NodeJS

LABA

JAVASCRIPT DATA TYPES

In javascript, we have 2 main kinds of types, Primitive and Object types, the primitive types are:

Numbers	Symbol
Strings	Null, Undefined
Booleans	BigInt

Object types are any value that’s not a primitive type.

Solvd NodeJS LABA

NUMBER

```
const n1 = 2;
console.log(n1);
// Output: 2

const n2 = 1.3;
console.log(n2);
// Output: 1.3

const n3 = Infinity;
console.log(n3);
// Output: Infinity

const n4 = 'something here too' / 2;
console.log(n4);
// Output: NaN
```

NUMBER

```
const a = 5;
a.constructor
// Output: Number() { [native code] }

a.toString
// Output: toString() { [native code] }
```

NUMBER

```
console.log(Number.MAX_VALUE);
// Output: 1.7976931348623157e+308

console.log(Number.MAX_VALUE * 2);
// Output: Infinity

console.log(9007199254740992 + 1 === 9007199254740992);
// Output: true (precision lost)

console.log(BigInt(9007199254740992) + BigInt(1));
// Output: 9007199254740993n
```

STRINGS

```
'A string'
"Another string"

"A \
string"

'I\'m a developer'

"A " + "string"

`a string`
```

STRINGS

```
`a string with ${something}`
`a string with ${something + somethingElse}`
`a string with ${obj.something()}`

`a string
with
${something}`
```

Solvd
NodeJS LABA

BOOLEAN

```
0
-0
NaN
undefined
null
'' // empty string
```

Solvd
NodeJS LABA

NULL AND UNDEFINED

```
typeof variable === 'undefined'

typeof 0; // number

typeof 10n; // bigint

typeof true; // boolean

typeof "foo"; // string

typeof Symbol("id"); // symbol
```

Solvd
NodeJS LABA

OBJECT

```
// Object literal
const obj1 = { name: "Alice", age: 25 };

// Object constructor
const obj2 = new Object();
obj2.name = "Bob";

// Object.create()
const obj3 = Object.create({ base: "prototype" });
```

Solvd NodeJS LABA

JAVASCRIPT OPERATORS (+)

```
// Number + Number => Addition
let x = 1 + 2;
console.log(x);
// Output: 3

// Number + String => Concatenation
let y = 5 + "hello";
console.log(y);
// Output: 5hello
```

Solvd
NodeJS LABA

JAVASCRIPT OPERATORS (-)

```
// Number - Number => Subtraction
let x = 10 - 7;
console.log(x);
// Output: 3

// Number + String => Concatenation
let y = "Hello" - 1;
console.log(y);
// Output: NaN
```

Solvd NodeJS LABA

JAVASCRIPT OPERATORS (*)

```
// Number * Number => Multiplication
let x = 3 * 3;
let y = -4 * 4;
console.log(x); // Output: 9
console.log(y); // Output: -16

let a = Infinity * 0;
let b = Infinity * Infinity;
console.log(a); // Output: NaN
console.log(b); // Output: Infinity

let z = 'hi' * 2;
console.log(z); // Output: NaN
```

Solvd
NodeJS LABA

JAVASCRIPT OPERATORS (/)

```
// Number * Number => Division
let x = 5 / 2;
let y = 1.0 / 2.0;
console.log(x); // Output: 2.5
console.log(y); // Output: 0.5

let a = 3.0 / 0;
let b = 4.0 / 0.0;
console.log(a); // Output: Infinity
console.log(b); // Output: Infinity

let z = 2.0 * -0.0;
console.log(z); // Output: -Infinity
```

JAVASCRIPT OPERATORS (%)

```
// Number % Number => Modulus of the number
let x = 9 % 5;
let y = -12 % 5;
let z = 1 % -2;
let a = 5.5 % 2;
let b = -4 % 2;
let c = NaN % 2;

console.log(x); // Output: 4
console.log(y); // Output: -2
console.log(z); // Output: 1
console.log(a); // Output: 1.5
console.log(b); // Output: -0
console.log(c); // Output: NaN
```

