Day

01 December 2023

09:25

Diff btw   
1) =, ==, === in JS  
2) Subscribe Functionality in LWC, why function why not directly variable , is java script using pass by value or pass by reference

3) What is the purpose of defining Variable inside the method as well, if we already define above to the method in LWC

1) =, ==, === in JS

In JavaScript, =, ==, and === serve different purposes and have different behaviours. They are used in assignments and comparisons, and each has its own rules for how it evaluates values.

* 1. **= (Assignment Operator):**
     + **The = operator is used for assignment. It assigns the value on the right to the variable on the left.  
         
         
       Let x = 5; // Assigns the value 5 to the variable x**

* 1. **== (Equality Operator):**
     + **The == operator is the equality operator. It compares values for equality after performing type coercion (type conversion).  
         
         
       console.log(5== '5'); // true, because the string '5' is coerced to a number for comparison**

Type coercion can sometimes lead to unexpected results, so it's generally recommended to avoid using == for strict equality checks.

* 1. **=== (Strict Equality Operator):**
     + **The === operator is the strict equality operator. It compares both the value and the type of the operands without performing type coercion.  
         
         
       console.log(5=== '5'); // false, because the values are not of the same type**

Using === is often preferred when you want to ensure both value and type are identical. It avoids unexpected type coercion and leads to more predictable code.

**2) Subscribe Functionality in LWC, why function why not directly variable , is java script using pass by value or pass by reference**

In Lightning Web Components (LWC) and JavaScript in general, the subscription functionality is often implemented using functions rather than directly passing variables. This is related to how JavaScript handles data and the principles of pass by value and pass by reference.

JavaScript is a "pass by value" language, but when it comes to objects (which includes arrays), what gets passed is a reference to the object's location in memory. This can sometimes lead to confusion because it may seem like JavaScript is passing by reference, but it's technically passing a reference by value.

Here's a brief explanation:

* 1. **Pass by Value:**
     + **For primitive data types (like numbers, strings, booleans), JavaScript passes the actual value. When you pass a variable with a primitive value to a function, a copy of the value is passed.**

Let x = 10;

function updateValue(val) {

val = 20;

}

updateValue(x);

console.log(x); // Output: 10

In this case, changing the value inside the function doesn't affect the original variable.

* 1. **Pass by Reference (Reference to Object):**
     + **For objects (including arrays), what gets passed is a reference to the object's memory location. If you modify the object inside the function, it affects the original object because both the original and the copy point to the same memory location.  
         
         
       Let arr = [1, 2, 3];**

Function updateArray(myArray) {

myArray.push(4);

}

updateArray(arr);

console.log(arr); // Output: [1, 2, 3, 4]

In this case, the original array is modified because both arr and myArray reference the same array in memory.

Why a Function for Subscription:

* 1. A function is used for subscription because it allows components to respond dynamically to incoming messages. The callback function can execute logic within the context of the subscribing component, making it versatile and allowing for dynamic handling of messages.
  2. This approach aligns with JavaScript's event-driven and asynchronous nature. Components can register interest in a message channel and specify how they want to handle incoming messages.

3) What is the purpose of defining Variable inside the method as well, if we already define above to the method in LW  
  
*searchDetail is array , when you push data to the array , it is not considered reactive, so array value will be changed but the html will not render because the array is considered to be not changed.*

*Thus when we assign the value to the array inside the handleCustomer function, the engine will considered the array variable is changed and will render the component again.*

For this reason we will be able to see the changed data in the component.

**import { LightningElement, track } from 'lwc';**

**export default class Parentcustomer extends LightningElement {**

**searchDetail=[];**

**handleCustomer(event){**

**this.searchDetail=[];**

**try{**

**event.detail.forEach(element => {**

**var custData={**

**id:element.Id,**

**name:element.Name,**

**email:element.Email\_\_c,**

**status:element.Status\_\_c**

**}**

**this.searchDetail.push(custData);**

**});**

**}catch(e){**

**console.log(e);**

**}**

**}**

**handleClear () {**

**this.searchDetail = [];**

**}**

**}**