Test Automation Framework Design Principles

Thank you for coming to this webinar

1100+ Registered

Through multiple channels

360 Cities

Chennai, the highest and followed by Bengaluru

14 Countries

Mostly from India and followed by United States

4 Continents

Asia, United States, Australia and Europe

TestLeaf

Always Ahead

In last 3 weeks, we got more than 170 QA people lost their jobs (based on our data only in Chennai). May be worst yet to happen !

Out of which, 31 of them are sole earners in their family !!

TestLeaf
Always Ahead

Can we make an effort to help those job losers to get back to work?

TestLeaf
Always Ahead

You can assist !!

Share the job opportunities you know in any medium

Do not switch job for next few months unless it is so needed, rather give it to someone else !!

Help them to upskill, motivate them in whatever form you can !!

TestLeaf
Always Ahead

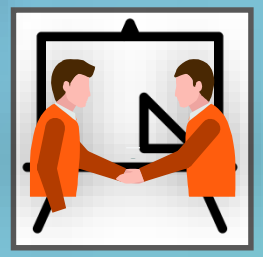
About TestLeaf & Myself

TestLeaf

Always Ahead

TestLeaf founded in 2009 as ed tech company and now, we spread across markets through our training, consulting, staffing and products.

Team Size: 44



Training Services

TestLeaf specializes on personalized training services for corporates and public. Our deeply skilled mentors, creative course content and delivery model is so unique.

Staffing & Consulting Services

If you are looking for manual QA / test automation staff for a short period of time and immediate onboarding, our temporary staffing is the most flexible form of employment mediation.

Products

Tuna, the codeless test automation product that can automate your tests incredibly fast on web and mobile applications. Integration with CI and build tools is a big plus.

Babu

Coach & Mentor

20 years experience

60000+ Scripts

1 Million+ Tests

Let’s Get Really Started

Test Automation Framework Design Principles

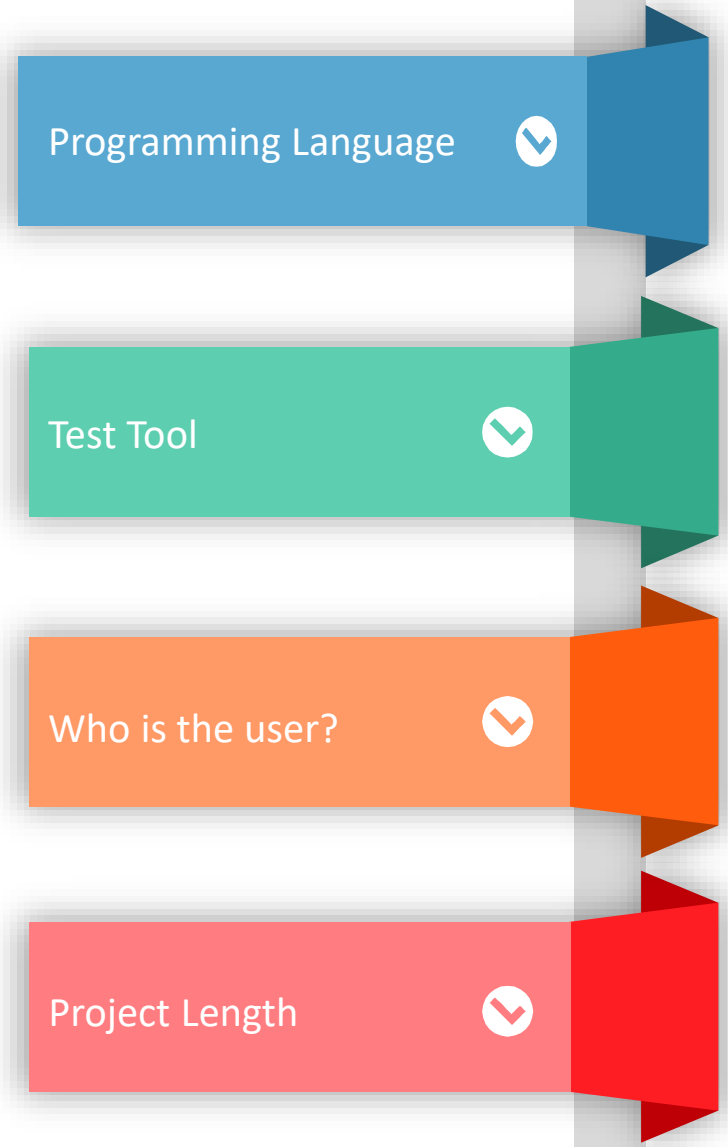
TestLeaf
Always Ahead

~ Build a House

Step 1) Identify the influencers

TestLeaf
Always Ahead

Influencers that you should know

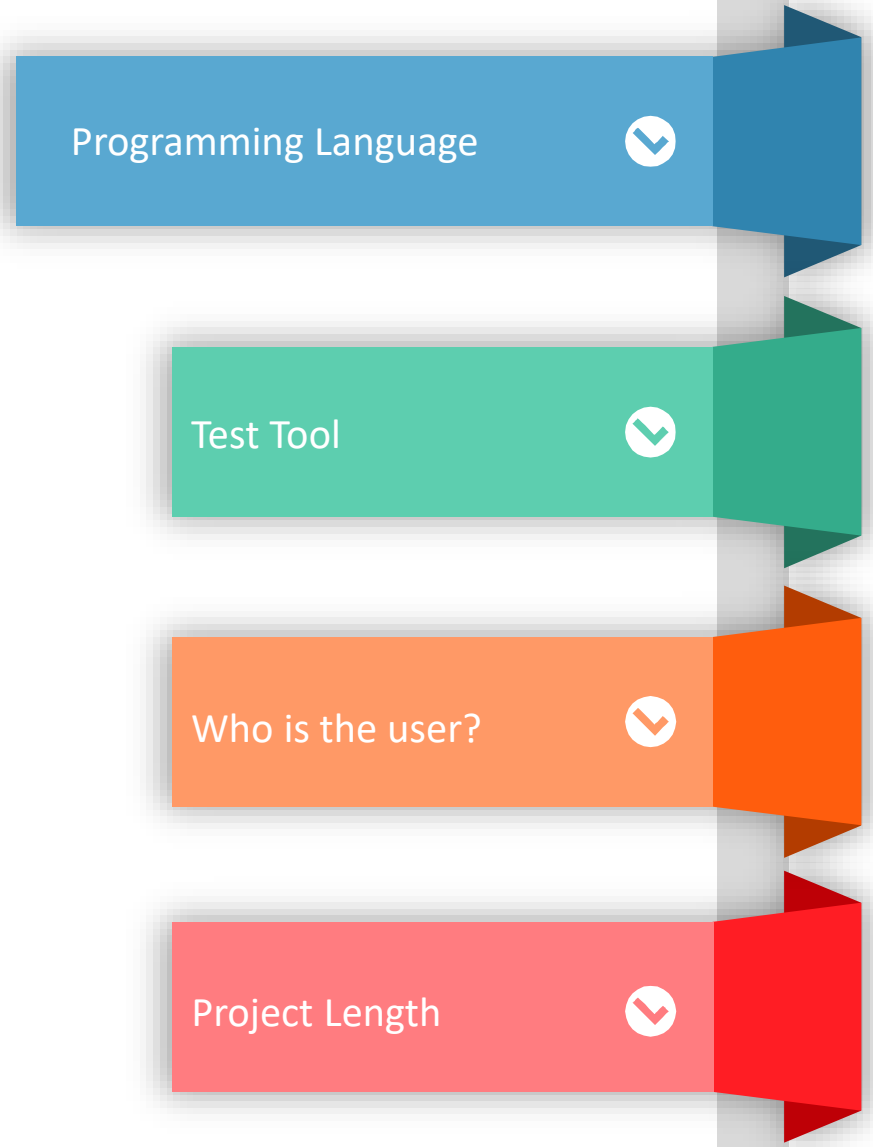


Programming Language

Build test automation frameworks in the same language of development

Why?

In agile world, you can collaborate more with your developers. Shift Left



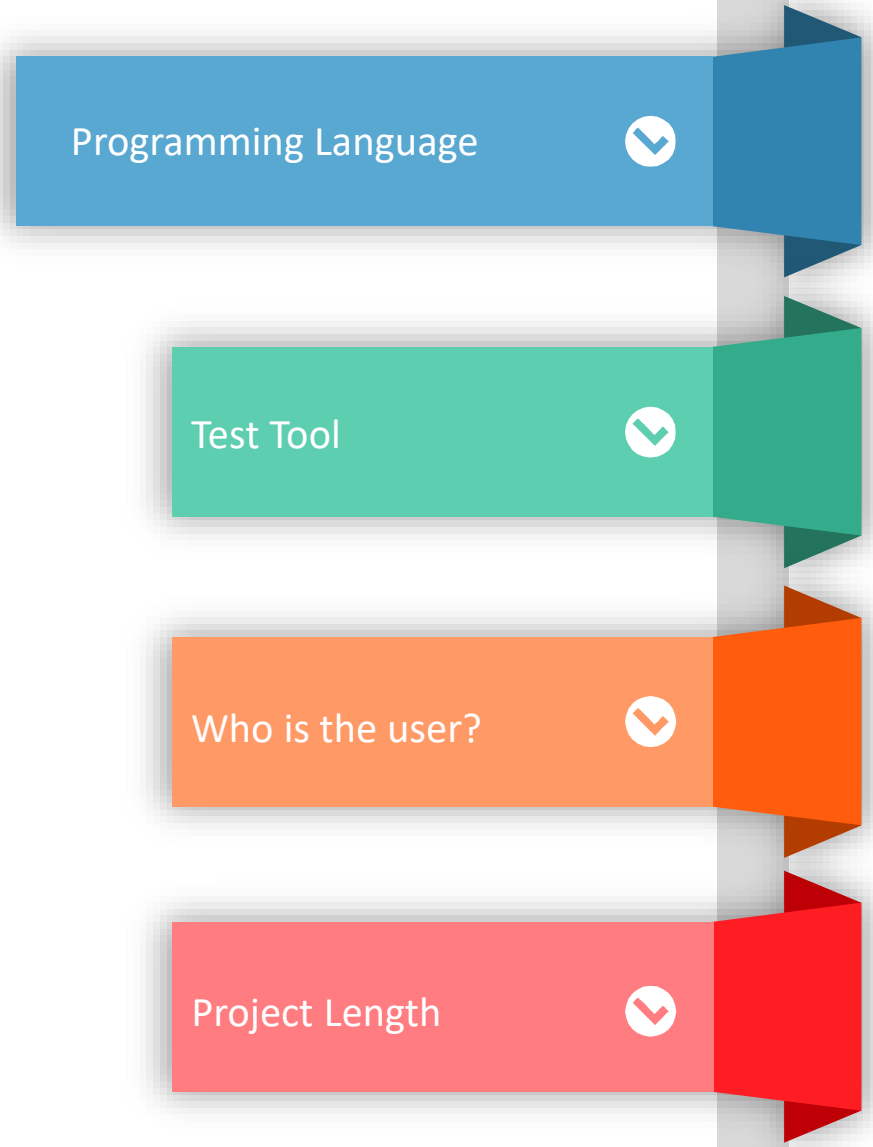
Programming Language

Build test automation frameworks in the same language of development

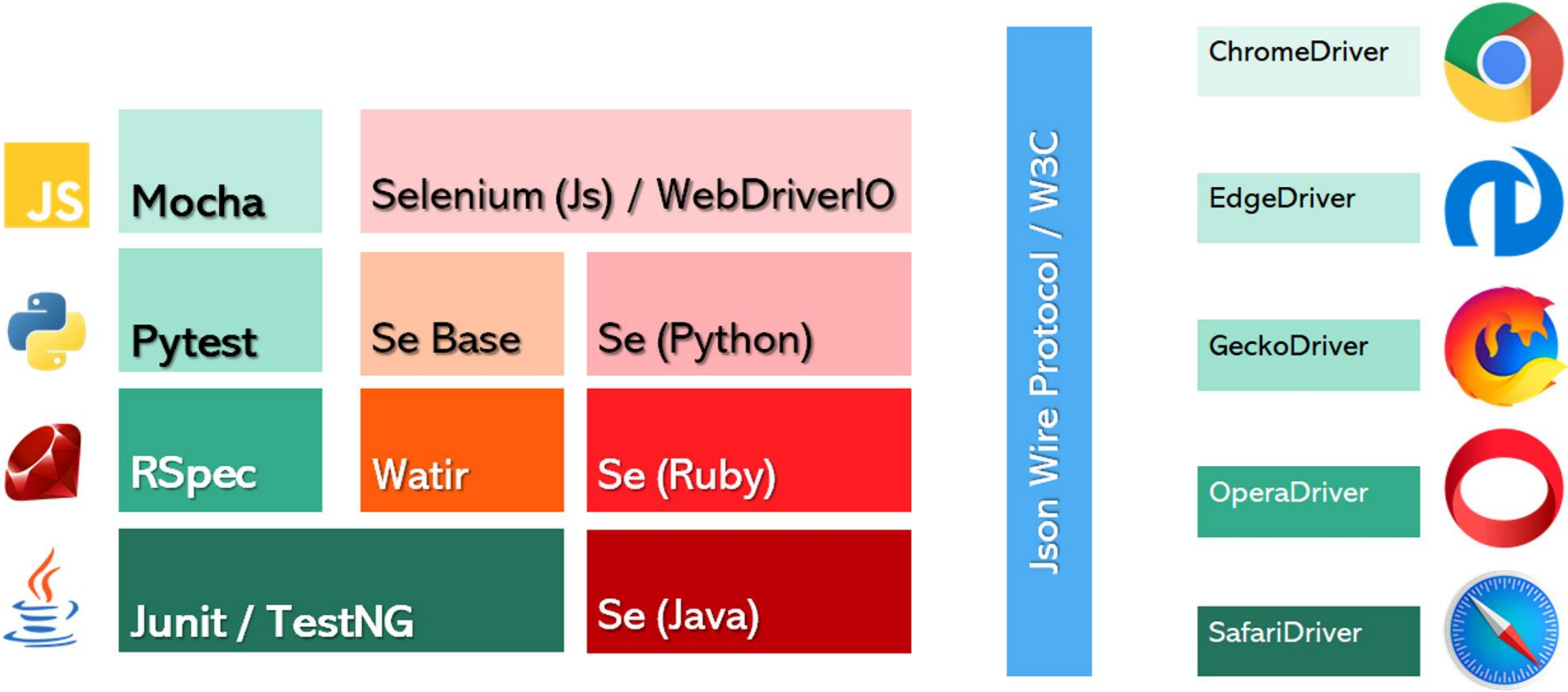
Front end or back end language?

Depends, on what you do !!

If you are testing UI, then front-end language and if you testing the API, then the backend.

