TA Talk

Automating Browser-based tests using Selenium WebDriver

CS 506 – Spring 2021

Adil Ahmed

Files and Slides

All scripts created during the demo have been pushed to GitHub:
https://github.com/adilahmed31/CS506SeleniumDemo

The slides are available on Canvas and on the remote repository.

Introduction to Selenium

- Selenium is a web automation testing tool.

Selenium WebDriver

Selenium IDE

Selenium Grid

Source: https://www.selenium.dev

Introduction to Selenium

- Selenium is a web automation testing tool.

Selenium WebDriver

Selenium IDE

Selenium Grid

Source: https://www.selenium.dev

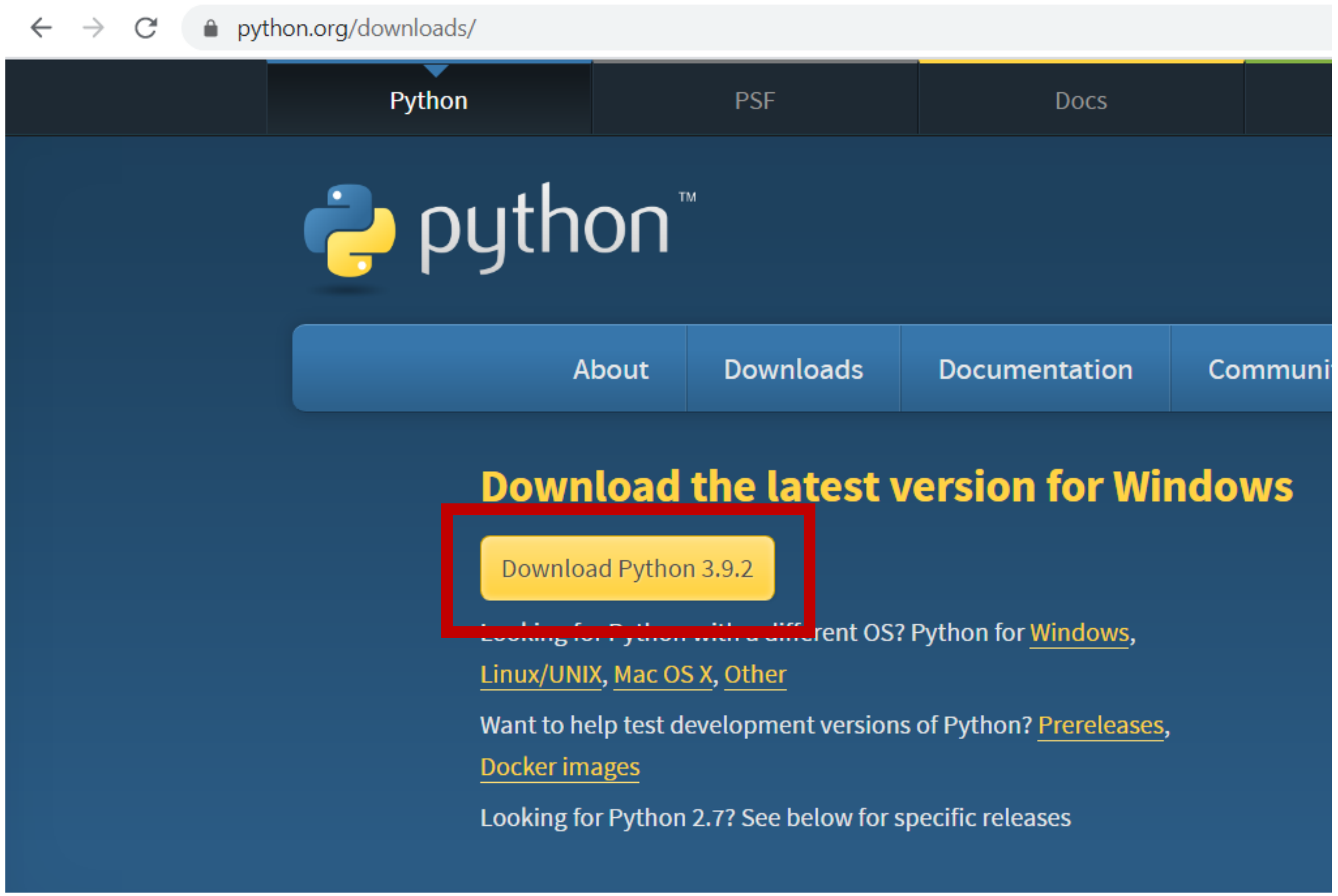
Download and Install

1. Download Python
2. Install Python
3. Install Selenium Python Libraries
4. Download and Install PyCharm IDE

Download and Install

1. Download Python
2. Install Python
3. Install Selenium Python Libraries
4. Download and Install PyCharm IDE

1. Visit https://www.python.org/downloads/
2. Download the installer



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

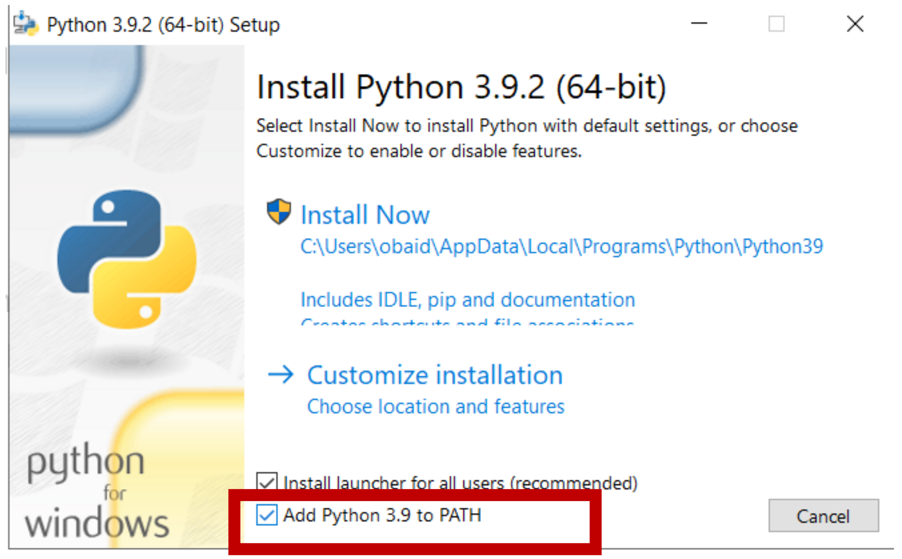
Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

Download and Install

1. Download Python
