SALESFORCE JOURNEY

---

1) **Login.salesforce.com**
2) **Test.salesforce.com**

Custom Domain → Instance

Density (ORG WISE)

Q) Comfy and compact layout

Compact layout → more space will be available.

Comfy: (comfortable) → expanded view

Theme and branding (ORG WISE)

(Loading change and based on festive theme change)

(Clone option there and just change banner images….)

USER MANAGEMENT AND USER INTERFACE SETTING

Company Information:

1) Fiscal Year
2) Holiday
3) Business Hour
4) Language
5) Data privacy / Encryption
6) System Maintenance
7) Usage-based Entitlement (platform as service)

User:- (top-level security setting)

User Licenses:- (like pass) (What they can view, What they can edit)

1) You can set Profile under it (Standard Permission set)

Profile:

1) System Profile
2) Custom Profile
3) User Profile

Permission set:

Feature licenses (provide additional capabilities to the user to behave in general (functi)

Salesforce App Development:

1) Create DataModel
2) Create Application

3) Create Apex Class (SYNC/ ASYNC)
4) Access Data using SOQL or perform DML
5) Test Class

6) Create Trigger/ PB
7) Test Trigger/ PB

8) Create Lightning Components
9) Create Component with a server side controller

10) Control Access Using Permission Set
11) Use Salesforce Platform API (OAUTH Authentication)

12) Test the App
13) Deploy/ Publish the App

1) AppManager → App Setting (change branding and navigation item and app visibility)
   Console Navigation : Items will come in Menu
   Standard Navigation: item will not come in menu
2) App Menu → Reordering and visibility of app in app launcher
3) Home Page/Any Page --> Visibility and highlighting:
   1) Org Wise
   2) App Wise
   3) UserWise/ Profile Wise
4) App visibility:
   1)App Manager
   2) App Menu

5) Page:- 1) Record type
   2) Lightning record page

Q) To see which record type is used?

A) Set Standard record type field on page layout
   Or in Code `getRecordTypeInfosByName(), getRecordTypeInfosByDeveloperName()`

Q) How to see which page layout is used ?

A) Page Layout Assignment (Check record type and profile)
B) Edit page → preview mode

Q) How to prevent other users from seeing all record types?

Ans) define the access of record type in permission set, otherwise it will take by default that is defined to profile.

Organisation Security: Password, Login, Ip, Trusted Network, Encryption, Remote etc

1) App setting (App Manager → Profile) → **Assign Profile, App Items(tab)**
2) Tab Visibility (App, Profile) → **Permission Set and name change is object plural**
3) Object(CRUD)(Without read, they won't able to see it also)(Delete→see where it is used)
4) Fields (visibility , read, edit) (Searching is not restricted)
5) Record (owd -> sharing (visible, read, edit)(Searching is restricted)
   a) **Role Hierarchy b) Manual Sharing c) Apex Sharing d) Criteria Based Sharing**
6) View All and modify all
7) Page Layout, Compact Layout
8) Search Layout (Global Search, Recent View)
9) Record type (Visibility)
   (only applicable while creating)(can be restricted with sharing rule(permission s)
   (Viewing and updation are not restricted)
10) Component (Visibility)(lightning record page)

Monitoring tool: Health Check, Event Monitoring, Login History, SetUp Audit, Jobs
NOTE: recommended to clone the standard profile and use them.

Fields:

1) Text
2) Text Area
3) Long Text
4) Rich Text
5) Date
6) Time
7) Date/Time
8) Phone
9) Email
10) URL
11) Password

12) Picklist
13) Multi-Picklist

14) Lookup
15) External Lookup
16) Master-Details
17) RollUp Summary
    Note: for N-N create Junction Object

18) Auto Number**(Read Only)**
19) Formula Field**(Read Only)**

20) Field Dependence : Creating Dependence in b/w fields
21) Set Field Tracking : Track the History of the Value Updation

Set Field Security : To see Field level security (visibility and read only)
View Field Accessibility: To see profile level visibility
Where it is Used: To see Dependence

Standard fields:

1) Created by (API: CreatedByID) (Lookup to User) (read only)
2) Last modified by(API: LastModifyByID) (Lookup to User) (read only)
3) Owner (API: OwnerId) (Lookup to User) (Standard Owner Field)(Modifiable)
   (but here we are transferring the record ownership not editing the field)
4) Field Name(Text)(Api: Name) **(Editable)**
   Field Number (AutoNumber) (Api: Name) **(read only)**

Delete Field:

Delete→ undelete it (15 days) (will be in recycle bin for 15 days)

Page Layout:

1) Fields
2) Button
3) Custom Link
4) Quick Action
5) Mobile & lightning Action
   Viewing:- 1) Same Window
   2) New window
   3) New Window with sidebar
   4) pop up
   Type:- Detail Page Link
   Detail Page Button
   List Button
6) Related List:

Field Set: Group[of custom field but doesn't work in lightning

Search layout: Changing the layout of searching and **recent view** layout also gets changed

---

RelationShip:

Master-Detail relationship:

1) Created on Child Object
2) Must necessary have one parent object
   (Parent gets deleted object also gets deleted)
3) Roll-Up Summary Created on Parent object.
   Sum and function show on parent object (read only)

LookUp relationship:

1) No need of parent object
   (just a lookup to jump on other record)
   (record Won't get deleted when other object get deleted)

Formula: (read only) :

Validation Rule:

---

1) Validation Rule: Validating the Data(who is modifying, what data is getting entered)
2) Filter: Filtering out Data
3) Formula: MatheMatic Calculation
4) Assignment:
5) Approval Process:
6) Escalation:
7) Relation: Data (Schema) Relationship

It's Like Hulchl or Automation to provide or to deal with Customer engaging

Process Automation:-

Process Builder:

Deactivate ---> Same purpose, come under version

When to fire:

a) Record changes**(create, update) (UpdateWhat(particularFieldorAny field2fire)**
b) Received outbound message
c) Invoke by another process

Process Rule:-

1) Which Object
2) What record Criteria to fire action

Process Action:-

1) Immediate
2) Schedule

Email Notification
Post to Chatter
Quip
Approval
Call Other Processes

Need to define criteria (not necessary) what record you wanna update and with which record field value you wanna update (necessary).

Note: (best approach to use trigger in case of value update with reference)

1) **Restricted to execute when only condition meet.**
2) **Reevaluate the workflow/ process builder if two work flow working together.**

WorkFlow:-

Workflow rules and Workflow action:-

Workflow rules:-

1) Criteria when to fire workflow**(Create,Update,UpdateWhat(particularFieldorAny field2fire)**
2) Which Object to it is

Workflow action:-

a) **What record Criteria to fire action**
1) TimeBased
2) Regular

Creating a task: When people meet a Certain cr. Task needs to create to perform
Field Update: Updating a field based on certain cr.
Email Alert: when records meet certain cr. give an email alert.

Ex: a customer is not engaging with company create a email alert for agent
Ex2: a customer is engaging with more create a private VIp alert to manager

Outbound Message: Sending msg to external system.

Tool to monitor action: Monitoring time based workflow

ScreenFlow:

UI:

Create,Update,Get, ExternalID(Access External System Apex Action)

UI	For(Loop)	CRUD	Triggering Action
	IF	Get, Update, delete, Create	Triggering other flow
	Operator(Assignment)		

Key: Screen flow… Loop (Each item, last item) and back from action also.

Note:-

JSON format, Variable format/ assignment (Primitive/ Non Primitive(record type).
Object/ Record ID(variable must be recordId), Global Constant(True/False),
Action: Post to chatter, Submit for approval, log a call, email notification *etc.*
Resources Manager, Debug/Run, Input/output field available to pass record id/etc.

Code:

UI: Visual, Aura, LWC

Apex: Pretty Much Everything

Debugging: Process Builder, Workflow:--- saving and activate
Screen flow: debug option is there
DeveloperConsole: Debug log

Note: Strictly restricted when to execute the flow in specified condition.

Get→ Record // Processed in Loop.

Update Record → Outside Loop and one shot with Collection of record /Assignment Processed for which to update.

Record Id ⇒ Available for input

Criteria: Criteria that cause the workflow rule to run
Object: Object on which you need to perform an action.
Record Criteria:
Action: Immediate actions

Approval Process:

Criteria (What, Whois approving, All approval require)

Initial set of Action:

When a record is submitted for approval, what action need to trigger initially

Action By Concerned Person:

Approved	Reject
Approved Action	Reject Action

Final: (All Done)

Final Approval	Final Rejection

Recall for editing

---

1) Invoking other process/ Flow
2) Invoking Apex Class
3) Email/ Mobile Notification/ QUIP
4) Sending Outbound Message

---

1) Visual Force Page, Aura Component, LWC
2) WorkFlow, Process Builder
3) Screenflow
4) Classic, Lightning

Apex:- (need for business logic implementation)

Apex is a strongly typed, object-oriented programming language that allows developers to execute flow and transaction control statements on Salesforce servers in conjunction with calls to the API.

Apex code can be initiated by Web service requests and from triggers on objects.
Developers write and save Apex code to the platform, and end users trigger the execution of the Apex code.

Apex enables developers to add business logic to most system events, including button clicks, related record updates, and Visualforce pages.

Apex is closely bound with Database. (Any issue with DML cause failing in execution)

Insertion
Deletion
Updation
Searching
Sorting
Traversing

Node
Constructor

Add: adding an element to the end of the list.
Get: get the element of that index value
Set: set the element of that index value
Put: put the value and index

Key to Hack:
Strongly Typed: you need to define type of variable.
If you don't define a variable it will have **null value.**
isNull ⇒ String not having any value and not even space
isBlank => String is not having any value but space can be there

---

Concat : +
List initialize: { }
E.g: List<Integer> li = new List<Integer>{1,2};
System.debug(li);

Map initialize: {‘ ’ => ‘ ’}
E.g: Map<String, String> mi = new Map<String, String>{'a'=>'raam'};
System.debug(mi.get('a'));

SOQL Dynamic: =:
Comment: ***//*** **,** ***/* */***

---

classname.methodName ===> need to define method as static
Object_of_Class.method Name ⇒   no need to define method as static

---

Constructor : no need to define any return data type/ void and static
Method include 6 parameter:
Public static/ with sharing virtual/abstract return type method_name(){}
private/ protected global without sharing final

1) public: public accessible within namespace
2) private: not accessible outside of that class
3) Protected: inner classes accessible
4) global: globally accessible all over to any org (Main purpose Web service call out)
Move From top to bottom, if bottom is restricted than it doesn't matter what top is.
By default method or variable is `private`.

5) static: referring to that part/ box only
6) transient: temp value holder

6) with sharing: Sharing rule will also get execute (User Mode)
(Object/field/ record level security of that particular user enforce)
(fail if user doesn't have access on any field of that object)
7) without sharing: Sharing rule will not get execute (System mode)
8) Inherited sharing : will take security from parent or from that class which is calling

8) virtual: method overridden
9) abstract: only definition given not declaration (need to give definition)
Note: virtual is not allowed on constructor

10) final: final / constant

Operator:

1) = Assignment
2) == Equal (ignore case sensitivity)
3) === Type Equal to
4) | or operator (Both value compare)
5) & And operator (Both value compare)
6) || Logical Operator (Short circuit value) (Depending on first value outcome)
7) && Logical And Operator (Short circuit value) (Depending on first value outcome)
8) += equal and plus to
9) ++ increment
10) -- decrement
11) ** power operator
12) ? : Ternary operator

Topic in basic Apex:

1) **Class, Object**
2) **Method (void, primitive, non primitive:- data type) (name) (value)**
   Access, Static method, constructor
   E.g: public static void method_name(){
   return;
   }
3) **For loop,enhanced for loop, While loop, do loop**
4) **Switch statement**

```
switch on expression {
    when value1 {        // when block 1
        // code block 1
    }
    when value2 {        // when block 2
        // code block 2
    }
    when value3 {        // when block 3
        // code block 3
    }
    when else {            // default block, optional
        // code block 4
    }
}
```

E.g:

```
Integer i = 2;
switch on i {
  when 1 {              // when block 1
    // code block 1
    System.debug('1');
  }
  when 2 {              // when block 2
    // code block 2
    System.debug('2');
  }
  when 3 {              // when block 3
    // code block 3
  }
  when else {                   // default block, optional
    // code block 4
  }
}
```

sObject instance of Account that is a

```
switch on sobject {
    when Account a {
        System.debug('account ' + a);
    }
    when Contact c {
        System.debug('contact ' + c);
    }
    when null {
        System.debug('null');
    }
    when else {
        System.debug('default');
    }
}
```

5) **If condition**
6) **Primitive data type**

ID: 18 character (sensitive, non sensitive included) 15 char for case sen.

(Integer(32 bit), long(64bit), double(64 bit decimal), decimal(32 bit decimal) , (string, boolean(true, false, null), date, time, dateTime)

String/ Date/ Time/ Date-Time is a Predefined class.

Standard of primitive data type is : Object

Standard of non primitive data type is : sObject

E.g:

```
Object i = 0;
System.debug('i'+i);

Object str = 'i am good';
System.debug('str'+str);

Object d = 21.3;
System.debug('d');

Integer a;
Integer a =5;

List<Integer> li = new List<Integer>();
List<Integer> li = new List<Integer>{1,2,3};

Account a = new Account();
Custom__obj co = new Custom__obj();

List<Account> li = [select id from Account];
```

7) **Collection data type, Method of collection data type**: popup show(**documentation**)

List, Set, Map :

Map: Key should be primitive data type and should be unique.

All Variables are by default initialized to NULL value even boolean type also if value is not defined.

8) **Standard Class/ Custom class (salesforce object)**
9) **Standard class member accessibility, function**

Static Case:

ClassName.attribute;

ClassName.MethodName();

Non Static Case:

Instance.attribute;

Instance.MethodName()

10) **Type, Type conversion**

Object i = 10; Integer j = (Integer)i;

Type:
How to see DataType of a Variable
Popup show
Type Conversion:
Type Conversion is pretty tricky:--

1) String to integer or integer to string
   (value of) e.g: Integer.valueOf(st);
2) List to Set or set to list :
   E.g: new set<string>(li); here need to pass

Some special function :

1) equals
2) indexOf
3) Split and Join
4) Final Keyword (Constant)

```
static final Integer PRIVATE_INT_CONST = 200;
static final Integer PRIVATE_INT_CONST2;
```

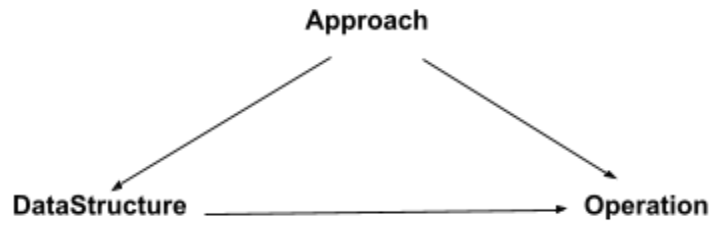
Trying to change the value of final variables will give you the error. Use it to assign only

---

10) List of structure
11) List of primitive data type
12) Nested List
13) Method defined with list of structure
14) SOQL, Dynamic with variable, Loop

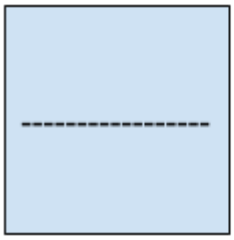
Application
component
Package
module

Class
Function
For
If {



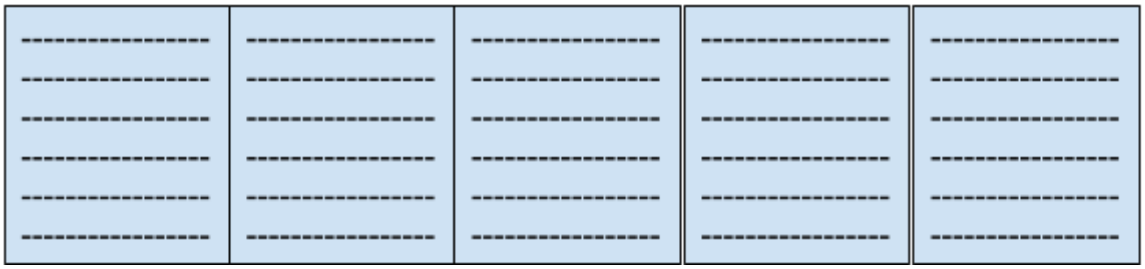
}

NUMBER



IIST OF PRIMITIVE DATA TYPE

IIST OF NON PRIMITIVE DATA TYPE



Operation:
iNSERTING, DELETING, SEARCHING, UPDATION, SORTING, COUNTING, MERGING

Add, remove, get, set, size, clear, index, typeOf

Push, append, insert
Pop, remove, delete, clear
Get, set
Sort, reverse
Size, length

Salesforce documentation is the best place to get information.

Best Practical/ clearance:

---

1) **Static Method**
   Can call with the class name
2) **Without static Method**
   Need to define object first then can call
   E.g:

```
public class FunctionCalling {
    public static final Integer i = 10;
    public static Integer i1 = 1;

    public static void main1 (Integer i){
        Integer i1 = 0;
        System.debug('Integer1:'+i1);
    }
    public void main2(Integer i){
        Integer i2 = i;
        i2 = 5;
        System.debug('Integer2:' + i2);
    }
}
```

```
---------------------------------------------------
FunctionCalling.main1(50);
FunctionCalling f = new FunctionCalling();
f.main2(50);
```

---

3) **Method calling outside the class**
   Applies static and non-static run and call either with class name or object instance
4) **Method calling in the same class**
   Applies static and non-static run and call either with class name or object instance
   Flexibility is there, without a class name you can also call.

E.g:

```
public class FunctionCalling {
    public static final Integer i = 10;
    public static Integer i1 = 1;

    public static void main1 (Integer i){
        FunctionCalling fc = new FunctionCalling();
        fc.main2(10);
        Integer i1 = 0;
        System.debug('Integer1:'+i1);
    }
    public void main2(Integer i){
        //FunctionCalling.main1(5);
        Integer i2 = i;
        i2 = 5;
        System.debug('Integer2:' + i2);
    }
}
```

```
-------------------------------------------------------------
FunctionCalling.main1(50);
//FunctionCalling f = new FunctionCalling();
//f.main2(50);
```

---

5) **Variable defining and accessing:**

Inside a method:

Defining:

Can not define a variable inside a function with static, final, public.

The normal variable declaration you can do.

E.g:

```
public void main2(Integer i){
    //FunctionCalling.main1(5);
    Integer i2 = i;
    i2 = 5;
    System.debug('Integer2:' + i2);
}
```

Accessing: We can not access variables outside of a method.

