nginx -s stop

2021 年 3 月 10 日

1 引言

就在三天之前，我为这个博客创建了一个 Next.JS 项目，并且由此启动了这个博客基础架构的现代化转型进程. 事实上，这部分有些出乎我的意料，我完全没想到会这样. 在此之前，博客是基于 Gatsby.JS 的，如同我在一篇关于本站的说明中提到的那样，并且在当时看起似乎是：Next.JS 虽然看起来比 Gatsby.JS 好一点，但是也不算绝对的优势，而且我选择使用 Next.JS 其实是出于一种尝试将博客的数据与样式分离的想法，将博客的数据与样式分离实际上也是我这几天一直在做的事.

下面我将开始逐一介绍发生的改变：

2 样式的更新

在创建 Next.JS 项目的过程中，我其实是顺带地就重新又写了一遍原有的主题（如果可以称之为主题的话），效果就是：1）主题颜色从 solarized light 阳光色换成了纯白色，2）并且主页的文章入口和关于页面的文章入口的框框去掉了，仅在友链页面的入口保留了框框，3）与此同时仍然保留了友链页面的随机排序，但是不再是在运行时进行排序了，而是在每一次页面加载的那一瞬间进行排序.

图 2-1: 新

图 2-2: 旧

两种样式都差不多. 放在一起对比来看似乎还是旧的更加简洁，事实上也是旧的比较耐看.

3 引入 TypeScript

在 Next.JS 项目中引入 TypeScript 非常简单，在项目根目录 touch 一个 tsconfig.json 文件，然后 Ctrl+C 关掉当前运行着 Next.js 的 Node 程序，然后再启动 Next.JS，则系统会自动为你写好 tsconfig.json 文件，以及会创建一个用于声明类型和接口的 next-env.d.ts 文件，再对自己想要进行 TypeScript 化的文

件重命名就行，改成 TypeScript 后，具体的项目启动代码 (starter) 可以参看 Vercel 在 GitHub 上面的 next-learn-starter 项目．引入 TypeScript 的第一个好处就是配合 VS Code 编辑器，语法自动提示变得更加聪明、更加智能而且更加周全了，如果有什么错误基本上甚至都不需到编译期，在编辑期就能被发现，可以说是很可靠．概括地来说就是语法提示变得智能了，项目的开发也变得规范了．

4 数据与样式分离

当我还在用 Hugo 写博客的时候，Markdown 文本文件以及 Markdown 里面要引用到的图片都是放在同一个 Hugo 仓库里，这样子虽然 Self-contai，但是久而久之，仓库会变得非常庞大和臃肿，而且图片以及其他二进制文件一般来说都是不可 Git 的，况且图片容易过大，图片大了 CDN 就不愿意对它进行缓存．数据当然也包括文章，所以文章我也移出了这个博客的仓库，并且单独见了一个仓库来存放它们．这样做的好处有什么呢?

一是支持热更新 (hot-update)，我可以在 React 的 Component 里写下能够在客户端浏览器里执行的 JavaScript 代码，然后这段代码会轮询数据节点服务器，一旦发现有更新，就调用 setState 函数将新的文章添加到列表，也就是说，正在浏览者博客的访客也能无需刷新缓存就能及时看到最新内容．

二是支持页面的请求时构建，也就是说，当一个 HTTP 请求发生时，Next.JS 才去访问我的数据节点服务器，拉取数据来构建页面，这样一来一定可以保证每时每刻用户看到的页面都是最新的，我就无需再在博客的每一次更新时去做 Git 提交并且重新部署，从而也就实现了更加高效便捷且有效的最新内容分发效果，加快实现了博客的数字化转型，又好又快地将博客的发展堆上一个新台阶．

三是更好地支持了内容与样式分离的总体设计方针，博客的 Next.JS 仓库就专注于样式的设计，也就是展示效果的设计，而对样式的修改不会影响到数据，与此同时，对于同样的数据，也可以轻松应用不同的渲染方案，以此实现更加灵活便捷的博客开发的工作流程．

