实验 08：使用本地存储

一、实验目的

1、掌握本地存储的核心概念与区别，理解其在“数据持久化”中的作用；

2、学会使用 localStorage/sessionStorage 的 API，实现数据的本地存储与读取。

二、实验学时

2 学时

三、实验类型

综合性

四、实验需求

1、硬件

每人配备计算机 1 台，建议优先使用个人计算机开展实验。

2、软件

安装 Visual Studio Code，以及 Edge 浏览器。

3、网络

本地主机能够访问互联网和实验中心网络。

4、工具

无。

五、实验任务

1、实现用户偏好设置

2、实现购物车本地缓存

3、实现待办事项列表

六、实验内容及步骤

1、用户偏好设置

步骤 1：创建 HTML 结构

XML/HTML

```
<h3>用户偏好设置</h3>
<div class="pref-group">
  <label>主题：</label>
  <button id="themeLight" class="theme-btn">浅色</button>
  <button id="themeDark" class="theme-btn">深色</button>
</div>
<div class="pref-group">
  <label>语言：</label>
  <select id="languageSelect">
    <option value="zh-CN">中文</option>
    <option value="en-US">英文</option>
  </select>
</div>
```

步骤 2：实现 CSS 样式，示例如下。

CSS

```css
/* 默认浅色主题 */
body {
  background-color: #f5f5f5;
  color: #333;
}
/* 深色主题 */
body.dark-theme {
  background-color: #1a1a1a;
  color: #fff;
}
.theme-btn {
  padding: 5px 10px;
  margin-right: 10px;
  cursor: pointer;
}
```

步骤 3：实现 JavaScript 代码，示例如下。

JavaScript

```javascript
// 1. 获取DOM元素
const themeLight = document.getElementById('themeLight');
const themeDark = document.getElementById('themeDark');
const languageSelect = document.getElementById('languageSelect');
const body = document.body;

// 2. 页面加载时：读取本地存储的主题/语言并应用
const savedTheme = localStorage.getItem('userTheme');
const savedLanguage = localStorage.getItem('userLanguage');

// 应用保存的主题
if (savedTheme === 'dark') {
  body.classList.add('dark-theme');
}
// 应用保存的语言（示例仅做演示，可扩展翻译逻辑）
languageSelect.value = savedLanguage || 'zh-CN';

// 3. 主题切换事件
themeLight.addEventListener('click', () => {
  body.classList.remove('dark-theme');
  localStorage.setItem('userTheme', 'light'); // 存储浅色主题
});
themeDark.addEventListener('click', () => {
  body.classList.add('dark-theme');
  localStorage.setItem('userTheme', 'dark'); // 存储深色主题
});

// 4. 语言选择事件
languageSelect.addEventListener('change', (e) => {
  localStorage.setItem('userLanguage', e.target.value); // 存储语言选择
  alert(`语言已切换至: ${e.target.value}`);
});
```

2、购物车本地缓存

步骤 1：创建 HTML 结构

XML/HTML

```xml
<h3>商品列表</h3>
<div class="product-list">
  <div class="product" data-id="1" data-name="苹果" data-price="5">
    <p>苹果 - 5元/斤</p>
    <button class="add-to-cart">加入购物车</button>
  </div>
  <div class="product" data-id="2" data-name="香蕉" data-price="3">
    <p>香蕉 - 3元/斤</p>
    <button class="add-to-cart">加入购物车</button>
  </div>
</div>

<h3>购物车</h3>
<div id="cart" class="cart">
  <p class="empty-cart">购物车为空</p>
</div>
```

步骤 2：定义 CSS 样式

CSS

```css
.product-list {
  display: flex;
  gap: 20px;
  margin-bottom: 20px;
}
.product {
  border: 1px solid #eee;
  padding: 10px;
  border-radius: 8px;
  cursor: pointer;
}
.cart {
  border: 1px solid #eee;
  padding: 10px;
  border-radius: 8px;
  min-height: 100px;
}
.cart-item {
  padding: 5px 0;
  border-bottom: 1px solid #f5f5f5;
}
.empty-cart {
  color: #999;
  text-align: center;
}
```

