LDS: Lightning Record Edit Form

Create Record using lightning-record-edit form base Component
- Why to use Lightning record edit form to create the record
  If we want to customise form layout or predefined fields we can use lightning edit form

Create a Record
Lightning-record-edit-form
- customise Form Layout, Predefined field values
- customise rendering of data
- Editable fields => lightning input field
- Read-only fields => lighting -output-field
- Handles field level validation error and LDS error automatically.
- Lighting- message to display error messages
- Doesn't provide own Save and Cancel buttons
- To Reset form fields use reset()
- Event : error, load , submit , success

Implement LDS, Not Required Apex to create , take care FLS and sharing.

Sample code to create record using lightning record edit form

```
<template>
    <div class="slds-box" style=background-color:lightsteelblue>
        <lightning-record-edit-form object-api-name={caseObject} onsuccess=
{handleSuccess}>
            <lightning-messages></lightning-messages>
           <div class="slds-grid slds-wrap slds-gutters">
                <div class="slds-col slds-size_1-of-3 slds-p-around_small">
                    <lightning-input-field field-name={statusField}> </lightning-
input-field>
                </div>
                <div class="slds-col slds-size_1-of-3 slds-p-around_small">
                    <lightning-input-field field-name={originFiled}> </lightning-
input-field>
                </div>
                <div class="slds-col slds-size_1-of-3 slds-p-around_small">
                    <lightning-input-field field-name={reasonFiled}> </lightning-
input-field>
                </div>
            </div>
                <div class="slds-grid">
                    <div class="slds-col">
                        <lightning-input-field field-name={addiotnalField} value=
{origialAddtional}> </lightning-input-field>
                        <lightning-input-field field-name={subjectField}>
</lightning-input-field>
                        <lightning-input-field field-name={descriptionField}>
</lightning-input-field>
                    </div>
                </div>
                <div class="slds-align_absolute-center">
                    <lightning-button type="cancel" class="slds-p-around_small"
variant="brand" label="Cancel" onclick={handleCancel}> </lightning-button>
                    <lightning-button type="submit" variant="brand"
label="Create Case"> </lightning-button>

                </div>
        </lightning-record-edit-form>
    </div>

</template>
```

JS File :

```
import { LightningElement } from 'lwc';
import  CASE_OBJECT from "@salesforce/schema/Case";
import CASENUMBER_FIELD from "@salesforce/schema/Case.CaseNumber";
import STATUS_FIELD from "@salesforce/schema/Case.Status";
import ORIIGIN_FIELD from "@salesforce/schema/Case.Origin";
import REASON_FIELD from "@salesforce/schema/Case.Reason";
import DESCRIPTION_FIELD from "@salesforce/schema/Case.Description";
import ADDTIONAL_FIELD from "@salesforce/schema/Case.Additional_info__c";
import SUBJECT__FIELD from "@salesforce/schema/Case.Subject";
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
export default class CreateCaseRecordWithEditForm extends LightningElement {
    caseObject = CASE_OBJECT;
    statusField = STATUS_FIELD;
    originFiled = ORIIGIN_FIELD;
    reasonFiled = REASON_FIELD;
    subjectField = SUBJECT__FIELD;
    descriptionField = DESCRIPTION_FIELD;
    addiotnalField = ADDTIONAL_FIELD;
    origialAddtional = 'Test';
    handleSuccess(event){
        console.log(event.detail.id);
        const evt = new ShowToastEvent({
            title : 'Case Created',
            message: 'Record Id'+event.detail.id,
            variant:'Success'
        });
        this.dispatchEvent(evt);
    }
    handleCancel(event){
        const inputFileds = this.template.querySelectorAll('lightning-input-
field');
        if(inputFileds){
            inputFileds.forEach(field => {
                field.reset();
            });
        }
    }
}
```