

22 春 字节前端 第 4 课: Todo MVC 示例

GitHub

<https://github.com/ycaptain/modernjs-todomvc>

需求分析

<https://github.com/tastejs/todomvc/blob/master/app-spec.md#functionality>

- No Todos: 隐藏 `#main` 和 `#footer`
- New Todo:
 - 利用 `autofocus` 属性自动聚焦输入框
 - 回车创建 `todo`
 - `trim()` 清空输入前后的空格
 - 空检测, 无内容不创建新 `todo`
- Mark all as complete
 - 标记全部 `todos` 为完成
- Item
 - 点击 `checkbox` 修改 `todo` 的状态为 完成/未完成
 - 双击 `<label>` 进入编辑模式, 将 `.editing` 加入 `li` 的 `class`
 - `hover` 时展示销毁按钮, 点击删除 `.destroy` 该 `todo`
- Editing
 - 编辑态隐藏 `item` 的其它按钮
 - 当失焦 `blur` 或点回车时, 保存编辑结果, 并从 `li` 的 `class` 中移除 `.editing`
 - `trim()` 清空输入前后的空格
 - 空检测, 无内容则删除该 `todo`
 - 当按下 退出键 `escape` 时, 退出编辑态, 恢复编辑前的数据
- Counter
 - 展示当前剩余未完成 `todo` 数量
 - 数字用标签 `` 包装
 - 处理单复数, `item or items`

- Clear completed button
 - 点击 `Clear completed` 删除所有完成项
 - 当没有未完成项时隐藏该按钮
- Persistence
 - 将数据保存到 `localStorage`
 - 使用字段名 `id`, `title`, `completed`
 - `localStorage` 字段名: `todos-[framework]`
 - 编辑状态不保存
- Routing
 - 设置路由状态为 `#/` (all - 默认), `#/active`, `#/completed`
 - 当路由变化时, 过滤 `todo list` 中的数据为对应数据
 - 更新 `class selected` 到对应的路由按钮上

开发过程

初始化

创建 GitHub Repo

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `git@github.com:ycaptain/modernjs-todomvc.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# modernjs-todomvc" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:ycaptain/modernjs-todomvc.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:ycaptain/modernjs-todomvc.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

ProTip! Use the URL for this page when adding GitHub as a remote.

创建项目

1 `npx @modern-js/create modernjs-todomvc`

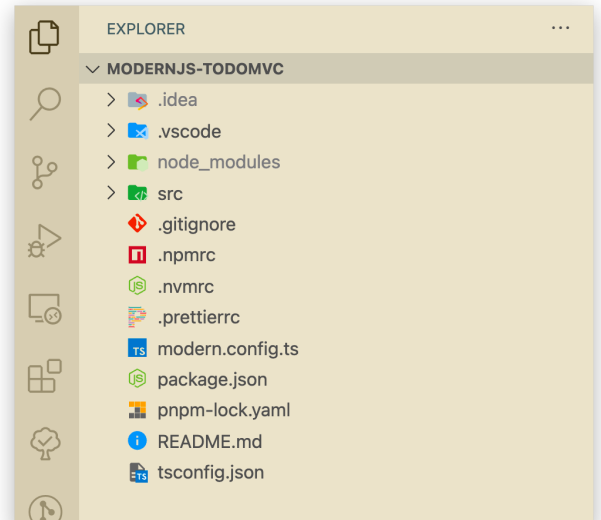
```
ycaptain@ycaptaindeM1Max:~/codespace/github
npx: 1 安装成功, 用时 1.398 秒
? 请选择你希望创建的工程类型 应用
? 请选择开发语言 TS
? 请选择包管理工具 pnpm
? 是否需要支持以下类型应用 不需要
? 是否需要调整默认配置? 否

Update available! 6.24.0 -> 6.32.3.
Change log: https://github.com/pnpm/pnpm/releases/tag/v6.32.3
Run pnpm add -g pnpm to update.

Follow @pnpmjs for updates: https://twitter.com/pnpmjs

Downloading registry.npmirror.com/typescript/4.6.3: 11.4 MB/11.4 MB, done
Packages: +1327
Packages are hard linked from the content-addressable store to the virtual store.
Content-addressable store is at: /Users/ycaptain/.pnpm-store/v3
Virtual store is at:
node_modules/.pnpm
Progress: resolved 1256, reused 1174, downloaded 66, added 1327, done
node_modules/.pnpm/core-js@3.9.1/node_modules/core-js: Running postinstall script...
node_modules/.pnpm/core-js@3.9.1/node_modules/core-js: Running postinstall script, done in 44ms
node_modules/.pnpm/styled-components@5.3.5_react-dom@17.0.2_react@17.0.2/node_modules/styled-components: Running postinstall script
, done in 49ms
node_modules/.pnpm/husky@1.3.0/node_modules/husky: Running install script, done in 168ms
node_modules/.pnpm/husky@1.3.0/node_modules/husky: Running postinstall script, done in 525ms
node_modules/.pnpm/esbuild@0.13.19/node_modules/esbuild: Running postinstall script, done in 168ms

dependencies:
+ @modern-js/runtime 1.2.3
+ react 17.0.2
```









```
[INFO] 依赖自动安装成功
[INFO] git 仓库初始化成功
[INFO] 创建成功!
可在新项目的目录下运行以下命令:
pnpm run dev      # 按开发环境的要求, 运行和调试项目
pnpm run build    # 按产品环境的要求, 构建项目
pnpm run start    # 按产品环境的要求, 运行项目
pnpm run lint     # 检查和修复所有代码
pnpm run new      # 继续创建更多项目要素, 比如应用入口
[INFO] 你可以执行 'cd modernjs-todomvc' 进入目录
```




绑定 GitHub

```
1 cd modernjs-todomvc
2 git remote add origin <GitHub 仓库地址>
3 git push origin -u origin main
```

```
ycaptain@ycaptaindeM1Max:~/codespace/github/modernjs-todomvc — 92x24
→ cd modernjs-todomvc

modernjs-todomvc on  main is  v0.1.0 via  v14.18.2
→ git remote add origin git@github.com:ycaptain/modernjs-todomvc.git

modernjs-todomvc on  main is  v0.1.0 via  v14.18.2
→ git push -u origin main
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 10 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (19/19), 126.10 KiB | 416.00 KiB/s, done.
Total 19 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:ycaptain/modernjs-todomvc.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

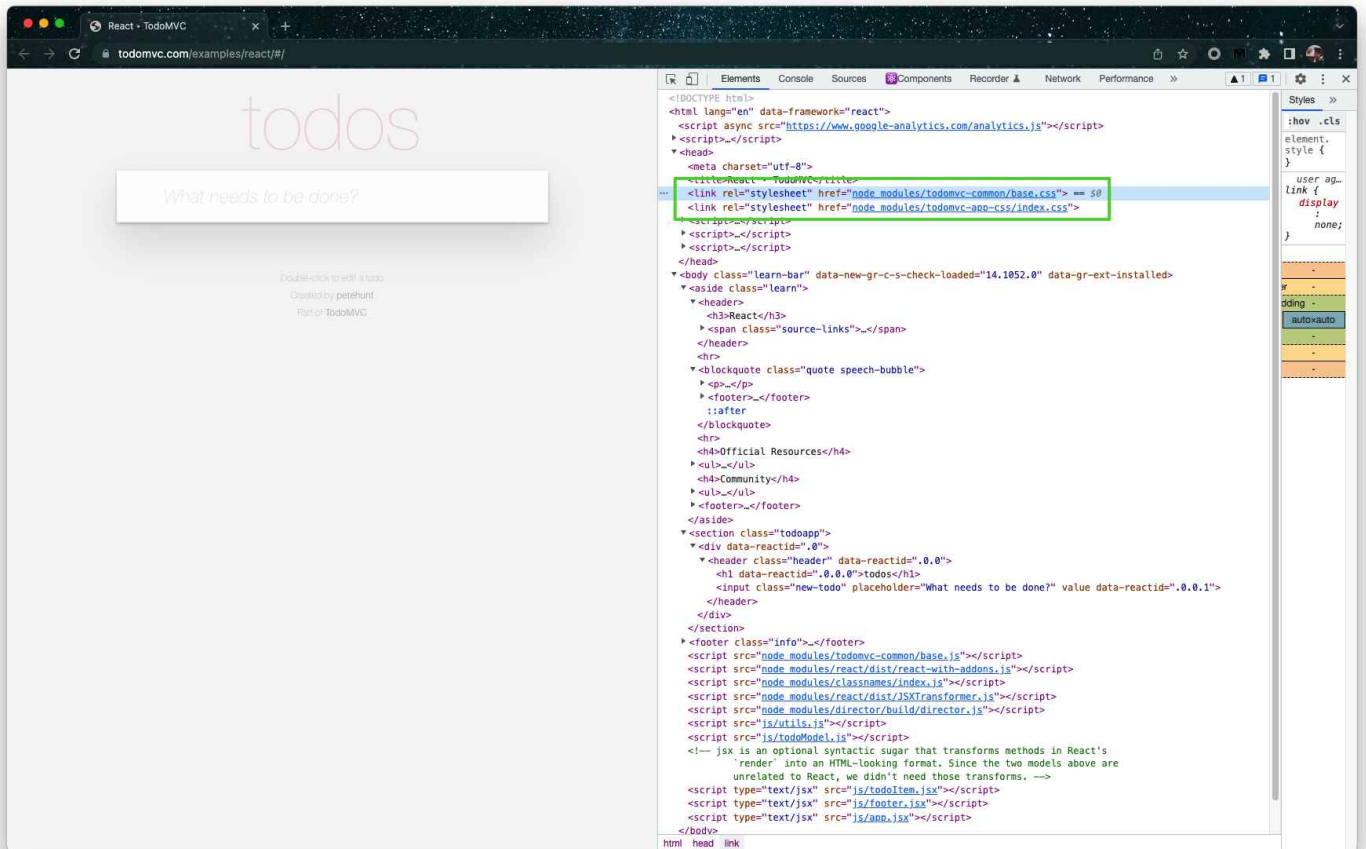
modernjs-todomvc on  main is  v0.1.0 via  v14.18.2 took 5s
→
```

