# Selenium

Behzad Khalaji

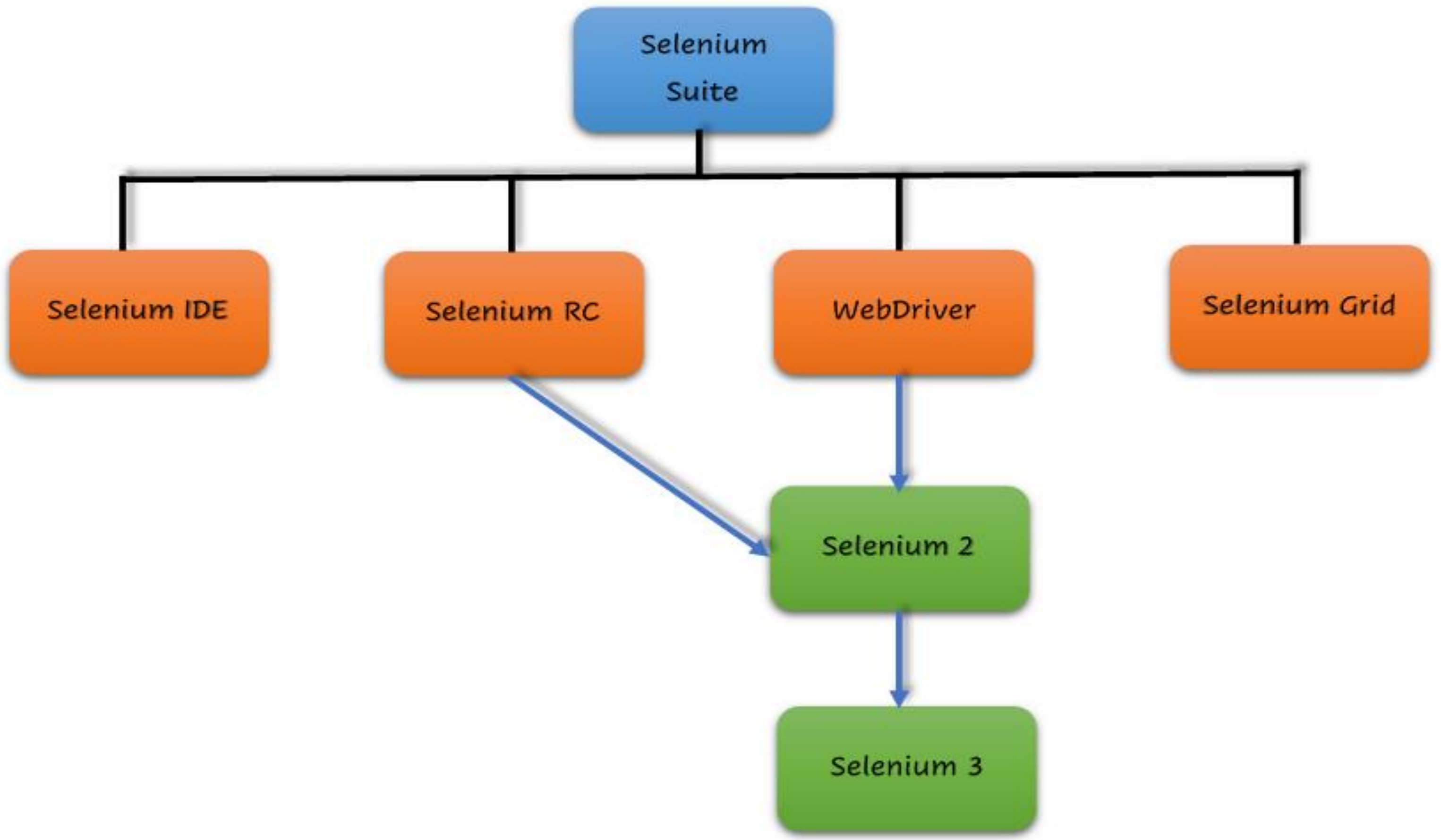
Profs: Dr. Hassan Haghighi

## Introduction

- **Selenium** is an open-source project for a range of tools and libraries aimed at supporting web browser automation
- Selenium automates web browsers. It is most famous for enabling rapid, repeatable web-app testing, which allows developers to ship new releases faster and with confidence.
- You can use multiple programming languages like Java, C#, Python etc to create Selenium Test Scripts.
- Selenium runs on Windows, Linux, and macOS.

