

SELENIUM COMMANDS

Here are the types of Selenium commands:

- Browser Commands
- Navigation Commands
- Window Management Commands
- Frame and IFrame Commands
- Alert Commands
- Element Interaction Commands
- Element Action Commands
- Wait Commands
- Keyboard and Mouse Action Commands (Advanced User Interactions)
- Cookies Management Commands
- Screen Capture Commands
- Session Management Commands
- Browser Options and Capabilities Commands
- Logs Management Commands
- JavaScript Execution Commands
- File Upload and Download Commands
- Proxy Management Commands
- Mobile Web Testing Commands (Specific to Appium)
- Remote WebDriver Commands
- WebDriver Timeout Commands

Browser Commands

1. `driver.get(URL)`: Navigates to the specified URL in the current browser window.
2. `driver.navigate().to(URL)`: Navigates to the specified URL.
3. `driver.navigate().back()`: Moves the browser backward by one page.
4. `driver.navigate().forward()`: Moves the browser forward by one page.
5. `driver.navigate().refresh()`: Refreshes the current page.
6. `driver.manage().window().maximize()`: Maximizes the current browser window.
7. `driver.manage().window().minimize()`: Minimizes the current browser window.
8. `driver.manage().window().fullscreen()`: Sets the browser window to full screen.
9. `driver.manage().window().getSize()`: Retrieves the size (width and height) of the current browser window.
10. `driver.manage().window().setSize(new Dimension(width, height))`: Sets the size of the browser window.
11. `driver.manage().window().getPosition()`: Retrieves the current position of the browser window.
12. `driver.manage().window().setPosition(new Point(x, y))`: Sets the position of the browser window.
13. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets the implicit wait time for finding elements.
14. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Sets the maximum time to wait for a page to load.

SELENIUM COMMANDS

15. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Sets the maximum time to wait for a script to execute.
16. `driver.switchTo().frame(frame)`: Switches focus to a frame using its name or ID.
17. `driver.switchTo().frame(int index)`: Switches focus to a frame using its index number.
18. `driver.switchTo().frame(webElement)`: Switches focus to a frame using a WebElement.
19. `driver.switchTo().defaultContent()`: Switches back to the main document after working within a frame.
20. `driver.switchTo().parentFrame()`: Switches to the parent frame of the current frame.
21. `driver.switchTo().alert()`: Switches to an alert box.
22. `driver.switchTo().alert().accept()`: Accepts the currently displayed alert.
23. `driver.switchTo().alert().dismiss()`: Dismisses the currently displayed alert.
24. `driver.switchTo().alert().getText()`: Retrieves the text from the alert box.
25. `driver.switchTo().activeElement()`: Switches to the currently focused element.
26. `driver.switchTo().window(windowHandle)`: Switches focus to the specified window handle.
27. `driver.getWindowHandle()`: Retrieves the handle of the current window.
28. `driver.getWindowHandles()`: Retrieves all window handles.
29. `driver.manage().addCookie(new Cookie("name", "value"))`: Adds a cookie to the current browser session.
30. `driver.manage().deleteCookieNamed("cookieName")`: Deletes the cookie with the specified name.
31. `driver.manage().deleteAllCookies()`: Deletes all cookies.
32. `driver.manage().getCookies()`: Retrieves all cookies.
33. `driver.manage().getCookieNamed("cookieName")`: Retrieves a specific cookie by name.
34. `driver.getCurrentUrl()`: Retrieves the URL of the current page.
35. `driver.getTitle()`: Retrieves the title of the current page.
36. `driver.executeScript(script, arguments)`: Executes JavaScript code in the context of the current page.
37. `driver.executeAsyncScript(script, arguments)`: Executes asynchronous JavaScript code in the context of the current page.
38. `driver.findElement(By.id("id"))`: Locates an element by its unique ID.
39. `driver.findElement(By.name("name"))`: Locates an element by its name attribute.
40. `driver.findElement(By.className("className"))`: Locates an element by its class name.
41. `driver.findElement(By.tagName("tagName"))`: Locates an element by its tag name.
42. `driver.findElement(By.linkText("text"))`: Locates a hyperlink element by its exact text.
43. `driver.findElement(By.partialLinkText("partialText"))`: Locates a hyperlink element by partial text.
44. `driver.findElement(By.cssSelector("selector"))`: Locates an element using a CSS selector.
45. `driver.findElement(By.xpath("xpath"))`: Locates an element using an XPath expression.
46. `driver.findElements(By.id("id"))`: Locates all elements matching the ID.
47. `driver.findElements(By.name("name"))`: Locates all elements matching the name attribute.
48. `driver.findElements(By.className("className"))`: Locates all elements matching the class name.
49. `driver.findElements(By.tagName("tagName"))`: Locates all elements matching the tag name.
50. `driver.findElements(By.linkText("text"))`: Locates all hyperlink elements matching the exact text.
51. `driver.findElements(By.partialLinkText("partialText"))`: Locates all hyperlink elements matching partial text.

SELENIUM COMMANDS

52. `driver.findElements(By.cssSelector("selector"))`: Locates all elements matching the CSS selector.
53. `driver.findElements(By.xpath("xpath"))`: Locates all elements matching the XPath expression.
54. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: An updated way to set the implicit wait time using the Duration class.
55. `driver.manage().window().getRect()`: Retrieves the position and size of the browser window as a Rectangle object.
56. `driver.manage().window().setRect(new Rectangle(x, y, width, height))`: Sets the position and size of the browser window.
57. `driver.switchTo().frame(WebElement)`: Switches focus to a frame using a WebElement reference.
58. `driver.navigate().to("URL")`: Opens a new URL in the browser.
59. `driver.navigate().back()`: Moves backward in the browser's history.
60. `driver.navigate().forward()`: Moves forward in the browser's history.
61. `driver.navigate().refresh()`: Refreshes the current page.

Navigation Commands

1. `driver.get(URL)`: Navigates to the specified URL in the current browser window.
2. `driver.navigate().to(URL)`: Navigates to the specified URL.
3. `driver.navigate().back()`: Moves the browser backward by one page in the browser's history.
4. `driver.navigate().forward()`: Moves the browser forward by one page in the browser's history.
5. `driver.navigate().refresh()`: Refreshes the current page.
6. `driver.manage().window().maximize()`: Maximizes the current browser window.
7. `driver.manage().window().minimize()`: Minimizes the current browser window.
8. `driver.manage().window().fullscreen()`: Sets the browser window to full screen.
9. `driver.manage().window().getSize()`: Retrieves the size (width and height) of the current browser window.
10. `driver.manage().window().setSize(new Dimension(width, height))`: Sets the size of the browser window.
11. `driver.manage().window().getPosition()`: Retrieves the current position of the browser window.
12. `driver.manage().window().setPosition(new Point(x, y))`: Sets the position of the browser window.
13. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets the implicit wait time for finding elements.
14. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Sets the maximum time to wait for a page to load.
15. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Sets the maximum time to wait for a script to execute.
16. `driver.switchTo().frame(frame)`: Switches focus to a frame using its name or ID.
17. `driver.switchTo().frame(int index)`: Switches focus to a frame using its index number.
18. `driver.switchTo().frame(webElement)`: Switches focus to a frame using a WebElement.
19. `driver.switchTo().defaultContent()`: Switches back to the main document after working within a frame.

SELENIUM COMMANDS

20. `driver.switchTo().parentFrame()`: Switches to the parent frame of the current frame.
21. `driver.switchTo().alert()`: Switches to an alert box.
22. `driver.switchTo().alert().accept()`: Accepts the currently displayed alert.
23. `driver.switchTo().alert().dismiss()`: Dismisses the currently displayed alert.
24. `driver.switchTo().alert().getText()`: Retrieves the text from the alert box.
25. `driver.switchTo().activeElement()`: Switches to the currently focused element.
26. `driver.switchTo().window(windowHandle)`: Switches focus to the specified window handle.
27. `driver.getWindowHandle()`: Retrieves the handle of the current window.
28. `driver.getWindowHandles()`: Retrieves all window handles.
29. `driver.manage().addCookie(new Cookie("name", "value"))`: Adds a cookie to the current browser session.
30. `driver.manage().deleteCookieNamed("cookieName")`: Deletes the cookie with the specified name.
31. `driver.manage().deleteAllCookies()`: Deletes all cookies.
32. `driver.manage().getCookies()`: Retrieves all cookies.
33. `driver.manage().getCookieNamed("cookieName")`: Retrieves a specific cookie by name.
34. `driver.getCurrentUrl()`: Retrieves the URL of the current page.
35. `driver.getTitle()`: Retrieves the title of the current page.
36. `driver.executeScript(script, arguments)`: Executes JavaScript code in the context of the current page.
37. `driver.executeAsyncScript(script, arguments)`: Executes asynchronous JavaScript code in the context of the current page.
38. `driver.findElement(By.id("id"))`: Locates an element by its unique ID.
39. `driver.findElement(By.name("name"))`: Locates an element by its name attribute.
40. `driver.findElement(By.className("className"))`: Locates an element by its class name.
41. `driver.findElement(By.tagName("tagName"))`: Locates an element by its tag name.
42. `driver.findElement(By.linkText("text"))`: Locates a hyperlink element by its exact text.
43. `driver.findElement(By.partialLinkText("partialText"))`: Locates a hyperlink element by partial text.
44. `driver.findElement(By.cssSelector("selector"))`: Locates an element using a CSS selector.
45. `driver.findElement(By.xpath("xpath"))`: Locates an element using an XPath expression.
46. `driver.findElements(By.id("id"))`: Locates all elements matching the ID.
47. `driver.findElements(By.name("name"))`: Locates all elements matching the name attribute.
48. `driver.findElements(By.className("className"))`: Locates all elements matching the class name.
49. `driver.findElements(By.tagName("tagName"))`: Locates all elements matching the tag name.
50. `driver.findElements(By.linkText("text"))`: Locates all hyperlink elements matching the exact text.
51. `driver.findElements(By.partialLinkText("partialText"))`: Locates all hyperlink elements matching partial text.
52. `driver.findElements(By.cssSelector("selector"))`: Locates all elements matching the CSS selector.
53. `driver.findElements(By.xpath("xpath"))`: Locates all elements matching the XPath expression.
54. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: An updated way to set the implicit wait time using the Duration class.

SELENIUM COMMANDS

55. `driver.manage().window().getRect()`: Retrieves the position and size of the browser window as a Rectangle object.
56. `driver.manage().window().setRect(new Rectangle(x, y, width, height))`: Sets the position and size of the browser window.
57. `driver.switchTo().frame(WebElement)`: Switches focus to a frame using a WebElement reference.
58. `driver.navigate().to("URL")`: Opens a new URL in the browser.
59. `driver.navigate().back()`: Moves backward in the browser's history.
60. `driver.navigate().forward()`: Moves forward in the browser's history.
61. `driver.navigate().refresh()`: Refreshes the current page.

Window Management Commands

1. `driver.manage().window().maximize()`: Maximizes the current browser window.
2. `driver.manage().window().minimize()`: Minimizes the current browser window.
3. `driver.manage().window().fullscreen()`: Sets the browser window to full screen.
4. `driver.manage().window().getSize()`: Retrieves the size (width and height) of the current browser window.
5. `driver.manage().window().setSize(new Dimension(width, height))`: Sets the size of the browser window.
6. `driver.manage().window().getPosition()`: Retrieves the current position of the browser window.
7. `driver.manage().window().setPosition(new Point(x, y))`: Sets the position of the browser window.
8. `driver.manage().window().getRect()`: Retrieves the position and size of the browser window as a Rectangle object.
9. `driver.manage().window().setRect(new Rectangle(x, y, width, height))`: Sets the position and size of the browser window.
10. `driver.getWindowHandle()`: Retrieves the handle of the current window.
11. `driver.getWindowHandles()`: Retrieves all window handles.
12. `driver.switchTo().window(windowHandle)`: Switches focus to the specified window handle.
13. `driver.close()`: Closes the current window.
14. `driver.quit()`: Closes all windows and ends the WebDriver session.
15. `driver.switchTo().newWindow(WindowType.TAB)`: Opens a new tab in the current browser window.
16. `driver.switchTo().newWindow(WindowType.WINDOW)`: Opens a new browser window.
17. `driver.switchTo().window(driver.getWindowHandles().iterator().next())`: Switches to the first window in the list of window handles.
18. `driver.switchTo().window(driver.getWindowHandles().toArray()[index])`: Switches to a window using its index in the list of window handles.
19. `driver.switchTo().window(driver.getWindowHandles().stream().filter(handle -> handle.equals(windowHandle)).findFirst().orElse(null))`: Switches to a specific window handle using a stream filter.
20. `driver.switchTo().window(driver.getWindowHandles().stream().reduce((first, second) -> second).orElse(null))`: Switches to the last opened window handle.
21. `driver.switchTo().window(driver.getWindowHandles().stream().findFirst().orElse(null))`: Switches to the first opened window handle.

