后盾网 人人做后盾

www.houdunwang.com

Vue.js 2

后盾网 2011-2017 v2.0

Vue.js是当下很火的一个JavaScript MVVM库，它是以数据驱动和组件化的思想构建的。相比于Angular.js，Vue.js提供了更加简洁、更易于理解的API，使得我们能够快速地上手并使用Vue.js。

如果你之前已经习惯了用jQuery操作DOM，学习Vue.js时请先抛开手动操作DOM的思维，因为Vue.js是数据驱动的，你无需手动操作DOM。它通过一些特殊的HTML语法，将DOM和数据绑定起来。一旦你创建了绑定，DOM将和数据保持同步，每当变更了数据，DOM也会相应地更新。

Vue热度越来越高，很多大型项目已经在使用vue.js如饿了吗、美团等，Laravel框架已经将vue.js内置其中做为默认的前端框架。

Vue.js

Vue.js 的核心是一个允许采用简洁的模板语法来声明式的将数据渲染进 DOM

```
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<div id="app">
    {{name}}
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            name: 'houdunren.com',
        }
    })
</script>
```

每个 Vue.js 应用都是通过构造函数 Vue 创建一个 Vue 的根实例启动的，现在数据和 DOM 已经被绑定在一起，一切都是响应式的。

声明式渲染

有时我们想将输出的html内容使用浏览器进行解析，这时可以使用v-html指令完成。

```
<div id="app">
    <span v-html="name"></span>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            name: '<h1>你好后盾网houdunwang.com</h1>'
        }
    })
</script>
```

输出保留HTML格式

难免我们要在视图中进行计算，vue.js支持在双花括号中使用JS表达式。

```
<div id="app">
    {{price>10?20:30}}
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            price: 23
        }
    })
</script>
```

只能使用单个表达式，不能使用复杂的嵌套表达式，下面的代码将无效。

```
{{ if (ok) { return message } }}
```

使用javascript表达式

除了文本插值，我们还可以采用这样的方式绑定 DOM 元素属性：

```
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<div id="app">
    <a href="" v-bind:id="name">后盾人</a>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            name: 'houdunren.com',
        }
    })
</script>
```

绑定属性

控制切换一个元素的显示也相当简单。v-if当值为false时隐藏元素，v-if隐藏元素是会删除DOM的。

```
<div id="app">
    <div v-if="status">当status为真时我才显示</div>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            status: true,
        }
    })
</script>
```

v-if控制显示与隐藏

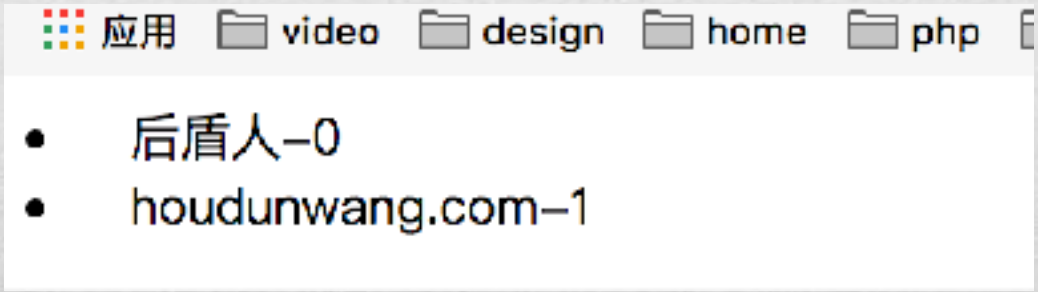
通过v-show也可以控制显示或隐藏，v-show使用display来进行控制。

```
<div id="app">
    <div v-show="status">当status为真时我才显示</div>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            status: false,
        }
    })
</script>
```

v-show控制显示与隐藏

v-for指令用于循环遍历数据。

```
<div id="app">
    <li v-for="(v,k) in todos">{{v.title}}-{{k}}</li>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            todos: [
                {title: '后盾人', id: 1},
                {title: 'houdunwang.com', id: 2}
            ]
        }
    })
</script>
```



循环输出

修饰符（Modifiers）是以半角句号 . 指明的特殊后缀，用于指出一个指令应该以特殊方式绑定。例如，.prevent 修饰符告诉 v-on 指令对于触发的事件调用 event.preventDefault()。

下例通过为submit事件添加prevent修释符来阻止表单提交的默认行为。

```
<div id="app">
    <form action="" method="post" v-on:submit.prevent="submit">
        <input type="submit">
    </form>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
        },
        methods:{
            submit:function(){
                alert(3);
            }
        }
    })
</script>
```

