

# 目录

前言	1.1
初始化	1.2
基本操作	1.3
查找定位元素	1.3.1
输入文字	1.3.2
点击元素	1.3.3
触发搜索	1.3.3.1
获取元素属性	1.3.4
心得和总结	1.4
代码运行时却找不到元素	1.4.1
select不能用于ul	1.4.2
偶尔Chrome右键元素不是你要的	1.4.3
偶尔刷新后才能找到元素	1.4.4
举例	1.5
模拟百度搜索	1.5.1
附录	1.6
文档和教程	1.6.1
参考资料	1.6.2

# Selenium知识总结

- 最新版本: v2.0
- 更新时间: 20210429

## 简介

介绍PC端Web自动化最主流工具Selenium。先介绍了如何初始化Selenium开发环境，再介绍基本操作，包括查找元素、输入文字、点击元素、获取元素属性等；以及总结一些开发经验，包括如何等待元素出现、select不能用于ul、偶尔Chrome右键找到的元素不是你要的，偶尔刷新后才能找到元素等；给出具体实例，比如模拟百度搜索并解析搜索结果；给出相关文档和资料。

## 源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

### Gitbook源码

- [crifan/selenium\\_summary: Selenium知识总结](#)

### 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

- [Selenium知识总结 book.crifan.com](#)
- [Selenium知识总结 crifan.github.io](#)

### 离线下载阅读

- [Selenium知识总结 PDF](#)
- [Selenium知识总结 ePUB](#)
- [Selenium知识总结 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。其中部分內容参考自网络，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 21:23:18

# 初始化

核心逻辑：

- Python中安装Selenium库
  - pip install selenium
- 再安装webdriver
  - 比如： Chrome 的 driver : chromedriver
    - 需要下载到二进制的chromedriver，并确保PATH中能找到

## 安装 selenium

```
pip3 install selenium
```

附上完整log

```
□ pip3 install selenium
Looking in indexes: http://mirrors.aliyun.com/pypi/simple/
Collecting selenium
  Downloading http://mirrors.aliyun.com/pypi/packages/80/d6/selenium-3.141.0-py3-none-any.whl (904 kB)
[██████████] 904 kB 1.2 MB/s
Requirement already satisfied: urllib3 in /Users/crifan/.pyenv/versions/3.7.3/lib/python3.7/site-packages (from selenium)
Installing collected packages: selenium
Successfully installed selenium-3.141.0
```

## 安装driver

Selenium 的运行依赖于具体的浏览器（的内核），此处叫做： driver

- Chrome 的 driver : WebDriver = chromedriver

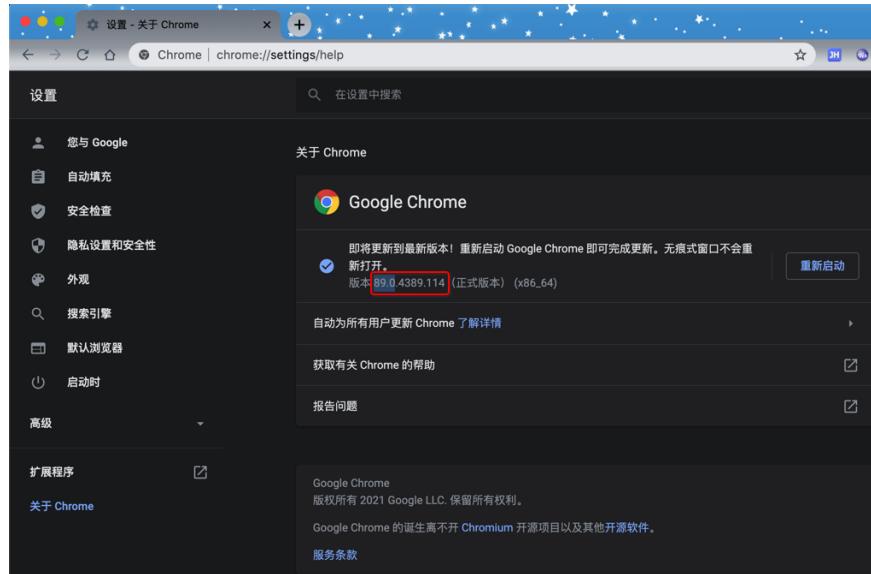
## 安装Chrome的driver： WebDriver

下载 WebDriver

要下载和你的Chrome版本一致的WebDriver

此处查看到Chrome的版本是： 89.0

## 触发搜索



所以要下载的 Chromedriver 也是要于此版本一致的，即下载 Chromedriver 89.0 的版本

- 下载源1：Chrome官网
  - [Chromedriver - WebDriver for Chrome](#)
    - 此时最新版是：Chromedriver 89.0.4389.23

A screenshot of the Chromedriver download page at chromedriver.chromium.org. The page title is 'Chromedriver - WebDriver for Chrome'. On the left, there's a sidebar with links like 'CHROMEDRIVER', 'CAPABILITIES &amp; CHROMEOPTIONS', 'CHROME EXTENSIONS', 'CHROMEDRIVER CANARY', 'CONTRIBUTING', 'DOWNLOADS', 'VERSION SELECTION', 'GETTING STARTED', 'ANDROID', 'CHROMEOS', 'LOGGING', 'PERFORMANCE LOG', 'MOBILE EMULATION', and 'NEED HELP?'. The main content area has a heading 'Chromedriver' and a paragraph about WebDriver. Below that, it says 'All versions available in Downloads' and lists 'Latest stable release: Chromedriver 89.0.4389.23' and 'Latest beta release: Chromedriver 90.0.4430.24'. At the bottom, there's a 'Chromedriver Documentation' section with links to 'Getting started with Chromedriver on Desktop (Windows, Mac, Linux)', 'Chromedriver with Android', 'Chromedriver with ChromeOS', 'ChromeOptions, the capabilities of Chromedriver', 'Mobile emulation', 'Security Considerations', and 'Chrome Extension installation'.

