



Selenium – Web Automation

James R. Small, Principal Architect

Michigan! /usr/group

mug.org – A Free and Open Source Michigan Community

Assumed Background

- You are comfortable with Python – this could also be done in the other core/popular languages: C#, Ruby, Java, JavaScript, PHP
- You are comfortable with HTML, CSS, and the basics of JavaScript
- You are comfortable with Chrome and its DevTools
- Experience with web app development and testing a plus

About Selenium

- Created at Thoughtworks in 2004
 - Version 2 released in 2007
 - Version 3 released in 2016
 - Version 3 (WebDriver) standardized under the W3C in 2019
 - Version 4 paint still drying – released October 2021
 - Version 4 is also W3C standard
 - WebDriver
 - WebDriver Protocol also – more powerful

How Popular is Selenium?

According to Enlyft:

- Selenium dominates testing tools
- Over 54,000 users

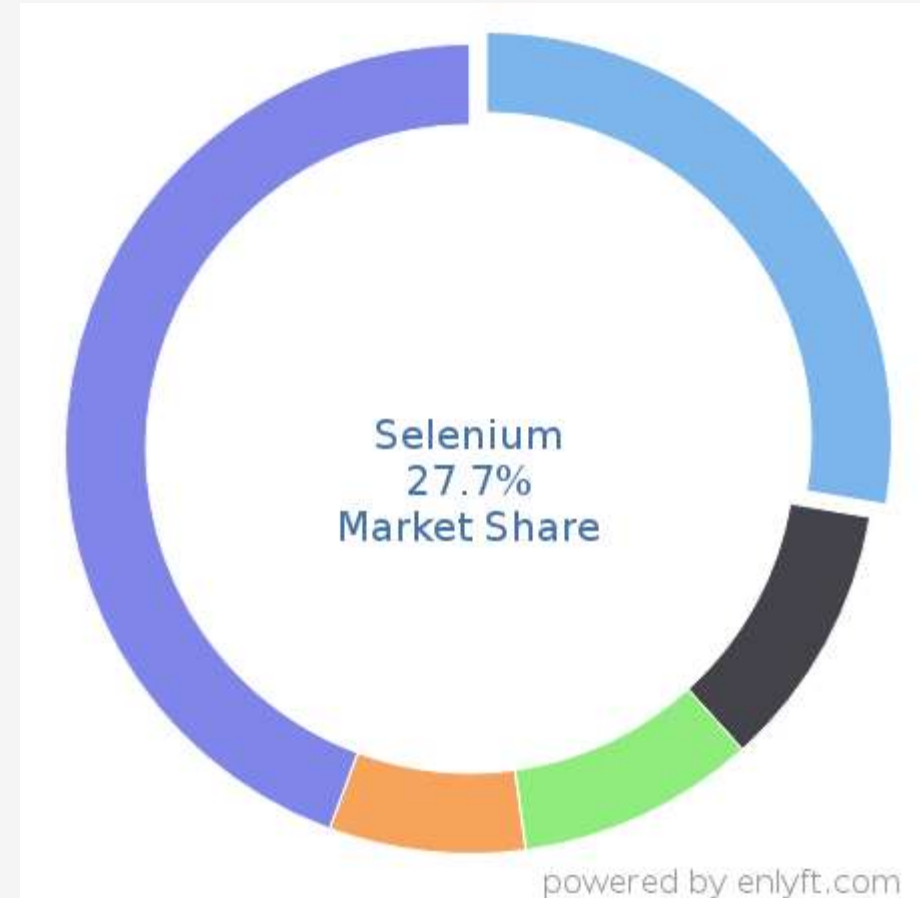
Software Testing Tools

Selenium (27.70%)

Apache Jmeter (10.62%)

HP Unified Functional Testing (9.51%)

HP Quality Center (7.83%)



Why is Selenium so Popular?

- Allows controlling browsers through your language of choice
- Free and open source
- Google became an avid user, created/contributed Grid
- Standardized (WebDriver) through the W3C
- Portable – Windows, macOS, Linux, iOS/Android (3rd party)
- Integrates with CI/CD, development platforms, SaaS options

What Exactly is Selenium?

- WebDriver – core of Selenium, allows browser automation
- IDE – browser extension (Firefox, Chrome, Edge)
 - Allows recording and playback, essentially a no/low-code option
- Grid – fully distributed testing solution, completely reworked in v4

Languages Supported by Selenium WebDriver

Core:



Usage: 17.6%



Usage: 6.1%



Usage: 67%



Usage: 31%



Usage: 21.4%

3rd Party:

- PHP (1.5% Usage, supported by Facebook)
- Also Go, Haskell, Perl, R, Dart

Browsers Supported by Selenium WebDriver



Usage:

67%



33%



98%



Usage:

?%



13.5%



29%

Use Case

- Asked to help support web app
- App was out of date – components out of support for years
- Business unit owner afraid to allow changes – would frequently break app
- No tests – basic manual inspection done on demand

Step One

- Work with application analyst to define tests
- Document tests including screenshots
- Whenever changes made to app:
 - Analyst goes through document to validate functionality
 - Tedious, slow, manual process...

Problems with Manual Testing

- In addition to being tedious and slow:
 - Error prone
 - Often skipped for “small” changes
- What about testing in multiple browsers and/or versions?
- Mobile???
- Way to provide fast, consistent feedback?

Step Two

- I had heard of Selenium and thought it sounded interesting
- Seemed like a good fit, so...
 - Install Python
 - `pip install selenium`

Getting Started with Selenium



Selenium with Python

Author: [Baiju Muthukadan](#)

License: This document is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](#).