# Selenium Component

- ✓ Selenium Integrated Development Environment (IDE)
- ✓ Selenium Remote Control (RC)
- ✓ WebDriver
- ✓ Selenium Grid



## History

- Selenium was originally developed by **Jason Huggins** in 2004 as an internal tool at ThoughtWorks.
- He was working on a web application that required frequent testing. Having realized that the repetitious Manual Testing of their application was becoming more and more inefficient, he created a JavaScript program that would automatically control the browser’s actions. He named this program as the “JavaScriptTestRunner.”
- Seeing potential in this idea to help automate other web applications, he made JavaScriptRunner open-source which was later re-named as Selenium Core.

# Timeline of Selenium

**2004**

**Paul Hammant**, decided to create a server that will act as an HTTP proxy to "trick" the browser into believing that Selenium Core and the web application being tested come from the same domain. This system became known as the Selenium Remote Control

**2006**

**Shinya Kasatani** of Japan created **Selenium IDE**, a Firefox extension that can automate the browser through a record-and-playback feature. He came up with this idea to further increase the speed in creating test cases. He donated Selenium IDE to the Selenium Project in **2006**.

**2007**

**Simon Stewart** created **WebDriver** circa **2006** when browsers and web applications were becoming more powerful and more restrictive with JavaScript programs like Selenium Core. It was the first cross-platform testing framework that could control the browser from the OS level.

**2007**

Selenium Grid was developed by **Patrick Lightbody** to address the need of minimizing test execution times as much as possible. He initially called the system "**Hosted QA.**" It was capable of capturing browser screenshots during significant stages, and also of sending out Selenium commands to different machines simultaneously.

