

# Web自动化测 试技术之 Selenium

22210180053

黄珂玮

# SELENIUM 环境搭建

---

Anaconda环境搭建

Docker环境搭建

# SELENIUM 初始化操作

---

设置chrome文件位置

添加启动参数

添加拓展应用

添加实验选项

# 如何进行 元素定位

---

Selenium的元素查找函数

简单定位

高级定位

# SELENIUM 动作&等待

---

模拟键盘输入和鼠标操作

强制等待，隐形等待，显性等待

# 网页窗口 动作

---

窗口前进/后退

窗口切换

# 例子

---

平安复旦

股票简报制作

# WEB自动化测试

---

什么是Web自动化测试？

从UI（用户界面）层面进行的自动化测试，测试人员通过编写自动化程序（测试用例脚本）来打开浏览器测试网站的业务逻辑。

什么Web项目适合做自动化测试？

- 1.需求变动不频繁
- 2.项目周期长
- 3.项目需要回归测试

# SELENIUM

---

1. Selenium 是用于测试 Web 应用程序用户界面 (UI) 的常用框架。Selenium 能够在一个或多个浏览器中执行这些测试。
2. Selenium 测试直接运行在浏览器中，就像真正的用户在操作一样。

00

环境搭建

# 详解整套配置

1. 准备python环境，推荐大家使用Anaconda为每一个项目创建一个虚拟环境
2. 使用pip下载selenium

```
% pip install selenium  
...  
如果采用anaconda虚拟环境，则需要在cmd(windows)或者终端(macOS)中  
conda activate XXX  
pip install selenium  
...
```

Python

3. 安装浏览器和驱动参考

chrome驱动地址：<https://chromedriver.chromium.org/downloads>

根据版本下载对应的驱动（查看版本可以通过在Chrome中输入chrome://version/）



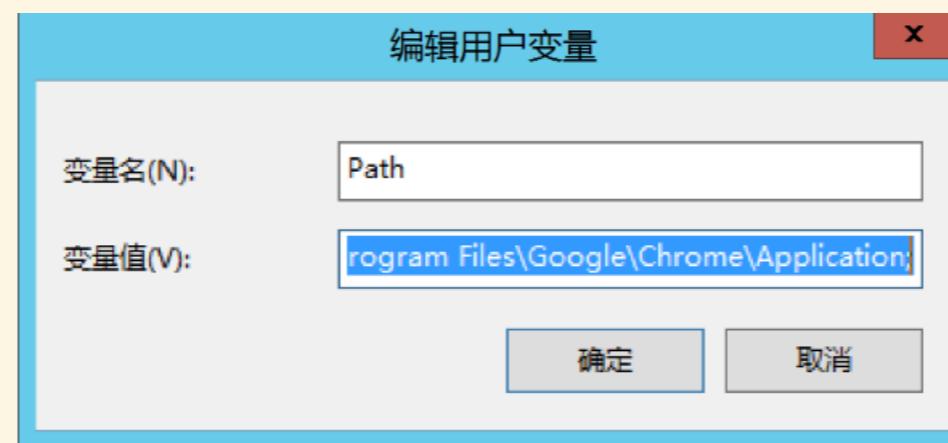
- If you are using Chrome version 106, please download [ChromeDriver 106.0.5249.21](#)
- If you are using Chrome version 105, please download [ChromeDriver 105.0.5195.52](#)
- If you are using Chrome version 104, please download [ChromeDriver 104.0.5112.79](#)

# 详解整套配置

## 4. 设置浏览器驱动路径

1. macOS : Finder >> Shift+⌘+G >> 输入 /usr/local/bin/chromedriver >> Enter , 将下载解压后的chromedriver拖到该处。

2. win : 将下载解压后的文件拖到特定文件夹（避免以后更新的时候找不到），添加文件夹路径到 控制面板\系统和安全\系统\高级系统设置\环境变量 的PATH中，以防万一可以全局变量和用户变量均添加



# 详解整套配置（一些参考配置）

---

## 5. Jupyter Notebook

这个东西不是必须的，不过用它来做实验会比较方便。并且如果你愿意忙活的话，可以去租一台有外网IP的服务器，这样就可以开放8888或者8889端口，能够远程访问你的Jupyter notebook(Lab)。

## 6. IDE的选择

在PC端，我一般使用Vscode和Pycharm，很少使用浏览器进行Jupyter notebook的编辑。

在iPad端，Pythonista太弱。Juno是iPad上最好的的Jupyter notebook（甚至可以说python）的编辑器。如果你有上面一条中的云服务器，你也可以尝试Juno Connect，它的界面十分优美。

# DOCKER启动虚拟环境

---

优点：适用Windows、MacOS、CentOS等多种操作系统，并且一般不会因为系统不同报错。环境建立仅需几条命令。

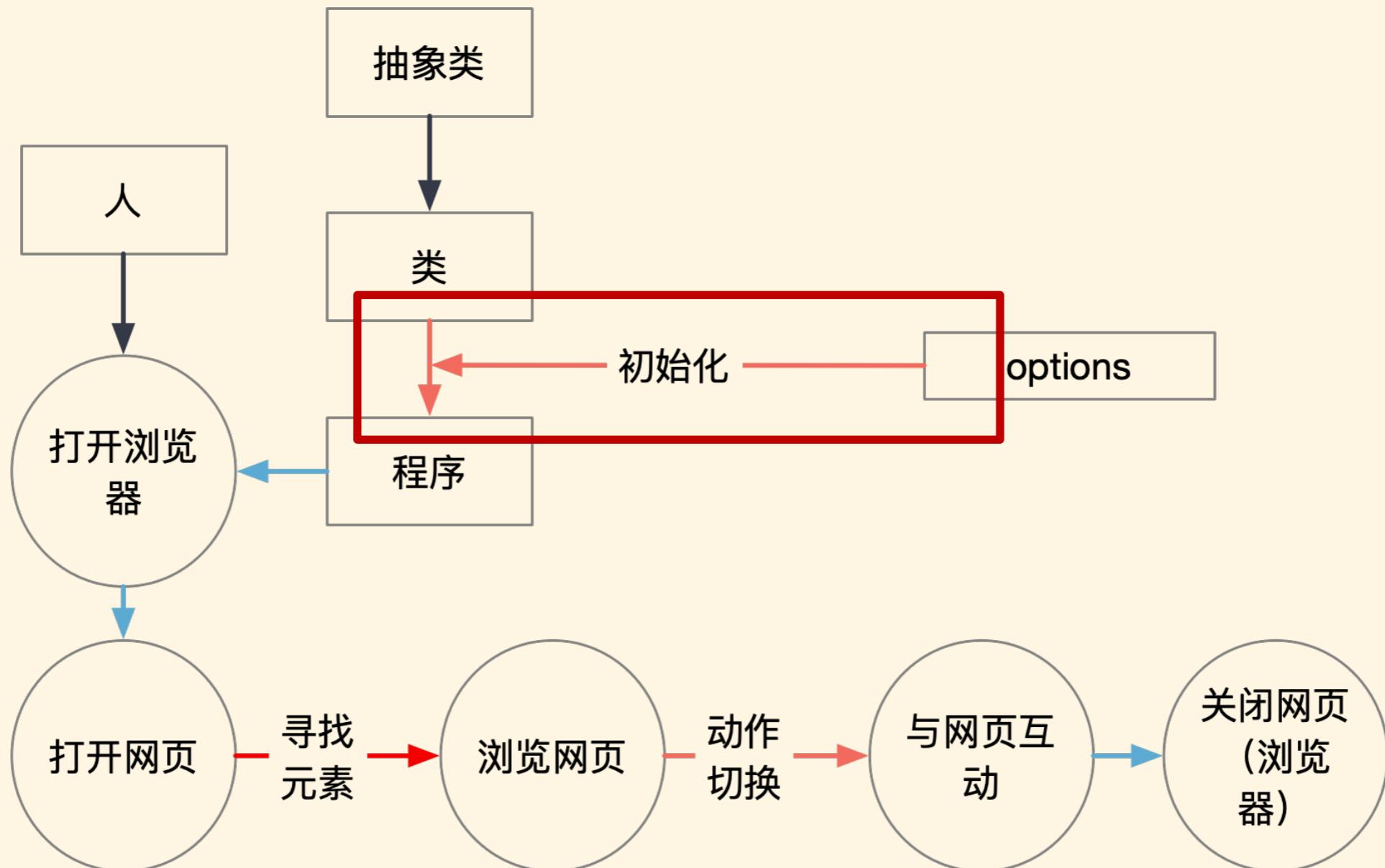
1. Clone我的文件夹 <https://github.com/BryanHuang66/selenium-cookbook>
2. 各系统配置各自的Docker环境 <https://www.docker.com/>（下载完，基本不需要什么操作）
3. 启动Docker，注册，登录用户（否则无法连接hub下载镜像）
4. 输入命令，进入1步骤中解压的根目录，例如：```cd users/creative/python/selenium```
5. 输入命令，```docker build -f \$(pwd)/Dockerfile -t jupyter-selenium .```，建立虚拟环境
6. 输入命令，```docker run -it -v \$(pwd)/work:/home/jovyan/work -p 8888:8888 --name jupyter-selenium jupyter-selenium```，启动容器
7. 根据终端指示进入Jupyter Notebook

```
To access the server, open this file in a browser:  
file:///home/jovyan/.local/share/jupyter/runtime/jpserver-7-open.html  
Or copy and paste one of these URLs:  
http://3dcab5d432a5:8888/lab?token=536d3dc249f489958c1b70a55661f5c5b278f  
998602a620b  
or http://127.0.0.1:8888/lab?token=536d3dc249f489958c1b70a55661f5c5b278f998  
602a620b
```

0 ||

初始化

# PYTHON是面向对象的语言



# 初始化中OPTIONS常用属性和方法

```
from selenium import webdriver  
from selenium.webdriver.chrome.options import Options
```

Python

## options常用方法为：

1. add\_argument() : 添加启动参数
2. add\_extension() : 添加本地插件
3. add\_experimental\_option() : 添加实验选项

## Options.add\_argument()常用参数为：

1. no-sandbox : 不使用沙盒
2. headless: 无界面模式

# OPTIONS常用启动参数

通过`.add_argument()`可以添加启动参数，如初始化窗口尺寸，隐身模式等。

1. 设置无头浏览器：从而使得浏览器在不显示图形界面的情况下，通过编程来控制自动执行各种任务，比如做测试，给网页截屏等。

```
options = Options()
options.add_argument('headless')
driver = webdriver.Chrome(options=options)
driver.get("http://www.baidu.com")
driver.save_screenshot('screenshot_google.png')
time.sleep(3)
driver.close()
Image("screenshot_google.png",width=300)
```

✓ 5.7s

Python



# OPTIONS常用启动参数

## 2. 设置chrome浏览器位置

```
options.binary_location = '/usr/bin/google-chrome'
```

```
## 打开浏览器
options = Options()
## 如果使用的是docker虚拟环境，设置google-chrome路径
if os.path.exists('/usr/bin/google-chrome'):
    options.binary_location = '/usr/bin/google-chrome'
driver = webdriver.Chrome(options=options)
driver.get("http://www.baidu.com")
driver.close()
```

✓ 7.4s

Python

## 3. 设置user-agent：指定用户客户端-模拟手机浏览

```
options.add_argument('user-agent="MQQBrowser/26 Mozilla/5.0 (Linux; U; Android 2.3.7; zh-cn; MB200 Build/GRJ22; CyanogenMod-7) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1"')
```

```
options = Options()
options.add_argument('user-agent="MQQBrowser/26 Mozilla/5.0 (Linux; U; Android 2.3.7; zh-cn; MB200 Build/GRJ22; CyanogenMod-7) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1"')
driver = webdriver.Chrome(options=options)
agent = driver.execute_script("return navigator.userAgent")
print(f'user-agent 为 {agent}')
driver.close()
```

✓ 2.1s

Python

user-agent 为 "MQQBrowser/26 Mozilla/5.0 (Linux; U; Android 2.3.7; zh-cn; MB200 Build/GRJ22; CyanogenMod-7) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1"

原来的UA：

user-agent 为 Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36

# OPTIONS常用启动参数

## 4. 设置代理：绕开反爬策略

```
options.add_argument("--proxy-server=http://120.194.55.139:6969")
```

```
options = Options()
options.add_argument("--proxy-server=http://58.20.184.187:9091")
driver = webdriver.Chrome(options=options)
driver.get("http://httpbin.org/ip")
print(driver.page_source)
driver.close()
```

✓ 4.5s

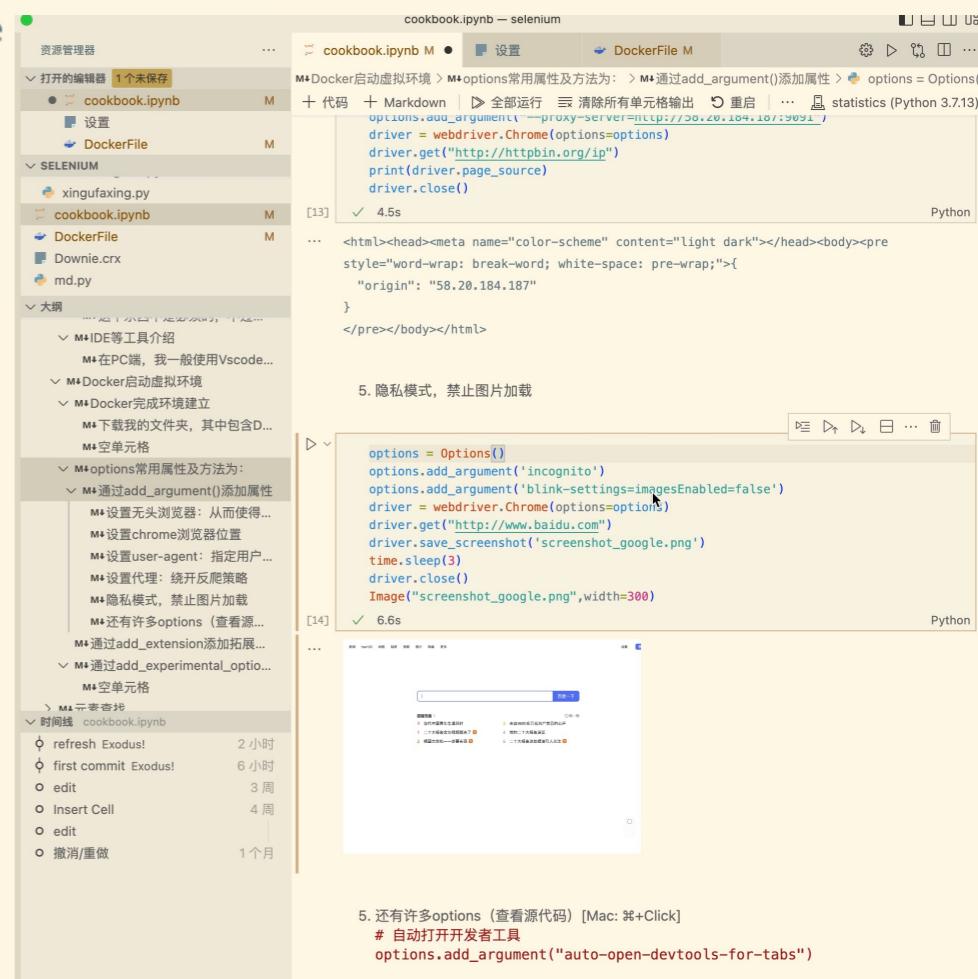
Python

```
<html><head><meta name="color-scheme" content="light dark"></head><body><pre style="word-wrap: break-word; white-space: pre-wrap;">
  "origin": "58.20.184.187"
</pre></body></html>
```

## 5. 隐私模式，禁止图片加载

```
options.add_argument('incognito')
```

```
options.add_argument('blink-settings=imagesEnabled=false')
```



