
Stay Classy, Facebook

FBgraph is a nodejs module that provides easy access to the facebook graph api

downloads 40k/month

Oh noooooooooesss - MOAR facebook

I created this because I wanted to access FB's graph from [node](#). The libraries I found, felt clunky to me, and I needed an excuse to create a node module.

All calls will return [json](#). Facebook sometimes (on friend requests, deleting test users, access token request) decides to just return a [string](#) or [true](#) or redirects directly to the image. I say **nay-nay!** Let's make it Disney, and keep things consistent!

Installation via npm

```
1 $ npm install fbgraph
2
3 var graph = require('fbgraph');
```

Authentication

If you get an accesstoken via some other Oauth module like everyauth , connect-auth or node-oauth you can just set the access token directly. Most [get](#) calls, and pretty much all [post](#) calls will require an [access_token](#)

Static access token (used on all calls)

```
1 graph.setAccessToken(access_token);
```

To use a specific access token for a particular request

```
1 // pass it in as part of the url
2 graph.post(userId + "/feed?access_token=007", wallPost, function(
3   err, res) {
4   // returns the post id
5   console.log(res); // { id: xxxxx}
6 });
```

This is how you would get authenticated using only the `fbgraph` module. More details below on the **express app** section

```
1 // get authorization url
2 var authUrl = graph.getOauthUrl({
3   "client_id":    conf.client_id
4   , "redirect_uri": conf.redirect_uri
5 });
6
7 // shows dialog
8 res.redirect(authUrl);
9
10 // after user click, auth `code` will be set
11 // we'll send that and get the access token
12 graph.authorize({
13   "client_id":    conf.client_id
14   , "redirect_uri": conf.redirect_uri
15   , "client_secret": conf.client_secret
16   , "code":       req.query.code
17 }, function (err, facebookRes) {
18   res.redirect('/loggedIn');
19 });
```

Securing API calls

Facebook recommends adding the `appsecret_proof` parameter to all API calls to verify that the access tokens are coming from a valid app. You can make this happen automatically by calling `graph.setAppSecret(app_secret)`, which will be used on all calls to generate the `appsecret_proof` hash that is sent to Facebook. Make sure you also set the access token for the user via `graph.setAccessToken`.

Extending access token expiration time

If you want to extend the expiration time of your short-living access token, you may use `extendAccessToken` method as it is shown below:

```
1 // extending static access token
2 graph.extendAccessToken({
3   "client_id":    conf.client_id
4   , "client_secret": conf.client_secret
5 }, function (err, facebookRes) {
6   console.log(facebookRes);
7 });
8
9 // extending specific access token
```

```
10 graph.extendAccessToken({
11   "access_token": client_access_token
12   , "client_id":   conf.client_id
13   , "client_secret": conf.client_secret
14 }, function (err, facebookRes) {
15   console.log(facebookRes);
16 });
```

How requests are made

All calls are made using the request nodejs module **Why?** something to do with wheels and re-invention.

Request options are directly mapped and can be set like so:

```
1 var options = {
2   timeout: 3000
3   , pool:   { maxSockets: Infinity }
4   , headers: { connection: "keep-alive" }
5 };
6
7 graph
8   .setOptions(options)
9   .get("zuck", function(err, res) {
10     console.log(res); // { id: '4', name: 'Mark Zuckerberg'... }
11   });
```

Possible options can be found on the request github page

`followRedirect` cannot be overridden and has a default value of `false` `encoding` will have `utf-8` as default if nothing is set

Request Object

The request object is exposed as a property on graph object. So that all the request api can be accessed.

```
1 var graphObject = graph
2   .get("zuck", function(err, res) {
3     console.log(res); // { id: '4', name: 'Mark Zuckerberg'... }
4   });
5
6 // abort the request.
7 graphObject.request.abort();
```

Pagination

Pagination in Facebook is done either with a [cursor](#) or a [next](#) url to call. To simplify the fbgraph API, it's possible to use a fully constructed URL in order to get the next page. See the following example:

```
1 // note: you might want to prevent the callback hell :)
2 graph.get('likes', {limit: 2, access_token: "foobar"}, function(err,
  res) {
3   if(res.paging && res.paging.next) {
4     graph.get(res.paging.next, function(err, res) {
5       // page 2
6     });
7   }
8 });
```

Setting the version of the Graph Api

```
1 graph.setVersion("2.8");
```

See Facebook API changelog for available versions.

