# 3 第三章

## 3.1 一些杂项

1. 大部分数组的更改都会触发重新渲染，对于某些不能触发重新渲染的操作，Vue中提供Vue.set()来进行更改，就可以触发重新渲染了。
2. ES6模块化语法
3. 指令v-html和v-text的区别：v-html会将里面的html标签渲染正确，而v-text不管里面有没有标签，都会原样输出。
4. data选项最好不要使用引用赋值，会导致有依赖的组件一起被改变。一定要使用引用赋值的话，将data赋值为一个匿名函数，返回值为原data选项。

建议都使用匿名函数，比较安全。

1. v-for可以遍历数组、对象

<h1 v-for=”item in array”></h1>

<h1 v-for=”(item, index) in array”></h1>

<h1 v-for=”val in obj”></h1>

<h1 v-for=”(val, key) in obj”></h1>

1. v-bind用来绑定html属性, 绑定的属性会同步更新；v-bind:src=”data选项的属性”冒号后面的src为参数，表示绑定src属性

v-bind:href缩写为:href

1. Vue2.0的组件名支持驼峰命名，以前不支持
2. 用v-bind绑定class属性的时候，绑定的值除了字符串变量，还可以为对象或数组。

对象：对象的属性名为类名，属性值为bool，表示是否渲染；

数组：用来渲染多个类名

1. v-bind绑定内联样式的时候，变量值可以为对象，像{ ‘color’: ‘red’ }
2. 条件渲染：由v-if和v-show两种，共同点都是传入一个布尔值；

v-if是渲染到DOM上还是不渲染到DOM上，

v-show是显示或隐藏

v-else用来配合v-if和v-show使用

## 3.2 事件绑定

1. 事件绑定使用v-on:事件指令，如v-on:click；v-on:的缩写为@，如@click

2. 常用修改器：

a) stop，阻止冒泡

v-on:click.stop

b) 一些按键，如v-on:keydown.enter表示按下enter键；也可以使用keyCode，如v-on:keydown.13

3. 重点：自定义事件（用来给父子组件通信）

## 3.3 表单事件绑定

1. 利用v-model进行表单value值的绑定

2. v-model的修改器：

v-model.lazy：不会随着用户的输入更新，当焦点离开输入框的时候才触发同步更新。

v-model.number：将value的值的类型转成number。

v-model.trim：去除value值的前后空格。

## 3.3 计算属性和数据监听

1. computed选项里面放置计算属性，适合用来对数据进行一些处理。

2. 计算属性可以取代一部分的自定义事件所能完成的功能。

3. 另外，计算属性的功能可以由methods里面的方法代替，就是说，计算属性可以做到的，methods里面的方法也可以做到。

这两种方式的区别是：

1. 只要数据更改了，计算属性就会更新；而方法要被调用了才会更新。
2. 计算属性要求的数据是data选项中的数据，而方法没有这个要求。

4. 使用watch选项可以监听一个数据的改变，如果改变了会调用用户定义的回调函数，而新的值和旧的值会作为回调函数的参数传入。

watch: {

aData: function (val, oldVal) {

console.log(val, oldVal);

}

}

# 4 第四章

## 4.1 深入Vue组件

1. 组件树结构（重点注意）

2. 一个项目只有一个根组件，也就是说只能有一个new Vue()。

main.js文件的代码：

import Vue from ‘vue’;

import App from ‘./App’;

new Vue({

el: “#app”,

render: function (h) {

// 渲染根组件

return h(App);

}

});

3. vue文件其实相当于配置文件，就是导出导入Vue实例的选项对象。

4. 可以通过is属性来引入组件:

<component is=”ComA”></component>

所以，可以实现动态引入组件了:

