

## 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
- 

**Global Action:** Action can be accessed from anywhere, no matter which page you're on

**Utility Bar:** Mostly used component

**Activities:** Action for particular screen

**Detail page link:** viewing information of related records

---

- 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 HulchI 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 Vlp 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	Node
Deletion	Constructor
Updation	
Searching	
Sorting	
Traversing	

---

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 **Undefined**

---

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 )

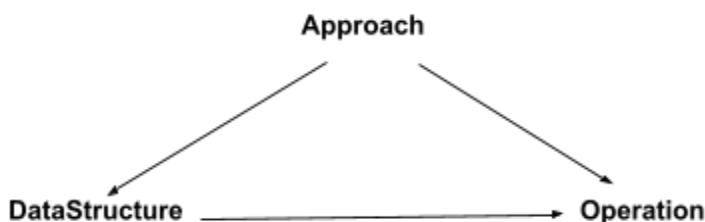
```
static final Integer PRIVATE_INT_CONST = 200;  
static final Integer PRIVATE_INT_CONST2;
```

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

-----

LIST OF PRIMITIVE DATA TYPE

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

LIST 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(ji.1);
List<Json> ji1 = {{1:2}} :	System.debug(ji1[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

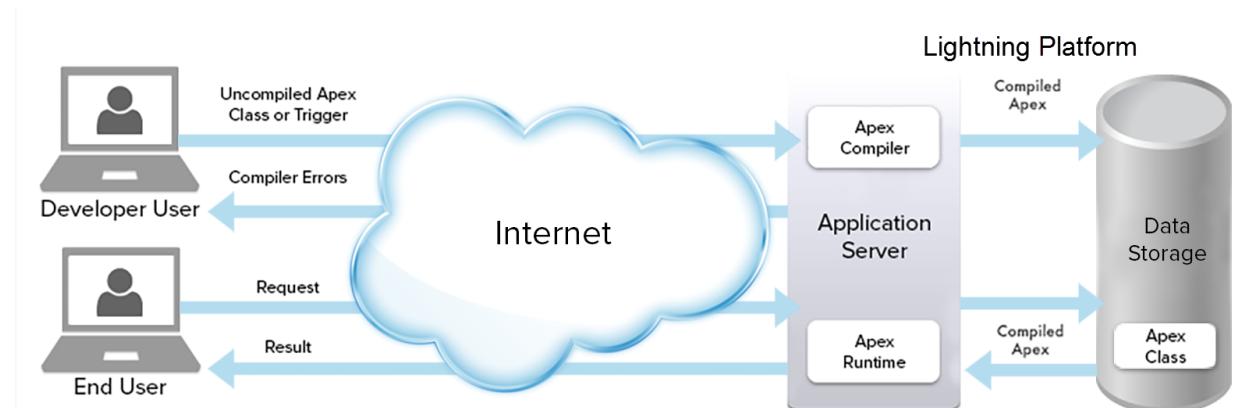
**JSON Format:** Index, Key, Value

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 DateTlme(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 constructor');  
    }  
    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 Method 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 beginning 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 class.

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
- 

**Utility Class:** Utility class are those class which share common functionality in b/w component

**Wrapper Class:** Calling Multiple classes of a component together to get a particular result.

**Constant Class:** Define all global constants to use in multiple classes.

**DataFactory:** Define common Data to get.

---

### 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> schemaMapOfAllSObject = 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.getType()+';'+dfield.isUnique()+';'+dfield.isExternalId()+';'+dfield.isNullable());
}
```

---

## 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 store  
        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.

#### **Messaging API: Messaging API To send an email**

**Messaging EMAIL Services (Inbound and outbound) (single, mass Email)**

**Email sending and receiving to the customer( Feed, Promotion) (Single, Mass)**

#### **Email services: to attach email and reply to thread**

Create a apex class

Expose that class to email address

Use that email address for inbound

Note: parsing of email address and attachment already can be done by it

**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 (list)
- 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 InvoiceGenerator 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 always 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('here we go');  
    System.debug(Trigger.new);  
  
    if(Trigger.new.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 InvoiceGenerator 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 InvoiceGenerator 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 useaddError to display based on certain condition

E.g:

```

trigger InvoiceGenerator2 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 assertEquals**

System.assertEquals is used to check expected and actual result

It will throw an error if we don't 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.assertEquals();
    }
}
```

**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 will get 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 = hwsTestData.createAccount();
        ac.recordTypeId = '';
        Insert ac;
    }
}

Public class hwsTestData{
    Public static account createAccount(){
        Account acc = new Account();
        acc.name = 'accountName';
        acc.recordTypeId = HWS_Util.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 parallelly.**

**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 devRecordTypeld=  
Schema.SObjectType.Contact.getRecordTyeInfosByDeveloperName().get('Customer').getRecordTypeld();
```

### **Getting PickList value and data:**

Using Dynamic apex, we can achieve this.on object of type picklist, 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 callout for web service
```

### **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<Subject>

// 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 FristBatchc implements Database.Batchable<Subject> {
// 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:**

Running future and batch together and want to sync the operation together then use 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.enqueueJob(cn); // Id JobID = system.enqueueJob(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 object 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 by 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                      Search in File ( cmp inside cmp, page, community page)  
                                    Debugger

//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:- Add debugger statement in javascript

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) Visual Force Page
  - 2) AURA COMPONENT
  - 3) LIGHTNING WEB COMPONENTS (LWC)
- 

### HTML: DOM Structured Language

#### Tag / Element Name

- 1) Block element : which takes entire width and start with new line  
<div> <p> <header> <footer>
- 2) Inline element : which does not take entire width and doesn't start with new line  
<img> <a> <span> <label>

#### Element Attribute

#### Element specified Name (Class/ ID/ Tag)

Note: only in dom, for comment we use <!-- --> rest everywhere we use // /\* \*/

### CSS (Cascading styling sheet) :

Selector {property : value }

**Selector:** HTML doc tag

**Property:** attribute ( color, background-color, font-size )

**Value:**

**Margin >> Border >> padding >> content**

#### Id vs class:

Id : particular to the item

Class: group of a item

Three way to include css in your dom structured file

- 1) Inline : within file
  - 2) External file : separate file from any other folder in same CDN
  - 3) External file : separate file from any other CDN
- 

#### Note:

External css included(Salesforce design system) Use CDNS to import salesforce design system

```
<link rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/design-system/2.15.
      3/styles/salesforce-lightning-design-system.min.css" />
```

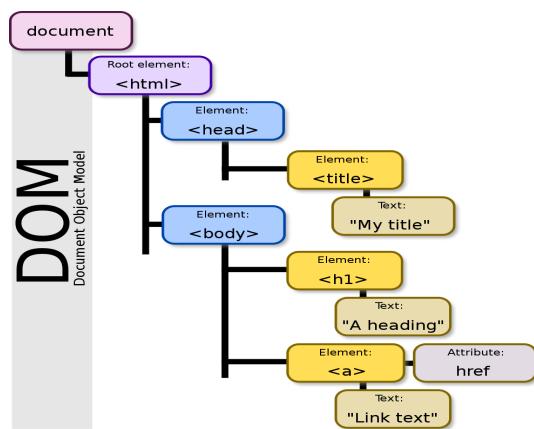
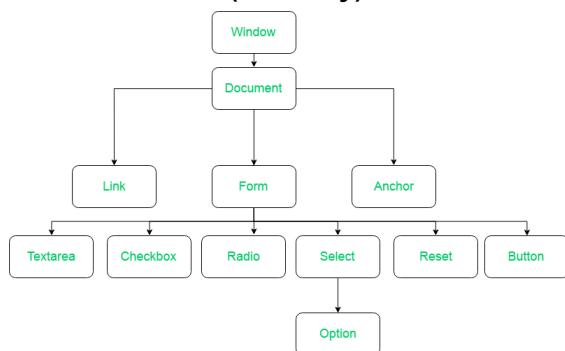
Now use it to style your dom.

For multiple classes either you can use separate div or in one div

```
class="slds-box slds-text-heading_large"
```

**Note:**

- in dom we mainly use = :: because of different index property  
 In css we mainly use : :: because of same index property  
 In javascript or any other :: Depending on the schenerion
- 

**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.getElementById() )****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**

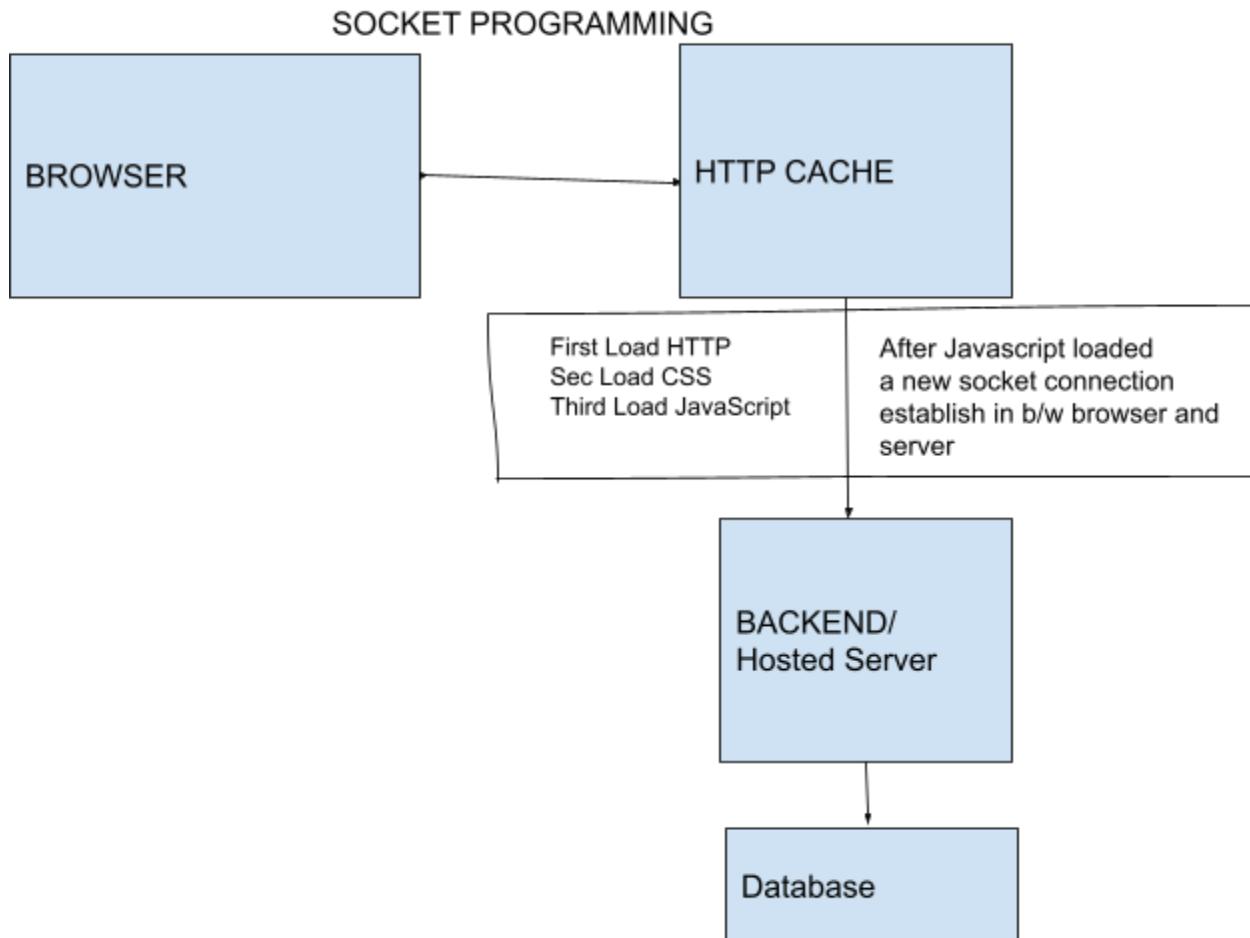
---

### **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

Memory / Environment Variables  Code Hierarchy to store Provide runtime env.	Code/ Thread of execution  One line at a time it gets executed	
------------------------------------------------------------------------------------	----------------------------------------------------------------------	--

**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("<b>LESSON 1: PRINT HELLO WORLD USING VARIABLES</b><br><br>");

**For Specific dynamic html attribute : ( it will change the content of HTML)**

```
document.getElementById("demo").innerHTML = "";
```

---

**Log:-** Console.log(x);

**Debugging:-** debugger

---

**GET PARTICULAR ELEMENT ID Query Selector : in apex not needed**

document.getElement	: particular member // individual item
document.getElementById("IdName");	: particular class // group of diff tag
document.getElementsByClassName("");	: particular package// group in same tag
document.getElementsByTagName("");	

```
document.querySelector();  
document.querySelectorAll();
```

**IN LWC it becomes this.template.querySelector()**

**In aura it becomes 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 or HTML DOM Contains each property of tag

**Means, type, value, name, label, placeholder, description, required, unique, event/action, id**

**Example:**

```
Var name = document.getElementById("name").value;  
Var name1= document.getElementById("").etc....
```

```
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:**

Number: to create number define var a =10; var a =10.5;

Bigint: to create bigint define var a = 10n;

string: to create string define var a ="stri" , var a ='stri'

Boolean : to create var a = true, var a = false;

**Note:**

To see data type of variable : typeOf()

By default in javascript if value is not initialized means it's undefined.

In apex if value is not initialize means its value is null

Undefined value data type is undefined, Null value data type is object

---

**Equality Comparison :**

= : assign

== : comparison

==== : comparison 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**

Array : to define array var a =[];       ::: index format, starts with 0

object : to create var a ={};       ::: key value format, key should be primitive type

**Note you can access with [] and .**

**Example:**

```
var obj1 = {"name":"shubham"};  
document.write(obj1.name + "<br/>");  
document.write(obj1["name"] + "<br/>");
```

```
obj1.enter = "my name";  
obj1["enter2"] ="emter2";  
document.write(obj1.enter + "<br/>");  
document.write(obj1["enter2"] + "<br/>");
```

**Where do we use:** in case of nested Or in case of space in the key

---

### **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

## **Array Method**

Map() :: loop over the array and return new array based on value return  
forEach() :: method for each array element

every() :: return true/ false if every element satisfy the condition  
filter() :: return those element which satisfy the condition  
some() :: return true/false if atleast one element satisfy the condition

sort() :: sort  
reduce() :: remove the variable by one from let to right

### **Syntax:**

```
array_name.methodName(function(currentItem, index, array){  
    Return  
});
```

### **Example:**

```
Var x = 5;  
let arr = [2,3,4,5,6,7,8,9];  
  
let arr1 = arr.filter(function(currentItem, index, array){  
    return currentItem >x;  
});  
document.write(arr1);
```

---

## **Object Method**

Key : Object.keys()

Value : Object.values()

```
let obj1= {"name": "shubham"}  
document.write(Object.keys(obj1));  
document.write(Object.values(obj1));
```

---

**In apex we use different way to access entire key and value information  
For accessing member is same but for entire information format is different**

---

### **JSON Method: ( javascript object annotation )**

JSON.stringify() :: object to string :: server always understand string format  
JSON.parse() :: string to Object

```
let obj1= {"name": "shubham",
age: 23
}
var sr = JSON.stringify(obj1);
document.write(sr);
document.write(JSON.parse(sr).name);
```

---

### **String Method:**

- 1) **toLowerCase()**
  - 2) **toUpperCase()**
  - 3) **slice() :: str.slice(startIndex, endIndex)**
  - 4) **trim() :: remove white space**
  - 5) **indexOf() :: return first index of matching, -1 if no result :: str.indexOf()**
  - 6) **startsWith() :: return true/false :: str.startsWith()**
  - 7) **includes() :: return true/false :: str.includes()**
- 

### **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

**No 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.getMinutes();, sh.getSeconds();

### **Note:**

**typeOf : return the type of the variable.**

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

---

## **Spread Operator: ...**

Usage:

- 1) Expand String
- 2) Combine Array
- 3) Add value to Array
- 4) Combine Object
- 5) Create Shallow Copy of array and object

### **Issue resolved of referencing the object**

Var a = [];

Var b = a;

So if i do any changes in b it will also change the value store in a

So to do shallow copy we use spread operator

Exception: in case of nested we avoid spread operator because it doesn't work there

So for that : we use **JSON.parse(JSON.stringify())**;

### **Example:**

```
// expand string  
let arr1 ="shubham";  
let final1 = [...arr1];  
document.write(final1);
```

```
// combine string  
let arr2 = ["shubham"];  
let arr3 = ["shubham"];  
let final2 = [...arr2, ...arr3];  
document.write(final2);
```

```
// add value to array  
let arr21 = ["shubham"];  
let arr221 = [...arr21, "shubham"];  
document.write(arr221);
```

```
// combine object  
let a = {"1":"2"};  
let b = {"3":"4"};  
let c = {...a,...b};  
document.write(c["1"] + c["3"] +"<br/>");
```

```
// create shallow copy of object  
var d =[...arr2]; d.push("dj");  
document.write(d);  
document.write(arr2);
```

---

---

**Destructuring:**

```
Let arr = [ "", "" ];  
let [arr1,arr2] = [ "", "" ];
```

```
Let obj = { "name": "shubham", "age":23};  
Let {name, age} = obj;
```

Example:

```
let obj1 = {  
  "name": "shubham",  
  age : 23  
}
```

```
let {age, name} = obj1;  
document.write(name);  
document.write(age);
```

---

**String Interpolation:**

Instead of using ", " , + quote and worrying about all spaces etc.

