目录

跨平台Web自动化神器：Playwright

- 最新版本：`v2.0`
- 更新时间：`20210730`

简介

介绍支持跨平台的Web自动化神器，playwright。先进行概览介绍；如何初始化搭建Playwright开发环境；在介绍一些基本操作，包括查找和定位元素、查找并直接点击元素、输入文字、等待元素出现、模拟按键输入、获取元素属性等；在列出常见的问题和心得，包括如何初始化Sync的playwright的实例、using Sync API inside the asyncio loop、所有页面加载都超时、browser was not found、给playwright加代理等；以及给出详尽的例子，包括百度搜索自动化、获取谷歌搜索结果以及其他一些小例子；最后给出教程和资料。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

Gitbook源码

- crifan/web_automation_tool_playwright: 跨平台Web自动化神器：Playwright

如何使用此Gitbook源码去生成发布为电子书

详见：crifan/gitbook_template: demo how to use crifan gitbook template and demo

在线浏览

- 跨平台Web自动化神器：Playwright book.crifan.com
- 跨平台Web自动化神器：Playwright crifan.github.io

离线下载阅读

- 跨平台Web自动化神器：Playwright PDF
- 跨平台Web自动化神器：Playwright ePub
- 跨平台Web自动化神器：Playwright Mobi

版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

更多其他电子书

本人 `crifan` 还写了其他 `100+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

Playwright概览

- `Playwright`
  - 一句话简介
    - 中文：微软开源的 `Python` 自动化神器 `Playwright`
    - 英文：Node.js library to automate Chromium, Firefox and WebKit with a single API
  - 特点
    - 绿色环保ever-green
    - 能力强capable
    - 可靠性高reliable
    - 速度快fast
    - 跨平台==支持多个系统 + 支持多种浏览器（内核）

	Linux	macOS	Windows
Chromium 90.0.4430.0	✅	✅	✅
WebKit 14.2	✅	✅	✅
Firefox 87.0b10	✅	✅	✅

      - 跨平台
        - `Windows`
        - `MacOS`
        - `Linux`
      - 支持多种浏览器（内核）
        - `Chromium`
        - `Firefox`
        - `WebKit`
  - 说明
    - 微软新出的 `Python` 库，仅用一个API即可自动执行 `Chromium` 、 `Firefox` 、 `WebKit` 等主流浏览器自动化操作
    - 微软公司2020年初发布的新一代自动化测试工具，相较于目前最常用的Selenium，它仅用一个API即可自动执行Chromium、Firefox、WebKit等主流浏览器自动化操作。作为针对Python语言纯自动化的工具，在回归测试中可更快的实现自动化。

初始化环境

初始化Playwright开发环境

Mac

安装playwright

```
pip install playwright
```

初始化安装web的driver

```
playwright install
```

- 或

  ```
  python -m playwright install
  ```

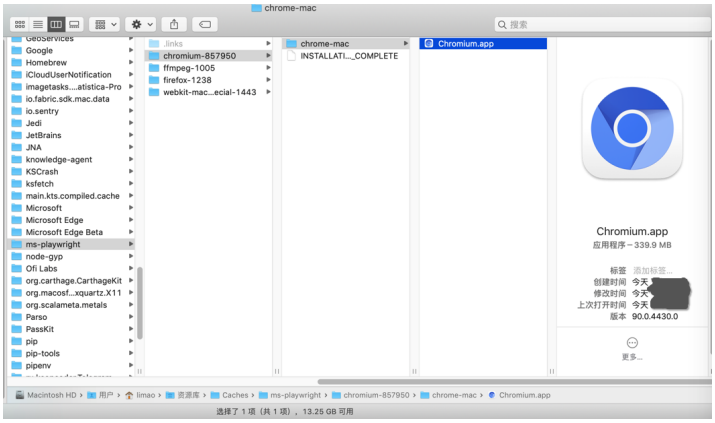
注：安装浏览器驱动文件（安装过程稍微有点慢）

详细日志

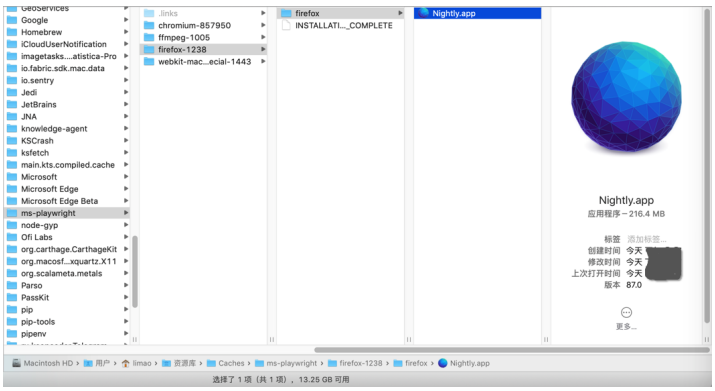
```
□ playwright install
Downloading chromium v857950 - 113.9 Mb [==================
chromium v857950 downloaded to /Users/limao/Library/Caches/
Downloading firefox v1238 - 75 Mb [====================] 10
firefox v1238 downloaded to /Users/limao/Library/Caches/ms-
Downloading webkit v1443 - 52 Mb [====================] 100
webkit v1443 downloaded to /Users/limao/Library/Caches/ms-p
Downloading ffmpeg v1005 - 1.3 Mb [====================] 10
ffmpeg v1005 downloaded to /Users/limao/Library/Caches/ms-p
```

此处下载安装了：

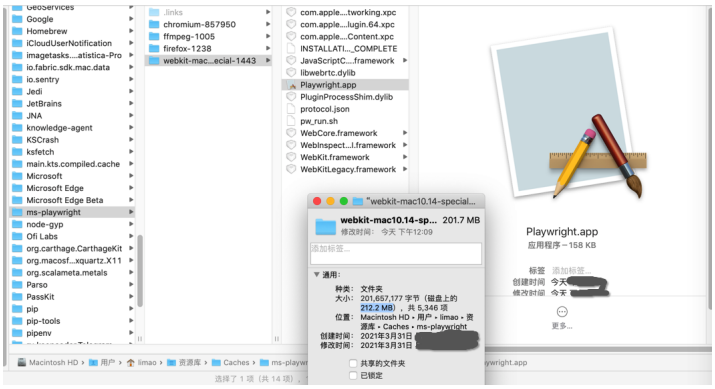
- chromium
  - 位置：`/Users/limao/Library/Caches/ms-playwright/chromium-857950`
  - 效果：



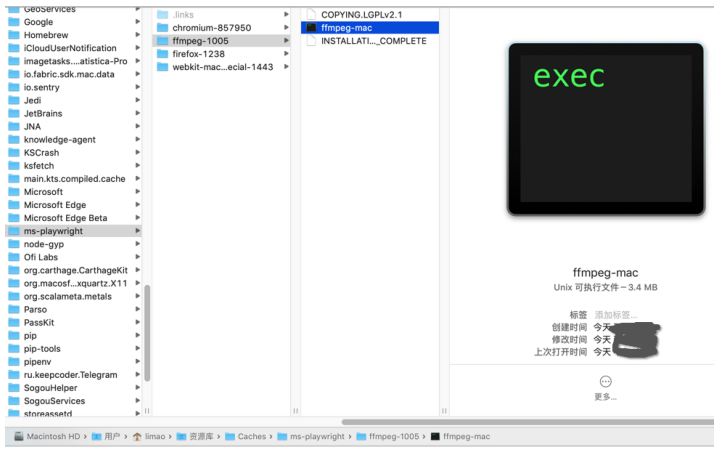
- firefox
  - 位置：`/Users/limao/Library/Caches/ms-playwright/firefox-1238`
  - 效果：



- webkit
  - 位置：`/Users/limao/Library/Caches/ms-playwright/webkit-mac10.14-special-1443`
  - 效果：



- ffmpeg
  - 位置：`/Users/limao/Library/Caches/ms-playwright/ffmpeg-1005`
  - 效果：



测试代码

```
# Function: Playwright demo baidu search
# Author: Crifan Li
# Update: 20210331

from playwright.sync_api import sync_playwright

with sync_playwright() as p:
    chromiumBrowserType = p.chromium
    print("chromiumBrowserType=%s" % chromiumBrowserType)
    browser = chromiumBrowserType.launch(headless=False)
    # chromiumBrowserType=<BrowserType name=chromium execut
    print("browser=%s" % browser)
    # browser=<Browser type=<BrowserType name=chromium exec
    page = browser.new_page()
    print("page=%s" % page)
    # page=<Page url='about:blank'>
    page.goto('http://www.baidu.com')
    print("page=%s" % page)

    # page=<Page url='https://www.baidu.com/'>
    page.screenshot(path=f'example_chromium.png')
    browser.close()
```

输出：

```
chromiumBrowserType=<BrowserType name=chromium executable_p
browser=<Browser type=<BrowserType name=chromium executable
page=<Page url='about:blank'>
page=<Page url='https://www.baidu.com/'>
```

效果：

附带

语法=帮助信息

```
□ playwright --help
Usage: npx playwright [options] [command]

Options:
  -V, --version                                output the versio
  -h, --help                                   display help for

Commands:
  open [options] [url]                         open page in brows
  codegen [options] [url]                      open page and gene
  debug <app> [args...]                        run command in deb
  install [browserType...]                     ensure browsers ne
  install-deps [browserType...]                install dependenc
  cr [options] [url]                           open page in Chro
  ff [options] [url]                           open page in Fire
  wk [options] [url]                           open page in WebK
  screenshot [options] <url> <filename>        capture a page sc
  pdf [options] <url> <filename>               save page as pdf
  help [command]                               display help for
```

基本操作

查找元素

从页面中 `寻找` = `定位` = `获取` 元素的函数是：

- element_handle
  - `element_handle.query_selector(selector)`
    - https://playwright.dev/python/docs/api/class-elementhandle#element_handlequery_selectorselector
  - `element_handle.query_selector_all(selector)`
    - https://playwright.dev/python/docs/api/class-elementhandle#element_handlequery_selector_allselector
- page
  - `page.query_selector(selector)`
    - https://playwright.dev/python/docs/api/class-page#pagequery_selectorselector
  - `page.query_selector_all(selector)`
    - https://playwright.dev/python/docs/api/class-page#pagequery_selector_allselector
      - 注：返回的是 `JSHandle` 的 `list`

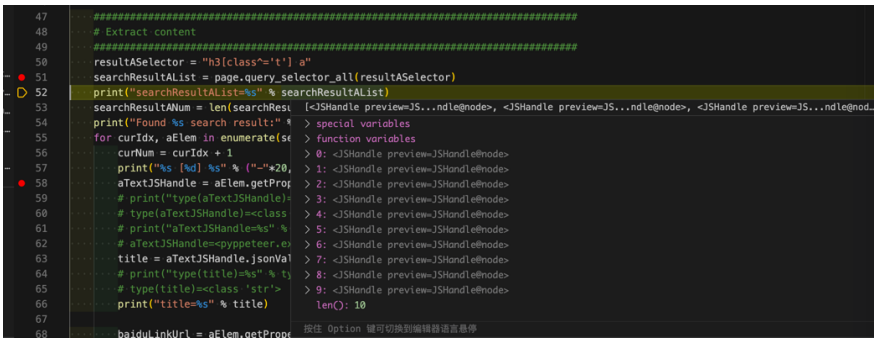
举例

百度搜索结果中的标题部分

```
    resultASelector = "h3[class^='t'] a"
    searchResultAList = page.query_selector_all(resultASele
    print("searchResultAList=%s" % searchResultAList)
```