<component v-bind:is=”ComA”></component>

5. 重点：父子组件通信。

父传给子通过props：父组件中直接在使用子组件的地方用html属性的形式传值，子组件中通过props选项声明接收什么属性。

6. props的值可以为一个数组，里面放置所有要接收的属性名。

props的值也可以为一个对象，对象的属性名就是html属性名（最好加引号），属性值为期待接收的数据类型，形如String；如果是多个可以用数组，形如[Number, String, Object]

props和data的使用几乎一样。

7. html标签是大小写不敏感的，所以命名要用-的方式，而在js中会自动转化为驼峰，所以html中用-，js用驼峰即可。

## 4.2 动态属性传递

1. 父子组件通信中，如果父组件想给子组件传递动态属性的话，就需要使用v-bind来绑定一个数据了。

父组件使用子组件的地方：

<com-a v-bind:my-value=”myVal”></com-a>

子组件中同样需要props选项来声明该数据

## 4.3 子组件向父组件传递信息

1. 子组件中通过this.$emit(‘my-event’, aData)触发自定义事件来向外传递信息。

2. 父组件中在使用子组件的地方监听对应的自定义事件。

<com-a v-on:my-event=”onMyEvent”></com-a>

子组件要传递的信息作为父组件监听自定义事件的回调函数的参数被传入。

上面的例子中，就是传入到onMyEvent的参数

onMyEvent: function (data) {}

## 4.4 插槽

1. 插槽是父组件向子组件传递信息的一种方式。是通过模板传递的。

父组件使用子组件的地方：

<com-a>这是要传递的数据</com-a>

子组件的模板中定义插槽

<template>

<div>

<!—这是插槽 -->

<slot>这是默认值</slot>

<div>

</template>

2. 可以在子组件中给插槽起名字，从而在一个组件中使用多个插槽。

子组件的模板中定义插槽

<template>

<div>

<!—这是插槽 -->

<slot name=”header”>这是默认值</slot>

<slot name=”footer”>这是另外一个插槽</slot>

<div>

</template>

父组件使用子组件的地方：

<com-a>

<p slot=”header”>指定名为header的插槽</p>

<p slot=”footer”>指定名为footer的插槽</p>

</com-a>

## 4.5 动态组件

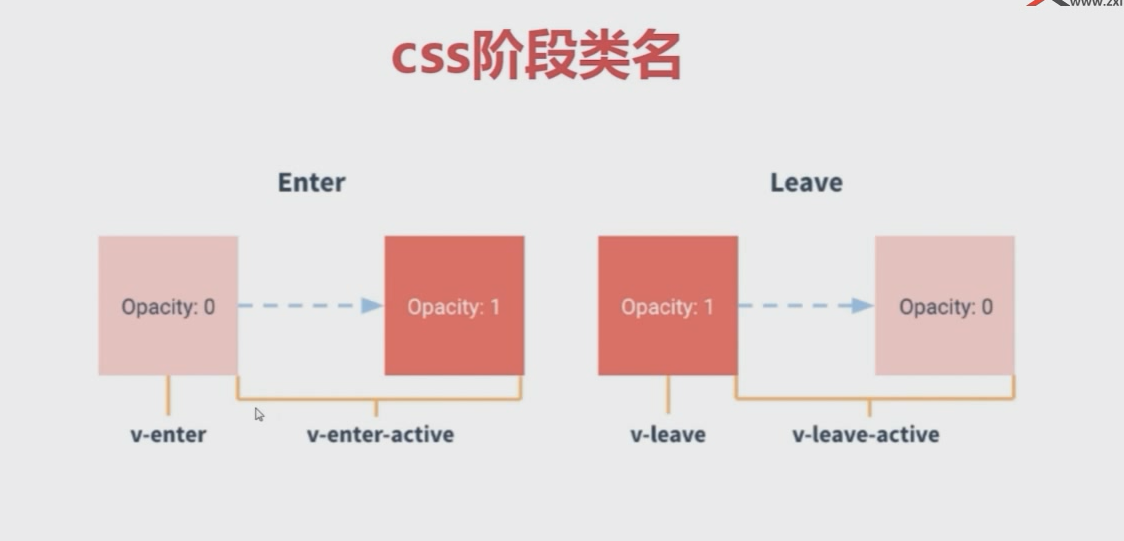
1. 动态组件就是通过v-bind来绑定is属性的值来指定要使用的组件。

2. 动态组件适合用在一些tab切换中，每一个tab都是一个组件，这里可以配合<keep-alive></keep-alive>内置组件使用。

## 4.6 Vue高级功能-过渡/动画

Vue的过渡有两种，一种是css过渡，另一种是js过渡。

### 4.6.1 css过渡



模板中（过渡需要配合v-show指令使用，v-if也可以）：

<transition name=”my-fade”>

<p v-show=”show”>现在你看到我了</p>

</transition>

样式（按照transition组件中的name属性来起样式名）：

<style>

.my-fade-enter-active,

.my-fade-leave-active {

transition: all 0.5s;

