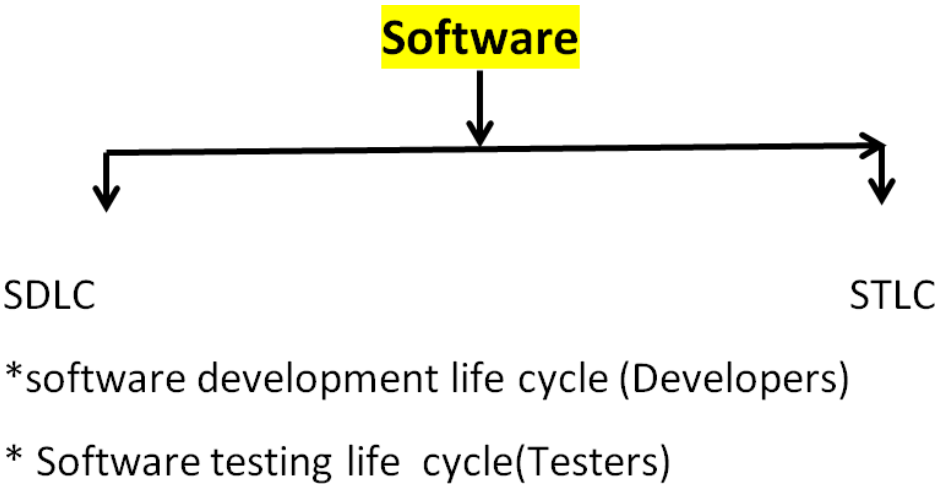


1. What is Software testing? What is the purpose of testing?

Software Testing is part of the software development process and verifying the Behavior or functionality of an application. Which is developed by developers and test by the testers meet the user requirements. The testers find the bugs and developers fixed those bugs, finally produced the quality of products to end-users or customers.

The purposes of testing are following:

- Quality assurance.
- Verification and Validation,
- To find the bugs before the product is released to customer.
- To improve the quality of the product
- The Purpose of Testing is to evaluate that the product is according to requirements.

Software testing types: Software testing is 2 types such as

1. Manual Testing 2. Automation Testing.

1) Manual Testing: Manual testing is the process of testing the software manually to find the defects. Tester should have the perspective of end users and to ensure all the features are working as mentioned in the requirement document. In this process, tester as execute the test cases and generate the reports manually without using any automation tools.

2) Automation Testing: Automation testing is the process of testing the software using an automation tools to find the defects. In this process, testers execute the test scripts and generate the test results automatically by using automation tools. Some of the famous automation testing tools for functional testing are **QTP/UFT and Selenium**.

2.Why there are bugs insoftware?

- Miscommunication or no communication
- Software complexity.
- Programing errors.
- Changing requirements
- Lack of skill testers Etc..

3. What is product and project?

- If software application is developed for specific customer requirement then it is called Project.
- If software application is developed for multiple customers requirement then it called Product.

Step 1: Software budding (project / product)

- **a proposal to develop new s.w**

Step2: Kick of meeting

- **Selection of P.O (project manager)**
- **Overall project plan**
- **Project / product initiation note (PIN)**

Step3: Business Analyst (BA)

- **Requirements gathering.**

Step4: System Analyst (SA)

- **Analysis and Detailed plan**

Step5: Technical Architecture (TA)

- **Designing (HLD high level designing and LLD low level designing)**

Step6: Coding

- **Development team (programmers)**

Step7: Testing

- **Testing team (testers)**

Step8: Release

- **Deployed code to production department**

Step9: Services / Maintains/Support

Software Development Life Cycle (SDLC)

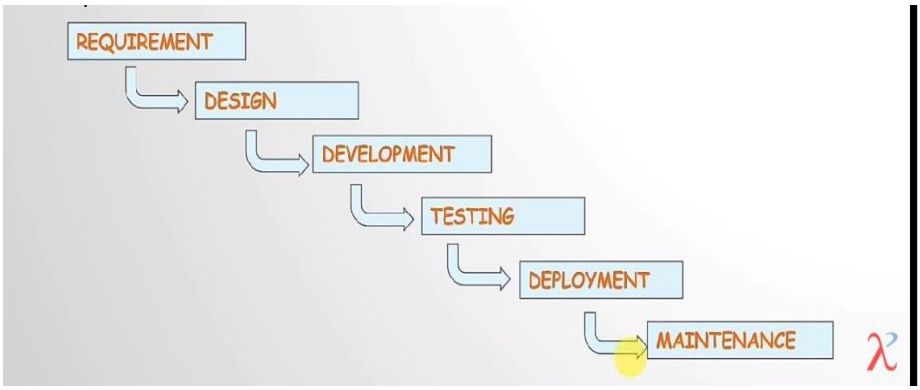
"SDLC stands for Software Development Life Cycle. SDLC is process used by software industry to design, develop and test high quality of software's. The SDLC aims to **produce high quality software and meets customer expectations**.SDLC defines the standard phases involved throughout the software development. These phases are

- **Requirement Analysis**
- **Project planning**
- **Project designing**
- **Development**
- **Testing**
- **Deployment**
- **Maintenance**

SDLC Models: 1.Waterfall Model 2. Incremental Model 3. Spiral model

4. V- Model and 5. Agile Model *.

1.*waterfall model: The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, **each phase must be completed before the next phase can begin and there is no overlapping in the phases.**



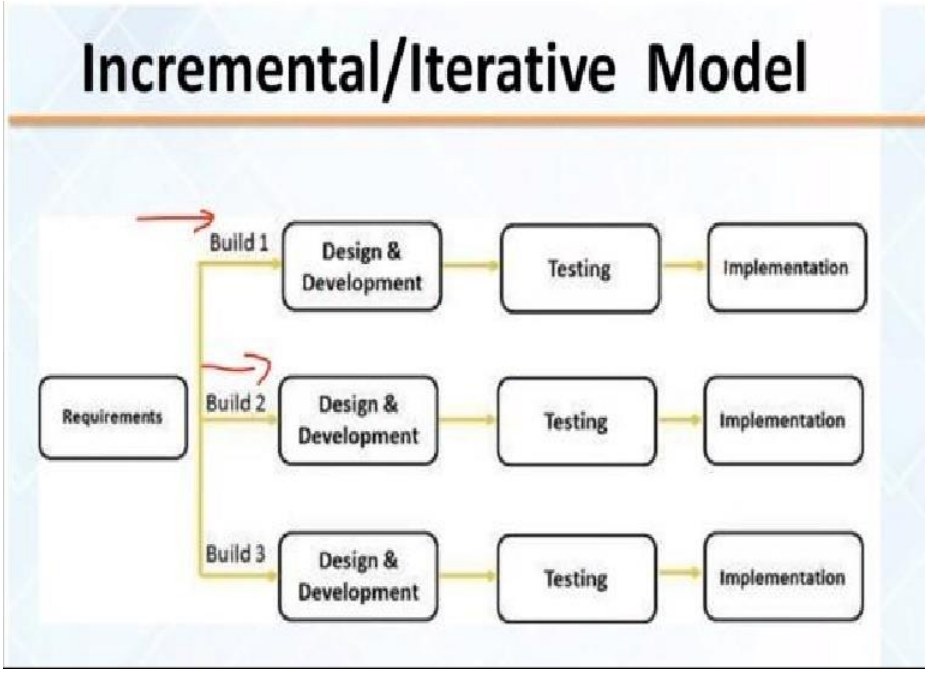
The a **linear**-sequential phases in Waterfall model are −

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document. The requirement specifications form first phase are studied (RSD).
- **System Design** −This phase system design is prepared. This system design helps in **specifying hardware and system requirements and helps in defining the overall system architecture.**
- **Development (Implementation)** − the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released and also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

2.Incremental Model: Iterative Model Or Incremental Model is a process of software development **, here requirements divided into multiple standalone modules of the software**

development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

The various phases of incremental model are as follows:



1. Requirement analysis: In the first phase of the incremental model, **the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team**. To develop the software under the incremental model, this phase performs a crucial role.

2. Design & Development: In this phase **the design of the system functionality and the development method are finished with success.** When software develops new practicality, the incremental model uses style and development phase.

3.Testing: The testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

4. Implementation: Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product.

Advantage of Incremental Model

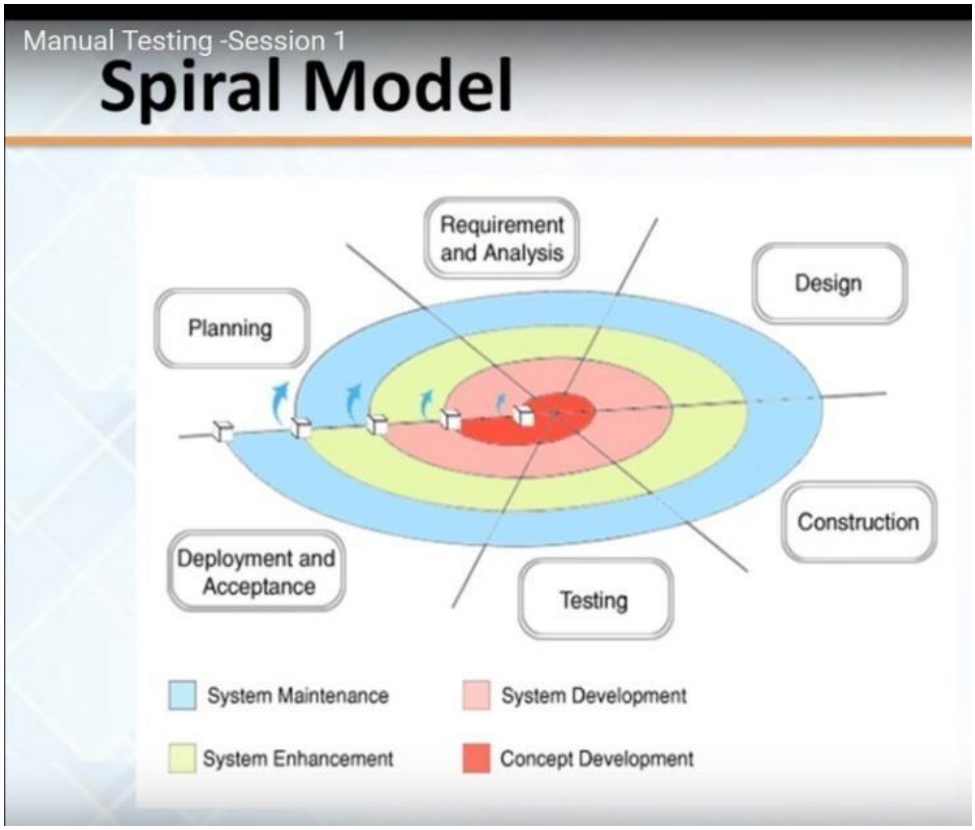
- Errors are easy to be recognized

- Easier to test and debug.
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

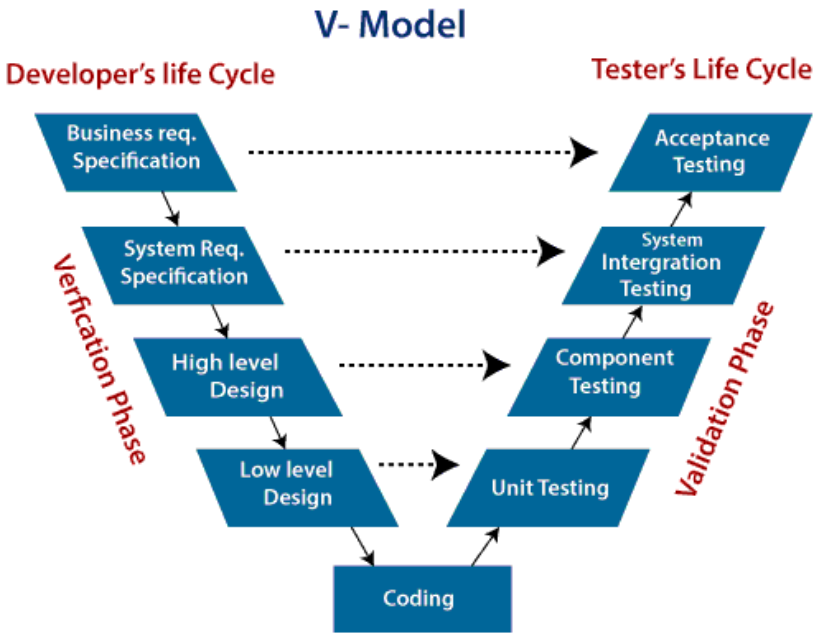
Disadvantage of Incremental Model

- Need for good planning
- Total Cost is high.
- Well defined module interfaces are needed.

3.spiral Model: A Spiral Model of Software Development and Enhancement as an iterative and risk-driven model of software development.It is one of the most preferred Software Development Life Cycle models for large and high-risk projects. The entire process of development is divided into four Planning, Risk Analysis, Engineering and Evaluation. different stages which keep on repeating until the entire project is completed. These stages This entire process is represented in a spiral diagram and thus known as the spiral model.



***4.V-model:** **(Verification & Validation):**V-Model also referred to as the Verification and Validation Model. In this, each phase of SDLC must complete before the next phase starts. It follows a sequential design process same as the waterfall model. Testing of the device is planned in parallel with a corresponding stage of development.



Verification: It involves a static analysis method (review) done **without executing code**. It is the process of evaluation of the product development process to find whether **specified requirements meet.**

Validation: It involves dynamic analysis method (functional, non-functional**),** testing is done by executing code. Validation is the process to classify the software after the completion of the development process to **determine whether the software meets the customer expectations and requirements.**

There are the various phases of Verification Phase of V-model:

1.Business requirement analysis: This is the first step where product requirements understood from the customer's side. This phase contains detailed communication to understand customer's expectations and exact requirements.

2. **Software requirements specification (SRS):** SRS is a document that describes what the software **will do?** and how it **will** be expected to perform. An SRS minimizes **the time, effort required by developers to achieve desired goals and minimizes the development cost.** A good SRS defines how an application will interact with system hardware, other programs and users in a wide variety of real-world situations. Parameters such as operating speed, response time, availability, portability, maintainability, security and speed of recovery etc…

3.The high-level design: (HLD) phase focuses on **system architecture and design.** It provide overview of solution, platform, system, product and service/process. An **integration test plan is created in this phase** as well in **order to test the pieces of the software systems ability to work together.**

4.The low-level design: (LLD) phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. **Component tests** are created in this phase as well.

5.Coding Phase: This is at the bottom of the V-Shape model. Module design is converted into code by developers. After designing, the coding phase is started. Based on the requirements, a suitable programming language is decided. There are some guidelines and standards for coding. Before checking in the repository, the final build is optimized for better performance, and the code goes through many code reviews to check the performance.

There are the various phases of Validation Phase of V-model:

1. **Unit Testing:** In the V-Model, Unit Test Plans (UTPs) are developed **during the module design phase.** These UTPs are executed to eliminate errors at code level or unit level. A unit is the smallest entity which can independently exist, e.g., a program module. Unit testing verifies that the smallest entity can function correctly when isolated from the rest of the codes/ units.

2. **Integration Testing:** Integration Test Plans are **developed during the Architectural Design Phase.** These tests verify that groups created and tested independently can coexist and communicate among themselves.

3. **System Testing:** System Tests Plans are developed **during System Design Phase**. System Tests Plans are composed by **the client?s business team**. System Test ensures that expectations from an application developer are met.

Acceptance Testing: Acceptance testing is related to the business requirement analysis part. It includes testing the software product in user atmosphere. Acceptance tests reveal the compatibility problems with the different systems, which is available within the user atmosphere. It conjointly discovers the non-functional problems like load and performance defects within the real user atmosphere.

When to use V-Model?

- When the requirement is well defined and not ambiguous.
- The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
- The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.

***5. Agile Model**: Agile is also one of the software development processes. Agile method proposes incremental and iterative approach to software design**.**

1. "**Agile process model**" refers to a software development approach based on iterative development. Agile methods **break tasks into smaller iterations or parts** do not directly involve long term planning.

2. The project "**scope and requirements"** are set down at the beginning of the development process. Plans regarding **the number of iterations**, **duration and the scope of each iteration are clearly defined in advance.**
3. Each iteration is considered as a short time **"frame"**, time period is **two to four weeks**.
4. The entire project into smaller parts helps to **minimize the project risk and to reduce the overall project delivery time.**
5. Every iteration involves cross functional teams working simultaneously on various areas including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client. At the end of the iteration, a working product is displayed to the customer and important stakeholders.

phases in the Agile model are as follows:

1. Requirements gathering.
2. Design the requirements.
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback.

1. Requirements gathering: In this phase, you must define the requirements. You should explain **business opportunities and plan the time and effort needed to build the project.** Based on this information, you can evaluate technical and economic feasibility.

2. Design the requirements: When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

3. Construction/ iteration: When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

4. Testing: In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

5. Deployment: In this phase, the team issues a product for the user's work environment.

6. Feedback: After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback

About Scrum: **Scrum** is **agile framework**. it is a project management framework that is applicable to any project. Scrum is mostly used to **complex and product development of Software's** based on an iterative and incremental process.

1. Scrum is **fast, flexible and effective agile framework** ,

2. In Scrum, projects move forward via a series of **iterations** called **sprints. Each iteration** Duration is 2 to 4weeks

Silent features of Scrum are:

- Scrum is light-weighted framework
- Scrum emphasizes self-organization
- Scrum is simple to understand
- Scrum framework help the team work together

Advantage of using Scrum framework:

- Scrum framework is fast moving and money efficient.
- Scrum framework works by dividing the large product into small sub-products. It's like a divide and conquer strategy
- In Scrum customer satisfaction is very important.
- Scrum is adaptive in nature because it have short sprint.
- As Scrum framework rely on constant feedback therefore the quality of product increases in less amount of time.

Different Roles in Scrum: Scrum team **focuses on building quality software.** The owner of a Scrum project focuses on defining what are the characteristics that the product must have to build (what to build, what not and in what order) and to overcome any problems(obstacle) that could hinder the task of the development team. The role may involve:

- Facilitating the daily Scrum and Sprint initiatives.
- Communicating between team members about evolving requirements and planning
- Coaching team members on delivering results
- Handling administrative tasks such as conducting meetings, facilitating collaboration, and eliminating hurdles affecting project progress.
- Shielding team members from external interferences and distractions

Scrum team: A **Scrum** Team is a collection of individuals. Scrum team members between **five and nine members** working together to deliver the required product increments. The Scrum framework encourages a high level of communication among team members. There are three roles in it, and their responsibilities are: The scrum can set up the master team

- **Scrum Master: arrange the meeting and remove problems(obstacles) for the process.**
- **Product owner:** The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.
- **Scrum Team** (**Developers and testers):** The team manages its work and organizes the work to complete the sprint or cycle.

Scrum master:

1. The Scrum Master is not the manager of the Team members, nor is they a project manager, team lead, or team representative.
2. the Scrum Master is **services the Team**.

3. Scrum Master is in-charge of keeping Scrum **up to date, providing coaching, mentoring and training to the teams in case it needs it.**
4. Scrum Master ensures **team co-ordination and supports the progress of the project between individual team members.**
5. The Scrum Master takes the instructions from the Product Owner and ensures that the tasks are performed accordingly.
6. A Scrum Master protects the team from external and internal distractions.
7. A Scrum Master removes problem (mistakes /impediments) to the team can focus on the work at hand and follow scrum practices.

Product owner(p.o):

1. The Product Owner is **the interface between the business, the customers**,
2. The product owner **represents the stakeholders of the project.**
3. Expressing **Product Backlog items clearly and** items to best achieve goals and missions.
4. **Optimizing the value of the work the Team performs.**
5. The role is **primarily responsible for setting the direction for product development or project progress.**
6. P.o owns the **Product backlog** and writes **user stories** and **acceptance criteria**.
7. p.o **prioritizing the Product Backlog and decides the release date and the content.**
8. A Product Owner **accepts or rejects product backlog item.**
9. A P.O. **power to cancel the Sprint,** if he thinks the Sprint goal is redundant.
10. the necessary soft skills to communicate the requirements to the product development team.
11. The Product Owner also understands **the long-term business vision and aligns the project with the needs and expectations of all stakeholders.**
12. End-user feedback is taken into account to determine appropriate next-best action plans for the development throughout the project cycle.
13. Product Owner maintains the Product Backlog (which is the repository for all of this information**), keeping it up to date and at the level of detail and quality the Team requires.**

Development Teams: Development Teams are structured and empowered by the organization to organize and manage their own work. The resulting synergy optimizes the Development Team's overall efficiency and effectiveness. Development Teams have the following characteristics:

- They are self-organizing. No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into Increments of potentially releasable functionality;
- Development Teams are cross-functional, with all the skills as a team necessary to create a product Increment;
- Scrum recognizes no titles for Development Team members, regardless of the work being performed by the person;

- Scrum recognizes no sub-teams in the Development Team, regardless of domains that need to be addressed like testing, architecture, operations or business analysis;
- Individual Development Team members may have specialized skills and areas of focus, but accountability belongs to the Development Team as a whole.

Sprint: The heart of Scrum is a "**Sprint** ". It is a time-box sprint duration is **2-4weeks or one month or less**. During which **a potentially releasable product increment is created**. A new Sprint starts immediately after the conclusion of the previous Sprint. Sprints consist of the Sprint planning, daily scrums, the development work, the Sprint review, and the Sprint retrospective.

- In Sprint planning, the work to be performed, in the **Sprint is planned collaboratively by the Scrum Team.**
- The Daily Scrum Meeting is a 15-minute time-boxed event for the Scrum Team to synchronize the activities and create a plan for that day.
- A Sprint Review is held at the end of the Sprint to inspect the Increment and make changes to the Product Backlog, if needed.

Daily Scrum: The objective of the Daily Scrum is to "**evaluate the progress and trend until the end of the Sprint", synchronizing the activities and creating a plan for the next 24 hours**. The Daily Scrum Meeting is a 15-minute time-boxed. It is a brief meeting that takes place daily during the Sprint period. Three questions are answered individually:

What did I do yesterday?

What am I going to do today?

What help do I need?

The Scrum Master should try to solve problems or obstacles that arise. All team members are required to attend the daily scrum. The people who must attend the Daily Scrum are only members of the Development Team. They are responsible for getting it right.
The **Scrum** Master, the Product Owner, or any Stakeholder may **attend** as listeners.

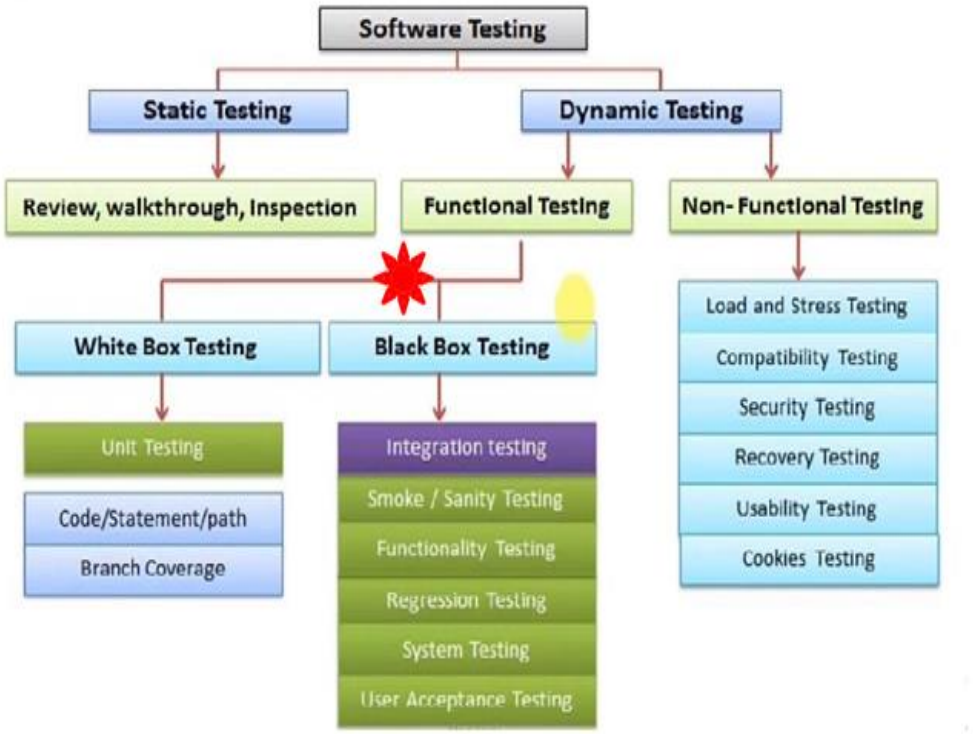
Product backlog: The primary purpose of a backlog grooming session is **to ensure the next few sprints worth of user stories** in the product backlog are prepared for the sprint planning. This is a repository where requirements are tracked with details on the no of requirements(user stories) to be completed for each release. It should be maintained and prioritized by Product Owner, and it should be distributed to the scrum team. Team can also request for a new requirement addition or modification or deletion.

Sprint Planning: At the start of each sprint, a sprint planning meeting is held, The goal of the Sprint Planning is to define **what is going to be done in the Sprint and how it is going to be done.** During which the product owner presents the top items on the product backlog to the team. The Scrum team selects the work they can complete during the coming sprint. That work is then moved from the product backlog to a sprint backlog, which is the list of tasks needed to complete the product backlog items the team has committed to complete in the sprint. Each Sprint is composed of different features. The **sprint planning meeting** is attended by the product owner, Scrum Master and the entire **Scrum** team. Outside stakeholders may attend by invitation of the team, although this is rare in most companies. During the **sprint planning meeting**, the product owner describes the highest priority features to the team.

Sprint review meeting: At the end of each sprint, the team demonstrates the completed functionality at a sprint review meeting, during which, the team shows what they accomplished during the sprint. The goal of the sprint review is to show **what work has been completed with regards to the product backlog for future deliveries.** Participants in the **sprint review** meeting the product owner, the **Scrum** team, the ScrumMaster, management, customers and developers from other projects.

Sprint retrospective: At the end of each sprint, the team conducts a sprint retrospective, The goal of the sprint retrospective is **to identify possible process improvements and generate a plan to implement them in the next Sprint.** which is a meeting during which the team (including its ScrumMaster and product owner) reflect on how well Scrum is working for them and what changes they may wish to make for it to work even better. The team reviews the completed goals of the finished sprint, write down the good and the bad, so as not to repeat the mistakes again. This stage serves to implement improvements from the point of view of the development process. The entire team, including both the ScrumMaster and the product owner should participate.

Types of Software Testing:



I) Static Testing Techniques **II) Dynamic Testing Techniques**

I.VERIFICATION: (Static Testing): It is also called **static testing.** Verification is verifying the documents. **whatever the testing we are conducting in the documents is called as verification.** verification is done by the **devlopers.** Here testing the **requirement documents, design doc, test plan and test script find out errors**. Testing of the software work products manually or with a set of tools. Verify we are developing/ building the product right way.

1. Beginning of that phase.
2. It's a Low-Level Activity.
3. Verification takes place before validation.
4. We don't execute the code.
5. It is less cost job.
6. It is not time consuming job or risk.

In this Activities are involved Inspections, Walkthroughs, Reviews and Desk checking.

Inspections: search as document for **specific check list** find out the defects. It Prepare the document one project person and It is checking by other Project people.

Ex: **project X** A, B, C **project Y** X,Y,Z In above example let X review by the document B, project X

Walkthroughs: reading the document from first to last. It is done by the same project people but other review authors explain.

Review: Review is backbone of software, **Comparing the two similar documents. It is done by the same project people but author need not be present during review.**

- Requirement reviws
- Design reviws
- Code reviews
- Test plan reviews
- Test case reviews
- Code Reviews are done by developers.
- Test plan review done by tester. Ets….

Desk checking: Another project person go to the desk of author (who wrote the document) do quick check.

Dynamic testing is an approach to test the actual software by giving inputs and observing results.

II.VALIDATION: (Dynamic Testing): **whatever the testing we are conducting in the s.w is called as validation**. Validation is done by the **testers.** It is also called dynamic testing.

- Validation the **actual and expected output of the software.**
- **Verify whether we are developing/building the product right.(check the product)**
- It checks the **functional behavior of software system**.
- **Overall the performance of the system.**
- **"**Activities involved in this is Testing the **software application.**

The process during at the **end of the development process** to determine whether it satisfies specified requirements.

1. Am I building a right product?
2. It's a High Level Activity.
3. Validation testing the real product.
4. We execute the code.
5. It is time consuming job or risk.

During this testing we are using the 2 test designing techniques these are:

1. Functional testing (white box testing , black box testing)
2. Non-functional testing

Functional Testing: Functional testing is a type of testing which verifies that each "**function of the software application"**operates in conformance with the requirement specification.

- This testing mainly involves **black box testing**, and it is not concerned about the **source code of the application.**
- Every functionality of the system is tested by providing **appropriate input, verifying the output, and comparing the actual results with the expected results.**
- This testing involves checking of User Interface, APIs, Database, security, client/ server

applications and functionality of the Application under Test.The testing can be done either manually or using automation. Functional testing classified into 2 types

1. White box testing
2. Black box testing

White box Testing: white box testing conducted on the "**source code"** by developers to check does the source code is working as expected or not**, Unit testing and integration testing done by the developers and show the code.** It comes under white box testing. White box testing is also called as glass box, structural, clear box testing.

What is the need of white box testing?

- The defects that are identified in white box testing are very economical to resolve.
- To reduce the defects as early as possible white box testing is helpful.
- To ensure 100% code coverage.

***Black box Testing:** Black Box testing is **conducted on the application** by test engineers or by domaine experts to check whether the application is working according to Customer requirement.

What is the need of Black Box Testing?

System and user acceptance testing don't show the code. It comes under the black box testing**.**

1) Programmers will conduct white box testing in a **positive perception**.

Tester will conduct black box testing with a **negative perception** where there is a more chance of finding more defects. The more defects you identify results in a quality system.

2) White box testing will not cover **non-functional areas.**

3) As functional requirements are also very important for production system those are covered in black box testing.

4) Whereas the objective of black box testing is **100% customer business requirement coverage.**

5) Black Box Testing = System Testing + User Accepting Testing which is called as **requirement Based Testing (or) Specification Based Testing and Behavioral Testing.**

Black box Test design techniques:

Test Design Techniques

- Software testing Techniques help you design better cases. Since exhaustive testing is not possible; Testing Techniques help reduce the number of test cases to be executed while increasing test coverage. They help identify test conditions that are otherwise difficult to recognize.

- Boundary Value Analysis (BVA)
- Equivalence Class Partitioning
- Decision Table based testing.
- State Transition
- Error Guessing

QA Manual Testing Full Course for Beginners Part-1

Boundary Value Analysis (BVA)

- Boundary value analysis is based on testing at the boundaries between partitions.
- It includes maximum, minimum, inside or outside boundaries.

Example1

- Minimum boundary value is 18
- Maximum boundary value is 56

- Valid Inputs: 18,19,55,56
- Invalid Inputs: 17 and 57

- Test case 1: Enter the value 17 (18-1) = Invalid
- Test case 2: Enter the value 18 = Valid
- Test case 3: Enter the value 19 (18+1) = Valid
- Test case 4: Enter the value 55 (56-1) = Valid
- Test case 5: Enter the value 56 = Valid
- Test case 6: Enter the value 57 (56+1) =Invalid

AGE Enter Age *Accepts value 18 to 56

BOUNDARY VALUE ANALYSIS		
Invalid (min -1)	Valid (min, +min, -max, max)	Invalid (max +1)
17	18, 19, 55, 56	57

Example2

- Minimum boundary value is 6
- Maximum boundary value is 12

Name Enter Name *Accepts characters length (6 - 12)

BOUNDARY VALUE ANALYSIS		
Invalid (min -1)	**Valid (min, +min, -max, max)**	**Invalid (max +1)**
5 characters	6, 7, 11, 12 characters	13 characters

- Valid text length is 6, 7, 11, 12
- Invalid text length is 5, 13

- Test case 1: Text length of 5 (min-1) = Invalid
- Test case 2: Text length of exactly 6 (min) = Valid
- Test case 3: Text length of 7 (min+1) = Valid
- Test case 4: Text length of 11 (max-1) = Valid
- Test case 5: Text length of exactly 12 (max) = Valid
- Test case 6: Text length of 13 (max+1) = Invalid

Equivalence Partitioning

- In equivalence partitioning, inputs to the software or system are divided into groups that are expected to exhibit similar behavior, so they are likely to be proposed in the same way. Hence selecting one input from each group to design the test cases.

- It helps to reduce the total number of test cases from infinite to finite. The selected test cases from these groups ensure coverage of all possible scenarios.

Example1

- Valid Input: 18 – 56
- Invalid Input: less than or equal to 17 (<=17), greater than or equal to 57 (>=57)

- Valid Class: 18 – 56 = Pick any one input test data from 18 – 56
- Invalid Class 1: <=17 = Pick any one input test data less than or equal to 17
- Invalid Class 2: >=57 = Pick any one input test data greater than or equal to 57

- We have one valid and two invalid conditions here.

AGE Enter Age *Accepts value 18 to 56

EQUIVALENCE PARTITIONING		
Invalid	**Valid**	**Invalid**
<=17	18-56	>=57

Example2

- Valid input: 10 digits
- Invalid Input: 9 digits, 11 digits

- Valid Class: Enter 10 digit mobile number = 9876543210

EQUIVALENCE PARTITIONING		
Invalid	Valid	Invalid
987654321	9876543210	98765432109

Decision Table

- Decision Table is also called as Cause-Effect Table.
- This test technique is appropriate for functionalities which has logical relationships between inputs (if-else logic).
- In Decision table technique, we deal with combinations of inputs.
- To identify the test cases with decision table, we consider conditions and actions.
- We take conditions as inputs and actions as outputs.

Example-1

- Take an example of transferring money online to an account which is already added and approved.
- Here the conditions to transfer money are
  - ACCOUNT ALREADY APPROVED
  - OTP (One Time Password) MATCHED
  - SUFFICIENT MONEY IN THE ACCOUNT
- And the actions performed are
  - TRANSFER MONEY
  - SHOW A MESSAGE AS INSUFFICIENT AMOUNT
  - BLOCK THE TRANSACTION INCASE OF SUSPICIOUS TRANSACTION

DECISION TABLE

ID	CONDITIONS/ACTIONS	TEST CASE 1	TEST CASE 2	TEST CASE 3	TEST CASE 4	TEST CASE 5
Condition 1	Account Already Approved	T	T	T	T	F
Condition 2	OTP (One Time Password) Matched	T	T	F	F	X
Condition 3	Sufficient Money in the Account	T	F	T	F	X
Action 1	Transfer Money	Execute				
Action 2	Show a Message as 'Insufficient Amount'		Execute			
Action 3	Block The Transaction Incase of Suspicious Transaction			Execute	Execute	X

T = True, F = False, X = Not possible

Example-2

- Take another example -Login page validation. Allow user to login only when both the 'User ID' and 'Password' are entered correct.
- Here the Conditions to allow user to login are
  - Enter Valid User Name
  - Enter Valid Password.
- The Actions performed are
  - Displaying home page
  - Displaying an error message that User ID or Password is wrong.

