Selenium Webdriver

Com Python

Selenium Projects

Selenium Webdriver

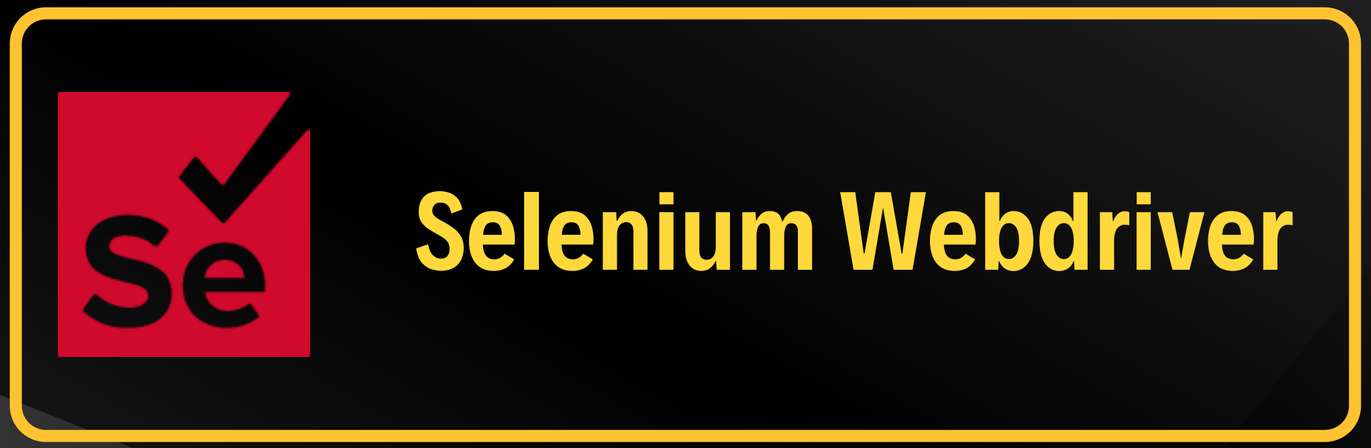
Manipula navegadores Web nativamente, suportando várias linguagens de programação.

Selenium IDE

Extensão do Chrome, Firefox e Edge. gravar e reproduzir interações com o navegador.

Selenium Grid

Usa o Selenium Webdriver para rodar testes em várias máquinas ao mesmo tempo.



É uma Biblioteca / Módulo para interagir com navegadores

Também pode ser considerado uma API

Usado para automatizar testes de GUI

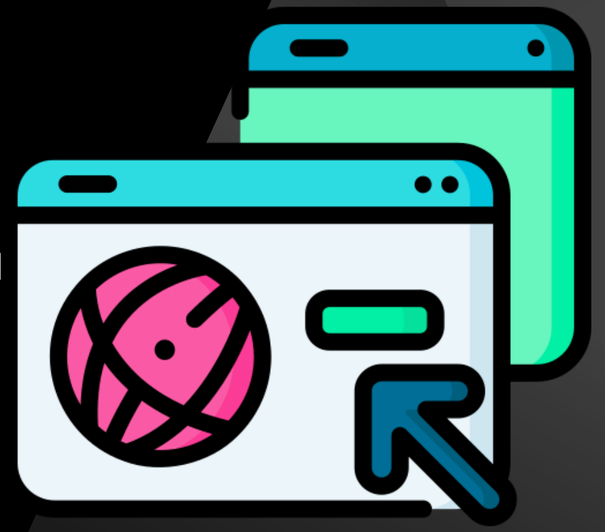
Graphic User Interface

Como funciona...

