

## 16.Vuex 状态入门

学习要点：

- 1.Vuex 概念
- 2.store 模式

本节课我们来开始了解 **Vuex** 状态管理器的概念和 **store** 模式。

### 一. Vuex 概念

1. 首先安装脚手架的时候，勾选 **Vuex** 状态管理器：**store**(仓库)目录；
2. **Vuex** 用于解决组件与组件之间共享的状态，集中存储管理所有组件状态；
3. 支持调试工具 **devtools** 查看分析，安装时选带 **Vuex** 插件的脚手架；
4. 然后，我们在 **About.vue** 创建一个简单的计数器功能：

```
<h1>This is an about page : {{count}}</h1>
<button @click="addCount">计数</button>
```

```
export default {
  name : 'about',
  data() {
    return {
      count : 0
    }
  },
  methods : {
    addCount() {
      this.count++
    },
  }
}
```

PS：这是基础课程中的计数器，它只能作用于这个组件，跳到其它组件即失效；

### 二. store 模式

1. 手册有这么一句话：如果不是大型的单页面应用，使用 **Vuex** 会繁琐冗余；
2. 也就是说，你的应用特别简单，不需要真么重的插件，可以使用 **store** 模式；
3. **store** 模式支持你在极少的共享数据中，就好比只近视 **50** 度，可以不戴眼镜；
4. 创建一个 **store/index.js**，代码根据手册再度削减如下：

```
const store = {
  //共享属性
  state: {
    count: 0
  },

  //共享数据的累加
```

```
    increment () {  
      this.state.count++  
    },  
  };  
};
```

```
export default store
```

5. 创建一个新的 Text.vue，用于生成共享数据的组件；

```
<template>  
  <div class="count">  
    共享: {{storeState.count}}<br>  
    <button @click="increment">store 计数</button>  
  </div>  
</template>  
  
<script>  
  import store from "@store"  
  
  export default {  
    name: "Count",  
    data() {  
      return {  
        storeState : store.state  
      }  
    },  
    methods : {  
      increment() {  
        store.increment()  
      }  
    }  
  }  
</script>
```

PS: 其它页面以同样的方式构建或者只需显示视图数据即可；