A Dive into CORS

Aprofundando nas políticas de Cors e preflight

Olá!

Eu sou a **Ana Coimbra**
Dev Lead @ Kobe

#Android #Web #IxD #Firebase

@anacoimbrag

Quem já se deparou com esse problema?

Access to fetch at 'http://localhost:8080/' from origin 'http://localhost:3000' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

TypeError: Origin http://localhost:3000 is not allowed by Access-Control-Allow-Origin.

E como vocês resolveram?

All you need to do is add an HTTP header to the server:

```
Access-Control-Allow-Origin: http://localhost:3000
```

Or, for simplicity:

```
Access-Control-Allow-Origin: *
```

In Windows, paste this command in **run** window

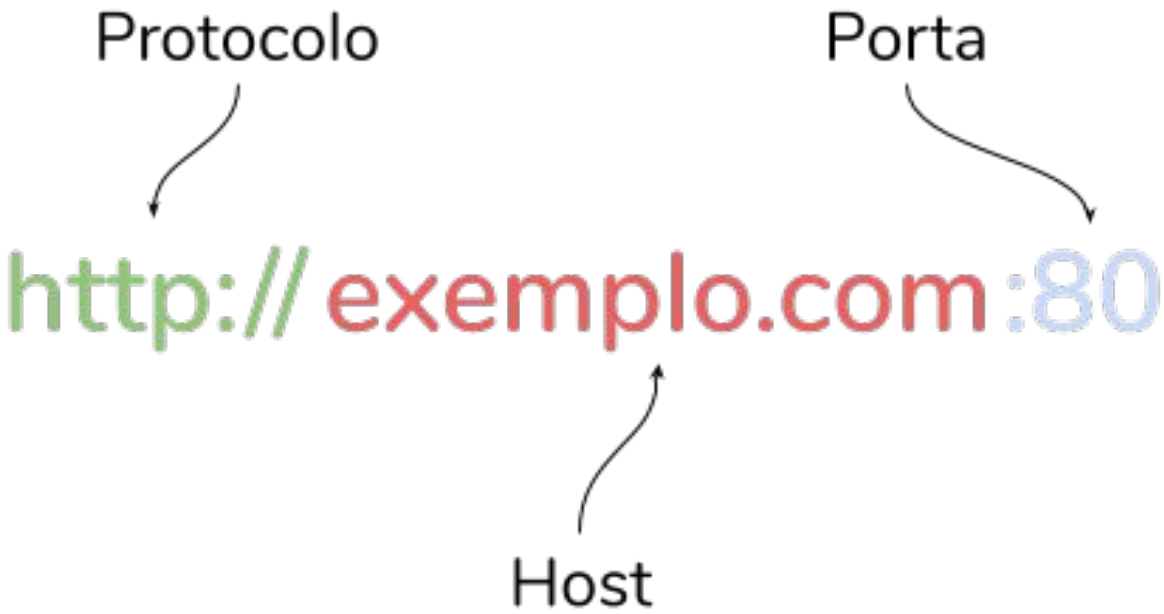
```
chrome.exe --user-data-dir="C:/Chrome dev session" --disable-web-security
```

this will open a new **chrome** browser which allow access to no **'access-control-allow-origin'** header request.

Cross - **O**rigin **R**esource **S**haring

Mecanismo dos navegadores que gerencia o compartilhamento de recursos entre diferentes origens.

O que é uma origem?



same-origin

A: Oi Vizinho! Estou precisando de uma mangueira, você tem aí pra me emprestar?

B: Claro! Pode descer aqui para buscar

cross-origin

A: Oi! Estou te ligando para te pedir emprestado uma mangueira, você tem?

C: Claro! Mas preciso autorizar a sua entrada na portaria do prédio quando vier pegar.

