PLAYWRIGHT CHEAT SHEET

BY TESTHUSET

Browser Start and Close

`const { chromium } = require('playwright');`
`const browser = await chromium.launch();`
	Start a Chromium browser instance

`const { chromium } = require('playwright');`
`const browser = await chromium.launch({ channel: 'chrome' });`
	Start a Chrome browser instance

`const { firefox } = require('playwright');`
`const browser = await firefox.launch();`
	Start a Firefox browser instance

`await browser.close();`	Close the browser instance

Context Management

`const context = await browser.newContext();`	Create a new browser context
`await context.close();`	Close the browser context

Page / Tab Management

`const page = await context.newPage();`	Opens a new page / tab
`const pages = context.pages();`	List all pages
`await page.bringToFront();`	Makes the page / tab the active one
`await page.close();`	Close the current page / tab
`const isClosed = page.isClosed();`	Check if the page / tab is closed

Page Information

`const url = page.url();`	Get the current page URL
`const title = await page.title();`	Get the current page title

Navigation

`await page.goto('https://testhuset.dk);`	Navigates to a URL
`await page.reload();`	Reloads the current page
`await page.goBack();`	Navigates back in history
`await page.goForward();`	Navigates forward in history

Page Assertions

Requires `import { expect } from '@playwright/test'`

`await expect(page).toHaveURL('https://example.com/dashboard');`	Assert the URL of a page equals to a specific value

`await expect(page).toHaveURL(url => {`
`  return url.pathname === '/results'`
`    && url.searchParams.get('page') === '1';`
`});`
	Assert the URL of a page using a function

`await expect(page).toHaveTitle('Dashboard');`	Assert the title of a page equals a specific value

Element Selection

`const element = page.locator('#element');`	Select element using a CSS selector
`const element = page.getByText('Submit');`	Select element containing specific text
`const element = page.getByLabel('Username');`	Select a form element by label
`const element = page.getByRole('button', { name: ‘Submit’ });`	Select element by its ARIA role and name
`const element = page.getByPlaceholder('Enter your email');`	Select input by placeholder
`const element = page.getByAltText('Company Logo');`	Select image element by placeholder text
`const element = page.getByTitle('Close');`	Select element by its title attribute
`const element = page.getByTestId('submit-button');`	Select element by its data-testid attribute

Waiting for Element States

`await page.locator('#menu').waitFor({ state: 'attached' });`	Wait for an element to be present in the DOM
`await page.locator('#menu').waitFor({ state: 'visible' });`	Wait for an element to be visible on the page

`await page.locator('#menu').waitFor({`
`    state: 'visible',`
`    timeout: 30 * 60 * 1000 // 30 minutes`
`});`
	Wait for an element to be visible on the page with a custom timeout

`await page.locator('#menu').waitFor({ state: 'hidden' });`	Wait for an element to be hidden or removed from the page
`await page.locator('#menu').waitFor({ state: 'detached' });`	Wait for an element to be removed from the DOM.

Element Interactions

`await page.locator('#input').focus();`	Focus an element
`await page.locator('#input').blur();`	Remove focus from an element
`await page.locator('#element').scrollIntoViewIfNeeded();`	Scroll an element into view
`await page.locator('#element').selectText();`	Focuses an element and selects all its text content

Element State

`const text = await page.locator('#element').textContent();`	Get the text content of an element
`const text = await page.locator('#element').innerText();`	Get the inner text of an element
`const html = await page.locator('#element').innerHTML();`	Get the inner HTML of an element
`const html = await page.locator('#element').outerHTML();`	Get the outer HTML of an element
`const href = await page.locator('#element').getAttribute('href');`	Get value of a specific attribute of an element.
`const value = await page.locator('#input').inputValue();`	Get the value of an input element
`const isVisible = await page.locator('#element').isVisible();`	Check if element is visible on the page
`const isHidden = await page.locator('#element').isHidden();`	Check if element is hidden on the page
`const isEnabled = await page.locator('#element').isEnabled();`	Check if element is enabled
`const isDisabled = await page.locator('#element').isDisabled();`	Check if element is disabled
`const isChecked = await page.locator('#checkbox').isChecked();`	Check if checkbox or radio button is checked

