脚本程序设计

TodoMVC

学号: 1750226

姓名: 陆昱珉

指导老师: 徐凯

一、项目简介

Mytodo 是一个移动端日程管理网页。在Mytodo中，用户可以输入、查看、编辑管理日程，并且所添加的日程会被自动保存以便下次查看

本次项目已完成部署，用户可访问https://matrixtzuzt.github.io/TodoList/来体验本项目

二、开发环境

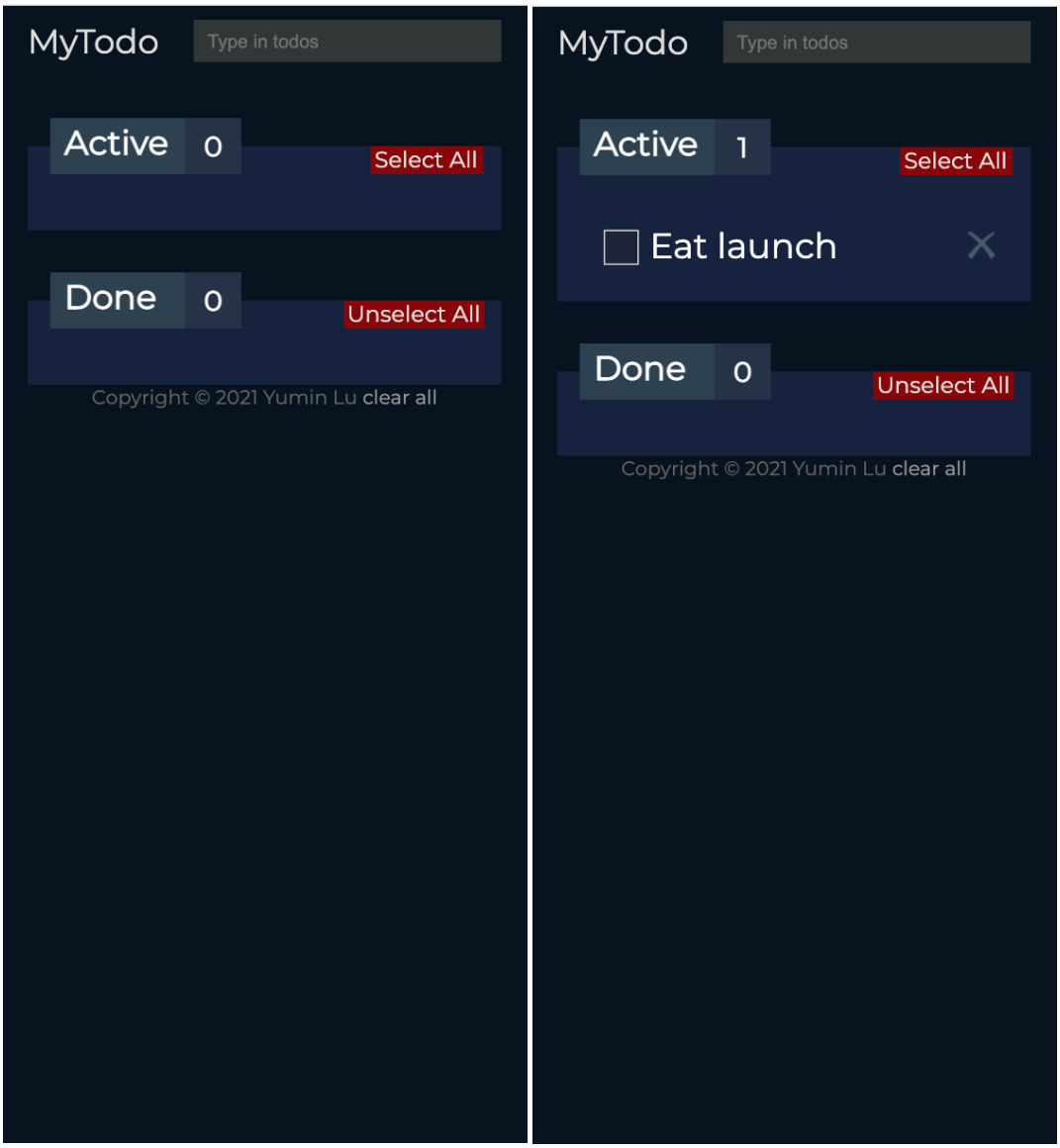
- 开发语言：html，javascript，css
- 开发工具：Visual Studio Code， Chrome

