Here are the YouTube links for the 10 essential videos to help you prepare for SDET/Automation Testing roles:  
1. SDET Role Explained – Naveen AutomationLabs  
<https://lnkd.in/gpYdwyVx>  
2. Java for Testers Crash Course – Amigoscode  
<https://lnkd.in/gBdz4WFN>  
3. Selenium WebDriver Basics – Naveen AutomationLabs  
<https://lnkd.in/g29xkwJw>  
4. TestNG Tutorial – Mukesh Otwani  
<https://lnkd.in/gegaJyeU>  
5. REST API Automation with Postman – Automation Step by Step  
<https://lnkd.in/grziPhzY>  
6. API Testing with RestAssured (Java) – Rahul Shetty  
<https://lnkd.in/gZUwKUTM>  
7. Framework Design from Scratch (Hybrid/BDD) – Naveen AutomationLabs  
<https://lnkd.in/gFZ7geYp>  
8. CI/CD with Jenkins for Automation – The Testing Academy  
<https://lnkd.in/gfc-ward>  
9. Real-time SDET Mock Interview – Let’s Kode It  
<https://lnkd.in/g_6TZsD9>  
10. Top 20 SDET Interview Questions – QAFox  
<https://lnkd.in/gGzdhXHm>  
Activate to view larger image,

Practice Your QA Automation Skills with Real-World Examples  
If you’re an aspiring QA Automation Engineer looking to enhance your expertise and get hands-on experience with automation testing, these platforms offer excellent opportunities:  
  
Top Platforms for QA Automation Practice:  
• The Internet Herokuapp: <https://lnkd.in/dCHeAsQR>  
• UI Testing Playground: <https://lnkd.in/dfNVbw9A>  
• The Playground: <https://lnkd.in/dbRYd2qv>  
  
Best Demo Sites Featuring E-Commerce and Banking:  
• ParaBank: <https://lnkd.in/dmvrMPue>  
• Luma - Magento eCommerce: <https://lnkd.in/dr3tCdse>

Interview Questions

How to choose a date from date picker from a webpage using selenium

How to get a particular value from dynamic web table

Arrange String in ascending order

How to switch between windows using set

What is Data driven and Keyword driven frameworks

What are the testing techniques we use in manual tesing

which methodology and type we follow

How we Write test cases , on what basis

what is Residual defects

Here are the Top Interview Questions, Exp Range: 1 to 3 yrs for SDETs  
  
>>𝗔𝗣𝗜𝘀:  
1) What are the components of an HTTP request?  
2) What is the difference between API and unit testing?  
3) What is an HTTP response?  
4) How we can add validation points in postman?  
5) What do you understand by Server-side validation?  
6) What is 3 tier Architecture?  
7) Difference between webservice & APIs  
8) What is Rest, Soap & GraphQL in APIs  
9) What do you test in standalone API?  
10) What do test in 3rd party integrated APIs?  
  
>>𝗣𝗼𝘀𝘁𝗺𝗮𝗻:  
1) When to use collection, environment & global variables  
2) How to execute a collection E2E  
3) How to validate a API response has correct status code?  
4) What happens when API response returns Form Data instead of Json, how to validate it?  
5) How to setup Basic Auth in Postman?  
6) Where do you store Environment credentials?  
7) How to save a demo response for an API request?  
8) How will you validate an API rqst if VPN is required for it to work?  
9) How do you filter results in a API request using Postman?  
10) How to setup custom headers in Postman?  
  
>>𝗦𝗲𝗹𝗲𝗻𝗶𝘂𝗺 𝗪𝗲𝗯𝗗𝗿𝗶𝘃𝗲𝗿:  
1) What are different types of locators in Selenium?  
2) When do you use Xpath over CSS locators?  
3) How to get a specific value from a dropdown and reuse it in verifications?  
4) When do we use JavaScript Executors?  
5) Is it possible to validate Captcha using Selenium, if Yes how?  
6) What should be the ideal way to store data using Selenium WebDriver only?  
7) Is it possible to use Xpath like a parent/child/node/.. ? If yes, then provide an example where to use it?  
8) What happens if you receive browser notifications in between Test Automation Execution?  
9) Why does Stale Element exception occur and how to handle it?  
10) What is an Invalid Certificate Exception?  
  
>>𝗚𝗶𝘁:  
1) What are the different stages in committing the code to GitHub?  
2) Is it possible to revert back changes in a remote repo? If yes, how?  
3) When do you commit your code? After committing how do you validate everyone has the updated code now?  
4) How to merge stashed changes in a local repo?  
5) Why do we need a GitIgnore file? How to add ignore files in it?  
  
  
>>𝗧𝗲𝘀𝘁𝗡𝗚:  
1) What is the execution format of Tests in TestNG?  
2) Can priority be negative for methods? If Yes, then what is the execution flow as per priority?  
3) What is the difference in DependonMethod & dependsongroups?  
4) What are different ways to exclude tests in TestNG  
5) What does ThreadPoolSize mean in TestNG? How does it work?  
6) Why do we need BeforeSuite & AfterSuite annotations?  
  
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~  
  
Deloitte  
1 Tell me about yourself / Give a brief introduction.  
2. Can you explain the framework you are currently working on in your automation projects?  
3. Write a Java program to count the occurrences of each character in a given string.  
4. Write a Java program to find duplicate characters in a string and remove them.  
5. How do you select a value from a dropdown menu using Selenium WebDriver?  
6. What is the difference between Checked and Unchecked Exceptions in Java?  
7. How do you handle "StaleElementReferenceException" in Selenium?  
8. Which types of exceptions have you handled in your automation framework?  
9. How do you achieve parallel execution in Selenium TestNG framework?  
10. What are the different types of waits in Selenium? Explain Explicit Wait with syntax.  
11. How would you switch to a browser window based on a specific title in Selenium?

PayPal  
  
PayPal Interview Experience – Technical Round (3-4 Years Experience, 45 mins)  
  
Core Java & Data Structures -  
Explain the internal working of ConcurrentHashMap. How does it achieve thread safety, and what are its performance trade-offs?  
Describe the differences between WeakHashMap and HashMap. When would you use each?  
Given a large dataset (millions of records), how would you efficiently search for duplicate transactions?  
How does the Java Memory Model impact multi-threaded applications?  
Design an in-memory key-value store that supports TTL (time to live) for entries.  
  
Multithreading & Concurrency  
How would you design a thread pool from scratch?  
Explain how Java handles false sharing in multi-core processors.  
What are the main drawbacks of using synchronized methods? How does ReentrantLock improve performance?  
Implement a multi-threaded rate limiter that handles API requests.  
How would you handle thread starvation in high-throughput applications?  
  
Spring Boot & Microservices  
How does Spring Boot’s auto-configuration determine which beans to load?  
What is Circuit Breaker, and how does it prevent cascading failures in microservices?  
How would you handle blue-green deployments in a Spring Boot microservices setup?  
Explain Eventual Consistency. How would you ensure data consistency across distributed services?  
Implement a rate-limited API in Spring Boot that handles 10,000 requests per second.  
  
APIs & RESTful Services  
How would you design an API Gateway to handle dynamic routing and security policies?  
Explain the challenges of handling pagination in REST APIs for massive datasets.  
How would you manage API timeouts and retries in a distributed system?  
What’s the best way to implement WebSockets in a fintech application?  
How would you enforce idempotency in payment APIs?  
  
System Design Basics  
Design a high-throughput, low-latency order-matching system for a stock exchange.  
How would you ensure data integrity in a multi-region database setup?  
Explain Leader Election. How would you implement it in a microservices-based system?  
What are the trade-offs between CQRS and traditional CRUD-based systems?  
How does a Distributed Message Queue (like Kafka) handle backpressure?  
  
tip: They look for scalability thinking, real world problem solving, and strong system design intuition. If you’re aiming for backend-heavy fintech roles, focus on concurrency, API optimizations, and microservices resilience patterns. And don’t forget system design  
  
Scenario Based Test Automation Interview Questions :  
  
(1) You are asked to automate a functionality that is not yet fully developed. How do you handle this?  
(2) If a Test fails, what will be your next step?  
(3) If the application has minor changes, what would be your approach to modifying the Automation scripts?  
(4) How would you automate login functionality for a website?  
(5) How would you automate a Test scenario where you need to check if an email is sent after a user registration?  
(6) If there is a scenario that takes a long time to execute, would you prefer Manual TestNG or Automation Testing? Why?  
(7) How would you automate a scenario where you need to validate the contents of a downloaded file after clicking a button on a webpage?  
(8) How would you automate a scenario where you need to verify a specific colour, font, and position of an element on a webpage?  
(9) How would you handle pop-up windows or alert boxes in your Automation script?  
(10) How would you automate a scenario where you need to verify if a user is able to scroll down a webpage until the footer section is visible?  
(11) You've been asked to automate a legacy application. What is your approach?  
(12) A script you wrote was working fine yesterday but is failing today. How do you troubleshoot?  
(13) Your Automation scripts are running slowly. How can you improve the speed?  
(14) Your Automation script is failing due to a change in the application. How do you handle this?  
(15) How would you automate a scenario where you need to verify a specific colour, font, and position of an element on a webpage?

Que  
  
RegEx in Java  
Alternate for sendKeys  
How to add Map to List  
GIT conflicts  
SQL joints  
How to get and avoid duplicates in ArrayList  
Cucumber report how to attach Screenshot  
Why we can’t implement Interface in Interface  
We cannot implement anything in to methods  
  
Windows pop up handle  
  
Sequli  
Split string and store in array  
Day to day activities  
SQL which has m in name  
SQL fetch duplicates  
Image handle  
Where we prefer css selector over xpath  
Burn up and burn down chart in jira  
Interface  
Polymorphism  
StaleElementReferenceException  
Cucumber report  
Occurrence oh char in a word

After wasting the last 3 months on interview prep, I found only 10 videos you need to watch for SDET / Automation Test  
  
  
  
