

目录

前言	1.1
Playwright概览	1.2
初始化环境	1.3
基本操作	1.4
查找元素	1.4.1
查找并点击元素	1.4.2
输入文字	1.4.3
等待元素出现	1.4.4
模拟按键	1.4.5
获取元素属性值	1.4.6
举例	1.5
百度搜索自动化	1.5.1
附录	1.6
教程和资料	1.6.1
参考资料	1.6.2

跨平台Web自动化神器： Playwright

- 最新版本： v1.0
- 更新时间： 20210717

简介

介绍支持多个平台的Web自动化神器， playwright。

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。如有发现侵犯您的版权，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： 2021-07-17 17:40:18

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。

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： 2021-07-02 22:22:37

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）

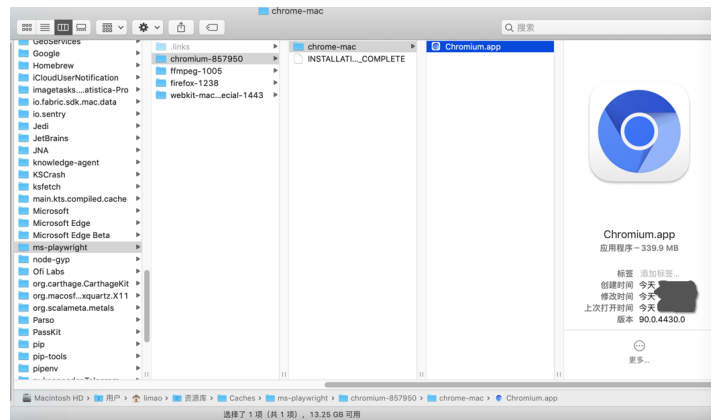
详细日志

```
❏ playwright install
Downloading chromium v857950 - 113.9 Mb [=====]
chromium v857950 downloaded to /Users/limao/Library/Caches/
Downloading firefox v1238 - 75 Mb [=====] 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：

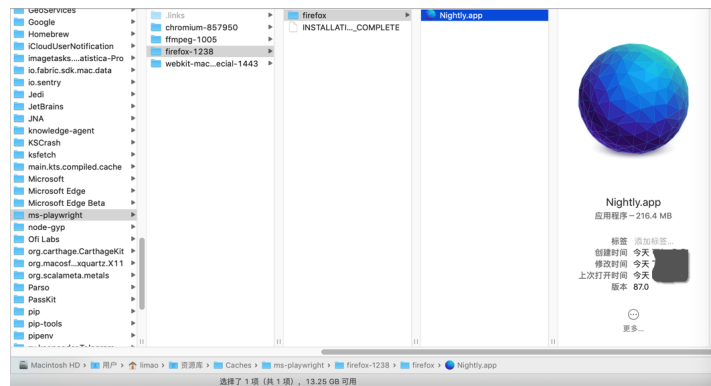
- chromium
 - 位置： /Users/limao/Library/Caches/ms-playwright/chromium-857950
 - 效果：

查找元素



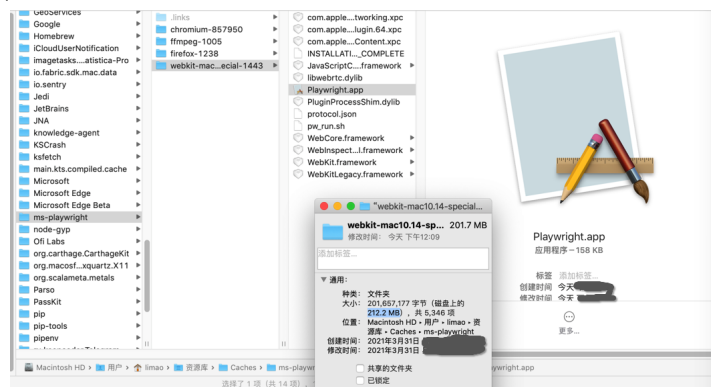
- firefox

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



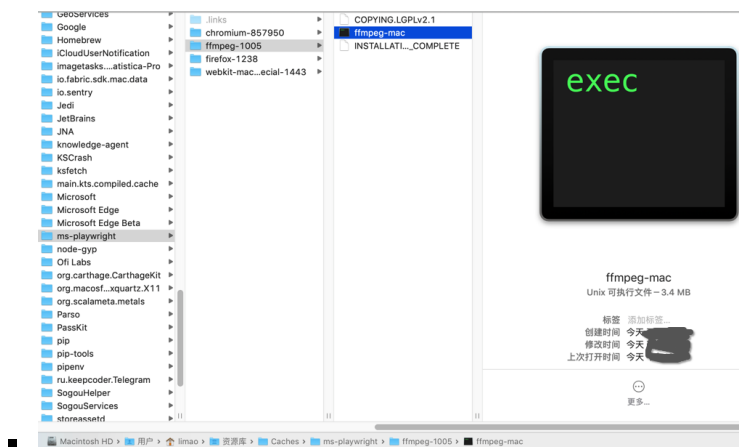
- webkit

- 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

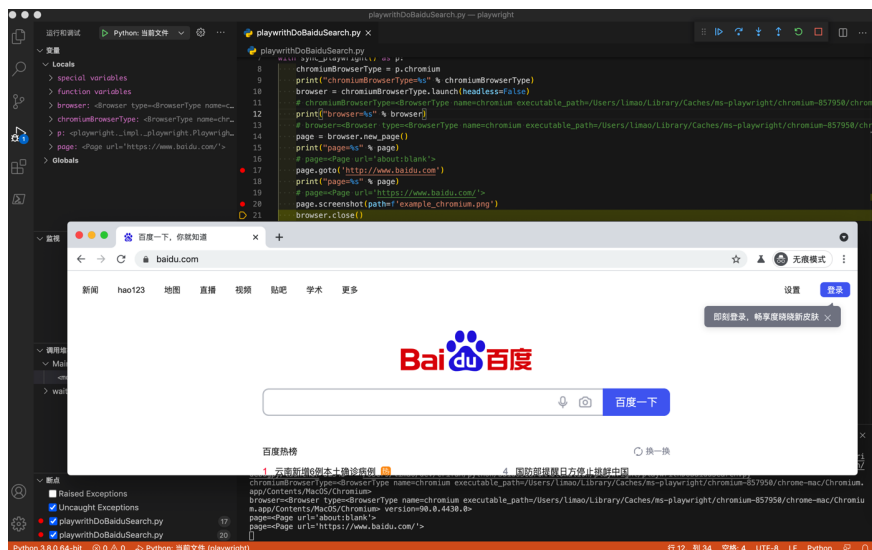
with sync_playwright() as p:
    chromiumBrowserType = p.chromium
    print("chromiumBrowserType=%s" % chromiumBrowserType)
    browser = chromiumBrowserType.launch(headless=False)
    # chromiumBrowserType=<BrowserType name=chromium executable_path=/Applications/Chromium.app/Contents/MacOS/Chromium>
    print("browser=%s" % browser)
    # browser=<Browser type=<BrowserType name=chromium executable_path=/Applications/Chromium.app/Contents/MacOS/Chromium>
    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_path=/Applications/Chromium.app/Contents/MacOS/Chromium>
browser=<Browser type=<BrowserType name=chromium executable_path=/Applications/Chromium.app/Contents/MacOS/Chromium>
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 number
  -h, --help                   display help for command

Commands:
  open [options] [url]         open page in browser
  codegen [options] [url]      open page and generate code
  debug <app> [args...]        run command in debug mode
  install [browserType...]     ensure browsers are installed
  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: 2021-07-17 17:32:24

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: 2021-07-02 21:42:22

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

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

输出：

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

