You're probably making an API client

Haroen Viaene - Paris.js January 2018

API economy

API economy

GitHub

API economy

GitHub

TMDb

API economy

GitHub

TMDb

Netlify

API economy

GitHub

TMDb

Netlify

Algolia

API economy

GitHub

TMDb

Netlify

Algolia

Facebook

API economy

GitHub

TMDb

Netlify

Algolia

Facebook

Firebase

API economy

GitHub

TMDb

Netlify

Algolia

Facebook

Firebase

Twitter

API economy

GitHub

TMDb

Netlify

Algolia

Facebook

Firebase

RATP

Twitter

API economy

GitHub

TMDb

Netlify

Algolia

Facebook

Firebase

RATP

Twitter

Dark Sky

API economy

GitHub

TMDb

Netlify

Algolia

Facebook

Firebase

RATP

Twitter

<your own API>

Dark Sky

API economy

Official API client

Facebook

GitHub

Twitter

TMDb

Netlify

Algolia

Firebase

RATP

<your own API>

Dark Sky

API economy

Official API client

Unofficial API clients

Facebook

GitHub

Twitter

TMDb

Netlify

Algolia

Firebase

RATP

Dark Sky

<your own API>

API economy

Official API client	**Unofficial API clients**	**Just REST**
Facebook	GitHub	RATP
Twitter	TMDb	Netlify
Algolia		Dark Sky
Firebase		*<your own API>*

“Use the official API client”

– Me, just now

“Yes, but it doesn’t support **node**”

– You, maybe

"Yes, but it doesn't support **some endpoint**"

– You, maybe

"Yes, but my environment can't use **XHR**"