  
~~~~~~~~~#Automation #Testing~~~~~~~~~

Xpath  
  
𝐗𝐏𝐚𝐭𝐡 - 𝐐𝐮𝐢𝐜𝐤 𝐠𝐮𝐢𝐝𝐞.  
  
XPath is a key tool for automating web testing. While basic XPath is useful, mastering advanced techniques can significantly improve your test automation. Here’s a rundown of powerful XPath methods:  
  
1. Absolute XPath  
Full, exact path from the root to the element.  
Example: `/html/body/div[2]/table/tr[3]/td[4]`  
  
2. Relative XPath  
Locates elements using attributes without needing the full path.  
Example: `//input[@type='text' and @name='username']`  
  
3. Wildcard XPath  
Uses `\*` to match any element, helpful for dynamic attributes.  
Example: `//\*[contains(@class, 'button')]`  
  
4. Indexed XPath  
Targets a specific element by index within matching elements.  
Example: `(//div[@class='item'])[3]`  
  
5. Sibling XPath  
Selects elements at the same level as the target.  
Example: `//h1/following-sibling::p[1]`  
  
6. Parent XPath  
Selects the parent of a target element.  
Example: `//div[@id='container']/parent::body`  
  
7. Ancestor XPath  
Finds any ancestor of the target element.  
Example: `//div[@id='content']//h2/ancestor::div`  
  
Additional Techniques:  
  
8. Text-Based XPath  
Locates elements by visible text.  
Example: `//button[text()='Submit']`  
  
9. `contains()` XPath  
Finds elements with partial matching text or attributes.  
Example: `//div[contains(@id, 'error')]`  
  
10. `starts-with()` XPath  
Locates elements whose attribute values begin with a specific string.  
Example: `//input[starts-with(@name, 'user')]`  
  
11. Chained XPath  
Combines multiple conditions to precisely locate elements.  
Example: `//div[@class='container']//span[@class='label'][text()='Username']`  
  
12. Following XPath  
Selects elements that come after the target element.  
Example: `//div[@class='header']/following::input[1]`  
  
13. Preceding XPath  
Selects elements that come before the target element.  
Example: `//h2[@id='title']/preceding::div[2]`  
  
14. OR and AND in XPath  
Combines multiple conditions for more flexible searches.  
Example: `//input[@type='text' or Type Email

#Important Selenium Methods  
  
• Navigation Methods:  
get(url): Navigate to a webpage.  
navigate().to(url): Similar to get(url).  
navigate().back(): Go back to the previous page.  
navigate().forward(): Go forward to the next page.  
navigate().refresh(): Refresh the current page.  
  
• Element Interaction Methods:  
  
findElement(By locator): Find an element on the webpage.  
findElements(By locator): Find multiple elements on the webpage.  
click(): Click on an element.  
sendKeys(String text): Send keys to an element (e.g., text input).  
clear(): Clear the text from an element.  
getText(): Get the text from an element.  
getAttribute(String name): Get the attribute value of an element.  
isSelected(): Check if an element is selected (e.g., checkbox).  
isEnabled(): Check if an element is enabled.  
isDisplayed(): Check if an element is displayed.  
  
• Waiting Methods:  
  
implicitly Wait (long time, TimeUnit unit): Set implicit wait for element presence.  
WebDriverWait wait = new WebDriverWait(driver, time): Create an explicit wait instance.  
wait.until(ExpectedConditions.condition): Wait for a specific condition to occur.  
  
• Alert and Window Methods:  
  
switchTo().alert(): Switch to an alert.  
dismiss(): Dismiss an alert.  
accept(): Accept an alert.  
sendKeys(String text): Send keys to an alert.•  
  
Cookie Methods:  
  
manage().getCookies(): Get all cookies.  
manage().getCookieNamed (String name): Get a cookie by name.  
manage().addCookie (Cookie cookie): Add a cookie.  
manage().deleteCookie (Cookie cookie): Delete a cookie.  
manage().delete AllCookies(): Delete all cookies.  
  
• Other Methods:  
  
getTitle(): Get the title of the webpage.  
getCurrentUrl(): Get the current URL.  
getPageSource(): Get the HTML source of the webpage.  
quit(): Quit the driver.  
close(): Close the browser window.

Wildcard  
Design pattern  
  
In a string with 4 words, which word starts in Capital letter that word should be replaced  
  
Functional Interface in Java 8  
It will allow  
Abstract, static and normal methods .  
  
SQL query to get Highest salary  
Groups and orders  
  
Method(Int var, int var, String null, String null)  
Obj.method(value, value)  
Is above correct, will Java accept this ?  
  
How to add value to last string only ?  
Obj.method(value, value, ‘’, Value)  
  
Difference between constant and final variable  
  
How to clean db after every run  
In a array fetch a index of which  
The sum of values in right side is equal to sum of values in left side

𝗤𝗔 𝗔𝘂𝘁𝗼𝗺𝗮𝘁𝗶𝗼𝗻 𝗶𝗻𝘁𝗲𝗿𝘃𝗶𝗲𝘄 𝗾𝘂𝗲𝘀𝘁𝗶𝗼𝗻𝘀 𝗳𝗿𝗼𝗺 𝗮 𝗖𝗮𝗽𝗴𝗲𝗺𝗶𝗻𝗶  
  
1. Write a program to find duplicate elements in a string array.  
2. How would you introduce yourself briefly and effectively?  
3. Explain the framework you have worked with in your automation testing process.  
4. How do you handle closing the second window of a browser in automation testing?  
5. Differentiate between XPath and CSS selectors.  
6. What is the syntax for a LinkText XPath locator?  
7. What changes or setups do you perform before starting execution in your framework?  
8. How do you handle change requests in your application? Describe the steps you follow.  
9. How often do you trigger regression test scripts? How do you manage them in your repository?  
10. What challenges have you encountered in automation testing, and how did you overcome them?  
11. Explain the differences between GET and POST methods in API testing.  
12. What are the essential components of the GET and POST methods?  
13. Discuss HTTP status codes like 401 and 503.  
14. How do you validate the response code in API testing?  
15. What format do you use for assertions in your tests?  
16. Explain the difference between 200 and 201 HTTP status codes.  
17. Provide the syntax for query parameters in API requests.  
  
 Sharing knowledge is the first step toward growth! I hope these questions add value to your preparation.

Topics  
  
String loop forward and reverse  
Array reverse loop  
Object array  
Arrays functions  
Sort array  
Duplicates in array  
Search element  
String split  
Primitive non primitive

Operators  
  
Int a = 1  
Int b = 2  
Int c = a+b  
  
In above  
a b c are variables  
Int is datatype  
a+b is expression  
+ is operator  
= is assignment operator  
  
Once variables involved in expression then they are operants  
So a and b are operants

Why Use static in Java?  
In Java, static makes variables, methods, or inner classes belong to the class itself rather than any specific instance. This can be especially valuable in test-driven environments common to SDET roles. Here’s how it’s useful:  
  
Static Variables – Shared across all instances. Great for things like counters or shared resources across test classes.    
  
Static Methods – Call these without needing an object! Ideal for utility methods, like reusable test data generators.    
  
Static Blocks – Perfect for one-time setups like initializing constants or configuration details.    
  
Static Inner Classes – Can be used without an outer class instance, making them perfect for helper classes in test frameworks.    
  
Interview Tip:  
If you’re asked about static in an interview, try to explain it with practical examples—mention where you’ve implemented it in your Automation Framework for resource sharing, utility functions, or configuration setup.

MongoDB is a popular open-source NoSQL database that stores data in a document- object oriented format, unlike traditional relational databases which store data in tables.  
  
MongoDB stores data as JSON-like documents, making it more flexible and scalable. The name MongoDB is derived from the word humongous  
  
BSON documents are the format used for data stored in MongoDB  
  
BSON binary script object notation  
JSON JavaScript object notation  
  
Collections instead of Tables  
Documents(BSON) instead of row  
Fields instead of columns  
MongoDB is schemaless  
  
Features of MongoDB  
  
Aggregation  
Aggregation is a powerful feature in MongoDB that allows you to process and analyze data across multiple documents in a collection. It enables you to perform complex data manipulations, transformations, and calculations on large datasets.  
  
Schema-less  
Document-Oriented :  
Data stored in form of documents  
Sharding

Flaky tests  
  
Sometimes pass , sometimes fails  
Synchronisation issues  
Environment issues

𝗜𝗻𝘁𝗲𝗿𝘃𝗶𝗲𝘄𝗲𝗿: You have 2 minutes to solve this SQL query.  
Retrieve the department name and the highest salary in each department from the `employees` table, but only for departments where the highest salary is greater than $70,000.  
  
𝗠𝗲: Challenge accepted!  
  
SELECT department, MAX(salary) AS highest\_salary  
FROM employees  
GROUP BY department  
HAVING MAX(salary) > 70000;  
  
I used `GROUP BY` to group employees by department, `MAX()` to get the highest salary, and `HAVING` to filter the result based on the condition that the highest salary exceeds $70,000. This solution effectively shows my understanding of aggregation functions and how to apply conditions on the result of those aggregations.  
  
𝗧𝗶𝗽 𝗳𝗼𝗿 𝗦𝗤𝗟 𝗝𝗼𝗯 𝗦𝗲𝗲𝗸𝗲𝗿𝘀:  
It's not about writing complex queries; it's about writing clean, efficient, and scalable code. Focus on mastering subqueries, joins, and aggregation functions to stand out!

Java coding interview questions ||  
#SDET  
1. Reverse a String:  
  Write a Java program to reverse a given string.  
2. Find the Largest Element in an Array:  
  Find and print the largest element in an array.  
3. Check for Palindrome:  
  Determine if a given string is a palindrome (reads the same backward as forward).  
4. Factorial Calculation:  
  Write a function to calculate the factorial of a number.  
5. Fibonacci Series:  
  Generate the first n numbers in the Fibonacci sequence.  
6. Check for Prime Number:  
  Write a program to check if a given number is prime.  
7. String Anagrams:  
  Determine if two strings are anagrams of each other.  
  
8. Array Sorting:  
  Implement sorting algorithms like bubble sort, merge sort, or quicksort.  
  
9. Binary Search:  
  Implement a binary search algorithm to find an element in a sorted array.  
  
10. Duplicate Elements in an Array:  
  Find and print duplicate elements in an array.  
  
11. Linked List Reversal:  
  Reverse a singly-linked list.  
  
12. Matrix Operations:  
  Perform matrix operations like addition, multiplication, or transpose.  
  
13. Implement a Stack:  
  Create a stack data structure and implement basic operations (push, pop).  
  
14. Implement a Queue:  
  Create a queue data structure and implement basic operations (enqueue, dequeue).  
  
15. Inheritance and Polymorphism:  
  Implement a class hierarchy with inheritance and demonstrate polymorphism.  
  
16. Exception Handling:  
  Write code that demonstrates the use of try-catch blocks to handle exceptions.  
17. File I/O:  
  Read from and write to a file using Java's file I/O capabilities.  
18. Multithreading:  
  Create a simple multithreaded program and demonstrate thread synchronization.  
19. Lambda Expressions:  
  Use lambda expressions to implement functional interfaces.  
20. Recursive Algorithms:  
  Solve a problem using recursion, such as computing the factorial or Fibonacci sequence.

Realtime 40 QA Interview Questions   
  
1. What is Constructor and Abstraction in Java?  
2. What is SDLC and STLC? And Explain its phases.  
3. Define your roles and responsibility.  
4. What is regression testing?  