输出：

```
searchResultAList=[<JSHandle preview=JSHandle@<a target="_
```



查找定位google搜索结果

背景：

对于playwright来说，html元素选择，即支持xpath，也支持css selector。

此处对于google搜索结果的定位：

html代码：

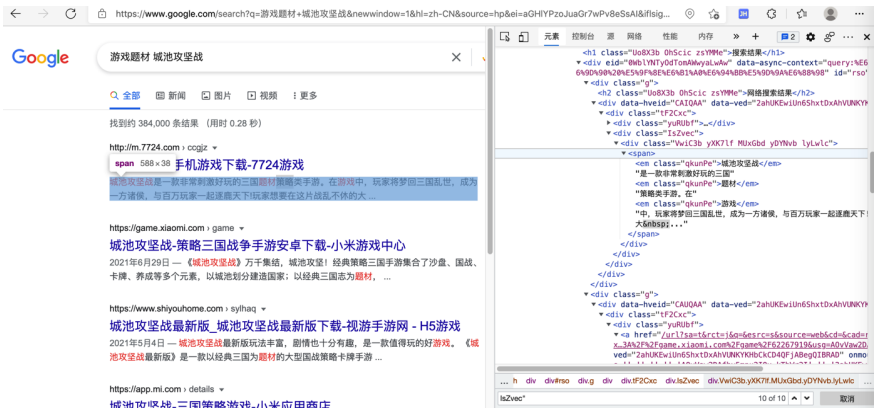
```
<div class="IsZvec">
    <div class="VwiC3b yXK7lf MUxGbd yDYNvb lyLwlc"><span>
                class="qkunPe">题材</em>策略类手游。在<em
                class="qkunPe">游戏</em>中，玩家将梦回三国乱世，
</div>
。。。
                <div class="IsZvec">
                    <div class="VwiC3b yXK7lf MUxGbd yDYNvb
                        </span><span>推荐理由：<em class="qk
                            class="qkunPe">游戏</em>以多
                </div>
。。。
```

`div class=IsZvec` 后面的 `div` 中：

- 第一个是只有一个 `span` ，是描述文字
- 之后每个都是2个 `span` ，前一个=第一个，是日期；后一个=最后一个，才是描述文字

定位第一个结果

第一个结果中，div下，只有一个span，是描述内容

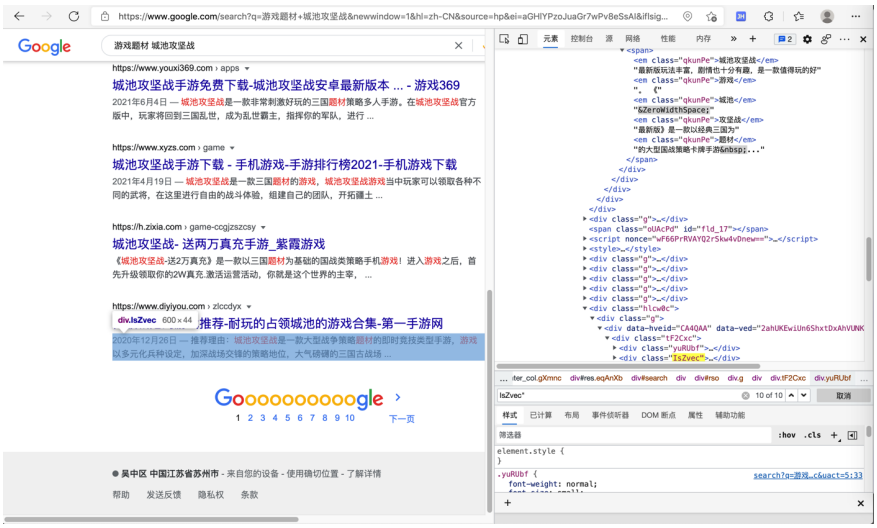


对应定位元素的 `css selector` 的写法：

```
div[class='g'] div[class='IsZvec'] div span:first-child
```

定位最后一个结果

页面：



想要定位到，最后一个的span，具体语法是：

```
div[class='g'] div[class='IsZvec'] div span:last-child
```

相关代码：

```
searchResultSelector = "div[class='g'] div[class='IsZvec']
searchResultList = page.query_selector_all(searchResultSele
```

相关完整代码：

```
    searchResultSelector = "div[class='g'] div[class='IsZve
    # searchResultSelector = "div[class='g'] div[class='IsZ
    searchResultList = page.query_selector_all(searchResult
    print("searchResultList=%s" % searchResultList)
    searchResultNum = len(searchResultList)
    print("Found %s search result:" % searchResultNum)
    for curIdx, spanElem in enumerate(searchResultList):
        curNum = curIdx + 1
        print("%s [%d] %s" % ("-"*20, curNum, "-"*20))
        print("spanElem=%s" % spanElem)
        title = spanElem.text_content()
        print("title=%s" % title)
        # title=城池攻坚战是一款非常刺激好玩的三国题材策略类手游。在
```

查找并点击元素

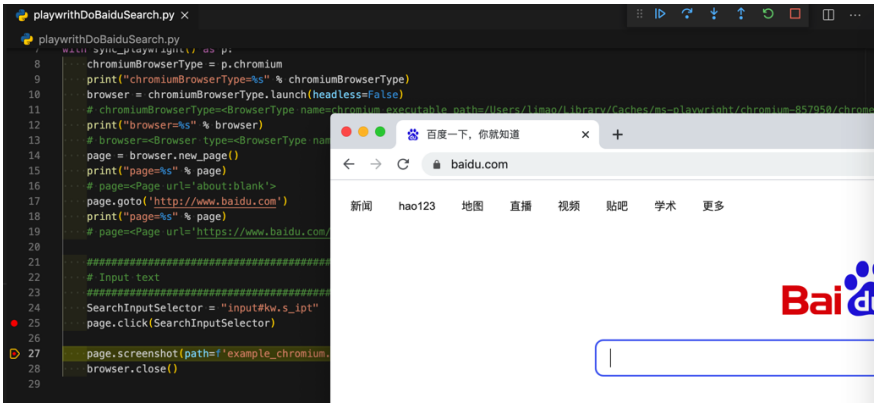
对于百度主页搜索输入框，html是：

```
<input id="kw" name="wd" class="s_ipt" value="" maxlength="
```

查找到该元素，并且点击该元素，的代码：

```
SearchInputSelector = "input#kw.s_ipt"
page.click(SearchInputSelector)
```

效果：点击了百度的输入框后的效果：

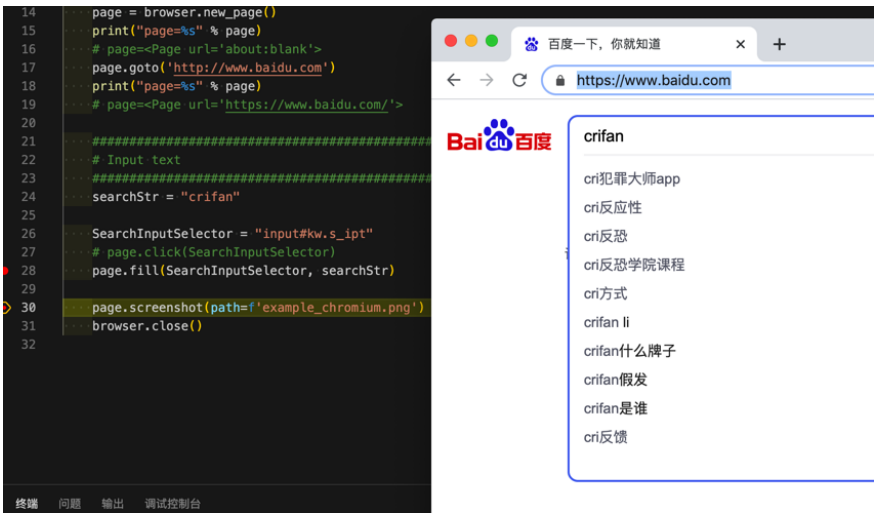


输入文字

给百度输入框中输入文字，的代码：

```
searchStr = "crifan"
SearchInputSelector = "input#kw.s_ipt"
page.fill(SearchInputSelector, searchStr)
```

效果：给百度搜索输入框中输入了文字



另：估计是先用 `Selector` 选择元素，再去用元素的 `fill` 也是可以的。

相关文档：Text input

其他几种 `fill`

另外还支持几种的 `fill` ：

- `page.fill(selector, value[, options])`
  - https://playwright.dev/docs/api/class-page#pagefillselector-value-options
- `frame.fill(selector, value[, options])`
  - https://playwright.dev/docs/api/class-frame#framefillselector-value-options
- `elementHandle.fill(value[, options])`
  - https://playwright.dev/docs/api/class-elementhandle#elementhandlefillvalue-options

等待元素出现

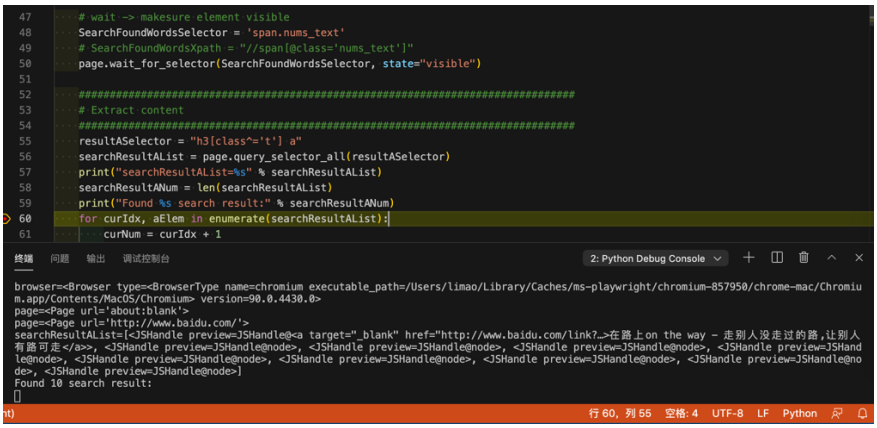
用 `page` 的 `wait_for_selector`

举例

等待百度搜索结果页显示

```
    SearchFoundWordsSelector = 'span.nums_text'
    page.wait_for_selector(SearchFoundWordsSelector, state=
```