Navigation

- [1. Installation](#)
- [2. Getting Started](#)
- [3. Navigating](#)
- [4. Locating Elements](#)
- [5. Waits](#)
- [6. Page Objects](#)
- [7. WebDriver API](#)
- [8. Appendix: Frequently Asked Questions](#)

Related Topics

[Documentation overview](#)

- Next: [1. Installation](#)

Quick search

Note:

This is not an official documentation. If you would like to contribute to this documentation, you can [fork this project in GitHub](#) and [send pull requests](#). You can also send your feedback to my email: baiju.m.mail AT gmail DOT com. So far 50+ community members have contributed to this project (See the closed pull requests). I encourage contributors to add more sections and make it an awesome documentation! If you know any translation of this document, please send a PR to update the below list.

Translations:

- [Chinese](#)
- [Japanese](#)

- [1. Installation](#)
 - [1.1. Introduction](#)
 - [1.2. Installing Python bindings for Selenium](#)
 - [1.3. Instructions for Windows users](#)
 - [1.4. Installing from Git sources](#)
 - [1.5. Drivers](#)

Hello, World – 3.x:

```
1  #! /usr/bin/env python3.10
2
3  import time
4
5  # old/3.x:
6  from selenium import webdriver
7
8  # Need to download Chrome web driver:
9  # Go to Selenium home page, downloads, Browsers and follow link to Chrome:
10 # https://www.selenium.dev/downloads/
11 # https://chromedriver.chromium.org/downloads
12 # Make sure chromedriver.exe is in PATH
13 # Alternatively can pass location:
14 # driver = webdriver.Chrome(executable_path=r'C:\path\to\chromedriver.exe')
15
16 driver = webdriver.Chrome()
17 driver.get('https://selenium.dev')
18 time.sleep(3)
19 driver.quit()
20
```

Hello, World – 4.x:

```
1 #! /usr/bin/env python3.10
2
3 import time
4
5 # New/4.x:
6 from selenium import webdriver
7 from selenium.webdriver.chrome.service import Service as ChromeService
8
9 # Need to download Chrome web driver:
10 # Go to Selenium home page, downloads, Browsers and follow link to Chrome:
11 # https://www.selenium.dev/downloads/
12 # https://chromedriver.chromium.org/downloads
13 # Make sure chromedriver.exe is in PATH
14 # Alternatively can pass location:
15 # service = ChromeService(executable_path=r'C:\path\to\chromedriver.exe')
16 # driver = webdriver.Chrome(service=service)
17
18 driver = webdriver.Chrome(service=ChromeService())
19 driver.get('https://selenium.dev')
20 time.sleep(3)
21 driver.quit()
22
```

Tedium...

- Must download a driver for each browser
- Driver must match browser (32/64-bit, version)
- Browsers auto-upgrade
- If driver doesn't match browser get strange errors...



Python WebDriver Manager

- `pip install webdriver-manager`
- Auto-downloads right driver!



Hello, World – 4.x, improved:

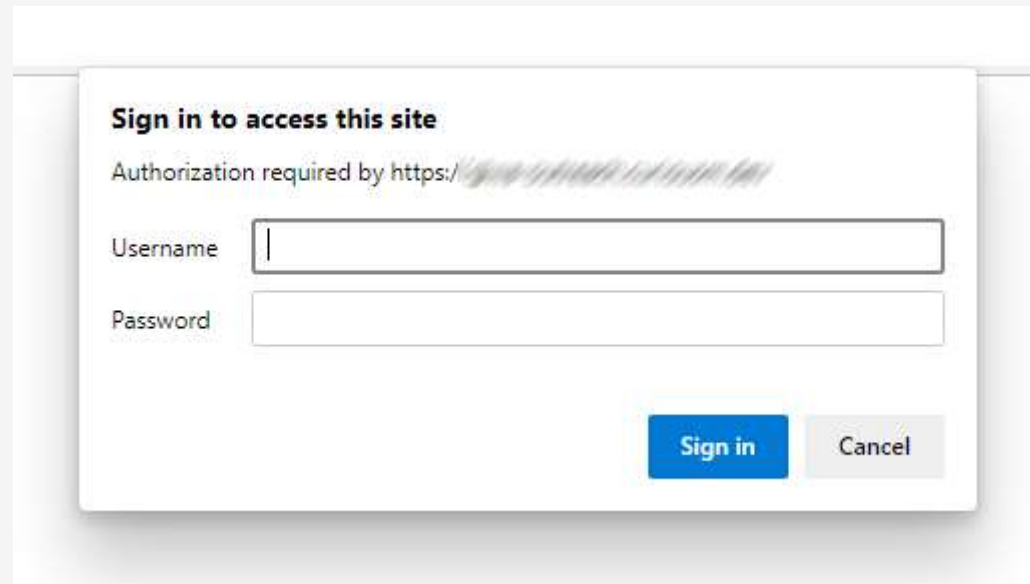
```
1 #! /usr/bin/env python3.10
2
3 import time
4
5 # New/4.x:
6 from selenium import webdriver
7 from selenium.webdriver.chrome.service import Service as ChromeService
8 from webdriver_manager.chrome import ChromeDriverManager
9
10 driver = webdriver.Chrome(service=ChromeService(ChromeDriverManager().install()))
11 driver.get('https://selenium.dev')
12 time.sleep(3)
13 driver.quit()
14
```

Step Three – Convert Manual Tests into Automated Ones

Test Environment for App:

- Login to test site
- Login to app
- From starting calendar, navigate to specific date
- Select specific meeting
- Find link to download PDF attachment for meeting
- Confirm PDF retrieved and validate it's the right one