SELENIUM COMMANDS

22. `driver.getTitle()`: Retrieves the title of the current page.
23. `driver.getCurrentUrl()`: Retrieves the URL of the current page.
24. `driver.navigate().to(URL)`: Opens a new URL in the current window.
25. `driver.manage().window().setSize(new Dimension(width, height))`: Sets the browser window size to the specified width and height.
26. `driver.manage().window().setPosition(new Point(x, y))`: Moves the browser window to the specified position on the screen.
27. `driver.manage().window().maximize()`: Maximizes the browser window to fill the screen.
28. `driver.manage().window().fullscreen()`: Switches the browser to full-screen mode.
29. `driver.manage().window().minimize()`: Minimizes the browser window to the taskbar.
30. `driver.manage().window().getRect()`: Gets the current size and position of the browser window as a Rectangle object.
31. `driver.manage().window().setRect(new Rectangle(x, y, width, height))`: Sets the size and position of the browser window.
32. `driver.manage().window().getSize().getWidth()`: Retrieves the width of the browser window.
33. `driver.manage().window().getSize().getHeight()`: Retrieves the height of the browser window.
34. `driver.manage().window().getPosition().getX()`: Retrieves the X-coordinate of the browser window's position.
35. `driver.manage().window().getPosition().getY()`: Retrieves the Y-coordinate of the browser window's position.
36. `driver.manage().window().setSize(new Dimension(1920, 1080))`: Sets the browser window size to 1920x1080 pixels.
37. `driver.manage().window().setPosition(new Point(100, 100))`: Moves the browser window to the position (100, 100) on the screen.
38. `driver.manage().window().maximize()`: Expands the browser window to fill the entire screen.
39. `driver.switchTo().window(driver.getWindowHandles().toArray()[1])`: Switches to the second window handle in the list.
40. `driver.switchTo().window(driver.getWindowHandles().stream().findAny().orElse(null))`: Switches to any available window handle.
41. `driver.switchTo().window(driver.getWindowHandles().stream().sorted().findFirst().orElse(null))`: Switches to the first sorted window handle.
42. `driver.manage().window().getSize().getWidth()`: Gets the width of the current browser window.
43. `driver.manage().window().getSize().getHeight()`: Gets the height of the current browser window.
44. `driver.manage().window().getPosition().getX()`: Gets the X position of the browser window.
45. `driver.manage().window().getPosition().getY()`: Gets the Y position of the browser window.
46. `driver.manage().window().setRect(new Rectangle(0, 0, 1366, 768))`: Sets the browser window size to 1366x768 pixels and position to (0, 0).
47. `driver.manage().window().setRect(new Rectangle(200, 200, 1024, 768))`: Sets the browser window size to 1024x768 pixels and position to (200, 200).
48. `driver.manage().window().getRect().getWidth()`: Gets the width of the current browser window rectangle.

SELENIUM COMMANDS

49. `driver.manage().window().getRect().getHeight()`: Gets the height of the current browser window rectangle.
50. `driver.manage().window().getRect().getX()`: Gets the X position of the current browser window rectangle.
51. `driver.manage().window().getRect().getY()`: Gets the Y position of the current browser window rectangle.
52. `driver.manage().window().setSize(new Dimension(800, 600))`: Sets the browser window size to 800x600 pixels.
53. `driver.manage().window().setPosition(new Point(50, 50))`: Moves the browser window to the position (50, 50) on the screen.

Frames and iframes Commands

1. `driver.switchTo().frame(frame)`: Switches focus to a frame using its name or ID.
2. `driver.switchTo().frame(int index)`: Switches focus to a frame using its index number.
3. `driver.switchTo().frame(webElement)`: Switches focus to a frame using a WebElement reference.
4. `driver.switchTo().defaultContent()`: Switches back to the main document from a frame.
5. `driver.switchTo().parentFrame()`: Switches to the parent frame of the currently focused frame.
6. `driver.switchTo().frame("frameName")`: Switches focus to a frame using its name.
7. `driver.switchTo().frame("frameId")`: Switches focus to a frame using its ID.
8. `driver.switchTo().frame(0)`: Switches focus to the first frame on the page (index 0).
9. `driver.switchTo().frame(1)`: Switches focus to the second frame on the page (index 1).
10. `driver.switchTo().frame(driver.findElement(By.tagName("iframe")))`: Switches focus to the first iframe found on the page.
11. `driver.switchTo().frame(driver.findElement(By.name("frameName")))`: Switches focus to a frame identified by its name.
12. `driver.switchTo().frame(driver.findElement(By.id("frameId")))`: Switches focus to a frame identified by its ID.
13. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@name='frameName']")))`: Switches focus to a frame identified by an XPath expression.
14. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe#frameId")))`: Switches focus to a frame identified by a CSS selector.
15. `driver.switchTo().frame(driver.findElement(By.className("frameClassName")))`: Switches focus to a frame identified by its class name.
16. `driver.switchTo().defaultContent()`: Returns to the main content from a nested frame.
17. `driver.switchTo().parentFrame()`: Returns to the immediate parent frame from a nested frame.
18. `driver.switchTo().frame(driver.findElement(By.id("nestedFrameId")).findElement(By.tagName("iframe")))`: Switches to an iframe inside another frame.
19. `driver.switchTo().frame(driver.findElement(By.id("nestedFrameId")).findElement(By.name("nestedIframe")))`: Switches to an iframe inside another frame identified by name.
20. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@id='nestedFrameId']")).findElement(By.cssSelector("iframe#nestedIframe")))`: Switches to a nested iframe using XPath and CSS selector.

SELENIUM COMMANDS

21. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[1]")))`: Switches focus to the first iframe found on the page.
22. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[last()]")))`: Switches focus to the last iframe found on the page.
23. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[position()=2]")))`: Switches focus to the second iframe on the page using XPath.
24. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[contains(@src, 'frameName')]")))`: Switches focus to a frame based on a partial match of the src attribute.
25. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[starts-with(@id, 'frameIdPrefix')]")))`: Switches focus to a frame based on a prefix match of the ID attribute.
26. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[not(@id='excludedFrameId')]")))`: Switches focus to a frame excluding a specific ID.
27. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[class*='frameClassName']")))`: Switches focus to a frame based on a partial class name match.
28. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[id^='frameIdPrefix']")))`: Switches focus to a frame based on an ID prefix match using CSS selector.
29. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[id$='frameIdSuffix']")))`: Switches focus to a frame based on an ID suffix match using CSS selector.
30. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[class~='frameClassName']")))`: Switches focus to a frame based on a class name containing a specific value using CSS selector.
31. `driver.switchTo().frame(driver.findElement(By.xpath("@title='frameTitle')"))`: Switches focus to a frame identified by its title attribute.
32. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@src='frameSrc']")))`: Switches focus to a frame identified by its src attribute.
33. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[not(@src)]")))`: Switches focus to a frame that does not have a src attribute.
34. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[title='frameTitle']")))`: Switches focus to a frame identified by its title attribute using CSS selector.
35. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[src$='frameSrcSuffix']")))`: Switches focus to a frame based on an ending match of the src attribute using CSS selector.
36. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[src^='frameSrcPrefix']")))`: Switches focus to a frame based on a beginning match of the src attribute using CSS selector.
37. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[src*='frameSrcSubstring']")))`: Switches focus to a frame based on a substring match of the src attribute using CSS selector.
38. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@id='frameId'][@class='frameClass']")))`: Switches focus to a frame based on both ID and class attributes using XPath.
39. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@name='frameName' and @title='frameTitle']")))`: Switches focus to a frame based on both name and title attributes using XPath.
40. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@id='frameId' and @src='frameSrc']")))`: Switches focus to a frame based on both ID and src attributes using XPath.

SELENIUM COMMANDS

41. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[contains(@title, 'frameTitleSubstring')])))`: Switches focus to a frame based on a substring match of the title attribute using XPath.
42. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@id='frameId' or @src='frameSrc'])))`: Switches focus to a frame based on either ID or src attribute using XPath.
43. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@id='frameId' and not(@src)])))`: Switches focus to a frame based on ID attribute and without src attribute using XPath.
44. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@class='frameClass' and contains(@src, 'frameSrcSubstring')])))`: Switches focus to a frame based on class and substring match of the src attribute using XPath.
45. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@class='frameClass' or @id='frameId'])))`: Switches focus to a frame based on class or ID attribute using XPath.
46. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[not(contains(@src, 'excludedSrcSubstring'))])))`: Switches focus to a frame excluding a substring match of the src attribute using XPath.
47. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[contains(@title, 'titleSubstring') or @id='frameId'])))`: Switches focus to a frame based on title substring or ID attribute using XPath.
48. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[id|='frameIdPrefix']")))` : Switches focus to a frame based on a hyphen-separated list of ID prefixes using CSS selector.
49. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[title$='frameTitleSuffix'])))`: Switches focus to a frame based on an ending match of the title attribute using CSS selector.
50. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[class*='frameClassSubstring'])))`: Switches focus to a frame based on a class name substring match using CSS selector.
51. `driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[title*= 'titleSubstring']")))`: Switches focus to a frame based on a title substring match using CSS selector.
52. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@name='frameName'][contains(@src, 'frameSrcSubstring')])))`: Switches focus to a frame based on name and src substring match using XPath.
53. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@id='frameId' and @title='frameTitle'])))`: Switches focus to a frame based on both ID and title attributes using XPath.
54. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@name='frameName'][not(@src='excludedSrc')])))`: Switches focus to a frame based on name and excludes specific src using XPath.
55. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[@class='frameClass'][not(@title='excludedTitle')])))`: Switches focus to a frame based on class and excludes specific title using XPath.
56. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[contains(@id, 'frameIdSubstring')])))`: Switches focus to a frame based on an ID substring match using XPath.

SELENIUM COMMANDS

57. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[contains(@class, 'frameClassSubstring')])))`: Switches focus to a frame based on a class substring match using XPath.
58. `driver.switchTo().frame(driver.findElement(By.xpath("//iframe[starts-with(@id, 'frameIdPrefix')])))`: Switches focus to a frame based on a prefix match of the ID attribute using XPath.

Alert Commands

1. `driver.switchTo().alert()`: Switches to the currently displayed alert.
2. `driver.switchTo().alert().accept()`: Accepts the currently displayed alert.
3. `driver.switchTo().alert().dismiss()`: Dismisses the currently displayed alert.
4. `driver.switchTo().alert().getText()`: Retrieves the text from the currently displayed alert.
5. `driver.switchTo().alert().sendKeys("text")`: Sends the specified text to the currently displayed prompt alert.
6. `driver.switchTo().alert().accept()`: Accepts the alert or prompt.
7. `driver.switchTo().alert().dismiss()`: Dismisses the alert or prompt.
8. `driver.switchTo().alert().getText()`: Retrieves the text of the alert or prompt.
9. `driver.switchTo().alert().sendKeys("input text")`: Inputs text into a prompt alert.
10. `driver.switchTo().alert().accept()`: Accepts the confirmation dialog.
11. `driver.switchTo().alert().dismiss()`: Dismisses the confirmation dialog.
12. `driver.switchTo().alert().getText()`: Gets the text of the confirmation dialog.
13. `driver.switchTo().alert().sendKeys("sample text")`: Sends text to the prompt dialog.
14. `driver.switchTo().alert().accept()`: Accepts a prompt dialog with text.
15. `driver.switchTo().alert().dismiss()`: Dismisses a prompt dialog without text.
16. `driver.switchTo().alert().getText()`: Retrieves text from a prompt dialog.
17. `driver.switchTo().alert().sendKeys("text input")`: Inputs text into a prompt dialog.
18. `driver.switchTo().alert().accept()`: Confirms an alert dialog.
19. `driver.switchTo().alert().dismiss()`: Cancels an alert dialog.
20. `driver.switchTo().alert().getText()`: Retrieves the text from an alert dialog.
21. `driver.switchTo().alert().sendKeys("input")`: Types input into a prompt alert.
22. `driver.switchTo().alert().accept()`: Accepts an alert with a single button.
23. `driver.switchTo().alert().dismiss()`: Dismisses an alert with a single button.
24. `driver.switchTo().alert().getText()`: Gets the text content of an alert with a single button.
25. `driver.switchTo().alert().sendKeys("text")`: Sends a text string to a prompt alert.
26. `driver.switchTo().alert().accept()`: Accepts a prompt alert with text.
27. `driver.switchTo().alert().dismiss()`: Dismisses a prompt alert without entering text.
28. `driver.switchTo().alert().getText()`: Retrieves the text from a prompt alert.
29. `driver.switchTo().alert().sendKeys("sample")`: Inputs sample text into a prompt alert.
30. `driver.switchTo().alert().accept()`: Confirms a prompt alert with provided text.
31. `driver.switchTo().alert().dismiss()`: Cancels a prompt alert with provided text.
32. `driver.switchTo().alert().getText()`: Gets the prompt alert's message.
33. `driver.switchTo().alert().sendKeys("example text")`: Types example text into a prompt alert.
34. `driver.switchTo().alert().accept()`: Accepts a prompt dialog with example text.
35. `driver.switchTo().alert().dismiss()`: Dismisses a prompt dialog with example text.
36. `driver.switchTo().alert().getText()`: Gets the message from a prompt dialog with example text.

SELENIUM COMMANDS

37. `driver.switchTo().alert().sendKeys("input example")`: Enters input example into a prompt dialog.
38. `driver.switchTo().alert().accept()`: Accepts an example prompt dialog.
39. `driver.switchTo().alert().dismiss()`: Dismisses an example prompt dialog.
40. `driver.switchTo().alert().getText()`: Retrieves text from an example prompt dialog.
41. `driver.switchTo().alert().sendKeys("testing text")`: Inputs testing text into a prompt dialog.
42. `driver.switchTo().alert().accept()`: Accepts a testing prompt dialog.
43. `driver.switchTo().alert().dismiss()`: Dismisses a testing prompt dialog.
44. `driver.switchTo().alert().getText()`: Gets the testing text from a prompt dialog.
45. `driver.switchTo().alert().sendKeys("test input")`: Sends test input to a prompt dialog.
46. `driver.switchTo().alert().accept()`: Confirms a test prompt dialog.
47. `driver.switchTo().alert().dismiss()`: Cancels a test prompt dialog.
48. `driver.switchTo().alert().getText()`: Retrieves the text from a test prompt dialog.
49. `driver.switchTo().alert().sendKeys("text entry")`: Types text entry into a prompt alert.
50. `driver.switchTo().alert().accept()`: Accepts a text entry prompt alert.
51. `driver.switchTo().alert().dismiss()`: Dismisses a text entry prompt alert.
52. `driver.switchTo().alert().getText()`: Gets the text from a text entry prompt alert.

Element Interaction Commands

1. `driver.findElement(By.id("elementId"))`: Finds an element by its ID attribute.
2. `driver.findElement(By.name("elementName"))`: Finds an element by its name attribute.
3. `driver.findElement(By.className("className"))`: Finds an element by its class name.
4. `driver.findElement(By.tagName("tagName"))`: Finds an element by its tag name.
5. `driver.findElement(By.linkText("linkText"))`: Finds a link element by its visible text.
6. `driver.findElement(By.partialLinkText("partialText"))`: Finds a link element by partial match of its visible text.
7. `driver.findElement(By.xpath("xpathExpression"))`: Finds an element using an XPath expression.
8. `driver.findElement(By.cssSelector("cssSelector"))`: Finds an element using a CSS selector.
9. `driver.findElement(By.xpath("//input[@name='username']"))`: Finds an input element with the name attribute 'username'.
10. `driver.findElement(By.cssSelector("input[name='username']"))`: Finds an input element with the name attribute 'username' using CSS selector.
11. `driver.findElement(By.xpath("//button[text()='Submit']"))`: Finds a button element with the text 'Submit'.
12. `driver.findElement(By.cssSelector("button.submitButton"))`: Finds a button element with the class 'submitButton'.
13. `driver.findElement(By.xpath("//div[@class='container']//input"))`: Finds an input element inside a div with class 'container'.
14. `driver.findElement(By.cssSelector("div.container input"))`: Finds an input element inside a div with class 'container' using CSS selector.
15. `driver.findElement(By.xpath("//input[@type='text']"))`: Finds an input element with type 'text'.
16. `driver.findElement(By.cssSelector("input[type='text']"))`: Finds an input element with type 'text' using CSS selector.

