Java-8

    => Lambdas

    Functional Programming

those feature that define functional programming

streams

Executor (Future)

Concurrency Collection

Style :

Traditional : Imperative

    (HOW)

    #exposing the steps how to perform an operation
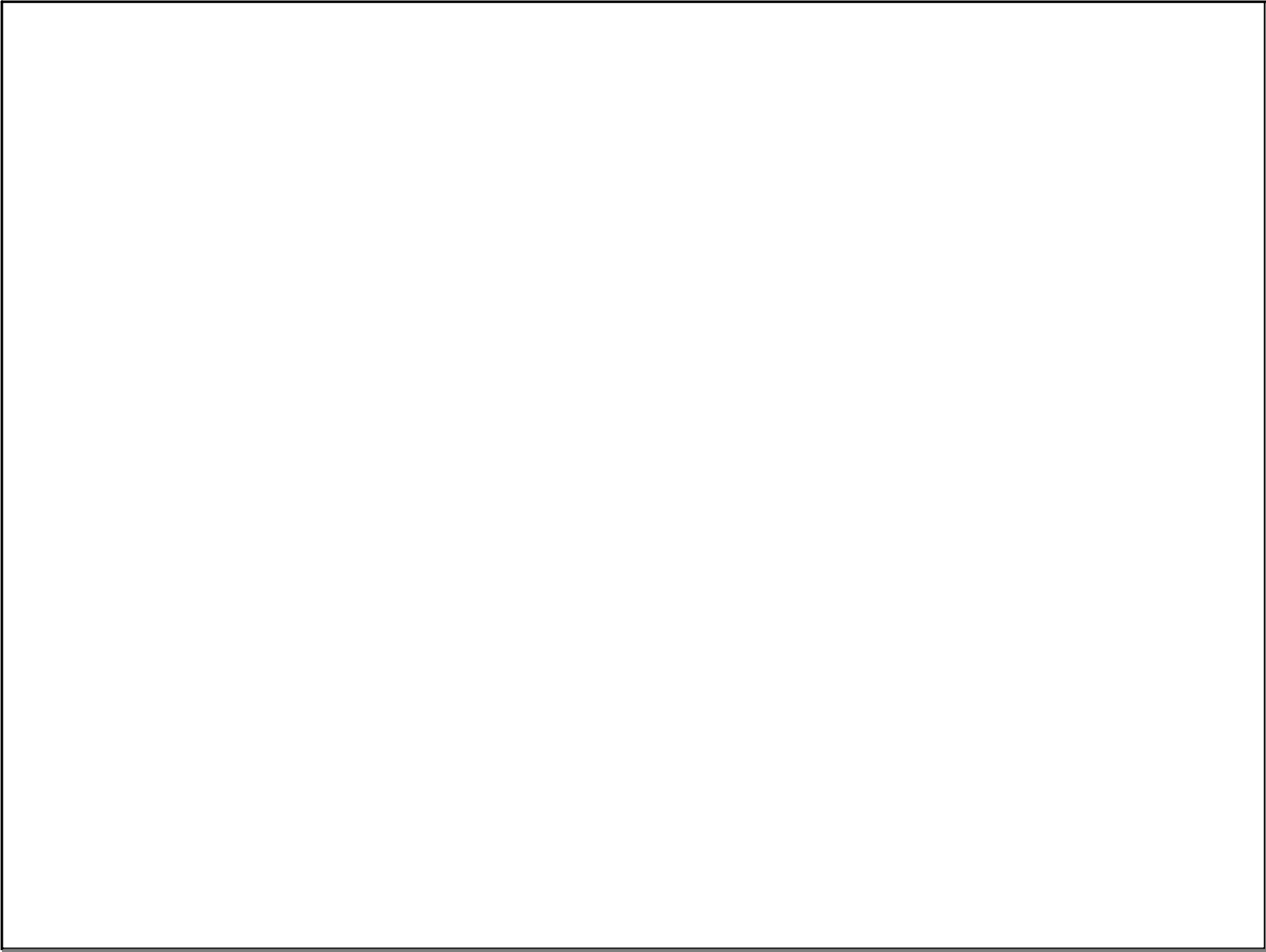
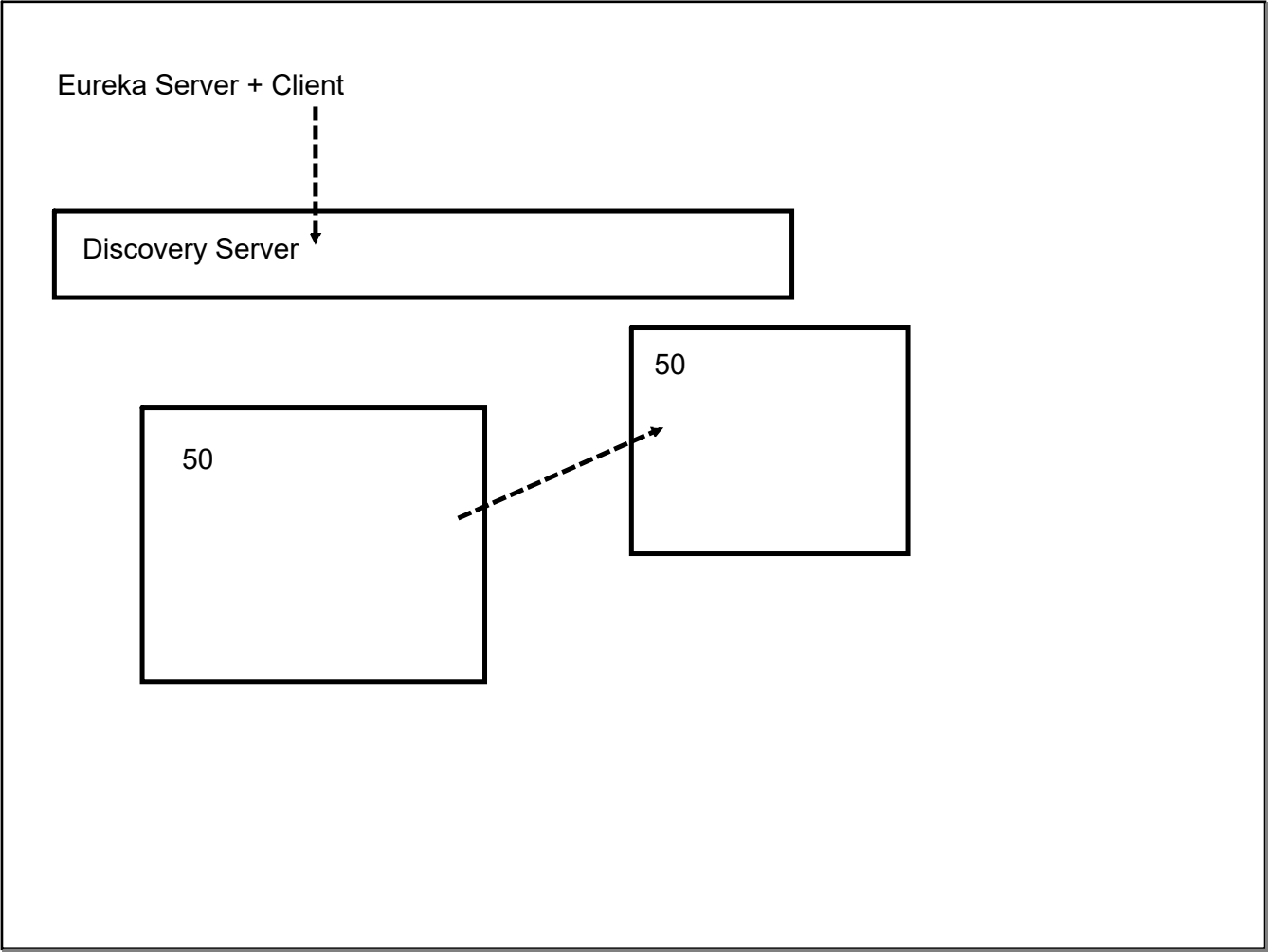   # embrace object mutability (not in sync with concurrency)

Functional : Declarative

    (What) : result
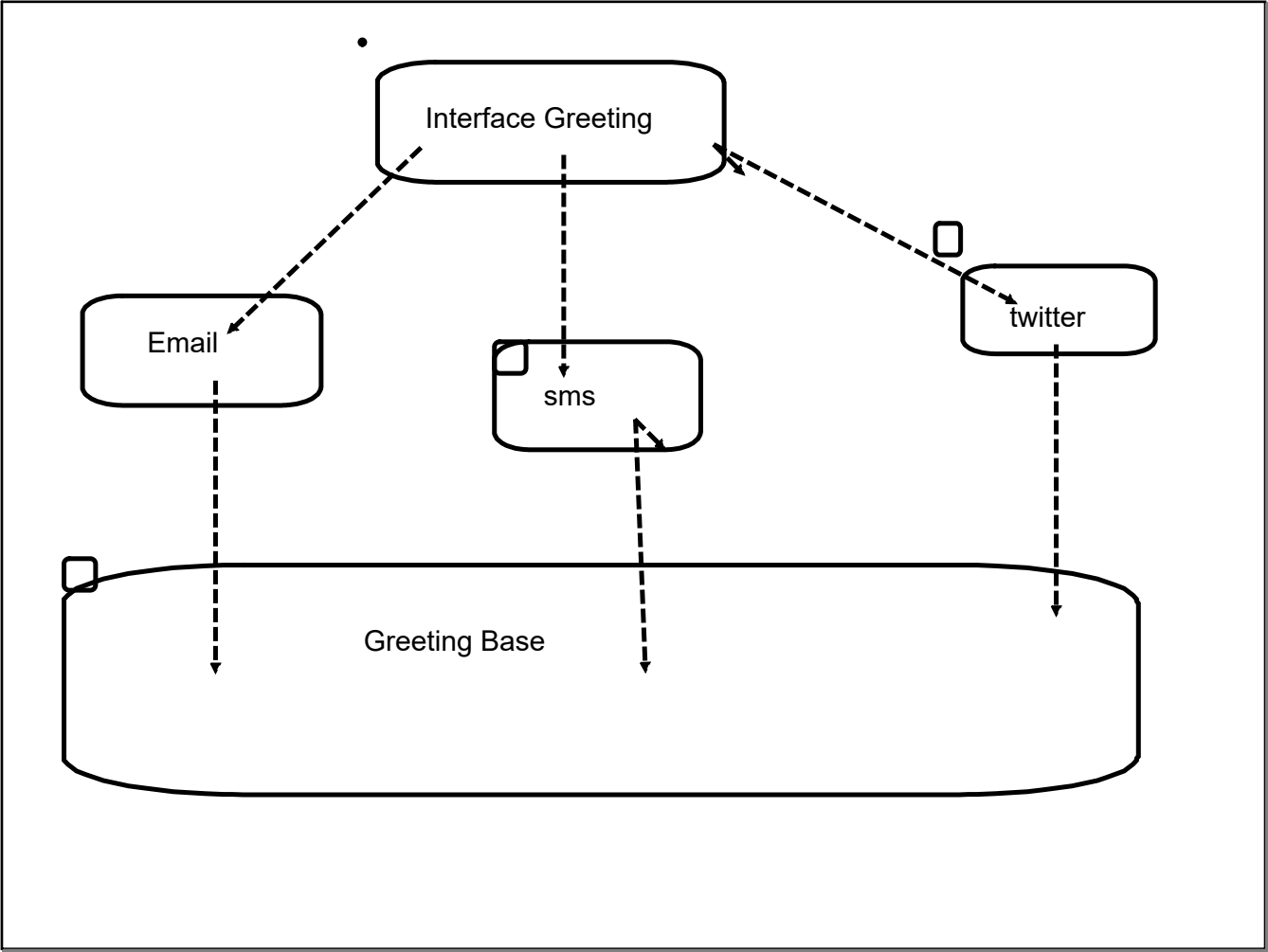
    immutability

    Analogous SQL

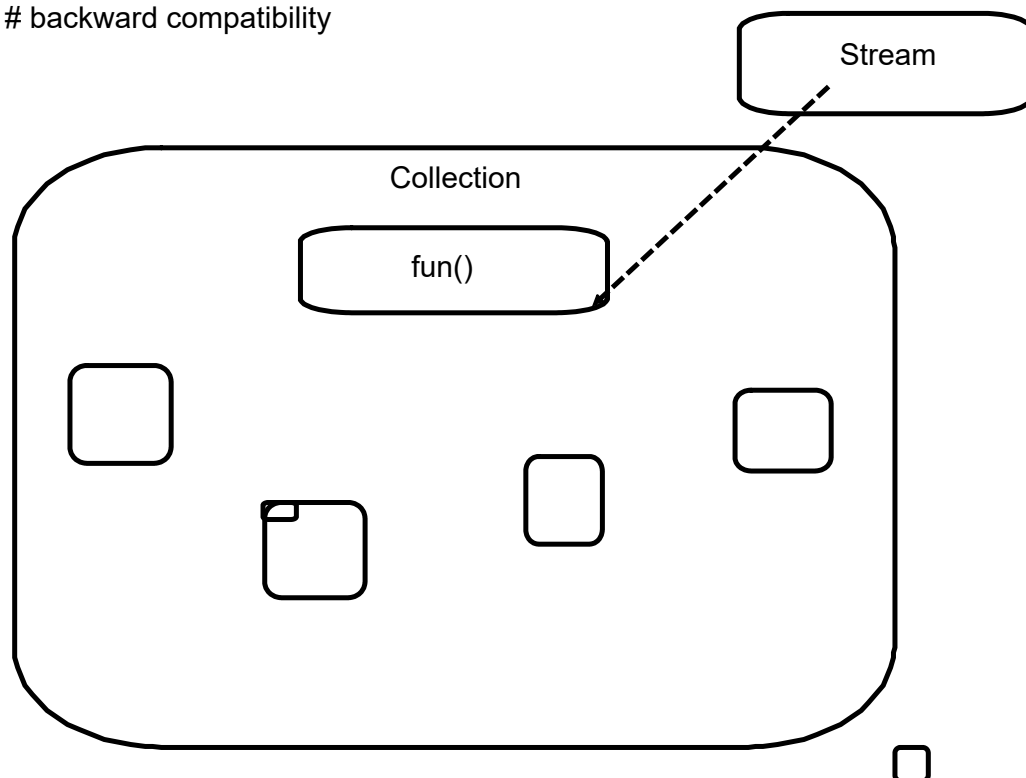Eureka Server + Client

Discovery Server

50

50

Interface :

 # default method (definition)

     # multiple implementations can be inheritance

     # backward compatibility

Stream

Collection

fun()

Escape from OOPs

independent Functions (not wrapped inside an object)

Relationship between interface and function

1. interface must have only one abstract method (any number of default/static) :

Functional Interface : Annotation @FunctionalInterface

2. single method signature must match with function implementation

```
Lambda expression
    (<arg1>,<arg2>) -> {
}

    arg1 -> {
    }

    () -> {
    }

    (<arg1>) -> <return> <single instruction>

    (a,b) -> <return>a+b;

    (a,b) -> {
        return a+b;
    }
```

Pre defined functional interfaces

    => Runnable

    => Comparator

Explicit Functional Interface

    # Consumer

        void accept(<>);

        DoubleConsumer() // specialized implementations on primitive

        BiConsumer

            void accept(<>,<>);

    # Predicate (test)

        boolean test(<>)

    # Supplier

        <> get()

    # Function

        <> apply(<>)

Stream :

not a data structure

immutable (Thread safe)
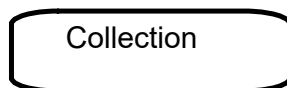
Collection

Stream   p / for   Terminal

                conveyer belt

Stream : LAZY Processes

Stream can have 2 type of activities

      intermidiate activities (filter/map/flat map)

      Terminal activity : terminate/close the stream

               forEach()

               collect()

IF terminal activity is not present : stream will not initiate

groupingBy(<return> Function(student))

return value : would become a group

Transforms

y map(x)

flatmap() : Collection into stream

map:
[""," "]

[""," ",""]

[""," "]

flat map
["","","","","","","",""]

return type fixed : stream of data passed as argument

(Stream of) Multiple collection

into (Stream of)single collection

Stream :

   # Sequential Stream

   # Parallel Stream

Terminal
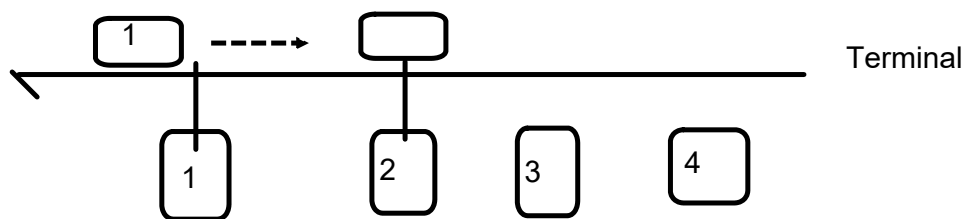
Parallel Streaming not commended if working on external mutable data (not thread safe)

Activities that are inherently complex

Binary Operator : variant Function

y Function(x) : x and y can be of different type

z BinaryOperator(x,y) : x,y,z : must be of same type

Multithreading :

interleaved (Threaded Multitasking)

    1. Multiple activities waiting for I/O : that time can be used by tasks

    2. Multi-core architecture of micro-processor

Base Interface :

    Runnable (run)

Implementation:

    Core Functionality of Multithreading (Thread)

inheriting Runnable

inheriting Thread

Traditional approach:

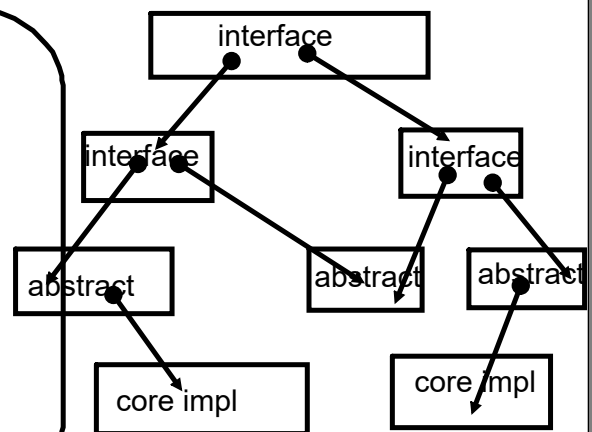   create, execute, manage

    Thread per task

    Not possible to keep a track of number of thread

Executor framework :

    do all management + improve in performance

ThreadPool : create a predefined pool of thread

interface

interface

interface

abstract

core impl

core impl

interface Executor

execute()

Method allows to allot thread from thread pool and run the task

interface ExecutorService

submit()
shutdown()
shutdownnow()

AbstractThreadPoolExecutor

interface ScheduledExecutorService

schedule() | scheduleAtFixedRate()

custom

Direct Object
Factory method (preconfig)

(Class)
Executors

newFixedThreadPool()

newSingleThreadedExecutor()

newCachedThreadPool()

newScheduledThreadPool()

Core implementation class
ThreadPoolExecutor

ScheduledThreadPoolExecutor

Need to create instance of ThreadPoolExecutor

FixedThreadPool (number of thread are predefined(extra task alloted will added to queue)

CustomThreadPoolExecutor

    <corePoolSize> : number of threads to always keep even if they are idle (2)

    <maxPoolSize>: max no of thread (5)

    <keepAliveTime> : time to wait before idle thread gets removed/released from thread pool

    <TimeUnit> :

    <queue capacity>: capacity of queue

    <RejectedHAndler> : what to do if a task is rejected from queue

SingleThreadExecutor()

FixedThreadExecutor(1)

   can change the thread capacity

CachedThreadPool() : Unbounded ThreadPool : Max Integer Val

   if demand decreases : can tear down thread

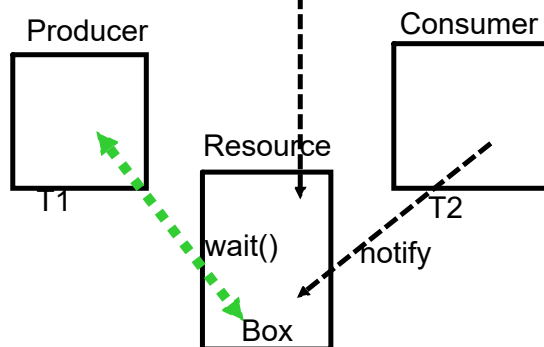   default keep alive time : 1 min

ScheduleThreadPool()

Thread returning some value

 Special Interface Callable (call())

ThreadPool can work on Callable

 wait/notify/notifyAll() : Object

Producer   Consumer

Resource

T1   wait()   notify   T2

Box

ExecutorCompletionService

: will going to get results in order of completion of task

Future : blocking

CompletableFuture <callback : logic to follow when task is done>

Functional interfaces
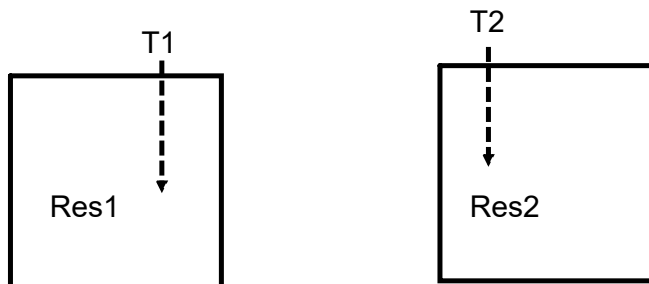

Runnable

Callable

=> Supplier

the method to associate a callback function

1. thenApply(Function); // transform

2. thenAccept(Consumer); // consuming and using

CompleatableFuture by default uses the inbuilt thread pool

ForkJoinPool.commonPool();

Executor ThreadPool

T1

T2

Res1

Res2

Common Resource Shared among multiple threads (Thread safe)

Resolve Data inconsistency

T1 - - - - - - -→ obj.manageArray()

T2 - - - - - - -→ obj.manageArray();

synchronized method

synchronized block

T1 - - - - - - -→ obj.manageArray()

T2 - - - - - - -→ obj2.changeArray();

monitor

T2.ob1
obj1

T2.obj2

T1
changeArray()

manageArray()

changeArray()

manageArray()

23
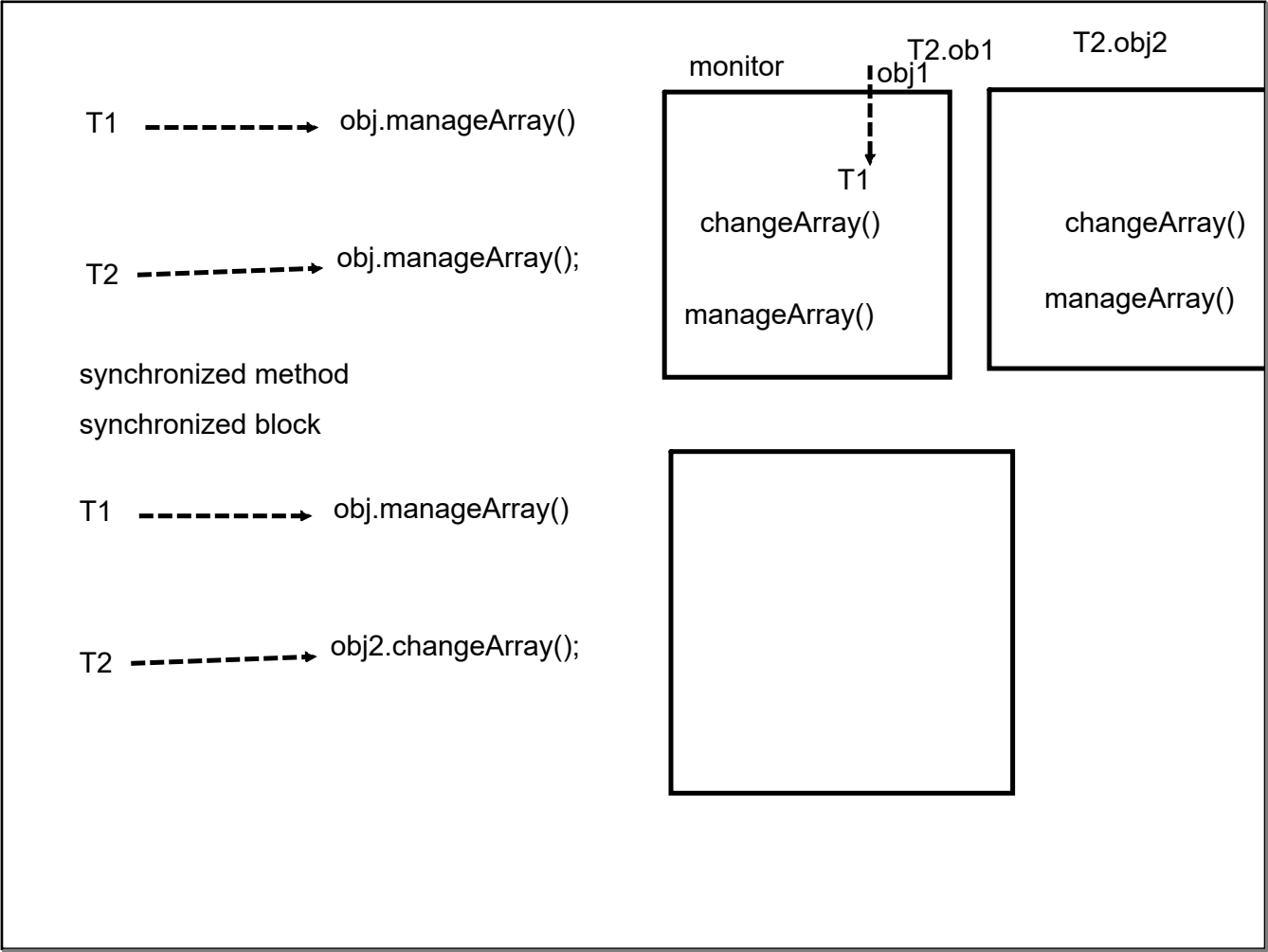
locking :

=>wide spectrum locking : (synchronized...)

=>granular locking

Collection API

1 .Traditional : 2

    1. HashTable

    2. Vector

2. To get a Thread safe variant of those class

    Collections.concurrentList();
all methods are sync

java.util.concurrent.

  API : Granular locking on resources

Atomic operation : single CPU instruction

n=10; // Thread safe operations

assignment long/double are non-atomic

 Concurent API : Focus on granular locking

 Provides Atomic Variant of type : allow to convert non-atomic activities into atomic

 # multiple approach for ThreadSafety along with high level of concurrency

ConcurrentHashMap

    create segments internally

    segment : HashTable

    by default it create 16 segment

n elements in collection

n segment

n threads can use
hashmap in thread safe

Servlet Technology

Spring MVC_{receive} respond
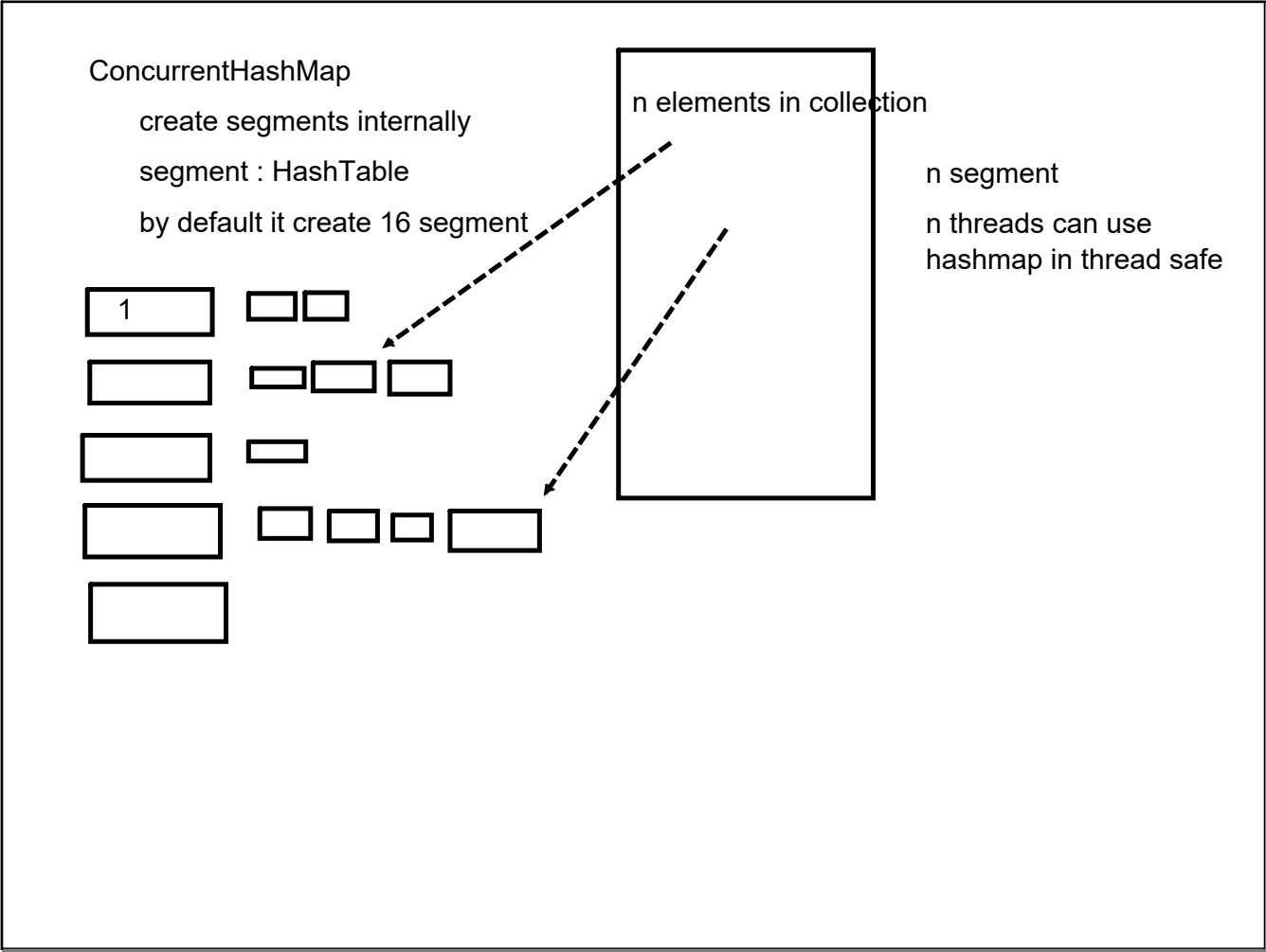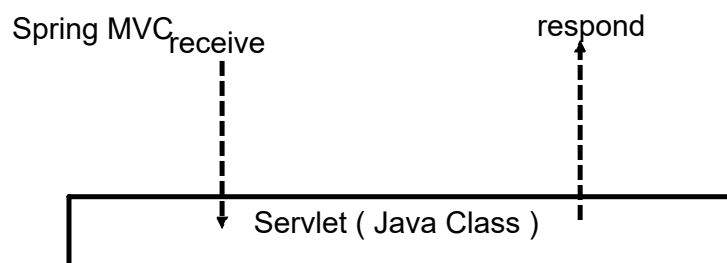
Servlet (Java Class)

How to define java class as Servlet

Extends

HttpServlet/GenericServlet

GenericServlet : does not classifies between various HTTP Verbs

HttpServlet : can identify

GET/POST/PUT/DELETE/PATCH

Life Cycle

=> init

=> generic service() / http (doPost()/doGet()/doDelete()

