

目录

前言	1.1
概述	1.2
举例	1.2.1
百度搜索自动化	1.2.1.1
调试页面元素	1.2.1.1.1
PC端	1.3
Web端	1.3.1
Selenium	1.3.1.1
puppeteer	1.3.1.2
初始化环境	1.3.1.2.1
基本操作	1.3.1.2.2
查找定位元素	1.3.1.2.2.1
输入文字	1.3.1.2.2.2
模拟按键	1.3.1.2.2.3
等待元素出现	1.3.1.2.2.4
获取元素属性	1.3.1.2.2.5
举例	1.3.1.2.3
百度搜索自动化	1.3.1.2.3.1
资料	1.3.1.2.4
Playwright	1.3.1.3
初始化环境	1.3.1.3.1
基本操作	1.3.1.3.2
查找元素	1.3.1.3.2.1
查找并点击元素	1.3.1.3.2.2
输入文字	1.3.1.3.2.3
等待元素出现	1.3.1.3.2.4
模拟按键	1.3.1.3.2.5
移动端	1.4
Android	1.4.1
uiautomator2	1.4.1.1
iOS	1.4.2
facebook-wda	1.4.2.1

附录	1.5
参考资料	1.5.1

解放你的双手：自动化测试

- 最新版本: v2.2
- 更新时间: 20210510

简介

介绍如何通过自动化测试，去解放你的双手，提高测试效率。包括进行概述，以及详细介绍PC端和移动端。包括PC端的Web端的Selenium、puppeteer、Playwright和移动端的常见框架，比如Android的uiautomator2、iOS的facebook-wda等自动化工具。

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/free_hand_test_automation](#): 解放你的双手：自动化测试

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

- [解放你的双手：自动化测试 book.crifan.com](#)
- [解放你的双手：自动化测试 crifan.github.io](#)

离线下载阅读

- [解放你的双手：自动化测试 PDF](#)
- [解放你的双手：自动化测试 ePUB](#)
- [解放你的双手：自动化测试 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](#)

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新：2021-05-12 09:07:22

概述

- 自动化测试
 - = test automation
 - = automation testing
 - 根据目标平台和设备不同可分为
 - **PC端**
 - **Web领域=Web端=浏览器操作自动化**
 - 常见框架
 - Selenium
 - [Selenium知识总结](#)
 - puppeteer
 - Playwright
 - **移动端**
 - 概览
 - [移动端自动化测试概览](#)
 - 常见框架
 - **Android**
 - uiautomator2
 - [安卓自动化测试利器: uiautomator2](#)
 - **iOS**
 - facebook-wda
 - [iOS自动化测试利器: facebook-wda](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook 最后更新: 2021-04-29 14:06:40

举例

下面通过实例例子来说明，自动化操作，对于不同平台大概是什么样的，以便于有个宏观的，具体的了解。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2021-04-29 14:06:40

百度搜索自动化

下面就通过例子：百度搜索自动化，来说明不同平台的自动化具体是什么样子：

PC端

selenium

代码：

- 文件：[seleniumDemoBaiduSearch.py](#)
- 贴出来是

```
# Function: demo selenium do baidu search and extract result
# Author: Crifan Li
# Update: 20210327

from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

from bs4 import BeautifulSoup
import re

chromeDriver = webdriver.Chrome()

#####
# Open url
#####
baiduUrl = "https://www.baidu.com"
chromeDriver.get(baiduUrl)
print("title=%s" % chromeDriver.title)

assert chromeDriver.title == "百度一下，你就知道"
# assert '百度' in chromeDriver.title

#####
# Find/Locate search button
#####
SearchInputId = "kw"
searchInputElem = chromeDriver.find_element_by_id(SearchInputId)
print("searchInputElem=%s" % searchInputElem)
# searchInputElem=<selenium.webdriver.remote.webelement.WebElement: search input>

#####
# Input text
#####
searchInputElem.clear()
print("Clear existed content")

searchStr = "crifan"
searchInputElem.send_keys(searchStr)
print("Entered %s to search box" % searchStr)

#####
# Click button
#####

# Method 1: emulate press Enter key
# searchButtonElem.send_keys(Keys.RETURN)
# print("Pressed Enter/Return key")
```

```

# Method 2: find button and click
BaiduSearchId = "su"
baiduSearchButtonElem = chromeDriver.find_element_by_id(BaiduSearchId)
print("baiduSearchButtonElem=%s" % baiduSearchButtonElem)
baiduSearchButtonElem.click()
print("Clicked button %s" % baiduSearchButtonElem)

#####
# Wait page change/loading completed
#   -> following element makesure show = visible
#   -> otherwise possibly can NOT find elements
#####
MaxWaitSeconds = 10
numTextElem = WebDriverWait(chromeDriver, MaxWaitSeconds).until(EC.presence_of_element_located((By.XPATH, "//span[@class='c-container']")))
print("Search complete, showing: %s" % numTextElem)

#####
# Extract result
#####

# Method 1: use Selenium to extract title list
searchResultAList = chromeDriver.find_elements_by_xpath("//span[@class='c-container']")
print("searchResultAList=%s" % searchResultAList)
searchResultANum = len(searchResultAList)
print("searchResultANum=%s" % searchResultANum)
for curIdx, curSearchResultAElem in enumerate(searchResultAList):
    curNum = curIdx + 1
    print("%s [%d] %s" % ("-"*20, curNum, "-"*20))
    baiduLinkUrl = curSearchResultAElem.get_attribute("href")
    print("baiduLinkUrl=%s" % baiduLinkUrl)
    title = curSearchResultAElem.text
    print("title=%s" % title)

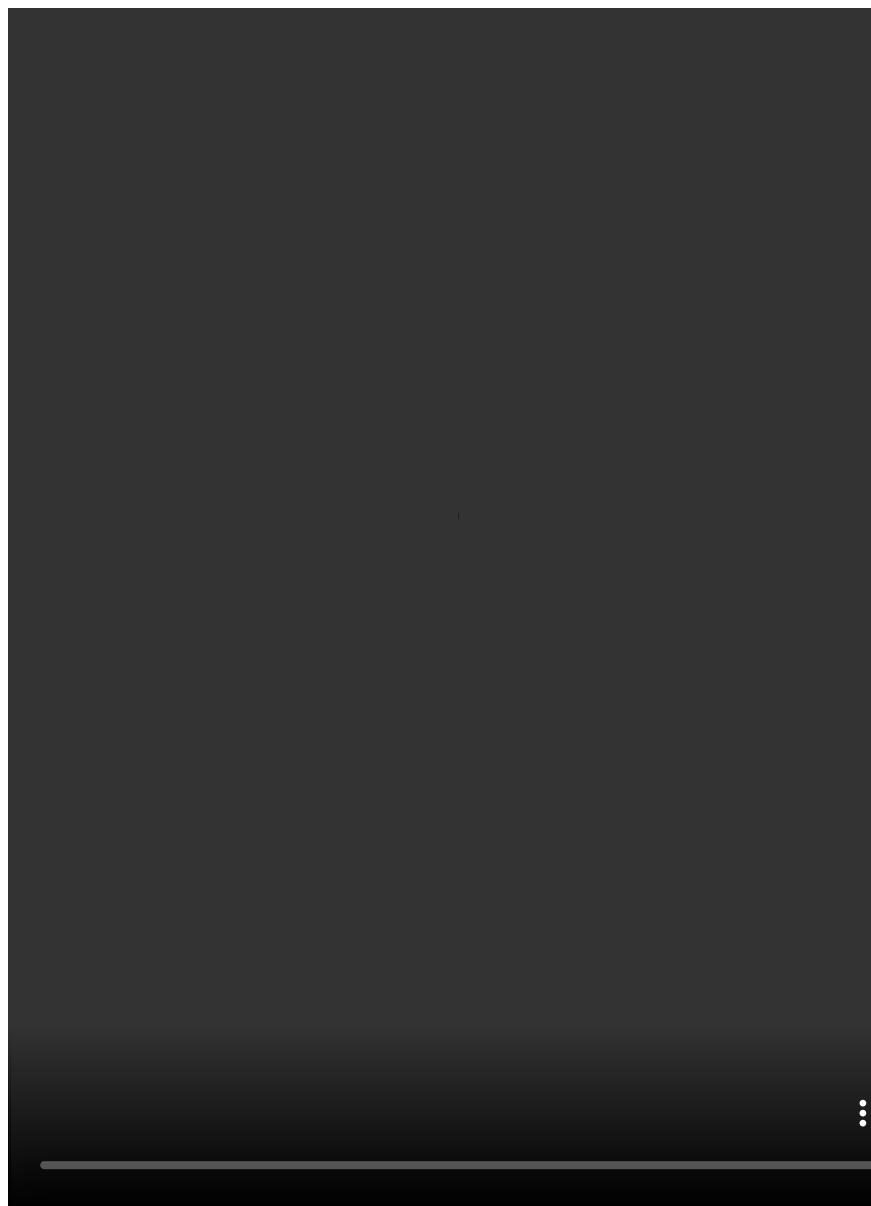
# # Method 2: use BeautifulSoup to extract title list
# curHtml = chromeDriver.page_source
# curSoup = BeautifulSoup(curHtml, 'html.parser')
# beginTP = re.compile("^t.*")
# searchResultH3List = curSoup.find_all("h3", {"class": "c-container"})
# print("searchResultH3List=%s" % searchResultH3List)
# searchResultH3Num = len(searchResultH3List)
# print("searchResultH3Num=%s" % searchResultH3Num)
# for curIdx, searchResultH3Item in enumerate(searchResultH3List):
#     curNum = curIdx + 1
#     print("%s [%d] %s" % ("-"*20, curNum, "-"*20))
#     aElem = searchResultH3Item.find("a")
#     # print("aElem=%s" % aElem)
#     baiduLinkUrl = aElem.attrs["href"]
#     print("baiduLinkUrl=%s" % baiduLinkUrl)

```

```
#     title = aElem.text
#     print("title=%s" % title)

#####
# End close
#####
chromeDriver.close()
```

效果：



puppeteer

代码：

- 文件：[puppeteerDemoBaiduSearch.py](#)
- 贴出来是

```
# Function: pypeteer (python version puppeteer) do bai
# Author: Crifan Li
# Update: 20210330

import asyncio
from pypeteer import launch

async def main():
    browser = await launch(headless=False)
    page = await browser.newPage()

    await page.setJavaScriptEnabled(enabled=True)

    baiduUrl = "https://www.baidu.com"
    await page.goto(baiduUrl)
    # await page.screenshot({'path': 'baidu.png'})

    #####
    # Input text
    #####
    searchStr = "crifan"

    # SearchInputSelector = "input[id=kw]"
    SearchInputSelector = "input[id='kw']"

    # SearchInputXpath = "//input[@id='kw']"
    # searchInputElem = page.xpath(SearchInputXpath)

    # # Input method 1: selector + click + keyboard type
    # searchInputElem = await page.querySelector(SearchInputSelector)
    # print("searchInputElem=%s" % searchInputElem)
    # await searchInputElem.click()
    # await page.keyboard.type(searchStr)

    # Input method 2: focus then type
    # await page.focus(SearchInputSelector)
    # await page.keyboard.type(searchStr)

    # Input method 3: selector and input once using type
    await page.type(SearchInputSelector, searchStr, delay=0)

    #####
    # Trigger search
    #####
    # Method 1: press ENTER key
    await page.keyboard.press('Enter')

    # # Method 2: locator search button then click
    # SearchButtonSelector = "input[id='su']"
    # searchButtonElem = await page.querySelector(SearchButtonSelector)
```

```

# searchButtonElem = await page.querySelector('#search')
# print("searchButtonElem=%s" % searchButtonElem)
# await searchButtonElem.click()
# # await searchButtonElem.press("Enter")

#####
# Wait page reload complete
#####
SearchFoundWordsSelector = 'span.nums_text'
SearchFoundWordsXpath = "//span[@class='nums_text']"

# await page.waitForSelector(SearchFoundWordsSelect
# await page.waitFor(SearchFoundWordsSelector)
# await page.waitForXPath(SearchFoundWordsXpath)
# Note: all above exception: 发生异常: ElementHandle
# so change to following

# # Method 1: just wait
# await page.waitFor(2000) # millisecond

# Method 2: wait element showing
SingleWaitSeconds = 1
while not await page.querySelector(SearchFoundWords
    print("Still not found %s, wait %s seconds" % (Se
    await asyncio.sleep(SingleWaitSeconds)
    # pass

#####
# Extract result
#####

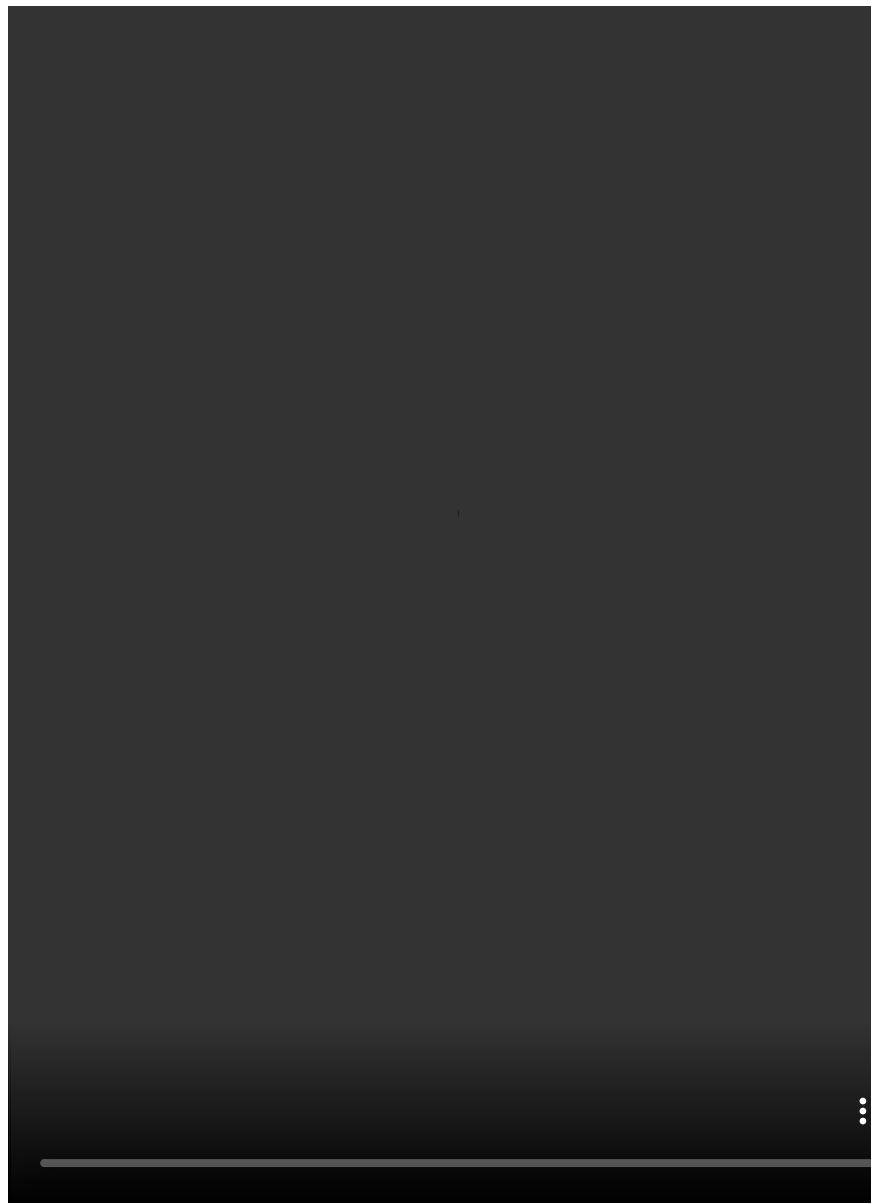
resultASelector = "h3[class^='t'] a"
searchResultAList = await page.querySelectorAll(res
# print("searchResultAList=%s" % searchResultAList)
searchResultANum = len(searchResultAList)
print("Found %s search result:" % searchResultANum)
for curIdx, aElem in enumerate(searchResultAList):
    curNum = curIdx + 1
    print("%s [%d] %s" % ("—" * 20, curNum, "—" * 20))
    aTextJSHandle = await aElem.getProperty('textCont
# print("type(aTextJSHandle)=%s" % type(aTextJSHa
# type(aTextJSHandle)=<class 'puppeteer.executio
# print("aTextJSHandle=%s" % aTextJSHandle)
# aTextJSHandle=<puppeteer.execution_context.JSHa
    title = await aTextJSHandle.jsonValue()
# print("type(title)=%s" % type(title))
# type(title)=<class 'str'>
    print("title=%s" % title)

    baiduLinkUrl = await (await aElem.getProperty("hr
    print("baiduLinkUrl=%s" % baiduLinkUrl)