样式还原

进入示例网站，分析样式来源

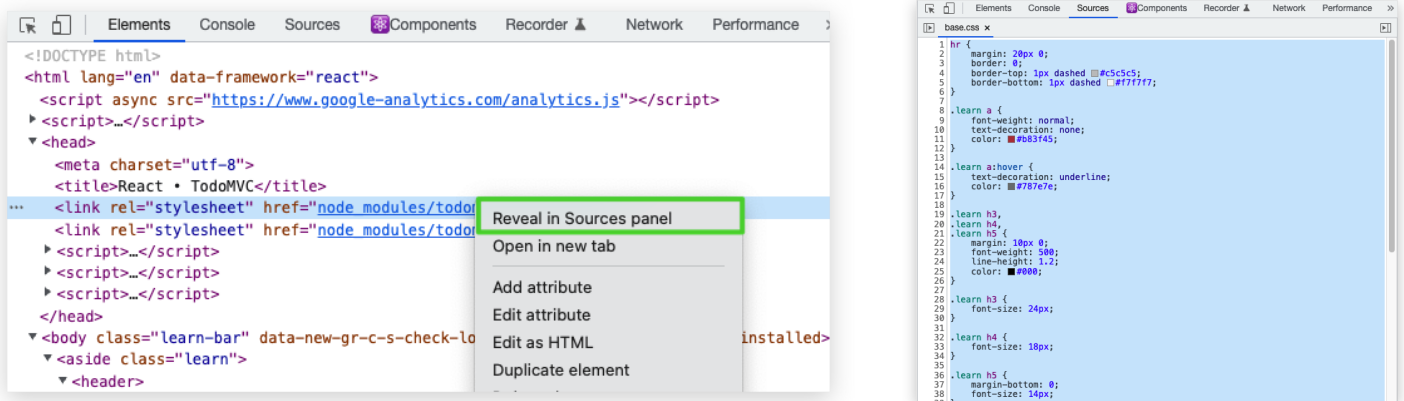
示例网站: <https://todomvc.com/examples/react/#/>

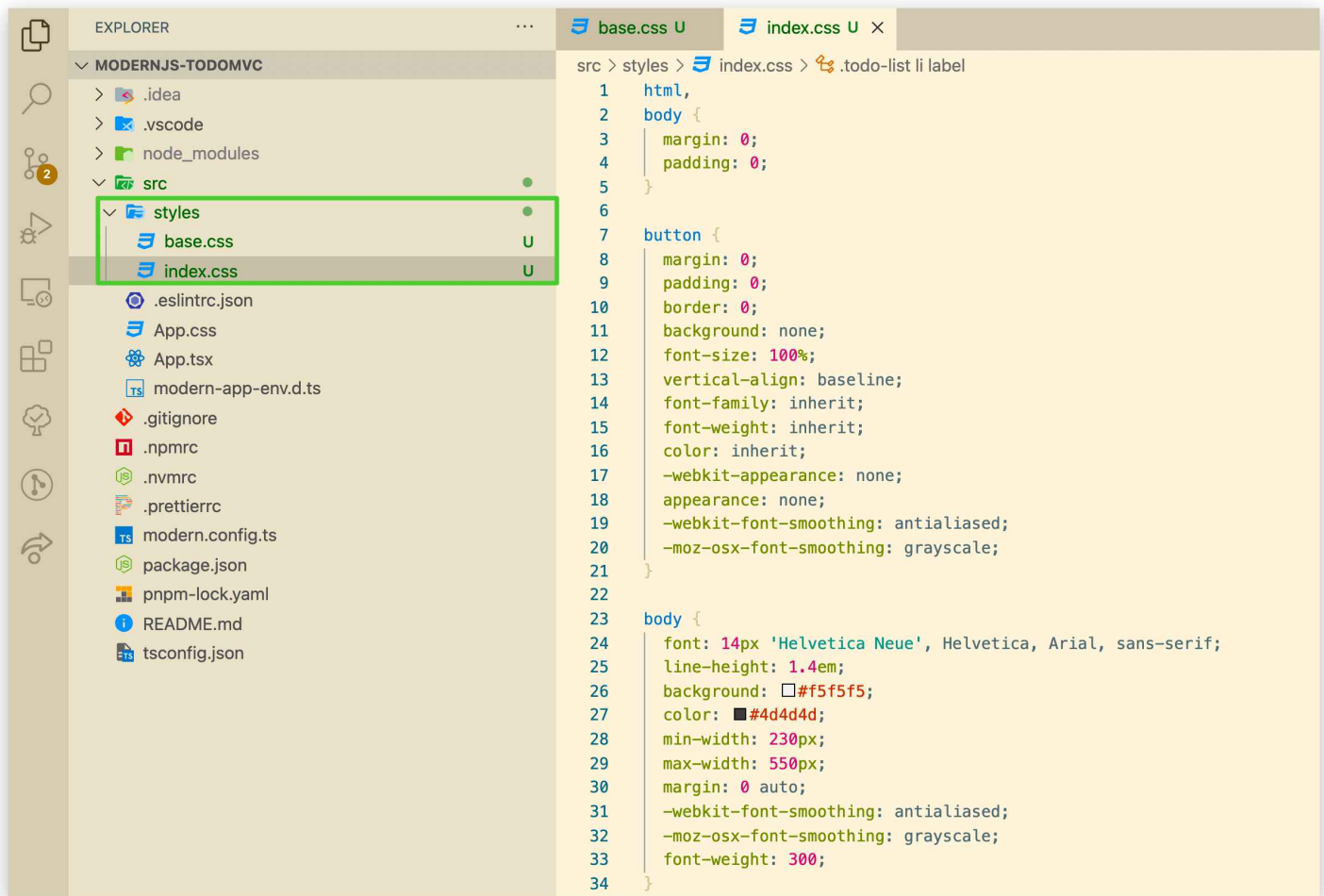
进入示例网站，打开 DevTools，分析得知，项目中只有两个 CSS 文件，并且所有样式均来自这两个 CSS 文件，不存在行内样式。



复制样式文件

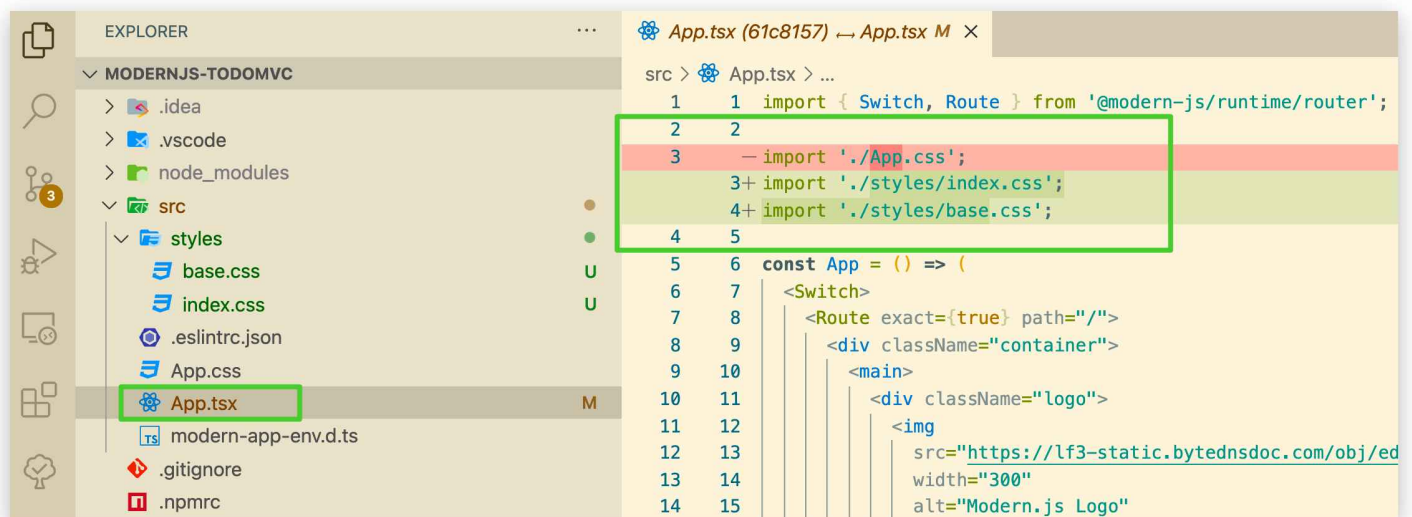
右键点击 CSS 文件链接，将 `base.css` 和 `index.css` 文件复制到项目中。