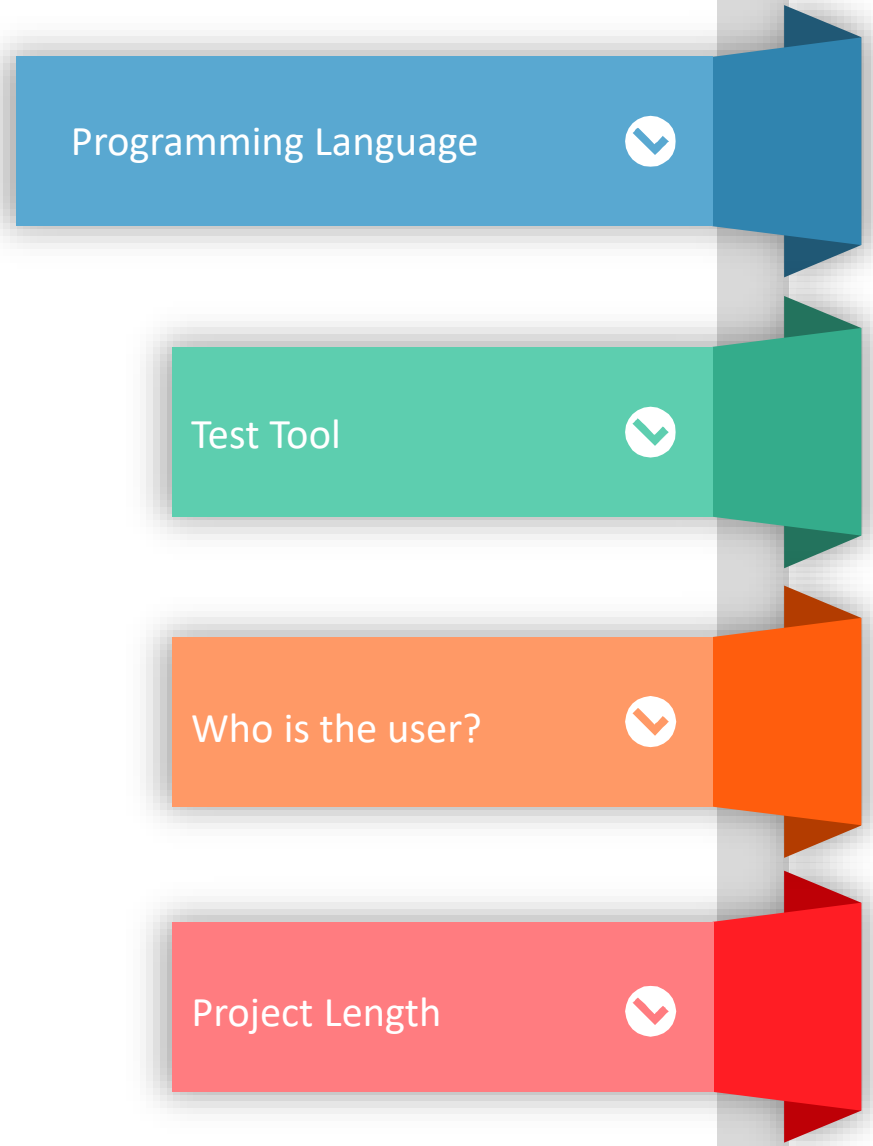


Programming Language



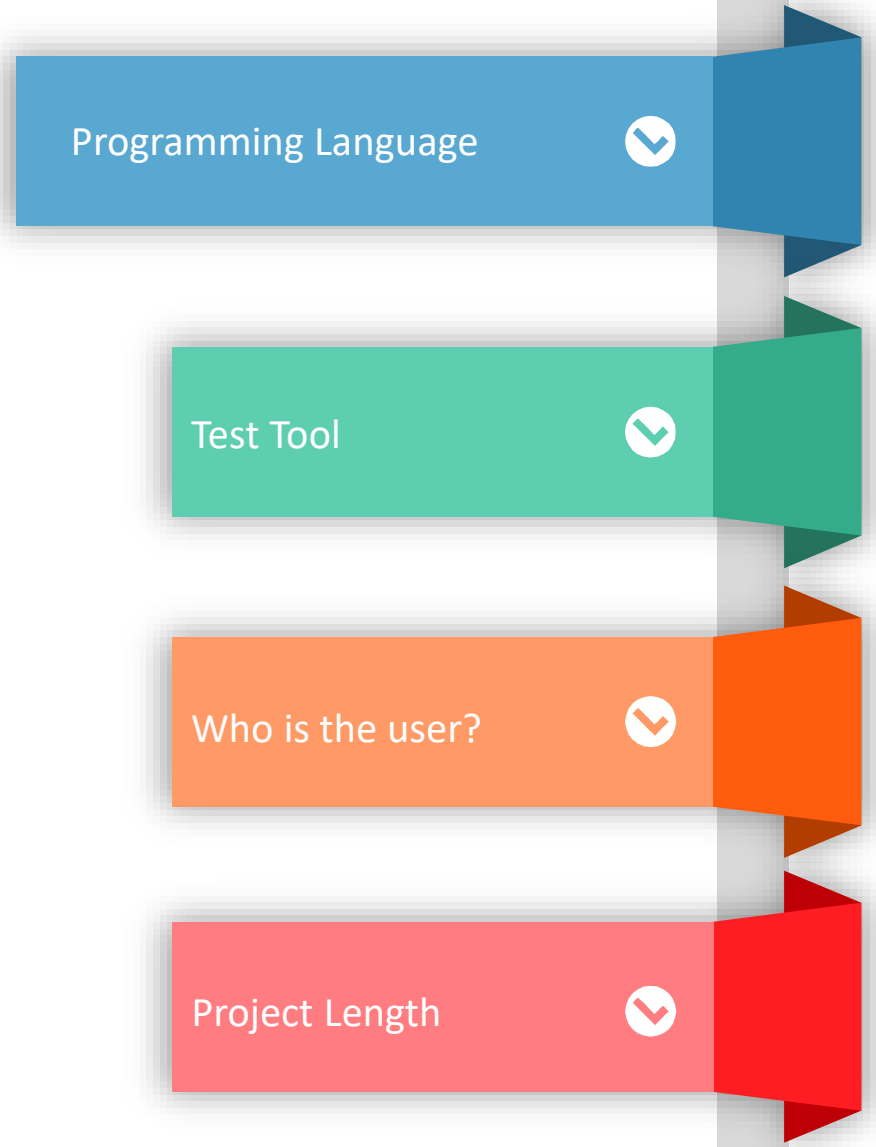
Question to ask yourself !!

Should I know ReactJs to automate ReactJs application?



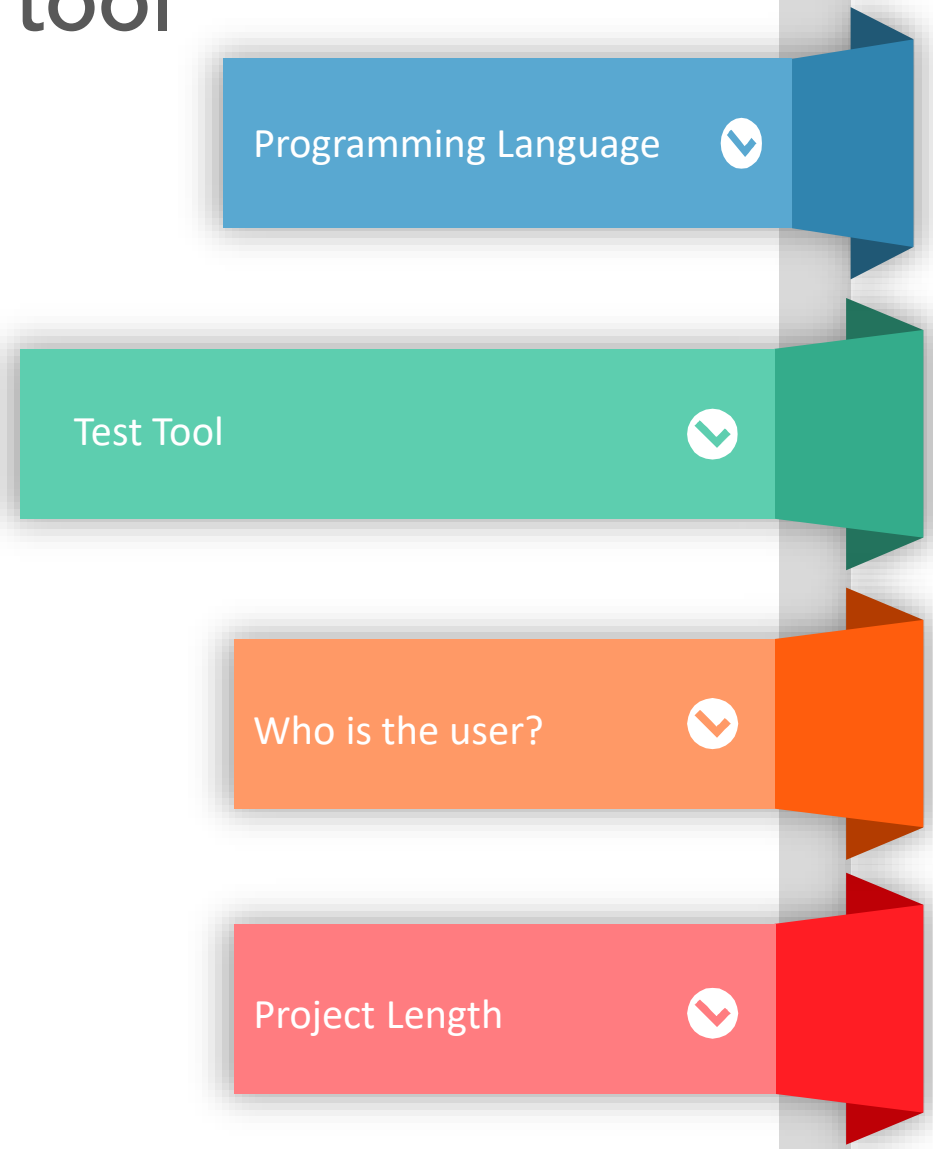
Take Away 1

Preferably, design the framework in the same programming language as developers.



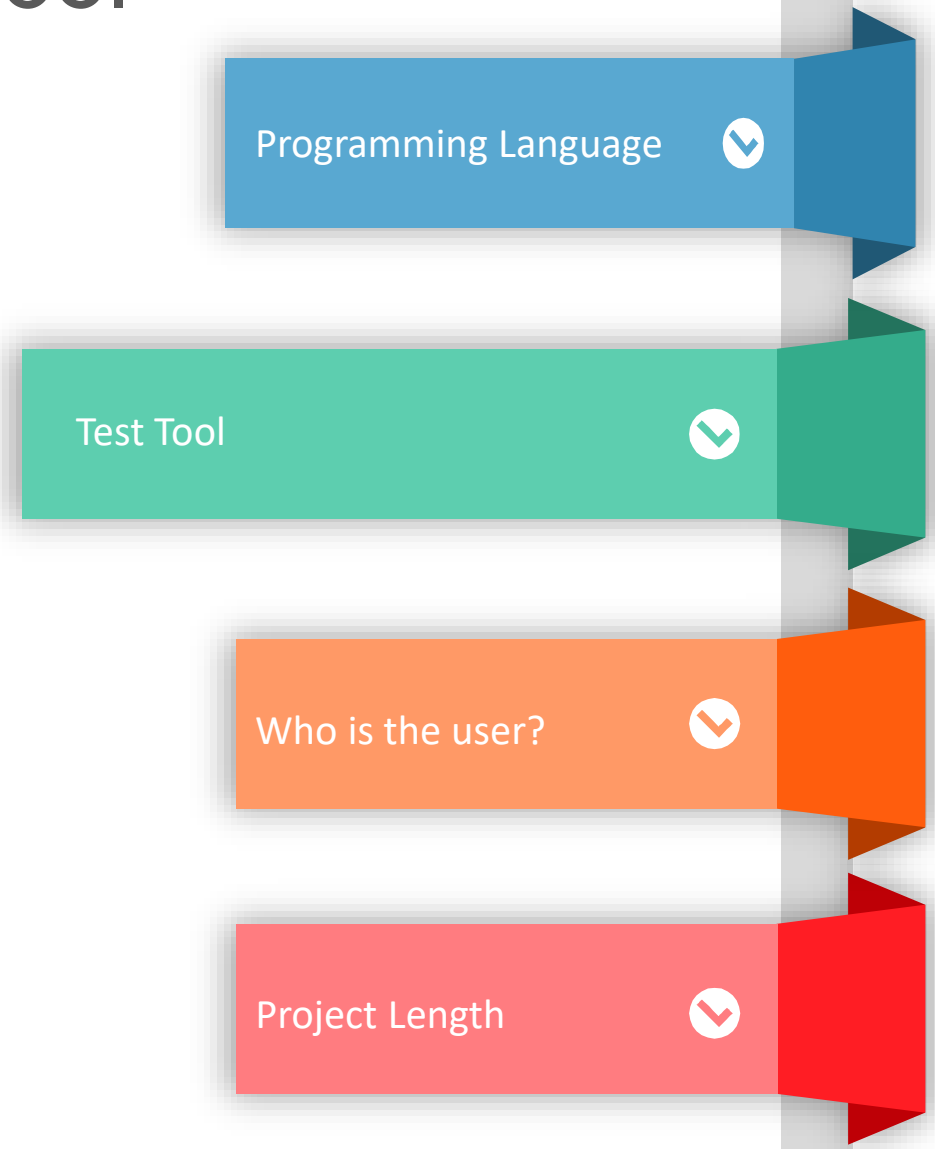
Framework design as abstract of the tool

Frameworks are Convenient abstractions designed to remove as much boilerplate as possible from tool test setup and execution.



Framework design as abstract of the tool

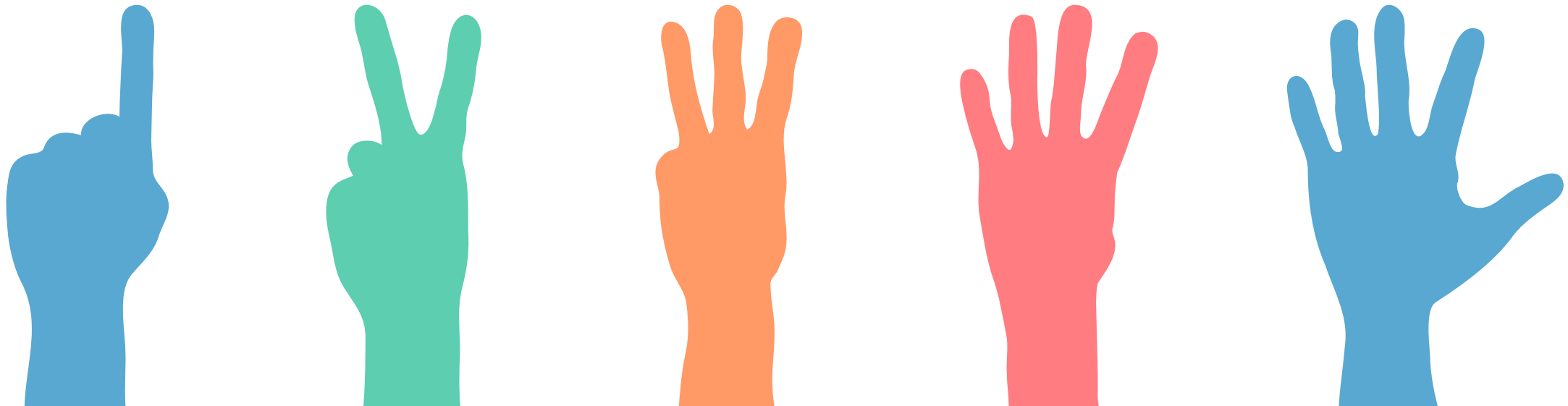
So you (test engineers) can focus primarily on test logic.



Which tool to use?

