Testleaf
Always Ahead

Selenium Cheat Sheet

Selenium Navigators

Navigate to url :
driver.get(String url);
driver.navigate().to(String url);

Refresh page : driver.navigate().refresh();
Navigate forwards in browser history : driver.navigate().forward();
Navigate backward in browser history : driver.navigate().back();

Selenium Locators :command which picks the respective WebElement
By.id(String id)
By.name(String name)
By.className(String className)
By.linkText(String linkText)
By.partialLinkText(String partialLinkText)
By.tagName(String tagName)
By.cssSelector(String cssSelector)
By.xpath(String xpath)

findElement :command uses By object to locate Element
Returns a single **WebElement**
Syntax: WebElement ele =driver.findElement(By.locator()) ;
Exception: NoSuchElementException

findElements : command uses By object to locates Element with multiple occurrence
Returns an **Empty List of WebElement Object**
Syntax: List< WebElement> element=driver.findElements(By.locator()) ;
Exception: Doesn't throw any exception (returns empty list)

Drop Down: -Uses Select class of selenium
Select class facilitate 3 methods to select options of dropdown
selectByIndex (int index) - To select the option at the given index
selectByValue(String value) - To select the option that matches the value with the arguments
selectByVisibleText(String text) - To select the option that matches the text with the aruguments

Implicitly Wait
This wait can be considered as element detection timeout. Once defined in a script, this wait will be set for all the Web Elements on a page. Selenium keeps polling to check whether that element is available to interact with or not.
Syntax: driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(20));

Explicitly Wait
This wait can be considered as conditional wait, and is applied to a particular Web Element with a condition

Syntax:
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));
wait.until(ExpectedConditions.<condition>);
Example:
wait.until(ExpectedConditions.Visibilityof(WebElement));

Exception: TimeoutException

Element Verification: to verify the state of the WebElement
All the below methods return **Boolean** value
isEnabled() - return true if the element is enabled else return false.
isDisplayed() - return true if the element is displayed else return false.
isSelected() - return true if the element is selected else return false.

Browser Methods to get details from WebPage
All the below methods return **String** value
driver.getTitle() –To retrieve the title of the current webpage.
driver.getCurrrentUrl() –To retrieve the url of the current webpage
driver.getPageSource() –To retrieve the source of the webpage
ele.getText() –To retrieve the text in the WebElement
ele.getAttribute(String attributeName) –To retrieve the value of the attribute that we passed.

Alert : Application Popup message
driver.switchTo().alert().getText(); - To read the text from the alert
driver.switchTo().alert().accept(); - To Accept the alert
driver.switchTo().alert().dismiss(); - To cancel the alert
driver.switchTo().alert().sendKeys(String value); - To pass the values in alert text box
Exceptions:
NoAlertPresentException - Trying to handle an alert which is not present
UnhandledAlertException - Trying to access main page without handling an alert
Frames: (iframe –DOM inside DOM)
driver.switchTo().frame(int index); - switch to frame using index
driver.switchTo().frame(String nameOrId); - switch to frame using name or id
driver.switchTo().frame(WebElement frameElement); - switch to frame using WebElement of frame
driver.switchTo().parentFrame(); – Applicable for nested frames. Control will move from child frame to immediate parent frame.
driver.switchTo().defaultContent(); – Move the control from frame to the main html DOM
Exceptions:
NoSuchFrameException - When the given frame (using index / id / name) is not available in the DOM

Window Handling –handling multiple opened automated browers
String windowHandle=driver.getWindowHandle() – To get the reference for the current window
Set <String> windowHandles=getWindowHandles() - To get the references for all the windows opened by WebDriver
driver.switchTo().window(String nameorHandle); -To Switch the Focus or control to the specified window
driver.close() – To close the current window.
driver.quit() –To close all the windows opened by WebDriver.
Exception:
NoSuchWindowException - When the window handle does not exist

Advanced User Interactions - Mouse Actions
Uses Actions Class to interact with mouse and keyboards events Syntax : Actions builder = new Actions(driver); builder.methods().perform();
Click() –Click the current mouse location.
ClickAndHold() - Clicks at the present mouse location (without releasing)
contextClick() -Performs a context-click (right click) at the current mouse location.
doubleClick() –Performs a double-click at the current mouse location.
dragAndDrop(WebElement source,WebElement target) - Invokes click-and-hold at the source location and moves to the location of the target element before releasing the mouse. source – element to grab, target – element to release
movetoElement() - Moves the mouse to the middle of the element.
dragAndDropBy(WebElement source,int xOffset,int yOffset) - Invokes click-and-hold at the source location and moves to the location and release at the given offset
release() -Releases the depressed left mouse button at the current mouse location

Keyboard Actions
sendKeys(): Use this method to simulate typing into an element, which may set its value.
keyDown(): Sends a key press without release it. Subsequent actions may assume it as pressed. (example: Keys.ALT, Keys.SHIFT, or Keys.CONTROL)
keyUp(): Performs a key release.

TakeScreenShot
A Screenshot in Selenium WebDriver is used for bug analysis.
Syntax: File src = driver.getScreenshotAs(OutputType.FILE); FileUtils.copyFile(scr,new File(path));