Accepting Data: Handling Form Data

This lesson teaches how to make your custom web server accept data from a form.

We'll cover the following

- Handling form Data

So far, your web server offers a read-only service: it publishes some data but doesn't accept any... Until now!

As you saw in a previous chapter, information submitted to a web server can be either form data or JSON data.

Handling form Data

Form data comes encapsulated into the HTTP `POST` request sent by the client to the server. The first server task is to extract this information from the request. The simplest way to do this is to use a specialized npm package, such as multer. Install it with the `npm install multer` command or directly in your app dependencies.

```
"dependencies": {
  ...
  "multer": "^1.3.0"
},
```

Once *multer* is installed, add the following code towards the beginning of your server main file.

```
// Load the multer package as a module
const multer = require("multer");
```

```
// Access the exported service
const upload = multer();
```

The following route accepts form data sent to the `"/animals"` route. Notice the use of `app.post()` instead of `app.get()` to handle `POST` HTTP requests, and the addition of `upload.array()` as a second parameter to add a `body` object containing the fields of the form to the `request` object.

```
// Handle form data submission to the "/animals" route
app.post("/animals", upload.array(), (request, response) => {
const name = request.body.name;
const vote = request.body.strongest;
response.send(`Hello ${name}, you voted: ${vote}`);
});
```

The values of the name and vote variables are extracted from the request body, and a string is constructed and sent back to the client.