南华大学

计算机学院

实 验 报 告

（ 2024 ~2025学年度 第 1 学期 ）

**课程名称 Web 全栈开发**

**姓 名 邹伟**

**学 号 20210440924**

**专 业 数据科学与大数据技术**

# 一、实验目的

创建一个简单的寻宝游戏网页应用，实现玩家信息的存储、任务的展示与执行、进度的更新以及背景音乐的控制。

# 二、实验环境

VS Code

# 三、实验步骤

1. **设置文档结构和样式**：
   1. 创建 HTML 文档结构，包括头部、主体和页脚。
   2. 引入 CSS 样式表，设置页面的整体风格，包括背景颜色、字体样式、容器样式、任务样式、按钮样式等。
2. **定义全局变量和常量**：
   1. 确定玩家信息存储的键 PLAYER\_KEY。
   2. 定义任务数组 tasks，包含任务名称和持续时间。
   3. 初始化玩家数据 playerData。
   4. 获取 DOM 元素的引用，如 playerInfoDiv、tasksContainer、progressDiv、backgroundMusic 和 toggleMusicButton。
3. **实现功能函数**：
   1. loadPlayer()：从本地存储中加载玩家信息，如果没有则创建新玩家并保存。
   2. savePlayer()：将玩家信息保存到本地存储。
   3. displayPlayerInfo()：显示玩家的信息，包括 ID、昵称和已完成任务数。
   4. displayTasks()：展示任务列表，根据玩家完成情况设置按钮的状态。
   5. startTask()：开始一个任务，设置定时器模拟任务执行时间。
   6. completeTask()：完成任务后更新玩家历史记录，保存玩家信息，更新进度并显示相应提示。
   7. updateProgress()：更新任务进度的显示，当所有任务完成时显示恭喜信息。
   8. toggleMusicButton.addEventListener()：处理背景音乐的开关操作。
4. **初始化游戏**：
   1. initGame()：依次调用 loadPlayer()、displayTasks() 和 updateProgress() 来初始化游戏。

# 四、代码示例

1.function startTask(index) {

const task = tasks[index];

alert(`任务 "${task.name}" 任务已开始！请稍候...`);

setTimeout(() => {

completeTask(task.name);

}, task.duration);

}

开始一个任务，显示提示信息，并使用 setTimeout 模拟任务的执行时间，时间到后调用 completeTask 函数完成任务

2.function updateProgress() {

progressDiv.innerHTML = `任务进度: ${playerData.history.length}/${tasks.length}`;

if (playerData.history.length === tasks.length) {

alert("恭喜您，完成所有任务，获得宝藏！");

}

displayPlayerInfo();

displayTasks();

}

更新任务进度显示，并在所有任务完成时显示恭喜信息，同时重新显示玩家信息和任务列表。

3.function savePlayer() {

localStorage.setItem(PLAYER\_KEY, JSON.stringify(playerData));

}

将玩家数据以 JSON 格式保存到本地存储中。

# 五、实验结果

1. 打开网页后，会显示一个带有寻宝游戏标题的页面，页面背景为渐变的深色。
2. 页眉部分显示游戏标题，背景为特定颜色，文字为白色。
3. 游戏容器中会显示玩家的信息，包括玩家 ID、昵称和已完成任务数。
4. 下方会展示任务列表，每个任务包含名称、时长和一个开始任务的按钮（如果该任务未完成）。已完成的任务按钮会被禁用。
5. 再下方会显示任务进度。
6. 页脚有一个用于开关背景音乐的按钮，按钮颜色醒目（例如红色），呈圆形。
7. 当点击开始任务按钮后，会弹出提示框告知任务已开始，并在设定的时间后提示任务完成。
8. 完成任务后，任务进度会更新，已完成任务数会增加，相应的任务按钮会被禁用。当完成所有任务时，会弹出恭喜完成所有任务的提示。

# 六、问题与解决

**问题**：

1. 本地存储数据读取和写入异常。
2. 任务状态管理可能出现混乱，导致任务完成情况显示不正确。
3. 定时器的使用可能导致任务执行时间不准确。
4. 页面样式的调整可能出现兼容性问题，导致在某些浏览器中显示效果不理想。

**解决方法**：

1. 对于本地存储数据读取和写入异常的问题，需要仔细检查代码中使用 localStorage 的部分，确保键值对的设置和读取正确。可以通过打印日志或使用浏览器的开发者工具来检查存储的数据是否符合预期。
2. 为了避免任务状态管理混乱，在 completeTask 函数中，仔细检查任务是否已经完成，只有未完成的任务才会被添加到历史记录中，并进行相应的处理。
3. 定时器的时间可能会受到系统性能等因素的影响，导致任务执行时间不准确。可以根据实际情况进行适当的调整和测试，以确保任务的执行时间在可接受的范围内。
4. 对于页面样式的兼容性问题，可以使用 CSS 预处理器（如 Sass 或 Less）来编写更具可维护性的样式代码，并使用工具进行兼容性测试，确保页面在各种主流浏览器中都能正常显示。此外，还可以参考 CSS 重置库来统一不同浏览器的默认样式，减少兼容性问题的出现。

# 七、实验总结

* 通过这个实验，深入了解了如何使用 HTML、CSS 和 JavaScript 来创建一个具有一定交互性的网页应用。
* 学会了如何使用本地存储来保存玩家信息，实现数据的持久性。
* 掌握了如何通过定时器来模拟任务的执行时间，以及如何根据任务的完成情况更新页面的显示和玩家数据。
* 在处理 DOM 操作和事件监听时，提高了对 JavaScript 与 HTML 交互的理解和掌握能力。
* 通过设计 CSS 样式，增强了对页面布局和美化的能力，提高了用户体验。
* 在开发过程中，遇到了一些问题，如数据的存储和读取、任务状态的管理等，通过不断调试和改进，提高了解决问题的能力。
* 总体来说，这个实验不仅提高了技术水平，还培养了逻辑思维和项目开发的能力。