

The type conversion only used in three data types which are **String**, **number**, and **boolean**.

— To String Conversion :

In this conversion, the number is converted to a string when we need it.

JavaScript provides different types of built-in methods or functions used in the Explicit conversion.

1.String()


The String() method is a Global method, and it is used to convert numbers to strings.

It can also be used to convert literals and expressions.

Syntax :

```
String (ValueToConvert)
```

Example:



```
String(902); // OutPut >> "902"
String(3.14); // OutPut >> "3.14"
String(-12.123); // OutPut >> "-12.123"
String(1 + 1); // OutPut >> "2"
String(true); // OutPut >> "true"
String(false); // OutPut >> "false"
String(null); // OutPut >> "null"
String(NaN); // OutPut >> "NaN"
String(undefined); // OutPut "undefined"
```

As in the above example, the String() method converts numbers into a string as we expected.

We can pass null, NaN, and undefined as the values to the method then String() method converts them to string as well, without throwing errors.

Note:

undefined is the variable in which it is not assigned any value during its creation phase.

null is the value in which it is empty or not initialized or not defined, or it has a zero value.

NaN refers as Not-a-Number, it used to indicate an error to enter a valid number.

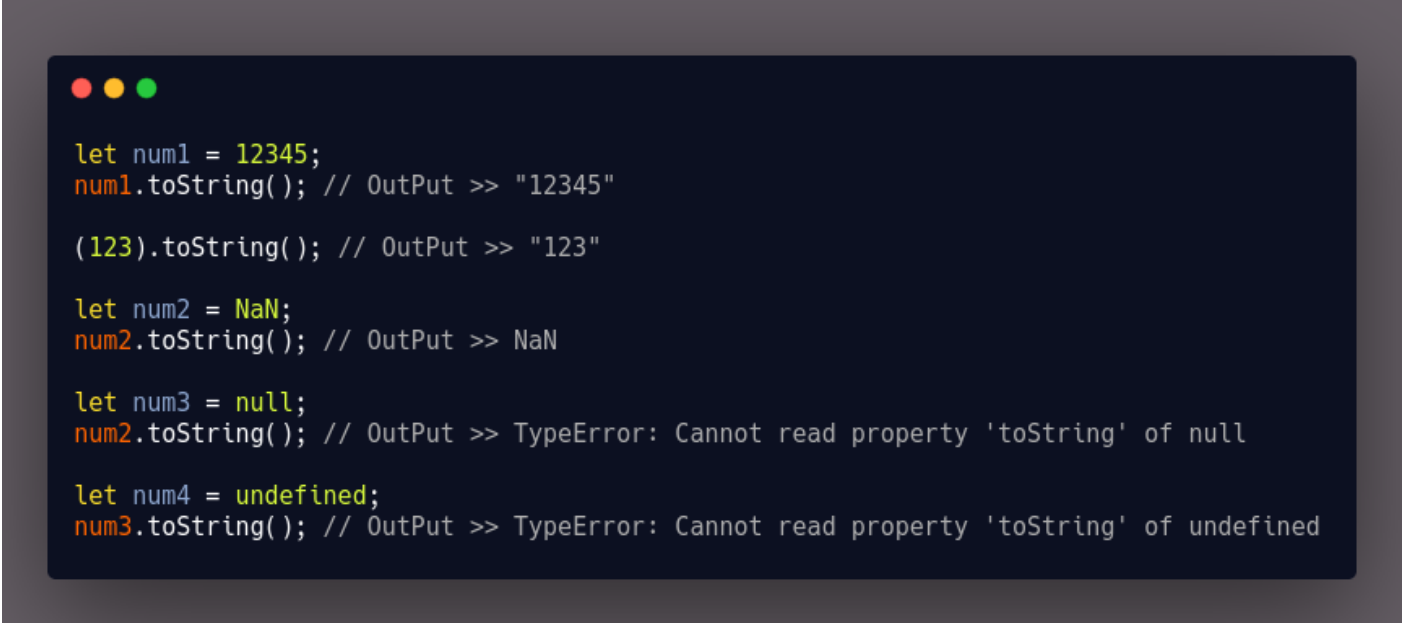
2. toString()

It does the same job as String() method.