Login to test site



Sign in to access this site

Authorization required by https://~~https://www.testsite.com~~

Username

Password

Login to test site

Accessing the test site requires basic authentication – Selenium provides a few options:

1. Encode into get request: `https://<username>:<password>@<URL>`

```
1 TEST_URL = 'app.example.com'
2 ACCESS_USERNAME = 'accessuser'
3 ACCESS_PASSWORD = 'accesspass'
4
5 # Initial access login:
6 driver.get(f'https://{ACCESS_USERNAME}:{ACCESS_PASSWORD}@{TEST_URL}')
7
```

2. JavascriptExecutor for basic HTTP Authentication (not needed in my use case)
3. In 4.x, new BiDirectional API – can register basic auth, but couldn't figure out how to get it to work in Python...

Login to App

REQUEST ACCESS

First Name *

Last Name *

Organization *

Email Address *

Phone Number *

Role *

REGISTER

LOGIN

Email Address

Password

LOGIN

[Forgot Your Password?](#)

Login to App

App Access requires user login:

1. Enter user Email address
2. Enter user Password
3. Click Login

We need to do programmatically:

- Finding web elements within the DOM
- Simulate user input/interaction

Login to App – Locate/Complete Forms with WebDriver Locators

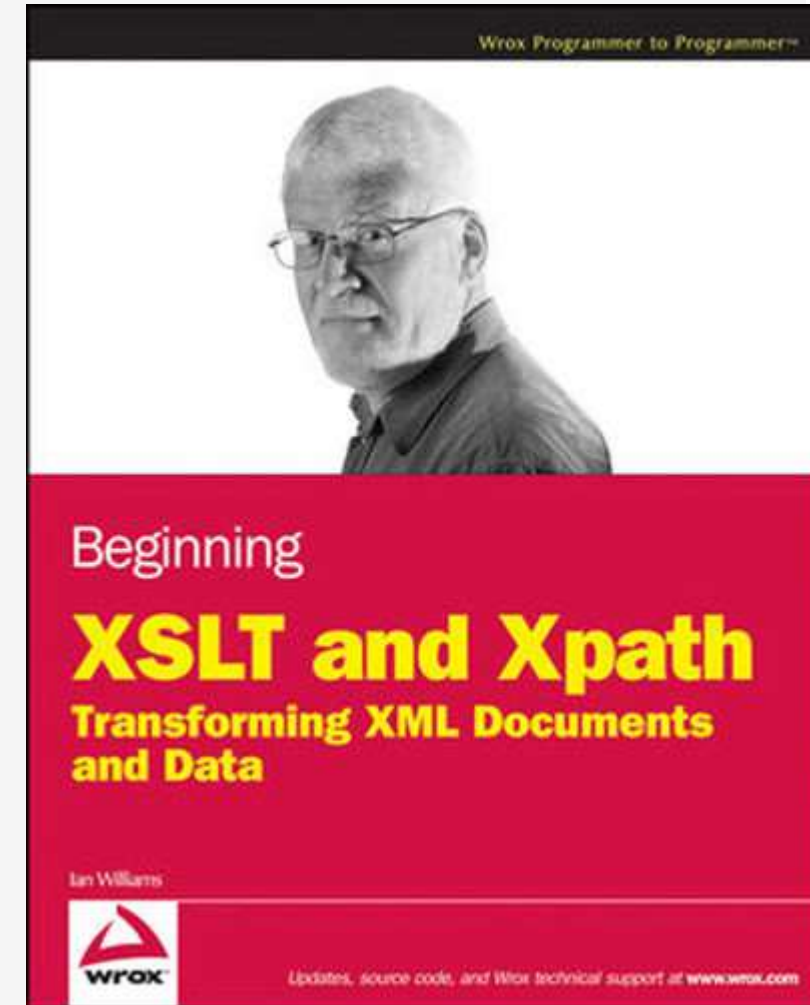
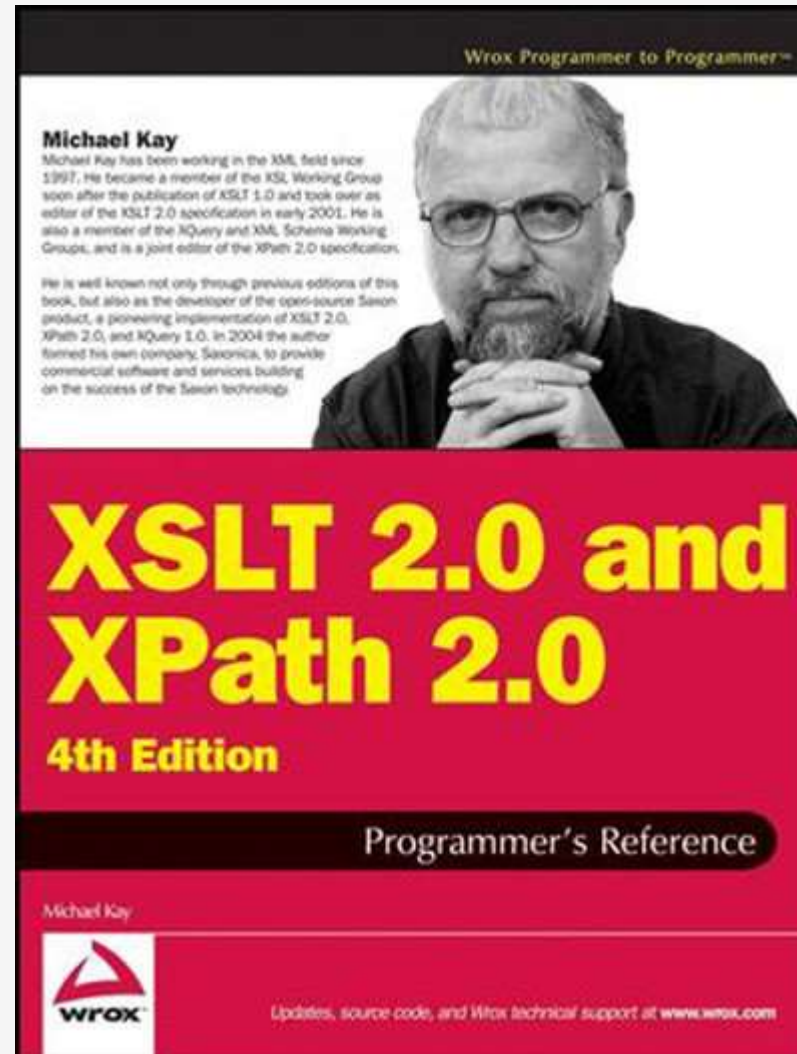
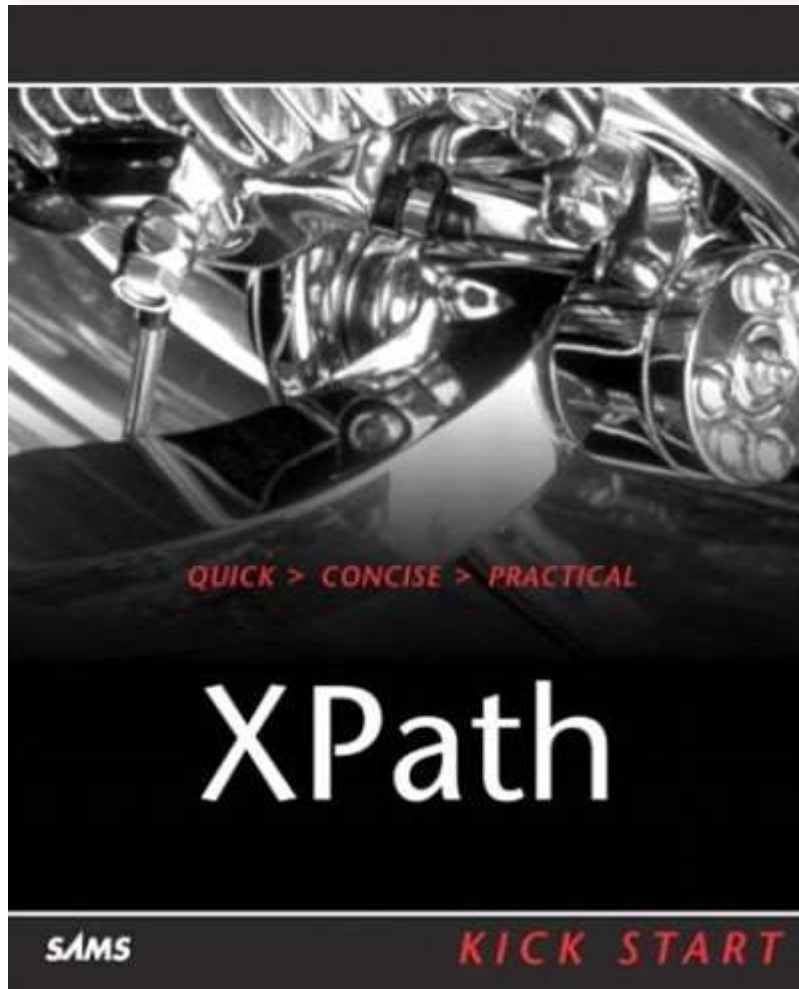
Locator	Example
Class Name	<code><p class="content">Site content goes here.</p></code> <code>driver.find_element(By.CLASS_NAME, 'content')</code>
CSS Selector	<code><p class="content">Site content goes here.</p></code> <code>driver.find_element(By.CSS_SELECTOR, 'p.content')</code>
ID	<code><form id="loginForm"></code> <code>driver.find_element(By.ID, 'loginForm')</code>
Link Text/Partial Link Text	<code>Continue</code> <code>driver.find_element(By.LINK_TEST, 'Continue')</code> <code>driver.find_element(By.PARTIAL_LINK_TEST, 'Conti')</code>
Name	<code><input name="username" type="text" /></code> <code>driver.find_element(By.NAME, 'username')</code>
Tag Name	<code><h1>Welcome</h1></code> <code>driver.find_element(By.TAG_NAME, 'h1')</code>
XPath	<code><input name="username" type="text" /></code> <code>driver.find_element(By.XPATH, '//input[@name="username"]')</code>

Which WebDriver Locator(s) to Use?

