20分钟入门Vue.js

Vue.js是什么

> " Vue.js 是一个用来开发web界面的前端库。它也有配套的周边工具。如果把这些东西都算在一起，那么你也可以叫它一个『前端框架』。但我个人更倾向于把它看做是一套可以灵活选择的工具组合。"
>
> ----Vue.js的作者尤雨溪

安装

为了学习方便，可以直接引入：

```
<script src="https://unpkg.com/vue"></script>
```

也可以用npm，或者Vue-cli命令行工具安装。具体请参考 Vue安装。

数据绑定

不废话，直接上代码。

```
<div id="app">
    {{message}}
</div>

<script>
    var app = new Vue({
        el: '#app',
        data: {
            message: 'Hello Vue!'
        }
    });
</script>
```

运行结果：

> Hello Vue!

上面的代码中， el属性指向页面Dom的id， {{ }}用于将data里的数据绑定到Dom。 {{ }}支持简单的表达式，比如：将 message 反向输出 `{{message.split('').reverse().join('')}}`。

试试在浏览器的 console 中输入 app.message = "hello Ctrip!" 看看有什么变化？

> 模型层(model)只是普通 JavaScript 对象，修改它则更新视图(view)。

这就是Vue的响应式系统。

计算属性

模板内的表达式虽然便利，但却不是最好的选择。过多的表达式会使模板变得难以维护。 Vue.js中引入了 computed 属性来解决字段的复杂逻辑。让我们修改一下上面的代码：

```
<div id="app">
    {{reverseMessage}}
</div>

<script>
    var app = new Vue({
        el: '#app',
        data: {
            message: 'Hello Vue!'
        },
        computed : {
            reverseMessage : function(){
                return this.message.split('').reverse().join('');
            }
        }
    });
</script>
```

对于复杂逻辑，应尽量避免直接在模板中操作表达式。 其实用 methods 或者 watched 属性也可以达到相同的效果，具体差异请自行研究：computed VS methods VS watched。

指令

> 指令（Directives）是带有 v- 前缀的特殊属性。

继续上面的代码，我们用指令为他添加一个点击事件：

```
<div id="app" v-on:click="sayHi()">
    {{message}}
</div>

<script>
    var app = new Vue({
        el: '#app',
        data: {
            message: 'Hello Vue!'
        },
        methods: {
            sayHi: function(){ alert("hi!");}
        }
```

```
    });
</script>
```

我们用 v-on 创建了一个事件指令，这个指令在监听到 click 事件时会执行 methods 中绑定的函数。Vue.js中提供了丰富的指令，比如：

- 用于标签属性绑定的 v-bind
- 用于条件渲染的 v-if
- 用于列表渲染的 v-for
- 用于表单绑定的 v-model
- 用于事件绑定的 v-on

主要掌握这5种指令，基本就可以构造出一个简单的Vue页面应用了。（想要获取更多请参考 Vue指令）

组件化

> 组件 (Component) 是 Vue.js 最强大的功能之一。组件可以扩展 HTML 元素，封装可重用的代码。

组件的支持使得 Vue.js 能够用于大的项目中。首先，需要注册一个组件：

```
Vue.component('my-component',{
    props: ['item'],
    template: '<div>{{item.text}} <input type="text" v-model="item.value"></div>'
});
```

props表示组件可以接受哪些属性，template表示组件的模板。 接着，我们可以在页面中使用这个组件了：

```
<div id="component">
    <my-component v-bind:item="item_data"></my-component>
</div>
```

这里的 v-bind 表示把 Vue 实例的 item_data 绑定到组件的 item 属性。 最后，创建一个 Vue 实例：

```
var myComponent = new Vue({
    el: '#component',
    data: {
        item_data: {
            text: "年龄",
            value: "18岁"
        }
    }
});
```

在这个例子中，Vue 实例中的data通过页面Dom指令绑定到组件的props之中，组件再把接受到的props渲染到模板之中，最后呈现到页面。 这样，我们就封装并使用了一个基本的表单输入框。

核心插件

在构建单页应用上，可以使用 vue-router 来设置路由； 类似于react.js的redux，Vue.js 也提供了 Vuex 来管理程序状态。