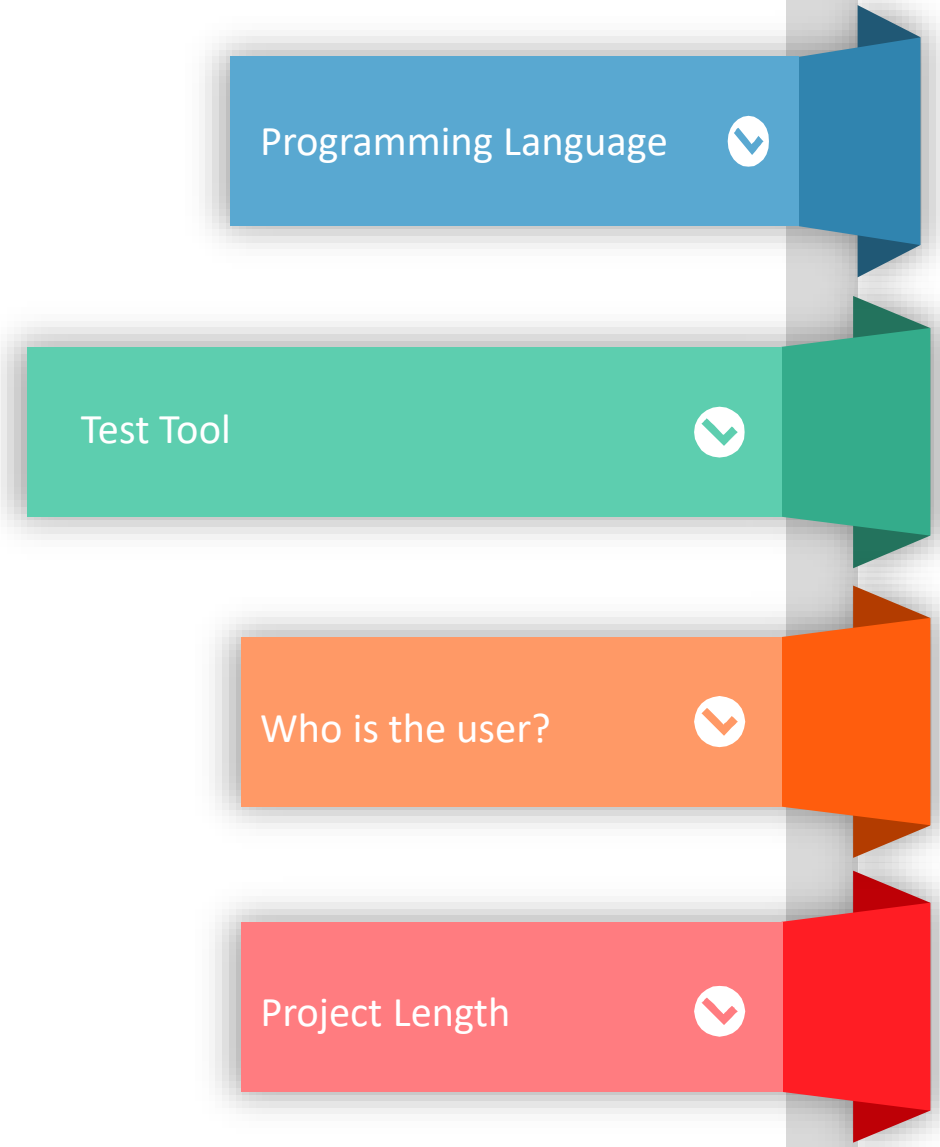
TestLeaf
Always Ahead

1. Need & Purpose
2. Tenure & Features
3. Commits & Support
4. Language & Platform
5. Power of Integration



Question to ask yourself !!

Should I not have one framework abstracting multiple tools?

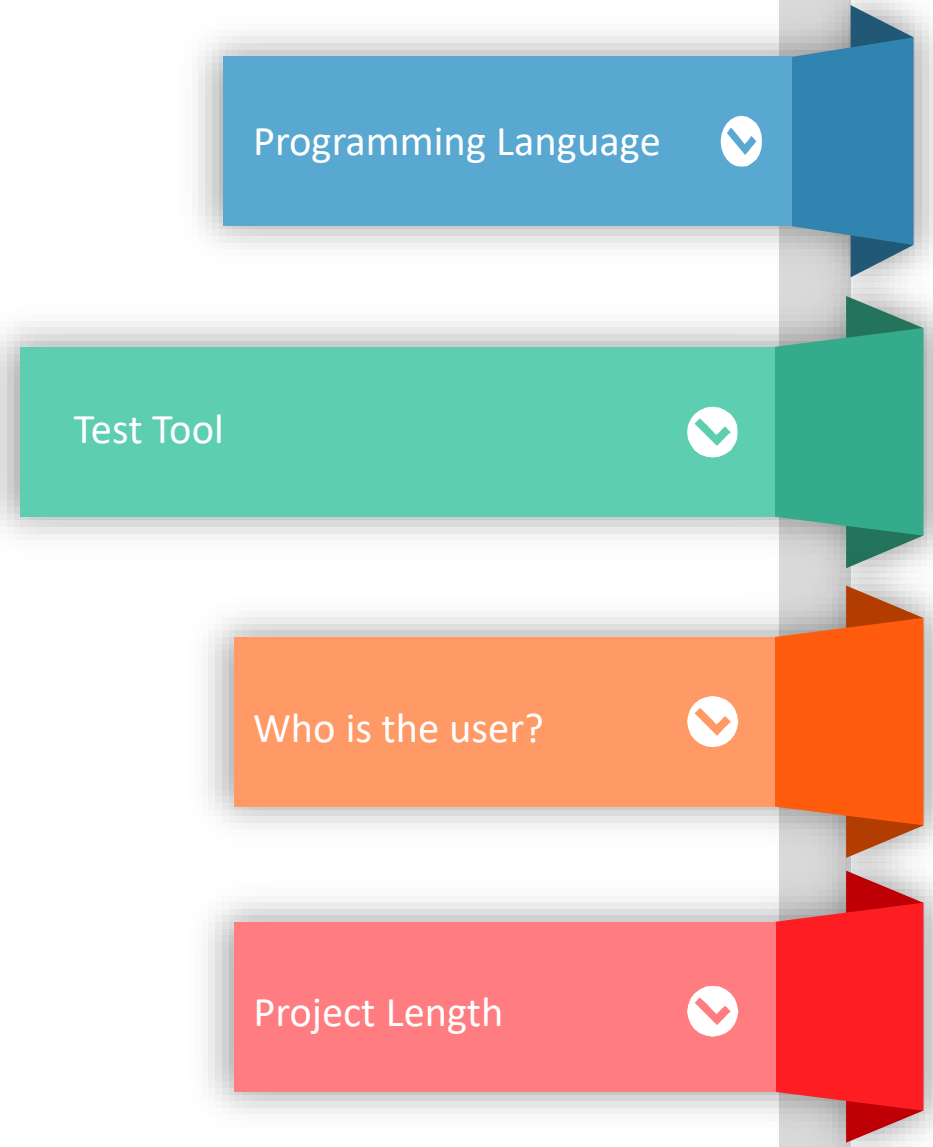


Question to ask yourself !!

Should I not have one framework abstracting multiple tools?

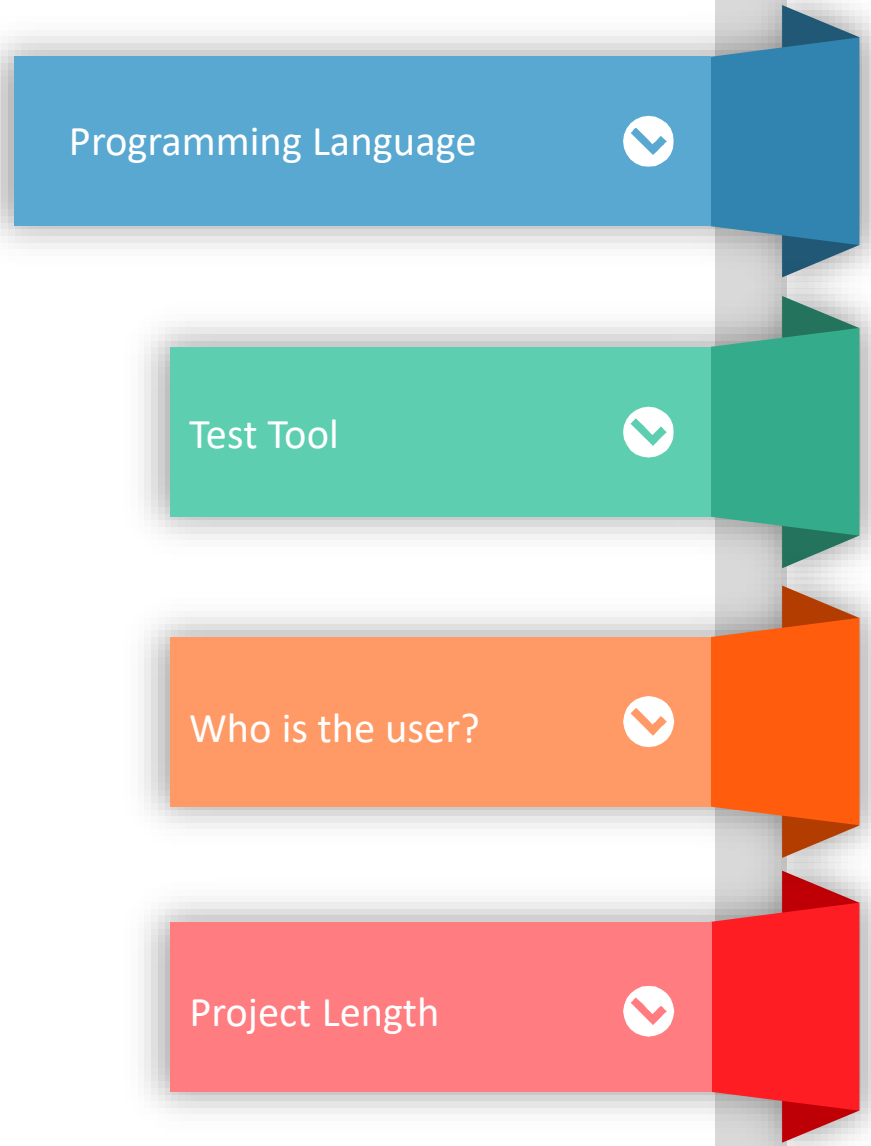
S.O.L.I.D is an acronym for the **first five object-oriented design(OOD)**** principles** by Robert C. Martin

- **S** - Single - responsibility principle



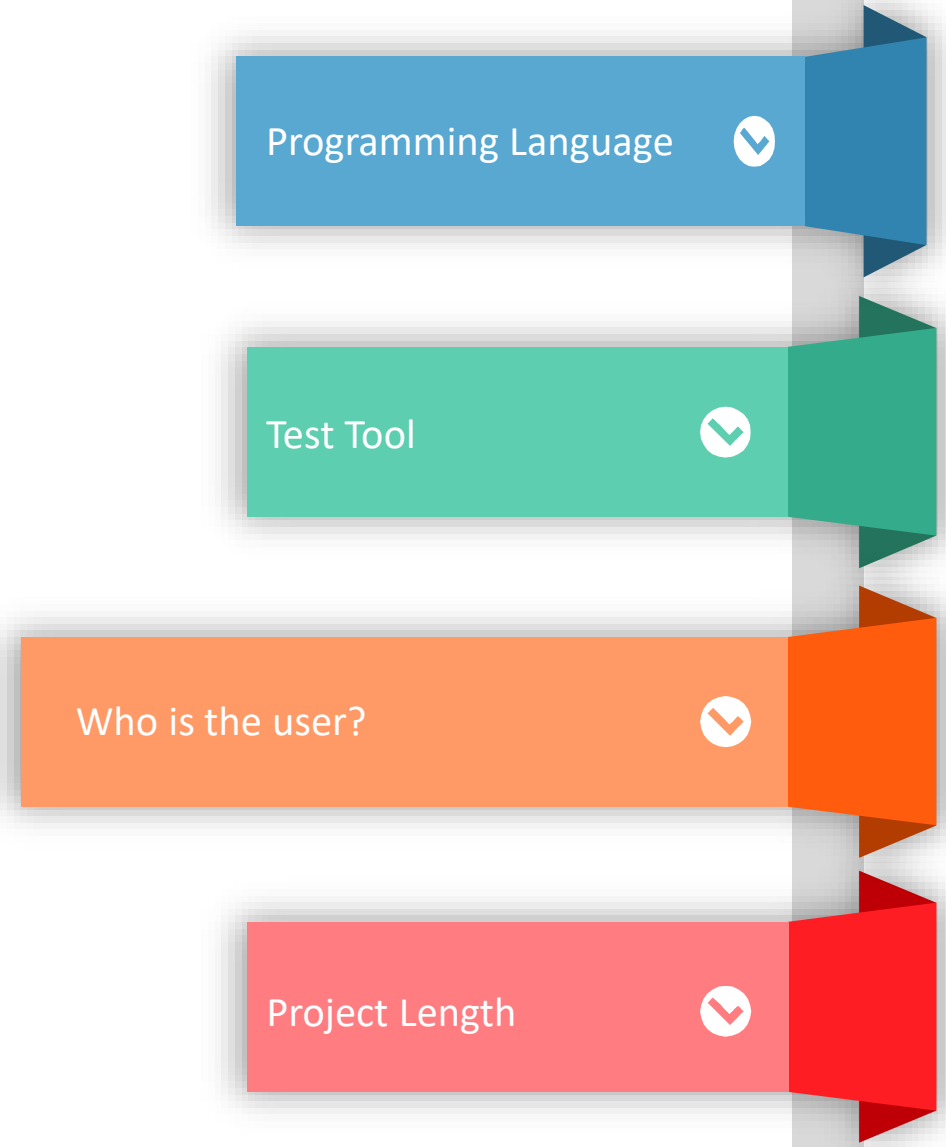
Take Away 2

Design the framework as a convenient abstractions on top of the single tool.



Who is the consumer of framework?

Find the daily users of the framework and that can change the perspective of your design thinking.



Who is the consumer of framework?

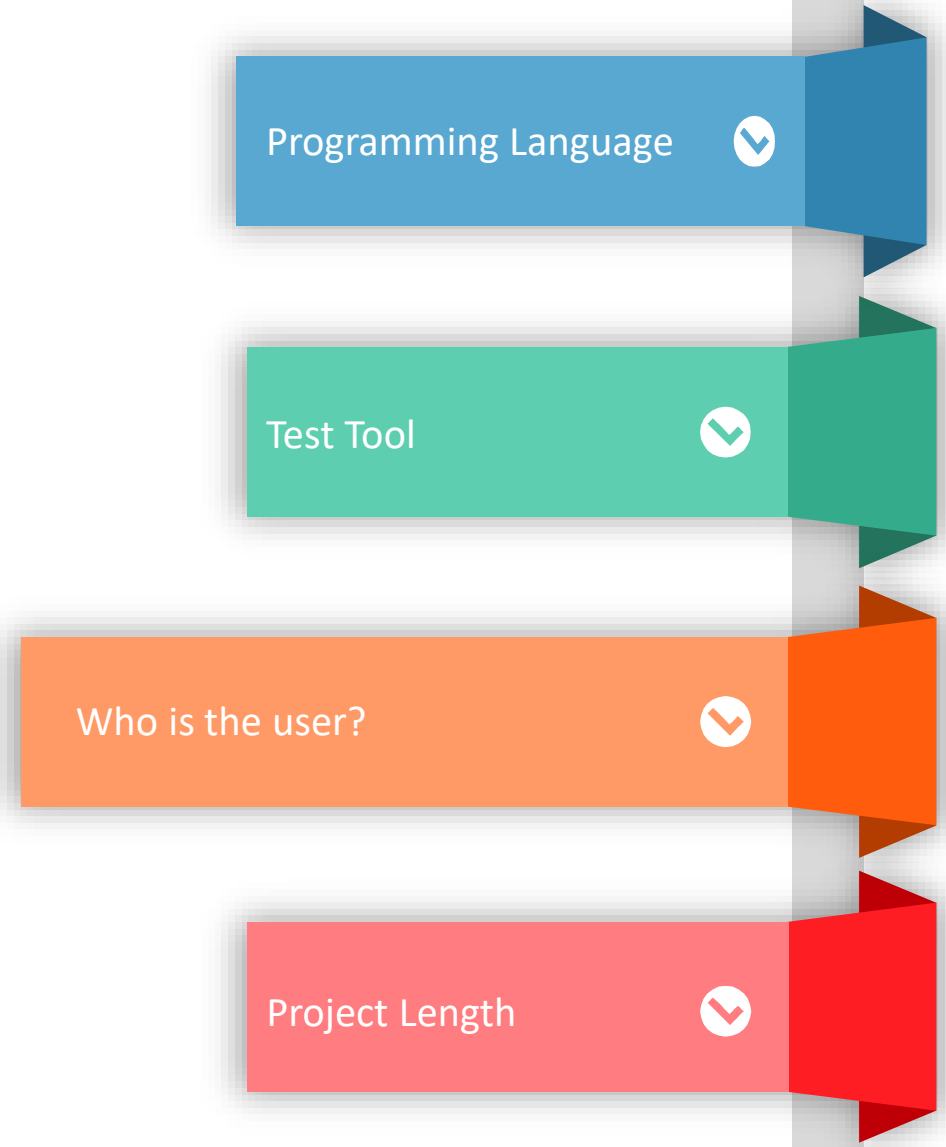
Internal vs External

Reports : Simple vs Elegant

Browser : One vs Many (Versions / OS)