效果：可以找到后续元素了

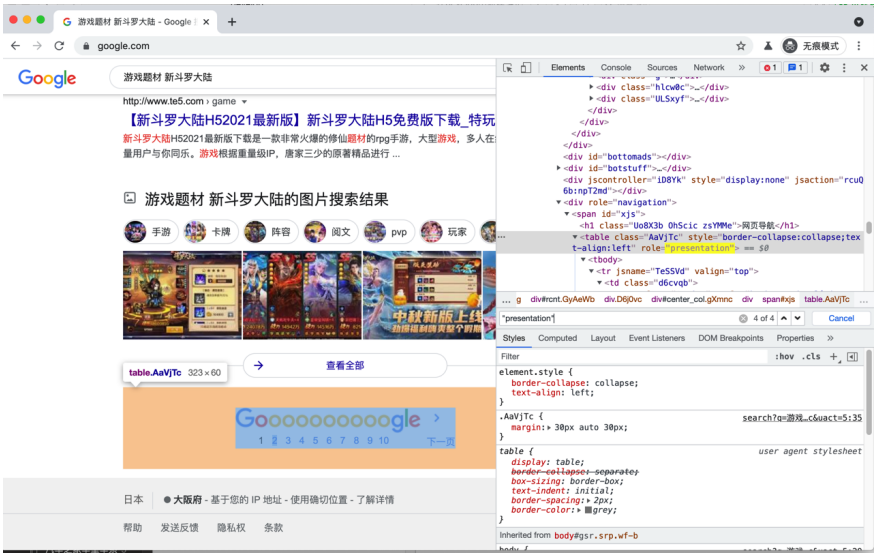


等待google搜索结果页显示

代码：

```
    # 底部 Gooooooooogle 多页面导航的部分，确保出现 -》 避免页面加
    # <table class="AaVjTc" style="border-collapse:collapse
    bottomNaviPageSelector = "table[role='presentation']"
    page.wait_for_selector(bottomNaviPageSelector, state="
```

页面：



即可实现希望的效果：

- 确保底部多页导航部分也的确显示了
- 确保了页面全部已加载完毕

-> 后续搜索结果就正常了：可以找到全部的结果，一般是8，9，10个。

官网资料

- 相关资料
  - page.wait_for_event(event, **kwargs)
  - page.wait_for_function(expression, **kwargs)
  - page.wait_for_load_state(**kwargs)
  - page.wait_for_selector(selector, **kwargs)
  - page.wait_for_timeout(timeout)

模拟按键

举例：进入百度主页，已输入了文字，想要触发搜索，有2种方式：

- 全局直接输入 `回车键`

```
EnterKey = "Enter"
# Method 1: press Enter key
page.keyboard.press(EnterKey)
```

- 定位到 `百度一下` 按钮，再按 `回车键`

```
EnterKey = "Enter"
# Method 2: locate element then click
SearchButtonSelector = "input#su"
page.press(SearchButtonSelector, EnterKey)
```

注：估计定位到按钮后，再click点击，也是可以的。有空再深究。

效果：触发了百度搜索后，显示出搜索结果

获取元素属性值

在找到元素后，获取属性值，可以用：

- `text_content()`：获取文本值
  - 文档：element_handle.text_content()
- `get_attribute("attribute_name")`：获取属性值
  - 文档：element_handle.get_attribute(name)
  - 举例：
    - `get_attribute("href")`
- `inner_html()`：获取html值
  - 文档：element_handle.inner_html()
- `inner_text()`：获取内部文本值
  - 文档：element_handle.inner_text()

举例

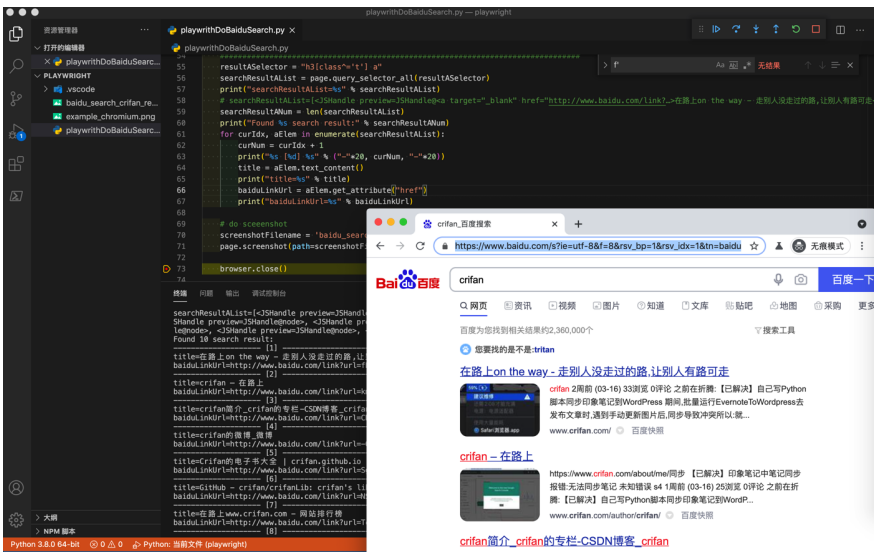
从百度搜索后的结果，解析提取每个结果的标题和链接的代码如下：

```
    ############################################################
    # Extract content
    ############################################################
    resultASelector = "h3[class^='t'] a"
    searchResultAList = page.query_selector_all(resultASele
    print("searchResultAList=%s" % searchResultAList)
    # searchResultAList=[<JSHandle preview=JSHandle@<a targ
    searchResultANum = len(searchResultAList)
    print("Found %s search result:" % searchResultANum)
    for curIdx, aElem in enumerate(searchResultAList):
        curNum = curIdx + 1
        print("%s [%d] %s" % ("-"*20, curNum, "-"*20))
        title = aElem.text_content()
        print("title=%s" % title)
        baiduLinkUrl = aElem.get_attribute("href")
        print("baiduLinkUrl=%s" % baiduLinkUrl)
```

输出结果：

```
searchResultAList=[<JSHandle preview=JSHandle@<a target="_
Found 10 search result:
---------------------- [1] ----------------------
title=在路上on the way - 走别人没走过的路,让别人有路可走
baiduLinkUrl=http://www.baidu.com/link?url=fB3F0xZmwig9r2M_
---------------------- [2] ----------------------
title=crifan - 在路上
baiduLinkUrl=http://www.baidu.com/link?url=kmvgD1PraoULnnjU
---------------------- [3] ----------------------
title=crifan简介_crifan的专栏-CSDN博客_crifan
baiduLinkUrl=http://www.baidu.com/link?url=CHLWAQKOMgb23Gmz
---------------------- [4] ----------------------
title=crifan的微博_微博
baiduLinkUrl=http://www.baidu.com/link?url=-QwlZ5SEmZD1R2Q
---------------------- [5] ----------------------
title=Crifan的电子书大全 | crifan.github.io
baiduLinkUrl=http://www.baidu.com/link?url=Sgrbyd_pBsm-BTAN
---------------------- [6] ----------------------
title=GitHub - crifan/crifanLib: crifan's library
baiduLinkUrl=http://www.baidu.com/link?url=NSZ5IzQ2Qag3CpGl
---------------------- [7] ----------------------
title=在路上www.crifan.com - 网站排行榜
baiduLinkUrl=http://www.baidu.com/link?url=Tc4cbETNKpQXj-kX
---------------------- [8] ----------------------
title=crifan的专栏_crifan_CSDN博客-crifan领域博主
baiduLinkUrl=http://www.baidu.com/link?url=OLkrWu8q9SRZuBN-
---------------------- [9] ----------------------
title=User crifan - Stack Overflow
baiduLinkUrl=http://www.baidu.com/link?url=t1rc0EGg33A-uJU
---------------------- [10] ----------------------
title=crifan - Bing 词典
baiduLinkUrl=http://www.baidu.com/link?url=8z-3hYeLAQ8T4ef
```

效果：



常见问题和心得

此处整理Playwright常见的一些问题。

初始化Sync的playwright的实例

Playwright的代码，初始化Playwright实例：

之前的常见的with写法是

```
from playwright.sync_api import sync_playwright

with sync_playwright() as p:
    for browser_type in [p.chromium, p.firefox, p.webkit]:
        browser = browser_type.launch()
        ...
```

此处希望初始化一个全局的playwright的实例，供后续使用。

所以就去掉with，改为普通赋值：

```
    p = sync_playwright()
```

报错：

错误原因：不是很懂。

解决办法：找到了规避的办法：

参考官网和别人例子，加上 `.start()` ：

```
p = sync_playwright().start()
```

即可。

using Playwright Sync API inside the asyncio loop

背景

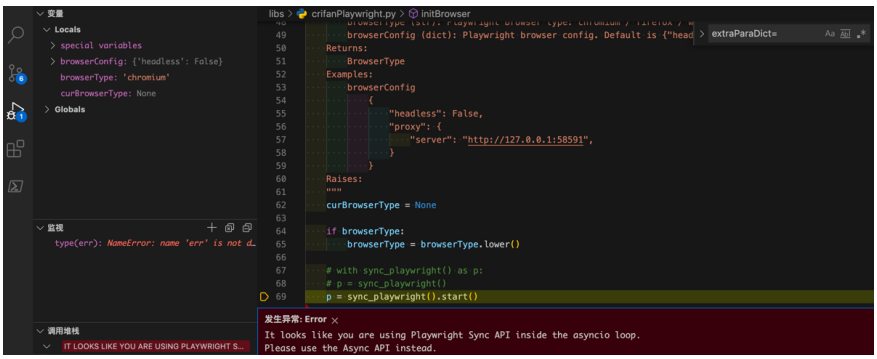
正常写了，初始化Playwright的sync的api的代码后：

```
def parseUrl(inputUrl, page=None):
    """Parse (redirected final long) url, title, html from

    Args:
        inputUrl (dict): input original (short link) url
        page (Page): Playwright page. Default is None. If N
    Returns:
        parse result(dict)
    Raises:
    """
    respValue = None

    if not page:
        page = initPage()

def initPage(pageConfig=None, browser=None):
    """Init playwright browser new page

    Args:
        pageConfig (dict): page config. Default is None.
        browser (BrowserType): Playwright browser. Default
    Returns:
        Page
    Examples:
        pageConfig
            {"pageLoadTimeout": 10}
    Raises:
    """
    if not browser:
        browser = initBrowser()
。。。

def initBrowser(browserType="chromium", browserConfig={"hea
    """
    For playwright, init to create a browser. For later use

    Args:
        browserType (str): Playwright browser type: chromiu
        browserConfig (dict): Playwright browser config. De
    Returns:
        BrowserType
    Examples:
        browserConfig
            {
                "headless": False,
                "proxy": {
                    "server": "http://127.0.0.1:58591",
                }
            }
    Raises:
```

```
    """
    curBrowserType = None

    if browserType:
        browserType = browserType.lower()

    # with sync_playwright() as p:
    # p = sync_playwright()
    p = sync_playwright().start()
```

别处去调用，结果报错

```
发生异常: Error
It looks like you are using Playwright Sync API inside the
```



以及：

后续类似代码：

```
def testParseSpeed(parseFunc, shortLinkList, testRoundNum,
    for curRoundIdx in range(testRoundNum):
        curRoundResult = parseShorLink_Common(parseFunc, sh
```

->

```
def parseShorLink_Common(parseFunc, shortLinkList, configDi
        browser = crifanPlaywright.initBrowser(browserConfi
```

->

```
def initBrowser(browserType="chromium", browserConfig={"hea
    p = sync_playwright().start()
```

又报同样错误。

错误原因：Playwright的内部实现机制，个人感觉，虽然看起来很高级，但是实际上用起来很是麻烦。导致，如果你是多次循环类的调用，或者是Python中的函数作为参数等情况，去初始化 `browser` 时，就会被内部判断为，视为，异步调用。

