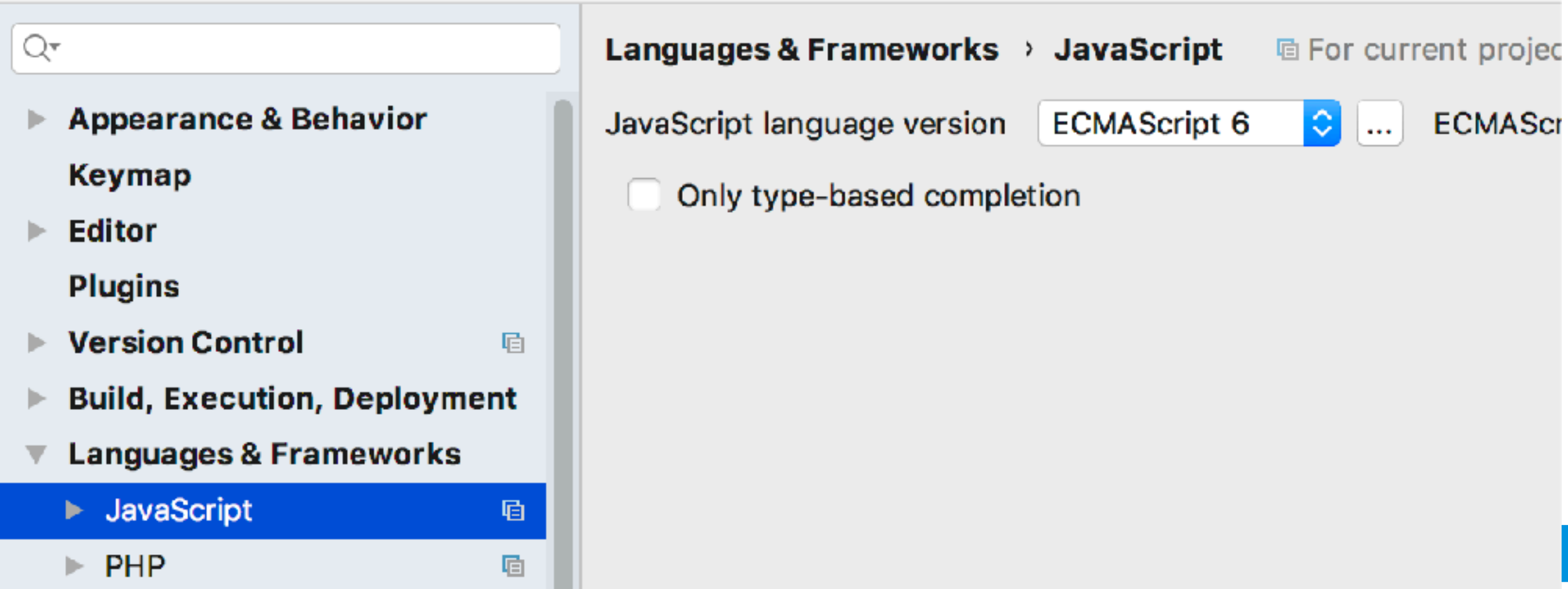
Lab Class Week 4: Progressive Web Apps

Prof. Fabio Ciravegna
University of Sheffield
f.ciravegna@shef.ac.uk

class not found

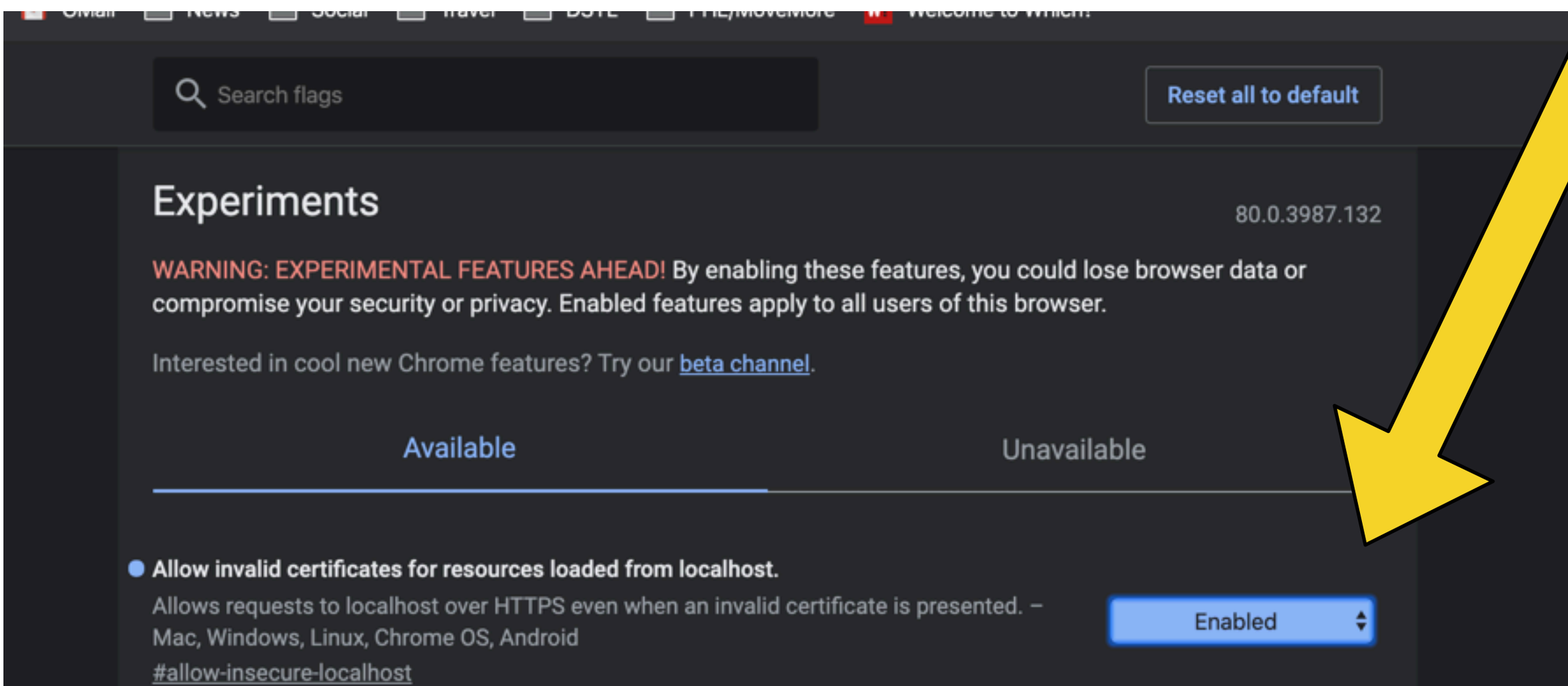
- in routes/index.js?
- you must use a different version of Javascript
- go to Intellij > (file >) preferences



Before we start

- In order to work on the Lab computers you MUST follow these steps and always use only that version of Chrome with https
- Open Chrome
- type chrome://flags/ in the address bar

