

# Lab 5

Start Assignment

- Due Tuesday by 11:59pm
- Points 100
- Submitting a file upload
- File Types zip
- Available Feb 27 at 12am - Mar 12 at 11:59pm

## JSON Routes

For this lab, you will create a simple server that will provide data from an API.

You will be downloading JSON files from the following GitHub Gists:

- [people.json](https://gist.github.com/graaffixnyc/448017f5cb43e0d590adb744e676f4b5/raw/495e09557914db)  
(<https://gist.github.com/graaffixnyc/448017f5cb43e0d590adb744e676f4b5/raw/495e09557914db>)
- [companies.json](https://gist.github.com/graaffixnyc/90b56a2abf10cfd88b2310b4a0ae3381/raw/f43962e103672e)  
(<https://gist.github.com/graaffixnyc/90b56a2abf10cfd88b2310b4a0ae3381/raw/f43962e103672e>)

For this lab, you **must** use the `async/await` keywords (not Promises). You will also be using [axios](https://github.com/axios/axios) (<https://github.com/axios/axios>), which is a HTTP client for Node.js; you can install it with `npm i axios`.

## General Notes


Lecture videos and demos tend to show JSON as "pretty", but your browser may not natively do that -- that's fine!

There are extensions for most major browsers that add that functionality, such as:

1. [JSONView for Firefox](https://addons.mozilla.org/en-US/firefox/addon/jsonview/) (<https://addons.mozilla.org/en-US/firefox/addon/jsonview/>)
2. [JSONView for Chrome](https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc?hl=en)  
(<https://chrome.google.com/webstore/detail/jsonview/chklaanhfefbnpoihckbnefhakgolnmc?hl=en>)

## Folder Structure

You will use the folder structure in the stub for the data & routes module, and other project files. There is an extra file in the stub called helpers.js. You can add all your helper/validation functions in that file to use in your other modules.

**YOU MUST use the directory and file structure in the code stub, or points will be deducted.** You can download the starter template here: [lab5\\_stub.zip](https://sit.instructure.com/courses/71954/files/13095386?wrap=1)  
(<https://sit.instructure.com/courses/71954/files/13095386?wrap=1>) 

([https://sit.instructure.com/courses/71954/files/13095386/download?download\\_frd=1](https://sit.instructure.com/courses/71954/files/13095386/download?download_frd=1))

([https://sit.instructure.com/courses/68061/files/11614220/download?download\\_frd=1](https://sit.instructure.com/courses/68061/files/11614220/download?download_frd=1))

(<https://sit.instructure.com/courses/61386/files/10351577?wrap=1>)\_ PLEASE NOTE: THE STUB DOES NOT INCLUDE THE PACKAGE.JSON FILE. YOU WILL NEED TO CREATE IT! DO NOT ADD ANY OTHER FILE OR FOLDER APART FROM PACKAGE.JSON FILE. Do not forget to add the start command the the type module property!

## Your routes

`/people`

When making a GET request to `http://localhost:3000/people`, this route will return all the JSON data that is returned from the axios call to `people.json`. It will respond with all 1000 people. In your route, you will use the `res.json()` method. Notice that your server will respond with the JSON data, meaning your browser will display it with quotes around the key names. You do not have to do anything special for this, `res.json` stringifys the object before it sends it to the browser automatically.

