

51CTO WOT

World Of Tech 2024

# WOT全球技术 创新大会

智启新纪  
慧创万物



# 使用Astro做独立开发者的探索之路

@i5ting

狼叔

《狼书》作者

全栈技术的实践者



# 目录

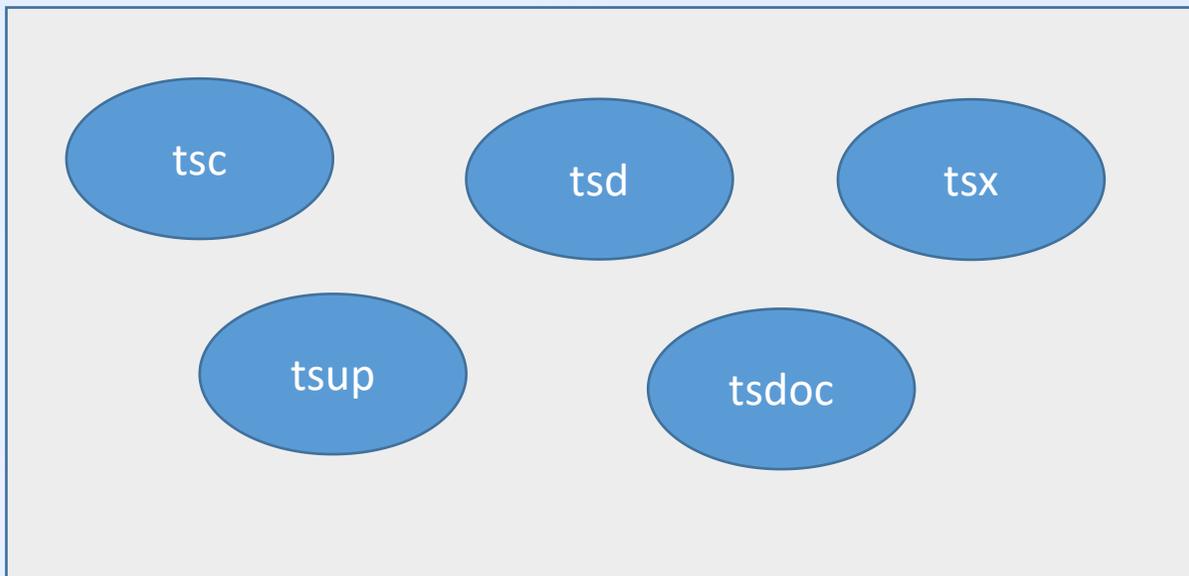
1. 写好一个专业的前端项目有多难？
2. *trpc*和*Astro*实现原理，以及从中获得的启发；
3. 使用*Astro*和AI辅助，做独立开发者的探索之路；
4. 前端谋生的*N*种可能。

01

## 有多难?

写好一个专业的前端项目

# 躲不过去的TS



IoC/装饰器

设计模式

OOP

类型体操

基础，搞定个Equal就很久

# Menoreopo

## What is in this repository? [🔗](#)

- [Tsup](#) as a TypeScript universal package.
- [Tsx](#) as a Node.js enhanced with esbuild to run TypeScript & ESM
- [Tsd](#) as type test runner
- [Tsdoc](#) as document
- [PNPM](#) as workspace manager and package manager.
- [Vitest](#) as a test runner.
- [Size Limit](#) as a size limit plugin.
- [Prettier](#) as a code formatter.
- [ESLint](#) as a code linter.
- [NX](#) as cacheable operations.
- [Changesets](#) as a way to manage changes and releases.
- [c8](#) as coverage
- [supertest](#) as server test
- [cypress](#) as e2e test

<https://github.com/npmstudy/your-node-v20-menoreopo-project>

# 难

- 降本增笑
- 更卷
- 换血

# 还有的玩吗？

- 验收工具

02

## 新技术

TRpc和Astro实现原理，以及从中获得的启发

能使用现代Web规范的地方尽量使用。

- 1、在淘宝做了2年前端智能化，d2c和p2c。
- 2、商用难，营销创意大众化我觉得可以
- 3、我自己用Notion、Github Copilot、Sider
- 4、曾经想让chatgpt帮我写一个rust版本的koa，未果



思维  
体系



学



试

悲观又期待，以前围绕os、以后会不会围绕llm?

# SDK: Python、Java、Node.js



OpenAI

65.4k followers <https://openai.com/> Verified

- **The Continuous Delivery open source SDKs** for [Python](#), [Java](#) and [Node.js](#) allow developers to programmatically interact with Toolchains and Tekton pipelines.
- **The Continuous Delivery Toolchain HTTP APIs** for [Python](#), [Java](#) and [Node.js](#) provide the following functions, based on the user's IAM permissions:
  - Create, read, update and delete toolchains.
  - Read, provision and update tool integrations.
  - Create, read, update and delete toolchain integrations.
- **The Continuous Delivery Tekton Pipeline HTTP APIs** for [Python](#), [Java](#) and [Node.js](#) provide the following functions, based on the user's IAM permissions:
  - Create and delete Tekton pipeline instances.
  - Get and update Tekton pipeline data.

[whisper](#) Public

Robust Speech Recognition via Large-Scale Weak Supervision

Python 47.2k 5.4k

[openai-python](#) Public

The official Python library for the OpenAI API

Python 13.3k 1.9k

[openai-node](#) Public

The official Node.js / Typescript library for the OpenAI API

TypeScript 5.2k 500

<https://www.ibm.com/blog/announcement/new-java-python-and-node-sdks-and-apis-for-ibm-cloud-continuous-delivery/>

## Stack

- Auth: [Clerk](#)
- App logic: [Next.js](#)
- VectorDB: [Pinecone](#) / [Supabase pgvector](#)
- LLM Orchestration: [Langchain.js](#)
- Image Model: [Replicate](#)
- Text Model: [OpenAI](#)
- Text streaming: [ai sdk](#)
- Deployment: [Fly](#)



ai sdk



我是 vercel 的 contributor 了

你做的叫 ai 相关



哇哈哈



<https://github.com/a16z-infra/ai-getting-started>

# Make your doc site intelligent

An open-source tool for embedding AI chat dialog into your doc site to answer user questions based on your content.

[Get Started](#)[View on GitHub](#)

Ask your docs a question...

2. **Easy to Use:** No AI or vector search knowledge is required. You can integrate Documate into your site in minutes by following a few steps.
3. **Fully Controllable:** You own the code and data, and you can choose which pages of your content to index.
4. **Fully Accessible:** Documate is designed for accessibility, supporting full keyboard navigation and compatibility with screen readers.
5. **Fully Customizable:** While providing a default UI tailored for your chosen framework, Documate is also adaptable to meet your specific needs.

