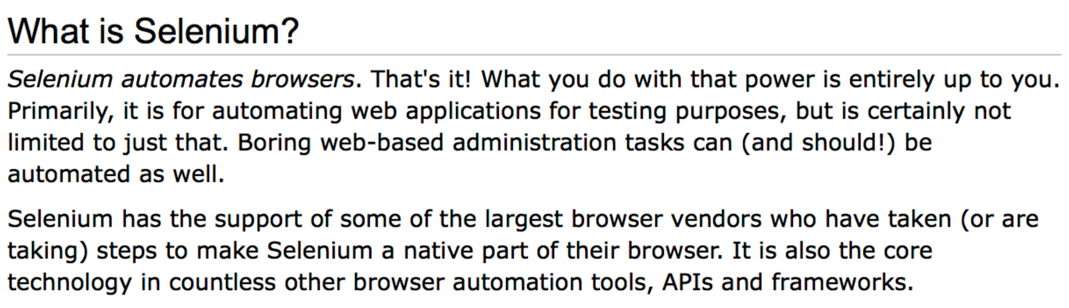
selenium介绍

官方文档:https://www.seleniumhq.org



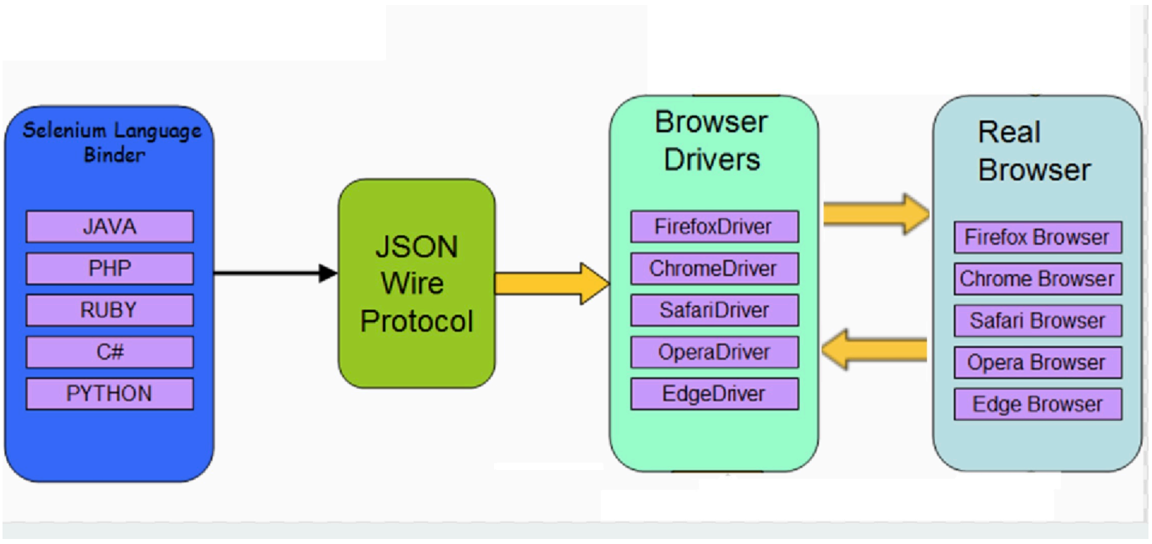
What is Selenium?

Selenium automates browsers. That's it! What you do with that power is entirely up to you. Primarily, it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should!) be automated as well.

Selenium has the support of some of the largest browser vendors who have taken (or are taking) steps to make Selenium a native part of their browser. It is also the core technology in countless other browser automation tools, APIs and frameworks.

简单来说就是做web自动化测试框架,可测试不同的浏览器.

webdriver系统架构



环境搭建

- python2.7或者3.6
- pycharm编辑器
- chrome浏览器
- chrome webdriver

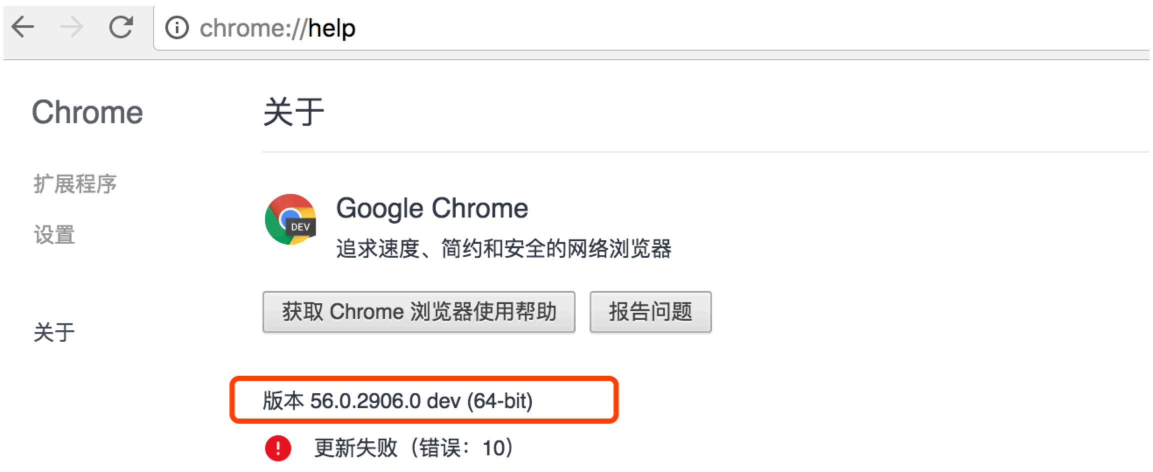
安装selenium

https://pypi.org/project/selenium/

```
pip install selenium
```

chrome webdriver选择版本

查看chrom浏览器的版本,需要下载其对应版本的chrome webdriver.



chrome://help

Chrome

扩展程序

设置

关于

关于

Google Chrome

追求速度、简约和安全的网络浏览器

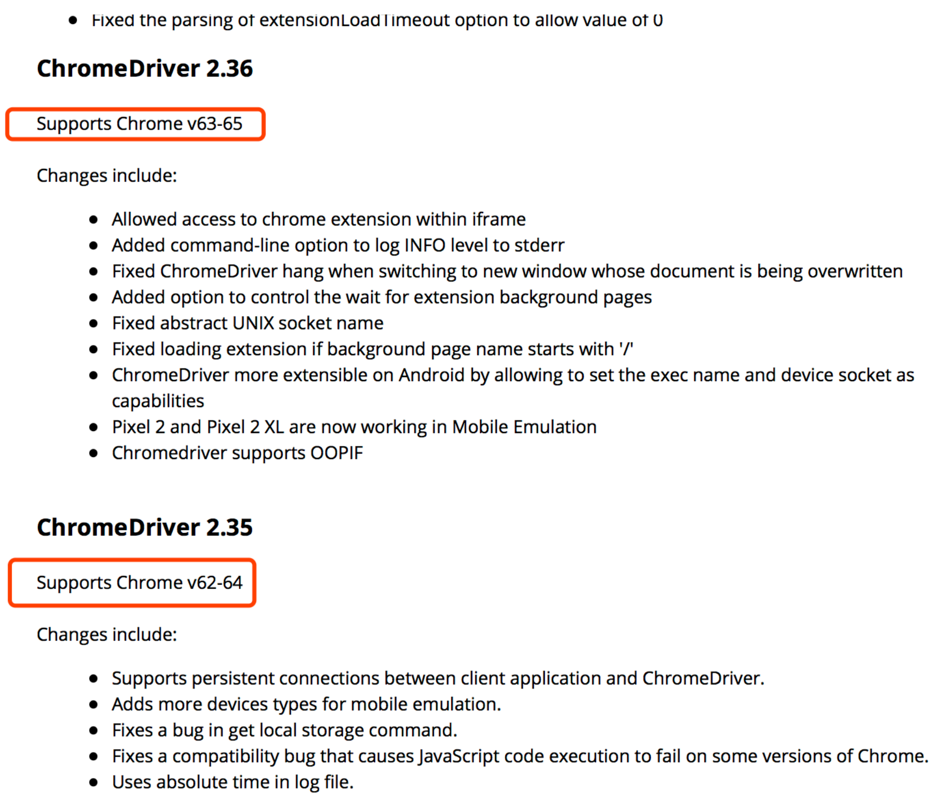
获取 Chrome 浏览器使用帮助

报告问题

版本 56.0.2906.0 dev (64-bit)

更新失败（错误：10）

下载对照表
https://sites.google.com/a/chromium.org/chromedriver/downloads



- Fixed the parsing of extensionLoadTimeout option to allow value of 0

ChromeDriver 2.36

Supports Chrome v63-65

Changes include:

- Allowed access to chrome extension within iframe
- Added command-line option to log INFO level to stderr
- Fixed ChromeDriver hang when switching to new window whose document is being overwritten
- Added option to control the wait for extension background pages
- Fixed abstract UNIX socket name
- Fixed loading extension if background page name starts with '/'
- ChromeDriver more extensible on Android by allowing to set the exec name and device socket as capabilities
- Pixel 2 and Pixel 2 XL are now working in Mobile Emulation
- Chromedriver supports OOPIF

ChromeDriver 2.35

Supports Chrome v62-64

Changes include:

- Supports persistent connections between client application and ChromeDriver.
- Adds more devices types for mobile emulation.
- Fixes a bug in get local storage command.
- Fixes a compatibility bug that causes JavaScript code execution to fail on some versions of Chrome.
- Uses absolute time in log file.

