1. Setting up selenium web driver in js

```
const { Builder } = require('selenium-webdriver');
const chrome = require('selenium-webdriver/chrome');

(async function setupSelenium() {
  let driver = await new Builder()
    .forBrowser('chrome')
    .setChromeOptions(new chrome.Options())
    .build();

  try {
    console.log('Selenium WebDriver setup successful!');
    await driver.get('https://www.example.com'); // Open a test website
    await driver.sleep(2000); // Wait for 2 seconds
  } catch (err) {
    console.error('Error:', err);
  } finally {
    await driver.quit(); // Close the browser
  }
})();
```

- **Behavior:** Opens Chrome, navigates to https://www.example.com, waits 2 seconds, then closes.
- **Console Output:**

Selenium WebDriver setup successful!

2. Locating elements with selenium webdriver in js

```
const { Builder, By } = require('selenium-webdriver');
const chrome = require('selenium-webdriver/chrome');

(async function locateElements() {
  let driver = await new Builder()
    .forBrowser('chrome')
    .setChromeOptions(new chrome.Options())
    .build();

  try {
    await driver.get('https://www.wikipedia.org');

    // Locate elements using different locators
    let searchBox = await driver.findElement(By.name('search')); // Locate by name
    let searchButton = await driver.findElement(By.xpath("//button[@type='submit']")); // Locate by
XPath
    let logo = await driver.findElement(By.css('.central-textlogo__image')); // Locate by CSS selector

    console.log('Elements located successfully!');

  } catch (err) {
    console.error('Error:', err);
  } finally {
    await driver.quit();
  }
})();
```

- Navigates to Wikipedia homepage.
- Locates search box, search button, and Wikipedia logo.
- **Console Output:**

Elements located successfully!

3. Handling alerts in selenium webdriver

```
const { Builder, By, Key, until } = require('selenium-webdriver');
const chrome = require('selenium-webdriver/chrome');

(async function handleAlerts() {
  let driver = await new Builder()
    .forBrowser('chrome')
    .setChromeOptions(new chrome.Options())
    .build();

  try {
    await driver.get('https://the-internet.herokuapp.com/javascript_alerts');
    await driver.sleep(2000); // Wait 2 seconds to see the page load

    // Click button to trigger alert
    let alertButton = await driver.findElement(By.xpath("//button[text()='Click for JS Alert']"));
    await driver.sleep(2000); // Pause before clicking
    await alertButton.click();
    console.log('Clicked the alert button!');

    // Wait for the alert and accept it
    await driver.sleep(2000); // Wait to see the alert
    let alert = await driver.switchTo().alert();
    console.log('Alert Text:', await alert.getText());
    await driver.sleep(2000); // Pause before accepting
    await alert.accept(); // Click OK
    console.log('Alert accepted!');

    await driver.sleep(2000); // Pause to see the result before closing

  } catch (err) {
    console.error('Error:', err);
  } finally {
    await driver.quit();
  }
})();
```

- Opens a test page with JavaScript alerts.
- Clicks alert button, prints alert text, accepts alert.
- **Console Output:**

Clicked the alert button!
Alert Text: I am a JS Alert
Alert accepted!

4. Handling multiple windows/tabs in selenium webdriver

```
const { Builder, By, Key } = require("selenium-webdriver");
const chrome = require("selenium-webdriver/chrome");

(async function handleMultipleWindows() {
  let driver = await new Builder()
    .forBrowser("chrome")
    .setChromeOptions(new chrome.Options())
    .build();

  try {
    // Open first URL
    await driver.get("https://www.google.com");
    console.log("Opened Google");

    // Open a new tab using JavaScript
    await driver.executeScript("window.open('https://www.wikipedia.org', '_blank');");
    await driver.sleep(2000); // Wait for the tab to open

    // Get all window handles
    let windowHandles = await driver.getAllWindowHandles();
    console.log("Window Handles:", windowHandles);

    // Ensure the handles are strings before switching
    if (windowHandles.length > 1 && typeof windowHandles[1] === "string") {
      console.log("Switching to second window...");
      await driver.switchTo().window(windowHandles[1]); // Switch to new tab
      console.log("Switched to Wikipedia tab");
    } else {
      console.error("No valid second window handle found");
    }

    await driver.sleep(3000); // Wait to see the new tab

  } catch (err) {
    console.error("Error:", err);
  } finally {
    await driver.quit(); // Close browser
  }
})();
```

