# Tutorial 6: NodeJS (2)

## Objectives

In this tutorial, we will mostly practice in the client side (front-end) to consume the node server APIs we created from the previous tutorial.

- Front-end:
    - Using **fetch()** API with plain text (promise style)
    - Using **fetch()** API with JSON (promise style)
    - Getting user input via HTML form as query params

## Tutorial Exercises

In this tutorial, you will work in-pairs to build a completed dictionary application with CRUD. In IT, **CRUD** stands for Create, Retrieve, Update and Delete. In details for this application,

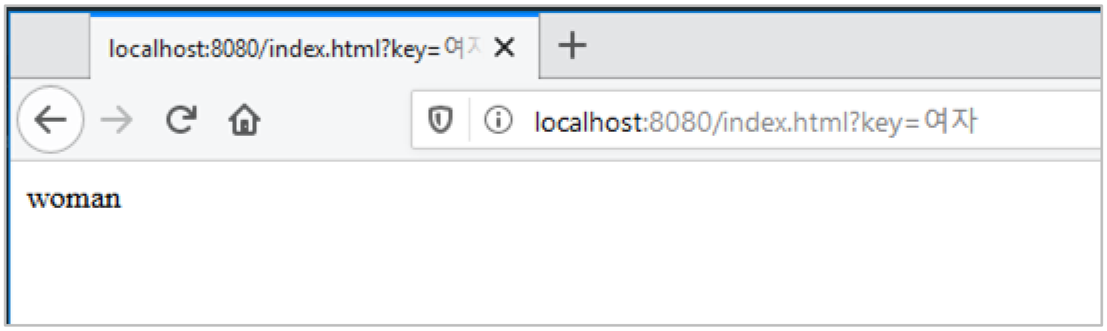
- **Create**: insert a new word-definition into the dictionary
- **Update**: update a word with modified definition
- **Retrieve**: loop up the dictionary for definition of a given word
- **Delete**: delete a word from the dictionary

Download the **starter_pack** and rename to `tut06/dictionary/`, and complete the exercises below.

### Exercise 1: [R] Dictionary – Look up (15 mins)

Recall: in the previous tutorial, you built an express server API to look up a word in the dictionary. With the query param is the word, you are required to print out the meaning of the word.

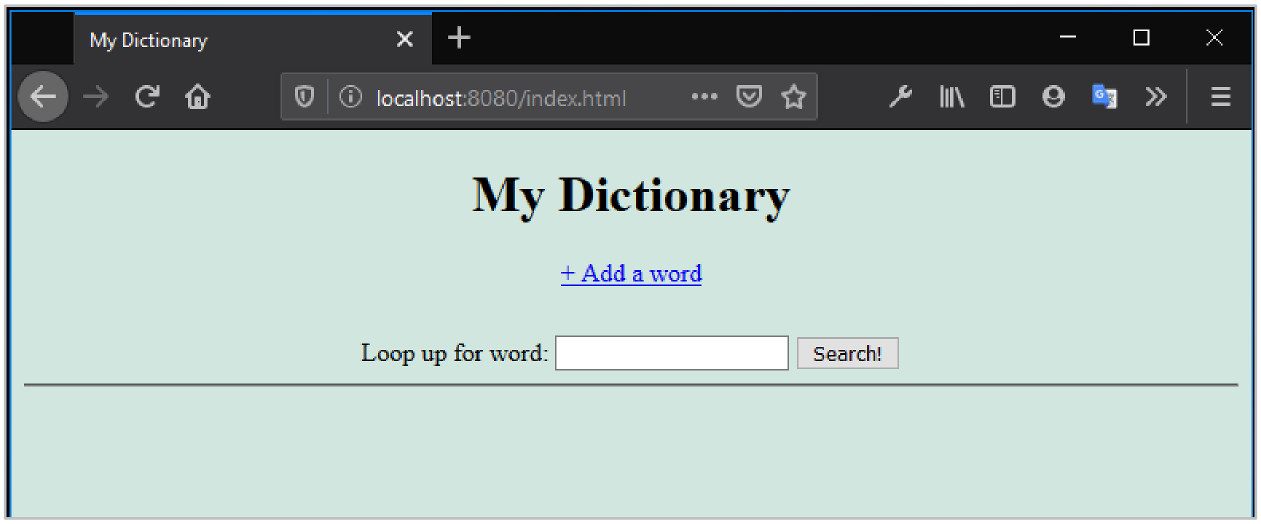
For example, http://localhost:8080/lookup?key=여자


localhost:8080/index.html?key=여자
localhost:8080/index.html?key=여자

woman

To make it easier for the next exercises, we will create an English dictionary instead (Korean requires a specific keyboard settings).

From solution of the previous tutorial, copy the content of folder `/express-server/` & start.



