

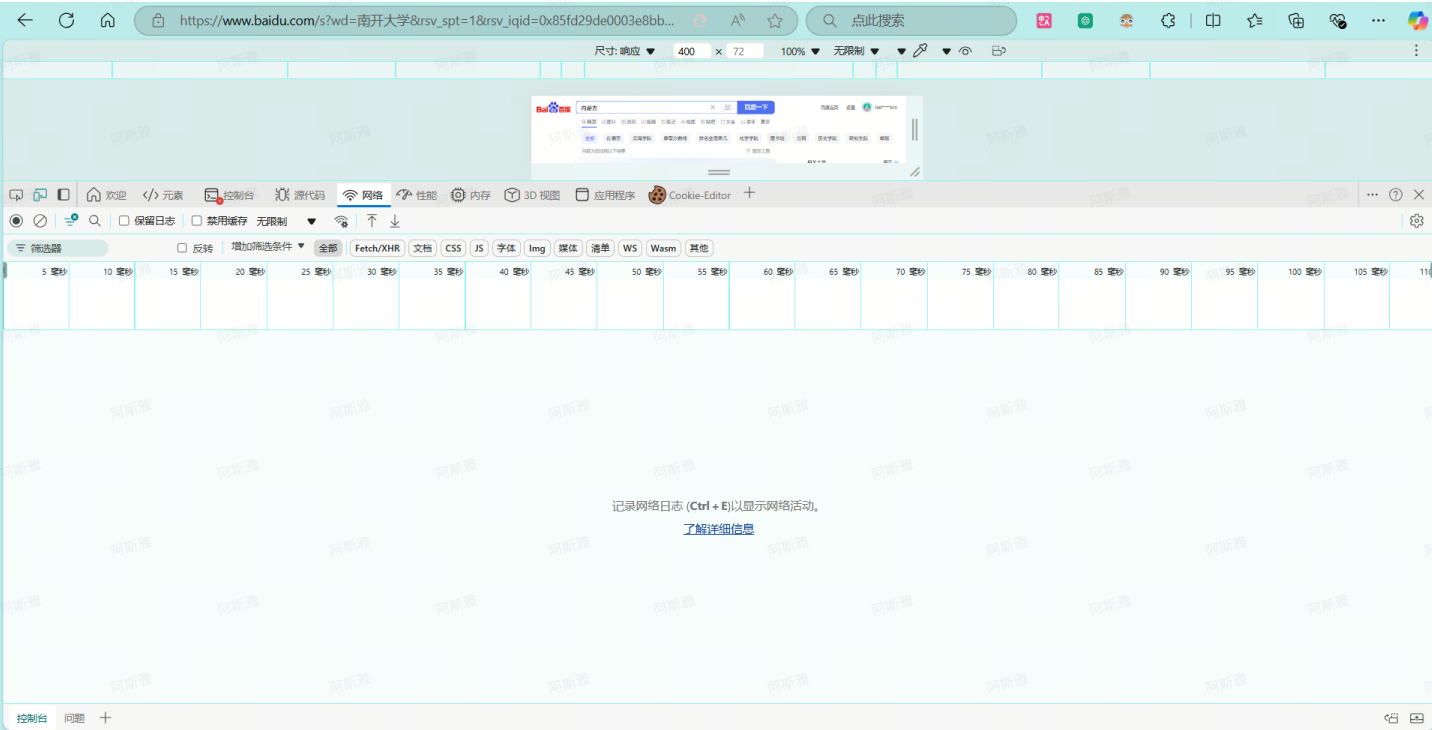
前端初探

学号：2210737 姓名：阿斯雅

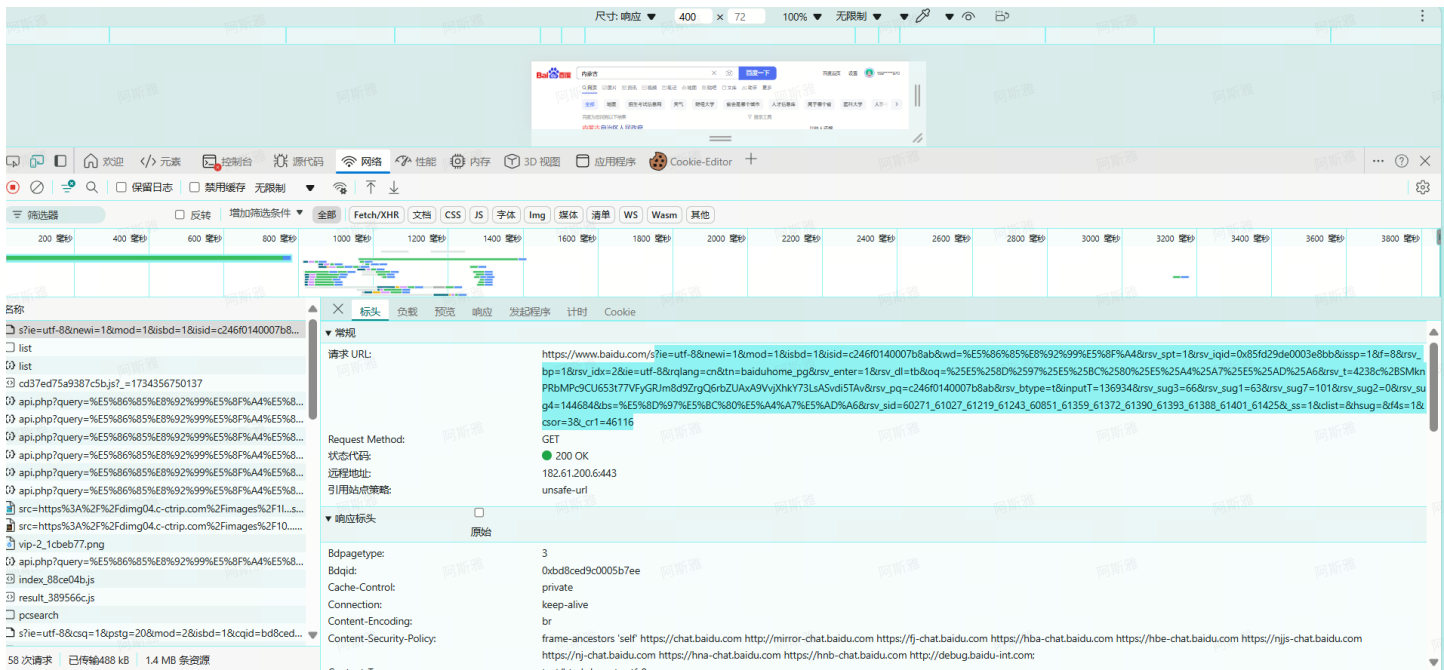
一、调研不同网络请求

1.1、GET请求

我们可以在百度中搜索内蒙古，然后在F12中监听网络事件。

