本地优先vs云端原生：从个人电脑到AI的张力

刘果 · 2026-09-27 · 中文

演讲与问答全文

演讲与问答可读版逐字稿。2026 年 9 月 29 日二次校订：对照 Soniox 转录、Zoom 字幕和演示材料，整理重复起句、断句及术语，保留发言顺序、例子、限定条件与讲者观点。时间对应交付视频。部分疑点音频另经 Soniox 复核。仍无法确认的表述已标注；未逐句人工回听。

开场：选择本地优先

00:00:00 · 刘果

对，我觉得好。我确实通过很多去中心化、本地优先的工具，认识了很多很好的朋友。我有一个假设：即使不考虑效率或其他收益，选择这类工具也是一种主动的选择。这种选择背后有审美，也有所谓技术人的是非观。

00:00:26 · 刘果

我觉得，对这个方向感兴趣的人，常常会在效率、收益之外有更多考量。所以，可能也比较符合我们通常对“好人”的定义。

00:00:39 · 主持人

是。这次组建社区，就感觉遇到了一些稀奇古怪、但在技术上有社会意识的朋友。

00:00:56 · 刘果

是什么意识？

00:00:58 · 主持人

在技术上有社会的意识。

00:01:00 · 刘果

是，有一种责任感。

00:01:04 · 主持人

对，我觉得国内这一点挺难得的。很多人纯粹做工程，但其实也选择了自己的立场，就是选择大公司。今天则是选择 AI 公司，觉得 AI 开源就很正义、很正确。然后说美国坏，美国也这么说中国坏。



演示页 1 · 00:01:33

00:01:31 · 主持人

那……

00:01:32 · 刘果

对。

00:01:32 · 主持人

就做出了一个对人很不友好的选择。

云端与本地的张力

00:01:37 · 刘果

是的。

00:01:40 · 主持人

所以，做个人计算、关注本地优先或者去中心化的，算是一批能动性比较强的人。我就不多说了。

00:01:50 · 刘果

能动性。

00:01:52 · 主持人

那……

00:01:52 · 刘果

是的。

00:01:53・主持人

对，我不多说了。

00:01:55・刘果

好的。我觉得刚才那个主题确实很有意思。我们都知道，技术对社会的影响非常大，可能是现在形塑整个社会最大的因子。但是，作为搞技术的人，不管自称码农、软件工程师还是设计师，我们这些做软件、做产品的人，常常在大厂打工打惯了，就把自己放在一个被动的位置。

00:02:20・刘果

好像大海需要什么、潮流往哪里走，我们就去做什么。但这样就没有人在选择，没有人在发挥能动性、尝试思考这个世界该是什么样子。其实，唯一能去想这件事的人，只有那些做技术、做产品的人，因为技术产品正在如此深刻地改变世界。所以，我很赞同刚才主持人的观点。

00:02:47・刘果

本地优先，或者之前比较流行的分布式概念，其实有很多相近的理念。我一直觉得这群人很宝贵：他们是少数在做产品、做技术，有能力改变世界，并且还在想怎样让世界变得更好的人。他们知道能动性仍在自己手里，没有把它交出去，没有把自己完全变成一个工具。这确实是很难得的一群人。那我就开始今天准备的材料。这几天，大家可能听到了很多前沿技术和进展，我来讲一些个人计算机和互联网这段短暂历史里的片段。

00:03:40・刘果

这不是一段完整的历史，而是一些我比较喜欢的片段，也是我最近在想的问题。我觉得讨论会比较好玩，所以大家随时可以打断我，或者在聊天里说。很多历史大家可能都知道，知道其中的片段和野史。一起把它们串起来，就会看到一些骨架，也会看到刚才说的那些问题：技术，尤其是底层结构，怎样影响社会。我这里没有逐一展开社会影响，想先集中在技术和历史的讨论上，但有很多可以延伸的地方，大家可以随时点评。我想讲的主题是“云端与本地的张力”。我自己非常喜欢本地优先，

00:04:34・刘果

非常喜欢去中心化、软件完全开源，让用户拥有最大的能动性，也就是 *agency*，以及选择权。这是很多做软件的人的梦想和审美。但它未必与更大的历史潮流相符。比如逐利：常常需要建立一个更大的中心，让更多人依赖你，才会获得更多收入；能够垄断市场，价格才会更高。另一方面，很多技术突破确实需要巨大的中心，集中大量资源才能实现。我好奇的就是：这两者到底如何相伴相生？所以，我会讲几个切片。

00:05:26・刘果

一个是个人计算机兴起时很有意思的转折；另一个是在互联网上发布内容这件事，从最早发布网页到后来各种平台出现，其中有好几次云端与本地之间的来回摆动；最后是 AI，它怎样体现这种张力。

00:05:51・刘果

我会提出一些看待这种张力、这种动态的假设。最后，也会讲讲现在做青苔这个本地优先的网站发布工具时，我怎样设计它、回应平时的思考。关于我自己：我做了很多年不同形式的发布工具，也做过线上杂志，都是通过不同方式帮助人们，或者把人们组织起来，发布好的内容，让更多人看到。我最喜欢的还是产品这一部分：怎样把发布的管道、渠道，把内容的容器做好。我自己是开发者，喜欢可靠、让人自己做主的工具。在工作方式上，我喜欢小团队，



演示页 2 · 00:06:33

00:06:46 · 刘果

希望能够独立工作。所以我经常想，现在的 AI 浪潮，以及此前几次技术演进，常常能看到大公司的推动；那我作为个人或小团队，能做什么，让其他个人和小团队受益？这是我思考问题、做产品时的一个出发点。

00:07:15 · 刘果

这也让我时不时翻一翻短短几十年的信息技术史，甚至更早的技术史。有时把它当作娱乐，想一想从中生出的中心化与去中心化、云端与本地之间的张力。希望这些片段也能让大家觉得有意思。

个人计算机的许诺



演示页 3 · 00:08:03

00:07:38 · 刘果

第一个切片是个人计算机的兴起，从六七十年代讲起。我觉得其中很多地方已经预示了后来的演变。在个人计算机之前，是大型机，也就是 mainframe。这是计算机本身兴起的阶段。

00:08:03 · 刘果

当时很大程度上由美国政府支持，背后是与苏联的冷战竞争。苏联抢先把卫星送上天后，美国突然慌了，觉得自己赶不上，于是投入很多资金和人力到实验室，发展计算机技术。当时投入的方向是 mainframe。现在叫大型机，说白了就是一个服务器，大家通过 terminal，也就是终端，连上去共享。它非常贵。右边这台是 1952 年的早期计算机之一，需要一整个房间、专业团队来维护，持续消耗很多成本。大型机阶段持续了很长时间，当时没有多少人在考虑个人计算机。但回头看，大型机时代已经有了很多关键突破。



演示页 4 · 00:09:33

00:09:00 · 刘果

比如 Engelbart 的演示，后来被称为“所有演示之母”，The Mother of All Demos。它是历史上第一个计算机演示、产品演示。如果去看 1968 年的录像，现在仍会感到震撼：我们今天使用的很多交互雏形，那时已经出现了。

00:09:29 · 刘果

它展示不同视图、地图、远程协作，以及怎样在文档里工作。文档、文件夹、视频会议等很多想法，那时已经在使用。人们第一次看见相互关联的文档、鼠标，以及键盘与显示屏的配合。

00:09:54 · 刘果

这些后来都成为了基础。但这些重大突破发生在大型机时期，发生在高度中心化、还没有个人计算机的时候。计算机系统设计也一样：time-sharing，也就是分时，让不同终端共享一个系统的时间；后来个人电脑上的 multitasking，也就是多任务，让不同应用分配 CPU 时间，其实是一脉相承的系统。早期电子邮件也是从大型机上不同用户互相留便条、互发消息发展出来的。包括这些在内，很多技术都是在大型机时期、在大型国家实验室里产生的。



演示页 5 · 00:10:33

00:10:46 · 刘果

当时也已经开始出现一些社区化的使用，虽然背后仍是大型组织。例如，人们在唱片店里共享一个终端，把信息从账号发到大型机上，互相询问哪里找唱片、哪里有好吃的东西。



演示页 6 · 00:11:03

00:11:12 · 刘果

这时已经能感到一种压力：只有一个终端，但所有人都想用，似乎更自然的方式是每个人都有一台电脑。但站在当时看，这个想法非常奇怪。至少在 PARC 于 1973 年真正尝试为个人做一台电脑时，很多人都反对，

00:11:38 · 刘果

觉得计算机需要那么大的空间和人力物力，缩成桌上的小电脑，算力又弱，能拿来做什么？PARC 背后的施乐通过复印机、打印机赚了很多钱，就想试试看。你可以看到，当时的设备非常原始，又大又慢，一开始能用的地方很少。反对也有道理，因为确实不太用得起来。但它有一个最好的地方：一旦走上个人计算机这条路，你就能在物理层面改造它。可以扩展内存，自己组装过电脑的人都知道，这很好玩，因为你可以任意改造它，



