1 语法，生命周期，过滤器

2020年1月6日

9:13

# 第一部分 安装

**1、独立版本**

我们可以在 Vue.js 的官网上直接下载 vue.min.js 并用 **<script>** 标签引入。

**2、使用 CDN 方法**

以下推荐国外比较稳定的两个 CDN，国内还没发现哪一家比较好，目前还是建议下载到本地。

* + **Staticfile CDN（国内）** : <https://cdn.staticfile.org/vue/2.2.2/vue.min.js>
  + **unpkg**：<https://unpkg.com/vue/dist/vue.js>, 会保持和 npm 发布的最新的版本一致。
  + **cdnjs** : <https://cdnjs.cloudflare.com/ajax/libs/vue/2.1.8/vue.min.js>

**3、NPM 方法**

# 实现

**每个Vue应用都需要实例化一个Vue**

var vm = new Vue({ //注意 V大写大写大写  
 // 选项  
})

# 

# e.g.

<script>

var vm=new Vue({

el:"#app",

data:{

message:"标题1",

myname:""

}

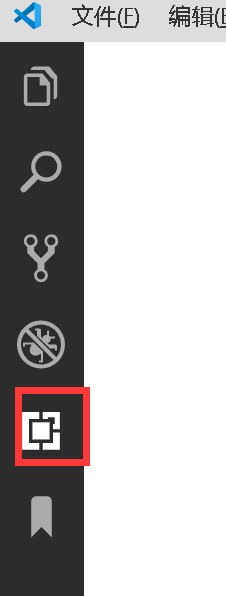
});

</script>

**vscode安装完毕后**

安装插件

Vetur 
Pine Wu I I (140) I 
VS 
Ant Design Vue helper 
Vue 3 Snippets 
hollowtree 1 (13) 1 
A Vwjs And 2 Code S nip*' 
Chinese (Simplified) Lan 
14,093.116 (41) 



点这里进行搜索即可

# 

# 

# 

# 

# 

# 第二部分 常用选项参数

el: 在页面上存在DOM元素作为Vue的挂载目标。只能选取一个元素，不能是多个 ,所以大多时候使用id选择器

也可以el: document.getElementById("app")

data: Vue实例的数据对象,可以是多个数据，数据也可以是数组类型

methods: Vue方法集合，可以在Vue直接调用，或者绑定到DOM元素的事件

computed: Vue的计算属性集合

watch: 观察 Vue实例变化的一个表达式或者计算属性函数

components: 包含Vue实例可用的组件的哈希表

fliters: 包含Vue实例可用的过滤器的哈希表

template: 定义字符串模板作为Vue实例的标志使用

注意，不同的参数间使用 逗号隔开

# 模板语法 用于实现模型与数据的绑定

文本绑定

{{data中的数据}} e.g. {{message}}

文本绑定也可以在元素添加属性 v-text="message"

html绑定

元素添加属性 v-html="message"

双向绑定 常用于表单控件

v-model="message"

绑定输入框：

<input v-model="message" placeholder="编辑我……">

绑定复选框：复选框如果是一个为逻辑值，如果是多个则绑定到同一个数组

(input type: "chec ked") 
< label checked 
kbox" 
- checkbox" value="Taobao" 
(input 
(label 
(input 
(label 
< input 
(label 
type-"chec kbox" id-" runoob" value-"Runoob" 
for-" 
for:" google") Google< 'label > 
type-- 
•a-model- "chec kedNames" > 
(span {{ checkedNarnes 
< / div) 
csc rip O 
new 
el: Wagg • , 
data: 
checked . 
false, 
chec kedNames: [l 

绑定单选按钮：

(input 
type: " radio" 
id=" runoob" 
v -model: "pic ked" > 
<label 
for-" runoob" / label > 
< input: 
type=" radio" 
dabel 
for." google" / label > 

绑定select 列表

(div 
< select ted" 
'option > 
value- "WWI.]. 
value: "NNW. 
(option 
(option 
(option 
< / select> 
value: 

**v**-model修饰符

**v-model.number,将输入的数据（默认字符串）自动转换为数字**

**v-model.trim 去除输入的前后空格**

**v-model.lazy 取消默认输入时就同步数据，改为change时同步**

绑定属性

v-bind: 可以直接简写为 冒号":"

e.g.1

<h1 v-text="message" v-bind:style="s1"></h1> s1的内容为“color:red”