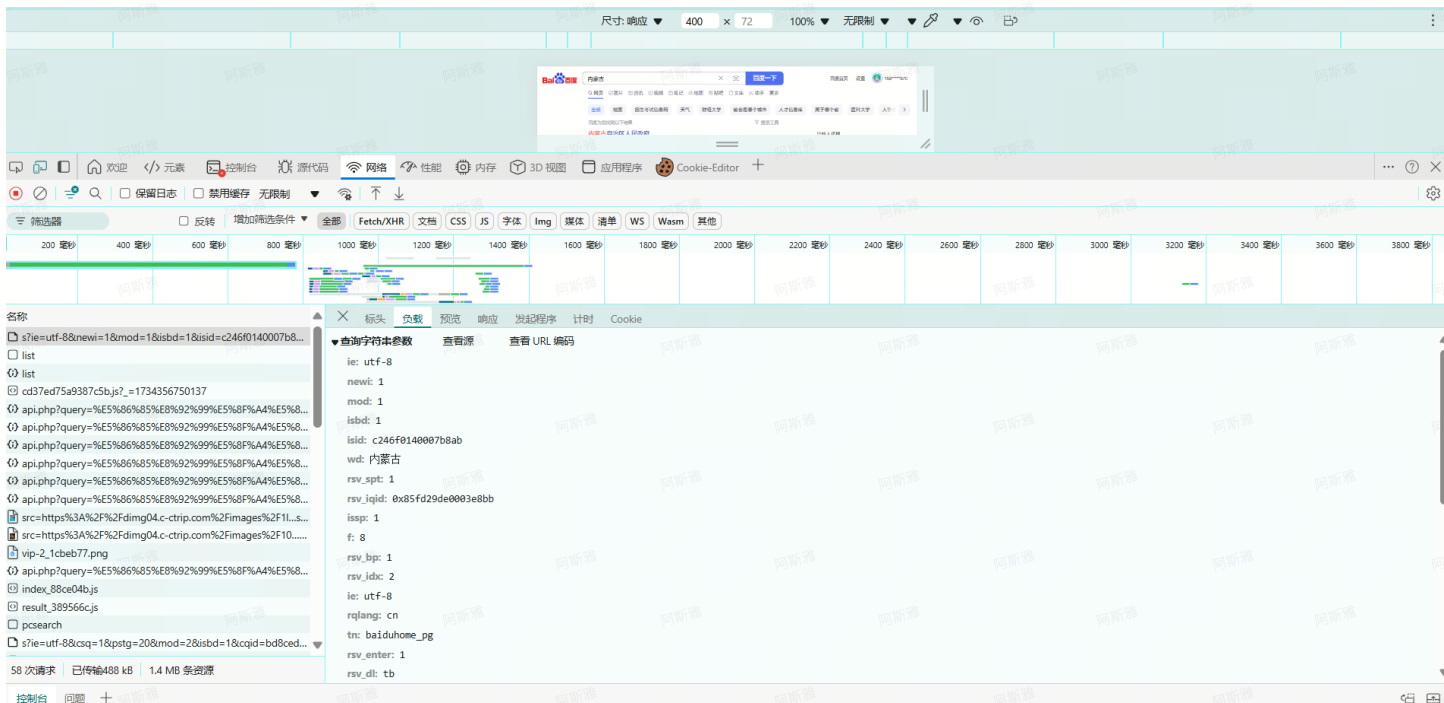


捕获到的数据包如下：

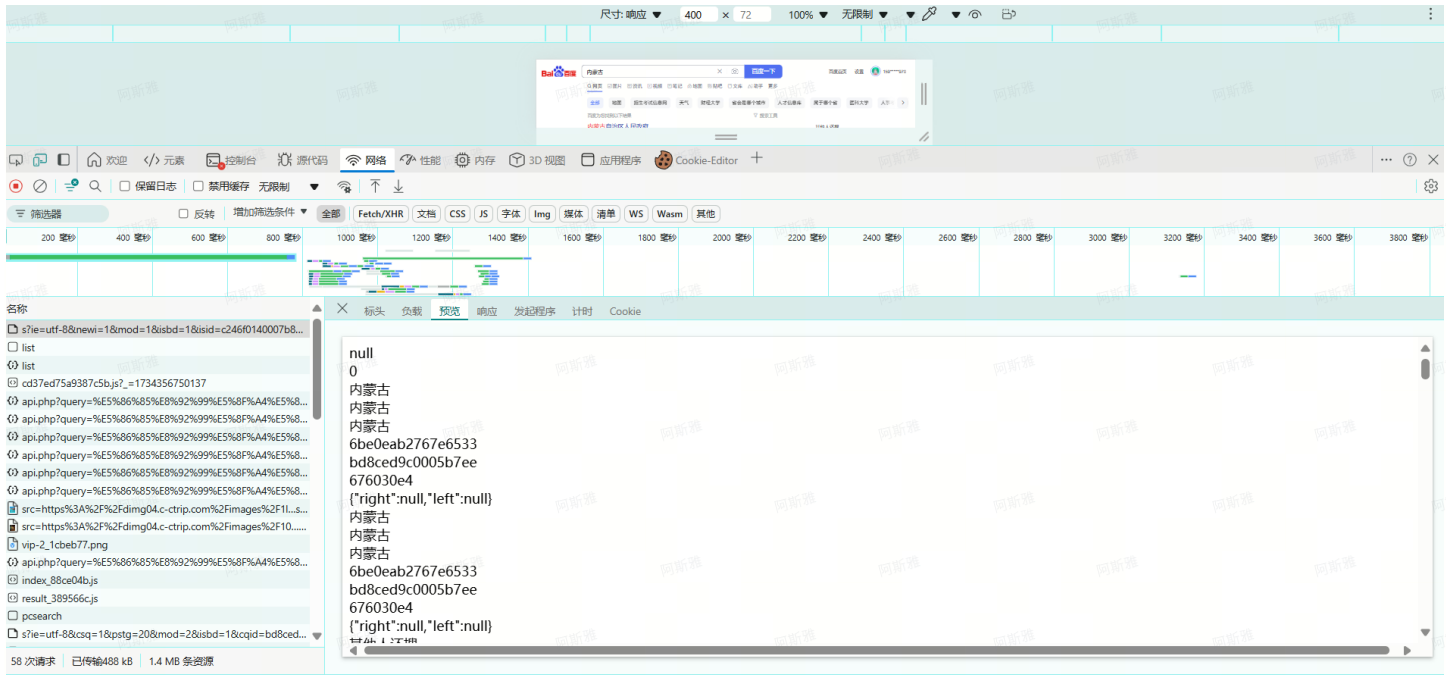


可以发现这个网络请求是get方式：在url后面拼接参数，并且只能以文本的形式传递参数，安全性低，将信息显示在地址栏。

