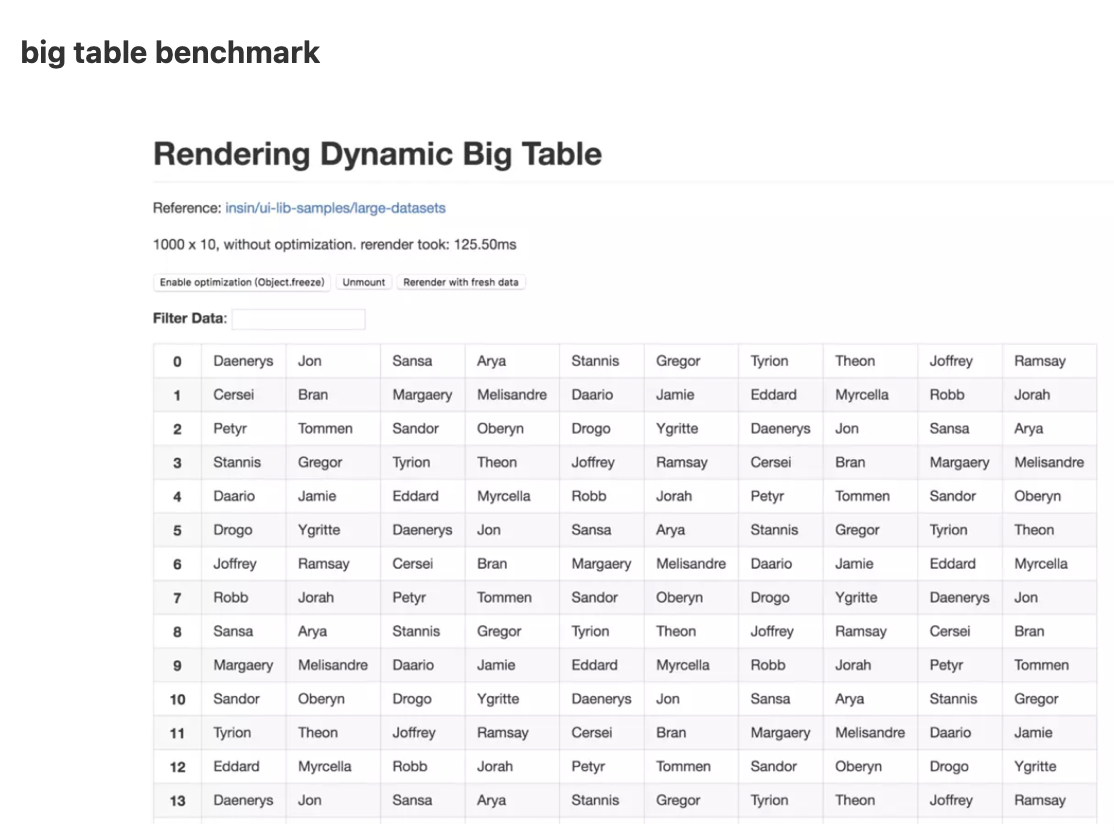
</vue-lazy-component>

**6. 非响应式数据**

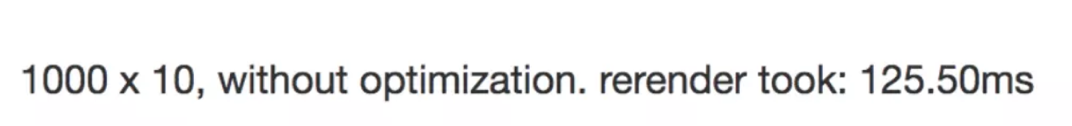
初始化时，vue会对data做getter、setter改造。在 Vue 的文档中介绍数据绑定和响应时，特意标注了对于经过 Object.freeze() 方法的对象无法进行更新响应。

性能提升对比

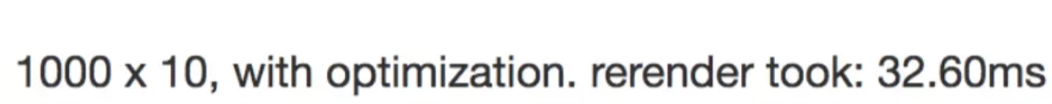
在基于 Vue 的一个 big table benchmark里，可以看到在渲染一个一个 1000 x 10 的表格的时候，开启 Object.freeze()前后重新渲染的对比。



