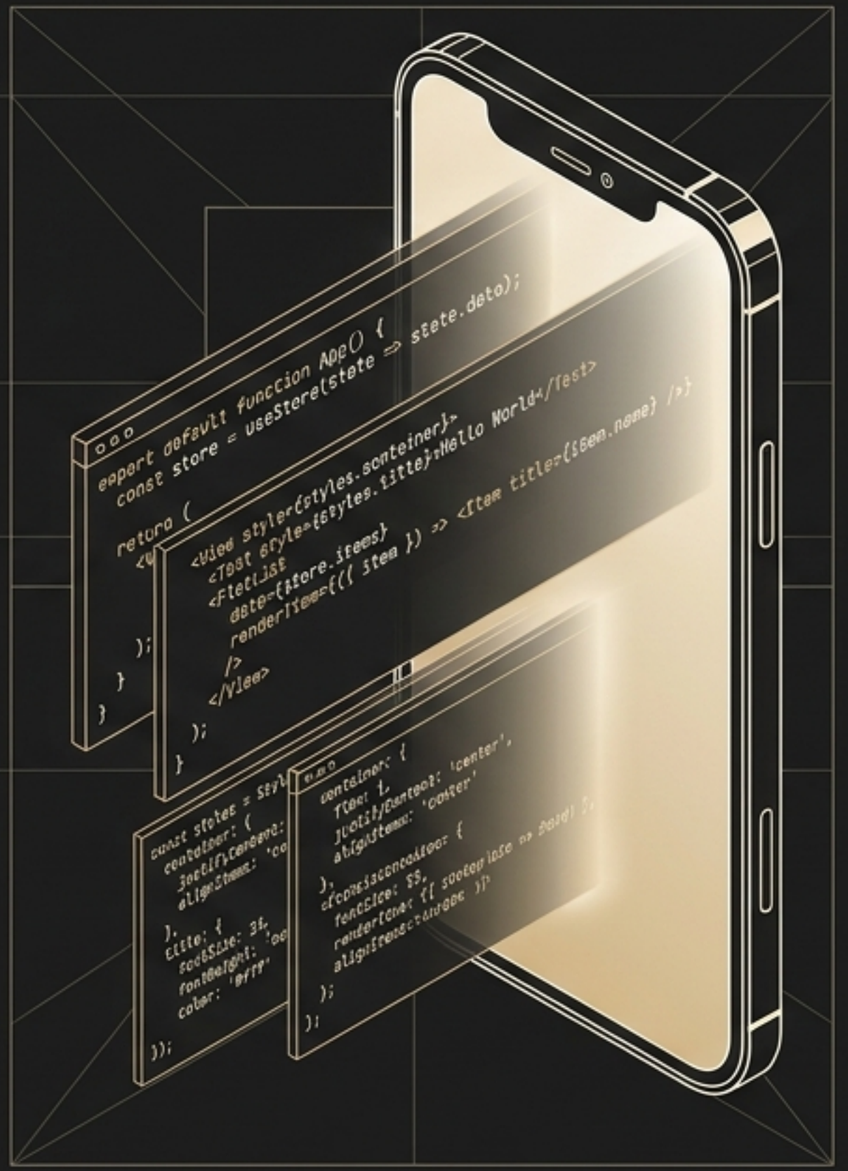
>_ SYSTEM_LOG: ÉLAN 开发手记 / Part 4

一套代码搬上手机

Next.js 到 React Native（Expo）的真实解剖

Target: iOS / Android
Tech: TypeScript, Zustand, Expo SDK 55



手机不是加分项，是必选项

Web / PWA 的断层

[ERROR_ALBUM]
