PRACTICAL LESSON PLAN

Subject:	Web Applications Development	**Degree:**	Informatics Engineering
Topic:	Data brokering II	**Professor:**	Noélia Correia (ncorreia@ualg.pt)
Type:	☒ Programming ☐ Setup	**Institution:**	Universiy of Algarve

Goals

As mentioned in the previous lab, in web development you will often perform extraction, transformation and loading operations on data from available APIs. The goal of this new lab is to extract data from [1] that is chunked encoded by the server, so that data can be retrieved piece by piece while content is being processed. This means the HTTP response includes the `Transfer-Encoding` header line set to `chunked`. This directive is used to send data in a chunk format (see [2]). Again, you will be using TypeScript and the built-in Fetch API.

Required Resources

You need:

- ◇ Your laptop.
- ◇ Node and npm installed.

Tasks

Read Chunks

The Fetch API allows us to fetch resources across the network, and its response includes a property that is a getter exposing the body contents as a readable stream. Reading the streaming body requires attaching a reader to it, which is done using the `getReader()` method [3].
The `fs` module, together with the just mentioned readable stream, allow us to retrieve and store data in a memory-efficient way. Instead of reading all data into memory at once, we can retrieve small chunks of data (memory consumption is kept low) while storing piece by piece.
Here you must:

- ◇ Extract the data chunks from source [1];
- ◇ Attach a reader using `getReader()` method;

Use async/await when extracting data chunks, and catch any errors thrown. This simplifies the syntax necessary to consume promise-based APIs, which is the case of Fetch.

Write Data Chunks to File

Now that you have a reader attached to the readable stream, read data chunks (pieces of binary data) and store them.
Here you must:

- ◇ Read one chunk at a time using the `read()` method [3];
- ◇ Use `fs.createWriteStream()`, and write one chunk at a time to an output file.

Use `await` in both steps (single Promise per chunk and read/write operation). Include this code inside the try/catch previously built. Finally, mark every chunk when writing to the output file, and also to the console for us to follow what is happening. Write 'Done' to the console when you are done. For this you should use the events from the writable stream.

Main References

[1] `https://jigsaw.w3.org/HTTP/ChunkedScript`
[2] `https://www.geeksforgeeks.org/http-headers-transfer-encoding/`
[3] `https://developer.mozilla.org/en-US/docs/Web/API/Streams_API/Using_readable_streams`