演示页 7 · 00:12:03



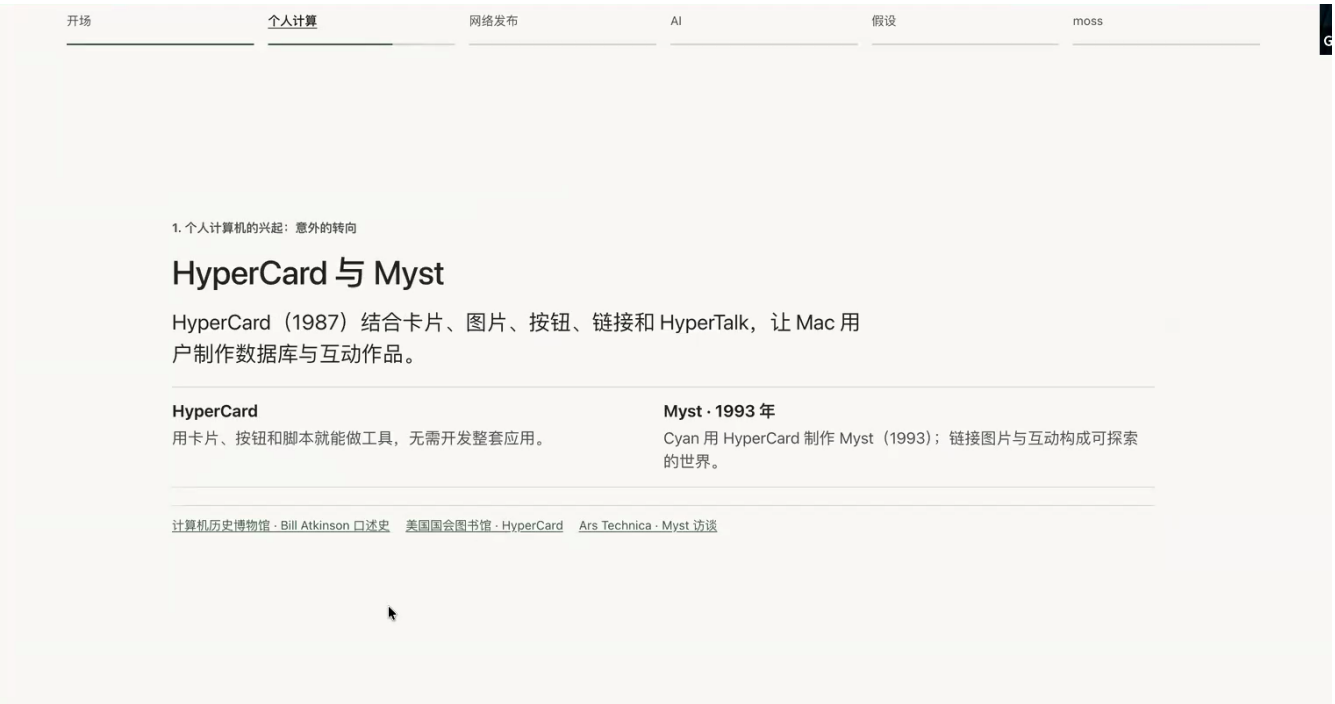
演示页 8 · 00:12:19

00:12:30 · 刘果

从硬件一直到软件。这时出现的创新，与大型机时代非常不同。大型机时代的创新更根本、更偏架构，让后来的很多东西成为可能，比如性能、多线程，以及电子邮件底层的一些思维模型。到了个人电脑，哪怕性能弱、界面糟糕，也会出现完全不同的东西。比如 1979 年的 VisiCalc，当时在终端界面里使用，却已经很像后来的 Excel。大型机时期，其实不会出现这么复杂、专门给个人使用的软件。



演示页 9 · 00:13:03



演示页 10 · 00:13:33

00:13:24 · 刘果

对计算机历史感兴趣的朋友，可能还记得 HyperCard 和 Myst。它们的设计很有意思，也能让人联想到 2000 年互联网泡沫之后的平台演变。比如 HyperCard 让你通过卡片、按钮和脚本做工具，很像后来的 no-code development，也就是不用写代码就能开发。

00:13:50 · 刘果

到现在的 vibe coding，也有相似之处：不断降低开发应用的门槛，让人不必直接写代码就能做应用。Myst 则基于 HyperCard。你也能从中看到后来社交媒体、社交平台的一些雏形，让人们在内容库、内容网络之间协作，相互写作、阅读、认识。这些很早，在 1987 年就出现了。你会看到，一旦计算变得个人化，创新速度非常快，方向也与最初不同。再往后，就是大家更熟悉的云端化。我之前和主持人聊过：今天说“本地优先”，其实有一个前提，就是我们已经经历了云端化。



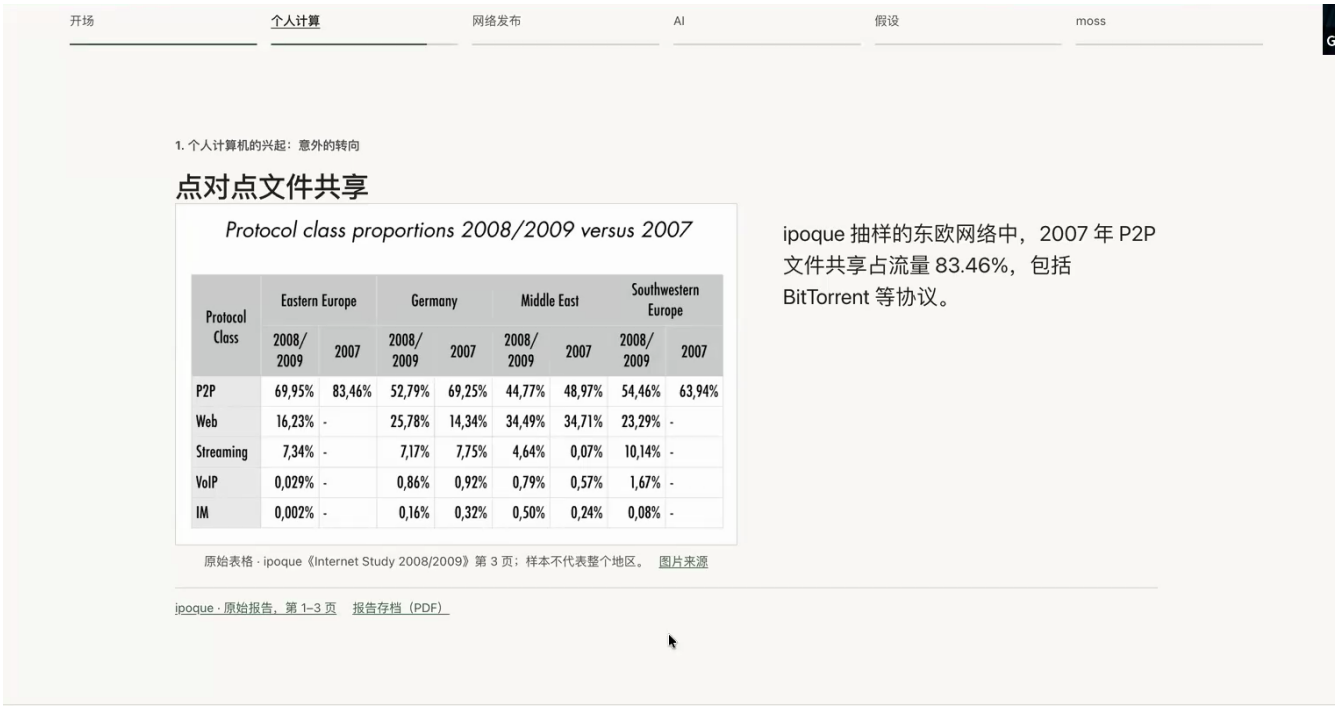
演示页 11 · 00:14:33

00:14:46 · 刘果

云端化大约发生在 2000 年以后，可能是 2005、2006 年之后，接近互联网泡沫那段时期。在此之前，所有东西天然都是 local-first。整个个人计算机革命，可以看作一次本地优先的革命：把计算资源从大型机搬到个人电脑，搬到桌面上。看当时的录像，人们讲的梦想，比如让每个人都拥有一座像亚历山大图书馆一样的人类资源库，其实都是在把信息和 agency，也就是主动性，放到本地，交给使用它的个人，而不是机构、政府或团队。

00:15:37 · 刘果

这是一件很个人主义的事。但我们现在说的 local-first，是经历重新转向服务器之后的本地优先。这个转向发生得很晚，是互联网兴起之后的事，很大一部分原因是手机普及，因为手机算力很弱。更早一些，一直到 2007、2008、2009 年，不管是 BitTorrent、eMule，还是 eDonkey，年纪稍长的朋友可能都用过，我自己当时也很喜欢。P2P 流量最高时占整个互联网流量的 80% 以上，它才是当时的主流。