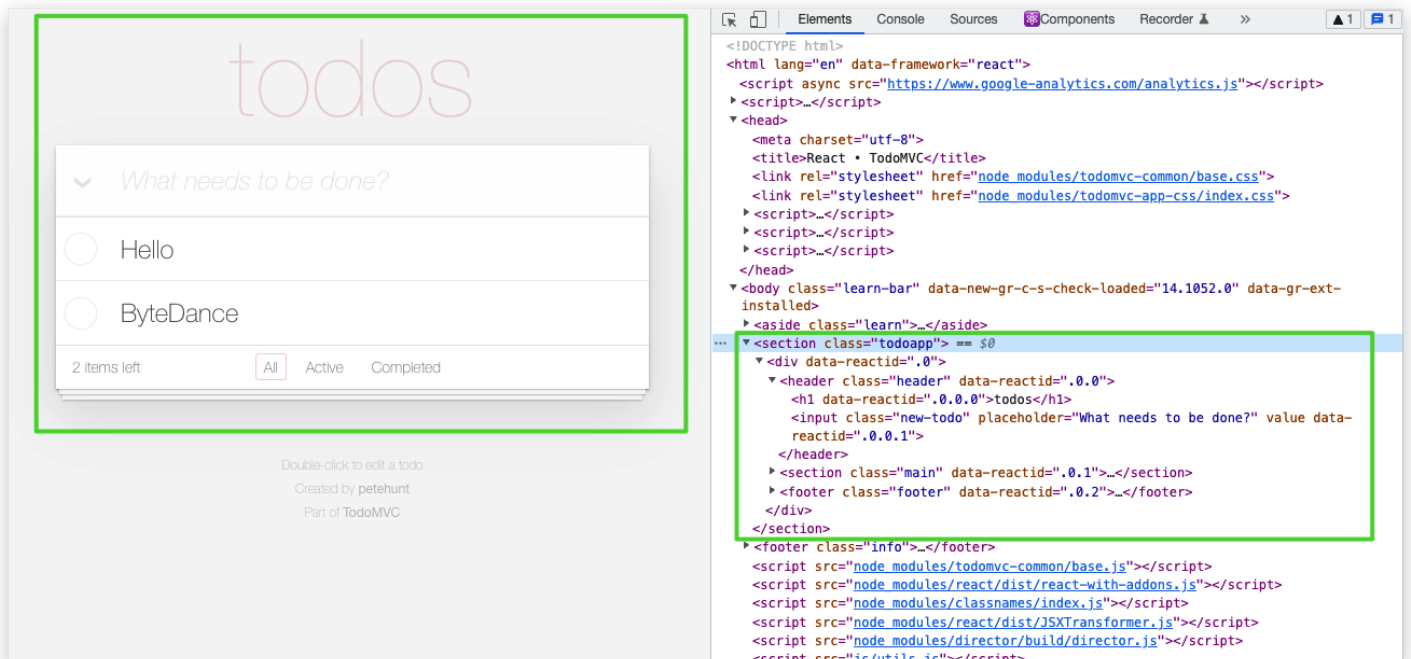


载入样式文件

在 `src/App.tsx` 中替换原来的样式文件



查看示例 DevTools Elements 面板，分析 HTML 结构

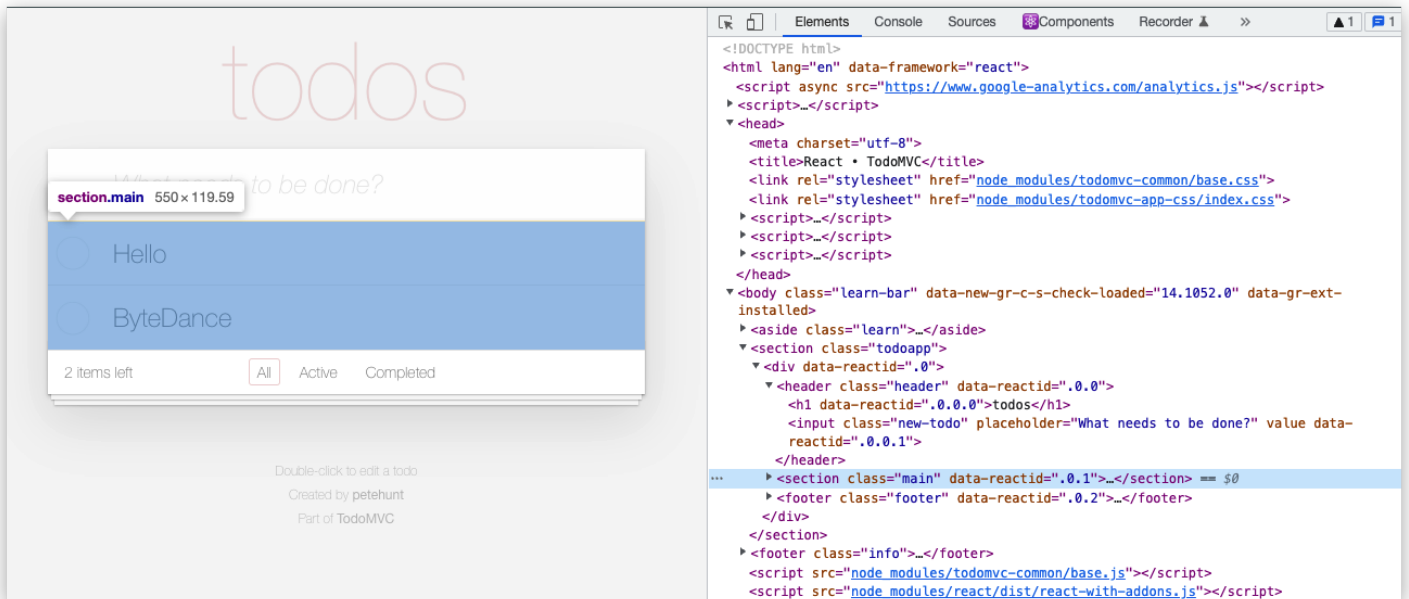


分析得知，Todo MVC 相关 HTML 结构都在 class 属性为 `todoapp` 的 `<section>` 元素内。

section 元素下用一个 `<div>` 标签包裹，其下分为三大模块：

- header: 包含一个大标题 todos 和一个 input 输入框
- section:
 - 全选按钮 input 和样式标签 label
 - 无序列表元素 ul，包含多个 li，其中每个 li 中包含
 - div
 - checkbox，标记该 todo 完成情况
 - label，表示该元素的内容
 - button，删除按钮
 - input，编辑态的输入框，当 li 的 class 包含 `editing` 时展示
- footer
 - span，展示剩余未完成 todo 个数
 - ul，过滤路由，切换 All, Active, Completed 三种状态

Tips: 鼠标 hover 到 Elements 面板中的标签上，网页对应的模块会高亮展示



还原 header

创建文件夹 `src/components`，用于存放视图组件

创建文件 `src/components/Header.tsx`

```
1 import React from 'react';
2
3 export function Header() {
4   return (
5     <header className="header">
6       <h1>todos</h1>
7       <input className="new-todo" placeholder="What needs to be done?" />
8     </header>
9   );
10 }
```

移除 `src/App.tsx` 默认代码，导入 Header 组件

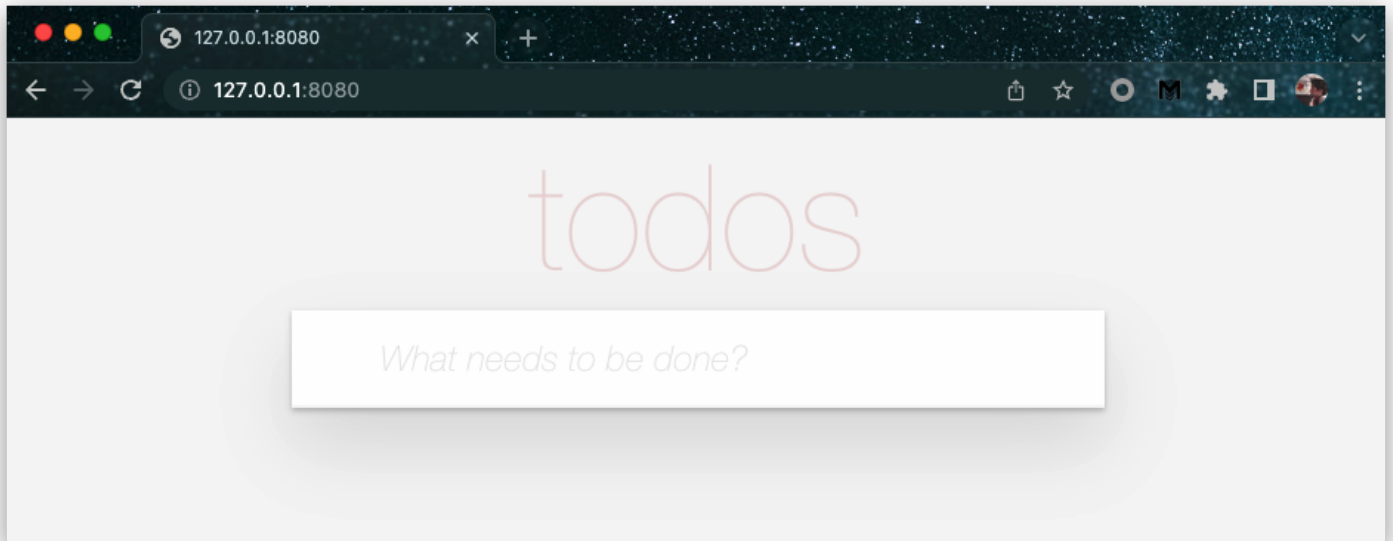
```
1 import React from 'react';
2 import { Header } from './components/Header';
3 import './styles/index.css';
4 import './styles/base.css';
5
6 function App() {
7   return (
8     <section className="todoapp">
9       <Header />
10     </section>
11   );
12 }
```



```
12 }  
13 export default App;
```

运行程序，查看效果

```
1 pnpm dev
```



还原 `main`

创建文件 `src/components/Main.tsx`

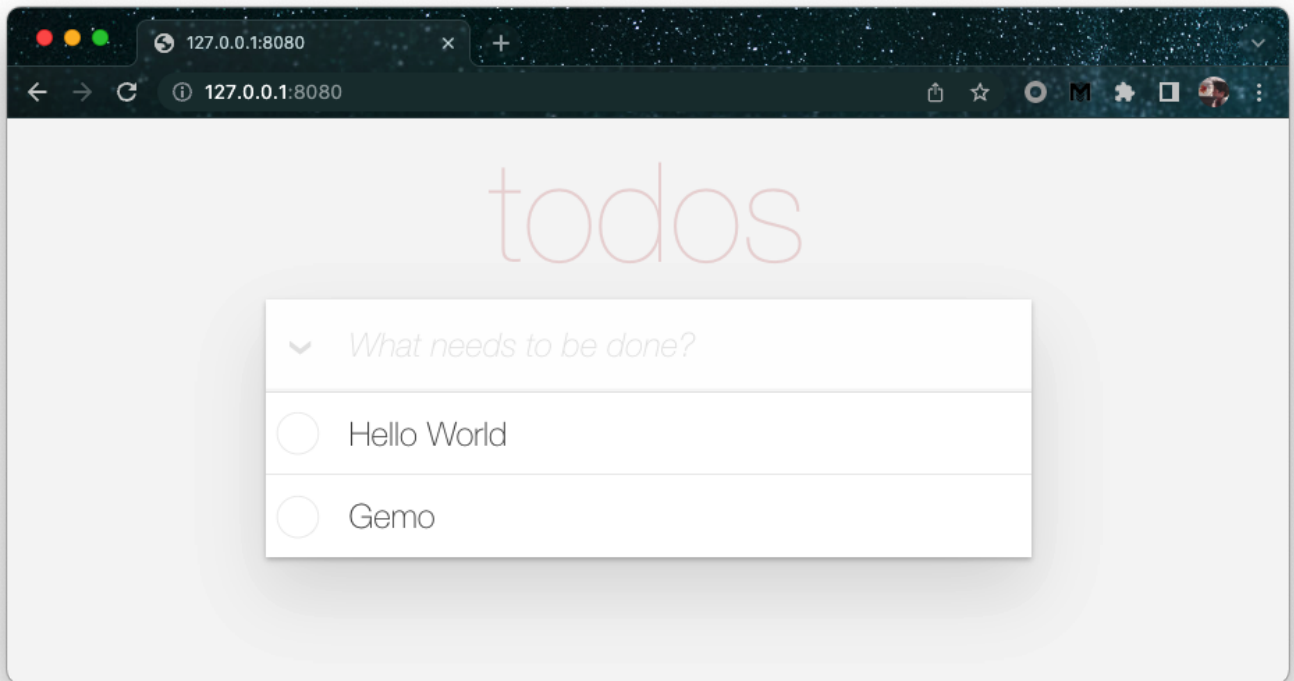
```
1 import React from 'react';  
2  
3 export function Main() {  
4   return (  
5     <section className="main">  
6       <input id="toggle-all" className="toggle-all" type="checkbox" />  
7       <label htmlFor="toggle-all"></label>  
8       <ul className="todo-list">  
9         <li>  
10          <div className="view">  
11            <input className="toggle" type="checkbox" />  
12            <label>Hello World</label>  
13            <button className="destroy" type="button"></button>  
14          </div>  
15          <input className="edit" value="Hello" />  
16        </li>  
17        <li>  
18          <div className="view">
```

```
19         <input className="toggle" type="checkbox" />
20         <label>Gemo</label>
21         <button className="destroy" type="button"></button>
22     </div>
23     <input className="edit" value="ByteDance" />
24 </li>
25 </ul>
26 </section>
27 );
28 }
```