Inside a class or outside a method: (Like a function declaration, it works)

Defining:

Two ways:

1) **Declare with final :** You can just use the variable to get the value of that variable.
2) **Declare without final :** The value will change based on method execution

E.g:

```
public class FunctionCalling {
        public static final Integer i = 10;
        public static Integer i1 = 1;
}
```

Accessing:

Accessing outside of the class:

E.g:
```
public class FunctionCalling2 {
    public static void main1(){
        System.debug(FunctionCalling.i);
    }
}
```

Accessing inside a method of that class:

(flexibility is there without a class name you can call that variable)

E.g:

```
public class FunctionCalling {
    public static final Integer i = 10;
    public static Integer i1 = 1;
    static Integer i2 = 6;
    public static void main1 (){
        i1 = 5;
        i2 =8;
        System.debug('Integer1:'+i2);
        System.debug('Integer1:'+i1);
        System.debug('Integer1' + i) ;
    }
    public static void main2(){
        //FunctionCalling.main1(5);
        System.debug('Integer1:'+i2);
        System.debug('Integer2:' + i1);
    }
}-
//FunctionCalling.main1();
FunctionCalling.main2();
//FunctionCalling f = new FunctionCalling();
//f.main2(50);
```

Example: AND Some Advance Basic Cover

```
List<Integer> li = new List<Integer>();
li.add(1);
li.add(2);
System.debug('List'+ li);
System.debug('List first' + li[0]);

List<Customer__c > lic = [select id, customer_status__c from Customer__c Limit 2];
System.debug(''+ lic);

Map<Id, Integer> mi = new Map<Id, Integer>();
mi.put('a012y000009JzalAAC',1);
System.debug('Map'+ mi);
Map<Integer, List<Customer__c >> mic = new Map<Integer, List<Customer__c >>();
mic.put(1,lic);
System.debug('Map'+ mic.get(1)[0].id);
```

List: Collections of Items and iteration through List name with Index value of that item

() :::: li[0]

({},{}) ::::: li[0].id

Trigger.new ::::: ({},{}) ::::: li[0].id

Trigger.old ::::: ({},{}) ::::: li[0].id

Map:

Map{a012y000009JzalAAC=1}

Map{1=(Customer__c:{Id=a012y000009JzalAAC, Customer_Status__c=Active},
Customer__c:{Id=a012y000009JzboAAC, Customer_Status__c=Active})}

```
System.debug('Map'+ mic.get(1)[0].id);
```

Plain SOQL Query:: ({},{})

SOQL Query with parent/child relationship: Consider Invoices__r as a element and proceed

```
List<Customer__c > lic = [select id, customer_status__c,(select id from invoices__r) from
Customer__c Limit 2];

Map<Integer, List<Customer__c >> mic = new Map<Integer, List<Customer__c >>();
mic.put(1,lic);
System.debug('Map'+ mic.get(1)[0].invoices__r);
```

Index

Type ::::

Variable_name

:::: pointer with index ======> gives u value

:::: Pointer without index ============> doesn't give u value shows only it is a pointer Value

---

Loop Iteration:

Enhanced for loop: it goes by index value with Variable name means li[0]

Variable name with index value always give the value
If index is always 0 then variable name give the value
If index is varying then need to define index value
If member there in any particular index then need to define variable name also

Integer a = 5	System.debug(a)
List<Integer> li = [1,2,3]	System.debug(li[0])
List<account> li = [select id from account]	System.debug(li[0].id)
Map<key, value> mi = {1:2};	System.debug(mi.get(1));
List<Json> ji = {1:2} :	System.debug(li.1);
List<Json> ji1 = {{1:2}} :	System.debug(li[0].1)

```
List<Integer> li = new List<Integer>();
li.add(1);
System.debug('List'+ li);
System.debug('List first' + li[0]);
```

Li[0] : Variable Name with Index ⇒ give you value

JSON = {‘type’ : ‘ram’, ‘category’:’shyam’}
JSON IS the variable name and index is always 0 so in this case json give us the value
JSON.type give us the value.

```
List<Customer__c > lic = [select id, customer_status__c from Customer__c];
System.debug(''+ lic[0]);
System.debug(''+ lic[0].id);
```

Lic[0] : index with variable name give u the value
Id is specific member that is accessible with . give u value.

() ::::: Different-2 index value / passing value or by reference
{} ::::: JSON value related to one/ that particular object/ index values/ value defining
[] ::::: index
: :::: assigning value to same index value / value is going inside of the box (this value)
= :::: assigning value to a different index value/ Assign/ pointing to the box
. :::: iterating that variable name /Member refer: Pointing to the members of box

---

Version Setting:

Version is used to tell which classes version of Apex Need to use in
SOAP API Callout, in Managed package
Apex Class → Version → Package
Note: Every Class and Trigger name should be unique regardless to the Version

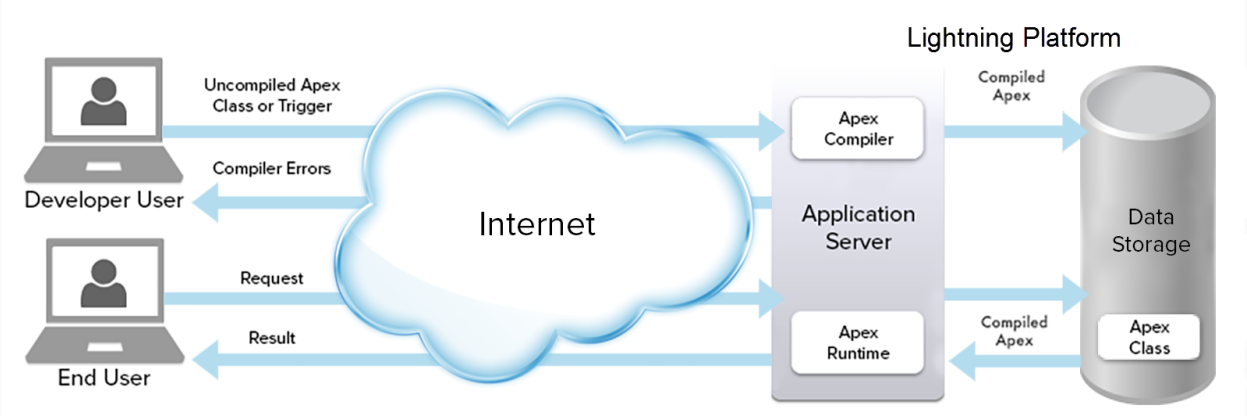
Create Copy of Org
Setup -> Sandbox -> define full/ partial etc

Adding Apex Classes to the package:
All Apex classes that are adding to the package must have cumulative 75% test coverage.
All Test Class run by default and cumulative it counts 75% test coverage.
Manual: run all and system method is there also to trigger

While the package is installing you can also specify which particular test class should run
By annotating: `@isTest(OnInstall=true)`
For a successful package installation and deployment, all test classes must cover 75% code coverage cumulative.

DataType:- (Common)
List: Name. index, value
Map: Key, Value
JSON Format: Index, Key, Value
Type: Lightning Component
SObjectType : Dynamix Apex

Every statement needs to be ended with a semicolon.
Group of statements comes in curly {} braces.



Basic of Apex Programming completed

Apex Code → Apex Compiler → Metadata(Compiled Instruction) ← Apex Runtime interpreter ← trigger
Keynote:
Write your Code
Test your Code
Deploy your code

Advance Apex Programming

Blob:

The Blob is a collection of Binary data which is stored as object. This will be used when we want to store the attachment in salesforce into a variable. This data type converts the attachments into a single object. If the blob is to be converted into a string, then we can make use of the toString and the valueOf methods for the same.

A collection of binary data stored as a single object. You can convert this data type to String or from String using the `toString` and `valueOf` methods, respectively. Blobs can be accepted as Web service arguments, stored in a document (the body of a document is a Blob), or sent as attachments. For more information, see Crypto Class

Final / Constant

Value can not change.

Public static final integer i = 5;

Transient

variables that can't be saved, or dont take any permanent variable(Dynamic)

Use In Visual Force page or DateTIme(Where value need to change regularly)

```
Transient Integer currentTotal; mainly use in dateTime instance
```

instanceOf

Check whether that instance belong to particular class or not

Book__c a = new Book__c();

boolean result = a instanceOf Account;

System.assertEquals(false, result);

Super()

Only classes that are extending from `virtual` or `abstract` classes can use `super`.

Super is like an instance of the virtual or abstract class.

To call constructor with super()

Constructor can be called inside a constructor of extending class only.

And only one constructor at a time can be called if more than one constructor defined

To call variable with super()

To call method with super()

E.g:

```
public virtual class SuperCla {
        public SuperCla(){
                System.debug('Here is the first constuctor');
      }
       public SuperCla(String x){
                System.debug('here is the second constructor'+ x);
      }
   public void myname2(){
        System.debug('This is good to go');
        }
    public Integer myinterger =5;
     public virtual void myname(){
        System.debug('good to proceed with virtula my name method');
}}
```

```
public class SuperCal2 extends SuperCla {
        public SuperCal2(){
                super('String');
                System.debug('this is the constructor of supercal2'); }
public void main_cla(){
        super.myname2();
        System.debug(super.myinterger);  }
public override void myname(){
        System.debug(super.myinterger);
         super.myname();
        System.debug('good to go');   } }
```

This:

1) Referring to variables of the class but can not use in static context

2) Passing the value in constructor of the same class **(Note you can not use Super keyword)**

```
public class ThisClass {
        public String s;
        public ThisClass(String s1)
        System.debug(s1);}
  public ThisClass(){  this('Dhanuka');     }
  public void main(){
         this.s = 'Shubham';
         System.debug(this.s);
 } }
```

NODE:

hasNext()

Next()

Try/ Catch/ Finally:-

Try:

Put pieces of code that can cause problems in future.

Catch: Generic (Exception e) ⇒ e.exceptionType, e.getMessage

Finally:

False Case:

if an exception occurs where to roll back & what operation needs to go check and go back if multiple operations are performing.

True case:

```
if (connection != null) connection.close();
```

Example:

```
public  class ParentClass {
public static string s;
public static void parent_method(){
 Try{
 system.debug('small case'+ s.isAllLowerCase()); }
catch(Exception e){  System.debug(e.getMessage());
}   } }
```

Interface: (Implement Define Functionality of where, what need to be done)

```
Public interface interface_name{
// Declare method here }
Public class class_name implements Interface_name{
// Define method here
}
```

Abstract-OverRide: (Need of implementation of every method/definition of every method)

```
public abstract class ParentClass {
public abstract void parent();
public void parent_method(){   System.debug('Parent class method ');    }
}----------------------------------------------------------------------------------------------------
public class ChildClass extends ParentClass{
        public override void parent(){       }
        public void childMethod(){   ChildClass PC = new ChildClass();
PC.parent_method(); }  }
```

Virtual-OverRide: **(No need of override every method/ definition already given)**
But if you are using the same method name then need to override it.
Variable can not be declare as a virtual only method

```
==========================================================
public virtual class ParentClass {
   public virtual void parent_method(){
      System.debug('Parent class method ');
   }
}
==========================================================
public class ChildClass extends ParentClass{
public override void parent_method(){
      system.debug('Child Class Mehod overridden1');
 } }
==========================================================
```

Inheritance:(Taking property of parent class)

```
Public virtual class My_Virtual{
        Public virtual void My_Method(){
        }
}
Public sub_Class extends My_Virtual{
        Public override void My_Method(){
        }
}
```

InnerClass: **You can go inner to outer but can't go outer to inner.**

```
public class InnerClass {
            class innerc{
               public void show(){
                  System.debug('Here u go innner class method');
               }
            }
            public void outerclassMethod(){
               innerc ic = new innerc();
               ic.show();
            }
        }
```

InnerClass.innerc; Not allowed
InnerClass.outerclassMethod(); is allowed

Constructor: **No static, No type**

```
Public class good{
        Public good(){

}
}
```

Note:

constructor is useful when we want to initialize variables at the begging of the class.
Note that the constructor does not automatically call, it calls when we define the instance of the class.
Now constructor can be void and parameterized. In both cases we need to define instances accordingly.

Class:

```
Public class Class_name{
}
```

Note:

You must use one of the access modifiers (such as `public` or `global`) in the declaration of a top-level class.

You do not have to use an access modifier in the declaration of an inner cl

While declaring an object… you are calling by default constructor.

Method Member accessibility:-

You can not access the member of a function

To Access the member of a function define function like get and set :

1) Set: return type void and pass the parameter
2) Get : return type not void and dont pass the parameter

Example:

```
public void setRight(String rt){
    this.right = rt;
}

public Integer getData(){
    return this.data;
}
```

Class Member Accessibility:-

To access the member of a class

Use **This keyword** to refer to the attribute if you are using within the class

Else classname.attribute_name

Classic → Show Dependence: To show dependent items
Search Files : To see where it is used.

Best Practice:

1) **Don't Use Future Method inside loop.**
2) **Don't use Updation / Database Operation inside loop.**
3) **Particularly reference id/ record which u wanna update.**
4) **To stop recursive in trigger and code use static variable or create flag**
5) **One trigger on one Object and Check ASYNC Operation before Operating.**
6) **Test class : System.runAs to Avoid mixed DML Operation**
7) **Test class coverage : Minimize or particularly use Try / Catch**
   And Put Logic to execute Try Block and Also Put Logic to execute Catch Block
8) **Null Value Condition Need to Check when processing data.**
9) **Sharing Rule Must be created(with sharing / without sharing)**
10) **Maximum stack: same name of function in javascript and apex controller(follow naming convention)**
11) **Method Member accessibility, Define get and Set**
12) **Method only can be pass in JavaScript as a reference (in 3 ways)**
13) **After validation rule, run test classes**

APEX Automatic Variables Property:
Setting Get and set value

Older way:--

```
public class GetSet {
        public static Integer i=0;
        Public void set(Integer j){
                i =j;
        }
        Public Integer get(){
                return i;
        }
}
GetSet v = new GetSet();
System.debug('i initialize' + v.get());
v.set(5);
System.debug('i after ' + v.get());
```

```
public class AutomaticProperty {
public double MyReadWriteProp { get; set; }
}
AutomaticProperty ap = new AutomaticProperty();
ap.MyReadWriteProp = 5;
System.assertEquals(5, ap.MyReadWriteProp);
```

Dynamic Apex

---

1) MetaData Information

(Object , Fields, PickList Value(No Need to use hard coded value use metadata to work for you)

2) Dynamic SOQL

Standard Class: **Schema**

Method Available:

1) **describeGlobal() : retrieve a list of all Objects in the org**
2) **describeSObject() : retrieve metadata about individual object**
3) **getGlobalDescribe(): returns map of all object and token for standard and custom**
4) **getDescribe(): Retrieve metadata about an ind field**

Example:

```
List<SObjectType> li = Schema.getGlobalDescribe().values();
for(SobjectType s: li){
   System.debug('List'+s.getDescribe().getName()+ s.getDescribe().fields.getMap().Values());
}
```

Example2:

```
Schema.DescribeSObjectResult acc=  Account.sObjectType.getDescribe();
System.debug('acc' + acc.fields.getMap().values());
Schema.DescribeSObjectResult acc=  Account.sObjectType.getDescribe();
System.debug('acc' + acc.fields.getMap().values());
for(Schema.SObjectField s: acc.fields.getMap().values()){
   System.debug(s);     }
```

Example3:

```
List<Schema.PicklistEntry> cus
=Schema.sObjectType.Customer__c.fields.Customer_Status__c.getPickListValues();
System.debug('Cus' + cus);
```

5) **getRecordTypeInfos() : record type name**
6) **getRecordTypeInfosByDeveloperName() : record type api name**
7) **getChildRelationship() : child relationship for subject describe**
8) **getPickListValues(): picklist values**
9) **getName(): field name**
10) **getLabel(): label name**
11) **getDefaultValue() : value**
12) **getDigits() : digit**

E..g: How to get all the field and related information of that object

```
Map<String, Schema.SObjectType> schemaMapOdAllSObject = Schema.getGlobalDescribe();
```

```
Map<String, Schema.SObjectField> MapofdesiredObject  =
schemaMapofAllSObject.get('HWS_Product_Serviceable_Sales_Item__c').getDescribe().fields.getMap();
for(Schema.SobjectField sObjectField : MapOfDesiredObject.Values()){
        Schema.describefieldresult dfield = sObjectfield.getDescribe();
        System.debug(dfield.getName()+';'+dfield.getLabel()+';'+dfield.getLength()+';'+dfield.get
        Type()+';'+dfield.isUnique.+';'+dfield.isExternalId()++';'+dfield.isNillable());
}
```

Working on Provided Advance Salesforce Standard Class

---

1) **Email Messaging Services**
2) **UserInfo.getUserInfo()**

Email Services:

1) Inbound services
2) Outbound services

   In outbound there are two type:

   1) Single message servicing
   2) Mass message servicing

For Outbound:-

To send Email first we need to set OWD Email address:

Setup => org wide email

Example:

```
Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
mail.setToAddresses(‘’);
```

you can also provide a list here

Example:

```
String[] emailaddresses = new String[]{‘shubham@gmail.com’};
```

Or

```
List<String> li = new List<String>();
li.add(‘dhanuka@gmail.com’);
mail.setToAddress(li);

mail.setReplyTo('');
mail.setSenderDisplayName('');
mail.setSubject('param1');
Messaging.sendEmail(mail);
```

Example:

```
Messaging.MassEmailMessage mem = new Messaging.MassEmailMessage();
```

FOR inbound:

1) Class need to be global
2) Need to implement Messaging.InboundEmailHandler

Example:

```
global class email implements Messaging.InboundEmailHandler{


}
global class Email implements Messaging.InboundEmailHandler {
    global Messaging.InboundEmailResult test (Messaging.InboundEmail em,
Messaging.InboundEnvelope env){
        Messaging.InboundEmailResult result = new Messaging.InboundEmailResult();
        // value we want to store3
        String name = em.fromName;
        String email = em.fromAddress;
        String body = em.plainTextBody;
        Account a = new Account(Name = name);
        insert null;
    }

}
```

//Check Code

Now to receive email services we Need to create Email Services:

Setup ⇒ email services (define class and email which u are expecting to send email)

We need to set that particular address with the salesforce provided address.

Trigger: (Run time Context to perform changes in certain record using dml)

Why do we need triggers??

Ans: Because triggers allow us to do changes in record before and after event operation.

Event on which a trigger can fire:

1) Insert
2) Update
3) Upsert
4) Delete
5) Undelete
6) Merge

Operation could be before and after the event Operation.