接着可以查看每个URL参数的意义：

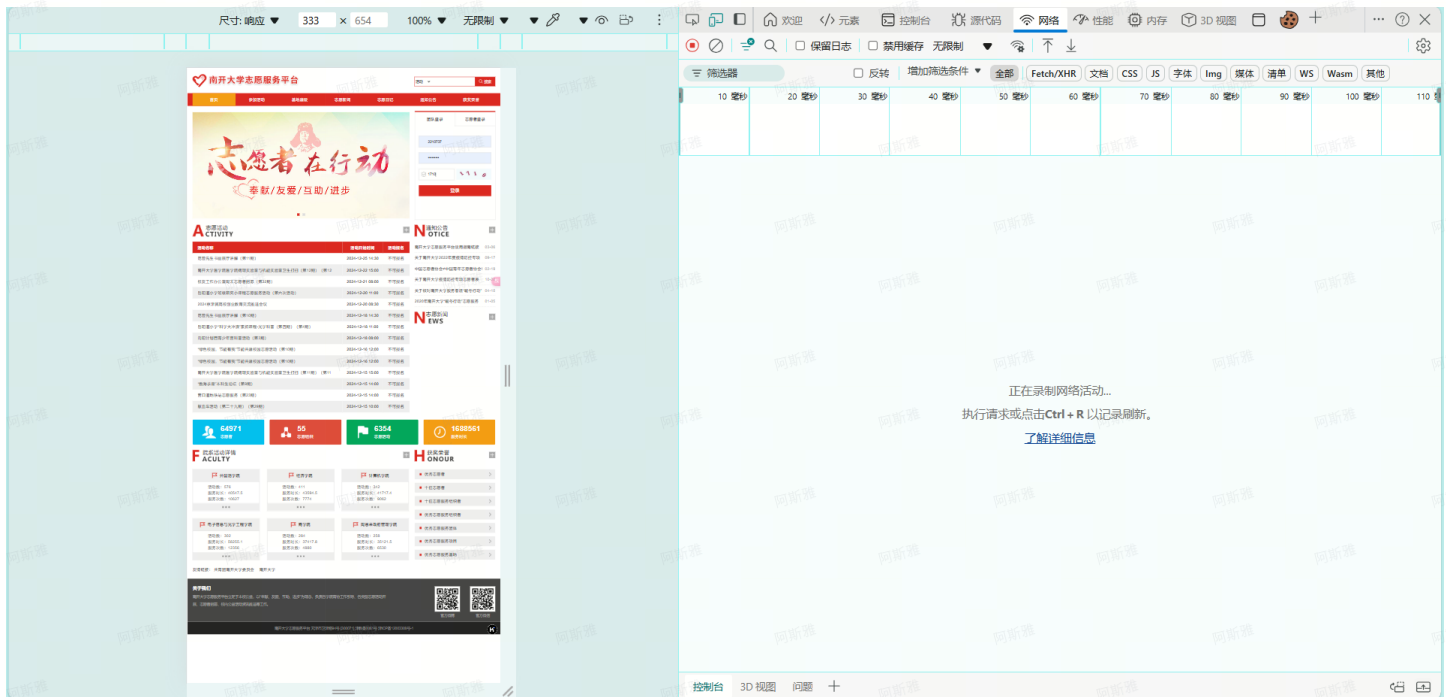


产生的数据包：

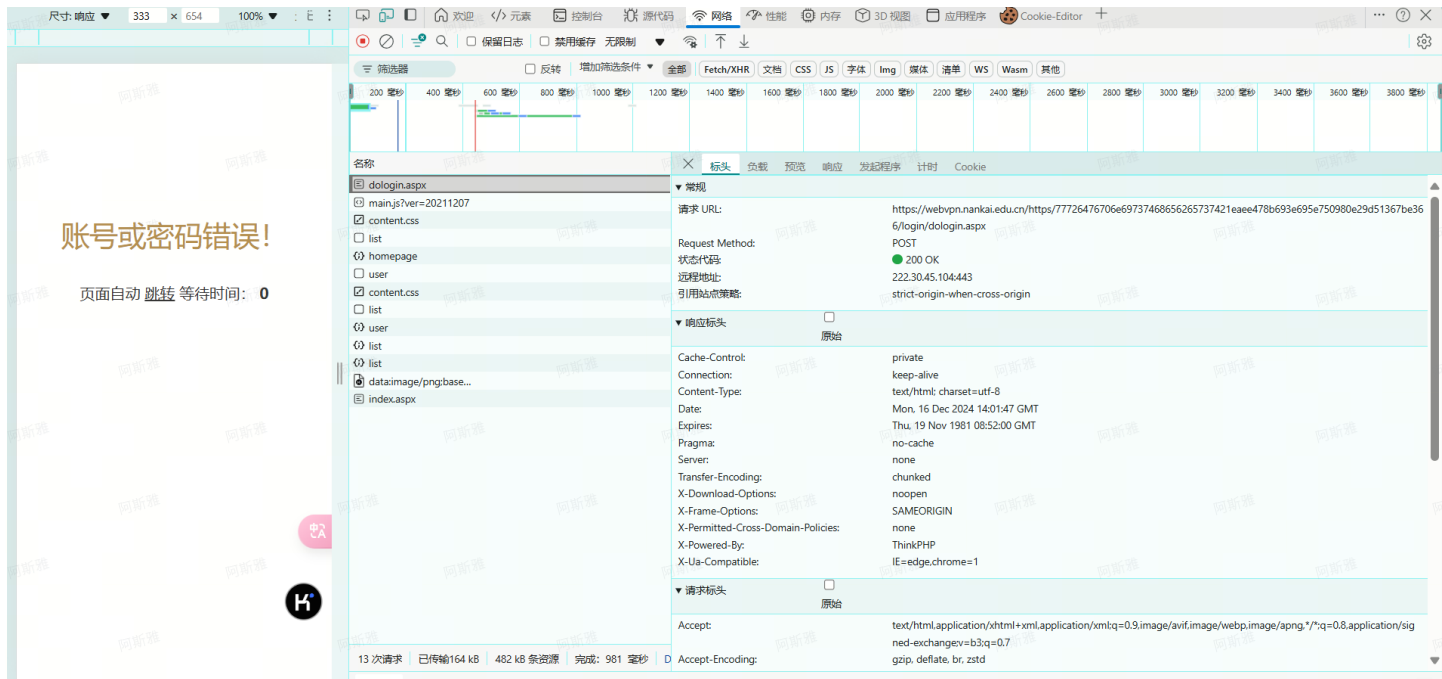


1.2、POST请求

接着登录南开大学志愿服务平台。