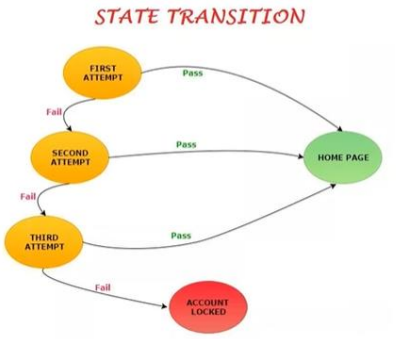
ID	CONDITIONS/ACTIONS	TEST CASE 1	TEST CASE 2	TEST CASE 3	TEST CASE 4
Condition 1	Valid User ID	T	T	F	F
Condition 2	Valid Password	T	F	T	F
Action 1	Home Page	Execute			
Action 2	Show a Message as 'Invalid User Credentials'		Execute	Execute	Execute

State Transition

- In State Transition technique changes in input conditions change the state of the Application Under Test (AUT).
- This testing technique allows the tester to test the behavior of an AUT.
- The tester can perform this action by entering various input conditions in a sequence.
- In State transition technique, the testing team provides positive as well as negative input test values for evaluating the system behavior.

Example1

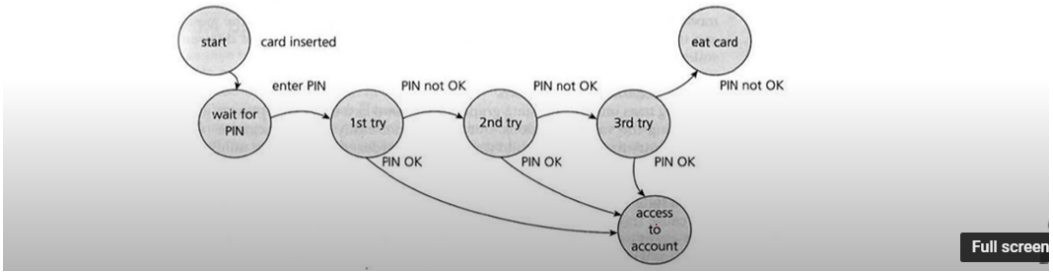
- Take an example of login page of an application which locks the user name after three wrong attempts of password.



STATE	LOGIN	CORRENT PASSWORD	INCORRECT PASSWORD
S1	First Attempt	S4	S2
S2	Second Attempt	S4	S3
S3	Third Attempt	S4	S5
S4	Home Page		
S5	Display a message as "Account Locked, please consult Administrator"		

Example2

- The figure shows an example of entering a Personal Identity Number (PIN) to a bank account. The states are shown as circles, the transitions as lines with arrows and the events as the text near the transitions.
- **States:** States can be numbered as S1, S2 or as alphabets A, B, C etc
  S1:Start, **S2**:Wait for Pin, **S3**: 1st try, **S4**: 2nd Try, **S5**: 3rd Try, **S6**: access to account, **S7**: eat card
  Events: **Event1**:Card inserted, **Event 2**: enter Pin, **Event 3**: Pin OK, **Event 4**: Pin not OK
  Actions: (not shown in the above example) could be : Messages on the screen – error or otherwise.



	Event 1 (Insert card)	Event 2 (Enter Pin)	Event 3 (Pin OK)	Event 4 (Pin not OK)
S1:Start	S2	-	-	-
S2:Wait for Pin	-	S3	-	-
S3: 1st try	-	-	S6	S4
S4: 2nd Try	-	-	S6	S5
S5: 3rd Try	-	-	S6	S7
S6: access to account	-	-	-	-
S7: eat card	S1	-	-	-

Error Guessing

- Error guessing is one of the testing techniques used to find bugs in a software application based on tester's prior experience. In Error guessing we don't follow any specific rules.
- Some of the examples are:
  - Submitting a form without entering values.
  - Entering invalid values such as entering alphabets in the numeric field.

- **Guidelines for Error Guessing:**
- The test should use the previous experience of testing similar applications
- Understanding of the system under test
- Knowledge of typical implementation errors
- Remember previously troubled areas
- Evaluate Historical data & Test results

***Black box testing types or What are software test levels?**

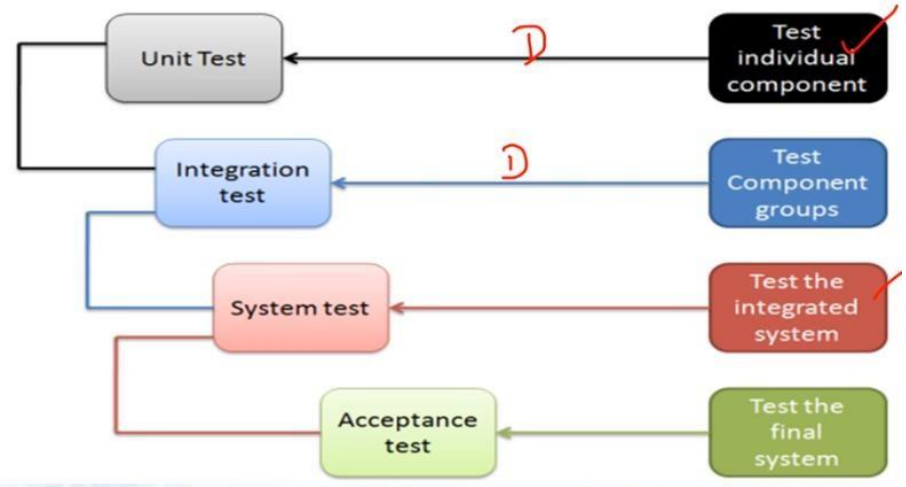
We have Four levels of Testing for General or independent Software Applications. These are

i) **Unit Testing (Component) – conducted by developers**

ii) **Integration testing (**interfaces testing). **– conducted by developers**

iii) **System testing (End-to –End) – conducted by Testers.**

iv) **Acceptance testing (UAT- stands for user acceptancy testing). Conducted by end users.**



I.Unit Testing / Component:

1. **U**nit Testing is also called as **"component Testing".**
2. It is conducted by Developers.
3. "**Testing of individual units of source code/** functioning as expected. **Or "Testing of software components or smallest testable part of s.w**".
4. It is white box testing technique.
5. The goal of unit testing is **each part of the program and test that the individual parts are working correctly.**
6. It is the first level of testing and is performed prior to ***Integration Testing***.

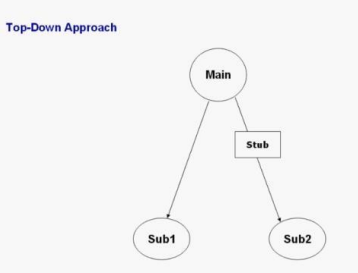
Unit testing techiniques:

- Basic path coverage testing
- Control structure testing
  - ➢ Conditional coverage
  - ➢ Loops coverage
- Mutation testing

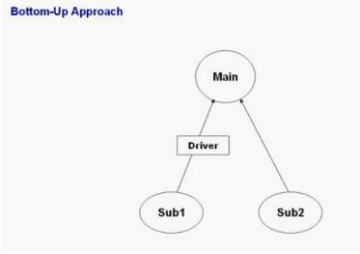
I) Integration Testing: Individual s.w. modules are integrated logically and tested as a groups. (OR) Individual software's are merged and tested those s.w are called integration testing".

1. It is the second level of testing.
2. It is also called "**interfaces testing**".
3. Conducted by Developers.
4. Using white box design testing technics. In this testing developer can doing different approaches.
   a) TOP-DOWN APPROCH
   b) BUTTOM-TOP APPROCH
   c) HYBRID-APPROCH
   d) SYSTEM-APPROCH

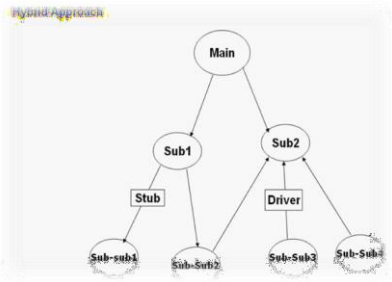
a) TOP-DOWN APPROCH: Developers integrated components or units from top to bottom. Any module is under constructions or missing there replaces using "Stub". Stub is a temporary program. **Stub is called by the module under test.**



b) BUTTOM-TOP APPROCH: Developers integrated components or units modules from bottom to top. Any module is under constructions or missing there replace using "driver". Driver is a temporary program.



c) HYBRID-APPROCH: It is **sandwich approach**. It is combination of top to down and bottom to up approaches. But companies are real time not using this.



d).System approaches: After Completions coding all modules or components developers are integrated all at time it is called **system approaches or big bang approaches.**

*III.System Testing:

System Testing is 3rd level of software testing.This is black box type of testing. Normally, independent Testers can perform System Testing. System testing is also called **End-to –End.** It is conducting on the **software and hardware components**. **System Testing is the testing of a complete and fully integrated software product** based on the software requirements specification (SRS) document. The main focus of this testing is to **Business / Functional / End-user requirements.**

Where external working of the software is evaluated with the help of requirement documents & it is totally based on **Users point of view**. For this type of testing do not required knowledge of internal design or structure or code.

This testing is to be carried out only after *System Integration Testing* is completed where both Functional & Non-Functional requirements are verified. In the integration testing testers are concentrated on finding bugs/defects on integrated modules. But in the ***Software System Testing*** testers are concentrated on finding bugs/defects based on software application behavior, software design and expectation of end user.

Why system testing is important:

- In Software Development Life Cycle the System Testing is perform as the first level of testing where the System is tested as a whole.
- In this step of testing check if system meets functional requirement or not.
- *System Testing* enables you to test, validate and verify both the Application Architecture and Business requirements.
- The application/System is tested in an environment that particularly resembles the effective production environment where the application/software will be lastly deployed.

- Reduces the troubleshooting and maintenance issue, after deployment.
- Demands dedicated team of testers, independent of development team.

Types Of System Testing:

i) **User interface testing (GUI).**
ii) **Usability testing.**
iii) **Functional system testing. (component testing)**
iv) **Non-functional system testing.**

1)UI / GUI Testing:

What is GUI Testing?

- **Graphical User-interface Testing** or **GUI testing** is a process of **testing** the user interface of an application.
- A **graphical user interface** includes all the elements such as menus, checkbox, buttons, colors, fonts, sizes, icons, content, and images.

GUI Testing Checklist

- Testing the size, position, width, height of the elements.
- Testing of the error messages that are getting displayed.
- Testing the different sections of the screen.
- Testing of the font whether it is readable or not.
- Testing of the screen in different resolutions with the help of zooming in and zooming out.
- Testing the alignment of the texts and other elements like icons, buttons, etc. are in proper place or not.
- Testing the colors of the fonts.
- Testing whether the image has good clarity or not.
- Testing the alignment of the images.
- Testing of the spelling.
- The user must not get frustrated while using the system interface.
- Testing whether the interface is attractive or not.
- Testing of the scrollbars according to the size of the page if any.
- Testing of the disabled fields if any.
- Testing of the size of the images.
- Testing of the headings whether it is properly aligned or not.
- Testing of the color of the hyperlink.
- Testing UI Elements like button, textbox, text area, check box, radio buttons, drop downs ,links etc.

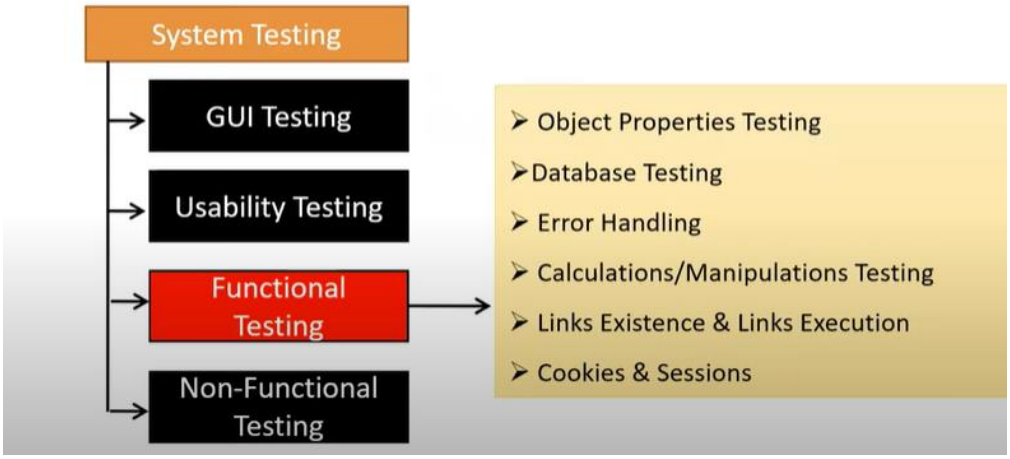
***2.Usability Testing:** Checking how easily the end user is able to **understand and operate the application.**

Usability Testing

- During this testing validates application provided context sensitive help or not to the user.
- Checks how easily the end users are able to understand and operate the application is called usability testing.

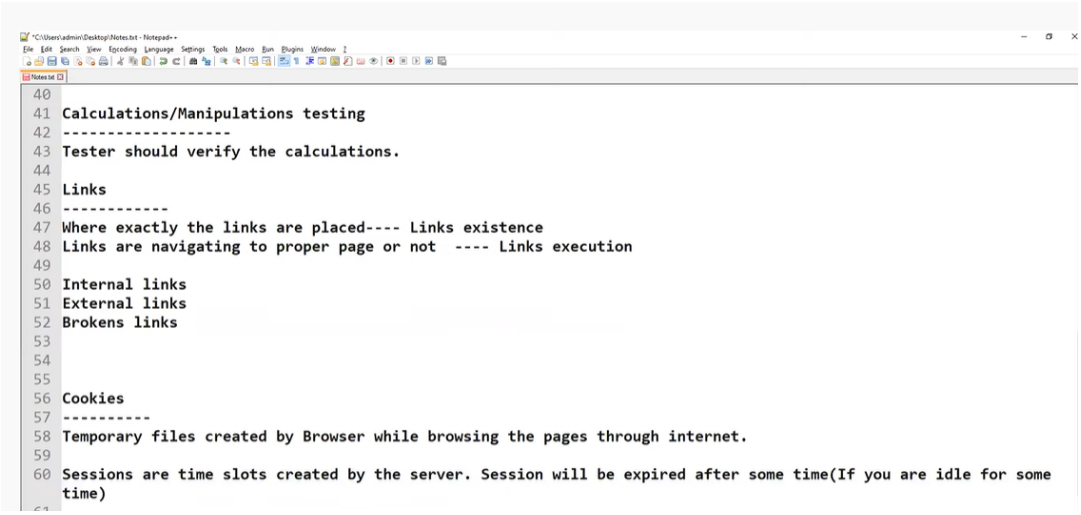
Functional Testing

- Functionality is nothing but behavior of application.
- Functional testing talks about how your feature should work.



```
Functionality Testing
--------
Object properties testing:   Check the properties of objects present on the Application.
  Ex: Enable, disable, visible, Focus......

Database Testing/Backend testing:
-------------------

Checking database operations w,r,t
DML  Operations  ( Data Manipulation Language)

insert
update
delete
select

Table & Column level validations( Column type, column length, number of columns...)
Relation between the tables ( Normalization)
Functions
Precedures
Triggers
Indexes
Views
etc......
```

```
Error Handling Testing
----
Testser verify the error messages while performing incorrect actions on the application.
Error messages should be readle.
User understandable/Simple language.

Incorrect data
Invalid user

Calculations/Manipulations testing
--------------------
Tester should verify the calculations.
```



```
Calculations/Manipulations testing
-------------------
Tester should verify the calculations.

Links
------------
Where exactly the links are placed---- Links existence
Links are navigating to proper page or not  ---- Links execution

Internal links
External links
Brokens links



Cookies
----------
Temporary files created by Browser while browsing the pages through internet.

Sessions are time slots created by the server. Session will be expired after some time(If you are idle for some
time)
```

System Testing following steps needs to be executed:

Step 1) System Test Plan: The initial step of the process involves test plan creation, where in the **lead or test manager define the scope & objective of testing, determines the strategies, decides between manual and automated testing, define the exit & entry criteria**, **assigns roles and responsibilities,** among other things.

Here is list of standard point to be considered while creating System Test Plan:

Goals & Objective

- Scope
- Critical areas, Area to focus
- Test Deliverable
- Testing Strategy for System testing
- Testing Schedule
- Entry and exit criteria
- Suspension & resumption criteria for system testing
- Test Environment
- Roles and Responsibilities
- Glossary.

Step 2) Creation of Test Cases: It is from this step that the process of testing is initiated by the team. **Test cases** are prepared on the basis of **use cases** and the requirements of testing as well as the client/user, such as technical, UI, functional, non-functional, performance, etc. Here you should consider different type of testing like….

Functional testing,Regression testing, Smoke testing, Sanity testing, Ad-hoc testing, Exploratory testing, Usability testing, GUI software testing, Compatibility testing, Performance testing, Load testing, Stress testing, Volume testing, Error handling testing, Scalability testing, Security testing, Capacity testing, Installation testing, Recovery testing, Reliability testing, Accessibility testing etc……

While writing test case you need to check that test cases are covering all functional, non-functional, technical & UI requirements or not.

Sample Test Case Format:

Test Case ID	Test Suite Name	How to Test?	Test Data	Expected Result	Actual Result	Pass/Fail

Step 3) Creation of test data:

Once the test cases are developed by the team, they work together to select or create the required test data, which plays a critical role in test execution. These are the inputs that help the team get expected results.

Step 4) Execution of normal test case & update test case if using any test management tool.

Finally, the test cases created earlier are executed by the team, monitor the process and record any issues encountered by them during the process. The output of the testing is also recorded here.

Step 5) Bug Reporting, Bug verification & Regression testing.

this stage of the process that the team reports all the recorded bugs and issues to the concerned member of the team. Once reported, the programmer or the developer work with the testing team to fix and resolve the issue.

Step 6) Repeat testing life cycle (if required).

After all the issues and bugs are resolved and fixed, the team repeats the test cycle to get the expected results.

Functional Testing Tools: Selenium , QTP, Junit ,SoapUI , Watir.

I.FUNCTIONAL SYSTEM TESTING : It will be conducted both in a positive perception and also in a negative perception.

```
Functional Testing
------------------
a) Input Domain Coverage

b) Output Domain Coverage

c) Database Testing
        Data Integrity
        Data Manipulations
        Data Comparisions
        Data Retriaval
        Data back-up and recovery operations etc...

d) Error Handling

e) Order of Functionalities
```

Positive Testing : Testing conducted on the application in a positive approach to determine what system suppose to do is called positive testing. Positive testing is helpful to check whether the customer requirements are justifying by the application or not.

Negative Testing: Testing a software application with a negative perception to check what system not supposed to do is called negative testing. **Negative testing is helpful to find defects from the software.**

Functional System Testing Approach:

a.Smoke Testing ? What is Smoke testing? Or Build verification testing (BVT)?

It is a kind of quick testing on the application/**build/software** to determine **whether the application is testable or not for move the further testing.** Smoke testing is also called as "**build verification testing or build level testing".**it is initial process testing.

1. Smoke testing is doing only **positive testing**. And test the **important scenarios** only.
2. Smoke testing can be performed by the **tester or developer**.
3. Smoke testing is **day zero check** and it are performed earlier.

b.What is Regression testing?

Regression testing is performing on receiving the software **after fixing the bugs.** It verifies Whether the bugs are fixed and checks if the entire software is working fine with the changes. Regression testing verifies that a code change in the software does not have an impact on the existing functionality of the product / application.

- Regression testing is done on **passed test cases.**
- Regression testing **makes sure that they are no side effects.**
- Re-gression testing is not being always specific to any defect fix, and also regression can be executed for some modules or all modules.
- Re-gression concern with **executing test cases that was passed in earlier builds**.
- Re-testing has **higher priority** over regression, but in some case retesting and regression testing are carried out in parallel.

Regression Testing is important:

- The application works even after the alteration in the code were made.
- The original functionality continues to work as specified even after doing changes in the software application.
- The alteration to the software application has not introduced any new bugs.
- We can perform regression testing manually, but it requires lots of effort.
- To choose the way of doing the regression testing is totally depends on the initial testing approach. If the initial testing approach was manual testing, then the regression testing is usually performed manually.

In case, if the initial testing approach was automated testing, then the regression testing is usually performed by automated testing. Automated regression testing is very easy task.

c.What is Re-testing?

When a bug is fixed by development team, then testing of fixed bug is called Re-testing.
OR Testing functionality repetitively is called re- testing. Re-testing comes in the following 2 scenarios.

- Testing functionality with multiple inputsto confirm if the business validations are implemented or not.
- Testing functionality on the modified build to confirm the bug fixers are made correctly or not.

1. Re-testing is testing of a **particular bug after it has been fixed.**
2. Re-testing is done to **verify defects fixes.**
3. It is **planned testing.**

4. Retesting **concern with executing those test cases that are failed earlier. (Previously failed test cases this time passed).**

5. Re-testing has **higher priority** over regression, but in some case retesting and regression testing are carried out in parallel.

6. Retesting doing on **failed test cases.**

7. Retesting makes sure the **original defect has been corrected.**

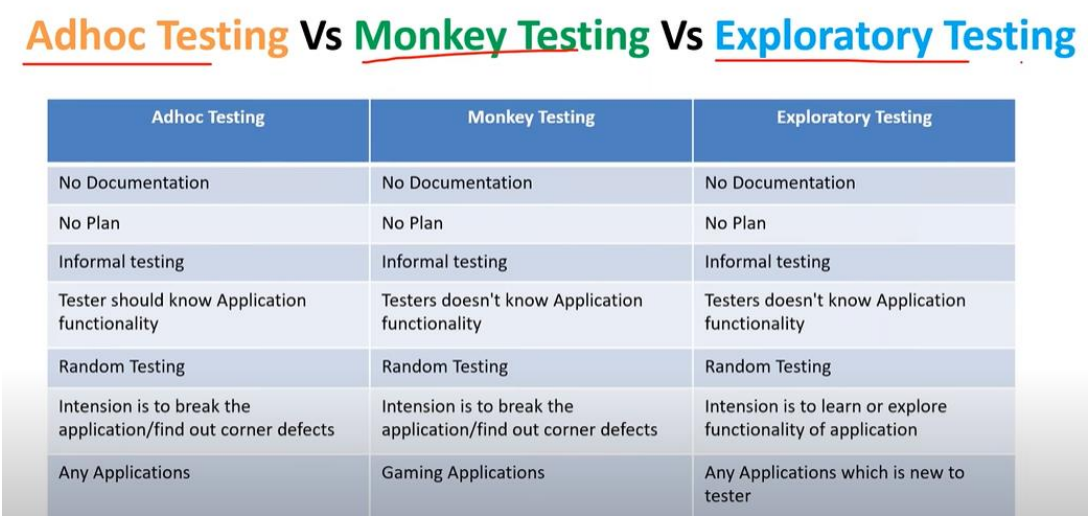
d. What is Sanity testing?

1. Sanity testing is subset of **regression testing.**
2. Sanity is performed after **the smoke testing.**
3. Sanity testing is done during the **release phase.**
4. To check the **major functionalities of the application without going to deeper.**
5. Sanity testing is performed **after receiving the s.w build**, with in minor changes in a code or functionality to ascertain that the bugs have been fixed and no further
6. issues are introduced due to these changes.
7. Sanity is performed by **the tester or the developer**
8. Sanity testing is performed when **new build is released after fixing bugs.**

e.Formal Testing: If you tested software application by following all **preplan procedures and proper documentation** then it is called formal testing.

.**Exploratory Testing:** when there is a requirement or an application to test but with the no relevant document or details provided to tester explores the application and keep checking the features to figure out the issues is known as exploratory testing.

f. Adhoc Testing: **If you test software** without following any procedures and documentation then it is called adhoc-testing**. It is also called informal testing. OR** Testing conducted on an application unevenly or zig zag way with an intension of finding tricky defects is called monkey testing.



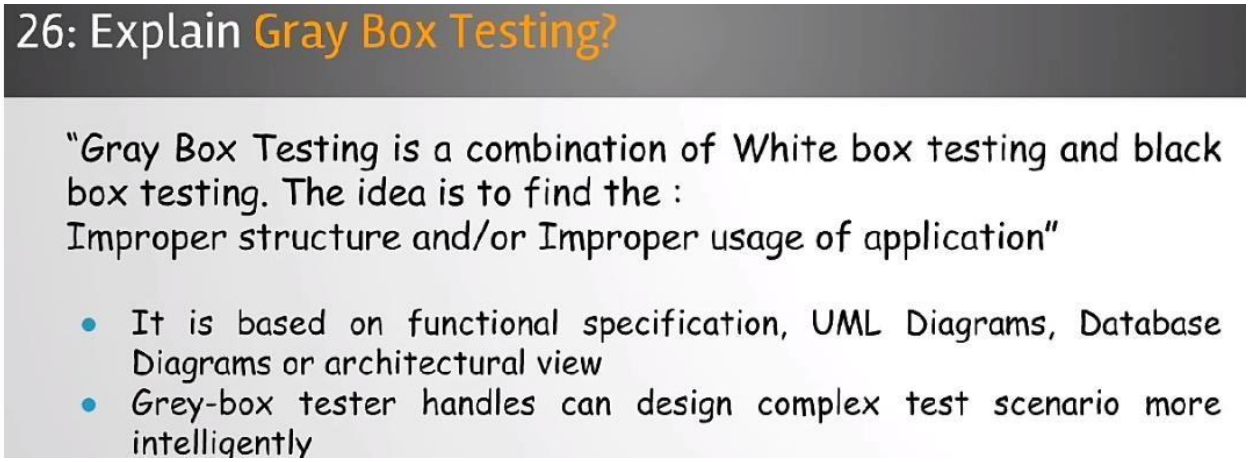
Adhoc Testing	Monkey Testing	Exploratory Testing
No Documentation	No Documentation	No Documentation
No Plan	No Plan	No Plan
Informal testing	Informal testing	Informal testing
Tester should know Application functionality	Testers doesn't know Application functionality	Testers doesn't know Application functionality
Random Testing	Random Testing	Random Testing
Intension is to break the application/find out corner defects	Intension is to break the application/find out corner defects	Intension is to learn or explore functionality of application
Any Applications	Gaming Applications	Any Applications which is new to tester

g.Risk Based Testing (or) Priority Based Testing: Identifying the critical functionality in the system and testing it first**. (Or)** Conducting testing in the same order of priority is called risk based testing or priority based testing.

i.End-to-End Testing: Testing the overall functionalities of the system including the data integration among all the modules is called end-to-end testing.

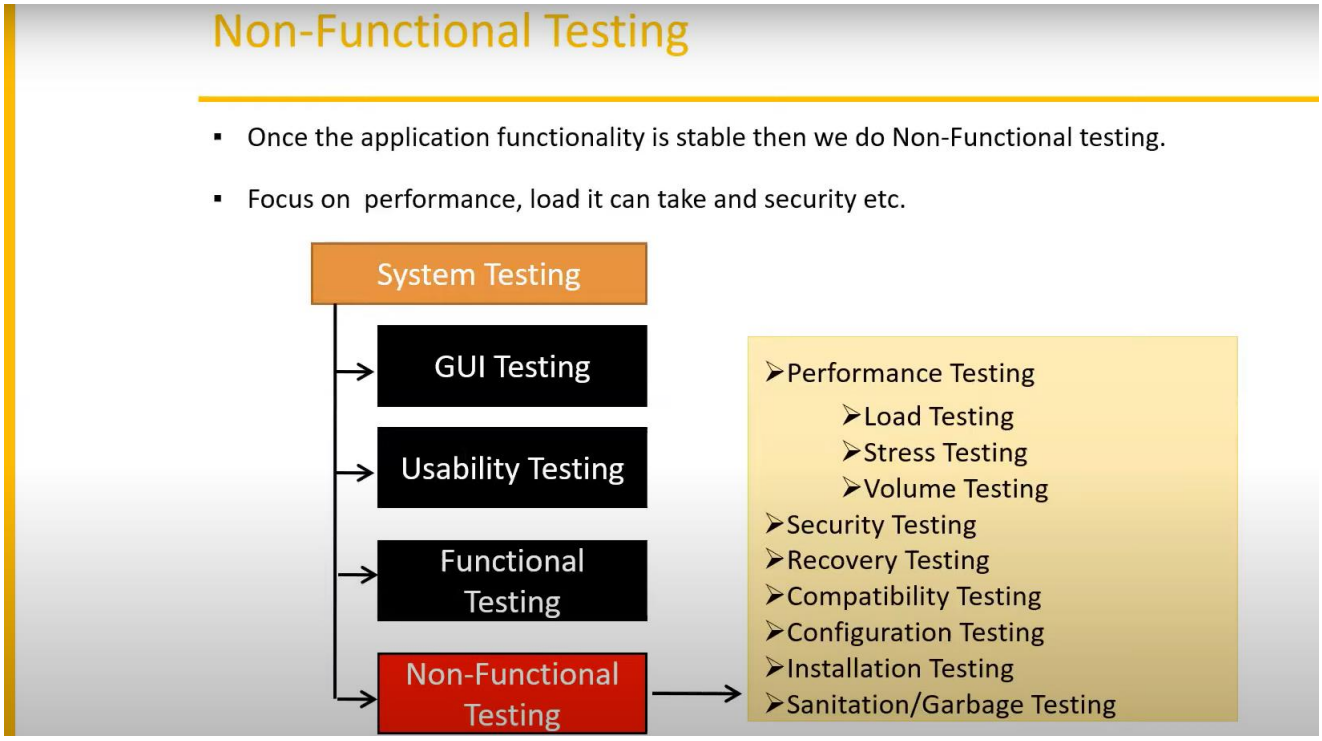
Globalization Testing: checking if the application having a provision of setting and changing languages, date and time format and currency etc. If it is designed for global users.

Localization Testing: checking default languages, currency, date and time format etc. If it is designed for a particular locality



II. Non-Functional System Testing

Validating various non functional aspects of the system such as user interfaces, user friendliness, security, compatibility, load, stress and performance etc is called non-functional system testing. Non-functional testing is equally important as functional testing and affects client satisfaction.



*

*1.Performance Testing:

```
Non-Functional Testing
------
Performance---- Speed of the application
    Load
    Stress
    Volume

    Load: gradaully Increasing the load on the application then check the speed of the application.
    Stress : suddenly increase/decrease the load on the application and check speed of the application.
    Volume: Check how much data is able to handle by the application.
```

Performance testing is the process of determining the speed, responsiveness and stability of a computer, network, software program or device under a workload.

- Speed - Determines whether the application responds quickly
- Scalability - Determines maximum user load the software application can handle.
- Stability - Determines if the application is stable under varying loads.

Types of Performance testing:

- **Load Testing:** To evaluate the **behavior of a system at an increasing workload.**
- **Stress Testing:** Stress Testing is also known as Endurance Testing. Stress Testing is defined as a type of Software Testing that verified the stability & reliability of the system. This test mainly

determines the system on its robustness and error handling under extremely heavy load conditions.

It is done to make sure the software **can handle the expected load over a long period of time.**

- **Spike Testing: T**o evaluate the behavior of a system **when the load is suddenly and substantially increased.**

2. Security Testing:

```
Security testing: How secure our application.

   Authentication ---> Users are valid or not
   Autherization/Access Control ---> permissions of the valid user.
```

***3.Recovery Testing:** check the system change to abnormal or normal stage

***4.Compatibility Testing:** checking if the application is compatible with the different software and hardware environments.

```
Compatibility testing
---------------------
Forward  Compatibility
Backward Compatibility
Hardware Compatibility  Configuration testing
```

Installation Testing: checking if we are able to install the software successfully or not as per the guidelines given in installation document.

```
Installation testing
------------
Check screens are clear to understand.
Screens navigation
Simple or not
Un-installation
```

```
Security testing: How secure our application.

   Authentication ---> Users are valid or not
   Autherization/Access Control ---> permissions of the valid user.

Recovery testing :
            check the the system change to abnormal to normal.

Compatibility testing
---------------------
Forward  Compatibility
Backward Compatibility
Hardware Compatibility (Configuration testing)


Installation testing
-------------
Check screens are clear to understand.
Screens navigation
Simple or not
Un-installation

Sanitation/Garbage testing
-------
If any application provides extra features/functionality then we consider them as bug.
```

11. Documentation testing: It helps to estimate the required testing efforts and track the requirements. Software documentation includes a test plan, test cases, and requirements section. It tests the documented artifacts.

12) Reliability: The extent to which any software system continuously performs the specified functions without failure.

14) Flexibility: The term refers to the ease with which the application can work in different hardware and software configurations. Like minimum RAM, CPU requirements.

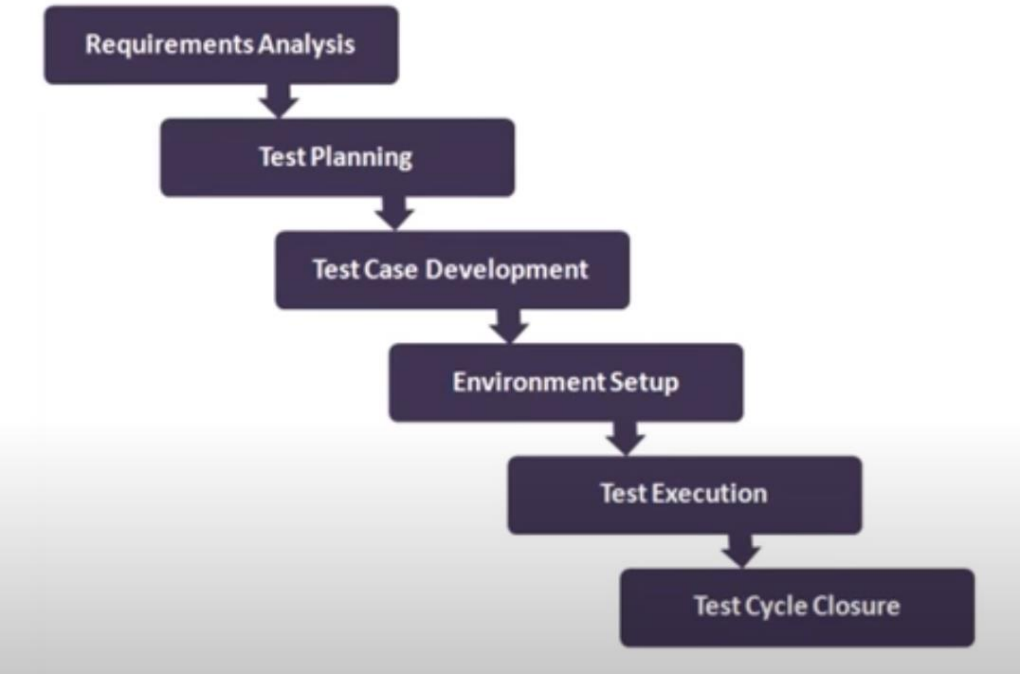
15) Volume testing : testing performance with a high volume of data, not necessarily high number of users, but could be one user performing a high-volume task, such as a multiple-file upload.**Reusability:** It refers to a portion of the software system that can be converted for use in another application.