1) Before insert, before update, before delete
2) After insert, after update, after delete,
3) After Undelete

Accessing of Record for before and after operation: (Context Variables)

Trigger.new:
Trigger.old :
Trigger.oldMap:

Example:

```
trigger trigger_name on Object_name (before, after events){
        for(Object_name  o : Trigger.new){

        }
}
```

Note:

1) Inside enhanced for Loop don't use any SOQL St., While iterating use SOQL Statement
   It will iterate values one by one from the quired list.
   E.g: **: in loop it just iterated through value. (type, name, value not index)**

```
for(Account ac: [select id from Account]){
        System.debug(ac.id);
}
```

   If Nested for loop schnerio used Map to mapped
2) Upsert Trigger fire in Both case before and after(insert and update)
3) Merge Trigger fire in both case before and after(delete), before (update)
4) Field history is not record until Trigger process is done
5) Web Callout Trigger should be implemented ASYNC
6) One Trigger on One Object
   (And Written only on those object which fire trigger based on their record changes)
7) After Trigger Operation, record will be only in read mode

To get Run time Context about what is happening:

System.Trigger

isInsert: True if trigger is fired due to insert operation
IsUpdate: True if trigger is fired due to update operation
isDelete: True if trigger is fired due to delete operation
isExecuting: True if it is fired because of apex code
Not visual force page, web services, api call
isBefore: True if triggered was fired before record saved
isAfter: True if triggered was fired after record saved
isUndelete: True if triggered was fired after record is recovered from recycle bin

E.g:

```
trigger InvoiceGenerattor2 on Customer__c (before insert, before update) {
    if(Trigger.isBefore){
        if(Trigger.isInsert){
            System.assertEquals('expected', 'actual');
        }
    }
}
```

To get Record List on which we need to fire or process something**(CONTEXT Variable)**

New: new version of sobject list record id (iist)
newMap: new Version of sObject RecordId (map)
Old: old version of sObject list of record id(list)
oldMap: old version of sObject RecordId(map)
size(): total number of record (before and after all operation)

Note:

Before insert in trigger and insert in class will cause some error:

1) Id can not be assigned in insert operation
2) Flag need to set else will go in recursive mode
3) Can not use insert as it will cause duplicates in the system.

So just do updation in class and leave as it is:-

```
public class Book {
    public static Boolean isFirstTime = true;
    public static void main_book_method(List<Book__c> bc){
        for(Book__c b1 : bc){  b1.price__c = b1.price__c * 0.9;
        isFirstTime = false;
    }
}
trigger Book on Book__c (before insert) {
    if(Book.isFirstTime ==true){
    Book.main_book_method(Trigger.New);
    }
}
```

Example:

```
trigger InvoiceGenerattor on Customer__c (after insert, after update) {
   List<Invoice__c> listOfInvoices = new List<Invoice__c>();
   for(Customer__c cus:Trigger.new){
          if(cus.Customer_Status__c=='Active'){
          Invoice__c IC  = new Invoice__c ();
          IC.Customer__c = cus.Id;
          Ic.Amount_Paid__c = 200;
          Ic.Description__c ='This is good to go, we are awesome';
          listOfInvoices.add(Ic);
          }
   }
   Database.insert(listOfInvoices, false);

}
```

ISSUE:

This will alway fire the trigger when any field of that record changes and will create the invoice record.
(Restrictive, Recursive, re-evaluate)

How to make it to fire when only certain conditions related to that field changes(how to make it efficient)? So that it wont fire every time?

Ans: For that we have a Context Variable to get New Version of records that are modified.
Use Static variable outside the method in the class.
E.g Class:

```
public class Book {
   public static Boolean isFirstTime = true;
   public static void main_book_method(List<Book__c> bc){
      for(Book__c b1 : bc){
                  b1.price__c = b1.price__c * 0.9;
          }
            isFirstTime = false;
   }
}
```

Trigger:

```
trigger Book on Book__c (before insert) {
   System.debug('hre we go');
   System.debug(Trigger.new);

   if(Book.isFirstTime ==true){
   Book.main_book_method(Trigger.New);

   }
}
```

Trigger.new: gives a new version
Trigger.old: gives a old version
Trigger.oldMap: gives a old version of record with ID
Trigger.newMap: gives a new version of record with ID

How to make it so that only a particular field value changes, then it will fire the trigger?
(it will satisfy the condition that is fine but value should change at that moment not only satisfy the condition and make a record)

Ans: Condition in old and new need to set.

E.g:

```
trigger InvoiceGenerattor on Customer__c (after update) {
   List<Invoice__c> listOfInvoices = new List<Invoice__c>();
   for(Customer__c cus:Trigger.old){
            if(cus.Customer_Status__c!='Active'){
         for(Customer__c cus1:Trigger.new){
           if(cus1.Customer_Status__c=='Active'){

                        Invoice__c IC  = new Invoice__c ();
                        IC.Customer__c = cus1.Id;
                        Ic.Amount_Paid__c = 200;
                        Ic.Description__c ='This is good to go, we are awsm';
                        listOfInvoices.add(Ic);
              }
   }
  }
}
   Database.insert(listOfInvoices, false);
  }
```

Or you can write if condition together

```
trigger InvoiceGenerattor on Customer__c (after update) {
   List<Invoice__c> listOfInvoices = new List<Invoice__c>();
   for(Customer__c cus:Trigger.new){
            if(cus.Customer_Status__c =='Active' && Trigger.oldMap.get(cus.Id).Customer_Status__c != 'Active'){

                        Invoice__c IC  = new Invoice__c ();
                        IC.Customer__c = cus.Id;
                        Ic.Amount_Paid__c = 200;
                        Ic.Description__c ='This is good to go, we are awsm';
                        listOfInvoices.add(Ic);
              }

}
   Database.insert(listOfInvoices, false);

}
```

Note: This will cause an issue while inserting and status = active

To fix this :

1) **Either use separated insert and update trigger**
2) **If u are using in same then use::::**

```
trigger InvoiceGenerattor on Customer__c (after insert, after update) {
   List<Invoice__c> listOfInvoices = new List<Invoice__c>();
   for(Customer__c cus:Trigger.new){
      if(Trigger.oldMap != null ){
            if(cus.Customer_Status__c =='Active' && Trigger.oldMap.get(cus.Id).Customer_Status__c != 'Active'){
                  Invoice__c IC  = new Invoice__c ();
                  IC.Customer__c = cus.Id;
                  Ic.Amount_Paid__c = 200;
                  Ic.Description__c ='This is good to go, we are awsm';
                  listOfInvoices.add(Ic);
              }
      }else{
         if(cus.Customer_Status__c =='Active'){
            Invoice__c IC  = new Invoice__c ();
                  IC.Customer__c = cus.Id;
                  Ic.Amount_Paid__c = 200;
                  Ic.Description__c ='This is good to go, we are awsm';
                  listOfInvoices.add(Ic);
         }
      }
}
   Database.insert(listOfInvoices, false);

}
```

How to stop recursive && Async apex calling ?

Ans: use flag

E.g:

Trigger Design Patterns:-

1) **Bulk Trigger Design Patterns:-**
   By default all trigger are for bulk trigger, just keep in mind governor limit
   List to update in SOQL outside to loop
2) **Trigger Helper Class:-**
   Do interfacing in short, don't write the code inside trigger, writer apex class and in apex class, call this apex class in trigger.

```
E.g: --
trigger InvoiceGenerattor on Customer__c (after insert, after update) {
   for(Customer__c cus:Trigger.new){
      if(Trigger.oldMap != null ){
            if(cus.Customer_Status__c =='Active' && Trigger.oldMap.get(cus.Id).Customer_Status__c != 'Active'){
         Id id= cus.Id;
         NewClass.main(id);
              }
      }else{
         if(cus.Customer_Status__c =='Active'){
            Id id= cus.Id;
            NewClass.main(id);
         }
      }
}
}
```

```
public class NewClass {

   public static void main(Id id){
      List<Invoice__c> listOfInvoices = new List<Invoice__c>();
            Invoice__c IC  = new Invoice__c ();
                        IC.Customer__c = Id;
                        IC.Amount_Paid__c = 200;
                        IC.Description__c ='This is good to go, we are awsm';
                        listOfInvoices.add(IC);
      insert listOfInvoices;
   }
   }
```

---

3) **Single Trigger on Each SObject:-**
   To do this U can use run time context Variables, ASYNC flag to check first then process

---

Error Mainly handle:

1) System.assertEquals();
2) Or can use addError to display based on certain condition
   E.g:

```
trigger InvoiceGenerattor2 on Customer__c (before insert, before update) {
   if(Trigger.isBefore){
      if(Trigger.isInsert){
         //System.assertEquals('expected', 'expected');
         for( Customer__c c: trigger.new){
               c.addError('Not in mood what are u doing man');
      }
      }
   }
```

BULK UPLOAD: (AND IN CHILD Relation Update Any field based on parent field set) with best approach

Trigger:

```
trigger InvoiceGenerattor on Customer__c (after insert, after update) {
    System.debug('Trigger.new' + Trigger.new );
    System.debug('Trigger.oldMap' + Trigger.oldMap);
    NewClass.create(Trigger.new, Trigger.oldMap);
    System.debug('Trigger.newMap' + Trigger.newMap);
    NewClass.updateInvoice(Trigger.new, Trigger.newMap);
}
```

CLASS:

```
public class NewClass {
    public static void create(List<Customer__c> li, Map<Id, Customer__c> mi){
        List<Invoice__c> listOfInvoices = new List<Invoice__c>();
        for(Customer__c cus : li){
            if(Trigger.oldMap != null ){
              if(cus.Customer_Status__c =='Active' && mi.get(cus.Id).Customer_Status__c != 'Active'){
                        Invoice__c IC  = new Invoice__c ();
                                IC.Customer__c = cus.Id;
                                IC.Amount_Paid__c = 200;
                                IC.Description__c ='This is good to go, we are awsm';
                                listOfInvoices.add(IC);
              }
            }else{
                    if(cus.Customer_Status__c =='Active'){
                                Invoice__c IC  = new Invoice__c ();
                                        IC.Customer__c = cus.Id;
                                        IC.Amount_Paid__c = 200;
                                        IC.Description__c ='This is good to go, we are awsm';
                                        listOfInvoices.add(IC);
              }

            }
    }
         insert listOfInvoices;
    }
    public static void updateInvoice(List<Customer__c> li, Map<Id, Customer__c> newmi){
        List<Customer__c> customerListwithInvoice = [select id, Customer_Status__c, (select id, Description__c from
Invoices__r) from Customer__c where Id In: newmi.keySet()];
        System.debug('customerListwithInvoice' + customerListwithInvoice);
        List<Invoice__c> listOfInvoices = new List<Invoice__c>();
        List<Invoice__c> newList = new List<Invoice__c>();
        for(Customer__c li2 : customerListwithInvoice){
            newList.add(li2.Invoices__r);
        }
        System.debug('newList'+newList);
        for(Invoice__c IC: newList){
            IC.Description__c ='i am updated baby';
            listOfInvoices.add(IC);
        }
        update listOfInvoices;
    }
```

More Updated version of code, because it will break if already invoices are there.

```
   public static void updateInvoice(List<Customer__c> li, Map<Id, Customer__c> newmi){
      List<Customer__c> customerListwithInvoice = [select Id, (select id, Description__c from Invoices__r) from
Customer__c where Id In: newmi.keySet()];
      System.debug('This is new');
      System.debug('customerListwithInvoice' + customerListwithInvoice);

      List<Invoice__c> listOfInvoices = new List<Invoice__c>();
      List<Invoice__c> newList = new List<Invoice__c>();

      if(customerListwithInvoice.size()>0){
      for(Customer__c li2 : customerListwithInvoice){
         for(Invoice__c IC: li2.Invoices__r){
         newList.add(IC);
         }
      }
      System.debug('newList'+newList);

     for(Invoice__c IC: newList){
         IC.Description__c ='i am updated baby';
         listOfInvoices.add(IC);
      }
        update listOfInvoices;
      }
   }
}
```

TestClass:

it is also a apex class, this is for to check whether class is giving expected output or not

Unit test methods take no arguments, commit no data to the database, and send no emails.

Class is defined as @isTest

Method is defined as testMethod, @isTest,

static and void because just need to check assertEqual

System.assertEqual is used to check expected and actual result

It will throw an error if we dont get value that is expected

Won't cause any issue or error if we get value that is expected.

Example:

```
@isTest
Public class class_name{
        Public @isTest/ testMethod static void Method_name( parameter){
                //Setup the data
                // system.assertEqual();
        }
}
```

Example2:

Test Class:

```
@isTest
public class FirstTestClass {
   @isTest
   public static void main_test(){
      Integer a = NewClass.Test(5);
      System.assertEquals(5, a);

   }
}
```

Main Class:

```
public class NewClass {
   public static Integer Test(Integer a ){
      return a;
  }
}
```

To execute:

Classic: Run Test Class

Developer Console: Test → Run Test (And particular method is also there to run and test)

Deployment: Select class which needs to run.

@IsTest(OnInstall=true): Will test class wll gets execute upon **installing a package**

Coverage:

Classic: use extension : Code Coverage

Developer Console :(Show all classes) ---> (Pick one class)--> (Select Code Coverage)

How to Calculate Test coverage percentage?

(CoveredLine/ Total line) *100

Test Data Setup:

To Test with some production data or env data:

@IsTest(SeeAllData=true) : method or class will have access to all org data

1) **To Test with Manually Created Data (All transaction will be roll back so no need to worry)**
   Object: Ultimately operation/ class gonna perform certain Data operation
   So if your data is what we are expecting in the class we are ok.
   E.g:

```
@isTest
Public class Test{
        @isTest
        public static void main_test3(){
                Customer__c c1 = new Customer__C();
                c1.Name = 'My name is anthony';
                c1.Customer_Status__c = 'Active';
                insert c1;
        }
 }
```

Not good for all Methods, use one testSetup method in a class.

2) **@testSetup**: create Test data**(All changes will roll back) (use Annotation only)**

E.g:

```
@isTest
Public class Test{
        @testSetup static void testDataSetupMethod(){

        }
}
```

Not good for all module classes, use one separate class and follow the naming con.

3) **UtilClass or DataFactory class**: common data that is used in different-2 classes.
HWS_Constants: for a single variable.

E.g: with util class

```
@isTest
Public class Test{
        @testSetup static void testDataSetupMethod(){
            Id customerAccountType = Hws_Utility.getRecordTypeByName('Account','HWS_constants.customer');
        }
}

Public class HWS_Utility{
        Public static Id getRecordTypeByName(String ObjectName, String RecordTypeName){
        Return id  = schema.getGlobalDescribe().get(objectName).getDescribe().
                    getRecordTypeInfosByDeveloperName().get(recordTypeName).getRecordTypeId();
        }
}

Public class HWS_constant{
        Public static final Customer ='customer';
}
```

E.g2: with DataFactory class

```
@isTest
Public class Test{
        @testSetup static void testDataSetupMethod(){
           Account ac =  hws_testData.createAccount();
           ac.recordTypeId = '';
           Insert ac;
        }
}


Public class hws_testData{
        Public static account createAccount(){
                Account acc = new Account();
                acc.name = 'accoountName';
                acc.recordTypeId = HWS_Utilit.getRecordId('Account' , HWS_Constant.Legal_Entity);
                Return acc;

        }
}
```

Q) testsetup method call automatically or what?

Ans: it gets called by the salesforce processing engine by default in the Test class where it is defined, before doing any other test method calls.

E.g:-

```
@isTest
public class FirstTestClass {
    @isTest
    public static void main_test(){
        Integer a = NewClass.Test(5);
        System.assertEquals(5, a);

    }
    @isTest
    public static void main_test2(){
        Customer__c c1 = new Customer__C();
        c1.Name = 'My name is anthony';
        insert c1;
        System.Test.startTest();
        c1.Customer_Status__c = 'Active';
        update c1;
        System.Test.stopTest();
        System.debug('customer id' + c1.Id);
        List<Invoice__c> li = [select id from Invoice__c where Customer__c =: c1.Id];
        System.debug('list ' + li);
        System.assertEquals(1,li.size());
    }
    @isTest
    public static void main_test3(){
        Customer__c c1 = new Customer__C();
        c1.Name = 'My name is anthony';
        c1.Customer_Status__c = 'Active';
        insert c1;
    }
}
```

Note:

Test Class ---> Direct Class Method Calling, Fine
Class to Other Class Method Calling and that class, Test Class calling, Fine
Test Class → DataBase Commiting and Trigger triggering and then Method calling

How to find it?

Search in Files **Class Name** and if needed Particular Method Name

Note:

Every trigger must have some test coverage.

All classes and triggers must compile successfully.

@testVisible: private method needs to call out

Test.startTest and **Test.stopTest**:::----

From Where Code Needs to Execute and where it needs to stop. (Governor Limit)

Asynchronous apex should gets completed before Test.stopTest();

System.runAs() must be used only in a test method

Test.isRunningTest(): Disabling trigger from calling to avoid Soql query or job test before execution

@IsTest(isParallel=true): **Test classes can run parallely**.

Note:

- When deploying Apex to a production organization, each unit test in your organization namespace is executed by default.

Keynote:

Make Sure Every unit is covered including positive and negative cases,

As well as bulk and single records.

Tool:-

Developer console coverage: Tip: to get to know which is covered which is not
Show overall coverage for each classes and u can click and go to particular classes Coverage also

Note: we cannot see, which class method test coverage in which test class

So for that we need to **search with the method name in files** of that class & find out test class.

(We can also check dependent class and from dependent class method in test class)

Apex Class Detail page coverage / : Tip to figure it out error, coverage of particular class
Code Coverage extension// : Same as dev console(show coverage in classic)

Synchronous Apex and Asynchronous Apex:-

Synchronous Apex: Don't wait for resources just execute at once.

Synchronous term means existing or occurring at the same time. Synchronous Apex means entire Apex code is executed in one single go.

1) Class, Test Class, Trigger

Asynchronous Apex: Will execute When resources are available to execute.

1) **Future**
2) **Schedulable**
3) **Queueable**
4) **Batch**

Related Points to Understand:-

sObject: both standard and custom classes object can be define with sObject

Get all record:-

```
Map<String, Schema.SObjectType> m  = Schema.getGlobalDescribe() ;
Schema.SObjectType s = m.get('API_Name_Of_SObject') ;
Schema.DescribeSObjectResult r = s.getDescribe() ;
Map<String,Schema.SObjectField> fields = r.fields.getMap() ;
```

Id devRecordTypeId= Schema.SobjectType.Contact.getRecordTyeInfosByDeveloperName().get('Customer').getRecordTypeId();

Getting PickList value and data:

Using Dynamic apex, we can achieve this.on object of type pickilist, call **getDescribe()**. then call the **getPicklistValues()** method. iterate over result and create a list. bind it to <apex:selectOptions>.

