

09-Pinia: vue3状态管理

一、相关介绍

1. Pinia 介绍

1.1 Pinia是什么



Pinia 是一个状态管理工具，Vue 的专属状态管理库，它允许你跨组件或页面共享状态。

1.2 为什么用Pinia

程序组件多, 数据多

1. 不同组件数据保持同步

2. 数据的修改都是可追踪

举个例子:

一个服装店有两名店员，小比和小特

一天早上，他们分别对衬衫的数量做了一次盘点，发现一共有20条，

小比卖出去12条，他以为库存里还有8条

小特卖出去8条，他以为库存里还有12条

而事实上是，库存现在已经为零

如果他们再接受客户的预订，就会出现库存不足的情况

小比和小特因为没有保持库存的数量的同步导致了尴尬，这个就是所谓的 数据保持同步

店长需要知道, 谁卖出了多少条，这个行为我们称之为 数据修改是可追踪的

1.3 Pinia中存什么

多个组件共享的数据，才存储在pinia中

某个组件中的私有数据，依旧存储在组件自身内部

例如：

- 登陆的用户名需要在**首页、个人中心页、搜索页、商品详情页** 使用, 则用户名存储在Pinia中
- 新闻详情数据, 只有在新闻详情页查看, 则存储在组件自身内部

1.4 总结

1. pinia是什么?

答: Pinia 是一个简单实用的状态管理工具, 和菠萝一样 香

2. 何时用pinia?

答: 当多个组件共享数据的时候

3. pinia存什么数据?

答: 多个组件共享的数据

二、Pinia的使用

1. Pinia学习例子准备

1.1 目标

创建项目, 为学习Pinia做准备

- 需求1: App.vue(作为根组件)
- 需求2: 子组件Add和子组件Sub, 作为在App.vue的子组件
- 需求3: 三个组件共享库存数据(保持同步)

1.2 效果

根组件
库存总数:

Add组件
已知库存数: 0

Sub组件
已知库存数: 0

1.3 代码示例

components/Add.vue

```
1 <script setup></script>
```

```
2
3 <template>
4   <div>
5     <h3>Add组件</h3>
6     <p>已知库存数: 0</p>
7     <button>库存+1</button>
8   </div>
9 </template>
```

components/Sub.vue

```
1 <script setup></script>
2
3 <template>
4   <div>
5     <h3>Sub组件</h3>
6     <p>已知库存数: 0</p>
7     <button>库存-1</button>
8   </div>
9 </template>
```

App.vue

```
1 <script setup>
2   import Add from '@components/Add.vue'
3   import Sub from '@components/Sub.vue'
4 </script>
5 <template>
6   <div>
7     <h1>根组件</h1>
8     <p>
9       <span>库存总数: </span>
10      <input type="number" />
11    </p>
12  </div>
13  <hr />
14  <Add />
15  <hr />
16  <Sub />
17 </template>
18
19 <style>
20   #app {
```

```
21     width: 350px;
22     margin: 60px auto;
23     border: 1px solid #ccc;
24     padding: 4px;
25   }
26 </style>
```

1.4 总结

1. 准备工作做了什么？

答：App下使用了Add和Sub, 之后要在3个组件中用 Pinia共享数据

2. Pinia - 准备store并渲染数据

2.1 目标

使用Pinia使用，管理共享的库存数据

2.2 步骤

2.2.1 下载 pinia

```
1 yarn add pinia
2
3 # or
4
5 npm i pinia
```

2.2.2 创建 pinia 实例 并注册

```
1 // main.js
2
3 // 导入
4 import { createPinia } from 'pinia'
5
6 // 创建 pinia 实例
7 const pinia = createPinia()
8
9 // 注册实例
10 app.use(pinia)
```

2.2.3 定义仓库

创建 `src/store/stock.js` 模块, 用来管理共享的库存数据

```
1 import { defineStore } from 'pinia'
2
3 // 你可以任意命名 `defineStore()` 的返回值, 但最好使用 store 的名字, 同时以 `use` 开头
  且以 `Store` 结尾。(比如 `useUserStore`, `useCartStore`, `useProductStore`)
4
5 // 第一个参数是你的应用中 Store 的唯一 ID。
6 export const useXxxStore = defineStore('仓库名称', Setup函数或Option对象)
7
8 // 这个名字, 也被用作 id, 是必须传入的, Pinia 将用它来连接 store 和 devtools。
9 // 为了养成习惯性的用法, 将返回的函数命名为 use... 是一个符合组合式函数风格的约定。
10
11 // 第二个参数可接受两类值: Setup 函数或 Option 对象。
```

