A project report on

SALESFORCE TECHNOLOGY

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology
In
Computer Science & Engineering

by

ANSH GOEL (20BCE1798)

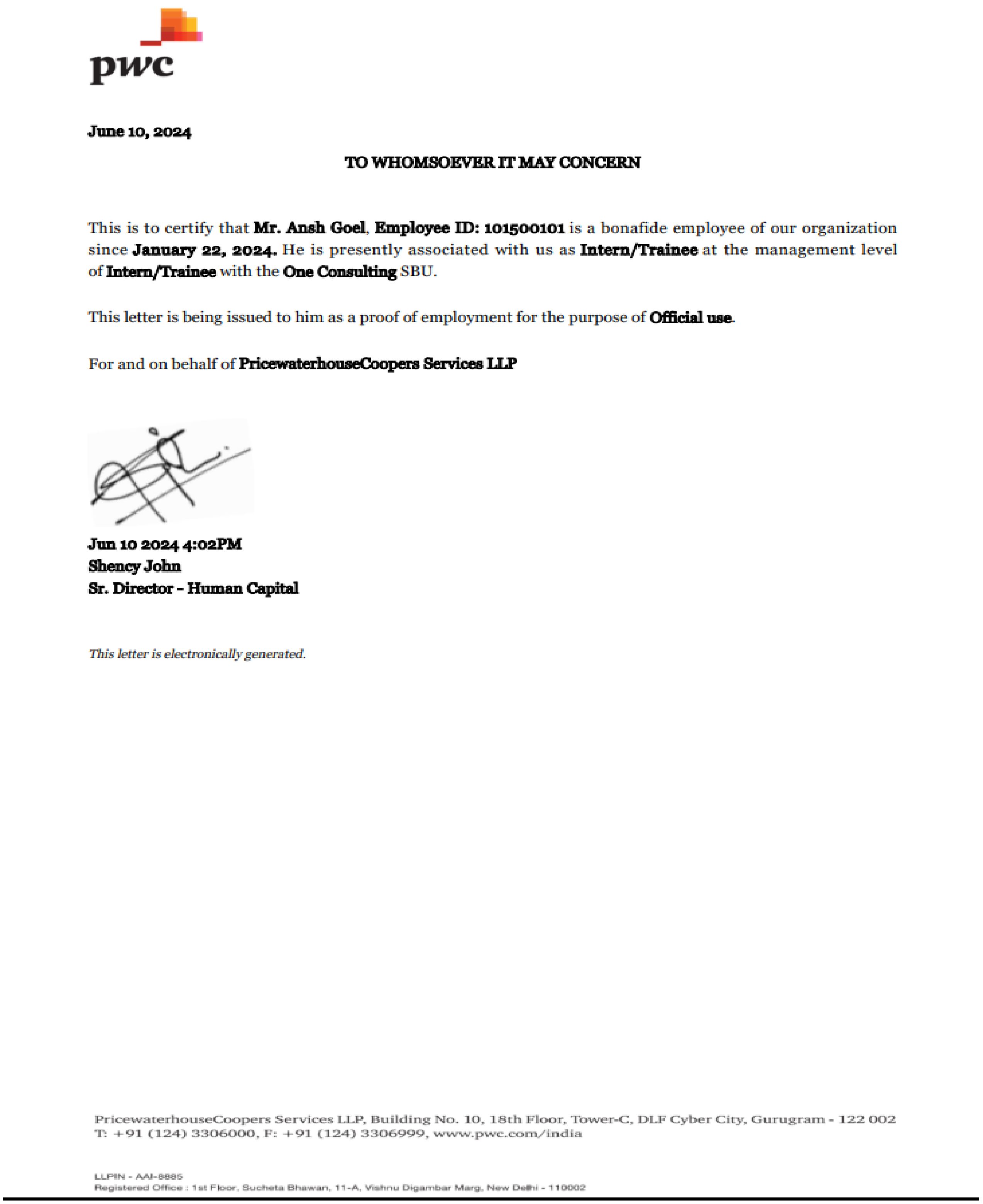
VIT®

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE ENGINEERING (SCOPE)

June, 2024

Internship Undergoing Certificate



June 10, 2024

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Ansh Goel**, **Employee ID: 101500101** is a bonafide employee of our organization since **January 22, 2024.** He is presently associated with us as **Intern/Trainee** at the management level of **Intern/Trainee** with the **One Consulting** SBU.

This letter is being issued to him as a proof of employment for the purpose of **Official use**.

For and on behalf of **PricewaterhouseCoopers Services LLP**

[illegible]

Jun 10 2024 4:02PM
Shency John
Sr. Director - Human Capital

This letter is electronically generated.

PricewaterhouseCoopers Services LLP, Building No. 10, 18th Floor, Tower-C, DLF Cyber City, Gurugram - 122 002
T: +91 (124) 3306000, F: +91 (124) 3306999, www.pwc.com/india

LLPIN - AAI-8885
Registered Office : 1st Floor, Sucheta Bhawan, 11-A, Vishnu Digambar Marg, New Delhi - 110002

<u>DECLARATION</u>

I hereby declare that the thesis entitled "**SALESFORCE TECHNOLOGY**" submitted by me, for the award of the degree *Bachelor of Technology in Computer Science and Engineering* in VIT is a record of bonafide work carried out by me under the supervision of **Mrs. Sobitha Ahila S**.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

[illegible]

Place: Chennai

Date: 15th June, 2024

Signature of the Candidate

ABSTRACT

Salesforce.com, Inc. is a cloud-based software firm situated in San Francisco, California. It focuses on sales, customer service, marketing automation, analytics, and application development via its customer relationship management (CRM) software and apps.

Former Oracle executive Marc Benioff, together with Parker Harris, Dave Moellenhoff, and Frank Dominguez, launched the firm as a software as a service (SaaS) start-up in 1999. Larry Ellison and Halsey Minor provided the firm with its first round of finance.

Salesforce's first annual Dreamforce conference took place in San Francisco in 2003.
The firm raised US$110 million in its first public offering on the New York Stock Exchange in June 2004 under the stock code CRM.
Salesforce developed IdeaExchange in 2006, a platform that connects customers with product managers within the firm.

Salesforce Platform (also known as Force.com) is a platform as a service (PaaS) that allows developers to construct add-on apps for the primary Salesforce.com application. Salesforce.com's infrastructure hosts these third-party apps. Lightning [further explanation needed] and Apex, a proprietary Java-like programming language for Force.com, as well as Visualforce, a framework [31] that includes an XML syntax commonly used to produce HTML, are used to build Force.com applications. Every year, three complete versions of the Force.com platform are released. [33] Because the platform is given as a service to its developers, all of these upgrades are applied to every single development instance.
Lightning Components, a new framework for creating user interfaces, was released in beta in 2015.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Mr. Shakir Iqbal, Senior Director, PwC INDIA, for giving me this opportunity of being a part of this prestigious firm to learn from the experts in the field of Salesforce Technology.

It is with gratitude that I would like to extend my thanks to the visionary leader Dr. G. Viswanathan our Honorable Chancellor, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. G V Selvam Vice Presidents, Dr. Sandhya Pentareddy, Executive Director, Ms. Kadhambari S. Viswanathan, Assistant Vice-President, Dr. V. S. Kanchana Bhaaskaran Vice-Chancellor, Dr. T. Thyagarajan Pro-Vice Chancellor, VIT Chennai and Dr. P. K. Manoharan, Additional Registrar for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dr. Ganesan R, Dean, Dr. Parvathi R, Associate Dean Academics, Dr. Geetha S, Associate Dean Research, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant state, I express ingeniously my whole-hearted thanks to Dr. Nithyanandam P, Head of the Department, B.Tech. CSE and the Project Coordinators for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staffs at Vellore Institute of Technology, Chennai who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date: 15th June, 2024 **Ansh Goel**

Contents

LIST OF FIGURES xi

LIST OF TABLES xi

LIST OF ACRONYMS xii

WHAT IS CRM? 12

Advantages of CRM: 12

WHAT IS SALESFORCE? 13

- Sales Cloud 13
- Service Cloud 13
- Data Cloud 13
- Marketing Cloud 13
- Collaboration Cloud 13
- Analytics Cloud 13
- Custom Cloud including Force.com 13

SALESFORCE ARCHITECTURE 14

DATA MODELING IN SALESFORCE 15

OBJECTS 15

- Standard Objects 15
- Custom Objects 15

RECORDS 16

FIELDS 16

- Standard fields 16
- Custom fields 16

RELATIONSHIPS IN SALESFORCE 17

➔ One to Many Relationships 17

1. Lookup relationships 17

- It is a type of one to many relationship (One parent and n children) 17
- It is a loosely coupled relationship between the child object and the parent object 17

● The lookup relationship field is created on the child object record (Employees) 17
2. **Master-Detail Relationships** 19
Roll-up summary Fields 20
➔ Many to Many Relationships 22
SCHEMA BUILDER 23
RECORD TYPES 24
PERMISSION SETS 25
Salesforce Permission Sets 25
ORGANIZATION-WIDE DEFAULT 26
SALESFORCE ROLE HIERARCHY 28
SALESFORCE ACCOUNTS 29
Important Fields 29
SALESFORCE CONTACTS 30
SALESFORCE LEAD CONVERSION 31
When a lead is converted? 31
SALESFORCE OPPORTUNITIES 32
Important Fields 32
APEX CLASS 34
APEX DATABASE 36
APEX TRIGGERS 37
What are apex triggers? 37
Types of Triggers 38
Trigger context variables 38
APEX TESTING 40
Test method syntax 41
HTML (HYPER TEXT MARKUP LANGUAGE) 45
HTML element 45
HTML Attributes 46
HTML data -* attribute 46
Types of CSS styles 46
CSS Box Model 47
JAVASCRIPT 47
Variables 48

Data Types 48

Spread Operator 49

The javaScript spread operator (...), has multiple uses, as listed below 49

1. Expanding String - This converts a string into an array, whose elements consist of the individual characters present in the string 49

2. Combining Arrays - This includes combining multiple arrays, or adding values to pre-existing arrays. 49

3. Combining Objects - This includes combining multiple objects, or adding values to pre-existing objects. 49

4. Creating a new shallow-copy of objects and arrays 49

Object Destructuring 49

String Methods 49

Object/JSON operations 50

Promise 51

QuerySelector 52

Events 52

LWC FUNDAMENTALS 54

Introduction 54

Benefits of LWC 55

Naming conventions in LWC 55

Steps for App Creation in LWC 56

Local Properties and Data Binding in LWC 56

Methods and two-way Data Binding in LWC 57

@track property in LWC 57

LWC directives in HTML 57

Template looping 58

Styling in LWC 58

Component Lifecycle Hooks in LWC 59

Component Communication in LWC 60

Parent to child 60

Child to Parent 61

Unrelated components 61

COMMUNICATION BETWEEN LWC, AURA COMPONENTS AND VISUALFORCE 61

Lightning Message Services 61

AURA COEXISTENCE 66

SALESFORCE RESOURCE, COMPONENT CONTEXT AND NOTIFICATION 67

1. Fetching images from static resources 67
2. Getting used to third party javascript libraries in LWC 67
3. Getting used to third party CSS libraries in LWC 68
4. Content asset files 69
5. Internationalisation 70
6. Access labels 70
7. Check permissions 71
8. Access clients from factors 71
9. Get information about current users 72
10. Toast notifications 73

LIST OF FIGURES

Figure 1 SALESFORCE CRM……………………………………………………………………14
Figure 2 SALESFORCE ARCHITECTURE………………………………………………...15
Figure 3 LOOKUP RELATIONSHIP…………………………………………………………18
Figure 3.1 MASTER-DETAIL RELATIONSHIP…………………………………………...19
Figure 3.2 MANY TO MANY RELATIONSHIP……………………………………………23
Figure 4 SCHEMA BUILDER…………………………………………………………………24
Figure 5 PERMISSION SETS…………………………………………………………………26
Figure 6 ORGANISATION-WIDE DEFAULT………………………………………………...27
Figure 7 APEX CLASS……………………………………………………………………...35
Figure 8 APEX TESTING…………………………………………………………………...41
Figure 9 WEB TECHNOLOGY……………………………………………………………..45
Figure 10 LIGHTNING WEB COMPONENT…………………………………………………55
Figure 10.1 CREATION IN LWC………………………………………………………………57
Figure 10.2 HOOKS IN LWC…………………………………………………………………..60
Figure 10.3 COMS IN LWC…………………………………………………………………….61

LIST OF TABLES

Table 1 DATA MODELLING IN SALESFORCE……………………………………………..16

LIST OF ACRONYMS

BU – Business Unit
SBU – Sub Business Unit
LoS – Line of Service
PoC – Proof of Concept
B2C – Business to Consumer
B2B – Business to Business
B2B2C – Business to Business to Consumer - Marketplace
API – Application Program Interface
DNS – Domain Name System
PPT – Power Point Presentation
FY – Financial Year
RS – Relationship

SALESFORCE ADMIN

WHAT IS CRM?

CRM stands for Customer Relationship Management. It is a tool or a process that helps businesses to strengthen their relationships with customers and to better understand them.

Advantages of CRM:

- It helps us to find new customers.
- It helps the business to provide better service to its customers.
- It helps to gather all the customer data in one single place instead of it being scattered across excel sheets.
- It helps the sales team and marketing teams to collaborate and work in synergy so that each of them is aware of what the other is doing.
- The efficiency of the sales staff increases as they can close more deals in less time and it simplifies the sales and marketing processes to a great extent.
- It helps businesses to track customer data based on their interactions, and track important metrics and it helps teams collaborate and gather insights from various social media platforms and communicate via emails, phone, etc.
- It ultimately helps increase the revenues and profits of the business as they can handle their customers in a better way.

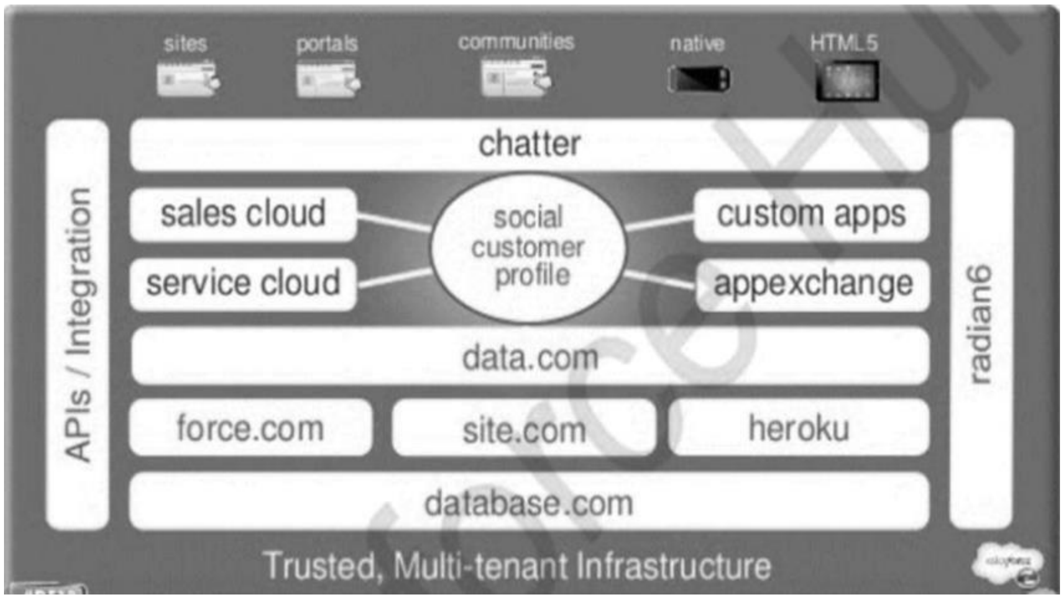
WHAT IS SALESFORCE?

Salesforce is a cloud computing company based out of San Francisco, California. It provides a product by the same name that is 'Salesforce' which is a highly customizable CRM.

With the help of Salesforce, you can run your business from anywhere in the world because it is a cloud CRM where all of the data is stored on the cloud.

It provides some standard features and products to engage with customers, employees, and partners and it also allows to customize and personalize the experience for customers, partners, and employees and easily extends beyond out-of-the-box functionality.Salesforce CRM comprises:

- Sales Cloud
- Service Cloud
- Data Cloud
- Marketing Cloud
- Collaboration Cloud
- Analytics Cloud
- Custom Cloud including Force.com

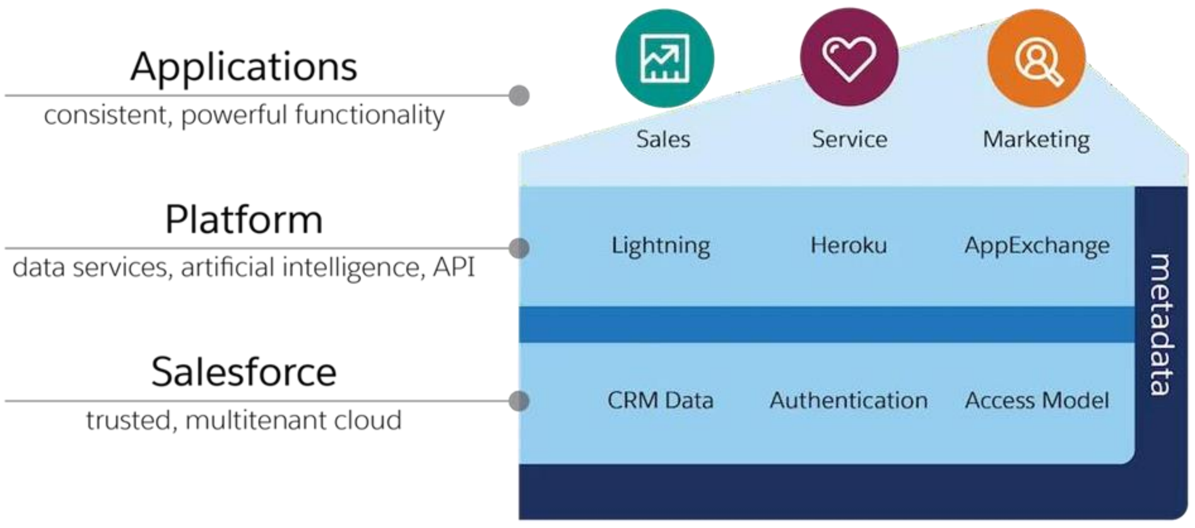


SALESFORCE ARCHITECTURE

Salesforce is based on a Multitenant Architecture. To understand it in simple words, a salesforce is like an apartment building where your company or business has its own space or a flat in the building but you are using the same resources for water, electricity, and building maintenance. You are sharing the resources with other companies in the same building.

Salesforce provides a core set of services to all our customers in the multi tenant cloud. No matter the size of your business, you get access to the same computing power, data storage, and core features. But your data is secure and is isolated from other tenants.

Because of this architecture, any developer can create and upload any application on the salesforce cloud and it can be shared with multiple people or groups. Since multiple people use the same server it is very cost-effective and all the customer data is saved in a single database.



DATA MODELING IN SALESFORCE

A database is used to store information in an organized way.

Relational Database	Salesforce
Data is stored in tables	In salesforce, we use objects to store data that corresponds to a table in a database
Each table consists of several columns of a particular type such as number, text, date or email address, etc.	Each object has several fields that correspond to the columns in a database.
Tables can be related to each other using Primary key and foreign key concepts.	Objects can be related to each other using relationship fields.

OBJECTS

They are used to store a particular type of information in salesforce.

- Standard Objects

These objects are by default given by the salesforce to us. . Example: Accounts, Leads, and Opportunities. They cannot be deleted.

- Custom Objects

These objects are created to store information unique to an organization according to the business needs and requirements. They can be deleted.

RECORDS

It is defined as an instance of an object. It is created when we fill the required fields of that particular object. It is analogous to the rows of data in a table(object).