四是为数据安全提供了更加强力的保障，对于数据，我们有专门的 Git 仓库来追踪其变动，并且安全的 Host 在专业服务中，所以是安全无虞的．并且，在博客网站重新设计时，甚至如果将来从 Next.JS 迁移到另一个框架，也没有数据迁移的难题出现．

5 Pretty URL 功能

我们通过 Next.JS 的 URL 重写 (Rewrite) 功能实现了 Pretty URL，具体来说，一个具有语义 URL 且简单易懂，并且包含文章标题的 URL 可以叫做 Pretty URL，例如：

```
https://exploro.one/posts/the-title-of-an-article
https://your.domain.com/articles/how-i-improve-my-programming-skills
```

都可以称为 Pretty URL, 因为它们看起来很友好，而：

```
https://some.domian.com/articles/1.html
https://some.domain.com/3d5ea905-14c0-4bc7-8764-60dd29d7a87e
https://example.com/index.php/34
```

则不能称为 Pretty URL, 因为在它们之中看不到标题，也看不到语义．类似地说，Pretty URL 有点接近 RESTFul 风格，有兴趣地读者不妨自行查阅资料多多了解一下．

在 Next.JS 构建静态页面的步骤，我们加入了实现 Pretty URL 的一部分代码，假如说一篇文章的 URL 原本是

```
https://resource.domain.com/pdf/title-of-article.pdf
```

我们则对它进行字符串替换，得到

```
/posts/title-of-article
```

然后我们在一个规则列表中添加一条规则：

```
rules.push({
    source: "/posts/title-of-article",
    destination: "https://resource.domain.com/pdf/title-of-article.pdf"
})
```

最后将这个规则列表 rules 放入 Next.JS 的项目根目录配置文件：next.config.js 中使得 URL 重写功能生效．从此以后，博客文章的地址将会是：

```
https://domain.com/posts/title-of-article
```

这指向 Next.JS 的服务器，而服务器会根据重写规则将它重写为

```
https://resource.domain.com/pdf/title-of-article.pdf
```

并向重写后的地址（也就是原本的地址）发送请求，再将得到的内容以

```
https://domain.com/posts/title-of-article
```

的名义发回客户端，在客户端看来，就好像始终是在访问

```
https://domain.com/posts/title-of-article
```

一样，并没有意识到这底下 (underneath) 的这一系列 URL 重写操作的发生 (图 5-1)．

顺便说，利用这个重写功能，旧的站点 beyondstars.xyz 亦已无缝地并入新博客的大家庭，它的文章现在已经与新博客 exploro.one 的文章并列 (图 5-2)．