}

.my-fade-enter,

.my-fade-leave-active {

opacity: 0;

}

</style>

1. css中，定位的left, right等值不会被transition元素接收。

2. transition组件中需要配合v-show或v-if或动态组件使用。

3. transition组件中如果配合动态组件使用的话（即控制多个元素的显示隐藏），其默认的切换模式是新租件先进入，旧组件再消失，即默认为in-out；我们可以设置为out-in，这样就是旧的先消失，新的再出现。

<transition name=”my-fade” mode=”out-in”></transition>

4. transition配合v-if和v-else一起使用也属于上面说的控制多个元素显示隐藏，也是建议设置mode为out-in

如果v-if和v-else的标签名是一样的话，会不执行过渡效果，因为Vue的机制，不会触发重新渲染。这时需要设置key属性来将这两个标签区别开。

<transition name=”my-fade” mode=”out-in”>

<p v-show=”show” key=”1”>现在你看到我了</p>

<p v-else key=”2”>现在你看不到我了</p>

</transition>

### 4.2.2 js过渡

1. 如果是标准的Vue项目，要引入类似jQuery等第三方库的时候，建议在项目的index.html文件中引入。index.html文件在Vue项目中相当于一个入口页面。

2. Vue的js过渡的使用方式，可以参照css过渡。

因为是js，所以不需要name属性了。

js过渡还需要用v-on绑定过渡的各个时间段。

为了避免transition组件检查css样式，还需要绑定css属性为false

<transition

v-on:before-enter=”beforEnter”

v-on:enter=”enter”

v-on:leave=”leave”

v-bind:css=”false”

>

<p v-show=”show” key=”1”>现在你看到我了</p>

<p v-else key=”2”>现在你看不到我了</p>

</transition>

// 组件中methods选项部分，需要显示隐藏的元素会作为参数传入回调函数

// done参数在enter和leave中必须使用，是一个回调函数，需要在动画完成的时候调用

methods: {

beforeEnter: function (el) {},

enter: function (el, done) {},

leave: function (el, done) {}

}

## 4.7 Vue高级功能-自定义指令

### 4.7.1 局部自定义指令

在当前组件的选项中使用directives选项来定义局部自定义指令。

directives: {

color: function (el, binding) {

// el指当前使用该指令的元素

// binding是一个对象，里面包含了一些关于绑定的信息

// v-color=”data”里的data，可以是字符串、变量

var value = binding.value;

}

}

使用这个指令的时候，需要加上v-前缀，即v-color。

### 4.7.2 全局自定义指令

1. 在整个项目中，把directives选项设在根节点中就可以被全局使用了。

### 4.7.3 自定义指令的钩子函数

有bind、inserted、update三个钩子函数

**directives**: {  
 // 可选钩子函数来进行更精细的控制  
 **focus**: {  
 // 被绑定元素插入父节点时使用  
 inserted: function (el, binding) {  
 // js原生自带  
 el.focus();  
 }  
 }  
}

