vue框架中的key详解

在之前进行列表渲染的时候，我们在里面都看到了一个`key`。之前给同学样去给同学样讲解`key`的时候，这个东西不能重复，所以我们一般都会使用索引来进行

不推荐使用index做吸

> 如果执行是静态渲染，则使用`index`做为key无可厚非，如果执行的是动态的渲染，则`index`千万不能做为key,否则vue内部的状态机就容易出错

```
<body>
    <div id="app">
        <ul>
            <li v-for="(item,index) in stuList" :key="index">
                学号：{{item.sid}}---姓名：{{item.sname}}
            </li>
        </ul>
    </div>
</body>
<script src="./js/vue.global.js"></script>
<script>
     Vue.createApp({
        data(){
            return {
                stuList:[
                    {sid:"H1001",sname:"张珊"},
                    {sid:"H1002",sname:"李四"},
                    {sid:"H1003",sname:"王五"},
                ]
            }
        }
    }).mount("#app")
</script>
```

这就是一个表态渲染，因为数组`stuList`后面不会发生改变了

存在的问题

```
<body>
    <div id="app">
        学号:<input type="text" v-model="sid">
        姓名:<input type="text" v-model="sname">
        <button type="button" @click="addData">添加</button>
        <ul>
            <li v-for="(item,index) in stuList" :key="index">
                <input type="checkbox">
                学号：{{item.sid}}---姓名：{{item.sname}}
            </li>
        </ul>
    </div>
</body>
```

```
<script src="./js/vue.global.js"></script>
<script>
    Vue.createApp({
        data(){
            return  {
                sid: "",
                sname: "",
                stuList: [{
                        sid: "H1001",
                        sname: "张珊"
                    },
                    {
                        sid: "H1002",
                        sname: "李四"
                    },
                    {
                        sid: "H1003",
                        sname: "王五"
                    },
                ]
            }
        },
        methods: {
            addData() {
                this.stuList.unshift({
                    sid: this.sid,
                    sname: this.sname
                });
                this.sid = "";
                this.sname = "";
            }
        }
    }).mount("#app")
</script>
```

添加数据之前

学号: 姓名: 添加

- 学号：H1001---姓名：张珊
- 学号：H1002---姓名：李四
- 学号：H1003---姓名：王五

添加数据的时候

学号: H123123 姓名: 标哥 添加

- ☐ 学号：H1001---姓名：张珊
- ☑ 学号：H1002---姓名：李四
- ☐ 学号：H1003---姓名：王五

添加完成的时候

学号: 姓名: 添加

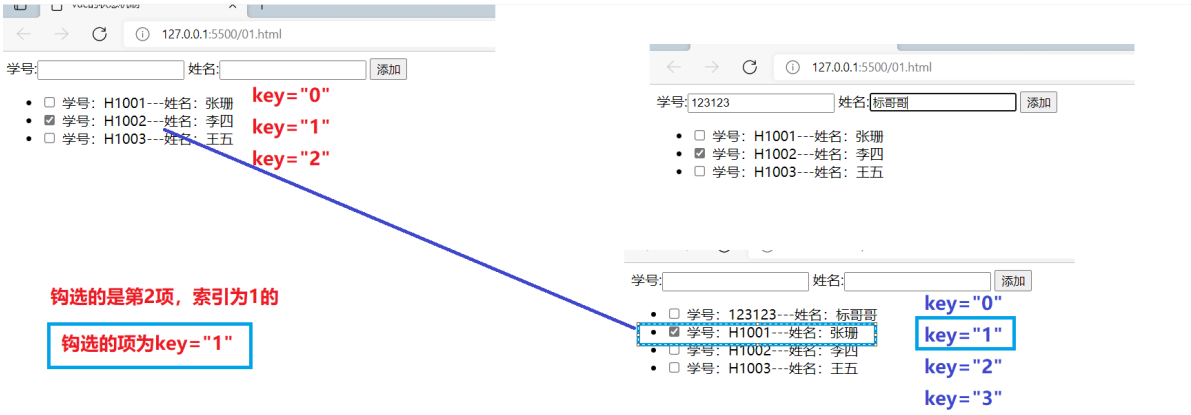
- ☐ 学号：H123123---姓名：标哥
- ☑ 学号：H1001---姓名：张珊
- ☐ 学号：H1002---姓名：李四
- ☐ 学号：H1003---姓名：王五