在 `src/App.tsx` 中载入 Main 组件

```
1 import React from 'react';
2 import { Header } from './components/Header';
3 import { Main } from './components/Main';
4 import './styles/index.css';
5 import './styles/base.css';
6
7 function App() {
8     return (
9         <section className="todoapp">
10             <Header />
11             <Main />
12         </section>
13     );
14 }
15 export default App;
```

预览效果



还原 footer

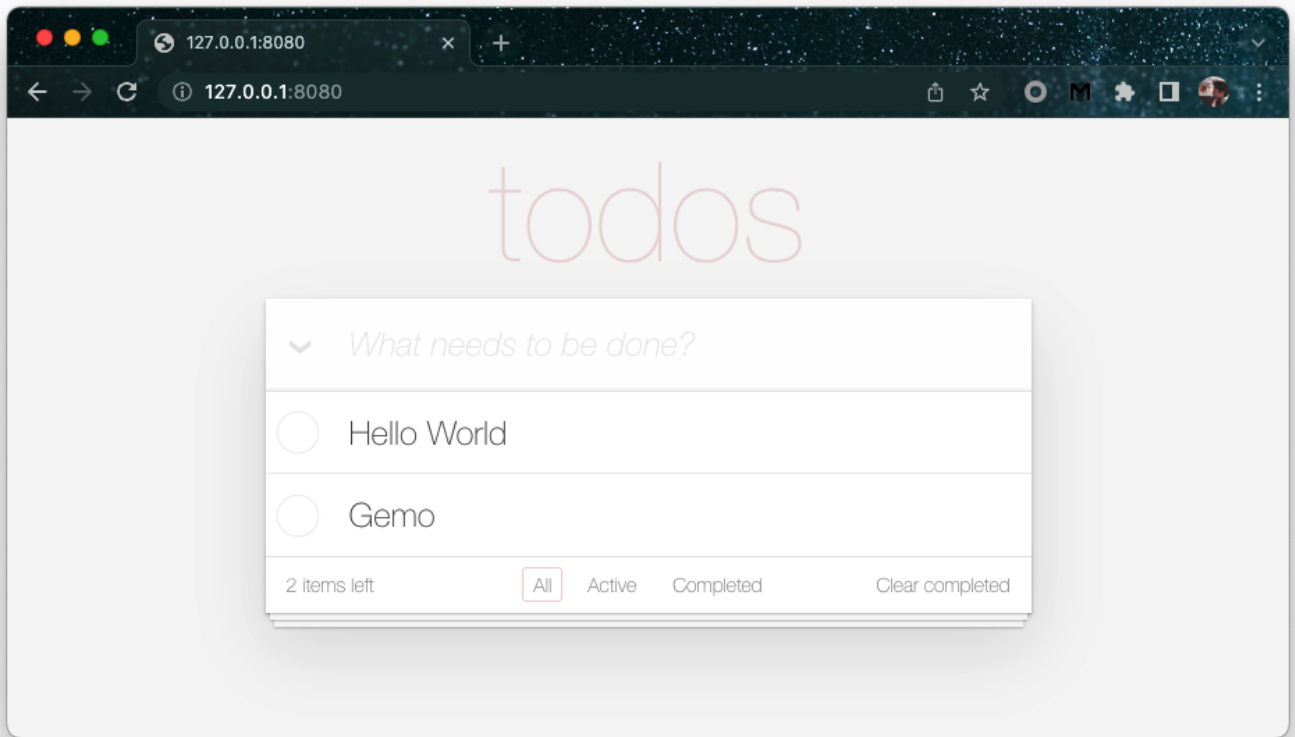
创建文件 `src/components/Footer.tsx`

```
1 import React from 'react';
2
3 export function Footer() {
4   return (
5     <footer className="footer">
6       <span className="todo-count">
7         <strong>2</strong>
8         <span> </span>
9         <span>items</span>
10        <span> left</span>
11      </span>
12      <ul className="filters">
13        <li>
14          <a href="#/" className="selected">
15            All
16          </a>
17        </li>
18        <span> </span>
19        <li>
20          <a href="#/active">Active</a>
21        </li>
22        <span> </span>
23        <li>
```

```
24         <a href="#/completed">Completed</a>
25     </li>
26 </ul>
27     <button className="clear-completed" type="button">
28         Clear completed
29     </button>
30 </footer>
31 );
32 }
```

在 `src/App.tsx` 中载入 Footer 组件

```
1 import React from 'react';
2 import { Header } from './components/Header';
3 import { Main } from './components/Main';
4 import { Footer } from './components/Footer';
5 import './styles/index.css';
6 import './styles/base.css';
7
8 function App() {
9     return (
10         <section className="todoapp">
11             <Header />
12             <Main />
13             <Footer />
14         </section>
15     );
16 }
17 export default App;
```



功能实现

所有功能逻辑都在 Main 和 Footer 组件，Header 组件不包含功能逻辑。

回顾 [需求分析](#)

下载所需依赖

```
1 pnpm add nanoid classnames
2 pnpm add -D @types/nanoid
```

创建 `todoModel` 管理 `todos` 的「增删改查」

创建 `src/utlis.ts`

```
1 import { nanoid } from 'nanoid';
2 import { TodoItem } from '../models/todoModel';
3
4 // 生成唯一 id
5 export function generateId() {
6   return nanoid();
7 }
```

```

7 }
8 // 从 localStorage 获取或向 localStorage 同步数据
9 export function store(namespace: string, data?: Array<TodoItem>) {
10   if (data) {
11     return localStorage.setItem(namespace, JSON.stringify(data));
12   }
13
14   const localStore = localStorage.getItem(namespace);
15   return (localStore && JSON.parse(localStore)) || [];
16 }

```

创建 `src/models/todoModel`

```

1 import { generateId, store } from '../utils';
2
3 export class TodoModel {
4   // 成员变量类型定义
5   key: string;
6
7   todos: Array<TodoItem>;
8
9   onChanges: Array<Subscriber>;
10
11   // 初始化 todos, 通过 key 从 store 中读取历史缓存
12   constructor(key: string) {
13     this.key = key;
14     this.todos = store(key);
15     this.onChanges = [];
16   }
17
18   // 添加监听器
19   subscribe(onChange: Subscriber) {
20     this.onChanges.push(onChange);
21   }
22
23   // 触发监听器, 并更新 store
24   inform() {
25     store(this.key, this.todos);
26     this.onChanges.forEach(function (cb) {
27       cb();
28     });
29   }
30
31   // 添加 todo
32   addTodo(title: string) {

```

```
33     this.todos = this.todos.concat({
34         id: generateId(),
35         title,
36         completed: false,
37     });
38
39     this.inform();
40 }
41
42 // 切换所有 todo 到 完成/未完成 状态
43 toggleAll(checkered: boolean) {
44     this.todos = this.todos.map(function (todo) {
45         return {
46             ...todo,
47             completed: checked,
48         };
49     });
50
51     this.inform();
52 }
53
54 // 切换特定 todo 状态
55 toggle(todoToToggle: TodoItem) {
56     this.todos = this.todos.map(function (todo) {
57         return todo !== todoToToggle
58             ? todo
59             : {
60                 ...todo,
61                 completed: !todo.completed,
62             };
63     });
64
65     this.inform();
66 }
67
68 // 删除特定 todo
69 destroy(todo: TodoItem) {
70     this.todos = this.todos.filter(function (candidate) {
71         return candidate !== todo;
72     });
73
74     this.inform();
75 }
76
77 // 修改特定 todo 内容
78 save(todoToSave: TodoItem, text: string) {
79     this.todos = this.todos.map(function (todo) {
```



```

80     return todo !== todoToSave
81     ? todo
82     : {
83         ...todo,
84         title: text,
85     };
86 });
87
88     this.inform();
89 }
90
91 // 删除所有已完成 todo
92 clearCompleted() {
93     this.todos = this.todos.filter(function (todo) {
94         return !todo.completed;
95     });
96
97     this.inform();
98 }
99 }
100
101 // 订阅者类型定义
102 export type Subscriber = () => void;
103 // 单个 Todo 类型定义
104 export interface TodoItem {
105     id: string;
106     title: string;
107     completed: boolean;
108 }

