

JavaScript ComplexBF Library

(Complex BigFloat)

Version 2.01

By Henrik Vestermark (hve@hvks.com)

Contents

Design note	4
Use ComplexBF when	5
You probably do not need ComplexBF when.....	6
Practical rule of thumb.....	6
ComplexBF	7
Constructor.....	7
Method	8
Functions.....	8
Constants.....	9
Miscellaneous	9
API	9
ComplexBF.abs().....	10
ComplexBF.arg()	10
ComplexBF.add().....	10
ComplexBF.acos()	11
ComplexBF.acosh()	12
ComplexBF.asin().....	12
ComplexBF.asinh().....	13
ComplexBF.atan().....	13
ComplexBF.atanh().....	14
ComplexBF.beta().....	14
ComplexBF.cos()	15
ComplexBF.cosh()	15
ComplexBF.conj()	16
ComplexBF.div()	16
ComplexBF.equal().....	17
ComplexBF.exp().....	19
ComplexBF.tgamma()	19
ComplexBF.i.....	20
ComplexBF.imag()	20
ComplexBF.lgamma()	21
ComplexBF.log()	21
ComplexBF.log10()	22

ComplexBF.mul()	22
ComplexBF.negate()	23
ComplexBF.notequal()	24
ComplexBF.norm()	24
ComplexBF.one	25
ComplexBF.polar()	25
ComplexBF.pow()	26
ComplexBF.proj(z)	26
ComplexBF.real()	27
ComplexBF.sin()	28
ComplexBF.sinh()	28
ComplexBF.sub()	29
ComplexBF.sqrt()	29
ComplexBF.tan()	30
ComplexBF.tanh()	30
ComplexBF.toExponential()	31
ComplexBF.toFixed()	32
ComplexBF.toPrecision()	32
ComplexBF.toString()	33
ComplexBF.valueOf()	34
ComplexBF.zero	34
parseComplexBF(s [, precision [, rounding]])	35

Design note

ComplexBF is designed as a thin, predictable layer on top of BigFloat. Complex arithmetic is only as good as the underlying real arithmetic, so the library treats precision and rounding as first-class numeric control parameters.

Two practical goals drive the constructor design:

1. Local control when needed
When you construct a value from literals or strings, you often want to force a specific precision and rounding behavior for that conversion. The options object gives a direct way to do that without relying on global defaults.
2. Natural propagation in composed computations
When ComplexBF values are built from existing BigFloat values, the most useful default is to inherit their numeric context. This keeps precision and rounding stable across intermediate results, and avoids surprising “downshifts” to global defaults.

Constructor precision and rounding rules.

Call form	Precision used	Rounding used	Notes
<code>new ComplexBF()</code>	<code>BigFloat.defaultPrecision</code>	<code>BigFloat.defaultRounding</code>	Returns <code>ComplexBF.zero</code>
<code>new ComplexBF(real)</code>	From options if given, otherwise from <code>real</code> if it is a <code>BigFloat</code> , otherwise default	Same rule	Imaginary part is set to zero
<code>new ComplexBF(real, imag)</code>	From options if given, otherwise inferred from <code>BigFloat</code> arguments, otherwise default	Same rule	Both parts are converted to <code>BigFloat</code>
<code>new ComplexBF(real, imag, { precision, rounding })</code>	<code>precision</code>	<code>rounding</code>	Explicit override
<code>ComplexBF(value)</code>	From options if given, otherwise from <code>value</code> if it is a <code>BigFloat</code> , otherwise default	Same rule	Conversion mode, no <code>new</code>
<code>ComplexBF(z)</code> where <code>z</code> is <code>ComplexBF</code>	Unchanged	Unchanged	Returns <code>z</code> unchanged
<code>ComplexBF(value, { precision, rounding })</code>	<code>precision</code>	<code>rounding</code>	Conversion with explicit

			numeric control
--	--	--	--------------------

Numeric model

ComplexBF is built directly on top of BigFloat and inherits its numeric control model. All complex values are represented internally as pairs of BigFloat values, one for the real part and one for the imaginary part. Precision and rounding are therefore an integral part of every ComplexBF value and are either explicitly specified at construction time or inherited from the underlying BigFloat operands.

