You're probably making an API client

Haroen Viaene - OdessaJS July 2018

Hi there

Haroen Viaene - @haroenv

Hi there

Haroen

Hi there

Haroen

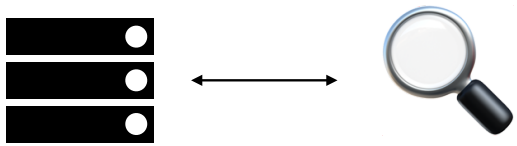
Hi there

Haroen

🔍

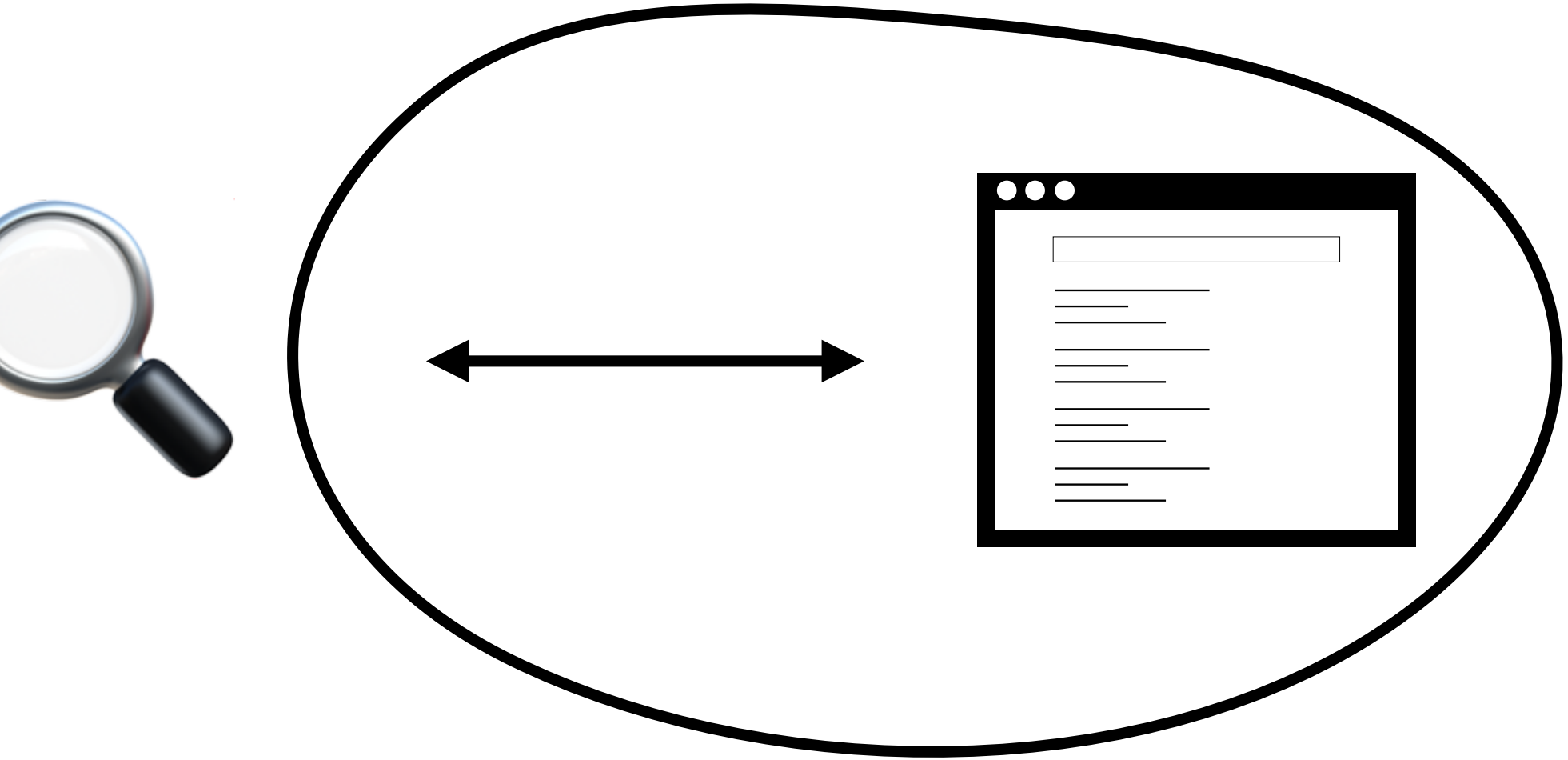
Hi there

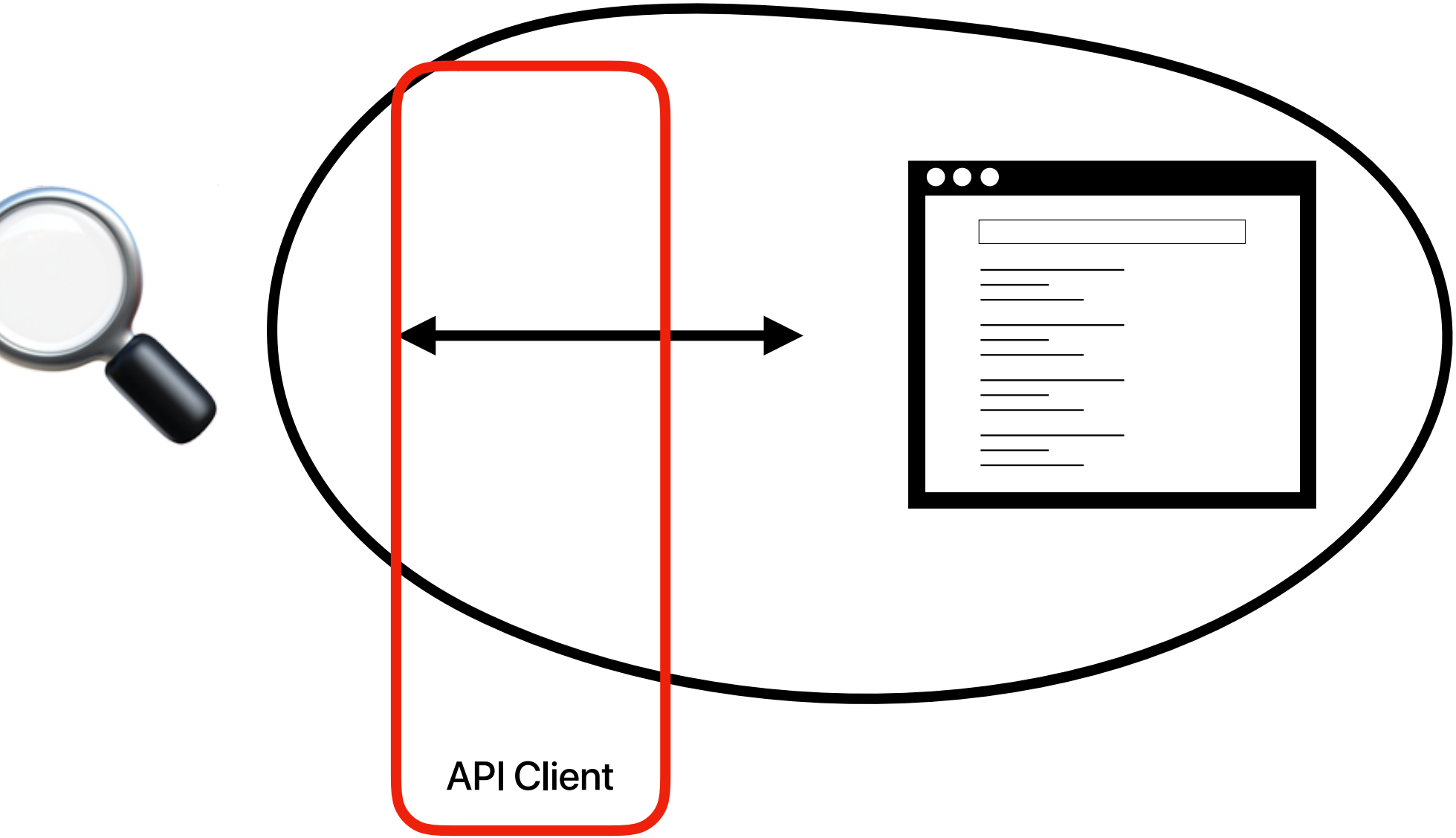
Haroen

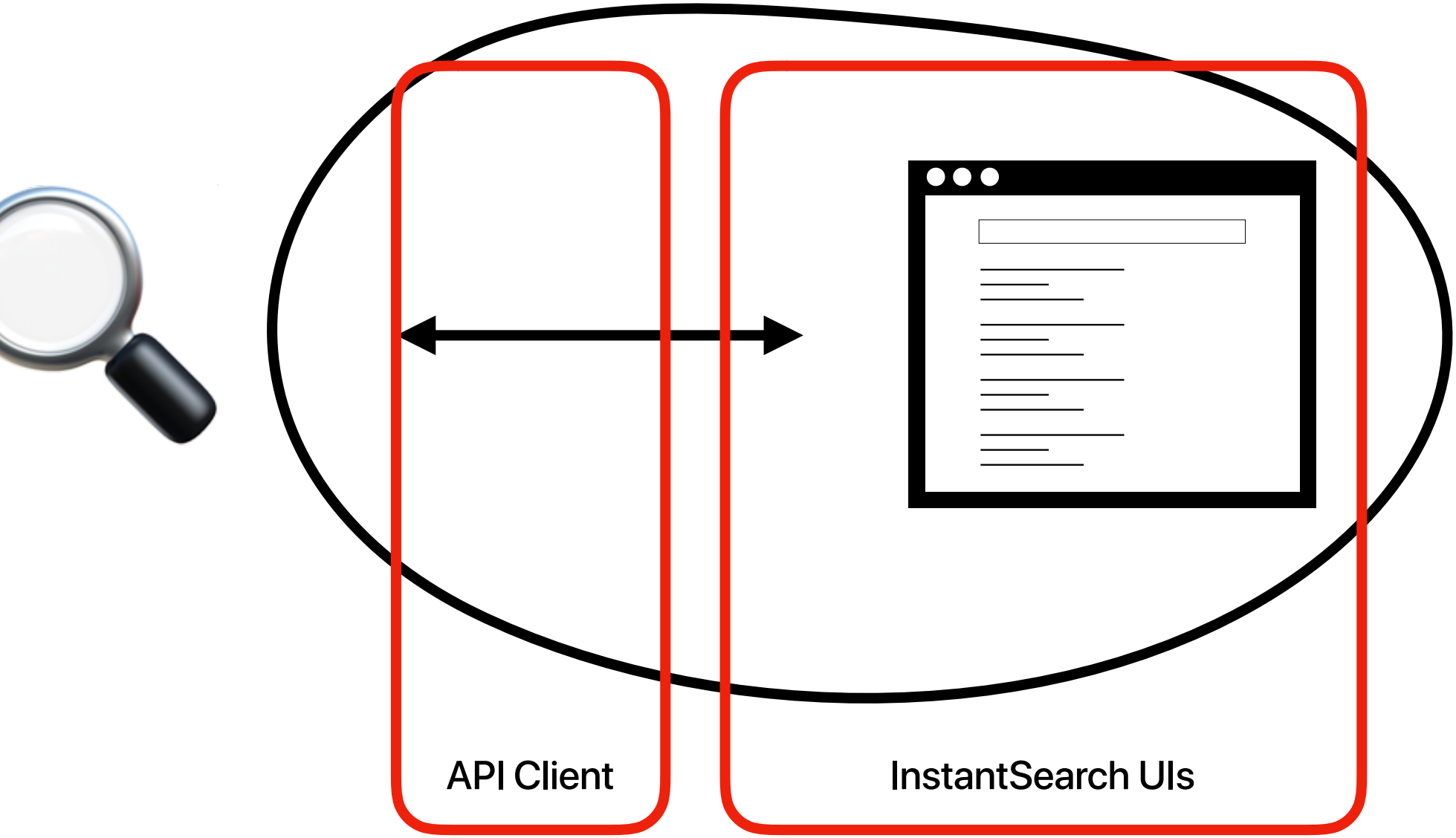


Hi there

Haroen




API Client


API Client
InstantSearch UIs

What will be covered

Terminology

Why API clients

Anatomy of an API client

How to make a good API client

What is

What is an API?

What is an API?