```

引入 todoModel

修改 `src/App.tsx`

创建新 todo

header 包含以下业务逻辑

- 编辑新 todo: `handleChange`
- 快捷键 Enter 创建新 todo: `handleNewTodoKeyDown`

修改 `src/components/Header.tsx`

```

1 import React, { ChangeEvent, KeyboardEvent } from 'react';
2
3 interface Props {
4   newTodo: string;
5   handleNewTodoKeyDown: (event: KeyboardEvent<HTMLInputElement>) => void;
6   handleNewTodoChange: (event: ChangeEvent<HTMLInputElement>) => void;
7 }
8
9 export function Header(props: Props) {
10   const { newTodo, handleNewTodoChange, handleNewTodoKeyDown } = props;
11
12   return (
13     <header className="header">
14       <h1>todos</h1>
15       <input
16         className="new-todo"
17         placeholder="What needs to be done?"
18         value={newTodo}
19         onKeyDown={handleNewTodoKeyDown}
20         onChange={handleNewTodoChange}
21         autoFocus={true}
22       />
23     </header>
24   );
25 }

```

修改 `src/App.tsx`

```

1 import React, { ChangeEvent, KeyboardEvent, useState } from 'react';
2 import { Header } from './components/Header';
3 import { Main } from './components/Main';
4 import { Footer } from './components/Footer';
5 import { TodoModel } from './models/todoModel';
6 import './styles/index.css';
7 import './styles/base.css';
8
9 const KeyCodes = {
10   Enter: 'Enter',
11   Escape: 'Escape',
12 };
13
14 const model = new TodoModel('TodoMVC');
15
16 function App() {

```

```

17  const [newTodo, setNewTodo] = useState('');
18
19  const handleNewTodoKeyDown = (event: KeyboardEvent<HTMLInputElement>) => {
20    if (event.code !== KeyCodes.Enter) {
21      return;
22    }
23
24    event.preventDefault();
25
26    const val = newTodo.trim();
27
28    if (!val) {
29      return;
30    }
31
32    model.addToDo(val);
33    setNewTodo(val);
34  };
35
36  const handleNewTodoChange = (event: ChangeEvent<HTMLInputElement>) => {
37    setNewTodo(event.target.value);
38  };
39
40  return (
41    <section className="todoapp">
42      <Header
43        newTodo={newTodo}
44        handleNewTodoChange={handleNewTodoChange}
45        handleNewTodoKeyDown={handleNewTodoKeyDown}
46      />
47      <Main />
48      <Footer />
49    </section>
50  );
51 }
52
53 export default App;

```

todo-list 业务逻辑

之前我们已经像素级还原了 `todo-list`，接下来需要基于真实数据来渲染

分析得出，每一个 `todo` 都有相同的功能，它们应作为组件复用，同时拥有相同的接口

对于组件外部，接口包括

- `todo`: 元数据，包含 `todo` 的内容，完成状态和 `id`
- `onToggle`: 切换 `todo` 的完成状态的回调函数
- `onEdit`: 进入编辑态时触发，修改顶层维护的编辑对象（用于表示处于编辑态的 `todo`）的回调函数
- `onSave`: 保存创建新 `todo` 时的回调函数
- `onCancel`: 取消 `todo` 编辑态时的回调函数
- `onDestroy`: 删除 `todo` 时的回调函数

对于组件内部，接口包括

- `editText`: 临时保存的编辑态内容，初始值为 `todo.title`
- `handleEdit`: 进入编辑态
- `handleSubmit`: 创建新 `todo`
- `handleChange`: 修改编辑态内容
- `handleKeyDown`: 处理快捷键逻辑

首先，将之前像素级还原的相同结构的 `todo` 抽成一个独立的组件

创建文件 `src/components/ToDo.tsx`

```
1 import React, { ChangeEvent, useState, KeyboardEvent } from 'react';
2 import cx from 'classnames';
3 import { TodoItem } from '../models/todoModel';
4
5 interface Props {
6   todo: TodoItem;
7   editing: boolean;
8   onToggle: () => void;
9   onEdit: () => void;
10  onSave: (v: string) => void;
11  onCancel: () => void;
12  onDestroy: () => void;
13 }
14
15 const KeyCodes = {
```

```
16   Enter: 'Enter',
17   Escape: 'Escape',
18 };
19
20 export function Todo(props: Props) {
21   const { todo, editing, onToggle, onEdit, onSave, onCancel, onDestroy } =
22     props;
23
24   const [editText, setEditText] = useState(todo.title);
25
26   const handleChange = (event: ChangeEvent<HTMLInputElement>) => {
27     setEditText(event.target.value);
28   };
29
30   const handleEdit = () => {
31     onEdit();
32     setEditText(todo.title);
33   };
34
35   const handleSubmit = () => {
36     const val = editText.trim();
37
38     if (!val) {
39       onDestroy();
40       return;
41     }
42
43     onSave(val);
44     setEditText(val);
45   };
46
47   const handleKeyDown = (event: KeyboardEvent<HTMLInputElement>) => {
48     if (event.code === KeyCode.Enter) {
49       handleSubmit();
50     } else if (event.code === KeyCode.Escape) {
51       onCancel();
52       setEditText(todo.title);
53     }
54   };
55
56   return (
57     <li
58       className={cx({
59         editing,
60         completed: todo.completed,
61       })}>
62     <div className="view">
```

```

63     <input
64         className="toggle"
65         type="checkbox"
66         checked={todo.completed}
67         onChange={onToggle}
68     />
69     <label onDoubleClick={handleEdit}>{todo.title}</label>
70     <button className="destroy" onClick={onDestroy} type="button"></button>
71 </div>
72 <input
73     className="edit"
74     value={editText}
75     onChange={handleChange}
76     onBlur={handleSubmit}
77     onKeyDown={handleKeyDown}
78 />
79 </li>
80 );
81 }

```

修改 src/components/Main.tsx

```

1  import React from 'react';
2  import { TodoItem } from '../models/todoModel';
3  import { Todo } from './Todo';
4
5  interface Props {
6      todos: Array<TodoItem>;
7      editing: string;
8      onToggle: (todo: TodoItem) => void;
9      onEdit: (todo: TodoItem) => void;
10     onSave: (v: TodoItem, val: string) => void;
11     onCancel: () => void;
12     onDestroy: (todo: TodoItem) => void;
13 }
14
15 export function Main(props: Props) {
16     const { todos, editing, onToggle, onEdit, onSave, onCancel, onDestroy } =
17         props;
18
19     return (
20         <section className="main">
21             <input id="toggle-all" className="toggle-all" type="checkbox" />
22             <label htmlFor="toggle-all"></label>
23             <ul className="todo-list">

```

```

24     {todos.map(todo => (
25         <Todo
26             key={todo.id}
27             todo={todo}
28             editing={editing === todo.id}
29             onCancel={onCancel}
30             onDestroy={() => onDestroy(todo)}
31             onEdit={() => onEdit(todo)}
32             onSave={(v: string) => onSave(todo, v)}
33             onToggle={() => onToggle(todo)}
34         />
35     ))}
36 </ul>
37 </section>
38 );
39 }

```

修改 `src/App.tsx`

```

1  import React, { ChangeEvent, KeyboardEvent, useState } from 'react';
2  import { Header } from './components/Header';
3  import { Main } from './components/Main';
4  import { Footer } from './components/Footer';
5  import { TodoModel, TodoItem } from './models/todoModel';
6  import './styles/index.css';
7  import './styles/base.css';
8
9  const KeyCodes = {
10     Enter: 'Enter',
11     Escape: 'Escape',
12 };
13
14 const model = new TodoModel('TodoMVC');
15
16 function App() {
17     const [newTodo, setNewTodo] = useState('');
18     const [editing, setEditing] = useState('');
19
20     const handleNewTodoKeyDown = (event: KeyboardEvent<HTMLInputElement>) => {
21         if (event.code !== KeyCodes.Enter) {
22             return;
23         }
24
25         event.preventDefault();
26

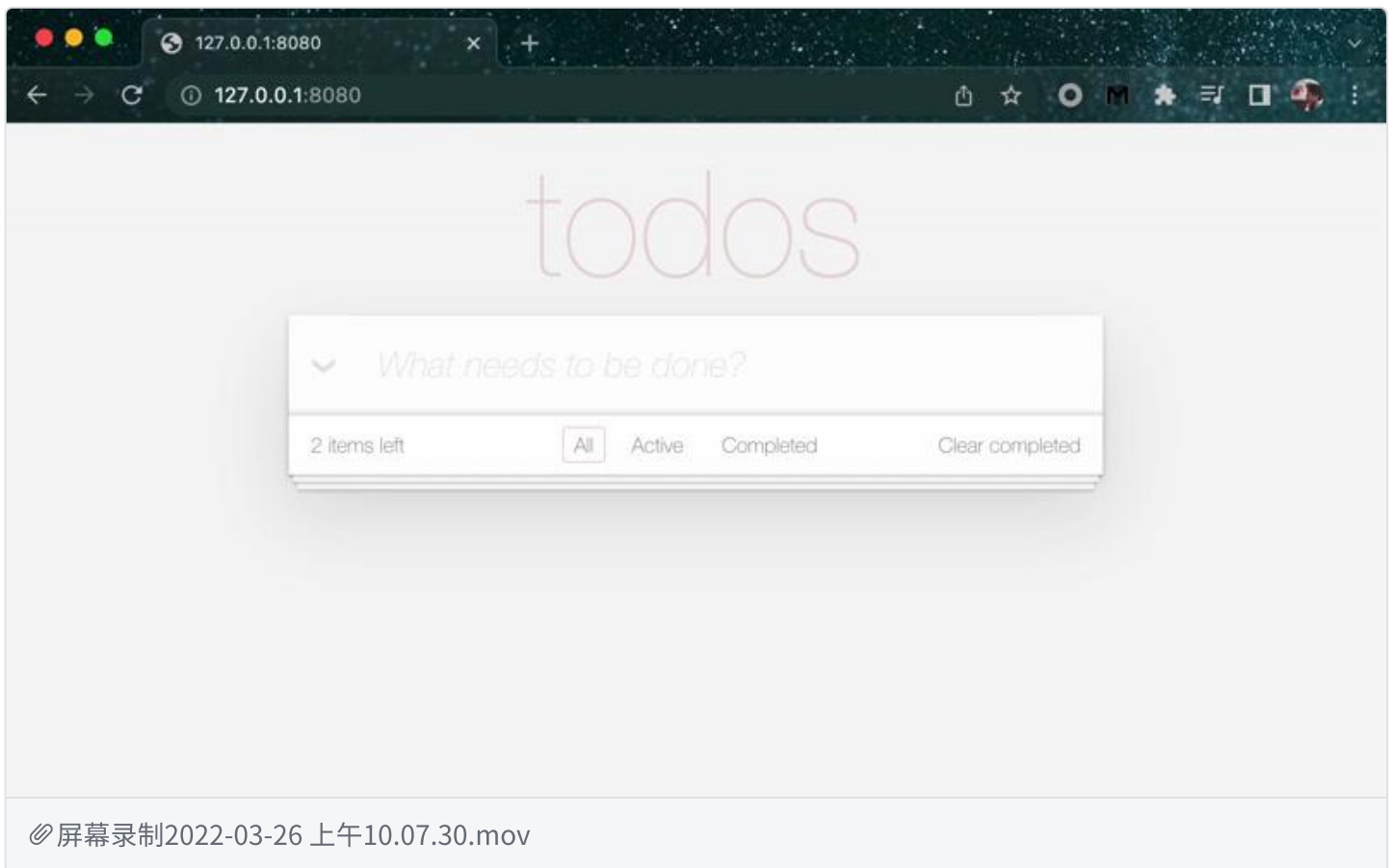
```



```
27     const val = newTodo.trim();
28
29     if (!val) {
30         return;
31     }
32
33     model.addToDo(val);
34     setNewTodo(val);
35 };
36
37 const handleNewTodoChange = (event: ChangeEvent<HTMLInputElement>) => {
38     setNewTodo(event.target.value);
39 };
40
41 const onToggle = (todo: TodoItem) => {
42     model.toggle(todo);
43 };
44
45 const onDestroy = (todo: TodoItem) => {
46     model.destroy(todo);
47 };
48
49 const onEdit = (todo: TodoItem) => {
50     setEditing(todo.id);
51 };
52
53 const onSave = (todo: TodoItem, val: string) => {
54     model.save(todo, val);
55     setEditing('');
56 };
57
58 const onCancel = () => {
59     setEditing('');
60 };
61
62 return (
63     <section className="todoapp">
64         <Header
65             newTodo={newTodo}
66             handleNewTodoChange={handleNewTodoChange}
67             handleNewTodoKeyDown={handleNewTodoKeyDown}
68         />
69         <Main
70             editing={editing}
71             todos={model.todos}
72             onToggle={onToggle}
73             onDestroy={onDestroy}
```

```
74     onEdit={onEdit}
75     onSave={onSave}
76     onCancel={onCancel}
77   />
78   <Footer />
79 </section>
80 );
81 }
82
83 export default App;
```

