JS | Numbers as Data Types

Escribiendo grandes números

Si tienes que trabajar con números extremadamente grandes, puedes agregar la letra ***e*** al número especificando cuántos ceros hay en el número.

```
const someVar = 1000000000;

○

const someVar = 1e9;
```

Para calcularlo:

```
1e9 = 1 * 1000000000
4.5e6 = 4.5 * 1000000
```

Esto es aplicable también a los muy pequeños

```
const someSmallNumber = 1e-3;
console.log(someSmallNumber);
//0.001
```

Así funciona:

```
1e-3 = 1/1000 ===> 0.001
4.5e-6 = 4.5/1000000 ===> 0.0000045
```

Métodos numéricos para redondear

Funciones para redondear:

Math.floor() => redondea hacia abajo:

3.1 se convierte en 3

-1.1 se convierte en -2

Math.ceil() => redondea hacia arriba:

3.1 se convierte en 4

-1.1 se convierte en -1

Math.round() => redondea al entero más cercano:

3.1 se convierte en 3

3.6 se convierte en 4

Redondear un número a uno, dos o tres dígitos después del punto

- Multiplica y divide, Math.round()

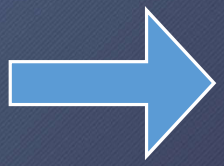
```
let anyNumber = 5.679345;
let roundedToOne = Math.round(anyNumber*10)/10;
let roundedToTwo = Math.round(anyNumber*100)/100;
let roundedToThree = Math.round(anyNumber*1000)/1000;

console.log(roundedToOne); // <== 5.7
console.log(roundedToTwo); // <== 5.68
console.log(roundedToThree); // <== 5.679
```

Método .toFixed(n)

para redondear el número a ***n*** dígitos después del punto.

```
let anyNumber = 5.679345;
let roundedToOne = anyNumber.toFixed(1);
console.log(roundedToOne); // <== "5.7"
console.log(typeof roundedToOne); // <== string
let roundedToTwo = anyNumber.toFixed(2);
console.log(roundedToTwo); // <== "5.68"
```



Notar que son string.

Para convertir un string (como “5,7”) en el número mismo, utilizar **Number()**

```
let anyNumber = 5.679345;
let roundedToOne = anyNumber.toFixed(1);
console.log(Number(roundedToOne)); // <== 5.7
console.log(typeof Number(roundedToOne)); // <== number
```

Cálculo no preciso

- *infinity*

Si intentamos guardar o manipular un número extremadamente grande, nos retornará infinity

```
console.log(1e400);
// <== Infinity
```

Otros métodos matemáticos sobre números

- **Math.random()** => devuelve un número aleatorio de 0 a 1 (incluido 0 pero no incluido 1)

```
console.log(Math.random()); // <== 0.010086087097095797
console.log(Math.random()); // <== 0.24143918045188073
console.log(Math.random()); // <== 0.23920890331219713
```

Cada vez te dará un número aleatorio diferente

Otros métodos matemáticos sobre números

- Math.max(a, b, c ...) / Math.min(a, b, c ...)

devuelve el mayor / menor del número arbitrario de argumentos.

```
console.log(Math.max(2, 8, -10, 0, -4)); // <== 8
console.log(Math.min(1, 2, 0, -5)); // <== -5
```

Otros métodos matemáticos sobre números

- Math.pow (n, power)

devuelve *n* elevado a la potencia dada

```
console.log(Math.pow(2, 3)) // <== 8
```

0.1 + 0.2 = 0.3

¿Esto es verdadero?

- Fuera de los cálculos

Esta afirmación es falsa.

Entonces, ¿cuánto es 0.1 + 0.2?

```
console.log(0.1 + 0.2);
// 0.30000000000000004
```

en el sistema de numeración binaria, 1/10 es una fracción binaria sin fin.

¿Cómo resolver este problema?

Use el método .toFixed(n) (pero tenga en cuenta que devuelve un string)