


Assignment 5 - Async and Express

[Start Assignment](#)

Due Feb 14 by 11:59pm **Points** 17 **Submitting** a file upload **Available** Feb 6 at 8am - Feb 16 at 11:59pm

Introduction




In this assignment, you will write a web app that calls the [Random User Generator web service](https://randomuser.me/)  (<https://randomuser.me/>) and displays data from the response from this web service in the browser. You will also add a named Express middleware function to maintain and print certain statistics.

Learning Outcomes

- How can we modify the DOM tree? (Module 4, MLO 4)
- What are DOM events and how can we use them to create interactive web applications? (Module 4, MLO 5)
- How to write asynchronous JavaScript programs using promises? (Module 5, MLO 2)
- How to write asynchronous JavaScript programs using the `await` and `async` keywords? (Module 5, MLO 3)
- How is the concept of modules supported in JavaScript? (Module 5, MLO 4)
- What is Express middleware and what are its uses? (Module 5, MLO 5)

Instructions

Write a web app that calls the Random Person API, <https://randomuser.me/api/>,  (<https://randomuser.me/api/>) from the browser and from an Express app, and maintains statistics using a named Express middleware function. Your app will consist of:

- A homepage and client-side JavaScript code.
 - A static HTML page that is displayed as the homepage (i.e., displayed in the browser for the URL <http://localhost:3000/>),  (<http://localhost:3000/>) and includes 2 links. Clicking on the 2 links executes JavaScript code to send 2 different HTTP requests
- An Express app listening at port 3000.
 - This app serves the homepage.

- Provides an endpoint for `GET` requests to the URL `/random-person`
- Includes a named middleware function to maintain and print statistics to the console running the Express app.

Starter Code

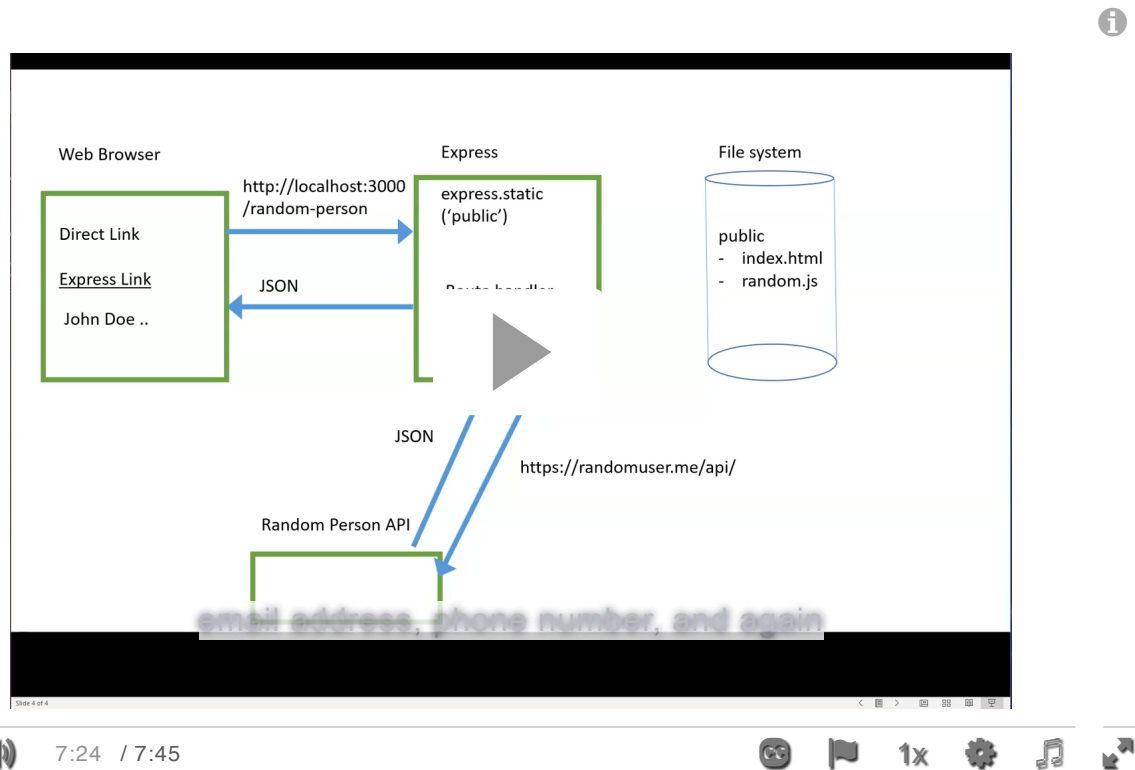
We have provided you with [a zip file \(https://canvas.oregonstate.edu/courses/1901690/files/96835238?wrap=1\)](https://canvas.oregonstate.edu/courses/1901690/files/96835238?wrap=1)  [\(https://canvas.oregonstate.edu/courses/1901690/files/96835238/download?download_frd=1\)](https://canvas.oregonstate.edu/courses/1901690/files/96835238/download?download_frd=1) that you must use as starter code.