```

```
await browser.close()  
  
asyncio.get_event_loop().run_until_complete(main())
```

效果：



playwright

代码：

- 文件：[playwrithDemoBaiduSearch.py](#)
- 贴出来是

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

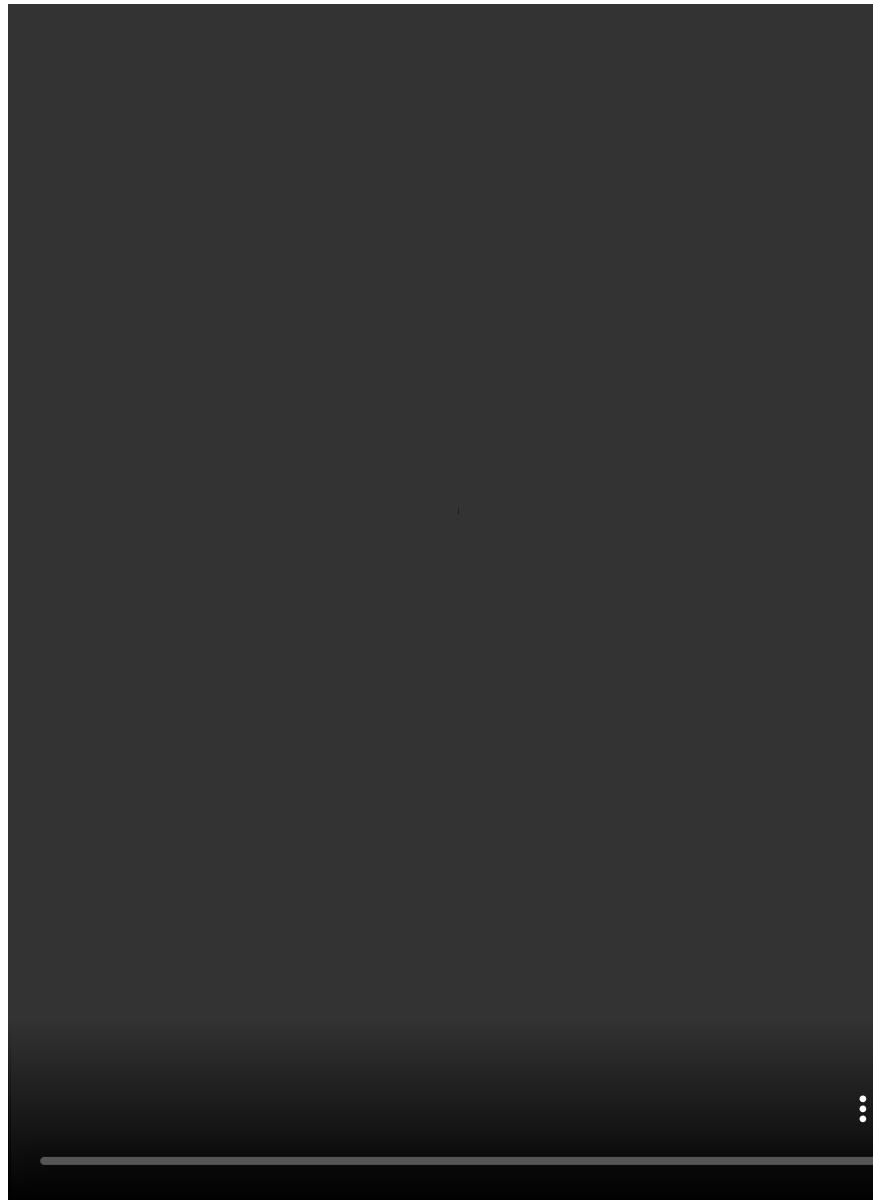
    # wait -> makesure element visible
    SearchFoundWordsSelector = 'span.nums_text'
    # SearchFoundWordsXpath = "//span[@class='nums_text'
    page.wait_for_selector(SearchFoundWordsSelector, st
```

```
#####
# Extract content
#####
resultASelector = "h3[class^='t'] a"
searchResultAList = page.query_selector_all(resultASelector)
print("searchResultAList=%s" % searchResultAList)
# searchResultAList=[<JSHandle preview=JSHandle@<a href="http://www.baidu.com/link?url=fE...
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
    baiduLinkUrl = aElem.get_attribute("href")
    print("baiduLinkUrl=%s" % baiduLinkUrl)
    # baiduLinkUrl=http://www.baidu.com/link?url=fE...

    # do sceenshot
    screenshotFilename = 'baidu_search_%s_result.png' % curNum
    page.screenshot(path=screenshotFilename)

browser.close()
```

效果：



移动端

Android端

uiautomator2

代码：

- 文件：[uiautomator2DemoBaiduSearch.py](#)
- 贴出来是

```
# Function: uiautomator2 demo baidu search
# Author: CriFan Li
# Update: 2020417

# import time
import uiautomator2 as u2

d = u2.connect() # connect to device
print("d.info=%s" % d.info)
# d.info={'currentPackageName': 'com.android.browser', 'device': 'Android', 'display': '1080x1920', 'language': 'zh', 'model': 'Xiaomi Mi 9T Pro', 'platform': 'Android', 'product': 'Mi 9T Pro', 'version': '9.0', 'width': 1080, 'height': 1920}

# for debug: get current app info
# curApp = d.app_current()
# print("curApp=%s" % curApp)

# for debug: get running app list
# activeAppList = d.app_list_running()
# print("activeAppList=%s" % activeAppList)

#####
# Launch browser
#####
Browser_XiaomiBuiltIn = "com.android.browser"
browserPackage = Browser_XiaomiBuiltIn
# d.app_start(browserPackage)
d.app_start(browserPackage, stop=True)

# wait util browser launch complete -> appear 我的 tab
# MustShowTabName = "主页"
MustShowTabName = "我的"
# d(text=MustShowTabName).exists(timeout=10)
d(text=MustShowTabName, packageName=browserPackage).exists()
print("Browser homepage loaded")

#####
# Open baidu homepage
#####
SearchInputId = "com.android.browser:id/b4w"

# # open new window
# windowUiObj = d(resourceId="com.android.browser:id/dm")
# windowUiObj.click()

# # click add to new window
# addNewWindowUiObj = d(resourceId="com.android.browser:id/dm")
# addNewWindowUiObj.click()

# for debug
# curPageXml = d.dump_hierarchy(compressed=False, pretty=False)
```

```
# print("curPageXml=%s" % curPageXml)

# find input box inside address bar

# # Method 1: use driver pass in parameter
# inputUiObj = d(resourceId=SearchInputId, className="android.widget.EditText")
# # inputUiObj = d(resourceId=SearchInputId)
# print("type(inputUiObj)=%s" % type(inputUiObj)) # type()
# print("inputUiObj=%s" % inputUiObj) # inputUiObj=<ui.WebDriverWrapper
# inputUiObjectInfo = inputUiObj.info
# print("type(inputUiObjectInfo)=%s" % type(inputUiObjectInfo))
# print("inputUiObjectInfo=%s" % inputUiObjectInfo) # inputUiObjectInfo={<
# isFoundInput = inputUiObj.exists # True

# # Method 2: use xpath
# inputXPathSelector = d.xpath("//android.widget.TextView[@resource-id='SearchInputId']")
# # inputXPathSelector = d.xpath("//*[@resource-id='SearchInputId']")
# print("type(inputXPathSelector)=%s" % type(inputXPathSelector))
# inputXPathElem = inputXPathSelector.get()
# print("type(inputXPathElem)=%s" % type(inputXPathElem))
# print("inputXPathElem=%s" % inputXPathElem) # inputXPathElem=<ui.WebDriverWrapper
# print("type(inputXPathElem.attrib)=%s" % type(inputXPathElem.attrib))
# print("inputXPathElem.attrib=%s" % inputXPathElem.attrib)
# print("inputXPathElem.getAttribute('resource-id')=%s" % inputXPathElem.getAttribute('resource-id'))
# isFoundInput = inputXPathSelector.exists # True

# trigger into input page

# Method 1
inputUiObj = d(resourceId=SearchInputId, className="android.widget.EditText")
inputUiObj.click()
print("Clicked search box")

# # Method 2
# inputXPathSelector = d.xpath("//android.widget.TextView[@resource-id='SearchInputId']")
# inputXPathSelector.click()

# input baidu homr url
BaiduHomeUrl = "https://www.baidu.com/"
AddressInputId = "com.android.browser:id/bqi"
searchUiObj = d(resourceId=AddressInputId, className="android.widget.EditText")
searchUiObj.set_text(BaiduHomeUrl)
print("Inputed baidu homepage url: %s" % BaiduHomeUrl)

# trigger jump to baidu home
EnterKey = "enter"
d.press(EnterKey)
print("Emulated press key %s" % EnterKey)

# wait util baidu home loaded
# d(text="百度一下", resourceId="com.android.browser:id/bq3")
```

```
d(text="百度一下,你就知道", className="android.view.View").exists()
print("Baidu home loaded")

#####
# Input text
#####
searchStr = "crifan"

baiduSearchKeywordUiObj = d(resourceId="index-kw", className="android.widget.EditText")
baiduSearchKeywordUiObj.set_text(searchStr)
print("Inputed baidu search text %s" % searchStr)

#####
# Trigger baidu search
#####

# # Method 1: press key
# TriggerSearchKey = "enter" # work
# # TriggerSearchKey = "search" # not work
# # TriggerSearchKey = "go" # not work
# # TriggerSearchKey = "done" # not work
# d.press(TriggerSearchKey)
# print("Emulated press key %s" % TriggerSearchKey)

# Method 2: find 百度一下 button then click
baiduSearchButtonUiObj = d(resourceId="index-bn", className="android.widget.Button")
baiduSearchButtonUiObj.click()
print("Clicked baidu search button")

#####
# Extract search result content
#####

# Special: for fixbug of get page xml is not latest, so use
d.service("uiautomator").stop()
d.service("uiautomator").start()
# time.sleep(1)

# for debug
# get page source xml
# curPageXml = d.dump_hierarchy(compressed=False, pretty=False)
# print("curPageXml=%s" % curPageXml)
# with open("baidu_search_%s_result_pageSource_reloaded.xml" % searchStr, "w") as fp:
#     fp.write(curPageXml)

d(resourceId="results").exists(timeout=10)