Read data from the Graph Api

```
1 graph.get("zuck", function(err, res) {
2   console.log(res); // { id: '4', name: 'Mark Zuckerberg'... }
3 });
```

params in the url

```
1 graph.get("zuck?fields=picture", function(err, res) {
2   console.log(res); // { picture: 'http://profile.ak.fbcdn.net/'... }
3 });
```

params as an object

```
1 var params = { fields: "picture" };
2
3 graph.get("zuck", params, function(err, res) {
4   console.log(res); // { picture: "http://profile.ak.fbcdn.net/..." }
5 });
```

GraphApi calls that **redirect** directly to an image will return a [json](#) response with relevant fields

```
1 graph.get("/zuck/picture", function(err, res) {
2   console.log(res); // { image: true, location: "http://profile.ak.fb
  ..." }
```

```
3 });
```

Search data from the Graph Api

Search for public posts that contain **brogramming**

```
1 var searchOptions = {
2   q:    "brogramming"
3   , type: "post"
4 };
5
6 graph.search(searchOptions, function(err, res) {
7   console.log(res); // {data: [{id: xxx, from: ...}, {id: xxx, from:
8   ...}]}
9 });
```

Publish data to the Graph Api

All publish requests will require an `access token`

only needs to be set once

```
1 graph.setAccessToken(accessToken);
```

Post a message on the user's wall

```
1 var wallPost = {
2   message: "I'm gonna come at you like a spider monkey, chip!"
3 };
4
5 graph.post("/feed", wallPost, function(err, res) {
6   // returns the post id
7   console.log(res); // { id: xxxxx}
8 });
```

Delete a Graph object

To delete a graph object, provide an `object id` and the response will return `{data: true}` or `{data: false}`

```
1 graph.del(postID, function(err, res) {
2   console.log(res); // {data:true}/{data:false}
3 });
```

Performing a batch request

Batching allows you to pass instructions for several operations in a single HTTP request.

```
1 graph.batch([
2   {
3     method: "GET",
4     relative_url: "me" // Get the current user's profile information
5   },
6   {
7     method: "GET",
8     relative_url: "me/friends?limit=50" // Get the first 50 friends of
          the current user
9   }
10 ], function(err, res) {
11   console.log(res);
12   // [
13   //   {
14   //     "code": 200,
15   //     "headers": [
16   //       { "name": "Content-Type", "value": "text/javascript; charset=
          UTF-8" }
17   //     ],
18   //     "body": "{ \"id...\": \"\" }"
19   //   },
20   //   {
21   //     "code": 200,
22   //     "headers": [
23   //       { "name": "Content-Type", "value": "text/javascript; charset=
          UTF-8" }
24   //     ],
25   //     "body": "{ \"data\": ...[{}]} "
26   //   }
27   // ]
28 });
```

Performing a FQL query

A single FQL query is done by sending a query as a string

```
1 var query = "SELECT name FROM user WHERE uid = me()";
2
3 graph.fql(query, function(err, res) {
4   console.log(res); // { data: [ { name: 'Ricky Bobby' } ] }
5 });
```

You can specify additional options by adding a JSON object

```
1 var query = "SELECT name FROM user WHERE uid = me()";
```

```
2 var options = {access_token: "foobar"};
3
4 graph.fql(query, options, function(err, res) {
5   console.log(res); // { data: [ { name: 'Ricky Bobby' } ] }
6 });
```

