libnpmsearch

`libnpmsearch` is a Node.js library for programmatically accessing the npm search endpoint. It does **not** support legacy search through `/-/all`.

Table of Contents

Example

```
const search = require('libnpmsearch')

console.log(await search('libnpm'))
=>
[
  {
    name: 'libnpm',
    description: 'programmatic npm API',
    ...etc
  },
  {
    name: 'libnpmsearch',
    description: 'Programmatic API for searching in npm and compatible registries',
    ...etc
  },
  ...more
]
```

Install

```
$ npm install libnpmsearch
```

API

`opts` for `libnpmsearch` commands

The following opts are used directly by `libnpmsearch` itself:

- `opts.limit` - Number of results to limit the query to. Default: 20

- `opts.from` - Offset number for results. Used with `opts.limit` for pagination. Default: 0
- `opts.detailed` - If true, returns an object with `package`, `score`, and `searchScore` fields, with `package` being what would usually be returned, and the other two containing details about how that package scored. Useful for UIs. Default: false
- `opts.sortBy` - Used as a shorthand to set `opts.quality`, `opts.maintenance`, and `opts.popularity` with values that prioritize each one. Should be one of `'optimal'`, `'quality'`, `'maintenance'`, or `'popularity'`. Default: `'optimal'`
- `opts.maintenance` - Decimal number between `0` and `1` that defines the weight of `maintenance` metrics when scoring and sorting packages. Default: `0.65` (same as `opts.sortBy: 'optimal'`)
- `opts.popularity` - Decimal number between `0` and `1` that defines the weight of `popularity` metrics when scoring and sorting packages. Default: `0.98` (same as `opts.sortBy: 'optimal'`)
- `opts.quality` - Decimal number between `0` and `1` that defines the weight of `quality` metrics when scoring and sorting packages. Default: `0.5` (same as `opts.sortBy: 'optimal'`)

`libnpmsearch` uses `npm-registry-fetch`. Most options are passed through directly to that library, so please refer to its own `opts` documentation for options that can be passed in.

A couple of options of note for those in a hurry:

- `opts.token` - can be passed in and will be used as the authentication token for the registry. For other ways to pass in auth details, see the n-r-f docs.

`> search(query, [opts]) -> Promise`

`query` must be either a String or an Array of search terms.

If `opts.limit` is provided, it will be sent to the API to constrain the number of returned results. You may receive more, or fewer results, at the endpoint's discretion.

The returned Promise resolved to an Array of search results with the following format:

```
{
  name: String,
  version: SemverString,
  description: String || null,
  maintainers: [
    {
      username: String,
      email: String
    },
    ...etc
  ] || null,
  keywords: [String] || null,
  date: Date || null
}
```

If `opts.limit` is provided, it will be sent to the API to constrain the number of returned results. You may receive more, or fewer results, at the endpoint's discretion.

For streamed results, see `search.stream`.

Example

```
await search('libnpm')
=>
[
  {
    name: 'libnpm',
    description: 'programmatic npm API',
    ...etc
  },
  {
    name: 'libnpmsearch',
    description: 'Programmatic API for searching in npm and compatible registries',
    ...etc
  },
  ...more
]
```

`> search.stream(query, [opts]) -> Stream`

`query` must be either a String or an Array of search terms.

If `opts.limit` is provided, it will be sent to the API to constrain the number of returned results. You may receive more, or fewer results, at the endpoint's discretion.

The returned Stream emits one entry per search result, with each entry having the following format:

```
{
  name: String,
  version: SemverString,
  description: String || null,
  maintainers: [
    {
      username: String,
      email: String
    },
    ...etc
  ] || null,
  keywords: [String] || null,
  date: Date || null
}
```

For getting results in one chunk, see `search`.

Example

```
search.stream('libnpm').on('data', console.log)
=>
// entry 1
{
  name: 'libnpm',
  description: 'programmatic npm API',
  ...etc
```

```
}
// entry 2
{
  name: 'libnpmsearch',
  description: 'Programmatic API for searching in npm and compatible registries',
  ...etc
}
// etc
```