Cubstart Lab 6

Building an API with Express

[start recording]

Administrivia

- HW 5: OpenWeatherMap is due tonight
    - Please ask questions on the Ed Megathread!
- HW 6: Quizlet-ish (Part 1) is due next week
- Lab next week will be optional / make-up session

Kahoot!

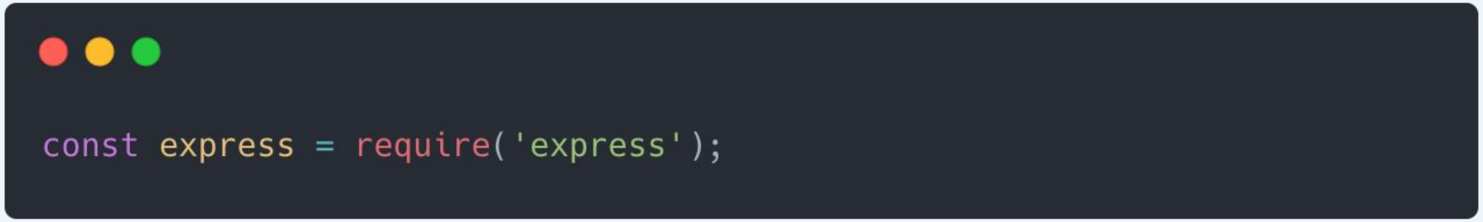
Recap

Node.js

- Server environment
- Allows the programmer to run JavaScript on a web server

Modules

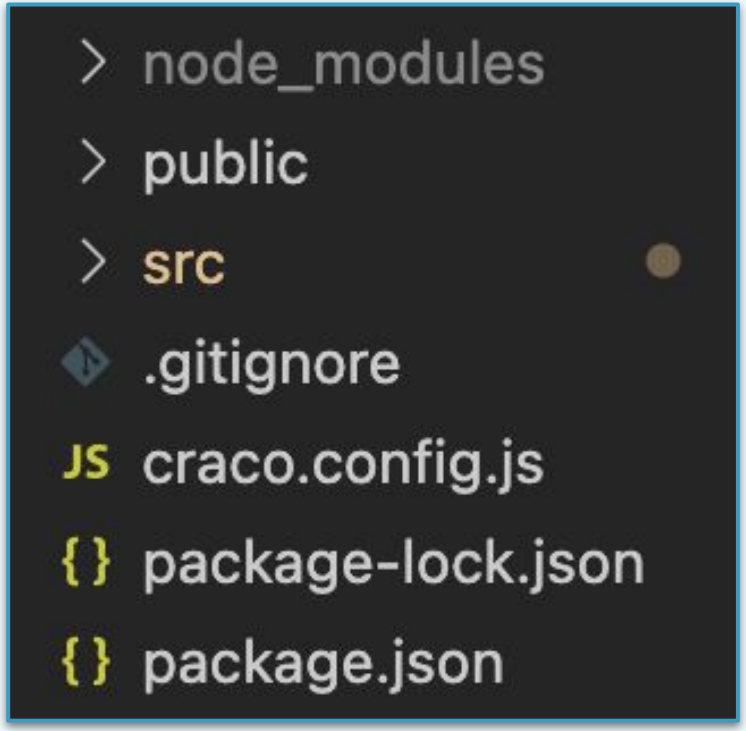
- Pieces of reusable code that you can include in your application without reinventing the wheel
- Examples of importing modules:
  - Data 8: import datascience (allows you to create tables, etc.)
  - Python: Import math, import pandas as pd, import numpy as np
  - Use functions from module: np.array(), math.maxint
- **require() over import**



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

npm - Node Package Manager

- Manage any dependencies (modules, libraries) within a project (frontend/backend)
- Lists all dependencies in a project in *`package.json`* and the actual code in a folder called *`node_modules`*
- **`npm init`**
  - `Creates package.json`
- **`npm install`**
  - `Installs module`
  - `Updates package.json`

