סיכום שיעור רביעי:

Web automation

1. HTML

- HTML stands for hypertext Markup Language, and it is the most widely used language to write Web Pages.
- HTML makes use of various tags to format the content.
- These tags are enclosed within angle braces <Tag Name>.
- The HTML document is divided into two major parts:
    - **HEAD**: contains information about the document:
        - Title of the page
        - Meta tags: used to describe the content
        - JavaScript and Style sheets generally require statements in the Head
    - **BODY**: Contains the actual content of the document
        - This is the part that will be displayed in the browser window
- <input font="David" SIZE=5 COLOR="blue" CLASS="main" ID="123"> </FONT>

https://www.w3schools.com/Html/

2. CSS

- CSS defines how HTML elements will be designed displayed
- Styles were added to HTML 4.0 to solve a problem
- CSS saves a lot of work
- External Style Sheets are stored in CSS files
- The CSS file will be called from the HTML file:
    - <head> <1ink rel="stylesheet" type="text/css" href="mystyle.css"> </head>
    -

https://www.w3schools.com/Css/

JS (JavaScript) .3

- JS Is a high-level, dynamic, untyped, and interpreted programming language.
- Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production.
- The majority of websites employ it, and all modern Web browsers support it without the need for plug-ins.
- JavaScript can be placed in the <body> and the <head> sections of an HTML page.
- In HTML, JavaScript code must be inserted between <script> and </script> tags.

  https://www.w3schools.com/jS/default.asp

Selenium .4

- Selenium WebDriver is a tool for automating web application testing, and in particular to verify that they work as expected
- Selenium web driver :
  - Create robust, browser-based regression automation suites and tests
  - Scale and distribute scripts across many environments
  - Use a collection of languages to build your tests

Browsers support	Languages support	OS support
Google Chrome	Python	Windows
Internet explorer	Java	Linux
Firefox	C#	Mac
Opera	Ruby	
Android	PHP	
iOS	Perl	

- In order to use Selenium libraries, simply open CMD/Terminal and type:

```
pip install selenium
```

- If everything worked well, you should see:

```
Successfully installed selenium-
```

- To validate Selenium is installed, you can type **from selenium import webdriver**
- Default pip installation will be at **\\Python34\Lib\site-packages**

  https://www.seleniumhq.org

Initialization:

- Selenium-WebDriver makes direct calls to the browser using each browser's native support for automation.
- Each browser has it's compatible driver.
- Selenium has the ability to work with mobile devices browsers as well
- As we learned before, selenium supports many browsers, and the driver name will be accordingly:
  - self.driver = webdriver.Chrome();
  - self.driver = webdriver.Firefox()
  - ……..
- Before using drivers we will need to download the corresponding driver.
- We can find those with a simple Google search: "selenium X driver" or find it in selenium download page: http://www.seleniumhq.org/download/
- In order to use the browser, we have two options:
  - Add driver executable to PATH system variable (permanent approach)
  - The second way is adding the driver executable path into the brackets using executable_path argument:

```
from selenium import webdriver

driver = webdriver.Chrome(executable_path="C:\\Users\dgotl\Desktop\ChromeDriver.exe")
```

WebDriver methods:

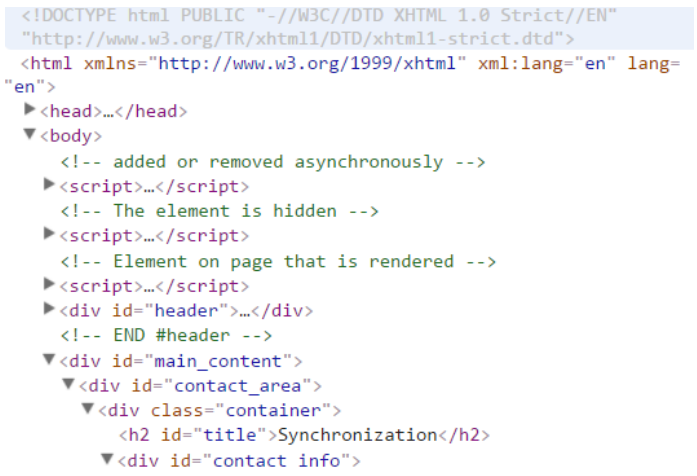
שימוש	פונקציה
פתיחת דף אינטרנט	get
מחזירה את כתובת דף האינטרנט	current_url
מחזירה את הכותרת של ה tab	title
מחזירה את מקור (source) דף האינטרנט	page_source
סגירת טאב/דפדפן	close/quit
מחזירה את המזהה היוניקי של דף האינטרנט	window_handle