```

47 #####
48 # Extract content
49 #####
50 resultASelector = "h3[class^='t'] a"
51 searchResultAList = page.query_selector_all(resultASelector)
52 print("searchResultAList=%s" % searchResultAList)
53 searchResultANum = len(searchResultAList)
54 print("Found %s search result:" % searchResultANum)
55 for curIdx, aElem in enumerate(searchResultAList):
56     curNum = curIdx + 1
57     print("%s [%d] %s" % ("-"*20, curNum, aElem.get_property('title')))
58     aTextJSHandle = aElem.get_property('text')
59     # print("aTextJSHandle=%s" % aTextJSHandle)
60     # type(aTextJSHandle)=<class 'JSHandle'>
61     # print("aTextJSHandle=%s" % aTextJSHandle)
62     # aTextJSHandle=ppp.pppeteer.JSHandle
63     title = aTextJSHandle.json_value('title')
64     # print("type(title)=%s" % type(title))
65     # type(title)=<class 'str'>
66     print("title=%s" % title)
67     len(): 10
68     hrefLinkUrl = aElem.get_property('href')

```

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： 2021-07-02 21:40:41

查找并点击元素

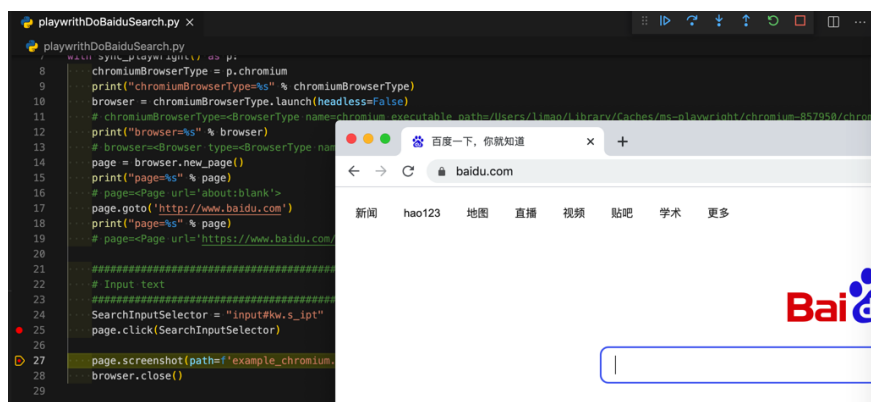
对于百度主页搜索输入框，html是：

```
<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：



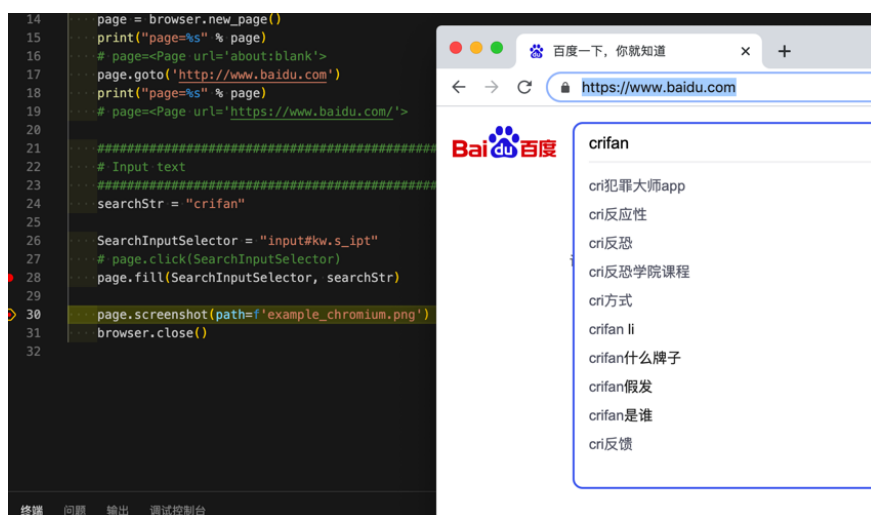
crifan.com，使用署名4.0国际(CC BY 4.0)协议发布 all right reserved,
powered by Gitbook最后更新： 2021-07-02 21:38:51

