Aim: Install and explore Selenium for automated testing. write a simple program in JavaScript and perform testing using selenium

Objective:

To understand the installation and setup of Selenium in a JavaScript environment.

To create and execute a simple automated test script.

To observe browser automation and interaction with web element

Description: Selenium is a powerful tool for automating web browsers through scripts. With the help of WebDriver and JavaScript (Node.js), we can create scripts that interact with websites like a real user—clicking buttons, filling forms, and validating content.

Implementation:

1. Prerequisites Installation

a. Install Node.js

- Download Node.js from: https://nodejs.org
- Install it using the downloaded .msi installer.
- Verify installation:

```
C:\Users\geeta>node -v
v22.13.0

C:\Users\geeta>npm -v
11.0.0

C:\Users\geeta>
```

```
PS C:\Users\geeta\OneDrive\Desktop\selenium-js-test> npm init -y
Wrote to C:\Users\geeta\OneDrive\Desktop\selenium-js-test\package.json:

{
  "name": "selenium-js-test",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "type": "commonjs",
  "description": ""
}


PS C:\Users\geeta\OneDrive\Desktop\selenium-js-test>
```

```
PS C:\Users\geeta\OneDrive\Desktop\selenium-js-test> npm install selenium-webdriver

added 17 packages, and audited 18 packages in 5s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\geeta\OneDrive\Desktop\selenium-js-test>
```

Download and Set Up ChromeDriver on Windows

Step 1: Check Your Chrome Browser Version

1. Open Google Chrome.
2. Click on the three dots in the top-right corner.
3. Go to Help > About Google Chrome.
4. Note the version number (e.g., 114.0.5735.198).

Step 2: Download ChromeDriver

1. Go to: https://chromedriver.chromium.org/downloads
2. Click on the ChromeDriver version that matches your Chrome browser version.
   - If your exact version isn't listed, choose the closest *major* version (e.g., for Chrome 114.x, select ChromeDriver 114.x).
3. On the version page, download the file for Windows:
   - Click on chromedriver_win32.zip.

Step 3: Extract the ChromeDriver

1. Locate the downloaded file (chromedriver_win32.zip) in your Downloads folder.
2. Right-click on the ZIP file and select Extract All….
3. Choose a location to extract the file (for example, extract it into your project folder or C:\Tools\chromedriver\).
4. You should now see the file: chromedriver.exe.

Step 4 (Option A): Add ChromeDriver to Project Folder

1. Move the chromedriver.exe into your Node.js project folder.
2. When you run your JavaScript test script, Selenium will automatically find chromedriver.exe in the current working directory.

```
Starting ChromeDriver 135.0.7049.95 (de2eb485a1951079e63bdb57ce25544d2dc79c15-refs/branch-hea
ds/7049@{#1836}) on port 0
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keepi
ng ChromeDriver safe.
ChromeDriver was started successfully on port 63340.
```

