`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

```
    <div id="root">
        <!-- 使用 v-for 循环遍历数组对象 -->
        <ul>
            <li v-for="(p,index) in preson" :key="index">
                {{p.id}}--{{p.name}}--{{p.age}}--{{index}}
            </li>
        </ul>
    </div>

    <script>
        new Vue({
            el:'#root',
            data:{
               preson:[
                {id:'001',name:"张三",age:18},
                {id:'002',name:"徐思",age:20},
                {id:'003',name:"冯宝宝",age:22}
               ]
            }
        })
    </script>
```

使用 v-for 循环遍历 对象

```
    <div id="a">
        <!-- 使用 v-for 循环遍历对象 -->
        <ul>
            <li v-for="(value,k) of cars" :key="k">
                {{value}} ____ {{k}}
            </li>
        </ul>
    </div>

    <script>
        const vm = new Vue({
            el:"#a",
            data:{
                    //  对象
                cars:{
```

```
                    name:'奥迪A8',
                    price:70,
                    ml:'80L'
                }
            }
        })
    </script>
```

遍历字符串

用处不多

```
    <div id="a">
        <!-- 遍历字符串，用处不多 -->
        <ul>
            <li v-for="(value,k) of string" :key="k">
                {{value}} ____ {{k}}
            </li>
        </ul>
    </div>

    <script>
        const vm = new Vue({
            el:"#a",
            data:{
                string:'hello'
            }
        })
    </script>
```

遍历指定次数

```
    <div id="a">
        <!-- 遍历字符串，用处不多 -->
        <ul>
            <li v-for="(value,k) of 5" :key="k">
                {{value}} ____ {{k}}
            </li>
        </ul>
    </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。

```
    <div id="a">
        <button @click.once="add">添加</button>
        <ul>
            <li v-for="p of persons" :key="p.id">
                {{p.name}}-{{p.age}}
                <input type="text" >
            </li>
        </ul>
    </div>

    <script>
        const vm = new Vue({
            el:"#a",
            data:{
                persons:[
                    {id:'001',name:'行三',age:18},
                    {id:'002',name:'露丝',age:19},
                    {id:'003',name:'李武',age:20}
                ],
            },
            methods: {
                add(){
                    const p= {id:'004',name:'老六',age:30};
                    this.persons.unshift(p);
                }
            },
        })
    </script>
```

列表过滤

使用`watch`监视属性实现列表过滤

```
<div id="a">
    <input type="text" placeholder="输入用户信息" v-model="keyword">
    <!-- 遍历数组 -->
    <ul>
        <li v-for="(p,index) of filPersos" :key="index">
            {{p.name}}-{{p.age}}-{{p.sex}}
        </li>
    </ul>
</div>

<script>
    new Vue({
        el:'#a',
        data:{
            keyword:'',
            persons:[
                {id:'001',name:'马冬梅',age:18,sex:'女'},
                {id:'002',name:'周冬雨',age:19,sex:'女'},
                {id:'003',name:'周杰伦',age:20,sex:'男'},
                {id:'003',name:'温兆伦',age:20,sex:'男'},
            ],
            filPersos:[]
        },
        watch:{
            keyword:{
                immediate:true,
                handler(val){
                    this.filPersos = this.persons.filter((p)=>{
                        return p.name.indexOf(val) !== -1;
                    })
                }
            }
        }
    })
</script>
```

使用`computed`计算属性实现列表过滤

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

```

```
    <script>
        new Vue({
            el:'#a',
            data:{
                keyword:'',
                persons:[
                    {id:'001',name:'马冬梅',age:18,sex:'女'},
                    {id:'002',name:'周冬雨',age:19,sex:'女'},
                    {id:'003',name:'周杰伦',age:20,sex:'男'},
                    {id:'003',name:'温兆伦',age:20,sex:'男'},
                ],
            },
            computed:{
                filPersons(){
                    return this.persons.filter((p)=>{
                        return p.name.indexOf(this.keyword) !== -1;
                    })
                }
            }
        })
    </script>
```

列表排序

```
    <div id="a">
        <input type="text" placeholder="输入用户信息" v-model="keyword">
        <button @click="sortType=2">年龄升序</button>
        <button @click="sortType=1">年龄降序</button>
        <button @click="sortType=0">原顺序</button>
        <!-- 遍历数组 -->
        <ul>
            <li v-for="(p,index) of filPersons" :key="index">
                {{p.name}}-{{p.age}}-{{p.sex}}
            </li>
        </ul>
    </div>

    <script>
        new Vue({
            el:'#a',
            data:{
                sortType:0,     //0 代表原顺序，1降序，2升序
                keyword:'',
                persons:[
                    {id:'001',name:'马冬梅',age:18,sex:'女'},
                    {id:'002',name:'周冬雨',age:19,sex:'女'},
                    {id:'003',name:'周杰伦',age:20,sex:'男'},
                    {id:'003',name:'温兆伦',age:20,sex:'男'},
                ],
            },
            computed:{
```

```
                filPersons(){
                    const arr =  this.persons.filter((p)=>{
                        return p.name.indexOf(this.keyword) !== -1;
                    })
                    // 判断一下是否需要排序
                    if(this.sortType){
                        arr.sort((p1,p2) =>{
                            return this.sortType === 1 ? p2.age-p1.age : p1.age-p2.age
                        })
                    }
                    return arr;
                }
            }
        })

    </script>
```