Hve@hvks.com

JavaScript Fraction Library

Version 1.0

By Henrik Vestermark

Hve@hvks.com

Contents

Hve@hvks.com

Fraction

Support for Fraction number arithmetic in JavaScript. A fraction number has a numerator n and a denominator d. Usually we defined this as (n:d).

Fraction Constructor

---

new Fraction(*numerator*,*denominator*) // Constructor
Fraction(value) // Conversion

Arguments

numerator The numerator part of a Fraction number
denominator The denominator part of a Fraction number

If the denominator argument is omitted it is treated as one
If there are no arguments, it's treated as a Fraction zero
If *Fraction* is invoked as a conversion, the value parameter is converted to a *Fraction number* and returned.

Returns

Returns a Fraction object initialize with the numerator and denominator value. If *Fraction* is invoked as a conversion, the *value* parameter is converted to a *Fraction number* and returned. If *value* is another Fraction number then that is returned if *value* is undefined a *Fraction.zero* is returned.

Javascript BigInt and Number.

A fraction number can handle both the regular JavaScript Numbers and JavaScript BigInt numbers, but not mixed in a fraction object. If either the numerator or the denominator is of type BigInt they will both be converted to a BigInt number. This ensures that both the numerator and the denominator will be of the same JavaScript type of either "number" or "bigint". See Below table

Type of Numerator/denominator	Number	BigInt
Number	Number	BigInt
BigInt	BigInt	BigInt

Normalized Numbers.

Notice that a fraction number is always return as normalized after an arithmetic operation. E.g. addition, subtraction, multiplication or division. A normalized number is reduced to its simple form. E.g. 20:30 is normalized to 2:3. 3:12 is normalized to 1:4 etc.

Fraction sign

A denominator sign is always stored as a positive sign. A numerator sign can be either positive or negative. If the denominator sign is set to negative, the negative sign is propagated to the numerator by multiplying it with -1. E.g.

(1:2) =>(1:2)
(-1:2)=> (-1:2)
(1:-2)=> (-1:2)
(-1:-2)=>(1:2)

Methods

abs	Return the absolute value of a Fraction number
denominator	Return the Denominator part of the Fraction number.
inverse	Swap the numerator and denominator.
negate	Return a new Negated Fraction Object.
nominator	Return the nominator part of the Fraction number.
toExponential	Converts a Fraction number to a string using exponential notation with the specified number of digits after the Decimal place.
toFixed	Converts a Fraction number to a string that contains a specified number of digits after the decimal place.
toPrecision	Convert a Fraction number to a string using the specified number of precision digits. Uses exponential or fixed point notation depending on the size of the number and the number of significant digits specified.
toString	Convert a Fraction number to a string using a specified radix(base)
typeof	Return the type of the fraction e.g. return either “number” or “bigint”
valueOf	The primitive real number value of this Fraction number object.

Functions

abs()	Return the absolute value of a Fraction number
add()	Return the addition of two Fraction numbers
div()	Return the division of two Fraction numbers.
equal()	Return the Boolean value (true, false) of the equality of two Fraction numbers.
gcd()	Return the greatest coming divisor of two numbers.

greater()	Return the Boolean value (true, false) of the' >' of two Fraction numbers.
greaterequal()	Return the Boolean value (true, false) of the' >=' of two Fraction numbers.
less()	Return the Boolean value (true, false) of the' <' of two Fraction numbers.
lessequal()	Return the Boolean value (true, false) of the' <=' of two Fraction numbers.
mul()	Return the product of two Fraction numbers.
notequa()	Return the Boolean value (true, false) of the non-equality of two Fraction numbers.
reduce()	Return the reduce fraction. E.g. a fraction where the whole() number has been subtracted.
sub()	Return the difference between two Fraction numbers.
whole()	Returns the whole integer of the fraction

Constants

zero	return a new Fraction(0,1) object. Notice the denominator is 1.
one	return a new Fraction(1,1) object

Miscellaneous

parseFraction() Parse a Fraction float or BigInt number string

API

Fraction.abs()

Return the absolute value of the Fraction number

Synopsis

Fraction object.abs()

Returns

The absolute value of the Fraction number is returned.