We have `` back ticks

example

```
let name ="shubham";  
document.write(`my name is ${name}`);
```

**Where do we use this : URL formation**

Thing to keep in mind: `` and \${}

---

**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();

**Main purpose of using arrow function is to ensure outer value is get properly when needed  
not pointing undefined in nested function.**

**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**

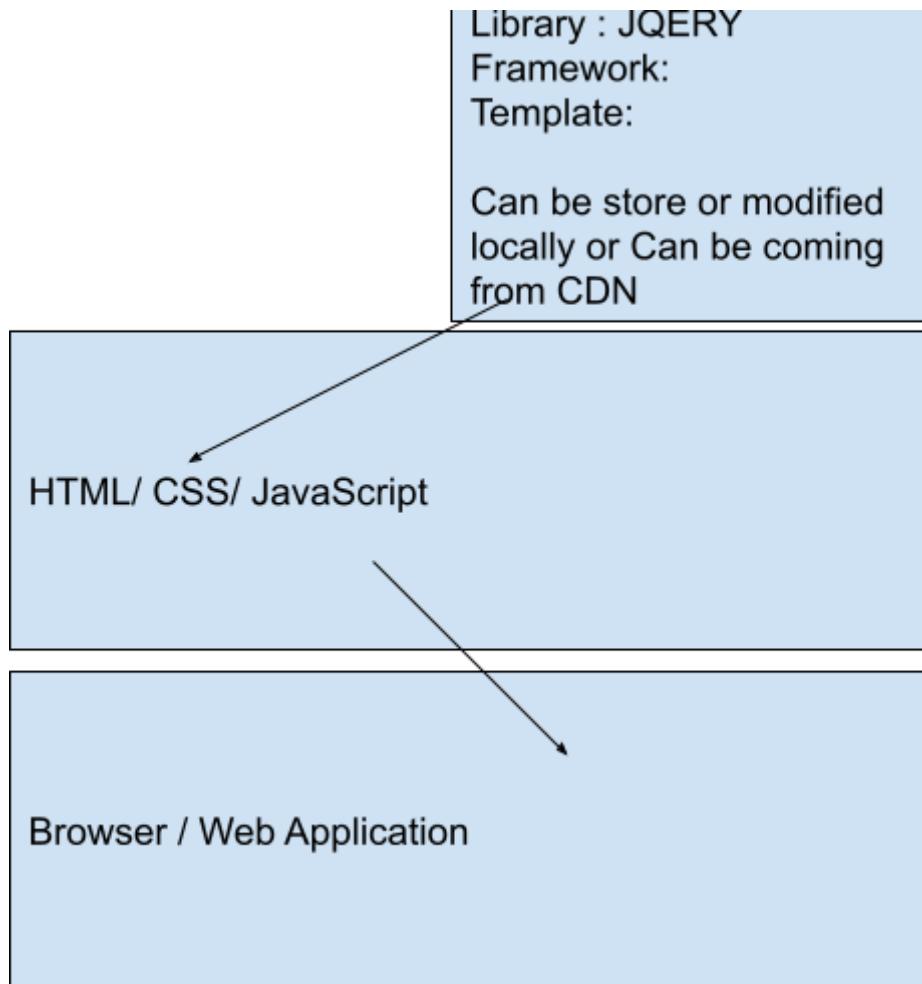
- 9) **Window Class:**

**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

#### **Regular Expression**

**Var reg= new RegExp("E00","i"); reg.test();**

---

**End of Basic Javascript Learning**

---

---

## Advance Concept of JavaScript

---

### Function

---

#### CallBack Function:

In Javascript Function can also be passed as an object/ parameter 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 : // window is parent dom or class

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

###### 3) Arrow :

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

---

### General Business scenario of understanding:

From UI user inserted the input

Those input goes to attribute that is define on dom

Those attribute then taken to client side controller

Client side it perform whatever it needs to perform

Maybe some business logic calculation

data processing and passing to other component by firing the events /data binding

fetching rendered data from cache memory through wire and so on

After That calculation happened through LDS lightning data services

Single call out happened to server

And After verification data got saved to the salesforce database.

---

**Event:** (Something happens automatically or by human interaction and do something)  
Packet to pass the information from one component to another component  
or communicating between one page to another page. ( Pass the value in the component )

---

#### Type of event:

- 1) **System Event/ Platform Event:- (`render()`, `connectedCallback()`)**  
System, Platform Event: automatically triggered during component lifecycle.
  - 2) **Browser Event/ dom event:- (`onclick()`, `onload()` )**  
browser events are predefined in JS API/**user interact with browser**
  - 3) **Component Event / Application Event / Navigation Event:-(`custom 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.
- 

#### Browser Event : -Dom Event: always begin with on keyword

##### Example:

Alert  
Onchange  
OnRefresh  
OnError  
OnFocus  
oneyup  
OnKeydown  
OnKeyPress  
OnDrag

- 1) `onclick =""`
- 2) `onmouseover=""`
- 3) `onmouseout=""`
- 4) `onLoad=""`
- 5) `onunLoad=""`
- 6) `onResize=""`

#### Platform Event :-System Event :- read hook or life cycle of framework/ platform

#### Custom Event Defining and Using:

##### Customize Event:-

- 1) Define action method
- 2) register the event & define the event
- 3) fire the event
- 4) capture the event and parse the value

##### More details below with framework and example

---

### **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

---

### **How Event gets triggered:-**

- 1) Click on a button / action method call
- 2) Click on a Custom Link
- 3) Click on Extension/ Component
- 4) Execute When Related Record Modified

### **In other component how to pass the value or event information:**

- 1) Reusability mechanism
  - 2) Hook
  - 3) Indirect action method
- 

### **Event handling:**

**Some standard method::: useful in movement of event journey in b/w component**

```
event.getSource();      //|      event.getSource().getName();
event.pause()
event.resume()
event.stopPropagation()
event.preventDefault()
```

### **Note:**

event.target.value :: passing through event :: standard browser event

event.detail.value(variable\_name):: passing through component :: custom event

## GENERAL QUERIES::::

How to get to know which one parent and which one child?

If in component 1 we are calling component 2

Then parent component is component 1 and child component is component 2

What is binding ?

If we are passing value from one component to another component then on the child component if data changes then it should be reflected on the parent component also that is called two way binding.

Note:

LWC is one way binding . means child value changes, it doesn't reflect on parent

Aura is two way binding. Whatever value changes on the attribute side it gets reflected on the javascript side as well.

### **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 vs Helper:**

Controller: specific for that component

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

More advantage is the faster rendering and minimizing memory allocation of javascript.

Controller can be called when any event occurs on markup

But the helper function can be called by controller, by helper itself, by rendering and by other comp.

### **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

**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).

---

## **Stateful, Stateless**

**Promise :** which promise to produce some result in future (Multiple ASYNC operation)

### **When to use promise:**

Two async process are together

### **Use case**

- 1) Fetching data from server
- 2) Fetching data from cache
- 3) Loading Multiple files from static resources.

### **Promise has three state:**

- 1) Pending()      ---> (resolve ---> full fill ) -> (reject → rejected)
- 2) Fulfilled()
- 3) Rejected()

### **Promise handling:**

- 1) Handling any standard promise operation
- 2) Creating any promise services and then handling

### **Handling any standard promise operation:**

Example:

```
myName("success1").then(function(result){  
    document.write(result);  
}).catch(function(error){  
    document.write(error);  
})
```

### **Creating any promise services and then handling :**

Example:

```
function myName(data){  
    return new Promise(function(resolve,reject){  
        if(data=="success"){  
            return resolve("Successfully");  
        }else{  
            return reject("Rejected");  
        }})  
}///To see what is return
```

```
myName("success1").then(function(result){  
    document.write(result);  
}).catch(function(error){  
    document.write(error);  
})
```

---

**Finally method is also there**

**Server response is always a promise aysnc call**

**If multiple promise are coming**

```
fetch(url).then(function(result){  
    Return result.json();  
}).then(function(response){  
    document.write(response);  
}  
))
```

---

**For multiple load of script and styling use promise.all**

**See loadScript and loadStyle for more information section (External resources usages)**

---

Communication Architecture:

Framework => Web callout => string => bit => Networking

---

## 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                                                     |

### **Page Tag: 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>

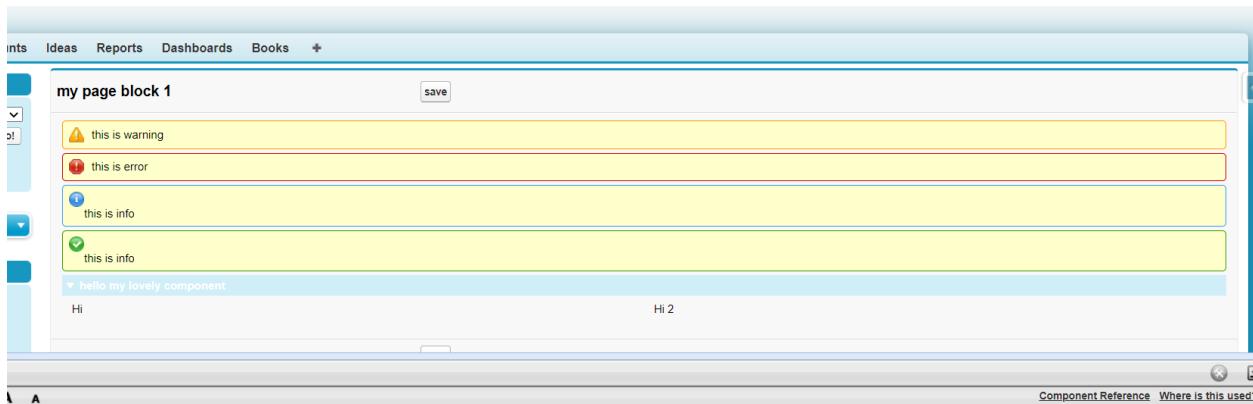
<apex:tabPanel switchType="client">
<apex:tab label="home"> Hello1 </apex:tab>
<apex:tab label="home2"> Hello2 </apex:tab>
</apex:tabPanel>
```



### 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>
```

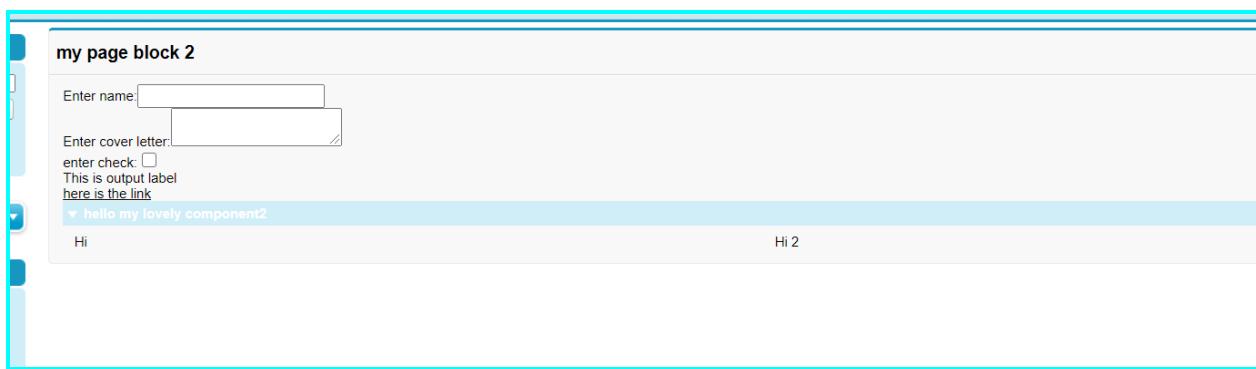
### **OutPut tag: display output on the screen**

```
<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>
```

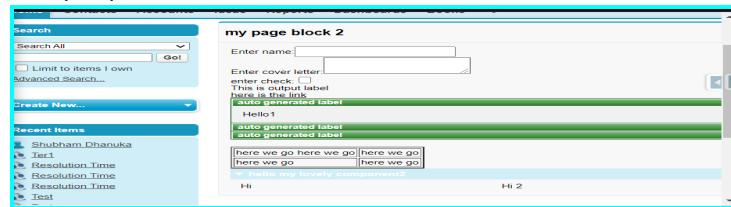


- 8) Apex panel group**
- 9) Apex panel bar**
- 10) Apex panel bar item**
- 11) Apex tab panel**
- 12) Apex panelGrid**
- 13) Apex detail**
- 14) Apex repeat**

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,

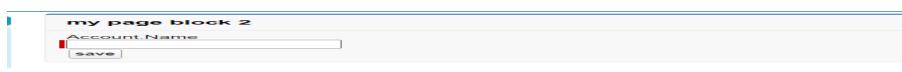
#### Save, quicksave, edit, delete, cancel, list

How to invoke an action method?

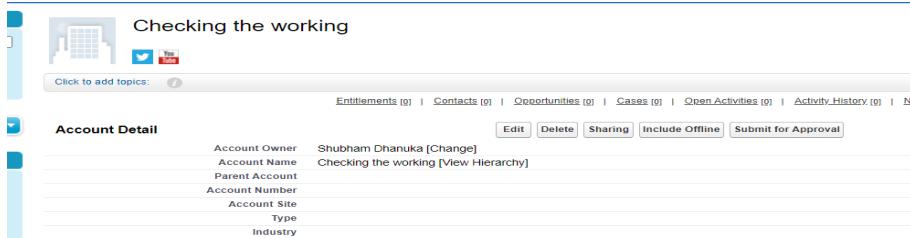
Button, link, page loaded, method/event (standard or custom) invoking.

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>
```

Account Name
Burlington Textiles Corp of America
Checking the working
Dhanuka
Dhanuka Boy
Dickenson plc
Edge Communications
Express Logistics and Transport
GenePoint
Grand Hotels & Resorts Ltd
Pyramid Construction Inc.
sForce
United Oil & Gas Corp.
United Oil & Gas, Singapore
United Oil & Gas, UK

**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>
```

Grand Hotels SLA
Pyramid Emergency Generators
United Oil Emergency Generators
United Oil Installations
<a href="#">previous</a> <a href="#">next</a>

### **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>
```

Account Name
Checking the working
Dhanuka Boy
Dhanuka
Edge Communications
Grand Hotels & Resorts Ltd
United Oil & Gas Corp.
Express Logistics and Transport
University of Arizona
United Oil & Gas, UK
United Oil & Gas, Singapore

### **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>
```

Account Name
Checking the working
Dhanuka Boy
Dhanuka
Edge Communications
Grand Hotels & Resorts Ltd
United Oil & Gas Corp.
Express Logistics and Transport
University of Arizona
United Oil & Gas, UK
United Oil & Gas, Singapore

---

### Aura Component:

---

- 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: Calling Server Method:**

**Client side Definition:-** Aura : Controller=" " c.method name: Defining which method to call

**Server Side Annotation:-** @AuraEnabled Allowing them to call.

---

#### **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

---

#### **Structure, Styling, Scripting, SVG, Metadata all are provided.**

#### **How to use an external library?**

For external library:

First you need to upload in static resources and then you need to use in your component

#### **E.g: For aura component:**

```
<ltng:require
    styles=" {!$Resource.customSR +'/customCss.css'}"
    scripts=" {!$Resource.customSR +'/jquery.min.js'}" />

```

---

**Implements:** It is used to define which interface can use lightning components

Until unless the component is exposed, we can not use the lightning component.

Standard while creating component :::

Tab

Quick Action

Flexi Page

Record Page

Community page

## **NOTE :::**

### **v stands for variable**

```
<aura:attribute name="int1" type="Integer" default="1" />  
{!v.int1}
```

### **C stands for component**

```
{!c.component_name}  
<lightning:input type="button" name = "button" value="button" onclick= "{!c.comp}" />
```

### **Why use it !**

To differentiate in between HTML and Javascript dynamic variables use

### **Why use \$**

To differentiate in between standard and custom component / rerender onchanges vari

### **Why Use V and C**

To differentiate in between variable and component calling

### **Why Use {}**

Values are coming dynamically in structured language that is defined in framework and for the same index. E.g: value =”{!v.attribute\_name}” precedence from left to right like arithmetic op.

### **Why do we use C: while calling components in application??**

Ans: This is the namespace.