三、功能说明

基础功能

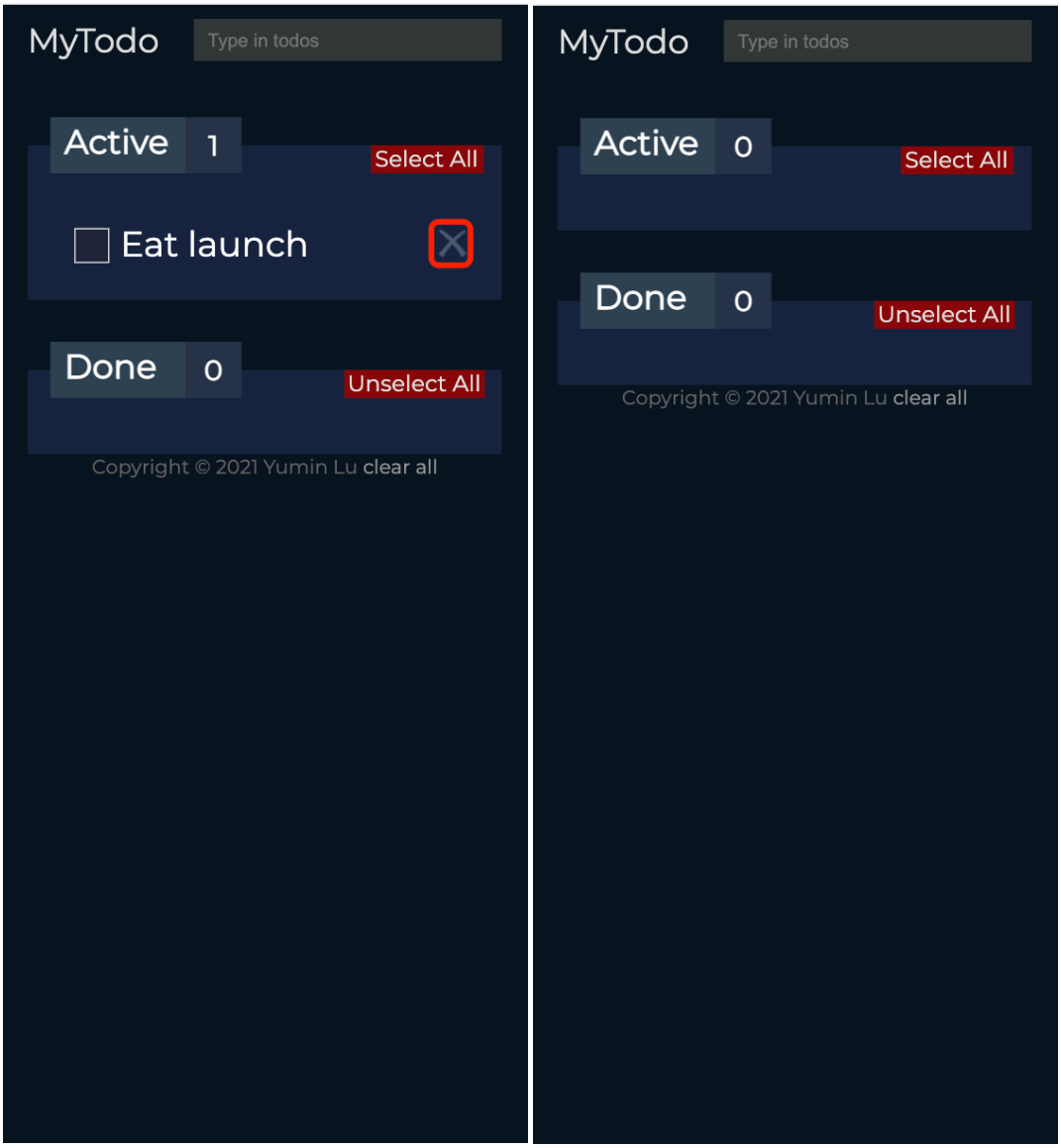
- 新增Todo项

  在顶部输入框输入相应内容并按下**Enter/完成**按键即可添加一条Todo项



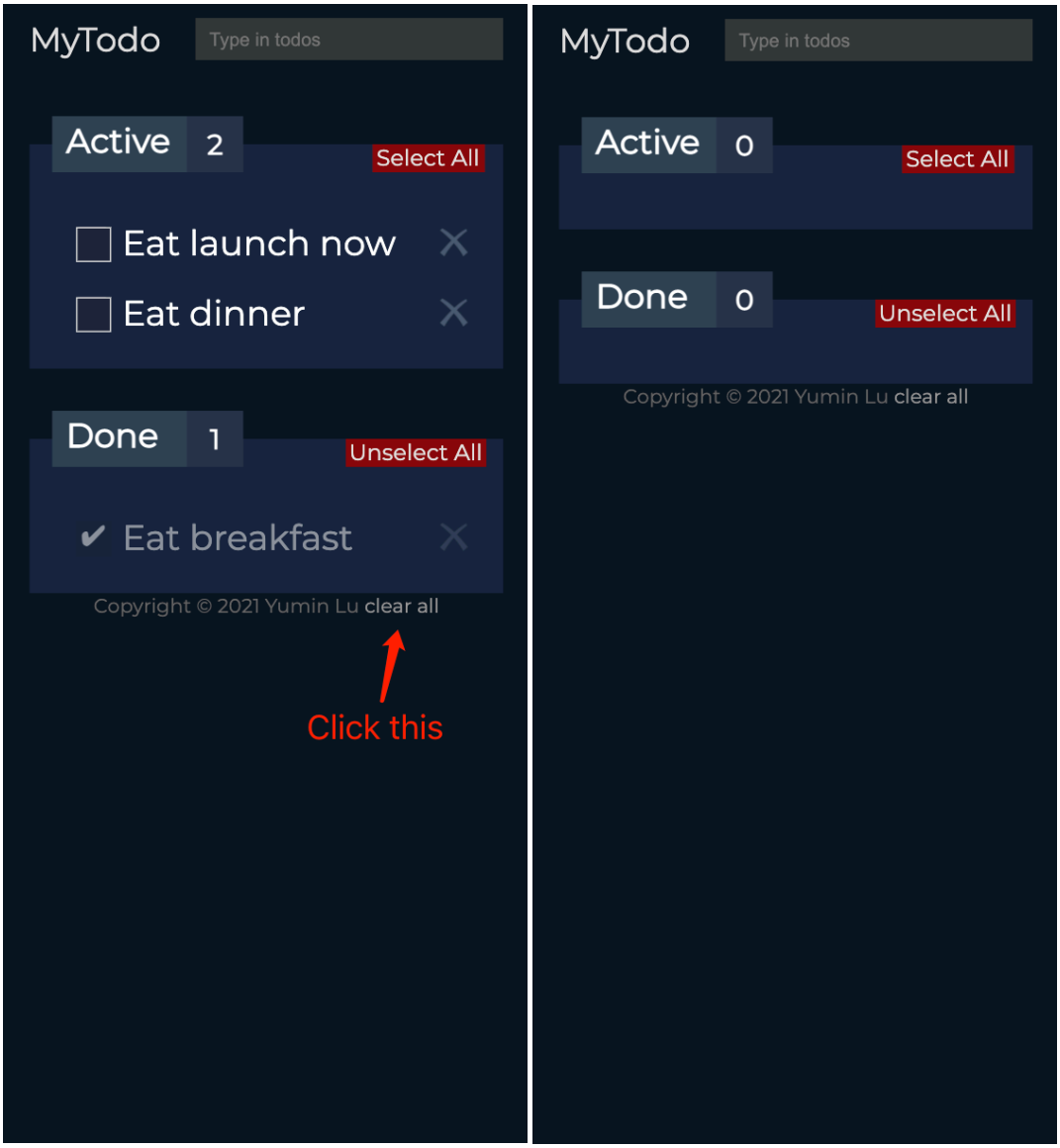
- 删除单条Todo项

  点击相应Todo项右侧的“❌”图标即可删除相应Todo项



