19) toLocalLowerCase()

- This method converts a string to lowercase letters.
- Return value is string.

Syntax

```
string.toLocaleLowerCase()
```

Example

```
var str = "HELLO WORLD";
var res = str.toLocaleLowerCase();
console.log(res);
```

20) toLocalUpperCase()

- This method converts a string to uppercase letters.
- Return value is string.

Syntax

```
string.toLocaleUpperCase()
```

Example

```
var str = "hello world";
var res = str.toLocaleUpperCase();
console.log(res)
```

21) toUpperCase()

- This method converts a string to uppercase letters.
- Return value is string.

Syntax

```
string.toUpperCase()
```

Example

```
var str = "Hello World!";
var res = str.toUpperCase();
console.log(res)
```

22) toLowerCase()

- This method converts a string to lowercase letters.
- Return value is string.

Syntax

```
string.toLowerCase()
```

Example

```
var str = "Hello World!";
var res = str.toLowerCase();
console.log(res)
```

23) toString()

- This method returns the value of a String object.
- Return value is string.

Syntax

```
string.toString();
```

Example

```
var str = "Hello World!";
var res = str.toString();
console.log(res)
```

24) trim()

- This method removes whitespace from both sides of a string.
- Whitespace in this context is all the whitespace characters (space, tab, no-break space, etc.) and all the line terminator characters (LF, CR, etc.).
- The "trimEnd()" & "trimStart()" method is used for removing whitespace from the end & start of a string.
- Return value is string.

Syntax

```
string.trim()
```

Example

```
var str = "       Hello World!       ";
var abc=str.trim();
console.log(abc)
```

25) valueOf()

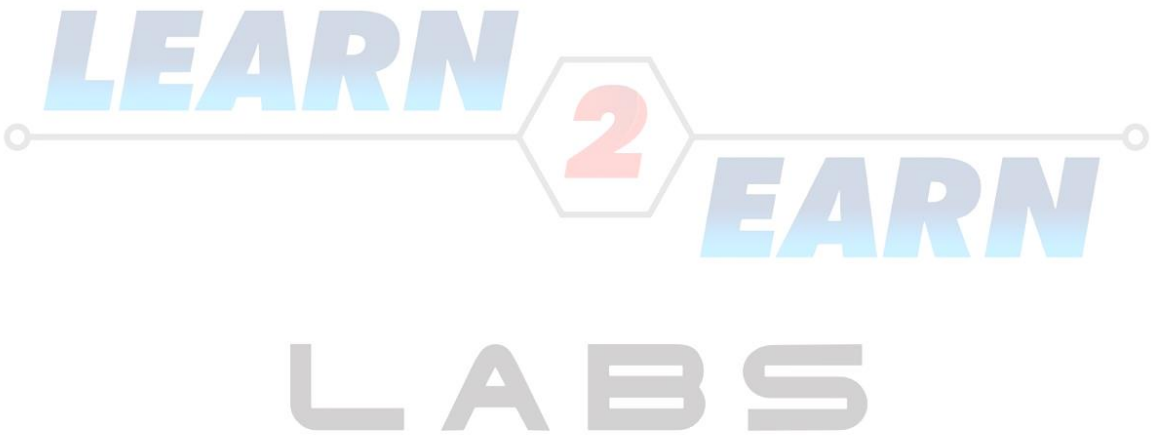
- This method Return the primitive value of a string object.

Syntax

```
string.valueOf()
```

Example

```
var str = "Hello World!";
var res = str.valueOf();
console.log(res)
```



Number Data Type

- The Number is a primitive data type in JavaScript.
- Number type represents integer, float, hexadecimal, octal or exponential value.
- First character in a Number type must be an integer value and it must not be enclosed in quotation marks.

Example

```
var int = 100;
var float = 100.5;
var hex = 0xfff;
var exponential = 2.56e3;
var octal = 0o30;

console.log(int)
console.log(float)
console.log(hex)
console.log(exponential)
console.log(octal)
```

"Number()" Constructor

- The Number JavaScript object is a wrapper object allowing you to work with numerical values.
- A Number object is created using the Number() constructor.

The primary uses of the Number object are :-

a) If the argument cannot be converted into a number, it returns NaN.
b) In a non-constructor context (i.e., without the new operator), Number can be used to perform a type conversion.