Example

```
var z = new Fraction(-3, 4 );
z.abs()                          // result (3:4)
```

See Also

Fraction.abs()

Fraction.add()

Add two Fraction numbers

Synopsis

Fraction.add(a,b)

Arguments

a,b	The Fraction numbers to be added.

Returns

The result of the Fraction addition.

Example

```
var z = new Fraction(3,4);
var y=new  Fraction(1,2);

Fraction.add(z,y)              // result (5:4)
```

See Also

Fraction.div(), Fraction.mul(), Fraction.sub()

Fraction.denominator()

Return or set the denominator part of the Fraction number

Synopsis

Fraction object.denominator(*d*)

Arguments

d	The optional denominator value when setting the denominator to a new value. If omitted the call returns the actual denominator value.

Returns

Return the denominator part of the Fraction number.

Example

```
var z = new Fraction(3,4);
z.denominator();              // result 4
z.denominator(8);             // result 8 and z becomes (3:8)
```

See Also

Fraction.numerator()

Fraction.div()

Divide two Fraction numbers

Synopsis

Fraction.div(a,b)
Arguments

a,b The Fraction numbers to be divided.

Returns

The result of the Fraction division a/b.

Example

```
var z = new Fraction(3,4);
var y=new  Fraction(1,2);

Fraction.div(z,y)                    // result (3:2)
```

See Also

Fraction.add(), Fraction.mul(), Fraction.sub()

Fraction.equal()

Compare two Fraction numbers for equality

Synopsis

Fraction.equal(a,b)

Arguments

a,b The Fraction numbers to be compare for

Returns

The Boolean value of the equal comparison.

Example

```
var z = new Fraction(3,4);
var y=new  Fraction(1,2);

if( Fraction.equal(z,y))…          // result false
if( Fraction.equal(z,z))…          // result true
if( !Fraction.equal(z,y))…         // result true ! to do “notequal” comparison
```

See Also

Fraction.notequal(), Fraction.less(), Fraction.lessequal(), Fraction.greater(), Fraction.greaterequal()

Fraction.gcd()

Calculate the greates common divisior (gcd) of two numbers

Synopsis

Fraction.gcd(a,b)

Arguments

a,b the numbers to calculate the gcd value

Returns

Return the gcd of the two numbers.

Example

```
Fraction.gcd(32,4);          // result 4
```

See Also

Fraction.greater()

Compare two Fraction numbers for greater

Synopsis

Fraction.greater(a,b)

Arguments

a,b The Fraction numbers to be compare for greater

Returns

The Boolean value of the greater comparison.

Example

```
var z = new Fraction(3,4);
var y=new  Fraction(1,2);

if( Fraction.greater(z,y))…          // result false
if( Fraction.greater(z,z))…          // result false
if( !Fraction.greater(z,y))…         // result true ! to do “lessequal” comparison
```

See Also
Fraction.notequal(), Fraction.equal(), Fraction.lessequal(), Fraction.less(), Fraction.greaterequal()

Fraction.greaterequal()

Compare two Fraction numbers for greater equal

Synopsis

Fraction.greaterequal(a,b)

Arguments

a,b The Fraction numbers to be compare for greater equal

Returns

The Boolean value of the greater equal comparison.

Example

```
var z = new Fraction(3,4);
var y=new  Fraction(1,2);

if( Fraction.greaterequal(z,y))…        // result false
if( Fraction.greaterequal(z,z))…        // result true
if( !Fraction.greaterequal(z,y))…       // result true ! to do “less” comparison
```

See Also
Fraction.notequal(), Fraction.equal(), Fraction.lessequal(), Fraction.less(), Fraction.greater()

Fraction.inverse()

Inverse the Fraction number.