16) Scalability testing : checking a system's ability to scale with increased usage and to what extent performance is affected.

Functional Testing Vs Non-Functional Testing

Functional Testing	Non-functional Testing
▪ Validates functionality of Software. ▪ Functionality describes what software does. ▪ Concentrates on user requirement. ▪ Functional testing takes place before Non-functional testing.	▪ Verify the performance, security, reliability of the software. ▪ Non-Functionality describes how software works. ▪ Concentrates on user expectation. ▪ Non-Functional testing performed after finishing Functional testing.

II.STLC – Software Testing Life Cycle



SNO	PHASE	Input	Activities	Responsibility	Out Come
1	Test Planning	Project Plan	➢Identify the Resources	Test Lead/Team Lead (70%)	Test Plan Document
	What to test	Functional Requirements	➢Team Formation	Test Manager (30%)	
	How to test		➢Test Estimation		
	when to test		➢Preparation of Test Plan		
			➢Reviews on Test Plan		
			➢Test Plan Sign-off		
2	Test Designing	Project Plan	➢Preparation of Test Scenarios	Test Lead/Team Lead(30%)	Test Cases Document
		Functional Requirements	➢Preparation of Test Cases	Test Engineers(70%)	Traceability Matrix
		Test Plan	➢Reviews on Test Cases		
		Design Docs	➢Traceability Matrix		
		Use cases	➢Test Cases Sign-off		
3	Test Execution	Functional Requirements	➢Executing Test cases	Test Lead/Team Lead(10%)	Status/Test Reports
		Test Plan	➢Preparation of Test Report/Test Log	Test Engineers (90%)	
		Test Cases	➢Identifying Defects		
		Build from Development Team			
4	Defect Reporting & Tracking	Test Cases	➢Preparation of Defect Report	Test Lead/Team Lead(10%)	Defect Report
		Test Reports/Test Log	➢Reporting Defects to Developers	Test Engineers (90%)	
5	Test Closure/Sign-Off	Test Reports	➢Analyzing Test Reports	Test Lead/Test Manger(70%)	Test Summary Reports
		Defect Reports	➢Analyzing Bug Reporting	Test Enginners(30%)	
			➢Evaluating Exit Criteria		

Software Testing Life Cycle (**STLC**) is the testing process. In **STLC** process, different activities are carried out to improve the quality of the product**.** STLC involved below stages…….

1) **Requirement Analysis.**
2) **Test Planning (Test Strategy, Test Plan, Test Bed Creation).**
3) **Test Designing / Development (Test Procedures, Test Scenarios, Test Cases,rtm).**
4) **Test Environment.**
5) **Test Executions.**
6) **Bug/Defect reporting or Tracking.**
7) **Test Reporting.**
8) **Test closing.**

1. Requirement Analysis: Requirement Analysis is the very first step in **Software Testing Life Cycle (STLC)**. In this step **Quality Assurance (QA)** team understands the requirement in terms of **what we will testing & figure out the testable requirements**. If any conflict, missing or not understood any requirement, then QA team follow up with the various stakeholders, Business Analyst, System Architecture, Client, Technical Manager/Lead etc.. to better understand the detail knowledge of requirement.

From very first step QA involved in the where STLC which helps to prevent the introducing defects into Software under test. The requirements can be either Functional or Non-Functional like Performance, Security testing.

2. Test Planning: Once the requirement gathering phase is completed based on the requirement analysis, start preparing **the Test Plan.**

- **Test plan is a document,** Test Planning is most important phase *of* STLC.

- It is prepared by Team **leader /Test lead .**
- In this involved to determine the effort and **cost estimates for entire project** The Result of Test Planning phase will be T**est Plan/Test strategy/Testing Effort estimation documents.**
- It is detail **day to day work plan, scope, approach, resources ,time lines,what is the features to tested what are the features not to be tested and work schedules etc.** Once test planning phase is completed the QA team can start with test cases development activity

***3.Test Design:** Test team start with test case development. T**est team can prepare the test senarios**, **test cases, test procedures and test scripts**. Once the test cases are prepared these are reviewed by **Test lead or senior tester, And a**lso test team prepares the **Requirement Tracebility Matrix. RTM** trace the requirements to the test cases that are need to be verify whether the requirements are full field.

Explain Test Designing options:

a. use case . b. Test scenario c. Test Case documentation d. Test Data collection.

a. Use case : A use case is a methodology used in system analysis to identify, clarify and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. The method creates a document that describes all the steps taken by a user to complete an activity.

- Use case is a document ,
- it is prepared by business analyst,
- it is part of functional requirement.

Every use case contains three essential elements:

The actor. The system user -- this can be a single person or a group of people interacting with the process.

The goal. The final successful outcome that completes the process.

The system. The process and steps taken to reach the end goal, including the necessary functional requirements and their anticipated behaviors.

b. Creating Test Scenarios?

Test scenario is a high level test case.

It is prepared by test engineer.

"**What to be tested is called test scenario".**

Possible areas to be tested**.**

Based on use case prepared Test scenario.

"*A Test Scenario is a statement describing the functionality of the application to be*

tested. It is used for end-to-end testing of a feature and is generally derived from the use cases." A single test scenario can cover one or more test cases. Therefore a test scenario has a one-to-many relationship with the test cases.

Test Case V/s Test Scenario

- Test case consist of set of input values, execution precondition, excepted Results and executed post condition, developed to cover certain test Condition. While Test scenario is nothing but test procedure.
- Test cases are derived (or written) from test scenario. The scenarios are derived from use cases.
- In short, Test Scenario is '*What to be tested*' and Test Case is '*How to be tested*'.

- **Example:-**
- Test Scenario: Checking the functionality of Login button
  - TC1: Click the button without entering user name and password.
  - TC2: Click the button only entering User name.
  - TC3: Click the button while entering wrong user name and wrong password.

c. What is Test Case documentation?

***1.What is Test Case?**

Test case is a document.

It is prepared by test engineer.

"How to test? is called as test case**"**,

based on Test scenario prepared test case.

Test case describes the test steps.

Test cases are the set of positive and negative executable steps of test scenario. **Test case is a set of actions executes to verify a particular functionality of s.w. A set of input values, execution preconditions, expected results and actual result. Or**

1. **Who Prepare Test Cases:** In System Testing Level **Testers prepare** test cases, under the guidance of Test Lead. In Unit Testing Level **developers** prepare Unit test cases.

2. Give a sample Test Case Template:

i) **Test Case Id:** a Unique name/number (Alfa-numeric)
ii) **Test Case Name:** Name of Test Case
iii) **Test Suite ID:** Unique name/number (Alfa-numeric)
iv) **Pre-Condition:** Status before Test Case Execution
v) **Priority (p1,p2,p3…)** it describe the execution order of the test case.
vi) **Steps:** Steps for Executing the Test Case
vii) **Post-Condition:** Status after Test Case Execution
viii) **Expected Result:** Expected Result as per Requirements.

ix) Actual Results: ----------

x) Test Results: Pass / Fail

x) **Remarks:** Comments (Optional) ----------

Note 1: You prepare this Test Case Template in Excel Sheet

Note 2: Test Case Template may vary from one company to another and one project to another.

Note 3: In the above template Actual Results and Test Results fields can be filled in Test Execution phase, Remaining fields in Test Design phase.

Note: In Industry practically we prepare only functionality test cases. Three types of functional test cases: **1) Positive test cases 2) Negative test cases 3) Business validation test cases**

Positive Test Case:A Test case prepared in a positive perception to check what a system suppose to do is called positive test case.

Example:- Tc1: Check Login with valid inputs.

Negative Test Case: A Test case prepared to check what a system not suppose to do is called negative test case.

Example:- Tc1: Check Login with invalid inputs.

Business Validation Test case: A test case prepared to check business conditions is called business validation test cases.

Example:- Check fund transfer with 1000, 10000, 15000, etc is called business validation test case.

What is test suite?

Test suit is a group of test cases which belong to same category.

What is Test Data?

Test data is an input for executing the test cases. It contains the positive and negative data, positive and negative testing. we do place all the test data in database or excel sheet or csv file. Most of the companies get the real time historical data from client and use it while testing.

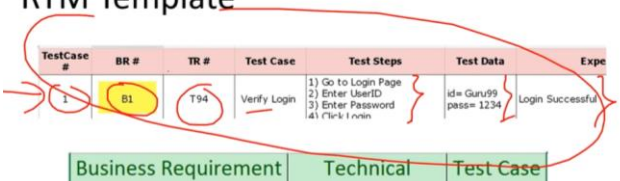
How to collect test data:

1. Some test data prepare the testers.
2. Some data collected from developers.
3. Some test data collected from customers. And some other resources

Requirement Traceability Matrix(RTM)

- What is RTM (Requirement Traceability Matrix)?
- Requirement Traceability Matrix or RTM captures all requirements proposed by the client.
- In other words, it is a document that maps and traces user requirement with test cases.
- The main purpose of Requirement Traceability Matrix is to see that all test cases are covered so that no functionality should miss while doing Software testing.
- Requirement Traceability Matrix – Parameters include
  - Requirement ID (Business/Technical)
  - Test case ID

RTM Template



Business Requirement #	Technical Requirement #	Test Case ID
B1	T94	1
B2	T95	3
B3	T96	3
B4	T97	4

Requirement Traceability Matrix

Advantage of Requirement Traceability Matrix

- It confirms 100% test coverage
- It highlights any requirements missing or document inconsistencies

5.Test Environment: Test enviroment setup is done based on the S.W and H.W requirement list. Test enviroment is the dummy configuration of s.w and H.w. where tester executes their test cases. In some cases test team may not be involves in this phase. Dev team or customer team provides the test Enviroment. While test team should prepare the smoke test cases to check the readiness of the given test environment.

Or

Test Environment

Setting up a right test environment ensures software testing success. Any flaws in this process may lead to extra cost and time to the client.

What is a Test bed?

In general, a test bed is a software development environment. It allows the developers to test their modules without affecting the live production servers. The test bed is not confined to developers only but also used by testers. It is referred as test environment as well.

What is a Test Environment?

A testing environment is a setup of software and hardware for the testing teams to execute test cases. In other words, it supports test execution with hardware, software and network configured.

Test bed or test environment is configured as per the need of the Application Under Test. At few occasion, test bed could be the combination of the test environment and the test data it operates.

1. Test Executions:

Test Execution

- During this phase test team will carry out the testing based on the test plans and the test cases prepared.
- **Entry Criteria:** Test cases , Test Data, Test Plan or Test Strategy.
- **Activities:**
  - Test cases are executed based on the test planning.
  - Status of test cases are marked, like Passed, Failed, Blocked, Run, and others.
  - Documentation of test results and log defects for failed cases is done.
  - All the blocked and failed test cases are assigned bug ids.
  - Retesting once the defects are fixed.
  - Defects are tracked till closure.

- **Deliverables:** Provides defect and test case execution report with completed results.

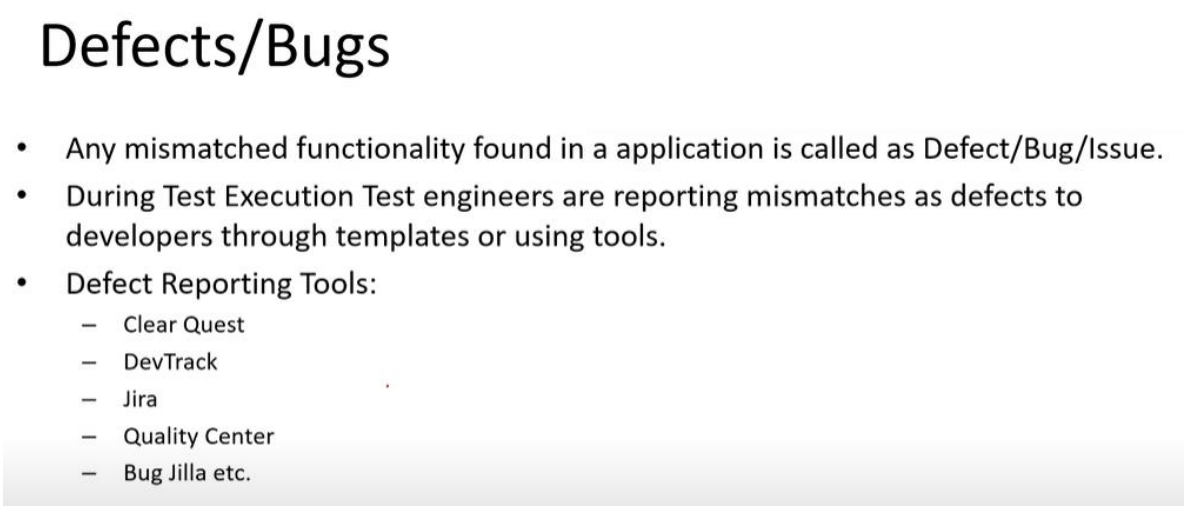
Guidelines for Test Execution

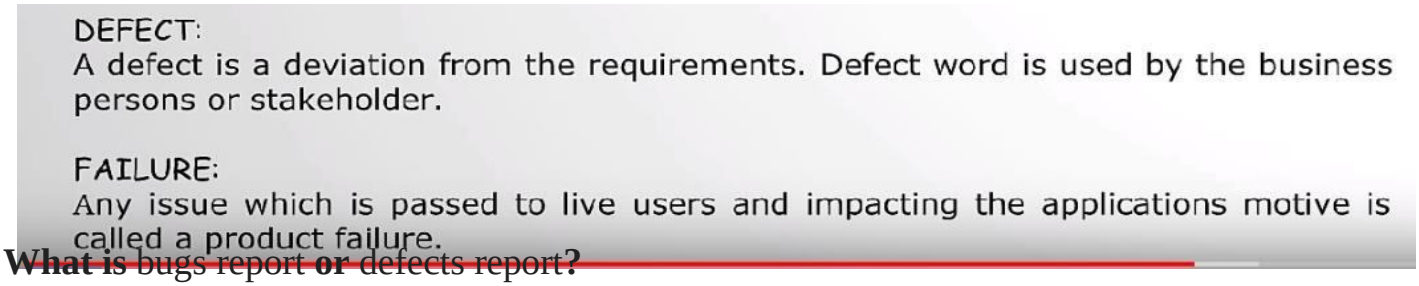
- The Build being deployed to the quality assurance environment is the most important part of the test execution cycle.
- Test execution is done in Quality Assurance (QA) environment.
- Test execution happens in at least two cycles.
- Test execution phase consists Executing the test cases + test scripts(if automation).
- Exploratory tests are carried out once the build is ready for testing.

Test team starts executing the test cases based on the planned test cases. If a test case result is Pass/Fail then the same should be updated in the test cases. Defect report should be prepared for failed test cases and should be reported to the Development Team through bug tracking tool (e.g. Quality Center) for fixing the defects. Retesting will be performed once the defect was fixed.

. Bug/Defect reporting or Tracking. And management.

What is Bug?





What is bugs report or defects report?

A bugs report or defects report is a list of bugs found out testers while testinga software product in testing phase under a testing environment.....Bugs ordefects report is the most vital tool for developers to understand where the product that has been built by them is lacking its functionality or performance.

Bug Reporting Template

Defect Report Contents

- **Defect_ID** - Unique identification number for the defect.
- **Defect Description** - Detailed description of the defect including information about the module in which defect was found.
- **Version** - Version of the application in which defect was found.
- **Steps** - Detailed steps along with screenshots with which the developer can reproduce the defects.
- **Date Raised** - Date when the defect is raised
- **Reference**- where in you Provide reference to the documents like . requirements, design, architecture or may be even screenshots of the error to help understand the defect
- **Detected By** - Name/ID of the tester who raised the defect
- **Status** - Status of the defect , more on this later
- **Fixed by** - Name/ID of the developer who fixed it
- **Date Closed** - Date when the defect is closed
- **Severity** which describes the impact of the defect on the application
- **Priority** which is related to defect fixing urgency. Severity Priority could be High/Medium/Low based on the impact urgency at which the defect should be fixed respectively

Explain Defect Severity And Priority In Software Testing?

In general defect severity and priority's proposition to each other but in some situation that may vary.

What is Severity?

"Severity describes the impact of the defect on the application". Severity is given by the teser it could not be changed.The severities are defined **as Critical or blocker, Major, Medium and low .**

Critical: A critical means blocker, cant proceed farther testing an application , major or critical functionalities can't be working. OR

Critical issue **"where a large functionality or major system component is completely broken and there is no workaround to move further testing"**.

Major: A major severity is an issue "major **functionality or major system component is completely broken and there is a workaround to move further testing".**

For example: login application give the valid input meas enter valid user name and valid pass word then successfully login.

give the invalid input meas enter in-valid user name and in-valid pass able to long in its wrong. It is called critical.

Medium: A medium severity is an issue that **"imposes some loss of functionality, but for which there is an acceptable & easily reproducible workaround".**

For example, font family or font size or color or spelling issue.

low: spellings, colors, alianments ete….

What is Priority?

"**The priority describes the time line on the defect" and importance of the defect**.

Priority **is** defined **as the order in which a** defect **should be fixed.** Priority can be changed by tester, developer, stakeholder. **Higher the** priority **the sooner should be resolved.**

The priorities are defined as P1(High), P2(Medium) and p3 (low). Defect priority can be defined as how soon the defect should be fixed. It gives the order in which a defect should be resolved. Developers decide which defect they should take up next based on the priority.

Defect Priority

- Priority describes the importance of defect.
- Defect Priority states the order in which a defect should be fixed. Higher the priority the sooner the defect should be resolved.
- **Defect priority can be categorized into three class**
  - **P1 (High) :** The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed
  - **P2 (Medium):** During the normal course of the development activities defect should be resolved. It can wait until a new version is created
  - **P3 (Low):** The defect is an irritant but repair can be done once the more serious defect have been fixed

- **A very low severity with a high priority:**
  - A logo error for any shipment website, can be of low severity as it not going to affect the functionality of the website but can be of high priority as you don't want any further shipment to proceed with wrong logo.

- **A very high severity with a low priority:**
  - For flight operating website, defect in reservation functionality may be of high severity but can be a low priority as it can be scheduled to release in a next cycle.

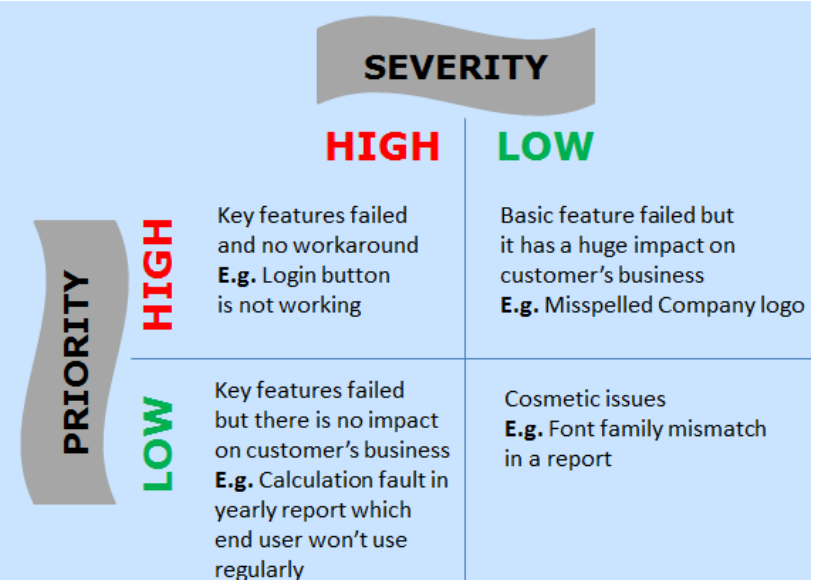
- **Example for High Priority & High Severity defect:**
  - If 'Login' is required for an Application and the users are unable to login to the application with valid user credentials. Such defects need to be fixed with high importance. Since it is stopping the customer to progress further.

- **Example for Low Priority and High Severity defect:**
  - If an application crashes after multiple use of any functionality i.e. if 'Save' Button (functionality) is used for 200 times and then the application crashes, such defects have High Severity because application gets crashed, but Low Priority because no need to debug right now you can debug it after some days.

- **Example for High Priority & Low Severity defect:**
  - If in a web application, company name is miss spelled or Text "User Nam:" is displayed instead of "User Name:" on the application login page. In this case, Defect Severity is low as it is a spell mistake but Priority is high because of its high visibility.

More examples...

- **Low priority-Low severity** - A spelling mistake in a page not frequently navigated by users.
- **Low priority-High severity** - Application crashing in some very corner case.
- **High priority-Low severity** - Slight change in logo color or spelling mistake in company name.
- **High priority-High severity** - Issue with login functionality.



High Priority & High Severity: A critical issue where a large piece of functionality or major system component is completely broken**.**

For example: 1. Submit button is not working on a login page and customers are unable to login to the application

2. On a bank website, an error message pops up when a customer clicks on transfer money button.

3. Application throws an error 500 response when a user tries to do some action. 500 Status Codes:

The server has problems in processing the request and these are mainly server errors and not with the request.

These kinds of showstoppers come under High Priority and High Severity. There won't be any workaround and the user can't do the further process.

Low Priority & High Severity: An issue which won't affects customers business but it has a big impact in terms of functionality.

For example:1. Crash in some functionality which is going to deliver after couple of releases.

2. There is a crash in an application whenever a user enters 4 digits in the age field which accepts max 3 digits.

High Priority & Low Severity: A minor issue that imposes **some loss of functionality,** but for which there is an **acceptable & easily reproducible workaround**. Testing can proceed without interruption but it **affects customer's reputation.**

For example: 1. Spelling mistake of a company name on the homepage

2. Company logo or tagline issues

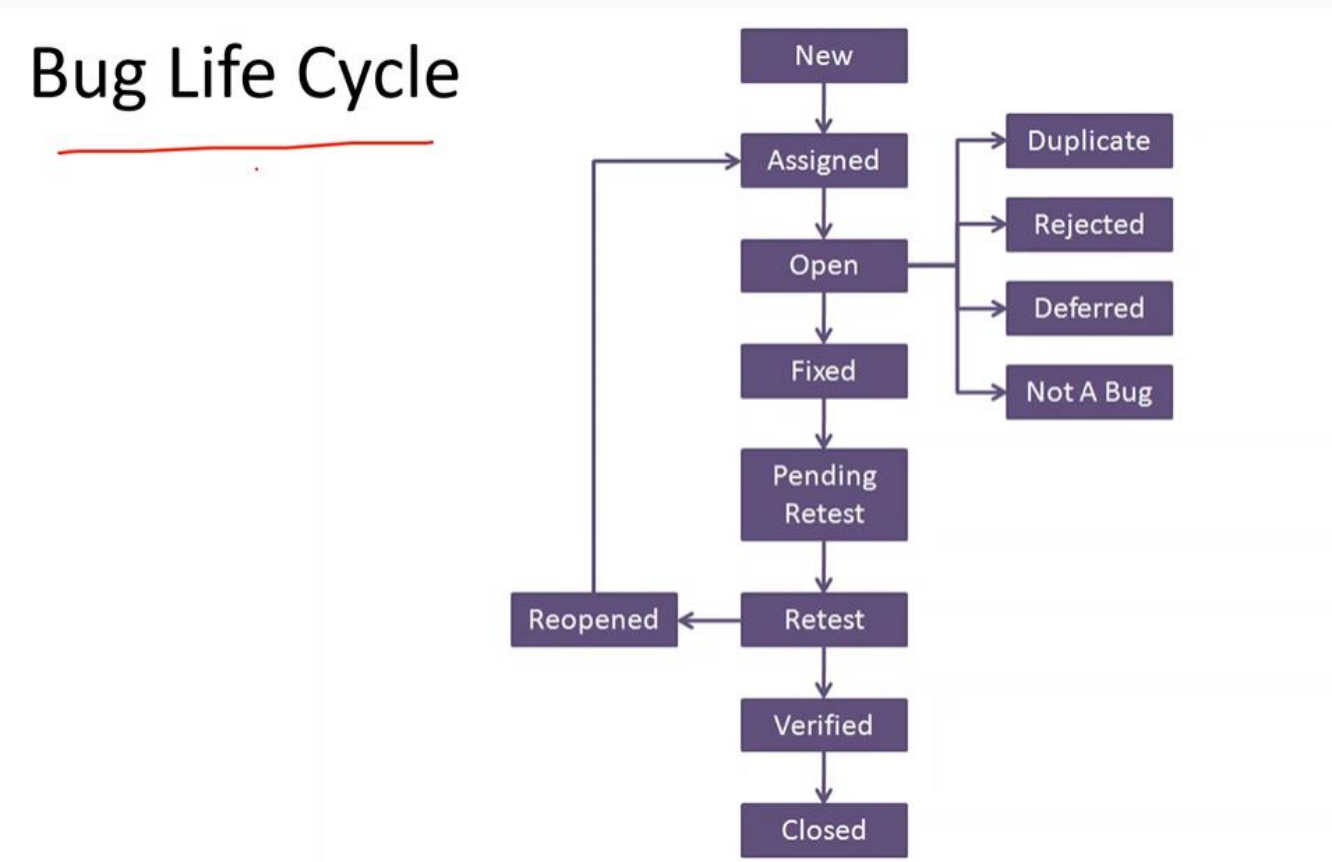
It is important to fix the issue as soon as possible, although it may not cause a lot of damage.02

Defect Resolution

- After receiving the defect report from the testing team, development team conduct a review meeting to fix defects. Then they send a Resolution Type to the testing team for further communication.
- **Resolution Types:-**
  - Accept
  - Reject
  - Duplicate
  - Enhancement
  - Need more information
  - Not Reproducible
  - Fixed
  - As Designed

What is bug/ Defect Life Cycle?

Defect life cycle, also known as **Bug Life cycle** is the journey of a defect cycle, which a defect goes through during its lifetime. It varies from organization to organization, and also from project to project. The software testing process depends upon the tools used(JIRA, Qc and HP ALM).



Bug can be defined as the abnormal behavior of the software. Bug starts when defect is found and ends when a defect is closed, after ensuring it is not reproduced.

When a tester finds a bug and posting it very first time then the status of the defect is "NEW"

Tester should provide proper way of documentation to development team, to reproduced fix the defect.

Once the defect is assigned to the Development Team then the status of the defect is "ASSIGNED"

Assigned: Once the bug is posted by the tester, the test lead approves the bug and assigns the bug to developer team. There can be two scenario,first that the defect can directly assign to the developer, who owns the functionality of the defect.
Second, it can also be assigned to the Dev Lead and once it is approved with the Dev Lead, he or she can further move the defect to the developer.

Once the Development Team starts analyzing and works on the defect fix then the status is "OPEN"

Defect assigned to the dev team from test lead/ project lead/ project manager. if the dev team accepts the bug then the defect status will be changes that is open.

When developer makes necessary code change and verifies the change, then the status will be changed as "FIXED"

Fixed: When developer makes necessary code changes and verifies the changes then he/she can make bug status as 'Fixed'. This is also an indication to the Dev Lead that the defects on Fixed status .

If the defect is fixed and ready to test then the status is "TEST"

Retest: At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.

Once the latest build is pushed to the environment, Dev lead move all the Fixed defects to Retest. It is an indication to the testing team that the defects are ready to test.

If the defect remains same after the retest, then the tester changes the status to "REOPEN"

Reopened: If the bug still exists even after the bug is fixed by the developer, the tester changes the status to "reopened". The bug goes through the life cycle once again. Defect remains same after the rest then Tester post the defect using defect retesting document. Some times while doing retest same bug is encounter the bug was not fixed properly then status has reopen

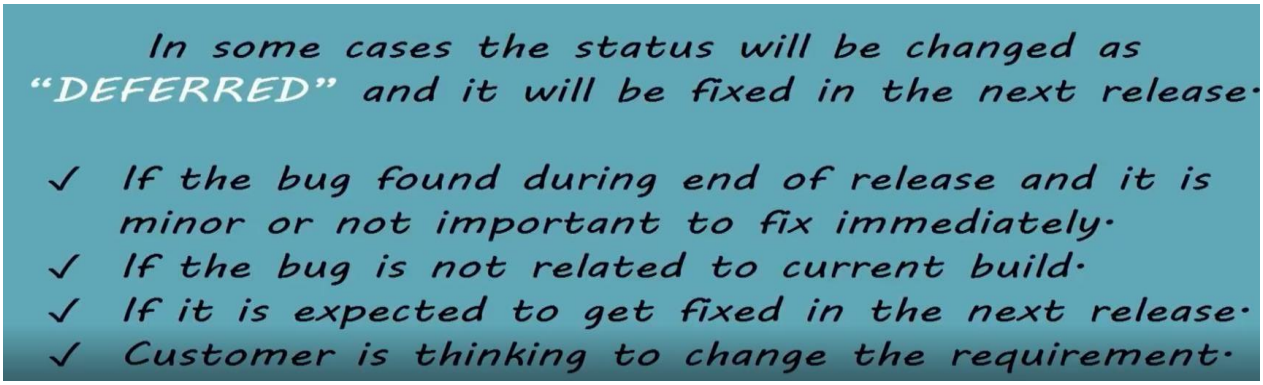
If there is no bug detected while doing retest in the software, then the bug is fixed and the status assigned is "VERIFIED"

After verified the fix, if the bug is no longer exits then the status of bug will be assigned as "CLOSED"

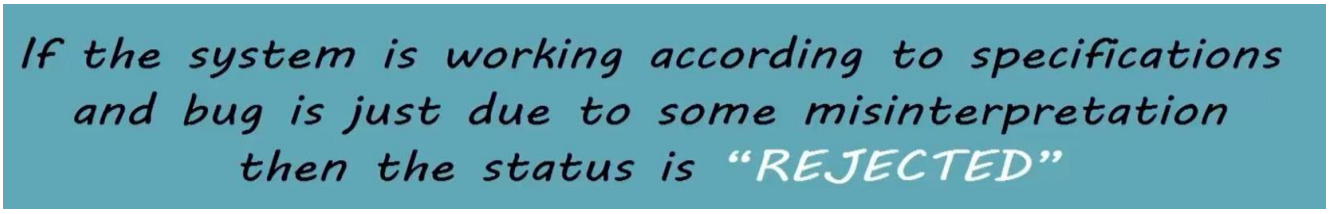
.***Closed:*** *Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, tester changes the status of the bug to "closed". This state means that the bug is fixed, tested and approved.*

If the defect is repeated then the status is changed to "DUPLICATE"

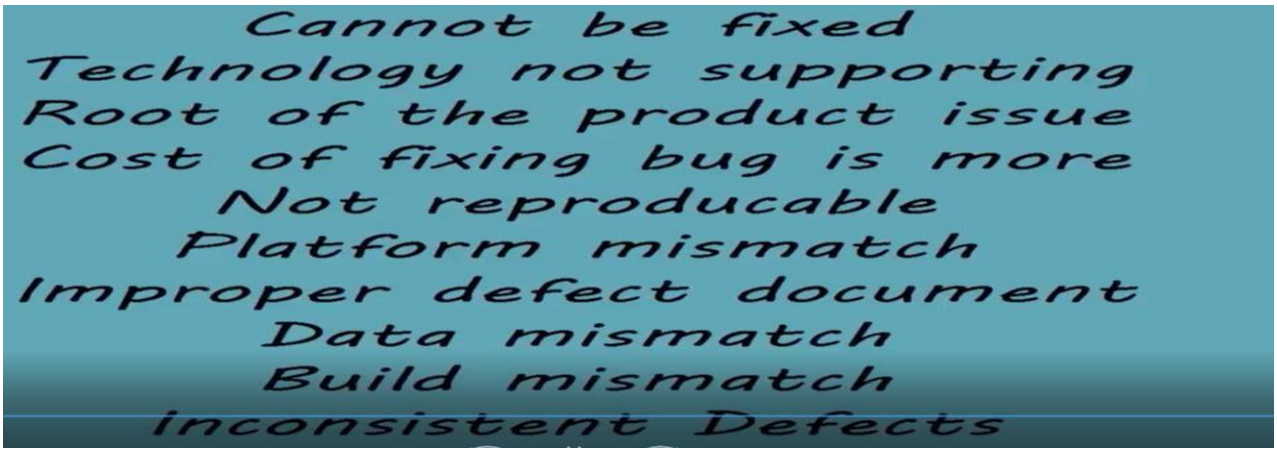
Duplicate : *If the bug is repeated twice or the two bugs mention the same concept of the bug, then the recent/latest bug status is changed to "duplicate***".**



Deferred: The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are *priority* of the bug may be low, lack of time for the release or the bug may not have major effect on the software.