2.2.3.1 组合式API(Setup函数)

```
1 import { defineStore } from 'pinia'
2 import { ref, computed } from 'vue'
3
4 // 定义一个名为 stock的store 并导出
5 export const useStockStore = defineStore('stock', () => {
6   // 初始化库存数据
7   const stock = ref(20)
8
9   // 计算库存的两倍
10  const doubleStock = computed(() => {
11    return stock.value * 2
12  })
13
14  // 增加
15  const addStock = () => {
16    stock.value++
17  }
18  // 减少
19  const subStock = () => {
20    stock.value--
21  }
22
23  // 返回共享数据和操作函数
24  return {
25    stock,
26    doubleStock,
27    addStock,
```

```
28     subStock
29   }
30 })
```

2.2.3.2 选项式API(Option对象)

```
1 import { defineStore } from 'pinia'
2
3 // 定义一个名为 stock的store 并导出
4 export const useStockStore = defineStore('stock', {
5   // 共享数据
6   state: () => ({
7     // 库存初始值
8     stock: 20
9   }),
10  // 基于共享数据的计算属性
11  getters: {
12    // 计算库存的两倍
13    doubleStock: (state) => state.stock * 2
14  },
15  // 修改共享数据的方法
16  actions: {
17    // 增加库存
18    addStock() {
19      this.stock++
20    },
21    // 减少库存
22    subStock() {
23      this.stock--
24    }
25  }
26 })
```

2.2.4 使用仓库

Add.vue

```
1 <script setup>
2   // 导入 使用库存仓库
3   import { useStockStore } from '@/store/stock.js'
4
5   // 创建库存仓库
6   const stockStore = useStockStore()
7 </script>
```

```

8
9 <template>
10   <div>
11     <h3>Add组件</h3>
12     <p>已知库存数: {{ stockStore.stock }}</p>
13     <button @click="stockStore.addStock()">库存+1</button>
14   </div>
15 </template>

```

Sub.vue

```

1 <script setup>
2   // 导入 使用库存仓库
3   import { useStockStore } from '@/store/stock.js'
4
5   // 创建库存仓库
6   const stockStore = useStockStore()
7 </script>
8
9 <template>
10   <div>
11     <h3>Sub组件</h3>
12     <p>已知库存数: {{ stockStore.stock }}</p>
13     <button @click="stockStore.subStock()">库存-1</button>
14   </div>
15 </template>

```

App.vue

```

1 <script setup>
2   import Add from './components/Add.vue'
3   import Sub from './components/Sub.vue'
4
5   import { useStockStore } from '@/store/stock'
6
7   const stockStore = useStockStore()
8 </script>
9
10 <template>
11   <h1>根组件</h1>
12   库存总数:
13   <input
14     type="number"

```

```

15     v-model.number="stockStore.stock" />
16     <p>stock 的翻倍值 = {{ stockStore.doubleStock }}</p>
17     <hr />
18     <Add />
19     <hr />
20     <Sub />
21 </template>
22
23 <style>
24   #app {
25     width: 350px;
26     margin: 100px auto;
27     padding: 5px 10px;
28     border: 1px solid #ccc;
29   }
30 </style>

```

2.3 总结

1. 如何定义 store ?

```

1 // 导入
2 import { defineStore } from 'pinia'
3 // 定义 -> 配置 -> 导出
4 export const useXxxStore = defineStore('仓库名称', Setup函数或Option对象)

```

2. 如何使用 store ?

```

1 // 导入
2 import { useXxxStore } from './xxx'
3
4 // 创建
5 const xxxStore = useXxxStore()
6
7 // 使用
8 xxxStore.数据名
9 xxxStore.计算属性名
10 xxxStore.方法名(实参列表)