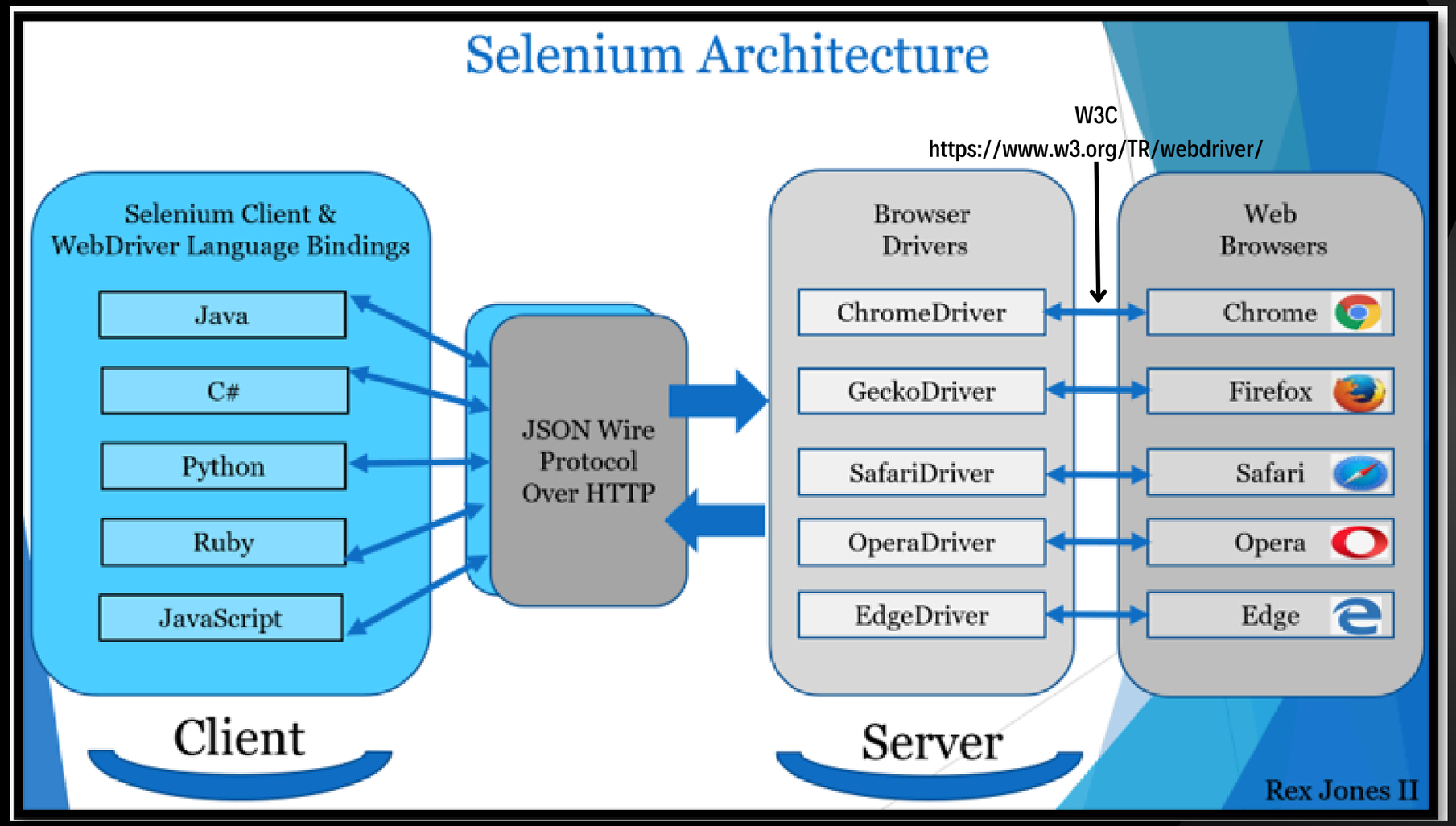
AUTOMATION CODE

WEBDRIVER

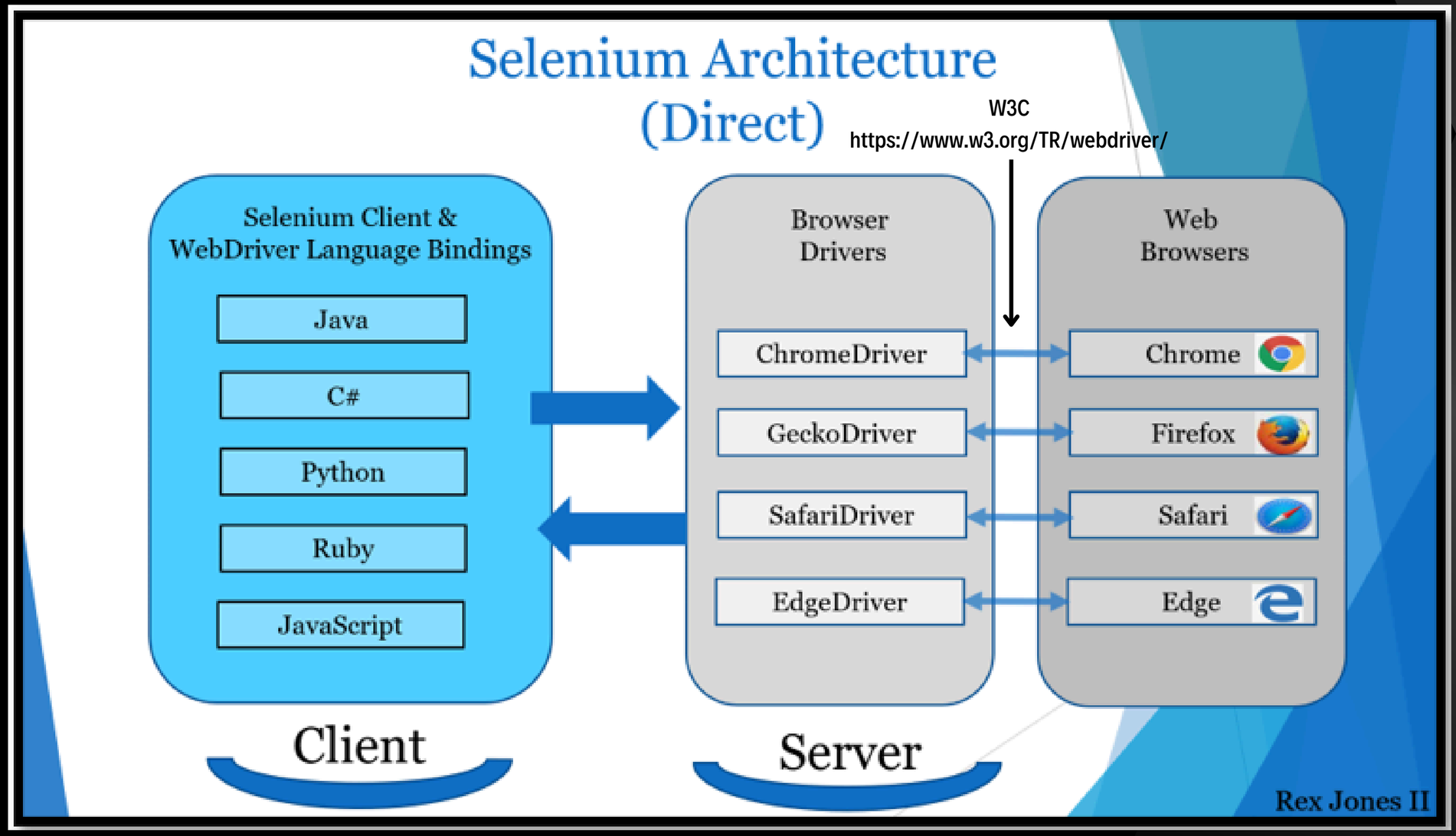
BROWSER



Selenium 3


Selenium Architecture
W3C
https://www.w3.org/TR/webdriver/
Selenium Client &
WebDriver Language Bindings
Java
C#
Python
Ruby
JavaScript
JSON Wire
Protocol
Over HTTP
Browser
Drivers
ChromeDriver
GeckoDriver
SafariDriver
OperaDriver
EdgeDriver
Web
Browsers
Chrome
Firefox
Safari
Opera
Edge
Client
Server
Rex Jones II