Element Assertions

`await expect(page.locator('#element')).toBeAttached();`	Assert that element is attached to the DOM
`await expect(page.locator('#element')).toBeVisible();`	Assert that element is visible on page
`await expect(page.locator('#element')).toBeHidden();`	Assert that element is hidden on page
`await expect(page.locator('#element')).toContainText('Welcome Master Bruce');`	Assert that element contains specific text
`await expect(page.locator('#element')).not.toContainText('Error');`	Assert that element does not contain specific text
`await expect(page.locator('#input')).toHaveValue('Hello World');`	Assert that input has specific value
`await expect(page.locator('#multi-select')).toHaveValues([ 'red', 'green' ]);`	Assert multi-select to have specific selected values
`await expect(page.locator('#element')).toHaveClass("error");`	Assert that element contains specific CSS class
`await expect(page.locator('#element')).not.toHaveClass("error");`	Assert that element does not contain specific CSS class
`await expect(page.locator('#element')).toHaveCSS( 'display', 'block' );`	Assert that element has a specific CSS class
`await expect(page.locator('#element')).toHaveAttribute('alt-text');`	Assert that element has a specific attribute with a specific value
`await expect(page.locator('#checkbox')).toBeChecked();`	Assert that checkbox or radio button is checked
`await expect(page.locator('#checkbox')).not.toBeChecked();`	Assert that checkbox or radio button is not checked
`await expect(page.locator('#element')).toBeEnabled();`	Assert that an element is enabled
`await expect(page.locator('#element')).toBeDisabled();`	Assert that an element is disabled
`await expect(page.locator('#element')).toBeFocused();`	Assert that an element is focused

Element Click / Hover / Drag and Drop

`await page.locator('#button').click();`	Click on an element
`await page.locator('#button').click({ button: 'right' });`	Right click on an element
`await page.locator('#button').dblclick();`	Double click on an element
`await page.locator('#button').click({ modifiers: ['Control'] });`	Click on an element with keyboard modifiers (e.g., Ctrl, Shift)
`await page.locator('#button').hover();`	Hover over an element
`await page.locator('#source').dragTo(page.locator('#target'));`	Drag an element and drop it onto another element

Mouse

`await page.mouse.move(100, 200);`	Move the mouse to specific coordinates relative to the viewport

`await page.mouse.down();`
`await page.mouse.up();`
	Click the mouse at the current position

`await page.mouse.down();`	Press the mouse button down
`await page.mouse.down({ button: 'right'});`	Press the mouse button down with options
`await page.mouse.up();`	Release the mouse button
`await page.mouse.wheel(0, 100);`	Scroll the mouse wheel by deltaX and deltaY

Keyboard

`await page.keyboard.press('Enter');`	Press a key
`await page.keyboard.press('Control+A');`	Press a key chord(e.g., Control+A to select all)
`await page.keyboard.down('Shift');`	Hold a key down (without releasing it)
`await page.keyboard.up('Shift');`	Release a key that is being held down

Form Input Element Interactions

`await page.locator('#input').fill('Hello World');`	Fill a text input
`await page.locator('#input').press('Enter');`	Use press to type text input
`await page.locator('#input').press('Control+A');`	Use press to send a key chord
`await page.locator('#input').pressSequentially( 'Hello', { delay: 100 } );`	Use pressSequentially to type text input with a delay between each key
`await page.locator('#input').clear();`	Clear a text input
`await page.locator('#checkbox').check();`	Check a checkbox or radio button
`await page.locator('#checkbox').uncheck();`	Uncheck a checkbox or radio button
`await page.locator('.select-color').selectOption('Red');`	Select an option in a dropdown by label or value
`await page.locator('.select-color').selectOption([ 'Red', 'Green' ]);`	Select multiple options in a multi-select dropdown