prisma-labs / graphql-request

🛰 Minimal GraphQL client supporting Node and browsers for scripts or simple apps

#graphql #graphql-client #nodejs #typescript #lightweight #graphql-request

134 commits **28** branches **0** packages **26** releases **14** contributors MIT

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download** | **Gitpod**

divyenduz Merge pull request #114 from brikou/feature/npmignore ... ✓ Latest commit `b0abe80` on 21 Sep 2018

.circleci	Revert "fix: build by installing peer dep in ci"	2 years ago
examples	Use TData interface	2 years ago
src	Revert "feat: support parsed query (#92)"	2 years ago
tests	Revert "feat: support parsed query (#92)"	2 years ago
.gitignore	remove log	3 years ago
.npmignore	Add missing .npmignore	2 years ago
LICENSE	Create LICENSE	3 years ago
README.md	Merge pull request #110 from brikou/feature/examples_ts_typings_only	2 years ago
package.json	[RFR] Add examples dir (#105)	2 years ago
renovate.json	Update renovate.json	2 years ago
tsconfig.json	fix: add esnext.asynciterable to tsconfig.json	2 years ago
tslint.json	refactor: Fixed linting	2 years ago

yarn.lock [RFR] Add examples dir (#105) 2 years ago

README.md

graphql-request

CircleCI npm package 1.8.2

📡 Minimal GraphQL client supporting Node and browsers for scripts or simple apps

Features

- Most **simple and lightweight** GraphQL client
- Promise-based API (works with `async` / `await`)
- Typescript support (Flow coming soon)

Install

```
npm install graphql-request
```

Quickstart

Send a GraphQL query with a single line of code. ▶️ Try it out.

```
import { request } from 'graphql-request'

const query = `{
  Movie(title: "Inception") {
    releaseDate
    actors {
      name
    }
  }
}`

request('https://api.graph.cool/simple/v1/movies', query).then(data =>
  console.log(data)
)
```

Usage

```
import { request, GraphQLClient } from 'graphql-request'

// Run GraphQL queries/mutations using a static function
request(endpoint, query, variables).then(data => console.log(data))

// ... or create a GraphQL client instance to send requests
const client = new GraphQLClient(endpoint, { headers: {} })
client.request(query, variables).then(data => console.log(data))
```

Examples

Authentication via HTTP header

```
import { GraphQLClient } from 'graphql-request'

async function main() {
  const endpoint = 'https://api.graph.cool/simple/v1/cixos23120m0n0173veiiwrjr'

  const graphQLClient = new GraphQLClient(endpoint, {
    headers: {
      authorization: 'Bearer MY_TOKEN',
    },
  })

  const query = /* GraphQL */ `
    {
      Movie(title: "Inception") {
        releaseDate
        actors {
          name
        }
      }
    }
  `

  const data = await graphQLClient.request(query)
  console.log(JSON.stringify(data, undefined, 2))
}

main().catch(error => console.error(error))
```

TypeScript Source

Passing more options to fetch

```
import { GraphQLClient } from 'graphql-request'

async function main() {
  const endpoint = 'https://api.graph.cool/simple/v1/cixos23120m0n0173veiiwrjr'

  const graphQLClient = new GraphQLClient(endpoint, {
    credentials: 'include',
    mode: 'cors',
  })

  const query = /* GraphQL */ `
    {
      Movie(title: "Inception") {
        releaseDate
```

```
          actors {
            name
          }
        }
      }
    `

    const data = await graphQLClient.request(query)
    console.log(JSON.stringify(data, undefined, 2))
  }

  main().catch(error => console.error(error))
```

TypeScript Source

Using variables

```
  import { request } from 'graphql-request'

  async function main() {
    const endpoint = 'https://api.graph.cool/simple/v1/cixos23120m0n0173veiiwrjr'

    const query = /* GraphQL */ `
      query getMovie($title: String!) {
        Movie(title: $title) {
          releaseDate
          actors {
            name
          }
        }
      }
    `

    const variables = {
      title: 'Inception',
    }

    const data = await request(endpoint, query, variables)
    console.log(JSON.stringify(data, undefined, 2))
  }

  main().catch(error => console.error(error))
```

TypeScript Source

Error handling

```
import { request } from 'graphql-request'

async function main() {
  const endpoint = 'https://api.graph.cool/simple/v1/cixos23120m0n0173veiiwrjr'

  const query = /* GraphQL */ `
    {
      Movie(title: "Inception") {
        releaseDate
        actors {
          fullname # "Cannot query field 'fullname' on type 'Actor'. Did you mean 'name'?"
        }
      }
    }
  `

  try {
    const data = await request(endpoint, query)
    console.log(JSON.stringify(data, undefined, 2))
  } catch (error) {
    console.error(JSON.stringify(error, undefined, 2))
    process.exit(1)
  }
}

main().catch(error => console.error(error))
```

TypeScript Source

Using `require` instead of `import`

```
const { request } = require('graphql-request')

async function main() {
  const endpoint = 'https://api.graph.cool/simple/v1/cixos23120m0n0173veiiwrjr'

  const query = /* GraphQL */ `
    {
      Movie(title: "Inception") {
        releaseDate
        actors {
          name
        }
      }
    }
  `

  const data = await request(endpoint, query)
  console.log(JSON.stringify(data, undefined, 2))
}

main().catch(error => console.error(error))
```

Cookie support for `node`

```
npm install fetch-cookie
```

```
require('fetch-cookie/node-fetch')(require('node-fetch'))

import { GraphQLClient } from 'graphql-request'

async function main() {
  const endpoint = 'https://api.graph.cool/simple/v1/cixos23120m0n0173veiiwrjr'
```

```
    const graphQLClient = new GraphQLClient(endpoint, {
      headers: {
        authorization: 'Bearer MY_TOKEN',
      },
    })

    const query = /* GraphQL */ `
      {
        Movie(title: "Inception") {
          releaseDate
          actors {
            name
          }
        }
      }
    `

    const data = await graphQLClient.rawRequest(query)
    console.log(JSON.stringify(data, undefined, 2))
  }

  main().catch(error => console.error(error))
```

TypeScript Source

Receiving a raw response

The `request` method will return the `data` or `errors` key from the response. If you need to access the `extensions` key you can use the `rawRequest` method:

```
  import { rawRequest } from 'graphql-request'

  async function main() {
    const endpoint = 'https://api.graph.cool/simple/v1/cixos23120m0n0173veiiwrjr'

    const query = /* GraphQL */ `
      {
        Movie(title: "Inception") {
          releaseDate
          actors {
            name
          }
        }
      }
    `

    const { data, errors, extensions, headers, status } = await rawRequest(
      endpoint,
      query
    )
    console.log(
      JSON.stringify({ data, errors, extensions, headers, status }, undefined, 2)
    )
  }

  main().catch(error => console.error(error))
```

TypeScript Source

More examples coming soon...

- Fragments
- Using `graphql-tag`

FAQ

What's the difference between `graphql-request`, Apollo and Relay?

`graphql-request` is the most minimal and simplest to use GraphQL client. It's perfect for small scripts or simple apps.

Compared to GraphQL clients like Apollo or Relay, `graphql-request` doesn't have a built-in cache and has no integrations for frontend frameworks. The goal is to keep the package and API as minimal as possible.

So what about Lokka?

Lokka is great but it still requires a lot of setup code to be able to send a simple GraphQL query. `graphql-request` does less work compared to Lokka but is a lot simpler to use.

Help & Community slack 33265

Join our Slack community if you run into issues or have questions. We love talking to you!

< we ❤ open source />