JavaScript Number Type

Summary: in this tutorial, you will learn about the JavaScript `Number` type and its useful methods for working with numbers effectively.

Introduction to JavaScript Number type

Besides the primitive number type, JavaScript also provides the `Number` reference type for numeric values.

To create a `Number` object, you use the `Number` constructor and pass in a number value as follows:

```
var numberObject = new Number(100);
```

This example defines a `numberObject` with a numeric value of `100`.

To get the primitive value out of a `Number` object, you use the `valueOf()` method as follows:

```
console.log(numberObject.valueOf()); // 100
```

To get a number value as a string, you use the `toString()` or `toLocaleString()` methods.

The `toString()` method accepts an optional argument that determines the radix in which to present the number. The radix (or base) is the number of unique digits that represent numbers in a positional numeral system.

For example, the decimal system uses ten digits from 0 through 9, therefore, the radix is 10.

See the following example:

```
var aNumber = new Number(10);
console.log(aNumber.toString()); // "10"
```

In this example the `aNumber` has a primitive value of 10, therefore, the `toString()` method returns 10 in the decimal system.

However, the following example returns the binary form of the `aNumber` variable.

```
console.log(aNumber.toString(2)); // "1010"
```

If you call a method on a primitive number value, JavaScript will convert it into a `Number` object temporarily. This feature is called primitive wrapper types in JavaScript. For example:

```
let x = 10;
console.log(x.toString(16)); // "a"
```

Formatting numbers

To format a number with a specified number of decimal points, you use the `toFixed()` method.

The `toFixed()` method accepts an argument that indicates how many decimal points should be used.

```
numberObject.toFixed(decimalPlaces);
```

The `toFixed()` method returns the corresponding string of the number using fixed-point notation. Here is an example.

```
var distance = 19.006
console.log(distance.toFixed(2)); // 19.01

distance = 19.004;
console.log(distance.toFixed(2)); // 19.00
```

It's important to note that web browsers may use rounding methods differently. Therefore, you should be careful when using the `toFixed()` method, especially for applications that deal with monetary values.

To format a number in e-notation, you use the `toExponential()` method as shown in the following example.

```
var x = 10, y = 100, z = 1000;

console.log(x.toExponential());
console.log(y.toExponential());
console.log(z.toExponential());

// "1e+1"
// "1e+2"
// "1e+3"
```

To get a string representation of a number object to the specified precision, you use the `toPrecision()` method.

```
numberObject.toPrecision(precision);
```

The `precision` argument determines the number of significant digits.

The `toPrecision()` method returns the string representation of a `Number` object in exponential notation or fixed point rounded to precision significant digits.

Notice that if you omit the precision argument, the `toPrecision()` method will behave like the `toString()` method. See the following examples:

```
let x = 9.12345;

console.log(x.toPrecision());    // '9.12345'
console.log(x.toPrecision(4));   // '9.123'
console.log(x.toPrecision(3));   // '9.12'
console.log(x.toPrecision(2));   // '9.1'
console.log(x.toPrecision(1));   // '9'
```

The e-notation may be returned in some cases for example:

```
x = 123.5;
console.log(x.toPrecision(2)); // "1.2e+2"
```

JavaScript Number object vs. primitive number

The following table illustrates the differences between a `Number` object and a primitive number:

Operator	Number object	Number value
typeof	"object"	"number"
instanceof Number	true	false

And here are examples:

```
let numberObject = new Number(10);
let number = 10;

// typeof
console.log(typeof numberObject);
console.log(typeof number);
// instanceof
console.log(numberObject instanceof Number); // true
console.log(number instanceof Number); // false
```

In this tutorial, you have learned about the `Number` type and some useful methods for formatting numbers.