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>

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：2021-07-02 21:35:36

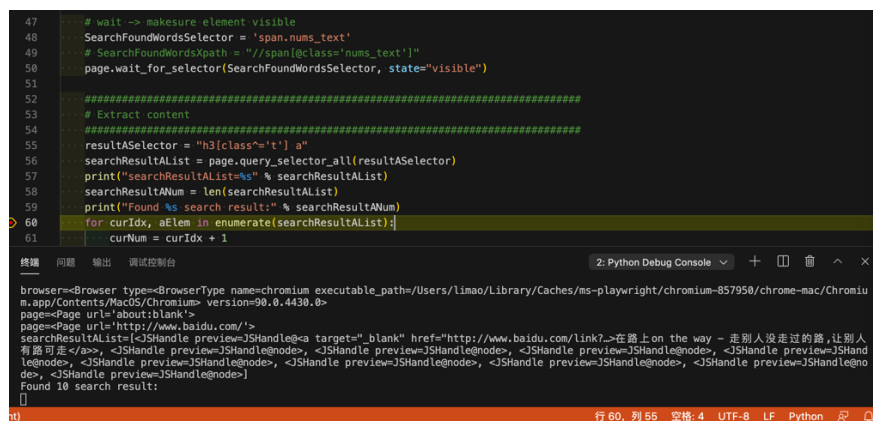
等待元素出现

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 script in a code editor and its execution output in a Python Debug Console. The script defines a selector 'span.nums_text' and uses 'page.wait_for_selector' to wait for it. It then extracts content from the page and prints the search results. The console output shows the browser type, page URL, and the search results found on the page.

```
47 # wait -> makesure element visible  
48 SearchFoundWordsSelector = 'span.nums_text'  
49 # SearchFoundWordsXPath = "//span[@class='nums_text']"  
50 page.wait_for_selector(SearchFoundWordsSelector, state="visible")  
51  
52 #####  
53 # Extract content  
54 #####  
55 resultSelector = "h3[class='t'] a"  
56 searchResultAList = page.query_selector_all(resultSelector)  
57 print("searchResultAList=%s" % searchResultAList)  
58 searchResultANum = len(searchResultAList)  
59 print("Found %s search result:" % searchResultANum)  
60 for curIdx, aElem in enumerate(searchResultAList):  
61     curNum = curIdx + 1
```

终端 问题 输出 调试控制台 2: Python Debug Console

```
browser=<Browser type=Chromium name=chromium executable_path=/Users/limao/Library/Caches/ms-playwright/chromium-857950/chrome-mac/Chromiu  
m.app/Contents/MacOS/Chromium> version=90.0.4430.0>  
page=<Page url='about:blank'>  
page=<Page url='http://www.baidu.com/'>  
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>, <JSHandle preview=JSHandle@node>, <JSHandle preview=JSHand  
le@node>, <JSHandle preview=JSHandle@node>]  
Found 10 search result:  
[]
```

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： 2021-07-02 21:41:51

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

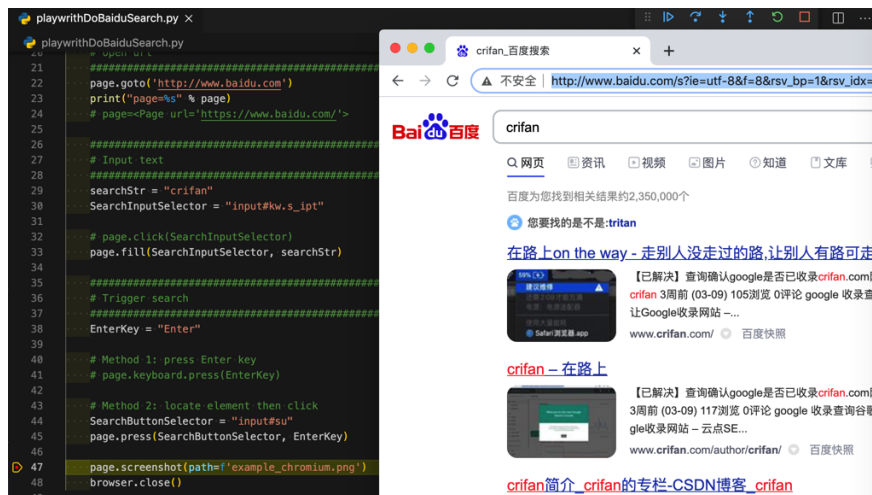
```
EnterKey = "Enter"
# Method 1: press Enter key
page.keyboard.press(EnterKey)
```