What is an API client?

What is an API?

What is an API client?

What is an abstraction?

Something which gives you access to a capability

Something which gives
you access to **a capability**

Something which gives
you access to a capability

Something which **gives you access** to a capability

What is an abstraction?

Why use an API client?

Abstraction

1

Avoid repetition

Easier testing

3

Edge cases are handled

4

Allows API to be more complex

When should you use an API client?

“Use the official API client”

– Me, just now

“When you do multiple requests to an endpoint”

– Me, just now

MVP

Haroen Viaene - @haroenv

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 query = 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
```

CERTIFICATE *of* ACHIEVEMENT

THIS ACKNOWLEDGES THAT

OdessaJS attendees

HAS SUCCESSFULLY COMPLETED THE

API Client Course

2018

SIGNED, *Haroen Viaene*

algolia

Haroen Viaene - @haroenv algolia

What I've learned

What I've learned

Design **Performance** **Maintainability**

Design

Provide value

Provide value

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

Provide value

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

Provide value

```
query(query)
```

Design

Handle credentials

Haroen Viaene - @haroenv

algolia

Handle credentials

```
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());
```

Handle credentials

PSEUDO CODE

PSEUDO CODE

Handle credentials

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

PSEUDO CODE

Handle credentials

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

PSEUDO CODE

Handle credentials

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

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

PSEUDO CODE

Handle credentials

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

const client = createClient("OFOCS02NSE4");

client.query(body);
```

Design

Use “named” arguments

Haroen Viaene - @haroenv

algolia

Use “named” arguments

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

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

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

Design

Use the same names as your API

Use the same names as your API

```
const query = ({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());
```

Use the same names as your API

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

Design

Fill the gaps in the API

Haroen Viaene - @haroenv

algolia

Fill the gaps in the API

`/batch`

Catch-all

Split up in separate methods (`addObjects`, `deleteObjects`)

Consistency in client > consistency in API

Design

Allow overriding

Allow overriding

Sunil Pai
@threepointone

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

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

Allow overriding

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

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

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

Allow overriding

RequestOptions

Known headers are sent as headers

The rest as URL parameters

Design

Separate request from wrappers

Separate request from wrappers

```
const method = (...) =>
  fetch(...)
  // ...
```

```
const method = (...) =>
  request(...)

const request = (...) =>
  fetch(...)
```

Separate request from wrappers

Separate request from wrappers

```
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 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
  });
```

Design

Everything is async

Everything is async

```
await client.search()
```

```
client.generateApiKey()
```

Everything is async

```
await client.search()
```

```
await client.generateApiKey()
```