- Unzip the zip file, go to the directory where the files have been unzipped, then run the commands `npm install` once (to install the necessary packages) and then `npm start` to start the web app corresponding to the starter code.
- Don't change the names of the files we have provided to you.
- You can modify these files, with some exceptions, as noted below.
- The file `index.html` in the directory `public` is the homepage.
 - Modify this file to add the HTML elements required in the homepage.
 - **Do not add any JavaScript code in `index.html`.**
- The file `random.js` is in the directory `public`.
 - This file is referenced in the file `index.html` using the `script` tag.
 - Modify this file to add all your client-side JavaScript code.
 - Note: You must use the DOM API introduced in Module 4, and you are not allowed to use any libraries or framework in `random.js`. For example, you cannot use React or jQuery or any other library.
- The file `.env` specifies the port number (3000) that the server side Express app will listen on.
 - Don't change this file.
- The file `package.json` must not be changed without approval from the instructor.
 - This file includes the dependencies `dotenv`, `express`, `express-async-handler`, `node-fetch` and `nodemon`.
 - The `start` property is set to start the app using the file `server.mjs`.
 - Because you have been provided this file, you don't need to run the command `npm init` to initialize the app.
 - You can simply run the command `npm install` (once) to install the dependencies and then run `npm start` whenever you want to start the app.
 - Note that the app is run using `nodemon`. This means that when you make any changes to the files `random.js` or `server.mjs` and save that file, the Express app will automatically restart and pick up these code changes.
 - If you want to make any changes to `package.json`, e.g., to add other dependencies in this file, you must get the approval of the instructor. Don't make any changes in this file without our approval.

- The file `server.mjs` will contain the code you will write for the Express app, i.e., the server-side of your web app.
 - Add code in this file for a route handler, and for maintaining and printing statistics.
- You are also allowed to (but not required to) add a file (or files) for CSS stylesheet(s) to the web app.
- Other than the optional CSS stylesheets, don't add any other files without approval from the instructor.

Video Describing the Functionality of the Web App

Here is a video that describes the functionality at a high-level.



Homepage and Client-side JavaScript Code


The homepage must include

- An `h1` element
- Some descriptive text, and

- 2 links, using `<a>` elements, with short content that describes the behavior of the links. We will refer to these links as "The Direct Link" and "The Express Link." The behavior of each link is described below.

The Direct Link

A user click on this link, i.e., the click event on the link, must result in the execution of JavaScript code in the file `random.js` that

- Uses **the fetch API**  [_\(https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch\)_](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch) to send a request to the Random User API.
- Displays the first name, last name, phone number and email of the random person sent in the response sent by the Random User API by adding one or more nodes in the page using the DOM API.

The Express Link


A user click on this link, i.e., the click event on the link, must result in the execution of JavaScript code in the file `random.js` that

- Uses the fetch API to send a request to the Express app's endpoint `GET /random-person`.
- Displays the first name, last name, phone number and email of the random person sent in the response sent by the Express API by adding one or more nodes in the page using the DOM API.

The Server-Side Express App

Endpoint to Call Random Person API

Code a route handler for the endpoint `GET /random-person`.