2. Install Python
3. Install Selenium Python Libraries
4. Download and Install PyCharm IDE

1. Run the downloaded executable.
2. Follow the install steps (make sure the highlighted option is checked)



Python 3.9.2 (64-bit) Setup

Install Python 3.9.2 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\obaid\AppData\Local\Programs\Python\Python39

Includes IDLE, pip and documentation

Customize installation
Choose location and features

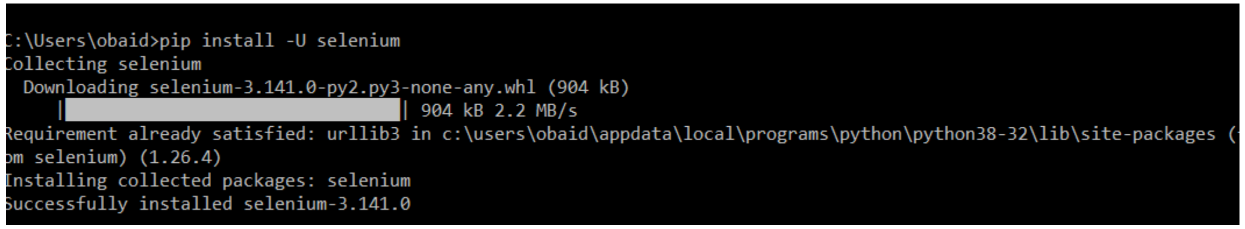
☑ Install launcher for all users (recommended)
☑ Add Python 3.9 to PATH

Cancel

Download and Install

1. Download Python
2. Install Python
3. Install Selenium Python Libraries
4. Download and Install PyCharm IDE

Run the command *pip install –U selenium*

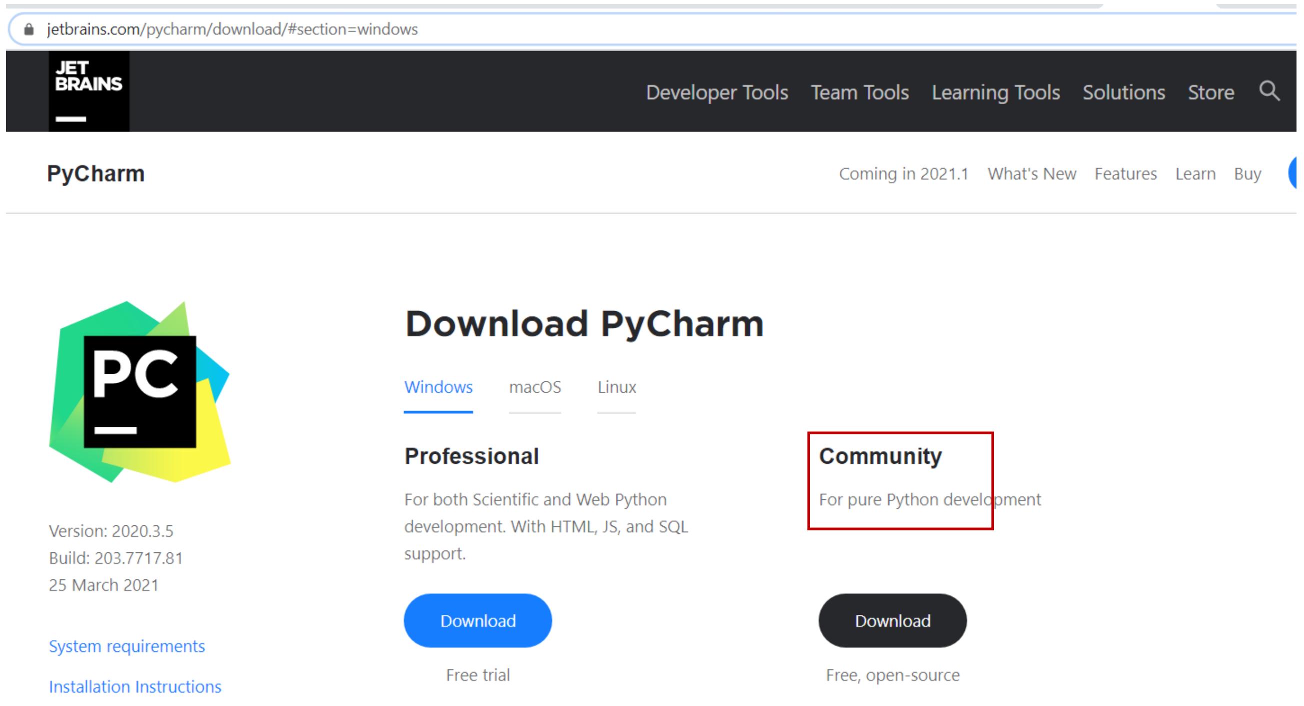


```
:\Users\obaid>pip install -U selenium
ollecting selenium
  Downloading selenium-3.141.0-py2.py3-none-any.whl (904 kB)
     |████████████████████████████████| 904 kB 2.2 MB/s
equirement already satisfied: urllib3 in c:\users\obaid\appdata\local\programs\python\python38-32\lib\site-packages (
m selenium) (1.26.4)
nstalling collected packages: selenium
uccessfully installed selenium-3.141.0
```