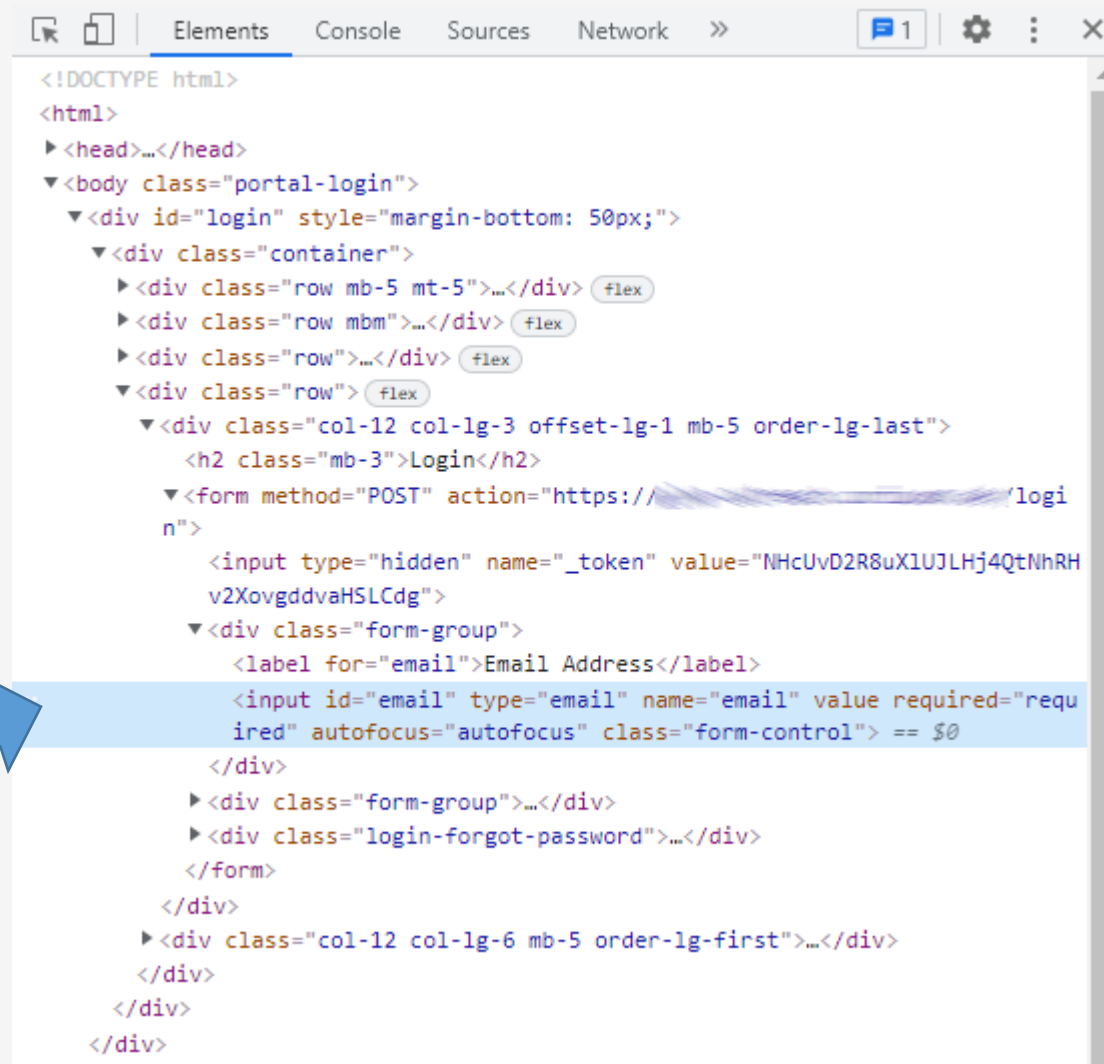
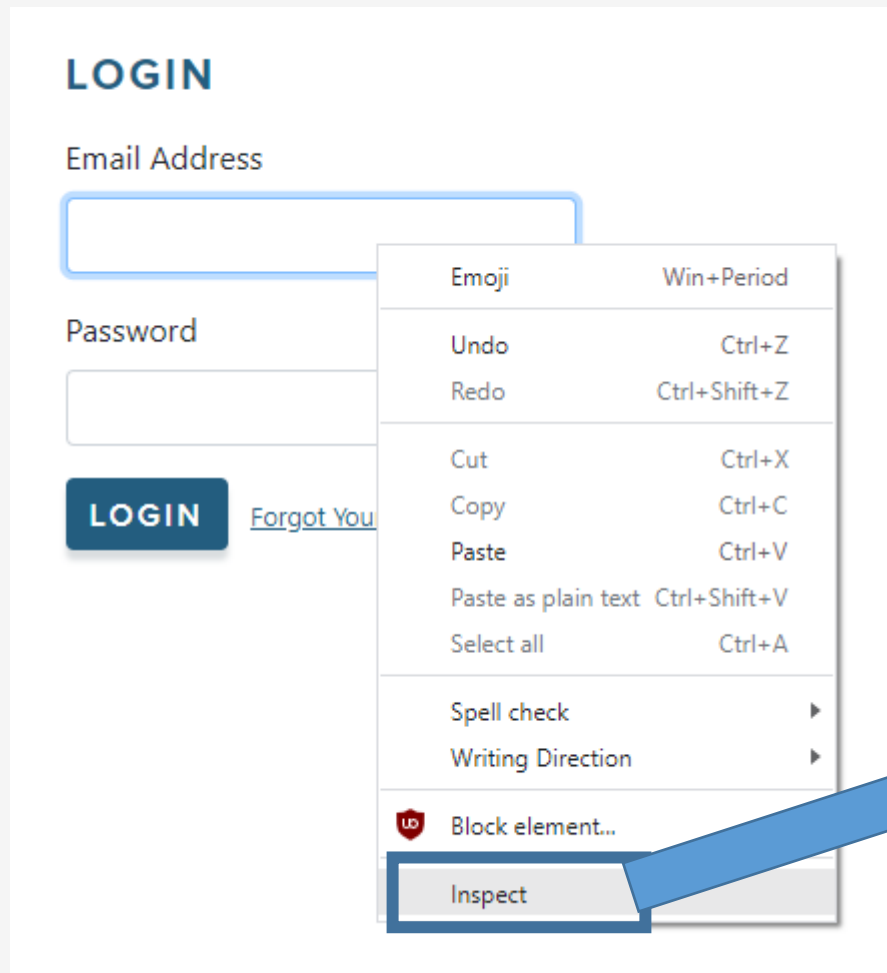
From the [Automation Panda](#):

1. ID (if unique)
2. Name (if unique)
3. Class Name
4. CSS Selector
5. XPath without text or indexing
6. Link Text / Partial Link Text
7. XPath with text and/or indexing

Why is XPath at the bottom?



