Complete Playwright Functions Reference Guide

1. Browser Management Functions

Browser Launch & Control

- `playwright.chromium.launch()` - Launch Chromium browser
- `playwright.firefox.launch()` - Launch Firefox browser
- `playwright.webkit.launch()` - Launch WebKit browser
- `browser.close()` - Close browser instance
- `browser.newContext()` - Create new browser context
- `browser.newPage()` - Create new page directly
- `browser.contexts()` - Get all open contexts
- `browser.version()` - Get browser version
- `browser.isConnected()` - Check if browser is connected

Browser Context Functions

- `context.newPage()` - Create new page in context
- `context.close()` - Close browser context
- `context.pages()` - Get all pages in context
- `context.cookies()` - Get cookies
- `context.addCookies()` - Add cookies
- `context.clearCookies()` - Clear cookies
- `context.storageState()` - Get storage state
- `context.setGeolocation()` - Set geolocation
- `context.setOffline()` - Set offline mode
- `context.setExtraHTTPHeaders()` - Set HTTP headers
- `context.grantPermissions()` - Grant permissions
- `context.clearPermissions()` - Clear permissions
- `context.addInitScript()` - Add initialization script

2. Page Navigation Functions

Basic Navigation

- `page.goto(url)` - Navigate to URL
- `page.goBack()` - Go back in history
- `page.goForward()` - Go forward in history
- `page.reload()` - Reload current page
- `page.url()` - Get current URL
- `page.title()` - Get page title
- `page.close()` - Close page

Advanced Navigation

- `page.waitForNavigation()` - Wait for navigation
- `page.waitForURL()` - Wait for specific URL
- `page.waitForLoadState()` - Wait for load state
- `page.setDefaultNavigationTimeout()` - Set navigation timeout
- `page.setDefaultTimeout()` - Set default timeout

3. Element Locator Functions

Basic Locators

- `page.locator(selector)` - Generic locator
- `page.getByRole(role)` - Locate by ARIA role
- `page.getByText(text)` - Locate by text content
- `page.getByLabel(text)` - Locate by label
- `page.getByPlaceholder(text)` - Locate by placeholder
- `page.getByAltText(text)` - Locate by alt text
- `page.getByTitle(text)` - Locate by title attribute
- `page.getByTestId(testId)` - Locate by test ID
- `page.frameLocator(selector)` - Locate frame

Locator Chain Methods

- `locator.first()` - Get first matching element

- `locator.last()` - Get last matching element
- `locator.nth(index)` - Get nth element
- `locator.filter()` - Filter locator results
- `locator.locator()` - Chain locators
- `locator.and()` - Combine locators with AND
- `locator.or()` - Combine locators with OR

4. Element Interaction Functions

Click Actions

- `locator.click()` - Click element
- `locator.dblclick()` - Double click element
- `locator.hover()` - Hover over element
- `locator.rightClick()` - Right click element
- `locator.clickCount()` - Multiple clicks

Input Actions

- `locator.fill(value)` - Fill input field
- `locator.type(text)` - Type text slowly
- `locator.clear()` - Clear input field
- `locator.press(key)` - Press keyboard key
- `locator.pressSequentially(text)` - Type text sequentially

Form Interactions

- `locator.check()` - Check checkbox/radio
- `locator.uncheck()` - Uncheck checkbox
- `locator.selectOption()` - Select dropdown option
- `locator.setInputFiles()` - Upload files

Element State

- `locator.isVisible()` - Check if visible
- `locator.isHidden()` - Check if hidden
- `locator.isEnabled()` - Check if enabled

- `locator.isDisabled()` - Check if disabled
- `locator.isChecked()` - Check if checked
- `locator.isEditable()` - Check if editable

5. Wait Functions

Element Waits

- `locator.waitFor()` - Wait for element state
- `page.waitForSelector()` - Wait for selector
- `page.waitForFunction()` - Wait for JS function
- `page.waitForTimeout()` - Wait for timeout

Page State Waits

- `page.waitForLoadState()` - Wait for load state ('load', 'domcontentloaded', 'networkidle')
- `page.waitForEvent()` - Wait for page event
- `page.waitForRequest()` - Wait for network request
- `page.waitForResponse()` - Wait for network response

