

W

WEICHUANG

謝成



第一节

vue-cli 安装

- <https://cn.vuejs.org/v2/guide/installation.html>
- <https://github.com/vuejs/vue-cli>

安装

- 安装**vue-cli**脚手架构建工具，**npm install vue-cli -g**， 安装完成之后输入 **vue -V**（注意这里是大写**npm install vue-cli**的“V”） 如果出现相应的版本号，则说明安装成功
- 在硬盘上找一个文件夹放工程用的,**cd** 目录路径
- 安装**vue**脚手架输入：**vue init webpack projectname**， 注意这里的“**projectname**”是项目的名称可以说是随便的起名， 但是需要主要的是“不能用中文”
- 安装项目依赖：**npm install**（时间比较长）
- **cd** 项目目录 **cd projectname**
- 启动项目， 输入：**npm run dev**

目录结构

```
-- build // 项目构建(webpack)相关代码
|   |-- build.js // 生产环境构建代码
|   |-- check-version.js // 检查node、npm等版本
|   |-- dev-client.js // 热重载相关
|   |-- dev-server.js // 构建本地服务器
|   |-- utils.js // 构建工具相关
|   |-- webpack.base.conf.js // webpack基础配置
|   |-- webpack.dev.conf.js // webpack开发环境配置
|   |-- webpack.prod.conf.js // webpack生产环境配置
-- config // 项目开发环境配置
|   |-- dev.env.js // 开发环境变量
|   |-- index.js // 项目一些配置变量
|   |-- prod.env.js // 生产环境变量
|   |-- test.env.js // 测试环境变量
-- src // 源码目录
|   |-- components // vue公共组件
|   |-- store // vuex的状态管理
|   |-- App.vue // 页面入口文件
|   |-- main.js // 程序入口文件，加载各种公共组件
-- static // 静态文件，比如一些图片，json数据等
|   |-- data // 群聊分析得到的数据用于数据可视化
-- .babelrc // ES6语法编译配置
-- .editorconfig // 定义代码格式
-- .gitignore // git上传需要忽略的文件格式
-- README.md // 项目说明
-- favicon.ico
-- index.html // 入口页面
-- package.json // 项目基本信息
```

第二节

vue-router

vue-router是**Vue.js**官方的路由插件，它和**vue.js**是深度集成的，适合用于构建单页面应用。**vue**的单页面应用是基于路由和组件的，路由用于设定访问路径，并将路径和组件映射起来。传统的页面应用，是用一些超链接来实现页面切换和跳转的。在**vue-router**单页面应用中，则是路径之间的切换，也就是组件的切换。

网站：<https://router.vuejs.org/zh-cn/>

安装：`npm install vue-router`

注意：在**vue-cli**中已经包含了**vue-router**，不需要再安装

```
"dependencies": {  
    "vue": "^2.5.2",  
    "vue-router": "^3.0.1"  
},
```

router-link

新建Test.vue组件，并在App.vue中配置



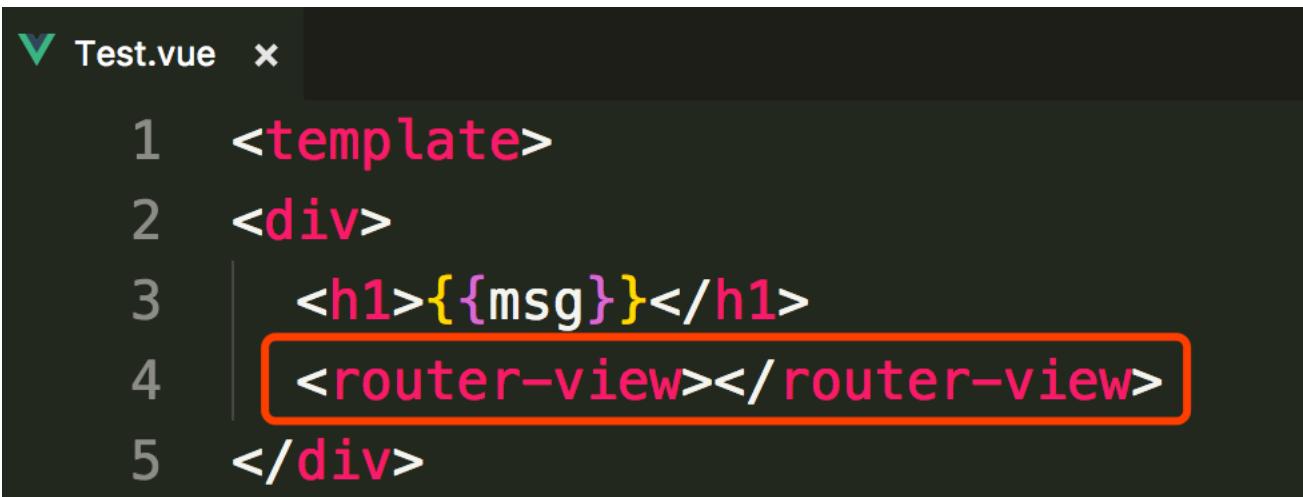
```
V App.vue x
1 <template>
2   <div id="app">
3     
4     <div>
5       <router-link to="/">首页</router-link>
6       <router-link to="/test">测试</router-link>
7     </div>
8     <router-view/>
9   </div>
10 </template>
```

