Monday

DATE : 18 : 01 : 2021

Testing

Software testing :- Once the application is build, verifying whether the application is build working as per the requirement is called.

* 2 divisions in Software testing,

1) Manual
   Verification of application done by human interaction

2) Automation
   ↳ Automation tools are used to verification.

* **Selenium** :- it is an Open Source test automation tool for testing web applications

* Selenium is also used for web Administration Service i.e, for e.g. I am the Gmail admin and i needed to Create an account for 100's of users, this can be easily done by using Selenium tool

* History of Selenium

* In 2004 JSON Huggins @ Thought-Works while testing web applications found some repetitive tasks and to test it he developed a Javascript program to run tasks

DATE : ☐☐ : ☐☐ : ☐☐☐☐

* Later he thought it could be used for other web apps as well, but there was an issue i.e, JS program was not satisfying the same origin policy

Same origin policy

↳ JS of one application cannot access or interact with pages of any other application. This is bcz of security reason

To resolve this they developed on Server with logics so that it could believe that the JS running and JS of applications are looks to same. The name for Server they had given is S.R.C Selenium Remote Control

Finally officially Selenium core [JS program + Selenium Remote Control] is launched to automate web applications At end 2006

In 2006 again S.R.C was enhanced and it takeover Selenium Core and all Selenium core libraries were dropped and alone S.R.C was released officially which is also called as Selenium 1

Page No. :

DATE : ☐☐ : ☐☐ : ☐☐☐☐

* Selenium RC had to trick the browser and perform browser interactions via J.S engine via J.S Server. It was very slow

* In 2011 webdriver was introduced which was created to perform browser interactions using native events which was faster.

Native events means interacting directly with browsers using browser API's and not through J.S Engine. So webdriver was faster than Compare to RC

* In 2011 officialy [Selenium 2.0] was released with the Combination of [Selenium 1 + Webdriver]

* In 2016 Selenium 3.0 was released officialy which had webdriver

* Selenium 4.0 ⇒ [Webdriver + W3C Webdriver]

W3C webdriver
↳ world wide web consortium
have their own Standard protocalls

* Selenium 5.0 ⇒ only W3C. webdriver

DATE : □□ : □□ : □□□□

* In 2007 one new Component of Selenium was released to perform Compatibility testing of web applications. It was named as Selenium Grid

* Using Selenium Grid we can test the web applications on different machines and different browsers

* In 2006 Selenium IDE was developed by Japanese and donated it to Selenium, which will perform record & playback of web applications on firefox browser

Selenium Core	→	IS pgm + S.R.C.
Selenium 1	→	Enhanced version of S.R.C
Selenium 2	→	Selenium 1 + webdriver

* Advantages of Selenium :-

1) Free and open source
   * Cost of the project is reduced
   * It can be Customized

2) Platform independent
   It support all the OS platforms
   (operating system)

DATE : ☐☐ : ☐☐ : ☐☐☐☐

3) Selenium Supports 14+ languages

- Supports 14+ languages
- Both program languages and Automation language are same developers can also help automation engineer

4) Supports multiple browsers

- **Disadvantages** :-

1) It can automate only web applications, It can not automate Standalone applications, mobile applications, window based applications

2) 100 % Security is not possible as it is open-source.

3) OTP, Audio, Video, Captcha, Fingerprints Cannot be automated

- Selenium Architecture

DATE : ☐☐ : ☐☐ : ☐☐☐☐

* Components to be installed

I. install Java-Script — i.e, Node.js
Node.js is a wrapper written around Javascript

II. install Javascript Language binding — i.e, Selenium webdriver
npm install Selenium-Webdriver

III. install Drivers e.g Chrome

IV. unzip and Set path of drivers in System env variable

V. Install V.S Code

* Basics

1) Creat a folder from that folder open Cmd promt and type Cmd `npm init -y` this will install the package.json file inside the folder. Package.Json Consist of language dependencies that related to your project

2) `npm install Selenium-webdriver` installs node_modules in your folder, node-modules is a Consist of JS lang -uage bindings

DATE : ☐☐:☐☐:☐☐☐☐

3) Create a folder manually with .JS extension
Code inside it

Const {Builder, Browser} = require ('Selenium-webdriver')
new Builder().forBrowser(Browser.CHROME).build()

* Frequently used methods :-

get(), getTitle(), getCurrentURL(), getPageSource()

navigate().to(), navigate().back(), navigate().forward(), navigate().refresh()

getwindowHandle(), getwindowHandles(), Close(), Quit()

* Create 2 .js files to demonstrate the frequently used methods 1) launch.js 2) windowHandle.js

19/01/21
Tuesday

* Locator Programs

1) id 2) name 3) TagName 4) ClassName 5) LinkText

6) PartialLinkText 7) CSS 8) Xpath.

