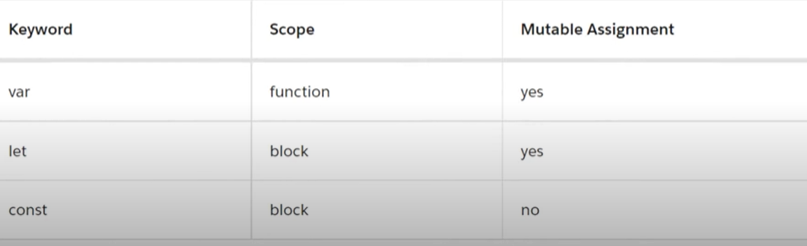
**LWC**

**<lightning-card>**  🡪 It is also a LWC component

Property in LWC 🡪 it is like a variable using which we can show any value to the user

**ConnectedCallback()** 🡪 Whenever our LWC renders in the salesforce org, this function automatically gets called. Inside the function, whatever variable I declare or whatever business logic I write will gets called.

**Difference between let, var, const**

****

**Difference between functional scope and block scope**

Var is used in functional scope(Inside a function braces)

Let, const are used in block scope,

If we declare a variable inside an if block we cannot use it outside if block for let and const.

**Window.alert(“String”)** 🡪 whatever value we show in window.alert will be showed to user when page loads

**# @api**

Let’s say we have two component

* childComponent //Name of component
* parentComponent //Name of component

If we want to access child component from parent component we use <c-child-component></c-child-component> in lightning card tag of HTML file of parent component.

**//Example**

**//Child Component**

**//HTML**

<template>

    <lightning-card title="Child Card">

        {itemName}

    </lightning-card>

</template>

**//JavaScript**

import { LightningElement } from 'lwc';

export default class ChildComponent extends LightningElement {

    itemName="Salesforce NOOB";

}

**//XML**

<?xml version="1.0" encoding="UTF-8"?>

<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">

    <apiVersion>62.0</apiVersion>

    <isExposed>true</isExposed>

    <targets>

    <target>lightning\_\_RecordPage</target>

    </targets>

</LightningComponentBundle>

**//Parent Component**

<template>

    <lightning-card>

        <c-child-component></c-child-component>

    </lightning-card>

</template>

**//Javascript**

import { LightningElement } from 'lwc';

export default class ParentComponent extends LightningElement {}

**//XML**

<?xml version="1.0" encoding="UTF-8"?>

<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">

    <apiVersion>62.0</apiVersion>

    <isExposed>true</isExposed>

    <targets>

    <target>lightning\_\_RecordPage</target>

    </targets>

</LightningComponentBundle>

**Use @api for Property**

**#As this itemName property value is “Salesforce NOOB”, It’s in child component If we want that whenever this parent component call the child component then itemName property values changes to “Parent Salesforce Name”.**

So, we have to change this property directly from parent component by accessing child component instead of changing in child child component

**Solution:**

To do that, We have to add @api decorator and publicly declare the property in child component in js.

Since This is declared publicly it can be accessed in parentcomponent and can be edited.

**//JavaScript**

import { LightningElement , api} from 'lwc';

export default class ChildComponent extends LightningElement {

    @api itemName="Salesforce NOOB";

}

**//Parent Component**

**//HTML**

<template>

    <lightning-card>

        <c-child-component item-name=”Parent Salesforce Name”></c-child-component>

    </lightning-card>

</template>

**Use @api for function**

**#To expose a public method, decorate it with @api. Public methods are part of a component’s API.**

* To communicate down the containment hierarchy, owner and parent components can call JavaScript methods on child components.

**Solution:**

**Step 1:**

**//In child component Js**

First create a public function in childComponent

@ api handleChangeValue(){

this.itemName = “Salesforce LWC demo”;

}

The above function changes the property value to what we want in child component.

Declare @api is a must and import api at the beginning as well.

**Step 2:**

Create a button

**In parentComponent html**

<lightning-button variant="brand" label="submit" onclick={handleClick}></lightning-button>

Here onclick tells what happens on clicking button.

So, on clicking button a handleClick function will be called for its Js.

**Step 3:**

Make a handleClick() function in parentComponent Js:

handleClick(){

        this.template.querySelector("c-child-component").handleChangeValue();

    }

//

this.template 🡪because its in template

this.template. querySelector 🡪 this is standard DOM API, It allows you to efficiently target specific elements within your component's DOM without breaking Salesforce’s **Shadow DOM** encapsulation.

In LWC, **template.querySelector** is used to select a single DOM element within the component's **template**.

**Element 🡪** A complete opening and closing tag together called an element.

**("c-child-component")** 🡪 Inside the parenthesis, write the name of class where we want to fetch data in kebab case naming convention.

**handleChangeValue()🡪** use dot operator and call the function of child class now and end it with semicolon.