子路由 (1)

① App.vue

```
<router-link to="/">首页</router-link>
<router-link to="/test">测试</router-link>
<router-link to="/test/test1">测试子路由1</router-link>
<router-link to="/test/test2">测试子路由2</router-link>
```

② Test.vue



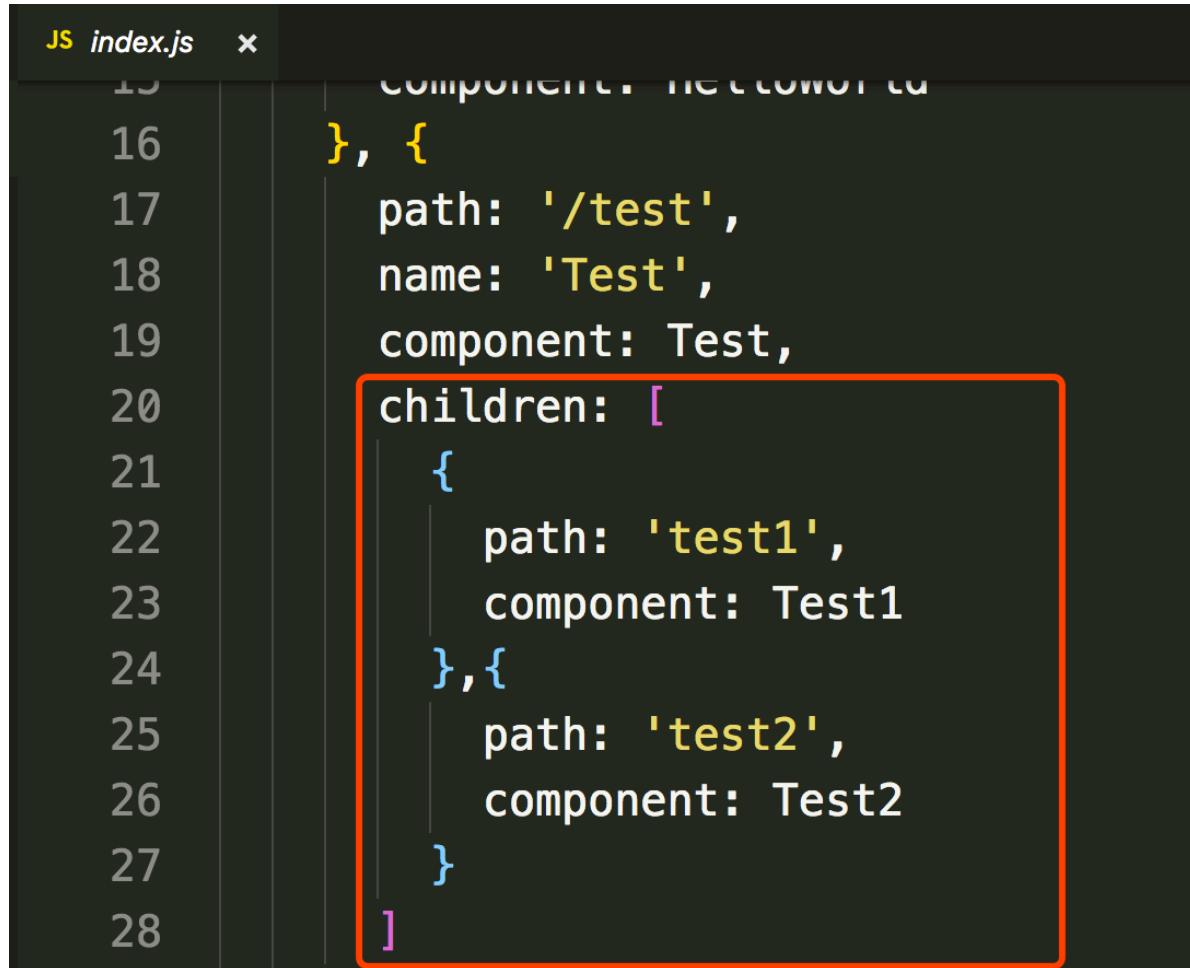
The screenshot shows a code editor window with a dark theme. The title bar says "Test.vue". The code in the editor is:

```
1 <template>
2 <div>
3   <h1>{{msg}}</h1>
4   <router-view></router-view>
5 </div>
```