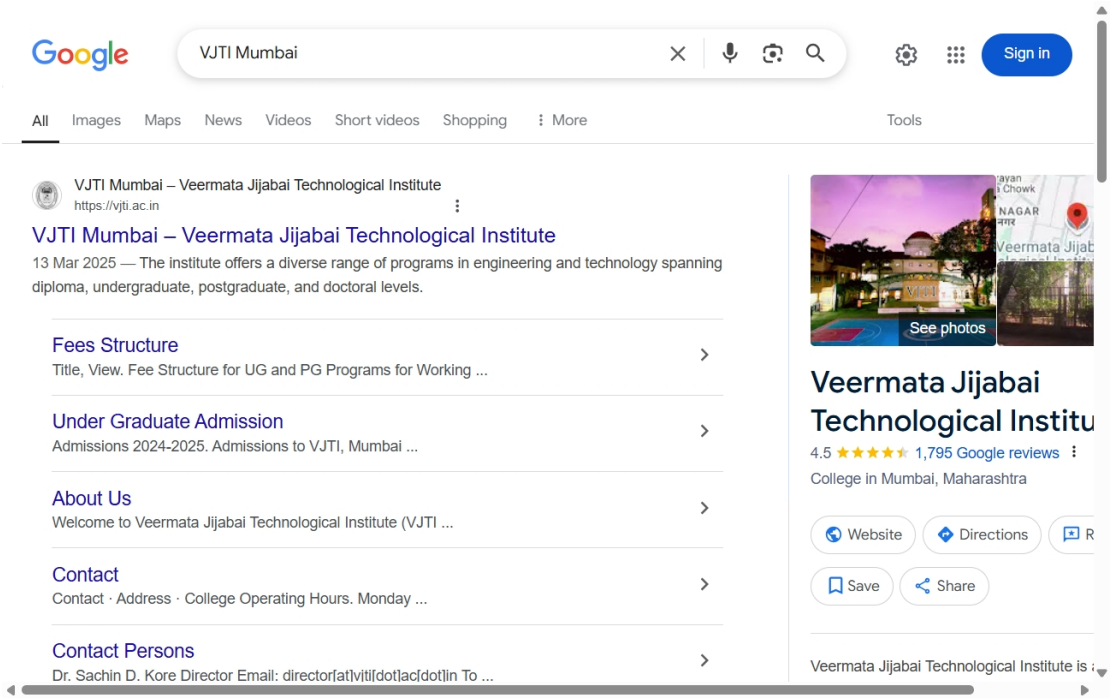
Welcome | test.js

test.js > ...

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

(async function searchVJTI() {
  // ✅ Corrected: create new Chrome options instance with `new`
  let options = new chrome.Options();
  // Uncomment the line below to run in headless mode
  // options = options.headless();

  let driver = await new Builder()
    .forBrowser('chrome')
    .setChromeOptions(options)
    .build();

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

    // Accept cookies (if shown)
    try {
      const agreeBtn = await driver.wait(
        until.elementLocated(By.xpath("//button[contains(., 'Accept') or contains(., 'I agree')]")),
        3000
      );
      await agreeBtn.click();
      console.log("Accepted cookies.");
    } catch (e) {
      console.log("No cookie popup found.");
    }

    // Search for "VJTI Mumbai"
    await driver.findElement(By.name('q')).sendKeys('VJTI Mumbai', Key.RETURN);

    // Wait for search results to load
    try {
      await driver.wait(until.elementLocated(By.css('h3')), 15000);
      console.log("Search results found!");
    } catch (e) {
      console.error("Search results not found.");
    }

    // Save screenshot
    const screenshot = await driver.takeScreenshot();
    fs.writeFileSync('search_result_final.png', screenshot, 'base64');
    console.log("Screenshot saved as search_result_final.png");

    // Show the page title
    const title = await driver.getTitle();
    console.log('Page Title:', title);

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

Run the Test:

```
PS C:\Users\geeta\OneDrive\Desktop\selenium-js-test> node test.js

DevTools listening on ws://127.0.0.1:63652/devtools/browser/5bb4a06b-e3cf-4ed9-8ea0-1397eb13d7f7
No cookie popup found.
DevTools listening on ws://127.0.0.1:63652/devtools/browser/5bb4a06b-e3cf-4ed9-8ea0-1397eb13d7f7
No cookie popup found.
Search results found!
Search results found!
Screenshot saved as search_result_final.png
Page Title: VJTI Mumbai - Google Search
PS C:\Users\geeta\OneDrive\Desktop\selenium-js-test>
```



Outcome:

Google Chrome should launch automatically.

The term " VJTI Mumbai" should be entered into the search box.

Google search results should appear.

