

Why Javascript Numbers Are Weird

(And How to Fix It)



@frontstuff_io



@frontstuff_io



Dinero.js is a library for working with monetary values in JavaScript.

license

MIT

build

passing

npm

v1.8.0

coverage

98%



semantic-release

[See full API docs](#)

Features

- Immutable and chainable API.
- Global settings support.
- Extended formatting and rounding options.
- Native Intl support (no additional locale files).
- Currency conversion.

Download/install

Dinero.js provides builds for different environments. It also comes with polyfilled versions for older browsers.

The recommended way of install is via [npm](#) or [Yarn](#):



Dinero.js is a library for working with monetary values in JavaScript.

license

MIT

build

passing

npm

v1.8.0

coverage

98%



semantic-release

[See full API docs](#)

Features

- Immutable and chainable API.
- Global settings support.
- Extended formatting and rounding options.
- Native Intl support (no additional locale files).
- Currency conversion.

Download/install

Dinero.js provides builds for different environments. It also comes with polyfilled versions for older browsers.

The recommended way of install is via [npm](#) or [Yarn](#):



Dinero.js is a library for working with monetary values in JavaScript.

license

MIT

build

passing

npm

v1.8.0

coverage

98%



semantic-release

[See full API docs](#)

Features

- Immutable and chainable API.
- Global settings support.
- Extended formatting and rounding options.
- Native Intl support (no additional locale files).
- Currency conversion.

Download/install

Dinero.js provides builds for different environments. It also comes with polyfilled versions for older browsers.

The recommended way of install is via [npm](#) or [Yarn](#):

We use
a decimal system.



@frontstuff_io

0

12

3 4 5

6 7 8 9



@frontstuff_io

3 4 5



@frontstuff_io

3 4 5



300

+



40

+



5



@frontstuff_io

3 4 5



300

3×100



40

4×10



5

5×1



@frontstuff_io

3 4 5



300

+

40

+

5

3×100

4×10

5×1

3×10^2

4×10^1

5×10^0



@frontstuff_io

Computers use a
binary system.



@frontstuff_io



Canon EOS M50 Mirrorless Camera Kit w/EF-M15-45mm and 4K Video - Black

★★★★★ 512

[Black](#)

\$1001010111

Ships to France

Only 6 left in stock - order soon.

More Buying Choices

(25 used & new offers)

Also available in White



Fujifilm 16643000 X100V Digital Camera - Black

★★★★★ 5

[Black](#)

\$10101110111

Ships to France

In stock on March 28, 2020.

More Buying Choices

(3 new offers)

Also available in Silver



Kodak PIXPRO Friendly Zoom FZ53-BK 16MP Digital Camera with 5X Optical Zoom and 2.7" LCD Screen (Black)

★★★★★ 592

[Black](#)

\$1000101

Ships to France

More Buying Choices

(24 used & new offers)

Price may vary by color



Computers must convert
decimal to binary.



@frontstuff_io

5



@frontstuff_io

5

1

$1 \times 2^2 (4)$

$$5 - 4 = 1$$



@frontstuff_io

5

1 0

$0 \times 2^1 (2)$

$1 - 0 = 1$



@frontstuff_io

5

1 0 1

$1 \times 2^0 (1)$

$1 - 1 = 0$



@frontstuff_io

1 0 1



4

+

0

+

1



@frontstuff_io

Double-precision floating-point format



@frontstuff_io

Java

```
1 System.out.println(.1 + .2);
2
3 // 0.3000000000000004
```

Python

```
1 print(.1 + .2)
2
3 # 0.3000000000000004
```

Ruby

```
1 puts 0.1 + 0.2
2
3 # 0.3000000000000004
```

C#

```
1 Console.WriteLine("{0:R}", .1 + .2);
2
3 // 0.3000000000000004
```

JavaScript

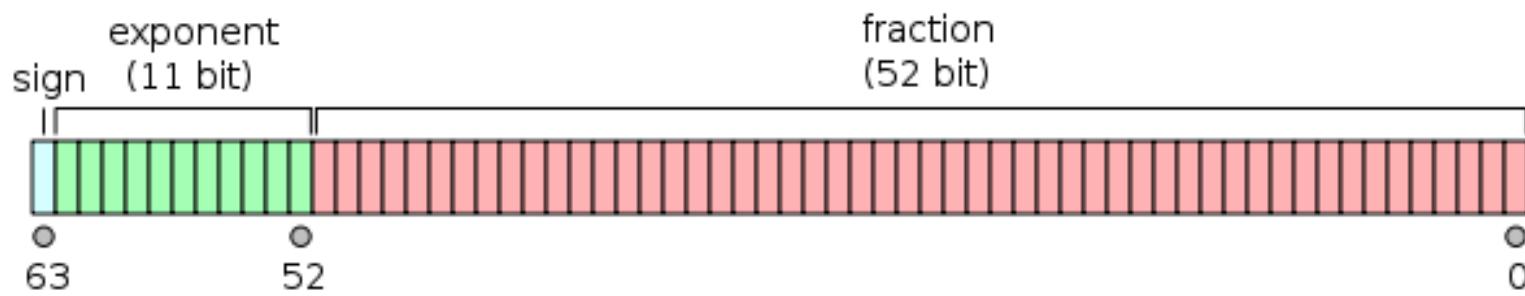
```
1 console.log(.1 + .2);
2
3 // 0.3000000000000004
```