To get started with a General Vue Project, you can follow these steps:

1. Initialize Documate in your Vue project.
2. Add the Documate UI component to your project, which renders a button as the start point. 3

Type `⌘` + `/` to open `esc` to close

Powered by  Documate

## 实现

The screenshot shows the GitHub interface for the repository `AirCodeLabs / documate`. The navigation bar includes links for `Code`, `Issues`, `Pull requests`, `Discussions`, `Actions`, `Projects`, `Security`, and `Insights`. The left sidebar displays the file tree, with the `backend` directory expanded to show files like `README.md`, `aircode.config.json`, `ask.js`, `generate.js`, `package.json`, and `upload.js`. The main content area shows the `package.json` file in the `backend` directory, committed by `adcentury`. The code content is as follows:

```
1  {
2    "dependencies": {
3      "@orama/orama": "1.2.2",
4      "ai": "2.2.11",
5      "aircode": "0.6.2",
6      "gpt-3-encoder": "1.1.4",
7      "openai": "4.5.0"
8    }
9  }
```

Files

main

Go to file

- .github
- alternative
- backend
  - README.md
  - aircode.config.json
  - ask.js
  - generate.js
  - package.json
  - upload.js
- cli
- docs
- examples
- ui
  - .editconfig
  - .gitattributes
  - .gitignore
  - .prettierrc
  - CODE\_OF\_CONDUCT.md
  - CONTRIBUTING.md
  - LICENSE
  - README.md
  - SECURITY.md
  - package.json
  - pnpm-lock.yaml

documate / backend / generate.js

Code Blame 48 lines (38 loc) · 1.15 KB

```

4
5   const PagesTable = aircode.db.table('pages');
6
7   async function generateEmbeddings(project) {
8     // Find all the pages without embeddings
9     const pages = await PagesTable
10      .where({ project, embedding: null })
11      .find();
12
13     if (!pages || pages.length === 0) {
14       return { ok: 1 };
15     }
16
17     // Replace newlines with spaces for OpenAI embeddings
18     const input = pages.map(page => page.content.replace(/\n/g, '
19
20     try {
21       const openai = new OpenAI({
22         apiKey: process.env.OPENAI_API_KEY,
23       });
24
25       const { data, usage } = await openai.embeddings.create({
26         model: 'text-embedding-ada-002',
27         input,
28       });
29
30       const updatedPage = pages.map((page, index) => ({
31         _id: page._id,
32         embedding: data[index].embedding,
33       }));
34
35       for (let i = 0; i < updatedPage.length; i += 100) {
36         await PagesTable.save(updatedPage.slice(i, i + 100));
37       }
38
39       return { ok: 1 };
40     } catch (error) {
41       console.error(`Failed to generate embeddings for ${project}`
42         throw error;
43     }
44   }
45
46   module.exports = {
47     generateEmbeddings,
48   }

```

Files

- main
- Go to file
- .github
- alternative
- backend
  - README.md
  - aircode.config.json
  - ask.js
  - generate.js
  - package.json
  - upload.js
- cli
- docs
- examples
- ui
  - .editconfig
  - .gitattributes
  - .gitignore
  - .prettierrc
  - CODE\_OF\_CONDUCT.md
  - CONTRIBUTING.md
  - LICENSE
  - README.md
  - SECURITY.md
  - package.json
  - pnpm-lock.yaml

documate / backend / ask.js

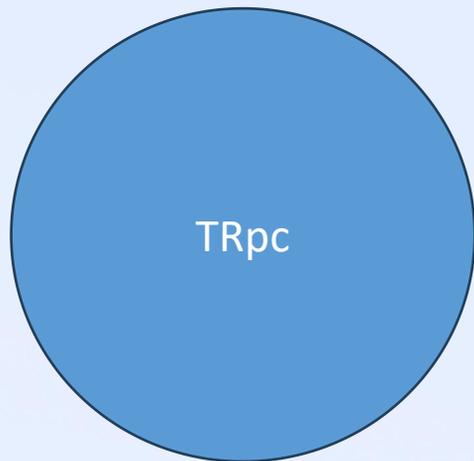
Code Blame 130 lines (110 loc) · 3.8 KB

```

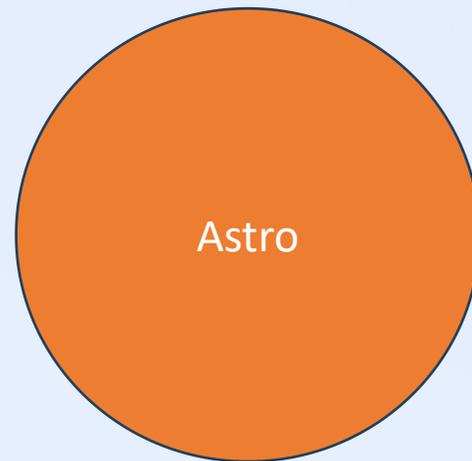
62   // Search vectors to generate context
63   const memDB = await create({
64     schema: {
65       path: 'string',
66       title: 'string',
67       content: 'string',
68       embedding: 'vector[1536]',
69     },
70   });
71   await insertMultiple(memDB, pages);
72
73   const { hits } = await searchVector(memDB, {
74     vector: embedding,
75     property: 'embedding',
76     similarity: 0.8, // Minimum similarity. Defaults to `0.8`
77     limit: 10, // Defaults to `10`
78     offset: 0, // Defaults to `0`
79   });
80
81   let tokenCount = 0;
82   let contextSections = '';
83
84   for (let i = 0; i < hits.length; i += 1) {
85     const { content } = hits[i].document;
86     const encoded = tokenizer.encode(content);
87     tokenCount += encoded.length;
88
89     if (tokenCount >= MAX_CONTEXT_TOKEN && contextSections !== '')
90       break;
91   }
92
93   contextSections += `${content.trim()}\n---\n`;
94 }
95
96 // Ask gpt
97 const prompt = `You are a very kindly assistant who loves to help
98
99 Context sections:
100 ${contextSections}
101
102 Question:
103 ${question}
104
105 Answer as markdown (including related code snippets if available):`
106

```

# 我关注的2个技术 能否低成本引入AI?



协作关系



探索独立开发者

未来

trpc

```
server.ts
...
13 import { initTRPC } from "@trpc/server";
12 import { z } from "zod";
11
10 const t = initTRPC.create();
9
8 export const appRouter = t.router({
7   greeting: t.procedure
6     .input(
5       z.object({
4         name: z.string().optional(),
3       })
2     )
1     .query(({ input }) => {
14       //   ^? (parameter) input: { name?: string |
        undefined; }
1       return {
2         msg: `Hello ${input.name ?? "World"}`,
3       };
4     }
5   });
6
7   export type AppRouter = typeof appRouter;
8
```

```
client.ts 1 x
You, 29 seconds ago | 1 author (You)
12 import { createTRPCProxyClient, httpBatchLink } from "@trpc/
client";
11 import type { AppRouter } from "./server";
10
9 async function main() {
8   const client = createTRPCProxyClient<AppRouter>({
7     links: [
6       httpBatchLink({
5         url: "http://localhost:3000/api/trpc",
4         maxURLLength: 2083,
3       }
2     ],
1   });
13
1   const res = await client.greeting.query({
2     msg: "John",
3   });
4
5   console.log(res.msg);
6   //   ^? const res: { msg: string; }
7 }
8
9 main();
10
```