Selenium 4


Selenium Architecture
(Direct)
W3C
https://www.w3.org/TR/webdriver/
Selenium Client &
WebDriver Language Bindings
Java
C#
Python
Ruby
JavaScript
Browser
Drivers
ChromeDriver
GeckoDriver
SafariDriver
EdgeDriver
Web
Browsers
Chrome
Firefox
Safari
Edge
Client
Server
Rex Jones II

https://www.selenium.dev/

Baixando e instalando o Python e PyCharm

https://www.python.org

https://www.jetbrains.com/

Baixando e instalando o VS Code

https://code.visualstudio.com/download

Baixando os Drivers dos navegadores

e colocar em uma pasta chamada "webdrivers" ou "drivers" no C:

https://www.selenium.dev/

Criando Ambientes Vituais no Python

Usando PyCharm e VSCode

Configurando o Selenium Webdriver

e criando seu primeiro script

https://www.selenium.dev/

Mapeando Elementos da tela

Primeiros Passos

Mapeando Elementos da tela

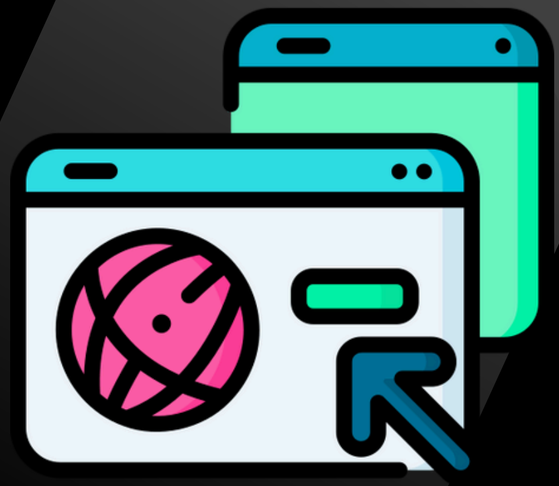
Tipos de Locators

Locators