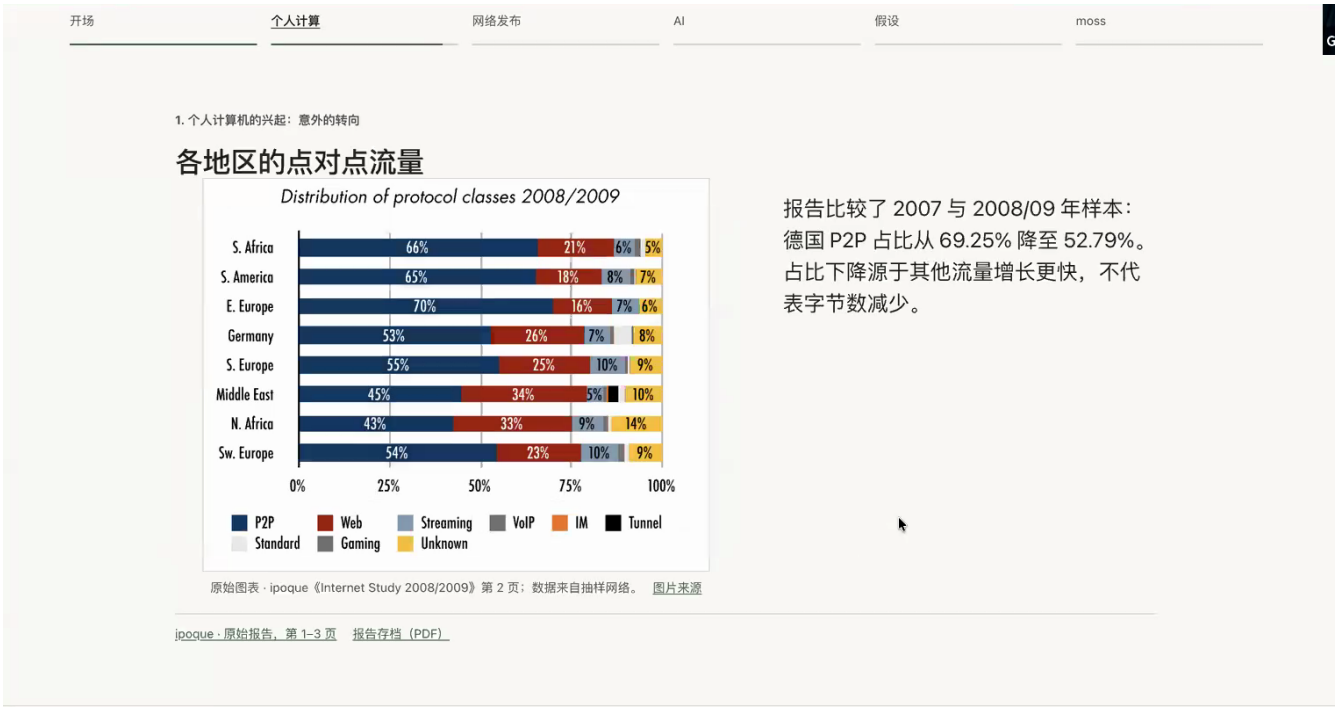


演示页 12 · 00:16:03

00:16:30 · 刘果

那时，主流并不是现在这种中心化的客户端到服务器的流量。但后来发生了转向。看当时世界各地的对比，也很明显：P2P 流量才是那时的互联网。

云端化与互联网流量



演示页 13 · 00:16:37

00:16:49 · 刘果

很快，随着手机兴起，很大一部分计算转向了服务器。现在互联网的计算是高度混合的。做技术的人常常需要考虑怎样拆分：哪些计算放本地，手机能承载多少，哪些放服务器？像我这样的人，会尽量放手机、放客户端，少放服务器；也有人采取相反思路，更多放服务器。不管怎样，基础已经成熟，更重要的是开发者有了选择：做本地优先架构，还是服务器优先架构，都能做到，没有技术上的限制。这就回到了开头说的，它是一种选择，



演示页 14 · 00:16:49

00:17:43 · 刘果

而对做工具的人来说，有选择也意味着责任。后来还有小红书这样的各种平台。这就引出另一个历史切片：怎样在互联网上发布内容。在这些平台兴起之前，先有互联网本身，而它最初就是用来发布内容的。

互联网与个人网站



演示页 15 · 00:17:51

00:18:11 · 刘果

发布内容和个人计算机一样，经历了中心化与去中心化、云端与本地之间的来回摆动。最早它就是一个混合体。Tim Berners-Lee 提出万维网、HTML 标准，和其他人做出早期浏览器雏形时，主要支持来自一个很大的中心化机构，是 CERN，欧洲原子能机构。［编注：CERN 的正式名称为欧洲核子研究组织（European Organization for Nuclear Research）。此处保留口述称谓。］那是在 1989 到 1990 年。但 Tim Berners-Lee 的目的又非常个人化：让具体的科学家能够自己写 HTML、写网页，发给彼此。

00:19:09 · 刘果

服务器也放在各自本地的计算机上。所以一开始就是很有意思的混合：有大机构和资源支持，目标却非常个人化，从本地搭服务器、写 HTML。再比如那个被一些人称为第一个博客、也有人称为第一个个人网站的例子，看它怎样做出来也很有意思。作者是一个本科生，可能是大一或大二。他一开始在自己的文件夹里手写 HTML，然后本地启动一个服务器，开放出来。后来因为本地 IP 经常变化，才迁到学校服务器。从这些问题里，就能看到这个过程怎样摆动。最开始，发布内容本来就是很本地优先的事，



演示页 16 · 00:19:33

00:20:07 · 刘果

从本地出发，在本地写作。这里插入一个我觉得好玩的转折：Markdown 的产生。Aaron Swartz 写了最早的版本，当时目的很单纯。［编注：Markdown 由 John Gruber 创建；其官方介绍感谢 Aaron Swartz 对语法设计提供的反馈。这里保留讲者对起源的口述。］他在博文里说，自己是人，不是机器，为什么每天都要手写 HTML？他想用最适合人写作的方式，也就是纯文本，再用最简单的标记表示标题、强调、斜体、引用，于是有了 Markdown。它的设计是面向人的写作。我一直喜欢、一直使用 Markdown，但以前不敢在面向普通人的软件里采用，因为担心他们不习惯。没想到 AI 出现之后，



演示页 17 · 00:20:33

00:21:03 · 刘果

所有人都不得不用 Markdown 了。这也有道理：用纯文本写出带结构的内容，Markdown 已经是很小的表达单元，至少总得有一种方式标记标题。它从那时发展到现在，成为一种写作规范。回到发布这条线，Markdown 也很重要，后面很多东西都基于它。



演示页 18 · 00:21:33

00:21:30 · 刘果

一个很大的转折点是 WordPress。它在 2003 年出现，想让不懂技术、不会写 HTML 的人，也能发布自己的网站和独立内容。它做了一个取舍：采用 CMS，也就是内容管理系统。后来中国团队也做过一个类似的、用 PHP 写的版本，名字我现在想不起来。后来出现了很多这样的东西。WordPress 提出的想法是，让普通人通过直观的图形界面操作；但它预设整个软件管理运行在服务器上，而且是动态的。每次有人访问网页，

00:22:21 · 刘果

它就现场生成网页，而不是像最早那样预先写好静态页面，直接快速提供出去。这增加了成本和复杂性。更重要的是，它采用服务器优先架构：普通人发布容易了，托管 WordPress 的服务器却不容易。后来，托管 WordPress 反而成了一门好生意。

00:22:48 · 刘果

这是一个转折。此后的内容发布工具和渠道，一路向中心化、云端化发展。比如 Squarespace、Wix 这些建站工具，都是闭源的。WordPress 至少还有 self-host，也就是自己托管的选项；这些闭源产品则只能整套使用它们的云服务。

00:23:16 · 刘果

更不用说熟悉的社交网络平台，比如小红书、微信。严格来说，我觉得它们甚至都不是互联网，因为它们不互联，而是在筑墙，故意让你无法相互连接。小红书不想让流量出去；Instagram 限制主页链接，只能放一个，就是不想让你互相链接。它们和最初的互联网在往相反方向走，而且越来越快。倒过来看，WordPress 仍坚定站在互联网这一边，只是已经开始转向服务器。但开发者不会甘心把自己关在平台里，肯定会想办法。很多人连 WordPress 都已经用得不太爽了，



演示页 19 · 00:24:03

00:24:11 · 刘果

更不会所有人都甘于使用 Wix、Squarespace 或小红书。2008 年，GitHub 联合创始人 Tom Preston-Werner，他自己也是写作者，很不喜欢 WordPress：为什么要用一个又大、又丑、又慢的 CMS？

00:24:37 · 刘果

为什么不能像当年写 HTML 那样，简单地写个网站？这时 Markdown 已经流行起来了。他想，最简单的办法，不就是在自己的文件夹里写 Markdown，再写一个脚本把它转成网页吗？这样写起来舒服，也不会被绑定到某个专门的软件，或者某套特定规范上。后来这类工具被称为 Static Site Generator，也就是静态网站生成器。懂技术的人都知道，它很常见，几乎每种语言都有两三个。Jekyll 是第一个。这类工具做的事很直接：把 Markdown、模板和文件夹结构，在本地建构成 HTML 资源，

00:25:31 · 刘果

然后部署到任何地方。它做的事情简单，但正因为通用，所以很好用：内容在 Markdown、模板和文件夹结构里，可以换不同的生成器。另一方面，它高度 hackable，因为整个建构过程在本地。就像最早的个人计算机，硬件放在本地，你就能改、能魔改。静态网站生成器也一样，于是出现了很多模板和设计。它还能部署到任何地方，因为就是本地生成的一组 HTML 和相关文件，打包上传即可，非常成熟、便宜。Jekyll 之后，这条技术路线发展得很快。