## Query、Mutation、Subscription

- 它利用了 TypeScript 的强大推断功能来推断出你 API 路由的类型定义。

# 51CTO WOT Aircode

The screenshot displays the Aircode IDE interface. On the left, a 'Files' sidebar shows a search bar and a file named 'hello.js'. Below it, a 'Dependencies' section shows 'aircode' version '0.1.2-2'. The main editor area shows the code for 'hello.js':

```
1 // @see https://docs.aircode.io/guide/functions/  
2 const aircode = require('aircode');  
3  
4 module.exports = async function(params, context) {  
5   console.log('Received params:', params);  
6   return {  
7     message: 'Hi, AirCode.',  
8   };  
9 }  
10
```

Below the code is a 'Console' section with 'Database' and 'Console' tabs. The 'Console' tab shows two log entries:

- 11:02:20.307 Welcome to AirCode. You can now write code and debug online, then click "Deploy" button to ship your function. For more information, check our developer guides at <https://docs.aircode.io>
- 11:02:20.307 Runtime: Node.js v16 (Stable). Function execution timeout: 60 seconds.

On the right side, there is a 'Debug' panel with tabs for 'Debug', 'Environments', 'Deployments', 'Logs', and 'Schedules'. The 'Debug' tab is active, showing a dropdown menu with 'hello.js' and a 'Debug' button. Below this, there are tabs for 'Params' and 'Headers', with a 'Use online requests' link. The 'Params' tab shows a JSON object:

```
1 {  
2   "hi": "aircode"  
3 }
```

At the bottom of the debug panel, there is a 'Response' section with a left-pointing arrow.

Basic

Middlewares

Context

Services

Transaction

Vercel/Next

server.ts

```
function add(x: number, y: number) {
  return x + y;
};

function greet(name: string) {
  return `Hello ${name}`;
};

export default {
  add,
  greet
}
```

client.ts

```
import createClient from "@your-package/api-client";

const client = createClient({
  endpoint: "http://api.domain.com"
});

let result = await client.add(1, 2);
// result: 3

let message = await client.greet("Edith");
// message: "Hello Edith"
```

```
import { createClient } from '@tomrpc/client'  
  
const client = createClient({  
  host: '127.0.0.1',  
  port: 3000,  
  namespace: '',  
});  
const res = await client.a('hello');  
console.dir(res);
```

实际上, 执行

```
http://127.0.0.1:3000/a?${p}=[%22hello%22]
```

对应的 Server 函数

```
rpc.fn('a', function (a: string) {  
  return a;  
});
```

```
rpc.fn('com.yourcompony.a', function (a: string) {  
  if (this.method === 'post'){  
    return return `${this.path} , ${a} `;  
  }  
});
```

# 隐藏http协议

```
JavaScript ▾  
http://127.0.0.1:3000/add?$p=[1,2] GET  
  
JavaScript ▾  
POST /update  
Content-Type: application/json  
[{"name": "new-name"}]
```

# Post

当你在aircode里做POST请求，代码如下

```
module.exports = async function(params, context) {
  return {
    method: context.method,
    paramsFromPost: params
  };
}
```

其实就是Koa的ctx透传了，没有做啥其他方法。

我的server写法，直接用this（此处的this就函数被apply的ctx）。

```
rpc.fn('com.yourcompany.a', function (a: string) {
  if (this.method === 'post'){
    return return `${this.path} , ${a} `;
  }
});
```

```
import { createClient } from '@tomrpc/client';

const client = createClient({
  host: '127.0.0.1',
  port: 3000,
  namespace: 'com.yourcompany',
  methodFilter: function (lastKey: string) {
    if (lastKey === 'a') {
      return 'post';
    } else {
      return 'get';
    }
  },
});

const res = await client.a('hello');
console.dir(res);
```

# 整体设计

## Tomcat

npm v2.0.3 Node.js CI passing

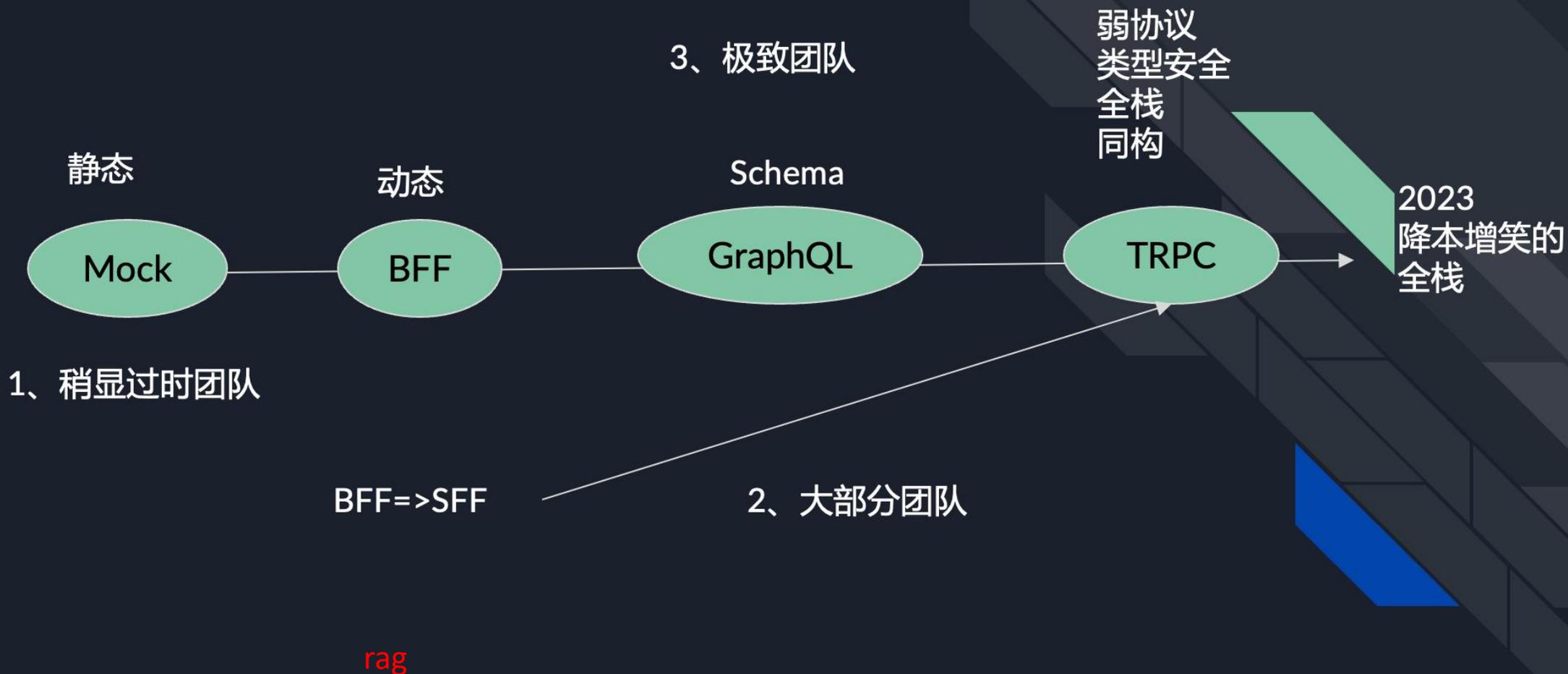
Build function-based API with minimal code