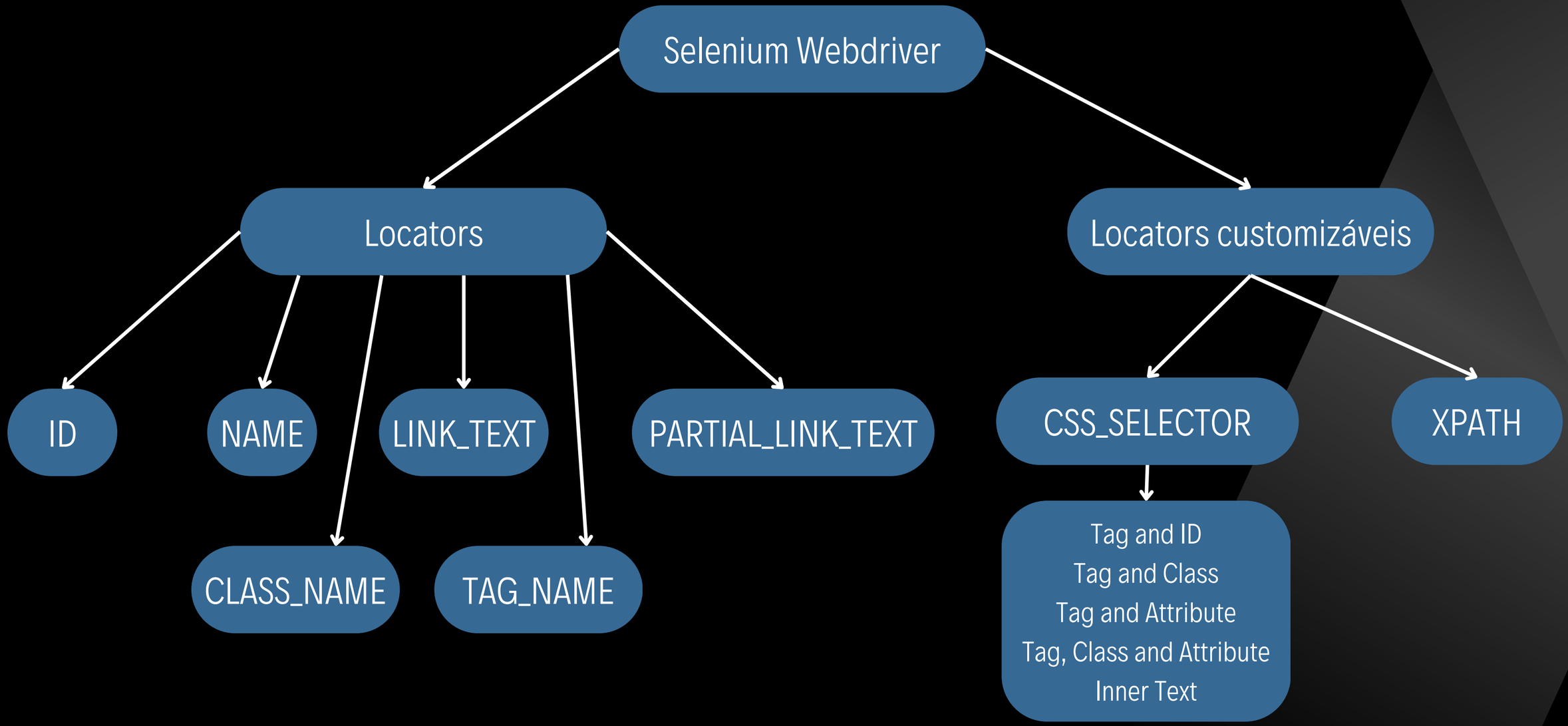
Tag Atributo Valor Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password"
data-test="password" id="password" name="password" autocorrect="off"
autocapitalize="none" value fdprocessedid="84hy06">
```

Estrutura de um elemento



Tipos de Locators



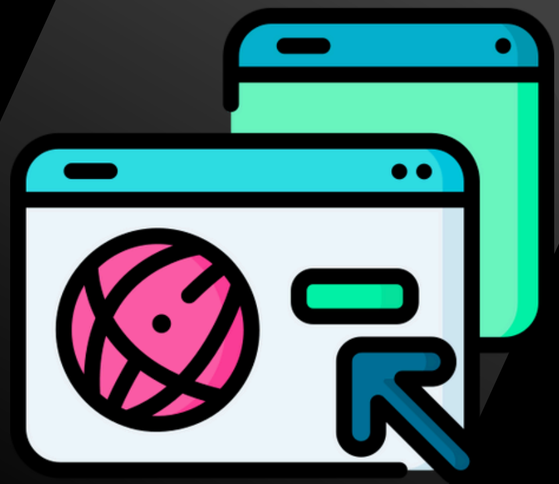
Locators

Tag Atributo Valor Atributo Valor

```
<input class="input_error form input" placeholder="Password" type="password"
data-test="password" id="password" name="password" autocorrect="off"
autocapitalize="none" value fdprocessedid="84hy06">
```

By.ID

```
browser.find_element(By.ID, "password")
```



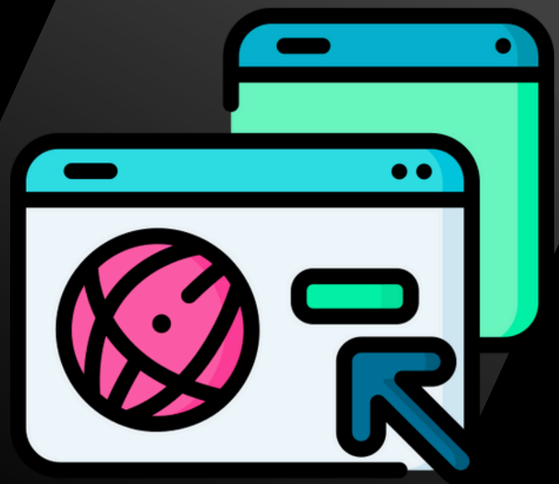
Locators

Tag Atributo Valor Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password"
data-test="password" id="password" name="password" autocorrect="off"
autocapitalize="none" value fdprocessedid="84hy06">
```