Solvd NodeJS LABA

JAVASCRIPT OPERATORS (**)

```
// Number ** Number => Exponential of the number
let y = -(4 ** 2);
let z = 2 ** 5;
let a = 3 ** 3;
let b = 3 ** 2.5;
let c = 10 ** -2;
let d = 2 ** 3 ** 2;
let e = NaN ** 2;

console.log(y); // Output: -16
console.log(z); // Output: 32
console.log(a); // Output: 27
console.log(b); // Output: 15.588457268119896
console.log(c); // Output: 0.01
console.log(d); // Output: 512
console.log(e); // Output: NaN
```

Solvd
NodeJS LABA

JAVASCRIPT OPERATORS (**)

```
-2 ** 3;
// SyntaxError: Unary operator used immediately before exponentiation expression.
+2 ** 3;
// SyntaxError: Unary operator used immediately before exponentiation expression.
~5 ** 2;
// SyntaxError: Unary operator used immediately before exponentiation expression.
!1 ** 2;
// SyntaxError: Unary operator used immediately before exponentiation expression.
```

JAVASCRIPT OPERATORS (**)

```
(-2) ** 3
// Output: 8 (negates 2 first, then raises -2 to the power of 3)
(+2) ** 3
// Output: 8 (applies positive to 2, then raises to the power of 3)
~(5 ** 2)
// Output: ~25 = -26 (raises 5 to the power of 2, then applies bitwise NOT)
!(1 ** 2)
// Output: !1 = false (raises 1 to the power of 2, then applies logical NOT)
```

JAVASCRIPT OPERATORS (++)

```
// Postfix
let a = 2;
let b = a++; // b = 2, a = 3

// Prefix
let x = 5;
let y = ++x; // x = 6, y = 6

console.log(a); // Output: 3
console.log(b); // Output: 2
console.log(c); // Output: 6
console.log(d); // Output: 6
```

Solvd
NodeJS LABA

JAVASCRIPT OPERATORS (-=)

```
let c = 3;
c -= 1; // Output: 2
c; // Output: 2
```

Solvd
NodeJS LABA

JAVASCRIPT OPERATORS (*=)

```
let d = 5;
d *= 2; // Output: 10
d; // Output: 10
```

JAVASCRIPT OPERATORS (<<=)

```
let yoo = 5;
// Expected output 20 (in Binary 10100)
console.log(yoo <<= 2);
```

Solvd
NodeJS LABA

JAVASCRIPT OPERATORS (&&=)

```
let name = {
  firstName: "Ram",
  lastName: "",
};

console.log(name.firstName);

// Changing the value using logical AND assignment operator
name.firstName &&= "Shyam";

// Here the value changed because name.firstName is truthy
console.log(name.firstName);
console.log(name.lastName);
```

JAVASCRIPT OPERATORS (||=)

```
let name = {
  firstName: "Ram",
  lastName: "",
};

console.log(name.firstName);

// Changing the value using logical OR assignment operator
name.firstName ||= "Shyam";
```

Solvd
NodeJS LABA

JAVASCRIPT OPERATORS (??=)

```
let x = 12;
let y = null;

let z = 13;

// The value of x will become unchanged because x is not nullish.
x ??= z;
// The value of y will be changed because y is nullish.
y ??= z;

console.log(x); // 12
console.log(y); // 13
```

Solvd
NodeJS LABA

JAVASCRIPT OPERATORS (??=)

Initial Value	`x ??= "default"` Result	`x ||= "default"` Result
`null`	`"default"` ✅	`"default"` ✅
`undefined`	`"default"` ✅	`"default"` ✅
`""` (empty string)	`""` (unchanged)	`"default"` ✅
`0`	`0` (unchanged)	`"default"` ✅
`false`	`false` (unchanged)	`"default"` ✅
`NaN`	`NaN` (unchanged)	`"default"` ✅
`"hello"`	`"hello"` (unchanged)	`"hello"` (unchanged)
`42`	`42` (unchanged)	`42` (unchanged)

COMPARISON OPERATORS

```
console.log(10 > 5);
// Output: true

console.log(10 === "10");
// Output: false
```

LOGICAL OPERATORS

```
// Illustration of (==) operator
let x = 5;
let y = '5';

console.log(x == 5); // Output: true
console.log(y == 5); // Output: true
console.log(x == y); // Output: true

console.log(NaN == NaN); // Output: false
console.log(0 == false); // Output: true
console.log(0 == null); // Output: false
```