Data : Single vs Multiple Source

Perfect vs Working

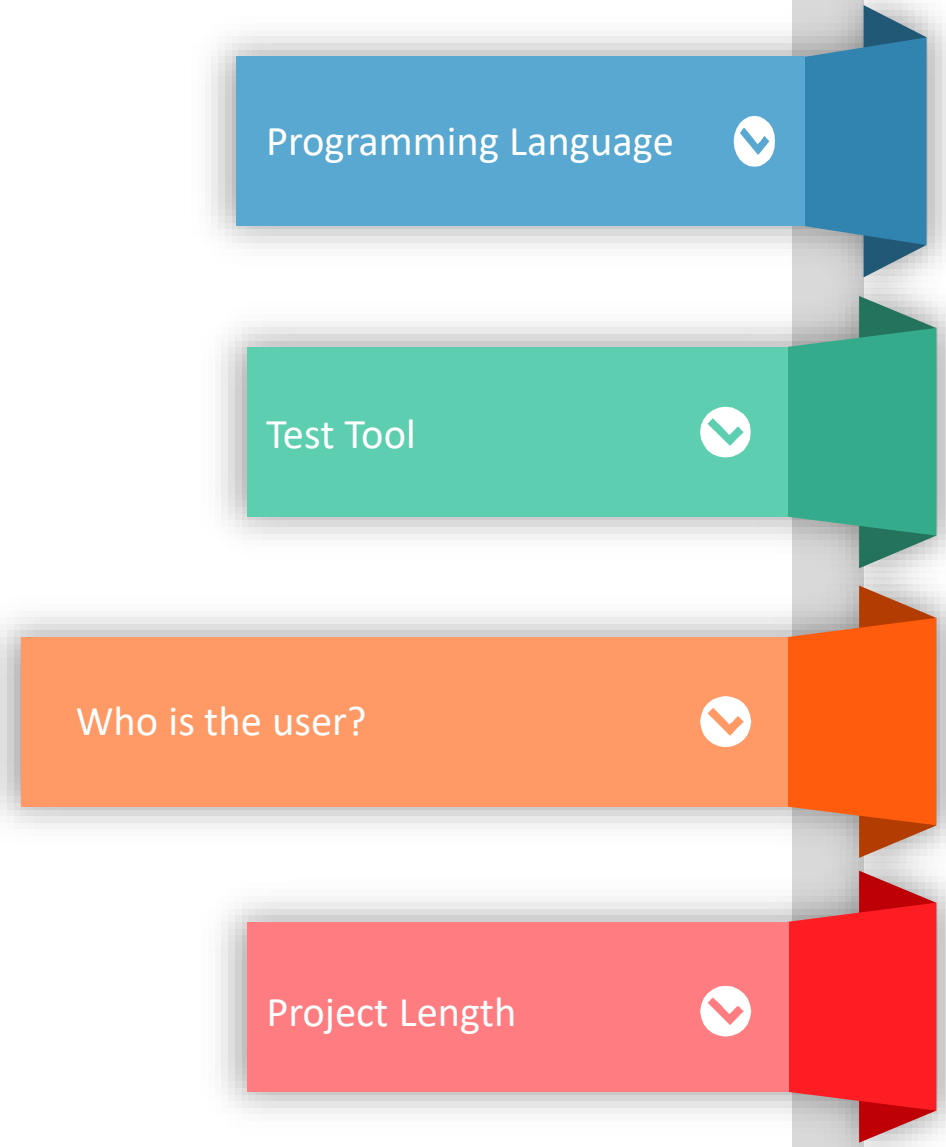


Who is the consumer of framework?

Business vs Technical

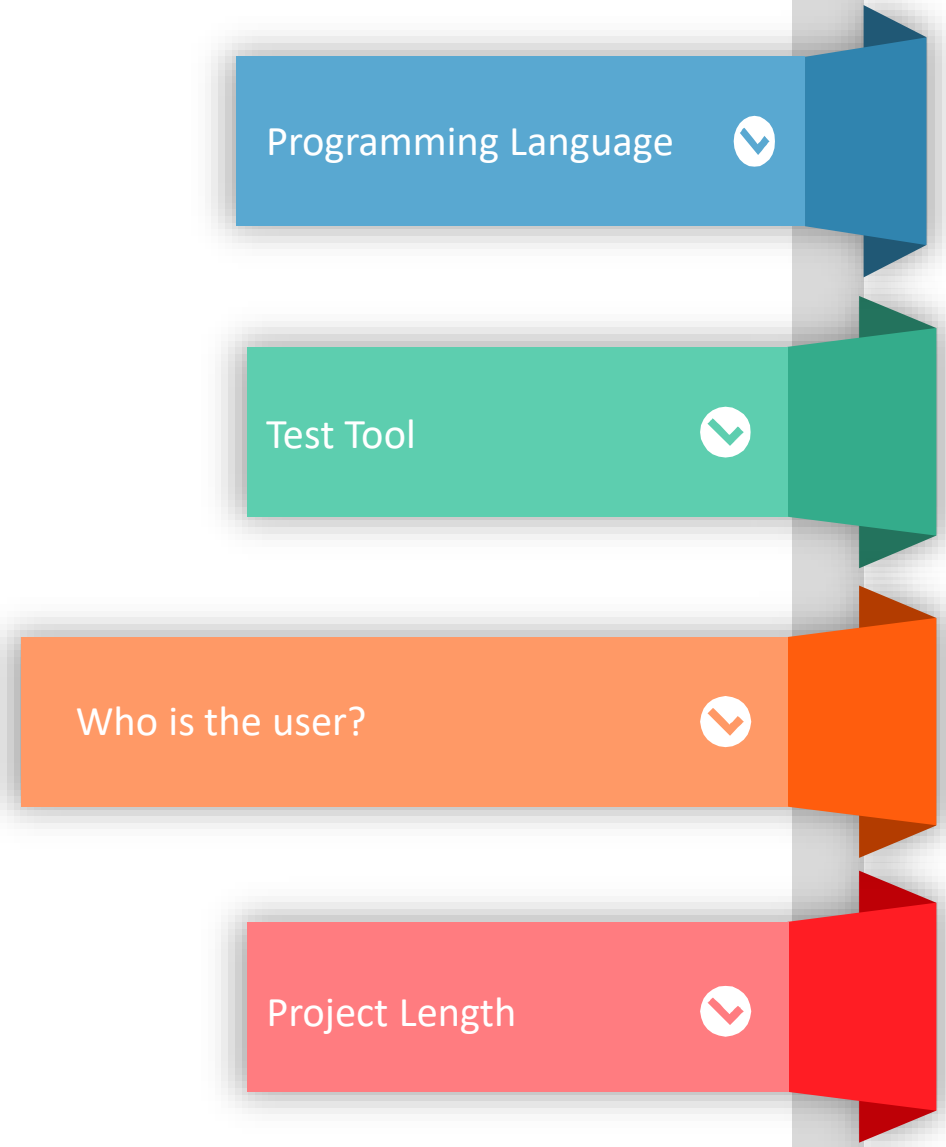
Properties vs Classes

Excel vs JSON



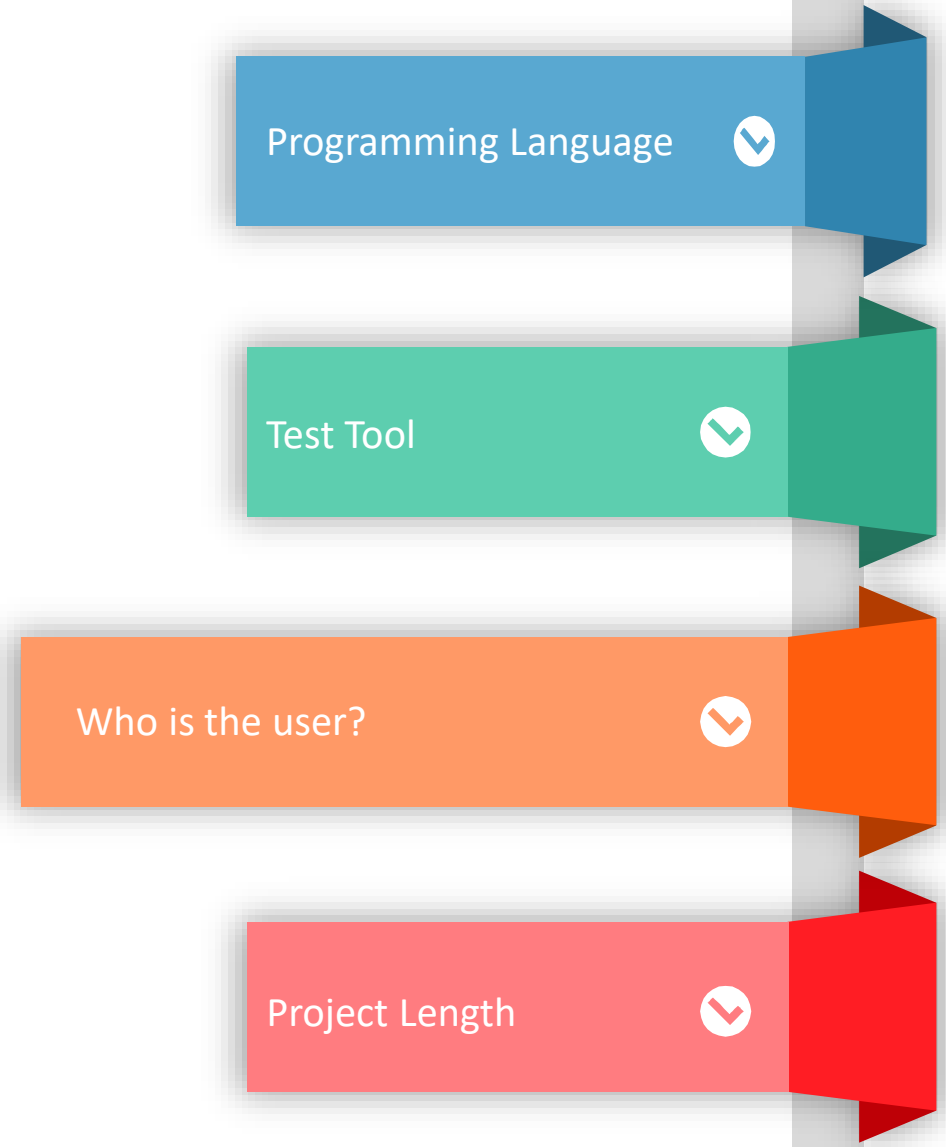
Who is the consumer of framework?

If you are unsure, do Discovery workshop to know what is really needed to be built !!



Take Away 3

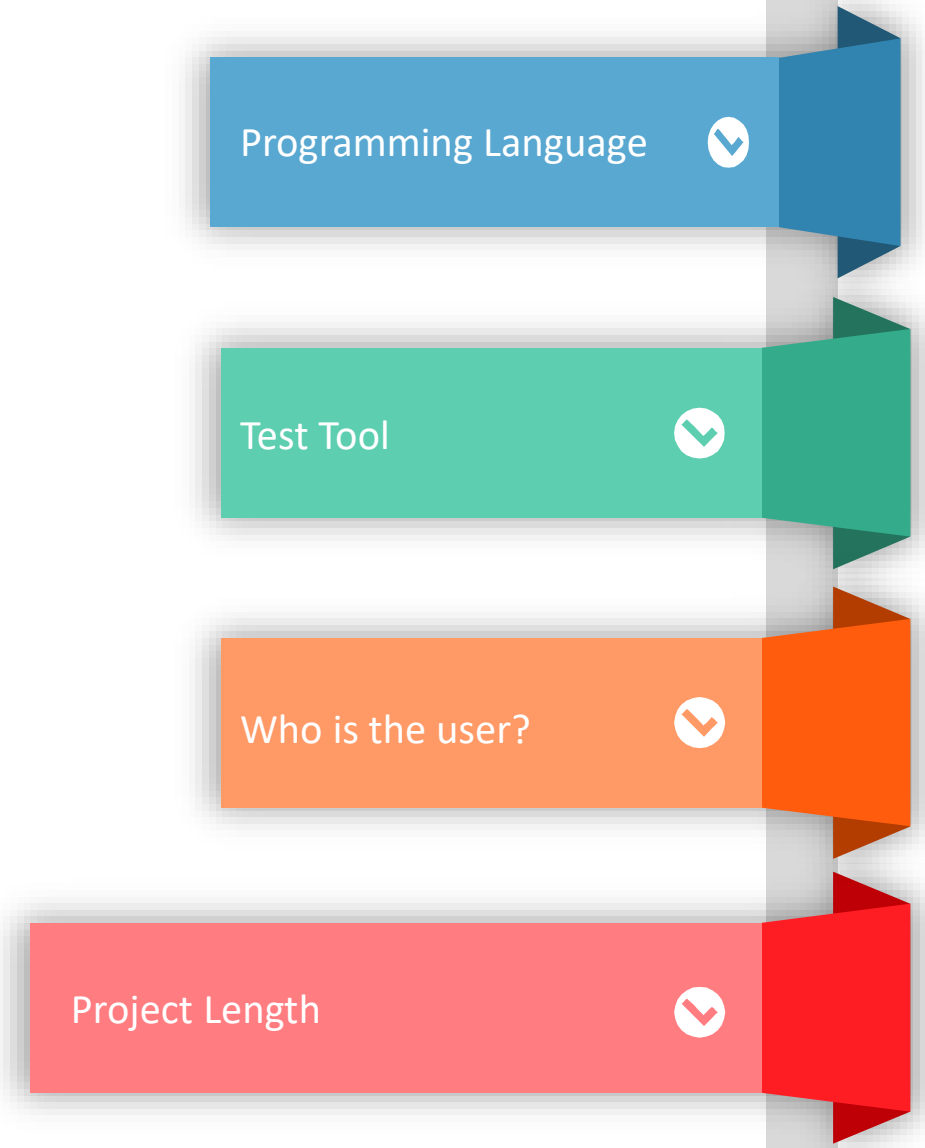
Build the framework for the consumer (based on their feedback) and not something you know or comfortable with.