Making a request to <http://localhost:3000/people>. ➡ (<http://localhost:3000/books>) returns (sample output only shows the first 5 people in the data, you would need to return ALL the people:

```
[
  {
    "id": "fa36544d-bf92-4ed6-aa84-7085c6cb0440",
    "first_name": "Archambault",
    "last_name": "Forestall",
    "email": "aforestall0@usnews.com",
    "phone_number": "702-503-4409",
    "address": "07322 Sugar Avenue",
    "city": "Las Vegas",
    "state": "Nevada",
    "postal_code": "89140",
    "company_id": "ed37ae87-f461-42d2-bf24-8631aad856de",
    "department": "Services",
    "job_title": "Social Worker"
  },
  {
    "id": "619a24c2-3950-4ca2-9020-2b349d6ce888",
```

```
"first_name": "Hewet",
"last_name": "Grinikhin",
"email": "hgrinikhin1@netvibes.com",
"phone_number": "806-534-4820",
"address": "42989 Oxford Drive",
"city": "Amarillo",
"state": "Texas",
"postal_code": "79182",
"company_id": "37f4e2b1-6006-47d7-8887-fe8a14be86ce",
"department": "Research and Development",
"job_title": "Geologist I"
},
{
  "id": "41455e7d-b133-4aae-ac5d-a3a80567feb0",
  "first_name": "Sondra",
  "last_name": "Perrett",
  "email": "sperrett2@microsoft.com",
  "phone_number": "202-570-7005",
  "address": "2 Declaration Trail",
  "city": "Washington",
  "state": "District of Columbia",
  "postal_code": "20525",
  "company_id": "383e3daf-de20-4436-b367-67a64a51c9fc",
  "department": "Sales",
  "job_title": "Librarian"
},
{
  "id": "f533ff08-d013-48c8-8857-3cb457ef9058",
  "first_name": "Judith",
  "last_name": "Alsford",
  "email": "jalsford3@google.it",
  "phone_number": "405-753-1520",
  "address": "704 Tony Terrace",
  "city": "Oklahoma City",
```

```

"state": "Oklahoma",
"postal_code": "73152",
"company_id": "c8f8fe75-a0da-43a4-9998-5128c98bee81",
"department": "Accounting",
"job_title": "Programmer II"
},
{
  "id": "12b7c9b7-c803-4d68-add1-20eb98d2c488",
  "first_name": "Prue",
  "last_name": "Shieldon",
  "email": "pshieldon4@businessinsider.com",
  "phone_number": "540-606-5053",
  "address": "90 Holmberg Hill",
  "city": "Roanoke",
  "state": "Virginia",
  "postal_code": "24029",
  "company_id": "4f778208-f6b1-4e1f-a94d-839a6bfc64fb",
  "department": "Marketing",
  "job_title": "Office Assistant III"
}, .....more results
]

```

`/people/:id`

When making a GET request to `http://localhost:3000/people/:id`, this route will respond with the JSON data for a single person from the dataset. If the ID cannot be found in the List of people (i.e. there is no person with that ID the route should respond with a person not found error) you will return a 404 status code along with a "Person Not Found!" error message.


**So for example: Making a request to** <http://localhost:3000/books/5d7a28c2-e076-47c3-a327-cbd123b423f0> <http://localhost:3000/people/fa36544d-bf92-4ed6-aa84-7085c6cb0440> <http://localhost:3000/people/fa36544d-bf92-4ed6-aa84-7085c6cb0440> **returns:**

```
{
```

```
"id": "fa36544d-bf92-4ed6-aa84-7085c6cb0440",
"first_name": "Archambault",
"last_name": "Forestall",
"email": "aforestall0@usnews.com",
"phone_number": "702-503-4409",
"address": "07322 Sugar Avenue",
"city": "Las Vegas",
"state": "Nevada",
"postal_code": "89140",
"company_id": "ed37ae87-f461-42d2-bf24-8631aad856de",
"department": "Services",
"job_title": "Social
}
```

[/companies](#)

When making a GET request to <http://localhost:3000/companies>, this route will return all the JSON data that is returned from the axios call to companies.json. It will respond with all companies from companies.json. In your route, you will use the `res.json()` method. Notice that your server will respond with the JSON data, meaning your browser will display it with quotes around the key names. You do not have to do anything special for this, `res.json` stringifys the object before it sends it to the browser automatically.

Making a request to <http://localhost:3000/>  <http://localhost:3000/authors> companies returns (sample output only shows the first 5 companies in the data, you would need to return ALL the companies:

```
[
  {
    "id": "fb90892a-f7b9-4687-b497-d3b4606faddf",
    "name": "Yost, Harris and Cormier",
    "street_address": "71055 Sunbrook Circle",
    "city": "Austin",
    "state": "TX",
    "postal_code": "78715",
    "industry": "Apparel"
  },
  {
```

```
"id": "bfe1ddff-f8ba-40a2-a42e-5062a6bec303",
"name": "Runte Inc",
"street_address": "68166 Rowland Road",
"city": "Tulsa",
"state": "OK",
"postal_code": "74184",
"industry": "Major Pharmaceuticals"
},
{
"id": "d6a867d5-6e71-48d8-977a-8243154126e5",
"name": "Homenick, Skiles and Konopelski",
"street_address": "1 Stuart Drive",
"city": "Topeka",
"state": "KS",
"postal_code": "66667",
"industry": "Military/Government/Technical"
},
{
"id": "ce50b9d1-c164-44d0-8299-408d217ca287",
"name": "Gutkowski-Rosenbaum",
"street_address": "2520 Comanche Court",
"city": "Colorado Springs",
"state": "CO",
"postal_code": "80940",
"industry": "n/a"
},
{
"id": "f9165a9d-b56a-48f6-9c30-baf6f6ce3edb3",
"name": "Schaden LLC",
"street_address": "58081 Mandrake Junction",
"city": "Richmond",
"state": "VA",
"postal_code": "23277",
"industry": "n/a"
}
```

```
} , ...more results  
]
```