进行登录



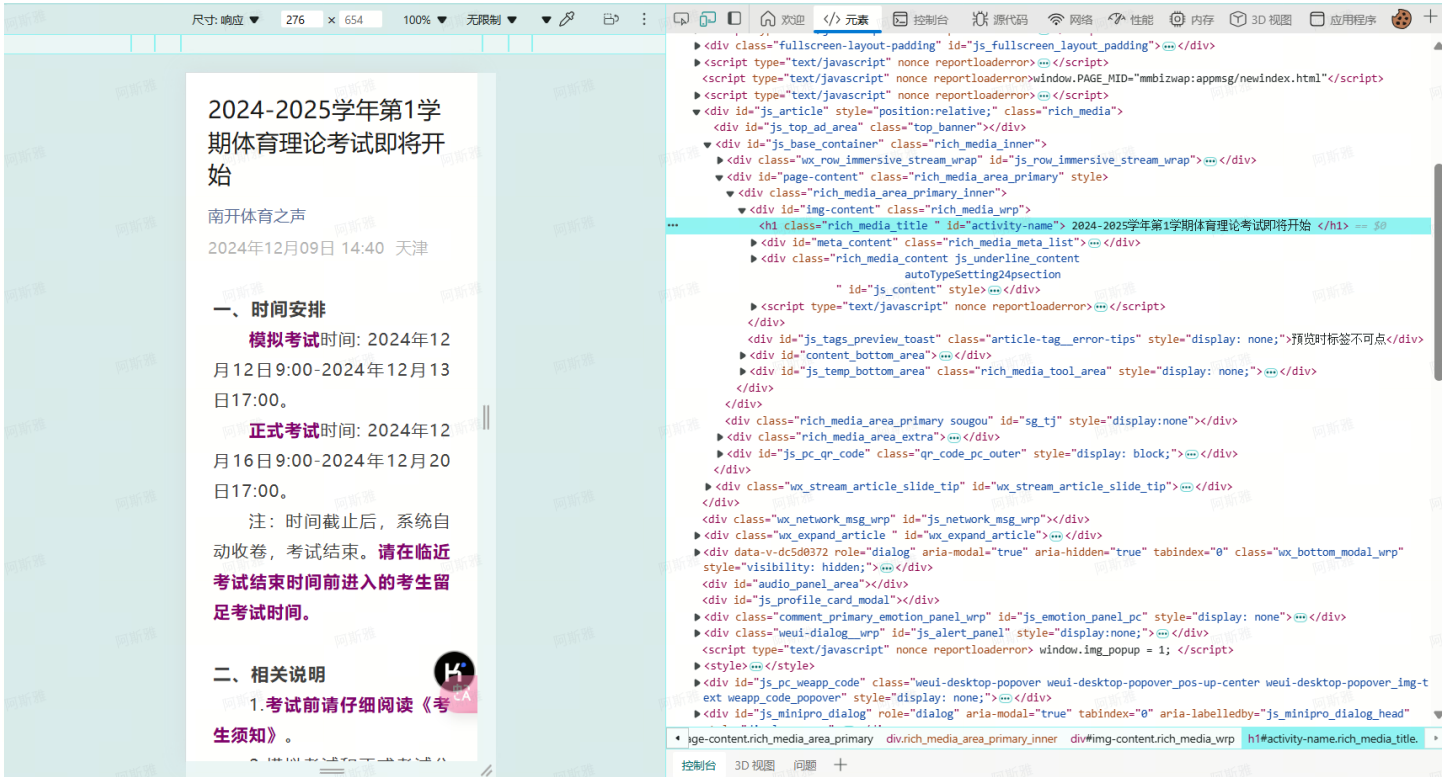
可以发现请求方式是POST，并且用户并不能直接从URL中看到有效信息。

二、初探jquery

2.1、修改文章标题

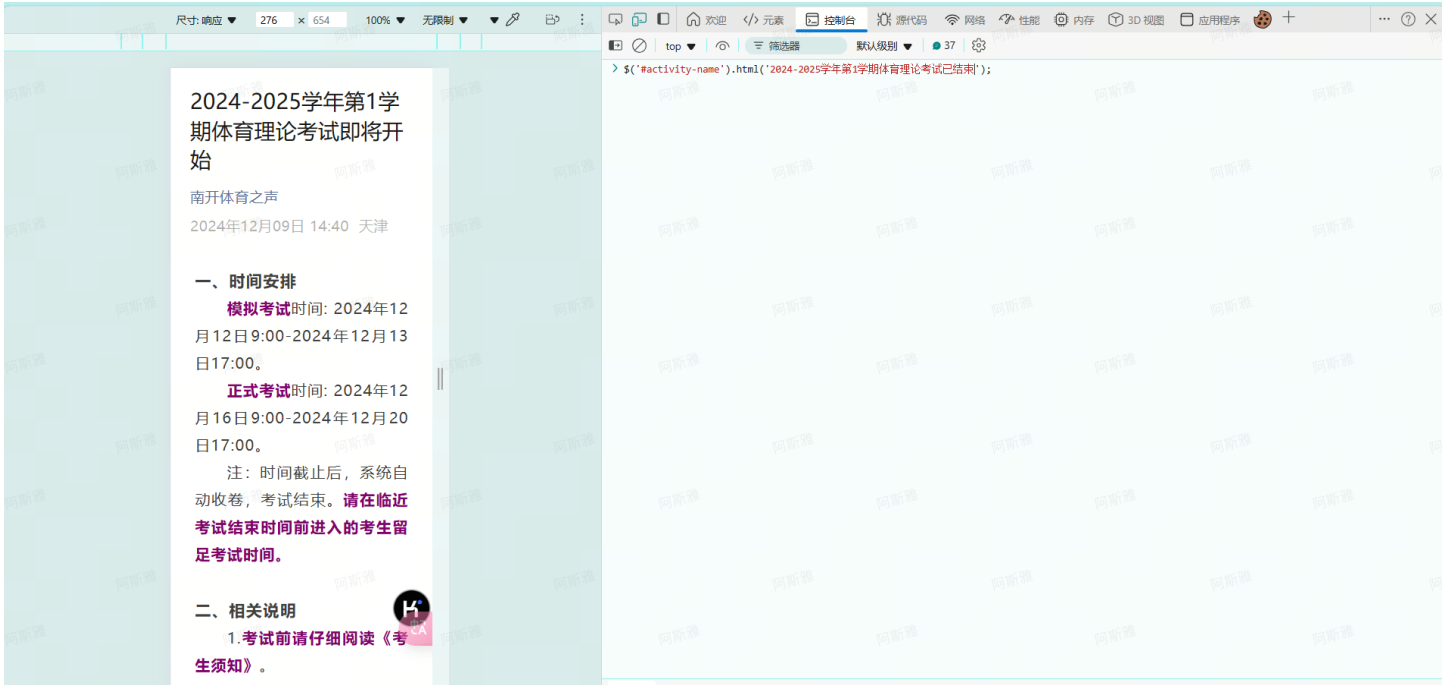
