# PLAYWRIGHT COMMANDS FOR AUTOMATION

**Browser Commands**

- chromium.launch([options]) – Launch Chromium browser
- firefox.launch([options]) – Launch Firefox browser
- webkit.launch([options]) – Launch WebKit browser
- browser.close() – Close the entire browser along with all contexts and pages
- browser.newContext([options]) – Create a new browser context
- browser.contexts() – List all open contexts
- browser.version() – Get the browser version
- browser.isConnected() – Check if the browser is connected
- browser.on('disconnected', callback) – Event triggered when the browser closes
- browser.on('targetcreated', callback) – Event triggered when a new page or context is created
- browser.on('targetdestroyed', callback) – Event triggered when a page or context is destroyed
- browser.startTracing() – Start browser-level tracing for debugging
- browser.stopTracing() – Stop browser-level tracing
- browser.launchPersistentContext(userDataDir, options) – Launch a browser with a persistent profile

---

# PLAYWRIGHT COMMANDS FOR AUTOMATION

**Context Commands**

- context.newPage() – Open a new page/tab in the context
- context.close() – Close the browser context
- context.cookies() – Get all cookies for the context
- context.addCookies(cookies) – Add cookies to the context
- context.clearCookies() – Clear all cookies from the context
- context.storageState([options]) – Save or load storage state
- context.setDefaultTimeout(ms) – Set default timeout for commands in the context
- context.setDefaultNavigationTimeout(ms) – Set default navigation timeout
- context.setOffline(true/false) – Simulate offline mode
- context.setGeolocation({ latitude, longitude }) – Set the geolocation for the context
- context.setPermissions([permissions]) – Grant permissions (camera, geolocation, etc.)
- context.tracing.start([options]) – Start tracing for all pages in the context
- context.tracing.stop({ path }) – Stop tracing and save to a file
- context.addInitScript(script) – Inject JS into every page in the context before scripts run
- context.exposeBinding(name, func) – Expose a Node.js function to the page
- context.exposeFunction(name, func) – Expose a Node.js function callable from page scripts

---

# PLAYWRIGHT COMMANDS FOR AUTOMATION

**Page / Navigation Commands**

- page.goto(url, [options]) – Navigate to a URL
- page.reload([options]) – Reload the current page
- page.goBack([options]) – Navigate back
- page.goForward([options]) – Navigate forward
- page.setViewportSize({ width, height }) – Set the viewport size
- page.content() – Get the full HTML content of the page
- page.title() – Get the page title
- page.url() – Get the current URL
- page.bringToFront() – Bring the page to the foreground
- page.close() – Close the page
- page.setContent(html, [options]) – Set the page content directly
- page.addInitScript(script) – Inject JavaScript into the page before any scripts run
- page.emulateMedia({ media, colorScheme }) – Emulate print, dark, or light mode
- page.pause() – Pause execution for debugging

**Page Events:**

- 'console' – Capture console messages
- 'pageerror' – Capture JavaScript errors
- 'request' / 'response' – Track network requests/responses
- 'dialog' – Listen for alert, confirm, prompt dialogs
- 'close' – Triggered when page closes
- 'framenavigated' – Triggered when frame navigation occurs

---

# PLAYWRIGHT COMMANDS FOR AUTOMATION

**Locator / Element Commands**

- page.locator(selector) – Create a locator for an element
- locator.click([options]) – Click the element
- locator.dblclick([options]) – Double-click the element
- locator.fill(value) – Fill an input field
- locator.type(text, [options]) – Type text into an element
- locator.press(key, [options]) – Press a key while focused on the element
- locator.hover([options]) – Hover over the element
- locator.check() – Check a checkbox
- locator.uncheck() – Uncheck a checkbox
- locator.selectOption(value) – Select an option from a dropdown
- locator.scrollIntoViewIfNeeded() – Scroll element into view
- locator.screenshot([options]) – Take a screenshot of the element
- locator.innerText() – Get the inner text of the element
- locator.textContent() – Get the text content of the element
- locator.getAttribute(name) – Get the value of an attribute
- locator.isVisible() – Check if the element is visible
- locator.isHidden() – Check if the element is hidden
- locator.isEnabled() – Check if the element is enabled
- locator.isDisabled() – Check if the element is disabled
- locator.nth(index) – Select the nth element in a set
- locator.first() – Select the first matching element
- locator.last() – Select the last matching element
- locator.boundingBox() – Get the element's position and size

