

---

## NodeJS Library for Facebook

build unknown

With facebook-node-sdk you can now easily write the same code and share between your server (nodejs) and the client (Facebook Javascript SDK).

This SDK will report usage of which AppID is using it directly to Facebook.

**Author:** Thuzi

**License:** Apache v2

### Installing facebook-node-sdk

```
1 npm install fb
```

```
1 var FB = require('fb');
```

### Running Samples

Update `appId` and `appSecret` in `samples/scrumptious/config.js`

```
1 npm install
2 cd samples/scrumptious
3 npm install
4 node app.js
```

### Graph Api

#### Get

```
1 var FB = require('fb');
2
3 FB.api('4', function (res) {
4   if(!res || res.error) {
5     console.log(!res ? 'error occurred' : res.error);
6     return;
7   }
8   console.log(res.id);
9   console.log(res.name);
10 });
```

---

## Passing Parameters

```
1 var FB = require('fb');
2
3 FB.api('4', { fields: ['id', 'name'] }, function (res) {
4   if(!res || res.error) {
5     console.log(!res ? 'error occurred' : res.error);
6     return;
7   }
8   console.log(res.id);
9   console.log(res.name);
10 });
```

## Post

```
1 var FB = require('fb');
2 FB.setAccessToken('access_token');
3
4 var body = 'My first post using facebook-node-sdk';
5 FB.api('me/feed', 'post', { message: body}, function (res) {
6   if(!res || res.error) {
7     console.log(!res ? 'error occurred' : res.error);
8     return;
9   }
10   console.log('Post Id: ' + res.id);
11 });
```

## Delete

```
1 var FB = require('fb');
2 FB.setAccessToken('access_token');
3
4 var postId = '1234567890';
5 FB.api(postId, 'delete', function (res) {
6   if(!res || res.error) {
7     console.log(!res ? 'error occurred' : res.error);
8     return;
9   }
10   console.log('Post was deleted');
11 });
```

## Facebook Query Language (FQL)

### Query

---

```
1 var FB = require('fb');
2 FB.setAccessToken('access_token');
3
4 FB.api('fql', { q: 'SELECT uid FROM user WHERE uid=me()' }, function (
  res) {
5   if(!res || res.error) {
6     console.log(!res ? 'error occurred' : res.error);
7     return;
8   }
9   console.log(res.data);
10 });
```

### Multi-query

```
1 var FB = require('fb');
2 FB.setAccessToken('access_token');
3
4 FB.api('fql', { q: [
5   'SELECT uid FROM user WHERE uid=me()',
6   'SELECT name FROM user WHERE uid=me()'
7 ] }, function(res) {
8   if(!res || res.error) {
9     console.log(!res ? 'error occurred' : res.error);
10    return;
11  }
12  console.log(res.data[0].fql_result_set);
13  console.log(res.data[1].fql_result_set);
14 });
```

### Named Multi-query

```
1 var FB = require('fb');
2 FB.setAccessToken('access_token');
3
4 FB.api('fql', { q : {
5   id: 'SELECT uid FROM user WHERE uid=me()',
6   name: 'SELECT name FROM user WHERE uid IN (SELECT uid FROM #id)'
7 } }, function(res) {
8   if(!res || res.error) {
9     console.log(!res ? 'error occurred' : res.error);
10    return;
11  }
12  console.log(res.data[0].fql_result_set);
13  console.log(res.data[1].fql_result_set);
14 });
```

---

## Batch Requests

```
1 var FB = require('fb');
2 FB.setAccessToken('access_token');
3
4 var extractEtag;
5 FB.api('', 'post', {
6   batch: [
7     { method: 'get', relative_url: '4' },
8     { method: 'get', relative_url: 'me/friends?limit=50' },
9     { method: 'get', relative_url: 'fql?q=' + encodeURIComponent('
10      SELECT uid FROM user WHERE uid=me()') }, /* fql */
11     { method: 'get', relative_url: 'fql?q=' + encodeURIComponent(
12       JSON.stringify([
13         'SELECT uid FROM user WHERE uid=me()',
14         'SELECT name FROM user WHERE uid=me()'
15       ])) }, /* fql multi-query */
16     { method: 'get', relative_url: 'fql?q=' + encodeURIComponent(
17       JSON.stringify({
18         id: 'SELECT uid FROM user WHERE uid=me()',
19         name: 'SELECT name FROM user WHERE uid IN (SELECT
20           uid FROM #id)'
21       }) ) }, /* named fql multi-query */
22     { method: 'get', relative_url: '4', headers: { 'If-None-Match':
23       '"7de572574f2a822b65ecd9eb8acef8f476e983e1"' } }, /* etags
24       */
25     { method: 'get', relative_url: 'me/friends?limit=1', name: 'one-
26       friend' /* , omit_response_on_success: false */ },
27     { method: 'get', relative_url: '{result=one-friend:$.data.0.id
28       }/feed?limit=5'}
29   ]
30 }, function(res) {
31   var res0, res1, res2, res3, res4, res5, res6, res7,
32     etag1;
33
34   if(!res || res.error) {
35     console.log(!res ? 'error occurred' : res.error);
36     return;
37   }
38
39   res0 = JSON.parse(res[0].body);
40   res1 = JSON.parse(res[1].body);
41   res2 = JSON.parse(res[2].body);
42   res3 = JSON.parse(res[3].body);
43   res4 = JSON.parse(res[4].body);
44   res5 = res[5].code === 304 ? undefined : JSON.parse(res[5].body);
45   // special case for not-modified responses
```