1. Requisições simples

Requisições que não exigem uma pré-autorização

Requisições simples

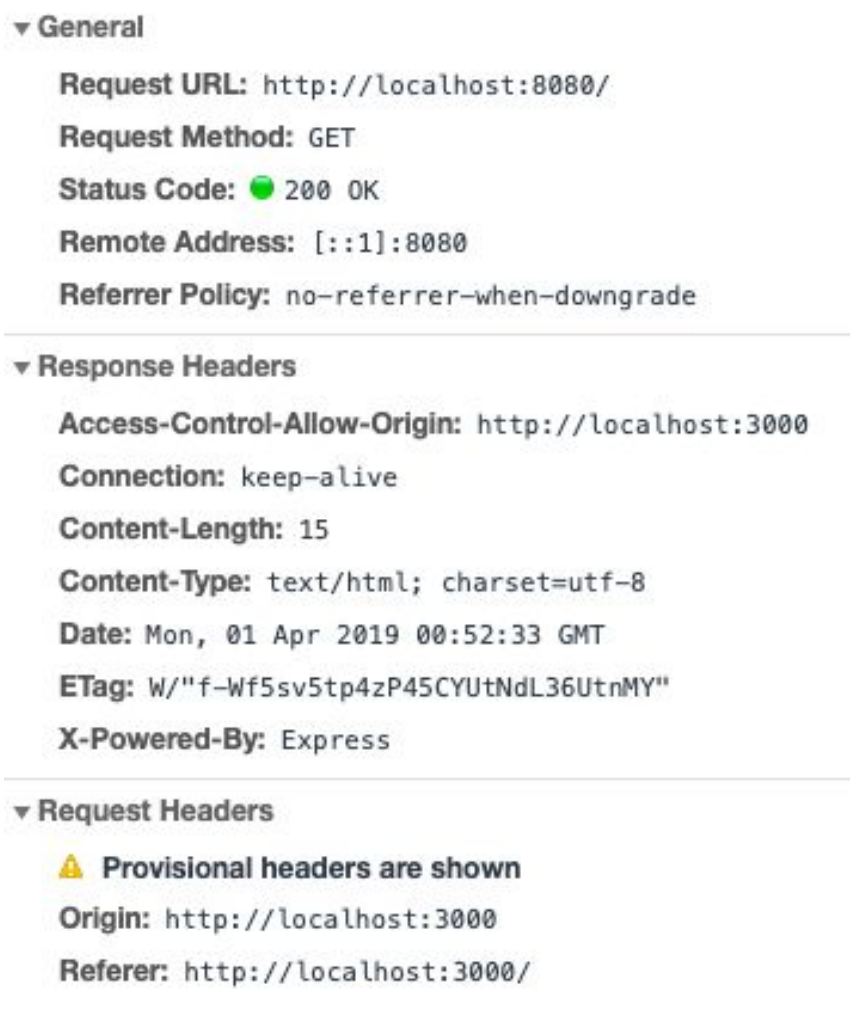
Requisições que não exigem uma **pré-autorização**

- *Métodos permitidos:* ***GET, HEAD, POST****
- *Cabeçalhos permitidos:* ***Accept, Accept-Language, Content-Language, Content-Type*, DPR, Downlink, Save-Data, Viewport-Width, Width***
- *Valores permitidos para Content-Type:* ***application/x-www-form-urlencoded, multipart/form-data, text/plain***

```
fetch('http://localhost:8080', {
    method: 'GET'
})
```

```
app.get("/", (req, res) => {
  res.setHeader("Access-Control-Allow-Origin", "http://localhost:3000");
  res.send("Hello Cors Test");
});
```

Requisições simples


General
Request URL: http://localhost:8080/
Request Method: GET
Status Code: 200 OK
Remote Address: [::1]:8080
Referrer Policy: no-referrer-when-downgrade
Response Headers
Access-Control-Allow-Origin: http://localhost:3000
Connection: keep-alive
Content-Length: 15
Content-Type: text/html; charset=utf-8
Date: Mon, 01 Apr 2019 00:52:33 GMT
ETag: W/"f-Wf5sv5tp4zP45CYUtNdL36UtnMY"
X-Powered-By: Express
Request Headers
Provisional headers are shown
Origin: http://localhost:3000
Referer: http://localhost:3000/

