Example 2

```
<template>

    <h1>welcome to first program</h1>

    <h2> {greetings} </h2>

    <lightning-card title="My first lightning card">

<lightning-input label="show a value in a text field"
value={greetings}></lightning-input>


    </lightning-card>

</template>


import { LightningElement } from 'lwc';


export default class Myfirstlwc extends LightningElement {

    greetings='Example on data binding input';


}
```

.....

Example on change event

```
<template>

    <h1>welcome to first program</h1>

    <h2> {greetings} </h2>

    <lightning-card title="My first lightning card">

<lightning-input label="show a value in a text field" value={greetings}
onchange={handlevent}></lightning-input>


    </lightning-card>

</template>

import { LightningElement } from 'lwc';


export default class Myfirstlwc extends LightningElement {
```

```
    greetings='Example on data binding input';
  handlevent(event)
  {
    this.greetings=event.target.value;
  }
}
```

.....

Example on Getter Method

No change in Html File

```
import { LightningElement } from 'lwc';

export default class Myfirstlwc extends LightningElement {
    //greeting='Example on data binding input';
  get greetings()
  {
    return 'hi this is return method';
  }
}
```

.....

Example on add two numbers

```
<template>
    <div class="slds-box">
        <lightning-input placeholder="Enter first Number" label="First
Number" onchange={firstnoh}></lightning-input>
        <lightning-input placeholder="Enter second Number" label="Second
Number" onchange={lastnoh}></lightning-input>
        <lightning-button label="Add Two numbers"
onclick={resulthandler}></lightning-button>
        <br/>
        <hr/>
```

```
        Result: {res}
           </div>
</template>
import { LightningElement } from 'lwc';

export default class Lwc2 extends LightningElement {

    fno=0;
    sno=0;
    res=0;
    firstnoh(event)
    {
     this.fno=event.target.value;
    }

    lastnoh(event)
    {
        this.sno=event.target.value;
    }

    resulthandler(event)
    {
      this.res=Number(this.fno)+Number(this.sno);
    }
}
```

Example on conditional rendering through a template

```
<template>
    <div class="slds-box">
```

```
        <lightning-input placeholder="Enter first Number" label="First
Number" onchange={firstnoh}></lightning-input>

        <lightning-input placeholder="Enter second Number" label="Second
Number" onchange={lastnoh}></lightning-input>

        <lightning-button label="Add Two numbers"
onclick={resulthandler}></lightning-button>

        <br/>

        <hr/>

        Result: {res}

    <template if:true={showresult}>

             <h1> The result: {res}</h1>

           </template>

           <template if:false={showresult}>

            <h1>  Click on a command button to view a result</h1>

           </template>

        </div>

</template>

import { LightningElement } from 'lwc';


export default class Lwc2 extends LightningElement {


    fno=0;

    sno=0;

    res=0;

    showresult=false;

    firstnoh(event)

    {

     this.fno=event.target.value;

    }


    lastnoh(event)

    {
```

```
        this.sno=event.target.value;
    }

    resulthandler(event)
    {
        this.showresult=true;
      this.res=Number(this.fno)+Number(this.sno);
    }

}
```

.....

Example on Array

```
<template>
    <lightning-card title="Display contacts">
        <ul class="slds-m-around_medium">
            <template for:each={contacts} for:item="contact">
                <li key={contact.id}>
                    {contact.name},<br/>{contact.title}
                </li>




            </template>


        </ul>
    </lightning-card>

</template>
import { LightningElement } from 'lwc';
```

```
export default class Lwc3 extends LightningElement {
    contacts=[
        {
            id:1,
            name:'Amrita',
            title: 'vp',
        },
        {
            id:2,
            name:'manisha',
            title: 'sales',
        }

    ];
}
```

.....

Create a calculator for Add subtract and divide

Another Example on TEMPLATE RENDERING

template if:true Conditional Rendering LWC

template if:true Conditional Rendering LWC(Lightning Web Component)

To render HTML conditionally, add the `if:true|false` directive to a nested `<template>` tag that encloses the conditional content. template ***if:true|false*** directive is used to display conditional data.

Render DOM Elements Conditionally Lightning Web Component

The **if:true|false={property}** directive binds data to the template and removes and inserts DOM elements based on whether the data is a truthy or falsy value.

template if:true LWC Example

Let's see a simple example to show content based on the selection of checkbox. Example contains a checkbox labeled **Show details**. When a user selects or deselects the checkbox., based on that content is visible. **templateIFTrueExampleLWC.html**

```
1
2
3
4
5
6
7
8
9
10
11
12
```

```
<template>
    <lightning-card
title="TemplateIFTrueCondition
alRendering" icon-
name="custom:custom14">
        <div class="slds-m-
around_medium">
            <lightning-input
type="checkbox" label="Show
details"
onchange={handleChange}></ligh
tning-input>
            <template
if:true={areDetailsVisible}>
                <div
class="slds-m-
vertical_medium">
                    These are
the details!
                </div>
            </template>
        </div>
    </lightning-card>
</template>
```

templateIFTrueExampleLWC.js

```
1
2
3
4
5
6
7
```

```
import { LightningElement }
from 'lwc';
export default class
TemplateIFTrueExampleLWC
extends LightningElement {
    areDetailsVisible = false;
    handleChange(event) {
        this.areDetailsVisible
= event.target.checked;
    }
}
```

……