//

set

---

```

38     res6 = res[6] === null ? null : JSON.parse(res[6].body);
39     res7 = res6 === null ? JSON.parse(res[7].body) : undefined; // set
      result as undefined if previous dependency failed
40
41     if(res0.error) {
42         console.log(res0.error);
43     } else {
44         console.log('Hi ' + res0.name);
45         etag1 = extractETag(res[0]); // use this etag when making the
      second request.
46         console.log(etag1);
47     }
48
49     if(res1.error) {
50         console.log(res1.error);
51     } else {
52         console.log(res1);
53     }
54
55     if(res2.error) {
56         console.log(res2.error);
57     } else {
58         console.log(res2.data);
59     }
60
61     if(res3.error) {
62         console.log(res3.error);
63     } else {
64         console.log(res3.data[0].fql_result_set);
65         console.log(res3.data[1].fql_result_set);
66     }
67
68     if(res4.error) {
69         console.log(res4.error);

```

```

res5
as
undefined
if
response
wasn
'
t
modified
.

```

```

70     } else {
71         console.log(res4.data[0].fql_result_set);
72         console.log(res4.data[0].fql_result_set);
73     }
74
75     // check if there are any new updates
76     if(typeof res5 !== "undefined") {
77         // make sure there was no error
78         if(res5.error) {
79             console.log(error);
80         } else {
81             console.log('new update available');
82             console.log(res5);
83         }
84     }
85     else {
86         console.log('no updates');
87     }
88
89     // check if dependency executed successfully
90     if(res[6] === null) {
91         // then check if the result it self doesn't have any errors.
92         if(res7.error) {
93             console.log(res7.error);
94         } else {
95             console.log(res7);
96         }
97     } else {
98         console.log(res6.error);
99     }
100 });
101
102 extractETag = function(res) {
103     var etag, header, headerIndex;
104     for(headerIndex in res.headers) {
105         header = res.headers[headerIndex];
106         if(header.name === 'ETag') {
107             etag = header.value;
108         }
109     }
110     return etag;
111 };

```

## Post

```

1 var FB = require('fb');
2 FB.setAccessToken('access_token');
3
4 var message = 'Hi from facebook-node-js';

```

---

```

5  FB.api('', 'post', {
6      batch: [
7          { method: 'post', relative_url: 'me/feed', body: 'message=' +
              encodeURIComponent(message) }
8      ]
9  }, function (res) {
10     var res0;
11
12     if(!res || res.error) {
13         console.log(!res ? 'error occurred' : res.error);
14         return;
15     }
16
17     res0 = JSON.parse(res[0].body);
18
19     if(res0.error) {
20         console.log(res0.error);
21     } else {
22         console.log('Post Id: ' + res0.id);
23     }
24 });

```

## OAuth Requests

*This is a non-standard behavior and does not work in the official client side FB JS SDK.*

facebook-node-sdk is capable of handling oauth requests which return non-json responses. You can use it by calling `api` method.

### Get facebook application access token

```

1  var FB = require('fb');
2
3  FB.api('oauth/access_token', {
4      client_id: 'app_id',
5      client_secret: 'app_secret',
6      grant_type: 'client_credentials'
7  }, function (res) {
8      if(!res || res.error) {
9          console.log(!res ? 'error occurred' : res.error);
10         return;
11     }
12
13     var accessToken = res.access_token;
14 });

```

---

## Exchange code for access token

```
1 var FB = require('fb');
2
3 FB.api('oauth/access_token', {
4   client_id: 'app_id',
5   client_secret: 'app_secret',
6   redirect_uri: 'http://yoururl.com/callback',
7   code: 'code'
8 }, function (res) {
9   if(!res || res.error) {
10     console.log(!res ? 'error occurred' : res.error);
11     return;
12   }
13
14   var accessToken = res.access_token;
15   var expires = res.expires ? res.expires : 0;
16 });
```

