

> Simplified HTTP requests [![Build Status](https://travis-ci.org/sindresorhus/got.svg?branch=master)](https://travis-ci.org/sindresorhus/got) [![Coverage Status](https://coveralls.io/repos/github/sindresorhus/got/badge.svg?branch=master)](https://coveralls.io/github/sindresorhus/got?branch=master) [![Downloads](https://img.shields.io/npm/dm/got.svg)](https://npmjs.com/got) A nicer interface to the built-in [`http`](http://nodejs.org/api/http.html) module. It supports following redirects, promises, streams, retries, automagically handling gzip/deflate and some convenience options. Created because [`request`](https://github.com/request/request) is bloated \*(several megabytes!)\*. ## Install \*\*WARNING: Node.js 4 or higher is required for got@6 and above.\*\* For older Node.js versions use [got@5](https://github.com/sindresorhus/got/tree/v5.x). ``` $ npm install --save got ``` ## Usage ```js const fs = require('fs'); const got = require('got'); got('todomvc.com') .then(response => { console.log(response.body); //=> ' ...' }) .catch(error => { console.log(error.response.body); //=> 'Internal server error ...' }); // Streams got.stream('todomvc.com').pipe(fs.createWriteStream('index.html')); // For POST, PUT and PATCH methods got.stream returns a WritableStream fs.createReadStream('index.html').pipe(got.stream.post('todomvc.com')); ``` ### API It's a `GET` request by default, but can be changed in `options`. #### got(url, [options]) Returns a Promise for a `response` object with a `body` property, a `url` property with the request URL or the final URL after redirects, and a `requestUrl` property with the original request URL. ##### url Type: `string`, `object` The URL to request or a [`http.request` options](https://nodejs.org/api/http.html#http\_http\_request\_options\_callback) object. Properties from `options` will override properties in the parsed `url`. ##### options Type: `object` Any of the [`http.request`](http://nodejs.org/api/http.html#http\_http\_request\_options\_callback) options. ###### body Type: `string`, `buffer`, `readableStream`, `object` \*This is mutually exclusive with stream mode.\* Body that will be sent with a `POST` request. If present in `options` and `options.method` is not set, `options.method` will be set to `POST`. If `content-length` or `transfer-encoding` is not set in `options.headers` and `body` is a string or buffer, `content-length` will be set to the body length. If `body` is a plain object, it will be stringified with [`querystring.stringify`](https://nodejs.org/api/querystring.html#querystring\_querystring\_stringify\_obj\_sep\_eq\_options) and sent as `application/x-www-form-urlencoded`. ###### encoding Type: `string`, `null`

Default: `'utf8'` Encoding to be used on `setEncoding` of the response data. If `null`, the body is returned as a Buffer. ###### json Type: `boolean`

Default: `false` \*This is mutually exclusive with stream mode.\* Parse response body with `JSON.parse` and set `accept` header to `application/json`. ###### query Type: `string`, `object`

Query string object that will be added to the request URL. This will override the query string in `url`. ###### timeout Type: `number`, `object` Milliseconds to wait for a server to send response headers before aborting request with `ETIMEDOUT` error. Option accepts `object` with separate `connect` and `socket` fields for connection and socket inactivity timeouts. ###### retries Type: `number`, `function`

Default: `5` Number of request retries when network errors happens. Delays between retries counts with function `1000 \* Math.pow(2, retry) + Math.random() \* 100`, where `retry` is attempt number (starts from 0). Option accepts `function` with `retry` and `error` arguments. Function must return delay in milliseconds (`0` return value cancels retry). \*\*Note:\*\* if `retries` is `number`, `ENOTFOUND` and `ENETUNREACH` error will not be retried (see full list in [`is-retry-allowed`](https://github.com/floatdrop/is-retry-allowed/blob/master/index.js#L12) module). ###### followRedirect Type: `boolean`