- 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



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： 2021-07-02 21:37:59

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

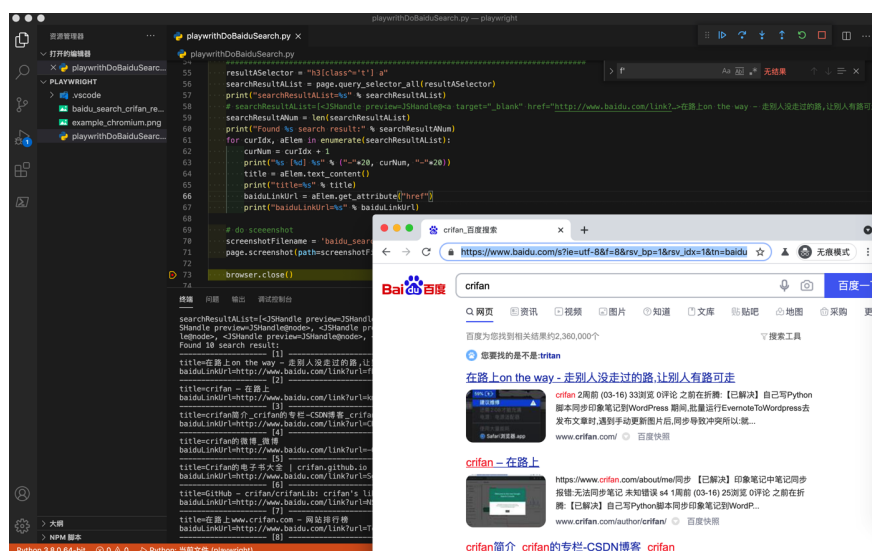
从百度搜索后的结果，解析提取每个结果的标题和链接的代码如下：

```
#####
# Extract content
#####
resultASelector = "h3[class^='t'] a"
searchResultAList = page.query_selector_all(resultASelector)
print("searchResultAList=%s" % searchResultAList)
# searchResultAList=[<JSHandle preview=JSHandle@<a target=
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=kmvgD1PraouLnnjl
----- [3] -----
title=crifan简介_crifan的专栏-CSDN博客_crifan
baiduLinkUrl=http://www.baidu.com/link?url=CHLWAQKOMgb23Gm:
----- [4] -----
title=crifan的微博_微博
baiduLinkUrl=http://www.baidu.com/link?url=-QwLZ5SEmZD1R2Qc
----- [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=NSZ5IzQ2Qag3CpGI
----- [7] -----
title=在路上www.crifan.com - 网站排行榜
baiduLinkUrl=http://www.baidu.com/link?url=Tc4cbETNkpQXj-k)
----- [8] -----
title=crifan的专栏_crifan_CSDN博客-crifan领域博主
baiduLinkUrl=http://www.baidu.com/link?url=0LkrWu8q9SRZuBN-
----- [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-3hYeLAQ8T4efC
```

效果：



查找元素

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: 2021-07-02 21:39:20

举例

下面给出具体的playwright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: 2021-07-02 21:38:08

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 execu
    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_ip"