---

ID/ Class/ Tag: is used as unique identifier

Placeholder / label: is used as

Default/ value : value defines in that variable

event/ action : fire the action method

Name: used in same section of attribute attribute for one action

( radio, picklist, select/option/ checkbox)

Type: type of attribute

---

### **Dom comment :::**

```
<!-- -- >
```

---

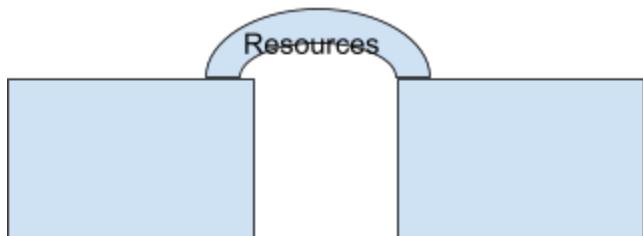
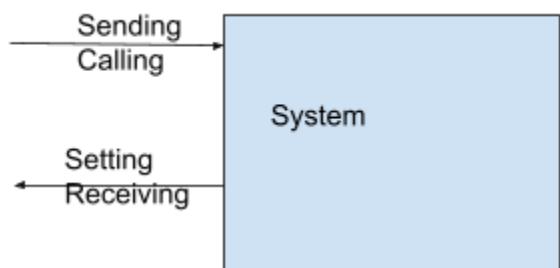
### **How to call one component in another component?**

In component 1 <C:Component2 />

### **How to pass value from parent component to child component ?**

```
<c:Component2 component_2Attribute ="{!v.component_1Attribute}" />
```

Attribute Defines	Text Text Area Long Text Rich Text	Style	Lookup ExternalLookup Master details Rollup summary
Type Name Value/ Default	Number Phone Email Password	Border Margin Padding	Auto number Formula field
Label PlaceHolder Description	Radio Checkbox Toggle Range	Opacity Fontsize Fontfamily Fontstyle	Date Time Date/time
Required Unique	Link/ URL/Address	Display Color	Picklist MultiPickList Depend PickList
Id/class/ tag  event/action	Form/ table section/ div Button Fileupload Display image		



### **Standard Component**

#### **Initializer tag like html**

```
<aura:component>  
</aura:component>
```

#### **Layout and layout item:**

```
<lightning:layout>  
<lightning:layoutItem></lightning:layoutItem>  
</lightning:layout>
```

---

#### **For defining attribute/ Variables in Lightning Component:**

```
<aura:attribute />
```

#### **For taking input from User:**

```
<lightning:input />
```

#### **How to assign user input to attribute variables?**

By Using Expression Value="={!v.variablesName}"

---

#### **Note for Select Option/PickList input tag:**

In select option tag or picklist value tag

We need two attribute to store the value

First attribute is used to store the value from option

Second attribute is used to select the value from select tag which will be final from drop down,

#### **Note for Checkbox/ radio button input tag:**

In checkbox and radio button: name should be the same in order to pick one value

---

#### **For showing output:**

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

---

#### **For Iteration:**

```
<aura:iteration items="" var="">  
</aura:iteration>
```

#### **For Conditional:**

```
<aura:if isTrue="={!v.str}"> <h1>My name is anothy </h1>  
<aura:set attribute="else">False</aura:set>  
</aura:if>
```

---

#### **For Button: ( Note: button label also can changes after event occurs )**

```
<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")
```

```
component.get("v.num");
```

**Get the value/ call server side controller:**

```
component.get("c.server_controller");
```

**Setting Value**

```
component.set("v.num" , 10);
```

```
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.enqueueAction(action);
```

---

## **Styling :**

To get aura framework styling :

### **In application:**

extends = "force:slds"

### **In component:**

Use design system to do your styling

**Note: instead of writing your styling in markup use styling sheet/file to write your code**

### **Styling sheet: Note: .should be attached with .THIS**

.THIS.className{  
}

E.g:

.THIS.MyName{  
    font-size: 100px;  
}

## **JavaScript:**

```
myAction : function(component, event, helper) {  
    component.set(v.list, [1,2,3]);  
},
```

### **Parameter passing: component, event, helper**

Component: which is calling method unique component reference id is passing.

Event: can be any type ( browser, platform, application ) passing.

Helper : helper resource id

Helper method calling :

```
comp: function(component, event, helper) {  
    helper.myName();  
}
```

// in helper class:

```
myName : function(){  
    alert('hi');
```

}

**Note: Custom/Standard Event Accessing in the learning of Event section.**

---

### **Documentation :**

- 1) Prepare the documents
- 2) View the documents

### **To Prepare the document:**

Aura: description : documentation

Aura: example : demo / visual output: you can explicitly define your component it will give preview mode

### **To view the document:**

You need to hit the URL

domainName.lightning.force.com/auradocs/app\_name

### **Render Resources:( HOOK)**

In most of the case aura framework itself takes care but more advanced logic context function are there to use:::

render() : first rendering

rerender() : in between first rendering and undering, if values get change then rerendering gets call

afterrender() : after first rendering happen

unrender() : after time period gets over.

**Browsers first render the corresponding javascript for a timeframe but After a particular time period if you wanna render the particular function again there we use this.**

**(After rendering then-- wanna re render a function )**

**E.g: overview:::**

```
// call the render  
// interact with dom  
// returning the rending value.
```

```
render:function(component){  
    var a = this.superRender();  
    console.log("checking the rendering");  
    return a;  
},  
afterRender: function(component){  
    var a = this.superAfterRender();  
    console.log("checking the after rendering");  
    return a;  
}
```

**Design:**

While adding component in lightning app builder there is option to define component properties

**What are these properties?**

Increase reusability of the component. (same component using while passing refre value)

**How can we define them?**

In component design sheet

**How can we use them?**

In Lightning App builder, while adding components to an application.

**SVG**

Shape // icon on the component

---

**Wrapper Component:**

A Nested Component is called a Wrapper component means two components wrap to one component and then the final component calls all other components.

E.g: Multiple components in one component.

Now suppose multiple components using LDS (lightning data services ) and u wrap all necessary components in one bundle that particular component is call wrapper component.

Encapsulation and wrapping are called in the same context but have little bit different meaning.

---

**Note: while calling the component in application u can also pass the value in component**

---

**Global Value Provider: // standard //**

GlobalId: :: unique id of each component

\$Browser :: isAndroid, isIOS, isWinodwPhone

\$Label

\$Locale :: date, timezone

**E.g: in html tag**

`{!$Browser.isDesktop}`

`{!$Browser.isTablet}`

---

---

## LIGHTNING WEB COMPONENTS (LWC)

---

### PreRequisites of tool and working structure

#### First install prerequisites:

- 1) Java
- 2) Python
- 3) Node

#### Some Tool Configuration:::

- 1) Visual Studio Code
- 2) Salesforce CLI
- 3) Salesforce Extension Pack,
- 4) Some work optimize extension ( live server, code prettier, xml, log analyzer)

#### Need a live server: to start:

**Note: By default salesforce CLI install development server if it doesn't then:-**

**To check sfcli plugins --core**

#### To install Local development server:::

sfcli plugins:install @salesforce/lwc-dev-server

sfcli plugins:update

#### Facing general issues: After npm install

npm install -g node-gyp

npm install --global --production windows-build-tools

npm install --global windows-build-tools@4.0.0

choco install python visualcpp-build-tools -y

npm config set msvs\_version 2017

#### Why do we need to live on a server?

Create a Local development server to render component UI locally.

calls to Lightning Data Service and Apex are sent to your scratch org

---

#### My domain/ To enable dev org::

**My domain will give u a unique url for your dev org.**

**(Enabling org: Allow u to manage org from CLI, view information about scratch org)**

Setup -> My domain

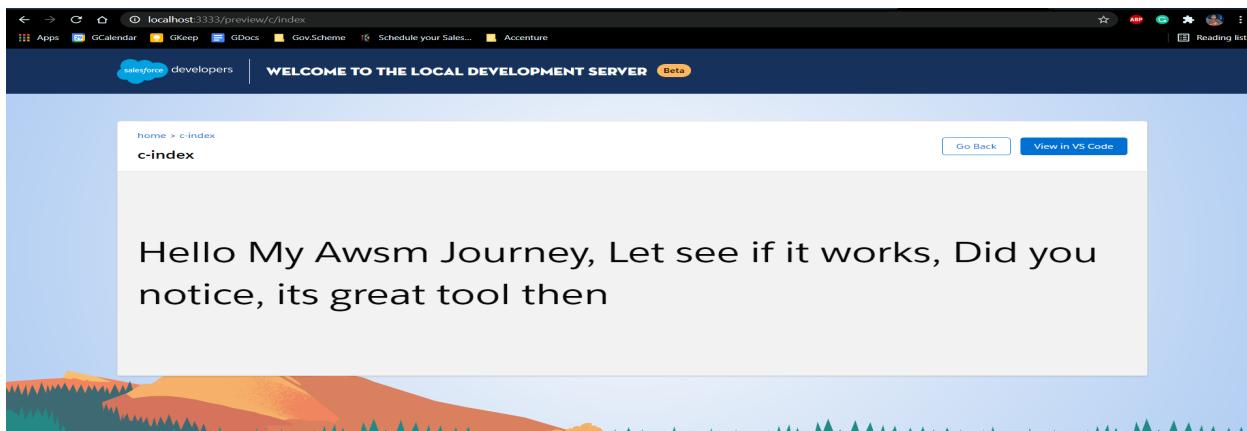
Setup -> dev hub -> enable dev hub

**Authorized dev org/ create a scratch org**

**Authorized dev org** :::: Ctrl+shift + p → authorized org

**Create a scratch org** :::: Ctrl+shift + p → Create scratch org

**Start a local server** :::: Ctrl+shift + p → start local server



**Question:**

**What is the difference between dev org and scratch org?**

Developer org are mainly used for development purpose which has no lifecycle/ time period

And scratch org are temporary org which has life cycle and mainly used for component development in modularized way (life cycle is 1-30 )

And if u want some sample data in your scratch org for that

In Project-scratch-org.json add “**hasSampleData**”:true

---

**Deploying // pushing the Code:**

App creation → Page creation → Component Deployment → Component assign

**Before deployment we need to make our component available:**

In metaData: isExposed = true

**And we also need to configure for which lightning page this component should be available**

```
<targets>
<target>lightning__AppPage</target>
</targets>
```

---

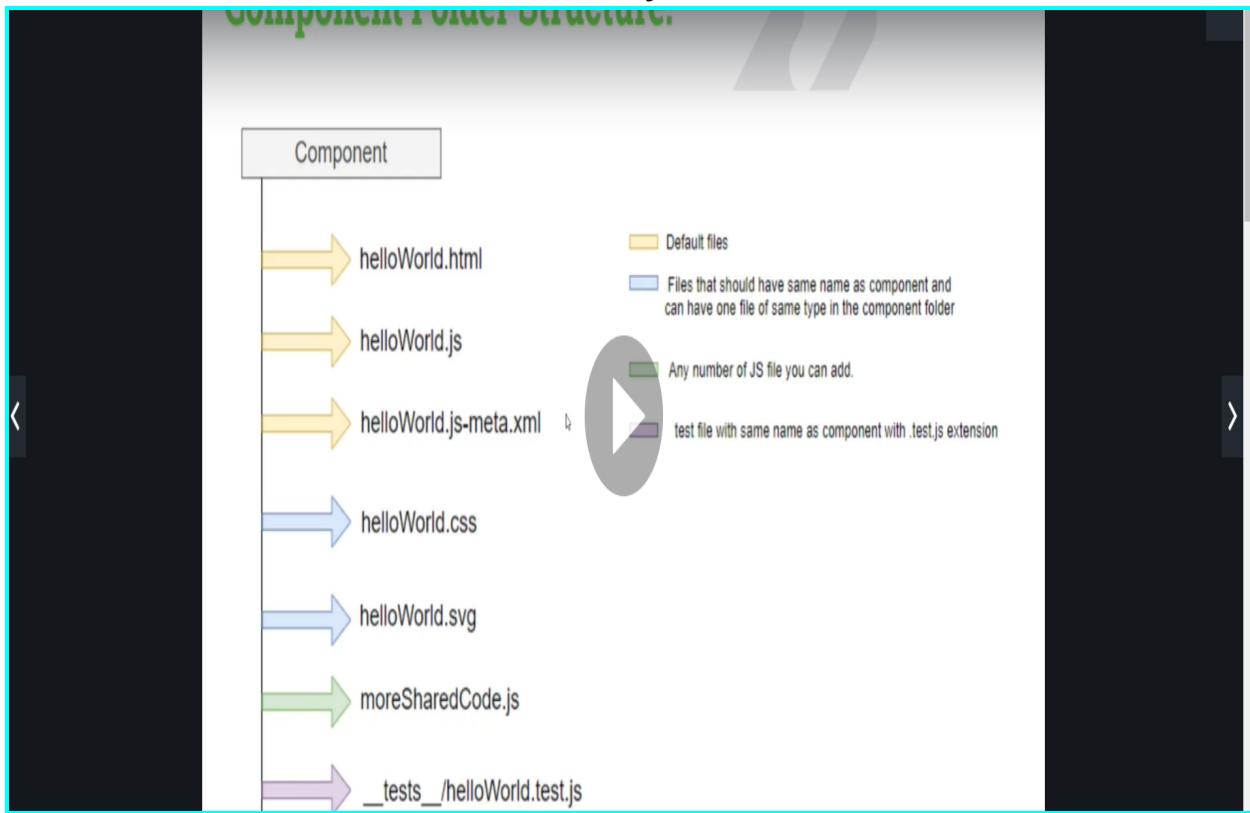
Ctrl+shift + p → Push to scratch org

---

### Component Folder Structure:

1) HTML 2) CSS 3) JavaScript 4) Helper JavaScript 5) SVG 6) MetaData 7) Test Code

Each file name should be the same followed by extension



**Naming Convention:** unique name Should be in entire namespace

**Camel Case:** First letter with small letter and then followed by caps letter, (ComponentName)

**Pascal Case:** All first letter of the word should be in caps (ClassName)

**Kebab Case:** all letter should be small followed by - (Attribute)

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.

---

**Meta Data 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>
```

Meta file : (**for more:- search Meta config in LWC** )

- 1) description can be added for component
  - 2) Component availability can also be done for community
  - 3) Target configuration can also be done, property set/ modification etc
- 
- 1) Component in Tab
  - 2) Component in Utility bar (All can be done in meta config)

#### **Overview:**

```
<template>
<lightning-card>
<lightning-accordion>

<lightning-layout>
<lightning-layout-item>

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

</lightning-layout-item>
</lightning-layout>

</lightning-accordion>
</lightning-card>
</template>
```

---

---

## BASIC OF LIGHTNING WEB COMPONENT

---

**Creating Component:**

Ctrl+shift + p → Create Lightning web component

**Initializer tag: Supports two directive (if, loop)**

```
<template>  
</template>
```

**Division tag: Lightning\_card : Creates a Container Which contains Header, Body, Footer**

```
<lightning-card>  
</lightning-card>
```

**Layout and layout item:**

```
<lightning-layout>  
  <lightning-layout-item> </lightning-layout-item>  
</lightning-layout>
```

**Accordion:**

```
<lightning-accordion>  
  <lightning-accordion-section></lightning-accordion-section>  
</lightning-accordion>
```

**TabBar:**

```
<lightning-tabset>  
  <lightning-tab> </lightning-tab>  
</lightning-tabset>
```

**Table :**

```
<lightning-datable>  
</lightning-datable>
```

**Form:**

lightning-record-form

lightning-record-view-form

lightning-record-edit-form

**Input:**

```
lightning-input  
lightning-input type="text"  
lightning-input type="password"  
lightning-input type="checkbox"  
lightning-input type="file"
```

**For input field:**

```
lightning-input-field
```

**For Text area:**

```
Lightning-textarea
```

**For Radio Button:**

```
Lightning-radio-group
```

**For Checkbox:**

```
lightning-checkbox-group
```

**For PickList:**

```
Lightning-combobox
```

**Output:**

```
lightning-output-field  
lightning-formatted-url  
lightning-formatted-datetime  
lightning-formatted-number  
lightning-formatted-rich-text  
lightning-formatted-text  
lightning-formatted-time
```

**Button**

```
Lightning-button
```

**Toast Message:**

```
lightning-platform-show-toast-event
```

**Resources Loading:**

```
<lightning-platform-resource-loader>  
</lightning-platform-resource-loader>
```

---

**Note for Select Option/PickList input tag:**

Two Time value needs to define

**Note for Checkbox/ radio button input tag:**

Name should be the same

---

## Dynamic Binding:

### HTML → Client Controller

For defining attribute/ Variables in Lightning Component:

For taking input from User:

How to assign user input to attribute variables? : not needed if no option there

If I speak frankly, there is no need for an attribute variable in dom structure.

Because whatever variable we needed we can import in javascript and use in HTML.

For taking input from User to javascript variable:

Use lightning-input and **on\_event** by passing event properties

**Example:**

```
<lightning-button class="MyClas" label=" i am Button"
value='This is great' onclick={handleButton}></lightning-button>
export default class Index extends LightningElement {
    handleButton (event) {
        var string = event.target.value;
        console.log(string);
        alert(string);
        this.handleButton2(5,6);
    }
}
```

More precisely:

### @track operation:

Why do we need it?

Because objects and arrays there is a limit of value monitoring, or they dont track if single value changes in object, to ensure value changes we need this.

How does it monitor?

==== with previous value to current value if not it rerenders the component.

**How to import and use this?**

Import{track, LightningElement} from lwc;

@track objectName ={}  
}

**Better approach: use spread Operator:**

{...this.object\_name, "city":event.target.value}

@track : to hold current latest value because user changes the input

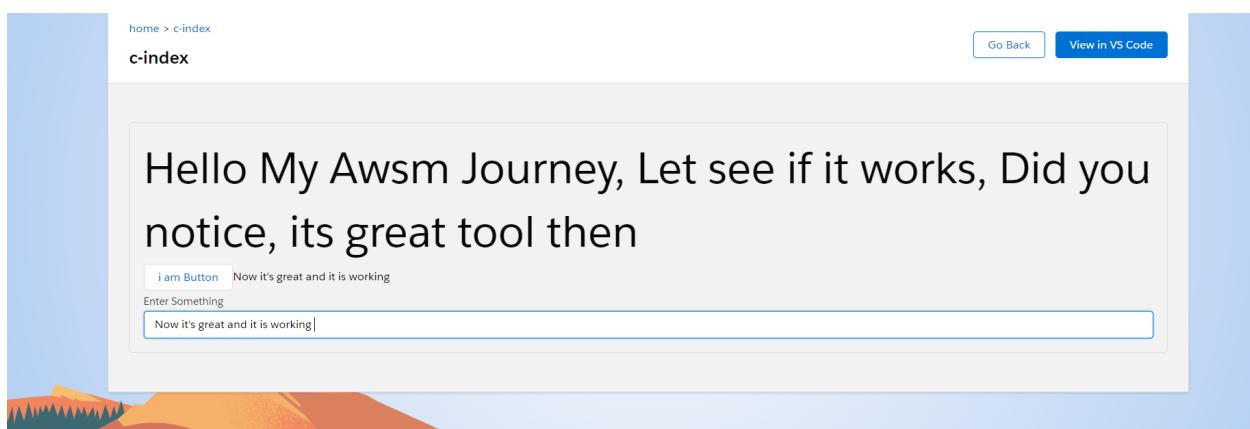
For complex, object data type structure (**Reactive Private property**)

Example:

