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

1.3 代码示例

components/Add.vue

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

```vue

<template>
  <div>
    <h3>Add组件</h3>
    <p>已知库存数：0</p>
    <button>库存+1</button>
  </div>
</template>
```

components/Sub.vue

```vue
<script setup></script>

<template>
  <div>
    <h3>Sub组件</h3>
    <p>已知库存数：0</p>
    <button>库存-1</button>
  </div>
</template>
```

App.vue

```vue
<script setup>
  import Add from '@/components/Add.vue'
  import Sub from '@/components/Sub.vue'
</script>
<template>
  <div>
    <h1>根组件</h1>
    <p>
      <span>库存总数：</span>
      <input type="number" />
    </p>
  </div>
  <hr />
  <Add />
  <hr />
  <Sub />
</template>

<style>
  #app {
```

```
    width: 350px;
    margin: 60px auto;
    border: 1px solid #ccc;
    padding: 4px;
  }
</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

```
yarn add pinia

# or

npm i pinia
```

2.2.2 创建 pinia 实例 并注册

```
// main.js

// 导入
import { createPinia } from 'pinia'

// 创建 pinia 实例
const pinia = createPinia()

// 注册实例
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

```
import { defineStore } from 'pinia'

// 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`)

// 第一个参数是你的应用中 Store 的唯一 ID。
export const useXxxStore = defineStore('仓库名称', Setup函数或Option对象)

// 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。
// 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。

// 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)

```
import { defineStore } from 'pinia'
import { ref, computed } from 'vue'

// 定义一个名为 stock的store 并导出
export const useStockStore = defineStore('stock', () => {
  // 初始化库存数据
  const stock = ref(20)

  // 计算库存的两倍
  const doubleStock = computed(() => {
    return stock.value * 2
  })

  // 增加
  const addStock = () => {
    stock.value++
  }
  // 减少
  const subStock = () => {
    stock.value--
  }

  // 返回共享数据和操作函数
  return {
    stock,
    doubleStock,
    addStock,
```

```
    subStock
  }
})
```

2.2.3.2 选项式API(Option对象)

```
import { defineStore } from 'pinia'

// 定义一个名为 stock的store 并导出
export const useStockStore = defineStore('stock', {
  // 共享数据
  state: () => ({
   // 库存初始值
    stock: 20
  }),
  // 基于共享数据的计算属性
  getters: {
    // 计算库存的两倍
    doubleStock: (state) => state.stock * 2
  },
  // 修改共享数据的方法
  actions: {
    // 增加库存
    addStock() {
      this.stock++
    },
    // 减少库存
    subStock() {
      this.stock--
    }
  }
})
```

2.2.4 使用仓库

Add.vue

```
<script setup>
  // 导入 使用库存仓库
  import { useStockStore } from '@/store/stock.js'

  // 创建库存仓库
  const stockStore = useStockStore()
</script>
```

```vue

<template>
  <div>
    <h3>Add组件</h3>
    <p>已知库存数: {{ stockStore.stock }}</p>
    <button @click="stockStore.addStock()">库存+1</button>
  </div>
</template>
```

Sub.vue

```vue
<script setup>
  // 导入 使用库存仓库
  import { useStockStore } from '@/store/stock.js'

  // 创建库存仓库
  const stockStore = useStockStore()
</script>

<template>
  <div>
    <h3>Sub组件</h3>
    <p>已知库存数: {{ stockStore.stock }}</p>
    <button @click="stockStore.subStock()">库存-1</button>
  </div>
</template>
```

App.vue

```vue
<script setup>
  import Add from './components/Add.vue'
  import Sub from './components/Sub.vue'

  import { useStockStore } from '@/store/stock'

  const stockStore = useStockStore()
</script>

<template>
  <h1>根组件</h1>
  库存总数:
  <input
    type="number"
```

```
    v-model.number="stockStore.stock" />
  <p>stock 的翻倍值 = {{ stockStore.doubleStock }}</p>
  <hr />
  <Add />
  <hr />
  <Sub />
</template>

<style>
  #app {
    width: 350px;
    margin: 100px auto;
    padding: 5px 10px;
    border: 1px solid #ccc;
  }
</style>
```

2.3 总结

1. 如何定义 store ?

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

2. 如何使用 store ?