Synopsis

Fraction object.inverse()

Returns

Return the inverse Fraction number. Swapping the numerator and denominator

Example

```
var z = new Fraction(3,4);
z.inverse()                             // result (4:3)
```

See Also

Fraction.less()

Compare two Fraction numbers for less

Hve@hvks.com

Synopsis

Fraction.less(a,b)

Arguments

a,b The Fraction numbers to be compare for less

Returns

The Boolean value of the less comparison.

Example

```
var z = new Fraction(3,4);
var y=new  Fraction(1,2);

if( Fraction.less(z,y))…          // result true
if( Fraction.less(z,z))…          // result false
if( !Fraction.less(z,y))…         // result false ! to do “greaterequal” comparison
```

See Also
Fraction.notequal(), Fraction.equal(), Fraction.lessequal(), Fraction.greater(), Fraction.greaterequal()

Fraction.lessequal()

Compare two Fraction numbers for less equal

Synopsis

Fraction.lessequal(a,b)

Arguments

a,b The Fraction numbers to be compare for less equal

Returns

The Boolean value of the less equal comparison.

Example

```
var z = new Fraction(3,4);
var y=new  Fraction(1,2);

if( Fraction.lesseual(z,y))…              // result true
if( Fraction.lessequal(z,z))…             // result true
if( !Fraction.lessequal(z,y))…  // result false ! to do “greater” comparison
```

See Also

Fraction.notequal(), Fraction.equal(), Fraction.lesse(), Fraction.greater(), Fraction.greaterequal()

Fraction.mul()

Multiply two Fraction numbers

Synopsis

Fraction.mul(a,b)

Arguments

a,b	The Fraction numbers to be multiplied.

Returns

The result of the Fraction multiplication.

Example

```
var z = new Fraction(3,4);
var y=new  Fraction(1,2);

Fraction.mul(z,y)                 // result (3:8)
```

See Also

Fraction.add(), Fraction.div(), Fraction.sub()

Fraction.negate()

Return the negated Fraction number

Synopsis

Fraction object.negate()

Returns

Return the negated Fraction number.

Example

```
var z = new Fraction(3,4);
z.negate()                    // result (-3:4)
```

See Also

Fraction.notequal()

Compare two Fraction numbers for non-equality

Synopsis

Fraction.notequal(a,b)

Arguments

a,b	The Fraction numbers to be compare for

Returns

The Boolean value of the not equal comparison.

Example

```
var z = new Fraction(3,4);
var y=new  Fraction(1,2);

if( Fraction.notequal(z,y))…          // result true
```

```
if( Fraction.notequal(z,z))…          // result false
if( !Fraction.notequal(z,y))…         // result false ! to do “equal” comparison
```

See Also

Fraction.equal(), Fraction.less(), Fraction.lessequal(), Fraction.greater(), Fraction.greaterequal()

Fraction.one

Return Fraction one

Synopsis

Fraction.one

Returns

The Fraction constant one (1:1).

Example

```
var z = Fraction.one;          // z=(1:0)
```

See Also

Fraction.zero

Fraction.numerator()

Return or set the numerator part of the Fraction number

Synopsis

Fraction object.numerator(*n*)

Arguments

n	The optional numerator value when setting the numerator to a new value. If omitted the call returns the actual numerator value.

Returns

Return the numerator part of the Fraction number.

Example

```
var z = new Fraction(3,4);
z.numerator();                  // result 3
z.numerator(5);                 // return 5 and set z to (5:4)
```

See Also

Fraction.denominator()

Fraction.reduce()

Return the whole integer of the fraction number and subtract the while number from the fraction.

Synopsis

Fraction.reduce()

Returns

An integer number which value is the integer value of the *Fraction number*. The return number can be either “number” or “bigint” number depending on which type the fraction is. This is the same as returning Math.trunc(numerator/denominator). The fraction is subtracted with the whole number.

Example

```
var z = new Fraction( 12,5 );
z.reduce();             // result  2 and the fraction is reduced to (2:5)
var y=new Fraction(12n:4n);
y.reduce()              // result 3n (bigint) and the fraction is reduced to (0:4n)
```

See Also

Fraction.typeof(), Faction.whole()

Fraction.sub()

Subtract two Fraction numbers

Synopsis

Fraction.sub(a,b)

Arguments

a,b The Fraction numbers to be subtracted.