Default: `true` Defines if redirect responses should be followed automatically. #### Streams #### got.stream(url, [options]) `stream` method will return Duplex stream with additional events: ##### .on('request', request) `request` event to get the request object of the request. \*\*Tip\*\*: You can use `request` event to abort request: ```js got.stream('github.com') .on('request', req => setTimeout(() => req.abort(), 50)); ``` ##### .on('response', response) `response` event to get the response object of the final request. ##### .on('redirect', response, nextOptions) `redirect` event to get the response object of a redirect. The second argument is options for the next request to the redirect location. ##### .on('error', error, body, response) `error` event emitted in case of protocol error (like `ENOTFOUND` etc.) or status error (4xx or 5xx). The second argument is the body of the server response in case of status error. The third argument is response object. #### got.get(url, [options]) #### got.post(url, [options]) #### got.put(url, [options]) #### got.patch(url, [options]) #### got.head(url, [options]) #### got.delete(url, [options]) Sets `options.method` to the method name and makes a request. ## Errors Each error contains (if available) `statusCode`, `statusMessage`, `host`, `hostname`, `method` and `path` properties to make debugging easier. In Promise mode, the `response` is attached to the error. #### got.RequestError When a request fails. Contains a `code` property with error class code, like `ECONNREFUSED`. #### got.ReadError When reading from response stream fails. #### got.ParseError When `json` option is enabled and `JSON.parse` fails. #### got.HTTPError When server response code is not 2xx. Contains `statusCode` and `statusMessage`. #### got.MaxRedirectsError When server redirects you more than 10 times. ## Proxies You can use the [`tunnel`](https://github.com/koichik/node-tunnel) module with the `agent` option to work with proxies: ```js const got = require('got'); const tunnel = require('tunnel'); got('todomvc.com', { agent: tunnel.httpOverHttp({ proxy: { host: 'localhost' } }) }); ``` ## Cookies You can use the [`cookie`](https://github.com/jshttp/cookie) module to include cookies in a request: ```js const got = require('got'); const cookie = require('cookie'); got('google.com', { headers: { cookie: cookie.serialize('foo', 'bar') } }); ``` ## Form data You can use the [`form-data`](https://github.com/form-data/form-data) module to create POST request with form data: ```js const fs = require('fs'); const got = require('got'); const FormData = require('form-data'); const form = new FormData(); form.append('my\_file', fs.createReadStream('/foo/bar.jpg')); got.post('google.com', { body: form }); ``` ## OAuth You can use the [`oauth-1.0a`](https://github.com/ddo/oauth-1.0a) module to create a signed OAuth request: ```js const got = require('got'); const crypto = require('crypto'); const OAuth = require('oauth-1.0a'); const oauth = OAuth({ consumer: { key: process.env.CONSUMER\_KEY, secret: process.env.CONSUMER\_SECRET }, signature\_method: 'HMAC-SHA1', hash\_function: (baseString, key) => crypto.createHmac('sha1', key).update(baseString).digest('base64') }); const token = { key: process.env.ACCESS\_TOKEN, secret: process.env.ACCESS\_TOKEN\_SECRET }; const url = 'https://api.twitter.com/1.1/statuses/home\_timeline.json'; got(url, { headers: oauth.toHeader(oauth.authorize({url, method: 'GET'}, token)), json: true }); ``` ## Unix Domain Sockets Requests can also be sent via [unix domain sockets](http://serverfault.com/questions/124517/whats-the-difference-between-unix-socket-and-tcp-ip-socket). Use the following URL scheme: `PROTOCOL://unix:SOCKET:PATH`. - `PROTOCOL` - `http` or `https` \*(optional)\* - `SOCKET` - absolute path to a unix domain socket, e.g. `/var/run/docker.sock` - `PATH` - request path, e.g. `/v2/keys` ```js got('http://unix:/var/run/docker.sock:/containers/json'); // or without protocol (http by default) got('unix:/var/run/docker.sock:/containers/json'); ``` ## Tip It's a good idea to set the `'user-agent'` header so the provider can more easily see how their resource is used. By default, it's the URL to this repo. ```js const got = require('got'); const pkg = require('./package.json'); got('todomvc.com', { headers: { 'user-agent': `my-module/${pkg.version} (https://github.com/username/my-module)` } }); ``` ## Related - [gh-got](https://github.com/sindresorhus/gh-got) - Convenience wrapper for interacting with the GitHub API - [travis-got](https://github.com/samverschueren/travis-got) - Convenience wrapper for interacting with the Travis API ## Created by [![Sindre Sorhus](https://avatars.githubusercontent.com/u/170270?v=3&s=100)](https://sindresorhus.com) | [![Vsevolod Strukchinsky](https://avatars.githubusercontent.com/u/365089?v=3&s=100)](https://github.com/floatdrop) ---|--- [Sindre Sorhus](https://sindresorhus.com) | [Vsevolod Strukchinsky](https://github.com/floatdrop) ## License MIT © [Sindre Sorhus](https://sindresorhus.com)