演示页 20 · 00:26:33

00:26:25 · 刘果

做过前端的人，应该用过后来受它影响的框架。十几年前有个很有名的提法叫 Jamstack，就是 JavaScript、Ajax、Markdown 形成的技术栈。［编注：JAM 原指 JavaScript、APIs、Markup；此处保留讲者的口述展开。］像 Next.js，以及 Vercel、Netlify 这些公司所依托的前端框架，都广义地被称为 Jamstack。

00:26:54 · 刘果

它们都是从 Jekyll 这条路发展出来的。从开发者角度看，最快的方式是在本地建构一个静态网站；需要服务器资源、需要 API 时，再接入就好。需要动态页面、登录、身份，都可以加，再复杂也可以。但本质上仍是一个在本地建构的静态网站框架。后来当然越做越复杂，

00:27:21 · 刘果

比如 Next.js，你常常把建构搬到服务器上，做 server-side rendering，也就是服务端渲染，再交到前端，或者做 single-page application，也就是单页应用。这些现代框架大家很熟悉，但来源仍是那个想法：作为 hacker，我怎样写东西、做网站？在本地写、本地建构，再部署出去。

00:27:46 · 刘果

后来做得很好的 Astro、Next.js，仍保留这个特性：第一步是静态网页生成，生成后再部署。它也不会把你绑定到某个具体框架，因为内容仍在具体的文件和 Markdown 里，迁移很容易。但有个问题：这条路线上的技术都不是给普通用户用的。这和 WordPress 有一个本质区别。



演示页 21 · 00:28:59

00:28:17 · 刘果

WordPress，以及那个我一时想不起名字的中文类似项目，都是面向普通用户。静态网站生成器却拥有我觉得最好的一组特性：本地优先、基于文件，不把你绑定到具体设计和框架上，但都是给开发者的。

00:28:43 · 刘果

它不像 WordPress、Squarespace、Wix 或小红书，面向普通人。静态网站生成变成了开发者手里的利器。比如 GitHub 上有一个很长的中文独立博客列表，现在仍有很多人写独立网站，隔几天就能看到新的提交。当然，列表放在 GitHub 上也会影响样本，但看看其中的网站，几乎全是技术类，都是懂技术的人在写。即使已经有 WordPress、Wix、Squarespace，即使 AI 技术这么发达，建网站仍高度局限于技术人员。

00:29:39 · 刘果

对我来说，这既是机会，也是遗憾：它是最有创造力、最自由、最能体现个人能动性的媒介之一，却基本上只有技术人员能用。第三个切片就是大家熟悉的 AI。大家有任何想法都可以随时提，我一个人讲，也不知道哪些地方对大家更有意思，可以多听听大家的想法。

AI 与本地工作环境



演示页 22 · 00:30:03

00:30:09 · 主持人

我还是想围绕刚才的 Markdown 说一下。我最近也在做类似应用，遇到一个问题：开发者很擅长用符号生成东西。Markdown 严格来说，就是用简单的符号作标记，生成一个视觉上可以交互的东西。

00:30:32 · 主持人

但有一个假设是：我妈不会用 Markdown，她一定是用手指点点、拖拽。所以……

00:30:46 · 刘果

是的。

00:30:46 · 主持人

这就导致现在做前端，或者用 AI 做前端，出现了一个分叉：一边是符号标记的路线，一条一条地写；另一边是直接许愿、炼丹。

00:31:08 · 刘果

嗯。

00:31:09・主持人

我不知道具体区别是什么，但明显能感觉到，产品思路、和用户打交道的方式不同。有些用户喜欢通过符号交互，最普通的用户则直接许愿。

00:31:28・刘果

是的，这里面有很多相互矛盾的需求。一类是非技术用户，不想弄清楚东西怎样工作；另一类像 hacker、极客，买了一辆车就想掀开引擎盖，看看引擎怎样运转。这是很不一样的两类人，张力一直存在。但所有做产品的人，都有一个很理想化的目标，

00:31:57・刘果

我自己也非常想做到：高天花板、低地板。天花板高，意味着想做什么都可以，拆引擎、改装都行；地板低，意味着不在意这些的人拿起来就能用。这两点常常互相冲突。具体到 Markdown，就是能不能既基于纯文本，又标准化、能在不同环境中通用，同时让我妈也能用？这是很大的设计挑战。聊天里也有朋友提到，很多编辑器想做的，就是给 Markdown 配一个任何人都能用的界面：既是 Markdown，又能拖拽操作。我自己也尝试过，但确实很难。我觉得一个好的例子是福特 T 型车，

00:32:53・刘果

也就是 Model T，历史上非常流行的车。它结实、好用，同时又特别适合改装。人们不断改装它，也形成了一整套文化。福特会很细致地告诉你怎样改装、怎样修理。

00:33:20・刘果

随着 T 型车成长起来的美国年轻人，以及其他进口这种车的地方的年轻人，很多都成了更好的机修工。这一直是我很喜欢的例子。作为技术人员，看到 AI、看到 Obsidian，我也更有信心。我有很多非技术朋友会用 Obsidian，这让我觉得 Markdown 是很好的基础，因为它可移植性强，能和不同软件、协议通用。很多人就在死磕：能不能坚持 Markdown，同时让所有人都用得上？就像设计 T 型车，坚持做一辆可以自己改装的车，又让所有人都能用。

00:34:18・刘果

我看到聊天里有人说，用模型把自然语言自动整理成 Markdown。我觉得这件事永远很难彻底解决，因为哪些是标题、哪些是正文，只有写作者自己知道。我写东西时经常列大量 bullet points，也就是列表，层级关系只有我知道，而且层级很重要：哪个标题属于哪个标题。我只能通过 Markdown，或者至少用空格，把关系标出来。就像纸上的草稿，必须依赖空间信息和符号，才能完整表达。这已经是表达思维和想法的一种很简洁的方式。如果反过来，只把纯文字喂给 AI，

00:35:12・刘果

让它去猜，逻辑上就已经丢失了一部分信息，而那些信息只在我心里。所以类似 Markdown 的东西仍有必要。我想坚持的是：能不能把 Markdown 编辑器做得让我妈也能用？我现在真的在教她用，就是想看看编辑器能不能做得足够好。

00:35:40・刘果

大家可以随时开麦聊。

00:35:46・主持人

刚才这个问题里，AI 能自动生成很多符号，但似乎涉及一个人机交互的本质问题：一边是用户意图，另一边是计算机语言。

00:36:04・刘果

是的。

00:36:04・主持人

中间需要翻译。你要么做复杂的工作，精准地为用户定制；要么让用户多做一点语义上的工作，把意图说清楚，减少后面的复杂性。

00:36:19・刘果

是，这涉及很多本质问题。用户意图是绕不开的，不管用图形界面还是其他交互方式，都得表达出来。对计算机而言，最简单、最开放的表达方式仍是字符串。通过特定编辑器，让字符串形成适合人思考的空间结构和符号。计算机处理字符串也是绕不开的底层逻辑。LLM 刚出现时，大家还不知道能让它做什么；现在发现，它最能解决的仍是写软件、写代码，因为软件本身就是一个由计算机处理的长字符串，LLM 也在处理字符串，背后是同一套逻辑。

00:37:14・刘果

就是图灵机，一个处理字符串的图灵机。不管把它做得多复杂，不管通过训练怎样得到软件，最后仍绕不开这个架构。所以，人要表达一种最简洁、能与计算机对接的意图，仍然是通过字符串。

00:37:34・观众

我最近也在尝试优化网站。几个月前，Claude 出了一个设计功能，当时还挺大方，design 的额度和 chat、code 分开计算。我感觉它用了不少设计数据，现在审美做得不错，Codex 也可以。对于没有受过前端或设计训练的普通人，它一下就能给出这样的审美，

00:38:16・刘果

尤其是……

00:38:17・观众

再加上各种 skill，对我来说真的挺 impressive，很令人印象深刻。

00:38:23・刘果

是，它把基准水线拉高了很多，现在至少对于普通人使用的界面，你很难再接受做得很差的 UI。但倒过来，再复杂的工程，你想让 AI 建立一个大项目、去 harness 它，最后还是要写 Markdown，因为那是很简洁、开放的意图文档。写成 Word 或其他格式，它读起来会麻烦；只写纯文本，不标章节、不标要点，有时连你自己都很难表达清楚。

00:39:03・刘果

绕来绕去，还是回到 Markdown。不过，我们可以反过来让 AI 把 Markdown 编辑器做得更好。谢谢大家的讨论，很多问题可能也是其他人心里在想的。说到 AI，大家最近应该都有很多新体会。过去短短四年，LLM 的变化很有意思，这种张力已经开始出现，而且才刚刚上演。