Example

```
var abc = Number("123") // Number Object
console.log(abc) // 123
```

Example

```
var abc = new Number("123") // Number constructor
console.log(abc) // Number{123}
```

Example

```
var num = new Number(23);
console.log(typeof num); // Object
console.log(num);
```

Example

```
var num = new Number();
num = 23;
console.log(typeof num); // Number
console.log(num); // 23
```

Number Properties

The 'number' data type in JavaScript have following properties :-

a) MAX_VALUE -- This proprty returns the maximum value of JavaScript.
b) MIN_VALUE -- This proprty returns the minimum Value of JavaScript.
c) NEGATIVE_INFINITY -- This proprty returns negative Infinity.
d) POSITIVE_INFINITY -- This proprty returns the positive Infinity.
e) NaN -- This property represents a value that is not a number.
f) MAX_SAFE_INTEGER -- The maximum safe integer in JavaScript is (2^53-1)
g) MIN_SAFE_INTEGER -- The minimum safe integer in JavaScript (-2^53-1)

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
</head>
<body>
      <script>
      console.log(Number.MAX_VALUE);
      console.log(Number.MIN_VALUE);
      console.log(Number.NaN);
      console.log(Number.POSITIVE_INFINITY);
      console.log(Number.NEGATIVE_INFINITY);
      console.log(Number.MAX_SAFE_INTEGER);
      console.log(Number.MIN_SAFE_INTEGER);

   </script>
</body>
</html>
```

Number Methods

a) isFinite()

This method Check whether a number is a finite i.e., a legal number.

Syntax

```
Number.isFinite(operand_value);
```

Example

```
console.log(Number.isFinite(123)) //true
console.log(Number.isFinite(-1.23)) //true
console.log(Number.isFinite(5-2)) //true
console.log(Number.isFinite(0)) //true
console.log(Number.isFinite('123')) //false
console.log(Number.isFinite('Hello')) //false
console.log(Number.isFinite('2005/12/12')) //false
console.log(Number.isFinite(Infinity)) //false
console.log(Number.isFinite(-Infinity)) //false
console.log(Number.isFinite(0 / 0)) //false
```

b) isInteger()

This method checks whether a value is an integer or not.

Syntax

```
Number.isInteger(operand_value);
```

Example

```
Number.isInteger(123) //true
Number.isInteger(-123) //true
Number.isInteger(5-2) //true
Number.isInteger(0) //true
Number.isInteger(0.5) //false
Number.isInteger('123') //false
Number.isInteger(false) //false
Number.isInteger(Infinity) //false
Number.isInteger(-Infinity) //false
Number.isInteger(0 / 0) //false
```

c) isNaN()

This method checks whether a value is NaN(Not a Number).

Syntax

```
Number.isNaN(operand_value);
```

Examples

```
isNaN(NaN);       // true
isNaN(undefined); // true
isNaN({});        // true
isNaN(true);      // false
isNaN(null);      // false
isNaN(37);        // false

// strings
isNaN('37');      // false: "37" is converted to the number 37 which is not NaN
isNaN('37.37');   // false: "37.37" is converted to the number 37.37 which is not NaN
isNaN("37,5");    // true
isNaN('123ABC');  // true:  parseInt("123ABC") is 123 but Number("123ABC") is NaN
isNaN('');        // false: the empty string is converted to 0 which is not NaN
isNaN(' ');       // false: a string with spaces is converted to 0 which is not NaN

// dates
isNaN(new Date());                // false
isNaN(new Date().toString());     // true
```

d) isSafeInteger()

- This method checks whether a value is a safe integer.
- A safe integer is an integer that can be exactly represented as an IEEE-754 double precision number (all integers from (2^53 - 1) to -(2^53 - 1)).

Syntax

```
Number.isSafeInteger(operand_value);
```

Example

```
Number.isSafeInteger(3);                  // true
Number.isSafeInteger(Math.pow(2, 53));    // false
Number.isSafeInteger(Math.pow(2, 53) - 1); // true
Number.isSafeInteger(NaN);                // false
Number.isSafeInteger(Infinity);           // false
Number.isSafeInteger('3');                // false
Number.isSafeInteger(3.1);                // false
Number.isSafeInteger(3.0);                // true
```

e) toExponential(x)

This method convert a number into an exponential notation.

Syntax