## Selenium Programmers

**Paul Hammant**

Selenium Remote Control

**Shinya Kasatani**

Selenium IDE

**Simon Stewart**

WebDriver

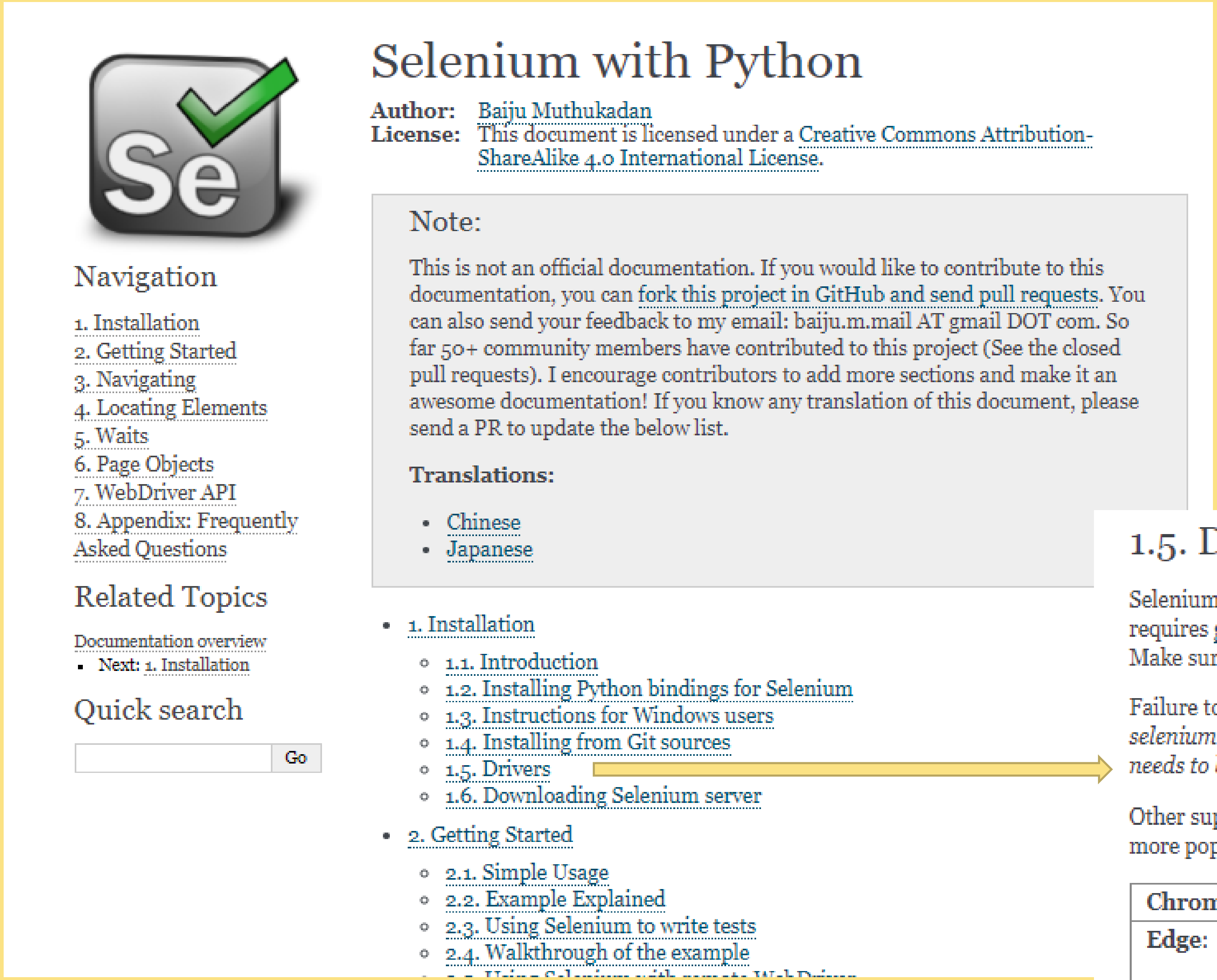
**Patrick Lightbody**

Selenium Grid

## Installation

- ➢ Selenium Python bindings provides a simple API to write functional/acceptance tests using Selenium WebDriver. Through Selenium Python API you can access all functionalities of Selenium WebDriver in an intuitive way.
- ❖ Pip install selenium
- ❖ Installing a desired browser like Chrome, Firefox,...
- ❖ Downloading a compatible Driver ⇒ Move Driver somewhere that Python and Selenium will be able to find it

