[vue笔记]

[本人自行整理]

[2020-12-17]

目录

一．组件使用： 3

二．组件传值： 3

1)父组件给子组件传值/方法： 3

2)父组件主动获取子组件的数据和方法 4

3)子获取父的数据 4

4)非父子组件传值（有了xv更方便） 5

三．Vue中的路由 以及默认路由跳转 5

路由的配置 5

四．请求数据方法/模板 6

1） Vue-resource（vue官方提供插件） 6

2） Axios（第三方） 7

3） Fetch-json 7

五．vue的框架 7

1.基于Vue的Ui框架 7

2.基于Vue的Ui框架 7

element UI组件的单独使用（第一种方法）： 8

element UI组件的单独使用（第二种方法）： 9

六．Vuex 9

vuex的使用： 10

七.vscode使用： 12

1） vue模板自定义：文件--首选项—用户片段—创建一个vue.json内容如下 12

常用：

1.数组添加：this.list.push({\*\*\*})

2.删除某个数组值：this.list..splice(key,1)

3.打印回车：

事件@keydowm="doadd($event)"

doadd(e){

if(e.keycode==13){}

# 一．组件使用：

1） 引入组件：import Home from './components/Home.vue''

2） 注册组件：components:{

'v-home':Home

}

3） 使用组件：<v-home></v-home>

# 二．组件传值：

## 1)父组件给子组件传值/方法：

第一步：父引用子的时候绑定值

<v-home :title='title'></v-home>

export default {

  data () {

    return {

      title:'父传子'

    }

  },

  components:{

    'v-head':Head

  }

}

在子组件取出值：在data平级使用props接手值

  props:['title'],

## 2)父组件主动获取子组件的数据和方法

绑定子组件的时候绑定一个ref="xx"，通过this.$refs.xx.子组件的数据名/方法

父页面代码：

 <v-body ref="bodydata"></v-body>

  <button @click='up()'>父组件主动获取子主键数据、方法</button>

 methods: {

    run(){

      up(){

      alert(this.$refs.bodydata.sj)

    }}

子页面代码：

export default {

name:'',

  data () {

    return {

        sj:'ref获取子的数据和方法'

    };

  },

 }

## 3)子获取父的数据

通过this.$parent.父的数据/f方法名

子页面代码：

 <button @click="titledata()">获取父数据</button>

export default {

  methods: {

      titledata(){

          alert(this.$parent.titletiem)

      }

  }

}

父页面代码：

export default {

  data () {

    return {

      titletiem:'fu data'

    }

  }}}

## 4)非父子组件传值（有了xv更方便）

创建一个vuexxx.js文件->引入vue->实例化->曝光实例

代码如下：vueemit.js

import Vue from 'vue'

var vueemit=new Vue();

export default vueemit

在需要广播的地方引入实例->广播 （通过vueemit.$emit('广播名','数据')）

<button @click='emitdata()'>非父子之间通过emit传值</button>

import vueemit from '../mode/vueemit.js'

export default {

name:'',

 methods: {

   emitdata(){

     vueemit.$emit('emdata','okkkkkkkkkkkkkkkkkkkk!')

   }

 }

在需要接受的地方引入实例->接受数据(使用vueemit.$on('广播名',function(){接收数据}))

import vueemit from '../mode/vueemit.js'

 mounted () {

      alert(vueemit.$on('emdata',function(data){

          alert(data)

      }))

  }

# 三．Vue中的路由 以及默认路由跳转

## 路由的配置

https://router.vuejs.org/

1).安装

npm install vue-router --save / cnpm install vue-router --save

2)、引入并 Vue.use(VueRouter) (main.js)

import VueRouter from 'vue-router'

Vue.use(VueRouter)

3)、配置路由

[1]、创建组件 引入组件

[2]、定义路由 （建议复制s）

const routes = [

{ path: '/foo', component: Foo },

{ path: '/bar', component: Bar },

{ path: '\*', redirect: '/home' } /\*默认跳转路由\*/

]

[3]、实例化VueRouter

const router = new VueRouter({

routes // （缩写）相当于 routes: routes

})

