

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
Playwright	1.3.1.3
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: v1.9
- 更新时间: 20210417

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：2021-04-29 07:59:14

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: 2021-03-17 22:23:00

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：2021-04-13 21:31:20

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"
searchInputElement = chromeDriver.find_element_by_id(SearchInputId)
print("searchInputElement=%s" % searchInputElement)
# searchInputElement=<selenium.webdriver.remote.webelement.WebElement (session="e0d8b72f2fc31e27220f66fcfdb22bfc", element="21aa0e9d-04fa-4385-b16b-d82998068887")>

#####
# 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='nums_text']")))
)
print("Search complete, showing: %s" % numTextElem)

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

# Method 1: use Selenium to extract title list
searchResultAList = chromeDriver.find_elements_by_xpath("//h3[contains(@class, 't')]/a")
)
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": beginTP})
# 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 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=20)

#####
# 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 Evaluation failed: TypeEr
ror: MutationObserver is not a constructor
#   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, SingleW
aitSeconds))
    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 object at 0x10309c9b0>
    title = await aTextJSHandle.jsonValue()
    # print("type(title)=%s" % type(title))
    # type(title)=<class 'str'>
    print("title=%s" % title)

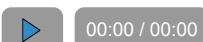
    baiduLinkUrl = await (await aElem.getProperty("href")).jsonValue()
    print("baiduLinkUrl=%s" % baiduLinkUrl)

    await browser.close()

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

```

效果：



playwright

代码：

- 文件：[playwrightDemoBaiduSearch.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 executable_path=/Users/limao/Library/Caches/ms-playwright/chromium-857950/chrome-mac/Chromium.app/Contents/MacOS/Chromium>
print("browser=%s" % browser)
# browser=<Browser type=<BrowserType name=chromium executable_path=/Users/limao/Library/Caches/ms-playwright/chromium-857950/chrome-mac/Chromium.app/Contents/MacOS/Chromium> version=90.0.4430.0>
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 "visible")

#####
# Extract content
#####
resultASelector = "h3[class^='t'] a"
searchResultAList = page.query_selector_all(resultASelector)
print("searchResultAList=%s" % searchResultAList)
# searchResultAList=[<JSHandle preview=JSHandle@<a target="_blank" href="http://www.baidu.com/link?...>在路上on the way - 走别人没走过的路,让别人有路可走</a>>, <JSHandle preview=JSHandle@node>, . . . , <JSHandle preview=JSHandle@node>]

```

```
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,让别人有路可走
    baiduLinkUrl = aElem.get_attribute("href")
    print("baiduLinkUrl=%s" % baiduLinkUrl)
    # baiduLinkUrl=http://www.baidu.com/link?url=fB3F0xZmwig9r2M_1pK4BJG00xFPLjJ85
    X39GheP_fzEA_zJIjX-IleEH_ZL8pfo

    # do sceenshot
    screenshotFilename = 'baidu_search_%s_result.png' % searchStr
    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', 'displayHeight': 2201, 'displayRotation': 0, 'displaySizeDpX': 393, 'displaySizeDpY': 873, 'displayWidth': 1080, 'productName': 'atom', 'screenOn': True, 'sdkInt': 29, 'naturalOrientation': True}

# 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(timeout=10)
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/akr")
# 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.TextView")
# # inputUiObj = d(resourceId=SearchInputId)

