复习：

见思维导图

1.小知识：组件中的HTML模板的声明方法

(1)<script>

var c = {

template: `<div>innerHTML...</div>`

}

</script>

(2)<script type="text/x-template" id="myTpl2">

<div>innerHTML</div>

</script>

<script>

var c = {

template: '#myTpl2'

}

</script>

(3)<template id="myTpl3">

<div>innerHTML</div>

</template>

<script>

var c = {

template: '#myTpl3'

}

</script>

2.总结：组件间通信的三种方法

**(1) Props Down, Event Up**

通过自定义属性把父组件的数据传递给子组件；

通过自定义事件把子组件的数据传递给父组件；

数据传递范围可控，推荐使用。

**(2) $refs 和 $parent**

<my-parent ref="childRef">

父组件获得子组件的引用： this.$refs.childRef

子组件获得父组件的引用： this.$parent

数据传递范围不可控，不推荐使用。

**(3) 通信总线(Bus)**

var bus = new Vue()

bus.$emit('事件', 数据)

bus.$on('事件, fn)

用于在兄弟组件间传递数据。

3.Vue.js中的动画效果

Vue 在插入、更新或者移除 DOM 时(v-if、v-show、v-for、组件间切换等)，提供多种不同方式的应用过渡效果。

包括以下工具：

(1)在 CSS 过渡和动画中自动应用 class

(2)可以配合使用第三方 CSS 动画库，如 Animate.css

(3)在过渡钩子函数中使用 JavaScript 直接操作 DOM

(3)可以配合使用第三方 JavaScript 动画库，如 Velocity.js

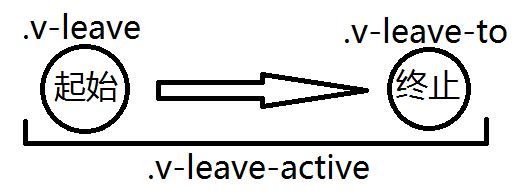
**方式1：使用CSS过渡(transition)动画**

Vue.js为添加在<transtion>标签中的即将消失的DOM元素提供了三个class：

.v-leave-active：离开动画进行中的class

.v-leave：即将离开时的class，动画的第一帧

.v-leave-to：离开动画终止时的class，动画的最后一帧



Vue.js为添加在<transtion>标签中的即将显现的DOM元素提供了三个class：

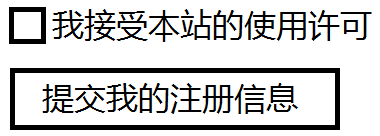
.v-enter-active：进入动画进行中的class

.v-enter：即将进入时的class，动画的第一帧

.v-enter-to：进入动画终止时的class，动画的最后一帧

提示：若页面中有多个需要实现不同动画效果的元素，只需要给<transtion>添加不同的name属性即可，上述六个class的前缀不再是v-，变为name值。

练习：“不接受许可”则按钮向左滑动离开并渐隐；“接受许可”则按钮从右面滑动进入并渐显。



午间自学：Animate.css —— 第三方动画效果库

|  |
| --- |
| Animate.css是一个使用CSS3 Keyframes实现的动画效果库，提供了几十个常用的动画效果，使用非常简单：  <ANY class="animated 动画效果类">  <ANY class="animated infinite 动画效果类"> |

**方式2：配合Animate.css动画库实现动画**

<transition

enter-active-class="元素进入的动画效果类"

leave-active-class="元素离开的动画效果类"