Login to App – Find Web Elements and Appropriate Locators



Login to App – Ensure Locator is Unique

From Chrome DevTools:

- Use Control-F to search for Locator
- Locator: ID="email"
- Search for "#email"
- Note: Chrome states 1 of 1 – it's unique!



Login to App – Find Web Elements and Appropriate Locators

- User Email address – Unique ID
- User Password – also has Unique ID
- Login Button – no unique ID...



```
<div class="login-button">
```

```
<button type="submit" class="btn btn-primary">Login</button>
```

```
</div>
```

Use XPATH: `'//*[@class="login-button"]/button'`


Login to App – Code

```
1 # To support By.<name> in find_element:
2 from selenium.webdriver.common.by import By
3
4 TEST_URL = 'app.example.com'
5 ACCESS_USERNAME = 'accessuser'
6 ACCESS_PASSWORD = 'accesspass'
7 SITE_USERNAME = 'test@example.com'
8 SITE_PASSWORD = 'testpass'
9
10 # Initial access login:
11 driver.get(f'https://{ACCESS_USERNAME}:{ACCESS_PASSWORD}@{TEST_URL}')
12
13 # App - user login:
14 driver.find_element(By.ID, 'email').send_keys(SITE_USERNAME)
15 driver.find_element(By.ID, 'password').send_keys(SITE_PASSWORD)
16 driver.find_element(By.XPATH, '//*[@class="login-button"]/button').click()
17
```

From starting calendar view, navigate to specific date

JANUARY 2022

TODAY < > List View Select Month ▼ Select Year ▼



SUN	MON	TUE	WED	THU	FRI	SAT
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

From starting calendar view, navigate to specific date

JUNE 2022						
SUN	MON	TUE	WED	THU	FRI	SAT
			1	2	3	
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

From starting calendar view, navigate to specific date

JUNE 2022

TODAY < > List View June Select Year

SUN	MON	TUE	WED	TH
			1	
5	6	7	8	
12	13	14	15	
19	20	21	22	23
26	27	28	29	30

- Back Alt+Left Arrow
- Forward Alt+Right Arrow
- Reload Ctrl+R
- Save as... Ctrl+S
- Print... Ctrl+P
- Cast...
- Search images with Google Lens
- Create QR Code for this page
- Translate to English
- View page source
- Inspect

```
<div class="col-md-12">
  <div class="row">...</div>
  <div class="row">...</div>
  <div id="fullCalendar" class="fc fc-unthemed fc-ltr">
    <div class="fc-toolbar fc-header-toolbar">
      <div class="fc-left">...</div>
      <div class="fc-right">
        <button type="button" class="fc-today-button fc-button fc-state
        orner-left fc-corner-right">Today</button>
        <div class="fc-button-group">...</div>
        <button type="button" class="fc-myCustomButton-button fc-button
        ault fc-corner-left fc-corner-right">List View</button>
      <div class="form-group">
        <select class="select_month form-control month_dp" style="wid
        == $0
        <option value>Select Month</option>
        <option value="01">January</option>
        <option value="02">February</option>
        <option value="03">March</option>
        <option value="04">April</option>
        <option value="05">May</option>
        <option value="06">June</option>
        <option value="07">July</option>
        <option value="08">August</option>
        <option value="09">September</option>
        <option value="10">October</option>
        <option value="11">November</option>
        <option value="12">December</option>
      </select>
    </div>
    <div class="form-group" style="margin-left:7px;width:150px;">...</div>
  </div>
</div>
```

select_month 1 of 1 Cancel

Chrome can help with XPath Expressions

```
<div class="form-group">
  <select class="select_month form-control month_dp" style="width:150px;"
    == $0
    <option value>Select Month</option>
    <option value="01">January</option>
    <option value="02">February</option>
    <option value="03">March</option>
    <option value="04">April</option>
    <option value="05">May</option>
    <option value="06">June</option>
    <option value="07">July</option>
    <option value="08">August</option>
    <option value="09">September</option>
    <option value="10">October</option>
    <option value="11">November</option>
    <option value="12">December</option>
  </select>
</div>
<div class="form-group" style="margin-left:7px;width:150px;">...</div>
<div class="fc-center"></div>
<div class="fc-clear"></div>
</div>
```

... serindex.kt-pagebody.container div.row.mbm div.col-md-12 div#fullCalendar.fc.fc-unthe ...