5. What are different methodologies of SDLC? Explain each.  
6. Define Agile?  
7. Define Scrum And Sprint?  
8. What is the estimation in Sprint?  
9. What is sprint backlog?  
10. What are the different reports in Testing?  
11. What are the key components of the TestCase report?  
12. What are the components of a defect report?  
13. What is Jira?  
14. How do you log defect in Jira?  
15. How do you link bugs with the user story?  
16. What is sprint?  
17. Define black box and white box testing,  
18. Define functional testing.  
19. Define the oops concept in java.  
20. Give me examples of oops which you used in your framework.  
21. What is TestNG?  
22. What is usability testing?  
23. What are the steps for reporting the defect in Jira?  
24. Define Structure of Selenium?  
25. How will you handle the dropdown in selenium?  
26. Different types of wait in selenium? Explain each of them.  
27. Difference between hard and soft assertion?  
28. Why are we using "WebDriver driver = new ChromeDrive ()"?  
Why can't we write RemoteDriver driver = new ChromeDrive();  
29. Explain the different Annotations in TestNG?  
30. Define Priority and Severity of the Bug?  
31. How to maximize the screen in Selenium?  
32. What are the different closure reports?  
33. What is the difference between a Test Plan and a Test Strategy document?  
34. Define Bug lifecycle of JIRA.  
35. What is final keyword?  
36. What is Constructor Overloading?  
37. How do you handle frames in Selenium?  
38. Write Scenario outline in Cucumber.  
39. What is Explicit wait and write syntax.  
40. What is difference between Quit and Close in Selenium?  
  
If you find it useful, please like and comment!

CAR  
Statista  
  
India’s best workplaces for IT  
  
Ambition box  
  
Companies entering India 2024  
  
Target At least 25 companies  
  
How Can I contribute to the growth opportunities in your company  
  
Google Gemini

JavaScript

1. What is JavaScript?  
2. What are the data types supported by JavaScript?  
3. What is the difference between `let`, `const`, and `var`?  
4. Explain how `==` and `===` differ.  
5. What is a closure?  
6. What is hoisting?  
7. Explain the concept of "this" in JavaScript.  
8. What are JavaScript prototypes?  
9. What is the difference between `null` and `undefined`?  
10. How does JavaScript handle asynchronous operations?  
11. What is a promise?  
12. What are async/await functions?  
13. Explain event delegation in JavaScript.  
14. What are JavaScript modules?  
15. How can you prevent a function from being called multiple times?  
16. What is the event loop?  
17. What is the difference between `apply()` and `call()` methods?  
18. What is `bind()` method used for?  
19. What is a JavaScript event loop?  
20. Explain the concept of "event bubbling" and "event capturing".  
21. What is the difference between `deep copy` and `shallow copy`?  
22. What are generator functions?  
23. What is the `new` keyword used for?  
24. How do JavaScript’s `setTimeout` and `setInterval` work?  
25. What is a `WeakMap` and how is it different from a `Map`?  
26. What is a `Set` in JavaScript?  
27. What is `Object.create()` used for?  
28. How does JavaScript’s garbage collection work?  
29. What are "decorators" in JavaScript?  
30. Explain the difference between `prototype` and `\_\_proto\_\_`.  
31. What is the purpose of `Object.assign()`?  
32. What are "template literals"?  
33. What is the `spread` operator?  
34. What is the `rest` parameter?  
35. Explain the `for...of` loop.  
36. What are `async` and `await` keywords used for?  
37. What is `Symbol` used for in JavaScript?  
38. How do you create a class in JavaScript?  
39. What is destructuring in JavaScript?  
40. What is the `Proxy` object?

JavaScript  
  
Keywords::  
Let -> local scope — to keep private  
Var -> Global scope — upto outer function  
const -> Value is constant, cannot change  
  
Node.js is Javascript runtime environment  
To run Java script we will need this node.js  
Installed  
  
Console.log(“print”) - this is print statement  
  
Console.log(typeof(10)) - this will print the data type of variable  
Above is number  
  
Console.log(typeof(“10”))-  
Above is String  
  
Console.log(typeof(true)) -  
Above is Boolean  
  
Console.log(typeof(null)) -  
Above is object  
  
let a  
Console.log(a)  
If you try to print without initialising a value is will print - undefined  
Console.log(typeof(a)) -  
Its type also undefined  
  
If you store null as a value to variable, it will print null  
If didn’t specified a value , it is print as undefined  
  
Data types  
Number  
String  
Boolean  
Null  
Undefined  
  
Create file name.js  
  
How to Create function  
  
Function functionName() {  
Logic }  
  
To run we need to call function name  
Example:  
  
Function demo(){  
Logic }  
  
Demo()  
  
NaN- Not a Number  
  
  
Arrays  
=====  
To add value  
Use push()  
Ref.push(1);  
The is will add the value in last of array  
We can also add multiple values  
Ref.push(1,2,3,4)  
  
To remove value  
Ref.pop()  
By default it will remove last value  
  
Ref.unshift(10)  
To add value in first  
  
Ref.shift()  
Remove the first value  
  
Ref.fill(0)  
This will replace all value with 0  
  
Ref.fill(0, 1, 3)  
This will replace values in between 1 and 3  
1 is inclusive and 3 is exclusive  
  
Ref.sort()  
To make array in ascending order  
  
Ref.reverse()  
To make array n descending order

𝐎𝐧𝐞 𝐨𝐟 𝐦𝐲 𝐟𝐫𝐢𝐞𝐧𝐝𝐬 𝐰𝐚𝐬 𝐫𝐞𝐜𝐞𝐧𝐭𝐥𝐲 𝐚𝐬𝐤𝐞𝐝 𝐭𝐡𝐢𝐬 𝐪𝐮𝐞𝐬𝐭𝐢𝐨𝐧 𝐢𝐧 𝐡𝐢𝐬 𝐭𝐞𝐜𝐡𝐧𝐢𝐜𝐚𝐥 𝐫𝐨𝐮𝐧𝐝:  
𝐐𝐮𝐞𝐬𝐭𝐢𝐨𝐧:Can you walk me through the steps involved in setting up a Selenium Java testing framework from scratch?  
  
𝐀𝐧𝐬𝐰𝐞𝐫:  
𝟏. 𝐒𝐞𝐭 𝐔𝐩 𝐘𝐨𝐮𝐫 𝐃𝐞𝐯𝐞𝐥𝐨𝐩𝐦𝐞𝐧𝐭 𝐄𝐧𝐯𝐢𝐫𝐨𝐧𝐦𝐞𝐧𝐭:  
- Install JDK, choose an IDE (e.g., IntelliJ IDEA or Eclipse), and set up Maven.  
  
𝟐. 𝐂𝐫𝐞𝐚𝐭𝐞 𝐚 𝐌𝐚𝐯𝐞𝐧 𝐏𝐫𝐨𝐣𝐞𝐜𝐭:  
- Start a Maven project and configure the project structure (src/main/java for code, 𝐬𝐫𝐜/𝐭𝐞𝐬𝐭/𝐣𝐚𝐯𝐚 for tests).  
  
𝟑. 𝐀𝐝𝐝 𝐃𝐞𝐩𝐞𝐧𝐝𝐞𝐧𝐜𝐢𝐞𝐬 𝐢𝐧 𝐩𝐨𝐦.𝐱𝐦𝐥:  
- Include essential dependencies like Selenium, TestNG, Log4j, WebDriver Manager, and ExtentReports.  
  
𝟒. 𝐒𝐞𝐭 𝐔𝐩 𝐖𝐞𝐛𝐃𝐫𝐢𝐯𝐞𝐫 𝐌𝐚𝐧𝐚𝐠𝐞𝐫:  
- Automate browser driver management to simplify setup.  
  
𝟓. 𝐂𝐫𝐞𝐚𝐭𝐞 𝐭𝐡𝐞 𝐁𝐚𝐬𝐞 𝐂𝐥𝐚𝐬𝐬:  
- Implement WebDriver initialization, @𝐁𝐞𝐟𝐨𝐫𝐞𝐂𝐥𝐚𝐬𝐬 𝐬𝐞𝐭𝐮𝐩, and @𝐀𝐟𝐭𝐞𝐫𝐂𝐥𝐚𝐬𝐬 𝐭𝐞𝐚𝐫𝐝𝐨𝐰𝐧 methods.  
  
𝟔. 𝐂𝐫𝐞𝐚𝐭𝐞 𝐏𝐚𝐠𝐞 𝐎𝐛𝐣𝐞𝐜𝐭 𝐌𝐨𝐝𝐞𝐥 (𝐏𝐎𝐌):  
- Define separate classes for each web page or component to encapsulate interactions with web elements.  
  
𝟕. 𝐖𝐫𝐢𝐭𝐞 𝐓𝐞𝐬𝐭 𝐂𝐚𝐬𝐞𝐬:  
- Use TestNG to write and manage test cases in the src/test/java directory.  
  
𝟖. 𝐋𝐨𝐠𝐠𝐢𝐧𝐠 𝐚𝐧𝐝 𝐑𝐞𝐩𝐨𝐫𝐭𝐢𝐧𝐠:  
- Integrate 𝐋𝐨𝐠𝟒𝐣 for logging and ExtentReports for generating detailed HTML test reports.  
  
𝟗. 𝐑𝐮𝐧𝐧𝐢𝐧𝐠 𝐓𝐞𝐬𝐭𝐬:  
- Create a TestNG XML configuration file to manage test execution.  
  
𝟏𝟎. 𝐂𝐨𝐧𝐭𝐢𝐧𝐮𝐨𝐮𝐬 𝐈𝐧𝐭𝐞𝐠𝐫𝐚𝐭𝐢𝐨𝐧 (𝐂𝐈):  
- Set up CI tools like 𝐉𝐞𝐧𝐤𝐢𝐧𝐬 or GitLab CI to automate test execution and reporting in your build pipeline.  
  
  
This comprehensive approach not only ensures a well-structured testing framework but also aligns with best practices in test automation.   
  
Feel free to share your thoughts or additional tips in the comments! 

Barclays Interview Questions for #QA(Automation) role  
  
1. Explain about your profile and experience, skill set in brief.  
2. Do you know OOPs concepts and the framework where and how you have implemented them?  
3. Can we declare a private class?  
4. What is the difference between == and equals?  
5. How is string immutable?  
6. Where do strings get stored, and where does reference get stored?  
7. Can you please explain the memory location and how the string is immutable?  
8. What can be used if you do not want to use the String class?  
9. Difference between String and StringBuffer.  
10. What collections have you used? Have you used HashMap?  
11. List declaration?  
12. Where used Set?  
13. You have an application like Flipkart, and you want to buy a pen, so you have added that item two times to the cart using the add to cart button, but in the cart, only one entry for the item should be displayed with quantity as 2, so how you’ll test this?  
14. Numbers 1 to 100 will be flashed on the screen only once, and you must find the missing number.  
15. I have a table and want to store all table data; which collection should be used and why?  
16. What will HashMap return?  
17. How do we achieve inheritance without having an interface?  
18. Where in the framework have you used Method overloading and method overriding?  
19. If I want my class not to be extended and another class cannot create an instance, how do I declare a class?  
20. How do you store multiple values in one reference?  
21. In cucumber, in which class do you have gluecode, how many classes for gluecode, and what was the program line limit for the class?  
22. How do we find missing implementation in Cucumber?  
23. Used static data or dynamic data, what was the approach?  
24. How do you resolve conflicts while pushing code in Git?  
25. Write a program to merge two sorted arrays.  
26. Input: a[] = {1,3,5,7,9} b[]={2,4,6,8}  
27. Output: C[]={1,2,3,4,5,6,7,8,9}  
28. What is Shift Left testing, and where do we use it?  
  
Book 1:1 Call With Me - <https://lnkd.in/gwPCeYXJ>  
  
Join My Testing Telegram Channel - <https://lnkd.in/g6kx488g>

SDET Java coding interview question:  
  
Input string = "I love my country"  
  
Output string ="I olev ym ocnurty"  
  
Swap the character with its adjacent character , if the adjacent one is space then don't swap it, keep it as it is.

Angular app automation  
We need to use ng-webdriver to automate Angular-based applications. Since Angular or Angular-based applications will use Protactor to automate the test, but we can also do using Java Selenium using this ng-webdriver library.  
Protractor is based on Javascript and TypeScript  
AngularJS application is used generally for single-page application, where the data is very dynamic, where the HTML content is dynamic, and it is used with Angular framework. So Angular is a JavaScript framework. JavaScript technology, UI developers, they are using Angular provided by Google, and people are using that. Now after Angular 2, we don't call AngularJS, we call only Angular.  
  
But in Angular application, guys, what happens? If you talk about Angular application, in Angular application, there are some different types of contents that are available. So we have, let's see, if you've seen ng-binding is there. Then we have ng-model is there. Then we have ng-something like this. Options are available. So these are the different HTML tags that are available. So wherever you see that, OK, something like this, ng-model, ng-binding, ng-receptor, or ng-repeater, or ng-options, or ng-modeling, something like this, OK, it means that is an Angular-based application.  
  
Can I use WebDriver Java? Yes, we can do that without any problem. So, for such application, what you can do that is, you can simply use that. You can create your own XPath and everything. You can do that for these applications. Like, let's see, XPath and the rate ngBind is equal to something like this. Or, simply you can do that. Or, as a selector, you can do that. With the proper custom wait mechanism, you can use a proper explicitly wait, or custom wait, or fluent wait, you can use that. Proper synchronization, custom synchronization, you can write, and then you can do that. But sometimes, what happens is, your custom JavaScript code, sorry, custom WebDriver wait also is not working, or explicitly wait is also not working, your synchronization is not happening properly, and then you start getting some failure cases, and all those things, and then, today it's working, tomorrow it's not working, with a typical normal WebDriver Java and Python. That is a problem. We have seen that for Angular-based applications I'm talking about. Right, I'm not talking about the normal application, I'm talking about Angular or AngularJS-based applications.  
  
