import {List} from 'immutable';

import {Value} from './index';

/\*\*

\* Sass's [number type](https://sass-lang.com/documentation/values/numbers).

\*

\* @category Custom Function

\*/

export class SassNumber extends Value {

/\*\*

\* Creates a new number with more complex units than just a single numerator.

\*

\* Upon construction, any compatible numerator and denominator units are

\* simplified away according to the conversion factor between them.

\*

\* @param value - The number's numeric value.

\*

\* @param unit - If this is a string, it's used as the single numerator unit

\* for the number.

\*

\* @param unit.numeratorUnits - If passed, these are the numerator units to

\* use for the number. This may be either a plain JavaScript array or an

\* immutable [[List]] from the [`immutable`

\* package](https://immutable-js.com/).

\*

\* @param unit.denominatorUnits - If passed, these are the denominator units

\* to use for the number. This may be either a plain JavaScript array or an

\* immutable [[List]] from the [`immutable`

\* package](https://immutable-js.com/).

\*/

constructor(

value: number,

unit?:

| string

| {

numeratorUnits?: string[] | List<string>;

denominatorUnits?: string[] | List<string>;

}

);

/\*\* This number's numeric value. \*/

get value(): number;

/\*\* Whether [[value]] is an integer according to Sass's equality logic. \*/

get isInt(): boolean;

/\*\*

\* If [[value]] is an integer according to [[isInt]], returns [[value]]

\* rounded to that integer. If it's not an integer, returns `null`.

\*/

get asInt(): number | null;

/\*\*

\* This number's numerator units as an immutable [[List]] from the

\* [`immutable` package](https://immutable-js.com/).

\*/

get numeratorUnits(): List<string>;

/\*\*

\* This number's denominator units as an immutable [[List]] from the

\* [`immutable` package](https://immutable-js.com/).

\*/

get denominatorUnits(): List<string>;

/\*\* Whether this number has any numerator or denominator units. \*/

get hasUnits(): boolean;

/\*\*

\* If [[value]] is an integer according to [[isInt]], returns it rounded to

\* that integer. Otherwise, throws an error.

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

assertInt(name?: string): number;

/\*\*

\* Returns [[value]] if it's within `min` and `max`. If [[value]] is equal to

\* `min` or `max` according to Sass's equality, returns `min` or `max`

\* respectively. Otherwise, throws an error.

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

assertInRange(min: number, max: number, name?: string): number;

/\*\*

\* If this number has no units, returns it. Otherwise, throws an error.

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

assertNoUnits(name?: string): SassNumber;

/\*\*

\* If this number has `unit` as its only unit (and as a numerator), returns

\* this number. Otherwise, throws an error.

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

assertUnit(unit: string, name?: string): SassNumber;

/\*\* Whether this number has `unit` as its only unit (and as a numerator). \*/

hasUnit(unit: string): boolean;

/\*\*

\* Whether this has exactly one numerator unit, and that unit is compatible

\* with `unit`.

\*/

compatibleWithUnit(unit: string): boolean;

/\*\*

\* Returns a copy of this number, converted to the units represented by

\* `newNumerators` and `newDenominators`.

\*

\* @throws `Error` if this number's units are incompatible with

\* `newNumerators` and `newDenominators`; or if this number is unitless and

\* either `newNumerators` or `newDenominators` are not empty, or vice-versa.

\*

\* @param newNumerators - The numerator units to convert this number to. This

\* may be either a plain JavaScript array or an immutable [[List]] from the

\* [`immutable` package](https://immutable-js.com/).

\*

\* @param newDenominators - The denominator units to convert this number to.

\* This may be either a plain JavaScript array or an immutable [[List]] from

\* the [`immutable` package](https://immutable-js.com/).

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

convert(

newNumerators: string[] | List<string>,

newDenominators: string[] | List<string>,

name?: string

): SassNumber;

/\*\*

\* Returns a copy of this number, converted to the same units as `other`.

\*

\* @throws `Error` if this number's units are incompatible with `other`'s

\* units, or if either number is unitless but the other is not.

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*

\* @param otherName - The name of the function argument `other` came from

\* (without the `$`) if it came from an argument. Used for error reporting.

\*/

convertToMatch(

other: SassNumber,

name?: string,

otherName?: string

): SassNumber;

/\*\*

\* Returns [[value]], converted to the units represented by `newNumerators`

\* and `newDenominators`.

\*

\* @throws `Error` if this number's units are incompatible with

\* `newNumerators` and `newDenominators`; or if this number is unitless and

\* either `newNumerators` or `newDenominators` are not empty, or vice-versa.