The library is designed to preserve numeric context during computations. When complex values are constructed from existing BigFloat values, their precision and rounding mode are propagated unless explicitly overridden. This makes ComplexBF suitable for multi-step, arbitrary-precision calculations that require consistency and predictability.

Scope and intent

ComplexBF focuses on correctness, reproducibility, and explicit numeric control. It is intended for scientific, mathematical, and educational use where standard double-precision complex arithmetic is insufficient. Some advanced special functions are included for completeness, but may not yet be implemented in the current version.

Use ComplexBF when you need complex arithmetic that is more accurate, more controllable, or wider in range than JavaScript `Number` can provide.

Use ComplexBF when

- Double precision is not enough
 - You need more than about 15 to 16 decimal digits of reliable accuracy.
 - Small errors accumulate across many operations, for example, long recurrences, continued fractions, FFT style workflows, or iterative solvers.
- You must control precision and rounding
 - You want deterministic results across platforms and runs.
 - You need to match a target accuracy or validate algorithms using higher precision as a reference.
- You work with very large or very small magnitudes
 - Values underflow or overflow with `Number`, or you need stable scaling over huge ranges.
- You implement or test complex functions in arbitrary precision
 - `exp`, `log`, `pow`, `trig`, and hyperbolic functions, and related identities where branch behavior and loss of significance matter.
- You need stable comparisons and tolerances
 - You want to compare results at a chosen precision rather than relying on machine epsilon.

You probably do not need ComplexBF when

- You are doing everyday engineering math where `Number` precision is fine.
- Performance is the main concern, and you do not need more than double precision.
- Inputs are noisy enough that extra precision does not improve the real result.

Practical rule of thumb

If your expected relative error budget is tighter than about $1e-12$, or your computation is long enough that round-off becomes visible, switch to ComplexBF and choose a working precision with a safety margin (often 2 to 4 times the digits you ultimately need).

ComplexBF

Support for ComplexBF arbitrary precision number arithmetic in JavaScript. Require BigFloat.js as the basis of the ComplexBF packages. BigFloat.js provides the underlying arbitrary-precision arithmetic that forms the basis of the ComplexBF packages. To get acquainted with the library, please read the BigFloat JavaScript package document.

Constructor

```
new ComplexBF(real [, imaginary [, options ]])    // Constructor
ComplexBF(value [, options ])                   // Conversion
```

Arguments

real The real part of a ComplexBF number
imaginary The imaginary part of a ComplexBF number. Optional
 If the imaginary argument is omitted, it is treated as zero
Options Optional. An Object specifying numeric control parameters:

<i>precision</i>	The Precision in decimal digits used for the internal BigFloat value
<i>rounding</i>	The rounding mode used during conversion

If there are no arguments, it is treated as a ComplexBF zero
 If *ComplexBF* is invoked as a conversion, the value parameter is converted to a *ComplexBF number* and returned. The value can be a ComplexBF, BigFloat, number or string.

Remarks

If no arguments are provided, the result is ComplexBF.zero.
 If ComplexBF is invoked without new, it acts as a conversion.
 If value is already a ComplexBF, it is returned unchanged.
 If real or imaginary are BigFloat values, their precision and rounding are propagated to the resulting ComplexBF unless explicitly overridden by options.
 If precision or rounding are not specified, the defaults BigFloat.defaultPrecision and BigFloat.defaultRounding are used (see the BigFloat user manual for details)

The resulting object stores its effective precision and rounding mode.

Returns

Returns a ComplexBF object initialized with the specified real and imaginary values, using the resolved precision and rounding mode.