=>destroy

Ist request

Servlet Technology

(Controlled By web.xml)

Multiple request simultaneously

instance of servlet class is created

init phase gets activated

service() (Responsible for receive request/and respond)

Web Servers tends to cache the servlet object

destroy

Object is released
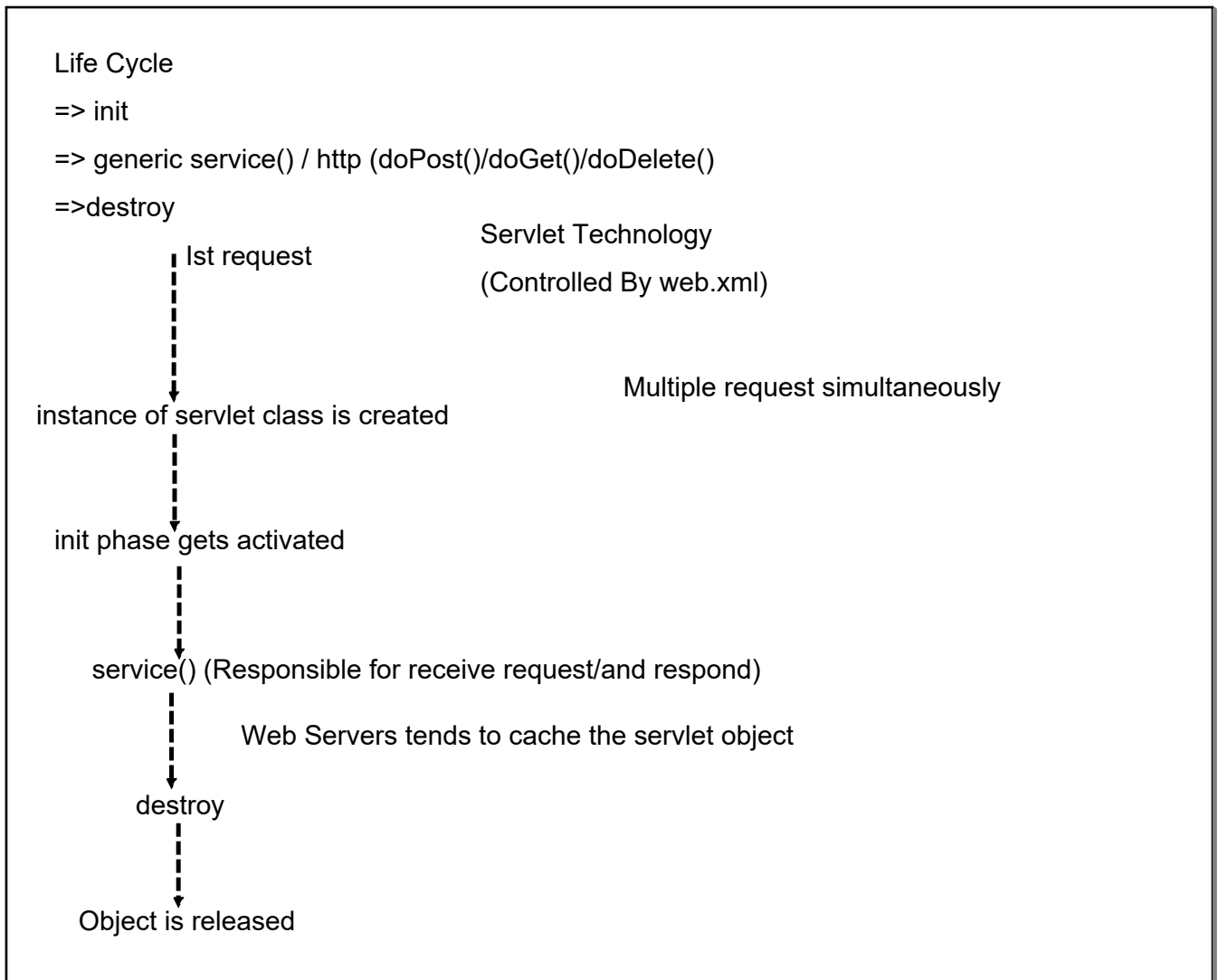
service method and variant

have an access over HttpRequest/HttpResponse object
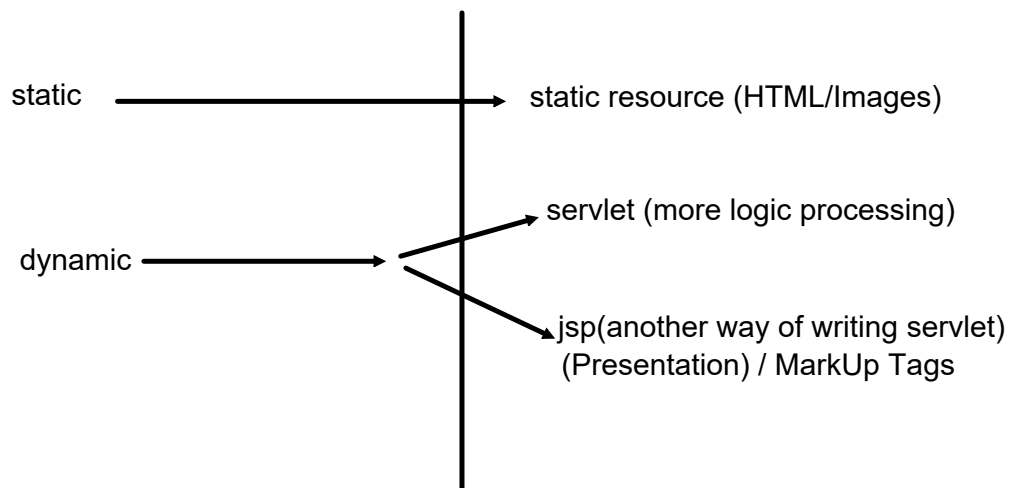
JSP : another way of writing servlet

static ⟶ static resource (HTML/Images)

dynamic ⟶ servlet (more logic processing)

jsp(another way of writing servlet)
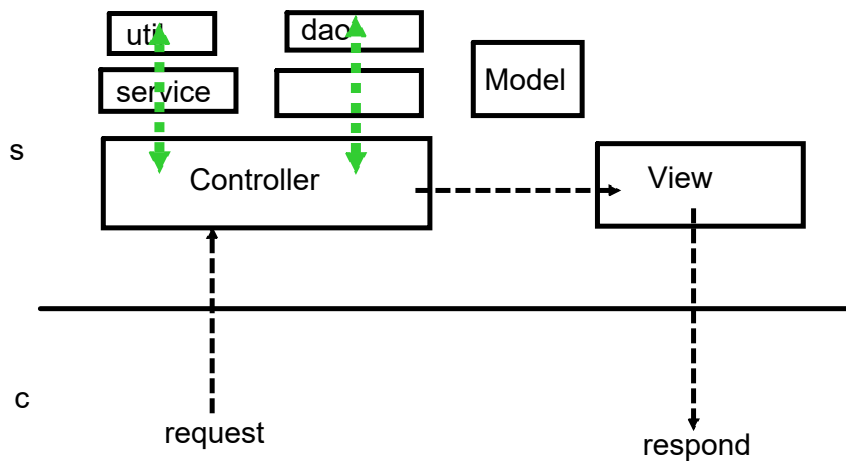(Presentation) / MarkUp Tags

Spring uses Servlet Technology:

    But provides a high level abstraction over complexities/ boilerplate req / config

and enhances the seperation of concerns

 MVC architecture

Controller : to receive request / process it

Servlet

      service method as task :

      assign it to thread

Client   request   response

Server

Front Controller : Inbuilt Servlet [DispatcherServlet]

req/resp 1

Model

2

Handler
Mapper (inbuilt)

Keeps a track
of all the
url mapping

Controller
(custom)

Handler Code
resides here

View
Resolver (inbuilt)
(config)
=>Template
=>location

View
(Custom)
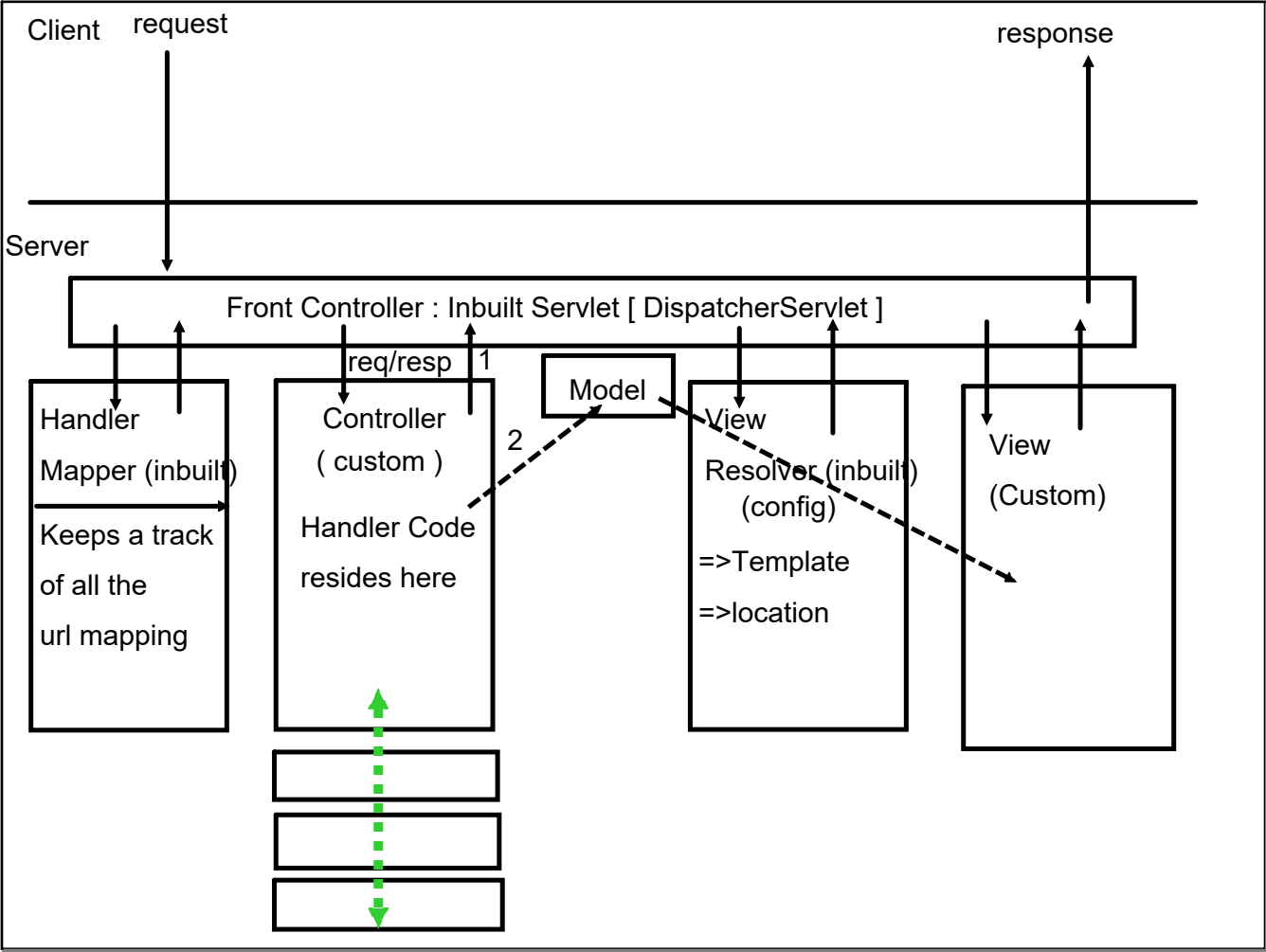
we need to register your app resources (servlet spec)

Servlet :

need to register

registeration can also be done using annotation

Register DispatcherServlet

Config of Spring in place

xml file

java

Need Spring config to connect with DS

xml : <servlet-name>-servlet.xml

View Resolver : location + template (jsp+jstl) [extension]

Controller : "index"

create a complete path

```xml
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
     <property name="prefix" value="/WEB-INF/views/" />
     <property name="suffix" value=".jsp"/>
  </bean>
```
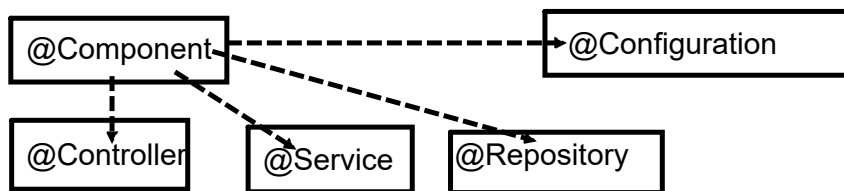
Controller return : "index"

/WEB-INF/views/index.jsp

@Service   @Repository
@Component   @Component   DB

@Controller
@Component

Controller   Class   Class

Model   business

View   logic
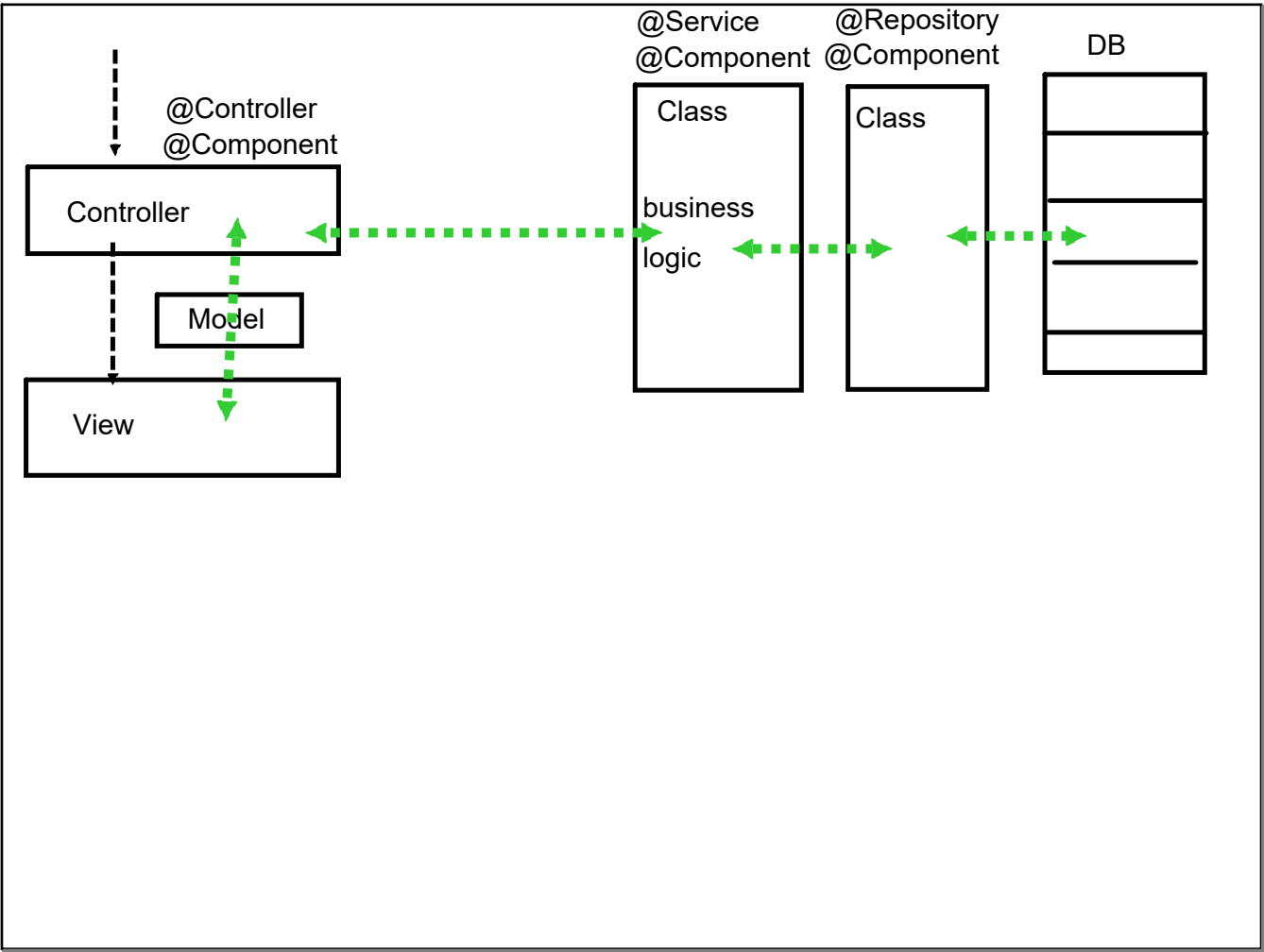
web.xml : ~ java config class

dispatcher-servlet.xml : ~ java config class

1. alternate for packaging : maven war plugin

Spring provides an inbuilt class to register DS

Spring-Boot

=> dependency management

=> configuration

dependency management

library + external other resources

spring-boot :

    dependency

    spring-starter-project

Spring - starter -(parent)project
Support of Auto-configuration / less amount of code
manage Embedded Server

Created/Developed

and Launched

stand-alone way

web-mvc

data-jpa

tomcat
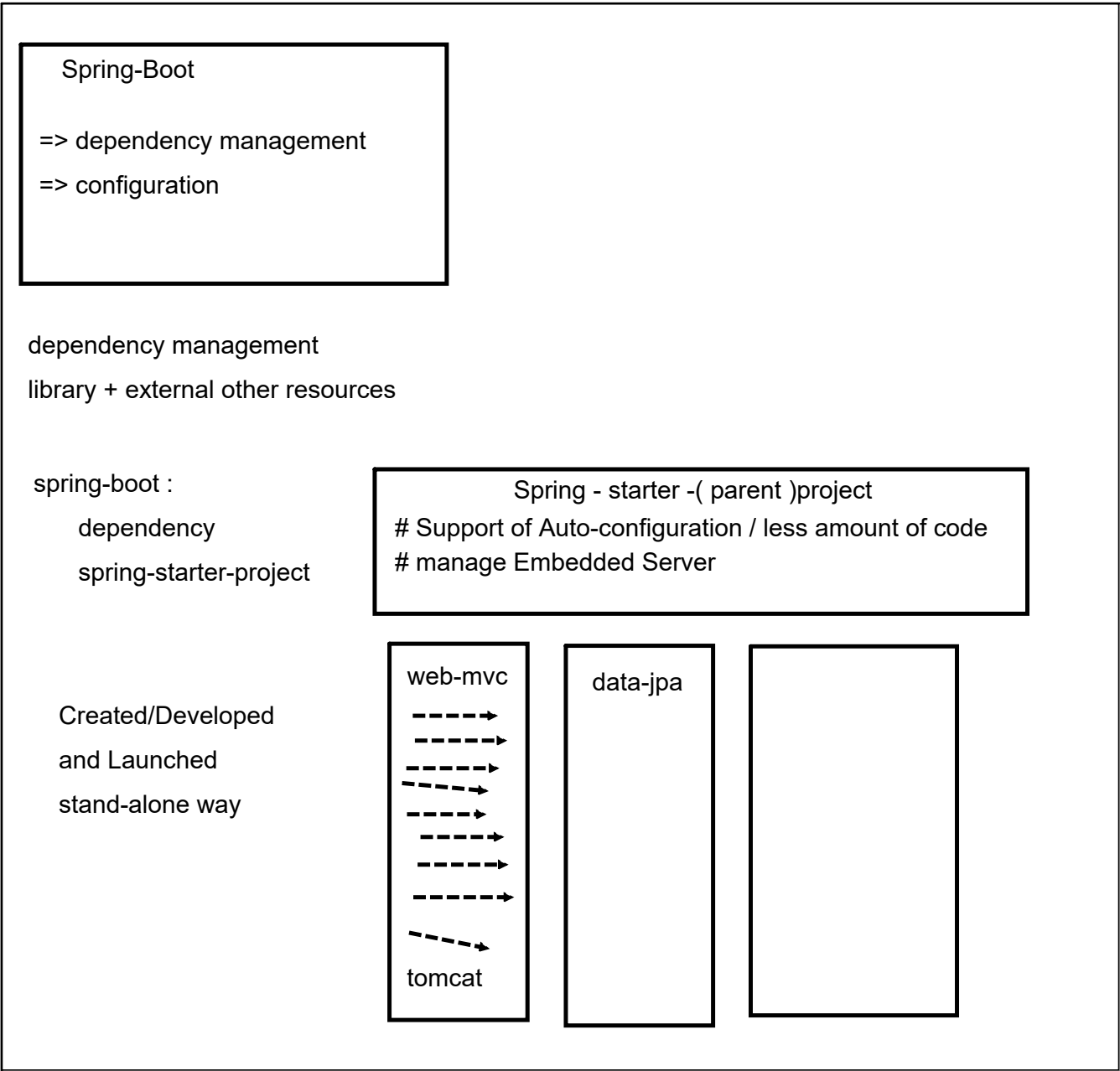
start.spring.io

maven cli

maven command

Configuration

    # Spring boot Annotation

    # Dependency

    # Customization : special file application.properties

       key=value

key : predefined keys from different spring projects

    : possible values

    : custom keys/values

spring : yaml

     : heirarchy

     : application.yaml

Spring Boot Annotation

    curated list of multiple annotation

EnableAutoConfiguration

    # tracking the dependencies

    # based on dependencies added: defined in config-file

        add default config   cli : key-values

        expose the key

    eg:

        maven-web : Spring mvc:

            DS servlet

        spring-security

            add default security

            expose username/passed

    # tracking the properties files

        looks for custom key-values pairs

mvc application

    controller

    view

pre-configured to use thymeleaf

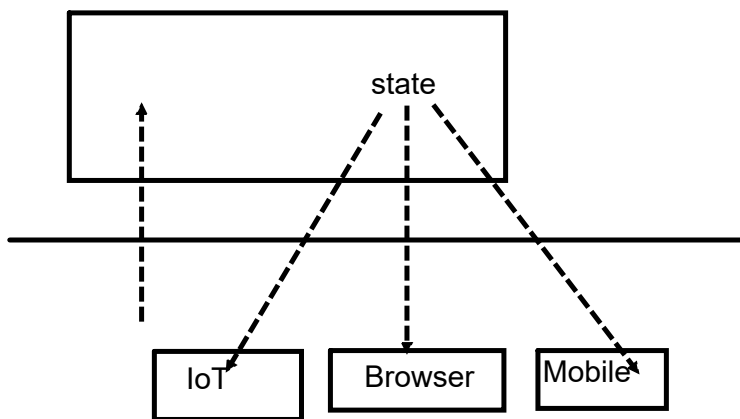View pages :
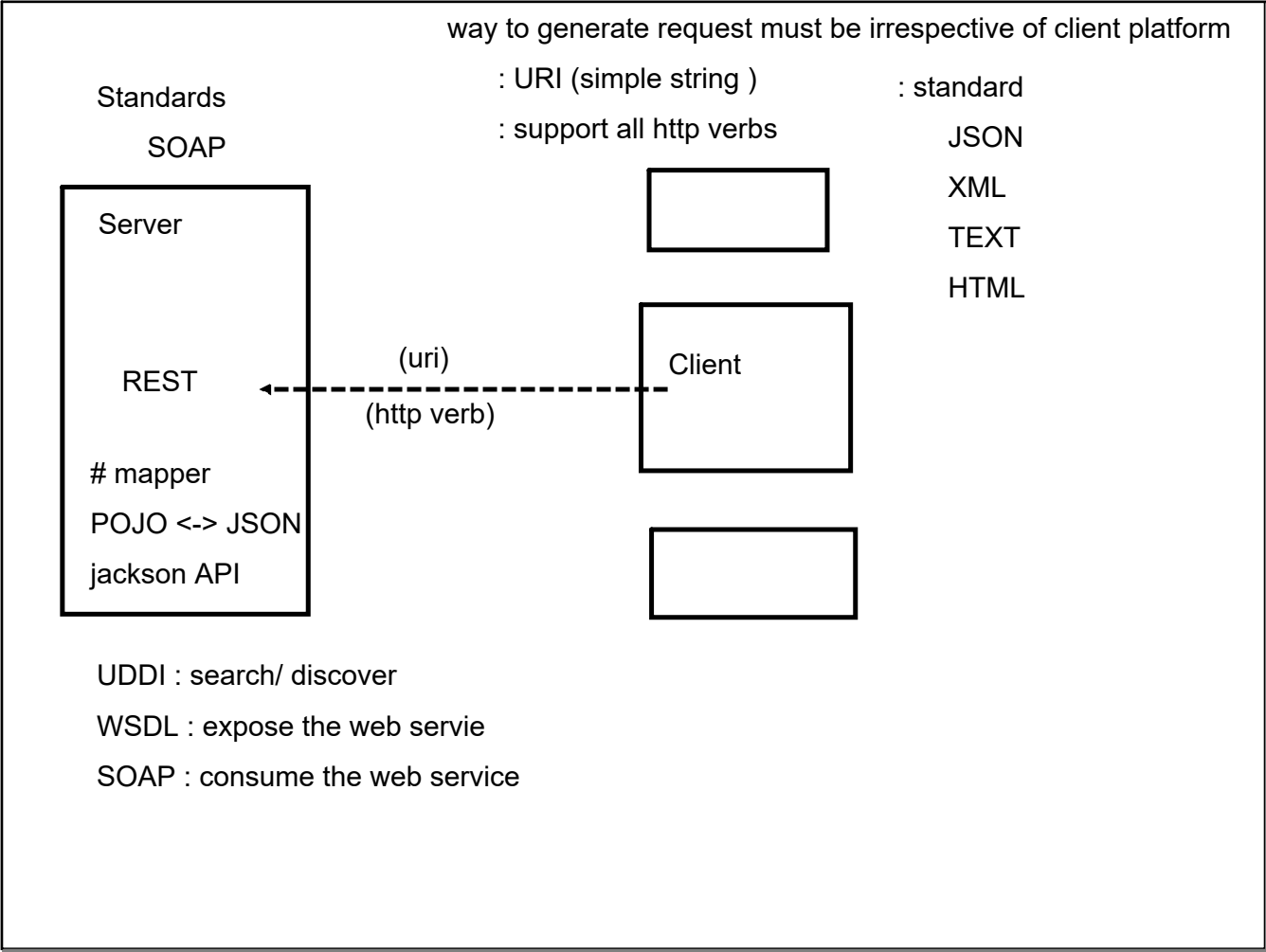
View Templates

Jsp-jstl

Thymeleaf

Mustache

FreeMArker

Tile

Velocity

Rest Based service

exposing server based info as Rest Service

state

IoT   Browser   Mobile

way to generate request must be irrespective of client platform

: URI (simple string)

: support all http verbs

Standards

: standard

SOAP

JSON

Server

XML

TEXT

HTML

(uri)

REST   Client

(http verb)

mapper

POJO <-> JSON

jackson API

UDDI : search/ discover

WSDL : expose the web servie

SOAP : consume the web service

@RestController : interconversion take care of

client intention

GET : data retrieval

POST : add new data

PUT : edition

DELETE : delete

Student /student

/getAll

Employee /employee

/getAll

@RestController : auto needs mapper in classpath (jackson)
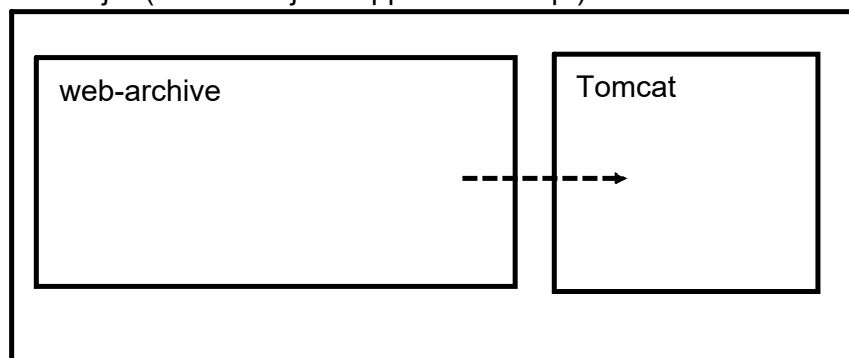
@Controller

  add mapper to classpath

  produce   1.launch tomcat
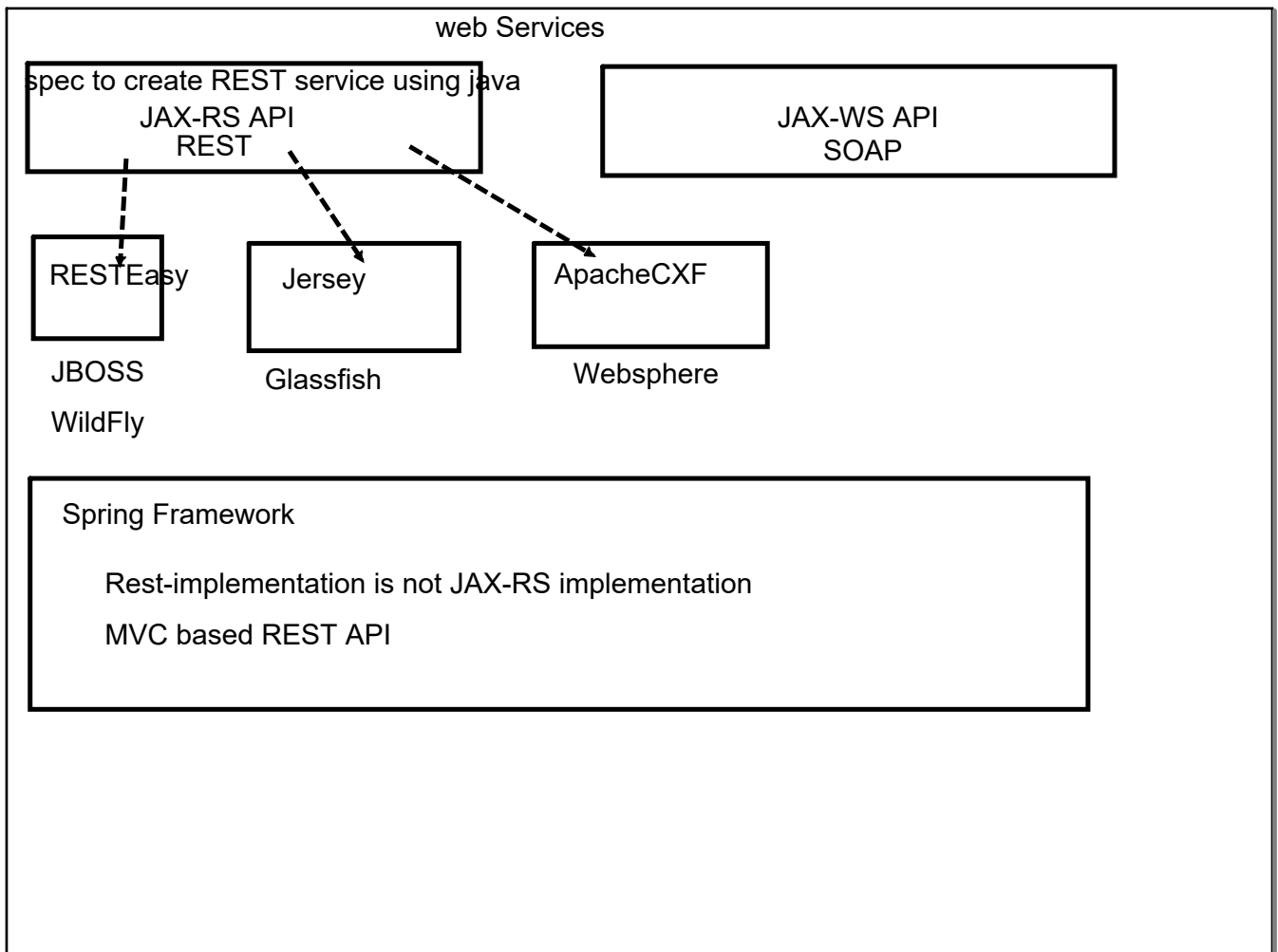
  consume   2. deploy web-archive over
                                                                               tomcat
  @ResponseBody : content to be shared directly to client

    jar (additional java app build on top)

web-archive	Tomcat

  dev-tools: live reloading of server

web Services

spec to create REST service using java
JAX-RS API
REST

JAX-WS API
SOAP

RESTEasy

Jersey

ApacheCXF

JBOSS

Glassfish

Websphere

WildFly

Spring Framework

Rest-implementation is not JAX-RS implementation

MVC based REST API

actuator : exposes rest endpoint

Microservice architecture implements

Dividing a single large sized monolith application into

multiple smaller (independent) application

 microservices : responsible to expose a particular service

   DataDriven/Rest based

   Stateless

   Service Oriented Architecture : SOA :

