一次 serverless + puppeteer 的实践与填坑

上班多喝热水 2023-08-03 1,109 阅读4分钟 关注

从生成PDF需求引起的问题

因为项目中涉及到了 `PDF` 文件的生成，目前生成 `PDF` 使用的纯前端 `html2canvas + jspdf` 的方案，这种方案虽然不需要服务端的支持，但用户体验也大打折扣，最终权衡以下弊端舍弃了该方案，转而拥抱 `puppeteer`.

弊端：

1. 内容中文字**不能选中**，链接也**不能点击**，本质上就是先绘制图片，再将图片插入到 `PDF` 中
2. 图像本身的属性比如 `obejct-fit` 设置也不起效果，导致导出后的图像被**拉伸**，
3. 伪元素的效果设置也无效，且导出的 `PDF` 内容中伪元素大小显示不一致，比如 `li:marker`
4. 生成的**文件过大**，因为是图片，生成的内容如果不通过放大增加分辨率，最终生成的内容将会非常模糊，但是放大之后文件就变得非常大，同样的内容使用服务端导出大小在 `300KB` 左右，前端导出在 `10M` 左右，这是一个**非常不能接受**的缺点

为什么选择 puppeteer

虽说前端还有 `print` 方法可以生成质量不错的 `PDF`，但是在用户体验上来说还是差了些，需要用户多次点击按钮操作，弹出的一些选项需要用户自己进行勾选，所以综合考虑还是使用无头浏览器来将这一系列操作都自动化！

解决服务器问题

要在服务端生成 `PDF`，首先得有一个服务器来放后端服务吧，但是自己又不想买服务器，碍于域名、备案等等一系列事情，想想就头大。所以最终我选择直接挂到 `serverless` 云函数上，在挑选了一通免费服务后我选择了 `Netlify Functions`

使用puppeteer实现PDF导出

进入正题，根据自己的项目需求，实现了一个大致的导出框架

代码实现

```js
const puppeteer = require("puppeteer");

exports.handler = async function (event, context) {
  const { content, style, link } = JSON.parse(event.body);
  // 开启一个浏览器进程
  const browser = await puppeteer.launch({ headless: "new" });
  // 打开一个页面
  const page = await browser.newPage();
  // 设置页面内容
  await page.setContent(
      `<html>
          <head>
              <meta charset="UTF-8" />
              <meta name="viewport" content="width=device-width, initial-scale=1.0" />
          </head>
          <body>${content}</body>
       </html>`
  );
  // 给页面添加样式
  await page.addStyleTag({ url: link });
  // 生成pdf
  const pdf = await page.pdf({
    width: 794,
    height: 1123,
    printBackground: true
  });
```

```
    statusCode: 200,
    body: JSON.stringify({
      msg: "导出成功~",
      pdf
    }),
  };
};

```

效果测试

以上代码测试过后没有任何问题，可以导出，就是速度不是很理想（但毕竟是免费的，还要什么自行车呢），测试没问题之后直接就部署该函数

serverless环境使用puppeteer报错

上线后使用导出功能结果抛出了一个错误："错误的启动浏览器进程..."

查阅文档会发现 `puppeteer` 依赖 `chromuim` 浏览器的环境，所以如果要使用它的话，那么前提是服务所处环境有 `chromium` 浏览器，那么我猜测在 `serverless` 环境是没有 `chromuim` 的，那怎么办？这个时候就需要给它提供一个环境了，怎么提供？

puppeteer执行环境配置

本地开发和线上部署使用不同配置

本地环境

在本地开发的时候，我们提供本机的 `chrome` 执行文件地址给 `puppeteer`，访问 **chrome://version**

线上环境

线上环境自定义 `chromuim` 路径，查阅了许多文档都说使用 `chrome-aws-lambda` 这个库可以实现在云上使用 `chromium`，但是经过我的多次尝试发现并不可行，`@sparticuz/chromium` 这个库是行得通的