DATE : ☐☐ : ☐☐ : ☐☐☐☐

* Take the Login page and demonstrate all the Locator programs e.g. facebook Login

* CSS Selector :-

* Cascading Style Sheets

Syntax :- Tagname [attrname = " attrvalue "]

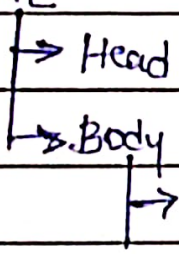
⇒ Shortcut for id e.g: div#username

. (dot) ⇒ Shortcut for ClassName div.inlet

* Drawback of CSS Selector is it doesnot supports text method.

* Xpath :- path of an element in HTML tree

for. e.g



To identify a Element we use ./html[1]/body[1]/a[1]

or

html/body/a

DATE : ☐☐:☐☐:☐☐☐☐

* How to verify xpath in browsers

1) In Chrome

I) press Ctrl + Shift + i

II) Ctrl + F ⇒ Search bop will be displayed

You can verify your xpath in that Search bop

2) In FireFox browse

1) we have to install an Add-on in FireFox

2) Go to tools in FireFox browser

3) Click on Add-on

4) Click on extensions

5) In Searchbop field enter try xpath and hit enter

6) you'll get Try xpath click on it and Click on Add to firefox

7) Now you will get an icon on ~~Title~~ Tool bar try xpath

There you can verify your X-path

Tools → Addons → Extensions → Search try xpath → install

↓

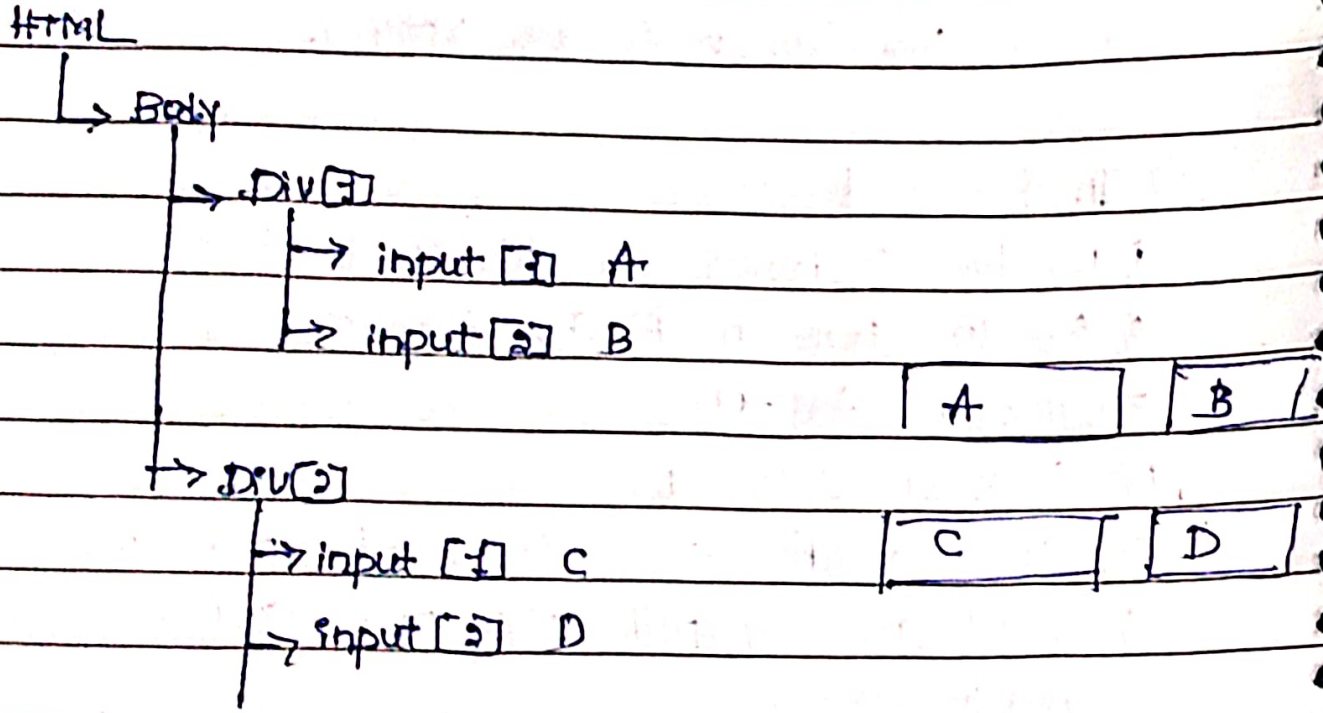
Click ok

DATE : □□:□□:□□□□

2 types of X.path

1) Absolute xpath 2) Relative X.path

1) Absolute xpath



i) TO locate all ABCD

html/body/div/input

2) TO locate A

html/body/div[1]/input[1]

like for B, C, D

DATE : ☐☐ : ☐☐ : ☐☐☐☐

3) TO locate AB

html/body/div[1]/input

like for CD

4) To locate AC, [we have to use or (|) operator]

html/body/div[1]/input[1] | html/body/div[2]/input[1]

like for AD, BC, BD

In real time we are not using absolute X-path bcz length of expression is large so we use Relative xpath

2) Relative X-path

* Starts with // (double forward Slash)

1) TO locate A,B,C,D

//input

2) To locate AB

//div[1]/input like for A, B, C, D

3) TO locate A and C

//div[1]/input[1] | //div[2]/input[1]

or

like B and C, BD, etc....

//div/input[1]

DATE : □□ : □□ : □□□□

* **XPath by attribute** :-

Syntax: //Tagname [@attributename = 'attrvalue']

* We can use multiple attributes in single xpath expression by using and Operator

e.g <input type="text" value="A"> | A | B |