FIELDS

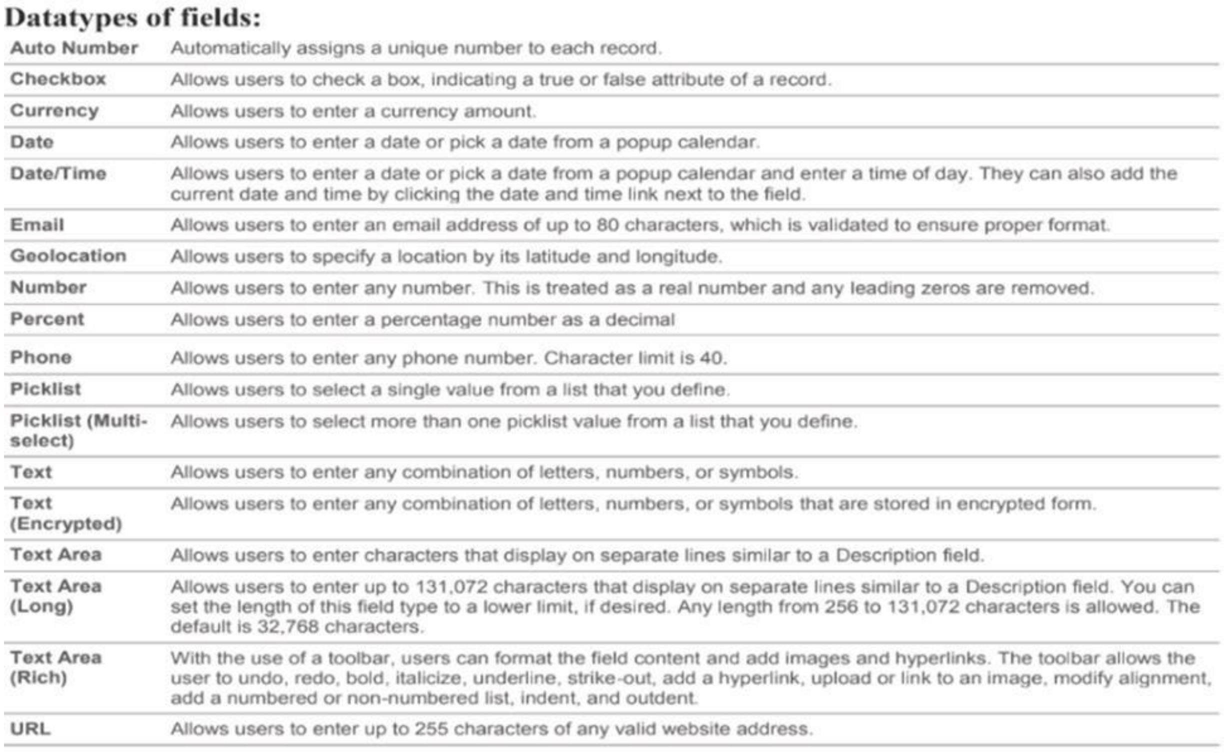
These are the values that are populated in order to create a record for that object.

- Standard fields

For standard objects, These are the fields that are present by default. Eg- Created By, Last Modified By, Owner, and the field created. They are also present when a custom object is created. These fields cannot be deleted or edited and they are always required.

- Custom fields

These fields are added by the admin/developer to meet the business needs of the organization. They can be created on both Standard objects and custom objects.



Datatypes of fields:

Auto Number	Automatically assigns a unique number to each record.
Checkbox	Allows users to check a box, indicating a true or false attribute of a record.
Currency	Allows users to enter a currency amount.
Date	Allows users to enter a date or pick a date from a popup calendar.
Date/Time	Allows users to enter a date or pick a date from a popup calendar and enter a time of day. They can also add the current date and time by clicking the date and time link next to the field.
Email	Allows users to enter an email address of up to 80 characters, which is validated to ensure proper format.
Geolocation	Allows users to specify a location by its latitude and longitude.
Number	Allows users to enter any number. This is treated as a real number and any leading zeros are removed.
Percent	Allows users to enter a percentage number as a decimal
Phone	Allows users to enter any phone number. Character limit is 40.
Picklist	Allows users to select a single value from a list that you define.
Picklist (Multi-select)	Allows users to select more than one picklist value from a list that you define.
Text	Allows users to enter any combination of letters, numbers, or symbols.
Text (Encrypted)	Allows users to enter any combination of letters, numbers, or symbols that are stored in encrypted form.
Text Area	Allows users to enter characters that display on separate lines similar to a Description field.
Text Area (Long)	Allows users to enter up to 131,072 characters that display on separate lines similar to a Description field. You can set the length of this field type to a lower limit, if desired. Any length from 256 to 131,072 characters is allowed. The default is 32,768 characters.
Text Area (Rich)	With the use of a toolbar, users can format the field content and add images and hyperlinks. The toolbar allows the user to undo, redo, bold, italicize, underline, strike-out, add a hyperlink, upload or link to an image, modify alignment, add a numbered or non-numbered list, indent, and outdent.
URL	Allows users to enter up to 255 characters of any valid website address.

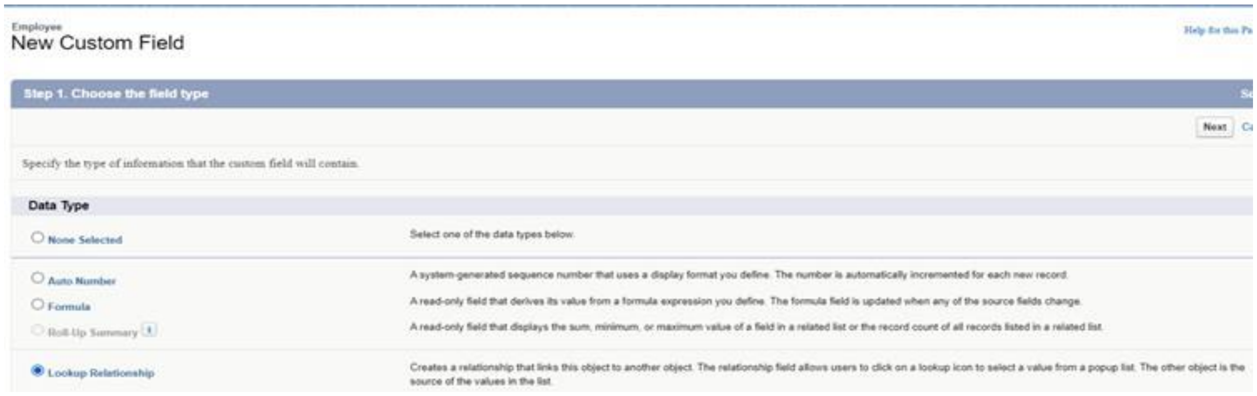
RELATIONSHIPS IN SALESFORCE

➔ One to Many Relationships

1. Lookup relationships
    - It is a type of one to many relationship (One parent and n children).
    - It is a loosely coupled relationship between the child object and the parent object.
    - The lookup relationship field is created on the child object record (Employees).

Here we are creating a lookup relationship between a custom object (Employee) and a standard object (Account). Since one account can have multiple employees so here Employee is the child object and Account is the Parent object.

STEP 1:



Employee
New Custom Field

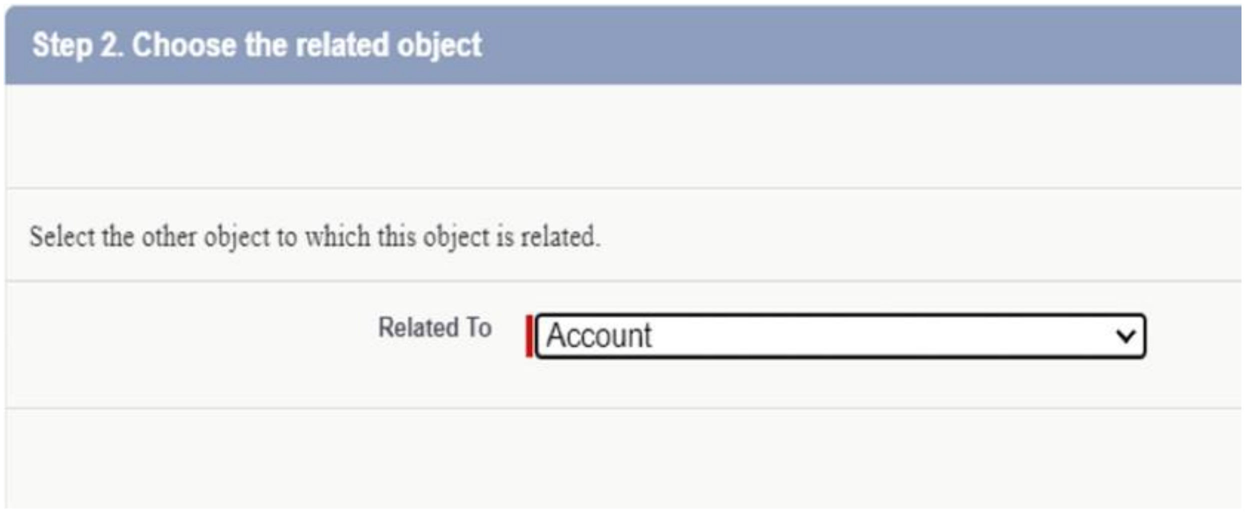
Step 1. Choose the field type

Specify the type of information that the custom field will contain.

Data Type

None Selected	Select one of the data types below.
Auto Number	A system-generated sequence number that uses a display format you define. The number is automatically incremented for each new record.
Formula	A read-only field that derives its value from a formula expression you define. The formula field is updated when any of the source fields change.
Roll-Up Summary	A read-only field that displays the sum, minimum, or maximum value of a field in a related list or the record count of all records listed in a related list.
[✓ Lookup Relationship]	Creates a relationship that links this object to another object. The relationship field allows users to click on a lookup icon to select a value from a popup list. The other object is the source of the values in the list.

STEP 2:

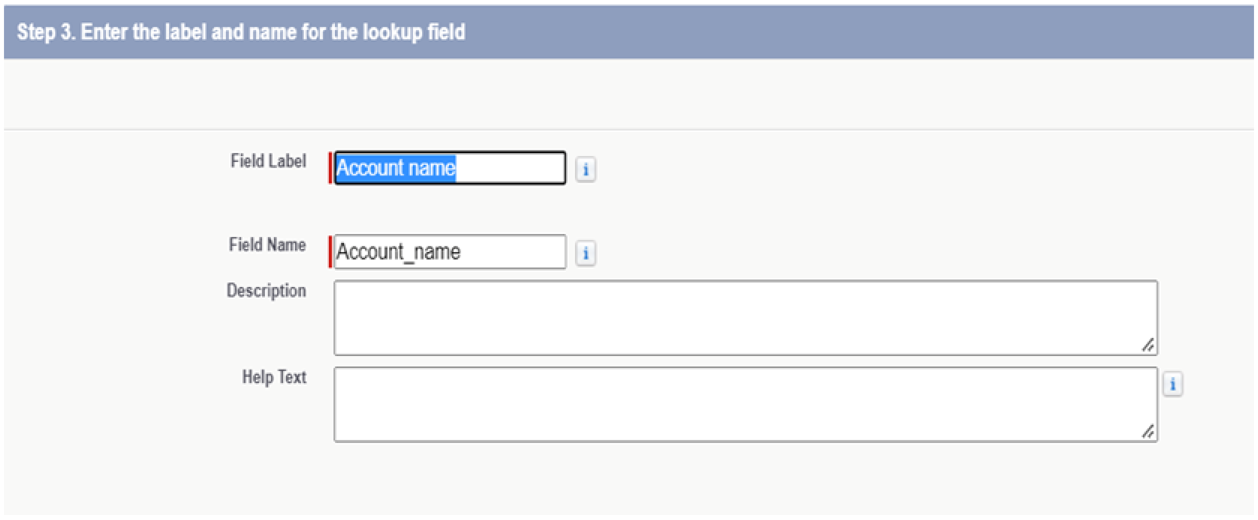


Step 2. Choose the related object

Select the other object to which this object is related.

Related To: Account

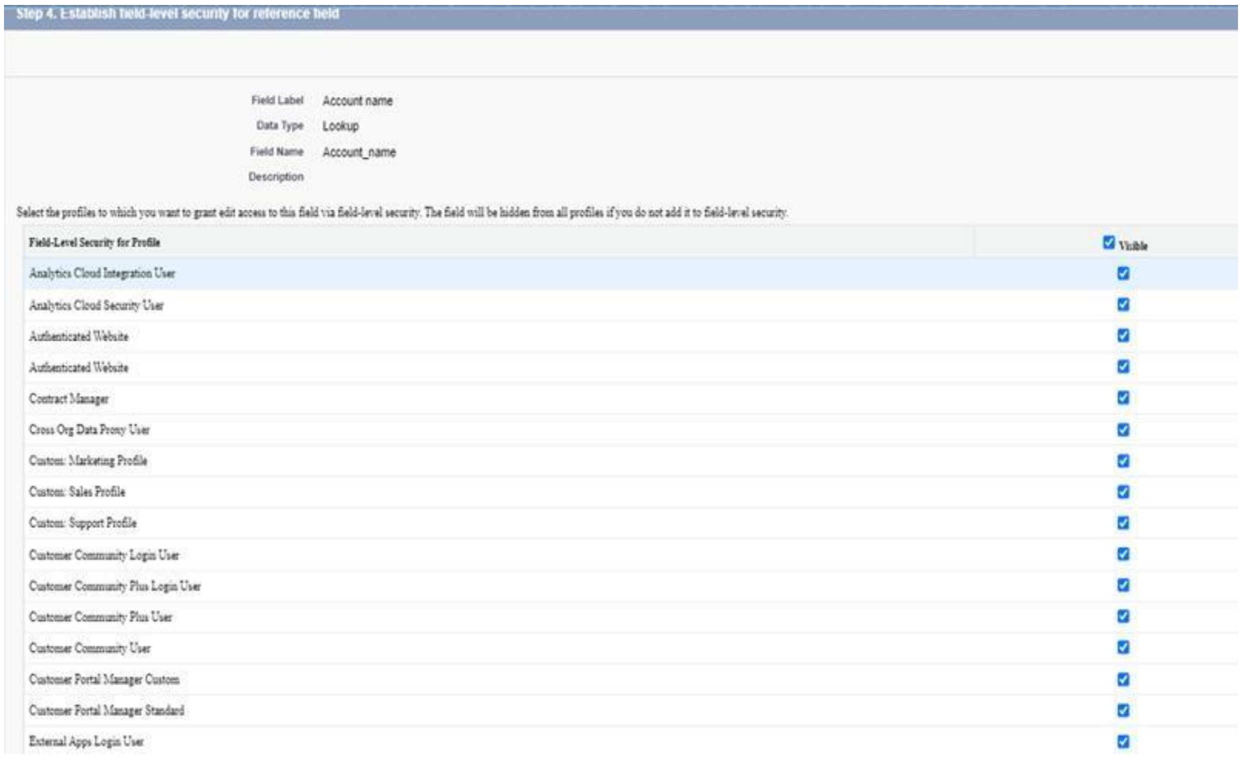
STEP 3:



Step 3. Enter the label and name for the lookup field

Field Label	Account name
Field Name	Account_name
Description	
Help Text	

STEP 4:



Step 4. Establish field-level security for reference field

Field Label	Account name
Data Type	Lookup
Field Name	Account_name
Description	

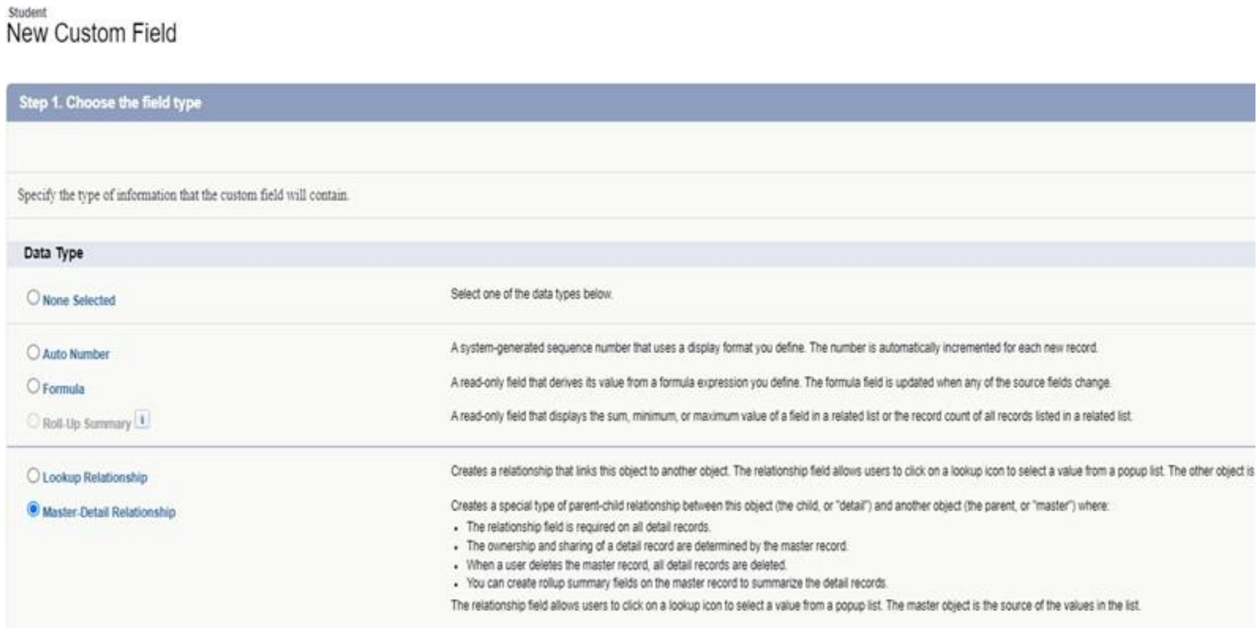
Select the profiles to which you want to grant edit access to this field via field-level security. The field will be hidden from all profiles if you do not add it to field-level security.

Field-Level Security for Profile	✓ Visible
Analytics Cloud Integration User	✓
Analytics Cloud Security User	✓
Authenticated Website	✓
Authenticated Website	✓
Contract Manager	✓
Cross Org Data Proxy User	✓
Custom: Marketing Profile	✓
Custom: Sales Profile	✓
Custom: Support Profile	✓
Customer Community Login User	✓
Customer Community Plus Login User	✓
Customer Community Plus User	✓
Customer Community User	✓
Customer Portal Manager Custom	✓
Customer Portal Manager Standard	✓
External Apps Login User	✓

2. **Master-Detail Relationships**

- It is also a one-to-many relationship between two objects.
- Child records cannot exist without parent records which means that the master-detail relationship field is compulsory to fill on the child object.
- We create a master-detail relationship field on the child object when we want the child record to be deleted if the parent record gets deleted.
- We have two custom objects, Student and Class. One Class can have many Students so here Class is the parent object and student is the child object.

STEP 1



Student
New Custom Field

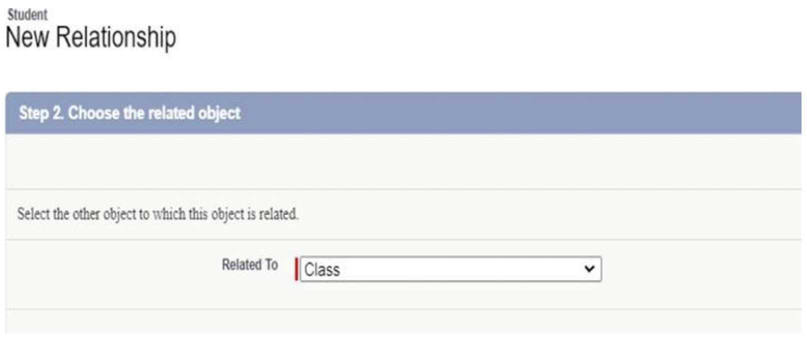
Step 1. Choose the field type

Specify the type of information that the custom field will contain.

Data Type

None Selected	Select one of the data types below.
Auto Number	A system-generated sequence number that uses a display format you define. The number is automatically incremented for each new record.
Formula	A read-only field that derives its value from a formula expression you define. The formula field is updated when any of the source fields change.
Roll-Up Summary	A read-only field that displays the sum, minimum, or maximum value of a field in a related list or the record count of all records listed in a related list.
Lookup Relationship	Creates a relationship that links this object to another object. The relationship field allows users to click on a lookup icon to select a value from a popup list. The other object is
[✓ Master-Detail Relationship]	Creates a special type of parent-child relationship between this object (the child, or "detail") and another object (the parent, or "master") where: • The relationship field is required on all detail records. • The ownership and sharing of a detail record are determined by the master record. • When a user deletes the master record, all detail records are deleted. • You can create rollup summary fields on the master record to summarize the detail records. The relationship field allows users to click on a lookup icon to select a value from a popup list. The master object is the source of the values in the list.

STEP 2



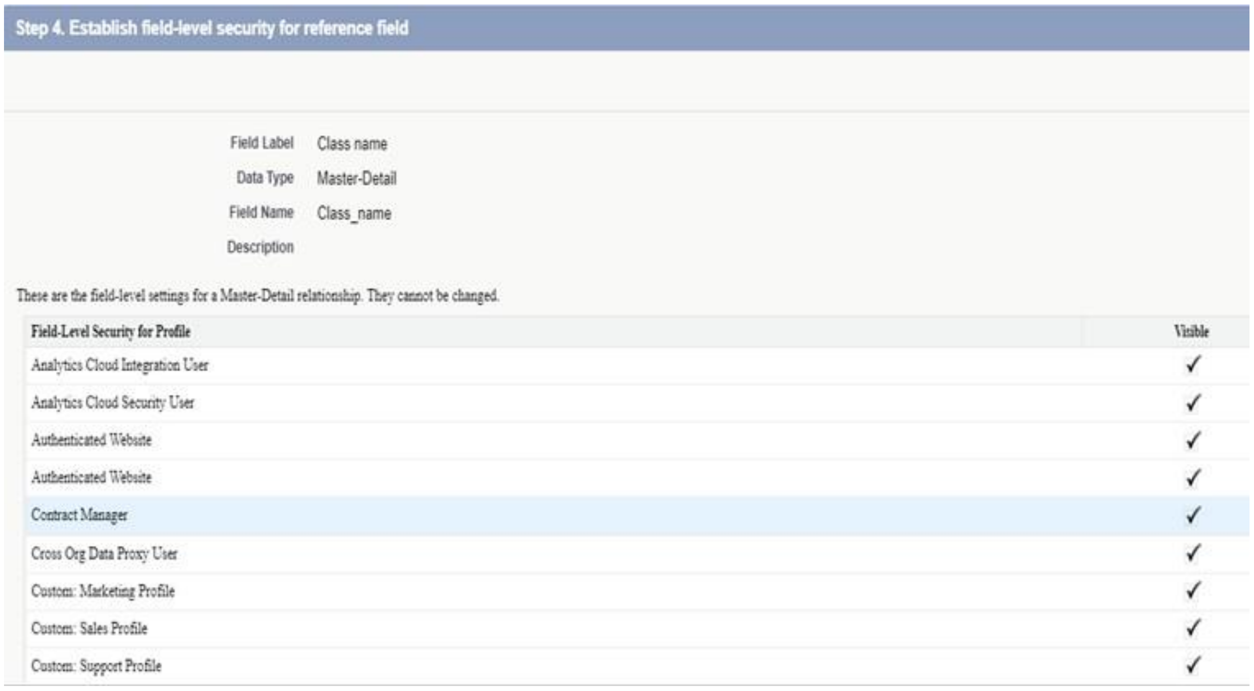
Student
New Relationship

Step 2. Choose the related object

Select the other object to which this object is related.