# OPTIONS常用启动参数

---

## 6. 自动打开开发者工具

```
options.add_argument("auto-open-devtools-for-tabs")
```

## 7. 设置窗口尺寸，注意宽高之间使用逗号而不是x

```
options.add_argument('window-size=300,600')
```

## 8. 设置窗口启动位置（左上角坐标）

```
options.add_argument('window-position=120,0')
```

## 9. 禁用gpu渲染：杜绝gpu造成的一部分浏览器定位失败

```
options.add_argument('disable-gpu')
```

## 10. 全屏启动

```
options.add_argument('start-fullscreen')
```

## 11. 全屏启动，无地址栏

```
options.add_argument('kiosk')
```

## 12. 启动时，不激活（前置）窗口

```
options.add_argument('no-startup-window')
```

**注意：如果要在Docker环境内运行需要**

```
options.add_argument('--no-sandbox')
```

```
options.add_argument('disable-gpu')
```

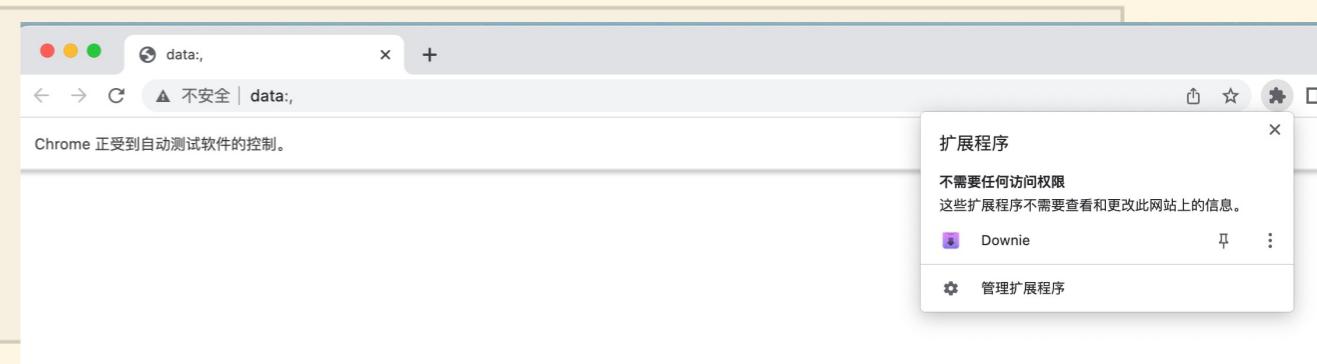
**在设置参数时，添加-- 和不添加的效果是一样的**

# 其他拓展

## 1. 通过add\_extension()添加拓展应用

```
options.add_extension('Downie.crx')
```

```
options = Options()
options.add_extension('Downie.crx')
driver = webdriver.Chrome(options=options)
✓ 2.4s
```



## 2. 通过add\_experimental\_option()添加实验选项

```
prefs["credentials_enable_service"] = False
```

```
prefs["profile.password_manager_enabled"] = False
```

```
options = Options()
prefs = {}
# 设置这两个参数就可以避免密码提示框的弹出
prefs["credentials_enable_service"] = False
prefs["profile.password_manager_enabled"] = False
options.add_experimental_option("prefs", prefs)
driver = webdriver.Chrome(chrome_options=options)
driver.get('https://www.baidu.com/')
```

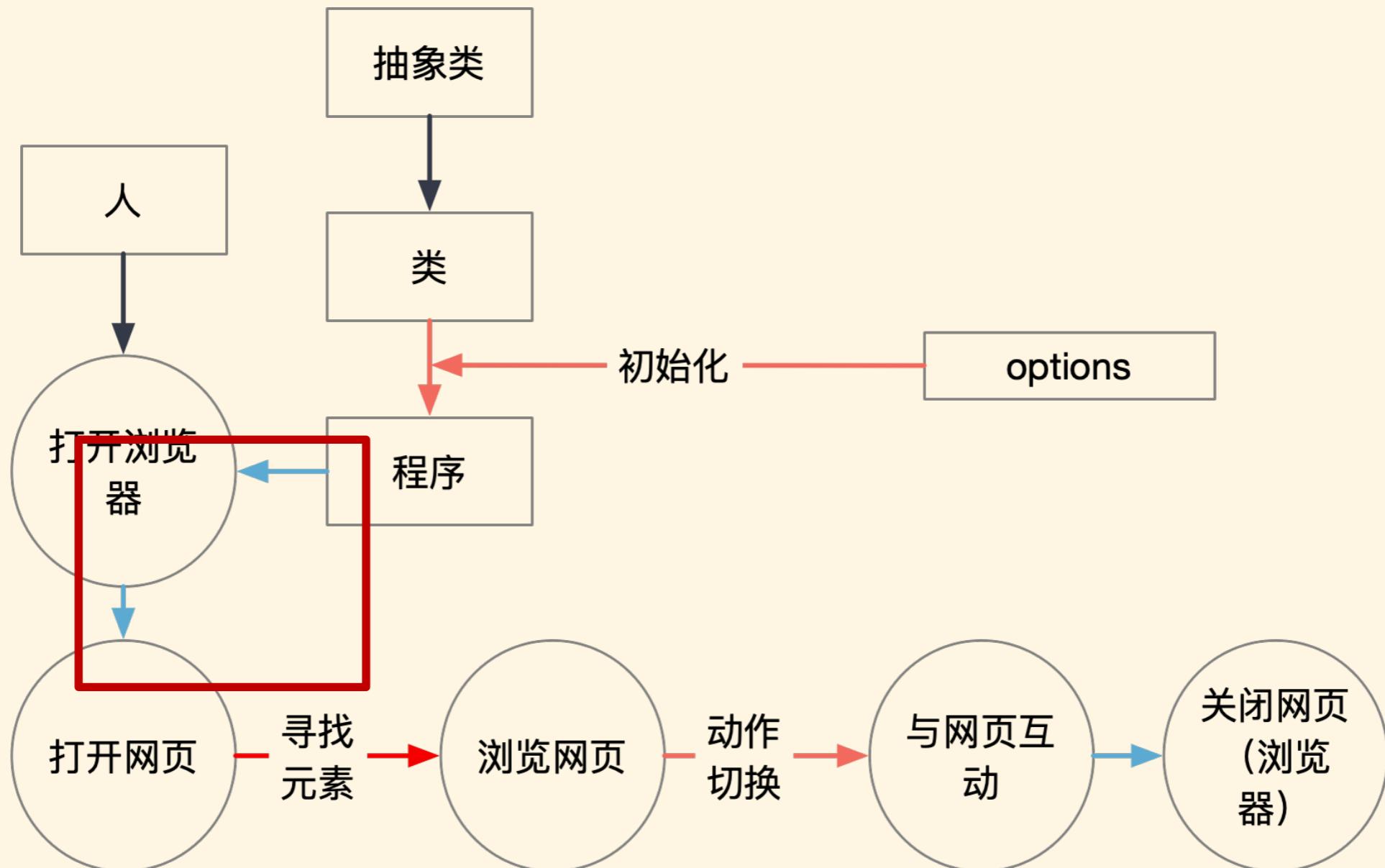
✓ 3.3s

Python

02

元素查找

# PYTHON是面向对象的语言



# 打开浏览器和网页

```
(class) Chrome(executable_path: str = DEFAULT_EXECUTABLE_PATH, port: int =  
DEFAULT_PORT, options: Options = None, service_args: Any | None = None,  
desired_capabilities: Any | None = None, service_log_path: Any | None =  
DEFAULT_SERVICE_LOG_PATH, chrome_options: Any | None = None, service: Service =  
None, keep_alive: Any | None = DEFAULT_KEEP_ALIVE)
```

启动服务，然后创建 chrome 驱动程序的新实例

参数：

- executable\_path - Deprecated: path to the executable. If the default is used it assumes the executable is in the \$PATH
- options - this takes an instance of ChromeOptions
- service - Service object for handling the browser driver if you need to pass extra details

1. 通过executable\_path实例化webdriver

```
driver = webdriver.Chrome(options=options)
```

2. 通过service实例化webdriver

```
from selenium import webdriver  
from selenium.webdriver.firefox.service import Service as FirefoxService  
from webdriver_manager.firefox import GeckoDriverManager  
driver = webdriver.Firefox(service=FirefoxService(GeckoDriverManager().install()))
```

```
from selenium import webdriver  
from selenium.webdriver.chrome.service import Service as ChromeService  
from webdriver_manager.chrome import ChromeDriverManager  
driver = webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()))
```

# 打开浏览器和网页

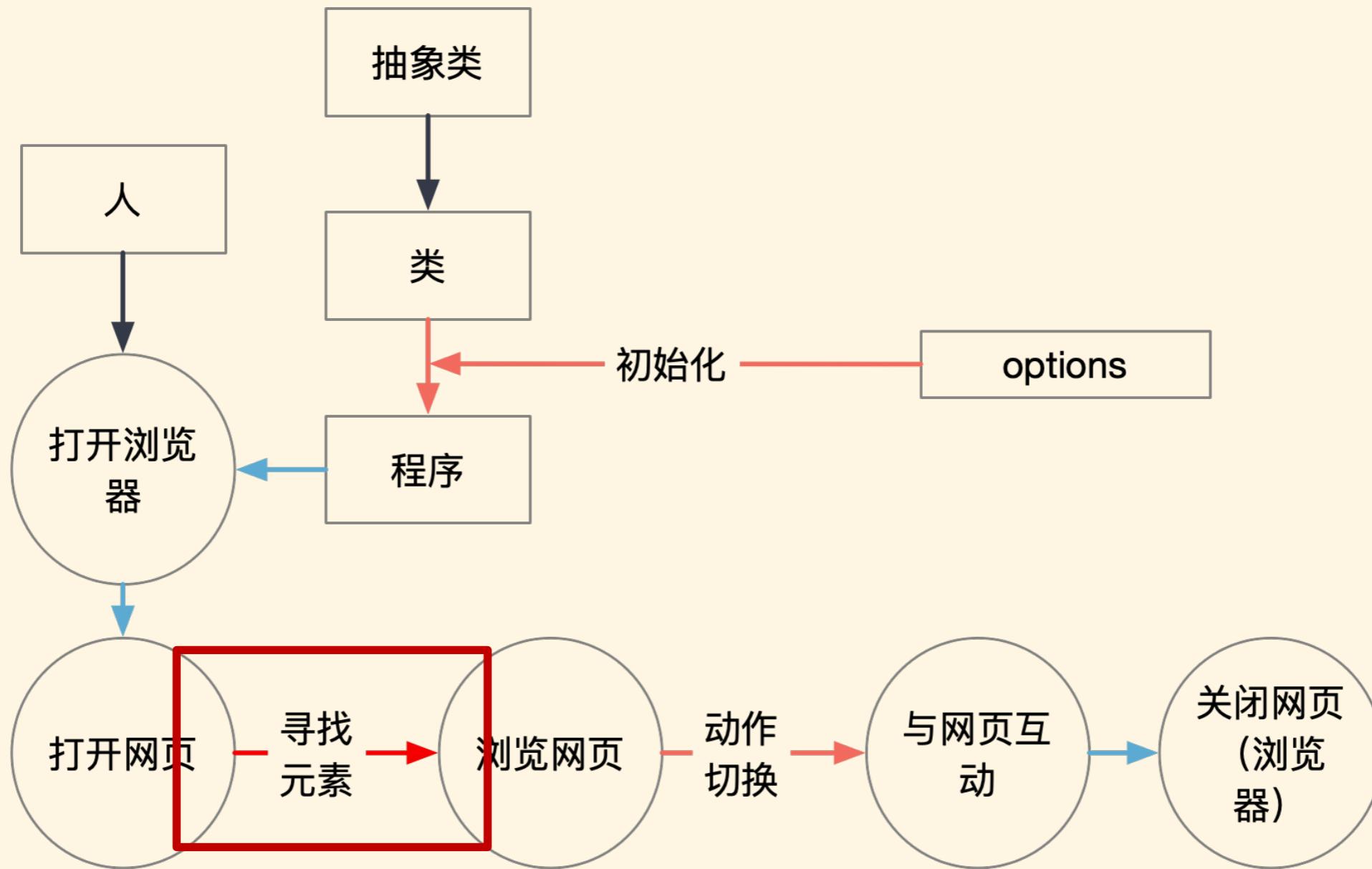
---

(method) get: (url: str) -> None

通过 driver.get() 打开网页：

```
from selenium import webdriver
## 打开浏览器
driver = webdriver.Chrome(options=options)
## 打开网页
driver.get("http://www.baidu.com")
```

# PYTHON是面向对象的语言



# HTML源代码和审查元素

返回

前进

重新加载

---

存储为...

打印...

投放...

使用 Google Lens 搜索图片

---

发送到您的设备

为此页面创建二维码

---

翻成中文（简体）

 Get cookies.txt

 Open this page in IINA

显示网页源代码

检查

Web前端三剑客：HTML、CSS、JavaScript（TypeScript）

个人观点：

HTML和CSS静态的组成了整个网页。

Javascript则动态的生成数据，这一部分会在用户访问网页后执行，所以产生的信息也就不会出现在网页源代码中。检查的时候则会包括源代码+JS动态渲染的内容，即最终展示的HTML内容

与其他的访问网页不同（比如requests库就是发送请求，获取网页的源代码，没有办法获得渲染后的数据），Selenium可以看到网页加载后的元素，所以能够更加直观地对网页进行分析。

# HTML标签和属性

```
<p class="greeting"><font size="5" color="red">Hello, World!</font></p>
```

Hello, World!

一般形式： <标签名 属性名1="属性值" 属性名2="属性值" ... 属性名N="属性值">内容</标签名>

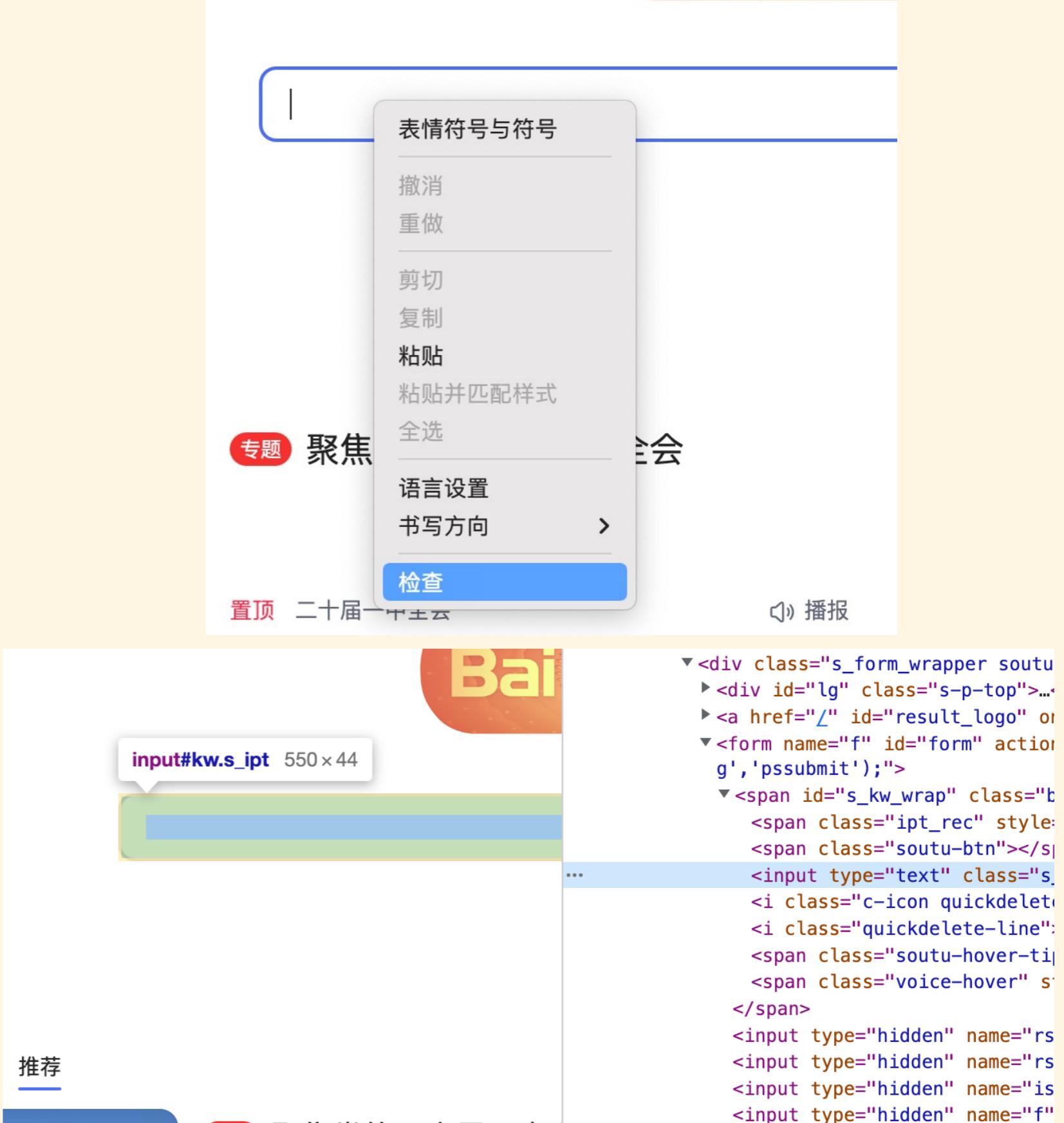
HTML属性一般都出现在HTML标签中，HTML属性是HTML标签的一部分。

1. 标签可以有属性，它包含了额外的信息，属性的值一般要在引号中
2. 标签可以拥有一个或多个属性，也可以没有属性；
3. 属性一般由属性名和值成对出现。



# HTML检查元素

1. 打开需要分析的网页
2. 右键要分析的元素
3. 点击检查
4. 此时对应的元素代码就会在右边栏显示



## 元素定位失败的原因

---

1. 元素没有加载完成
2. 元素不可用、读、见
3. 动态属性 动态的Div类

# 元素定位失败的原因

## 使用requests获取网页源代码

```
import requests
import re
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from bs4 import BeautifulSoup
x = requests.get('https://kcb.sse.com.cn/renewal/')
x.encoding = x.apparent_encoding
texts = BeautifulSoup(x.text, 'html5lib')
```

