JavaScript

- NaN is the only number which is not equal to itself. (Because these are invalid numbers)
  → **use case** : whenever we have to show any invalid number, then we use NaN in JavaScript.

* **Special numbers**
  +0 -0 NaN
  Infinity -Infinity

* **Equality operators :—**
  == → abstract quality operator
  === → strict equality operator

*** Both == and === checks the type.

== → it checks the type of both operands,
↳ if it is same, then it calls ===
↳ if types are not same then type conversion occurs (coursion) & then comparision is done.

=== → it checks types of both operands.
↳ if types are different it return false
↳ if types are same, then value comparision happens.

:: == equal (double equals to) actually do type conversion
=== doesn't do type conversion

major difference →

Page No.
Date / /

Example:- 1 == "1" It will do type conversion, It will convert string to number, then comparision will happen.

1 == 1 → numbers

number

→ True

1 === 1

→ 1 == "sachin"

→ We can't convert string to a number and it will lead to a invalid number
→ And invalid number in JS is NAN.

1 == NaN

→ False

- Typeof null is null not an object (Historical mistake)

* Abstraction operation :-
These are some set of algorithms, that is present in the ecmascript docs, but they are not available for usage in ecamascript.
i.e. we as developers cannot use these operation directly

Types

1) ToNumber():

2 - null // 2 - 0 = 2
2 - undefined // 2 - undefined = NaN

null == +0
undefined == NaN

Symbol == TypeError

Four pillars 2) Scopes 4) Asynchronous.

Page No.
Date / /

* **ToBoolean :—**
undefined, null, NaN, +0, -0, false

they are falsy values

* **Abstract Equality Comparision (==)**

null == undefined → True

- if x is ~~string~~ Number, y is string → converts y → Tonumber then again does the comparision.

Example 12 == "12" → True

- if x is string, y is number → converts x → to numbers then again comparision

- x → boolean → Convert x → toNumber & then again do comparision.

false == "0" → True

- y → boolean → Convert y → to Number and do the comparision again

- x → number | Symbo | String and y → object → we need to convert y to y → Toprimitive & compare again

- x → object and y → number | Symbol | String

convert x → to Primitive & compare again.

otherwise return → false

example :—

null == false → false

y → boolean → ToNumber → 0

null == 0 → false

* Strict Equality Comparision :- (===)

- if type(x) is different from type(y), return false
- if type(x) is Number, then
  a. if x is NaN, return false
  b. if y is NaN, return false
  c. if x is same as number value as y, return true
  d. if x is +0 and y is -0, return true
  e. if x is -0 and y is +0, return true
  f. Return false

NaN === NaN → False

* let obj1 = {x:10};
let obj2 = {x:10};
let obj3 = {y:10};

obj1 === obj2 → false } memory location is
obj1 === obj3 → false } different
obj1 === obj1 → True → m/m allocation is same

{x:10} === {x:10} → false

Because m/m allocation is different

== → does coercion

=== → does not coercion

→ let obj = { x: 10, y: 20 };
console.log(' ${obj} '); // console.log(" " + obj);
If an object is going to get converted to a string the default toString will give you object Object
output: [object Object]

3 > 2 > 1 // false

* NaN :- some Cases

```
console.log(Number("0xa"));                    // 10
let x = 12; console.log(x == NaN);             // False
console.log(isNaN(x));                         // false
console.log(isNaN("sanket"));                  // true
~~console.log~~
console.log(Number.isNaN("sanket"));           // false
console.log(Number.isNaN(x));                  // false
```

* NegativeZero Cases :-

```
let x = -0
console.log(x === 0);                          // true
console.log(Object.is(x, -0));                 // true
console.log(Object.is(x, 0));                  // false

Console.log(Math.sign(-3));                    // -1
Console.log(Math.sign(2));                     // 1
Console.log(Math.sign(-0));                    // -0
Console.log(Math.sign(0));                     // 0
```