- [Index of /89.0.4389.23/](#)
- [chromedriver\\_mac64.zip](#)

The screenshot shows a table listing files from the 'Index of /89.0.4389.23/' page. The columns are 'Name', 'Last modified', 'Size', and 'ETag'. The files listed are:

Name	Last modified	Size	ETag
Parent Directory			
<a href="#">chromedriver_linux64.zip</a>	2021-01-28 17:30:52	5.57MB	24686a3cc3ccf8cbc60cf744baa47692
<a href="#">chromedriver_mac64.zip</a>	2021-01-28 17:30:53	7.97MB	a6620cd6a6804fa08365dfc6e8c8724e6
<a href="#">chromedriver_mac64_m1.zip</a>	2021-01-28 17:30:55	7.17MB	1544d2a1b1aefbd55f5f5dac0b48d89f
<a href="#">chromedriver_win32.zip</a>	2021-01-28 17:30:57	5.68MB	0b4fb39f34cee67f5f95af8a24c191
<a href="#">notes.txt</a>	2021-01-28 17:31:00	0.00MB	4e3ff6354a7462edfe5d95ab8690f7c8

- 下载源2：淘宝的npm源
  - <http://npm.taobao.org/mirrors/chromedriver/>
  - ->

- <http://npm.taobao.org/mirrors/chromedriver/89.0.4389.23/>
- ->
- [http://npm.taobao.org/mirrors/chromedriver/89.0.4389.23/chromedriver\\_mac64.zip](http://npm.taobao.org/mirrors/chromedriver/89.0.4389.23/chromedriver_mac64.zip)

## 确保命令行中能调用到chromedriver

想要让命令行中，可以调用到 chromedriver，即：  
把 chromedriver 放到环境变量 PATH 中：

下载后解压得到二进制的： chromedriver

把 chromedriver 放到 PATH 中

- 方式1：移动到系统相关目录

```
sudo mv /xxx/chromedriver /usr/local/bin
```

- 方式2：放到某个路径下，把该路径加到PATH中

- 此处放到

了： /Users/crifan/dev/dev\_tool/selenium/chromedriver

- 把路径加到PATH中

- 编辑启动脚本：

```
vi ~/.zshrc
```

- 在文件最后加

上： PATH=\$PATH:/Users/crifan/dev/dev\_tool/selenium

- 使其立刻生效：

```
source ~/.zshrc
```

然后去确认命令行中能找到：

```
which chromedriver
```

确保能输出对应了路径，表示找到了。

顺带也可以去：看看版本：

```
chromedriver --version
```

此处输出是： ChromeDriver 89.0.4389.23

## 写测试代码，确认环境正常

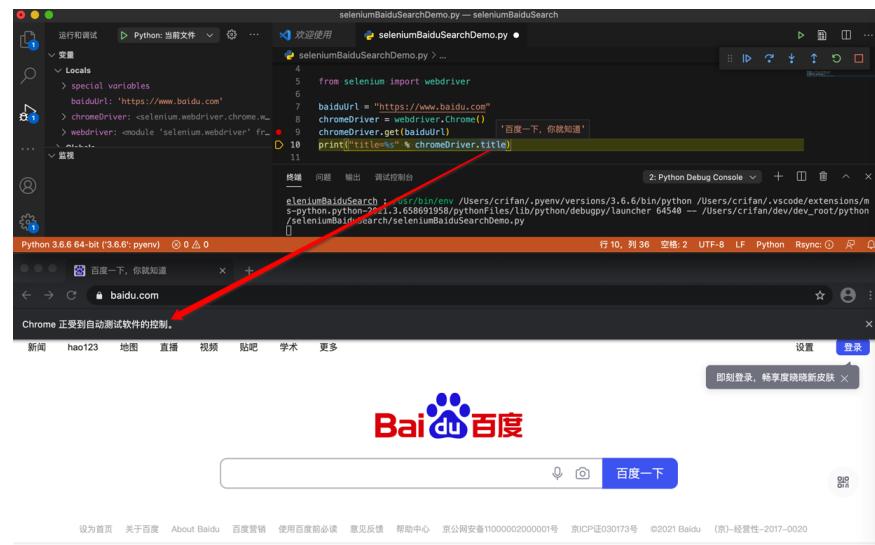
可以用代码：

```
from selenium import webdriver

baiduUrl = "https://www.baidu.com"
chromeDriver = webdriver.Chrome()
chromeDriver.get(baiduUrl)
print("title=%s" % chromeDriver.title)
```

确认Selenium是否正常工作：可以启动Chrome浏览器，打开百度首页。

正常的效果：



其中可以注意到： Selenium 操作的 Chrome 会有提示： Chrome正受到自动测试软件的控制

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 13:55:24

触发搜索

# 基本操作

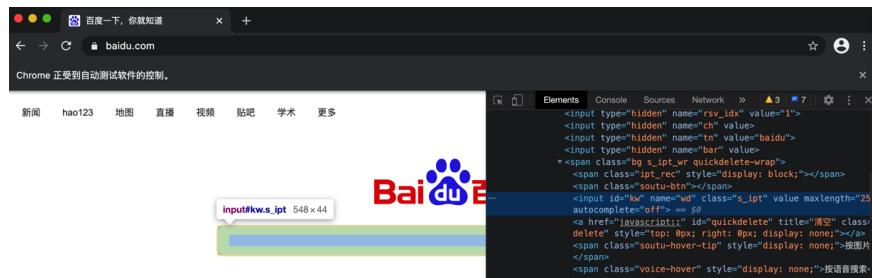
crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved,  
powered by Gitbook最后更新: 2021-04-29 13:55:24

## 查找定位元素

查找元素 = 定位元素

举例：

对于页面：



的html是：

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

对应查找该元素的典型方式是：

- `find_element_by_id(id_)`
  - 代码

```
driver.find_element_by_id("kw")
```

- 输出

```
<selenium.webdriver.remote.webelement.WebElement (s
```

- 文档

- [4.1. Locating by Id](#)

- `find_element_by_id(id_)`

- 注意：确保此处的id是唯一的

- `find_element(by='id', value=None)`
  - 代码

```
driver.find_element(by="kw")
```

- 文档

- `find_element(by='id', value=None)`

## 查找元素的更详细介绍

去Selenium中定位和查找元素：

方法有很多，常见的有：

- `find_element_by_id`
- `find_element_by_name`
- `find_element_by_xpath`
- `find_element_by_link_text`
- `find_element_by_partial_link_text`
- `find_element_by_tag_name`
- `find_element_by_class_name`
- `find_element_by_css_selector`

如果页面中有多个该元素，则可以用：

- `find_elements_by_name`
- `find_elements_by_xpath`
- `find_elements_by_link_text`
- `find_elements_by_partial_link_text`
- `find_elements_by_tag_name`
- `find_elements_by_class_name`
- `find_elements_by_css_selector`
- 官网文档
  - 中文
    - [4. 查找元素 — Selenium-Python中文文档 2 documentation](#)
  - 英文
    - [4. Locating Elements — Selenium Python Bindings 2 documentation](#)