预览效果



可以看到，这时我们可以创建新的 `todo`，但是切换 `todo` 完成状态和删除 `todo` 都没有直接生效，需要刷新或者进入编辑态才能拿到正确的视图。

React 函数组件只有当内部的 `state` 或者传入的 `props` 发生变更时才会重渲染，重新走一遍函数内的逻辑。

实际上，即使是创建 `todo` 也不应响应到视图上。这是因为 `model` 是从 React 组件外传入，一次性执行。对于 `Main` 组件来说，上层组件传给它的 `todos` 已经固定，不会触发视图的更新。只有当页面刷新，从缓存中重新获取数据后才渲染了正确的视图。

这里我们切换 `todo` 完成状态和删除 `todo` 没有直接生效是符合预期的。

但是创建 `todo` 或进入编辑态时，我们更新了 `src/App.tsx` 中的状态，导致组件重渲染，将此时新的状态重新传递到了下层组件。

监听 model 更新

`todoModel` 中，我们简单实现了「观察者模式」，每当 `todos` 修改时，会调用 `inform()` 去通知所有通过 `model.subscribe` 订阅 `model` 事件的观察者。

注意：在订阅中直接重新更新整个应用不是常规操作，实际应用中可以尝试自己封装 hook 或者引入 `redux`, `mobx`, `xstate` 等状态管理库，利用第三方库封装好的接口进行状态同步和视图更新。

修改 `src/App.tsx` ,引入 `forceUpdate` 刷新组件

```
1 import React, { ChangeEvent, KeyboardEvent, useEffect, useState } from 'react';
2 import { Header } from './components/Header';
3 import { Main } from './components/Main';
4 import { Footer } from './components/Footer';
5 import { TodoModel, TodoItem } from './models/todoModel';
6 import './styles/index.css';
7 import './styles/base.css';
8
9 const KeyCodes = {
10   Enter: 'Enter',
11   Escape: 'Escape',
12 };
13
14 const model = new TodoModel('TodoMVC');
15
16 function App() {
17   const [newTodo, setNewTodo] = useState('');
18   const [editing, setEditing] = useState('');
19   const [, forceUpdate] = useState(0);
20
21   useEffect(() => {
22     model.subscribe(() => forceUpdate(prev => prev + 1));
23   }, []);
24
25   const handleNewTodoKeyDown = (event: KeyboardEvent<HTMLInputElement>) => {
26     if (event.code !== KeyCodes.Enter) {
```

```
27     return;
28 }
29
30 event.preventDefault();
31
32 const val = newTodo.trim();
33
34 if (!val) {
35     return;
36 }
37
38 model.addToDo(val);
39 setNewTodo('');
40 };
41
42 const handleNewTodoChange = (event: ChangeEvent<HTMLInputElement>) => {
43     setNewTodo(event.target.value);
44 };
45
46 const onToggle = (todo: TodoItem) => {
47     model.toggle(todo);
48 };
49
50 const onDestroy = (todo: TodoItem) => {
51     model.destroy(todo);
52 };
53
54 const onEdit = (todo: TodoItem) => {
55     setEditing(todo.id);
56 };
57
58 const onSave = (todo: TodoItem, val: string) => {
59     model.save(todo, val);
60     setEditing('');
61 };
62
63 const onCancel = () => {
64     setEditing('');
65 };
66
67 return (
68     <section className="todoapp">
69         <Header
70             newTodo={newTodo}
71             handleNewTodoChange={handleNewTodoChange}
72             handleNewTodoKeyDown={handleNewTodoKeyDown}
73         />
```

```

74     <Main
75         editing={editing}
76         todos={model.todos}
77         onToggle={onToggle}
78         onDestroy={onDestroy}
79         onEdit={onEdit}
80         onSave={onSave}
81         onCancel={onCancel}
82     />
83     <Footer />
84 </section>
85 );
86 }
87
88 export default App;

```

如此，视图将按预期更新。

Toggle All

修改 `src/components/Main.tsx`，新增 `onToggleAll` props，计算当前未完成 todo 数量

```

1  import React from 'react';
2  import { TodoItem } from '../models/todoModel';
3  import { Todo } from './Todo';
4
5  interface Props {
6      todos: Array<TodoItem>;
7      editing: string;
8      onToggle: (todo: TodoItem) => void;
9      onToggleAll: (checked: boolean) => void;
10     onEdit: (todo: TodoItem) => void;
11     onSave: (v: TodoItem, val: string) => void;
12     onCancel: () => void;
13     onDestroy: (todo: TodoItem) => void;
14 }
15
16 export function Main(props: Props) {
17     const {
18         todos,
19         editing,
20         onToggle,
21         onToggleAll,
22         onEdit,

```

```

23     onSave,
24     onCancel,
25     onDestroy,
26   } = props;
27
28   const activeTodoCount = todos.reduce(function (accum, todo) {
29     return todo.completed ? accum : accum + 1;
30   }, 0);
31
32   return (
33     <section className="main">
34       <input
35         id="toggle-all"
36         className="toggle-all"
37         type="checkbox"
38         checked={activeTodoCount === 0}
39         onChange={event => onToggleAll((event.target as any).checked)}
40       />
41       <label htmlFor="toggle-all"></label>
42       <ul className="todo-list">
43         {todos.map(todo => (
44           <Todo
45             key={todo.id}
46             todo={todo}
47             editing={editing === todo.id}
48             onCancel={onCancel}
49             onDestroy={() => onDestroy(todo)}
50             onEdit={() => onEdit(todo)}
51             onSave={(v: string) => onSave(todo, v)}
52             onToggle={() => onToggle(todo)}
53           />
54         ))}
55       </ul>
56     </section>
57   );
58 }

```

修改 `src/App.tsx`

```

1 import React, { ChangeEvent, KeyboardEvent, useEffect, useState } from 'react';
2 import { Header } from './components/Header';
3 import { Main } from './components/Main';
4 import { Footer } from './components/Footer';
5 import { TodoModel, TodoItem } from './models/todoModel';
6 import './styles/index.css';

```

```
7 import './styles/base.css';
8
9 const KeyCodes = {
10   Enter: 'Enter',
11   Escape: 'Escape',
12 };
13
14 const model = new TodoModel('TodoMVC');
15
16 function App() {
17   const [newTodo, setNewTodo] = useState('');
18   const [editing, setEditing] = useState('');
19   const [, forceUpdate] = useState(0);
20
21   useEffect(() => {
22     model.subscribe(() => forceUpdate(prev => prev + 1));
23   }, []);
24
25   const handleNewTodoKeyDown = (event: KeyboardEvent<HTMLInputElement>) => {
26     if (event.code !== KeyCodes.Enter) {
27       return;
28     }
29
30     event.preventDefault();
31
32     const val = newTodo.trim();
33
34     if (!val) {
35       return;
36     }
37
38     model.addToDo(val);
39     setNewTodo('');
40   };
41
42   const handleNewTodoChange = (event: ChangeEvent<HTMLInputElement>) => {
43     setNewTodo(event.target.value);
44   };
45
46   const onToggle = (todo: TodoItem) => {
47     model.toggle(todo);
48   };
49
50   const onToggleAll = (checked: boolean) => {
51     model.toggleAll(checked);
52   };
53
```



```

54  const onDestroy = (todo: TodoItem) => {
55      model.destroy(todo);
56  };
57
58  const onEdit = (todo: TodoItem) => {
59      setEditing(todo.id);
60  };
61
62  const onSave = (todo: TodoItem, val: string) => {
63      model.save(todo, val);
64      setEditing('');
65  };
66
67  const onCancel = () => {
68      setEditing('');
69  };
70
71  return (
72      <section className="todoapp">
73          <Header
74              newTodo={newTodo}
75              handleNewTodoChange={handleNewTodoChange}
76              handleNewTodoKeyDown={handleNewTodoKeyDown}
77          />
78          <Main
79              editing={editing}
80              todos={model.todos}
81              onToggle={onToggle}
82              onToggleAll={onToggleAll}
83              onDestroy={onDestroy}
84              onEdit={onEdit}
85              onSave={onSave}
86              onCancel={onCancel}
87          />
88          <Footer />
89      </section>
90  );
91  }
92
93  export default App;

