BROWSER AUTOMATION USING PYTHON-SELENIUM

SANDEEP SURYAPRASAD

COURSE INDEX

USEFUL LINKS

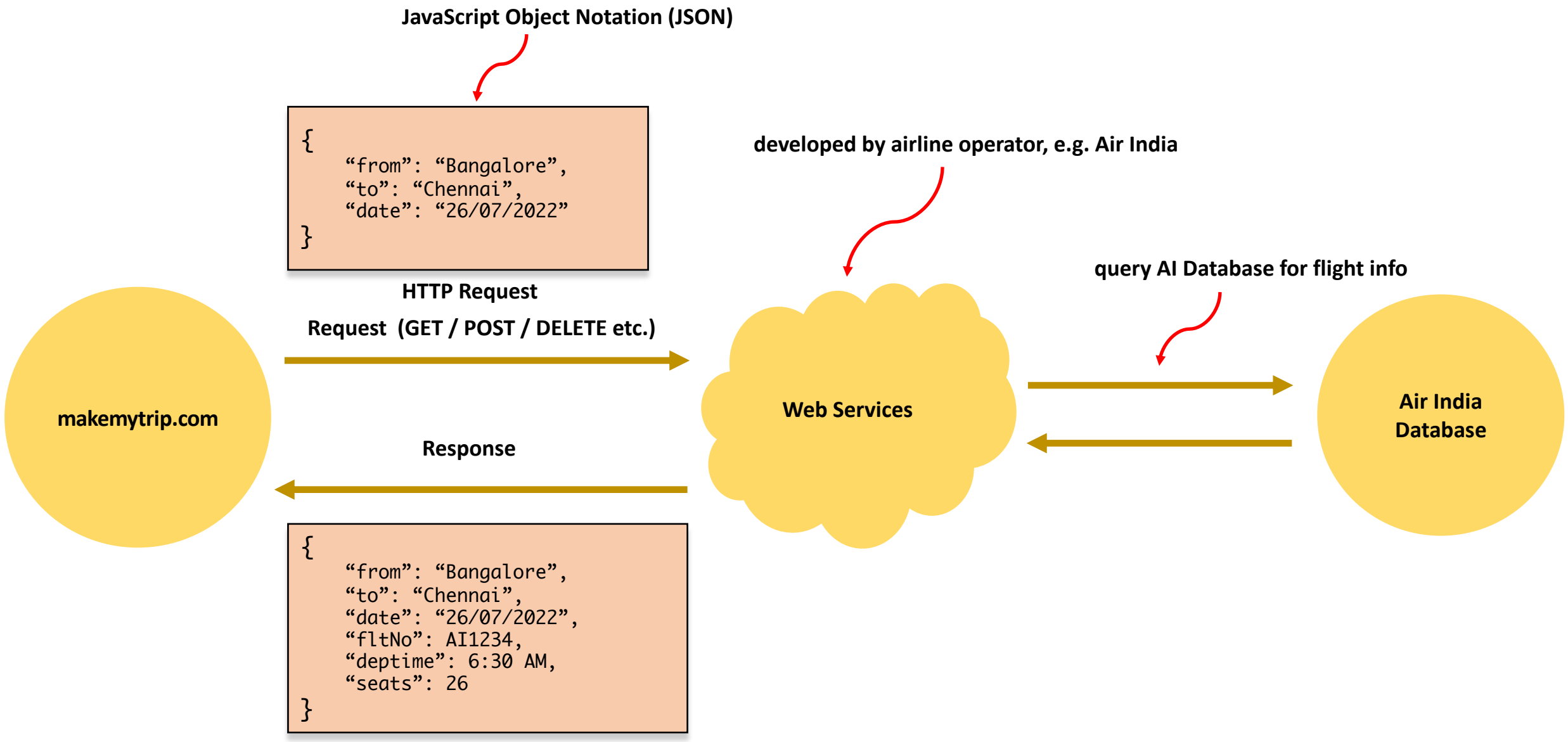
Demo Websites for Practice

- http://demowebshop.tricentis.com/
- https://services.smartbear.com/samples/TestComplete14/smartstore/
- https://demo.actitime.com/login.do
- https://opensource-demo.orangehrmlive.com/index.php/auth/login