可以发现这个公众号文章的主题

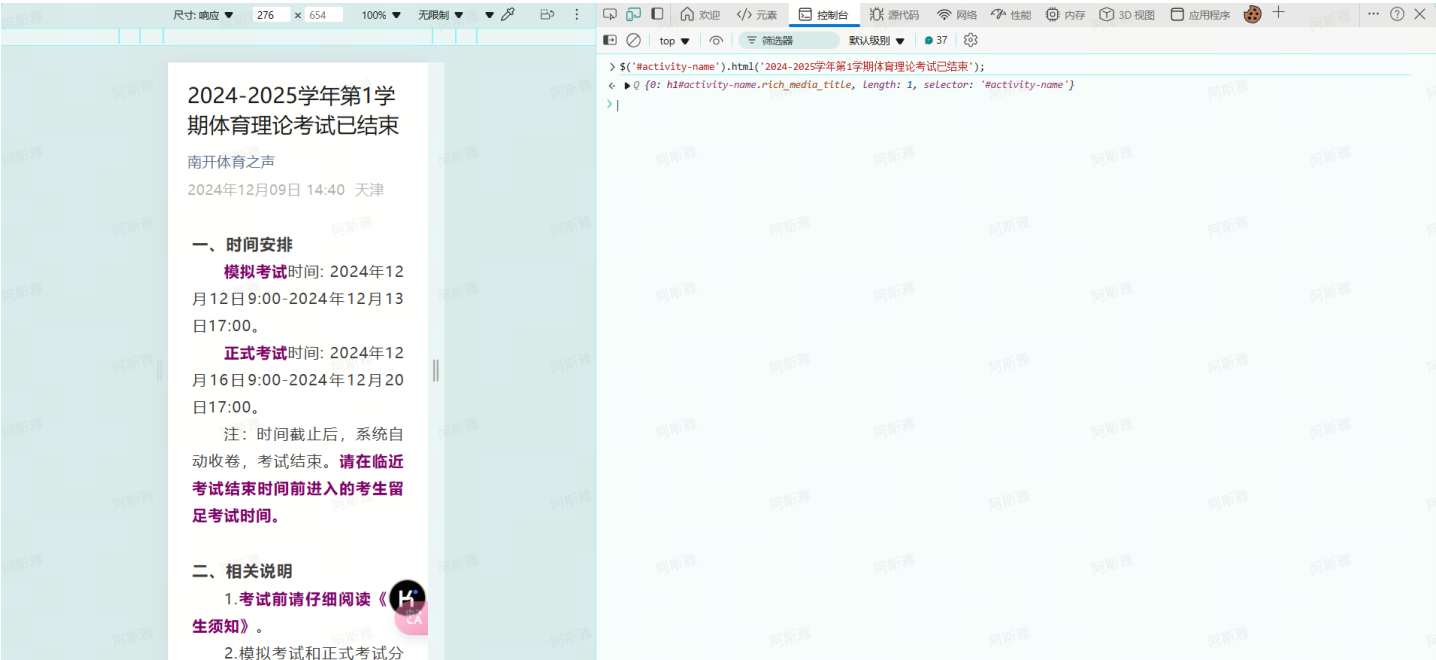




所以可以写一个jquery语句把这个标题给修改。

```
1 $('#activity-name').html('2024-2025学年第1学期体育理论考试已结束');
```