```
<lightning-input type="text" label="Enter Something"
onkeyup={handleEvent}></lightning-input>

obj_name = {"name": "shubham", age: 23};

handleEvent(event) {
    this.obj_name = {...this.obj_name, "name": event.target.value};
}
```



### Client Controller → HTML

#### Attribute usages In Javascript side:

And in export default class bind with variable

Simply, Variable\_name = field\_Name

#### Attribute usages in HTML:

In HTML: no need to use ! mark

{variable\_name}

#### Note:

Primitive and Object data binding is allowed but not Array or any other non primitive data type

E.g: {name} {obj.name} **but not** {name[0]}, {2+ 2}

### So How will you handle the list element or any operation ??

**First method** is to take the first dedicated value and assign it to a variable then use it.

#### Second method **Getter method:**

#### Example:

```
<lightning-formatted-text value={function_name} >
</lightning-formatted-text> <br/>
array_list = ['12', '3'];
get function_name(){
    return this.array_list[0];
}
```

**Note:**

**It is one way binding**

So javaScript to html changes gets reflected

But from HTML to javascript you need to use **event properties.**

```
function_name(event){  
    this.varName= event.target.value;  
}
```

For two way binding:

event.target .value

For checkbox value:

Event.target.checked

For same type of variable : Distinguish by using :

event.target.name

**IN HTML wherever is binding:**

No need to use ! mark       ::::: because one way binding

No need to use "" mark       ::::: because string formation is not allowed.

No need to use v or c letter   ::: because lwc engine is smart enough

For variable or controller same format {variable\_name}{controller\_name} →**simple(get )or event(not needed)**

---

**Directive:**

Iterator vs for each:

Iterator mainly used to get to know first and last element value directly

**For Iteration:**

```
<template for:each={array_list} for:item="item"  
for:index="index">  
    <p key={item}>{index} 'th' index {item} value</p>  
  
</template>  
array_list = ['12', '3'];
```

**Example2:**

```
<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'
    }
  ];
}
```

**For Conditional:**

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

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

```
  areDetailsVisible = false;
}
```

**HTML:**

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

**Javascript:**

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

---

## Styling :

No need to import force:slds already import in dom  
No need to link or attach style sheet already in bundle  
No need to use this keyword

Just use an already defined class or define your own class with corresponding style.

- 1) Inline styling `<p style="color:red">Hello Query selector</p>`
- 2) External styling ( element tag, class , pseudo tag)

```
p{  
    font-size: 30px;  
}  
.Hello{  
    border: 2px solid black;  
}  
.Hello:hover{  
    background-color: brown;  
}
```

- 3) Lightning design system (e.g: for multiple classes `class="class1 class2"`)
- 4) Design token ( color, font family) (`var(--lwc-name)`)
- 5) Shared CSS lib. Create separate component with only `css(@import 'c/cssLibrary')`
- 6) Dynamic CSS: getter method with tactics ``$ {this.}``
- 7) CSS across shadow dom (`rerendercallback(){}`)

Question:

Why no option to `extends="force:slds"` because to make sure we comply with salesforce design  
Use standard

---

## JavaScript

```
import { LightningElement } from 'lwc';  
export default class Index extends LightningElement {  
}
```

Lwc is the module/ framework  
`{ LightningElement }` is the class  
Import is used to import the Lightning Element class

Export default is to make available  
Extending LightningElement is to inherit the class.

---

### **Variable Declaration in client controller:**

Global declaration / parameterized declaration: **Without any keyword**

Local declaration: **With var keyword**

Global to local use / parameterized to local use: **With this keyword**

### **DataType:**

Primitive and non Primitive data type already discussed in Advance skills of javascript

### **Function declaration:**

```
function_name(){}
```

Function calling:

```
this.function_name();
```

### **Example:**

```
import { LightningElement } from 'lwc';
export default class Index extends LightningElement {
    handleButton () {
        this.handleButton2(5, 6);
    }
    handleButton2(x, y) {
        var z = this.x + this.y;
        alert("ooohh button is working" );
    }
}
```

In normal HTML framework :: we call

```
onclick = ="myfunction()"
```

```
Function myfunction() {}
```

In Aura framework we work ::

```
onclick ="{!c.myfunction}"
```

```
Function : myfunction(){}  
}
```

In LWC framework: it work :: onclick ={myfunction}

```
myfunction(){
}
```

**No !, Not in string, Not with c**

---

## Equivalence in between lightning component

# Equivalence of components



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

lightning - namespace

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

# Equivalence of components



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

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

---

**Basic of LWC Completed**

---

---

## ADVANCE CONCEPT OF LWC(LIGHTNING WEB COMPONENT)

---

### How to embed other components in lightning web components?

In kebab case

```
<c-lightning-component />
```

Replace all capital letter with small case and prefixed by hyphen

E.g: sampleDemoLWC

```
<c-sample-demo-l-w-c><c-sample-demo-l-w-c>  
vF2LWC_Journey2  
<c-v-f2-l-w-c _journey2></c-v-f2-l-w-c _journey2>
```

Shadow dom is the dom which does not allow css, query selector property to overwrite from parent to child vice versa

### HTML attribute finder in JavaScript:

```
this.template.querySelector()  
this.template.querySelectorAll()
```

### Don't use id in selector.

#### Example:

```
<lightning-button label="i am Button2" value='This is great'  
onclick={justTocheck}></lightning-button>  
<div class="HI" >  
<p>Hello Query selector</p>  
</div>
```

```
justTocheck() {  
    var querySelector = this.template.querySelector('.HI');  
    querySelector.style.border = "1px solid red";  
    querySelector.innerHTML ="This is inner HTML tag";  
}
```

## LWC LifeCycle:

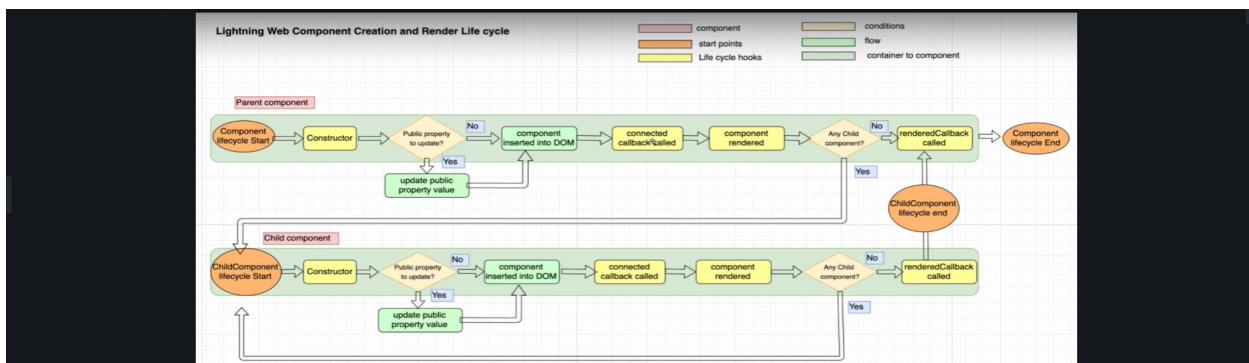
### 1) Mounting Phase

constructor(), connectedCallback(), render(), renderedCallback()

### 2) UnMounting Phase

disconnectedCallback()

### 3) Error Phase : errorCallback()



**Note: rendered callBack only gets called once all child component rendering is done.**

#### Constructor:

##### Example:

In Js File: constructor(){ super(); }

#### Connected Callback:

Mainly use for fetching up the data, setup cache, listening of event

In JS file: connectedCallback(){ }

#### Render:

(very power full method ) (call automatically when anything change in component)

- 1) Can use in advance version of directive ( if condition ) (Sign UP/ Sign In )
- 2) Back button/ previous button and reset button functionality can be done here

##### Example: return default html

render(){

Return this.selected btn === 'SignUp' ? signupTemplate:

    This.selectedbtn === 'SignIn' ? signinTemplate: renderTemplate

}

#### rendered Callback:

This method re render when expression used in template are reevaluate  
( anything change in component cause this to call)

**Don't use in wire adapter and update property of component**

##### In js File:

renderedCallback(){}

### **UnMounting Phase:**

```
disconnectedCallback(){}
```

### **Where do we use this?**

Whenever any component removed from dom or  
window.removeEventListener() ( or window event)

### **Error Phase : any error occur in component mounting and unmounting phase**

```
errorCallback(err, stack{})
```

e.g: throw new error () ;

### **Summary:**

LifeCycle:

- 1) Constructor : main usage is to establish the connection (Super(), No return)
  - 2) Connected Callback: main use is assign any property, any dom related changes
  - 3) renderCallback: after rendering of component is done ( ASYNC Operation )
  - 4) disconnectedCallback: to remove any kind of connection / dom construction / event firing
- 

### **Communication in between component:-**

#### **Normal:**

#### **Reusability way**

- 1) attribute
- 2) method

#### **Import and export way:-**

```
Import {method_name} from 'c/file_name';
```

@api: public property

----->

Import and export : Public property

-----<

### **Communication in between component if hierarchy :-**

#### **Event:**

```
Constructor(){
  this.template.addEventListener('eventName', action_method_name.bind(this));
}
```

Also defined bubbles: true while dispatching

### **If separated component, not in hierarchy ,**

how will you communicate

#### **Hubsub**

#### **Lightning messaging services**

## Event Handling: ( Register / Defining / Handle )

- 1) Create an event and then Register the event at Source level
- 2) Create the action method and define the action method
- 3) Set the value of event attribute and fire the event
- 4) Capture the event (to capture <aura:handler>) and parse the value of attribute

Parent to child : Capture phase

Child to Parent : bubble phase

Sibling component communication : capture phase first should go to the top and then navigate  
Lightning Messaging services: (communication in between VF, AURA, LWC)

## Component communication:

- 1) Reusability ( component referencing ) ( api decorator)
- 2) Import and export data passing.
- 3) Event ( by using reusability concept, by using hook, by using action method )

### @api :: Accessiblity of variables

Import decorator: import {api} from 'lwc';

Use decorator: @api

### Difference in between @api and export

Api is used to make properties of a action method public so that it can be use in reusable

Pass the value in that decorator-----> Api decorator make public

While module is to use the value import -----export

### Primitive Data Passing:

Parent

```
<template>
<lightning-card title="I am parent component">
    <c-child-component
        capturing = 'Hello This is premitive data passing'
    ></c-child-component>
</lightning-card>
</template>
```

Child :

```
import { api, LightningElement } from 'lwc';
export default class ChildComponent extends LightningElement {
    @api capturing
}
```

## Non Primitive Data Passing:

Parent HTML:

```
<template>
    <lightning-card title="I am parent component">
        <lightning-button label="i am Button2" value='This is
great' onclick={function_name}></lightning-button>
        <c-child-component
            capturing = {obj_name}
        ></c-child-component>
        {pre_data_pass}
    </lightning-card>
</template>
```

Parent JS:

```
obj_name= [{"name":"shubham", age:25}, {"name":"dhanuka",
age:25}]
function_name() {
    console.log(this.obj_name)
}
```

Child HTML;

```
<template>
    <lightning-card title="I am Child component">
        <template for:each={capturing} for:item="item">
            <p key={item.name}>This is data captured from parent
{item.name}</p>
            <p key={item.name}>This is data captured from parent
{item.age}</p>
        </template>
    </lightning-card>
</template>
```

CHILD JS

```
export default class ChildComponent extends LightningElement {
    @api capturing
}
```

Till now we have seen attribute using api decorator:

**Method calling in between component using api decorator:**

For that in parent we need reference:

**Example:**

```
this.template.querySelector('c-child-component').method_name();
```

---

**Import export exposed**

```
Import component: import { pre_data_pass } from 'c/parentComponet';
Export export const pre_data_pass = 'Hello This is premitive data passing';
```

**Example: is pending because it is not working**

It is not working because you are passing value from parent to child

Rendering does not work in this way because first all child components get rendered then the parent. So while the child component is getting rendered it is taking undefined value.

**Example:**

```
//parent component
import { pre_data_pass } from 'c/childComponent';
console.log(pre_data_pass);
//child component
export const pre_data_pass = 'Hello This is premitive data';
console.log(pre_data_pass);
```

---

**Module import and export // Util Class Make available component/ class/ variables to use**

**Export attribute and function**

Export const name = "shubham"

Export function getName(){}

**//export together**

```
export{const name ,getName}
```

**//export with alias name**

```
export{const name as cn, getName}
```

**Import attribute and function**

Import ModuleName from Path/file\_name

Import {v1, v2, v3} from path/file\_name

Import \* as utils from path/fileName                   ::: utils is alias name

**Export default**

---

**Event:****1) Standard Event : Browser event****With default value: Parent HTML**

```
<template>
    <lightning-card title="I am parent component">
        <lightning-button label="i am Button2" value='This is great'
            onclick={function_name}></lightning-button>
        <c-child-component capturing={value}></c-child-component>
            {pre_data_pass}
    </lightning-card>
</template>
```

**Parent js**

```
export default class ParentComponet extends LightningElement {
    obj_name=[{ "name":"shubham", age:25}, {"name":"dhanuka", age:25}]
    value= [{ "name":"d", age:25}]

    function_name() { this.value= this.obj_name; }

}
```

**With user creating / event value: (Dynamic value binding )**

```
<lightning-input type="text" label ='enter something'
    value={value1} onkeyup={function_name}></lightning-input>
<c-child-component capturing = {value} value1 = {value1} >
</c-child-component>

value1 = 'kuch nahi';
obj_name=[{ "name":"shubham", age:25}, {"name":"dhanuka", age:25}]

function_name(event){
    //console.log(this.obj_name)
    this.value= this.obj_name;
    this.value1 = event.target.value;
}
```

## 2) CUSTOM EVENT: (Register/ Define/ fire )

### Standard JavaScript way

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

**Declarative way : do with event referencing**

**programmatic way : add event listener**

### In aura

- 1) **Create an event** and then **Register the event at Source level**
- 2) **Create the action method** and **define the action method**
- 3) **Set the value of event attribute** and **fire the event (event.fire() )**
- 4) **Capture the event** (to capture <aura:handler>) and **parse the value of attribute**

Define component extensible = "true", Define component extends="c.base"

#### 1) 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");
E.g: Var comevent = $A.get("e.c.eventName");
comevent.setParams("searchKey":event.target.value");
comevent.fire();
```

#### 3) Render will get event and perform the action

```
<aura:handler name="init" value="{ !this}" action=" {!c.init}" />
(Constructor/hook) value={!this} --> passing the current information to controller
<aura:handler name="Event_name" event="file_name_where_event_define"
action=" {!c.do}" />
```

#### 4) Now in that controller, to get the value param of the event:

```
Var x = event.getParams(searchKey);
```

### **More information about init handler:**

**Init event:**

Init event is the constructor of the event which will get fired when construction of the child component gets started.

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

So basically when a child component or second component construction gets started, init events get called and init events we define an action method that also gets executed and in that action method we do data binding.

NOTE: init action method → render -> after render -> rerender -> un render and so on.

Other than the init event, change also helps if value changes...

```
<aura:handler name="change" value="" action="" />
```

Whenever the value of this changes from default action method will get executed.

#### **Note:**

For standard level event 3 phases are already taken care so we need to focus on 4th point Structure is:

```
<aura:handler name="" value="" action="" />
```

For customize event:

```
<aura:handler name="low priority" event="c:firstEvent" action ="{!c.doEvent}"/>
```

---

#### **Note:**

##### **First Point:**

When event is capturing, action method is calling then use action keyword else how will u call controller:::

Action

Generally we use events to call an action method. But in event we need to use action keyword

##### **Second Point:**

##### **When value is define**

Event.target.value

##### **When parameter needs to taken**

event.getParam

```
var a = event.getParam("firstEventAttribute");
```

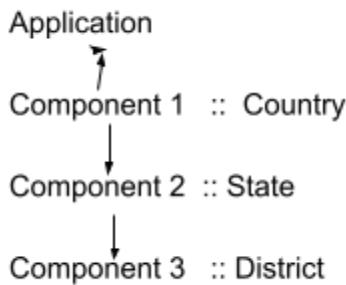
**Please Note that while taking attribute no v, no params**

##### **Third Point:**

If we don't define phase then it will be bubble phase only by default, to convert use Phase=""

---

Example:::



Now some emergency occurs :::

For low priority : communication on district level

For high priority : communication on state level

For Ultra priority : communication on country level

### **Code: Practical:::**

#### **Country level code:**

```
<aura:component>
    <h1>This is Country Component</h1>
    <c:State />
</aura:component>
```

#### **StateLevel Code:**

```
<aura:component>
    <h1>This is State Component</h1>
    <c:District />
    <aura:handler name="lowpriority" value="{!v.firstEventAttribute}" event="c:firstEvent" action
    ="{!c.doEvent}" />
</aura:component>
```

#### **Controller:**

```
{
    myAction : function(component, event, helper) {
    },
    doEvent : function(component, event, helper) {
        var a = event.getParam("firstEventAttribute");
        //console.log(a);
        alert(a);
    }
})
```

### District level code:

```
<aura:component>
    <h1>This is District Component</h1>
    <aura:registerEvent name="lowpriority" type="c:firstEvent" />
    <lightning:input type="button" value="fire the event" onclick="{!!c.dofire}" />
</aura:component>
```

### Controller :

```
{
    myAction : function(component, event, helper) {

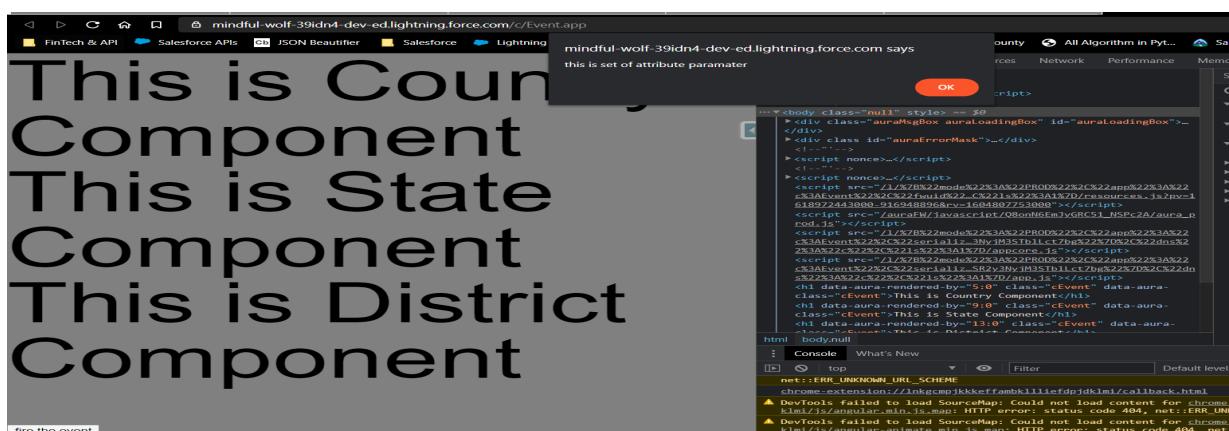
        },
        dofire: function(component, event, helper) {
            //alert("check");
            var captureEvent = component.getEvent("lowpriority");
            captureEvent.setParams({"firstEventAttribute":"this is set of attribute parameter"});
            captureEvent.fire();
        }
}
```

### Application

```
<aura:application>
    <c:Country />
</aura:application>
```

### Event:

```
<aura:event type="COMPONENT" description="Event template" >
<aura:attribute name ="firstEventAttribute" type="String" default="this is default"/>
</aura:event>
```



Declarative way : use event referencing  
programmatic way : add event listener

## In LWC

**In lwc it is much simple to create event, with the help of customevent class**

Create an action method and define an action method.