e.g.2

<a :href="url" > 超链接</a> url为data中的数据，值可以为http:sina.com.cn

e.g.3

根据data中的数据use的值判断是否应用样式，如果false则不应用样式，true应用

<div v-bind:class="{'class1': use}">

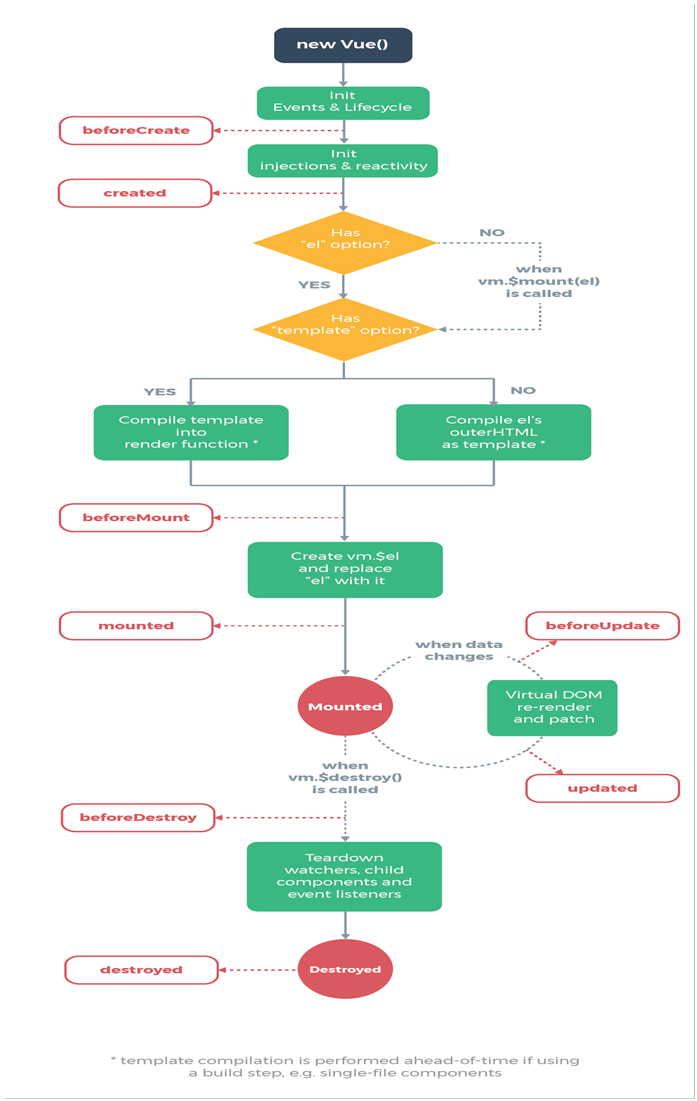


**注意：如果模板绑定的数组，修改数组的某个元素，并不一定能直接更新到视图，**

**但是，如果是删除、或者增加元素，则会更新上去。**

**解决不更新的办法是自己给自己赋值一次，如 this.arr=this.arr**

# 第三部分 生命周期



**1、created方法**，实例创建完成已经准备好数据，可以对数据进行操作，但是模板组件尚未绑定

**2、mounted方法**，模型已经绑定，业务逻辑一般在这里开始，也可以调用AJAX

**3、beforeDestory方法，**实例销毁以前调用。主要绑定一些使用addEventListener监听事件。清除计时器，清除非指令绑定事件等。

# 第四部分 过滤器

**1、语法**

filters:{

过滤器名:function(val){

过滤语句，返回处理后的数据

}

}

**2、调用 （多用于格式化字符串的结果）**

{{数据模型 | 过滤器名}}

使用| 管道进行调用，会将模型数据作为参数，传递给过滤器，然后将过滤器返回值显示出来

不对原始的数据模型做修改

2 基本指令与条件渲染指令

2020年8月28日

11:17

# 第一部分 模板语法

模板语法包含 **插值、表达式和指令** 组成

**1、插值**

<p>{{ userName }}</p> 双大括号插值

<p v-once>{{ userName }}</p> 只渲染一次数据

<p v-text="userName"></p> 等同于双大括号插值

<p v-html="userName"></p> 解析为html插值

<input type="text" v-model="userName" /> 双向绑定

**其中v-once使数据不在后续变化，其它会根据userName数据的变化，自动改变**

**2、表达式**

**仅支持单个表达式（应该有返回值），不支持语句和流控制，不能使用用户自定义的全局变量，只能使用js的全局变量。**

**3、指令**

v-if,v-bind,v-on，v-for等都是指令

指令可以有参数

如：<a v-bind:href="url"></a> <a v-on:click="doSomeThing">

指令中的修饰符

如：<form v-on:submit.prevert="onSubmit"></form>

后面会详细讲解

# 第二部分 绑定类样式、内联样式

Vue中对属性绑定，一般使用 **v-bind** 指令，可以缩写为 “：”

因为class,sytle都是属性，v-bind常用来操作clsss和style

**注意：v-bind:属性=属性值，**

**绑定多个属性的话可以写成 v-bind="attrs"，然后定义attrs数据，属性，属性值的形式**

**1、绑定方式分为对象语法，数组语法，（还有最普通的字符串）**

**字符串语法**

<p v-bind:class="test">111</p>

**数组语法**

<p v-bind:class="test2">111</p>

**对象语法**

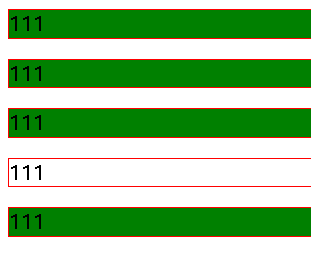
<p v-bind:class="{c1:on,c2:on}">111</p>

代码示例见2

**2、绑定class和style**

7 
8 
(script src= 
" •/js/vue 
9 
. cl border:lpx solid 
10 
.c2 background-color: 
11 
</style> 
12 
head > 
13 
14 
15 
(div id=" 
app" > 
v-bind : 
16 
cp v-bind:class= 
17 
-bind : class= 
18 
19 
20 
21 
22 
</html> 
23 
<script> 
24 
25 
26 
28 
29 
30 
31 
32 
33 
34 
35 
36 
'red} 
a 
green 
*tæhfit$ßRClass 
NH"it9ßRClass 
(p v-bind : 
(p v-bind : style—I' [classl, class2] 
var vm=new Vue 
el : "#app" , 
data : 
test: "cl c2", 
test2: "cl 
on: true, 
classl 
: {"border": "Ipx solid red" 
class2: { "background-color"• "green" 

显示结果如下



# 第三部分 条件渲染指令

v-if,v-else-if,v-else,v-show

(div 
(div - 
< / div) 
(div v-else-if-"type 
< (div) 
(div v-else-if-"type 
</diV) 
(div v-else> 
Not A/g/C 
c/div> 
< / div > 
< sc ript> 
el: 'ttapp', 
data : 
type : 
< / script) 

注意：V-show与v-if类似，条件成立则显示，不成立则不显示，区别在于，带有v-show指令的元素始终会被渲染保留在DOM中，只是控制他的diplay属性为none.

因此，v-show不支持<template>(因为始终会被渲染)，也不支持v-else

# 第四部分 事件绑定

**1、v-on: 指令 缩写 “@”，用于监听DOM事件**

**除了可以绑定methods参数内定义的方法外，还支持内联js语句，但是仅限一条。**

**一个元素事件可以使用v-on绑定多个事件，用","或“；”隔开**

(label " 
(input type=" number" " " > 
<label </label> 
(input type=" number" v-mode1="num2" id="txtNum2" > 
(button 
<button 
<button 
(button 
v-on: > b 
</butt 
@click="eval(vm. 
. numl *vm. num2) 
<1abe1 
<input type=" number" v-model="result" id="txtResu1t"> 
<script src=" 
<script> 
= new Vue 
var vm 
el. 
"#app"' 
data: 
num1:Ø, 
num2: 
e, 
result : e 
methods : 
doAdd : function e 
this . res ult=Number (this . numl ) +Number ( this . num2 
console . " 
doSub•.function e 
this . result—this . numl- this . num2; 
console . log 
doDiv:function e 
try{ 
this . result=this . numl/this . num2; 
}catch(ex 
alert ex. message ; 
console . log 
</script> 

**2、事件修饰符**

Vue.js 为 v-on 提供了事件修饰符来处理 DOM 事件细节，如：event.preventDefault() 或 event.stopPropagation()。

Vue.js 通过由点 **.** 表示的指令后缀来调用修饰符。可以使用.链式写法，写多个修饰符

**事件修饰符列表**

.stop - 阻止冒泡 （即事件不向上传递，如 div中的button的click.stop事件，不会触发div的click事件）

.prevent - 阻止默认事件

.capture - 阻止捕获

.self - 只监听触发该元素的事件 （只能通过元素自身的事件触发而不能通过子元素事件）

.once - 只触发一次

**以下仅用于鼠标事件修饰符，如click\mousedown\mouseup等**

.left - 左键事件

.right - 右键事件

.middle - 中间滚轮事件

**键盘事件修饰符，注意，如果使用链式写法写多个键，可以形成组合键**

可以用keycode或者按键别名，用作键盘键盘事件如click\keyup\keydown的修饰符，来判断按下什么键才响应事件。

示例

<p><!-- Alt + C -->  
<input @keyup.alt.67="clear" /> clear是写好的清空事件  
</p>

<!-- Ctrl + Click -->  
<div @click.ctrl="doSomething">Do something</div>

**按键别名与按键kecode表 请查看附录**

VS Code使用

ctrl + "+"放大 ctrl + "-"缩小

链式写法

Div#d1>p.red\*3>a

3 v-for\计算属性\侦听属性

2020年1月18日

16:30

# 第一部分 v-for

遍历数组 v-for="(item,index) in list" :key="index"

遍历对象 v-for="(value,name,index) in user"

