项目申请书

- 项目名称：为洛书语言编写交互式学习手册
- 项目导师：neuqmiao
- 申请人：车沭遥
- 日期：2025.06.15
- 邮箱：2263608676@qq.com

项目背景

洛书语言正处于 Beta 阶段（v2.x），面向社区开发者与学习者，但目前缺乏系统、易上手的交互式学习文档，社区用户在了解安装与入门、程序结构、数据类型、编码技巧等方面仍存在较高的学习门槛。因此，编写一份基于 WASM 具备 PlayGround 环境的交互式学习手册，并以在线站点部署到社区服务器，能够有效降低学习门槛、提升用户体验，推动洛书语言生态发展。

项目基本需求

1. 文档内容编写
    - 梳理洛书语言 Beta v2.x 的安装与入门指引，包括本地环境依赖、WASM 运行环境要求、快速安装脚本等。
    - 详述程序结构：语言核心组件、模块划分、插件机制（如有）、编译或解释流程示意。
    - 数据类型说明与编码技巧：基本类型与复合类型、安全实践、常见编码范式、性能优化建议等。
    - 示例集合：从简单示例到较复杂的实战案例，并配合可交互的 PlayGround 演示。
2. 交互式运行支持
    - 基于 WASM 技术，将洛书语言解释器或编译器（若可编译为 WASM）嵌入浏览器环境，提供在线 PlayGround，使用户可在手册中直接运行、修改并测试示例代码。
    - 设计并实现前端交互组件：代码编辑区、运行按钮、输入/输出展示区、错误提示等。
    - 确保运行沙箱的安全性：隔离执行环境、防止无限循环或资源滥用（如超时机制、内存限制）。
3. 在线站点部署
    - 选用静态站点生成器或轻量化框架（如 VitePress、Gatsby、Docusaurus、VuePress 等），以 Markdown 为源，集成交互式 PlayGround 组件。
    - CI/CD 流程：文档或示例更新后，自动构建并部署至社区服务器（如通过 GitHub Actions、GitLab CI、或其他持续集成工具），保证站点内容与代码同步、及时上线。
    - 兼容常见浏览器，保证移动端或桌面端访问体验。
4. 开源协议
    - 整个项目（文档、示例、PlayGround 代码等）采用 MIT LICENSE，符合社区开源规范。
5. 项目产出要求

1. 根据社区工作进展，完成文档内容编写，全部使用 Markdown 格式，内容结构清晰、可读性高。
2. 接入交互式运行功能，实现基于浏览器的 PlayGround 环境，兼容主要现代浏览器。
3. 将项目打包并部署至社区服务器，提供稳定可访问的在线手册入口。

6. 项目成果仓库
    - 主仓库地址： https://gitee.com/chen-chaochen/lpk

技术方法及可行性

1. 文档与站点框架
    - 选用轻量且支持组件的静态站点生成器（如 VitePress 或 VuePress），以 Markdown 为主源格式。
    - 通过内置或自定义插件，将示例代码与交互组件挂载在文档页面中，保持结构清晰、易于维护。
    - 本地开发模式支持快速预览，CI/CD 流程完成时自动生成静态资源并部署到社区服务器。
2. WASM 运行时集成
    - 将洛书语言解释器或核心子集编译为 WebAssembly：若已有 C/C++/Rust 实现，可直接编译；若无，可实现一个轻量版仅支持示例所需特性。
    - 在浏览器端通过 JavaScript 加载 WASM 模块，并暴露简单的接口（如 `run(code: string)`），在安全沙箱（Web Worker 或受限环境）中执行示例代码。
    - 实施超时与内存限制，避免无限循环或资源滥用；对运行异常进行友好提示。
3. 前端 PlayGround 组件
    - 使用主流编辑器组件（如 Monaco Editor 或 CodeMirror）提供代码高亮、基础语法提示；UI 包含运行、重置按钮和输出区域。
    - 在文档中通过短代码或自定义标记引用示例，自动渲染为可编辑的交互区域；示例存放在仓库 `/examples/`，便于管理与测试。
    - 捕获并显示运行结果或错误信息，开发阶段可记录详细日志，生产环境只展示必要内容。
4. CI/CD 与部署
    - 在 Gitee/GitHub Actions 等平台配置自动化流程：当文档或示例更新时，执行 Markdown 校验、站点构建、PlayGround 基本加载测试，并将结果部署到社区服务器。
    - 部署方式可选静态托管（rsync/SSH 上传）或容器化/Serverless，根据社区现有环境决定。支持回滚机制，确保出现问题时能快速恢复。
5. 安全与性能
    - 通过 Web Worker 或 iframe 运行 WASM，防止阻塞主线程；设置合理的执行超时（如几秒）和内存上限。
    - 优化 WASM 模块体积（编译优化、按需加载），减少首屏加载时间；对静态资源配置缓存策略，加速后续访问。
    - 定期在主流浏览器（Chrome、Firefox、Edge、Safari）上测试，记录兼容性注意事项；对不支持 WASM 的环境提供降级提示或仅展示静态示例。
6. 可行性分析

- 洛书社区已有基础，洛书编译语言已经具备成熟的体系。
- 个人经验：本人具备前端开发基础、熟悉node.js、具有 Markdown 文档编写经验，拥有一定写作功底，可支撑整体工作。
- 技术风险：WASM 兼容性（需要测试各浏览器）、解释器编译（可能需优化内存与性能）、安全沙箱机制。通过原型测试和分阶段迭代，可降低风险。
- 时间与资源：社区服务器与仓库已有，CI/CD 可复用现有平台（如 GitHub/Gitee Actions）。