Related To: Class

STEP 3


Step 4. Establish field-level security for reference field

Field Label	Class name
Data Type	Master-Detail
Field Name	Class_name
Description	

These are the field-level settings for a Master-Detail relationship. They cannot be changed.

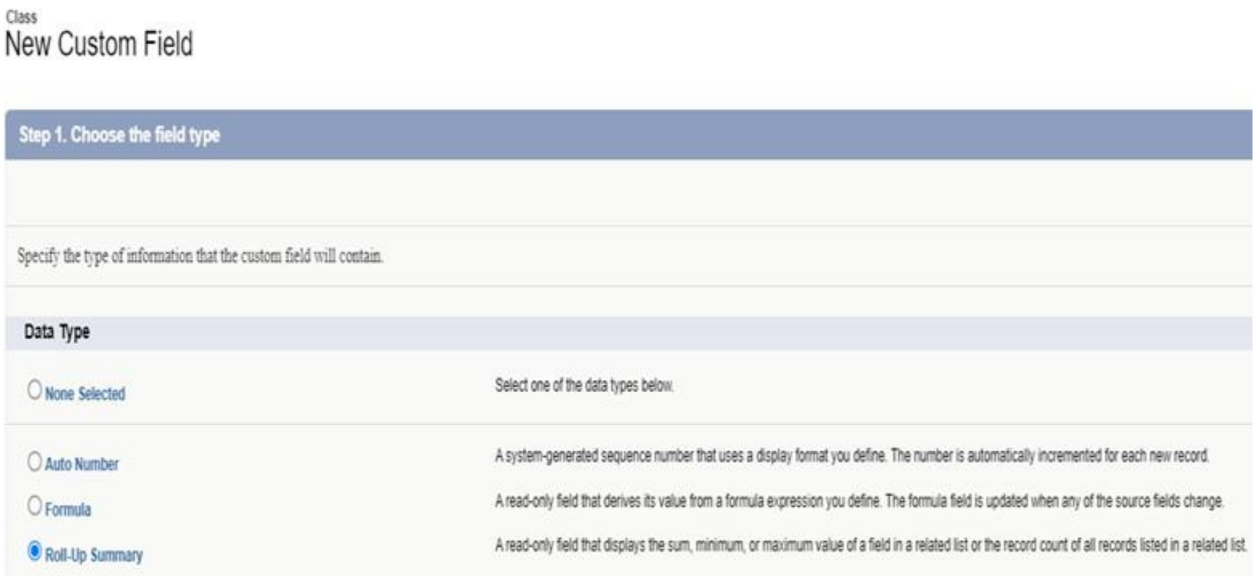
Field-Level Security for Profile	Visible
Analytics Cloud Integration User	✓
Analytics Cloud Security User	✓
Authenticated Website	✓
Authenticated Website	✓
Contract Manager	✓
Cross Org Data Proxy User	✓
Custom: Marketing Profile	✓
Custom: Sales Profile	✓
Custom: Support Profile	✓

Now if a Class Record is deleted then its associated student records will get deleted.

Roll-up summary Fields

- It gives the count of children or various values such as min, max, aggregate, etc to its parent record. It is always created in a Master-Detail Relationship.
- This field is created on the parent record, in this case, Class is the parent and Student is the child.

The following images show the process for creating a roll up summary.


Class

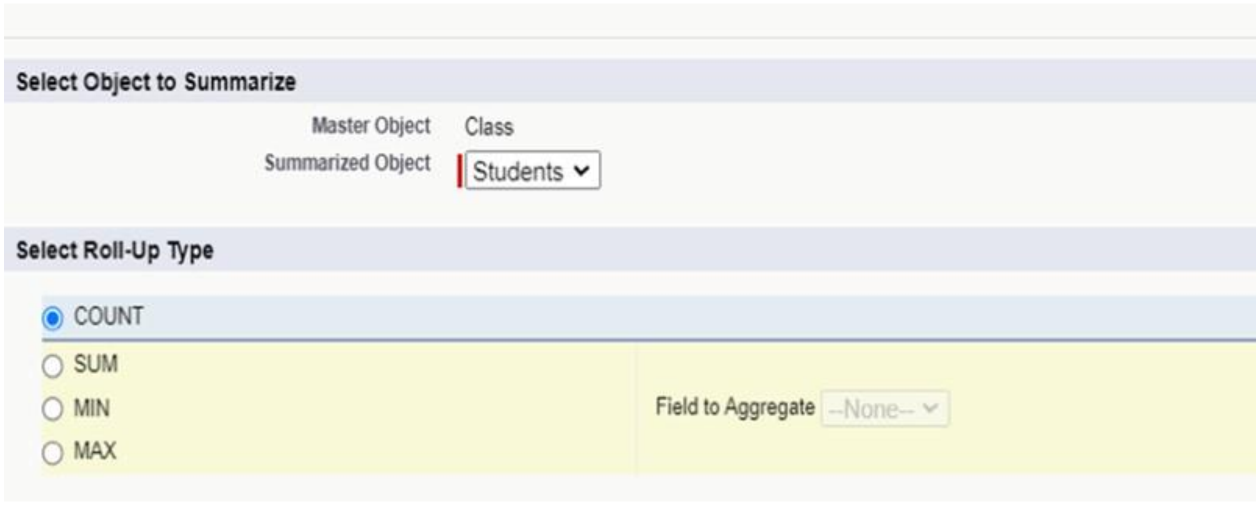
New Custom Field

Step 1. Choose the field type

Specify the type of information that the custom field will contain.

Data Type

None Selected	Select one of the data types below.
Auto Number	A system-generated sequence number that uses a display format you define. The number is automatically incremented for each new record.
Formula	A read-only field that derives its value from a formula expression you define. The formula field is updated when any of the source fields change.
[✓ Roll-Up Summary]	A read-only field that displays the sum, minimum, or maximum value of a field in a related list or the record count of all records listed in a related list.


Select Object to Summarize

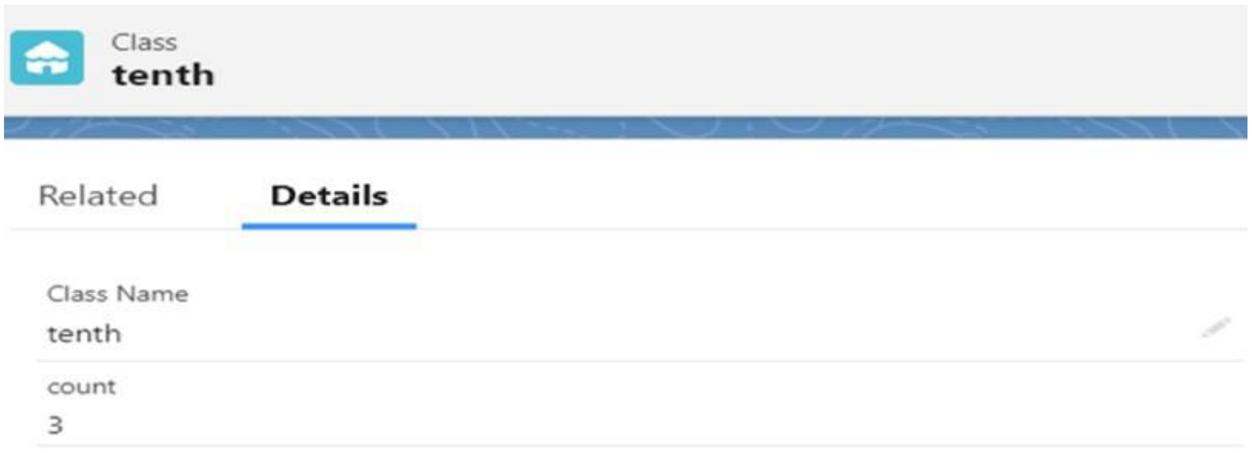
Master Object: Class

Summarized Object: Students

Select Roll-Up Type

[✓ COUNT] SUM MIN MAX

Field to Aggregate: --None--


Class

tenth

Related Details

Class Name

tenth

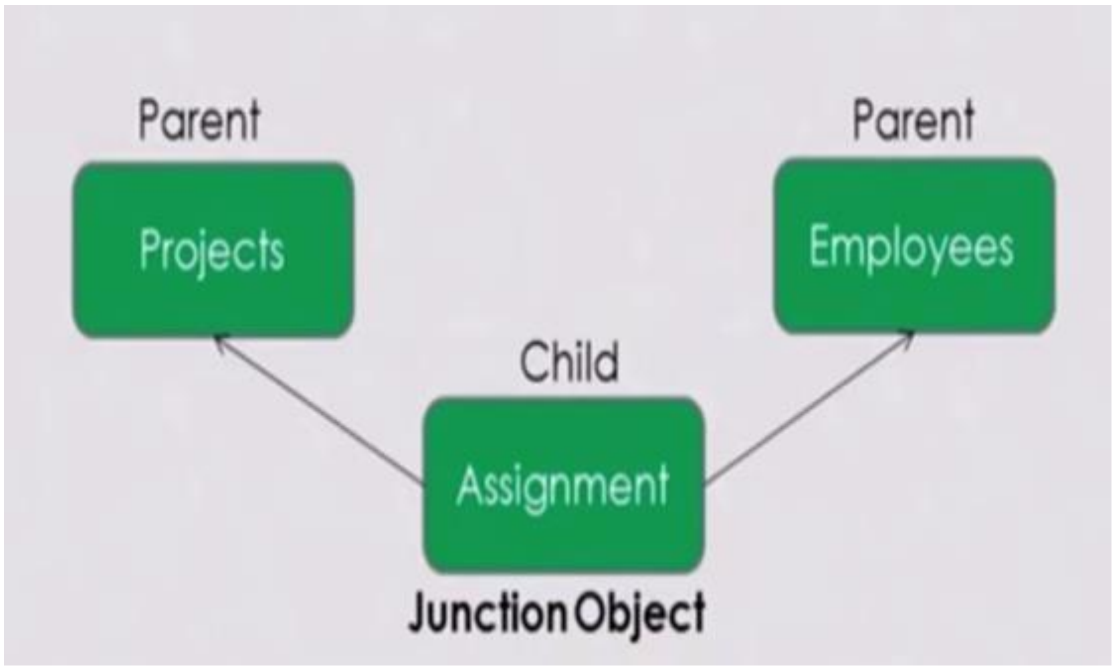
count

3

➔ Many to Many Relationships

If we have two objects namely Employees and Projects then a Project can have many employees working on it and an Employee can work on many projects also.

In such cases, we create a junction object. Here the junction object(Assignment) will have a master-detail relationship with both the objects, Project, and Employees.

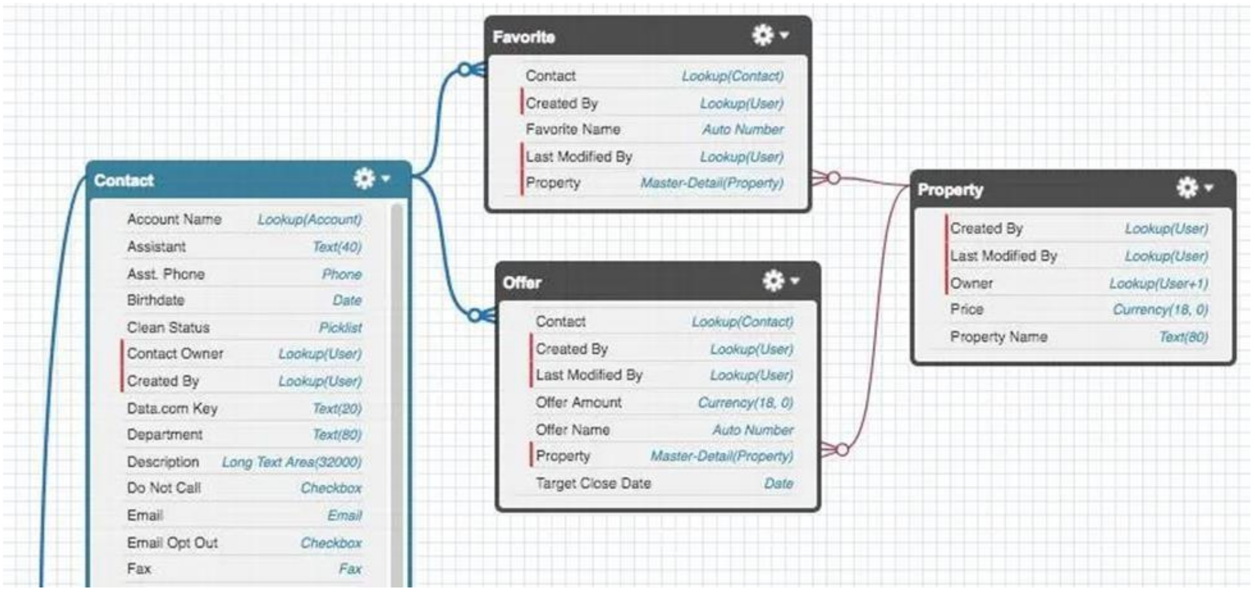


SCHEMA BUILDER

Schema Builder provides a dynamic environment for modifying and viewing all the relationships and objects in a Salesforce app. This greatly simplifies the task of designing, modifying and implementing the data model or the schema.

It can be used to view existing schema and inactively add new custom objects, fields, and relationships, simply by dragging and dropping. This eliminates the need to click from one page to another to find the details of a relationship or to add a new custom field to an object in the schema.

Schema Builder provides details such as the field values, fields that are required and how objects are related by displaying lookup and master-detail relationships. The fields & relationships can be viewed for both standard objects and custom objects.

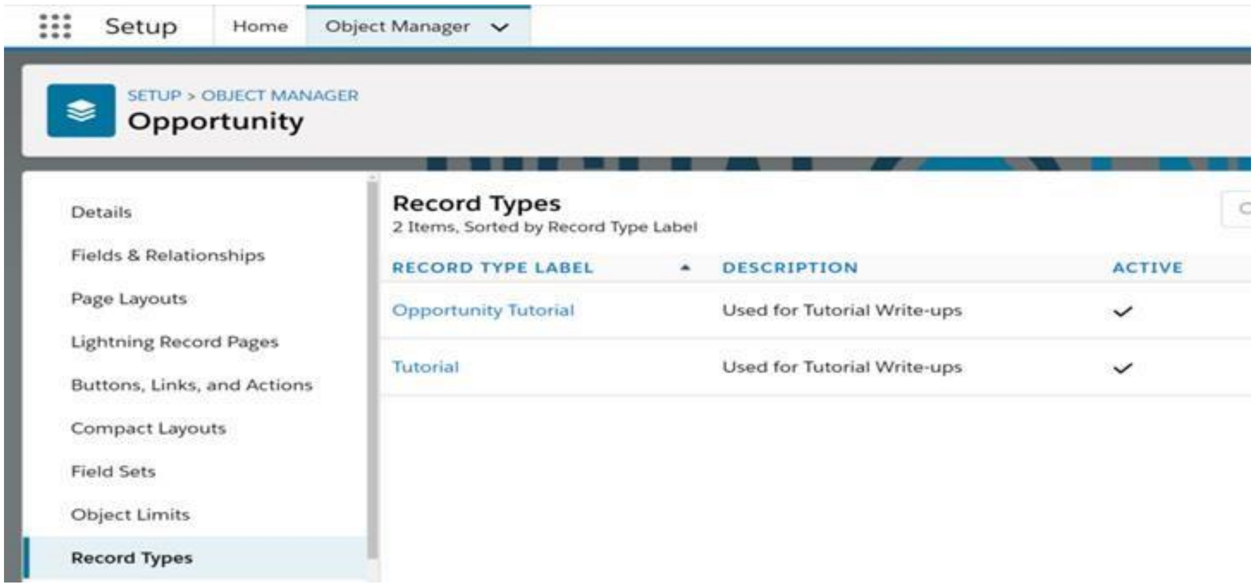


Schema Builder is enabled by default and it let the administrators add the following to the schema:

- Custom Objects
- All custom fields except Geolocation
- Lookup relationships
- Master-detail relationships

RECORD TYPES

Salesforce Record Types allow us to specify a category of records that display different picklist values and different page layouts. Administrators can associate record types with profiles so that different types of users should see different picklist values and different types of page layouts in the record detail page.



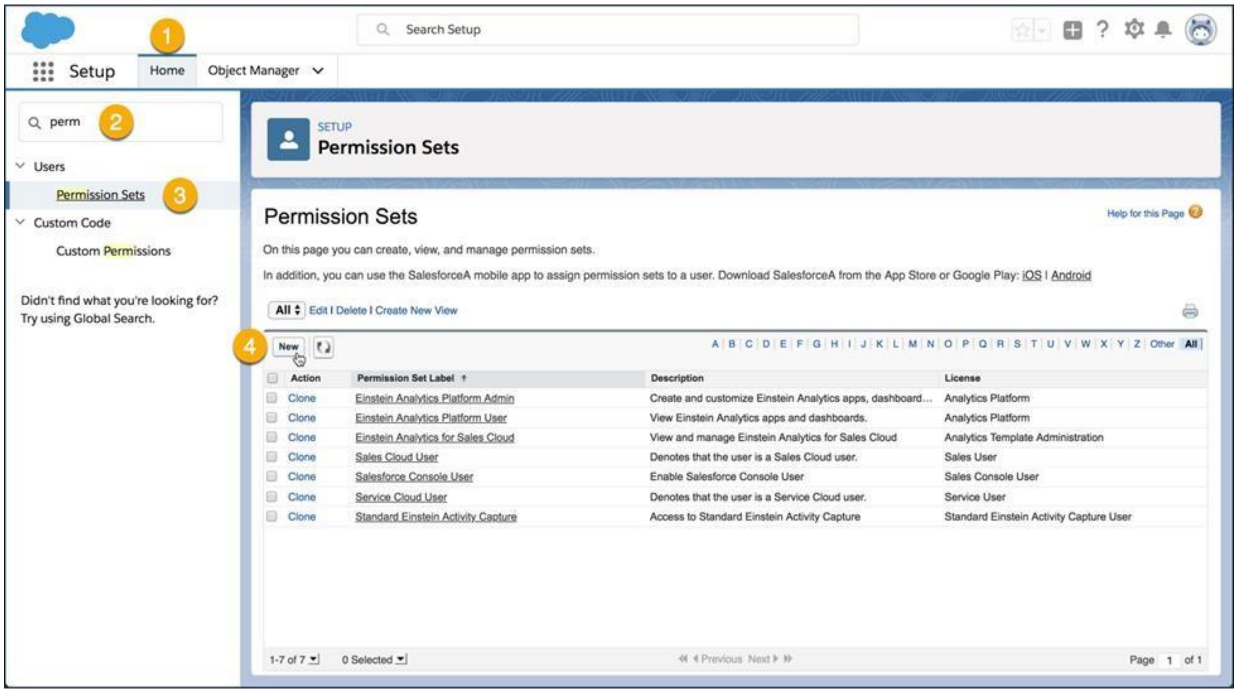
Record Types
2 Items, Sorted by Record Type Label

RECORD TYPE LABEL	DESCRIPTION	ACTIVE
Opportunity Tutorial	Used for Tutorial Write-ups	✓
Tutorial	Used for Tutorial Write-ups	✓

PERMISSION SETS

Salesforce Permission Sets

Permission sets in Salesforce are a collection of settings and permissions that determine user’s access to various tools and functions on the platform. Settings and permissions are available in permission sets also found in profiles but permission sets extend the functionality of users without changing their profiles. Use permission sets to grant additional access to specific users on top of their existing profile permissions, without having to modify an existing profile,create new profiles or grant an administrator profile where it’s not necessary.



Search Setup
1
Setup
Home
Object Manager
perm
2
Users
Permission Sets
3
Custom Code
Custom Permissions
Didn't find what you're looking for? Try using Global Search.
SETUP
Permission Sets
Help for this Page
All
Edit | Delete | Create New View
4
New
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All
1-7 of 7
0 Selected
Previous Next
Page 1 of 1

Permission Sets

On this page you can create, view, and manage permission sets.

