**GENERAL TOPIC TO KEEP CHECKING**.

**Plan your recipe.**

**READ and Watching video tutorial knowledge and Understanding is different from Hands-on practical exercises. YOUR EXPERIENCE COMES FROM PRACTICAL HANDS-ON WORK.**

YOU HAVE 24HOURS EVERYDAY, PRIORITIZE AND USE IT WISELY…THINK OF TASK TO ACHIEVE.

**DEEP LEARNING. SET GOALS TO BE ACHIEVED, IT THE GOAL THAT WILL KEEP YOU GOING IN THAT HABIT OF CONSISTENCY. E.g to learn and understand how to use JavaScript program.**

Technical Skills – knowledge and skills about a specific software technology.

Study Software Development, Software Engineering and Software Programming.

**Note: Learn & Practice, then apply it on the project.**

**Object Oriented Programming thinking, Think like a programmer**. (State and behaviour).

HOW does each programming language implement OOP in their program syntax.

Learn to do framework setup and Object-Oriented Principles (Programming).

**YOU NEED TO KNOW DIFFERENT TYPES OF DATA and their Object TYPES –** Value types (Data) and Reference value (Data).

Data on the right side of the = sign is either a Reference value, Primitive value (value type).

Understand Value Types, Reference Types and Primitive Value (Data) and Reference value (Data).

How to access an object value. Dot and bracket notations.

Understand each PROGRAM (inputs & process) or Object – their state & behaviour.

**ReferenceType referenceVariable = new ReferenceValue;**

**The ReferenceValue will create an object in the computer OS and return a reference Value memory location address.**

**You must determine what type of object was returned, e.g Object, String, RestClient, int, and so on.**

AS YOU ARE UPSKILLING, ALSO STUDY THE BUSINESS SIDE TOO.

YOU ARE A TEST DEVELOPER (A software tester and A software Developer at the same time).

Explore what and how to use software testing tools.

**A good API makes it easy to develop a program by providing all the building blocks; the programmer puts the blocks together.**

Computer Science, Software Engineering and Development, Software architecture.

Knowledge(theory) and Skill (practical hands-on)

**Frontend tools (client tools) – Middleware tools (Web Services, API, RestAPI, SOAPUI, Servers, Services) – Backend tools (database like MongoDB, MySql, Microsoft server sql, Postgre, SQLlite)**

**BACK TO BASICS – WHAT and HOW, WHY, WHEN.**

**Review, Revisit, Re-learn, Re-read, Re-Practice. (LOGICAL guilds the PHYSICAL).**

**Know what happens behind the scenes, at the backend side, in the background– business logic processing side, underlying or base implementation.**

**STUDY and PRACTICE EVEN IF IT IS FOR 10 minutes.**

**7 days of studying and practicing a program.**

SDET – Test Automation Engineer (Software Quality Engineer).

Client - Server Architecture. (Where is the software application program installed or located – local or remote?)

Local and Remote computer (system) machines.

Types of clients and servers. (Determine if a software application program is either a client or server depending on the current context).

Computer Science (Hardware and Software systems), Software Engineering and Development (SDLC).

Back to Basics (WHAT and HOW).

Lifecycle of software PROGRAM.

Always have a design or development or construction plan.

Review, Re-Study everything – Lifelong Learning.

ALWAYS HAVE A PLAN, A PLAN IS YOUR MAP BEFORE YOU START THE JOURNEY.

WHAT AND HOW.

What is a System?

What is a software

**BUILD ALL KINDS OF PROJECTS – LEARN OTHER DEVELOPERS PROBLEM SOLVING LOGIC.**

**USE REAL WORLD OR LIFE EXAMPLES.**

**THINK LOGICAL LEFT, RIGHT, UP AND DOWN**

**Familiarize yourself, try, study and learn it, do it now, so that next time, it does not look new to you. You continue and gain more the next time you encounter it.**

Understand WHAT the theoretical logical knowledge is, HOW to Implement the practical hands-on exercises, Apply to a specific problem.

UNDERSTAND **Front-end – Middleware – Backend technologies** used for developing software application, **WHAT** software testing tools are built for testing these applications and **HOW** to use them to solve testing needs.