6. Data Extraction Functions

Element Content

- `locator.textContent()` - Get text content
- `locator.innerText()` - Get inner text
- `locator.innerHTML()` - Get inner HTML
- `locator.inputValue()` - Get input value
- `locator.getAttribute()` - Get attribute value
- `locator.count()` - Get element count
- `locator.allTextContents()` - Get all text contents
- `locator.allInnerTexts()` - Get all inner texts

Page Content

- `page.content()` - Get page HTML content
- `page.evaluate()` - Execute JavaScript on page
- `page.evaluateAll()` - Execute JS on all matching elements

- `page.$eval()` - Evaluate on single element
- `page.$$eval()` - Evaluate on multiple elements

7. Screenshot & PDF Functions

Screenshots

- `page.screenshot()` - Take page screenshot
- `locator.screenshot()` - Take element screenshot
- `page.pdf()` - Generate PDF (Chromium only)

Screenshot Options

- `fullPage: true` - Full page screenshot
- `clip: {x, y, width, height}` - Clipped screenshot
- `path: 'file.png'` - Save to file
- `type: 'png'|'jpeg'` - Image format

8. Network Functions

Request/Response Handling

- `page.route()` - Intercept and modify requests
- `page.unroute()` - Remove route handler
- `page.setExtraHTTPHeaders()` - Set HTTP headers
- `context.setExtraHTTPHeaders()` - Set context headers

Network Monitoring

- `page.on('request')` - Listen to requests
- `page.on('response')` - Listen to responses
- `page.on('requestfailed')` - Listen to failed requests
- `request.url()` - Get request URL
- `request.method()` - Get request method
- `request.headers()` - Get request headers
- `response.status()` - Get response status
- `response.json()` - Get response as JSON
- `response.text()` - Get response as text

9. Mobile & Device Emulation

Device Emulation

- `devices['iPhone 12']` - Predefined device configs
- `context.setViewportSize()` - Set viewport size
- `context.setUserAgent()` - Set user agent
- `context.setDeviceScaleFactor()` - Set device scale
- `context.setMobile()` - Enable mobile mode

Touch & Gestures

- `locator.tap()` - Tap element (mobile)
- `page.touchscreen.tap()` - Touch screen tap
- `page.mouse.click()` - Mouse click
- `page.keyboard.press()` - Keyboard press

10. Frame Handling Functions

Frame Navigation

- `page.frame()` - Get frame by name/URL
- `page.frames()` - Get all frames
- `page.mainFrame()` - Get main frame
- `frame.childFrames()` - Get child frames
- `frame.parentFrame()` - Get parent frame

Frame Operations

- `frameLocator.locator()` - Locate within frame
- `frame.goto()` - Navigate frame
- `frame.evaluate()` - Execute JS in frame
- `frame.content()` - Get frame content

11. Dialog & Alert Functions

Dialog Handling

- `page.on('dialog')` - Listen to dialogs

- `dialog.accept()` - Accept dialog
- `dialog.dismiss()` - Dismiss dialog
- `dialog.message()` - Get dialog message
- `dialog.type()` - Get dialog type
- `dialog.defaultValue()` - Get default input value

12. Storage Functions

Local Storage

- `page.evaluate(() => localStorage.getItem())` - Get local storage
- `page.evaluate(() => localStorage.setItem())` - Set local storage
- `page.evaluate(() => localStorage.clear())` - Clear local storage

Session Storage

- `page.evaluate(() => sessionStorage.getItem())` - Get session storage
- `page.evaluate(() => sessionStorage.setItem())` - Set session storage

Cookies & Storage State

- `context.storageState()` - Save storage state
- `browser.newContext({storageState})` - Load storage state

13. Video & Tracing Functions

Video Recording

- `recordVideo: {dir: 'videos/'}` - Enable video recording
- `page.video()` - Get video object
- `video.path()` - Get video file path
- `video.saveAs()` - Save video to file

Tracing

- `context.tracing.start()` - Start tracing
- `context.tracing.stop()` - Stop and save trace
- `context.tracing.startChunk()` - Start trace chunk
- `context.tracing.stopChunk()` - Stop trace chunk