Download and Install

1. Download Python
2. Install Python
3. Install Selenium Python Libraries
4. Download and Install PyCharm IDE

1. Download the installer from: https://www.jetbrains.com/pycharm/download
2. Run the downloaded executable


jetbrains.com/pycharm/download/#section=windows

Download PyCharm

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

Free trial

Community

For pure Python development

Free, open-source

Version: 2020.3.5
Build: 203.7717.81
25 March 2021

Get Started

1. Create a new project in PyCharm
2. Add Selenium Scripts
3. Potential Error: “no module named selenium”
   1. Change python interpreter path in PyCharm to installed python location

   OR

   2. Simply re-run *pip install –U selenium* from within the PyCharm terminal
4. Install driver for browser (e.g. Chrome)
5. Add chromedriver path

Get Started

1. Create a new project in PyCharm
2. Add Selenium Scripts
3. Potential Error: “no module named selenium”
    1. Change python interpreter path in PyCharm to installed python location

    OR

    2. Simply re-run *pip install –U selenium* from within the PyCharm terminal
4. Install driver for browser (e.g. Chrome)
5. Add chromedriver to PATH

Get Started

1. Create a new project in PyCharm
2. Add Selenium Scripts
3. Potential Error: “no module named selenium”
    1. Change python interpreter path in PyCharm to installed python location

    OR

    2. Simply re-run *pip install –U selenium* from within the PyCharm terminal
4. Install driver for browser (e.g. Chrome)
5. Add chromedriver to PATH

Get Started

1. Create a new project in PyCharm
2. Add Selenium Scripts
3. Potential Error: “no module named selenium”
    1. Change python interpreter path in PyCharm to installed python location

    OR

    2. Simply re-run *pip install –U selenium* from within the PyCharm terminal
4. Install driver for browser (e.g. Chrome)
5. Add chromedriver to PATH

Get Started

1. Create a new project in PyCharm
2. Add Selenium Scripts
3. Potential Error: “no module named selenium”
    1. Change python interpreter path in PyCharm to installed python location

    OR

    2. Simply re-run *pip install –U selenium* from within the PyCharm terminal
4. Install driver for browser (e.g. Chrome)
5. Add chromedriver path