STUDY and familiarize and play around with **COMPUTER SOFTWARE, PROGRAMS and Programming languages, Scripting Languages**. E.g Specflow, TestNg, Nunit, Java, CSharp, RestAssured, Karate API, Selenium, Python and many more.

Packages, namespace, module contains API for programmers to solve a need programmatically.

**Computer Files / Document and Data –** file types are defined by the file extensions (they add or extend additional functionality to a software application) and **WHAT** kind of data can be stored in file and **HOW** to construct or create the data syntax vocabulary rules in the file and how to access data in the file**.**

**SYNTAX is set of Rules for constructing semantic statements for a language. Every line of code you construct has a purpose or reasons for a problem specific it solves.**

**Computer Programs allows a user to complete a specific task.**

* Every program has a documentation or API user guide. Look for it and read it.
* SDET – Test Developer, Test Automation Engineer
* Test Automation RoadMap
* Software Engineering, Computer Science and Software development.
* **Front-end, Middleware and Backend software.**
* Study order developers code, study how they logically solve a programming problem.
* Computer Networking, Computer programming, Computer software application, computer system.
* CICD, Jenkin, Selenium GRID
* ROI – Return on investment.
* UML diagram and Entity relationship modelling.
* Model an object, modular programming.
* All Objects communicate to each other using a common interface.
* **SOLID** principles.
* **Database – JDBC, MYSQL, ORACLE, MICROSOFT**
* RemoteWebdriver, desiredcapabilities.
* Unit Testing Frameworks - Junit, TestNG, NUnit
* **Logical Reasoning and Analysis.**
* **Understand what a software and types of software is.**
* **Computer hardware and Computer software or computer program.**
* **Computer Software (Program)**
* **System software, Operating Systems (Linux, windows, Android, IOS), Network Operating Systems.**
* **Application software (Command-Line, Software libraries – API, COM), GUI web-based, desktop, mobile, virtual software application.**
* **Mathematics and Physics.**
* **Understand singular and plural English.**
* Characters for data – data forms information – information forms knowledge – knowledge forms wisdom.
* **Forms of learning – read text, watch videos, visual images or diagrams, explain and apply to solve a specific problem**.
* Base64 mechanism.

**LOGICAL THINKING – THINK IN OOP like a PROGRAMMER**

**BUILD PROBLEM SOLVING SKILLS**

===============================================================

* **Task/Need/Problem Solving Skills (identify the objects input, process and output) - Algorithm (logical steps flow-chat, pseudocode) – Implementation – Maintenance.**
* **PLAN LOGICAL the UML and ER diagram WORKFLOW (precondition/pre-requisites – now condition – post condition) model when BUILDING Test Automation Frameworks to give you a MAP to guild you as you build.**
* **A LOGICAL WORKFLOW is a sequence or series of operations or processes through which a task or piece of work goes through from the start (initialization) to completion, breaking each piece into (Analytical Skills) unit by completing each step for the task.**
* **PLANNING helps to breakdown a workflow**.
* Encoded and Decoded meanings.

