techSmartHire

Javascript/TypeScript SDET - Test Automation

Question 1: Single Choice

Question: In a Playwright test suite, multiple tests run in parallel and need to maintain independent authentication states. Test A logs in as an admin user, while Test B logs in as a regular user. What is the MOST appropriate approach to ensure complete isolation between these tests?

1. Use a single browser instance and clear cookies between tests using **context.clearCookies()**

2. Create separate **BrowserContext** instances for each test, each with its own isolated storage and cookies

3. Use the same **BrowserContext** but create different Page instances for each test

4.Restart the entire browser instance between each test using **browser.close()** and **browser.launch()**

Correct Answer: Option 2

Explanation: Creating separate **BrowserContext** instances provides complete isolation with independent storage, cookies, and cache. This is more efficient than restarting the browser while providing better isolation than just using separate pages. Each context acts like a separate incognito session.

techSmartHire

Question 2: Single Choice

Question: A Playwright test needs to verify that a success message appears after submitting a form. The message is dynamically inserted into the DOM by JavaScript after an API call completes. The test fails intermittently with "element not found" errors. What is the BEST solution?

1. Add **await page.waitForTimeout(3000)** before checking for the message

2. Use **await page.waitForSelector('.success-message', { state: 'visible' })** before asserting

3. Wrap the assertion in a try-catch block and retry if it fails

4. Use **await page.locator('.success-message')** without any wait, as Playwright auto-waits

Correct Answer: Option 2

Explanation: While Playwright has auto-waiting for most actions, explicitly waiting for the element to be visible ensures the test only proceeds when the dynamic content is rendered. This is more reliable than fixed timeouts and more explicit than relying solely on auto-waiting for assertions on elements that may not exist initially.

techSmartHire

Question 3: Multi-Select

Question: Which of the following are recommended best practices when building a Playwright automation framework? (Select all that apply)

1. Use **page.getByRole()** and **page.getByLabel()** locators to make tests more resilient to UI changes
2. Store test data and configuration directly in test files for easier access
3. Use **test.beforeEach()** to ensure each test starts with a clean, predictable state
4. Implement Page Object Model to encapsulate page interactions and locators
5. Use **page.waitForTimeout()** extensively to handle all synchronization issues
6. Enable trace recording in CI/CD pipelines to debug failures effectively

Correct Answer: Option 1,3,4,6

- Option 1: ✓ Semantic locators (role, label) are more resilient and align with accessibility best practices
- Option 3: ✓ beforeEach() hooks ensure test isolation and consistent starting states
- Option 4: ✓ POM pattern improves maintainability by centralizing page logic
- Option 6: ✓ Traces provide detailed debugging information when tests fail in CI

techSmartHire

Question 4: Fill in the blanks

In Playwright, when you need to locate a button that users would identify by its visible text label "Submit Order" rather than by its CSS class or ID, you should use the _____ locator method.

```
page.getByRole('button', { name: 'Submit Order' }) or getByRole
```

Correct Answer Explanation

Explanation: The **getByRole()** locator finds elements by their ARIA role and accessible name, matching how users and assistive technologies interact with the page. This approach is more resilient to implementation changes than CSS selectors or XPath.

techSmartHire

Question 5: Coding Question

Write a Javascript logic to reverse the given String

javascript

```
      function reverseString(str) {
         return str. split (‘ ‘).reverse(). join(‘ ’);
}
    // Example usage:
console.log(reverseString("hello")); // Output: "olleh"
console.log(reverseString("JavaScript")); // Output: "tpircSavaJ"
```

Using a loop (more traditional):

javascript

```
        function reverseString (str) {
          let reversed = ‘ ’;
            for(let i = str.length() - 1; i >= 0; i--) {
            reversed += str[i];
        }
        return reversed;
}
```

Continued in the next page

techSmartHire

Question 5: Coding Question- Answer Continued..

Using spread operator (modern ES6):

```javascript
function reverseString(str) {
  return [...str].reverse().join('');
}
```

Using Array.from():

```javascript
function reverseString(str) {
  return Array.from(str).reverse().join('');
}
```

Using reduce():

```javascript
function reverseString(str) {
  return str.split(' ').reduce((reversed, char) => char + reversed, '');
}
```

techSmartHire

Question 6: Single Choice

Question: What will be the output of the following code?

```
console.log('Start');

setTimeout(() => {
  console.log('Timeout');
}, 0);

Promise.resolve().then(() => {
  console.log('Promise');
});

console.log('End');
```

1. Start, Timeout, Promise, End

2. Start, End, Timeout, Promise

3. Start, End, Promise, Timeout

4. Start, Promise, End, Timeout

Correct Answer: Option 3

Explanation: JavaScript's event loop processes microtasks (Promises) before macrotasks (setTimeout). The execution order is: synchronous code first ('Start', 'End'), then microtasks ('Promise'), then macrotasks ('Timeout'). Even though setTimeout has a 0ms delay, it's queued as a macrotask.

techSmartHire

Question 7: Single Choice

Question: You are using Playwright to test API endpoints. After sending a POST request to create a user, you need to verify that the response status is 201 and the response body contains the created user's email. Which approach is MOST appropriate?

1. Use page.request.post() and check response.status() and await response.json() to validate the response body

2. Use page.goto() to navigate to the API endpoint and check the displayed JSON

3. Use fetch() inside page.evaluate() to make the API call from the browser context

4. Use page.waitForResponse() to intercept the network call made by the browser

Correct Answer: Option 1

Explanation: Playwright's page.request API is specifically designed for API testing. It allows direct HTTP requests without a browser context, making it efficient for API validation. You can check response.status() for the status code and await response.json() to parse and validate the response body.

techSmartHire

Question 8: Single Choice

Question: A Playwright test suite needs to authenticate users via API before running UI tests. The authentication token must be reused across multiple tests to avoid repeated login API calls. What is the BEST approach?

1. Make a login API call in test.beforeEach() for every test

Use test.beforeAll() to authenticate once per worker and store the token in global state

3. Authenticate in the first test and pass the token to subsequent tests via test parameters

4. Store authentication credentials in localStorage before each test using page.evaluate()

Correct Answer: Option 2

Explanation: Using test.beforeAll() authenticates once per worker process, making tests faster while maintaining isolation between different worker processes. Playwright's storage state feature can persist authentication across tests. This is more efficient than authenticating before each test while maintaining proper test isolation.

techSmartHire

Guidelines & Learning Resources

Prepare yourself to success

Important Exam Guidelines

- Sit in a **quiet place** with a plain background and **good lighting** to ensure a **professional environment**
- Ensure **stable internet connection** throughout the exam to avoid disruptions
- Keep your **camera ON** at all times during the assessment for proctoring purposes
- Do **NOT switch tabs or windows** during the exam - this may result in automatic disqualification

Learning Resources

Need guidance to clear the assessment with a great score? Check out the systematic learning resources below to learn from scratch.

https://courses.rahulshettyacademy.com/p/java-becomew-full-stack-qa-automation-expert

☀️ All the Best! ☀️
You've prepared well, and we believe in your abilities!
Stay calm, focused, and confident during the assessment.
Success is just around the corner!