56. Retrieve Records, Button Icon, Button Disabled, Input Field Read Only,

Spinners, Toast Message, Conditional Statements - 23 June 2022

Today we will learn:

1] Retrieve Records

2] Toast Message

3] Spinners

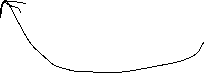
4] HTML Template Conditional Statement

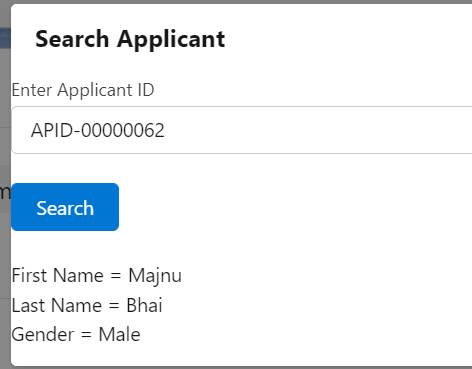
5] Button Icons

6] Enable / Disable Buttons

7] Input Field – Read Only / Disabled/ Editable

1] Retrieve Records





<template>

    <lightning-card title="Search Applicant">

        <lightning-input  data-formfield="applicantID"  type="text" label="Enter Applicant ID" name="appID"> </lightning-input> <br />

        <lightning-button variant="brand" label="Search" onclick={searchRecord}></lightning-button> <br/><br/>

        First Name = {objApplicant.First\_Name\_\_c} <br />

        Last Name = {objApplicant.Last\_Name\_\_c} <br />

        Gender = {objApplicant.Gender\_\_c}

    </lightning-card>

</template>

import { LightningElement } from 'lwc';

import searchApplicantRecord from '@salesforce/apex/ApplicantProvider.searchApplicantRecord';

export default class ApplicantSearch extends LightningElement {

    objApplicant = { 'sObjectType' : 'Applicant\_\_c' };

    searchRecord(){

      this.objApplicant.Name =  this.template.querySelector('lightning-input[data-formfield="applicantID"]').value;

      //Calling searchApplicantRecord

      searchApplicantRecord({objApp : this.objApplicant})

      .then((result) =>{

        console.log('Result = '+ JSON.stringify(result) );

        this.objApplicant = result;

      })

      .catch((error) =>{

        console.log('error = '+error);

      });

    }

}

public with sharing class ApplicantProvider {

   @AuraEnabled

   public static Applicant\_\_c searchApplicantRecord(Applicant\_\_c objApp){

    try {

            if(objApp != null){

             return  ([select Id, First\_Name\_\_c, Last\_Name\_\_c, Gender\_\_c from Applicant\_\_c where Name=: objApp.Name LIMIT 1] );

            }

            else{

                return null;

            }

       } catch (Exception e) {

           throw new AuraHandledException(e.getMessage());

       }

   }

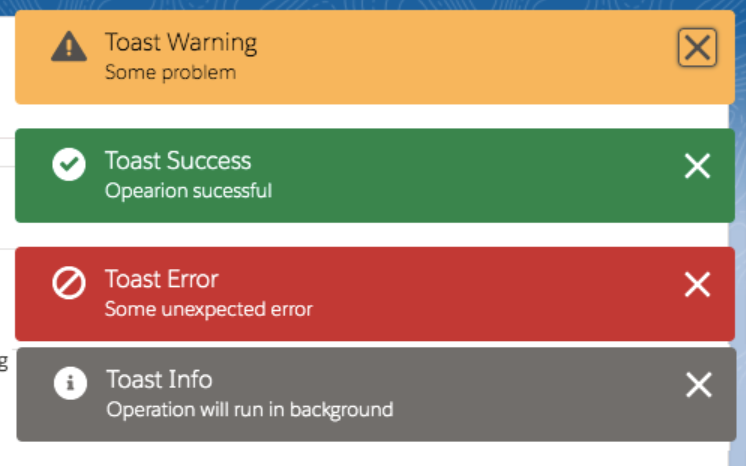
}

2] Toast Message

JS Script Alert:

alert("Record Found...!!!");

Toast Message



import { ShowToastEvent } from 'lightning/platformShowToastEvent' ;

showSuccessToast() {

const evt = new ShowToastEvent({

title: 'Message',

message: this.result,

variant: **'success'**,

mode: 'dismissable'

});

this.dispatchEvent(evt);

}

|  |  |
| --- | --- |
| Success (Green) | success |
| Information (Yellow) | info |
| Warning (Orange) | warning |
| Error (Red) | error |