<input type="text" value="B">
<input type="button" value="A">

1) using and operator to identify A

//input [@type='text' and @value="A"]

⇒ 1 matching element

2) using or operator

//input [@type="text" or @value="A"]

⇒ 2 matching elements

3) using not operator

//input [not(@type="button")]

⇒ No. of matches 2

DATE : □□:□□:□□□□

* Xpath by text() and Contains() functions :-

* If attribute is matching with xle elements & if attribute is not present then we can identify element by using Text

1) text() function

* Syntax:-

//TagName [text() = "value"]

e.g 1) login btn of actitime
2) user menu btn present in Actitime home page

2) Contains() function :-

Syntax:-

//TagName [Contains (@attrName, "value")]

or

//TagName [Contains(text(), "value")]

e.g username field and password field in actitime login page

DATE : □□ : □□ : □□□□

* Xpath by Axes

+ Traversing from 1 element to another element we use axes

* Following are the important xpath axes

1) Child /
2) Descendent //
3) Parent /..
4) Ancestor
5) Following Siblings and Preceding Siblings

e.g Create one html table and explain every thing

```
<html>
  <body>
    <table>
      <tr>
        <td>      <td>
        —||—
        —||—
```

td
element / parent :: tr
tr / parent :: table
table / ancestor :: html etc.

Sl.no	Cars	bikes
1	BMW	Duca

DATE : □□ : □□ : □□□□

* Following-Sibling :-

td[2]
td[3]

//td[text()="SL.NO"]/following-Sibling::td

=> it will match cars and bikes

* preceding-Sibling

td[1]
td[2]

//td[text().="Bikes"]/preceding-Sibling::td

=> it will match cars and SL.NO

* Handling Dynamic Elements :-

* Independent and Dependent xpath
  * Unique element is called independent element
  * Dynamic element is called dependent element
  . bcz it is dependent on unique element

e.g

01	KTM	18

e.g. identify Xpath of telphno of mumbai present in irctc.com

//h6[text()="Mumbai"]/../li[2]

independent elemt parent Child[2] desendent

DATE : □□:□□:□□□□

* Steps to identify the dynamic web element

I) Identify the independent web-element

II) Through independent element find Common parent which is Common for dependent and independent element