\*

\* @param newNumerators - The numerator units to convert [[value]] to. This

\* may be either a plain JavaScript array or an immutable [[List]] from the

\* [`immutable` package](https://immutable-js.com/).

\*

\* @param newDenominators - The denominator units to convert [[value]] to.

\* This may be either a plain JavaScript array or an immutable [[List]] from

\* the [`immutable` package](https://immutable-js.com/).

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

convertValue(

newNumerators: string[] | List<string>,

newDenominators: string[] | List<string>,

name?: string

): number;

/\*\*

\* Returns [[value]], converted to the same units as `other`.

\*

\* @throws `Error` if this number's units are incompatible with `other`'s

\* units, or if either number is unitless but the other is not.

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*

\* @param otherName - The name of the function argument `other` came from

\* (without the `$`) if it came from an argument. Used for error reporting.

\*/

convertValueToMatch(

other: SassNumber,

name?: string,

otherName?: string

): number;

/\*\*

\* Returns a copy of this number, converted to the units represented by

\* `newNumerators` and `newDenominators`.

\*

\* Unlike [[convert]] this does \*not\* throw an error if this number is

\* unitless and either `newNumerators` or `newDenominators` are not empty, or

\* vice-versa. Instead, it treats all unitless numbers as convertible to and

\* from all units without changing the value.

\*

\* @throws `Error` if this number's units are incompatible with

\* `newNumerators` and `newDenominators`.

\*

\* @param newNumerators - The numerator units to convert this number to. This

\* may be either a plain JavaScript array or an immutable [[List]] from the

\* [`immutable` package](https://immutable-js.com/).

\*

\* @param newDenominators - The denominator units to convert this number to.

\* This may be either a plain JavaScript array or an immutable [[List]] from

\* the [`immutable` package](https://immutable-js.com/).

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

coerce(

newNumerators: string[] | List<string>,

newDenominators: string[] | List<string>,

name?: string

): SassNumber;

/\*\*

\* Returns a copy of this number, converted to the units represented by

\* `newNumerators` and `newDenominators`.

\*

\* Unlike [[convertToMatch]] this does \*not\* throw an error if this number is

\* unitless and either `newNumerators` or `newDenominators` are not empty, or

\* vice-versa. Instead, it treats all unitless numbers as convertible to and

\* from all units without changing the value.

\*

\* @throws `Error` if this number's units are incompatible with `other`'s

\* units.

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*

\* @param otherName - The name of the function argument `other` came from

\* (without the `$`) if it came from an argument. Used for error reporting.

\*/

coerceToMatch(

other: SassNumber,

name?: string,

otherName?: string

): SassNumber;

/\*\*

\* Returns [[value]], converted to the units represented by `newNumerators` and

\* `newDenominators`.

\*

\* Unlike [[convertValue]] this does \*not\* throw an error if this number is

\* unitless and either `newNumerators` or `newDenominators` are not empty, or

\* vice-versa. Instead, it treats all unitless numbers as convertible to and

\* from all units without changing the value.

\*

\* @throws `Error` if this number's units are incompatible with

\* `newNumerators` and `newDenominators`.

\*

\* @param newNumerators - The numerator units to convert [[value]] to. This

\* may be either a plain JavaScript array or an immutable [[List]] from the

\* [`immutable` package](https://immutable-js.com/).

\*

\* @param newDenominators - The denominator units to convert [[value]] to.

\* This may be either a plain JavaScript array or an immutable [[List]] from

\* the [`immutable` package](https://immutable-js.com/).

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*/

coerceValue(

newNumerators: string[] | List<string>,

newDenominators: string[] | List<string>,

name?: string

): number;

/\*\*

\* Returns [[value]], converted to the units represented by `newNumerators`

\* and `newDenominators`.

\*

\* Unlike [[convertValueToMatch]] this does \*not\* throw an error if this

\* number is unitless and either `newNumerators` or `newDenominators` are not

\* empty, or vice-versa. Instead, it treats all unitless numbers as

\* convertible to and from all units without changing the value.

\*

\* @throws `Error` if this number's units are incompatible with `other`'s

\* units.

\*

\* @param name - The name of the function argument `this` came from (without

\* the `$`) if it came from an argument. Used for error reporting.

\*

\* @param otherName - The name of the function argument `other` came from

\* (without the `$`) if it came from an argument. Used for error reporting.

\*/

coerceValueToMatch(

other: SassNumber,

name?: string,

otherName?: string

): number;

}