00:39:36・刘果

如果对应早期计算机的年代，现在其实还很早，还在大型机时代。我再举一个乔布斯演讲时经常用的小例子：内燃机、电机怎样进入工厂。最开始的内燃机很大，一个工厂共用一台，通过轴把各个地方连起来，后来内燃机、电机小型化，各个工位出现小型电机，甚至手钻也用电机，内燃机则进入汽车。这个过程带来一系列生产方式和社会组织的改变。个人计算机经历过，AI 现在也在经历早期阶段。2022 年，ChatGPT 第一次让大家觉得，LLM 好像有点能力、好像能思考了。

00:40:28 · 刘果

它是完全闭源、托管的服务，第一次突破了那个临界点，让人觉得好玩，可以问问看。但真正让它有用的是 coding agent。用过的人知道，过去几年的几个突破口，首先是 harness 的变化：它接入了大量已有的本地命令行工具，用来处理字符串、处理文件。接入之后，它就能快速、持续地处理本地文本，而代码本质上都是文本。内部的循环，也能让它依据你之前的意图和文档，沿着思路继续工作。编程代理很大程度上是把能力放回本地，因为你需要表达的意图本来就在本地，

00:41:28 · 刘果

你的 agency、项目文件都在本地。个人电脑已经有完善的字符串处理工具，本地交互又最快、最及时、最可控。这样，用户就有了能动性，也能自己改造它。我记得 Claude Code 最早的广告语是“the most hackable coding agent”。一旦能改造，程序员就能玩起来。

00:41:55 · 刘果

后来的 skill 文件、MCP，都在这个过程中发展出来。我觉得 MCP 现在甚至快过时了。另一个有意思的例子是 OpenClaw：它进一步把 agency 拉回本地。不只是 harness 在本地，它内部接了 Pi 作为 harness，同时广泛开放接口，把各种本地应用都接进来。循环在本地持续运行，调用远端 LLM，进一步把用户对 LLM 的控制权放到本地。所以［公司简称听辨不清］就很不爽它。它的创新速度比 Anthropic 快很多，



演示页 23 · 00:42:33

00:42:50 · 刘果

你会发现，后来 Claude Code 的不少好设计，最早来自 OpenClaw。这是一段很有趣的历史。所以，计算究竟发生在哪里？目前最前沿、最困难的 LLM 计算仍在云端，但 harness 及其带来的 agency，正一步步推向本地。我喜欢的一个 OpenClaw 例子是：



演示页 24 · 00:43:13

00:43:15 · 刘果

有个人喜欢酿酒，就用 OpenClaw 接入酿酒系统，把酿酒、建网站、卖酒整个过程自动化。这是因为它能把 LLM 接入本地蓝牙，再由本地酿酒软件和酿酒师决定能做什么。这是很具身、很现实的应用。

00:43:42 · 刘果

现在看也许不那么令人耳目一新，但第一次看到时，我觉得很好玩。这种能动性、这种可改造性，只有在广义的本地优先中，控制权放在本地时，才做得出来。所以，现在大家都会面临这个问题：本地代理肯定要用，代理肯定在本地，虽然 Anthropic、OpenAI 都在努力把人推回云端，用它们的 harness，做 remote control 等等，大家也看得出来，因为它们的赚钱方式就是把你拉回云端。但开发者几乎仍会使用本地 harness，因为工程文件、记忆、Markdown 文件都在本地。再进一步，



演示页 25 · 00:44:03



演示页 26 · 00:45:03

00:44:35 · 刘果

会出现一个更尖锐的矛盾：我们把越来越多的思考和工作委托给 AI，它替我们思考、决定，也进一步影响我们。这时隐私越来越重要，因为它能看到所有东西，尤其是医疗、私人信息等敏感内容。即使只是一般地关注个人隐私，暴露的信息也会越来越多。这成为推动 AI、LLM 以某种形式本地化的新方向。这里的本地化可以是私有云、自己管理的 NAS 服务器，或者租用的虚拟机，关键是能够全权控制，不让数据离开自己的机器。

00:45:34 · 刘果

这样才能最大程度保留隐私。现在已经出现混合状态：最前沿、最难的工作，比如最近数学、物理方面的突破，仍由 OpenAI、Anthropic 的闭源前沿模型推动；但开源模型与它们的差距在缩小。



演示页 27 · 00:45:35



演示页 28 · 00:45:47

00:46:02 · 刘果

对于创新产品，比如刚才的酿酒例子，或者嵌入式 AI、嵌入式 LLM，则需要能够下载的开放权重模型，在本地或者自己的私有云里运行。这种张力已经出现，还会继续演变。时间不多了，我后面稍微讲快一点。

新工具如何改变工作



演示页 29 · 00:46:33

00:46:29 · 刘果

看完这些片段，我有一些还不成熟的假设。对本地优先感兴趣的人，一方面想做自己喜欢的软件，另一方面也希望它符合时代潮流、能够活下来。我觉得，大的、困难的技术突破，往往依赖大公司、政府、研究机构高度集中的资源。个人计算机的历史里，就依赖美苏之间的竞争，这种国际关系促成了资源投入。现在 AI 的投入，也可能有很多来自中美竞争，政府和大公司集中资源做重要创新。但要让这些创新进入日常、让普通人用起来，又需要更加开放，让个人拥有更强的改造能力，

00:47:26 · 刘果

让个人和小团队能去 hack、去创新，技术才会成为生活的一部分。电力、内燃机改变工厂和生产方式的过程也类似，有一个技术分散、技术下放的过程。很多人说过，技术成熟之后，就会消失在日常生活中。而当技术融入日常、不再显眼时，它也往往已经成为个人可以使用、修改、改造的东西。对应到软件，常常就是本地优先、去中心化，以及刚才有人提到的 file-based，也就是基于文件。比如 coding agent，



演示页 30 · 00:47:31



演示页 31 · 00:48:03

00:48:18 · 刘果

本质上就是处理文件的软件。Markdown 越来越繁荣，也因为它基于文件，各种编辑器都能读。浏览器本质上也是一个渲染引擎，可以从不同地方、不同来源渲染各种文件。比如我现在讲的幻灯片，就是一个本地 HTML 文件。浏览器已经无处不在，成为互联网最基本的基础设施，任何地方有 WebView 就能渲染。讲完历史，我最后花几分钟介绍现在做的新产品。一方面来自我对历史的兴趣，

moss 与现场演示



演示页 32 · 00:49:13

00:49:11 · 刘果

另一方面也结合自己想做的软件：既想做觉得对的事情，又想让它活下来，这些目标之间也有矛盾。我一直做发布工具，有件事始终耿耿于怀：个人计算机和互联网，本应是人能拥有的最有创造力的创作媒介，最后却变成了消费媒介。比如我们有“浏览器”，却没有“发布器”。这是一个很有意思的不对称：所有人都能浏览，很少人能发布。刚才也看到了，不论中文还是英文，独立网站仍很小众，基本上是极客和技术人员的事。但互联网又是我们能拥有的最有创造力的媒介。

00:50:04 · 刘果

所以我的问题是：如果有一个与互联网浏览器对应的“互联网发布器”，它应该长什么样？它应该支持互联网能支持的各种内容：文字、图片、视频、照片、地图、互动，包括 Jupyter Notebook。它也应该像浏览器一样，支持不同目的地、运行不同脚本。浏览器能读取任何地方，发布器也应该能发布到不同地方。这个雏形其实已经有了。讲内容发布历史时提到的那些模块，技术人员都很熟悉：Markdown、基于 Markdown 的静态网站生成器，



演示页 33 · 00:51:03

00:50:55 · 刘果

以及静态网站托管。本地服务器，每种语言都有好几个；服务器上可以搭 Caddy，GitHub Pages、Cloudflare 也有免费服务，国内也有不少服务商。它们容易迁移、容易发布，都是现成的。工作流也现成：本地有一组文件，打包建构，然后部署托管。还有最后“连接”的部分，也有很多快速演变、尚未整合的模块。比如前几天江江老师讲的 Secure Scuttlebutt，也就是 SSB，以及 Nostr、AT Protocol 等本地优先或去中心化协议。

00:51:50 · 刘果

这些都可以用来发布静态网页、HTML、字符串，只是目前还没有那么多客户端支持。模块已经有了。再比如，发布网站时同时扩散到社交媒体、多种渠道，以及 Newsletter，这些都有成熟做法。独立网站领域有个缩写叫 POSSE，就是先在自己的网站发布，再同步到其他平台。这些都能自动化。对我来说，问题就是怎样把它们放在一起。我可以演示几个例子。