```

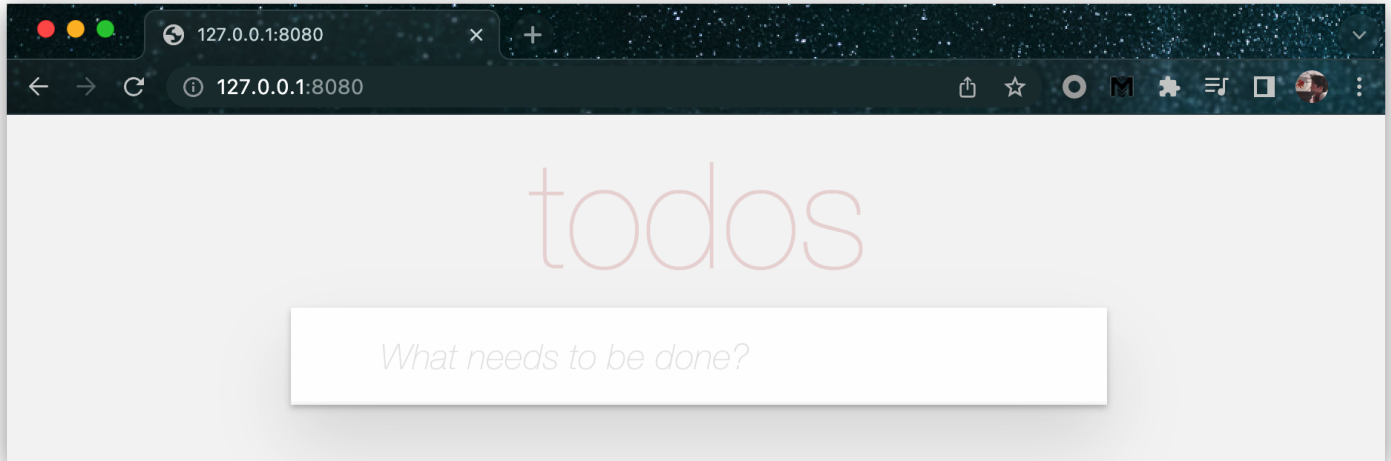
存在 todos 数据时才展示 Main 和 Footer

修改 `src/App.tsx`

```
1 import React, { ChangeEvent, KeyboardEvent, useEffect, useState } from 'react';
2 import { Header } from './components/Header';
3 import { Main } from './components/Main';
4 import { Footer } from './components/Footer';
5 import { TodoModel, TodoItem } from './models/todoModel';
6 import './styles/index.css';
7 import './styles/base.css';
8
9 const KeyCodes = {
10   Enter: 'Enter',
11   Escape: 'Escape',
12 };
13
14 const model = new TodoModel('TodoMVC');
15
16 function App() {
17   const [newTodo, setNewTodo] = useState('');
18   const [editing, setEditing] = useState('');
19   const [, forceUpdate] = useState(0);
20
21   useEffect(() => {
22     model.subscribe(() => forceUpdate(prev => prev + 1));
23   }, []);
24
25   const handleNewTodoKeyDown = (event: KeyboardEvent<HTMLInputElement>) => {
26     if (event.code !== KeyCodes.Enter) {
27       return;
28     }
29
30     event.preventDefault();
31
32     const val = newTodo.trim();
33
34     if (!val) {
35       return;
36     }
37
38     model.addToDo(val);
39     setNewTodo('');
40   };
41
42   const handleNewTodoChange = (event: ChangeEvent<HTMLInputElement>) => {
43     setNewTodo(event.target.value);
44   };
45
46   const onToggle = (todo: TodoItem) => {
47     model.toggle(todo);
```

```
48 };
49
50 const onToggleAll = (checked: boolean) => {
51     model.toggleAll(checked);
52 };
53
54 const onDestroy = (todo: TodoItem) => {
55     model.destroy(todo);
56 };
57
58 const onEdit = (todo: TodoItem) => {
59     setEditing(todo.id);
60 };
61
62 const onSave = (todo: TodoItem, val: string) => {
63     model.save(todo, val);
64     setEditing('');
65 };
66
67 const onCancel = () => {
68     setEditing('');
69 };
70
71 return (
72     <section className="todoapp">
73         <Header
74             newTodo={newTodo}
75             handleNewTodoChange={handleNewTodoChange}
76             handleNewTodoKeyDown={handleNewTodoKeyDown}
77         />
78         {Boolean(model.todos.length) && (
79             <>
80                 <Main
81                     editing={editing}
82                     todos={model.todos}
83                     onToggle={onToggle}
84                     onToggleAll={onToggleAll}
85                     onDestroy={onDestroy}
86                     onEdit={onEdit}
87                     onSave={onSave}
88                     onCancel={onCancel}
89                 />
90                 <Footer />
91             </>
92         )}
93     </section>
94 );
```

```
95 }  
96  
97 export default App;
```



清空已完成 todo

修改 `src/components/Footer.tsx`

```
1  import React from 'react';  
2  
3  interface Props {  
4    hasCompletedTodos: boolean;  
5    onClearCompleted: () => void;  
6  }  
7  
8  export function Footer(props: Props) {  
9    const { hasCompletedTodos, onClearCompleted } = props;  
10  
11    return (  
12      <footer className="footer">  
13        <span className="todo-count">  
14          <strong>2</strong>  
15          <span> </span>  
16          <span>items</span>  
17          <span> left</span>  
18        </span>  
19        <ul className="filters">  
20          <li>  
21            <a href="#/" className="selected">  
22              All  
23            </a>  
24          </li>  
25          <span> </span>
```

```

26     <li>
27       <a href="#/active">Active</a>
28     </li>
29     <span> </span>
30     <li>
31       <a href="#/completed">Completed</a>
32     </li>
33   </ul>
34   {hasCompletedTodos && (
35     <button
36       className="clear-completed"
37       onClick={onClearCompleted}
38       type="button">
39       Clear completed
40     </button>
41   )}
42 </footer>
43 );
44 }

```

修改 `src/App.tsx`

```

1  import React, { ChangeEvent, KeyboardEvent, useEffect, useState } from 'react';
2  import { Header } from './components/Header';
3  import { Main } from './components/Main';
4  import { Footer } from './components/Footer';
5  import { TodoModel, TodoItem } from './models/todoModel';
6  import './styles/index.css';
7  import './styles/base.css';
8
9  const KeyCodes = {
10    Enter: 'Enter',
11    Escape: 'Escape',
12  };
13
14  const model = new TodoModel('TodoMVC');
15
16  function App() {
17    const [newTodo, setNewTodo] = useState('');
18    const [editing, setEditing] = useState('');
19    const [, forceUpdate] = useState(0);
20
21    const activeTodoCount = model.todos.reduce(function (accum, todo) {
22      return todo.completed ? accum : accum + 1;
23    }, 0);

```

```
24
25  const completedTodoCount = model.todos.length - activeTodoCount;
26
27  useEffect(() => {
28    model.subscribe(() => forceUpdate(prev => prev + 1));
29  }, []);
30
31  const handleNewTodoKeyDown = (event: KeyboardEvent<HTMLInputElement>) => {
32    if (event.code !== KeyCodes.Enter) {
33      return;
34    }
35
36    event.preventDefault();
37
38    const val = newTodo.trim();
39
40    if (!val) {
41      return;
42    }
43
44    model.addTodo(val);
45    setNewTodo('');
46  };
47
48  const handleNewTodoChange = (event: ChangeEvent<HTMLInputElement>) => {
49    setNewTodo(event.target.value);
50  };
51
52  const onToggle = (todo: TodoItem) => {
53    model.toggle(todo);
54  };
55
56  const onToggleAll = (checked: boolean) => {
57    model.toggleAll(checked);
58  };
59
60  const onDestroy = (todo: TodoItem) => {
61    model.destroy(todo);
62  };
63
64  const onEdit = (todo: TodoItem) => {
65    setEditing(todo.id);
66  };
67
68  const onSave = (todo: TodoItem, val: string) => {
69    model.save(todo, val);
70    setEditing('');
```

```

71   };
72
73   const onCancel = () => {
74     setEditing('');
75   };
76
77   const onClearCompleted = () => {
78     model.clearCompleted();
79   };
80
81   return (
82     <section className="todoapp">
83       <Header
84         newTodo={newTodo}
85         handleNewTodoChange={handleNewTodoChange}
86         handleNewTodoKeyDown={handleNewTodoKeyDown}
87       />
88       {Boolean(model.todos.length) && (
89         <>
90           <Main
91             editing={editing}
92             todos={model.todos}
93             onToggle={onToggle}
94             onToggleAll={onToggleAll}
95             onDestroy={onDestroy}
96             onEdit={onEdit}
97             onSave={onSave}
98             onCancel={onCancel}
99           />
100          <Footer
101            onClearCompleted={onClearCompleted}
102            hasCompletedTodos={completedTodoCount !== 0}
103          />
104        </>
105      )}
106    </section>
107  );
108 }
109
110 export default App;