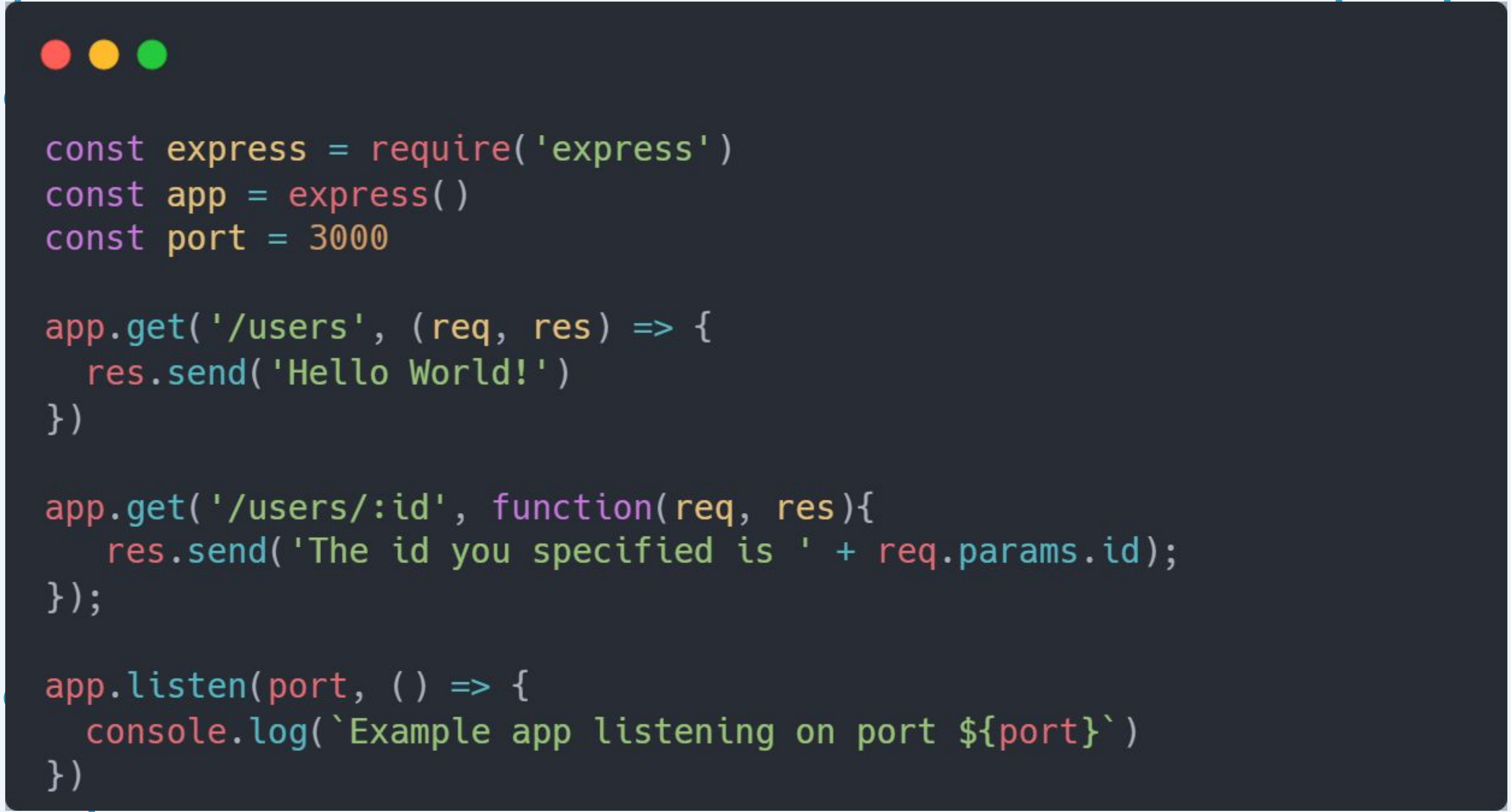

```
> node_modules
> public
> src
.gitignore
craco.config.js
package-lock.json
package.json
```

