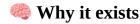
In JavaScript, **BigInt** is a special numeric type used to represent **very large integers** — numbers larger than what the normal **Number** type can safely handle.



The regular Number type in JS uses 64-bit floating point format (IEEE 754), which can safely store integers only up to

```
2^53 - 1 \rightarrow 9007199254740991.
```

If you go beyond that, precision errors happen:

```
console.log(9007199254740992 === 9007199254740993); // true \mathbb{Q} (wrong!)
```

To fix that, JavaScript introduced **BigInt**.

How to use it

You can create a BigInt in two ways:

1. By adding n at the end of an integer:

```
const big = 123456789012345678901234567890n;
```

2. By using the constructor:

```
const big = BigInt("123456789012345678901234567890");
```

AOperations with BigInt

You can do normal math operations — but only with other BigInts:

```
const a = 10n;
const b = 20n;
console.log(a + b); // 30n
console.log(a * b); // 200n
```

Nou can't mix Number and BigInt directly:

```
console.log(10n + 5); // TypeError 🗙
```

If needed, convert manually:

```
Number(10n) + 5; // \sqrt{} 15
```

***** Example use case

When working with IDs, timestamps, cryptography, or very large counters:

```
const userId = 987654321987654321987654321n;
console.log(userId + 1n);
```



Note

- BigInt cannot be used with some built-in functions like ${\tt Math.sqrt}($).
- JSON does not support BigInt directly (you'd need to convert to string).