14. Keyboard & Mouse Functions

Keyboard Actions

- `page.keyboard.press(key)` - Press single key
- `page.keyboard.down(key)` - Key down
- `page.keyboard.up(key)` - Key up
- `page.keyboard.type(text)` - Type text
- `page.keyboard.insertText(text)` - Insert text

Mouse Actions

- `page.mouse.click(x, y)` - Click at coordinates
- `page.mouse.move(x, y)` - Move mouse
- `page.mouse.down()` - Mouse button down
- `page.mouse.up()` - Mouse button up
- `page.mouse.wheel(deltaX, deltaY)` - Mouse wheel

15. Assertion Functions (expect)

Element Assertions

- `expect(locator).toBeVisible()` - Assert visible
- `expect(locator).toBeHidden()` - Assert hidden
- `expect(locator).toBeEnabled()` - Assert enabled
- `expect(locator).toBeDisabled()` - Assert disabled
- `expect(locator).toBeChecked()` - Assert checked
- `expect(locator).toBeEditable()` - Assert editable
- `expect(locator).toBeFocused()` - Assert focused

Content Assertions

- `expect(locator).toHaveText()` - Assert text content
- `expect(locator).toContainText()` - Assert contains text
- `expect(locator).toHaveValue()` - Assert input value
- `expect(locator).toHaveAttribute()` - Assert attribute
- `expect(locator).toHaveClass()` - Assert CSS class

- `expect(locator).toHaveCSS()` - Assert CSS property
- `expect(locator).toHaveCount()` - Assert element count

Page Assertions

- `expect(page).toHaveTitle()` - Assert page title
- `expect(page).toHaveURL()` - Assert page URL
- `expect(page).toHaveScreenshot()` - Assert screenshot match

16. Configuration Functions

Test Configuration

- `test.setTimeout()` - Set test timeout
- `test.slow()` - Mark test as slow
- `test.fixme()` - Mark test as broken
- `test.skip()` - Skip test
- `test.only()` - Run only this test

Setup & Teardown

- `test.beforeEach()` - Before each test
- `test.afterEach()` - After each test
- `test.beforeAll()` - Before all tests
- `test.afterAll()` - After all tests

Fixtures

- `test('test name', async ({ page }) => {})` - Page fixture
- `test('test name', async ({ context }) => {})` - Context fixture
- `test('test name', async ({ browser }) => {})` - Browser fixture

17. Advanced Utility Functions

Debugging

- `page.pause()` - Pause execution (debug mode)
- `page.setDefaultTimeout()` - Set timeout
- `page.getByRole().highlight()` - Highlight element

Performance

- `page.clock.install()` - Mock system time
- `page.clock.fastForward()` - Fast forward time
- `page.clock.setFixedTime()` - Set fixed time

API Testing

- `request.newContext()` - Create API context
- `request.get()` - GET request
- `request.post()` - POST request
- `request.put()` - PUT request
- `request.delete()` - DELETE request

Common Usage Patterns

Basic Test Structure

```javascript
import { test, expect } from '@playwright/test';

test('basic test', async ({ page }) => {
  await page.goto('https://example.com');
  await page.locator('input').fill('text');
  await page.locator('button').click();
  await expect(page.locator('result')).toBeVisible();
});
```

Multi-browser Testing

```javascript
import { test, devices } from '@playwright/test';

test.use(devices['iPhone 12']);
```

Custom Fixtures

```javascript
```

```
test.extend({
  customPage: async ({ browser }, use) => {
    const page = await browser.newPage();
    // Setup
    await use(page);
    // Teardown
  }
});
```

Key Points to Remember

1. **Locators are lazy** - They don't search until an action is performed
2. **Auto-waiting** - Most actions automatically wait for elements
3. **Strict mode** - Locators must match exactly one element
4. **Cross-browser** - Same API works across Chromium, Firefox, WebKit
5. **Async/Await** - All Playwright functions are asynchronous
6. **Test isolation** - Each test gets fresh browser context
7. **Built-in assertions** - Use `expect()` for web-specific assertions
8. **Debugging tools** - Use `--debug` flag and `page.pause()`