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 13:55:24

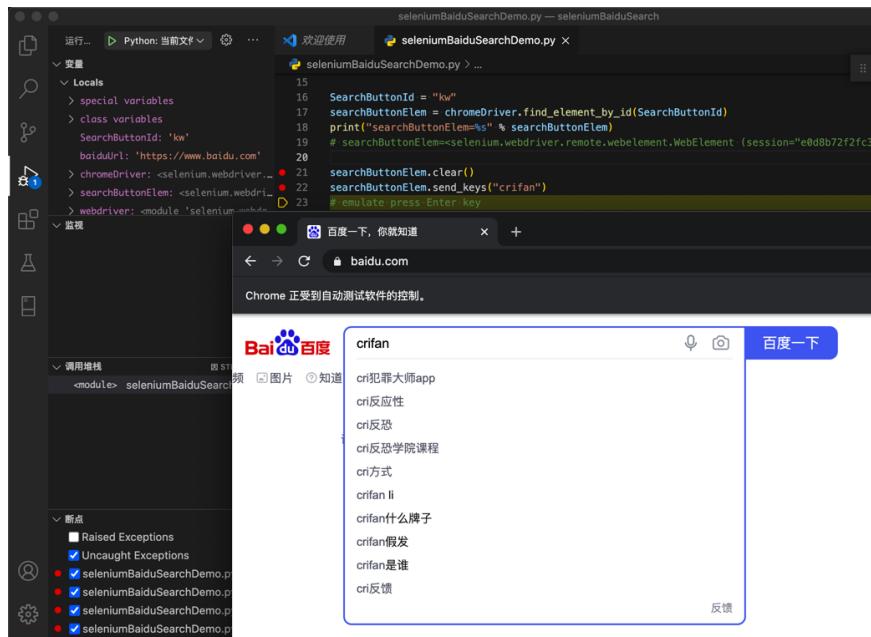
## (给元素) 输入文字

用 `send_keys`

举例：

```
searchButtonElem.send_keys("crifan")
```

效果：



## 相关：清除文字

在输入之前，往往可以或需要清楚（已输入的）文字：

```
searchButtonElem.clear()
```

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 13:55:24

## 点击元素

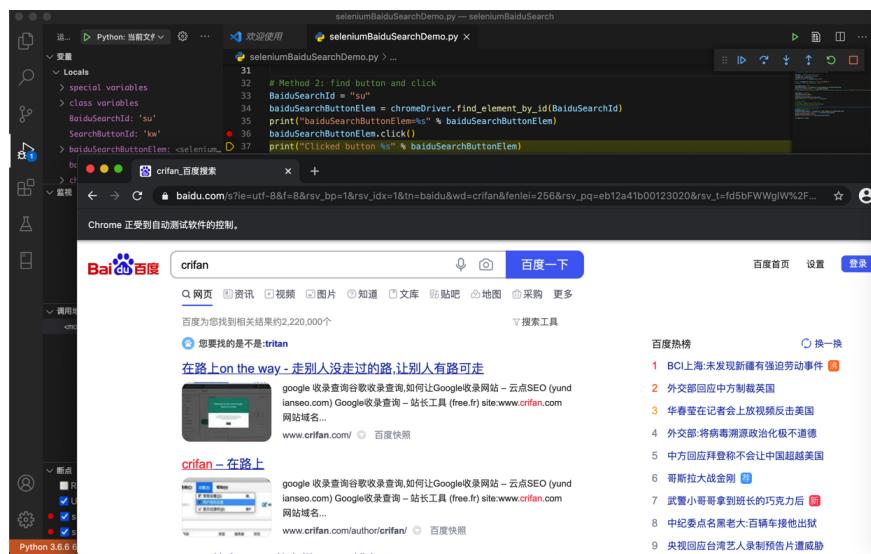
对于元素直接用 `click()` 即可。

举例：

```
baiduSearchButtonElem.click()
```

即可实现点击百度的 搜索一下 按钮

点击后的效果：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：2021-04-29 13:55:24

## 触发搜索

对于想要触发百度首页中的搜索来说，除了点击元素外，还可以模拟输入回车键。

- 触发百度搜索

- 方式1：点击 百度一下 按钮

```
BaiduSearchId = "su"  
baiduSearchButtonElem = chromeDriver.find_element_by_id(BaiduSearchId)  
baiduSearchButtonElem.click()
```

- 方式2：模拟输入 回车键

```
from selenium.webdriver.common.keys import Keys  
# Method 1: emulate press Enter key  
searchButtonElem.send_keys(Keys.RETURN)
```

附上：模拟百度输入并搜索的相关代码

```
searchStr = "crifan"  
searchButtonElem.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)
```

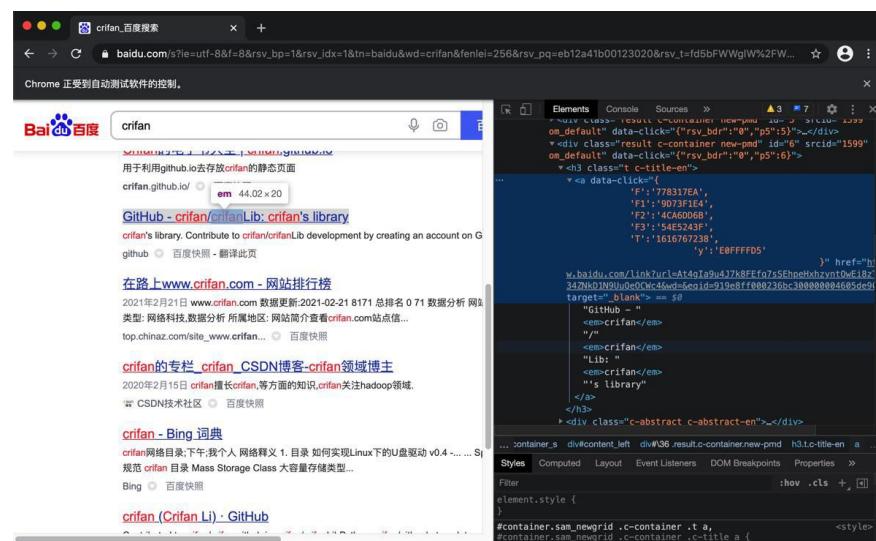
## 获取元素属性

获取元素属性的典型用法，比如：

- a元素
    - 获取a的href链接
      - `aElement.get_attribute("href")`
    - 获取a的文本值
      - `aElement.text`

## 举例：解析提取百度搜索结果

对于页面：



的html是：

```
<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=At4gIaS
```

已经通过代码：

```
searchResultAList = chromeDriver.find elements by xpath("//a[@class='list-item']")
```

然后就可以用 `htmlElement.get_attribute("href")` 去获取 `href` 的 url链接：

```
for curIdx, curSearchResultAElem in enumerate(searchResult):
    print("%s [%d] %s" % ("-"*20, curIdx, "-"*20))
    aHref = curSearchResultAElem.get_attribute("href")
    print("aHref=%s" % aHref)
```

类似的，想要获取文本值，用 `text`：

```
aText = curSearchResultAElem.text
print("aText=%s" % aText)
```

此处输出：

```
----- [0] -----
aHref=http://www.baidu.com/link?url=LMF5vQH-Qg0uEhaq5huV3bl
aText=在路上on the way - 走别人没走过的路,让别人有路可走
----- [1] -----
aHref=http://www.baidu.com/link?url=n4QoZVrJ5gncFIpJZhRcdmc
aText=crifan - 在路上
... 
```

