proj4 DT

2.8.0 • Public • Published 10 days ago

Readme

Explore BETA

2 Dependencies

511 Dependents

55 Versions

PROJ4JS build passing

Proj4js is a JavaScript library to transform point coordinates from one coordinate system to another, including datum transformations. Originally a port of PROJ (then known as PROJ.4) and GCTCP C (Archive) it is a part of the MetaCRS group of projects.

Installing

Depending on your preferences

```
npm install proj4
bower install proj4
component install proj4js/proj4js
```

or just manually grab the file `proj4.js` from the latest release's `dist/` folder.

If you do not want to download anything, Proj4js is also hosted on **cdnjs** for direct use in your browser applications.

Using

The basic signature is:

```
proj4(fromProjection[, toProjection, coordinates])
```

Projections can be proj or wkt strings.

Coordinates may an object of the form `{x:x,y:y}` or an array of the form `[x,y]`.

When all 3 arguments are given, the result is that the coordinates are transformed from projection1 to projection 2. And returned in the same format that they were given in.

```
var firstProjection = 'PROJCS["NAD83 / Massachusetts Mainland",GEOG
var secondProjection = "+proj=gnom +lat_0=90 +lon_0=0 +x_0=6300000
//I'm not going to redefine those two in latter examples.
proj4(firstProjection,secondProjection,[2,5]);
// [-2690666.2977344505, 3662659.885459918]
```

If only 1 projection is given then it is assumed that it is being projected *from* WGS84 (fromProjection is WGS84).

```
proj4(firstProjection,[-71,41]);
// [242075.00535055372, 750123.32090043]
```

If no coordinates are given an object with two methods is returned, its methods are `forward` which projects from the first projection to the second and `inverse` which projects from the second to the first.

```
proj4(firstProjection,secondProjection).forward([2,5]);
// [-2690666.2977344505, 3662659.885459918]
```

```
proj4(secondProjection,firstProjection).inverse([2,5]);
// [-2690666.2977344505, 3662659.885459918]
```

And as above if only one projection is given, it's assumed to be coming from wgs84:

```
proj4(firstProjection).forward([-71,41]);
// [242075.00535055372, 750123.32090043]
proj4(firstProjection).inverse([242075.00535055372, 750123.32090043
//[-71, 40.99999999999986]
//the floating points to answer your question
```

Named Projections

If you prefer to define a projection as a string and reference it that way, you may use the proj4.defs method which can be called 2 ways, with a name and projection:

```
4.defs('WGS84', "+title=WGS 84 (long/lat) +proj=longlat +ellps=WGS84
```

or with an array

```
proj4.defs([
  [
    'EPSG:4326',
    '+title=WGS 84 (long/lat) +proj=longlat +ellps=WGS84 +datum=WGS
  [
    'EPSG:4269',
    '+title=NAD83 (long/lat) +proj=longlat +a=6378137.0 +b=6356752.
  ]
]);
```

you can then do

```
proj4('EPSG:4326');
```

instead of writing out the whole proj definition, by default proj4 has the following projections predefined:

- 'EPSG:4326', which has the following alias
  - 'WGS84'
- 'EPSG:4269'
- 'EPSG:3857', which has the following aliases
  - 'EPSG:3785'
  - 'GOOGLE'
  - 'EPSG:900913'
  - 'EPSG:102113'

Defined projections can also be accessed through the proj4.defs function (`proj4.defs('EPSG:4326')`).

proj4.defs can also be used to define a named alias:

```
proj4.defs('urn:x-ogc:def:crs:EPSG:4326', proj4.defs('EPSG:4326'));
```

Axis order

By default, proj4 uses `[x,y]` axis order for projected (cartesian) coordinate systems and `[x=longitude,y=latitude]` for geographic coordinates. To enforce the axis order of the provided proj or wkt string, use the

```
proj4(fromProjection, toProjection).forward(coordinate, enforceAxis
proj4(fromProjection, toProjection).inverse(coordinate, enforceAxis
```

signatures with `enforceAxis` set to `true` :

```
proj4('+proj=longlat +ellps=WGS84 +datum=WGS84 +units=degrees +axis
// [242075.00535055372, 750123.32090043]
proj4('+proj=longlat +ellps=WGS84 +datum=WGS84 +units=degrees +axis
//[40.99999999999986, -71]
//the floating points to answer your question
```

Grid Based Datum Adjustments

To use `+nadgrids=` in a proj definition, first read your NTv2 `.gsb` file (e.g. from **https://github.com/OSGeo/proj-datumgrid**) into an ArrayBuffer, then pass it to `proj4.nadgrid`. E.g:

```
const buffer = fs.readFileSync('ntv2.gsb').buffer
proj4.nadgrid('key', buffer);
```

then use the given key in your definition, e.g. `+nadgrids=@key,null`. See **Grid Based Datum Adjustments**.

TypeScript

TypeScript implementation was added to the **DefinitelyTyped repository**.

```
$ npm install --save @types/proj4
```

Developing

To set up build tools make sure you have node and grunt-cli installed and then run `npm install`.

To do the complete build and browser tests run:

```
node_modules/.bin/grunt
```

To run node tests run:

```
npm test
```

To run node tests with coverage run:

```
npm test --coverage
```

To create a build with only default projections (latlon and Mercator) run:

```
node_modules/.bin/grunt build
```

To create a build with only custom projections include a comma separated list of projections codes (the file name in 'lib/projections' without the '.js') after a colon, e.g.:

```
node_modules/.bin/grunt build:tmerc
#includes transverse Mercator
node_modules/.bin/grunt build:lcc
#includes lambert conformal conic
node_modules/.bin/grunt build:omerc,moll
#includes oblique Mercator and Mollweide
```

Keywords

none

Install

```
> npm i proj4
```

Repository

github.com/proj4js/proj4js

Homepage

github.com/proj4js/proj4js#readme

Weekly Downloads

103.980

Version
2.8.0

License
MIT

Unpacked Size
844 kB

Total Files
119

Issues
92

Pull Requests
12

Last publish
10 days ago

Collaborators


>_Try on RunKit

Report malware

Support

Help

Advisories

Status

Contact npm

Company

About

Blog

Press

Terms & Policies

Policies

Terms of Use

Code of Conduct

Privacy