图 5-1: 浏览器地址栏的 Pretty URL 现已启用

图 5-2: 旧的站点已经并入新站

6 新增了若干个友链

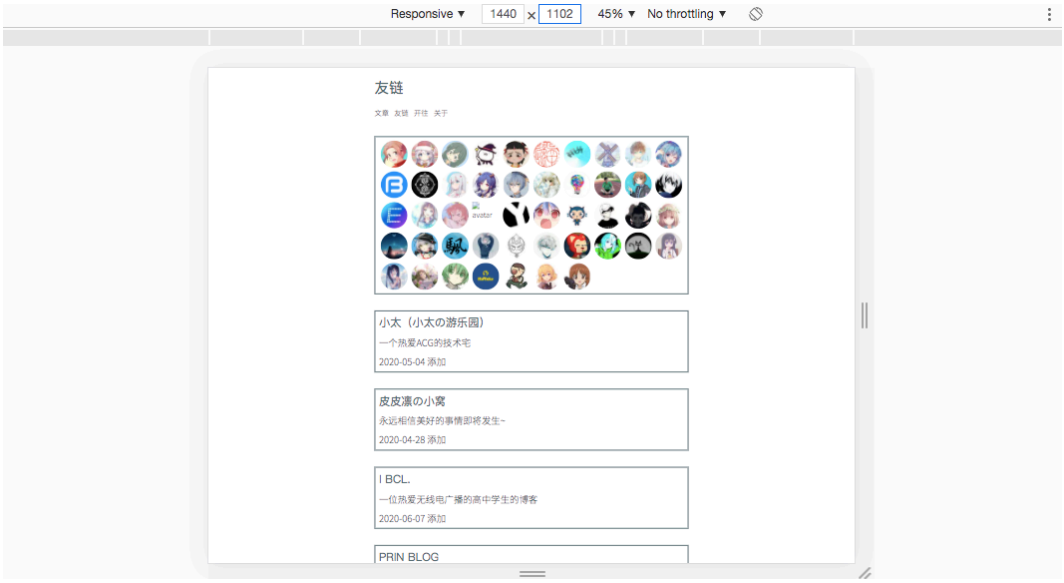

图 6-1: 友链页面预览

从样式上看：头像从方框改成了圆框，并且纯白色的背景、圆框头像和灰绿色的方框组合起来看起来更加干净 (clean).

值得注意的是，本站成立于 2020 年的 3 月初，至今也已刚好满一年，所以我就申请了让本站加入十年之约并且也获得了准许.

更多的友链意味着更好的连接性，每一个个人博客都是浩瀚信息海洋上的一座孤岛，连接起来，我们可以变得更强.

7 引入了 umami 分析工具

由于博客是部署到了 Vercel，所以相比以前的 Cloudflare + 个人 VPS 的 Nginx 方案，所以查看日志的方式也有所不同. 那么为了能够方便的查看网站的流量情况，我为本站点引入了 umami—一款开源的基于 Next.JS 的分析工具，它主要功能是记录访客人数，并且界面非常简洁 (图 7-1):

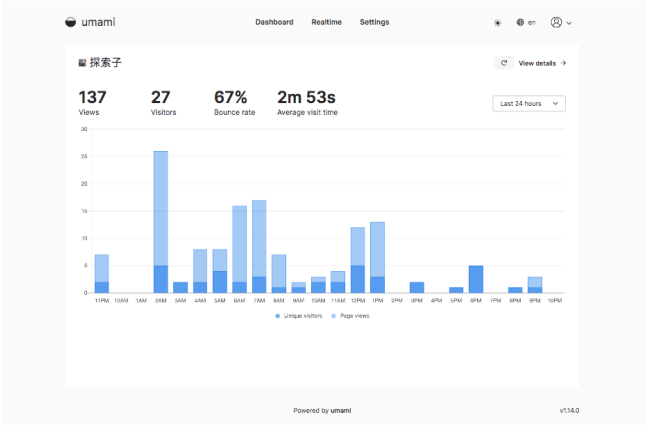

图 7-1: umami 的后台界面

8 彻底从 VPS 迁移到云平台

随着 Jamstack 架构的兴起和越来越多的部署平台变得流行，如 Netlify, Vercel, Cloudflare Worker，Heroku, FireBase, KubeSail 等，个人博客的搭建方案也有了越来越多的选择，并且搭建个人博客的门槛和难度也越来越低．就以本站为例，它现在已经是完全部署在 Vercel 上的．Vercel 的优势有许多：它提供 CD 支持，可以非常容易地与 GitHub 集成，当你在本地用 git 工具将更新提交至 GitHub 时，与 GitHub 集成的 Vercel 就会自动拉取你的仓库并且开始构建 (build)，这实际上也是现如今大多数静态网站的部署方案和工作流程．

包括旧站 beyondstars.xyz 和 Gatsby.JS 版本的本站 都已经悉数依靠 Vercel 的强力上传到云端，博客的数据和元数据也都是部署到了 Vercel，几乎一切可以上 Vercel 的项目都上了 Vercel，所以，随着一条命令：

```
nginx -s stop
```

的执行，宣告着我的博客以及围绕着它的所有服务都已完全上云．如有必要，我可以立即关闭我所有在租的 VPS 机器，而不会影响网站丝毫．