- Opens Google, then opens Wikipedia in a new tab.
- Switches to Wikipedia tab.
- Prints window handles.
- **Console Output:**

```
Opened Google
Window Handles: [ 'CDwindow-...', 'CDwindow-...' ]
Switching to second window...
Switched to Wikipedia tab
```

5. Taking a screenshot using selenium webdriver

```
const { Builder } = require('selenium-webdriver');
const chrome = require('selenium-webdriver/chrome');
const fs = require('fs');

(async function takeScreenshot() {
  let driver = await new Builder()
    .forBrowser('chrome')
    .setChromeOptions(new chrome.Options())
    .build();

  try {
    await driver.get('https://www.wikipedia.org');

    // Take a screenshot
    let screenshot = await driver.takeScreenshot();
    fs.writeFileSync('screenshot.png', screenshot, 'base64');

    console.log('Screenshot saved successfully as screenshot.png!');

  } catch (err) {
    console.error('Error:', err);
  } finally {
    await driver.quit();
  }
})();
```

Opens Wikipedia, takes screenshot screenshot.png in script folder.

- **Console Output:**

Screenshot saved successfully as screenshot.png!

6. Perform mouse actions using the actions class using selenium webdriver

```
const { Builder, By, Key, until } = require('selenium-webdriver');
const chrome = require('selenium-webdriver/chrome');
const { Actions } = require('selenium-webdriver');

(async function performMouseActions() {
  // Set up the WebDriver instance
  let driver = await new Builder()
    .forBrowser('chrome')
    .setChromeOptions(new chrome.Options())
    .build();

  try {
    // Navigate to Wikipedia
    await driver.get('https://www.wikipedia.org');

    // Find elements on the page for testing actions
    const englishLink = await driver.findElement(By.id('js-link-box-en')); // Find the English language
link

    // Create an instance of Actions
    let actions = driver.actions({ async: true });

    // --- Mouse Hover Action ---
    console.log('Hovering over the English language link...');
    await actions
      .move({ origin: englishLink })  // Move the mouse to the English link
      .perform();  // Perform the hover action
    await driver.sleep(2000); // Wait for 2 seconds to see the hover effec

  } catch (err) {
    console.error('Error:', err);
  } finally {
    // Close the browser
    // await driver.quit();
  }
})();
```

- Opens Wikipedia.
- Mouse hovers over English language link.
- **Console Output:**

Hovering over the English language link…

7. Perform keyboard actions using the actions class using selenium webdriver

```
const { Builder, By, Key, until } = require('selenium-webdriver');
const chrome = require('selenium-webdriver/chrome');
const { Actions } = require('selenium-webdriver');

(async function performKeyboardActions() {
  // Set up the WebDriver instance
  let driver = await new Builder()
    .forBrowser('chrome')
    .setChromeOptions(new chrome.Options())
    .build();

  try {
    // Navigate to a URL
    await driver.get('https://www.google.com');

    // Find the search input field
    const searchBox = await driver.findElement(By.name('q'));

    // Perform a series of keyboard actions using the Actions class
    let actions = driver.actions({ async: true });

    // Type a query, press 'Enter', and use a combination of keyboard actions
    await actions
      .click(searchBox) // Click the search box to focus
      .sendKeys('Selenium WebDriver') // Type text
      .sendKeys(Key.RETURN) // Press the "Enter" key (Search action)
      .perform(); // Execute the actions

    // Wait for the page to load
    await driver.sleep(2000); // Wait for 2 seconds

  } catch (err) {
    console.error('Error:', err);
  } finally {
    // Close the browser
    // await driver.quit();
  }
})();
```

Opens Google.

Focuses search box, types “Selenium WebDriver”, presses Enter.

Console Output:
(No console.log in code, but no error if successful)

Google search results page for “Selenium WebDriver” loads.

8. Handling browser navigation(back, Forward, refresh) using selenium webdriver

```
const { Builder, By, Key, until } = require('selenium-webdriver');
const chrome = require('selenium-webdriver/chrome');

(async function handleBrowserNavigation() {
  // Set up the WebDriver instance
  let driver = await new Builder()
    .forBrowser('chrome')
    .setChromeOptions(new chrome.Options())
    .build();

  try {
    // Navigate to the first URL (Google)
    console.log('Navigating to Google...');
    await driver.get('https://www.google.com');
    await driver.sleep(2000); // Wait for 2 seconds for the page to load

    // Navigate to a second URL (Example)
    console.log('Navigating to Example.com...');
    await driver.get('https://www.wikipedia.org/');
    await driver.sleep(2000); // Wait for 2 seconds for the page to load

    // --- Handle Browser Back Navigation ---
    console.log('Navigating back...');
    await driver.navigate().back();
    await driver.sleep(2000); // Wait for 2 seconds for the page to load

    // --- Handle Browser Forward Navigation ---
    console.log('Navigating forward...');
    await driver.navigate().forward();
    await driver.sleep(2000); // Wait for 2 seconds for the page to load

    // --- Handle Browser Refresh ---
    console.log('Refreshing the page...');
    await driver.navigate().refresh();
    await driver.sleep(2000); // Wait for 2 seconds for the page to refresh

  } catch (err) {
    console.error('Error:', err);
  } finally {
    // Close the browser
    await driver.quit();
  }
})();
```