   Microservice : + technology/approach/design pattern

Monolith issues

involve light wight VS for deploying service components

Deployment :

Multi-Technology service component

DB : ideally must be using independent DB

Scaling : individual service comp

Robust in implementation

Design Guideline : MS (12 factor)

Design Pattern

Lightweight : concern/runtimes/data exchanging

Reactive : highly concurrent/longer processing

Stateless: scale better

Atomic : core design principle

Externalized config : config server

Consistent : style

Resilient : eliminate bottleneck

Good Citizens : expose usage statistics

well versioned :

Design Pattern:

    Decompositions :

a) business capabilities

    business-oriented rather than technical

b) sub-domain (technical)

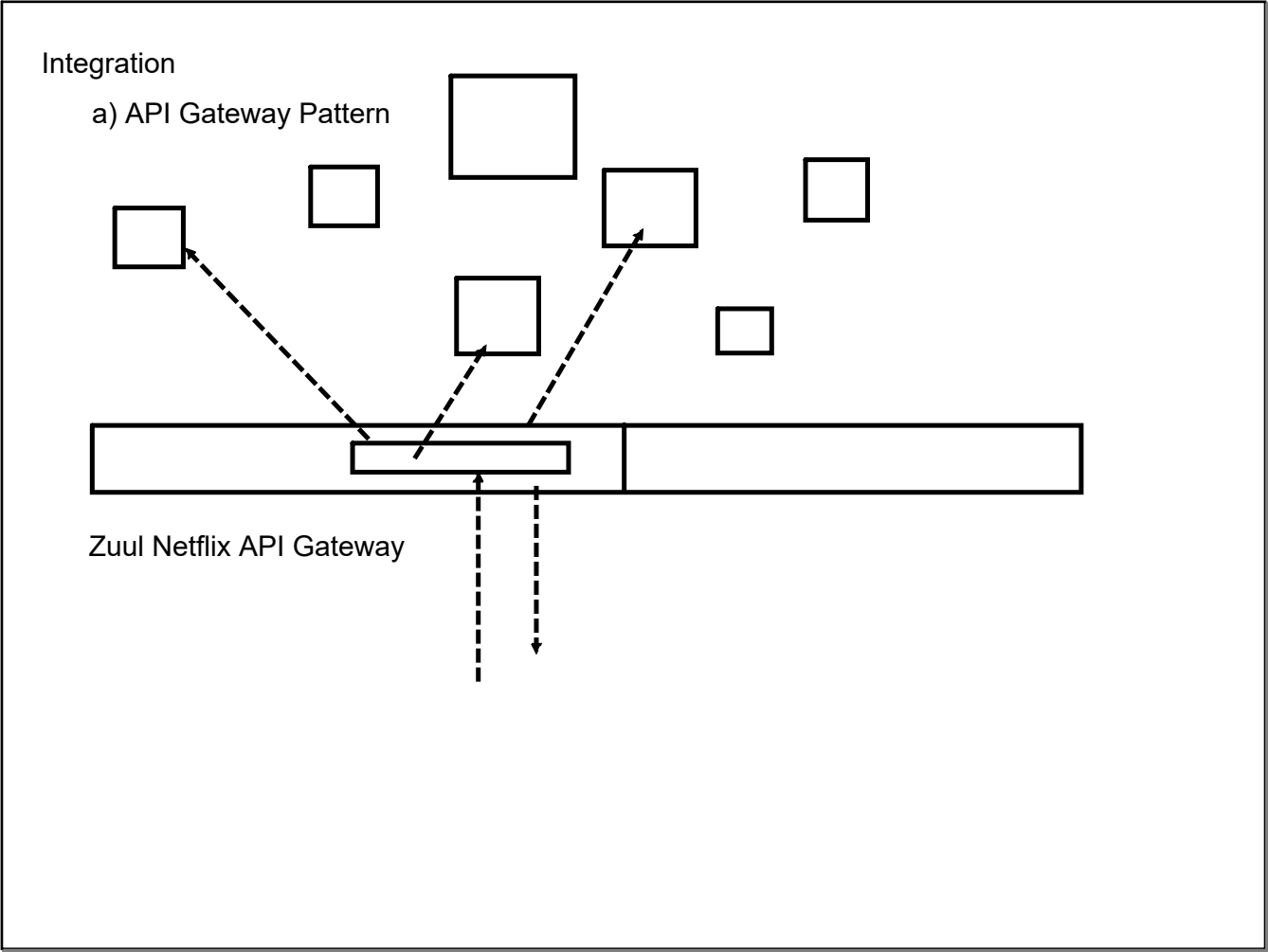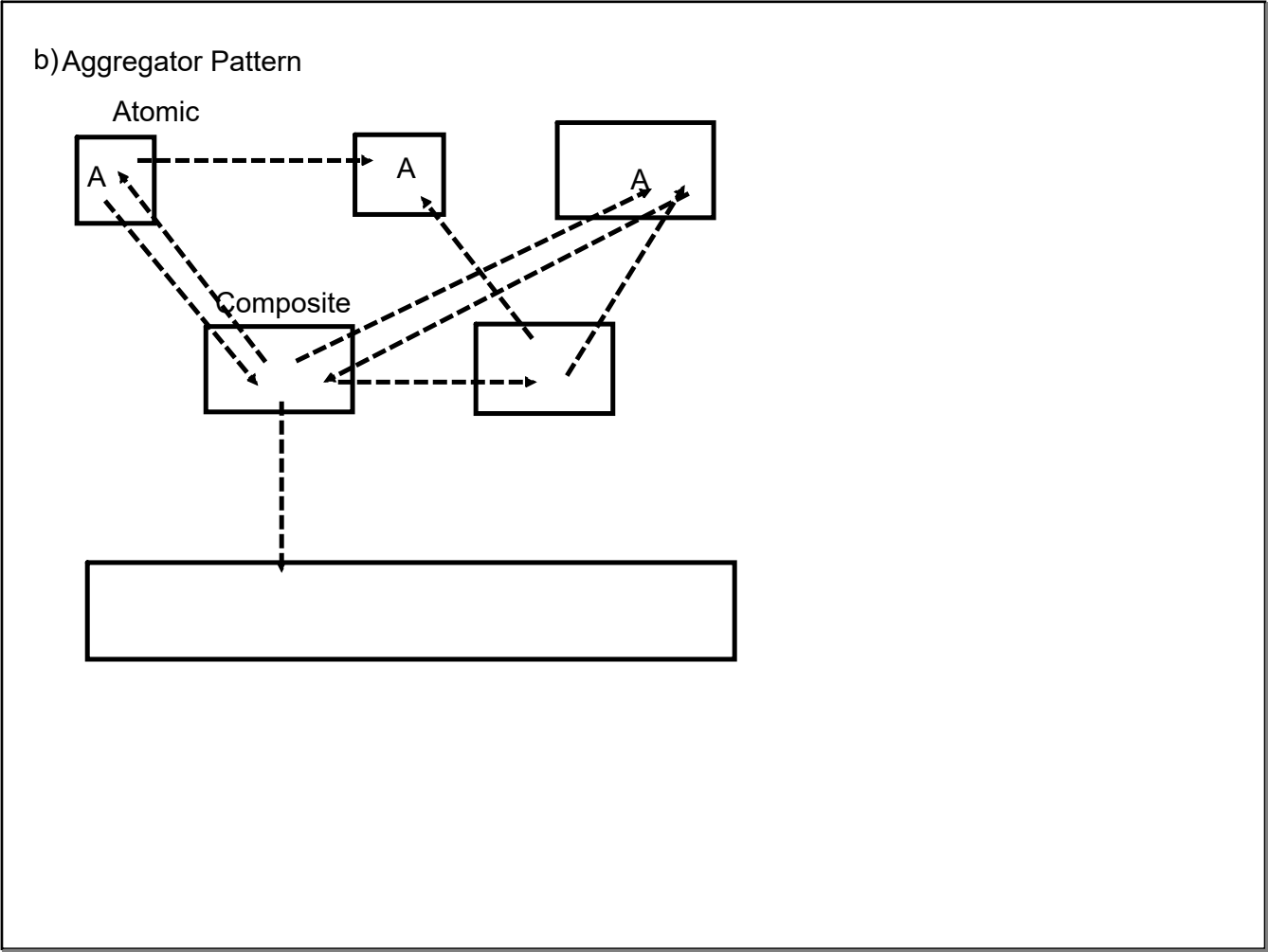   domain class (parent/God classes)

    DDD : bounded context

sub-domains : BC with parent model

c) Strangler patterns

    brownfield : converting monolith into MS

    refactoring smaller req...

Integration

a) API Gateway Pattern

Zuul Netflix API Gateway

b) Aggregator Pattern

Atomic

A

A

A

Composite

c) Client-side UI Composition
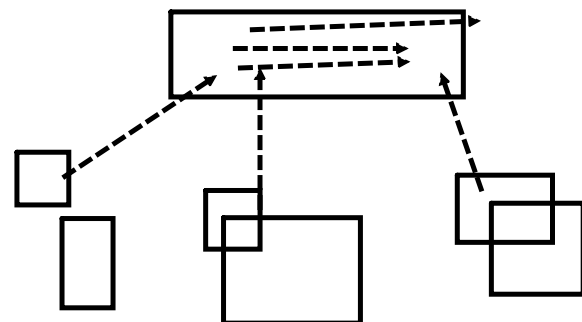
    SPA : UI compositions

DataBase :

    a) Ideal Pattern : Database Per Service

    b) Shared Database :

       2-3 MS (DDD)
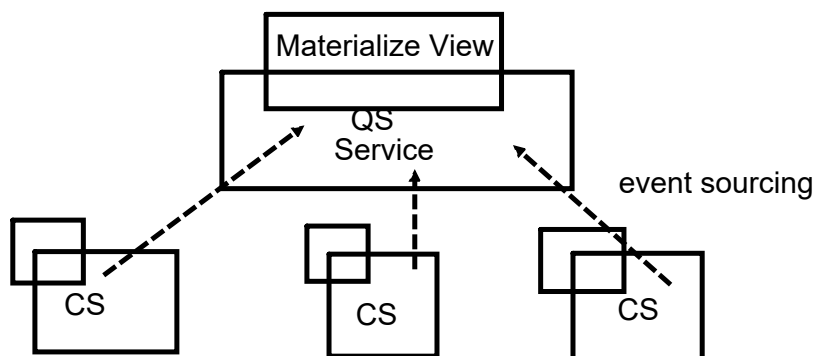
c) CQRS

Command Query Responsibility Segregation

Command Side (create/update/delete)

Query Side (retreive)

Materialize View

QS
Service

event sourcing

CS

CS

CS

SAGA PAtterns:

saga enabled   saga enabled   saga enabled

MS   MS   MS

info

event bus   :compensation system

Kafka

Framework ::

 Choreography :

  Axon :

  Orchestration :
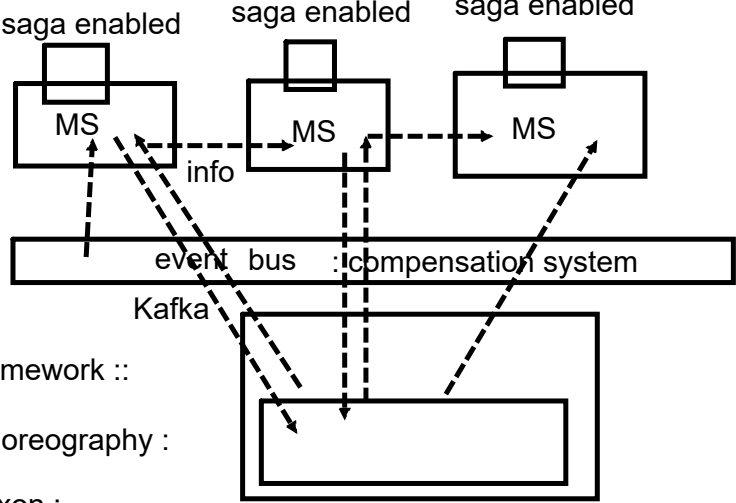
Observability PAttern

    a) Log Aggregation:

        Centralized Logging pattern in place

    track the log on request basis,

    search

    analysis

    triggers alert

PCF : Pivotal Cloud Foundary

    AWS Cloud Watch

    b) Performance based

        Centralized Metric service

            push/pull model

=>NewRelics

=>Prometheus

c) Distributed Tracing

   system to track a request end-to-end

request id

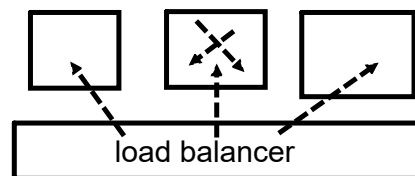Zipkin Server

Spring Cloud Slueth

load balancer

 d) Health Check

   actuators /health :

   Ribbon

Cross-Cutting Concerns

a) External Configuration

Spring Cloud Config Server

b) Service Discovery Pattern

all service shall register with registry system

Netflix Eureka Server

AWS ALB

x   x   x

| 1 | | 2 | | 3 |

Discovery Server (Server Side)

Ribbon (Client Side)

y
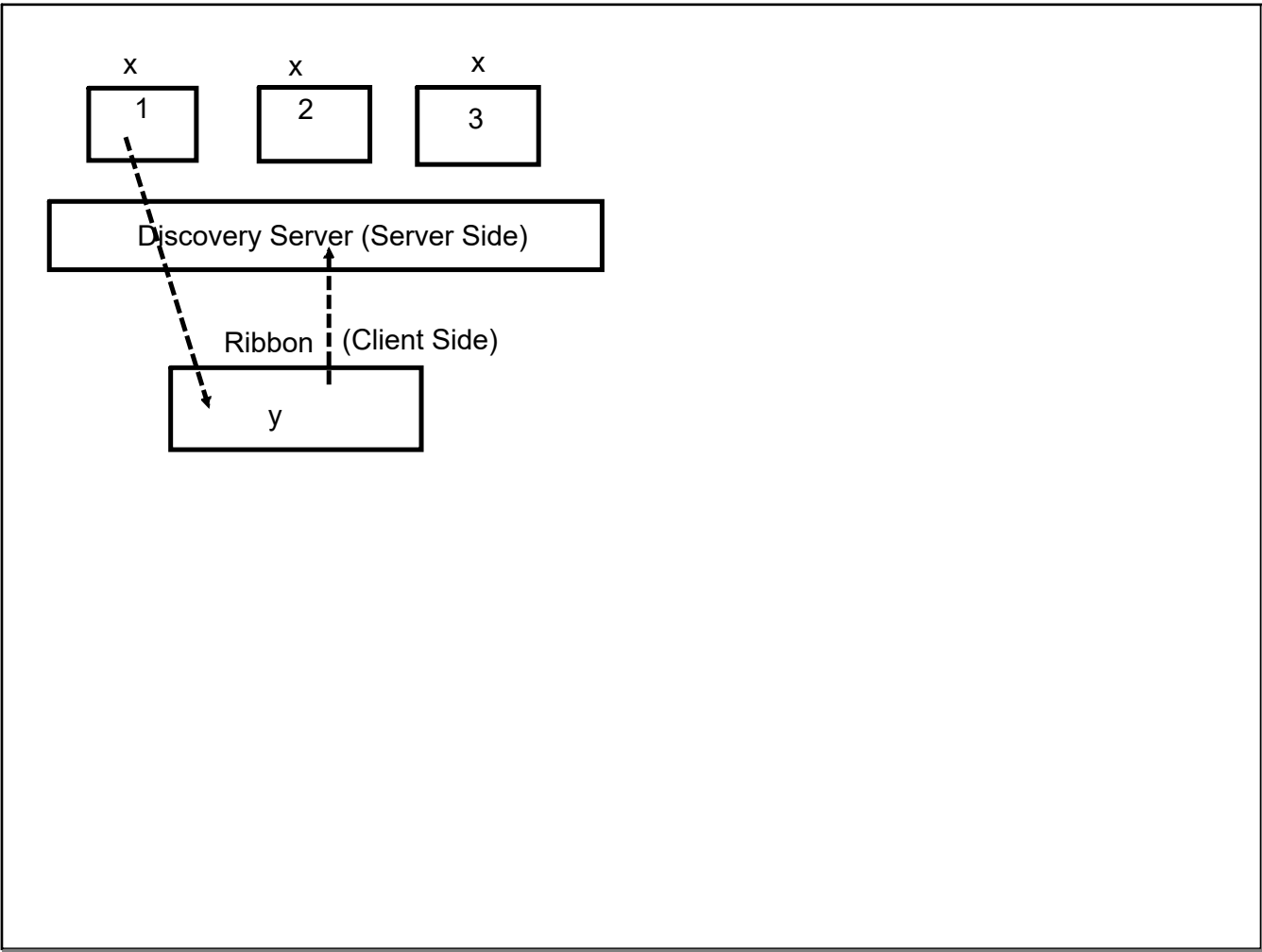
c) Circuit Breaker Pattern

threshold

default response

keep on trying

Netflix Hystrix

10 sec

5

fallback

mysql

Discovery

Angular

React

User Microservice

Registration / Login

Ribbon

RestTemplate

sql

feign client

Post + Comment

Possibilty

to divide

API

Security

Admin Microservice

59
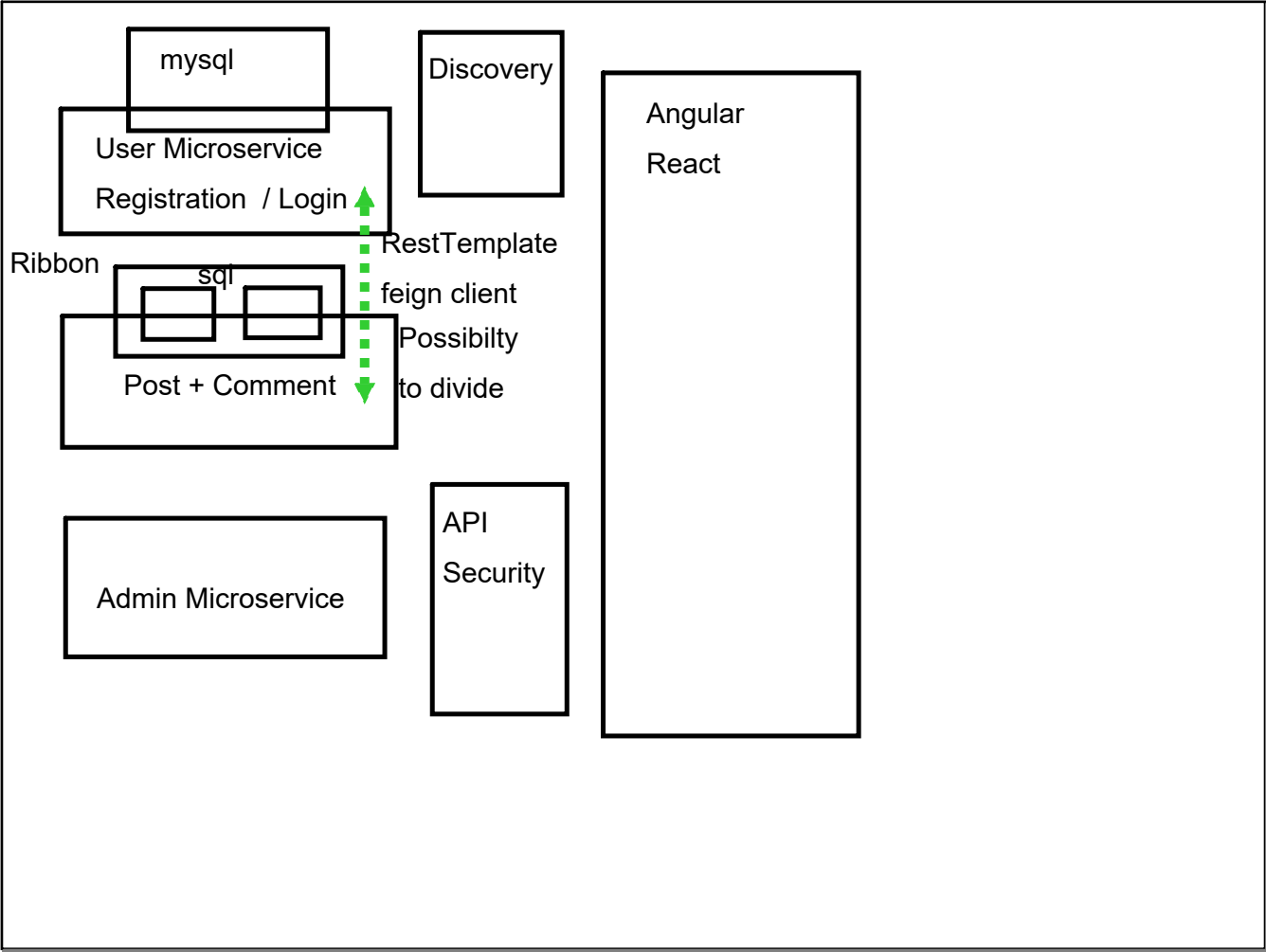
Spring Cloud :

Controllers

Service

dao/repository

entity : POJO <----> DB schema

DTO : POJO

Validation

Exception

Spring Data JPA

inbuilt repository (generic) with implementation

that can be customized/extend

CRUD
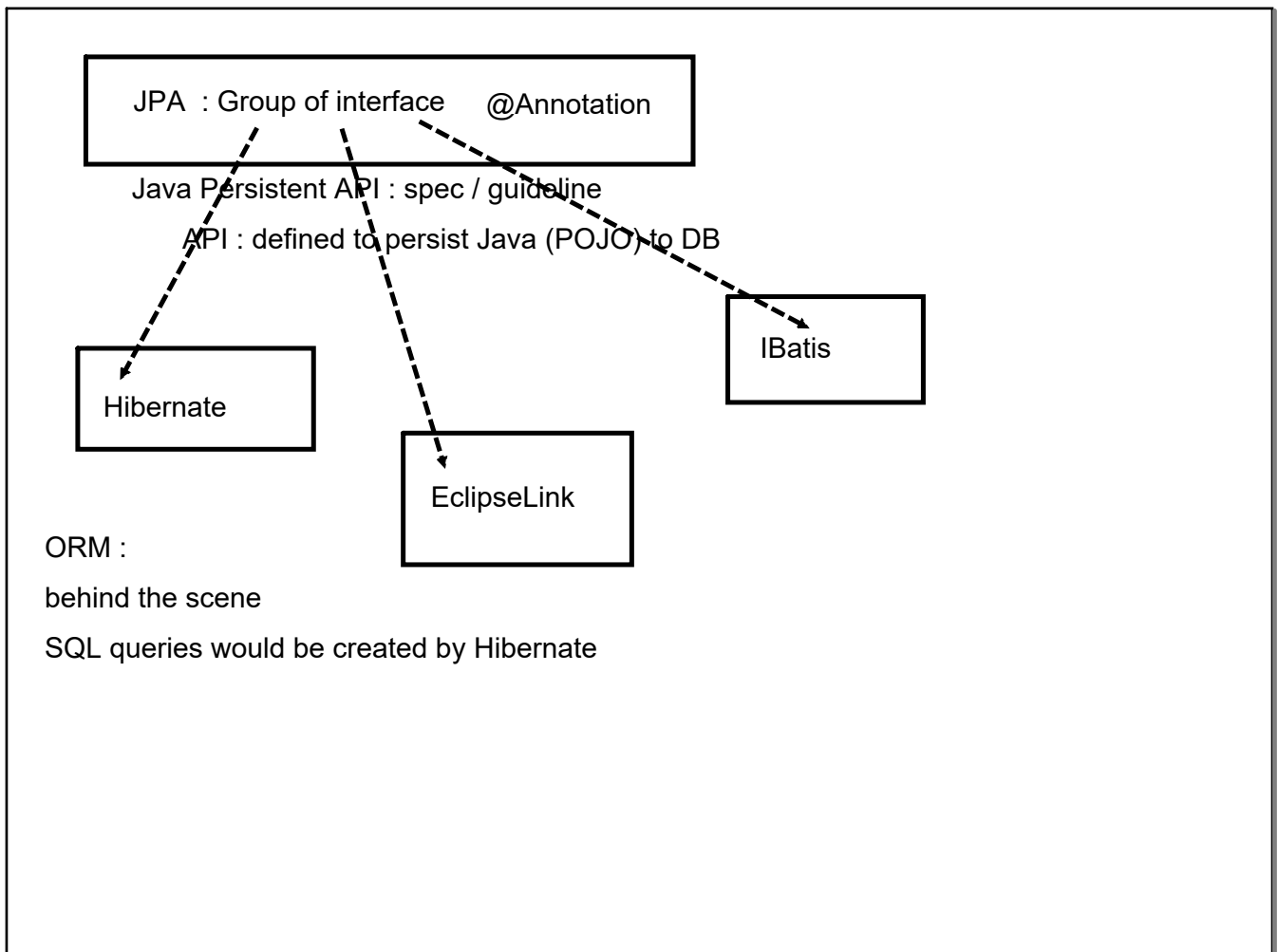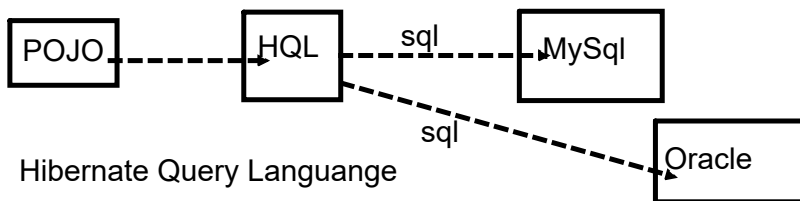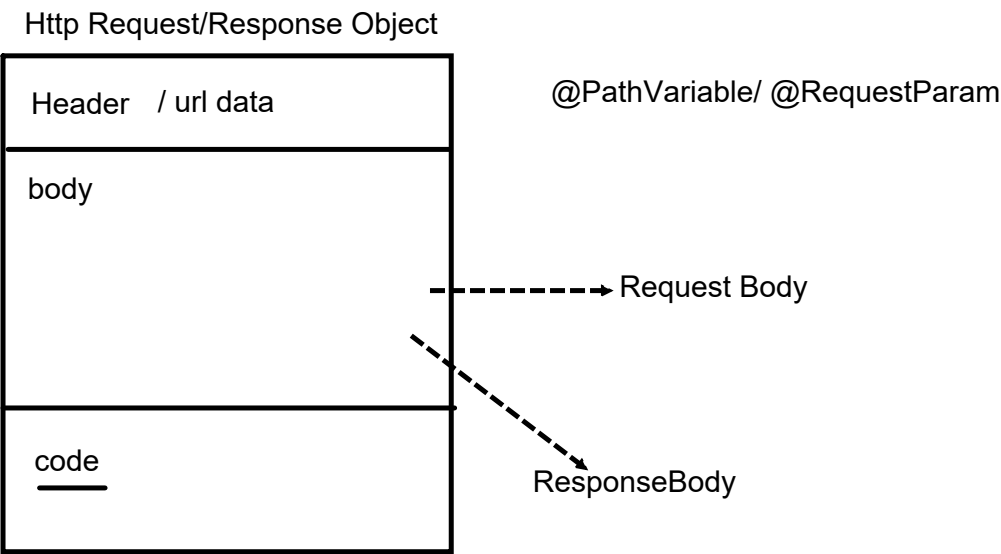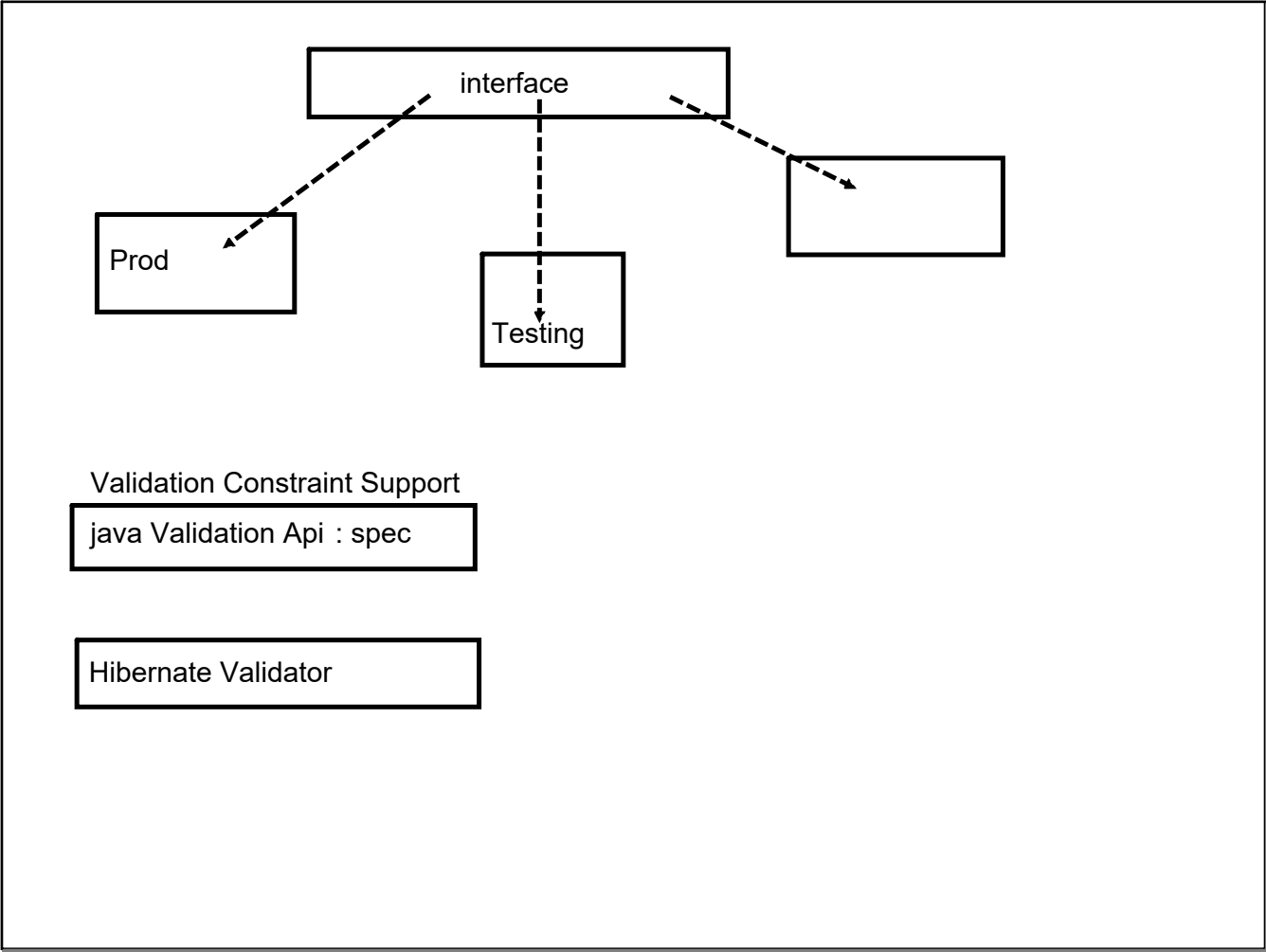
select * from <table-name>

=>table

=>datatype of primary key

Need to configure properties file

for DB connection

: MySql-Driver : dependency

JPA : Group of interface   @Annotation

Java Persistent API : spec / guideline

API : defined to persist Java (POJO) to DB

IBatis

Hibernate

EclipseLink

ORM :

behind the scene

SQL queries would be created by Hibernate

POJO - - - - → HQL — sql - - - - → MySql

HQL — sql - - - - → Oracle

Hibernate Query Languange

Need to specify the dialect

c3p0 : dependency

Http Request/Response Object

Header   / url data   @PathVariable/ @RequestParam

body

- - - - - - → Request Body

ResponseBody

code

All Rest Endpoints must have provision to
customized the http Response object

interface

Prod

Testing

Validation Constraint Support

java Validation Api : spec

Hibernate Validator

Client Expecting : UserDetailDto (Success status)

Exception : UserExceptionDto (Failed status): throw an exception on client end of type mismatch

Server shall respond with appropriate status code

REst Client have provisions to check the status code

Adding a new data : instance/info about newly added data

Updating the data : instance/info about update data

Deleting the record : instance/info about deleted data

DTO - entity   DTO ->

4 + 3 ---> DB

Client : 7 fld (primary

When a new Post is added :   post-micro   user-micro

Info about the newly added post

    id

    title,tags,post

    userId

    name, emailId

         DTO

client

Multi-Module

common

post-micro

API-Gateway

Details of that user (user-micro)

List of Posts

Eureka-client
post-micro

user-micro

Ribbon (Client)

AS soon as we run MS,

they will start looking for DS (8761)

(Default)

9091

9092

names

Discovery Server/ Naming Server   register

Eureka (Netflix)

Spring boot + Eureka API

Two tables

1. User credential

2. Roles

User-Credentials

table ("users")

    # username : String

    # password : String

    # enabled : boolean

Roles

    table ("authorities")

username : String

authority : String

password : encrypted form

Spring security supports multiple encryption

eg:

Plain-Text   Bcrypt (one way)

 abc{noop}   {bcrypt}2afdhfldron98

Roles:

    Manager ~ Role_Manager

Token
/url

API-GATEWAY

Security Filter

config

/url

Auth

Registration

First

abc

Bcrypt

{noop}
2afdlf98{bcrypt}

Token

Security
Config

First

abc → bcrypt

3 core elements

HTML : Structure

CSS : Presentation

JavaScript : Behavior

HTML-5

   Validations

   Drag n Drop

   Semantic Tags

   Web Workers

   Offline functionlity

   Geolocation

New Semantic Tag (Backward Compatible)

  # purpose full (specific to req)

=> container

=> attributes -- Form based extention

Smooth Renderring (outline algo)

more compatible to search algo

in sync with Assisstive Tech

Standardized Error handling algo : Developers (Debug)

images/audio/videos : third party plugins : HTML5 tags + API (control)

Built-in APIs

traditional:

    <p>, <span>, <div>

article

section

aside

header

footer

Start

A block of flow content
(not inline phrasing content)

Is it a major
navigation block?