Rejected: *If the developer feels that the bug is not genuine, developer rejects the bug. Then the state of the bug is changed to "rejected".*Some other status of bugs are



Not a bug/Enhancement: *The state given as "Not a bug/Enhancement" if there is no change in the functionality of the application. For an example: If customer asks for some change in the look and field of the application like change of color of some text then it is not a bug but just some change in the looks of the application.*

Defect Metrics

Defect Rejection Ratio

- (No. of defects rejected / Total no. of defects raised) * 100

Defect Leakage Ratio

- (Number defect missed/ total defects of software) * 100

In the above scenario, you can calculate the **defection rejection ratio** (DRR) is **20/84 = 0.238 (23.8 %).**

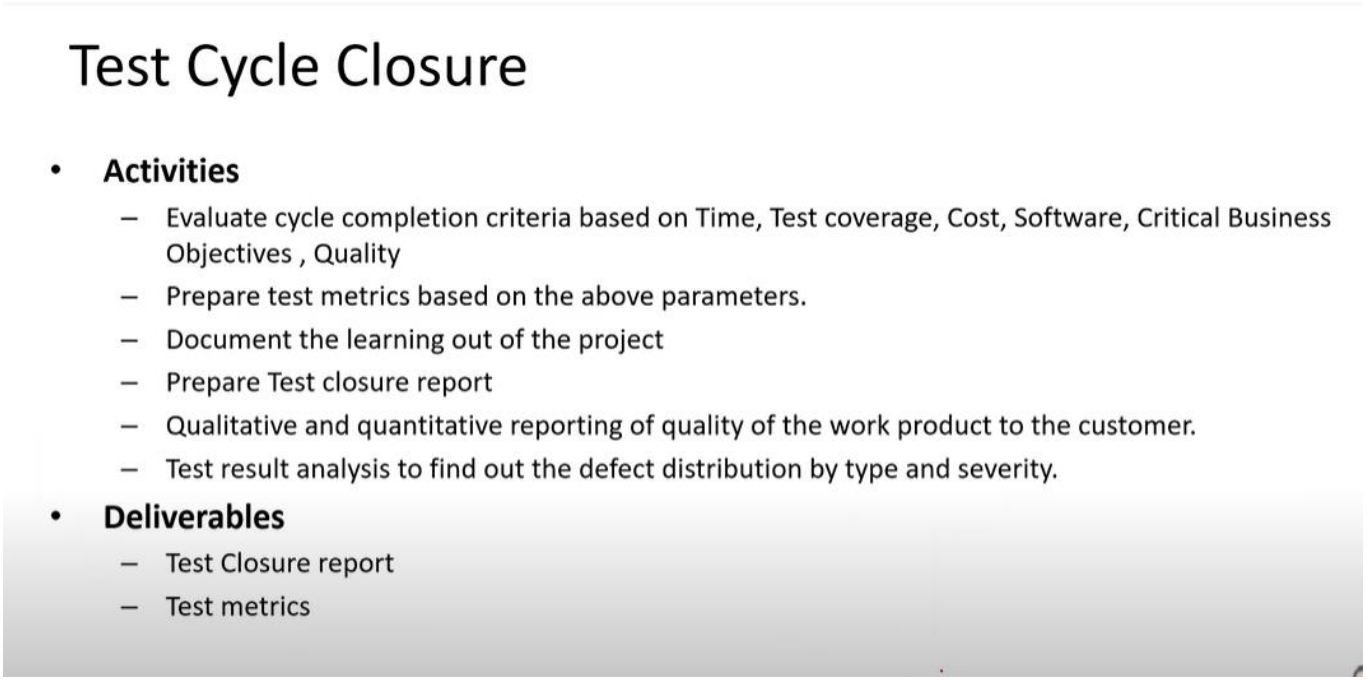
Another example, supposed the software has total **64** defects, but your testing team only detect **44** defects i.e. they missed **20** defects.

Therefore, you can calculate the **defect leakage ratio (DLR)** is 20/64 = **0.312 (31.2 %).**

$$Defect\ reject\ ratio = 23.8\%$$
$$Defect\ leakage\ ratio = 31.2\%$$

Defect leakage ratio?
When the tester could not identified the defect an application but customer found the defect an application and reported is called as defect leakage.

10.Test closing.

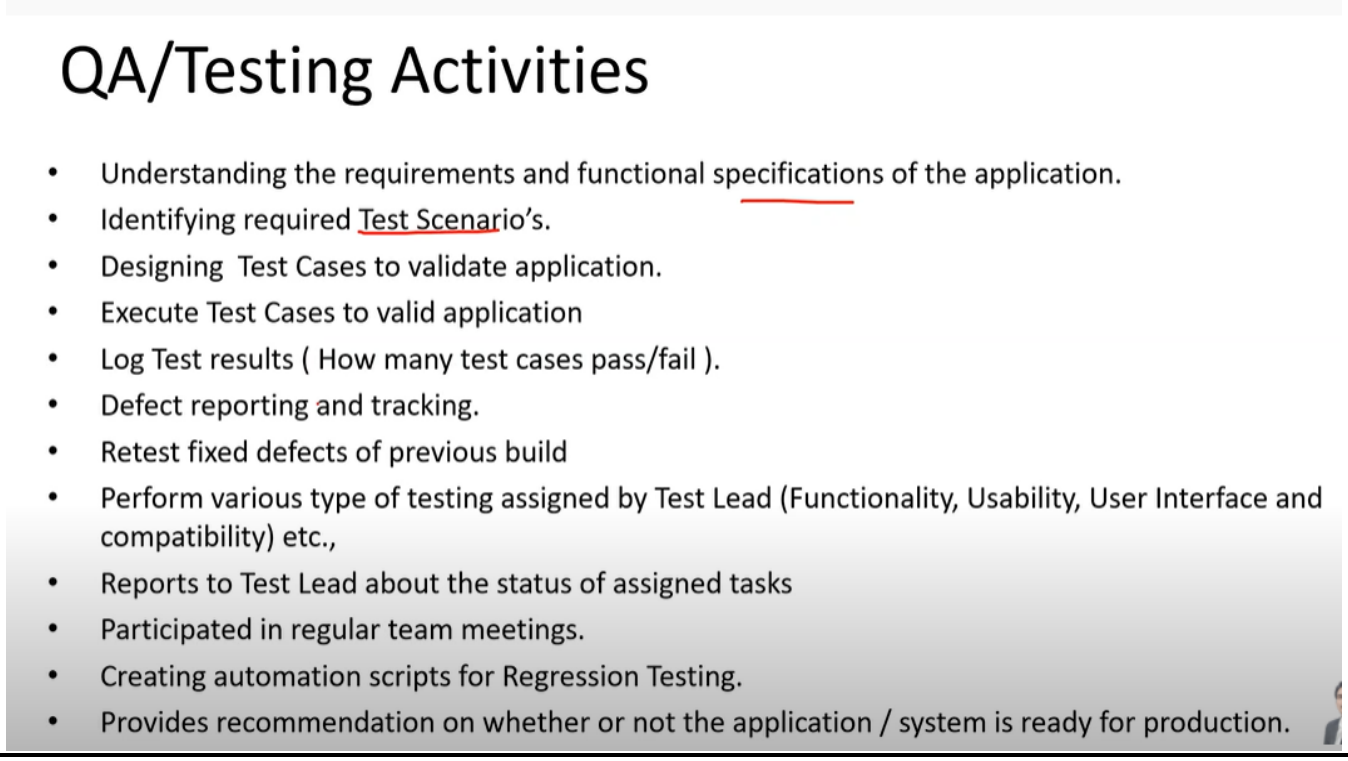


Tasks for the closure activities include the following:

– Check for the completion of the test. Whether all the test cases are executed or mitigated deliberately. Check there are no severity 1 defects opened.

– Do lessons learnt meeting and create lessons learnt document.(Include what went well, where are the scope of improvements and what can be improved).

*tester roles and resposibilities



Test Metrics

SNO	Required Data
1	No. Of Requirements
2	Avg. No. of Test Cases written Per Requirement
3	Total No.of Test Cases written for all Requirement
4	Total No. Of test cases Executed
5	No.of Test Cases Passed
6	No.of Test Cases Failed
7	No.of Test cases Blocked
8	No. Of Test Cases Un Executed
9	Total No. Of Defects Identified
10	Critical Defects Count
11	Higher Defects Count
12	Medium Defects Count
13	Low Defects Count
14	Customer Defects
15	No.of defects found in UAT

Test Metrics

- **% of Test cases Executed:**
  - No.of Test cases executed / Total No. of Test cases written) * 100
- **% of test cases NOT executed:**
  - (No.of Test cases NOT executed/Total No. of Test cases written) * 100
- **% Test cases passed**
  - (No.of Test cases Passed /Total Test cases executed) * 100
- **% Test cases failed**
  - (No.of Test cases failed / Total Test cases executed) * 100
- **%Test cases blocked**
  - (No.of test cases blocked / Total Test cases executed) * 100

Test Metrics

- **Defect Density: Number of defects identified per requirement/s**
  - No.of defects found / Size(No. of requirements)
- **Defect Removal Efficiency (DRE):**
  - (A / A+B) * 100
  - (Fixed Defects / (Fixed Defects + Missed defects)) * 100
    - A- Defects identified during testing/ Fixed Defects
    - B- Defects identified by the customer/Missed defects
- **Defect Leakage:**
  - (No.of defects found in UAT / No. of defects found in Testing) * 100
- **Defect Rejection Ratio:**
  - (No. of defect rejected /Total No. of defects raised) * 100
- **Defect Age:** Fixed date-Reported date
- **Customer satisfaction** = No.of complaints per Period of time

7. User Acceptance Testing (UAT)

User Acceptance Testing (UAT)/ **End-User testing/ Operational Acceptance Testing (OAT).cceptance testing** – User acceptance is a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon. This is beta testing of the product & evaluated by the actual end users. The main purpose of this testing is to validate the end to end business flow.

1. This testing is conducted by stakeholders/Customers /End users/ Client. Where two or more end users will be involved.
2. It is performed after **System Testing**, it is done all most of them major **defects** have been fixed.
3. **Where the product is evaluated as per the end users requirements.**
4. **Product is evaluated that whether it matches with business requirement or not and whether to accept or reject the product.**

5. This testing happens in "**the final phase of testing before moving the software application to Market or Production environment".** Such type of testing is executed by client in separate environment (similar to production environment) & confirms whether system meets the requirements as per requirement specification or not.
6. The Acceptance Testing is Black Box Testing**,** means UAT users doesn't aware of internal structure of the code. They just specify the input to the system & check whether systems respond with correct result.

During **UAT**, actual software users **test** the software to make sure it can handle required tasks in real-world scenarios, according to specifications.

The main purpose of this testing is to validate the end to end business flow. It does NOT focus on the cosmetic errors, Spelling mistakes or System testing. UAT has 2 types of testing **A) Alpha Testing B) Beta Testing.**

Alpha testing: Alpha testing done to **find out the all bugs in a software before releasing product to customers.**

- First phase of testing in Customer Validation.
- White box and / or Black box testing techniques are involved
- Issues/ Bugs are logged into the identified tool directly and are fixed by developer at high priority

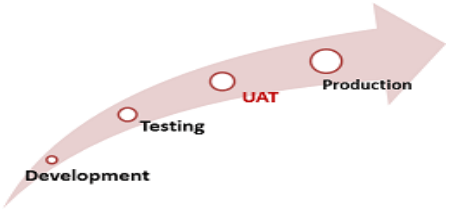
Beta testing:

- Beta testing is done **after releasing product or software to market.**
- Second phase of testing in Customer Validation.
- Performed in real environment, and hence activities cannot be controlled.
- Functionality,Usability,Reliability,Security testing are all given equal

- importance to be performed
- Only Black box testing techniques are involved.
- Build released for Beta Testing is called Beta Release.
- Issues/Bugs are collected from real users in the form of suggestions/feedbacks and are considered as improvements for future releases.

Need of User Acceptance Testing: Once software has under gone Unit, Integration and System testing the need of Acceptance Testing may seem redundant. **But Acceptance Testing is required because.**

- Developers done software code based on requirements document which is their "own" understanding the requirements and may not actually be what the client needs from the software.
  - Requirements changes during the course of the project may not be



communicated effectively to the developers.

How is UAT Performed?

Business Requirements must be available.

- Application Code should be fully developed
- Unit Testing, Integration Testing & System Testing should be completed.
- No Showstoppers, High, Medium defects in System Integration Test Phase -Only Cosmetic error are acceptable before UAT.
- Regression Testing should be completed with no major defects.
- All the reported defects should be fixed and tested before UAT
- Traceability matrix for all testing should be completed
- UAT Environment must be ready
- Sign off mail or communication from System Testing Team that the system is ready for UAT execution.

*7.What is Quality Assurance (QA), Quality Control QC?

QA always concertation **on processing**. , QE/QC always concertation **on product.**

"QA is a systematic and scientific approach of monitoring and improving the software development process."

*9. Software Test Automator Role?

An automated test engineer should have good understanding of **what he needs to test- GUI designs, load or stress testing.**

He should be proficient in automation of software testing, and he should be able to **design test suites accordingly.**

A software test automator should be comfortable using various kinds of automation tools and should be capable of **upgrading their skills with changing trends.**

He should **also have programming skills.**

He is able to **write test scripts without any issues.** The responsibilities of a tester at this position are as follows:

1. He should be able to understand the requirement and design test procedures and test cases for automated software testing.
2. Design automated test scripts that are reusable.
3. Ensure that all automated testing related activities are carried out as per the standards defined by the company.

What are the common problems with software automation?

1. Purchasing the license of tool (QTP, selenium, QC, LR)
2. Lack of skilled Tester to run the tool
3. Expectation that automated tests will find a lot of new defects
4. Maintenance of automated tests
5. Technical problems of tools
6. Why you choose automated testing over manual testing?

1. Frequency of use of test case
2. Time Comparison
3. Re-usability of Automation Scripts.
4. Adaptability of test case for automation.
5. Exploitation of automation tool.
6. Test engineer productivity.
7. Coverage of regression testing.
8. Consistency in testing.
9. Test interval reduction
10. Reduced software maintenance cost
11. Increased test effectiveness.

Tell me abuot your daily activities as a test engineer...

1) undestanding the frs/brs doc and use cases
2) if any ambiguity is there in requirment report to team lead /Business analyst

3) after getting the clarification identify the test scenario and prepare the test scenarios document

4)prepare the GUI test case documnet

5) write the GUI test cases in exel sheel

6) Build is ready for testing we need to do the sanity testing whether forther testing build is stable or not

7) perform the functional testing with the positive and nagitive inputs

8) 8)if any defect are identified the report/assign to developer

9)after fixing the defect retest the same componant/build

10)perform the regression testing

11)perform the compatability testing as per the requirment

12)generate the test report and send to the Lead/PM

1. Understanding the BRS and Usecases Document
2. Giving system demo to PM, System analyst, designer, Dev lead.
3. Preparing the Test Actions in xls sheet.
4. Updating the Test Actions based on review commnets by System analyst/Business Analyst.
5. Preparing the Testcases and Datasets(System level and global level datasets) in word document
6. updating the Test Cases based on review comments by System analyst.
7. Installing the application-Testing environment set
8. Performing Functional,GUI,System,Compatibility testing(If necessary), Regression testing based on Test cases
9. Preparing the defect report, Bug tracking list and sending daily status report to PM, leads.

- Stand-up meeting with the project team to discuss the **tasks** (to do or done);
- Performing **tasks** assigned by PM;
- Writing **test** cases;
- Reporting new defects;
- Reviewing fixed bugs;
- Communication with developers;

***10. What is the build and release?**

BUILD: A "build" is given by **development team** to **the test team**. A software application is released to **test** in a number of stages called **builds**. It depends upon the application that how many **builds** will be released before it actually goes on air. Every **build** comprises of a set of new features.

RELEASE: A build when tested and certified by the test team it is given to the customers

as "release". A release is the final version of an application. A new or upgraded application. A release is preceded by the distribution of alpha and then beta versions of the software.

Gamma : Gamma Testing is done when software is ready for release with specified requirements, this testing done directly by skipping all the in-house testing activities. The software is almost ready for final release. No feature development or enhancement of the software is undertaken and tightly scoped bug fixes are the only code.

*12. What are the key challenges of software testing?

1.Understanding requirements, Domain knowledge and business user perspective understanding.
2. Testing always under time constraint.
3. Time pressures.
4. Which tests to execute first?
5. Testing the Complete Application.
6. Regression testing.
7. Lack of skilled testers.
8. Changing requirements.
9. Lack of resources, tools and training.
10. Application is not testable.
11. Ego problems.
12. Defect in defect tracking system
13. Miscommunication or no Communication.
14. Bug in software development tools.

*14. Explain Bug Leakage, Bug Release, Bug Age?

Bug (defect) Leakage: When a Bug (defect) found by customer / end user missed by the testing team to detect, while testing the software it is called bug leakage.

Bug (defect) Release: When we released any version of an application or software **with a group of some known bugs or issues it is called as** bug release**.** These released issues or bugs are also described in the Release Notes. These issues or bugs are generally **low severity and low priority bugs.**

Bug (defect) Age: Bug (defect) Age can be defined as the time interval between "**the date of defect is detected** and **the defect was fixed/closed.**

```
Defect Age = Date of defect closure - Date of defect detection

Reported Date: 1st Jan 2017
Defect Closure Date: 5th Jan 2017.

Defect Age = 5 days
```

15. What is data driven testing?

Data Driven is an automation testing part, in which test input or output values, these values are read from data files. It is performed when the values are changing by the time. The different data files may include data pools, csv files, Excel files. The data is then loaded into variables in recorded or manually coded scripts. For data driven testing we use Parameterzing and Regular expression Technique.

Ex: To evaluate login functionality, we use different user name and password combinations, variables are used to access different username and password. The list of username and password are stored in a data table or excel sheet.

*16. **Explain the difference between debugging and testing?**

Testing: Testing is a process of **finding defects.**

Debugging: debugging is a process of **finding the cause of those defects and fix them.**

Debugging is not testing.

Debugging always occurs as a consequence of testing activity.

8) What are Software Test Documents?

i) Test Policy: A high level (company level) document describes principles, approach and major objectives of the organization regarding Testing.

ii) Test Strategy: A high level document of the Test Levels to be performed and the Testing within those levels for an Organization.

iii) Test Plan: A document describing the scope, approach, resources, and schedule of intended activities.

iv) Test Scenario: An item or event of a component or system that could be verified by one or more Test cases.

v) Test Case: A set of input values, execution preconditions, expected result and execution post conditions developed for a particular objective or Test condition.

vi) Test Data: Data that exits before a test is executed and that effects or is effected by the component or System under test.

vii) Defect Report: A document reporting of ant flaw in a component or System that can cause

the component or system to fail to perform its required function.

viii) Test Summary Report: A document summarizing testing activities and Result. It also contains evaluation of the corresponding test items against exit criteria.

SOFTWARE QUALITY CONTROL (SQC) **is a set of activities for ensuring quality in software products. Software Quality Control is limited to the Review/Testing phases of the Software Development Life Cycle and the goal is to ensure that the products meet requirements.**

It includes the following activities: *Reviews*

- Requirement Review
- Design Review
- Code Review
- Deployment Plan Review
- Test Plan Review
- Test Cases Review

- *Testing*
- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

The process of Software Quality Control (SQC) is governed by Software Quality Assurance (SQA). While SQA is oriented towards **prevention**, SQC is oriented towards **detection**.

SOFTWARE QUALITY ASSURANCE (SQA) is a set of activities for ensuring quality in software engineering processes (that ultimately result in the quality of software products).

It includes the following activities:
Process

- definition and implementation
- Auditing
- Training

Processes include:

- Software Development Methodology
- Project Management
- Configuration Management
- Requirements Development/Management.
- Estimation
- Software Design
- Testing *etc*……..

Once the processes have been defined and implemented, Quality Assurance has the following responsibilities:

- identifying weaknesses in the processes
- correcting those weaknesses to continually improve the processes

The quality management system under which the software system is created is normally based on one or more of the following models/standards:

- CMMI
- Six Sigma
- ISO 9000

Note: There are many other models/standards for quality management but the ones mentioned above are the most popular.

Software Quality Assurance encompasses the entire software development lifecycle and the goal is to ensure that the development and/or maintenance processes are continuously improved to produce products that meet specifications/requirements.

The process of Software Quality Control (SQC) is also governed by Software Quality Assurance (SQA). Read Differences between Software Quality Assurance and Software Quality Control

What is Automated Testing?

The process of performing testing automatically which reduces the human intervention this is automation testing. The automation testing is carried out with the help of the some automation tool like QTP, Selenium, WinRunner etc. In automation testing we use a tool that runs the test script to test the application; this test script can be generated manually or automatically. When testing is completed then tools automatically generate the test report and report.

What are the fields in a bug report?

Following important fields should be included in a good bug report:

1. A unique ID

2. Defect description – a short describing what the bug is

3. Steps to reproduce – details about how to arrive at the error, exact test data, the time at which defect was found(if applicable) environment – any information that will help re-encounter the issue

4. Module/section of the application (if applicable)

5. Severity

6. Screenshot

7. Responsible QA – in case of any follow-up questions regarding this issue.

Can you tell me about your QA interview experience?

My actuality job position is QA Engineer. As a QA person, I used a variety of platforms and operating systems including Windows XP and windows 10.

I have experience with testing applications developed in Java and selenium webdriver.I have

also tested web Applications, window applications and non html element (videos) to make sure that the applications also works accordingly.

- Written Test Plans, Test Cases, Checklist.
- Tested Web-based applications as well as client server applications.
- Strong knowledge of Smoke Testing, Functional Testing, Backend Testing, BlackBox Testing, Integration Testing, Regression Testing and UAT (User Acceptance Testing) Testing
- Worked in databases like Oracle sql (wrote SQL queries to retrieve data from the database).
- I participated in meetings.

Automation testing

1. What is Automation?

Automation is a process which performs repetitive tasks with minimum human assistance to reduce human efforts. If a person has to repeat a job more than once, then instead of doing the same process, again and again, we create a small piece of software or a bot that does the job to reduce human efforts.

2. What is Automation Testing?

Automation testing is the process of testing software or application using an automation testing tool to find the defects. In this process, executing the test scripts and generating the results are performed automatically by automation tools.

3. Why is Automation Testing required?

- It is required when we have a huge amount of regression test cases.
- It is required to save time & money
- It is required to increase the test coverage
- It is required to run tests anywhere & anytime
- It is required to generate robust reports
- It is required to test non-functional aspects of an application
- It is required when we run tests with multiple sets of data
- It is required when testing manually is impossible
- It is required to test on several different hardware or software platforms and configurations

4. Is it possible to achieve 100% automation?

It is impossible to automate everything. Achieving 100% automation is difficult because there are some scenarios where a registration page has a captcha or some test cases which we don't execute often. By automating these types of test cases will not add value to the automation.

5. Can we automate CAPTCHA or can we automate ReCAPTCHA?

It's not possible to automate CAPTCHA or ReCAPTCHA.

CAPTCHA stands for Completely Automated Public Turing test to tell Computers and Humans Apart. The reason for creating a CAPTCHA is to control bots.

If someone is capable of automating CAPTCHA then it means that the CAPTCHA is not able to tell computers and humans apart. So the idea behind CAPTCHA is failed.

If someone is able to automate CAPTCHA then it's not a CAPTCHA.

6. How do you handle CAPTCHA in your test automation project?

To automate an application which uses CAPTCHA, you have to consult your development team to provide a workaround.

- To keep the CAPTCHA static in your test environment
- To disable the CAPTCHA in your test environment

By making CAPTCHA static in your test environment helps you to automate CAPTCHA by providing one specific value as a CAPTCHA on every run.

7. What are the benefits of Automation Testing?

This is one of the common interview questions in any Automation testing job.

1. Saves time and money. Automation testing is faster in execution.
2. Reusability of code. Create one time and execute multiple times with less or no maintenance.
3. Easy reporting. It generates automatic reports after test execution.
4. Easy for compatibility testing. It enables parallel execution in the combination of different OS and browser environments.
5. Low-cost maintenance. It is cheaper compared to manual testing in a long run.
6. Automated testing is more reliable.
7. Automated testing is more powerful and versatile. Automation tools allow us to integrate with Cross Browser Testing Tools, Jenkins, Github etc.,
8. It is mostly used for regression testing. Supports execution of repeated test cases.
9. Minimal manual intervention. Test scripts can be run unattended.
10. Maximum coverage. It helps to increase the test coverage.

8. When do you prefer Manual Testing over Automation Testing?

Refer this link

9. When do you prefer Automation Testing over Manual Testing?

Refer this link

10. List some advantages and disadvantages of Manual testing?

Advantages:

1. Manual testing can be done on all kinds of applications
2. It is preferable for short life cycle products
3. Newly designed test cases should be executed manually
4. The application must be tested manually before it is automated
5. It is preferred in the projects where the requirements change frequently and for the products where the GUI changes constantly
6. It is cheaper in terms of initial investment compared to Automation testing
7. It requires less time and expense to begin productive manual testing
8. It allows testers to perform ad-hoc testing
9. There is no necessity to the tester to have knowledge on Automation Tools

Disadvantages:

1. Manual Testing is time-consuming mainly while doing regression testing.

2. Expensive over automation testing in the long run

11. List some advantages and disadvantages of Automation testing
Advantages:

1. Automation testing is faster in execution
2. It is cheaper compared to manual testing in the long run
3. Automated testing is more reliable
4. Automated testing is more powerful and versatile
5. It is mostly used for regression testing
6. It does not require human intervention. Test scripts can be run unattended
7. It helps to increase the test coverage

Disadvantages:

1. It is recommended only for stable products
2. Automation testing is expensive initially
3. Most of the automation tools are expensive
4. It has some limitations such as handling captcha, fonts, color
5. Huge maintenance in case of repeated changes in the requirements
6. Not all the tools support all kinds of testing. Such as windows, web, mobility, performance/load testing

11. Difference between Manual Testing & Automation Testing (Automation Testing vs Manual Testing)?
Refer this link
12. What type of tests have you automated?
Our main focus is to automate test cases to do *Regression testing, Smoke testing, and Sanity testing.* Sometimes based on the project and the test time estimation, we do focus on End to End testing.
13. How do you identify the test cases which are suitable for automation?
Automation testers should understand what to actually automate. Identifying the appropriate test cases for automation plays a vital role in the success of automation testing.

- Tests that run for multiple builds.
- Tests that leads to human error.
- Tests that require multiple sets of data.
- Tests that involve high risk.
- Tests that are impossible to perform manually.
- Tests that take a lot of time and effort when performing manually
- Tests that run on several different hardware or software platforms and configurations.

14. How many test cases you have automated per day?
It depends on Test case scenario complexity and length. I did automate 2-5 test scenarios per day when the complexity is limited. Sometimes just 1 or fewer test scenarios in a day when the complexity is high.
15. What type of test cases you won't pick up to automate?
Before picking up the test cases to automate, I do check whether the application is stable or not. So based on this, I don't pick up test cases when the AUT changes frequently and the test cases

which I run rarely and run only one time. Also, I don't automate while doing usability and exploratory testing.

16. Name some popular automation tool that you are aware of?

Some of the popular automation tools are Selenium WebDriver, QTP/UFT, JMeter, LoadRunner, Ranorex, etc.,

17. How do you select an automation tool?

Selecting an automation tool is essential for test automation. There are a lot of automation testing tools on the market. Some of the factors involved in selecting an automation tool are as follows.

- Supports for your platforms and technology
- Ease of use, setup, and accessibility
- Good debugging facility
- Type of support available for the tools like documentation, tutorials, training etc.,
- Cost and budget
- CI, DevOps support
- Good reporting system

18. What is Framework?

A framework defines a set of rules or best practices which we can follow in a systematic way to achieve the desired results.

19. Why Framework?

In a test automation project, we do perform different tasks by using different types of files. To organize and manage all the files and to finish all the tasks in a systematic approach we use a framework.

20. Can we do Automation testing without a framework?

We can perform automation testing without a framework. A framework is a set of rules or best practices which we can follow in a systematic way to achieve the desired results. If we understand the automation tool, then we can do automation testing without a framework but if we create and follow a framework then it will be helpful in many ways.

21. Have you created any Framework?

If you are a beginner: *No, I didn't get a chance to create a framework. I have used the framework which is already available.*

If you are an experienced tester: *Yes, I have created a framework (Or) No, but I have involved in the creation of the framework.*

22. What are the advantages of the Automation framework?

The advantage of Test Automation framework

- Reusability of code
- Maximum coverage
- Recovery scenario
- Low-cost maintenance
- Minimal manual intervention
- Easy Reporting

23. What are the different types of frameworks?

There are different types of automation frameworks and the most common ones are:

- Modular Testing Framework

- Data Driven Testing Framework
- Keyword Driven Testing Framework
- Hybrid Testing Framework
- Behavior Driven Development Framework

Detailed Explanation: Types of Framework

24. What are some coding practices to follow while writing automation scripts or framework?

Some of the good coding practices are:

- Naming standards should follow
- Comments should be added
- Place reusable functions in a separate file
- Avoid duplicate code
- Add appropriate assertions

Selenium:selenium is a automation tool, automate the web applications. Selenium doesn't test desk top and other applications.

Selenium is not just a single tool, it is a suite of tools. Selenium IDE, Selenium RC, Selenium WebDriver, and Selenium Grid are the tools in this Selenium's tool suite. Selenium automates **browsers**. Selenium's primary purpose is Testing **Web Applications**, and It also supports Automating **Web-based administration tasks.** Selenium is an Open source software for **Functional & Regression** Testing of Web Applications in System Testing and Acceptance Testing levels.

Feactures:

Selenium supports various Operating systems to conduct testing. MS Windows , unix Linux ,Macintosh.

• Selenium supports various programming languages to write & execute test cases.

Ex: Java , Python ,C#, Ruby ,JavaScript and Kotlin.

• Selenium supports various browsers to conduct testing.

ex: Google Chrome, Mozilla Firefox ,Microsoft IE/Edge, Opera and Safari.

• Selenium automation test cases(test suits) are reusable and can be tested across multiple browsers, and operating systems.

• Selenium is not an all-inclusive test tool, it needs **third-party frameworks and add-ons** to broaden the scope of testing.

Selenium, integrated with tools testNG , junit for managing test cases and generate the reports.

• Selenium **supports Data-Driven Testing, Batch Testing, Cross Browser Testing, and Database Testing.**

Note: Selenium IDE doesn't support programming, Selenium IDE supports Chrome and Firefox browsers only, Selenium Grid supports only Test execution, and Selenium RC was outdated and removed from Selenium's latest versions.

Selenium Components/Selenium's Tool Suite.

Selenium is not just a single tool but a suite of software, each with a different approach to supporting automation testing. It comprises four major components which include:

History of the Selenium Project

• Selenium was first launched in 2004.

• In 2006, Selenium WebDriver was launched at Google.

• In 2008, the whole selenium team decides to merge Selenium WeDriver with Selenium RC, in order from a power tool called Selenium 2.0.

Selenium 1.0
(Selenium IDE, Selenium RC, Selenium Grid)

Selenium 1.0 + Selenium WebDriver = Selenium 2.0

Selenium 2.0
(Selenium IDE, Selenium RC, Selenium WebDriver, Selenium Grid)

Selenium 3.0
(Selenium IDE, Selenium WebDriver, Selenium Grid)

Note: Selenium RC (Remote Control) was removed from this version.

Selenium 4.0 coming
(Selenium IDE, Selenium WebDriver, Selenium Grid)

The suite package constitutes of the following sets of tools:

1. ***Selenium Integrated Development Environment (IDE)***
2. ***Selenium Remote Control (RC)***
3. ***Selenium WebDriver***
4. ***Selenium Grid.***

1. Selenium Integrated Development Environment (IDE)

Selenium IDE was developed by Shinya Kasatani.

1. Selenium IDE doesn't support programming, it is record and play back tool.
2. It is used to write and execute test cases(test suites).
3. Test Engineer can edit test suites (Add/Update/Delete).

4. Test Engineer can debug test cases, and add comments.
5. Selenium IDE should only be used as a prototyping tool, not an overall solution for developing and maintaining complex test suites.
6. *Selenium IDE supports only Mozilla Firefox web browser.*
7. *Firefox plug-in, Selenium IDE supports only.*
8. *Creates test scripts could be executed only on Firefox.*
9. No support for Data-Driven testing.
10. No centralized maintenance of objects/elements.
11. It doesn't generate detailed test results.

A few more loopholes make this tool inappropriate to be used for complex test scripts. Thus, other tools introduced like Selenium RC, WebDriver.

2. Selenium Remote Control (RC): **Selenium RC** came into existence in **2006**.Selenium RC is an important component in the Selenium test suite. It is a testing framework that enables a QA or a developer to write test cases in any programming language in order to automate UI tests for web applications against any HTTP website.

- Execution of test scripts is time-consuming as Selenium RC uses JavaScript commands as instructions to the browser. This results in slow performance.

3. Selenium WebDriver: WebDriver is a web automation framework that allows you to execute your tests against different browsers, not just Firefox, Chrome (unlike Selenium IDE).

WebDriver also enables you to **use a programming language** in creating your test scripts (not possible in Selenium IDE).

You can now use **conditional operations** like if-then-else or switch-case. You can also perform looping like do-while.

- **Multiple Browser Support**: Selenium WebDriver supports a diverse range of web browsers such as Firefox, Chrome, Internet Explorer, Opera and many more. It also supports some of the non-conventional or rare browsers like HTMLUnit.

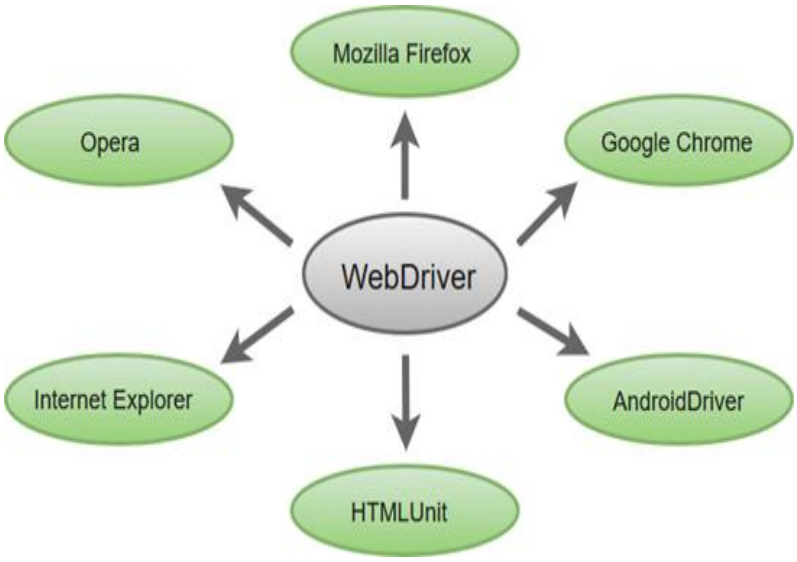
Following programming languages are supported by WebDriver

- Java .Net PHP Python Perl

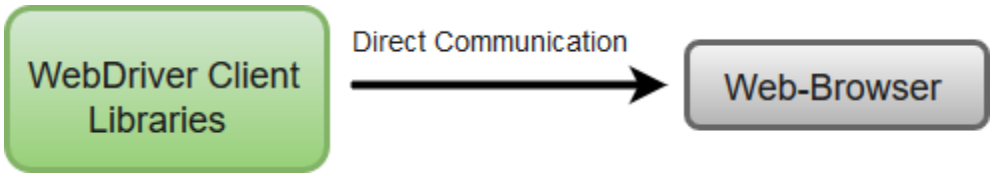
Ruby Selenium WebDriver- Features

Some of the most important features of Selenium WebDriver are:

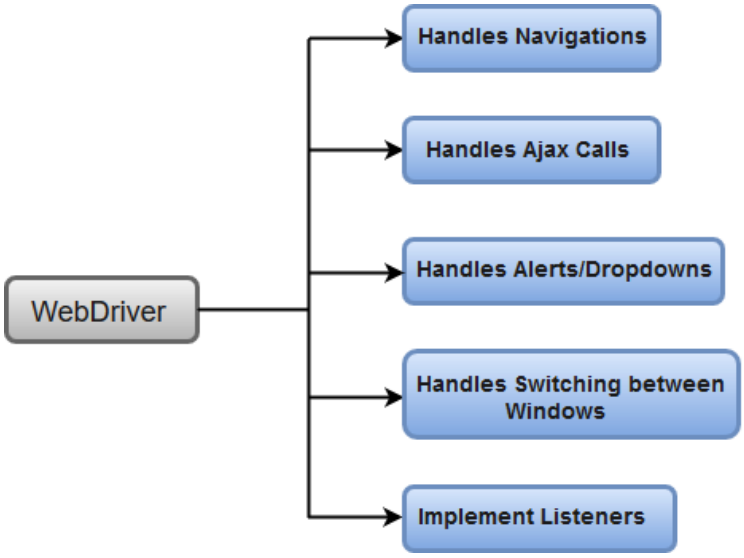
- **Multiple Browser Support**: Selenium WebDriver supports a diverse range of web browsers such as Firefox, Chrome, Internet Explorer, Opera and many more. It also supports some of the non-conventional or rare browsers like HTMLUnit.