`/companies/:id`

When making a GET request to `http://localhost:3000/companies/:id`, this route will respond with the JSON data for a single company. If the ID cannot be found in the list of companies (i.e. there is no company with that ID) you will return a 404 status code along with a "Company Not Found!" error message.

**So for example: Making a request to** <http://localhost:3000/authors/a7da4a33-75a6-4059-83a6-493c1117fce2> **http://localhost:3000/comapnies/fb90892a-f7b9-4687-b497-d3b4606faddf **returns:****

```
{  
  "id": "fb90892a-f7b9-4687-b497-d3b4606faddf",  
  "name": "Yost, Harris and Cormier",  
  "street_address": "71055 Sunbrook Circle",  
  "city": "Austin",  
  "state": "TX",  
  "postal_code": "78715",  
  "industry": "Apparel"  
}
```

## Packages you will use:

You will use the **express** package as your server.

You will use the **axios** package to get data from the API.

You can read up on [express \(http://expressjs.com/\)](http://expressjs.com/) on its home page. Specifically, you may find the [API Guide section on requests \(http://expressjs.com/en/4x/api.html#req\)](http://expressjs.com/en/4x/api.html#req) useful.

You may use the [lecture 5 code \(https://github.com/stevens-cs546-cs554/CS-546/tree/master/lecture\\_05\)](https://github.com/stevens-cs546-cs554/CS-546/tree/master/lecture_05) as a guide.

**You must save all dependencies to your package.json file**

# Requirements

1. You **must not submit** your node\_modules folder
2. You **must remember** to save your dependencies to your package.json folder
3. You **must remember** to update your package.json file to set `app.js` as your starting script!
4. You **must** submit a zip archive or you will lose points, named in the following format:

`LastName_FirstName_CS546_SECTION.zip`

You will lose points for not submitting an archive.

## Lab 5 Spring 2024



Criteria	Ratings		Pts
<p>/people</p> <p>You MUST return the DATA exactly as shown, naming a key wrong, having the wrong output, will result in the full points off for the route. Even if you have a typo/naming in a single key name wrong will result in FULL points for the function off.</p>	<p><b>25 pts</b> <b>Full Marks</b></p> <p>1 test case. It either returns the correct results from the JSON files or does not.</p>	<p><b>0 pts</b> <b>No Marks</b></p>	25 pts
<p>/people/:id</p> <p>You MUST return the DATA exactly as shown, naming a key wrong, having the wrong output, will result in the full points off for the route. Even if you have a typo/naming in a single key name wrong will result in FULL points for the function off.</p>	<p><b>25 pts</b> <b>Full Marks</b></p> <p>12.5 pts for test case with an existing id, 12.5 pts for test case with non-existent id.</p>	<p><b>0 pts</b> <b>No Marks</b></p>	25 pts
<p>/companies</p> <p>You MUST return the DATA exactly as shown, naming a key wrong, having the wrong output, will result in the full points off for the route. Even if you have a typo/naming in a single key name wrong will result in FULL points for the function off.</p>	<p><b>25 pts</b> <b>Full Marks</b></p> <p>1 test case. It either returns the correct results from the JSON files or does not.</p>	<p><b>0 pts</b> <b>No Marks</b></p>	25 pts
<p>/companies/:id</p> <p>You MUST return the DATA exactly as shown, naming a key wrong, having the wrong output, will result in the full points off for the route. Even if you have a typo/naming in a single key name wrong will result in FULL points for the function off.</p>	<p><b>25 pts</b> <b>Full Marks</b></p> <p>12.5 pts for test case with an existing id, 12.5 pts for test case with non-existent id.</p>	<p><b>0 pts</b> <b>No Marks</b></p>	25 pts
Total Points: 100			