# Note: following syntax can NOT find elements
# resultsSelector = d.xpath("//*[@resource-id='results']")
# titleButtonSelectorList = resultsSelector.xpath("//android.widget.TextView")
# titleButtonSelectorList = resultsSelector.xpath("./android.widget.TextView")
```

```
# Xpath chain search can find elements
titleButtonElementList = d.xpath("//*[@resource-id='result_main']/div/div[1]/div/div[2]/div[1]/div[1]/div[1]")
titleButtonNum = len(titleButtonElementList)
print("Found %s search result title" % titleButtonNum)

# descriptionElementList = d.xpath("//*[@resource-id='result_main']/div/div[1]/div/div[2]/div[1]/div[1]/div[2]")
descriptionElementList = d.xpath("//*[@resource-id='result_main']/div/div[1]/div/div[2]/div[1]/div[1]/div[2]")
descriptionNum = len(descriptionElementList)
print("Found %s description" % descriptionNum)

# # sourceWebsiteElementList = d.xpath('//*[@resource-id="result_main"]/div/div[1]/div/div[2]/div[1]/div[1]/div[3]')
# sourceWebsiteElementList = d.xpath('//*[@resource-id="result_main"]/div/div[1]/div/div[2]/div[1]/div[1]/div[3]')
# sourceWebsiteNum = len(sourceWebsiteElementList)
# print("Found %s source website" % sourceWebsiteNum)

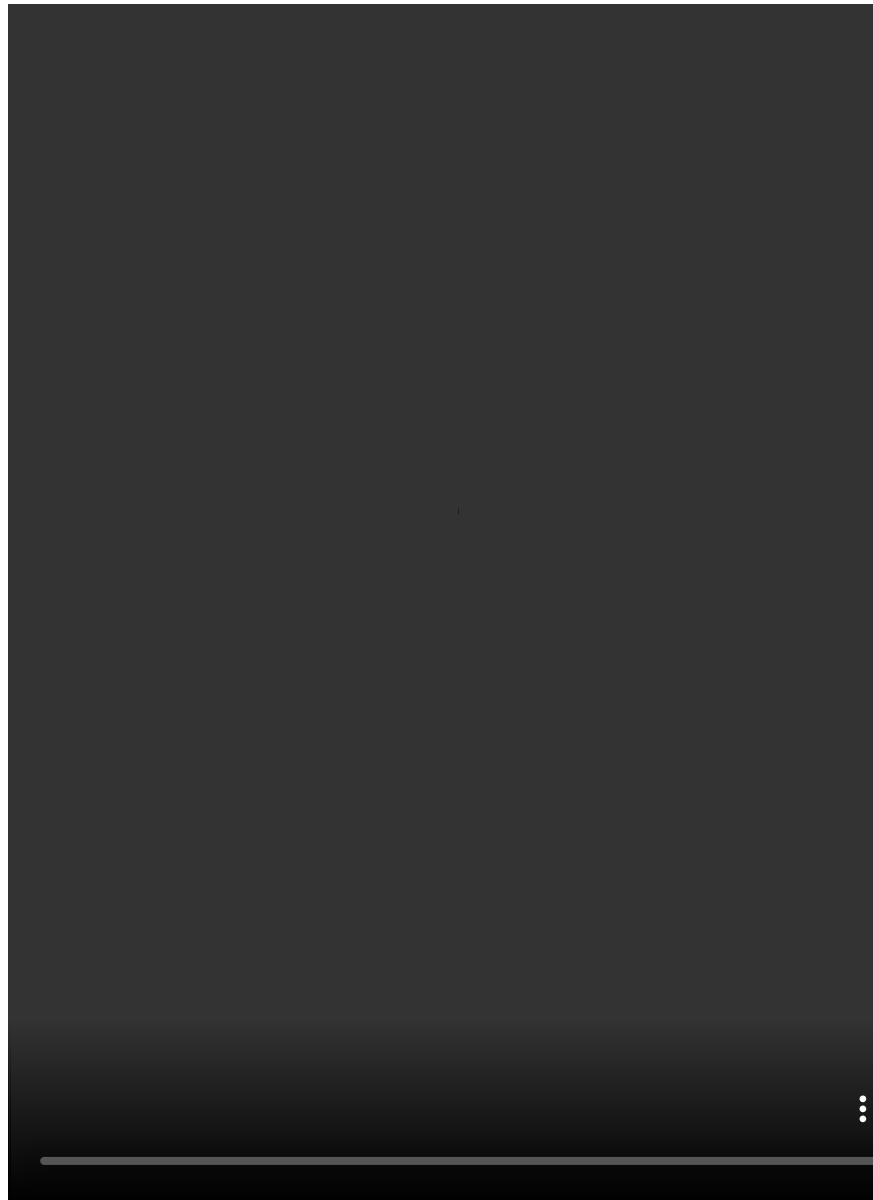
for curIdx, eachTitleButtonElement in enumerate(titleButtonElementList):
    curNum = curIdx + 1
    print("%s [%d/%d] %s" % ("="*20, curNum, titleButtonNum))
    # eachTitleButtonElemAttrib = eachTitleButtonElement.attrib
    # print("title attrib: %s" % eachTitleButtonElemAttrib)
    # curTitle = eachTitleButtonElemAttrib["text"]
    curTitle = eachTitleButtonElement.text
    print("title=%s" % curTitle)

    curDescriptionElem = descriptionElementList[curIdx]
    curDescription = curDescriptionElem.text
    print("description=%s" % curDescription)

    # curSourceWebsiteElem = sourceWebsiteElementList[curIdx]
    # curSourceWebsite = curSourceWebsiteElem.text
    # print("curSourceWebsite=%s" % curSourceWebsite)

print("Demo baidu search complete")
```

效果：



iOS端

facebook-wda

代码：

- 文件：[facebookWdaDemoBaiduSearch.py](#)
- 贴出来是

```
# Function: facebook-wda demo baidu search
# Author: Crifan Li
# Update: 20210410

import wda

# for debug
# Enable debug will see http Request and Response
# wda.DEBUG = True

c = wda.Client('http://localhost:8100')

curStatus = c.status()
print("curStatus=%s" % curStatus)
```

注：由于苹果开发者过期，导致：未完待续

详见：

【未解决】Mac中用facebook-wda自动操作安卓手机浏览器实现百度搜索

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2021-04-29 14:06:40

调试页面元素

在(用 `selenium`、`puppeteer`、`playwright` 等)自动化操作浏览器期间，往往涉及到：

搞清楚当前页面中的某些元素的html源码，以便于转换成 `xpath` 或 `css selector` 等方式去定位查找元素。

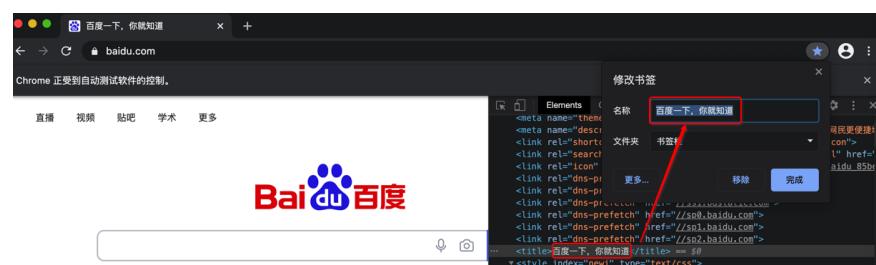
而调试页面元素的最常用办法就是： Chrome 的 开发者工具

其中： `Chrome = Chromium`

下面通过此处用到的实例例子来说明，具体如何操作。

用Chrome或Chromium查看百度首页中各元素的html源码

查看百度首页title标题



即： 百度一下，你就知道

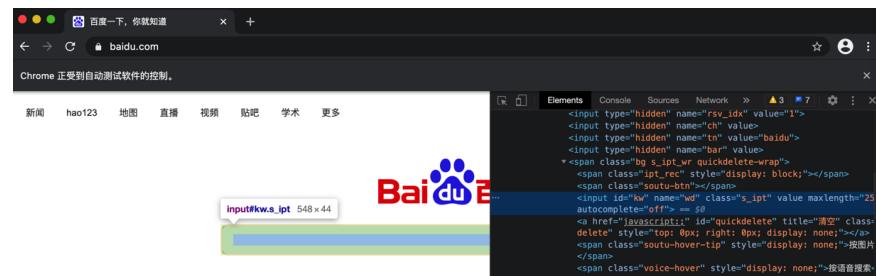
找到输入框对应的元素

可以右键 输入框 检查

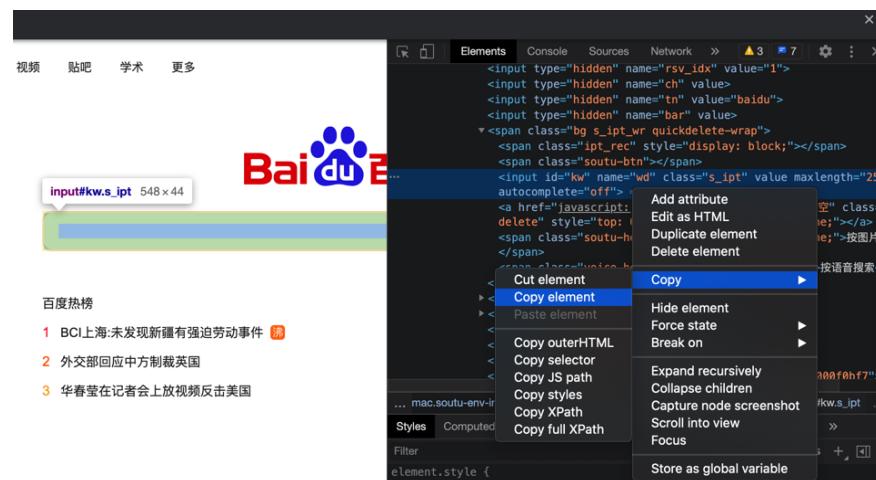
视频 贴吧 学术 更多



打开 Chrome的 开发者工具



可以看到对应的html，且可以右键去Copy拷贝出来对应html



另外 Playwright调用的Chromium中效果是：



拷贝出来是：

```
<input id="kw" name="wd" class="s_ipt" value="" maxlength="255" type="text"/>
```

后来注意到：

Chromium中 调试工具已实时显示出 定位元素的Selector的， 可以写成：

```
input#kw.s_ipt
```

其中：

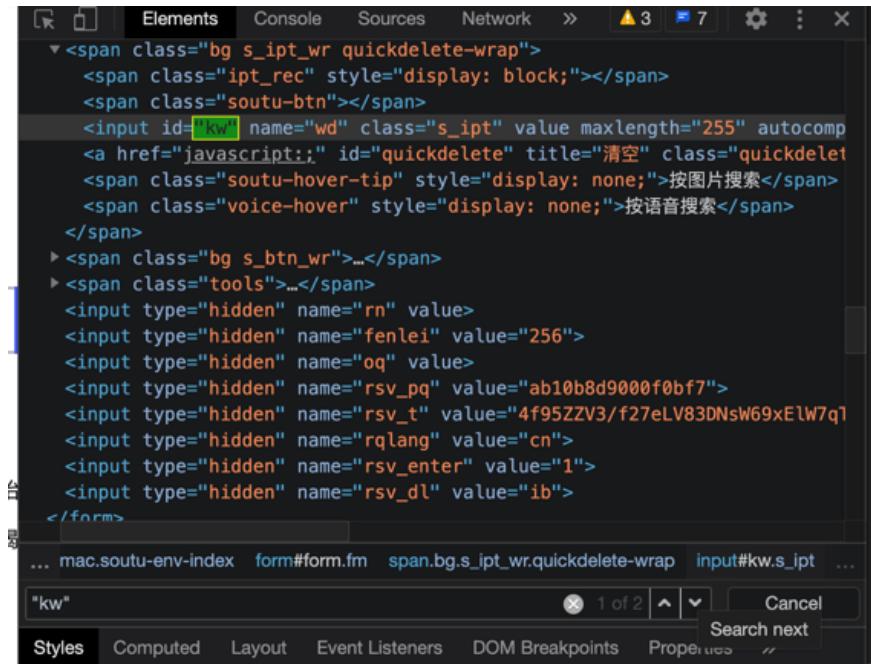
- `input` : 元素名 tag
- `kw` : 是 id
- `s_ipt` : 是 class

→ 后续代码中定位元素的CSS的Selector，则可以借鉴，甚至直接用这个写法

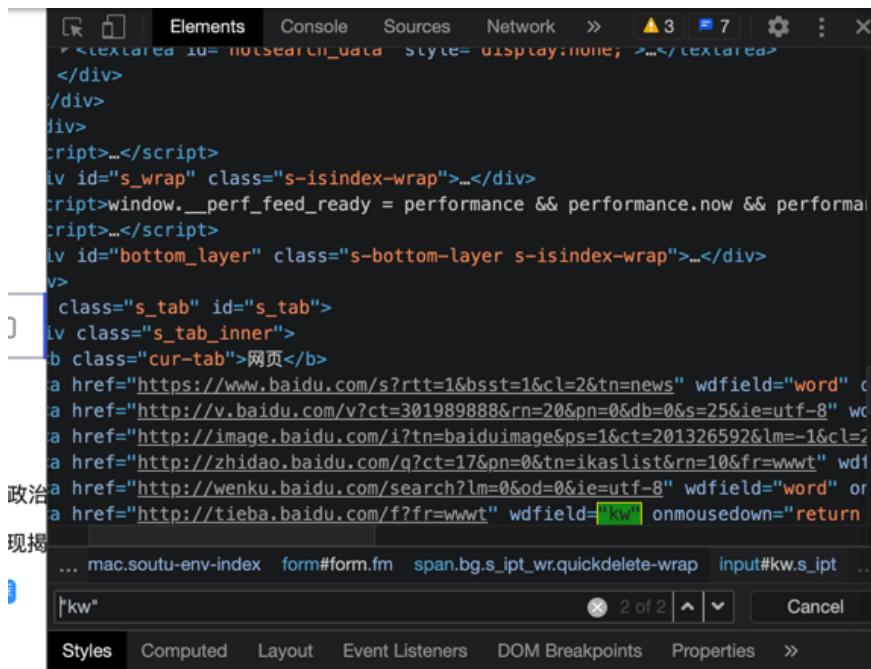
确认id是否唯一

此处可以通过查找，确认此处的id值 kw 是否唯一：

去搜一下此处的id： `kw`



此处可以看到搜到了2个，不过很明显，另外一个不是id：



证明对于： id="kw" 是唯一的

->后续代码，可以直接用 id="kw" 去定位元素（而可以不用其他属性，比如class等值）

找 百度一下 按钮的html

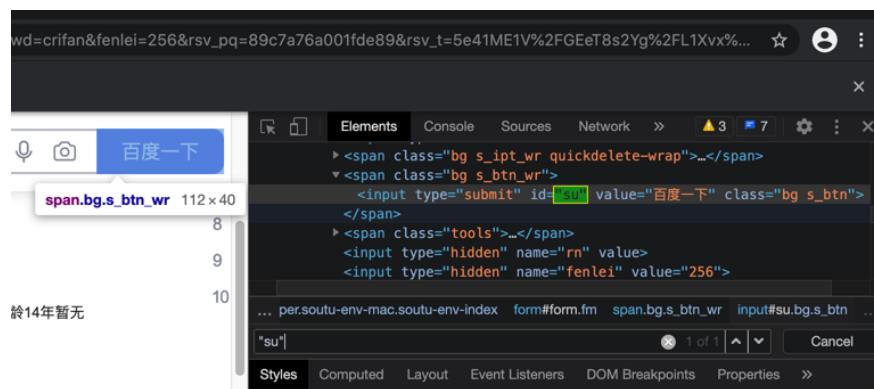
找 百度一下 按钮，和之前类似，去 右键检查：



可以看到html是：