Opens Google → Example.com → navigates back → forward → refresh.

- **Console Output:**

Navigating to Google...
Navigating to Example.com...
Navigating back to Google...
Navigating forward to Example.com...
Refreshing the page...

- Browser visibly navigates through pages accordingly.

9. Handling cookies demonstration (Get, Set and deleting the cookies) using selenium webdriver

```
const { Builder, By, Key, until } = require('selenium-webdriver');
const chrome = require('selenium-webdriver/chrome');

(async function handleCookies() {
  // Set up the WebDriver instance
  let driver = await new Builder()
    .forBrowser('chrome')
    .setChromeOptions(new chrome.Options())
    .build();

  try {
    // Navigate to a website (Google)
    await driver.get('https://www.google.com');

    // --- Set a Cookie ---
    await driver.manage().addCookie({
      name: 'testCookie',
      value: 'testValue',
      domain: 'google.com',  // Set domain to 'google.com' (correct domain)
      path: '/',
      secure: false,
      httpOnly: true,
      expiry: Math.floor(Date.now() / 1000) + 60 * 60, // Expire in 1 hour
    });

    console.log('Cookie Set Successfully!');

    // --- Get the Cookie ---
    const cookie = await driver.manage().getCookie('testCookie');
    console.log('Retrieved Cookie: ', cookie);

    // --- Delete a specific Cookie ---
    await driver.manage().deleteCookie('testCookie');
    console.log('Cookie Deleted Successfully!');

  } catch (err) {
    console.error('Error:', err);
  } finally {
    // Close the browser
    await driver.quit();
  }
})();
```

Output:

- Opens Google.
- Adds a cookie named testCookie.
- Prints confirmation.
- Retrieves and prints the cookie object.
- Deletes the cookie.
- Prints confirmation.

Expected console output:

Cookie Set Successfully!
Retrieved Cookie: {
  name: 'testCookie',
  value: 'testValue',
  domain: 'google.com',
  path: '/',
  secure: false,
  httpOnly: true,
  expiry: <timestamp>
}
Cookie Deleted Successfully!

- expiry will be a Unix timestamp approx. 1 hour from current time.
- The exact formatting may vary depending on console logging but it will display the cookie object.

10. Handling drop downs using the select class

```
const {Builder, By, Key, until, WebDriver} = require('selenium-webdriver');
const {Select} = require('selenium-webdriver');  // Select is available directly from the selenium-webdriver package

(async function handleDropdown() {
  let driver = await new Builder().forBrowser('chrome').build();
  try {
    // Open Wikipedia's language selector page
    await driver.get('https://www.wikipedia.org');

    // Wait until the dropdown is present
    await driver.wait(until.elementLocated(By.id('searchLanguage')), 10000);

    // Locate the dropdown element (the language dropdown)
    let dropdownElement = await driver.findElement(By.id('searchLanguage'));

    // Create a Select object
    let select = new Select(dropdownElement);

    // Select an option by visible text (e.g., "English")
    await select.selectByVisibleText('English');

    // Optional: Wait for the page to load after selecting the language
    await driver.sleep(2000);  // Sleep for 2 seconds

    // Select an option by index (e.g., selecting the first option)
    await select.selectByIndex(0);

    // Optional: Wait for the page to load after selecting the index option
    await driver.sleep(2000);  // Sleep for 2 seconds

    // Select an option by value (e.g., "en")
    await select.selectByValue('en');

    // Optional: Wait for the page to load after selecting the value option
    await driver.sleep(2000);  // Sleep for 2 seconds

  } finally {
    await driver.quit();
  }
})();
```

Opens Wikipedia homepage.

Waits for the language dropdown.

Selects language by visible text ("English").

Waits 2 seconds.

Selects first option by index (0).

Waits 2 seconds.

Selects by value ("en").

Waits 2 seconds.

Closes the browser.