`//*[@class="select_month"]/option` 0 of 0 Cancel

Console Search

Filter Default levels 1 Issue: 1

```
> $x('//select[@class="select_month form-control month_dp"]/option[7]')
< ▶ [option]
8 landing:441
> $x('//select[@class="select_month form-control month_dp"]/option[@value="06"]')
< ▶ [option]
9 landing:441
10 landing:441
0 landing:441
>
```

Navigate to Specific Date – Code

```
12
13 # App - user login:
14 driver.find_element(By.ID, 'email').send_keys(SITE_USERNAME)
15 driver.find_element(By.ID, 'password').send_keys(SITE_PASSWORD)
16 driver.find_element(By.XPATH, '//*[@class="login-button"]/button').click()
17
18 # select June, 2021 calendar:
19 driver.find_element(By.CLASS_NAME, 'select_month').click()
20 driver.find_element(By.XPATH,
21     '//select[@class="select_month form-control month_dp"]/option[@value="06"]').click()
22 driver.find_element(By.ID, 'calendar-year-dp').click()
23 driver.find_element(By.XPATH,
24     '//select[@id="calendar-year-dp"]/option[@value="2021"]').click()
25
```

Select Specific Meeting


June 2021					
Operations & Technology Committee Meeting	Hybrid	100th Board	Wednesday, June 9th 2021	11:00 am	1:00 pm
June 2021					
Legal & Risk Board of Directors Meeting	-	100th Board	Wednesday, June 9th 2021	1:00 pm	3:00 pm
June 2021					
Executive Leadership Team	Zoom	100th Board	Thursday, June 10th 2021	12:00 am	3:00 pm
June 2021					
100th Committee Meeting	Hybrid	100th Board	Tuesday, June 15th 2021	10:00 am	12:00 am
June 2021					
Asset Management and Portfolio	Zoom	Zoom	Wednesday, June 23rd 2021	8:30 am	10:00 am
June 2021					
Legal Committee Meeting	-	100th Board	Wednesday, June 23rd 2021	1:00 pm	2:00 pm
June 2021					
Board of Directors Meeting	-	100th Board	Wednesday, June 23rd 2021	2:00 pm	4:00 pm



Flakiness and Race Conditions

- When try to follow link to specific meeting, sometimes it fails:

```
Traceback (most recent call last):
  File "C:\working\github\sockduct\selenium\example.py", line 92, in <module>
    main()
  File "C:\working\github\sockduct\selenium\example.py", line 59, in main
    driver.find_element(By.LINK_TEXT, 'Committee Meeting').click()
  File "C:\working\github\sockduct\selenium\.venv\lib\site-packages\selenium\webdriver\remote\
webdriver.py", line 1244, in find_element
    return self.execute(Command.FIND_ELEMENT, {
  File "C:\working\github\sockduct\selenium\.venv\lib\site-packages\selenium\webdriver\remote\
webdriver.py", line 424, in execute
    self.error_handler.check_response(response)
  File "C:\working\github\sockduct\selenium\.venv\lib\site-packages\selenium\webdriver\remote\
errorhandler.py", line 247, in check_response
    raise exception_class(message, screen, stacktrace)
selenium.common.exceptions.NoSuchElementException: Message: no such element: Unable to locate
element: {"method":"link text","selector":"Committee Meeting"}
(Session info: chrome=96.0.4664.110)
Stacktrace:
Backtrace:
    ordinal0 [0x00A56903+2517251]
    (...)
```



Allowing Time for Web Elements to Load:

```
9
10 # Adjust as necessary:
11 WAIT_TIME = 3
12
13 # Add implicit wait time - allow time for everything to load on page:
14 driver.implicitly_wait(WAIT_TIME)
15
```

Select Specific Meeting – Code

```
15
16 # Initial access login:
17 driver.get(f'https://{ACCESS_USERNAME}:{ACCESS_PASSWORD}@{TEST_URL}')
18
19 # App - user login:
20 driver.find_element(By.ID, 'email').send_keys(SITE_USERNAME)
21 driver.find_element(By.ID, 'password').send_keys(SITE_PASSWORD)
22 driver.find_element(By.XPATH, '//*[@class="login-button"]/button').click()
23
24 # select June, 2021 calendar:
25 driver.find_element(By.CLASS_NAME, 'select_month').click()
26 driver.find_element(By.XPATH,
27     '//*[@class="select_month form-control month_dp"]/option[@value="06"]').click()
28 driver.find_element(By.ID, 'calendar-year-dp').click()
29 driver.find_element(By.XPATH,
30     '//*[@id="calendar-year-dp"]/option[@value="2021"]').click()
31
32 # select June 15, 2021 Meeting:
33 driver.find_element(By.LINK_TEXT, 'Committee Meeting').click()
34
```

Find Link to Download PDF Attachment

Calendar

[Back to calendar](#)