WebDriver navigation:

- Example: driver.back()/forward()/refresh()…
- Purpose: Allowing the driver to access the browser's history and to navigate.
- *All methods can be easily found in the IDE when typing driver.<all options will show>
- And of course a further documentation in Selenium website..

WebDriver locators:

- Selenium uses what is called locators to find and match the elements of your page that it needs to interact with. There are a few locators strategies included in Selenium, here are some of them:
  - **ID**
  - **Name**
  - **LinkText**
  - **Partial LinkText**
  - **Class Name**
  - **Xpath**



  - The syntax is: driver.find_element_by_**<Locator Type>**(<Locator Value>))

- For example: `driver.find_element_by_name("q")`
- or `driver.find_element(By.ID,value="q")`

Locator	Webpage example	Syntax – driver.find_element_by_
ID		id(“watch”))
Name	<input name=“user”/>	name(“user”))
LinkText	<class=“a">first</…>	link_text(“firstl"))
Partial LinkText	<class=“a">first</…>	partial_link_text(“fi"))
Class Name	<div class=“test">	class_name(“test”))
Xpath	All elements can be found like that	XPath("//*[@id='rightbar']))

- In Selenium automation, if the elements are not found by the general locators like id, class, name, etc. then XPath is used to find an element on the web page.
- XML path expression. XPath is used to find the location of any element on a webpage using HTML DOM structure.
- Xpath is extremely useful when the other locators don’t exist or are not reliable.
- The syntax is as following:

```
Xpath=//tagname[@attribute='value']
```

  - To find the above controller I can use Type or value or even both as locators

```
▶ <div id="gt-lang-tgt">…</div>
▼ <div id="gt-lang-submit">
    <input type="submit" id="gt-submit" value="Translate" tabindex="0" class="jfk-button jfk-button-action">
  </div>
</div>
```

```
driver.find_element_by_xpath("//input[@type='submit']"))
driver.find_element_by_xpath("//input[@tabindex='0']"))
driver.find_element_by_xpath("//input[@type='submit' and @tabindex='0']"))
```

Relative / absolute path:

- There are two types of **XPath**:
    - **Absolute Xpath** - It is the direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed.
    - The key characteristic of XPath is that it begins with the single forward slash(/) ,which means you can select the element from the root node.
    - Example:

```
html/body/div[1]/section/div[1]/div/div/div[1]/div[3]/div[1]/div/h4[1]/b
```

    - **Relative Xpath** - For Relative Xpath the path starts from the middle of the HTML DOM structure. It starts with the double forward slash (//), which means it can search the element anywhere at the webpage.
    - You can starts from the middle of the HTML DOM structure and no need to write long xpath.

```
//div/h4[1]/b
```

    - There are many free plugins that can help you find them

5. Controllers

- Controller is an Element located on DOM (Document Object Model) which gives you the ability to control some aspects in you web application
- The simplest example for an element which is a controller is <Button>
- Examples for elements that are not a controllers = <head> <p> <body>….
- The following list has examples of controllers supported by Selenium:

פונקציה	שימוש
click	לחיצה
submit	שליחתform(דף שדות טקסט)
sendKeys	שליחת תווים (טקסט) לשדה טקסט
Clear	מחיקת תווים (טקסט) משדה טקסט

```
button_element.click()
```

```
driver.find_element_by_iid("myTextArea ")).sendKeys("My Name Is..")
```

```
driver.findElement(By.id("myTextArea ")).clear()
```

Verification and assert .6

Purpose	Syntax	Returns
Verify element is on screen	element.**is_displayed()**	boolean
Verify if element is enabled	element.**is_enabled()**	boolean
Verify if element is selected	element.**is_selected()**	boolean
Gets element text	element.**text()**	String
Gets element attribute	element.**get_attribute("value")**;	String