**开启优化之前**



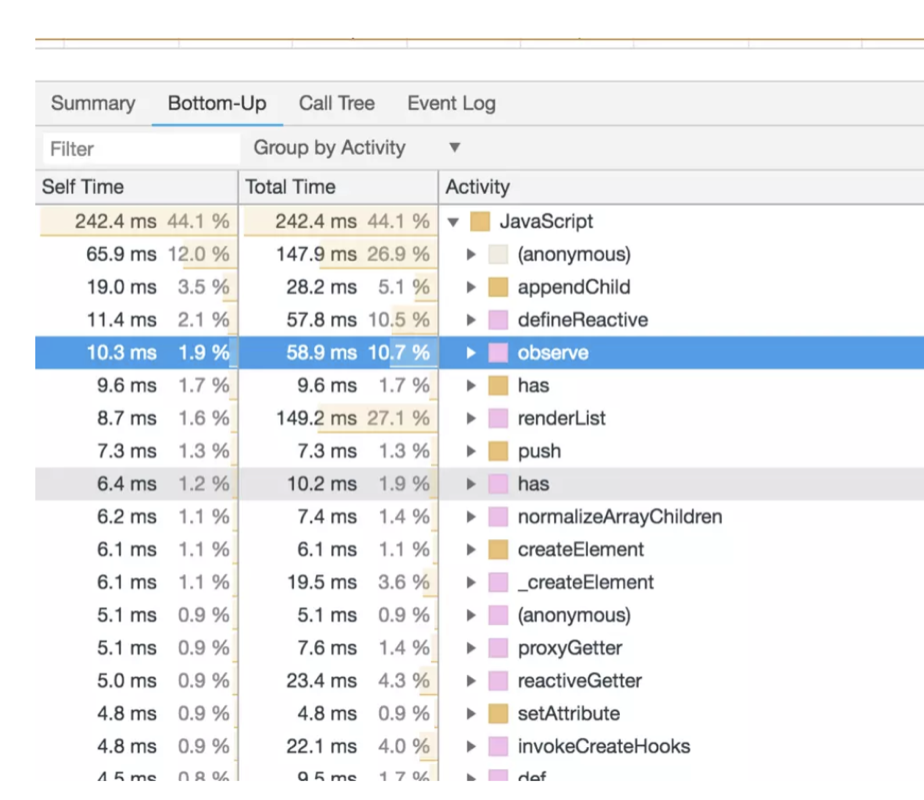
**开启优化之后**



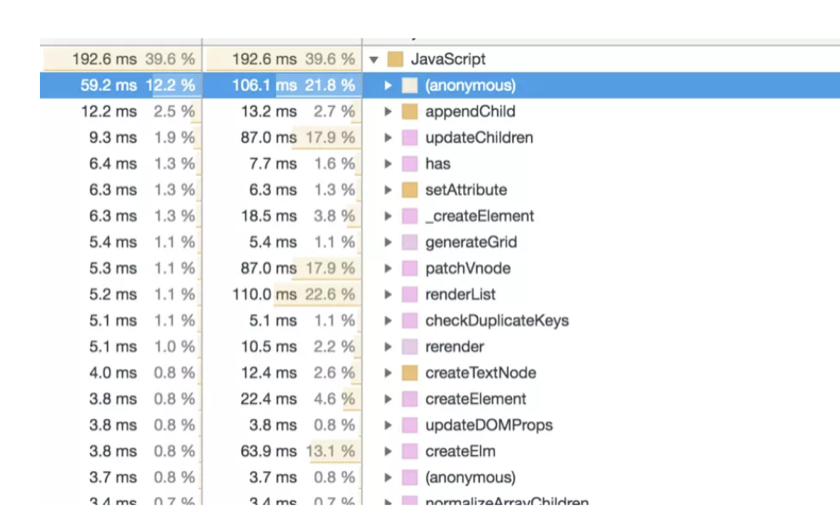
在这个例子里,使用了 Object.freeze()比不使用快了 4 倍

为什么Object.freeze()的性能会更好

**不使用Object.freeze()的CPU开销**



**使用Object.freeze()的CPU开销**



对比可以看出，使用了Object.freeze()之后，减少了 observer 的开销。

**7. 不要将所有的数据都放到data中**

data 中的数据都会增加 getter 和 setter，又会收集watcher，这样还占内存。不需要响应式的数据我们可以定义在实例上。



**8. v-for元素绑定事件代理**

事件代理作用主要是 2 个

1. 将事件处理程序代理到父节点，减少内存占用率