```
operand_value.toExponential();
```

Example

```
var numObj = 77.1234;
console.log(numObj.toExponential());  // logs 7.71234e+1
console.log(numObj.toExponential(4)); // logs 7.7123e+1
console.log(numObj.toExponential(2)); // logs 7.71e+1
console.log(77.1234.toExponential()); // logs 7.71234e+1
console.log(77 .toExponential());     // logs 7.7e+1
```

f) toFixed(x)

The toFixed() method converts a number into a string, keeping a specified number of decimals.

Syntax

```
operand_value.toFixed();
```

Example

```
var numObj = 12345.6789;
numObj.toFixed();       // Returns '12346': note rounding, no fractional part
numObj.toFixed(1);      // Returns '12345.7': note rounding
numObj.toFixed(6);      // Returns '12345.678900': note added zeros
(1.23e+20).toFixed(2);  // Returns '123000000000000000000.00'
(1.23e-10).toFixed(2);  // Returns '0.00'
2.34.toFixed(1);        // Returns '2.3'
2.35.toFixed(1);        // Returns '2.4'. Note it rounds up
2.55.toFixed(1);        // Returns '2.5'. Note it rounds down - see warning above
-2.34.toFixed(1);       // Returns -2.3 (due to operator precedence, negative number literals don't return a string...)
(-2.34).toFixed(1);     // Returns '-2.3'
```

g) toPrecision(x)

The toPrecision() method formats a number to a specified length.

Syntax

```
operand_value.toPrecision();
```

Example

```
var numObj = 5.123456;
console.log(numObj.toPrecision());    // logs '5.123456'
console.log(numObj.toPrecision(5));   // logs '5.1235'
console.log(numObj.toPrecision(2));   // logs '5.1'
console.log(numObj.toPrecision(1));   // logs '5'

numObj = 0.000123
console.log(numObj.toPrecision());    // logs '0.000123'
console.log(numObj.toPrecision(5));   // logs '0.00012300'
console.log(numObj.toPrecision(2));   // logs '0.00012'
console.log(numObj.toPrecision(1));   // logs '0.0001'
```

h) toString()

The "toString()" method converts a number to a string.

Syntax

```
operand_value.toString();
```

Example

```
var count = 10;
console.log(count.toString());    // displays '10'
console.log((17).toString());     // displays '17'
console.log((17.2).toString());   // displays '17.2'

var x = 6;
console.log(x.toString(2));       // displays '110'
console.log((254).toString(16));  // displays 'fe'
console.log((-10).toString(2));   // displays '-1010'
console.log((-0xff).toString(2)); // displays '-11111111'
```

i) valueOf()

The valueof() method in JavaScript is used to return the primitive value of a number.

Syntax

```
operand_value.valueOf();
```

Example

```
var num=213;
document.write("Output : " + num.valueOf());
```

Boolean

- Boolean is a primitive data type in JavaScript.
- The boolean type has only two values: true and false.
- This type is commonly used to store yes/no values: true means “yes, correct”, and false means “no, incorrect”.
- It is useful in controlling program flow using conditional statements like if..else, switch, while, do..while.

"Boolean()" Constructor

- The "Boolean()" creates a boolean object in JavaScript.
- There is a Boolean function, which can be used as an ordinary function which returns a boolean primitive.
- The Boolean function can also be used as a constructor with the 'new' keyword.
- The Boolean function "Boolean()" returns a primitive value, while Boolean constructor "new Boolean()" returns an object.

Syntax

```
Boolean() // boolean function
Or,
new Boolean() // boolean constructor
```

Example

```
var bool = new Boolean();
console.log(bool)
```

Example

```
var var1 = new Boolean(); // false{}
var var2 = new Boolean(0); // false{}
var var3 = new Boolean(null); // false{}
var var4 = new Boolean(''); // false{}
var var5 = new Boolean(false); // false{}
var var6 = new Boolean(true); // true{}
var var7 = new Boolean('true'); // true{}
var var8 = new Boolean('false'); // true{}
var var9 = new Boolean('Hello World'); // true{}
var var10 = new Boolean([]); // true{}
var var11 = new Boolean({}); // true{}
```

Example

```
Boolean(10 > 9)  // true
```

Example

```
let abc = true;
let bac = false;
console.log(typeof abc);
console.log(typeof bac);
```

Example

```
let abc = 4 > 1;
console.log( abc ); // true
```

Example

```
var a = 2, b = 5, c = 10;
alert(b > a) // Output: true
alert(b > c) // Output: false
```

Object Built-In Methods

- JavaScript have couple of different built-in methods for objects.
- These built-in methods simplifies our work.

List Of Built-In Methods

a) Object.entries()

- Object.entries() method in JavaScript returns an array consisting of enumerable property [key, value] pairs of the object which are passed as the parameter.
- Object.entries() is used for listing properties related to an object.

Example