**Syntax construction and rules.**

* **Program** – input - process – output - storage (variables, indexers, index, parameter, key/value pair concept).
* **Object** – state and behaviour.
* **It can be called Program, Type, Element, Object, Resource, Service, Entity, Record, NODE.**
* **It can be called property, index, field, attribute, input, key, column, parameter.**
* **Data – set of value (singular or plural)**
* **Data-Type - the data determines what type can hold that value/arguments.**
* **Grouping Related Required logical objects as attributes or properties or fields or columns or parameters or inputs or keys that REPRESENT an OBJECT.**
* **Operations, Mechanism, Actions, Process, Function (statements), Method, Implementation of HOW.**
* Defects and bug.
* Maven build-tool.
* HTML, CSS, JavaScript, XPATH, CSS – Selector, DOM, NODEE.
* Debugging Tools.
* Download COMPLETE REFERNCE BOOKS.
* **Study with textbooks, Videos, explain and apply to a specific problem**.
* Nth tier layer (phase) architecture. Loosely couple architecture.
* Presentation – Business Logic layer – Implementation Layer.
* Design Patterns – Creational, Structural and Behavioural (Flowchart, pseudocode), Data Structures – linear and non-linear.
* UML AND Entity Relationship diagram.
* **Problem Solving Skills – Algorithm – Implementation - maintenance.**
* **Analytical Skills and Soft skills.**
* Lambda.
* **Program -> Object -> Type -> Value and Reference Types.**
* Follow instructions, Rules, Constraints, Requirements, SPECIFICATIONS.
* Version Control Systems GIT and GITHUB, BITBUCKET, Software application programs.
* Java, JavaScript, C#, Python.
* Clues, ideas, guess the keywords (define it and look for object relationship).
* Object REPRESENTATION.
* Client Server architecture.
* Scripting Languages.
* Required relation properties that REPRESENT an object, e.g. Student.
* PUSH and always PULL mindset.
* **Annotations and Attributes (metadata) (A programmer gives a set of logical instructions to the compiler to process a program).**
* **A programmer instructs the COMPILER to execute the source code in a sequentially orderly manner, HE direct the execution workflow.**
* **Unit Testing Frameworks.**
* Serialization and Deserialization.
* Apache poi, yaml
* TestNG, Specflow, Cucumber, Nunit, Pytest.
* Features or Functionality on a program, API, Software application.
* IDE – **Compiler.**
* UNDERSTAND **Compile time, Build-time, Runtime Objects or Types.**
* Explicit and Implicit meaning.
* **WHAT and HOW.**
* **Local and Remote machines.**
* **Client Server architecture.**
* **There are many approaches or ways to apply a solution to a problem.**

**Software Testing**

* Specifications, Requirements (needs)
* Defect
* Types and Kinds of testing.
* Testing books.
* Software testing manual and automation tools.
* Verification, Validation and Assert.
* Think Positive and Negative testing, Think right, left, up and down.

**Object Oriented Programming Languages**

* Java – access (getters) and mutators (setters).
* C#
* Python
* JavaScript

**Functional Programming Styles VS Object-Oriented Programming Style.**

**Study programming and BUILD, BUILD, BUILD**

* Understand the specification/requirements in Problem Solving statement (Skill - identify the **object** (nouns) (apply **Object Oriented Principles**) and their state and behaviour, input process output.
* Create **loosely (creational and structural)** coupled (**Behavioural**) layered architecture (implementation - apply OOP principles), then Algorithm step by step workflow.
* Presentation (WHAT layer) – Business Logic Layer (implementation of HOW) – Database layer.
* **Design Pattern**: Creational – Structural – Behavioural.
* **Study other developers code to find out how they applied logical solutions to solve a problem programmatically**.
* **Workflow, statements**.
* **You need to know the RULE and CONSTRAINTS of the problem, know what is mandatory and optional.**
* **THINK LOGICAL.**
* **Study technical prep interview contents (videos and texts, blogs).**
* **Mathematic and English.**
* **Logical Sequential workflow.**
* **Cucumber, Specflow, Behave.**

**What and How Function / Method is constructed**

* A function performs a specific functionality or feature.
* One data is used more than once, then it must be created inside a function and reused.
* Once a step in a function has sub-steps to get an output, create a separate function and call it into that main function.
* Function names are action **verbs** (action words).
* Function contains a block of statements.
* Step by step execution of statements.
* Data type – e.g (Boolean condition) followed by an identifier, and it must be named according to reason or purpose.
* The data determines the Type used to hold the data.

condition is the data

Boolean is the Type

* Expression – lambda
* Time and Space complexity

=============OOP PRINCIPLES=============

Polymorphism

Inheritance

Abstraction.