Syntax:

```
number.toString()
```

Example:



```
let num1 = 12345;
num1.toString(); // OutPut >> "12345"

(123).toString(); // OutPut >> "123"

let num2 = NaN;
num2.toString(); // OutPut >> NaN

let num3 = null;
num2.toString(); // OutPut >> TypeError: Cannot read property 'toString' of null

let num4 = undefined;
num3.toString(); // OutPut >> TypeError: Cannot read property 'toString' of undefined
```

First, we declare and assign the value to the variable then using the variable name we call the toString() method using the dot(.) operator.

If we try to pass null and undefined values, then the JavaScript engine shows TypeError.

3. toExponential()

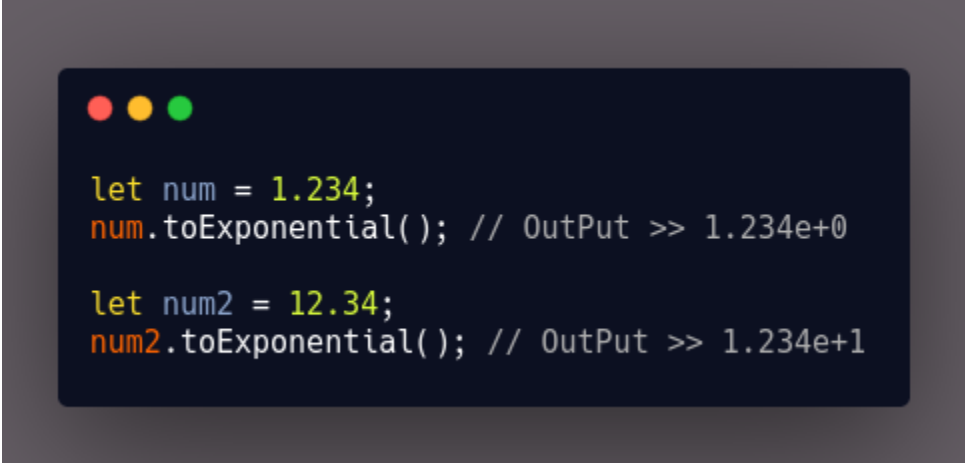
This method can be used to convert the number into a string exponential symbol(e+number).

Syntax:

```
number.toExponential(OptionalParameter)
```

The parameter is optional which represents the number of digits after the decimal point.

Example:



```
let num = 1.234;  
num.toExponential(); // OutPut >> 1.234e+0  
  
let num2 = 12.34;  
num2.toExponential(); // OutPut >> 1.234e+1
```

4. toFixed()

The toFixed() method in JavaScript is used to convert the number into a string by specifying the number of digits to the right after the decimal point.

Syntax:

```
number.toFixed(optionalParameter)
```

The parameter is optional which represents the number of digits after the decimal point.

Example:



```
let num = 123.456;  
num.toFixed(); // OutPut >> 123  
  
let num2 = 123.456;  
num2.toFixed(2); // OutPut >> 123.46
```

5. toPrecision()

In this method the number is converted to a string, to format the number to the fixed length.

Syntax:

```
number.toPrecision(optianlParamater)
```

The parameter is optional which represents the number of digits.

Example :



```
let num = 123.456;  
num.toPrecision(); // OutPut >> 123.456  
  
let num2 = 123.456;  
num2.toPrecision(3); // OutPut >> 123
```

— To Number Conversion :

The String to Number is converted using some built-in methods in JavaScript.

Built-in methods are as follows.

1.Number()

The string value is converted to a number using the Global number() method.

It can be used to convert numerical text as well as boolean values to numbers.

Syntax:

```
Number (valueToConvert)
```

Example:



```
Number('123'); // OutPut >> 123
Number(true); // OutPut >> 1
Number(false); // OutPut >> 0
Number('abc'); // OutPut >> NaN
Number(null); // OutPut >> 0
Number(undefined); // OutPut >> NaN
```

If we pass the invalid string and undefined value then the result always printed as NaN. The null value has a default value of 0.

2. parseInt()

The parseInt() method first parses or analyzes the string, and then it converts to integer i.e. number.

Syntax:

```
parseInt(String, Radix)
```

In this method the first string parameter is mandatory.