Tenure vs Framework Features

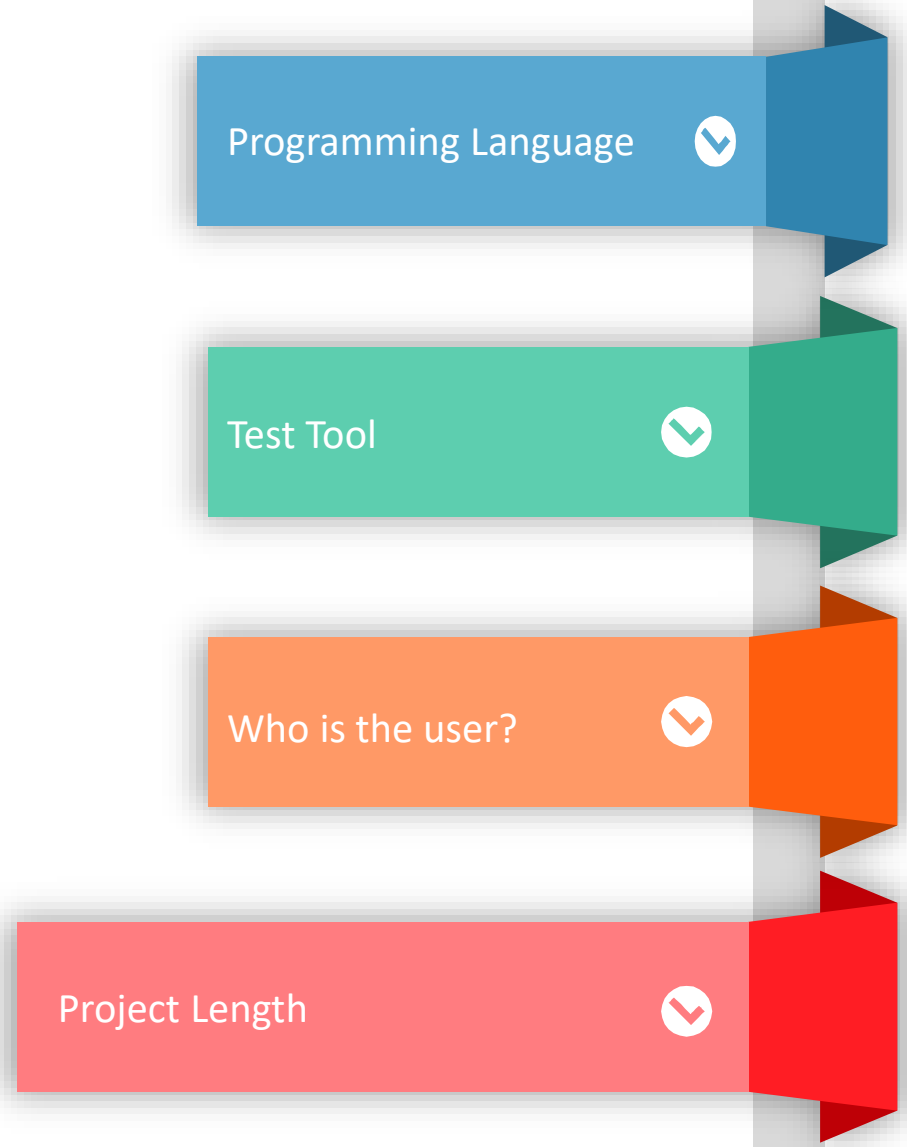
The higher the tenure, the framework standard should be higher:

- features
- usability
- documentation



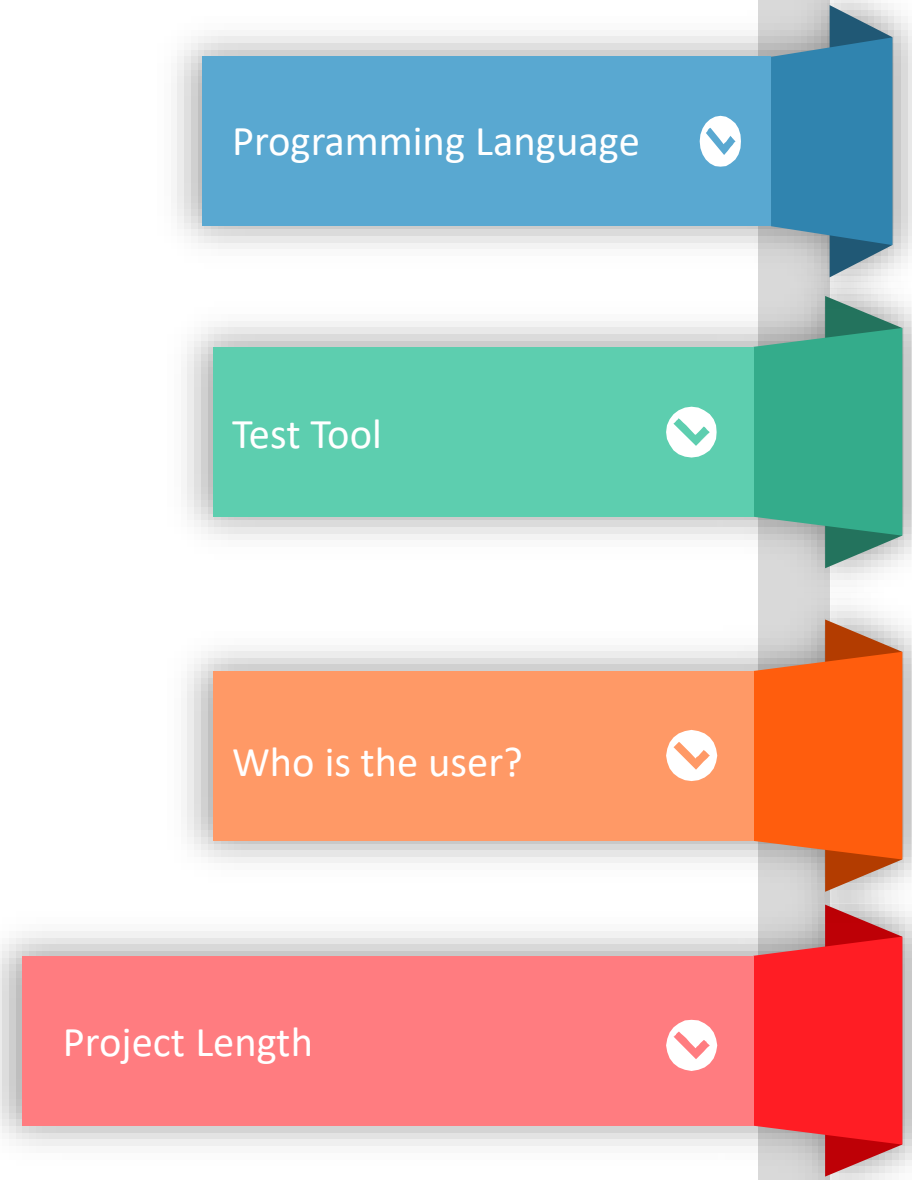
Tenure vs Framework Features

Start as Minimalist



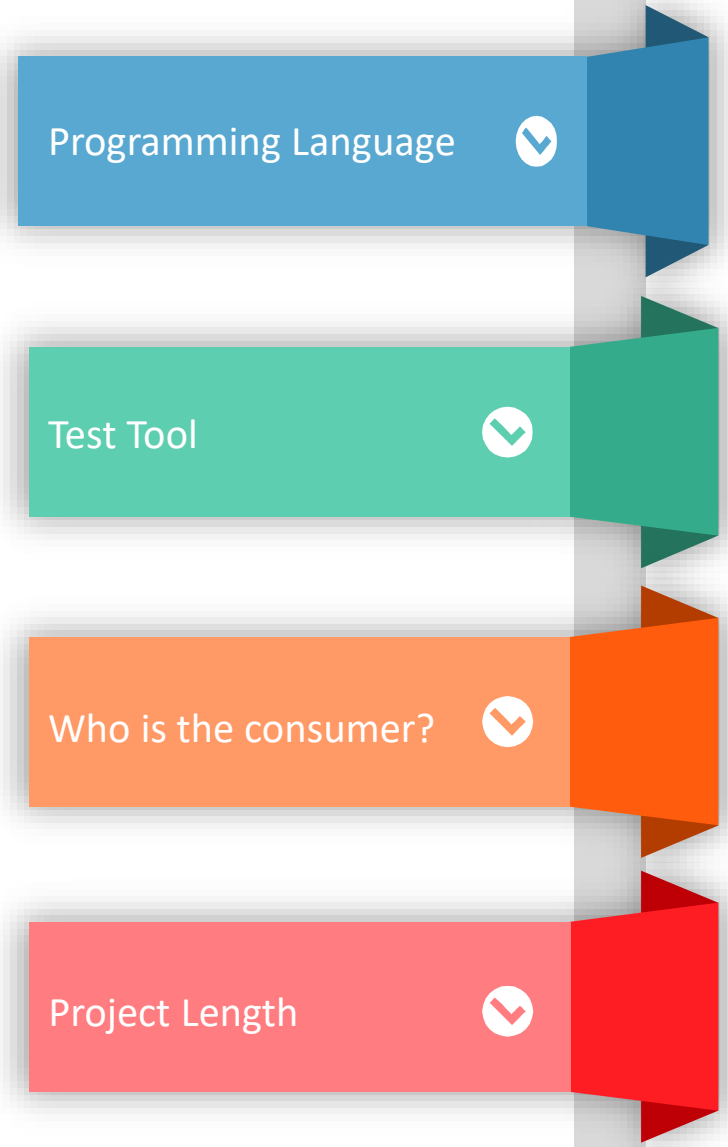
Take Away 4

Build and expand the framework as time progresses. Do not build everything in Sprint Zero.



Influencers Summary

Remember that the influencers decides what, how and when design?



When you like to buy car, which one you would choose?

01

02

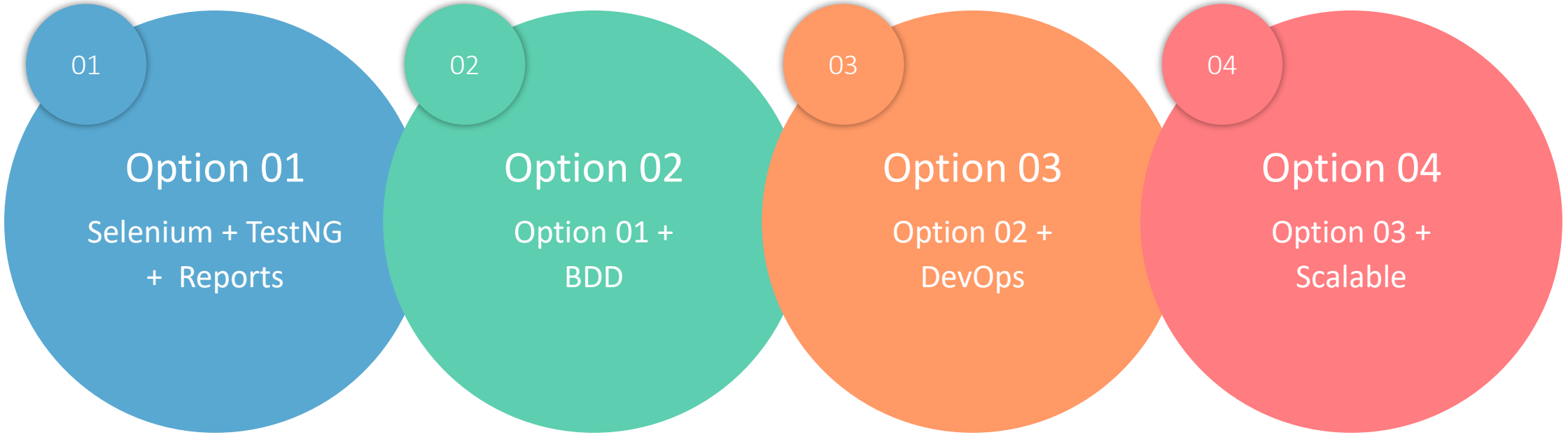
03

04

TestLeaf
Always Ahead

Take Away 5

Design Multiple frameworks and Not One !!

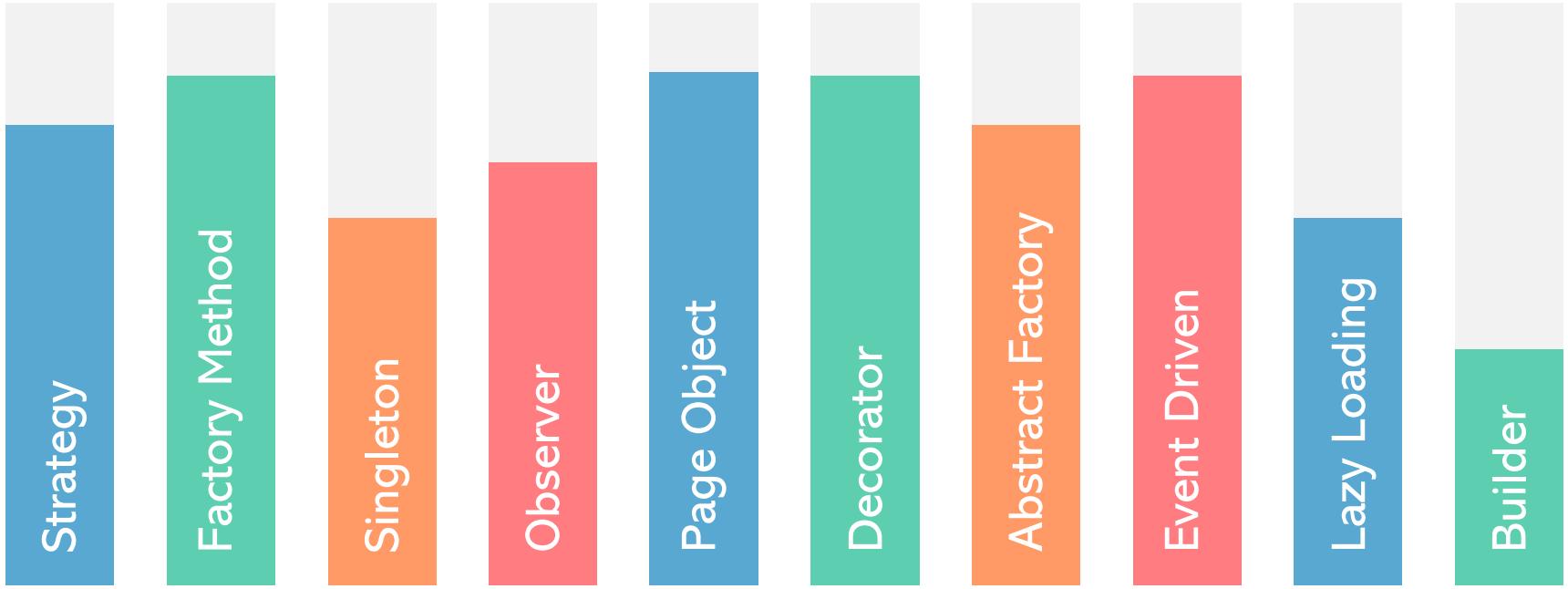


Let the consumer choose and not you.