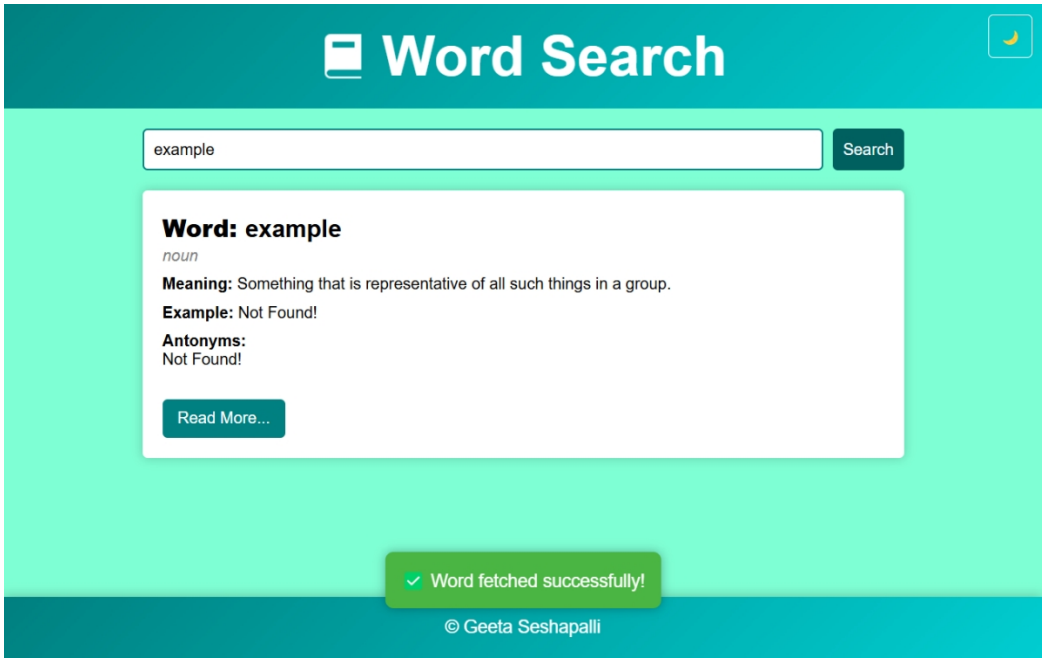
Console should log Search results found!

Handling:

- Cookie handling Detects and clicks "Accept all" or similar button
- Broader result wait Waits for any <h3> tag, which all result titles use
- Screenshot saves a screenshot (vjti_search_result.png) for debugging if something fails

Example 2:

```
JS test.js > ...
const {Builder, By, until} = require('selenium-webdriver');
const assert = require('assert');
const fs = require('fs');
async function testDictionaryApp() {
    // Start a new browser session
    let driver = await new Builder().forBrowser('chrome').build();
    try {
        // Open the dictionary app
        await driver.get('file:///C:/Users/geeta/OneDrive/Desktop/WordSearch/index.html');

        // Find the input element and submit a word
        await driver.findElement(By.css('input[type="text"]')).sendKeys('example');
        await driver.findElement(By.css('button[type="submit"]')).click();

        // Wait for the result to load
        const resultDiv = await driver.wait(until.elementLocated(By.css('.result')), 10000);
        await driver.sleep(2000);

        // Take a screenshot after the result is loaded
        let screenshot = await driver.takeScreenshot();
        fs.writeFileSync('result_screenshot.png', screenshot, 'base64');

        // Get the text of the result and assert that the word is fetched
        const resultText = await resultDiv.getText();
        console.log('Result Text:', resultText);
        assert(resultText.includes('Word: example'), 'The word is not fetched correctly.');

    } catch (error) {
        console.error('Test failed:', error);
    } finally {
        // Quit the driver
        await driver.quit();
    }
}
// Run the test
testDictionaryApp();
```



Description: result_screenshot.png

```
noun
Meaning: Something that is representative of all such things in a group.
Example: Not Found!
DevTools listening on ws://127.0.0.1:60795/devtools/browser/3e99c17f-d655-4cb7-b52c-b424430c9a1e
Result Text: Word: example
noun
Meaning: Something that is representative of all such things in a group.
Example: Not Found!
Result Text: Word: example
noun
Meaning: Something that is representative of all such things in a group.
Example: Not Found!
Antonyms:
Example: Not Found!
Antonyms:
Antonyms:
Not Found!
Not Found!
Read More...
PS C:\Users\geeta\OneDrive\Desktop\WordSearch>
```

