Fix two: send your request to a proxy

You can't ask your users to trick their browsers by installing a plugin that applies an header in the frontend. But you can control the backend address that the web app's API requests are going to.

The cors-anywhere server is a proxy that adds CORS headers to a request. A proxy acts as an intermediary between a client and server. In this case, the cors-anywhere proxy server operates in between the frontend web app making the request, and the server that responds with data. Similar to the Allow-control-allow-origin plugin, it adds the more open `Access-Control-Allow-Origin: *` header to the response.

It works like this. Say your frontend is trying to make a GET request to:

`https://joke-api-strict-cors.appspot.com/jokes/random`

But this api does not have a `Access-Control-Allow-Origin` value in place that permits the web application domain to access it. So instead, send your GET request to:

`https://cors-anywhere.herokuapp.com/https://joke-api-strict-cors.appspot.com/jokes/random`

The proxy server receives the `https://joke-api-strict-cors.appspot.com/jokes/random` from the url above. Then it makes the request to get that server's response. And finally, the proxy applies the `Access-Control-Allow-Origin: *` to that original response.

This solution is great because it works in both development and production. In summary, you're taking advantage of the fact that the same origin policy is only implemented in browser-to-server communication. Which means it doesn't have to be enforced in server-to-server communication!

The one downside of the cors-anywhere proxy is that can often take a while to receive a response. The latency is high enough to make your applications appear a bit sluggish.