Example on wire

Create a class first

```
public with sharing class contactlist {
    @AuraEnabled(Cacheable=true)
    public static list<contact> getcont()
    {
        list<contact> conlist=[select firstname from contact limit 10];
        return conlist;
    }

}
```

Create Lwc Component

```
<template>

    <template for:each={conlist.data} for:item="contact">

        <lightning-card key={contact.FirstName} icon-
name="standard:contact">
<p>
    {contact.FirstName}

</p>
        </lightning-card>
    </template>
</template>

import { LightningElement,track,wire } from 'lwc';
import getcont from '@salesforce/apex/contactlist.getcont';
```

```
export default class Contactlistwire extends LightningElement {
    @track conlist;
@wire(getcont) conlist;
}
```

……

Call into aura application

```
<aura:application extends="force:slds">
   <c:contactlistwire></c:contactlistwire>

</aura:application>
```

……

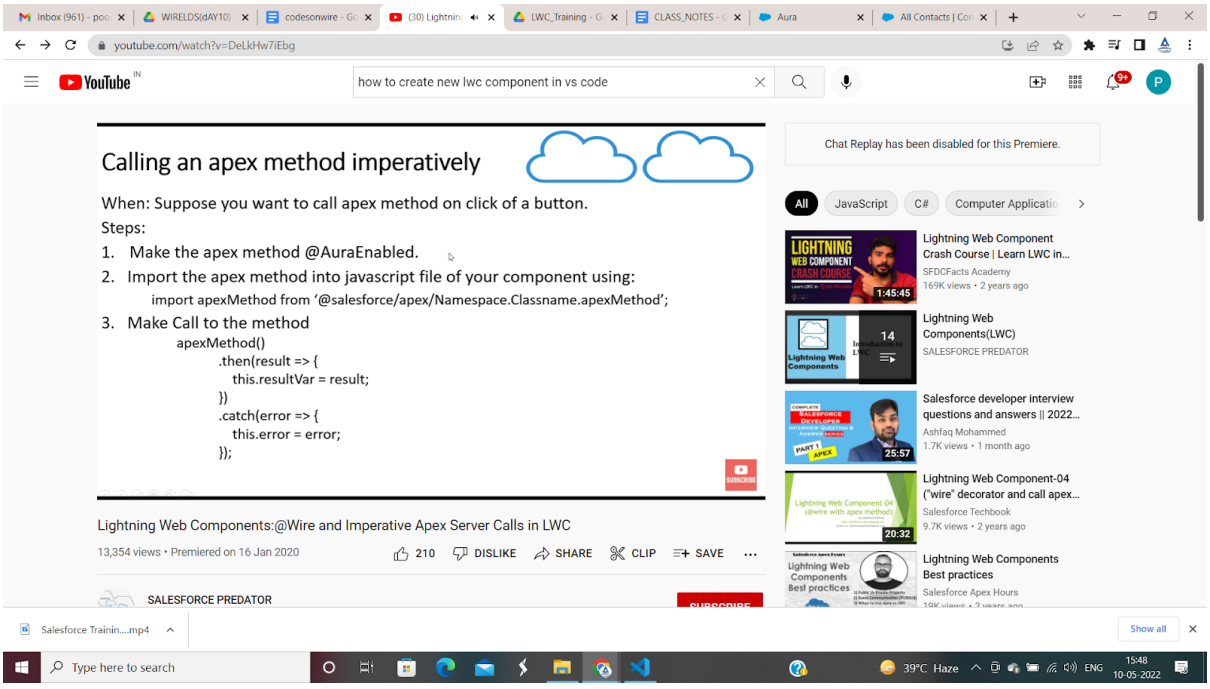
In case user want to add an error also

```
import { LightningElement,track,wire } from 'lwc';
import getcont from '@salesforce/apex/contactlist.getcont';

export default class Contactlistwire extends LightningElement {
    @track conlist;
//@wire(getcont) conlist;
@wire(getcont) contactl({data,error}){

    if(data){
        this.conlist=data;
    }
    else if(error){
        console.log('error #'+ error);
    }
}
//@wire executed before the connectedcallback and render method
//when component is ready after that wire works
```

```
}
<template>


    <template for:each={conlist} for:item="contact"> //from here
reemovedata


        <lightning-card key={contact.FirstName} icon-
name="standard:contact">
<p>
    {contact.FirstName}


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

.....

In case if user want data on a button call then user can use imperatively method



See an example

```
<template>


    <template for:each={conlist} for:item="contact">


        <lightning-card key={contact.FirstName} icon-
name="standard:contact">

<p>

    {contact.FirstName}


</p>

        </lightning-card>

    </template>


        <lightning-button label="getcontacts"
onclick={getconts}></lightning-button>


</template>
```

In js

```
/*@wire(getcont) contactl({data,error}){


    if(data){

        this.conlist=data;

    }

    else if(error){

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

    }

}
```

```
//@wire executed before the connectedcallback and render method
//when component is ready after that wire works
*/
//using imperative call
getconts(){
    getcont().then(result=>{this.conlist=result;
    })
    .catch(error=>{

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

}
```

……..

Write an Apex method for dynamic parameter

Add parameter of boolean with name showcontact in class

Then add this code in js

```
import { LightningElement,track,wire } from 'lwc';
import getcont from '@salesforce/apex/contactlist.getcont';

export default class Contactlistwire extends LightningElement {
    @track conlist;
//@wire(getcont) conlist;
@track showcontact=false;
@wire(getcont,{flag:'$showcontact'}) contactl({data,error}){

    if(data){
        this.conlist=data;
    }
    else if(error){
        console.log('error #'+ error);
```

```
    }
}
handlechange(event){
    this.showcontact=event.target.checked;

}

}
```

In html

```
<template>
<template if:true={showcontact}>

    <template for:each={conlist} for:item="contact">

        <lightning-card key={contact.FirstName} icon-
name="standard:contact">
<p>
    {contact.FirstName}

</p>
        </lightning-card>
    </template>
</template>

      <lightning-input type="checkbox" label="show contacts"
checked={showcontact} onchange={handlechange}></lightning-input>

</template>
```

……..

……

Create messagechannel folder in Sample_MessageChannel.messageChannel-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningMessageChannel xmlns="https://soap.sforce.com/2006/04/metadata">
    <description>This is a sample Lightning Message Channel.</description>
    <isExposed>true</isExposed>
    <lightningMessageFields>
        <description>Variable 1</description>
        <fieldName>variable1</fieldName>
    </lightningMessageFields>
    <masterLabel>Sample Message Channel</masterLabel>
</LightningMessageChannel>
```