解决办法：只能把代码，改为，整套代码运行期间，**全局只能初始化一次** `browser` 。

否则如果有多次初始化调用 `sync_playwright().start()` 就会报错。

比如我此处最后的代码是：

```
    elif parseFunc == crifanPlaywright.parseUrl:
        # browser = extraParaDict["browser"]
        # browser = extracPara["browser"]
        # crifanPlaywright.closeBrowser(browser)
        pass
```

即：单轮测试之前，不去close这个Playwright的browser

```
def testParseSpeed(parseFunc, shortLinkList, testRoundNum,
    global gDnsFailedHost


    extracPara = {}
    # workaround for: It looks like you are using Playwrigh
    if parseFunc == crifanPlaywright.parseUrl:
        isCurHeadless = configDict["headless"]
        browser = crifanPlaywright.initBrowser(browserConf

        extracPara = {
            "browser": browser,
        }


。。。
    for curRoundIdx in range(testRoundNum):
。。。
        curRoundResult = parseShorLink_Common(parseFunc, sh
```

即：testParseSpeed中把browser的初始化，只运行一次

还要把全局的移动到最顶层：

```
    extracPara = {}
    # workaround for: It looks like you are using Playwrigh
    isCurHeadless = configDict["headless"]
    browser = crifanPlaywright.initBrowser(browserConfig={'

    for eachParseFunc in parseFuncList:
        if eachParseFunc == crifanPlaywright.parseUrl:
            extracPara = {
                "browser": browser,
            }
        else:
            extracPara = {}

        testParseSpeed(eachParseFunc, shortLinkList, testRo


。。。
def testParseSpeed(parseFunc, shortLinkList, testRoundNum,

        curRoundResult = parseShorLink_Common(parseFunc, sh
```

最终才实现了：代码全局只能和确保只初始化一次 `sync_playwright().start()` ，才能规避此报错。。。

感慨：用起来很是垃圾，不好用啊

所有页面加载都超时

背景

之前代码：

```
page.goto(eachShorLink)
```

报错：`Exception Timeout 10ms exceeded`

然后调试其余其他页面url，结果都报此错误。

错误原因：Playwright的页面超时参数的单位是**毫秒**。之前误以为秒，传入了：

```
curPageLoadTimeout = 10
page.set_default_navigation_timeout(curPageLoadTimeout)
page.set_default_timeout(curPageLoadTimeout)
```

导致：所有页面都报超时的错误

因为：都加载时间都超过10毫秒

解决办法：传入参数改为毫秒值即可

具体做法：

```
    curPageLoadTimeout = configDict["pageLoadTimeout"]
    curPageLoadTimeoutMilliSec = curPageLoadTimeout * 1000

        page.set_default_navigation_timeout(curPageLoadTime
        page.set_default_timeout(curPageLoadTimeoutMilliSec
```

详见：

官网文档：Page | Playwright Python

> page.set_default_timeout(timeout)
>
> - `timeout` `<float>` Maximum time in **milliseconds**

browser was not found

代码：

```
browser = p.chromium.launch(**browserConfig)
```

报错：

```
发生异常: Error
============================================================
"chromium" browser was not found.
Please complete Playwright installation via running
    "python -m playwright install"
```

错误原因：没有找到Playwright所安装的，此处用到的，底层的driver浏览器：chromium

背景：但是其实之前已在当前Mac中全局安装过

`playwright install`

安装到了：

`/Users/limao/Library/Caches/ms-playwright/chromium-888113`

对应chromium二进制路径是：

`/Users/limao/Library/Caches/ms-playwright/chromium-888113/chrome-mac/Chromium.app/Contents/MacOS/Chromium>`

但是此处竟然找不到。

根本原因：估计是此处是某Python项目的虚拟环境，导致没检测到，之前Python全局安装的chromium?

解决办法：只需，只能，重新安装一次即可

具体步骤：

```
python -m playwright install
```

附录：

正常启动输出：

```
curBrowserType=<BrowserType name=chromium executable_path=/
browser=<Browser type=<BrowserType name=chromium executable
```

给playwright加代理

此处给playwright（的chromium）加上代理的方式是：

- 给全局加代理

```
from playwright.sync_api import sync_playwright

googleHomeUrl = "https://www.google.com/"

PROXY_HTTP = "http://127.0.0.1:58591"
# PROXY_SOCKS5 = "socks5://127.0.0.1:51837"

browserLaunchOptionDict = {
    "headless": False,
    "proxy": {
        "server": PROXY_HTTP,
    }
}

with sync_playwright() as p:
    chromiumBrowserType = p.chromium

    browser = chromiumBrowserType.launch(**browserLaunchOpt

    page = browser.new_page()

    page.goto(googleHomeUrl)
```

即可实现：全局所有页面，自动用上代理 -》 可以正常打开google

举例

下面给出具体的playwright的实例案例供参考。

百度搜索自动化

此处给出用 `playwright` 模拟百度搜索，即百度搜索自动化的完整例子。

代码

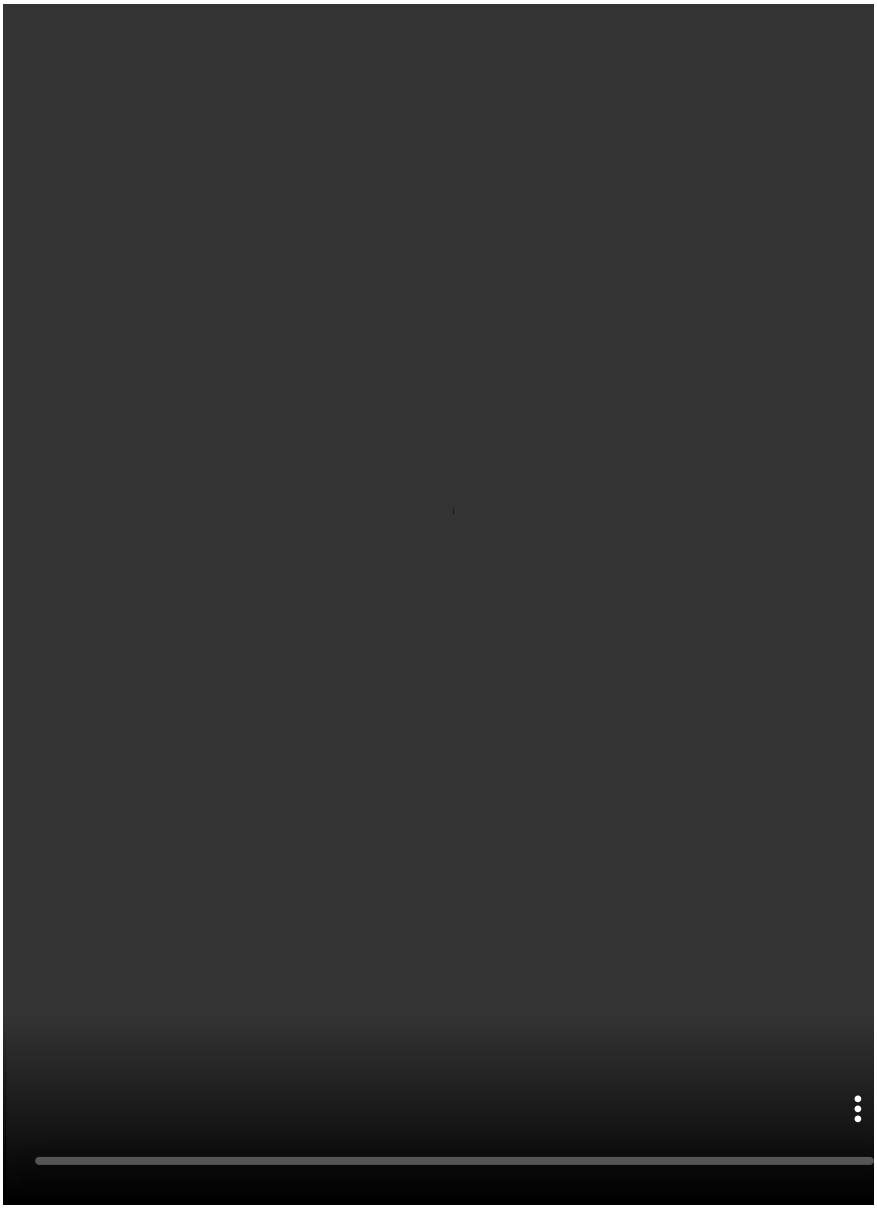
- 文件下载：playwrithDemoBaiduSearch.py
- 贴出代码

```
# Function: Playwright demo baidu search
# Author: Crifan Li
# Update: 20210331

from playwright.sync_api import sync_playwright

# here use sync mode
with sync_playwright() as p:
    chromiumBrowserType = p.chromium
    print("chromiumBrowserType=%s" % chromiumBrowserType)
    browser = chromiumBrowserType.launch(headless=False)
    # chromiumBrowserType=<BrowserType name=chromium execut
    print("browser=%s" % browser)
    # browser=<Browser type=<BrowserType name=chromium exec
    page = browser.new_page()
    print("page=%s" % page)
    # page=<Page url='about:blank'>

    ############################################################
    # Open url
    ############################################################
    page.goto('http://www.baidu.com')
    print("page=%s" % page)
    # page=<Page url='https://www.baidu.com/'>

    ############################################################
    # Input text
    ############################################################
    searchStr = "crifan"
    SearchInputSelector = "input#kw.s_ipt"

    # page.click(SearchInputSelector)
    page.fill(SearchInputSelector, searchStr)

    ############################################################
    # Trigger search
    ############################################################
    EnterKey = "Enter"

    # Method 1: press Enter key
    # page.keyboard.press(EnterKey)

    # Method 2: locate element then click
    SearchButtonSelector = "input#su"
    page.press(SearchButtonSelector, EnterKey)

    # wait -> makesure element visible
    SearchFoundWordsSelector = 'span.nums_text'
    # SearchFoundWordsXpath = "//span[@class='nums_text']"
    page.wait_for_selector(SearchFoundWordsSelector, state=
```

```
    ##########################################################
    # Extract content
    ##########################################################
    resultASelector = "h3[class^='t'] a"
    searchResultAList = page.query_selector_all(resultASele
    print("searchResultAList=%s" % searchResultAList)
    # searchResultAList=[<JSHandle preview=JSHandle@<a targ
    searchResultANum = len(searchResultAList)
    print("Found %s search result:" % searchResultANum)
    for curIdx, aElem in enumerate(searchResultAList):
        curNum = curIdx + 1
        print("%s [%d] %s" % ("-"*20, curNum, "-"*20))
        title = aElem.text_content()
        print("title=%s" % title)
        # title=在路上on the way - 走别人没走过的路,让别人有路可
        baiduLinkUrl = aElem.get_attribute("href")
        print("baiduLinkUrl=%s" % baiduLinkUrl)
        # baiduLinkUrl=http://www.baidu.com/link?url=fB3F0

    # do sceeenshot
    screenshotFilename = 'baidu_search_%s_result.png' % sea
    page.screenshot(path=screenshotFilename)

    browser.close()
```

效果

视频



图


playwrithDoBaiduSearch.py — playwright
playwrithDoBaiduSearch.py
baidu_search_crifan_result.png
crifan_百度搜索
https://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd
crifan
百度一下
在路上on the way - 走别人没走过的路,让别人有路可走
crifan – 在路上
Python 3.8.0 64-bit