Selenium Documentation

- https://www.selenium.dev/documentation/en/

ONLINE FLIGHT RESERVATION

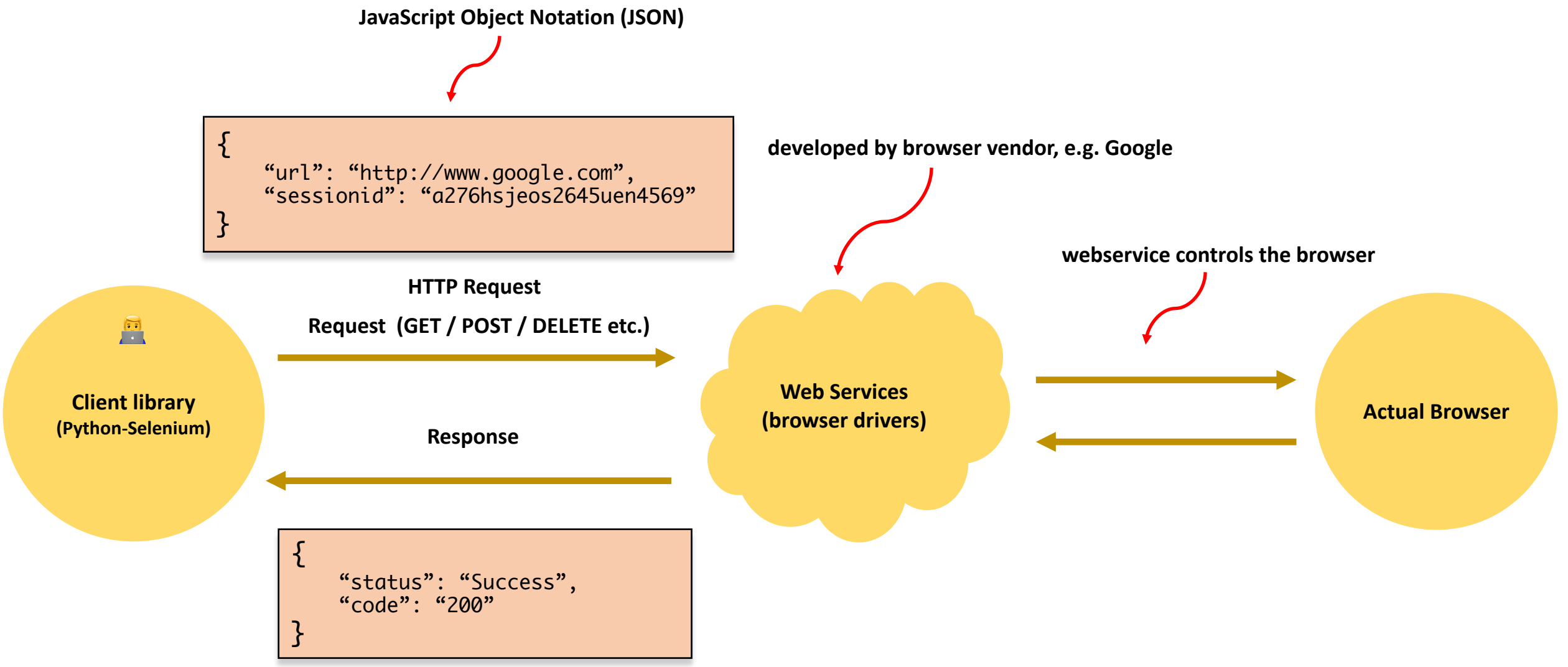


BROWSER AUTOMATION
USING
PYTHON-SELENIUM

SANDEEP SURYAPRASAD

SELENIUM ARCHITECTURE

SELENIUM ARCHITECTURE



CLIENT LIBRARIES / LANGUAGE BINDINGS

Client libraries

- 👉 Selenium supports multiple languages such as Python, Java, C#, JavaScript, Ruby.
- 👉 Client libraries provide various methods to perform different browser actions. e.g. get, title, find_element
- 👉 As automation developers we call these methods from our development environment. e.g. VSCode
- 👉 Once we execute the script, the client libraries convert our code that we have written into a JSON (JavaScript Object Notation) format and sent as a request to Driver over HTTP.

