perplexity

JavaScript Comparison Operators with Key Properties (Including null vs undefined)

Essential Comparison Operators

- **Equality (==)**
  - Performs type coercion, so values of different types might compare as equal.
  - Example: `null == undefined // true` (special case)[1] [2] [3].
- **Strict Equality (===)**
  - Compares both value and type with *no* coercion.
  - Example: `null === undefined // false`[1] [2] [3].
- **Inequality (!=)**
  - Like `==`, applies type coercion.
  - Example: `null != undefined // false` (since `null == undefined` is true)[1] [2] [3].
- **Strict Inequality (!==)**
  - No coercion; compares value and type.
  - Example: `null !== undefined // true`[1] [2] [3].
- **Relational Operators (`>`, `<`, `>=`, `<=`)**
  - Convert operands to numbers for comparison (except when dealing with objects or NaN).
  - Example: `null < 1 // true` (null is converted to 0), `undefined < 1 // false` (undefined cannot be converted, results in false)[4] [3].

Special Rules for `null` and `undefined` in Comparisons

1. **Loose Equality (==) Special Case:**
   - `null == undefined` is `true`
   - `null == false` is `false`
   - `undefined == false` is `false`
   - `null == 0` is `false`
   - `undefined == 0` is `false`[1] [3] [4]
2. **Strict Equality (===) Always:**
   - `null === undefined` is `false` (types are different)[1] [2] [3]
3. **Relational Operators:**

- Comparisons with `null`: gets converted to 0 (`null < 1` is true, `null > -1` is true).
- Comparisons with `undefined`: always `false` (`undefined < 1 // false`, `undefined > -1 // false`)[4] [3].

4. **typeof Results:**
   - `typeof null` is `'object'` (quirk in JS)
   - `typeof undefined` is `'undefined'`[1] [2] [4]
5. **NaN with undefined:**
   - Any arithmetic operation with `undefined` results in `NaN`
   - With `null`, arithmetic will treat `null` as 0 (e.g., `null + 2 // 2`)[4].

Interview Tips and Key Properties

- **Use `===` and `!==` for most comparisons** to avoid surprises from type coercion[1] [3].
- **Explicit null/undefined checks:**
  - `if (val == null)` is true for both `null` and `undefined` (because `==` treats them equal).
  - `if (val === null)` or `if (val === undefined)` only matches exact type.
- **Falsy Values:** Both `null` and `undefined` are falsy, but so are `0`, `''`, `false`, and `NaN`.
- **Safe Defaulting:** Use `??` (nullish coalescing) to assign default values only if a variable is `null` or `undefined`, *not* for other falsy values[5].
  - Example: `let value = someVar ?? 'default';`
- **Objects:** `[] == [] // false`; object comparison is by reference, not structure or content.

Quick Interview Table

Expression	Result	Reason
null == undefined	true	Special loose equality rule[1] [2] [3]
null === undefined	false	Strict: different types[1] [2] [3]
null == 0	false	null only loosely equals undefined
undefined == 0	false	undefined only loosely equals null
null < 1	true	null coerced to 0[4]
undefined < 1	false	undefined coerced to NaN (always false)[4] [3]
typeof null	object	Historical bug/quirk[1] [2] [4]
typeof undefined	undefined	As expected

Summary Table: null vs undefined

	null	undefined
Type	object	undefined
Use	Programmer assigned	System assigned (uninitialized, missing argument) [1] [4] [2]
Equality to each other (==)	true	true
Strict equality (===)	false	false
Arithmetic	Converts to 0	Converts to NaN

Code Snippet Showcasing Comparisons

```
let a;
console.log(a == null);        // true      (a is undefined here)
console.log(a === undefined);  // true
console.log(null == undefined); // true
console.log(null === undefined); // false
console.log(a == false);       // false
console.log(a == 0);           // false
console.log(null < 1);         // true
console.log(undefined < 1);    // false
```

TL;DR for Interviews

- **Use `===` and `!==` for most comparison operations.**
- **`null == undefined` is true, but `null === undefined` is false.**
- **Relational operators treat `null` as 0 and `undefined` as NaN.**
- **Both are falsy, but their roles differ: `null` is intentional emptiness, `undefined` is absence by default.**
- **The nullish coalescing operator (`??`) only defaults on null or undefined, not other falsy values.**

Mastering these will help you answer almost any JS comparison question confidently in an interview[1] [2] [3] [4].

⁂

1. https://www.geeksforgeeks.org/javascript/undefined-vs-null-in-javascript/
2. https://atomicjuggernaut.hashnode.dev/javascript-interview-question-difference-between-null-and-undefined
3. https://www.hellojavascript.info/docs/general-javascript-questions/javascript-fundamentals/comparisons-operators
4. https://www.scaler.com/topics/javascript/null-and-undefined-in-javascript/

5. https://www.w3schools.com/js/js_comparisons.asp