

v-for 指令

1. 用于展示列表数据
2. 语法: `v-for="(item,index) in/of xxx" :key='yyy'`
3. 可以遍历: **数组、对象、字符串**(用的很少)、**指定次数**(用的很少)

使用 v-for 循环遍历 数组对象

```
1      <div id="root">
2          <!-- 使用 v-for 循环遍历数组对象 -->
3          <ul>
4              <li v-for="(p,index) in preson" :key="index">
5                  {{p.id}}--{{p.name}}--{{p.age}}--{{index}}
6              </li>
7          </ul>
8      </div>
9
10     <script>
11         new Vue({
12             el: '#root',
13             data: {
14                 preson: [
15                     {id: '001', name: "张三", age: 18},
16                     {id: '002', name: "徐思", age: 20},
17                     {id: '003', name: "冯宝宝", age: 22}
18                 ]
19             }
20         })
21     </script>
```

使用 v-for 循环遍历 对象

```
1      <div id="a">
2          <!-- 使用 v-for 循环遍历对象 -->
3          <ul>
4              <li v-for="(value,k) of cars" :key="k">
5                  {{value}} ____ {{k}}
6              </li>
7          </ul>
8      </div>
9
10     <script>
11         const vm = new Vue({
12             el: "#a",
13             data: {
14                 // 对象
15                 cars: {
```

```

16         name: '奥迪A8',
17         price: 70,
18         ml: '80L'
19     }
20 }
21 })
22 </script>

```

遍历字符串

用处不多

```

1     <div id="a">
2         <!-- 遍历字符串，用处不多 -->
3         <ul>
4             <li v-for="(value,k) of string" :key="k">
5                 {{value}} ____ {{k}}
6             </li>
7         </ul>
8     </div>
9
10    <script>
11        const vm = new Vue({
12            el: "#a",
13            data: {
14                string: 'hello'
15            }
16        })
17    </script>

```

遍历指定次数

```

1     <div id="a">
2         <!-- 遍历字符串，用处不多 -->
3         <ul>
4             <li v-for="(value,k) of 5" :key="k">
5                 {{value}} ____ {{k}}
6             </li>
7         </ul>
8     </div>

```

key的内部原理

1. 虚拟DOM中 key 的作用

1. key 是虚拟DOM对象的标识，当数据发生变化时，Vue会根据【新数据】生成【新的虚拟DOM】，随后Vue进行【新虚拟DOM】与【旧虚拟DOM】的差异比较，比较规则如下：

2. 对比规则

1. 旧虚拟DOM中找到了与新虚拟DOM相同的 key：

1. 如果虚拟DOM中内容没变，直接使用之前的真实DOM
2. 如果虚拟DOM中内容发生变化，则生成新的真实DOM，随后替换页面中之前的真实DOM

2. 旧虚拟DOM中未找到与新虚拟DOM相同的 key

1. 创建新的真实DOM，随后渲染到页面

3. 用 index 作为 key 可能会引发的问题：

1. 如果对数据进行：逆序添加、逆序删除等破坏顺序操作：

1. 会产生没有必要的真实DOM更新 ——> 界面效果没问题，但是效率低

2. 如果结构中还包含了输入类的DOM

1. 会产生错误DOM更新 ——> 界面有问题

4. 开发中如何选择 key？

1. 最好使用每条数据的唯一标识作为 key，比如 **id、手机号、身份证号、学号**等唯一值

2. 如果不存在对数据的逆序添加、逆序删除等破坏顺序操作，仅用于渲染列表用于展示

1. 使用 index 作为 key 是没有问题的。

```
1 <div id="a">
2   <button @click.once="add">添加</button>
3   <ul>
4     <li v-for="p of persons" :key="p.id">
5       {{p.name}}-{{p.age}}
6       <input type="text" >
7     </li>
8   </ul>
9 </div>
10
11 <script>
12   const vm = new Vue({
13     el:"#a",
14     data:{
15       persons:[
16         {id:'001',name:'行三',age:18},
17         {id:'002',name:'露丝',age:19},
18         {id:'003',name:'李武',age:20}
19       ],
20     },
21     methods: {
22       add(){
23         const p= {id:'004',name:'老六',age:30};
24         this.persons.unshift(p);
25       }
26     },
27   })
28 </script>
```

