Vue.js实战-读书笔记

目录

**键入章标题(第 1 级)1**

键入章标题(第 2 级)2

键入章标题(第 3 级)3

**键入章标题(第 1 级)4**

键入章标题(第 2 级)5

键入章标题(第 3 级)6

# 第1章初识Vue.js

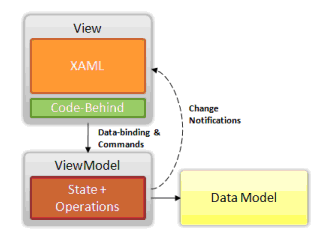
可以 渐进式（Progressive），一步一步、有阶段性地来使用Vue.js。

MVVM(Model-View-ViewModel)模式：双向绑定（data-binding）

**View：** 通过DataContext和ViewModel绑定，负责界面和显示，界面构成元素有window, controls, page, dataTemplete, custom controls….。

**ViewModel：**是view和model的桥梁，是对Model的抽象，主要包括界面逻辑和模型数据封装，Behavior/Command事件响应，绑定的属性定义等，维护视图状态

**Model：**数据和业务逻辑



每一个框架的产生都是为了解决某个具体的问题。 Angular、React、Vue.js为了解决更复杂的业务场景，比如：SPA(单页面富应用)、组件解耦，提高开发效率、降低维护成本。

# 第2章数据绑定和第一个Vue应用

## 缩写

v-bind: 缩写为 :

v-on: 缩写为 @

document.getElementById('app') 缩写为 '#app'

## v-bind绑定属性

v-bind  主要用于Html属性绑定， 当绑定变量发生变化时，对应的属性也会发生变化.

|  |
| --- |
| <a v-bind:href="url"></a>  <ul class="box"  v-bind:class="{‘textColor‘:isColor, ‘textSize’:isSize}"> |

## v-on 绑定事件监听器

v-on主要用于属性绑定， 当绑定变量发生变化时，对应的属性也会发生变化.

|  |
| --- |
| <button v-on:click="handleClick" >点击隐藏/显示</button>  methods: {  handleClick:function(){  this.show=(this.show)?false:true;  }  } |

如果绑定的事件要处理复杂的业务逻辑，建议在methods里声明一个业务方法。

## v-model 双向绑定数据

虽然很像使用了**双向数据**绑定的 Angular 的 ng-model，但是 Vue 是单项数据流，v-model 只是语法糖而已，简写为**：** 下面三种写法是一样的

|  |
| --- |
| <input v-model="sth" />  <input **v-bind:value="sth"** **v-on:input="sth = $event.target.value"** />  <input **:**value="sth" **@**input="sth = $event.target.value" /> |

**v-bind:value="sth"： 绑定变量变化时，控件的value发生变化**

**v-on:input="sth = $event.target.value"控件输入发生变化时，绑定变量发生变化**

最终实现了v-model的双向数据绑定

## 生命周期

create: 实例创建完成后调用，此时el还未挂载不可用，作初始化处理一些数据。

mounted: el挂载到实例上调用， 开始做业务逻辑

beforeDestory: 实例销毁之前调用， 解绑一些使用addEventListener监听事件等

## v-pre

v-pre这个元素内的内容不会被编译。就像html里<pr>这个标签可以将其包起来的文字排版、格式，原封不动的呈现出来

<span v-pre>{{ 这里的内容是不会被编译的 }}</span>

## v-html

v\_html元素插入一段html, 例如：动态插入一段html

## filters 过滤器

在{{}}插值的尾部添加一个管道符“|”对数据进行过滤，经常用于格式化文本

<p > {{ date | **formatDate**(**days**) }} </p>

new Vue({

//el:document.getElementById("app"),

el: '#app',

data: {

date : new Date(),

days:0

},

**filters**: {

**formatDate** : function(value,days){

。。。。

return 。。。;

}

}

})

可以串联多个过滤器， 也可以给过滤器加参数

# 第3章计算属性

## Computed 计算属性

computed: {

prices() {

get(){

… 计算价钱

return prices;

}

set(){

设置价钱后的业务操作

}

}

}

计算属性所依赖的数据(当前Vue实例的数据、其他实例的数据)发生变化时，它会重新计算值。

每一个计算属性包含一个getter和一个setter.