## 特殊：对于html的解析，一般更常用专用的库：**BeautifulSoup**

对于html的解析，元素的获取等操作，往往换专用的html解析库：**BeautifulSoup**

举例，此处对应代码：

```
# 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)
for curIdx, searchResultH3Item in enumerate(searchResultH3List):
    print("%s [%d] %s" % ("-"*20, curIdx, "-"*20))
    aElem = searchResultH3Item.find("a")
    # print("aElem=%s" % aElem)
    baiduLinkUrl = aElem.attrs["href"]
    print("baiduLinkUrl=%s" % baiduLinkUrl)
    title = aElem.text
    print("title=%s" % title)
```

调试效果：

最终，同样的输出：

```
----- [0] -----
baiduLinkUrl=http://www.baidu.com/link?url=DVUb0ETLyMZLC5c_
title=在路上on the way - 走别人没走过的路,让别人有路可走
----- [1] -----
baiduLinkUrl=http://www.baidu.com/link?url=xA8mzlRBwfRb_I-I
title=crifan - 在路上
...

```

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 21:19:14

## Selenium心得和总结

### 单个WebElement本身好像不支持截图

详见：

[WebElement.screenshot\(filename\)](#)

和：

[WebElement.screenshot\\_as\\_png\(\)](#)

以为单个WebElement也不支持截图，但是试了试：

```
multipleXpathRule = '//div/.....'  
priceSpanElement = driver.find_element_by_xpath(multipleXpathRule)  
priceSpanElement.screenshot("priceElement.png")
```

结果报错：

```
selenium.common.exceptions.WebDriverException: Message:  
unknown command:  
session/7160497aa2d4dc2029bcb5c4d2e8045a/element/0.078535956  
38566757-1/screenshot
```

所以算了，不去管这个了。感觉是单个WebElement本身好像不支持截图

### get的url没法back或forward，而navigate的url可以

详见：[URL Loading in Selenium Webdriver: All about get\(\) and navigate\(\) – Make Selenium Easy](#)

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 21:20:31

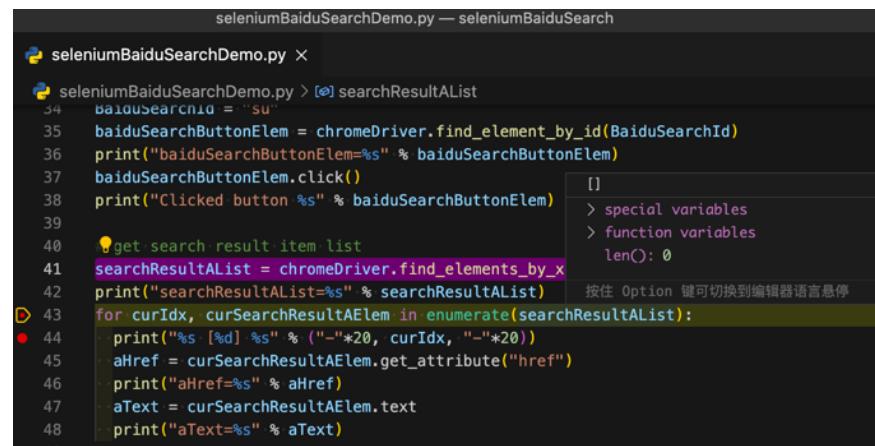
# 代码运行时却找不到元素

现象：查找元素的代码

```
searchResultAList = chromeDriver.find_elements_by_xpath("//
```

调试时可以找到元素。

但是直接运行代码，却找不到元素，返回是空：



The screenshot shows a code editor window with a tooltip displayed over a variable named 'searchResultAList'. The tooltip content is: '[]> special variables> function variableslen(): 0'.

```
seleniumBaiduSearchDemo.py — seleniumBaiduSearch
❷ seleniumBaiduSearchDemo.py ×
❸ seleniumBaiduSearchDemo.py > [e] searchResultAList
34 baiduSearchId = "su"
35 baiduSearchButtonElem = chromeDriver.find_element_by_id(BaiduSearchId)
36 print("baiduSearchButtonElem=%s" % baiduSearchButtonElem)
37 baiduSearchButtonElem.click()
38 print("Clicked button %s" % baiduSearchButtonElem)
39
40 #get search result item list
41 searchResultAList = chromeDriver.find_elements_by_X
42 print("searchResultAList=%s" % searchResultAList)
43 for curIdx, curSearchResultAElem in enumerate(searchResultAList):
44     print("%s [%d] %s" % ("-*20, curIdx, -*20))
45     aHref = curSearchResultAElem.get_attribute("href")
46     print("aHref=%s" % aHref)
47     aText = curSearchResultAElem.text
48     print("aText=%s" % aText)
```

原因：（网页）页面元素重新加载了，比如百度搜索导致页面重新加载，显示搜索结果内容，但是此时代码运行时，搜索结果还没加载出来，导致搜不到。

而调试时，由于有足够的暂停的时间，使得页面已加载新的搜索结果内容，所以再继续调试，可以搜索到。

解决办法：加上等待机制：等待页面加载完毕。

而判断页面加载完毕，则是需要具体问题具体分析。

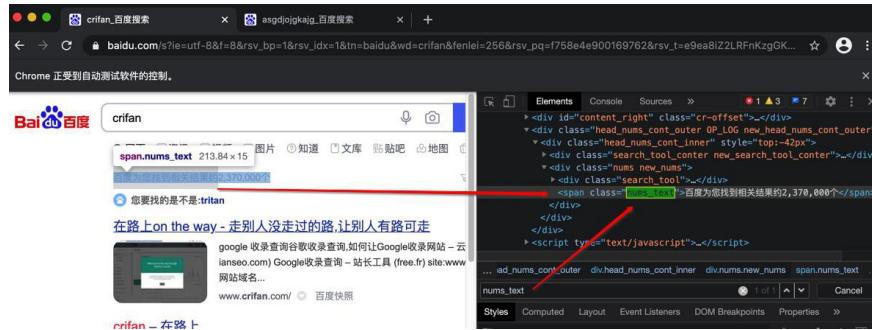
核心逻辑是：找到加载后的页面中，必然会出现的元素，作为判断的依据，判断该元素可见，则视为页面的确已加载完毕。