SELENIUM COMMANDS

17. `driver.findElement(By.xpath("//textarea[@id='comments']"))`: Finds a textarea element with the ID 'comments'.
18. `driver.findElement(By.cssSelector("textarea#comments"))`: Finds a textarea element with the ID 'comments' using CSS selector.
19. `driver.findElement(By.xpath("//input[contains(@name, 'user')]"))`: Finds an input element with a name containing 'user'.
20. `driver.findElement(By.cssSelector("input[name*='user']"))`: Finds an input element with a name containing 'user' using CSS selector.
21. `driver.findElement(By.xpath("//button[@class='btn btn-primary']"))`: Finds a button element with classes 'btn' and 'btn-primary'.
22. `driver.findElement(By.cssSelector("button.btn.btn-primary"))`: Finds a button element with classes 'btn' and 'btn-primary' using CSS selector.
23. `driver.findElement(By.xpath("//div[@id='main']//span[text()='Hello']"))`: Finds a span element with text 'Hello' inside a div with ID 'main'.
24. `driver.findElement(By.cssSelector("div#main span.hello"))`: Finds a span element with class 'hello' inside a div with ID 'main' using CSS selector.
25. `driver.findElement(By.xpath("//a[@href='/home']"))`: Finds an anchor element with href attribute '/home'.
26. `driver.findElement(By.cssSelector("a[href='/home']"))`: Finds an anchor element with href attribute '/home' using CSS selector.
27. `driver.findElement(By.xpath("//input[@type='checkbox' and @checked]"))`: Finds a checked checkbox input element.
28. `driver.findElement(By.cssSelector("input[type='checkbox']:checked"))`: Finds a checked checkbox input element using CSS selector.
29. `driver.findElement(By.xpath("//input[@type='radio' and @value='option1']"))`: Finds a radio button input element with value 'option1'.
30. `driver.findElement(By.cssSelector("input[type='radio'][value='option1']"))`: Finds a radio button input element with value 'option1' using CSS selector.
31. `driver.findElement(By.xpath("//select[@name='options']//option[text()='Option 1']"))`: Finds an option element with text 'Option 1' inside a select element with name 'options'.
32. `driver.findElement(By.cssSelector("select[name='options'] option[value='1']"))`: Finds an option element with value '1' inside a select element with name 'options' using CSS selector.
33. `driver.findElement(By.xpath("//form[@id='loginForm']//input[@name='password']"))`: Finds a password input element inside a form with ID 'loginForm'.
34. `driver.findElement(By.cssSelector("form#loginForm input[name='password']"))`: Finds a password input element inside a form with ID 'loginForm' using CSS selector.
35. `driver.findElement(By.xpath("//div[contains(@class, 'alert')]/button"))`: Finds a button element inside a div with class containing 'alert'.
36. `driver.findElement(By.cssSelector("div.alert button"))`: Finds a button element inside a div with class 'alert' using CSS selector.
37. `driver.findElement(By.xpath("//table[@id='data']//tr[1]/td[2]"))`: Finds a table cell in the first row and second column of a table with ID 'data'.
38. `driver.findElement(By.cssSelector("table#data tr:nth-child(1) td:nth-child(2)"))`: Finds a table cell in the first row and second column using CSS selector.
39. `driver.findElement(By.xpath("//div[@id='sidebar']//a[contains(text(), 'Home')]"))`: Finds an anchor element containing text 'Home' inside a div with ID 'sidebar'.

SELENIUM COMMANDS

40. `driver.findElement(By.cssSelector("div#sidebar a:contains('Home')"))`: Finds an anchor element containing text 'Home' inside a div with ID 'sidebar' using CSS selector.
41. `driver.findElement(By.xpath("//input[@name='search']"))`: Finds a search input element by its name attribute.
42. `driver.findElement(By.cssSelector("input[name='search']"))`: Finds a search input element by its name attribute using CSS selector.
43. `driver.findElement(By.xpath("//button[contains(@class, 'submit')]"))`: Finds a button element with class containing 'submit'.
44. `driver.findElement(By.cssSelector("button.submit"))`: Finds a button element with class 'submit' using CSS selector.
45. `driver.findElement(By.xpath("//a[@class='nav-link'][text()='Contact']"))`: Finds a link with text 'Contact' and class 'nav-link'.
46. `driver.findElement(By.cssSelector("a.nav-link:contains('Contact')"))`: Finds a link with text 'Contact' and class 'nav-link' using CSS selector.
47. `driver.findElement(By.xpath("//input[@type='text' and @placeholder='Search']"))`: Finds a text input element with placeholder 'Search'.
48. `driver.findElement(By.cssSelector("input[type='text'][placeholder='Search']"))`: Finds a text input element with placeholder 'Search' using CSS selector.
49. `driver.findElement(By.xpath("//form[@id='searchForm']/input[@name='query']"))`: Finds an input element with name 'query' inside a form with ID 'searchForm'.
50. `driver.findElement(By.cssSelector("form#searchForm input[name='query']"))`: Finds an input element with name 'query' inside a form with ID 'searchForm' using CSS selector.
51. `driver.findElement(By.xpath("//span[@class='error-message']"))`: Finds a span element with class 'error-message'.
52. `driver.findElement(By.cssSelector("span.error-message"))`: Finds a span element with class 'error-message' using CSS selector.
53. `driver.findElement(By.xpath("//input[@type='submit']"))`: Finds a submit button input element.
54. `driver.findElement(By.cssSelector("input[type='submit']"))`: Finds a submit button input element using CSS selector.
55. `driver.findElement(By.xpath("//div[@id='content']/button[text()='Submit']"))`: Finds a submit button inside a div with ID 'content'.
56. `driver.findElement(By.cssSelector("div#content button.submit"))`: Finds a submit button inside a div with ID 'content' using CSS selector.
57. `driver.findElement(By.xpath("//input[@type='email' and @name='email']"))`: Finds an email input element with name 'email'.
58. `driver.findElement(By.cssSelector("input[type='email'][name='email']"))`: Finds an email input element with name 'email' using CSS selector.
59. `driver.findElement(By.xpath("//select[@id='dropdown']/option[text()='Option A']"))`: Finds an option with text 'Option A' inside a select element with ID 'dropdown'.
60. `driver.findElement(By.cssSelector("select#dropdown option[value='A']"))`: Finds an option with value 'A' inside a select element with ID 'dropdown' using CSS selector.
61. `driver.findElement(By.xpath("//input[@type='file']"))`: Finds a file input element.
62. `driver.findElement(By.cssSelector("input[type='file']"))`: Finds a file input element using CSS selector.
63. `driver.findElement(By.xpath("//textarea[@rows='4']"))`: Finds a textarea element with 4 rows.

SELENIUM COMMANDS

64. `driver.findElement(By.cssSelector("textarea[rows='4']"))`: Finds a textarea element with 4 rows using CSS selector.
65. `driver.findElement(By.xpath("//button[@type='button' and text()='Cancel']"))`: Finds a button element with type 'button' and text 'Cancel'.
66. `driver.findElement(By.cssSelector("button[type='button']:contains('Cancel')"))`: Finds a button element with type 'button' and text 'Cancel' using CSS selector.

Element Action Commands

1. `driver.click()`: Clicks on the specified element.
2. `driver.sendKeys("text")`: Sends the specified text to the element, simulating keyboard input.
3. `driver.clear()`: Clears the text input field.
4. `driver.submit()`: Submits the form containing the element.
5. `driver.getAttribute("attributeName")`: Retrieves the value of the specified attribute of the element.
6. `driver.getText()`: Gets the visible text of the element.
7. `driver.isDisplayed()`: Checks if the element is visible on the page.
8. `driver.isEnabled()`: Checks if the element is enabled.
9. `driver.isSelected()`: Checks if the element is selected, applicable for checkboxes and radio buttons.
10. `driver.hover()`: Hovers the mouse over the specified element (requires Actions class).
11. `driver.doubleClick()`: Double-clicks on the specified element (requires Actions class).
12. `driver.rightClick()`: Performs a right-click on the element (requires Actions class).
13. `driver.dragAndDrop(source, target)`: Drags an element from source to target (requires Actions class).
14. `driver.moveToElement(element)`: Moves the mouse to the center of the specified element (requires Actions class).
15. `driver.clickAndHold()`: Clicks and holds the specified element (requires Actions class).
16. `driver.release()`: Releases the held element (requires Actions class).
17. `driver.perform()`: Executes all actions in the Actions class (requires Actions class).
18. `driver.sendKeys(Keys.ENTER)`: Sends the Enter key to the element.
19. `driver.sendKeys(Keys.TAB)`: Sends the Tab key to the element.
20. `driver.sendKeys(Keys.BACK_SPACE)`: Sends the Backspace key to the element.
21. `driver.sendKeys(Keys.ESCAPE)`: Sends the Escape key to the element.
22. `driver.sendKeys(Keys.CONTROL, "a")`: Sends Ctrl+A key combination to the element.
23. `driver.sendKeys(Keys.CONTROL, "c")`: Sends Ctrl+C key combination to the element.
24. `driver.sendKeys(Keys.CONTROL, "v")`: Sends Ctrl+V key combination to the element.
25. `driver.sendKeys(Keys.CONTROL, "x")`: Sends Ctrl+X key combination to the element.
26. `driver.executeScript("arguments[0].scrollIntoView(true);", element)`: Scrolls the element into view using JavaScript.
27. `driver.executeScript("arguments[0].click()", element)`: Clicks on the element using JavaScript.
28. `driver.executeScript("arguments[0].value='text';", element)`: Sets the value of an input field using JavaScript.
29. `driver.executeScript("arguments[0].style.border='2px solid red';", element)`: Highlights the element by changing its border color using JavaScript.

SELENIUM COMMANDS

30. `driver.executeScript("return arguments[0].innerHTML;", element)`: Retrieves the HTML content of the element using JavaScript.
31. `driver.executeScript("arguments[0].scrollBy(0, -100);", element)`: Scrolls the element up by 100 pixels using JavaScript.
32. `driver.executeScript("arguments[0].focus()", element)`: Focuses on the element using JavaScript.
33. `driver.executeScript("arguments[0].blur()", element)`: Removes focus from the element using JavaScript.
34. `driver.executeScript("arguments[0].select()", element)`: Selects the content of a text input field using JavaScript.
35. `driver.executeScript("arguments[0].setAttribute('attributeName', 'value')", element)`: Sets an attribute of the element using JavaScript.
36. `driver.executeScript("arguments[0].click()", element)`: Clicks the element using JavaScript.
37. `driver.sendKeys(Keys.PAGE_DOWN)`: Sends the Page Down key to scroll down the page.
38. `driver.sendKeys(Keys.PAGE_UP)`: Sends the Page Up key to scroll up the page.
39. `driver.sendKeys(Keys.HOME)`: Sends the Home key to move the cursor to the beginning of the line.
40. `driver.sendKeys(Keys.END)`: Sends the End key to move the cursor to the end of the line.
41. `driver.sendKeys(Keys.ARROW_DOWN)`: Sends the Down Arrow key to navigate down.
42. `driver.sendKeys(Keys.ARROW_UP)`: Sends the Up Arrow key to navigate up.
43. `driver.sendKeys(Keys.ARROW_LEFT)`: Sends the Left Arrow key to navigate left.
44. `driver.sendKeys(Keys.ARROW_RIGHT)`: Sends the Right Arrow key to navigate right.
45. `driver.findElement(By.xpath("xpath")).click()`: Clicks on an element located by XPath.
46. `driver.findElement(By.cssSelector("cssSelector")).sendKeys("text")`: Sends text to an element located by CSS selector.
47. `driver.findElement(By.id("elementId")).clear()`: Clears the text input field of an element located by ID.
48. `driver.findElement(By.name("elementName")).submit()`: Submits the form containing the element located by name.
49. `driver.findElement(By.className("className")).getAttribute("attributeName")`: Retrieves the value of the specified attribute from an element located by class name.
50. `driver.findElement(By.tagName("tagName")).getText()`: Gets the visible text from an element located by tag name.
51. `driver.findElement(By.linkText("linkText")).click()`: Clicks on a link element located by visible text.
52. `driver.findElement(By.partialLinkText("partialText")).click()`: Clicks on a link element located by partial text.
53. `driver.findElement(By.xpath("//input")).sendKeys("text")`: Sends text to an input element located by XPath.
54. `driver.findElement(By.cssSelector("input")).click()`: Clicks on an input element located by CSS selector.
55. `driver.findElement(By.xpath("//button")).click()`: Clicks on a button element located by XPath.
56. `driver.findElement(By.cssSelector("button")).click()`: Clicks on a button element located by CSS selector.

