Express Request Parameters and Bodies

Learning Objectives

- Understand how Express handles route parameters
- Understand how Express handles request bodies

Material

Route Parameters

If you browse the web and pay attention to the URL bar, you'll notice that URLs often contain identifier sequences . For example, take this Twitter URL https://twitter.com/_mikepound. The "_mikepound" is clearly a username, but it would be impossible for Twitter to create an endpoint for each possible user e.g.

```
app.get("/_mikepound", (req, res) => { ... });

app.get("/tim_cook", (req, res) => { ... });

// ... and so on
```

Instead, we want to generalise this pattern so it works for any username. Express lets us do this like so:

```
app.get("/:username", (req, res) => {
      console.log(req.params);
      // { username: "_mikepound" }
});
```

The colon (`:`) indicates that the section is a parameter. This endpoint will fire irrespective of the particular username used. For example, a GET request to https://twitter.com/tim_cook would also hit this endpoint. We can access what the actual value of the variable section is using the `req.params` object, which Express populates for us.

Here's another example to illustrate the principle:

Express route path	`/users/:userId/status/:statusId`
Request URL	http://localhost:3000/users/33/status/71
`req.params`	`{"userId": "33", "statusId": "71"}`

Request Bodies

Often it's more convenient to put data in the HTTP request's body than embed it in the URL. In order for Express to be able to read request bodies, we need to add the following two lines of code:

```
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
```

Our app will now check for a body on incoming requests and, if it is json or url-encoded, will parse it and place it on a req.body object in our endpoint's callback.

```
app.post("/users", (req, res) => {
      console.log(req.body);
      // { username: "_mikepound", password: "password123" }
});
```

Core Assignments

Express Read and Write

Create a simple express server. Your server should contain a global `users` object to store user information:

```
const express = require('express');

const app = express();
const users = {};
```

The keys of your user object should be integers - user IDs. These should be unique. Each value should be a user object which should include a name and an age, e.g.

```
// users
{
  '1': { name: 'Rick', age: 63 },
  '2': { name: 'Joanna', age: 42 },
}
```

Your server should contain two endpoints:

- `POST /users` (which expects the name and age to be included in the request body)
- `GET /users/<userId>` (which returns the data for that particular user)

Test your endpoints using Postman.