Example

```
const z1 = new ComplexBF(1.5, -2.0);
const z2 = ComplexBF("3.25");
const z3 = new ComplexBF(1, 2, { precision: 256 });
const z4 = ComplexBF(BigFloat("1.1", 128), BigFloat("2.2", 128));
```

Method

abs()	Return the magnitude of a ComplexBF number
arg()	Return the angle of the polar representation of the ComplexBF number.
conj()	Return a new Conjugated ComplexBF object.
imag()	Return the Imaginary part of the ComplexBF number.
negate()	Return a new Negated ComplexBF Object.
norm()	Return the norm of the ComplexBF number.
real()	Return the real part of the ComplexBF number.
toExponential()	Converts a ComplexBF number to a string using exponential notation with the specified number of digits after the Decimal place.
toFixed()	Converts a ComplexBF number to a string that contains a specified number of digits after the decimal place.
toPrecision()	Convert a ComplexBF 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 ComplexBF number to a string using a specified radix(base)
valueOf()	The primitive real number value of this ComplexBF number object.

Functions

abs()	Return the magnitude of a ComplexBF number
add()	Return the addition of two ComplexBF numbers
acos()	Return arc cosine of the ComplexBF number
acosh()	Return the arc cosine hyperbolic of the ComplexBF number
asin()	Return the arcsine of the ComplexBF number
asinh()	Return the arc sine hyperbolic of the ComplexBF number
atan()	Return the arc tangent of the ComplexBF number
atanh()	Return the arctangent hyperbolic of the ComplexBF number
beta()	Return the beta function of a ComplexBF number
cos()	Return cosine of the ComplexBF number
cosh()	Return the cosine hyperbolic of the ComplexBF number
div()	Return the division of two ComplexBF numbers.
equal()	Return the Boolean value (true, false) of the equality of two ComplexBF numbers.
exp()	Return the ComplexBF power of e.
lgamma()	Return the logarithm of the gamma function of a ComplexBF number

<code>log()</code>	Return the ComplexBF natural logarithm.
<code>log10()</code>	Return the ComplexBF base 10 logarithm.
<code>mul()</code>	Return the product of two ComplexBF numbers.
<code>polar()</code>	Return the ComplexBF number of the polar representation.
<code>pow()</code>	Return the ComplexBF power of x^y .
<code>proj()</code>	Return the Riemann projections
<code>sin()</code>	Return the sine of the ComplexBF number
<code>sinh()</code>	Return the sine hyperbolic of the ComplexBF number
<code>sub()</code>	Return the difference between two ComplexBF numbers.
<code>sqrt()</code>	Return the ComplexBF square root.
<code>tan()</code>	Return the tangent of the ComplexBF number
<code>tanh()</code>	Return the tangent hyperbolic of the ComplexBF number
<code>tgamma()</code>	Return the gamma function of a ComplexBF number

Constants

<code>ComplexBF.zero</code>	Return a new ComplexBF(0,0) object
<code>ComplexBF.one</code>	Return a new ComplexBF(1,0) object
<code>ComplexBF.i</code>	Return a new ComplexBF(0,1) object

Miscellaneous

<code>parseComplexBF()</code>	Parse a ComplexBF float number string
-------------------------------	---------------------------------------

API

ComplexBF.abs()

Return the absolute magnitude of the ComplexBF number

Synopsis

ComplexBF object.abs()

Returns

The magnitude of the ComplexBF number is returned. Special calculation are made to prevent intermediate result to overflow.

Example

```
var z = new ComplexBF( 3, 4 );  
z.abs()           // result 5
```

See Also

ComplexBF.norm()

ComplexBF.arg()

Return the angle of the polar representation of the ComplexBF number

Synopsis

ComplexBF object.arg()

Returns

Return the angle in radians of the polar representation of the ComplexBF number

Example

```
var z = new ComplexBF( 3, 4 );  
z.arg()           // result 0.927295...
```

See Also

ComplexBF.add()

Add two ComplexBF numbers

Synopsis

```
ComplexBF.add(a,b)
```

Arguments

a,b The ComplexBF numbers to be added.

Returns

The result of the ComplexBF addition.