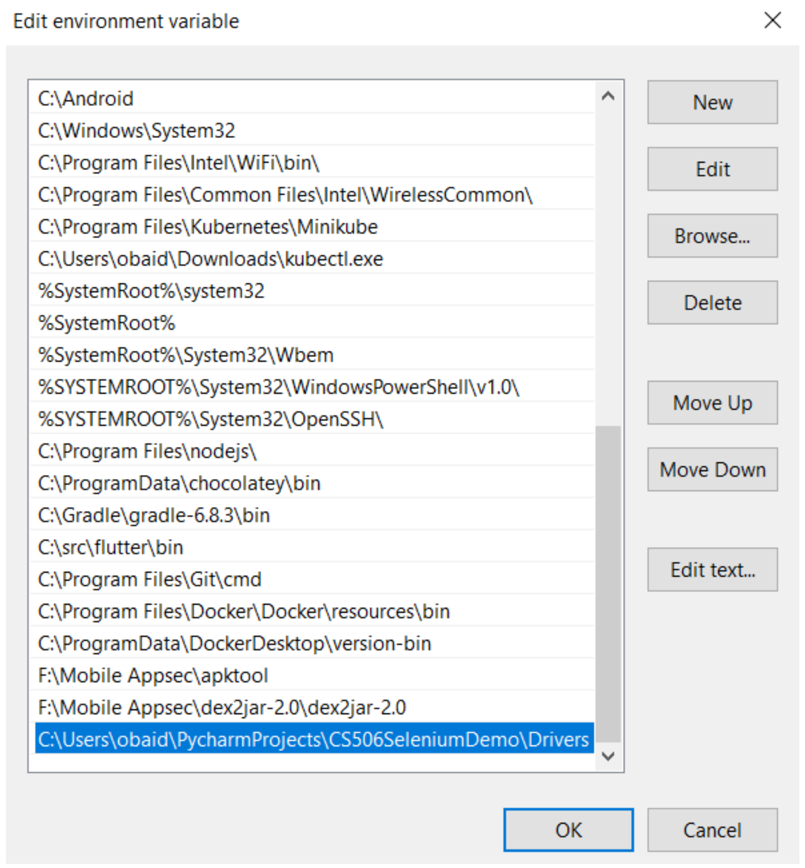
1. Check Chrome Version
2. Download corresponding driver from: https://sites.google.com/a/chromium.org/chromedriver/downloads
3. Extract exe from zip

Get Started

1. Create a new project in PyCharm
2. Add Selenium Scripts
3. Potential Error: “no module named selenium”
    1. Change python interpreter path in PyCharm to installed python location

    OR

    2. Simply re-run *pip install –U selenium* from within the PyCharm terminal
4. Install driver for browser (e.g. Chrome)
5. Add chromedriver to PATH



Edit environment variable

```
C:\Android
C:\Windows\System32
C:\Program Files\Intel\WiFi\bin\
C:\Program Files\Common Files\Intel\WirelessCommon\
C:\Program Files\Kubernetes\Minikube
C:\Users\obaid\Downloads\kubectl.exe
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\nodejs\
C:\ProgramData\chocolatey\bin
C:\Gradle\gradle-6.8.3\bin
C:\src\flutter\bin
C:\Program Files\Git\cmd
C:\Program Files\Docker\Docker\resources\bin
C:\ProgramData\DockerDesktop\version-bin
F:\Mobile Appsec\apktool
F:\Mobile Appsec\dex2jar-2.0\dex2jar-2.0
C:\Users\obaid\PycharmProjects\CS506SeleniumDemo\Drivers
```

Running Tests in Selenium

$$\begin{tikzcd}
\text{Automate} \arrow[r] & \text{Test} \arrow[r] & \text{Report}
\end{tikzcd}$$

Selenium WebDriver

unittest

html-testRunner

WebDriver API

1. Browser Manipulation
2. Locating Elements
3. Waits
4. Alerts
5. Keyboard

WebDriver

1. Browser Manipulation
2. Locating Elements
3. Waits
4. Alerts
5. Keyboard

Chosen Examples (Python)	
Navigate to	*driver.get("http://google.com")*
Back	*driver.back()*
Forward	*driver.forward()*
Refresh	*driver.refresh()*
Get Window Handle	*driver.current_window_handle*
Switch Window	*driver.switch_to.window(window_handle)*

Source: https://www.selenium.dev/documentation/en/webdriver/browser_manipulation/

WebDriver

1. Browser Manipulation
2. Locating Elements
3. Waits
4. Alerts
5. Keyboard

WebElement represents a DOM element.

You can get element by:

1. class name
2. css selector
3. id
4. Name
5. link text
6. partial link text
7. Tag name
8. xpath

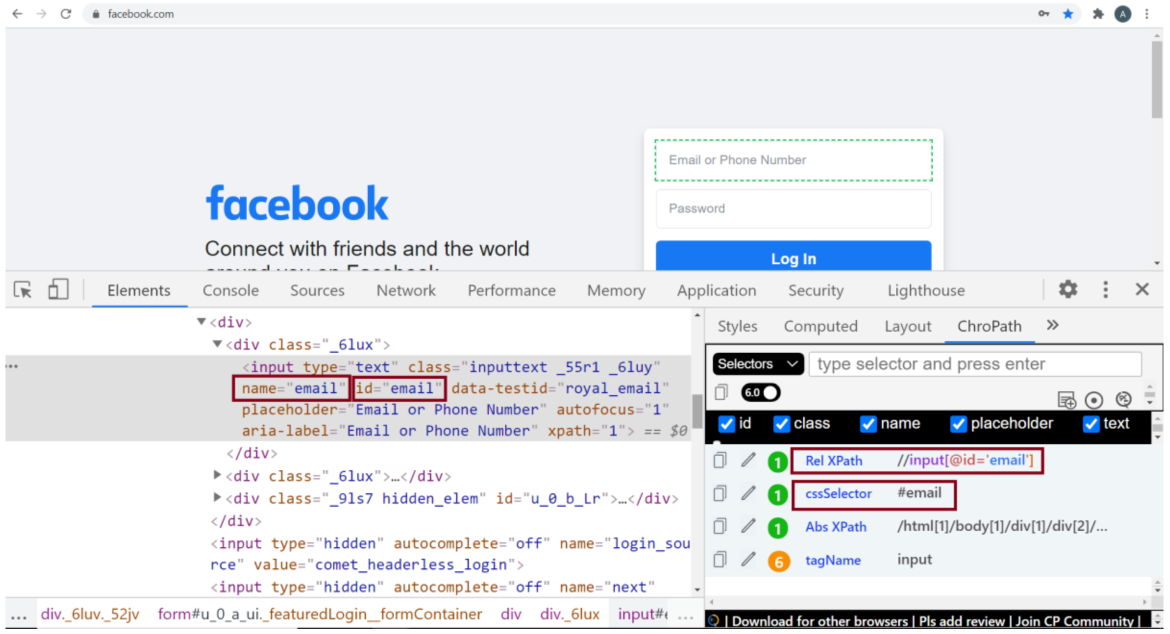
Tip: Use the Chropath Chrome extension for getting locators