4）、挂载

new Vue({

el: '#app',

router，

render: h => h(App)

})

5） 、根组件的模板里面放上这句话 <router-view></router-view>

6）、路由跳转

<router-link to="/foo">Go to Foo</router-link>

<router-link to="/bar">Go to Bar</router-link>

四．请求数据方法/模板（安装建议带—save可以安装到package.json可以看到）

## 1） Vue-resource（vue官方提供插件）

安装：npm install vue-resource —save

安装完毕后，在main.js中导入，如下所示：

Import VueResource from 'vue-resource'

使用插件

Vue.use(VueResource)

请求代码：

this.$http.get(地址').then(function(res){document.write(res.body); },function(){console.log('请求失败处理'); });

## 2） Axios（第三方）

安装：npm install axios —save

安装完毕后，哪里用哪里安装，如下所示：

Import Axios from 'axios'

请求代码：

axios.get('https://www.runoob.com/try/ajax/json\_demo.json').then(response => (this.info = response)).catch(function (error) { // 请求失败处理console.log(error); });

## 3） Fetch-json

和axios用法一样

# 五．vue的框架

## 1.基于Vue的Ui框架

饿了么公司基于vue开的的vue的Ui组件库

Element Ui 基于vue pc端的UI框架

MintUi 基于vue 移动端的ui框架

http://element.eleme.io/

http://mint-ui.github.io/#!/en

mintUI的使用：

1）.找官网

2）.安装 npm install mint-ui -S -S表示 --save

3）.引入mint Ui的css 和 插件

import Mint from 'mint-ui';

Vue.use(Mint);

import 'mint-ui/lib/style.css'

4）.看文档直接使用

在mintUi组件上面执行事件的写法

@click.native

<mt-button @click.native="sheetVisible = true" size="large">点击上拉 action sheet</mt-button>

## 2.基于Vue的Ui框架

饿了么公司基于vue开的的vue的Ui组件库

Element Ui 基于vue pc端的UI框架

MintUi 基于vue 移动端的ui框架

http://element.eleme.io/

element UI的使用：

1）找官网 http://element.eleme.io/#/zh-CN/component/quickstart

2）.安装 cnpm i element-ui -S -S表示 --save

3）.引入element UI的css 和 插件

import ElementUI from 'element-ui';

import 'element-ui/lib/theme-chalk/index.css';

Vue.use(ElementUI);

4）、\*webpack.config.js 配置file\_loader http://element.eleme.io/1.4/#/zh-CN/component/quickstart

{

test: /\.(eot|svg|ttf|woff|woff2)(\?\S\*)?$/,

loader: 'file-loader'

}

5）.看文档直接使用。

## element UI组件的单独使用（第一种方法）：

1））、cnpm install babel-plugin-component -D

2））、找到.babelrc 配置文件

把

{

"presets": [

["env", { "modules": false }],

"stage-3"

]

}

改为 注意：

{

"presets": [["env", { "modules": false }]],

"plugins": [

[

"component",

{

"libraryName": "element-ui",

"styleLibraryName": "theme-chalk"

}

]

]

}

3））、

import { Button, Select } from 'element-ui';

Vue.use(Button)

Vue.use(Select)

## element UI组件的单独使用（第二种方法）：

import { Button, Select } from 'element-ui';

Vue.use(Button)

Vue.use(Select)

引入对应的css

import 'element-ui/lib/theme-chalk/index.css';

如果报错： webpack.config.js 配置file\_loader

{

test: /\.(eot|svg|ttf|woff|woff2)(\?\S\*)?$/,

loader: 'file-loader'

}

# 六．Vuex

Vuex 是一个专为 Vue.js 设计的状态管理模式

vuex解决了组件之间同一状态的共享问题。当我们的应用遇到多个组件共享状态时，会需要：

多个组件依赖于同一状态。传参的方法对于多层嵌套的组件将会非常繁琐，并且对于兄弟组件间的状态传递无能为力。这需要你去学习下，vue编码中多个组件之间的通讯的做法。

来自不同组件的行为需要变更同一状态。我们经常会采用父子组件直接引用或者通过事件来变更和同步状态的多份拷贝。

以上的这些模式非常脆弱，通常会导致无法维护的代码。来自官网的一句话：Vuex 是一个专为 Vue.js 应用程序开发的状态管理模式。

它采用集中式存储管理应用的所有组件的状态。这里的关键在于集中式存储管理。这意味着本来需要共享状态的更新是需要组件之间通讯的，而现在有了vuex，就组件就都和store通讯了。问题就自然解决了。