不同浏览器的driver

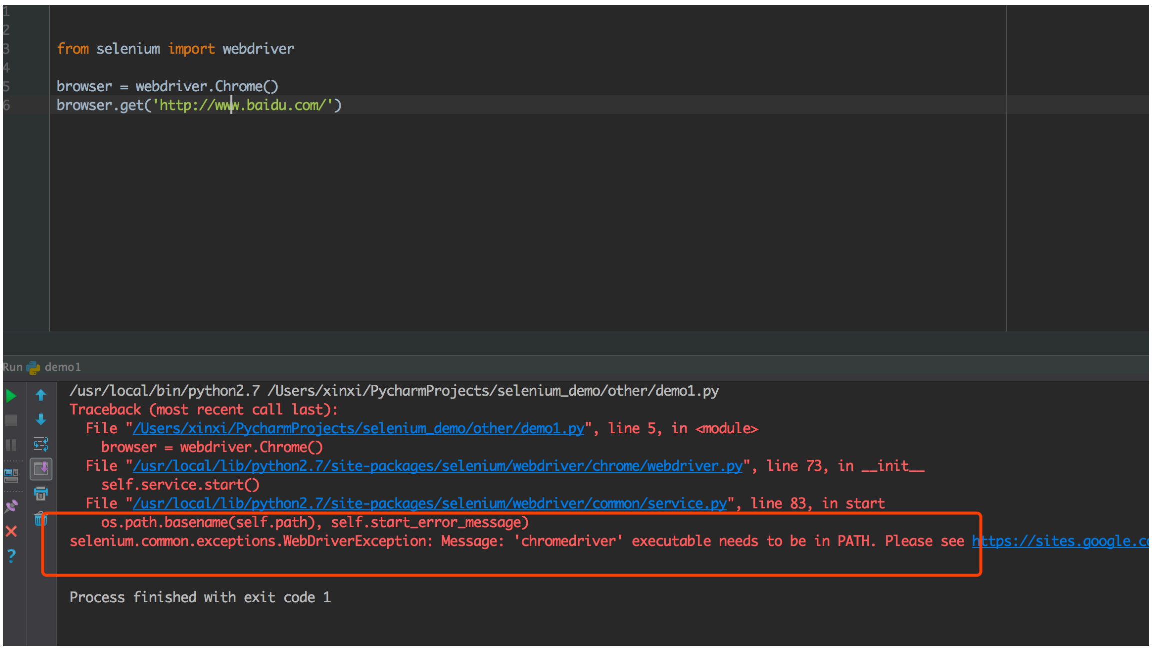
- browser = webdriver.Chrom()
- browser = webdriver.Firefox()
- browser = webdriver.Safari()
- browser = webdriver.Ie()

第一个demo

使用chrome浏览器打开百度

```
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('http://www.baidu.com/')
```



```python
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('http://www.baidu.com/')
```

```
/usr/local/bin/python2.7 /Users/xinxi/PycharmProjects/selenium_demo/other/demo1.py
Traceback (most recent call last):
  File "/Users/xinxi/PycharmProjects/selenium_demo/other/demo1.py", line 5, in <module>
    browser = webdriver.Chrome()
  File "/usr/local/lib/python2.7/site-packages/selenium/webdriver/chrome/webdriver.py", line 73, in __init__
    self.service.start()
  File "/usr/local/lib/python2.7/site-packages/selenium/webdriver/common/service.py", line 83, in start
    os.path.basename(self.path), self.start_error_message)
selenium.common.exceptions.WebDriverException: Message: 'chromedriver' executable needs to be in PATH. Please see https://sites.google.c

Process finished with exit code 1
```

报错提示需要指定"chromedriver"路径.

解决方案下载对应版本的chromedriver,代码改动如下:

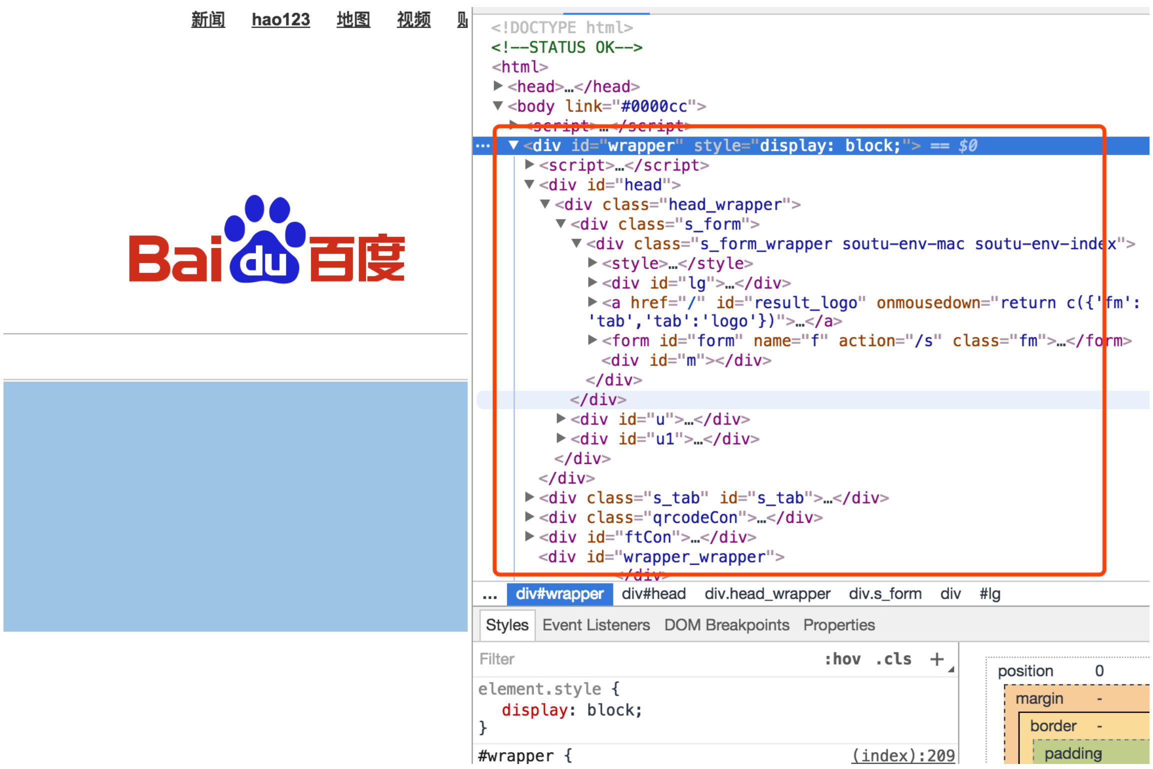
```python
from selenium import webdriver
browser =
webdriver.Chrome(executable_path="/Users/xinxi/PycharmProjects/selenium_demo/webdriver/chromedriver_mac")
# executable_path来指定chromedirver路径
browser.get('https://www.baidu.com')
```

指定代码,会启动一个chrome浏览器并且打开百度首页.

元素定位

web页面所有的元素最终都是以html格式展示.

使用chrome浏览器,右键查看页面元素.把鼠标定位到元素上,页面会自动定位到页面上.



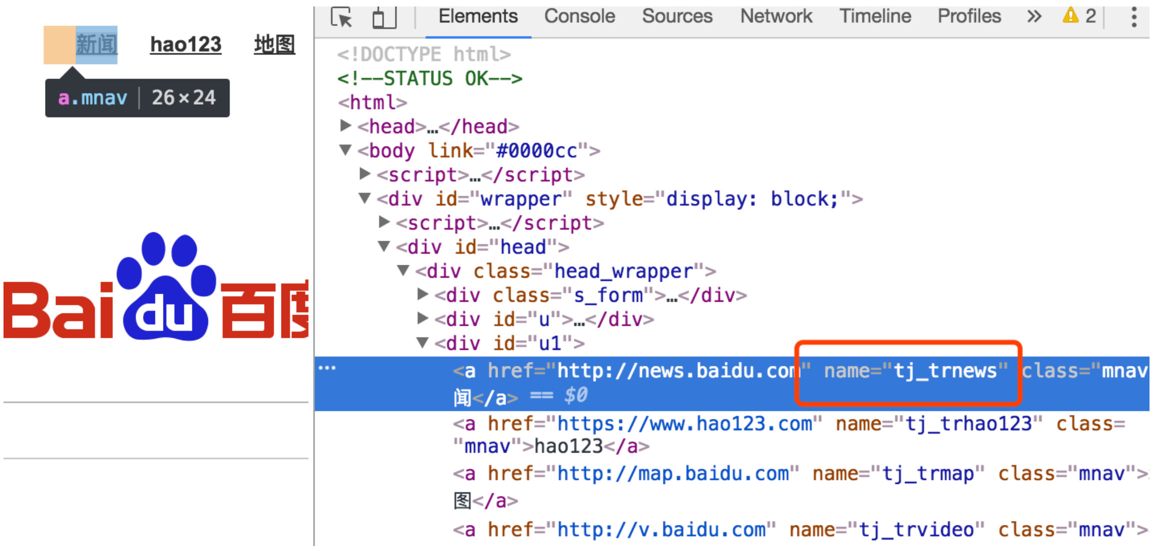
```
<!DOCTYPE html>
<!--STATUS OK-->
<html>
▶<head>…</head>
▼<body link="#0000cc">
  ▶<script>…</script>
… ▼<div id="wrapper" style="display: block;"> == $0
    ▶<script>…</script>
    ▼<div id="head">
      ▼<div class="head_wrapper">
        ▼<div class="s_form">
          ▼<div class="s_form_wrapper soutu-env-mac soutu-env-index">
            ▶<style>…</style>
            ▶<div id="lg">…</div>
            ▶<a href="/" id="result_logo" onmousedown="return c({'fm':
            'tab','tab':'logo'})">…</a>
            ▶<form id="form" name="f" action="/s" class="fm">…</form>
             <div id="m"></div>
            </div>
          </div>
        ▶<div id="u">…</div>
        ▶<div id="u1">…</div>
        </div>
      </div>
    ▶<div class="s_tab" id="s_tab">…</div>
    ▶<div class="qrcodeCon">…</div>
    ▶<div id="ftCon">…</div>
     <div id="wrapper_wrapper">
       </div>
```

```
… div#wrapper  div#head  div.head_wrapper  div.s_form  div  #lg
Styles  Event Listeners  DOM Breakpoints  Properties
Filter                                    :hov  .cls  +
element.style {
  display: block;
}
#wrapper {                               (index):209
position    0
margin  -
border  -
padding
```