In addition, you can use the SalesforceA mobile app to assign permission sets to a user. Download SalesforceA from the App Store or Google Play: iOS | Android

Action	Permission Set Label ↑	Description	License
Clone	Einstein Analytics Platform Admin	Create and customize Einstein Analytics apps, dashboard...	Analytics Platform
Clone	Einstein Analytics Platform User	View Einstein Analytics apps and dashboards.	Analytics Platform
Clone	Einstein Analytics for Sales Cloud	View and manage Einstein Analytics for Sales Cloud	Analytics Template Administration
Clone	Sales Cloud User	Denotes that the user is a Sales Cloud user.	Sales User
Clone	Salesforce Console User	Enable Salesforce Console User	Sales Console User
Clone	Service Cloud User	Denotes that the user is a Service Cloud user.	Service User
Clone	Standard Einstein Activity Capture	Access to Standard Einstein Activity Capture	Standard Einstein Activity Capture User

ORGANIZATION-WIDE DEFAULT

Organization-Wide default or otherwise known as Organization-Wide sharing settings determine the baseline level of access for all records of an object. Organization-Wide defaults or Organization-Wide can never grant users more access than they have through their object permissions.

Organization-Wide defaults should be the most restrictive in record level security because other record-level security implementations only grant additional accesses; they cannot restrict the access of records provided by Organization-Wide defaults.

Organization-Wide defaults can be set to any of the 3 following given below:

1. Public Read/Write:

   All users can view, edit and report on all records.

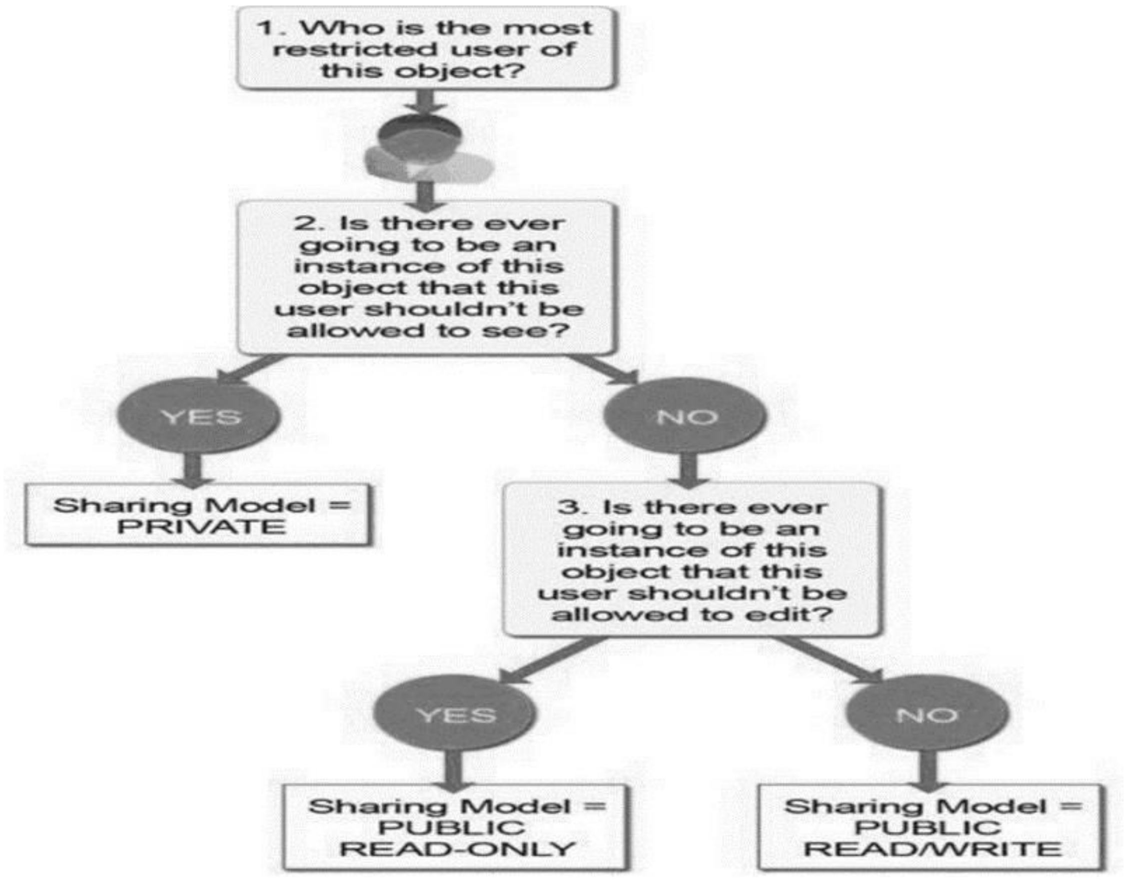
2. Public Read-Only:

   All the users can view and report on records but not edit them. Only the owner and users above that role in the hierarchy, can only edit those records.

3. Private:

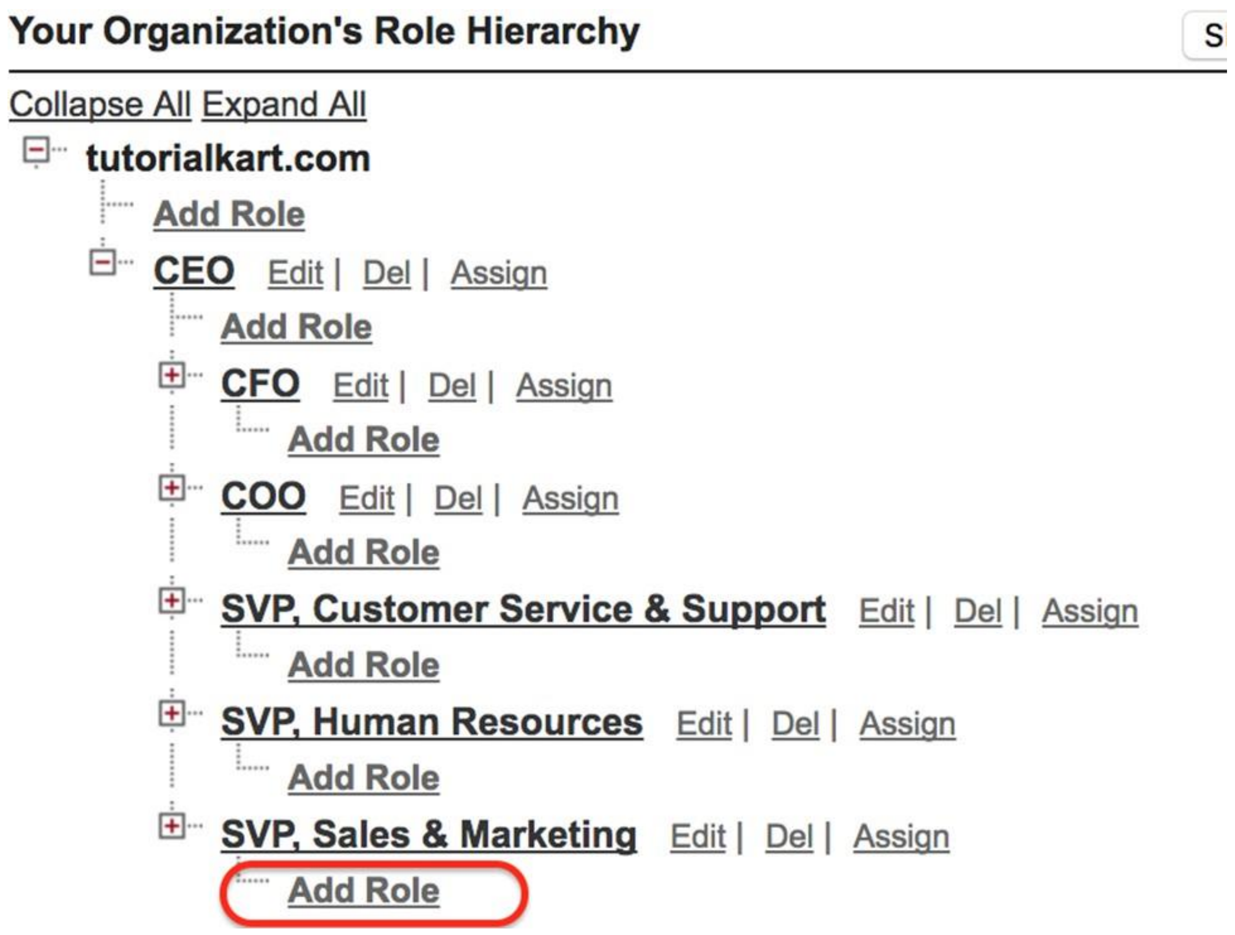
   Only the record owner and users above that role in the hierarchy, can view, edit, and report on those records.

To determine the Organization-Wide default of an object consider the following give below diagram:


1. Who is the most restricted user of this object?
2. Is there ever going to be an instance of this object that this user shouldn't be allowed to see?
YES
NO
Sharing Model = PRIVATE
3. Is there ever going to be an instance of this object that this user shouldn't be allowed to edit?
YES
NO
Sharing Model = PUBLIC READ-ONLY
Sharing Model = PUBLIC READ/WRITE

SALESFORCE ROLE HIERARCHY

Every Salesforce organization maintains a role hierarchy for the organization who uses Salesforce. The role hierarchy defines the hierarchy of the users working in the respective organization. Salesforce Role hierarchies don't have to match your organization chart exactly. Instead, each role in the hierarchy can just represent a level of data access that a user or group of users needs. The roles can control the level of visibility that users have into the data, by the Organization's sharing settings. Users at any given role level can view, edit and report on all data shared with users below them in the role hierarchy, unless the organization's sharing model for an object specifies otherwise.



Your Organization's Role Hierarchy

Collapse All Expand All

```
tutorialkart.com
    Add Role
    CEO  Edit | Del | Assign
        Add Role
        CFO  Edit | Del | Assign
            Add Role
        COO  Edit | Del | Assign
            Add Role
        SVP, Customer Service & Support  Edit | Del | Assign
            Add Role
        SVP, Human Resources  Edit | Del | Assign
            Add Role
        SVP, Sales & Marketing  Edit | Del | Assign
            Add Role
```

SALESFORCE ACCOUNTS

An Account is a business entity, it's basically a company. We need insight into our business and our data and that starts with the people we are doing business with. We can store information about all our customers using accounts and contacts. Accounts are basically the companies that we are doing business with.

An Account is an organization that is nothing but a qualified potential customer, an existing customer, partner, competitor or has a relationship of similar significance. Accounts should be created for organizations (i.e; companies, nonprofits, foundations), Individuals.

The account is nothing but the glue that holds the relationships (contacts) and interactions (i.e; activities, opportunities, cases, etc.) with an organization. The standard Salesforce account is configured for business to business (B2B) relationships.

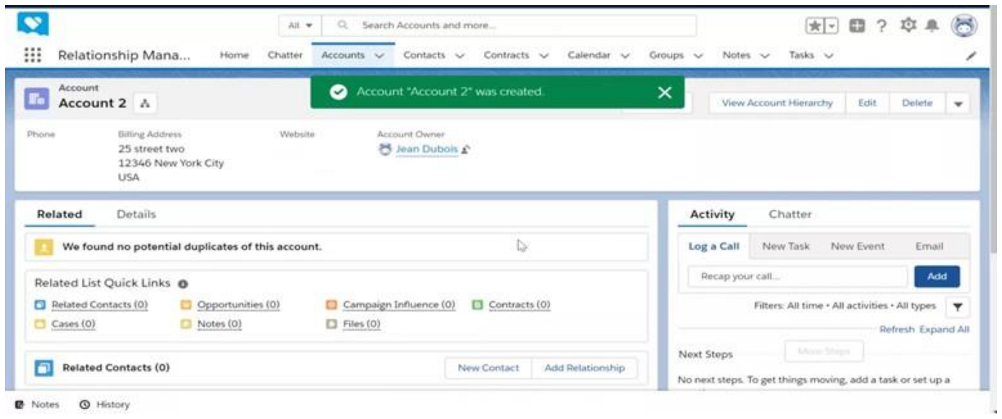
Important Fields:

1. Parent Account

   This is a self-relationship field. It relates two (2) accounts with each other. The parent account is selected from the lookup dialog and the other one on which the field exists is the child account. A parent account cannot be considered as a child itself.

2. Account Name

   The name of the company with whom we are doing business.



Account
Account 2

Account "Account 2" was created.

Phone

Billing Address
25 street two
12346 New York City
USA

Website

Account Owner
Jean Dubois

We found no potential duplicates of this account.

Filters: All time • All activities • All types

No next steps. To get things moving, add a task or set up a

SALESFORCE CONTACTS

Contacts in Salesforce store an individual's demographic information, such as phone numbers and email addresses which are linked to accounts. If a contact is not linked to an account then it is a "private" contact and can only be viewed by the contact owner or Salesforce administrator.

Contact Role fields are as follows:

1. Contact

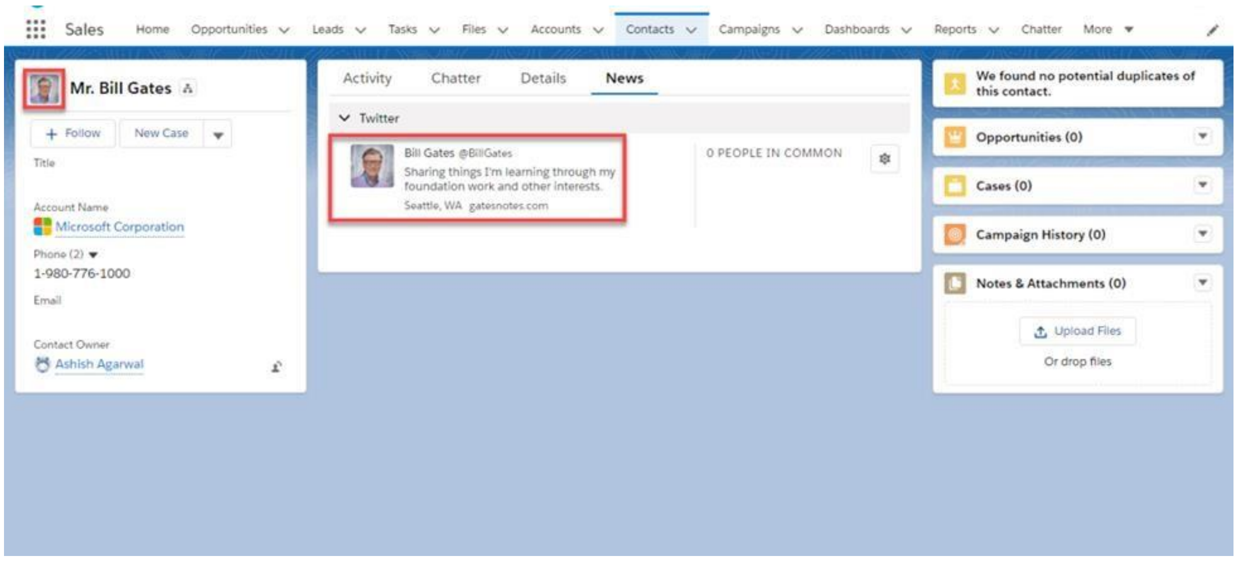
   The name of the contact or person account. Selecting an existing account or creating a new one.

2. Primary

   When selected, it identifies the person as the primary contact for the record. The Primary option isn't available for cases, instead, the contact listed in the Contact Name field on the case record is automatically the primary contact. There can be only one primary role in this list for one account at a time.

3. Role

   The role of contact for the record. The administrators can customize the selections for this picklist on a particular object.



Mr. Bill Gates

Title

Account Name
Microsoft Corporation

Phone (2)
1-980-776-1000

Email

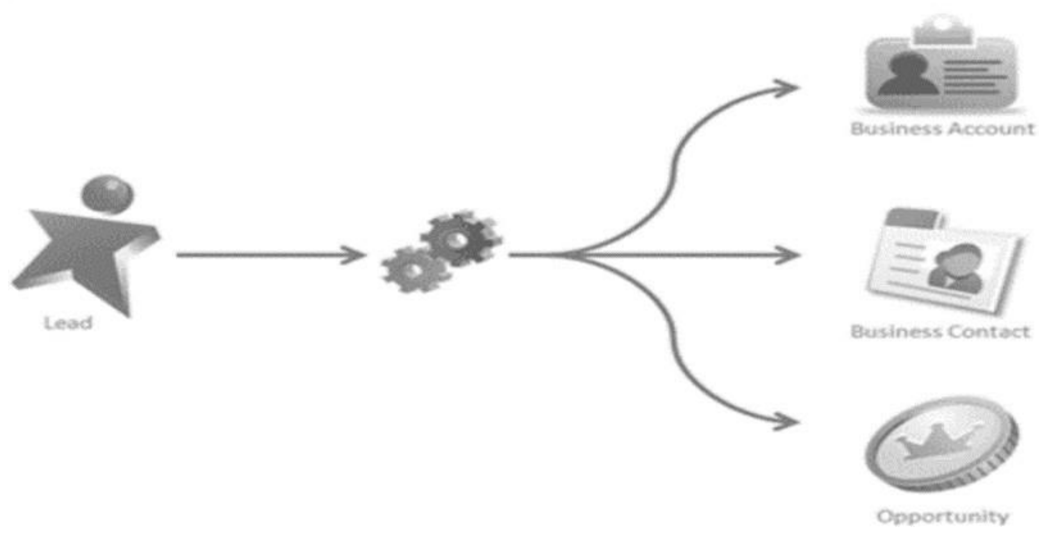
Contact Owner
Ashish Agarwal

Bill Gates @BillGates
Sharing things I'm learning through my foundation work and other interests.
Seattle, WA gatesnotes.com

We found no potential duplicates of this contact.

SALESFORCE LEAD CONVERSION

Salesforce Lead conversion is a process in which a lead record is converted into Accounts, Contacts & Opportunities. This happens when a lead is qualified for a Sales prospect.



When a lead is converted?

- A contact, an account & an opportunity are created and populated with the lead's data.
- The converted lead field is changed from false to true.
- There is no way to convert a lead into an existing opportunity.
- When we convert a lead, Salesforce attempts to find an account with the same name as the field "Company" on the record. If an account name contains the company's name, then will have the option to use the existing record.
- If we attach lead to an existing account and the lead name matches the name of an existing contact, then you will have the option to use the existing contact record also.
- Converted leads cannot be modified at all.
- It's always not necessary to create an opportunity when converting a lead.
- The default record type for the user performing the lead conversion is selected, automatically for the records created in the conversion process.
- Once lead is converted it cannot be reverted to an unconverted state.

- Salesforce does not overwrite existing Account or Contact data on lead conversion.

SALESFORCE OPPORTUNITIES

Opportunities in Salesforce are nothing but a representation of a transaction between your company and an Account. Typically this is a potential sales transaction that would include the information about the specific products and/or services one of your sales representatives is presenting to a prospective customer.

Important Fields

1. Opportunity Name

   The required text field represents the name of the specific deal as you want it to appear on your list of opportunities or on a pipeline report.

2. Amount:

   This field displays the total cost of a specific opportunity.

3. Close Date

   This field describes when you'll close a specific deal. It is a necessary field. This field is commonly used to track the date on which the deal will be/is closed.

4. Expected Revenue

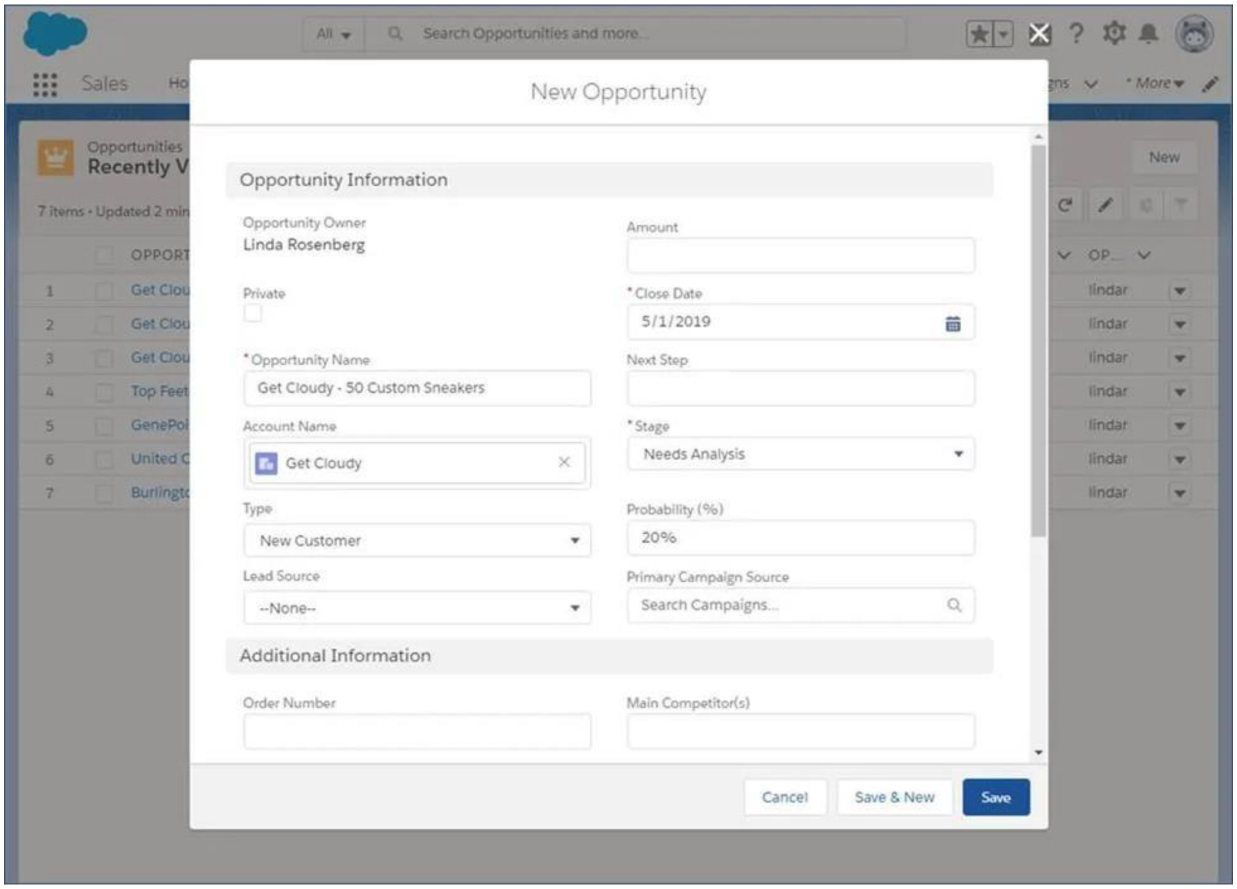
   This is a read-only field which is automatically generated by multiplying the Amount field by the Probability field.