```
let user = {
  name: 'Rohit',
  age: 30
};
console.log(Object.entries(user))
```

b) Object.create(proto,[objectProperties])

- The Object.create() method creates a new object, using an existing object as the prototype of the newly created object.
- Object.create() method is used to create a new object with the specified prototype object and properties.

Example

```
var person = Object.create({
   fname:"Erik",
   lname:"James"
});
console.log("Hello "+person.fname+" "+person.lname);
```

Example

```
var person = {
  isHuman: false,
  printIntroduction: function () {
    console.log(`My name is ${this.name}. Am I human? ${this.isHuman}`);
  }
};
const me = Object.create(person);
me.name = "Erik"; // "name" is a property set on "me", but not on "person"
me.isHuman = true; // inherited properties can be overwritten
me.printIntroduction();
```

c) Object.seal()

- Object.seal() is used for sealing objects and arrays.
- This method is used to make an object immutable.
- We cannot add new properties in the sealed objects.

Example

```
var obj1 = { fname: 'Rohit'};
Object.seal(obj1);
obj1.lname = 'Kelvin';
console.log(obj1.lname);
```

d) Object.isSealed()

Object.isSealed() method is used to determine if an object is sealed or not.

Example

```
var obj1 = { fname: 'Mohit'};
Object.seal(obj1);
obj1.lname = 'Kelvin';
console.log(Object.isSealed(obj1))
console.log(obj1.lname);
```

e) Object.keys()

Object.keys() is used for returning enumerable properties of a simple array/objects.

Example

```
var obj1 = { fname: 'Mohit',lname:"Singh"};
console.log(Object.keys(obj1));
```

f) Object.values()

This method returns the values of the object properties(keys).

Example

```
var obj1 = { fname: 'Mohit',lname:"Singh"};
console.log(Object.values(obj1));
```

g) Object.freeze()

- Object.freeze() is used to freeze an object.
- Freezing an object does not allow new properties to be added to an object and prevents from removing or altering the existing properties.
- Object.freeze() is used for freezing objects and arrays.
- Object.freeze() is used to make an object immutable.

Example

```
const obj1 = { fname: 'Mohit'};
Object.freeze(obj1);
obj1.fname = 'Kelvin';
console.log(obj1.fname);
```

h) Object.isFrozen()

Object.isfrozen() is used for checking whether an object is frozen or not.

Example

```
const obj1 = { fname: 'Mohit'};
Object.freeze(obj1);
console.log(Object.isFrozen(obj1));
obj1.fname = 'Kelvin';
console.log(obj1.fname);
```

i) Object.assign()

- Object.assign() is used to copy the values and properties from one or more source objects to a target object.
- Object.assign() is used for cloning an object.
- Object.assign() is used to merge object with same properties.

Example

```
var obj1 = { a: 10 };
var obj2 = Object.assign({}, obj1);
console.log(obj2);
```

Example

```
var obj1 = { a: 10 };
var obj2 = { b: 20 };
var obj3 = { c: 30 };
var obj4 = Object.assign(obj1, obj2, obj3);
console.log(obj4);
```

Example

```
var obj1 = { a: 10, b: 10, c: 10 };
var obj2 = { b: 20, c: 20 };
var obj3 = { c: 30 };
var obj4 = Object.assign({}, obj1, obj2, obj3);
console.log(obj4);
```

j) Object.is()

- Object.is() method is used to determine whether two values are same or not.
- Object.is() is used for comparison of two strings, numbers & objects.
- Two values can be same if they hold one of the following properties:-

a) If both the values are undefined.
b) If both the values are null.
c) If both the values are true or false.
d) If both the strings are of the same length with the same characters and in the same order.
e) If both the values are numbers and both are "+0".
f) If both the values are numbers and both are "-0".
g) If both the values are numbers and both are "NaN" or both non-zero and both not NaN and both have the same value.

Example

```
var check = Object.is('Mohit', 'Mohit');
console.log(check)
```

Example

```
var obj = { a: 100 };
var check = Object.is(obj, obj);
console.log(check);
```

k) Object.preventExtensions()

This method prevents adding properties to an object.

Example

```
var obj = { a: 100 };
var check = Object.preventExtensions(obj)
check.b = 200;
console.log(check);
```