```

路由

修改 `src/App.tsx`，新增路由监听，todos 过滤


```
1 import React, { ChangeEvent, KeyboardEvent, useEffect, useState } from 'react';
2 import { useLocation } from '@modern-js/runtime/router';
3 import { Header } from './components/Header';
4 import { Main } from './components/Main';
5 import { Footer, NowShowing } from './components/Footer';
6 import { TodoModel, TodoItem } from './models/todoModel';
7 import './styles/index.css';
8 import './styles/base.css';
9
10 const KeyCodes = {
11   Enter: 'Enter',
12   Escape: 'Escape',
13 };
14
15 const model = new TodoModel('TodoMVC');
16
17 // eslint-disable-next-line max-statements
18 function App() {
19   const [newTodo, setNewTodo] = useState('');
20   const [editing, setEditing] = useState('');
21   const [nowShowing, setNowShowing] = useState<NowShowing>('all');
22   const [, forceUpdate] = useState(0);
23
24   const { hash } = useLocation();
25
26   const activeTodoCount = model.todos.reduce(function (accum, todo) {
27     return todo.completed ? accum : accum + 1;
28   }, 0);
29
30   const completedTodoCount = model.todos.length - activeTodoCount;
31
32   let showingTodos = model.todos;
33
34   if (nowShowing === 'active') {
35     showingTodos = model.todos.filter(todo => !todo.completed);
36   }
37
38   if (nowShowing === 'completed') {
39     showingTodos = model.todos.filter(todo => todo.completed);
40   }
41
42   useEffect(() => {
43     model.subscribe(() => forceUpdate(prev => prev + 1));
44   }, []);
45
46   useEffect(() => {
47     if (hash.includes('active')) {
```

```
48     setNowShowing('active');
49   } else if (hash.includes('completed')) {
50     setNowShowing('completed');
51   } else {
52     setNowShowing('all');
53   }
54 }, [hash]));
55
56 const handleNewTodoKeyDown = (event: KeyboardEvent<HTMLInputElement>) => {
57   if (event.code !== KeyCodes.Enter) {
58     return;
59   }
60
61   event.preventDefault();
62
63   const val = newTodo.trim();
64
65   if (!val) {
66     return;
67   }
68
69   model.addToDo(val);
70   setNewTodo('');
71 };
72
73 const handleNewTodoChange = (event: ChangeEvent<HTMLInputElement>) => {
74   setNewTodo(event.target.value);
75 };
76
77 const onToggle = (todo: TodoItem) => {
78   model.toggle(todo);
79 };
80
81 const onToggleAll = (checked: boolean) => {
82   model.toggleAll(checked);
83 };
84
85 const onDestroy = (todo: TodoItem) => {
86   model.destroy(todo);
87 };
88
89 const onEdit = (todo: TodoItem) => {
90   setEditing(todo.id);
91 };
92
93 const onSave = (todo: TodoItem, val: string) => {
94   model.save(todo, val);
```

```
95     setEditing('');
96   };
97
98   const onCancel = () => {
99     setEditing('');
100  };
101
102  const onClearCompleted = () => {
103    model.clearCompleted();
104  };
105
106  return (
107    <section className="todoapp">
108      <Header
109        newTodo={newTodo}
110        handleNewTodoChange={handleNewTodoChange}
111        handleNewTodoKeyDown={handleNewTodoKeyDown}
112      />
113      {Boolean(model.todos.length) && (
114        <>
115          <Main
116            editing={editing}
117            todos={showingTodos}
118            onToggle={onToggle}
119            onToggleAll={onToggleAll}
120            onDestroy={onDestroy}
121            onEdit={onEdit}
122            onSave={onSave}
123            onCancel={onCancel}
124          />
125          <Footer
126            activeTodoCount={activeTodoCount}
127            nowShowing={nowShowing}
128            onClearCompleted={onClearCompleted}
129            hasCompletedTodos={completedTodoCount !== 0}
130          />
131        </>
132      )}
133    </section>
134  );
135 }
136
137 export default App;
```

```
1 import React from 'react';
2 import cx from 'classnames';
3
4 interface Props {
5   nowShowing: NowShowing;
6   hasCompletedTodos: boolean;
7   onClearCompleted: () => void;
8 }
9
10 export type NowShowing = 'all' | 'active' | 'completed';
11
12 export function Footer(props: Props) {
13   const { nowShowing, hasCompletedTodos, onClearCompleted } = props;
14
15   return (
16     <footer className="footer">
17       <span className="todo-count">
18         <strong>2</strong>
19         <span> </span>
20         <span>items</span>
21         <span> left</span>
22       </span>
23       <ul className="filters">
24         <li>
25           <a href="#/" className={cx({ selected: nowShowing === 'all' })}>
26             All
27           </a>
28         </li>
29         <span> </span>
30         <li>
31           <a
32             href="#/active"
33             className={cx({ selected: nowShowing === 'active' })}>
34             Active
35           </a>
36         </li>
37         <span> </span>
38         <li>
39           <a
40             href="#/completed"
41             className={cx({ selected: nowShowing === 'completed' })}>
42             Completed
43           </a>
44         </li>
```

```

45     </ul>
46     {hasCompletedTodos && (
47         <button
48             className="clear-completed"
49             onClick={onClearCompleted}
50             type="button">
51             Clear completed
52         </button>
53     )}
54 </footer>
55 );
56 }

```

footer 复数

修改 `src/utils.ts`，新增复数判断函数

```

1  import { nanoid } from 'nanoid';
2  import { TodoItem } from './models/todoModel';
3
4  // 生成唯一 id
5  export function generateId() {
6      return nanoid();
7  }
8  // 从 localStorage 获取或向 localStorage 同步数据
9  export function store(namespace: string, data?: Array<TodoItem>) {
10     if (data) {
11         return localStorage.setItem(namespace, JSON.stringify(data));
12     }
13
14     const localStore = localStorage.getItem(namespace);
15     return (localStore && JSON.parse(localStore)) || [];
16 }
17
18 export function pluralize(count: number, word: string) {
19     return count === 1 ? word : `${word}s`;
20 }

```

修改 `src/App.tsx`，为 Footer 组件传入 activeTodoCount

```

1  import React, { ChangeEvent, KeyboardEvent, useEffect, useState } from 'react';
2  import { useLocation } from '@modern-js/runtime/router';

```

```
3 import { Header } from './components/Header';
4 import { Main } from './components/Main';
5 import { Footer, NowShowing } from './components/Footer';
6 import { TodoModel, TodoItem } from './models/todoModel';
7 import './styles/index.css';
8 import './styles/base.css';
9
10 const KeyCodes = {
11   Enter: 'Enter',
12   Escape: 'Escape',
13 };
14
15 const model = new TodoModel('TodoMVC');
16
17 function App() {
18   const [newTodo, setNewTodo] = useState('');
19   const [editing, setEditing] = useState('');
20   const [nowShowing, setNowShowing] = useState<NowShowing>('all');
21   const [, forceUpdate] = useState(0);
22
23   const { hash } = useLocation();
24
25   const activeTodoCount = model.todos.reduce(function (accum, todo) {
26     return todo.completed ? accum : accum + 1;
27   }, 0);
28
29   const completedTodoCount = model.todos.length - activeTodoCount;
30
31   useEffect(() => {
32     model.subscribe(() => forceUpdate(prev => prev + 1));
33   }, []);
34
35   useEffect(() => {
36     if (hash.includes('active')) {
37       setNowShowing('active');
38     } else if (hash.includes('completed')) {
39       setNowShowing('completed');
40     } else {
41       setNowShowing('all');
42     }
43   }, [hash]);
44
45   const handleNewTodoKeyDown = (event: KeyboardEvent<HTMLInputElement>) => {
46     if (event.code !== KeyCodes.Enter) {
47       return;
48     }
49   }
```

```
50     event.preventDefault();
51
52     const val = newTodo.trim();
53
54     if (!val) {
55         return;
56     }
57
58     model.addToDo(val);
59     setNewTodo('');
60 };
61
62 const handleNewTodoChange = (event: ChangeEvent<HTMLInputElement>) => {
63     setNewTodo(event.target.value);
64 };
65
66 const onToggle = (todo: TodoItem) => {
67     model.toggle(todo);
68 };
69
70 const onToggleAll = (checked: boolean) => {
71     model.toggleAll(checked);
72 };
73
74 const onDestroy = (todo: TodoItem) => {
75     model.destroy(todo);
76 };
77
78 const onEdit = (todo: TodoItem) => {
79     setEditing(todo.id);
80 };
81
82 const onSave = (todo: TodoItem, val: string) => {
83     model.save(todo, val);
84     setEditing('');
85 };
86
87 const onCancel = () => {
88     setEditing('');
89 };
90
91 const onClearCompleted = () => {
92     model.clearCompleted();
93 };
94
95 return (
96     <section className="todoapp">
```

```

97     <Header
98         newTodo={newTodo}
99         handleNewTodoChange={handleNewTodoChange}
100         handleNewTodoKeyDown={handleNewTodoKeyDown}
101     />
102     {Boolean(model.todos.length) && (
103         <>
104             <Main
105                 editing={editing}
106                 todos={model.todos}
107                 onToggle={onToggle}
108                 onToggleAll={onToggleAll}
109                 onDestroy={onDestroy}
110                 onEdit={onEdit}
111                 onSave={onSave}
112                 onCancel={onCancel}
113             />
114             <Footer
115                 activeTodoCount={activeTodoCount}
116                 nowShowing={nowShowing}
117                 onClearCompleted={onClearCompleted}
118                 hasCompletedTodos={completedTodoCount !== 0}
119             />
120         </>
121     )}
122 </section>
123 );
124 }
125
126 export default App;

```

修改 `src/components/Footer.tsx`，根据未完成 todos 展示数量及单位

```

1  import React from 'react';
2  import cx from 'classnames';
3  import { pluralize } from '@/utils';
4
5  interface Props {
6      activeTodoCount: number;
7      nowShowing: NowShowing;
8      hasCompletedTodos: boolean;
9      onClearCompleted: () => void;
10 }
11
12 export type NowShowing = 'all' | 'active' | 'completed';

```



```
13
14 export function Footer(props: Props) {
15   const { activeTodoCount, nowShowing, hasCompletedTodos, onClearCompleted } =
16     props;
17
18   return (
19     <footer className="footer">
20       <span className="todo-count">
21         <strong>{activeTodoCount}</strong>
22         <span> </span>
23         <span>{pluralize(activeTodoCount, 'item')}</span>
24         <span> left</span>
25       </span>
26       <ul className="filters">
27         <li>
28           <a href="#" className={cx({ selected: nowShowing === 'all' })}>
29             All
30           </a>
31         </li>
32         <span> </span>
33         <li>
34           <a
35             href="#/active"
36             className={cx({ selected: nowShowing === 'active' })}>
37             Active
38           </a>
39         </li>
40         <span> </span>
41         <li>
42           <a
43             href="#/completed"
44             className={cx({ selected: nowShowing === 'completed' })}>
45             Completed
46           </a>
47         </li>
48       </ul>
49       {hasCompletedTodos && (
50         <button
51           className="clear-completed"
52           onClick={onClearCompleted}
53           type="button">
54             Clear completed
55         </button>
56       )}
57     </footer>
58   );
59 }
```

上传 GitHub

```
1 git commit -am "feat: Modern.js Todo MVC"
2 git push
```