The line containing the `<router-view></router-view>` tag is highlighted with a red rectangular box.

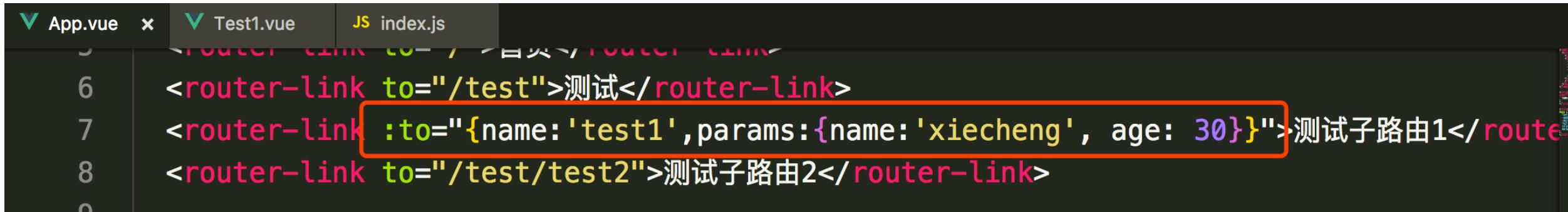
子路由（2）

- ③ 新建Test1.vue 和 Test2.vue
- ④ index.js中配置子路由



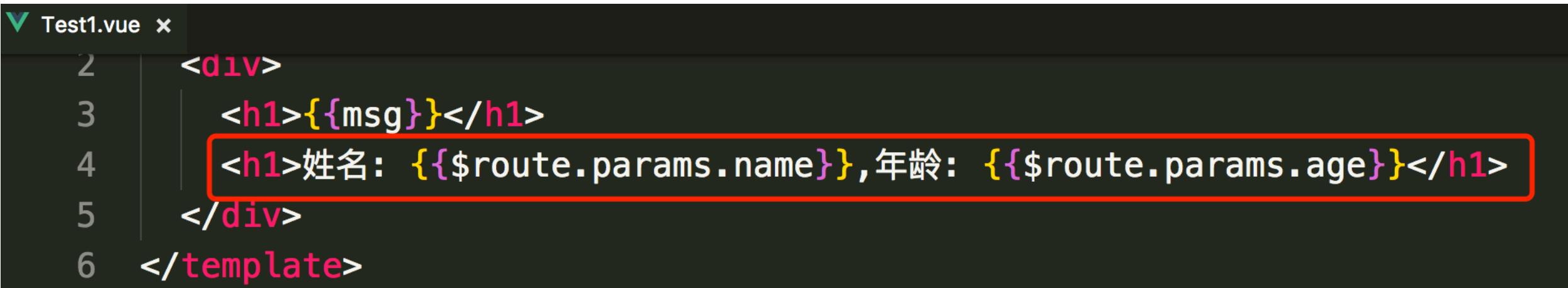
```
JS index.js ×
  ...
15     component: Network
16   }, {
17     path: '/test',
18     name: 'Test',
19     component: Test,
20     children: [
21       {
22         path: 'test1',
23         component: Test1
24       },
25       {
26         path: 'test2',
27         component: Test2
28       }
29     ]
30   ]
31 }
```