```

```

# print("type(inputUiObj)=%s" % type(inputUiObj)) # type(inputUiObj)=<class 'uiautomator2.
session.UiObject'>
# print("inputUiObj=%s" % inputUiObj) # inputUiObj=<uiautomator2.session.UiObject object a
t 0x10a0bea00>
# inputUiObjectInfo = inputUiObj.info
# print("type(inputUiObjectInfo)=%s" % type(inputUiObjectInfo)) # type(inputUiObjectInfo)=
<class 'dict'>
# print("inputUiObjectInfo=%s" % inputUiObjectInfo) # inputUiObjectInfo={'bounds': {'botto
m': 172, 'left': 160, 'right': 797, 'top': 107}, 'childCount': 0, 'className': 'android.wi
dget.TextView', 'contentDescription': '搜索框', 'packageName': 'com.android.browser', 'reso
urceName': 'com.android.browser:id/b4h', 'text': '', 'visibleBounds': {'bottom': 172, 'lef
t': 160, 'right': 797, 'top': 107}, 'checkable': False, 'checked': False, 'clickable': Tru
e, 'enabled': True, 'focusable': False, 'focused': False, 'longClickable': False, 'scrolla
ble': False, 'selected': False}
# 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)) # type(inputXPathSelecto
r)=<class 'uiautomator2.xpath.XPathSelector'>
# inputXPathElem = inputXPathSelector.get()
# print("type(inputXPathElem)=%s" % type(inputXPathElem)) # type(inputXPathElem)=<class 'u
iautomator2.xpath.XMLElement'>
# print("inputXPathElem=%s" % inputXPathElem) # inputXPathElem=<uiautomator2.xpath.XMLElem
ent object at 0x108585d30>
# print("type(inputXPathElem.attrib)=%s" % type(inputXPathElem.attrib)) # type(inputXPathE
lem.attrib)=<class 'lxml.etree._Attrib'>
# print("inputXPathElem.attrib=%s" % inputXPathElem.attrib) # inputXPathElem.attrib={'inde
x': '1', 'text': '', 'resource-id': 'com.android.browser:id/b4h', 'package': 'com.android.
browser', 'content-desc': '搜索框', 'checkable': 'false', 'checked': 'false', 'clickable':
'true', 'enabled': 'true', 'focusable': 'false', 'focused': 'false', 'scrollable': 'false'
, 'long-clickable': 'false', 'password': 'false', 'selected': 'false', 'visible-to-user':
'true', 'bounds': '[160,107][797,172]'}
# isFoundInput = inputXPathSelector.exists # True

# trigger into input page

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

# # Method 2
# inputXPathSelector = d.xpath("//android.widget.TextView[@resource-id=%s]" % SearchInputI
d)
