DEV ACADEMY
TE KURA HANGARAU
O AOTEAROA

Consuming External APIs

Agenda for Today

- APIs on the web
- Fetching from an endpoint (Pokémon)
- Fetching from secured APIs (TMDB)
- Using a proxy server
- CORS and SOP (Affirmations)

APIs on the web

There are many APIs available on the web for us to use.

Some points to remember:

- Other people have collated and shared data
- You'll need to read the docs!
- Some need an API key
- Only some enable CORS (more on this later)

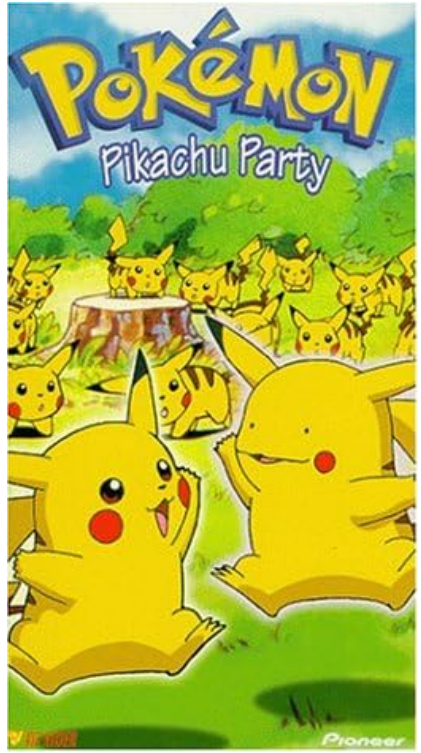
Fetching from an endpoint

Demo

Our target API: https://pokeapi.co/

Today's checklist:

- Find the API and read the docs
- Test the endpoints with Insomnia
- Write a frontend API client function to consume the data
- Use QuickType to generate typescript types
- Enjoy the data⚡



Fetching from secured APIs

Demo

Our target API: https://www.themoviedb.org/

Checklist:

- Find the API and read the docs
- Sign up for an API key
  - (TMDB has manual verification, other APIs are instant)
- Write the API client and server proxy routes
- Generate types with QuickType
- Protect the API token with dotenv
- See some movies! 🎞

Using headers with superagent:

```
await request.get('API_ENDPOINT_HERE')
.set('accept', 'application/json')
.set('Authorization', 'YOUR_TOKEN_HERE')
```

Installing dotenv:

```
npm install dotenv --save
```

.env file:

```
MOVIEDB_API_TOKEN = "Secret Token"
```

dotenv in the server:

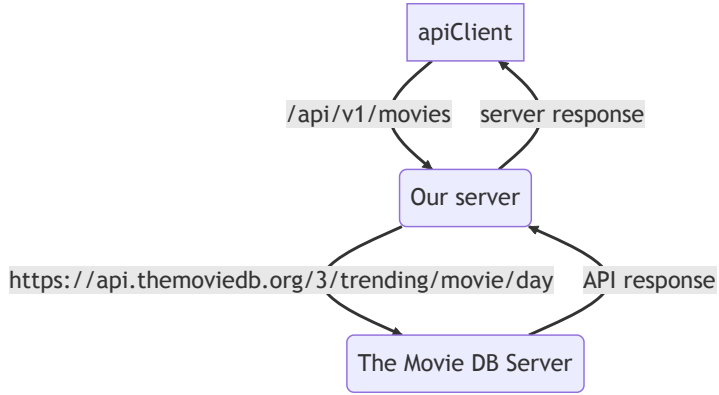
```
import 'dotenv/config'
console.log(process.env.MOVIEDB_API_TOKEN)
```

Using a proxy server

We can use a proxy on the server to send requests

This helps us with a few things:

- Our authorisation tokens aren't shown on the browser
- We can use environment variables
- We can bypass CORS restrictions



The steps:

- Make a new server route e.g. `` `/ap1/v1/movies` ``
- Make a request to the external API from the new route
- Return the data in the shape we want it
- Make a request from the API client to our server route

"It is not a sprint, it is a marathon.
One step at a time"

-`affirmations.dev `

SOP and CORS

Demo

Our target API: https://www.affirmations.dev/

SOP or same-origin policy is a protection measure for JavaScript in browsers:

- Dynamic requests must can only be sent from URLs that match the origin
- To have the same origin protocol, domain and port must match

E.g. for `` `http://localhost:3000` ``:

`

URL	SOP	Why?
`http://localhost:3000/puppies`	Yes	Same protocol, host and port
`https://localhost:3000/puppies`	No	Different protocol
`https://localhost:5173/puppies`	No	Different port
`http://google.com/search`	No	Different domain

SOP and CORS (continued) Demo

Our target API: https://www.affirmations.dev/

SOP is quite restrictive, so APIs allow access through CORS (Cross-origin resource sharing)

- Some open APIs like https://pokeapi.co/ have a special CORS header set:
  - `Access-Control-Allow-Origin` (allows any origin to connect)
- Without that access, we can't visit secured APIs from the browser
  - `blocked by CORS policy: No 'Access-Control-Allow-Origin' header`
- In that case we need to make requests using our server as a proxy

A nice breakdown here

Useful tips

- We can provide a generic to useState to specify custom types (read more)

```
const [pokemon, setPokemon] = useState<Pokemon | null>(null)
```

- If your types have the same name as your component you have two options:
  - Declare it as a `type` when importing
  - Rename the type with the `as` keyword
- Check the API's docs to see how you should send your API key or token
- Don't push your API keys to GitHub!
  - Instead store your API Key in a `.env` and use it on your server (dotenv)
- If the API is secured, use a proxy route on your server to connect

Demo