Express

- Minimalistic middleware framework build atop Node.js
- APIs that abstract away Node.js low-level functionality for HTTP methods
- Makes it easy to create a robust API

Middleware

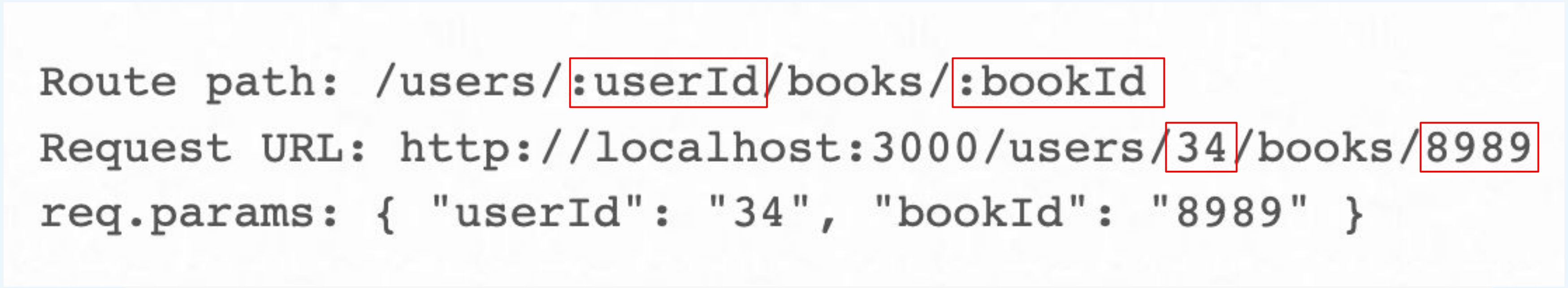
- Middleware literally means anything you put in the middle of one layer of the software and another
- **Access to the request object (req) and the response object (res)**