Define name of event and set the value of attribute using custom event class

Fire the event. ( this.dispatch)

Capture the event (to capture use reusable. Hook ) and add addeventListener

### Child Component:

```
<lightning-button label="event calling" value='This is great'  
onclick={function_name}></lightning-button>  
  
function_name(event) {  
    console.log(event.target.value);  
    // here value assignment is happening with event  
    // you can also define static value  
    // you can also pass using input variable, dont forget to use  
    this keyword  
  
  
    let eventvar = new CustomEvent("name",  
        {bubbles:true, detail:{value:event.target.value}});  
    console.log(eventvar.detail.value);  
    // fire the event  
    this.dispatchEvent(eventvar);  
}
```

### Parent Component:

**In HTML:**

```
<c-child-component  
    capturing = {value}  
    value1 = {value1}  
    // on is the key word here name is name of event and function2  
    // is name of the function that gets called.  
    // This is one way, second way could be by using hook method to  
    handle the event. renderedCallback(), constructor()  
    onname = {function2}  
></c-child-component>
```

### In javascript: by using reusable item

```
Var ="";

function2 (event) {
    // This is declarative way here value is variable way
    Ideal way is this.var = event.detail;

    console.log(event.detail.value);
    console.log("this is executing");

    //This is programmatic way
    this.template.addEventListener('name', (data) =>{
        alert(data.detail.value);
    })
}

}
```

### In Javascript using hook :

```
renderedCallback() {

    //console.log(event.target.value) ;
    //alert(event.target.value);

    this.template.addEventListener('name', (data) =>{
        console.log(data);
        alert(data.detail.value);
    })
}
```

### Third method : In javascript using action method:

This is pending, need to check

Because inside event u want to call other event that is not possible use value assignment in reusable context and use that variable.

**HTML file :**

```
<c-child-component
    capturing = {value}
    value1 = {value1}
    onname = {function2}
></c-child-component>
{pre_data_pass}
<!-- onname = {function2} -->
<lightning-button label="event call"
onclick={function_name2}></lightning-button>
```

**JS file**

```
Value_event;
function2 (event) {
    this.template.addEventListener('name', (data) =>{
        this.value_event = data.detail.value;
    })
}
function_name2 (event) {
    alert(this.value_event);
}
```

**EventListener:**

addEventListener() :: register an event handler  
removeEventListener() :: remove an event handler

**Hook, Flow, Load component :::**

When component get load  
First parent component constructor, connected rollback, gets called

After that all child related hooks get called , rendering gets done.

After that parent rendering( Rendered Callback) gets started and  
(rendering, after rendering, rerendering, unrendering) done

---

**Example: myEvent: is the name of event, detail: is list of object**

```
<script>
//defining the event and dispatch the event
function myName(){
    let event_fire = new CustomEvent("myEvent",{
        Detail:{ name:"shubham", age : 23 } })
    document.dispatchEvent(event_fire);
}

// capture the event
document.addEventListener("myEvent",function(data){
    document.write(data.detail["name"]);
    document.write(data.detail.name);
})

//calling the action method to fire the event
myName();
</script>
```

**Example 2:**

```
<script>
//defining the event and dispatch the event
function myName(){
    let event_fire = new CustomEvent("myEvent",{
        Detail:[{ name:"shubham", age : 23, type: {name:"i am good",age : 21}},
                {name:"dhanuka", age : 21, type: {name:"dhanuka",age : 21}}]

})
    document.dispatchEvent(event_fire);
}

// capture the event
document.addEventListener("myEvent",function(data){
//document.write(data.detail["name"]);
//document.write(data.detail.name);
    document.write(data.detail[1].type.name);
})

//calling the action method to fire the event
myName();
</script>
```

---

### To overwrite component value/ attribute passing :

Parent component js file :

```
settervalueObject = { "name": "my name is another", age: 25 }
```

Parent component template file:

```
<c-child-component settervalue = {settervalueObject}>
</c-child-component>
```

Child Component js file:

```
userdetail
  @api
  get settervalue(){ return this.userdetails }
  set settervalue(data){
    this.userdetails = {...data, name:"i am your father"}
    console.log(data)
  }
```

Child Component template file:

```
<p>This is data captured from parent {userdetail.name}</p>
<p>This is data captured from parent {userdetail.age}</p>
```

---

### To pass HTML MarkUp from one component to another component:

Using Naming slot

Parent component HTML:

```
<c-child-component >
  <p slot="first">Hello my slot component </p>
  <p slot="second">Hello my slot component2 </p>
</c-child-component>
```

Child Component HTML

```
<template>
  <slot name="first"></slot>
  <br />
  <slot name="second"></slot>
</template>
```

onSlotChange={} function is there in case of advance configuration

---

### To pass styling from one component to another component:

Styling can not pass from one component to another component.

---

**Aura Coexistences:**

Lightning web components can't contain aura components.

Whereas you can embed lightning web components into Aura components.

How to embed lightning web components into an aura component?

Same as we embed other lightning aura component

```
<aura:component> <c:parentComponet></c:parentComponet>
</aura:component>
```

How to pass the data?

Same as we pass with the help of api decorator and event

---

**Lightning Messaging services: Communication b/w cf, aura, lwc**

Communication between VF, Aura, LWC Lightning components

**First Component is used for publishing the message(data)**

**Any other components are used for subscribing/ unSubscribing the data.**

**Steps1:**

Create a folder **messageChannels** under the path **force-app/main/default**

**Steps2:**

Create a file **channel\_name.messageChannel-meta.xml**

**Step3:**

Add the xml definition to newly created file

```
<?xml version= "1.0" encoding= "UTF-8" ?>
<LightningMessageChannel xmlns=
"http://soap.sforce.com/2006/04/metadata" >
    <masterLabel>channelName</masterLabel>
    <isExposed>true</isExposed>
    <description>This is a sample Channel.</description>

    <lightningMessageFields>
        <fieldName>recordId</fieldName>
        <description>This is the data changed</description>
    </lightningMessageFields>
</LightningMessageChannel>
```

Here master label is the channel Name that we created in step 2

Lightning message field is the variable / attribute which will be used to pass the data

**Step 4:**

Update the package.xml file with

Check whether the package.xml file is present or not.

If present there then update

```
<types>
<members>channelName</members>
    <name>LightningMessageChannel</name>
</types>
```

If not then create and update

To create.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>CUSTOM OBJECT API NAME HERE</members>
        <name>CustomObject</name>
    </types>
    <version>46.0</version>
</Package>
```

And remove from **forceignore** file

**To import Lightning messaging services:**

```
import { APPLICATION_SCOPE, MessageContext,
createMessageContext, releaseMessageContext,
    subscribe, unsubscribe } from 'lightning/messageService';

import SAMPLEMC from
"@salesforce/messageChannel/channelName__c";
```

## 1. publish(messageContext, messageChannel, message)

PARAMETER	TYPE	DESCRIPTION
messageContext	object	The MessageContext object provides information about the Lightning web components that are using the Lightning message service. Get this object via the MessageContext wire adapter or via createMessageContext().
messageChannel	object	The message channel object. To import a message channel, use the scoped module @salesforce/messageChannel. To create a message channel in an org, use the LightningMessageChannel metadata type.
message	object	A serializable JSON object containing the message published to subscribers. A message can't contain functions or symbols.

## 2. subscribe(messageContext, messageChannel, listener, subscriberOptions)

PARAMETER	TYPE	DESCRIPTION
messageContext	object	The MessageContext object provides information about the Lightning web components that are using the Lightning message service.
messageChannel	object	To import a message channel, use the scoped module @salesforce/messageChannel. To create a message channel in an org, use the LightningMessageChannel metadata type.
listener	function	A function that handles the message once it is published.

subscriberOptions	object	(Optional) An object that, when set to {scope: APPLICATION_SCOPE}, specifies the ability to receive messages on a message channel from anywhere in the application. Import APPLICATION_SCOPE from lightning/messageService.
-------------------	--------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**APPLICATION\_SCOPE:** Import and pass as the fourth parameter in the subscribe() method (`{scope: APPLICATION_SCOPE}`) to receive messages on a message channel from anywhere in the application. Import APPLICATION\_SCOPE from lightning/messageService.

## 3. unsubscribe()

To clear any previous data, we call the Lightning Message Service's unsubscribe() method and set the subscription variable to null.

## How to embed LMS in VF, Aura, LWC Component?

SampleMessageChannel\_c

More information about lightning messaging services:

<https://developer.salesforce.com/blogs/2019/10/lightning-message-service-developer-preview.html>

Example:

Controller LWC:

```
<template>
    <lightning-button class="MyClass" label="publish" value='This
is great' onclick={handleButton}></lightning-button>
</template>
```

Controller LWC JS:

```
import { LightningElement, wire } from 'lwc';
import { APPLICATION_SCOPE, MessageContext, createMessageContext,
releaseMessageContext,
        publish, subscribe, unsubscribe } from 'lightning/messageService';
import SAMPLEMC from "@salesforce/messageChannel/channelName_c";

export default class LMSLearning extends LightningElement {
    @wire(MessageContext)
    context
    message1 ="This is learning to publish";
    handleButton() {
        const message = {
            recordId: this.message1
        };
        publish(this.context, SAMPLEMC , message);
    }
    disconnectedCallback() {
        releaseMessageContext(this.context);
    }
}
```

Render LWC:

```
<template>
<p> this is {receivedMessage}</p>
</template>
```

Render LWC JS:

```
import { LightningElement, track, wire } from 'lwc';
import { APPLICATION_SCOPE, MessageContext, createMessageContext,
releaseMessageContext,
        subscribe, unsubscribe } from 'lightning/messageService';

import SAMPLEMC from "@salesforce/messageChannel/channelName__c";
export default class LMSLearning2 extends LightningElement {
    @wire(MessageContext)
    context;
    receivedMessage = "empty";
    connectedCallback() { this.subscribeMC(); }
    subscribeMC() {
        this.subscription = subscribe(this.context, SAMPLEMC,
(massage) => {this.handleMessage(massage)}, {scope:
APPLICATION_SCOPE});
    }
    handleMessage(message) {
        console.log("message"+message);
        this.receivedMessage = message ? message.recordId: 'no message
payload';
        console.log("this.receivedMessage" + this.receivedMessage)
    }
}
```

Lightning messaging services:

```
<?xml version= "1.0" encoding= "UTF-8" ?>
<LightningMessageChannel xmlns= "http://soap.sforce.com/2006/04/metadata" >
    <masterLabel>channelName</masterLabel>
    <isExposed>true</isExposed>
    <description>This is a sample LightningChannel</description>
    <lightningMessageFields>
        <fieldName>recordId</fieldName>
        <description>This is the record Id changed</description>
    </lightningMessageFields>
</LightningMessageChannel>
```

Aura Component HTML:

```
<aura:component >
    <aura:attribute name="recordValue" type="String"/>
    <lightning:formattedText value="{ !v.recordValue}" />
    <lightning:messageChannel type="channelName__c"
        onMessage="{!c.handleMessage}"/>
</aura:component>
```

Aura component Controller:

```
({ handleMessage: function(cmp, message, helper) {
    // Read the message argument to get the values in the message
    if (message != null && message.getParam("recordId") != null) {
        console.log(message.getParam("recordId"));
        cmp.set("v.recordValue", message.getParam("recordId").value);
    }
})
```

---

## Basic configuration leaning: Salesforce Resources, Component context and Notification

---

- 1) Image from static resources
  - 2) Using Third party JS
  - 3) Using Third party css
  - 4) Access Content Asset Files
  
  - 5) Internationalism
  - 6) Access Label
  - 7) Permission Check
  - 8) User Information
  
  - 9) Access client form factor
  - 10) Fetch record id and object name
  - 11) Toast Notification
- 

### 1) Static resources usage in LWC:

Upload css, js, image, zip, jar file in salesforce org.

To import

```
Import alias_name from '@salesforce/resourceUrl/resources_name';
```

alias\_name : the name will use in programme

Resources\_name: name of image/ file that are importing

Example:

Upload the image in static resource under setup

HTML file code:

```
<template>
    <lightning-card title="Basic Needful Learning">
        <p> Hello My basic concept learning </p>
        <img src={myImage} alt="my IMage" />
    </lightning-card>
</template>
```

JS File code:

```
import { LightningElement } from 'lwc';
import myImage from '@salesforce/resourceUrl/Image';
export default class BasicLearning extends LightningElement {
    myImage = myImage
}
```

## 2) Using Third party JS

### Import static resource

```
Import alias_name from '@salesforce/resourceUrl/resources_name';
```

### Import method from platform resource loader module

```
Import {loadScript} from 'lightning/platformResourceLoader';
```

### loadScript(reference to the component, fileUrl) : promise

```
renderedCallback(){
```

```
    Promise.all([
        loadScript(this, aliasName+'path')
    ]).then(()=>{
        // perform action
    })
})
```

Example:

HTML:

```
<template>
    <img src={myImage} alt="my Image" />
    <p>Timer {Timer}</p>
</template>
```

JavaScript:

```
import { LightningElement } from 'lwc';
import myImage from '@salesforce/resourceUrl/Image';
import timer from '@salesforce/resourceUrl/Timer';
import {loadScript} from 'lightning/platformResourceLoader';
export default class BasicLearning extends LightningElement {
    Timer="empty";
    isLoaded = false;
    myImage = myImage;
    renderedCallback() {
        if(this.isLoaded) {
            return
        }else{
            Promise.all([loadScript(this, timer)]).then(()=>{
```

```

        this.Timer = moment().subtract(10,
'days').calendar();
    )).catch((error) =>{
    console.error(error);
})isLoading = true;  }  }

```

### 3) Using External Style sheet:

#### Import static resource

Import alias\_name from '@salesforce/resourceUrl/resources\_name';

#### Import method from platform resource loader module

Import {loadStyle} from 'lightning/platformResourceLoader';

#### loadStyle(reference to the component, fileUrl) : promise

Example:

#### HTML File:

```

<template>
<div class="animate__animated animate__fadeInLeft animate__bounce
animate-delay: 20s">
<p> Hello my styling</p>
</div>
<!-- Your code -->
<lightning-card title="Basic Needful Learning">
<p> Hello My basic concept learning </p>
<img src={myImage} alt="my IMage" />
<p>Timer {Timer}</p>
</lightning-card>
</template>

```

#### JS FILE:

```

import { LightningElement } from 'lwc';
import myImage from '@salesforce/resourceUrl/Image';
import timer from '@salesforce/resourceUrl/Timer';
import animate from '@salesforce/resourceUrl/animate';
import {loadScript, loadStyle} from
'lightning/platformResourceLoader';
export default class BasicLearning extends LightningElement {
    Timer="empty";
    isLoading = false;
    myImage = myImage;
}

```

```

renderedCallback() {
    if(this.isLoaded) {
        return
    }else{
        Promise.all([
            loadScript(this, timer),
            loadStyle(this, animate)
        ]).then(()=>{
            this.isLoaded = true;
            this.Timer = moment().subtract(10,
'days').calendar();
        }).catch((error)=>{
            console.error(error);
        })
    }
}
}

```

#### **4) Content Asset File:**

It is used for Community template and Custom app.

Import mycontent from '@salesforce/contentAssetUrl/contentAssestReference';

Example:

HTML CODE:

```
<p><a href={file}>file</a></p>
```

Javascript:

```

import myAsset from
'@salesforce/contentAssetUrl/asset_2018090617';
export default class BasicLearning extends LightningElement {
    file = myAsset; }
```

## 5) Internalization :

What is internalization:

Based on the country if we want to change time/ language / currency it is allow us to do

By default for users it should be set for default currency , default language, default time zone.  
Users can personalize/ customize according to their preference.

Syntax:

Import internationalPropertyName from @salesforce/i18n/internationalizationProperty

INTERNATIONALIZATION PROPERTY			
	DESCRIPTION	SAMPLE VALUE	
lang	Language	en-US	
dir	Direction of text	ltr	
locale	Locale	en-CA	
currency	Currency code	CAD	
firstDayOfWeek	First day of the week	1	
dateTime.shortDateFormat	Short style date format	MM/dd/yyyy	
dateTime.mediumDateFormat	Medium style date format	MM d, yyyy	
dateTime.longDateFormat	Long style date format	MM dd, yyyy	
dateTime.shortDateTimeFormat	Short style datetime format	MM/dd/yyyy h:mm a	
dateTime.mediumDateTimeFormat	Medium style datetime format	MM d, yyyy h:mm:ss a	
dateTime.shortTimeFormat	Short style time format	h:mm a	
dateTime.longTimeFormat	Long style time format	h:mm:ss a	
number.currencyFormat	Currency format	\$, ##0.00	
number.currencySymbol	Currency symbol	\$	
number.decimalSeparator	Decimal separator	.	
number.groupingSeparator	Grouping separator	,	
number.numberFormat	Number format	, ##0.###	
number.percentFormat	Percent format	, ##0.##%	

Example:

JS:

```
import Locale from '@salesforce/i18n/locale';
import currency from '@salesforce/i18n/currency'
```

```
formattedNumber = new Intl.NumberFormat(Locale, {
    style:'currency',
    currency: currency,
    currencyDisplay: 'symbol'}).format(this.number);
```

HTML:

```
{formattedNumber}
```

## **5) Label**

Mainly used for multilingual language or to move your static text to label

Custom labels are custom text values that can be accessed from Apex classes, Visualforce pages, Lightning pages, or Lightning components.

Syntax:

In Apex use the `System.Label.Label_name` syntax.

- In Visualforce, use the `$Label` global variable.
- In Aura components, use the `$Label.c.labelName` syntax for the default namespace or `$Label.namespace.labelName` if your org has a namespace or to access a label in a managed package.
- In Lightning web components, import the label using the `@salesforce/label/namespace.Label_name` syntax.
- In Lightning App Builder component labels and attributes, use the `{ !$Label.customLabelName }` expression.

## **6) Form Factor: using to get information about devices .**

```
import formFactorPropertyName from  
'@salesforce/client/formFactor'
```

## **7) Get information about current user:**

```
import property from '@salesforce/user/property';  
import Id from '@salesforce/user/Id';
```

## **8) Fetch record id and Object name :**

```
@api recordId  
@api objectApiName  
recordId and Object-api-name works only on record page not home page or app page
```

## **9) Check Permission :**

```
import hasPermission from '@salesforce/userPermission/PermissionName';
```

```
import hasPermission from '@salesforce/customPermission/namespace__PermissionName';  
Return true and false
```

## 10) toast notification

```
import { ShowToastEvent } from
'lightning/platformShowToastEvent';

const evt = new ShowToastEvent({
    title: this._title,
    message: this.message,
    variant: this.variant,
}) ;
this.dispatchEvent(evt);
```

**Mode: sticky will remain till user close it**

**It can be success, error , info, warning in any mode of information to display**

**You can also pass url in the message as well**

Example:

```
functionToast() {
    const evt = new ShowToastEvent({
        title: this.title,
        message: '{0}{1}',
        messageData:['salesforce', {
            url: 'www.google.com',
            label: 'click here'
        }],
        variant: this.variant,
    }) ;
    this.dispatchEvent(evt);
}
```

## **Navigation services:**

### **Type:**

Open for creating new record

Open for creating new record with default value

Open Home page with recently list view

Open List view

Open record page ( with view and edit mode )

Navigate to record relationship page

Navigate from one component to another component ( VF, AURA, LWC)

Navigate to different tab

Navigate to chatter

Navigate to a webpage

How to take current page reference

preview file

---

### **Import lightning/navigation module:**

```
1. import { NavigationMixin } from 'lightning/navigation';
```

### **Apply the NavigationMixin function to your component's base class.**

```
1. export default class MyCustomElement extends  
    NavigationMixin(LightningElement) {}
```

### **Create a plain JavaScript PageReference object that defines the page.**

### **To dispatch the navigation request,**

```
NavigationMixin.Navigate(pagereference, replace)
```

```
navigateNext() {  
    this[NavigationMixin.Navigate]({  
        type: 'standard__navItemPage',  
        attributes: {  
            apiName: this.tabName,  
        },  
    });  
}
```

---

**Navigate to Home:**

```
navigate() {  
    this[NavigationMixin.Navigate] ({  
        type: 'standard__namedPage',  
        attributes: {  
            pageName: 'home'  
        },  
    }) ;  
}
```

**2) Navigate to Chatter:**

```
navigatetochatter() {  
    this[NavigationMixin.Navigate] ({  
        type: 'standard__namedPage',  
        attributes: {  
            pageName: 'chatter'  
        },  
    }) ;  
}
```

**3) Navigate to new Record :**

```
navigatetonewrecord() {  
    this[NavigationMixin.Navigate] ({  
        type: 'standard__objectPage',  
        attributes: {  
            objectApiName: 'Contact',  
            actionName: 'new'  
        },  
    }) ;  
}
```

**4) Navigate to new record with default value:**

```
import { encodeDefaultFieldValues } from
'lightning/pageReferenceUtils';
navigateToNewRecordWithValue() {
    const defaultValues = encodeDefaultFieldValues({
        FirstName: 'Morag',
        LastName: 'de Fault',
        LeadSource: 'Other'
    });
    this[NavigationMixin.Navigate]({
        type: 'standard__objectPage',
        attributes: {
            objectApiName: 'Contact',
            actionName: 'new'
        },
        state: {
            defaultFieldValues: defaultValues
        }
    });
}
```

**5) Navigate to list view:**

```
navigateToListView() {
    this[NavigationMixin.Navigate]({
        type: 'standard__objectPage',
        attributes: {
            objectApiName: 'Contact',
            actionName: 'list'
        },
        state: {
            filterName: 'Recent'
        }
    });
}
```

**6) Navigate to file:**

```
navigatetofile() {
    this[NavigationMixin.Navigate] ({
        type: 'standard__objectPage',
        attributes: {
            objectApiName: 'ContentDocument',
            actionName: 'home'
        },
    }) ;
}
```

**7) Navigate to record page with view and edit mode:**

```
navigatetorecordpage() {
    this[NavigationMixin.Navigate] ({
        type: 'standard__recordPage',
        attributes: {
            objectApiName: 'Contact',
            recordId: '003N000001oPmYNIA0',
            actionName: 'view'
        },
    }) ;
}

navigatetorecordpage() {
    this[NavigationMixin.Navigate] ({
        type: 'standard__recordPage',
        attributes: {
            objectApiName: 'Contact',
            recordId: '003N000001oPmYNIA0',
            actionName: 'edit'
        },
    }) ;
}
```

**8) Navigation to tab:**

```
navigationtotab() {  
    this[NavigationMixin.Navigate] ({  
        type: 'standard__navItemPage',  
        attributes: {  
            apiName: 'Test'  
        }  
    }) ;  
}
```

**9) navigation to relationship page:**

```
navigationtorelationshippage() {  
    this[NavigationMixin.Navigate] ({  
        type: 'standard__recordRelationshipPage',  
        attributes: {  
            objectApiName: 'Account',  
            recordId: '001N0000021oEAkIAM',  
            relationshipApiName: 'Contacts',  
            actionName: 'view'  
        }  
    }) ;  
}
```

Note: contacts is the api name

**Navigate to external web page:**

```
navigatetoexternalpage() {  
    this[NavigationMixin.Navigate] ({  
        type: 'standard__webPage',  
        attributes: {  
            url: 'https://www.google.com'  
        }  
    }) ;  
}
```

```
}
```

#### **Navigate to LWC page:**

```
navigatetoLWCPage() {
    var definition = {
        componentDef: 'c:index',
        attributes: {
            recordId: '87887898'
        }
    }
    this[NavigationMixin.Navigate]({
        type: 'standard__webPage',
        attributes: {
            url: 'one/one.app#' + btoa(JSON.stringify(definition))
        }
    });
}
```

#### **Navigate to aura:**

```
navigatetoAura() {
    this[NavigationMixin.Navigate]({
        type: 'standard__component',
        attributes: {
            componentName: 'c__Aura2LwcCommunication'
        }
    });
}
```

#### **Navigate to VF :**

**Page reference:**

```
import { CurrentPageReference } from 'lightning/navigation';
@wire(CurrentPageReference)
pageRef;
currentpage() {
var msg=this.pageRef ? JSON.stringify(this.pageRef,null,2) : '';
    alert(msg);
}
```

PageReference : current page information

State : data that is passing in url after ? q=

PageID :

## Accessing Data from Salesforce: (3 ways )

- 1) Lightning data services
- 2) Lightning data services based component
- 3) Server side controller (@AuraEnabled)

### Lightning data services:

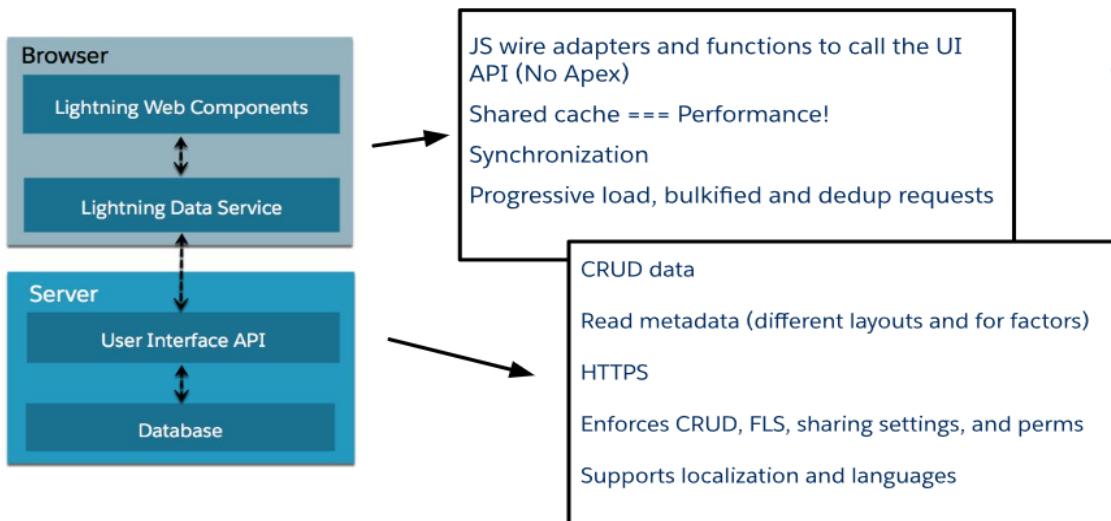
Instead of making a separate call from your lightning components to salesforce do your operation at once in lightning and then communicate to salesforce

- 1) No need to write server side controller
- 2) cache result by default to client
- 3) one call if different component is requesting same data
- 4) respect security model

API:

```
1) uiObjectInfoApi : metadata , picklist  
  
2) uiRecordApi : (getRecord) get record, getFieldValue,  
    getFieldDisplayvalue  
  
    (getRecordUI: layout and metadata information )  
  
3) uiListApi : getList( record and metadata of list view)  
4) uiAppsApi : navigation menu
```

## Lightning Data Service



**Wire :::: will go with catch memory and then go for the server call**

**Syntax:**

```
@wire(adapter, {adapterConfig});
```

propertyorFunction;

**Example:**

```
@wire(getRecord, {recordId:"", fields:["", ""]})  
UserInfoMethod({data,error}){  
    if(data){ }  
    if(error){ }  
}
```

**Note:**

- 1) Get and set parameter server side controller

Note: In setting the parameter use \$ sign to pass the value. '\$recordId'

- 2) Result ( data, error )

( data null and without null)  
(without null then parsing to object and store the value )  
(error then show toast)

**Example with code:**

```
@wire(getRecord, { recordId: '$recordId', fields:  
[ 'Account.Name' ] })  
  
Account({ data, error }) {  
    console.log(data);  
    if (data) {  
        this.name1 = data.fields.Name.value;  
        this.Accountdata = true  
        console.log("name is" + this.name1)  
    } if (error) {  
        console.log(error);  
    }  
}
```

**in markup for Null value condition**

If property use : Account.data

If function use: Account

**in Javascript for setting up the values:**

For function: `this.name1 = data.fields.Name.value;`

For property: `Get name1(){ this.name1 =  
this.Account.data.fields.Name.value;  
}`

---

### **Example: Create:**

```
( operation, object_information, field_information, response in record_id)
Import {createRecord} from '@lightning/uiRecordApi';
```

```
Const field_info ={field_api:data}
```

```
Const record = {APIName: Account, filed_info }
```

```
createRecord(record)
```

### **View:**

**(framework and response )**

**What we need to pass ( operation, record\_id, object\_information, field\_information )**

**(Object to store retrieve information )**

```
import { LightningElement, track, wire } from 'lwc';
```

```
Import {getRecord} from '@lightning/uiRecordApi';
```

```
@track record = this.response.id;
```

```
@track field_Array = ['filed to get info like Account.Name']
```

```
@wire(getRecord, {recordId:$recordId, fields:field_Array})
```

```
accountRecord
```

Now to parse data there is two ways

1) accountRecord.data.fields.Name.value

2) get Method

### **Example with code: HTML:**

```
<template>
    <lightning-card title="Navigation">
        <lightning-layout>
            <lightning-layout-item>
                <p> Hello Lightning data services </p>
                <lightning-input class="slds-p-around_medium" label="Name" name="accountName"
                    onchange={nameChangedHandler}>
                </lightning-input>
                <lightning-input class="slds-p-around_medium" label="Account Number" name="accountNumber"
                    onchange={numberChangedHandler}></lightning-input>
                <lightning-input class="slds-p-around_medium" label="Phone" type="phone" name="accountPhone"
                    onchange={phoneChangedHandler}></lightning-input>
                <lightning-button label="CreateAccount" onclick={createaccount}></lightning-button>
            </lightning-layout-item>
        </lightning-layout>
    </lightning-card>

    <lightning-card title="view Record">
        <template if:true={Account.data}>
            <div class="slds-m-around_medium">
                <p>{name}</p>
            </div>
        </template>
    </lightning-card>
</template>
```

Javascript:

```
import { LightningElement, track, wire, api } from 'lwc';
import { getRecord } from 'lightning/uiRecordApi';
import { createRecord } from 'lightning/uiRecordApi';
export default class LDSLearning extends LightningElement {
    strName;
    strAccountNumber;
    strPhone;
    @api recordId;
    @wire(getRecord, { recordId: '$recordId', fields: ['Account.Name'] })
    Account;
    get name() { return this.Account.data.fields.Name.value; }
    nameChangedHandler(event) {
        this.strName = event.target.value;
    }
    numberChangedHandler(event) {
        this.strAccountNumber = event.target.value;
    }
    phoneChangedHandler(event) {
        this.strPhone = event.target.value;
    }
    createaccount() {
        var fields = {
            'Name': this.strName,
            'AccountNumber': this.strAccountNumber,
            'Phone': this.strPhone
        };
        var objRecordInput = { 'apiName': 'Account', fields };
        createRecord(objRecordInput).then(response => {
            alert('Account created with Id: ' + response.id);
            this.recordId = response.id;
        }).catch(error => {
            alert('Error: ' + JSON.stringify(error));
        });
    }
}
```

## 2) Lightning data services based component

- 1) lightning-record-form
- 2) lightning-record-view-form
- 3) lightning-record-edit-form

FEATURE	LIGHTNING-RECORD-FORM	LIGHTNING-RECORD-VIEW-FORM	LIGHTNING-RECORD-EDIT-FORM
Create Records	✓		✓
Edit Records	✓		✓
View Records	✓	✓	
Read-Only Mode	✓	✓	
Layout Types	✓		
Multi Column Layout	✓		
Custom Layout for Fields		✓	✓
Custom Rendering of Record Data		✓	✓

Example: (**operation, object\_information, field\_information, response in record\_id**)

Create a record:

```
<lightning-record-edit-form object-api-name="Account"
onsuccess={successHandler}>
<lightning-input-field field-name= 'Name'
></lightning-input-field>
<lightning-button class="slds-m-top_small" variant="brand"
type="submit" name="update" label="submit"></lightning-button>
</lightning-record-edit-form>
```

View a record:

```
<lightning-layout-item title="view Record">
<lightning-record-view-form record-id= {recordId}
object-api-name="Account" >
<lightning-output-field field-name="Name" >
</lightning-output-field></lightning-record-view-form>
```

Javascript file:

```
@track recordId;
successHandler(event) {
    this.recordId = event.detail.id;
}
```

### **Lightning-record-form::**

**(object\_information, field\_information, response in record\_id)**

**In operation by default it is in edit form once submit go to view mode { Mode(edit, view) }**

**Layout ( type full, compact)**

**Column (1,2)**

HTML file:

```
<lightning-record-form object-api-name="Account" fields="Name"
record-id= {recordId} onsuccess={successHandler}>
</lightning-record-form>
```

JS file

```
@track recordId;
successHandler(event) { this.recordId = event.detail.id; }
```

1) hard reference :: field-Api define at the form level

2) soft reference :: import field-Api and use reference.

**( one field, Some field, important field using layout type compact, all field using layout type full)**