```

3. Setup函数或Option对象 用法选用哪种?

答：二者没有本质区别, 只是写法不同, 自己爱用哪种用哪种

三、Pinia重构代办任务案例

1. 效果



2. 思路

1. Pinia管理共享数据
2. 组件使用共享数据

3. 静态代码

components/ToDoHeader.vue

```
1 <script setup></script>
2
3 <template>
4   <header class="header">
5     <h1>todos</h1>
6     <input class="new-todo" placeholder="What needs to be finished?" autofocus
7   </header>
8 </template>
```

components/ToDoMain.vue

```
1 <script setup></script>
2
3 <template>
4   <section class="main">
5     <input id="toggle-all" class="toggle-all" type="checkbox" />
6     <label for="toggle-all">Mark all as complete</label>
7     <ul class="todo-list">
8       <li>
9         <div class="view">
```

```

10     <input class="toggle" type="checkbox" />
11     <label>Buy eggs</label>
12     <button class="destroy"></button>
13   </div>
14 </li>
15 <li>
16   <div class="view">
17     <input class="toggle" type="checkbox" checked />
18     <label>打王者</label>
19     <button class="destroy"></button>
20   </div>
21 </li>
22 </ul>
23 </section>
24 </template>

```

components/TodoFooter.vue

```

1 <script setup></script>
2
3 <template>
4   <footer class="footer">
5     <span class="todo-count"><strong>0</strong> item left</span>
6     <ul class="filters">
7       <li>
8         <a href="#" class="selected">All</a>
9       </li>
10      <li>
11        <a href="#/active">Active</a>
12      </li>
13      <li>
14        <a href="#/completed">Completed</a>
15      </li>
16    </ul>
17    <button class="clear-completed">
18      Clear completed
19    </button>
20  </footer>
21 </template>

```

App.vue

```

1 <script setup>

```

```

2   import './assets/style.css'
3   import TodoHeader from './components/TodoHeader.vue'
4   import TodoMain from './components/TodoMain.vue'
5   import TodoFooter from './components/TodoFooter.vue'
6 </script>
7
8 <template>
9   <section class="todoapp">
10     <todo-header />
11     <todo-main />
12     <todo-footer />
13   </section>
14 </template>

```

assets/style.css

```

1 @charset 'utf-8';
2
3 html,
4 body {
5   margin: 0;
6   padding: 0;
7 }
8
9 button {
10   margin: 0;
11   padding: 0;
12   border: 0;
13   background: none;
14   font-size: 100%;
15   vertical-align: baseline;
16   font-family: inherit;
17   font-weight: inherit;
18   color: inherit;
19   -webkit-appearance: none;
20   appearance: none;
21   -webkit-font-smoothing: antialiased;
22   -moz-osx-font-smoothing: grayscale;
23 }
24
25 body {
26   font:
27     14px 'Helvetica Neue',
28     Helvetica,
29     Arial,