# 第二部分 计算属性 computed

**语法**

computed:{

computed1:function(){..}

}

**模板调用** {{computed1}}

**与方法的区别**

1、只能通过名称调用，保存的是数据，本身不是方法，不可以通过 名称(参数)调用

一般声明的时候不带参数。

2、再遍历大数组或者做大量计算时，计算属性更高效

**list.filter()方法的说明，如下例**

\_this=this;

return this.items.filter(function(val){return val.indexOf(\_this.wordKey) !=-1});

其中val代表items中的每一项，如果unction返回false，则该项val从items中剔除，否则则保留

注意**list.filter()方法**不同于Vue过滤器

# 第三部分 侦听属性 watch

**基本语法：**

watch:{

数据名:function(新值，旧值){ 语句 }，

…

}

**注意：**

**1、watch只监听数据变化，最初绑定的时候并不会执行。**

**如果想要以开始让最初绑定值的时候就执行，需要使用handler方法和immediate属性**

**2、当模型数据为对象，默认不能侦听对象的属性变化，如果要侦听，则需要使用deep属性，默认false，设置为true即可**

watch:{

数据名:{

handler:function(新值，旧值){ 语句 },

immediate:true, //默认为flase

deep:true

}

…

}

4 Vue组件Component

2020年2月4日

10:17

# 第一部分 组件的注册和使用

**一、实现步骤和语法：**

**创建组件：**有三种方法，字符串方式，js，template标签方法

**注册组件：**注册为全局或者局部组件。

**使用组件：**在组件中直接使用标签。

< head> 
<meta charset=" utf-8" > 
<tit1e>Vue - . 
(div 
template: " #temp2"} 
template: " #temp3"} 
template : "#temp2" 
template : "#temp3" 
<script 
</head > 
src-" ./js/vue . min . js " 
compl> 
<my-comp2></my-comp2> 
<my-comp3></my-comp3> 
<my- my- comp4> 
<my-comp5></my-comp5> 
3, 
<template id="temp2"> 
<div> 
<script • - fiii3 
</div> 
var templ=" 
Vue. component 
"my- compl " , 
Vue. component 
"my- comp2" , 
Vue. component 
"my- comp3" , 
template: templ 
new Vue 
el: "#a 
components : 
"my- comp4" • 
"my- comp5 " • 
"my- comp6" • 
</script> 
template : templ 

**二、组件中的Data数据**

必须使用function()返回数据

**data:function(){**

**return {**

**属性：属性值,**

**…**

**}**

**}**

**组件中的数据与vm实例中的数据不能直接互相访问。**

new Vue 
el: "#app" , 
components : 
"my- comp4" : 
"my- comps " : 
"my- comp6" : 
<template 
</div> 
template 
<script type="text/x-template" • — 
Id—" 
<div> 
(script) 
"my- compl " , {template : templ, data : function 
template: "#temp3" 
var 
templ=" hl 
Vue . component 
Vue . component 
Vue . component 
return( msg: " 
"my- comp2" , {template : "#temp2" , data : function 
"my- comp3" , {template : "#temp3 " , data : function( ) { 
return 
return 
msg: 
msg : 
return{ msg: 
template : templ, data : function 
template: "#temp2" , data: msg: 
"{Efr6cF%MiÉ"}}} 
data : function 
return{ msg: 

# 

模 板 1 组 件 1 中 的 数 据 
模 板 2 组 件 2 中 的 数 据 
模 板 3 组 件 3 中 的 数 据 
模 板 1 组 件 4 中 的 数 据 
模 板 2 组 件 5 中 的 数 据 
模 板 3 组 件 6 中 的 数 据 

# 

# 第二部分 组件之间的通讯

**一、父子组件**

**在注册组件的时候，还可以在其中注册子组件。**

**最大的父元素可以认为时vm对象**

**二、使用props属性向子组件传递数据**

**组件是相互独立的，可以通过props属性把父组件数据传递给子组件**

**实现方法：**

**示例一：vm数据msg传递到父组件（1234步） （父组件是vm的子组件）**

**示例二：父组件传递pmsg到子组件 （567步）**

<div id=" app" > 
<parent-temp : pmsg=msg < parent-temp) 
</div> 
<template 
<hl>{{pmsg}}</hl> 
<child -temp : submsg=pmsg></child-temp> 
</div> 
template 
<template " > 
<hl>{{submsg}}</hl> 
</template> 
var vm=new Vue 
el: "#app" , 
components : 
" parent-temp " • 
[ " pmsg " 
props : 
data: function 
return 
template: "#tempp" , 
components : 
"child- temp" : 
props: " submsg " 
7, 
pmsgl : " cde 
template : "#tempc" 
data : 
msg 
</script> 
: " aaa" 

# 

# 

# 

# 第三部分 插槽

1、**默认插槽**和**具名插槽**。

从前面的的练习和学习可以看除，组件不够灵活，标签中加入内容不能显示，会被忽略。

灵活的使用组件，需要使用插槽。即在创建组件时，插入slot标签。

插槽分为**默认插槽**和**具名插槽**。

<div id="app"> 
<temp-f> 3. 
tri, 
(template 
<template 
<template id="temp"> 
<div> 
<div> 
</div> 
</div> 
</template> 
c/ body) 
html > 
var vm=new Vue 
el : "#app" , 
components : 
"temp-f" . 
(slot 
slot </slot> 
(slot 
2. 
template : " #temp " 
</script> 

头 部 唐 
默 认 槽 
0 部 槽 

**2、作用域插槽**

**插槽的一个重要作用：可以将子组件绑定的数据，传递给父组件**

(div id=" app" > 
<temp-f> 3s 
<template slot="fruit" 
< li 
v-for="f in props 
</div> 
(template id="temp"> 1, 
<div> 
(slot 
<script> 
var vm=new Vue 
el: "#app" 
components : 
temp — f • • : 
data: function 
return flist: ' 
template: "#temp" 
s lots 

\*Vue Router

2020年9月1日

16:54

# 第一部分 简介，Vue路由可以实现单页应用

**一、Vue.js 路由**

本章节我们将为大家介绍 Vue.js 路由。

Vue.js 路由允许我们通过不同的 URL 访问不同的内容。

通过 Vue.js 可以实现多视图的单页Web应用（single page web application，SPA）。

Vue.js 路由需要载入 [vue-router 库](https://github.com/vuejs/vue-router)。

中文文档地址：[vue-router文档](http://router.vuejs.org/zh-cn/)。

**安装**

**1、直接下载 / CDN**

<https://unpkg.com/vue-router/dist/vue-router.js>

**NPM**

推荐使用淘宝镜像：

cnpm install vue-router

**二、简单实例 路由实现过程**

Vue.js + vue-router 可以很简单的实现单页应用。

实现步骤

(script 
<script 
</head> 
<body> 
(div 
src . /js/ vue . min . js " script > 
"./js/ vue-router . js" 
(router-link 
<router-link 
<router-link to="/page2" 
(router- view> 
body> 
var home= template. 
var " 
var : " 
const routes= 
path: • / 
• " , name: "index" , component: home 
• "pagel" , component : pltemp} , 
path: " / pagel " , 
name . 
path: " / page2" , name : "page2" , component : p2temp 
const VueRouter 
routes 
/ I—RI, 2, 
const app—new Vue 
router 
. $mount "#app" 
script>x 
< / html > 