- **Multiple Languages Support**: WebDriver also supports most of the commonly used programming languages like Java, C#, JavaScript, PHP, Ruby, Pearl and Python. Thus, the user can choose any one of the supported programming language based on his/her competency and start building the test scripts.
- **Speed**: WebDriver performs faster as compared to other tools of Selenium Suite. Unlike RC, it doesn't require any intermediate server to communicate with the browser; rather the tool directly communicates with the browser.



- **Simple Commands**: Most of the commands used in Selenium WebDriver are easy to implement. For instance, to launch a browser in WebDriver following commands are used:
  WebDriver driver = **new FirefoxDriver();** (Firefox browser)
  WebDriver driver = **new ChromeDriver();** (Chrome browser)
  WebDriver driver = **new InternetExplorerDriver();** (Internet Explorer browser)
- **WebDriver- Methods and Classes**: WebDriver provides multiple solutions to cope with some potential challenges in automation testing.
  WebDriver also allows testers to deal with complex types of web elements such as checkboxes, dropdowns and alerts through dynamic finders.



Speed:

WebDriver is faster than Selenium RC since it speaks directly to the browser uses the browser's own engine to control it.

WebDriver interacts with page elements in a more realistic way. For example, if you have a disabled text box on a page you were testing, WebDriver really cannot enter any value in it just as how a real person cannot.

Selenium 2.0 (Web driver support) replaced **RC in 2011**.
Selenium 3.0 birthed in **Oct' 2016**.
The core Selenium Webdriver API will work as an interface. Browser vendors will independently provide client drivers.

Even the Firefox now has its GECKO driver implementation of the Webdriver 3.0 APIs. GECKO driver follows the W3C WebDriver spec. You can find the latest API specification from here.

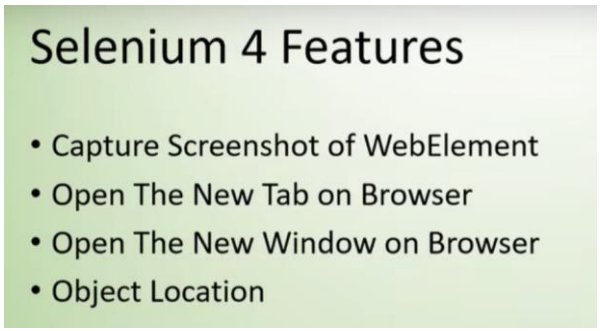
2. Support for Safari is available via Apple's Safari driver.
And for IE, it is the Edge driver that integrates with the Selenium Webdriver.

3. Some other changes are as follows.

- The minimum JRE version is 8.0.
- Supported IE version is >= 9.0.
- Brought back support of Firefox 47.0.1 and earlier versions
- For the newer Firefox version, use the GECKO driver.
- WebDriver is a tool for testing web applications **across different browsers** using different programming languages.

- You are now able to make powerful tests because WebDriver **allows you to use a programming language** of your choice in designing your tests.
- WebDriver is **faster than Selenium RC** because of its simpler architecture.
- WebDriver **directly talks to the browser** while Selenium RC needs the help of the RC Server in order to do so.
- WebDriver's API is more **concise** than Selenium RC's.
- WebDriver **can support Html Unit** while Selenium RC cannot.
- The only drawbacks of WebDriver are:
- It **cannot readily support new browsers**, but Selenium RC can.
- It **does not have a built-in command** for automatic generation of test results.

Selenium 4 version:



Not: we are using the API webdrivermanager added in maven repository we no need to use the system.set.property and browser specific drivers.

In program webdrivermanager.chromedriver().set();
In this webdriver manager class can be imported from "io.github.bonigarcia.wdm.webdrivermanager". package.

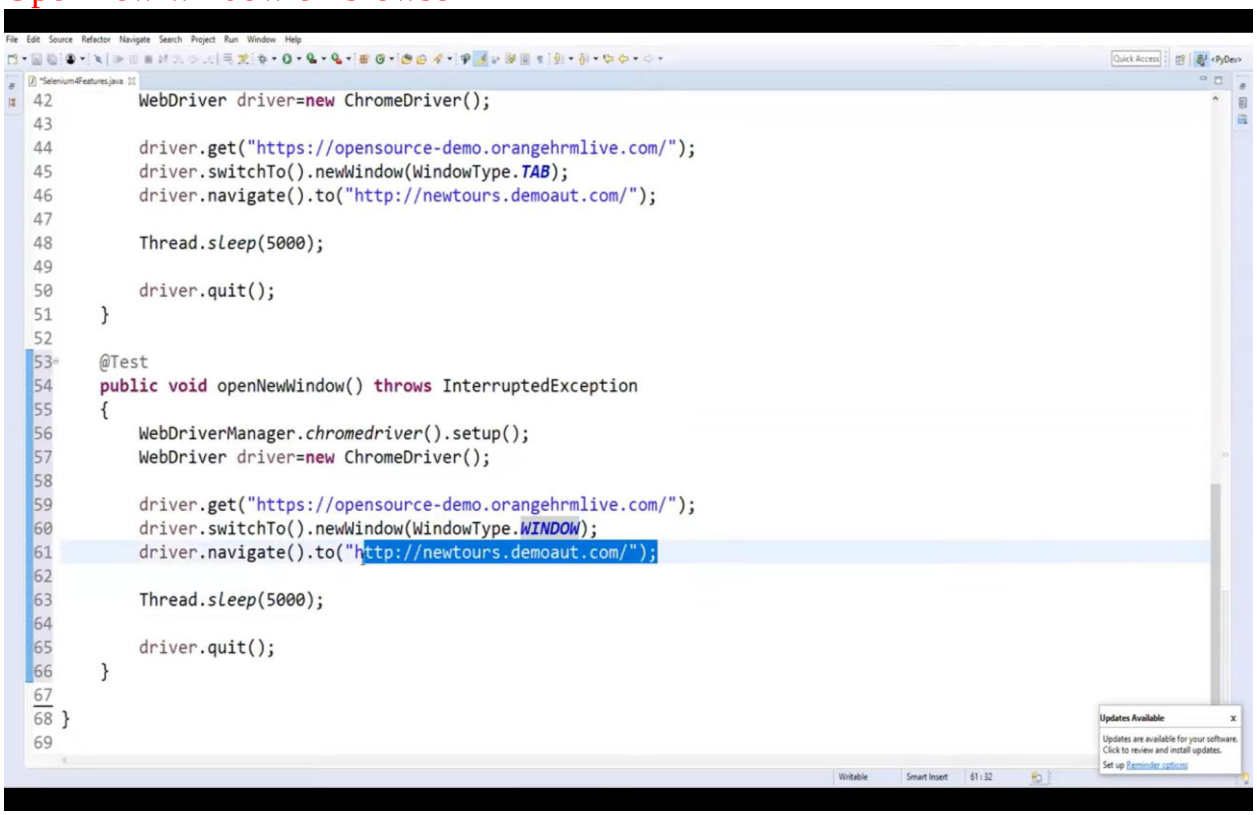
Take The Screen Shot Of Specific Element

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;

import io.github.bonigarcia.wdm.WebDriverManager;

public class Selenium4Features {

    @Test
    public void screenshotTest() throws IOException
    {
        WebDriverManager.chromedriver().setup();
        WebDriver driver=new ChromeDriver();

        driver.get("https://opensource-demo.orangehrmlive.com");
        driver.manage().window().maximize();

        WebElement logo=driver.findElement(By.xpath("//div[@id='divLogo']//img"));

        File file=logo.getScreenshotAs(OutputType.FILE);

        File destfile=new File("logo.png");
        FileUtils.copyFile(file,destfile );

        driver.quit();
    }
```

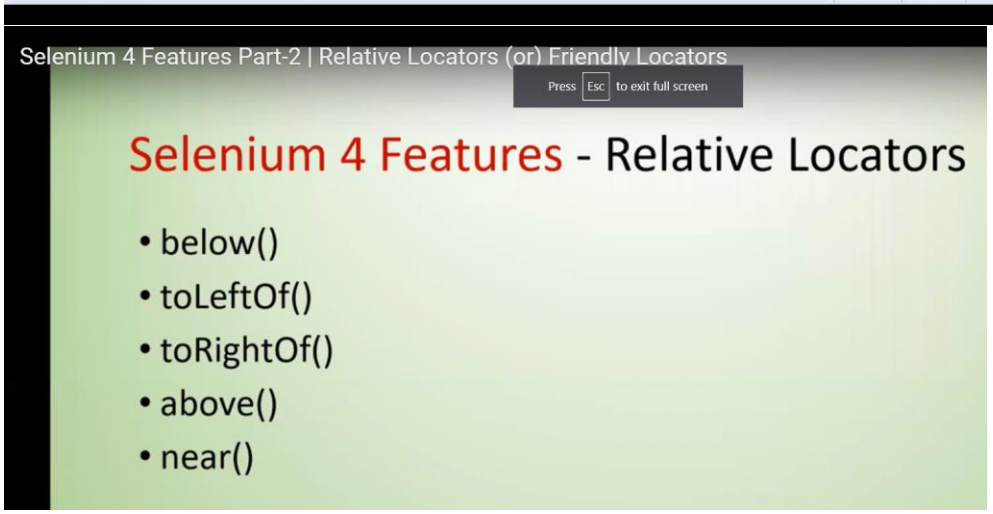
Open new tab on browser

```
    driver.get("https://opensource-demo.orangehrmlive.com");
    driver.manage().window().maximize();

    WebElement logo=driver.findElement(By.xpath("//div[@id='divLogo']//img"));

    File file=logo.getScreenshotAs(OutputType.FILE);

    File destfile=new File("logo.png");
    FileUtils.copyFile(file,destfile );

    driver.quit();
}

@Test
public void openNewTab()
{
    WebDriverManager.chromedriver().setup();
    WebDriver driver=new ChromeDriver();

    driver.get("https://opensource-demo.orangehrmlive.com/");
    driver.switchTo().newWindow(WindowType.TAB);
    driver.navigate().to("http://newtours.demoaut.com/");

    Thread.sleep(5000);
}
```

Open new window on browser



```
64
65          driver.quit();
66      }
67
68
69      //@Test
70          public void Location() throws IOException, InterruptedException
71          {
72              WebDriverManager.chromedriver().setup();
73              WebDriver driver=new ChromeDriver();
74
75              driver.get("https://opensource-demo.orangehrmlive.com");
76              driver.manage().window().maximize();
77
78              WebElement logo=driver.findElement(By.xpath("//div[@id='divLogo']//img"));
79
80              System.out.println("Height:"+logo.getRect().getDimension().getHeight());
81              System.out.println("Width:"+logo.getRect().getDimension().getWidth());
82
83              System.out.println("X Location:"+logo.getRect().getX());
84              System.out.println("X Location:"+logo.getRect().getY());
85
86              Thread.sleep(5000);
87
88
89          }
90
91
```




```
27          driver.get("https://automationbookstore.dev/");
28          driver.manage().window().maximize();
29
30      }
31
32      @AfterClass
33      public void tearDown()
34      {
35          driver.close();
36      }
37
38      //@Test(description="Test Book5 is left of Book6 & below Book1")
39      public void test1()
40      {
41          WebElement book5=driver.findElement(RelativeLocator.withTagName("li").toLeftOf(By.id("pid6")).below(By.id("pi
42          String id=book5.getAttribute("id");
43          System.out.println(id);
44          Assert.assertEquals("pid5",id);
45      }
46
47
48
49
50  }
51
52
53
54
```

```java
    public void tearDown()
    {
        driver.close();
    }

    //@Test(description="Test Book5 is left of Book6 & below Book1")
    public void test1()
    {
        WebElement book5=driver.findElement(RelativeLocator.withTagName("li").toLeftOf(By.id("pid6")).below(By.id("pic
        String id=book5.getAttribute("id");
        System.out.println(id);
        Assert.assertEquals("pid5",id);
    }

    //Test Book2 is above Book6 & right of Book1"
    @Test(description="Test Book2 is above Book6 & right of Book1")
    public void test2()
    {
        WebElement book2=driver.findElement(RelativeLocator.withTagName("li").toRightOf(By.id("pid1")).above(By.id("p
        String id=book2.getAttribute("id");
        System.out.println(id);
        Assert.assertEquals("pid2",id);
    }


}
```

Selenium 4 Features

- Screenshot of Page
- Screenshot of a Section in a Page
- Screenshot of Specific Web Element

```java
    WebDriver driver;

    @Test(description="Screenshot of a complete Page")
    public void screenshotOfaPAge() throws IOException
    {
        WebDriverManager.chromedriver().setup();
        driver=new ChromeDriver();

        driver.get("https://demo.nopcommerce.com/");
        driver.manage().window().maximize();

        TakesScreenshot ts=(TakesScreenshot)driver;

        File src=ts.getScreenshotAs(OutputType.FILE);

        File trg=new File("HomePage.png");

        FileUtils.copyFile(src,trg);

        driver.close();
    }

}
```

```java
@Test(description = "Screenshot of Section of a Page")
public void ScreenShotOfaSectionInaPage() throws IOException
{
    WebDriverManager.chromedriver().setup();
    driver=new ChromeDriver();

    driver.get("https://demo.nopcommerce.com/");
    driver.manage().window().maximize();

    WebElement pageSection=driver.findElement(By.xpath("//div[@class='product-grid home-pag

    File src=pageSection.getScreenshotAs(OutputType.FILE);

    File trg=new File("FeatureProducts.png");

    FileUtils.copyFile(src,trg);

    driver.close();

}
}
```

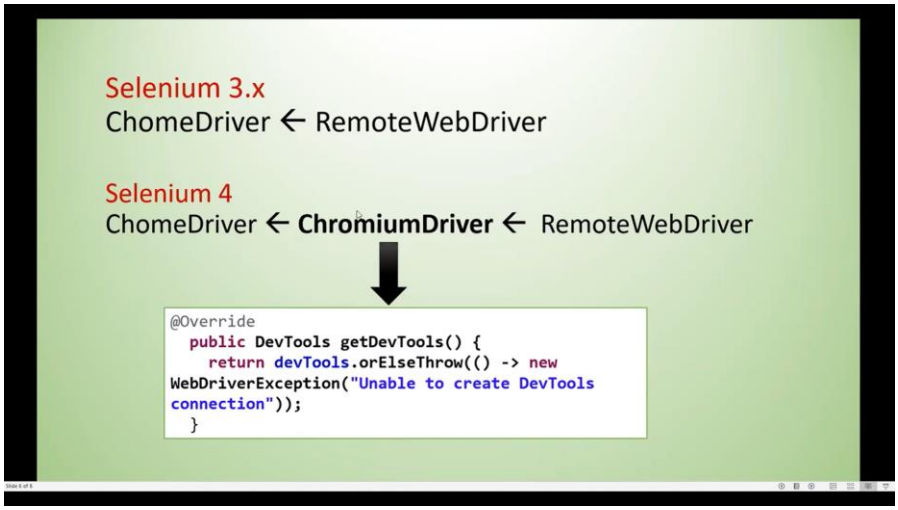
```java
@Test(description = "Screenshot of a WebElement")
public void ScreenShotOfanElement() throws IOException
{
    WebDriverManager.chromedriver().setup();
    driver=new ChromeDriver();

    driver.get("https://demo.nopcommerce.com/");
    driver.manage().window().maximize();

    WebElement logoElement=driver.findElement(By.xpath("//div[@class='header-logo']//a//img

    File src=logoElement.getScreenshotAs(OutputType.FILE);

    File trg=new File("Logo.png");

    FileUtils.copyFile(src,trg);

    driver.close();

}
}
```

Selenium 4 Features

- Chrome Dev Tools
  - Enable Network Offline
  - Enable Network Online
  - Get Console Logs
  - Load Insecure Web Site

1. Go to google settings
2. Then select the more tools then developer tools.
3. In this available some options like elements, console, preferences, memory, applications, security, **networks** …….

4. Select networks observe in this options



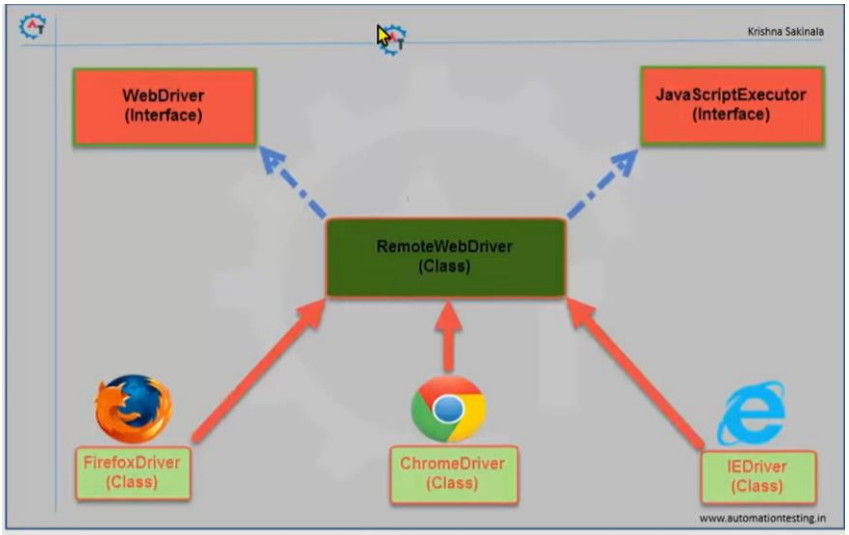

```java
import org.testng.annotations.Test;

import io.github.bonigarcia.wdm.WebDriverManager;

public class ChromeDevToolsDemo {

    WebDriver driver;

    @Test(priority=1,description = "Enable Network Offline")
    public void enableNetworkOffline()
    {
        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();

        DevTools devTools = ((ChromiumDriver) driver).getDevTools();

        devTools.createSession();

        devTools.send(Network.enable(Optional.of(1000000), Optional.empty(), Optional.empty()));
        devTools.send(emulateNetworkConditions(true, 100, 1000, 2000, Optional.of(ConnectionType.wifi)));

        devTools.addListener(loadingFailed(),loadingFailed -> assertEquals(loadingFailed.getErrorText(), "net::ERR_IN

        driver.get("https://demo.nopcommerce.com/");
    }

    }
```

```
@Test(priority=2,description = "Enable Network Online")
public void enableNetworkOnline()
{
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver();

    DevTools devTools = ((ChromiumDriver) driver).getDevTools();

    devTools.createSession();

    devTools.send(Network.enable(Optional.of(1000000), Optional.empty(), Optional.empty()));
    devTools.send(emulateNetworkConditions(false, 100, 1000, 2000, Optional.of(ConnectionType.wifi)));

    driver.get("https://demo.nopcommerce.com/");
}

}
```

```
@Test(priority=3,description="Get Console Logs")
public void consoleLogs()
{
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver();

    DevTools devTools = ((ChromiumDriver) driver).getDevTools();

    devTools.createSession();

    devTools.send(Log.enable());

    //add listener to verify the console message
    devTools.addListener(Log.entryAdded(), entry -> System.out.println(entry.asSeleniumLogEntry()));

    driver.get("https://demo.nopcommerce.com/");
}
```

```
@Test(priority=4,description = "Load Insecure Web Site")
public void loadInsecureWebsite()
{
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver();

    DevTools devTools = ((ChromiumDriver) driver).getDevTools();

    devTools.createSession();

    devTools.send(Security.setIgnoreCertificateErrors(true));

    driver.get("https://expired.badssl.com/");
}
```

WebDriver driver
=
new FirefoxDriver();



WebDriver is an **interface**. An interface contains only empty methods that have been defined but not implemented. These methods can be implemented by anyone as long as the method type and signatures are not violated. Therefore, an interface is also known as contract, because you can use an interface as you like but you cannot change the way it has been defined. And, since it has empty methods you won't actually need to instantiate it and so you cannot instantiate it.

```
interface WebDriver
{
        void get();
        void getTitle[];
}
```

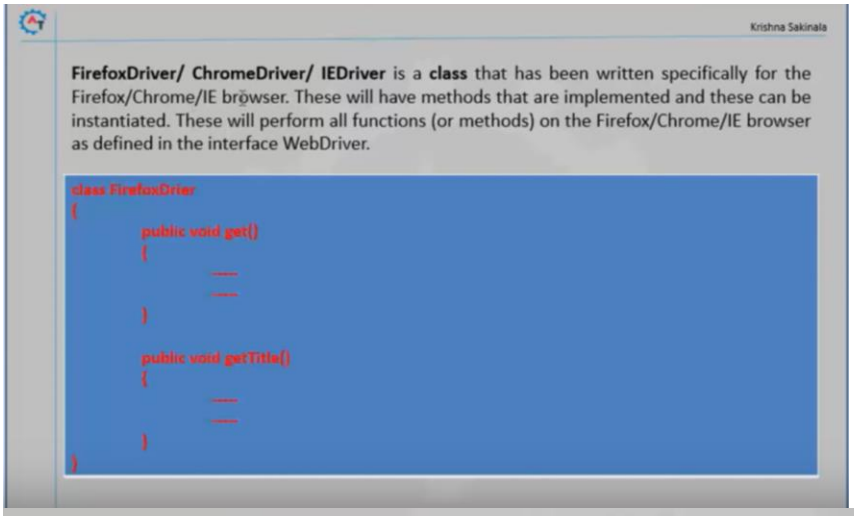
Firstly,What is WebDriver?

- WebDriver is an interface provided by Selenium WebDriver.
- Interface is a collection of abstract methods(methods without implementation)
- WebDriver interface acts as a contract that each browser specific driver implements.
- WebDriver interface declares the following basic methods:

Usually will carte an object to a class like:

```
ClassA obj = new ClassA();
```

1. Web driver is a interface in java.
2. We can’t create objects.
3. We can create methods only but not implemented. We can implements these methods using method types or signatures.
4. Every interface having empty methods.



So in the above statement, we are actually telling **FirefoxDriver/ ChromeDriver/ IEDriver** class that "you can automate the various methods that you want on the **Firefox/Chrome/IE** browser but you need to stick to the contract defined in **WebDriver**". So we declare a reference variable of type **WebDriver** and then use it to instantiate **FirefoxDriver/ ChromeDriver/ IEDriver**, which means that the object (driver) is of type **WebDriver** but points to the memory allocation to all data and methods in **FirefoxDriver/ ChromeDriver/ IEDriver** (and, as mentioned above, the **FirefoxDriver/ChromeDriver/ IEDriver** class already has the implemented version of methods in WebDriver).

ParentTest.java:

FirefoxDriver driver;

ChildTest.java extends ParentTest

1. driver = new FirefoxDriver() → Class Names are Same ✓
2. driver = new ChromeDriver() → Classes are different → ✗
3. ChromeDriver driver = new ChromeDriver() → can not create same object name for different classes. ✗
4. ChromeDriver driver1 = new ChromeDriver() → Same code will not execute for different browsers as the object is different. ✗

So, we will use

WebDriver driver = new FirefoxDriver(); ✓

Imp points:

FriefoxDriver driver;

Here Firefox Driver is a class.

Driver is a class object.

Class name is same accepted.

Cant accepted the same class objective names from different classes.

One class name extending to another class.

WebDriver driver1 = new Firefox Driver ();

Here

webDriver is an interface(parent). Which is available in selenium?

Driver1 is a reference variable of objective.
(here **New Firefox driver** (child class) is object).
New is key word.
Friefoxdriver is class. Which implements web driver interface.
Child class object can be reffered by parent interface reff variable. Here driver1 is reff variable.
Implements the all menthods in webdriver and friefoxdriver. This will refer to firefoxobect.you can call methods in firefoxdriver which are executed in Firefox driver1.

AGENDA

- Concept of JDK.
- Concept of JRE.
- Concept of JVM.
- Relationship among JDK, JRE and JVM.
- Architecture of JVM.

CONCEPT OF JDK

Writing Java Code has two aspects

- Development
- Execution

CONCEPT OF JDK

- It is called Java Development Kit.
- Provides environment for developers to develop Java programs as well as execute and run our program using JRE & JVM.
- It includes the JRE, an **interpreter/loader**(Java), a **compiler**(javac), an **archiver**(jar), a **documentation generator**(javadoc) and other tools needed in Java development.

CONCEPT OF JRE

- It is called Java Runtime Environment.
- Provides the minimum requirements for executing a Java program.
- JRE consists of Java Virtual Machine(JVM), core classes, and supporting files.

CONCEPT OF JVM

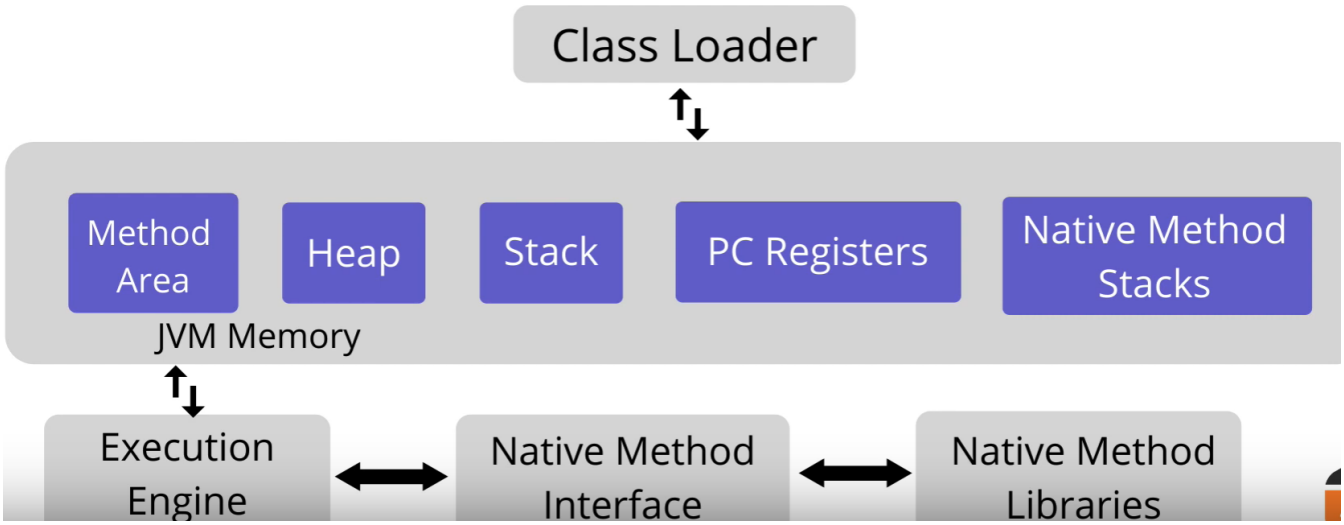
- It is called Java Virtual Machine.
- It is the most crucial part of both JDK and JRE as it executes the code and is responsible for allocating memory space.
- Whatever Java program you run, goes into JVM, and JVM is responsible for executing the code line by line , hence it is also known as **INTERPRETER**.

RELATION BETWEEN JDK, JRE & JVM

- **JDK** = Java Runtime Environment (JRE) + Development Tools

- **JRE** = Java Virtual Machine (JVM) + Library Classes

ARCHITECTURE OF JVM



CLASS LOADER

- Loading the .class files
- It performs loading, l initialisation.

METHOD AREA

- Stores all class level info like class name, immediate parent class name, methods, variable info, constant runtime pool etc.

HEAP

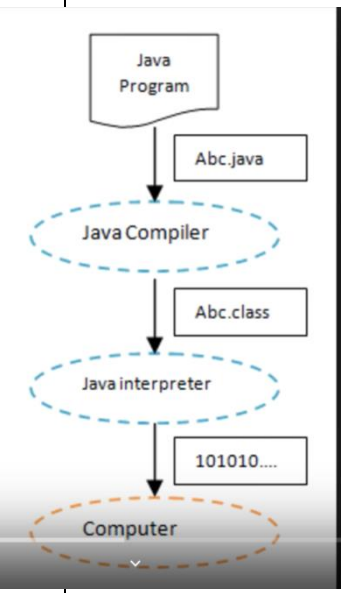
- Runtime data area tha about objects, their r variables, arrays etc.
- It is the memory space of program.

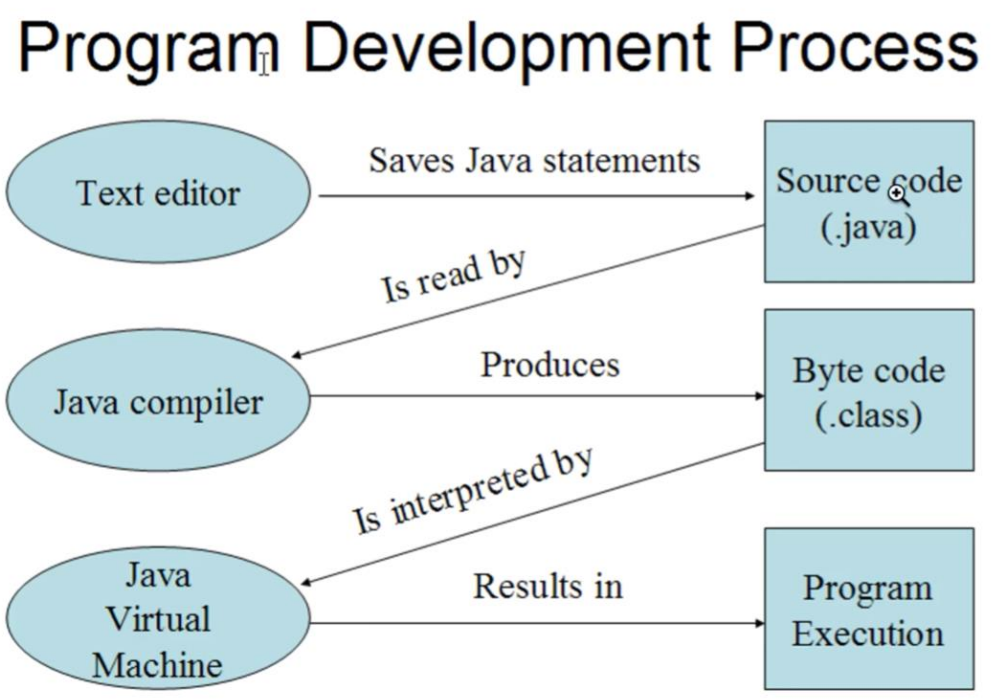
STACK

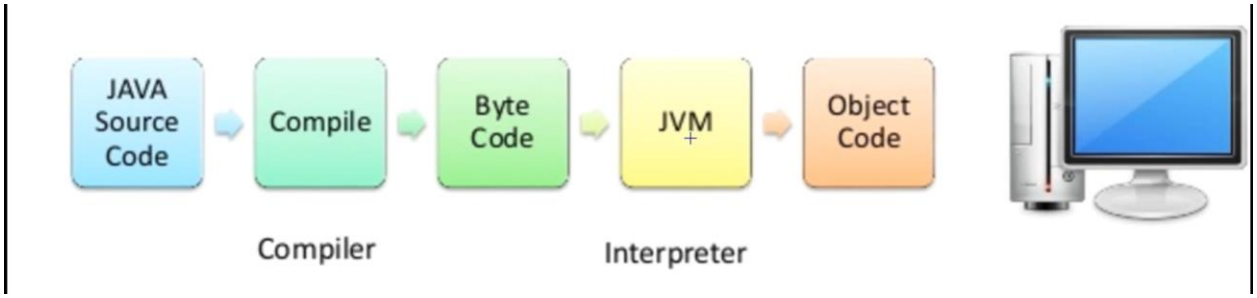
- Used to store local variables, partial results and data for method invocation and returns.

EXECUTION ENGINE

- Reads the .class(bytecode) file line by line, use data and info available in memory areas and execute the instructions.
- It contains Interpreter, JIT and Garbage Collector.







TakesScreenshot

Taking Screenshot in Selenium is a 3 Step process

1. Convert web driver object to TakeScreenshot
TakesScreenshot scrShot =((TakesScreenshot)webdriver);

2. Call getScreenshotAs method to create image file
File SrcFile=scrShot.getScreenshotAs(OutputType.FILE);

3. Copy file to Desired Location
save it as Test.jpeg

Locaters in selenium

What is locaters in selenium?

Locators are the way to identify an *HTML* element on a web page.

How to locate a web element in DOM?

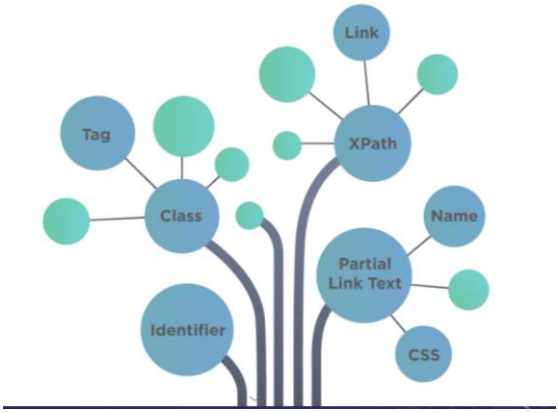
1.DOM can be accessed in Google Chrome either by pressing ***F12*** or by ***right click*** on the web page and then by selecting ***Inspect*** *(as shown in the screenshot below)*.

2. Once we click on the "*Inspect option,"* it will open the ***Developer Tools console,*** as shown below. By default, it will open the "*Elements*" tab, which represents the complete ***DOM*** structure of the web page. Now, if we hover the mouse pointer over the ***HTML*** tags in the DOM, it will highlight the corresponding elements it represents on the webpage.

3. Now, the main point is, how do we find the web element in the ***DOM***. Click on the "*Mouse Icon*" arrow (*as designated by Marke 1 in below screenshot*) and then select the web element on the web page. It will automatically highlight the corresponding ***HTML*** element in the *DOM.* Suppose we want to find the *HTML* elements corresponding to the banner image (*as shown below by marker 3*). When we select the mouse point and click on the banner image, it will automatically highlight the corresponding *HTML* element, as shown by ***marker 2***, in the below screenshot:

What locators are supported by Selenium?

There are various types of locators, using which we can identify a web element uniquely on the Webpage. The following figure shows a good depiction of several types of locators that Selenium supports.



To access all these locators, Selenium provides the "***By***" class, which helps in locating elements within the *DOM*. It offers several different methods (*some of which are in the image below*) like ***className, cssSelector, id, linkText, name, partialLinkText, taName,*** *and* ***xpath, etc.,*** which can identify the web elements based on their corresponding locator strategies. As we can see, Selenium supports the following locators:

- **ClassName** – A ClassName operator uses a class attribute to identify an object.
- **cssSelector** – CSS is used to create style rules for webpages and can be used to identify any web element.
- **Id** – Similar to class, we can also identify elements by using the 'id' attribute.
- **linkText** – Text used in hyperlinks can also locate element
- **name** – Name attribute can also identify an element
- **partialLinkText** – Part of the text in the link can also identify an element
- **tagName** – We can also use a tag to locate elements
- **xpath** – Xpath is the language used to query the XML document. The same can uniquely identify the web element on any page.

How to locate a web element by using the "id" attribute?

ID" as a locator is one of the most common ways of identifying elements on a web page. According to ***W3C***, an ID for a web element always needs to be unique. The ***ID*** is one of the fastest and unique ways of locating web elements and is considered as one of the most reliable methods for determining an element. But with the advancement in technologies and more of the ***dynamic web pages***, "*IDs*" are generated dynamically and generally not the reliable way to locate a web element, as they change for different users.

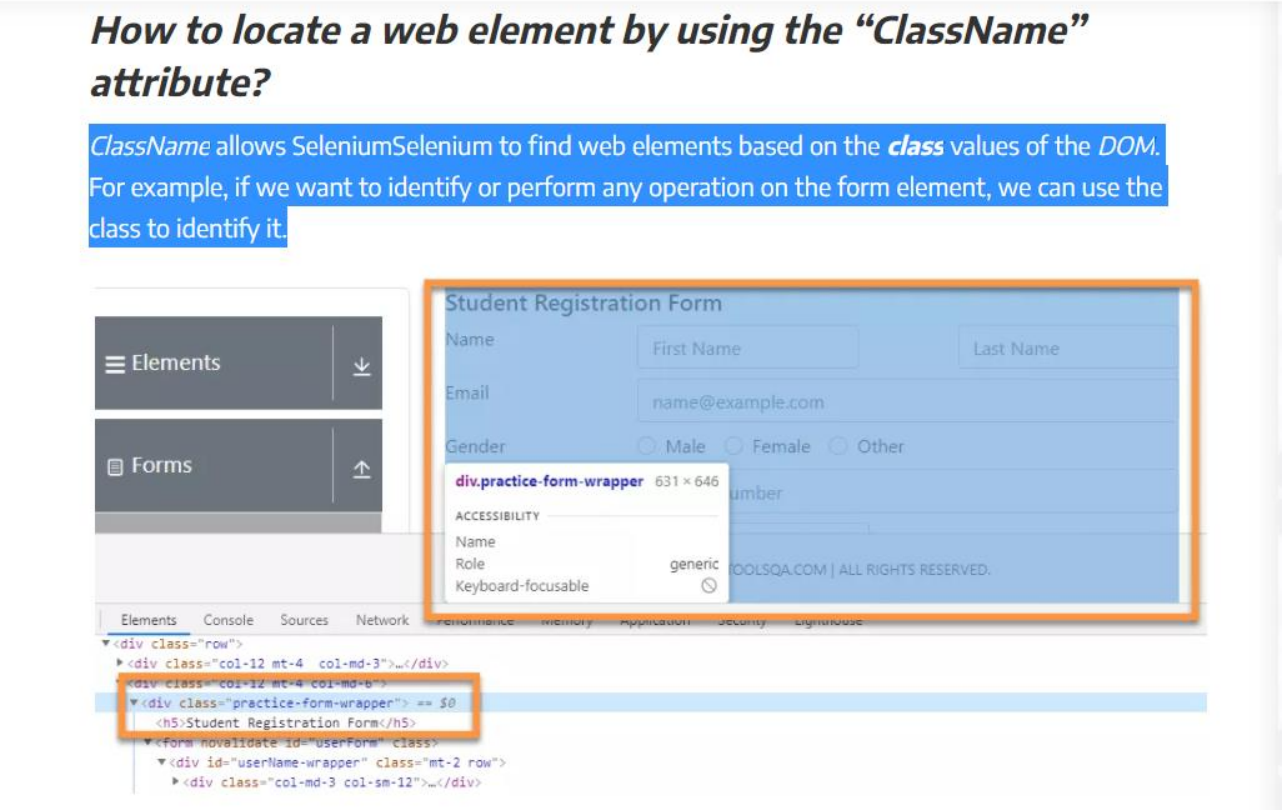
How to locate a web element by using the "name" attribute?

a web page can have multiple elements with the same "*name*" attribute. So, if we are going to use the name attribute for the identification of a web element, we should always make sure that the name attribute must contain a unique value. Otherwise, it may identify several different elements on the same page, which may have the same name value. If multiple elements have the same value of the '*name*' attribute, then, Selenium will just select the first values in the page which matches the search criteria. So, we always recommend making sure that the value of the ***name*** attribute should be unique when we select it for locating a web element.

How to locate a web element by using the "ClassName" attribute?

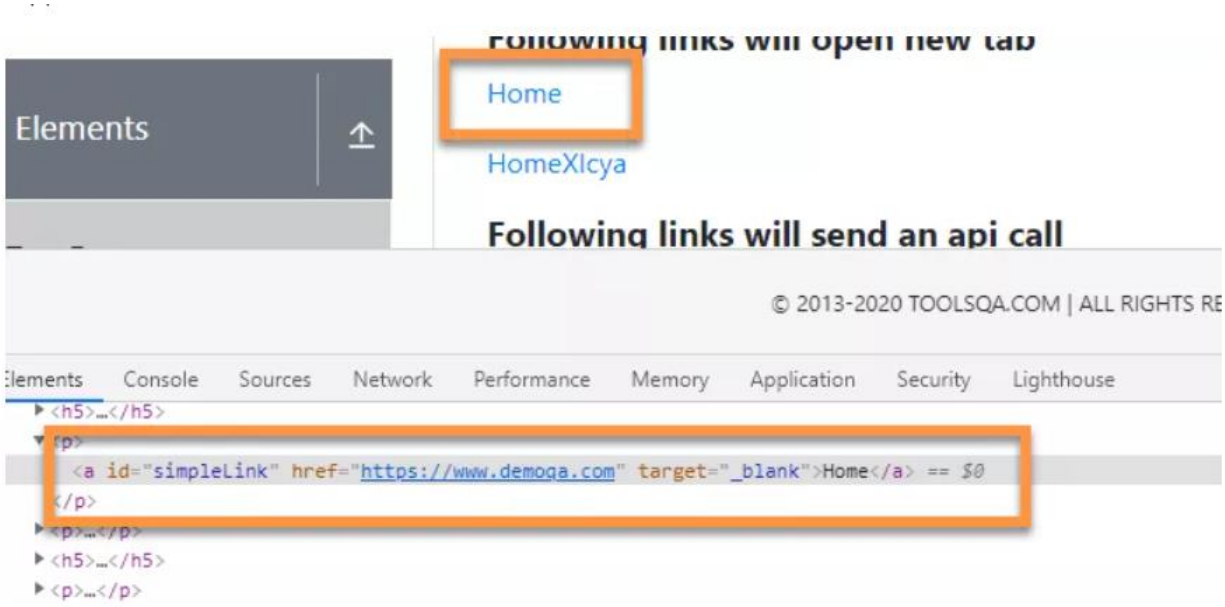
ClassName allows SeleniumSelenium to find web elements based on the ***class*** values of the *DOM*.

For example, if we want to identify or perform any operation on the form element, we can use the class to identify it.



How to locate a web element by using the "LinkText" and partialLinkText" attribute?

LinkText and **partialLinkText** both are quite similar in functionality and locate web elements by using the hyperlink texts. We can only use them for elements containing **anchor**(<a>) **tags**. Similar to other locator strategies, if multiple hyperlinks with the same texts are present on the page, then Selenium will always choose the first one.
Suppose we need to click on the link as shown below for the "Home" link:



As we can see, the anchor element has the following properties and attributes:

