WHAT IS CORS?

Cross-Origin Resource Sharing

CORS is a browser policy about allowing JS-based service calls to endpoints that are on a different domain/port than the currently loaded page.

This is done for security reasons.

BEFORE CORS: WILD WEST

To understand why we have CORS, you have to understand life BEFORE CORS.

1st try: browser JS can do anything, anywhere

- Security problems, particularly with cookies (What if I call a service on your bank site from my cat videos webpage?)

BEFORE CORS: SAME ORIGIN POLICY

2nd try: Same Origin Policy (SOP)

- Pages can only load resources from the same "origin" (protocol + domain + port)
- Exceptions for loading images, JS files, and CSS files

Secure, but people WANTED Cross-Origin resources

- Including their own subdomains (e.g. **http://example.com** and **http://api.example.com**)
- Workarounds included JSONP (hiding a service call as a JS file to load and run)
  - Which is NOT secure

ADOPTING CORS

3rd Try: CORS

- response headers say what the service allows
- browser refuses to give data to JS if not allowed
- ENFORCED BY BROWSER

In addition, requests that aren't "simple" send a "preflight" request

- An OPTIONS (http method) request to check the response headers before any data is sent
- Browser automatically sends before a non-simple CORS request

TRIGGERING CORS

Simply load a page, then run some JS that makes a `fetch()` call to a different origin.

```
$ serve public/
```

In browser `Devtools > Console`:

```
fetch('http://example.com/api/');
```

MISLEADING CORS MESSAGE

```
Access to fetch at 'http://example.com/api/' from origin 'http://127.0.0.1:9000' has
```

I hate this message.

- `no-cors` is almost NEVER helpful - it will NOT let you see the response.
- The error is because the server didn't have the correct CORS headers
- Often the issue is that the service returned an error, and didn't return CORS headers on the error response
  - So CORS is not the source of the error

WHAT ABOUT CORB?

A related browser-enforced security block is CORB. This blocks a resource if it appears to be the wrong kind. (Example, you request an image, but the server returns HTML).

- As with CORS, you are not allowed to see the response
- This might be a server-side issue
- Or you might need to make it clear what you expect with an 'accepts' header
- It can show up on sites that give a 404 PAGE (html) to any 404 requests (data)

CORS WORKAROUNDS

What if CORS is blocking you?

Don't try to get around it - it's a security feature, so any workaround will be fixed.

Options:

- Fix the server side
- Have a backend proxy (you talk to service that you can talk to, it makes the cross-origin request and feeds you the data)

CORS TAKEAWAYS

Understand the following:

- CORS is enforced by the browser
- It exists for good security reasons
- "Fixes/workarounds" should be done on the server
- CORS error messages can be misleading - always make sure you know what problem you are solving

CORS AND CREATE REACT APP

Developing a CRA app often has CORS issues

- dev server is one url:port
- service service is a different url:port

If the prod server will be the SAME url:port

- need a solution during dev

CRA PROXY

1. Create your service + static server on port AAAA
2. Run your dev server on port BBBB
3. add a `proxy` line in your `package.json`
    - points to domain:AAAA of service server
4. Calls to dev server that don't exist there will go to proxy instead

Browser never talks directly to service server

- no browser = no CORS

When you build static files, everything will be on service + static server

- Same domain:port = no CORS