1. Starting a Server
   1. Express is a Node module, so in order to use it, we will need to import it into our program file. To create a server, the imported express function must be invoked:

const express = require('express'); // ‘require’ imports the Express library

const app = express(); //here we invoke an instance of the Express application

* 1. The purpose of a server is to listen for requests, perform whatever action is required to satisfy the request, and then return a response. In order for our server to start responding, we have to tell the server where to listen for new requests by providing a port number argument to a method called app.listen(). The server will then listen on the specified [port](https://en.wikipedia.org/wiki/Port_(computer_networking)) and respond to any requests that come into it. The second argument is a callback function that will be called once the server is running and ready to receive responses.

const PORT = 4001;

app.listen(PORT, () => {

console.log(`Server is listening on port ${PORT}`);

});

1. Writing a Route
   1. To tell our server how to deal with any given request, we register a series of *routes*. Routes define the control flow for requests based on the request’s *path* and HTTP verb. The HTTP verb is always included in the request, and it is one of a [finite number of options](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods) used to specify expected functionality. GET requests are used for retrieving resources from a server. Express uses app.get() to register routes to match GET requests. Express routes (including app.get()) usually take two arguments, a path (usually a string), and a callback function to handle the request and send a response.

const moods = [{ mood: 'excited about express!'}, { mood: 'route-tastic!' }];

app.get('/moods', (req, res, next) => {

// Here we would send back the moods array in response

});