操作步骤：

此处经过调试，百度搜索结果中一定会出现：

**百度为您找到相关结果约 xxx**

## 触发搜索



对应的html是：

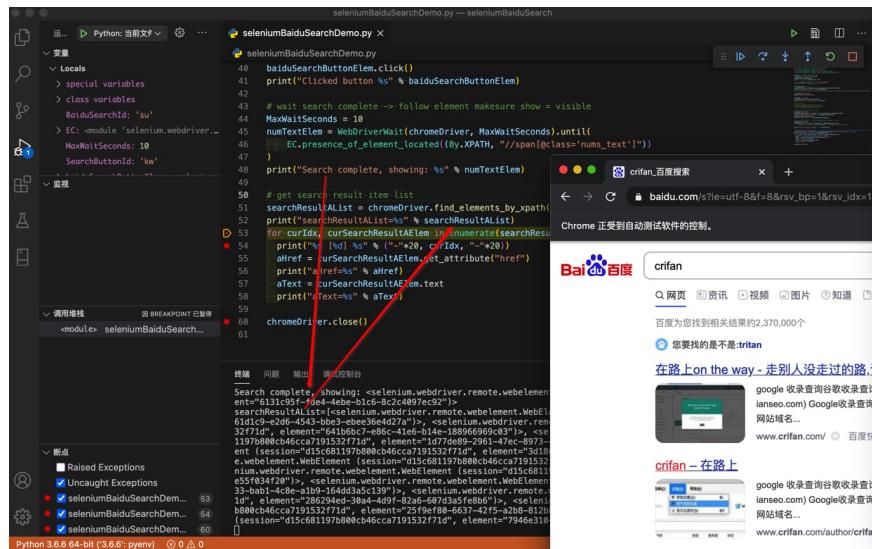
```
<span class="num_text">百度为您找到相关结果约2,370,000个</span>
```

对应的，等待一段时间，确保该元素出现的代码是：

```
# wait search complete -> follow element makesure show = visible
MaxWaitSeconds = 10
numTextElem = WebDriverWait(chromeDriver, MaxWaitSeconds).until(
    EC.presence_of_element_located((By.XPATH, "//span[@class='num_text']"))
)
print("Search complete, showing: %s" % numTextElem)
```

即可解决问题。

后续代码可以找到元素了：



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 21:21:08

## select不能用于ul

即，`select` 无法用于，非 `select` 的 `option` 下拉选项列表

`select` 只能用于

```
<select>
  <option>
```

才可以。其他的元素，比如之前遇到的：

[【已解决】Selenium如何点击下拉框并选择某个值](#)

中的

```
<ul>
  <li role="option">
```

是用不了的。

所以最后就是用普通的，去 `ul` 下找到 `li` 的列表，通过`index`获得对应的元素，然后再去操作。

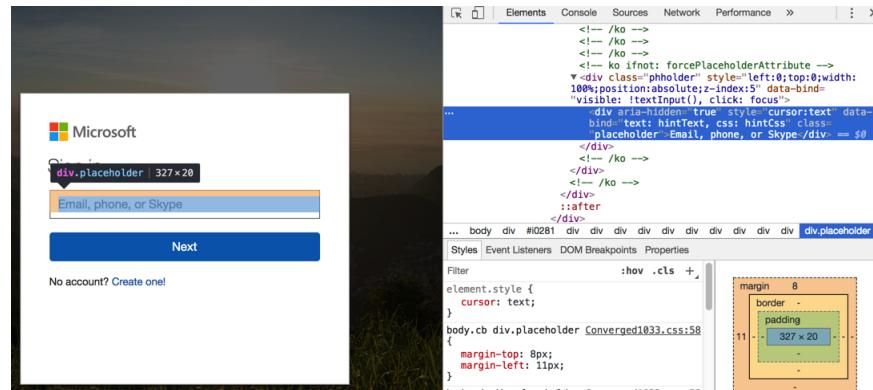
相关代码如下：

```
cartNumOptionElemList = driver.find_elements_by_xpath('>//ul')
cartNumOptionCount = len(cartNumOptionElemList)
logging.info("cartNumOptionElemList=%s, cartNumOptionCount=%s", cartNumOptionElemList, cartNumOptionCount)
if cartNumOptionCount < gCfg["msStore"]["onceBuyNum"]:
    logging.error("Current Cart select max number %s < expected %s", cartNumOptionCount, gCfg["msStore"]["onceBuyNum"])
    driver.quit()
toSelectIdx = gCfg["msStore"]["onceBuyNum"] - 1
# carNumSelect = Select(cartNumOptionElemList)
# carNumSelect.select_by_index(gCfg["msStore"]["onceBuyNum"])
carNumSelectElem = cartNumOptionElemList[toSelectIdx]
logging.info("carNumSelectElem=%s", carNumSelectElem)
carNumSelectElem.click()
# aLinkElem = carNumSelectElem.find_element_by_link_text(str(gCfg["msStore"]["onceBuyNum"]))
# logging.info("aLinkElem=%s", aLinkElem)
# aLinkElem.click()
```