✓ 0.3s

## 结果

```
re.findall('无锡硅动力微电子股份有限公司', str(texts))
✓ 0.1s
[]

re.findall('无锡硅动力微电子股份有限公司', str(page))💡
✓ 0.1s
['无锡硅动力微电子股份有限公司']
```

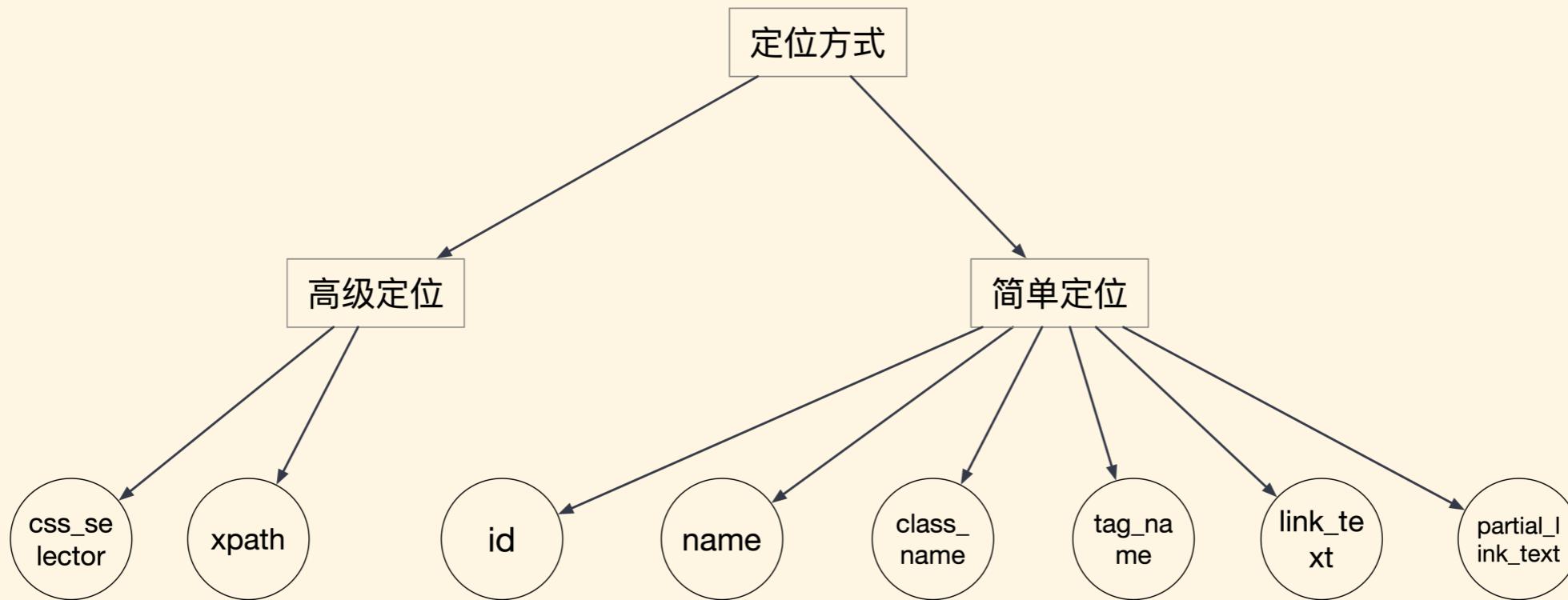
## 使用ChromeDriver获取网页源代码

```
chrome_options = Options()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--disable-gpu')
driver = webdriver.Chrome(options=chrome_options)
driver.get("https://kcb.sse.com.cn/renewal/")
page = BeautifulSoup(driver.page_source, 'html5lib').body
```

✓ 2.9s

序号 发行人全称 审核状态 注册地 证监会行业 保荐机构						
1	无锡硅动力 微电子股份 有限公司	已问询	江苏	计算机、 通信和其 他电子设 备制造业	安信证 券股份 有限公 司	
2	深圳威迈斯 新能源股份 有限公司	已问询	广东	汽车制造 业	东方证 券承销 保荐有 限公司	

# 定位方法



`find_element & find_elements`

(method)`find_elements:(by:str By.ID,value:Any | None,None)->List [WebElement]`

(method)`find_element:(by:str By.ID,value:Any | None, None)->WebElement`

**\*\*\*提前注意的点\*\*\***

1. 在Selenium3.\* 中，允许使用`find_element_by_XXXXXX`的函数实现HTML中元素的查找，但是 Selenium4.0 中已经完全废除了该类函数，即使依然使用 3.\* 版本，依旧建议使用 `find_element` 方法。
2. 并非只有driver实例可以有`find_element`方法，WebElement也能有此方法。

# 简单定位

## 1. 通过ID定位

HTML检查元素：

```
<form name="f" id="form" action="/s" class="fm has-soutu has-voice" onsubmit='<br/>    <span id="s_kw_wrap" class="bg s_ipt_wr new-pmd quickdelete-wrap"><br/>        <span class="ipt_rec" style="display: block;"></span><br/>        <span class="soutu-btn"></span><br/>        <input type="text" class="s_ipt" name="wd" id="kw" maxlength="100" autocomplete="off" value="复旦大学"/><br/>        <i class="c-icon quickdelete c-color-gray2" title="清空">清空</i><br/>        <i class="quickdelete-line"></i><br/>        <span class="soutu-hover-tip" style="display: none;">按图片搜索</span><br/>        <span class="voice-hover" style="display: none;">按语音搜索</span><br/>    </span>
```

通过ID（唯一）找到输入框：

```
## 加载网页
driver.get("http://www.baidu.com")
## 定位元素，设置值
driver.find_element(By.ID,"kw")[0].send_keys("复旦大学")
## 展示
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
```

✓ 0.7s

效果：



# 简单定位

## 2. 通过name定位

HTML检查元素：

```
<form name="f" id="form" action="/s" class="fm has-soutu has-voice" onsubmit="<br/><span id="s_kw_wrap" class="bg s_ipt_wr new-pmd quickdelete-wrap"><br/><span class="ipt_rec" style="display: block;"></span><br/><span class="soutu-btn"></span><br/><input type="text" class="s_ipt" name="wd" id="kw" maxlength="100" autocomplete="off" /><br/><i class="c-icon quickdelete c-color-gray2" title="清空">清空</i><br/><i class="quickdelete-line"></i><br/><span class="soutu-hover-tip" style="display: none;">按图片搜索</span><br/><span class="voice-hover" style="display: none;">按语音搜索</span><br/></span>
```

通过name找到输入框：

```
## 加载网页  
driver.get("http://www.baidu.com")  
## 定位元素，设置值  
driver.find_element(By.NAME,"wd").send_keys("复旦大学")  
driver.save_screenshot('screenshot_google.png')  
Image("screenshot_google.png",width=800)
```

✓ 0.8s

效果：



# 简单定位

## 3. 通过class\_name定位

HTML检查元素：

```
<div id="s-top-left" class="s-top-left-new s-isindex-wrap">
    <a href="http://news.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">新闻</a>
    <a href="https://www.hao123.com?src=from_pc_logon" target="_blank" class="mnav c-font-normal hao123">hao123</a>
    <a href="http://map.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">地图</a>
    <a href="http://tieba.baidu.com/" target="_blank" class="mnav c-font-normal c-color-t">贴吧</a>
    <a href="https://haokan.baidu.com/?sfrom=baidu-top" target="_blank" class="mnav c-font-normal"></a>
    <a href="http://image.baidu.com/" target="_blank" class="mnav c-font-normal c-color-t"> 图片</a>
    <a href="https://pan.baidu.com?from=1026962h" target="_blank" class="mnav c-font-normal c-co
```



```
driver.get("http://www.baidu.com")
elements = driver.find_elements(By.CLASS_NAME, "mnav c-font-normal c-color-t")
elements
```

✓ 0.8s

通过class\_name查找：

OR

新闻  
hao123  
地图  
贴吧  
视频  
图片  
网盘



```
driver.get("http://www.baidu.com")
elements = driver.find_elements(By.CLASS_NAME, "mnav.c-font-normal.c-color-t")
for element in elements:
    print(element.text)
elements[0].click()
windows = driver.window_handles
driver.switch_to.window(windows[-1])
driver.save_screenshot('screenshot_google.png')
```

True

✓ 2.4s

# 简单定位

## 4. 通过Link Text定位

具有 href 属性的内容，

一些可以点击的链接跳转上面的文字，就是 link text

HTML 检查元素：

```
▼ <div id="s-top-left" class="s-top-left-new s-isindex-wrap">
  <a href="http://news.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">新闻</a>
  <a href="https://www.hao123.com?src=from_pc_logon" target="_blank" class="mnav c-font-normal hao123</a>
  <a href="http://map.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">地图</a>
  <a href="http://tieba.baidu.com/" target="_blank" class="mnav c-font-normal c-color-t">贴吧</a>
  <a href="https://haokan.baidu.com/?sfrom=baidu-top" target="_blank" class="mnav c-font-normal c-color-t">好客</a>
  <a href="http://image.baidu.com/" target="_blank" class="mnav c-font-normal c-color-t">图片</a>
  <a href="https://pan.baidu.com?from=1026962h" target="_blank" class="mnav c-font-normal c-color-t">网盘</a>
```

通过 Link Text 查找：

```
## 加载网页
driver.get("http://www.baidu.com")
## 定位元素，点击
driver.find_element(By.LINK_TEXT,"新闻").click()
windows = driver.window_handles
driver.switch_to.window(windows[-1])
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
```

✓ 2.7s



效果：

# 简单定位

## 5. 通过Partial Link Text定位

HTML检查元素：

```
<div id="s-top-left" class="s-top-left-new s-isindex-wrap">
    <a href="http://news.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">新闻</a>
    <a href="https://www.hao123.com?src=from_pc_logon" target="_blank" class="mnav c-font-normal hao123</a>
    <a href="http://map.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">地图</a>
    <a href="http://tieba.baidu.com/" target="_blank" class="mnav c-font-normal c-color-t">贴吧</a>
    <a href="https://haokan.baidu.com/?sfrom=baidu-top" target="_blank" class="mnav c-font-normal</a>
    <a href="http://image.baidu.com/" target="_blank" class="mnav c-font-normal c-color-t"> 图片</a>
    <a href="https://pan.baidu.com?from=1026962h" target="_blank" class="mnav c-font-normal c-co
```

通过Partial  
Link Text查找：

```
## 加载网页
driver.get("http://www.baidu.com")
## 定位元素，点击
driver.find_element(By.PARTIAL_LINK_TEXT,"新").click()
windows = driver.window_handles
driver.switch_to.window(windows[-1])
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
```

✓ 2.3s

效果：



# 简单定位

## 6. 通过tag name定位

HTML检查元素：

```
<div data-sg-type="combobox" class="search-suggest-combobox">
  <input id="q" name="q" aria-label="请输入搜索文字" accesskey="s" autofocus="true">
</div>
<input type="hidden" name="commend" value="all">
<input type="hidden" name="ssid" value="s5-e" autocomplete="off">
```

通过tag name查找

```
加载网页
driver.get('https://www.taobao.com/')
## 定位元素，点击
driver.find_element(By.TAG_NAME,"input").send_keys('123456')
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
```