列表过滤

使用 `watch` 监视属性实现列表过滤

```
1      <div id="a">
2          <input type="text" placeholder="输入用户信息" v-model="keyword">
3          <!-- 遍历数组 -->
4          <ul>
5              <li v-for="(p,index) of filPersos" :key="index">
6                  {{p.name}}-{{p.age}}-{{p.sex}}
7              </li>
8          </ul>
9      </div>
10
11     <script>
12         new Vue({
13             el: '#a',
14             data: {
15                 keyword: '',
16                 persons: [
17                     {id: '001', name: '马冬梅', age: 18, sex: '女'},
18                     {id: '002', name: '周冬雨', age: 19, sex: '女'},
19                     {id: '003', name: '周杰伦', age: 20, sex: '男'},
20                     {id: '003', name: '温兆伦', age: 20, sex: '男'},
21                 ],
22                 filPersos: []
23             },
24             watch: {
25                 keyword: {
26                     immediate: true,
27                     handler(val) {
28                         this.filPersos = this.persons.filter((p) => {
29                             return p.name.indexOf(val) !== -1;
30                         })
31                     }
32                 }
33             }
34         })
35     </script>
```

使用 `computed` 计算属性实现列表过滤

```
1      <div id="a">
2          <input type="text" placeholder="输入用户信息" v-model="keyword">
3          <!-- 遍历数组 -->
4          <ul>
5              <li v-for="(p,index) of filPersons" :key="index">
6                  {{p.name}}-{{p.age}}-{{p.sex}}
7              </li>
8          </ul>
9      </div>
10
```

```

11 <script>
12   new vue({
13     el: '#a',
14     data: {
15       keyword: '',
16       persons: [
17         {id: '001', name: '马冬梅', age: 18, sex: '女'},
18         {id: '002', name: '周冬雨', age: 19, sex: '女'},
19         {id: '003', name: '周杰伦', age: 20, sex: '男'},
20         {id: '003', name: '温兆伦', age: 20, sex: '男'},
21       ],
22     },
23     computed: {
24       filPersons() {
25         return this.persons.filter((p) => {
26           return p.name.indexOf(this.keyword) !== -1;
27         })
28       }
29     }
30   })
31 </script>

```

列表排序

```

1 <div id="a">
2   <input type="text" placeholder="输入用户信息" v-model="keyword">
3   <button @click="sortType=2">年龄升序</button>
4   <button @click="sortType=1">年龄降序</button>
5   <button @click="sortType=0">原顺序</button>
6   <!-- 遍历数组 -->
7   <ul>
8     <li v-for="(p,index) of filPersons" :key="index">
9       {{p.name}}-{{p.age}}-{{p.sex}}
10     </li>
11   </ul>
12 </div>
13
14 <script>
15   new vue({
16     el: '#a',
17     data: {
18       sortType: 0, // 0 代表原顺序, 1 降序, 2 升序
19       keyword: '',
20       persons: [
21         {id: '001', name: '马冬梅', age: 18, sex: '女'},
22         {id: '002', name: '周冬雨', age: 19, sex: '女'},
23         {id: '003', name: '周杰伦', age: 20, sex: '男'},
24         {id: '003', name: '温兆伦', age: 20, sex: '男'},
25       ],
26     },
27     computed: {

```

```
28         filPersons(){
29             const arr = this.persons.filter((p)=>{
30                 return p.name.indexOf(this.keyword) !== -1;
31             })
32             // 判断一下是否需要排序
33             if(this.sortType){
34                 arr.sort((p1,p2) =>{
35                     return this.sortType === 1 ? p2.age-p1.age :
p1.age-p2.age
36                 })
37             }
38             return arr;
39         }
40     }
41 })
42
43 </script>
```