```
<lightning-record-form object-api-name="Account" record-id=
{recordId} layout-type="Full" onsuccess={successHandler}>
</lightning-record-form>
```

### **Soft reference:**

Import object\_name from '@salesforce/schema/object'; standard object: custom object \_\_c

Import field\_name from '@salesforce/schema/object.field\_name';

**Import relationship\_field '@salesforce/schema/object.relationship.field\_name';**

**Need to check may be work \_\_r.field\_name**

### **Note:**

You can customize the label :

To customize: just first hide default label and define your own label

You can customize the validation rule:

### 3) Server side controller (@AuraEnabled) with wire @auraEnabled(cachable=true)

- 1) Define server side controller
- 2) use wire or second way is imperative way
- 3) use iterator to show data

#### Wire Apex method :

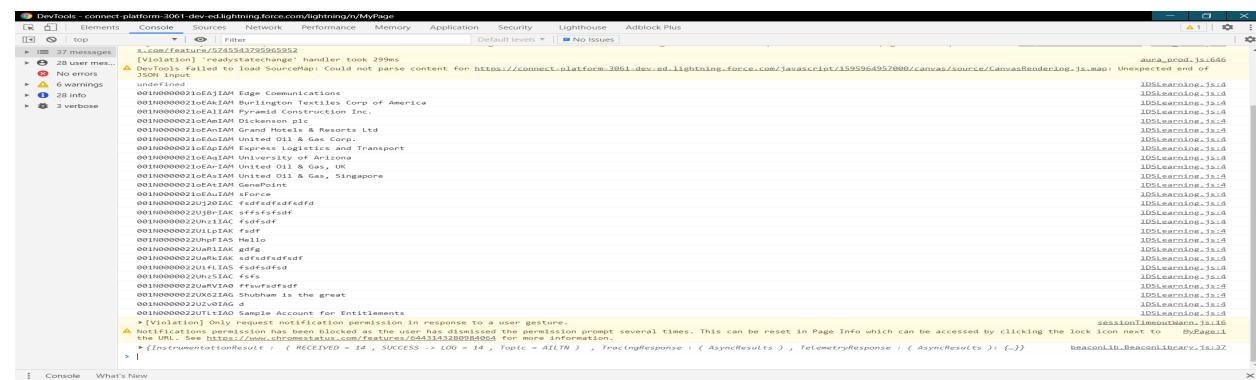
```
array.forEach(function(currentValue, index, arr), thisValue)  
var data = [{name:'shubham', age:44},  
            {'name':'dhanuka', age:14}];
```

```
data.forEach((index)=>{  
    document.write(index.name);  
})
```

For each hold current item value that's it.

Example:

```
@wire(GetAccount)  
Account1({ data, error }){  
    if (data){  
        //console.log(data);  
        //console.log(JSON.stringify(data));  
        data.forEach((index) => {  
            //console.log(index)  
            console.log(index.Id + " " + index.Name)  
        })  
    }  
}
```



Note while importing method don't use {} sign in wire call

**Imperative call method:**

```
handleoutput()  {
    GetAccount().then((result) => {
        console.log("result is " + result);
    }).catch((error) => {
        console.error(error);
    })
}
```

### **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 test10 {  
        // 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 : for writing and searching in file**
  - 3) **Salesforce inspector :: for data**
  - 4) **Advance Code Searcher : for searching file**
  - 5) **Code Coverage : Test Class**
  - 6) **Organizer : for deploying the metadata**
  - 7) **Online individual tools: for css, javascript ( javascript compiler), codepen Lightning component  
Apex anonymous block of code,**
-

### **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  
Diff-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: Application programming interface**

Provide an interface to access an application to execute a programme

**Ultimately, calling at end point class, method with annotation & parameter passing.**

### **Type of API:**

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

### **Operation:**

Request  
Processing  
Response

### **Security:**

- 1) Person security (person who is sending the data should be valid)
- 2) Data Security (data is coming should be secure and valid)
- 3) Site Security (data from where it got originate should be secure)

### **Operation in security:**

- 1) Authentication
- 2) Authorization

## **SITE SECURITY:**

Endpoint url formation: unique\_identifier/folder\_name/class\_name?q=

Uniqueidentifier: server (**path**)

Classname (defined in resources url ) = programme/ folder\_name / **class**

Method: **invoking Action method (get, put, post annotation)**

?Q = query **parameter passing**

## **DATA SECURITY: ( SAML, JSON Web token)**

### **Data Transformation:**

XML and JSON format the data

### **Data Transfer:**

REST, SOAP, XML/ RPC to transfer the data

HTTP/ HTTPS/ SSL ( Encryption, Encoding and scrambling )

## **PERSON SECURITY:**

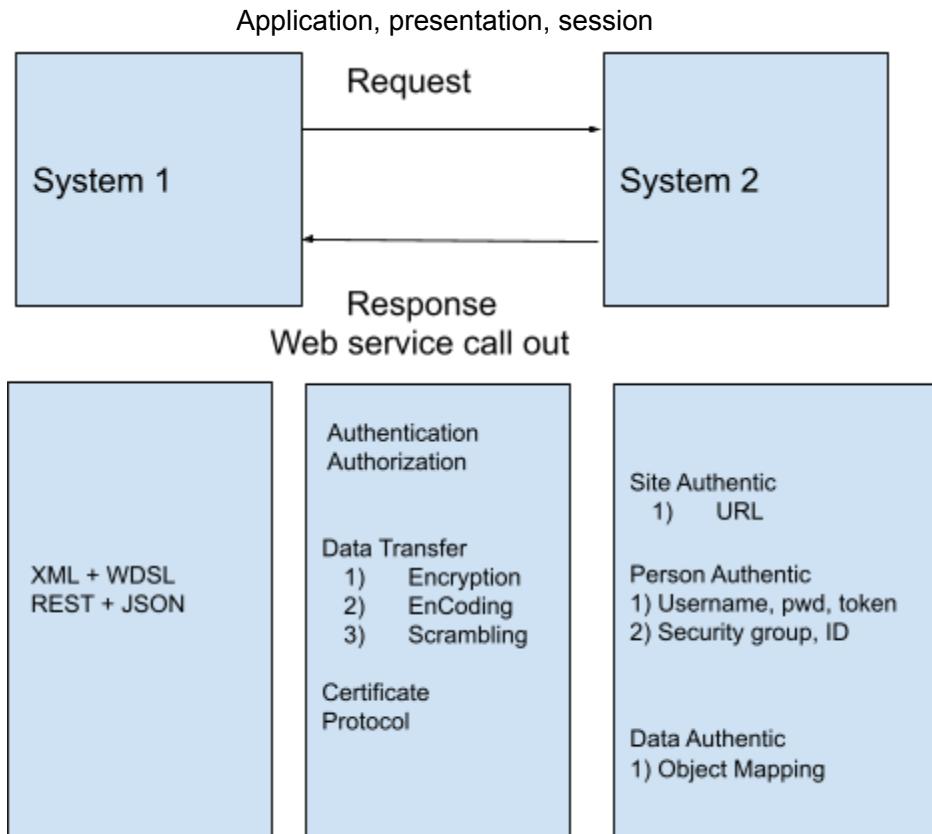
User name , password , security token

Two factor authentication

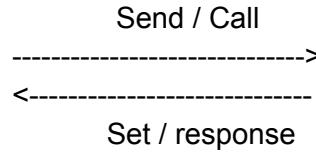
OAuth : Authentication in between two different server (OAuth 1.0, OAuth 2.0)

Open ID

## General Architecture:



- |                           |                                |
|---------------------------|--------------------------------|
| 1) Session setting        | 1) Custom Label                |
| 2) Session management     | 2) Custom URL                  |
| 3) Platform Encryption    | 3) Custom Setting              |
| 4) Certificate & Key Mang | 4) Remote site Setting         |
|                           | 5) Named Convention            |
|                           | 6) Trusted URL for redirect    |
|                           | 7) CSP Trusted sites           |
|                           | 8) Outbound connection setting |



Connected app ( client system call to salesforce system )

Web callout ( Salesforce call to client system )

- 1) Client Controller : CSP Site
- 2) Server Controller : Remote site setting

Live update in between system to sync data ( Salesforce send info to client system )

### **Operation:**

Request  
Processing  
Response

### **Four things in Operation:**

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

### **Request:**

- 1) URI : **unique identification** / subjects/account
- 2) Header: Metadata of request (**Operation (get)**, version(1.1), status, query parameter etc )
- 3) new line: separate from header to body ( **Request Type: JSON/ content-Type**)
- 4) Body: content ( **Request Body** )

### **Processing:**

### **Response:**

- 1) URI : **unique identification** / subjects/account
- 2) Header:Metadata of request (**Operation(get)**, version(1.1), status(200), query parameter, etc )
- 3) new line: separate from header to body ( **Request Type: JSON/ content-Type**)
- 4) Body: content ( **Request Body** ) ( **HTML Page** )

### **Tool Useful in development and simulation :**

Workbench  
SOAP UI  
Postman

### **Troubleshooting:-**

ping/ tracert/ nslookup/ netstat

SOAP → XML

REST → XML and JSON

XML → Markup Type Language

JSON → Object or Array Type Language

### **SOAP Authentication**

URL: <https://login.salesforce.com/services/Soap/c/50.0>

User ID, Password, Security Token

Will give u url and session id.

### **SOAP Authorization :**

URI : https://brave-bear-1zteug-dev-ed.my.salesforce.com/services/Soap/c/50.0/00D5g000004yyFV

Session id will be in XML

Example:

SOAP Request:

1) Authentication

1) URL: <https://login.salesforce.com/services/Soap/c/50.0>

2) Method: POST

3) Header:

Content-type : text/xml

SOAPAction : -

4) Body:

```
<?xml version="1.0" encoding="utf-8" ?>
<env:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
    <env:Body>
        <n1:login xmlns:n1="urn:enterprise.soap.sforce.com">
            <n1:username>dhanuka.shubham@brave-bear-1zteug.com</n1:username>
            <n1:password>Dhanuka12@hlfGkKHBojlwjpfzaaEF4KbV</n1:password>
            </n1:login>
        </env:Body>
    </env:Envelope>
```

In response : got session id and login url

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns="urn:enterprise.soap.sforce.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<loginResponse>
<result>
<metadataServerUrl>https://brave-bear-1zteug-dev-ed.my.salesforce.com/services/Soap/m/50.0/00D5g000004yyFV</met
adataServerUrl>
<passwordExpired>false</passwordExpired>
<sandbox>false</sandbox>
<serverUrl>https://brave-bear-1zteug-dev-ed.my.salesforce.com/services/Soap/c/50.0/00D5g000004yyFV</serverUrl>
<sessionId>00D5g000004yyFV!AQwAQKi.tyhJEL5Z_UoHpwfkJHogiqOrTYNewqlVIQUI6_Inec5iK1ldNYhFFOKJDgi7M8q3
r2N6JBjd27PZ42zUbCAjo1I</sessionId>
<userId>0055g000008Vkb8AAC</userId>
<userInfo>
<accessibilityMode>false</accessibilityMode>
<chatterExternal>false</chatterExternal>
<currencySymbol>$</currencySymbol>
<orgAttachmentFileSizeLimit>26214400</orgAttachmentFileSizeLimit>
<orgDefaultCurrencyIsoCode>USD</orgDefaultCurrencyIsoCode>
<orgDefaultCurrencyLocale>en_US</orgDefaultCurrencyLocale>
<orgDisallowHtmlAttachments>false</orgDisallowHtmlAttachments>
<orgHasPersonAccounts>false</orgHasPersonAccounts>
<organizationId>00D5g000004yyFVEAY</organizationId>
<organizationMultiCurrency>false</organizationMultiCurrency>
<organizationName>TCS</organizationName>
<profileId>00e5g0000031dR9AAI</profileId>
<roleId xsi:nil="true"/>
<sessionSecondsValid>7200</sessionSecondsValid>
<userDefaultCurrencyIsoCode xsi:nil="true"/>
<userEmail>shubhamdhanuka11@gmail.com</userEmail>
<userFullName>Shubham Dhanuka</userFullName>
<userId>0055g000008Vkb8AAC</userId>
<userLanguage>en_US</userLanguage>
<userLocale>en_US</userLocale>
<userName>dhanuka.shubham@brave-bear-1zteug.com</userName>
<userTimeZone>America/Los_Angeles</userTimeZone>
<userType>Standard</userType>
<userUiSkin>Theme3</userUiSkin>
</userInfo>
</result>
</loginResponse>
</soapenv:Body></soapenv:Envelope>
```

## USING Session id and XMI Perform standard controller Action.

### Example:

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <ns1:urn="urn:enterprise.soap.sforce.com">
    <soapenv:Header>
      <urn:SessionHeader>
```

<urn:sessionId>00D5g000004yyFV!AQwAQKi.tyhJEL5Z\_UoHpwfkjHogiqOrTYNewqlVIQUi6\_Inec5iK1ldNYhFFOKJDgi7M8q3r2N6JBjd27PZ42zUbCAj0lI</urn:sessionId>
</urn:SessionHeader>
</soapenv:Header>
<soapenv:Body>
 <urn:query>
 <urn:queryString>SELECT id,Student\_\_Name\_\_c FROM Student\_\_c </urn:queryString>
 </urn:query>
</soapenv:Body>
</soapenv:Envelope>

### Example 2:

```
<urn:create>
  <urn:sObjects xsi:type="urn1:Student__c">
    <Student__Name__c> Sim Aulakh 3 </Student__Name__c>
    <Phone__c> 4567890000 </Phone__c>
    <School__c>a025g000002w1tDAAQ</School__c>
    <Class__Enrolled__c>7 </Class__Enrolled__c>
  </urn:sObjects>
</urn:create>
```

**Exposed class as soap services : (Custom Controller Action)**

Class Should be **global**

Method should be with annotation **webservice static**

Example:

```
global class SoapServices {  
    webservice static List<Account> getAccount(){  
        return [SELECT Id, Name FROM Account];  
    }  
}
```

## Accessing the class

URL: <https://brave-bear-1zteug-dev-ed.my.salesforce.com/services/Soap/class/SapServices>

Method: post

Header: SOAPAction -" content-type: text/xml

Request body:

```
<?xml version="1.0" encoding="utf-8"?>  
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns="http://soap.sforce.com/schemas/class/SoapServices">  
    <soapenv:Header>  
        <SessionHeader>  
            <sessionId>00D5g000004yyFV!AQwAQKi.tyhJEL5Z_UoHpwfkjHogiqOrTYNewqIVIQUi6_lne5iK1IdNYhFFOKJDgi7  
M8q3r2N6JBjd27PZ42zUbCAjol11  
                </sessionId>  
            </SessionHeader>  
        </soapenv:Header>  
        <soapenv:Body>  
            <getAccount>  
                </getAccount>  
            </soapenv:Body>  
    </soapenv:Envelope>
```

Without parameter : just pass the method name

With parameter: pass the parameter in tag

Example:

```
<soapenv:Body>  
    <getStudentById>  
        <stdId>a015g00000KbVZ3AAN</stdId>  
    </getStudentById>  
</soapenv:Body>
```

**Practical of SOAP Authentication / Authorization:**

We need to generate Standard WSDL and then use their methods.

Standard WSDL (Apex class which has XML, Method and Session Related information )

- 1) Enterprise wsdl: for customer : it will gets updated whenever any configuration changes
- 2) Partner wsdl : for partner : it will not gets updated

Custom WSDL ( Apex class on which we define the Custom Operation )

**SOAP API:**

- 1) Authentication XML
- 2) STANDARD WSDL ( XML AND JSON FORM)
- 3) Custom WSDL operation ( XML AND JSON FORM)

Standard WSDL will give us flexibility to set header params etc

## REST Authentication

In Authentication:

- 1) Client id
- 2) Client Security key
- 3) User Name
- 4) Password
- 5) Security token

To get client id and client security key create connected APP

Example:

```
https://login.salesforce.com/services/oauth2/token?grant_type=password&client_id=3MVG9fe4g9fhX0E5aqHBK6Cd  
KcfnwdAG5SkWPu0KN9zP.ymCIU2eX8O6enjseDQE5O3PehOPWE_7iSm7bHtNw&client_secret=55A3E105FB1C8  
A818A7C592DECCDABD45D0E5EA6C06233DE127F46A8C5F3AD1A  
&username=dhanuka.shubham@brave-bear-1zteug.com&password=Dhanuka12@hlfGkKHBojlwjpfzaaEF4KbV
```

**Will give u URL, Token Type, Access Token**

**REST Authorization :**

**Use URL and token type and Access token in Header**

Example:

In Authorization :

URI : <https://brave-bear-1zteug-dev-ed.my.salesforce.com>

Token type and token

Bearer

[00D5g000004yyFV!AQwAQGorjmhqQDr.1A5J4hUVRVz6dFmrpBudkFu4ISbcz4Z8JCpYd6p1i0b4H\\_3T3w95DJMVqsAjy\\_XEvhmmoE0Rm4Aa4BP.](#)

The screenshot shows a Postman API request configuration. The method is set to POST, and the URL is https://brave-bear-1zteug-dev-ed.my.salesforce.com/services/data/v50.0/sobjects/Account. The Headers tab is selected, showing a table with one row: Authorization (Value: Bearer 00D5g000004yyFV!AQwAQGorjmh...). The Body tab is also visible. The Params, Authorization, Tests, and Settings tabs are present but not selected.

**USING** Token type and token perform standard controller actions.

<https://brave-bear-1zteug-dev-ed.my.salesforce.com/services/data/v50.0/sobjects/Account>

**Method : Post**

**Header : Authorization**

**Request body: it can be either XML or JSON**