✓ 3.2s

效果：



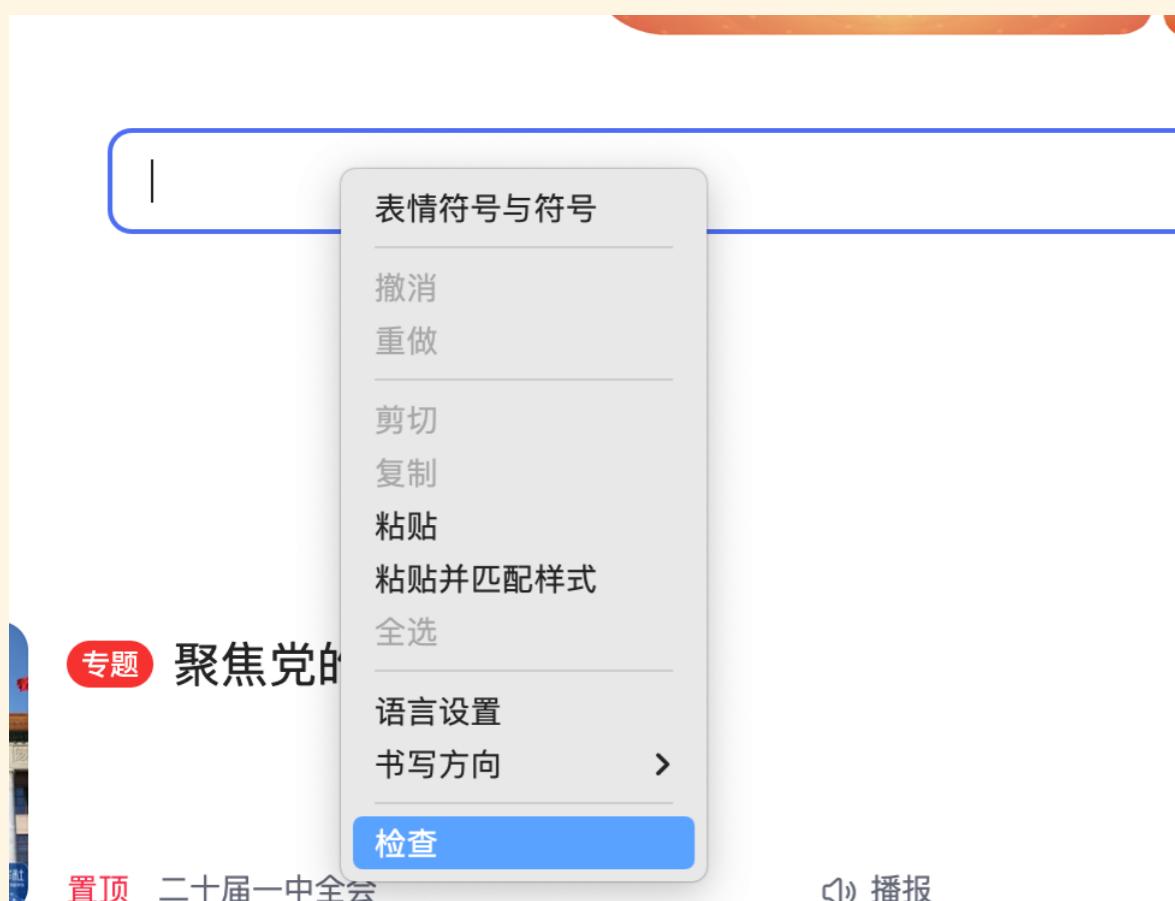
# 高级定位

## 1. 通过css\_selector定位

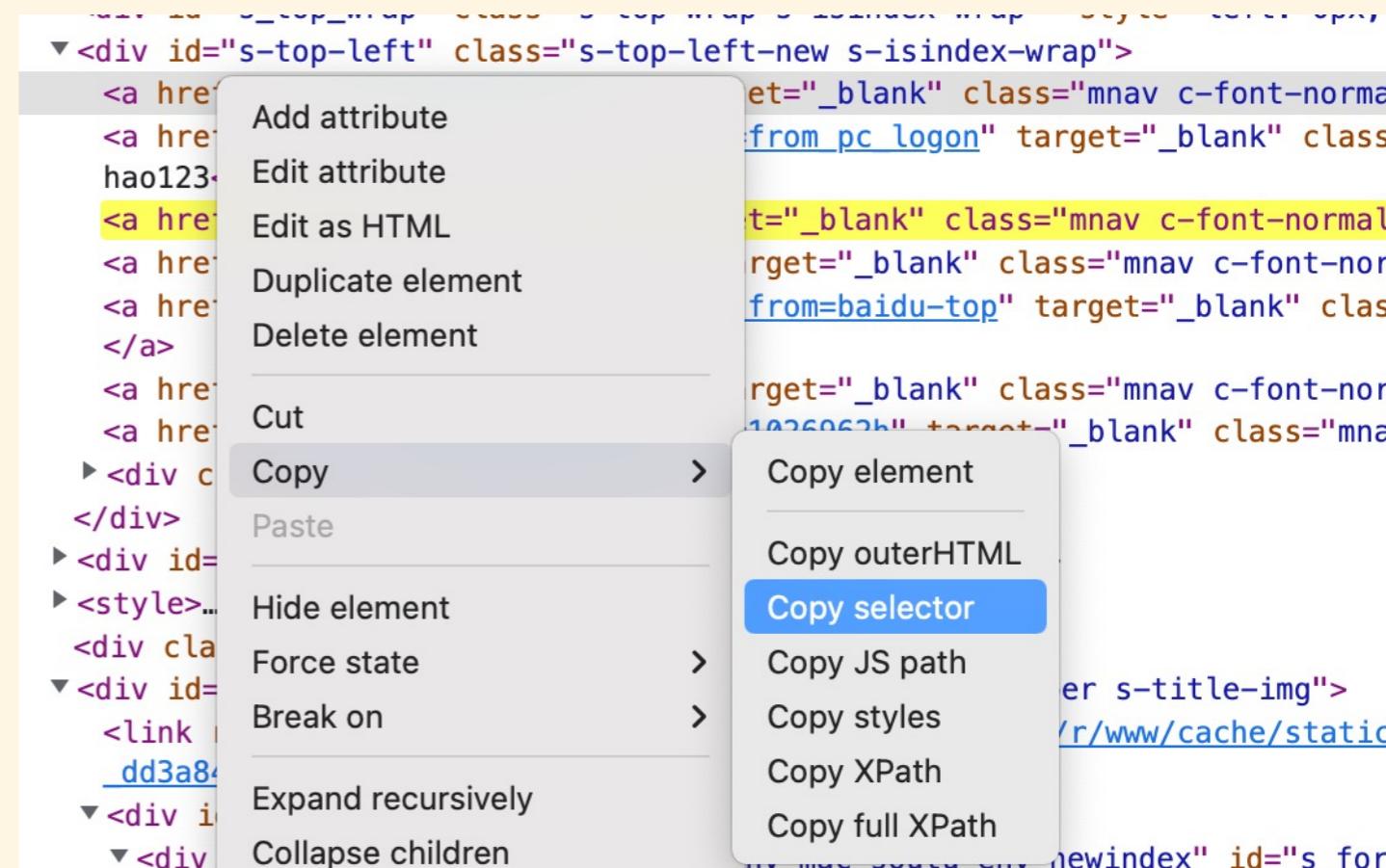
CSS (Cascading Style Sheets)是一种样式表语言，是所有浏览器内置的，用于描述以HTML或XML编写的文档的外观和样式。CSS Selector用于选择样式化的元素。

得到Selector的简易方法：

右键想要定位的元素 > 检查



右键 > Copy > Copy selector



# 通过css\_selector定位

---

1) 根据 `id` 属性选择元素的语法是，在 `id` 前面加上一个 "#" 号： `#id值`

```
driver.find_elements(By.ID,"kw") [0].send_keys("复旦大学")
```

```
driver.find_elements(By.CSS_SELECTOR,"#kw") [0].send_keys("复旦大学")
```

2) 根据 `class` 属性选择元素的语法是，在 `class` 值前面加上一个"."： `.class值`

```
elements = driver.find_elements(By.CLASS_NAME, "mnav.c-font-normal.c-color-t")
```

```
elements = driver.find_elements(By.CSS_SELECTOR, ".mnav.c-font-normal.c-color-t")
```

3) 根据标签选择元素的语法是： `标签`

```
driver.find_element(By.TAG_NAME,"input").send_keys('123456')
```

```
driver.find_element(By.CSS_SELECTOR , "input").send_keys('123456')
```

# 通过css\_selector定位

## 4) 根据子元素定位

如果 元素2 是 元素1 的 直接子元素， 使用 ">" 作连接符，  
css selector 选择直接子元素的语法是：

/\* 一个层级 \*/

元素1 > 元素2

/\* 多个层级 \*/

元素1 > 元素2 > 元素3 > 元素4

```
▼ <form data-sg-type="form" target="_top" action="//s.taobao.com/search" name="q" class="search-panel-focused">
  <i class="search-split"></i>
  ▶ <div class="search-button">...</div>
  <div data-sg-type="placeholder" style="left: 11px;">小白鞋</div>
  ▼ <div data-sg-type="combobox" class="search-suggest-combobox">
    <input id="q" name="q" aria-label="请输入搜索文字" accesskey="s" autofocus="" aria-haspopup="true" aria-combobox="list" role="combobox" x-webkit-gra...
  </div>
```

```
## 加载网页
driver.get('https://www.taobao.com/')
## 定位元素，点击
driver.find_element(By.CSS_SELECTOR , "form>div>input").send_keys('123456')
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
```

✓ 2.1s

# 通过css\_selector定位

## 5) 根据后代元素定位

如果元素2是元素1的后代元素（后代元素包含子元素），通过空格隔开元素即可，css selector 选择后代元素的语法是：

/\* 一个层级 \*/

元素1 元素2

/\* 多个层级 \*/

元素1 元素2 元素3 元素4

```
▼ <form data-sg-type="form" target="_top" action="//s.taobao.com/search" name="q" class="search-panel-focused">
  <i class="search-split"></i>
  ▶ <div class="search-button">...</div>
  <div data-sg-type="placeholder" style="left: 11px;">小白鞋</div>
  ▼ <div data-sg-type="combobox" class="search-suggest-combobox">
    <input id="q" name="q" aria-label="请输入搜索文字" accesskey="s" autofocus="" aria-haspopup="true" aria-combobox="list" role="combobox" x-webkit-gra...
  </div>
```

```
## 加载网页
driver.get('https://www.taobao.com/')
## 定位元素，点击
driver.find_element(By.CSS_SELECTOR , "form input").send_keys('123456')
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
```

✓ 1.8s

# 通过css\_selector定位

## 6) 根据兄弟(弟弟)元素定位

```
▼ <div id="head" class="...>
  ▶ <div id="s_top_wrap" class="s-top-wrap s-isindex-wrap " style="left: 0px;">...</div>
  ▼ <div id="s-top-left" class="s-top-left-new s-isindex-wrap">
    <a href="http://news.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">新闻</a> == $0
    <a href="https://www.hao123.com?src=from_pc_logon" target="_blank" class="mnav c-font-normal c-color-t">hao123</a>
    <a href="http://map.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">地图</a>
    <a href="http://tieba.baidu.com/" target="_blank" class="mnav c-font-normal c-color-t">贴吧</a>
    <a href="https://haokan.baidu.com/?sfrom=baidu-top" target="_blank" class="mnav c-font-normal c-color-t">
```

```
## 加载网页
driver.get("http://www.baidu.com")
## 定位元素，点击
driver.find_element(By.CSS_SELECTOR,'body>div #s-top-left>a+a').click()
windows = driver.window_handles
driver.switch_to.window(windows[-1])
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
```

✓ 2.7s



如果使用find\_elements()会是怎样的效果？

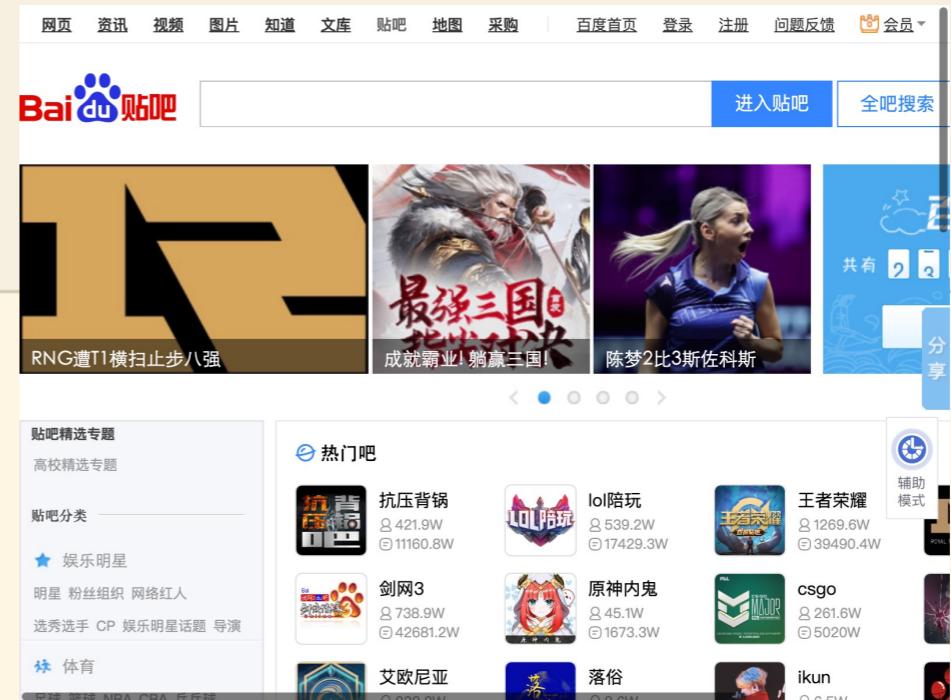
# 通过css\_selector定位

## 7) 属性定位

支持通过任何属性来选择元素，语法是用一个方括号[]；标签[属性名1 = 属性值1] [属性名2 = 属性值2]..

```
▼ <div id="head" class>
  ▶ <div id="s_top_wrap" class="s-top-wrap s-isindex-wrap " style="left: 0px;">...</div>
  ▼ <div id="s-top-left" class="s-top-left-new s-isindex-wrap">
    <a href="http://news.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">新闻</a> == $0
    <a href="https://www.hao123.com?src=from_pc_logon" target="_blank" class="mnav c-font-normal c-color-t">
      hao123</a>
    <a href="http://map.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">地图</a>
    <a href="http://tieba.baidu.com/" target="_blank" class="mnav c-font-normal c-color-t">贴吧</a>
    <a href="https://haokan.baidu.com/?sfrom=baidu-top" target="_blank" class="mnav c-font-normal c-color-t">
```

```
## 加载网页
driver.get("http://www.baidu.com")
💡 定位元素，点击
driver.find_element(By.CSS_SELECTOR,'body>div>div[id="s-top-left"]>a[href="http://tieba.baidu.com/"]').click()
windows = driver.window_handles
driver.switch_to.window(windows[-1])
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
✓ 4.4s
```



# 通过css\_selector定位

---

## 8) 根据次序选择子节点

### 1. 父元素的第n个子节点

使用 `:nth-child(n)`，可以指定选择父元素的第几个子节点。

### 2. 父元素的倒数第n个子节点

使用 `:nth-last-child(n)`，可以倒过来，选择的是父元素的倒数第几个子节点。

### 3. 父元素的第几个某类型的子节点

使用 `:nth-of-type(n)`，可以指定选择的元素是父元素的第几个某类型的子节点。

### 4. 父元素的倒数第几个某类型的子节点

使用 `:nth-last-of-type(n)`，可以倒过来，选择父元素的倒数第几个某类型的子节点。

### 5. 奇数节点和偶数节点

如果要选择的是父元素的偶数节点，使用 `:nth-child(even)`

如果要选择的是父元素的奇数节点，使用 `:nth-child(odd)`

如果要选择的是父元素的某类型偶数节点，使用 `:nth-of-type(even)`

# 通过css\_selector定位

## 8) 根据次序选择子节点

```
▼ <div id="head" class="...>
  ▶ <div id="s_top_wrap" class="s-top-wrap s-isindex-wrap " style="left: 0px;">...</div>
  ▼ <div id="s-top-left" class="s-top-left-new s-isindex-wrap">
    <a href="http://news.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">新闻</a> == $0
    <a href="https://www.hao123.com?src=from_pc_logon" target="_blank" class="mnav c-font-normal c-color-t">hao123</a>
    <a href="http://map.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">地图</a>
    <a href="http://tieba.baidu.com/" target="_blank" class="mnav c-font-normal c-color-t">贴吧</a>
    <a href="https://haokan.baidu.com/?sfrom=baidu-top" target="_blank" class="mnav c-font-normal c-color-t">
```

```
## 加载网页
driver.get("http://www.baidu.com")
## 定位元素，点击
driver.find_element(By.CSS_SELECTOR,'body>div div[id="s-top-left"]>a:nth-of-type(even)').click()
windows = driver.window_handles
driver.switch_to.window(windows[-1])
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
```

✓ 7.7s



# 通过css\_selector定位

## 8) 根据次序选择子节点

注意：nth-of-type(even)会先寻找出该路径下所有type类型节点，而后再去取出偶数节点，nth-child则是会取出所有节点，而后判断是否某个节点既符合类型要求，又是偶数节点。

```
<section>
    <div>我是一个普通的div标签</div>
    <p>我是第1个p标签</p>
    <p>我是第2个p标签</p>  <!-- 希望这个变红 -->
</section>
```

```
p:nth-child(2) { color: red; }
```

```
p:nth-of-type(2) { color: red; }
```

我是一个普通的div标签

我是第1个p标签

我是第2个p标签

我是一个普通的div标签

我是第1个p标签

我是第2个p标签

# 高级定位

## 2. 通过XPath定位

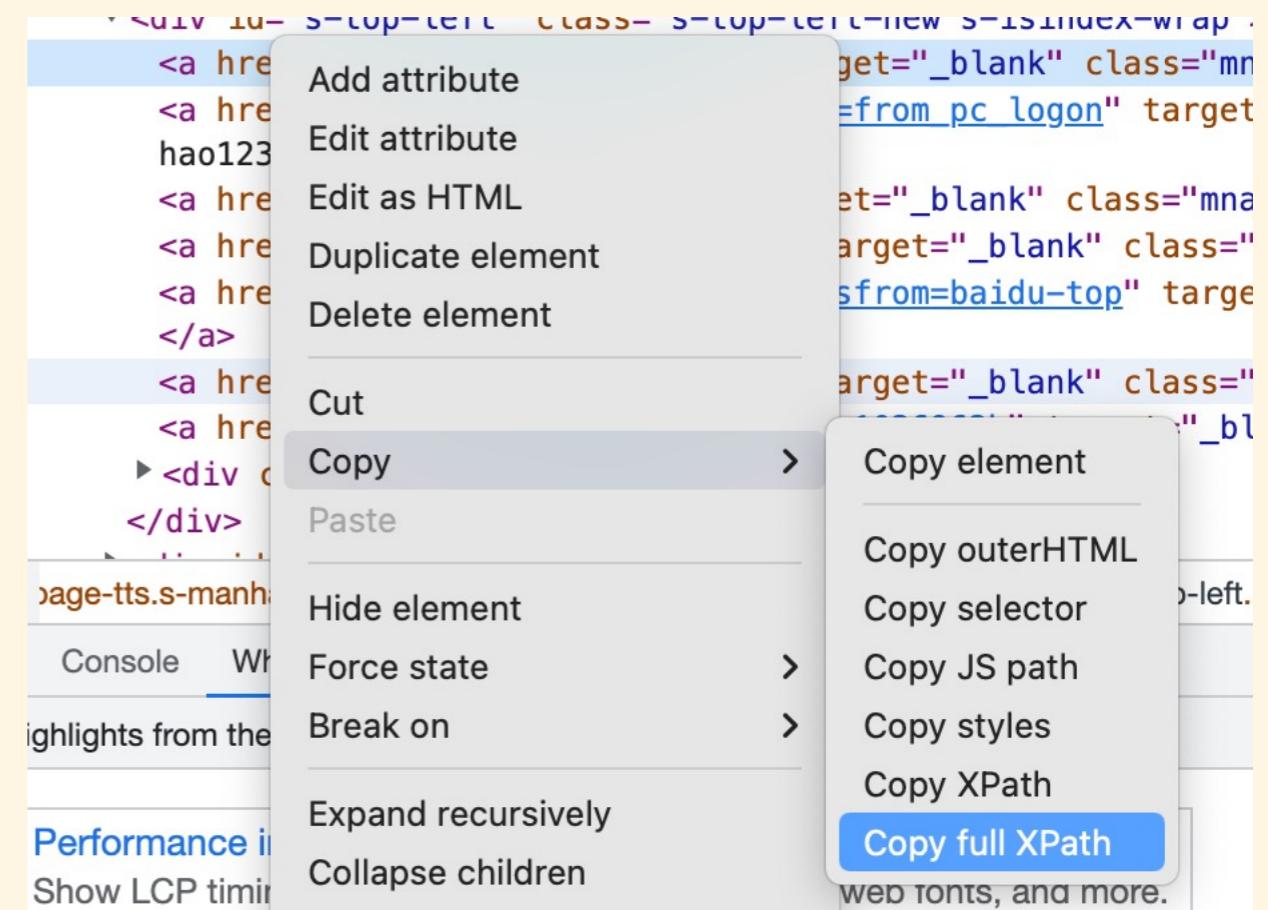
“XPath即为XML路径语言（ XML Path Language ），它是一种用来确定XML文档中某部分位置的语言。 XPath是相对路径， full XPath是绝对路径。