First of all, why it's difficult to automate AngularJS applications with WebDriver and Java? Because what happens is that Angular renders dynamic content. Whenever any action is performed on the web page, like you're clicking on a button or you're clicking on a link or something like this, what happens is that an action is rendered behind the scene, in the backend. So, let's see if there is an AngularJS application, and you simply fill this particular form, and there is a button is there, one submit button is there. The moment you click on this particular submit button, what happens is that you're clicking on this particular button, some backend process will render, and one request will be sent to the server, this is my backend server, and then you get the response or let's say you're doing a POST call or GET call or whatever you are doing it, and then, by the time everything is happening in the backend, and then you have to write a custom JavaScript code over there, as a UI developer, that this JavaScript will find what is the state of this particular request that we have sent, and once we get the response from the server, then only the result will be displayed on the UI. So this is a typical mechanism of JavaScript AngularJS engine. So, with this situation, our normal WebDriver Java synchronization won't work. That is the first problem. Sometimes.  
  
And the second problem is that the different locators that, okay, different, you know, HTML tags and, okay, locators that Angular applications they are using, we don't have any support for in Java and Ruby and Python because we have By.id, class name, name, tag name, xpath, CSS selectors, link text, and partial link text. We don't have any ngBind or let's say py.ngbinding or py.model or py.options. We don't have those things. So that's why we have to use Protector for that.

Top 10 Rest Assured Functions that every SDET should know!  
  
1. given(): The given() method is used to set up the request specification. You can add headers, parameters etc.  
  
Ex: given()  
.header("Content-Type", "application/json")  
.param("key", "value")  
  
2. when(): The when() method is used to define the HTTP method (GET, POST, PUT, DELETE) & the endpoint for the request  
  
Ex: when()  
.get("/api/users/2")  
  
3. then(): method is used to define the response validations & assertions  
  
Ex: then()  
.statusCode(200)  
.body("[data.id](http://data.id/)", equalTo(2))  
  
Use Case: Validating the status code & response body after making a request  
  
4. statusCode(): The statusCode() method is used to assert the expected HTTP status code of the response  
  
Ex:  
then() .statusCode(200) (as can be seen above)  
  
5. body(): The body() method is used to assert specific parts of the response body  
  
Ex:  
then() .body("data.email", equalTo("[janet.weaver@reqres.in](mailto:janet.weaver@reqres.in)"))  
  
Use Case:  
Validating that the response body if it contains the expected data  
  
6. log().all(): The log().all() method is used to log the details of the request and response for debugging purposes  
  
Ex:  
given() .log().all() when() .get("/api/users/2") .then() .log().all()  
  
7. header(): The header() method is used to add a single header to the request  
  
Ex:  
given() .header("Authorization", "Bearer token")  
  
Use Case:  
Including an authorization token in the request header  
  
8. param(): The param(), or queryParam(), method is used to add query parameters to the request  
  
Example:  
given() .param("page", 2)  
  
Use Case:  
Sending a query parameter to paginate the results of a GET request  
  
9. contentType(): The contentType() method is used to set the content type of the request.  
Example:  
given() .contentType(ContentType.JSON)  
  
Use Case:  
Specifying that the request payload is in JSON format  
  
10. extract():  
The extract() method is used to extract parts of the response for further validation or use  
  
Example:  
Response response = given() .when() .get("/api/users/2") .then() .extract() .response(); String email = response.path("data.email");  
  
Use Case:  
Extracting the response data to perform additional assertions or operations  
  
-x-x-  
  
To prepare for SDET interviews, you can use my latest updated guide (paid): <https://lnkd.in/g5hr9bea>  
  
Use 950+ Q&A bank for SDETs, created as per recent interviews in 2024: <https://lnkd.in/gSU-m2F7>

Barclays Interview Questions for #QA(Automation) role  
  
1. Explain about your profile and experience, skill set in brief.  
2. Do you know OOPs concepts and the framework where and how you have implemented them?  
3. Can we declare a private class?  
4. What is the difference between == and equals?  
5. How is string immutable?  
6. Where do strings get stored, and where does reference get stored?  
7. Can you please explain the memory location and how the string is immutable?  
8. What can be used if you do not want to use the String class?  
9. Difference between String and StringBuffer.  
10. What collections have you used? Have you used HashMap?  
11. List declaration?  
12. Where used Set?  
13. You have an application like Flipkart, and you want to buy a pen, so you have added that item two times to the cart using the add to cart button, but in the cart, only one entry for the item should be displayed with quantity as 2, so how you’ll test this?  
14. Numbers 1 to 100 will be flashed on the screen only once, and you must find the missing number.  
15. I have a table and want to store all table data; which collection should be used and why?  
16. What will HashMap return?  
17. How do we achieve inheritance without having an interface?  
18. Where in the framework have you used Method overloading and method overriding?  
19. If I want my class not to be extended and another class cannot create an instance, how do I declare a class?  
20. How do you store multiple values in one reference?  
21. In cucumber, in which class do you have gluecode, how many classes for gluecode, and what was the program line limit for the class?  
22. How do we find missing implementation in Cucumber?  
23. Used static data or dynamic data, what was the approach?  
24. How do you resolve conflicts while pushing code in Git?  
25. Write a program to merge two sorted arrays.  
26. Input: a[] = {1,3,5,7,9} b[]={2,4,6,8}  
27. Output: C[]={1,2,3,4,5,6,7,8,9}  
28. What is Shift Left testing, and where do we use it?  
#QaFox  
credit: Arun Motoori

Insufficient balance  
  
Expiry card  
Credit card limit  
Repeated transactions  
Payment decline  
Timeout  
  
Expiry coupon  
Future date coupons  
Reused coupons  
Using 2 coupons in one  
Check stores where it’s only applicable

API Testing Rest Assured  
  
Postman support both soap and rest api?  
  
What is 2 tier arch  
  
What is API in local and how it will be look like  
  
  
API testing also called as Backend testing  
Without even UI we can do this testing  
  
Tool used to do API testing  
Postman (manual)  
Automation API testing tools  
Rest API  
SOAP UI  
  
Client: Application we have in machine, visible to us  
Server: where Application is available or hosted  
Consist of basic code part to do actions  
Program logic and database  
In application we raise request , it will go to server and get the response  
  
1Tier Architecture  
Client and server in Same machine  
Example: Notepad and system apps  
  
2 Tier Architecture  
App in one place and server in another place  
Eg, Zomato and google map  
2 app access s same server  
  
Different client application accessing the same server  
  
3 Tier Architecture  
Client Tier (App UI)  
  
Business logic(App layer which contains program for validation)  
  
Database Tier (layer holds data)  
Here server is divided into two  
Program logic and Database  
  
Request will be sent from client to business logic  
Business logic will send the request and receive the response from database and share to the client  
  
  
API - Application programming interface  
  
API acts a a mediator that communicates between Front end and Backend  
  
Front end application - web testing  
|  
Middle Layer (API server) - API testing  
|  
Back end - Database testing  
  
Without front layer (UI app) we can do API testing  
  
We can directly connect with API layer and get response  
We can start testing before UI is developed  
  
We can do testing in application(API) layer  
  
API in local is API  
API published in Internet is Webservice  
  
API made to public is accessible by any application like Google Map, payment apps  
  
Http - not secure without encryption  
Https- secure with encryption  
  
Encryption and decryption means  
The data we will pass will be transcript and go to server  
Again it will transcript and come back to us  
  
Host path resource  
In url without https it’s URI  
Complete url is url  
Resource name is URN

**On JUL 13 UPTO THIS COVERED**

 Most Asking TestNG Questions for SDET Role   
  
 1. What is TestNG?  
Be ready to explain what TestNG is and its key features. Describe how it facilitates test execution, reporting, and test configuration.  
  
 2. TestNG Annotations:  
Understand the different annotations provided by TestNG, such as @Test, @BeforeSuite, @BeforeClass, @BeforeMethod, @AfterMethod, etc. Be prepared to explain when and how to use them in test scripts.  
  
 3. TestNG Data Providers:  
TestNG offers data providers to feed test data dynamically. Familiarize yourself with how to use data providers and handle test data from external sources like Excel or databases.  
  
 4. TestNG Assertions:  
Be ready to discuss different assertion methods provided by TestNG, such as assertEquals, assertTrue, assertFalse, etc. Understand how to use them effectively in test scripts for result verification.  
  
 5. TestNG Test Suites:  
Explain how to create and execute test suites using TestNG. Understand how to group tests, define dependencies, and prioritize test cases within a test suite.  
  
 6. TestNG Listeners:  
TestNG allows the use of listeners for customizing test execution. Familiarize yourself with different listeners provided by TestNG, such as ITestListener, ISuiteListener, IInvokedMethodListener, etc. Be prepared to explain when and how to use listeners.  
  
 7. TestNG Reporting:  
Understand TestNG s built-in reporting capabilities and how to generate meaningful test reports. Explain the different report formats available and how to customize reports for better readability.  
  
 8. TestNG Parallel Execution:  
Be ready to discuss how TestNG supports parallel test execution, including methods for parallelizing tests at the suite, class, or method level. Explain the benefits and challenges of parallel test execution.  
  
 9. TestNG Data-driven Testing:  
TestNG supports data-driven testing using various approaches such as  
DataProviders, Excel files, CSV files, etc. Understand how to implement data-  
driven tests and explain the advantages of this approach.  
  
 10. TestNG Dependencies:  
Be prepared to discuss how TestNG handles test dependencies, allowing tests to be executed in a specific order based on their dependencies. Understand how to define dependencies using annotations or XML configuration.

Why Java 8 is Still Widely Used Despite Newer Versions.  
  
Java 8 remains popular even with many newer versions available. Here’s why many companies continue to use it:  
  