Solvd
NodeJS LABA

STRICT EQUALITY OPERATORS

```
let x = 5;
let y = '5';

console.log(x === 6); // Output: false
console.log(y === '5'); // Output: true
console.log(x === y); // Output: false

console.log(NaN === NaN); // Output: false
console.log(0 === false); // Output: false
console.log(0 === null); // Output: false
```

GREATER THAN OR EQUAL OPERATOR

```
let x = 5;
let y = '5';

console.log(x >= 5); // Output: true
console.log(y > "15"); // Output: true
console.log(x > "5"); // Output: true
console.log(y > 15); // Output: false
```

Solvd
NodeJS LABA

TERNARY OPERATOR

```
condition ? trueExpression : falseExpression;

let PMarks = 50;

let res = PMarks > 39 ? "Pass" : "Fail";

console.log(res);
```

NESTED TERNARY OPERATOR

```
function checkAge(age) {
  return age >= 18 ? "Adult" : "Minor";
}

console.log(checkAge(20)); // Output: Adult
console.log(checkAge(15)); // Output: Minor

function sayHello(name) {
  console.log(`Hello, ${name}!`);
}

function sayGoodbye(name) {
  console.log(`Goodbye, ${name}!`);
}

let isLeaving = true;
let name = 'Geeks';

isLeaving ? sayGoodbye(name) : sayHello(name);
```

Solvd
NodeJS LABA

NESTED TERNARY OPERATOR

```
let hour = 15;
let message;

if (hour < 12) {
  message = "Good morning";
} else {
  message = "Good afternoon";
}

console.log(message); // Output: Good afternoon

// with ternaries
message = hour < 12 ? "Good morning" : "Good afternoon";
console.log(message); // Output: Good afternoon
```

Solvd NodeJS LABA

IN OPERATOR

```
let languages = ["HTML", "CSS", "JavaScript"];

// true (index 1 exists in the array)
console.log(1 in languages); // Output: true

// false (index 3 doesn't exist in the array)
console.log(3 in languages); // Output: false
```

IN OPERATOR

```
const Data = {
  name: "Rahul",
  age: 21,
  city: "Noida",
};

// true ("name" property exists in the object)
console.log("name" in Data);

// false ("address" property doesn't exist in the object)
console.log("address" in Data);
```

Solvd NodeJS LABA

INSTANCEOF OPERATOR

```
let languages = ["HTML", "CSS", "JavaScript"];

console.log(languages instanceof Array); // Output: true
console.log(languages instanceof Object); // Output: true
console.log(languages instanceof String); // Output: false
console.log(languages instanceof Number); // Output: false
```

Solvd
NodeJS LABA

BIGINT TYPE

```
// Parameter in decimal format
let bigNum = BigInt("12342222222222222222222222222222");
console.log(bigNum);
// Output: 12342222222222222222222222222222n

// Parameter in hexadecimal format
let bigHex = BigInt("0x1fffffeeeeeeeeeeeeeeeeeeeeeef");
console.log(bigHex);
// Output: 265845598628795424485665828099109246 3n

// Parameter in binary format
let bigBin = BigInt("0b10101010101010111111111111111");
console.log(bigBin);
// Output: 178962431n
```

Solvd
NodeJS LABA

BIGINT TYPE

```
typeof 100n === 100 // Output: false
typeof 100n == // Output: true due to coercion
typeof 100n == 'bigint' // Output: true
100n < 101 // Output: true due to coercion
```

Solvd
NodeJS LABA

SORTING

```
let arr = [4, 2, 5n, 2n];

arr.sort()

console.log(arr) // Output: [2, 2n, 4, 5n];
```

Solvd
NodeJS LABA

OPTIONAL CHAINING

```
let Value = user.dog && user.dog.name;
let Value = user.dog?.name;

const user = {
    dog: {
      name: "Alex",
    }
};

console.log(user.cat?.name); // Output: undefined
console.log(user.dog?.name); // Output: Alex
console.log(user.cat.name);
// TypeError: cannot read properties of undefined (reading "name");
```