# <router-link> 相关属性

接下来我们可以了解下更多关于 <router-link> 的属性。

**to**

表示目标路由的链接。 当被点击后，内部会立刻把 to 的值传到 router.push()，所以这个值可以是一个字符串或者是描述目标位置的对象。

<!-- 字符串 -->  
<router-link to="home">Home</router-link>  
<!-- 渲染结果 -->  
<a href="home">Home</a>

<!-- 使用 v-bind 的 JS 表达式 -->  
<router-link v-bind:to="'home'">Home</router-link>

<!-- 不写 v-bind 也可以，就像绑定别的属性一样 -->  
<router-link :to="'home'">Home</router-link>

<!-- 同上 -->  
<router-link :to="{ path: 'home' }">Home</router-link>

<!-- 命名的路由 -->  
<router-link :to="{ name: 'user', params: { userId: 123 }}">User</router-link>

<!-- 带查询参数，下面的结果为 /register?plan=private -->  
<router-link :to="{ path: 'register', query: { plan: 'private' }}">Register</router-link>

**replace**

设置 replace 属性的话，当点击时，会调用 router.replace() 而不是 router.push()，导航后不会留下 history 记录。

<router-link :to="{ path: '/abc'}" replace></router-link>

**append**

设置 append 属性后，则在当前 (相对) 路径前添加其路径。例如，我们从 /a 导航到一个相对路径 b，如果没有配置 append，则路径为 /b，如果配了，则为 /a/b

<router-link :to="{ path: 'relative/path'}" append></router-link>

**tag**

有时候想要 <router-link> 渲染成某种标签，例如 <li>。 于是我们使用 tag prop 类指定何种标签，同样它还是会监听点击，触发导航。

<router-link to="/foo" tag="li">foo</router-link>  
<!-- 渲染结果 -->  
<li>foo</li>

**active-class**

设置 链接激活时使用的 CSS 类名。可以通过以下代码来替代。

<style>  
 .\_active{  
 background-color : red;  
 }  
</style>  
<p>  
 <router-link v-bind:to = "{ path: '/route1'}" active-class = "\_active">Router Link 1</router-link>  
 <router-link v-bind:to = "{ path: '/route2'}" tag = "span">Router Link 2</router-link>  
</p>

注意这里 **class** 使用 **active\_class="\_active"**。

**exact-active-class**

配置当链接被精确匹配的时候应该激活的 class。可以通过以下代码来替代。

<p>  
 <router-link v-bind:to = "{ path: '/route1'}" exact-active-class = "\_active">Router Link 1</router-link>  
 <router-link v-bind:to = "{ path: '/route2'}" tag = "span">Router Link 2</router-link>  
</p>

**event**

声明可以用来触发导航的事件。可以是一个字符串或是一个包含字符串的数组。

<router-link v-bind:to = "{ path: '/route1'}" event = "mouseover">Router Link 1</router-link>

以上代码设置了 event 为 mouseover ，及在鼠标移动到 Router Link 1 上时导航的 HTML 内容会发生改变。

**NPM 路由实例**

接下来我们演示了一个使用 npm 简单的路由实例，开始前，请先下载该实例源代码：

[路由实例](http://static.runoob.com/download/vue-2.0-simple-routing-example-master.zip)

你也可以在 Github 上下载：<https://github.com/chrisvfritz/vue-2.0-simple-routing-example>

下载完后，解压该目录，重命名目录为 vue-demo，vu 并进入该目录，执行以下命令：

# 安装依赖，使用淘宝资源命令 cnpm  
cnpm install

# 启动应用，地址为 localhost:8080  
cnpm run dev

如果你需要发布到正式环境可以执行以下命令：

cnpm run build

执行成功后，访问 <http://localhost:8080> 即可看到如下界面：

Home About 
Welcome home 

\*Vue AJAX(axios)

2020年9月1日

16:54

5-6 Ant Design of Vue

2020年9月1日

16:54

下载antd.min.js，antd.min.css

js文件夹下需要包含

antd.min.js 
antd.min.js.map 
antd-with-locdes.min.js 
antd-with-locales.minjs.map 
vue.minjs 

cli项目可以安装Antdesign

# 第一部分 组件的使用

一、组件使用语法

<a- 组件名 属性=值></a- 组件名>

# 第二部分 布局 组件（布局、导航菜单、面包屑）

**一、布局组件**

布局组件包含尺寸，交互，视觉3个原则。

<a-layout>

<a-layout-header></a-layout-header>

<a-layout>

<a-layout-sider></a-layout-sider>

<a-layout-content></a-layout-content>

</a-layout>

<a-layout-footer></a-layout-footer>

</a-layout>

**sider组件的属性**

|  |  |
| --- | --- |
| **属性** | **说明** |
| breakpoint | 触发响应式布局的断点 |
| class | 表示class样式 |
| collapsed | 表示当前收起状态 |
| collapsedWidth | 默认值为80，表示收缩宽度，设置为0会出现特殊trigger |
| collapsible | 默认值为false，表示是否可收起 |
| defaultCollapsed | 默认值为false，表示是否默认收起 |
| reverseArrow | 默认值为false，表示是翻转折叠提示箭头的方向，当Sider在右边时可以使用 |
| style | 指定样式 |
| theme | 默认值为dark，表示主题颜色，只有两个值light和dark |
| trigger | 表示自定义trigger，设置为null时隐藏 |
| width | 默认值为200，表示宽度 |

**布局组件可用的事件**

|  |  |
| --- | --- |
| **事件** | **说明** |
| collapse | 展开-收起时的回调函数 |
| breakpoint | 触发响应式布局断点时的回调函数 |

**二、导航菜单组件<a-menu> <a-menu-item>**

**导航菜单组件**常用属性如下面表

|  |  |
| --- | --- |
| **属性** | **说明** |
| defaultOpenKeys | 表示初始展开的SubMenu菜单项key数组 |
| defaultSelectedKeys | 表示初始选中的菜单项key数组 |
| mode | 默认值是vertical，表示菜单类型，现在支持垂直（vertical）、水平（horizontal）和内嵌模式（inline）三种。 |
| multiple | 默认值是false，表示是否允许多选 |
| selectable | 默认值是true，表示是否允许选中 |
| theme | 默认值是light，表示主题颜色，只有light和dark颜色 |

导航**菜单项**包含的属性

|  |  |
| --- | --- |
| **属性** | **说明** |
| disabled | 表示是否禁用 |
| key | 表示唯一标志 |
| title | 表示设置收缩时展示的悬浮标题 |

* 注意，使用title效果不好，因为鼠标指向会右文字框提示
* 可以在<span slot="title">标题<span>的方式实现

<a-menu mode="vertical" theme="light"> //vertical\light为默认值

<a-menu-item key=1>眼镜</a-menu-item>

<a-sub-menu key=2>

<span slot="title">服装</span> //注意，也可以直接在sub-menu的属性写 title=服装，区别是后者指向会显示元素名称服装

<a-menu-item key=3>男装</a-menu-item>

<a-menu-item key=4>女装</a-menu-item>

</a-sub-menu>

</a-menu>



**mode="inline" theme="dark"**

男 装 
女 装 

mode="horizontal" theme="light"

foc 臨 
一 Ite 

**三、面包屑组件 <a-breadcrumb> <a-breadcrumb-item>**

<a-breadcrumb> 
<a-breadcrumb 
a -breadcrumb-item> 
«a - a -breadcrumb-item> 
Ka - a - breadcrumb-item> 
Ka - a - breadcrumb-item> 
/ a-breadcrum+l 