1. Stability and Maturity :  
Java 8 has been around since 2014 and has proven to be stable and reliable. Its extensive testing and long-term support (LTS) have made it a trusted choice.   
  
2. Backward Compatibility :  
Java 8 introduced significant features while maintaining backward compatibility with previous versions, making it easier for companies to upgrade without breaking existing code.   
  
3. Key Features :  
Java 8 introduced several powerful features that significantly improved the language, including:  
  
- Lambdas and Streams : Simplified code and enabled functional programming paradigms.   
  
- Optional : Enhanced null handling to avoid `NullPointerException`.  
  
- New Date and Time API : Replaced the outdated `java.util.Date` and `Calendar` classes.   
  
Main Usage and Features of Java 8  
  
Apart from the features already mentioned, here are other key features that make Java 8 a popular choice:  
  
- Default Methods in Interfaces : Allowed interfaces to have method implementations, enabling more flexible and reusable code without breaking existing implementations.  
  
- CompletableFuture API : Simplified handling of asynchronous programming  
  
Common Projects Built with Java 8  
  
Java 8 is used in various types of projects across different industries:  
  
- Enterprise Applications : Many large-scale enterprise applications rely on Java 8 due to its stability and mature ecosystem. Companies in finance, healthcare, and e-commerce often use Java 8 for backend systems.  
  
- Web Applications : Java 8’s improvements in concurrency and the introduction of Streams API make it suitable for building robust and scalable web applications.  
  
- Big Data Solutions : Java 8’s functional programming features and improved performance are advantageous in big data processing frameworks like Apache Hadoop and Apache Spark.  
  
- Microservices : The enhancements in Java 8, such as CompletableFuture and the new Date and Time API, make it a good fit for microservices architecture, enabling better asynchronous programming and date/time handling.  
  
Conclusion  
  
While newer versions of Java offer additional features and improvements, Java 8 remains a popular choice due to its stability, extensive feature set, and backward compatibility. Its introduction of significant enhancements like lambdas, streams, and a new Date and Time API have left a lasting impact, making it a reliable and powerful option for various types of projects.

Manual testing  
  
Testing ensures the software quality  
  
SDLC - model - Approach  
  
BRS. Business  
FRS functional  
SRS system requirement specification  
  
System.exit  
Inside try block will won’t allow finally block to execute  
As it exit the execution  
This is only condition where finally block won’t execute

Encapsulation For AutomationQA & SDET  
  
 What is Encapsulation?  
  
Encapsulation is one of the fundamental principles of Object-Oriented Programming (OOP) in Java. It is the mechanism of wrapping data (variables) and code (methods) together as a single unit, known as a class. The main idea behind encapsulation is to hide the internal implementation details of an object from the outside world, and provide a well-defined interface to interact with the object.  
  
 Give an example from Automation ?  
  
public class WebDriverManager {  
private static WebDriver driver;  
  
private WebDriverManager() {  
// Private constructor to prevent instantiation  
}  
  
public static WebDriver getDriver() {  
if (driver == null) {  
// Create a new instance of the WebDriver  
System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");  
driver = new ChromeDriver();  
}  
return driver;  
}  
  
public static void quitDriver() {  
if (driver != null) {  
driver.quit();  
driver = null;  
}  
}  
}  
  
In this example, the WebDriverManager class encapsulates the WebDriver instance, which is a core component of the Selenium WebDriver API. The driver variable is declared as private static, meaning it can be accessed within the class, and there will be only one instance of the WebDriver shared across the entire test suite.  
  
The class provides two static methods:  
  
getDriver():  
This method returns the instance of the WebDriver. If the instance doesn't exist, it creates a new instance of the Chrome WebDriver (ChromeDriver). This method ensures that there is only one instance of the WebDriver at any given time, following the Singleton design pattern.  
  
quitDriver():  
This method is responsible for closing the WebDriver instance when it's no longer needed, ensuring proper cleanup of resources.  
  
  
 Why do we need encapsulation?  
  
  
Data Hiding:  
By making the instance variables private, the internal state of an object is hidden from the outside world, preventing direct access and unintended modifications.  
  
Code Reusability:  
Encapsulated classes can be easily reused in other parts of the program or in different applications, as they provide a well-defined interface to interact with the object.  
  
Data Integrity:  
By controlling access to the internal state through methods, encapsulation helps in maintaining data integrity by enforcing validation rules or business logic before modifying the data.  
  
Code Maintainability:  
Encapsulation is a fundamental concept in Java and OOP, as it helps in writing clean, maintainable, and reusable code by separating the implementation details from the external interface.

Why PlayWright is better then Selenium?  
  
I'll list down some straight forward advantages that PlayWright has over Selenium:  
  
1. No special code to handle browsers like selenium  
2. Storage state  
3. Built in Reporting and screenshots  
4. Retries are made easier  
5. Auto waiting and retry assertions  
6. VScode extension  
7. Isolated sessions  
8. Simple integration with GitHub Actions  
9. UI Mode and Trace Viewer  
10. Does API Testing and is great at it  
13. Grouping tests using a Describe Block  
14. Accessibility Testing  
15. Not having to manage drivers for the Browsers  
16. Infinitely better Locator strategies  
  
Now, let's breakdown if PlayWright framework is really better then Selenium Framework! :  
  
Reporting supported by PlayWright is only available with JS, except this I think all other pros mentioned are far better in PlayWright over Selenium.  
  
But what you do? If you are looking for a new job?  
  
Learn Selenium or Adopt Selenium as its a go to!  
  
Why?  
  
Because most of the teams have already invested heavily on its Framework setup and execution cycles.  
  
P.S. Do share your thoughts if by any means my analogy is incorrect :)  
  
-x-x-  
  
Crack Test Automation Interviews with Java coding: <https://lnkd.in/g5hr9bea>  
  
Become a SDET and Future SDET Manager + 950+ SDET Interview Prep Q&A Bank: <https://lnkd.in/gusymgFi>  
  
Read my Technical blogs: <https://lnkd.in/gCC34Vv2>  
  
#japneetsachdeva

Explain the differences between functional and non-functional testing.  
2. What is the role of an SDET in an agile development process?  
3. How do you approach test automation for a new project?  
4. Describe your experience with continuous integration and continuous deployment (CI/CD) tools.  
5. How do you handle flaky tests in your automation suite?  
6. What programming languages are you proficient in for test automation?  
7. How do you design a test plan for a complex application?  
8. Explain the concept of test-driven development (TDD).  
9. What is the difference between white-box and black-box testing?  
10. How do you prioritize test cases in a test suite?  
11. Describe a challenging bug you found and how you resolved it.  
12. What are some best practices for writing maintainable test scripts?  
13. How do you ensure test coverage and what tools do you use to measure it?  
14. Explain the use of mocking in unit testing.  
15. What is the importance of code reviews in the context of test automation?  
16. How do you stay updated with the latest testing tools and techniques?  
17. What is the role of performance testing, and how do you conduct it?  
18. Describe your experience with API testing.  
19. How do you manage test data for automated tests?  
20. Explain the process of debugging a failing test case.  
21. What is the difference between integration testing and system testing?  
22. How do you ensure the security of your test environment?  
23. What strategies do you use for effective collaboration with developers and other stakeholders?  
24. Explain how you handle testing in a microservices architecture.  
25. What are some common challenges in test automation and how do you overcome them?  
26. Do you design a test automation framework from scratch?  
27. Explain how you would test a distributed system with multiple microservices.  
28. How do you handle test dependencies and ensure test isolation in your test suites?  
29.Describe a situation where you had to optimize your test suite to reduce execution time significantly. What approaches did you take?  
30. Tell me would you approach testing a system with high concurrency and potential race conditions?  
31.Explain how you would integrate security testing into your CI/CD pipeline.  
32.What strategies do you use for testing applications that heavily rely on third-party APIs?  
33. Do you ensure the reliability and scalability of your automated tests?  
Describe your approach to handling and testing large datasets.  
34. Do you deal with the challenge of testing in a cloud environment with dynamic infrastructure?  
35.Explain the process of migrating from a legacy test automation framework to a new one.

Java  
  
JDK - Java development kit  
|  
JRE - Java runtime environment  
|  
JVM - Java virtual machine  
  
JVM runs the program it will look for main method  
  
Concatenation operator + ( inside sysout)  
  
Primitive and non primitive data type  
  
Non primitive- using data type we will create object to access and we can store multiple data  
Primitive- Directly assigned to variable, we can store only one value for a reference  
  
Identifiers  
Should start in alphabet or \_ or $  
Cannot start with numeric or any other symbols  
Identifiers cannot take the key words which is predefined in Java  
Identifiers are case sensitive  
  
Operators:  
  
The default value of a boolean data type is false .  
  
Default Values  
￼  
  
Any property that performs more than one function is method overloading  
  
Scanner s = new Scanner(System.in)  
  
String variable  
Variable is also an object that can call different methods  
  
Java string pool in heap memory stores string  
  
String is immutable  
Objects are instance to the class  
  
Instance variables  
Controlled by objects  
Each object we create will have a separate copy of instance variable  
  
Stream is a channel between input and output to flow  
IO devices  
Input and Output devices  
  
Null pointer exception  
Array  
String  
Object  
  
  
CTA  
Call to function  
  
Xpath travel in both directions forward and backward  
CSS moved only forward  
CSS is fast than Xpath  
  
CSS selector  
# - ID  
  
  
Java archive - JAR  
Web archive - WAR  
Enterprise archive - EAR  
  
In Java, declaring a class as static is only applicable to inner classes. If an inner class is declared as static, it can be instantiated without needing an instance of the enclosing class. This means the inner static class can access the static members of the outer class directly. However, you cannot declare a top-level class as static. Here is an example:  
  