得到XPath的简易方法：

右键想要定位的元素 > 检查



右键 > Copy > Copy XPath/ Copy Full XPath



# 通过XPath定位

## 1) 根据子元素定位

如果某个XPath表达式可以定位到多个元素标签，可以通过(**xpath:表达式**)**[index]**定位第**index**个元素，**index**索引从1开始。

```
/* 一个层级 */
//元素1 / 元素2 / child::元素3
/* 多个层级 */
//元素1 / 元素2 / 元素3 / 元素4
```

```
<html class="sui-componentWrap">
  > <head>...</head>
  > <body class="open-homepage-tts s-manhattan-index" style="background-color: #f0f0f0; font-family: 'Microsoft YaHei', 'SimHei', sans-serif; margin: 0; padding: 0; width: 100%; height: 100%;">
    >   <a id="aging-total-page" role="pagedescription" aria-label="欢迎进入 百度一下，你就知道，盲人用户进入读屏幕模式请按快捷键Ctrl加Alt加R；阅读详细操作说明请按快捷键Ctrl加Alt加问号键。" tabindex="0" href="javascript:void(0)"></a>
    >   <script>...</script>
    >   <textarea id="s_is_result_css" style="display:none;">...</textarea>
    >   <textarea id="s_index_off_css" style="display:none;">...</textarea>
    >   <div id="wrapper" class="wrapper_new">
      >     <div class="s-skin-container s-isindex-wrap"></div>
      >     <div id="head" class="s-top-wrap s-isindex-wrap" style="left: 0px; position: absolute; top: 0px; width: 100%; height: 100%; z-index: 1000; background-color: #fff; border-bottom: 1px solid #ccc; transition: all 0.3s ease; ">
        >       <div id="s_top_wrap" class="s-top-wrap s-isindex-wrap" style="left: 0px; position: absolute; top: 0px; width: 100%; height: 100%; z-index: 1000; background-color: #fff; border-bottom: 1px solid #ccc; transition: all 0.3s ease; ">
          >         <div id="s-top-left" class="s-top-left-new s-isindex-wrap">
            >           <a href="http://news.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">新闻</a> == $0
            >           <a href="https://www.hao123.com?src=from_pc_logon" target="_blank" class="mnav c-font-normal c-color-t">好搜</a> == $1
          >         </div>
        >       </div>
      >     </div>
    >   </div>
  > </body>
</html>
```

```
## 加载网页
driver.get("http://www.baidu.com")
## 定位元素，点击
driver.find_element(By.XPATH,"/html/body/div[1]/div[1]/div[3]/a[1]").click()
windows = driver.window_handles
driver.switch_to.window(windows[-1])
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
```

# 通过XPath定位

## 2) 根据后代元素定位

```
/* 一个层级 */  
//元素1 // 元素2 /descendant::元素3  
/* 多个层级 */  
//元素1 // 元素2 // 元素3 // 元素4
```

```
▼<body class="open-homepage-tts s-manhattan-index" style>  
  <a id="aging-total-page" role="pagedescription" aria-label="欢迎进入 百度一下，你就知道，盲人用户进入读屏幕模式时  
    可以按Alt加R；阅读详细操作说明请按快捷键Ctrl加Alt加问号键。" tabindex="0" href="javascript:void(0)"></a>  
  ▶<script>...</script>  
  ▶<textarea id="s_is_result_css" style="display:none;">...</textarea>  
  ▶<textarea id="s_index_off_css" style="display:none;">...</textarea>  
  ▼<div id="wrapper" class="wrapper_new">  
    <div class="s-skin-container s-isindex-wrap"></div>  
    ▼<div id="head" class>  
      ▶<div id="s_top_wrap" class="s-top-wrap s-isindex-wrap " style="left: 0px;">...</div>  
      ▼<div id="s-top-left" class="s-top-left-new s-isindex-wrap">  
        <a href="http://news.baidu.com" target="_blank" class="mnav c-font-normal c-color-t">新闻</a> == $0  
        <a href="https://www.hao123.com?src=from_pc_logon" target="_blank" class="mnav c-font-normal c-color-t">好搜</a> == $1  
      
```

```
## 加载网页  
driver.get("http://www.baidu.com")  
## 定位元素，点击  
driver.find_element(By.XPATH,"//body//div[3]/a[1]").click()  
windows = driver.window_handles  
driver.switch_to.window(windows[-1])  
driver.save_screenshot('screenshot_google.png')  
Image("screenshot_google.png",width=800)
```

# 通过XPath定位

## 3) 根据兄弟(弟弟)元素定位

使用"/following-sibling::element"或者"/preceding-sibling::element"

```
## 加载网页
driver.get("http://www.baidu.com")
## 定位元素，点击
driver.find_element(By.XPATH,"//body//div[3]/a[1]/following-sibling::a").click()
windows = driver.window_handles
driver.switch_to.window(windows[-1])
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
```



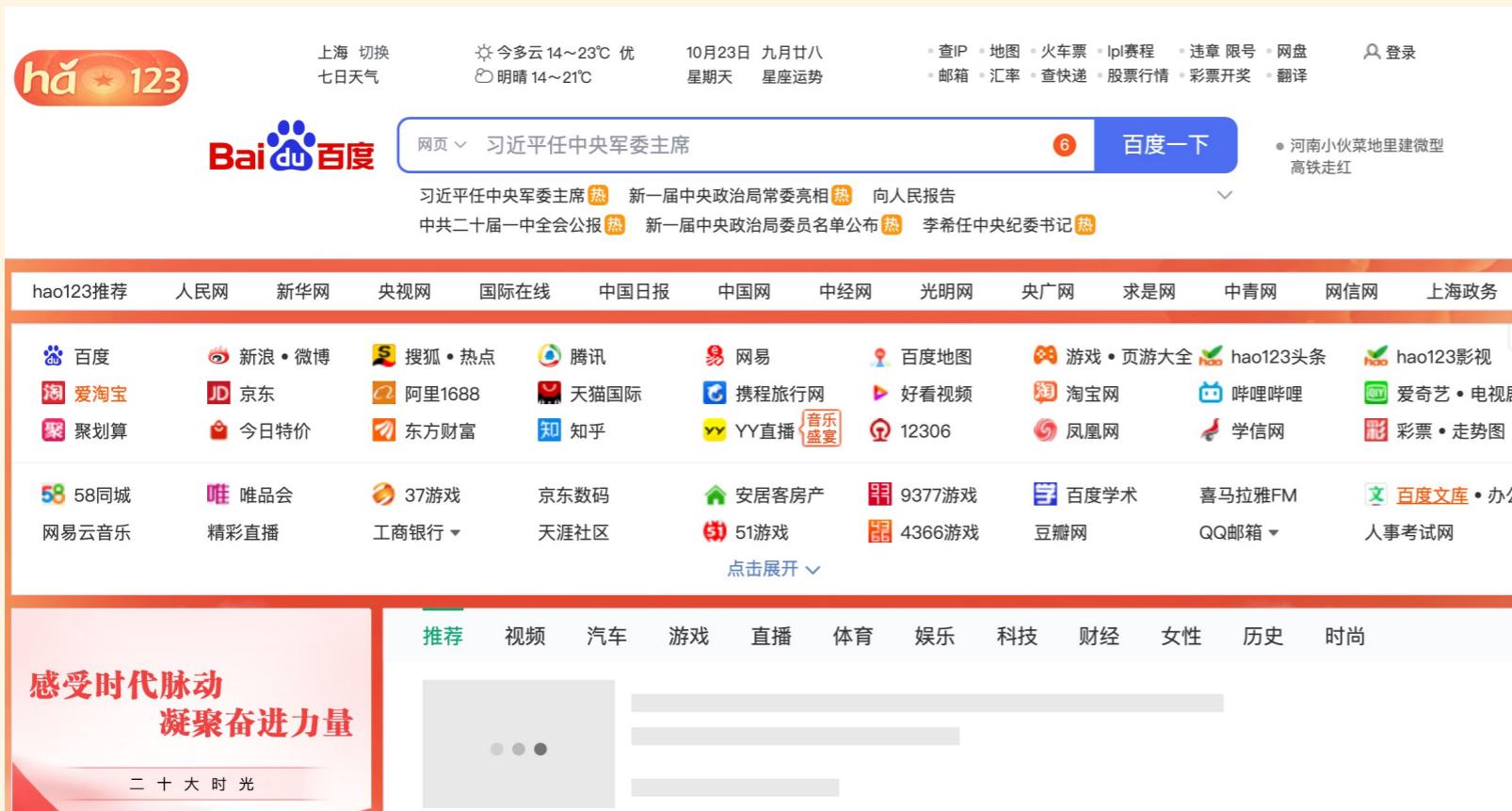
## 通过XPath定位

#### 4) 根据前辈元素定位

使用".."，"/parent::element"或者"/ancestor::element"

```
## 加载网页
driver.get("http://www.baidu.com")
## 定位元素，点击
driver.find_element(By.XPATH,"//body//div[3]/a[2]/..../a[2]").click()
windows = driver.window_handles
driver.switch_to.window(windows [-1])
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
```

✓ 4.7s



# 通过Xpath定位

## 5) 属性定位

支持通过任何属性来选择元素，语法是用一个方括号[@属性名=属性值]

```
## 加载网页
driver.get("http://www.baidu.com")
# 定位元素，点击
driver.find_element(By.XPATH,"//body//div[3]/a[@href='http://news.baidu.com']").click()
windows = driver.window_handles
driver.switch_to.window(windows[-1])
driver.save_screenshot('screenshot_google.png')
Image("screenshot_google.png",width=800)
```

## 6) 逻辑元素

XPath可以使用‘and’ 或者 ‘or’ 连接两个属性

= 匹配前缀，\$= 匹配后缀，\*= 包含某个字符串

```
driver.find_element(By.XPATH,"//body//div[3]/a[@target='_blank' and @class='mnav c-font-normal c-color-t'][3]").click()
```

# 通过Xpath定位

---

## 7) Xpath Text 方法

使用文本内容完全匹配

```
Xpath = //*[text()='text_value']  
driver.find_element(By.XPATH,"//body//div[3]/a[text()='新闻']").click()
```

## 8) Xpath Starts-with 方法

```
Xpath = //*[@type, 'start_text']  
Xpath = //*[@name, 'start_text']  
Xpath = //*[@class, 'start_text']  
Xpath = //*[@id, 'start_text']  
Xpath = //*[@text(), 'start_text']  
Xpath = //*[@href, 'start_text']  
  
driver.find_element(By.XPATH,"//body//div[3]/a[starts-with(text(),'新')]").click()
```

## 9) Xpath Contains 方法

```
Xpath = //*[@type, 'partial_text']  
Xpath = //*[@name, 'partial_text']  
Xpath = //*[@class, 'partial_text']  
Xpath = //*[@id, 'partial_text']  
Xpath = //*[@text(), 'partial_text']  
Xpath = //*[@href, 'partial_text']  
  
driver.find_element(By.XPATH,"//body//div[3]/a[contains(text(),'新')]").click()
```

# css\_selector和Xpath的比较

---

1. XPath通过遍历的方式从XML文档中选择节点，CSS Selector是一种匹配模式定位，因此CSS Selector比 XPath 执行效率更高。
2. Xpath可以通过文本来定位，而CSS Selector不能。
3. Xpath可以通过子节点来定位父节点，CSS Selector是前向的，不能利用子节点定位父节点。
4. CSS Selector语法相比Xpath更加简洁

# 继续定位

注意到，我们分析网页的时候可能会去定位同一个目录下的多个元素，可以采用bs4+正则搜索的方式提取其中的信息。

当然也可以采取继续定位的方式，循环提取信息。

```
## 加载网页
driver.get("http://www.baidu.com")
## 定位元素，点击
info = driver.find_element(By.CSS_SELECTOR, "#s-top-left")
link_list = info.find_elements(By.CSS_SELECTOR, '.mnav.c-font-normal.c-color-t')
✓ 1.3s
```

```
for item in link_list:
    print(item.text)
✓ 0.4s
```

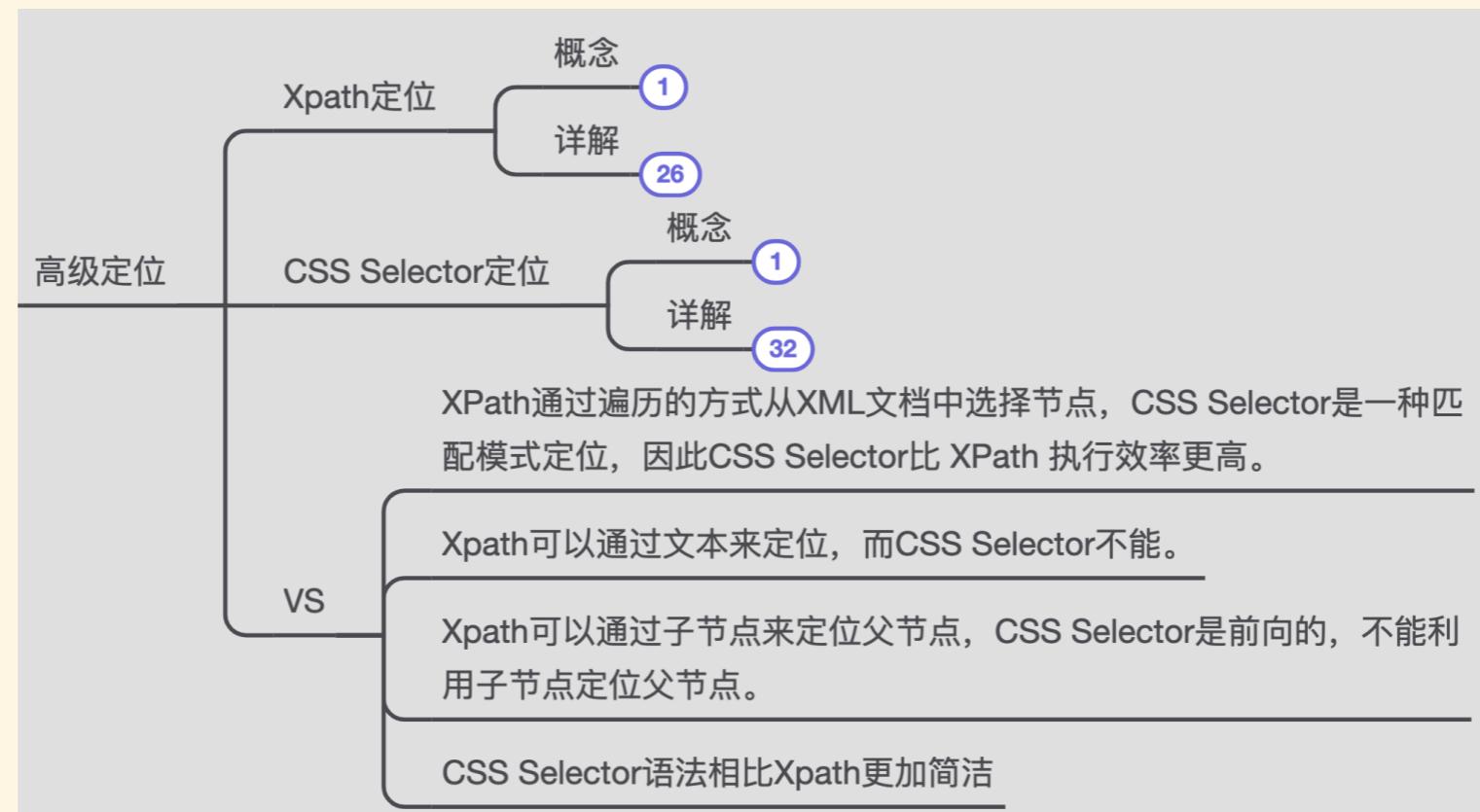
新闻  
hao123  
地图  
贴吧  
视频  
图片  
网盘

# 总结

我将所有的关于查找元素的大纲写在了[幕布上](#)，方便大家查找，对于一般的任务，简单定位就足够了

使用id属性定位	driver.find_element(By.ID,"kw")
使用class_name属性定位	driver.find_elements(By.CLASS_NAME, "mnav.c-font-normal.c-color-t")
使用name属性定位	driver.find_element(By.NAME,"wd")
使用Link Text链接文本定位	driver.find_element(By.LINK_TEXT,"新闻")
使用Partial Link Text部分链接文本定位	driver.find_element(By.PARTIAL_LINK_TEXT,"新")
使用 tag 标签定位	driver.find_element(By.TAG_NAME , "input")