输出

```
chromiumBrowserType=<BrowserType name=chromium executable_p
browser=<Browser type=<BrowserType name=chromium executable
page=<Page url='about:blank'>
page=<Page url='https://www.baidu.com/'>
searchResultAList=[<JSHandle preview=JSHandle@<a target="_b
Found 10 search result:
-------------------- [1] --------------------
title=在路上on the way - 走别人没走过的路,让别人有路可走
baiduLinkUrl=http://www.baidu.com/link?url=fB3F0xZmwig9r2M_
-------------------- [2] --------------------
title=crifan - 在路上
baiduLinkUrl=http://www.baidu.com/link?url=kmvgD1PraoULnnjU
-------------------- [3] --------------------
title=crifan简介_crifan的专栏-CSDN博客_crifan
baiduLinkUrl=http://www.baidu.com/link?url=CHLWAQKOMgb23Gmz
-------------------- [4] --------------------
title=crifan的微博_微博
baiduLinkUrl=http://www.baidu.com/link?url=-QwlZ5SEmZD1R2Q0
-------------------- [5] --------------------
title=Crifan的电子书大全 | crifan.github.io
baiduLinkUrl=http://www.baidu.com/link?url=Sgrbyd_pBsm-BTAN
-------------------- [6] --------------------
title=GitHub - crifan/crifanLib: crifan's library
baiduLinkUrl=http://www.baidu.com/link?url=NSZ5IzQ2Qag3CpGl
-------------------- [7] --------------------
title=在路上www.crifan.com - 网站排行榜
baiduLinkUrl=http://www.baidu.com/link?url=Tc4cbETNKpQXj-kX
-------------------- [8] --------------------
title=crifan的专栏_crifan_CSDN博客-crifan领域博主
baiduLinkUrl=http://www.baidu.com/link?url=OLkrWu8q9SRZuBN-
-------------------- [9] --------------------
title=User crifan - Stack Overflow
baiduLinkUrl=http://www.baidu.com/link?url=t1rc0EGg33A-uJU3
-------------------- [10] --------------------
title=crifan - Bing 词典
baiduLinkUrl=http://www.baidu.com/link?url=8z-3hYeLAQ8T4ef0
```

模拟谷歌搜索并获取结果

- 注：最新代码详见
  - https://github.com/crifan/crifanLibPython/blob/master/python3/crifanLib/thirdParty/crifanPlaywright.py

---

此处实现了，用Playwright模拟google谷歌搜索，并解析出第一页的搜索结果：

代码

浏览器和页面初始化

初始化浏览器和页面相关函数：

```
from playwright.sync_api import sync_playwright

def initBrowser(browserType="chromium", browserConfig={"hea
    """
    For playwright, init to create a browser. For later use

    Args:
        browserType (str): Playwright browser type: chromiu
        browserConfig (dict): Playwright browser config. De
    Returns:
        BrowserType
    Examples:
        browserConfig
            {
                "headless": False,
                "proxy": {
                    "server": "http://127.0.0.1:58591",
                }
            }
    Raises:
    """
    curBrowserType = None

    if browserType:
        browserType = browserType.lower()

    # with sync_playwright() as p:
    # p = sync_playwright()
    p = sync_playwright().start()
    # 多次调用，会：
    # 发生异常：Error
    # It looks like you are using Playwright Sync API insid
    # Please use the Async API instead.

    if browserType == "chromium":
        curBrowserType = p.chromium
    elif browserType == "firefox":
        curBrowserType = p.firefox
    elif browserType == "webkit":
        curBrowserType = p.webkit
    print("curBrowserType=%s" % curBrowserType)
    # curBrowserType=<BrowserType name=chromium executable_

    if not curBrowserType:
        print("Unsupported playwright browser type: %s" % b
        return None

    # browser = curBrowserType.launch(headless=False)
    # browser = curBrowserType.launch(**browserLaunchOptior
    browser = curBrowserType.launch(**browserConfig)
```

```
    print("browser=%s" % browser)
    # browser=<Browser type=<BrowserType name=chromium exec

    return browser

def initPage(pageConfig=None, browser=None):
    """Init playwright browser new page

    Args:
        pageConfig (dict): page config. Default is None.
        browser (BrowserType): Playwright browser. Default
    Returns:
        Page
    Examples:
        pageConfig
            {"pageLoadTimeout": 10}
    Raises:
    """
    if not browser:
        browser = initBrowser()

    page = browser.new_page()
    # print("page=%s" % page)

    if pageConfig:
        if "pageLoadTimeout" in pageConfig:
            curPageLoadTimeout = pageConfig["pageLoadTimeou
            curPageLoadTimeoutMilliSec = curPageLoadTimeout

            page.set_default_navigation_timeout(curPageLoad
            page.set_default_timeout(curPageLoadTimeoutMil

    return page

def closeBrowser(browser):
    """
    For playwright, close browser

    Args:
        browser (BrowserType): Playwright browser
    Returns:
    Raises:
    """
    browser.close()
```

获取google搜索结果

模拟google搜索，返回搜索结果：

```
def getGoogleSearchResult(searchKeyword, browser=None, isAu
    """
    Emulate google search, return search result

    Args:
        searchKeyword (str): str to search
        browser (BrowserType): Playwright browser. Default
        isAutoCloseBrowser (bool): whether auto close brows
    Returns:
        result dict list
    Raises:
    Examples:
        '游戏题材 城池攻坚战'
    """
    GoogleHomeUrl = "https://www.google.com/"

    searchResultDictList = []

    # if not browser:
    #     browser = initBrowser()

    # page = browser.new_page()
    # print("page=%s" % page)

    page = initPage(browser=browser)

    page.goto(GoogleHomeUrl)

    # <input class="gLFyf gsfi" jsaction="paste:puy29d;" ma
    SearchInputSelector = "input.gLFyf.gsfi"
    # page.click(SearchInputSelector)
    page.fill(SearchInputSelector, searchKeyword)

    EnterKey = "Enter"
    page.keyboard.press(EnterKey)

    # wait -> makesure element visible
    # <div id="result-stats">找到约 384,000 条结果<nobr> (用
    SearchFoundSelector = 'div#result-stats'
    page.wait_for_selector(SearchFoundSelector, state="vis


    """
        <table class="AaVjTc" style="border-collapse:collap
            <tbody>
                <tr jsname="TeSSVd" valign="top">
                    <td class="d6cvqb"><span class="SJajHc"
                            style="background:url(/images/
                    </td>
```

```
                    <td class="YyVfkd"><span class="SJajHc"
                            style="background:url(/images/n
                    </td>
        ° ° °
                    <td><a aria-label="Page 10" class="fl"
                            href="/search?q=%E6%B8%B8%E6%88
                                class="SJajHc NVbCr"
                                style="background:url(/imag
                    </td>
                    <td aria-level="3" class="d6cvqb" role=
                            href="/search?q=%E6%B8%B8%E6%88
                            id="pnnext" style="text-align:
                                style="background:url(/imag
                                style="display:block;margin
                </tr>
            </tbody>
        </table>

    """
    # 底部 Gooooooooogle 多页面导航的部分，确保出现 -》 避免页面加
    # <table class="AaVjTc" style="border-collapse:collapse
    bottomNaviPageSelector = "table[role='presentation']"
    page.wait_for_selector(bottomNaviPageSelector, state="

    searchResultDictList = parseGoogleSearchResult(page)
    # searchResultNum = len(searchResultDictList)
    # print("searchResultNum=%s" % searchResultNum)

    page.close()

    if isAutoCloseBrowser:
        # close browser
        closeBrowser(browser)

    return searchResultDictList
```

解析google搜索结果

期间调用了：从搜索结果页面解析提取搜索结果内容：

```
def parseGoogleSearchResult(page):
    """
    Parse google search result from current (search result)

    Args:
        page (Page): playwright browser Page
    Returns:
        result dict list
    Raises:
    """
    searchResultDictList = []

    """
    <div class="g">
        <h2 class="Uo8X3b OhScic zsYMMe">网络搜索结果</h2>
        <div data-hveid="CAIQAA" data-ved="2ahUKEwiUn6ShxtD
            <div class="tF2Cxc">
                <div class="yuRUbf"><a
                        href="/url?sa=t&amp;rct=j&amp;q=&ar
                        data-ved="2ahUKEwiUn6ShxtDxAhVUNKYK
                        onmousedown="return rwt(this,'','
                        target="_blank" rel="noopener" data
                        data-cthref="/url?sa=t&amp;rct=j&ar
                        <h3 class="LC20lb DKV0Md">城池攻坚战
                        。。。。。。
                    </div>
                </div>
                <div class="IsZvec">
                    <div class="VwiC3b yXK7lf MUxGbd yDYNvb
                                class="qkunPe">题材</em>策略
                                class="qkunPe">游戏</em>中,
                </div>
            </div>
        </div>
    </div>
    。。。

    <div class="g">
        <div data-hveid="CA4QAA" data-ved="2ahUKEwiUn6ShxtD
            <div class="tF2Cxc">
                <div class="yuRUbf"><a href="https://www.d
                        data-ved="2ahUKEwiUn6ShxtDxAhVUNKYK
                        onmousedown="return rwt(this,'','
                        target="_blank" rel="noopener"><br>
                        <h3 class="LC20lb DKV0Md">占领城池的
                        。。。。。。
                </div>
                <div class="IsZvec">
                    <div class="VwiC3b yXK7lf MUxGbd yDYNvb
```

```
                        </span><span>推荐理由: <em class="qk
                                class="qkunPe">游戏</em>以多
                </div>
            </div>
        </div>
    </div>

    ...

    <div class="g">
        <div data-hveid="CAgQAA" data-ved="2ahUKEwi5upCe5N
            <div class="tF2Cxc">
                <div class="yuRUbf"><a href="https://www.yo
                        data-ved="2ahUKEwi5upCe5NzxAhXLbt4K
                        ping="/url?sa=t&amp;source=web&amp
                        <h3 class="LC20lb DKV0Md">城池攻坚战
                        <div class="TbwUpd NJjxre"><cite c
                                class="dyjrff qzEoUe">
                    </a>
                    <div class="B6fmyf">
                        <div class="TbwUpd"><cite class="iU
                                class="dyjrff qzEoUe">
                        <div class="eFM0qc"><span>
                                <div jscontroller="hiU8Ie"
                                    aria-expanded="fals
                                    jsaction="PZcoEd;ke
                                    data-ved="2ahUKEwi5
                                        class="gTl8xb">
                                <ol class="action-menu-
                                    jsaction="keydown:K
                                    data-ved="2ahUKEwi5
                                    <li class="action-m
                                            href="https
                                            ping="/url?
                                    </li>
                                </ol>
                            </div>
                        </span></div>
                    </div>
                </div>
                <div class="IsZvec">
                    <div class="VwiC3b MUxGbd yDYNvb lyLwlc
                        </span>城池攻坚战是一款非常刺激好玩的三国
                </div>
            </div>
        </div>
    </div>

    """
    # searchResultSelector = "div[class='g'] div[class='IsZ
    # searchResultSelector = "div[class='g'] div[class='IsZ
```

```
    searchResultSelector = "div[class='g']"
    searchResultList = page.query_selector_all(searchResult
    # print("searchResultList=%s" % searchResultList)
    # searchResultList=[<JSHandle preview=JSHandle@node>, <
    searchResultNum = len(searchResultList)
    # print("Found %s search result" % searchResultNum) # 9

    # for debug
    # if searchResultNum < 8:
    if searchResultNum < 5:
        print("Unexpcted too little result count: %s" % sea

    for curIdx, curResultElem in enumerate(searchResultList
        curNum = curIdx + 1
        # print("%s [%d] %s" % ("-"*20, curNum, "-"*20))
        # print("curResultElem=%s" % curResultElem)

        # urlTitleElemSelector = "div[class='tF2Cxc']"
        urlTitleSelector = "div[class='tF2Cxc'] div[class=
        urlTitleElem = curResultElem.query_selector(urlTit
        # print("urlTitleElem=%s" % urlTitleElem) # urlTit

        urlSelector = "a[href^='http']"
        urlElem = urlTitleElem.query_selector(urlSelector)
        # print("urlElem=%s" % urlElem) # urlElem=JSHandle@
        urlStr = urlElem.get_attribute("href")
        # print("urlStr=%s" % urlStr) # urlStr=https://www.

        titleSelector = "h3[class='LC20lb DKV0Md']"
        titleElem = urlTitleElem.query_selector(titleSelect
        # print("titleElem=%s" % titleElem) # titleElem=JSH
        titleStr = titleElem.text_content()
        # print("titleStr=%s" % titleStr) # titleStr=城池攻

        dateDescSelector = "div[class='IsZvec'] div[class^=
        dateDescElem = curResultElem.query_selector(dateDes

        # spanElemList = dateDescElem.query_selector_all("s
        dateStr = ""
        originDateStr = ""
        dateSelector = "span[class^='MUxGbd']"
        dateElem = dateDescElem.query_selector(dateSelector
        if dateElem:
            originDateStr = dateElem.text_content()
            # print("originDateStr=%s" % originDateStr) #
            dateStr = originDateStr.replace(" — ", "")
            # print("dateStr=%s" % dateStr) # '2021年4月19日

        dateDescStr = dateDescElem.text_content() # '下载后计
        dateDescStr = dateDescStr.strip()
        # print("dateDescStr=%s" % dateDescStr)
```

```
        descStr = dateDescStr.replace(originDateStr, "")
        # print("descStr=%s" % descStr)

        curSearchResultDict = {
            "url": urlStr,
            "title": titleStr,
            "date": dateStr,
            "description": descStr,
        }
        # print("curSearchResultDict=%s" % curSearchResultD
        # curSearchResultDict={'url': 'https://www.325sy.co
        searchResultDictList.append(curSearchResultDict)

    return searchResultDictList
```