**JS Script:**

import { LightningElement } from 'lwc';

import { ShowToastEvent } from 'lightning/platformShowToastEvent' ;

import searchApplicantRecord from '@salesforce/apex/ApplicantProvider.searchApplicantRecord';

export default class ApplicantSearch extends LightningElement {

    objApplicant = { 'sObjectType' : 'Applicant\_\_c' };

    searchRecord(){

      this.objApplicant.Name =  this.template.querySelector('lightning-input[data-formfield="applicantID"]').value;

      //Calling searchApplicantRecord

      searchApplicantRecord({objApp : this.objApplicant})

      .then((result) =>{

        console.log('Result = '+ JSON.stringify(result) );

        this.objApplicant = result;

        this.showSuccessToast("Record Found", 'success');

      })

      .catch((error) =>{

        console.log('error = '+error);

        this.showSuccessToast("Record Not Found", 'error');

      });

    }

    showSuccessToast(message, variantName) {

        const evt = new ShowToastEvent({

            title: 'Message',

            message: message,

            variant: variantName,

            mode: 'dismissable'

        });

        this.dispatchEvent(evt);

    }

    /\*

    showErrorToast() {

        const evt = new ShowToastEvent({

            title: 'Message',

            message: 'Record Not Found...!!!!',

            variant: 'error',

            mode: 'dismissable'

        });

        this.dispatchEvent(evt);

    }

    \*/

}

**3] Spinners**

<div class="exampleHolder">

<lightning-spinner alternative-text="Loading" size="medium"></lightning-spinner>

</div>

4] HTML Template Conditional Statement

<template if:true={value}>

         My Name is Lakhan..!!!

</template>

<template if:false={value}>

         Pushpa / Pushparaj hai mai…!!!! Fire

</template>

**5] Button Icons**

<https://www.lightningdesignsystem.com/icons/>

<lightning-button variant="brand" icon-name="utility:search" label="Search" onclick={searchRecord}></lightning-button> <br/><br/>

<template>

    <lightning-card title="Search Applicant">

        <lightning-input  data-formfield="applicantID"  type="text" label="Enter Applicant ID" name="appID"> </lightning-input> <br />

        <lightning-button variant="brand" icon-name="utility:search" label="Search" onclick={searchRecord}></lightning-button> <br/><br/>

        <template if:true={showInfoFlag}>

        First Name = {objApplicant.First\_Name\_\_c} <br />

        Last Name = {objApplicant.Last\_Name\_\_c} <br />

        Gender = {objApplicant.Gender\_\_c} <br/>

   </template>

   <template if:true={showSpinnerFlag}>

            <div class="exampleHolder">

                <lightning-spinner alternative-text="Loading" size="medium"></lightning-spinner>

            </div>

   </template>

   <template if:true={showNameFlag}>

    My Name is Lakhan..!!!

</template>

<template if:false={showNameFlag}>

    Pushpa / Pushpa sunke flower samza kya, Fire hai mei...!!!!

</template>

    </lightning-card>

</template>

import { LightningElement } from 'lwc';

import { ShowToastEvent } from 'lightning/platformShowToastEvent' ;

import searchApplicantRecord from '@salesforce/apex/ApplicantProvider.searchApplicantRecord';

export default class ApplicantSearch extends LightningElement {

    objApplicant = { 'sObjectType' : 'Applicant\_\_c' };

    showSpinnerFlag = false;

    showNameFlag = false;

    showInfoFlag = false;

    searchRecord(){

       this.showSpinnerFlag = true;

      this.objApplicant.Name =  this.template.querySelector('lightning-input[data-formfield="applicantID"]').value;

      //Calling searchApplicantRecord

      searchApplicantRecord({objApp : this.objApplicant})

      .then((result) =>{

        console.log('Result = '+ JSON.stringify(result) );

        this.objApplicant = result;

        this.showSuccessToast("Record Found", 'success');

        this.showInfoFlag = true;

        this.showSpinnerFlag = false;

      })

      .catch((error) =>{

        console.log('error = '+error);

        this.showSuccessToast("Record Not Found", 'error');

        this.showSpinnerFlag = false;

        this.showInfoFlag = false;

      });

    }

    showSuccessToast(message, variantName) {

        const evt = new ShowToastEvent({

            title: 'Message',

            message: message,

            variant: variantName,

            mode: 'dismissable'

        });

        this.dispatchEvent(evt);