参数传递



A screenshot of a code editor showing the `App.vue` component. It contains three `router-link` components. The second one, which has a red box around its `:to` attribute, is used to demonstrate parameter passing. The code is as follows:

```
<router-link to="/test">测试</router-link>
<router-link :to="{name:'test1',params:{name:'xiecheng', age: 30}}">测试子路由1</router-link>
<router-link to="/test/test2">测试子路由2</router-link>
```



A screenshot of a code editor showing the `Test1.vue` template. It contains a `div` element with two `h1` elements. The second `h1` element, which has a red box around its content, uses `$route.params.name` and `$route.params.age` to display user information. The code is as follows:

```
<div>
  <h1>{{msg}}</h1>
  <h1>姓名: {{$route.params.name}}, 年龄: {{$route.params.age}}</h1>
</div>
</template>
```

利用URL传参

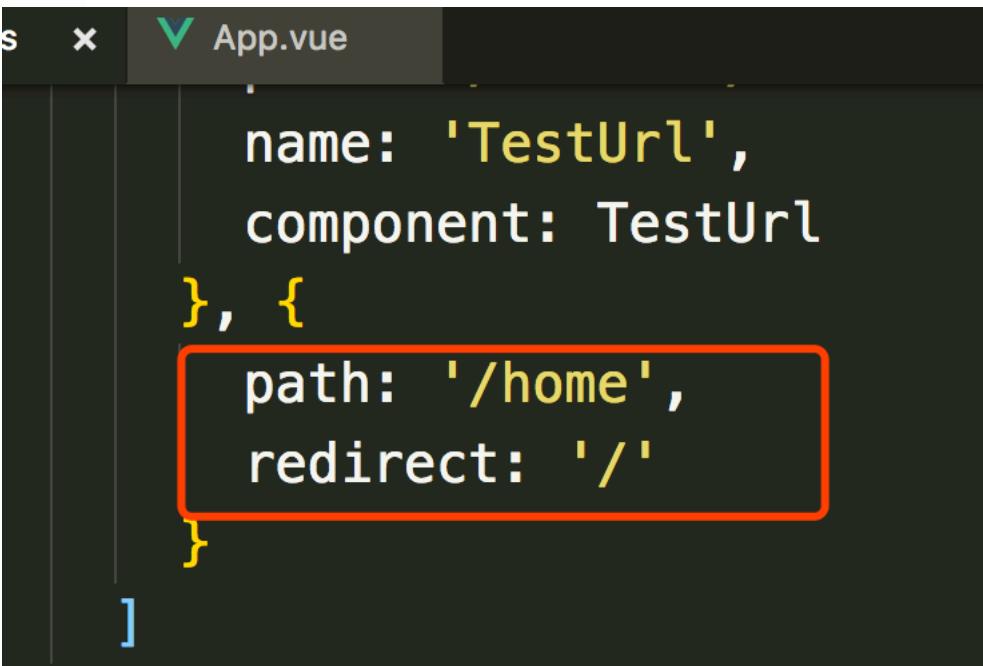
- 创建TestUrl.vue 并在index.js中引入

```
JS index.js x
35
36 }, {
37   path: '/testurl/:userId(\d+)/:userName',
38   name: 'TestUrl',
39   component: TestUrl
40 }
```

```
V App.vue x
<router-link :to="{name:'test1',params:{name:'xiecheng', age: 30} }>测试子路由1</router-link>
<router-link to="/test/test2">测试子路由2</router-link>
<router-link to="/testurl/5/zhangsan">测试URL传参</router-link>
<p>{$route.name}</p>
</div>
```

```
V TestUrl.vue x
1 <template>
2   <div>
3     <h1>用户ID: {{$route.params.userId}}</h1>
4     <h1>用户名称: {{$route.params.userName}}</h1>
5   </div>
6 </template>
```

重定向



A screenshot of a code editor showing a portion of `App.vue`. The code defines a router configuration:

```
  name: 'TestUrl',
  component: TestUrl
}, {
  path: '/home',
  redirect: '/'
]
```

The `path: '/home'` and `redirect: '/'` lines are highlighted with a red rectangle.