Does it make
sense on its own?

*e.g. in a
feed reader*

Is it required
to understand the
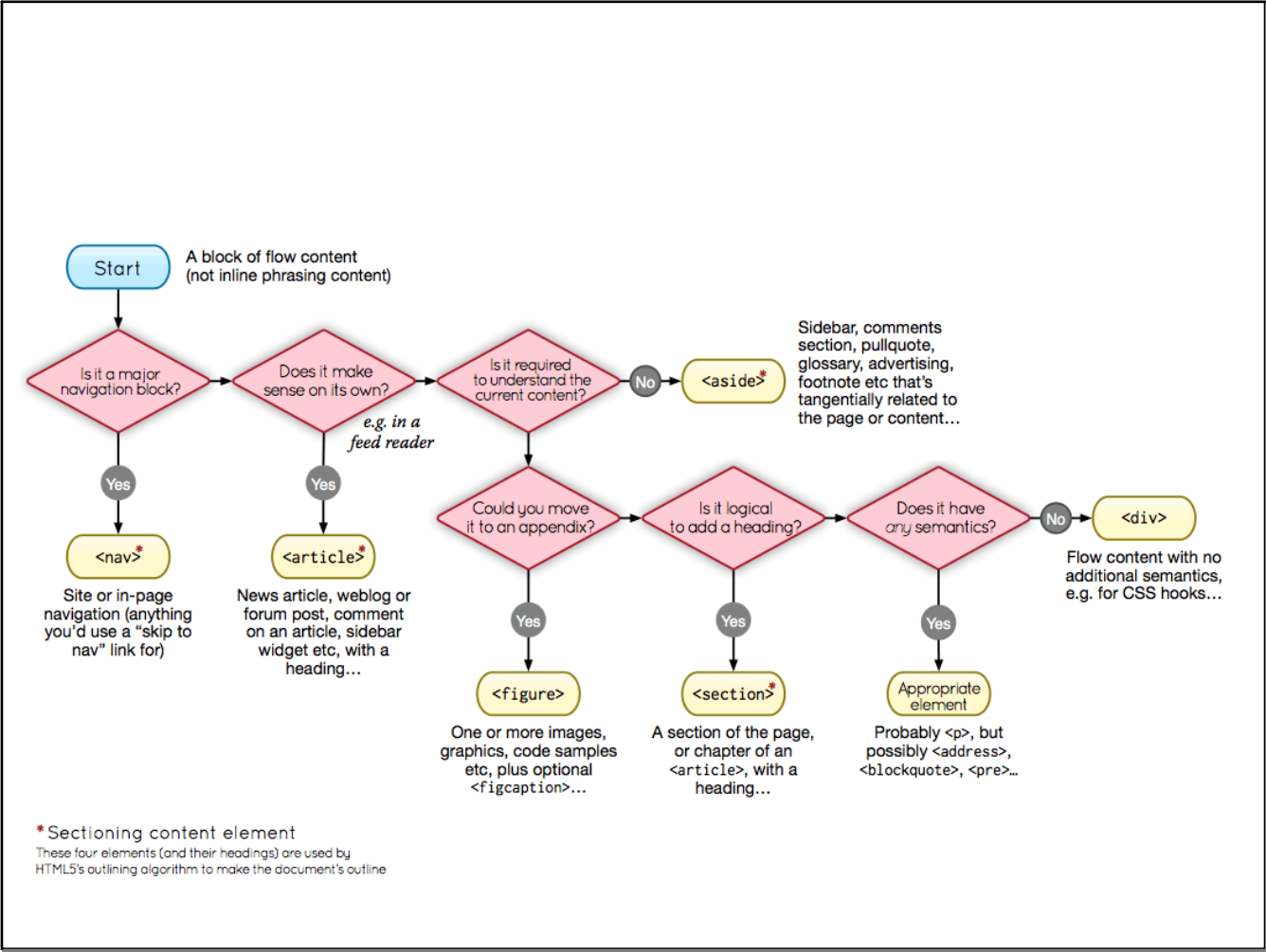current content?

No

<aside>

Sidebar, comments
section, pullquote,
glossary, advertising,
footnote etc that's
tangentially related to
the page or content…

Yes

<nav>

Site or in-page
navigation (anything
you'd use a "skip to
nav" link for)

Yes

<article>

News article, weblog or
forum post, comment
on an article, sidebar
widget etc, with a
heading…

Could you move
it to an appendix?

Is it logical
to add a heading?

Does it have
any semantics?

No

<div>

Flow content with no
additional semantics,
e.g. for CSS hooks…

Yes

<figure>

One or more images,
graphics, code samples
etc, plus optional
<figcaption>…

Yes

<section>

A section of the page,
or chapter of an
<article>, with a
heading…

Yes

Appropriate
element

Probably <p>, but
possibly <address>,
<blockquote>, <pre>…

* Sectioning content element
These four elements (and their headings) are used by
HTML5's outlining algorithm to make the document's outline

Html form :

    # specialized form inputs

    # validation : required/pattern

    # special att : custom behavior of form

```
<form>
        _____
        _____
        _____
        _____   [S]
      _____
      _____
</form>
```

_____

   _____

Canvas API

DOM Tree managed by the browser

Html component(Tag) : JS - object

User Interaction : presentation : CSS

Cascade style sheet

Stylesheet : : set of rules 'presented'

Cascade : set of rules : resolve the conflict of multiple ss applied on a element

Specificity

controlling over where to apply the style

CSS rule :

CSS Selector

CSS declaration

selector {

property : value

}

selector : css rule would be applied to which HTML elem

Selector

  Type (most varied : wide spectrum : which type HTML element)

  ID

  class

eg :

p{

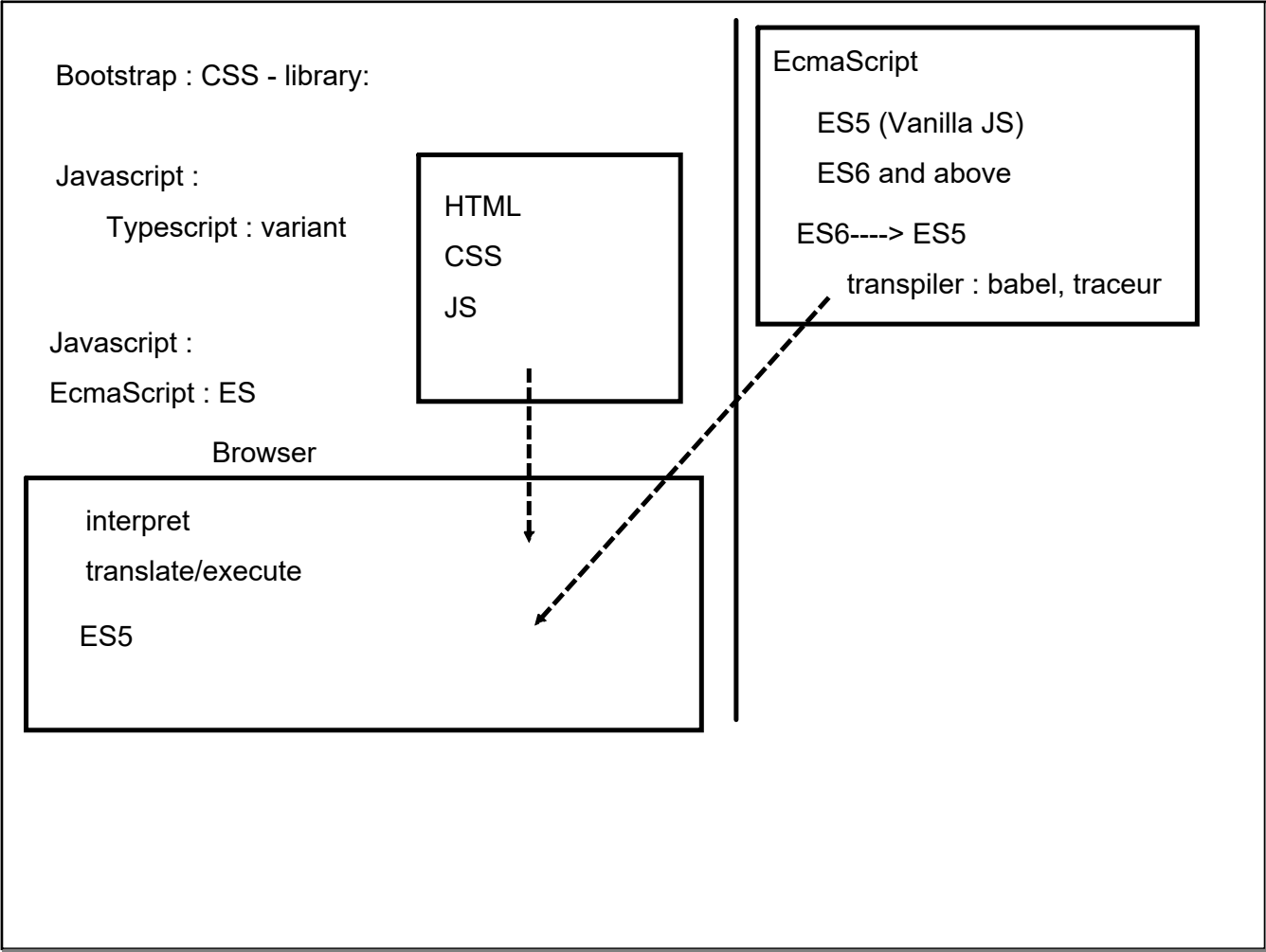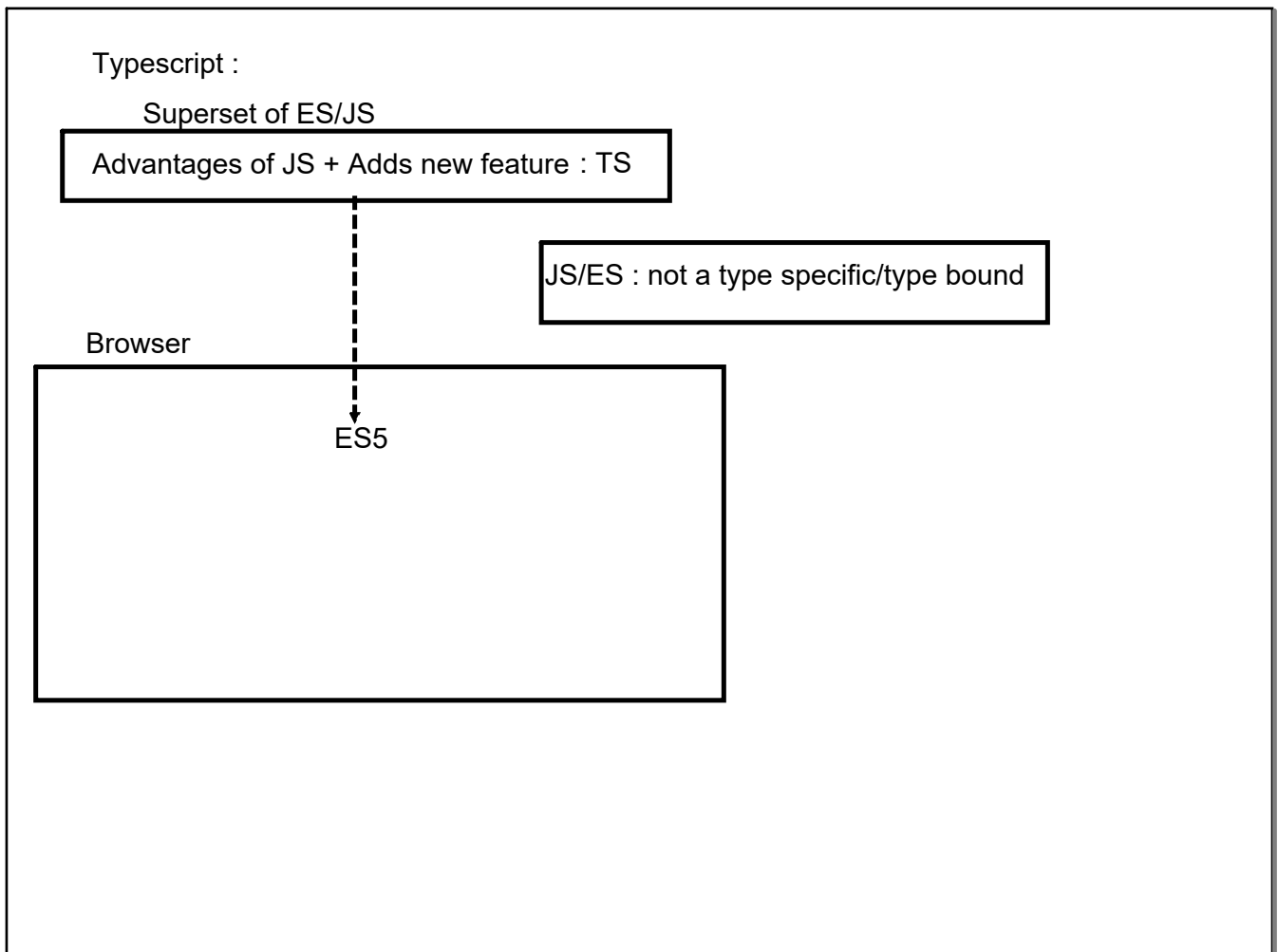 -------------

  ----

}

ID : very specific

#canvastest{

}

class

  .mclass{

  }

Bootstrap : CSS - library:

Javascript :

   Typescript : variant

Javascript :

EcmaScript : ES

HTML
CSS
JS

EcmaScript

   ES5 (Vanilla JS)

   ES6 and above

ES6----> ES5

   transpiler : babel, traceur

   Browser

 interpret

 translate/execute

 ES5

Typescript :

Superset of ES/JS

Advantages of JS + Adds new feature : TS

JS/ES : not a type specific/type bound

Browser

ES5

```
                    Javascript
         function add(num1, num2){
            // validation check
            return num1 + num2;
         }
```

call : add(20, 30); // arithmatic addition

   : add('hello', 'world'); // string concatenation

Unwanted behavior at runtime

Typescript :

Named Types...

NextGen JS features

NonJS features like Interface/Generics

Decorators (Meta-Programming)

More Config options

Transpiler : Typescript compiler

Javascript based resource, managed way

management tool :

nodejs : npm : node package manager

yarn

NodeJs : installed + system path

(npm) : cli

NodeJs : Framework that allows to use JS for server side programming:

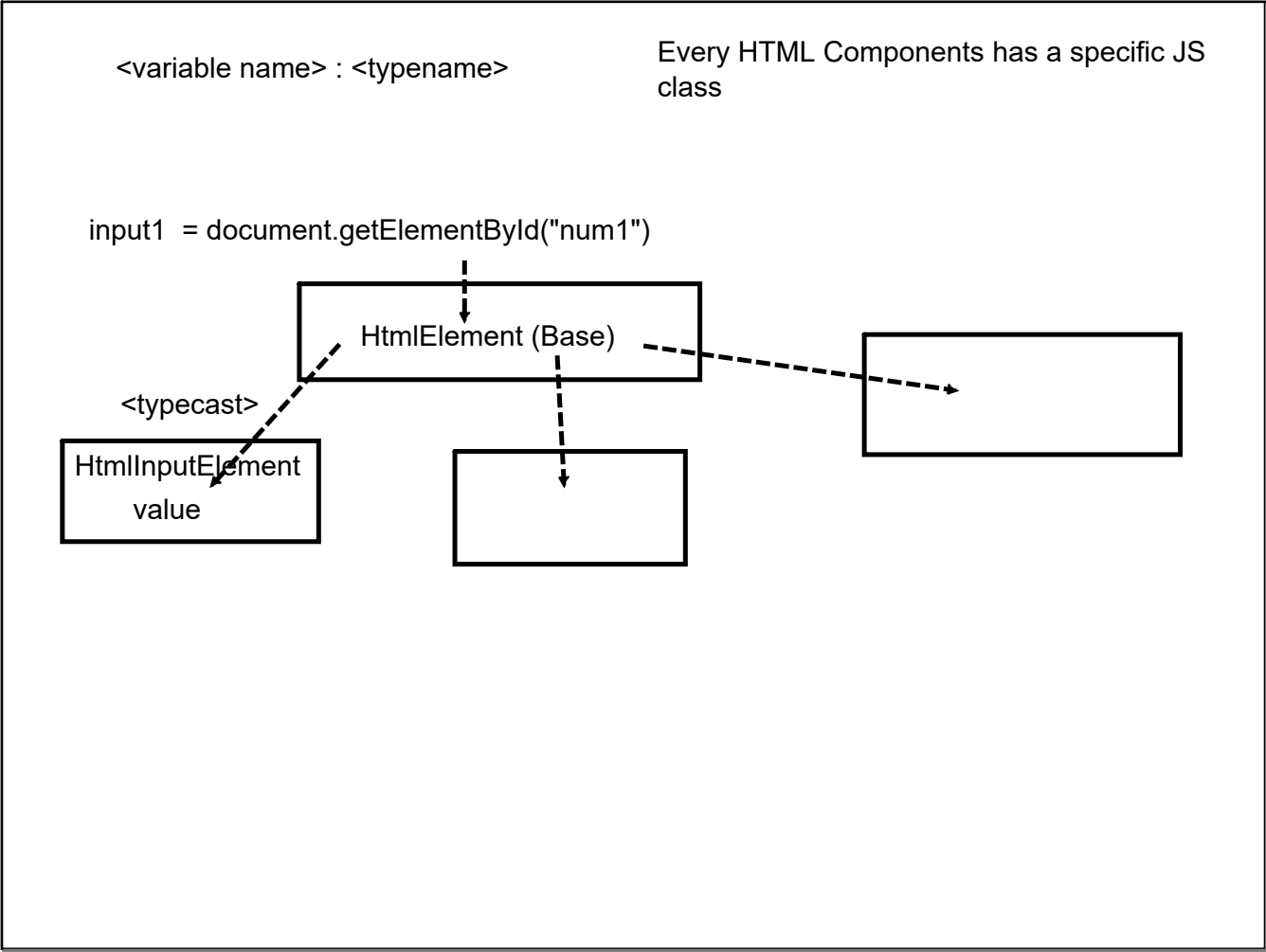non-blocking, asynchronous server implementation

npm : is a project management tool for JS related project management

Need to install typescript compiler

>npm install -g <tool> (global installation)

> npm install -g typescript

Typescript file must have ext : .ts

<variable name> : <typename>

Every HTML Components has a specific JS class

input1 = document.getElementById("num1")

HtmlElement (Base)

<typecast>

HtmlInputElement
value

var ~ ES6 : const / let

Core Types

> number : integer/fractions
>
> string : 'hello', "hello", \`hello\`
>
> boolean : true,false
>
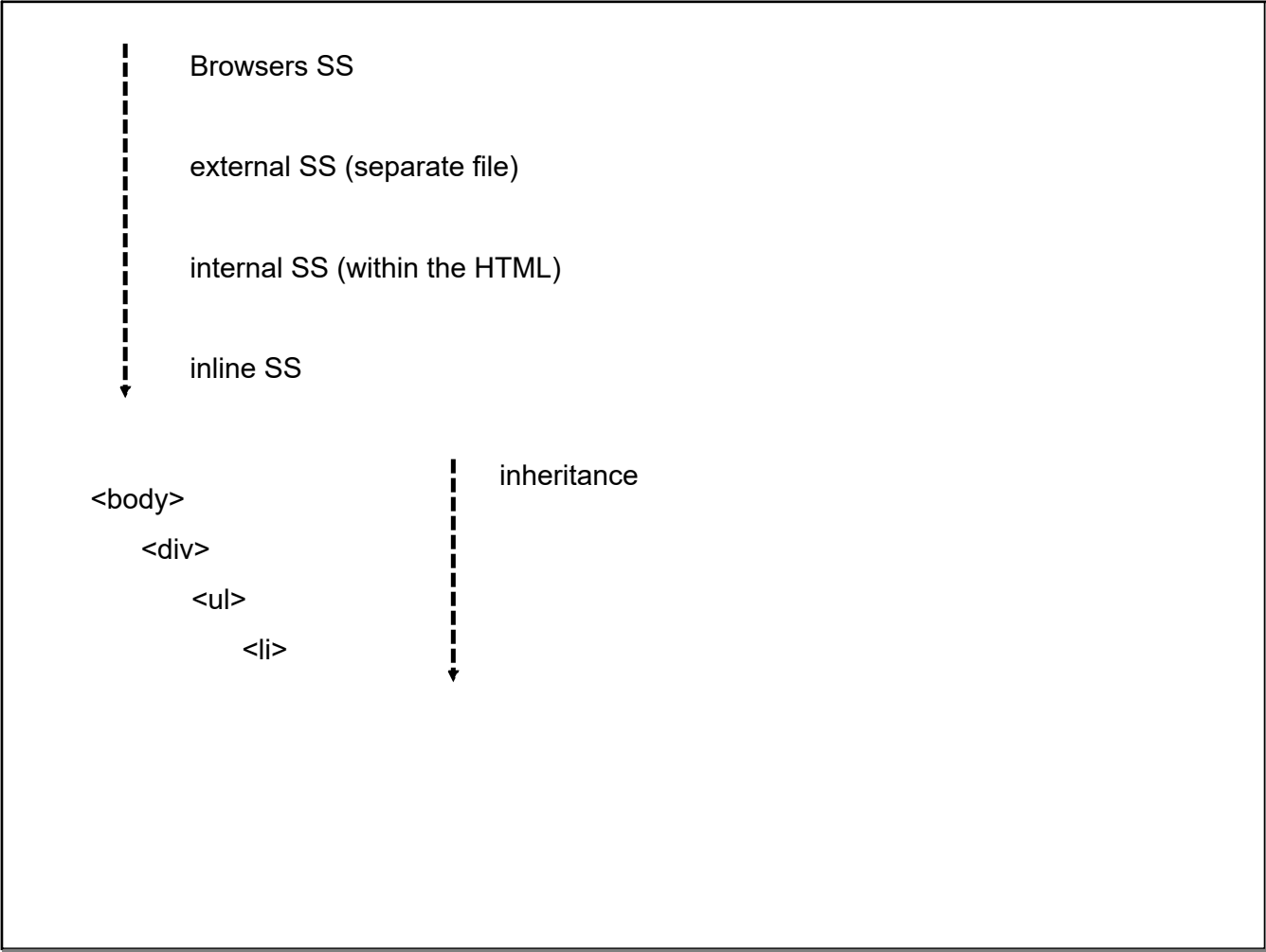> object : Javascript object (more type specific)Object Notation
>
> Array : JS has way to create array of heterogenous nature (TS : homogenous)
>
> Tuple : Fixed length : Type
>
> # Union : specify multiple types
>
> Enum : enumerated Datatype
>
> any : default JS type

Browsers SS

external SS (separate file)

internal SS (within the HTML)

inline SS

inheritance

```
<body>
    <div>
        <ul>
            <li>
```

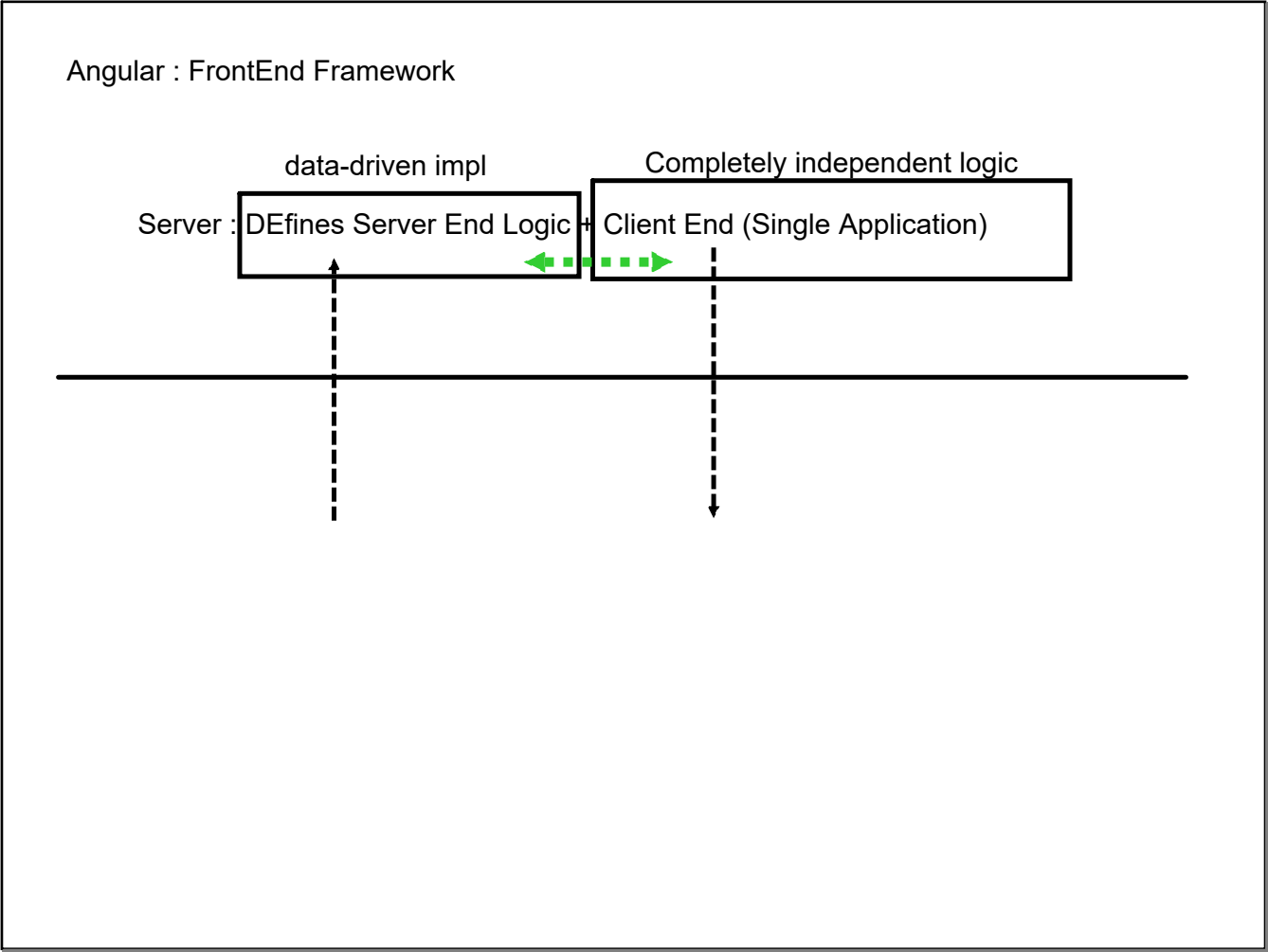Classes : high level way :

Closures :

    have global variable(memory retains across function calls) with local scope

 # static variables of C functions

Angular : FrontEnd Framework

data-driven impl   Completely independent logic

Server : DEfines Server End Logic + Client End (Single Application)

Loose coupling of Server Side (backend logic) and Client Side (Frontend logic)