```java  
public class OuterClass {  
static class StaticInnerClass {  
// Static inner class code  
}  
}  
```  
  
In this example, `StaticInnerClass` can be instantiated without an instance of `OuterClass`.

This was the most requested Q&A bank in my DMs, It covers Selenium WebDriver, APIs, Postman, TestNG, Git  
  
APIs:  
  
1) What are the components of an HTTP request?  
2) What is the difference between API and unit testing?  
3) What is an HTTP response?  
4) How we can add validation points in postman?  
5) What do you understand by Server-side validation?  
6) What is 3 tier Architecture?  
7) Difference between webservice & APIs  
8) What is Rest, Soap & GraphQL in APIs  
9) What do you test in standalone API?  
10) What do test in 3rd party integrated APIs?  
  
Postman:  
  
1) When to use collection, environment & global variables  
2) How to execute a collection E2E  
3) How to validate a API response has correct status code?  
4) What happens when API response returns Form Data instead of Json, how to validate it?  
5) How to setup Basic Auth in Postman?  
6) Where do you store Environment credentials?  
7) How to save a demo response for an API request?  
8) How will you validate an API rqst if VPN is required for it to work?  
9) How do you filter results in a API request using Postman?  
10) How to setup custom headers in Postman?  
  
Selenium WebDriver:  
  
1) What are different types of locators in Selenium?  
2) When do you use Xpath over CSS locators?  
3) How to get a specific value from a dropdown and reuse it in verifications?  
4) When do we use JavaScript Executors?  
5) Is it possible to validate Captcha using Selenium, if Yes how?  
6) What should be the ideal way to store data using Selenium WebDriver only?  
7) Is it possible to use Xpath like parent/child/node/.. ? If yes, then provide an example where to use it?  
8) What happens if you receive browser notifications in between Test Automation Execution?  
9) Why does Stale Element exception occurs and how to handle it?  
10) What is Invalid Certificate Exception?  
  
Git:  
1) What are the different stages in committing the code to GitHub?  
2) Is it possible to revert back changes in a remote repo? If yes, how?  
3) When do you commit your code? After commit how do you validate everyone has the updated code now?  
4) How to merge stashed changes in local repo?  
5) Why do we need GitIgnore file? How to add ignore files in it?  
  
TestNG:  
  
1) What is the execution format of Tests in TestNG?  
2) Can priority be negative in for methods? If Yes, then what is the execution flow as per priority?  
3) What is the difference in DependonMethod & dependsongroups?  
4) What are different ways to exclude tests in TestNG  
5) What does ThreadPoolSize mean in TestNG? How does it work?  
6) Why do we need BeforeSuite & AfterSuite annotations?

Guide to breaking down Cucumber for SDETs  
  
Cucumber in QA automation is a tool that facilitates behavior-driven development (BDD) by allowing collaboration between developers, testers, and non-technical stakeholders  
  
1. Installation: Install Cucumber in your project by adding the necessary dependencies based on your programming language. For Java, use Maven or Gradle  
  
2. Understanding Gherkin Syntax: Learn the Gherkin syntax used in Cucumber feature files. Gherkin is a plain-text language that describes the behavior of an application in a structured way  
  
3. Create a Feature File: Write a feature file (e.g., example.feature) using Gherkin syntax. Define the feature, scenarios, and steps that represent the behavior you want to test  
  
Feature: Example Feature  
Scenario: Login with valid credentials  
Given the user is on the login page  
When the user enters valid username and password  
Then the user should be logged in successfully  
  
4. Create Step Definitions:Create step definition files (e.g., StepDefinitions.java for Java) to map Gherkin steps to executable code. Define methods for each step  
  
public class StepDefinitions {  
@Given("the user is on the login page")  
public void navigateToLoginPage() {}  
  
@When("the user enters valid username and password")  
public void enterValidCredentials() {}  
  
@Then("the user should be logged in successfully")  
public void verifyLoginSuccess() {}  
  
5. Tagging Scenarios: Use tags to organize and selectively run scenarios. Add tags to scenarios in the feature files and specify them when running tests  
  
6. Parameterisation: Use scenario outline tables and examples for parameterisation to run the same scenario with different input values  
  
7. Hooks:Implement hooks to set up and tear down actions before and after scenarios or features  
  
8. Use Background: It is always the best practice to put steps that are repeated in every scenario into the Background. Background step executes before each scenario.  
  
Feature: I want to login into the site with valid and invalid data  
Background:  
Given I navigate to the Website  
Scenario: Login as a new sign-up user with valid data  
  
9. Use Data table: It’s recommended to use Data Table to store the data. We can give data as parameters within the step but once we have a large set of data.  
  
10. Scenario outline: A scenario outline allows you to run the same scenario with different sets of data. It uses the Examples keyword to provide a table of inputs and expected outcomes.  
  
-x-x-

TestNG is a Framework  
  
POM - page object model  
It used structure the project, by organising the code logics, it’s a design pattern.  
  
  
TestNG is used organise test cases efficiently and flexible  
We can do grouping and prioritising tests  
Supports parallel execution  
We can use XML file to manage suite level configuration and parameterisation  
We have listeners and reports to listen the test and prepare report  
  
Annotations and Attributes are two important things in TestNG  
  
Annotation customise the order of execution  
  
Attribute is a condition  
Dependency and grouping like this  
  
@Test(Enabled = True/False)  
Default is true  
  
@Test(alwaysRun = True/False)  
  
@Test(dataProviderClass = “name”)  
  
If data provider method in other class means  
@Test(dataProvider = “name”, dataProviderClass = ClassName.class)  
  
@Test(groups = “smoke”)  
  
@Test(groups = {“smoke” , “regression”})  
  
@Test(dependsOnGroups = {“smoke” , “regression”})  
  
@Test(dependsOnMethods= {“method1” , “method2”})  
  
@Test(expectedExceptions = IOException.class)  
  
@Test(invocationCount = 5)  
  
@Test(priority = 1)

SQL a day can keep interview rejection away.  
  
Yes practicing at least one question a day can help you crack any Data Engineering SQL interview.  
  
Consistency is the key!!  
  
𝐇𝐞𝐫𝐞 𝐚𝐫𝐞 𝐬𝐨𝐦𝐞 𝐨𝐟 𝐭𝐡𝐞 𝐛𝐞𝐬𝐭 𝐩𝐥𝐚𝐭𝐟𝐨𝐫𝐦𝐬 𝐭𝐨 𝐩𝐫𝐚𝐜𝐭𝐢𝐜𝐞 𝐒𝐐𝐋 𝐩𝐫𝐨𝐛𝐥𝐞𝐦𝐬 𝐝𝐚𝐢𝐥𝐲 -  
  
1. HackerRank  
2. LeetCode  
3. DataLemur  (Ace the SQL & Data Interview)  
4. 8 Week Sql Challenge <https://lnkd.in/dj9FdZk8>  
5. StrataScratch  
6. [sqlbolt.com](http://sqlbolt.com/)

It's been two years since I joined Zoho.   
  
I have been getting DMs asking how to prepare for Zoho and for a structured plan to crack it.  
  
I can't give anyone a structured plan. What I can do is share my preparation journey.  
  
- I think it was November 2020, when I got to know about the core idea of habits from a YouTube video summarizing the book "Atomic Habits."  
  
- So, I searched for that one habit that I could do every day, and that’s when I got to know about GeeksforGeeks, where every resource for interview preparation is available.  
  
- I decided to solve at least one DSA problem a day on GeeksforGeeks, no matter what.  
  
- Note that I didn’t know Java very well before starting this. I just read the problem statement and directly jumped into the solution editorial or some YouTube video explaining it.  
  
- Then, to add one extra habit, I took a web development bootcamp course on Udemy. It was a 65-hour course, so I decided to watch 1 hour of the course every day.  
  
- As days passed, as I never missed a streak for 6 months, the compounding effect worked. Solving easy and easy-medium DSA problems became like a cakewalk.  
  
- After completing that web development bootcamp course, I got some idea of web development and learned MongoDB, ReactJS, NodeJS, GIT, SQL etc.  
  
- Since solving DSA problems had become like a game for me, I solved problems on all the platforms I could find. Some of them are LeetCode, HackerRank, HackerEarth, etc.  
  
- I also tried solving some problems on CodeChef and Codeforces. I mostly failed to solve or even understand most of the questions except the first question of the contest, which was easy.  
  
- I also binge-watched YouTube videos related to web development and DSA every day.  
  
- Then, Zoho came to our college to hire summer interns.  
  
- As Zoho essentially tests our problem-solving skills and critical thinking, I was able to crack it.  
  
This is my journey of interview preparation. What I learned is that when we consistently do something early enough, we don't need to prepare when it's time. 

ostly Asked Selenium Interview Questions:  
  
1. What is Selenium?  
→ Selenium is an open-source automation testing tool designed for web application testing.  
  
2. Explain different Selenium components.  
→Selenium WebDriver, Selenium IDE, and Selenium Grid are the main components.  
  
3. What is Selenium WebDriver?  
→ Selenium WebDriver is a web automation framework that allows you to interact with web elements and perform actions on web applications.  
  
4. How do you locate elements in Selenium WebDriver?  
→ Elements can be located using methods like ID, Name, XPath, CSS Selectors, and more.  
  
5. What is the difference between findElement() and findElements() in Selenium?  
→findElement() is used to identify a single element present in a webpage whereas findElements() is used to identify multiple elements present in a window. findElement() returns the first matching element, while findElements() returns a list of all matching elements.  
  
6. What is a WebElement in Selenium?  
→ A WebElement is an interface in Selenium that represents an HTML element on a web page.  
  
7. Explain the difference between Implicit Wait and Explicit Wait.  
→Implicit Wait sets a global timeout for all elements, while Explicit Wait is applied to specific elements with a custom timeout.  
  
8. What is the Page Object Model (POM) in Selenium?  
→POM is a design pattern that helps to create an object repository for web elements.POM improves code readability and reusability. It helps to reduce code duplication and improves test case maintenance.  
  
9. How do you handle pop-up windows in Selenium?  
→ You can use the switch To method to handle pop-up windows, alert boxes, and frames.  
  
10. What is TestNG, and why is it used in Selenium?  
→TestNG is an automation testing framework that facilitates test configuration, parallel execution, and reporting in Selenium.  
  
11. Explain the difference between driver.get() and driver.navigate().to()?  
→Both methods navigate to a URL, but driver.navigate().to() allows for forward and backward navigation within the browser history.  
  
12. How can you simulate mouse actions in Selenium?  
→ Actions class in Selenium is used to simulate mouse actions like click, drag and drop, and context-click.  
  
13. What is the purpose of the Selenium Grid?  
→Selenium Grid is used for parallel test execution on multiple machines and browsers.  
  
14. Explain the difference between driver.close() and driver.quit)?  
→driver.close() closes the current browser window while driver.quit() closes the entire browser session.  
  
15. What is Apache POI?  
→Apache POI is the most popular Java library/API used to interact with Microsoft Excel Sheets. It is used to read data from Excel sheets and write data into Excel sheets.

Rest Assured- Interview Questions  
  
1) Difference between Path and Query Parameters with an example  
  
2)How to send a GET request using Rest Assured?  
  
3)How to log a response in Rest Assured only in the case of an error.  
  
4) Explain different ways of extracting a single field from a response body. [like using response, JSONPath, XMLPath) and also they will give you the response of a request and ask you to extract the response of a particular field.  
  
5) How to mask header information in API testing using Rest Assured?  
  
6)How to download a file using rest assured?  
  
7)How do you handle form parameters and multipart parameters [uploading media files]?  
  
8) They will give you an end to end scenario and ask how you will write the rest assured code for that they are trying to understand how well you can do the API chaining here, you can just explain also]  
  
9) What import statement will you use for Rest Assured to work?  
  
10) How to check that a specific item is present in a collection using Rest Assured? [we can use Matchers here  
  
11)What are the common exceptions you encounter in Rest Assured?  
  