Step 2) Identify the design patterns

TestLeaf
Always Ahead

Design Patterns are many …

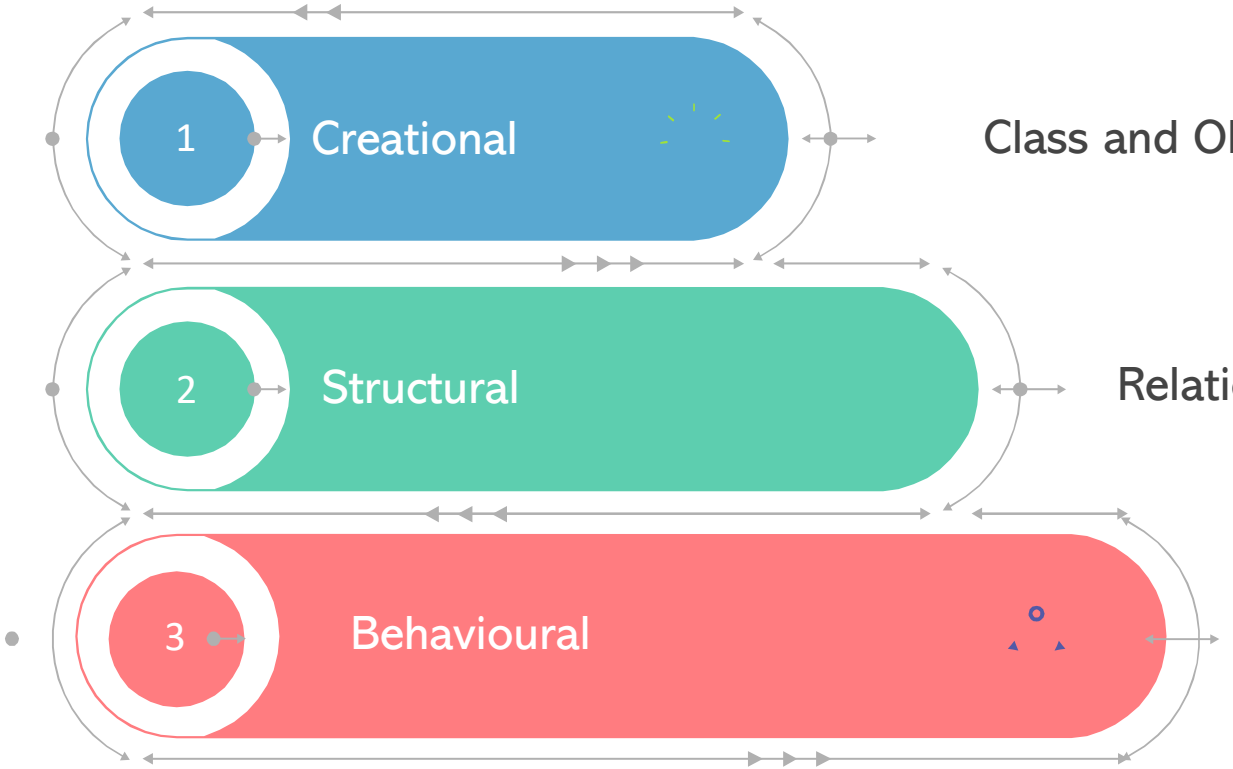


TestLeaf
Always Ahead

Java Design Patterns

https://java-design-patterns.com/patterns/

Design Pattern Type



Class and Object composition

Relationship between entities

Communication patterns between objects

TestLeaf
Always Ahead

Should I use all of them?

TestLeaf
Always Ahead

A) Creational Patterns

- **Abstract Factory**
  Creates an instance of several families of classes
- **Builder**
  Separates object construction from its representation
- **Factory Method**
  Creates an instance of several derived classes
- **Object Pool**
  Avoid expensive acquisition and release of resources by recycling objects that are no longer in use
- **Prototype**
  A fully initialized instance to be copied or cloned
- **Singleton**
  A class of which only a single instance can exist

TestLeaf
Always Ahead

Factory Design Pattern / Object Pool

One of the scenario:

- You need to run against multiple browsers,
- Each browser with different capabilities,
- Like to reuse the object - create, reuse and dismiss WebDriver instances

https://github.com/barancev/webdriver-factory

TestLeaf
Always Ahead

B) Structural Patterns

- **Adapter**
  Match interfaces of different classes
- **Bridge**
  Separates an object's interface from its implementation
- **Composite**
  A tree structure of simple and composite objects
- **Decorator**
  Add responsibilities to objects dynamically
- **Facade**
  A single class that represents an entire subsystem
- **Flyweight**
  A fine-grained instance used for efficient sharing
- **Private Class Data**
  Restricts accessor/mutator access
- **Proxy**
  An object representing another object

TestLeaf
Always Ahead

Decorator Pattern

Add responsibilities to objects dynamically

Challenges and solutions:

- Timing related failures, for which we like to have more intelligent waits
- handles unhandled alerts,
- handles "stale" elements and attempts to find them again to perform the action on the "restored" element

https://github.com/testleaf-software/devtools-selenium

TestLeaf
Always Ahead

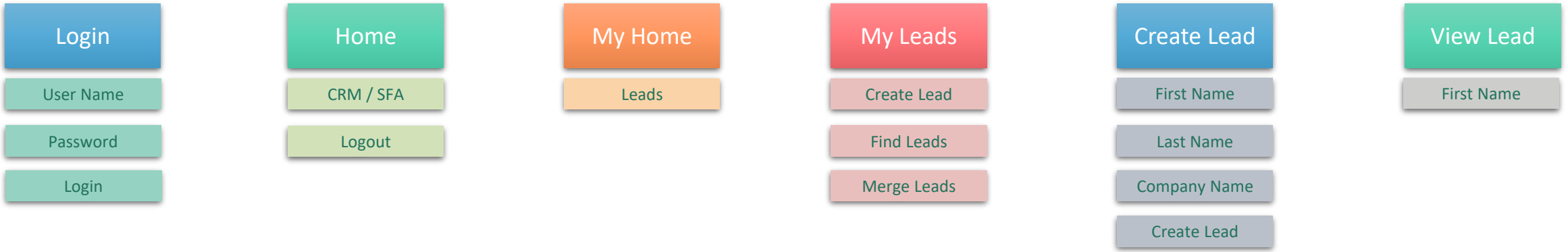
C) Behavioural Patterns

- **Chain of responsibility**
  A way of passing a request between a chain of objects
- **Command**
  Encapsulate a command request as an object
- **Interpreter**
  A way to include language elements in a program
- **Iterator**
  Sequentially access the elements of a collection
- **Mediator**
  Defines simplified communication between classes
- **Memento**
  Capture and restore an object's internal state
- **Null Object**
  Designed to act as a default value of an object
- **Observer**
  A way of notifying change to a number of classes
- **State**
  Alter an object's behavior when its state changes

TestLeaf
Always Ahead

Page Object Design Patterns

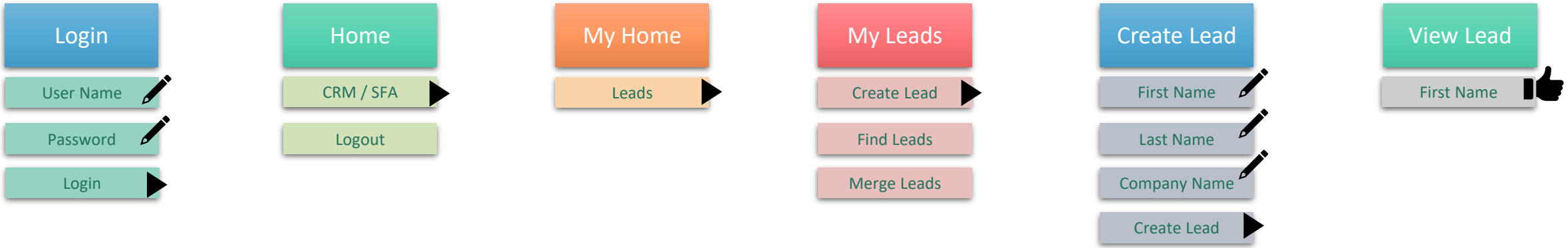
Page & Patterns



TestLeaf
Always Ahead

Page Object Design Patterns

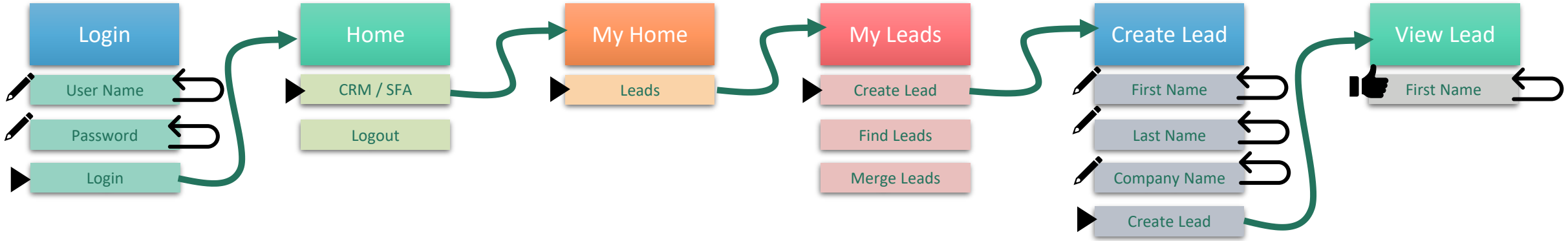
Page & Patterns



TestLeaf
Always Ahead

Page Object Design Patterns

Page & Patterns



TestLeaf
Always Ahead

BDD Pattern

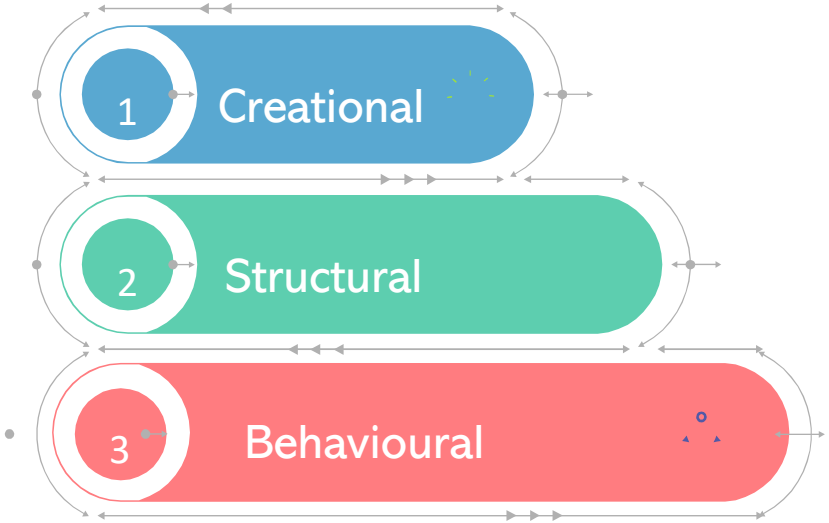
Using Cucumber, SpecFlow

TestLeaf
Always Ahead

Take Away 6

Use the appropriate design patterns based on your influencer need.

It is okay to choose multiple design pattern under one framework as long as it **does the needed** and does not conflict the solution



TestLeaf
Always Ahead

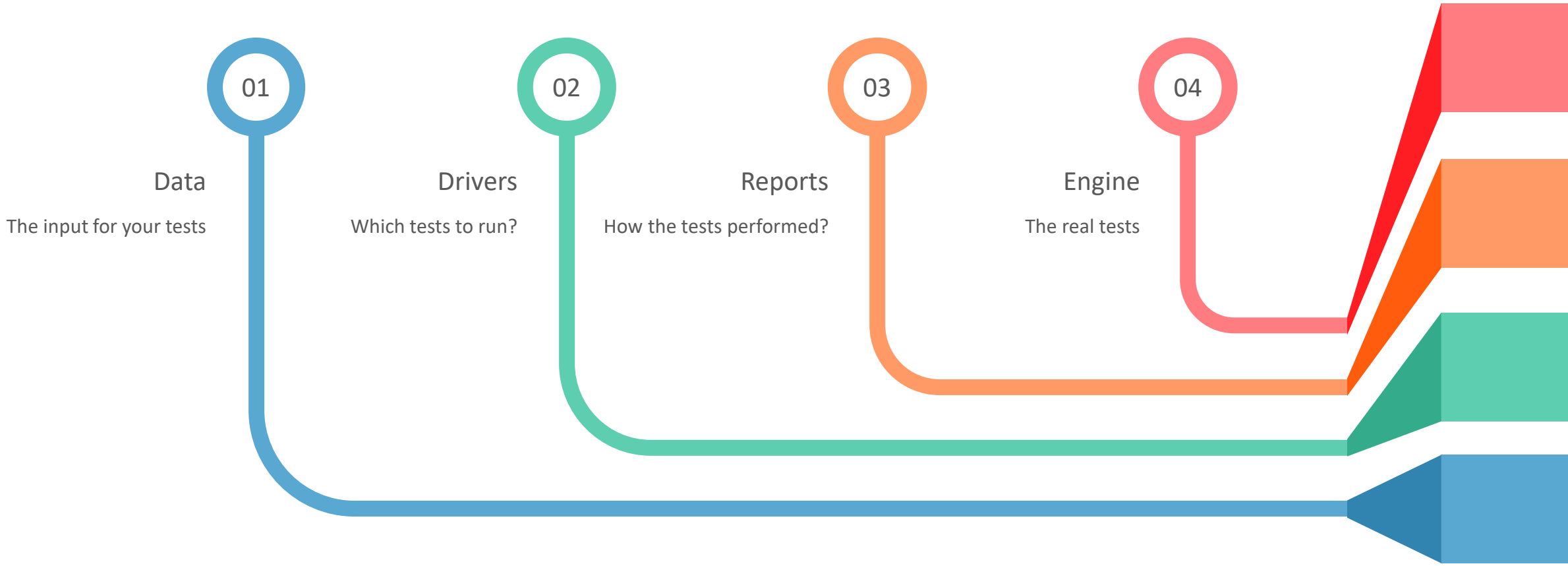
Step 3) Identify the Essential components

TestLeaf
Always Ahead

Common across all Cars

TestLeaf
Always Ahead

Essential Framework Components



Test Data Source – Build your own data engine

1 Excel / CSV

2 JSON / XML

3 Database

4 API

TestLeaf
Always Ahead

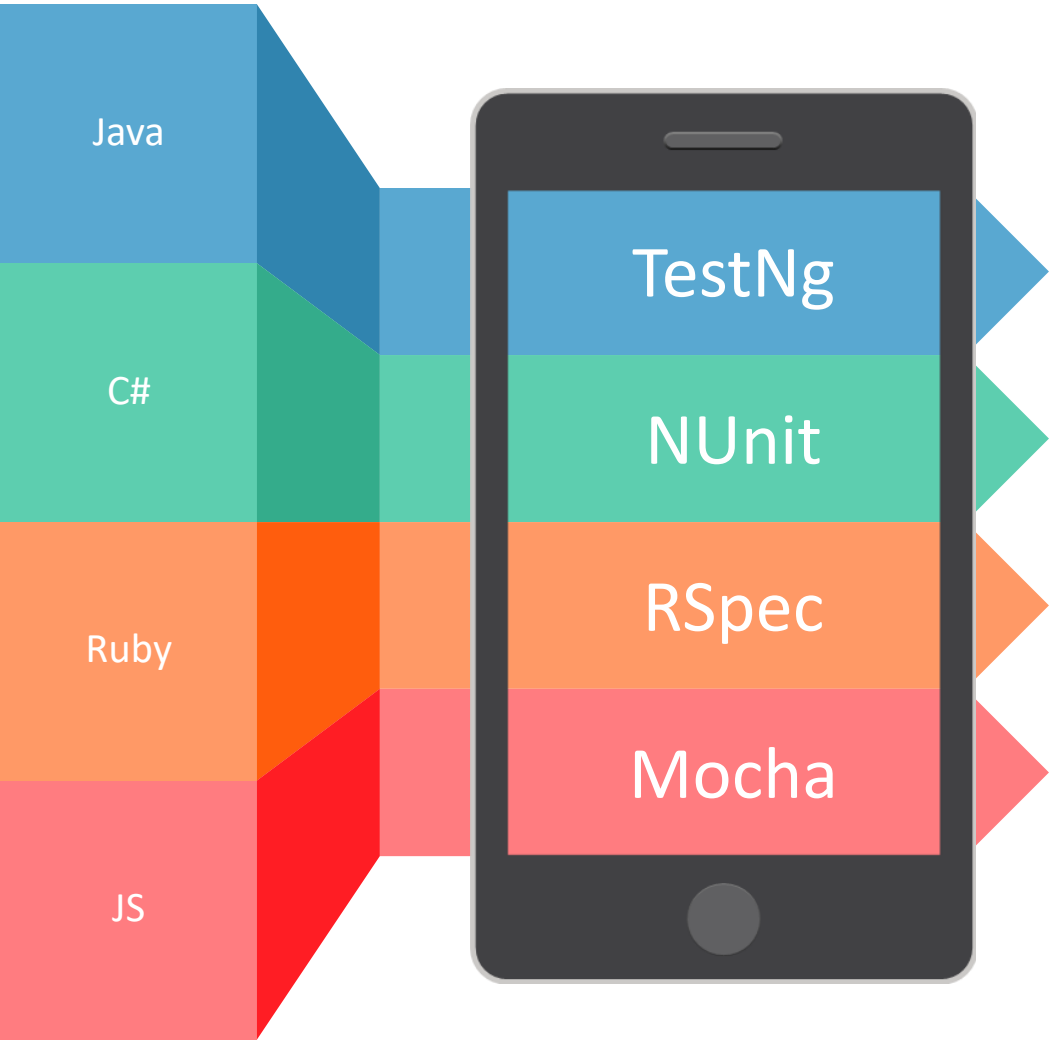
Take Away 7

Build your test data adaptor based on your data source.

