# Lab 4 [Getting started with NodeJS]

|  |  |
| --- | --- |
| Lab Title: | Getting started with NodeJS |
| **Expected duration** (hours): | **1 hour 45 mins** |

|  |
| --- |
| Objectives |
| Understanding how to create a basic node JS app |
| Learn how to add express as a web server |
| Learn how to create product list and create product endpoints |

|  |
| --- |
| Requirements |
| A laptop or a desktop with Windows or Mac as an operating system |
| Visual studio code or something similar as a text editor |
| A modern web browser like chrome |

## Lab Instructions:

# Understanding how to create a NodeJS app

This lab will help you to get familiar with basic layout and setup for a NodeJS app, you will add multiple endpoints to list product, and create product. None of these endpoints will use anything but JavaScript. The purpose of this lab is to getting students familiar with a backend library like NodeJS, express as a web server and show the power of it.

### Part 1 - Create a basic NodeJS app

First, ensure you have Node.js and npm installed.

If not, you can download them from the [official Node.js website](https://nodejs.org/).

### **Step 1: Initialize a new Node.js project**

mkdir products-app  
cd products-app  
npm init -y

### **Step 2: Install the required packages**

npm install express

### **Step 3: Create the app**

Create a new file named app.js and add the following code:

import express from 'express';  
  
const app = express();  
const PORT = 3000;  
  
const products = [  
 { id: 1, name: 'Product 1', price: 100 },  
 { id: 2, name: 'Product 2', price: 200 },  
 { id: 3, name: 'Product 3', price: 300 },  
];  
  
app.get('/products', (req, res) => {  
 res.json(products);  
});  
  
app.listen(PORT, () => {  
 console.log(`Server is running on http://localhost:${PORT}`);  
});

### **Step 4: Enable ES6 module support**

To use ES6 modules, you need to add the following line in your package.json file:

"type": "module"

Your package.json should look like this:

{  
 "name": "products-app",  
 "version": "1.0.0",  
 "description": "",  
 "main": "app.js",  
 "type": "module",  
 "scripts": {  
 "start": "node app.js"  
 },  
 "author": "",  
 "license": "ISC",  
 "dependencies": {  
 "express": "^4.18.2"  
 }  
}

### **Step 5: Run the app**

Start the server by running:

npm start

### **Step 6: Test the app**

Open your browser or use a tool like Postman to navigate to <http://localhost:3000/products>.

You should see the list of products in JSON format.

Great work – you just created your very first NodeJS app using express as a web server.

### Part – 2 : Add a create product endpoint

### 

Let's add an endpoint to create a new product with validations. We'll use the express-validator package to handle input validations.

First, install the required packages:

npm install body-parser express-validator

Now, modify the app.js file to include the new endpoint and validations:

import express from 'express';  
import bodyParser from 'body-parser';  
import { body, validationResult } from 'express-validator';  
  
const app = express();  
const PORT = 3000;  
  
const products = [  
 { id: 1, name: 'Product 1', price: 100 },  
 { id: 2, name: 'Product 2', price: 200 },  
 { id: 3, name: 'Product 3', price: 300 },  
];  
  
// Middleware to parse JSON bodies  
app.use(bodyParser.json());  
  
// Get list of products  
app.get('/products', (req, res) => {  
 res.json(products);  
});  
  
// Create a new product with validation  
app.post('/products',  
 // Validation middleware  
 [  
 body('name').isString().withMessage('Name must be a string'),  
 body('price').isFloat({ gt: 0 }).withMessage('Price must be a positive number')  
 ],  
 (req, res) => {  
 const errors = validationResult(req);  
 if (!errors.isEmpty()) {  
 return res.status(400).json({ errors: errors.array() });  
 }  
  
 const { name, price } = req.body;  
 const newProduct = {  
 id: products.length + 1,  
 name,  
 price  
 };  
 products.push(newProduct);  
 res.status(201).json(newProduct);  
 }  
);  
  
app.listen(PORT, () => {  
 console.log(`Server is running on http://localhost:${PORT}`);  
});

### **Explanation:**

1. **Middleware to Parse JSON Bodies**: app.use(bodyParser.json()); is used to parse the JSON payload of incoming requests.
2. **Validation Middleware**: express-validator is used to validate the incoming request data. We check that name is a string and price is a positive number.
3. **Error Handling**: If there are validation errors, we return a 400 status code with the error details.
4. **Creating a New Product**: If the validation passes, we create a new product, add it to the products array, and return the new product with a 201 status code.

### **Running the App:**

Start the server:

npm start

To create a new product, you can use a tool like Postman to send a POST request to <http://localhost:3000/products> with a JSON body, for example:

{  
 "name": "Product 4",  
 "price": 400  
}

If the request passes validation, you should receive a response with the new product details.

The response should have a status code 201 means record is created with an Id generated.

If there are validation errors, you'll receive a response detailing the errors.

e.x. If you try this below payload -

{

"name": "Product 4",

"price": ""

}

You will receive a validation error with 400 status code, a message saying “Price must be a positive number".