13) How do you handle data in Rest Assured? [POJO, Excel, config file, HashMaps]  
  
14) What is the use of ResponseSpecification in Rest Assured?  
  
15)How do you handle authentication and authorization in Rest Assured tests? [basic, oauth, digest.custom]  
  
16) What are the common pitfalls or challenges you have faced while using Rest Assured, and how did you overcome them?  
  
17)What is the difference between given(), when(). and then() methods in Rest Assured and explain with an example.  
  
18)How do you handle cookies in Rest Assured tests?  
  
19)How can you handle timeouts and retries in Rest Assured tests?  
  
20) Reporting in Rest Assured.  
  
21) How do you enable parallel execution of Rest Assured tests? [TestNG, XML]  
  
22)How do you verify the status code of an HTTP response using Rest Assured?  
  
23)How do you handle dynamic status codes or scenarios where the status code may change between test runs?  
  
24)How can you handle dynamic data or parameters in Rest Assured requests?  
  
For more such information content follow me!

Here are commonly asked questions about the Java Collection Framework:  
  
1. What is the Java Collection Framework?  
  
2. Explain the purpose and features of the Java Collection Framework.  
  
3. What are the main interfaces in the Collection Framework?  
  
4.Discuss the key interfaces such as List, Set, Map, Queue, and their implementations. Difference between List, Set, and Map?  
  
5. Explain the differences between List, Set, and Map interfaces in terms of characteristics and use cases.  
  
6. What is the difference between ArrayList and LinkedList?  
  
7. Compare the characteristics and performance of ArrayList and LinkedList in terms of insertion, deletion, and traversal operations.  
  
8. What is the difference between HashSet and TreeSet?  
  
9. Compare the characteristics and use cases of HashSet and TreeSet in terms of ordering, uniqueness, and performance.  
  
10. Explain the HashMap and HashTable differences?: Discuss the differences between HashMap and HashTable in terms of synchronization, null key and values, and iterator fail-fast behavior.  
  
11. What is the Comparable and Comparator interface?  
  
12. Explain the purpose of Comparable and Comparator interfaces and how they are used for sorting objects in Java.  
  
13.Explain the Iterator and ListIterator?  
  
14.Discuss the differences between Iterator and ListIterator interfaces and their usage in traversing collections.  
  
15. What is the difference between fail-fast and fail-safe iterators?  
  
16. Explain the concepts of fail-fast and fail-safe iterators and give examples of collections that support each.

Basic Java coding interview questions for QA  
  
1. String reverse. Write a method that will take one string as an argument and will return the reverse version of this string.  
  
2. Array reverse. Write a method that will take an array as an argument and reverse it.  
  
3. Reverse words. Write a method that will take a string as an argument. The method will reverse the position of words and return it.  
  
4. String palindrome. A palindrome is a word, phrase, number, or sequence of words that reads the same backward as forward.  
  
5. Number palindrome. A palindrome is a word, phrase, number, or sequence of words that reads the same backward as forward.  
  
6. Max/min number from an array. Write a method that will accept an array of int as an argument and it returns the max/min number from a given array.  
  
7. Find the second min/max number from an array. Write a method that accepts int array as an argument and returns second or n min/max number from the given array.  
  
8. Static keyword in Java. The static keyword is a very popular question in the interviews.  
  
9. String Pool and == operator to compare references in Java. Let's see an example, what's output from this program? Why this output?  
  
10. Swap values of two variables without direct reassignment and without creating any extra variables.  
  
11. Two string anagram. An anagram is when all the letters in one string exist in another but the order of letters does not matter. Write a method that accepts two string arguments and returns true if they are anagram and false if they are not.  
  
12. Remove duplicates from a string. Write a method that accepts one string argument and returns it without duplicates.

Mostly Asked Selenium Interview Questions:  
  
1. What is Selenium?  
→ Selenium is an open-source automation testing tool designed for web application testing.  
  
2. Explain different Selenium components.  
→Selenium WebDriver, Selenium IDE, and Selenium Grid are the main components.  
  
3. What is Selenium WebDriver?  
→ Selenium WebDriver is a web automation framework that allows you to interact with web elements and perform actions on web applications.  
  
4. How do you locate elements in Selenium WebDriver?  
→ Elements can be located using methods like ID, Name, XPath, CSS Selectors, and more.  
  
5. What is the difference between findElement() and findElements() in Selenium?  
→findElement() is used to identify a single element present in a webpage whereas findElements() is used to identify multiple elements present in a window. findElement() returns the first matching element, while findElements() returns a list of all matching elements.  
  
6. What is a WebElement in Selenium?  
→ A WebElement is an interface in Selenium that represents an HTML element on a web page.  
  
7. Explain the difference between Implicit Wait and Explicit Wait.  
→Implicit Wait sets a global timeout for all elements, while Explicit Wait is applied to specific elements with a custom timeout.  
  
8. What is the Page Object Model (POM) in Selenium?  
→POM is a design pattern that helps to create an object repository for web elements.POM improves code readability and reusability. It helps to reduce code duplication and improves test case maintenance.  
  
9. How do you handle pop-up windows in Selenium?  
→ You can use the switch To method to handle pop-up windows, alert boxes, and frames.  
  
10. What is TestNG, and why is it used in Selenium?  
→TestNG is an automation testing framework that facilitates test configuration, parallel execution, and reporting in Selenium.  
  
11. Explain the difference between driver.get() and driver.navigate().to()?  
→Both methods navigate to a URL, but driver.navigate().to() allows for forward and backward navigation within the browser history.  
  
12. How can you simulate mouse actions in Selenium?  
→ Actions class in Selenium is used to simulate mouse actions like click, drag and drop, and context-click.  
  
13. What is the purpose of the Selenium Grid?  
→Selenium Grid is used for parallel test execution on multiple machines and browsers.  
  
14. Explain the difference between driver.close() and driver.quit)?  
→driver.close() closes the current browser window while driver.quit() closes the entire browser session.  
  
15. What is Apache POI?  
→Apache POI is the most popular Java library/API used to interact with Microsoft Excel Sheets. It is used to read data from Excel sheets and write data into Excel sheets

selenium architecture  
  
Package with any language  
Protocol (json wire protocol, W3C protocol)  
Browser driver  
Real browser  
  
The difference between Selenium-3 and Selenium-4  
1st,  
So in selenium 3, so we have used JSON wire protocol, but however, in selenium 4 we are using this W3C protocol  
  
2nd,  
In selenium 3, the chrome driver was extended a by Remote WebDriver directly, but however, in selenium 4 the chromium driver extends the Remote web driver, and the chrome Driver & Edge Driver extends the chromium driver  
Chromium Driver is the browser engine for many browsers  
  
3rd,  
Optimised Selenium Grid in Selenium 4 ::  
Unlike Selenium 3, testers would no longer be required to start the hub and node jars every time they want to perform automation testing. In Selenium 4, hub and node are packed in a single jar file. Selenium Grid 4 architecture supports four processes—Session Map, Node, Router, and Distributor. Selenium Grid 4 has more scalable and traceable infrastructure. There are some additional perks like enhanced GUI and built-in support for Docker.  
  
4,  
Selenium 4 IDE is available for Firefox and Chrome browser. It is more than just a record and playback testing tool. There is a side-runner tool which allows us to run Selenium tests parallely on local Selenium Grid and cloud-based Selenium Grid. They have also improved the GUI for a better user experience.  
  
5. Relative locators  
Relative locator has been introduced in selenium 4 like (above, below, towriteof, toleftof, near )  
So we can choose one locator, and we can locate different locator using the above commands  
But selenium 3, we have to use a series of find commands to locate the surrounding elementsc  
Selenium uses a Java script function getBoundingClientRect() to determine the size and positions of element on a page and uses this information to locate the neighbouring element  
  
6, aChrome DevTools  
  
We can manipulate the geographical location Like if you want to test how your people from different countries use this website by using this function we can point to that geographical location and we can access the application  
Also, we can test in various network conditions like 2G 3G and 4G  
  
7,  
In selenium 4, decidedcapabilities class has been replaced by options class  
So this function helps us to use different browser versions to test application in different browser versions And also OS versions  
  
8,  
So new method has been introduced in action class  
ContextClick()  
Click()  
ClickAndHold()  
doubleClick() and  
release()  
  
HTML  
HyperText Markup Language is the code used to structure a web page and its content  
  
So basically HTML structures the web page. So how a content should be visible. So what are the contents should be added to a particular function. Since we have like n number of elements in HTML. So we can use those things to customize a web page. So we have different tags for different function. Like a text in a web page or if you want to add a image. Also if you want to give some heading or a link. So for each and every different contents in a web page we have different tag fnames. So we can enclose such inputs in between those tag names to get the customized web output. So HTML consists a series of elements which we can use to structure our web page with the desired output.  
Begins and closes with html tags  
DOM HTML is made of tags and attributes  
  
Element  
  
so element so basically element is a part of a web page so whatever we see in a web page so we call them as a function or element. In XML or HTML element can contain some data some text or some image any item it can contain or else it can also be empty it's also considered as a element so a customizable or a typical element includes so a tag a opening tag which consists of some attributes and also some content either text or a link and the particular element tag will be closed so it will also have a closing tag at the end  
  
We use the attribute inside the tags to access the element or function  
  
For testing purposes we can add dummy attributes into a tag name which doesn’t affect the function  
  
Attribute contains  
Attribute name and value  
  
Nested tags  
Tag and tag  
Both tags point to same function  
  
CSS  
CSS, Cascading Style Sheets, is the code that styles web content. It is used to decorate my webpage with background images and colors. Like HTML, CSS is not a programming language, it's not a markup language either. CSS is a style sheet language.  
  
Structure tags  
<html> The root of the HTML document specifies it as HTML.  
  
<head> Contains head elements such as title, style, and meta tags in the HTML file.  
Also contains all scripts  
  
<title> defines the title of HTML document  
  
<body> defines the body of an HTML document containing content like images, tables, and lists, etc.  
  
  
HTML tags ::  
  
<div> Define a division or a section in an HTML document.  
  
So division means so we can group some of elements under division tag. So we can have n number of division under one division tags. So in a webpage we can divide the webpage into different division. So each division can also have n number of divisions. So depends on the functionalities..  
  
<span> So, under div, we can use span tags to group certain elements  
Defines , a generic in-line container  
  
Uses  
Maven - xml  
Gradle - groovy scripting  
  
Interface having only one method is functional interface  
  
CSS selector only travel in forward (Top to bottom)  
Xpath can travel both forward and backward.  
  
Comparing both CSS selector is fast  
  
For CSS selector  
To use ID we have to use # before value  
Like if Id = home  
Then #home  
  
Navigate-to  
Navigate-forward  
Like these methods coming from  
WebDriver-Navigation  
| |  
Interface Class  
  
Moving kerosal  