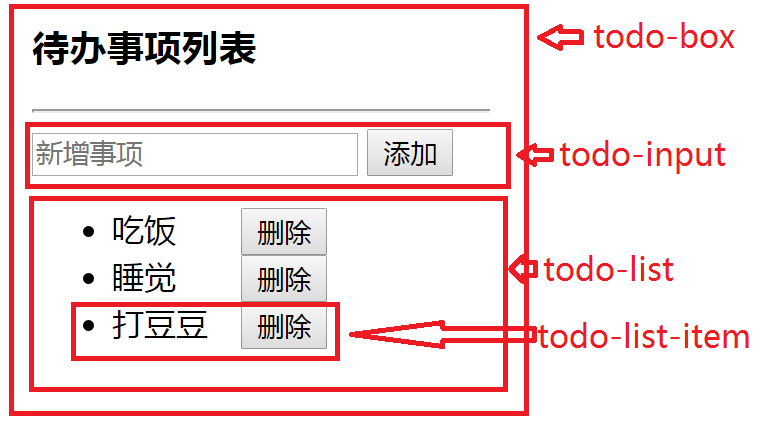
>

<ANY></ANY>

</transition>

4.终极练习：Todo List





**(1)设计基本的组件包含关系**

**(2)显示已有待办项(Props Down x 2)**

todoBox:existingTodoItem =>

todoList:list /(item in list) =>

todoListItem:itemcontent

**(3)添加新的待办项(Event Up)**

v-model + @click => $emit('useradditem', newTodoItem)

todoBox: @useradditem=fn

**(4)删除已有的待办项(Event Up x 2)**

todoListItem: this.$emit('userdeleteitem', itemId)

todoList: this.$emit('listdeleteitem', itemId)

todoBox: this.existingTodoItem.splice(itemId, 1) 从数组中删除1个元素

5.Vue.js阶段项目 —— “学子商城后台管理系统”框架重构

重构：Refactor，指开发中对于已有代码使用更好的方式“重新实现”一遍，体现设计和实现方面的优势，以更好的适应未来的应用场景。

除了Vue.js还需要两方面的辅助技术：

(1)SPA技术

(2)单文件式组件技术

6.SPA应用

Single Page Application：单页面应用，指整个项目有且仅有一个页面具备完整的HTML DOM树（引入外部各种资源）；不同的内容可以放在不同的“模板页面”中(只是一段HTML DIV片段而已)。

|  |  |
| --- | --- |
| **传统的多页应用** | **SPA应用** |
| 项目中的多个网页，每个都是完整的HTML内容 | 整个项目只有一个页面具有完整的HTML结构，其它来回切换的“页面”其实只是一段DIV片段 |
|  |  |
| 每次浏览器都要加载完整的DOM结构 | 每次所谓的“页面切换”其实只需要改变DOM树的部分内容，加载和渲染速度更快——尤其适用于移动端应用 |
| 页面间切换就是旧DOM树的删除和新DOM树的加载和重建——不可能有“过场动画”效果 | “页面切换”其实只是两个DIV在同一棵DOM树上切换，可以实现任意的过场动画 |

SPA应用的实现原理简介：

(1)定义“路由词典”：

[

{path: '/login', component: {template:'<div>...'}},

{path:'/main', component:{template:'<div>...'}},

...

]

(2)定义唯一“完整”的HTML页面：index.html

<html>

<head>js/css...</head>

<body>

....

<特殊的接口></特殊的接口>

....

</body>

</html>

(3)客户端发起请求：

http://127.0.0.1/index.html#/login

浏览器会先请求index.html创建DOM树，然后解析location.hash，若发现以'#/'开头，查找路由词典，动态异步加载指定的模板片段即可。

“页面跳转”其实只是在修改#/后的内容：

<a href="http://127.0.0.1/index.html#/main"，浏览器根本无需删除/重建DOM树，只需要再次查找路由词典，异步请求对应的组件模板，替换掉“特殊的接口”中已有的内容。

|  |
| --- |
| 说明：Vue.js本身并不具备SPA应用功能，必须依赖于官方提供的“插件”——Vue-Router |

Vue-Router插件的使用步骤：

(1)在“唯一完整的页面”中引入vue.js和vue-router.js

(2)在“唯一完整的页面”中指定用于挂载各种“模板页面(各种改锥头)”的接口：

<router-view></router-view>

(3)在JS中定义完整的“路由词典”——就是各种组件与请求地址的对应关系

var routes = [

{ path: '/', component: xx }, //默认路由地址

{ path: '/login', component: xx },

{ path: '/main', component: yy },

]

(4)定义路由器，封装路由词典：

var router = new VueRouter({

routes

})

(5)在根组件中声明路由器

var vm = new Vue(){

el: '#app',

router

}

课后练习：

(1)再次重现Todo List练习。

(2)自学VueRouter手册，通过“超链接”和“JS”两种方式实现模板页面的跳转，编写出类似学子商城“管理员登录页”和“管理员监控主页面”的页面跳转效果。