```
<input type="submit" id="su" value="百度一下" class="bg_s_bt
```

且搜了下，确保只有一个：“su”



找到百度搜索页面肯定会出现的元素：百度为您找到相关结果约

去找百度搜索后，确保会出现的内容

找到这个：

```
百度为您找到相关结果约xxx个
```

百度搜索自动化



crifan_百度搜索 x asgdjojgkajg_百度搜索 x +

baidu.com/s?ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd=crifan&fenlei=256&rsv_pq=

Chrome 正受到自动测试软件的控制。

Baidu 百度 crifan 百度一下

Q 网页 资讯 视频 图片 知道 文库 贴贴吧 地图 采购 更多

百度为您找到相关结果约2,370,000个

您要找的是不是:tritan

在路上on the way - 走别人没走过的路,让别人有路可走

google 收录查询谷歌收录查询,如何让Google收录网站 – 云点SEO (yundianseo.com) Google收录查询 – 站长工具 (free.fr) site:www.crifan.com 网站域名... www.crifan.com/ 百度快照

crifan – 在路上

google 收录查询谷歌收录查询,如何让Google收录网站 – 云点SEO (yundianseo.com) Google收录查询 – 站长工具 (free.fr) site:www.crifan.com 网站域名... www.crifan.com/author/crifan/ 百度快照

且去确认了，故意搜不到内容，页面也会出现这个：



crifan_百度搜索 x asgdjojgkajg_百度搜索 x +

baidu.com/s?ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd=asgdjojgkajg&fenlei=256&rsv_pq=

Chrome 正受到自动测试软件的控制。

Baidu 百度 asgdjojgkajg 百度一下

Q 网页 资讯 视频 图片 知道 文库 贴贴吧 地图 采购 更多

百度为您找到相关结果约0个

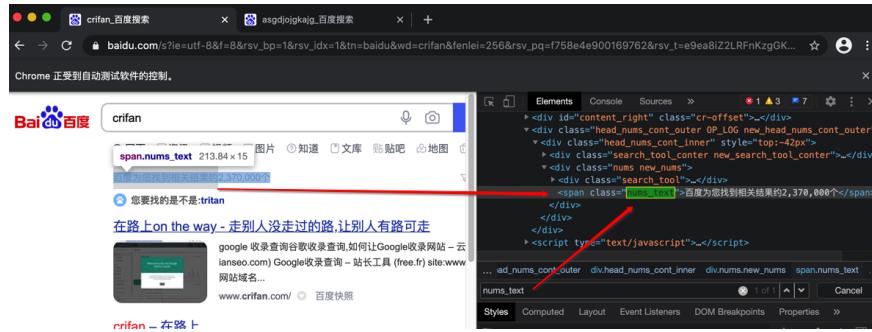
抱歉没有找到与“asgdjojgkajg”相关的网页。

温馨提示：
请检查您的输入是否正确
如网页未收录或者新站未收录，请[提交网址](#)给我们
如有任何意见或建议，请及时[反馈给我们](#)

相关搜索
[gk是谁](#) [objk](#)

去看看其html：

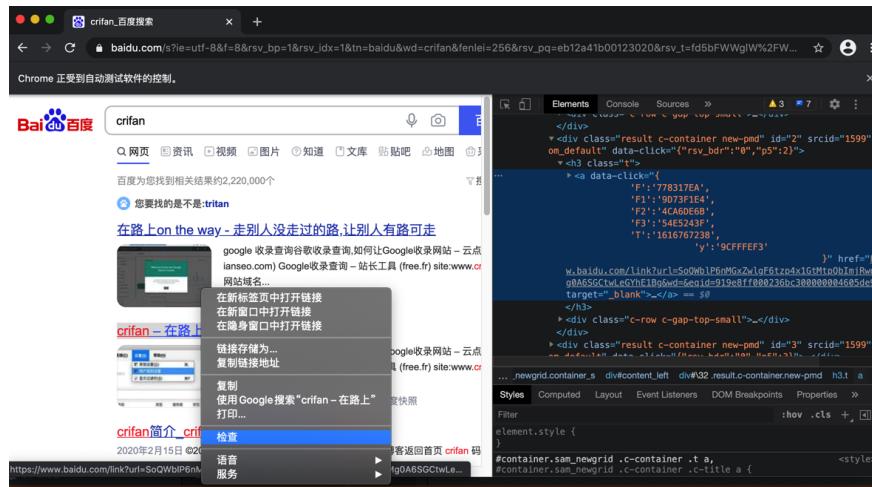
```
<span class="nums_text">百度为您找到相关结果约2,370,000个</span>
```



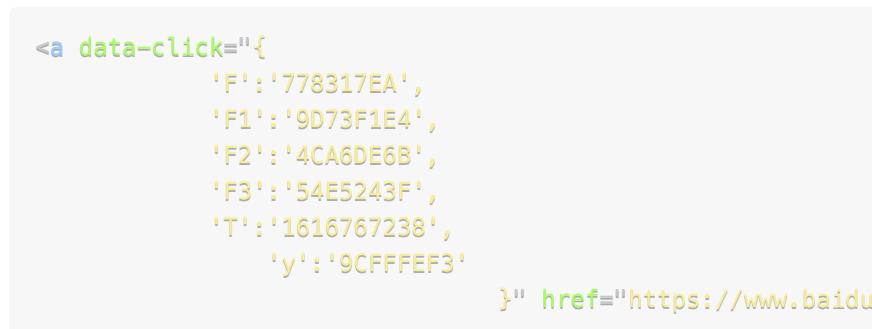
百度搜索的每条结果的html

去搞清楚，本身此处的每条搜索结果的内容的html是什么

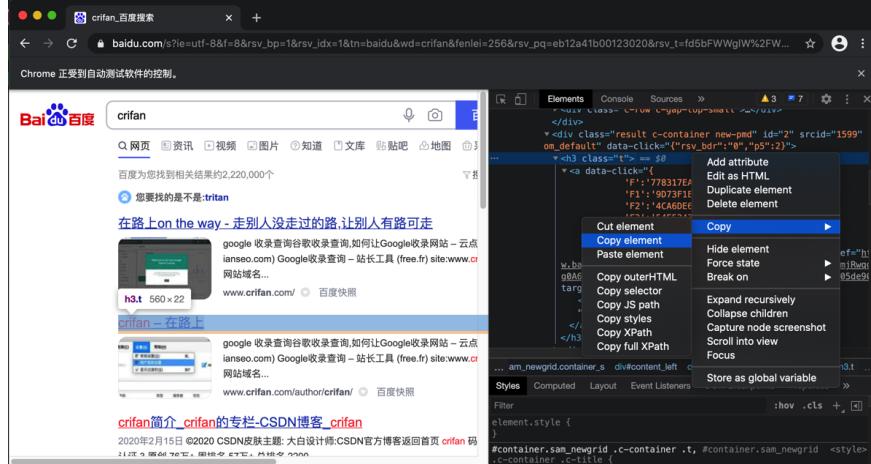
右键 检查：



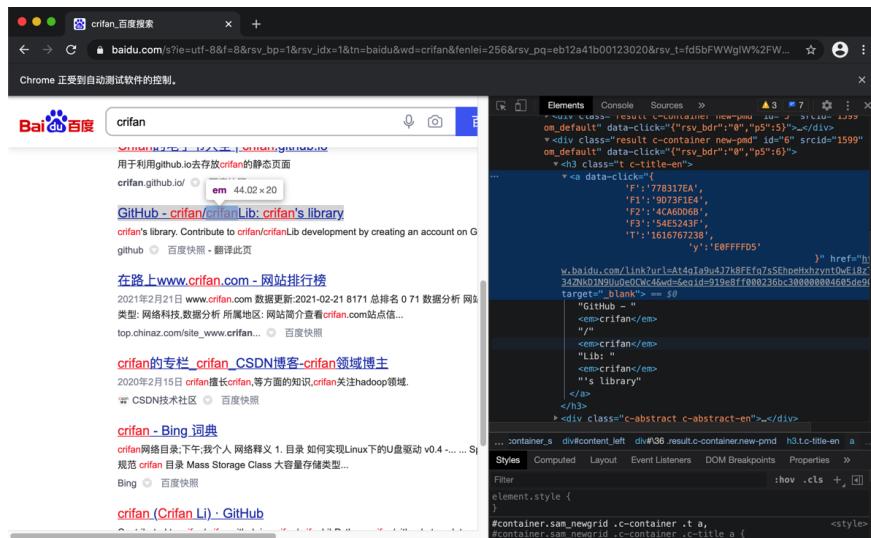
找到是：



上层父节点的元素是：



多看看几个结果，是否都是同样格式：

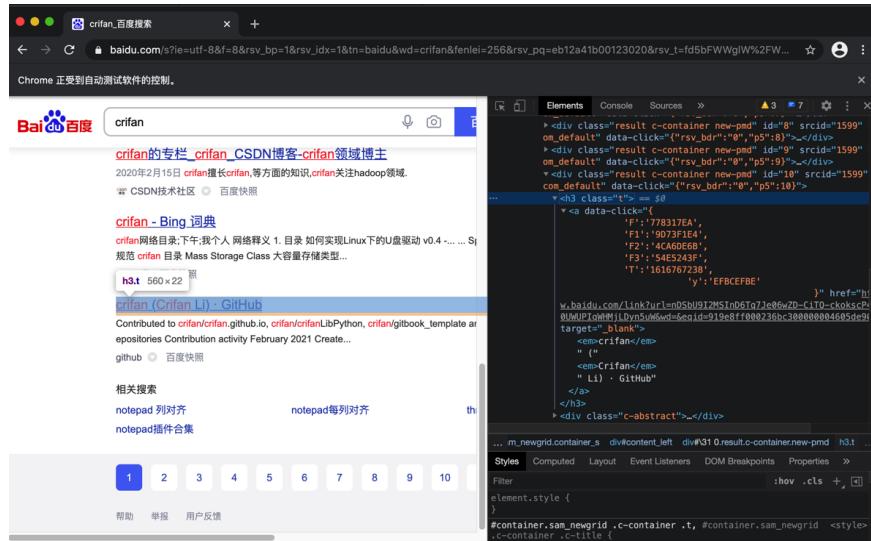


这个稍微复杂点：

```
<h3 class="t c-title-en"><a data-click="{
  'F': '778317EA',
  'F1': '9D73F1E4',
  'F2': '4CA6DD6B',
  'F3': '54E5243F',
  'T': '1616767238',
  'y': 'E0FFFFD5'
}" href="#">
```

以及另外一个：

