Puppeteer Testing & Debugging Cheatsheet

A practical guide to testing and debugging Puppeteer scripts, covering common techniques and tools for robust automation.

Debugging Fundamentals

Console Logging

Using `console.log` within Puppeteer's `evaluate` method is crucial for inspecting the page's state.

Example:

```
await page.evaluate(() => {
  console.log('Current URL:', location.href);
  console.log('Document body:', document.body.innerHTML);
});
```

To view console messages from within the browser, listen to the `console` event on the page.

Example:

```
page.on('console', msg => {
  console.log('PAGE LOG:', msg.text());
});
```

Debugging with Chrome DevTools

Launch Puppeteer with the `headless: false` option to use Chrome DevTools.

Example:

```
const browser = await puppeteer.launch({ headless: false, slowMo: 250 });
```

Set `slowMo` to add a delay, making debugging easier.

Use `debugger` statements within `page.evaluate()` to pause execution and inspect the page in DevTools.

Example:

```
await page.evaluate(() => {
  debugger;
  console.log('This will pause execution.');
});
```

Alternatively, use `await page.pause();` to pause execution and open Chrome DevTools.

Example:

```
await page.goto('https://example.com');
await page.pause();
```

Error Handling

Wrap Puppeteer code in `try...catch` blocks to handle exceptions gracefully.

Example:

```
try {
  await page.goto('https://example.com');
  await page.click('#nonExistentButton');
} catch (error) {
  console.error('An error occurred:', error);
}
```

Listen to the `pageerror` event to catch JavaScript errors within the page.

Example:

```
page.on('pageerror', err => {
  console.error('PAGE ERROR:', err);
});
```

Testing Techniques

Basic Assertions

Use assertion libraries like `assert` or `Chai` for verifying expected outcomes.

Example with `assert`:

```
const assert = require('assert');

const title = await page.title();
assert.strictEqual(title, 'Example Domain', 'Title should match');
```

Visual Regression Testing

Capture screenshots and compare them against baseline images to detect visual changes.

Example:

```
const fs = require('fs');

await page.screenshot({ path:
'screenshot.png' });
// Compare screenshot.png with a
baseline image.
```

Consider using libraries like `jest-image-snapshot`.

Using Jest with image snapshot

```
const puppeteer = require('puppeteer');
const { toMatchImageSnapshot } =
require('jest-image-snapshot');

expect.extend({ toMatchImageSnapshot });

describe('Visual Regression Testing', ()
=> {
  it('should capture and compare a
screenshot', async () => {
    const browser = await
puppeteer.launch();
    const page = await
browser.newPage();
    await
page.goto('https://example.com');
    const image = await
page.screenshot();


expect(image).toMatchImageSnapshot();

    await browser.close();
  });
});
```

End-to-End Testing

Simulate user interactions to test complete workflows.

Example:

```
await page.goto('https://example.com');
await page.type('#searchBox',
'Puppeteer');
await page.click('#searchButton');
await
page.waitForSelector('.searchResults');
```

Advanced Debugging

Tracing

Use tracing to analyze performance and identify bottlenecks.

Example:

```
await page.tracing.start({path:
'trace.json'});
await page.goto('https://example.com');
await page.tracing.stop();
```

Open `trace.json` in Chrome DevTools to analyze the trace.

Network Interception

Monitor and modify network requests and responses.

Example:

```
await page.setRequestInterception(true);
page.on('request', request => {
  console.log('Request:',
request.url());
  request.continue();
});
```

To abort requests based on URL:

```
await page.setRequestInterception(true);
page.on('request', request => {
  if (request.url().endsWith('.png')) {
    request.abort();
  } else {
    request.continue();
  }
});
```

EvaluateOnNewDocument

Inject scripts that run before any other script on the page.

Example:

```
await page.evaluateOnNewDocument(() => {
  window.alert = () => {}; // Disable
alerts
});
```

Test Framework Integration

Jest Integration

Example setup with Jest:

```
const puppeteer = require('puppeteer');

describe('Example Test', () => {
  let browser;
  let page;

  beforeAll(async () => {
    browser = await puppeteer.launch();
    page = await browser.newPage();
    await
page.goto('https://example.com');
  });

  afterAll(async () => {
    await browser.close();
  });

  it('should display the correct title',
async () => {
    const title = await page.title();
    expect(title).toBe('Example
Domain');
  });
});
```

Mocha & Chai Integration

Example setup with Mocha and Chai:

```
const puppeteer = require('puppeteer');
const assert = require('chai').assert;

describe('Example Test', () => {
  let browser;
  let page;

  before(async () => {
    browser = await puppeteer.launch();
    page = await browser.newPage();
    await
page.goto('https://example.com');
  });

  after(async () => {
    await browser.close();
  });

  it('should display the correct title',
async () => {
    const title = await page.title();
    assert.equal(title, 'Example
Domain', 'Title should match');
  });
});
```

Continuous Integration (CI)

Configure your CI environment to run Puppeteer tests in a headless environment. Ensure necessary dependencies are installed (e.g., Chrome or Chromium).

Use environment variables to configure Puppeteer's launch options for CI.

Example using GitHub Actions:

```
name: Puppeteer Tests

on: [push]

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Use Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '14.x'
    - name: Install dependencies
      run: npm install
    - name: Run Puppeteer tests
      run: npm test
```