Example

```
var z = new ComplexBF( 3, 4 );  
var y=new ComplexBF(1,2);  
  
ComplexBF.add(z,y)            // result (4+i6)
```

See Also

ComplexBF.div(), ComplexBF.mul(), ComplexBF.sub()

ComplexBF.acos()

Return the arc cosine of the ComplexBF number

Synopsis

```
ComplexBF.acos(a)
```

Returns

Return the arc cosine of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.acos(z)            // result (0.9368124611557193-i2.3055090312434685)
```

See Also

ComplexBF.asin(), ComplexBF.atan()

ComplexBF.acosh()

Return the arc cosine hyperbolic of the ComplexBF number

Synopsis

```
ComplexBF.acosh(a)
```

Returns

Return the arc cosine hyperbolic of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.acosh(z) // result (2.305509031243477+i0.9368124611557199)
```

See Also

ComplexBF.asinh(), ComplexBF.atanh()

ComplexBF.asin()

Return the arcsine of the ComplexBF number

Synopsis

```
ComplexBF.asin(a)
```

Returns

Return the arc sine of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.asin(z) // result (0.6339838656391773+i2.3055090312434685)
```

See Also

ComplexBF.acos(), ComplexBF.atan()

ComplexBF.asinh()

Return the arc sine hyperbolic of the ComplexBF number

Synopsis

ComplexBF.asinh(a)

Returns

Return the arc sine hyperbolic of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.asinh(z) // result (2.2999140408792695+i0.9176168533514787)
```

See Also

ComplexBF.acosh(), ComplexBF.atanh()

ComplexBF.atan()

Return the arctangent of the ComplexBF number

Synopsis

ComplexBF.atan(a)

Returns

Return the arc tangent of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.atan(z) // result (1.4483069952314644+i0.15899719167999904)
```

See Also

ComplexBF.acos(), ComplexBF.asin()

ComplexBF.atanh()

Return the arc tangent hyperbolic of the ComplexBF number

Synopsis

ComplexBF.atanh(a)

Returns

Return the arc tanh hyperbolic of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.atanh(z) // result (0.11750090731143381+i1.4099210495965755)
```

See Also

ComplexBF.acosh(), ComplexBF.asinh()

ComplexBF.beta()

Return the beta function of the ComplexBF numbers

Synopsis

ComplexBF.beta(x,y)

Returns

Return the beta function of the ComplexBF number a and b.

Example

```
var x=new ComplexBF( 3, 4 ), y=ComplexBF(2,-3);  
ComplexBF.beta(x,y) // result (0.000594954...,0.000791355...)
```

See Also

ComplexBF.gamma(), ComplexBF.lgamma()

ComplexBF.cos()

Return the cosine of the ComplexBF number

Synopsis

ComplexBF.cos(a)

Returns

Return the cosine of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.cos(z)           // result (-27.034945603074224-i3.851153334811777)
```

See Also

ComplexBF.sin(), ComplexBF.tan()

ComplexBF.cosh()

Return the cosine hyperbolic of the ComplexBF number

Synopsis

ComplexBF.cosh(a)

Returns

Return the cosine hyperbolic of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.cosh(z)          // result (-6.580663040551157-i7.581552742746545)
```

See Also

ComplexBF.sinh(), ComplexBF.tanh()

ComplexBF.conj()

Return the conjugated form of the ComplexBF number

Synopsis

ComplexBF object.conj()

Returns

Return the ComplexBF conjugated form.

Example

```
var z = new ComplexBF( 3, 4 );  
z.conj()           // result (3-i4)
```

See Also

ComplexBF.negate()

ComplexBF.div()

Divide two ComplexBF numbers

Synopsis

ComplexBF.div(a,b)

Arguments

a,b The ComplexBF numbers to be divided. Special calculation are made to prevent intermediate result to overflow.

Returns

The result of the ComplexBF division a/b.

