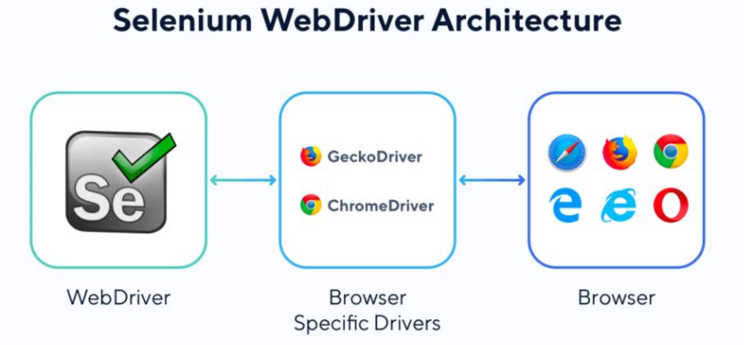
# Selenium Webdriver

Open source automation tool used for automating **web-based** application testing.

# How to install selenium

- PYTHON - [Download Python | Python.org]
- SELENIUM - in cmd type - pip install selenium
- WEBDRIVER - https://chromedriver.chromium.org/downloads
- PYCHARM - https://www.jetbrains.com/pycharm/download/#section=windows



# Navigation

driver.get(‘URL’)

# Locators

Find_element_by_ (id, name, link_text, class_name, css selector, xpath)

# Actions

.send_keys() = to enter input value in a blank
.click()          = to give click command
.clear()          = to clear the input field
.text              = to copy the text

# CSS Selector (Locator)

| FROM | SYNTAX |
|---|---|
| Class, Attribute & Value | tagname.classvalue[attribute = ‘value’] |
| Attribute & Value | tagname[attribute = ‘value’] |
| ID | tagname#IDvalue |
| Class | tagname.classvalue |

Note: tagname is optional

To check the uniqueness from console, syntax => **$(“tagname[attribute=’value’]”)**

# XPATH (Locator)

| FROM | SYNTAX |
|---|---|
| Attribute & Value | //tagname[@attribute = ‘value’] |
| text | //tagname[text() = ‘type text here’] |
| Parents to child | //tagname[@attribute = ‘value’]/tagname |
| Parents to last child | //tagname[@attribute = ‘value’]/tagname[last()] |
| Grand parent to child | //tagname[@attribute = ‘value’]/tagname/tagname |
| Child to any ancestor | //tagname[@attribute = ‘value’]/ancestor::tagname[@attribute = ‘value’] |
| Starts with | //tagname[starts-with(@attribute,’starting values’)] |
| contains | //*[contains(@attribute,’value’)] |
| Starts with and contains | //tagname[starts-with(@attribute,'starting values') and contains(@attribute,’value')] |

To check the uniqueness from console, syntax => **$x(“//tagname[@attribute=’value’]”)**

# Multiple Checkboxes

Create one xpath which will be common for all checkboxes then use for loop to click all

# Static Dropdown (select tagname)

Import Select object, give locator inside it, keep this in one variable and use Select functions

# Dynamic Dropdown

Create xpath which will be common for all suggestions, then use for loop to select the particular

# Pop up Alert

```
popup = driver.switch_to_alert()
```

# File Upload

Locate element with tagname “input” then in the send_keys, give the complete path of the file.

# Implicit Wait

driver.implicitly_wait()

# Explicit Wait

first import **By**, **expected_conditions** and **WebDriverWait** then use explicit wait with conditions

# Mouse Hover, Double-click, Right-click

Import **ActionChains**

# Multiple Windows, Tabs

```
driver.switch_to_window(driver.window_handles[x])
```

# Frames

driver.switch_to_frame('x'), x can be id value, class value, name value or driver.find_element things also.

# Get attribute

```
driver.find_element_by_xpath(" ").get_attribute("attribute name")
```

# Pytest Framework

pip install pytest, Python file name & function name should start with test_

# Fixtures & Conftest.py

Conftest.py to store the fixture & html report modifications

# Pytest Html Report

pip install pytest-html (to download the package) , –html=report.html (to download the report)

# Logs (Code on next slide)

Log is defined as records in programming.

```
logger = logging.getLogger()
filehandler = logging.FileHandler("logfile.log")
formatter = logging.Formatter('%(asctime)s: %(levelname)s: %(module)s: %(funcName)s: %(message)s',
datefmt='%d/%m/%Y %I:%M:%S %p')
filehandler.setFormatter(formatter)
logger.addHandler(filehandler)
logger.setLevel(logging.DEBUG)
logger.debug("Debug message")
logger.info("Information regarding the test case")
logger.warning("Test case pass but with a Warning message")
logger.error("Test case fail")
logger.critical("Important test case fail on which other test case depends")
```

# Scrolling

driver.execute_script("window.scrollTo(0, X)") => X is the vertical height measured in pixels

driver.execute_script("window.scrollTo(0, document.body.scrollHeight);") => scroll to the last of the page

driver.execute_script("arguments[0].scrollIntoView();", X) => scroll to specific element X

```
y = 1000
    for step in range(0,50):
         driver.execute_script("window.scrollTo(0, "+str(y)+")")
         y += 1000
         time.sleep(1)
```

# Pytest Parametrize

@pytest.mark.parametrize('count' ,[1,2,3])

# Network Calls

pip install selenium-wire - website

# CSS Property

```
rgb = driver.find_element_by_xpath('xpath of element').value_of_css_property('background-color')
#to get background colour of an element
```