Tutorial 7: ExpressJS + MongoDB

Objectives

- Continue practicing with ExpressJS to complete RESTful CRUD API end-points
- Improve our code on frontend with *async/ await* instead of *Promise .then()*
- Get started with MongoDB using the command-line tool

In this tutorial, you will **work in pairs** to solve these exercises.

Activities

Activity 1: Form normal submission (15 mins)

Quick discuss these questions:

```
<form action="http://localhost:3000/words" method="GET" id="form-create">
    <label for="word">Word: </label>
    <input id="word" name="word" />

    <label for="definition">Definition: </label>
    <input id="definition" name="definition" />

    <button type="submit">Submit</button>
</form>
```

1. What is the purpose of form action?
2. What are differences between form method GET & POST???

```
function onCreate(event) {
    event.preventDefault();
    // ...
```

```
}

const formCreate = document.querySelector('#form-create');
formCreate.addEventListener('submit', onCreate);
```

- Pay attention on the url with params. Where are they from?
- How about the names? Delete one of them then re-run and observe what happen. So how to send data with a form?
- Change method to 'POST' then re-run and observe what happen. So did data send to server?

3. What is *event.preventDefault()* ???

Tutorial Exercises

Create folder `tut07/`, and two sub-folders `/nodejs & /expressjs` and complete the tasks below.

Exercise 1: ExpressJS Basic CRUD (30 mins)

Continue to complete the API end-points to CRUD words in the flashcards app.

Recall: CRUD stands for:

- [C]reate create a new word
- [R]etrieve all words
- [U]pdate a specified word
- [D]elete a specified word

Note: For updating & delete tasks, you are suggested to use the route params & message body.

Exercise 2: Async/ Await (15 mins)

Refactor code of our two functions: show all flashcards & add a flashcard in the flashcards app to use *fetch()* with *async/ await*.

Exercise 3: MongoDB command-line (20 mins)

In this exercise, we aim to create the database for our dictionary web application. Start the mongodb command-line then:

- Name all databases that your server has.
- Switch to use database with name: *eng-dict* (created automatically if db name not exist)
- Name all collections that this db has.
- Add a new word into *words* collection: *{word: 'dog', definition: 'friend'}*
- Add a new word into *words* collection: *{word: 'cat, definition: 'boss'}*
    - Query all *words* to check if success inserted.
- Add some more words (at least 5)
- Query for definition of the word: '*dog'*
- Update definition of the word '*dog'* from '*friend'* into '*woof woof*'
    - Query all *words* to check if success inserted.
- Set all words to have definition: '*empty: to-update*'
- Delete the word 'dog' from the words collection
    - Query all *words* to check if success inserted.
- Delete all words from collection *words*.
    - Query all *words* to check if success inserted.
- Delete the collection *words* from database
    - Name all collection that this db has to check if success.