Example

```
var z = new ComplexBF( 3, 4 );  
var y=new ComplexBF(1,2);  
  
ComplexBF.div(z,y)           // result (2.2+i0.4)
```

See Also

ComplexBF.add(), ComplexBF.mul(), ComplexBF.sub()

ComplexBF.equal()

Compare two ComplexBF numbers for equality

Synopsis

ComplexBF.equal(a,b)

Arguments

a,b The ComplexBF numbers to be compare for

Returns

The Boolean value of the equal comparison.

Example

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

if( ComplexBF.equal(z,y))...      // result false
if( ComplexBF.equal(z,z))...      // result true
if( !ComplexBF.equal(z,y))...     // result true ! to do "notequal" comparison
```

See Also

Complex.notequal()

ComplexBF.erf()

Return the error function of a real or complex number.

Synopsis

ComplexBF.erf(z)

Arguments

z A complex or real value.

Returns

Returns $\operatorname{erf}(z)$. For purely real inputs, the result is typically a purely real Complex number (value + i0).

Example

```
ComplexBF.erf(0.5)           // (0.5204998778 + i0)
ComplexBF.erf(new Complex(1, 1)) // (... + i ...)
```

Notes

If the JavaScript runtime does not provide `Math.erf`, the library defines `Math.erf(x)` as a wrapper that returns the real part of `ComplexBF.erf(x)`.

See Also

`ComplexBF.erfc()`

`ComplexBF.erfc()`

Return the complementary error function of a real or complex number.

Synopsis

`ComplexBF.erfc(z)`

Arguments

z A complex or real value.

Returns

Returns $\operatorname{erfc}(z) = 1 - \operatorname{erf}(z)$.

Example

```
ComplexBF.erfc(0.5)           // (0.4795001222 + i0)
ComplexBF.erfc(new Complex(1, 1)) // (... + i ...)
```

Notes

If the JavaScript runtime does not provide `Math.erfc`, the library defines `Math.erfc(x)` as a wrapper that returns the real part of `ComplexBF.erfc(x)`.

See Also

ComplexBF.erf()

ComplexBF.exp()

Compute e^x

Synopsis

ComplexBF.exp(x)

Arguments

x A ComplexBF numbers to be used as the exponent

Returns

e^x , e raised to the power of the specified exponent x , where e is the base of the natural logarithm, with a value of approximately 2.71828.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.exp(z)...        // ez approximately (-13.128+i5.200)
```

See Also

ComplexBF.log(), ComplexBF.log10(), ComplexBF.pow()

ComplexBF.tgamma()

Return the gamma function of the ComplexBF number. Not implemented yet

Synopsis

ComplexBF.gamma(x)

Returns

Return the gamma function of the ComplexBF number x .

Example

```
var x=new ComplexBF( 3, 4 );  
ComplexBF.tgamma(x)           // result (0.00522553...,0.1725470...)
```

See Also

ComplexBF.lgamma(), ComplexBF.beta()

ComplexBF.i

Return ComplexBF i

Synopsis

ComplexBF.i

Returns

The ComplexBF constant i (0+i1).

Example

```
var z = ComplexBF.i;           // z=(0+i1)
```

See Also

ComplexBF.one, ComplexBF.zero

ComplexBF.imag()

Return the imaginary part of the ComplexBF number

Synopsis

ComplexBF object.imag()

Returns

Return the imaginary part of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
z.imag() // result 4
```

See Also

ComplexBF.real()

ComplexBF.lgamma()

Return the logarithm gamma function of the ComplexBF number. Not implemented yet.

Synopsis

ComplexBF.lgamma(x)

Returns

Return the logarithm gamma function of the ComplexBF number x.

Example

```
var x=new ComplexBF( 3, 4 );  
ComplexBF.lgamma(x) // result (-1.756626...,4.742664...)
```

See Also

ComplexBF.tgamma(), ComplexBF.beta()

ComplexBF.log()

Compute the natural logarithm of x

Synopsis

ComplexBF.log(x)

Arguments

x A ComplexBF numbers not equal to zero

Returns

Return log(x)

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.log(z)...      // log(x) approximately (1.609+i0.927)
```