# 第三部分 按钮和全局提示组件

**一、全局提示组件**

$message.success(content,[duration],onClose)

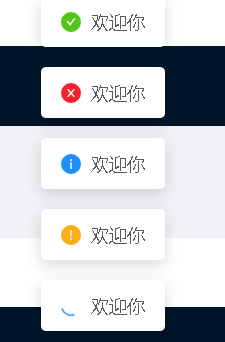
$message.error(content,[duration],onClose)

$message.info(content,[duration],onClose)

$message.warn(content,[duration],onClose)

$message.loading(content,[duration],onClose)

|  |  |
| --- | --- |
| **参数** | **说明** |
| content | 提示内容 |
| duration | 自动关闭的延时，单位秒。设为0时不自动关闭。 |
| onClose | 关闭时触发的回调函数 |



**二、按钮组件<a-button>**

**按钮组件的属性**

|  |  |
| --- | --- |
| **属性** | **说明** |
| disabled | 表示按钮失效状态 |
| ghost | 表示幽灵按钮，即是否使按钮背景透明。 |
| htmlType | 设置button原生的type值，可选值请参考HTML标准 |
| icon | 设置按钮的图标类型 |
| loading | 设置按钮载入状态 |
| shape | 设置按钮形状，可选值为circle或者不设 |
| size | 设置按钮大小，可选值为small（小）、large（大）或者不设 |
| type | 设置按钮类型，可选值为primary、dashed或danger |
| block | 将按钮宽度调整为其父宽度的选项 |

Smessage.success 
Smessage,error 
$message,info 
Smessage,warn 

# 第四部分 表格组件

**一、表格组件常用属性**

|  |  |
| --- | --- |
| **属性** | **说明** |
| bordered | 是否展示外边框和列边框 |
| childrenColumnName | 指定树形结构的列名 |
| columns | 表格列的配置描述 |
| dataSource | 数据数组 **注意在html中绑定需要写成 data-source** |
| defaultExpandAllRows | 初始时，是否展开所有行 |
| defaultExpandedRowKeys | 默认展开的行 |
| showHeader | 是否显示表头 |
| size | 正常或迷你类型 |
| title | 表格标题 |
| scroll | 设置横向或纵向滚动，也可用于指定滚动区域的宽和高 |

**二、表格组件常用事件**

|  |  |
| --- | --- |
| **事件** | **说明** |
| expandedRowsChange | 展开的行变化时触发函数 |
| change | 分页、排序及筛选变化时触发函数 |
| expand | 点击展开图标时触发函数 |

**三、表格列属性**

|  |  |
| --- | --- |
| **属性** | **说明** |
| title | 列头显示文字 |
| width | 列宽度 |
| slots | 使用columns时，可以通过该属性配置支持slot的属性，如slots: { filterIcon: “XXX”} |
| scopedSlots | 使用columns时，可以通过该属性配置支持slot-scope的属性，如scopedSlots: { customRender: “XXX”} |
| align | 设置列内容的对齐方式left、rigt和center，默认值是left |
| colSpan | 表头列合并，当设置为0时，表示不渲染列合并 |
| dataIndex | 列数据在数据项中对应的key，此属性是必须设置属性 |
| key | Vue需要的key，如果已经设置了唯一的dataIndex，可以忽略这个属性 |

<a-table v-bind : columns="columns" " 
(span 
href— 
—" j avas cript : doRemove (text) ; " a > 
/ a-table 
bordered > 

datal 
2, 
id: 
id: 
1, 
name : 
n ame 
name: ' 
age : 
age: 
age: 
32, 
33, 
address: 
address: 
address: ' 

# 

# 

# 

methods: { 
doRemove:function(id){ 

# 

# 第五部分 栅格组件

栅格组件，将一行分为24列

**一、列属性**

|  |  |
| --- | --- |
| 属性 | 说明 |
| xs | 可视区宽度<576px 响应式栅格，可为栅格数或一个包含其他属性的对象 |
| sm | 可视区宽度≥576px 响应式栅格，可为栅格数或一个包含其他属性的对象 |
| md | 可视区宽度≥768px 响应式栅格，可为栅格数或一个包含其他属性的对象 |
| lg | 可视区宽度≥992px 响应式栅格，可为栅格数或一个包含其他属性的对象 |
| xl | 可视区宽度≥1200px 响应式栅格，可为栅格数或一个包含其他属性的对象 |
| xxl | 可视区宽度≥1600px 响应式栅格，可为栅格数或一个包含其他属性的对象 |
| offset | 栅格左侧的间隔格数，间隔内不可以有栅格 |
| order | 栅格顺序，flex布局模式下有效 |
| pull | 栅格向左移动格数 |
| push | 栅格向右移动格数 |
| span | 栅格占位格数，为0时相当于display: none |

**二、行属性**

|  |  |
| --- | --- |
| **属性** | **说明** |
| align | flex布局下的垂直对齐方式：top、middle和bottom |
| gutter | 栅格间隔，可以写成像素值或支持响应式的对象写法{ xs: 8, sm: 16, md: 24} |
| justify | flex布局下的水平排列方式：start、end、center、space-around和space-between |
| type | 布局模式，可选flex，在现代浏览器下有效 |

type=flex justify=space-around需要配合使用

<a-row > 
<a-col span-IO style="background-color: 
<a-col span-IO offset* style="background-color: Orosybrown">l</a-col> 
< / a-row) 

# 第六部分 输入组件

**一、输入框组件**

输入框自建常用属性

|  |  |
| --- | --- |
| 属性 | 说明 |
| prefix | 设置带有前缀图标的输入框 |
| size | 输入框大小，标准表单内的输入框大小限制为large。可选值为large、default、small、string和default |
| suffix | 设置带有后缀图标的输入框 |
| type | 声明输入框类型，同原生input标签的type属性一致 |
| value | 输入框内容 |
| addonAfter | 设置带后置标签的输入框 |
| addonBefore | 设置带前置标签的输入框 |
| defaultValue | 输入框默认内容 |
| disabled | 是否禁用状态，默认为false |
| id | 输入框的唯一标示 |

<a-input > 
<a-icon slot="prefix" 
<a-input > 
<a-icon 
</a- input) 
<a-input defaultVa1ue= 
<span 
<span 
</a-input> 
<a-input > 
</a- input> 
<a-button type="primary" 



# 二、选择器组件（下拉菜单）

选择器组件的常用属性

|  |  |
| --- | --- |
| **属性** | **说明** |
| allowClear | 支持清除 |
| autoClearSearchValue | 是否在选中项后清空搜索框，只在mode为multiple或tags时有效 |
| autoFocus | 默认获取焦点 |
| defaultValue | 指定默认选中的条目 |
| dropdownClassName | 下拉菜单的className属性 |
| mode | 设置Select的模式为多选或标签，可选值为default、multiple、tags和combobox |
| placeholder | 选择框默认文字 |
| showSearch | 使单选模式可搜索 |
| showArrow | 是否显示下拉小箭头 |
| size | 选择框大小，可选值为large和small |
| value | 指定当前选中的条目 |