1. Server Side is reusable

2. Client Side is also reusable (flexible)

3. More independent implementation

4. Load Distribution among client machine (renderring the dynamic web-pages : JS)
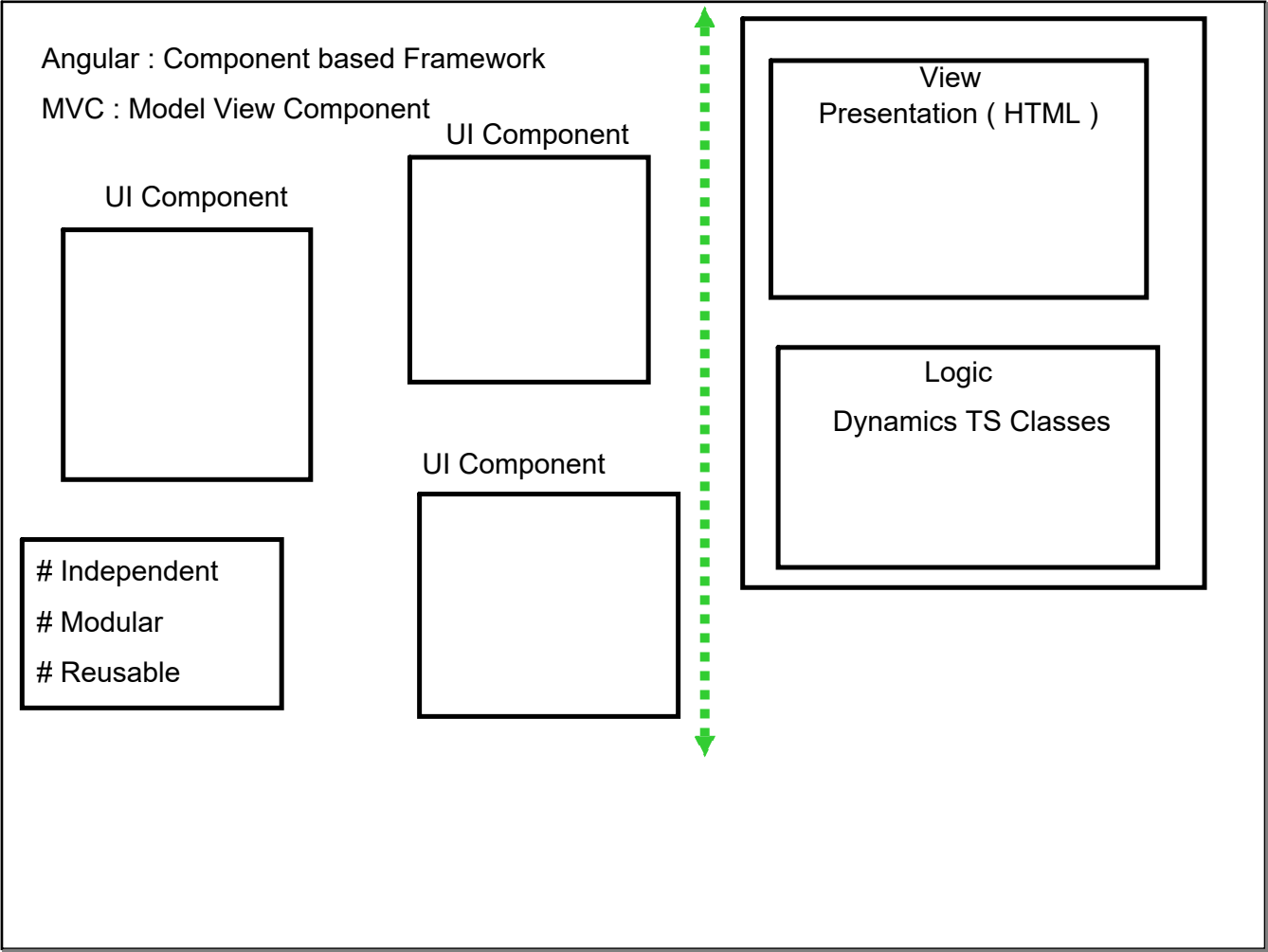
5. Client End Renderring can Highly customized

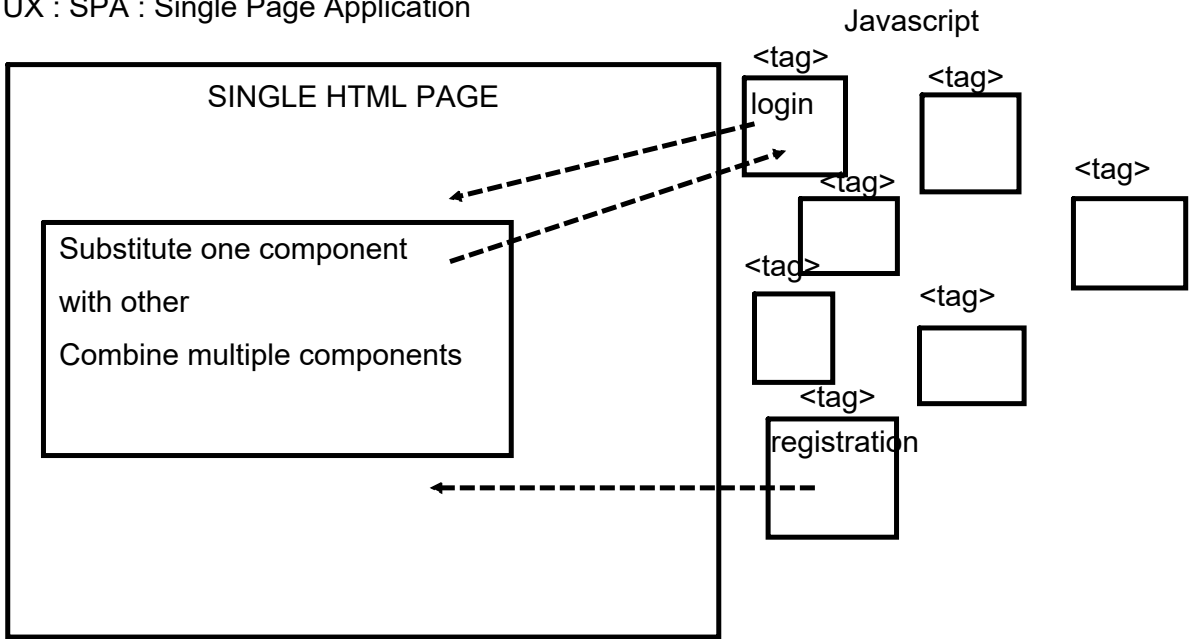Angular Framework

Complete Framework

Base Script : TS

Resources : Client Side JS Community Library

npm to manage angular application

Angular : Component based Framework

MVC : Model View Component

UI Component

UI Component

UI Component

Independent

Modular

Reusable

View
Presentation (HTML)

Logic

Dynamics TS Classes

Relationship : (logic + presentation) decide the output

UX : SPA : Single Page Application

Javascript

SINGLE HTML PAGE

<tag>

login

<tag>

<tag>

<tag>

<tag>

<tag>

<tag>

Substitute one component

with other

Combine multiple components

<tag>

registration

Angular/CLI Project needs to be installed

Download angular CLI/installed

(by default latest version)

> npm install -g @angular/cli

Angular CLI will expose angular specific command

> ng <option> (syntax)

> ng new   <project-name>

1. Complete folder/file structure required as Angular Framework project

2. Download default Angular lib

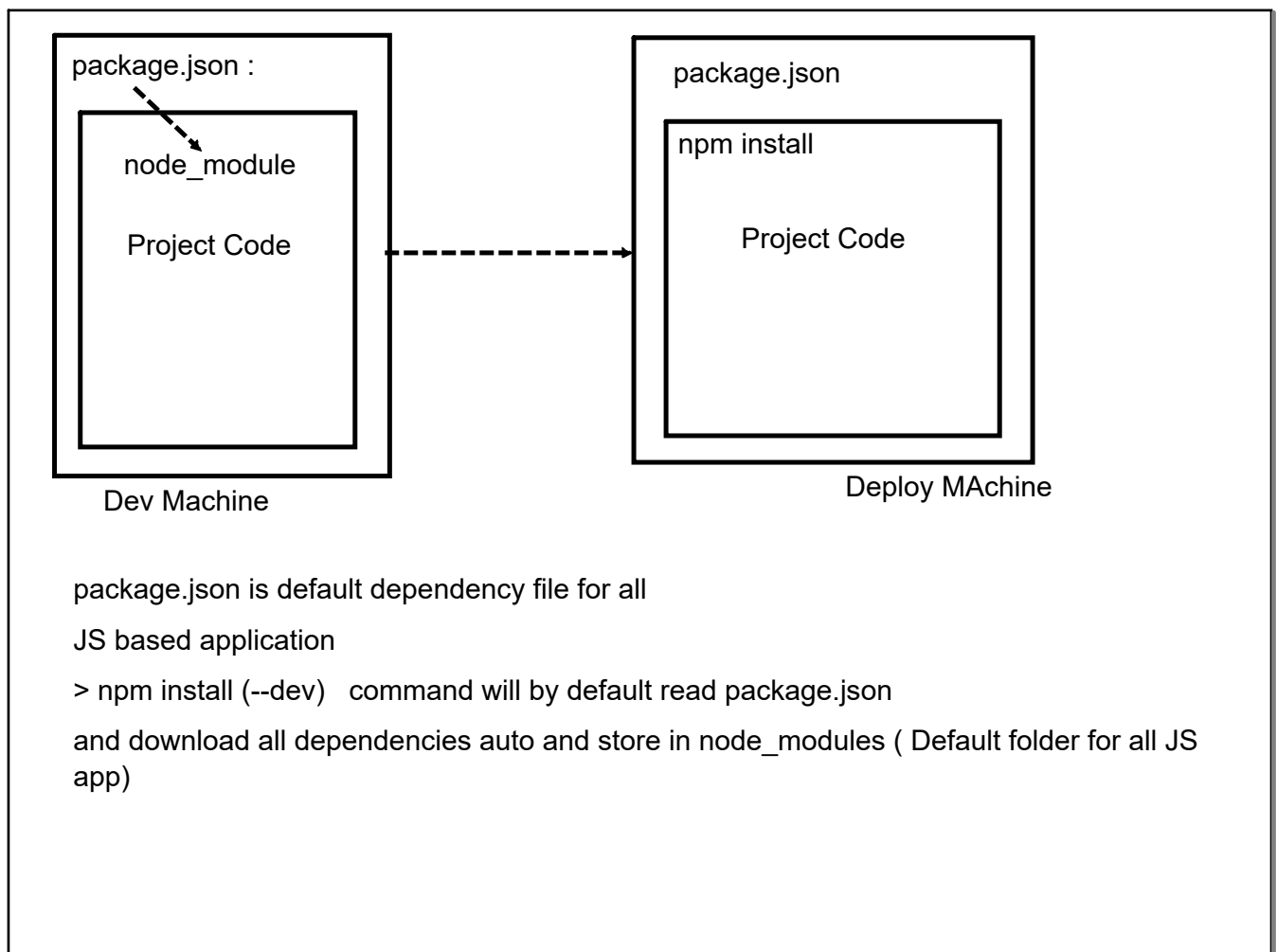> Add routing module (Y)

> Stylesheet : CSS(default)

Feature Set for Unit/Integration Testing and End-To-End Testing

1. Jasmine Framework : JS Testing Framework (Write Test case unit/integration + e2e)

2. Test Runner : Unit Test (Karma)

3. Test Runnner/Framework : End-To-End Testing (Protractor)

e2e : supposed to contain test cases/config related to End-To-End Testing

node_module : All lib are stored in this folder
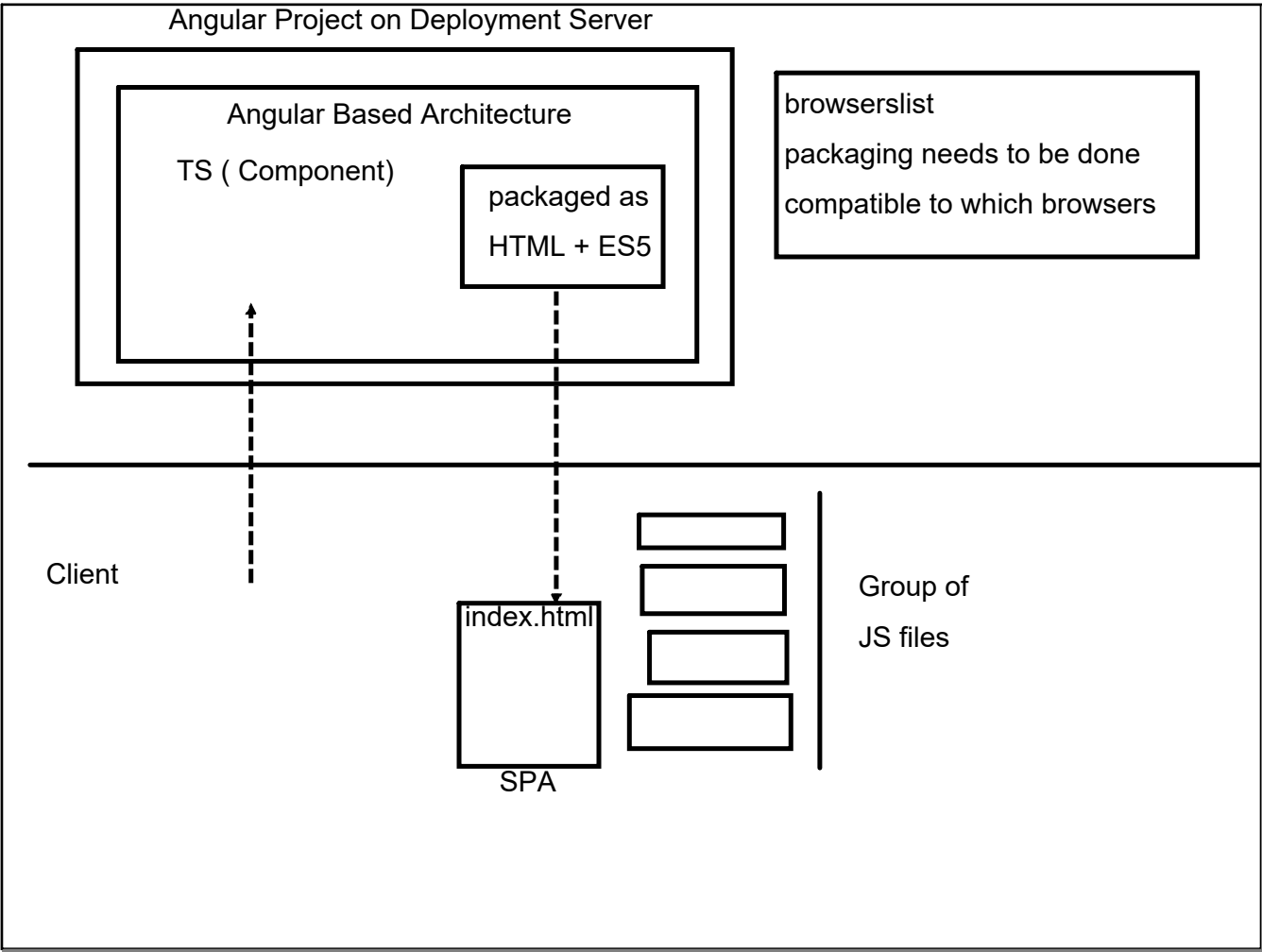
src : All Angular code goes here

package.json :

node_module

Project Code

Dev Machine

package.json

npm install

Project Code

Deploy MAchine

package.json is default dependency file for all

JS based application

> npm install (--dev)   command will by default read package.json

and download all dependencies auto and store in node_modules (Default folder for all JS app)

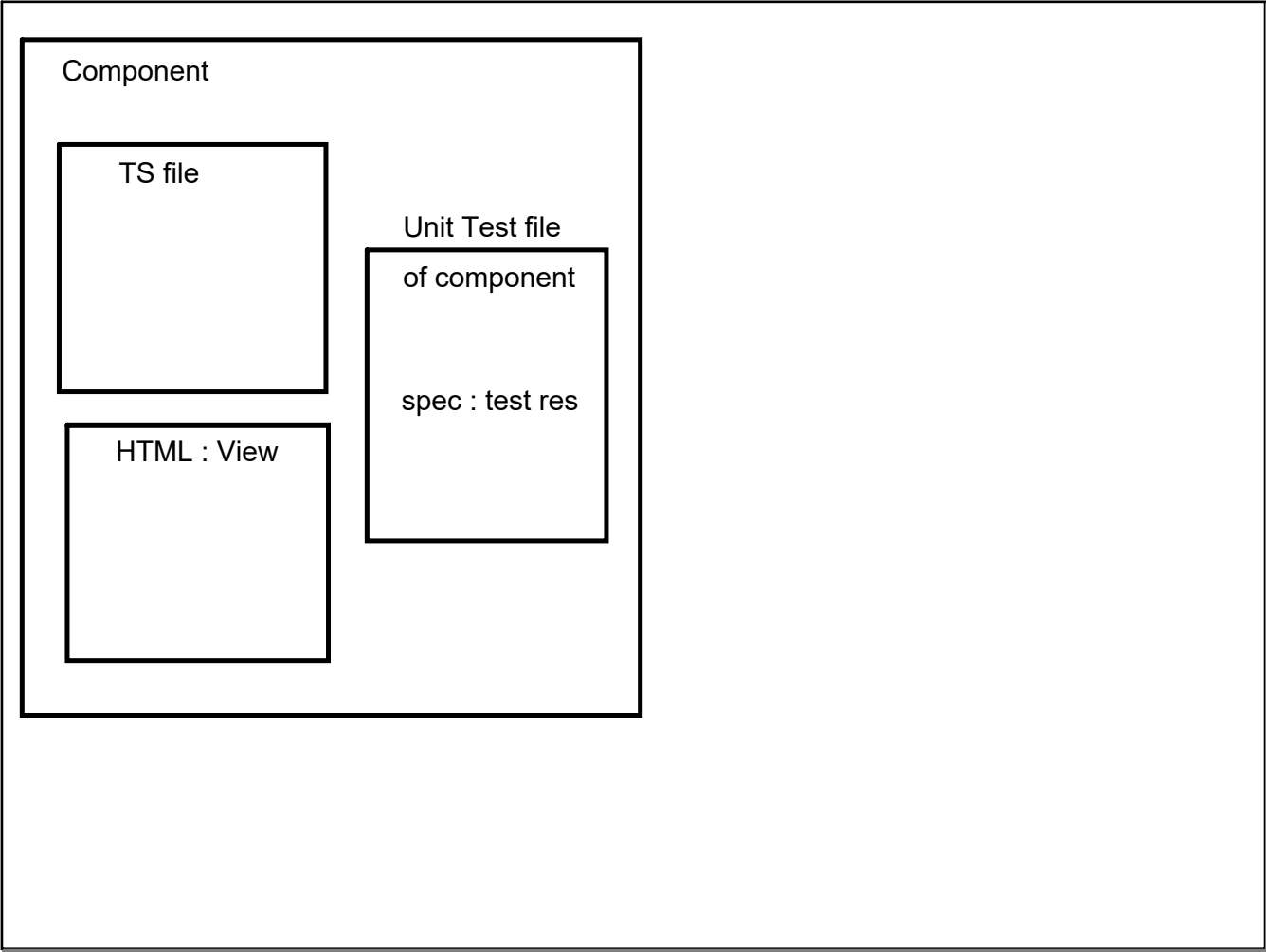Adding a new Dependency:

    1. add an entry in package.json

    2. npm install (download the dependency and add it to node_module)

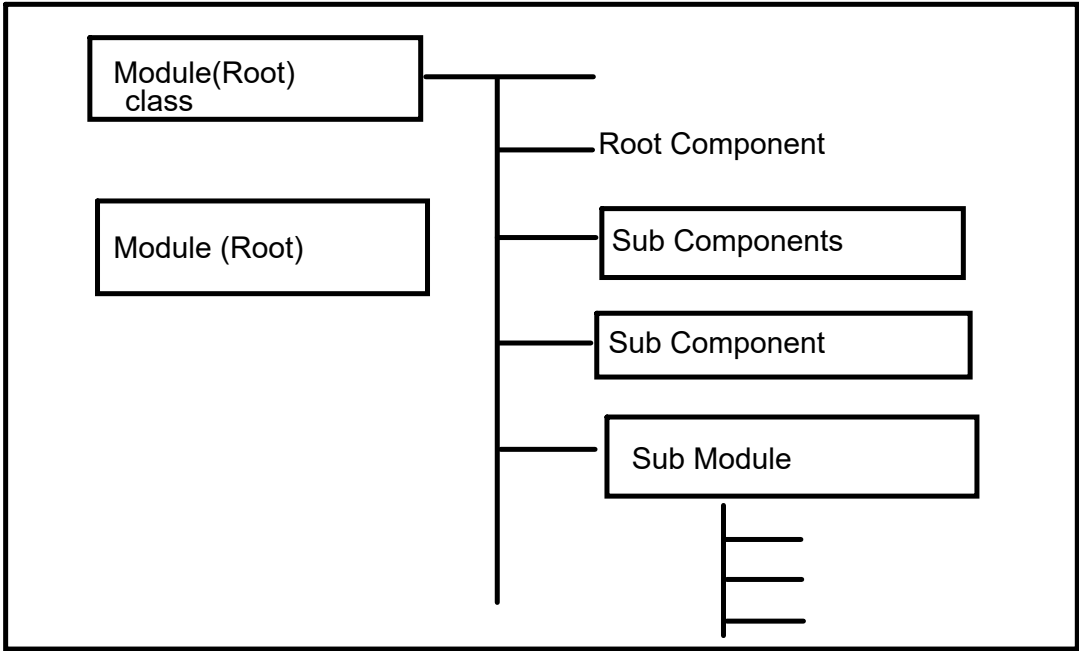1. npm install -g <lib-name> (install library globally in my system)

~ npm install --save --dev <lib-name>

1. add a entry in package.json(update)

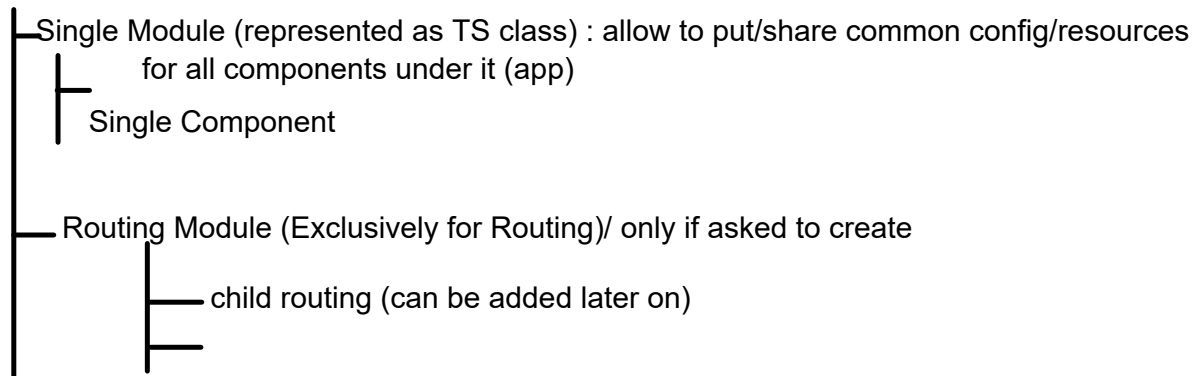2. down load dependency and save it in node_module

Angular Project on Deployment Server

Angular Based Architecture

TS (Component)

packaged as

HTML + ES5

browserslist

packaging needs to be done

compatible to which browsers

Client

index.html

SPA

Group of

JS files

Component

TS file

Unit Test file

of component

spec : test res

HTML : View

Angular Code Construct

Module(Root)
  class

Module (Root)

Root Component

Sub Components

Sub Component

Sub Module

By Default:

    Single Module (represented as TS class) : allow to put/share common config/resources
          for all components under it (app)

     Single Component

    Routing Module (Exclusively for Routing)/ only if asked to create

          child routing (can be added later on)

```
import

import <class name> from <library>

import {<class name1>,<class name2>} from <library>
```

Component :

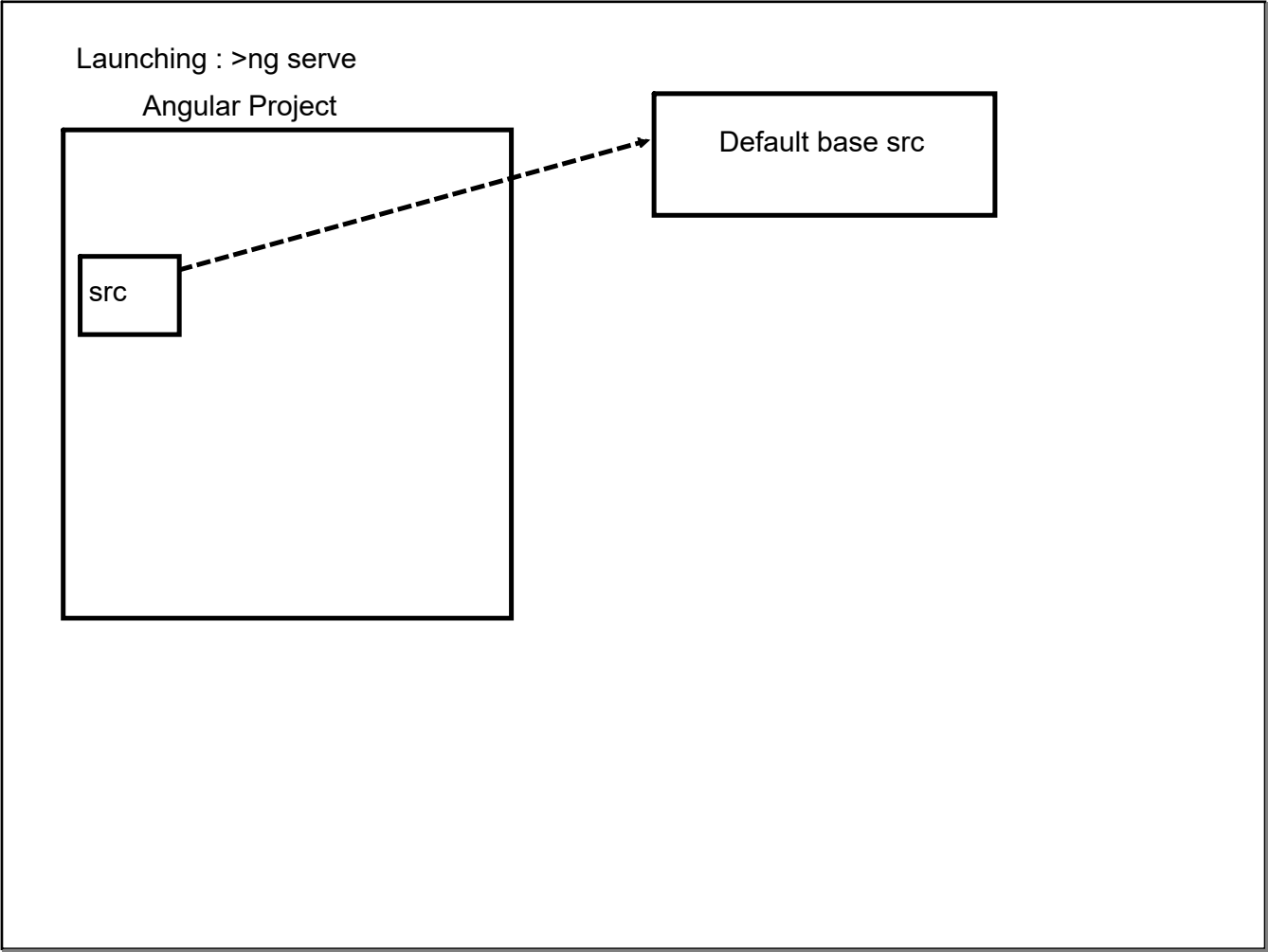   TS class : supported by presentation (View)

By default :

    Angular : 4 files for each component
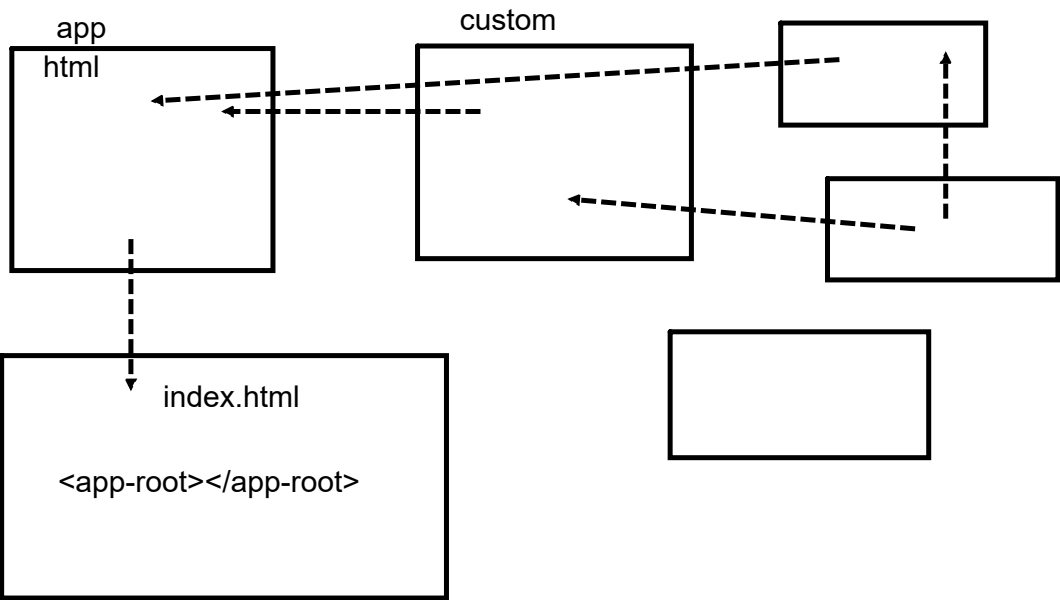
    TS class (mandatory)

    HTML file (View)