```
<a id="simpleLink" href="https://www.demoqa.com" target="_blank">Home</a>
```

For identifying elements using ***link text or partial link text,*** we need to use the hyperlink text, as shown below:

```
By.linkText("Home");
```

Similarly, partial link text can also recognize the element by using ***part of the hyperlink text,*** as shown below:

```
By.partialLinkText("Ho");
```

It will identify any link that contains '*Ho*' in the hyperlink text. Likewise, if several links are containing "*Ho,*" then Selenium will select the first one.

How to locate a web element by using the "TagName"?

This locator type uses ***tag names*** to identify elements in a web page. The ***tag name*** is the *HTML* tag, such as ***input, div, anchor tag, button, etc***.

The following syntax finds the web elements with a ***tagName:***

```
Selenium Find By Tag Name
By.tagName("a");
```

The ***tagName*** locator returns all the elements from the page that contains a specified tag.

How to locate a web element by using the "CSSselector"?

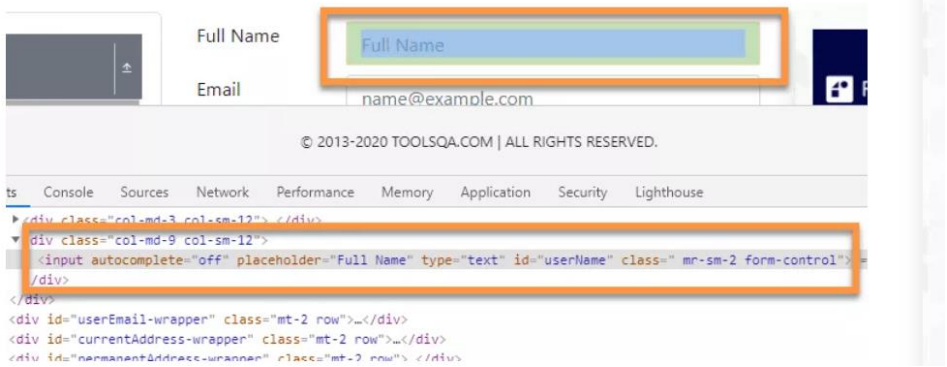
CSS or Cascading style sheets are used extensively to style webpages and hence can be an effective medium to locate various web elements. These days most of the web pages are dynamically designed. Thus its quite challenging to get a unique id, name, or class to locate element. In this type of situation, ***CSS selectors*** can be a great alternative as these are way faster compared to another locator strategies.

The basic syntax of identifying a web element using ***CSS*** is as follows:

The basic syntax of identifying a web element using ***CSS*** is as follows:

```
1 css=(HTML Page)[Attribute=Value]
```

For example, let's say we want to find the following input textbox using CSS selector.



As we can see, the input element has the following properties and attributes:

```
1 <input autocomplete="off" placeholder="Full Name" type="text" id="userName" class=" mr-sm-2 form
```

Now, to find the element using the CSS selector, we have to use the following syntax:

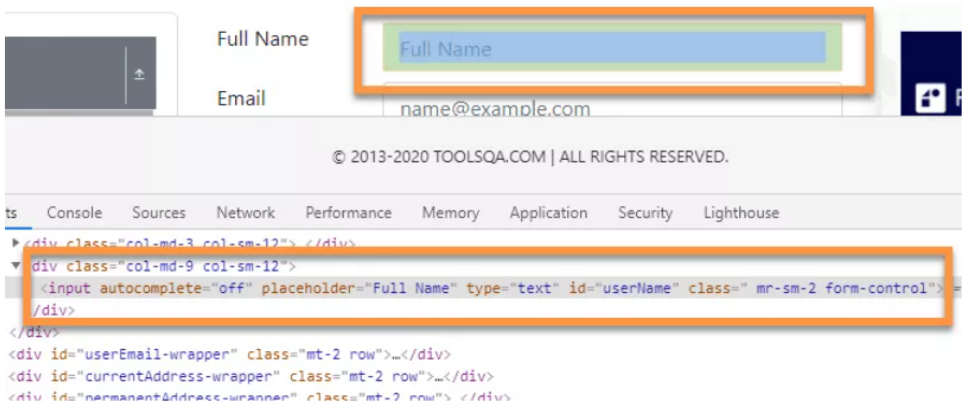
```
Selenium Find By CSS Selector
1 By.cssSelector("input[id= 'userName']");
```

Similarly, we can use other attributes along with the tags to define different CSS locators. We will discuss more details about the CSS selector in another tutorial.

How to locate a web element by using the "xpath" attribute?

XPath uses the ***XML*** expression to locate an element on the webpage. Similar to CSS selectors, Xpath is quite useful in locating dynamic elements on a webpage. Xpath can access any element present in the webpage even when they have dynamic properties.
The basic syntax of identifying a web element using the XPath locator strategy is as follows:

For example, let's say we want to find the following input textbox using ***XPath***:



The XPath for the above example will be:

```
1 //input[@id='userName']
```

So, sample code to find the element using XPath will be the following:

So, sample code to find the element using XPath will be the following:

```
Selenium Find By XPath
1 By.xpath("//input[@id='userName']");
```

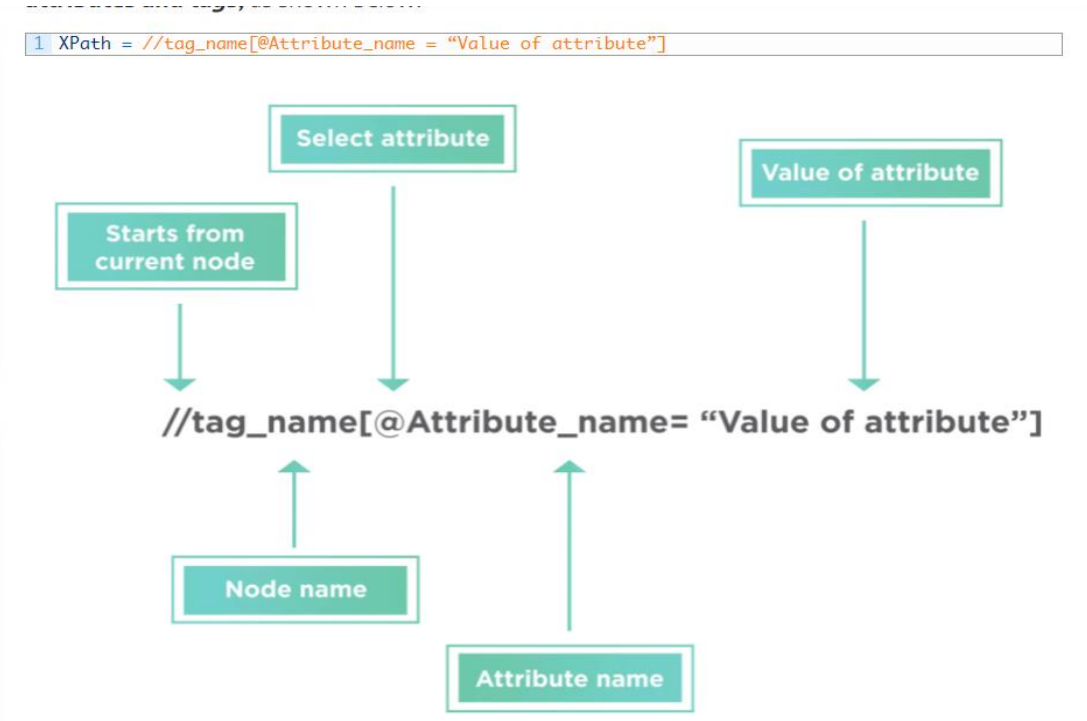
Here we have used the ***input tag and id attribute*** to identify the element. We will cover the more in-depth details of the "*XPath locator*" strategy in another article.

Let's now see the usage of these locators' strategies to identify various web elements.

XPath in Selenium

XML stands for extensible markup language. A markup language is a set of codes, or tags, that describes the text in a digital document. The most famous markup language is hypertext markup language (HTML), which is used to format Web pages.
XML Path or commonly known as **XPath,** is a query language for **XML** documents. It is made of *path expression* governed by some pre-defined conditions. Moreover, ***XPath*** **can be effectively used to identify and locate almost any element present on a web page**. It is quite helpful while working on a dynamic web page where the web element's unique attributes are not constant. Moreover, it allows us to write **XML document navigation flow** for locating web elements. Consequently, let's first understand what syntax does *XPath* uses to locate the web elements:
Syntax of XPath: *XPath* finds any element within a *webpage* by using *DOM*. So, its syntax is also made up of ***DOM attributes and tags,*** as shown below:

```
1 XPath = //tag_name[@Attribute_name = "Value of attribute"]
```



- The XPath syntax generally starts with "//" double slash. That is to say, it will begin with the current node defined by the tag name.

- The next part is **tag_name; it** denotes the HTML tag name of the node.
- Subsequently, anything present inside the node encloses in the square brackets.
- Additionally, the "**@**" sign selects the attribute.
- Moreover, the "**Attribute_name**" is the name of the attribute of the node.
- And, the "**Value of attribute**" denotes the attribute value from the node.

What is HTML/XML DOM?

All the ***HTML*** web pages are internally represented as an ***XML*** document. Additionally, the XML document has a *tree-like structure*, where we have different tags and attributes. For example, if we open the *Chrome Dev tools* section by *right-clicking* on the page "***https://demoqa.com/***" and selecting the "***Inspect***" option, the *HTML* structure will look as follows:

t starts with a tag ***<html>*** which has two child nodes ***<head>*** and ***<body>***. Here ***<body>*** again has a child node like *** ,*** which has its child nodes. Further, you can also see the first *** *** tag has an "***id***" attribute, which has a value "***app".*** Similarly, the child node of the *** *** also has its own set of attributes and values.

One thing that we can notice here is every node that opens is again closed by using *forward slash* before the tag name, such as the ***<body>*** tag is closed using ***</body>***. So, this way, we can see that all the *HTML* pages are internally represented as an *XML* document and constitute the *XML DOM (Document Object Model).*

Syntax Element	***Details***	***Example***	***Example Details***
Single slash "***/***"	It selects the node from the root. In other words, if you want to select the first available node, this expression is used.	/html	It will look for the *HTML* element at the start of the document.
Double slash "***//***"	It selects any element in the *DOM* that matches the selection. Additionally, it doesn't have to be the exact next node and can be present anywhere in the *DOM*.	//input	It will select the input node present anywhere in the *DOM*.

Address sign "@"	It selects a particular ***attribute*** of the node	//@text	It will select all the elements which have text attribute.
Dot "***.***"	It selects the ***current*** node.	//h3/.	It will give the currently selected node, i.e., ***h3.***
Double dot "***..***"	It selects the ***parent*** of the current node.	//div/input/..	It will select the parent of the current node. The current node is input so that it will select the parent, i.e., "***div".***
Asterisk "*******"	It selects ***any element*** present in the node	//div/*	This matches with any of the child nodes of the "***div".***
Address and Asterisk "@*******"	It selects ***any attribute*** of the given node.	//div[@*]	It matches any of the ***div*** nodes that contain at least one attribute of any type.
Pipe "|"	This expression is used to select a ***different*** path.	//div/h5|//div/form	We have selected two different paths i.e. "***//div/h5***" and "***//div/form***" using "| ". Moreover, here we have selected two different paths originating from the same parent node "***div*** ". Using this *XPath,* we can select both expressions.

Apart from the syntax as mentioned above components, *XPath in Selenium* provides a few advanced concepts. Moreover, we can find a web element at a specified position if the locator's *XPath* has resulted in multiple elements. These are ***Predicates***. Subsequently, let's understand *how we can use Predicates to locate web elements?*

What are the different types of XPaths in Selenium?

Selenium uses two strategies to locate elements using *XPaths* :

- *Locating a web element using an **Absolute XPath.***
- *Locating a web element using **Relative XPath**.*

What is Absolute XPath in Selenium?

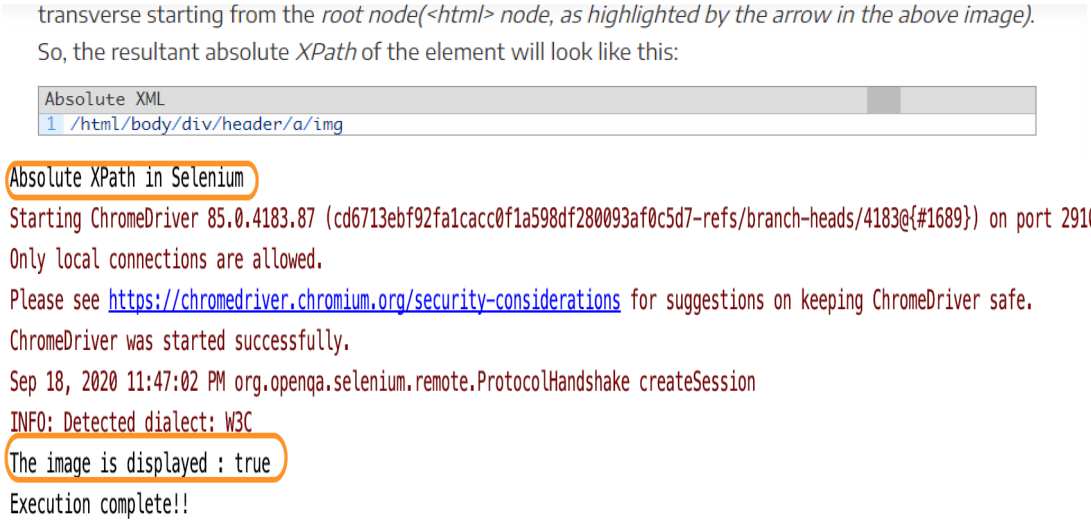
Absolute XPath is the **direct way of finding the element**.

An absolute xpath starts with the / symbol.

Absolute XPath select the element from the root node.

Moreover, it starts from the *first/root* node of the *XML/HTML* document and goes all the way to the required node following one node at a time. For better understanding, let's take an example, showing the *DOM* of page "***https://demoqa.com/***":

In the above document, if we want to find the ***absolute path of <img> node,*** then we will have to transverse starting from the *root node(<html> node, as highlighted by the arrow in the above image)*. So, the resultant absolute *XPath* of the element will look like this:



One major disadvantage or limitation of absolute XPath is that **if there is any change in any of the elements on the webpage, then the XPath for any subsequent element will change.** Hence, resulting in the failure of test scripts trying to locate the web element. So, generally, it is not recommended to use the absolute XPath for locating the Web Elements.

What is Relative XPath in Selenium?

Relative XPath starts from *any node* inside the *HTML DOM or* **Relative Xpath** starts from the middle of HTML DOM structure.

It can search elements anywhere on the webpage, means no need to write a long xpath and you can start from the middle of HTML DOM structure.

It beings with a ***double forward slash.***

For better understanding, let’s take the same example as the absolute path above.

In the previous example, we followed the whole node hierarchy starting from "***<html>***" till the "***<img?*** " tag. It allowed us to trace the whole path and trace the element's exact position on the page. But imagine after some time if one more node changes, this previous absolute *XPath* will become redundant, so this is where *relative XPath* comes handy.

The relative *XPath* uses the relative position of a particular node on the page. For example, if we want to write a relative *XPath* of the same image element that we defined in the absolute *XPath* section, we don't need to write all the nodes. Similarly, if we want to write the *XPath* for the <*img*> element, we don't need to start from the ***<html> tag***. We can begin to directly from the ***<img>*** tag, as shown below:

```
//img[@src = "/images/Toolsqa.jpg"]
```

As we can see, relative *XPath* can start from any node. It also overcomes the fragile nature of *Absolute XPath*. Even if any preceding element changes or removes, it will not affect the ***Relative XPath.***

Let's modify the above code snippet to locate the image using the *relative XPath in Selenium:*

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class XPathDemo {
    public static void main(String[] args) throws InterruptedException{

        System.out.println("Relative XPath in Selenium");
        WebDriver driver = new ChromeDriver();
        driver.get("https://demoqa.com");

        //Locate the web element using relative XPath
        WebElement headerImage = driver.findElement(By.xpath("//img[@src = '/images/Toolsqa.jpg
        // Validate that the header image is displayed on the web page
        System.out.println("The image is displayed : " + headerImage.isDisplayed());

        driver.close();
        System.out.println("Execution complete!!");

    }
}
```

When you execute the above code snippet, it will show the output as follows:

```
Relative XPath in Selenium
Starting ChromeDriver 85.0.4183.87 (cd6713ebf92fa1cacc0f1a598df280093af0c5d7-refs/branch-heads/4183@{#1689}) on port 33660
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Sep 18, 2020 11:55:57 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
The image is displayed : true
Execution complete!!
```

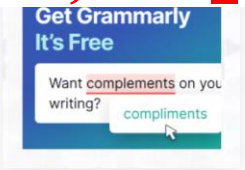
XPath in Selenium provides various *xpath functions and axes*(*relationships*), which helps tow*rite effective*

XPaths of the web elements and defining a unique locator for the web elements. In this article, we will cover details of all those functions and axes, using which we can choose the *XPath* of a web element very effectively and uniquely, by covering the details under the following topics:

What are XPath Functions in Selenium?

1.Contains() function: We can use it if "part of the value of any attribute ***changes dynamically"***. The syntax for using the XPath *contains()* method is:

//tag_name[contains(@attribute,'value_of_attribute')]



Ex2: <input autocomplete="off" placeholder="name@example.com" type="email" id="userEmail" class="mr-sm-2 form-control">

Syntax ://input[contains(@placeholder, "example")]

Ex3: <input autocomplete="off" placeholder="name@example.com" type="email" id="userEmail" class="mr-sm-2 form-control">

//input[contains(@placeholder, "example")]

2. XPath Starts-with(): The **starts-with()** function is used to **finds the element which has *attribute value starting with the specific given character or character sequence*.** This function is quite useful while dealing with dynamic web pages. Imagine an element that has an **attribute value that keeps on changing with every page load or page operation**. Usually, these dynamic elements

have few common starting characters, followed by random dynamic texts. Apart from the dynamic attribute, this can also identify static elements.

Syntax://tag_name[starts-with(@attribute,'Part_of_Attribute_value')]

<input autocomplete="off" placeholder="Full Name" type="text" id="userName" class=" mr-sm-2 form-control">

1//input[starts-with(@placeholder,"Fu")]

3. Text() function :This function uses the text of the web element for locating the element on the *webpage*. This function is quite useful if **your element contains the text**, for example, *labels*, which always contain a static text.where the text() method returns the text of the web element identified by the *tag_name*, and it compares the value with the string provided on the right side**.**

Syntax: //tag_name[text()='Text of the element']

Ex: <label class="form-label" id="userEmail-label">Email</label>

//label[text()="Email"]

4. AND & OR operators : The "and " operator is used to combining two different conditions or attributes to identify any element from a webpage using XPath efficiently. *For example, if we have two attributes, a and b, we can combine both to uniquely identify an element on the webpage using the "and " operator.*

The syntax for using the "***and*** " operator is:

Syntax: //tag_name[@name = 'Name value' and @id = 'ID value']

Ex: //input[@placeholder ='Full Name' and @type = 'text']

The "***or*** " operator is used to locate an element based **on any of the conditions** segregated by the "***or*** " operator. We use it majorly based on certain run-time conditions. The attributes of the element can contain any of the values, so putting a condition with "***or*** " will ensure that the element is locatable with any one of those conditions.

The syntax for using the "***or*** " operator is:

Syntax: 1 //tag_name[@name = 'Name value' or @id = 'ID value']

//input[@placeholder ='Full Name' or @type = 'text']

What are the XPath Axes in Selenium?

XPath axes are methods to identify those dynamic elements which are not possible to find by normal XPath method such as ID, Classname, Name, etc. Axes are so named because they tell about the axis on which elements are lying relative to an element.

Dynamic web elements are those elements on the webpage whose attributes dynamically change on refresh or any other operations.

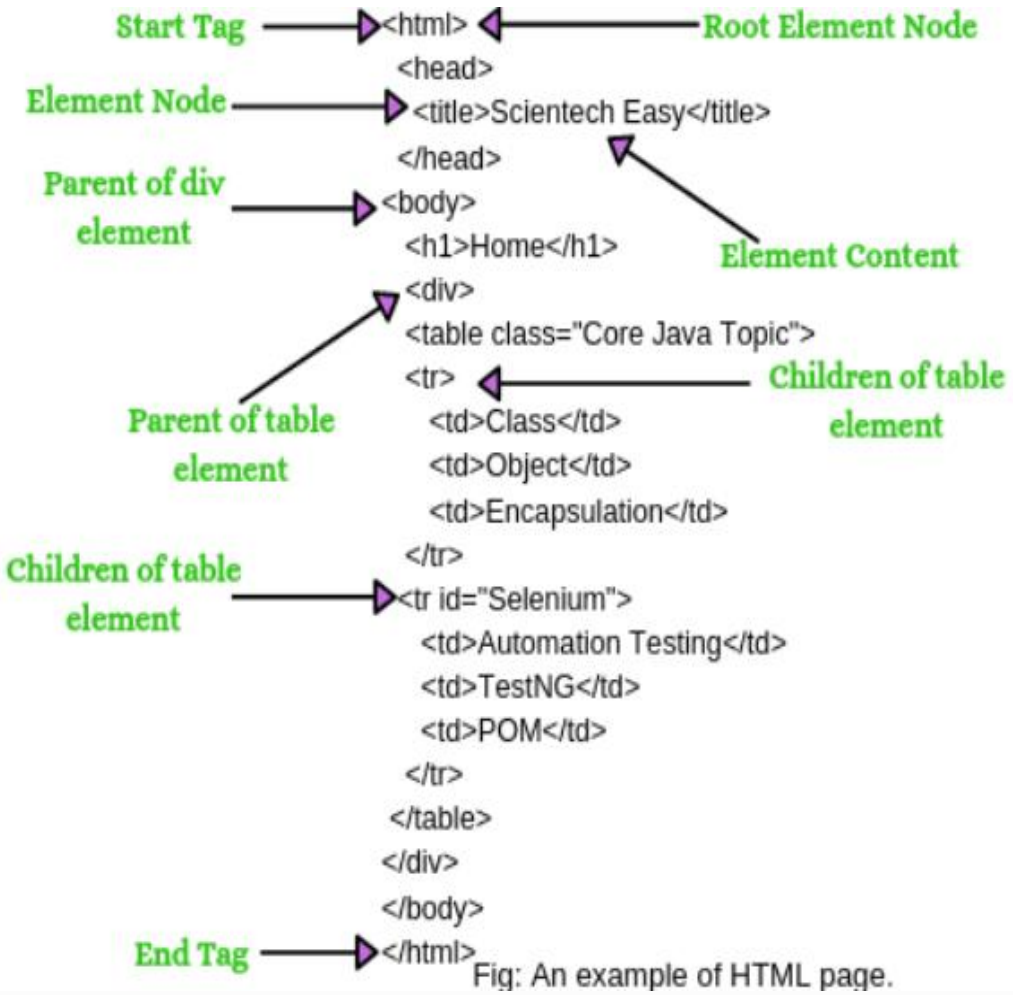


Fig: An example of HTML page.

1. **Nodes:** DOM represents as trees of nodes. The topmost element of the tree is called root node. The example of nodes in the XML document above:
"html" is the root of element node, "title" is a element node. XPath contains seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document nodes.

2. **Atomic values:** The node which has no parents or children is called atomic values. For example, Automation Testing, TestNG, POM.
3. **Parents:** Each element and attribute has one parent like father or mother. For example, "body" is the parent of div element. "div" is the parent of the table element.

4. **Children:** Element nodes that may contain zero, one or more children. For example, tr is children of table element, div is the children of body element, table is children of div element.

5. **Siblings:** The node that has the same parent is called siblings. In the above XML document, title and body elements both are siblings.

XPath Axes Methods:

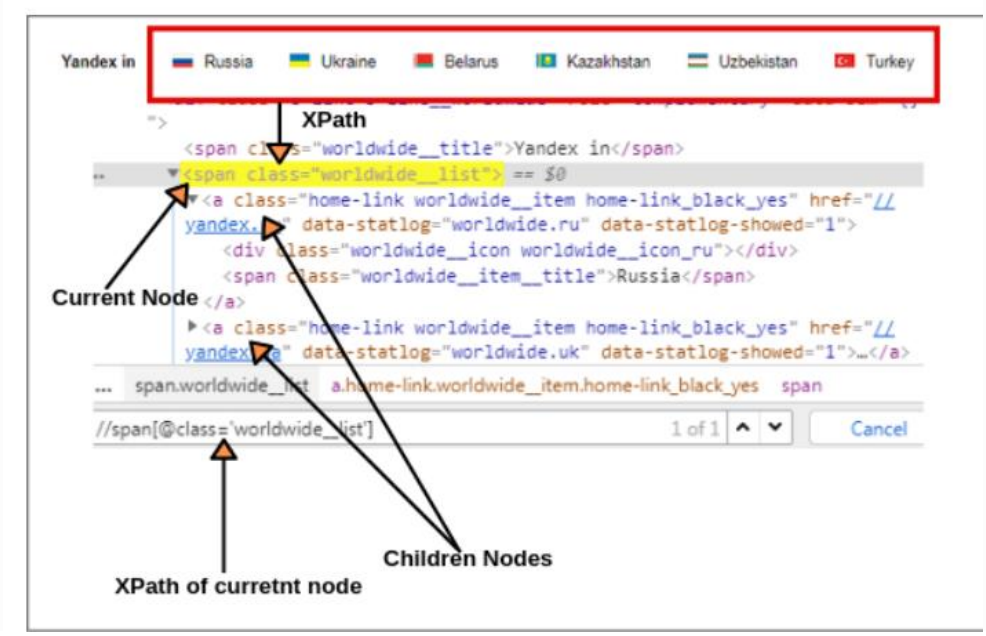
To navigate the hierarchical tree of nodes in an XML document, XPath uses the concept of axes. The XPath specification defines a total of 13 different axes that we will learn in this section.