A screenshot of a code editor showing a portion of `App.vue`. The template contains:

```
<router-link to="/testurl/5/zhangsan">测试URL传参</router-link>
<router-link to="/home">redirect</router-link>
<p>{{ $route.name }}</p>
```

The second `<router-link>` tag is highlighted with a red rectangle.

重定向带参数

```
JS index.js x
42     redirect: '/'
43 }, {
44     path: '/redirectparam/:userId(\d+)/:userName',
45     redirect: '/testurl/:userId(\d+)/:userName'
46 }
```

```
V App.vue x
<router-link to="/home">redirect</router-link>
<router-link to="/redirectparam/6/lisi">redirectparam</router-link>
```

alias别名

```
JS index.js  x
46      }, {
47        path: '/test1',
48        component: Test1,
49        alias: '/abc'
50      }
```

```
V App.vue  x
<router-link to="/redirectparam/6/lisi">redirectpa
<router-link to="/abc">alias</router-link>
```

- **redirect:** 仔细观察URL, `redirect`是直接改变了`url`的值, 把`url`变成了真实的`path`路径。
- **alias:** URL路径没有变, 这种情况更友好, 让用户知道自己访问的路径, 只是改变了`<router-view>`中的内容。

路由过度动画

- **fade-enter**: 进入过渡的开始状态，元素被插入时生效，只应用一帧后立刻删除。
- **fade-enter-active**: 进入过渡的结束状态，元素被插入时就生效，在过渡过程完成后移除。
- **fade-leave**: 离开过渡的开始状态，元素被删除时触发，只应用一帧后立刻删除。
- **fade-leave-active**: 离开过渡的结束状态，元素被删除时生效，离开过渡完成后被删除。

```
App.vue
```

```
<transition name="fade" mode="out-in">
  <router-view/>
  <!-- <router-view name="view1"/>
  <router-view name="view2"/> -->
</transition>
```

```
App.vue
```

```
}
.fade-enter {
  opacity: 0;
}
.fade-leave {
  opacity: 1;
}
.fade-enter-active {
  transition: opacity 0.5s;
}
.fade-leave-active {
  opacity: 0;
  transition: opacity 0.5s;
}
```

mode & 404

- **history**去掉URL上的井号
- 默认值: hash

- 处理404 新建Error.vue

```
JS index.js ×  
11 export default new Router({  
12   mode: 'history',  
13   routes: [  
14     {
```

```
JS index.js ×  
51   alias: '/abc'  
52 }, {  
53   path: '*',  
54   component: Error  
55 }  
56 ]
```

路由钩子函数（1）

```
JS index.js x
38 }, {
39   path: '/testurl/:userId(\d+)/:userName',
40   name: 'TestUrl',
41   component: TestUrl,
42   beforeEnter: (to, from, next) => {
43     console.log(to);
44     console.log(from);
45     next();
46   }
47 }
```

在index.js配置文件中配置，可以配置**beforeEnter**钩子

to: 路由将要跳转的路径信息，信息是包含在对象里边的。

from: 路径跳转前的路径信息，也是一个对象的形式。

next: 路由的控制参数，**next(true)**和**next(false)**。

路由钩子函数（2）

在模板中配置：

- **beforeRouteEnter**: 在路由进入前的钩子函数。
- **beforeRouteLeave**: 在路由离开前的钩子函数。



```
9  name: 'HelloWorld',
10 data () {
11   return {
12     msg: 'Welcome to Your Vue.js App'
13   }
14 },
15 beforeRouteEnter: (to, from, next) => {
16   console.log("准备进入路由模板");
17   next();
18 },
19 beforeRouteLeave: (to, from, next) => {
20   console.log("准备离开路由模板");
21   next();
22 }
23 }
```

编程式导航

V App.vue x

```
2 <div id="app">
3   
4   <div>
5     <button @click="goBack">后退</button>
6     <button @click="goForward">前进</button>
7     <button @click="goHome">返回首页</button>
8   </div>
```