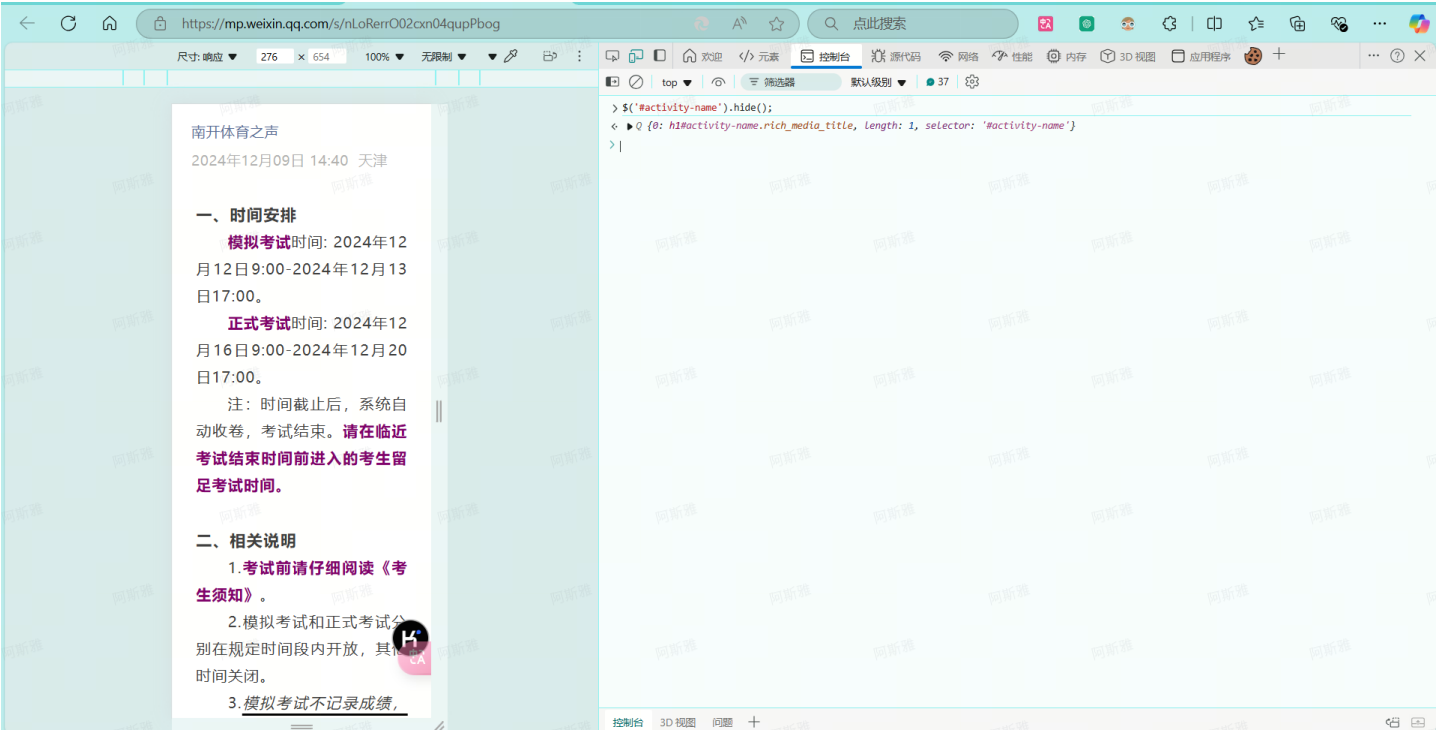




2.2、隐藏文章标题

接着我们可以使用hide命令隐藏这个标题：

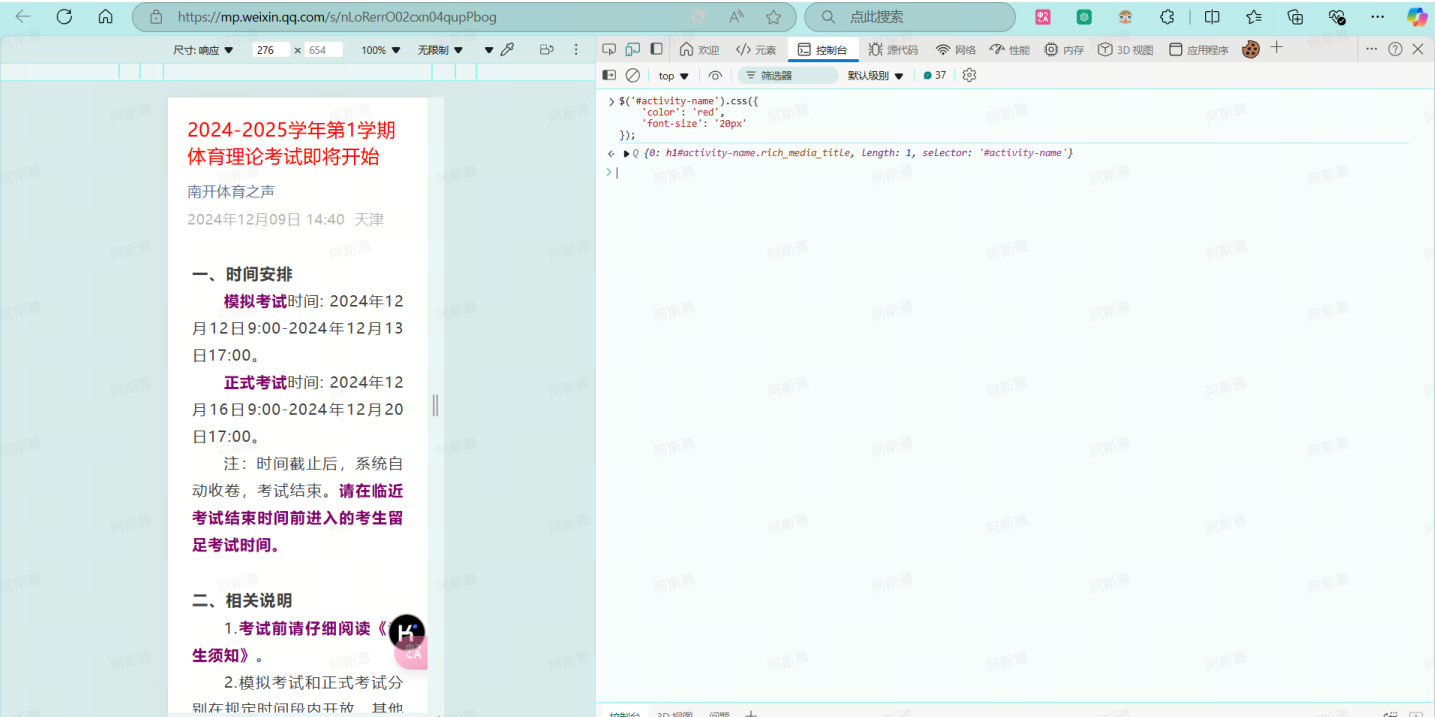
```
1 $('#activity-name').hide();
```



2.3、修改标题格式

最后我们可以使用.css命令修改这个标题的显示格式：


```
1 $('#activity-name').css({
2   'color': 'red',
3   'font-size': '20px'
4 });
```



三、浏览器插件设计

我本次做的是实现全局截图的小插件，平台是谷歌浏览器。


3.1、Background Script

 这个脚本负责插件的核心功能和事件处理，它通常在浏览器加载时就启动，并且在插件的生命周期内持续运行。Background script 负责处理一些全局事件、监听浏览器的特定行为（如标签页变化、浏览器请求、消息传递等），以及与其他脚本（如content script）进行通信。Background script 常常用于处理插件与浏览器 API 的交互。

```
1 chrome.action.onClicked.addListener((tab) => {
2     chrome.tabs.captureVisibleTab(null, { format: "png" }, (image) => {
3         const link = document.createElement('a');
4         link.href = image;
5         link.download = 'screenshot.png';
6         link.click();
7     });
8 });
```

这段代码用于 Chrome 浏览器扩展的 JavaScript 代码。它监听用户点击插件图标的事件，并在用户点击时执行一段操作，这段操作是抓取当前可见的标签页的截图并将其下载为 PNG 图片。

3.2、Popup Script

 Popup script 是插件的弹出窗口（通常是插件图标点击后出现的小窗口）的 JavaScript 文件。这个脚本控制了插件弹出窗口中的内容和行为，比如用户点击按钮时的响应、动态加载数据等。Popup 脚本通常用于处理插件图标的交互逻辑，如打开弹出窗口时加载数据或处理用户的输入。

```
1 document.getElementById('capture').addEventListener('click', () => {
2     chrome.tabs.captureVisibleTab(null, { format: "png" }, (image) => {
3         const link = document.createElement('a');
4         link.href = image;
5         link.download = 'screenshot.png';
6         link.click();
7     });
8 });
```

这段代码是用来监听网页中的一个按钮点击事件，当按钮被点击时，它会调用 Chrome 扩展的 API 来抓取当前可见标签页的屏幕截图，并将该截图保存为 PNG 格式的文件。