网页端相册访问受限，上传体验割裂。

[ERROR_SAVE]
无法一键保存，只能绕路长按保存图片。

[ERROR_SHARE]
系统原生分享菜单不存在。

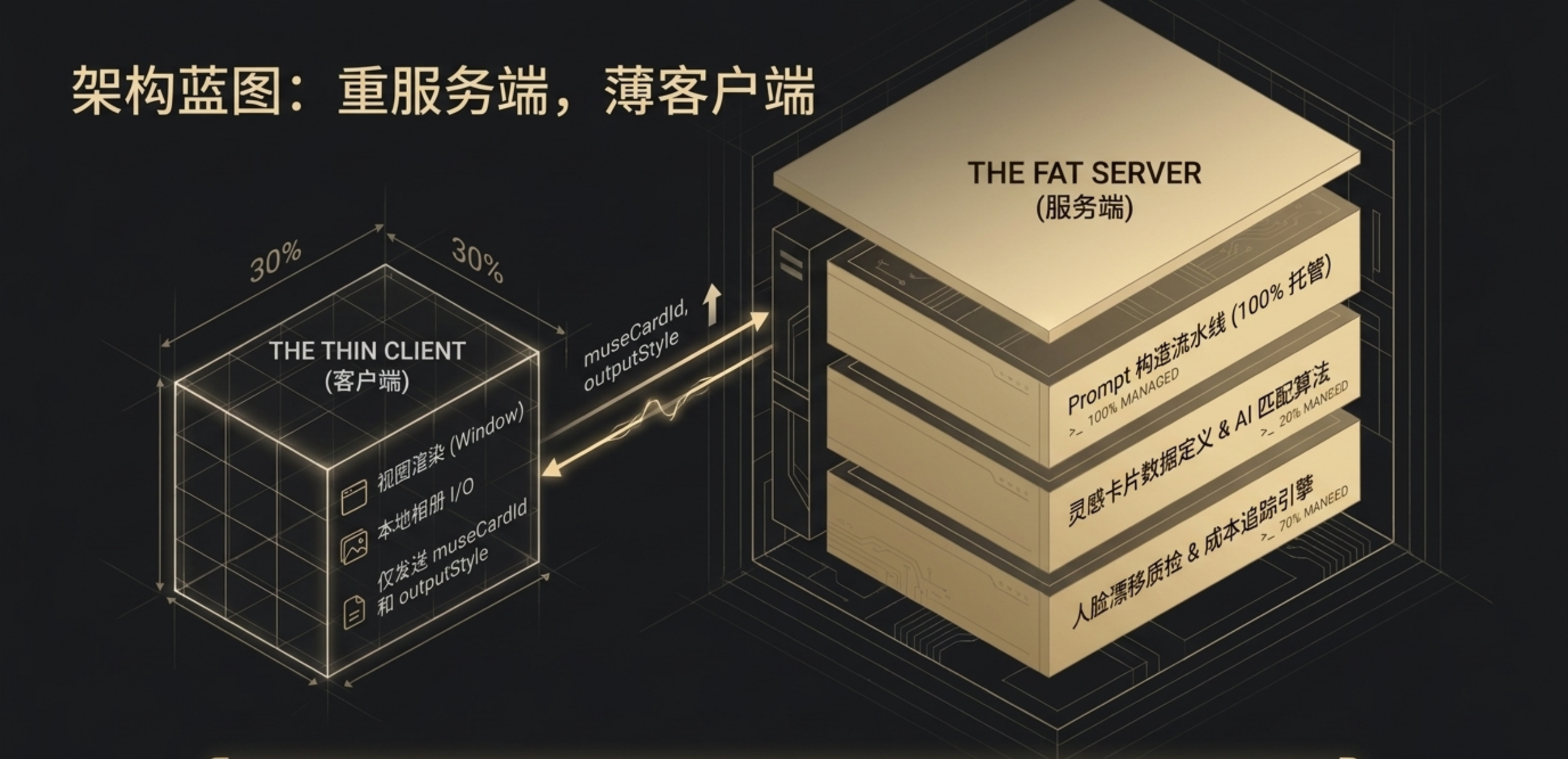
结论：每多一步网页跳转，转化率就掉一截。PWA 体验差到让核心用户放弃。

架构抉择：三条路，选哪条？

	性能 / 生态	现有 TS 逻辑复用	维护成本
Flutter	[OK] 成熟	[FAIL] 0% (需用 Dart 另起炉灶)	[WARN] 高