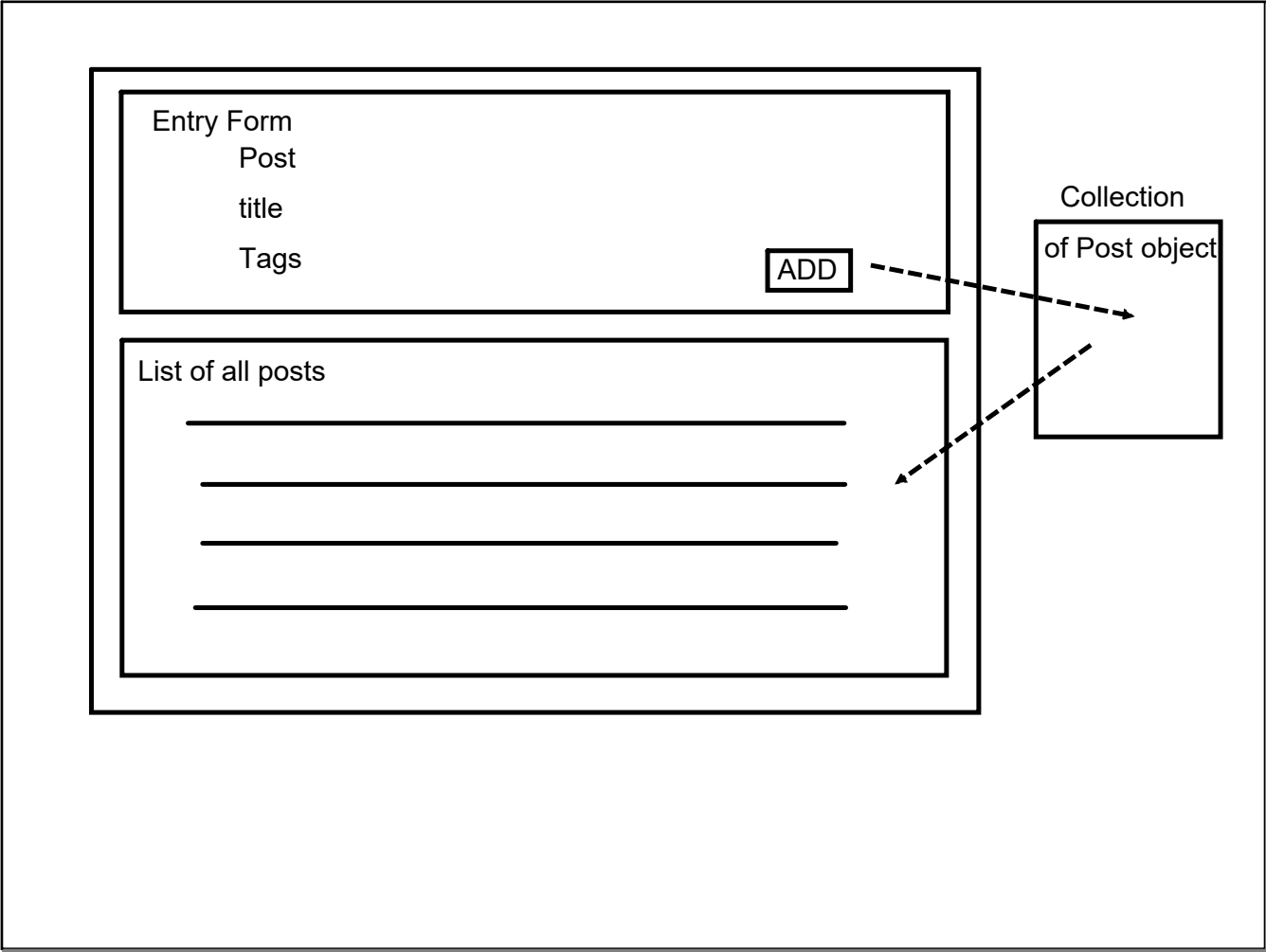   CSS (contain exclusive classes for that component)
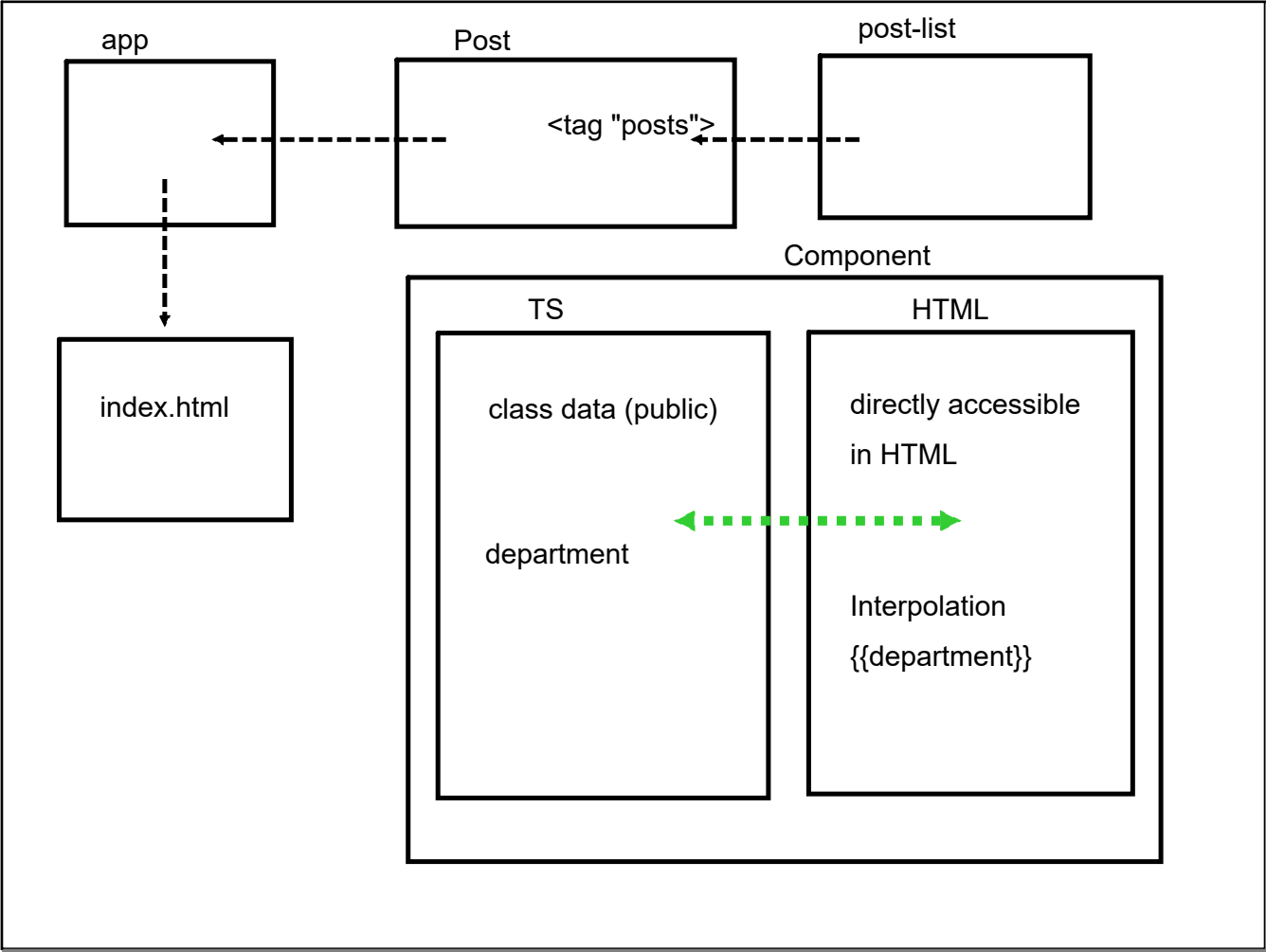
    Test : unit test code for that component

Launching : >ng serve

Angular Project

src

Default base src

Creating a new component

> ng generate component <comp-name>

app

html

custom

index.html

Entry Form
   Post

   title

   Tags   ADD

Collection
of Post object

List of all posts

app

Post

post-list

<tag "posts">

Component

index.html

TS

HTML

class data (public)

directly accessible

in HTML

department

Interpolation

{{department}}

Angular : Directives (Dynamic in HTML)

HTML

HTML features Extended by directive
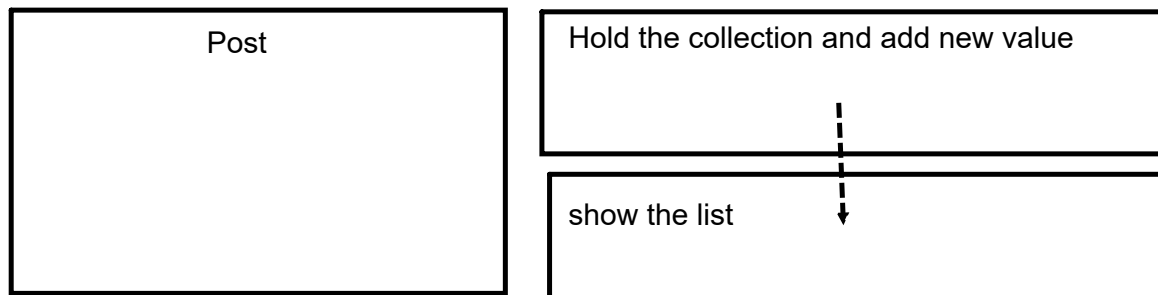
    <new tags>

    < new attributes> along with existing

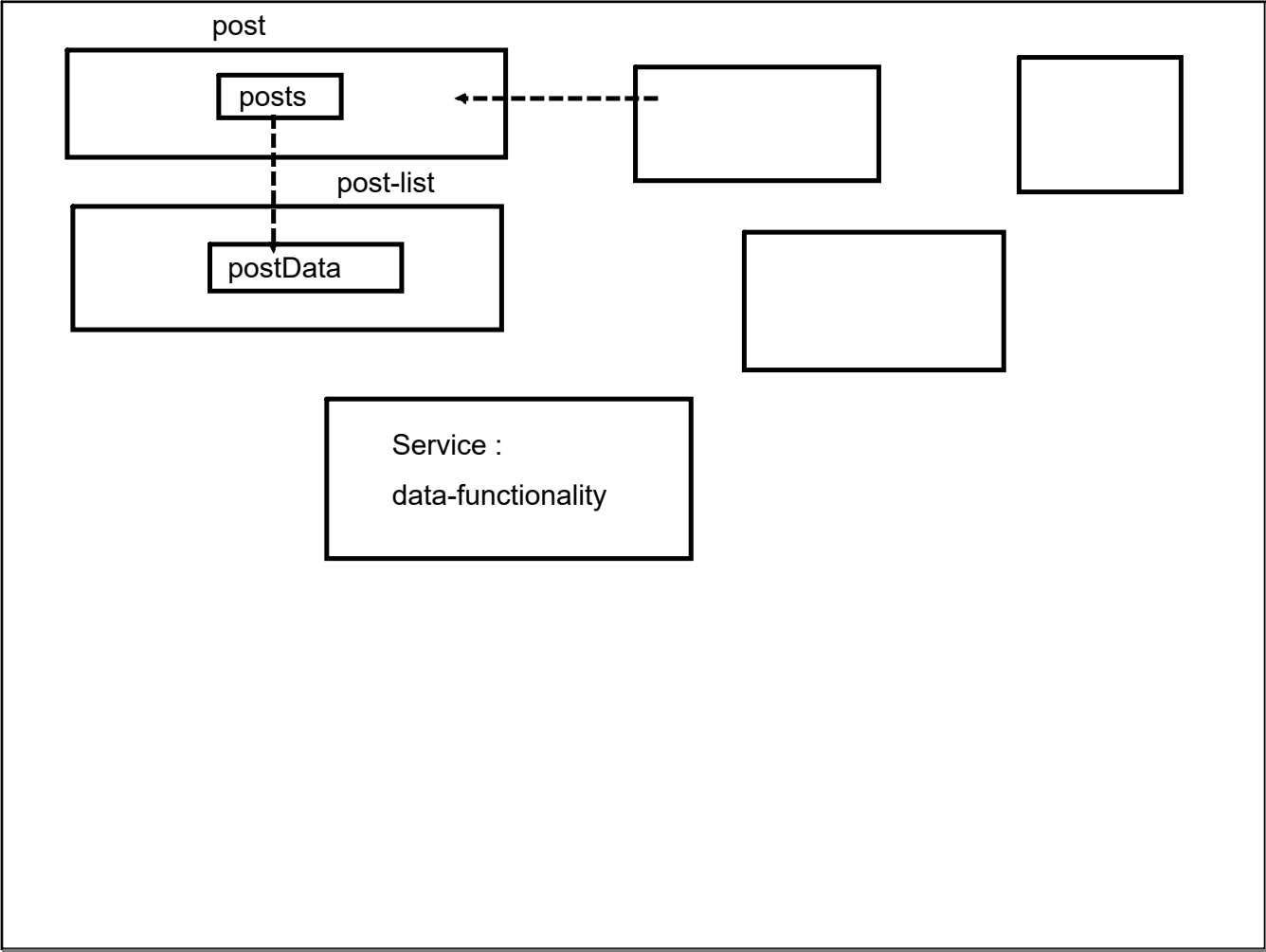HTML attributes, new attributes are
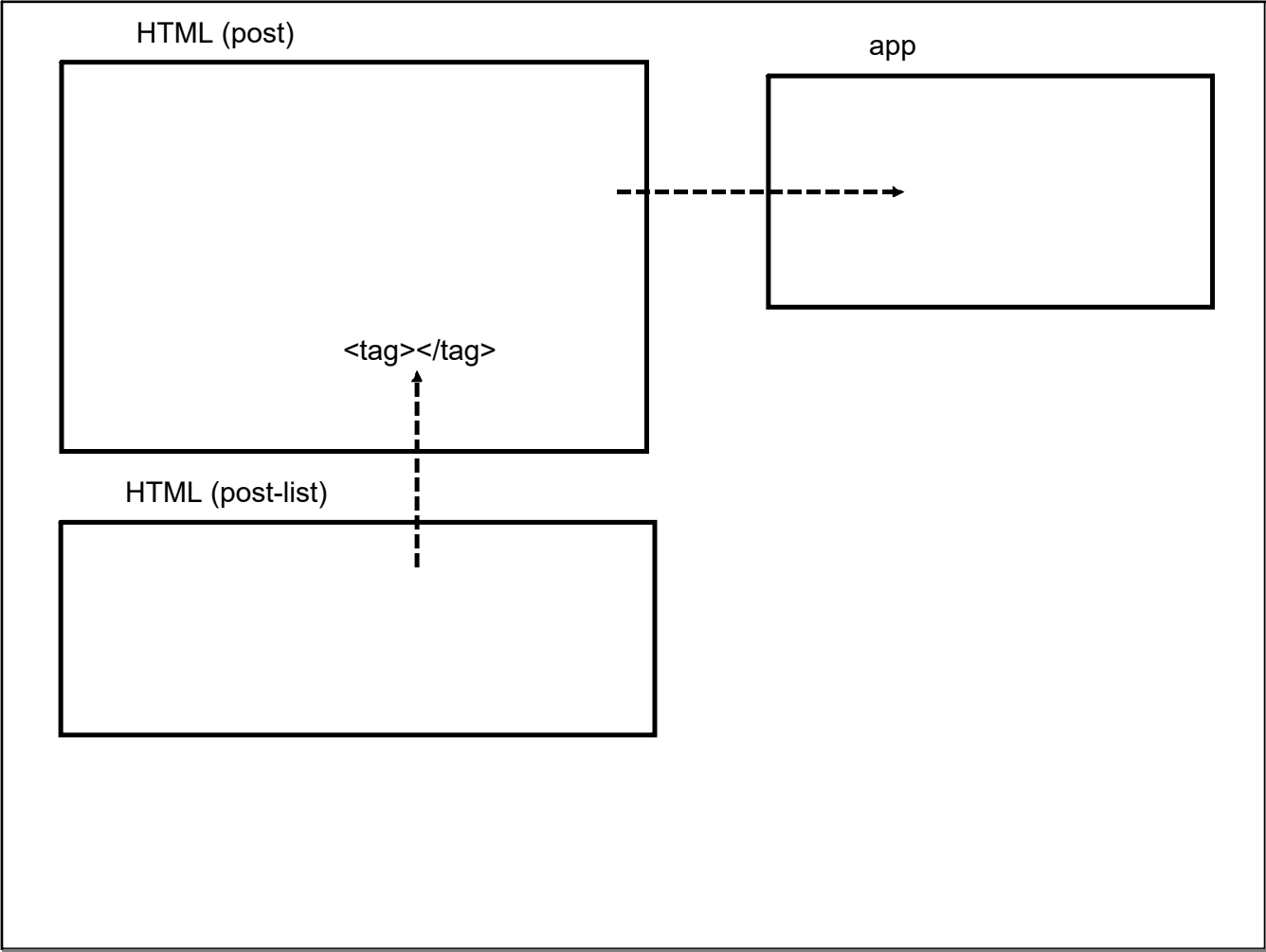
provided by Angular Directives

eg : for loop directive
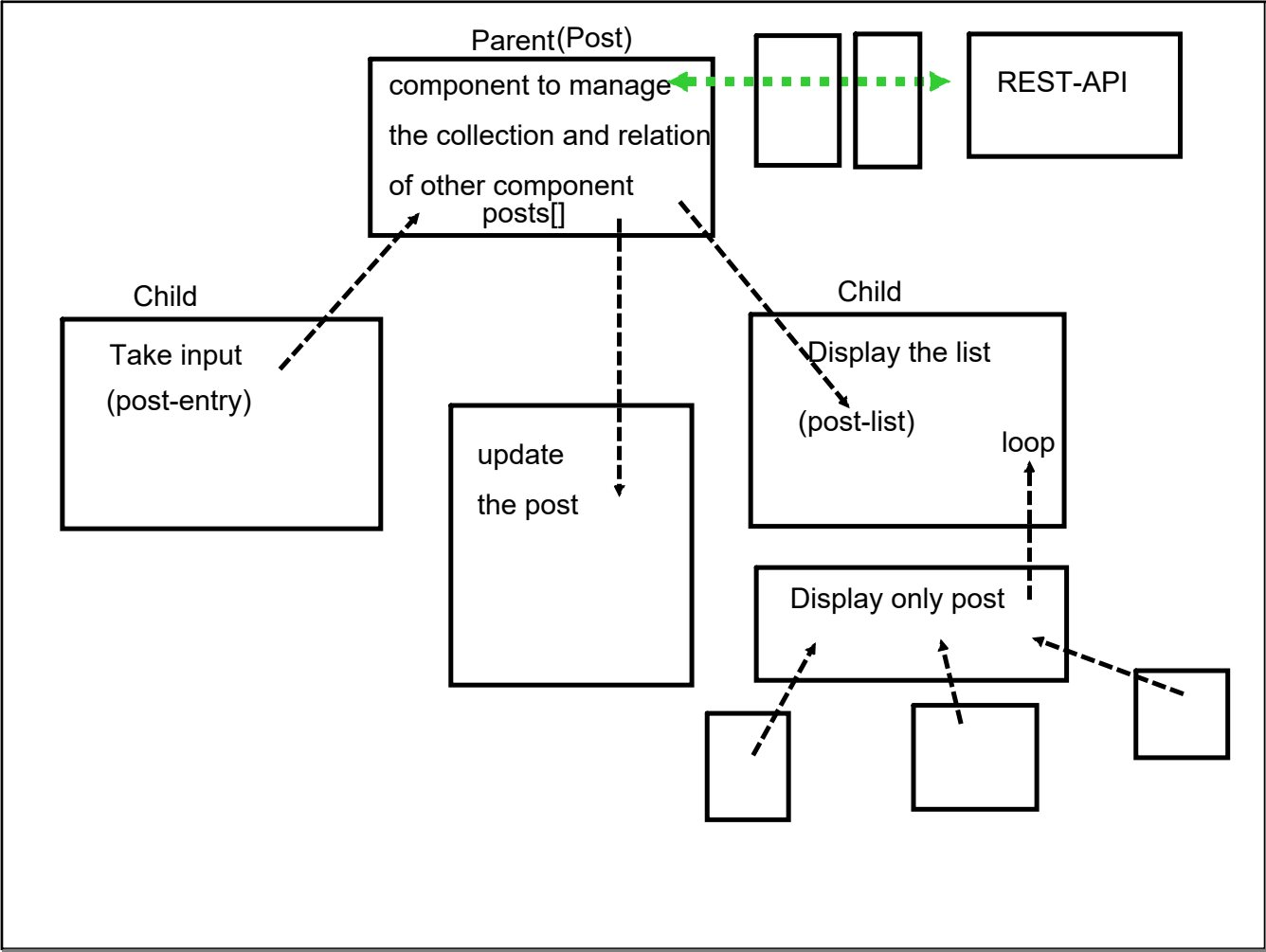
Angular allows to associate a variable with HTML elements

var txtTitle: HTLMInputElement = document.getElementById("")! as HTMLInputElement;

Angular : Synthetic events : allows to call TS class methods on events

Post	Hold the collection and add new value
	show the list

post

posts

post-list

postData

Service :

data-functionality

HTML (post)

app

HTML (post-list)

Parent(Post)

component to manage

the collection and relation

of other component

posts[]

REST-API

Child

Take input
(post-entry)

Child

Display the list

(post-list)

loop

update

the post

Display only post

1. Delegated Entry UI to entry component

2. Add button event handler code also needed to be delegated

Parent(HTML)

handle a click event

<tag (event)="<event handler>" />

<input (click)="addPost(?)" />

<entry (newpost)/>

Post Entry(</entry>

newpost

1. Custom Event

2. Programmatically emit an event + send some data to event handler of another component

Directives :

    *ngIf : Controls the visibility of any component

    *ngIf="<condition>"

    true : Component is visible
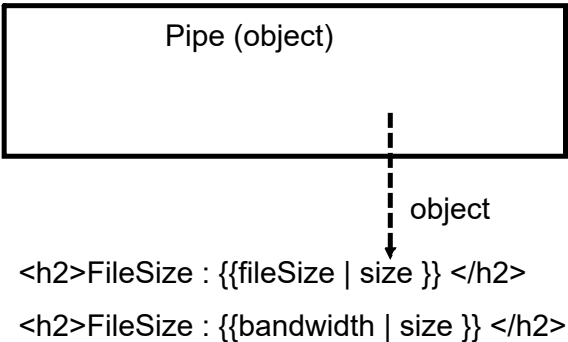
    false : not visible

Pipes : transform the data for presentation purpose

      pipe : |

TS class represents a Pipe

Test File

Function : pure/impure

Pipe (object)   singleton / prototype

↓ object

pure : every time you pass same input,

same output will be received : shared

impure : internal state of function will decide

can't be shared

<h2>FileSize : {{fileSize | size }} </h2>

<h2>FileSize : {{bandwidth | size }} </h2>

Pipe : is pure : singleton

: impure : prototype

Handling Form in Angular

Good Library support

inbuilt modules :

    1. FormsModule
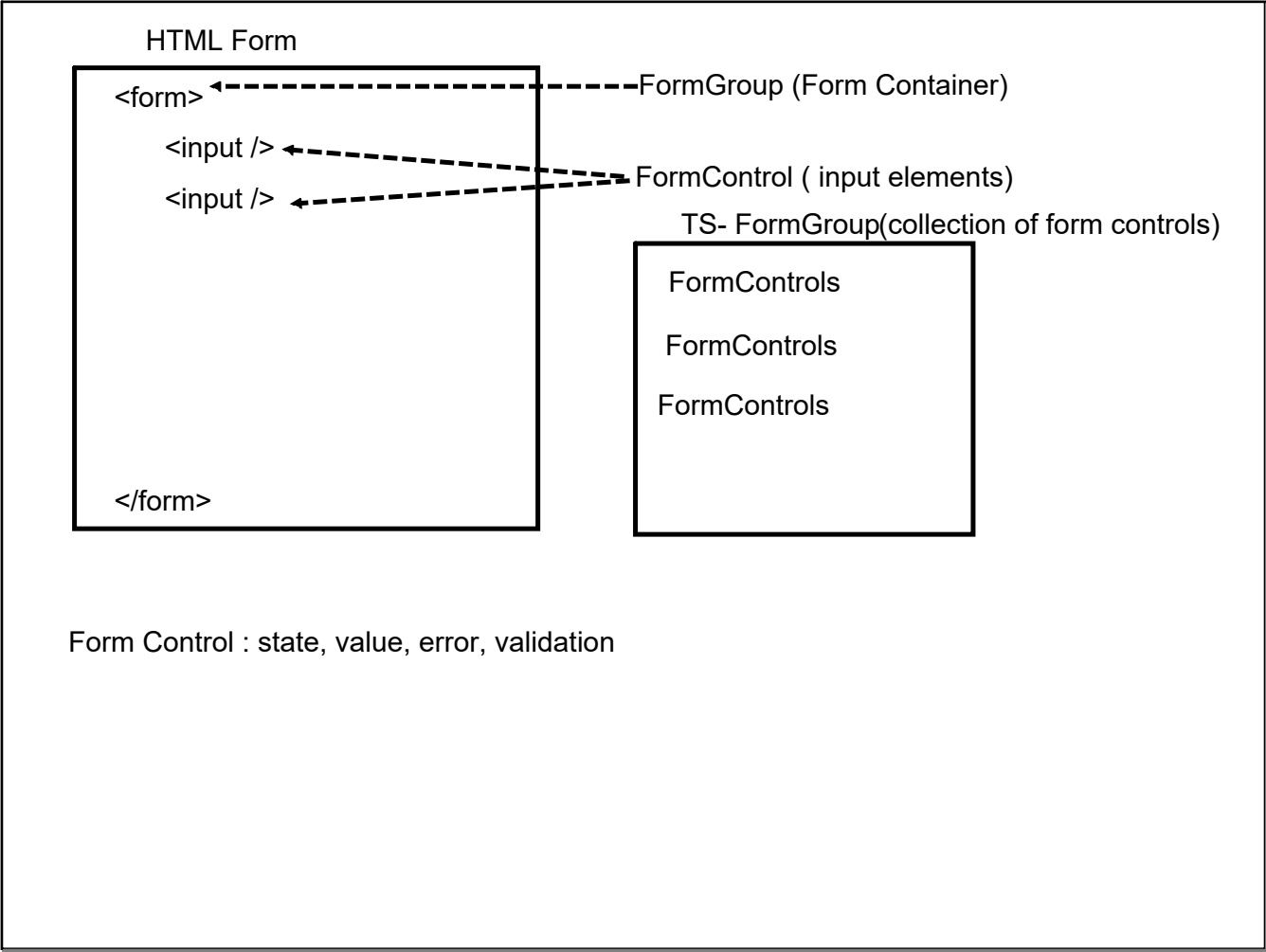
    2. ReactiveFormsModule

Two Different Way :

1. Template

2. Model (Reactive)

TS

Angular Object

HTML

Object Oriented
Implementation
DOM Object : JS

HTML Form

<form> ← ----------------- ---- FormGroup (Form Container)

    <input /> ←

    <input /> ← ---- FormControl (input elements)

                      TS- FormGroup(collection of form controls)

                    FormControls

                    FormControls

                    FormControls

</form>

Form Control : state, value, error, validation

FormsModule(Template)

    FormGroup : ngForm (directive)

    FormControl : ngModel (directive)
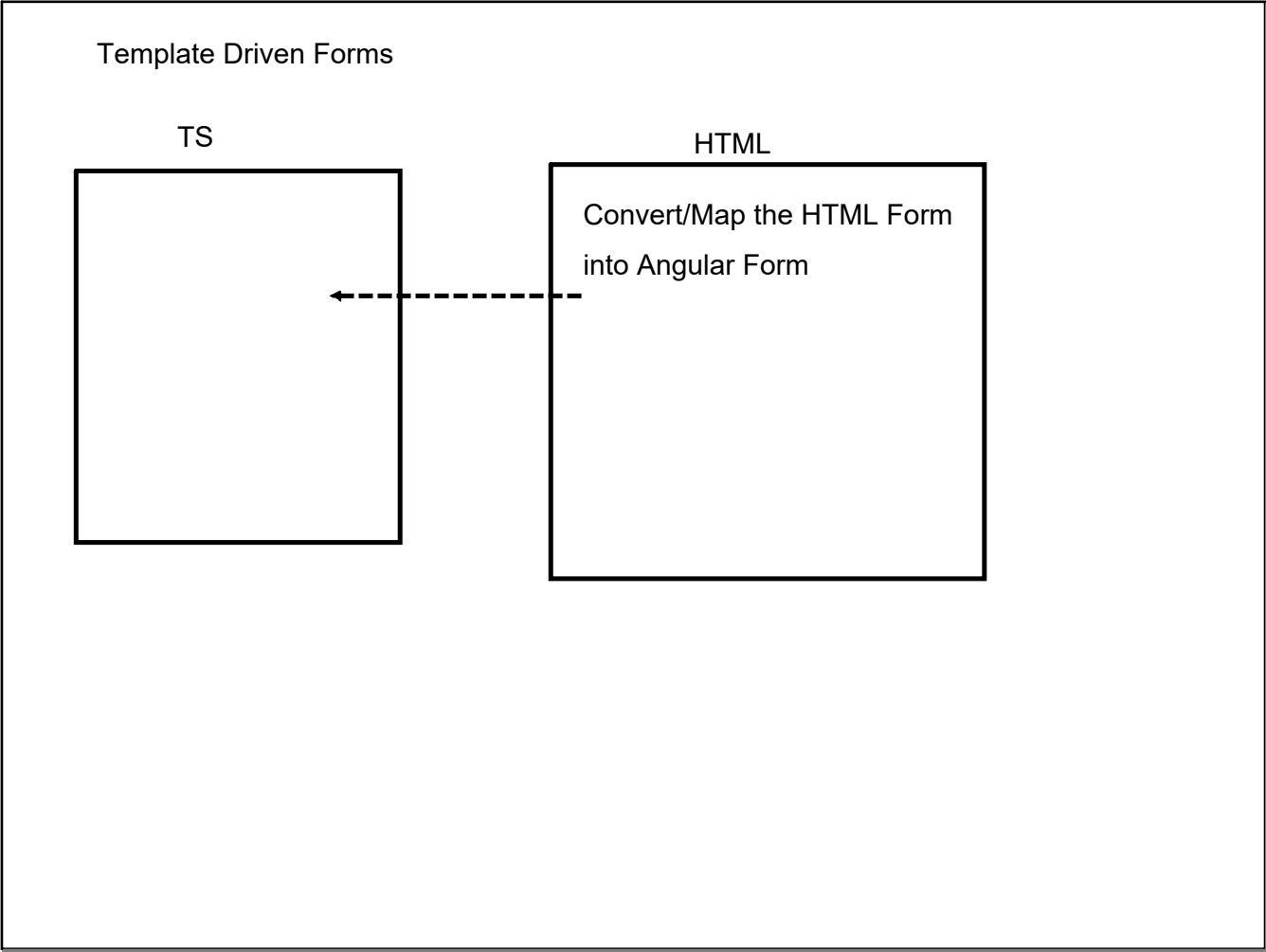
ReactiveFormsModule

    FormsGroup : formGroup
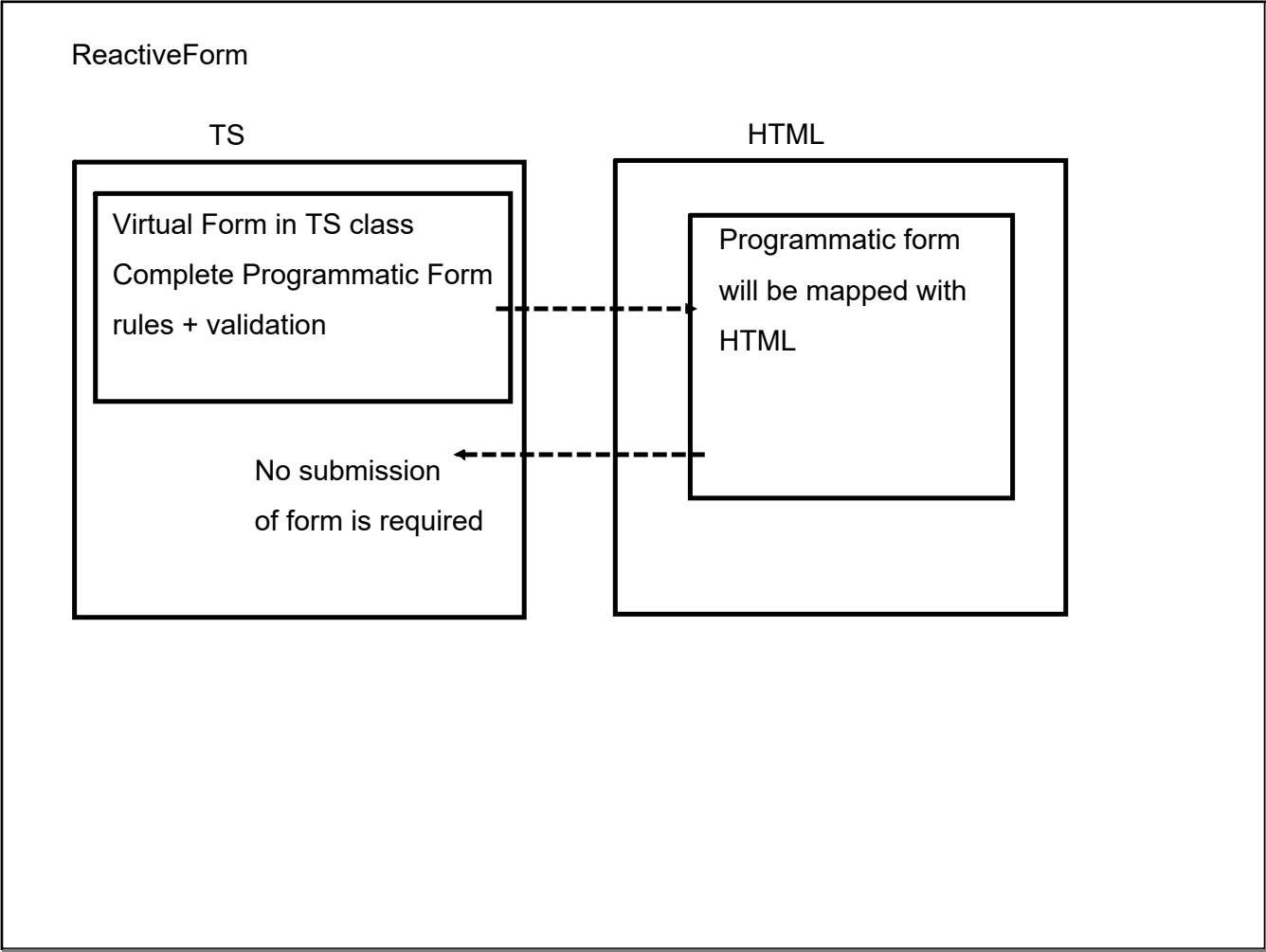
    FormControl : formControl

