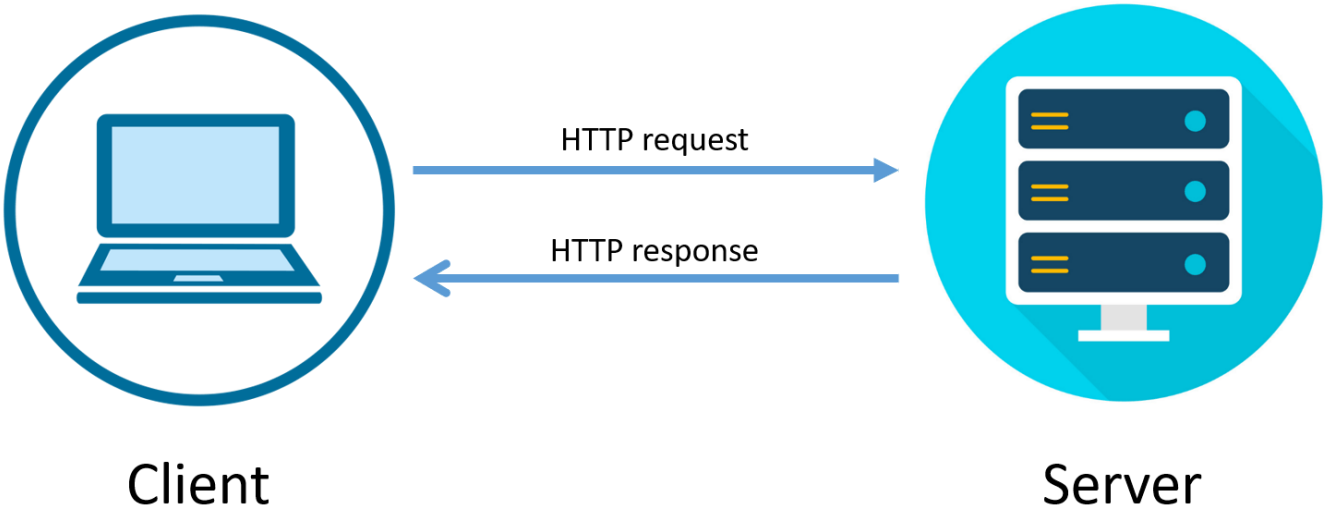
Fetch API



> Source: https://bytesofgigabytes.com/networking/how-http-request-and-response-works/

1. fetch overview
2. fetch with headers
3. Handle errors

1. fetch overview

- The Fetch API provides an interface for fetching resources source
- Web APIs -> global method **fetch()**.

fetch

- it starts the process of fetching a resource from the network
- returns a promise which fulfilled once the response is available
- it only rejects when a network error is encountered.
- it doesn't reject on HTTP errors (4xx, 5xx)
- you can check **Response.ok** to handle error properly.

```
const url = 'https://js-post-api.herokuapp.com/api/students?_page=1';
const init = {
    method: 'POST', // GET, PUT, PATCH. DELETE
    headers: {
        'Content-Type': 'application/json',
        Authorization: 'Bearer YOUR_TOKEN_HERE'
    },
    body: JSON.stringify({ name: 'Long Do' }),
}
```

```
const promise = fetch(url, init);
```

Init object

#	Name	Desc
1	method	GET, POST, PUT, PATCH, DELETE, ...
2	headers	'Content-Type', Authorization, x-*
3	body	URLSearchParams, FormData, Blob, ...
4	signal	an AbortSignal used to abort if desired(Hủy bỏ request) read more

Common Content-Type

Content-Type	Desc
application/json	JSON data (most of the time)
multipart/form-data	FormData (upload file: images, pdf, ...)
application/x-www-form-urlencoded	key-value on URL

> Source: https://developer.mozilla.org/en-US/docs/Web/API/fetch#parameters

```
fetch('https://js-post-api.herokuapp.com/api/students?_page=1')
    .then(response => response.json())
    .then(data => console.log(data));
```

2. fetch with headers / body

- GET, DELETE: mình không dùng body.
- POST/PUT/PATCH: mình có gửi kèm body

```
// Add new student
fetch('https://js-post-api.herokuapp.com/api/students', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        name: 'Long Do',
        age: 18,
        mark: 9,
        gender: 'male',
    })
}).then(...).catch(...)
```

```
// Update partial info of student
fetch('https://js-post-api.herokuapp.com/api/students/e0df2354-1014-4f40-97c4-
76e1dac88ab5', {
    method: 'PATCH',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        name: 'Long',
        age: 20,
    })
}).then(...).catch(...)
```

```
// Delete a student
fetch('https://js-post-api.herokuapp.com/api/students/e0df2354-1014-4f40-97c4-
76e1dac88ab5', {
    method: 'DELETE',
    headers: {
        'Content-Type': 'application/json',
    },
}).then(...).catch(...)
```

> Source: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

3. Handle errors

> Lưu ý: khi gọi 1 đường dẫn bất kỳ mà lỗi (4xx or 5xx) thì nó không nhảy vô catch (lỗi) và nhảy thẳng vô then, nên nhiệu vụ của mình phải handle lỗi bằng cách dùng `response.ok` để check.

```
fetch('https://js-post-api.herokuapp.com/api/invalid-endpoint', {
    method: 'GET',
    headers: {
        'Content-Type': 'application/json',
    },
})
.then(response => {
    if (response.ok) return response.json();
    // TODO: How you handle errors here? --> It depends on what your API
    returns
    // Solution 1: return Promise.reject(new Error('Something wrong!'));
    // Solution 2: throw new Error(response.statusText);
    // sau khi throw error thì nó sẽ nhảy lỗi vô catch.
    return response.json().then(data => {
        throw new Error(data?.message || 'Something went wrong!');
    })
 })
.catch(error => {
    console.log(error);
```

```
    // Toast message
    // Send report to log server (Sentry)
    })
```

Response status codes

Status Code	Name
1xx	Informational responses
2xx	Successful responses
3xx	Redirects
4xx	Client errors
5xx	Server errors

Common status code

Status	code Name
200	OK
201	Created
301	Moved Permanently (chuyển hướng)
400	Bad Request (Gửi thông tin sai)
401	Unauthorized
403	Forbidden (không có quyền truy cập)
404	Not Found
429	Too Many Requests (gọi quá nhiều request trong 1 time cho phép)
500	Internal Server Error (Lỗi server)
502	Bad Gateway (Không truy cập được server)

> Source: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Tóm lại

- API là chuẩn giao tiếp giữa 2 hệ thống với nhau.
- HTTP API là chuẩn giao tiếp thông qua HTTP để 2 hệ thống có thể nói chuyện với nhau.
- HTTPS giúp mình bảo mật gói tin trên đường truyền internet, trong khi HTTP thì sẽ được truyền đi dạng plain/text. Dễ bị tấn công middleman.
- REST API là chuẩn giao tiếp client-server (cùng một số đặc điểm khác) giúp mình có được một số quy tắc nhất định trong việc quy định về resource, method để client và server có thể "nói chuyện" được với

nhau.

Về nguyên tắc đơn giản,

- Client: gửi một HTTP Request lên server
- Server: sau khi nhận được request sẽ xử lý và trả về một HTTP Response