指令修释符

我们可以用 v-on 指令绑定一个调用 Vue 实例方法的事件监听器。

```
<div id="app">
    <button type="button" v-on:click="show">点我触发vue事件函数</button>
</div>
<script>
    new Vue({
        el: '#app',
        data: {},
        methods: {
            show: function () {
                alert('后盾人');
            }
        }

    })
</script>
```

绑定事件

Vue 还提供了 v-model 指令，使表单输入和应用状态间的双向绑定变得轻而易举。

```
<div id="app">
    <input type="text" v-model="name">
    <br/>
    {{name}}
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            name: '后盾人'
        }
    })
</script>
```

表单双向绑定

Vue.js 允许你自定义过滤器，可被用作一些常见的文本格式化，过滤器设计目的就是用于文本转换，过滤器支持链式操作使用。

```
<div id="app">
    {{title|splice(2)|star}}
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            title: '后盾人人人做后盾'
        },
        filters: {
            //这是定义的一个字符串截取过滤器
            splice: function (con, len) {
                return con.substr(0, len);
            },
            star: function (con) {
                return con + '***';
            }
        }
    })
</script>
```

过滤器

有些指令使用非常频繁，vue.js对这些使用频繁的指令提供了简写的方式。

```
<div id="app">
    <span :title="title">这是简写的v-bind</span>
    <hr>
    <button @click="show">这是简写的v-on</button>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            title: '后盾人人人做后盾'
        },
        methods:{
            show:function(){
                alert('houdunwang.com');
            }
        }
    })
</script>
```

缩写

有时我们想显示一个内容，但是这个内容是需要计算才可以得到的，这时我们可以使用vue.js中的computed属性进行处理。

```
<div id="app">
    <input type="text" v-model="n1"> +
    <input type="text" v-model="n2"> =
    <span>{{sum}}</span>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            n1: '',
            n2: ''
        },
        methods: {
            show: function () {
                alert('houdunwang.com');
            }
        },
        computed: {
            sum: function () {
                return this.n1 * 1 + this.n2 * 1;
            }
        }
    })
</script>
```

计算属性

vue.js提供了丰富的处理class样式类的方式 。

```
<style>
    .red{color:red}
    .size{font-size:100px;}
</style>
<div id="app">
   <span v-bind:class="['red','size']">后盾人 houdunren.com</span>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
        }
    })
</script>
```

设置class

有时我们的前台计算需要呈现不同的样式，这时vue.js的动态设置class就是最好的选择了。

```
<div id="app">
   <li v-for="v in todos" :class="{red:v.isDel}">{{v.title}} <button v-
on:click="del(v)">删除</button></li>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            todos:[
                {title:'后盾人',isDel:false},
                {title:'hdphp',isDel:false},
            ]
        },
        methods:{
            del:function(item){
                item.isDel=true;
            }
        }
    })
</script>
```

动态设置class

vue.js除了支持设置class也同时支持设置行级样式，使用原理与:class 相似。

```
<div id="app">
   <span :style="{color:'red',fontSize:size+'px'}">后盾人
houdunren.com</span>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            size:100
        }
    })
</script>
```

绑定行内样式

vue.js支持在视图使用 v-if/v-else/v-else-if 进行流程控制，根据不同场景显示不同内容。

```
<div id="app">
    <input type="radio" v-model="sex" value="girl"> 女孩
    <input type="radio" v-model="sex" value="boy"> 男孩
    <div v-if="sex=='girl'">
        你是女孩子，你应该喜欢看韩剧
    </div>
    <div v-else-if="sex=='bod'">
        你是男孩子，去打篮球吧:)
    </div>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            sex:''
        }
    })
</script>
```

流程控制

可以在事件方法中传递$event参数，用来阻止冒泡或阻止元素的默认事件行为。

```
<div id="app">
    <div @click="parent($event)">
        父元素
        <h1 @click="child($event)">子元素</h1>
        <a href="http://houdunwang.com" @click="link($event)">阻止元素默认行为</a>
    </div>
</div>
<script>
    new Vue({
        el: '#app',
        data: {},
        methods: {
            parent: function (event) {
                alert('我是父级')
            },
            child: function (event) {
                //阻止冒泡
                event.stopPropagation();
                alert('我是子级');
            },
            link:function(event){
                //阻止了默认行为
                event.preventDefault();
            }
        }
    })
</script>
```

事件中传递event对象

在事件处理程序中调用 event.preventDefault() 或 event.stopPropagation() 是非常常见的需求。vue.js为我们提供了修饰符的方式处理这些常见的需求。

```
<!-- 阻止单击事件冒泡 -->
<a v-on:click.stop="doThis"></a>

<!-- 提交事件不再重载页面 -->
<form v-on:submit.prevent="onSubmit"></form>

<!-- 修饰符可以串联 -->
<a v-on:click.stop.prevent="doThat"></a>

<!-- 只有修饰符 -->
<form v-on:submit.prevent></form>

<!-- 添加事件侦听器时使用事件捕获模式 -->
<div v-on:click.capture="doThis">...</div>

<!-- 只当事件在该元素本身（而不是子元素）触发时触发回调 -->
<div v-on:click.self="doThat">...</div>
```