BROWSER DRIVERS

Browser drivers

- Each browser has its own implementation of **"WebDriver" protocol** (mandatory services that need's to be implemented in order for selenium to interact with browser) called drivers.
- The browser drivers are responsible for controlling the actual browser's since the browser implementation details will be known only to the developer of driver.
- Each browser driver will be maintained by respective browser vendor. e.g. Chrome Driver is maintained by Google and Safari Driver is maintained by Apple.
- Each method in the client library is mapped to a specific web-service in the driver.
- The driver interprets the incoming request from the client and controls the actual browser.
- Once the browser operation is complete, the response is sent back to the client/client library by driver in JSON format.
- Client library interprets the JSON response and prints the response in readable format in the VSCode console.

SELENIUM BINARIES

Install selenium

```
pip install selenium
```

Browser drivers

Browser	Maintained By	Download Link
CHROME	GOOGLE	DOWNLOAD
SAFARI	APPLE	BUILT IN
FIREFOX	MOZILLA	DOWNLOAD
EDGE	MICROSOFT	DOWNLOAD

BROWSER AUTOMATION
USING
PYTHON-SELENIUM

SANDEEP SURYAPRASAD

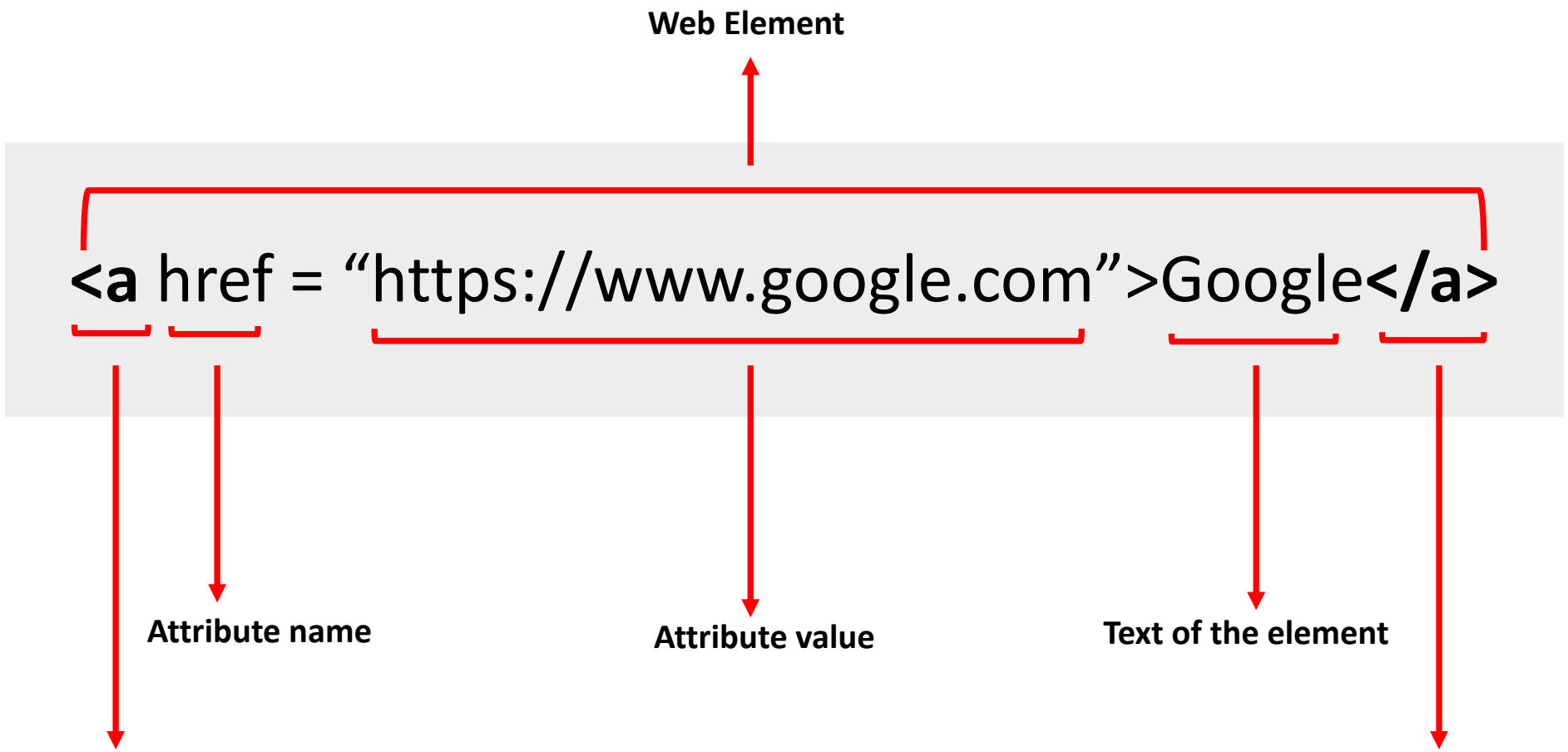
HTML FUNDAMENTALS

HTML

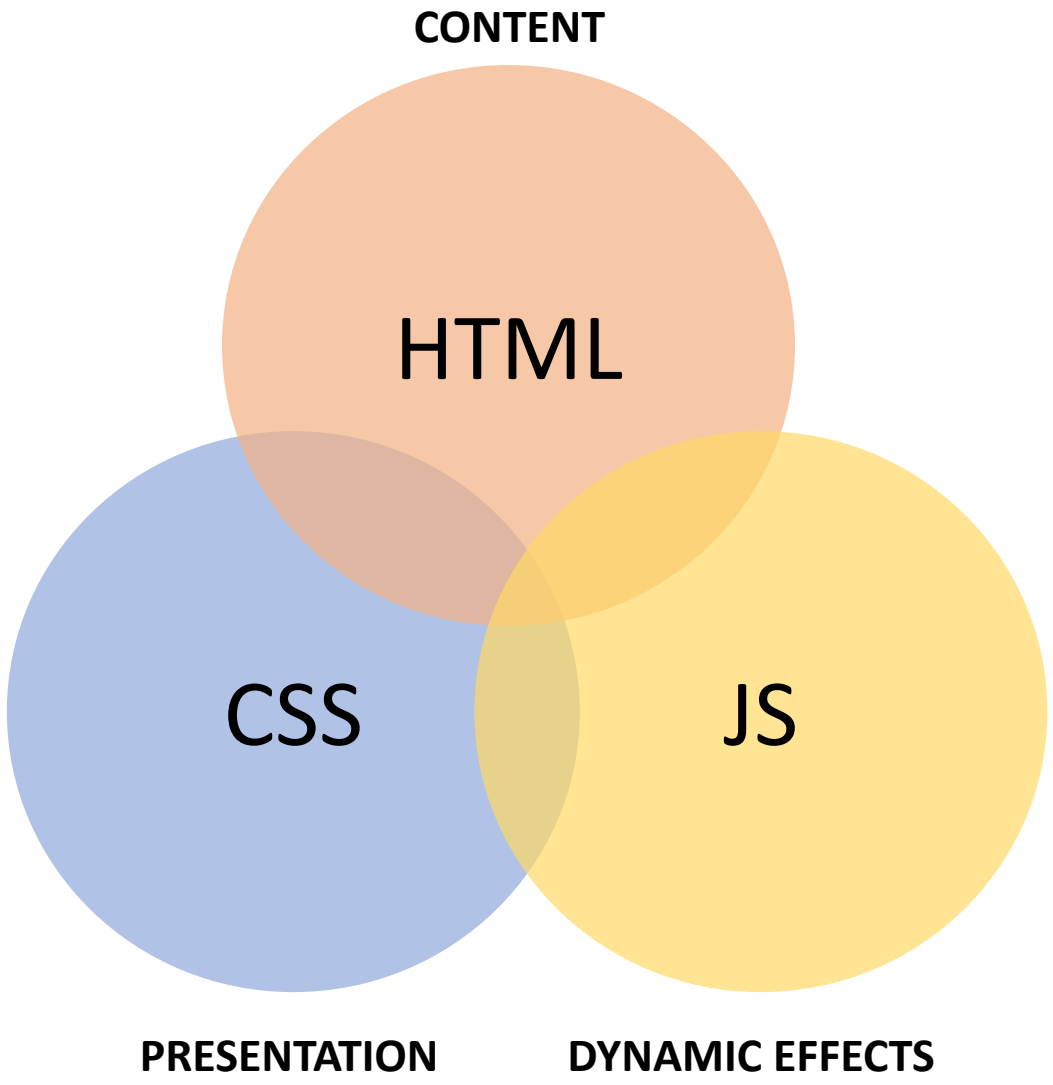
HTML

- **H**yper **T**ext **M**arkup **L**anguage
- HTML is a markup language that web developers use **to structure and design the content** of a webpage (HTML is not a programming language)
- HTML consists of **elements (web elements)** that describe different types of contents like links, headings, images, text boxes, radio buttons, check-boxes etc.
- Browser understands HTML and **renders HTML code as websites**