Some dynamic images or links  
Like the top content in hotstar which will have 5 highlights which will display sequential like presentation  
  
Page load means it will wait for complete html DOM to load  
Default time is 300seconds  
  
Script load - 30 seconds  
  
To get time  
System.currentTimeMillis()

22/04 - DOM  
HTML CSS

Why we need software ?  
Cut down the cost  
Improve revenue by reaching customers  
To sustain in market by adopting technology  
Migrating to different technologies  
  
SDLC methodologies  
One of the traditional method  
Waterfall (Linear sequential method)  
V model (Validation or verification)  
  
Prototype model  
Develop in a rapid manner.  
Dummy model , initial model , rough draft  
If new things added after original requirement is signed that’s is called change request  
SOW - Statement of work  
  
SRS- System requirement specification  
  
The Spiral model  
Combo of spiral and Iterative  
  
Iterative model or Incremental model  
Project is broken to lot of small pieces  
  
Big Bang model  
Time and material for billing  
Fixed bed for billing  
  
Agile model  
Agile- meaning Quickly and easily  
Sprint meaning fast  
  
Stages  
Client with give requirements which is  
Business Requirement Document BRD  
-Clients might have an IT team or not  
  
Client will give business requirements, But we use System Requirement means technical stuff other than Business stuffs  
-The person who converts BR to SR is Business Analyst / Business Management  
  
Once requirements are prepared  
Then We need Infrastructure to deploy the application which is physical server(ENV)  
Infrastructure is like a basements and pillar  
Which is took care by software Architect who is also called as Tech Lead who understands the complete infrastructure  
  
Then we need Design, Design for each module which will be done by Designers UI/UX Designers  
UI - User interface  
UX - User experience  
  
Then Start building a website  
Developers will work on coding and once a module is done they will test it  
  
Test environment/ Test Bed/ Sand box layer  
Which should be similar to production environment but load limit can be less  
  
Testing methodologies  
  
Levels / Types of testing  
—————————————————  
  
Agile methodology  
Scrum model  
  
Business stakeholders will give the requirement (Input)  
Product manager converts requirements to features  
Product owner converts same to user stories  
Scrum master is like manager for team he is managing everything  
Retrospective— Retro - Go back  
Inspect - do inspect  
  
Scrum call - Normal status call  
We discuss what we did yesterday  
What we will do today  
If Any road blockers  
  
Each user story will have lot of subtask in it  
  
Testing techniques  
Boundary value analysis  
Equivalence case partition technique  
Decision table based testing (Cause Effect table)  
State transition technique  
Error guessing technique  
Use case widely used in developing tests at systems or acceptance level  
  
Types of test case design techniques  
-Specification based  
-Structure based  
-Experience based  
  
  
Augmented reality is a mixed reality / Virtual reality  
This gives a AI kind of idea for all .  
  
  
Monolithic and micro services  
  
QA and QA  
  
RAID - Risks Assumptions Issues Dependencies  
  
Risk  
Mitigation- Avoidance  
Contingency- Issue happened and how to rectify  
  
  
Test Strategies- OTS - Organisational test strategy  
Responsive web design  
Types and level testing we have to concentrate  
Time and materials - Not Budget concern  
Fixed bed - Fixed budget  
  
Test planing and Information we give to clients  
List out all our activities & Strategy  
Requirements to carry out testing  
Deliverable  
RAID  
  
Exploratory testing purposes:  
End user will explore the application based on his knowledge  
Like a end user  
Other than requirements we will act like end user to find defects  
  
Adhoc testing  
Randomness  
Like in a payment page if we click refresh or  
back button then page can become unresponsive  
This will be not a good sign from user side  
To Avoid this do random testing  
  
Data center  
Contains servers and databases for a application  
  
Server storage technology  
Cloud computing  
  
Scaling  
Vertical scaling - increasing same server capacity  
Horizontal - Increasing no of servers ( renting servers)  
  
Types of cloud computing  
Iaas paas SaaS  
Load balancer will clear traffic and divert to server  
  
Compatibility testing  
Depends on hardware (device, mobile, system)  
Software ( Browser apps, OS platform )  
  
Contract testing  
Testing the ApI contact is done properly  
  
Browser stack  
Emulator and simulator  
Emulator- mobile  
Here we can use different systems, os to test our app  
  
  
Lambda test is competitive for browser stack  
  
Devops will monitor and visualise  
The application is availability  
Monitor give logs  
Visualise conforms availability  
  
Axe and pally are automated tools for accessibility testing  
  
Agile is flexible model  
PO decide the release  
  
Legacy website  
Legacy - old  
Some thing is available for long time/ old  
  
Product backlog  
Repository having User stories that yet to prioritise  
  
Epic  
|  
User Stories  
|  
Sub tasks  
  
Test scenario is high level action going to perform  
Not revealing much information (Abstract)  
  
Test case is low level action on how many ways we are going to perform a action  
  
SLA  
Standard level agreement  
Service level agreement  
  
Defect Triage  
Call to prioritise the defects  
  
3 Amigos  
PO  
Dev  
QA  
  
  
Java is Object Oriented  
It runs with object  
  
Echo - to create or to write  
American standard code - 9 14  
  
Concatenation operator +

Xoriant | QA Automation | Interview Qs  
  
1) Rate yourself out of 5 in Java, Selenium, and Cucumber.  
2) Tell me about yourself.  
3) What are your roles and responsibilities in your current/previous role?  
4) Can you explain the framework you've worked with?  
5) Explain the different ways to handle dropdowns in Selenium.  
6) Give a scenario where you need to click on Electronics in Flipkart using the Action class.  
7) What is Polymorphism in Java?  
8) What is an interface in Java?  
(May ask for deeper details based on your answer)  
9) What is Cucumber, and how does it fit in your testing strategy?  
10) What is the difference between Scenario Outline and Data Tables in Cucumber?  
11) Explain what Hooks are in Cucumber.  
12) What is Tagging in Cucumber, and how is it useful?  
13) What is the difference between Dry Run and Strict in Cucumber?  
14) Does Cucumber support TestNG?  
15) What is encapsulation? Provide an example with code.  
16) Write a Java program to find duplicate characters from a word without using for loops.  
17) Explain the use of an object repository in Selenium (.properties file).  
18) How do you store data in a .properties file?  
19) What is Page Factory in Selenium?  
20) What is the difference between @BeforeClass and @BeforeTest in TestNG?  
21) What are the tasks typically written under the @BeforeTest annotation?  
22) How can you skip a test case in Cucumber?  
23) How do you rerun failed test cases in TestNG?  
24) How do you group and run test cases in TestNG?  
25) What would you include under the @BeforeSuite annotation?  
26) Can you explain the different locators used in Selenium?  
27) What is a collection in Java?  
28) Explain the hierarchy of exceptions in Java.  
  
  
Stay curious, keep learning, and keep sharing  
  
  
Boost Your QA & Automation Skills with My Udemy Courses!   
Whether you're a beginner or looking to upskill, these courses cover everything from manual testing to advanced automation using the latest tools and frameworks. All content is designed with real-time examples, live projects, and hands-on exercises to help you become job-ready!    
  
##############################  
 Udemy Course Links   
##############################  
  
 Playwright with TypeScript   
 <https://lnkd.in/gUTACH7S>  
 Playwright with JavaScript   
 <https://lnkd.in/g2ZGeTKT>  
 Generative AI for Software Testing   
 <https://shorturl.at/Uk2eM>  
 Manual Testing + Agile with Jira Tool   
 <https://lnkd.in/gZgqAs4T>  
 Selenium with Java + Cucumber   
 <https://lnkd.in/gyDYWcgn>  
 Selenium with Python & PyTest   
 <https://lnkd.in/grHWFBDm>  
 Selenium with Python using Robot Framework   
 <https://lnkd.in/ginw2FhF>  
 API Testing (Postman, RestAssured & SoapUI)   
 <https://lnkd.in/fYs_i2n>  
 Web & API Automation using Cypress with JavaScript   
 <https://lnkd.in/gHXnAWBK>  
 JMeter - Performance Testing   
 <https://lnkd.in/gZv2jdhp>  
 SDET Essentials (Full Stack QA)   
 <https://lnkd.in/gyWHb2Xx>  
 Appium - Mobile Automation Testing   
 <https://lnkd.in/gzhXSybZ>  
 Java Collections   
 <https://lnkd.in/gPnvjYaB>  
 Cucumber BDD Framework   
 <https://lnkd.in/grW8K6GM>  
  
  
Framework AI  
Github link : <https://lnkd.in/gfQR_Anz>

Level Up Your Automation Testing Skills – FREE Full Courses on YouTube!  
  
Whether you're starting your journey in automation testing or looking to deepen your expertise, I’ve compiled a set of complete, high-quality training courses — all available for free on YouTube.  
  
These courses cover the most in-demand tools and frameworks used by QA professionals and SDETs worldwide:  
  
 Free Complete Courses:  
  
 JMeter Full Course – Master JMeter in 5 Hours  
 <https://lnkd.in/gH9pu6kZ>  
  
 Cypress.io Full Course – Master Cypress in 7 Hours  
 <https://lnkd.in/gHxuqsrt>  
  
 Selenium Python Full Course 2025 – Master Web Automation with Python  
 <https://lnkd.in/gq9xTh6N>  
  
 Postman API Testing Full Course – Master API Testing in 2 Hours  
 <https://lnkd.in/dc54Kpst>  
  
 Playwright with JavaScript Full Course – Master Playwright in 10 Hours  
 <https://lnkd.in/gkFKTWMc>  
  
 Selenium Java Full Course – Become an Automation Pro in 5 Hours  
 <https://lnkd.in/d_Gdjjfy>  
  
 Python for Testers Full Course – Master Python in 8 Hours  
 <https://lnkd.in/gzNAbBnw>  
  
 Building Cypress.io POM with Cucumber in One Hour  
 <https://lnkd.in/ghjg2xDr>  
  
 These are perfect for QA engineers, test automation professionals, and anyone preparing for real-world projects or interviews.  
  
  
Preparing for a Selenium Interview? Here's The Secret List of Interview Questions They Ask  
  
I had thought of keeping it a secret. But a number of posts with old questions popping now and then on LinkedIn made me worried about you.  
  
1. What is Selenium, and what was its land dispute with Mercury?  
2. How loudly can you scream "Selenium" before a Playwright user files a harassment complaint?  
3. Differentiate between Selenium 3 and 4—without saying “deprecated” or “I was on bench that year.” Be creative. Pretend you gave a crap.  
4. `findElement()` vs `findElements()`— Why is findElement() always the chosen one? What about the feelings of findElements() huh?  
5. Dropdowns: What do you do when <select> vanishes like your team lead pulled a “Sutta break, back in 5” and never returned during a bug triage.  
6. Frames and iframes: Why are Apple IFrames expensive whereas they create the same mess as Frames?  
7. Name all Expected Conditions in Selenium in the tone of someone still waiting for their PF transfer. Bonus points if you sigh audibly.  
8. Implicit vs Explicit Wait: Which one of them is recommended in a framework marked "Under Parental Guidance"  
9. Fluent Wait: If Fluent Wait’s so fluent, why’s your vocab still ‘yaar’ and ‘bro’?  
10. `StaleElementReferenceException`—What’s your go-to method to pretend it's not your fault if it goes unhandled?  
11. `JavaScriptExecutor`: Give an instance where you went full jugaadu with it.  
12. File uploads: which hack feels least illegal under India’s IT Rules 2021?  
13. Broken link validation: When would you stop using Selenium for this? Pick a date, any date, and swear you’ll be done by then.  
14. Capturing network logs: Which packet do you enjoy snooping the most—purely for “test coverage,” of course?  
15. Page Object Model: List five ways you’ve over-engineered it until your IDE begged for mercy.  
16. Page Factory: How many annotations per square inch before the Fire Department labels it a structural hazard?  
17. Dynamic elements: What’s your favourite XPath crime against humanity? Bonus if it includes a div[17].  
18. `DesiredCapabilities`: What are your desires in life? When are you going to build capabilities to fulfil those?  
19. TestNG Listeners: Wondering what is it doing in a Selenim question list? Take your time. Wonder for a few mins.  
20. Headless mode: Tell us why Ramsay brothers used it in Tahkhana.  
21. Authentication pop‑ups: Why do they pop-up rather than pop-down? What disciplinary action would you recommend?  
22. CAPTCHA handling: Tell us proudly about your experience of breaking it for functional test automation and not reporting it as a critical security issue.

|  |  |
| --- | --- |
|  | ReplyForward  Add reaction |