What I've learned

Design **Performance** **Maintainability**

Performance

Know one layer of abstraction lower

Know one layer of abstraction lower

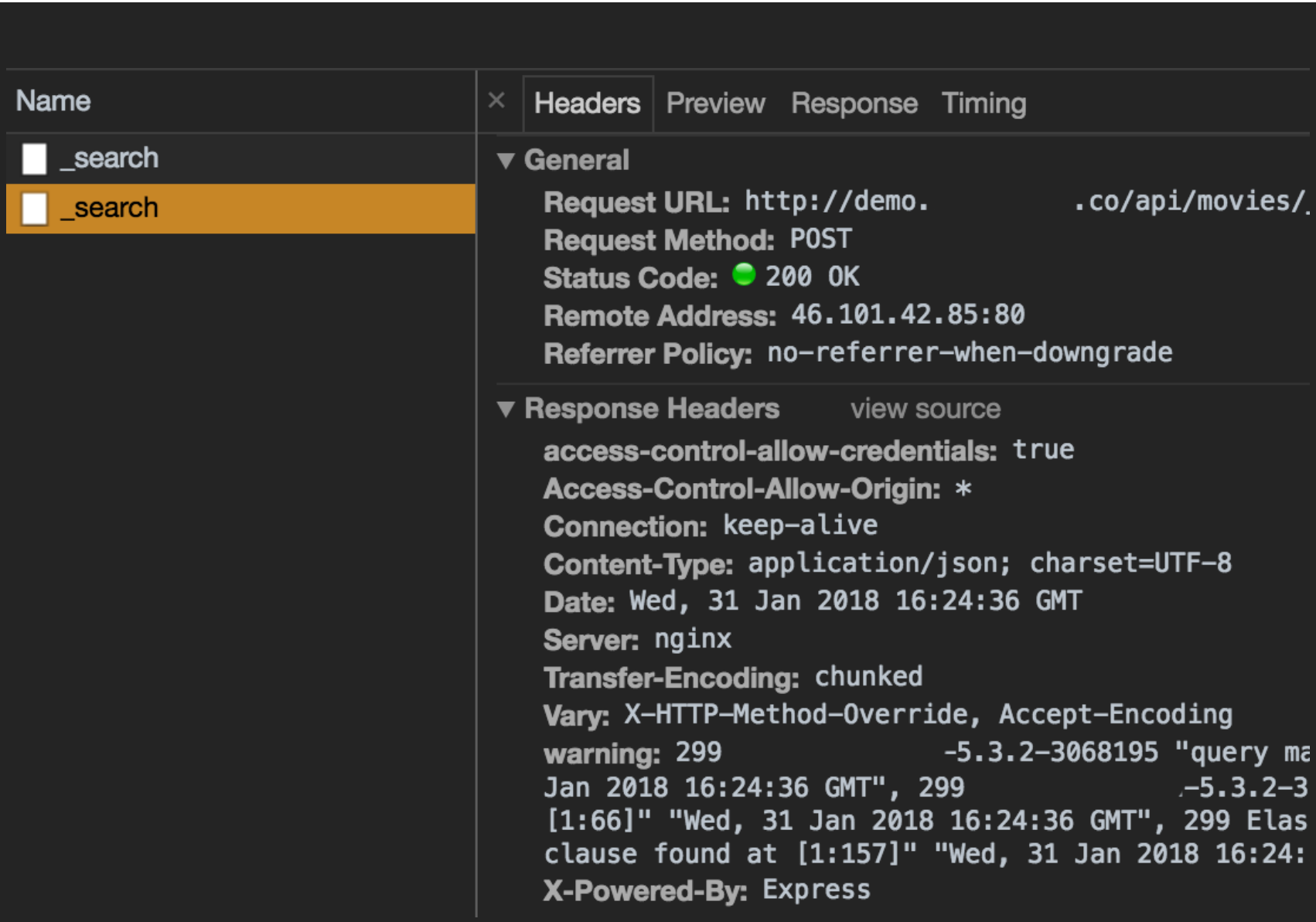
All abstractions are leaky

CORS

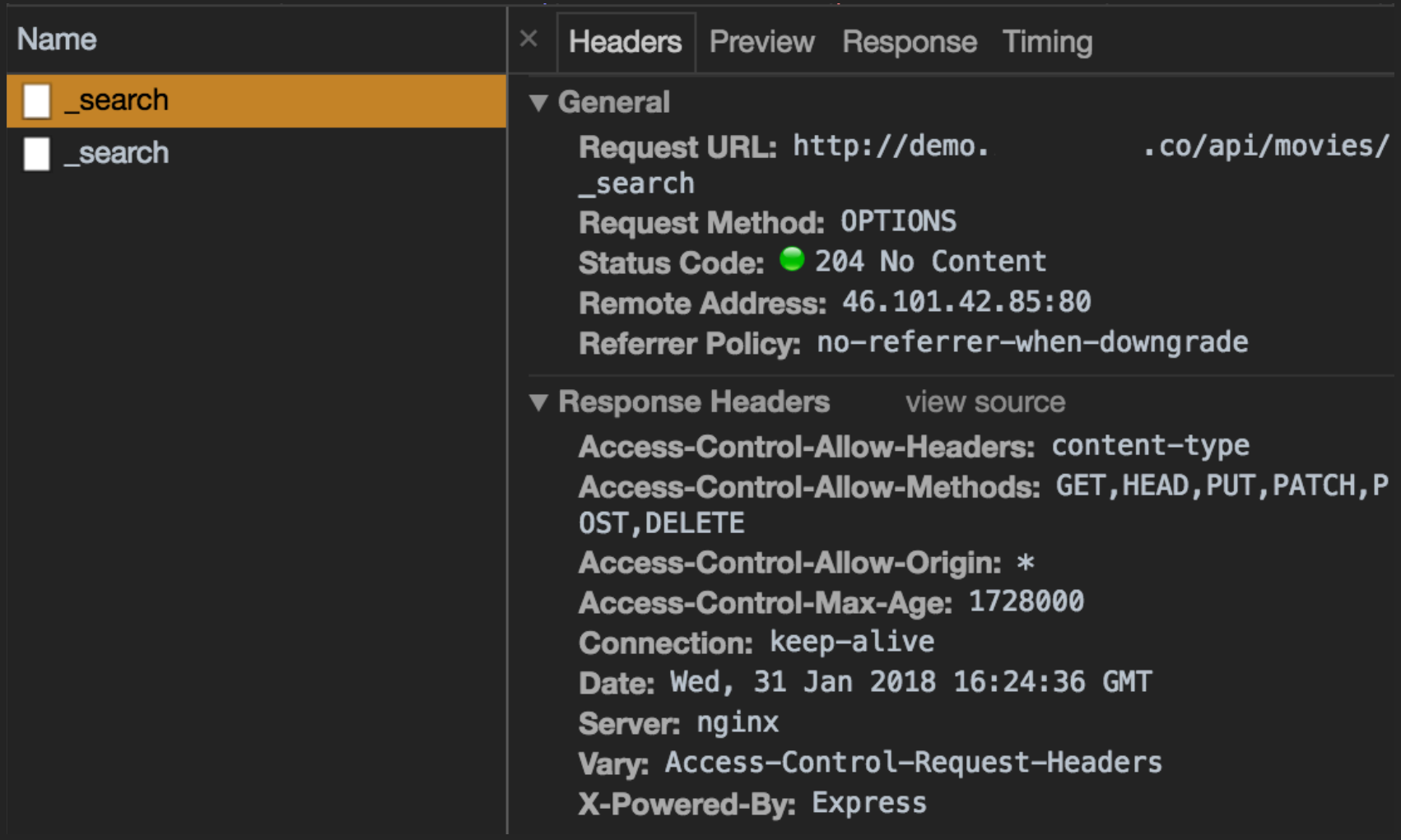
Know one layer of abstraction lower

All abstractions are leaky

Cross Origin Resource Sharing



Haroen Viaene - @haroenv algolia


Name
_search
_search
Headers
Preview
Response
Timing
General
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
Response Headers
view source
Access-Control-Allow-Headers: content-type
Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,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

Haroen Viaene - @haroenv

SimpleRequests

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

Specific headers

Specific methods

Specific content type

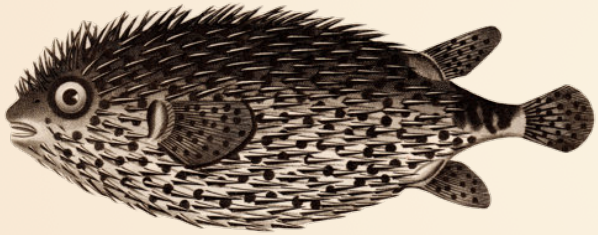
What I've learned

Design **Performance** **Maintainability**

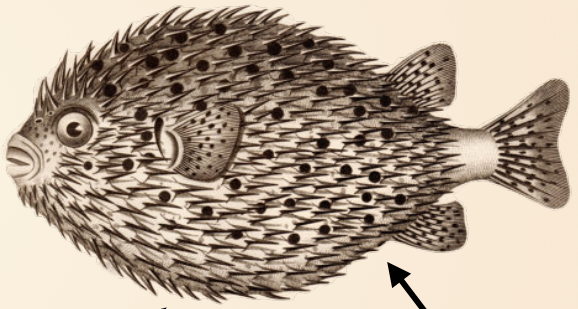
Maintainability

Change over time

Your API over time



New



Now

Encyclopaedia Londinensis or The Universal Dictionary of Arts, Sciences & Literature

Removing code!