5. Opportunity Owner

   The person in your organization who owns the opportunity. Though an opportunity record has only one owner, many users can still collaborate on an opportunity with the help of an opportunity team related list.

6. Probability

This is a percentage field. The probability is the confidence factor associated with the likelihood that you will win the respective opportunity. At each sales stage that your company defines is associated with a default probability to close. When you win a deal, the probability is 100% because you won it, but, if you lost it then it is 0%. System administrators can modify these percentages and also add new stages.



New Opportunity

Opportunity Information

Opportunity Owner	Linda Rosenberg
Amount	
Private	☐
*Close Date	5/1/2019
*Opportunity Name	Get Cloudy - 50 Custom Sneakers
Next Step	
Account Name	Get Cloudy
*Stage	Needs Analysis
Type	New Customer
Probability (%)	20%
Lead Source	--None--
Primary Campaign Source	Search Campaigns...

Additional Information

Order Number	
Main Competitor(s)	

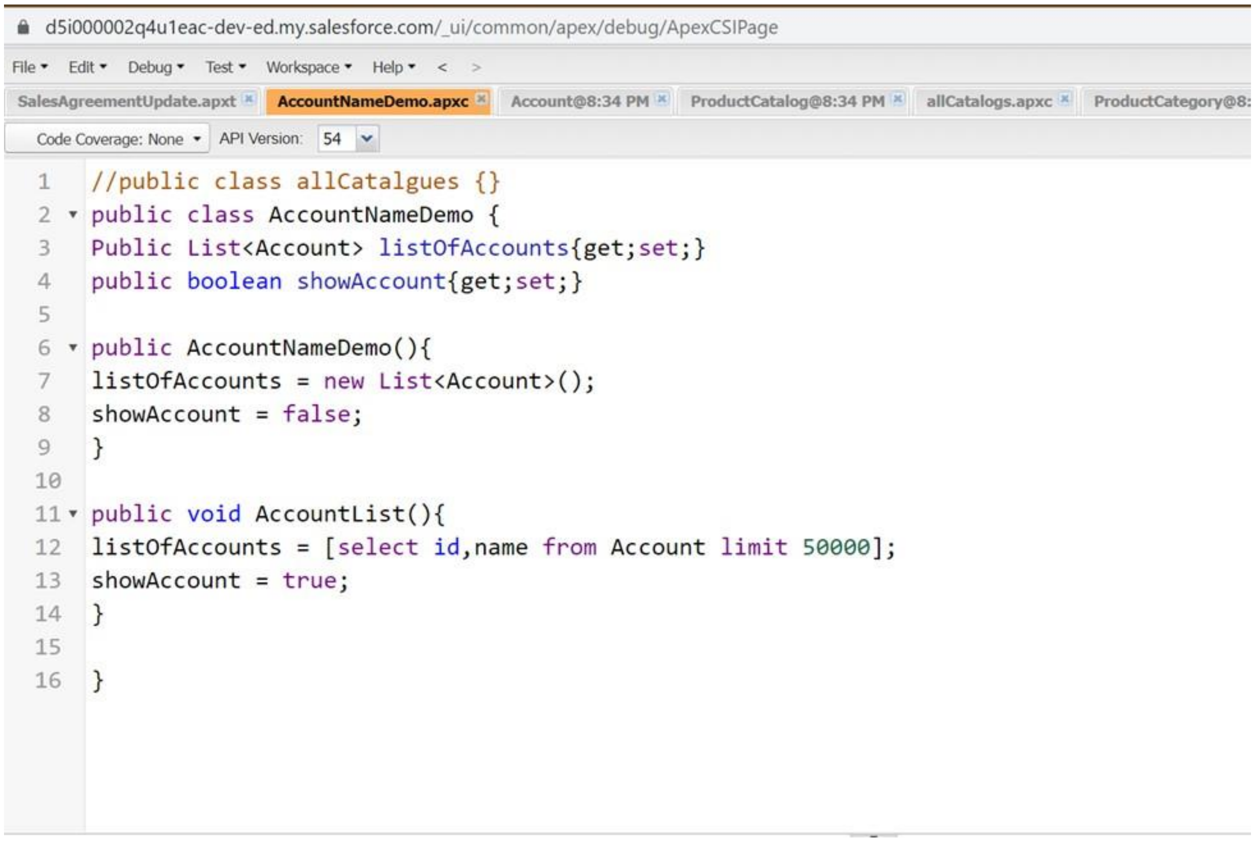
Apex : The Salesforce Programming Language

APEX CLASS

The significance of apex in salesforce is that it helps the developers to fetch/get the data from the back-end of salesforce. Along with that, it also helps provide client server interface for the creation of third party applications.

The syntax of Apex is mostly similar to that of java. We often integrate Apex with Lightning Web Components (LWC) for getting a customized component in the salesforce org UI.

Here is an example of an Apex code that was written during the training period.The following code gets a list of all the associated accounts from the backend of the salesforce org.



```
//public class allCatalgues {}
public class AccountNameDemo {
Public List<Account> listOfAccounts{get;set;}
public boolean showAccount{get;set;}

public AccountNameDemo(){
listOfAccounts = new List<Account>();
showAccount = false;
}

public void AccountList(){
listOfAccounts = [select id,name from Account limit 50000];
showAccount = true;
}

}
```

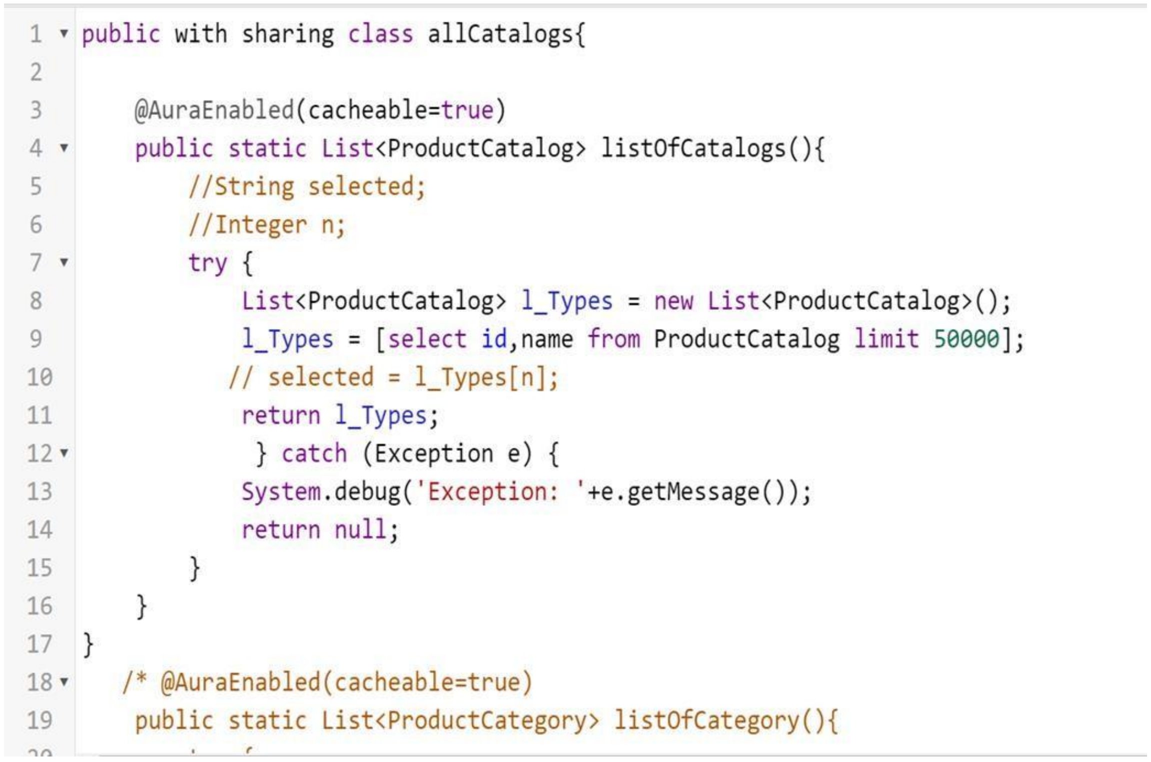
This Apex code consists of three parts:

1. An apex class
2. An apex method
3. SOQL query.

Bringing the familiarity of java into the picture, we can pretty much figure out how the class and method work. The one difference that we observe is the SOQL query which stands for Salesforce Object Query Language. We will be talking about that in detail further.

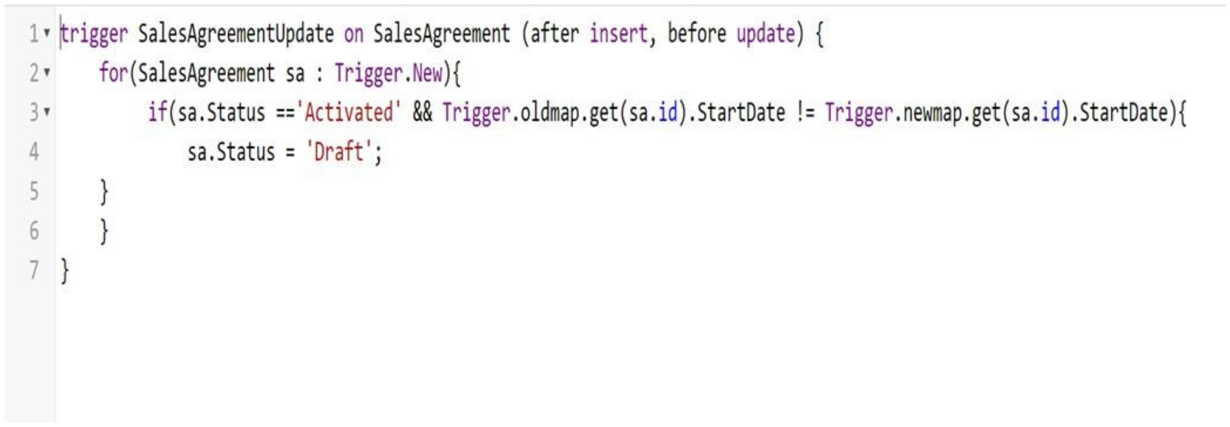
Coming back to the apex code, there are a couple of apex codes that had to be written during internship.

1. Code which gets a list of catalogs from the backend and when a catalog is selected, it returns a list of all the available categories associated with it.



```
public with sharing class allCatalogs{

    @AuraEnabled(cacheable=true)
    public static List<ProductCatalog> listOfCatalogs(){
        //String selected;
        //Integer n;
        try {
            List<ProductCatalog> l_Types = new List<ProductCatalog>();
            l_Types = [select id,name from ProductCatalog limit 50000];
           // selected = l_Types[n];
            return l_Types;
             } catch (Exception e) {
            System.debug('Exception: '+e.getMessage());
            return null;
        }
    }
}
   /* @AuraEnabled(cacheable=true)
    public static List<ProductCategory> listOfCategory(){
```

2. Code to change the status of the agreement to draft when the agreement start date is changed.



```
trigger SalesAgreementUpdate on SalesAgreement (after insert, before update) {
    for(SalesAgreement sa : Trigger.New){
        if(sa.Status =='Activated' && Trigger.oldmap.get(sa.id).StartDate != Trigger.newmap.get(sa.id).StartDate){
            sa.Status = 'Draft';
    }
    }
}
```

APEX DATABASE

Now, coming to SOQL, which is used to get only the required data from the backend is a database query language, similar to SQL. The only difference is that other than the existing data types, SOQL uses their own data type which is known as sObjects.

Here are a few examples of the SOQL queries:

1. select id,name from Account limit 50000
2. select id,name from ProductCatalog limit 50000
3. select id,name from ProductCategory WHERE CatalogID=:abc limit 50000

We can use SOQL inside the apex code.

This was about how we use Apex classes to customize user experience and make sure the client gets what exactly they require in their business org.

Now, let's move on to apex triggers.

APEX TRIGGERS

What are apex triggers?

Apex triggers are pieces of code that perform a certain specified action when a trigger condition is met. For example, if you want for an account to have only 10 contacts associated with it, you can activate a trigger that will pop up when the user tries to enter the 11th contact.

For working with triggers, you need to use the following syntax:

```
Trigger TriggerName on ObjectName (Trigger_events) {

     //  Code;

}
```

The code refers to the content which decides the trigger events and how to work with them.

There are the following types of trigger events:

1. Before insert
2. Before delete
3. Before update
4. After insert
5. After update
6. After delete
7. After undelete.

Following is an example of a trigger which occurs when someone is about to insert an account and displays a message.

```
Trigger HelloWorldTrigger onAccount (before insert) {

          System.debug('Hello World!');

}
```

Types of Triggers

There are two types of triggers:

1. Before Triggers : these are used to update or validate records before the data has been inserted into the database.
2. After Triggers : already set field values are accessed using after triggers. If we want any changes to the record, the records which fire the after trigger are read only.

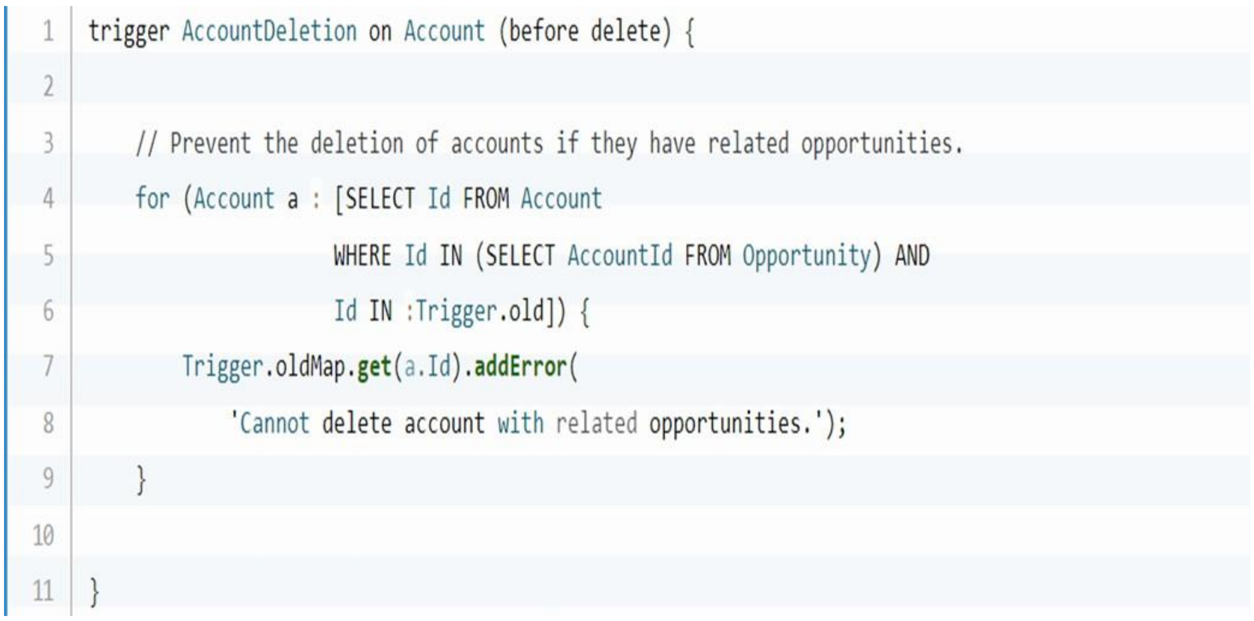
Trigger context variables

Following are the types of trigger context variables :

1. isExecuting - If the current context for the Apex code is a trigger, rather than a Visualforce page, a Web service, or an executeanonymous() API request, it returns true.
2. isInsert - If this trigger was triggered by an insert operation from the Salesforce user interface, Apex, or API, it returns true.
3. isUpdate - If this trigger was triggered by an update action from the Salesforce user interface, Apex, or API, it returns true.
4. isDelete - If this trigger was triggered by a delete operation from the Salesforce user interface, Apex, or API, it returns true.
5. isBefore - Returns true if this trigger occurs before the record is saved.
6. isAfter - Returns true if this trigger is fired after all records have been saved.
7. isUndelete - Returns true if this trigger is triggered after the record has been restored from the Recycle Bin. This restore can be done after a restore operation via the Salesforce UI, Apex, or API.
8. New - Returns a list of new versions of sObject records.This sObject list is only available with Insert, Update, and Undelete triggers, and records can only be modified with Before triggers.
9. newMap - Mapping the ID to a newer version of the sObject record. This mapping is only available for pre-update, post-insert, post-update, and post-restore triggers.
10. oldMap - Mapping the ID to an older version of the sObject record. This mapping is only available for update and delete triggers.

11. Old - Returns a list of older versions of sObject records.This sObject list can only be used with update and delete triggers.
12. operationType - Returns a System.TriggerOperation enumeration that corresponds to the current operation. Possible values for the System.TriggerOperation enumeration are BEFORE_INSERT, BEFORE_UPDATE, BEFORE_DELETE, AFTER_INSERT, AFTER_UPDATE, AFTER_DELETE, and AFTER_UNDELETE. If you want to modify your programming logic based on different trigger types, consider using switch statements in different sequences of unique trigger execution enumeration states.
13. Size - The total number of records for the trigger call, both old and new.

➔ Following is the example of a trigger which fires when a user tries to delete an account with related opportunities.



```
trigger AccountDeletion on Account (before delete) {

    // Prevent the deletion of accounts if they have related opportunities.
    for (Account a : [SELECT Id FROM Account
                     WHERE Id IN (SELECT AccountId FROM Opportunity) AND
                     Id IN :Trigger.old]) {
        Trigger.oldMap.get(a.Id).addError(
            'Cannot delete account with related opportunities.');
    }

}
```

APEX TESTING

The Apex test framework allows you to create and run tests for Apex classes and triggers on the Lightning platform. Apex unit tests guarantee high quality Apex code and allow you to meet Apex deployment requirements. Testing is the key to successful long-term development and an important part of the development process.

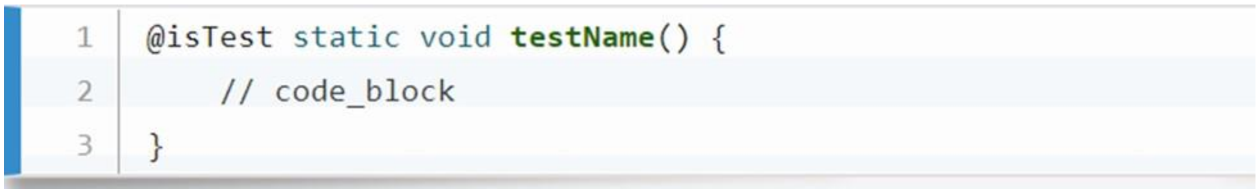
The Apex testing framework makes it easy to test your Apex code. Apex code can only be written in a sandbox environment or a developer org, not in a production environment. Apex code can be deployed from the production org sandbox. In addition, application developers can distribute Apex code from their developer organization to their customers by uploading the package to the Lightning Platform AppExchange.

In addition to being essential for quality assurance, Apex unit tests are a prerequisite for deploying and distributing Apex. Below are the benefits of Apex unit testing.

Make sure your Apex classes and triggers are working as expected. A series of regression tests that can be rerun every time a class or trigger is updated ensures that future updates to your app won't compromise existing functionality.

Meet code coverage requirements for deploying Apex to production or distributing Apex to customers via packages. Deliver high-quality apps to your production org, increasing the productivity of your production users. High-quality apps delivered to package subscribers that build trust with customers.