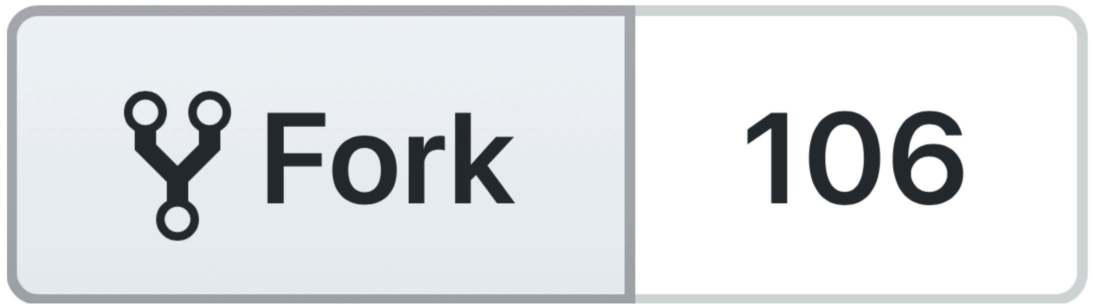
– *You, maybe*

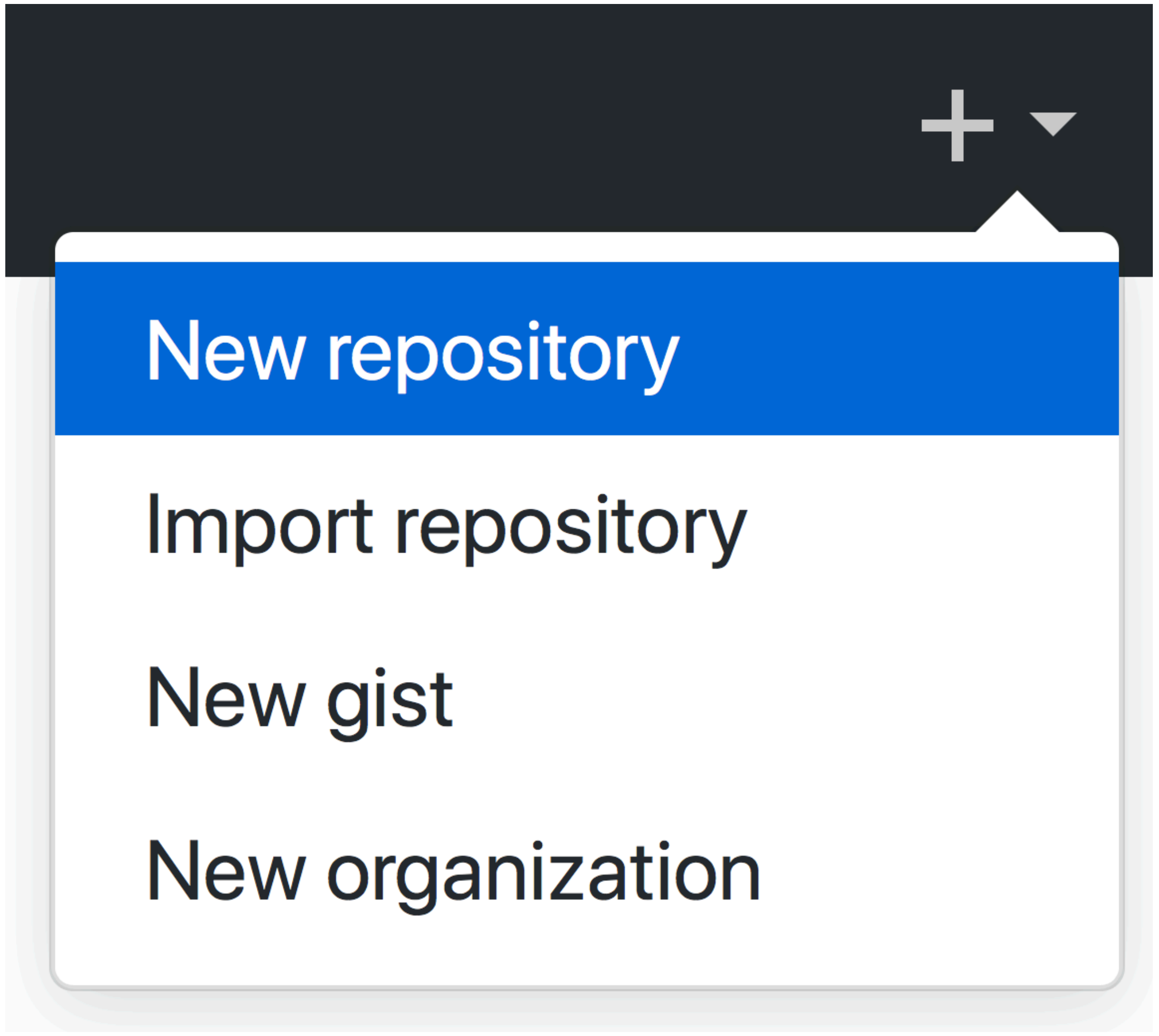
"Yes, but there is none..."

– *You, maybe*

What are my options?

What are my options?





MVP

```
fetch("https://api.com", {
  method: "POST",
  body,
}).then(res ⇒ res.json());
```