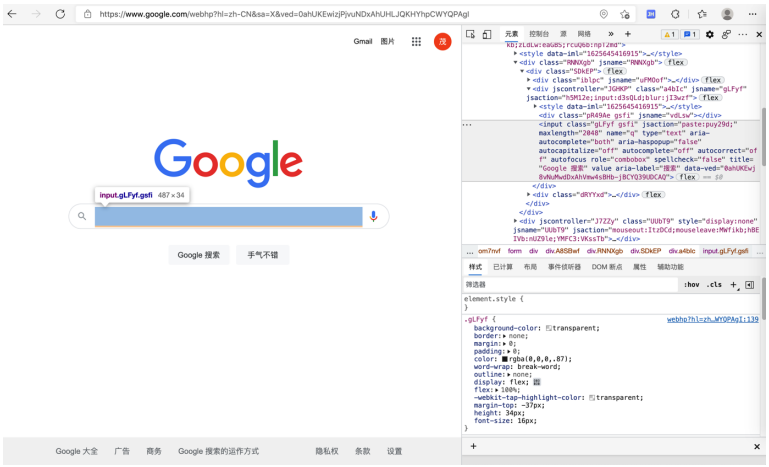
代码调用

```
# test code
PROXY_HTTP = "http://127.0.0.1:58591"
PROXY_SOCKS5 = "socks5://127.0.0.1:51837"
browserConfig = {
    "headless": False,
    # "headless": True,
    "proxy": {
        "server": PROXY_HTTP,
    }
}
browser = initBrowser(browserConfig=browserConfig)

searchStr = '游戏题材 新斗罗大陆'

resultDictList = getGoogleSearchResult(searchStr, browser=b
resultNum = len(resultDictList)
print("Google search %s found %s result" % (searchStr, resu

searchStr = '游戏题材 城池攻坚战'
resultDictList = getGoogleSearchResult(searchStr, browser=b
resultNum = len(resultDictList)
print("Google search %s found %s result" % (searchStr, resu

closeBrowser(browser)
```

页面和html和调试

模拟google搜索

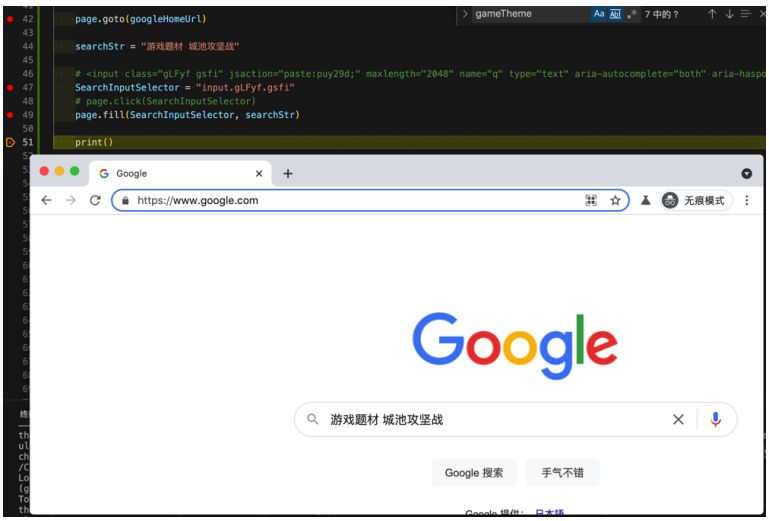
- 搜索输入框



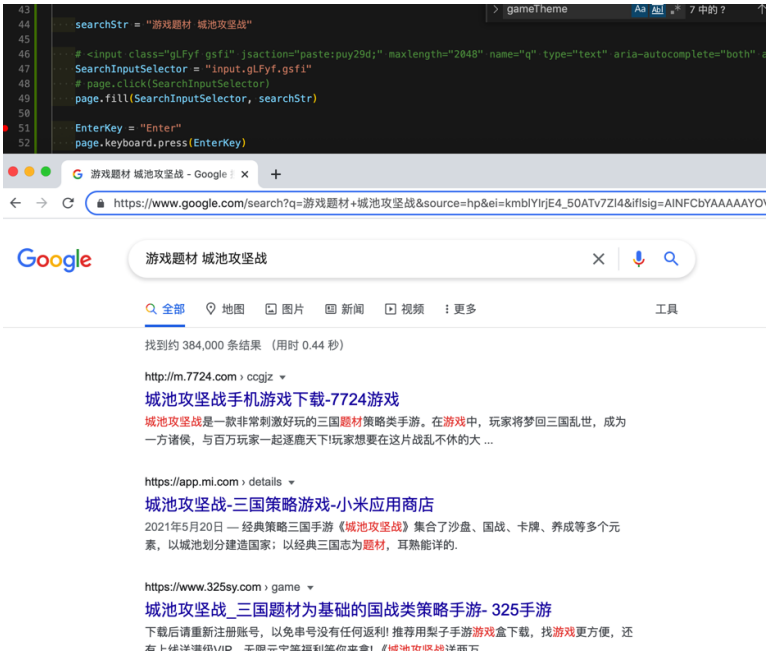
- html

```
<input class="gLFyf gsfi" jsaction="paste:puy29d;"
```

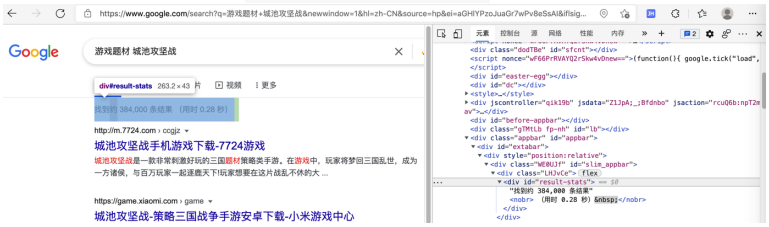
- 已输入要搜索的文字



- 触发了搜索



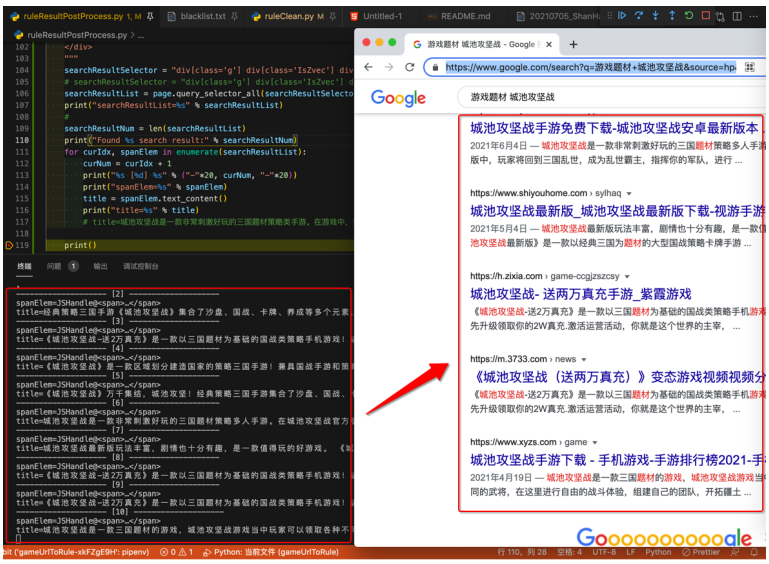
- 等待页面加载完毕：出现 找到约 xxx 条结果



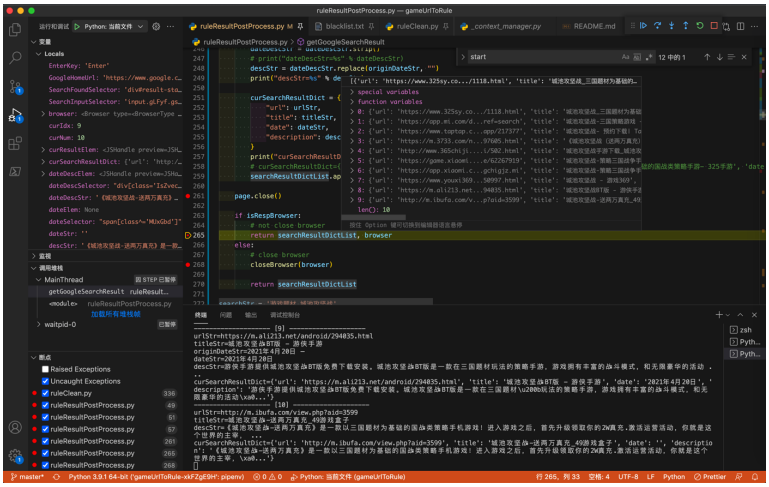
- html

```
<div id="result-stats">找到约 384,000 条结果<nobr>
```

