Title: Write a note on number data type

Answer:

JavaScript has only one number type, there is no separate designation for integers and floating-point numbers. Because of this, numbers can be written in JavaScript with or without decimals:

```
let num1 = 93;
let num2 = 93.00;
```

In both cases above, the data type is a number and is the same regardless of whether or not the number has decimal points.

Scientific exponential notation can be used in JavaScript to abbreviate very large or small numbers, as in the following examples:

```
let num3 = 987e8;       // 98700000000
let num4 = 987e-8;      // 0.00000987
```

Numbers in JavaScript are considered to be accurate up to 15 digits. That means that numbers will be rounded after the 16th digit is reached:

```
let num5 = 999999999999999;     // remains as 999999999999999
let num6 = 9999999999999999;    // rounded up to 10000000000000000
```

In addition to representing numbers, the JavaScript number type also has three symbolic values available:

- `Infinity` — a numeric value that represents a positive number that approaches infinity
- `-Infinity` — a numeric value that represents a negative number that approaches infinity
- `NaN` — a numeric value that represents a non-number, standing for not a number

`Infinity` or `-Infinity` will be returned if you calculate a number outside of the largest possible number available in JavaScript. These will also occur for values that are undefined, as when dividing by zero:

```
let num7 = 5 / 0;   // will return Infinity
let num8 = -5 / 0;  // will return -Infinity
```

In technical terms, `Infinity` will be displayed when a number exceeds the number `1.797693134862315E+308`, which represents the upper limit in JavaScript.

Similarly, `-Infinity` will be displayed when a number goes beyond the lower limit of `-1.797693134862316E+308`.

The number `Infinity` can also be used in loops:

```
while (num9 != Infinity) {
    // Code here will execute through num9 = Infinity
}
```

For numbers that are not legal numbers, `NaN` will be displayed. If you attempt to perform a mathematical operation on a number and a non-numeric value, `NaN` will be returned. This is the case in the following example:

```
let x = 20 / "Shark";   // x will be NaN
```

Since the number `20` cannot be divided by the string `"Shark"` because it cannot be evaluated as a number, the returned value for the `x` variable is `NaN`.

However, if a string can be evaluated as a numeric value, the mathematical expression can be performed in JavaScript:

```
let y = 20 / "5";   // y will be 4
```

In the above example, since the string `"5"` can be evaluated as a numeric value in JavaScript, it is treated as such and will work with the mathematical operator for division, `/`.

When assigning the value `NaN` to a variable used in an operation, it will result in the value of `NaN`, even when the other operand is a legal number:

```
let a = NaN;
let b = 37;
let c = a + b;  // c will be NaN
```

There is only one number data type in JavaScript. When working with numbers, any number you enter will be interpreted as the data type for numbers; you are not required to declare what kind of data type you are entering because JavaScript is dynamically typed.

Tags: javascript, numbers