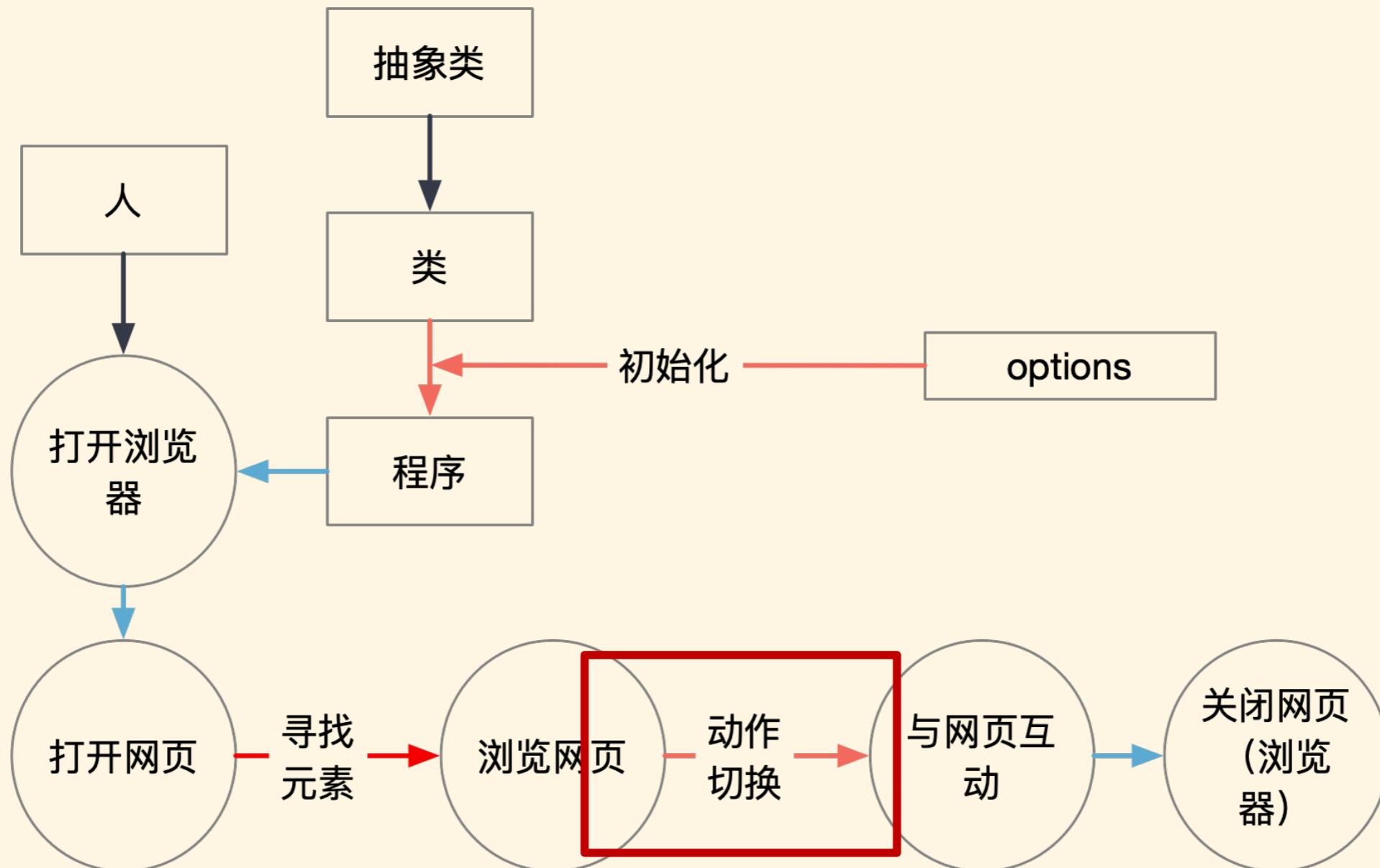
CSS\_Selector和Xpath多用于比较复杂的网页分析。



03

动作和等待

# PYTHON是面向对象的语言

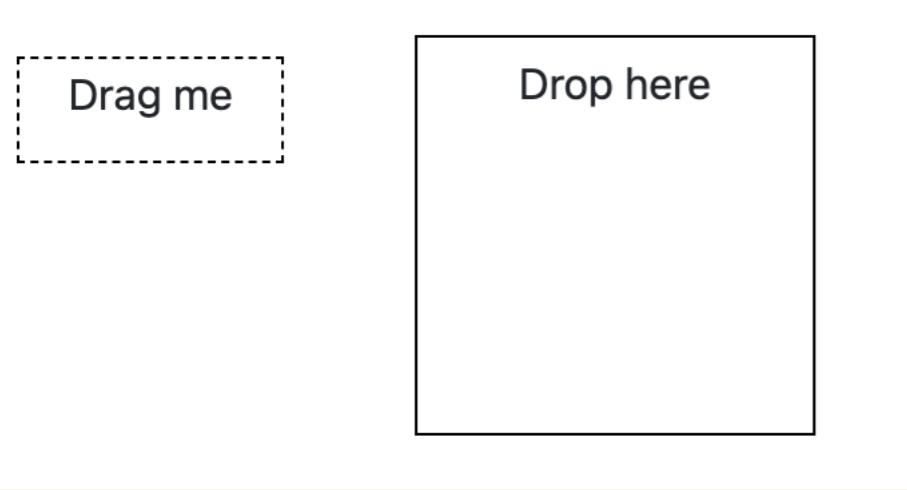


# 通过ActionChain传递动作

```
from selenium.webdriver.common.action_chains import ActionChains  
actChain=ActionChains(driver)
```

<code>click(on_element=None)</code>	鼠标左键单击
<code>click_and_hold(on_element=None)</code>	单击鼠标左键, 不松开
<code>context_click(on_element=None)</code>	单击鼠标右键
<code>double_click(on_element=None)</code>	双击鼠标左键
<code>drag_and_drop(source,target)</code>	拖拽到某个元素然后松开
<code>drag_and_drop_by_offset(source,xoffset,yoffset)</code>	拖拽到某个坐标然后松开
<code>key_down(value,element=None)</code>	按下键盘上的某个键
<code>key_up(value, element=None)</code>	松开键盘上的某个键
<code>move_by_offset(xoffset, yoffset)</code>	鼠标从当前位置移动到某个坐标
<code>move_to_element(to_element)</code>	鼠标移动到某个元素
<code>pause(seconds)</code>	暂停所有输入
<code>perform()</code>	执行所有操作
<code>reset_actions()</code>	结束已经存在的操作并重置
<code>release(on_element=None)</code>	在某个元素位置松开鼠标左键
<code>send_keys(*keys_to_send)</code>	发送某个键或者输入文本
<code>send_keys_to_element(element, *keys_to_send)</code>	发送某个键到指定元素

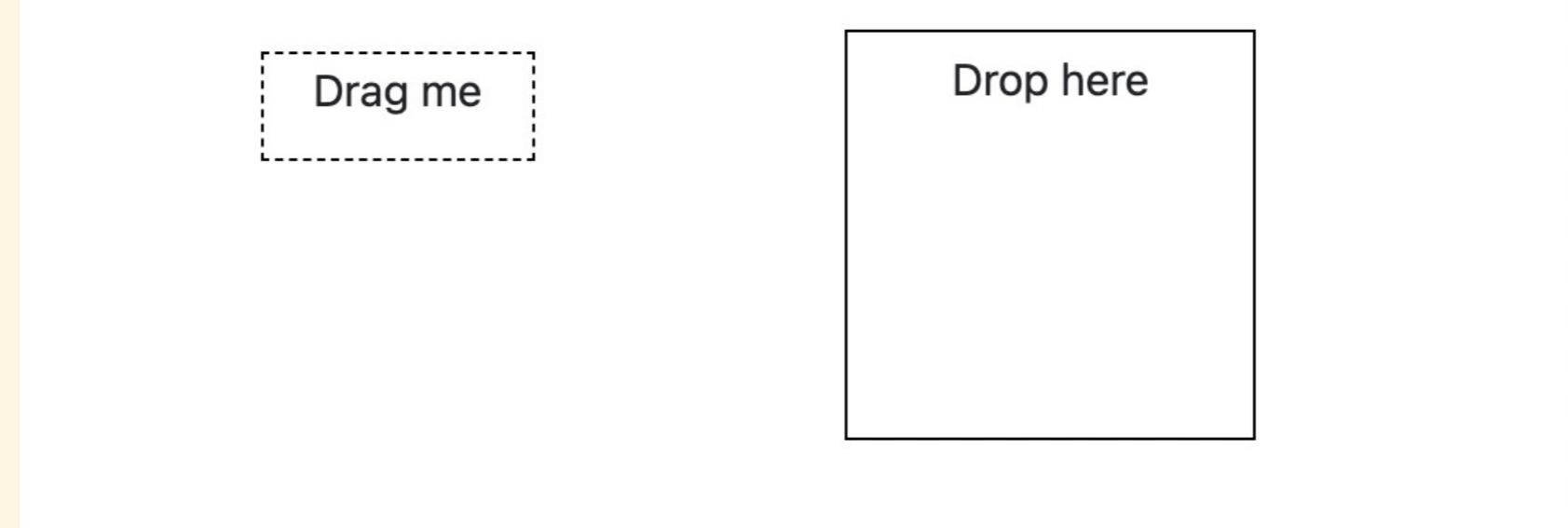
# 通过ActionChain传递动作



```
▼ <div class="simple-drop-container" id="simpleDropContainer"> flex
  <div id="draggable" class="drag-box mt-4 ui-draggable ui-draggabl
  le" style="position: relative;">Drag me</div>
  ▼ <div id="droppable" class="drop-box ui-droppable">
    <p>Drop here</p>
  </div>
</div>
```

```
driver.get('https://demoqa.com/droppable/')
drag_element = driver.find_element(By.CSS_SELECTOR,".simple-drop-container #draggable")
drop_element = driver.find_element(By.CSS_SELECTOR,".simple-drop-container .drop-box.ui-droppable")
actChain.click_and_hold(drag_element)
actChain.drag_and_drop(drag_element,drop_element)
actChain.perform()
```

Simple      Accept      Prevent Propogation      Revert Draggable



## 通过ActionChain传递动作（键盘特殊键）

<code>send_keys(Keys.BACK_SPACE)</code>	删除键
<code>send_keys(Keys.SPACE)</code>	空格键
<code>send_keys(Keys.TAB)</code>	缩进键
<code>send_keys(Keys.ESCAPE)</code>	ESC 键
<code>send_keys(Keys.ENTER)</code>	回车键
<code>send_keys(Keys.CONTROL, 'c')</code>	Control 键
<code>send_keys(Keys.F1)</code>	F 键

需要注意的是：有时候`send_keys(Keys.CONTROL,'c')`可能不能复制文字，可以尝试：

```
actions.key_down(Keys.CONTROL)
actions.send_keys("c")
actions.key_up(Keys.CONTROL)
```

# 窗口操作

## 网页前进/后退

```
driver.forward()      #前进  
driver.back()        # 后退
```

## 窗口切换

```
driver.current_window_handle ## 目前所在的窗口  
driver.window_handles -> List[Str] ## 所有窗口  
driver.switch_to.window ##切换窗口
```

```
driver = webdriver.Chrome()  
## 加载网页  
driver.get("http://www.baidu.com")  
window_1 = driver.current_window_handle  
## 定位元素，点击  
driver.find_element(By.XPATH,"//body//div[3]/a[starts-with(text(),'新')]").click()  
windows = driver.window_handles  
driver.switch_to.window(windows[-1]) ## 找到最新的窗口并切换  
driver.switch_to.window(window_1) ## 退到最早的窗口  
driver.save_screenshot('screenshot_google.png')  
Image("screenshot_google.png",width=800)
```

✓ 6.9s



# 下拉菜单

(class) Select(webElement: WebElement)

Constructor. A check is made that the given element is, indeed, a SELECT tag. If it is not, then an UnexpectedTagNameException is thrown.

```
▶ <select id="s1Id">...</select>
<br>
<br>
▶ <select id="s2Id">...</select>
<br>
<br>
▶ <select id="s3Id">...</select> == $0
.
.
```

```
s1 = Select(driver.find_element(By.ID, 's1Id')) # 实例化Select

s1.select_by_index(1) # 选择第二项选项: o1
time.sleep(3)
s1.select_by_value("o2") # 选择value="o2"的项
time.sleep(3)
s1.select_by_visible_text("o3") # 选择text="o3"的值, 即在下拉时我们可以看到的文本
time.sleep(3)
driver.save_screenshot('screenshot_google.png')
driver.close()
Image("screenshot_google.png",width=800)
```



# WEBELEMENT属性和方法

定位到元素后我们除了希望进行一些动作操作外，也希望能够获得该元素的其他属性以及包含的内容

序号	方法/属性	描述
1	WebElement.id	获取元素的标示 '1b2a928d-da26-45d6-83b9-08e3adf58bfe'
2	WebElement.size	获取元素的宽与高，返回一个字典 {'height': 77, 'width': 1118}
3	WebElement.rect	除了获取元素的宽与高，还获取元素的坐标 {'height': 77, 'width': 1118, 'x': 41, 'y': 1341.8515625}
4	WebElement.tag_name	获取元素的标签名称 'tr'
5	WebElement.text	获取元素的文本内容
6	WebElement.get_attribute( )	获取相关属性，比如class；可以通过innerHTML获取该项网页源代码
7	WebElement.find_element( )	定位相关元素

# Sleep强制等待和隐性等待

**Sleep()**：简单粗暴，根据网站的响应速度和自己的网速来设置合理的休眠时间

```
from selenium import webdriver
driver = webdriver.Chrome() #设置等待2秒钟
driver.get('http://www.baidu.com')
time.sleep(3)
driver.close()
```

**implicitly\_wait()**:设置一个最长等待时间，如果在规定时间内网页加载完成，则执行下一步，否则一直等到时间截止，然后执行下一步，超出设置的时长还没有定位到元素，则抛出异常。

```
from selenium import webdriver
driver = webdriver.Chrome() #设置等待2秒钟
driver.get('http://www.baidu.com')
driver.implicitly_wait(4)
driver.close()
```

# 显式等待

**WebDriverWait()** : 观察网页某个元素如果达到预期要求，则进行下一步，如果超出预期时间依然没有出现预期要求，则抛出异常

```
from selenium.webdriver.support import expected_conditions as EC  
from selenium.webdriver.support.ui import WebDriverWait
```

WebDriverWait(driver, timeout, poll\_frequency, ignored\_exceptions).until(可执行方法, 超时时返回的信息)

WebDriverWait(driver, timeout, poll\_frequency, ignored\_exceptions).until\_not(可执行方法, 超时时返回的信息)

driver: 传入WebDriver实例

timeout: 超时时间, 等待的最长时间 ( 同时要考虑隐性等待时间 )

poll\_frequency: 调用until或until\_not中的方法的间隔时间, 默认是0.5秒

ignored\_exceptions: 忽略的异常, 如果在调用until或until\_not的过程中抛出这个元组中的异常, 则不中断代码, 继续等待, 默认只有NoSuchElementException。