Custom setting data accessing:-

```
SO_Country__c code = ISO_Country__c.getInstance('˜INDIA');
//To return a map of data sets defined for the custom object (all records in the custom object),
//you would use:
Map<String,ISO_Country__c> mapCodes = ISO_Country__c.getAll();
// display the ISO code for India
System.debug('˜ISO Code: '˜+mapCodes.get('˜INDIA').ISO_Code__c);
//Alternatively you can return the map as a list:
List<String> listCodes = ISO_Country__c.getAll().values();
```

Async-Future:

Purpose:

Make a callout to external web services:::-Http callout Class

Avoid MIXED_DML_OPERATION exception:- Working on two standard user ex: Account and

Void, Static, Primitive data type Parameter. And @future Annotation is used

Example:

```
Public class class_name{
        @future
        Public static void method_name(){
        }
}
```

@future(callout = true); :: http call out for webservice

ASYNC-Schedulable:

Schedule a class when to execute

For that need to implement a interface

```
Public class class_name implements schedulable{
        Public void execute(SchedulableContext SC){


}
}
```

Execute a schedulable class:

```
String cronExp = “20 30 10 0 2”;
Class_name co = new Class_name();
Id JobId = System.schedulable(cronExp, co);
```

Tracking:

Schedule Job in Quick Find

BatchClass:

To process large amounts of data, batch ASYNC jobs need to be implemented.
Minimum of Batch record size 1, Default is 200 and Max is 2000

Database.Batchable interface needs to be implemented.

E.g:

global class FristBathc implements Database.Batchable<Sobject> {}

It has the following three methods that need to be implemented −

- **Start : To define which sets of records are we gonna proceed**
- **Execute: To define what need to proceed on those set of records**
- **Finish: To define what need to done after processing(post activity)**

Start:

```
global Database.Querylocator start (Database.BatchableContext BC){}
Simple query to process the data

Global iterable start (Database.BatchableContext BC){} :
Complex query logic
```

Execute:

```
global void execute(Database.BatchableContext BC, list<sobject>) {}
```

Finish:

```
global void finish(Database.BatchableContext BC) {}
```

Calling or executing:

```
Class_name co = new Class_name();
Id batchId = Database.executeBatch(co, 200);
```

Tracking of batch Job:

1) AsyncApexJob job = [SELECT Id, Status, JobItemsProcessed, TotalJobItems, NumberOfErrors FROM AsyncApexJob WHERE ID = :batchId];
   System.debug('Job' + job);
2) Apex Job in setup

E.g:

```
FristBathc co = new FristBathc();
Id batchId = Database.executeBatch(co, 200);
AsyncApexJob job = [SELECT Id, Status, JobItemsProcessed, TotalJobItems, NumberOfErrors
FROM AsyncApexJob WHERE ID = :batchId ];
System.debug('Job' + job);
```

Example:

```
//public or global both can defined  ( but if its global then need to define global )
global class class_name implements Database.Batchable<Sobject>{

// Start Method: - what need to processed ( call - once )
global Database.Querylocator start (Database.BatchableContext BC){
        return Database.getQueryLocator(select id from account);

}
// Execute Method: How need to processed ( call - in batch defined)
global void execute(Database.BatchableContext BC, list<sobject<) {

}
// Finish : what need to be done ( email alert, notification ) ( call once)
global void finish(Database.BatchableContext BC) {


}
}
```

E.g: ---

```
global class FristBathc implements Database.Batchable<Sobject> {
// start Method
global Database.Querylocator start (Database.BatchableContext BC){
    //get list of customer record
    return Database.getQuerylocator('select id, Customer_Status__c, Customer_Description__c from Customer__c');

}
//execute method
global void execute(Database.BatchableContext BC, List<sobject> scope) {
    //if customer status is active, update customer status and customer discription
    //either you can define it as customer list or you can cast it in iteration: to case Customer c = (customer__c)scope;
    List<Customer__c> li = scope;
    List<Customer__c> lic = new List<Customer__c>();
    for(Customer__c c: li){
        if(c.Customer_Status__c == 'Active'){
            c.Customer_Status__c = 'Paid';
            c.Customer_Description__c = 'This is updated by batch';
            lic.add(c);
        }
    }
    if(lic.size()!=0){
    Database.update(lic);
    }
}
//finish method
global void finish(Database.BatchableContext BC) {
   //send an email to customer


}

}
```

Note:

1) **Along with Batch also needs to write Test classes**
2) Each Batch does not maintain the state so if u want to maintain state Then need to implement stateful interface

   Example:

   ```
   Public class class_name implements Database.Batchable<Sobject>,
   Database.Stateful{

   }
   ```

3) Scheduable and batchable classes work together

   Example:

   ```
   Global void execute(SchedulableContext SC){
       BatchDemo  BD = new BatchDemo();
       Database.executeBatch(BD);
   }
   ```

4) 5 batch @ time and 100 Schduable jobs can at max

4) Async- Queueable:

Respect order of execution because future method doesnt

```
Public class class_name implements system.queueable{
Public void execute(QueueableContext QC){

}
}
```

Execute:

```
Class_name cn = new Class_name();
system.enqueJob(cn);   // Id JobID = system.enqueJob(cn);
```

Tracking:

Apex Job in setup => quick find

SOQL QUERY: (Object Query)

Select **Column_Name** from **Table_name** where **String_matching**

Traversing the record

Aggregate Function:

SUM, AVG, COUNT, MAX, MIN

Output will be AggregateResult[] Ar

E.g:-

```
List<AggregateResult> rds= [select count(Amount_Paid__c), SUM(Amount_Paid__c),
MAX(Amount_Paid__c), MIN(Amount_Paid__c) from Invoice__c where
Customer__r.Name='Customer3'];
```

AggregateResult:{expr0=71, expr1=7100.0, expr2=100, expr3=100})

```
System.debug('ar' + rds[0].get('expr0'));
```

Note: why are we defining 0 here because aggregateResult is also a List

Dynamic Assignment:

By using colon

Dynamic SOQL (At run time use this for better result)

Database.query();

Database.getQueryLocator();

E.g:

```
String str= 'select id from Account';
System.debug(Database.query(str));
```

E.g:

```
return Database.getQuerylocator('select id, Customer_Status__c,
Customer_Description__c from Customer__c');
```

Multiple or relation Queries:

1) Fetching parents records

2) Fetching child records

Parent to child : ---> block in table, for this use () Operator in parent query to search

Child: Table with multiple entries or (. operator) (in master- detail) detail object

E.g:

Parent to Child:

```
select id, (select id, Name from Invoices__r ) from Customer__c
Select id, (select id from Contacts) from Account
```

Child to parent:

```
select id, Account.Name from Contact
select Customer__r.Name from Invoice__c
```

Note:
we need to specify relationship elements also that we are querying
Object will come **__c**
but relationship will come **__r** and u can query parent record also
Plural s keep in mind while querying inside parent to child if standard
---
Group BY HAVING (summarizing the row data together, Having for filtering the row data)
In a field column rolling up the data using aggregate function)
Order By ASC, DESC
LIMIT How many record need to display

E.g:
Popular to find duplicate record:
Select count(id) from account group by account_name having count(id)>1
Associate Contact related to account:
select count(id) from Contact where Account.Name ='Burlington Textiles Corp of America'

WIldCard :
Operator:
=, != , > , <, >= , <=
And:
To Join multiple where list together
E.g: where Name='shubham' and billing_city ='canada';

Like Supported to regular expression
%: one more
_: only one
\: only for special character **(exactly matches special character)**
E.g: Where Name Like 'Test%'
E.g: Name Like 'Tier%'
Name Like 'Tier\%'

IN:
E.g: where Billing_state in ('NY','CF')
E.g:-
```
List<String> li = new List<String>();
li.add('1');
li.add('2');
List <Customer__c> li1 = [select id from Customer__c where Customer_Description__c in :li];
System.debug(li1);
({},{})
(Customer__c:{Id=a012y000009JzcKAAS}, Customer__c:{Id=a012y000009JzcLAAS})
```

NOT IN:

E.g: where Billing_state Not IN ('NY','CF')

INCLUDES:

E,g: where subject includes('computer;english','hindi')

E.g: where subject ='computer;english'

Where subject includes ('computer;english','math'); where computer and english or math

EXCLUDES:

E.g: where subject excludes('computer;english','english');

Some standard function can be also used

E.g: Today(), Calendar_Year(createdDate)

SOSL: (Object Search)

FIND ' ' IN **Columns(ALL FIELDS)** RETURNING Field_searching

***ABC**

SOQL VS SOSL:

1) **OQ: Object Query (Query in Object to find the record)**
2) **OS: Object Search(Multiple Records Column of Objects)**
   Particular String in Multiple Objects (Nested Query also possible 20 op)
   When we know string/Data present in Object USE SOQL else Use SOSL
   Most efficiently used in SOAP and REST
   SOQL: select from where (Wildcard: like, as , in)
   SOSL: FInd in returning (wildcard: with)

DML Operation:(Data Manipulation Language)

1) **Insert (for both standard and custom object)**
2) **Update**
3) **Upsert**
4) **Delete**
5) **Undelete (ALL ROWS) must define inorder to perform**
   (It also look in recycle bin as well if we define ALL ROWS)

DataBaseMethod:(Same as DML but more flexible to perform action)
More Flexible:

1) **Partial Updation is allowed.**
2) **Save and failed record we can get separately to proceed.**

Database.insert(List, false);
False: partial updation allowed
True: Partial updation is not allowed (By default it is true only)

```
E.g:- Database.SaveResult[] results = Database.insert(accounts);
```

Database.Update(List, false);

```
E.g:-Database.SaveResult[] srList = Database.update(updatedInvoiceList, false);
// Iterate through each returned result by the method
for (Database.SaveResult sr : srList) {
   if (sr.isSuccess()) {
      // This condition will be executed for successful records and will fetch
      // the ids of successful records
      System.debug('Successfully updated Invoice. Invoice ID is : ' + sr.getId());
   } else {
      // This condition will be executed for failed records
      for(Database.Error objErr : sr.getErrors()) {
         System.debug('The following error has occurred.');

         // Printing error message in Debug log
         System.debug(objErr.getStatusCode() + ': ' + objErr.getMessage());
         System.debug('Invoice oject field which are affected by the error:'
            + objErr.getFields());
      }
   }
}
```

Database.insert()
Database.update()
Database.saveResult[];
Database.savePoint():- SavePoint if any failure occur
Database.rollback():- Roll Back to previous state
Database.delete();
Database.deleteResult

Database.merge();
Database.mergeResult

Note:
1) Field permission of read should be there in order to see Object Queried data.

2) You can also define Alias Name if the Object name is getting too lengthy.
E.g: from contact c where

3) Return of SOQL Query always be a List

List<Invoice__c> li = [select id from Invoice__c]

4) For Multi select picklist we can define Multiple value
includes('computer;english')
; And , or

Related list will automatically added to master object
(Game of Record ID, null, wildCards in SOQL)

How to Prevent SOQL injection:

1) **Always use static variable (them trim and apply validation)**
2) **Always use define set of list (to check you are getting expected value)**

Cross Site Request forgery:

To provide a seamless experience to user google track or server provide a session. During that active session if buy mistake you visited some hacker website and click on something which redirects you to banking website.
Now your session is active and through that session your information is shared to attacker

(can be prevented by token but fail if state changing operation are there)

Tracking/ Debugging/ workaround:-

Tracking:-

1) Data:
    a) Extension : SalesforceSimplifier
    b) Salesforce Track: Set History Tracking (fields)
2) Organisation Wise: (field and Action wise)
    a) View Setup Audit trail : To see Fields related changes
3) File and code related in metadata:
    a) TortoiseGit---> show Log
4) Bulk data job:
    a) Bulk data load job / Apex job

Debugging:

Wrong in data input supply (Expecting something but System getting something else to process)
What action gets performed when i do certain things @ Run time:
Null Value/ Wrong Value(Hack) / Wrong CalCulation(/0), out of bound (recursive, out limit)
Dependent value deleted/ or not pass value or anything which is not expected.

Process of Debugging: (Top 2 Bottom)

1) **Check in Lower Env(Mismatching of code or what) (Multiple profile, Muser, Menv)**
   Declarative and Programmatic Approaches of Comparing
   E.g: Profil, permission, sharing setting or code difference
2) **Test Class**

3) **Debug Log** **CheckPoint Debugger** **Search in File** (cmp inside cmp, page, community page)

//Discuss with Team

Debug Log:

1) Developer Log
2) System Debug

Both are the same only but in Developer log sometimes we misses bunch of log so it for that we can use debug log.

For an Example:

Front-End(HTML, JavaScript)

Google Dev tool:-

Element, Console, Source
Network, Performance, Memory
Application, Security, Audit
HTML: attribute show
Java Script: console.log

HandFull:

1) Enable Debug Mode
2) Disable Cache performance
3) Salesforce Lightning Inspector extension

BackEnd(Apex Related)

Developer Console

You can set to see the error for particular users.

What you wanna track(define **Debug log level**) according to that , then perform action, and track @System level.

Apex: system.debug

Application: System Debug log

Process Builder/ Automation : Save/ Activation / Debug ⇒ Get an error (Debug Log)

ScreenFlow: Debug

ChutPut for a User: Debug Mode for that User

API: Simulation (check Point)

Workaround:

1) Take backup
2) Pull the code **(Pull the file)**
3) Do your changes **(Add your changes)**
4) **Tested it** (Testing of code)
5) if it work, keep it and **(Push the changes)**
6) If it dont work then Undo your changes but keep a **note of diff** so it **won't overwrite (UNDO)**

Duplicate Cod Removal/ DataCleanUp:-

Tools: SonarQube:

Continuous Code Inspection for Quality

1) **Code Reliability**
2) **Application Security**
3) **Technical Debt**

Guidance:

1) In Which class you need to do updation
2) What you need to do
3) Where you need to do

Classic Example:

1) Code can not be empty : Put comment
2) Section of code can not be commentated out : remove comment of code
3) UnUsed Variable: if it is unused (only in declaration) remove it, else put system.assert
4) SeeAllTrue: in webauthen. Needed else u can use test data factory
5) Nested if block/ function operation more than 15 and etc.

LIGHTNING COMPONENT

---

1) AURA COMPONENT
2) LIGHTNING WEB COMPONENTS (LWC)

BASIC OF JAVASCRIPT

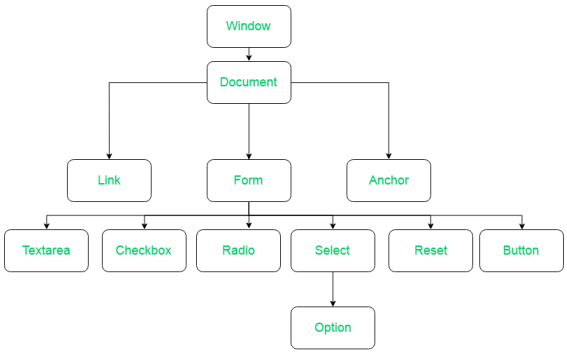
Model View Controller (MVC):
Document Object Model (DOM) // document tree structure

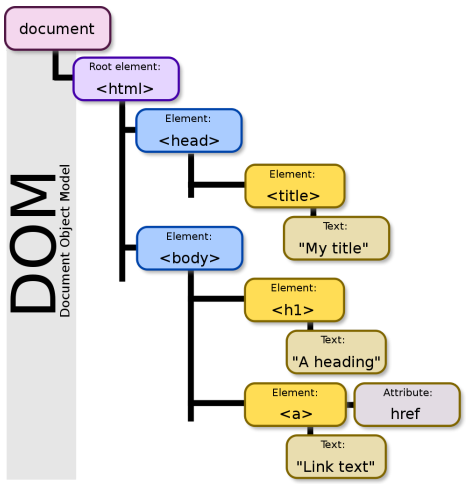
HTML: Add Structure to WebPage
CSS : Add Style to the Structure and on webpage
JavaScript: Add Functionality in Structure and on web action

Question is : How does the HTML structured tag interact with javascript??
Answer: Because of DOM Structured.

HTML Structure---> DOM Structured ---> Interpreted By JavaScript(document.getElementBy)
Note: First browser render javascript then HTML tag/ Structure then css

Class → instance → Object → Attribute, Method / Action access.
Browser → Ram (memory) → block in hash code → store the information



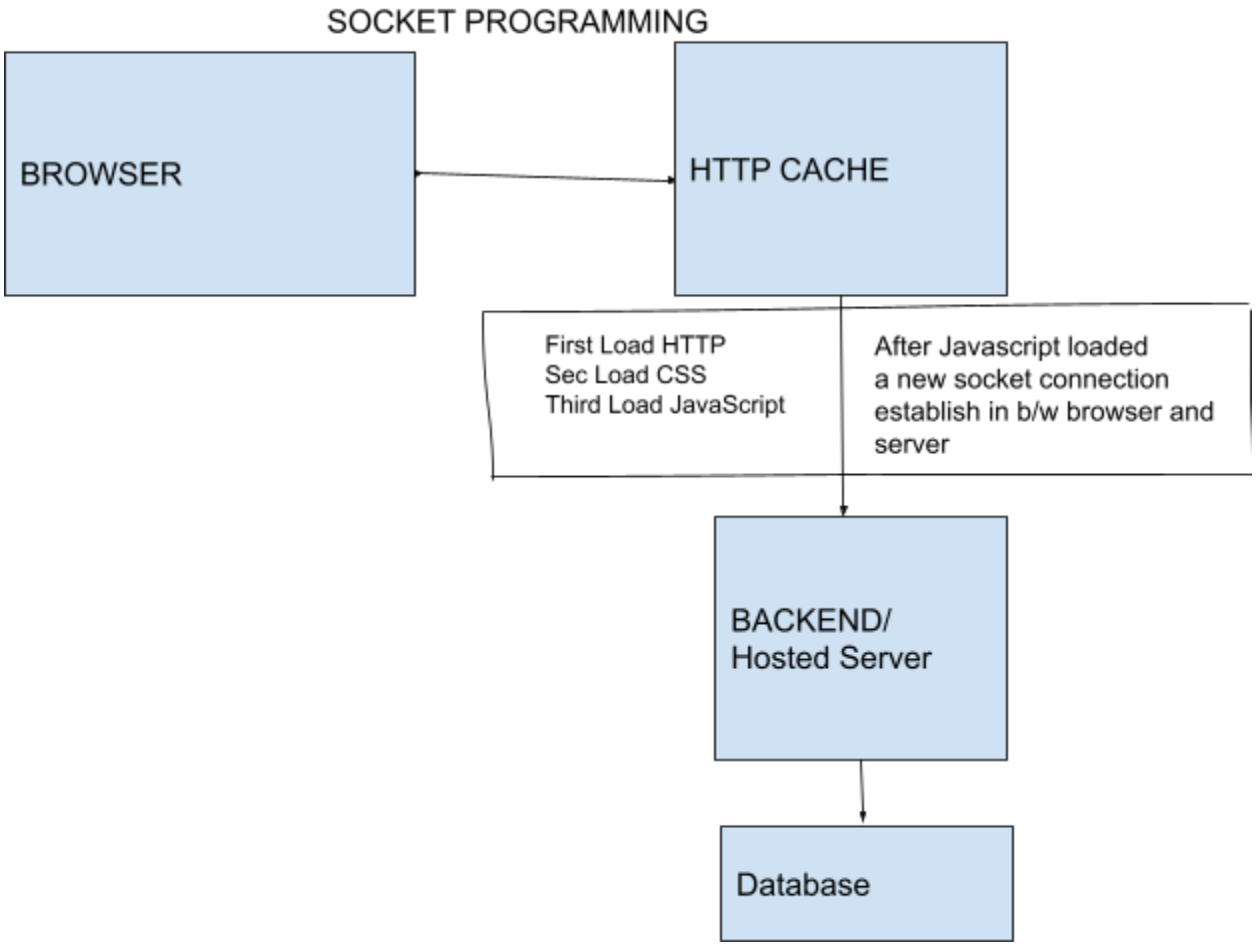


JavaScript Basic:

JavaScript is a client / server side scripting language.