By.NAME

```
browser.find_element(By.NAME, "password")
```



Locators

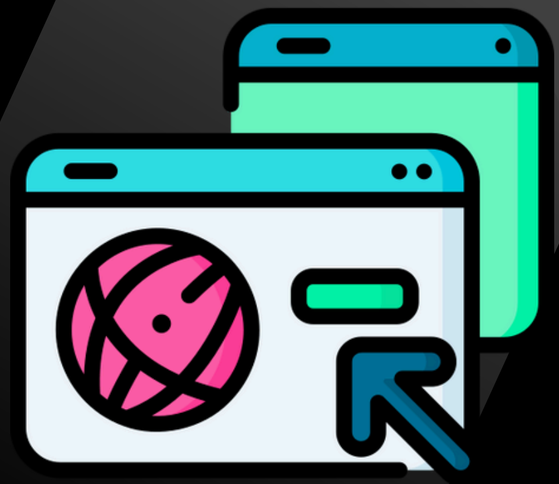
href: atributo de link

Link text

```
<a href="https://twitter.com/saucelabs" target="_blank" rel="noreferrer">Twitter
</a>
```

By.LINK_TEXT

```
browser.find_element(By.LINK_TEXT, "Twitter")
```



Locators

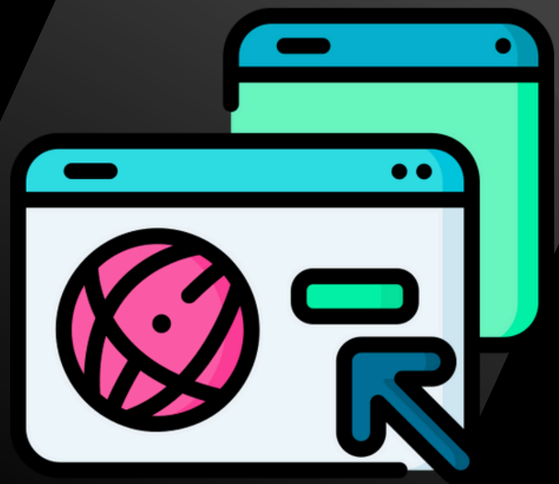
href: atributo de link

Link text

```
<a href="https://twitter.com/saucelabs" target="_blank" rel="noreferrer">Twitter
</a>
```

By.PARTIAL_LINK_TEXT

```
browser.find_element(By.PARTIAL_LINK_TEXT, "Twi")
```



Locators

Tag Atributo Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password"
data-test="password" id="password" name="password" autocorrect="off"
autocapitalize="none" value fdprocessedid="84hy06">
```

By.CLASS_NAME

Encontra a primeira ocorrência

```
browser.find_element(By.CLASS_NAME, "input_error form_input")
browser.find_elements(By.CLASS_NAME, "input_error form_input")
```

Retorna todas as ocorrências

Locators

Tag Atributo Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password"
data-test="password" id="password" name="password" autocorrect="off"
autocapitalize="none" value fdprocessedid="84hy06">
```

By.TAG_NAME

Encontra a primeira ocorrência

```
browser.find_element(By.TAG_NAME, "input")
browser.find_elements(By.TAG_NAME, "input")
```

Retorna todas as ocorrências

Locators

Tag Atributo Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password"
data-test="password" id="password" name="password" autocorrect="off"
autocapitalize="none" value fdprocessedid="84hy06">
```

By.CSS_SELECTOR

tag_name#id_value (tag é opcional)