Returns

The result of the Fraction subtraction.

Example

```
var z = new Fraction(3,4);
var y=new  Fraction(1,2);

Fraction.sub(z,y)                // result (1:4)
```

See Also

Fraction.add(), Fraction.div(), Fraction.mul()

Fraction.toExponential()

Format a fraction number using exponential notation

Synopsis

Fraction.toExponential(digits)

Arguments

Digits The number of digits that will appear after the decimal point. This may be a value between 0 and 20, inclusive. If this argument is omitted, as many digits as necessary will be used. A Fraction number is always formatted as:

(*numerator : denominatort*)

Returns

A string representations of the Fraction number, in exponential notation, with one digit before the decimal place and *digits* digits after the decimal place. The fractional part of the Fraction number is rounded or padded with zeros, as necessary, so that is has the specified length.

Example

```
var z = new Fraction( 123456789, 123456789 );
z.toExponential(1);          // result (1.2e+4:1.2e+4)
z.toExponential(5);          // result (1.23457e+4:1.23457e+4)
z.toExponential(10);         // result (1.23456789000e+4:1.23456789000e+4)
z.toExponential();           // result (1.23456789e+4:1.23456789e+4)
```

See Also

Fraction.toFixed(), Fraction.toPrecision(), Fraction.toString()

Fraction.toFixed()

Format a number using fixed-point notation

Synopsis

Fraction.toFixed(digits)

Arguments

Digits The number of digits that will appear after the decimal point. This may be a value between 0 and 20, inclusive. If this argument is omitted, it is treated as zero. A Fraction number is always formatted as:
(*numerator : denominator*)

Returns

A string representations of the Fraction number, that does not used exponential notation and has exactly *digits* digits after the decimal point. The *Fraction number* is rounded as necessary, and the fraction part is padded with zeros if necessary so that it has the specified length. If the *Fraction number* is greater than 1e+21, this method simple calls *number*.toString() and return a string in exponential notation.

Example

```
var z = new Fraction( 123456789, 123456789 );
z.toFixed(5);          // result (12345.7:12345.7)
z.toFixed(6);          // result (12345.678900:12345.678900)
z.toFixed();           // result (12346:1234.6)
```

See Also

Fraction.toExponential(), Fraction.toPrecision(), Fraction.toString()

Fraction.toPrecision()

Format the significant digits of a Fraction number

Synopsis

Fraction.toPrecision(digits)

Arguments

Digits The number of significant digits to appear in the returned string. This may be a value between 1 and 21, inclusive. If this argument is omitted, the toString() method is used instead tyo convert the Fraction number to a base-10 value. A Fraction number is always formatted as:
(*numerator : denominator*)

Returns

A string representations of the *Fraction number*, that contains *precisions* significant digits. If *precision* is large enough to include all the digits of the integer part of number, the returned string uses fixed-point notation. Otherwise exponential notation is used with one digit before the decimal place and *precision* – 1 digits after the decimal place. The number is rounded or padded with zeros as necessary.

Example

```
var z = new Fraction( 123456789, 123456789 );
z.toPrecision(1);          // result (1e+4:1e+4)
z.toPrecision(3);          // result (1.23e+4:1.2e+4)
z.toPrecision(5);          // result (12346:12346)
```

See Also

Fraction.toExponential(), Fraction.toFixed(), Fraction.toString()

Fraction.toString()

Format the significant digits of a Fraction number

Synopsis

Fraction.toString(radix)

Arguments

Radix If omitted, base 10 will be used to convert the Fraction number to a string. Otherwise the radix will be used (2..36). A Fraction number is always formatted as:
(*numerator : denominator*)

Returns

A string representation of the *Fraction number*, in the indicated radix.
Example

```
var z = new Fraction( 123456789, 123456789 );
z.toString();          // result (12346789:12346789)
```

See Also

Fraction.toExponential(), Fraction.toFixed(), Fraction.toPrecision()

Fraction.typeof()

Return the type of the Fraction as either “number”or “bigint”.