Scripting language is used to make interaction b/w browser and server.

Something that needs to be interpreted by the browser.



The newer version of java script is: ES6, ES2015:

Start:-

To tell browser that where is javaScript Code for that:

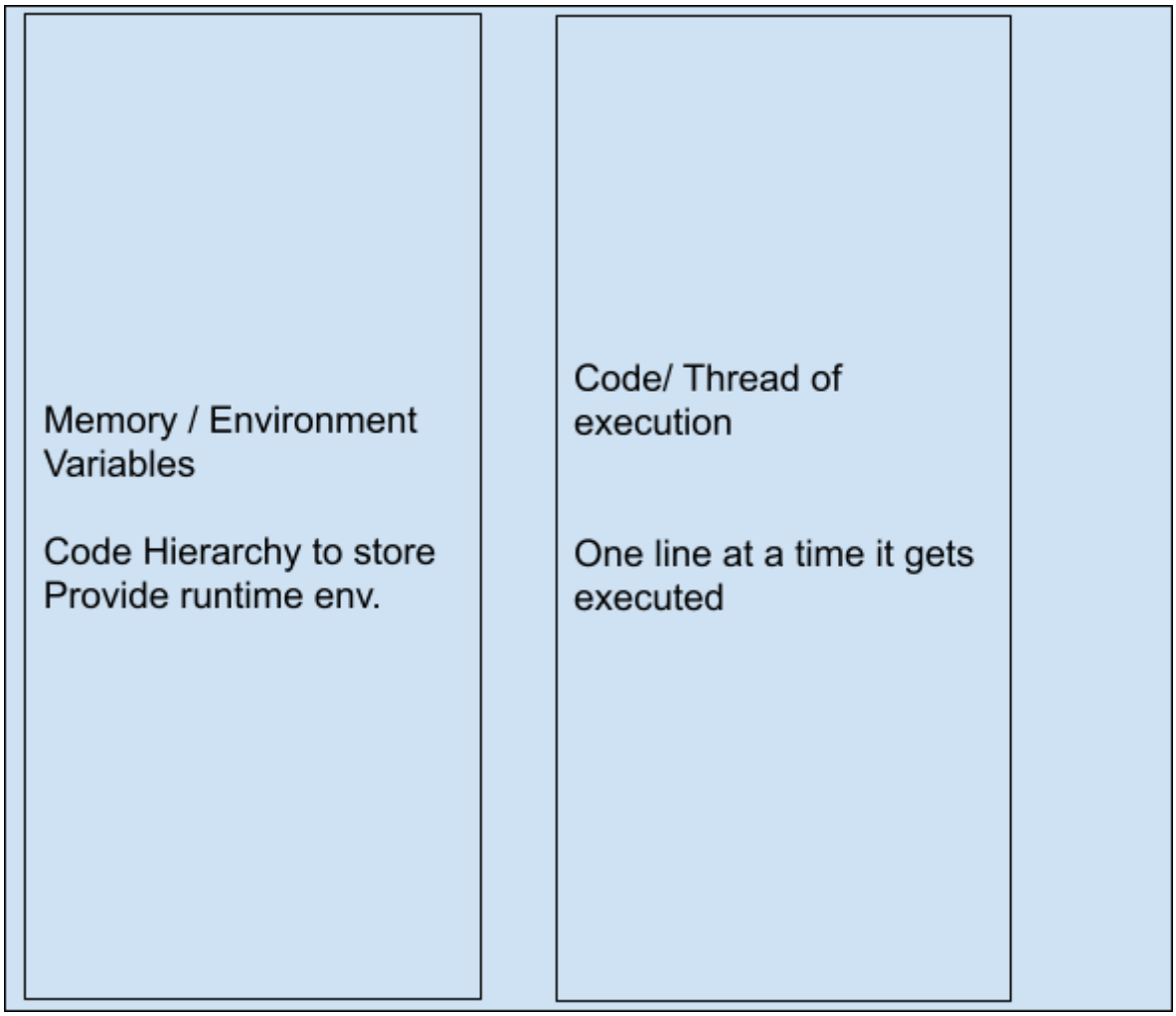
Include Javascript

2 ways:

1) **<Script type="text/javaScript" src=""> why type: because 15 type of scripts out there. </Script>**
2) **<link rel="" type="" href=""/>**

Execution of javaScript

Execution of javascript happens in Execution context/ Container



Variables Declaration: **in apex we define by using type name value but javascript loosely type**

In Older version, variable declare:

Var : use when u need to define variables globally...

E.g: Var x = 3; without var keyword also u can declare the variable like x =3; but remains global

There were some issues of strict mode, to overcome those issues new versions were launched.

In newer versions. Variables declare globally and locally:

use when you are using variables inside a block.

Let : when u want values gets changed as per business logic

Const when the value of variables does not need to change.

Basically for resolving this and globally declaring variables issues.

Dynamic value assignment:
in apex we don't directly deal with the user so not needed
${Variable_name}
{c.component_name} {v.variable_name}
Var s = prompt(“”,””);
document.write(‘hi ${value}’});

For HTML we need to use ! for dynamic value reference.
PRINT: in apex we use System.debug(‘x’);
In javascript :::: document.write(‘x’);

Note: HTML tags are also supported in javascript : it should be inside Quotation : “”
E.g: document.write("LESSON 1: PRINT HELLO WORLD USING VARIABLES

");

For Specific html attribute : `document.getElementById("demo").innerHTML = “”;`
HTML DOM Contains each property of tag
Means, type, value, name, label, placeholder, description, required, unique, event/action, id
e.g:
document.getElementById(“”).value
document.getElementById(“”).etc….

Log:-
Console.log(x);

debugging:-
Debugger

GET PARTICULAR ELEMENT ID in apex not needed

document.getElement
document.getElementById(“IdName”); : **particular member // individual item**
document.getElementByClassName(“”); **: particular class // group of diff tag**
document.getElementByTagName(“”); **: particular package// group in same tag**

Component.find

Note: Class & Tag Name return array of item and ID returns individual item and these are in key value pair structure so to get value:
E.g: Var x = document.getElementByID(“”);
Var name = document.getElementById(“name”).value;

Var para = document.getElementByTagName(“p”); //p is the tag // returns array of paragraph
P[0].style.font = 24;

Comment: **Same as apex**

Single Line: //

Paragraph : /* */

Data Type:: Type, Name, Value:-

Primitive Data type:

Note:

== : only checks the value

=== : values with data type

```
if (5==5){
        document.write('true first functon');
}
if(5 == '5'){
        document.write('true second functon');
}
if(5==='5'){
        document.write('true third functon');
}
Else{
        document.write('true else functon');
}
```

Logical Operator: logical operator are same as apex &&, ||, ! etc.

Associativity and Precedence:

Arithmetic Associativity: It starts from left to right

Assignment Associativity: it starts from right to left

```
var sum1= 2 +3 + "good " + 3+4;
document.write(sum1);
```

Output is : 5good 34

Precedence: BODMAS

Concatenation: **Same as apex**

```
var sum =2+3;
var sum2 = 'hello' +'paaji';
document.write(sum2);
document.write(sum);
```

Non Primitive Data Type

1) **Set 2) Map 3) List**

```
var a =['1','2','3'];
document.write(a);
```

Some Popular method for arrayList:-

```
a.sort()
a.reverse()
a.push('1');
a.pop(1);
```

Join
concat

Class:

```
class shubham{
constructor(){}

dhanuka(){
document.write('this is good to go');
}

}
var sh = new shubham();
sh.dhanuka();
```

Calling of a Class By using Object Instance:

Var x = new Class_Name(); // by default constructor calling

Not need to define class name while creating object instance here.

Some Standard Class Defined :

Math Class

Math.sqrt()
Math.pow()

Date Class

Var sh = new Date();
sh,getHours(); ,sh.getMinute();, sh.getSecond();

Note:

typeOf : return the type of the variable.
InstanceOf : return true if the instance of an object type.

Condition: **Same as apex**

```
if(){
}else if(){
}else{}
```

Note: multiple condition check with && same as apex

Iteration/ loop : **Same as apex**

E.g:

```
for (var i=1; i<=10; i=i+1) {
    if (i<10) {
        document.write(", ");
    }
}

while(){}
do{}while();
```

Note: break and default in case of nested

Switch Statement: **both in apex and javascript it is different**

In apex:

```
Switch on expression{
    When case1{
    }
    When else{
    }
    When default{

    }
}
```

In javascript:

```
Switch (){
    Case 1:
    Case 2:
    Default:

}
```

Note: break and default needs to define in when/ case condition

Function (3 ways) : in apex we define function in a single way that is type name (){}

Outside of the class:

```
Function function_name(){
}
```

Calling of a function:

```
function_name();
```

Parameterized function:

Return statement:

E.g:

```
        Function function_name( x,  y){
                Return x+y;
        }

        Var z = function_name(2,3);
```

Note: in parameterized function we don't need to define var variables

Note2: in class we don't need to define keyword function just name and parenthesis

```
class shubham{
        constructor(){}
        dhanuka(){
                document.write('this is good to go');
        }
}
var sh = new shubham();
sh.dhanuka();
```

Anonymous Function:

```
Function (name){
        Return name.toUpperCase();
}
```

Declaration of Anonymous function in new way:- (ARROW FUNCTION)

```
Name => name.toUpperCase();
```

Note in javascript Method you can pass as a argument in another method

Function shubham(function(x){

}}

Advance version of declaring function:

```
const arr = [1, 2, 3].map(function(x) {
            return x ** 2;
})
const arr = [1, 2, 3].map(x => x ** 2);
```

```
sh = name => {
            console.log({name});
            document.write('this is testing');
            document.write('this is working');
}
sh('dhanuka is great boy');

const sh = (name)  => {
            console.log({name});
            document.write('this is testing');
            document.write('this is working');
}
sh('chalene do');
```

Different index / not this :-

```
const obj = {
  foo: function() {
    console.log('foo')
  }
}
obj.foo();

const obj2 = {
  foo: () => {
    console.log('foo')
  }
}
obj2.foo();
```

Same Index / this Operator:-

```
const obj3 = {
  hi : 'heelo',
  foo: () => {
    console.log('foo')
  }
}
console.log(obj3.hi);
console.log(obj3.foo());
```

Basically :-

Outside of the class:

function keyword not needed & function_name also defined in different way :const x = () => {}

`Function function_name(){} is replaced by () =>{}`

Function function_name (x){} is replaced by (x) => {}

Two different purpose: Normal/ JSON format:

```
const sh1= (name)=>{document.write("Here we go");}
document.write(sh1());


const sh = {
myname: (name)=>{document.write("Here we go");}
}

sh.myname();
```

Inside a class: function keyword not needed : function_name(){}

```
class shubham{
        constructor(){}
        dhanuka(){
                document.write("hello");
        }
}
var sh1 =  new shubham();
sh1.dhanuka();
```

Javascript Item:: using jquery library ::

Range/ Slider
Horizontal Bar/ Vertical bar
Pagination

Effect / Animation
FadeIN/ FadeOut
ZoomIN/ ZoomOut

Date Picker Widget
Timer

MessageBox
AutoComplete
Draggable
Sortable
Resize

Note:

1) **in HTML use ! but not in JavaScript**

2) **Type → HTML , Object → Apex, Sobject,**
Var, let const::Dynamic Approaches:::no need of data type
Function

3) **Destructive Changes:**

Const A =[1,2,3];
Const x = A[0];
Const y = A[1];
Const z = A[2];
Instead of this use this way same:-
Const[x,y,z] = A;

Const JSON_Format = {'type':'shubham','category':'dhanuka','surname':'dhanuka'};
Const type = JSON_Format.type;
Const category = JSON_Format.category;
Const surname = JSON_Format.surname;
Const {type ,category , surname } = JSON_Format ;

4) APEX: initialize value is NUll and in javaScript initial value is Undefined.

5) JavaScript is Single Threaded, Synchronous Programming language.
Means, Specific order and one command/ process at a time.

6) JavaScript is a loosely/ Dynamic type, case sensitive language.

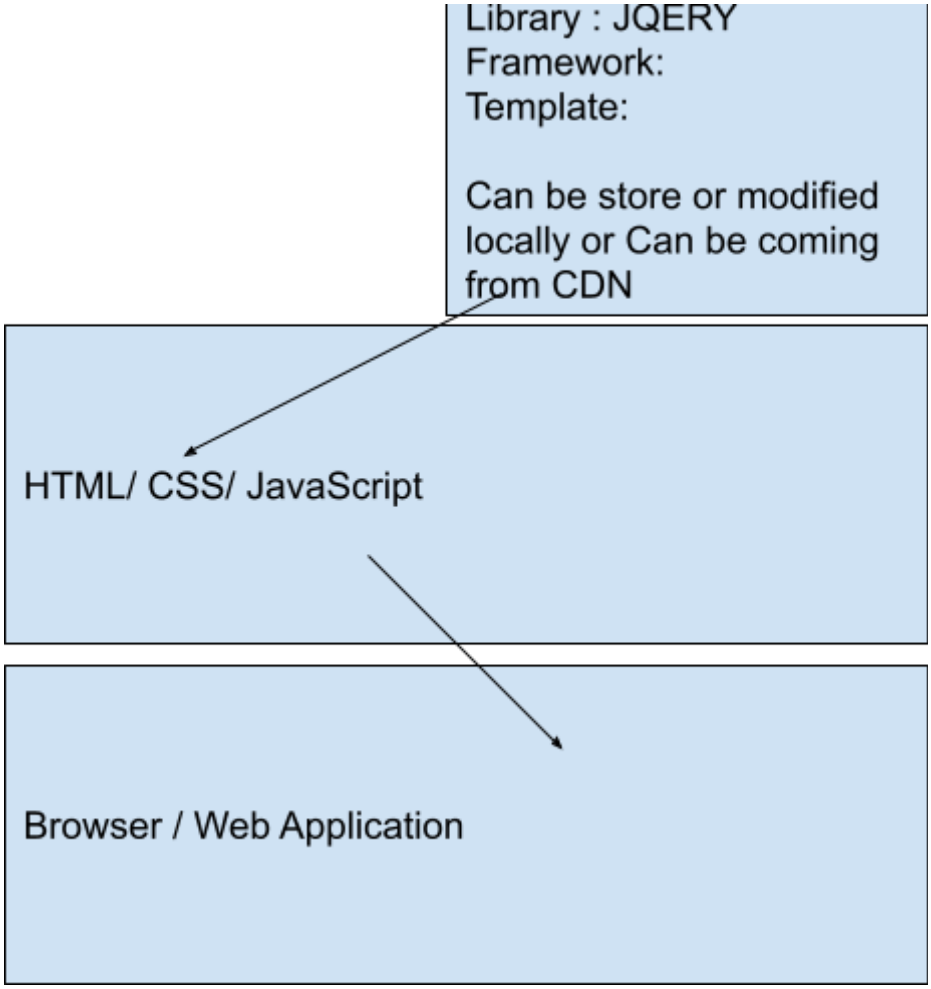
7) Code Editor Setup : visual studio code: Theme/ AutoSave/ Live Server /

8) in anchor/ redirect tag: # represent it own url

9) Window Class:
All these method comes under is Window object:
Like window.setTimeOut(); without window also it works.

setTimeOut() ::: action will take place after a particular timeout
clearTimeOut()
SetInterval() ::: action will be on repetitive mode after that particular timeout
clearInterval()

10)



HTML / CSS/ Javascript are used to create web applications.
To support and to do the development fast/ (instead of writing same generic piece of code again and again) we use library/ framework / template

Library : is specific to one thing, provide enhancement in rich e.g: JQUERY
Framework: group of specific item E.g: Angular
Template: complete combo pack to start E.g: on sale available item

---

End of Javascript Learning

Visual force Page

Tag based markup language.
Which helps developers to overwrite any existing salesforce standard page to own custom build page or to create their own custom visual force page.

The Visualforce page is MVC (model view controller) . Means controllers can be built separately and we can view it on the visual force page.

Model: data model
view : UI
controller : Business logic

Visual force can be integrated with html, css, ajax, and can be useful with available lib like jquery etc.

How to create Visual force page:

1) U can create in developer console
2) U can enable development mode and play in live server plugin
3) U can create in standard way ::: setup -> visual force page

E.g:-

```
<apex:page >   <h1>Congratulations</h1>   </apex:page>
```

Tag Category:

1) Page tag ::: browser web Page / section for particular item
2) Form tag ::: entry/ information to capture
3) Input tag ::: to take input from user
4) Action tag ::: JavaScript

5) Select tag ::: List for iteration / picklist value shown or to display
6) Output tag ::: to show or display
7) Style tag ::: CSS

PageTag: Example:-

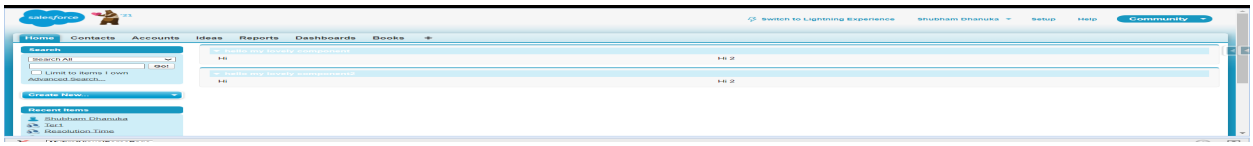
```
<apex:page>
<apex:pageBlock>
<apex:pageBlockTable>
<apex:pageBlockSection>
<apex:pageBlockSectionItem>
<apex:pageBlockButtons>
<apex:pageMessage>
```

Initializer: :::
<apex:page>
</apex:page>

First tag to create a visual force page like < html></html>
All tag goes inside of it.