原生 (Swift/Kotlin)	[OK] 最佳	[FAIL] 0%	[FAIL] 极高 (一人维护三套代码)
RN + Expo SDK 55	[OK] 极简内置	[OK] 同语言、同状态管理库 (Zustand)	[OK] 业务复用，仅重写 UI

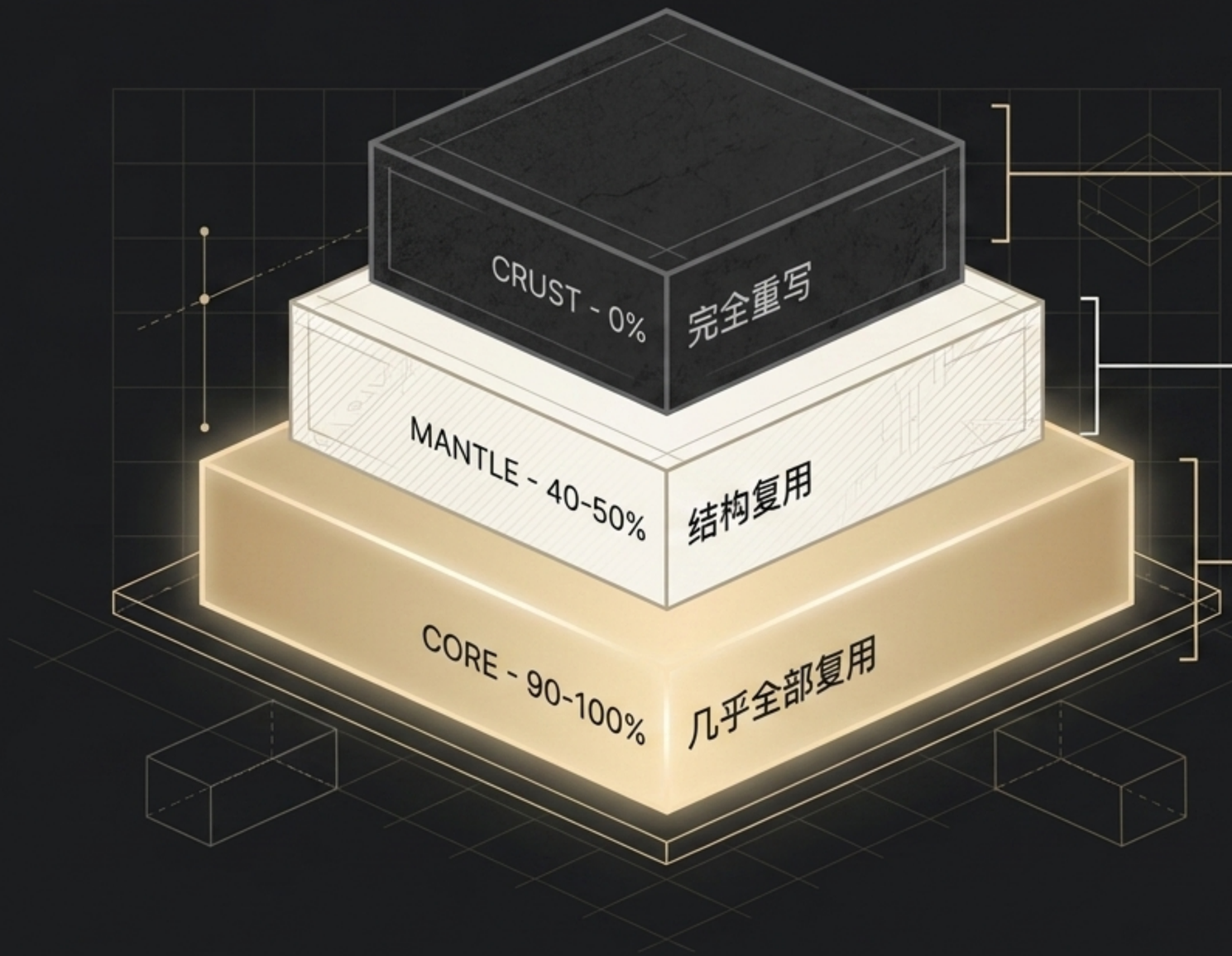
> _ 放弃标准 Monorepo：对于单一消费者模型，手动同步类型和 store 是最实用的妥协。

