Lesson:

Number

PW SKILLS

Topics

- Number properties
- Number methods

Number properties

In Javascript, Number properties are the properties of the global Number object. These properties can be used to access information about numbers, such as their maximum and minimum values, and to perform certain operations on numbers, such as converting them to different formats.

Here are some of the commonly used Number properties

MAX_VALUE
It is the properties of a number that represent the maximum numeric value representable in JavaScript. The maximum number value is approximately equal to 1.7976931348623157e+308.
Example

```
Unset
// MAX_VAlUE
console.log(Number.MAX_VALUE); // 1.7976931348623157e+308

// function to check whether the given number is within the range of Number
function validateNumber(num) {
  if (num < 0 || num > Number.MAX_VALUE) {
    return "Number out of valid range.";
  }
  return "Number is within the valid range.";
}

console.log( validateNumber(55))
```

MIN_VALUE
It is the properties of a number that represent the smallest number of values representable in JavaScript. This is the smallest positive value that can be represented, which is approximately equal to 5e-324
Example

```
Unset
// MIN_VALUE
console.log(Number.MIN_VALUE); // 5e-324


const smallestPositiveValue = Number.MIN_VALUE;

console.log("Is it zero? " + (smallestPositiveValue === 0));  // false
console.log("Is it less than 1? " + (smallestPositiveValue < 1));  // true
```

MAX_SAFE_INTEGER
It is the properties of a number that represent the maximum safe integer in JavaScript. The maximum safe number integer is approximately equal to 9007199254740991
Example

```
Unset
// MAX_SAFE_INTEGER
console.log(Number.MAX_SAFE_INTEGER); // 9007199254740991
```

MIN_SAFE_INTEGER
It is the properties of a number that represent the minimum safe integer in JavaScript. The minimum safe number integer is approximately equal to -9007199254740991
Example

```
Unset
// MIN_SAFE_INTEGER
console.log(Number.MIN_SAFE_INTEGER); // -9007199254740991
```

Number methods

In Javascript, Number methods are the methods of the global Number object. These methods can be used to perform various operations on numbers, such as converting them to different formats, rounding them, and checking their properties.

Here are some of the commonly used Number methods

1. toString()
2. toExponential()
3. toFixed()
4. toPrecision()
5. ValueOf()

1. toString() -
The toString() method converts a number to a string
Example

```
Unset
// syntax ---
<numberVaraibleName>.toString()
<numberVaraibleName>.toString(radix)

 // radix (optional) - An integer in the range 2 through 36 specifying the base
 to use for representing the number value. Defaults to 10.
```

```
// toString(radix) example
const numb = 11
const binaryString = numb.toString(2) // convert to binary (base 2)
console.log(binaryString) // 1011

const octalString = numb.toString(8) // convert to octal (base 8)
console.log(octalString) // 13
```

```
const numb = 123.456;

// Convert the number to a string in base 10.
const numbStr=numb.toString();
console.log(numbStr); // "123.456"
console.log(typeof numbStr); // string
```

2. toEXponential() -

The toExponential() method converts a number to a string in exponential notation. The exponential notation is a way of representing very large or very small numbers in a more compact form.

Example

```
Unset
// syntax ---
<numberVaraibleName>.toExponential()
<numberVaraibleName>.toExponential(fractionDigits)

// Note - fractionDigits (optional) - an integer specifying the number of
digits after the decimal point. Default to as many digits as necessary to
specify the number.
```

```
const numb = 10.555
console.log(numb.toExponential()) // 1.0555e+1
console.log(numb.toExponential(2)) // 1.06e+1

console.log(numb.toExponential(3)) // 1.055e+1
```

3. toFixed() -

The toFixed() method converts a number to a string with a fixed number of decimal places.

Example

```
Unset
// syntax ---
<numberVaraibleName>.toFixed()
<numberVaraibleName>.toFixed(digit)

// Note - digit(optional) - The number of digits to appear after the decimal
point; should be a value between 0 and 100, inclusive. If this argument is
omitted, it is treated as 0.

const numb = 10.555
console.log(numb.toFixed()) // 11
console.log(numb.toFixed(2)) // 10.55
console.log(numb.toFixed(3)) // 10.555
```

4. toPrecision() -

The toPrecision() method converts a number to a string with a specified precision. The precision is the number of significant digits in the number.

Example

```
Unset
// syntax ---
<numberVaraibleName>.toPrecision()
<numberVaraibleName>.toPrecision(precision)

// Note -precision(optional) - An integer specifying the number of significant
digits.

const numb = 10.555
console.log(numb.toPrecision()) // "10.555"
console.log(numb.toPrecision(2)) // "11"
console.log(numb.toPrecision(3)) // "10.6"
```

5. ValueOf() -

The valueOf() method returns the primitive value of a number object. This is the same value that would be returned if you were to use the typeof operator on the number object.

The valueOf() method is used internally in JavaScript to convert Number objects to primitive values

Example

```
Unset
// syntax ---
<numberVaraibleName>.valueOf()

const numObj = new Number(10);
console.log(typeof numObj); // object

const num = numObj.valueOf();
console.log(num); // 10
console.log(typeof num); // number
```