第三次作业报告

刘叙杨　学号：2023010838
liuxuyan23@mails.tsinghua.edu.cn

1　实现思路

本次作业在第二次作业的基础上，增加了流量统计功能、时间流动和用户名密码储存，实现了用户流量使用情况的动态跟踪和持久化存储。

在页面外观上，保留了前两次作业的页面样式。在技术实现方面，主要使用 JavaScript 完成核心逻辑功能，并利用 localStorage 实现数据的持久化存储。

登录页面逻辑实现：

- **用户认证与注册**：在登录按钮点击事件中，通过 `querySelector` 获取用户名和密码输入框的值，检查是否为空。如果用户已注册，验证密码正确性；如果是新用户，自动注册并保存用户信息。验证通过后，使用 `localStorage.setItem('currentUsername', username)` 存储当前用户名，并通过 `window.location.href` 跳转到成功页面。
- **用户名密码储存功能**：实现了完整的用户管理系统，包括：
  - **用户数据存储**：使用 `localStorage.setItem('registeredUsers', JSON.stringify(registered))` 将用户注册信息以 JSON 格式存储，数据结构为 `{ " 用户名": " 密码" }`。
  - **用户数据读取**：通过 `JSON.parse(localStorage.getItem('registeredUsers') || '')` 读取已注册用户信息，如果不存在则初始化为空对象。
  - **用户验证逻辑**：检查用户名是否存在于注册列表中，如果存在则验证密码匹配性，如果不存在则自动注册新用户。
  - **数据持久化**：用户注册信息在浏览器关闭后仍然保留，确保用户无需重复注册。

登录成功页面逻辑实现：

- **用户名显示**：页面加载时通过 `DOMContentLoaded` 事件监听器，使用 `localStorage.getItem('currentUsernar` 获取存储的用户名，并更新到页面显示。
- **时间流动功能**：实现了实时在线时长统计系统，包括：
  - **时间初始化**：页面加载时记录开始时间 `startTime = Date.now()`，用于计算在线时长。
  - **时间计算**：通过 `computeTime()` 函数计算从页面加载到当前的时间差，将毫秒转换为小时、分钟、秒数。
  - **时间格式化**：使用 `padStart(2, '0')` 确保时间显示为两位数格式（HH:MM:SS），如”01:23:45”。
  - **实时更新**：通过 `setInterval(computeTime, 1000)` 每秒更新一次时间显示，确保在线时长的实时性。

  - **定时器管理**：在登出时通过 `clearInterval(timeInterval)` 清理定时器，防止内存泄漏。

- **流量统计功能**：实现了完整的流量跟踪系统，包括：
  - **流量数据加载**：从 localStorage 中读取用户历史流量数据，格式为 `{ " 用户名": 流量值 }`，如果是首次登录则初始化为 0GB。
  - **流量动态更新**：通过 `setInterval(computeTraffic, 1000)` 每秒调用一次流量计算函数，每次增加 3.33GB，最大限制为 50GB。
  - **流量数据存储**：使用 `localStorage.setItem('trafficUsage', JSON.stringify(traffic))` 将流量数据持久化存储。
  - **流量显示格式**：使用 `trafficUsage.toFixed(2) + ' GB'` 确保流量显示保留两位小数。
- **进度条动态更新**：通过 `updateProgressBar()` 函数实现，根据当前流量使用量计算百分比（流量使用量/50GB*100），并通过 `progressFill.style.width` 动态设置进度条宽度。
- **登出功能**：监听登出按钮点击事件，清除所有定时器防止内存泄漏，使用 `localStorage.removeItem('current` 清除当前用户信息但保留流量数据，并通过 `window.location.href = '../index.html'` 返回登录页面。

数据传递与存储机制：使用浏览器的 `localStorage` API 实现数据的持久化存储，包括用户注册信息、当前登录状态和流量使用数据，确保页面间的数据一致性和用户数据的持久性。

2 使用说明

将文件下载后，可以发现/src/index.html。可以直接用浏览器打开。此外/src/csstable 为样式表，/src/hw1 为第一次作业所做页面，这里对于部分内容做了更改后拿来使用，src/source 为图片资源。整体上，对于/src/hw1/index_success.html 和/src/index.html 进行了部分修改，以实现本次作业目标。

3 问题及解决办法

在实现过程中遇到的主要问题包括：

- **流量统计逻辑错误**：最初使用基于总秒数的计算方式 `seconds * 3.33`，导致流量一次性增加过多。后改为使用定时器每秒增加固定值 `trafficUsage += 3.33`，确保流量按预期增长。
- **流量数据持久化**：最初在登出时删除所有流量数据 `localStorage.removeItem('trafficUsage')`，导致用户数据丢失。后改为保留流量数据，只删除当前用户名，确保用户流量使用历史的连续性。
- **用户名密码储存安全性**：最初考虑使用简单的字符串存储用户信息，后发现使用 JSON 格式更便于管理和扩展。为此我学习了 JSON 相关的内容，通过 `JSON.stringify()` 和 `JSON.parse()` 实现数据的序列化和反序列化，确保数据结构的完整性。

4 参考资料

除了前两次作业中的参考内容之外，本次作业中，为了实现流量统计功能，参考了 JavaScript 定时器 API 文档、localStorage 数据存储最佳实践，以及 JSON 数据格式处理的相关资料。此外使用了大模型进行代码注释的补全，以及代码格式的规范，确保代码的可读性和可维护性。