E.g:
```
<apex:page >
<apex:pageBlock >
  <apex:pageBlockSection title="hello my lovely component">
  <apex:pageBlockSectionItem > Hi </apex:pageBlockSectionItem>
  <apex:pageBlockSectionItem > Hi 2</apex:pageBlockSectionItem>
  </apex:pageBlockSection>
</apex:pageBlock>
</apex:page>
```

E.g:2
```
<apex:page >
  <apex:pageBlock >
  <apex:pageBlockSection title="hello my lovely component">
  <apex:pageBlockSectionItem > Hi </apex:pageBlockSectionItem>
  <apex:pageBlockSectionItem > Hi 2</apex:pageBlockSectionItem>
  </apex:pageBlockSection>

  </apex:pageBlock>
   <apex:pageBlock >
  <apex:pageBlockSection title="hello my lovely component2">
  <apex:pageBlockSectionItem > Hi </apex:pageBlockSectionItem>
  <apex:pageBlockSectionItem > Hi 2</apex:pageBlockSectionItem>
  </apex:pageBlockSection>
</apex:pageBlock>
</apex:page>
```



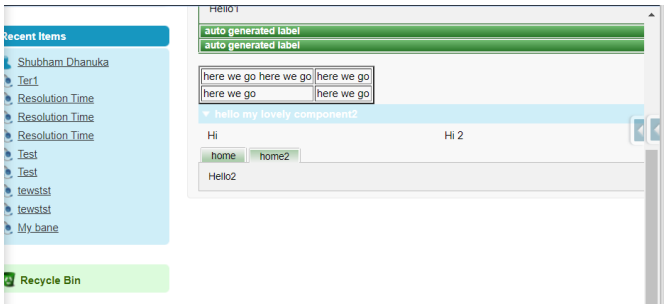
page, pageBlock, section, section item

1)Button

```
<apex:pageBlockButtons >
<apex:form >
<apex:commandButton value="save"/>
</apex:form>
</apex:pageBlockButtons>
```

2) Tab

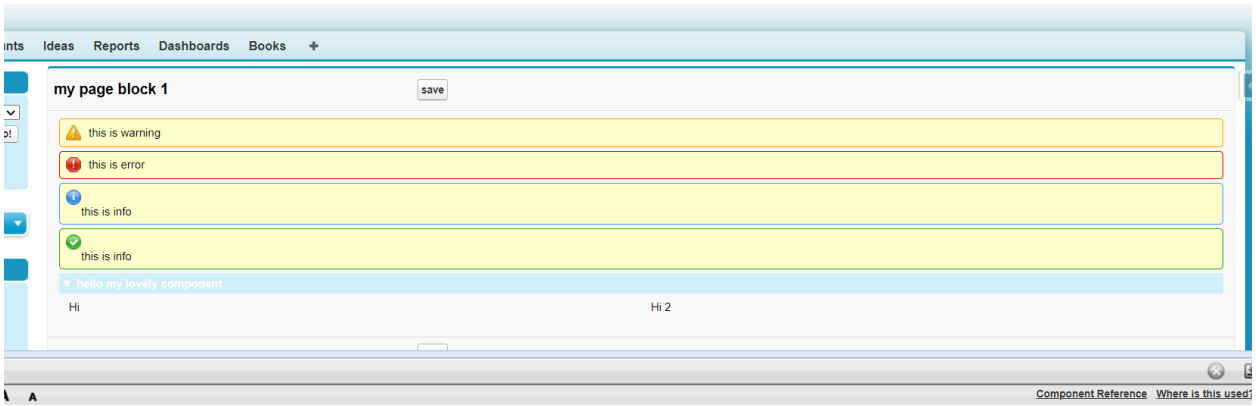
```
<apex:tabPanel>
<apex:tab>

<apex:tabPanel switchType="client">
<apex:tab label="home"> Hello1  </apex:tab>
<apex:tab label="home2"> Hello2  </apex:tab>
</apex:tabPanel>
```



3) Info/ Message Show…. Like we do in apex: by adding addError

Error: Info: Warning, confirm

```
<apex:pageMessage severity="warning" strength="1" summary="this is warning"></apex:pageMessage>
<apex:pageMessage severity="error"   strength="1" summary="this is error">  </apex:pageMessage>
   <apex:pageMessage severity="info"    strength="1">this is info</apex:pageMessage>
   <apex:pageMessage severity="confirm"    strength="1">this is info</apex:pageMessage>
```



Input tag:- Take input from user

apex:inputCheckbox :::: checkbox

apex:inputField :::::: input field

apex:inputFile :::::::: upload the file

apex:inputHidden :::: hidden

apex:inputSecret ::::: password

apex:inputText ::::: text

apex:inputTextarea ::::: cover letter/ large content

E.g:

```
<apex:form >
Enter name:          <apex:inputText title="Hello"  /> <br/>
Enter cover letter: <apex:inputtextarea /><br />
enter check:          <apex:inputCheckbox />
</apex:form>
```

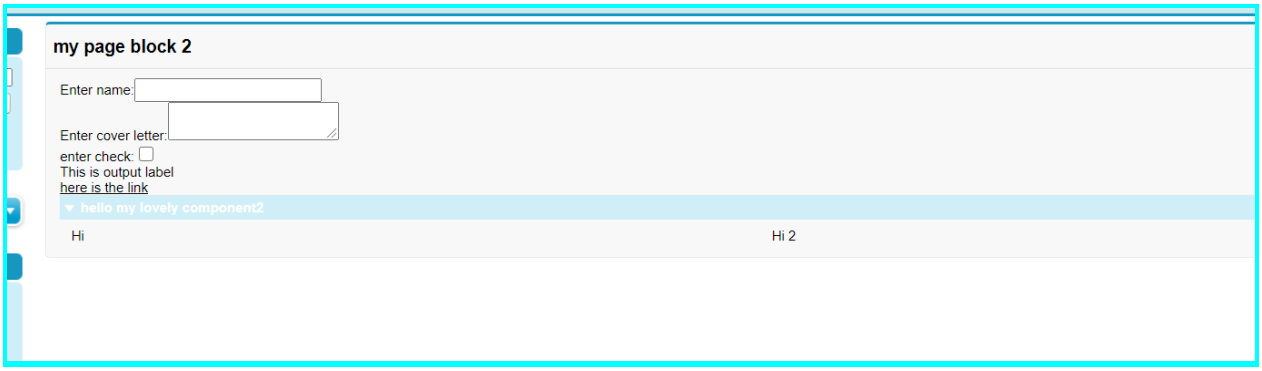
OutPut tag: display output on the screen

```
<apex:form>
<apex:outputLabel > This is output label </apex:outputLabel> <br />
<apex:outputLink >here is the link</apex:outputLink> <br/>
</apex:form>
```

Value = “” for=””

Value is used to display the text.

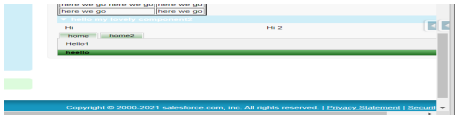
For is used for id so that output tag and input tag can be correlated.



Some important tag:

1) **Apex column**
2) **Apex tab**
3) **Apex param**
4) **Apex Message**
5) **Apex form**
6) **Apex toolbar group**
7) **Apex toolbar**

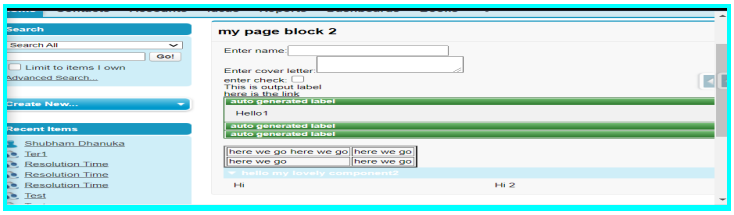
```
<apex:toolbar >
<apex:toolbarGroup >heello  </apex:toolbarGroup>
</apex:toolbar>
```



8) **Apex panel group**
9) **Apex panel bar**
10) **Apex panel bar item**
11) **Apex tab panel**
12) **Apex panelGrid**
13) **Apex detail**
14) **Apex repeat**

E.g:

```
<apex:panelGroup >
<apex:panelBar >
        <apex:panelBarItem >Hello1</apex:panelBarItem>
        <apex:panelBarItem >Hello2</apex:panelBarItem>
        <apex:panelBarItem >Hello3</apex:panelBarItem>
</apex:panelBar>
</apex:panelGroup>

<apex:panelGrid columnClasses="2" columns="2" border="2">
        <apex:panelGroup >
        <apex:outputText > here we go</apex:outputText>
        <apex:outputText > here we go</apex:outputText>
        </apex:panelGroup>
<apex:outputText > here we go</apex:outputText>
<apex:outputText > here we go</apex:outputText>
<apex:outputText > here we go</apex:outputText>
</apex:panelGrid>
```

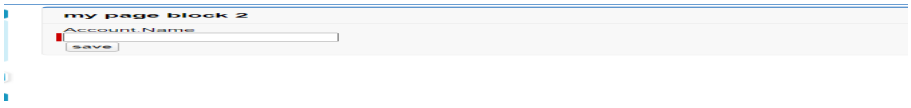


Action Method:
When an event occurs call action method.
By default define in standard controller: Save, quicksave, edit, delete, cancel, list

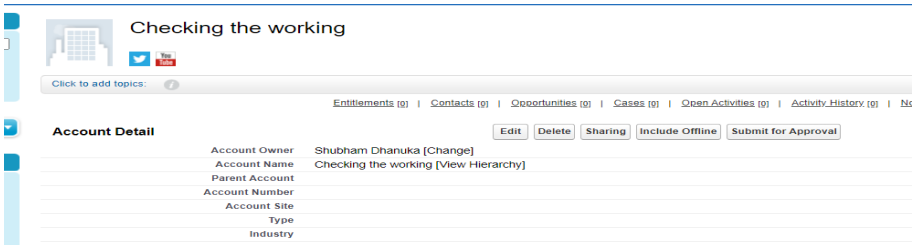
How do i call an action method in markup: by using ! sign. {!save}
How do i define an event in markup:
<apex:commandButton action="">
<apex:commandLink >
<apex:actionPoller> ::: periodically call an action
<apex:actionSupport>
<apex:actionfunction>
<apex:page>
E.g:
<apex:commandButton action="{!save}">

Associate a standard controller to visual force page:
<apex:page standardController = "" >
</apex:page>
Standard controller has there own inbuilt Action method define in standard apex class,
Save, quicksave, edit, delete, cancel, list
How to invoke an action method?
Button, link, page loaded, method/event (standard or custom) invoking.
E.g:

```
<apex:page standardController="Account">
  <apex:pageBlock title="my page block 2" >
    <apex:form >
    <apex:outputText >Account.Name</apex:outputText>
    <apex:inputField value="{!Account.Name}"/>
    <apex:commandButton value="save" action="{!save}"/>
     </apex:form>
  </apex:pageBlock>
</apex:page>
```



It will create a record

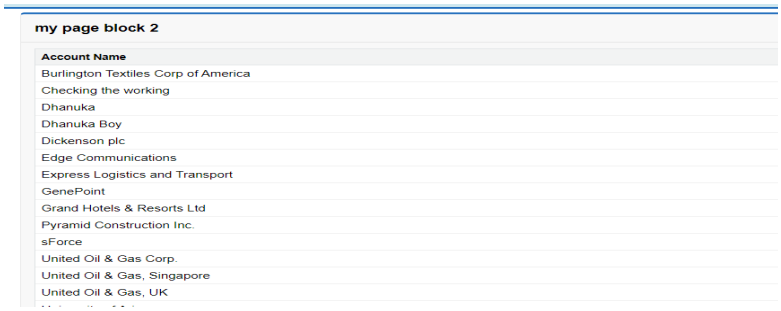


Fetch all the records from standard controller:

<apex:page standardController = “Account” recordSetVat = “accounts”>
</apex:page>

E.g:

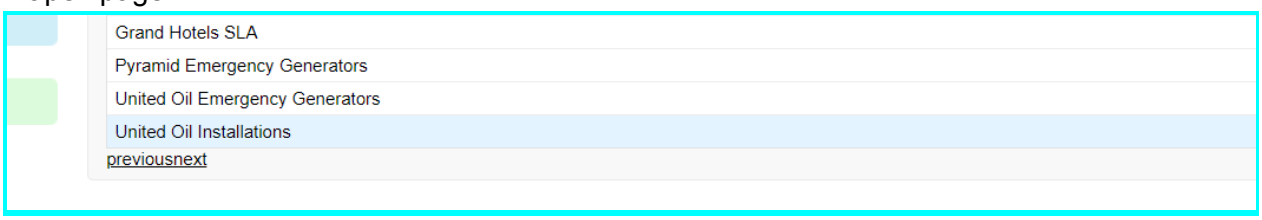
```
<apex:page standardController="Account" recordSetVar="account">
  <apex:pageBlock title="my page block 2" >
    <apex:form >
    <apex:pageBlockTable value="{!account}" var="a">
    <apex:column value="{!a.Name}"/>
    </apex:pageBlockTable>
     </apex:form>
  </apex:pageBlock>
</apex:page>
```



Note: Filter and list view some of the standard action methods are also defined in the standard controller. (list view/ filtration and go)

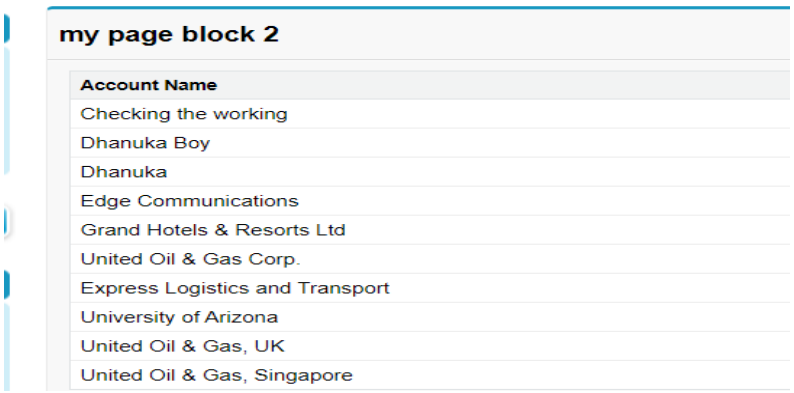
Standard List Controller: by default provided by force.com (pagination)
ActionMethod: // already given definition in apex class.
List, save, quick save, cancel, first, last, previous, next
Pagination:

```
<apex:page standardController="opportunity" recordSetVar="account">
  <apex:pageBlock title="my page block 2" >
    <apex:form >
    <apex:pageBlockTable value="{!account}" var="a">
    <apex:column value="{!a.Name}" />
    </apex:pageBlockTable>
    <apex:commandLink action="{!previous}" value="previous" />
    <apex:commandLink action="{!next}" value="next"/>
    </apex:form>
  </apex:pageBlock>
</apex:page>
```



Associate a Custom controller to visual force page:
Completely wanna define apex class and functionality by your own.
Custom controllers run entirely in system mode.

```
<apex:page controller="MyClass" >
  <apex:pageBlock title="my page block 2" >
    <apex:form >
    <apex:pageBlockTable value="{!account}" var="a">
    <apex:column value="{!a.Name}" />
    </apex:pageBlockTable>
    </apex:form>
  </apex:pageBlock>
</apex:page>
```



Overwrite a standard controller functionality into a visual force page: (extension)
Extension controllers run entirely in user mode.

```
<apex:page standardController="Account" extensions="MyClass" >
  <apex:pageBlock title="my page block 2" >
    <apex:form >
    <apex:pageBlockTable value="{!account}" var="a">
    <apex:column value="{!a.Name}" />
    </apex:pageBlockTable>
    </apex:form>
  </apex:pageBlock>
```

AuraComponent:

---

1) Application: For Application Testing(Including Component to see one App)
2) Component: HTML part (User Interface/ showing part)
3) Style: CSS Part
4) Controller: javascript part (declaring)
5) Helper: second Javascript part to call method (definition)
6) Render: web service callout/ automate process
7) Document: Documentation about component
8) Design: Re Usability design (Properties of component)
9) SVG: Graphic(Shape)

Server side Controller calling:

1) Aura : Controller=" "

Server Side Annotation:

@AuraEnabled

Naming convention of Defining controller in javascript and apex side:

Do, handle:- client side

Get: server side

Common DataType:

Type : HTML

Object : APEX

Sobject : Dynamic Apex

Attribute Defines

Type
Name
Value/ Default

Label
PlaceHolder
Description

Required
Unique

Id/class/ tag

event/action

Text
Text Area
Long Text
Rich Text

Number
Phone
Email
Password

Radio
Checkbox
Toggle
Range

Link/
URL/Address

Form/ table
section/ div
Button
Fileupload
Display image

Style

Border
Margin
Padding

Opacity

Fontsize
Fontfamily
Fontstyle

Display

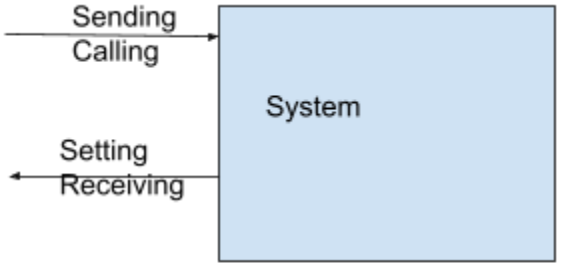
Color

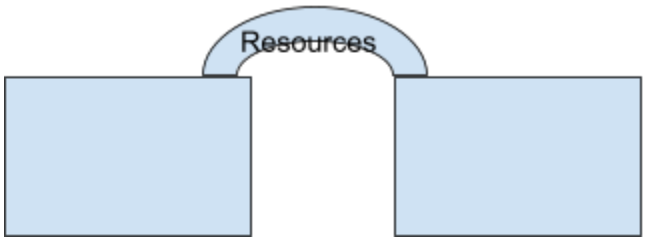
Lookup
ExtenalLookup
Master details
Rollup summary

Auto number
Formula field

Date
Time
Date/time

Picklist
MultiPickList
Depend PickList





Initializer tag like html
<aura:component>
</aura:component>

Layout and layout item:
<lightning:layout>
<lightning:layoutItem></lightning:layoutItem>
</lightning:layout>

For showing output:
<ui:outputText value="{!v.variable}" />
<lightning:formattedText value="{!v.variable}" />

For Iteration:
<aura:iteration items="" var="">
</aura:iteration>

For Button:
<lightning:button label="" onclick=""/>
<lightning:input type="button" label="" />

For attribute:
<aura:attribute type="" name="" value=""/>

Attribute usages in HTML:
For variable: {!v.attributeName}
For Function: {!c.Client_Controller}

Attribute usages In Javascript side:
For Variable: {v.attribute}
For Function: {c.server_Controller}

HTML attribute finder in JavaScript:
component.find("id").get("v.label")
event.getSource("id").get("v.label")

Get the value/ call server side controller:
component.get("c.server_controller");

Setting Value
component.set("v.attribute", "variable_input");

E.g:-
response.getReturnValue()
actionResult.getReturnValue()
Get the value from server side controller (value will be in JSON/Class Type)
So assignment variable type should be non primitive
Component.set("v.contact", response.getReturnValue());

Showing Value inConsole Log:
console.log(JSON.stringify(component.get("v.groupstructureList") , null, 4))

Complete Picture:

Var action = component.get("c.server_controller");
// PERFORM THIS ACTION AFTER SERVER METHOD GETS CALL
action.setCallBack(this, function(){
component.set("v.attribute".response.getReturnValue());
// **component.setParams("", "");**
});
//PERFORM ACTION ON SERVER(Queue that)
$A.enqueAction(action);

LIGHTNING WEB COMPONENTS (LWC)

Component Name should be in Camel Case

1) **HTML File**
2) **CSS File**
3) **JavaScript File**
4) **SVG File**
5) **XML File**

Please note that HTML file is not a mandatory file. (utility items etc.)
Only mandated file is Javascript and metadata files.