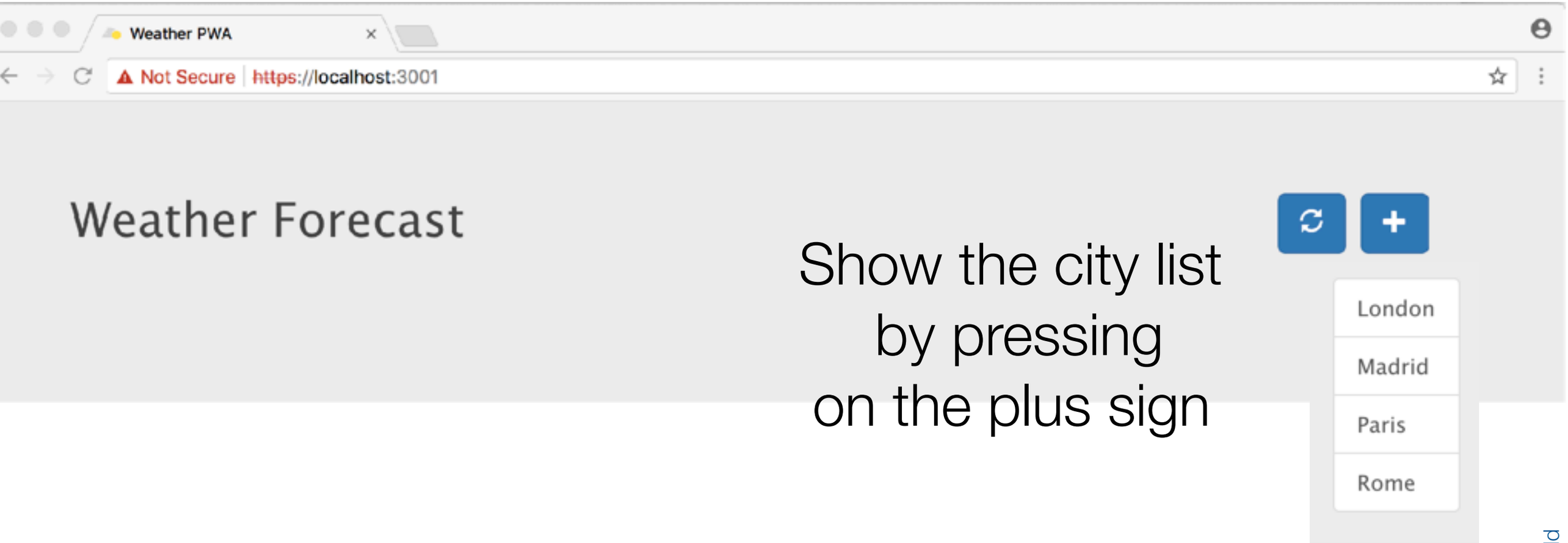
enable



Exercises

- You are given a simple progressive web app which retrieves weather forecasts from a server
  - The app enables selecting the cities to get forecasts for
  - It retrieves forecasts from a nodes server using Ajax
    - to simplify the forecasts are randomly generated on the server
  - it stores the retrieved data into localStorage
  - If the device is offline, it shows the data stored in localStorage

The University Of Sheffield.



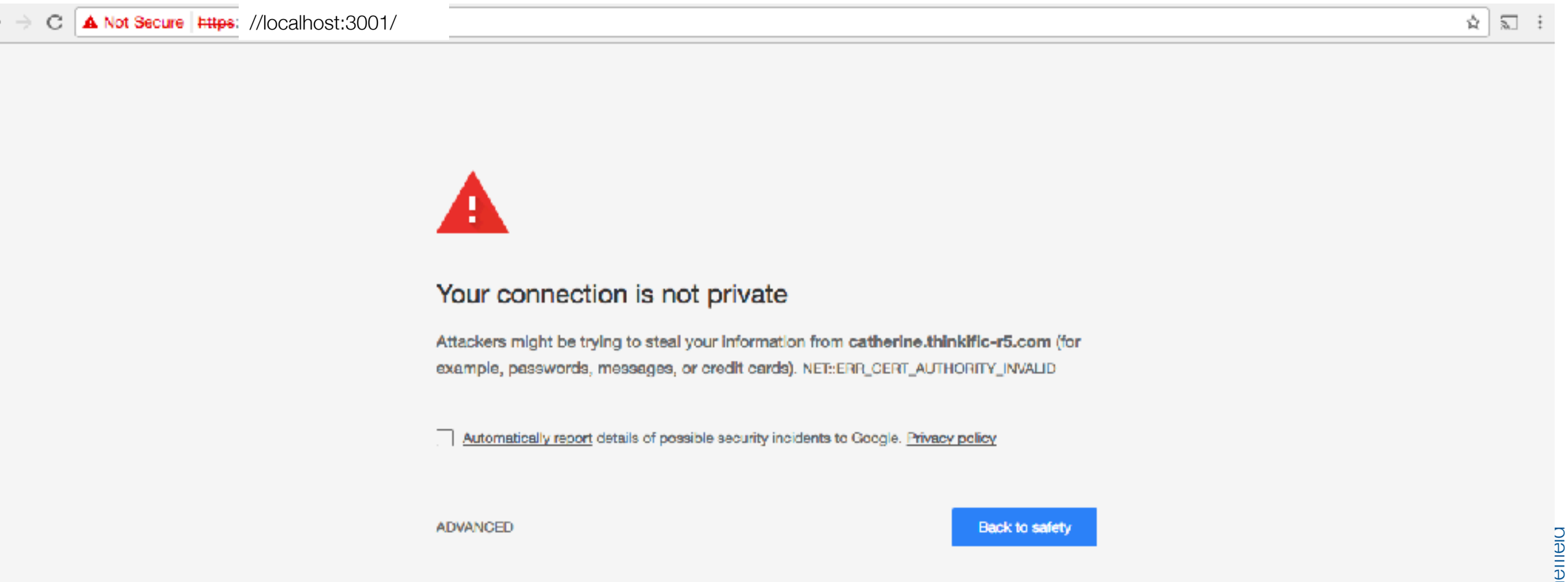
Show the city list by pressing on the plus sign