选择器组件的常用事件

|  |  |
| --- | --- |
| **事件** | **说明** |
| blur | 失去焦点时触发事件 |
| change | 选中option或input的value变化（combobox模式下）时，触发事件 |
| deselect | 取消选中时触发事件，参数为选中项的value（或key）值，仅在multiple或tags模式下生效 |
| focus | 获得焦点时触发事件 |
| search | 文本框值变化时触发事件 |
| select | 被选中时触发事件，参数为选中项的value（或key）值 |

# 级联菜单

<a-select v-model="selectp" 
<a-select 
< / a-select> 
: options="proviences" " > 

data: 
selectp: "O" , 
selectc: "0" , 
proviences : 
: "i}JÅt" , 
label 
value. " 
label: " , value : 
citys • 
label 
: , value. " " 
label 
" Iue 
, va 
{label. 
• , value: " , 
citiData: 
label 
: , value: "0" , 
label 
" v Iue 
label. 
value: "2" , 
label 
, value: "0" 
label. 
" lue . " 
• " ÉßEl" , value: "2" , 
label. 

doChange : function ( value 
this . selectp=value; 
this . citys —this . 
this . ; 
doChange2 : function 
this .$mess age . info( 
+this . proviences this . selectp) . label 
. selectp+ 
+this . citys this . selectc) . label 
"+this . selectc) 

# 三、单选框按钮

**Radio属性**

|  |  |
| --- | --- |
| 属性 | 说明 |
| autofocus | 自动获取焦点 |
| checked | 指定当前是否选中 |
| defaultChecked | 初始是否选中 |
| value | 根据value进行比较，判断是否选中 |

**RadioGroup属性**

|  |  |
| --- | --- |
| 属性 | 说明 |
| defaultValue | 默认选中的值 |
| disabled | 禁选所有子单选器 |
| name | RadioGroup下所有input[type="radio"]的name属性 |
| options | 以配置形式设置子元素 |
| size | 大小，只对按钮样式生效，可选值large、default和small |
| value | 用于设置当前选中的值 |
| buttonStyle | RadioButton的风格样式，目前有描边和填色两种风格 |

(a-radio-group 
<a-radio-button 
-radio-button) 
<a-radio-button 
-radio-button) 
(a-radio-button 
-radio-button > 
(a-radio-button 
-radio- 
(a-radio-button 
-radio-button > 
<a-radio-group v-mode1="resu1t2"> 
(a-radio value="a" 
(a-radio value="b" 
(a-radio 
a-radio 
<a-radio 
<a-radio 
< / a-radio-group) 
<a-radio-group v-mode1="resu1t3" : options —"items 
a-radio-button 

data : 
resultl 
result2: "a" 
result3: "a" , 
items : 
{label : " , value: " 
a"} 
label : "7jæ " , value • 
label : 
" " , value 
c'} 
label : " " , value. " 
label: " , value. " , 

木 星 水 星 火 星 上 星 
木 星 
· 全 星 
火 星 
木 星 
土 星 
土 星 

# 四、复选框组件

注意：a-checkbox为多选按钮，没有按钮效果

# 第七部分 表单组件

**一、表单域属性**

|  |  |
| --- | --- |
| 属性 | 说明 |
| label | label标签的文本 |
| labelCol | label标签布局，同<a-col>组件，设置span和offset值，如{span: 3, offset: 12}或sm:{span: 3, offset: 12} |
| required | 是否必填，如不设置，则会根据校验规则自动生成 |
| validateStatus | 校验状态，如不设置，则会根据校验规则自动生成，可选：success、warning、error和validating |

|  |  |
| --- | --- |
| colon | 配合label属性使用，表示是否显示label后面的冒号 |
| extra | 额外的提示信息，和help类似，当需要错误信息和提示同时出现时，可以使用这个属性 |
| hasFeedback | 配合validateStatus属性使用，展示校验状态图标，建议只配合输入框组件使用 |
| help | 提示信息，如不设置，则会根据校验规则自动生成 |

**二、验证过则**

|  |  |
| --- | --- |
| 属性 | 说明 |
| enum | 枚举类型 |
| len | 长度 |
| max | 最大长度 |
| min | 最小长度 |
| message | 错误提示 |
| pattern | 正则校验 |
| required | 必填或必选 |
| transform | 校验前转换字段值 |
| validator | 自定义校验处理方法 |
| whitespace | 必选时，空格是否视为错误 |

(a-form v-bind : 
(a-form-item label=" 
<a-input 
"email" , 
rules: [ 
a-form-item) 
type : "email" 
message : " 
required : true, 
message : 
<a-form-item 
<a- input type="password " 
"pwd", 
rules : [ 
v-decorator= 
required : true, 
message 
len:6, 
message: 
a-form-item> 
(a-form-item 
input " 
"pwdag " 
rules: [ 
v-decorator= 
required : true, 
message : 
validator: compareToFirstPassword 
a-form-item> 
<a-form-item 
<a-input v-decorator='[ 
"nike " , 
rules: [ 
required : true, 
message : 
whites pace : true, 
message . 
max: 10, 
message 
min : 2, 
message . 
a-form-item> 
<a-form-item> 
ca- checkbox v -decorator= 
" agreement " , 
rules : [ 
required : true, 
message : 
a-form-item> 
<a-button type="primary" html-type="submit" 
a- form) 

doSubmit: function e 
e . preventDefau1t 
this . form.va1idateFie1dsAndScr011( err, values 
this .$mess age . info ( " " ) 
console. log( 'Received values of form: 
compareToFirstPassword rule, value, 
callback 
const form = this . form; 
if (value && value ! form.getFie1dVa1ue( 'pwd ' 
callback ( " 
else 
callback 
beforeCreate : function 
this .$form. create Form this 
this . form= 
values 

00 地 址 不 能 力 窒 
登 录 码 
马 不 力 仝 
种 稱 确 认 
马 确 认 不 力 窒 
用 户 
这 须 墳 写 昵 你 
我 已 同 后 权 0 议 
、 勾 选 同 0 

# 第八部分 树形组件

**一、树形组件的树形**

|  |  |
| --- | --- |
| **属性** | **说明** |
| treeData | 树形组件数据源 |
| autoExpandParent | 是否自动展开父节点 |
| checkable | 节点前添加Checkbox复选框 |
| defaultCheckedKeys | 默认选中复选框的树节点 |
| defaultExpandAll | 默认展开所有树节点 |
| disabled | 设置树是否禁用 |
| draggable | 设置节点是否可拖拽 |
| multiple | 设置是否支持点选多个节点 |
| showIcon | 设置是否展示树节点标题前的图标 |
| showLine | 设置是否展示连接线 |

**二、树形组件的事件**

|  |  |
| --- | --- |
| **事件** | **说明** |
| check | 点击复选框触发 |
| dragstart | 开始拖拽时触发 |
| dragend | 结束拖拽时触发 |
| expand | 展开/收起节点时触发 |
| load | 节点加载完毕时触发 |
| rightClick | 在树节点上点击右键触发 |
| select | 点击树节点触发 |