# 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)

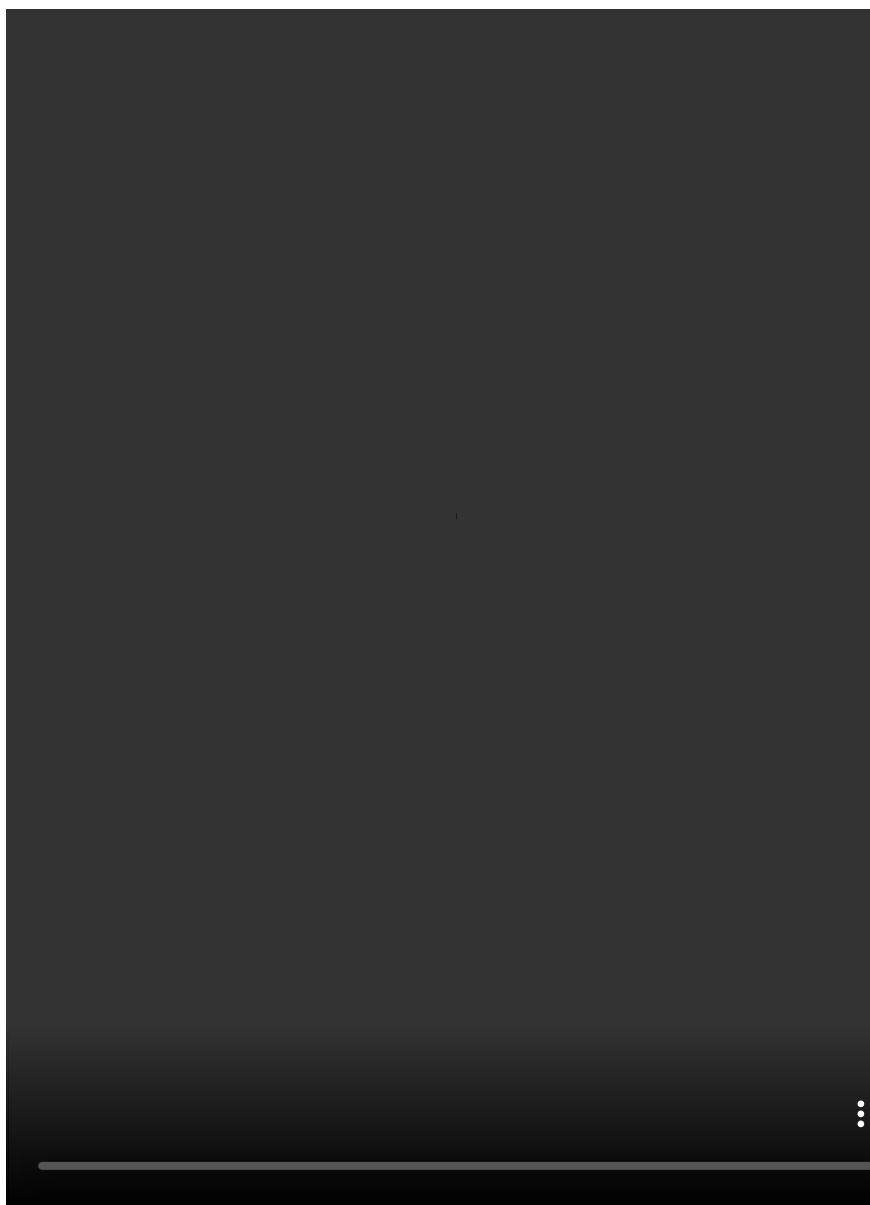
# wait -> makesure element visible
SearchFoundWordsSelector = 'span.nums_text'
# SearchFoundWordsXpath = "//span[@class='nums_text']"
page.wait_for_selector(SearchFoundWordsSelector, state=

```

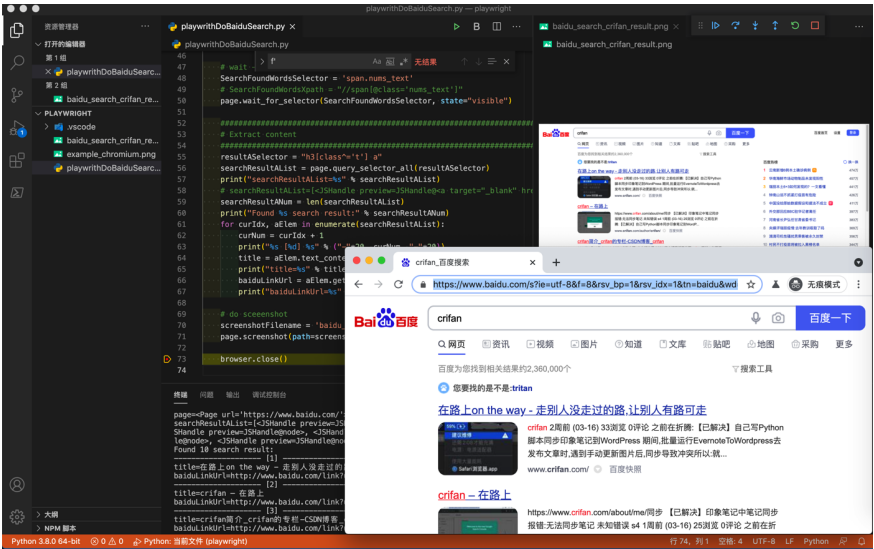
```
#####  
# Extract content  
#####  
resultASelector = "h3[class^='t'] a"  
searchResultAList = page.query_selector_all(resultASelector)  
print("searchResultAList=%s" % searchResultAList)  
# searchResultAList=[<JSHandle preview=JSHandle@<a target=blank href=#####>  
searchResultANum = len(searchResultAList)  
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=fB3F0x#####  
  
# do screenshot  
screenshotFilename = 'baidu_search_%s_result.png' % searchResultANum  
page.screenshot(path=screenshotFilename)  
  
browser.close()
```

效果

视频



图



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/'>
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=kmvgD1PraoULnnjl
----- [3] -----
title=crifan简介_crifan的专栏-CSDN博客_crifan
baiduLinkUrl=http://www.baidu.com/link?url=CHLWAQK0Mgb23Gm:
----- [4] -----
title=crifan的微博_微博
baiduLinkUrl=http://www.baidu.com/link?url=-QwLZ5SEmZD1R2Qc
----- [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=NSZ5IzQ2Qag3CpGI
----- [7] -----
title=在路上www.crifan.com - 网站排行榜
baiduLinkUrl=http://www.baidu.com/link?url=Tc4cbETNKpQXj-k)
----- [8] -----
title=crifan的专栏_crifan_CSDN博客-crifan领域博主
baiduLinkUrl=http://www.baidu.com/link?url=0LkrWu8q9SRZuBN-
----- [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(

```

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: 2021-07-02 21:41:48

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: 2021-07-02 21:39:19

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>

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: 2021-07-02 21:41:28

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
-
- [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\(name\)](#)
-

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: 2021-07-02 21:39:32