Minimum Viable Product

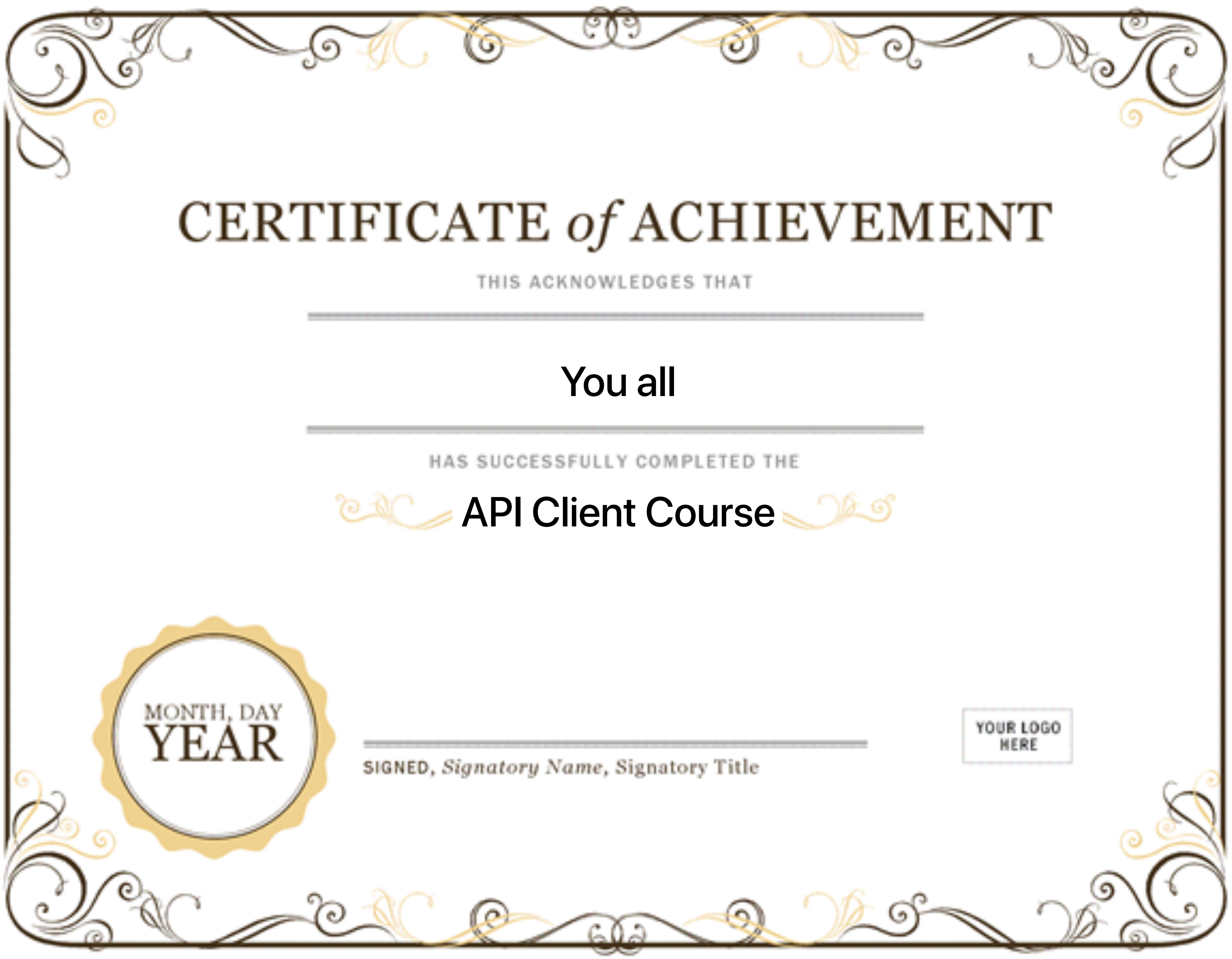
```
fetch("https://api.com", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    "X-API-Key": "OFOCS02NSE4",
  },
  body: JSON.stringify({
    query: "hello",
  }),
}).then(res => res.json());
```

Put them in a file maybe?

```
export const get = body ⇒
  fetch("https://api.com", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "X-API-Key": "OFOCS02NSE4",
    },
    body: JSON.stringify(body),
  }).then(res ⇒ res.json());

// ...other ones
```

Put them in a file maybe?



CERTIFICATE *of* ACHIEVEMENT

THIS ACKNOWLEDGES THAT

You all

HAS SUCCESSFULLY COMPLETED THE

API Client Course

MONTH, DAY
YEAR

SIGNED, *Signatory Name*, Signatory Title

YOUR LOGO HERE

Provide value over `fetch`

```
const get = query ⇒
  fetch("https://api.com", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "X-API-Key": "OFOCS02NSE4",
    },
    body: JSON.stringify({ query }),
  }).then(res ⇒ res.json());
```

Deal with API keys

```
const get = (body, apiKey) ⇒
  fetch("https://api.com", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "X-API-Key": apiKey,
    },
    body: JSON.stringify(body),
  }).then(res ⇒ res.json());
```

Deal with API keys

```
const createClient = apiKey ⇒
  allMethods.map(method ⇒ /* */);
```

Deal with API keys

```
const createClient = apiKey ⇒
  allMethods.map(method ⇒ (...args) ⇒
    method(...args, apiKey)
  );
```

Deal with API keys

```
const createClient = apiKey ⇒
  allMethods.map(method ⇒ ( … args) ⇒
    method( … args, apiKey)
  );

const client = createClient("OFOCS02NSE4");
```