- Basic
  - @tomrpc/core 最核心的函数机制
  - @tomrpc/client 客户端，支持Node.js和浏览器
- Advance
  - @tomrpc/mount 自动加载某目录下面的所有文件，如果e
- Application (Todo)
  - @tomrpc/app 组合core和mount，增加内置中间件、环境
  - 内置cors、serve、view、jwt中间件

```
import { join } from 'desm';  
import { createApp } from '@tomrpc/app';
```

```
const rpc = createApp({  
  name: 'tomapp',  
  base: import.meta.url,  
  port: 3001,  
  debug: false,  
  mount: './f',  
  buildin: {  
    serve: {  
      enable: true, root: join(import.meta.url, '.', 'public'), opts: {}  
    },  
    cors: { enable: true },  
    view: {  
      enable: true,  
      root: join(import.meta.url, '.', 'view'),  
      opts: {  
        map: {  
          html: 'ejs',  
        },  
      },  
    },  
  },  
});  
  
rpc.fn('a', function (a) {  
  return { a: a };  
});  
  
rpc.start();
```

# 总结一下





## 2、野心极大的Astro

# Astro

## What is Astro?

**Astro** is a JavaScript web framework optimized for building fast, content-driven websites.



### Server-First

Astro improves website performance by rendering components on the server, sending lightweight HTML to the browser with zero unnecessary JavaScript overhead.



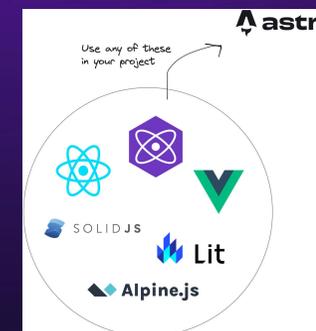
### Content-Driven

Astro was designed to work with your content, no matter where it lives. Load data from your file system, external API, or your favorite CMS.



### Customizable

Extend Astro with your favorite tools. Bring your own JavaScript UI components, CSS libraries, themes, integrations, and more.