所以做web自动化的关键点是如何操作这些元素,模拟点击、滑动、长按等操作.

selenium提供了八种元素定位方式

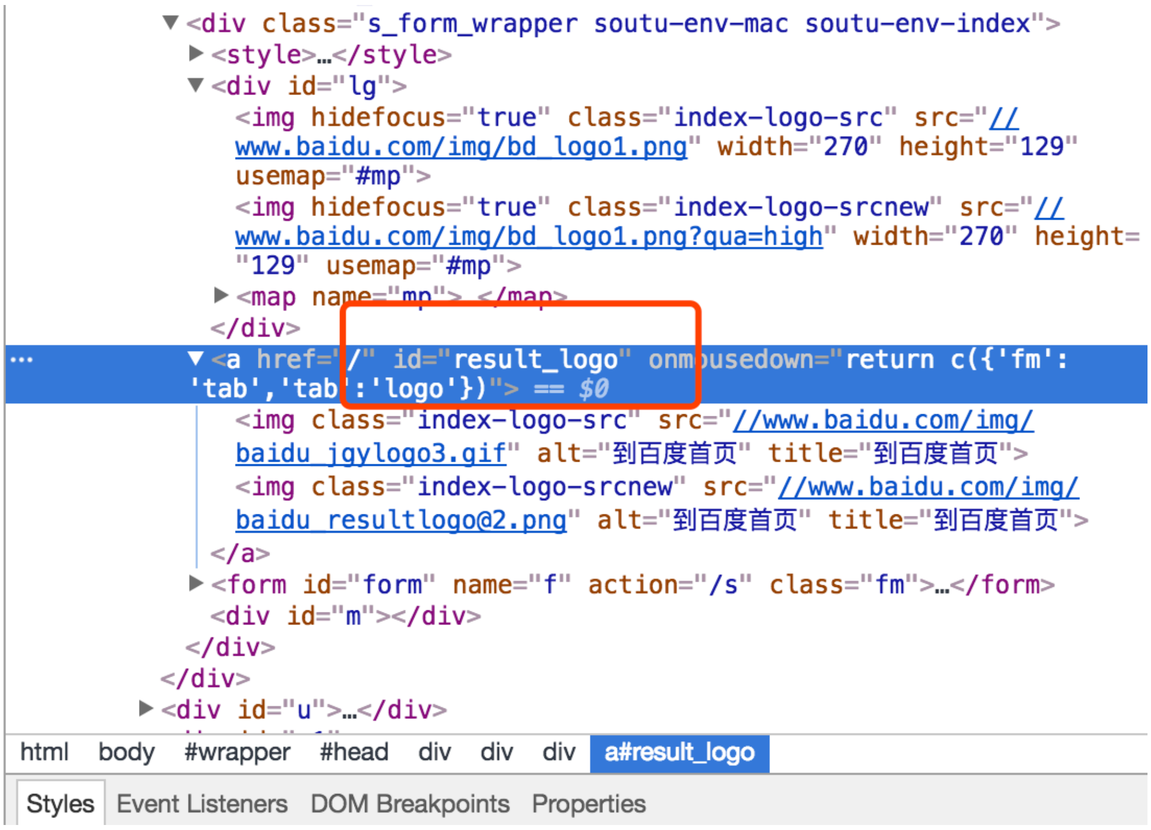
name定位



```
<!DOCTYPE html>
<!--STATUS OK-->
<html>
▶<head>…</head>
▼<body link="#0000cc">
  ▶<script>…</script>
  ▼<div id="wrapper" style="display: block;">
    ▶<script>…</script>
    ▼<div id="head">
      ▼<div class="head_wrapper">
        ▶<div class="s_form">…</div>
        ▶<div id="u">…</div>
        ▼<div id="u1">
…          <a href="http://news.baidu.com" name="tj_trnews" class="mnav
           闻</a> == $0
           <a href="https://www.hao123.com" name="tj_trhao123" class=
           "mnav">hao123</a>
           <a href="http://map.baidu.com" name="tj_trmap" class="mnav">
           图</a>
           <a href="http://v.baidu.com" name="tj_trvideo" class="mnav">
```

```
browser.find_element_by_name("tj_trnews").click()
```

id定位

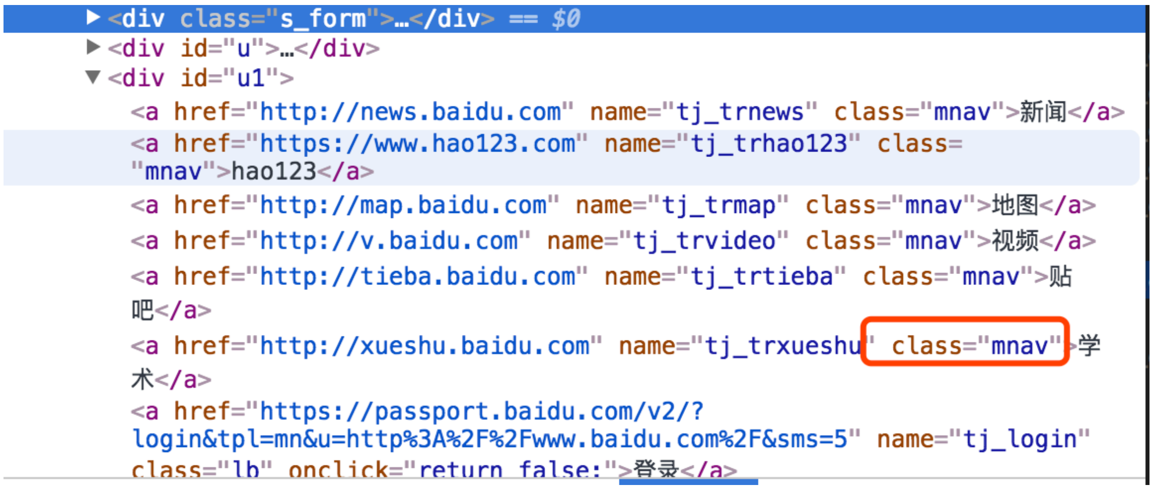

```
▼<div class="s_form_wrapper soutu-env-mac soutu-env-index">
  ▶<style>…</style>
  ▼<div id="lg">
     <img hidefocus="true" class="index-logo-src" src="//
     www.baidu.com/img/bd_logo1.png" width="270" height="129"
     usemap="#mp">
     <img hidefocus="true" class="index-logo-srcnew" src="//
     www.baidu.com/img/bd_logo1.png?qua=high" width="270" height=
     "129" usemap="#mp">
   ▶<map name="mp">…</map>
   </div>
 ▼<a href="/" id="result_logo" onmousedown="return c({'fm':
 'tab','tab':'logo'})"> == $0
     <img class="index-logo-src" src="//www.baidu.com/img/
     baidu_jgylogo3.gif" alt="到百度首页" title="到百度首页">
     <img class="index-logo-srcnew" src="//www.baidu.com/img/
     baidu_resultlogo@2.png" alt="到百度首页" title="到百度首页">
   </a>
 ▶<form id="form" name="f" action="/s" class="fm">…</form>
   <div id="m"></div>
  </div>
</div>
▶<div id="u">…</div>
```