# 偶尔Chrome右键元素不是你要的

即：有时候Chrome中直接右键找到的元素，并不一定是你想要的

比如：

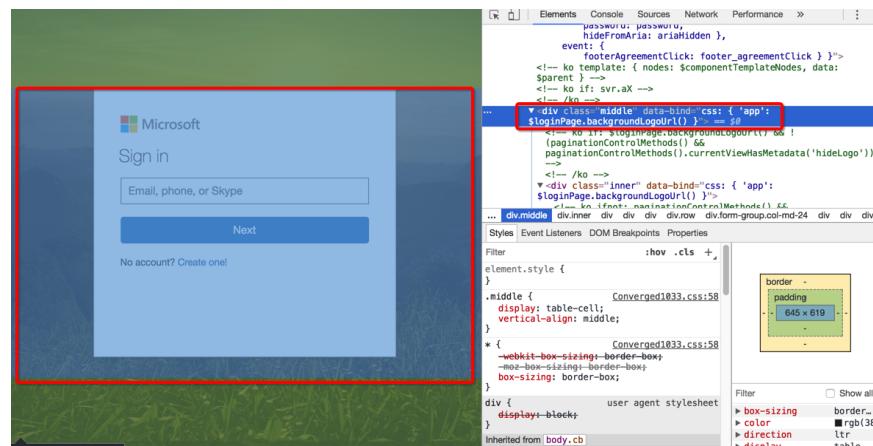


是个：

```
<div class="placeholder">
```

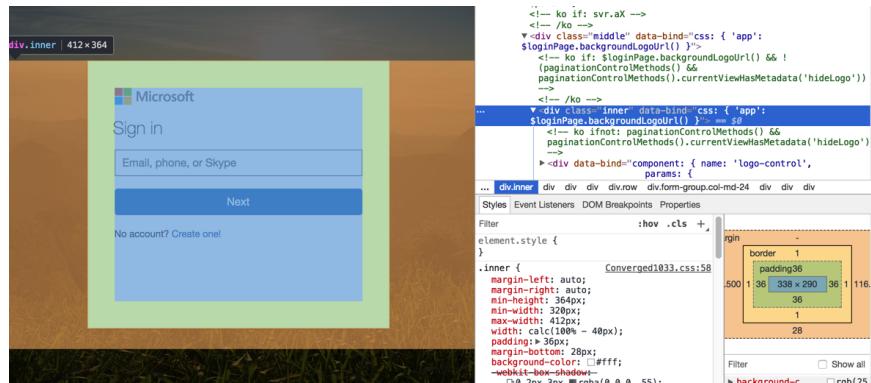
但是其实此处要找的是 可以允许输入的input输入框

而后来是无意间自己调试，从中间的区域，右键后：

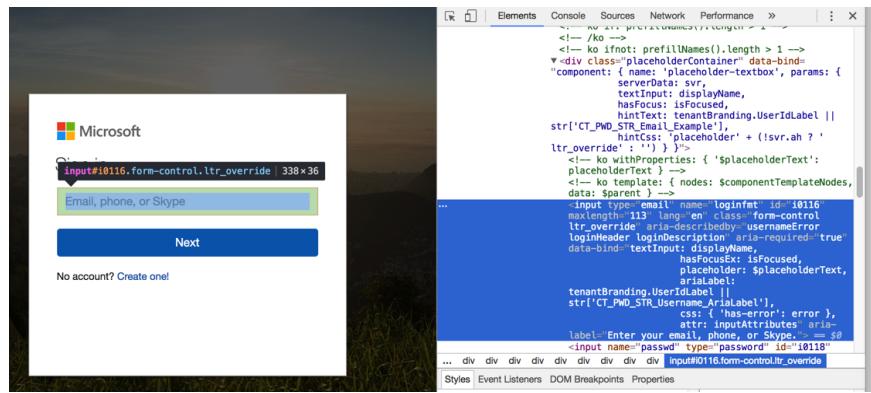


然后一点点点击看子元素：

触发搜索



最后找到真正的input的：



相关代码是：

```
<input type="email" name="loginfmt" id="i0116" .....  
attr: inputAttributes" aria-label="Enter yo
```

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 21:21:21

## 偶尔刷新后才能找到元素

即：有时候点击按钮后页面刷新且url地址也换了，再去用driver寻找元素之前，先要refresh然后才能找到

但是有时候却又不需要refresh也可以

最后是：

```
driver.refresh()  
inputEmailElement = driver.find_element_by_xpath('//div[@c...
```

或：

```
inputEmailElement = WebDriverWait(driver, 10).until(  
    EC.presence_of_element_located((By.XPATH, '//div[@clas...
```

好像都可以。

注：

不过还是不知道为何前面的代码：

```
placeholderElement = driver.find_element_by_xpath('//div[@c  
placeholderElement = WebDriverWait(driver, 10).until(  
    EC.presence_of_element_located((By.XPATH, '//div[@clas  
placeholderElement = driver.find_element_by_xpath('//div[@c  
pholderElement = driver.find_element_by_class_name('phold  
logging.info("phholderElement=%s", pholderElement)  
placeholderElement = pholderElement.find_element_by_class...
```

此处已确保 `xpath` 写的是对的，且查看页面元素的确是存在的，但却还是找不到元素。

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：2021-04-29 21:21:38

## 举例

下面给出Selenium的一些实例，来说明如何使用Selenium实现Web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：2021-04-29 21:20:07

## 模拟百度搜索

此处用Selenium，模拟百度搜索，并提取第一页搜索结果的信息。

### 代码

- 文件下载：[seleniumDemoBaiduSearch.py](#)
- 直接贴出源码

```
# Function: demo selenium do baidu search and extract results
# 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

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
#####
SearchButtonId = "kw"
searchButtonElem = chromeDriver.find_element_by_id(SearchButtonId)
print("searchButtonElem=%s" % searchButtonElem)
# searchButtonElem=<selenium.webdriver.remote.webelement.WebElement object at 0x10c3a2d0>

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

searchStr = "crifan"
searchButtonElem.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": 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)

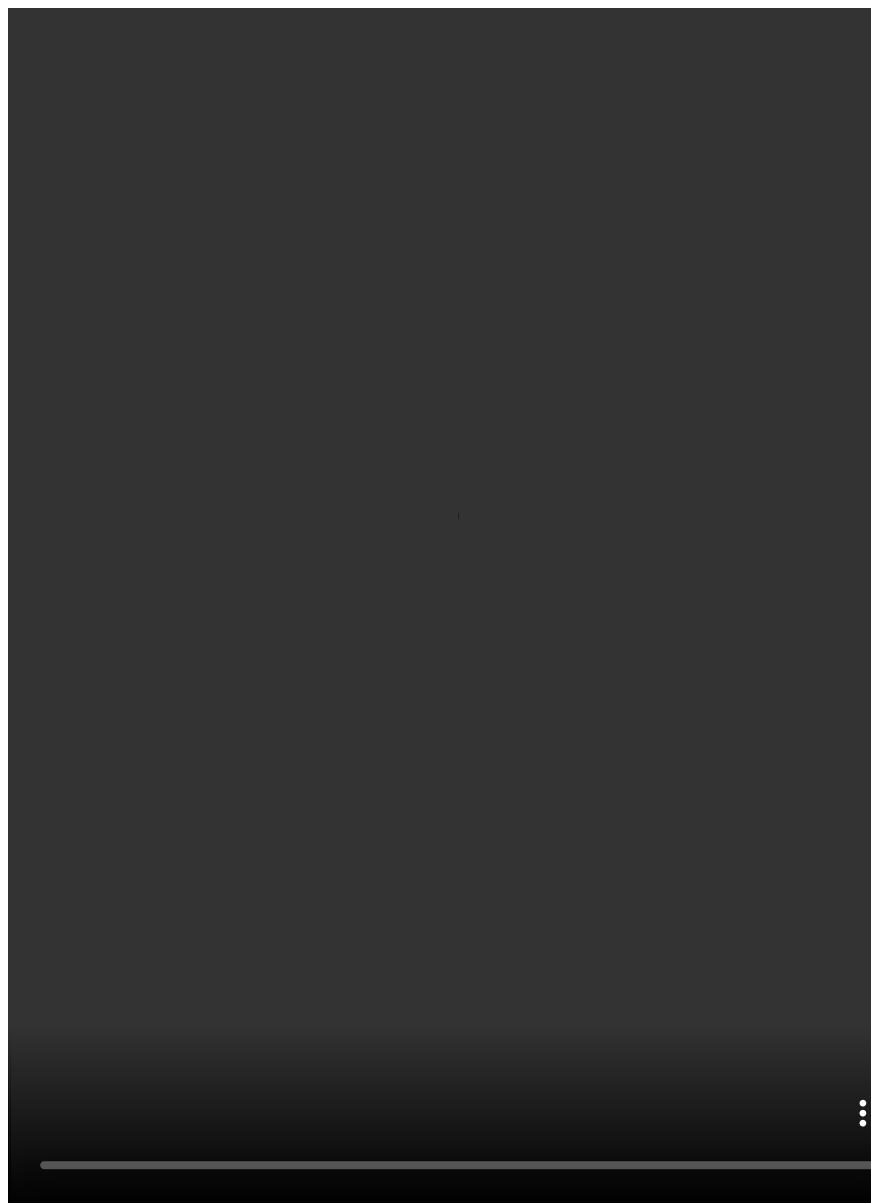
```