# 显式等待可执行方法

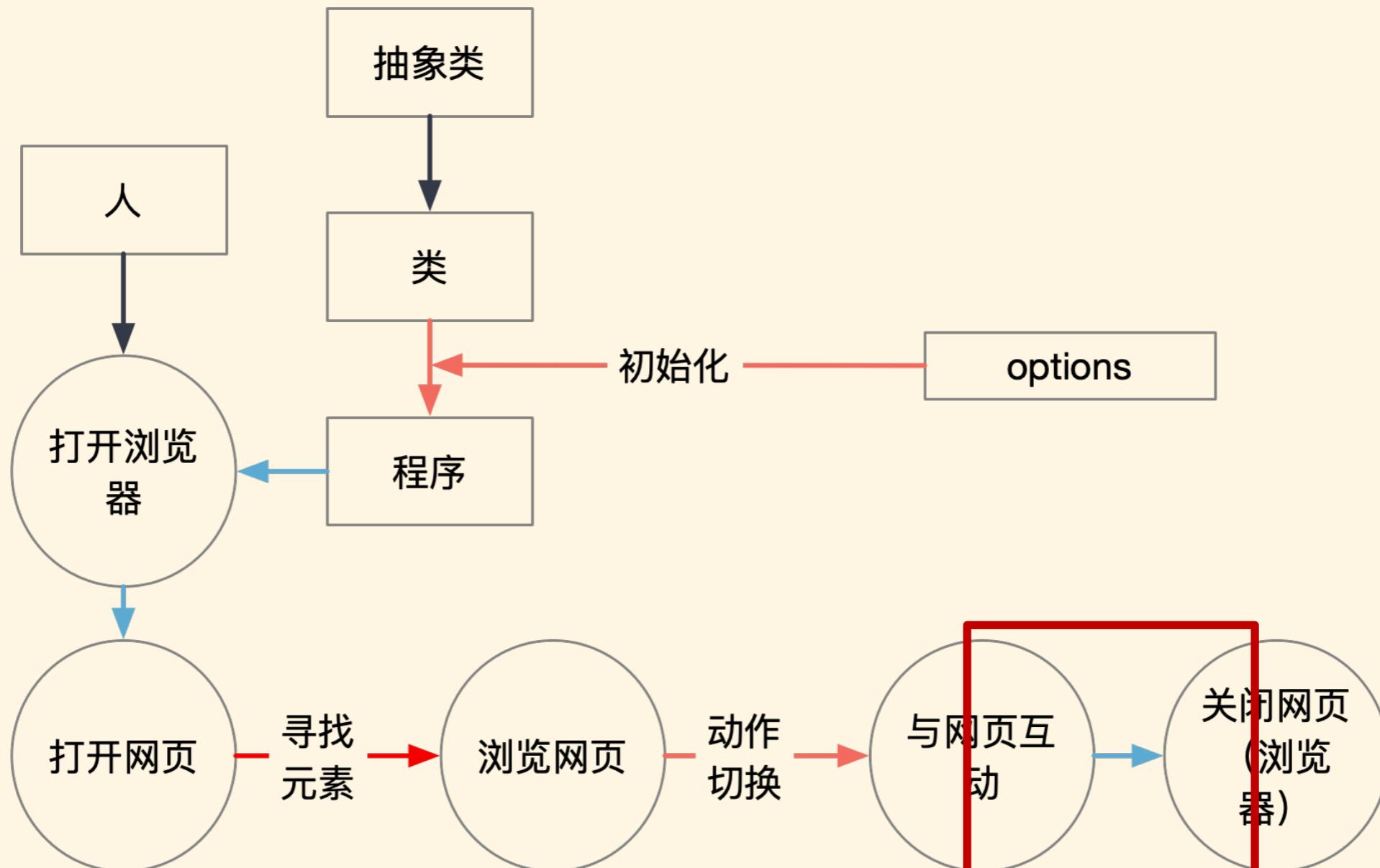
title_is	判断当前页面的title是否精确等于预期
title_contains	判断当前页面的title是否包含预期字符串
presence_of_element_located	判断某个元素是否被加到了dom树里，并不代表该元素一定可见
visibility_of_element_located	判断某个元素是否可见.可见代表元素非隐藏，并且元素的宽和高都不等于0
visibility_of	跟上面的方法做一样的事情，只是上面的方法要传入locator，这个方法直接传定位到的element就好了
presence_of_all_elements_located	判断是否至少有1个元素存在于dom树中。举个例子，如果页面上有n个元素的class都是'column-md-3'，那么只要有1个元素存在，这个方法就返回True
text_to_be_present_in_element	判断某个元素中的text是否包含了预期的字符串
text_to_be_present_in_element_value	判断某个元素中的value属性是否包含了预期的字符串
frame_to_be_available_and_switch_to_it	判断该frame是否可以switch进去，如果可以的话，返回True并且switch进去，否则返回False
invisibility_of_element_located	判断某个元素中是否不存在于dom树或不可见
element_to_be_clickable	判断某个元素中是否可见并且是enable的，这样的话才叫clickable
staleness_of	等某个元素从dom树中移除，注意，这个方法也是返回True或False
element_to_be_selected	判断某个元素是否被选中了,一般用在下拉列表
element_located_to_be_selected	跟上面的方法作用一样，只是上面的方法传入定位到的element，而这个方法传入locator
element_selection_state_to_be	判断某个元素的选中状态是否符合预期
element_located_selection_state_to_be	跟上面的方法作用一样，只是上面的方法传入定位到的element，而这个方法传入locator
alert_is_present	判断页面上是否存在alert

# 显式等待可执行方法

```
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait
```

```
driver = webdriver.Chrome()
driver.get('http://www.baidu.com')
try:
    element = WebDriverWait(driver,10).until(EC.presence_of_element_located((By.ID,'kw')))
    element.send_keys('123')
    driver.save_screenshot('screenshot_google.png')
except Exception as message:
    print('元素定位报错%s'%message)
finally:
    pass
Image("screenshot_google.png",width=800)
driver.close()
```

# PYTHON是面向对象的语言



# 关闭浏览器和网页

---

`driver.quit()` 和 `driver.close()`

`driver.quit()`关闭所有网页并关闭浏览器

`driver.close()`关闭当前网页

04

例子

# 自动平安复旦



The screenshot shows the Fudan Daily Health Report login interface. A yellow box highlights the password input field. The page includes a logo, a title '统一身份认证', and a notice about email封禁 lists. A context menu is open over the password field, listing options like '显示所有已保存的密码' (Show all saved passwords), '表情符号与符号' (Emojis and symbols), and various clipboard functions.

HTML DOM structure (partial):

```
<div class="IDCheckBack"></div>
<div class="header">...</div>
<div class="main">
  <div class="IDCheckMiddleCon">
    <div class="IDCheckLogin">
      <div class="IDCheckLoginLanguage">...</div>
      <div class="IDCheckLoginTitle">...</div>
      <div class="IDCheckLoginWarn">...</div>
    </div>
    ...
  </div>
</div>
```

Styles tab in developer tools:

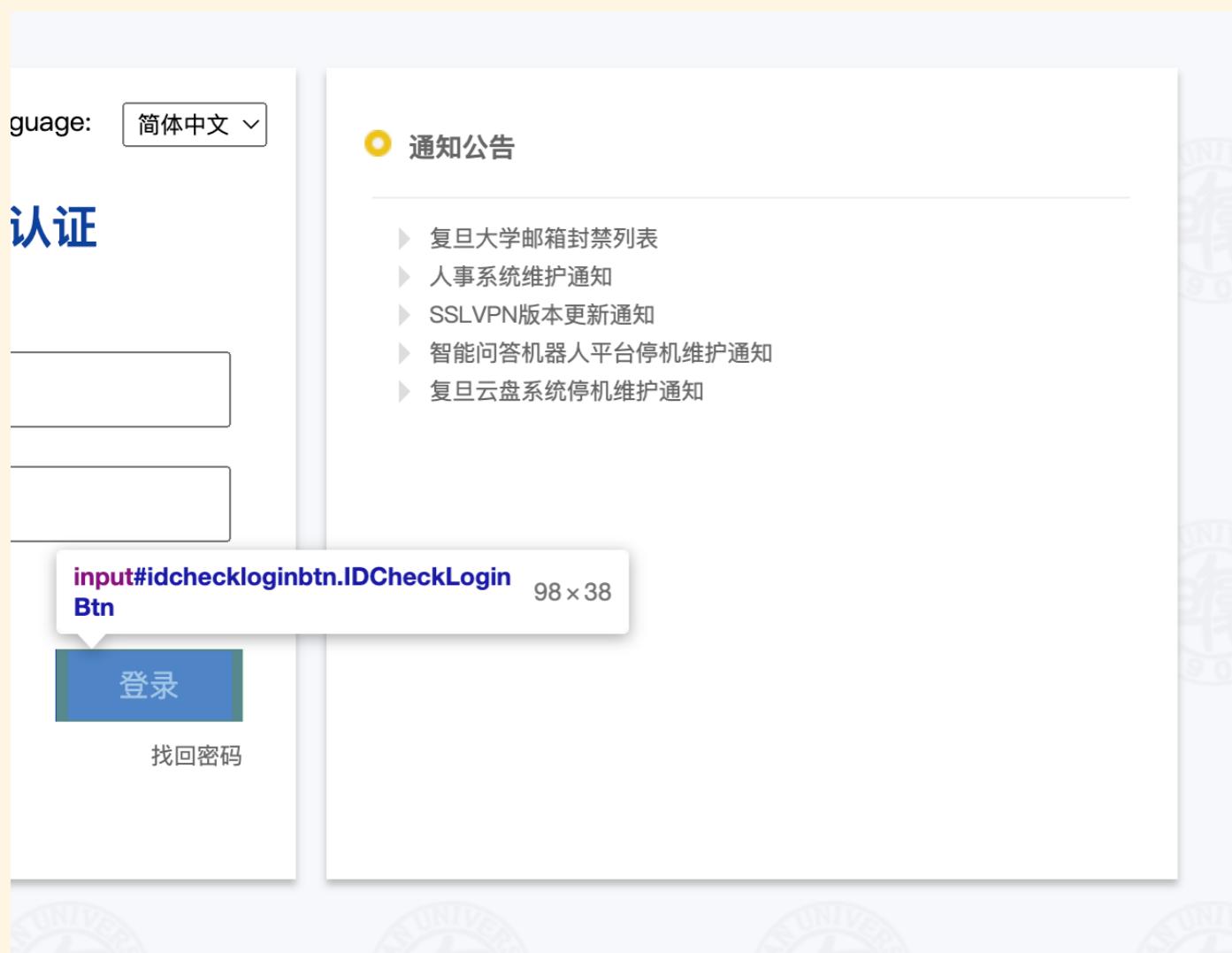
```
... form#casLoginForm div.main div.IDCheckMiddleCon div.IDCheckLogin div.IDCheckLc ...
```

Developer Tools:

- Styles
- Computed
- Layout
- Event Listeners
- DOM Breakpoints
- Properties

```
driver = webdriver.Chrome()
driver.get('https://zlapp.fudan.edu.cn/site/ncov/fudanDaily')
element = WebDriverWait(driver, 20).until(EC.presence_of_element_located((By.ID,"username")))
element.send_keys(username)
driver.find_element(By.ID,"password").send_keys(password)
driver.find_element(By.ID,"idcheckloginbtn").click()
```

# 自动平安复旦



```
driver = webdriver.Chrome()
driver.get('https://zlapp.fudan.edu.cn/site/ncov/fudanDaily')
element = WebDriverWait(driver,20).until(EC.presence_of_element_located((By.ID,"username")))
element.send_keys(username)
driver.find_element(By.ID,"password").send_keys(password)
driver.find_element(By.ID,"idcheckloginbtn").click()
```

# 自动平安复旦

```
element = WebDriverWait(driver, 20).until(EC.presence_of_element_located((By.CSS_SELECTOR,".wapat-btn.wapat-btn-ok")))
element.click()
driver.find_element(By.XPATH,"/html/body/div[1]/div/div[1]/section/div[4]/ul/li[4]/div/div/div[1]/span[1]").click()
driver.find_element(By.XPATH,"/html/body/div[1]/div/div[1]/section/div[4]/ul/li[6]/div/input").click()
time.sleep(15)
```

```
driver.find_element(By.PARTIAL_LINK_TEXT,"提交信息").click()
```

```
"cursor: pointer;">
▶ <span data-v-e714152c>...</span>
<input data-v-e714152c readonly="readonly" type="tex
t" placeholder="点击获取地理位置"> flex == $0
<div data-v-e714152c class="tishi-btn">无法获取地址?
</div> flex
</div>
```

```
"cursor: pointer;">
▶ <span data-v-e714152c>...</span>
<input data-v-e714152c readonly="readonly" type="tex
t" placeholder="地
址?>
<div data-v-e714152c>
</div> flex
</div>
</li>
▶ <li data-v-e714152c>
<!-->
<!-->
▶ <li data-v-e714152c>
▶ <li data-v-e714152c>
▶ <li data-v-e714152c>
<!-->
```

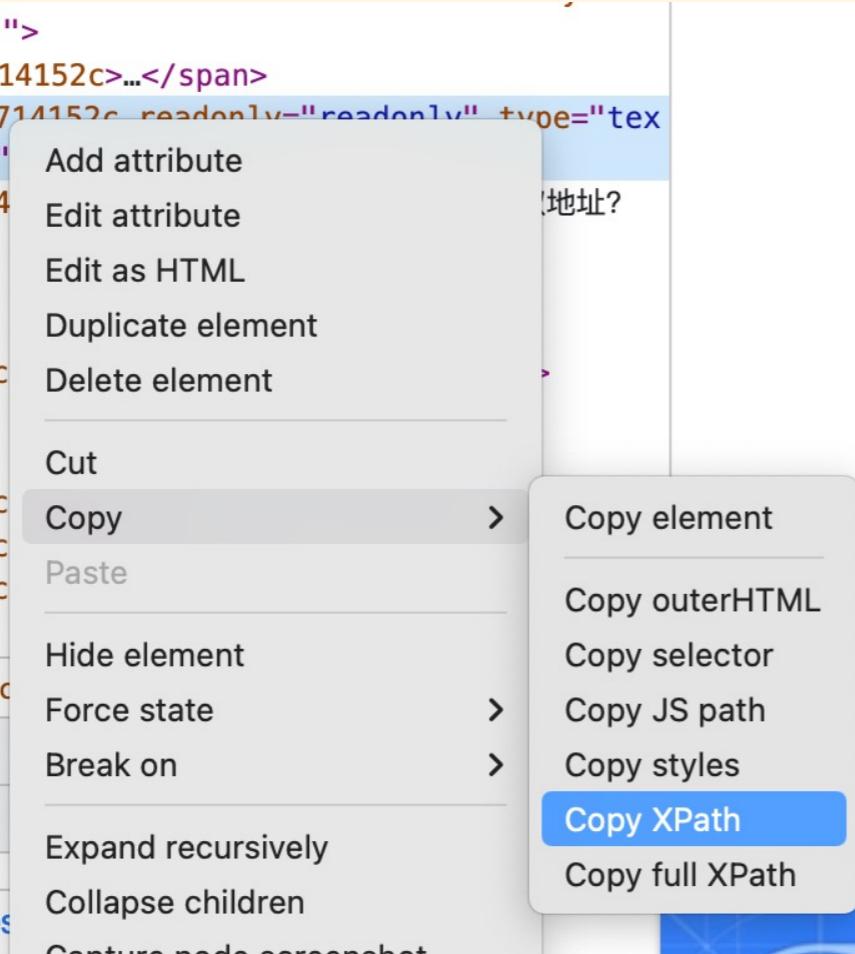
```
tem-buydate.form-detail2.ncc
```

```
What's New ×
```

```
the Chrome 106 update
```

```
ce insights panel updates
```

```
<div data-v-e714152c class="footers">
  <a data-v-e714152c class="wapcf-btn-qx" flex
    "提交信息 " == $0
    <em data-v-e714152c class="tab-title-desc-lit
    (Submit)</em>
  </a>
```



# 自动平安复旦

```
driver.find_element(By.CSS_SELECTOR,".wapcf-btn.wapcf-btn-ok").click()
img = driver.find_element(By.XPATH,"/html/body/div[1]/div/div[3]/div/div[2]/img")
time.sleep(1)
data = img.screenshot_as_png    ## 获取base64

client = AipOcr(APP_ID, API_KEY, SECRET_KEY)

res = client.basicGeneral(data, {})
res_word = res['words_result'][0]['words'].rstrip().replace(' ', '')

driver.find_element(By.XPATH,"/html/body/div[1]/div/div[3]/div/div[1]/input").send_keys(res_word)
driver.find_element(By.XPATH,"/html/body/div[1]/div/div[3]/div/div[3]/div").click()
```

## 如何处理验证码？

对于简单的验证码，可以编写自动化程序，点击“看不清”，获取它的验证码，通过深度学习的方式，学习识别验证码。

平安复旦的验证码十分幼稚，完全可以进行深度学习，当然，可以用成熟免费的百度OCR API，申请方式很简单。

# 自动平安复旦

Code 文件 编辑 选择 查看 转到 运行 终端 窗口 帮助

cookbook.ipynb — selenium

cookbook.ipynb settings.py DockerFile Downie.crx webdriver.py element.gif

cookbook.ipynb > M+ 元素定位失败 > M+ 页面等待 > M+ 小例子 (平安复旦自动填报) > from selenium import webdriver