调整代码

ok，准备就绪，现在需要对代码进行一些改动，同时还有一个需要注意的点，现在可以使用 `puppeteer-core` 来替代 `puppeteer`，因为 `puppeteer` 会自动下载与之匹配的 `chromium` 版本，但是 目前我们并不需要它帮我们下载了，我们自己提供给它，也就只需要使用它的核心代码部分即可

```js 复制代码
const chromium = require("@sparticuz/chromium");
const puppeteer = require("puppeteer-core");

const isDev = process.env.CONTEXT === "dev";

exports.handler = async function (event, context) {
// 省略不相关代码...
const browser = await puppeteer.launch({
      args: chromium.args,
      defaultViewport: chromium.defaultViewport,
      executablePath: isDev
        ? "/Applications/Google Chrome.app/Contents/MacOS/Google Chrome"
        : await chromium.executablePath(), // 生产环境使用我们自定义的chromium路径
      headless: chromium.headless,
    });
// 省略不相关代码...
};
```

`args` 参数用于传递命令行参数给 `chromium` 浏览器实例，以配置其行为和性能设置，而 `defaultViewport` 参数用于设置浏览器视口的默认大小，`executablePath` 用于指定 `chromium` 的执行路径。改动完之后部署上线再测试一波

测试过后没有问题，现在就可以愉快的白嫖了～

总结

其实看起来真的没多少内容，但是如果刚接触的朋友自己去实践这块内容会踩很多坑，社区的相关解决方案大多是过期的，现在已经是不实用了，我自己查阅了两天国内外的文档才解决这个头疼的问题，所以把这个方案也分享一下，有遇到相同问题的朋友可以参考一下，给这个 `bug` 画上一个完美的句号。

标签： 后端 Node.js Puppeteer 话题： 每天一个知识点

评论 0

登录 / 注册 即可发布评论！

暂无评论数据

目录

收起

相关推荐

20分钟玩转Puppeteer
2.3k阅读 · 40点赞

puppeteer基本用法
6.4k阅读 · 10点赞

Puppeteer 爬取豆瓣小组公开信息
2.0k阅读 · 9点赞

Puppeteer 爬取新年福字图片---助力你的扫福🧧大战
543阅读 · 6点赞

Linux服务器上运行Puppeteer的Docker部署指南
1.5k阅读 · 5点赞

精选内容

三十岁大厂程序员的反思：低级程序员的四个坏习惯，看看你中了几条?
东东拿铁 · 178阅读 · 4点赞

前端视角对Rust的浅析
转转技术团队 · 106阅读 · 3点赞

springboot集成nacos快速入门demo
HBLOG · 25阅读 · 0点赞

Java独有特性：注解(annotation)
shepherd111 · 109阅读 · 0点赞

为你推荐

puppeteer 生成pdf 全攻略
小野猪佩奇 | 2年前 | 6.2k | 16 | 8 前端

实践指南-网页生成PDF
凹凸实验室 | 2年前 | 8.8k | 140 | 36 前端

前端实现pdf转图片
Lgowen | 2年前 | 7.6k | 20 | 10 前端

记录一下puppeteer的使用
terminal007 | 1月前 | 1.0k | 5 | 评论 前端 JavaScript Node.js

前端在线预览Office文档
下落香樟树 | 2年前 | 9.1k | 45 | 16 前端

nodejs+nginx+puppeteer 实现pdf下载
我来滑水啦 | 2年前 | 1.5k | 点赞 | 评论 Docker

所见即所得 —— HTML转图片组件开发
政采云技术 | 1年前 | 12k | 64 | 13 Canvas HTML

无头浏览器与Puppeteer中PDF生成应用指南
AlexGeek | 2年前 | 4.5k | 39 | 2 JavaSc... 前端

nuxt3搭建中间层服务html生成PDF方案： 基于nuxt3 + puppeteer
赛博丁真Damon | 1年前 | 2.4k | 38 | 12 Nuxt.js 前端 Vue.js

HTML页面导出为PDF完整指南（实现篇）
码云之上 | 6月前 | 1.6k | 12 | 评论 前端 Puppet... Express

html转pdf
杨小芸 | 3年前 | 3.6k | 18 | 13 前端

纯前端生成PDF之jspdf使用及注意事项
zkat | 9月前 | 10k | 31 | 20 前端 JavaScript

HTML转PDF
Keranalice | 2年前 | 1.3k | 3 | 评论 JavaScript

Java端网页PDF生成方案（Chrome无头浏览器）
幕后煮屎者 | 2年前 | 2.8k | 4 | 评论 前端 Java

Puppeteer生成pdf
Harryqi | 3年前 | 1.9k | 1 | 2 Puppeteer