Description: output

1. **Initialize Selenium WebDriver**:
   - The script starts by importing necessary modules: selenium-webdriver, assert, and fs.
   - A new WebDriver instance for Chrome is created using new Builder().forBrowser('chrome').build().
2. **Open the Word Search App**:
   - The script opens the local dictionary app page (index.html) using the .get() method with the file path (file:///C:/Users/geeta/OneDrive/Desktop/WordSearch/index.html).
3. **Submit the Word**:
   - It finds the input field using the CSS selector input[type="text"], sends the word "example" using the .sendKeys() method.
   - The submit button is clicked using the CSS selector button[type="submit"].
4. **Wait for the Result**:
   - The script waits for the result div (.result) to load, using the until.elementLocated method for a maximum of 10 seconds.
5. **Take a Screenshot**:
   - After the result is loaded, the script waits for 2 seconds to ensure the page is fully updated, then takes a screenshot using driver.takeScreenshot().
   - The screenshot is saved as result_screenshot.png in base64 format using fs.writeFileSync().

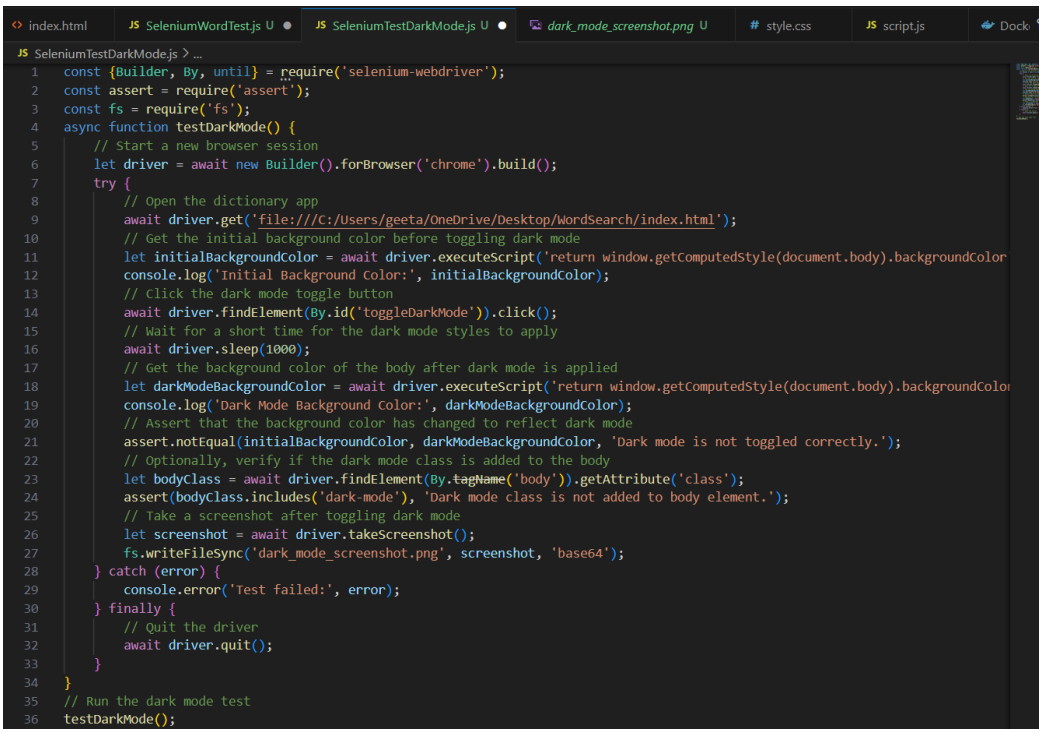
6. **Verify the Result**:
   - The text content of the result div is fetched with .getText(), and it is checked to see if it includes the word "example".
   - An assertion is made using assert(resultText.includes('Word: example')) to ensure the word is displayed in the result.
7. **Error Handling**:
   - If any error occurs during the test, it is caught in the catch block, and an error message is logged to the console.
8. **Quit the Browser**:
   - Finally, the WebDriver instance is closed using await driver.quit() to clean up the session.