这个时候发现原本应该是“李四”这一项被钩选，结果现在变成了“张三”。原因？？？？？？

在vue的内部，vue是通过key来控制状态的，而我们在渲染的时候是通过

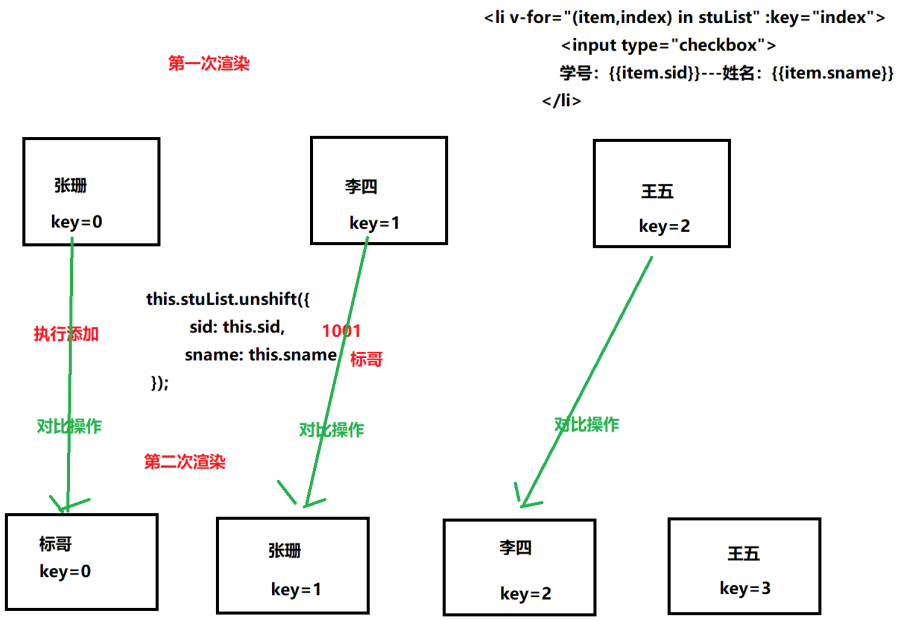
```
1    <li v-for="(item,index) in stuList" :key="index">
2        <input type="checkbox">
3        学号：{{item.sid}}---姓名：{{item.sname}}
4    </li>
```

我们的内部使用索引做了key，所以在变化之前，vue记住了是 `key` 为1 的这一项被钩选了，然后再下次重新渲染的时候，渲染完成以后，它还会将 `key` 为1的钩选，但是这个时候，发现key已经变了



---

vue在执行内部渲染的时候，使用了一种叫 `diff` 的算法，它把渲染之前和渲染之后的数据进行对比，然后只更改差异项



vue的内部是将渲染之前的数据与渲染之后的数据进行对比，这个时候对比依据就是key

它在执行对比的时候，发现每个key 都改变了，它又重新渲染了4个，但是本应该只渲染1次，上面的代码就极大的消耗了浏览器的性能，所以如果执行的是动态渲染，我们不推荐使用 `index` 做key

解决方法

```
<li v-for="(item,index) in stuList" :key="item.sid">
    <input type="checkbox">
    学号：{{item.sid}}---姓名：{{item.sname}}
</li>
```