# 何为面向内容



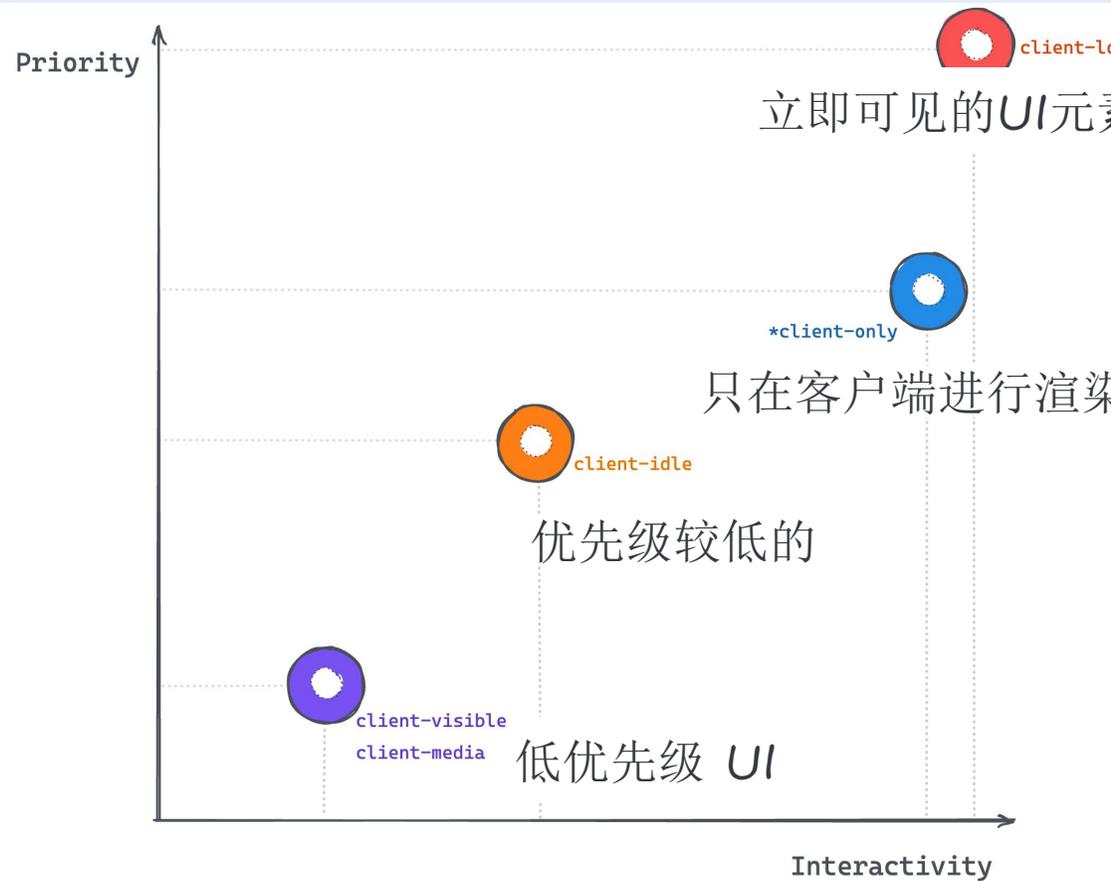
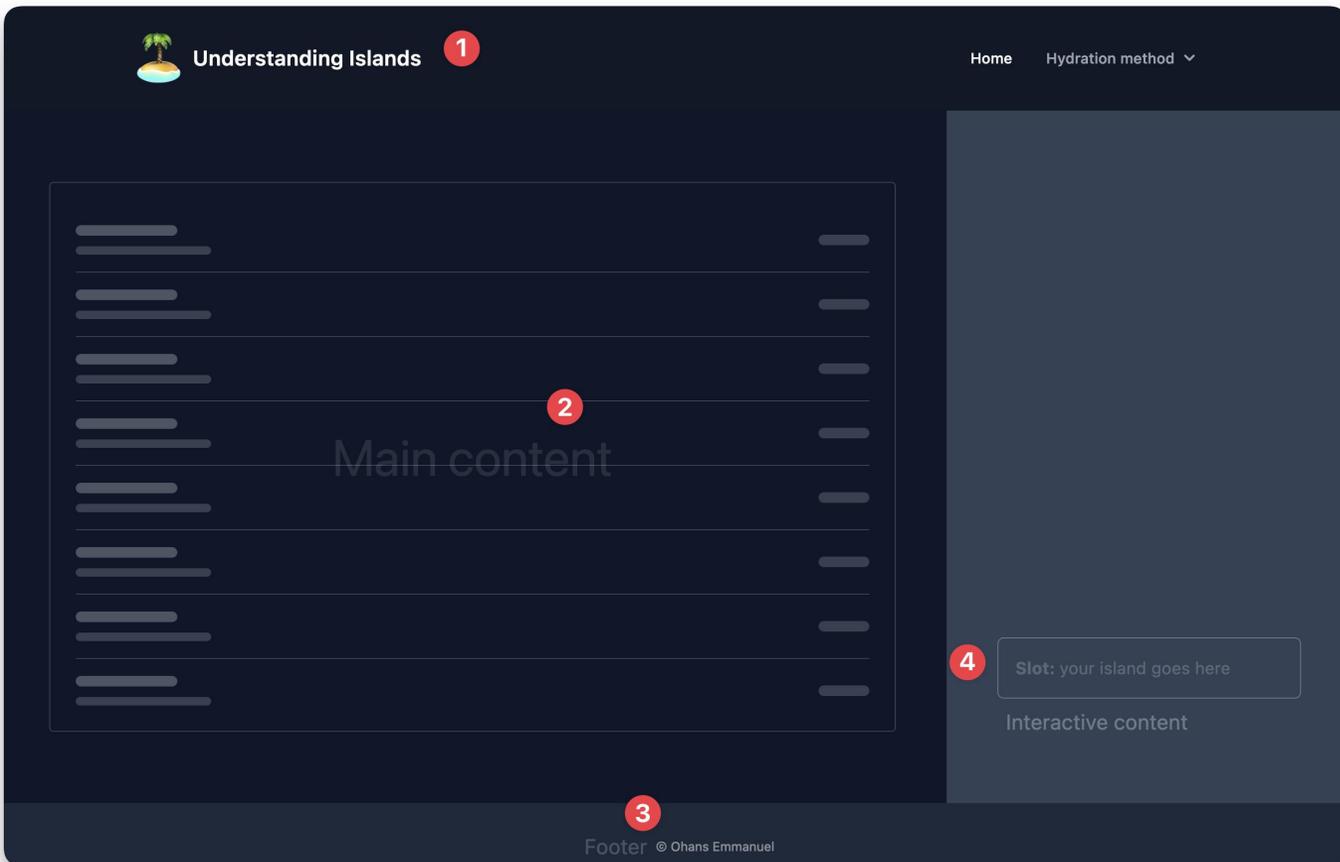
```
// 1. Import utilities from `astro:content`
import { z, defineCollection } from 'astro:content';

// 2. Define a `type` and `schema` for each collection
const blogCollection = defineCollection({
  type: 'content', // v2.5.0 and later
  schema: z.object({
    title: z.string(),
    tags: z.array(z.string()),
    image: z.string().optional(),
  }),
});

// 3. Export a single `collections` object to register your collection(s)
export const collections = {
  'blog': blogCollection,
};
```

通过mdx嵌入各种组件

# 核心：控制权



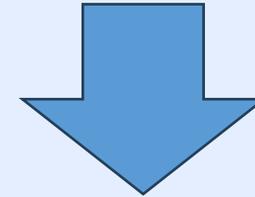
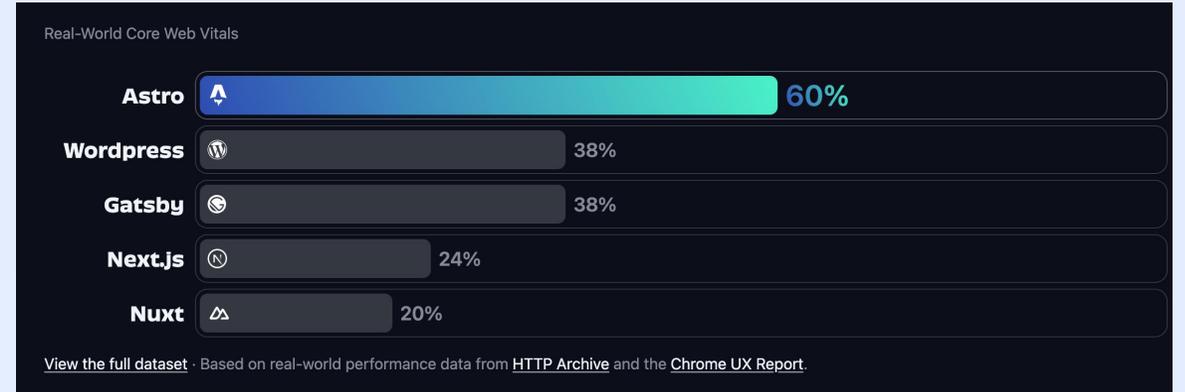
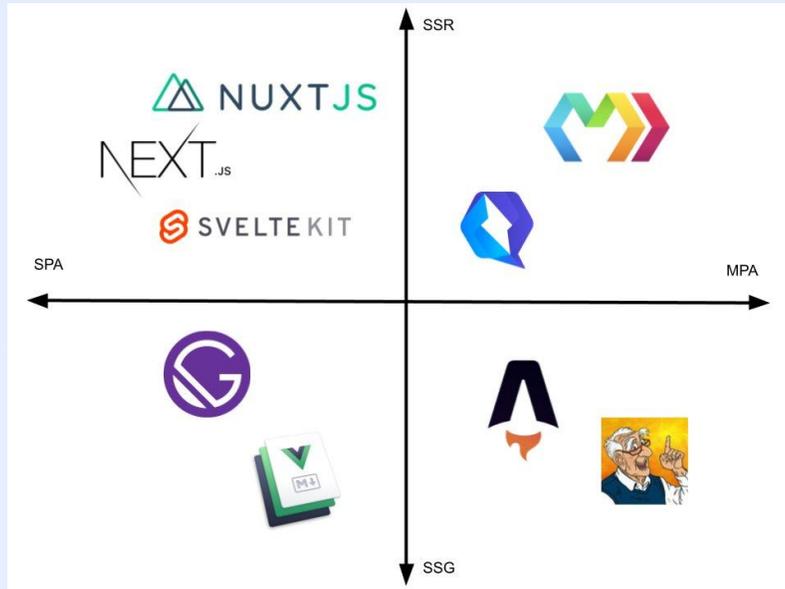
```
<SomeReactComponent client:only="react" />
<SomePreactComponent client:only="preact" />
<SomeSvelteComponent client:only="svelte" />
<SomeVueComponent client:only="vue" />
<SomeSolidComponent client:only="solid-js" />
```