触发搜索

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

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

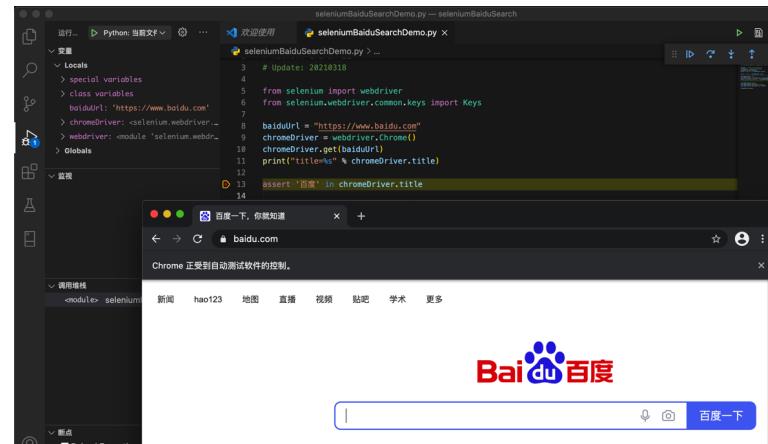
## 视频



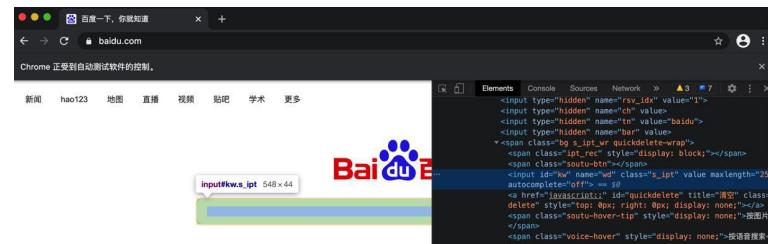
## 相关图片

- 打开百度首页

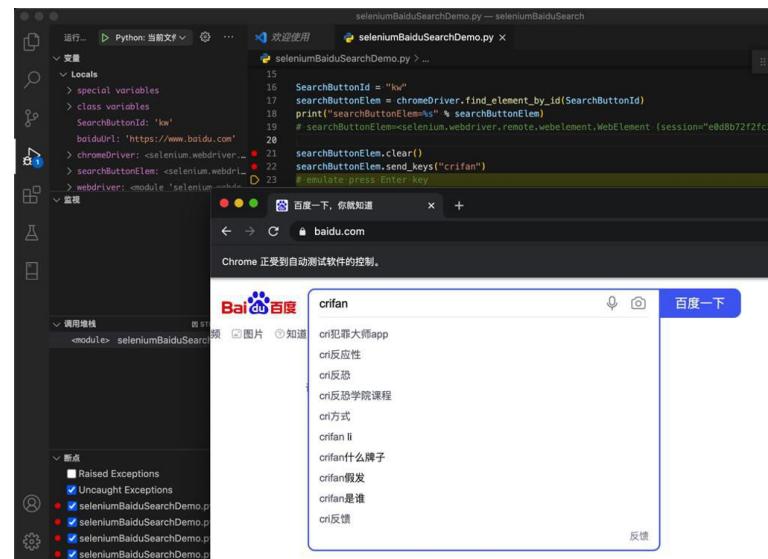
## 触发搜索



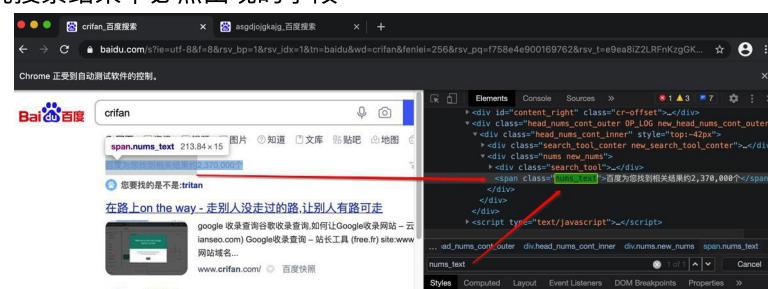
- 调试输入框的html



- 输入框中可以输入搜索关键字: crifan

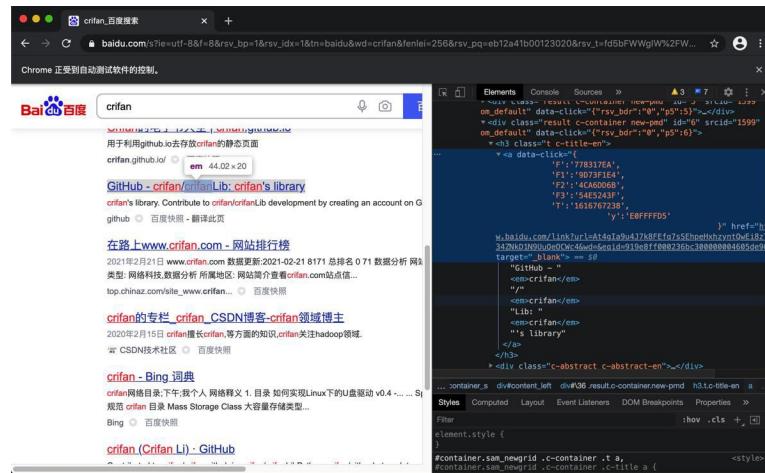


- 研究搜索结果中必然出现的字段

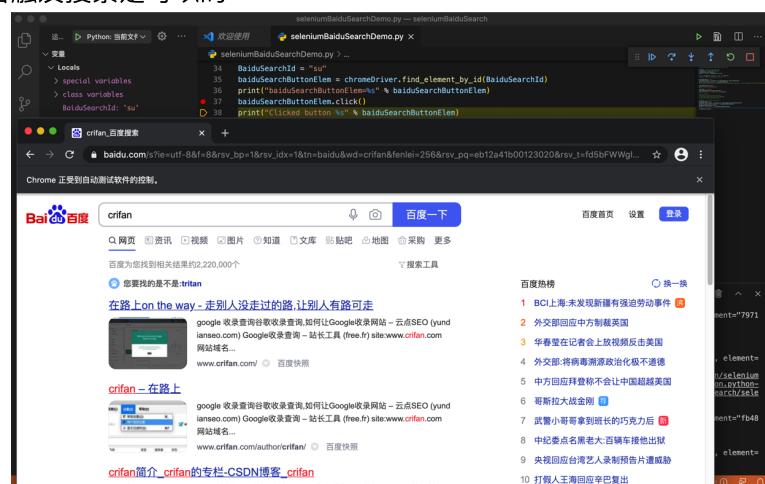


- 调试搜索结果每一项的html

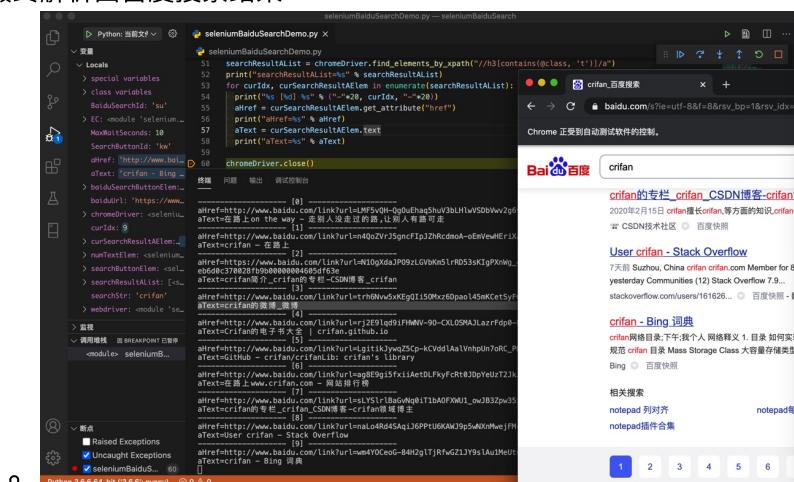
## 触发搜索



- 点击触发搜索是可以的



- 最终解析出百度搜索结果



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: 2021-04-29 21:19:43

## 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 13:55:24

# Selenium文档

## 官网文档

- 官网
  - 英文
    - 3 Navigating — Selenium Python Bindings 2 documentation
      - <https://selenium-python.readthedocs.io/navigating.html>
    - 4 Locating Elements — Selenium Python Bindings 2 documentation
      - <https://selenium-python.readthedocs.io/locating-elements.html>
    - 5 Waits — Selenium Python Bindings 2 documentation
      - <https://selenium-python.readthedocs.io/waits.html>
    - 6 Page Objects — Selenium Python Bindings 2 documentation
      - <https://selenium-python.readthedocs.io/page-objects.html>
    - 7 WebDriver API — Selenium Python Bindings 2 documentation
      - <https://selenium-python.readthedocs.io/api.html>
  - 中文
    - 4 查找元素 — Selenium-Python中文文档 2 documentation
      - <https://selenium-python-zh.readthedocs.io/en/latest/locating-elements.html>
    - 5 等待页面加载完成(Waits) — Selenium-Python中文文档 2 documentation
      - <https://selenium-python-zh.readthedocs.io/en/latest/waits.html>
    - 6 页面对象 — Selenium-Python中文文档 2 documentation
      - <https://selenium-python-zh.readthedocs.io/en/latest/page-objects.html>
    - 7 WebDriver API — Selenium-Python中文文档 2 documentation
      - <https://selenium-python-zh.readthedocs.io/en/latest/api.html>

## WebElement有很多函数和属性

整理如下，供有个概念：

- 函数
  - `clear()`
  - `click()`
  - `find_element(by='id', value=None)`
  - `find_element_by_class_name(name)`
  - `find_element_by_css_selector(css_selector)`
  - `find_element_by_id(id_)`
  - `find_element_by_link_text(link_text)`
  - `find_element_by_name(name)`
  - `find_element_by_partial_link_text(link_text)`
  - `find_element_by_tag_name(name)`
  - `find_element_by_xpath(xpath)`
  - `find_elements(by='id', value=None)`