Performing a FQL Multi-Query

FQL Multi-Queries are done by sending in an object containing the separate queries

```
1 var query = {
2   name:      "SELECT name FROM user WHERE uid = me()"
3   , permissions: "SELECT email, user_about_me, user_birthday FROM
4     permissions WHERE uid = me()"
5 };
6 graph.fql(query, function(err, res) {
7   console.log(res);
8   // { data: [
9     //   { name: 'name', fql_result_set: [{name: 'Ricky Bobby'}] },
10    //   { name: 'permissions', fql_result_set: [{email: 1, user_about_me
11      : 1...}] }
12    // ]}
13  });
```

Rockin' it on an Express App

This example assumes that you have a link on the main page / that points to </auth/facebook>. The user will click this link and get into the facebook authorization flow (if the user hasn't already connected) After **authorizing** the app the user will be redirected to </UserHasLoggedIn>

```
1 npm install --save express fbgraph method-override body-parser
  errorhandler pug
```

```
1 /**
2  * Module dependencies.
3  */
4
5 var express  = require('express')
6   , graph    = require('fbgraph');
7 var app = express();
8 var server = require("http").createServer(app);
9
10
11 // this should really be in a config file!
12 var conf = {
```

```
13     client_id:      'APP-PUBLIC-ID'
14     , client_secret: 'APP-SECRET-ID'
15     , scope:        'email, user_about_me, user_birthday, user_location
16                     , publish_actions'
17     // You have to set http://localhost:3000/ as your website
18     // using Settings -> Add platform -> Website
19     , redirect_uri:  'http://localhost:3000/auth'
20 };
21
22 // Configuration
23 var methodOverride = require('method-override');
24 var bodyParser = require('body-parser');
25 var errorHandler = require('errorhandler');
26
27 app.set('views', __dirname + '/views');
28 // Jade was renamed to pug
29 app.set('view engine', 'pug');
30 app.use(bodyParser.urlencoded({
31     extended: true
32 }));
33 app.use(methodOverride());
34
35 var path = require('path');
36 app.use(express.static(path.join(__dirname, '/public')));
37
38 var env = process.env.NODE_ENV || 'development';
39 if ('development' == env) {
40     app.use(errorHandler({ dumpExceptions: true, showStack: true }));
41 }
42
43 // Routes
44
45 app.get('/', function(req, res){
46     res.render("index", { title: "click link to connect" });
47 });
48
49 app.get('/auth', function(req, res) {
50
51     // we don't have a code yet
52     // so we'll redirect to the oauth dialog
53     if (!req.query.code) {
54         console.log("Performing oauth for some user right now.");
55
56         var authUrl = graph.getOAuthUrl({
57             "client_id":    conf.client_id
58             , "redirect_uri": conf.redirect_uri
59             , "scope":       conf.scope
60         });
61
62         if (!req.query.error) { //checks whether a user denied the app
```

```

        facebook login/permissions
63     res.redirect(authUrl);
64   } else { //req.query.error == 'access_denied'
65     res.send('access denied');
66   }
67 }
68 // If this branch executes user is already being redirected back with
69 // code (whatever that is)
70 else {
71   console.log("Oauth successful, the code (whatever it is) is: ", req
    .query.code);
72   // code is set
73   // we'll send that and get the access token
74   graph.authorize({
75     "client_id":      conf.client_id
76     , "redirect_uri":  conf.redirect_uri
77     , "client_secret": conf.client_secret
78     , "code":          req.query.code
79   }, function (err, facebookRes) {
80     res.redirect('/UserHasLoggedIn');
81   });
82 }
83 });
84
85
86 // user gets sent here after being authorized
87 app.get('/UserHasLoggedIn', function(req, res) {
88   res.render("index", {
89     title: "Logged In"
90   });
91 });
92
93
94 var port = process.env.PORT || 3000;
95 app.listen(port, function() {
96   console.log("Express server listening on port %d", port);
97 });

```

Running tests

Before running the test suite, add your Facebook `appId` and `appSecret` to `tests/config.js`. This is needed to create `test users` and to get a `test access_token`.

```

1 $ npm install
2 $ make test

```

Tests might fail if the Facebook api has an issue.

License

(The MIT License)

Copyright (c) 2011 Cristiano Oliveira <ocean.cris@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the 'Software'), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.