- This route handler must send a request to the Random Person API using the `npm` package **node-fetch**  [_\(https://www.npmjs.com/package/node-fetch\)_](https://www.npmjs.com/package/node-fetch).
- On success, the Random Person API sends back a response with status code 200 and JSON as the response body. In this case, the route handler must send back a response (to the request it received) with status code 200 and the JSON received from the Random Person API as the response body.
- You don't need to code for the case where the Random Person API sends back any other response code besides 200.

Named Middleware Function to Print Statistics

Write a **named middleware** function that maintains a count of how many HTTP requests have been received for the endpoint `GET /random-person` and prints these statistics to the console on every 10th request.

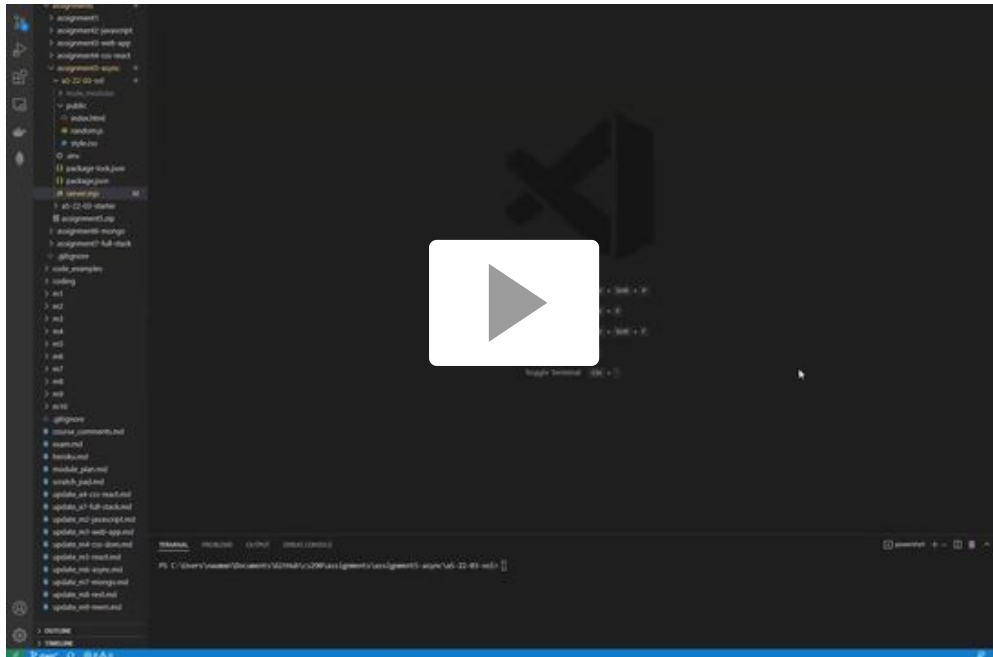
- This named middleware function must maintain a count of these requests received from the time the server was started.
 - In other words, keep the statistics in memory. Don't persist them to the file system or a database.
- The statistics must be printed to the console running the Express app after the 10th request and then every 10 requests after that.
 - i.e., after 10 retrieve requests are received, then after a total of 20 retrieve requests, then after 30, and so on.
- The maintenance and printing of the statistics must be done solely by this named middleware function, and not by the code of the `app.get` route handler for this endpoint.

Example: Printing Statistics

```
Total requests for random-person: 20
```



Video of a Sample Solution

Here is a video walk-through of a sample solution for Assignment 5.



Tips

- In your client-side and server-side JavaScript code you are free to write the asynchronous code using either the `then()` method or the `async/await` keywords.
 - You can choose to use one approach in the client-side code and the other in the server-side code, or use the same approach on both sides.
- **Exploration — Writing Asynchronous Code** (<https://canvas.oregonstate.edu/courses/1901690/pages/exploration-writing-asynchronous-code>) contains examples of web apps where the client-side JavaScript code has functionality very similar to what you need for this assignment.
 - Based on whether you want to write asynchronous code using the `then()` method or the `async/await` keywords, carefully study the corresponding example from that exploration. If you want, feel free to adapt the example for this assignment.
- Keep in mind that the file `random.js` gets downloaded to the browser and the code in the file is executed in the browser.
 - If you add any `console.log()` statements in `random.js`, the output will be printed in the browser console (and not in the terminal from where you started the server-side Express app).