https://selenium-python.readthedocs.io/



# Selenium with Python

**Author:** Baiju Muthukadan
**License:** This document is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

**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

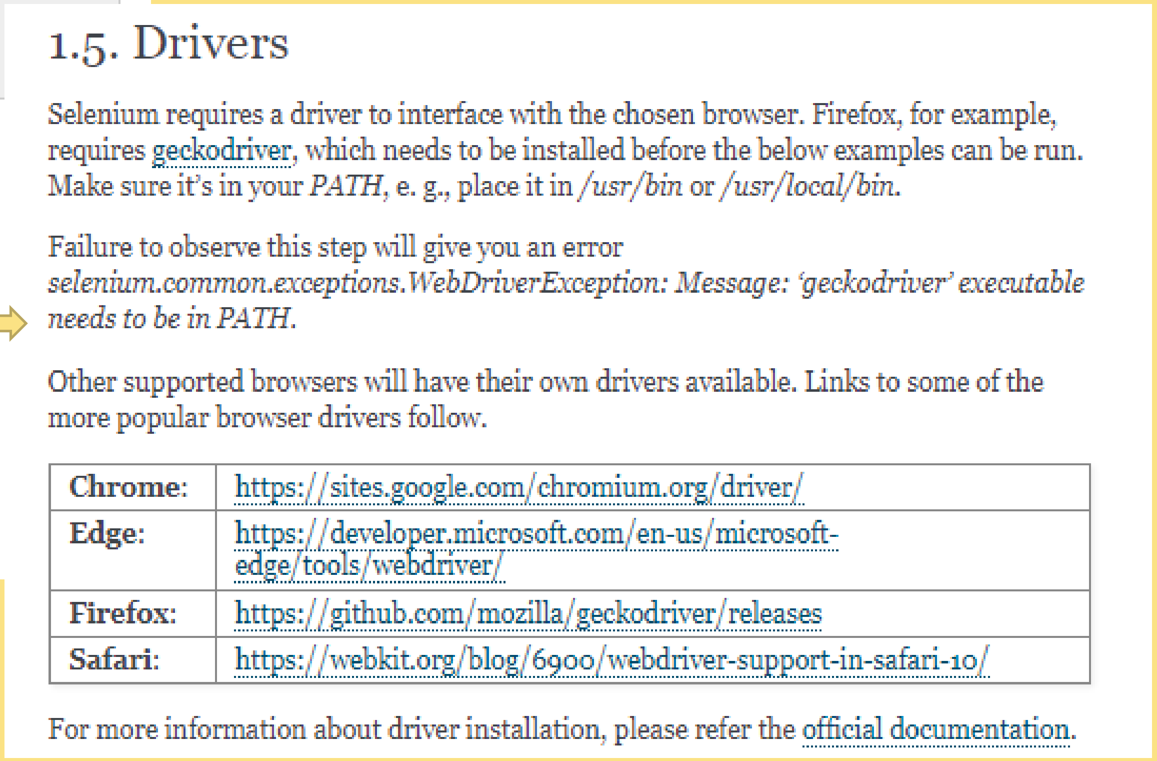
## Navigation

## Related Topics

Documentation overview
- Next: 1. Installation

## Quick search

Go



## 1.5. Drivers

Selenium requires a driver to interface with the chosen browser. Firefox, for example, requires geckodriver, which needs to be installed before the below examples can be run. Make sure it's in your *PATH*, e. g., place it in */usr/bin* or */usr/local/bin*.

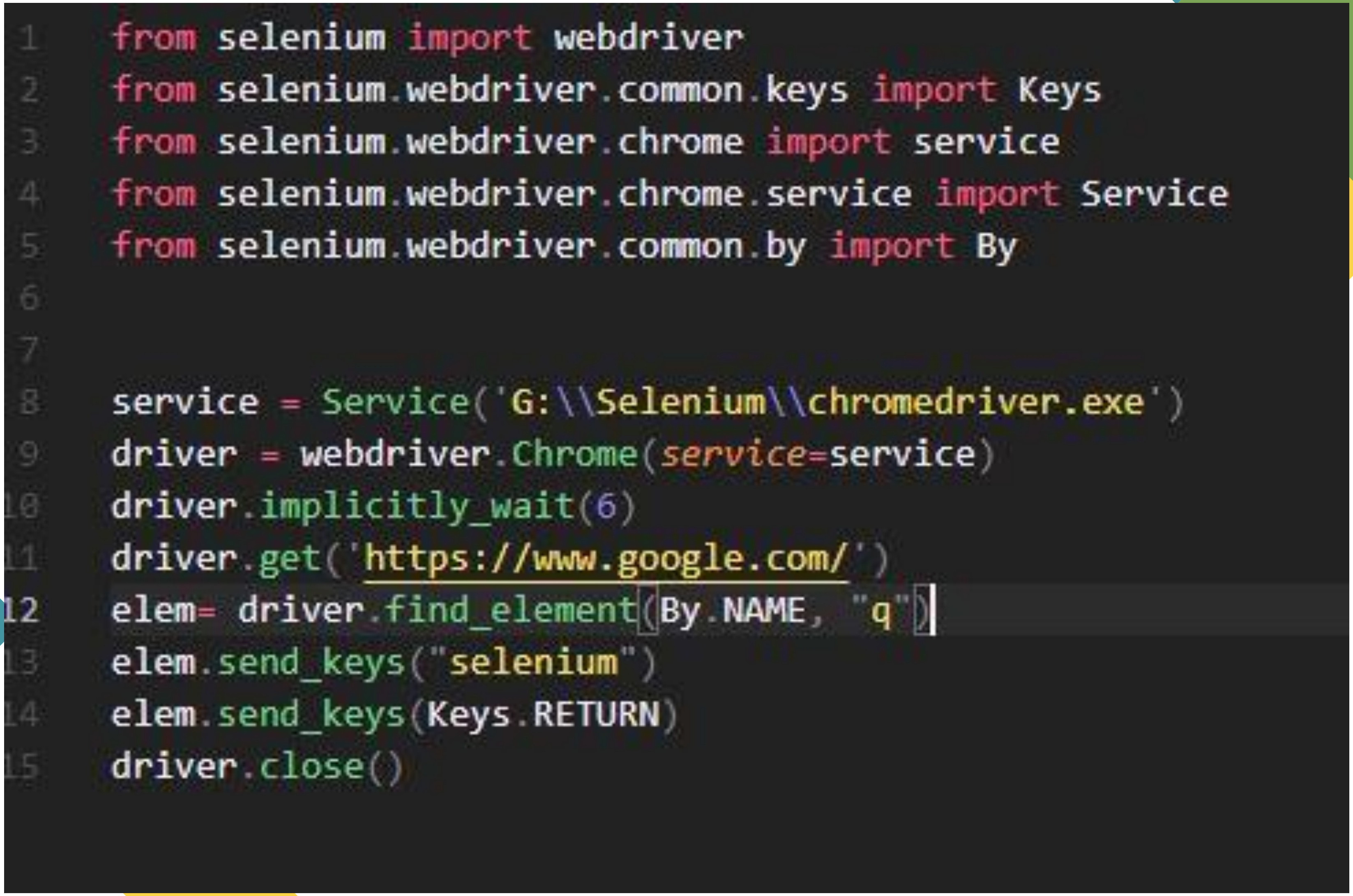
Failure to observe this step will give you an error *selenium.common.exceptions.WebDriverException: Message: 'geckodriver' executable needs to be in PATH.*