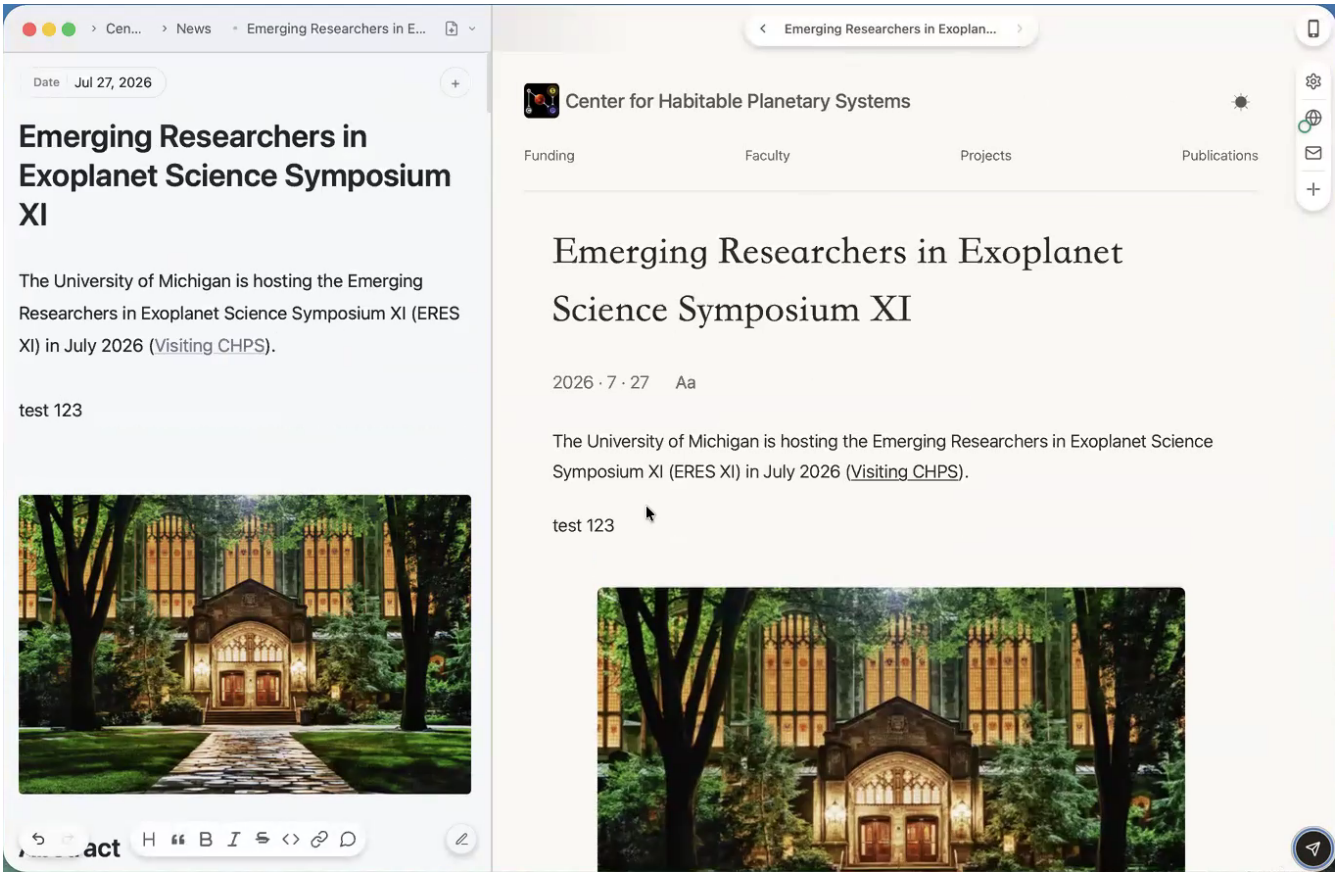
演示页 34 · 00:52:29

00:52:42 · 刘果

在今天分享前，我复刻了几个正在使用青苔的用户的网站。这几位都是我认识的朋友，所以可以给大家展示，也知道原文件在哪里。还有更多使用者，我并不知道他们的原文件在哪里。以后大家可能会慢慢看到更多青苔建构的网站。我选的几个例子比较有意思，可以展示它具体是什么样。

00:53:29 · 刘果

我刚建了一个新的青苔版本，又改了一下，希望不会出错。这是密歇根大学天文系一个叫“宜居行星中心”的网站。它实际上就是一个以网站名字命名的文件夹，内容都是 Markdown 文件。这就是大家熟悉的静态网站建构方式：Markdown、图片、新闻引用等都在里面。青苔做的事情，就是给这个熟悉的工作流加一个 UI，让我爸妈也能用。我真的也在教他们，让他们也能建网站。右键选择“Preview Website”，也就是预览网站，



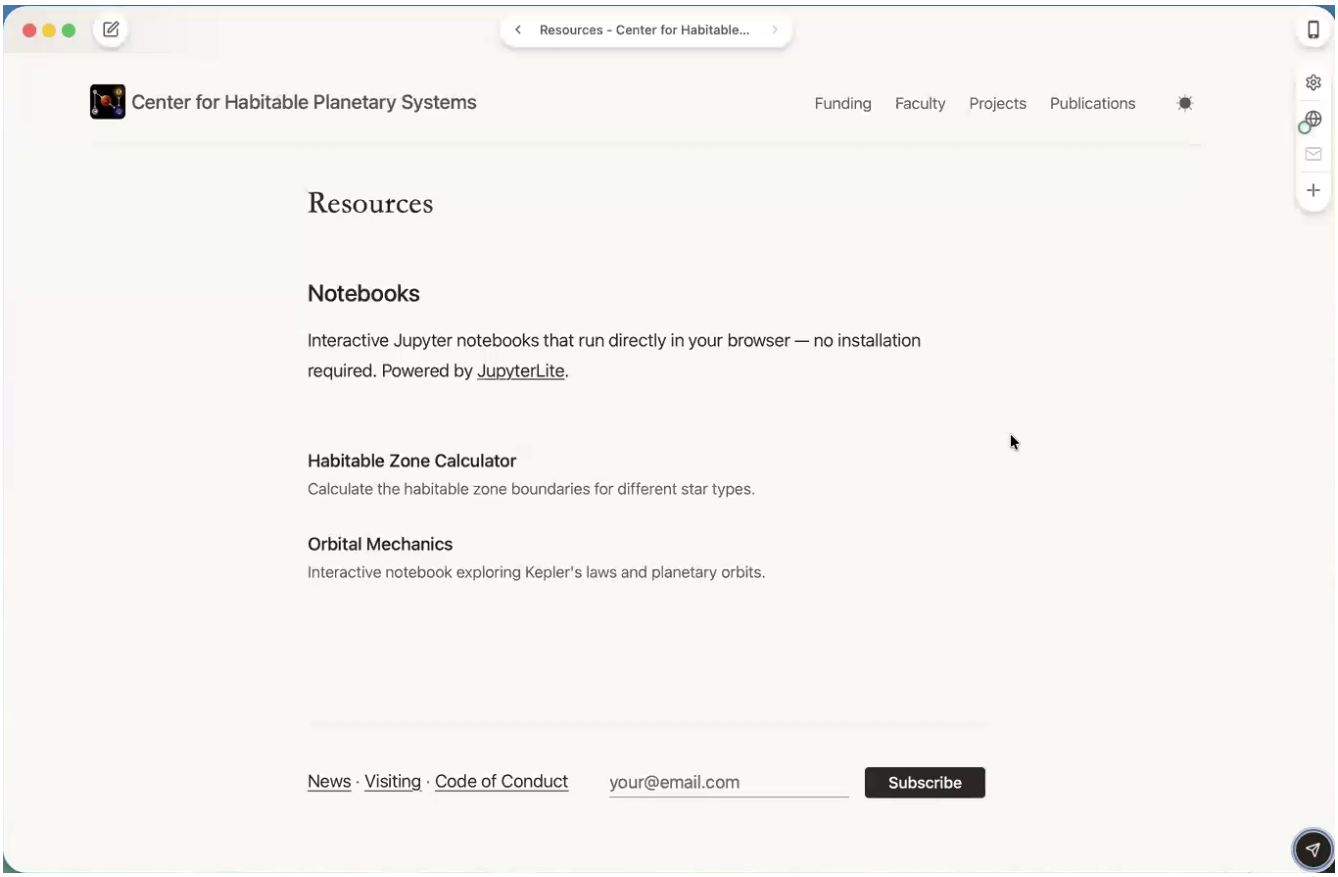
现场演示画面 · 00:54:42

00:54:25 · 刘果

整个网站就建构出来了。排版基于 Markdown 的结构。修改文件后，青苔会监测本地变化，很快更新预览。想集中精力改内容，就在编辑器里写。这也回到刚才的 Markdown 编辑器问题：它允许你不直接使用 Markdown 语法，和很多熟悉的编辑器类似。原文件仍是 Markdown，但编辑器提供其他操作方式。修改之后，右边的预览直接更新。

00:55:17 · 刘果

所以非技术用户也容易使用，所见即所得。想知道手机上是什么样，也可以打开右边的视图，像我们在浏览器里调试一样，查看手机效果。这个网站已经设置好了域名，如果还没有设置，可以在界面里搜索，点击后自动连接。对不懂技术的人来说，在本地买好域名，就能把域名和部署的网站关联起来，会比较方便。