#Need to add dependency of Module

=> Mapping of HTML to Angular Object is done in view file

=> TS is not having much control over mapping

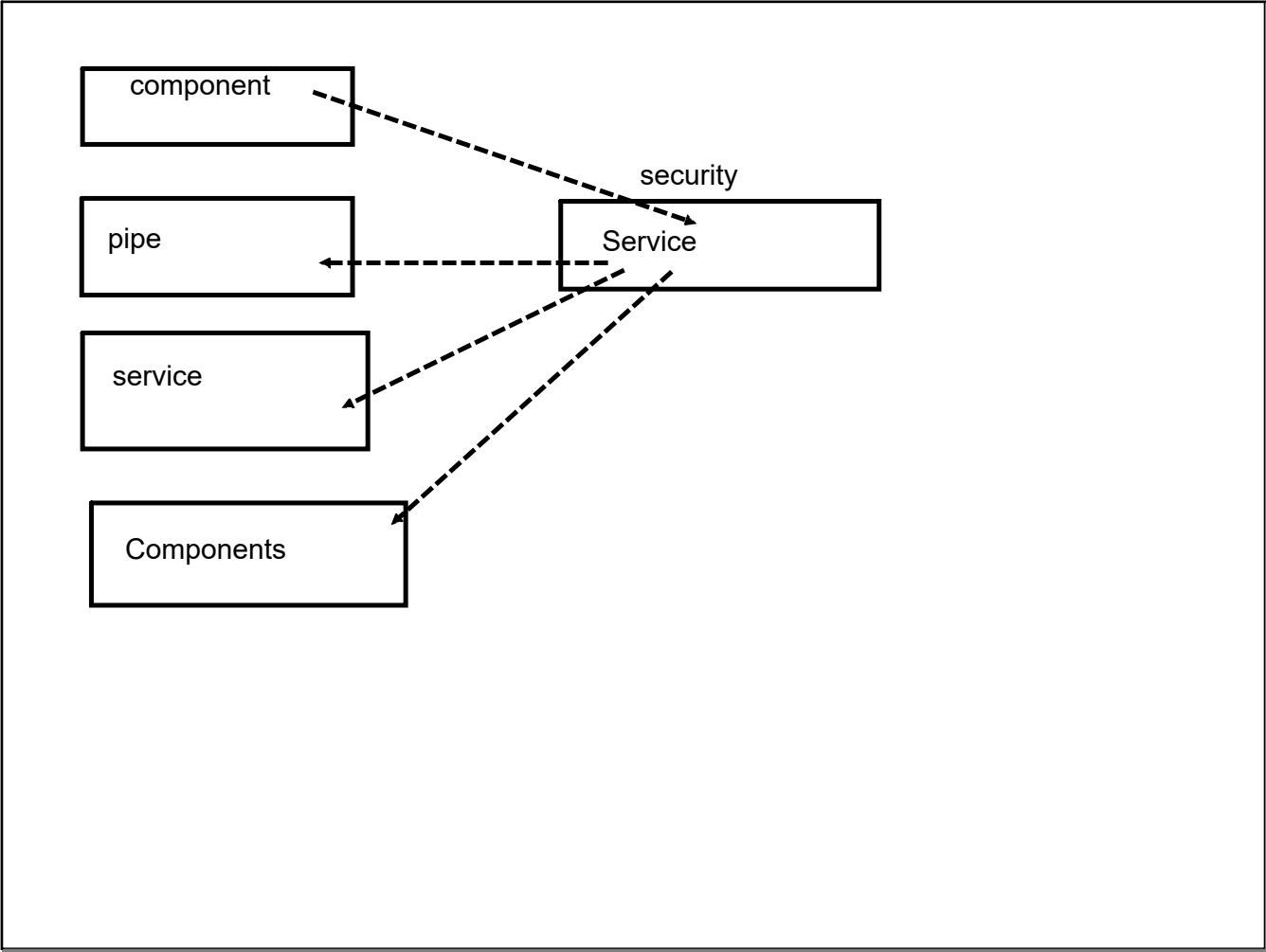=> Not providing feature for Validation

Template Driven Forms

TS   HTML

Convert/Map the HTML Form
into Angular Form

ReactiveForm

TS   HTML

Virtual Form in TS class

Complete Programmatic Form

rules + validation

Programmatic form

will be mapped with

HTML

No submission

of form is required

Routing : SPA

index.html

<router-outlet>

container to hold navigated component

HOME	About	Contact	Post

Search

component

security

pipe   Service

service

Components

Primary Routing Module

Securing url/route (component)

Auth-Guard Service

| H | A | C | ADMIN |

router-outlet

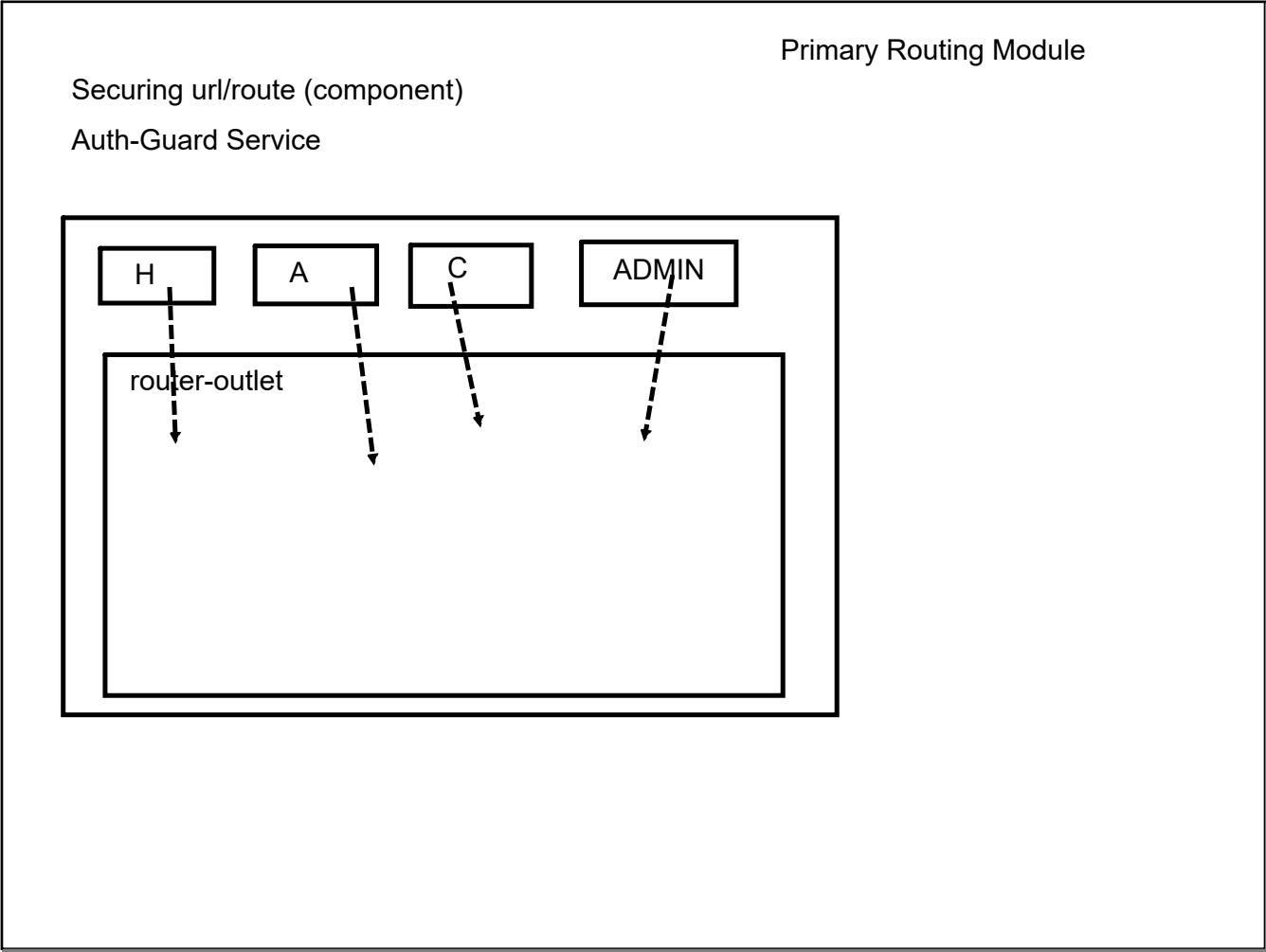app-component
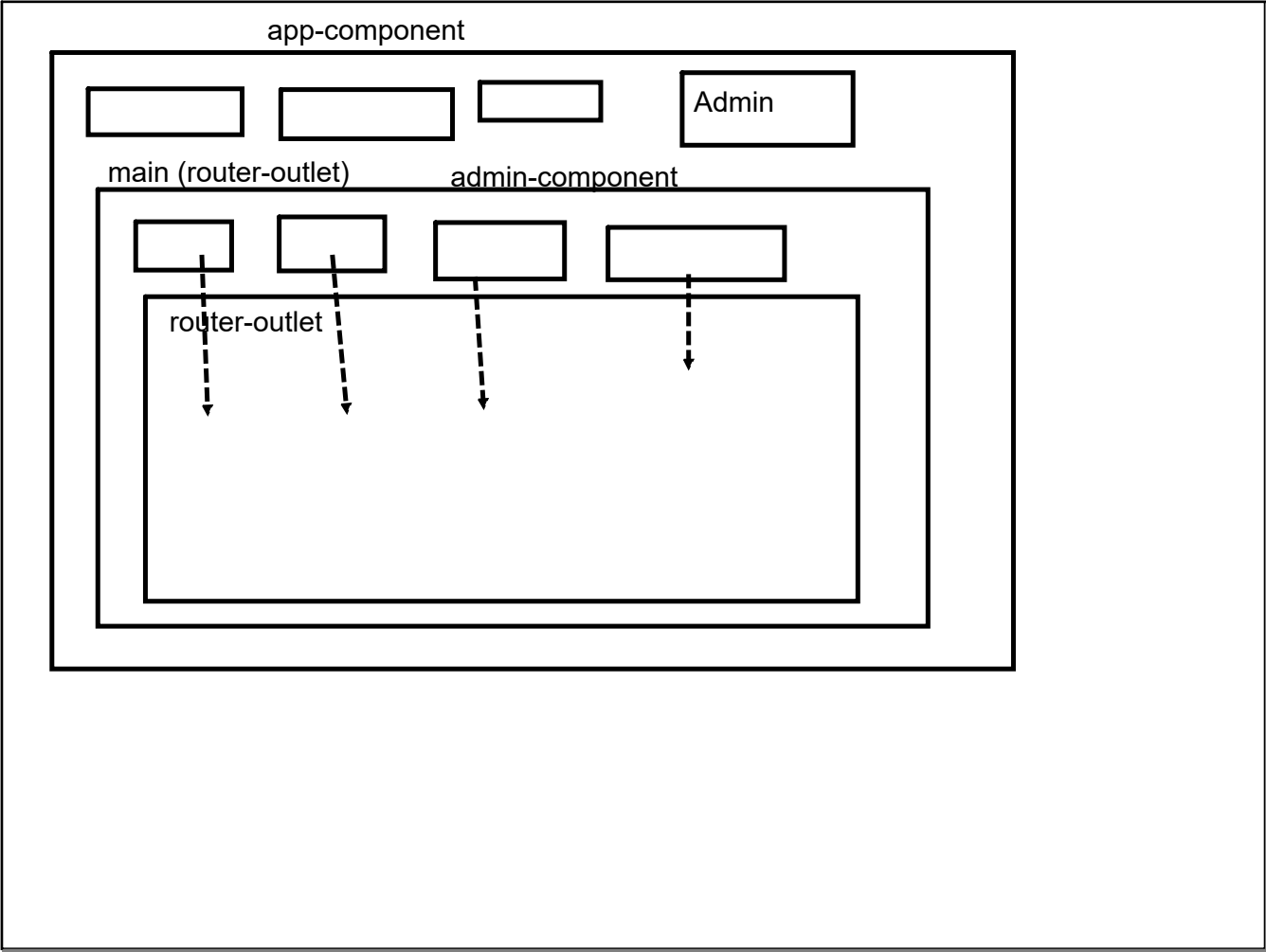
Admin

main (router-outlet)   admin-component

router-outlet

HttpClientModule : Http-Service

Dummy Server/Fake REST API : json-server

1. Allows you to use a json file as the backend DB

2. Exposes all Rest Endpoints on that Json File

Install : Json Server :

>npm install -g json-server

http://localhost:3000/post   : GET (get all)

http://localhost:3000/post/1   : GET (get by id)

http://localhost:3000/post : POST (new post) return the newly added record

http://localhost:3000/post : PUT (edit post) return the newly edited record

http://localhost:3000/post/1 : DELETE (delete that record)

JS : ES

Standard ES5 : support is by default available

jquery :

Library of JS (ES5):

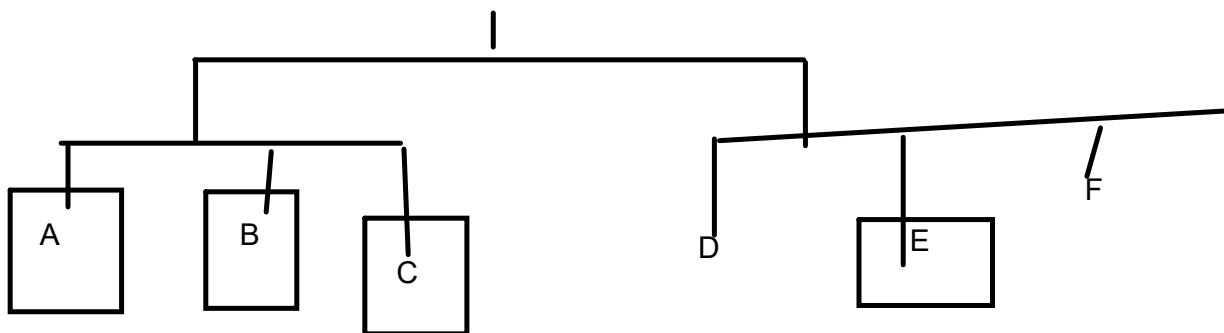ReactJS is just a library : exclusive to build effecient UI (V part of MVC)

Build UI of large complex application (frequently changing data)

: Renderring would be frequent

Traditional approach of page renderring:

Browser :

DOM TREE

IF any change in any part of DOM Tree, complete Tree is re-renderring

ReactJS : ES6 : needs to be transpiled : can't be directly used on browsers

React Component : JS functions : which generates an (UI) output whenever it is called

eg : render()

generates some output

```
<div>
    <h2>Hello All</h2>
    <p>10:30 AM</p>   // programmatically
</div>
```

ReactJS : Virtual DOM
In-memory representation of
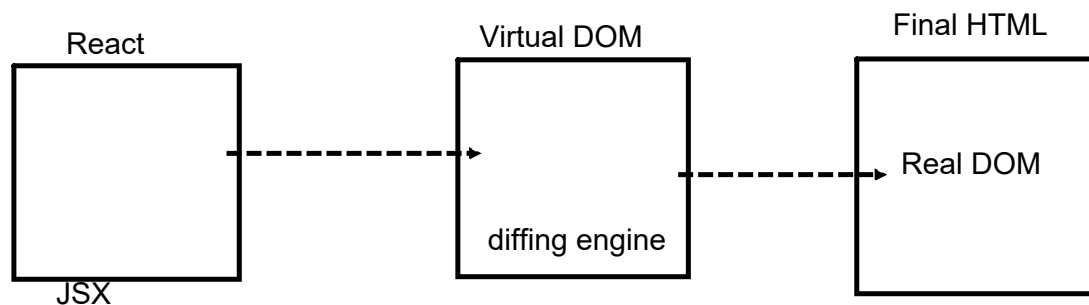real DOM:
diffing engine :

called after 1 min   only <p> component

```
<div>
        <h2>Hello All</h2>
        <p>10:31 AM</p>   // programmatically
    </div>
```

```
    document.getElementById("resp").value=""; // REACT JS Approach (granular approach)
    ES5 approach


ReactJS Component is JS Function
render(){
    // code a code generate a UI
    // JSX syntax : JavaScriptXml Syntax
    Integrates Javascript with HTML
}
```

React   Virtual DOM   Final HTML

```
┌──────────────┐        ┌──────────────┐               ┌──────────────┐
│              │        │              │               │              │
│              │ - - -▶ │              │               │  Real DOM    │
│              │        │              │    - - - -▶    │              │
│              │        │ diffing      │               │              │
│              │        │ engine       │               │              │
└──────────────┘        └──────────────┘               └──────────────┘
 JSX
```

React JS Library

Two Library

    1. react : Main ReactJS lib

    2. react-dom : Virtual DOM

> npm tool

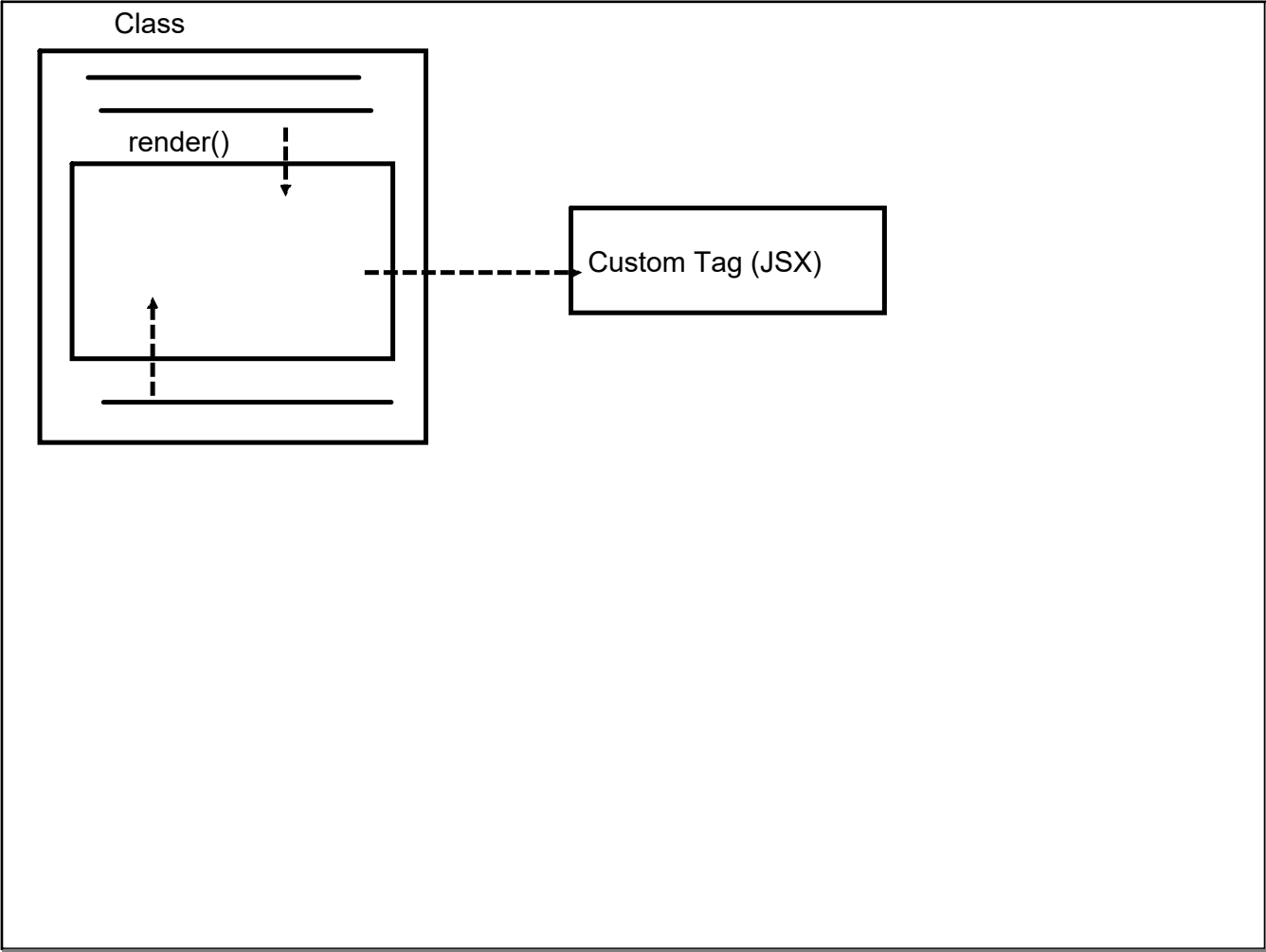for managing everything about ReactJS application

    create-react-app (cli)
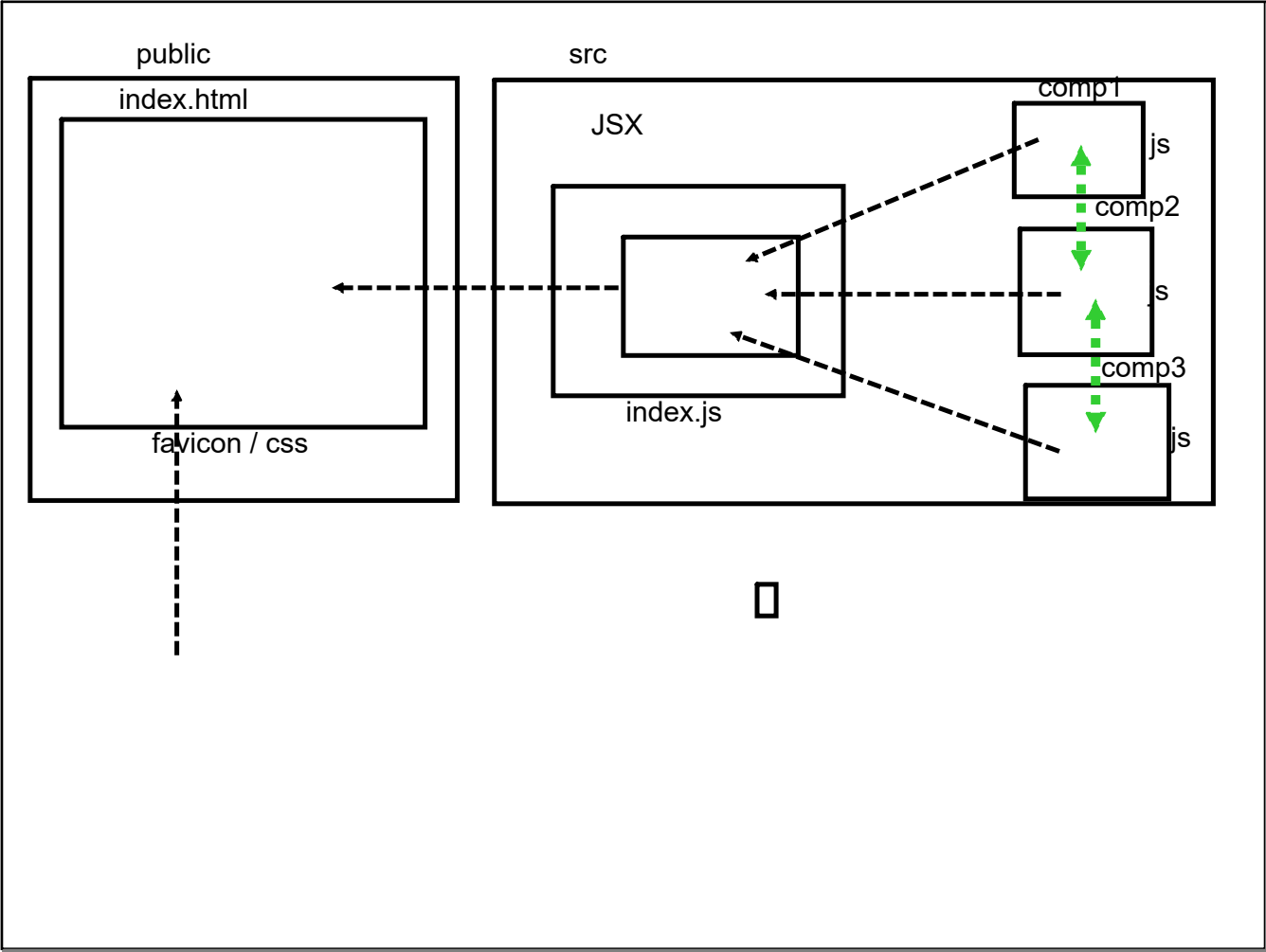
install:
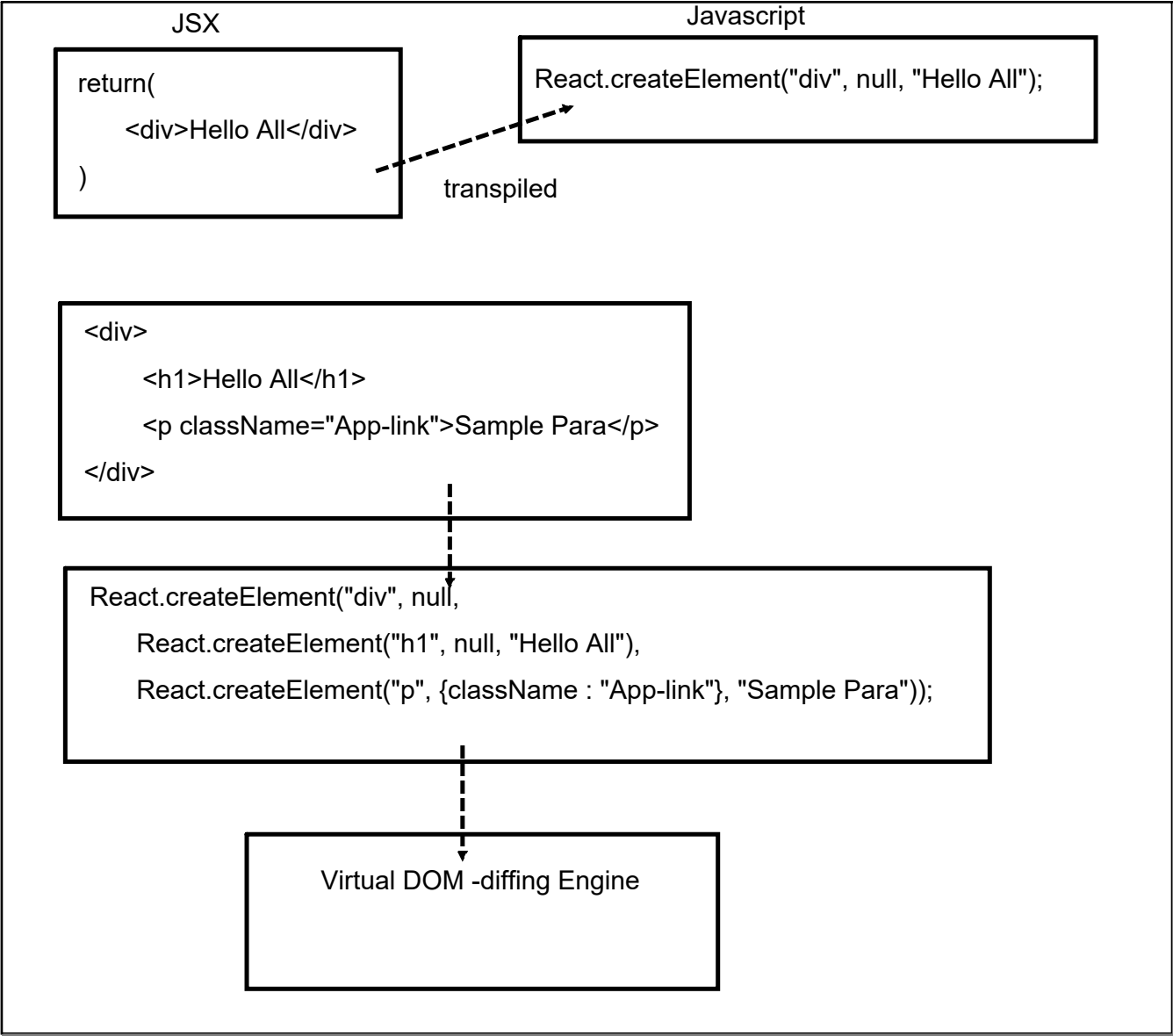
    > npm install -g create-react-app

After installed

    > create-react-app <app-name>

Class

render()

Custom Tag (JSX)

```
class App ....{
    render(){
    }
}
```

ReactDOM.render(
    <App/> , document.getElementById("root")
)

public

index.html

src

JSX

comp1

js

comp2

s

favicon / css

index.js

comp3

js

JSX

Javascript

```
return(
    <div>Hello All</div>
)
```

React.createElement("div", null, "Hello All");

transpiled

```
<div>
    <h1>Hello All</h1>
    <p className="App-link">Sample Para</p>
</div>
```

```
React.createElement("div", null,
    React.createElement("h1", null, "Hello All"),
    React.createElement("p", {className : "App-link"}, "Sample Para"));
```

Virtual DOM -diffing Engine

attributes

One way

Title ------------
Tags ------------
Post---------------- collection (props)