事件修饰符

在监听键盘事件时经常需要监测常见的键值。 Vue 允许为 v-on在监听键盘事件时添加按键修饰符。

```
<div id="app">
    <input type="text" @keyup.13="post">
</div>
<script>
    new Vue({
        el: '#app',
        data: {},
        methods: {
            post: function (event) {
                alert('你按了回车键');
            }
        }
    })
</script>
```

按键修饰符

vue.js为常见的按键起了别名。

```
<input type="text" @keyup.enter="post">
```

按键别名：

```
.enter
.tab
.delete (捕获 “删除” 和 “退格” 键)
.esc
.space
.up
.down
.left
.right
.ctrl
.alt
.shift
.meta
```

在Mac系统键盘上，meta对应命令键 (⌘)。在Windows系统键盘meta对应windows徽标键 (⊞)

按键修饰符别名

默认情况下单个复选框的值是bool类型，true或false。

```
<div id="app">
    <input type="checkbox" v-model="checkbox">
    {{checkbox}}
</div>
<script>
    new Vue({
        el: '#app',
        data: {checkbox:''}
    })
</script>
```

使用:true-value与:false-value设置checkbox的值

```
<div id="app">
    <div id="example-6" class="demo">
        <input type="checkbox" v-model="checkbox" :true-value="'a'" :false-
value="'b'">
        <span>{{ checkbox }}</span>
    </div>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            checkbox: ''
        }
    })
</script>
```

单个复选框

默认情况下单个复选框的值是bool类型，我们可以使用:true-value与:false-value来设置checkbox的值

```
<div id="app">
        <input type="checkbox" v-model="checkbox" :true-value="'a'" :false-value="'b'">
        <span>{{ checkbox }}</span>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            checkbox: ''
        }
    })
</script>
```

:true-value中可以写字符串也可以是vue的变量。

:true-value与:false-value

多个复选框需要将属性设置为数组[]形式即可。

```
<div id="app">
    <input type="checkbox" v-model="checkbox" value="hdphp">
    <input type="checkbox" v-model="checkbox" value="hdcms">
    {{checkbox}}
</div>
<script>
    new Vue({
        el: '#app',
        data: {checkbox:[]}
    })
</script>
```

复选框组

单选按钮可以设置value值，选中哪个单选按钮vue的值就是哪个按钮的value值。

```
<div id="app">
    <input type="radio" v-model="radio" value="hdphp">
    <input type="radio" v-model="radio" value="hdcms">
    {{radio}}
</div>
<script>
    new Vue({
        el: '#app',
        data: {radio:''}
    })
</script>
```

复选框组

列表框需要多选时将vue.js属性值设置为数组即可，这与复选框的使用方式一样。

```
<div id="app">
    <div id="example-6" class="demo">
        <select v-model="selected" multiple>
            <option>HDPHP</option>
            <option>HDCMS</option>
        </select>
        <br>
        <span>Selected: {{ selected }}</span>
    </div>
</div>
<script>
    new Vue({
        el: '#app',
        data: {selected: []}
    })
</script>
```

列表框

```
<div id="app">
    <div id="example-6" class="demo">
        <select v-model="selected" v-model="selected">
            <option v-for="v in category" :value="v.cid">{{v.catname}}</option>
        </select>
        <br>
        <span>Selected: {{ selected }}</span>
    </div>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            selected: 2,
            category: [
                {cid: 1, catname: '新闻'},
                {cid: 2, catname: '游戏'},
                {cid: 3, catname: '后盾网'}
            ]
        }
    })
</script>
```

使用v-for输出列表框

默认情况下我们在表单中输入内容后会立刻进行视图更新，有时为了效果我们希望离开表单后才更新。.lazy表单修释符就是这种情况下使用的。

```
<div id="app">
    <input type="text" v-model.lazy="name">
    <hr>
    {{name}}
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            name: ''
        }
    })
</script>
```

表单修释符.lazy

如果想自动将用户的输入值转为 Number 类型（如果原值的转换结果为 NaN 则返回原值），可以添加一个修饰符 number 给 v-model 来处理输入值：

```
<input v-model.number="age" type="number">
```

如果要自动过滤用户输入的首尾空格，可以添加 `trim` 修饰符到 `v-model` 上过滤输入：

```
<input v-model.trim="msg">
```

.number与.trim