Synchronization .7

- Synchronization is a mechanism which involves more than one components to work parallel with Each other.
- Generally in Test Automation, we have two components
  1. Application Under Test
  2. Test Automation Tool.
- Both these components will have their own speed.
- We should write our scripts in such a way that both the components should move with same and desired speed, so that we will not encounter "Element Not Found" error.
- Synchronization can be classified into two categories:

  1. Unconditional
  2. Conditional Synchronization

Unconditional

- In this we just specify timeout value only. We will make the tool to wait until certain amount of time and then proceed further.
- *The best example will be* *time.sleep(<AMOUNT OF SECONDS>);*
- The main disadvantage for the above statements are, there is a chance of unnecessary waiting time even though the application is ready.
- The advantages are like in a situation where we interact for third party systems like interfaces, it is not possible to write a condition or check for a condition.
- Here in this situations, we have to make the application to wait for certain amount of time by specifying the timeout value.

Conditional Synchronization

- We specify a condition along with timeout value, so that tool waits to check for the condition and then come out if nothing happens.
- It is very important to set the timeout value in conditional synchronization, because the tool should proceed further instead of making the tool to wait for a particular condition to satisfy.
- An implicit wait is to tell WebDriver to poll the DOM for a certain amount of time when trying to find an element or elements if they are not immediately available.
- The default setting is 0.
- Once when we define the implicit wait, it will set for the life of the WebDriver object instance.
- Implicit wait will not work all the commands/statements in the application such as: driver.manage().window().maximaize(); It will work only for “find_element" and “find_elements" statements.
- The wait will be defined with the length of the wait and with a time unit as shown below

```
self.driver.implicitly_wait(10)
```

Keys .8

- On occasion you'll come across functionality that requires the use of keyboard key presses in your tests.
- The solution is very easy, all that we need to do is to add the following import:

```
from selenium.webdriver.common.keys import Keys
```

- Then issue a key press by using the send_keys command:

```
driver.find_element(By.ID,value="123").send_keys(Keys.ENTER)
```

- Examples:
    - Keys.ENTER
    - Keys.TAB
    - Keys.ESCAPE
    - Keys.DELETE
    - Keys.F5
    - And many many more..

https://www.seleniumhq.org/docs/03_webdriver.jsp

תזכורת ממה שעשינו בכיתה:

```
# setting up chrome browser on a desired page
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
driver = webdriver.Chrome(executable_path="C:\\Users\dgotl\Desktop\Latest selenium
drivers\ChromeDriver.exe")
# opening chrome browser on a desired page
driver.get("http://www.walla.co.il")
# Getting browser current page URL
print("url: "+driver.current_url)
# Getting browser current page title
print("title: "+driver.title)
# Getting browser current tab handle
print("handle: "+driver.current_window_handle)
# Getting browser first tab handle
print("handles: "+driver.window_handles[0])
# Getting current page source
print("source: "+driver.page_source)
# Closing current tab
driver.close()
# Closing driver session
driver.quit()
# Refreshing current page
driver.refresh()
# Navigating to previous page
driver.back()
# Navigating to next page
driver.forward()
############### ELEMENTS METHODS################
# Finding elements
driver.find_element_by_name("q")
driver.find_element(By.ID,value="q")
# Clicking an element
driver.find_element_by_id("firstButton").click()
# Submitting a form using an element
driver.find_element_by_id("submitButton").submit()
# Sending keys to an element
driver.find_element_by_id("textField").send_keys("hello")
# Sending keyboard keys to an element
driver.find_element(By.ID,value="123").send_keys(Keys.ENTER)
# Clearing an element
driver.find_element_by_id("textField").clear()
############### ELEMENTS INFORMATION ################
# Checking if element is displayed
driver.find_element_by_id("element").is_displayed()
# Checking if element is enabled
driver.find_element_by_id("element").is_enabled()
# Checking if element is selected
driver.find_element_by_id("element").is_selected()
# Getting an element text
print(driver.find_element_by_id("element").text)
# Getting an element attribute
print(driver.find_element_by_id("element").get_attribute("value"))
############### WAITS ################
# Waiting for X amount of seconds
time.sleep(1)
# Waiting up to a certain amount of seconds to find element/s
driver.implicitly_wait(10)
```