架构蓝图：重服务端，薄客户端



核心收获：把智能放在服务端，客户端只是一个选照片、发 API、看结果的窗口。

代码复用的真实解剖



样式与鉴权： Tailwind 完全抛弃，重写为 StyleSheet。服务端 Cookie 改为 header 鉴权。

组件骨架： UI 拆分方式、Props 设计保留，但 JSX 零共享。路由概念对齐，但变为 Expo Router。

逻辑与契约： Prompt 构造全盘复用，API 接口契约（~95% 镜像），Zustand Store 模式（~85% 概念一致）。

UX 翻译：同样的流程，不同的手感

Web		Native
拖拽上传区 (Drag & Drop)	动作 1：选照片	调取原生相册，紧凑预览正脸。
响应式网格 + Hover 显示场景描述。	动作 2：选灵感卡	可滚动网格，长按从底部弹出 Bottom Sheet 完整预览。
AI 分析匹配百分比。	动作 3：灵感匹配	AI 自动选中，卡片折叠成一行摘要（减少关键点击）。

结果页进化：移动端新增骨架屏动画，以及本地文件系统自动缓存。

暗色模式：只花了两小时的白送福利

Light Mode Component:
暖奶油底色 + 暖炭灰文字

Brand Anchor:
香槟金强调色

Dark Mode Component:
深棕底色 + 浅奶油文字

```
[ 1. Expo SDK userInterfaceStyle: automatic ]
[ 2. React Navigation ThemeProvider ]
[ 2. React Navigation ThemeProvider ]
[ 3. useTheme() Hook 直接贴入 style ]
```

