Practical exercise 2: create and test a basic API

1. Create an API using node and express. Although we have seen that the format of multi-word names can be multiWord, multi_word, multi-word, the recommended approach is to use lower case and hyphen. The words should also be nouns, as they refer to resources, not verbs.

You should provide four endpoints to get date and time then test in the browser using localhost:3000/[endpoint]. E.g. localhost:3000/time-str. Typical responses are shown:

Path: '/time-str'

Response: "14:49:09"

Path: '/time-obj'

Response:


```
{
    "hours": 14,
    "minutes": 49,
    "seconds": 57
}
```


Path: '/date-str'

Response: Fri Aug 18 2023

Path: '/date-obj

Response:


```
{
    "dayOfWeek": 5,
    "dayOfMonth": 18,
    "month": 7,
    "year": 2023
}
```


Look up the Date object and its methods. E.g. here:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

The endpoints should be called directly from the browser, no need for front-end yet as we are creating an application server and an API. The returned json data should simply be displayed in the browser.

2. Based on the demo simpleAPI, design endpoints using appropriate path and query parameters to return jokes fom a json file I've provided. Use this as your backend data; this emulates a database until we get to the database stuff.

Endpoints should be provided for the following functional requirements:

Return all jokes

Return all jokes of a specified type, e.g. all programming

Return a fixed number of jokes of a specific type. You need to take into account that the user may ask for more jokes than there are

Return a fixed number of jokes of no specific type by calling the type "any"

The endpoints should be called directly from the browser, no need for front-end yet as we are creating an application server and an API. The returned json data should simply be displayed in the browser.

We will integrate these two practicals next week.