```
<Account>
<Name>Shubham Is the bosss</Name>
</Account>
```

**Response:**

```
{
  "id": "0015g00000EQIfPAAX",
  "success": true,
  "errors": []
}
```

### **Tool: Workbench**

For retrieve data:

**For Query :**

1) URI:

/query/?q=SELECT+Name+From+Account+WHERE+ShippingCity='San+Francisco'

2) Method: Get

3) Request Tpe: JSON

## Exposed class as rest services : ( Custom Controller )

Class Should be **global** and **@RestResource** should be defined

Method should be with annotation **global static**

**@HttpGet** : retrieve information

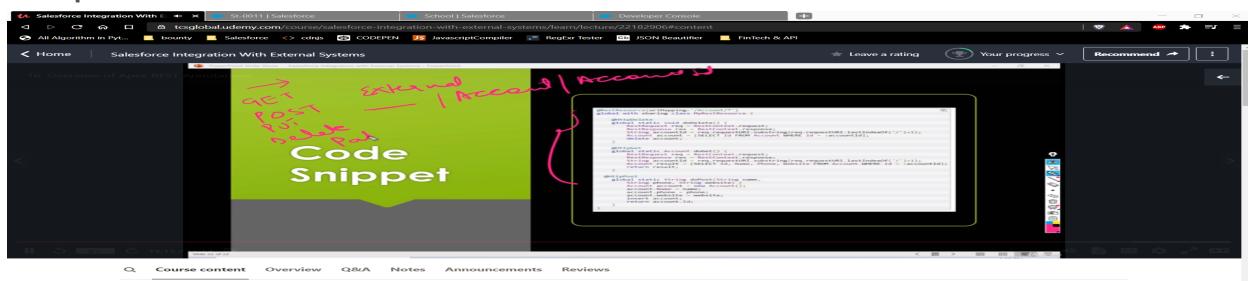
**@HttpDelete** : delete the record

**@HttpPost** : create record : (insert)

**@HttpPut** : create or update the record (upsert)

**@HttpPatch** : update one part of object parameter ( update specific part )

### Example:



```
@RestResource(urlMapping='/Account/*')
global class RestServices {
    @HttpGet
    global static List<Account> getRecord() {
        return [SELECT id, Name FROM Account];
    }
}
```

Accessing in URL

[https://brave-bear-1zteug-dev-ed.my.salesforce.com/services/apexrest/Student\\_\\_c](https://brave-bear-1zteug-dev-ed.my.salesforce.com/services/apexrest/Student__c)

## Practical of REST Authentication / Authorization:

We can use HTTP CLASS Methods

Server Sider Controller to call :

Endpoint URL and Token

Header Method

Data Information ( Request Body )

```
public class callout {  
    public static void hi(){  
        Http h = new Http();  
        HttpRequest hr = new HttpRequest();  
        hr.setEndpoint('https://postman-echo.com/get?foo1=bar1&foo2=bar2');  
        hr.setMethod('GET');
```

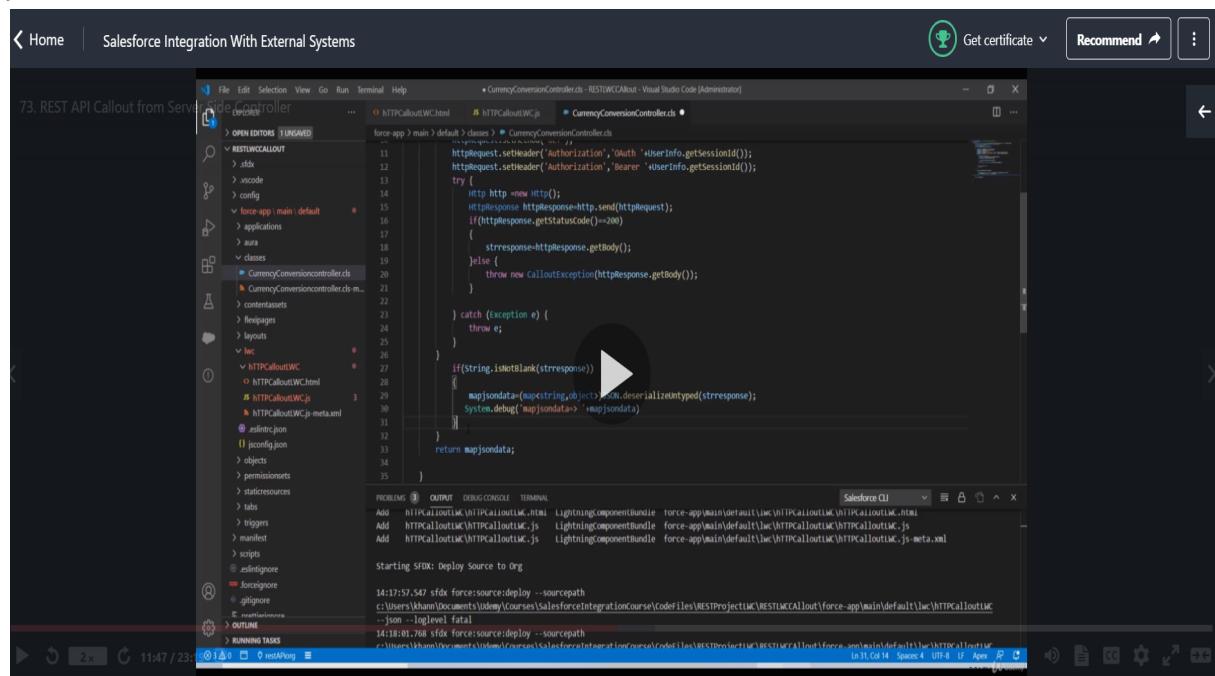
```
        HttpResponse hr1 = new HttpResponse();
```

```
        hr1 = h.send(hr);
```

```
        System.debug('body' + hr1.getBody());
```

```
        Map<String, Object> resultMap = (Map<String, Object>) JSON.deserializeUntyped(hr1.getBody());
```

```
    }  
}
```



```
14:17:57,547 sfdx force:source:deploy --sourcepath  
e:\Users\Wahid\Documents\Video\Course\SalesforceIntegrationCourse\CodeFiles\RESTProject\src\main\default\lwc\HTTPCalloutWC  
--json --logLevel fatal  
14:18:01,788 sfdx force:source:deploy --sourcepath  
e:\Users\Wahid\Documents\Video\Course\SalesforceIntegrationCourse\CodeFiles\RESTProject\src\main\default\lwc\HTTPCalloutWC
```

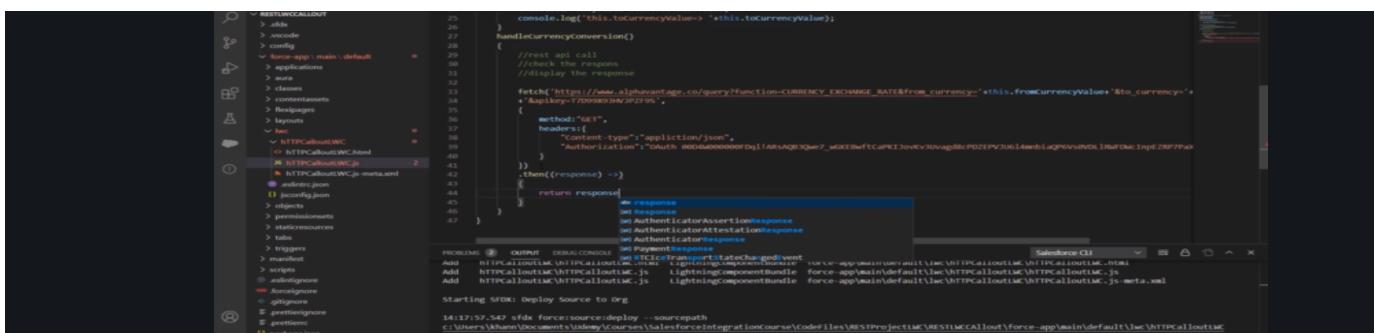
Q Course content Overview Q&A Notes Announcements Reviews



## Client side Controller to call :

In LWC:

```
fetch(url + query).then(response.json)
fetch(){}
Method: "GET"
Header:{}
Content-type: application/json
"Authorization" : "oauth Access token"
}
}
```



In Rest API Call we use:

Http Request calls to set this operation .

- 1) Remote site setting: ( server side controller )  
Salesforce to External System
- 2) CSP Site: ( client side controller )  
External System to Salesforce

---

Data Processing:

**XML + JSON + AJAX ---> Data parsing For call out :**

```
var xhr = new XMLHttpRequest();JSON / AJAX
```

```
RestRequest res = ResContext.response();
```

**Response Mapping, JSON Format, Roasted JSON format**

**Serializing and Deserializing**

Parameter passing/query/ information fetching is URI

**It is done after ?q = "**

```
JSON.stringify
```

```
JSON.parse
```

```
JSON.serialize
```

```
JSON.deserialize
```

## Practical Example :

### Accessing API Data:

Rest api call out : (web callout)

URL and parameter passing and api Key

Client side: define data parsing

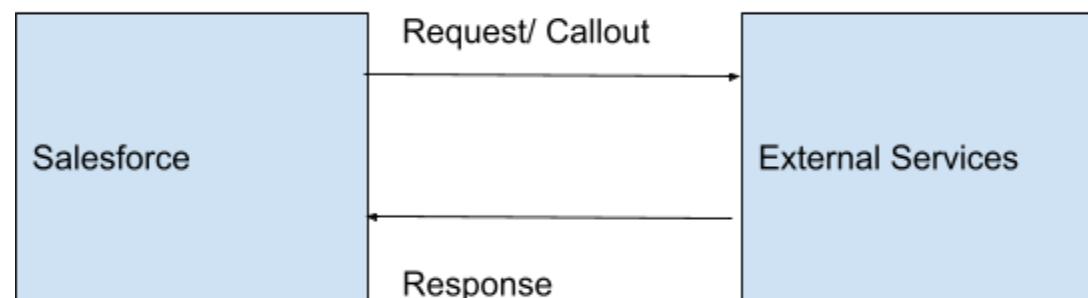
### Accessing Salersforce data on External Web

Use: user id, password with security token, client id, client secret key

URI, Token and token type and access

The screenshot shows a web application window titled "Student Info App". At the top, there are navigation tabs: "Home", "Student Details", and "Schools". Below the tabs, there is a table with four columns: "Student Name", "Student Address", "Phone Number", and "Student Email". The "Student Name" column contains several entries: "Sim Aulakh 1", "Sim Aulakh 2", "Sim Aulakh 3", "Sim Aulakh 4", "Shubham", "Rahul", "Sim Aulakh 3", "Sim Aulakh 4", "Sim Aulakh 2", "Sim Aulakh 1", "Sim Aulakh 3", "Sim Aulakh 4", "Sim Aulakh 3", "Sim Aulakh 4", and "Dhanuka". The "Student Address", "Phone Number", and "Student Email" columns are currently empty. The browser's taskbar at the bottom shows various open tabs and the system date and time as 19-05-2021 09:22.

### Accessing External web data on salesforce :::: HTTP Web Callout



### How to connect two Salesforce orgs through SOAP API?

The screenshot shows a web-based course interface for "Salesforce Integration With External Systems". The main content area displays a diagram of a SOAP API connection between two Salesforce orgs. On the left, a "Parent" object is shown with fields: "GeneratedApexClass", "trigger", "IntegrationObject", "IntegrationObjectVersion", "IntegrationObjectVersion", and "versionedApex". On the right, a "Child" object is shown with fields: "SOAP service", "CreatedContentID", "CreatedContentID", and "parentObject". A connection line between them is labeled "4 parameters". Below the diagram, the course navigation bar includes links for "Course content", "Overview", "Q&A", "Notes", "Announcements", and "Reviews". The browser's taskbar at the bottom shows various open tabs and the system date and time as 19-05-2021 14:29.

**BULK API:**

Create a Job process and then define a batch then close the job then get the result.

**Jobs:** A job specifies the type of operation and data object we're working with.

**Job process:-**

URI: jobs/ingest

Method: **Post**

Header: JSON

Body:

```
{  
  "operation" : "insert",  
  "object" : "Account",  
  "contentType" : "CSV",  
  "lineEnding" : "CRLF"  
}
```

**Add Data or batch:**

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 and Result:**

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

External System Mapping:

- 1) Exposed objects are those which are being created by a third party system.

It comes in an external object

**External Lookup :** lookup relationship in b/w exposed object

**Indirect lookup :** lookup relationship in b/w standard and exposed object

## Streaming API ( Live Notification to sync the data )

Streaming API is your radar. It lets you define events and push notifications to your client app when the events occur. so no need for unnecessary callout to sync the data.

Three ways: PushTopicObject, Record Changes, Platform events etc.

**Streaming API usage bayeux protocol and cometd behind the scenes.**

### USE CASES:

E.g: Emp API: (Showing Notification to Users about edited records or changes in salesforce )

EMP API: ( Message Streaming API)

Real time notification it provides it on the same record someone does an update.

(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
- 8) At the same time fire, the event capturing

### Way of working :

(youtube channel subscription, when video gets uploaded subscriber receive the notification )

- 1) do some changes in record and fire the event
- 2) Subscriber of the event
- 3) Unsubscriber of the event

### Terminology:

The screenshot shows a presentation slide with a green header bar containing the title 'Streaming API terms'. Below the header is a table with two columns: 'Term' and 'Description'. The table includes the following entries:

Term	Description
Event	The creation, update, delete, or undelete of a record. Each event might trigger a notification.
Notification	A message in response to an event. The notification is sent to a channel to which one or more clients are subscribed.
PushTopic	A PushTopic triggers notifications for changes in Salesforce records resulting from a create, update, delete, or undelete operation. A PushTopic notification is based on the criteria that you specify in the PushTopic record and the SOQL query that you define. Only the fields specified in the query are included in the notification. The PushTopic defines a subscription channel.
Channel	A stream of events to which a client can subscribe to receive event notifications.
Event Bus	A conduit in which a publisher sends an event notification. Event subscribers subscribe to a channel in the event bus to receive event notifications. The event bus supports replaying stored event messages.
Platform Event	A Salesforce entity that represents the definition of the custom data that you send in a platform event message. You create a platform event and define its fields in Salesforce. The subscription channel is based on the platform event name.
Change Data Capture Event	Similar to a PushTopic, Change Data Capture triggers notifications for changes in Salesforce records resulting from a create, update, delete, or undelete operation. Like PushTopic, Change Data Capture sends all changed fields of a record and doesn't require you to specify the fields in a query. Also, Change Data Capture sends information about the change in headers.

At the bottom of the slide, there is a navigation bar with links: Q, Course content, Overview, Q&A, Notes, Announcements, and Reviews. The slide is identified as 'Slide 23 of 40'.

PushTopic : Define the field that will cause to fire the event  
Change data event capture: Any things changes in custom object will fire the event  
Notification: notification  
Event : which takes the information  
Platform event: which takes the information  
Channel: Channel ( separate item )  
Event Bus: Cumulative events of all channel

---

### **PushTopic : it is a custom object/ class like attachment.**

Define Criteria to fire the event  
Pushtopic name is used to subscribe to the event.

#### **Example:**

```
PushTopic pushTopic = new PushTopic();
pushTopic.Name = 'StudentUpdates';
pushTopic.Query = 'SELECT Id, Student_Name__c, Phone__c, School__c FROM Student__c';
pushTopic.ApiVersion = 52.0;
pushTopic.NotifyForOperationCreate = true;
pushTopic.NotifyForOperationUpdate = true;
pushTopic.NotifyForOperationUndelete = true;
pushTopic.NotifyForOperationDelete = true;
pushTopic.NotifyForFields = 'Referenced';
insert pushTopic;
```

#### **How to get to know pushtopic**

Select id from pushtopic

#### **Platform Event:**

Setup -> platform event  
You need to define your custom field, you can not use standard fields.  
So event will get fired when something changes on these field  
So what is the use ? how will you populated the data in this field  
It is also sobject/ custom object so will write a trigger and update the value.  
Example: Event\_name\_\_e evt = new Event\_name\_\_e();

#### **Publish platform event:**

- 1) Through apex: eventbus.publish();
- 2) Through API: service/data/v40.0/sobject/event\_name\_\_e

#### **How to get to know platform event**

You can not

#### **Change Data capture:**

Setup -> change data capture

#### **How to get to know change data capture:**

You can not

### **How to Subscribe: Name that are using to subscribe**

Pushtopic:

- 1) Username
- 2) password with security token
- 3) topic/pushtopic\_name

Platform event

- 1) Username
- 2) password with security token
- 3) event/event\_name\_e

ChangeData Capture:

- 1) Username
- 2) Password with security token
- 3) data/object\_changeEvent

---

### **Difference b/w pushtopic and platform event and data capture**

Data capture: Entire Object (Entire object information )

Pushtopic : selected fields on the object ( real object data was getting transferred )

Platform event: custom separated field mapped with standard fields for custom messaging

---

### **FTP procedure:**

- Trigger the event
- Connect to ftp
- Read the file
- Transform or convert the information
- Connect to salesforce
- Insert the data

### **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

---

**Connected API for information transfer:**

- 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)

**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  
Transferring 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

### **General Question :**

- 1) \$A. : enqueueing action
- 2) \$: is used to tell the Dynamic/ latest data  
Value/ standard value/ Standard platform we are getting. `\${}` 2 (e.g \$browser.isDesktop, \$Label )
- 3) \$Label : standard error or message
- 4) CSS:  
Adding or removing the class  
Const class\_info = this.template.querySelector("");  
class\_info.classList.remove('slds-hide')
- 5) JavaScript:  
Array for loop with for each  
UseCase: Reset :
- 6) sObject : is a parent class of all standard and custom object