html body #wrapper #head div div div a#result_logo

Styles Event Listeners DOM Breakpoints Properties

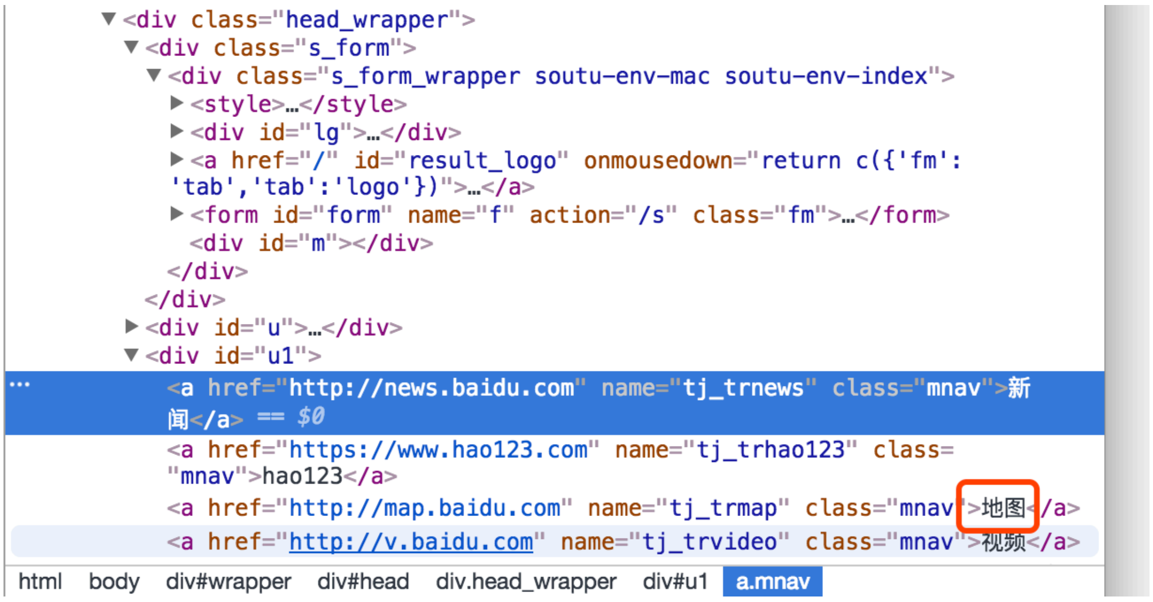
```
browser.find_element_by_id("result_logo").click()
```

class定位


```
▶<div class="s_form">…</div> == $0
▶<div id="u">…</div>
▼<div id="u1">
   <a href="http://news.baidu.com" name="tj_trnews" class="mnav">新闻</a>
   <a href="https://www.hao123.com" name="tj_trhao123" class=
   "mnav">hao123</a>
   <a href="http://map.baidu.com" name="tj_trmap" class="mnav">地图</a>
   <a href="http://v.baidu.com" name="tj_trvideo" class="mnav">视频</a>
   <a href="http://tieba.baidu.com" name="tj_trtieba" class="mnav">贴
   吧</a>
   <a href="http://xueshu.baidu.com" name="tj_trxueshu" class="mnav">学
   术</a>
   <a href="https://passport.baidu.com/v2/?
   login&tpl=mn&u=http%3A%2F%2Fwww.baidu.com%2F&sms=5" name="tj_login"
   class="lb" onclick="return false;">登录</a>
```

```
browser.find_element_by_class_name("mnav").click()
```

link text定位



```
▼<div class="head_wrapper">
  ▼<div class="s_form">
    ▼<div class="s_form_wrapper soutu-env-mac soutu-env-index">
      ▶<style>…</style>
      ▶<div id="lg">…</div>
      ▶<a href="/" id="result_logo" onmousedown="return c({'fm':
      'tab','tab':'logo'})">…</a>
      ▶<form id="form" name="f" action="/s" class="fm">…</form>
       <div id="m"></div>
     </div>
   </div>
  ▶<div id="u">…</div>
  ▼<div id="u1">
      <a href="http://news.baidu.com" name="tj_trnews" class="mnav">新
      闻</a> == $0
      <a href="https://www.hao123.com" name="tj_trhao123" class=
      "mnav">hao123</a>
      <a href="http://map.baidu.com" name="tj_trmap" class="mnav">地图</a>
      <a href="http://v.baidu.com" name="tj_trvideo" class="mnav">视频</a>
html  body  div#wrapper  div#head  div.head_wrapper  div#u1  a.mnav
```

```
browser.find_element_by_link_text("地图").click()
```

xpath定位

```
browser.find_element_by_xpath('//*[@id="result_logo"]').click()
```

css定位

```
browser.find_element_by_css_selector('#form').click()
```

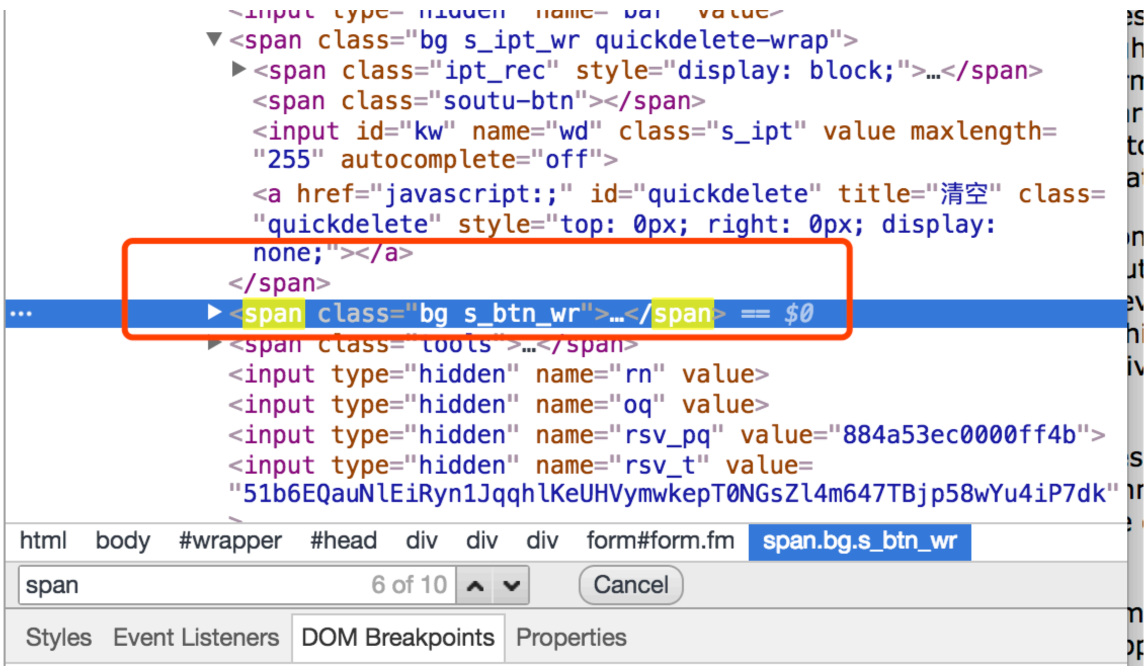
partial link text定位

通过链接文本的部分匹配来标识元素

```
browser.find_element_by_partial_link_text("地").click()
```

tag name定位

使用h1、a、span这种标签定位.



```
▼<span class="bg s_ipt_wr quickdelete-wrap">
  ▶<span class="ipt_rec" style="display: block;">…</span>
   <span class="soutu-btn"></span>
   <input id="kw" name="wd" class="s_ipt" value maxlength=
   "255" autocomplete="off">
   <a href="javascript:;" id="quickdelete" title="清空" class=
   "quickdelete" style="top: 0px; right: 0px; display:
   none;"></a>
 </span>
▶<span class="bg s_btn_wr">…</span> == $0
▶<span class="tools">…</span>
 <input type="hidden" name="rn" value>
 <input type="hidden" name="oq" value>
 <input type="hidden" name="rsv_pq" value="884a53ec0000ff4b">
 <input type="hidden" name="rsv_t" value=
 "51b6EQauNlEiRyn1JqqhlKeUHVymwkepT0NGsZl4m647TBjp58wYu4iP7dk"
```

html body #wrapper #head div div div form#form.fm span.bg.s_btn_wr
span 6 of 10
Cancel
Styles Event Listeners DOM Breakpoints Properties

```
browser.find_element_by_tag_name("span").click()
```