- Since the code is running in the browser, it has the object `document` available to it. Recall from [Exploration — The Document Object Model](https://canvas.oregonstate.edu/courses/1901690/pages/exploration-the-document-object-model) (<https://canvas.oregonstate.edu/courses/1901690/pages/exploration-the-document-object-model>) that `document` is the root of the DOM tree.
- The Node.js packages we have been using in our server-side code are not available in the browser. So don't try to import them into `random.js`.
- In fact, you don't need to import any packages in `random.js`.
- To debug `random.js` use the browser's Dev Tools
 - The browser Dev Tools include JavaScript debuggers. You can use these debuggers to set breakpoints, watch variables, etc.
 - See the following links for more info about JavaScript debuggers in the browsers
 - <https://developer.chrome.com/docs/devtools/javascript/>  (<https://developer.chrome.com/docs/devtools/javascript/>)
 - https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_are_browser_developer_tools#the_javascript_debugger  (https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_are_browser_developer_tools#the_javascript_debugger)
- Reload the homepage if you make changes to `random.js`.
 - If you change `random.js`, it will update the file stored on the file system, but the updated file will not automatically get reloaded into the browser.
 - The browser needs to send a request for the file for it be reloaded in the browser. If you reload the homepage, the changed `random.js` will be loaded into the browser.
- Keep in mind that the file `server.mjs` runs in the Express server and not the browser.
 - If you have a `console.log` statement in `server.mjs` you will see the output in the terminal/shell where you started your server-side Express app (and not in the browser console).
 - Since `server.mjs` runs in the Express server, it will not have (and doesn't need to have) access to the object `document` (which is only available to code running in the browser).
- Don't import `random.js` into `server.mjs` or try to run `random.js` using Node.
- You may find that the route handler you implemented for requests to `GET /random-person` returns the status code 304, instead of 200. Status code 304 tells the browser to use its cache. If this happens, you should disable the browser cache and then the status code should be 200. To disable the browser cache, do the following:
 - Open the browser's Dev Tools.
 - Go to the Network Tab.

- Make sure that the "Disable cache" checkbox is checked.
- Reload the page.

What to Turn In?

- Submit a single zip file with your code.
- This zip file must be named `youronid_assignment5.zip` where `youronid` should be replaced by your own ONID.
 - E.g., if `chaudhrn` was submitting the assignment, the file must be named `chaudhrn_assignment5.zip`.
 - When you resubmit a file in Canvas, Canvas can attach a suffix to the file, e.g., the file name may become `chaudhrn_assignment5-1.zip`. Don't worry about this name change as no points will be deducted because of this.
- In the zip file, you must include all the code for your application, except the `node_modules` directory.
- The grader will unzip your file, go to the root directory, run `npm install` and then run `npm start` to start your application and test it.

Grading Criteria

- This assignment is worth 8% of your final grade.
- The points for the assignment and the break-up for items is described in the grading rubric.

Rubric

For grading details please see the attached rubric.