See Also

ComplexBF.exp(), ComplexBF.log10(),

ComplexBF.log10()

Compute the base-10 logarithm of x

Synopsis

ComplexBF.log10(x)

Arguments

x A ComplexBF numbers not equal to zero

Returns

Return log10(x)

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.log10(z)...      // log10(3+i4) approximately (0.699+i0.403)
```

See Also

ComplexBF.exp(), ComplexBF.log(),

ComplexBF.mul()

Multiply two ComplexBF numbers

Synopsis

ComplexBF.mul(a,b)

Arguments

a,b The ComplexBF numbers to be multiplied.

Returns

The result of the ComplexBF multiplication.

Example

```
var z = new ComplexBF( 3, 4 );  
var y=new ComplexBF(1,2);  
  
ComplexBF.mul(z,y)            // result (-5+i10)
```

See Also

ComplexBF.add(), ComplexBF.div(), ComplexBF.sub()

ComplexBF.negate()

Return the negated ComplexBF number

Synopsis

ComplexBF object.negate()

Returns

Return the negated ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
z.negate()                    // result (-3-i4)
```

See Also

ComplexBF.conj()

ComplexBF.notequal()

Compare two ComplexBF numbers for non-equality

Synopsis

ComplexBF.notequal(a,b)

Arguments

a,b The ComplexBF numbers to be compare for non-equality

Returns

The Boolean value of the not-equal comparison.

Example

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

if( ComplexBF.notequal(z,y)...      // result true
if( ComplexBF.notequal(z,z)...      // result false
if( !ComplexBF.notequal(z,y)...     // result true
```

See Also

ComplexBF.equal()

ComplexBF.norm()

Return the norm (square magnitude) of the ComplexBF number

Synopsis

ComplexBF object.norm()

Returns

The norm (squared magnitude) of the ComplexBF number is returned.

Example

```
var z = new ComplexBF( 3, 4 );
z.norm()            // result 25
```

See Also

ComplexBF.abs()

ComplexBF.one

Return ComplexBF one

Synopsis

ComplexBF.one

Returns

The ComplexBF constant one (1+i0).

Example

```
var z = ComplexBF.one;           // z=(1+i0)
```

See Also

ComplexBF.zero, ComplexBF.i

ComplexBF.polar()

Convert polar coordinates into a ComplexBF number

Synopsis

ComplexBF.polar(mag,arg)

Arguments

<i>mag</i>	Magnitude of ComplexBF number
<i>arg</i>	Angle of ComplexBF number.

Returns

Return the ComplexBF number.

Example

```
var z = ComplexBF.polar( 4, 0.5 );
```

```
ComplexBF.polar(4,0.5)           // result (3.510+i1.918)
```

See Also

ComplexBF.pow()

Compute x^y

Synopsis

ComplexBF.pow(x,y)

Arguments

x A ComplexBF numbers to be raised to a power
y A ComplexBF power that x is raised to

Returns

X to the power of y. x^y

Example

```
var x = new ComplexBF( 3, 4 );  
var y = new ComplexBF( 1 , 2 )
```

```
ComplexBF.pow(x,y)...           //  $x^y$  approximately (-0.4198+i0.6605)
```

See Also

ComplexBF.exp()

ComplexBF.proj(z)

Description

Computes the projection of a complex number onto the Riemann sphere.

The projection is defined as:

- If *z* is finite, the result is *z*

- If z has an infinite real or imaginary part, the result is the complex infinity

This function is primarily provided for completeness and compatibility with complex analysis conventions, where the extended complex plane (Riemann sphere) is used.

Synopsis

ComplexBF.proj(z)

Arguments

- Z A ComplexBF value to be projected.

Returns

- A ComplexBF value.
If z is finite, the return value equals z .
If z contains an infinite component, the return value represents complex infinity.

Remarks

- This function does not modify precision or rounding mode.
- The operation is exact for finite values.
- Useful when implementing functions that formally operate on the extended complex plane.

Example