get: 默认, 当所依赖数据项发生变化触发重新计算。

set：当外部设置属性时时触发

# 第4章v-bind及class与style绑定

# 第5章内置指令

## v-cloak 编译前显示

它会在Vue实例结束编译时从绑定的HTML元素上移除，经常和CSS的display:none;配合使用。 解决加载慢导致页面闪动的最佳实践。

CSS文件代码：

[v-cloak] {

display: none;

}

HTML代码：

<div id=**"app"** v-cloak>

{{ message }}

</div>

展现: 开始是css文件显示空白页，加载完Vue.js后显示v-cloak的信息，加载完后显示编译后的正式界面.

## v-once 渲染一次

定义它的元素或组件只渲染一次，包括元素或组件的所有的子节点。首次渲染后，不再随数据的变化重新渲染，将被视为静态内容。

<span v-once>{{ message}}</span>

基本不用，优化性能时用到

## v-if、v-else-if、v-else 条件渲染指令

<p v-if="status == 1">当status为1时显示该行</p>

<p v-else-if="status==2">当status为2时显示该行</p>

<p v-else>否则显示该行</p>

Vue在渲染元素时，处于效率考虑，会**尽可能地复用已有的元素而非重新渲染**。

如果要避免元素复用，则给元素加**key值**（唯一的）

## v-show

改变元素的CSS属性display. V-show表达式为true时显示，否则隐藏

<p v-show="status == 1">当status为1时显示该行</p>

v-if加载快，适合不经常改变的场景， v-show加载慢，无切换开销适合频繁切换的场景。

还有区别是，v-show可以保存状态，而v-if切换后状态都丢了。（复用元素可以保留状态）

## v-for 列表循环渲染指令

当需要将一个数组遍历或枚举一个对象循环显示时，就会用到列表渲染指令v-for。它的表达式需结合in来使用。

<!-- 循环数据 -->

<ul v-for="(book,index) in books">

<li>书名：{{ book.name }}</li>

<li>作者：{{ book.author }}</li>

</ul>

<!-- 循环属性 -->

<ul v-for="(value,key,index) in user">

<li>{{ key }}:{{ value }}</li>

</ul>

<!-- 循环次数 -->

<ul><li v-for="n in 10"> {{ n }}、 </li></ul>

## 数组更新

Push() 末尾添加 pop()末尾取出 shift() 头部取出 unshift()头部添加 Splice()更改数组

从第几个开始，删除多少个，插入值。。。。

arrayObject.splice(index, howmany, Item1, Item2…. ItemX)

设置数组大小， 从中删除数组， 从中加入数组

## 数组过滤与排序

Filter() 过滤 sort()排序 reverse() 翻转

<!—数组过滤 -->

app.filterBooks = app.books.filter(function(item){

return item.name.match(/JavaScript/);

});

<!— 在自动计算里 数组排序 -->

Sortbooks(){

return this.books.sort(function(a,b){

return a.name.length < b.name.length;

});

}

## 方法与事件

<a href="http://www.apple.com" v-on:click="handleClick('禁止打开',$event)">打开链接</a>

handleClick(msg,event){

event.preventDefault();

}

传递事件， 阻止事件冒泡

## 事件修饰符、快捷键接收

<!-- 阻击单击事件冒泡 -->

<a @click.stop="handle"></a>

<!-- 提交事件不再重载页面 -->

<form @submit.prevent="handle"></form>

<!-- 添加事件侦听器使用事件捕获模式, 有捕获先触发事件，剩余按默认冒泡模式触发 -->

<div @click.capture="handle">...</div>

<!-- 只当事件在该元素本身（而不是子元素）触发时间触发回调 -->

<div @click.self="handle">...</div>

<!-- 只触发一次，组件同样适用 -->

<div @click.once="handle">...</div>

<!-- 只有修饰符 -->

<a @click.prevent></a>

<!-- 修饰符可以串联 -->

<a @click.stop.prevent="handle"></a>

表单元素上监听键盘事件，还可以使用按键修饰符，比如按下某个键时调用方法

快捷键名称：.enter .tab .delete .esc .space .up .down .left .right

.crtl .alt .shift .meat (mac的command,windows的窗口键)