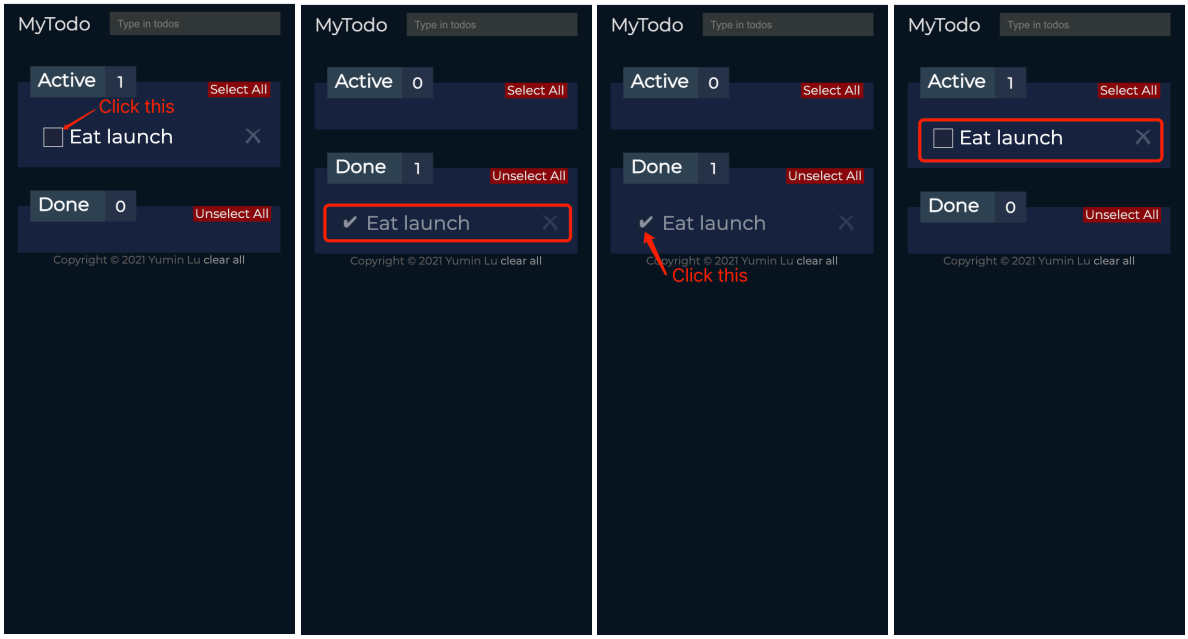
- 删除所有Todo项

  点击底部**clear all**即可删除所有Todo项



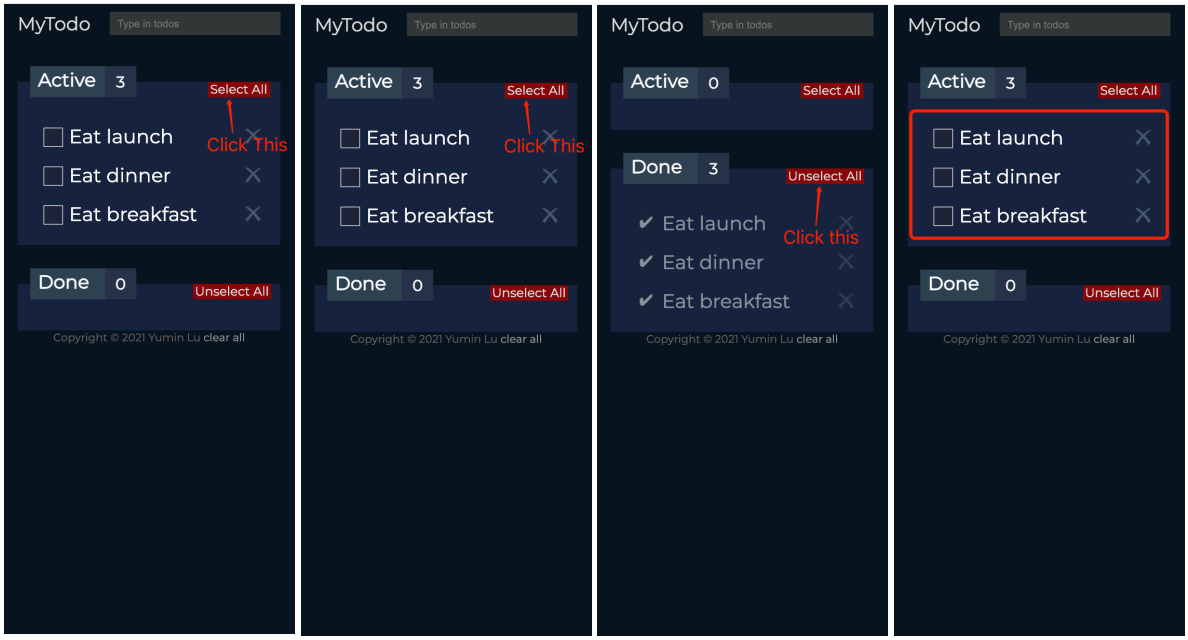
- 标记**单条/多条**Todo项为**已完成/未完成**
  - 标记**单条**Todo项为**已完成/未完成**