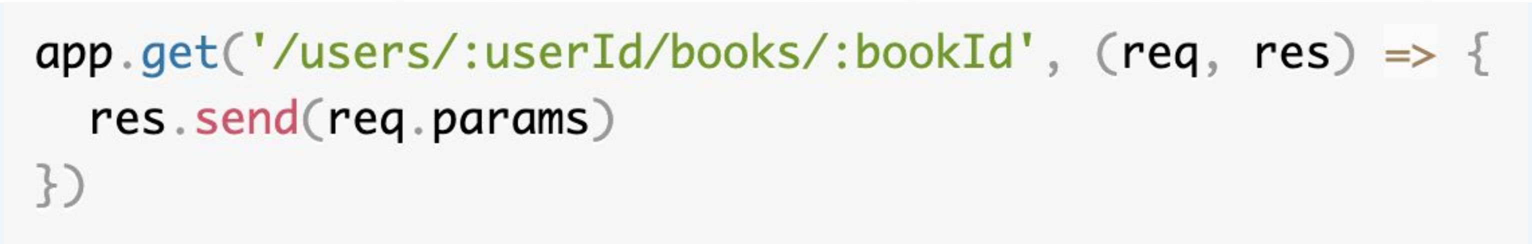
```
const express = require('express')
const app = express()
const port = 3000

app.get('/users', (req, res) => {
  res.send('Hello World!')
})

app.get('/users/:id', function(req, res){
   res.send('The id you specified is ' + req.params.id);
});

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

Route Parameters

- Route parameters are segments of the URL that capture values inputted by the client
  - Parameter names follow a colon
  - Captured values are put in the **req.params** object

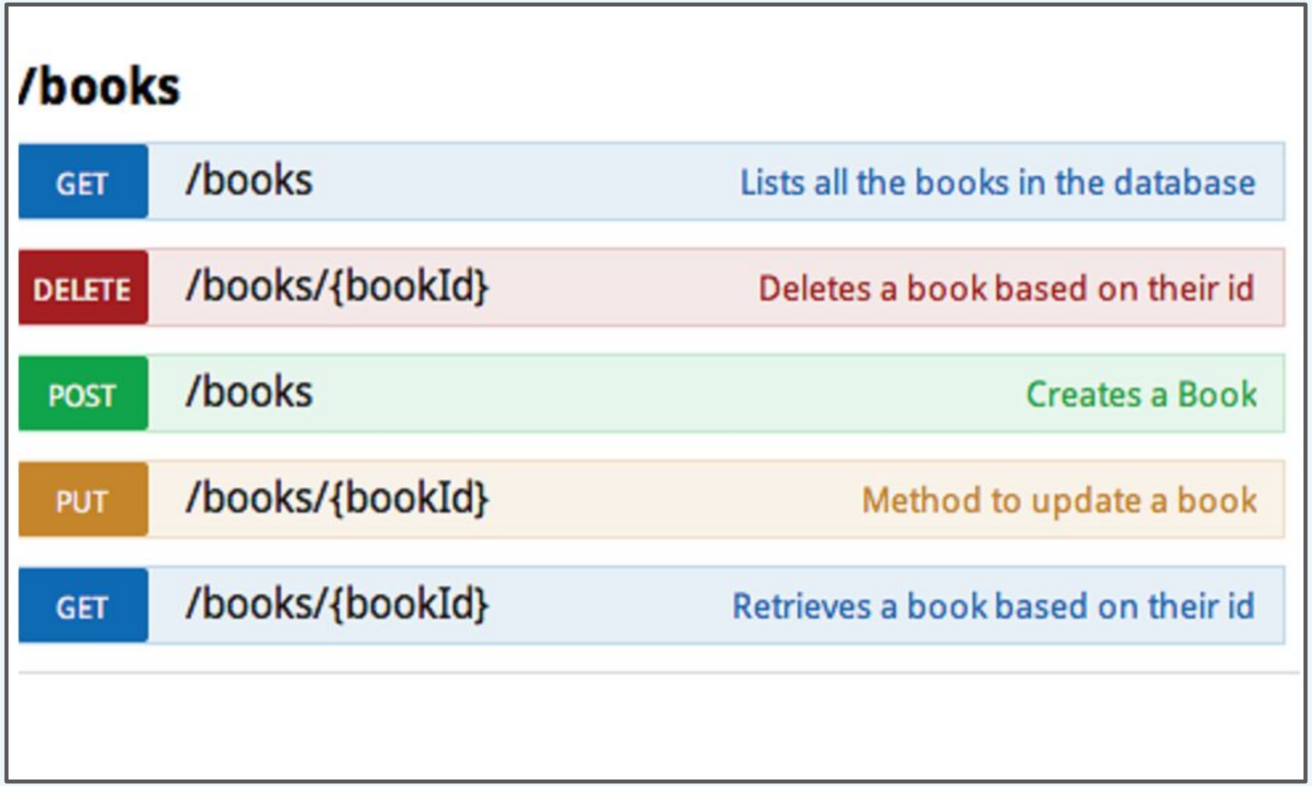


```
Route path: /users/:userId/books/:bookId
Request URL: http://localhost:3000/users/34/books/8989
req.params: { "userId": "34", "bookId": "8989" }
```