```

```
30     sans-serif;
31     line-height: 1.4em;
32     background: #f5f5f5;
33     color: #111111;
34     min-width: 230px;
35     max-width: 550px;
36     margin: 0 auto;
37     -webkit-font-smoothing: antialiased;
38     -moz-osx-font-smoothing: grayscale;
39     font-weight: 300;
40 }
41
42 .hidden {
43     display: none;
44 }
45
46 .todoapp {
47     background: #fff;
48     margin: 130px 0 40px 0;
49     position: relative;
50     box-shadow:
51         0 2px 4px 0 rgba(0, 0, 0, 0.2),
52         0 25px 50px 0 rgba(0, 0, 0, 0.1);
53 }
54
55 .todoapp input::-webkit-input-placeholder {
56     font-style: italic;
57     font-weight: 400;
58     color: rgba(0, 0, 0, 0.4);
59 }
60
61 .todoapp input::-moz-placeholder {
62     font-style: italic;
63     font-weight: 400;
64     color: rgba(0, 0, 0, 0.4);
65 }
66
67 .todoapp input::input-placeholder {
68     font-style: italic;
69     font-weight: 400;
70     color: rgba(0, 0, 0, 0.4);
71 }
72
73 .todoapp h1 {
74     position: absolute;
75     top: -100px;
76     width: 100%;
```

```
77 font-size: 50px;
78 font-weight: 200;
79 text-align: center;
80 color: #b83f45;
81 -webkit-text-rendering: optimizeLegibility;
82 -moz-text-rendering: optimizeLegibility;
83 text-rendering: optimizeLegibility;
84 }
85
86 .new-todo,
87 .edit {
88   position: relative;
89   margin: 0;
90   width: 100%;
91   font-size: 24px;
92   font-family: inherit;
93   font-weight: inherit;
94   line-height: 1.4em;
95   color: inherit;
96   padding: 6px;
97   border: 1px solid #999;
98   box-shadow: inset 0 -1px 5px 0 rgba(0, 0, 0, 0.2);
99   box-sizing: border-box;
100   -webkit-font-smoothing: antialiased;
101   -moz-osx-font-smoothing: grayscale;
102 }
103
104 .new-todo {
105   padding: 16px 16px 16px 60px;
106   height: 65px;
107   border: none;
108   background: rgba(0, 0, 0, 0.003);
109   box-shadow: inset 0 -2px 1px rgba(0, 0, 0, 0.03);
110 }
111
112 .main {
113   position: relative;
114   z-index: 2;
115   border-top: 1px solid #e6e6e6;
116 }
117
118 .toggle-all {
119   width: 1px;
120   height: 1px;
121   border: none; /* Mobile Safari */
122   opacity: 0;
123   position: absolute;
```

```
124     right: 100%;
125     bottom: 100%;
126 }
127
128 .toggle-all + label {
129     display: flex;
130     align-items: center;
131     justify-content: center;
132     width: 45px;
133     height: 65px;
134     font-size: 0;
135     position: absolute;
136     top: -65px;
137     left: -0;
138 }
139
140 .toggle-all + label:before {
141     content: ' ';
142     display: inline-block;
143     font-size: 22px;
144     color: #949494;
145     padding: 10px 27px 10px 27px;
146     -webkit-transform: rotate(90deg);
147     transform: rotate(90deg);
148 }
149
150 .toggle-all:checked + label:before {
151     color: #484848;
152 }
153
154 .todo-list {
155     margin: 0;
156     padding: 0;
157     list-style: none;
158 }
159
160 .todo-list li {
161     position: relative;
162     font-size: 24px;
163     border-bottom: 1px solid #ededed;
164 }
165
166 .todo-list li:last-child {
167     border-bottom: none;
168 }
169
170 .todo-list li.editing {
```

```

171 border-bottom: none;
172 padding: 0;
173 }
174
175 .todo-list li.editing .edit {
176 display: block;
177 width: calc(100% - 43px);
178 padding: 12px 16px;
179 margin: 0 0 0 43px;
180 }
181
182 .todo-list li.editing .view {
183 display: none;
184 }
185
186 .todo-list li .toggle {
187 text-align: center;
188 width: 40px;
189 /* auto, since non-WebKit browsers doesn't support input styling */
190 height: auto;
191 position: absolute;
192 top: 0;
193 bottom: 0;
194 margin: auto 0;
195 border: none; /* Mobile Safari */
196 -webkit-appearance: none;
197 appearance: none;
198 }
199
200 .todo-list li .toggle {
201 opacity: 0;
202 }
203
204 .todo-list li .toggle + label {
205 /*
206 Firefox requires `#` to be escaped -
207 https://bugzilla.mozilla.org/show_bug.cgi?id=922433
208 IE and Edge requires *everything* to be escaped to render, so we do that
209 instead of just the `#` - https://developer.microsoft.com/en-us/microsoft-
210 edge/platform/issues/7157459/
211 */
212 background-image:
213 url('data:image/svg+xml;utf8,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%2
214 %20width%3D%2240%22%20height%3D%2240%22%20viewBox%3D%22-10%20-
215 18%20100%20135%22%3E%3Ccircle%20cx%3D%2250%22%20cy%3D%2250%22%20r%3D%2250%22%20
216 fill%3D%22none%22%20stroke%3D%22%23949494%22%20stroke-
217 width%3D%223%22%2F%3E%3C%2Fsvg%3E');