核心技术点

- Markdown 文档组织与组件化：定义章节结构、示例标记方式、统一风格（主题、配色、排版）。
- 静态站点生成器定制：根据选型，开发插件或扩展，使其支持 PlayGround 组件自动注入。
- WASM 模块编译与优化：编译洛书解释器/运行时至 WASM，优化体积与启动速度，处理浏览器兼容性。
- 浏览器端沙箱执行：使用 Web Worker 或 iframe，结合超时与内存限制，确保运行安全。
- 前端交互编辑器集成：集成 Monaco 或 CodeMirror，提供代码高亮、自动完成（可选）、示例模板快捷插入。
- CI/CD 流程：Markdown 校验、示例测试、站点构建与自动部署脚本。
- 性能与安全测试：自动化测试脚本验证 PlayGround 在常见浏览器环境下正常运行，无明显性能瓶颈或安全漏洞。

项目实现细节梳理

1. 文档结构与内容管理
    - 目录规划：确定核心章节（我看见已有官方文档部分，则我的文档编写根据官方调整即可）。
    - Markdown 源文件：每章独立文件，示例代码保存在 `/examples/` 目录，通过引用或短代码插入到文档页面。
    - 本地预览：配置静态站点生成器（如 VitePress/VuePress）以支持本地 `dev` 模式，便于编写和校对。
2. PlayGround 集成
    - WASM 模块准备：将洛书解释器或其核心子集编译为 WebAssembly）。
    - 加载与运行接口：在前端通过 JavaScript 动态加载 WASM，提供 `run(code: string)` 等简易接口；在 Web Worker 或受限环境中执行，防止阻塞主线程并设置执行超时限制。
    - 编辑器组件：集成 Monaco Editor 或 CodeMirror，展示代码高亮、行号等基本功能；提供运行、重置按钮和输出区。
    - 文档中嵌入：使用静态站点生成器支持的短代码或自定义组件，将指定示例块渲染成可编辑的 PlayGround 区域。
3. 构建与部署流程
    - CI/CD 配置：在 Gitee 上配置流程，触发时依次执行：Markdown 校验（如链接检查）、站点构建、简单自动化检查（例如通过 Puppeteer/Playwright 验证 PlayGround 页面能加载基本示例），然后自动部署到社区服务器。

    - 部署方式：根据社区环境选择静态托管（如将生成的静态文件通过 rsync/SSH 上传）或容器化部署；确保有回滚机制以应对突发问题。
    - 本地与远程同步：示例代码和文档更新后，自动触发构建并推送，保证线上与仓库内容一致。
4. 安全与性能优化
    - 运行隔离：将 WASM 执行放在 Web Worker（或 iframe）中，添加超时（如几秒）和内存限制；对可能长时间运行或高资源示例，提供“仅展示代码、不执行”或简化版。
    - 体积与加载：编译时开启优化选项，拆分模块、按需加载；对静态资源配置合理缓存策略，加速首次及重复访问。
    - 兼容性测试：在主流现代浏览器（Chrome、Firefox、Edge、Safari）上简要测试，记录已知限制；对不支持 WASM 或 Worker 的旧环境，展示降级提示或静态示例。
5. 社区参与与维护
    - 贡献指南：编写简单易懂的 CONTRIBUTING.md、Issue/PR 模板，引导社区提交文档改进、示例补充或 PlayGround 功能增强。
    - 审校与反馈：邀请核心用户或志愿者审阅文档、测试示例；根据反馈修正并持续迭代。
    - 持续更新：结合社区需求和洛书语言发展动态，定期补充新示例、更新说明，并通过 CI/CD 自动部署。
    - 本地开发支持：提供一键启动脚本（如 `npm run dev`）、Docker 配置（如有必要），降低新贡献者的环境配置门槛。

项目产出

1. 交互式学习手册（Markdown 源文件），覆盖洛书语言 Beta v2.x 的安装、程序结构、数据类型、编码技巧、进阶功能、实战示例等。
2. 在线 PlayGround 组件及集成代码，实现浏览器端运行洛书示例、编辑与调试功能，具备安全沙箱与性能优化。
3. 静态站点工程：基于选定静态站点生成器的配置与插件，实现自动化构建流程；可在本地开发模式下预览文档与 PlayGround。
4. CI/CD 配置与部署脚本：自动化检查、构建、测试与部署到社区服务器，保证手册内容与代码示例及时上线。
5. 自动化测试脚本：用于验证 PlayGround 加载、示例运行、文档链接完整性等。
6. 项目总结报告与贡献指南：包含项目完成情况、遇到的挑战与解决方案、后续扩展建议，以及社区贡献流程说明。
7. 项目仓库（https://gitee.com/chen-chaochen/lpk）中的完整代码与文档，遵循 MIT LICENSE，便于社区 Fork、改进与维护。

项目开发时间计划

时间段	计划内容
07月01日 - 07月10日	深入调研洛书语言现有实现与社区需求，学习洛书编程语言相关知识

时间段	计划内容
07月11日 - 08月31日	根据社区工作进展，完成文档内容编写，编写接入交互式运行功能的模块
09月01日 - 09月30日	将项目打包部署至社区服务器，解决在中期验收阶段中发现的问题，对完成的内容进行更详细的测试

期望

本人第一次参与到大型开源项目的开发中，希望能借此机会使自己的水平有进一步提高，为自己以后做项目积累经验，并且扩展更多的技术栈。