III) Then from Common parent navigate/traverse to the dynamic element

* Xpath by group index :-

e.g

Sub	Cost
JS	500
JS	700

JS itself is duplicate so go for indexing

//td[text()="JS"][2]/../td[3]

=> matches 700 which is dynamic

DATE : □□:□□:□□□□

* Xpath additional ways

//a ⇒ All links ⇒ ABCDEF

//a[1] ⇒ All 1st links ⇒ ACE

(//a)[1] ⇒ A

(//a)[last()] ⇒ F

(//a)[last()-1] ⇒ E

(//a)[position()<3] ⇒ AB

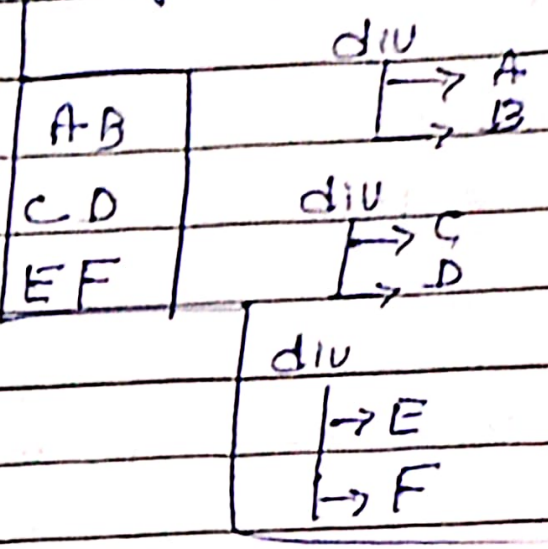
(//a)[position()>last()-2] ⇒ EF

(//a)[position() mod 2 = 0] ⇒ All Even position Links

(//a)[position() mod 2 = 1] ⇒ All odd position Links

eg <body>

AB
CD
EF



DATE : □□:□□:□□□□

* Login Script :-

* Scenario

1) Login to demo.actitime.com with username = admin
pwd = manager

2) Verify the title and URL

* How to verify webelement is present and displayed

Use try catch method

e.g.

```
driver.get ("https://www.demo.actitime.com")
try{
    let ele = driver.findElement(By.xpath(xpath))
    if (ele.isDisplayed()){
        log("Msg")
    } else {
        log("Msg")
    }
} catch (error){
    log (error.name)
}
```

DATE : □□:□□:□□□□

to verify

* write a Script, whether the textbox is empty or Not
  - use ele.getAttribute("value") and if, else blocks to print messages

* Different ways of Clicking a button
  1) ele.click()
  2) ele.SendKeys(Key.ENTER) [use vtiger login url]
     ↓
     add key in object destructuring.

* JavaScript executor :-
  * Using JS Code inside the Selenium i.e, inside the webpage (or app) opened by Selenium

e.g. Code to execute JS from Selenium to print hi msg from alert box

→ driver.get("Skillmary.com")
driver.executeScript("alert('hi from webdriver')");

DATE: ☐☐:☐☐:☐☐☐☐

* Scrolling webpage is not possible by Selenium

driver.executeScript("window.scrollBy(0, 500)")

↳ Scrolldown

driver.executeScript("window.scrollBy(0, -500)")

↳ Scrollup

* Scroll webpage to particular element [url ebay.com]

1) find the element uniquily

2) get the position of element by ele.getRect()

3) driver.~~find~~ executeScript("window.scrollBy(${position.x}, ${position.y})")

* Code to enter the value inside text field without using SendKeys()

driver.executeScript("document.getElementById("username").value='Mohan'")

DATE : □□:□□:□□□□

* Handling dropdowns :-

whenever we place mouse on the element it will displays list of options which is called as dropdown menu

* For mouse hovering we use move() method from actions() object

* Scenario :- Script to select clients and click on Design in ~~www.getstime.com~~ https://qaquag.com

1) first find the dropdown element and store it one variable
2) TO mouse hover use actions() object and move() metho
i.e, driver.actions().move(origin:element).perform()
3) Then Select the required element

* Scenario :- Select "News" and Click on "press Release" in www.istqb.org (Assignment)