  - `find_elements_by_class_name(name)`
  - `find_elements_by_css_selector(css_selector)`
  - `find_elements_by_id(id_)`
  - `find_elements_by_link_text(link_text)`
  - `find_elements_by_name(name)`
  - `find_elements_by_partial_link_text(link_text)`
  - `find_elements_by_tag_name(name)`
  - `find_elements_by_xpath(xpath)`
  - `get_attribute(name)`
  - `get_property(name)`
  - `is_displayed()`
  - `is_enabled()`
  - `is_selected()`
  - `screenshot(filename)`
  - `send_keys(*value)`
  - `submit()`
  - `value_of_css_property(property_name)`
- 属性
  - `id`
  - `location`
  - `parent`
  - `rect`
  - `screenshot_as_png`
  - `size`
  - `text`
  - `tag_name`

更多内容详见官网文档: [WebElement](#)

其中:

- `find_element_by_link_text`

触发搜索

- `find_element_by_partial_link_text`

指的是标签 `a` 的 `link`，而其他标签是用不了的。

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: 2021-04-29 21:18:32

## 参考资料

- 【已解决】Selenium中如何定位元素：百度首页中的输入框
- 【整理】用Chrome或Chromium查看百度首页中各元素的html源码
- 【已解决】Mac中搭建Selenium的Python开发环境
- 【已解决】Mac中下载Selenium的Chrome的driver: ChromeDriver
- 【已解决】Selenium中给百度搜索框中输入文字并触发搜索
- 【已解决】Selenium中Python解析百度搜索结果第一页获取标题列表
- 【已解决】Selenium中如何实现百度搜索结果标题元素的定位
- 【已解决】Mac中用Selenium自动操作浏览器实现百度搜索
- 【已解决】Selenium调试时能搜到元素但是直接运行找不到
- 【已解决】Selenium中如何获取到当前页面的html源码
- 
- 【整理】用Chrome或Chromium查看百度首页中各元素的html源码 – 在路上
- 【已解决】Selenium中如何定位元素：百度首页中的输入框 – 在路上
- 【已解决】Selenium中给百度搜索框中输入文字并触发搜索 – 在路上
- 【整理】用Chrome或Chromium查看百度首页中各元素的html源码
- 【已解决】Selenium再次出错：NoSuchElementException:  
Message: no such element: Unable to locate element
- Mac os上配置selenium并使用python操作web页面\_iluxiaoxiaoniao的博客-CSDN博客\_mac python webdriver
- mac 搭建selenium与ChromeDriver环境 - 简书
- selenium · PyPI
- 2. 快速入门 — Selenium-Python中文文档 2 documentation
- Selenium with Python — Selenium Python Bindings 2 documentation
- Selenium Form WebElement: TextBox, Button, sendkeys(), click()
- java - Selenium Webdriver: Entering text into text field - Stack Overflow
- How to locate and insert a value in a text box (input) using Python Selenium? - Stack Overflow
- java - Using Selenium Web Driver to retrieve value of a HTML input - Stack Overflow
- Selenium WebDriver get text from input field - Stack Overflow
- How to use Selenium to input text in Python - Stack Overflow
- Get value of an input box using Selenium (Python) - Stack Overflow
- 1. Navigating — Selenium Python Bindings 2 documentation
-

触发搜索

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 21:18:38