Kinds of Emptiness

There are three types of emptiness

1. Undeclared
2. Undefined
3. Uninitialized aka TDZ(Temporarily Dead Zone)

Undeclared means it is never been created in any scope that we have access to. Undefined means there is definitely a variable but at this moment it has no value.

Special Values NaN: Invalid Number

In JavaScript NaNs are not equal to each other. NaN is the only value that

does not have the identity property i.e. it is not equal to itself.

var myAge = Number("38");

var myCatAge = Number("n/a");

console.log(myAge - "my son's age"); // NaN console.log(isNaN(myAge)); // 38 console.log(isNaN(myCatAge)); // NaN

console.log(isNaN("my son's age"));

//NaN, here the string is coerced into number the number is NaN

In ES6 we have Number.isNaN. It does not coerces the value into numbers. console.log(Number.isNaN(myCatAge));//true

console.log(Number.isNaN("my son's age"));//false

Negative Zero

Arithmetic operations: When you perform arithmetic operations involving negative zero, it behaves like regular zero.

const negativeZero = -0;

console.log(negativeZero + 0); // Output: 0

console.log(negativeZero \* 10); // Output: 0

console.log(negativeZero / 5); // Output: 0

Equality and comparison: When comparing values, negative zero is considered equal to regular zero.

const zero = 0;

const negativeZero = -0;

console.log(zero === negativeZero); // Output: true

console.log(zero == negativeZero); // Output: true

console.log(zero > negativeZero);  // Output: false

console.log(zero < negativeZero);  // Output: false

However, there are some cases where negative zero is considered different from regular zero.

console.log(Object.is(zero, negativeZero)); // Output: false

String representation: When you convert negative zero to a string using toString() or String(), it is represented as "-0".

const negativeZero = -0;

console.log(negativeZero.toString()); // Output: "-0"

console.log(String(negativeZero));    // Output: "-0"