零条件 className，零 CSS 变量。移动端的显式样式控制，比网页端使用 Tailwind 更直观、更迅速。

最大的技术暗礁：iOS SSE 流式传输陷阱

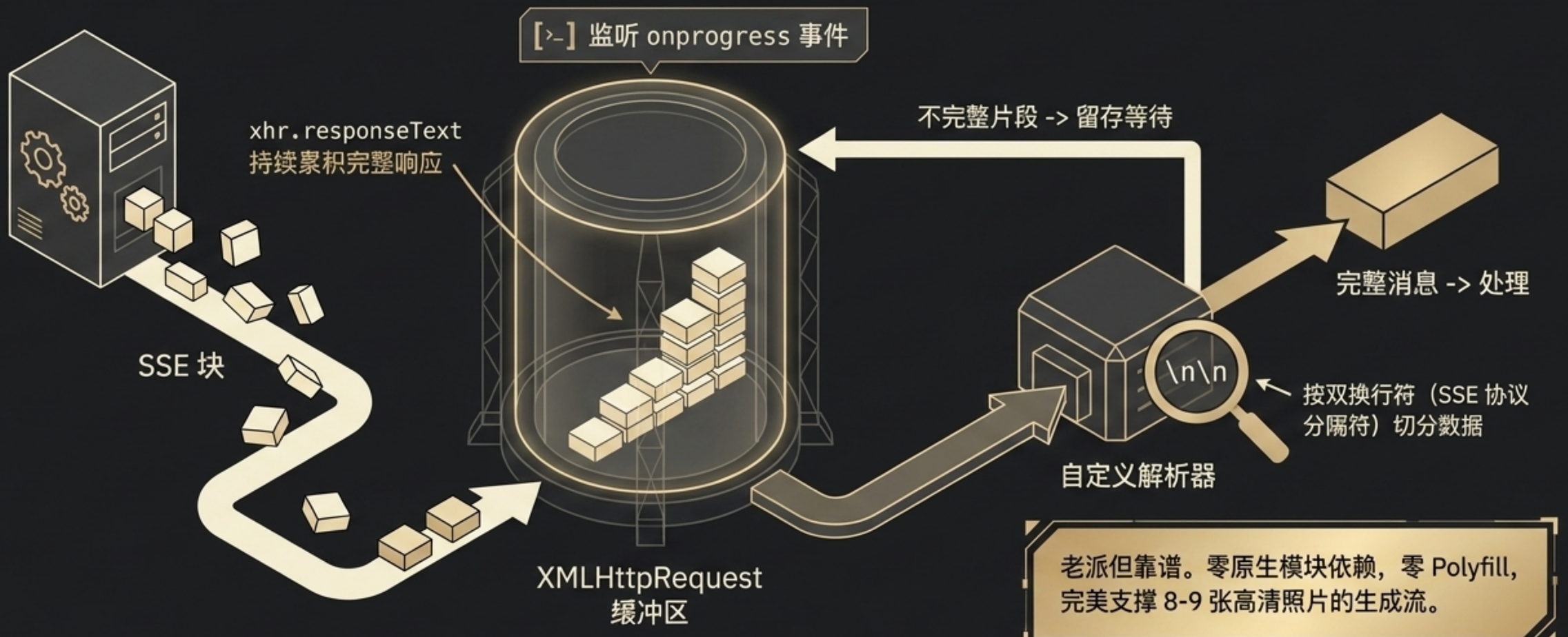
CONTEXT: ÉLAN 生成依赖 SSE 逐张发送进度事件。

Dead Ends

> 🚫 react-native-sse: 仅支持 GET，无法满足 POST 需求。
> 🚫 react-native-fetch-api: 依赖原生模块补丁，与 Expo 新架构冲突。

越狱方案：用最老的 API 解最现代的局

XMLHttpRequest（XHR）渐进式累积



老派但靠谱。零原生模块依赖，零 Polyfill，完美支撑 8-9 张高清照片的生成流。

Expo 体验冰火两重天：高光与纸割痛

比想象中简单（高光）

[+] 原生 API 开箱即用：
expo-image-picker 等。零 Xcode 工程调试，一个下午搞定照片选存。

[+] Zustand 无缝迁移：
API 完全一致，`create()` 模式原封不动运行。

[+] EAS Build 云端编译：
推代码 -> 跑命令 -> 15分钟出包。全程无需开启本地 IDE。

比想象中繁琐（纸割痛）

[-] SSE 限制：
耗费一天的试错与重写。

[-] 触摸目标规范：
Apple HIG 要求最小 44px。网页端间距必须逐个组件手动重调。

[-] 安全区域（Safe Area）：
刘海、状态栏会吞噬布局。必须频繁介入 insets，稍有遗忘就有遮挡 Bug。

最终账本：代码量对决与复盘

~12,000 行
网页版代码（Next.js）

~4,200 行
手机版代码（Expo/RN）

35%
手机代码占比（2周交付 MVP）

Developer Notebook

如果今天重新开始...

>_ RULE_01: 先做移动端。虽然网页迭代快，但移动端才是产品的最终形态（相册集成、系统分享）。应先上 Expo，再配管理后台。

>_ RULE_02: 第一天就提取共享类型包。手动同步类型初期勉强能忍，但长远看，`workspace` 拆分在第二周就能赚回成本。

客户端只是窗口，把智能与复杂度全部留在服务端。新功能上线，双端同时生效。