2. 动态生成子节点时能自动绑定事件处理程序到父节点

1、不使用事件代理，每个 span 节点绑定一个 click 事件，并指向同一个事件处理程序

<div>

<span

v-for="(item,index) of 100000"

:key="index"

@click="handleClick">

{{item}}

</span>

</div>

2、不使用事件代理，每个 span 节点绑定一个 click 事件，并指向不同的事件处理程序

<div>

<span

v-for="(item,index) of 100000"

:key="index"

@click="function () {}">

{{item}}

</span>

</div>

3、使用事件代理

<div @click="handleClick">

<span

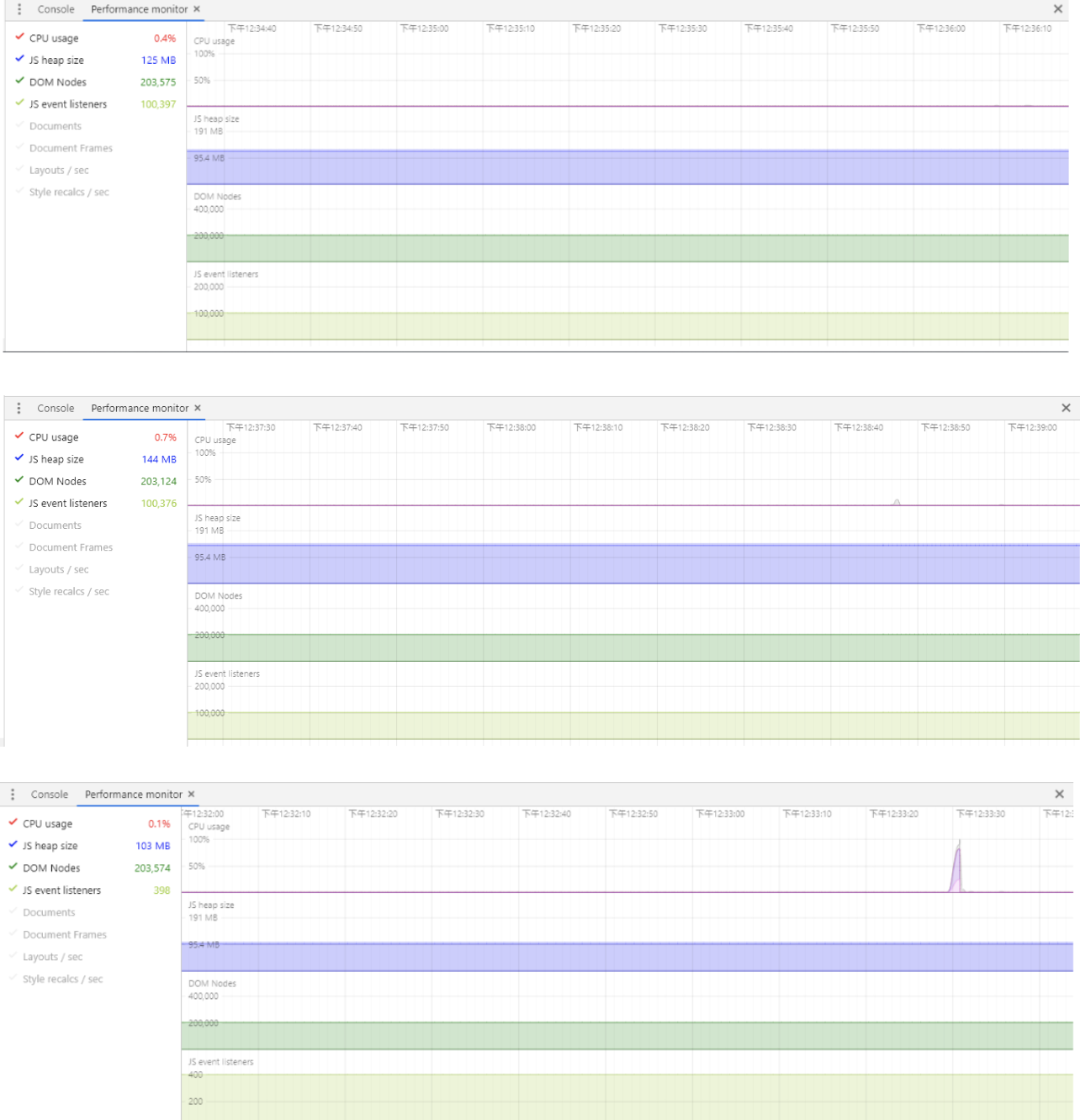
v-for="(item,index) of 100000"