Source: https://www.selenium.dev/documentation/en/webdriver/web_element/

WebDriver

1. Browser Manipulation
2. Locating Elements
3. Waits
4. Alerts
5. Keyboard

Example 1:



```
▼<div>
  ▼<div class="_6lux">
      <input type="text" class="inputtext _55r1 _6luy"
      name="email" id="email" data-testid="royal_email"
      placeholder="Email or Phone Number" autofocus="1"
      aria-label="Email or Phone Number" xpath="1"> == $0
    </div>
  ▶<div class="_6lux">…</div>
  ▶<div class="_9ls7 hidden_elem" id="u_0_b_Lr">…</div>
</div>
<input type="hidden" autocomplete="off" name="login_sou
rce" value="comet_headerless_login">
<input type="hidden" autocomplete="off" name="next"
```

… div._6luv._52jv form#u_0_a_ui._featuredLogin__formContainer div div._6lux input#e …

```
1  Rel XPath     //input[@id='email']
1  cssSelector   #email
1  Abs XPath     /html[1]/body[1]/div[1]/div[2]/...
6  tagName       input
```

The Email textbox can be accessed in any of the below ways:

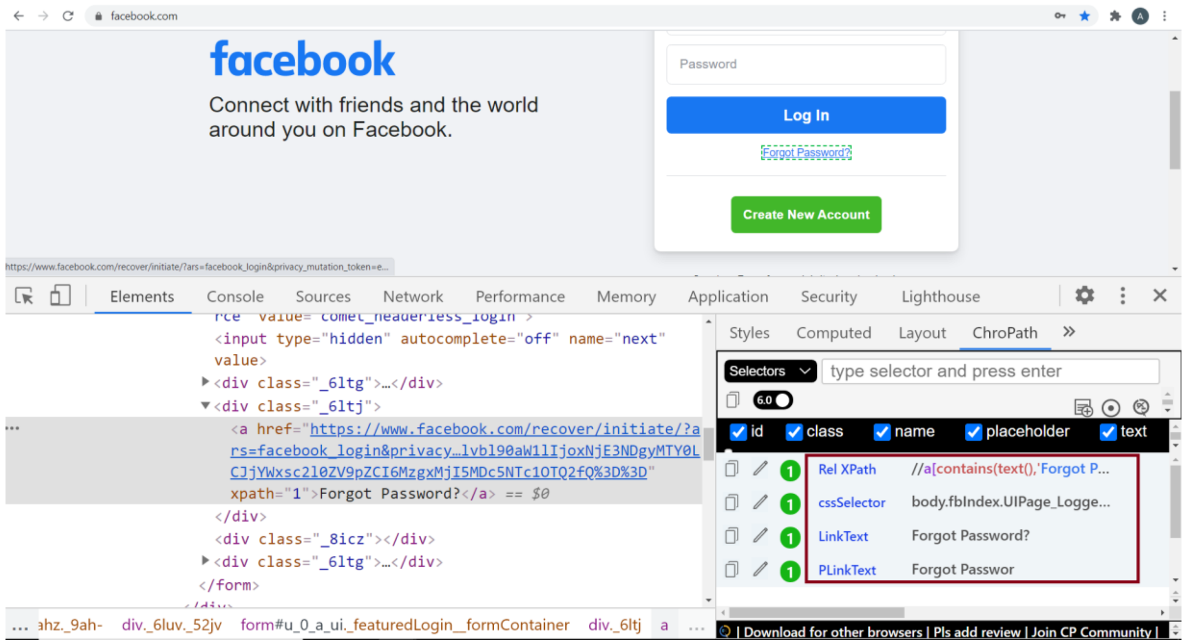
- *driver.find_element_by_name(“email”)*
- *driver.find_element_by_id(“email”)*
- *driver.find_element_by_xpath(“//input[@id=‘email’]”)*
- *driver.find_element_by_css_selector(“#email”)*