.astro组件和react组件，其实没啥本质区别，唯一需要注意的就是组件加载声明周期控制，即使用client指令。

- `client:load` 它的意思Load and hydrate the component JavaScript immediately on page load.
- `client:only={string}` 回跳过HTML server-rendering，并且只在client上渲染。

使用React组件，这里以评论为例

```
```js
<Comments client:only="react" />
```
```



# 野心

- Content Collections**  
Organize your Markdown and MDX with built-in TypeScript type-safety and frontmatter validation.
- Middleware**  
Wrap incoming requests with custom logic like authentication, logging, or data fetching.
- Deployment Adapters**  
Add an integration to instantly customize your project for Vercel, AWS, or your favorite hosting platform.
- Zero JavaScript, By Default**  
Astro only ships the JavaScript you need and automatically strips away the rest for a faster website.
- Actions**  
Write type-safe backend functions that you can call directly from your frontend JavaScript client code.
- UI Integrations**  
Bring your favorite UI frameworks and component libraries with Astro's flexible island architecture.

## Connect Your Data

- Data Fetching
- Astro DB
- Add Backend Services
- E-commerce
- Authentication
- Environment Variables

src/actions/index.ts

```
import { defineAction, z } from "astro:actions";

export const server = {
  click: defineAction({
    input: z.object({
      name: z.string(),
      message: z.string(),
    }),
    handler: async ({ name }) => {
      console.log(`Received name: ${name}`);
      // Do something such as querying a database
      return { success: true, data: `Hello ${name}` };
    },
  }),
};
```

src/components/ActionButton.tsx

```
import { actions } from "astro:actions";

export function ActionButton() {
  return (
    <button onClick={async (e) => {
      e.preventDefault();
      const result = await actions.click({
        name: 'liruifengv',
        message: 'hello',
      });
      console.log(result);
      if (result.success) {
        alert(result.data);
      }
    }}>
      click me
    </button>
  );
}
```

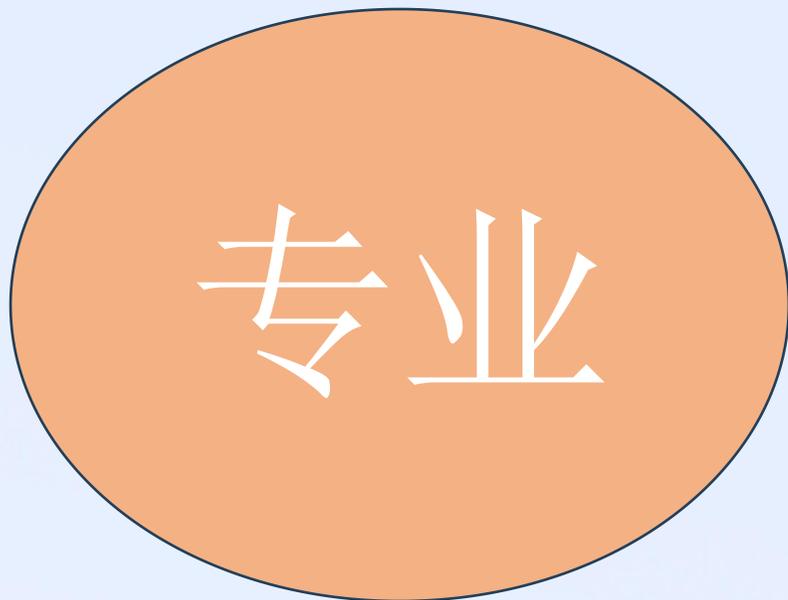
# Trpc和Astro的2个问号

为啥要简单?

为啥要内容优先?

时代变了

# 你站哪边？



TS/Nest + Next.js + TRPC



JS/Astro + 啥快用啥

03

## 独立开发者

使用Astro和AI辅助，做独立开发者的探索之路；

替代70%？

# 独立开发者 出海 看上去很美

← 主题帖

 **npmstudy**  
@i5ting

2gua提问：独立开发者的最佳技术栈是什么？

-----

基础4大件：node+astro+react+tailwind

后端node里已经有了。koa\express就够了，vercel上发布。  
当然，vercel funcion、next也行  
如果在补一个，shadcn/ui可以算一个。

至此。6大技术栈

- 1、node
- 2、astro
- 3、react
- 4、tailwind
- 5、next
- 6、shadcn/ui

使用场景

- 门户、博客、文档，astro无敌，性能好，开发简单，如果想ssr，也可以有很多node adapter
- 简单api，express、koa、vercel funcion、next都可以
- astro可以quick，也可以dirty。我个人比较习惯react，组件生态足够。实在需要，加个shadcn/ui
- 我不喜欢next，但next有一些生态是不错的，拿过来改改用，还是很爽的
- 关于css，tailwind写响应式，真是太爽了，不是那么关注可读性，那真是太快了
- 以上技术和各种现有服务都非常容易集成，比如clark、paddle等