---

# PLAYWRIGHT COMMANDS FOR AUTOMATION

**Keyboard Commands**

- page.keyboard.type(text, [options]) – Type text using the keyboard
- page.keyboard.press(key, [options]) – Press a key
- page.keyboard.down(key) – Key down
- page.keyboard.up(key) – Key up

---

**Mouse Commands**

- page.mouse.click(x, y, [options]) – Click at specific coordinates
- page.mouse.dblclick(x, y) – Double-click at coordinates
- page.mouse.move(x, y, [options]) – Move the mouse
- page.mouse.down([options]) – Mouse button down
- page.mouse.up([options]) – Mouse button up
- page.mouse.wheel(deltaX, deltaY) – Scroll with the mouse wheel

---

**Dialog / Alert Commands**

- page.on('dialog', callback) – Listen for dialogs
- dialog.accept([promptText]) – Accept a dialog
- dialog.dismiss() – Dismiss a dialog
- dialog.message() – Get the dialog message

---

# PLAYWRIGHT COMMANDS FOR AUTOMATION

**Network / Request Commands**

- page.route(urlOrPredicate, handler) – Intercept network requests
- page.unroute(urlOrPredicate) – Remove interception
- page.waitForResponse(urlOrPredicate) – Wait for a network response
- page.waitForRequest(urlOrPredicate) – Wait for a network request
- page.setExtraHTTPHeaders(headers) – Add custom headers to requests
- context.setHTTPCredentials({ username, password }) – Basic authentication
- context.setOffline(true/false) – Simulate offline
- page.routeFromHAR(harPath) – Replay network requests from HAR file

---

**Screenshot / Video / PDF Commands**

- page.screenshot([options]) – Take a screenshot of the page
- page.pdf([options]) – Export the page as PDF
- page.video().saveAs(path) – Save recorded video
- context.tracing.start([options]) – Start tracing for debugging
- context.tracing.stop({ path }) – Stop tracing and save

---

**Wait / Timeout Commands**

- page.waitForTimeout(ms) – Wait for a fixed time
- page.waitForLoadState([state]) – Wait for a page load state
- page.waitForSelector(selector, [options]) – Wait for an element to appear
- locator.waitFor([options]) – Wait for locator to satisfy conditions
- locator.waitForElementState('visible'|'hidden'|'enabled'|'disabled') – Wait for element state

---

# PLAYWRIGHT COMMANDS FOR AUTOMATION

**Evaluation / Assertion Commands**

- page.evaluate(fn) – Execute JavaScript in page context
- locator.evaluate(fn) – Execute JavaScript on an element
- expect(locator).toHaveText(text) – Assert element text
- expect(locator).toBeVisible() – Assert visibility
- expect(locator).toHaveValue(value) – Assert input value

---

**Storage / Cookies Commands**

- context.cookies() – Get cookies
- context.addCookies(cookies) – Add cookies
- context.clearCookies() – Clear all cookies
- context.storageState([options]) – Save/load storage state

---

**Advanced / Debugging Commands**

- page.pause() – Pause execution for debugging
- page.locator('selector').highlight() – Highlight element for debugging
- context.newCDPSession(page) – Access Chrome DevTools Protocol
- context.addInitScript(script) – Inject JS globally
- context.exposeBinding() / exposeFunction() – Expose Node.js functions
- context.tracing.start() / stop() – Record execution trace

---