Deal with API keys

```
const createClient = apiKey ⇒
  allMethods.map(method ⇒ (...args) ⇒
    method(...args, apiKey)
  );

const client = createClient("OFOCS02NSE4");

client.get(body);
```

Use “named” arguments

```
const createClient = ({ apiKey }) ⇒
  allMethods.map(method ⇒ args ⇒
    method(args, { apiKey })
  );

const client = createClient({
  apiKey: "OFOCS02NSE4"
});

client.get({ body });
```

Use the same names as your API

```
const get = ({ body }, { apiKey }) ⇒
  fetch("https://api.com/search", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "X-API-Key": apiKey
    },
    body: JSON.stringify(body)
  }).then(res ⇒ res.json());
```

Use the same names as your API

```
const search = ({ body }, { apiKey }) ⇒
  fetch("https://api.com/search", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "X-API-Key": apiKey
    },
    body: JSON.stringify(body)
  }).then(res ⇒ res.json());
```

Fill the gaps in the API

Fill the gaps in the API

Catch-all request (`batch`)

Fill the gaps in the API

Catch-all request (`batch`)

Split up in separate methods (`addObject`, `deleteObject`)

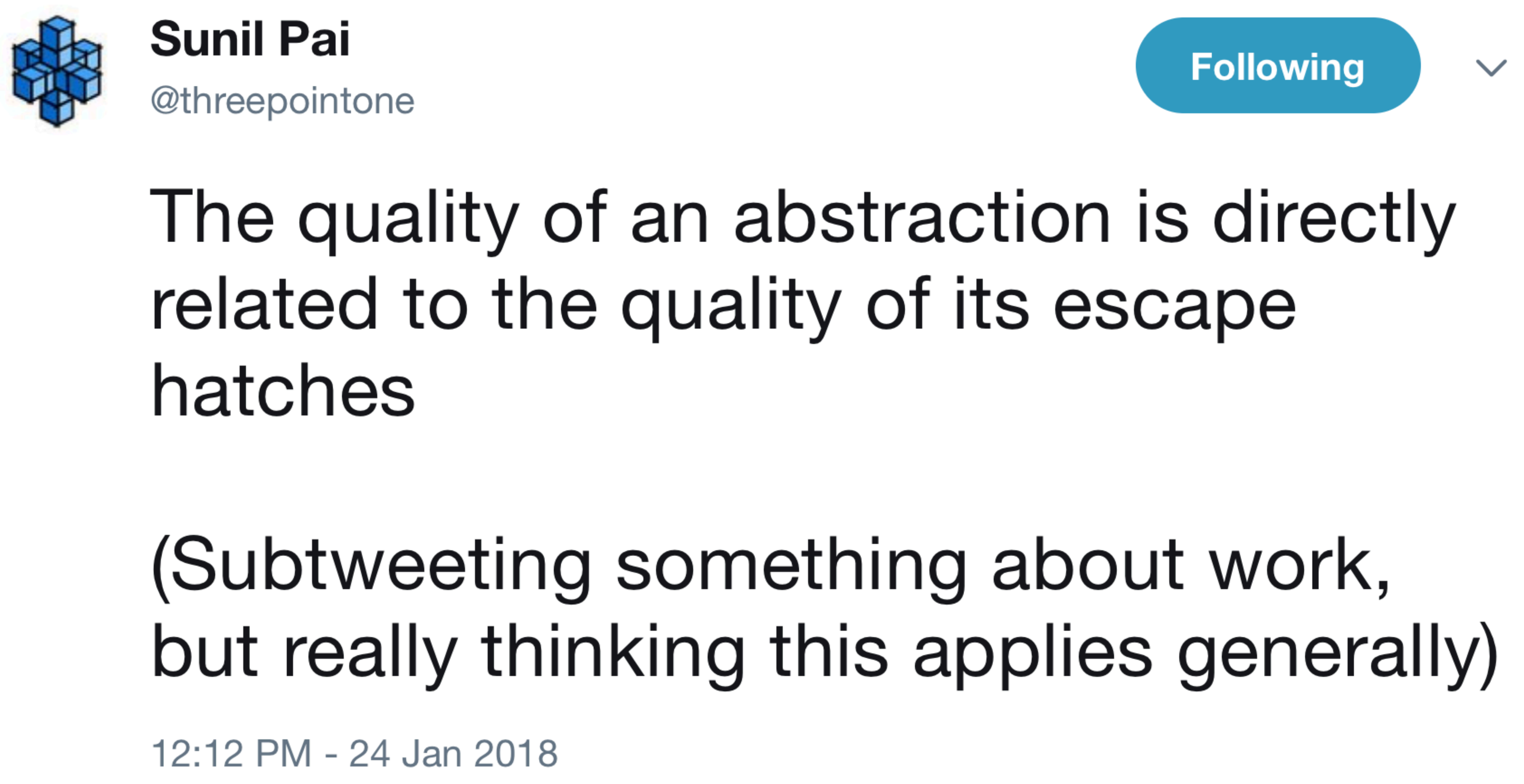
Fill the gaps in the API

Catch-all request (`batch`)

Split up in separate methods (`addObject`, `deleteObject`)

Consistency in client → consistency in API

Allow overriding



Sunil Pai
@threepointone

The quality of an abstraction is directly related to the quality of its escape hatches

(Subtweeting something about work, but really thinking this applies generally)

12:12 PM - 24 Jan 2018

twitter.com/threepointone/status/956122568557199360

Allow overriding

```
const createClient = ({ apiKey }) ⇒
  allMethods.map(method ⇒ (args, extra) ⇒
    method(args, { ...{ apiKey }, ...extra })
  );

const client = createClient({
  apiKey: "OFOCS02NSE4"
});

client.get(
  { body },
  { apiKey: "something-else" }
);
```

Allow overriding

```
const createClient = ({ apiKey }) ⇒
  allMethods.map(method ⇒ (args, extra) ⇒
    method(args, { ...{ apiKey }, ...extra })
  );

const client = createClient({
  apiKey: "OFOCS02NSE4"
});

client.get(
  { body },
  { apiKey: "something-else", requestOptions }
);
```

