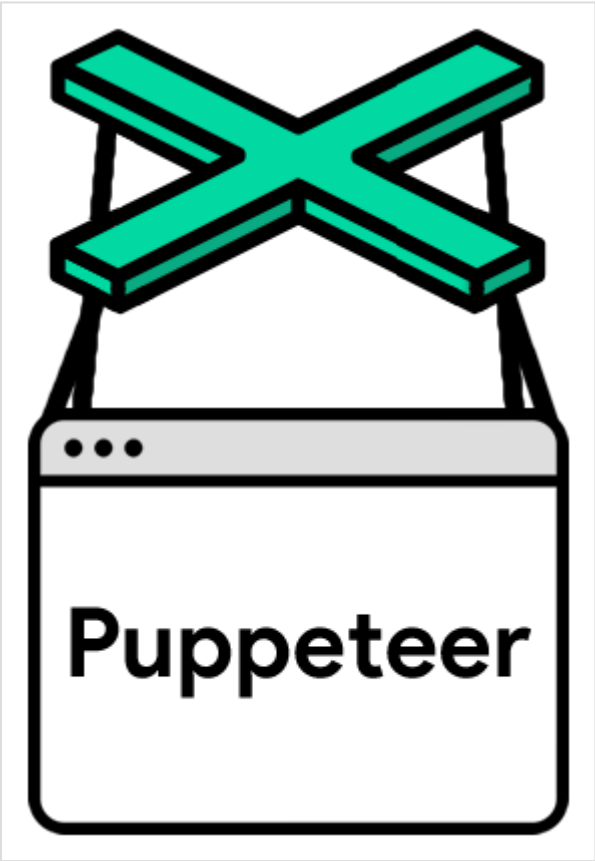
Puppeteer的入门教程和实践

原　puppeteer　chrome　javascript　教程　任乃千　1月21日发布

出现的背景

Chrome59(linux、macos)、 Chrome60(windows)之后，Chrome自带headless(无界面)模式很方便做自动化测试或者爬虫。但是如何和headless模式的Chrome交互则是一个问题。通过启动Chrome时的命令行参数仅能实现简易的启动时初始化操作。Selenium、Webdriver等是一种解决方案，但是往往依赖众多，不够扁平。



Puppeteer是谷歌官方出品的一个通过DevTools协议控制headless Chrome的Node库。可以通过Puppeteer的提供的api直接控制Chrome模拟大部分用户操作来进行UI Test或者作为爬虫访问页面来收集数据。

环境和安装

Puppeteer本身依赖6.4以上的Node，但是为了异步超级好用的async/await，推荐使用7.6版本以上的Node。另外headless Chrome本身对服务器依赖的库的版本要求比较高，centos服务器依赖偏稳定，v6很难使用headless Chrome，提升依赖版本可能出现各种服务器问题（包括且不限于无法使用ssh），最好使用高版本服务器。

Puppeteer因为是一个npm的包，所以安装很简单：

```
npm i puppeteer
```

> yarn add puppeteer

Puppeteer安装时自带一个最新版本的Chromium，可以通过设置环境变量或者npm config中的PUPPETEER_SKIP_CHROMIUM_DOWNLOAD跳过下载。如果不下载的话，启动时可以通过puppeteer.launch([options])配置项中的executablePath指定Chromium的位置。

使用和例子

Puppeteer类似其他框架，通过操作Browser实例来操作浏览器作出相应的反应。

```
const puppeteer = require('puppeteer');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('http://rennaiqian.com');
  await page.screenshot({path: 'example.png'});
  await page.pdf({path: 'example.pdf', format: 'A4'});
  await browser.close();
})();
```

上述代码通过puppeteer的launch方法生成了一个browser的实例，对应于浏览器，launch方法可以传入配置项，比较有用的是在本地调试时传入{headless:false}可以关闭headless模式。

```
const browser = await puppeteer.launch({headless:false})
```

browser.newPage方法可以打开一个新选项卡并返回选项卡的实例page，通过page上的各种方法可以对页面进行常用操作。上述代码就进行了截屏和打印pdf的操作。

一个很强大的方法是page.evaluate(pageFunction, ...args)，可以向页面注入我们的函数，这样就有了无限可能

```
const puppeteer = require('puppeteer');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('http://rennaiqian.com');

  // Get the "viewport" of the page, as reported by the page.
  const dimensions = await page.evaluate(() => {
    return {
      width: document.documentElement.clientWidth,
      height: document.documentElement.clientHeight,
      deviceScaleFactor: window.devicePixelRatio
    };
  });

  console.log('Dimensions:', dimensions);
  await browser.close();
})();
```

