2022，前端的天🌩要怎么变?

一、🚀 框架：“三足鼎立”到“两大集团对峙”

> 曾经： React 挟天子而令诸侯， Angular 虎踞江东， Vue 夺荆楚而谋天下。

但是，很明显从 2021年 开始，世界的格局发生了变化。

2021前端框架star增量如下：

Angular 在增量上甚至已经跌出了前3。

按“存量”+“增量”的计算方式来看，2022年基本来说 Vue 与 React 两家的霸主地位不会出现太大的偏差了。

再按照当前社区的活跃程度而言， Vue 与 React 的对比，也绝对不能只看其本身的对比。

一个库是否受欢迎，是否有潜力，更得考虑到其社区是否活跃，其生态是否足够丰富。

因此， Vue 生态和 React 生态在2022年，极有可能将会形成“两大集团军”并强的局面。

Svelte 还需要时间发展，而 Angular 确实已显出疲态。

至于 React 和 Vue 到底谁强谁弱，让我们看看**尤雨溪** 本尊怎么说：

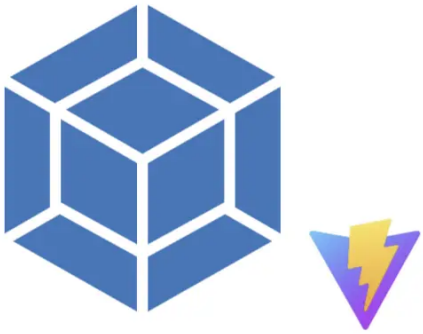
尤雨溪

前端开发等 3 个话题下的优秀答主

整体占有率[9]也没下降，甚至还提升了。npm 的数据跟 React 比维持在 1:4 左右的比例。react 的 npm 数据有相当一部分来自 React Native，纯 web 端的比较最靠谱的数据是看 Chrome 开发者插件的周活，React 在 3~4M 之间（可惜现在非作者看不到插件的具体周活了，只有一个大概范围），而 Vue 插件的周活是 1.92M（stable 1.7M + beta 0.22M），换言之纯 web 端的占有比例在 1:2 ~ 1:1.5 之间。

按**尤雨溪**的说法， `vue` 在国际上的占有率持续提升，但目前仍然距离 `React` 有一定差距。当然，在国内的话这个差距可能会进一步缩小。

二、🚀 打包： `webpack` 称王， `vite` 飞速增长



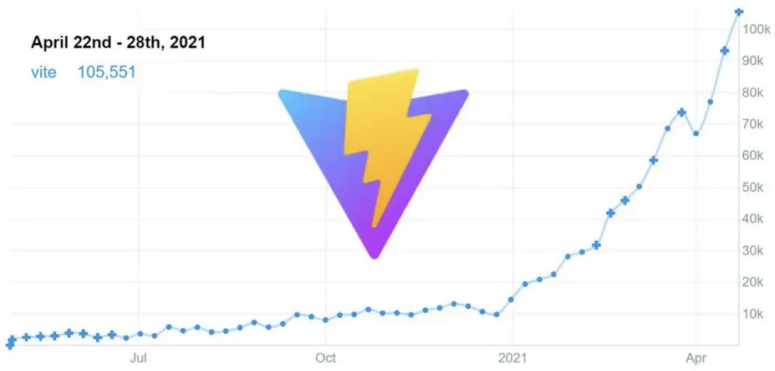
> webpack 开关迎敌，九国之师，逡巡而不敢进。

可以大胆而谨慎地预测，2022年，在 `webpack` 面前，依然没有一个真正能打的对手， `vite` 不行， `snowpack` 也不行，许许多多的构建工具依然没有出现可以真正威胁 `webpack` 历史地位的工具。

如果2022年你的精力只够学习一款前端构建工具，那毫无疑问你仍然应该把注意力放在 `webpack` 上。

活跃的社区、丰富的生态、模块联邦、 `webpack` 依然是绝对的王。

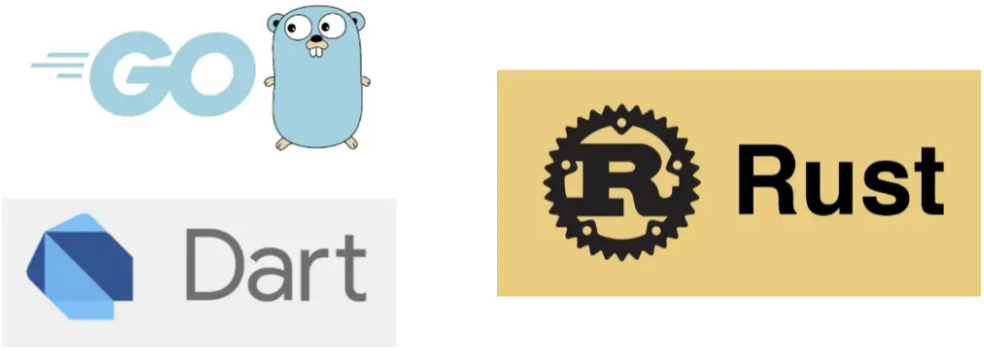
但 `vite` 肉眼可预见地也正在崛起。



只要是使用过 `vite` 的人，一定会惊诧于它的“快”。但截止目前 `2022-01-23` ，它依然只适合在**个人项目** 或 **小型项目** 上进行实践。