Allow overriding

RequestOptions

Known headers are sent as headers

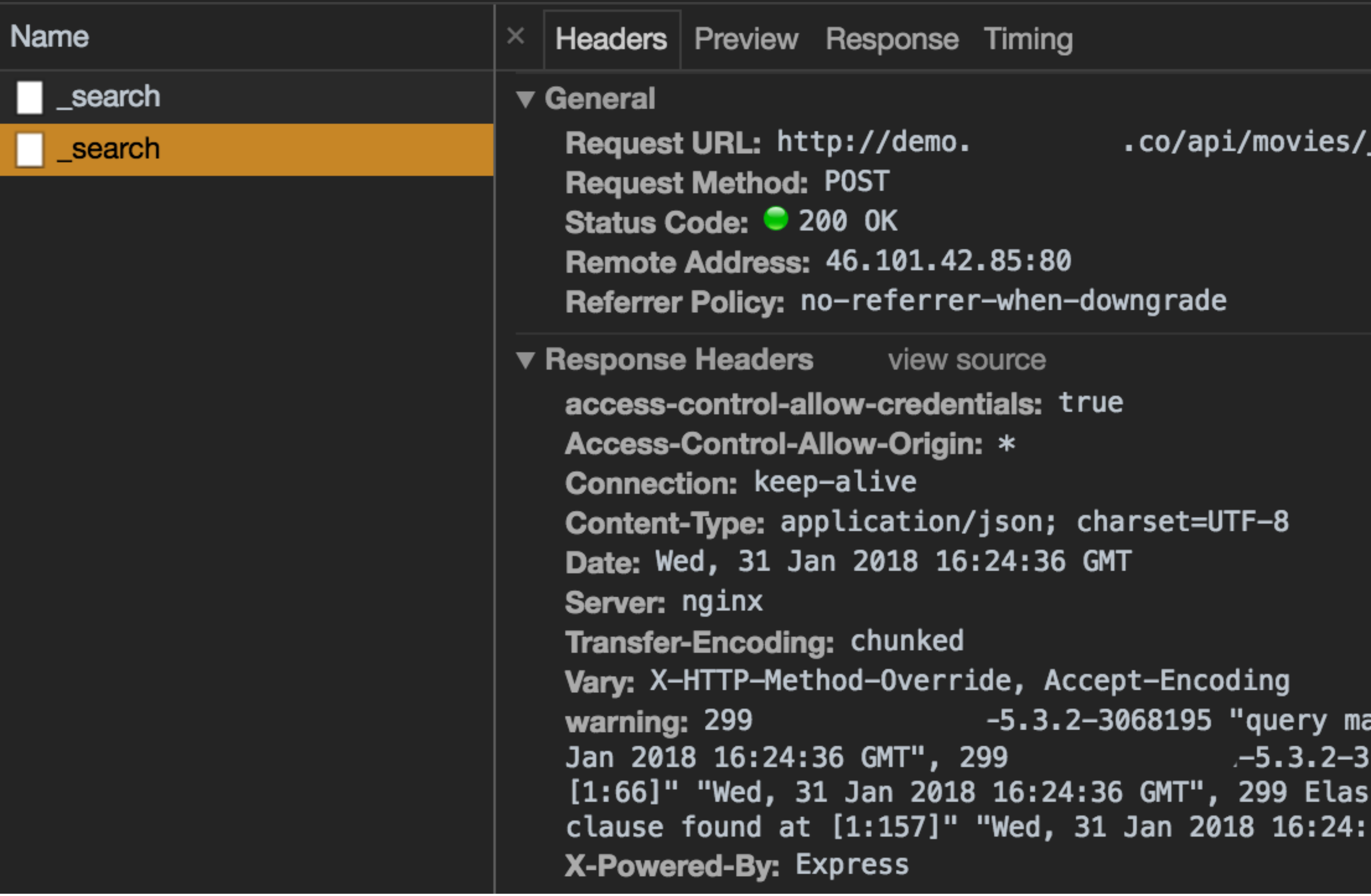
The rest as URL parameters

Separate request from wrappers

```
const request = ({
  path,
  method,
  body,
  apiKey
}) ⇒
  fetch(new URL("https://api.com", path), {
    method,
    headers: {
      "Content-Type": "application/json",
      "X-API-Key": apiKey
    },
    body: JSON.stringify(body)
  }).then(res ⇒ res.json());
```

Separate request from wrappers

```
const search = ({ query }, { apiKey }) ⇒
  request({
    path: "/search",
    method: "POST",
    body: { query },
    apiKey
  });
```



