**Difference between == and ===?**

== in JavaScript is used for comparing two variables, but it ignores the datatype of variable.

=== is used for comparing two variables, but this operator also checks datatype and compares two values**.**

**For e.g.**

<body>

<p id="demo"></p>

<p id="demo1"></p>

<script>

var a = 10;

var b = "10";

document.getElementById("demo").innerHTML = (a==b) // true

document.getElementById("demo1").innerHTML = (a===b) // false

</script>

</body>

**Hoisting:**

In JavaScript, a variable can be declared after it has been used.

In other words; a variable can be used before it has been declared.

e.g.1 and e.g.2 will produce the same result

**e.g.1**

<body>

<p id="demo"></p>

<script>

var x; // Declare x

x = 10; // Assign 5 to x

elem = document.getElementById("demo"); // Find an element

elem.innerHTML = x; // Display x in the element

</script>

</body>

**e.g.2**

<body>

<p id="demo"></p>

<script>

x = 19; // Assign 5 to x

elem = document.getElementById("demo"); // Find an element

elem.innerHTML = x; // Display x in the element

var x; // Declare x

</script>

</body>

**Datatypes:**

var length = 16;                               // Number  
var lastName = "Johnson";                      // String  
var x = {firstName:"John", lastName:"Doe"};    // Object

JavaScript Types are Dynamic

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

var x;           // Now x is undefined  
x = 5;           // Now x is a Number  
x = "John";      // Now x is a String

**What is the difference between Undefined and Null?**

undefined and null are equal in value but different in type

**For e.g.**

<body>

<p id="demo"></p>

<p id="demo1"></p>

<script>

document.getElementById("demo").innerHTML = (null == undefined); // true

document.getElementById("demo1").innerHTML = (null === undefined); //false

</script>

</body>

**Complex Data:**

The **typeof** operator can return one of two complex types:

* function
* object

The typeof operator returns "object" for objects, arrays, and null.

The typeof operator does not return "object" for functions.

**For e.g.**

typeof {name:'John', age:34} // Returns "object"  
typeof [1,2,3,4]             // Returns "object" (not "array", see note below)  
typeof null                  // Returns "object"  
typeof function myFunc(){}   // Returns "function"