You can safely extract the code from the url using the `url` module. Always make sure to handle invalid oauth callback as well as error.

```
1 var url = require('url');
2 var FB = require('fb');
3
4 var urlToParse = 'http://yoururl.com/callback?code=.....#_=_';
5 var result = url.parse(urlToParse, true);
6 if(result.query.error) {
7   if(result.query.error_description) {
8     console.log(result.query.error_description);
9   } else {
10     console.log(result.query.error);
11   }
12   return;
13 } else if (!result.query.code) {
14   console.log('not a oauth callback');
15   return;
16 }
17
18 var code = result.query.code;
```

## Extend expiry time of the access token

```
1 var FB = require('fb');
2
3 FB.api('oauth/access_token', {
4   client_id: 'client_id',
5   client_secret: 'client_secret',
```

---

```
6     grant_type: 'fb_exchange_token',
7     fb_exchange_token: 'existing_access_token'
8 }, function (res) {
9     if(!res || res.error) {
10         console.log(!res ? 'error occurred' : res.error);
11         return;
12     }
13
14     var accessToken = res.access_token;
15     var expires = res.expires ? res.expires : 0;
16 });
```

## Legacy REST Api

**Although Legacy REST Api is supported by facebook-node-sdk, it is highly discouraged to be used, as Facebook is in the process of deprecating the Legacy REST Api.**

### Get

```
1 var FB = require('fb');
2
3 FB.api({ method: 'users.getInfo', uids: ['4'], fields: ['uid', 'name']
4 }, function (res) {
5     if(!res || res.error_msg) {
6         console.log(!res ? 'error occurred' : res.error_msg);
7         return;
8     }
9
10    console.log('User Id: ' + res[0].uid);
11    console.log('Name: ' + res[0].name);
12 });
```

### Post

```
1 var FB = require('fb');
2 FB.setAccessToken('access_token');
3
4 var message = 'Hi from facebook-node-sdk';
5 FB.api({ method: 'stream.publish', message: message }, function (res) {
6     if(!res || res.error_msg) {
7         console.log(!res ? 'error occurred' : res.error_msg);
8         return;
9     }
10
11    console.log(res);
```

---

```
12 });
```

## Delete

```
1 var FB = require('fb');
2 FB.setAccessToken('access_token');
3
4 var postId = '.....';
5 FB.api({ method: 'stream.remove', post_id: postId }, function (res) {
6     if(!res || res.error_msg) {
7         console.log(!res ? 'error occurred' : res.error_msg);
8         return;
9     }
10
11     console.log(res);
12 });
```

## Access Tokens

### setAccessToken

*This is a non-standard api and does not exist in the official client side FB JS SDK.*

```
1 var FB = require('fb');
2 FB.setAccessToken('access_token');
```

If you want to use the api compatible with FB JS SDK, pass `access_token` as parameter.

```
1 FB.api('me', { fields: ['id', 'name'], access_token: 'access_token' },
2     function (res) {
3         console.log(res);
4     }
```

### getAccessToken

*Unlike `setAccessToken` this is a standard api and exists in FB JS SDK.*

```
1 var FB = require('fb');
2 FB.setAccessToken('access_token');
3 var accessToken = FB.getAccessToken();
```

---

## AppSecret Proof

For improved security, as soon as you provide an app secret and an access token, the library automatically computes and adds the `appsecret_proof` parameter to your requests.

## Configuration options

### options

*This is a non-standard api and does not exist in the official client side FB JS SDK.*

When this method is called with no parameters it will return all of the current options.

```
1 var FB = require('fb');
2 var options = FB.options();
```

When this method is called with a string it will return the value of the option if exists, null if it does not.

```
1 var timeout = FB.options('timeout');
```

When this method is called with an object it will merge the object onto the previous options object.

```
1 FB.options({accessToken: 'abc'}); //equivalent to calling
  setAccessToken('abc')
2 FB.options({timeout: 1000, accessToken: 'XYZ'}); //will set timeout and
  accessToken options
3 var timeout = FB.options('timeout'); //will get a timeout of 1000
4 var accessToken = FB.options('accessToken'); //will get the accessToken
  of 'XYZ'
```

The existing options are:

- \* `'accessToken'` string representing the facebook accessToken to be used for requests. This is the same option that is updated by the `setAccessToken` and `getAccessToken` methods.
- \* `'appSecret'` string representing the facebook application secret.
- \* `'proxy'` string representing an HTTP proxy to be used. Support proxy Auth with Basic Auth, embedding the auth info in the uri: `'http://[username:password@]proxy[:port]'` (parameters in brackets are optional).
- \* `'timeout'` integer number of milliseconds to wait for a response. Requests that have not received a response in *X* ms. If set to null or 0 no timeout will exist. On timeout an error object will be returned to the api callback with the error code of `'ETIMEDOUT'` (example below).