```
// 导入
import { useXxxStore } from './xxx'

// 创建
const xxxStore = useXxxStore()

// 使用
xxxStore.数据名
xxxStore.计算属性名
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

```vue
<script setup></script>

<template>
  <header class="header">
    <h1>todos</h1>
    <input class="new-todo" placeholder="What needs to be finished?" autofocus
/>
  </header>
</template>
```

components/TodoMain.vue

```vue
<script setup></script>

<template>
  <section class="main">
    <input id="toggle-all" class="toggle-all" type="checkbox" />
    <label for="toggle-all">Mark all as complete</label>
    <ul class="todo-list">
      <li>
        <div class="view">
```

```
          <input class="toggle" type="checkbox" />
          <label>Buy eggs</label>
          <button class="destroy"></button>
        </div>
      </li>
      <li>
        <div class="view">
          <input class="toggle" type="checkbox" checked />
          <label>打王者</label>
          <button class="destroy"></button>
        </div>
      </li>
    </ul>
  </section>
</template>
```

components/TodoFooter.vue

```
<script setup></script>

<template>
  <footer class="footer">
    <span class="todo-count"><strong>0</strong> item left</span>
    <ul class="filters">
      <li>
        <a href="#/" class="selected">All</a>
      </li>
      <li>
        <a href="#/active">Active</a>
      </li>
      <li>
        <a href="#/completed">Completed</a>
      </li>
    </ul>
    <button class="clear-completed">
      Clear completed
    </button>
  </footer>
</template>
```

App.vue

```
<script setup>
```

```
  import './assets/style.css'
  import TodoHeader from './components/TodoHeader.vue'
  import TodoMain from './components/TodoMain.vue'
  import TodoFooter from './components/TodoFooter.vue'
</script>

<template>
  <section class="todoapp">
    <todo-header />
    <todo-main />
    <todo-footer />
  </section>
</template>
```