CIP COMMITTEE MEETING

Tuesday, June 15th 2021 / 10:00 am - 12:00 am

Telephonic

Board Meetings

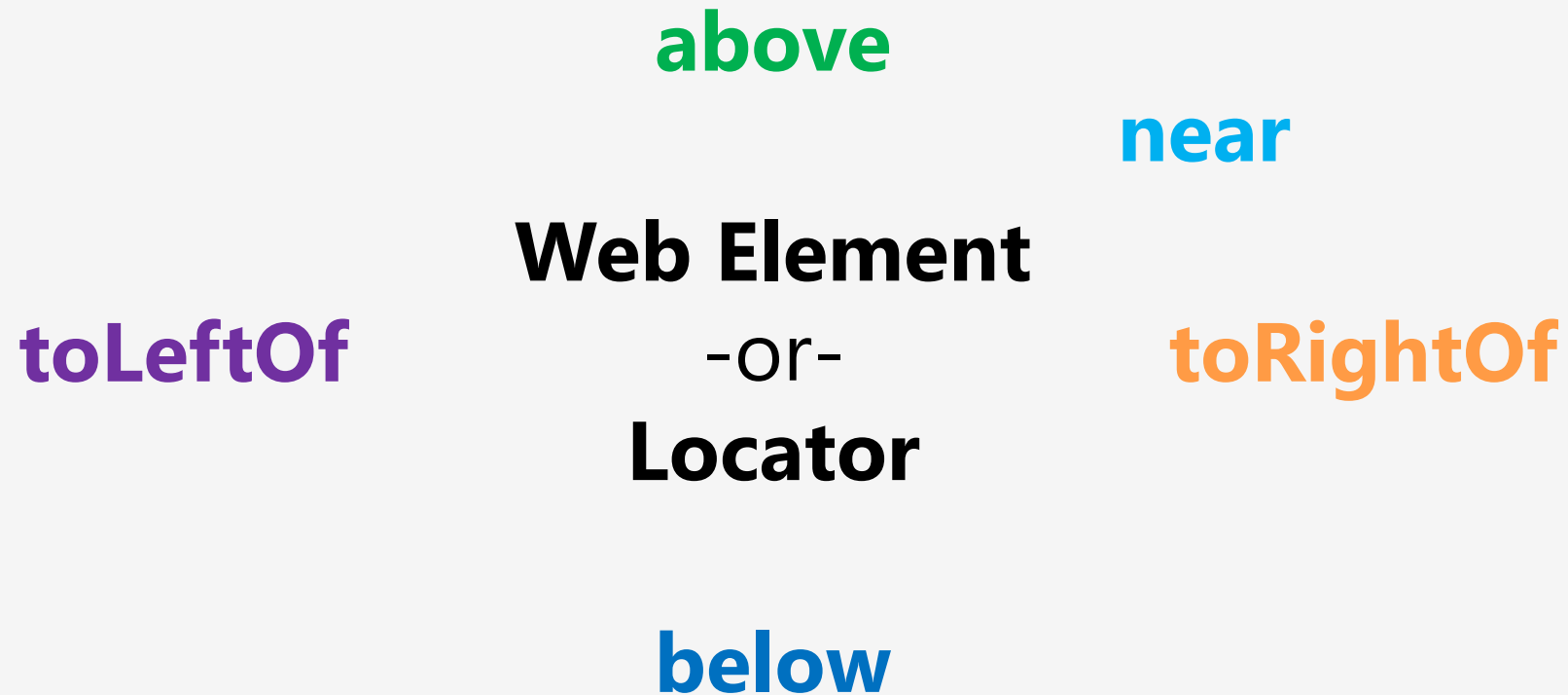
Call-In Number: 1-800-775-3889 Conference ID: 775-388-3889

CIP Committee Meeting - June 15 2021
Capital Planning Committee .pdf



```
▼<div class="kt-pagebody container">
  ▶<div class="row">...</div> flex
  ▶<div class="row mtm">...</div> flex
  ▼<div class="row"> flex
    ▼<table id="attachment-table">
      ▼<tbody>
        ▼<tr>
          ▶<td>...</td>
          ▼<td style="width: 50px;">
            ▼<a href="https://
              0.pdf" target="_blank"> == $0
                ▶<i class="fa fa-external-link">...</i>
              </a>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  <br>
```


Selenium 4 New Feature – Relative Locators



Find Link to Download PDF Attachment – Code

```
31
32 # select June 15, 2021 Meeting:
33 driver.find_element(By.LINK_TEXT, 'Committee Meeting').click()
34
35
36 # To support Selenium 4 relative locators:
37 from selenium.webdriver.support.relative_locator import locate_with
38
39 # Find PDF URL for Meeting:
40 agenda_text = driver.find_element(By.XPATH,
41     '//td/div[contains(text(), '
42     '"Notice - June 15 2021 Planning Committee.pdf")]')
43 pdf_url = driver.find_element(locate_with(By.TAG_NAME, 'a').
44     to_right_of(agenda_text)).get_attribute('href')
45
```

Retrieve, Parse, and Validate PDF

- Have URL
- Don't need to save file – retrieve as in-memory byte stream
- Use a library to parse the file – PyPDF2
- Look for key strings in the file (or however you want to confirm the PDF)

Retrieve and Parse PDF – Code

```
45
46
47 # To retrieve and parse PDF:
48 from io import BytesIO
49 from urllib import request
50 from PyPDF2 import PdfFileReader
51
52 # Download PDF to a byte stream:
53 with request.urlopen(pdf_url) as resp:
54     pdf_stream = BytesIO(resp.read())
55
56 # Parse PDF:
57 pdf_doc = PdfFileReader(pdf_stream)
58 pdf_page = pdf_doc.getPage(0)
59 pdf_text = pdf_page.extractText().replace('\n', '').lower()
60
61 # Define Document Checks and Validate PDF:
62 page_checks = ['public notice regarding', 'june 15, 2021, 10:00 a.m. meeting ',
63               'of the planning committee of the company board of directors']
64 if any(page_check not in pdf_text for page_check in page_checks):
65     print('Wrong Document...Dooooh!')
66 else:
67     print('Hooray - all tests pass!!!')
68
```

Next Steps

- No Error Handling - Address
- This presentation focuses on using Selenium, but...
- This is a set of end-to-end tests – need to wrap this code in a test framework:
 - Each step will be a test
 - The framework will show where things fail in a report
- Recommend using pytest

Free Hands-On Web App Test Automation Tutorial (3 hours)

TUTORIAL: HANDS-ON WEB APP TEST AUTOMATION

PRESENTED BY:

[Andrew Knight](#)

DESCRIPTION

When unit tests aren't enough, how can we write reliable automated tests for Web apps in live browsers? It's easy with Python! Let's build a test project from the ground up using `pytest` and `selenium` to test DuckDuckGo searches. We'll take a top-down approach and get our hands dirty with automation code at each layer. Learn everything from switching browsers to avoiding race conditions!

VIDEO



[Watch on YouTube](#)

Questions