Usage:

To run this script, ensure that the following requirements are met:

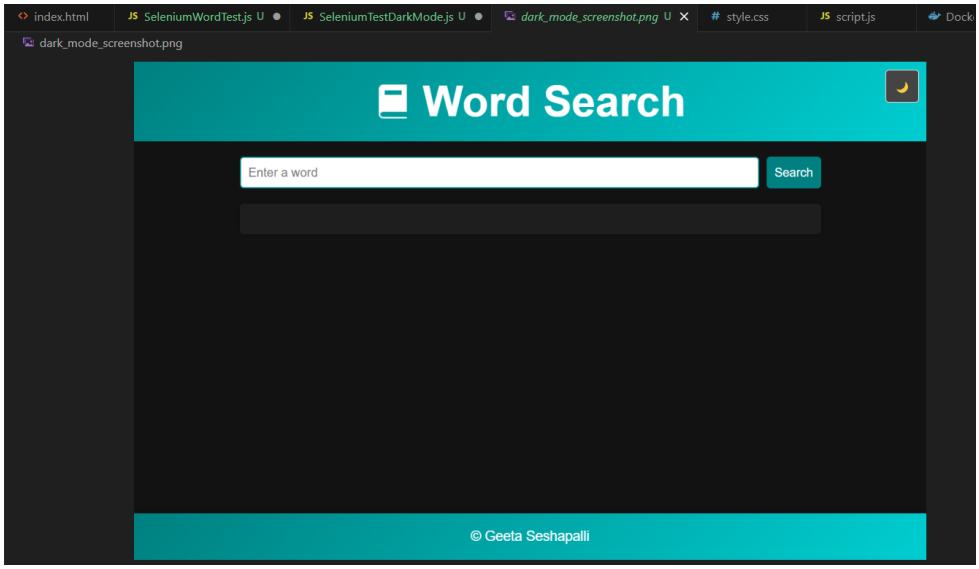
- Install necessary Node.js modules (selenium-webdriver and fs).
- Ensure that the local dictionary app (index.html) is available at the specified path.
- Run the script using Node.js.

This script is an automated test for the dark mode feature of a web application using Selenium WebDriver and Node.js. The test checks if the dark mode toggle button works correctly by verifying the background color change and the addition of a dark mode class. It also takes a screenshot after the dark mode is applied.