这就是为什么官网再次会提到Vuex构建大型应用的价值。如果您不打算开发大型单页应用，使用 Vuex 可能是繁琐冗余的。确实是如此——如果您的应用够简单，您最好不要使用 Vuex。

Vuex 是一个专为 Vue.js 应用程序开发的状态管理模式

1.vuex解决了组件之间同一状态的共享问题 （解决了不同组件之间的数据共享）

2.组件里面数据的持久化。

小项目不部建议用vuex

## vuex的使用：

1、src目录下面新建一个vuex的文件夹

2、vuex 文件夹里面新建一个store.js

3、安装vuex

cnpm install vuex --save

4、在刚才创建的store.js引入vue 引入vuex 并且use vuex

import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex)

5、定义数据

/\*1.state在vuex中用于存储数据\*/

var state={

count:1

}

6、定义方法 mutations里面放的是方法，方法主要用于改变state里面的数据

var mutations={

incCount(){

++state.count;

}

}

暴露

const store = new Vuex.Store({

state,

mutations

})

export default store;

**组建里面使用vuex：**

1.引入 store

import store from '../vuex/store.js';

2、注册

export default{

data(){

return {

msg:'我是一个home组件',

value1: null,

}

},

store,

methods:{

incCount(){

this.$store.commit('incCount'); /\*触发 state里面的数据\*/

}

}

}

3、获取state里面的数据

this.$store.state.数据

4、触发 mutations 改变 state里面的数据

this.$store.commit('incCount');

# 七.vscode使用：

## 1） vue模板自定义：文件--首选项—用户片段—创建一个vue.json内容如下

{

        "Print to console": {

          "prefix": "vue",

          "body": [

            "<template>",

            "  <div></div>",

            "</template>",

            "",

            "<script>",

            "export default {",

            "name:'',",

            "  data () {",

            "    return {",

            "    };",

            "  },",

            "",

            "<!-- 注册组件 -->",

            "  components: {},",

            "",

            "<!--  计算属性 -->",

            "  computed: {},",

            "",

            "<!--mounted:html加载完成后执行。执行顺序:子组件-父组件-->例如：页面初始化",

            "  mounted: {},",

            "",

            "<!--实例创建完成：axios入口 -->",

            "  created() {},",

            "",

            "<!--方法 -->",

            "  methods: {}",

            "}",

            "",

            "</script>",

            "<style lang='less' scoped>",

            "</style>"

        ],

          "description": "Log output to console"

        }

}

2） 插件安装：vue-beautify，vue-color，VueHelper，vetur（这个很有必要，安装之后代码不会跟文本一样一个颜色）

vscode提高效率有效使用插件：

vscode-Document This-----jsdoc注释生成。

vscode-eCSStractor-----抽取页面的class，生成一个css文档。

vscode-CSS Peek------class名字定义跳转。（终于拥有dw cc的这个css定义跳转功能）

vscode-Live Server-----http服务器（相当于使用nodejs的http-server ）。

VueHelper  Vue2代码段（包括Vue2 api、vue-router2、vuex2）

Quokka.js  可以即时显示js和ts的运行结果，想试验一些短代码的运行结果，但是又不想之后删除文件的时候可以用到

codeRunner 可以运行C, C++, Java, JavaScript, PHP, Python, Perl, Perl 6, Ruby, Go, Lua, Groovy, PowerShell, BAT/CMD, BASH/SH, F# Script, F# (.NET Core), C# Script,等开发语言并支持局部执行

IntelliSense for CSS class names  智能提示 css 的 class 名

Node.js Modules Intellisense  可以在导入语句中自动完成JavaScript / TypeScript模块。

vscode-faker   生成假数据，地址，电话，图片等等

Regex Previewer   测试正则的插件