1. Excel / CSV
2. JSON / XML
3. Database
4. API

TestLeaf
Always Ahead

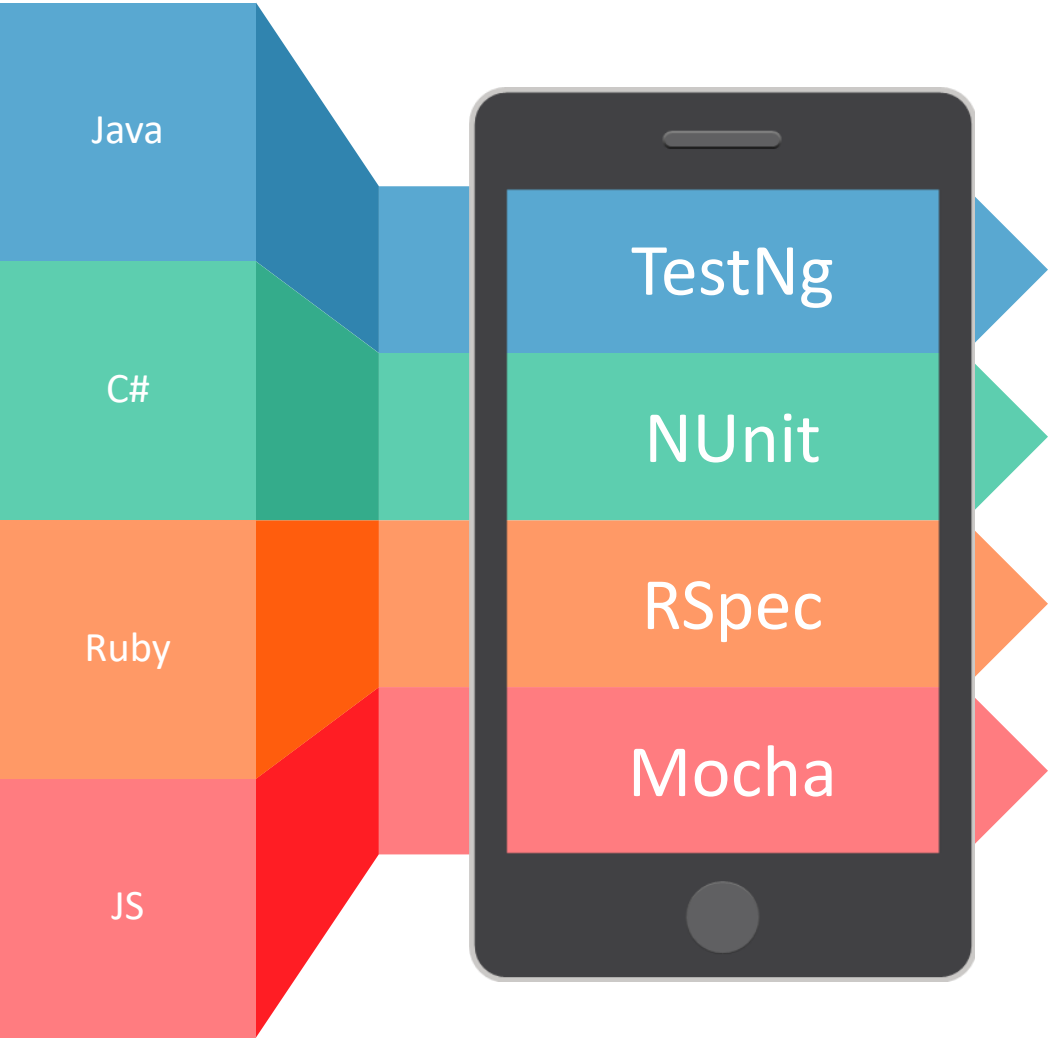
Drivers – Depends on your language & tool



TestLeaf
Always Ahead

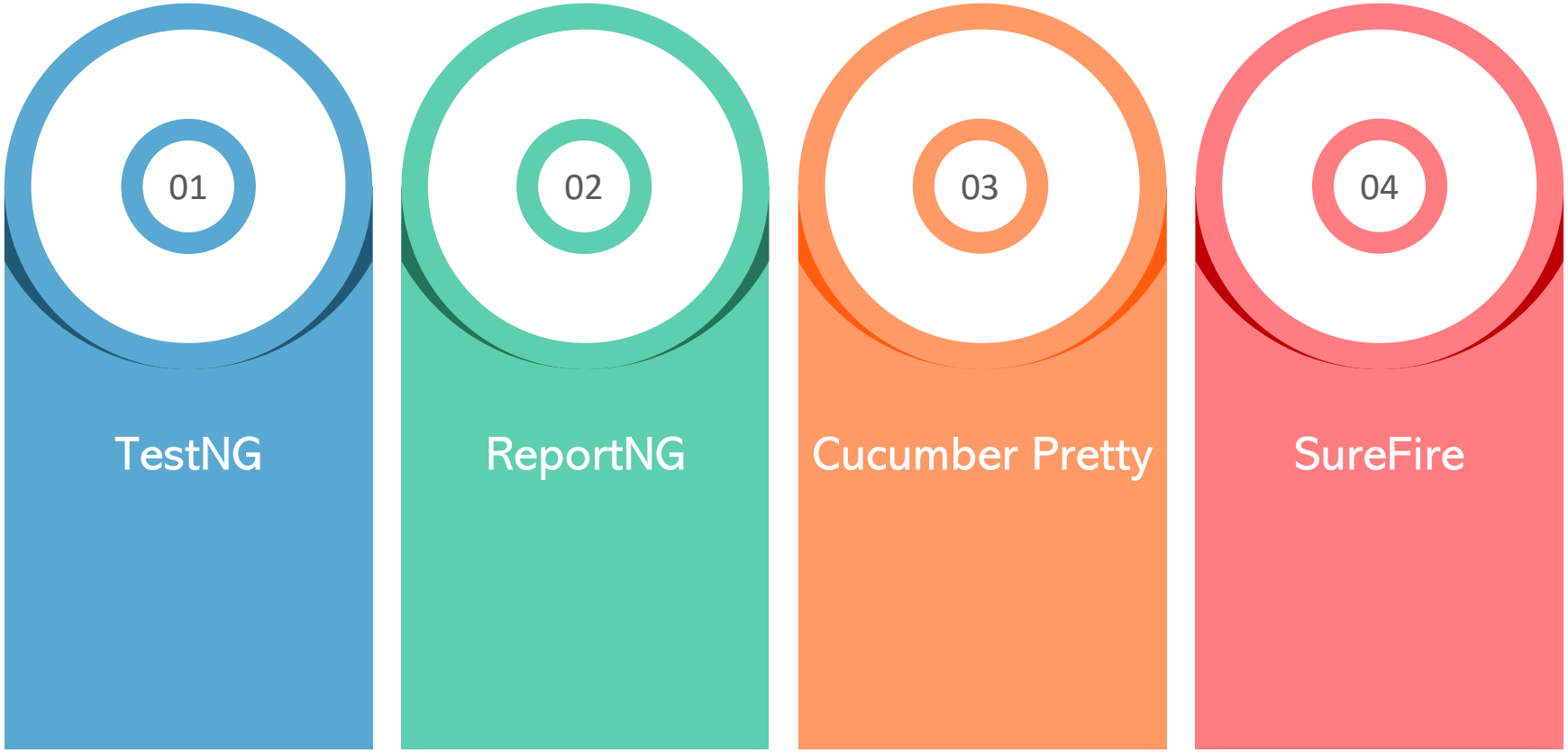
Take Away 8

Wrap your driver based on your language and tool.



TestLeaf
Always Ahead

Basic Reports through Listeners

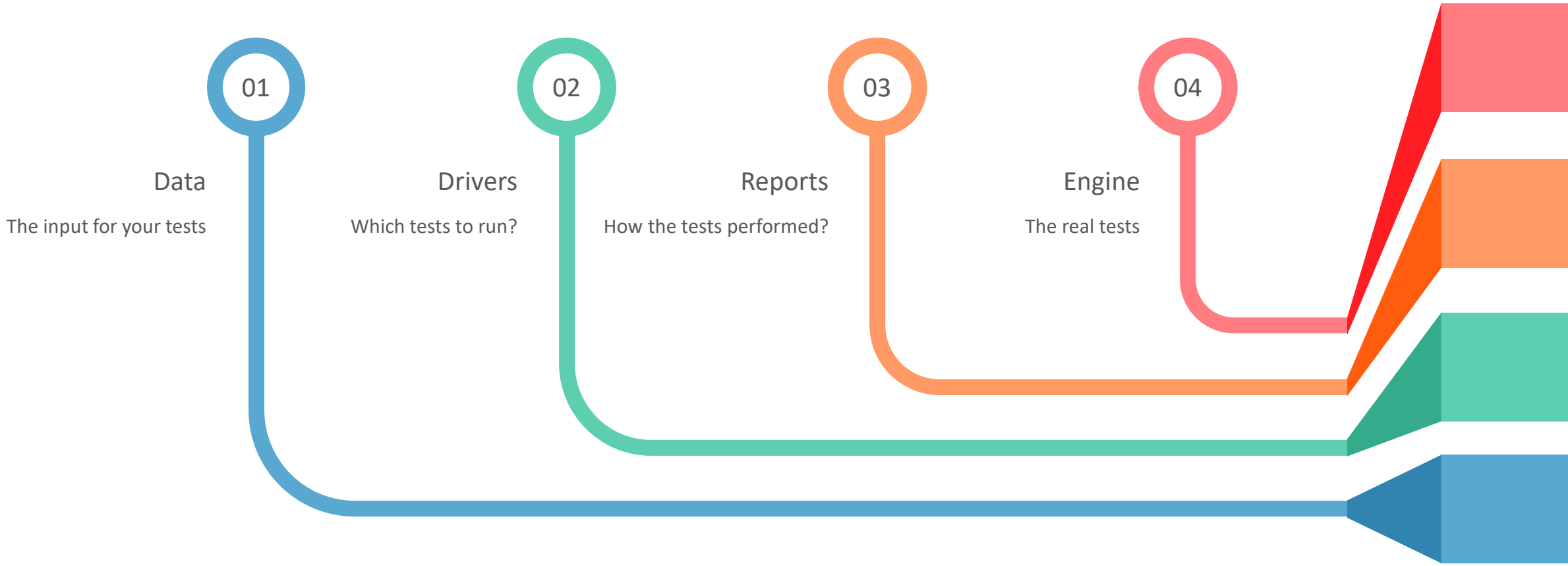


Powerful Dashboards
(With Charts, Graphs and Snaps)

Take Away 9

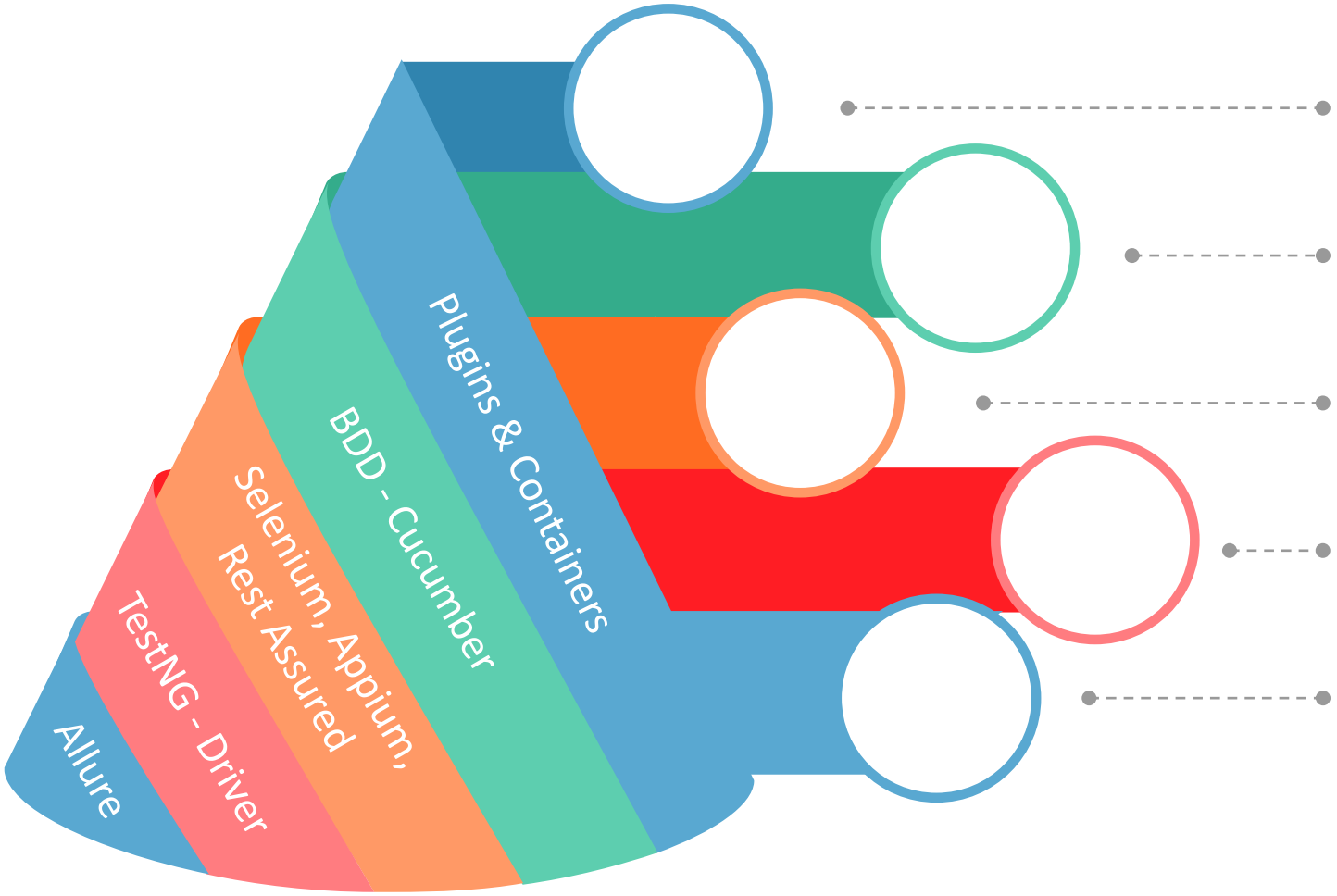
Integrate with your reporting engine based on your need, tool, language.

Engine – This is yours !!