<input @keyup.13 = "submit"> <!-- 回车 -->

<input @keyup.ctrl.83 = "handleSave"> <!-- ctrl + s -->

<div @click.ctrl = "doSomething">doSomething</div> <!-- ctrl + 单击 -->

# 第6章表单与v-model 55

6.1基本用法 55

6.2绑定值 61

6.3修饰符 63

# 第7章组件详解 65

7.1组件与复用 65

7.1.1为什么使用组件 65

7.1.2组件用法 66

7.2使用props传递数据 70

7.2.1基本用法 70

7.2.2单向数据流 72

7.2.3数据验证 74

7.3组件通信 75

7.3.1自定义事件 75

7.3.2使用v-model 77

7.3.3非父子组件通信 79

7.4使用slot分发内容 83

7.4.1什么是slot 83

7.4.2作用域 84

7.4.3slot用法 85

7.4.4作用域插槽 87

7.4.5访问slot 89

7.5组件高级用法 90

7.5.1递归组件 90

7.5.2内联模板 92

7.5.3动态组件 93

7.5.4异步组件 94

7.6其他 95

7.6.1$nextTick 95

7.6.2X-Templates 96

7.6.3手动挂载实例 97

7.7实战：两个常用组件的开发 98

7.7.1开发一个数字输入框组件 98

7.7.2开发一个标签页组件 106

# 第8章自定义指令 118

8.1基本用法 118

8.2实战 121

8.2.1开发一个可从外部关闭的下拉菜单 121

8.2.2开发一个实时时间转换指令v-time 126

第2篇进阶篇

# 第9章Render函数 133

9.1什么是VirtualDom 133

9.2什么是Render函数 136

9.3createElement用法 140

9.3.1基本参数 140

9.3.2约束 143

9.3.3使用JavaScript代替模板功能 147

9.4函数化组件 153

9.5JSX 157

9.6实战：使用Render函数开发可排序的表格组件 159

9.7实战：留言列表 172

9.8总结 183

# 第10章使用webpack 184

10.1前端工程化与webpack 184

10.2webpack基础配置 187

10.2.1安装webpack与webpack-dev-server 187

10.2.2就是一个js文件而已 188

10.2.3逐步完善配置文件 191

10.3单文件组件与vue-loader 194

10.4用于生产环境 201

# 第11章插件 206

11.1前端路由与vue-router 207

11.1.1什么是前端路由 207

11.1.2vue-router基本用法 208

11.1.3跳转 212[1]

11.1.4高级用法 213

11.2状态管理与Vuex 216

11.2.1状态管理与使用场景 216

11.2.2Vuex基本用法 217

11.2.3高级用法 221

11.3实战：中央事件总线插件vue-bus 227

第3篇实战篇

# 第12章iView经典组件剖析 235

12.1级联选择组件Cascader 236

12.2折叠面板组件Collapse 249

12.3iView内置工具函数 257

# 第13章实战：知乎日报项目开发 261

13.1分析与准备 261

13.2推荐列表与分类 265

13.2.1搭建基本结构 265

13.2.2主题日报 267

13.2.3每日推荐 271

13.2.4自动加载更多推荐列表 276

13.3文章详情页 278

13.3.1加载内容 278

13.3.2加载评论 281

13.4总结 286

# 第14章实战：电商网站项目开发 288

14.1项目工程搭建 288

14.2商品列表页 290

14.2.1需求分析与模块拆分 290

14.2.2商品简介组件 291

14.2.3列表按照价格、销量排序 297

14.2.4列表按照品牌、颜色筛选 306

14.3商品详情页 309

14.4购物车 313

14.4.1准备数据 314

14.4.2显示和操作数据 316

14.4.3使用优惠码 320

14.5总结 324

# 第15章相关开源项目介绍 325

15.1服务端渲染与Nuxt.js 325

15.1.1是否需要服务端渲染 325

15.1.2Nuxt.js 326

15.2HTTP库axios 327

15.3多语言插件vue-i18n 329[1]