# 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").exists(timeout=10)
d(text="百度一下,你就知道", className="android.view.View").exists(timeout=10)
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 using following code to refresh to
# get latest page source xml
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.Button[@clickable='true']").all()
# titleButtonSelectorList = resultsSelector.xpath("./android.widget.Button[@clickable='true']").all()

# Xpath chain search can find elements
titleButtonElementList = d.xpath("//*[@resource-id='results']/android.widget.Button[@clickable='true']").all()
titleButtonNum = len(titleButtonElementList)
print("Found %s search result title" % titleButtonNum)

# descriptionElementList = d.xpath("//*[@resource-id='results']/android.view.View[1]/android.view.View[1]/android.view.View[2]/android.view.View[1]/android.view.View[1]/android.view.View[1]/android.view.View[1]/android.view.View[1]").all()
descriptionElementList = d.xpath("//*[@resource-id='results']/android.view.View/android.view.View[1]/android.view.View[2]/android.view.View[1]/android.view.View[1]/android.view.View[1]/android.view.View[1]/android.view.View[1]").all()
descriptionNum = len(descriptionElementList)
print("Found %s description" % descriptionNum)

# # sourceWebsiteElementList = d.xpath('//*[@resource-id="results"]/android.view.View/android.view.View[1]/android.view.View[2]/android.view.View[1]').all()
# sourceWebsiteElementList = d.xpath('//*[@resource-id="results"]/android.view.View/android.view.View[1]/android.view.View[2]/android.view.View[2]').all()
# 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, "-"*20))
    # 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 07:58:05

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。

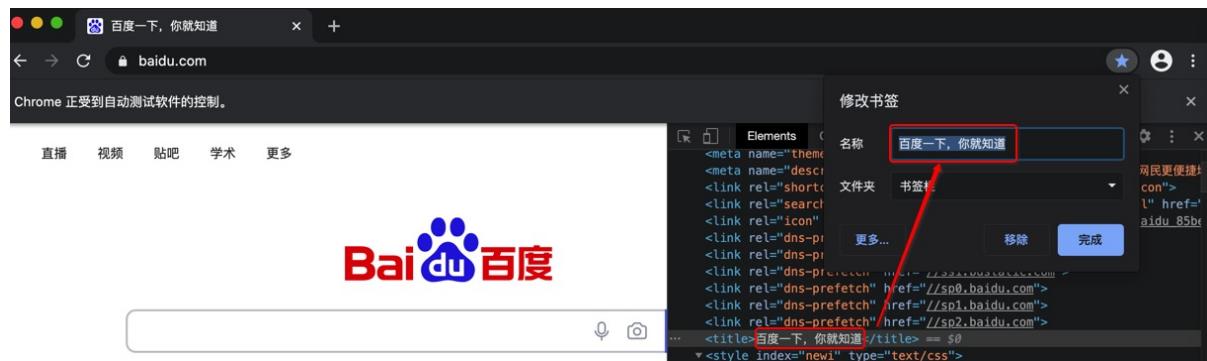
而调试页面元素的最常用办法就是： `Chrome` 的 开发者工具

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`

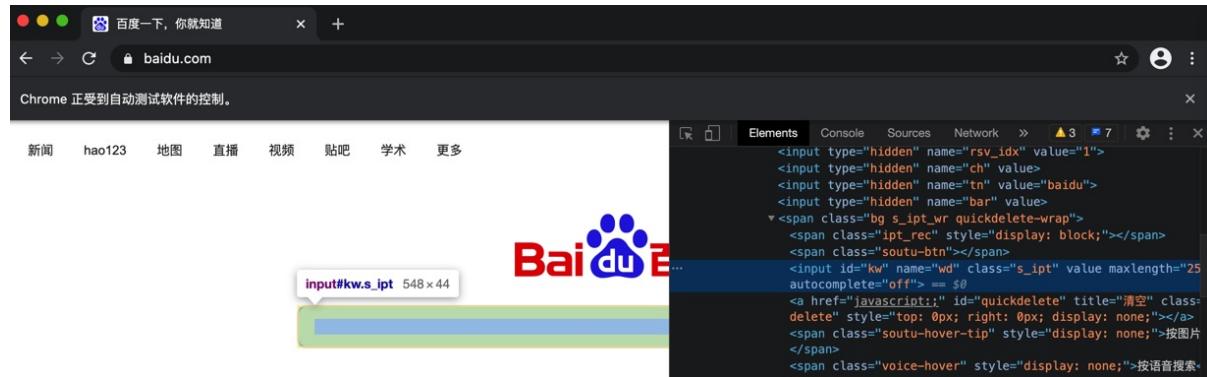
找到输入框对应的元素

可以右键 输入框 检查

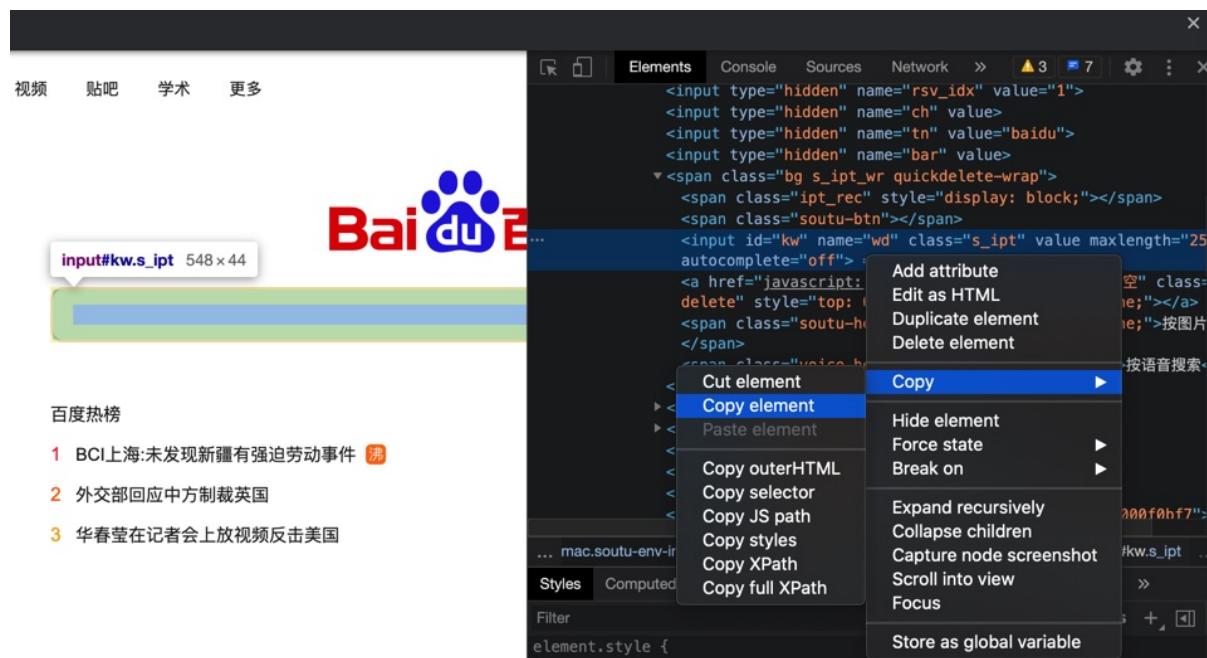
视频 贴吧 学术 更多



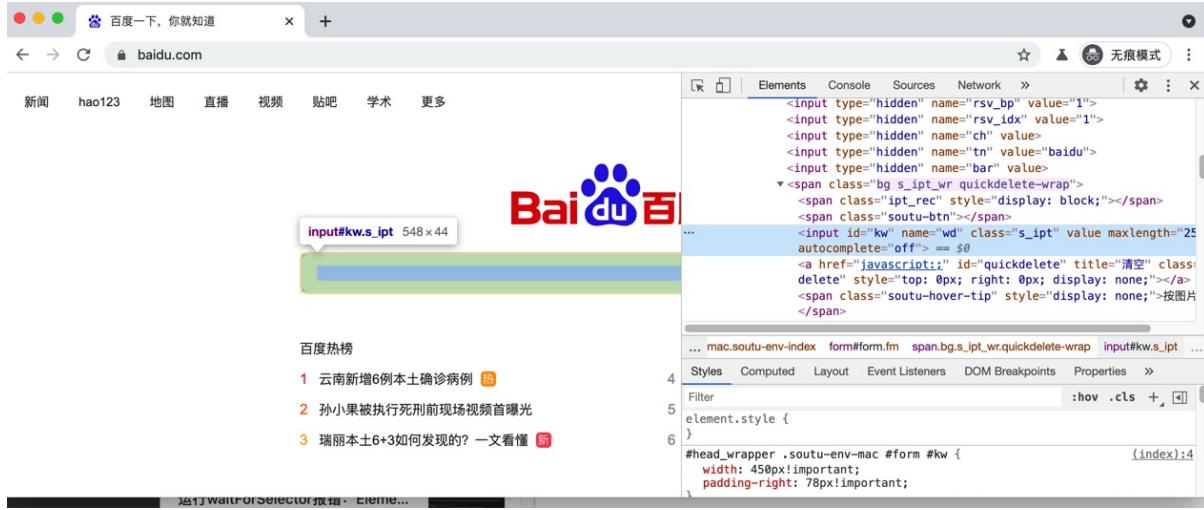
打开 Chrome 的开发者工具



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" autocomplete="off">
```