    点击相应Todo项左侧的方框即可改变此Todo项的状态为**已完成/未完成**



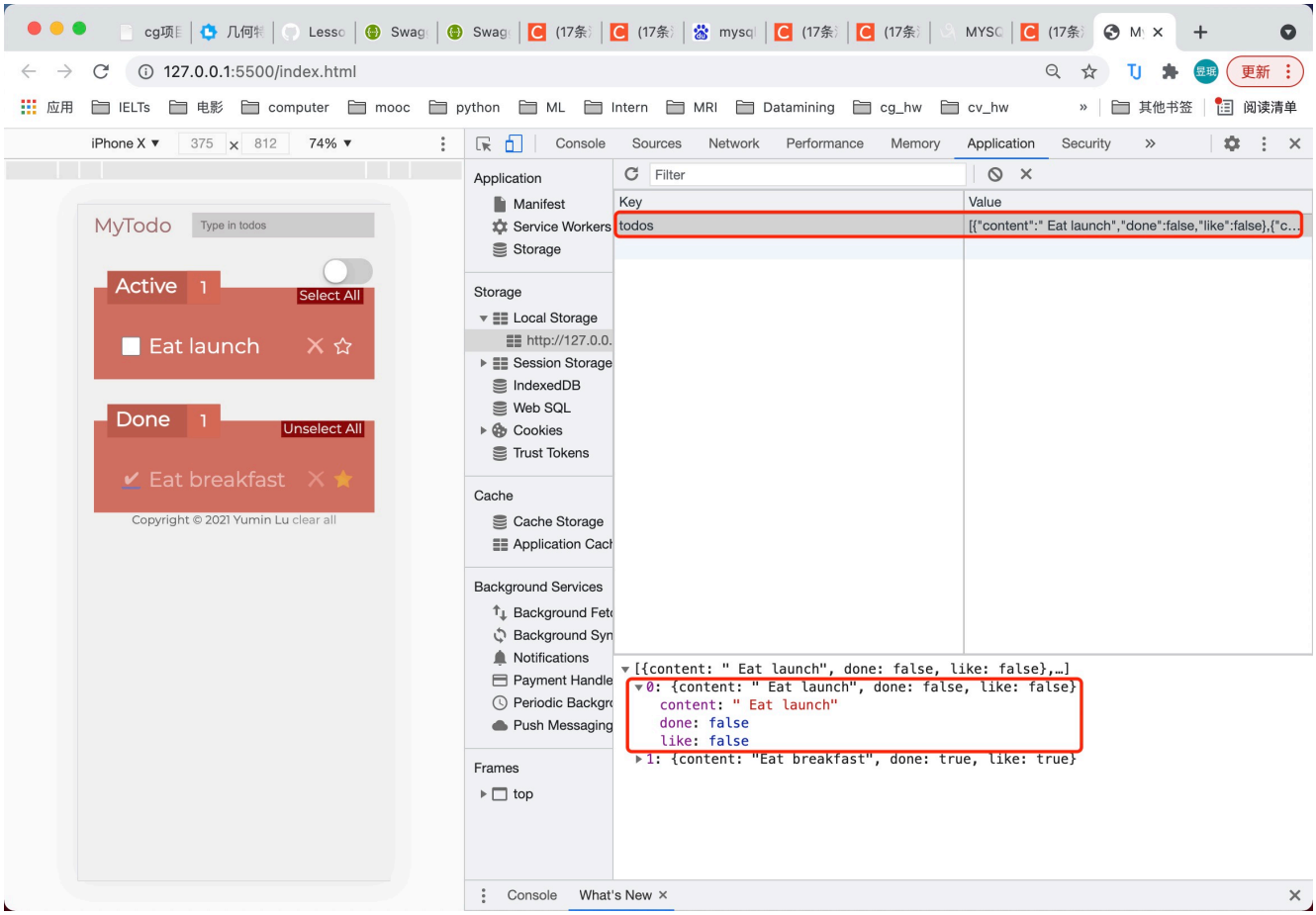
  - 标记**单多**Todo项为**已完成/未完成**

    点击左侧**Select All/Unselect All**即可改变该栏目内所有Todo项的状态为**已完成/未完成**



- 数据持久化

  使用**Local Storage**进行数据的持久化，实现页面刷新Todo数据保留的功能



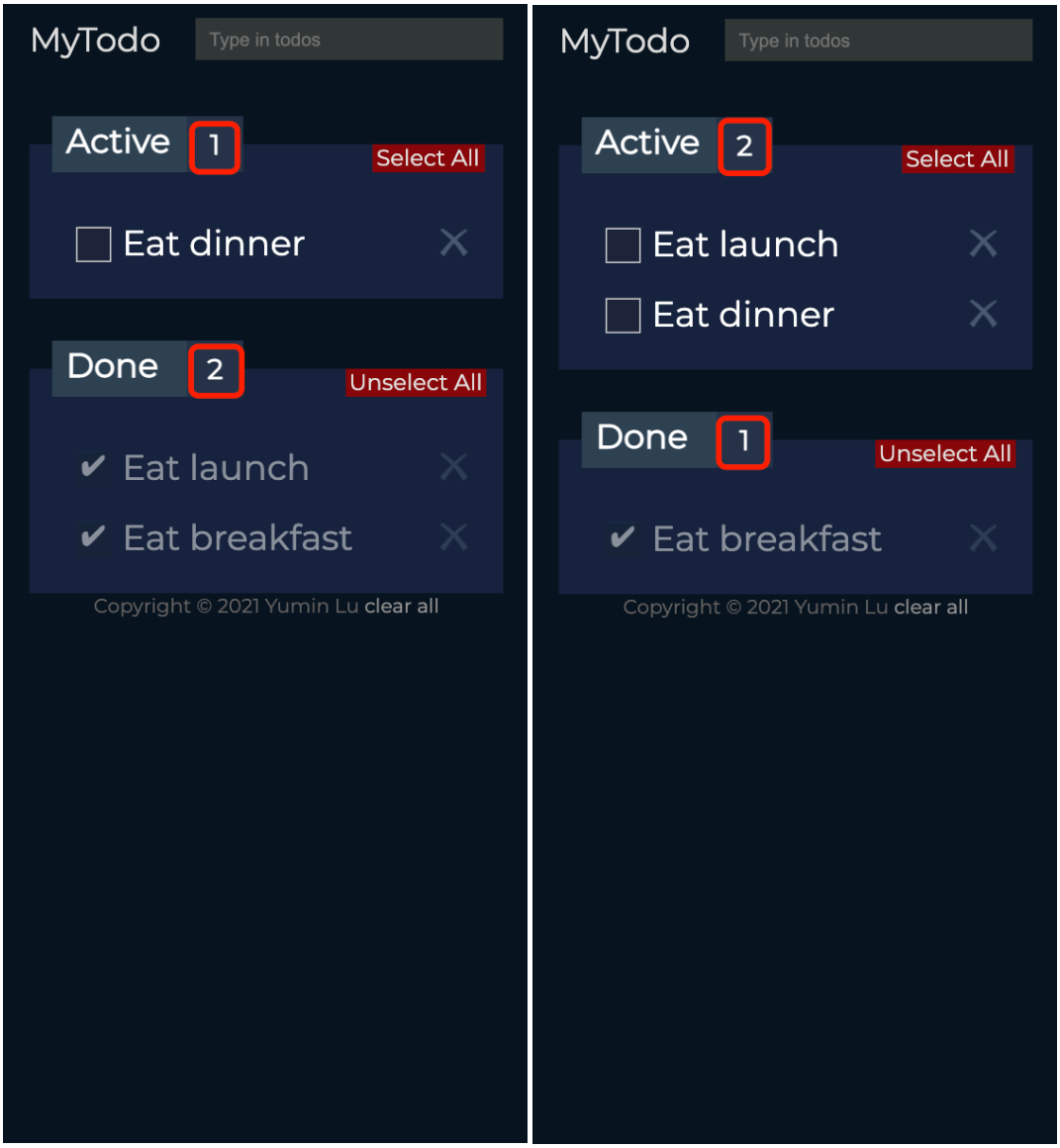
图中每一条Todo的数据结构如下方红色框中所示，具体含义为：

```
{
  content: "Todo content",   // Todo项所存储的内容
  done: false                // Todo项的完成状态：已完成/未完成
  like: false                // Todo项的收藏状态：已收藏/未收藏
}
```