```
<h3 class="t"><a data-click="{
  'F': '778317EA',
  'F1': '9D73F1E4',
  'F2': '4CA6DE6B',
  'F3': '54E5243F',
  'T': '1616767238',
  'y': 'EFBCEFBE'
}" href="#">
```

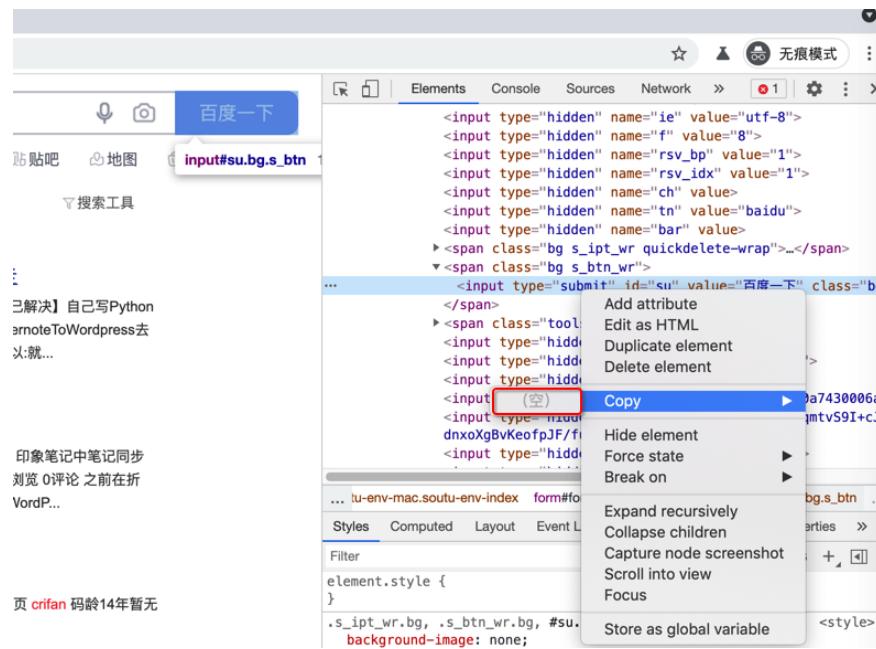


常见问题

Playwright的Chromium中无法右键拷贝元素 html

Playwright的Chromium中，虽然能打开 开发者工具

但是，右键无法复制copy元素的html，右键的copy是空



crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新: 2021-04-29 14:06:40

PC端

PC端 = 电脑端，的自动化操作，此处主要指的是：

- Web端 = 浏览器

的自动化操作，代替人手，自动操作浏览器。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2021-04-29 14:06:40

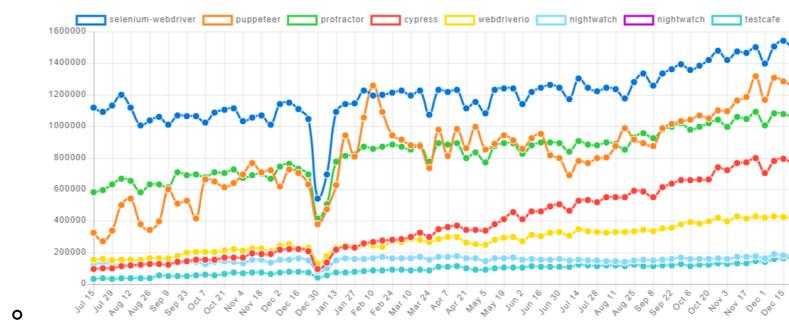
Web端

- Web端自动化操作
 - 目的：模拟人手工去操作浏览器
- 常见框架有
 - Selenium
 - puppeteer
 - Playwright

Selenium vs puppeteer

下面总结一下相关对比：

- 两者趋势



具体区别：

- Selenium

- Logo



- 有些网站能检测到是WebDriver，就无法继续爬取了
 - 注：通过 webdriver 对浏览器的每一步操作都会留下特殊的痕迹，会被很多网站识别到
 - 规避办法：必须通过重新编译chrome的webdriver才能实现
 - 麻烦得让人想哭
 - 某人评论：Selenium速度慢，现在都改用 puppeteer 了
 - 资料
 - 官网
 - [SeleniumHQ Browser Automation](#)

- Python版本
 - PyPI
 - [selenium · PyPI](#)
 - 文档
 - [Selenium with Python — Selenium Python Bindings 2 documentation](#)
- webdriver
 - 常见
 - Phantomjs
 - [官网](#)
 - 资料
 - [selenium-webdriver](#)
 - 优势
 - 历史悠久：2004年发布
 - 目前最主流的浏览器（web页面）自动化工具
 - 支持众多浏览器：
器：[Chrome](#)、[Firefox](#)、[Safari](#)、[IE](#)、[Opera](#) 等
 - 支持众多编程语言：[Java](#)、[C#](#)、[Python](#)、[Ruby](#) 等
 - 通过 Selenium IDE 支持录制功能
 - 支持测试平台：[Web](#)、（通过 [Appium](#)）支持移动端
 - 缺点
 - 速度相对([Puppeteer](#))慢一点
 - 安装和设置相对([Puppeteer](#))麻烦一些
 - 不支持跨平台
 - 截图只支持图片
- Puppeteer
 - Logo
 - 发布时间：2017年
 - 开发者：[Google](#)
 - 目标：简化前端测试(front-end test)和开发
 - 支持浏览器：[Chrome](#)、[Chromium](#)
 - 支持语言：[Javascript](#) ([Node.js](#))
 - 优势
 - 速度相对快一些
 - 安装和设置相对简单

- 支持跨平台
- 截图支持图片和PDF
- 缺点
 - 测试平台只支持: Web
- 相关
 - [puppeteer](#)
 - 是什么: Puppeteer 的 python 的 binding
 - Unofficial Python port of puppeteer JavaScript (headless) chrome/chromium browser automation library
 - 好处
 - 可以绕过很多网站对于WebDriver的检测
 - 可以对 js加密 降维打击
 - 完全无视 js加密 手段
 - 文档
 - [API Reference — Pypeteer 0.0.25 documentation](#)
 - 官网
 - GitHub
 - [miyakogi/pypeteer: Headless chrome/chromium automation library \(unofficial port of puppeteer\)](#)
 - 注: 代码已归档, 变只读了
 - pypi
 - [pypeteer · PyPI](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2021-04-29 14:06:40

Selenium

详见专门教程：

[Selenium知识总结](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新： 2021-04-29 14:06:40

puppeteer

背景

前端就有了对 headless 浏览器的需求，最多的应用场景有两个

1. UI自动化测试：摆脱手工浏览点击页面确认功能模式
2. 爬虫：解决页面内容异步加载等问题

前端经常使用的莫过于

- PhantomJS
 - <http://phantomjs.org/>
- selenium + webdriver
 - <http://seleniumhq.github.io/selenium/docs/api/javascript/>

但两个库有一个共性：

- 难用
 - 环境安装复杂
 - API 调用不友好

2017 年 Chrome 团队连续放了两个大招

- Headless Chromium
 - <https://chromium.googlesource.com/chromium/src/+/lkgr/headless/README.md>
- NodeJS API Puppeteer
 - <https://github.com/GoogleChrome/puppeteer>

-> 直接让 PhantomJS 和 Selenium IDE for Firefox 作者悬宣布没必要继续维护其产品

我们手工可以在浏览器上做的事情 Puppeteer 都能胜任

1. 生成网页截图或者 PDF
2. 爬取大量异步渲染内容的网页，基本就是人肉爬虫
3. 模拟键盘输入、表单自动提交、UI 自动化测试

概述

- puppeteer
 - 概述：微软开发的基于 Javascript 的 web 自动化的库
- 官网
 - [github](#)
 - puppeteer/puppeteer: Headless Chrome Node.js API

- <https://github.com/puppeteer/puppeteer>
- google
 - Puppeteer | Tools for Web Developers | Google Developers
 - <https://developers.google.com/web/tools/puppeteer>
- 优势
 - 可以用TypeScript编写测试
 - Devs还可以在运行测试时连接Chrome DevTools

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2021-05-12 09:06:47

初始化环境

此处介绍如何（在Mac中）初始化 `puppeteer` 开发环境。

下载和安装 `puppeteer`

- Mac中安装 `puppeteer`

```
pip install puppeteer
```

- 用 `puppeteer-install` 去下载浏览器内核

```
puppeteer-install
```

- 可以看到下载了chrome

- 此处位置是： /Users/crifan/Library/Application
Support/puppeteer/local-chromium/588429

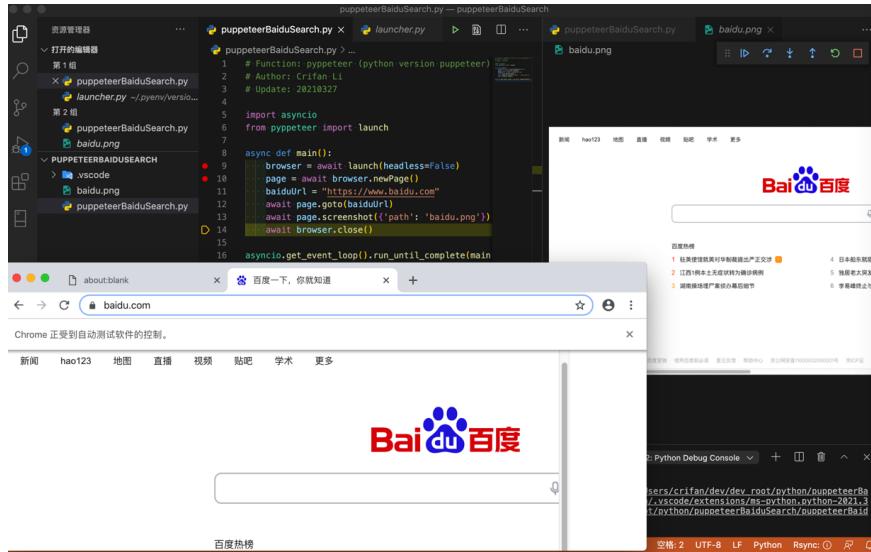
测试代码

```
import asyncio
from puppeteer import launch

async def main():
    browser = await launch(headless=False)
    page = await browser.newPage()
    baiduUrl = "https://www.baidu.com"
    await page.goto(baiduUrl)
    await page.screenshot({'path': 'baidu.png'})
    await browser.close()

asyncio.get_event_loop().run_until_complete(main())
```

即可，启动Chromium浏览器，并打开百度，和本地截图：



常见问题

puppeteer代码正常运行，但没有启动Chrome浏览器

现象： puppeteer 代码

```
browser = await launch()
```

是正常运行了，但是没看到Chrome浏览器启动

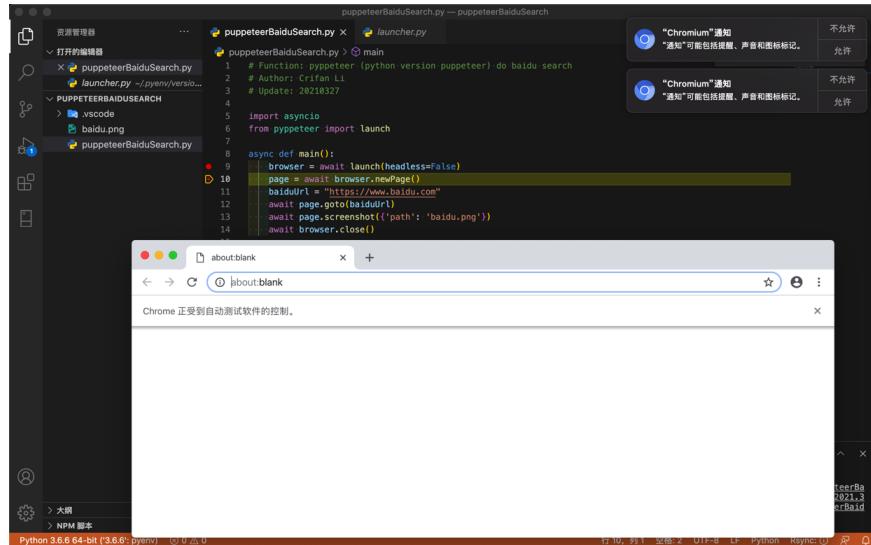
原因： puppeteer (puppeteer) 默认是启动 无头模式，所以内部其实启动了，只是没有界面显示，即看不到Chrome浏览器启动而已。

解决办法：加上参数，取消无头模式

代码：

```
browser = await launch(headless=True)
```

即可看到Chrome浏览器



参数传递方式也可以用dict字典方式

也可以写成dict字典的方式传参

```
browser = await launch({'headless': False})
```

效果是一样的

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新: 2021-05-10 13:34:52

基本操作

此处介绍puppeteer的常见的基本操作。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2021-05-12 09:06:13

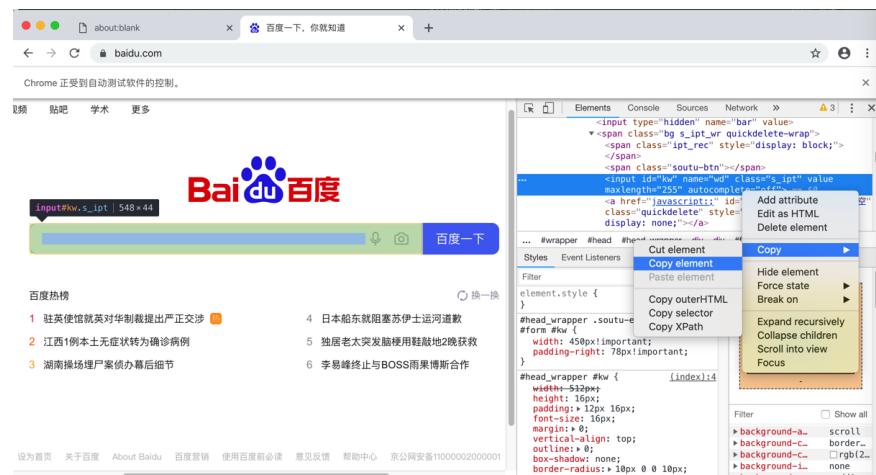
查找定位元素

查找元素相关函数：

- `puppeteer`
 - `Page.querySelector()`
 - 别名: `Page.J()`
 - `Page.querySelectorAll()`
 - 别名: `Page.JJ()`
 - `Page.xpath()`
 - 别名: `Page.Jx()`

单个查找 xpath

对于页面：



对应html：

```
<input id="kw" name="wd" class="s_ipt" value="" maxlength="255" type="text"/>
```

代码：