The radix is used to denote numbers to the numeral system i.e. the numbers parsed into hexadecimal to the decimal number. This parameter is optional.

Example:



```
parseInt('123$ThisWillBeIgnored'); // OutPut >> 123  
parseInt('JavaScript*123'); // OutPut >> NaN  
parseInt('321'); // OutPut >> 321
```

In the first example, the number will be printed '123', and the rest of the string from '\$' will ignore.

In the second example, the string value must start with the number otherwise the whole string will be ignored, and the output will be printed as NaN.

3. parseFloat()

The parseFloat() method is used to parse or analyze the string, and then it converts to a floating-point number i.e. 3.14.

Syntax:

```
parseFloat(String)
```

The single string type parameter can only be passed in the method.

Example:



```
parseFloat('3.14'); // OutPut >> 3.14
parseFloat(123); // OutPut >> 123
parseFloat('123.456*JavaScript'); // OutPut >> 123.456
parseFloat('JavaScript@321.123'); // OutPut >> NaN
```


In the above examples, the behavior of the `parseFloat()` method is the same as the `parseInt()` method.

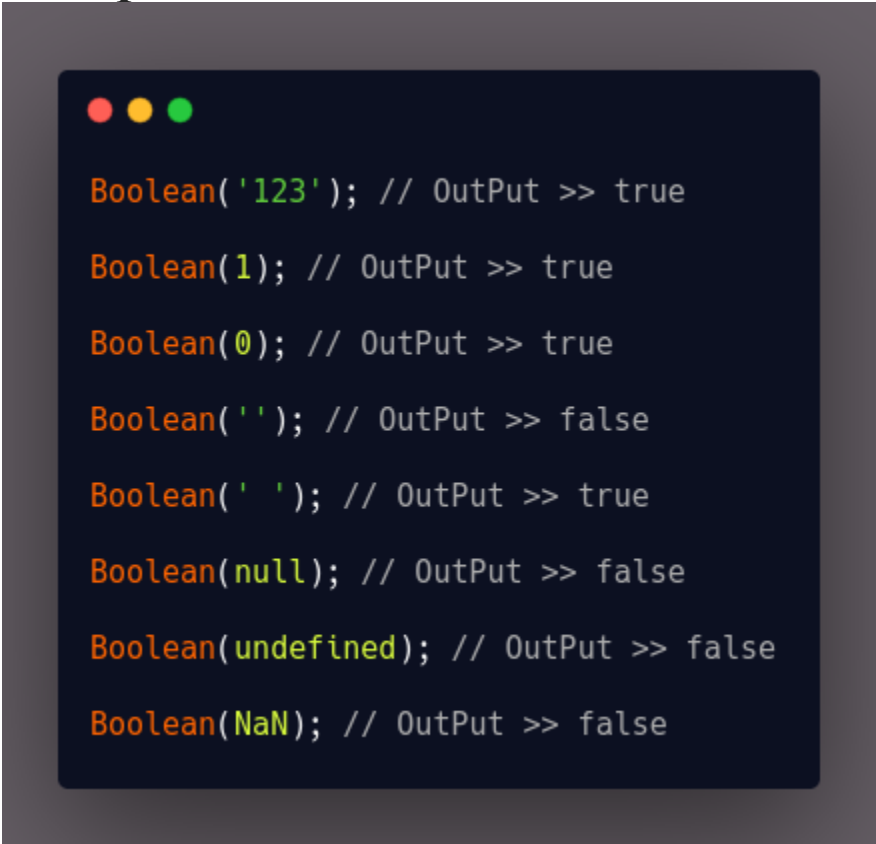
— To Boolean Conversion:

In boolean conversion, the values are converted to boolean values.

Syntax:

```
Boolean(valuesToBoolean)
```

Example:



```
Boolean('123'); // OutPut >> true
Boolean(1); // OutPut >> true
Boolean(0); // OutPut >> true
Boolean(' '); // OutPut >> false
Boolean(' '); // OutPut >> true
Boolean(null); // OutPut >> false
Boolean(undefined); // OutPut >> false
Boolean(NaN); // OutPut >> false
```

If we pass the values like empty string(""), null, undefined, and NaN then the result will always be false.

Type Coercion or Implicit Coercion :

Type coercion is similar to type conversion, but the only key difference is the coercion performed automatically or implicitly by the JavaScript engine.