```

```
210 background-repeat: no-repeat;
211 background-position: center left;
212 }
213
214 .todo-list li .toggle:checked + label {
215 background-image:
url('data:image/svg+xml;utf8,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000
%2Fsvg%22%20width%3D%2240%22%20height%3D%2240%22%20viewBox%3D%22-10%20-
18%20100%20135%22%3E%3Ccircle%20cx%3D%2250%22%20cy%3D%2250%22%20r%3D%2250%22%20
fill%3D%22none%22%20stroke%3D%22%2359A193%22%20stroke-
width%3D%223%22%2F%3E%3Cpath%20fill%3D%22%233EA390%22%20d%3D%22M72%2025L42%2071
%2027%2056l-4%204%2020%2020%2034-52z%22%2F%3E%3C%2Fsvg%3E')';
216 }
217
218 .todo-list li label {
219 overflow-wrap: break-word;
220 padding: 15px 15px 15px 60px;
221 display: block;
222 line-height: 1.2;
223 transition: color 0.4s;
224 font-weight: 400;
225 color: #484848;
226 }
227
228 .todo-list li.completed label {
229 color: #949494;
230 text-decoration: line-through;
231 }
232
233 .todo-list li .destroy {
234 display: none;
235 position: absolute;
236 top: 0;
237 right: 10px;
238 bottom: 0;
239 width: 40px;
240 height: 40px;
241 margin: auto 0;
242 font-size: 30px;
243 color: #949494;
244 transition: color 0.2s ease-out;
245 }
246
247 .todo-list li .destroy:hover,
248 .todo-list li .destroy:focus {
249 color: #c18585;
250 }
```



```
251
252 .todo-list li .destroy:after {
253   content: 'x';
254   display: block;
255   height: 100%;
256   line-height: 1.1;
257 }
258
259 .todo-list li:hover .destroy {
260   display: block;
261 }
262
263 .todo-list li .edit {
264   display: none;
265 }
266
267 .todo-list li.editing:last-child {
268   margin-bottom: -1px;
269 }
270
271 .footer {
272   padding: 10px 15px;
273   height: 20px;
274   text-align: center;
275   font-size: 15px;
276   border-top: 1px solid #e6e6e6;
277 }
278
279 .footer:before {
280   content: '';
281   position: absolute;
282   right: 0;
283   bottom: 0;
284   left: 0;
285   height: 50px;
286   overflow: hidden;
287   box-shadow:
288     0 1px 1px rgba(0, 0, 0, 0.2),
289     0 8px 0 -3px #f6f6f6,
290     0 9px 1px -3px rgba(0, 0, 0, 0.2),
291     0 16px 0 -6px #f6f6f6,
292     0 17px 2px -6px rgba(0, 0, 0, 0.2);
293 }
294
295 .todo-count {
296   float: left;
297   text-align: left;
```

```
298 }
299
300 .todo-count strong {
301   font-weight: 300;
302 }
303
304 .filters {
305   margin: 0;
306   padding: 0;
307   list-style: none;
308   position: absolute;
309   right: 0;
310   left: 0;
311 }
312
313 .filters li {
314   display: inline;
315 }
316
317 .filters li a {
318   color: inherit;
319   margin: 3px;
320   padding: 3px 7px;
321   text-decoration: none;
322   border: 1px solid transparent;
323   border-radius: 3px;
324 }
325
326 .filters li a:hover {
327   border-color: #db7676;
328 }
329
330 .filters li a.selected {
331   border-color: #ce4646;
332 }
333
334 .clear-completed,
335 html .clear-completed:active {
336   float: right;
337   position: relative;
338   line-height: 19px;
339   text-decoration: none;
340   cursor: pointer;
341 }
342
343 .clear-completed:hover {
344   text-decoration: underline;
```

```
345 }
346
347 .info {
348     margin: 65px auto 0;
349     color: #4d4d4d;
350     font-size: 11px;
351     text-shadow: 0 1px 0 rgba(255, 255, 255, 0.5);
352     text-align: center;
353 }
354
355 .info p {
356     line-height: 1;
357 }
358
359 .info a {
360     color: inherit;
361     text-decoration: none;
362     font-weight: 400;
363 }
364
365 .info a:hover {
366     text-decoration: underline;
367 }
368
369 /*
370  Hack to remove background from Mobile Safari.
371  Can't use it globally since it destroys checkboxes in Firefox
372  */
373 @media screen and (-webkit-min-device-pixel-ratio: 0) {
374     .toggle-all,
375     .todo-list li .toggle {
376         background: none;
377     }
378
379     .todo-list li .toggle {
380         height: 40px;
381     }
382 }
383
384 @media (max-width: 430px) {
385     .footer {
386         height: 50px;
387     }
388
389     .filters {
390         bottom: 10px;
391     }
```

```

392 }
393
394 :focus,
395 .toggle:focus + label,
396 .toggle-all:focus + label {
397   box-shadow: 0 0 2px 2px #cf7d7d;
398   outline: 0;
399 }