`'scope'` and `'redirectUri'` have been whitelisted in options for convenience. These value will not be automatically added when using any of the sdk apis unlike the above options. These are whitelisted so you can use it to pass values using the same `FB` object.

---

## version

*This is a non-standard api and does not exist in the official client side FB JS SDK.*

Gets the string representation of the facebook-node-sdk library version.

```
1 var FB = require('fb');
2 var version = FB.version;
```

## Parsing Signed Request

### parseSignedRequest

*This is a non-standard api and does not exist in the official client side FB JS SDK.*

```
1 var FB = require('fb');
2
3 var signedRequestValue = 'signed_request_value';
4 var appSecret = 'app_secret';
5
6 var signedRequest = FB.parseSignedRequest(signedRequestValue,
    appSecret);
7 if(signedRequest) {
8     var accessToken = signedRequest.oauth_token;
9     var userId = signedRequest.user_id;
10    var userCountry = signedRequest.user.country;
11 }
```

*Note: parseSignedRequest will return undefined if validation fails. Always remember to check the result of parseSignedRequest before accessing the result.*

If you already set the appSecret in options, you can ignore the second parameter when calling parseSignedRequest. If you do pass the second parameter it will use the appSecret passed in parameter instead of using appSecret from options.

If appSecret is absent, parseSignedRequest will throw an error.

```
1 var FB = require('fb');
2 FB.options({ 'appSecret': 'app_secret' });
3
4 var signedRequestValue = 'signed_request_value';
5
6 var signedRequest = FB.parseSignedRequest(signedRequestValue);
7 if(signedRequest) {
8     var accessToken = signedRequest.oauth_token;
9     var userId = signedRequest.user_id;
10    var userCountry = signedRequest.user.country;
11 }
```

---

## Error handling

*Note: facebook is not consistent with their error format, and different systems can fail causing different error formats*

Some examples of various error codes you can check for: \* `'ECONNRESET'` - connection reset by peer \* `'ETIMEDOUT'` - connection timed out \* `'E_SOCKETTIMEDOUT'` - socket timed out \* `'JSONPARSE'` - could not parse JSON response, happens when the FB API has availability issues. It sometimes returns HTML

```
1 var FB = require('fb');
2 FB.options({timeout: 1, accessToken: 'access_token'});
3
4 FB.api('/me', function (res) {
5     if(res && res.error) {
6         if(res.error.code === 'ETIMEDOUT') {
7             console.log('request timeout');
8         }
9         else {
10            console.log('error', res.error);
11        }
12    }
13    else {
14        console.log(res);
15    }
16 });
```

## Node style callback with FB.napi

*This is a non-standard api and does not exist in the official client side FB JS SDK.*

`FB.napi` takes the same input as `FB.api`. Only the callback parameters is different. In the original `FB.api`, the callback expects one parameter which is the response. In `FB.napi` the callback expects two parameters instead of one and follows the node standards. The first parameter is an error which is always of type `FB.FacebookApiException` and the second parameter is the same response as in `FB.api`. Error response can be accessed using `error.response` which is the same response as the response when using `FB.api`

```
1 var FB = require('fb');
2
3 FB.napi('4', function(error, response) {
4     if(error) {
5         if(error.response.error.code === 'ETIMEDOUT') {
6             console.log('request timeout');
7         }
8         else {
```

---

```
9         console.log('error', error.message);
10     }
11 } else {
12     console.log(response);
13 }
14 });
```

**FB.napi** was added especially to make it easier to work with async control flow libraries.

Here are some examples of using facebook-node-sdk with Step.

You will need to install **step**.

```
1 npm install step
```

### FB.api with Step

```
1 var FB      = require('fb'),
2     Step    = require('step');
3
4 Step(
5     function getUser() {
6         var self = this;
7         FB.api('4', function(res) {
8             if(!res || res.error) {
9                 self(new Error('Error occured'));
10            } else {
11                self(null, res);
12            }
13        });
14    },
15    function processResult(err, res) {
16        if(err) throw err;
17        console.log(res);
18    }
19 );
```

### FB.napi with Step

Simplified version of facebook-node-sdk async callbacks using **FB.napi**.

```
1 var FB      = require('fb'),
2     Step    = require('step');
3
4 Step(
5     function getUser() {
6         FB.napi('4', this);
```

---

```
7     },  
8     function processResult(err, res) {  
9         if(err) throw err;  
10        console.log(res);  
11    }  
12 );
```