SELENIUM COMMANDS

Wait Commands

1. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10))`: Sets an implicit wait of 10 seconds for the WebDriver to wait for elements to be available.
2. `WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10))`: Creates a WebDriverWait instance with a timeout of 10 seconds.
3. `wait.until(ExpectedConditions.visibilityOf(element))`: Waits until the specified element is visible on the page.
4. `wait.until(ExpectedConditions.elementToBeClickable(element))`: Waits until the specified element is clickable.
5. `wait.until(ExpectedConditions.titleIs("Page Title"))`: Waits until the page title matches the specified title.
6. `wait.until(ExpectedConditions.urlContains("partOfUrl"))`: Waits until the current URL contains the specified substring.
7. `wait.until(ExpectedConditions.textToBePresentInElement(element, "text"))`: Waits until the specified text is present in the given element.
8. `wait.until(ExpectedConditions.elementToBeSelected(element))`: Waits until the specified element is selected.
9. `wait.until(ExpectedConditions.invisibilityOf(element))`: Waits until the specified element is not visible.
10. `wait.until(ExpectedConditions.stalenessOf(element))`: Waits until the specified element is no longer attached to the DOM.
11. `wait.until(ExpectedConditions.alertIsPresent())`: Waits until an alert is present on the page.
12. `wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(frameElement))`: Waits until the specified frame is available and switches to it.
13. `wait.until(ExpectedConditions.numberOfElementsToBeMoreThan(By.xpath("xpath"), 5))`: Waits until the number of elements located by the XPath is greater than 5.
14. `wait.until(ExpectedConditions.elementToBeClickable(By.id("elementId")))`: Waits until the element located by ID is clickable.
15. `wait.until(ExpectedConditions.visibilityOfElementLocated(By.className("className")))`: Waits until the element located by class name is visible.
16. `wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//tagName")))`: Waits until the element located by XPath is visible.
17. `wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("cssSelector")))`: Waits until the element located by CSS selector is visible.
18. `wait.until(ExpectedConditions.textToBePresentInElementLocated(By.name("elementName"), "text"))`: Waits until the specified text is present in the element located by name.
19. `wait.until(ExpectedConditions.textToBePresentInElementLocated(By.xpath("//element"), "text"))`: Waits until the specified text is present in the element located by XPath.
20. `wait.until(ExpectedConditions.attributeToBe(By.id("elementId"), "attributeName", "value"))`: Waits until the specified attribute of the element located by ID has the specified value.
21. `wait.until(ExpectedConditions.attributeToBe(By.cssSelector("cssSelector"), "attributeName", "value"))`: Waits until the specified attribute of the element located by CSS selector has the specified value.
22. `wait.until(ExpectedConditions.textToBe(By.className("className"), "text"))`: Waits until the text of the element located by class name matches the specified text.

SELENIUM COMMANDS

23. `wait.until(ExpectedConditions.titleContains("partialTitle"))`: Waits until the page title contains the specified partial title.
24. `wait.until(ExpectedConditions.urlToBe("http://example.com"))`: Waits until the current URL matches the specified URL.
25. `wait.until(ExpectedConditions.urlMatches("regexPattern"))`: Waits until the current URL matches the specified regex pattern.
26. `wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(By.id("frameId")))`: Waits until the frame located by ID is available and switches to it.
27. `wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(By.xpath("//iframe")))`: Waits until the frame located by XPath is available and switches to it.
28. `wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//button")))`: Waits until the button element located by XPath is clickable.
29. `wait.until(ExpectedConditions.invisibilityOfElementLocated(By.id("elementId")))`: Waits until the element located by ID is not visible.
30. `wait.until(ExpectedConditions.stalenessOfElementLocated(By.name("elementName")))`: Waits until the element located by name is no longer attached to the DOM.
31. `wait.until(ExpectedConditions.alertIsPresent())`: Waits until an alert is present on the page.
32. `wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector("button.submit")))`: Waits until the button element located by CSS selector is clickable.
33. `wait.until(ExpectedConditions.textToBePresentInElementLocated(By.className("message"), "Success"))`: Waits until the text 'Success' is present in the element located by class name.
34. `wait.until(ExpectedConditions.elementToBeClickable(By.className("submitButton")))`: Waits until the element located by class name is clickable.
35. `wait.until(ExpectedConditions.titleContains("Dashboard"))`: Waits until the page title contains 'Dashboard'.
36. `wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//input[@type='submit']")))`: Waits until the submit button element located by XPath is clickable.
37. `wait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Login")))`: Waits until the link with text 'Login' is visible.
38. `wait.until(ExpectedConditions.textToBePresentInElementLocated(By.xpath("//div[@id='message']", "Welcome")))`: Waits until the text 'Welcome' is present in the element located by XPath.
39. `wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//a[@href='logout']")))`: Waits until the link element with href 'logout' is clickable.
40. `wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("span.error")))`: Waits until the span element with class 'error' is visible.
41. `wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//button[@type='submit']")))`: Waits until the submit button element located by XPath is clickable.
42. `wait.until(ExpectedConditions.textToBePresentInElementLocated(By.cssSelector("div.alert"), "Error"))`: Waits until the text 'Error' is present in the element located by CSS selector.
43. `wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//h1[text()='Welcome']")))`: Waits until the h1 element with text 'Welcome' is visible.
44. `wait.until(ExpectedConditions.urlToBe("https://example.com/home"))`: Waits until the current URL matches 'https://example.com/home'.
45. `wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("username")))`: Waits until the element with ID 'username' is visible.

SELENIUM COMMANDS

46. `wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//a[text()='Help']")));`
Waits until the link with text 'Help' is clickable.
47. `wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//input[@type='text']")));`
Waits until the text input field element located by XPath is clickable.
48. `wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("input[type='password']")));`: Waits until the password input field is visible.
49. `wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(By.xpath("//iframe[@name='frame']")));`: Waits until the iframe with name 'frame' is available and switches to it.
50. `wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//button[@id='submit']")));`: Waits until the button with ID 'submit' is clickable.
51. `wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("searchField")));`: Waits until the search field element located by name is visible.

Keyboard and Mouse Action Commands (Advanced User Interactions)

1. `Actions actions = new Actions(driver);`: Creates a new Actions instance for performing advanced user interactions.
2. `actions.moveToElement(element).perform();`: Moves the mouse to the center of the specified element and performs the action.
3. `actions.click(element).perform();`: Clicks on the specified element.
4. `actions.doubleClick(element).perform();`: Double-clicks on the specified element.
5. `actions.contextClick(element).perform();`: Performs a right-click on the specified element.
6. `actions.clickAndHold(element).perform();`: Clicks and holds the specified element.
7. `actions.release(element).perform();`: Releases the held element.
8. `actions.dragAndDrop(source, target).perform();`: Drags an element from the source to the target.
9. `actions.sendKeys(Keys.ENTER).perform();`: Sends the Enter key to the element.
10. `actions.sendKeys(Keys.TAB).perform();`: Sends the Tab key to the element.
11. `actions.sendKeys(Keys.BACK_SPACE).perform();`: Sends the Backspace key to the element.
12. `actions.sendKeys(Keys.ESCAPE).perform();`: Sends the Escape key to the element.
13. `actions.sendKeys(Keys.CONTROL, "a").perform();`: Sends Ctrl+A key combination to the element.
14. `actions.sendKeys(Keys.CONTROL, "c").perform();`: Sends Ctrl+C key combination to the element.
15. `actions.sendKeys(Keys.CONTROL, "v").perform();`: Sends Ctrl+V key combination to the element.
16. `actions.sendKeys(Keys.CONTROL, "x").perform();`: Sends Ctrl+X key combination to the element.
17. `actions.moveToElement(element, offsetX, offsetY).perform();`: Moves the mouse to the specified offset from the center of the element.
18. `actions.keyDown(Keys.SHIFT).sendKeys("text").keyUp(Keys.SHIFT).perform();`: Sends text with Shift key held down.
19. `actions.keyDown(Keys.ALT).sendKeys("text").keyUp(Keys.ALT).perform();`: Sends text with Alt key held down.
20. `actions.keyDown(Keys.CONTROL).sendKeys("text").keyUp(Keys.CONTROL).perform();`: Sends text with Ctrl key held down.
21. `actions.pause(Duration.ofSeconds(2)).perform();`: Pauses the execution for 2 seconds.

SELENIUM COMMANDS

22. `actions.moveByOffset(100, 200).perform()`: Moves the mouse by an offset of 100 pixels horizontally and 200 pixels vertically.
23. `actions.moveToElement(element).click().perform()`: Moves to the element and clicks on it.
24. `actions.doubleClick().perform()`: Double-clicks at the current mouse position.
25. `actions.contextClick().perform()`: Right-clicks at the current mouse position.
26. `actions.clickAndHold().perform()`: Clicks and holds at the current mouse position.
27. `actions.dragAndDropBy(source, 100, 200).perform()`: Drags an element from the source to a position offset by 100 pixels horizontally and 200 pixels vertically.
28. `actions.scrollByAmount(0, 100).perform()`: Scrolls vertically by 100 pixels.
29. `actions.scrollByAmount(100, 0).perform()`: Scrolls horizontally by 100 pixels.
30. `actions.sendKeys(Keys.PAGE_DOWN).perform()`: Sends the Page Down key to scroll down the page.
31. `actions.sendKeys(Keys.PAGE_UP).perform()`: Sends the Page Up key to scroll up the page.
32. `actions.sendKeys(Keys.HOME).perform()`: Sends the Home key to move the cursor to the start of the line.
33. `actions.sendKeys(Keys.END).perform()`: Sends the End key to move the cursor to the end of the line.
34. `actions.sendKeys(Keys.ARROW_DOWN).perform()`: Sends the Down Arrow key to navigate down.
35. `actions.sendKeys(Keys.ARROW_UP).perform()`: Sends the Up Arrow key to navigate up.
36. `actions.sendKeys(Keys.ARROW_LEFT).perform()`: Sends the Left Arrow key to navigate left.
37. `actions.sendKeys(Keys.ARROW_RIGHT).perform()`: Sends the Right Arrow key to navigate right.
38. `actions.moveToElement(element).keyDown(Keys.CONTROL).click().keyUp(Keys.CONTROL).perform()`: Moves to the element, holds Ctrl key, clicks on the element, and then releases Ctrl key.
39. `actions.moveToElement(element).keyDown(Keys.SHIFT).click().keyUp(Keys.SHIFT).perform()`: Moves to the element, holds Shift key, clicks on the element, and then releases Shift key.
40. `actions.moveToElement(element).clickAndHold().moveByOffset(50, 50).release().perform()`: Clicks and holds an element, moves by an offset, and then releases.
41. `actions.sendKeys(Keys.F5).perform()`: Sends the F5 key to refresh the page.
42. `actions.keyDown(Keys.CONTROL).sendKeys("t").keyUp(Keys.CONTROL).perform()`: Sends Ctrl+T key combination to open a new tab.
43. `actions.keyDown(Keys.CONTROL).sendKeys("w").keyUp(Keys.CONTROL).perform()`: Sends Ctrl+W key combination to close the current tab.
44. `actions.keyDown(Keys.CONTROL).sendKeys("n").keyUp(Keys.CONTROL).perform()`: Sends Ctrl+N key combination to open a new window.
45. `actions.keyDown(Keys.CONTROL).sendKeys("p").keyUp(Keys.CONTROL).perform()`: Sends Ctrl+P key combination to open the print dialog.
46. `actions.keyDown(Keys.CONTROL).sendKeys("s").keyUp(Keys.CONTROL).perform()`: Sends Ctrl+S key combination to open the save dialog.
47. `actions.keyDown(Keys.ALT).sendKeys("f4").keyUp(Keys.ALT).perform()`: Sends Alt+F4 key combination to close the current window.
48. `actions.sendKeys(Keys.INSERT).perform()`: Sends the Insert key to toggle between insert and overwrite mode.
49. `actions.sendKeys(Keys.DELETE).perform()`: Sends the Delete key to remove the next character.

SELENIUM COMMANDS

50. `actions.sendKeys(Keys.BACK_SPACE).perform()`: Sends the Backspace key to remove the previous character.
51. `actions.moveToElement(element).click().sendKeys("text").perform()`: Moves to the element, clicks on it, and sends text.
52. `actions.moveToElement(element).click().sendKeys(Keys.TAB).perform()`: Moves to the element, clicks on it, and sends the Tab key to move focus.
53. `actions.moveToElement(element).sendKeys(Keys.ENTER).perform()`: Moves to the element and sends the Enter key.
54. `actions.moveToElement(element).sendKeys(Keys.ESCAPE).perform()`: Moves to the element and sends the Escape key.

Cookies Management Commands

1. `driver.manage().getCookies()`: Retrieves all cookies visible to the current page.
2. `driver.manage().getCookieNamed("cookieName")`: Retrieves a specific cookie by its name.
3. `driver.manage().addCookie(new Cookie("name", "value"))`: Adds a new cookie with the specified name and value.
4. `driver.manage().deleteCookieNamed("cookieName")`: Deletes a specific cookie by its name.
5. `driver.manage().deleteCookie(cookie)`: Deletes the specified cookie object.
6. `driver.manage().deleteAllCookies()`: Deletes all cookies associated with the current domain.
7. `driver.manage().getCookieNamed("cookieName").getName()`: Retrieves the name of a specific cookie.
8. `driver.manage().getCookieNamed("cookieName").getValue()`: Retrieves the value of a specific cookie.
9. `driver.manage().getCookieNamed("cookieName").getDomain()`: Retrieves the domain of a specific cookie.
10. `driver.manage().getCookieNamed("cookieName").getPath()`: Retrieves the path of a specific cookie.
11. `driver.manage().getCookieNamed("cookieName").getExpiry()`: Retrieves the expiry date of a specific cookie.
12. `driver.manage().getCookies().size()`: Retrieves the number of cookies visible to the current page.
13. `driver.manage().getCookies().contains(new Cookie("name", "value"))`: Checks if a specific cookie exists.
14. `driver.manage().getCookies().stream().filter(cookie -> "name".equals(cookie.getName())).findFirst()`: Finds a specific cookie by name.
15. `driver.manage().getCookies().forEach(cookie -> System.out.println(cookie.getName() + ":" + cookie.getValue()))`: Prints all cookies and their values.
16. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date()))`: Adds a new cookie with domain, path, and expiry date.
17. `driver.manage().addCookie(new Cookie("name", "value", null, null, new Date()))`: Adds a new cookie with only name, value, and expiry date.
18. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path"))`: Adds a new cookie with domain and path.
19. `driver.manage().addCookie(new Cookie("name", "value", null, "/path"))`: Adds a new cookie with path only.