V App.vue x

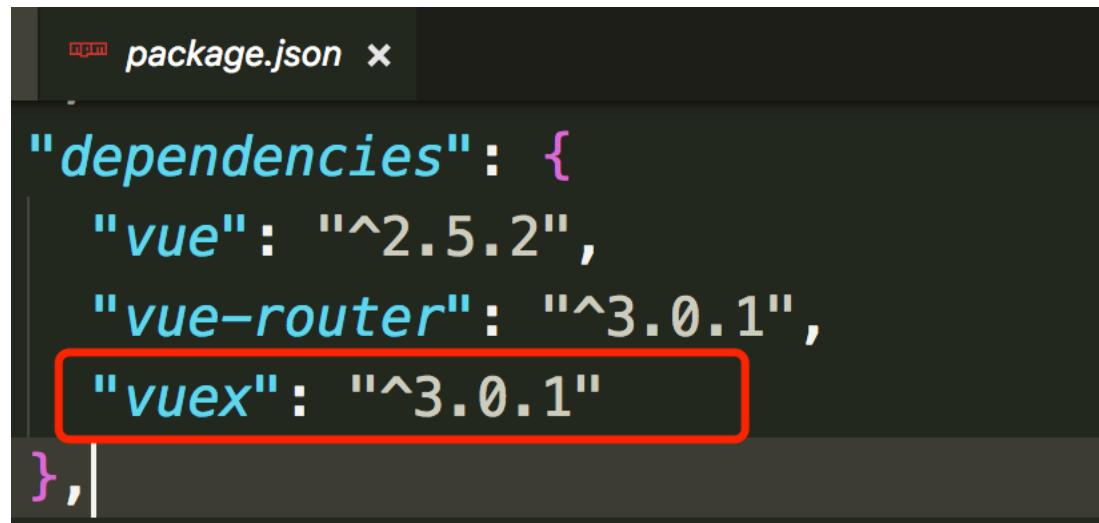
```
30 <script>
31   export default {
32     name: "app",
33     methods: {
34       goBack(){
35         this.$router.go(-1);
36       },
37       goForward(){
38         this.$router.go(1);
39       },
40       goHome(){
41         this.$router.push('/');
42       }
43     }
44   };
```

第三节

vuex

- **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，并以相应的规则保证状态以一种可预测的方式发生变化。
- <https://vuex.vuejs.org/zh-cn/intro.html>
- <https://github.com/vuejs/vuex>

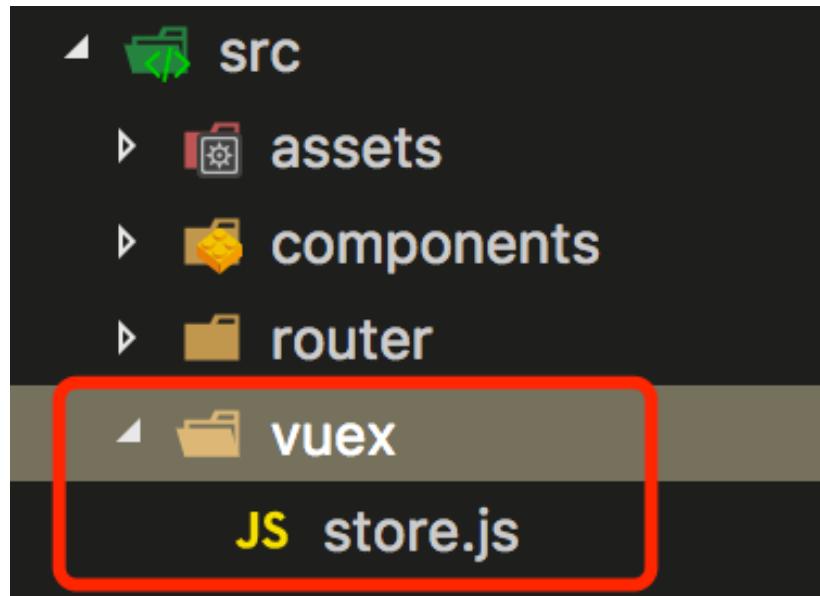
- **npm install vuex --save**



```
npm package.json x

"dependencies": {
  "vue": "^2.5.2",
  "vue-router": "^3.0.1",
  "vuex": "^3.0.1"
},|
```

