BLOG / FIELD NOTES

Astro 部署到 GitHub Pages：理清 site、base 与自定义域名

记录项目站点切换到自定义域名时出现的路径问题，以及统一链接、重定向和构建验证的方法。

2026-10-19 Astro / GitHub Pages / 部署

同一份 Astro 站点部署到 GitHub Pages，可能有两种不同的公开路径：

```
https://victorfengming.github.io/brief/
https://brief.colemak.cn/
```

前者是项目站点，仓库名属于路径的一部分；后者使用自定义域名，站点通常直接位于 `/`。如果部署脚本仍按仓库名设置 base，构建虽然成功，浏览器却会请求 `/brief/zh/`、`/brief/favicon.svg` 之类并不存在的地址。

site 和 base 解决不同问题

Astro 配置里的两个值很容易混淆：

- `site` 表示公开站点的完整 origin，主要用于 canonical URL 和 sitemap。
- `base` 表示站点在 origin 下的路径前缀，用于页面、资源和内部链接。

项目站点可以使用：

```
site = https://victorfengming.github.io
base = /brief
```

切到自定义域名后则应使用：

```
site = https://brief.colemak.cn
base = /
```

当前配置把这两个值交给构建环境，避免在源码里写死某一种部署方式：

```
export default defineConfig({
  site: process.env.SITE_URL || 'https://example.github.io',
  base: process.env.BASE_PATH || '/',
});
```

所有站内路径走同一个入口

只正确设置 `base` 还不够。组件中如果散落着以 `/` 开头的绝对路径，它们会绕过部署前缀。站点使用一个小型辅助函数，把 Astro 暴露的 `BASE_URL` 加到链接前面：

```
const base = import.meta.env.BASE_URL.replace(/\/$/, '');

export const path = (route = '') =>
  `${base}/${route.replace(/^\//, '')}`;
```

于是导航、图片、文章卡片和返回链接都通过 `path()` 生成。无论 base 是 `/brief/` 还是 `/`，调用方都不必自己拼接斜杠。

这里集中处理斜杠很关键。路径工具如果有时返回尾斜杠、有时没有，重定向和导航的比较逻辑会出现一些只在生产环境暴露的问题。

重定向页面也是站点的一部分

这次问题最容易漏掉的是旧入口和根路径。普通组件已经使用了统一路径，但重定向页面仍然写着：

```
<meta http-equiv="refresh" content="0;url=/zh/" />
```

在项目站点下，这会跳到域名根部的 `/zh/`，丢失 `/brief/`。修复方式是先得到经过 base 处理的目标地址，再同时交给 meta refresh、可点击链接和脚本回退：

```
const zhHome = path('zh/');
```

根路径中的客户端语言选择也要复用同一个 base。否则首屏链接正确，JavaScript 执行后却可能把用户带到另一个地址。

部署配置应表达最终公开地址

使用自定义域名时，工作流直接声明最终的站点地址和根路径：

```
- name: Configure site URL
  run: |
    echo "SITE_URL=https://brief.colemak.cn" >> "$GITHUB_ENV"
    echo "BASE_PATH=/" >> "$GITHUB_ENV"
```

根据仓库名自动推导 base 适合默认的 GitHub Pages 项目地址，但它无法知道自定义域名是否把站点映射到了根路径。部署配置显式表达结果，行为会更容易理解和排查。

验证构建产物，而不只验证源码

路径问题经常通过类型检查和组件测试，因为错误只有在 Astro 注入 base 并生成 HTML 后才出现。因此测试应检查最终构建结果：

- 中文和英文首页确实生成。
- 双语文章详情页位于预期目录。
- 根入口和旧入口包含当前 base 下的中文首页链接。
- 自定义域名构建中不再出现 `/brief/`。
- 页面 canonical URL 使用配置的 `SITE_URL`。

这种测试不需要解析整个 DOM。读取几个关键 HTML 文件并断言目标路径，已经足以覆盖这次故障的主要回归面。

排查顺序

遇到“本地正常、Pages 上 404”时，可以按下面的顺序缩小范围：

- 确认浏览器实际请求的 URL，是 origin 错了还是路径前缀错了。
- 对照公开地址，明确 `site` 和 `base` 的期望值。
- 搜索源码中的根绝对路径，确认它们是否绕过统一路径函数。
- 单独检查根入口、旧页面和脚本重定向。
- 用与 CI 相同的环境变量构建，再检查 `dist` 中的链接和 canonical。

部署路径不是某一个组件的局部配置。把公开 origin、部署前缀、链接生成和构建验证作为一套契约处理，站点从仓库子路径迁移到自定义域名时就会平稳得多。