SELENIUM COMMANDS

20. `driver.manage().addCookie(new Cookie("name", "value", "domain", null, new Date()))`: Adds a new cookie with domain and expiry date.
21. `driver.manage().addCookie(new Cookie("name", "value", null, null, new Date()))`: Adds a new cookie with expiry date only.
22. `driver.manage().addCookie(new Cookie("name", "value", "domain"))`: Adds a new cookie with domain only.
23. `driver.manage().addCookie(new Cookie("name", "value", "/path"))`: Adds a new cookie with path only.
24. `driver.manage().addCookie(new Cookie("name", "value"))`: Adds a new cookie with default domain and path.
25. `driver.manage().getCookieNamed("cookieName").toString()`: Converts a specific cookie to a string representation.
26. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), true, true))`: Adds a secure and HTTP-only cookie.
27. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), false, false))`: Adds a non-secure and non-HTTP-only cookie.
28. `driver.manage().deleteCookie(new Cookie("name", "value"))`: Deletes a cookie based on its name and value.
29. `driver.manage().deleteAllCookies()`: Deletes all cookies for the current domain.
30. `driver.manage().getCookies().stream().filter(cookie -> cookie.getExpiry() != null).collect(Collectors.toList())`: Retrieves all cookies with expiry dates.
31. `driver.manage().getCookies().stream().filter(cookie -> cookie.getDomain().equals("domain")).collect(Collectors.toList())`: Retrieves all cookies for a specific domain.
32. `driver.manage().getCookies().stream().filter(cookie -> cookie.getPath().equals("/path")).collect(Collectors.toList())`: Retrieves all cookies for a specific path.
33. `driver.manage().getCookieNamed("cookieName").isHttpOnly()`: Checks if a specific cookie is HTTP-only.
34. `driver.manage().getCookieNamed("cookieName").isSecure()`: Checks if a specific cookie is secure.
35. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), true))`: Adds a secure cookie with expiry date.
36. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", null, false))`: Adds a non-secure cookie without expiry date.
37. `driver.manage().getCookies().stream().filter(cookie -> "name".equals(cookie.getName())).forEach(cookie -> System.out.println(cookie.getValue()))`: Retrieves and prints the value of a specific cookie.
38. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), false, true))`: Adds a non-secure, HTTP-only cookie.
39. `driver.manage().getCookies().stream().filter(cookie -> cookie.getName().startsWith("prefix")).collect(Collectors.toList())`: Retrieves all cookies with names starting with a specific prefix.
40. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), true, false))`: Adds a secure, non-HTTP-only cookie.
41. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), false, true))`: Adds a non-secure, HTTP-only cookie.

SELENIUM COMMANDS

42. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", null, true, true))`: Adds a secure, HTTP-only cookie with no expiry date.
43. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), true, false))`: Adds a secure, non-HTTP-only cookie with expiry date.
44. `driver.manage().getCookies().stream().map(cookie -> cookie.getName() + "=" + cookie.getValue()).collect(Collectors.joining("; "))`: Creates a string representation of all cookies in a format suitable for sending in HTTP headers.

Screen Capture Commands

1. `driver.getScreenshotAs(OutputType.FILE)`: Captures a screenshot and returns it as a File object.
2. `File screenshot = driver.getScreenshotAs(OutputType.FILE)`: Captures a screenshot and stores it in a File variable.
3. `FileUtils.copyFile(screenshot, new File("path/to/screenshot.png"))`: Copies the screenshot file to a specified location.
4. `driver.getScreenshotAs(OutputType.BYTES)`: Captures a screenshot and returns it as a byte array.
5. `byte[] screenshotBytes = driver.getScreenshotAs(OutputType.BYTES)`: Captures a screenshot and stores it as a byte array.
6. `BufferedImage image = ImageIO.read(new File("path/to/screenshot.png"))`: Reads the screenshot file into a BufferedImage.
7. `ImageIO.write(image, "png", new File("path/to/screenshot.png"))`: Writes the BufferedImage to a file in PNG format.
8. `driver.manage().window().getSize()`: Retrieves the size of the current window for use in custom screenshot implementations.
9. `driver.manage().window().setSize(new Dimension(1024, 768))`: Resizes the current window to the specified dimensions before taking a screenshot.
10. `File screenshot = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE)`: Captures a screenshot from a WebDriver instance cast to TakesScreenshot.
11. `((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE)`: Directly captures a screenshot from a TakesScreenshot instance.
12. `FileUtils.copyFile(((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE), new File("path/to/screenshot.png"))`: Copies the screenshot file to a specified location.
13. `driver.getScreenshotAs(OutputType.FILE)`: Captures a screenshot of the current page and returns it as a File object.
14. `File screenshot = driver.getScreenshotAs(OutputType.FILE)`: Captures a screenshot and stores it as a File object.
15. `BufferedImage bufferedImage = ImageIO.read(screenshot)`: Converts the screenshot file to a BufferedImage.
16. `ImageIO.write(bufferedImage, "png", new File("path/to/screenshot.png"))`: Saves the BufferedImage as a PNG file.
17. `driver.getScreenshotAs(OutputType.BASE64)`: Captures a screenshot and returns it as a Base64-encoded string.
18. `String screenshotBase64 = driver.getScreenshotAs(OutputType.BASE64)`: Captures a screenshot and stores it as a Base64-encoded string.

SELENIUM COMMANDS

19. `byte[] screenshotBytes = ((TakesScreenshot)driver).getScreenshotAs(OutputType.BYTES);`
Captures a screenshot and stores it as a byte array.
20. `driver.getScreenshotAs(OutputType.BYTES);`: Retrieves the screenshot as a byte array for custom handling.
21. `FileUtils.copyFile(((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE), new File("path/to/screenshot.png"));`: Saves the screenshot to a specified file.
22. `BufferedImage image = ImageIO.read(new File("path/to/screenshot.png"));`: Reads the screenshot file into a BufferedImage for further processing.
23. `ImageIO.write(image, "png", new File("path/to/screenshot.png"));`: Writes the BufferedImage to a PNG file for storage.
24. `driver.getScreenshotAs(OutputType.FILE);`: Takes a screenshot and returns it as a file object for saving.
25. `FileUtils.copyFile(screenshotFile, new File("path/to/screenshot.png"));`: Copies the screenshot file to a specific path.
26. `BufferedImage screenshotImage = ImageIO.read(new File("path/to/screenshot.png"));`: Loads the screenshot image for manipulation or validation.
27. `ImageIO.write(screenshotImage, "png", new File("path/to/screenshot.png"));`: Saves the loaded screenshot image to a PNG file.
28. `driver.getScreenshotAs(OutputType.BYTES);`: Captures a screenshot as a byte array for further use in testing.
29. `byte[] screenshotData = driver.getScreenshotAs(OutputType.BYTES);`: Retrieves the screenshot data as a byte array.
30. `driver.getScreenshotAs(OutputType.BASE64);`: Gets the screenshot as a Base64-encoded string for web-based applications.
31. `String screenshotBase64String = driver.getScreenshotAs(OutputType.BASE64);`: Captures and stores the screenshot as a Base64-encoded string.
32. `driver.getScreenshotAs(OutputType.FILE);`: Takes a screenshot and stores it in a File object for saving or processing.
33. `FileUtils.copyFile(((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE), new File("path/to/screenshot.png"));`: Saves the screenshot to a file location.
34. `BufferedImage screenshotBufferedImage = ImageIO.read(new File("path/to/screenshot.png"));`: Converts the screenshot file to a BufferedImage for image processing.
35. `ImageIO.write(screenshotBufferedImage, "png", new File("path/to/screenshot.png"));`: Saves the BufferedImage to a PNG file.
36. `driver.getScreenshotAs(OutputType.FILE);`: Takes a screenshot and returns a File object containing the screenshot.
37. `File screenshot = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);`: Captures the screenshot as a File object for further use.
38. `BufferedImage bufferedImage = ImageIO.read(new File("path/to/screenshot.png"));`: Reads the screenshot image into a BufferedImage object.
39. `ImageIO.write(bufferedImage, "png", new File("path/to/screenshot.png"));`: Writes the BufferedImage to a file in PNG format.
40. `driver.getScreenshotAs(OutputType.BYTES);`: Retrieves the screenshot in byte array format for custom handling.
41. `byte[] screenshotBytes = ((TakesScreenshot)driver).getScreenshotAs(OutputType.BYTES);`: Gets the screenshot as a byte array.

SELENIUM COMMANDS

42. `driver.getScreenshotAs(OutputType.BASE64)`: Gets the screenshot as a Base64 string for transmission.
43. `String screenshotBase64 = driver.getScreenshotAs(OutputType.BASE64)`: Retrieves the screenshot as a Base64-encoded string.
44. `driver.getScreenshotAs(OutputType.FILE)`: Captures a screenshot and returns it as a File object.
45. `FileUtils.copyFile(((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE), new File("path/to/screenshot.png"))`: Saves the screenshot to a specified file path.
46. `BufferedImage bufferedImage = ImageIO.read(new File("path/to/screenshot.png"))`: Converts the screenshot file to a BufferedImage.
47. `ImageIO.write(bufferedImage, "png", new File("path/to/screenshot.png"))`: Saves the BufferedImage as a PNG file.
48. `driver.getScreenshotAs(OutputType.BYTES)`: Retrieves the screenshot as a byte array for processing.
49. `byte[] screenshotByteArray = driver.getScreenshotAs(OutputType.BYTES)`: Gets the screenshot in byte array format.
50. `driver.getScreenshotAs(OutputType.BASE64)`: Captures the screenshot as a Base64 string.
51. `String screenshotBase64String = driver.getScreenshotAs(OutputType.BASE64)`: Retrieves the screenshot as a Base64-encoded string.
52. `driver.getScreenshotAs(OutputType.FILE)`: Takes a screenshot and returns it as a File object for saving.
53. `FileUtils.copyFile(((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE), new File("path/to/screenshot.png"))`: Copies and saves the screenshot file.
54. `BufferedImage image = ImageIO.read(new File("path/to/screenshot.png"))`: Reads the screenshot image into a BufferedImage for manipulation.
55. `ImageIO.write(image, "png", new File("path/to/screenshot.png"))`: Writes the BufferedImage to a PNG file.

Session Management Commands

1. `driver.manage().getCookies()`: Retrieves all cookies visible to the current page, which can be used for session management.
2. `driver.manage().getCookieNamed("cookieName")`: Retrieves a specific cookie by its name, useful for checking session data.
3. `driver.manage().addCookie(new Cookie("name", "value"))`: Adds a new cookie, which can be used to manage session data.
4. `driver.manage().deleteCookieNamed("cookieName")`: Deletes a specific cookie, useful for ending a session.
5. `driver.manage().deleteCookie(cookie)`: Deletes a specific cookie object from the session.
6. `driver.manage().deleteAllCookies()`: Deletes all cookies associated with the current domain, effectively ending the session.
7. `driver.manage().getCookieNamed("cookieName").getName()`: Retrieves the name of a specific cookie for session management.
8. `driver.manage().getCookieNamed("cookieName").getValue()`: Retrieves the value of a specific cookie, useful for session validation.
9. `driver.manage().getCookieNamed("cookieName").getDomain()`: Retrieves the domain of a specific cookie, important for session scope.

SELENIUM COMMANDS

10. `driver.manage().getCookieNamed("cookieName").getPath()`: Retrieves the path of a specific cookie, affecting session management.
11. `driver.manage().getCookieNamed("cookieName").getExpiry()`: Retrieves the expiry date of a specific cookie, which can affect session duration.
12. `driver.manage().getCookies().size()`: Retrieves the number of cookies visible to the current page, which can indicate active sessions.
13. `driver.manage().getCookies().contains(new Cookie("name", "value"))`: Checks if a specific cookie exists, useful for verifying active sessions.
14. `driver.manage().getCookies().stream().filter(cookie -> "name".equals(cookie.getName())).findFirst()`: Finds a specific cookie by name for session validation.
15. `driver.manage().getCookies().forEach(cookie -> System.out.println(cookie.getName() + ": " + cookie.getValue()))`: Prints all cookies, useful for debugging sessions.
16. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date()))`: Adds a new cookie with domain, path, and expiry date, managing session attributes.
17. `driver.manage().addCookie(new Cookie("name", "value", null, null, new Date()))`: Adds a new cookie with only name, value, and expiry date, affecting session duration.
18. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path"))`: Adds a new cookie with domain and path, which can be used for session management.
19. `driver.manage().addCookie(new Cookie("name", "value", null, "/path"))`: Adds a new cookie with path only, affecting session scope.
20. `driver.manage().addCookie(new Cookie("name", "value", "domain", null, new Date()))`: Adds a new cookie with domain and expiry date, which affects session management.
21. `driver.manage().addCookie(new Cookie("name", "value", null, null, new Date()))`: Adds a new cookie with expiry date only, affecting session duration.
22. `driver.manage().addCookie(new Cookie("name", "value", "domain"))`: Adds a new cookie with domain only, used for session management.
23. `driver.manage().addCookie(new Cookie("name", "value", "/path"))`: Adds a new cookie with path only, affecting session scope.
24. `driver.manage().addCookie(new Cookie("name", "value"))`: Adds a new cookie with default domain and path, which can affect the session.
25. `driver.manage().getCookieNamed("cookieName").toString()`: Converts a specific cookie to a string representation, useful for debugging sessions.
26. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), true, true))`: Adds a secure and HTTP-only cookie for session management.
27. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), false, false))`: Adds a non-secure and non-HTTP-only cookie, affecting session security.
28. `driver.manage().deleteCookie(new Cookie("name", "value"))`: Deletes a cookie based on its name and value, ending a session.
29. `driver.manage().deleteAllCookies()`: Deletes all cookies for the current domain, effectively ending all sessions.
30. `driver.manage().getCookies().stream().filter(cookie -> cookie.getExpiry() != null).collect(Collectors.toList())`: Retrieves all cookies with expiry dates, affecting session duration.
31. `driver.manage().getCookies().stream().filter(cookie -> cookie.getDomain().equals("domain")).collect(Collectors.toList())`: Retrieves all cookies for a specific domain, useful for managing session scope.