assets/style.css

```
@charset 'utf-8';

html,
body {
  margin: 0;
  padding: 0;
}

button {
  margin: 0;
  padding: 0;
  border: 0;
  background: none;
  font-size: 100%;
  vertical-align: baseline;
  font-family: inherit;
  font-weight: inherit;
  color: inherit;
  -webkit-appearance: none;
  appearance: none;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

body {
  font:
    14px 'Helvetica Neue',
    Helvetica,
    Arial,
```

```
    sans-serif;
  line-height: 1.4em;
  background: #f5f5f5;
  color: #111111;
  min-width: 230px;
  max-width: 550px;
  margin: 0 auto;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  font-weight: 300;
}

.hidden {
  display: none;
}

.todoapp {
  background: #fff;
  margin: 130px 0 40px 0;
  position: relative;
  box-shadow:
    0 2px 4px 0 rgba(0, 0, 0, 0.2),
    0 25px 50px 0 rgba(0, 0, 0, 0.1);
}

.todoapp input::-webkit-input-placeholder {
  font-style: italic;
  font-weight: 400;
  color: rgba(0, 0, 0, 0.4);
}

.todoapp input::-moz-placeholder {
  font-style: italic;
  font-weight: 400;
  color: rgba(0, 0, 0, 0.4);
}

.todoapp input::input-placeholder {
  font-style: italic;
  font-weight: 400;
  color: rgba(0, 0, 0, 0.4);
}

.todoapp h1 {
  position: absolute;
  top: -100px;
  width: 100%;
```

```
  font-size: 50px;
  font-weight: 200;
  text-align: center;
  color: #b83f45;
  -webkit-text-rendering: optimizeLegibility;
  -moz-text-rendering: optimizeLegibility;
  text-rendering: optimizeLegibility;
}

.new-todo,
.edit {
  position: relative;
  margin: 0;
  width: 100%;
  font-size: 24px;
  font-family: inherit;
  font-weight: inherit;
  line-height: 1.4em;
  color: inherit;
  padding: 6px;
  border: 1px solid #999;
  box-shadow: inset 0 -1px 5px 0 rgba(0, 0, 0, 0.2);
  box-sizing: border-box;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

.new-todo {
  padding: 16px 16px 16px 60px;
  height: 65px;
  border: none;
  background: rgba(0, 0, 0, 0.003);
  box-shadow: inset 0 -2px 1px rgba(0, 0, 0, 0.03);
}

.main {
  position: relative;
  z-index: 2;
  border-top: 1px solid #e6e6e6;
}

.toggle-all {
  width: 1px;
  height: 1px;
  border: none; /* Mobile Safari */
  opacity: 0;
  position: absolute;
```

```
  right: 100%;
  bottom: 100%;
}

.toggle-all + label {
  display: flex;
  align-items: center;
  justify-content: center;
  width: 45px;
  height: 65px;
  font-size: 0;
  position: absolute;
  top: -65px;
  left: -0;
}

.toggle-all + label:before {
  content: ' ';
  display: inline-block;
  font-size: 22px;
  color: #949494;
  padding: 10px 27px 10px 27px;
  -webkit-transform: rotate(90deg);
  transform: rotate(90deg);
}

.toggle-all:checked + label:before {
  color: #484848;
}

.todo-list {
  margin: 0;
  padding: 0;
  list-style: none;
}

.todo-list li {
  position: relative;
  font-size: 24px;
  border-bottom: 1px solid #ededed;
}

.todo-list li:last-child {
  border-bottom: none;
}

.todo-list li.editing {
```

```
  border-bottom: none;
  padding: 0;
}

.todo-list li.editing .edit {
  display: block;
  width: calc(100% - 43px);
  padding: 12px 16px;
  margin: 0 0 0 43px;
}

.todo-list li.editing .view {
  display: none;
}

.todo-list li .toggle {
  text-align: center;
  width: 40px;
  /* auto, since non-WebKit browsers doesn't support input styling */
  height: auto;
  position: absolute;
  top: 0;
  bottom: 0;
  margin: auto 0;
  border: none; /* Mobile Safari */
  -webkit-appearance: none;
  appearance: none;
}

.todo-list li .toggle {
  opacity: 0;
}

.todo-list li .toggle + label {
  /*
    Firefox requires `#` to be escaped - https://bugzilla.mozilla.org/show_bug.cgi?id=922433
    IE and Edge requires *everything* to be escaped to render, so we do that instead of just the `#` - https://developer.microsoft.com/en-us/microsoft-edge/platform/issues/7157459/
  */
  background-image: url('data:image/svg+xml;utf8,%3Csvg%20xmlns%3D%22http%3A//www.w3.org/2000/svg%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%20fill%3D%22none%22%20stroke%3D%22%23949494%22%20stroke-width%3D%223%22/%3E%3C/svg%3E');
```

```
  background-repeat: no-repeat;
  background-position: center left;
}

.todo-list li .toggle:checked + label {
  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');
}

.todo-list li label {
  overflow-wrap: break-word;
  padding: 15px 15px 15px 60px;
  display: block;
  line-height: 1.2;
  transition: color 0.4s;
  font-weight: 400;
  color: #484848;
}

.todo-list li.completed label {
  color: #949494;
  text-decoration: line-through;
}

.todo-list li .destroy {
  display: none;
  position: absolute;
  top: 0;
  right: 10px;
  bottom: 0;
  width: 40px;
  height: 40px;
  margin: auto 0;
  font-size: 30px;
  color: #949494;
  transition: color 0.2s ease-out;
}

.todo-list li .destroy:hover,
.todo-list li .destroy:focus {
  color: #c18585;
}
```

```
251
252 .todo-list li .destroy:after {
253   content: '×';
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;
```

```
}

.todo-count strong {
  font-weight: 300;
}

.filters {
  margin: 0;
  padding: 0;
  list-style: none;
  position: absolute;
  right: 0;
  left: 0;
}

.filters li {
  display: inline;
}

.filters li a {
  color: inherit;
  margin: 3px;
  padding: 3px 7px;
  text-decoration: none;
  border: 1px solid transparent;
  border-radius: 3px;
}

.filters li a:hover {
  border-color: #db7676;
}

.filters li a.selected {
  border-color: #ce4646;
}

.clear-completed,
html .clear-completed:active {
  float: right;
  position: relative;
  line-height: 19px;
  text-decoration: none;
  cursor: pointer;
}

.clear-completed:hover {
  text-decoration: underline;
```

```
}

.info {
  margin: 65px auto 0;
  color: #4d4d4d;
  font-size: 11px;
  text-shadow: 0 1px 0 rgba(255, 255, 255, 0.5);
  text-align: center;
}

.info p {
  line-height: 1;
}

.info a {
  color: inherit;
  text-decoration: none;
  font-weight: 400;
}

.info a:hover {
  text-decoration: underline;
}

/*
  Hack to remove background from Mobile Safari.
  Can't use it globally since it destroys checkboxes in Firefox
*/
@media screen and (-webkit-min-device-pixel-ratio: 0) {
  .toggle-all,
  .todo-list li .toggle {
    background: none;
  }

  .todo-list li .toggle {
    height: 40px;
  }
}

@media (max-width: 430px) {
  .footer {
    height: 50px;
  }

  .filters {
    bottom: 10px;
  }
```

```
}

:focus,
.toggle:focus + label,
.toggle-all:focus + label {
  box-shadow: 0 0 2px 2px #cf7d7d;
  outline: 0;
}
```

