console.log(null >= 0); // true

In this case, JavaScript converts null to a number before performing the comparison. When null is converted to a number, it becomes 0. Therefore, the expression becomes:

console.log(0 >= 0); // true

console.log(null == 0); // false

This case uses the loose equality comparison (==), which does not perform type coercion in the same way as the greater than or equal to comparison. Instead, it checks the values while preserving their types. In JavaScript, null is only equal to undefined when using ==.

This type of type conversion and manipulation in JavaScript is called **"implicit type coercion"**.

In certain cases, JavaScript automatically converts values to other types to facilitate various operations, which is known as implicit type coercion. This happens in operations that involve comparisons, equality checks, and other mathematical operations.

**The first example** (null >= 0) demonstrates how JavaScript implicitly converts types, where null is converted to 0 when a mathematical comparison is needed.

**The second example** (null == 0) shows how JavaScript handles loose equality (==), which checks the values without converting the types, leading to different results compared to strict equality (===) or mathematical comparisons.