JavaScript TUTORIAL

JavaScript BigInt

Summary: in this tutorial, you'll learn about the `BigInt` type which is the built-in object that can represent very big whole numbers.

Introduction to the JavaScript BigInt

ES2020 introduced a new built-in object called `BigInt` that allows you to represent big whole numbers, which cannot represent by the number type (larger than $2^{53} - 1$).

The `bigint` is the primitive type like `number`, `string`, `symbol`, `boolean`, `undefined`, and `null`.

To make a `BigInt`, you append `n` to the end of the number literal, for example:

```
let bigInt = 9007199254740991n;
```

Alternatively, you can call the `BigInt()` function:

```
let bigInt = BigInt(9007199254740991);
```

See the following calculation with a `Number`:

```
let x = Number.MAX_SAFE_INTEGER;
x = x + 1; // 9007199254740992
x = x + 1; // 9007199254740992 (same as above)
```

and with a `BigInt`:

```
let x = BigInt(Number.MAX_SAFE_INTEGER);
x = x + 1; // 9007199254740992n
```

```
x = x + 1; // 9007199254740993n (correct now)
```

Type

The type of a `BigInt` is `bigint`. For example:

```
console.log(typeof bigInt); // bigint
console.log(typeof BigInt(100) === 'bigint'); // true
```

Operators

The `BigInt` supports the following operator `+`, `*`, `-`, `**`, `%`, bitwise operators except `>>>` (zero-fill right shift). It doesn’t support the unary operator (`+`).

The `/` operator will also work with the whole numbers. However, the result will not return any fractional digits. For example:

```
let result = 3n / 2n;
console.log(result); // 1n, not 1.5n
```

Comparisons

A `BigInt` is not strictly equal (`===`) to a `Number`:

```
console.log(1n === 1); // false
```

However, a `BigInt` is loosely equal to a number when you use the `==` operator:

```
console.log(1n == 1); // true
```

You can use the `<`, `<=`, `>`, `>=` to compare a `BigInt` with a `Number`:

```
console.log(1n < 2); // true
console.log(2n > 1); // true
console.log(2n >= 2); // true
```

Conditionals

A `BigInt` is converted to a `Boolean` via the `Boolean()` function in conditionals such as `if` statement or logical operators `||`, `&&,` `!`. In other words, it works like a `Number` in these cases. For example:

```
let count = 0n;

if(count) {
    console.log(true);
} else {
    console.log(false);
}
```

Summary

- The `BigInt` is a new primitive type that can represent big whole numbers bigger than $2^{53} - 1$, which is the largest number JavaScript can reliably represent with the `number` type.
- Append `n` to a literal integer or use `BigInt()` function to form a `bigint`.