For example, In the built-in `alert()` method when we pass any values the JavaScript engine will automatically convert those values into a string and display the result.

In type coercion, the same data types are used such as **String**, **number**, and **boolean**, converted into their desired type.

— To String Coercion :

When a string is added with the number or non-string value using the plus(+) operator then the output of the expression is always a string.

Example:



```
console.log('1' + 2); // Output >> 12
console.log(2 + '1' + true); // OutPut >> 21true
console.log('12' + undefined); // OutPut >> 12undefined
console.log('12' + null); // OutPut >> 12null
console.log('12' + NaN); // OutPut >> 12NaN
```

In type coercion, if the first operand is a string and the second operand is non-string then the result always prints as a string.


Here the '+' operator acts as a concatenation of two different operands.

— To Number Coercion:

The mathematical operations like Subtraction(-), Multiplication(*), Division(/), and Modulus(%) performed with the string then the output of the expression is converted into a number implicitly.

In number coercion, the plus(+) operator is not used.

Example :

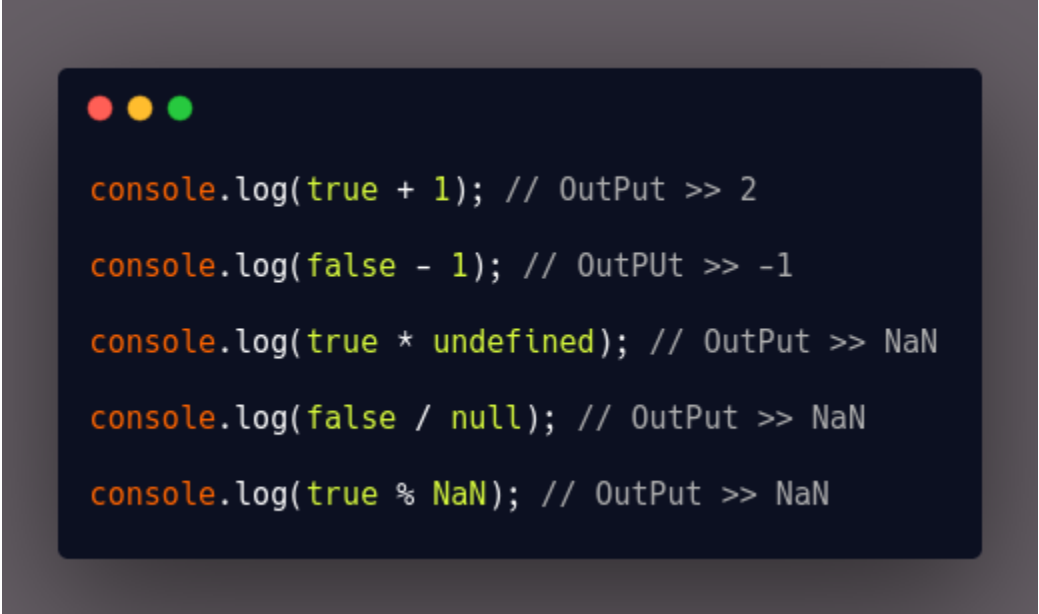


```
console.log('12' - 2); // OutPut >> 10
console.log('2' * 2); // OutPut >> 4
console.log('10' / 2); // OutPut >> 5
console.log('10' % 2); // OutPut >> 0
console.log('Hello' - 'World'); // OutPut >> NaN
console.log('Hello' * 2); // OutPut >> NaN
```

If any one of the string is non-numeric or both the strings are non-numeric then the JavaScript display the result as NaN i.e., not a number.

— To Boolean Coercion:

In boolean coercion, the boolean values such as true and false are converted to a number.



```
console.log(true + 1); // OutPut >> 2
console.log(false - 1); // OutPut >> -1
console.log(true * undefined); // OutPut >> NaN
console.log(false / null); // OutPut >> NaN
console.log(true % NaN); // OutPut >> NaN
```

In the above examples, JavaScript considers the **true** value as **1** and the **false** value as **0**.

Conclusion:

The conversion and coercion terms are quite confusing at first, but you have understood everything in this article. In the interview process, the interviewer may ask questions related to these types, and you will be able to answer them without any problem.