```
const z1 = ComplexBF(1.5, -2.0);  
const p1 = ComplexBF.proj(z1); // returns 1.5 - 2.0i  
  
const z2 = ComplexBF.infinity;  
const p2 = ComplexBF.proj(z2); // returns complex infinity
```

ComplexBF.real()

Return the real part of the ComplexBF number

Synopsis

ComplexBF object.real()

Returns

Return the real part of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
z.real()           // result 3
```

See Also

ComplexBF.imag()

ComplexBF.sin()

Return the sine of the ComplexBF number

Synopsis

ComplexBF.sin(a)

Returns

Return the sine of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.sin(z)       // result (3.853738037919377-i27.016813258003932)
```

See Also

ComplexBF.cos(), ComplexBF.tan()

ComplexBF.sinh()

Return the sine hyperbolic of the ComplexBF number

Synopsis

ComplexBF.sinh(a)

Returns

Return the sine hyperbolic of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.sinh(z)           // result (-6.580663040551157-i7.581552742746545)
```

See Also

ComplexBF.cosh(), ComplexBF.tanh()

ComplexBF.sub()

Subtract two ComplexBF numbers

Synopsis

ComplexBF.sub(a,b)

Arguments

a,b The ComplexBF numbers to be subtracted.

Returns

The result of the ComplexBF subtraction.

Example

```
var z = new ComplexBF( 3, 4 );  
var y=new ComplexBF(1,2);  
  
ComplexBF.sub(z,y)           // result (2+i2)
```

See Also

ComplexBF.add(), ComplexBF.div(), ComplexBF.mul()

ComplexBF.sqrt()

Compute a ComplexBF square root

Synopsis

ComplexBF.sqrt(x)

Arguments

x A ComplexBF numbers to be square rooted. Special calculation are made to prevent intermediate result to overflow.

Returns

The square root of x .

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.sqrt(z)... // Result (2+i1)
```

See Also

ComplexBF.tan()

Return the tangent of the ComplexBF number

Synopsis

ComplexBF.tan(a)

Returns

Return the tangent of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.tan(z) // result (-0.00018734620462947842+i0.9993559873814732)
```

See Also

ComplexBF.cos(), ComplexBF.sin()

ComplexBF.tanh()

Return the tangent hyperbolic of the ComplexBF number

Synopsis

ComplexBF.tanh(a)

Returns

Return the tanh hyperbolic of the ComplexBF number.

Example

```
var z = new ComplexBF( 3, 4 );  
ComplexBF.tanh(z) // result (-6.580663040551157-i7.581552742746545)
```

See Also

ComplexBF.cosh(), ComplexBF.sinh()

ComplexBF.toExponential()

Format a number using exponential notation

Synopsis

ComplexBF.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 ComplexBF number is always formatted as:

(real_part ±i imaginary_part)

Returns

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

Example

```
var z = new ComplexBF( 12345.6789, 12345.6789 );  
z.toExponential(1); // result (1.2e+4+i1.2e+4)  
z.toExponential(5); // result (1.23457e+4+i1.23457e+4)
```

```
z.toExponential(10);    // result (1.23456789000e+4+i1.23456789000e+4)
z.toExponential();     // result (1.23456789e+4+i1.23456789e+4)
```

See Also

ComplexBF.toFixed(), ComplexBF.toPrecision(), ComplexBF.toString()

ComplexBF.toFixed()

Format a number using fixed-point notation

Synopsis

```
ComplexBF.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 ComplexBF number is always formatted as:
(real_part ±i imaginary_part)

Returns

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

Example

