- BEONE H5 活动页 – 服务器集成指南（中文）
  - 概述
  - 1. 项目结构
  - 2. 后端 API 规范
    - 2.1 数据模型
  - 3. 前端代码改动（src/script.js）
    - 示例代码
  - 4. 统计与后台管理界面
    - 前端实现示例
  - 5. 部署检查清单
  - 6. 后续可选功能

BEONE H5 活动页 – 服务器集成指南（中文）

概述

本项目为 H5 活动页面，包含视频播放、表单提交（工号、姓名、寄语）以及本地 `localStorage` 数据保存。以下文档说明如何在服务器端接入后端 API，实现数据上报、统计展示和 CSV 导出功能，并提供部署与测试要点。

1. 项目结构

```
BEONE H5/
├─ index.html                # 主页面
├─ src/
│   ├─ styles.css           # 样式文件
│   ├─ script.js            # 前端逻辑
│   └─ layout_helper.js     # 调试辅助（生产环境删除）
├─ assets/                  # 视频、图片等资源
└─ README.md                # 本说明文档（中文）
```

2. 后端 API 规范

建议使用轻量级的 RESTful 接口（Node.js/Express、Flask、Spring Boot 等），提供以下四个端点：

方法	URL	请求体	返回示例
POST	/api/submit	{ "jobid": "string", "name": "string", "message": "string" }	{ "success": true, "id": "<record_id>" }
GET	/api/submissions	—	{ "submissions": [{ "id": "...", "jobid": "...", "name": "...", "message": "...", "timestamp": "ISO8601" }] }
GET	/api/stats	—	{ "total": 123, "lastSubmission": "2025-12-16T22:00:00Z" }
GET	/api/export	—	CSV 文件（UTF-8 BOM）下载

2.1 数据模型

```
{
  "id": "uuid",
  "jobid": "string",
  "name": "string",
  "message": "string",
  "timestamp": "ISO8601"
}
```

可使用 SQLite、MongoDB 或简单的 JSON 文件持久化。

3. 前端代码改动（src/script.js）

1. 删除本地保存函数 `saveData`，改为 `fetch` 调用后端 `/api/submit`。
2. 成功提交后显示统一提示（如 “我们相约 1 月 5 日”），并清空表单。
3. 错误处理：网络错误或后端返回 `success:false` 时弹出相应提示。

示例代码

```
function submitForm(e) {
  e.preventDefault();
  const payload = {
    jobid: document.getElementById('jobid').value.trim(),
    name: document.getElementById('username').value.trim(),
    message: document.getElementById('message').value.trim()
  };
  fetch('/api/submit', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(payload)
  })
    .then(r => r.json())
    .then(data => {
      if (data.success) {
        showToast('我们相约 1 月 5 日');
        wishForm.reset();
      } else {
        showToast('提交失败，请重试');
      }
    })
    .catch(() => showToast('网络错误'));
}

wishForm.addEventListener('submit', submitForm);
```

将上述代码替换原有的 `wishForm.addEventListener('submit', …)` 逻辑即可。

4. 统计与后台管理界面

原有的管理员弹窗读取 `localStorage`，改为从 `/api/submissions` 拉取数据并展示。

前端实现示例

```
fetch('/api/submissions')
  .then(r => r.json())
  .then(data => {
    const submissions = data.submissions;
    totalCount.textContent = submissions.length;
    dataPreview.innerHTML = submissions.slice(-5)
      .map(s => `[${s.jobid}] ${s.name}: ${s.message.substring(0,10)}...`)
      .join('<br>');
  });
```

在管理员弹窗中新增 **“导出 CSV”** 按钮，点击后访问 `/api/export` 即可下载全部记录。

5. 部署检查清单

- **静态资源**：确保服务器能够正确提供 `index.html`、`src/`、`assets/` 目录下的文件。
- **跨域**：若前端与 API 不在同一域名，需要在后端开启 CORS，允许前端来源。
- **HTTPS**：涉及个人信息时务必使用 HTTPS，防止数据泄露。
- **无需构建**：本项目使用原生 HTML/CSS/JS，无需额外打包步骤。
- **功能测试**：
  1. 视频播放 → 表单出现。
  2. 表单提交 → 服务器成功返回。
  3. 打开后台弹窗 → 正确显示统计数据和最近 5 条记录。
  4. 点击 “导出 CSV” → 下载文件并检查内容。

6. 后续可选功能

- **鉴权**：为后台接口添加简单 token 鉴权，防止未授权访问。
- **限流**：对提交接口做频率限制，防止刷单。
- **服务器端校验**：在后端再次校验 `jobid`、`name`、`message` 的合法性。
- **埋点统计**：记录视频播放、表单打开等行为，便于运营分析。