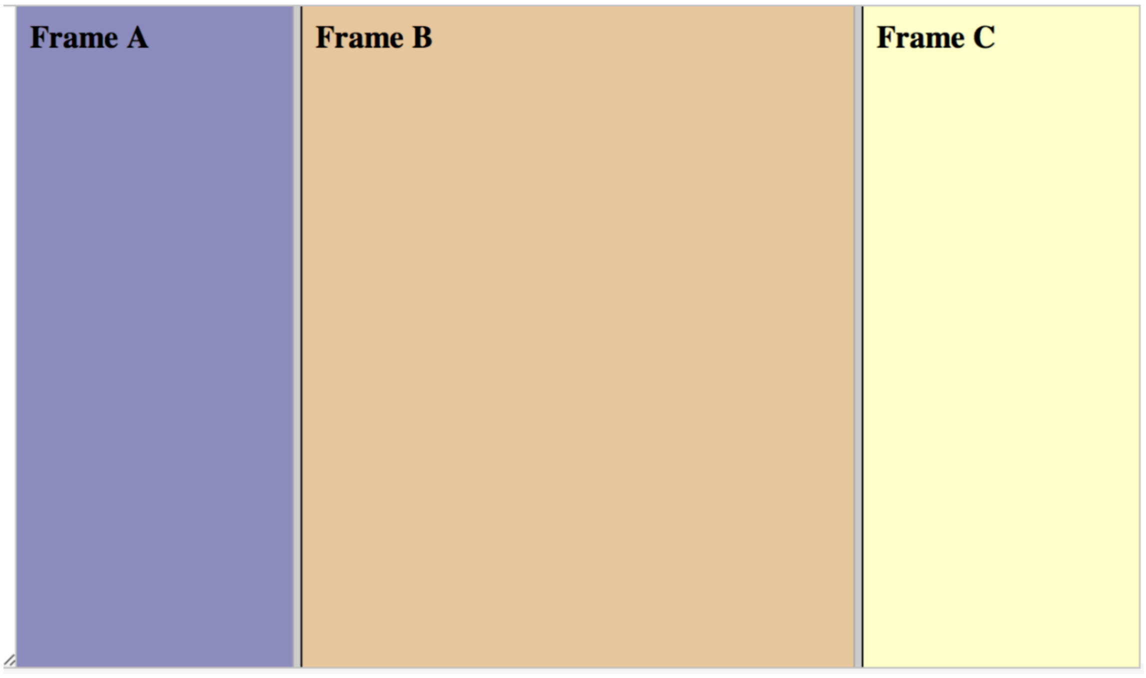
定位选择顺序

id > class > name > link_text > xpath > csss

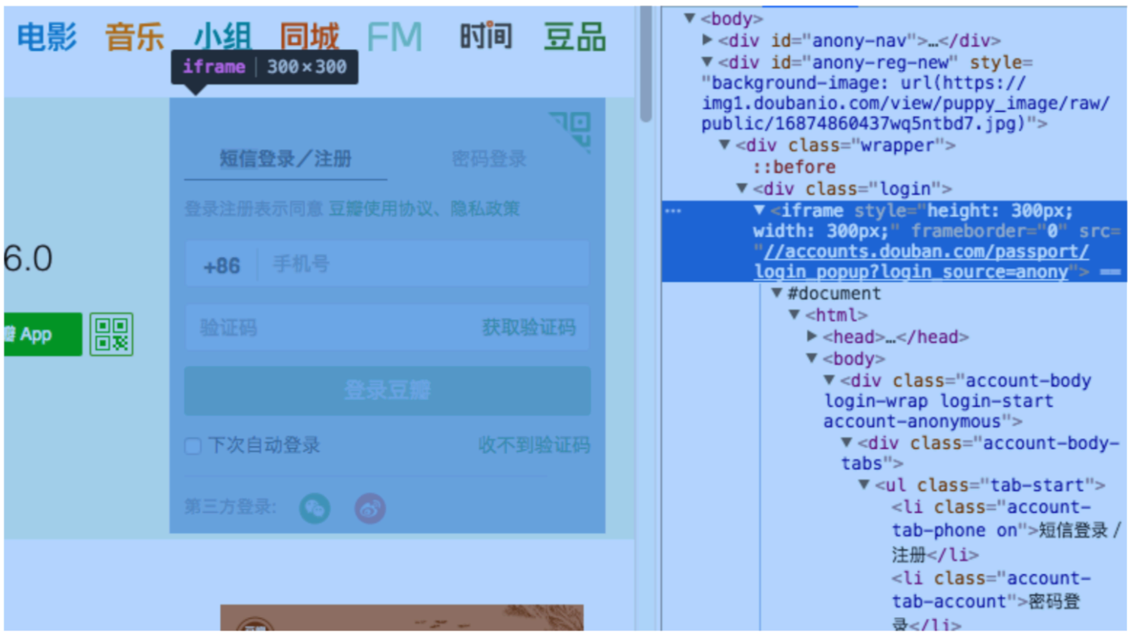
frame切换

<frame> 是HTML元素，它定义了一个特定区域，另一个HTML文档可以在里面展示.

```
<html>
<frameset cols="25%,50%,25%">
<frame src="/example/html/frame_a.html"> <frame src="/example/html/frame_b.html"> <frame
src="/example/html/frame_c.html">
</frameset> </html>
```



由此可见不同的frame包含不用的元素里里边. douban网首页为例例,通过元素检查登录区域是一个frame区域



```
▼<body>
  ▶<div id="anony-nav">…</div>
  ▼<div id="anony-reg-new" style=
  "background-image: url(https://
  img1.doubanio.com/view/puppy_image/raw/
  public/16874860437wq5ntbd7.jpg)">
    ▼<div class="wrapper">
        ::before
      ▼<div class="login">
        ▼<iframe style="height: 300px;
        width: 300px;" frameborder="0" src=
        "//accounts.douban.com/passport/
        login_popup?login_source=anony"> ==
          ▼#document
            ▼<html>
              ▶<head>…</head>
              ▼<body>
                ▼<div class="account-body
                login-wrap login-start
                account-anonymous">
                  ▼<div class="account-body-
                  tabs">
                    ▼<ul class="tab-start">
                        <li class="account-
                        tab-phone on">短信登录 /
                        注册</li>
                        <li class="account-
                        tab-account">密码登
                        录</li>
```

那用自动化脚本点击"密码登录"按钮,代码如下:

```
from selenium import webdriver
browser =
webdriver.Chrome(executable_path="/Users/xinxi/PycharmProjects/selenium_demo/webdriver/chromedriver_mac")
# executable_path来指定chromedirver路路径
browser.get("https://www.douban.com")
print("登录douban")
browser.find_element_by_class_name('account-tab-account').click()
print("点击密码登录")
```

错误提示如下:

```
selenium.common.exceptions.NoSuchElementException: Message: no such element: Unable to locate element:
{"method":"class name","selector":"account-tab-account"}
```

修改代码如下:

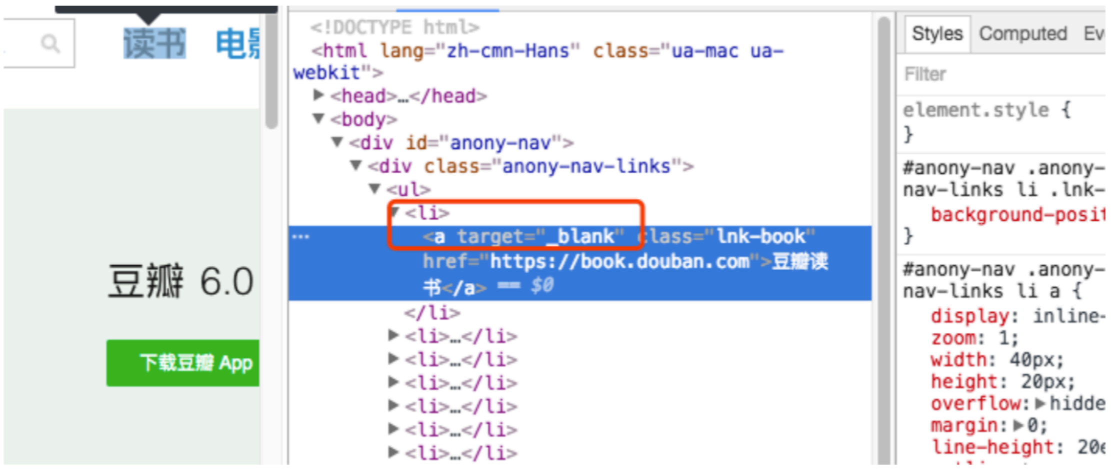
```
browser.switch_to.frame(browser.find_elements_by_tag_name("iframe")[0]) print("切换frame")
browser.find_element_by_class_name('account-tab-account').click() print("点击密码登录")
```

此时代码运行正常,说明点击"元素"之前说明需要切换到该frame上. selenium提供了了switch_to.frame方法 用法如下:

```
driver.switch_to.frame('frame_name') # 使用用名字
driver.switch_to.frame(0) # 使用用名字 # 使用用位置序号
driver.switch_to.frame(driver.find_elements_by_tag_name("iframe")[0]) # 使用用标签序号
```

windows切换

在web页页面面上经常有点击一个页面后,另外打开一个新窗口.是因为html中target参数是"_bank"控制.



```
<!DOCTYPE html>
<html lang="zh-cmn-Hans" class="ua-mac ua-
webkit">
  ▶<head>…</head>
  ▼<body>
    ▼<div id="anony-nav">
      ▼<div class="anony-nav-links">
        ▼<ul>
          ▼<li>
... <a target="_blank" class="lnk-book"
            href="https://book.douban.com">豆瓣读
            书</a> == $0
          </li>
          ▶<li>…</li>
          ▶<li>…</li>
          ▶<li>…</li>
          ▶<li>…</li>
          ▶<li>…</li>
          ▶<li>…</li>
```

```
element.style {
}
#anony-nav .anony-
nav-links li .lnk-
   background-posi
}
#anony-nav .anony-
nav-links li a {
   display: inline-
   zoom: 1;
   width: 40px;
   height: 20px;
   overflow:▶hidde
   margin:▶0;
   line-height: 20
```

所以在做自动化测试过程中,点击跳转以后.页面句柄还在当页面,所以不能点击跳转以后的页面元素. 所以此时需要切换windows,selenium提供了了switch_to.frame方法 用法如下:

```
driver.switch_to.window('main') window的名字
```

代码如下:

```
from selenium import webdriver
browser =
webdriver.Chrome(executable_path="/Users/xinxi/PycharmProjects/selenium_demo/webdriver/chromedriver_mac")
# executable_path来指定chromedirver路路径
browser.get("https://www.douban.com")
print("登录douban")
browser.find_element_by_class_name("lnk-book").click() print("点击读书")
handles_list = browser.window_handles print(handles_list) browser.switch_to.window(str(handles_list[-1]))
print("切换窗口口")
if browser.current_window_handle == str(handles_list[-1]): print("切换窗口口成功...")
```