* Different actions performed by using Actions() object
1) Drag and Drop
2) Drag and Drop inside iFrame
3) Cntrl + Click
4) Right Click
5) Double Click

DATE : □□:□□:□□□□

URL for Sample Code:-

1) https://demo.guru99.com/test/drag_drop.html ⇒ Drag & Drop

driver.actions().dragAndDrop(source, destination).perform()

2) Drag and Drop inside an i-Frame

URL ⇒ https://queryui.com/droppable/

* before performing dragAndDrop operation first switch to frame by using

iframeelement = driver.findElement(By.className(" "))

driver.switchTO().frame(iframe_element)

3) CTRL+Click and ~~Context Click~~

URL ⇒ https://demo.actitime.com

* identify the element on which you have to perform ContextClick and store it in a variable ele.

* driver.actions().sendKeys(Key.CONTROL).click(ele).perform()

* Context Click (Right Click)

driver.actions().contextClick().perform()

DATE : ☐☐ : ☐☐ : ☐☐☐☐

4) Double Click

URL ⇒ http://demo.guru99.com/test/simple_context_menu.html

ele = " "

driver.actions().doubleClick(ele).perform()

* Handling multiple elements :-

Scenario to Count no. of links present in web page (demo actiTime)

print all the text of identified elements (use for loop)

findElement()	findElements()
* returns first matching elem	* returns all matching element
* if locator is not matching throws noSuchElementError	* returns an empty array

* WebdriverIO :-

is a wrapper written on top of default Selenium webdriver js to enhance the functionality like running the script in synchronous and asynchronous mode.

* Installation Steps :-

npm init -y ⇒ package.json

npm install webdriverio@4.0

P.T.O

DATE : □□ : □□ : □□□□

npm install selenium-standalone --save-dev

.\node_modules\.bin\selenium-standalone install
(install different browser drivers and other stuffs)

.\node_modules\.bin\selenium-standalone start

* How to launch browser

```
Const {remote} = require ('webdriverio')
Const capabilities = {
        desiredCapabilities : {
                browserName : 'Chrome'
        }
}
remote (capabilities).init( )   or  Const browser = remote (Cap).init()
```

* Selectors :- * No findelement or findelements in webdriver
* instead we have $ (Select)

DATE : □□:□□:□□□□

Different type of Selectors

1) for Link text (login)
e.g Const link = $("=login")

2) for partial link text (*=)
e.g Const link = $("*=log")

3) Element with text
e.g. <td> hello </td>
$("td = hello") or $("td*= hell")

4) By id and Classname text
e.g welcome to Gmail
Const ele = $(".~~sam~~header = welcom to Gmail")
or
Const ele = $("#001 = welcome to Gmail")

5) Tagname
$(Tagname)

6) Xpath
$("//tagname[@att=attvalue]")

DATE : □□:□□:□□□□

* Methods in webdriverio (browser)

1) setValue ("value")
2) addValue ("value")
3) clearElement (Selector)
4) Click (Selector) ⇒ ele.click()
5) doubleClick (Selector)
6) SelectByAttribute ('Selector', 'attr', value) } facebook login page
7) SelectByIndex (Selector, index) } Birthday
8) SelectByValue (Selector, 'value')
9) SelectByVisibleText ('Selector', 'text')
10) SubmitForm ("Selector")
11) getAttribute (Selector, attr)
12) getElementSize (Selector)
13) getLocation (Selector, position)
14) getSource()
15) getTagName (Selector)
16) getText (Selector);
17) getTitle(); ⇒ Returns the title of application
18) getUrl() =
19) .getValue (Selector) ⇒ Returns the value
20) browser.alertDismiss()
21) browser.alertAccept()
22) browser.alertText ("text")

DATE : ☐☐:☐☐:☐☐☐☐

23) browser.back(), browser.forward(), browser.refresh()

24) browser.elementActive():

25) browser.frame(id)

id → String / number / null / webelementJSONObject

26) browser.saveScreenshot("location")

27) browser.url("")

28) browser.windowHandle(), browser.windowHandles()

29) browser.windowHandleMaximize()

30) browser.isEnables("Selector")

31) browser.isExisting("Selector")

32) browser.isSelected(".Selector")

33) browser.isVisible("Selector")

34) browser.waitForExist('Selector', 'MSG')

35) browser.waitForVisible('——ll——)