```
browser.find_element(By.CSS_SELECTOR, "input#password")
browser.find_element(By.CSS_SELECTOR, "#password")
```

Locators

Tag Atributo Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password"
data-test="password" id="password" name="password" autocorrect="off"
autocapitalize="none" value fdprocessedid="84hy06">
```

By.CSS_SELECTOR

tag_name.class_value (tag é opcional)

```
browser.find_element(By.CSS_SELECTOR, "input.input_error")
browser.find_element(By.CSS_SELECTOR, ".input_error")
```

Locators

Tag Atributo Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password"
data-test="password" id="password" name="password" autocorrect="off"
autocapitalize="none" value fdprocessedid="84hy06">
```

By.CSS_SELECTOR

tag_name[attribute=value] (tag é opcional)

```
browser.find_element(By.CSS_SELECTOR, "input[type=password]")
browser.find_element(By.CSS_SELECTOR, "[type=password]")
```

Locators

Tag Atributo Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password"
data-test="password" id="password" name="password" autocorrect="off"
autocapitalize="none" value fdprocessedid="84hy06">
```

By.CSS_SELECTOR

tag_name.class_name[attribute=value]

browser.find_element(By.CSS_SELECTOR, "input.input_error[type=password]")

XPath

- XPath é uma sintaxe para definir partes de um documento XML
- XPath pode ser usado para navegar por elementos e atributos em um documento XML
- XPath usa expressões de caminho para navegar em documentos XML
- XPath contém uma biblioteca de funções padrão
- XPath é usado para localizar elementos em uma página web por meio do DOM
- XPath é o endereço do elemento em uma página

https://www.w3schools.com/xml/xpath_intro.asp

XPath

Absolute/Full XPath

Relative/Partial XPath

COMANDOS

Comandos de navegação
Comandos da aplicação
Comandos de localizar elementos
Comandos condicionais
Comandos de interação com elementos
Comandos de espera

COMANDOS DE NAVEGAÇÃO

browser.

get()
maximize_window()
refresh()
back()
forward()
close()
quit()

COMANDOS DA APLICAÇÃO

browser.

title

current_url

page_source

COMANDOS PARA LOCALIZAR ELEMENTOS

browser.

find_element()

find_elements()

COMANDOS CONDICIONAIS

element.
is_displayed()
is_enabled()
is_selected()

Retorna True ou False

COMANDOS DE INTERAÇÃO COM ELEMENTOS

element.

click()

send_keys()

text

get_attribute()

COMANDOS DE ESPERA (WAIT)

element.

time.sleep()

implicitly_wait

explicitly_wait

Pytest

O que é o PyTest?

PyTest é um framework de teste que permite escrever códigos de teste usando a linguagem Python. Ele te ajuda a escrever casos de teste simples e escaláveis para bancos de dados, APIs ou interface do usuário.

Algumas das vantagens do pytest são:

- Muito fácil de começar devido à sua sintaxe simples e fácil.
- Pode executar testes em paralelo.
- Pode executar um teste específico ou um subconjunto de testes
- Detectar testes automaticamente
- Ignorar testes

pytest

Pytest

Para instalar: pip install pytest

O pytest identifica quais são os casos de teste automaticamente desde que o nome dos métodos de teste estejam no seguinte formato:
test_*.py ou *_test.py

Documentação: https://docs.pytest.org/en/7.1.x/getting-started.html

pytest

Page Object

A ideia do PageObject é separar os elementos em arquivos diferentes baseados nas páginas em que eles aparecem, assim, escrevemos todos os elementos e métodos específicos daquela página em seu arquivo que é uma classe e usamos diretamente nos scripts de teste.

```
# Fazendo login (sem page object)
    driver.find_element(By.ID, "user-name").send_keys("standard_user")
    driver.find_element(By.ID, "password").send_keys("secret_sauce")
    driver.find_element(By.ID, "login-button").click()

# Fazendo login (com page object)
login_page.fazer_login("standard_user", "secret_sauce")
```