Title -----------
Tags ------------
Post----------------

Title -----------
Tags ------------
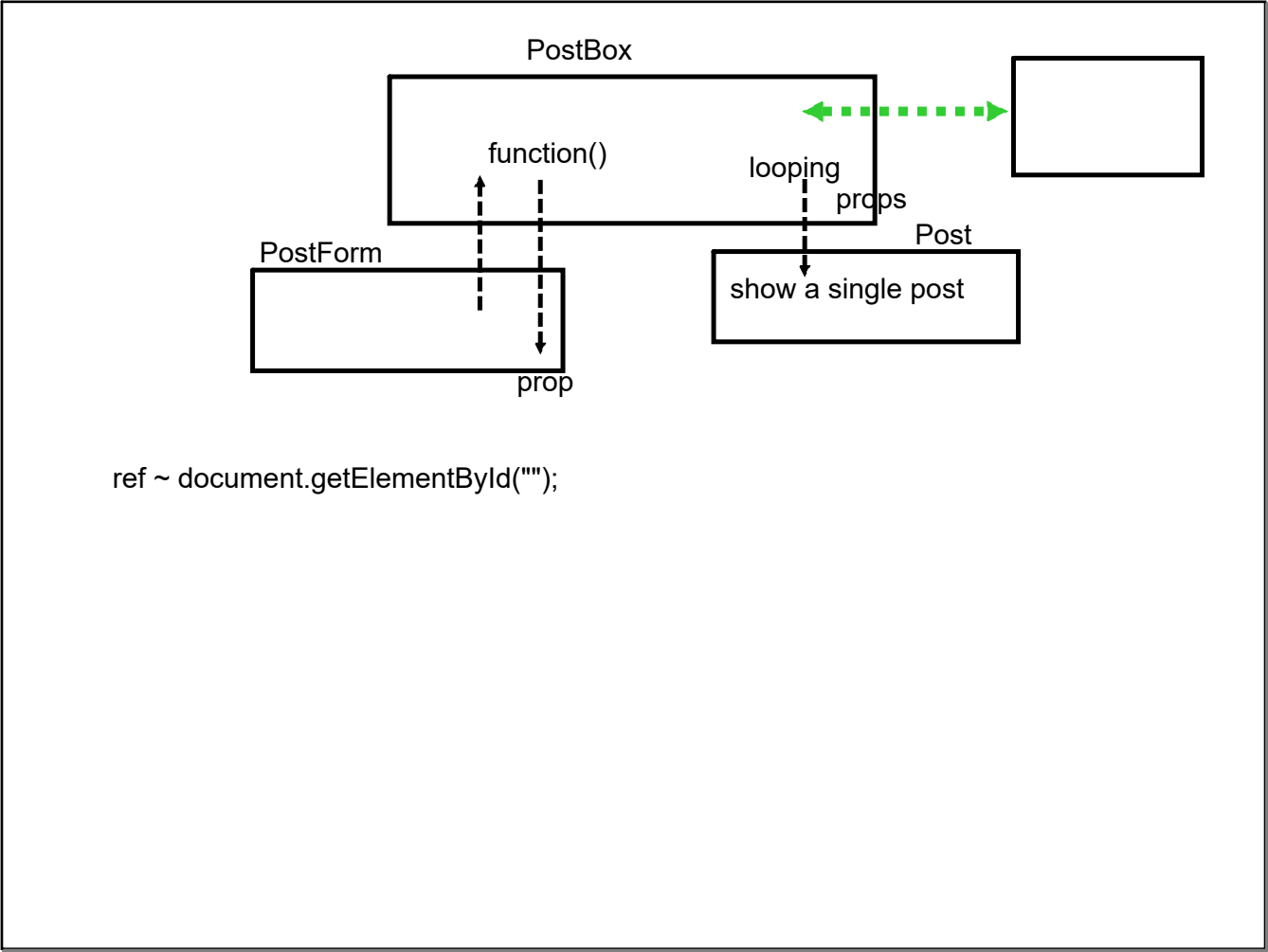Post----------------

ParentComponent
PostBox

Post

render() method call is going to define the UI change

Call to render is controlled by few factor

1. Props : any change in prop value would trigger render() call

2. State : inbuilt object (exclusive to a component) : any change will trigger render call

PostBox

function()

looping

props

PostForm

Post

show a single post

prop

ref ~ document.getElementById("");
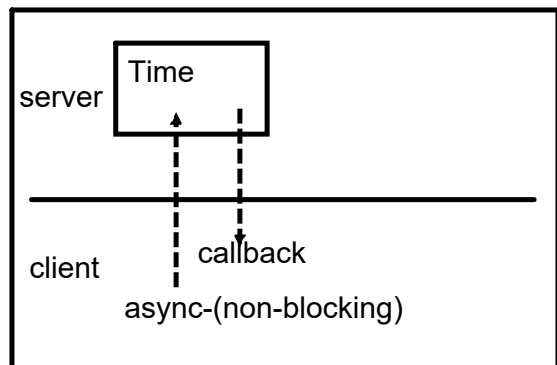
Make app talk with backend-server async

 AJAX call (jquery)

1. Traditional way : CDN Link / download lib

2. npm way

install and save dependency in package.json

>npm install --save jquery

Time

server

client   callback

async-(non-blocking)

Life Cycle of React Component :

When a component is used for renderring

   Instance is created

---

1. constructor

2. componentWillMount() : before renderring

   (only once : first time rendering : not with every rendering)

3. render() : (first call)

4. componentDidMount() : just after render (only once : after first rendering)

---

5. componentWillReceiveProps() ;   Netty Server

   whenever prop/state change

   invoked before next rendering (before every re-rendering)
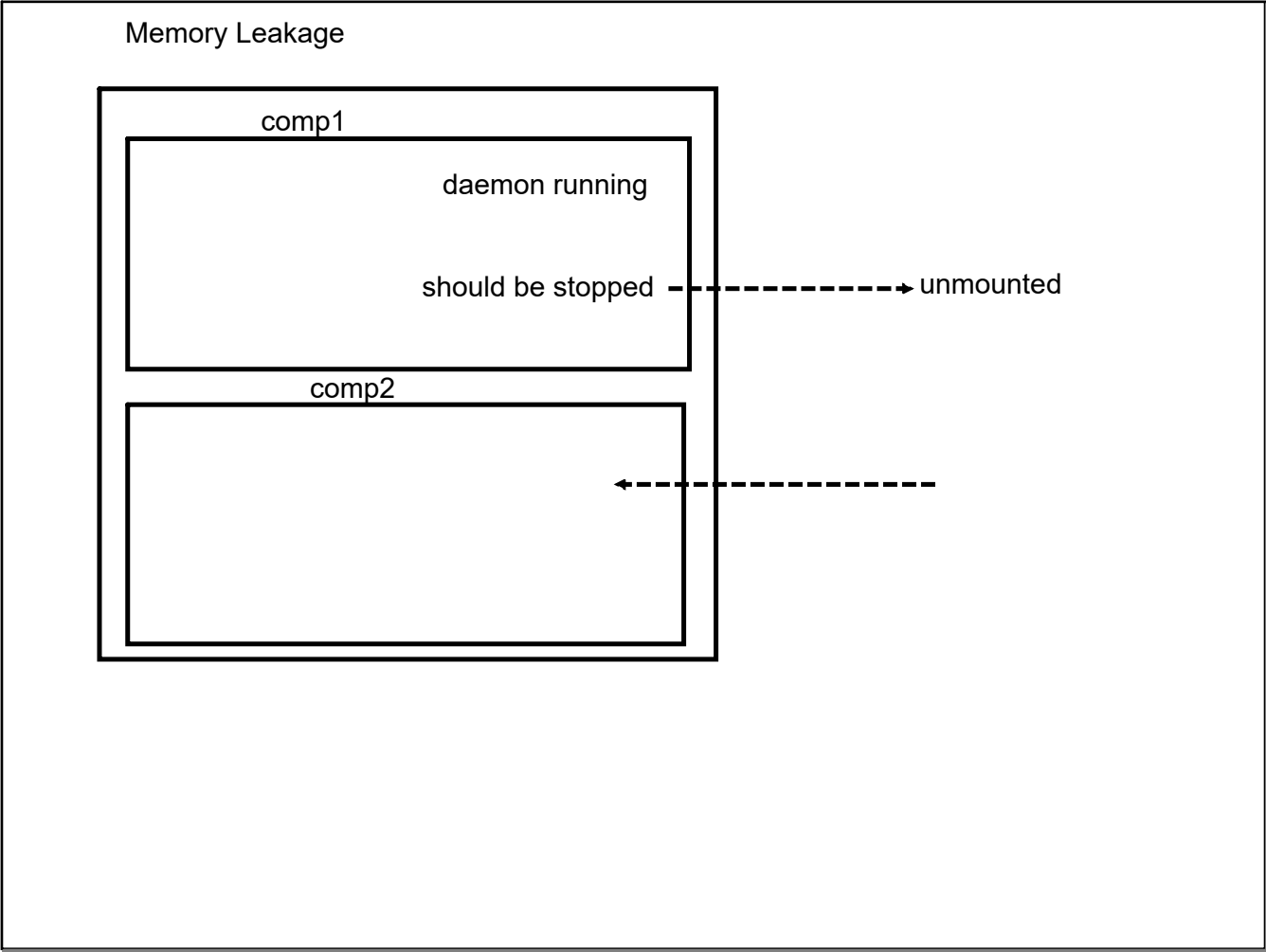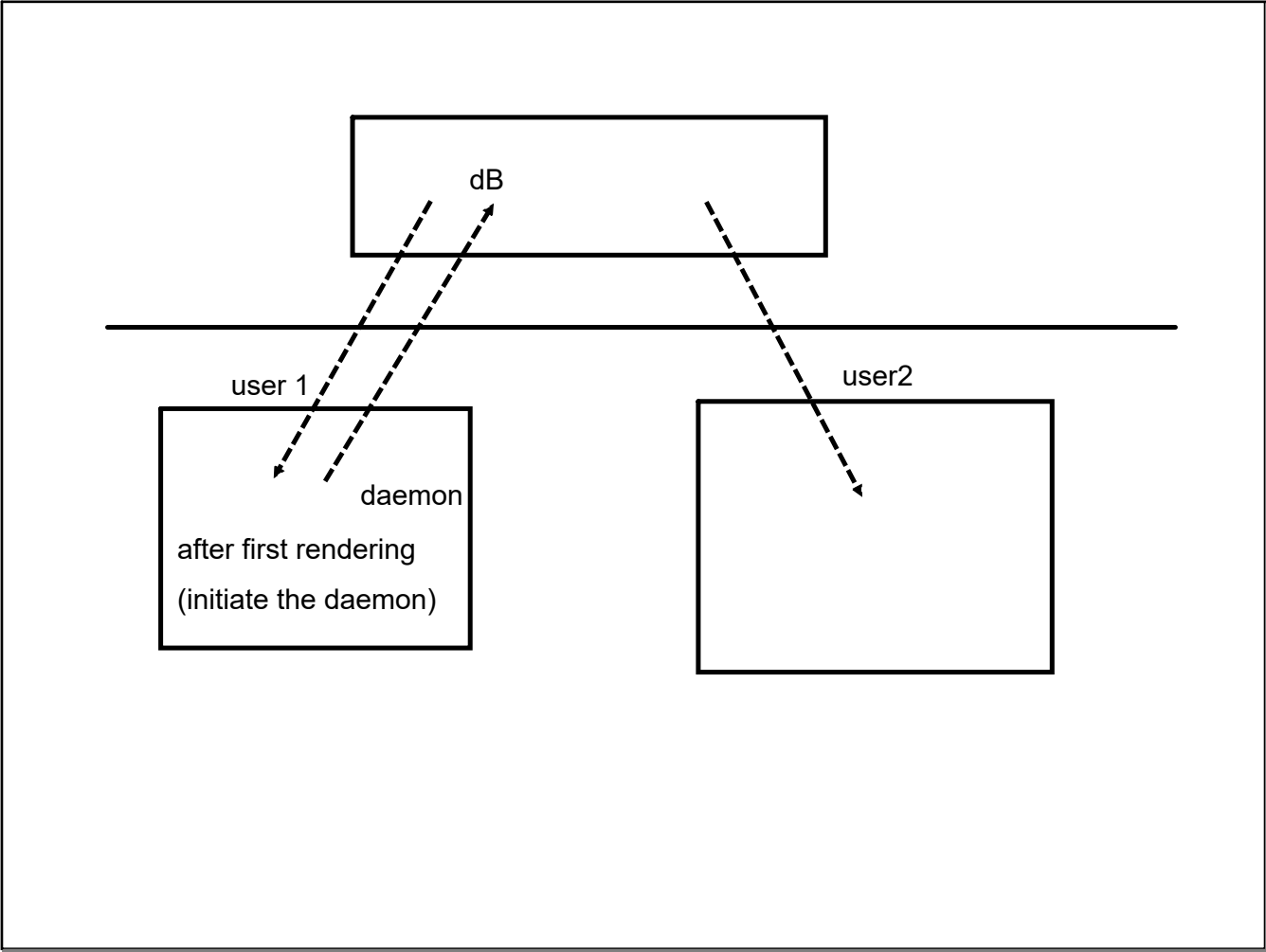
6. shouldComponentUpdate()

   #allows to customize the flow

returns boolean :

   true : re-rendering
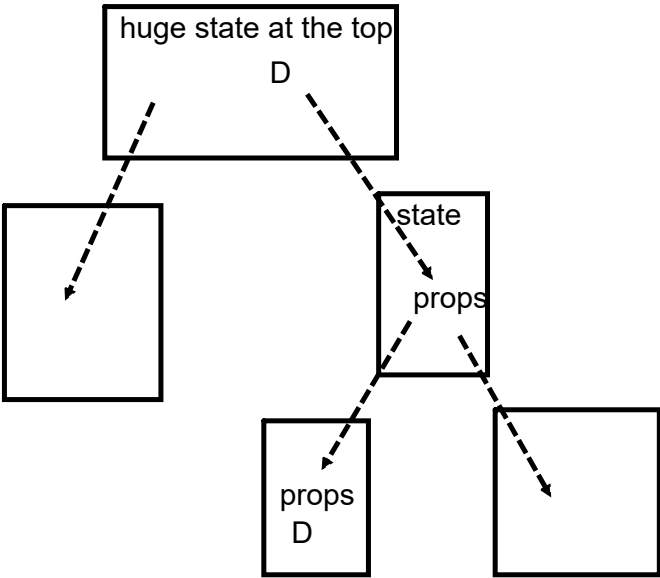
   false : no re-rendering

7. componentWillUpdate() : only of true is returned

8. render () : re-rendering

9. componentDidUpdate(); just after re-rendering

10. componentWillUnmount() : component is removed from Virtual DOM

Memory Leakage

comp1

daemon running

should be stopped - - - - - - - - → unmounted

comp2

dB

user 1

daemon

after first rendering

(initiate the daemon)

user2

Flux Design Pattern (Redux API)

Data Flow

huge state at the top
D

state

props
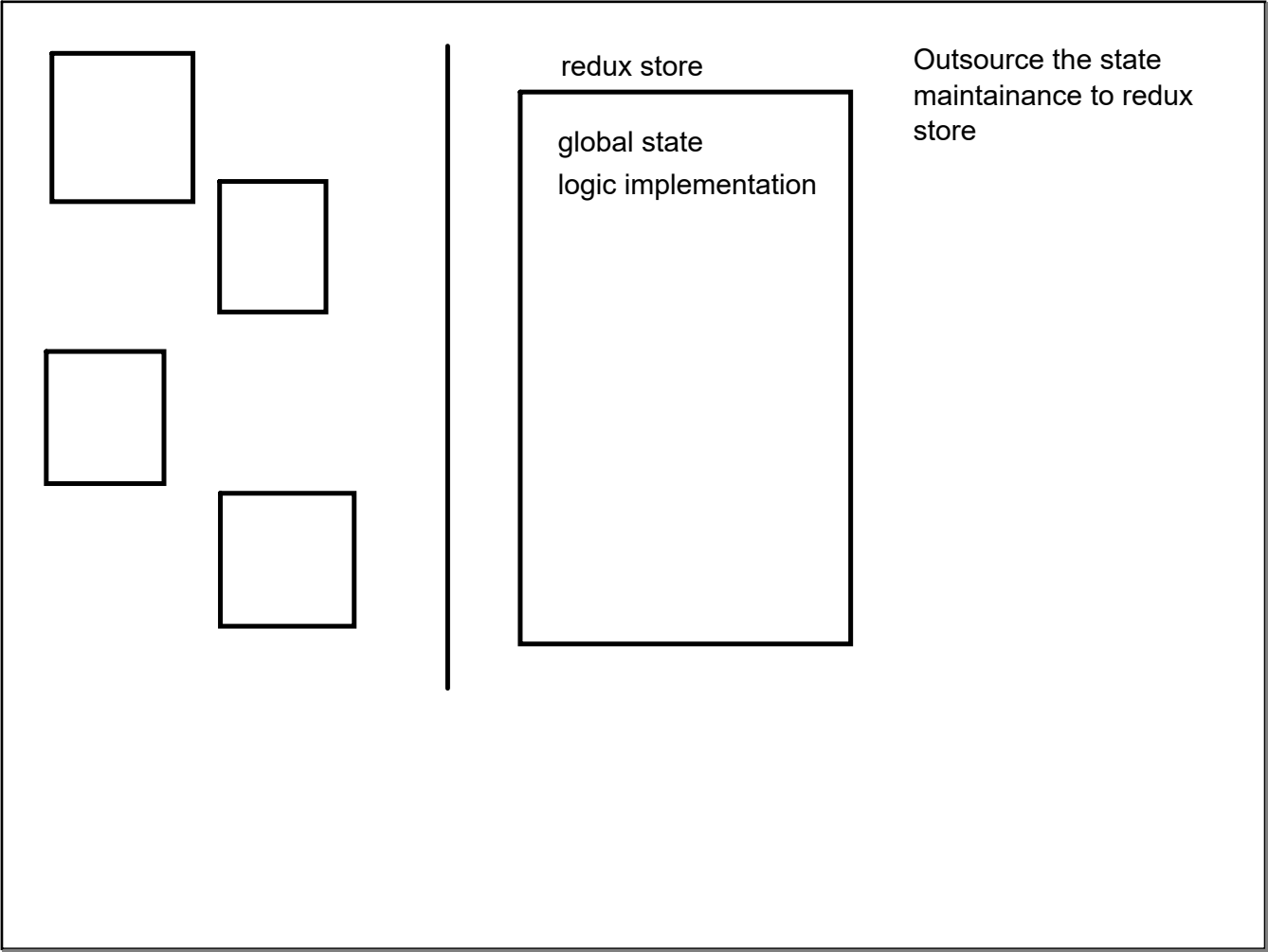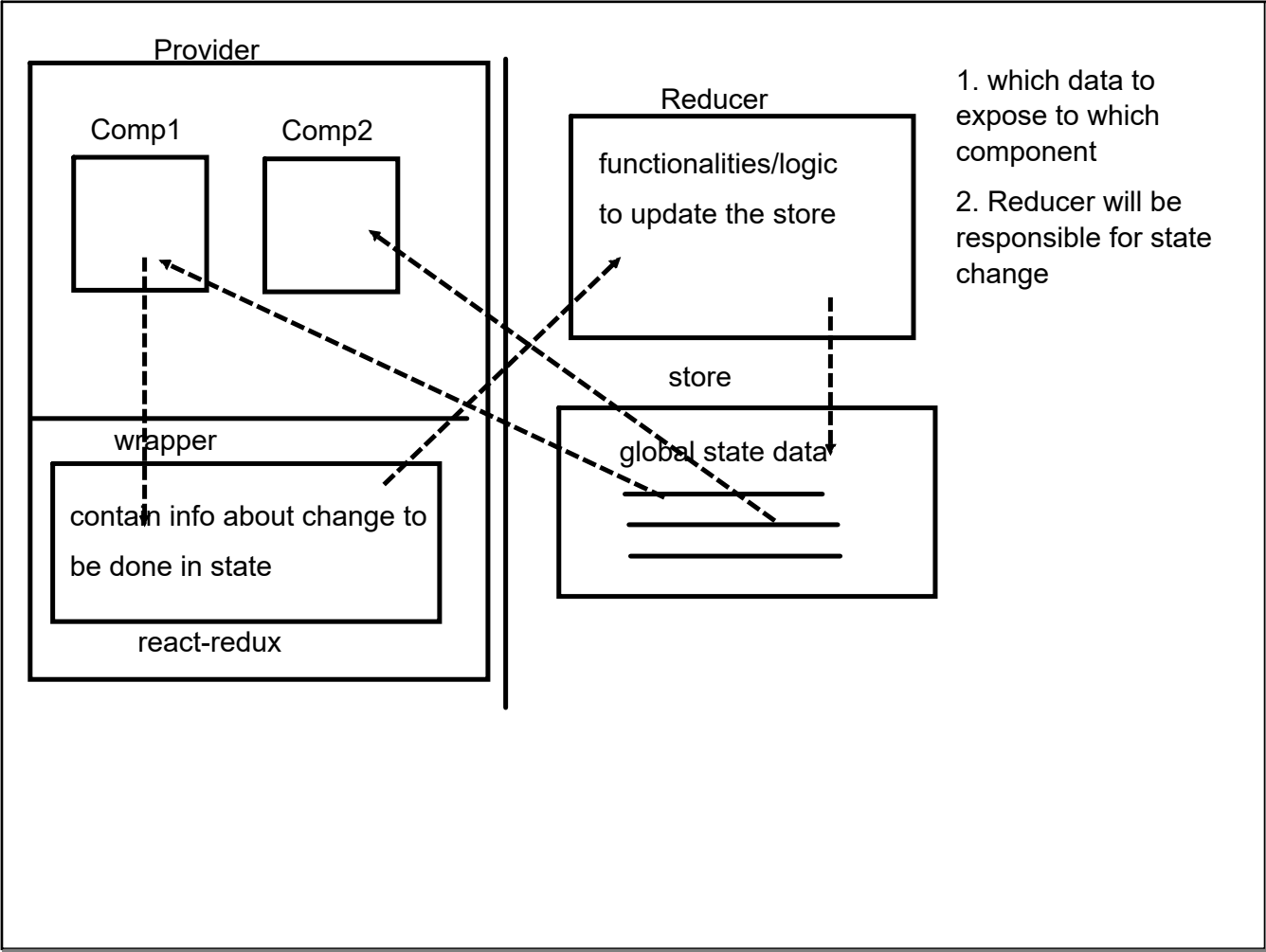
props
D

(inbox)list of email:

list -> email view

#fetch the data of that email from server

mark as read

unread mail counter needs to be reduced

url to be changed

all update needs to sent to server

redux store

global state

logic implementation

Outsource the state maintainance to redux store

Provider

Comp1   Comp2

Reducer

functionalities/logic
to update the store

1. which data to
expose to which
component

2. Reducer will be
responsible for state
change

wrapper

store

contain info about change to
be done in state

global state data

react-redux

install lib

    -->redux

    --> react-redux (plumbing)

> npm install --save redux react-redux

component

Counter

  Event (+)(-)

redux store

Counter value
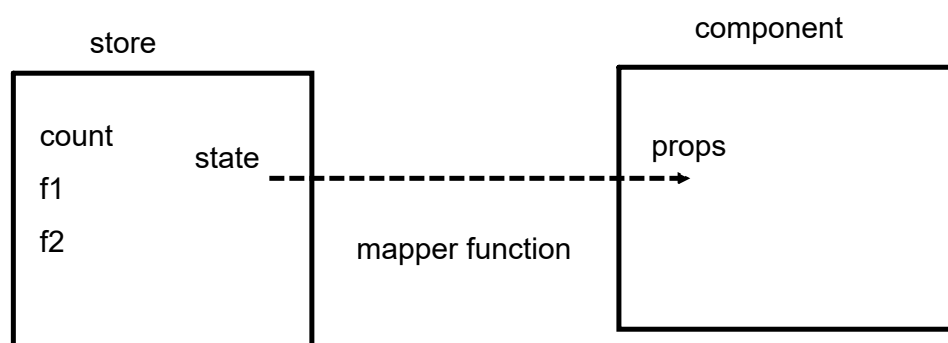
create a store management resource

expose the store to react component

configure the component to use store

Need to map state(redux-store) data to props for component

store   component

```
count
        state
f1  - - - - - - - - - - - - ->  props
f2
        mapper function
```

install an API for routing

> npm install --save react-router-dom

Server

prod package

Dev

index.html

js (ES5)
js (ES5)

MongoDB

High Performance : No SQL overheads

Document Oriented database

Schema Less :

   Json Object Format

Document based Query ~ Deep query-ability

Easy to scale (no constraints)

Reactive Driver for MongoDb : End to End Reactive App

RDBMS	MongoDB
	Database
Database	
Tables	Collection
row/tuple/record	document (each doc inside a collection can be of diff schema) JSON Object
column	fields of JSON Object
Table JOIN	Embedded Document
Primary Key	Primary Key (_id : string)

Using MongoDb in applications

Table
all records must follow the tableschema

```
Collection (Table)
    {
        id: 1,
Doc     name : "First",
        email : "first@mail.com"
    },
    {
        _id : hflhdf,
        studentName : "",
Doc    gpa : 3.4
        grade : 2
    }
```

Using MongoDb

    1. Embedded Mongo DB (in memory DB)

    2. MongoDb Community Server (download and install)

    3. MongoDb Atlas (Over cloud)

MongoDb Compass : GUI interface :

CLI

mongod : Mongo Db Server : mongod --dbpath "C:\data"

mongo : Mongo Db Client : mongo

Have a location on machine to store data

    c:\data : #needs to specified while launching server

```
mongodb uri :

uri : mongodb://[username]:[password]@[ip]:[port]/<dbname>

Index :

     db.<collection>.createIndex({<fldname> : 1/-1, <fldname> : 1/-1})

1 : asc

-1 :desc

db.<collection>.getIndexes()

<date time > : key criteria
```

Embedded Document : Relationships

SQL

Employee   Address

```
{
    empid : --- ,
    addressid : ----
}
```

MongoDb

Single Employee Collection

```
{
    name:
    email:
    address : [
        {
            location :
            lane :
        },
        {
        }]
}
```

Embedded Id

Employee   Address Collection

All Employess
{
 name : "A"
 // address :[{},{},{},{},{}]
addressid : ["","","",""]

}

{
}.

{
},

@Transactional

    1. By default implement everything in views :

    2. Commit only if all activities are success

    3. insert a new record : get a instance of newly added record

       # change values of that object : change the record in view

MongoDb : ACID Multi Document Acid Transaction (4.0

=> sharded documents ~ RDBMS Views

db.<collection>.start_transaction(s=session); // sharded document

    db.<collection>.insert([{},{}], s);

    db.--------

db.<collection>.commit_transaction();

Oracle Server

SQLPlus : Client (command terminal)

SQL Developer : GUI interface

PL/SQL : Procedural Extension language

portable

performance-oriented

transaction oriented

Structural:

PL/SQL Blocks

Procedure/Function

Anonymous/Named

```
DECLARE

    ---------

    ----------

BEGIN

    -----------

    ----------

EXCEPTION

    -----------------

    -----------------

END;
```

Variable follows the colomn naming convention / oracle type

Variable Declaration

    <name> <type> NULL;

    email varchar(20) NULL;

    id int NOT NULL := 1;

    msg varchar(20) DEFAULT 'Hello all'

Variables can be declared by getting properties from col of table

id employee.id%type :=1;

name employee.name%type;

PL/SQL can have nested scope : thus variables also have scope

```
DECLARE

    val1 number; -- global variables

BEGIN

    -------------------

    -------------------                    nested block

    DECLARE

        num1 number;   -- local variables

    BEGIN

        -----------------------

        -----------------------

    END;

    -----------------

END;
```

Decision Construct

IF-THEN-ELSE

IF <condition> THEN

ELSE / ELSEIF

END IF;

FOR i IN 1..10 LOOP

END loop;

STORED PROCEDURES/FUNCTION

    PARAMETERS

        IN : Input into sub-programs (read-only) : default

        OUT : Output from sub-program (read/write)

        IN OUT: INPUT,OUTPUT (read/write)

```
TEST_PROCEDURE(IN x, OUT y, INOUT z);
a=10;
b=20;
c=30;
CALL TEST_PROCEDURE(a,b,c);
b<--- 50
c<---100

Inside the Procedure
    x : 10
    y : null
    z : 30;
    y=50;
    z=100;
```

RETURN :

allow to return control back from sub-program

SUB-PROGRAM

PROCEDURE : Cannot return value using return stmt (OUT/INOUT)

FUNCTION : Can return values using return stmt (OUT/INOUT)

CREATE OR REPLACE PROCEDURE <procedure> ([parameter])
IS/AS

   ---------------------------- ←---------- Declaration

BEGIN

EXCEPTION

END;

IS : NESTED

AS : TOP LEVEL

```
CREATE OR REPLACE FUNCTION <function name> ([parameter])

RETURN <datatype>

IS/AS--------------------------        <---------- Declaration
        -------------------------
 BEGIN

        ----------------------------

        ----------------------------RETURN "";

EXCEPTION

        ----------------------------

END;
```

```
    a = my_func() + b + c;
```

average | max | min

Employee employee; int max, int min; int avg;

statistics(avg, max, min, employee); // calling stmt

   values will be available

create or replace procedure statistics(avg OUT number, max OUT number, min OUT number, employee OUT employee)

AS

BEGIN

   select average(age) INTO avg from employee;

   IF avg = 0 THEN

      RETURN;

   END IF;

   select max(age) INTO max from employee;

   select min(age) INTO min from employee;

END statistics;

Writing the test cases for our classes/solution

      unit test cases : MAX section : testing each functionality in isolation

      integrated test : test the integration and relationship of a group of services

      End-To-End : Complete application as client

JUnit - API : API to write the unit test cases for java codes

    ==> to organize the test cases

    ==> allow to test a given condition (Assertion)

Group of Testing APIs provided,

=>build on top of JUnit

=>compatible with JUnit

TDD : Test Driven Development

Assertion Based API :

assertj

hamcrest Matcher API

jayway (JSON)

skyscreamer (JSON)

For test cases to run, we need a runner : JUnit Runner

Mockito : MockitoRunner

Dependency needs to be mocked

need a Mock MVC

Mock : does not provide any default implementation of functionality

Spy : Does provide the default functionality , but can be simulated

```
fun(){
    if()
    --  abc()
    for()
    -- xyz()
    --
}
```

Not Recommended for writing test cases for private

=> powermock

powermock-api-mockito

TDD :

AWS :

Regions : AWS service are region scoped

Clustors of Data Center : Availibility Zones
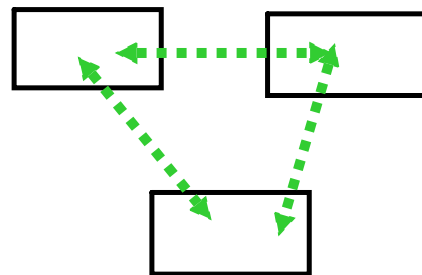
  min 2 data center (3-6)

eg: Regions : us-east-1

    us-east-1a

    us-east-1b

    us-east-1c

Putty : SSH Software allows to connect to an instance remotely

W10 : SSH support already available

>ssh -i .\MyVirtualServerKey.pem MyVirtualServer@54.87.217.81

> Permission : chmod 0400 ---

   private IP : remains same on restart of EC2 instance

   public IP : changes every time re-starts

   elastic IP : does not changes on restart

EBS Volume:

    EC2 instance is terminated : root volume (main drive) looses its data

    Manual / Unexpected Termination

Elastic Block Store

    Network Pen Drive (reusable)

USes network to communicate with EC2 instance (latency)

Bound to AZ

 us-east-1a (EBS)

us-east-1b (cannot be connected)

EBS Volumes can be provisioned for GB, IOPS

4 types of EBS Volumes

    GP2(SSD) : balanced price and performance(root)(1GB---16TB) 3IOPS/GB -- 16000IOPS

    IOI (SSD) : High performing (root) (4GB-16TB) 100IOPS 64000IOPS

    STI (HDD) : Low Cost , large workload , continous long running activities(500GB-16TB)

    SCI (HDD) : Lowest Costing , less frequent access(500GB-16TB)

After Connecting to EC2 instance :

Mount it over Root Volume

AWS document : Mounting EBS Volume on Root Volume