```
SearchButtonXpath = "//input[@id='kw']"
searchButtonElem = page.xpath(SearchButtonXpath)
print("searchButtonElem=%s" % searchButtonElem)
```

输出：

```
searchButtonElem=<coroutine object Page.xpath at 0x10f15cd1>
```

调试效果：

```

puppeteerBaiduSearch.py — puppeteerBaiduSearch
=====
1 # Function: puppeteer (python version puppeteer) do baidu search
2 # Author: Crifan Li
3 # Update: 20210327
4
5 import asyncio
6 from <coroutine object Page.xpath at 0x10f15cd0>
7
8 @asyncio.coroutine
9 > special variables
10 > cr_await: None
11 > cr_code: <code object xpath at 0x10feef930, file "/Users/crifan/.pyenv/versions/3.6.6/lib/python3.6/si
12 > cr_frame: <frame object at 0x10f1c9748>
13 > cr_running: False
14 S 按键 Option 键切换到编辑器语言停
15 searchButtonItem = page.xpath(SearchButtonXPath)
16 print("searchButtonItem=%s" % searchButtonItem)
17 | await browser.close()
18
19 asyncio.get_event_loop().run_until_complete(main())
20
    
```

调试器界面显示了当前的断点、堆栈跟踪（main->puppeteerBaiduSearch-><module>）、全局变量以及Python控制台输出。

批量查找 querySelectorAll

对于html

```

<h3 class="t"><a data-click="{
  'F': '778317EA',
  'F1': '9D73F1E4',
  'F2': '4CA6DE6B',
  'F3': '54E5243F',
  'T': '1616767238',
  'y': 'EFBCEFBE'
}" href="https://www.baidu.com/link?url=nDSbU9"/>
    
```

想要查找=定位（所有的）元素 a

```

h3ASelector = "h3[class^='t'] a"
aElemList = await page.querySelectorAll(h3ASelector)
print("aElemList=%s" % aElemList)
    
```

即可找到元素：

```

puppeteerBaiduSearch.py — puppeteerBaiduSearch
=====
40 # trigger search
41 #####
42 # <puppeteer.element_h...107f4855b, <puppeteer.element_h...107f4847b, <puppeteer.element_h...107f48e1...
43 # @param
44 # @return
45 # @meth
46 # @meth
47 # @meth
48 # @meth
49 # @meth
50 # @meth
51 # @meth
52 # @meth
53 # @meth
54 # @meth
55 # @meth
56 # @meth
57 # @meth
58 # @meth
59 # @meth
60 # @meth
61 len(): 10
    
```

代码块展示了如何使用 `querySelectorAll` 方法来批量查找所有符合条件的元素，并打印出它们的列表。

输入文字

对于页面元素：

```
<input id="kw" name="wd" class="s_ipt" value="" maxlength="100"
```

想要输入文字

不支持定位元素再输入

此处，无需，也没法实现：先定位元素，再输入

而是，只支持：直接定位并输入

多种实现方式

方式1：先 Selector 定位，再 type 输入

定位此处只适合用： Selector，具体写法是：

```
"input[id='kw']"
```

然后输入是用 type 函数

完整代码：

```
searchStr = "crifan"
SearchInputSelector = "input[id='kw']"

await page.type(SearchInputSelector, searchStr, delay=20)
```

方式2：先 focus，再 keyboard 的 type

先（通过selector） focus，再（用keyboard） type

完整代码：

```
searchStr = "crifan"
SearchInputSelector = "input[id='kw']"

await page.focus(SearchInputSelector)
await page.keyboard.type(searchStr)
```

方式3：先 focus，再 click，最后用 keyboard 的 type

先 selector，再click（类似于focus），最后用keyboard输入type

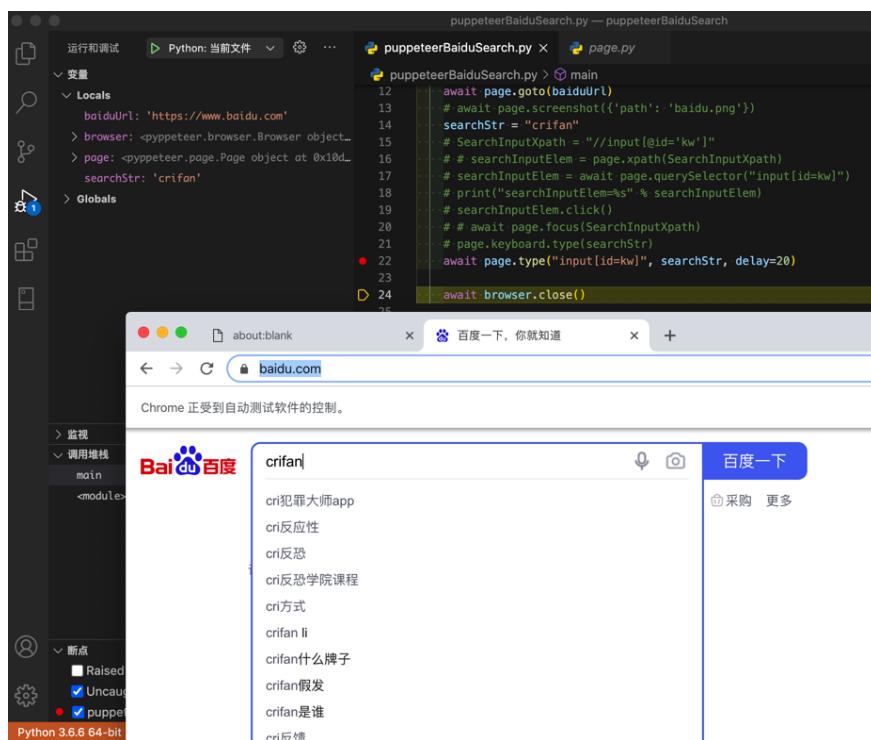
完整代码：

```
searchStr = "crifan"
SearchInputSelector = "input[id='kw']"

searchInputElem = await page.querySelector(SearchInputSelector)
await searchInputElem.click()
await page.keyboard.type(searchStr)
```

效果

百度首页的输入框中，输入了字符串 crifan 后的效果：



注意事项

加记得加 await

注意：一定要加await，否则：代码运行无效果。

且还会报警告：

```
/Users/crifan/dev/dev_root/python/puppeteerBaiduSearch/pup  
searchInputElement.click()
```

相关文档

- coroutine type(selector: str, text: str, options: dict = None, **kwargs)
→ None[source] API Reference — Pypeteer 0.0.25 documentation
 - <https://miyakogi.github.io/puppeteer/reference.html#puppeteer.page.Page.type>

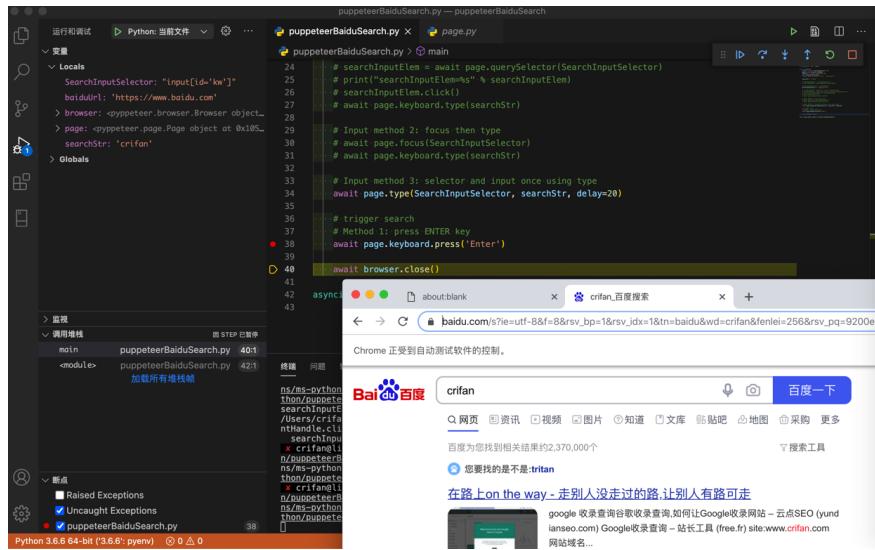
crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新: 2021-05-12 09:06:10

模拟按键

模拟 回车键 :

```
await page.keyboard.press('Enter')
```

此处效果：可以触发百度搜索，显示搜索结果：



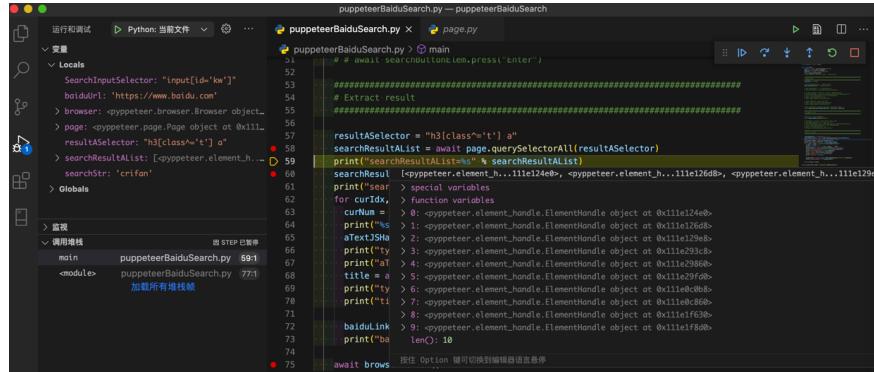
crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新: 2021-05-12 09:05:52

等待元素出现

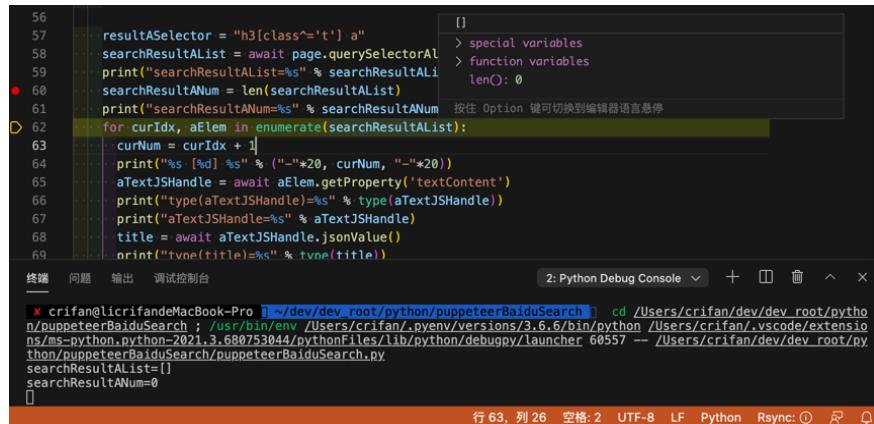
现象：对于代码：

```
resultASelector = "h3[class^='t'] a"
searchResultAList = await page.querySelectorAll(resultASelector)
```

调试时可以正常运行，可以找到元素：



直接运行时，却找不到元素了：



原因：页面重新加载了，但是内容还没显示出来。所以找不到元素。

解决办法：等待页面加载完毕。再去查找元素，就可以找到了。

如何确保页面加载完毕？

核心逻辑：找到页面加载完毕，一定会显示（出现）的元素，去等待其出现，即可。

此处，百度搜索后，一定会出现（显示）的元素是：

```
<span class="nums_text">百度为您找到相关结果约2,370,000个</span>
```

对应等待元素出现

- (好的) 方法1: `querySelector + sleep`

- 代码

```
SearchFoundWordsSelector = 'span.nums_text'
SearchFoundWordsXpath = "//span[@class='nums_te

# Method 2: wait element showing
SingleWaitSeconds = 1
while not await page.querySelector(SearchFoundW
print("Still not found %s, wait %s seconds" %
await asyncio.sleep(SingleWaitSeconds)
```

- (不够好的) 方法2: 直接wait等待

- 代码

```
# # Method 1: just wait
await page.waitFor(2000) # millisecond
```

- 评价: 不够好, 不能精确判断元素是否出现

waitFor系列的所有函数都无效

经过实际测试, `waitFor` 系列的各个函数, 此处都无效

```
# await page.waitForSelector(SearchFoundWordsSelector)
# await page.waitFor(SearchFoundWordsSelector)
# await page.waitForXPath(SearchFoundWordsXpath)
# Note: all above exception: 发生异常: ElementHandleError
```

都会报错: `ElementHandleError Evaluation failed: TypeError: MutationObserver is not a constructor`

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook 最后更新: 2021-05-12 09:06:23

获取元素属性

获取元素属性可以用: `someElement.getProperty('propertyName')`

举例:

```
<h3 class="t"><a data-click="{
  'F': '778317EA',
  'F1': '9D73F1E4',
  'F2': '4CA6DE6B',
  'F3': '54E5243F',
  'T': '1616767238',
  'y': 'EFBCEFBE'
}" href="https://www.baidu.com/link?url=nDSbU9I
```

中的 `a` 元素中的 `href` 和文本值

对于已经找到元素的列表:

```
resultASelector = "h3[class^='t'] a"
searchResultAList = await page.querySelectorAll(result)
# print("searchResultAList=%s" % searchResultAList)
searchResultANum = len(searchResultAList)
print("Found %s search result:" % searchResultANum)
```

后去获取文本值 `text` 和属性值 `href`:

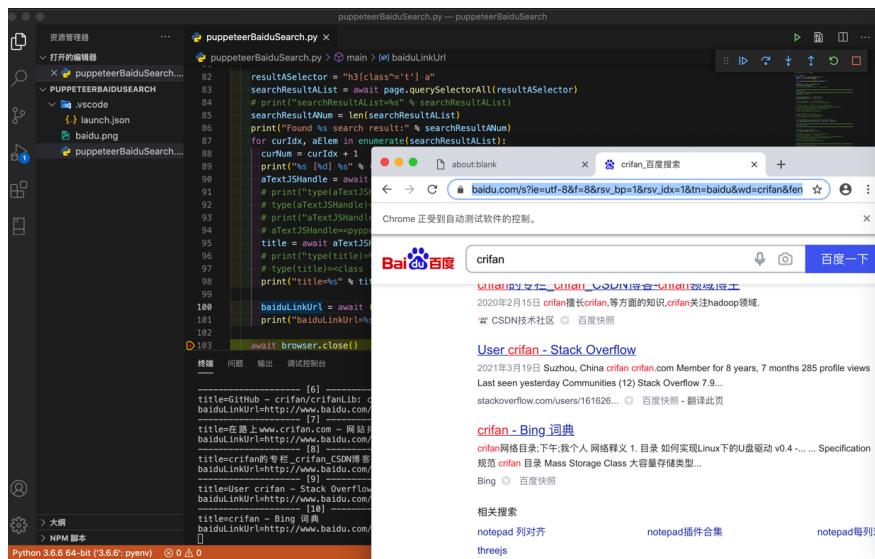
```
for curIdx, aElem in enumerate(searchResultAList):
    curNum = curIdx + 1
    print("%s [%d] %s" % ("-"*20, curNum, "-"*20))
    aTextJSHandle = await aElem.getProperty('textContent')
    # print("type(aTextJSHandle)=%s" % type(aTextJSHandle))
    # type(aTextJSHandle)=<class 'puppeteer.execution_context.JSHandle'>
    # print("aTextJSHandle=%s" % aTextJSHandle)
    # aTextJSHandle=<puppeteer.execution_context.JSHandle>
    title = await aTextJSHandle.jsonValue()
    # print("type(title)=%s" % type(title))
    # type(title)=<class 'str'>
    print("title=%s" % title)

    baiduLinkUrl = await (await aElem.getProperty("href"))
    print("baiduLinkUrl=%s" % baiduLinkUrl)
```

输出:

```
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=eGTzEXXlMw-hnvX  
----- [2] -----  
title=crifan - 在路上  
baiduLinkUrl=http://www.baidu.com/link?url=l6jXejlgARRwJ34C
```

效果：



crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新：2021-05-12 09:06:03

举例

下面给出具体的puppeteer的实例案例供参考。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2021-05-12 09:06:35

百度搜索自动化

此处给出用 `puppeteer` 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：[puppeteerDemoBaiduSearch.py](#)
- 贴出代码

```
# Function: pypeteer (python version puppeteer) do baidu search
# Author: Crifan Li
# Update: 20210330

import asyncio
from pypeteer import launch

async def main():
    browser = await launch(headless=False)
    page = await browser.newPage()

    await page.setJavaScriptEnabled(enabled=True)

    baiduUrl = "https://www.baidu.com"
    await page.goto(baiduUrl)
    # await page.screenshot({'path': 'baidu.png'})

    #####
    # Input text
    #####
    searchStr = "crifan"

    # SearchInputSelector = "input[id=kw]"
    SearchInputSelector = "input[id='kw']"

    # SearchInputXpath = "//input[@id='kw']"
    # searchInputElem = page.xpath(SearchInputXpath)

    # # Input method 1: selector + click + keyboard type
    # searchInputElem = await page.querySelector(SearchInputSelector)
    # print("searchInputElem=%s" % searchInputElem)
    # await searchInputElem.click()
    # await page.keyboard.type(searchStr)

    # Input method 2: focus then type
    # await page.focus(SearchInputSelector)
    # await page.keyboard.type(searchStr)

    # Input method 3: selector and input once using type
    await page.type(SearchInputSelector, searchStr, delay=100)

    #####
    # Trigger search
    #####
    # Method 1: press ENTER key
    await page.keyboard.press('Enter')

    # # Method 2: locator search button then click
    # SearchButtonSelector = "input[id='su']"
```

```

# searchButtonElem = await page.querySelector(SearchButtonSelector)
# print("searchButtonElem=%s" % searchButtonElem)
# await searchButtonElem.click()
# # await searchButtonElem.press("Enter")

#####
# Wait page reload complete
#####
SearchFoundWordsSelector = 'span.nums_text'
SearchFoundWordsXpath = "//span[@class='nums_text']"

# await page.waitForSelector(SearchFoundWordsSelector)
# await page.waitFor(SearchFoundWordsSelector)
# await page.waitForXPath(SearchFoundWordsXpath)
# Note: all above exception: 发生异常: ElementHandleError
# so change to following

# # Method 1: just wait
# await page.waitFor(2000) # millisecond

# Method 2: wait element showing
SingleWaitSeconds = 1
while not await page.querySelector(SearchFoundWordsSelector):
    print("Still not found %s, wait %s seconds" % (SearchFoundWordsSelector, SingleWaitSeconds))
    await asyncio.sleep(SingleWaitSeconds)
    # pass

#####
# Extract result
#####

resultASelector = "h3[class^='t'] a"
searchResultAList = await page.querySelectorAll(resultASelector)
# print("searchResultAList=%s" % searchResultAList)
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))
    aTextJSHandle = await aElem.getProperty('textContent')
    # print("type(aTextJSHandle)=%s" % type(aTextJSHandle))
    # type(aTextJSHandle)=<class 'puppeteer.execution_context.JSHandle'>
    # print("aTextJSHandle=%s" % aTextJSHandle)
    # aTextJSHandle=<puppeteer.execution_context.JSHandle>
    title = await aTextJSHandle.jsonValue()
    # print("type(title)=%s" % type(title))
    # type(title)=<class 'str'>
    print("title=%s" % title)

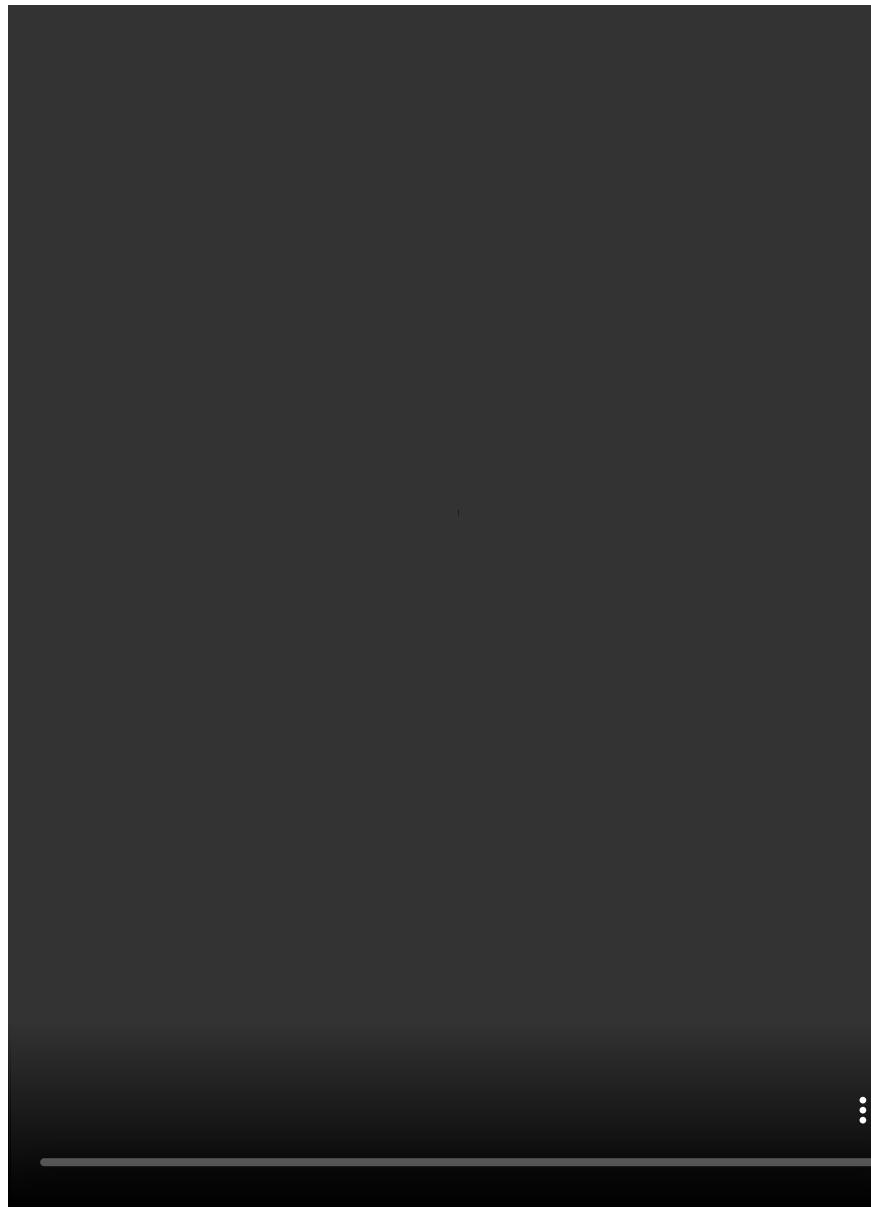
    baiduLinkUrl = await (await aElem.getProperty("href"))
    print("baiduLinkUrl=%s" % baiduLinkUrl)

```

```
await browser.close()  
asyncio.get_event_loop().run_until_complete(main())
```

效果

视频



图

百度搜索自动化

The screenshot illustrates a workflow for automating a search query on Baidu. On the left, a terminal window in VS Code runs a Python script using Puppeteer to interact with a browser instance. The script is designed to search for 'crifan' on Baidu and handle the search results. On the right, a browser window shows the search results for 'crifan' on Baidu, including links to various pages like Stack Overflow and GitHub profiles. The browser's developer tools are open, specifically the Network tab, which lists multiple requests made by the browser to fetch the search results.

输出

```
Still not found span.nums_text, wait 1 seconds
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=eGTzEXXlMw-hnvX
----- [2] -----
title=crifan - 在路上
baiduLinkUrl=http://www.baidu.com/link?url=l6jXejlgARrWj34C
----- [3] -----
title=crifan简介_crifan的专栏-CSDN博客_crifan
baiduLinkUrl=http://www.baidu.com/link?url=IIqPM5wuVE_QP7S
----- [4] -----
title=crifan的微博_微博
baiduLinkUrl=http://www.baidu.com/link?url=NnqeMlu4Jr_Ld-zc
----- [5] -----
title=Crifan的电子书大全 | crifan.github.io
baiduLinkUrl=http://www.baidu.com/link?url=uOZ-AmgNBNr3mGdt
----- [6] -----
title=GitHub - crifan/crifanLib: crifan's library
baiduLinkUrl=http://www.baidu.com/link?url=t42I1rYfn32DGw9C
----- [7] -----
title=在路上www.crifan.com - 网站排行榜
baiduLinkUrl=http://www.baidu.com/link?url=WwLwfXA72vK080by
----- [8] -----
title=crifan的专栏_crifan_CSDN博客-crifan领域博主
baiduLinkUrl=http://www.baidu.com/link?url=Cmcn2mXwiZr87FB
----- [9] -----
title=User crifan - Stack Overflow
baiduLinkUrl=http://www.baidu.com/link?url=yGgsq1z2vNDAAeW
----- [10] -----
title=crifan - Bing 词典
baiduLinkUrl=http://www.baidu.com/link?url=UatxhUBL3T_likP
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2021-05-12 09:06:32

puppeteer资料

- 官网
 - github
 - puppeteer/puppeteer: Headless Chrome Node.js API
 - <https://github.com/puppeteer/puppeteer>
 - google
 - Puppeteer | Tools for Web Developers | Google Developers
 - <https://developers.google.com/web/tools/puppeteer>
 - Quick start | Tools for Web Developers | Google Developers
 - <https://developers.google.com/web/tools/puppeteer/get-started>

Python 版 puppeteer : pypeteer

puppeteer 是基于 (NodeJS的) js 语言的

对应的Python版本的库是: pypeteer

- PyPI
 - pypeteer · PyPI
 - <https://pypi.org/project/pypeteer/>
- Github
 - 旧版 = miyakogi版
 - miyakogi/pypeteer: Headless chrome/chromium automation library (unofficial port of puppeteer)
 - <https://github.com/miyakogi/pypeteer>
 - 已经archive了
 - 最后更新: 8 May 2020
 - 对应文档
 - API Reference — Pypeteer 0.0.25 documentation
 - <https://miyakogi.github.io/pypeteer/reference.html>
 - Pypeteer's documentation — Pypeteer 0.0.25 documentation
 - <https://miyakogi.github.io/pypeteer/index.html>
 - pypeteer.page — Pypeteer 0.0.25 documentation
 - https://miyakogi.github.io/pypeteer/_modules/pypeteer/page.html
 - 新版 = pypeteer版
 - pypeteer/pypeteer: Headless chrome/chromium automation library (unofficial port of puppeteer)

- <https://github.com/puppeteer/puppeteer>
 - 但是是非官方的
 - 最后更新: 2021 9 Jan
- 对应文档
 - Puppeteer's documentation — Puppeteer 0.0.25 documentation
 - <https://puppeteer.github.io/puppeteer/>
 - API Reference — Puppeteer 0.0.25 documentation
 - <https://puppeteer.github.io/puppeteer/reference.html>

其他相关

CSS

如何写Selector去定位html元素，可参考：

- CSS选择器参考手册
 - https://www.w3school.com.cn/cssref/css_selectors.asp

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2021-05-12 09:06:40

Playwright

- `Playwright`
 - 一句话简介：微软开源 Python 自动化神器 `Playwright`
 - 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