高级功能

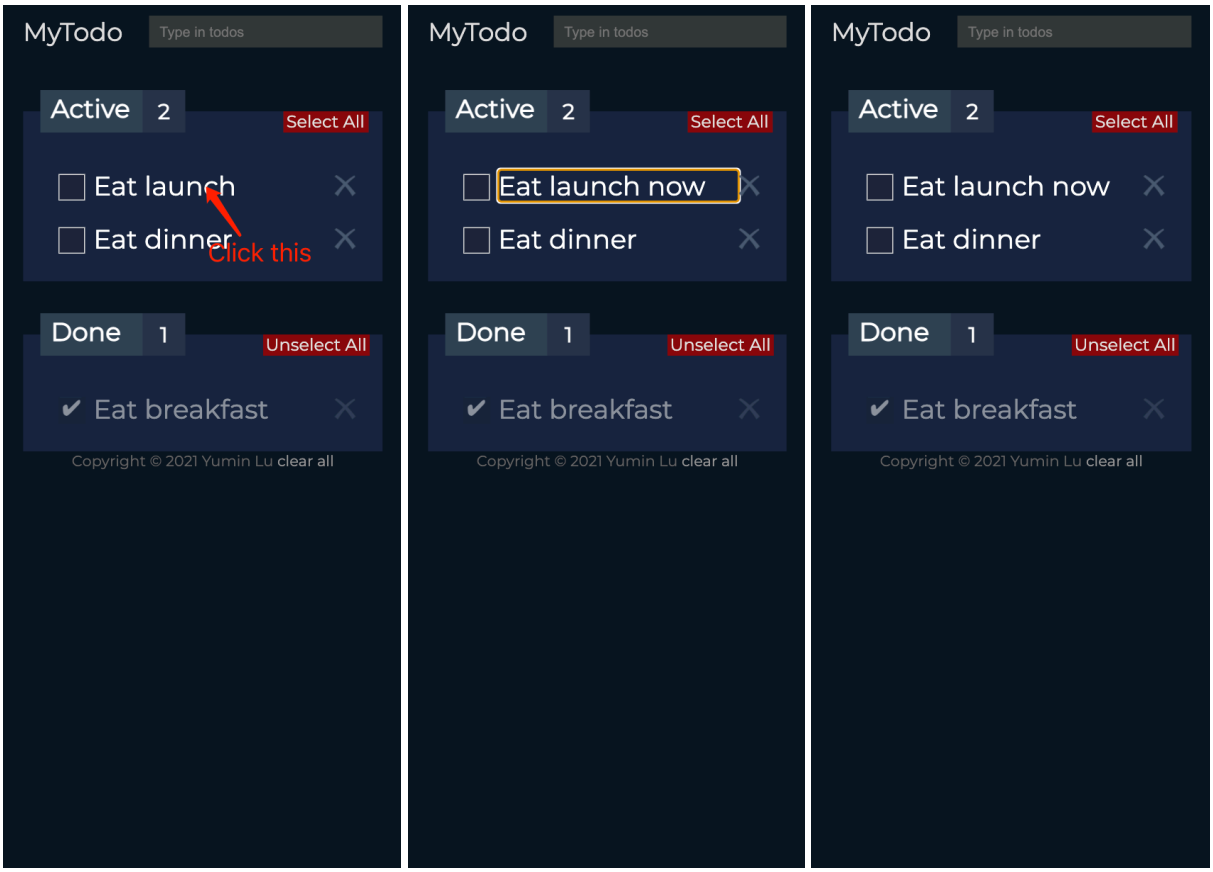
- 实时显示**已完成/未完成**Todo项数量

  当有**一条/多条**Todo项被改变状态时，相应栏目的Todo项计数器会自动调整



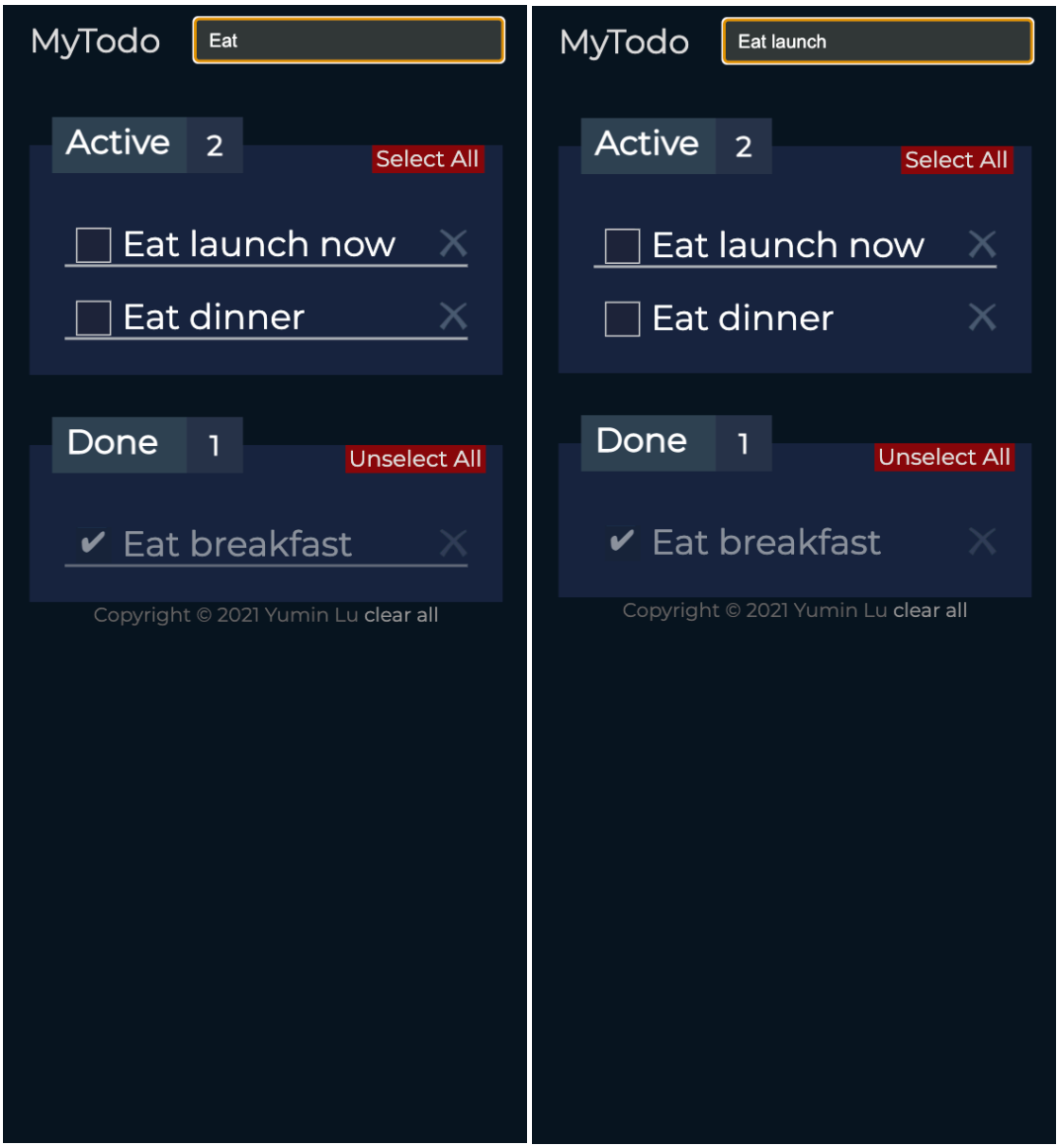
- 单条Todo内容编辑

  将**手指/鼠标**放置在一条Todo项的文字内容上并点击，即可开始对相应Todo项的编辑，编辑完成后按下**Enter/完成**即可完成更改



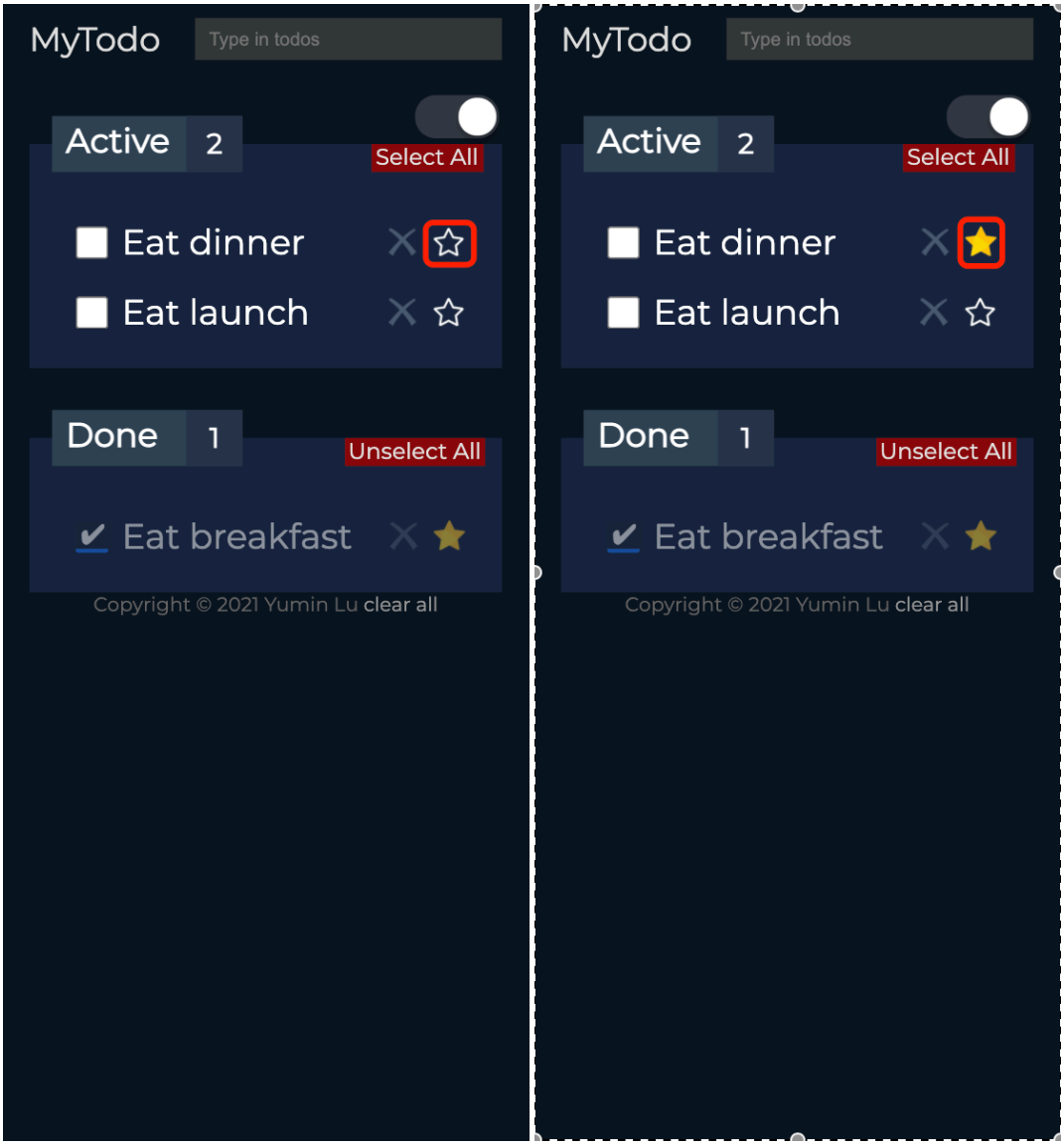
- 查找指定Todo项