## 4.8 Vue高级功能-插件

1. 插件可以理解为功能丰富、完善的组件。

2. VueRouter使用（VueRouter是前端路由工具）

引入VueRouter库后，Vue.use(VueRouter)来注册使用VueRouter

其他插件使用方法类似。

## 4.9 单文件组件

1. 单文件组件就是单个的后缀名为vue的文件，里面由template、script、style三部分组成。

# 5 第五章

## 5.1 vue-cli

1. vue-cli用来初始化一个Vue项目，是官方提供的一个脚手架工具。

2. vue-cli优点

成熟的vue项目架构设计

提供本地测试服务器

集成打包上线方案

3. vue-cli也是一个命令行工具

4. 使用vue-cli之前必须要有Node.js和Git

5. 安装Node更多是为了npm的功能

6. 安装完vue-cli后，命令行中输入vue –help查看帮助

7. vue list命令列出所有官方的模板

一般使用webpack版本，初始化命令为vue init webpack 项目文件夹名

一路默认，直到一个ESLink，确认使用会严格要求代码格式，可能会报一些奇怪的错误，所以建议不开。单元测试

8. 安装完之后进入到项目目录，如果没有安装依赖，需要npm install命令来安装package.json里面的依赖（好像新版会自动安装）

9. 都安装完了之后，使用npm run dev命令启动内置服务器，浏览器可以访问到主页。内置服务器是热更新的，不用刷新页面就可以看到效果。

10. 使用npm run build命令生成可以发布到线上的文件，存放在dist目录中。已经对源码进行了压缩，可以直接上线。

11. 开发都在src目录中。项目使用了大量的ES6语法，可以了解ES6模块化。

## 5.2 Vue-router前端路由

### 5.2.1 前端路由基本使用和概念

1. 路由就是根据不同的地址跳到不同的页面。

2. 安装vue-router，在vue项目中使用npm install vue-router –save命令，后面的save是为了把vue-router加到package.json中。安装完vue-router之后，就可以在项目的任意地方使用import来引入vue-router了。

3. 现在vue-cli很智能，可以帮我们引入vue-router了。

4. 使用时要为需要配置路由的组件配置一个地址，配置好了就可以在地址栏中访问该组件了。访问之前需要先指定<router-view></router-view>，这里决定了组件的渲染位置。

5. 我们还可以使用<router-link to=”/apple”></router-link>来访问配置了路由的组件。

6. 实例化Router的时候，加入一个mode配置，值为history的话就可以不用在地址栏中加#（localhost:8080/#/apple，默认mode为hash模式），也可以直接访问，像这样：localhost:8080/apple。history模式怕刷新；hash模式就是可能难看了一点。

### 5.2.2 前端路由参数

1. 直接在配置的时候就可以加入参数，如path: ‘/apple/:color’，后面的:color就是参数了，访问的时候类似localhost:8080/apple/red这样就可以了。但是不加后面的参数就无法访问该组件，因为需要匹配。

2. 组件中要想拿到参数，可以访问this.$router.params获取所有参数。获取的参数存进一个对象里。

3. 参数可以十分灵活地使用，形如/apple/:color/detail/:haha的都可以无限加，前面带冒号的就是参数。切记，地址需要满足这样的形式才能访问到组件。

### 5.2.3 前端路由的嵌套

1. 路由的嵌套就是子路由。

2. // 配置苹果组件的路由

// 后面的:color为参数，地址栏中为localhost:8080/apple/red

{

path: '/apple/:color',

name: 'Apple',

component: Apple,

// 配置子路由，可以配置多个

children: [

{

// 这里不加/，就是下一级，/表示根路径

path: 'detail',

component: AppleCHild

}

]

}

3. 配置了子路由后，该组件的渲染位置在父组件中，所以需要在父组件的模板中定义router-view组件。

### 5.2.4 命名的路由视图、路由重定向

1. 访问路由有两种方式（直接在地址栏输入不算），一种是声明式导航：利用router-view组件访问；另一种式编程式导航，使用router.push({path: ‘/apple’});注意router是路由器，不是之前的route了。

2. 命名路由：

{

path: '/apple/:color',

// 给路由起名字就叫命名路由

name: 'Apple',

component: Apple,

},

<!-- 访问命名路由 -->

<router-link v-bind:to="{ name: 'Banana' }">访问香蕉组件</router-link>

3. 命名视图：

给router-view组件命名，<router-view name=”viewA”></router-view>，命名之后在路由配置中可以决定在哪一个router-view中（同一页的多个router-view）中显示哪个组件。

4. 路由重定向：

使用十分简单，只需在路由配置一下就可以。

{

path: '/',

// 如果访问根路径，就重定向到/banana

redirect: '/banana'

},

### 5.2.5 使用过渡制作路由跳转动画

可以给router-view组件套上transition组件和keep-alive组件。

## 5.3 vuex

1. vuex用来进行状态管理。

2. 安装vuex，进入到项目目录，运行npm install vuex –save命令即可安装。

3. 使用的话和router差不多，不过vue-cli没有集成进去，所以要自己引入

Import Vuex from ‘vuex’