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：

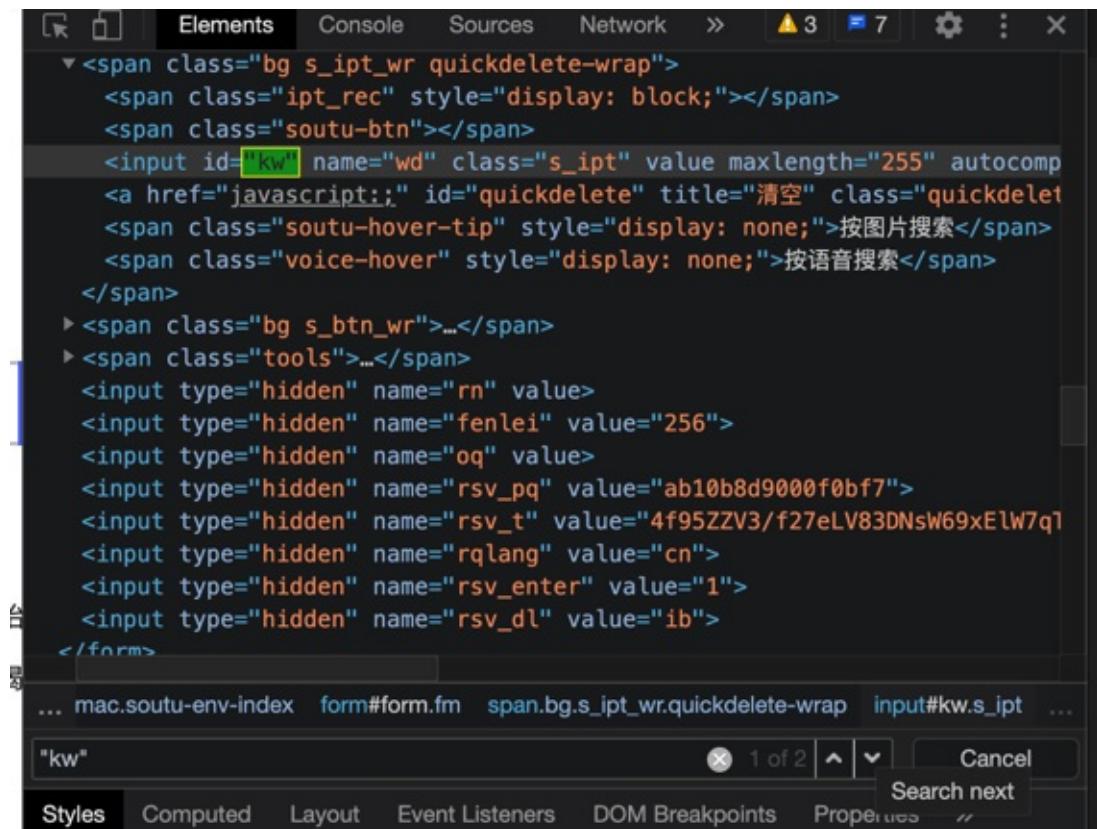
- `input` : 元素名 tag
- `kw` : 是 id
- `s_ipt` : 是 class