Other supported browsers will have their own drivers available. Links to some of the more popular browser drivers follow.

| | |
|---|---|
| **Chrome:** | https://sites.google.com/chromium.org/driver/ |
| **Edge:** | https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/ |
| **Firefox:** | https://github.com/mozilla/geckodriver/releases |
| **Safari:** | https://webkit.org/blog/6900/webdriver-support-in-safari-10/ |

For more information about driver installation, please refer the official documentation.

## Getting Started



```python
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome import service
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By


service = Service('G:\\Selenium\\chromedriver.exe')
driver = webdriver.Chrome(service=service)
driver.implicitly_wait(6)
driver.get('https://www.google.com/')
elem= driver.find_element(By.NAME, "q")
elem.send_keys("selenium")
elem.send_keys(Keys.RETURN)
driver.close()
```

## Find element by Id

```
<!DOCTYPE html>
<html>
<body>
<h1 id="myHeader">My Header</h1>
</body>
</html>
```

*myHeader = driver.find_element_by_id('myHeader')*

*myHeader = driver.find_element(By.ID, 'myHeader')*

## Find element by Name

```
<form action="/action_page.php" method="get">
  Choose your favorite subject:
  <button name="subject" type="submit" value="HTML">HTML</button>
</form>
```

*subjectButton = driver.find_element_by_name('subject')*

*subjectButton = driver.find_element(By.NAME, 'subject')*

## Find element by Tag

```
<html>
<body>
<p>Hello World!</p>
</body>
</html>
```

*p = driver.find_element_tag_name('p')*

*p = driver.find_element(By.TAG_NAME, 'p')*

## Find element by Link

### Find element by Link text

*a = driver.find_element_link_text(‘Cancel’)*

*a = driver.find_element(By.LINK_TEXT, ‘Cancel’)*

### Find element by  partial Link text

*a = driver.find_element_link_text(‘Can’)*

*a = driver.find_element(By.LINK_TEXT, ‘Can’)*

```
<html>
 <body>
 <p>Are you sure you want to do this?</p>
<a href="cancel.html">Cancel</a>
</body>
</html>
```

## Find element by Class

```
<!DOCTYPE html>
<html>
<body>
<h1 class="note">Heading</h1>
<p>This is some text.</p>
</body>
</html>
```

*h1 = driver.find_element_class_name(‘note’)*

*h1 = driver.find_element(By.CLASS_NAME, ‘note’)*

## Find element by CSS selector

```
<!DOCTYPE html>
<html>
<body>
<h1 class="note">Heading</h1>
<p>This is some text.</p>
</body>
</html>
```