Name
_search
_search

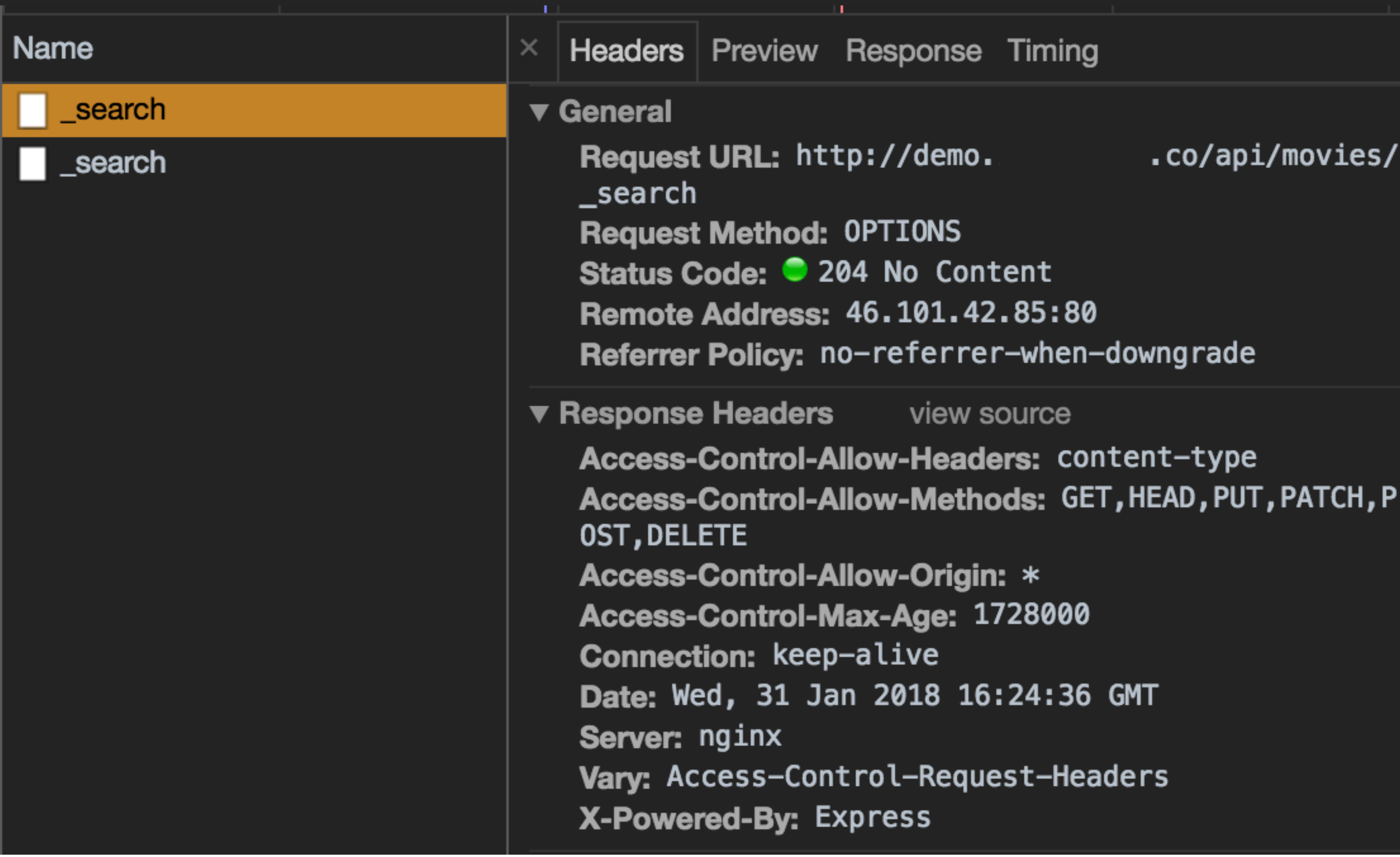
Headers Preview Response Timing

▼ General

```
Request URL: http://demo.          .co/api/movies/
Request Method: POST
Status Code: ● 200 OK
Remote Address: 46.101.42.85:80
Referrer Policy: no-referrer-when-downgrade
```

▼ Response Headers view source

```
access-control-allow-credentials: true
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Type: application/json; charset=UTF-8
Date: Wed, 31 Jan 2018 16:24:36 GMT
Server: nginx
Transfer-Encoding: chunked
Vary: X-HTTP-Method-Override, Accept-Encoding
warning: 299              -5.3.2-3068195 "query ma
Jan 2018 16:24:36 GMT", 299              -5.3.2-3
[1:66]" "Wed, 31 Jan 2018 16:24:36 GMT", 299 Elas
clause found at [1:157]" "Wed, 31 Jan 2018 16:24:
X-Powered-By: Express
```



```
Request URL: http://demo.          .co/api/movies/
_search
Request Method: OPTIONS
Status Code: ● 204 No Content
Remote Address: 46.101.42.85:80
Referrer Policy: no-referrer-when-downgrade
```

```
Access-Control-Allow-Headers: content-type
Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,P
OST,DELETE
Access-Control-Allow-Origin: *
Access-Control-Max-Age: 1728000
Connection: keep-alive
Date: Wed, 31 Jan 2018 16:24:36 GMT
Server: nginx
Vary: Access-Control-Request-Headers
X-Powered-By: Express
```

SimpleRequests

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Methods

`GET`

`HEAD`

`POST`

SimpleRequests

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Headers

```
Accept
                   DPR
Accept-Language
                   Save-Data
Content-Language
                   Viewport-Width
Content-Type
                   Width
Last-Event-ID
```

SimpleRequests

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Content Type

```
application/x-www-form-urlencoded

multipart/form-data

text/plain
```

Overview

Separate from other code

Don’t leak your abstraction

Have a “client” for API keys

Use SimpleRequests

Know when to deviate from your API

Look at other API clients

Thank you!