```
app.get('/users/:userId/books/:bookId', (req, res) => {
  res.send(req.params)
})
```

Building a Books API



/books

GET /books Lists all the books in the database

DELETE /books/{bookId} Deletes a book based on their id

POST /books Creates a Book

PUT /books/{bookId} Method to update a book

GET /books/{bookId} Retrieves a book based on their id

Step 0: Setup Project

Step 1: Create Server

Running a Server Locally

.listen() function

- Creates and starts a server
- Listens for connections (requests) on a specified port

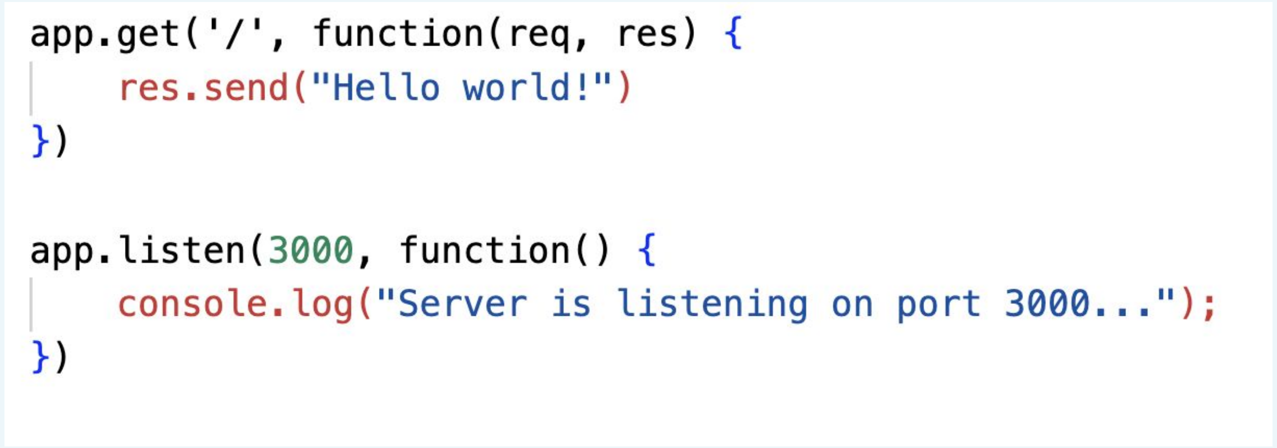
Port Number

Function that executes when the server starts running

```
app.listen(3000, () => {
      console.log("Listening on Port 3000”);
})
```

Running a Server Locally

Go to a browser and type in **http://localhost:3000/**. This is the “url” of your server. localhost represents your current computer that is running the server. When the page loads, it should display the “Hello world!”



```
app.get('/', function(req, res) {
    res.send("Hello world!")
})

app.listen(3000, function() {
    console.log("Server is listening on port 3000...");
})
```

Step 2: GET /books

Express Response Methods

The **res** object has a lot of built-in methods!

res.send(“Hello World!”)	Sends the HTTP response → “Hello World!”
res.json({ key: “value” })	Converts body to JSON, sends JSON response
res.sendStatus(200)	Sets the response HTTP status code to the parameter and sends the status as the response → “OK!”

Step 3: POST /books

Middleware: Body-Parser

- Used to parse the body of POST requests
- We'll be using JSON!

terminal: npm install body-parser

code: app.use(bodyParser.json())

Express Request Methods

The **req** object has a lot of built-in methods!

req.body	Parsed JSON body from body-parser

Client-Side

fetch()

- Syntax: fetch(apiURL, options) *second parameter optional

```
async function getData() {
    const response = await fetch(<... URL ...>)
    const data = await response.json()
    console.log(data)
}
```

Parses the response
body as JSON

POST requests with fetch()

You can send requests with **options** as the **second parameter**

```
const data = {“message”: “hello world!”}
const response = await fetch(“/api/message”, {
    method: “POST”,
    body: JSON.stringify(data),
    headers: {
        “Content-Type”: “application/json”
    }
})
```

server.js

```
const express = require('express')
const bodyParser = require('body-parser')
const app = express()

const books = ["Book 1", "Book 2"]

app.use(bodyParser.json())

app.get("/", (req, res) => {
res.json({ message: "hello world!" })
})

app.get("/books", (req, res) => {
res.json(books)
})

app.post("/books", (req, res) => {
books.push(req.body.name)
})
```

package.json

```
{
"name": "lab6",
"version": "1.0.0",
"description": "books API",
"main": "server.js",
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1",
"start": "node server.js"
},
"author": "",
"license": "ISC",
"dependencies": {
"body-parser": "^1.20.2",
"express": "^4.18.2"
}
}
```

[end recording]

Secret word:

Lab 6 attendance:
https://forms.gle/VZZYR7R9nCQySgAP9