等待

等待的目目的主要是为了页面加载完成才点击元素,避免找不到元素的现象. 一共有三种等待:

硬编码等待

```
time.sleep(1) ,不不推荐
```

全局隐示等待

```
browser.get("https://www.douban.com") print("登录douban")
browser.implicitly_wait(10) # 会等待每个元素最大大时间10s browser.find_element_by_class_name("lnk-
book").click() print("点击读书")
```

显示等待

```
WebDriverWait(browser, 5).until(EC.presence_of_element_located((By.CLASS_NAME, "lnk-book"))).click() # 显
示等待lnk-book 5s
```

使用用方方式:显示等待 > 全局隐示等待 > 硬编码等待

常用方法

获取窗口大小

```
print browser.get_window_size() # 获取窗口口页页面面当前高高和宽
```

获取cookies

```
print(browser.get_cookies())
```

设置网络

```
print(browser.set_network_conditions(
                offline=False,
                latency=5,  # additional latency (ms)
                download_throughput=500 * 1024,  # maximal throughput
                upload_throughput=500 * 1024)  # maximal throughput
            )
```

获取title

```
print(browser.title)
```

获取页面元素

```
print(browser.page_source)
```

获取页面url

```
print(browser.current_url)
```

页面放大

```
browser.maximize_window()
```

页面缩小

```
browser.minimize_window()
```

设置页面大小

```
browser.set_window_size(300,300)
```

向前

```
browser.forward()
```

后退

```
browser.back()
```

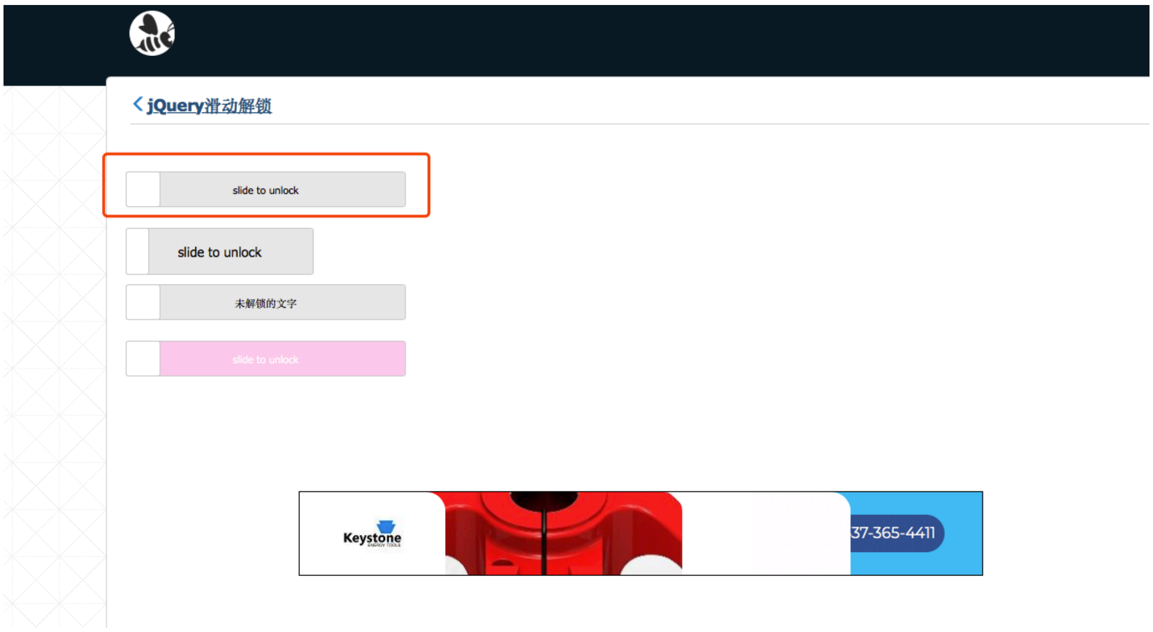
滑动

```
#滑到底部 js="window.scrollTo(0,document.body.scrollHeight)" browser.execute_script(js)
```

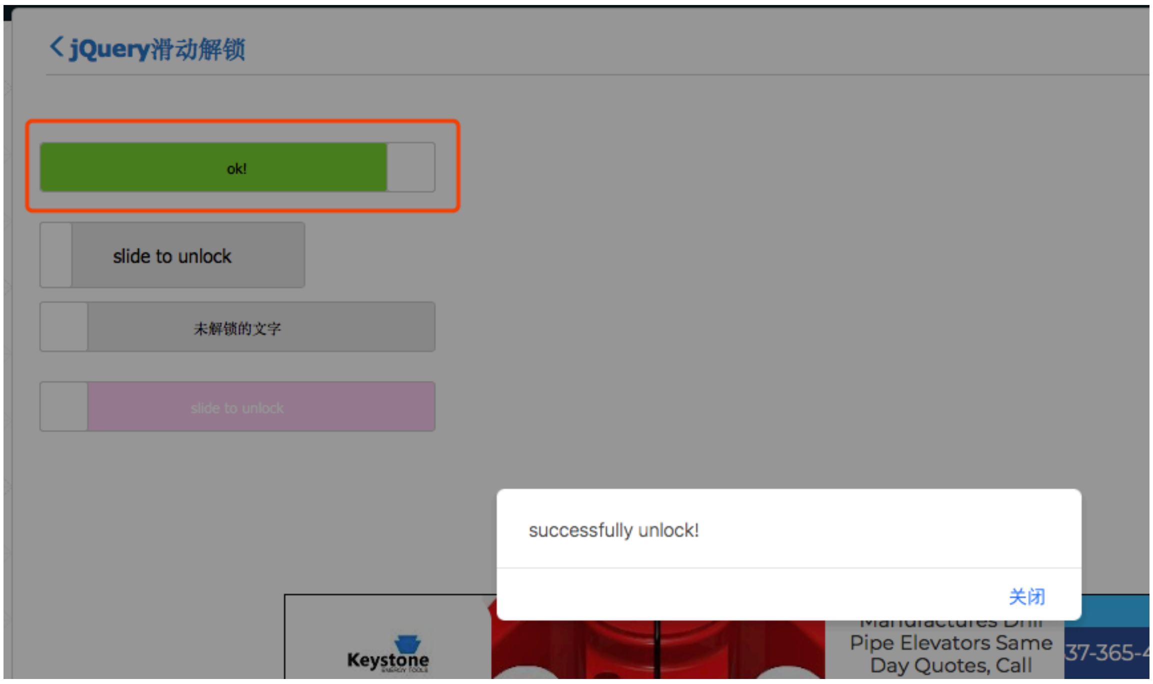
```
#滑动到顶部 js="window.scrollTo(0,0)" browser.execute_script(js)
```

滑动解锁

https://www.helloweba.net/demo/2017/unlock/



```
#获取拖动条
dragger = browser.find_elements_by_class_name("slide-to-unlock-handle")[0]
#获取拖动条进度条长度
dragger_text = browser.find_elements_by_class_name("slide-to-unlock-bg")[0]
x = dragger_text.location["x"]
action = ActionChains(browser)
 #鼠标左键按下不放
action.click_and_hold(dragger).perform()
#平行移动大于解锁的长度的距离
try:
    action.drag_and_drop_by_offset(dragger,x, 0).perform()
    print("滑动...")
except Exception as e:
    print("faild")
```


<jQuery滑动解锁

ok!

slide to unlock

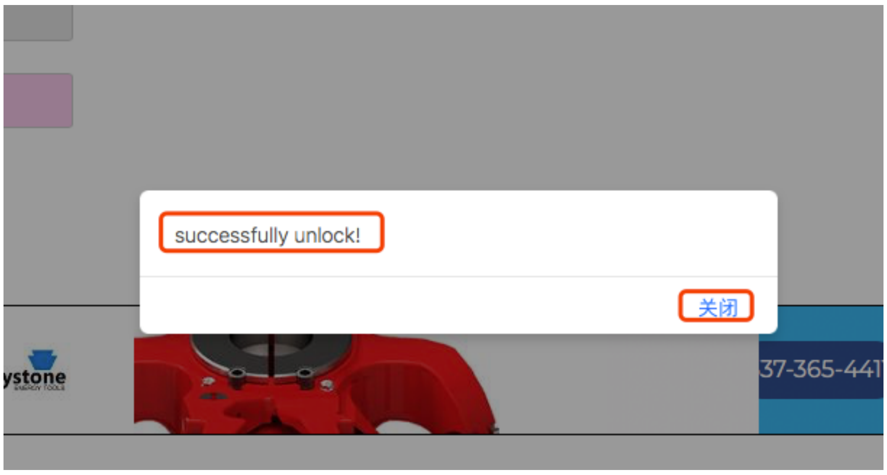
未解锁的文字

slide to unlock

successfully unlock!

关闭

关闭弹框


successfully unlock!