安装

- 安装playwright库
 - `pip install playwright`
- 安装浏览器驱动文件（安装过程稍微有点慢）
 - `python -m playwright install`

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2021-05-12 09:05:31

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

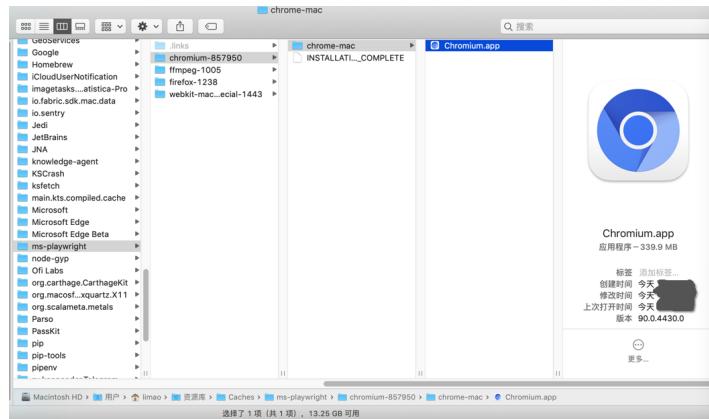
```
playwright install
```

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 [=====] 100%
firefox v1238 downloaded to /Users/limao/Library/Caches/ms-
Downloading webkit v1443 - 52 Mb [=====] 100%
webkit v1443 downloaded to /Users/limao/Library/Caches/ms-
Downloading ffmpeg v1005 - 1.3 Mb [=====] 100%
ffmpeg v1005 downloaded to /Users/limao/Library/Caches/ms-
```

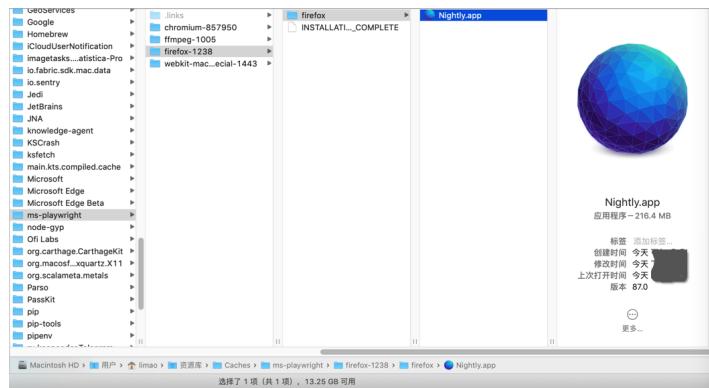
此处下载安装了：

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



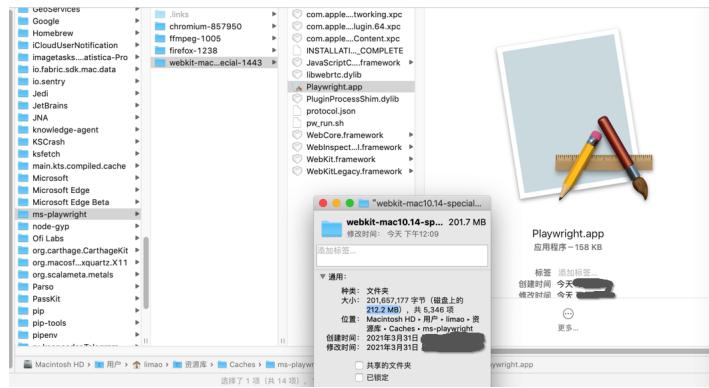
- **firefox**

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



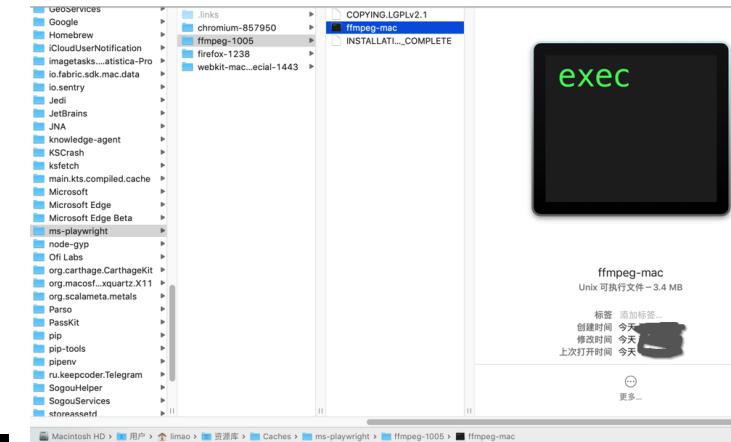
- **webkit**

- 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 executable_
    print("browser=%s" % browser)
    # browser=<Browser type=<BrowserType name=chromium executable_
    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_
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 version
  -h, --help              display help for command

Commands:
  open [options] [url]    open page in browser
  codegen [options] [url]  open page and generate
  debug <app> [args...]   run command in detached
  install [browserType...] ensure browsers needed
  install-deps [browserType...] install dependencies
  cr [options] [url]      open page in Chromium
  ff [options] [url]      open page in Firefox
  wk [options] [url]      open page in WebKit
  screenshot [options] <url> <filename> capture a page screenshot
  pdf [options] <url> <filename> save page as pdf
  help [command]         display help for command
```

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新: 2021-05-12 09:05:27

基本操作

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2021-05-12 09:05:03

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

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2021-05-12 09:04:53

查找并点击元素

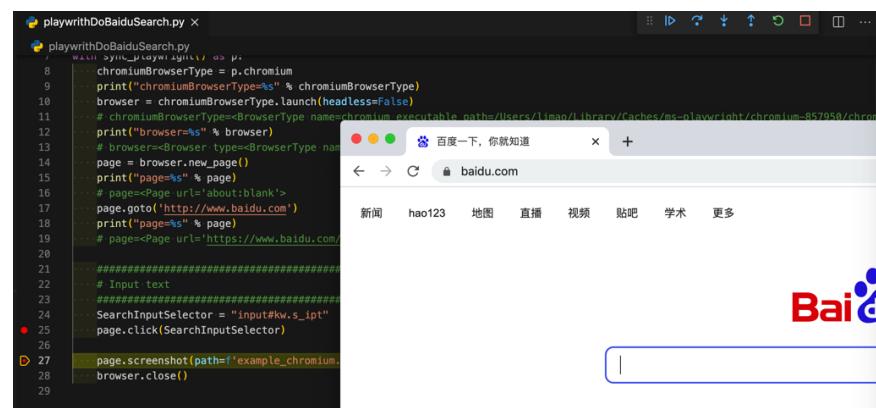
对于百度主页搜索输入框，html是：

```
<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：



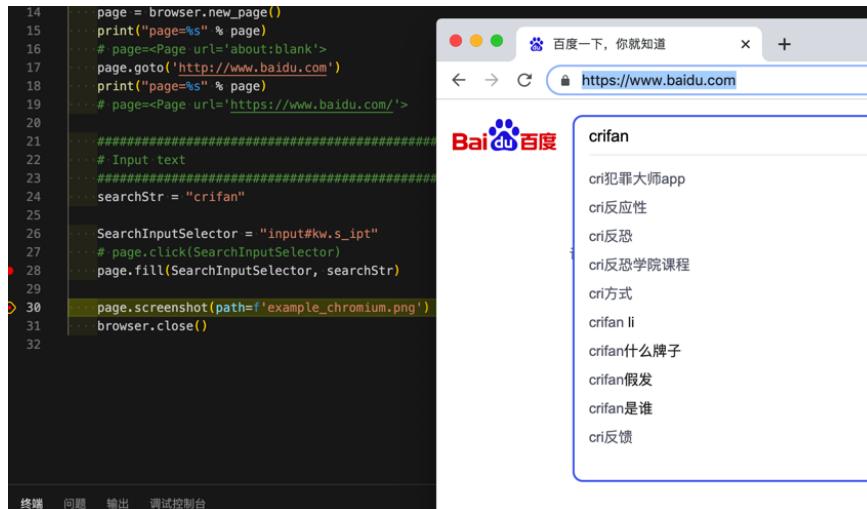
crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook 最后更新：2021-05-12 09:04:40

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>

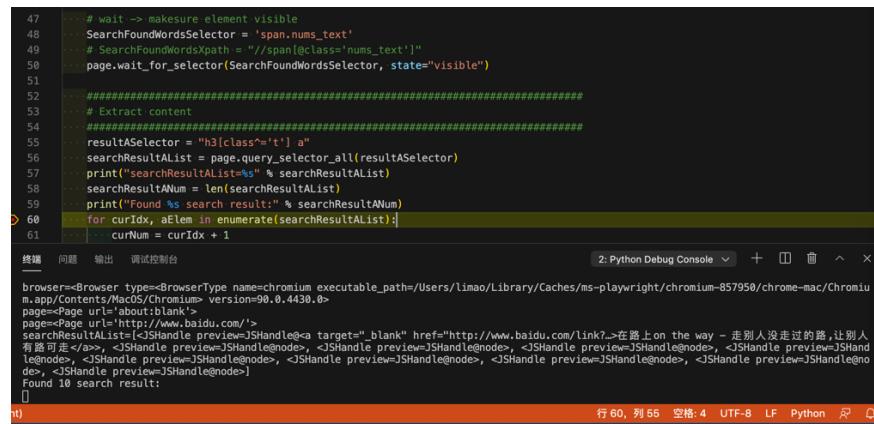
等待元素出现

用 `page` 的 `wait_for_selector`

举例：

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

效果：可以找到后续元素了



The screenshot shows a Python debugger interface with the following details:

- Code Area:** Shows a snippet of Python code related to searching for words on Baidu. It includes imports like `from playwright.sync_api import sync_playwright`, variable definitions like `browser` and `SearchFoundWordsSelector`, and a call to `page.wait_for_selector`.
- Output Area:** Displays the command-line interface of the browser. It shows the browser type (`chromium`), executable path (`/Users/limao/Library/Caches/ms-playwright/chromium-857958/chrome-mac/Chromium.app/Contents/MacOS/Chromium`), version (`90.0.4430.0`), and the URL (`http://www.baidu.com`). The output also shows the search results and the count of results found.
- Status Bar:** Shows the current file (`2: Python Debug Console`), line number (`行 60, 列 55`), and encoding (`UTF-8`).

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\(timeout\)](#)

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新：2021-05-12 09:05:10

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

The screenshot shows a terminal window on the left containing the code for `playwithDoBaiduSearch.py`. The code opens a browser, goes to Baidu, inputs 'crifan', triggers a search, and then takes a screenshot. The right side shows a browser window titled 'crifan_百度搜索' displaying search results for 'crifan'.

```
playwithDoBaiduSearch.py  
  1  #!/usr/bin/env python  
  2  # coding: utf-8  
  3  #  
  4  # Open url  
  5  #  
  6  page.goto('http://www.baidu.com')  
  7  print("page=%s" % page)  
  8  # page=<Page url='https://www.baidu.com/'>  
  9  
 10  #####  
 11  # Input text  
 12  #####  
 13  searchStr = "crifan"  
 14  SearchInputSelector = "input#kw.s_ipt"  
 15  
 16  # page.click(SearchInputSelector)  
 17  page.fill(SearchInputSelector, searchStr)  
 18  
 19  #####  
 20  # Trigger search  
 21  #####  
 22  EnterKey = "Enter"  
 23  
 24  # Method 1: press Enter key  
 25  # page.keyboard.press(EnterKey)  
 26  
 27  # Method 2: locate element then click  
 28  SearchButtonSelector = "input#su"  
 29  page.press(SearchButtonSelector, EnterKey)  
 30  
 31  page.screenshot(path='example_chromium.png')  
 32  browser.close()  
 33  
 34
```

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新：2021-05-12 09:04:25

移动端

详见专门教程：

[移动端自动化测试概览](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2021-04-29 14:06:40

Android

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2021-04-29 14:06:40

uiautomator2

详见专门教程：

[安卓自动化测试利器：uiautomator2](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2021-04-29 14:06:40

iOS

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2021-04-29 14:06:40

facebook-wda

详见专门教程：

[iOS自动化测试利器：facebook-wda](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新：2021-04-29 14:06:40

附录

下面列出相关参考资料。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,
powered by Gitbook最后更新: 2021-04-29 14:06:40

参考资料

- 【未解决】Mac中用facebook-wda自动操作安卓手机浏览器实现百度搜索
- 【已解决】puppeteer中提取百度搜索结果中的信息
- 【已解决】puppeteer中page.querySelectorAll运行时无法获取到结果
- 【规避解决】puppeteer不调试直接运行waitForSelector报错：
ElementHandleError Evaluation failed TypeError MutationObserver
is not a constructor at pollMutation
- 【已解决】Mac中初始化搭建Python版puppeteer的puppeteer的开发环境
- 【已解决】puppeteer如何给输入框中输入文字
- 【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
-
- 网络爬虫之使用puppeteer替代selenium完美绕过webdriver检测 阅读目录 - 知乎
- 爬虫神器puppeteer, 对 js 加密降维打击 - 掘金
- puppeteer(python版puppeteer)基本使用 - 白灰 - 博客园
- Selenium凭什么成为 Web 自动化测试的首选? (内附图谱) | 极客时间
- 为什么puppeteer比selenium好? - 掘金
- Selenium vs Puppeteer: testing the testing tools
- Selenium vs. Puppeteer for Test Automation: Is a New Leader Emerging? - Flood
- Selenium vs. Puppeteer - When to Choose What? | TestProject
- Puppeteer: 更友好的 Headless Chrome Node API - 知乎
- 微软开源 Python 自动化神器 Playwright - 知乎
-

crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新：2021-05-12 09:03:51