Architecture:
Bottom layer is Application Layer on top of that Base component, Experience component

Base Component:
Icon
Button
Badge

Experience Component:
Related List
Chart
Path
Chatter feed (utility items)
Through Lightning app builder Base component and experience component can be placed on the application layer.

Standard web organization :
Ecma
Tc39

Lightning web components support all maturity of the latest web standard.
1) Simplification
2) security
3) Cross browser support
4) Lightning data services
5) custom element / decorator / dom
6) event

Why LWC not VF or aura components??
Reason: it does the rendering on the client side rather than rendering it from the server side.

Earlier:every response goes to the server and server processes and sends the response to the client. First it loads the entire javascript and then structure and styling.

But now: whatever needed goes to server rest processing can be done on the client side itself.
I mean to say business logic, data needed only goes to the server, structuring part done on client side itself.

Visual force does server side rendering that is understandable that we need to do client side rendering so we are not using any more.
Aura does Client side rendering than why we are using lwc.
Because aura creates a framework in json format that is not in html and javascript that the browser can understand directly, after that it parses to HTML and javascript which takes time to load.

So LWC directly creates or uses HTML and javascript (structuring and logic) that the browser can understand and it also does client side rendering.

MetaData file defines where we use this component in our salesforce org.

E.g:
isExposed = true ::: available to use
<targets>
<target> lightning_AppPage/ lightning_HomePage / lightning_recordPage </target>
</targets>

HTML:
Lightning_card : Creates a Container
Which contains Header, Body, Footer

Import track from lwc;
@track: Binding lightning component structure to javascript

Tool: visual force studio code for writing the code.

LWC is one way bound only.

No special annotation for LWC on server side controller.
It is same @AuraEnabled

Variable uses:
hello.html

```
<template>
        {greeting}
</template>
```

Hello.js

```
Import {LightningElement} from lwc;
Export default class Hello extends LightningElement{
Greeting = ‘ Hello’;
ChagePropertyValue(){
  This.greeting = ‘ world‘;
}
}
```

Getter and Setters:

```
<template>
        {itemName}
</template>

Import {LightningElement} from lwc;
Export default class Hello extends LightningElement{
upperCaseitemName;
Get ItemName(){ return this.upperCaseItemName; }   // automatically call
// whenever value change
Set itemName(value){ this.upperCaseItemName= value.toUpperCase(); }

}
```

If condition :

```
<template>
     <template  If : true = {areDetailsVisible} >
          These are the details
     </template>
</template>

Import {LightningElement} from lwc;
Export default class Hello extends LightningElement{

areDetailsVisible = false;
}
```

Iteration:

```
<template>
    <template  for:each = {contacts}  for:item=”contacts”>
          <p key = {contacts.id}>
         {contacts.Name}
         </p>
    </template>
</template>
```

JS File

```
Import {LightningElement} from lwc;
Export default class Hello extends LightningElement{
Contacts = [
      { Id: ‘0000000000’,
        Name: ‘shubham’
      },
     { Id: ‘0000000000’,
        Name: ‘shubham’
      }
];
}
```

Value define:

HTML:

```
{!x}
<lightning-input label=”” onchange={mn} > </lightning-input>
```

Javascript:

```
Import track from lwc;
@track x = “” ;
mn(event){
This.x = event.target.value;
}
```

Selector:

Don't use id:

For specific to class:

```
.className{

}
```

For complete component:

```
:host{

}
```

Condition Iteration:

HTML:

<lightning-input type=”checkbox” label=”” onchange=”” ></lightning-input>
<template if: true = {x}>

</template>

Javascript:

@track x =””
mn(event){
This.x = event.target.value;
}

Loop Iteration :

HTML

<template iteration:it={} >
 {it.value}
</template>

Javascript:

@track cityList = [‘’,’’,’’]

Overview:

<template>
<lightning-card>
<lightning-layout>
<lightning-layout-item>

<lightning-input type=”” name =”” value=”” />

</lightning-layout-item>
</lightning-layout>
</lightning-card>
</template>

Equivalence in between lightning component

Equivalence of components

salesforce developers

Visualforce component	Lightning web component
apex:pageBlock	lightning-card
apex:pageBlockButtons	Set actions slot on lightning-card
apex:pageBlockSection	lightning-accordion and lightning-accordion-section
apex:pageBlockSectionItem	lightning-layout and lightning-layout-item
apex:toolbarGroup	lightning-layout and lightning-layout-item
apex:panelGrid	lightning-layout and lightning-layout-item
apex:panelGroup	lightning-layout and lightning-layout-item
apex:tabPanel	lightning-tabset
apex:tab	lightning-tab
apex:repeat	template for:each or iterator
apex:pageBlockTable	lightning-datatable
apex:dataTable	lightning-datatable

lightning - namespace

Visualforce component	Lightning web component
apex:inlineEditSupport	lightning-datatable with inline editing in editable columns
apex:image	lightning-platform-resource-loader
apex:stylesheet	lightning-platform-resource-loader
apex:includeScript	lightning-platform-resource-loader
apex:map	lightning-map
apex:form	lightning-record-form lightning-record-view-form lightning-record-edit-form
apex:input	lightning-input lightning-slider
apex:inputCheckbox	lightning-input type="checkbox" lightning-input type="checkbox-button"
apex:inputFile	lightning-input type="file" lightning-file-upload
apex:inputHidden	lightning-input class="slds-hide"

Equivalence of components

salesforce developers

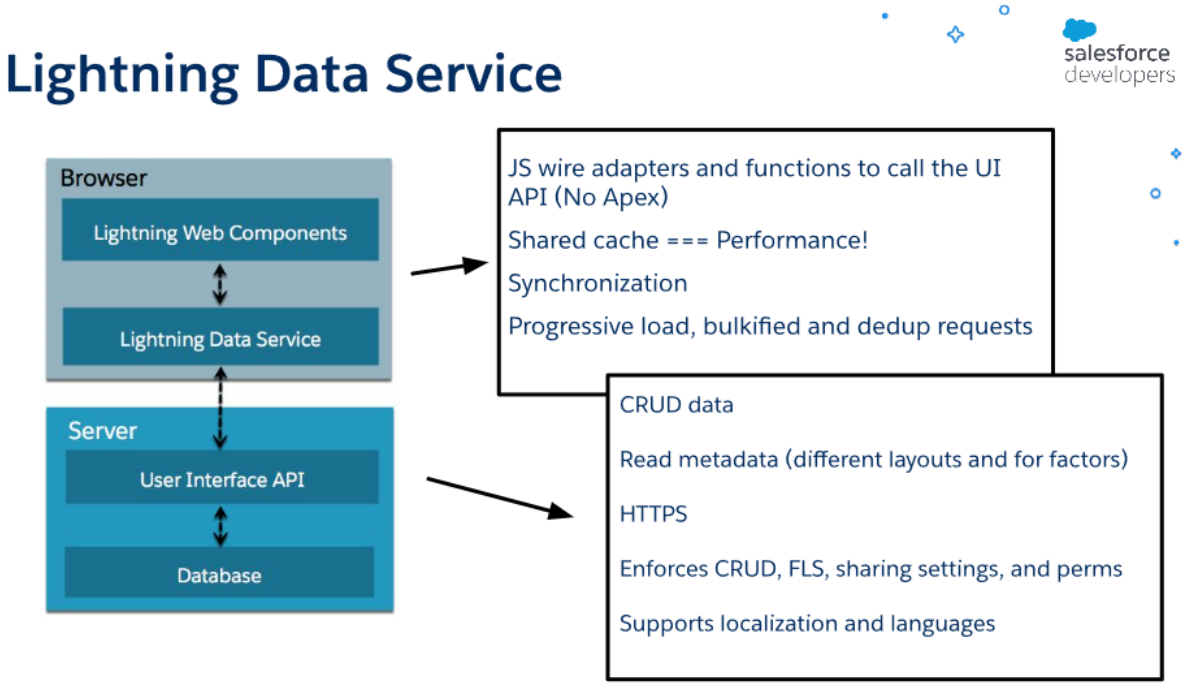
Visualforce component	Lightning web component
apex:inputSecret	lightning-input type="password"
apex:inputText	lightning-input type="text"
apex:inputTextArea	lightning-textarea
apex:inputField	lightning-input-field
apex:selectCheckboxes	lightning-checkbox-group
apex:selectList	lightning-combobox or lightning-dual-listbox
apex:selectRadio	lightning-radio-group
apex.outputLabel	Set label attribute on lightning-input
apex:outputField	lightning-output-field
apex:outputLink	lightning-formatted-url

Visualforce component	Lightning web component
apex:outputText	lightning-formatted-datetime lightning-formatted-number lightning-formatted-rich-text lightning-formatted-text lightning-formatted-time
apex:commandLink	lightning-button with bare variant
apex:pageMessage	lightning-platform-show-toast-event
apex:messages apex:message	Custom validity on lightning-input
apex:pageMessages	Automatic for lightning-record-form Use lightning-messages in lightning-record-view-form or lightning-record-edit-form

Overall Architecture: And Lightning data services:::

Lightning Data Services

Instead of making a separate call from your lightning components to salesforce do your operation at once in lightning and then communicate to salesforce



NOTE:::::

Stateful and stateless:

Whenever executing a server side controller, we need to pass the parameter because it does not maintain any state of the previous executed method .

@auraenabled (cacheable = true)

Use it if u wanna use **@wire annotation** and wanaa handle cacheable processing.

No syn, Asyn, just remote connection or promise

So the main point to use wire and LDS is to get latest data and make server call only once needed,

Client side we are using cache mechanism (wire mechanism)

For single call to server we are using **(LDS(lightning data services))**

Interoperability ::: in between lightning components: (vf, aura, lwc)

Lightning messaging services

Parent child relation: only expose api annotation.

Sibling: lightning messaging services

Function and Event

CallBack Function:
In Javascript Function can also be passed as an object in another function

Asynchronous JavaScript:
Way of Writing :
Function as a parameter

1) **Passing as parameter :**

```
Const Message = function(){
console.log("good to go");
}

setTimeOut(message, 3000);
```

2) **Anonymous Function :**

```
setTimeout(function(){
console.log("good to go");
},3000);
```

3) **Arrow :**

```
setTimeOut(()=>{
console.log("good to go");
},3000);
```

Event as Function ():

```
<button id="callback-fn"> click here </button>
document.querySelector("callback-fn").addEventListener("click", function(){
        console.log();
});
```

Event: (Something happens automatically or by human interaction and do something)
Packet to pass the information from one component to another component

EventHandler / aura handler/ Lightning component handler ::::
Can be put anywhere in your html page/ javascript Not necessarily put in scripting tag.

Events are basically used to pass the value from one component to component or communicating between one page to another page. **(Pass the value in the component)**

Type of event handler:

1) **System Event/ Platform Event :-**
   System, Platform Event: automatically triggered during component lifecycle.

2) **Browser Event :-**
   browser events are predefined in JS API/**user interact with browser**

3) **Component Event/ Application Event :-**
   A component event is fired from the instance of a component. It can be handled either by the component who has fired this event or any other component in the hierarchy that receives the component event and has a handler defined to handle that component event.

Event Navigation:

Events occur by any component and move in 2 fashion toward the component in which the function is defined.

1) **Bubble Phase**: bottom to top in hierarchy to root component (inner to outer)
2) **Capture Phase**: bottom => directly top⇒ hierarchy to source component
   (inner => root => outer)

   By default, it is bubble phase

   E.g: Bubble Phase:

   Customer => invoices and show total invoices amount on customer

   E.g: Capture Phase:

   Fire the event do subscription and unsubscription

Browser Event : -

Example:

Alert
Onclick
Onchange
Refresh
Click
Error
Focus
Keydown
KeyPress
Load
Unload
Drag
Mouse up, down, out, over, move
Resize

1) **onclick =””**
2) **onmouseover=””**
3) **onmouseout=””**
4) **onLoad=””**
5) **onunLoad=””**
6) **onResize=””**

Platform Event :-

System Event :-

Custom Event Defining and Using:

Customize Event:-

1) **register the event**
2) **define the event**
3) **fire the event**

You can use aura:handler for calling a handler that is defined in some other components.

Directly @ Component level

Or @ Controller level

Define component extensible = “true”

Define component extends=”c.base”

<aura:handler name=”” value="{!this}" action="{!c.init}" /> (Construction)

```
<aura:handler name="init" action="{!c.getContactsList}" value="{!this}" />
```

helper.thisMethod1()

1) Register Event
   Registering Event:
   <aura:registerEvent name=”Event_name” type=”file_name_where_event_define” >
   </aura:registerEvent>
   <aura:handler name=”init” value=”{!this}” action =”{!c.doinit}” /> (Like Constructor)
   Pass init value

2) **Controller will get and set value and fire**
   Var comevent = component.getEvent(“Event_name”);
   comevent.setParams(“”:””);
   comevent.fire();

3) **Render will get event and perform the action**
   <aura:handler name=”Event_name” event=”file_name_where_event_define”
   action=”{!c.do}” />

And U can also Use standard Event to handle action :
E.g: `onclick="{!c.handleClick}"`

```
<aura:handler name="init" action="{!c.getContactsList}" value="{!this}" />
```

it like a constructor for OOP:
<Aura> --> lightning specific keyword same as <apex> in salesforce classic.
<aura:handler> --> is used to handle standard / custom events.
name="init" --> this is defining the name of aura:handler
value={!this} --> passing the current information to controller
action ={!c.doInit} --> calling the "**doInit**" of the controller.

Firing the Event:-

1) **Click on a button**
2) **Click on a Custom Link**
3) **Click on Extension/ Component**
4) **Execute When Related Record Modified**

Bubble and Capture Phase:

Note:
Attribute: html to show content
console.log() : client side
system.debug(): server side
With in a class Define sequence of execution in Constructor

Calling Server Method:
At Server: @aura Enabled: Allowing them to call.
At Front: Controller =" " , c.method name: Defining which Class to call

Table:
1) Object Type
2) List (Column)

DATA, ACTION, NAVIGATION:
Controller, Render
Controller: which is getting value
Through Event we pass value from controller to rendering
Render: which is performing task based on the value (Browser rendering value based on calc)

Controller in Context of Apex:
Standard Controller:
Ability to call the Data without writing code with some standard functionality or apex code
Like: one record, all record, saveRecord, DeleteRecord etc

Controller vs Helper:
Controller: specific for that component
Note: in apex helper javascript is used as shared resource that is use by multiple component
Event handler is used to pass the value and also to call certain set of function to perform task

Custom Controller: Writing your own piece of code and then show
Extension:
enhancement in standard controller:
manipulate child records along with a parent or a
Enhancement in custom controller :
(this is often overlooked and is a way to provide common functionality across a number of pages).

Extension:

1) For Apex Class → .cls
2) For Trigger → .tgr
3) For VisualForce Page → .page
4) For Custom Object → __c
5) For Metadata → __mdt
6) For Big Object: __b
7) For Component: c.
8) For Variable: v.

Annotations:

@AuraEnabled

makes your methods available to your Lightning components Both LWC, AURA

@AuraEnabled(cacheable=true)

Using this annotation eliminates the need to call `setStorable()` in JavaScript code on every action that calls the Apex method.

@Deprecated

that can no longer be referenced in subsequent releases of the managed package in which they reside.

New subscribers cannot see the deprecated elements, while the elements continue to function for existing subscribers and API integrations.

@Future

Use the `future` annotation to identify methods that are executed asynchronously. `@future (callout=true)`

must be static methods, and can only return a void type

cannot take sObjects or objects as arguments.

ONLY Primitive data type allowed in arguments.

No Sequence of execution

Should not used in getter and setter method

Should not used in any annotated method

Invocable Method:- Calling Apex in process builder/ Rest API callout of a method

```
@InvocableMethod
public class AccountQueryAction {

@InvocableMethod(label='Get Account Names' description='Returns the list of account names corresponding to the specified account IDs.' category='Account')

public static List<String> getAccountNames(List<ID> ids) {
    List<String> accountNames = new List<String>();
    List<Account> accounts = [SELECT Name FROM Account WHERE Id in :ids];
    for (Account account : accounts) {
            accountNames.add(account.Name); }
        return accountNames;
} }
```

@InvocableVariable

Use the InvocableVariable annotation to identify variables used by invocable methods in custom classes.

@JsonAccess

The @JsonAccess annotation defined at Apex class level controls whether instances of the class can be serialized or deserialized. If the annotation restricts the JSON serialization and deserialization, a runtime JSONException exception is thrown.

Apex REST annotations: global static Apex method must be.

enables you to expose an Apex method as a REST resource

- @RestResource(urlMapping='/yourUrl')
- @HttpDelete :**HTTP DELETE request is sent, and deletes the specified resource.**
- @HttpGet: method is called when an HTTP GET request is sent
- @HttpPatch : and updates the specified resource.
- @HttpPost: and creates a new resource

  @HttpPut : and creates or updates the specified resource.

@IsTest : test class and test method

```
@isTest(SellAllData = true)
@isTest(OnInstall=true)
@isTest(isParallel=true)
```

@TestSetup

Methods defined with the `@testSetup` annotation are used for creating common test records that are available for all test methods in the class.

One testsetup method is allowed for each class.

Test class execution first execute testsetup method create test data then execute any other methods

If any method make any changes in test data, all will be roll back after that method execution

Entire test data will be roll back after class execution.

TEST FACTORY AND UTIL CLASSES

Note: is any method having `@isTest(SeeAllData = true)` for that method or class testMethod is not supported.

@TestVisible:

This annotation to allow test methods to access private or protected members of another class outside the test class.