FrameworkOne

Our Engine with Components



Plugins & Containers

Built for Jenkins and Dockers

BDD Layer

It can be Cucumber or SpecFlow for C#

UI + API Automation Layer

Selenium + Appium + Rest Assured / Rest Sharp

Driver Layer

TestNG / Junit / Nunit / RSpec Layer to run tests

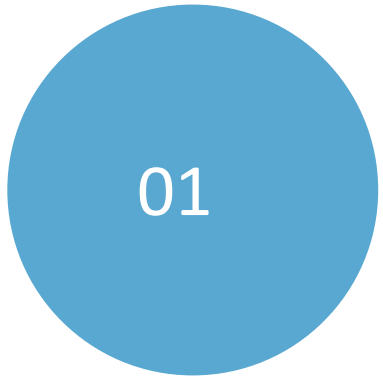
Reporting Layer

Tools: Allure / Extent Reports / Report Portal

Step 4) Identify the Necessary components

TestLeaf
Always Ahead

Retry Logic



Retry the failed locator

Retry failed test / feature

Retry the failed data

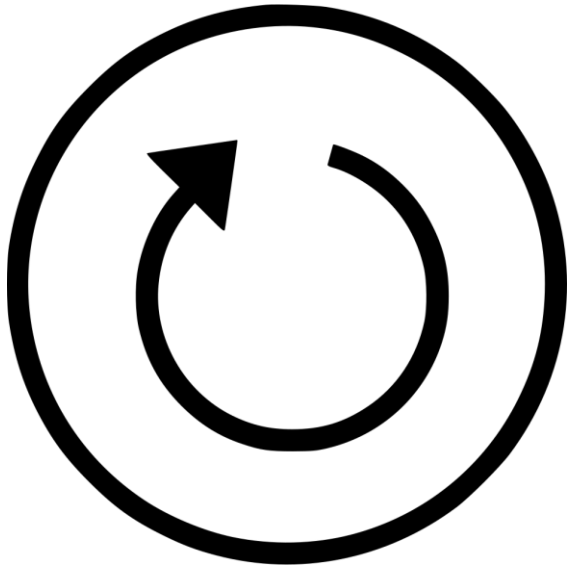
Retry the failed test suite

TestLeaf
Always Ahead

Take Away 10

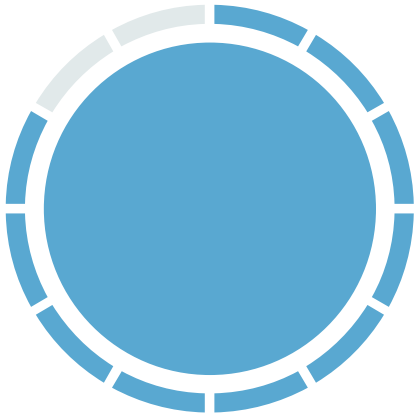
Building a powerful retry logic can save lot of time in debugging.

It is okay to retry for up to 2-3 with same or different scenario

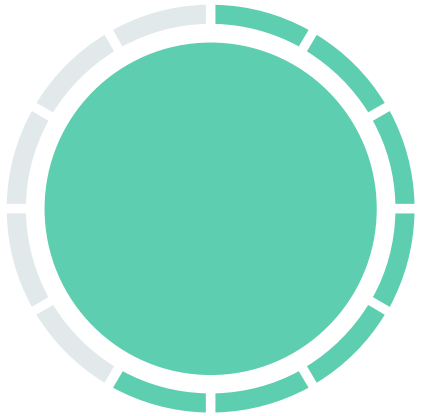


TestLeaf
Always Ahead

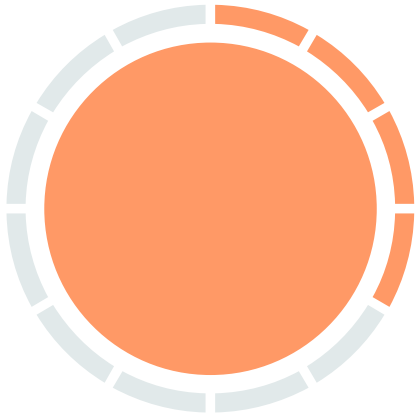
Configuration



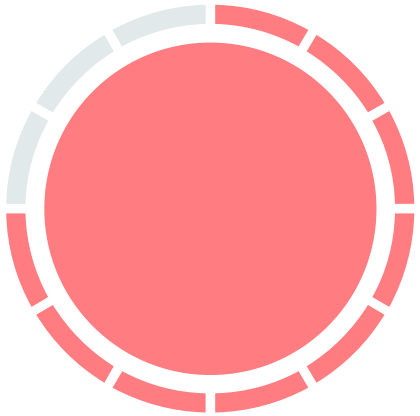
Environment



Objects



Language



Parallel

TestLeaf
Always Ahead

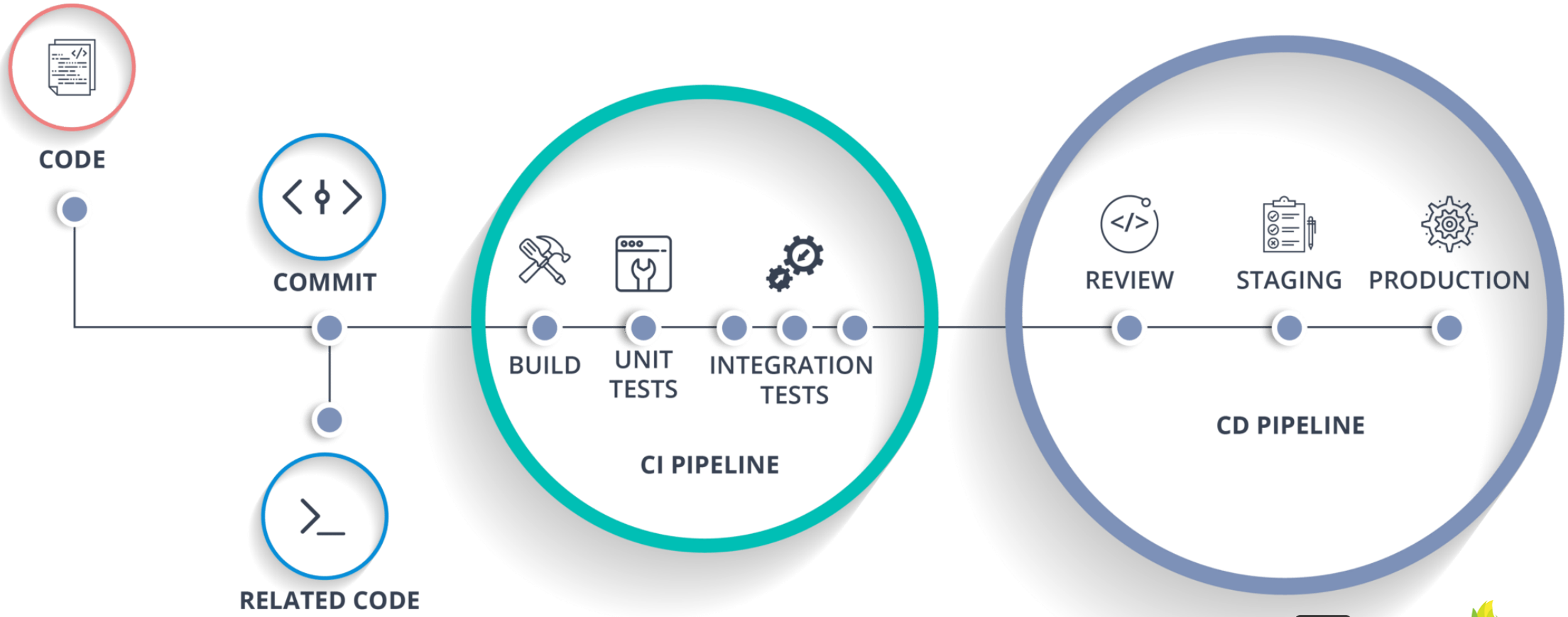
Take Away 11

Building a powerful configurator can allow to adjust your tests for the need.

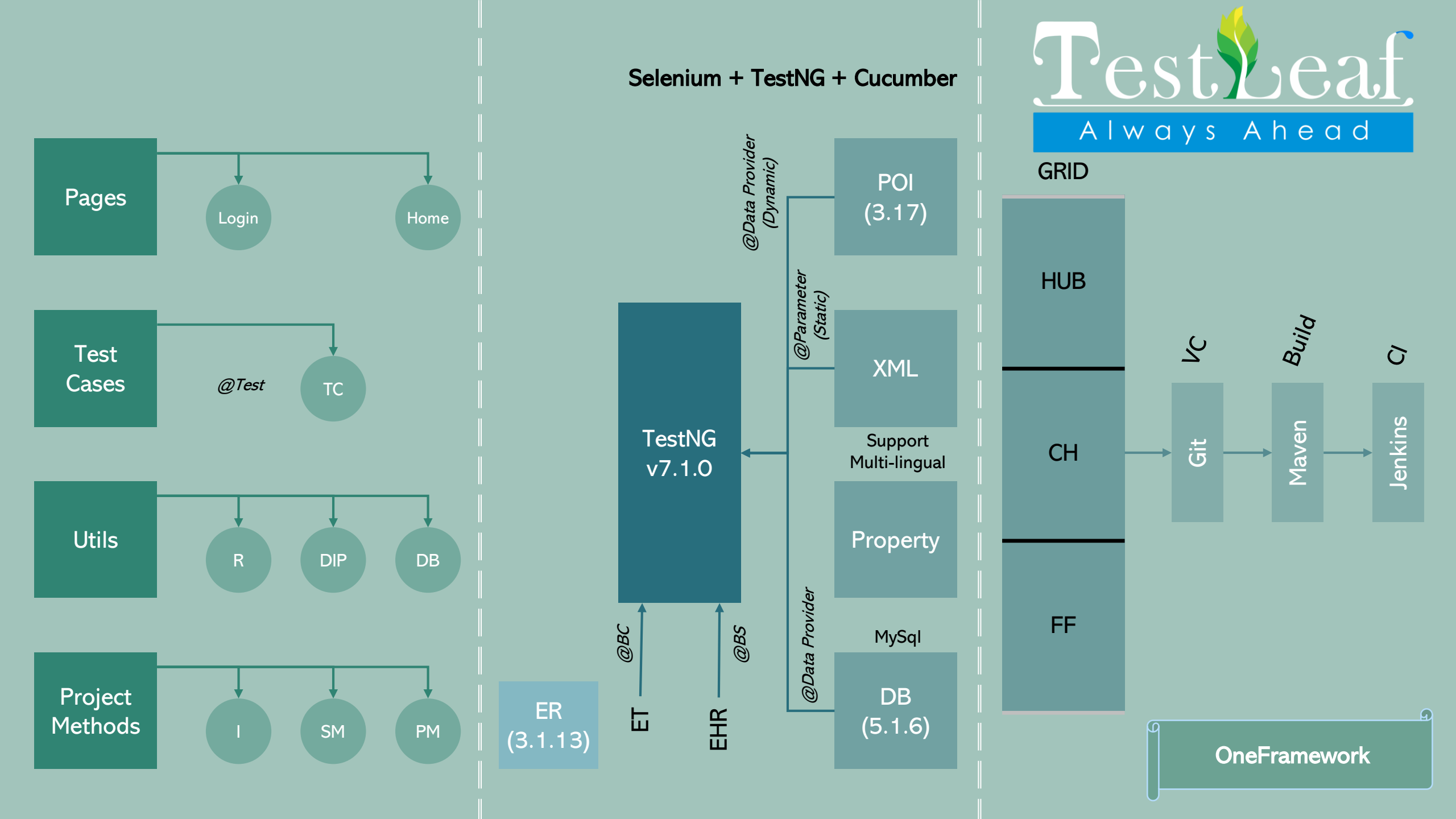
It is okay to have configurations in properties / XML / JSON (Flat file)

TestLeaf
Always Ahead

CI/CD/CT Integration



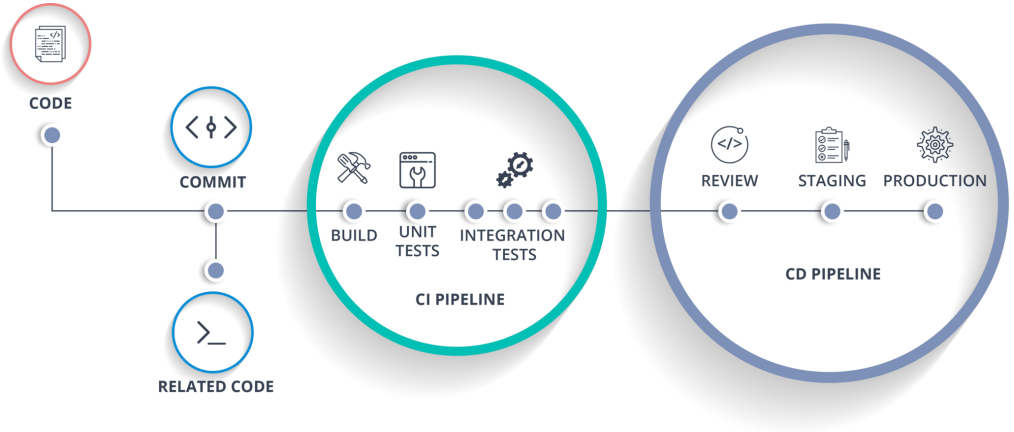
TestLeaf
Always Ahead


Selenium + TestNG + Cucumber
TestLeaf
Always Ahead
Pages
Login
Home
Test Cases
@Test
TC
Utils
R
DIP
DB
Project Methods
I
SM
PM
ER (3.1.13)
TestNG v7.1.0
@BC
@BS
ET
EHR
@Data Provider (Dynamic)
@Parameter (Static)
@Data Provider
POI (3.17)
XML
Support Multi-lingual
Property
MySql
DB (5.1.6)
GRID
HUB
CH
FF
VC
Git
Build
Maven
CI
Jenkins
OneFramework

Take Away 12

Building a feature that can allow integration with CI/CD/CT tool.

- Like API to run your tests
- Command Line Option



TestLeaf
Always Ahead

Multi Lingual

TestLeaf
Always Ahead

Take Away 13

Building a feature that can allow multi-lingual execution using:

- Properties/JSON/DB to store objects
- Object Mapping with Language

TestLeaf
Always Ahead

Step 5) Identify the Premium components

TestLeaf
Always Ahead

Distributable & Scalable

Selenium Grid

Dockerized Grid

Jenkins Master - Slave

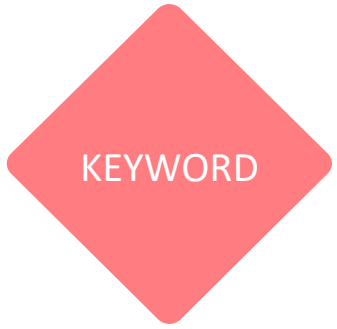
Zalenium

TestLeaf
Always Ahead

Machine Learning Algorithms

1

Defect Classification



4

Auto Fix (Sapienz)

2

Reinforcement Run Algorithms

3

Self Healing of Objects

TestLeaf

Always Ahead

Framework Accelerators

Locator Auto Identifiers

ChroPath, Ruto, Taiko

Code Generators

Automated code generators – Tuna, Elementer

Auto defect loggers

Using API, Plugins and through ML Models

Data & Document Generators

Tools like Yard

Additional Takeaways for you.

- Build your own annotations
- Use Lombok Project
- Use Documentation tool like Yard

TestLeaf
Always Ahead

MAY 2020 Online Programs

with TestLeaf

TestLeaf
Always Ahead

STARTING MAY 9TH	STARTING MAY 9TH	STARTING MAY 30	STARTING MAY 9TH	STARTING MAY 16TH
Ui	*Api*	*Core*	*Cloud*	*Angular*
SELENIUM FRAMEWORKS	REST + GRAPHQL	PYTHON + ADVANCED	AWS ARCHITECT	PRO - TRACTOR
~~10000~~	~~4720~~	~~11800~~	~~11800~~	~~8850~~
4999*	2499*	7500*	7500*	5000*
5 WEEKENDS	3 WEEKENDS	5 WEEKENDS	5 WEEKENDS	3 WEEKENDS

CALL +91 98405 77905

Thank you

TestLeaf
Always Ahead