→ 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

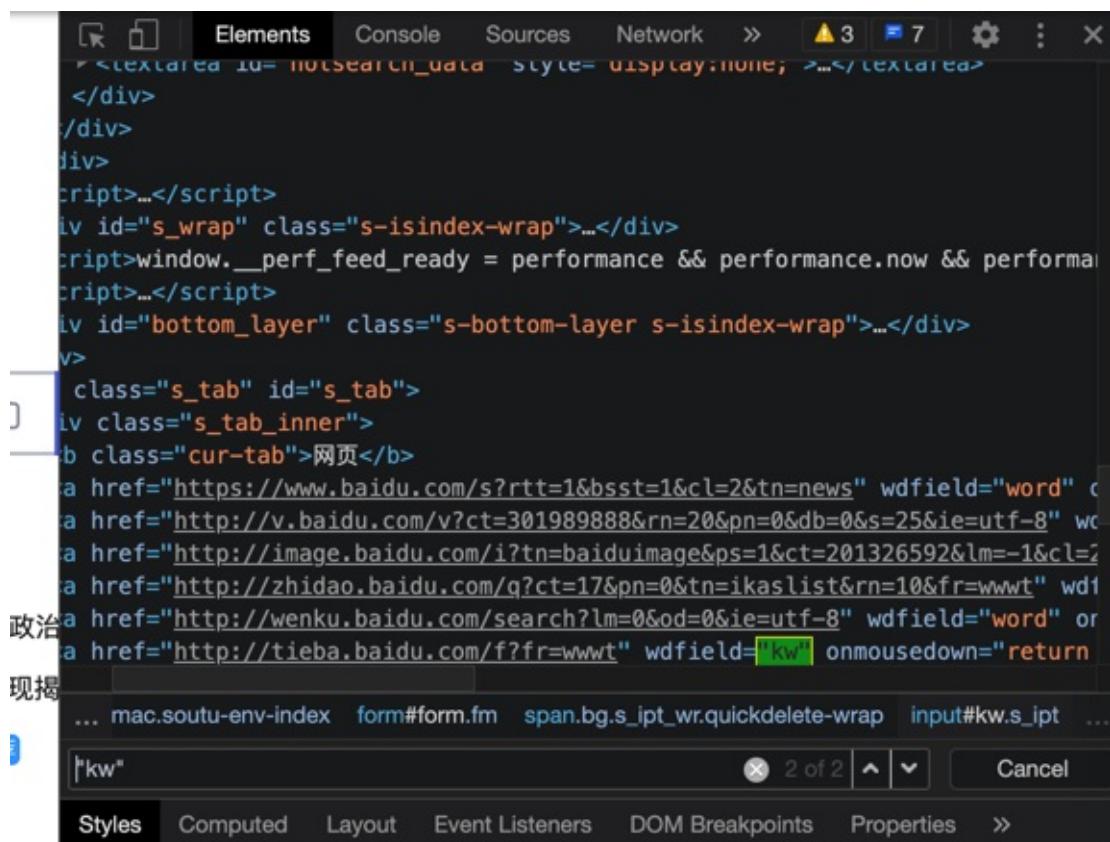
确认id是否唯一

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

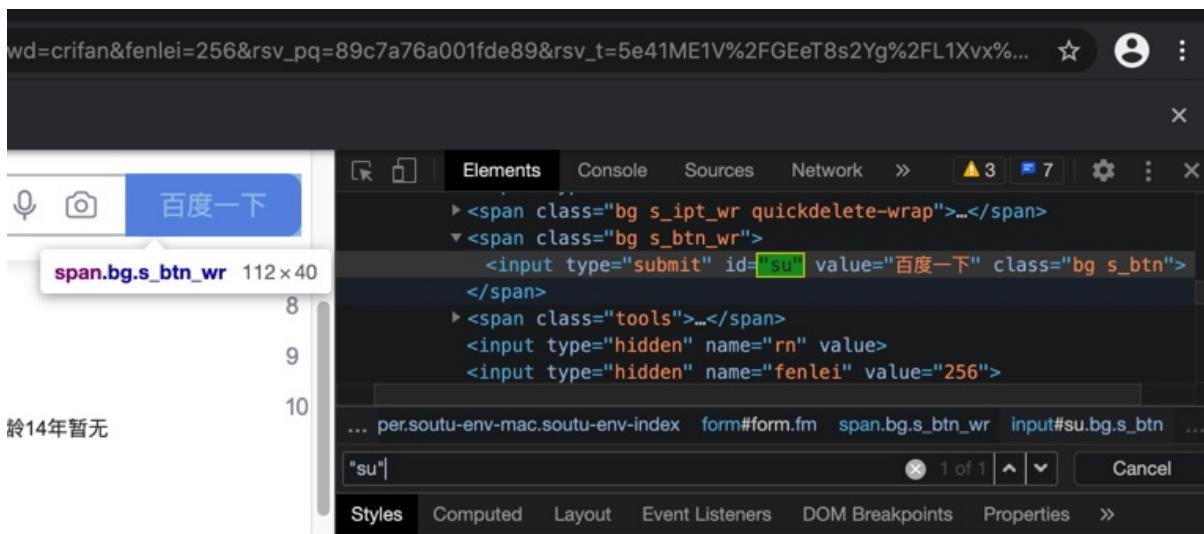
找 百度一下 按钮，和之前类似，去 右键检查：



可以看到html是：

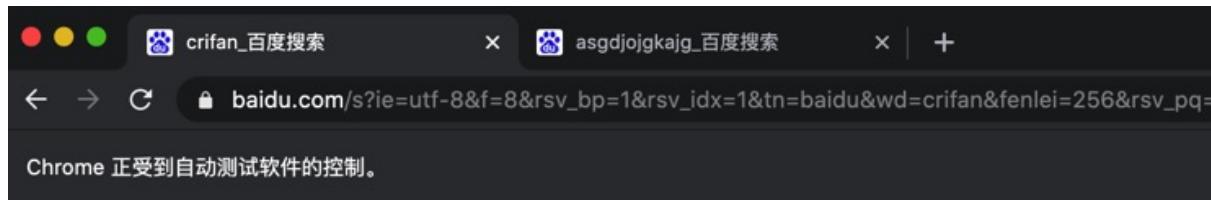
```
<input type="submit" id="su" value="百度一下" class="bg s_btn">
```