独立开发者要求快，能复用就复用，将精力放到产品设计上才是正经事。

以上。欢迎讨论

下午12:00 · 2024年3月15日 · 19.3万 查看

Q 搜索

### 相关用户

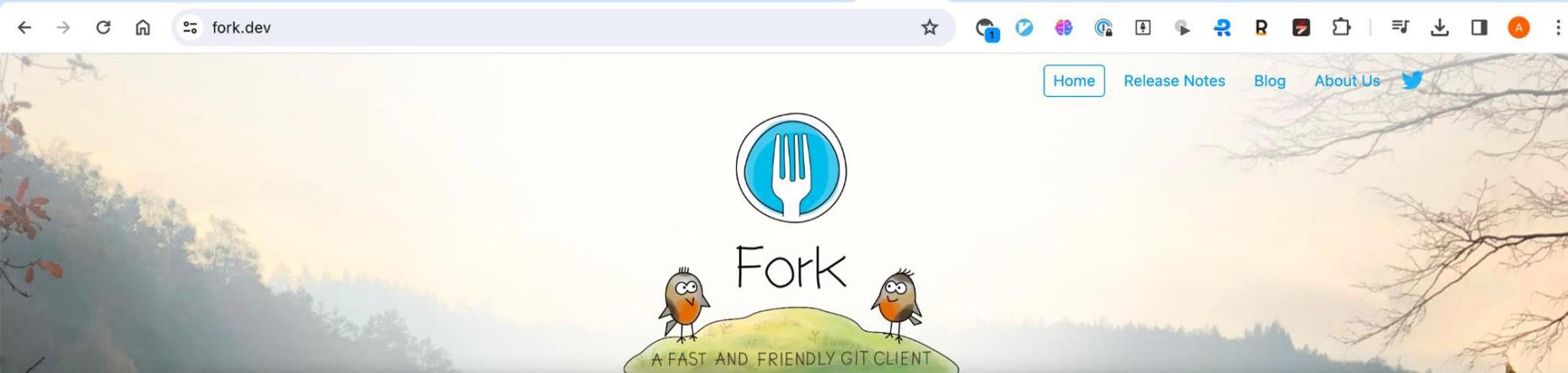
 **npmstudy**  
@i5ting  
关注microsaas，只聊技术和生活，仅代表个人观点！

### 你的趋势

- 台湾 的趋势 **TikTok** 109万 posts
- 台湾 的趋势 **#lookkaewkamollak** 6.25万 posts
- 商业 · 金融 趋势 **Solana** 44.9万 posts
- 台湾 的趋势 **#AIArtwork** 1.69万 posts
- 时尚 · 美容 趋势 **#GIRLFREEN** 1.87万 posts
- 商业 · 金融 趋势 **#blockchain** 4.21万 posts
- 台湾 的趋势 **#Alphoto** 7,047 posts
- 台湾 的趋势 **Jesus** 26.5万 posts
- 台湾 的趋势 **#hololivefesEXPO24**



Quick and dirty?



| Commit   | Author             | SHA            | Time                         |
|--|--------------------|----------------|------------------------------|
| AccessSummaryAnalysis: relax the verification of expected no-escape partial_a...   | Erik Eckstein      | 479517f        | Mar 29, 2021 at 13:21        |
| Merge pull request #36624 from rjmccall/executor-serialization                     | John McCall        | 63d5cf1        | Mar 29, 2021 at 09:05        |
| <b>Merge pull request #36620 from DougGregor/limit-...</b>                         | <b>Doug Gregor</b> | <b>d0551d9</b> | <b>Mar 29, 2021 at 07:50</b> |
| Merge pull request #36622 from gottesmm/pr-f547a5b1d52f9b19efd84b124f08...         | Michael Gottesman  | 360e32b        | Mar 29, 2021 at 03:20        |
| Merge pull request #36621 from rjmccall/continuation-abi                           | John McCall        | 30b4769        | Mar 28, 2021 at 23:46        |
| Merge pull request #36619 from slavapestov/fix-signature-rebuild-with-layout-re... | Slava Pestov       | 22506f9        | Mar 28, 2021 at 15:03        |
| Merge pull request #36615 from slavapestov/redundant-layout-constraints            | Slava Pestov       | 986d033        | Mar 27, 2021 at 23:40        |
| Merge pull request #36605 from apple/QuietMisdreavus/incremental-symbol-...        | QuietMisdreavus    | 98323f1        | Mar 27, 2021 at 14:23        |
| Merge pull request #36618 from davezarzycki/pr36618                                | swift-ci           | f...           |                              |

**Commit** Changes File Tree

**AUTHOR**  
 **Doug Gregor** dgregor@apple.com  
March 29, 2021 at 07:50:29 GMT+2

**COMMITTER**  
 **GitHub** noreply@github.com  
March 29, 2021 at 07:50:29

REFS: **main** origin/main  
SHA: d0551d93f5a7c529704a719a56254fff7e61105b  
PARENTS: [360e32b](#) [61b2218](#)

Merge pull request #36620 from DougGregor/limit-global-actor-propagation  
[Actor isolation] Limit propagation of unsafe global actors to instance members

Fork

a fast and friendly git client for Mac and Windows

License

**\$59<sup>99</sup>**

- Personal and commercial use
- One-time purchase
- 1 user, up to 3 machines at a time

Buy Fork

support@fork.dev

Home Release Notes Blog About Us Twitter License

Copyright © 2023 Danil Pristupov

# 8大技术栈

Node.js

Astro

MDX

React/Vue

Tailwind

Next.js

Remix

NextUI

# 场景

建站

Astro

文档

Starlight

博客

Astro | Next

集成服务

Supabase

加订单、  
支付

Paddle

开发完整  
saas

Unkey

海外支付

Stripe

150+  
240

Pricing

## Get unlimited access.

Discover the ideal plan, beginning at under \$2 per week.

Pay Yearly

Save 25%

Pay Quarterly

### Free

For starters and hobbyists that want to try out.

## Free

- ✓ 10 users included
- ✓ 2 GB of storage
- ✓ Help center access
- ✓ Email support

Continue with Free

### Pro

Most Popular

For small teams that have less than 10 members.

## \$72

/per year

- ✓ 20 users included
- ✓ 10 GB of storage
- ✓ Help center access
- ✓ Priority email support

Get started

### Team

For large teams that have more than 10 members.

## \$90

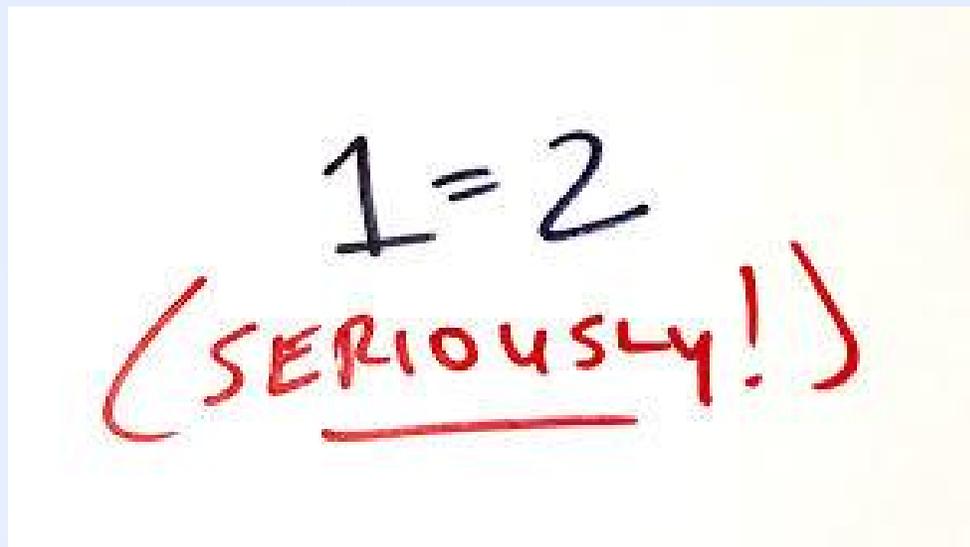
/per user/per year

- ✓ 50 users included
- ✓ 30 GB of storage
- ✓ Help center access
- ✓ Phone & email support