Source: https://www.selenium.dev/documentation/en/webdriver/web_element/

WebDriver

1. Browser Manipulation
2. Locating Elements
3. Waits
4. Alerts
5. Keyboard

Example 2:


facebook.com
facebook
Connect with friends and the world around you on Facebook.
Password
Log In
Forgot Password?
Create New Account
Elements Console Sources Network Performance Memory Application Security Lighthouse
Styles Computed Layout ChroPath
Selectors
type selector and press enter
id class name placeholder text
| Download for other browsers | Pls add review | Join CP Community |

```
<input type="hidden" autocomplete="off" name="next"
value>
▶<div class="_6ltg">…</div>
▼<div class="_6ltj">
  <a href="https://www.facebook.com/recover/initiate/?a
  rs=facebook_login&privacy…lvbl90aW1lIjoxNjE3NDgyMTY0L
  CJjYWxsc2l0ZV9pZCI6MzgxMjI5MDc5NTc1OTQ2fQ%3D%3D"
  xpath="1">Forgot Password?</a> == $0
</div>
<div class="_8icz"></div>
▶<div class="_6ltg">…</div>
</form>
```

```
… ahz._9ah-   div._6luv._52jv   form#u_0_a_ui._featuredLogin__formContainer   div._6ltj   a   …
```

Rel XPath	//a[contains(text(),'Forgot P...
cssSelector	body.fbIndex.UIPage_Logge...
LinkText	Forgot Password?
PLinkText	Forgot Passwor

The Forgot Password Link can be accessed in any of the below ways:

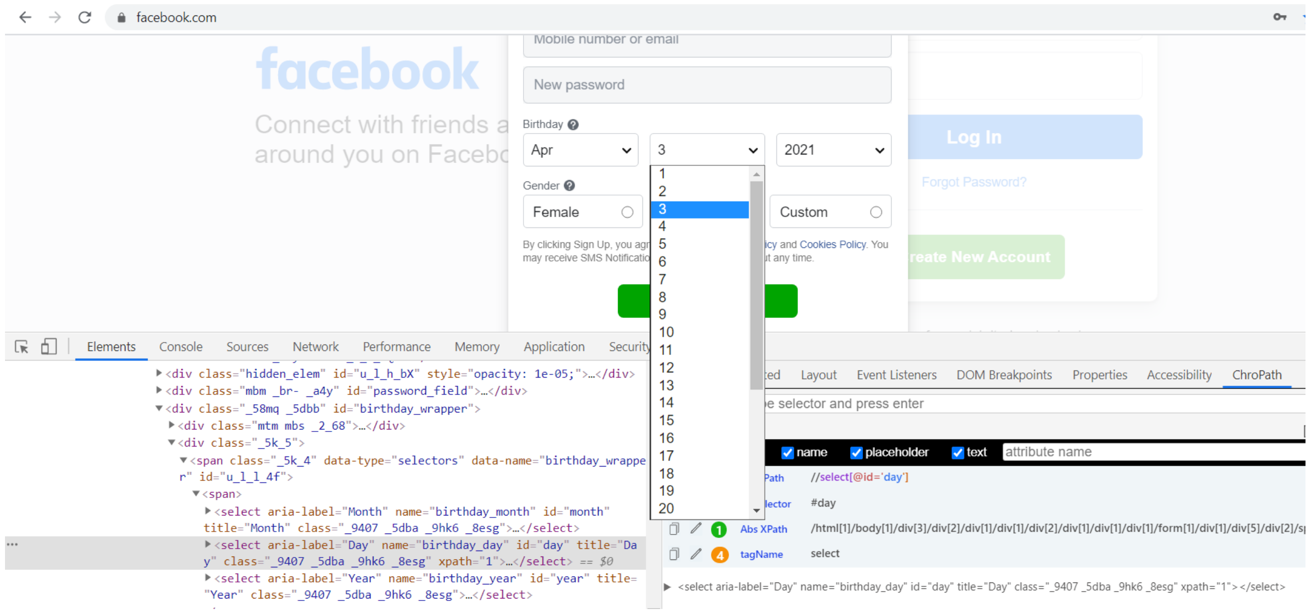
- *driver.find_element_by_xpath("//a[contains(text(),'Forgot Password?')]")*
- *driver.find_element_by_link_text("Forgot Password?")*
- *driver.find_element_by_partial_link_text("Forgot Passwor")*