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
```



且去确认了，故意搜不到内容，页面也会出现这个：



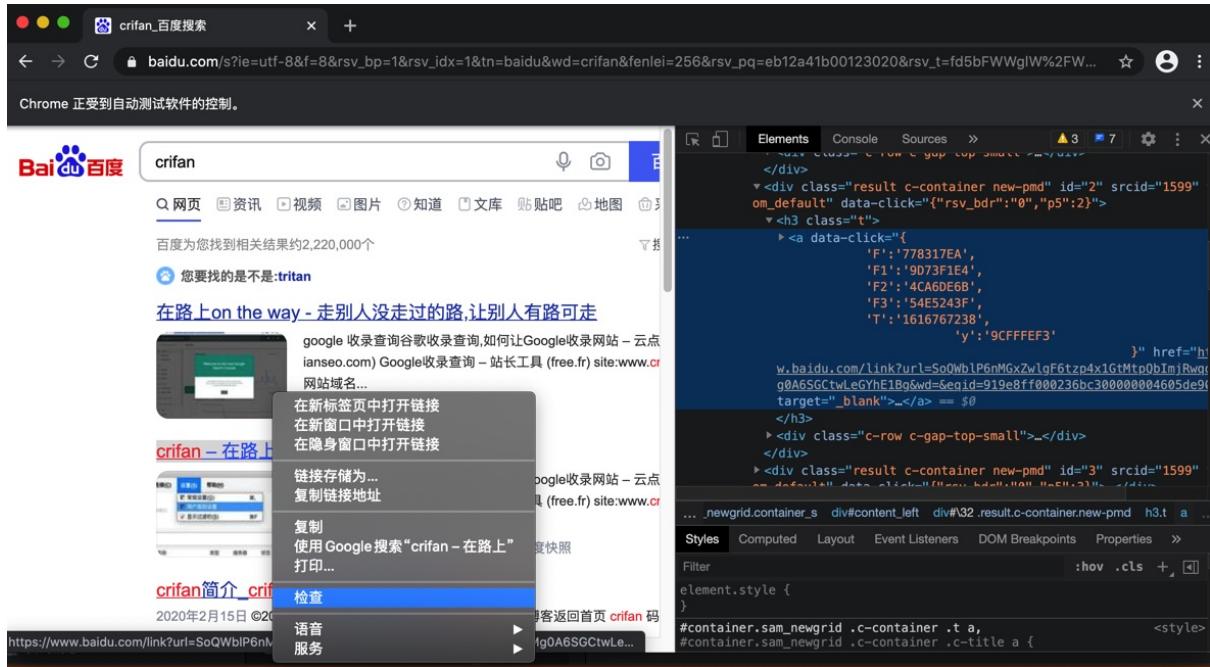
去看看其html：



百度搜索的每条结果的html

去搞清楚，本身此外的每条搜索结果的内容的html是什么

右键 检查:

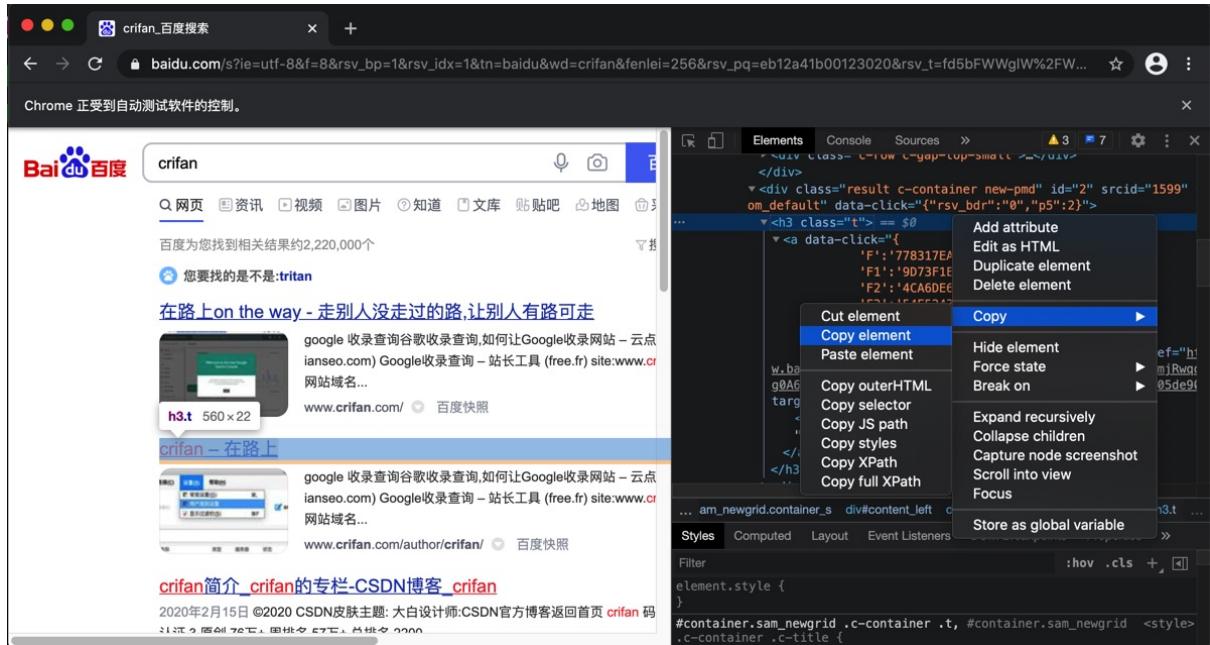


找到是：

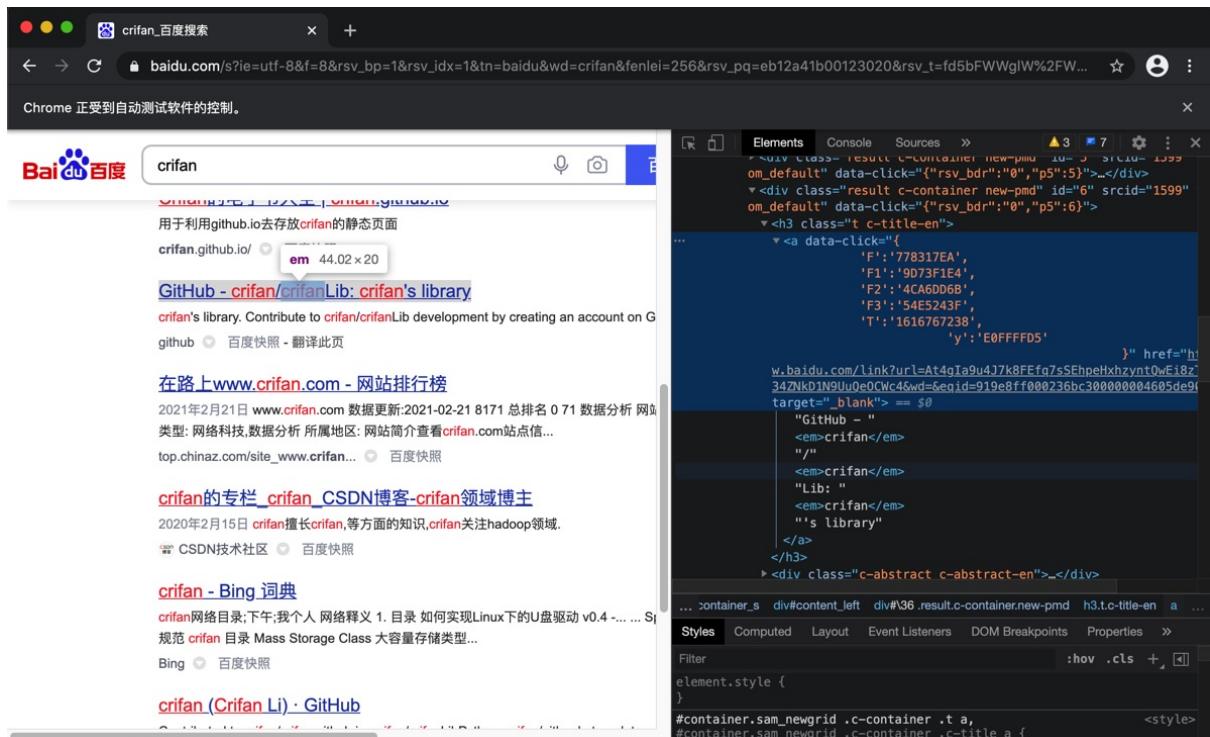
```
<a data-click="{
    'F': '778317EA',
    'F1': '9D73F1E4',
    'F2': '4CA6DE6B',
    'F3': '54E5243F',
    'T': '1616767238',
    'y': '9CFFFFE3'
}" href="https://www.baidu.com/link?url=SoQWb1P6nMGxZwlgF6tzp4x1GtMtpQbImjRwqok-NChASMg0A6SGCtwLeGYhE1Bg&wd=&eqid=919e8ff000236bc30000004605de906"
target="_blank"><em>crifan</em> - 在路上</a>
```

上层父节点的元素是：