SELENIUM COMMANDS

32. `driver.manage().getCookies().stream().filter(cookie -> cookie.getPath().equals("/path")).collect(Collectors.toList())`: Retrieves all cookies for a specific path, useful for managing session scope.
33. `driver.manage().getCookieNamed("cookieName").isHttpOnly()`: Checks if a specific cookie is HTTP-only, affecting session security.
34. `driver.manage().getCookieNamed("cookieName").isSecure()`: Checks if a specific cookie is secure, important for session security.
35. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), true))`: Adds a secure cookie with expiry date, managing session attributes.
36. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", null, false))`: Adds a non-secure cookie without expiry date, affecting session management.
37. `driver.manage().getCookies().stream().filter(cookie -> "name".equals(cookie.getName())).forEach(cookie -> System.out.println(cookie.getValue()))`: Retrieves and prints the value of a specific cookie for session validation.
38. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), false, true))`: Adds a non-secure, HTTP-only cookie, managing session security.
39. `driver.manage().getCookies().stream().filter(cookie -> cookie.getName().startsWith("prefix")).collect(Collectors.toList())`: Retrieves all cookies with names starting with a specific prefix, useful for session management.
40. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), true, false))`: Adds a secure, non-HTTP-only cookie with expiry date, managing session security.
41. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), false, true))`: Adds a non-secure, HTTP-only cookie with expiry date, managing session attributes.
42. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", null, true, true))`: Adds a secure, HTTP-only cookie with no expiry date, affecting session management.
43. `driver.manage().addCookie(new Cookie("name", "value", "domain", "/path", new Date(), true, false))`: Adds a secure, non-HTTP-only cookie with expiry date, managing session attributes.
44. `driver.manage().getCookies().stream().map(cookie -> cookie.getName() + "=" + cookie.getValue()).collect(Collectors.joining("; "))`: Creates a string representation of all cookies, useful for session management in HTTP headers.

Browser Options and Capabilities Commands

1. `driver.manage().window().setSize(new Dimension(width, height))`: Sets the browser window size, useful for testing different screen resolutions.
2. `driver.manage().window().setPosition(new Point(x, y))`: Sets the position of the browser window on the screen.
3. `driver.manage().window().fullscreen()`: Maximizes the browser window to full screen.
4. `driver.manage().window().maximize()`: Maximizes the browser window to fit the screen.
5. `driver.manage().window().minimize()`: Minimizes the browser window.
6. `driver.manage().window().getSize()`: Retrieves the current size of the browser window.
7. `driver.manage().window().getPosition()`: Retrieves the current position of the browser window.

SELENIUM COMMANDS

8. `driver.manage().window().getRect()`: Retrieves the size and position of the browser window.
9. `driver.manage().window().setRect(new Rectangle(x, y, width, height))`: Sets the size and position of the browser window.
10. `ChromeOptions options = new ChromeOptions()`: Initializes ChromeOptions for configuring Chrome browser settings.
11. `options.addArguments("start-maximized")`: Adds an argument to start the Chrome browser maximized.
12. `options.addArguments("disable-popup-blocking")`: Disables popup blocking in the Chrome browser.
13. `options.addArguments("incognito")`: Launches Chrome in incognito mode.
14. `options.addArguments("headless")`: Runs Chrome in headless mode (without GUI).
15. `options.addArguments("disable-extensions")`: Disables all extensions in the Chrome browser.
16. `options.addArguments("disable-gpu")`: Disables GPU hardware acceleration in Chrome.
17. `options.addArguments("no-sandbox")`: Runs Chrome without sandboxing for testing purposes.
18. `options.addArguments("remote-debugging-port=9222")`: Enables remote debugging on port 9222 for Chrome.
19. `options.setBinary("/path/to/chrome")`: Sets the path to a specific Chrome binary.
20. `FirefoxOptions options = new FirefoxOptions()`: Initializes FirefoxOptions for configuring Firefox browser settings.
21. `options.addArguments("-headless")`: Runs Firefox in headless mode (without GUI).
22. `options.addArguments("-private")`: Launches Firefox in private browsing mode.
23. `options.addArguments("-disable-popup-blocking")`: Disables popup blocking in Firefox.
24. `options.addArguments("-disable-extensions")`: Disables all extensions in Firefox.
25. `options.addArguments("-no-sandbox")`: Runs Firefox without sandboxing for testing purposes.
26. `options.setBinary("/path/to/firefox")`: Sets the path to a specific Firefox binary.
27. `DesiredCapabilities capabilities = new DesiredCapabilities()`: Initializes DesiredCapabilities for configuring browser settings.
28. `capabilities.setCapability("browserName", "chrome")`: Sets the browser name to Chrome in DesiredCapabilities.
29. `capabilities.setCapability("browserVersion", "89.0")`: Sets the browser version in DesiredCapabilities.
30. `capabilities.setCapability("platformName", "Windows 10")`: Sets the platform (OS) in DesiredCapabilities.
31. `capabilities.setCapability("acceptInsecureCerts", true)`: Accepts insecure SSL certificates in DesiredCapabilities.
32. `capabilities.setCapability("javascriptEnabled", true)`: Enables JavaScript in the browser.
33. `capabilities.setCapability("pageLoadStrategy", "eager")`: Sets the page load strategy to eager (waits until the page is interactive).
34. `capabilities.setCapability("timeouts", new HashMap<String, Object>() {{ put("implicit", 5000); put("pageLoad", 60000); }})`: Configures timeouts for implicit waits and page loads.
35. `options.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true)`: Accepts SSL certificates in browser options.

SELENIUM COMMANDS

36. `options.setCapability(CapabilityType.UNEXPECTED_ALERT_BEHAVIOUR, UnexpectedAlertBehaviour.IGNORE)`: Ignores unexpected alerts in the browser.
37. `options.setCapability("proxy", proxy)`: Configures a proxy server for browser options.
38. `capabilities.setCapability("maxInstances", 5)`: Sets the maximum number of browser instances to be run concurrently.
39. `capabilities.setCapability("browserName", "firefox")`: Sets the browser name to Firefox in DesiredCapabilities.
40. `capabilities.setCapability("browserVersion", "90.0")`: Sets the browser version to Firefox 90.0 in DesiredCapabilities.
41. `capabilities.setCapability("platformName", "macOS")`: Sets the platform (OS) to macOS in DesiredCapabilities.
42. `options.setCapability("browserName", "MicrosoftEdge")`: Sets the browser name to Microsoft Edge.
43. `options.setCapability("browserVersion", "latest")`: Sets the browser version to the latest version available.
44. `options.setCapability("platformName", "Linux")`: Sets the platform (OS) to Linux in DesiredCapabilities.
45. `options.addArguments("start-maximized")`: Configures the browser to start maximized by default.
46. `options.addArguments("disable-popup-blocking")`: Disables the popup blocking feature in the browser.
47. `options.addArguments("disable-gpu")`: Disables GPU hardware acceleration in the browser.
48. `options.addArguments("headless")`: Runs the browser in headless mode for testing without a GUI.
49. `options.setCapability("acceptInsecureCerts", true)`: Configures the browser to accept insecure SSL certificates.
50. `options.setCapability("proxy", proxy)`: Sets up a proxy for the browser.
51. `capabilities.setCapability("enableVNC", true)`: Enables VNC (Virtual Network Computing) for remote debugging.
52. `capabilities.setCapability("enableVideo", true)`: Enables video recording of the test session.

Logs Management Commands

1. `driver.manage().logs().get("browser")`: Retrieves browser logs, which can help in debugging browser-related issues.
2. `driver.manage().logs().get("driver")`: Retrieves driver logs, useful for debugging issues related to the WebDriver.
3. `driver.manage().logs().get("client")`: Retrieves client logs, providing information about the client-side of the browser session.
4. `driver.manage().logs().get("performance")`: Retrieves performance logs, which can be used to analyze browser performance.
5. `driver.manage().logs().get("server")`: Retrieves server logs, useful for analyzing server-side interactions.
6. `driver.manage().logs().get("logType")`: Retrieves logs for a specific type, where "logType" can be "browser", "driver", "client", "performance", or "server".
7. `driver.manage().logs().get("browser").getAll()`: Retrieves all browser logs from the current session.

SELENIUM COMMANDS

8. `driver.manage().logs().get("browser").get(0)`: Retrieves the first entry from the browser logs.
9. `driver.manage().logs().get("browser").get(1)`: Retrieves the second entry from the browser logs.
10. `driver.manage().logs().get("browser").get(2)`: Retrieves the third entry from the browser logs.
11. `driver.manage().logs().get("browser").get(3)`: Retrieves the fourth entry from the browser logs.
12. `driver.manage().logs().get("browser").get(4)`: Retrieves the fifth entry from the browser logs.
13. `driver.manage().logs().get("driver").getAll()`: Retrieves all driver logs from the current session.
14. `driver.manage().logs().get("driver").get(0)`: Retrieves the first entry from the driver logs.
15. `driver.manage().logs().get("driver").get(1)`: Retrieves the second entry from the driver logs.
16. `driver.manage().logs().get("driver").get(2)`: Retrieves the third entry from the driver logs.
17. `driver.manage().logs().get("driver").get(3)`: Retrieves the fourth entry from the driver logs.
18. `driver.manage().logs().get("driver").get(4)`: Retrieves the fifth entry from the driver logs.
19. `driver.manage().logs().get("client").getAll()`: Retrieves all client logs from the current session.
20. `driver.manage().logs().get("client").get(0)`: Retrieves the first entry from the client logs.
21. `driver.manage().logs().get("client").get(1)`: Retrieves the second entry from the client logs.
22. `driver.manage().logs().get("client").get(2)`: Retrieves the third entry from the client logs.
23. `driver.manage().logs().get("client").get(3)`: Retrieves the fourth entry from the client logs.
24. `driver.manage().logs().get("client").get(4)`: Retrieves the fifth entry from the client logs.
25. `driver.manage().logs().get("performance").getAll()`: Retrieves all performance logs from the current session.
26. `driver.manage().logs().get("performance").get(0)`: Retrieves the first entry from the performance logs.
27. `driver.manage().logs().get("performance").get(1)`: Retrieves the second entry from the performance logs.
28. `driver.manage().logs().get("performance").get(2)`: Retrieves the third entry from the performance logs.
29. `driver.manage().logs().get("performance").get(3)`: Retrieves the fourth entry from the performance logs.
30. `driver.manage().logs().get("performance").get(4)`: Retrieves the fifth entry from the performance logs.
31. `driver.manage().logs().get("server").getAll()`: Retrieves all server logs from the current session.
32. `driver.manage().logs().get("server").get(0)`: Retrieves the first entry from the server logs.
33. `driver.manage().logs().get("server").get(1)`: Retrieves the second entry from the server logs.
34. `driver.manage().logs().get("server").get(2)`: Retrieves the third entry from the server logs.
35. `driver.manage().logs().get("server").get(3)`: Retrieves the fourth entry from the server logs.
36. `driver.manage().logs().get("server").get(4)`: Retrieves the fifth entry from the server logs.
37. `driver.manage().logs().get("browser").get(0).getMessage()`: Retrieves the message of the first browser log entry.
38. `driver.manage().logs().get("browser").get(0).getTimestamp()`: Retrieves the timestamp of the first browser log entry.

SELENIUM COMMANDS

39. `driver.manage().logs().get("browser").get(0).getLevel()`: Retrieves the log level of the first browser log entry.
40. `driver.manage().logs().get("driver").get(0).getMessage()`: Retrieves the message of the first driver log entry.
41. `driver.manage().logs().get("driver").get(0).getTimestamp()`: Retrieves the timestamp of the first driver log entry.
42. `driver.manage().logs().get("driver").get(0).getLevel()`: Retrieves the log level of the first driver log entry.
43. `driver.manage().logs().get("client").get(0).getMessage()`: Retrieves the message of the first client log entry.
44. `driver.manage().logs().get("client").get(0).getTimestamp()`: Retrieves the timestamp of the first client log entry.
45. `driver.manage().logs().get("client").get(0).getLevel()`: Retrieves the log level of the first client log entry.
46. `driver.manage().logs().get("performance").get(0).getMessage()`: Retrieves the message of the first performance log entry.
47. `driver.manage().logs().get("performance").get(0).getTimestamp()`: Retrieves the timestamp of the first performance log entry.
48. `driver.manage().logs().get("performance").get(0).getLevel()`: Retrieves the log level of the first performance log entry.
49. `driver.manage().logs().get("server").get(0).getMessage()`: Retrieves the message of the first server log entry.
50. `driver.manage().logs().get("server").get(0).getTimestamp()`: Retrieves the timestamp of the first server log entry.
51. `driver.manage().logs().get("server").get(0).getLevel()`: Retrieves the log level of the first server log entry.
52. `driver.manage().logs().get("browser").get(0).get("message")`: Retrieves the message of the first browser log entry in a specific format.
53. `driver.manage().logs().get("driver").get(0).get("message")`: Retrieves the message of the first driver log entry in a specific format.
54. `driver.manage().logs().get("client").get(0).get("message")`: Retrieves the message of the first client log entry in a specific format.
55. `driver.manage().logs().get("performance").get(0).get("message")`: Retrieves the message of the first performance log entry in a specific format.
56. `driver.manage().logs().get("server").get(0).get("message")`: Retrieves the message of the first server log entry in a specific format.

JavaScript Execution Commands

1. `driver.executeScript("return document.title")`: Executes JavaScript to return the title of the current page.
2. `driver.executeScript("window.scrollTo(0, document.body.scrollHeight)")`: Scrolls to the bottom of the page.
3. `driver.executeScript("window.scrollTo(0, 0)")`: Scrolls to the top of the page.
4. `driver.executeScript("arguments[0].click()", element)`: Clicks on the specified element using JavaScript.