Each axis contains various nodes that depend on the current node. An XPath axis is a collection of nodes that satisfy the current navigation criteria.

1.Child Axis: The child axis is one of the 13 XPath axes that contains all the child nodes of the current context. It selects all children elements of the current node. The syntax of child axis is given below:

Syntax: //child::tagName

For example, open webpage https://www.yandex.com, right-click on Yandex.in and go to inspect option as shown in below screenshot.



Let's select all children elements of the current node as shown in the above screen. First, we will find XPath of the current node.

XPath of current node: //span[@class = 'worldwide__list']

Now we will find out XPath of children elements of current node using child axis as shown in above figure.

XPath of all children elements: //span[@class = 'worldwide__list']//child::a (1 of 6 matched).

This expression identified six children nodes using the child axis. We can get the XPath of different children elements according to the requirement by putting [1],[2]……and so on.

XPath(Russia): //span[@class = 'worldwide__list']//child::a[1] (1 of 1 matched).

XPath(Ukraine): //span[@class = 'worldwide__list']//child::a[2] (1 of 1 matched).

XPath(Belarus): //span[@class = 'worldwide__list']//child::a[3] and so on.

2.Parent Axis: The parent axis selects the parent of the current node. The parent node may be either root node or element node. The root node has no parent. Therefore, when the current node is root node, the parent axis is empty. For all other element nodes, the parent axis contains a maximum of one node.

The syntax of parents axis is given below:

Syntax: //parent::tagName

Let’s take an example. Open website yandex.com and right-click on the search box. Now we will find the parent of the current node. Let’s choose a search input box as a current node. So, we will first find XPath of current node (Search box).

XPath(Current node): //input[@id = ‘text’] (1 of 1 matched).

Now we will find the XPath of the parent element node of current node using parent syntax as shown in below screenshot.

XPath(Parent node): //input[@id = ‘text’]//parent::span (1 of 1 matched). It will select the parent node of the input tag of Id = ‘text’.

XPath(Parent node): //input[@id = ‘text’]//parent::* (1 of 1 matched).

Self Axis: Self axis selects element of the current node. It always finds only one node that represents self-element.

Syntax://self::tagName

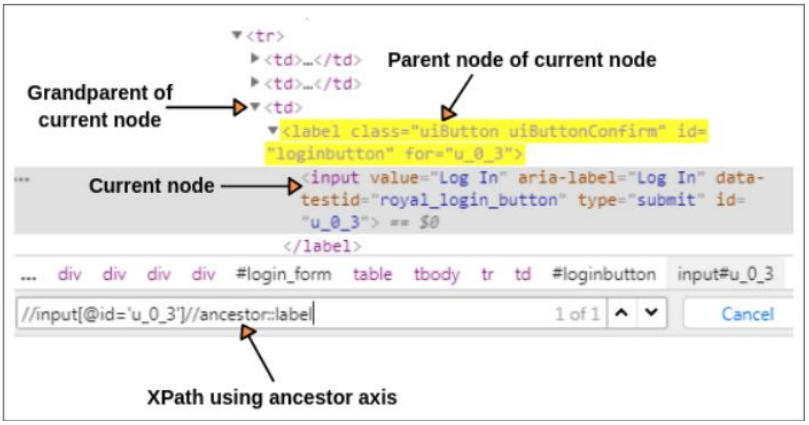
For example, we can find XPath of the current element node using self axis in the above screenshot.

XPath(Current node): //input[@id = ‘text’]//self::input (1 of 1 matched).

3.Ancestor Axis: The ancestor axis selects all ancestors element (parent, grandparent, great-grandparents, etc.) of the current node. This axis always contains the root node (unless the current node is the root node).

Syntax: //ancestor::tagName

Let’s take an example to understand the concept of ancestor axis. Open www.facebook.com, right-click on the login button and go to inspect option. Now you will see HTML code of login button as shown in below screenshot.



Let us consider the login button as current node. First, find the XPath of current node.

XPath(Current node): //input[@id = 'u_0_a']

Now, we will find XPath of parent and grandparent of current node.

XPath(Parent node): //input[@id = 'u_0_a']//ancestor::label

XPath(Grandparent node): //input[@id = 'u_0_a']//ancestor::td

In both cases, 1 of 1 node is matched by using "ancestor" axis.

4.Ancestor-or-self Axis: The descendant axis selects all descendants (children, grandchildren, etc) of the current node. Let us find XPath of current node (login button) by using the ancestor-or-self axis.

XPath(login button): //input[@id = 'u_0_3']//ancestor-or-self::input

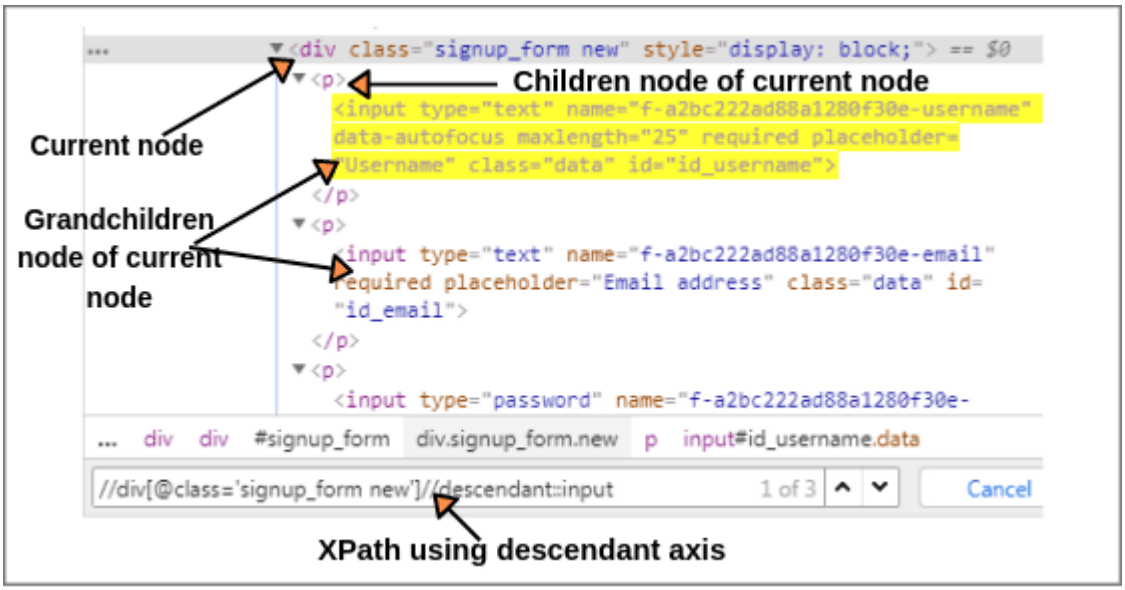
The above expression identified the current element node.

5. Descendant Axis: The descendant axis selects all descendants (children, grandchildren, etc) of the current node. Let's take an example to understand the concept of descendant axis.

Let's take an example to understand the concept of descendant axis.

Open webpage https://pixabay.com/accounts/register/?source=signup_button_header, right-click on Username element and go to inspect option. As shown in the below screenshot, let's suppose "signup_form new" as a current node. You can bring the cursor to this node to see

current node.



The XPath of current node will be as follow:

XPath(Current node): //div[@class = 'signup_form new']

Now using the descendant axis with above XPath, we can find easily all children, grandchildren elements, etc of current node.

XPath: //div[@class = 'signup_form new']//descendant::input (1 of 3)

The above XPath expression identified three elements like username, password, and email address. So, we can write XPath by putting 1, 2, and 3 in the above expression.

XPath(Username): //div[@class='signup_form new']//descendant::input[1] (1 of 1 matched).

XPath(Email address): //div[@class = 'signup_form new']//descendant::input[2]

XPath(Password): //div[@class = 'signup_form new']//descendant::input[3]

6.Descendant-or-self Axis: The descendant-or-self axis selects all descendants (children, grandchildren, etc) of the current node and current node itself. In the above screenshot, div is the current node. We can select this current node using the descendant-or-self axis.

The XPath of current node is

XPath(Current node): //div[@class = 'signup_form new']//descendant-or-self::div

The above expression identified 1 node. If we change tagname in the above XPath expression from div to input then we can get node of username, email address, and password.

Let's find XPath of these nodes.

XPath(Username): //div[@class = 'signup_form new']//descendant-or-self::input[1]

XPath(Email address): //div[@class = 'signup_form new']//descendant-or-self::input[2].

7.Following Axis:

The following axis selects all elements (nodes) in the document after closing tag of the current node. Let's consider "First name" input box as current node in the facebook webpage. The XPath of the current node is

XPath(Current node): //input[@id = 'u_0_r']

Now we will find all elements like Surname, Mobile number, etc by using the following axis of the current node. The below syntax will select the immediate node following the current node.

XPath: //input[@id = 'u_0_r']//following::input (1 of 23)

The above expression has identified 23 nodes matching by using "following" axis-surname, mobile number, new password, etc. If you want to focus on any particular element then you can change the XPath according to the requirement by putting [1],[2]……and so on like this.

XPath(Surname): //input[@id = 'u_0_r']//following::input[1] (1 of 1 matched)

With putting "1" as input, the above expression finds the particular node that is 'Surname' input box element.

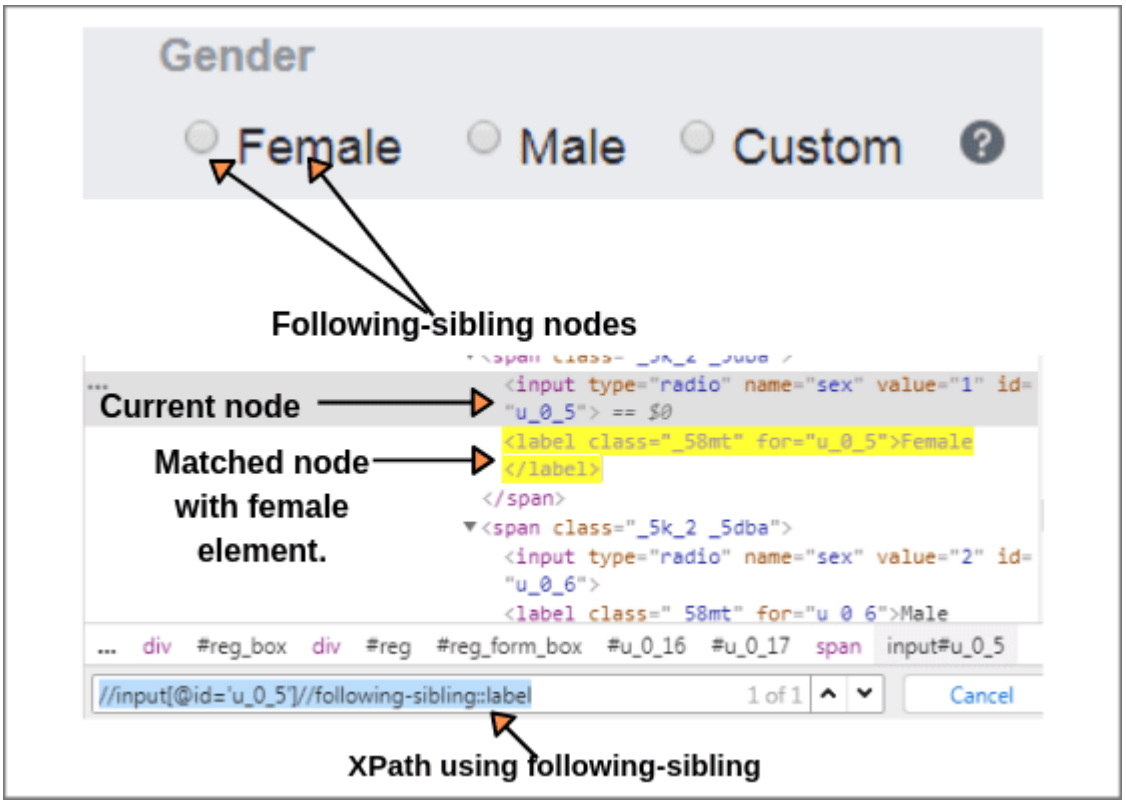
Similarly, on putting "2" as input,

XPath(Mobile number): //input[@id = 'u_0_r']//following::input[2] (1 of 1 matched).

8.Following-sibling Axis:

The following-sibling selects all sibling nodes after the current node at the same level. i.e. It will find the element after the current node.

For example, the radio button of female and female text both are siblings in the Facebook home page as shown in the below screenshot.



So, we will find XPath of current node i.e. XPath of the female radio button.

XPath(Radio button): //input[@id = 'u_0_5']

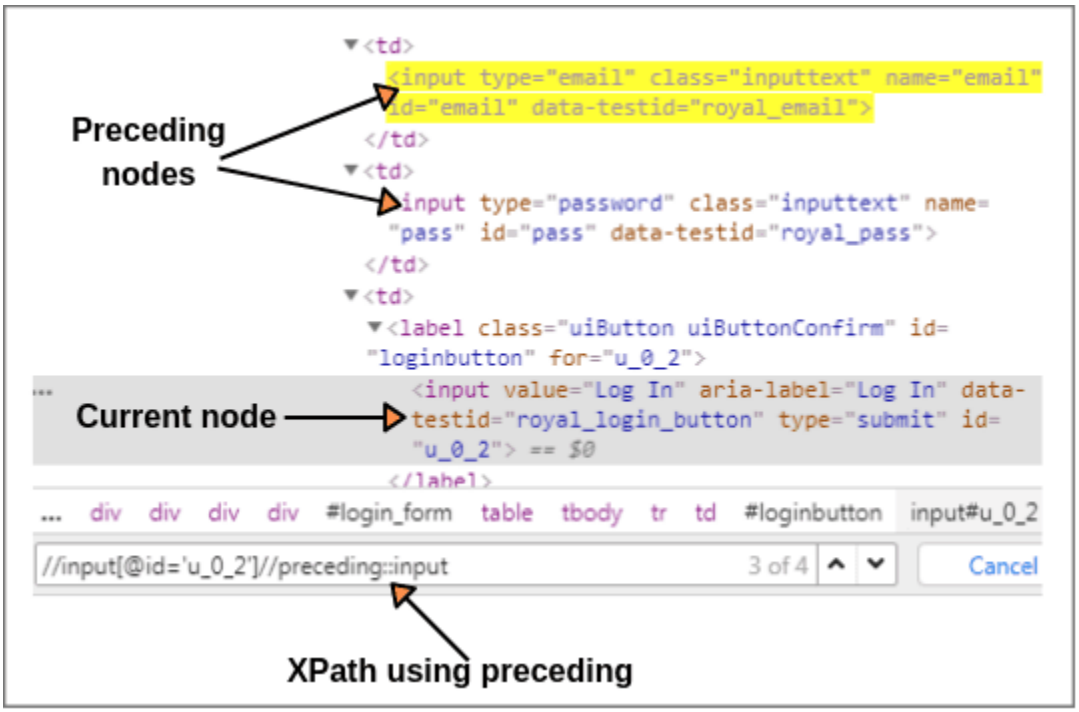
Using the following-sibling axis, we can easily find XPath of text "Female" at the same level.

XPath(Female): //input[@id = 'u_0_5']//following-sibling::label (1 of 1 matched).

The above expression identified one input nodes by using "following-sibling" axis.

9.Preceding Axis: The preceding axis selects all nodes that come before the current node in the document, except ancestor, attribute nodes, and namespace nodes.

Let us consider the login button as current node on the facebook web page as shown in below screenshot.



Let's first, find the XPath of current node i.e XPath of login button.

XPath(Current node): //input[@id = 'u_0_2']

Now we will select all nodes by using the preceding axis in the document that comes before the current node.

XPath: //input[@id = 'u_0_2']//preceding::input (1 of 4 matched).

The above expression identified all the input elements before "login" button using the preceding axis. 2 of 4 matches nodes are matched with Email and Password input element.

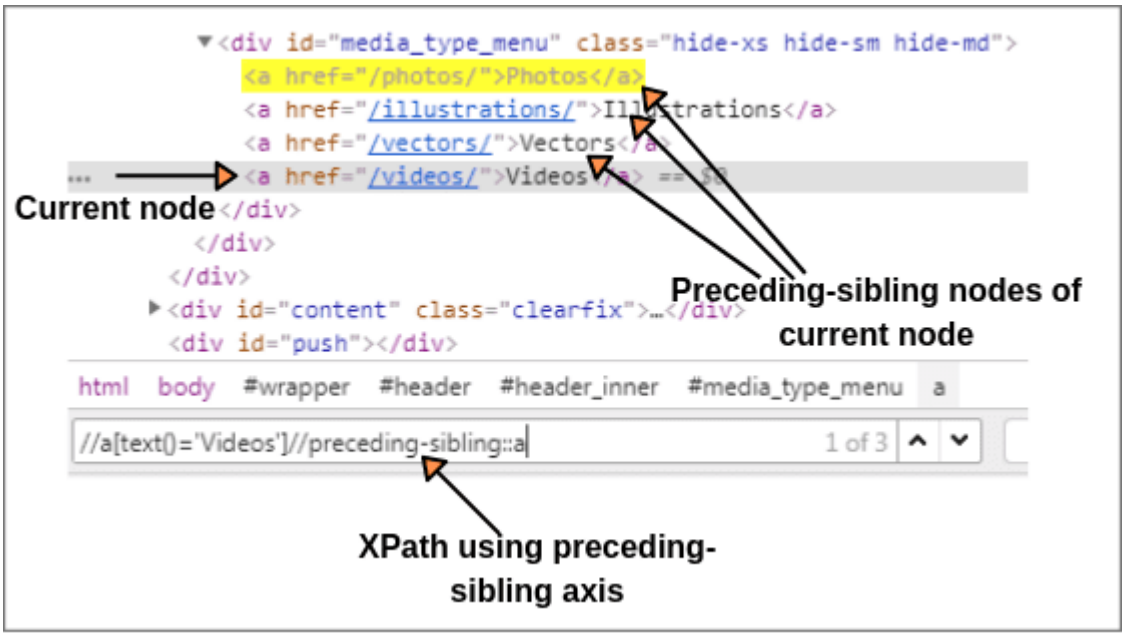
If you want to focus on any particular element like "Email" or "Password" then you can change the XPath according to the requirement by putting [1],[2]……and so on. For example:

XPath(Email): //input[@id = 'u_0_2']//preceding::input[2] (1 of 1)

XPath(Password): //input[@id = 'u_0_2']//preceding::input[1]

10. Preceding-sibling Axis: The preceding-sibling axis selects all siblings before the current node. Let's take an example to understand the concept of the preceding-sibling axis.

Open web page www.pixabay.com, right-click on videos link and go to inspect option. Let's consider videos link as current node as shown in below screenshot and find the XPath of current node by using text() method.



XPath(Current node): //a[text() = 'Videos']

Now we will find all nodes using preceding-sibling axis of the current node in the document.

XPath: //a[text() = 'Videos']//preceding-sibling::a (1 of 3)

The above expression identified three nodes before the current node (videos link) as shown in above screenshot. Using this expression, we can easily find XPath of preceding-sibling elements like Photos, Illustrations, and Vectors like this:

XPath(Photos): //a[text() = 'Videos']//preceding-sibling::a[3]

XPath(Illustrations): //a[text() = 'Videos']//preceding-sibling::a[2]

XPath(Vectors): //a[text() = 'Videos']//preceding-sibling::a[1]

11.Attribute Axis: This axis selects the element node on the basis of the attribute identifier (@) of the current node. If the current node is not an element node, this axis is empty. The expressions attribute::type and @type both are equivalent.

For example: Open the webpage www.pixabay.com, right-click on the search input box, and go to inspect. We can write XPath of search input box (current node) using the attribute axis.

XPath(Search box): //input[attribute::name = 'q']

12.Namespace Axis: The namespace axis is one of 13 XPath axes that selects all namespace nodes associated with current node. If the current node is not an element node then this axis will be empty.

The web elements in the *XML DOM* are related to each other via a hierarchical structure. *XPath* provides specific attributes that are called "XPath Axis", and these use the relationship between various nodes to locate those nodes in the *DOM* structure. The following table shows a few of those *Axis*, which can locate the elements on a web page using *XPath in Selenium*.

Axis	***Description***
ancestor	This axis locates the ancestors of the current node, which includes the parents up to the root node.
ancestor-or-self	This axis locates the current node as well as its ancestors.
attribute	This axis specifies the attributes of the current node.
Child	This axis locates the children of the current node
descendant	This axis locates the descendants of the current node, i.e., the node's children up to the leaf node.
descendant-or-self	This axis locates the current node and its descendants.
following	This axis locates all nodes that come after the current node.
following-sibling	This axis locates the below siblings of the context node. Siblings are at the same level as the current node and share its parent.
parent	This axis locates the parent of the current node.
preceding	This axis locates all nodes that come before the current node.
Self	This axis locates the current node.

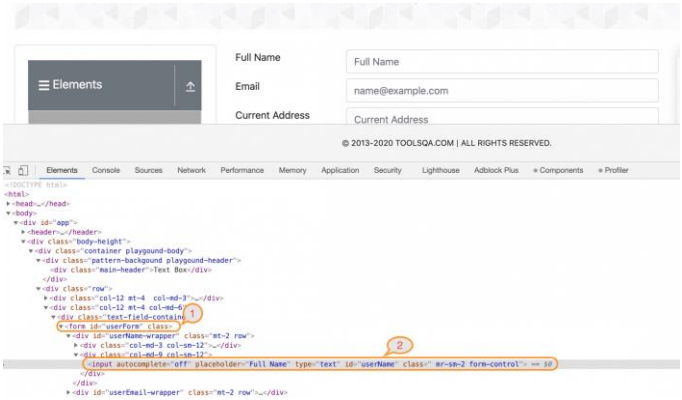
Let's understand a few of them in more detail with examples in the following sections:

Xpath Ancestor Axis:

The ancestor axis is used in *XPath* to select *all the parent (and parent of the parent)* elements of the current node. It's beneficial in scenarios where we can identify the node of the child element, but we are not able to recognize its parent or grandparent nodes. For these scenarios, we can take the child node as a reference point, and we can use the "***ancestor***" axis to recognize its parent node.

1 //tag[@attribute ='Attribute_Value']//ancestor::parent_node

Let's have a look at the following example. Let's say we want to locate the ***userForm****(shown by marker 1)* on the *webpage* "***https://demoqa.com/text-box",*** but due to its dynamic properties, we cannot directly identify it. But we can identify the "***Full Name***" label(*shown by marker 2*) present on the page. Here we can use the ***ancestor axis*** to identify the parent node.



Xpath Example: The *XPath Expression* for the *Full Name* label is:

1 **//label[text()=”Full Name”]**

But we want to locate the ***<form> tag,*** which is two hierarchy above this element. Let’s write the

1 //label[text()="Full Name"]/ancestor::form

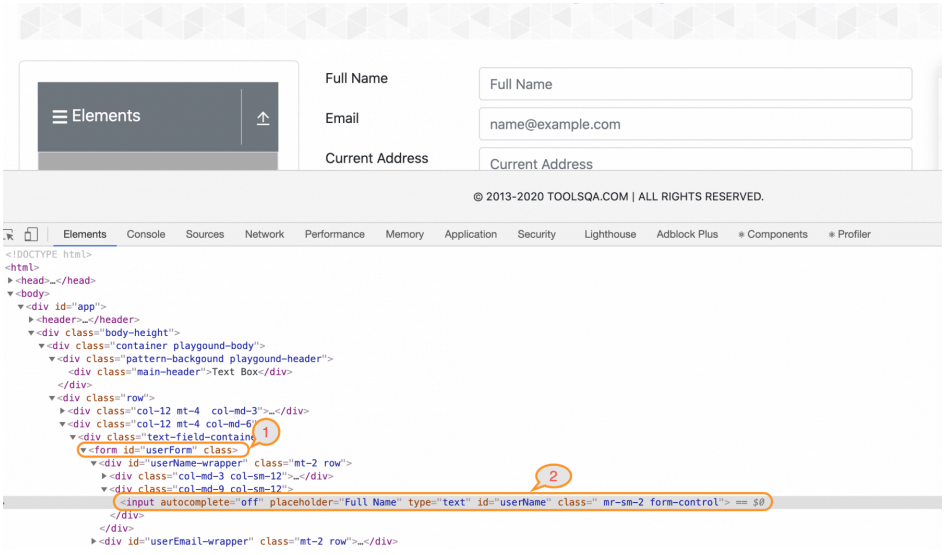
At first, we wrote *XPath* to locate the child node and then used the ***ancestor axis*** to find the form node. One thing that we should keep in mind is, if there is more than one parent node with the same tag, then this same *XPath* will identify several different elements.

XPath Child Axis: ***Child Axis*** in *XPath* is used to find all the ***child nodes of the current node***. Contrary to the ancestor axis, the child is quite helpful in locating elements if we can locate the parent element of the required node.

Syntax and usage of the child axis in XPath is:

1 //tag[@attribute ='Attribute_Value']//child::child_node

Let’s see the previous example; assume that we can locate the “***form***” element but not able to find the “***Full Name***” label.



Xpath Example: The *XPath Expression* for the ***form*** node is:

1 //form[@id='userForm']

But we want to locate the ***<label> tag,*** which is two hierarchy below this element. Let's write the *XPath Expression* that can locate the label node.

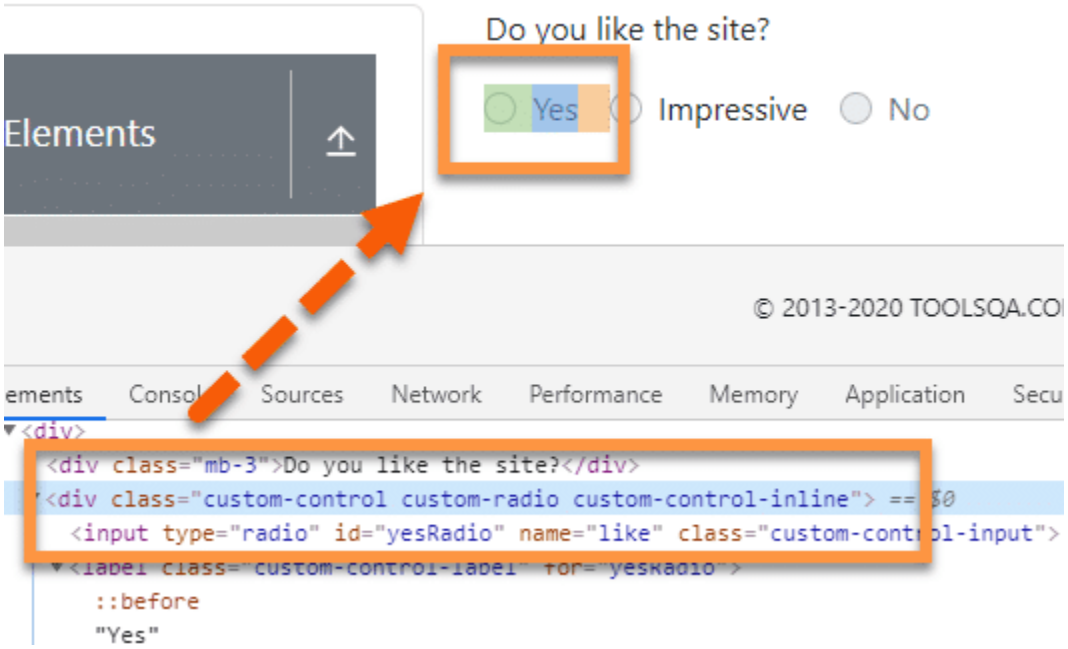
1 //form[@id='userForm']/child::div[1]//label

Here we can see that this *XPath* comprises two parts, first before "***child*** " and second part after the child. Both parts have to be in the proper format. Here we have first located the user form by using '*id.'* Then we used the "***child*** " ***axis*** to navigate to the child node. Here we have used *index* "***[1]*** " as we wanted to start from the first "***div***" if we wanted to start from the second div in the sequence, we could index it as "***[2]*** ". We have also used ***double slash,*** as we wanted to escape the *next div* to directly move to ***label tag,*** by using the *Relative XPath*.

XPath Descendant Axis: **The** *Descendant Axis* **is used to access** *all the children and sub-children of the current node*. **It is similar to the *child axis*, but the fundamental difference between descendant and child is; descendant identifies only the children and sub-children of the same node.**

Syntax and usage of the descendant axis in the XPath:

1 //node[attribute='value of attribute']//descendant::attribute



In the above image, we can see that the parent node, i.e., *div* contains two different descendant nodes or child nodes. First is the ***radio button*** defined by the ***input*** tag, and the second one is a ***label tag*** that has the label for the radio button.

Xpath Example: Let's assume we can recognize the ***parent tag,*** but need to locate the *radio button*. We can do this by using the *descendant axis* to create the following *XPath*.

1 //div[@class= 'custom-control custom-radio custom-control-inline']/descendant::input

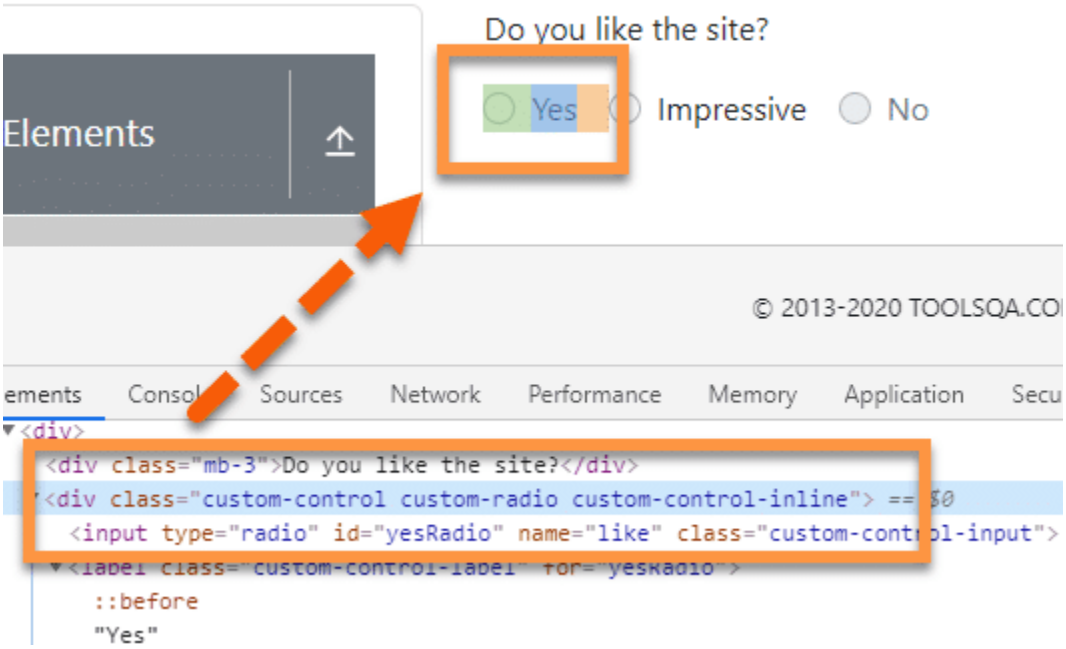
So, this *Xpath* will first search the node "***div*** " with the given class and then search the *input* tag using the ***descendant axis***.

XPath Parent Axis: The ***Parent Axis*** is similar to the *ancestor* axis. It recognizes the immediate parent of the current node. The significant difference between *parent and ancestor* is; *ancestor* will recognize all upper hierarchical nodes, whereas the parent axis will only locate the ***immediate parent node***.

Syntax and usage Parent Axis in XPath:

1 //tag[@attribute ='Attribute_Value']//ancestor::parent_node

Let's take an example of the element we used in the previous section:



Xpath Example: Here we have an *input* element that can be recognized. But let's assume we want to locate the *parent* element of this node. To do that, we will use the ***parent axis*** with the *XPath*. So, the *XPath Expression* for the above elements will become:

1 //input[@id='yesRadio']/parent::div

If we have used ancestor here, the *XPath* would have returned all the upper hierarchical nodes, but as we have used ***parent,*** it will return the ***immediate parent node*** of the currently identified node.

XPath Following Axis: **The** *following axis* **locates the element** *after the current node*. **It marks the *current node* as the base and finds the required element present in the *DOM* after the current node.**

Syntax and usage of the following axis in the XPath:

1 **//node[attribute='value of attribute']//following::attribute**

Xpath Example: In the above image, let's assume we have to locate the **Full Name** textbox by using ***id. Still, we*** also want to locate elements present after that *textbox*, for example, "***Current Address***" multi-line *textbox* or text area. So, the *XPath Expression* for the ***Current Address*** will be:

1 //input[@id="userName"]/following::textarea

One thing that we should note is when we use the ***following axis***, it will recognize *multiple tags* if present in the *DOM*. If there is more than one "***Textarea***" node after the given *textbox*, then the *XPath* will point to all of them. The best practice is the always create unique *XPaths*, and if that's not possible, we may use index.

XPath Following sibling Axis: The Xpath *following sibling axis* is similar to the *following* axis. *Following-sibling* recognizes only *sibling nodes*, i.e., *the nodes on the same level of the current node*. For example, in the below image, both the highlighted "*div* " are on the same level, i.e., are siblings, whereas the "*div* " just above them is the parent.