```

4. 完整代码

4.1 Option版

store/todo.js

```

1  import { defineStore } from 'pinia'
2
3  // 选中的类型
4  export const TODO_TYPE = {
5    all: 'all', // 所有
6    unfinished: 'unfinished', // 未完成
7    finished: 'finished' // 已完成
8  }
9
10 // 定义并导出
11 export const useTodoStore = defineStore('todo', {
12   // state: 提供共享的数据
13   state: () => ({
14     // 记录当前选择的类型，默认值是 all
15     type: TODO_TYPE.all,
16     // 共享的代办任务列表
17     todos: [
18       { id: 1, text: 'Buy milk', finished: false },
19       { id: 2, text: 'Buy eggs', finished: true },
20       { id: 3, text: 'Buy bread', finished: false }
21     ]
22   }),
23   // 计算属性
24   getters: {
25     // 小选是否全部选中
26     isAll: (state) => state.todos.every((item) => item.finished),
27     // 所有未完成的todos
28     unfinishedTodos: (state) => state.todos.filter((item) => !item.finished),
29     // 计算需要展示的todos
30     showTodos: (state) => {

```

```

31     switch (state.type) {
32       case TODO_TYPE.all:
33         // 返回所有数据
34         return state.todos
35       case TODO_TYPE.unfinished:
36         // 返回所有未完成的
37         return state.todos.filter((item) => !item.finished)
38       case TODO_TYPE.finished:
39         // 返回所有已完成的
40         return state.todos.filter((item) => item.finished)
41     }
42   },
43 },
44 // 定义修改数据的函数
45 actions: {
46   // 添加
47   addTodo(text) {
48     this.todos.push({
49       text,
50       id: this.todos.length + 1,
51       finished: false
52     })
53   },
54   // 删除
55   delTodo(i) {
56     if (window.confirm('确认删除么')) {
57       this.todos.splice(i, 1)
58     }
59   },
60   // 切换状态
61   toggleTodo(flag) {
62     this.todos.forEach((item) => (item.finished = flag))
63   },
64   // 清除已完成
65   clearTodo() {
66     this.todos = this.todos.filter((item) => !item.finished)
67   },
68   // 设置选中类型
69   setType(type) {
70     this.type = type
71   }
72 },
73 })

```

4.2 Setup版

```
1 import { defineStore } from 'pinia'
2
3 import { ref, computed } from 'vue'
4
5 // 选中的类型
6 export const TODO_TYPE = {
7   all: 'all', // 所有
8   unfinished: 'unfinished', // 未完成
9   finished: 'finished' // 已完成
10 }
11 // 定义并导出
12 export const useTodoStore = defineStore('todo', () => {
13   // todos 数组
14   const todos = ref([
15     { id: 1, text: 'Buy milk', finished: false },
16     { id: 2, text: 'Buy eggs', finished: true },
17     { id: 3, text: 'Buy bread', finished: false }
18   ])
19   // 选择的类型
20   const type = ref(TODO_TYPE.all)
21
22   // 小选是否全部选中
23   const isAll = computed(() => {
24     return todos.value.every((item) => item.finished)
25   })
26
27   // 所有未完成的todos
28   const unfinishedTodos = computed(() => {
29     return todos.value.filter((item) => !item.finished)
30   })
31   // 计算需要展示的todos
32   const showTodos = computed(() => {
33     switch (type.value) {
34       case TODO_TYPE.all:
35         return todos.value
36       case TODO_TYPE.unfinished:
37         return todos.value.filter((item) => !item.finished)
38       case TODO_TYPE.finished:
39         return todos.value.filter((item) => item.finished)
40     }
41   })
42   // 添加
43   const addTodo = (text) => {
44     todos.value.push({
```

```

45     text,
46     id: todos.value.length + 1,
47     finished: false
48   })
49 }
50 // 删除
51 const delTodo = (i) => {
52   if (window.confirm('确认删除么')) {
53     todos.value.splice(i, 1)
54   }
55 }
56 // 切换状态
57 const toggleTodo = (flag) => {
58   todos.value.forEach((item) => (item.finished = flag))
59 }
60
61 // 清除
62 const clearTodo = () => {
63   todos.value = todos.value.filter((item) => !item.finished)
64 }
65
66 // 设置选中类型
67 const setType = (selectType) => {
68   type.value = selectType
69 }
70
71 // 返回共享数据和方法
72 return {
73   todos,
74   type,
75   isAll,
76   unfinishedTodos,
77   showTodos,
78   addTodo,
79   delTodo,
80   toggleTodo,
81   clearTodo,
82   setType
83 }
84 })