关闭

```
time.sleep(10)
try:
    tt = browser.switch_to_alert()
    print("打印警告框提示...")
except Exception as e:
    print("faild")
success_text = tt.text
print(success_text)
tt.accept()
```

截图

保存截图

```
browser.save_screenshot("/Users/xinxi/PycharmProjects/selenium_demo/screen_folder/截图.png")
```

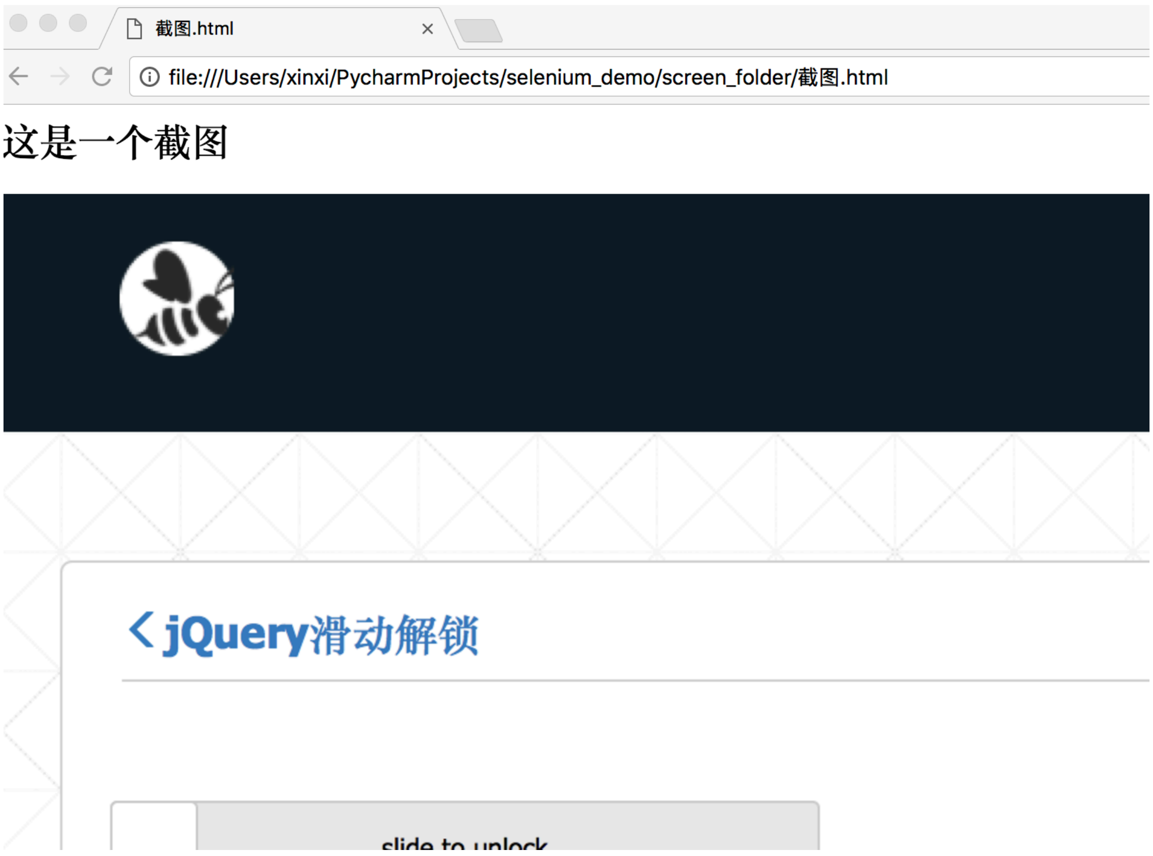
图片流截图

```
sc_str = browser.get_screenshot_as_png()
sc_path = "/Users/xinxi/PycharmProjects/selenium_demo/screen_folder/截图.png"
with open(sc_path,"w") as f:
    f.write(sc_str)
```

base64截图

```
sc_str = browser.get_screenshot_as_base64()
html_tmp = """
<html>
<body>

<h1>这是一个截图</h1>
<img src="data:image/png;base64,{}"/>
</body>
</html>
""".format(sc_str)
html_path = "/Users/xinxi/PycharmProjects/selenium_demo/screen_folder/截图.html"
with open(html_path,"w") as f:
    f.write(html_tmp)
```


截图.html
file:///Users/xinxi/PycharmProjects/selenium_demo/screen_folder/截图.html
这是一个截图
jQuery滑动解锁
slide to unlock

断言

PO模式

报告

jenkins持续集成

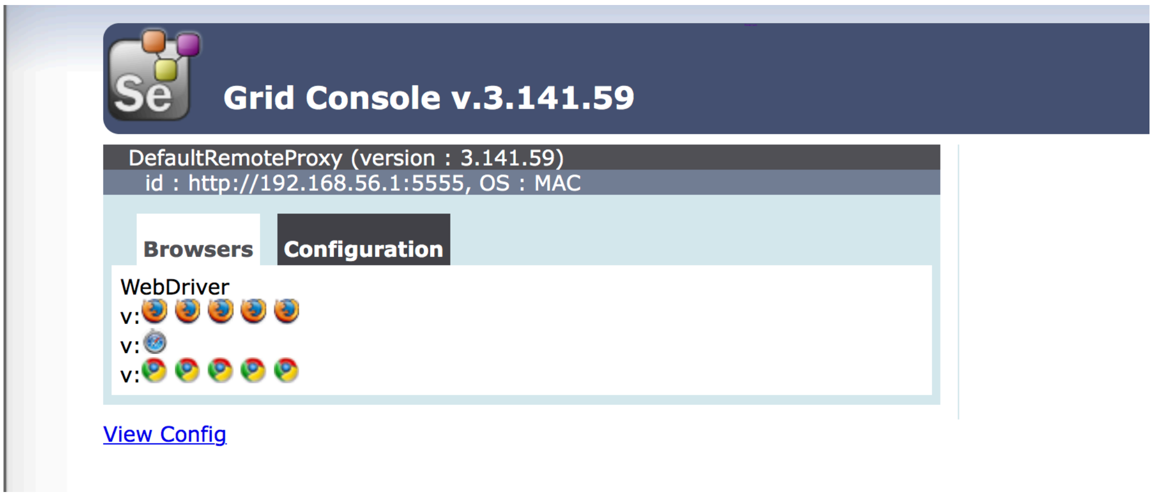
selenium分布式

官方文档:https://github.com/SeleniumHQ/selenium/wiki/Grid2

下载selenium-server-standalone-3.141.59.jar

启动hub

```
java -jar selenium-server-standalone-3.141.59.jar -role hub -port 4455
```

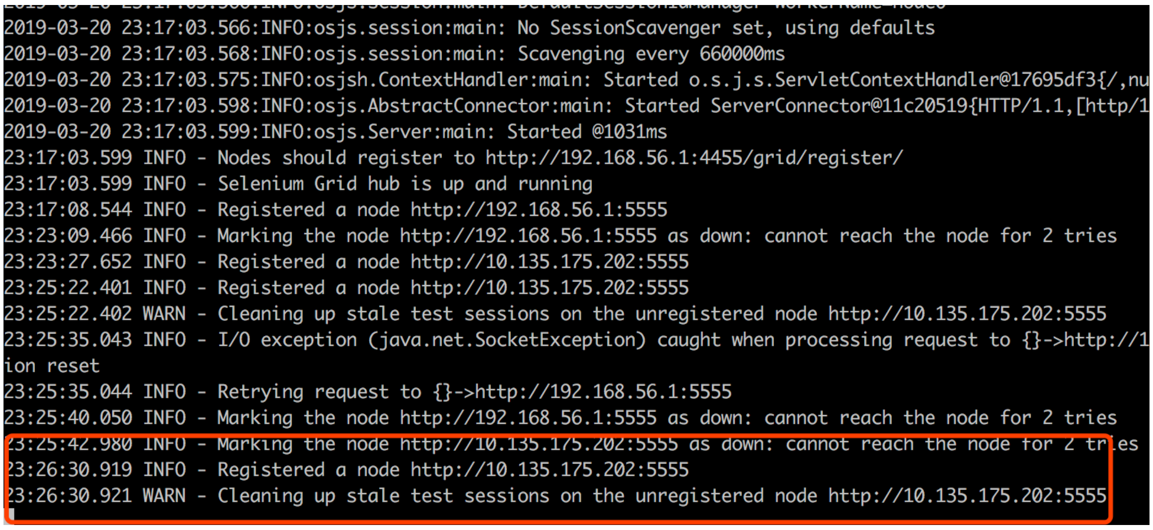


DefaultRemoteProxy (version : 3.141.59)
id : http://192.168.56.1:5555, OS : MAC

WebDriver
v:
v:
v:

启动node

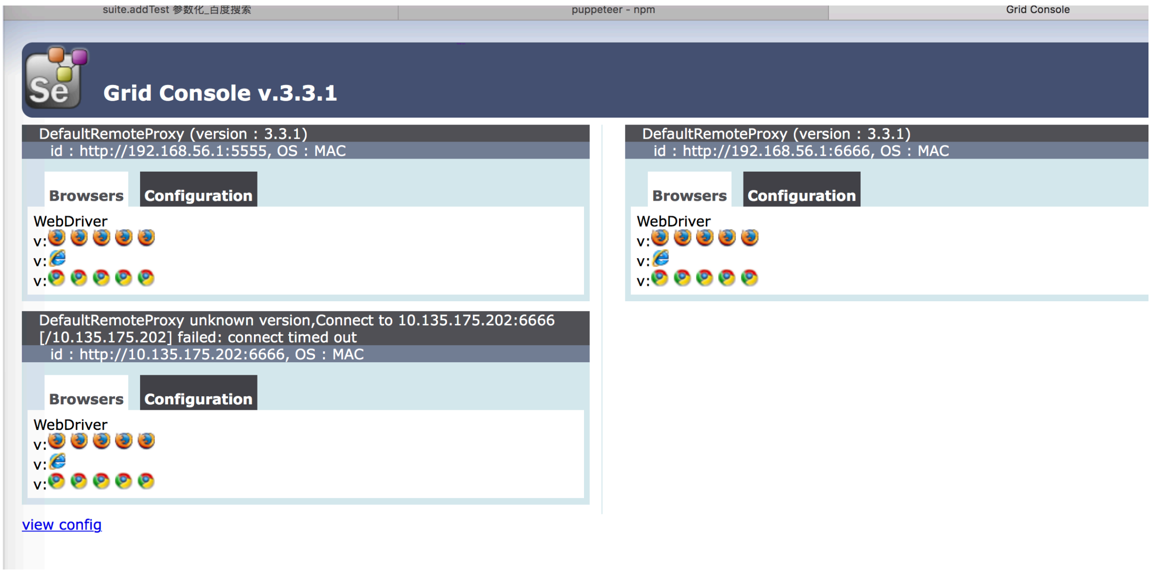
```
java -Dwebdriver.chrome.driver="/Users/xinxi/PycharmProjects/selenium_demo/webdriver/chromedriver_mac"
-jar selenium-server-standalone-3.3.1.jar -role node -port 5555
-hub http://192.168.56.1:4455/grid/registe
```



```
2019-03-20 23:17:03.566:INFO:osjs.session:main: No SessionScavenger set, using defaults
2019-03-20 23:17:03.568:INFO:osjs.session:main: Scavenging every 660000ms
2019-03-20 23:17:03.575:INFO:osjsh.ContextHandler:main: Started o.s.j.s.ServletContextHandler@17695df3{/,nu
2019-03-20 23:17:03.598:INFO:osjs.AbstractConnector:main: Started ServerConnector@11c20519{HTTP/1.1,[http/1
2019-03-20 23:17:03.599:INFO:osjs.Server:main: Started @1031ms
23:17:03.599 INFO - Nodes should register to http://192.168.56.1:4455/grid/register/
23:17:03.599 INFO - Selenium Grid hub is up and running
23:17:08.544 INFO - Registered a node http://192.168.56.1:5555
23:23:09.466 INFO - Marking the node http://192.168.56.1:5555 as down: cannot reach the node for 2 tries
23:23:27.652 INFO - Registered a node http://10.135.175.202:5555
23:25:22.401 INFO - Registered a node http://10.135.175.202:5555
23:25:22.402 WARN - Cleaning up stale test sessions on the unregistered node http://10.135.175.202:5555
23:25:35.043 INFO - I/O exception (java.net.SocketException) caught when processing request to {}->http://1
ion reset
23:25:35.044 INFO - Retrying request to {}->http://192.168.56.1:5555
23:25:40.050 INFO - Marking the node http://192.168.56.1:5555 as down: cannot reach the node for 2 tries
23:25:42.980 INFO - Marking the node http://10.135.175.202:5555 as down: cannot reach the node for 2 tries
23:26:30.919 INFO - Registered a node http://10.135.175.202:5555
23:26:30.921 WARN - Cleaning up stale test sessions on the unregistered node http://10.135.175.202:5555
```

查看节点log日志

```
http://localhost:4455/grid/console
```



Grid Console v.3.3.1

DefaultRemoteProxy (version : 3.3.1)
id : http://192.168.56.1:5555, OS : MAC

WebDriver
v:
v:
v:

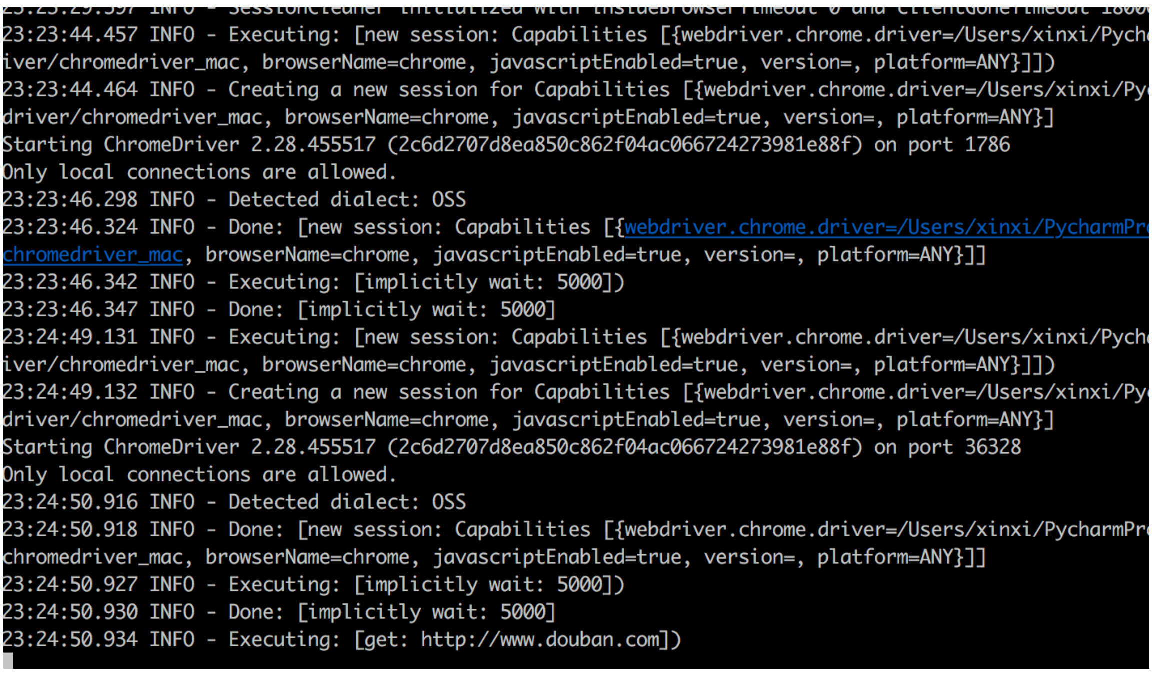
DefaultRemoteProxy (version : 3.3.1)
id : http://192.168.56.1:6666, OS : MAC

WebDriver
v:
v:
v:

DefaultRemoteProxy unknown version,Connect to 10.135.175.202:6666
[/10.135.175.202] failed: connect timed out
id : http://10.135.175.202:6666, OS : MAC

WebDriver
v:
v:
v:

多node测试



```
23:23:29.397 INFO - SessionCleaner initialized with insideBrowserTimeout 0 and clientGoneTimeout 1800
23:23:44.457 INFO - Executing: [new session: Capabilities [{webdriver.chrome.driver=/Users/xinxi/Pych
iver/chromedriver_mac, browserName=chrome, javascriptEnabled=true, version=, platform=ANY}]])
23:23:44.464 INFO - Creating a new session for Capabilities [{webdriver.chrome.driver=/Users/xinxi/Py
driver/chromedriver_mac, browserName=chrome, javascriptEnabled=true, version=, platform=ANY}]
Starting ChromeDriver 2.28.455517 (2c6d2707d8ea850c862f04ac066724273981e88f) on port 1786
Only local connections are allowed.
23:23:46.298 INFO - Detected dialect: OSS
23:23:46.324 INFO - Done: [new session: Capabilities [{webdriver.chrome.driver=/Users/xinxi/PycharmPr
chromedriver_mac, browserName=chrome, javascriptEnabled=true, version=, platform=ANY}]]
23:23:46.342 INFO - Executing: [implicitly wait: 5000])
23:23:46.347 INFO - Done: [implicitly wait: 5000]
23:24:49.131 INFO - Executing: [new session: Capabilities [{webdriver.chrome.driver=/Users/xinxi/Pych
iver/chromedriver_mac, browserName=chrome, javascriptEnabled=true, version=, platform=ANY}]])
23:24:49.132 INFO - Creating a new session for Capabilities [{webdriver.chrome.driver=/Users/xinxi/Py
driver/chromedriver_mac, browserName=chrome, javascriptEnabled=true, version=, platform=ANY}]
Starting ChromeDriver 2.28.455517 (2c6d2707d8ea850c862f04ac066724273981e88f) on port 36328
Only local connections are allowed.
23:24:50.916 INFO - Detected dialect: OSS
23:24:50.918 INFO - Done: [new session: Capabilities [{webdriver.chrome.driver=/Users/xinxi/PycharmPr
chromedriver_mac, browserName=chrome, javascriptEnabled=true, version=, platform=ANY}]]
23:24:50.927 INFO - Executing: [implicitly wait: 5000])
23:24:50.930 INFO - Done: [implicitly wait: 5000]
23:24:50.934 INFO - Executing: [get: http://www.douban.com])
```

linux下执行

离线安装:setuptools、selenium

安装allure

```
wget https://github.com/allure-framework/allure-core/releases/download/allure-core-1.5.2/allure-
commandline.tar.gz

tar -xvf allure-commandline.tar.gz

cd bin/
ls
allure  allure.bat

vi /etc/profile

export allure_home=/root
export PATH=${allure_home}/bin:${PATH}

 source /etc/profile
```

学习贴

linux下执行
https://blog.csdn.net/qq_41963758/article/details/80320309

selenium grid
https://www.jianshu.com/p/fb1587fb0822