步骤 3：实现 JavaScript 逻辑

JavaScript

```javascript
// 1. 获取DOM元素
const products = document.querySelectorAll('.product');
const cart = document.getElementById('cart');
const emptyCart = cart.querySelector('.empty-cart');

// 2. 初始化购物车：从本地存储读取数据
let cartData = JSON.parse(localStorage.getItem('cart')) || []; // 若无数据则初始化空数组

// 3. 渲染购物车函数
function renderCart() {
  // 清空购物车
  cart.innerHTML = '';
  // 若购物车为空，显示提示
  if (cartData.length === 0) {
    cart.appendChild(emptyCart.cloneNode(true));
    return;
  }
  // 渲染购物车项
  cartData.forEach(item => {
    const cartItem = document.createElement('div');
    cartItem.className = 'cart-item';
    cartItem.textContent = `${item.name} - ${item.price}元/斤 × ${item.quantity}`;
    cart.appendChild(cartItem);
  });
}

// 4. 加入购物车事件
products.forEach(product => {
  const addBtn = product.querySelector('.add-to-cart');
  addBtn.addEventListener('click', () => {
    const id = product.dataset.id;
    const name = product.dataset.name;
    const price = parseFloat(product.dataset.price);

    // 查找购物车中是否已有该商品
    const existingItem = cartData.find(item => item.id === id);
    if (existingItem) {
      existingItem.quantity += 1; // 数量+1
    } else {
      cartData.push({ id, name, price, quantity: 1 }); // 新增商品
```

```
    }

    // 保存到本地存储
    localStorage.setItem('cart', JSON.stringify(cartData));
    // 重新渲染购物车
    renderCart();
  });
});

// 5. 页面加载时渲染购物车
renderCart();
```

3、待办事项列表

步骤 1：HTML 结构搭建

XML/HTML

```
<h3>待办事项</h3>
<div class="todo-input">
  <input type="text" id="todoInput" placeholder="输入待办事项...">
  <button id="addTodo">添加</button>
</div>
<ul id="todoList" class="todo-list"></ul>
```

步骤 2：定义 CSS 样式

CSS

```css
.todo-input {
  display: flex;
  gap: 10px;
  margin-bottom: 20px;
}
#todoInput {
  flex: 1;
  padding: 8px;
}
.todo-list {
  list-style: none;
  padding: 0;
}
.todo-item {
  display: flex;
  align-items: center;
  padding: 10px;
  border-bottom: 1px solid #eee;
}
.todo-item input[type="checkbox"] {
  margin-right: 10px;
}
.todo-item.completed span {
  text-decoration: line-through;
  color: #999;
}
.todo-item .delete-btn {
  margin-left: auto;
  color: red;
  cursor: pointer;
}
```

步骤 3：JavaScript 逻辑实现

JavaScript

```javascript
// 1. 获取DOM元素
const todoInput = document.getElementById('todoInput');
const addTodoBtn = document.getElementById('addTodo');
const todoList = document.getElementById('todoList');

// 2. 初始化待办事项：从本地存储读取
let todos = JSON.parse(localStorage.getItem('todos')) || [];

// 3. 渲染待办事项函数
function renderTodos() {
  todoList.innerHTML = '';
  todos.forEach((todo, index) => {
    const li = document.createElement('li');
    li.className = `todo-item ${todo.completed ? 'completed' : ''}`;
    li.innerHTML = `
      <input type="checkbox" ${todo.completed ? 'checked' : ''} data-index="${index}">
      <span>${todo.text}</span>
      <span class="delete-btn" data-index="${index}">×</span>
    `;
    todoList.appendChild(li);
  });
}

// 4. 添加待办事项
addTodoBtn.addEventListener('click', () => {
  const text = todoInput.value.trim();
  if (text) {
    todos.push({ text, completed: false }); // 新增待办
    localStorage.setItem('todos', JSON.stringify(todos)); // 保存
    renderTodos(); // 重新渲染
    todoInput.value = ''; // 清空输入框
  }
});

// 5. 处理勾选/删除事件（事件委托）
todoList.addEventListener('click', (e) => {
  const index = e.target.dataset.index;
  if (!index) return;

  if (e.target.type === 'checkbox') {
    // 勾选/取消勾选
```

```
    todos[index].completed = e.target.checked;
  } else if (e.target.classList.contains('delete-btn')) {
    // 删除待办
    todos.splice(index, 1);
  }

  // 保存并重新渲染
  localStorage.setItem('todos', JSON.stringify(todos));
  renderTodos();
});

// 6. 页面加载时渲染待办事项
renderTodos();
```

七、实验考核

本实验考核采用【实验随堂查】方式开展。

每个实验完成后，在实验课上通过现场演示的方式向实验指导教师进行汇报，并完成现场问答交流。

每个实验考核满分 100 分，其中实验成果汇报 60 分，现场提问交流 40 分。

实验考核流程：

（1）学生演示汇报实验内容的完成情况，实验指导老师现场打分。

（2）指导老师结合实验内容进行提问，每位学生提问 2-3 个问题，根据回答的情况现场打分。

（3）实验考核结束后，进行公布成绩。