4. 完整代码

4.1 Option版

`store/todo.js`

```
import { defineStore } from 'pinia'

// 选中的类型
export const TODO_TYPE = {
  all: 'all', // 所有
  unfinished: 'unfinished', // 未完成
  finished: 'finished' // 已完成
}

// 定义并导出
export const useTodoStore = defineStore('todo', {
  // state: 提供共享的数据
  state: () => ({
    // 记录当前选择的类型，默认值是 all
    type: TODO_TYPE.all,
    // 共享的代办任务列表
    todos: [
      { id: 1, text: 'Buy milk', finished: false },
      { id: 2, text: 'Buy eggs', finished: true },
      { id: 3, text: 'Buy bread', finished: false }
    ]
  }),
  // 计算属性
  getters: {
    // 小选是否全部选中
    isAll: (state) => state.todos.every((item) => item.finished),
    // 所有未完成的todos
    unfinishedTodos: (state) => state.todos.filter((item) => !item.finished),
    // 计算需要展示的todos
    showTodos: (state) => {
```

```
      switch (state.type) {
        case TODO_TYPE.all:
          // 返回所有数据
          return state.todos
        case TODO_TYPE.unfinished:
          // 返回所有未完成的
          return state.todos.filter((item) => !item.finished)
        case TODO_TYPE.finished:
          // 返回所有已完成的
          return state.todos.filter((item) => item.finished)
      }
    }
  },
  // 定义修改数据的函数
  actions: {
    // 添加
    addTodo(text) {
      this.todos.push({
        text,
        id: this.todos.length + 1,
        finished: false
      })
    },
    // 删除
    delTodo(i) {
      if (window.confirm('确认删除么')) {
        this.todos.splice(i, 1)
      }
    },
    // 切换状态
    toggleTodo(flag) {
      this.todos.forEach((item) => (item.finished = flag))
    },
    // 清除已完成
    clearTodo() {
      this.todos = this.todos.filter((item) => !item.finished)
    },
    // 设置选中类型
    setType(type) {
      this.type = type
    }
  }
})
```