1. Let your users know
2. Measure usage
3. Delete

1 Let your users know

```
console.warn(
`The feature XXX is now deprecated.

Instead you can use YYY

https://github.com/XXX/wiki/deprecated#XXX`
);
```

1 Let your users know

```
console.warn(
`The feature XXX is now deprecated.

Instead you can use YYY

https://alg.li/js-deprecated#XXX`
);
```

1 Let your users know

```
const deprecated = ({
  deprecated,
  replacement,
  callback,
  args
}) => {
  const anchor = deprecated.toLowerCase().replace(/[\.\(\)]/g, '');
  console.warn(
    `The feature ${deprecated} is now deprecated.

    Instead you can use ${replacement}

    https://alg.li/js-deprecated#${anchor}`
  );
  return callback(args);
};
```

Removing code!

1. Let your users know
2. Measure usage
3. Delete

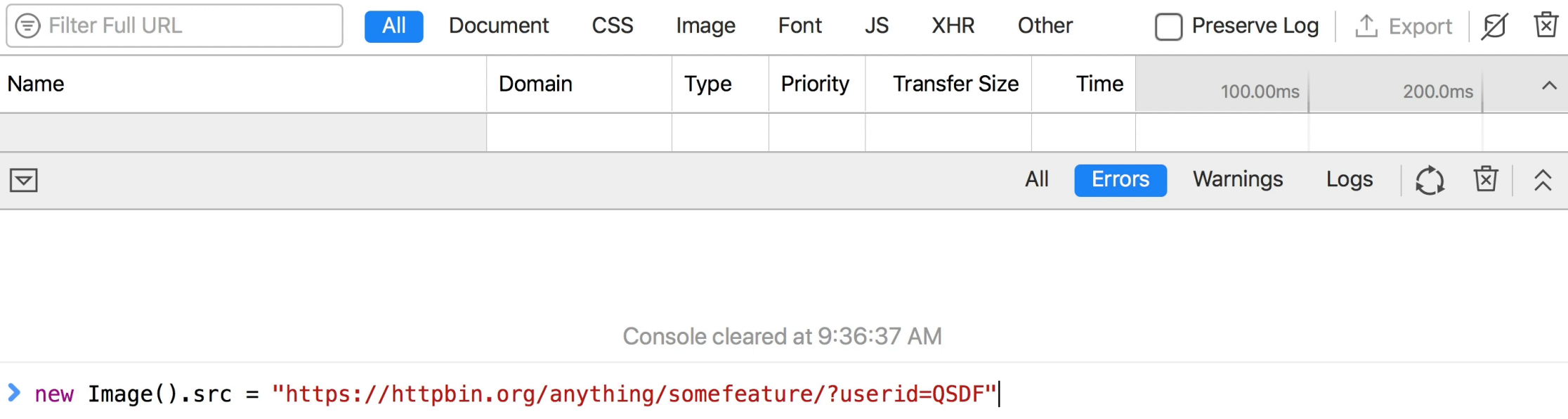
2 Measure usage

Existing analytics

```
/usage/somefeature/?userid=XFCSDS
```

```
new Image().src = '…'
```

2 Measure usage



2 Measure usage

```
app.get('/usage/:feature', function(req, res) {

  increment({
    feature: req.params.feature,
    userid: req.query.userid,
  });

  res.send('acknowledged');
});
```

2 Measure usage

```
app.get('/usage/:feature', function(req, res) {

  increment({
    feature: req.params.feature,
    userid: req.query.userid,
  });

  res.send('acknowledged');
});
```

Removing code!

1. Let your users know
2. Measure usage
3. Delete

3 Delete

Write deletable code

Modularity

Add maintainer notes

Removing code!

1. Let your users know
2. Measure usage
3. Delete

Maintainability

Testing

Testing

1. Arguments
2. Network
3. Functional

1 Arguments

Are the methods sending the correct requests?

Mock `request`

Snapshot testing

Testing

1. Arguments
2. Network
3. Functional

2 Network

Mock underlaying network layer

Test `request` as a pure function

Testing

1. Arguments
2. Network
3. Functional

3 Functional

Write as a user

Use the real API

Clean up afterwards

You can cheat

GitHub

Overview

Separation of concerns

Pick the right abstraction

Have a “client” for credentials

Know one level lower (SimpleRequests)

Know when to deviate from your server API

Deprecate with confidence

Look at other API clients

Thank you!

haroen.me/presentations

Haroen Viaene - @haroenv