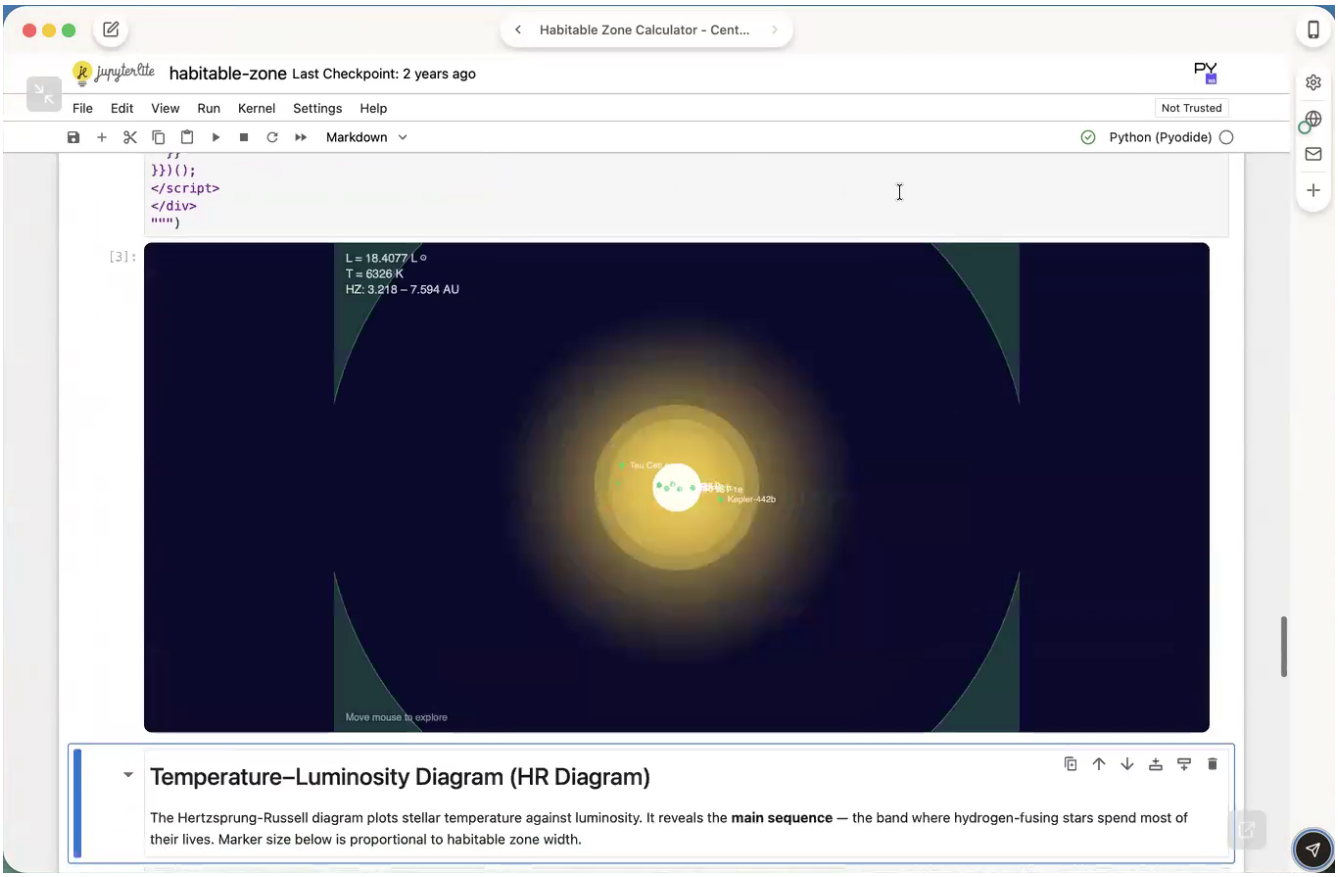
现场演示画面 · 00:56:42

00:56:09 · 刘果

我觉得这样的发布器，除了加图形界面，还应该像浏览器一样强大。浏览器其实很强大，用过 Jupyter Notebook，或者 p5.js 的人会发现，它能把计算和可视化结合起来。发布器是不是也可以做到？比如这个网站发布了一个 Jupyter Notebook，我给大家看看。

00:57:00 · 刘果

它真的就是一个 Notebook 文件。Markdown 里引用这个文件之后，青苔会自动把它打包，在前端渲染出来，然后就可以运行。它直接在前端运行。现在看到的是本地版本，一会儿可以看看他们已经发布到网上的版本。



现场演示画面 · 00:57:37

00:57:51 · 刘果

看在线托管的版本，会发现不需要登录，也不用担心消耗他们的服务器资源，因为 Notebook 里的交互计算全部发生在我的浏览器中，背后是一个 WebAssembly，也就是 Wasm 模块。我可以修改代码，写 Python、JavaScript，都在前端完成计算，后端甚至不需要服务器。这也是静态网站很好用的地方，不必担心计算资源，因为东西都在访问者的浏览器里运行。我想说的是，一个理想的发布器，既然对应浏览器，就应该能发布互联网支持的各种内容。

00:58:42 · 刘果

像 Notebook 这样强大、好用的东西，也应该能发布，更不用说网页本身、视频、图片，都可以打包处理。另一个我觉得有意思的地方，是本地优先这一部分。

00:59:09 · 刘果

为什么说它是 local-first？首先，网站的建构和发布都基于本地。不仅如此，往下面看，我可以打开一个……VS Code，看一下。这个是 alias，可能看不到。我换一个文件夹。这是另一个示例网站，并非真实用户的网站，而是青苔自带的示例。

01:00:06 · 刘果

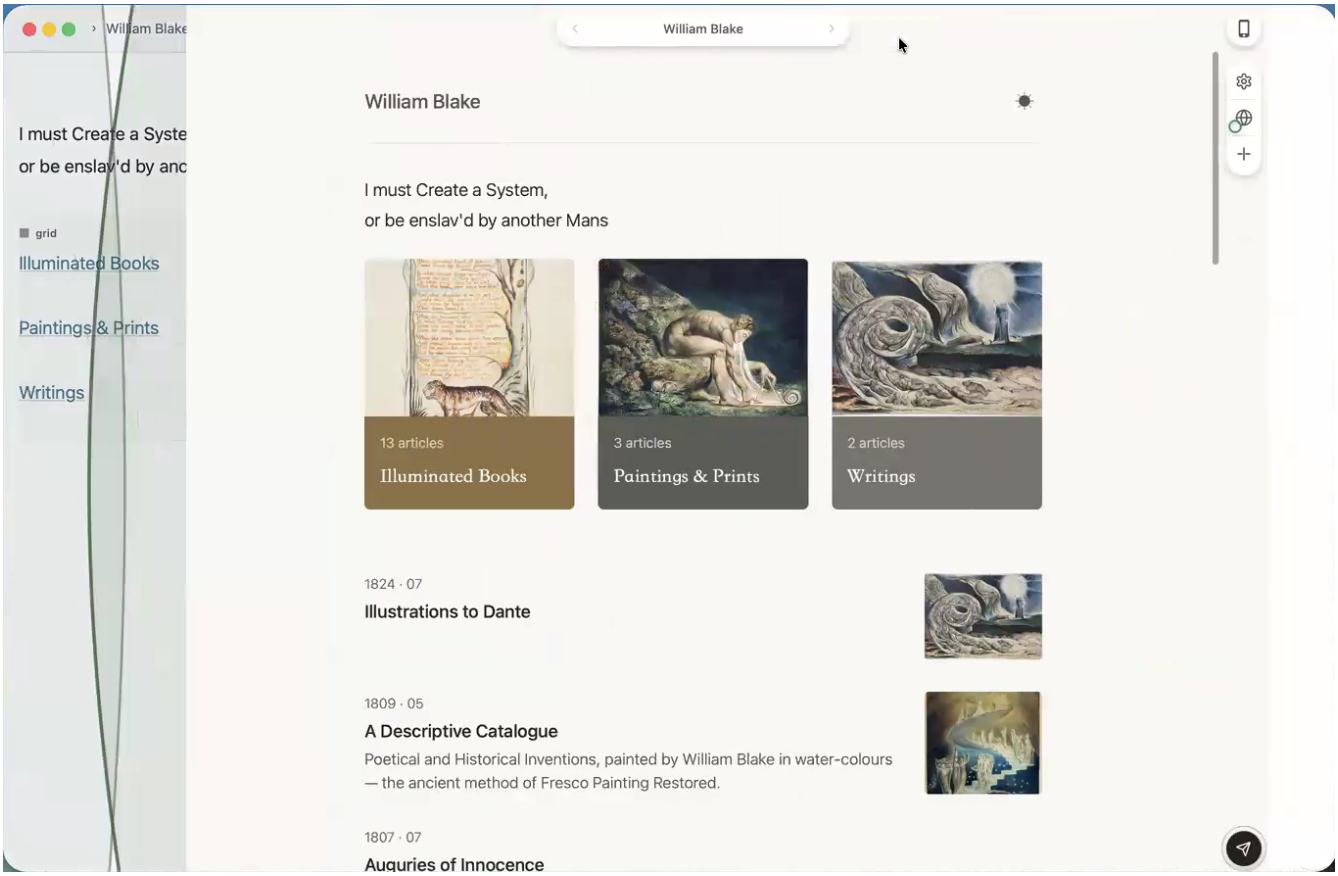
它有大量图片，但同样只是一个简单文件夹，里面放 Markdown、图片等。右键就能预览网站。从生成过程到建构模板、样式、主题，全部都在这个目录下面的隐藏文件夹 `.moss` 里。“全部”意味着什么？连邮件订阅者的列表也在本地；如果加了评论功能，评论也同步到本地；analytics，也就是访问分析数据，同样在本地，更不用说建构出来的网站。

