**Handling Popups:**

Popups are anything which gives information to user or takes information/confirmation from user

1. Web-Based Popup
2. Window-Based Popup

**1.Web-Based Popup:**

a. Javascript Popup

b. Hidden Division Popup (HTML popup)

c. Browser Notification popup

**a.Javascript Popup:**

Created by using JavaScript’s

Hence the name Types

1. Alert (Only OK button)

2. Confirmation (Both OK and Cancel button)

**Characteristics:**

1.Is it Colourful? No

2.Is it Movable? No

3.Is it Inspectable??? No

4.Until we Handle this popup we can’t continue with webpage.

**How to Handle it:**

First, we need to switch our control from webpage to popup

|  |
| --- |
| **Alert al = driver.switchTo().alert();** |

After switching we handle by using the abstract methods of Alert Interface

al.accept();//Clicks on OK button

al.dismiss();//Clicks on Cancel button

al.getText();// Used to get the message on the popup

al.sendKeys("String");// Used to type any message to popup

**Note:**

* If there are no Javascript popups to handle
* but still we try to switch and Handle
* then we get NoAlertPresentException.

------------------------------------------------ \*\*\*\*\* ----------------------------------------------------------

**b.Hidden Division Popup(HTML popup):**

**Characteristics:**

1.Is it Colourful? Yes

2.Is it Movable? No

3.Is it Inspectable? Yes

**How to Handle it:**

Since We can Inspect it we can handle by using findElement(),click(), sendKeys()

------------------------------------------------ \*\*\*\*\* ----------------------------------------------------------

**c.Browser Notification Popup:**

Change the browser settings by

**ChromeOptions Class or FirefoxOptions Class**

**use addArguments("--disable-notifications")**

------------------------------------------------ \*\*\*\*\* ----------------------------------------------------------

**TAKING SCREENSHOTS:**

We use getScreenshotAs() method of TakesScreenshot Interface to take a screenshot.

getScreenshotAs() is given implementation in RemoteWebDriver Class and it is extended in browser classes.

But as a Selenium standard we always upcast our browser object to WebDriver Interface

WebDriver driver=new ChromeDriver();

As a result getScreenshotAs() is hidden and inaccessible.

Hence to access it, we either typecast from WebDriver Interface to TakesScreenshot Interface

TakesScreenshot ts = (TakesScreenshot)driver;

OR

We Downcast from WebDriver Interface to RemoteWebDriver Class

RemoteWebDriver rwd = (RemoteWebDriver)driver;

OR

Since EventFiringWebDriver Class is also implementing it, we can Create Object and access it

EventFiringWebDriver e=new EventFiringWebDriver(driver);

By above ways we can take a full page Screenshot

But if we have to take a Particular Element screenshot

Then since WebElement interface extends TakesScreenshot interface

We can access through WebElement by getting the address of element and then

Taking screenshot of it

**EXAMPLE:**

WebElement imgAddress= driver.findElement(By.id("hplogo"));

File src = imgAddress.getScreenshotAs(OutputType.FILE);

------------------------------------------------ \*\*\*\*\* ----------------------------------------------------------