**三、树节点的属性**

|  |  |
| --- | --- |
| **属性** | **说明** |
| disableCheckbox | 禁用树节点的复选框 |
| disabled | 禁用树节点 |
| icon | 设置自定义图标 |
| key | 树节点的唯一标示 |
| selectable | 设置节点是否可被选中 |
| title | 树节点标题 |

**注意：在下例中，v-bind:属性=属性值，绑定多个属性的话可以写成 v-bind="attrs"，然后定义attrs数据，属性，属性值的形式**

<a-layout-content> 
v-bind="attrs" @se1ect="onSe1ect" @check="onCheck"></a-tree> 
ca-tree 
< / a-layout-content) 

var Vue 
el: "#app" , 
data : 
attrs : 
treeData : 
title: , 
title: "3±2 
" , key: " news2- 
key: " newsl " , 
children: 
title . 
title . 
title • 
• , key : " newsl-l " 
key 
key 
{title: "$3" , key : " news2-2" 
{title: " , key : " news2-3"} 
: " newsl-2" 
: " newsl-3" 
title : 
key: " news 2" , 
children: 
multiple : true , 
checkable: true, 
showLine : true , 
showlcon : true , 
1"} 
defaultExpandedKeys: ["newsl , 
" "news 2" 
defaultCheckedKeys: C "newsl 1 , 
- " "news2-1" 

onSe1ect: function selectKeys , info 
console . log ( " selected : " , selectKeys , info 
onCheck checkedKeys, info 
console . log( " checked : " , checkedKeys , info 

api+vue cli项目开发流程

2020年8月17日

15:15

=====================

core api流程

======================

1、创建core api项目

2、管理nuget包

安装

Microsoft.EntityFrameworkCore

Microsoft.EntityFrameworkCore.SqlServer：Sql Server数据库EF提供程序。

Microsoft.EntityFrameworkCore.Design：设计时使用到的EF共享库。

Microsoft.EntityFrameworkCore.Tools：EF的NuGet包管理器命令工具。

3、创建models文件夹

编写实体类

编写上下文文件

using Microsoft.EntityFrameworkCore;

namespace XiangmuApp.Models

{

public class ProjectContent:DbContext

{

public DbSet<User> Users { get; set; }

public DbSet<Project> Project { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer("Data Source=.;Initial Catalog=ProjectDB;Integrated Security=True");

}

}

}

4、工具-nuget包管理，控制台，生成数据库

Add-Migration：创建准备迁移的框架代码。

输入一个名字

Update-Database：将任何挂起的迁移应用到数据库。

数据库已经生成，可以插入数据

5、创建控制器 添加其操作使用API的控制器

using Microsoft.Extensions.Logging; //日志

using XiangmuApp.Models;

namespace XiangmuApp.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class UsersController : ControllerBase

{

private readonly ProjectContent \_context;

private ILogger<UsersController> \_logger;

public UsersController(ProjectContent context, ILogger<UsersController> logger)

{

\_context = context;

\_logger = logger; ///应用日志

\_logger.LogInformation("获取日志对象和context对象");

}

[HttpPost]

public async Task<ActionResult<User>> Login([FromBody]User user)

{

var \_user = await \_context.Users.FirstOrDefaultAsync(u => u.UserName == user.UserName && u.Password == user.Password);

if (\_user == null)

{

return NotFound();

}

\_user.Password = null;

return \_user;

}

}

6、修改startup.cs文件

依赖注入

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

//===========================

services.AddDbContext<ProjectContent>(); //数据库

services.AddCors(option => //跨域

{

option.AddPolicy("mycors", policy =>

{

policy.AllowAnyOrigin()

.AllowAnyHeader()

.WithMethods("GET", "POST", "HEAD", "PUT", "DELETE", "OPTIONS");

});

});

}

app.UseRouting();

app.UseCors("mycors");//跨域

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers().RequireCors("mycors");//所有请求都应用跨域请求策略

});

================

vue项目

================

1、创建项目 vue create 项目名称 选手动，选择则router和vuex

2、vscode 打开项目

在终端窗口中执行命令安装AXJX操作库

npm install axios --save

在终端窗口中执行命令安装ant-design-vue UI库

npm install ant-design-vue --save

安装router库

npm install vue-router

3、修改main.js

import Vue from 'vue'

import App from './App.vue'

import router from './router'

import store from './store'

import Antd from 'ant-design-vue'

import 'ant-design-vue/dist/antd.css'

import axios from 'axios'

Vue.config.productionTip = false

Vue.use(Antd)

Vue.prototype.$axios = axios

new Vue({

router,

store,

render: h => h(App)

}).$mount('#app')

启动项目 npm run serve

4、ant控件网址

<https://www.antdv.com/components/form-cn/>

按键别名和keycode

2020年8月28日

15:06

# 按键别名

.enter

.tab

.delete (捕获 "删除" 和 "退格" 键)

.esc

.space

.up

.down

.left

.right

.ctrl

.alt

.shift

.meta

# Keycode对照表（键码对照表）

**字母和数字键的键码值(keyCode)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **按键** | **键码** | **按键** | **键码** | **按键** | **键码** | **按键** | **键码** |
| A | 65 | J | 74 | S | 83 | 1 | 49 |
| B | 66 | K | 75 | T | 84 | 2 | 50 |
| C | 67 | L | 76 | U | 85 | 3 | 51 |
| D | 68 | M | 77 | V | 86 | 4 | 52 |
| E | 69 | N | 78 | W | 87 | 5 | 53 |
| F | 70 | O | 79 | X | 88 | 6 | 54 |
| G | 71 | P | 80 | Y | 89 | 7 | 55 |
| H | 72 | Q | 81 | Z | 90 | 8 | 56 |
| I | 73 | R | 82 | 0 | 48 | 9 | 57 |

**数字键盘上的键的键码值(keyCode)**

|  |  |  |  |
| --- | --- | --- | --- |
| **按键** | **键码** | **按键** | **键码** |
| 0 | 96 | 8 | 104 |
| 1 | 97 | 9 | 105 |
| 2 | 98 | \* | 106 |
| 3 | 99 | + | 107 |
| 4 | 100 | Enter | 108 |
| 5 | 101 | - | 109 |
| 6 | 102 | . | 110 |
| 7 | 103 | / | 111 |

**功能键键码值(keyCode)**

|  |  |  |  |
| --- | --- | --- | --- |
| **按键** | **键码** | **按键** | **键码** |
| F1 | 112 | F7 | 118 |
| F2 | 113 | F8 | 119 |
| F3 | 114 | F9 | 120 |
| F4 | 115 | F10 | 121 |
| F5 | 116 | F11 | 122 |
| F6 | 117 | F12 | 123 |

**控制键键码值(keyCode)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **按键** | **键码** | **按键** | **键码** | **按键** | **键码** | **按键** | **键码** |
| BackSpace | 8 | Esc | 27 | Right Arrow | 39 | -\_ | 189 |
| Tab | 9 | Spacebar | 32 | Dw Arrow | 40 | .> | 190 |
| Clear | 12 | Page Up | 33 | Insert | 45 | /? | 191 |
| Enter | 13 | Page Down | 34 | Delete | 46 | `~ | 192 |
| Shift | 16 | End | 35 | Num Lock | 144 | [{ | 219 |
| Control | 17 | Home | 36 | ;: | 186 | \| | 220 |
| Alt | 18 | Left Arrow | 37 | =+ | 187 | ]} | 221 |
| Cape Lock | 20 | Up Arrow | 38 | ,< | 188 | '" | 222 |