01:01:00・刘果

所以你对它有完全的控制权，想部署到哪里就部署到哪里。设计和修改也在本地。这个例子没有 theme 文件夹，有些网站则有，里面包含全部主题内容。它也有插件，比如这个网站同时发布到 IPFS 和 OnionPress，也就是 Tor 上。插件只是简单的 JavaScript 脚本，青苔运行脚本，脚本甚至可以调用任意二进制程序，把网站发布到任何地方。

01:01:54・刘果

做青苔时，我想保留开发者使用命令行工具时的高天花板，保留所有能自定义、能 hack 的地方，同时让普通人轻松用上。比如添加 front matter，也有直观的界面。把界面设计好之外，更重要的是，我希望它成为一个可以做实验的基础。发布到哪里，本来就应该可以尝试：不只发布到 HTTP 服务器，



现场演示画面・01:02:23

01:02:48・刘果

还可以同步到 Newsletter、洋葱网络，或者 GitHub Pages、IPFS、Scuttlebutt。整个设计就是：所有内容先在本地建构，然后想发哪里就发哪里。这是发布器的关键，也意味着它可以同步到不同社交网络。与此同时，所有信息、内容、建构过程，以及从设计到插件的控制，都在本地，都能自己改。这就是青苔大致的思路。如果大家感兴趣，

交流与问答



演示页 35 · 01:04:33

01:03:40 · 刘果

可以在网站上点击“Request Beta Access”，填写邮箱，免费使用默认托管服务器。也可以直接下载，在本地试用。它现在是开源的，不需要那个服务器也能玩，本地建构后想部署哪里都行；填邮箱则能一键部署到默认服务器。如果你也觉得这些实验有意思，特别是这几天听了 iroh、Scuttlebutt 等不同协议，又对内容感兴趣，自己或身边的人有好的内容想发布到不同地方，欢迎联系我，我们可以一起写插件。

01:04:34 · 刘果

比如怎样发布到 SSB、Nostr、AT Protocol。刚才看到的 IPFS 和 Tor 插件，就是之前的朋友写的，都是很好玩的实验。如果只是想要自己的网站，或者有具体设计，又喜欢青苔的思路，也可以联系。接下来几周，我会开始发布一个 theme library，也就是主题库，让普通人刚上手时，就有现成设计可以选。有想要的设计或想法，也可以告诉我，我们一起做一些有意思的设计，你自己也能用。现在青苔只有 Mac 版本，

01:05:26 · 刘果

接下来两周会发布 Windows 和 Linux 版本。想在这两个系统上试用、帮忙测试的朋友，也可以告诉我。今天的演示就到这里，看看大家有什么讨论或问题，谢谢大家。超了一点时间。

01:06:02 · 主持人

谢谢刘果老师。大家有什么问题，或者想讨论的?

01:06:20 · 观众

刘果老师你好。我是非程序员，处理代码时，习惯先让 agent 看一下，比如青苔这个平台怎样操作，然后把 API key 通过本地、偏加密的方式交给它，让它代我操作。像我这样不太愿意 coding、又想做些东西的人，这可能更方便。所以能不能提供标准化的操作方式，面向 agent，让它来操作这个平台?

01:07:13 · 刘果

是的，我觉得这会越来越常见。青苔设计时也考虑了。打开刚才 William Blake 的例子，`.moss` 下面有一个 agent 文件夹，里面有 skill，整个使用方式都包在 skill 里，引用相关资料。这是一个很有意思的副作用：软件一旦本地优先，再加上 file-based、命令行接口，处理的单元都是本地文件，就天然很适合 AI 使用。再把对应文件和文档包成 skill 交给它，

01:08:04 · 刘果

AI 就能帮你做。青苔设计时已经考虑了这部分，因为这本来就是现在常见的建站方式。用 Hugo、Jekyll 时，你告诉 AI“帮我用 Hugo 或 Jekyll 建网站”，它就会这么做。这是典型应用场景。

01:08:29 · 观众

好，谢谢。

01:08:35 · 刘果

我觉得这个例子很有意思。人们常说 AI 原生、软件要适配 AI，但倒过来，AI 也在不断适配人。市场上能活下来的产品，是让 AI 适配人的。与此同时，AI 住在电脑里，需要从底层长出来，再去迎接人。你会发现，它走向的很多东西，其实都是本地优先领域的最佳实践：harness 基于本地文件，需要 file-based，需要一个……

01:09:09 · 观众

好的。

01:09:10 · 刘果

terminal，也就是命令行接口。有了这些，AI 才能充分发挥作用；倒过来，这也正是程序员的最佳实践，因为 AI 是在人与电脑之间产生的东西。

01:09:24 · 观众

那简直是双向奔赴。

01:09:27 · 刘果

某种程度上，是的。

01:09:31 · 观众（聊天提问）

用远端主要是数据整洁，全部本地如何管理数据？

01:09:31 · 刘果

不客气。可以具体说说哪些数据吗？我能想到需要整理的，比如评论和访问分析数据。流量数据，你肯定想完整保留一份，因为可以分析；评论也想保留，但同时需要操作，比如删除一些评论、进行管理。青苔目前把这两类数据全部同步到本地，所以可以在本地分析、管理。在我的经验里，远端的数据整洁、清理，常常是跨用户的问题，例如跨用户建表，

01:10:24・刘果

给用户分组、推出新功能时，怎样让数据保持干净、整洁、精简。但在本地优先里，每个人的数据都是自己的：属于你的，就全部下载给你；很多数据从一开始就在本地生成。这里更多是 data schema，也就是数据结构的问题，不同版本的数据形状会改变，所以需要 schema migration，也就是数据结构迁移。这更像数据格式随时间演变，而不是数据整洁。属于我的数据都放在本地，之后怎么整理，是我自己的事。个人网站其实没有那么多需要整理的数据。

01:11:17・刘果

对，AI 需要整合计算机的能动性和人的能动性，它其实就是一个本地优先的文件处理器。

01:11:44・刘果

还有其他问题，或者对青苔的建议吗？相信很多人有个人网站，如果有什么目前没实现、想实现的功能，也可以提，想尝试也可以告诉我。聊天里提到“WordPress 同步数据库也差不多”，从最终能拿到所有数据这一点来说，确实差不多。但过程有很多不同。首先，修改 WordPress，现在虽然可以借助 AI，但它的可改造性比较受限：改动需要把插件放进 CMS，使用 CMS 内部的插件接口，而不是以文件或脚本为单位。

01:12:42・刘果

用 WordPress 多年的人会遇到很大的迁移问题，因为东西都在 CMS 里，要逐步迁移，包括所有插件。部署时，也只能部署到 WordPress 实例，不能直接放到任意静态文件托管处。实际一点说，如果要支持一个新文件类型，比如刚才的 Jupyter Notebook，在 WordPress 里做就很困难，或者很绕：怎样把 Notebook 放进 CMS，再提供给前端？如果计算跑在服务器上，访客很快就可能耗光资源。攻击 WordPress 很容易，但攻击静态网站很难：前面用 Cloudflare CDN 挡住，内容都是静态的，没有动态资源。

01:13:36・刘果

所以，即使 WordPress 最终能把数据库同步到本地，也不意味着建构、修改、让本地 AI agent 直接设计场景、写作等过程都在本地。写作仍要进入 CMS。即使最终拿到了本地数据，过程仍然很绕，因为它从一开始就是服务器优先设计。可以绕开这些设计，但绕开的过程有成本和代价。

01:14:17・主持人

我可以问一个问题吗？

01:14:20・刘果

你说。

01:14:22・主持人

如果不止一个用户，一起写作、一起做这个网站，现在有什么方案，或者未来有什么计划？

01:14:38・刘果

这也是本地优先、基于文件的一个好处：已有非常成熟的系统。我觉得 Obsidian 很早就把这一点表现出来了，也给了我很大启发。我们有很多云端共享文件夹，刚才展示的那些，其实基本都在共享。有的在 Dropbox，有的在 Google Drive，有的在 iCloud Drive。这些不是我自己的网站，是我帮别人设计或调整的网站。他们通过各种云盘把文件分享给我。在美国，

01:15:30・刘果

常见的有 Box、Dropbox、iCloud、Google Drive，中国也有对应服务。这时，你接入的是整个基于文件和文件系统的生态，这个本地优先的文件系统生态非常大。命令行选择更多，比如 rsync，文件夹之间来回同步很容易。消费级产品也有很多，比如 Google Drive。中文环境有百度网盘，也有其他网盘，也许有的做得没那么好。但只要能同步这些文件，就可以直接写作，任何拿到文件的人都可以发布网站，所以协作写作会很容易。

01:16:18・主持人

好的，谢谢。

01:16:27・主持人

看其他朋友还有没有问题。如果没有其他问题，就谢谢刘果老师。

01:17:00・刘果

谢谢大家，也谢谢主持人组织。欢迎大家来青苔尝试好玩的插件和新的发布目的地。谢谢大家。

01:17:14・主持人

感谢大家今天参加，谢谢。

01:17:20・刘果

拜拜。

01:17:21・主持人

拜拜。