  在顶端输入框输入Todo项内容时，如果输入内容与已存在的Todo项中的某一项获多项相同，系统会自动在相应Todo项下添加下划线以便用户查找



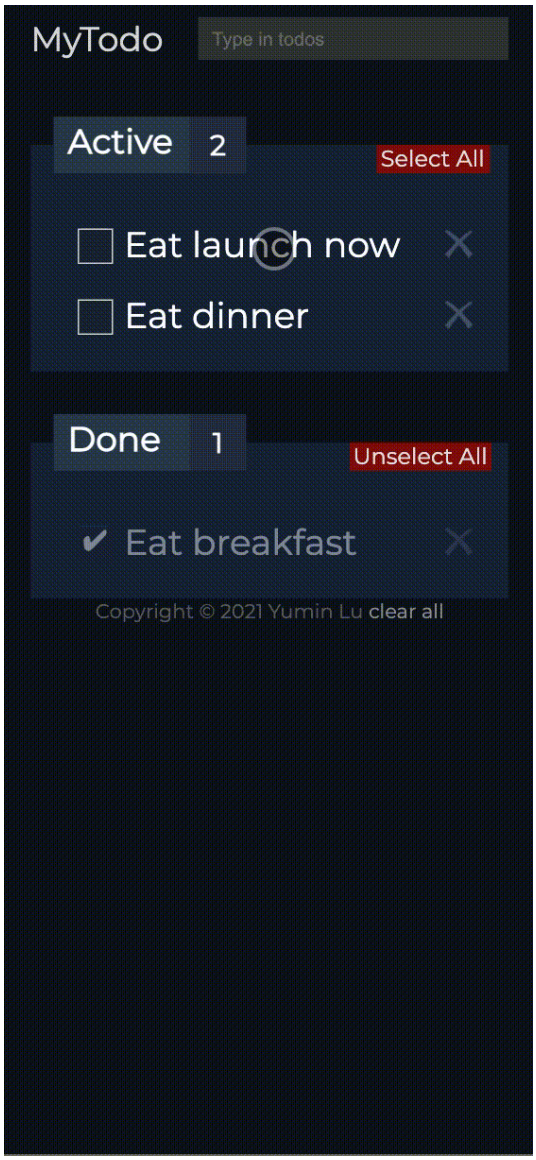
- 标记收藏Todo项

  点击下图红框中的五角星即可将该条Todo项标记为**收藏**，再次点击为**取消收藏**



- 调换相应Todo项显示次序

  用户可以调整在**Active栏（未完成）**中的Todo项的排列次序，具体做法为将**鼠标/手指**点击某一个Todo项并将其拖拽到另一个在同一栏目中的todo项之上，随后**松开鼠标/停止触摸**，即可完成Todo项的次序替换