### Task 1: UI

Create a UI just like the image above. (***completed from the starter_pack – have a look for JS tasks below)***

- "+ Add a word" is a link to navigate to another page for adding a new word into dictionary `add.html`

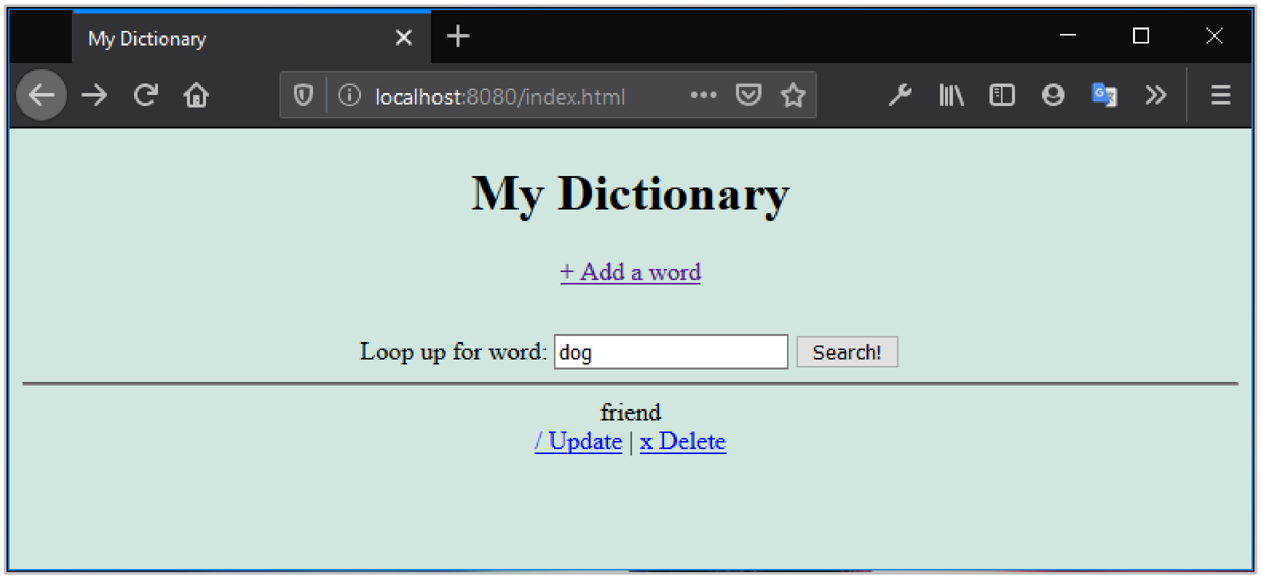
### Task 2: fetch()

In JavaScript, call **fetch()** to the `/lookup` API with the word that user inputted, then display result (word definition) (see the image above).

### Task 3: JSON

- **Back-end**: modify the `/lookup` API to return JSON object
- **Front-end**: modify the fetch to work with the JSON response from server.

Add: when the word exists, not only the definition of the word appears, two links to corresponding functionalities update/ delete that word are displayed.



My Dictionary

+ Add a word

Loop up for word: dog Search!

friend

/ Update | x Delete

## Exercise 2: [C] Dictionary – Add a word (20 mins)

### Task 1: Back-end

Create an API end point: `/add?word=…&definition=…` to add a new word with definition into the dictionary (**add new key-value pair into DICTIONARY object**).

**Note**: In practical, for creating,

- you should specify another method to use `app.post()` for updating, instead of `app.get()`
- you should NOT send data via query params like this. (use: message body)

### Task 2: Front-end

In `/public` folder, create a new file named **add.html**.

- Create form with inputs for word & definition.
- Then with user input data, call `fetch()` to the API created in *Task 1*.

If success then redirect user to */lookup* page (index.html) w3schools

Try to add some words from here:
https://www.oxfordlearnersdictionaries.com/wordlist/english/oxford3000/

Then test by lookup with the new added words.

## Exercise 3: [U] Dictionary – Update a word (20 mins)

### Task 1: Back-end

Create an API end point: `/update?word=…&definition=…` to update the definition of a given word in the dictionary. (**update existing key-value pair in DICTIONARY object)**

**Note**: In practical, for updating,

- you should specify another method to use `app.patch()` for updating, instead of `app.get()`
- you should NOT send data via query params like this. (use: message body)

### Task 2: Front-end

In `/public` folder, create a new file named **update.html**.

- Create form with inputs for word & definition.
- Then with user input data, call `fetch()` to the API created in *Task 1*.

If success then redirect user to */lookup* page (index.html) w3schools

## Exercise 4: [D] Dictionary – Delete a word (15 mins)

### Task 1: Back-end

Create an API end point: `/delete?word=…` to delete the given word from the dictionary. (**delete specified key-value pair from DICTIONARY object)**

**Note**: In practical, for deleting,

- you should specify another method to use `app.delete()` for updating, instead of `app.get()`
- allowing delete via GET, potentially delete all your data while indexing by the searching tools like Google, Bing…

### Task 2: Front-end

Practically, no specific page required for this task. We can handle the event user click on button delete then invoke API created in *Task 1* directly in index.html then refresh page if success. To make it simple & consistent:

In `/public` folder, create a new file named **delete.html**.

- call `fetch()` to the API created in *Task 1*.

If success then redirect user to */lookup* page (index.html) w3schools

## [Extra] Exercise 5: Dictionary Assessment

With this version of our dictionary app, you can look up normally and also fully manage the words in the dictionary. However, do you think about any limitation(s) of the current version of this app?

If any, then give your suggestion(s) to overcome those limitations.