3.3、Manifest Script



`manifest.json` 是浏览器扩展（如 Chrome 扩展）的配置文件，定义了扩展的基本信息、权限、功能和行为。它是每个浏览器扩展项目的核心文件，所有的扩展必须包含一个 `manifest.json` 文件，否则扩展无法被加载或运行。

```
1 {
2   "manifest_version": 3,
3   "name": "Full Screen Screenshot",
4   "version": "1.0",
5   "description": "Capture a full screen screenshot and download it.",
6   "permissions": [ "tabs", "activeTab", "storage" ],
7   "background": {
8     "service_worker": "background.js"
9   },
10  "action": {
11    "default_popup": "popup.html",
12    "default_title": "Capture Screenshot"
13  },
14  "icons": {
15    "48": "icon.png"
16  }
17 }
```

这个 `manifest.json` 文件定义了一个名为 “**Full Screen Screenshot**” 的 Chrome 扩展，旨在捕获屏幕截图并将其下载为 PNG 文件。它使用 **Manifest V3** 版本，指定了扩展的基本信息，包括名称、版本号和描述。扩展需要的权限包括 `tabs`、`activeTab` 和 `storage`，以便访问标签页信息、与当前活动标签页交互以及存储数据。

扩展的主要功能通过一个浏览器工具栏图标提供，用户点击图标时会弹出一个名为 `popup.html` 的用户界面，供用户操作截图功能。扩展的后台逻辑由一个 **Service Worker**（`background.js`）处理，而图标则通过 `48x48` 像素的图标文件 `icon.png` 进行显示。整体来说，这个 `manifest.json` 文件配置了扩展的基本行为、权限、图标及其后台逻辑，是该扩展正常运行的核心配置。

3.4、POPUP HTML



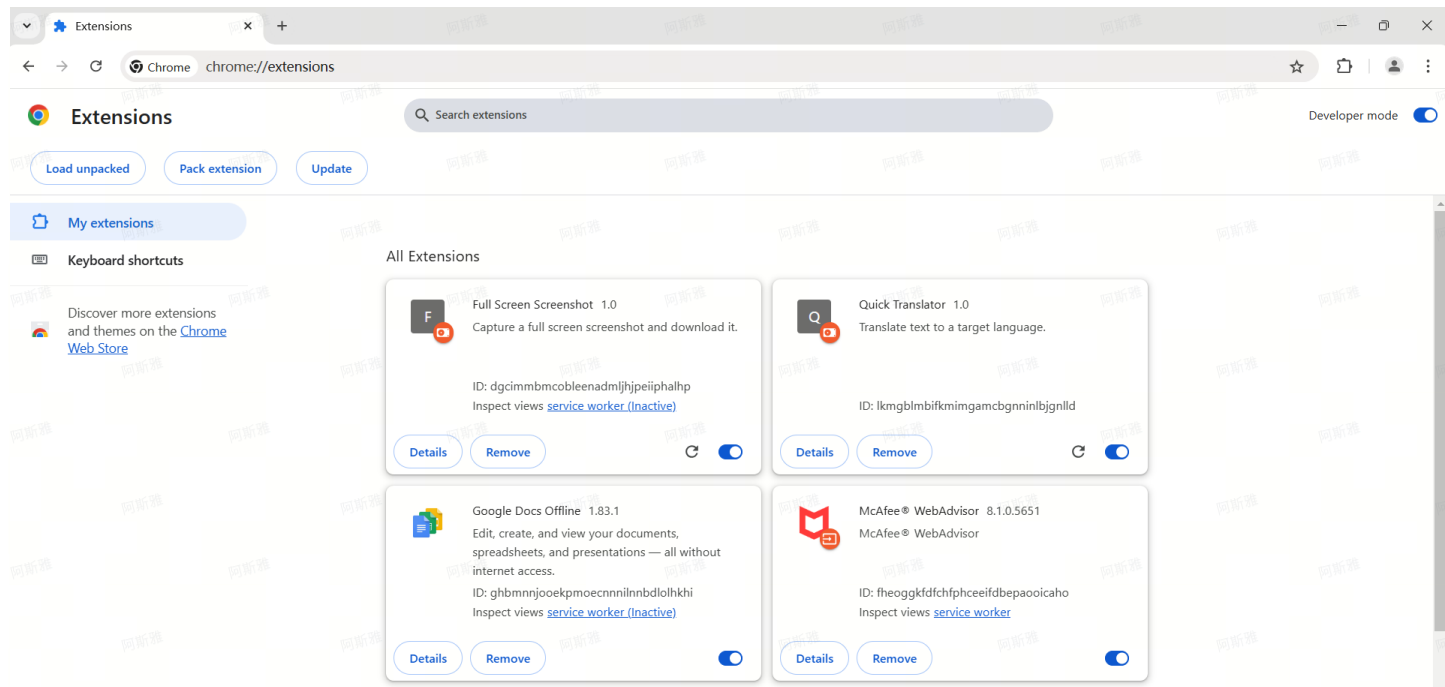
Popup HTML 是指在浏览器扩展中，当用户点击扩展图标时，弹出的小窗口（通常称为弹出界面）所使用的 HTML 文件。这个文件定义了扩展的用户界面，通常包含交互元素，如按