计数器（1）



JS store.js

```
1 import Vue from 'vue';
2 import Vuex from 'vuex';
3 Vue.use(Vuex);
4
5 const state = {
6   count: 1
7 };
8
9 const mutations = {
10   add(state) {
11     state.count++;
12   },
13   reduce(state) {
14     state.count--;
15   }
16 };
17
18 export default new Vuex.Store({
19   state,
20   mutations
21 })
```

计数器（2）

```
Count.vue ×  
1 <template>  
2   <div>  
3     <h1>计数器: {{$store.state.count}}</h1>  
4     <button @click="$store.commit('add')">加1</button>  
5     <button @click="$store.commit('reduce')">减1</button>  
6   </div>  
7 </template>  
8 <script>  
9 import store from '@/vuex/store'  
10 export default {  
11   data() {  
12     return {};  
13   },  
14   store  
15 };  
16 </script>
```

state访问状态对象

- ① 通过**computed**的计算属性直接赋值

```
computed:{  
  count(){  
    return this.$store.state.count;  
  }  
}
```

- ② 通过**mapState**的对象来赋值

```
import {mapState} from 'vuex';
```

```
computed:mapState({  
  count:state=>state.count  
})
```

- ③ 通过**mapState**的数组来赋值

```
computed:mapState(["count"])
```

这三种做法的好处是，
可以直接使用插值的
形式了 {{count}}

mutations修改状态--传值

JS store.js

```
9 const mutations = {
10   add(state, num) {
11     state.count += num;
12   },
13   reduce(state) {
14     state.count--;
15   }
16};
```

Count.vue

```
3 <h1>计数器: {{count}}</h1>
4 <button @click="$store.commit('add', 10)">加1</button>
5 <button @click="$store.commit('reduce')">减1</button>
```

mutations修改状态--直接调用方法

```
V Count.vue x
4 <button @click="$store.commit('add', 10)">加1</button>
5 <button @click="reduce">减1</button>
```

```
V Count.vue x
...
10 import { mapState, mapMutations } from "vuex";
```

```
V Count.vue x
...
25 methods: mapMutations(["add", "reduce"])
```

getters计算过滤操作

JS store.js ×

```
18 const getters = {  
19     count: function (state) {  
20         return state.count += 100;  
21     }  
22 }
```

Count.vue ×

```
9 import store from "@/vuex/store";  
10 import { mapState, mapMutations, mapGetters } from "vuex";
```

Count.vue ×

```
24     computed: {  
25         ...mapState(["count"]),  
26         ...mapGetters(["count"])  
27     },
```

actions异步修改状态

https://www.zhihu.com/question/48759748/answer/112823337?from=profile_answer_card

JS store.js ×

```
25
24 const actions ={
25   addAction({commit}){
26     commit('add',10);
27 },
28   reduceAction({commit}){
29     commit('reduce');
30 }
31 }
```

Count.vue ×

```
6 <p>
7   <button @click="addAction">+</button>
8   <button @click="reduceAction">-</button>
9 </p>
```

Count.vue ×

```
32 |   methods: {
33 |     ...mapMutations(["add", "reduce"]),
34 |     ...mapActions(["addAction", "reduceAction"])
35 |   }
36 |};
```

module模块组

```
JS store.js ×  
36  
37 const moduleA = {  
38   state,  
39   mutations,  
40   getters,  
41   actions  
42 }  
43 export default new Vuex.Store({  
44   modules: {  
45     a: moduleA  
46   }  
47 })
```

```
▼ Count.vue ×  
  . . .  
<h1>{{store.state.a.count}}</h1>
```



Thank you

谢

谢

观

看

多路由操作

V App.vue x

```
10  </ul>
11  <router-view/>
12  <router-view name="view1"/>
13  <router-view name="view2"/>
14  </div>
15  </template>
```

JS index.js x

```
9
10 export default new Router({
11   routes: [
12     {
13       path: '/',
14       name: 'Hello',
15       components: {
16         default: HelloWorld,
17         view1: Test1,
18         view2: Test2
19       }
20     }, {
```