```
var z = new ComplexBF( 12345.6789, 12345.6789 );
z.toFixed(5);          // result (12345.7+i12345.7)
z.toFixed(6);          // result (12345.678900+i12345.678900)
z.toFixed();           // result (12346+i1234.6)
```

See Also

ComplexBF.toExponential(), ComplexBF.toPrecision(), ComplexBF.toString()

ComplexBF.toPrecision()

Format the significant digits of a ComplexBF number

Synopsis*ComplexBF*.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 to convert the ComplexBF number to a base-10 value. A ComplexBF number is always formatted as:
 (*real_part* ±i *imaginary_part*)

Returns

A string representations of the *ComplexBF* 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 ComplexBF( 12345.6789, 12345.6789 );
z.toPrecision(1);            // result (1e+4+i1e+4)
z.toPrecision(3);            // result (1.23e+4+i1.2e+4)
z.toPrecision(5);            // result (12346+i12346)
```

See Also

ComplexBF.toExponential(), ComplexBF.toFixed(), ComplexBF.toString()

ComplexBF.toString()

Format the significant digits of a ComplexBF number

Synopsis*ComplexBF*.toString(radix)**Arguments**

Radix If omitted the base 10 will be used to convert the ComplexBF number to a string. Otherwise the radix will be used (2..36). A ComplexBF number is always formatted as:
 (*real_part* ±i *imaginary_part*)

Returns

A string representations of the *ComplexBF number*, in the indicated radix.

Example

```
var z = new ComplexBF( 12345.6789, 12345.6789 );
z.toString();           // result (1234.6789+i1234.6789)
```

See Also

ComplexBF.toExponential(), ComplexBF.toFixed(), ComplexBF.toPrecision()

ComplexBF.valueOf()

Return the primitive number value

Synopsis

ComplexBF object.valueOf()

Returns

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

Example

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

See Also

ComplexBF.zero

Return ComplexBF zero

Synopsis

ComplexBF.zero

Returns

The ComplexBF constant zero (0+i0).

Example

```
var z = ComplexBF.zero;
```

See Also

ComplexBF.one, ComplexBF.i

```
parseComplexBF(s [, precision [, rounding ]]))
```

Parses a textual representation of a complex number and returns a `ComplexBF` value.

The function accepts real numbers, imaginary numbers, and full complex expressions using `i` as the imaginary unit. If `precision` and `rounding` are not explicitly provided, the default `BigFloat` precision and rounding mode are used.

Synopsis

```
parseComplexBF(s)  
parseComplexBF(s, precision)  
parseComplexBF(s, precision, rounding)
```

Arguments

- | | |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>S</i> | A string containing the textual representation of a complex number. |
| <i>precision</i> | <i>Optional.</i> The precision (in bits) used for the internal <code>BigFloat</code> values created during parsing. If omitted, <code>BigFloat.defaultPrecision</code> is used. |
| <i>Rounding</i> | <i>Optional.</i> The rounding mode is applied when converting parsed decimal values to <code>BigFloat</code> . If omitted, <code>BigFloat.defaultRounding</code> is used. |

Returns

`parseComplexBF()` parses and return a new `ComplexBF` number contained in `s`.
`parseComplexBF()` return a `ComplexBF NaN` number if parsing fails.

A `ComplexBF` number can either be in the format:
 $(real_part \pm i\ imaginary_part)$

Where either the *real_part* or the *imaginary_part* can be missing but not both at the same time. `parseComplexBF()` can also parse a string, omitting the leading and trailing

parentheses.

Accepted formats

Real numbers:

"3.14"

"-2"

Pure imaginary numbers

"i"

"-i"

"2.5i"

Full complex numbers

"1+2i"

"3.5-4.25i"

"-1.2+i"

Remarks

- Whitespace in the input string is ignored.
- The imaginary unit must be written as *i*.
- The sign of the imaginary part must appear *before* *i*.
Forms such as "1+i" are valid, while "1+i+" or "1+ i" are not.
- Parsing errors result in an exception.
- This function is intended for user input and textual interchange, not for high-performance numeric construction.

Example

```
const z1 = parseComplexBF("1.25-3.75i");  
const z2 = parseComplexBF("2+i", 128);  
const z3 = parseComplexBF("-0.1+0.2i", 256, BigFloat.ROUND_HALF_EVEN);
```

See Also