钮、输入框、文本显示区域等，允许用户与扩展进行交互。

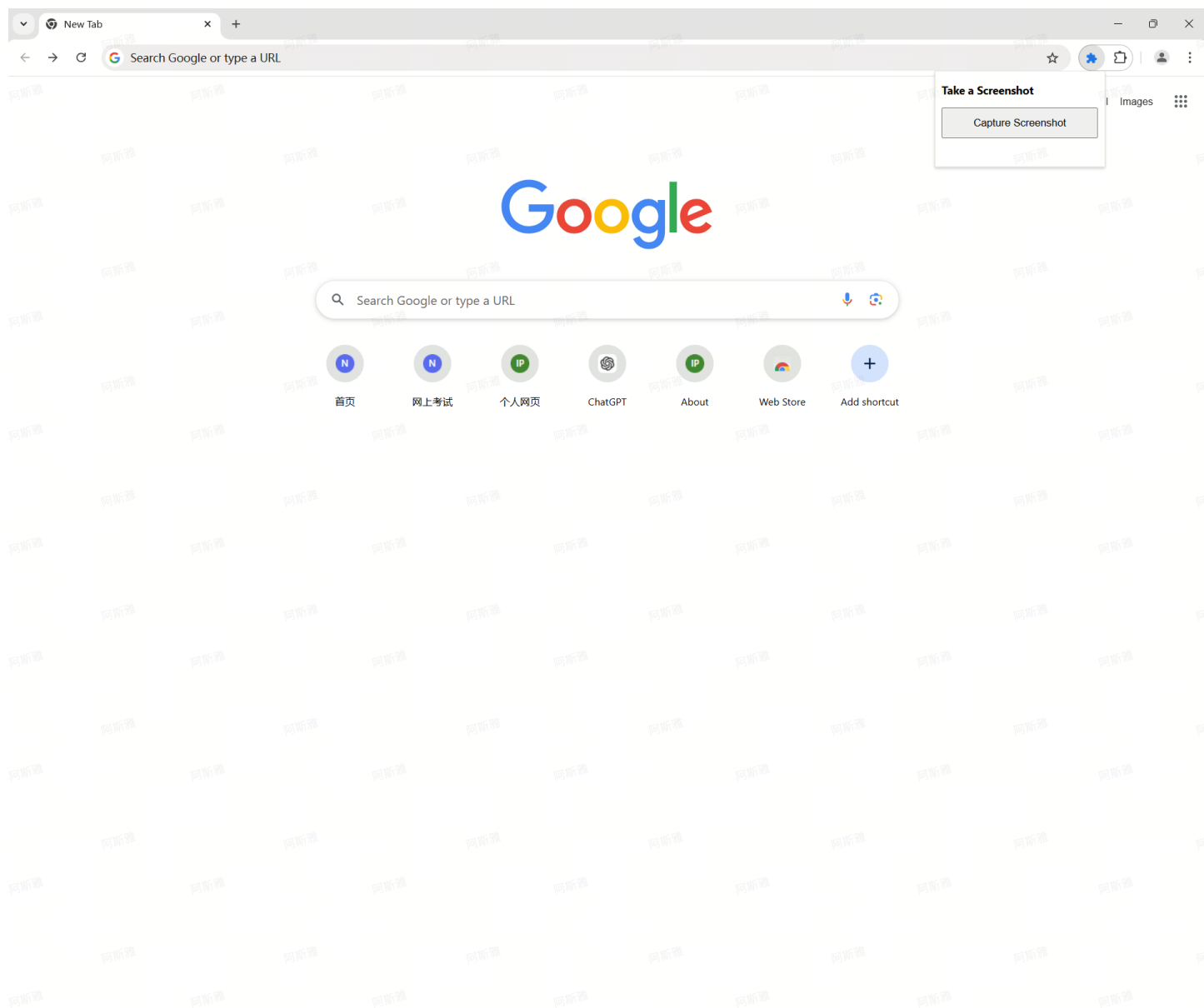
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Capture Screenshot</title>
5   <style>
6     body {
7       min-width: 200px;
8       min-height: 100px;
9     }
10
11     button {
12       padding: 10px;
13       width: 100%;
14     }
15   </style>
16 </head>
17 <body>
18   <h3>Take a Screenshot</h3>
19   <button id="capture">Capture Screenshot</button>
20   <script src="popup.js"></script>
21 </body>
22 </html>
```

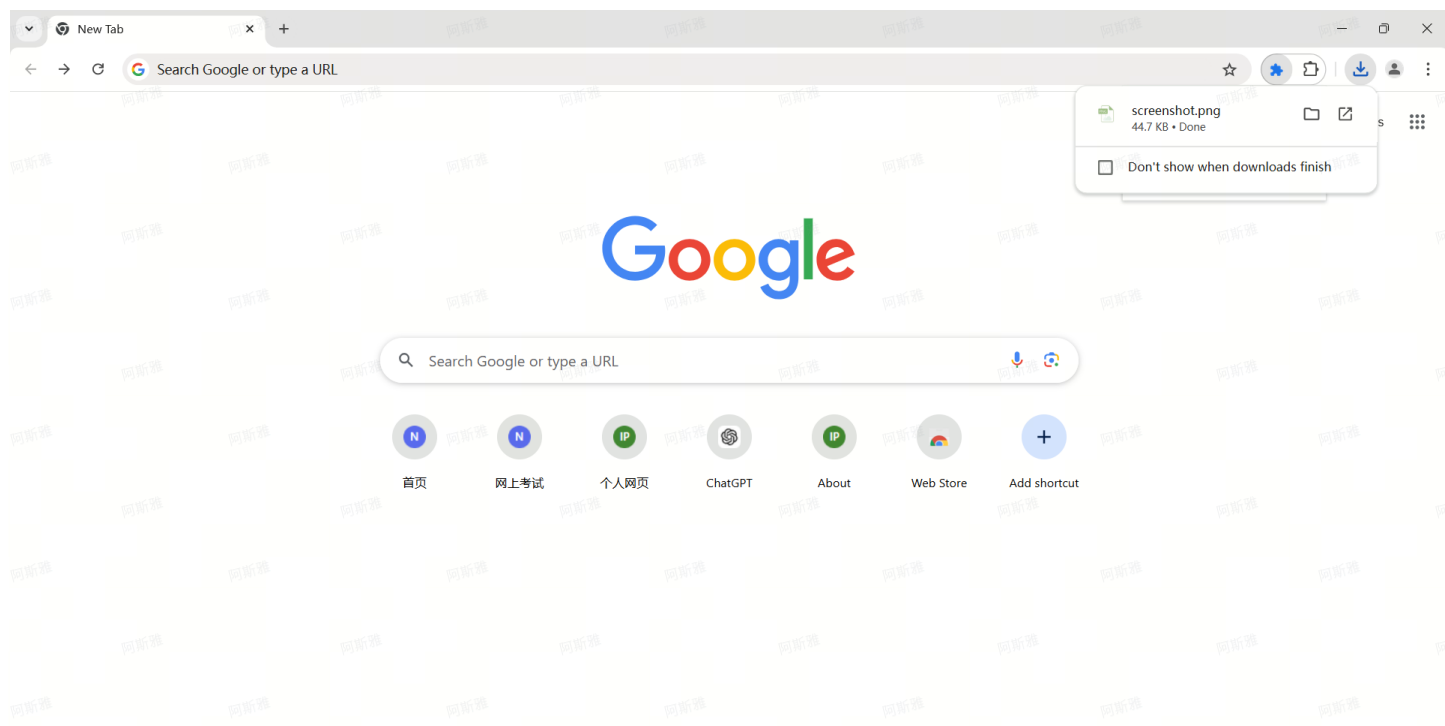
3.5、功能展示

首先我们在谷歌扩展中加载我们自定义的插件。



然后使用插件





打开图片文件后可以发现是要获得的截图

