TypeScript Cheat Sheet for Playwright

1. Basic Test Structure

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

test('basic test', async ({ page }) => {
  await page.goto('https://example.com');
  await expect(page).toHaveTitle(/Example/);
});
```

2. Locators with Strong Typing

```
const button = page.locator('button#submit');
await button.click();
await expect(button).toBeVisible();
```

3. Page Object Model (POM)

```
export class LoginPage {
  readonly page: Page;
  readonly usernameInput: Locator;
  readonly passwordInput: Locator;

  constructor(page: Page) {
    this.page = page;
    this.usernameInput = page.locator('#username');
    this.passwordInput = page.locator('#password');
  }

  async login(user: string, pass: string): Promise<void> {
    await this.usernameInput.fill(user);
    await this.passwordInput.fill(pass);
  }
}
```

4. Type Assertions

```
const count: number = await page.locator('.item').count();
expect(count).toBeGreaterThan(0);
```

5. Custom Test Fixtures

```
import { test as base } from '@playwright/test';

type MyFixtures = {
  userEmail: string;
};

export const test = base.extend<MyFixtures>({
  userEmail: async ({}, use) => {
```

```
    await use('test@example.com');
  },
});
```

6. Utility Functions with Types

```
export async function waitForApiResponse(
  page: Page,
  url: string
): Promise<Response> {
  return await page.waitForResponse((res) => res.url().includes(url) && res.status() === 200);
}
```

7. Error Handling with Async/Await

```
try {
  await page.click('button.danger');
} catch (error: unknown) {
  console.error('Click failed:', error);
}
```

8. Config with Type Safety

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  use: {
    headless: true,
    baseURL: 'https://yourapp.com',
  },
});
```

Why TypeScript for Playwright?

```
- Autocomplete & IntelliSense
- Fewer runtime errors
- Better team collaboration
- Clean, reusable code
```