Synopsis

Fraction.typeof()

Returns

A string representation of the type of the *Fraction number*. The type returned can be “number” or “bigint”.

Example

```
var z = new Fraction( 12,1 );
z.typeof();                 // result “number”
var y=new Fraction(12n:4n);
y.typeof()                  // result “bigint”
```

See Also

Fraction.valueof()

Return the primitive number value

Synopsis

Fraction object.valueof()

Returns

The primitive value of the *Fraction number* is returned, which is the same as *Fraction number*.numerator().

Example

```
var z = new Fraction(3,4);
z.valueOf()                 // return 3
```

See Also

Fraction.whole()

Return the whole integer of the fraction number.

Synopsis

Fraction.whole()

Returns

An integer number which value is the integer value of the *Fraction number*. The return number can be either “number” or “bigint” number depending on which type of the fraction is. This is the same of taking Math.trunc(numerator/denominator)

Example

```
var z = new Fraction( 12,5 );
z.whole();                  // result  2 since integer of 12/5 is 2.
var y=new Fraction(12n:5n);
y.whole()                   // result 2n (bigint)
```

See Also

Fraction.typeof(), Faction,.reduce()

Fraction.zero

Return Fraction zero

Synopsis

Fraction.zero

Returns

The Fraction constant zero (0:1). Notice that the denominator is set to one not zero.

Example

```
var z = Fraction.zero;   // z=(0:1)
```

See Also

Hve@hvks.com

Fraction.one

parseFraction()

Convert a string to a Fraction number

Synopsis

parseFraction(s)

Arguments

s The string to be parsed and converted to a *Fraction number.*

Returns

parseFraction() parses and return a new Fraction number contained in s. parseFraction() return a Fraction NaN number if parsing fails. A Fraction number can either be in the format:

(*numerator : denominator*)

Where either the *numerator* or the *denominator* can be missing but not both at the same time. parseFraction() can also parsed string omitting the leading and trailing parentheses. Notice that any negative denominator sign is moved to the sign of the numerator.

Example

```
var z = parseFraction( “(12:-34)”);     // result (-12:34)
z = parseFraction( “(12)” );            // result (12:1)
z = parseFraction( “(:-12)” );          // result (0:12)
z=parseFraction(“(-1:-2)”);             // result (1:2)
```

See Also

Example 1

Square root of x can be expressed by a continuous fraction:

$$\sqrt{x} = 1 + \cfrac{x-1}{2+\cfrac{x-1}{2+\cfrac{x-1}{2+\ddots}}}$$

```
// Example of using continuous fraction for finding the square root of a number.
function sqrt(x,i)
        {var cf=Fraction(x-1,2);
        for(;i>0;--i)      // i is the number of iterations
                {
                cf=Fraction.add(cf,Fraction(2));
                cf=Fraction.div(Fraction(x-1,1),cf);
                }
        return Fraction.add(cf,Fraction(1));
        }

var cf=sqrt(3,10);
```

Example 2

π can be expressed by a continuous fraction:

$$\pi = \cfrac{4}{1+\cfrac{1^2}{3+\cfrac{2^2}{5+\cfrac{3^2}{7+\ddots}}}}$$

```
// Example of using continuous fraction for finding the π.
function pi(i)  // i is the number of iterations
        {
        var cf=Fraction(0);
        for(;i>0;--i)
                {
                cf=Fraction.add(cf,Fraction(i*2+1));
                cf=Fraction.div(Fraction(i*i),cf);
                }
        return Fraction.div(Fraction(4),Fraction.add(cf,Fraction(1)));
        }
var cf=pit(10);
```

Hve@hvks.com

```
var f=cf.numerator()/cf.denominator();
alert("sqrt of "+(3).toString()+"="+cf.toString()+"="+f+" Error="+(Math.sqrt(3)-
f).toExponential(2));
```