36) browser.waitUntil(Condition[,timeout][,timeout-msg][,interval])

DATE : ☐☐ : ☐☐ : ☐☐☐☐

* Mocha Framework

* Mocha is a feature-rich JS test framework running on node JS and browser, making Asynchronus testing Simple

* It provides an organized way of testing apps
* provides test retry support
* Provides test-specific timeouts
* reports test-duration
* highlights slow tests
* Supports use of assertion library
* provids mocha hooks before(), beforeEach(), after() and

DATE : □□:□□:□□□□

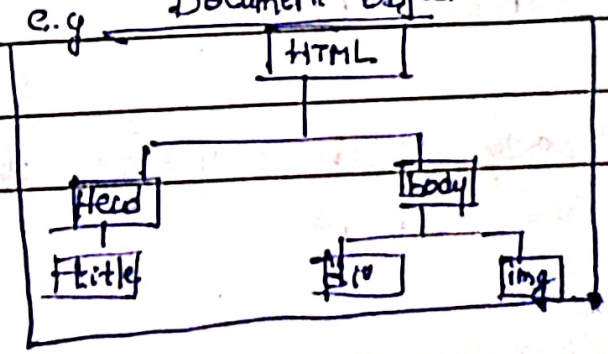
Document object Model :

* Object Oriented representation of web page
* Document is a webpage created using HTML, Every element of document is a part of DOM
* DOM Can be used to modify web page using Scripting language. Such a Java Script
* There a lot of properties, methods and events available in DOM for manupulating and Creating web pages
* DOM Can be used in Conjunction with any languages

Accessing the DOM

* Different browsers have different implimentations of DOM
* DOM Can be accessed using document object in JS
* document is nothing but HTML element
* document.CreateElement() - which will Create the element in HTML tree

e.g Document Object



Document is an object
Tags are the Contents of object
document.html
document.body

DATE : ☐☐ : ☐☐ : ☐☐☐☐

* Fundamental data types (interfaces)
  1. document 2. node 3. element

* node object :-

* Every object located within the document is a node
* An object can be a :- element node, text node, Attribute node

* Methods of node object :-
  e.g. : nodeobject.html in D:\Javascript-pratice

  appendChild(achild), Contains(otherNode), removeChild(child)

* Element object :-

* It refers to an element or a node of type element returned by a member of DOM API

* Frequently used methods of element object :

* innerHTML - It is a property used to get/set the HTML contained with in the element

DATE : ☐☐:☐☐:☐☐☐☐

* addEventListener (event, function)
  ↳ This method sets up a function that will be called whenever the specified event will be triggered
  e.g * Click, * keydown, keypress, keyup, dblclick
  * mousemove, mouseover, mouseout etc.

* getAttribute (attributeName)

* setAttribute (attributeName, value)

* Window Object :-

* window Object represent new window / browser

* frequently used methods / properties
  * window.innerheight, window.outerheight, window.innerwidth
  * window.open ("url") => it will open new tab
  * window.alert( )

* Screen and location Object :- go to Console
  * Screen.height, Screenwidth, Screen.availheight

DATE : ☐☐ : ☐☐ : ☐☐☐☐

* location object :-

property

* console.log (location) , * location.hash ↵ * location.hash = "123"

location.host ↵ , location.port , location.protocol

methods :-

location.assign ("https://google.com") | location.replace ("https://google.com")

location.reload()

* history and Navigation Object :-

history.length

history.back()

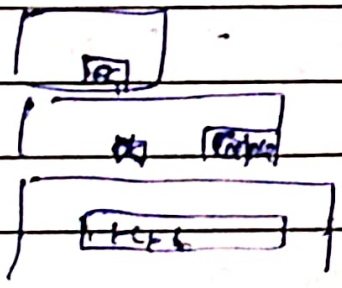
history.forward()

history.go (no.)

alert( ) =>

confirm( ) =>

prompt ( ) =>



DATE : □□:□□:□□□□

* Timeout Object

Settimeout (logic, waittime) [executes only once after 3 sec]

e.g Settimeout (( ) => { console.log("Hello") , 3000)

Set timeinterval (logic, waittime) [executes after completing each 3 sec]

e.g Set timeinterval (() => { console.log ("Hello") , 300)