# normalize-range

Utility for normalizing a numeric range, with a wrapping function useful for polar coordinates.

[Build Status](https://travis-ci.org/jamestalmage/normalize-range) [Coverage Status](https://coveralls.io/github/jamestalmage/normalize-range?branch=master) [Code Climate](https://codeclimate.com/github/jamestalmage/normalize-range) [Dependency Status](https://david-dm.org/jamestalmage/normalize-range) [devDependency Status](https://david-dm.org/jamestalmage/normalize-range#info=devDependencies)

[NPM](https://nodei.co/npm/normalize-range/)

## Usage

var nr = require('normalize-range');

nr.wrap(0, 360, 400);

//=> 40

nr.wrap(0, 360, -90);

//=> 270

nr.limit(0, 100, 500);

//=> 100

nr.limit(0, 100, -20);

//=> 0

// There is a convenient currying function

var wrapAngle = nr.curry(0, 360).wrap;

var limitTo10 = nr.curry(0, 10).limit;

wrapAngle(-30);

//=> 330

## API

### wrap(min, max, value)

Normalizes a values that "wraps around". For example, in a polar coordinate system, 270˚ can also be represented as -90˚. For wrapping purposes we assume max is functionally equivalent to min, and that wrap(max + 1) === wrap(min + 1). Wrap always assumes that min is *inclusive*, and max is *exclusive*. In other words, if value === max the function will wrap it, and return min, but min will not be wrapped.

nr.wrap(0, 360, 0) === 0;

nr.wrap(0, 360, 360) === 0;

nr.wrap(0, 360, 361) === 1;

nr.wrap(0, 360, -1) === 359;

You are not restricted to whole numbers, and ranges can be negative.

var π = Math.PI;

var radianRange = nr.curry(-π, π);

redianRange.wrap(0) === 0;

nr.wrap(π) === -π;

nr.wrap(4 \* π / 3) === -2 \* π / 3;

### limit(min, max, value)

Normalize the value by bringing it within the range. If value is greater than max, max will be returned. If value is less than min, min will be returned. Otherwise, value is returned unaltered. Both ends of this range are *inclusive*.

### test(min, max, value, [minExclusive], [maxExclusive])

Returns true if value is within the range, false otherwise. It defaults to inclusive on both ends of the range, but that can be changed by setting minExclusive and/or maxExclusive to a truthy value.

### validate(min, max, value, [minExclusive], [maxExclusive])

Returns value or throws an error if value is outside the specified range.

### name(min, max, value, [minExclusive], [maxExclusive])

Returns a string representing this range in [range notation](https://en.wikipedia.org/wiki/Interval_(mathematics)#Classification_of_intervals).

### curry(min, max, [minExclusive], [maxExclusive])

Convenience method for currying all method arguments except value.

var angle = require('normalize-range').curry(-180, 180, false, true);

angle.wrap(270)

//=> -90

angle.limit(200)

//=> 180

angle.test(0)

//=> true

angle.validate(300)

//=> throws an Error

angle.toString() // or angle.name()

//=> "[-180,180)"

#### min

*Required*  
Type: number

The minimum value (inclusive) of the range.

#### max

*Required*  
Type: number

The maximum value (exclusive) of the range.

#### value

*Required*  
Type: number

The value to be normalized.

#### returns

Type: number

The normalized value.

## Building and Releasing

* npm test: tests, linting, coverage and style checks.
* npm run watch: autotest mode for active development.
* npm run debug: run tests without coverage (istanbul can obscure line #'s)

Release via cut-release tool.

## License

MIT © [James Talmage](http://github.com/jamestalmage)