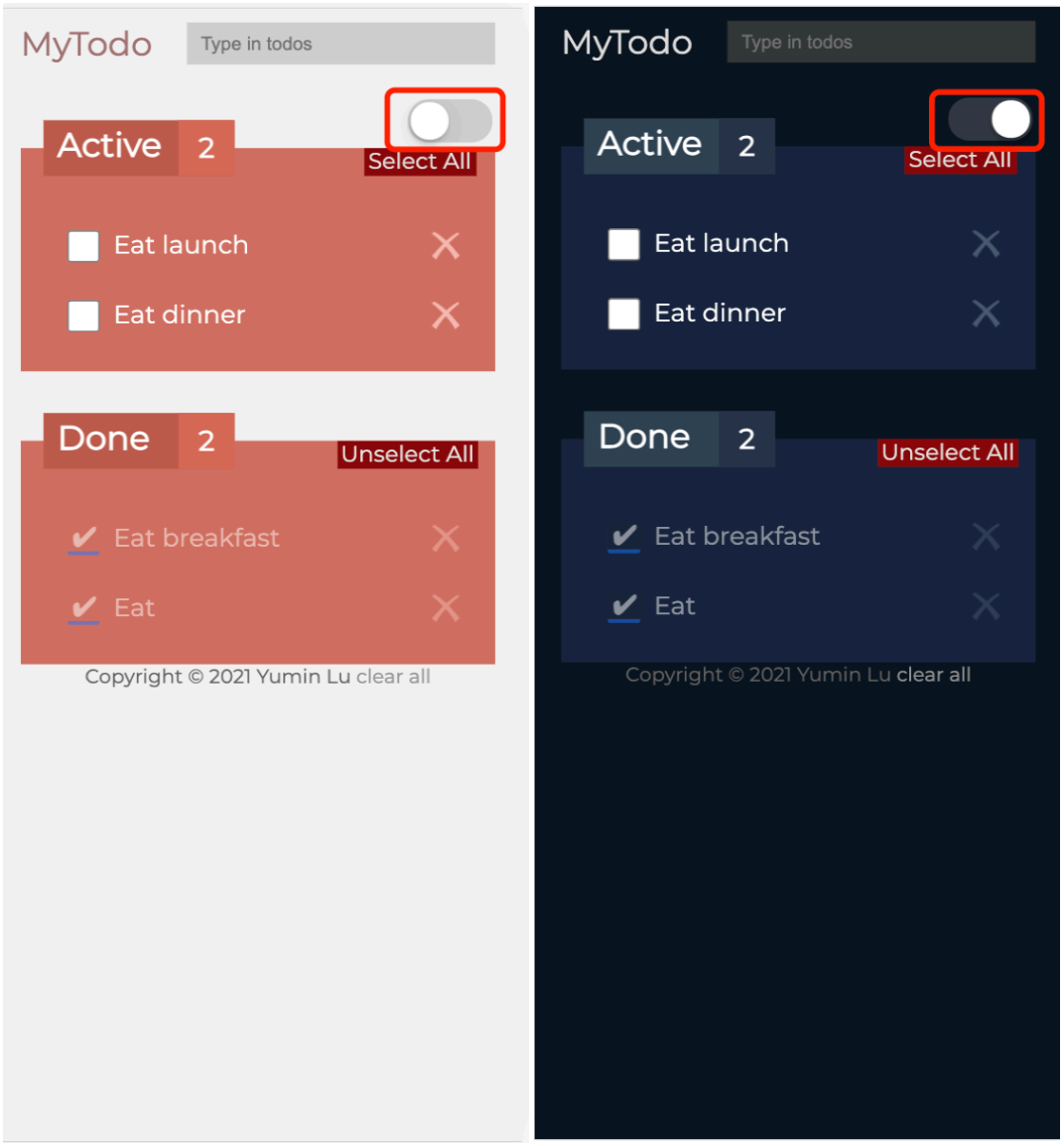


- 简洁优雅的style

  Mytodo整体设计风格采用扁平简约风格，总体外观简约大方，十分美观。

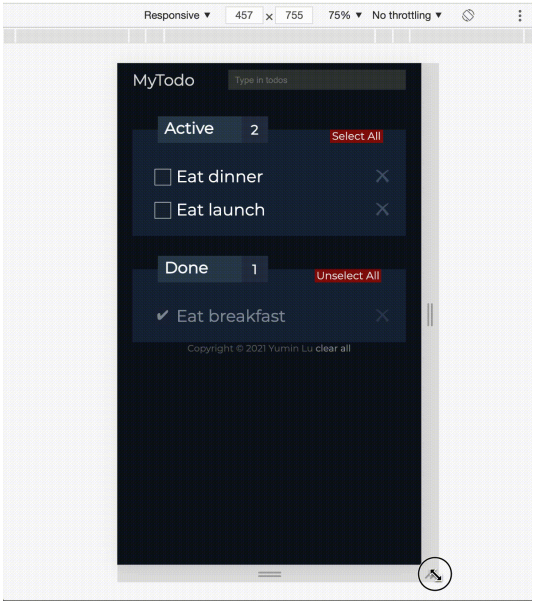
- 可切换的主题模式

  Mytodo拥有两种主题模式，分别为黑暗模式和明亮模式，可通过下图中的切换开关开启，项目打开时默认为明亮模式



- 响应式布局

  Mytodo可以适应各种不同的页面大小，并且根据页面大小实时调整布局，因此**Mytodo**可以完美适配PC网页/手机竖屏网页/手机横屏网页



四、项目结构

本项目包含的文件如下：index.html, todoList.css, todoList.js, *imgs(fold)*.

文件结构为

```
mytodo(project fold)
|
|--imgs
  |--x.png
|--index.html
|--dark.css
|--light.css
|--switch.css
|--todoList.js
```

各文件/文件夹的作用如下：

> Imgs

存放页面所需要的资源图片，如todoList页面中单条todo项需要的删除图标

> index.html

基础的文本内容

> dark.css/light.css

黑暗/明亮风格的网页样式

> switch.css

切换按钮的样式

> todoList.js

```
- 定义了todo数据持久化的数据结构，以及localStorage与model之间数据传输的方法
- 定义了model到view的渲染方法
- 定义了各种事件监听函数
```

todoList.js作为整个todoList项目的逻辑核心部分，相当于MCV模型中的Controller，充当了View与Model之间的传输媒介，其传输方式的具体实现如下：

> View ⇔ Model

在View与Model的切换之中，本程序实现了函数**render()**用以将Model中的数据渲染成视图。其中**todoLocalStore**为存储所有todo项及其状态的变量，其由存储与localStorage中的json数据结构转换而来

```
function render() {
    //clear
    done.innerHTML = '';
    todo.innerHTML = '';
    let i = 0
    for(etodo of todoLocalStore) {
        if(etodo.done) {
            done.append(genreateTodo(etodo.content, "checked", i, etodo.like));