```
<h3 class="t"><a data-click="{
    'F': '778317EA',
    'F1': '9D73F1E4',
    'F2': '4CA6DE6B',
    'F3': '54E5243F',
    'T': '1616767238',
    'y': '9CFFFFE3'
}" href="https://www.baidu.com/link?url=SoQWb1P6nMGxZwlgF6tzp4x1GtMtpQbImjRwqok-NChASMg0A6SGCtwLeGYhE1Bg&wd=&eqid=919e8ff000236bc30000004605de906"
target="_blank"><em>crifan</em> - 在路上</a></h3>
```



多看看几个结果，是否都是同样格式：



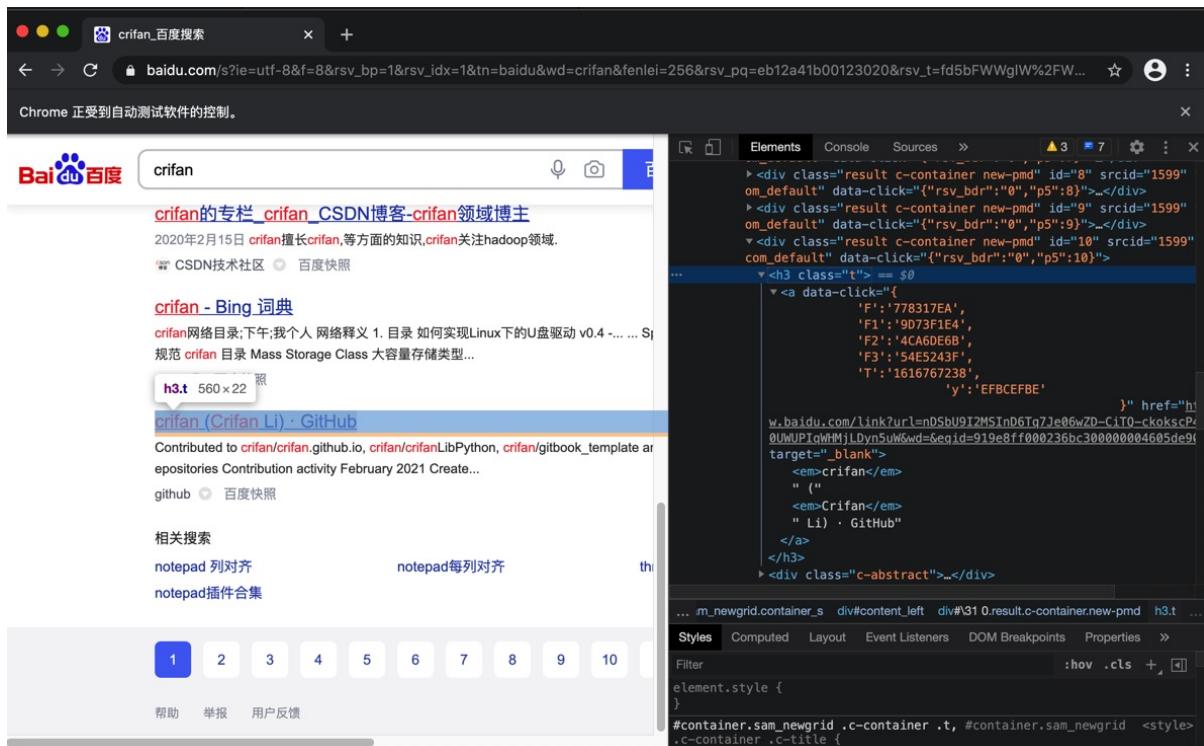
这个稍微复杂点：

```
<h3 class="t c-title-en"><a data-click="{
  'F': '778317EA',
  'F1': '9D73F1E4',
  'F2': '4CA6DD6B',
  'F3': '54E5243F',
  'T': '1616767238',
  'y': 'E0FFFFD5'
}" href="https://www.baidu.com/link?url=At4gTa9u4J7k8FEfa7sSEhpHkhzynt0wE18z342NkD1N9Uu0eOCwc4&wd=&eqid=919e8ff000236bc30000000450d5de90
target=_blank"> $0
  "GitHub - "
  <em>crifan</em>
  "/"
  <em>crifan</em>
  "Lib: "
  <em>crifan</em>
  "'s library"
</a>
</h3>
<div class="c-abstract c-abstract-en"></div>
```