Source: https://www.selenium.dev/documentation/en/webdriver/web_element/

WebDriver

1. Browser Manipulation
2. **Locating Elements**
3. Waits
4. Alerts
5. Keyboard

Example 3:



```
▸<div class="hidden_elem" id="u_l_h_bX" style="opacity: 1e-05;">…</div>
▸<div class="mbm _br- _a4y" id="password_field">…</div>
▾<div class="_58mq _5dbb" id="birthday_wrapper">
  ▸<div class="mtm mbs _2_68">…</div>
  ▾<div class="_5k_5">
    ▾<span class="_5k_4" data-type="selectors" data-name="birthday_wrappe
    r" id="u_l_l_4f">
      ▾<span>
        ▸<select aria-label="Month" name="birthday_month" id="month"
        title="Month" class="_9407 _5dba _9hk6 _8esg">…</select>
        ▸<select aria-label="Day" name="birthday_day" id="day" title="Da
        y" class="_9407 _5dba _9hk6 _8esg" xpath="1">…</select> == $0
        ▸<select aria-label="Year" name="birthday_year" id="year" title=
        "Year" class="_9407 _5dba _9hk6 _8esg">…</select>
```

```
Path        //select[@id='day']
lector      #day
1 Abs XPath /html[1]/body[1]/div[3]/div[2]/div[1]/div[1]/div[2]/div[1]/div[1]/div[1]/form[1]/div[1]/div[5]/div[2]/s
4 tagName   select
▸ <select aria-label="Day" name="birthday_day" id="day" title="Day" class="_9407 _5dba _9hk6 _8esg" xpath="1"></select>
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

The date can be set by selecting an element within the dropdown element.

dateDropDown= Select(driver.find_element_by_name(“birthday_day"))
date=dateDropDown.select_by_valye(“3")

Source: https://www.selenium.dev/documentation/en/webdriver/web_element/

Tip: Use the Chropath Chrome extension for getting locators

WebDriver

1. Browser Manipulation
2. Locating Elements
3. Waits
4. Alerts
5. Keyboard

Chosen Examples (Python)	
Find CSS Element	*attr = driver.find_element(By.CSS_SELECTOR, "h1").tag_name*
Find Element	*elements = driver.find_elements(By.TAG_NAME, 'p')*
Get Element Rect	*res = driver.find_element(By.CSS_SELECTOR, "h1").rect*
Get Element Text	*elements = driver.find_elements(By.TAG_NAME, 'p’)*
Get Window Handle	*driver.current_window_handle*

Source: https://www.selenium.dev/documentation/en/webdriver/web_element/

WebDriver

1. Browser Manipulation
2. Locating Elements
3. Waits
4. Alerts
5. Keyboard

Explicit Waits:

Chosen Examples (Python)
WebDriverWait(driver).until(document_initialised)
el = WebDriverWait(driver).until(lambda d: d.find_element_by_tag_name("p"))
WebDriverWait(driver, timeout=3).until(some_condition)

Source: https://www.selenium.dev/documentation/en/webdriver/browser_manipulation/

WebDriver

1. Browser Manipulation
2. Locating Elements
3. Waits
4. Alerts
5. Keyboard

Implicit Waits:

Chosen Example (Python)
Driver.implicitly_wait(10)

Source: https://www.selenium.dev/documentation/en/webdriver/waits/

WebDriver

1. Browser Manipulation
2. Locating Elements
3. Waits
4. Alerts
5. Keyboard

```
# Click the link to activate the alert
driver.find_element(By.LINK_TEXT, "See an example alert").click()

# Wait for the alert to be displayed and store it in a variable
alert = wait.until(expected_conditions.alert_is_present())

# Store the alert text in a variable
text = alert.text

# Press the OK button
alert.accept()
```

Source: https://www.selenium.dev/documentation/en/webdriver/js_alerts_prompts_and_confirmations/

WebDriver

1. Browser Manipulation
2. Locating Elements
3. Waits
4. Alerts
5. Keyboard

Chosen Examples (Python)	
sendKeys	*Driver.find_element(By.Name,”q”).send_keys(“webdriver”+ Keys.ENTER)*
keyDown	*webdriver.ActionChains(driver).key_down(Keys.CONTROL).send_keys("a").perform()*

Clear	*SearchInput = driver.find_element(By.NAME, "q")*
	SearchInput.send_keys("selenium")
	SearchInput.clear()

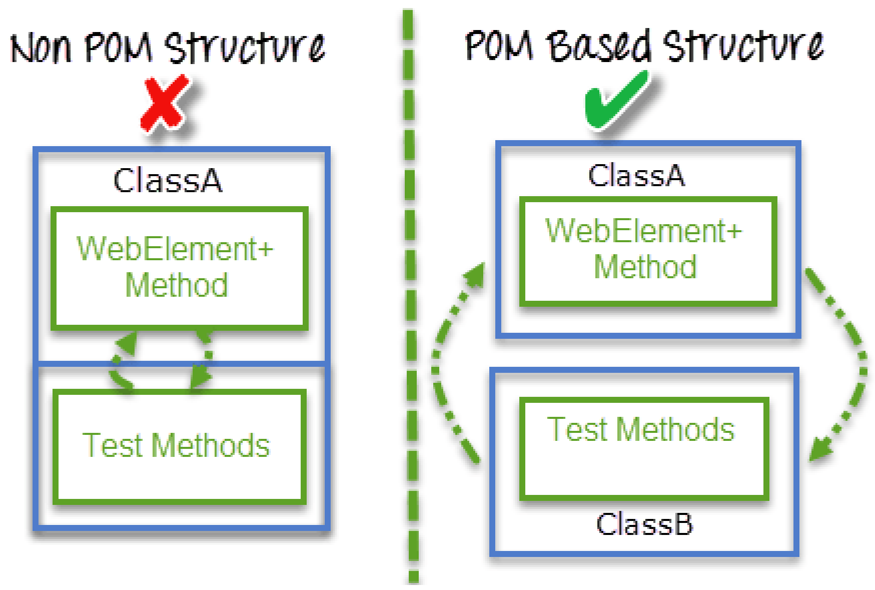
Source: https://www.selenium.dev/documentation/en/webdriver/keyboard/

Writing Tests and Generating Reports

- We use the unittest library in python
- Install the html-TestRunner package: *pip install html-testRunner*
- Add the following lines to your code to run from the command line