4.2 Setup版

store/todo.js

```
import { defineStore } from 'pinia'

import { ref, computed } from 'vue'

// 选中的类型
export const TODO_TYPE = {
  all: 'all', // 所有
  unfinished: 'unfinished', // 未完成
  finished: 'finished' // 已完成
}
// 定义并导出
export const useTodoStore = defineStore('todo', () => {
  // todos 数组
  const todos = ref([
    { id: 1, text: 'Buy milk', finished: false },
    { id: 2, text: 'Buy eggs', finished: true },
    { id: 3, text: 'Buy bread', finished: false }
  ])
  // 选择的类型
  const type = ref(TODO_TYPE.all)

  // 小选是否全部选中
  const isAll = computed(() => {
    return todos.value.every((item) => item.finished)
  })

  // 所有未完成的todos
  const unfinishedTodos = computed(() => {
    return todos.value.filter((item) => !item.finished)
  })
  // 计算需要展示的todos
  const showTodos = computed(() => {
    switch (type.value) {
      case TODO_TYPE.all:
        return todos.value
      case TODO_TYPE.unfinished:
        return todos.value.filter((item) => !item.finished)
      case TODO_TYPE.finished:
        return todos.value.filter((item) => item.finished)
    }
  })
  // 添加
  const addTodo = (text) => {
    todos.value.push({
```

```
      text,
      id: todos.value.length + 1,
      finished: false
    })
  }
  // 删除
  const delTodo = (i) => {
    if (window.confirm('确认删除么')) {
      todos.value.splice(i, 1)
    }
  }
  // 切换状态
  const toggleTodo = (flag) => {
    todos.value.forEach((item) => (item.finished = flag))
  }

  // 清除
  const clearTodo = () => {
    todos.value = todos.value.filter((item) => !item.finished)
  }

  // 设置选中类型
  const setType = (selectType) => {
    type.value = selectType
  }

  // 返回共享数据和方法
  return {
    todos,
    type,
    isAll,
    unfinishedTodos,
    showTodos,
    addTodo,
    delTodo,
    toggleTodo,
    clearTodo,
    setType
  }
})
```

TodoHeader.vue

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

```
  const todoStore = useTodoStore()
  // 回车按键添加
  const onEnter = (e) => {
    // 获取输入框的值，并去除首尾空格
    const text = e.target.value.trim()
    // 非空校验
    if (!text) return alert('任务名称不能为空')
    // 调用添加方法
    todoStore.addTodo(text)
    // 清空输入框
    e.target.value = ''
  }
</script>

<template>
  <header class="header">
    <h1>todos</h1>
    <input
      @keydown.enter="onEnter"
      class="new-todo"
      placeholder="What needs to be finished?"
      autofocus />
  </header>
</template>
```

TodoMain.vue

```
<script setup>
  import { computed } from 'vue'
  // 导入 useTodoStore 函数
  import { useTodoStore } from '@/store/todo'
  // 获取仓库
  const todoStore = useTodoStore()

  // 计算属性：小选和全选
  const isAllSelected = computed({
    // 获取自动触发，必须有返回值
    get() {
      return todoStore.isAll
    },
    // 赋值自动触发，接收要赋予的新值
    set(flag) {
      todoStore.toggleTodo(flag)
    }
  })
```

```vue
</script>

<template>
  <section class="main">
    <input
      v-model="isAllSelected"
      id="toggle-all"
      class="toggle-all"
      type="checkbox" />
    <label for="toggle-all">Mark all as complete</label>
    <ul class="todo-list">
      <li
        v-for="(item, index) in todoStore.showTodos"
        :key="item.id"
        :class="{ completed: item.finished }">
        <div class="view">
          <input
            v-model="item.finished"
            class="toggle"
            type="checkbox" />
          <label>{{ item.text }}</label>
          <button
            class="destroy"
            @click="todoStore.delTodo(index)"></button>
        </div>
      </li>
    </ul>
  </section>
</template>
```

TodoFooter.vue

```vue
<script setup>
  import { useTodoStore, TODO_TYPE } from '@/store/todo'

  const todoStore = useTodoStore()
</script>

<template>
  <footer class="footer">
    <span class="todo-count"
      ><strong>{{ todoStore.unfinishedTodos.length }}</strong> item left</span
    >
    <ul class="filters">
      <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>
```