Note!!!! the ur is https://localhost:3001

The University Of Sheffield.

remember to use the set the flag in Chrome

If you get this



- You are just using standard Chrome
  - rather than the chrome with the appropriate flags

Inspect the app

- public
  - fonts
  - images
  - scripts
    - app.js
    - bootstrap.min.js
    - database.js
    - jquery.min.js
  - styles
  - favicon.ico
  - manifest.json
  - service-worker.js
- routes
  - index.js
  - users.js
- views
  - error.ejs
  - index.ejs
- app.js

communication with the server via Ajax

it stores the data retrieved into localStorage

the manifest file for the progressive app

the service worker

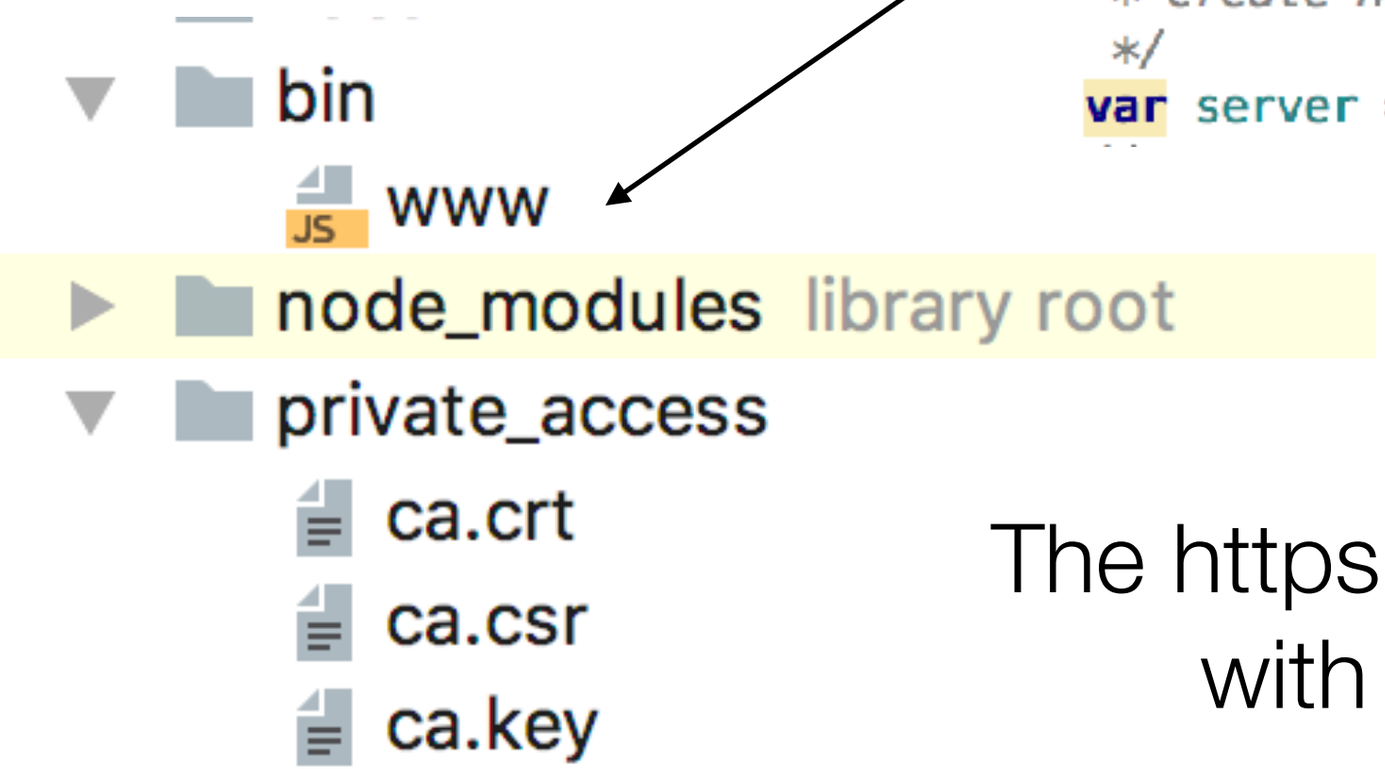
The main file: it loads the JS files

https

this is what you would normally do but on the lab computers you should use http as we did last week

```
/**
 * Create HTTPs server.
 */
var options = {
    key: fs.readFileSync('./private_access/ca.key'),
    cert: fs.readFileSync('./private_access/ca.crt')
};

/**
 * Create HTTPs server using the options
 */
var server = https.createServer(options, app);
```



The https options generated with the certificate

The exercises

- Goal of the exercise is to understand hands on how a service worker work and how it is possible to implement a PWA
- You will be asked to control the interplay between
  - requests made to the server
  - requests dealt with by the service worker
- You will be given a simple PWA and will have to modify it

It fetches updated values

it adds a new city

forecast

temperature

wind

precipitations

Weather Forecast

London	Overcast	16	67	6
Rome	Overcast	9	50	6

Exercise 1

- Inspect the app carefully and make sure to understand its parts
- Debug
  - the service worker (using Chrome developers tools)
  - the client's Ajax call (using Chrome developers tools)
  - the server (using IntelliJ)
- You must make sure to understand how it works
- Check the app both online and offline
  - see lecture slides

Modify the server

- So to return also the humidity percentage
  - i.e. modify the following class in routes/index.js

```
class WeatherForecast{
  constructor (location, date, forecast, temperature, wind,
      precipitations) {
    this.location= location;
    this.date= date,
    this.forecast=forecast;
    this.temperature= temperature;
    this.wind= wind;
    this.precipitations= precipitations;
  }
}
```

- then modify the client functions that retrieve the data via Ajax
  - i.e. modify the file public/scripts/apps.js
- finally, show the humidity on the web page
- remember to invalidate the cache for js files using Chrome developers