Encapsulation

Other principles- Aggregation, Dependency, Composition

========================================

**HOW to learn a Software or Program or Plugin/Extensions Application.**

* What is the software and what purpose or why was it created, what problem does it solve.
* How to use the software, check the API Documentation.
* What programming language is it built on.
* **What and how to construct the syntax rules and vocabulary in** the software.
* Learn the **inputs (inputs are types, the data determines what kind of data the input can hold, e. string type holds string value or data)**, **processes (process data held in the inputs)**, **output (process returns a value or void), and storage (store return or assigned data)** of the software.
* Learn the **Commands**, **features,** or **functionalities**.
* What are each input and processes and what and how are they used for.
* What other software does it integrates with.
* How to use the software features or functionalities.
* When to apply the software features.
* Why do need to use the software features.
* Some Objects are Built-time objects, Compile-time objects, Run-time objects, they instantiate data before or after execution.
* What kind of data does the software hold. E.g. voices data, video data, text and many more.
* Is the **software/Program** application a CLIENT or SERVER.
* Is it located on the LOCAL or REMOTE.
* Is the software program application a front-end, middleware, backend-end tool?
* Execution – Build-time, Compile-time, Run-time

**JavaScript (Program) Programming Language - Object Oriented Principles**

* How are object-oriented principles implemented in JavaScript?
* Syntax of OOP in JavaScript.
* JavaScript’s projects (JavaScript OOP)
* Scripting languages
* Webdriverio, JSON, Nodejs, Mocha, Cypress.
* JSONObject.
* After learning the programming language syntax and vocabularies, then start building projects for that program.
* **Start Building JavaScript projects.**
* **Study and Build with the software libraries (cypress) built with or on top of JavaScript programs**.
* Interpreter programming Languages.

**=========================================**

**When you want to use a software program / object.**

* **EVERYTHING IS AN OBJECT FIRST BEFORE ANYTHNING.**
* **All Objects / Program at least has PROPERTIES (Inputs) and Functions (processes).**
* **First what kind or type of OBJECT is this? – (Value Type (Primitive Type) or Reference type, Attribute type, Constructor type, File type, Class type, Boolean type, Function type, and many more).**
* **What does it do? What problem can it solve? What is it name? what software libraries can I find it?**
* **Why do I want to or need it? Is there another option than this one?**
* **How is the program created or instantiated?**
* **How does it syntax and vocabularies constructed to form a semantic statement.**
* **What are it inputs - state and processes – behaviours, output, storage.**
* **ObjectType – ObjectVariable = new ObjectValue.**
* **ReferenceType – ReferenceVariable = new ReferenceValue**
* **PrimitiveType – PrimitiveVariable = PrimitiveValue**
* Define WHAT TYPE OF OBJECT IS THIS? primitive or Reference type.

**ALL SOFTWARE PROGRAMS ARE OBJECTS**

**===========================================================**

**PROGRAMMER – ENGINE – COMPUTER OS.**

**ALL OBJECT/PROGRAM HAS EITHER A PROPERTY OR BOTH PROPERTIES (INPUT/STATE) AND FUNCTIONS (PROCESS/BEHAVIOUR). E.g. pdf file has a property when you right click it.**

**An Object has a role or a purpose why it was created and what it is used for. E.g FeatureContext object in Specflow software program is used for value/data storage.**

**STUDY how to Access an OBJECT properties and Functions to retrieve get data/value. E.g Object.property and Object.Method. String.IsEmptyOrNull();**

**=========================================**

**File object type –** the **file type** holds object **data type** that is converted into **machine code**. the File itself is a type and it holds it own data that is required to be held inside the file. All Files have a File-type that use a specified **file extension**. When the engine during execution run-time (compiler, interpreter) parses the file-type, it pares the data types held inside the file, either properties or functions are parsed. It creates an Object and assign reference of the File-Object and stored it in a **reference variable** after converted to machine code during run-time. e.g helpers.**testData.feature – Object or File Type,** com.mysql.jdbc.**Driver** – the Driver is the **program/object type**, which is similar to a File-Type.

IN SOFTWARE TESTING, in a testcase, WE GATHER THE object INPUTS, PROCESS AND VERIFY AND VALIDATE EXPECTED OUTCOME USING TOOLS LIKE, Software Program (Specflow, Cucumber, java, c#, TestNG).

**Note: So, when you are given a problem statement in English language, you must think of how to translate that English language words into a programming language word. How will programming language rewrite (model) that same English statement in its own language.**

**Note: Each line of code you write, has a purpose it wants to achieve and what problem it want to solve on each line of code.**

**TIPS TO CREATE A FUNCTION**

* The action verb for the function signature name
* Step by step statements are algorithms.
* A FUNCTION accomplishes a specific TASK.