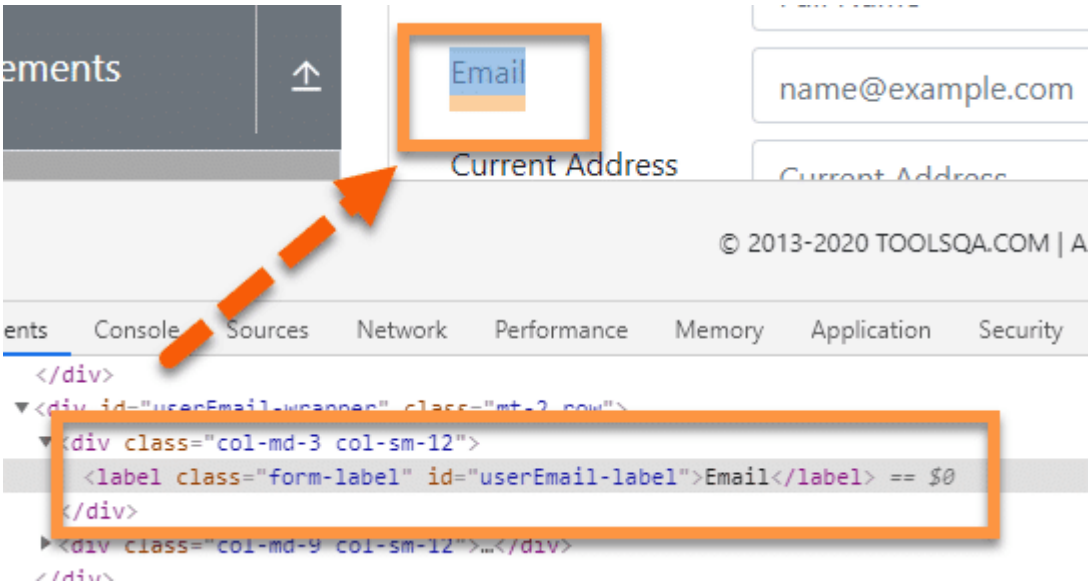
```
        <input autocomplete="off" placeholder="Full Name" type="text" id="u
      </div>
    </div>
  ▼<div id="userEmail-wrapper" class="mt-2 row">
    ▶<div class="col-md-3 col-sm-12">…</div>
    ▼<div class="col-md-9 col-sm-12">
        <input autocomplete="off" placeholder="name@example.com" type="emai
      </div>
    </div>
```

Syntax and usage of the following-sibling axis in the XPath:

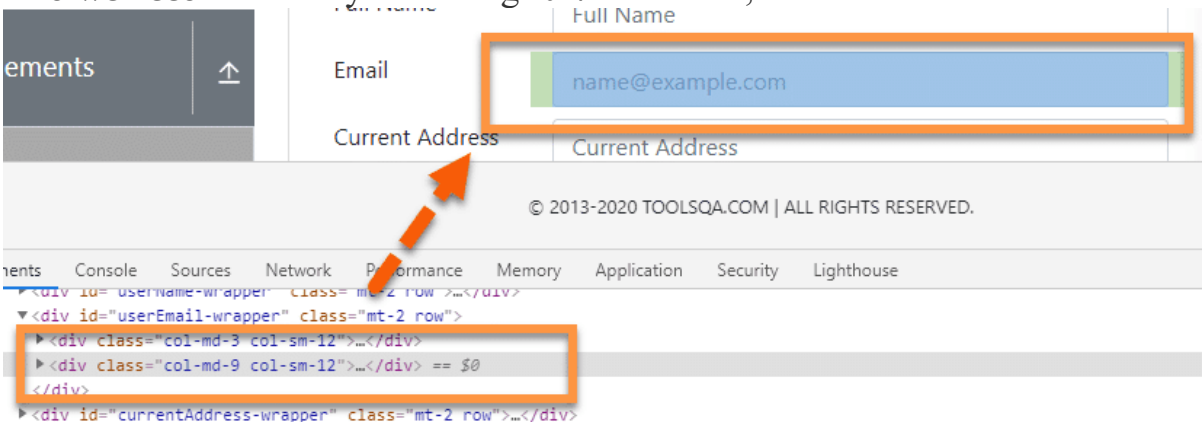
1//node[attribute='value of attribute']//following-sibling::attribute

Let's understand the details with the help of the following example on the web page "***https://demoqa.com/text-box***" :

Assume we have identified the following label "***Email*** " represented by the <***div***> element:



And we need to identify its sibling "***div*** " element, as shown below:



Xpath Example: To identify this, we can use the XPath ***following sibling axis***. So, the *XPath Expression* for this scenario will be:

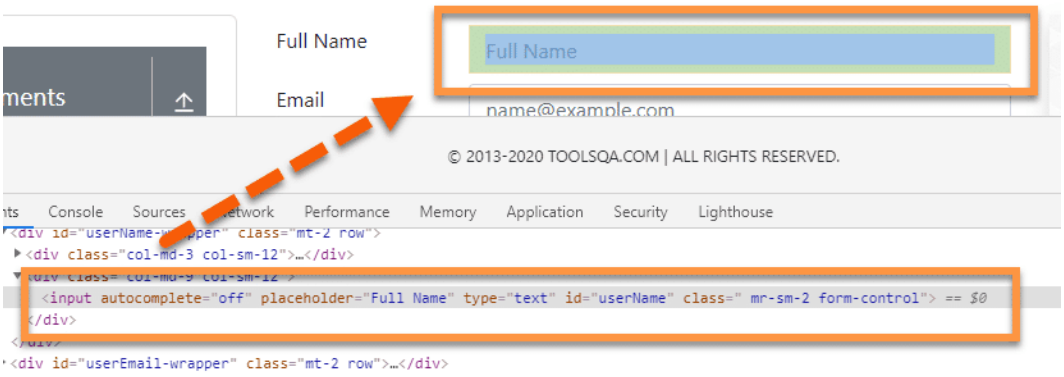
1//div[@class='col-md-3 col-sm-12']/following-sibling::div

But if there are multiple siblings with the same node, then *XPath* will identify all those various elements.

*XPath Preceding Axis:*The *preceding axis* is used in *XPath* to select *all the nodes present before the current node*. We use it when we can recognize any element above the mentioned element.Syntax and usage of the Preceding Axis in the XPath will be:

1//node[attribute='value of attribute']//preceding::attribute

Let's try to understand the usage of the ***Preceding Axis***, with the help of the following example:



In the above scenario, as the highlighted textbox contains "***id***", we can quickly locate it, but let's assume that we need to find the ***label*** of the element which is present ***just before that current node***. To handle these scenarios, we use the "***preceding axis".*** Now, we can write an *XPath Expression* to locate the *label element* as below:

1//input[@id='userName']/preceding::label

Similar to what we discussed earlier, if there is more than one element with the same node above the current element, *XPath* will return multiple elements. We always need to make sure that we write unique *XPaths* for efficient element identification.

XPath Example: Usage of XPath functions and Axes in Selenium

The following code snippet shows a sample example of how we can use the various *functions and axes* provided by *XPath in Selenium* to identify and locate web elements uniquely:

```
1. package TestPackage;
2. import org.openqa.selenium.By;
3. import org.openqa.selenium.WebDriver;
4. import org.openqa.selenium.chrome.ChromeDriver;
5. public class XPathDemo {
6. public static void main(String args[]) {
7. System.setProperty("webdriver.chrome.driver",
   "C:\\Selenium\\chromedriver\\chromedriver.exe");
8. WebDriver driver = new ChromeDriver();
9. driver.get("https://demoqa.com/text-box");
10.//Using contains() to locate full name and enter data
11.driver.findElement(By.xpath("//input[contains(@id, 'userN')]")).sendKeys("User
   Name");
12.//using placeholder
```

```
13.driver.findElement(By.xpath("//input[contains(@placeholder,
    'example')]")).sendKeys("Using Placeholder");
14.//using start-with()
15.driver.findElement(By.xpath("//input[starts-
    with(@placeholder,'Fu')]")).sendKeys("Using start with");
16.//using text() to get label
17.String text = driver.findElement(By.xpath("//label[text()='Email']")).getText();
18.System.out.println(text);
19.//using AND operator to locate full name
20.driver.findElement(By.xpath("//input[@placeholder ='Full Name' and @type =
    'text']")).sendKeys("AND operator");
21.//using OR operator to locate full name
22.driver.findElement(By.xpath("//input[@placeholder ='Full Name' or @type =
    'text']")).sendKeys("OR operator");
23.//using ancestor to locate form tag
24.boolean bol =driver.findElement(By.xpath("//label[text()='Full
    Name']/ancestor::form")).isDisplayed();
25.System.out.println("Form is displayed : "+bol);
26.//using child to locate full name textbox from form
27.String label =
    driver.findElement(By.xpath("//form[@id='userForm']/child::div[1]//label")).getText();
28.System.out.println("The label text is : "+ label);
29.//using decendent axis to locate yes radio
30.driver.get("https://www.demoqa.com/radio-button");
31.driver.findElement(By.xpath("//div[@class= 'custom-control custom-radio custom-
    control-inline']/descendant::input/following-sibling::label")).click();
32.//using parent axis to locate yes radio
33.boolean bo =
    driver.findElement(By.xpath("//input[@id='yesRadio']/parent::div")).isSelected();
```

```
34.System.out.println("The Yes radio is selected : "+bo);
35.//using following axis to locate current address
36.driver.get("https://demoqa.com/text-box");
37.driver.findElement(By.xpath("//input[@id=\"userName\"]/following::textarea")).sendKeys("Text Area locate following");
38.//using following-sibling to locate email
39.driver.findElement(By.xpath("(//div[@class='col-md-3 col-sm-12']/following-sibling::div/input)[2]")).sendKeys("abc@xyz.com");
40.//using preceding-axis to locate full name
41.String preceding = driver.findElement(By.xpath("//input[@id='userName']/preceding::label")).getText();
42.System.out.println("The value of preceding : "+preceding);
43.}
44.}
```

When we run the above test script, it will locate all the mentioned web elements uniquely. This way, we can utilize the *XPath* functions and axes to *choose effective XPath* for uniquely locating the elements on a web page.

Example: /html/body/div[1]/div/div[3]/div[1]/img

Types of XPaths
---
1) Absolute Xpath /Full XPath
/html/body/div[1]/div/div[3]/div[1]/img

2) Relative Xpath / Partial XPath
//*[@id="divLogo"]/img

Diff between Absolute & Relative Xpaths
---
1) Absolute xpath starts from root node
Relative xpath directly jump to element on DOM.

2) Absolute xpath start WITH /
Relative xpath STARTS WITH //

3) in Absolute xpath we use only tags/nodes
In Relative xpath we use attributes.

Syntax of Relative Xpath

Syntax: //tagname[@attribute='value']
Example: //input[@name="txtUsername"]
//*[@name="txtUsername"]

How to capture Xpath?
---
1) Right click on element-->Inspect--> Highlight html-->Right click-->Copy Xpath
2) Chropath extension
3) SelectorsHub

Which XPath is prefered? Why?
Ans: Relative

/html/body/div[1]/div/div[3]/div[1]/img

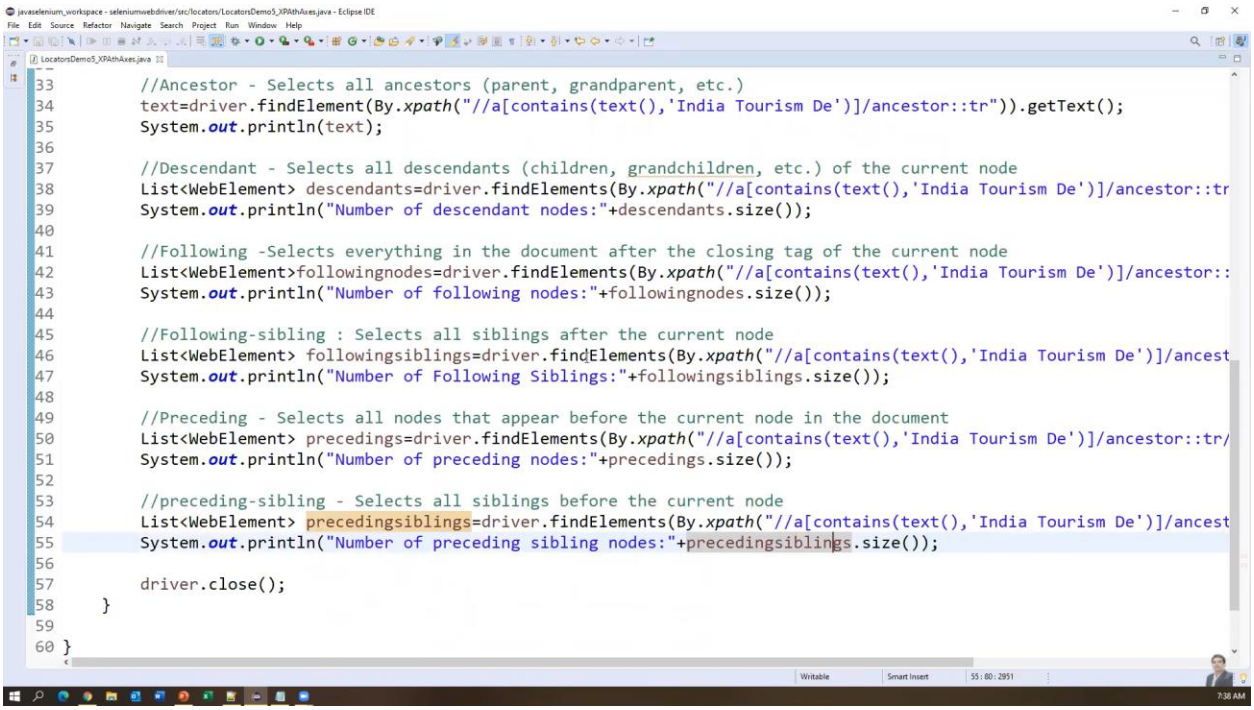
//*[@id="divLogo"]/img

Locators - XPath Axes

XPath axes are those axes that are used to search for the multiple nodes in the XML document from the current node context
These methods are mainly used when the web element is not identified with the help of ID, name, class name, link text, CSS

XPath Axes

Self
Parent
Child
Ancestor
Descendant
Following
Following-sibling
Preceding
preceding-sibling

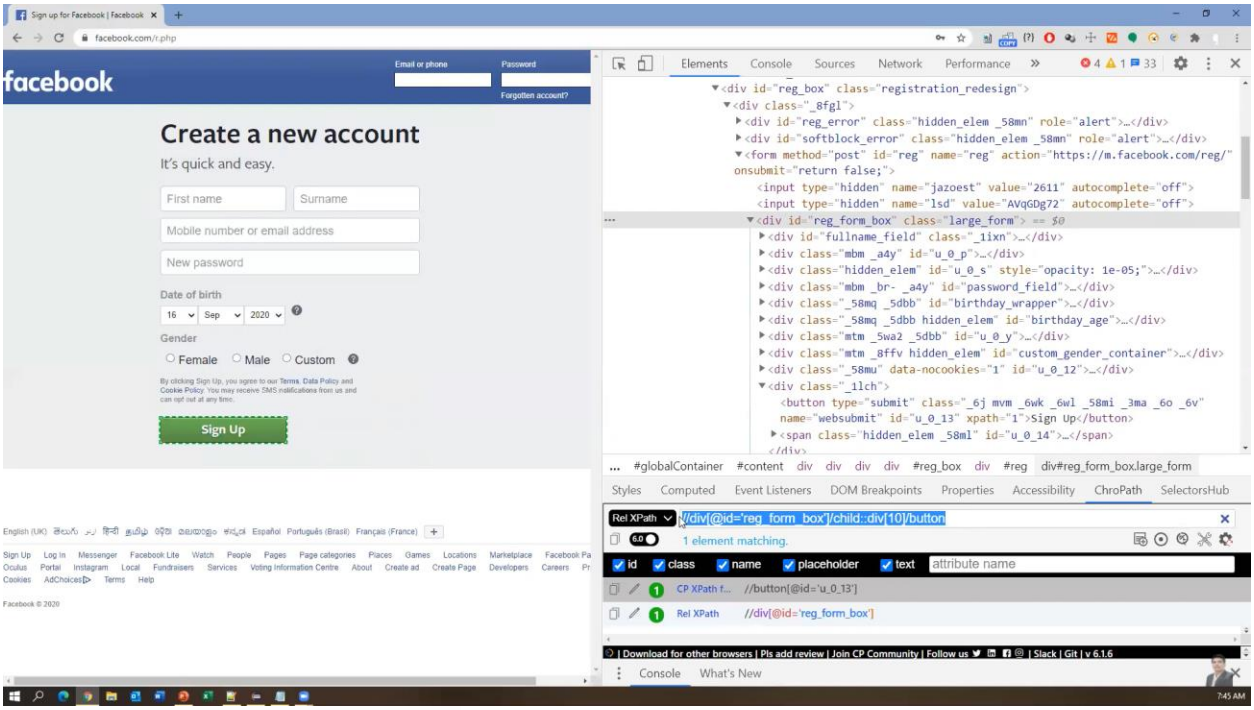


```
33          //Ancestor - Selects all ancestors (parent, grandparent, etc.)
34          text=driver.findElement(By.xpath("//a[contains(text(),'India Tourism De')]/ancestor::tr")).getText();
35          System.out.println(text);
36
37          //Descendant - Selects all descendants (children, grandchildren, etc.) of the current node
38          List<WebElement> descendants=driver.findElements(By.xpath("//a[contains(text(),'India Tourism De')]/ancestor::tr
39          System.out.println("Number of descendant nodes:"+descendants.size());
40
41          //Following -Selects everything in the document after the closing tag of the current node
42          List<WebElement>followingnodes=driver.findElements(By.xpath("//a[contains(text(),'India Tourism De')]/ancestor::
43          System.out.println("Number of following nodes:"+followingnodes.size());
44
45          //Following-sibling : Selects all siblings after the current node
46          List<WebElement> followingsiblings=driver.findElements(By.xpath("//a[contains(text(),'India Tourism De')]/ancest
47          System.out.println("Number of Following Siblings:"+followingsiblings.size());
48
49          //Preceding - Selects all nodes that appear before the current node in the document
50          List<WebElement> precedings=driver.findElements(By.xpath("//a[contains(text(),'India Tourism De')]/ancestor::tr/
51          System.out.println("Number of preceding nodes:"+precedings.size());
52
53          //preceding-sibling - Selects all siblings before the current node
54          List<WebElement> precedingsiblings=driver.findElements(By.xpath("//a[contains(text(),'India Tourism De')]/ancest
55          System.out.println("Number of preceding sibling nodes:"+precedingsiblings.size());
56
57          driver.close();
58      }
59
60  }
```

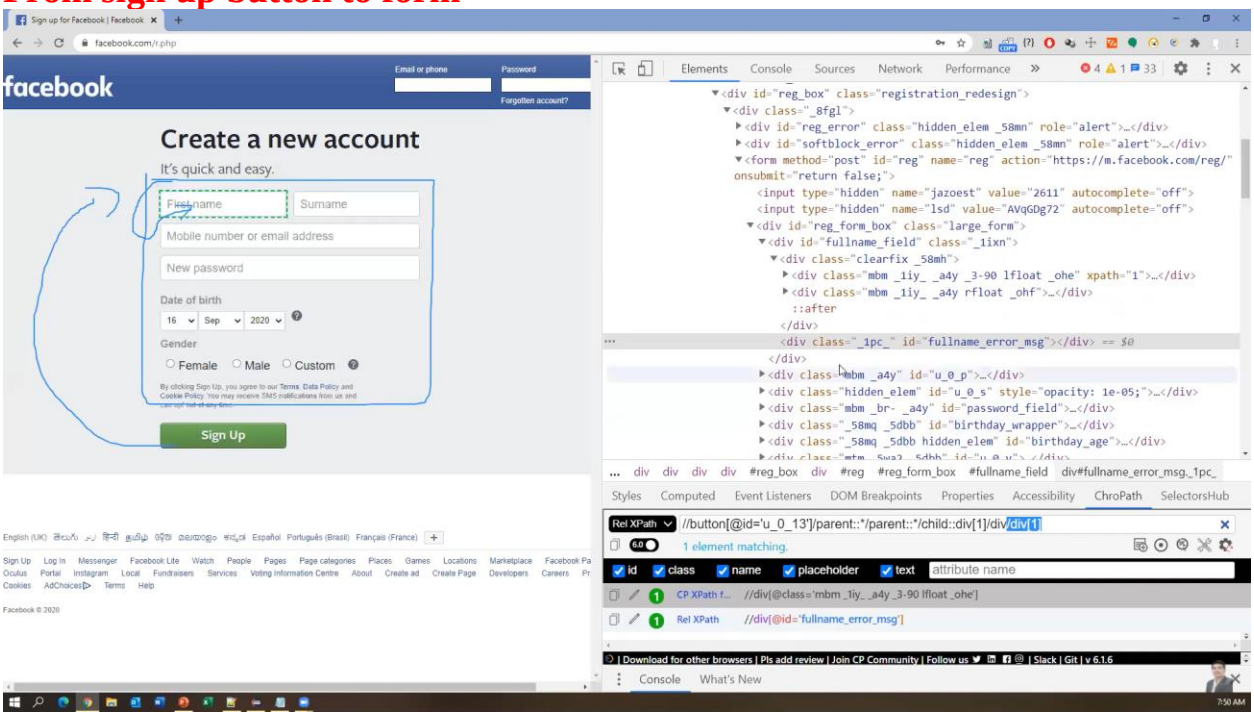
From form to signup button

1) Find the sign-up button from the registration form present in the Facebook application. (child)
//div[@id='reg_form_box']/child::div[10]/button

2) Locate Firstname field from SignUp button in facebook (Parent)



From sign up button to form



Session17.txt - Notepad

File Edit Format View Help

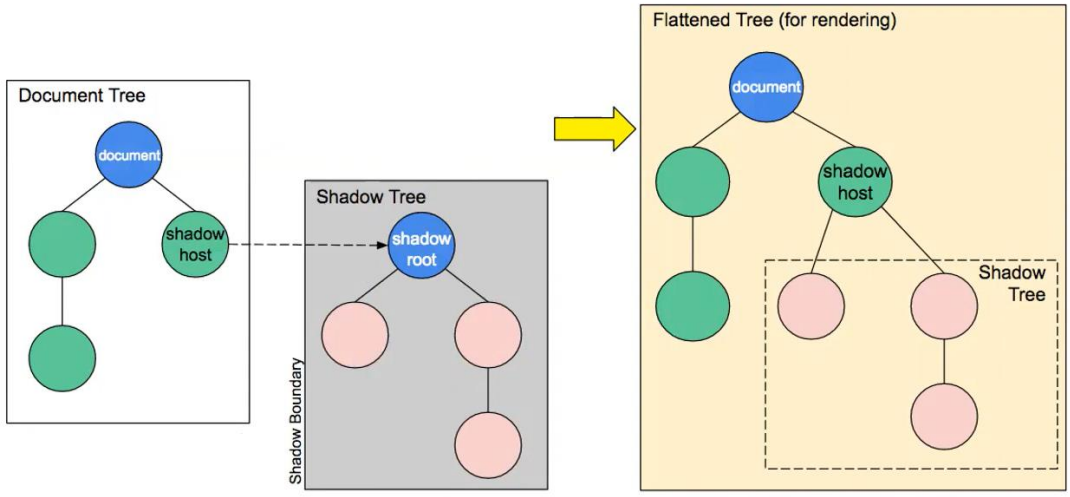
Preceding
preceding-sibling

```
1) Find the sign-up button from the registration form present in the Facebook application. (child)
//div[@id='reg_form_box']/child::div[10]/button

2) Locate Firstname field from SignUp button in facebook (Parent)
//button[@id='u_0_13']/parent::*/parent::*/child::div[1]/div/div[1]

3) Identify the Password from Mobile number filed in facebook.(Following)
//input[@id='u_0_r']/following::input[2]

4) Locate Mobile number from newpassword field in facebook (preceding)
//input[@id='password_step_input']/preceding::input[2]

5) Locate surname from female radio button in facebook(Ancestor)
//input[@id='u_0_4']/ancestor::div[2]/div[1]/div/div[2]
//input[@id='u_0_4']/ancestor::div[2]/input[@id='u_0_o']

6) Identify the search text box from the Google search button present in the Google search home page. (Parent)
//div[@class='FPdoLc tfB0Bf']//input[@name='btnK']/parent::*/parent::*/parent::*/div[1]

7) Identify the Today's Deals link from the amazon search text box present in the amazon home page.(Following)
//input[@id='twotabsearchtextbox']/following::a[contains(text(),"Today's Deals")]

8) Identify the Hello, Signin from the amazon search text box present in the amazon home page. (Following)
//input[@id='twotabsearchtextbox']//following::span[contains(text(),'Hello, Sign in')]

9) Identify Mobiles link which is part of Menu bar - Amazon  (Descendant)
//div[@id='nav-xshop']/descendant::a[contains(text(),'Mobiles')]
```

Ln 9, Col 1 100% Windows (CRLF) UTF-8 8:13 AM

What is DOM & Shadow DOM ?

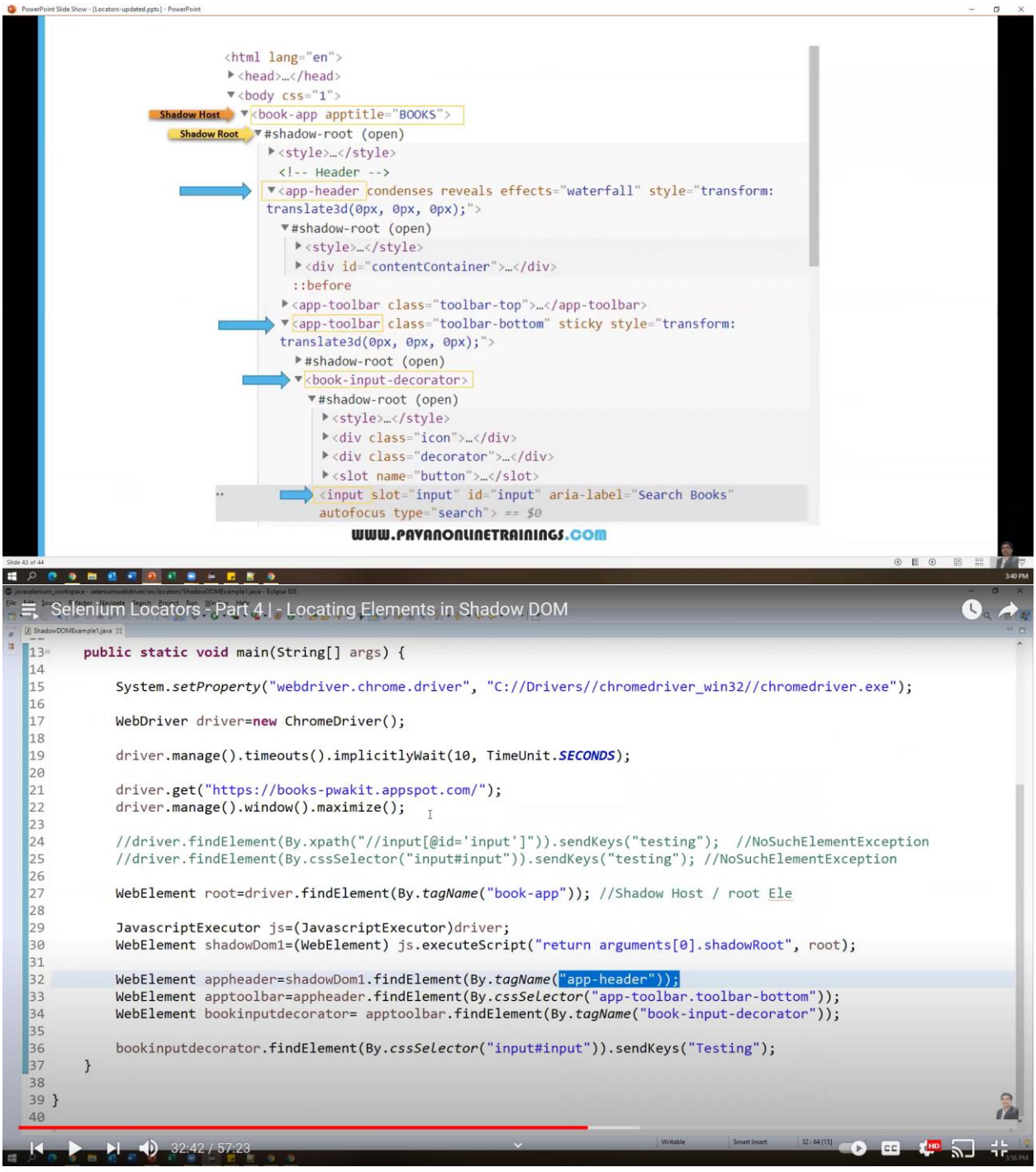
- **DOM (Document Object Model)** is Tree representation of HTML nodes.
- **Shadow DOM** subset of **DOM.**
- **Shadow host** is the regular node that the **Shadow DOM** is attached.



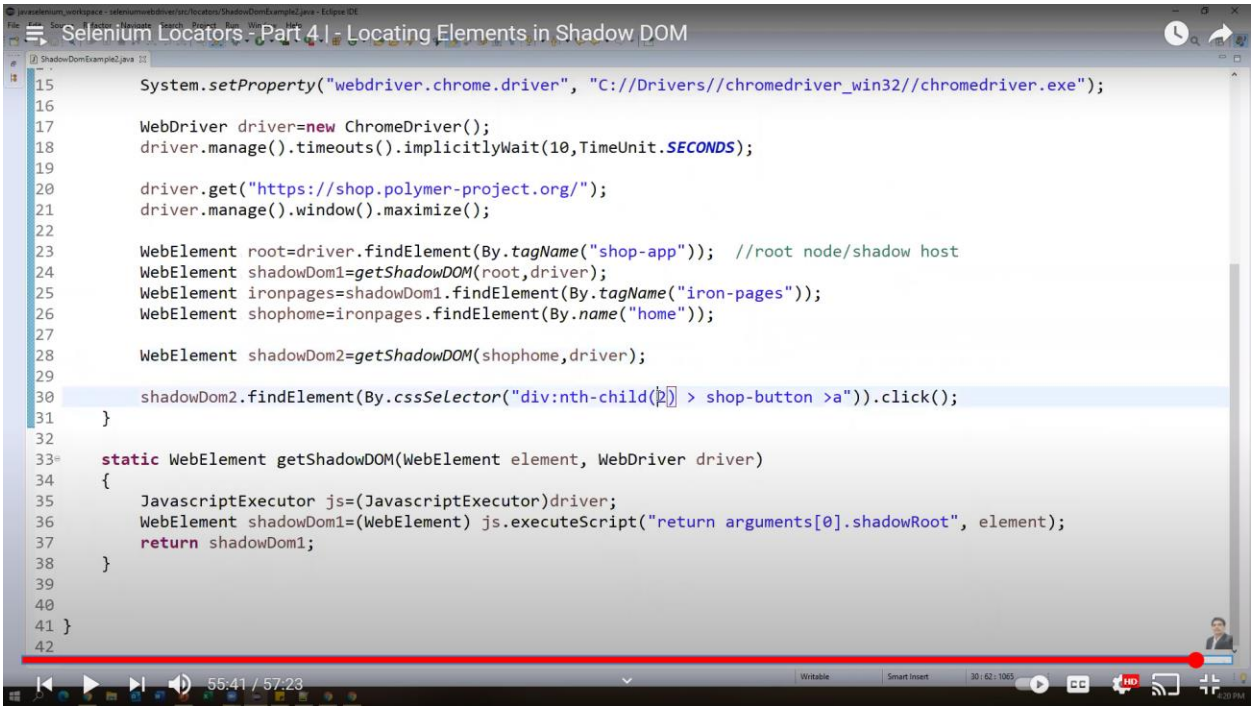
Shadow dom cant accept the xpaths.
It accept only css selectors , but css selectors are not using directly.

Site name: https://books-pwakit.appspot.com/

Locating Searching area



Multiple shadowdoms in web pages:



*Session-16.txt - Notepad
File Edit Format View Help

```
---------------------------------
id="firstname"

//tagname[contains(@attribute,'value')]

//input[@id='firstname']    --> Basic xpath

//input[contains(@id,'first')]   --> using contains()


//input[@id='start']
//input[@id='stop']

//input[contains(@id,'st')] --> Dynamic xpath

//input[starts-with(@id,'st')] --> Dynamic xpath

chained xpath
-----------
//form[@id='searchbox']//input[4]
//form[@id='searchbox']//input[@id='search_query_top']


//form[@id='searchbox']//button
//form[@id='searchbox']//button[@name='submit_search']
```

*Session15.txt - Notepad
File Edit Format View Help

```
Locators in Selenium
-----------------------


CSS Selectors
----------------
Tag and ID                      Tag#id
Tag and Class                   Tag.class
Tag and attribute               Tag[attribute=value]
Tag, class and attribute        Tag.class[attribute=value]
```

CSS Selector – *Tag* and *ID*

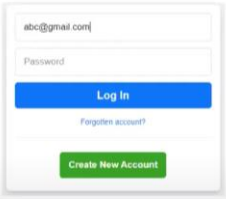
https://www.facebook.com/


```
<input type="text" class="inputtext _55r1 _6luy" name="email" id="email" data-testid="royal_email" placeholder="Email address or phone number" autofocus="1" aria-label="Email address or phone number"> == $0
```

```
driver.findElement(By.cssSelector("#email")).sendKeys("abc@gmail.com");

(or)

driver.findElement(By.cssSelector("input#email")).sendKeys("abc@gmail.com");
```

CSS Selector – *Tag* and Class

https://www.facebook.com/




```
<input type="text" class="inputtext _55r1 _6luy" name="email" id="email" data-testid="royal_email" placeholder="Email address or phone number" autofocus="1" aria-label="Email address or phone number"> == $0
```

```
driver.findElement(By.cssSelector(".inputtext")).sendKeys("abc@gmail.com");

(or)

driver.findElement(By.cssSelector("input.inputtext")).sendKeys("abc@gmail.com");
```

CSS Selector – *Tag* and Attribute

https://www.facebook.com/


```
<input type="text" class="inputtext _55r1 _6luy" name="email" id="email" data-testid="royal_email" placeholder="Email address or phone number" autofocus="1" aria-label="Email address or phone number"> == $0
```

```
driver.findElement(By.cssSelector("[name=email]")).sendKeys("abc@gmail.com");

(or)

driver.findElement(By.cssSelector("input[name=email]")).sendKeys("abc@gmail.com");
```

```
driver.get("https://www.facebook.com/");
driver.manage().window().maximize(); // maximize web page

//Tag & ID
//driver.findElement(By.cssSelector("input#email")).sendKeys("David");
//driver.findElement(By.cssSelector("#email")).sendKeys("David");

// Tag & Class
//driver.findElement(By.cssSelector("input.inputtext")).sendKeys("John");
//driver.findElement(By.cssSelector(".inputtext")).sendKeys("John");

//Tag & attribute
//driver.findElement(By.cssSelector("[name=email]")).sendKeys("Smith");
//driver.findElement(By.cssSelector("input[name=email]")).sendKeys("Smith");

//Tag , class & attribute
driver.findElement(By.cssSelector("input.inputtext[data-testid=royal_email]")).sendKeys("Smith");
driver.findElement(By.cssSelector("input.inputtext[data-testid=royal_pass]")).sendKeys("abc");
```

AGENDA

WHAT IS CROSS BROWSER TESTING?
WHY DO YOU NEED CROSS BROWSER TESTING?
HOW TO PERFORM CROSS BROWSER TESTING?
ADVANTAGES OF CROSS BROWSER TESTING

WHAT IS CROSS BROWSER TESTING?

Cross browser testing refers to testing the application in multiple browsers like IE, Chrome, Firefox so that we can test our application effectively. IE, Chrome, Firefox so that we can test our application effectively.

WHY CROSS BROWSER TESTING?

1. Browser computability with different OS
2. Image Orientation
3. Compatibility with new web framework
4. JS implementation can be different