  - otherwise there will be no visible change

Invalidating the cache

- Either
  - delete the file from the cache (using Chrome developers tools — see lecture slides)
  - OR
  - delete all the cached data and reload the service worker
    - remember
      - to tick the box on Chrome dev tools
      - close all tabs open on your website

Exercise 2

- Add a new route to the server so that the client can get an additional view
  - e.g. to get the current date
    - router.get ('/get_date', ...
    - which will display a simple page with the date
- you will have:
  - to add the route in routes/index.js
  - create a new Ajax call for the date
  - modify the service worker so that it fetches the route /get_date
    - hint: check the way we currently request

```
var dataUrl = '/weather_data';
//if the request is '/weather_data', post to the server
if (e.request.url.indexOf(dataUrl) > -1) {
```

The University Of Sheffield.

- remember to invalidate the service worker as well and to reload everything

Exercise 3

very difficult

- Refresh the pages each and every time you return one from the cache
- Method:
  - get the cached response
  - if it does not exists fetch from network
  - if it exists
    - create a promise fetching from the network
      - that will store the result into the cache

```
cache.put(event.request, networkResponse.clone());
```

    - launch the promise
    - return the cached result

- why would this work?
- because the promise is asynchronous so the cached result will be returned immediately and when the promise is fulfilled the cache is updated
- in the meantime the user will have received the cached result
  - so it is very efficient
- Solution in the next page

The University Of Sheffield.

```
/*
   * The app is asking for app shell files. In this scenario the app uses the
   * "Cache, then if network available, it will refresh the cache
   * see stale-while-revalidate at
   * https://jakearchibald.com/2014/offline-cookbook/#on-activate
   */
  event.respondWith(async function () {
      const cache = await caches.open('mysite-dynamic');
      const cachedResponse = await cache.match(event.request);
      const networkResponsePromise = fetch(event.request);

      event.waitUntil(async function () {
          const networkResponse = await networkResponsePromise;
          await cache.put(event.request, networkResponse.clone());
      }());

      // Returned the cached response if we have one, otherwise return
      // the network response.
      return cachedResponse || networkResponsePromise;
  }());
}
```