    }

    /\*

    showErrorToast() {

        const evt = new ShowToastEvent({

            title: 'Message',

            message: 'Record Not Found...!!!!',

            variant: 'error',

            mode: 'dismissable'

        });

        this.dispatchEvent(evt);

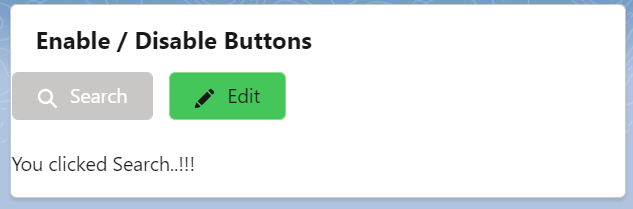
    }

    \*/

}

=============================================

**6] Enable / Disable Buttons**



<template>

    <lightning-card title="Enable / Disable Buttons">

        <lightning-button variant="brand" icon-name="utility:search" label="Search" onclick={searchMethod} disabled={disableSearchButtonFlag}  ></lightning-button> &nbsp;&nbsp;

        <lightning-button variant="success" icon-name="utility:edit" label="Edit" onclick={editMethod}   disabled={disableEditButtonFlag}></lightning-button> <br /> <br />

        <template if:true={showSearchMessageFlag}>

            You clicked Search..!!!

        </template>

        <template if:true={showEditMessageFlag}>

            You Clicked Edit..!!!!

        </template>

    </lightning-card>

</template>

import { LightningElement } from 'lwc';

export default class EnableDisableButton extends LightningElement {

    disableSearchButtonFlag = false;

    disableEditButtonFlag = true;

    showSearchMessageFlag = false;

    showEditMessageFlag = false;

    searchMethod(){

        this.disableSearchButtonFlag = true;

        this.disableEditButtonFlag = false;

        this.showSearchMessageFlag = true;

        this.showEditMessageFlag = false;

    }

    editMethod(){

        this.disableSearchButtonFlag = false;

        this.disableEditButtonFlag = true;

        this.showSearchMessageFlag = false;

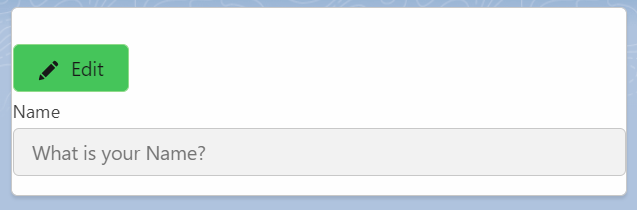
        this.showEditMessageFlag = true;

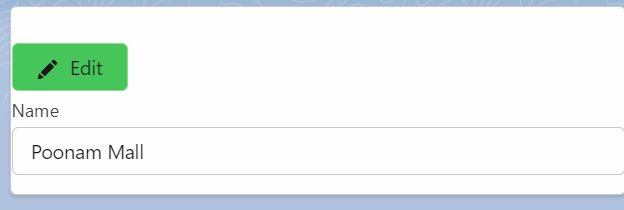
    }

}

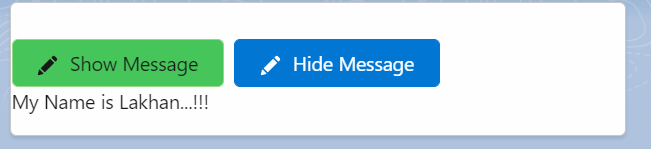
7] Input Field – Read Only / Disabled/ Editable

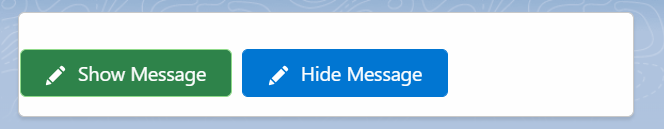
 <lightning-input  data-formfield="applicantID"  type="text" label="Enter Applicant ID" name="appID" disabled={disableFirstName}> </lightning-input> <br />





Assignment:





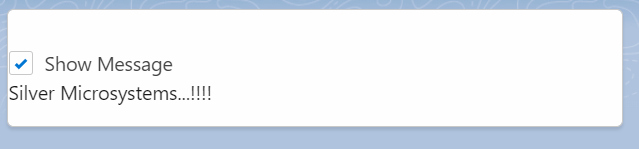
Scenario: Show Name “Silver Microsystems” when checkbox is checked. Else Not.

<lightning-input type="checkbox" label="Show the Name" onchange={showNameHandler} > </lightning-input>

showNameHandler(event){

this.showNameFlag = event.target.checked

}



**Assignment:**