:key="index">

{{item}}

</span>

</div>



可以看到使用事件代理无论是监听器数量和内存占用率都比前两者要少

同时对比 3 个图中监听器的数量并没有发现 vue 会自动做事件代理，

但是一般给 v-for 绑定事件时，都会让节点指向同一个事件处理程序（第二种情况可以运行，但是 eslint 会警告），

一定程度上比每生成一个节点都绑定一个不同的事件处理程序性能好，但是监听器的数量仍不会变，所以使用事件代理会更好一点。

代码使用

<ul @click="meths">

<li v-for="(item,key) in 10" :key="key" :data-index="key">{{item}}</li>

</ul>

meths(e) {

if (e.target.nodeName.toLowerCase() === 'li') {

console.log(e.target.innerHTML)

console.log(e.target.dataset)

}

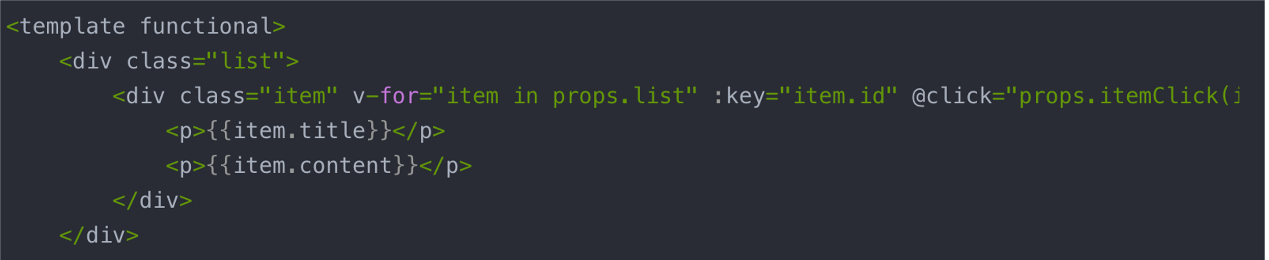
}

**9. 函数式组件**

函数式组件是无状态，它无法实例化，没有任何的生命周期和方法。创建函数式组件也很简单，只需要在模板添加functional声明即可。一般适合只依赖于外部数据的变化而变化的组件，因其轻量，渲染性能也会有所提高。