Contact us

## 举例：如何结合趋势去卖软件

- 如上所讲技术
- 到低码搞一个工具,
- 做landing page
- 如何运营销售



04

未来

前端谋生的N种可能

苟住或者下一个卷地，其他

# 去处

Web3  
或其他

Remote

NextUI  
Affine  
大圣

开源

形形色色  
短剧  
集卡

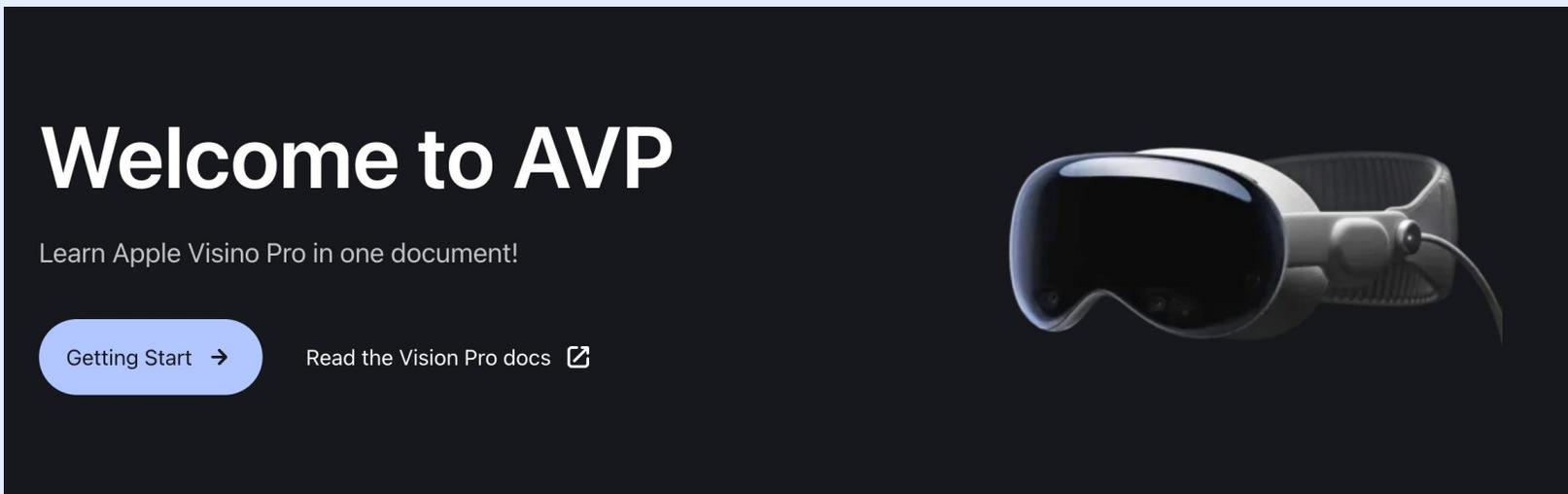
外包

做自己的产品  
探索新方向

Indie hacker

AI都在影响  
早做准备

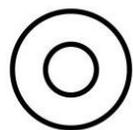
# AVP



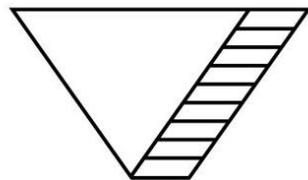
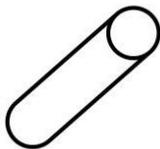
M2  
16G

3500刀

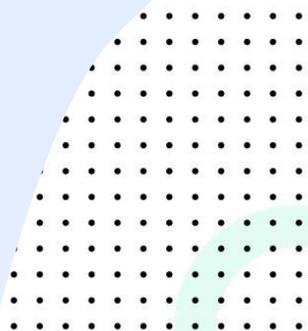
600克



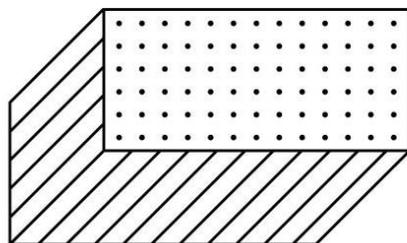
提效



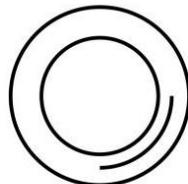
langhain



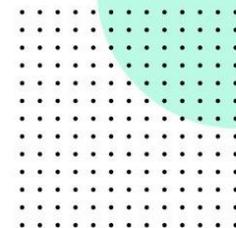
sdk



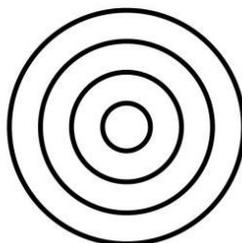
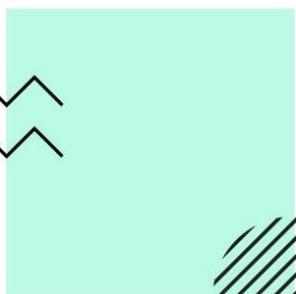
Llm



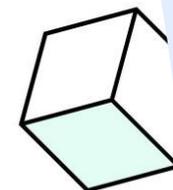
新的协作方式



Vr/ar



组装



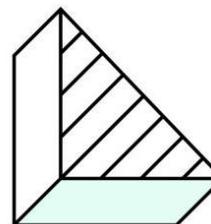
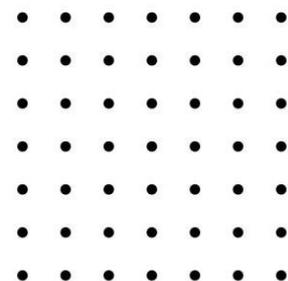
降本



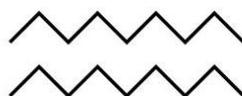
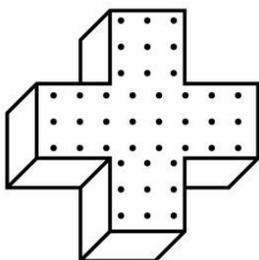
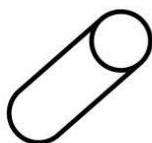
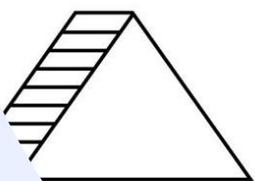
全栈



组件



开源



期待下一个十年

谢谢观看

THANKS