需要注意的是evaluate方法中是无法直接使用外部的变量的，需要作为参数传入，想要获得执行的结果也需要return出来。因为是一个开源一个多月的项目，现在项目很活跃，所以使用时自行查找api才能保证参数、使用方法不会错。

调试技巧

1. 关掉无界面模式，有时查看浏览器显示的内容是很有用的。使用以下命令可以启动完整版浏览器：

```
const browser = await puppeteer.launch({headless: false})
```

1. 减慢速度，slowMo选项以指定的毫秒减慢Puppeteer的操作。这是另一个看到发生了什么的方法：

```
const browser = await puppeteer.launch({
  headless:false,
  slowMo:250
});
```

3.捕获console的输出,通过监听console事件。在page.evaluate里调试代码时这也很方便：

```
page.on('console', msg => console.log('PAGE LOG:', ...msg.args));
await page.evaluate(() => console.log(`url is ${location.href}`));
```

4.启动详细日志记录，所有公共API调用和内部协议流量都将通过puppeteer命名空间下的debug模块进行记录

```
 # Basic verbose logging
 env DEBUG="puppeteer:*" node script.js

 # Debug output can be enabled/disabled by namespace
 env DEBUG="puppeteer:*,-puppeteer:protocol" node script.js # everything BUT protocol messages
 env DEBUG="puppeteer:session" node script.js # protocol session messages (protocol messages to
 env DEBUG="puppeteer:mouse,puppeteer:keyboard" node script.js # only Mouse and Keyboard API cal

 # Protocol traffic can be rather noisy. This example filters out all Network domain messages
 env DEBUG="puppeteer:*" env DEBUG_COLORS=true node script.js 2>&1 | grep -v '"Network'
```

爬虫实践

很多网页通过user-agent来判断设备，可以通过page.emulate(options)来进行模拟。options有两个配置项，一个为userAgent，另一个为viewport可以设置宽度(width)、高度(height)、屏幕缩放(deviceScaleFactor)、是否是移动端(isMobile)、有无touch事件(hasTouch)。

```
const puppeteer = require('puppeteer');
const devices = require('puppeteer/DeviceDescriptors');
const iPhone = devices['iPhone 6'];

puppeteer.launch().then(async browser => {
  const page = await browser.newPage();
  await page.emulate(iPhone);
  await page.goto('https://www.example.com');
```

```
  await browser.close();
});
```

上述代码则模拟了iPhone6访问某网站，其中devices是puppeteer内置的一些常见设备的模拟参数。

很多网页需要登录，有两种解决方案：

1. 让puppeteer去输入账号密码

常用方法：点击可以使用page.click(selector[, options])方法，也可以选择聚焦page.focus(selector)。
输入可以使用page.type(selector, text[, options])输入指定的字符串，还可以在options中设置delay缓慢输入更像真人一些。
也可以使用keyboard.down(key[, options])来一个字符一个字符的输入。

1. 如果是通过cookie判断登录状态的可以通过page.setCookie(...cookies)，想要维持cookie可以定时访问。

Tip：有些网站需要扫码，但是相同域名的其他网页却有登录，就可以尝试去可以登录的网页登录完利用cookie访问跳过扫码。

简单例子

```
const puppeteer = require('puppeteer');

(async () => {
  const browser = await puppeteer.launch({headless: false});
  const page = await browser.newPage();
  await page.goto('https://baidu.com');
  await page.type('#kw', 'puppeteer', {delay: 100});
  page.click('#su')
  await page.waitFor(1000);
  const targetLink = await page.evaluate(() => {
    return [...document.querySelectorAll('.result a')].filter(item => {
      return item.innerText && item.innerText.includes('Puppeteer的入门和实践')
    }).toString()
  });
  await page.goto(targetLink);
  await page.waitFor(1000);
  browser.close();
})()
```

求赞，另外欢迎访问我的博客

1月21日发布 ···

前端学习资源整理 129 收藏，2.3k 浏览

前端资源系列（4）-前端学习资源分享&前端面试资源汇总 1090 收藏，30.3k 浏览

免费开源图书 59 收藏，1.3k 浏览

评论

默认排序 | 时间排序

文明社会，理性评论

发布评论

任乃千

145 声望

关注作者

发布于专栏

任乃千的专栏

分享一些前端方面的知识

0 人关注

关注专栏