

1. **ElementNotSelectableException:**This exception belongs to InvalidElementStateException class. This exception occurs when the web element is present on the web page but cannot be selected.

**Solution:**

* + Try for other interfaces to select the element (**selectByIndex**, **selectByVisibleText**).
  + A wait command can also be added to wait until the element becomes clickable.

1. **ElementNotInteractableException:**This exception is thrown when an element is present in DOM, but it is not interactable, like unable to perform any action on it such as clicking or sending keys. This happens probably when the element to be interacted with is either hidden or disabled.

**Solution:**

* + Wait until the element is visible or clickable using Explicit wait.
  + Scroll until the element gets in display using the Actions class.
  + Use JS Executor to interact directly with the DOM.

1. **ElementNotVisibleException:**Selenium throws ElementNotVisibleException when an element is present in the DOM, but it is not visible to the user.

**Solution:**

* + Try to write unique locators that match with a single element only.
  + Wait until the element is visible or clickable using explicit wait.

1. **NoSuchElementException:** This is the most common Selenium exception which is a subclass of NotFoundException class and thrown by **[findElement()](https://www.browserstack.com/guide/findelement-in-selenium" \o "findElement vs findElements in Selenium" \t "_blank)** method of Selenium WebDriver. This exception occurs when the locators defined in the Selenium program is unable to locate the element in the DOM.

**Solution:**

* + Reverify the locator by inspecting the browser and checking whether the element is present on DOM or not. Try switching to more reliable locators.
  + Wait for the element to load using explicit wait.

1. **NoSuchFrameException:** This exception occurs when Selenium is unable to locate the desired frame or iframe using the specified iframe identifier (By iframe id, name or index). Iframe is a web page inside a web page and to work with the iframe elements, we need to first switch to the desired iframe. This exception triggers when Selenium is unable to find the iframe or if it is switching to an invalid iframe.

**Solution:**

* + Use wait after the action which triggers to open the iframe to ensure the iframe is loaded properly.
  + Ensure the iframe locator is correct. (Switch between iframe name, id or index and recheck)

1. **NoAlertPresentException:** This exception occurs when Selenium tries to switch to an alert box which is not available on the web page which means the driver is switching to an invalid or non-existing Alert pop up. Like iframe, the driver first needs to switch to the desired Alert box to interact with it and then perform actions on it such as clicking on OK/ Submit/ Cancel button or fetching Alert message.

**Solutions:**

* + Use wait after the action which triggers to open the Alert to ensure it is loaded properly before interacting.
  + Ensure alert locator is correct and visible on DOM by inspected browser. Try with different alert locators.

1. **NoSuchWindowException:** Selenium throws this exception when the WebDriver cannot find the desired browser window or tab using the specified window handle or name. This may occur when the window browser or tab we are attempting to work with is either not present, has been closed during the execution or is not completely loaded.**Solutions:**
   * Ensure that the window handle or name being used is accurate.
   * Wait for the browser window or tab to completely load and then switch the WebDriver to the desired window instance.
2. **StaleElementReferenceException:** This exception pops up when the element that was referenced by the locator is no longer present in the DOM or has become stale. It is a runtime exception that occurs when the page gets dynamically loaded, deleting the referenced element completely from the DOM, so that it becomes stale.

**Solutions:**

* + Refresh the page before accessing the element that causes StaleElementException.
  + Use try-catch block to handle the exception and attempt to locate the element again in the catch block.

1. **SessionNotFoundException:** SessionNotFoundException will occur when the Webdriver is trying to perform actions on the web application after the browser is closed or when the browser session is not available.

**Solution:** To handle such exceptions, we need to revisit our code and check if the code is not accidentally closing the browser. We need to make sure that the browser remains active throughout the execution and should be closed only at the end of the execution.

1. **TimeoutException:** In Selenium, TimeOutException occurs when a command takes longer than the wait time to avoid the ElementNotVisible Exception. Due to certain environment conditions such as low internet speed or web application taking more time than usual to completely load, the element to be interacted with does not load. And in such conditions when the WebDriver tries to find the element on the webpage, **TimeoutException** occurs.

**Solutions:**

* + Check the load time of the web element manually and add wait accordingly.
  + Add explicit wait using JavaScript executor until the page is loaded.
  + Try using some other property to locate the element such as CSS Selector or Xpath.

1. **WebDriverException:** This exception occurs when the WebDriver is acting immediately after closing the driver session.

**Solution:**Use **driver.close()** after the completion of all tests at the suite level and not at the test level. If using TestNG, use **driver.close()** in **@AfterSuite** and not in **@AfterTest**.

**How to Handle Exceptions:**

from selenium import webdriver  
from selenium.common.exceptions import NoSuchElementException, TimeoutException  
from selenium.webdriver.common.by import By  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected\_conditions as EC  
  
try:  
 # Initialize WebDriver  
 driver = webdriver.Chrome()  
  
 # Navigate to the webpage  
 driver.get("https://example.com")  
  
 # Example: Try to find an element  
 try:  
 element = driver.find\_element(By.ID, "nonexistent\_element")  
 print("Element found:", element.text)  
 except NoSuchElementException:  
 print("Element not found")  
  
 # Example: Handle timeout exception for WebDriverWait  
 try:  
 WebDriverWait(driver, 10).until(EC.presence\_of\_element\_located((By.ID, "some\_element")))  
 except TimeoutException:  
 print("Timeout occurred while waiting for element")  
  
 # Other code goes here...  
  
finally:  
 # Close the browser  
 driver.quit()