```
If __name__ == '__main__':
        unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(output='path'))
```

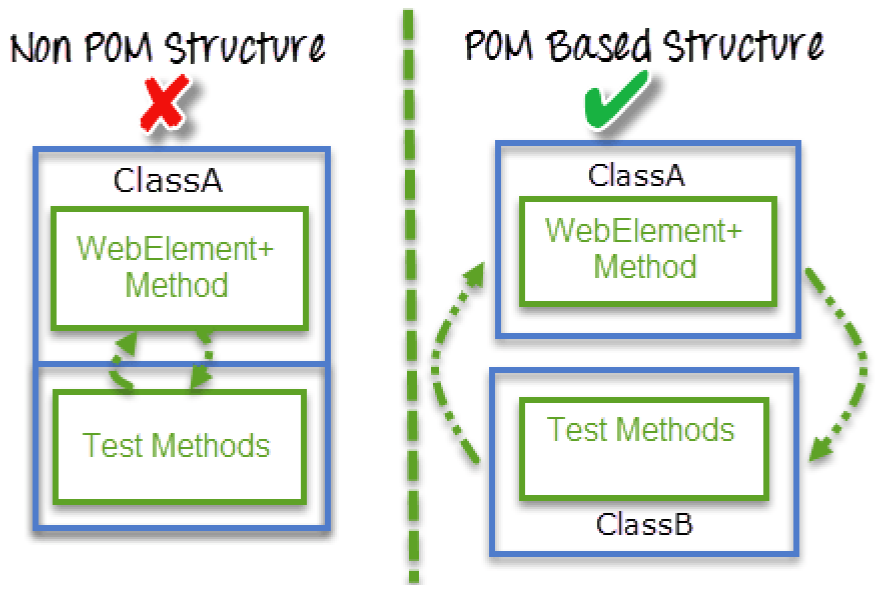
Page Object Model



- A page object is an object-oriented class that serves as an interface to a page of your AUT (Application Under Test).
- The tests then use the methods of this page object class whenever they need to interact with the UI of that page.
- There is a clean separation between test code and page specific code such as locators.
- There is a single repository for the services or operations offered by the page rather than having these services scattered throughout the tests.

Source: https://www.selenium.dev/documentation/en/guidelines_and_recommendations/page_object_models/

Page Object Model



- For each web page -> there is a corresponding page class.
- Page class will contain WebElements of the page and page methods to perform operations on those WebElements.

Source: https://www.selenium.dev/documentation/en/guidelines_and_recommendations/page_object_models/

Page Object Model

1. Create a separate class for each application page.

1. Define the elements as objects and the actions as methods in the class.

2. Call the methods from the test script.

Source: https://www.selenium.dev/documentation/en/guidelines_and_recommendations/page_object_models/

Selenium IDE

- Selenium IDE is a browser extension for recording and playing back tests.

Selenium WebDriver

Selenium IDE

Selenium Grid

Source: https://www.selenium.dev

Record a test case

1. Add the Selenium IDE Chrome Extension
2. Launch the extension
3. Click Record
4. Perform test cases
5. Stop Recording
6. Right Click on the test and export it

Source: https://www.selenium.dev/documentation/en/guidelines_and_recommendations/page_object_models/

Practice Exercise

Note: This is an unofficial exercise for practice and doesn’t carry any real points. The solutions are available on the git repo.

Automate the given two actions on the site: http://automationpractice.com/index.php . Use the Page Object Model design pattern. Create separate classes for Locators, Pages and Tests.

1. Search for a keyword and open the link for a particular result. Write a test to check if the product you were looking for was returned.
E.g. Search for the keyword “chiffon” and access the product page for the “Printed Chiffon Dress”

2. Sign-up as a new user and verify that the sign-up was successful.

Thank You!

For feedback and questions ->

1. Canvas Discussion Boards
2. Slack Channel

This is a tiny tip of the actual tip of the Selenium iceberg!

Have a use-case for Selenium that wasn’t covered in this brief demo? Send me a note and we can figure it out!

Examples:
1. Mouse Events
2. Robot Actions
3. Event listeners
4. Executing custom JavaScript during a test case
5. Proxying and logging Selenium traffic to run further tests and debugging

Email: oahmed4@wisc.edu