Test method syntax

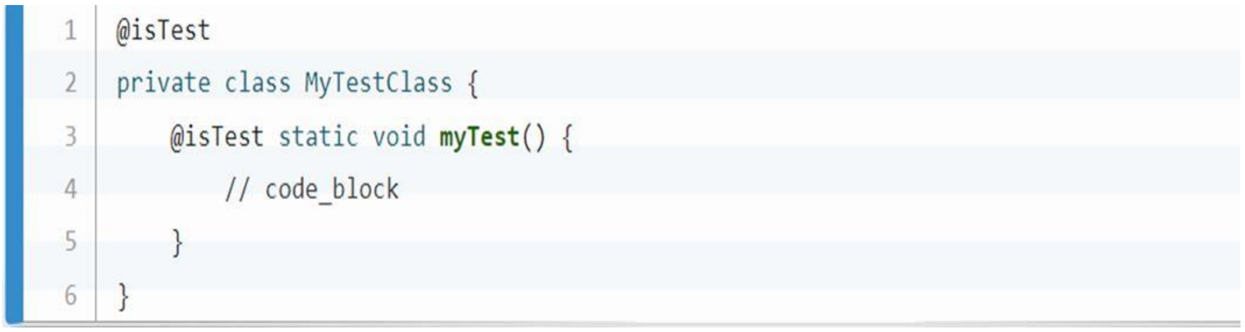


```
@isTest static void testName() {
    // code_block
}
```

The isTest annotation accepts multiple qualifiers, enclosed in parentheses and separated by spaces. We will deal with such parameters later.

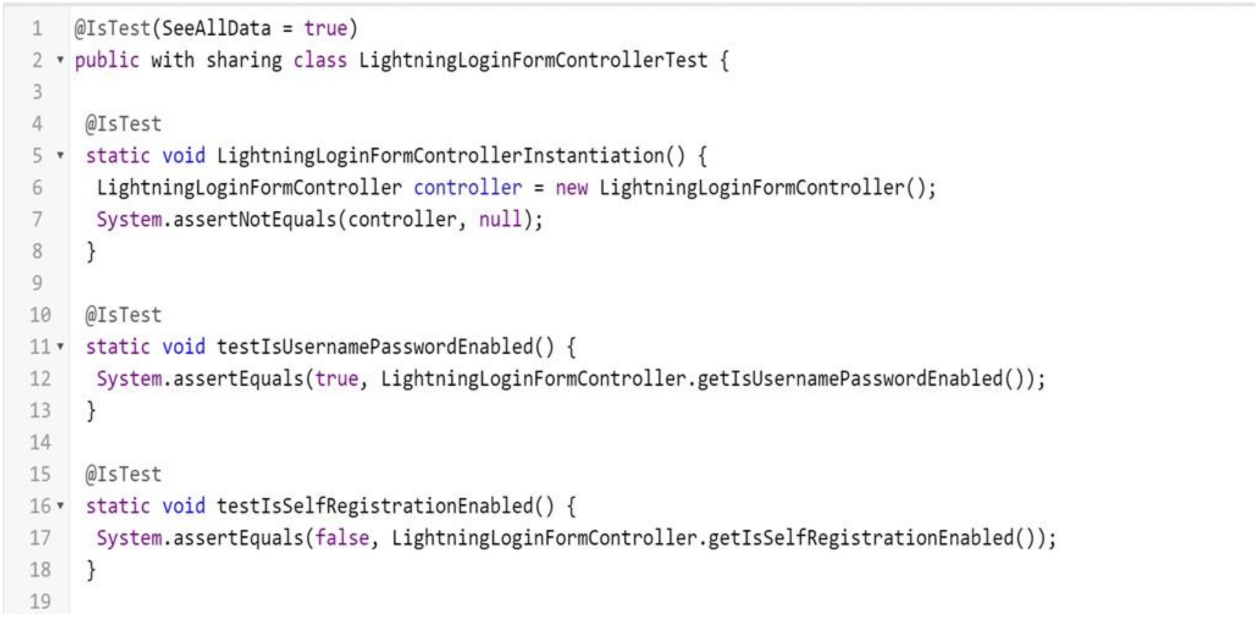
The visibility of the test method is not important. Therefore, it makes no difference to declare a test method as public or private because the test framework always has access to the test method. For this reason, access modifiers have been omitted from the syntax.

The test method must be defined in the test class, which is the class annotated with isTest. This sample class shows the definition of a test class using a test method.

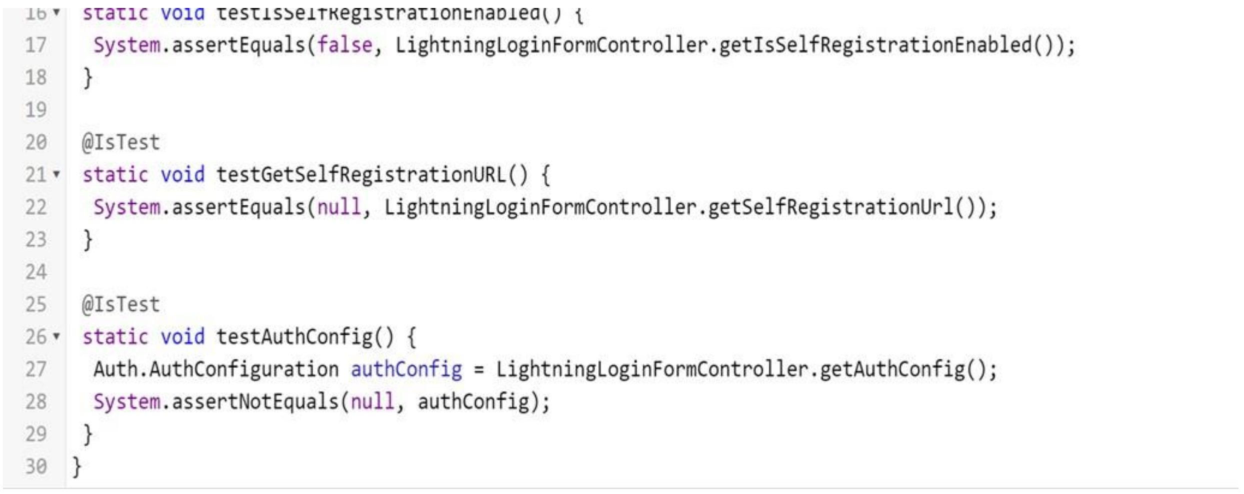


```
@isTest
private class MyTestClass {
    @isTest static void myTest() {
        // code_block
    }
}
```

Example of a test class for a lightning form controller apex class:



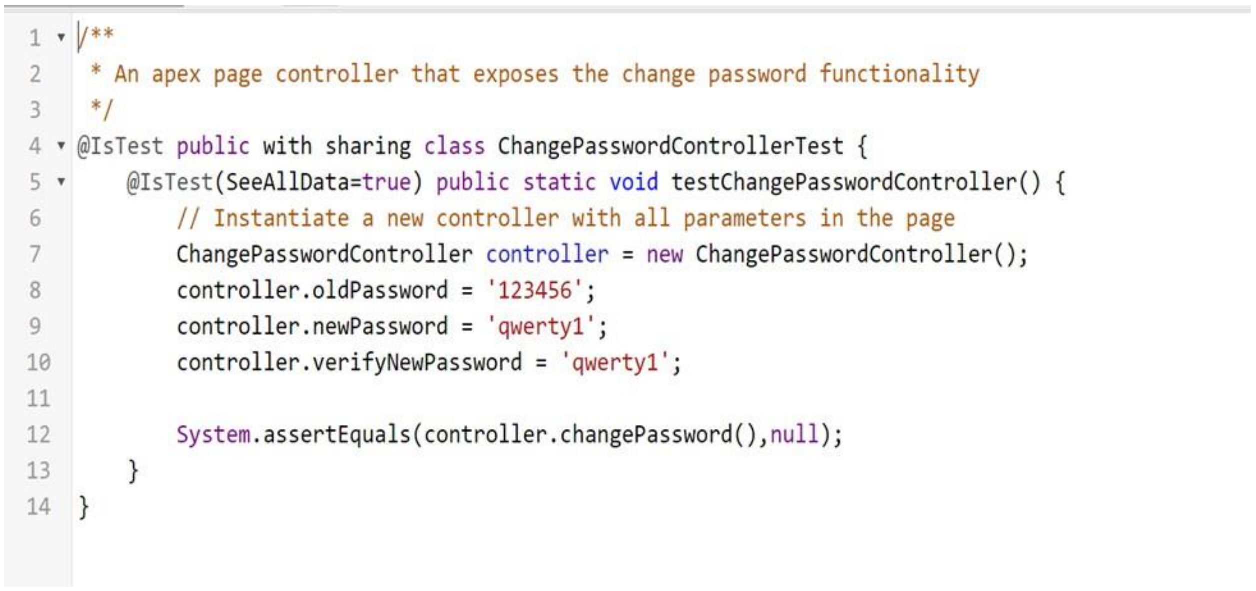
```
@IsTest(SeeAllData = true)
public with sharing class LightningLoginFormControllerTest {

 @IsTest
 static void LightningLoginFormControllerInstantiation() {
  LightningLoginFormController controller = new LightningLoginFormController();
  System.assertNotEquals(controller, null);
 }

 @IsTest
 static void testIsUsernamePasswordEnabled() {
  System.assertEquals(true, LightningLoginFormController.getIsUsernamePasswordEnabled());
 }

 @IsTest
 static void testIsSelfRegistrationEnabled() {
  System.assertEquals(false, LightningLoginFormController.getIsSelfRegistrationEnabled());
 }

```



```
 static void testIsSelfRegistrationEnabled() {
  System.assertEquals(false, LightningLoginFormController.getIsSelfRegistrationEnabled());
 }

```

```
 @IsTest
 static void testGetSelfRegistrationURL() {
  System.assertEquals(null, LightningLoginFormController.getSelfRegistrationUrl());
 }

 @IsTest
 static void testAuthConfig() {
  Auth.AuthConfiguration authConfig = LightningLoginFormController.getAuthConfig();
  System.assertNotEquals(null, authConfig);
 }
}
```

Example of a change password controller test:



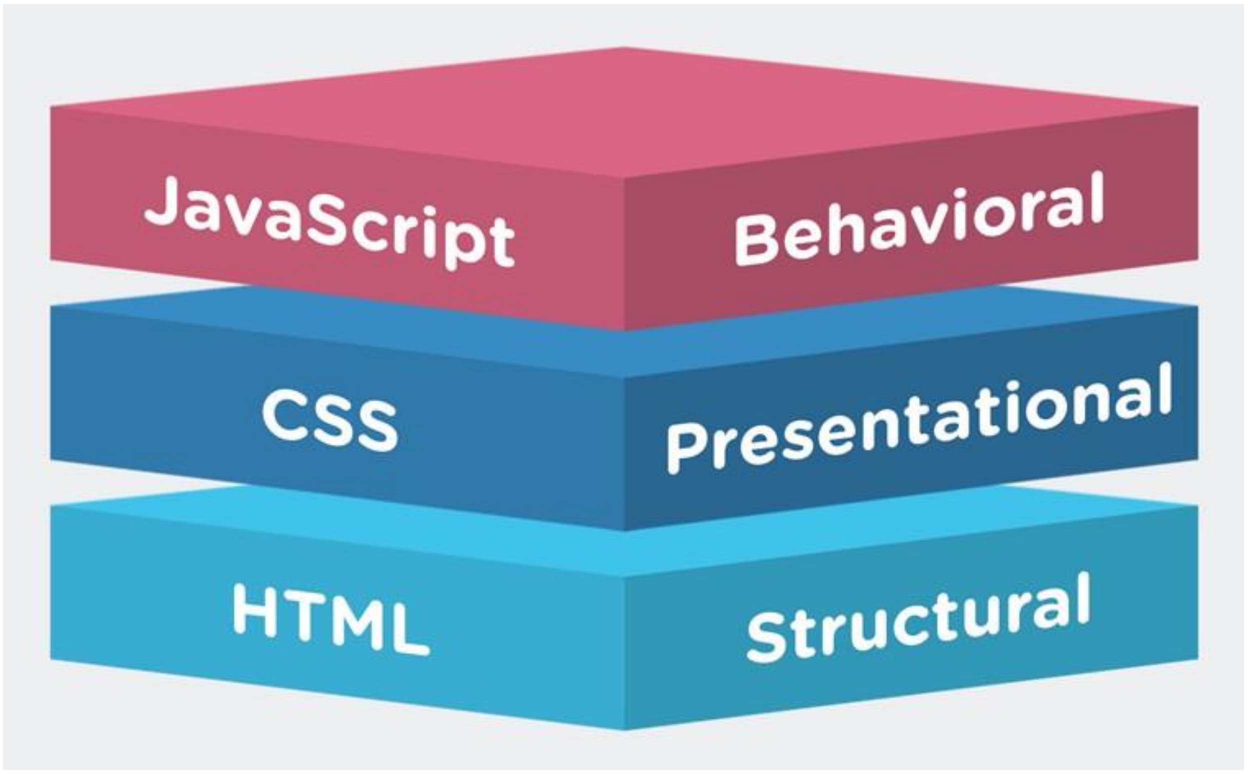
```
/**
 * An apex page controller that exposes the change password functionality
 */
@IsTest public with sharing class ChangePasswordControllerTest {
    @IsTest(SeeAllData=true) public static void testChangePasswordController() {
        // Instantiate a new controller with all parameters in the page
        ChangePasswordController controller = new ChangePasswordController();
        controller.oldPassword = '123456';
        controller.newPassword = 'qwerty1';
        controller.verifyNewPassword = 'qwerty1';

        System.assertEquals(controller.changePassword(),null);
    }
}
```

That was about Apex, Apex classes, Apex database, Apex triggers and Apex testing with how they are used and what is their significance.

WEB TECHNOLOGY

In the world we live in today, the internet has become the axis around which the world of information revolves. People learn through the internet. Commerce and transactions are facilitated through the internet. Social change is spearheaded on the internet. Since its mainstream adoption in the early 2000s, almost all traditional software applications have become web applications. Therefore, to leverage the internet in one’s favor, one must be able to create powerful web applications. To do just that, we will look at how to build a typical web page.

A web page is created by combining three languages - HTML, CSS and JavaScript.



HTML (HYPER TEXT MARKUP LANGUAGE)

Hypertext refers to the way in which web pages are linked together. Markup Language is a computer language that uses tags to define elements within a document. HTML is used to define and create the general structure of the web page. It is used to create and place elements including, but not limited to - headings, paragraphs, lists, etc. HTML elements pass data to the browser, communicating to it- how the content is to be displayed.

HTML element

A HTML element is defined by a start tag, some content, and an end tag. The syntax looks like this : <tagname> Content </tagname>. The tag is contained within angular brackets <>, and the closing tag contains a slash '/'. HTML elements can be divided into two main types

1. Block elements - These elements always start on a new line and take up the entire width. Ex - <div>, <p>, <h1>   <h6>, <ul>, etc
2. Inline elements - These elements don't start on a new line, and they take up as much space as necessary only. Ex - <span>, <img>, <strong>, <a>, <label>.There are some very commonly used HTML elements in Lightning Web Components. They are -
   - <div> - This tag is used to define a division of content within the HTML document.
   - <p> - This tag is used to define a paragraph that starts on a new line, and to add some space before and after the paragraph.
   - <h1>, <h2>, ……, <h6> - These tags are used to define headings of different sizes - <h1> being the biggest and <h6> being the smallest.
   - <a> - This is the anchor tag, used to define a hyperlink. This can redirect the user to another web page.
   - <img> - This tag helps the user insert images in the web page.
   - <ul> - This tag is the unordered list tag, and it is used to list multiple items using plain bullets.

HTML Attributes

HTML Attributes are used to provide additional information about HTML elements. The attributes are always defined/specified within the start tags, and they usually come in name=value pairs. Ex- <a href = “ .....”>. Here, <a> is the HTML element as discussed above, and href is the attribute of the anchor tag.

HTML data -* attribute

The data-* attribute is used to store custom data that is private to the web page or application. Ex - <div data-name = "Suhas"> Username </div>.

CSS (CASCADING STYLE SHEETS)

CSS is used to add some style to the underlying page structure, like applying text color, changing fonts, giving background color, etc.

CSS syntax - selector {property : value}

1. Selector - it is used to find the HTML element to which styling is to be given.

   There are three types of selectors -

   a. Element Selector - This uses the HTML tags as a reference for styling. For ex - div {color : blue;}
   b. ID Selector - This selector uses the ID attribute of an HTML element to select a specific element to give styling to. To select an element with a specific ID, we write the hash character '#', followed by the ID of the element. It must also be noted HTML elements have unique IDs within a page. Ex - #suhas {color : red;}
   c. Class Selector - This selector uses the class attribute of an HTML element to select a specific element to give styling to. To select an element with a specific class, we write the dot character '.', followed by the class of the element. Ex - .suhas {color : blue;}

2. Property - This refers to the CSS property the user wants to apply. Ex - color, border, background, etc.
3. Value - This refers to the value assigned to said property. For example, the color property could have a value as red.

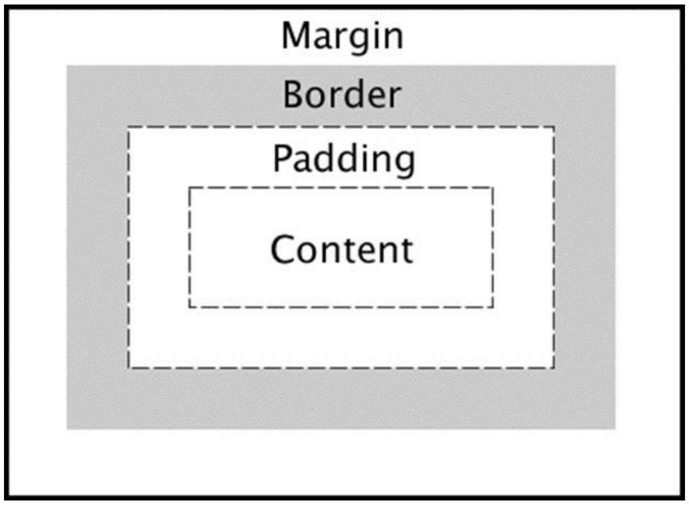
Types of CSS styles

There are broadly three types of CSS styles used.

1. Internal CSS (not used in Lightning Web Components)
2. Inline CSS - This is used to apply a unique style to a single HTML element by using its style attribute. For ex - <h1 style="color:blue;">A Blue Heading</h1>
3. External CSS or third party library - Here, the HTML refers to an external style sheet for changes in style. This external style sheet is referred to within the HTML document.

CSS Box Model

The CSS box model refers to a box that wraps around every HTML element. This box model consists of margins, borders, paddings, and the actual content of the HTML element. These constituents can be used to manipulate the overall style of these elements and by consequence, the web page itself. The box model is represented below -



JAVASCRIPT

Javascript is used to make the webpage interactive, for example - showing a popup, showing and hiding elements on click of a button, making server calls, etc. This language is used to implement complex features on a web page. In this section, we will examine concepts in JavaScript that are widely used in LWC.

Variables

These are containers for storing data. To declare a variable, we need to use one of these reserved keywords - var, const, and let, and to assign a value to a variable, we use the equals to '=' operator. For ex - var x = 50;

Data Types

There are 8 basic data types in JavaScript. They are

1. Number - stores all types of numbers ranging from integers to floating point numerals.
2. String - a sequence of 0 or more characters.
3. Boolean - can contain values true or false
4. bigInt - for integers of arbitrary length
5. Undefined - unassigned value
6. Null - unknown value
7. Objects - complex data structure
8. Symbol -

Note : The difference between 'null' datatype and 'undefined' data type is as follows. If a variable is declared but is not initialized or assigned any value, javaScript automatically initializes it with 'undefined'. On the other hand, when a variable is initialized to be empty, it is said to be of 'null'.

Spread Operator

The javaScript spread operator (...), has multiple uses, as listed below.

1. Expanding String - This converts a string into an array, whose elements consist of the individual characters present in the string.
2. Combining Arrays - This includes combining multiple arrays, or adding values to pre-existing arrays.
3. Combining Objects - This includes combining multiple objects, or adding values to pre-existing objects.
4. Creating a new shallow-copy of objects and arrays.

Object Destructuring

This is an operation that can be used on both objects and arrays. This enables the user to store the elements of an object or an array in separate variables.

String Methods

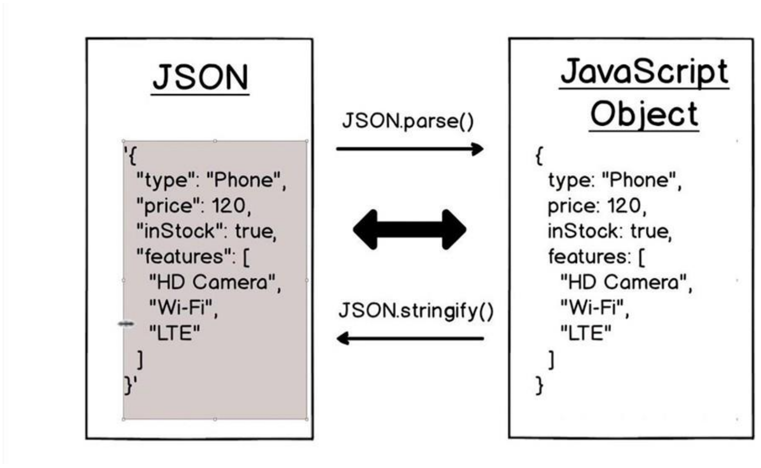
This refers to the multiple methods provided by JavaScript to manipulate strings. They are

1. includes() - This method returns 'true' if the string contains a specified value, and 'false' if it does not.
2. indexOf() - This method returns the first index (position) of a specified value in the string. If the value is not present in the string, the method returns '-1'.
3. startsWith() - This method returns 'true' if the string starts with a specified set of characters, and 'false' if it does not.
4. slice() - This method extracts part of a string and returns the extracted part in a new string.
5. toLowerCase() - This converts all characters in a string to lowercase.
6. toUpperCase() - This converts all characters in a string to uppercase.
7. trim() - This method removes whitespace from both sides of a string.