SELENIUM COMMANDS

5. `driver.executeScript("return arguments[0].innerHTML", element)`: Returns the inner HTML of the specified element.
6. `driver.executeScript("return arguments[0].value", element)`: Returns the value of the specified form element.
7. `driver.executeScript("arguments[0].style.backgroundColor = 'yellow'", element)`: Changes the background color of the specified element to yellow.
8. `driver.executeScript("return window.location.href")`: Returns the current URL of the page.
9. `driver.executeScript("window.location.href = 'http://example.com'")`: Navigates to the specified URL.
10. `driver.executeScript("return document.readyState")`: Returns the current state of the document (e.g., 'loading', 'interactive', 'complete').
11. `driver.executeScript("return document.querySelector('selector').textContent")`: Returns the text content of the specified element.
12. `driver.executeScript("document.querySelector('selector').value = 'new value'")`: Sets the value of the specified input element.
13. `driver.executeScript("document.querySelector('selector').setAttribute('attribute', 'value')")`: Sets an attribute of the specified element.
14. `driver.executeScript("return document.querySelector('selector').getAttribute('attribute')")`: Retrieves the value of an attribute of the specified element.
15. `driver.executeScript("document.querySelector('selector').focus()")`: Focuses on the specified element.
16. `driver.executeScript("return window.getComputedStyle(document.querySelector('selector')).getPropertyValue('property')")`: Returns the computed style property value of the specified element.
17. `driver.executeScript("document.querySelector('selector').blur()")`: Removes focus from the specified element.
18. `driver.executeScript("return document.querySelectorAll('selector').length")`: Returns the number of elements matching the selector.
19. `driver.executeScript("document.querySelector('selector').scrollIntoView()")`: Scrolls the specified element into view.
20. `driver.executeScript("document.querySelector('selector').remove()")`: Removes the specified element from the DOM.
21. `driver.executeScript("return document.querySelector('selector').classList")`: Returns the class list of the specified element.
22. `driver.executeScript("document.querySelector('selector').classList.add('class-name')")`: Adds a class to the specified element.
23. `driver.executeScript("document.querySelector('selector').classList.remove('class-name')")`: Removes a class from the specified element.
24. `driver.executeScript("return document.querySelector('selector').checked")`: Returns whether the specified checkbox or radio button is checked.
25. `driver.executeScript("document.querySelector('selector').click()")`: Clicks on the specified element using JavaScript.
26. `driver.executeScript("return document.querySelector('selector').dataset.attribute")`: Returns the value of a data attribute of the specified element.
27. `driver.executeScript("document.querySelector('selector').setAttribute('data-attribute', 'value')")`: Sets a data attribute of the specified element.

SELENIUM COMMANDS

28. `driver.executeScript("return document.querySelector('selector').innerHTML")`: Returns the inner HTML of the specified element.
29. `driver.executeScript("document.querySelector('selector').innerHTML = 'new content'")`: Sets the inner HTML of the specified element.
30. `driver.executeScript("return window.performance.timing")`: Returns performance timing information of the current page.
31. `driver.executeScript("return window.localStorage.getItem('key')")`: Retrieves a value from local storage.
32. `driver.executeScript("window.localStorage.setItem('key', 'value')")`: Sets a value in local storage.
33. `driver.executeScript("return window.sessionStorage.getItem('key')")`: Retrieves a value from session storage.
34. `driver.executeScript("window.sessionStorage.setItem('key', 'value')")`: Sets a value in session storage.
35. `driver.executeScript("return document.querySelector('selector').scrollHeight")`: Returns the scroll height of the specified element.
36. `driver.executeScript("return document.querySelector('selector').offsetWidth")`: Returns the offset width of the specified element.
37. `driver.executeScript("return document.querySelector('selector').offsetHeight")`: Returns the offset height of the specified element.
38. `driver.executeScript("return document.querySelector('selector').offsetTop")`: Returns the offset top of the specified element.
39. `driver.executeScript("return document.querySelector('selector').offsetLeft")`: Returns the offset left of the specified element.
40. `driver.executeScript("document.querySelector('selector').setAttribute('style', 'property:value')")`: Sets a CSS style property of the specified element.
41. `driver.executeScript("document.querySelector('selector').dispatchEvent(new Event('event-type'))")`: Dispatches a custom event on the specified element.
42. `driver.executeScript("return document.querySelector('selector').dataset")`: Returns all data attributes of the specified element.
43. `driver.executeScript("return document.querySelector('selector').hasAttribute('attribute')")`: Checks if the specified element has the given attribute.
44. `driver.executeScript("document.querySelector('selector').setAttribute('attribute', 'value')")`: Sets an attribute of the specified element.
45. `driver.executeScript("return document.querySelector('selector').childNodes.length")`: Returns the number of child nodes of the specified element.
46. `driver.executeScript("return document.querySelector('selector').parentNode")`: Returns the parent node of the specified element.
47. `driver.executeScript("return document.querySelector('selector').nextElementSibling")`: Returns the next sibling element of the specified element.
48. `driver.executeScript("return document.querySelector('selector').previousElementSibling")`: Returns the previous sibling element of the specified element.
49. `driver.executeScript("return window.getComputedStyle(document.querySelector('selector')).visibility")`: Returns the visibility style property value of the specified element.

SELENIUM COMMANDS

50. `driver.executeScript("return window.getComputedStyle(document.querySelector('selector')).display")`: Returns the display style property value of the specified element.

File Upload and Download Commands

File Upload Commands

1. `driver.findElement(By.cssSelector("input[type='file']")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with type 'file'.
2. `driver.findElement(By.id("file-upload")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific ID.
3. `driver.findElement(By.name("file")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific name.
4. `driver.findElement(By.xpath("//input[@type='file']")).sendKeys("/path/to/file")`: Uploads a file using XPath to locate the file input element.
5. `driver.findElement(By.cssSelector("input.upload")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to a file input element with a specific class.
6. `driver.findElement(By.cssSelector("input[type='file']")).sendKeys("/absolute/path/to/file")`: Uploads a file using the absolute path to the file input element.
7. `driver.findElement(By.id("uploadButton")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an element used for file uploads.
8. `driver.findElement(By.xpath("//input[@name='fileUpload']")).sendKeys("/path/to/file")`: Uploads a file using XPath to locate the file input element by its name attribute.
9. `driver.findElement(By.cssSelector("input#file-upload")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific CSS ID.
10. `driver.findElement(By.className("file-input")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific class name.
11. `driver.findElement(By.xpath("//input[@id='file-upload']")).sendKeys("/path/to/file")`: Uploads a file using XPath to locate the file input element by its ID.
12. `driver.findElement(By.cssSelector("input.file")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific class name.
13. `driver.findElement(By.name("uploadFile")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific name.
14. `driver.findElement(By.id("fileUpload")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific ID.
15. `driver.findElement(By.xpath("//input[contains(@id, 'upload')]")).sendKeys("/path/to/file")`: Uploads a file using XPath to locate the file input element by a partial ID match.
16. `driver.findElement(By.cssSelector("input[type='file'][name='fileUpload']")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with both type and name attributes.
17. `driver.findElement(By.cssSelector("input#upload")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific CSS ID.
18. `driver.findElement(By.name("fileUpload")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific name.
19. `driver.findElement(By.className("upload-file")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific class.

SELENIUM COMMANDS

20. `driver.findElement(By.xpath("//input[@type='file' and @name='file']")).sendKeys("/path/to/file")`: Uploads a file using XPath to locate the file input element by type and name attributes.
21. `driver.findElement(By.cssSelector("input[type='file'][id='fileUpload']")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with both type and ID attributes.
22. `driver.findElement(By.id("file-upload")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific ID.
23. `driver.findElement(By.xpath("//input[@type='file'][@id='fileUpload']")).sendKeys("/path/to/file")`: Uploads a file using XPath to locate the file input element by type and ID attributes.
24. `driver.findElement(By.name("fileUpload")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific name.
25. `driver.findElement(By.className("file-upload")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific class.
26. `driver.findElement(By.cssSelector("input.upload")).sendKeys("/path/to/file")`: Uploads a file by specifying its path to an input element with a specific class.
27. `driver.findElement(By.xpath("//input[@type='file']")).sendKeys("/path/to/file")`: Uploads a file using XPath to locate the file input element.

File Download Commands

1. `driver.get("http://example.com/file-download-url")`: Navigates to the URL for downloading a file.
2. `driver.findElement(By.linkText("Download File")).click()`: Clicks a link to download a file based on the link text.
3. `driver.findElement(By.cssSelector("a.download")).click()`: Clicks a download link based on its CSS class.
4. `driver.findElement(By.xpath("//a[text()='Download']")).click()`: Clicks a download link using XPath based on its visible text.
5. `driver.findElement(By.id("download")).click()`: Clicks a download button or link based on its ID.
6. `driver.findElement(By.name("downloadFile")).click()`: Clicks a download button or link based on its name attribute.
7. `driver.findElement(By.className("download-button")).click()`: Clicks a download button or link based on its class name.
8. `driver.findElement(By.xpath("//a[contains(@href, 'download')]")).click()`: Clicks a download link using XPath based on a partial href attribute.
9. `driver.findElement(By.cssSelector("a[href*='download']")).click()`: Clicks a download link based on a partial href attribute using CSS selector.
10. `driver.findElement(By.xpath("//button[@id='download']")).click()`: Clicks a download button based on its ID.
11. `driver.findElement(By.cssSelector("button.download")).click()`: Clicks a download button based on its CSS class.
12. `driver.findElement(By.linkText("Download PDF")).click()`: Clicks a download link based on the link text for PDF files.
13. `driver.findElement(By.xpath("//a[@href='/files/sample.pdf']")).click()`: Clicks a download link for a specific file using XPath based on its href attribute.

SELENIUM COMMANDS

14. `driver.findElement(By.cssSelector("a[href='/files/sample.docx']")).click()`: Clicks a download link for a specific file using CSS selector based on its href attribute.
15. `driver.findElement(By.xpath("//a[contains(text(), 'Sample Download')]")).click()`: Clicks a download link using XPath based on partial text.
16. `driver.findElement(By.id("downloadFileButton")).click()`: Clicks a download button based on its ID.
17. `driver.findElement(By.xpath("//a[@class='file-download']")).click()`: Clicks a download link using XPath based on its class attribute.
18. `driver.findElement(By.linkText("File Download")).click()`: Clicks a download link based on the link text.
19. `driver.findElement(By.cssSelector("a.download-link")).click()`: Clicks a download link based on its CSS class.
20. `driver.findElement(By.xpath("//button[text()='Download']]")).click()`: Clicks a download button using XPath based on its visible text.
21. `driver.findElement(By.cssSelector("button.file-download")).click()`: Clicks a download button based on its CSS class.
22. `driver.findElement(By.id("fileDownload")).click()`: Clicks a download button based on its ID.
23. `driver.findElement(By.name("fileDownload")).click()`: Clicks a download button based on its name attribute.
24. `driver.findElement(By.xpath("//a[@data-action='download']]")).click()`: Clicks a download link using XPath based on a custom data attribute.

Proxy Management Commands

1. `driver.manage().addCookie(new Cookie("name", "value"))`: Adds a new cookie to the browser session.
2. `driver.manage().deleteCookieNamed("name")`: Deletes a specific cookie by its name.
3. `driver.manage().deleteAllCookies()`: Deletes all cookies for the current domain.
4. `driver.manage().getCookieNamed("name")`: Retrieves a specific cookie by its name.
5. `driver.manage().getCookies()`: Retrieves all cookies for the current domain.
6. `driver.manage().window().setSize(new Dimension(width, height))`: Sets the size of the browser window.
7. `driver.manage().window().setPosition(new Point(x, y))`: Sets the position of the browser window on the screen.
8. `driver.manage().window().maximize()`: Maximizes the browser window.
9. `driver.manage().window().fullscreen()`: Makes the browser window fullscreen.
10. `driver.manage().window().minimize()`: Minimizes the browser window.
11. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets the default wait time for locating elements.
12. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Sets the maximum time to wait for a page to load.
13. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Sets the timeout for asynchronous scripts.
14. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets the implicit wait time for elements.
15. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Sets the maximum wait time for a page to load.

SELENIUM COMMANDS

16. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Sets the maximum time for asynchronous scripts.
17. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets the implicit wait time for locating elements.
18. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Sets the timeout for waiting for a page to load.
19. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Sets the maximum time for scripts to execute.
20. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets the implicit wait time for elements.
21. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Sets the maximum time to wait for a page to load.
22. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Sets the maximum time to wait for scripts to execute.
23. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets the default wait time for element location.
24. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Configures the timeout for waiting for a page to load.
25. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Configures the maximum time for asynchronous script execution.
26. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Configures implicit waits for element interactions.
27. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Configures maximum wait time for complete page loading.
28. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Configures script timeout duration.
29. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Configures default waiting time for elements.
30. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Sets the maximum time allowed for a page to fully load.
31. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Sets the maximum allowed time for scripts to run.
32. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Configures wait time for locating elements.
33. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Sets timeout for the page to load completely.
34. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Sets timeout for executing scripts.
35. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets wait time for locating elements.
36. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Configures maximum wait time for a page to load.
37. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Configures timeout for script execution.
38. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Configures implicit waits for element interactions.

SELENIUM COMMANDS

39. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Configures timeout for complete page loading.
40. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Configures maximum time for script execution.
41. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Configures wait time for locating elements.
42. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Configures maximum wait time for a page load.
43. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Configures timeout for executing scripts.
44. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets default wait time for elements.
45. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Configures timeout for waiting for page to load.
46. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Configures maximum time for script execution.
47. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets implicit wait time for elements.
48. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Sets the timeout for complete page loading.
49. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Configures maximum time for script execution.

Mobile Web Testing Commands (Specific to Appium)

1. `driver.findElement(By.id("element_id")).click()`: Clicks on a web element identified by its ID.
2. `driver.findElement(By.name("element_name")).sendKeys("text")`: Sends text to a web element identified by its name.
3. `driver.findElement(By.xpath("//xpath")).click()`: Clicks on a web element using XPath.
4. `driver.findElement(By.cssSelector("selector")).getText()`: Retrieves the text from a web element using CSS selector.
5. `driver.findElement(By.className("class_name")).isDisplayed()`: Checks if a web element is displayed.
6. `driver.findElement(By.linkText("link_text")).click()`: Clicks on a link identified by its text.
7. `driver.findElement(By.partialLinkText("partial_link_text")).click()`: Clicks on a link identified by partial text.
8. `driver.findElement(By.tagName("tag_name")).sendKeys("text")`: Sends text to a web element identified by its tag name.
9. `driver.findElement(By.id("element_id")).submit()`: Submits a form element identified by its ID.
10. `driver.findElement(By.name("element_name")).clear()`: Clears the text from a web element identified by its name.
11. `driver.findElement(By.xpath("//xpath")).getAttribute("attribute_name")`: Retrieves the value of an attribute from a web element using XPath.
12. `driver.findElement(By.cssSelector("selector")).getAttribute("attribute_name")`: Retrieves the value of an attribute from a web element using CSS selector.