当您需要构建企业级应用时，最好的做法依然是“回到webpack身边”，直到下一个“webpack杀手”出现。

三、🚀 语言：更多语言要学， `Rust` 领跑新基建



> 不会 `Dart` ， `Rust` ， `Golang` ，我经常感觉自己不是个好前端。（假的，我已经躺平了）

`Flutter` ， `dart-sass` 说自己属于大前端，你得学 `Dart` 。
`EsBuild` 说自己是前端组件，你得学 `Golang` 。
`swc` 、 `parcel` 、 `Webassembly` ，值得你去学习 `Rust` 的理由似乎更多。

但容我泼一盆冷水，对于80%以上的公司而言，可能并不需要你去学习以上三种语言，来进行所谓的“新基建”。这些可能非但无法帮助你所在的公司解决问题，反而更可能引入新的问题。

至于你说 `Node` 和 `Deno` ？拜托，这二者只是 `Javascript` 的执行环境。根本不算“新语言”。

如果非要在上面三种语言中学一种的话， `Rust` 显然更具优势：

- 性能好。 `Rust` 在运行效率和资源消耗上的优势十分明显，和 `C++` 同一个级别。
- 安全特性。内存安全和保证。
- 跨平台。 `Rust` 拥有优秀的跨平台性，支持交叉编译，一份代码可编译出支持 `windows` 、 `linux` 等多平台的二进制。
- 受欢迎。 `Rust` 连续3年成为 `Stack Overflow` 最受欢迎的语言。
- 前端新基建领军人。 `swc` 、 `parcel` 、 `Webassembly` ， `Rust` 目前在前端新基建的表现确实比其他语言都更加亮眼。

不过如果时间有限，我个人可能会选择做一些更有性价比的事情。

四、🚀 架构：微前端，用还是不用?

“ `微前端` 就是 `iframe` 的升级版。”

初次听到这个论调，我是拒绝的。但当我仔细站在使用方的角度思考这个问题后，我又不得不承认，他说的真的很有道理。

“微应用”要解决的最核心的问题是什么?

- 越来越庞大的“巨石应用”难以维护，开发成本会不断增大。
- 当技术栈升级时，应该如何让存量代码能正确在“新架构下”运行?
- 如何实现“大型企业级”项目的状态共享与业务拆分。

`IFrame` 曾是该问题的一个合理答案，但在 `SPA` 项目横行的当前，`IFrame` 显然要面临更多的问题：

- 内存占用大
- 状态共享艰难
- 资源重复加载
- 「弹个框弹到页面最中间」可能都很难做到

为了在 `SPA` 场景下，实现一种能解决 `IFrame` 困境的方案，于是有了“微前端”。

越来越多 的企业开始尝试使用“微前端”。

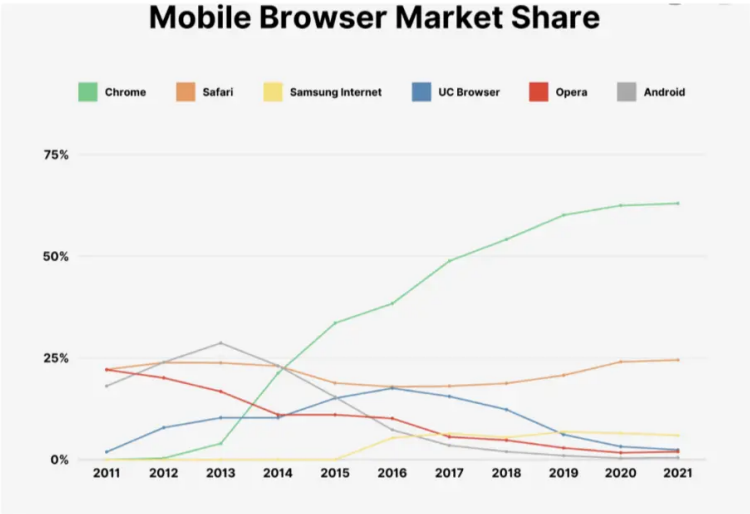
但使用 “微前端” 架构一定要谨慎考虑，它是对业务发展到一定场景后的增强，盲目使用“微前端”，可能并不理智。

五、🚀 浏览器：大人，这是最好的时代

近几年，最让前端感到“振奋”的消息是什么？我心中必然是“Edge切换为Chrome内核”。

微软宣布：“IE 浏览器将于 2022 年 6 月 15 日正式停用，这之后，用户们在使用最新的Windows系统时将无法使用IE浏览器”。

而谷歌浏览器（及其内核）近年来则一路高歌猛进，成为了浏览器市场上绝对的霸主。



垄断行业的 `Chrome` 是否会成为下一个 `IE` ？

这个没有人能给出准确的回答，但迄今为止，我们都能感受到 `Chrome` 前进的方向依然是“标准、安全、高效”。

对于前端而言，老一代浏览器的衰亡无疑是“重大利好”，我们或许无需再为了那些老一代浏览器的存量占有率，而使用 `Babel` 让我们的代码变得冗余难懂。

在此，我不由得想幻想一下“SAFARI”也拥抱 `Chrome 内核` 的那一天。（虽然不大可能）

对于前端而言，最好的时代永远可能正是今天。

结束