- 已解析可输出每个结果的标题

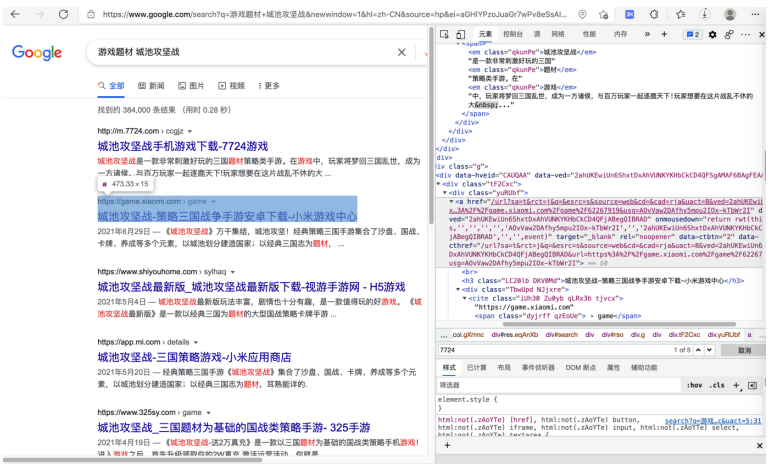


- 返回10个搜索结果



google搜索结果页

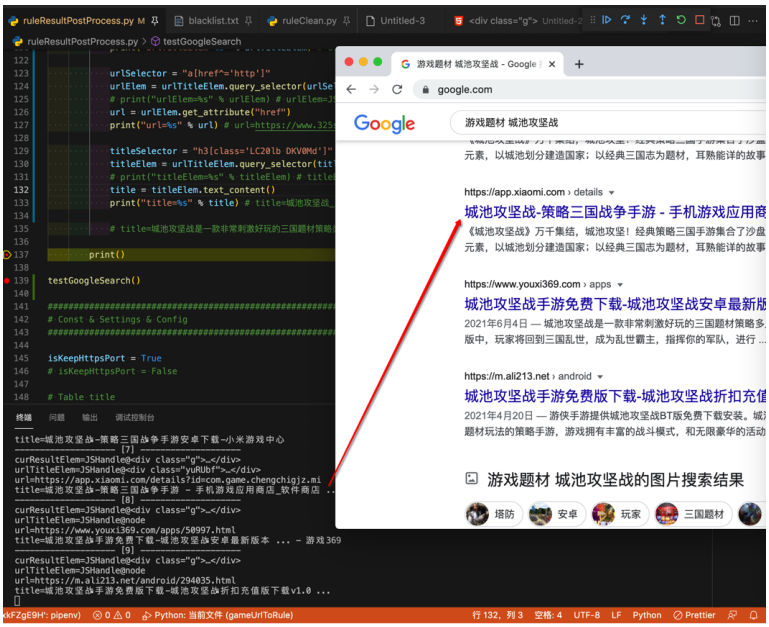
- 搜索结果页

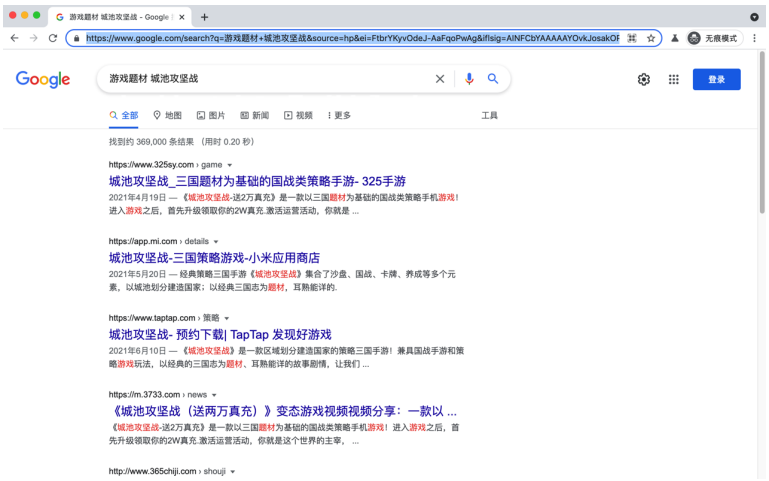


  - html

```
<div class="g">
    <div data-hveid="CAUQAA" data-ved="2ahUKE
        <div class="tF2Cxc">
            <div class="yuRUbf"><a
                    href="/url?sa=t&amp;rct=j
                    data-ved="2ahUKEwiUn6Shxt
                    onmousedown="return rwt(t
                    target="_blank" rel="noop
                    data-cthref="/url?sa=t&am

                    <h3 class="LC20lb DKV0Md"
                    <div class="TbwUpd NJjxre
                            class="dyjrff
                </a>
                <div class="B6fmyf">
                    <div class="TbwUpd"><cite
                            class="dyjrff
                    <div class="eFM0qc"><span
                            <div jscontroller
                                aria-expa
                                jsaction=
                                data-ved=
                                    class
                            <ol class="ac
                                jsaction=
                                data-ved=
                                <li class
                                    h
                                    o
                                    t
                            </ol>
                        </div>
                    </span></div>
                </div>
            </div>
            <div class="IsZvec">
                <div class="VwiC3b yXK7lf MUx
                    </span><span>《<em
                            class="qkunPe">城
                            class="qkunPe">题
            </div>
        </div>
    </div>
</div>
```


游戏题材 城池攻坚战 - Google
https://www.google.com/search?q=游戏题材+城池攻坚战&source=hp&ei=js3rYJPbAsPm-QbtxrIA&iflsig=AINFCbYAAAAAYOvbnrwn2ae63c
Google
游戏题材 城池攻坚战
登录
《城池攻坚战》万千集结，城池攻坚！经典策略三国手游集合了沙盘、国战、卡牌、养成等多个元素，以城池划分建造国家；以经典三国志为题材，耳熟能详的故事 ...
https://www.youxi369.com › apps
城池攻坚战手游免费下载-城池攻坚战安卓最新版本 ... - 游戏369
2021年6月4日 — 城池攻坚战是一款非常刺激好玩的三国题材策略多人手游。在城池攻坚战官方版中，玩家将回到三国乱世，成为乱世霸主，指挥你的军队，进行 ...
https://m.ali213.net › android
城池攻坚战手游免费版下载-城池攻坚战折扣充值版下载v1.0 ...
2021年4月20日 — 游侠手游提供城池攻坚战BT版免费下载安装。城池攻坚战BT版是一款在三国题材玩法的策略手游，游戏拥有丰富的战斗模式，和无限豪华的活动 ...
游戏题材 城池攻坚战的图片搜索结果
塔防
安卓
玩家
三国题材
下载
武将
查看全部
提供反馈


ruleResultPostProcess.py
blacklist.txt
ruleClean.py
Untitled-3
testGoogleSearch
print()
testGoogleSearch()
isKeepHttpsPort = True
# isKeepHttpsPort = False
# Table title
终端
问题
输出
调试控制台
游戏题材 城池攻坚战 - Google
google.com
游戏题材 城池攻坚战
https://app.xiaomi.com › details
城池攻坚战-策略三国战争手游 - 手机游戏应用商
https://www.youxi369.com › apps
城池攻坚战手游免费下载-城池攻坚战安卓最新版
https://m.ali213.net › android
城池攻坚战手游免费版下载-城池攻坚战折扣充值
游戏题材 城池攻坚战的图片搜索结果
塔防
安卓
玩家
三国题材
行 132，列 3
空格: 4
UTF-8
LF
Python
Prettier


游戏题材 城池攻坚战 - Google
https://www.google.com/search?q=游戏题材+城池攻坚战&source=hp&ei=FtbrYKyvOdeJ-AaFqoPwAg&iflsig=AINFCbYAAAAAYOvkJosakOF
Google
游戏题材 城池攻坚战
登录
全部
地图
图片
新闻
视频
更多
工具
找到约 369,000 条结果（用时 0.20 秒）
https://www.325sy.com › game
城池攻坚战_三国题材为基础的国战类策略手游- 325手游
2021年4月19日 — 《城池攻坚战-送2万真充》是一款以三国题材为基础的国战类策略手机游戏！进入游戏之后，首先升级领取你的2W真充.激活运营活动，你就是 ...
https://app.mi.com › details
城池攻坚战-三国策略游戏-小米应用商店
2021年5月20日 — 经典策略三国手游《城池攻坚战》集合了沙盘、国战、卡牌、养成等多个元素，以城池划分建造国家；以经典三国志为题材，耳熟能详的.
https://www.taptap.com › 策略
城池攻坚战- 预约下载| TapTap 发现好游戏
2021年6月10日 — 《城池攻坚战》是一款区域划分建造国家的策略三国手游！兼具国战手游和策略游戏玩法，以经典的三国志为题材、耳熟能详的故事剧情，让我们 ...
https://m.3733.com › news
《城池攻坚战（送两万真充）》变态游戏视频视频分享：一款以 ...
《城池攻坚战-送2万真充》是一款以三国题材为基础的国战类策略手机游戏！进入游戏之后，首先升级领取你的2W真充.激活运营活动，你就是这个世界的主宰，...
http://www.365chiji.com › shouji

- 部分的html

```
<div class="g">
    <h2 class="Uo8X3b OhScic zsYMMe">网络搜索结果</h2>
    <div data-hveid="CAIQAA" data-ved="2ahUKEwiUn6ShxtDxAh
        <div class="tF2Cxc">
            <div class="yuRUbf"><a
                    href="/url?sa=t&amp;rct=j&amp;q=&amp;es
                    data-ved="2ahUKEwiUn6ShxtDxAhVUNKYKHbC
                    onmousedown="return rwt(this,'','','
                    target="_blank" rel="noopener" data-ct
                    data-cthref="/url?sa=t&amp;rct=j&amp;q=

                    <h3 class="LC20lb DKV0Md">城池攻坚战手机游
                    。。。。。。
                </div>
            </div>
            <div class="IsZvec">
                <div class="VwiC3b yXK7lf MUxGbd yDYNvb lyL
                            class="qkunPe">题材</em>策略类手游
                            class="qkunPe">游戏</em>中，玩家
            </div>
        </div>
    </div>
</div>


<div class="g">
    <div data-hveid="CA4QAA" data-ved="2ahUKEwiUn6ShxtDxAh
        <div class="tF2Cxc">
            <div class="yuRUbf"><a href="https://www.diyiyo
                    data-ved="2ahUKEwiUn6ShxtDxAhVUNKYKHbC
                    onmousedown="return rwt(this,'','','
                    target="_blank" rel="noopener">

                    <h3 class="LC20lb DKV0Md">占领城池的游戏推
                    。。。。。。
            </div>
            <div class="IsZvec">
                <div class="VwiC3b yXK7lf MUxGbd yDYNvb lyL
                    </span><span>推荐理由：<em class="qkunPe
                            class="qkunPe">游戏</em>以多元化
            </div>
        </div>
    </div>
</div>


<div class="g">
    <div data-hveid="CAgQAA" data-ved="2ahUKEwi5upCe5NzxAh
        <div class="tF2Cxc">
            <div class="yuRUbf"><a href="https://www.youxi
```

```
                    data-ved="2ahUKEwi5upCe5NzxAhXLbt4KHXS
                    ping="/url?sa=t&amp;source=web&amp;rct=

                    <h3 class="LC20lb DKV0Md">城池攻坚战手游免
                    <div class="TbwUpd NJjxre"><cite class=
                                class="dyjrff qzEoUe"> › ap
                </a>
                <div class="B6fmyf">
                    <div class="TbwUpd"><cite class="iUh30
                                class="dyjrff qzEoUe"> › ap
                    <div class="eFM0qc"><span>
                            <div jscontroller="hiU8Ie" clas
                                    aria-expanded="false" a
                                    jsaction="PZcoEd;keydow
                                    data-ved="2ahUKEwi5upCe
                                        class="gTl8xb"></sp
                                <ol class="action-menu-pane
                                    jsaction="keydown:Xiq7v
                                    data-ved="2ahUKEwi5upCe
                                    <li class="action-menu-
                                            href="https://v
                                            ping="/url?sa=t
                                    </li>
                                </ol>
                            </div>
                        </span></div>
                </div>
            </div>
            <div class="IsZvec">
                <div class="VwiC3b MUxGbd yDYNvb lyLwlc"><s
                    </span>城池攻坚战是一款非常刺激好玩的三国题材
            </div>
        </div>
    </div>
</div>
```