Object/JSON operations

There are mainly four operations that are important in LWC

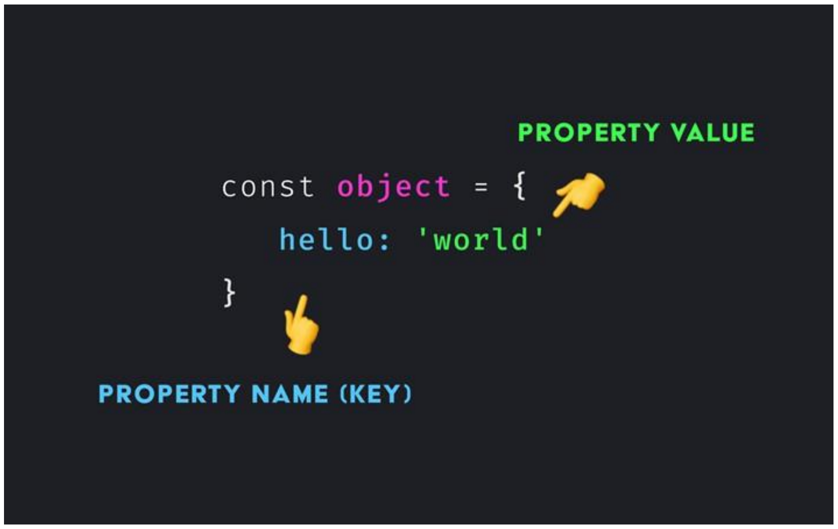
1. Object.keys() - This method extracts all the keys within an object and stores them in array format.
2. Object.values() - This method extracts all the values within an object and stores them in array format.
3. JSON.stringify() - This method converts an object to string format
4. JSON.parse() - This method converts the string of an object to an object.



```
'{
  "type": "Phone",
  "price": 120,
  "inStock": true,
  "features": [
    "HD Camera",
    "Wi-Fi",
    "LTE"
  ]
}'
```

```
{
  type: "Phone",
  price: 120,
  inStock: true,
  features: [
    "HD Camera",
    "Wi-Fi",
    "LTE"
  ]
}
```

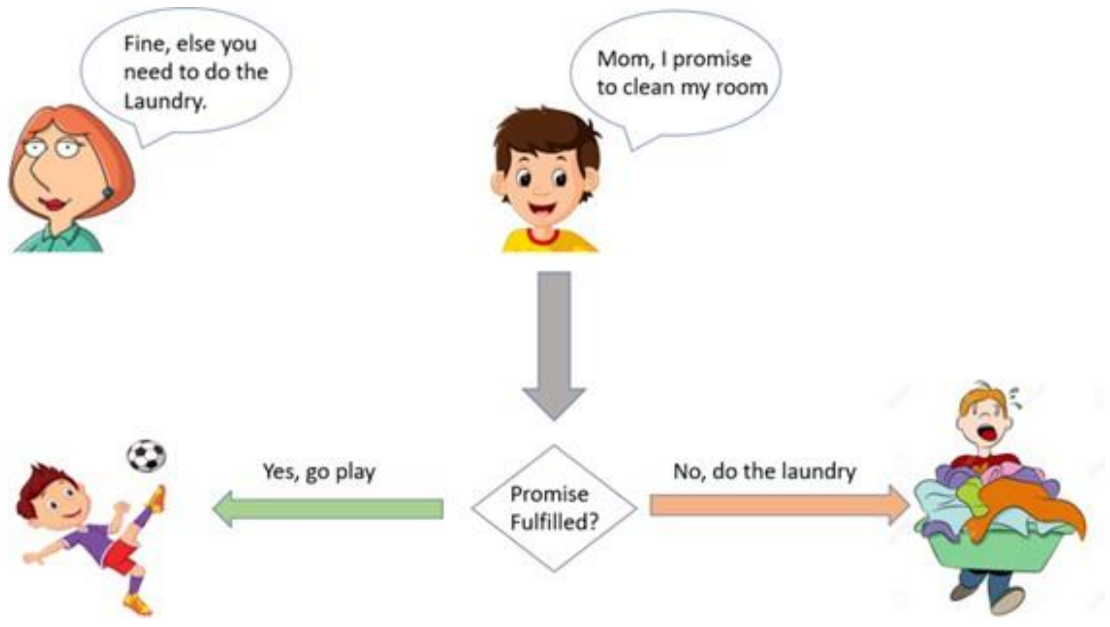
Format of a JavaScript Object

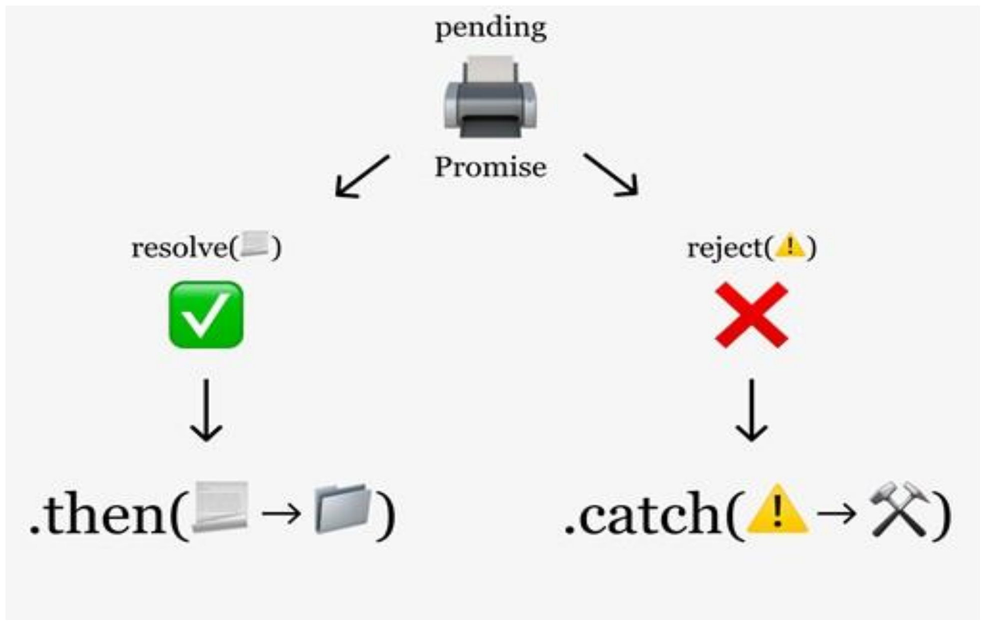


```
const object = {
    hello: 'world'
}
```

Promise

A promise in JavaScript is an object that may produce a single value sometime in the future. These are used to handle asynchronous operations in JavaScript (operations that depend on extraneous parameters like internet speed, file size, etc, and that may not render immediately). A promise can have three states - pending(), rejected(), and fulfilled().


Fine, else you need to do the Laundry.
Mom, I promise to clean my room
Yes, go play
Promise Fulfilled?
No, do the laundry



QuerySelector

This is used to select/return elements that match a specified CSS selector in the document.

1. querySelector() - returns the first element that matches a specified CSS selector in the document.

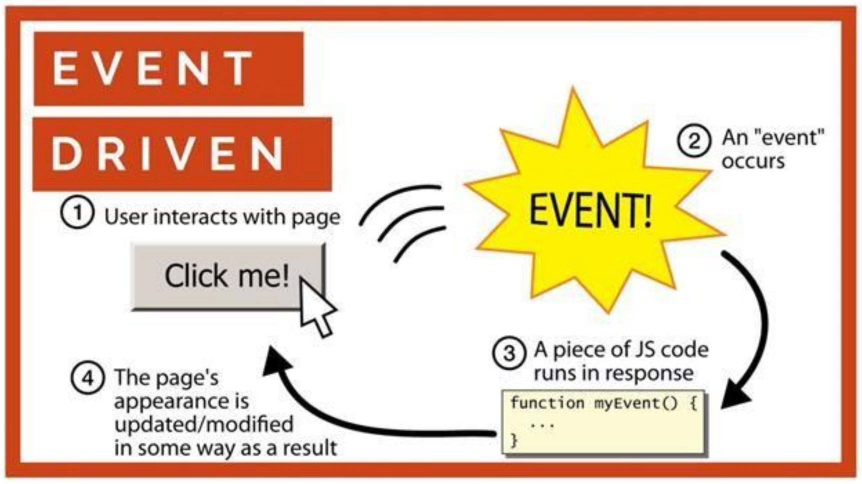
   Syntax - `document.querySelector(selector);`

2. querySelectorAll() - returns all elements that match a specified CSS selector in the document

   Syntax - `document.querySelectorAll(selector);`

Events

An event is an action that occurs in the web browser, which the web browser reflects to us, so we can respond to it. For example - When users click a button on a webpage, we may want to respond to this event by displaying an alert box. To respond to events, we use Event Handlers or Event Listeners - which are essentially blocks of code that execute when the event occurs.


EVENT
DRIVEN
1 User interacts with page
Click me!
EVENT!
2 An "event" occurs
3 A piece of JS code runs in response
function myEvent() {
...
}
4 The page's appearance is updated/modified in some way as a result

This brings us to the end of core web technology tools and parts of HTML, CSS, and Javascript that are important for LWC implementation.

LWC uses these web core technologies behind the scenes, and we had to learn these three technologies first to become proficient in LWC .

The tools needed for web development are -

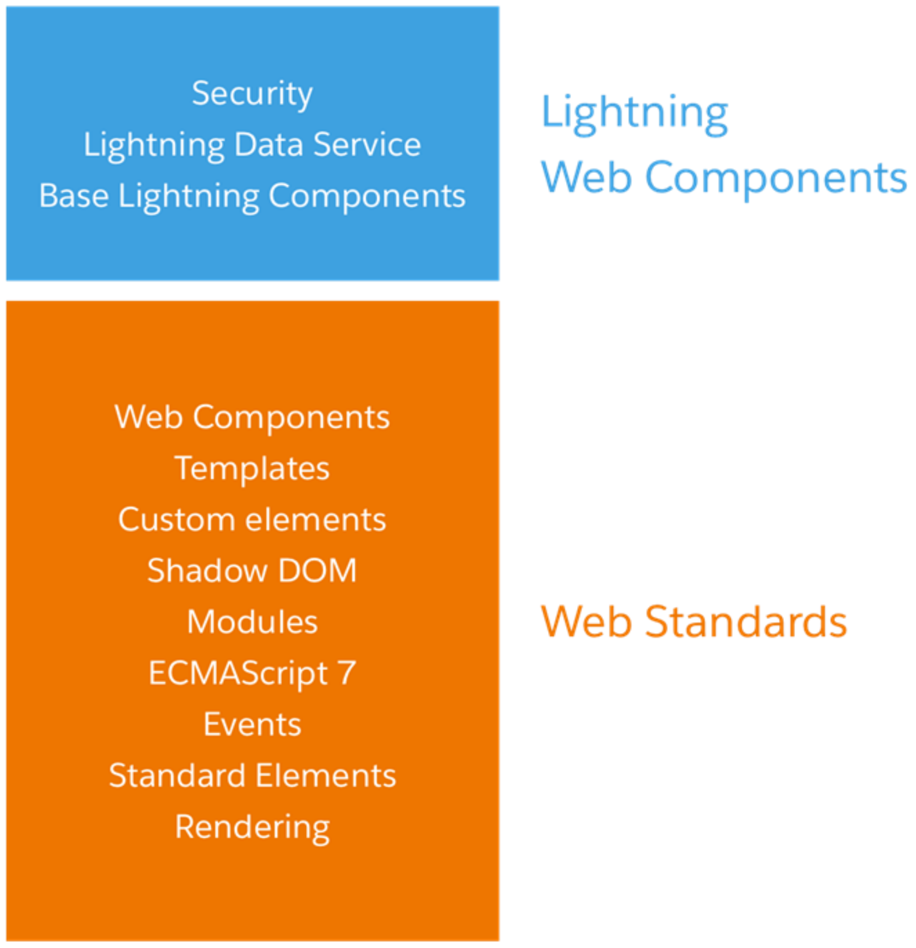
1. Browser - to view the output of HTML. Google Chrome is widely preferred.
2. Code Editor - we used VS Code
3. Extension - Live Server

LIGHTNING WEB COMPONENT

Introduction

The Lightning Component framework is a UI framework used to develop single page web applications for desktop and mobile devices. This framework uses two programming models - Aura Components Model and Lightning Web Components Model. We will be looking at the LWC model as that was the focus during the internship.

LWC is the new programming model introduced by Salesforce that is used for building Lightning Components on the web. This is built on core web standards as shown below.



Benefits of LWC

1. Lightweight framework - as most of the workload is carried by the browser web standards.
2. Better performance
3. Better security
4. Better testing frameworks
5. Interoperability with Aura Components

LWC FUNDAMENTALS

To create an LWC, we first install the Salesforce CLI extension in VS Code that enables LWC functionality in the editor. Once installed, we can create a new LWC. The component folder structure of an LWC consists of HTML file, JS file, meta configuration file to set the target for the component, and a CSS file (optional).

Naming conventions in LWC

1. camelCase - each word in the middle of the respective phrase begins with a capital letter. Ex - helloWorld

   Usage - naming the LWC

2. PascalCase - similar to camelCase, but the first letter is also capitalized. Ex- HelloWorld.

   Usage - naming the component class in the JavaScript file.

3. kebab-case - respective phrase is converted to all lowercase, with a hyphen(-) to separate words.

   Ex - hello-world

   Usage - referencing the component in the HTML file.

Steps for App Creation in LWC

1. First, we sign into our developer org on Salesforce platform. Then, we use Lightning App Builder to create a new app.
2. After that, we create a new page within our app, and select the appropriate page layout based on our requirements.
3. We then deploy the component created, to the developer org. This component could contain text, images and anything else.
4. We place the deployed component in the App Page.

Local Properties and Data Binding in LWC

In LWC, properties are variables that store certain values. Properties can store any data type. Now, there are scenarios where we need to interface between the front-end (HTML template) and the back-end (JavaScript controller) to display the values contained in the local properties on our web-page. For this, we use data binding- which is the synchronization between the controller and the HTML template. In the below example, data binding enables us to display the value contained in the property 'fullname', in the template.

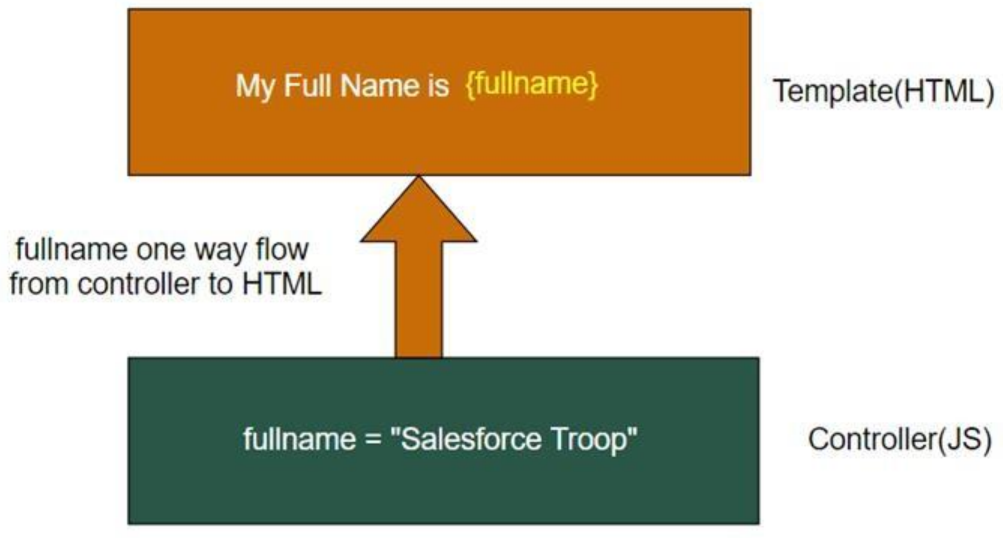
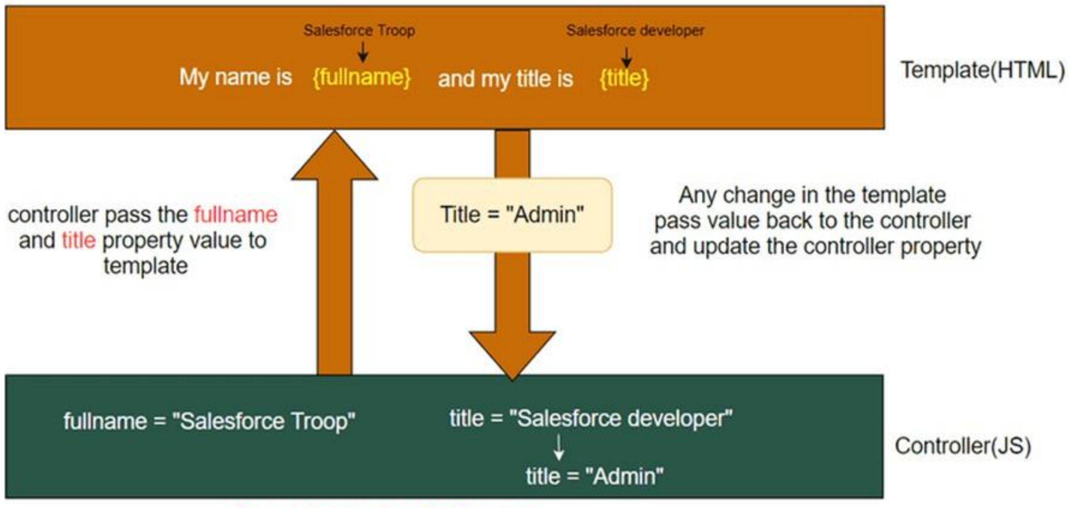

My Full Name is {fullname}
Template(HTML)
fullname one way flow from controller to HTML
fullname = "Salesforce Troop"
Controller(JS)

Fig 1.1 One Way Data Binding

Methods and two-way Data Binding in LWC

In LWC, methods are functions that compute logic in JavaScript. In one-way data binding, we saw how we can pass data from the controller to the template. But, we can also work the other way round and pass data from the template to the controller. How? One way to do this is by event listeners in the template (as discussed above), that can set the value of a property in the controller. In the example below, we could have an input field for the title in the template, call an event listener, and within the function, store the user input in the local property ‘title’. Hence we can pass data both ways.


Salesforce Troop
Salesforce developer
My name is {fullname} and my title is {title}
Template(HTML)
controller pass the fullname and title property value to template
Title = "Admin"
Any change in the template pass value back to the controller and update the controller property
fullname = "Salesforce Troop"
title = "Salesforce developer"
title = "Admin"
Controller(JS)
Fig 1.1 Two-Way Data Binding

@track property in LWC

When a property contains an object or an array, there is a limit to the depth of changes that can be tracked . We decorate said property with @track to tell the framework to observe changes to an object or an array.

LWC directives in HTML

Directives are special HTML attributes. The LWC programming model has a few custom directives that let us manipulate the front-end using the HTML markup. Two special LWC directives for conditional rendering are -

1. if:true - this is used to render elements in a template, if the specified expression is true.

2. if:false - this is used to render elements in a template, if the specified expression is false.

Template looping

This is used in scenarios where we have to render the same set of elements, with mostly the same styling, but with different data in the HTML. The types of template looping are -

1. for:Each - this takes array as the input and renders each element of said array.
2. Iterator - this loop is applied on the data we want to iterate through.

