**Session Plan: Using JavaScriptExecutor in Selenium**  
**Objective**: Learn JavaScriptExecutor methods in Selenium for basic interactions like clicking, sending keys, scrolling, getting values, and highlighting elements.

**Agenda**

1. **Introduction to JavaScriptExecutor**
2. **Basic Methods Overview**
3. **Demonstration of Methods**
   * **Click** using JavaScriptExecutor
   * **Send Keys** to an element
   * **Scroll** the page
   * **Get Element Value**
   * **Highlight an Element**
   * **Refresh the Page**
   * **Trigger an Alert**
4. **Hands-On Practice**
5. **Q&A**

**Introduction to JavaScriptExecutor**

* JavaScriptExecutor is an interface in Selenium for directly interacting with web pages using JavaScript.
* It is especially useful for interacting with elements that are hidden or dynamically loaded.

**Syntax to initialize JavaScriptExecutor**:

java

JavascriptExecutor js = (JavascriptExecutor) driver;

**Basic Methods Overview**

Here are some basic JavaScriptExecutor methods:

| **Action** | **JavaScriptExecutor Syntax** | **Use Case** |
| --- | --- | --- |
| **Click** | js.executeScript("arguments[0].click();", element); | Click on elements not clickable through WebDriver. |
| **Send Keys** | js.executeScript("arguments[0].value='text';", element); | Enter text in input fields. |
| **Scroll** | js.executeScript("window.scrollBy(0,500);"); | Scroll down the page. |
| **Get Element Value** | String value = (String) js.executeScript("return arguments[0].value;", element); | Retrieve input field value. |
| **Highlight Element** | js.executeScript("arguments[0].style.border='3px solid red'", element); | Highlight elements for debugging. |
| **Refresh the Page** | js.executeScript("location.reload();"); | Reload the current page. |
| **Trigger an Alert** | js.executeScript("alert('This is an alert!');"); | Generate browser alerts. |

**Demonstration of Methods**

**1. Click on an Element**

java

WebElement button = driver.findElement(By.id("submitButton"));

js.executeScript("arguments[0].click();", button);

**2. Send Keys to an Input Field**

java

WebElement inputField = driver.findElement(By.id("username"));

js.executeScript("arguments[0].value='TestUser';", inputField);

**3. Scroll Examples**

* **Scroll Down by Pixels**:

java

js.executeScript("window.scrollBy(0,500);");

* **Scroll to a Specific Element**:

java

WebElement footer = driver.findElement(By.id("footer"));

js.executeScript("arguments[0].scrollIntoView(true);", footer);

**4. Get Element Value**

Retrieve the current value of an input field:

java

WebElement inputField = driver.findElement(By.id("username"));

String value = (String) js.executeScript("return arguments[0].value;", inputField);

System.out.println("Input Value: " + value);

**5. Highlight an Element**

Useful for debugging or visually locating elements:

java

WebElement element = driver.findElement(By.id("highlightedElement"));

js.executeScript("arguments[0].style.border='3px solid red'", element);

**6. Refresh the Page**

java

js.executeScript("location.reload();");

**7. Trigger an Alert**

java

js.executeScript("alert('This is a JavaScriptExecutor alert!');");

**Hands-On Practice**

**Tasks**:

1. Open a demo website.
2. Perform the following:
   * Click a button that is not interactable through WebDriver.
   * Enter text into an input field using sendKeys.
   * Scroll to the bottom of the page.
   * Highlight the login button.
   * Display an alert after logging in.

**Website Suggestion**

* https://the-internet.herokuapp.com

**Q&A Session**

* Clarify doubts about the usage of JavaScriptExecutor.
* Discuss scenarios where JavaScriptExecutor is most effective.