+ 代码 + Markdown | 全部运行 清除所有单元格输出 转到 重启 变量表 Outline

statistics (Python 3.7.13)

from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.support import expected\_conditions as EC  
from selenium.webdriver.support.ui import WebDriverWait  
from aip import AipOcr  
from selenium import webdriver  
from settings import \*  
import time  
driver = webdriver.Chrome()  
driver.get('https://zlappt.fudan.edu.cn/site/WeChat/fudanDaily')  
element = WebDriverWait(driver, 20).until(EC.presence\_of\_element\_located((By.ID, "username")))  
element.send\_keys(username)  
driver.find\_element(By.ID, "password").send\_keys(password)  
driver.find\_element(By.ID, "idcheckloginbtn").click()  
  
element = WebDriverWait(driver, 20).until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, ".wapat-btn.wapat-btn-ok")))  
element.click()  
driver.find\_element(By.XPATH, "/html/body/div[1]/div/div[1]/section/div[4]/ul/li[4]/div/div/div[1]/span[1]").click()  
driver.find\_element(By.XPATH, "/html/body/div[1]/div/div[1]/section/div[4]/ul/li[6]/div/input").click()  
time.sleep(15)  
  
driver.find\_element(By.PARTIAL\_LINK\_TEXT, "提交信息").click()  
  
driver.find\_element(By.CSS\_SELECTOR, ".wapcf-btn.wapcf-btn-ok").click()  
img = driver.find\_element(By.XPATH, "/html/body/div[1]/div/div[3]/div/div[2]/img")  
time.sleep(1)  
data = img.screenshot\_as\_png ## 获取base64  
  
client = AipOcr(APP\_ID, API\_KEY, SECRET\_KEY)  
  
res = client.basicGeneral(data, {})  
res\_word = res['words\_result'][0]['words'].rstrip().replace(' ', '')  
  
driver.find\_element(By.XPATH, "/html/body/div[1]/div/div[3]/div/div[1]/input").send\_keys(res\_word)  
driver.find\_element(By.XPATH, "/html/body/div[1]/div/div[3]/div/div[3]/div").click()  
  
# driver.close()

[6] ⑥ 21.9s

KeyboardInterrupt Traceback (most recent call last)  
/var/folders/rp/hm8s3jw15879dx4wp\_lgk0lr000gn/T/ipykernel\_3575/1583370199.py in <module>  
18 driver.find\_element(By.XPATH, "/html/body/div[1]/div/div[1]/section/div[4]/ul/li[4]/div/div/div[1]/span[1]"  
19 driver.find\_element(By.XPATH, "/html/body/div[1]/div/div[1]/section/div[4]/ul/li[6]/div/input").click()  
--> 20 time.sleep(15)  
21  
22 driver.find\_element(By.PARTIAL\_LINK\_TEXT, "提交信息").click()

Can you please take a minute to tell us about your notebooks experience in VS Code?

来源: Jupyter (扩展)

Jupyter Server: Local 单元格 133/134 Colorize: 0 variables

姓名 筛选IPSS评分  
闵勤良 23  
贾志兴 15  
王志良 22  
吴兴才 15  
顾金娥 17  
吴东平 28  
胡龙华 32  
周文会 24  
陈招国 21  
唐庆明 28  
赵建强 16  
陈国其 14  
章勤鸿 22

资源管理器

打开的编辑器 1 个未保存

cookbook.ipynb settings.py DockerFile Downie.crx webdriver.py element.gif

SELENIUM

\_\_pycache\_\_  
pictures  
截屏2022-09-18 09.55.41.png  
截屏2022-09-18 10.13.18.png  
截屏2022-09-18 10.16.19.png  
截屏2022-09-18 10.55.32.png  
截屏2022-09-18 11.01.03.png  
截屏2022-09-20 16.13.25.png  
截屏2022-09-20 16.18.40.png  
element.gif  
work  
cookbook.ipynb  
DockerFile  
Downie.crx  
screenshot\_google.png  
settings.py

大纲

M+ Web自动化测试技术之Selenium  
M+ Web端自动化测试  
M+ 极高的效率执行  
M+ 空单元格  
M+ Selenium自动化环境搭建

时间线 cookbook.ipynb

edit 3 周  
Insert Cell 4 周  
edit 1 个月  
撤消/重做

Mac OS Dock

# 制作IPO简报

scripts/chuangyeban.py

打开<http://listing.szse.cn/projectdynamic/ipo/index.html>，发现table内元素动态加载，没办法通过get或者post该网址直接获取。采用selenium提取网页table标签内元素。

The screenshot shows a table of IPO projects with columns: 1. 友行人全称 (Issuer Name), 2. 申核状态 (Review Status), 3. 注册地 (Registration Location), 4. 证监会行业 (CSRC Industry), and 5. 保荐人 (Underwriter). The first row is highlighted in blue. The DOM structure on the right side of the table shows the HTML code for the table rows, with the first row's code highlighted in yellow.

STOCK EXCHANGE					
	友行人全称	申核状态	注册地	证监会行业	保荐人
1	杭州国芯科技股份有限公司	已问询	浙江	计算机、通信和其他电子设备制造业	中信建投证券股份有限公司
2	江西省江铜铜箔科技股份有限公司	已问询	江西	计算机、通信和其他电子设备制造业	中信建投证券股份有限公司
3	常州武进中瑞电子科技股份有限公司	已问询	江苏	金属制品业	华泰联合证券有限责任公司

```
<tr 1100x77>
  <td>1</td>
  <td>杭州国芯科技股份有限公司</td>
  <td>已问询</td>
  <td>浙江</td>
  <td>计算机、通信和其他电子设备制造业</td>
  <td>中信建投证券股份有限公司</td>
</tr>
```

```
<div class="ml10 mr10 table-out-con">
  <table class="reg-table" style="min-width:1100px;">
    <thead>...</thead>
    <tbody class="projectdynamic-tbody-con">
      <tr> == $0
        <td class="text-center">1</td>
        <td>
          <a target="_blank" href="/projectdynamic/ipo/detail?l?&id=1002266">杭州国芯科技股份有限公司 </a>
        </td>
        <!-- <td class="text-center"><span class="span-a-style" onClick="searchProject(8, '16')">创业板</span></td> -->
        <td>...</td>
        <td>...</td>
        <td>...</td>
        <td>...</td>
      </tr>
    </tbody>
  </table>
</div>
```

# 制作IPO简报

## scripts/chuangyeban.py

```
findname = re.compile(r'target="_blank">(.*) </a></td>')
findlink = re.compile(r'<td><a href="(.*?)" target="_blank">')
findzt = re.compile(r'">(.*?)</span></td> ')
finddate = re.compile(r'<td class="text_center">(.*?)</td>')

chrome_options = Options()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--disable-gpu')
driver=webdriver.Chrome(options = chrome_options)
driver.get("http://listing.szse.cn/projectdynamic/"+name+"/index.html")
time.sleep(1)

page=BeautifulSoup(driver.page_source,'html5lib').body
info = page.find_all("tr")

for item in info:
    item = str(item)
    if re.findall(findname,item) != []:
        title = re.findall(findname,item)[0]
        link = re.findall(findlink,item)[0]
        zt = re.findall(findzt,item)[1]
        date = re.findall(finddate,item)[2]
        # print(date)
        compare_date = datetime.datetime.strptime(date, '%Y-%m-%d')
        if compare_date < limit_date:
            flag = 1
            break
    finallink = "http://listing.szse.cn/" + link
```

原代码

```
from selenium import webdriver
from selenium.webdriver.support.ui import Select
from selenium.webdriver.common.by import By
import re
driver = webdriver.Chrome()
driver.get(
    'http://listing.szse.cn/projectdynamic/ipo/index.html')

] ✓ 7.8s Python Python

items = driver.find_elements(By.CSS_SELECTOR,
    'div .ml10.mr10.table-out-con> table > tbody>tr')

] ✓ 0.3s Python Python

for item in items:
    print(item.text)
    a = item.find_element(By.CSS_SELECTOR, 'a')
    print(a.get_attribute('href'))

] ✓ 2.3s Python

1 湖南飞沃新能源科技股份有限公司 上市委会议通过 湖南 通用设备制造业 民生证券 湖南启元律师事务所 天健会计师事务所（特殊普通合伙） 2022-10-24  
2020-09-30  
http://listing.szse.cn/projectdynamic/ipo/detail/index.html?id=1000949
```

更优代码

# 制作IPO简报

scripts/chuangyeban.py

The screenshot shows a table of IPO applications and the corresponding DOM elements for a pagination bar.

			其他电子设备制造业		
铜箔科技股	已问询	江西	计算机、通信和其他电子设备制造业	中信建投证券	
瑞电子科技司	已问询	江苏	金属制品业	华泰联合证券	
车装备股份	已问询	河北	汽车制造业	长江证券	

Below the table is a pagination bar with a tooltip for the 'next' button:

li.next 89 x 34

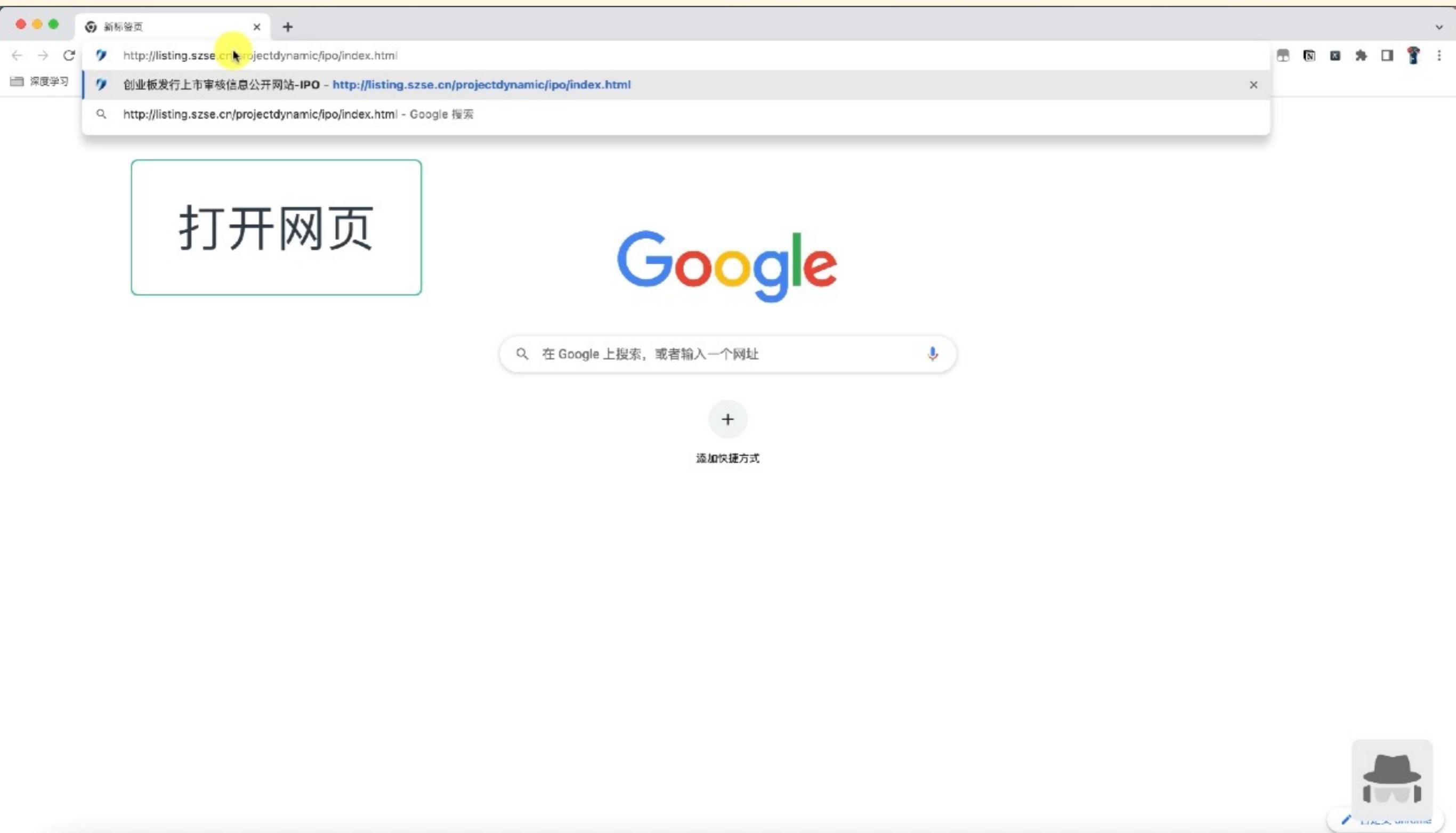
> 1 页 跳转

The right side of the screenshot shows the DOM structure of a pagination element:

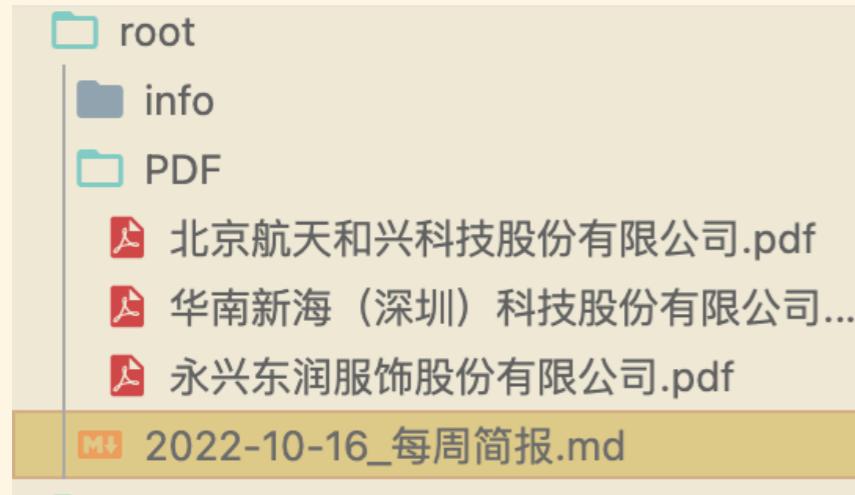
```
<li></li>
▶ <li>...</li>
<li></li>
▼ <li class="next" data-show="next">
...
    ▼ <a data-pi="1"> == $0
        "下一页"
        ::after
        </a>
    </li>
    <li></li>
    ▶ <li class="pagination-go intricacy">...</li>
</ul>
</div>
</div>
</div>
::after
</div>
...
body.childsite-body div.g-container-wrap div.g-container div.g-container
⋮ Console What's New ×
Highlights from the Chrome 106 update
```

```
try:
    WebDriverWait(driver, 10).until(lambda x: x.find_element(By.CLASS_NAME,'next'))
    time.sleep(1)
    a = driver.find_element(By.CLASS_NAME,'next')
    a.click()
except Exception as e:
    flag = 1
    pass
```

# 制作IPO简报



# 制作IPO简报



第六期公开发行股票招股书

- 苏州英华特涡旋技术股份有限公司
- 南通泰禾化工股份有限公司
- 四川沃文特生物工程股份有限公司
- 北京数聚智连科技股份有限公司
- 珠海市智迪科技股份有限公司
- 上海维科精密模塑股份有限公司
- 长城信息股份有限公司

2. 交易所审核终止情况

- 永兴东润服饰股份有限公司
- 北京航天和兴科技股份有限公司
- 华南新海（深圳）科技股份有限公司

3. 交易所注册终止情况

**科创板**

- 1. 交易所审核通过情况
  - 龙迅半导体（合肥）股份有限公司
- 2. 交易所审核终止情况
- 3. 交易所注册终止情况

**新发行股票**

本周新发行股票总共9只。

**创业板**

- 慧博云通
- 美好医疗

**科创板**

- 灿瑞科技
- 星环科技
- 哈铁科技

深证上审〔2022〕506号

永兴东润服饰股份有限公司：  
深圳证券交易所（以下简称本所）于 2021 年 12 月 29 日依  
法受理了你公司首次公开发行股票并在创业板上市的申  
请文件，  
并依法依规进行了审核。  
2022 年 10 月 13 日，你公司向本所提交了《永兴东润  
服饰股  
份有限公司关于撤回首次公开发行股票并在创业板上市  
申请文件  
的申请》，保荐人向本所提交了《广发证券股份有限公  
司关于撤回  
永兴东润服饰股份有限公司首次公开发行股票并在创  
业板上市申  
请文件的申请》。根据《深圳证券交易所创业板股票发