学号: 姓名: 添加

sid做key

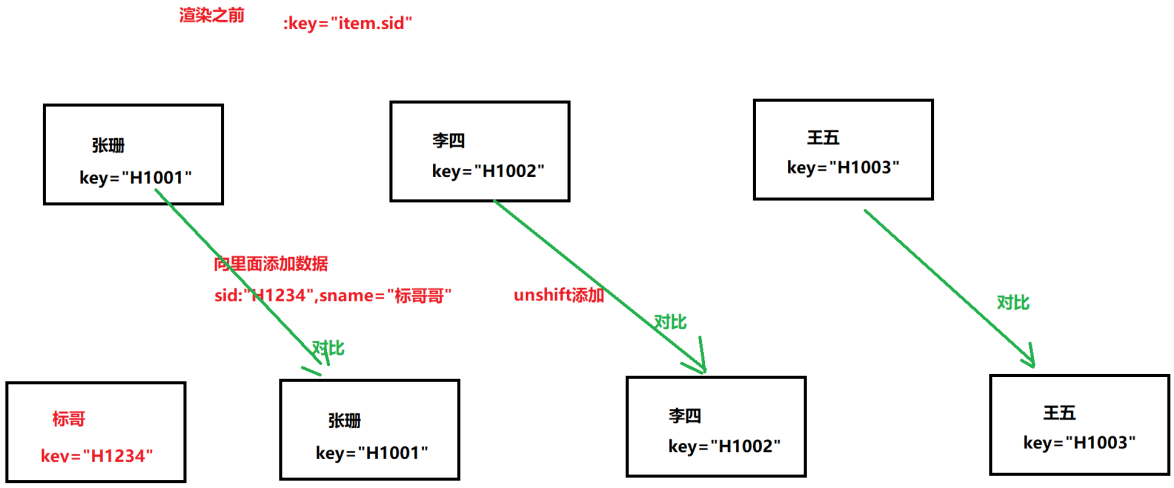
- 学号：H1001---姓名：张珊 key="H1001"
- 学号：H1002---姓名：李四 kev="H1002"
- 学号：H1003---姓名：王五 key="H1002"

vue的内部记住了一个状态，key="H1002"这一项被钩选了

学号: 姓名: 添加

- 学号：H1234---姓名：标哥哥 kev="H1234"
- 学号：H1001---姓名：张珊 key="H1001"
- 学号：H1002---姓名：李四 kev="H1002"
- 学号：H1003---姓名：王五 key="H1002"

我们现在再来看一下vue是怎么执行的内部对比



这个时候 vue在内部进行对比的时候，发现后面三项都没有变，只有前面多了一项，所以在渲染的时候也仅仅只会更改最前面的一项，这样效率非常高，也答会我们的数据驱动页面的本质

案例

```
<!DOCTYPE html>
<html lang="zh">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>transition中key</title>
    <style>
        .list-box {
            list-style-type: none;
            margin: 0;
            padding: 0;
            display: flex;
        }

        .list-box>li {
            width: 40px;
            height: 40px;
            background-color: lightseagreen;
            margin: 5px;
            display: flex;
            justify-content: center;
            align-items: center;
        }

        .aaa-enter {
            transform: translateY(-150%);
            opacity: 0;
        }

        .aaa-enter-to,
        .aaa-leave {
            transform: translateY(0);
            opacity: 1;
        }
```

```
37
38          .aaa-leave-to {
39              opacity: 0;
40              transform: translateY(150%);
41          }
42
43          .aaa-enter-active,
44          .aaa-leave-active {
45              transition: all 1s linear;
46          }
47      </style>
48  </head>
49
50  <body>
51      <div id="app">
52          <button type="button" @click="add">+</button>
53          <button type="button" @click="sub">-</button>
54          <hr>
55          <transition-group name="aaa" tag="ul" class="list-box">
56              <li v-for="(item,index) in numList" :key="item+'aaa'">{{item}}</li>
57          </transition-group>
58      </div>
59  </body>
60  <script src="./js/vue.js"></script>
61  <script>
62      new Vue({
63          el: "#app",
64          data: {
65              numList: [0, 1, 2, 3]
66          },
67          methods: {
68              add() {
69                  this.numList.push(this.numList.length);
70              },
71              sub() {
72                  let index = parseInt(Math.random() * this.numList.length);
73                  this.numList.splice(index, 1);
74              }
75
76          }
77      })
78  </script>
79
80  </html>
```

如果我们将上面的渲染过程里在的`:key="index"`，这样就实现不了我们的效果，因为vue在内部进行对比的时候找不到你的变化项（或者会把你的变化项判断错误）