SELENIUM COMMANDS

13. `driver.findElement(By.className("class_name")).click()`: Clicks on a web element identified by its class name.
14. `driver.findElement(By.xpath("//xpath")).sendKeys("text")`: Sends text to a web element using XPath.
15. `driver.findElement(By.name("element_name")).isEnabled()`: Checks if a web element identified by its name is enabled.
16. `driver.findElement(By.id("element_id")).getText()`: Retrieves the text from a web element identified by its ID.
17. `driver.findElement(By.linkText("link_text")).getText()`: Retrieves the text from a link identified by its text.
18. `driver.findElement(By.xpath("//xpath")).getSize()`: Retrieves the size of a web element using XPath.
19. `driver.findElement(By.cssSelector("selector")).getSize()`: Retrieves the size of a web element using CSS selector.
20. `driver.findElement(By.className("class_name")).getLocation()`: Retrieves the location of a web element identified by its class name.
21. `driver.findElement(By.id("element_id")).getLocation()`: Retrieves the location of a web element identified by its ID.
22. `driver.findElement(By.xpath("//xpath")).isSelected()`: Checks if a web element is selected using XPath.
23. `driver.findElement(By.cssSelector("selector")).isSelected()`: Checks if a web element is selected using CSS selector.
24. `driver.findElement(By.className("class_name")).getCssValue("property_name")`: Retrieves the value of a CSS property from a web element identified by its class name.
25. `driver.findElement(By.name("element_name")).getCssValue("property_name")`: Retrieves the value of a CSS property from a web element identified by its name.
26. `driver.findElement(By.id("element_id")).findElement(By.xpath("//child_xpath")).click()`: Clicks on a child element of a web element identified by its ID.
27. `driver.findElement(By.cssSelector("selector")).findElement(By.className("child_class_name")).click()`: Clicks on a child element of a web element identified by its CSS selector.
28. `driver.findElement(By.xpath("//xpath")).findElement(By.name("child_name")).click()`: Clicks on a child element of a web element identified by its XPath.
29. `driver.findElement(By.className("class_name")).findElement(By.linkText("child_link_text")).click()`: Clicks on a child element of a web element identified by its class name.
30. `driver.findElement(By.id("element_id")).findElement(By.tagName("tag_name")).sendKeys("text")`: Sends text to a child element of a web element identified by its ID.
31. `driver.findElement(By.xpath("//xpath")).findElement(By.cssSelector("child_selector")).getText()`: Retrieves the text from a child element using XPath.
32. `driver.findElement(By.cssSelector("selector")).findElement(By.xpath("//child_xpath")).getText()`: Retrieves the text from a child element using CSS selector.
33. `driver.findElement(By.className("class_name")).findElement(By.xpath("//child_xpath")).getAttribute("attribute_name")`: Retrieves the value of an attribute from a child element using class name.
34. `driver.findElement(By.id("element_id")).findElement(By.linkText("child_link_text")).getCssValue("property_name")`: Retrieves the value of a CSS property from a child element using ID.

SELENIUM COMMANDS

35. `driver.findElement(By.xpath("//xpath")).findElement(By.name("child_name")).getLocation()`: Retrieves the location of a child element using XPath.
36. `driver.findElement(By.cssSelector("selector")).findElement(By.className("child_class_name")).isDisplayed()`: Checks if a child element is displayed using CSS selector.
37. `driver.findElement(By.className("class_name")).findElement(By.cssSelector("child_selector")).isEnabled()`: Checks if a child element is enabled using class name.
38. `driver.findElement(By.id("element_id")).findElement(By.xpath("//child_xpath")).isSelected()`: Checks if a child element is selected using ID.
39. `driver.findElement(By.xpath("//xpath")).findElement(By.linkText("child_link_text")).clear()`: Clears the text from a child element using XPath.
40. `driver.findElement(By.cssSelector("selector")).findElement(By.className("child_class_name")).submit()`: Submits a child element using CSS selector.
41. `driver.findElement(By.className("class_name")).findElement(By.tagName("tag_name")).sendKeys("text")`: Sends text to a child element using class name.
42. `driver.findElement(By.xpath("//xpath")).findElement(By.cssSelector("child_selector")).getAttribute("attribute_name")`: Retrieves the value of an attribute from a child element using XPath.
43. `driver.findElement(By.cssSelector("selector")).findElement(By.tagName("tag_name")).getSize()`: Retrieves the size of a child element using CSS selector.
44. `driver.findElement(By.className("class_name")).findElement(By.xpath("//child_xpath")).getCssValue("property_name")`: Retrieves the value of a CSS property from a child element using class name.
45. `driver.findElement(By.id("element_id")).findElement(By.className("child_class_name")).getText()`: Retrieves the text from a child element using ID.
46. `driver.findElement(By.xpath("//xpath")).findElement(By.tagName("tag_name")).isDisplayed()`: Checks if a child element is displayed using XPath.
47. `driver.findElement(By.cssSelector("selector")).findElement(By.name("child_name")).getText()`: Retrieves the text from a child element using CSS selector.
48. `driver.findElement(By.className("class_name")).findElement(By.cssSelector("child_selector")).getAttribute("attribute_name")`: Retrieves the value of an attribute from a child element using class name.
49. `driver.findElement(By.id("element_id")).findElement(By.xpath("//child_xpath")).getSize()`: Retrieves the size of a child element using ID.
50. `driver.findElement(By.xpath("//xpath")).findElement(By.linkText("child_link_text")).isEnabled()`: Checks if a child element is enabled using XPath.
51. `driver.findElement(By.cssSelector("selector")).findElement(By.name("child_name")).getSize()`: Retrieves the size of a child element using CSS selector.
52. `driver.findElement(By.className("class_name")).findElement(By.xpath("//child_xpath")).isSelected()`: Checks if a child element is selected using class name.
53. `driver.findElement(By.id("element_id")).findElement(By.className("child_class_name")).getCssValue("property_name")`: Retrieves the value of a CSS property from a child element using ID.

SELENIUM COMMANDS

Remote WebDriver Commands

1. `driver.get("URL")`: Navigates to the specified URL in the current browser window.
2. `driver.navigate().to("URL")`: Navigates to a new URL.
3. `driver.navigate().back()`: Navigates back to the previous page in the browser history.
4. `driver.navigate().forward()`: Navigates forward to the next page in the browser history.
5. `driver.navigate().refresh()`: Refreshes the current page.
6. `driver.quit()`: Closes all browser windows and ends the WebDriver session.
7. `driver.close()`: Closes the current browser window.
8. `driver.manage().window().maximize()`: Maximizes the current browser window.
9. `driver.manage().window().minimize()`: Minimizes the current browser window.
10. `driver.manage().window().fullscreen()`: Switches the browser window to fullscreen mode.
11. `driver.manage().window().setSize(new Dimension(width, height))`: Sets the size of the browser window.
12. `driver.manage().window().setPosition(new Point(x, y))`: Sets the position of the browser window.
13. `driver.manage().addCookie(new Cookie("name", "value"))`: Adds a new cookie to the browser.
14. `driver.manage().deleteCookieNamed("name")`: Deletes a cookie by its name.
15. `driver.manage().deleteAllCookies()`: Deletes all cookies for the current domain.
16. `driver.manage().getCookieNamed("name")`: Retrieves a specific cookie by its name.
17. `driver.manage().getCookies()`: Retrieves all cookies for the current domain.
18. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets the implicit wait time for locating elements.
19. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Sets the maximum time to wait for a page to load.
20. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Sets the timeout for asynchronous scripts.
21. `driver.findElement(By.id("element_id")).click()`: Clicks on an element identified by its ID.
22. `driver.findElement(By.name("element_name")).sendKeys("text")`: Sends text to an element identified by its name.
23. `driver.findElement(By.xpath("//xpath")).getText()`: Retrieves the text from an element using XPath.
24. `driver.findElement(By.cssSelector("selector")).getAttribute("attribute_name")`: Retrieves an attribute's value from an element using CSS selector.
25. `driver.findElement(By.className("class_name")).isDisplayed()`: Checks if an element identified by its class name is displayed.
26. `driver.findElement(By.linkText("link_text")).click()`: Clicks on a link identified by its text.
27. `driver.findElement(By.partialLinkText("partial_link_text")).click()`: Clicks on a link identified by partial text.
28. `driver.findElement(By.tagName("tag_name")).sendKeys("text")`: Sends text to an element identified by its tag name.
29. `driver.findElement(By.id("element_id")).submit()`: Submits a form element identified by its ID.
30. `driver.findElement(By.name("element_name")).clear()`: Clears the text from an element identified by its name.
31. `driver.switchTo().frame("frame_id")`: Switches to a specific iframe by its ID.

SELENIUM COMMANDS

32. `driver.switchTo().defaultContent()`: Switches back to the main content from an iframe.
33. `driver.switchTo().window("window_handle")`: Switches to a specific window by its handle.
34. `driver.switchTo().alert().accept()`: Accepts an alert popup.
35. `driver.switchTo().alert().dismiss()`: Dismisses an alert popup.
36. `driver.switchTo().alert().getText()`: Retrieves the text from an alert popup.
37. `driver.switchTo().alert().sendKeys("text")`: Sends text to an alert popup.
38. `driver.executeScript("script")`: Executes a JavaScript command.
39. `driver.executeAsyncScript("script")`: Executes an asynchronous JavaScript command.
40. `driver.manage().logs().get(LogType.BROWSER)`: Retrieves browser logs.
41. `driver.manage().logs().get(LogType.PERFORMANCE)`: Retrieves performance logs.
42. `driver.manage().logs().get(LogType.CLIENT)`: Retrieves client logs.
43. `driver.manage().logs().get(LogType.SERVER)`: Retrieves server logs.
44. `driver.manage().window().getSize()`: Retrieves the size of the browser window.
45. `driver.manage().window().getPosition()`: Retrieves the position of the browser window.
46. `driver.getTitle()`: Retrieves the title of the current page.
47. `driver.getCurrentUrl()`: Retrieves the URL of the current page.
48. `driver.getPageSource()`: Retrieves the source of the current page.
49. `driver.navigate().refresh()`: Refreshes the current page.
50. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Configures implicit wait time.
51. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Configures maximum wait time for page load.
52. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Configures script timeout duration.
53. `driver.manage().window().getSize()`: Gets the dimensions of the current window.
54. `driver.manage().window().getPosition()`: Gets the position of the current window.

WebDriver Timeout Commands

1. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets the implicit wait time for locating elements.
2. `driver.manage().timeouts().implicitlyWait(Duration.ofMinutes(minutes))`: Sets the implicit wait time for locating elements.
3. `driver.manage().timeouts().implicitlyWait(Duration.ofHours(hours))`: Sets the implicit wait time for locating elements.
4. `driver.manage().timeouts().implicitlyWait(Duration.ofDays(days))`: Sets the implicit wait time for locating elements.
5. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Sets the maximum time to wait for a page to load.
6. `driver.manage().timeouts().pageLoadTimeout(Duration.ofMinutes(minutes))`: Sets the maximum time to wait for a page to load.
7. `driver.manage().timeouts().pageLoadTimeout(Duration.ofHours(hours))`: Sets the maximum time to wait for a page to load.
8. `driver.manage().timeouts().pageLoadTimeout(Duration.ofDays(days))`: Sets the maximum time to wait for a page to load.
9. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Sets the timeout for asynchronous script execution.

SELENIUM COMMANDS

10. `driver.manage().timeouts().setScriptTimeout(Duration.ofMinutes(minutes))`: Sets the timeout for asynchronous script execution.
11. `driver.manage().timeouts().setScriptTimeout(Duration.ofHours(hours))`: Sets the timeout for asynchronous script execution.
12. `driver.manage().timeouts().setScriptTimeout(Duration.ofDays(days))`: Sets the timeout for asynchronous script execution.
13. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Configures implicit wait time for locating elements.
14. `driver.manage().timeouts().implicitlyWait(Duration.ofMinutes(minutes))`: Configures implicit wait time for locating elements.
15. `driver.manage().timeouts().implicitlyWait(Duration.ofHours(hours))`: Configures implicit wait time for locating elements.
16. `driver.manage().timeouts().implicitlyWait(Duration.ofDays(days))`: Configures implicit wait time for locating elements.
17. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Configures maximum wait time for page load.
18. `driver.manage().timeouts().pageLoadTimeout(Duration.ofMinutes(minutes))`: Configures maximum wait time for page load.
19. `driver.manage().timeouts().pageLoadTimeout(Duration.ofHours(hours))`: Configures maximum wait time for page load.
20. `driver.manage().timeouts().pageLoadTimeout(Duration.ofDays(days))`: Configures maximum wait time for page load.
21. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Configures script execution timeout.
22. `driver.manage().timeouts().setScriptTimeout(Duration.ofMinutes(minutes))`: Configures script execution timeout.
23. `driver.manage().timeouts().setScriptTimeout(Duration.ofHours(hours))`: Configures script execution timeout.
24. `driver.manage().timeouts().setScriptTimeout(Duration.ofDays(days))`: Configures script execution timeout.
25. `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds))`: Sets default implicit wait time for elements.
26. `driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(seconds))`: Sets default maximum wait time for page loading.
27. `driver.manage().timeouts().setScriptTimeout(Duration.ofSeconds(seconds))`: Sets default maximum timeout for script execution.
28. `driver.manage().timeouts().implicitlyWait(Duration.ofMinutes(minutes))`: Sets implicit wait time for element operations.
29. `driver.manage().timeouts().pageLoadTimeout(Duration.ofMinutes(minutes))`: Sets maximum wait time for page load.
30. `driver.manage().timeouts().setScriptTimeout(Duration.ofMinutes(minutes))`: Sets timeout for script execution.
31. `driver.manage().timeouts().implicitlyWait(Duration.ofHours(hours))`: Sets implicit wait time for locating elements.
32. `driver.manage().timeouts().pageLoadTimeout(Duration.ofHours(hours))`: Sets timeout for complete page load.

SELENIUM COMMANDS

33. `driver.manage().timeouts().setScriptTimeout(Duration.ofHours(hours))`: Sets timeout for script execution.
34. `driver.manage().timeouts().implicitlyWait(Duration.ofDays(days))`: Configures implicit wait duration.
35. `driver.manage().timeouts().pageLoadTimeout(Duration.ofDays(days))`: Configures maximum time for page load.
36. `driver.manage().timeouts().setScriptTimeout(Duration.ofDays(days))`: Configures script execution timeout.