组件需要的一切都是通过context参数传递。它是一个上下文对象，具体属性查看文档。这里 props 是一个包含所有绑定属性的对象。

函数式组件



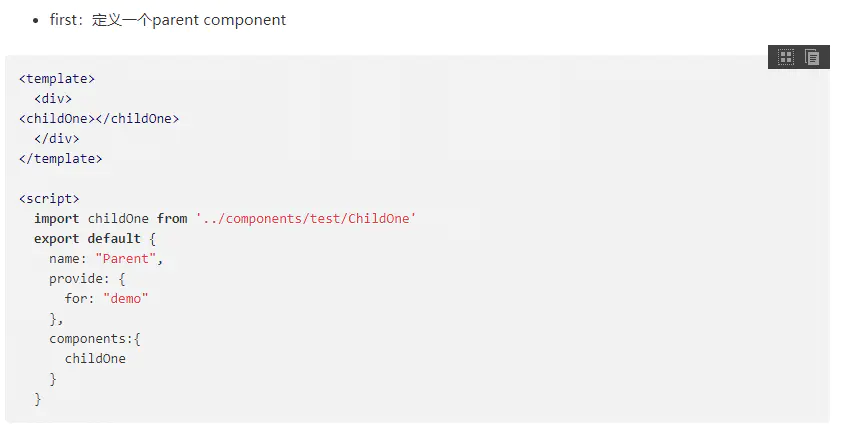
**10. provide和inject组件通信**

痛点：常用的父子组件通信方式都是父组件绑定要传递给子组件的数据，子组件通过props属性接收，一旦组件层级变多时，采用这种方式一级一级传递值非常麻烦，而且代码可读性不高，不便后期维护。

vue提供了provide和inject帮助我们解决多层次嵌套嵌套通信问题。在provide中指定要传递给子孙组件的数据，子孙组件通过inject注入祖父组件传递过来的数据，可以轻松实现跨级访问父组件的数据。

provide：是一个对象，或者是一个返回对象的函数。里面呢就包含要给子孙后代的东西，也就是属性和属性值。注意：子孙层的provide会掩盖祖父层provide中相同key的属性值。

inject：一个字符串数组，或者是一个对象。属性值可以是一个对象，包含from和default默认值，from是在可用的注入内容中搜索用的 key (字符串或 Symbol)，意思就是祖父多层 **provide** 提供了很多数据， **from** 属性指定取哪一个 **key** ;default指定默认值。







从上面这个例子可以看出，只要在父组件中调用了，那么在这个父组件生效的生命周期内，所有的子组件都可以调用inject来注入父组件中的值。

在使用场景中，肯定是希望父组件的数据一旦发生改变，子孙组件获取到的也是父组件更新后的数据。那么，怎么实现父组件与子孙组件所绑定的数据动态响应呢？

-------------------parent.vue----------------------

provide(){

return {

// keyName: {name:this.name}, // value 是对象才能实现响应式，也就是引用类型

keyName: this.changeValue // 通过函数的方式也可以[注意，这里是把函数作为value，而不是this.changeValue()]

// keyName: 'test' value 如果是基本类型，就无法实现响应式

}

},

data(){

return {

name:'张三'

}

},

methods: {

changeValue(){

this.name = '改变后的名字-李四'

}

}

-------------grandson.vue-----------------

inject:['keyName']

create(){

console.log(this.keyName) // 改变后的名字-李四

}

最后更新于 2021-09-06 14:15

如果觉得我的文章对您有用，请随意赞赏

赏

[4人已赞赏](javascript:void();)

[[](https://km.woa.com/user/bennzhang)](https://km.woa.com/user/bennzhang)[[](https://km.woa.com/user/kevinsqzhu)](https://km.woa.com/user/kevinsqzhu)[[](https://km.woa.com/user/journeylv)](https://km.woa.com/user/journeylv)[](https://km.woa.com/user/bartonwang)

仅供内部学习与交流，未经公司授权切勿外传

分类：[道聚城产品中心](https://km.woa.com/group/vasknowlege/articles?cat_id=166264" \t "_blank) 标签：[前端](https://km.woa.com/group/vasknowlege/articles?tag_id=960" \o "前端" \t "_blank)(1) [vue](https://km.woa.com/group/vasknowlege/articles?tag_id=112747" \o "vue" \t "_blank)(4) [优化](https://km.woa.com/group/vasknowlege/articles?tag_id=1019)(1) [iOS](https://km.woa.com/group/vasknowlege/articles?tag_id=28517)(1) [Android](https://km.woa.com/group/vasknowlege/articles?tag_id=2180)(1)



本文专属二维码，扫一扫还能分享朋友圈

想要微信公众号推广本文章？[点击获取链接](javascript:void(0);)

我顶 (39)

