周报计划

这周继续按照《移动WEB前端开发》的书籍进行学习。

这本书籍的最后一部分主要介绍MVVM框架BackboneJS。现在互联网公司基本放弃了BackboneJS，主要还是使用Vue，React，Angular。

这三大框架，相对来说Vue 是上手成本最低，中文文档最全，使用也是偏简单的，我们在这里也使用VueJS。

Vue (读音 /vjuː/，类似于 **view**) 是一套用于构建用户界面的**渐进式框架**。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与[现代化的工具链](https://cn.vuejs.org/v2/guide/single-file-components.html)以及各种[支持类库](https://github.com/vuejs/awesome-vue" \l "libraries--plugins" \t "https://cn.vuejs.org/v2/guide/_blank)结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。

声明式渲染：

<div id="app">

{{ message }}

</div>

var app = new Vue({

el: '#app',

data: {

message: 'Hello Vue!'

}

})

Vue 在背后做了大量工作。现在数据和 DOM 已经被建立了关联，所有东西都是**响应式的**。

<div id="app-2">

<span v-bind:title="message">

鼠标悬停几秒钟查看此处动态绑定的提示信息！

</span>

</div>

v-bind 特性被称为**指令**。指令带有前缀 v-，以表示它们是 Vue 提供的特殊特性。

<div id="app-3">

<p v-if="seen">现在你看到我了</p>

</div>

var app3 = new Vue({

el: '#app-3',

data: {

seen: true

}

})

V-if切换一个元素是否显示

<div id="app-4">

<ol>

<li v-for="todo in todos">

{{ todo.text }}

</li>

</ol>

</div>

var app4 = new Vue({

el: '#app-4',

data: {

todos: [

{ text: '学习 JavaScript' },

{ text: '学习 Vue' },

{ text: '整个牛项目' }

]

}

})

v-for 指令可以绑定数组的数据来渲染一个项目列表：

<div id="app-5">

<p>{{ message }}</p>

<button v-on:click="reverseMessage">逆转消息</button>

</div>

var app5 = new Vue({

el: '#app-5',

data: {

message: 'Hello Vue.js!'

},

methods: {

reverseMessage: function () {

this.message = this.message.split('').reverse().join('')

}

}

})

v-on 指令添加一个事件监听器，通过它调用在 Vue 实例中定义的方法：

<div id="app-6">

<p>{{ message }}</p>

<input v-model="message">

</div>

var app6 = new Vue({

el: '#app-6',

data: {

message: 'Hello Vue!'

}

})

Vue 还提供了 v-model 指令，它能轻松实现表单输入和应用状态之间的双向绑定。

组件系统是 Vue 的另一个重要概念，因为它是一种抽象，允许我们使用小型、独立和通常可复用的组件构建大型应用。仔细想想，几乎任意类型的应用界面都可以抽象为一个组件树：

Vue.component('todo-item', {

template: '<li>这是个待办项</li>'

})

<ol>

<!-- 创建一个 todo-item 组件的实例 -->

<todo-item></todo-item>

</ol>

Vue.component('todo-item', {

// todo-item 组件现在接受一个

// "prop"，类似于一个自定义特性。

// 这个 prop 名为 todo。

props: ['todo'],

template: '<li>{{ todo.text }}</li>'

})

当你把一个普通的 JavaScript 对象传给 Vue 实例的 data 选项，Vue 将遍历此对象所有的属性，并使用 Object.defineProperty 把这些属性全部转为 getter/setter。Object.defineProperty 是 ES5 中一个无法 shim 的特性，这也就是为什么 Vue 不支持 IE8 以及更低版本浏览器。

这些 getter/setter 对用户来说是不可见的，但是在内部它们让 Vue 追踪依赖，在属性被访问和修改时通知变化。这里需要注意的问题是浏览器控制台在打印数据对象时 getter/setter 的格式化并不同，所以你可能需要安装 vue-devtools 来获取更加友好的检查接口。

每个组件实例都有相应的 watcher 实例对象，它会在组件渲染的过程中把属性记录为依赖，之后当依赖项的 setter 被调用时，会通知 watcher 重新计算，从而致使它关联的组件得以更新。