1.js处理滚动条

以cms后台为例，进入卡券核销列表页后，屏幕下方的页码切换元素是不能直接被定位到的；如果尝试定位会抛出元素不可见的异常。而selniun也没有提供方法能够直接操作滚动条，但是selenium提供了操作js的方法：execute_script(),我们可以通过此方法借助js来实现窗口/页面滚动的功能。

```
# 滚动条定位到页面顶部
js="var q=document.getElementById('id').scrollTop=0"
driver.execute_script(js)

# 滚动条定位到页面底部
js="var q=document.documentElement.scrollTop=10000"
driver.execute_script(js)

# 实现窗口左右滚动
scrollLeft=10000
scrollLeft=0
```

1.1 采用元素聚焦的方式，让页面直接跳到元素出现的位置，哪怕元素当前不可见。

```
# 以cms后台为例，进入卡券核销页后不滚动屏幕，先定位底部的页码元素后通过
scrollIntoView()实现窗口滚动
target = driver.find_element_by_xpath('//*
[@id="content"]/div/div[3]/div/div/span[2]/div/div')
driver.execute_script("arguments[0].scrollIntoView();", target)
```

1.2 对于不可见的元素，如果仅需要点击这个元素，可以直接使用js执行click()方法的方式（直接点击不可见的目标元素，不用先进行页面滚动或者跳转），用法如下：

```
# 以cms后台为例，进入卡券核销页后不滚动屏幕，通过js直接点击底部的页码切换按扭
ele = driver.find_element_by_xpath('//*
[@id="content"]/div/div[3]/div/div/span[2]/div/div')
driver.execute_script("arguments[0].click();", ele)
```

1.3 如果不使用js直接点击的方式，也可以通过id、name、tagname、class、css selector等方式定位到元素后执行js滚动操作。以cms后台为例，我们可以通过class属性定位到右侧可滚动窗口，然后执行js脚本来实现窗口滚动。

```
# 通过class定位到滚动窗口后，用js实现窗口滚动
js = 'document.getElementsByClassName("content")[0].scrollTop=10000'
driver.execute_script(js)

# js支持如下5种元素选取方式
document.getElementById("id")     # 通过id获取
document.getElementsByName("Name")     # 通过Name选取元素
document.getElementsByTagName("tag")     # 通过标签名选取元素
document.getElementsByClassName("class")     # 通过class名选取元素
document.querySelectorAll("css selector")     # 通过css选择器选取元素
```

2.js处理日期控件/修改控件属性

js除了可以帮助我们实现页面的滚动、元素点击等操作外，还可以修改页面内控件的属性，如将只读控件变为可编辑、实现文本输入/修改等以满足我们的测试需要。用js去掉元素属性基本思路：先定位到元素，然后用removeAttribute("readonly")方法删除属性。

2.1 使用js修改控件的属性

以12306网站的日期控件为例，该日期控件不可以直接输入内容，查看该控件有一个readonly属性。我们利用js将此属性删除后就可以实现内容输入了。

```
from selenium import webdriver
driver = webdriver.Chrome()
driver.get("https://kyfw.12306.cn/otn/index/init")
# 去掉元素的readonly属性
js = 'document.getElementById("train_date").removeAttribute("readonly");'
driver.execute_script(js)
# 清空文本后输入值
driver.find_element_by_id("train_date").clear()
driver.find_element_by_id("train_date").send_keys("2019-06-02")
# 用js方法输入日期
js_value = 'document.getElementById("train_date").value="2020-06-02"'
driver.execute_script(js_value)
```

2.2 使用js处理句柄/多窗口

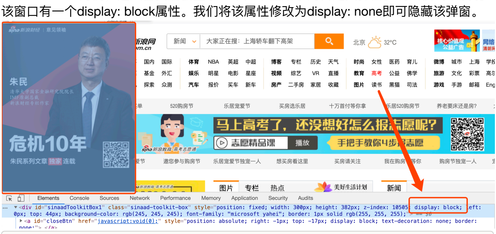
在第3节中我们通过driver的内置api（driver.switch_to.window()）来操作浏览器多个窗口的切换。在多个窗口之间来回切换无疑会增加我们的脚本复杂性。借助js，我们可以轻松地解决这个问题。

以sina.com为例，进入调试模式即可发现页面内超链接有一个相同的属性target='_blank',想要解决多窗口的问题，去掉这个属性就OK了。

```
from selenium import webdriver
import time
driver = webdriver.Chrome()
driver.get("https://www.sina.com/")
# 修改元素的target属性
js = 'document.querySelectorAll("body > div.main > div.main-nav >
div:nth-child(1) > ul:nth-child(1) > li:nth-child(1) > a")[0].target="";'
driver.execute_script(js)
time.sleep(3)
driver.find_element_by_css_selector('body > div.main > div.main-nav >
div:nth-child(1) > ul:nth-child(1) > li:nth-child(1) > a').click()
```

2.3 利用js关闭自定义弹窗

以sina.com为例，进入首页后会显示一个如下图所示的自定义弹窗。调试模式下发现该窗口有一个display: block属性。我们将该属性修改为display: none即可隐藏该弹窗。



```
from selenium import webdriver
import time
driver = webdriver.Chrome()
driver.maximize_window()
driver.get("https://www.sina.com/")
time.sleep(8)

# 修改元素的display属性为none
#js =
'document.getElementById("sinaadToolkitBox1").style.display="none";'
js = 'document.getElementsByClassName("sinaad-toolkit-box")
[0].style.display="none"'
driver.execute_script(js)
```