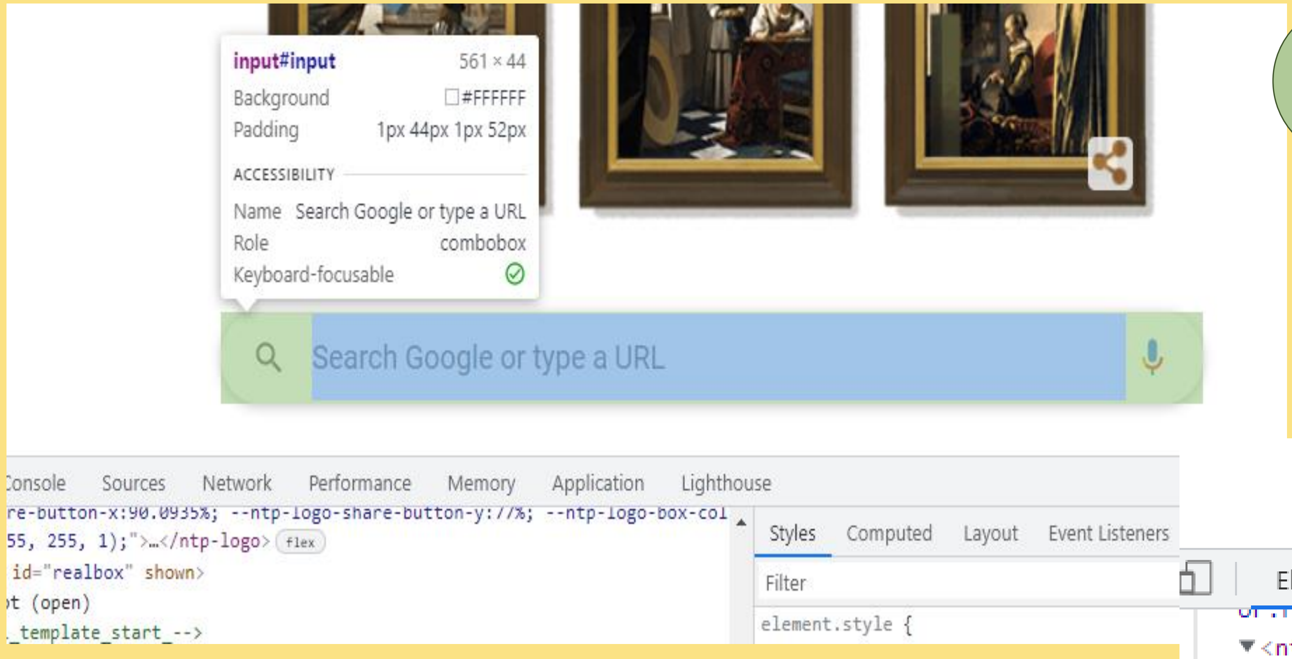
*h1 = driver.find_element_class_name(‘h1.note’)*

*h1 = driver.find_element(By.CLASS_NAME, ‘h1.note’)*

# Find element by XPath

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
</form>
</body>
</html>
```

```
username = driver.find_element_by_xpath("//form[input/@name='username']")
username = driver.find_element(By.XPATH, "//form[input/@name='username']")
```

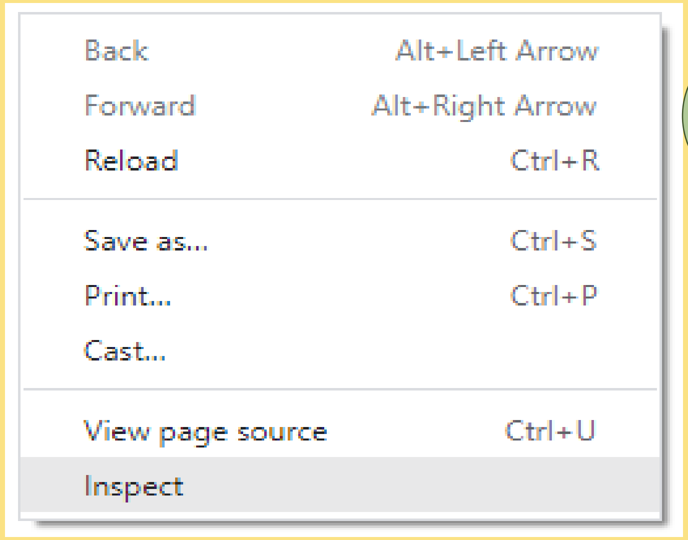

input#input 561 × 44
Background #FFFFFF
Padding 1px 44px 1px 52px
ACCESSIBILITY
Name Search Google or type a URL
Role combobox
Keyboard-focusable ✓
Search Google or type a URL
Console Sources Network Performance Memory Application Lighthouse
Styles Computed Layout Event Listeners
Filter