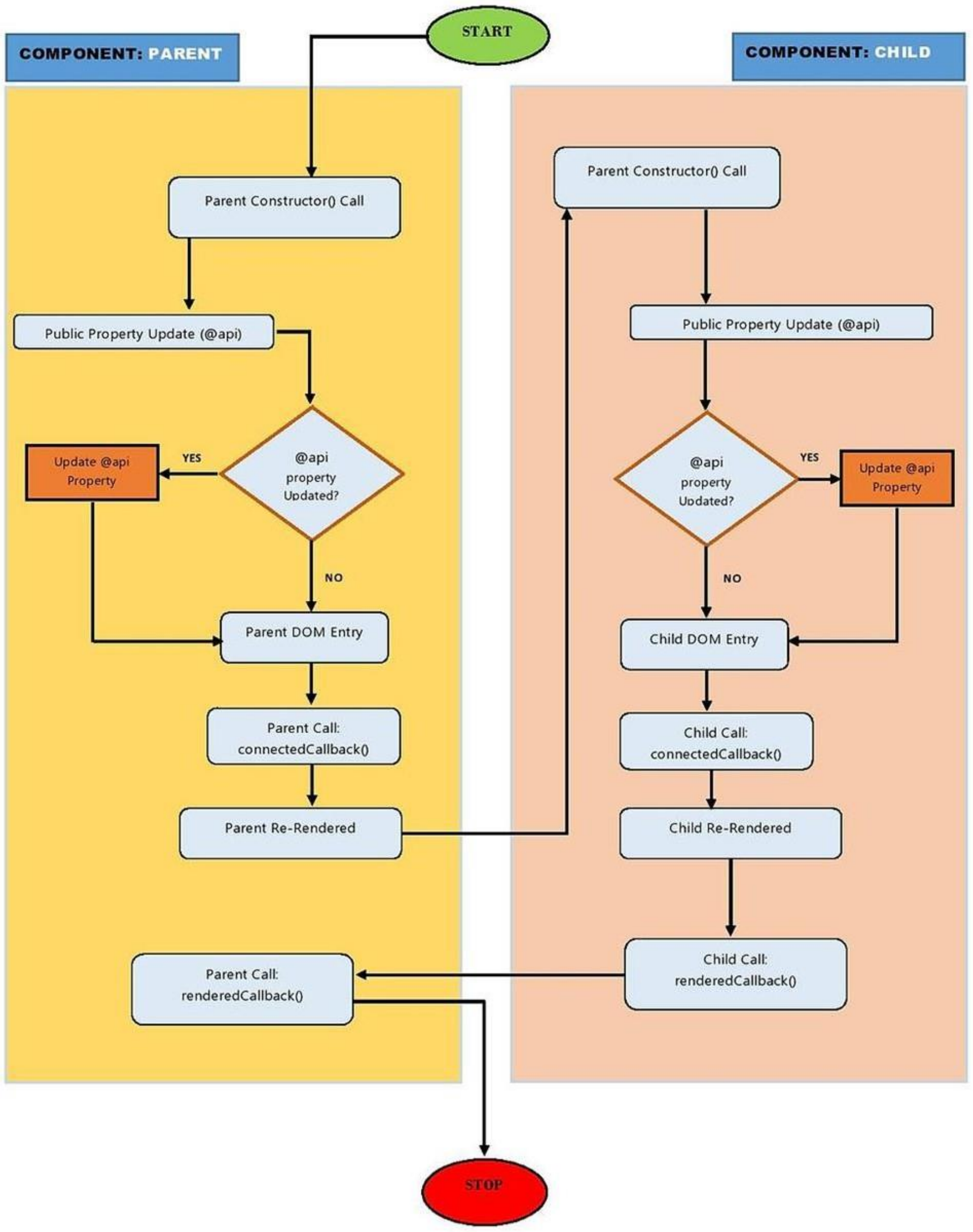
Styling in LWC

To provide the markup some style, we use the following ways to do the same:

1. Inline and External CSS - this method resembles what we discussed above in the CSS section - where we apply CSS properties to selectors, either inline (inside the HTML markup), or in an external CSS file that the HTML file refers to.
2. Salesforce Lightning Design System (SLDS) - This is a design library that Salesforce has created. It enables us to create UI that is consistent with Salesforce design principles. We can search for the component and the styling we need, and copy the classes to our markup.
3. SLDS Design Token - Design Tokens are style values of UI elements (ex - color, padding, font, etc), and they are the building blocks of any design system. Design tokens are used in place of hardcoded values, and thus are more flexible when design needs to be changed.
4. Shared CSS - sometimes, we may not want to create multiple CSS files for multiple LWCs. In this instance, we can create a separate style component with a stylesheet containing all style specifications. And we can import this style sheet wherever we are using it, thus having greater code reusability.

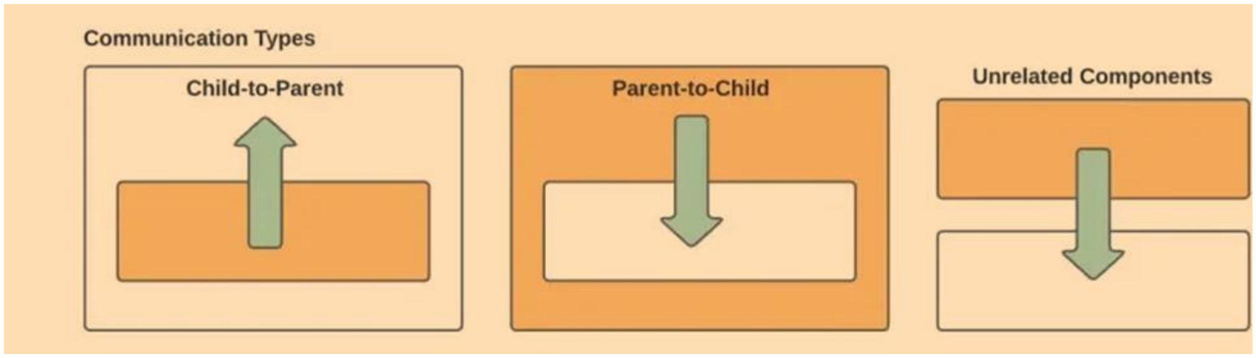
Component Lifecycle Hooks in LWC

Lifecycle hooks are callback methods that are triggered at specific phases of the lifecycle of the component instance.


START
COMPONENT: PARENT
COMPONENT: CHILD
Parent Constructor() Call
Public Property Update (@api)
@api property Updated?
YES
Update @api Property
NO
Parent DOM Entry
Parent Call: connectedCallback()
Parent Re-Rendered
Parent Constructor() Call
Public Property Update (@api)
@api property Updated?
YES
Update @api Property
NO
Child DOM Entry
Child Call: connectedCallback()
Child Re-Rendered
Child Call: renderedCallback()
Parent Call: renderedCallback()
STOP

1. constructor() - this is called on component creation. It flows from parent component to child component (it is called in the parent component first).
2. connectedCallback() - this is called after the element is inserted into a document, and it flows from parent to child. This hook is mainly used to perform initialization tasks.
3. renderedCallback() - this is called each time the component renders, and the hook flows from child to parent. This hook is used to interact with the component UI.
4. render() - this is called to update the UI of the component, mainly to render templates conditionally.
5. disconnectedCallback() - this is called when the element is removed from the document, and the hook flows from parent to child.
6. errorCallback() - called when the component throws an error.

Component Communication in LWC



Parent to child

This communication happens when the parent component contains the tag of its child component in the HTML file, and passes data to the child component. To do this, we use the @api decorator to define a variable in the child component, that is a public property and can be accessed by the parent. We then assign the value to this property from the parent by assigning the value as an attribute in the tag for the child component.

Child to Parent

We use a custom Event here, to pass data from the child to parent. We define a custom event in the child component and pass data inside this event in the form of an object. We then create an event listener in the parent component that listens to this custom event that is passed from the child. When the parent successfully listens to this event, it also retrieves the data being passed to it from the child component.

Unrelated components

We use the Lightning Message Service to communicate across the DOM between unrelated LWCs. We use a Lightning Message Channel to pass data from one component to another.

COMMUNICATION BETWEEN LWC, AURA COMPONENTS AND VISUALFORCE

Communication between unrelated LWCs, Aura components and Visualforce pages across DOM with another LWC, Aura component, or Visualforce pages is carried out using Lightning Message Services.

Lightning Message Services

To communicate across the DOM between Visualforce pages, Aura components, and Lightning web components, including components in a pop-out utility bar, use the Lightning messaging service.
To communicate over a Lightning message channel, use the Lightning message service functions. Import any exports from the lightning message Service module into a component's Javascript code. Using the scoped module, import a message channel using `@salesforce/messageChannel`.

- ➔ Steps to communicate using message channels.

1. Make a Channel for Messages.

Use the LightningMessageChannel metadata type to build a Lightning message channel.

2. Define the Message Service's Scope

   The Lightning message service allows you to choose the scope of where subscribing components in your application receive messages. You can either limit the scope to the application's active area or set it to the full application.

3. Use a Message Channel to publish your message

   Include the @salesforce/messageChannel scoped module in your component's JavaScript file and utilise the Lightning message service's publish() function to publish messages on a message channel from a Lightning web component.

4. Subscribe to a Message Channel and Unsubscribe

   Import the message channel from the @salesforce/messageChannel scoped module into your Lightning web component to subscribe and unsubscribe from messages on a message channel. Use the subscribe() and unsubscribe() functions of the Lightning message service.

5. Limitations of the Lightning Message Service

   When using the Lightning message service, keep the following in mind.

Various methods & wire adapters used:

1. createMessageContext()

   The MessageContext object is returned.

   In a service component that does not extend LightningElement, call this function. You can't use @wire(MessageContext) to build a MessageContext object in a service component. Instead, use the createMessageContext() function to generate a MessageContext object and assign it to a field, such as messageContext. Then, in the subscribe() function, pass messageContext. For service components, MessageContext is not automatically released. To remove any subscriptions associated with your Lightning web component's MessageContext, call releaseMessageContext(messageContext).

   When using the Lightning message service, keep the following in mind.

2. publish(messageContext, messageChannel, message)

   Publishes a message to a message channel that has been specified.

- messageContext - The MessageContext object describes the Lightning web component that uses the Lightning message service. This object can be obtained using the MessageContext wire adapter or the createMessageContext command ().
- messageChannel - The object that represents the message channel. Use the scoped module @salesforce/messageChannel to import a message channel. Use the LightningMessageChannel metadata type to construct a message channel in an org.
- Message - The message sent to subscribers is stored in a serializable JSON object. Functions and symbols are not allowed in a message.

3. releaseMessageContext(messageContext)

   Unsubscribes all associated subscriptions and releases a MessageContext object associated with a Lightning web component.

4. subscribe(messageContext, messageChannel, listener, subscriberOptions)

   Subscribes to a message channel that you specify. Unsubscribes you by returning a Subscription object.

   By default, communication over a message channel can occur only between Lightning web components, Aura components, or Visualforce pages in an active navigation tab, an active navigation item, or a utility item. Items that are used frequently are always in use. When you choose a navigation tab or item, it's active. The following are some of the tabs and things that can be found in the navigation menu:

- Standard navigation tabs
- Console navigation workspace tabs
- Console navigation subtabs
- Console navigation items

Pass the subscriberOptions parameter as scope: APPLICATION SCOPE to receive messages on a message channel from anywhere in the application. APPLICATION SCOPE should be imported from lightning/messageService.

  - messageContext - The MessageContext object describes the Lightning web component that uses the Lightning message service.
  - messageChannel - Use the scoped module @salesforce/messageChannel to import a message channel. Use the LightningMessageChannel metadata type to construct a message channel in an org.
  - Listener - Once the message has been published, this function takes care of it.
  - subscriberOptions - (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. APPLICATION SCOPE should be imported from lightning/messageService

5. unsubscribe(subscription)

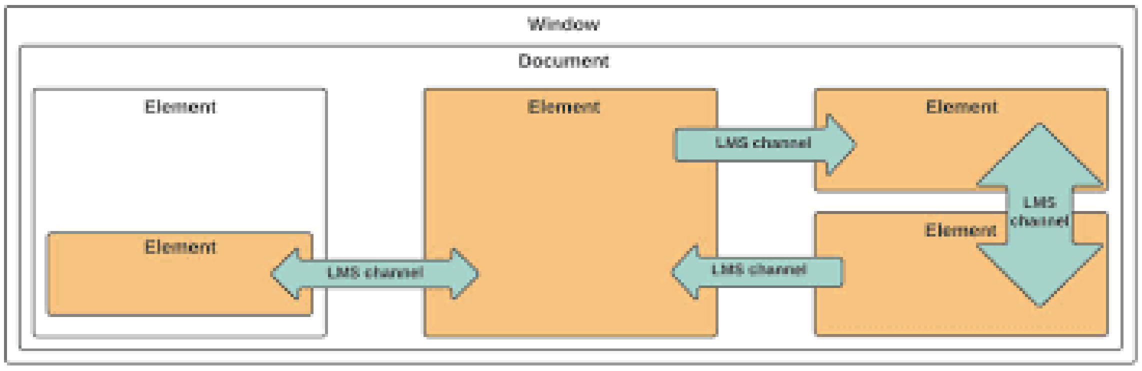
   Unsubscribes from a message channel.

6. MessageContext

   This is a wire adapter.

   A MessageContext object is returned.

   The MessageContext object holds details about the Lightning web component that uses the Lightning message service. The publish() and subscribe() operations both require the MessageContext object. You don't have to interact with any of the component's lifecycle events when using the @wire(MessageContext) adapter. When a component is destroyed, the Lightning messaging service features immediately unregister.



AURA COEXISTENCE

Lightning web components and Aura components can work together in an app thanks to an interoperability layer.

You can create new Lightning web components and include them in apps with Aura components. Alternatively, you can migrate your app in stages by replacing individual Aura components with Lightning web components.

Although Lightning web components and Aura components can be used together, there are some limitations to be aware of.

NOTE: Lightning web components may be found in Aura components. The opposite, however, is not true. Aura components cannot be used with Lightning web components.

1. Using Lightning Web Components, create Aura Components.

   Aura components can be created using Lightning web components, but not the other way around. Parents assign properties to their children in order to communicate down the hierarchy. It's important to understand facets and slots before deciding when and how to nest a Lightning web component in an Aura component.
2. Send Events to an Aura Component That Is Enclosing

   Dispatcher an event to convey from the youngster to the parent. DOM events are fired by Lightning web components. An enclosing Aura component, like an enclosing Lightning web component, can listen for these events. The event can be captured and handled by the Aura component that surrounds it. To communicate with other Aura components or the app container, the Aura component can optionally fire an Aura event.
3. LWC and Aura can share JavaScript code

   Put JavaScript code in an ES6 module in a service component to share it between Lightning web components and Aura components.

SALESFORCE RESOURCE, COMPONENT CONTEXT AND NOTIFICATION

1. Fetching images from static resources

Import static resources from the scoped module `@salesforce/resourceUrl`. Archive files (such as.zip and.jar files), images, style sheets, JavaScript, and other files are examples of static resources.

- myResource—This is the name of the static resource.

The name of the static resource is referenced by resourceReference.

Only underscores and alphanumeric characters are allowed in a static resource name, and it must be unique within your organisation. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

If the static resource is part of a managed package, this value is the managed package's namespace.

An archive file with a nested directory structure can be used as a static resource. To reference an item in an archive, concatenate a string to create the path to the item, as the example does to build einsteinUrl.

To reference a resource in a template, use {property} syntax, which is the same syntax you use to reference any JavaScript property.

2. Getting used to third party javascript libraries in LWC

With Lightning web components, you may use third-party JavaScript libraries. Use a library with interactive charts and graphs, or one that reduces code complexity.

- The library can be downloaded from the third-party library's website.
- Upload the library to your Salesforce organisation as a static resource, which is a Lightning Web Components content security policy requirement.
- Load the library and call its functions in a then() method in a JavaScript class that extends LightningElement.

3. Getting used to third party CSS libraries in LWC

Use the Lightning Design System to give your component the look and feel of the Lightning Experience. Write your own CSS to go your own way.

- Lightning Design System Style Components

  Salesforce Lightning Design System (SLDS) is a CSS framework that provides a look and feel that's consistent with Lightning Experience. Use SLDS styles to give your custom Lightning web components a UI that is consistent with Salesforce, without having to reverse-engineer our styles. And best of all, it just works with Lightning components running in Lightning Experience and in the Salesforce mobile application.
- Design Variations for Base Lightning Components

  Many standard Lightning components come in a variety of designs. Each design variant provides a component a unique appearance.
- Components of Style Styling Hooks from the Lightning Design System

  Salesforce Lightning Design System (SLDS) styling hooks make it easy to customise component styling and express your brand. CSS custom properties that correspond to SLDS component blueprints and design variations are provided by styling hooks. Declare a corresponding custom property in a component's style sheet to customise an SLDS style. Custom properties can be used to style both base and custom components.
- From an SLDS Blueprint, create a Component.

  If you can't find a base component for your use case, find the closest SLDS blueprint to help you build your own custom component.
- Tokens from the Lightning Design System can be used.

  Design tokens are named entities that store visual design attributes, such as margins and spacing values, font sizes and families, or hex values for colours.
- Use Aura Tokens that are unique to you.

  A Lightning web component's CSS file can use a custom Aura token created in your org or installed from an unmanaged package.

- Create a Component's CSS Style Sheet
  Create a stylesheet in the component's folder to bundle styles with it. The style sheet and the component must have the same name. The style sheet is automatically applied to the component.
- Make Hooks for Styling Your Components
  Use CSS custom properties to expose styling hooks for your custom components. CSS custom attributes also make code more readable and maintainable.
- CSS Style Rules to Share
  Using a common CSS module, give Lightning web components a consistent look and feel. Styles are defined in the CSS module, which is then imported into the components with which you want to share styles.

4. Content asset files

Import content asset files from the scoped module `@salesforce/contentAssetUrl`. To use a Salesforce file in custom apps and Experience Builder templates, convert it to a content asset file.

```
import myContentAsset from
'@salesforce/contentAssetUrl/contentAssetReference';
import myContentAsset from
'@salesforce/contentAssetUrl/namespace
contentAssetReference';
```

- myContentAsset—A name that refers to the asset file.
- contentAssetReference—The asset file's name.

Only underscores and alphanumeric characters are allowed in asset file names, and they must be unique within your organisation. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

If the asset file is part of a managed package, this value is the managed package's namespace.

5. Internationalisation

Import properties for internationalisation from the `@salesforce/i18n` scoped module. Lightning web components have internationalisation properties that you can use to adapt your components for users worldwide, across languages, currencies, and timezones.

In a single currency organisation, Salesforce administrators set the currency locale, default language, default locale, and default time zone for their organisations. On their personal settings pages, users can customise their language, locale, and time zone.

Base Lightning components adapt automatically to the language, locale, and time zone settings of the Salesforce org they run in. Use the internationalisation properties to internationalise your components so that they adapt as well.

- internationalizationPropertyName identifies the internationalisation property by its name.
- internationalizationProperty is a property for internationalisation.

6. Access labels

Labels from the `@salesforce/label` scoped module can be imported. Custom labels are text values in Salesforce that can be translated into any language supported by Salesforce. Use custom labels to create multilingual applications that present information (for example, help text or error messages) in a user's native language.

- labelName—An identifier for the label.
- labelReference—In the format namespace, the name of the label in your org.

We use this format because it's the same format used in managed packages, in Visualforce, and in other Salesforce technologies. Regardless of where you're accessing labels, you can use the same format, myns.labelName.

To use the labels in the template, use the same {property} syntax that you use to reference any JavaScript property.

Label files can be found in any subdirectory of `force-app/main/default` in a Salesforce DX project.

7. Check permissions

Import Salesforce permissions from the scoped modules `@salesforce/customPermission` and `@salesforce/userPermission`. Customise a component's behaviour based on the context user's rights.

To check whether a user has a permission, import a static reference to the permission and evaluate whether it’s true or undefined.

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

A namespace can be included in custom permissions. For their own customization and packages, orgs employ namespaces as unique identifiers. Prefix the namespace followed by _ to the custom permission name if it was installed from a managed package.

```
import hasPermission from
'@salesforce/customPermission/PermissionName';
import hasPermission from
'@salesforce/customPermission/namespace PermissionName';
```

The name of the static reference is your choice. To signal that the reference contains a boolean, we used the format hasPermission.

8. Access clients from factors

Import the `@salesforce/client/formFactor` scoped module to get the form factor of the hardware the browser is running on.

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

- \sformFactorPropertyName—A name that refers to the form factor of the hardware running the browser. The following are examples of possible values:
  - ➔ Large—A desktop client.
  - ➔ Medium—A tablet client.
  - ➔ Small—A phone client.

Pass the form factor to the getRecordCreateDefaults wire adapter to get the default layout and object information for creating a record.

```
import { getRecordCreateDefaults } from
'lightning/uiRecordApi';
import ACCOUNT OBJECT from '@salesforce/schema/Account';
import FORM FACTOR from '@salesforce/client/formFactor';

accountDefaults;
@wire(getRecordCreateDefaults,
objectApiName: ACCOUNT OBJECT,
formFactor: FORM FACTOR)
```

9. Get information about current users

Import data from the `@salesforce/community` scoped module about the current Experience Builder site.

The supported properties are:

```
import propertyName from '@salesforce/community/property';
```

- Id—The current site's identifier. Import the ID, for example, to use as a parameter in an API.
- propertyName—A name for the Experience Builder property that was imported.
- basePath—The base path is the section of the URL that follows the domain. So if your site domain name is UniversalTelco.force.com and myPartnerSite was the URL value added when you created the site, the community’s URL is UniversalTelco.force.com/myPartnerSite/s. The base path in this situation is myPartnerSite/s.

10. Toast notifications

A component can deliver a toast notice that appears when a success, error, or warning occurs. A toast can also serve as a source of knowledge. Import ShowToastEvent from the lightning/platformShowToastEvent module to display a toast notification in Lightning Experience or Experience Builder sites.

The lwc-recipes repo has a component that allows you to customise and send a toast so you can experiment with the different options.

ShowToastEvent can be imported directly from lightning/platformShowToastEvent. Create a ShowToastEvent event with title, message, variant, and mode properties and send it out. Because the default value for mode is used in this example, it is not provided.