收藏 (80)

* [转载 (1)](javascript:void(0);)

* [收录](javascript:void(0);)
* [评论](javascript:void(0);)(21)
* [反馈](javascript:void(0);)

分享到

大家评论

[](https://km.woa.com/user/calebguo)

[calebguo](https://km.woa.com/user/calebguo)

2021-08-31 10:53:28

十分受用，棒啊

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/kevinsqzhu)

[kevinsqzhu](https://km.woa.com/user/kevinsqzhu)

2021-08-31 10:54:17

顶，快捷有效的优化点 

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/kathycheng)

[kathycheng](https://km.woa.com/user/kathycheng)

2021-08-31 10:54:19

学习了~赞！

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/journeylv)

[journeylv](https://km.woa.com/user/journeylv)

2021-08-31 10:54:44

很受用 

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/bennzhang)

[bennzhang](https://km.woa.com/user/bennzhang)

2021-08-31 10:54:48

总结的真好 

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/waleefu)

[waleefu](https://km.woa.com/user/waleefu)

2021-08-31 10:55:48

666

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/grubbyzheng)

[grubbyzheng](https://km.woa.com/user/grubbyzheng)

2021-08-31 10:56:02

小胖儿666啊 

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/kennyge)

[kennyge](https://km.woa.com/user/kennyge)

2021-08-31 10:56:12

学习了~赞！

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/zoladong)

[zoladong](https://km.woa.com/user/zoladong)

2021-08-31 10:59:14

厉害厉害，虽然我读不懂

 顶 [回复](javascript:void(0);)

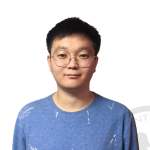
[](https://km.woa.com/user/felixiong)

[felixiong](https://km.woa.com/user/felixiong)

2021-08-31 11:00:19

顶！！！

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/morganyang)

[morganyang](https://km.woa.com/user/morganyang)

2021-08-31 11:01:49

666

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/anqiyang)

[anqiyang](https://km.woa.com/user/anqiyang)

2021-08-31 11:01:53

棒棒棒棒！

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/zhanyuzhang)

[zhanyuzhang](https://km.woa.com/user/zhanyuzhang)

2021-08-31 11:04:57

雄哥nb！(这里的每一个字符我都认识，可是连起来就不知道什么意思了(来自非技术🐶))

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/angelsli)

[angelsli](https://km.woa.com/user/angelsli)

2021-08-31 11:05:06

很有用

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/haryli)

[haryli](https://km.woa.com/user/haryli)

2021-08-31 11:20:14

666

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/goldenli)

[goldenli](https://km.woa.com/user/goldenli)

2021-08-31 11:21:59

666

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/yushizhen)

[yushizhen](https://km.woa.com/user/yushizhen)

2021-08-31 11:24:13

很有用~

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/granthou)

[granthou](https://km.woa.com/user/granthou)

2021-08-31 11:41:40

学到了！ 

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/v_ccqcao)

[v\_ccqcao](https://km.woa.com/user/v_ccqcao)

2021-09-01 18:12:47

牛啊 

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/lawtedwu)

[lawtedwu](https://km.woa.com/user/lawtedwu)

2021-09-02 04:08:11

🐂

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/xujiawang)

[xujiawang](https://km.woa.com/user/xujiawang)

2021-09-03 19:44:16

6666学到了

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/rickyrqzhao)



[切换到更多功能](javascript:void(0);)

关于作者

[](https://km.woa.com/user/bartonwang)

[bartonwang(王雄)](https://km.woa.com/user/bartonwang)

IEG\增值服务部\移动开发组员工

作者文章

* [【道聚城】Vue跨平台性能优化十法](https://km.woa.com/posts/show/519475?kmref=author_post)
* [【道聚城微分享】Flutter学习分享](https://km.woa.com/posts/show/508022?kmref=author_post)

Copyright©1998-2021 Tencent Inc. All Rights Reserved

腾讯公司研发管理部 版权所有

[广告申请](https://km.woa.com/chartlets/km/) [反馈问题](javascript:void(0);)

[423/454/511 ms]