Assignment 5

Criteria	Ratings			Pts
The homepage has all the required elements when it is first displayed.	1 pts Full Marks The homepage includes at least an h1 element, descriptive text in another element and two links using a elements. It is OK for the homepage to contain more elements.	0.5 pts Partially met The homepage is missing exactly one of the required elements.	0 pts No Marks The homepage is missing more than one of the required elements.	1 pts
A click on the Direct link invokes JavaScript code in the file random.js.	1 pts Full Marks A click on the Direct link invokes JavaScript code in the file random.js.	0 pts No Marks A click on the Direct link either doesn't invoke JavaScript code or it invokes JavaScript code but that code is not in the file random.js.		1 pts
A click on the Direct link sends a request to the Random Person API.	1 pts Full Marks A click on the Direct link results in sending a request to the Random Person API using the fetch API.	0.5 pts Partially met A click on the Direct link results in sending a request to the Random Person API but doesn't use the fetch API.	0 pts No Marks A click on the Direct link doesn't result in sending a request to the Random Person API.	1 pts
A click on the Direct link displays the required data from the response in the browser.	1 pts Full Marks A click on the Direct link displays first name, last name, phone number and email in the browser of the person in the response received from calling the Random Person API.	0.5 pts Partially met A click on the Direct link displays only 3 of the required properties of a person or it display any properties other the 4 required properties of the person in the response from the Random Person API.	0 pts No Marks A click on the Direct link displays fewer than 3 of the required properties of the person in the response from the Random Person API.	1 pts
A click on the Direct link displays data in the page by adding one or more nodes in the page using the DOM API.	2 pts Full Marks A click on the Direct link displays the required data in the page by adding one or more nodes in the page using the DOM API.	0 pts No Marks A click on the Direct link displays the data in a new page. Or the data is displayed in the homepage but one or more of the existing nodes in the page are no longer displayed.		2 pts

Criteria	Ratings			Pts
A click on the Express link invokes JavaScript code in the file random.js.	1 pts Full Marks A click on the Express link invokes JavaScript code in the file random.js.	0 pts No Marks A click on the Express link either doesn't invoke JavaScript code or it invokes JavaScript code but that code is not in the file random.js.		1 pts
A click on the Express link sends a request to the random-person route in the Express app.	1 pts Full Marks A click on the Express link sends a request to the random-person route in the Express app using the fetch API.	0.5 pts Partially met A click on the Express link sends a request to the random-person route in the Express app but doesn't use the fetch API.	0 pts No Marks A click on the Express link doesn't send a request to the random-person route in the Express app.	1 pts
A click on the Express link displays the required data from the response in the browser.	1 pts Full Marks A click on the Express link displays first name, last name, phone number and email in the browser of the person in the response received from calling the Express app.	0.5 pts Partially met A click on the Express link displays only 3 of the required properties of a person or it display any properties other the 4 required properties of the person in the response from the Express app.	0 pts No Marks A click on the Express link displays fewer than 3 of the required properties of the person in the response from the Express app.	1 pts
A click on the Express link displays data in the page by adding one or more nodes in the page using the DOM API.	2 pts Full Marks A click on the Express link displays the required data in the page by adding one or more nodes in the page using the DOM API.	0 pts No Marks A click on the Express link displays the data in a new page. Or the data is displayed in the homepage but one or more of the existing nodes in the page are no longer displayed.		2 pts
Named middleware function for count of requests	2 pts Full Marks A named middleware function has been implemented to maintain a count of requests	0 pts No Marks A named middleware function has not been implemented to maintain a count of requests for GET requests to the URL /random-person. Or		2 pts

Criteria	Ratings			Pts
	for GET requests to the URL /random-person since the Express app was started.	a named middleware function has been implemented but it doesn't maintain a count of requests for the required endpoint.		
The count of requests for GET /random-person is printed on the console running the Express app on every 10th request	1 pts Full Marks The count of requests for GET /random-person is printed on the console running the Express app on every 10th request	0.5 pts Partially met The count of requests for GET /random-person is printed on the console running the Express app for the 10th request but is not printed afterwards.	0 pts No Marks The count of requests for GET /random-person is never printed on the console running the Express app.	1 pts
Express route handler for GET /random-person	2 pts Full Marks A route handler is implemented in the Express app for requests to GET /random-person, it calls the Random Person API and returns the response from the Random Person API as its own HTTP response with a status code of 200.	0 pts No Marks A route handler has not been implemented in the Express app for requests to GET /random-person. Or a route handler has been implemented but it doesn't call the Random Person API. Or the route handler has been implemented, it calls the Random Person API, but it doesn't return the response from the Random Person API as its own HTTP response or the response status code is not 200.		2 pts
Express route handler for GET /random-person uses node-fetch	1 pts Full Marks Express route handler for GET /random-person uses node-fetch to call the Random Person API.	0 pts No Marks Express route handler for GET /random-person either doesn't call the Random Person API or uses some package other than node-fetch to call that API.		1 pts
Total Points: 17				