11.4.8 Reflection

Congratulations on completing your first Express.js application! You've expanded on your knowledge of modularizing applications and successfully separated the server logic from the routes. You've also created brand-new functionality for the zookeepers at the zoo!

Here's a recap of the things you accomplished in this lesson:

- Broke animal functions into their own files within `lib/`.
- Used `express.Router()` instead of `app`.
- Created new endpoints and data for zookeepers.

Now you can confidently set up an Express.js server that serves HTML and JSON to a client.

Here's a recap of everything you accomplished in this module:

- Used Express.js to create a Node.js web server.
- Set up GET routes to serve selective JSON data based on parameters (`req.param` and `req.query`).

- Used Heroku to deploy a server and learned about the difference between development and production environments.
- Created POST routes to allow your server to accept incoming data and tested it with Insomnia Core.
- Learned how and why to use middleware in Express.js.
- Updated your server to serve client-side code for users to interface with.
- Revisited the Fetch API to make requests to your server.
- Modularized and scaled your API by using the Express.js Router.
- Tested your application using Jest and helped make it predictable as you scaled up.

You now have a solid understanding of how to build servers to transfer data to and from a client. In the next module, you'll add performant data persistence to our back-end applications with SQL databases.