E.g:

```
public class TestVisibleExample {
        // Private member variable
        @TestVisible private static Integer recordNumber = 1;
        // Private method
        @TestVisible private static void updateRecord(String name) {
         // Do somethin   }}
@isTest
private class TestVisibleExampleTest {
         @isTest static void test1() {
        // Access private variable annotated with TestVisible
        Integer i = TestVisibleExample.recordNumber;
        System.assertEquals(1, i);
        // Access private method annotated with TestVisible
         TestVisibleExample.updateRecord('RecordName');
        // Perform some verification } }
```

Package Installation:

1. `@NamespaceAccessible`
2. `@ReadOnly`
3. `@RemoteAction`
4. `@SuppressWarnings`

Tools/ extension which can be used:

Salesforce Inspector: Data / Field Data related
Salesforce Advanced Code Searcher: Searching the files / PB
Salesforce organiser: Deploying the stuff

Code Coverage: Testing the Class (which coverage covering or not)
Google Dev Console: for javaScript related error
DevConsole : Apex Code
Visual studio code: To write the code

---

1) **Salesforce LWC Editor/ (visual studio code) / Google Dev Console**
2) **Developer Console/ Apex Debugger**
3) **Salesforce inspector/ Advance Code Searcher**
4) **Code Coverage**
5) **Organizer**

Governor Limit: (Need governor limit to maintain the system)

To do anything salesforce first check gov. Limit same like checking recycle bin for data
If it is hitting gov limit, salesforce will throw an error no matter u used catch block or not

How are they doing this ?

Ans: Limit.getDMLRows(), if it so then add error
Limit.getDMLStatements

Limits Method:-

Limit Method Use to get information all the governor Limit
Limits.getLimitQueries();

Limits.getDMLRows()(record proceed) , Limits.getDMLStatement() (insert li)
Limits.getLimitDmlRows()(10000);,Limits.getLimitDMLStatement()(150)

Limits.getLimitCpuTime();
Heapsize(), FutureCalls() etc

In get if Limit is mentioned, it will tell you what limit is of that operation

Some important Method:

Http WebCallouts: (100) : Limits.getLimitCallOuts()
Method for future allocation: (50) : Limits.getLimitFutureCalls()
Total Heap Size: (6 MB) : Limits.getLimitHeapSize()
SOQL (100) , SOSL(20), DML Operation(150), DML Rows(10000)

Transaction: set of operations executing as a single unit
Entire Transaction will be roll back if one operation fails to maintain sync

---

Best Way to Write Code to Avoid Governor Limit:

Task: Write a Trigger which creates invoices when a customer status changes to Active.
And Do the update the status of that invoices to ‘i am done by avoiding governor limit’

Ans:

Code Optimizing:

1) PMD Error
2) SonarQube

Deploying:

1) Github
2) GitLab
3) Tortoise Git

Service Cloud

---

1) Computer telephony Integration (Telephony integrated System(who, where and what)) (Who is calling, where to route and what is the issue)
2) Field Service Management (Workforce to Fields worker (what2 do, where2 do)
3) Omni Channel Routing (Agent Availability & Customer waited Time)
4) Einstein for services (Tips)
5) Service Console (Tool to handle)
6) Knowledge (Knowledge Base)
7) Dashboard & Reporting

Medium to Connect:

1) Social Medium / Communities
2) Chat/ Messaging
3) Phone / Email
4) Customer Portal/ website
5) Bot / FAQ
6) Snap in(video Accessibility)

Reporting and Dashboard

Reporting Type:

1) Tabular
2) Summary
3) Matrix
4) Joined report type

Chart and Highlighting of records...
Scheduling and Subscribe (Condition and preference based)

Custom Report: 1) Field label change 2) default index to show field
Bucket Fields(Group together), permission set group, field set group.

Note1: **Report and Dashboard setting** if Custom field is not showing in report creation (enable)
Note2: MD relationship fields always show in objects so no need to create custom reports type just create reports on child objects.
Note3: **Profile→ System Setting** (organisation setting for profile)

Dashboard → Chatter for snapshot
Enable chat feed setting for dashboard object

---

1) Report
2) Choose Report Type (Tabular, Summary, Matrix, Custom Report)

3) Select Column to appear
4) Grouping Row and Column
5) Ordering and Filtering
6) Chart and Run

Lookup ---> Cross Filter
MasterDetails--> Custom Report
Difff-2 --> Joined Report

Omni Channel Routing:

(Agent Availability & Customer waited Time) **(Skill Based Routing)**
(Take input from customer, analysis it, routed to right person to resolve with their availability)
(All Communication channel**(Multiple Socialized Account)** into one console to help)
Multiple person can be answered @ one time

Knowledge Base:

1) Article Created by Agent
2) Search in Community
3) Search on google baba/ web search
    1) Easy to Create
    2) Easy to Read
    3) Easy to Modify

Einstein Analytics:

(Who is getting how many cases and how much time it is taking to resolve a case)
(What are anomalies, performance , Spending time with customer)

Einstein Escalation Predictor:

Analysis:

What is the most common organization issue?
What need to work on based on severity?

Appointment Scheduler:

Meeting Scheduling

Book an Appointment by seeing date and time availability
Appointment notification to customer and agent with go to meeting link
Customer Become unavailable: so reschedule it
Agent Become Unavailable: Move it to other teammates and send change notification.

Workforce Management

Scheduling according to the locale and manage workforce
When most workforce needed in a day??
(Analysing based on per day call receive, customer waited time and etc)
Macro and QuickText:
Quick Action and History:
Synonym/ Pre populated Field/ Data category
Suggested Article/ Attach Article/Email
Tag defined on the product
Email Action and Email Notification:

Macro: Small Programme (set of instruction) Used to perform Quick tasks.
QuickText: Quick text to give quick reply
Both Macro and Quick Text are Standard Object
Mass Quick Action : create a record, update a record
ON Object
Feed:
1) Feed Action (Quick Find)
2) Feed Tracking (Quick Find)
3) Feed Filter (Quick Find)

---

1) Case Creation
2) Case Assignment
3) Case Escalation / reAssignment
4) Auto response
5) Case Process / resolved
6) Knowledge Article

Case Process:
Support Process ⇒ Quick Find → Support Process
Record Type → SupportProcess

Queue
Case Assignment

Escalation Rule
Escalation Action

Product
Entitlement Template
Entitlement Process → Service Contract → Milestone **(Service Provided to Customer)**
Milestone (Standard Component)(Alert in Working)
Milestone Action

Knowledge Article:-
Custom Object installed

Article Type: **(Structure of a Article) :It must be there before importing**
Datacategory: Classification of Data in category

knowledge component
Article Record Type
DataCategory---> Role
Topic:
Create, Approve, Publish, Feedback

Recommendation Article

Analysis:
Number of case created
Number of case closed
No need to know:
How much time spent
How much Article got created

Optimization:
Knowledge Search
Case Routing
Case Resolution

Mitigate Risk:
Proper Queue/ proper resources
Simplify/Simplification

Note:
Mass Transfer Record : Changing the ownership of a person
All Ownership related rule will be recalculated
All Manual Sharing Rule will be deleted
Field Service Lightning: Coordinate with Service Engineer across multiple territory

---

1) Essential Services (CLI, SDK, VDI)
2) Monitoring
3) Analytics
4) Security
5) Networking and Storing Services
6) BackUp

Community Cloud:

1) Community : Simple information sharing of knowledge article
2) Community Plus : Report and Dashboard
3) Partner Community : Contact Lead Opportunity are available

Account are of 2 type:

1) Person Account (can only login in Community and community plus)
2) Business Account (can login all 3 type)

Community User must have Account associate to login else won't be able to do
Account → Enable Partner → Enable partner User

Community like (Service, Support Portal, third party tracker) etc

Self-service portal:

Community URl:

1) Login
2) Customers
3) Developers
4) Partners

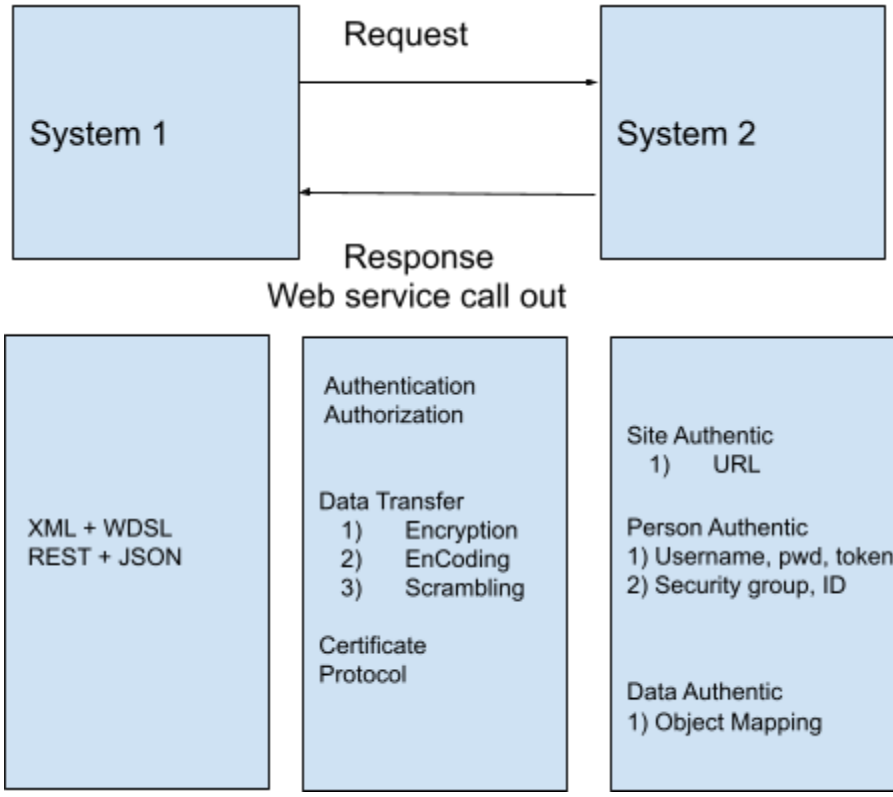
LiveChat:
Social Profile
Social customer service
Social conversation Component
Social Persona

==========

API:

1) Public
2) Private
3) Hybrid
4) Composite

Application, presentation, session



1) Session setting
2) Session management
3) Platform Encryption
4) Certificate & Key Mang

1) Custom Label
2) Custom URL

3) Custom Setting
4) Remote site Setting
5) Named Convention

6) Trusted URL for redirect
7) CSP Trusted sites
8) Outbound connection setting

Troubleshooting:-
pIng/ tracert/ nslookup/ netstat

Tool Useful in development and simulation :

Workbench
CURL
Postman

==========

Operation:

Obtain detailed information about a Salesforce object, such as Account, User, or a custom object.
Perform a query or search.
Update or delete records.
==========
sObject : is a parent class of all standard and custom object

- /services/data—Specifies that we're making a REST API request
- Version : what version we are using to call

==========
HTTP Method:

Get: get the data in plain text
Post: get the data in mask form
Put: update the record
Patch: update a particular thing of a record(no need to update, insert entire thing)

---,

REST API:

1) URI : unique identification
2) Method : get
3) Header : Metadata of request
4) Body : Request

==========
For retrieve data:

For Query :

1) URI: /query/?q=SELECT+Name+From+Account+WHERE+ShippingCity='San+Francisco'
2) Method: Get
3) Request Tpe: JSON

For Specific:

1) URI: sobjects/account/describe
2) Method: Get
3) RequestType: JSON

For Deploying data:

1) URI: sobjects/account
2) Method: POST
3) Request Type: JSON
4) Request Body

SOAP API:

Tool:

1) SOAP UI
2) WSDL File (Web service description file)
    1) Enterprise → Specific Org → Strongly Coupled
    2) Partner → Several org → Loosely bounded → does not impact any specific
    a) Apex WDSL
    b) Certificate
    c) Tooling

Project → WSDL FiLE
Login with **UserName , password, security token**
Create with **session id and Xml tag and Define URI**

BULK API:

Create a Job process and then define data.

Jobs: A job specifies the type of operation and data object we're working with. insert, update, upsert, or delete many records asynchronously,

Job process:-

URI: jobs/ingest
Method: **Post**

Header: JSON
Body:
{
  "operation" : "insert",
  "object" : "Account",
  "contentType" : "CSV",
  "lineEnding" : "CRLF"
}
Add Data:
URI: jobs/ingest/7502y000002aU3XAAU(JOb_Id)/batches
Method: put
Header(content-Type): text/csv
Body:
"Name"
"Sample Bulk API Account 1"
"Sample Bulk API Account 2"
"Sample Bulk API Account 3"
"Sample Bulk API Account 4"

Close the job:
URI: jobs/ingest/7502y000002aU3XAAU
Method: **patch**
Header: JSON
Body;
{
  "state" : "UploadComplete"
}
Monitor:
URI: jobs/ingest/7502y000002aU3XAAU
Method:Get
Header: Json

Or you can use
Bulk data load in salesforce Setup
Success Result: obs/ingest/7502y000002aU3XAAU/successfulResults
Failed Result: obs/ingest/7502y000002aU3XAAU/failedresult

Streaming API

Streaming API is your radar. It lets you define events and push notifications to your client app when the events occur.
PushTopicObject

E.g: Emp API: (Showing Notification to Users about edited records)

EMP API: (Message Streaming API)

Real time notification it provides it on the same record some one do updation.

(Record -> user -> Modifying Record → get notification)

Use case :

1) **Load the data using record data (in view/ edit mode/ take action)**
2) **Fire platform event**
3) **Do subscription**
4) **Do unsubscription**
5) **Refresh it/ save it / load it / get new Record**
6) **The same user creates a case again from that window**
7) **Multiple-use create a case from**

At the same time fire, the event capturing

Connected API:

1) **OAUTH, SSO**
2) **SAML**
3) **WDSL**

IDP: internet service identity provider

SDP: service provider

1) Download IDP certificate

Create or select IDP certificate

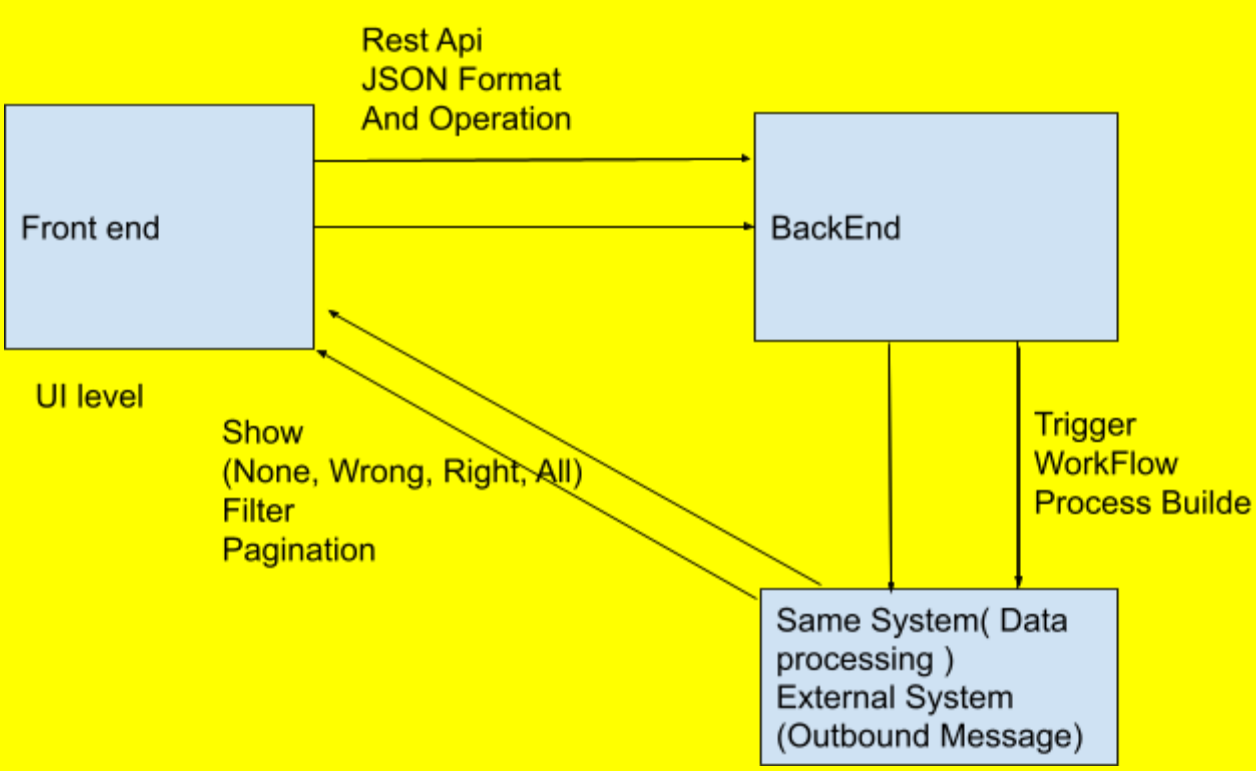
Generate IDP End points for site Authentication

2) Remote site setting in SDP (Authentication URL/ Site Authentication)

3) Single Sign On (SAML Sign ON)

4) federation id (Person Authenticator)

FRONT END ----> BACKEND : JSON



Data Management:

1) Import
2) Export
3) Update
4) Transfer
5) Mass Delete
6) Duplicate and matching rule
   Check whether they are creating duplicate records and prevent them from creating

7) External ID:
   Go to the external system, take id from there and fill it here in this field.

Salesforce field: Set this field as the unique record identifier from an external system

For connection:

1) Inbound and outbound message.
    A) Platform Encryption
    B) Certificate and Key management
    C) Named Credential: **For Secure API Key in Salesforce and Make Callouts to External System**
    D) Custom Label: **Variable**
    E) Custom URL: **Custom URL**
    F) Custom setting, Custom Metadata : **Data**
2) Inbound and outbound setting
    a) Outbound Connection Setting: Salesforce feature using external service
    b) CSP Trusted Site : Server as trusted Site
    c) Trusted URL for Redirect: Salesforce to external site redirect
    d) Remote site setting : Salesforce to external server

Static Resources, My Documents
Tool: Event Management, Jobs

---

1) Language
2) Data structure
3) Dynamic Programming
4) Algorithm
5) Math
6) Asymptotic Notation

Delegated Administer:(changing a lot)
custom tab
custom object
custom field
custom picklist
reset of password

Territory management (account criteria)
regardless off the ownership
multiple forecasting
Transerfing and delete will not work in case of opportunity
everything work in account case

Learning Note:
How to handle Recursive triggers?
Ans: define variable to check the condition before executing the trigger.

changing the ownership:
all ownership related rule will be recalculated
manual sharing will be deleted

Object specific quick action:
associate record that are being viewed
create a log

global quick action:
Create quick action type
display on the home page

Code Practices:

Show Picklist value/ Section of code/ items
None, Wrong, Right, Any
E.g:
1) Field Dependency: Hiding till it doesn't make sense to them
2) Validation Rule : to select any
3) Code: if select wrong: show error, else select right, save it
4) provide option to select Any

Searching/ filtering:
2) Provide flexibility to Search

Notification:

1) **Outbound Message**
2) **Post to chatter**
3) **Email Notification**