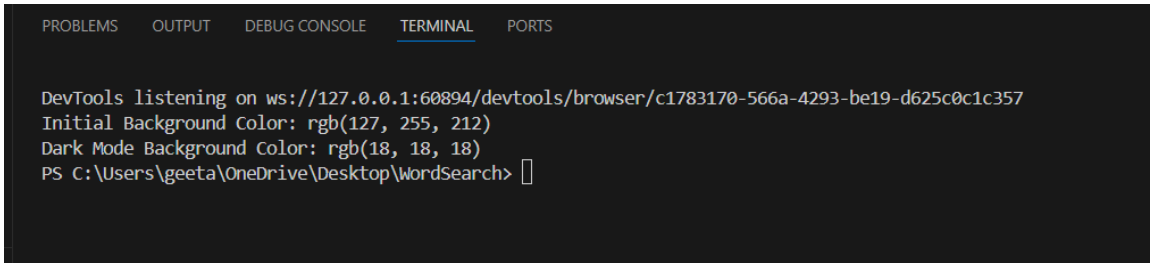



```
const {Builder, By, until} = require('selenium-webdriver');
const assert = require('assert');
const fs = require('fs');
async function testDarkMode() {
    // Start a new browser session
    let driver = await new Builder().forBrowser('chrome').build();
    try {
        // Open the dictionary app
        await driver.get('file:///C:/Users/geeta/OneDrive/Desktop/WordSearch/index.html');
        // Get the initial background color before toggling dark mode
        let initialBackgroundColor = await driver.executeScript('return window.getComputedStyle(document.body).backgroundColor
        console.log('Initial Background Color:', initialBackgroundColor);
        // Click the dark mode toggle button
        await driver.findElement(By.id('toggleDarkMode')).click();
        // Wait for a short time for the dark mode styles to apply
        await driver.sleep(1000);
        // Get the background color of the body after dark mode is applied
        let darkModeBackgroundColor = await driver.executeScript('return window.getComputedStyle(document.body).backgroundColo
        console.log('Dark Mode Background Color:', darkModeBackgroundColor);
        // Assert that the background color has changed to reflect dark mode
        assert.notEqual(initialBackgroundColor, darkModeBackgroundColor, 'Dark mode is not toggled correctly.');
        // Optionally, verify if the dark mode class is added to the body
        let bodyClass = await driver.findElement(By.tagName('body')).getAttribute('class');
        assert(bodyClass.includes('dark-mode'), 'Dark mode class is not added to body element.');
        // Take a screenshot after toggling dark mode
        let screenshot = await driver.takeScreenshot();
        fs.writeFileSync('dark_mode_screenshot.png', screenshot, 'base64');
    } catch (error) {
        console.error('Test failed:', error);
    } finally {
        // Quit the driver
        await driver.quit();
    }
}
// Run the dark mode test
testDarkMode();
```

Description: code



Description: dark_mode_screenshot.png



Description: output

Steps for Documentation:

1. **Initialize Selenium WebDriver**:
   - The necessary modules selenium-webdriver, assert, and fs are imported.
   - A new WebDriver instance for Chrome is created using new Builder().forBrowser('chrome').build().
2. **Open the Word Search App**:
   - The script opens the local dictionary app page (index.html) using the .get() method with the file path (file:///C:/Users/geeta/OneDrive/Desktop/WordSearch/index.html).
3. **Get Initial Background Color**:
   - Before toggling dark mode, the initial background color of the page is captured using executeScript to execute JavaScript on the page. The background color is logged to the console.
4. **Toggle Dark Mode**:
   - The script locates the dark mode toggle button using its id (toggleDarkMode) and simulates a click on it using .click().

5. **Wait for Dark Mode Styles to Apply**:
    - After clicking the toggle button, the script waits for 1 second (await driver.sleep(1000)) to ensure the dark mode styles have time to take effect.
6. **Get Background Color After Dark Mode**:
    - The script retrieves the background color of the body element again after the dark mode styles are applied and logs it to the console.
7. **Verify Background Color Change**:
    - An assertion is made to ensure that the background color has changed between the initial and dark mode states, indicating that the dark mode toggle has been applied correctly.
8. **Verify Dark Mode Class**:
    - The script checks if the class dark-mode is added to the body element. If not, the test will fail with an assertion error.
9. **Take a Screenshot**:
    - After toggling dark mode, the script captures a screenshot of the page using driver.takeScreenshot() and saves it as dark_mode_screenshot.png in base64 format using fs.writeFileSync().
10. **Error Handling**:
    - If any error occurs during the test, it is caught in the catch block, and an error message is logged to the console.
11. **Quit the Browser**:
    - Finally, the WebDriver instance is closed using await driver.quit() to clean up the session.

Usage:

To run this script, ensure the following:

- The dark mode toggle functionality is implemented in the dictionary app.
- The toggle button has the ID toggleDarkMode.
- Install the necessary Node.js modules (selenium-webdriver and fs).
- Ensure that the local dictionary app (index.html) is available at the specified path.
- Run the script using Node.js.

Conclusion:

This hands-on exercise demonstrates the practical use of Selenium for automating web interactions, which is a fundamental step in browser-based UI testing. It also highlights the importance of handling asynchronous content, wait conditions, and potential UI variations in real-world web automation scenarios.