**多媒体键码值(keyCode)**

|  |  |
| --- | --- |
| **按键** | **键码** |
| 音量加 | 175 |
| 音量减 | 174 |
| 停止 | 179 |
| 静音 | 173 |
| 浏览器 | 172 |
| 邮件 | 180 |
| 搜索 | 170 |
| 收藏 | 171 |

前后端分离

2020年2月10日

14:04

用前端直接访问api

<https://ant.design/index-cn>

扩展知识点

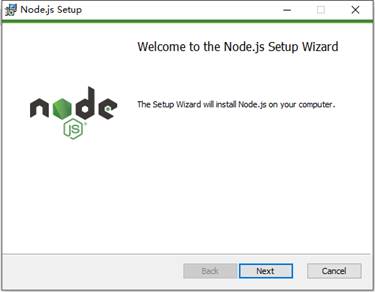
2022年2月12日

10:33

* 1. vue-cli（Vue脚手架）
     1. 安装环境：
        1. 下载并安装node.js：
           1. 下载地址： <https://nodejs.org/zh-cn/download/>



* 1. 安装node.js



* 1. 查看版本



注意：npm包管理器，是集成在node中的，所以安装了node也就有了npm,直接输入 npm -v 命令，显示npm的版本信息。

* 1. 安装cnpm（可选）

由于有些npm资源被屏蔽或者是国外资源的原因，经常会导致npm安装依赖包的时候失败，所以我们也可以使用npm的国内镜像——cnpm（淘宝npm）

npm install -g cnpm --registry=https://registry.npm.taobao.org

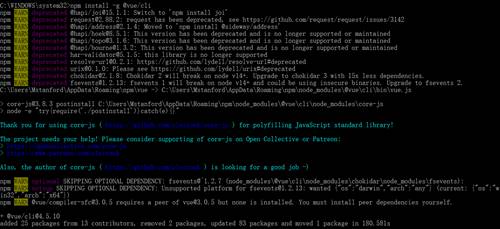


* 1. 安装vue-cli（脚手架）
     1. Vue-cli 2.0和之后版本命令不同，若之前安装过2.0可以先卸载

npm uninstall -g vue-cli

* 1. 安装vue-cli

npm install –g @vue/cli



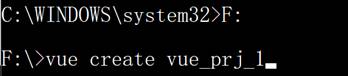
* 1. 查看版本



* 1. 使用vue create命令创建项目

vue create

* 1. 切换至指定目录
  2. 创建项目



* 1. 配置项目（手动配置）



上下进行选择，回车进行确认

* 1. 选择需要的配置项目



配置项说明：

( ) Babel//转码器，可以将ES6代码转为ES5代码，从而在现有环境执行。

( ) TypeScript// TypeScript是一个JavaScript（后缀.js）的超集（后缀.ts）包含并扩展了 JavaScript 的语法，需要被编译输出为 JavaScript在浏览器运行

( ) Progressive Web App (PWA) Support// 渐进式Web应用程序

( ) Router // vue-router（vue路由）

( ) Vuex // vuex（vue的状态管理模式）

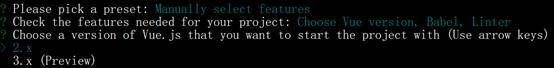
( ) CSS Pre-processors // CSS 预处理器（如：less、sass）

( ) Linter / Formatter // 代码风格检查和格式化（如：ESlint）

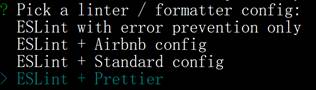
( ) Unit Testing // 单元测试（unit tests）

( ) E2E Testing // e2e（end to end）测试

* 1. 选择Vue版本



* 1. 选择ESLint代码校验规则



* 1. 选择进行代码校验时机



* 1. 选择如何存放配置



In dedicated config files // 独立文件放置

In package.json // 放package.json里

* 1. 是否存储当前配置



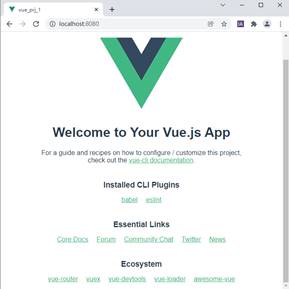
* 1. 创建项目



* 1. 启动项目
     1. 切换目录
     2. 运行服务



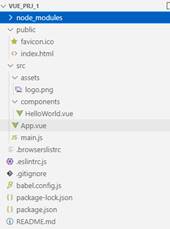
* 1. 访问项目



* 1. 终止Vue项目：ctrl+c或关闭命令行窗口



* 1. 使用vscode打开项目：
     1. 项目结构如下：



* 1. 项目结构说明：
     1. node\_modules：用于存放项目中各种依赖包
     2. public：用于存放静态资源
        1. index.html：生成项目的入口文件
     3. src：Vue的源代码文件
        1. assets：用于存放着各种静态文件，比如图片
        2. components：应用程序的组件
        3. App.vue：应用程序根组件
        4. main.js：入口文件，主要作用是初始化 vue 实例
     4. 其他：配置文件
        1. eslintrc.js: eslint代码检查的相关配置放到这里。
        2. .gitignore：配置git上传想要忽略的文件格式。
        3. babel.config.js：是一个工具链，主要用于在当前和较旧的浏览器或环境中将ES6的代码转换向后兼容（低版本ES）。
        4. package.json：模块基本信息项目开发所需要的模块，版本，项目名称
        5. package-lock.json：是在npm install时候生成的一份文件，用于记录当前状态下实际安装的各个npm package的具体来源和版本号
     5. public文件目录和src/assets目录区别
        1. public一般不用动，在vue-cli在进行build的时候，public下面的文件会原封不动的添加到dist中，不会被合并、压缩；不会被webpack打包工具所处理。
        2. src/assets目录，build之后，assets目录中的文件，会被合并到一个文件中，然后进行压缩。多用来存放业务级的js、css等。
  2. 在VS Code中运行项目
     1. 打开终端：“终端”菜单——》“新终端”



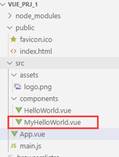
* 1. 输入命令即可



* 1. 若不记得命令，可以在README.md中查看



* 1. 编辑第一个Hello World项目
     1. 新建MyHelloWorld.vue文件



* 1. 编辑MyHelloWorld组件



* 1. 修改App.vue



* 1. 运行项目



|  |  |
| --- | --- |
| 执行流程： | 客户端请求——》index.html->main.js->app.vue->helloworld.vue->回到main.js绑定元素到index.html的app 的div中 |

二、前端路由

单页应用程序SPA single page application

路由：将路径和组件映射起来

1Vue-router 初始化是选择插件Router

2 在当前项目添加vue add router

如果无法执行则以管理员身份打开powershell,运行set-ExecutionPolicy RemoteSigned

三、组件

vue add vuex

大家在添加vuex 和 vue-router 的时候要是出错

方法一：重新添加一次

方法二：换个命令

vue-router：npm install --save vue-router

vuex：npm install --save vuex