Vue.use(Vuex);

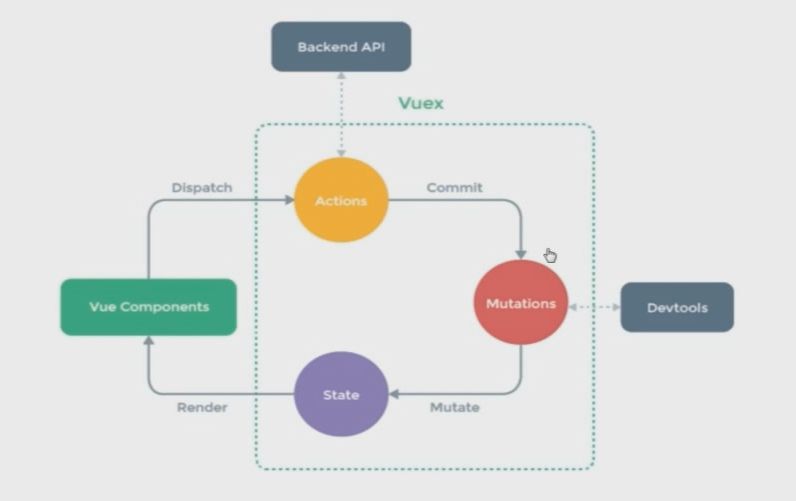
let store = new Vuex.Store({

配置

});

4. 如果项目不是很大，不推荐使用vuex，官网上的store模式就可以够用了。如果要构建一个中大型的单页应用，推荐使用vuex来管理状态。

5. vuex原理图：



6. actions可以进行异步操作，mutations只能进行同步操作。所以与后端交互的是actions。

7. actions只能调用mutation里面的方法，不能直接操作state的数据。

8. 除了state、mutations、actions这三个基本的之外，还可以加上getters选项，里面放置各种getter方法，主要用于获取state里面的数据。

9. vuex还支持模块化，每个模块都有自己的state、mutations、actions等。最后再合到一个store中。

const moduleA = {

state: {},

mutations: {},

actions: {}

};

const moduleB = {

state: {},

mutations: {},

actions: {}

};

const store = new Vuex.Store({

modules: {

a: moduleA,

b: moduleB

}

});

// 使用的时候按照某一个模块来取

store.state.a

store.state.b

10. 官方推荐的vuex在大型vue项目中的目录结构



# 6 第六章，项目开发

1. 如果v-for要渲染多个元素，可以使用template组件包起来，因为template标签不会渲染到页面中。
2. html原生class属性和v-bind绑定的class属性不冲突，可以同时存在。
3. 在vue-cli项目中，使用v-bind绑定img标签的src时，如果直接使用路径会出错，解决方法可以引入图片编码，使用require直接引入图片即可；或者将图片放在static目录下，该目录下的文件不会被改变。

其实：html、css方式都没问题，只要是用js引用，都需要require，比如v-bind或地址存在数组中。

1. Vue-resource的使用：

Vue-resource用来进行异步请求，有ajax和jsonp。

安装：npm install vue-resource

使用：和vue-router、vuex一样，需要Vue.use()，但不用加到配置中。

Use之后，就可以在任意组件中使用this.$http访问该功能。

1. json-server工具，可以从模仿后端取数据，实际上将数据存在一个json文件中。webpack-simple版本没有build文件夹，其文件在node modules文件夹中，搜索webpack-dev-server.js即可。（我觉得不好用，最好不用）
2. awesome-vue，vue的组件库。Github上找。
3. router-link组件还有tag和active-class属性，tag用于将ruoter-link渲染成其他标签（不指定的话默认渲染成a标签），active-class用于在激活的时候为其添加指定类（不指定的话也会有一个默认的激活类）。
4. 获取当前路由路径：this.$route.path
5. 想让logo点击会主页，可在img标签外套router-link即可。
6. 有些页面不得不存在，但不想被用户访问，可以使用路由重定向。