React and Services

With or without React:

- Your static files may be on one server
- Your services may be on a different server

Our setup

Development using two servers:

- the webpack-dev-server from react-scripts
- a server.js we run separately

Final product using one server:

- `npm run build` creates static files
- our server will serve those static files

Real world could stay two separate servers!

Dev setup

- Create a `server.js` for services
- Add `scripts` entry in `package.json`
- Add `proxy` entry in `package.json` for CORS

server.js

```
const express = require('express');
const app = express();
const PORT = 5000; // Why 5000?  dev-server uses 3000

// Not yet configured for static files!

app.get('/api/test', (req, res) => {
  res.send('server works');
});

app.listen(PORT, () => {
  console.log(`http://localhost:${PORT}/`);
});
```

package.json scripts entry

I **choose** to use same package.json:

```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject",
  "dev": "npx nodemon server.js"
},
```

- Why `dev`?
    - Just a name, so make it meaningful
- Remember to install express
- Remember to install nodemon
    - as dev dependency

Trying it out

- Terminal 1: `npm start`
- Terminal 2: `npm run dev`

Once everything is running:

- browser console:
  - `fetch('/api/test');`
  - inspect response in Network

Hey! That is the React page!

We requested a relative url

- so it hit this server
- which is only serving this one React page
    - regardless of path
- our /api/test is on a **different port**
- browser console:
    - `fetch('http://localhost:5000/api/test');`

CORS error

A CORS error makes sense

- We are requesting a different **origin**
- Server is not applying CORS headers to response

Two solutions:

- use a proxy -OR-
- set the CORS header

CORS Thinking

Which is better choice this time?

- use a proxy -OR-
- set the CORS header

We will use a proxy:

- CRA makes it easy
- We will not be on a different server in production

CORS headers make more sense if our services will always be on a separate server from our static files

Proxy

CRA provides a proxy ability:

- add a `proxy` line in `package.json`

```
"proxy": "http://localhost:5000",
```

Any request that doesn't match a file on CRA webpack-dev-server will go to proxy server

- restart `npm start`

Try it again

```
fetch('http://localhost:5000/api/test');
```

- Still errors
- But we aren't using the proxy!

```
fetch('/api/test');
```

- This works!
- With correct response!

Proxy summary

Notice what happens:

- browser requests /api/test
    - What port? What is full url of request?
- dev-server makes request to actual server
    - What port? What is full url of request?
- dev-server sends response to browser
- browser gets the response to original request
    - Doesn't know proxy was involved!

React can call services

More on this later

- but pure JS can call fetch
- proxy lets us handle two servers

Production setup

- Run `npm run build` to create static files
- Modify `server.js` to handle static files
- Add entry in `package.json` to run server

Creating static files

`npm run build` creates `build/` directory

- This is like previous `public/` directory
- CRA uses `public/` as INPUT, not OUTPUT

All `.js` and `.jsx` are **transpiled** and **bundled**

- Into static files to run in browser
- No static file runs on server

Serving static files

We have already done this before

- Just a different directory

`server.js`:

```
app.use(express.static('./build'));
```

Modifying package.json

`package.json`:

```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject",
  "dev": "npx nodemon server.js",
  "serve": "node server.js"
},
```

- Why not `nodemon`?
- Why not `npx`?
- Could replace `start`!
    - Is package for running or developing?
    - Move current `start` to a new name

Try it!

Remember which port the server uses!

What happens with proxy?

- Nothing is using proxy info
    - Now running only your `server.js`
        - Which doesn't try to proxy
- browser requests relative urls
    - Such as `/api/test`
    - And server has them!
- We are on different port than before
    - but relative urls didn't care

Proxy and Services Summary

- **Two** servers to develop, **One** server for production
    - Not the only way to do it!
- We are using one `package.json` for both
    - Also not the only way to do it!
- CRA uses `proxy` in `package.json` for dev-server
    - This lets us use relative urls
    - as long as relative urls work for prod
- React all becomes static JS files
    - static files on server, run in browser