```
re-button-x:90.0935%; --ntp-logo-share-button-y:77%; --ntp-logo-box-col
55, 255, 1);">…</ntp-logo> flex
 id="realbox" shown>
ot (open)
_template_start_-->
```

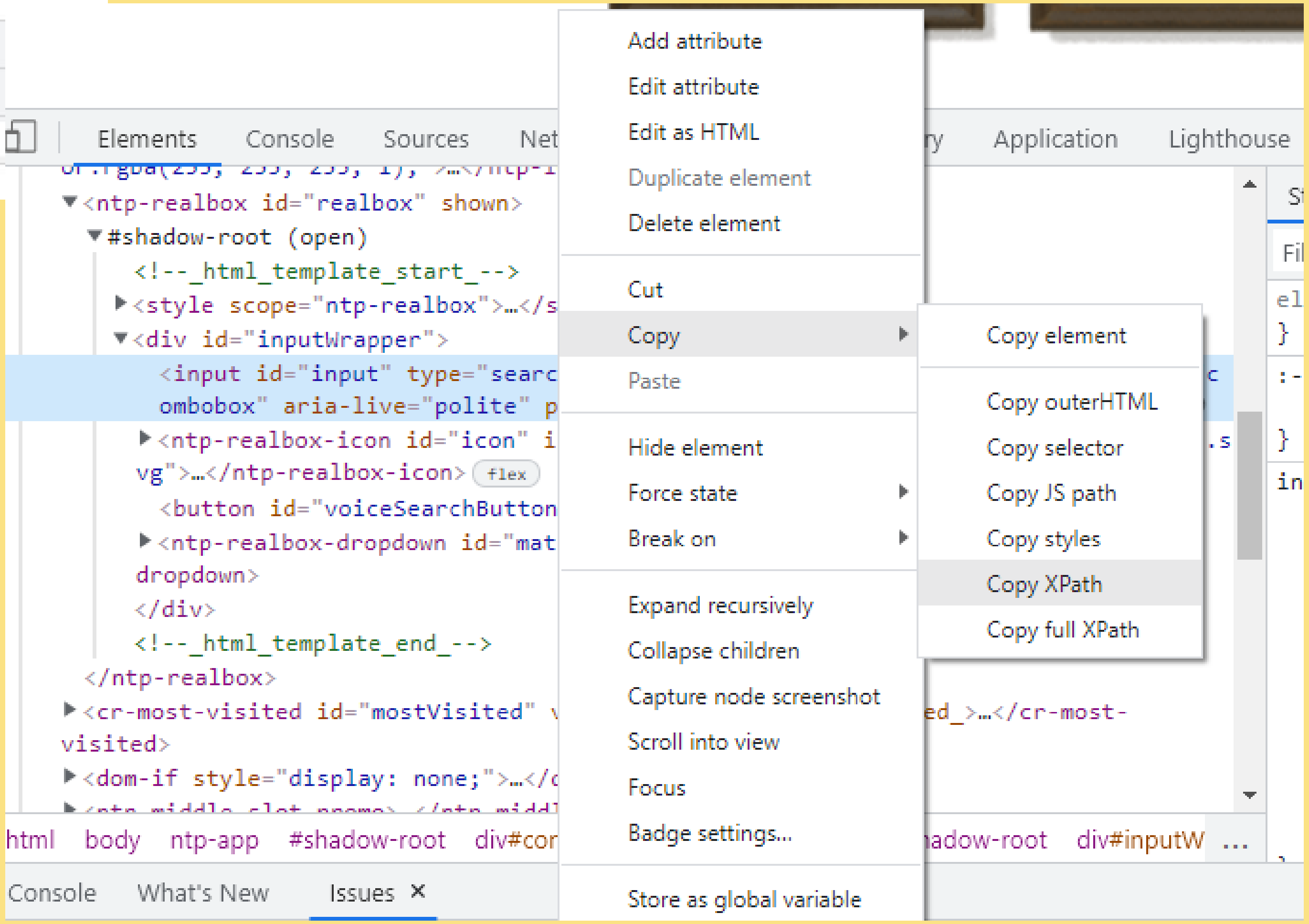
```
element.style {
```

2

3

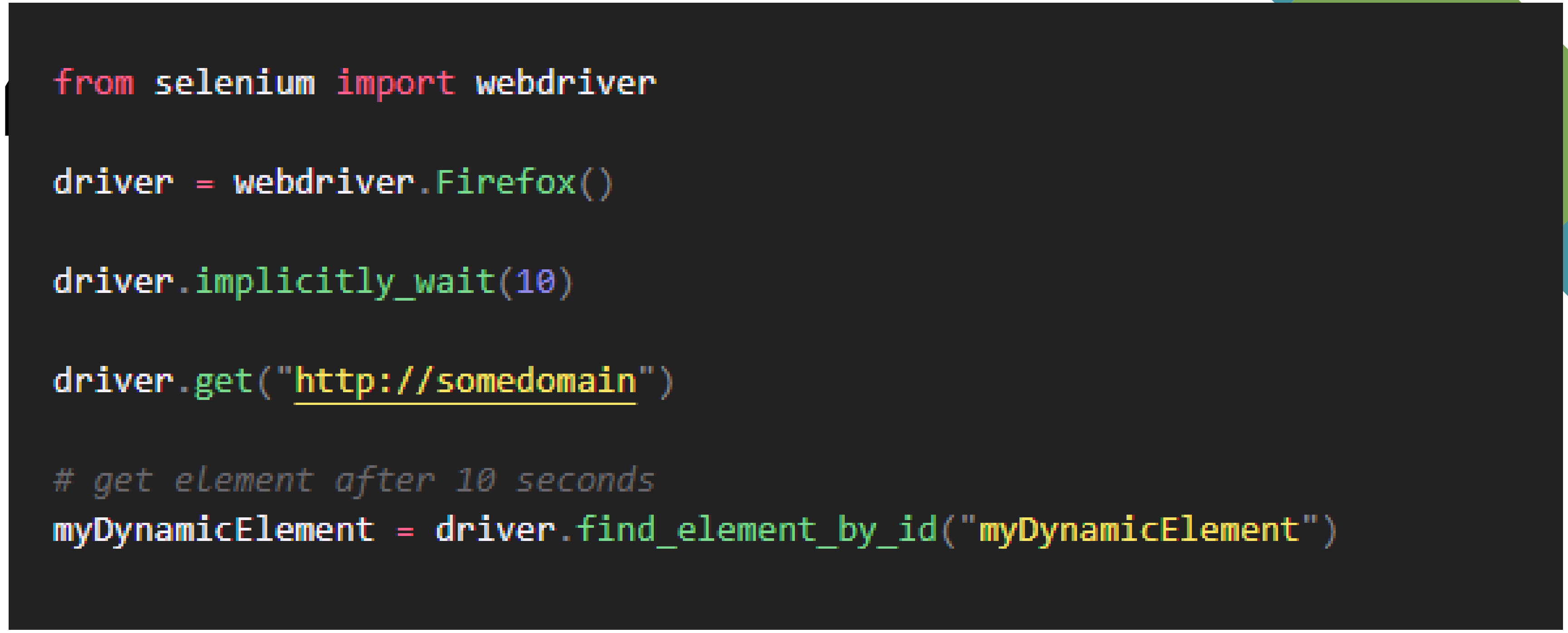


1