HTML ELEMENT



3 LANGUAGES OF FRONT-END



BROWSER AUTOMATION
USING
PYTHON-SELENIUM

SANDEEP SURYAPRASAD

BROWSER NAVIGATION

LAUNCHING DIFFERENT BROWSERS

PATH OF DRIVER EXECUTABLE

```
from selenium import webdriver

driver = webdriver.Chrome("./chromedriver")
```

CREATING AN INSTANCE OF CHROME BROWSER

```
from selenium import webdriver

driver = webdriver.Firefox("./geckodriver")
```

CREATING AN INSTANCE OF FIREFOX BROWSER

```
from selenium import webdriver

driver = webdriver.Safari()
```

CREATING AN INSTANCE OF SAFARI BROWSER

PATH OF SAFARI DRIVER WILL BE AUTOMATICALLY TAKEN FROM THE PATH /USR/BIN

COMMON BROWSER ACTIONS

```
from selenium import webdriver
driver = webdriver.Chrome("./chromedriver")
driver.get("http://www.google.com")
```

LAUNCH URL

```
from selenium import webdriver
driver = webdriver.Chrome("./chromedriver")
driver.get("http://www.google.com")
driver.maximize_window()
```

MAXIMIZES THE BROWSER

```
from selenium import webdriver
driver = webdriver.Chrome("./chromedriver")
driver.get("http://www.google.com")
driver.minimize_window()
```

MINIMIZES THE BROWSER

```
from selenium import webdriver
driver = webdriver.Chrome("./chromedriver")
driver.get("http://www.google.com")
driver.refresh()
```

REFRESH THE BROWSER

COMMON BROWSER ACTIONS

```
from selenium import webdriver
driver = webdriver.Chrome("./chromedriver")
driver.get("http://www.google.com")
current_title = driver.title
print(current_title)
```

FETCHES THE TITLE OF THE WEBPAGE

```
from selenium import webdriver
driver = webdriver.Chrome("./chromedriver")
driver.get("http://www.google.com")
url = driver.current_url
print(url)
```

FETCHES THE CURRENT URL OF THE WEBPAGE

```
from selenium import webdriver
driver = webdriver.Chrome("./chromedriver")
driver.get("http://www.google.com")
driver.quit()
```

CLOSES THE CURRENT BROWSER SESSION INCLUDING ANY POP-UP WINDOWS OPENED BY SELENIUM

```
from selenium import webdriver
driver = webdriver.Chrome("./chromedriver")
driver.get("http://www.google.com")
driver.close()
```

CLOSES THE CURRENT BROWSER SESSION, BUT DOES NOT CLOSE ANY POP-UP WINDOWS OPENED BY SELENIUM

BROWSER AUTOMATION USING PYTHON-SELENIUM

SANDEEP SURYAPRASAD

LOCATING ELEMENTS

find_element

- `find_element` method returns a web element if the element is found in the DOM/HTML.
- If no element is found, `find_element` method throws "`NoSuchElementException`"
- If there are multiple elements matching the same locator, `find_element` method returns the first matching element from the DOM.

LOCATING ELEMENTS

- In order for selenium to identify elements on the webpage, selenium makes uses of 8 different locators to identify the objects.
- There are 8 different built-in element location strategies in Webdriver

Locator	Description
name	Locates elements whose NAME attribute matches the search value
id	Locates elements whose ID attribute matches the search value
link text	Locates anchor elements (link) whose visible text matches the search value
partial link text	Locates anchor elements (link) whose visible text contains the search value. If multiple elements are matching, only the first one will be selected.
css selector	Locates elements matching a CSS selector
class name	Locates elements whose class name contains the search value (compound class names are not permitted)
xpath	Locates elements matching an XPath expression
tag name	Locates elements whose tag name matches the search value