```
4gIa9u4J7k8FEfq7sSEhpeHxhyntQwEi8zTkeWpSbv_34ZNkD1N9UuQeOCWc4&wd=&eqid=919e8ff000236bc300
000004605de906" target="_blank">GitHub - <em>crifan</em>/<em>crifan</em>Lib: <em>crifan</em>
's library</a></h3>
```

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="https://www.baidu.com/link?url=nD
SbU9I2MSInD6Tq7Je06wZD-CiTQ-ckokscP4kiXneJc50UWUPIqWHMjLDyn5uW&wd=&eqid=919e8ff000236bc300
000004605de906" target="_blank"><em>crifan</em> (<em>Crifan</em> Li) · GitHub</a></h3>
```

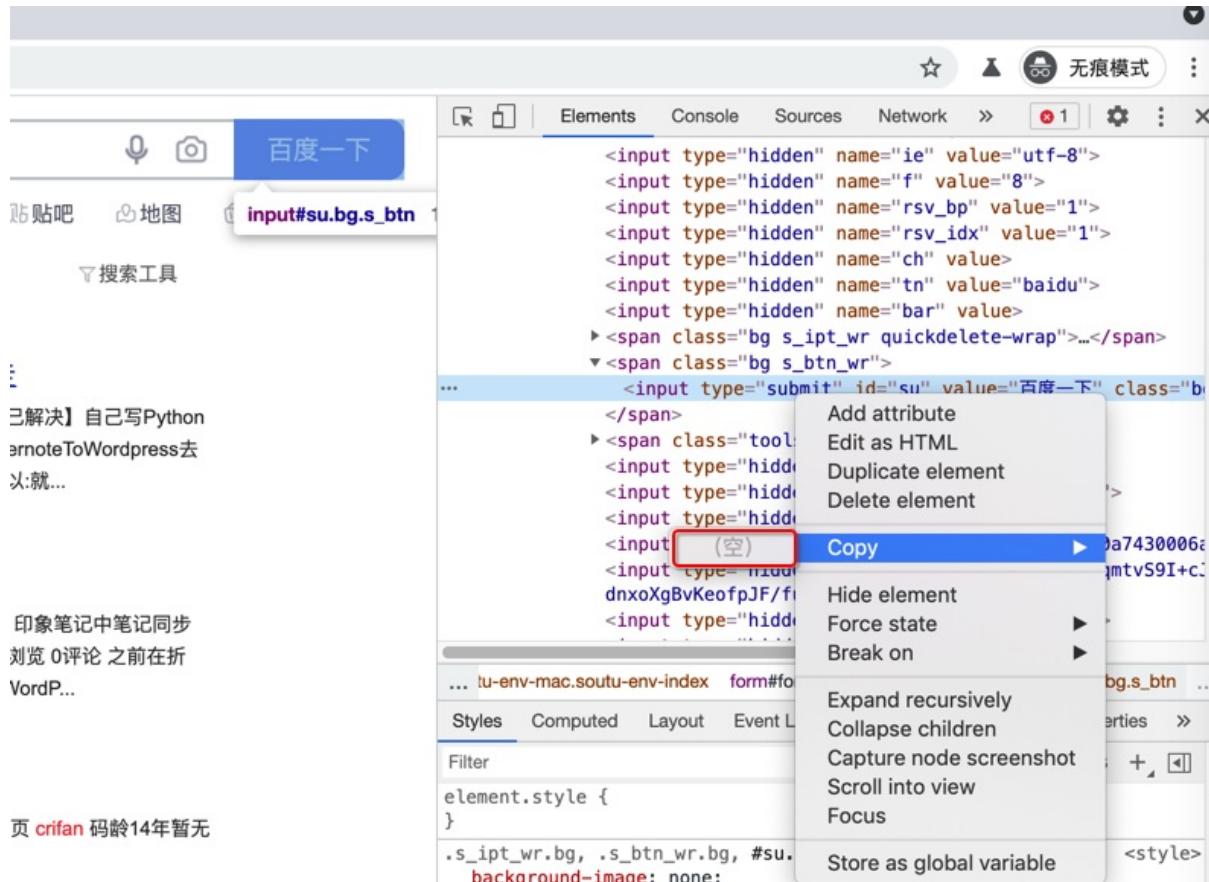


常见问题

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: 2021-04-17 11:39:04

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：2021-03-17 21:27:55

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： 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/puppeteer: Headless chrome/chromium automation library \(unofficial port of puppeteer\)](#)
 - 注: 代码已归档, 变只读了
 - pypi
 - [puppeteer · 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: 2021-03-18 21:11:14

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：2021-03-17 21:26:39

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

puppeteer资料

- github官网
 - [puppeteer/puppeteer: Headless Chrome Node.js API](#)
- google中
 - [Puppeteer | Tools for Web Developers | Google Developers](#)
- 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: 2021-03-17 22:16:21

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：2021-03-17 22:19:31

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：2021-03-17 22:20:48

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: 2021-03-17 21:20:13

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：2021-03-17 22:21:13

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: 2021-03-17 21:20:29

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：2021-03-17 22:21:31

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: 2019-06-25 22:00:32

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
-
- 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: 2021-04-13 21:40:04