

Express Routing Exercises

For this exercise, you will build an Express.js application that performs three statistical operations given an arbitrary amount of numbers:

1. **mean** (average)
2. **median** (midpoint)
3. **mode** (most frequent)

The operations are invoked via **one route per operation**.

Requirements

The three base routes are **`/mean`**, **`/median`**, **`/mode`**. All accept GET requests

Each route takes a query key of **`nums`** which is a comma-separated list of numbers. For example, if I want to get the mean of 1, 3, 5, and 7, that would look like be a GET request to **`/mean?nums=1,3,5,7`**.

The response of each operation should be JSON which looks like this:

```
response: {
  operation: "mean",
  value: 4
}
```

The app should “gracefully” handle the following errors:

- *Passing in an invalid number (NaN errors). For instance, **`/mean?nums=foo,2,3`** should respond with a **400 Bad Request** status code and a response that saying something like: **`foo is not a number`**.*
- *Empty input: **`/mean`** without passing any **`nums`** should respond with a **400 Bad Request** status code saying something like **`nums are required`**.*

Make sure you have unit tests for **`mean`**, **`median`** and **`mode`**.

Further Study

- Make a route called **`/all`** that does all three operations at the same time, with the response from each of them as a key in the JSON response. It can look like this:

```
response: {
  operation: "all",
  mean: 12
  median: 10,
  mode: 8
}
```

- Provide special handling for an optional query key called **`save`** that can be set to **`true`**. This means the operation will write to a file. For example, **`/median?nums=1,3,5&save=false`** will return a json response and will write to a file called **`results.json`**.

- Insert a timestamp for every operation that writes to a file.
- Honor the Accept header. Return json if the client requests application/json and return html if the client requests text/html.

Solution

[View our solution <solution/index.html>](#)