```
▼<ntp-realbox id="realbox" shown>
  ▼#shadow-root (open)
     <!--_html_template_start_-->
    ▶<style scope="ntp-realbox">…</s
    ▼<div id="inputWrapper">
       <input id="input" type="searc
       ombobox" aria-live="polite" p
      ▶<ntp-realbox-icon id="icon" i
      vg">…</ntp-realbox-icon> flex
       <button id="voiceSearchButton
      ▶<ntp-realbox-dropdown id="mat
      dropdown>
      </div>
      <!--_html_template_end_-->
   </ntp-realbox>
 ▶<cr-most-visited id="mostVisited"      ed_>…</cr-most-
 visited>
 ▶<dom-if style="display: none;">…</
```

## Implicit Wait



```
from selenium import webdriver

driver = webdriver.Firefox()

driver.implicitly_wait(10)

driver.get("http://somedomain")

# get element after 10 seconds
myDynamicElement = driver.find_element_by_id("myDynamicElement")
```

## Other Features

“

```
p = driver.find_elements(By.TAG_NAME, ‘p’)
text = p.text

html = driver.find_element_by_tag_name(‘img')
langValue = html.get_attribute(‘src')
```

## What is WCAG?

- Web Content Accessibility Guidelines, or WCAG, is considered to be the benchmark for website accessibility.
- Created by the World Wide Web Consortium (W3C),
- following WCAG guidelines is the best and the easiest way of making your website usable for all of your customers.
- Following this standard allows you to be certain that your website and online booking page are accessible to the widest possible audience.

https://rules.sonarsource.com/html/RSPEC-5254

# Thank you

Contact:

Email : behzad.khalaji14@gmail.com

Telegram : @behzadkhlj