Batch #12 / Back-end Class

Remote Learning Assignment - Week 3

Assignment 1: Your First Web Server

To build your first web server for development, follow the steps below:

1. Install Node.js
2. Create a Node.js project by NPM
3. Install Express module in your Node.js project by NPM
4. Write a simple web server program and start it
5. Show an HTML page when you enter http://localhost:3000/ in a browser's address bar (For example: a simple page including "Hello, My Server!" is an acceptable result.)

You may refer to getting started document in Express official website to complete this assignment.

Reminders:

1. You have to learn how to use command line interface in your computer.
2. Set up your GitHub repository to ignore folder node_modules, which includes all the modules installed in your Node.js project. Refer to Ignoring Files document.
3. All the assignments in this week should continue with the same Node.js project built in this assignment.
4. You don't need to split folders for each assignment this week, your folder structure could be like: remote-assignments/Week3/Assignments

Batch #12 / Back-end Class

Remote Learning Assignment - Week 3

Assignment 2: Build Backend API for Front-End

Now, try to modify your code executed on the server side to build a simple API. Your server should fulfill following client requests:

1. When user enter http://localhost:3000/getData in a browser's address bar, show "Lack of Parameter" message in the page.
2. When user enter http://localhost:3000/getData?number=xyz in a browser's address bar, show "Wrong Parameter" message in the page.
3. When user enter http://localhost:3000/getData?number=5, they should get the result of 1+2+....+5 in the page.
4. Generally speaking, when user enter http://localhost:3000/getData?number=正整數, they can get result of 1+2+....+正整數 in the page.

Hint: handle HTTP GET method and parameters with Express on the server side.

Batch #12 / Back-end Class

Remote Learning Assignment - Week 3

Assignment 3: Connect to Backend API by AJAX

You have built your first API in the backend, then let's get back to front-end. Follow the steps below to send an HTTP request to your backend API by AJAX.

1. Update your Express project to serve static files. You can refer to this document.
2. Serve a static HTML file named sum.html. It means you can enter http://localhost:3000/sum.html in a browser's address bar to get this HTML page.
3. Write some JavaScript code in sum.html to make an HTTP request by AJAX to http://localhost:3000/getData?number=10, and get the result 55 from server.
4. Write a simple user interface to let users enter a number and get result from server. (For a simple example, a text input and a button.)

Hint: refer to W3Schools or MDN for learning more about AJAX.

Batch #12 / Back-end Class

Remote Learning Assignment - Week 3

Assignment 4: HTTP Cookie

Cookie is an important mechanism for storing small piece of data in the browser. Modify your code executed in the backend to use cookies for user tracking.

1. Serve a URL http://localhost:3000/myName by your server.
2. When user connects to http://localhost:3000/myName, check cookies for user's name in the backend.
    a. If you can get user's name from cookies, show it in the web page. **Done.**
    b. If you cannot get user's name from cookies, show a HTML form including a text input and a button in the web page. **Go to step 3.**
3. User can enter his name in the text input, and then click button to submit form to a URL http://localhost:3000/trackName?name=使用者的輸入 which should be served from your server, too.
4. When user submits form to http://localhost:3000/trackName?name=使用者的輸入, you should get user's name from HTTP parameter and store it in the cookies.
5. Redirect user to http://localhost:3000/myName, user can see his name, finally. **Done.**

References:

1. Document for using cookie-parser with Express to get/check cookies in the backend.
2. Document for setting cookies in the backend.

Batch #12 / Back-end Class

Remote Learning Assignment - Week 3

Assignment 5: Algorithm Practice (Advanced Optional)

Given an array of integers, return indices of the two numbers such that they add up to a specific target. You may assume that each input would have exactly one solution, and you may not use the same element twice.

```
function twoSum(nums, target) {
	// your code here
}

/*
	For example:
		twoSum([2, 7, 11, 15], 9);
	Should returns:
		[0, 1]
	Because:
		nums[0]+nums[1] is 9
*/
```