```

TodoHeader.vue

```

1 <script setup>
2   import { useTodoStore } from '@store/todo'

```

```

3   const todoStore = useTodoStore()
4   // 回车按键添加
5   const onEnter = (e) => {
6     // 获取输入框的值，并去除首尾空格
7     const text = e.target.value.trim()
8     // 非空校验
9     if (!text) return alert('任务名称不能为空')
10    // 调用添加方法
11    todoStore.addToDo(text)
12    // 清空输入框
13    e.target.value = ''
14  }
15 </script>
16
17 <template>
18   <header class="header">
19     <h1>todos</h1>
20     <input
21       @keydown.enter="onEnter"
22       class="new-todo"
23       placeholder="What needs to be finished?"
24       autofocus />
25   </header>
26 </template>

```

TodoMain.vue

```

1 <script setup>
2   import { computed } from 'vue'
3   // 导入 useTodoStore 函数
4   import { useTodoStore } from '@/store/todo'
5   // 获取仓库
6   const todoStore = useTodoStore()
7
8   // 计算属性：小选和全选
9   const isSelected = computed({
10    // 获取自动触发，必须有返回值
11    get() {
12      return todoStore.isAll
13    },
14    // 赋值自动触发，接收要赋予的新值
15    set(flag) {
16      todoStore.toggleTodo(flag)
17    }
18  })

```



```

19 </script>
20
21 <template>
22   <section class="main">
23     <input
24       v-model="isAllSelected"
25       id="toggle-all"
26       class="toggle-all"
27       type="checkbox" />
28     <label for="toggle-all">Mark all as complete</label>
29     <ul class="todo-list">
30       <li
31         v-for="(item, index) in todoStore.showTodos"
32         :key="item.id"
33         :class="{ completed: item.finished }">
34         <div class="view">
35           <input
36             v-model="item.finished"
37             class="toggle"
38             type="checkbox" />
39           <label>{{ item.text }}</label>
40           <button
41             class="destroy"
42             @click="todoStore.delTodo(index)"></button>
43         </div>
44       </li>
45     </ul>
46   </section>
47 </template>

```

TodoFooter.vue

```

1 <script setup>
2   import { useTodoStore, TODO_TYPE } from '@/store/todo'
3
4   const todoStore = useTodoStore()
5 </script>
6
7 <template>
8   <footer class="footer">
9     <span class="todo-count"
10       ><strong>{{ todoStore.unfinishedTodos.length }}</strong> item left</span>
11     >
12     <ul class="filters">
13       <li>

```

```
14     <a
15         @click="todoStore.setType(TODO_TYPE.all)"
16         href="#"
17         :class="{ selected: todoStore.type === TODO_TYPE.all }"
18     >All</a>
19     >
20 </li>
21 <li>
22     <a
23         @click="todoStore.setType(TODO_TYPE.unfinished)"
24         href="#/active"
25         :class="{ selected: todoStore.type === TODO_TYPE.unfinished }"
26     >Active</a>
27     >
28 </li>
29 <li>
30     <a
31         @click="todoStore.setType(TODO_TYPE.finished)"
32         href="#/completed"
33         :class="{ selected: todoStore.type === TODO_TYPE.finished }"
34     >Completed</a>
35     >
36 </li>
37 </ul>
38 <button
39     class="clear-completed"
40     @click="todoStore.clearTodo()">
41     Clear completed
42 </button>
43 </footer>
44 </template>
```