2. Requisições com Preflight

Requisições que exigem uma pré-autorização

Requisições com preflight

Requisições que exigem uma **pré-autorização**

- *Tudo o que não se encaixa nas requisições simples*

Quando uma origem requisita algo que pode causar um efeito colateral na outra origem, é necessário uma pré-requisição que vai autorizar que essa chamada seja efetivada.

A: Eu sou o **Servidor A** e gostaria de fazer a ação **PUT** no seu recurso **BD**, estou autorizado?

B: **Servidor A** autorizado a fazer **PUT** no recurso **BD**.

A: Então, realize a ação **PUT** no **BD** com as informações **XYZ**

B: Ação realizada com **sucesso**.

preflight

```
fetch('http://localhost:8080', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: { key: 'value' }
})
```

```
app.options("/", (req, res) => {
  res.setHeader("Access-Control-Allow-Origin", "http://localhost:3000");
  res.setHeader("Access-Control-Allow-Headers", "Content-Type");
  res.send(true);
});

app.post("/", (req, res) => {
  res.setHeader("Access-Control-Allow-Origin", "http://localhost:3000");
  res.send("POSTED on root");
});
```

Requisições com preflight

▾ General

Request URL: http://localhost:8080/

Request Method: OPTIONS

Status Code: 200 OK

Remote Address: [::1]:8080

Referrer Policy: no-referrer-when-downgrade

▾ Response Headers

Access-Control-Allow-Headers: Content-Type

Access-Control-Allow-Origin: http://localhost:3000

Connection: keep-alive

Content-Length: 4

Content-Type: application/json; charset=utf-8

Date: Mon, 01 Apr 2019 00:29:51 GMT

ETag: W/"4-X/5TO4MPCKAyY0ipFgr6/IraRNs"

X-Powered-By: Express

▾ Request Headers

⚠ **Provisional headers are shown**

Access-Control-Request-Headers: content-type

Access-Control-Request-Method: POST

Origin: http://localhost:3000

Referer: http://localhost:3000/

▾ General

Request URL: http://localhost:8080/

Request Method: POST

Status Code: 200 OK

Remote Address: [::1]:8080

Referrer Policy: no-referrer-when-downgrade

▾ Response Headers

Connection: keep-alive

Content-Length: 14

Content-Type: text/html; charset=utf-8

Date: Mon, 01 Apr 2019 01:27:58 GMT

ETag: W/"e-aryBEvDCivTbVqo2qYWW6D8uX4o"

X-Powered-By: Express

▾ Request Headers

⚠ **Provisional headers are shown**

Content-Type: application/json

Origin: http://localhost:3000

Referer: http://localhost:3000/

```
app.use((req, res, next) => {
  res.setHeader("Access-Control-Allow-Origin", "http://localhost:3000");
  res.setHeader("Access-Control-Allow-Headers", "Content-Type");
  next();
});
```

Usando middleware

Podemos resolver isso no Front-end?

PROXY

Serviços de proxy que autorizam seus requests independente de origem, método, cabeçalho, etc.

JSONP

Uma forma de passar pelo CORS mas que funciona de uma forma "gambiarra" fazendo as requisições via DOM.

Chrome Dev Tools

É possível abrir o chrome com as políticas de segurança desabilitada ou por comando no terminal ou via plugin.

É importante lembrar que o **CORS** é um mecanismo de segurança e se não tratarmos corretamente, esse princípio não é atingido.

Obrigada!

Dúvidas? Observações? Sugestões?

oi@anacoimbra.dev

http://bit.ly/cross-origin

https://github.com/anacoimbrag/cors-preflight

anacoimbra.dev