输出结果

举例1：

```
Found 9 search result:
-------------------- [1] --------------------
curResultElem=JSHandle@node
urlTitleElem=JSHandle@<div class="yuRUbf">…</div>
url=https://www.325sy.com/game/1118.html
title=城池攻坚战_三国题材为基础的国战类策略手游- 325手游
-------------------- [2] --------------------
curResultElem=JSHandle@<div class="g">…</div>
urlTitleElem=JSHandle@node
url=https://app.mi.com/details?id=com.game.chengchigjz.mi&
title=城池攻坚战-三国策略游戏-小米应用商店
-------------------- [3] --------------------
curResultElem=JSHandle@<div class="g">…</div>
urlTitleElem=JSHandle@<div class="yuRUbf">…</div>
url=https://www.taptap.com/app/217377
title=城池攻坚战- 预约下载 | TapTap 发现好游戏
-------------------- [4] --------------------
curResultElem=JSHandle@<div class="g">…</div>
urlTitleElem=JSHandle@node
url=https://m.3733.com/news/197605.html
title=《城池攻坚战（送两万真充）》变态游戏视频视频分享：一款以三国 ...
-------------------- [5] --------------------
curResultElem=JSHandle@<div class="g">…</div>
urlTitleElem=JSHandle@node
url=http://www.365chiji.com/shouji/502.html
title=城池攻坚战手游下载_城池攻坚战游戏安卓版下载_三六五吃鸡
-------------------- [6] --------------------
curResultElem=JSHandle@<div class="g">…</div>
urlTitleElem=JSHandle@node
url=https://game.xiaomi.com/game/62267919
title=城池攻坚战-策略三国战争手游安卓下载-小米游戏中心
-------------------- [7] --------------------
curResultElem=JSHandle@<div class="g">…</div>
urlTitleElem=JSHandle@<div class="yuRUbf">…</div>
url=https://app.xiaomi.com/details?id=com.game.chengchigjz&
title=城池攻坚战-策略三国战争手游 - 手机游戏应用商店_软件商店 ...
-------------------- [8] --------------------
curResultElem=JSHandle@<div class="g">…</div>
urlTitleElem=JSHandle@node
url=https://www.youxi369.com/apps/50997.html
title=城池攻坚战手游免费下载-城池攻坚战安卓最新版本 ... - 游戏369
-------------------- [9] --------------------
curResultElem=JSHandle@<div class="g">…</div>
urlTitleElem=JSHandle@node
url=https://m.ali213.net/android/294035.html
title=城池攻坚战手游免费版下载-城池攻坚战折扣充值版下载v1.0 ...
```

举例2：

```
Found 8 search result:
-------------------- [1] --------------------
curSearchResultDict={'url': 'https://shouji.newyx.net/top/
-------------------- [2] --------------------
curSearchResultDict={'url': 'https://zhuanlan.zhihu.com/p/3
。。。
-------------------- [7] --------------------
curSearchResultDict={'url': 'https://www.taptap.com/app/135
-------------------- [8] --------------------
curSearchResultDict={'url': 'http://www.te5.com/game/450202
searchResultNum=8
Google search 游戏题材 新斗罗大陆 found 8 result
page=<Page url='about:blank'>
Found 9 search result:
-------------------- [1] --------------------
curSearchResultDict={'url': 'https://www.325sy.com/game/111
。。。
-------------------- [9] --------------------
curSearchResultDict={'url': 'https://www.3839.com/a/134478.
searchResultNum=9
Google search 游戏题材 城池攻坚战 found 9 result
```

其他小例子

此处给出一些其他的Playwright的应用案例：

用Playwright实现短链解析长链

代码：

```
def parseUrl(inputUrl, page=None):
    """Parse (redirected final long) url, title, html from

    Args:
        inputUrl (dict): input original (short link) url
        page (Page): Playwright page. Default is None. If N
    Returns:
        parse result(dict)
    Raises:
    """
    respValue = None

    if not page:
        page = initPage()

    try:
        page.goto(inputUrl)

        parsedLongLink = page.url # https://api.interactive
        logging.debug("parsedLongLink=%s", parsedLongLink)
        longLinkTitle = page.title() # '现金大派送'
        logging.debug("longLinkTitle=%s", longLinkTitle)
        longLinkHtml = page.content()
        logging.debug("longLinkHtml=%s", longLinkHtml)

        respValue = {
            "isParseOk": True,
            "url": parsedLongLink,
            "title": longLinkTitle,
            "html": longLinkHtml,
        }
    except Exception as err:
        errStr = str(err)
        # 'net::ERR_NAME_NOT_RESOLVED at http://dmh2.cn/9ja
        # 'net::ERR_CONNECTION_CLOSED at http://zhongan.cor
        # 'Timeout 10000ms exceeded.\n=====================
        #
        logging.debug("Playwright goto %s exception: %s",

        respValue = {
            "isParseOk": False,
            "errMsg": errStr,
        }

    return respValue
```

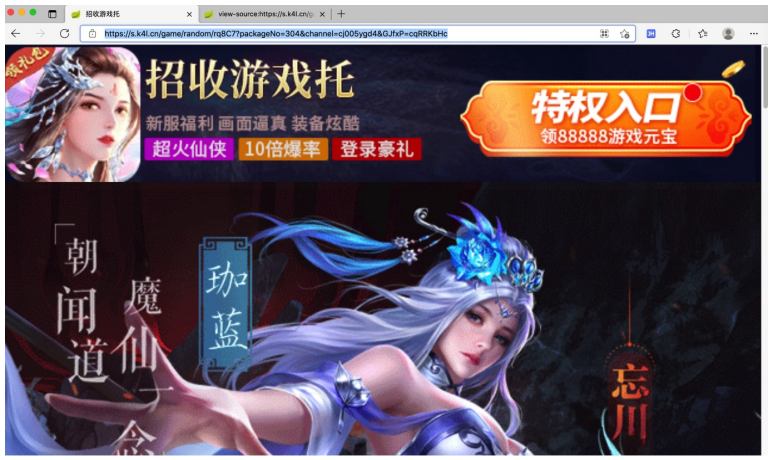
- 注：最新代码详见

    - https://github.com/crifan/crifanLibPython/blob/master/python3/crifanLib/thirdParty/crifanPlaywright.py

举例：

- 输入：
    - https://urldx.cn/9MUPowKt
- 输出


```
{
  "isParseOk": true,
  "url": "https://s.k4l.cn/game/random/rq8C7?packageNo=
  "title": "招收游戏托",
  "html": """\n<!DOCTYPE html>\n<html lang="zh-CN">\n<h
  }
```


- 长链页面
    - 

附录

下面列出相关参考资料。

教程和资料

- Node.JS版
  - GitHub
    - https://github.com/microsoft/playwright
  - 官网
    - Fast and reliable end-to-end testing for modern web apps | Playwright
      - https://playwright.dev/
    - 英文
      - Doc
        - Getting Started | Playwright
          - https://playwright.dev/docs/intro
        - Installation configuration
          - https://playwright.dev/docs/installation
      - API
        - Playwright | Playwright
          - https://playwright.dev/docs/api/class-playwright
        - Page
          - https://playwright.dev/docs/api/class-page/
        - Core concepts | Playwright
          - https://playwright.dev/docs/core-concepts
        - ElementHandle
          - https://playwright.dev/docs/api/class-elementhandle
    - 中文
      - Getting Started | Playwright 中文文档 | Playwright 中文网 (bootcss.com)
        - https://playwright.bootcss.com/docs/intro
      - Element selectors | Playwright 中文文档 | Playwright 中文网
        - https://playwright.bootcss.com/docs/selectors
- Python版
  - Getting Started | Playwright
    - https://playwright.dev/python/docs/intro/
  - 官网
    - Fast and reliable end-to-end testing for modern web apps | Playwright
      - https://playwright.dev/python/
  - API
    - Playwright | Playwright

        - https://playwright.dev/python/docs/api/class-playwright
    - ElementHandle | Playwright
        - https://playwright.dev/python/docs/api/class-elementhandle
    - Page | Playwright
        - https://playwright.dev/python/docs/api/class-page

参考资料

- 【已解决】Python的Playwright用page.query_selector_all找不到元素
- 【已解决】用Python的Playwright定位并点击百度搜索输入框
- 【已解决】Mac中安装Python版Playwright和初始化开发环境
- 【已解决】用Python的Playwright给百度搜索输入框中输入文字
- 【已解决】用Python的Playwright触发百度首页的搜索
- 【已解决】Python的Playwright去解析提取百度搜索的结果
- 【已解决】Mac中Playwright启动报错：chromium browser was not found
- 【已解决】Mac中playwright的launch代码报错：chromium browser was not found
- 【规避解决】Playwright初始化报错：It looks like you are using Playwright Sync API inside the asyncio loop
- 【记录】测试Playwright短链解析长链的速度
- 【已解决】Playwright的page的goto所有页面都超时Timeout exceeded
- 【已解决】把用Playwright模拟google搜索并返回结果抽象提取出函数
- 【已解决】Playwright模拟google搜索但返回第一页结果不完整
- 【已解决】Playwright代码报错：AttributeError PlaywrightContextManager object has no attribute chromium
- 【已解决】优化Playwright模拟google搜索返回结果：解析更多字段标题链接日期描述
- 【已解决】Mac中用playwright模拟google搜索并解析出搜索结果
- 【已解决】Chrome调试google搜索结果页面中描述文字的html和xpath和css定位语法
- 【已解决】如何用css定位写出html中div中div中的最后一个span元素
- 【已解决】playwright中如何给chromium设置代理以便能访问外网如google
-
- 微软开源 Python 自动化神器 Playwright - 知乎
- webautomation - Using Playwright for Python, how do I select (or find) an element? - Stack Overflow
- element_handle.inner_html()
- element_handle.inner_text()
- element_handle.text_content()
- element_handle.get_attribute(name)
-