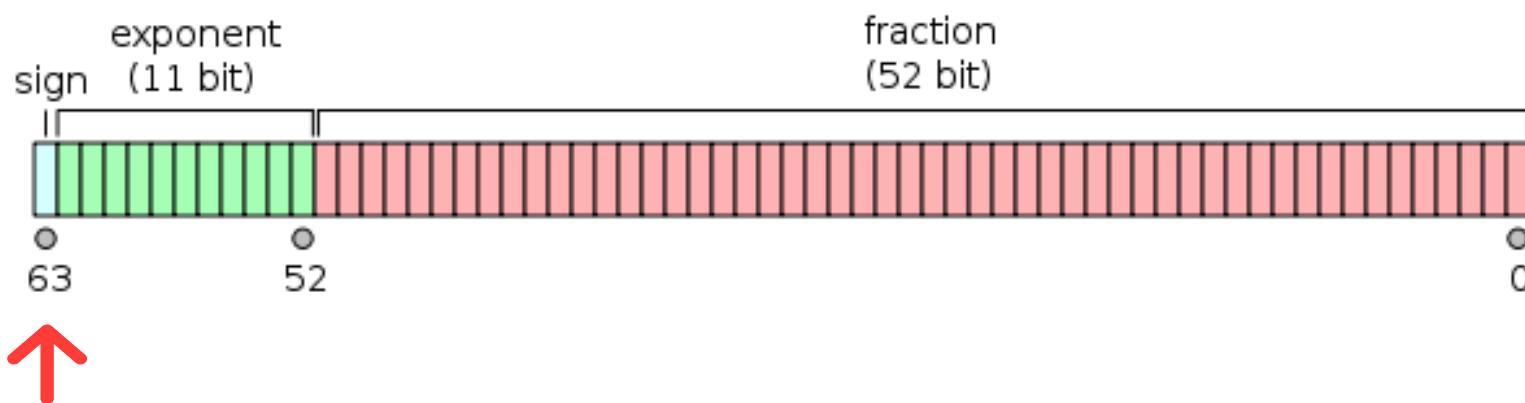
Before ES2020

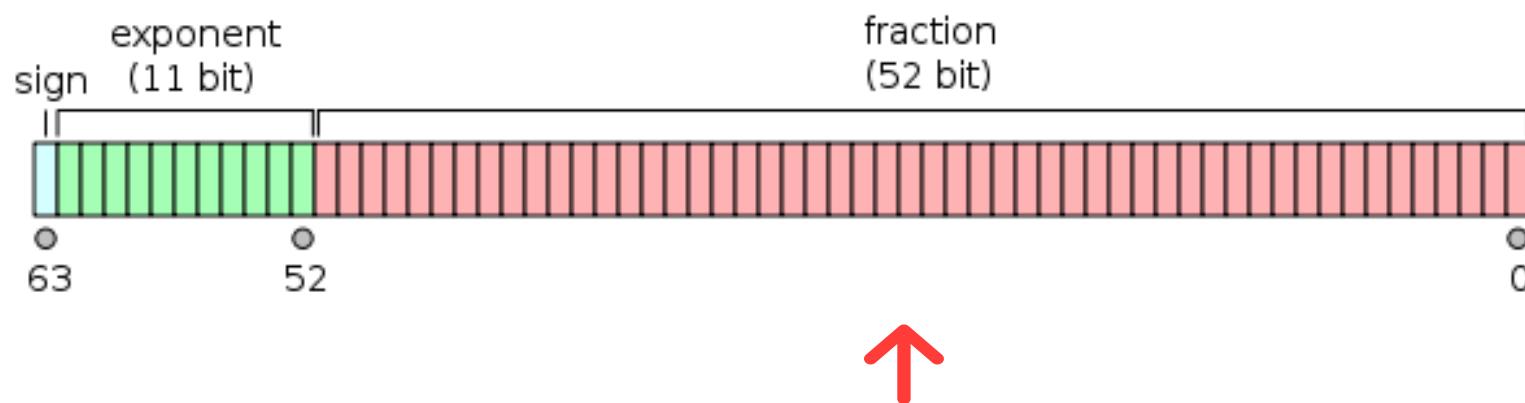
JavaScript only had
the number type

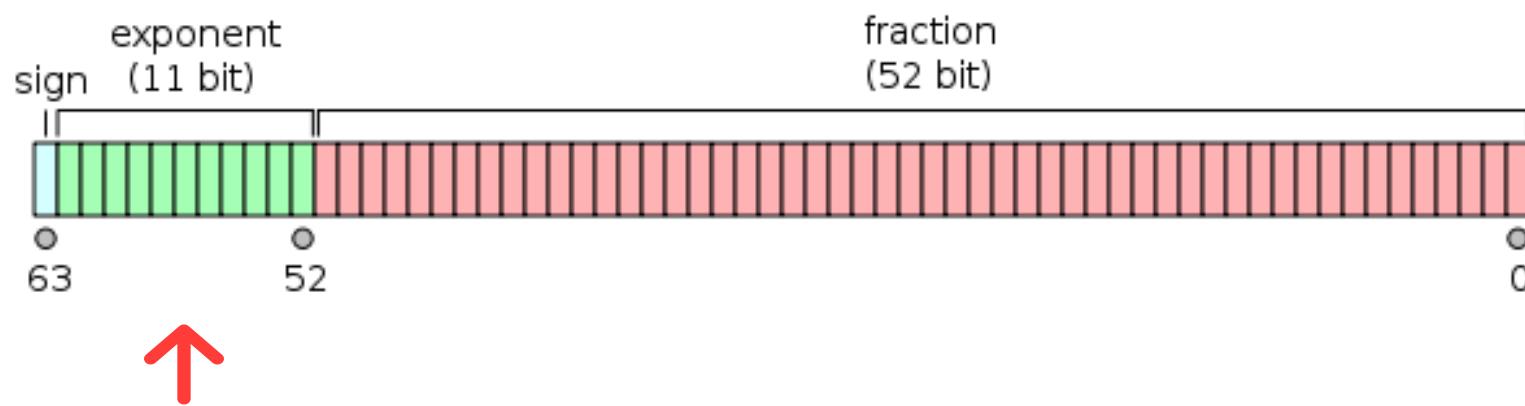


@frontstuff_io









5



@frontstuff_io

0.75



@frontstuff_io

0.75

1

$$0.75 \times 2 = 1.5$$



@frontstuff_io

0.75

1 1

$$0.5 \times 2 = 1$$



@frontstuff_io

0.75

0.11



@frontstuff_io

5.75



@frontstuff_io

A diagram illustrating a 64-bit binary number. The bits are represented by vertical bars of varying colors: green, light blue, and red. The first bit is green and labeled '1'. The next seven bits are light blue and labeled '0'. The eighth bit is green and labeled '1'. The ninth bit is light blue and labeled '0'. The tenth bit is red and labeled '1'. The remaining 54 bits are light blue and labeled '0'. A black dot is positioned below the 10th bit.

```
console.log(.1 + .2);  
// 0.3000000000000004
```

0.1



@frontstuff_io

0.1 + 0.2



@frontstuff_io

Same goes with
Large integers



@frontstuff_io

```
Number.MAX_SAFE_INTEGER;
```

```
// 9007199254740991
```

```
Number.MAX_SAFE_INTEGER + 2;
```

```
// 9007199254740992
```

9007199254740991

-9007199254740991



@frontstuff_io

A binary sequence diagram showing a sequence of bits. The first bit is blue, followed by four green bits (0, 1, 0, 0), then three red bits (1, 1, 0), and finally 31 black bits (1).

What to do instead?



@frontstuff_io

$$(1 + 2) / 10^1$$


@frontstuff_io

```
Dinero({ amount: 500, currency: "USD" });  
  
// represents $5
```

10n



@frontstuff_io

`9007199254740991n + 2n;`

`// 9007199254740993n`

Arbitrary precision math is
slower than floating point math.



@frontstuff_io

Arbitrary precision integers
are still new.



@frontstuff_io

bartaz.github.io/ieee754-visualization

0.3000000000000004.com

floating-point-gui.de



@frontstuff_io



JS

Everything you
never wanted to
know about
JavaScript **numbers***

Explained by

Bartek Szopka, @bartaz

*and you didn't know you could ask

youtu.be/MqHDDtVYJRI



@frontstuff_io

Thank you!



sarahdayan.dev



@frontstuff_io