        } else {
            todo.append(genreateTodo(etodo.content, "", i, etodo.like));
        }
        i ++;
    }
    //change cooresponding num
    $(".todoCount").innerText = todo.children.length;
    $(".doneCount").innerText = done.children.length;
    //data localization
    localStorage.setItem("todos", JSON.stringify(todoLocalStore));
}
```

> Model ⇔ Local Storage

在将Model中的数据持久化的环节中，本程序使用了localStorage实现上述目的。具体做法为将实现Model的**todoLocalStore**通过**JSON.stringify()**方法转换为Json格式的字符串存储与localStorage中。当需要读取本地持久化数据时本程序会将存储与localStorage中的Json格式字符串读取并通过**JSON.parse()**方法解析

- Model ⇒ Local Storage

  ```
  localStorage.setItem("todos", JSON.stringify(todoLocalStore));
  ```

- Model ⇐ Local Storage

  ```
  var todoLocalStore = localStorage.getItem("todos");
  todoLocalStore = JSON.parse(todoLocalStore);
  ```

> Todo项数据结构

单条Todo项的数据结构如下所示

```
[{"content":"Eat launch","done":false,"like":false},
 {"content":"Eat breakfast","done":true,"like":true}]
```

五、项目评估

1. 功能完成上，本项目完成了脚本程序设计课程项目所要求的基本功能点，同时实现了若干附加功能点：单条todo编辑、实时显示todo数量、查找指定todo项、收藏todo项、拖动todo项以调换次序的交互方式、简介美观的样式、兼容性良好的布局
2. 页面内各元素简洁整齐；去除了一些繁复的按钮代以达到简洁；使用常用的手势进行操作，符合移动端的设备特点和用户体验
3. 该项目可同时应用于pc端以及移动端，兼容性良好
4. 代码逻辑清晰，函数命名可读性强