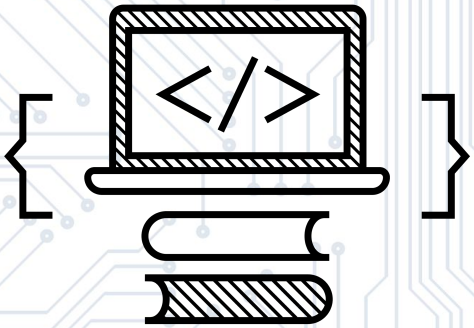


# Express Yourself

## Web Development Boot Camp

Lesson 11.2

# Do you feel this way sometimes?

“I feel like I’m just copying and pasting.”

# **About** “I feel like I’m just copying and pasting.”

**Response:**

Back-end code libraries mean you have to code less.

Often you are just copying and pasting “best practices” over and over again.

# Today’s Class

# Your Objectives Today

Before you walk out the door, you should:

Know how to create a generic Express server (copy and paste is fine).

Know how to create a basic Express GET route.

Know how to create an Express POST route.

Know what Postman is for.

Understand “conceptually” how to use AJAX to GET and POST data to an Express server.

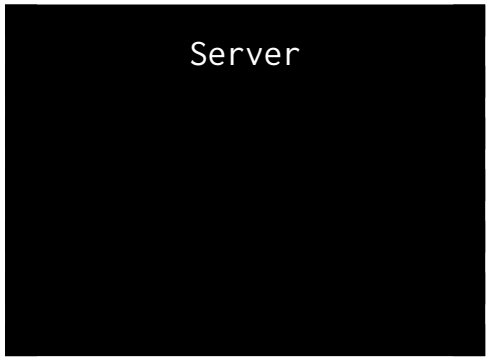
# A Moment to Refresh

# Activity: A Big Box

Take a few moments to answer the following questions with the person next to you:

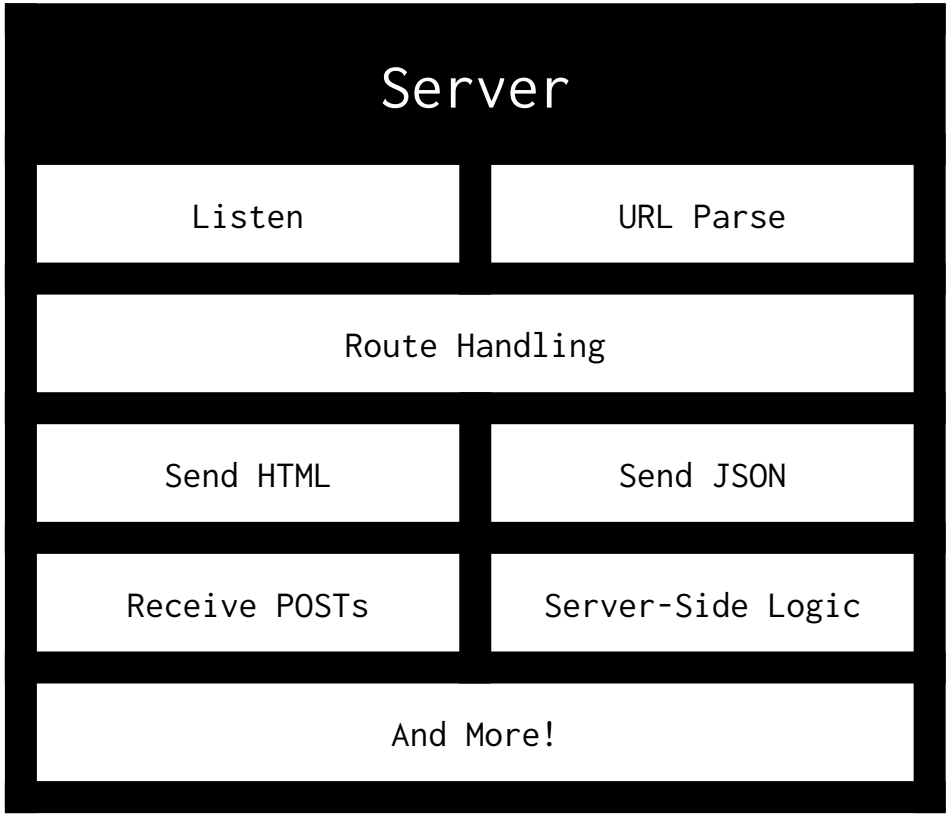
01 How does this box represent a server?

02 What modules or “sub-boxes” commonly exist inside?




# Inside the Box, and More!

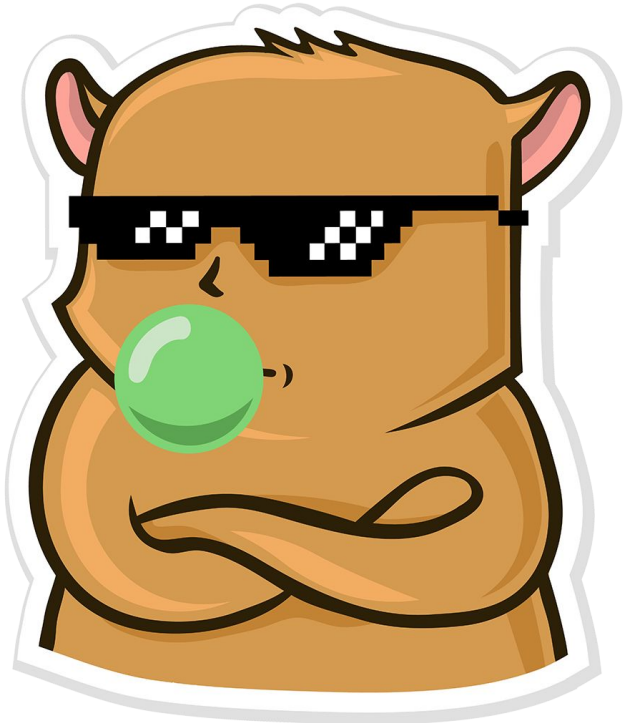
- In a way, servers can be represented as hardware boxes with modules of server-side code contained inside.
- Servers typically contain code for tasks such as listening, parsing URLs, route handling, and more.



# Server Routes

# Remind Me Again

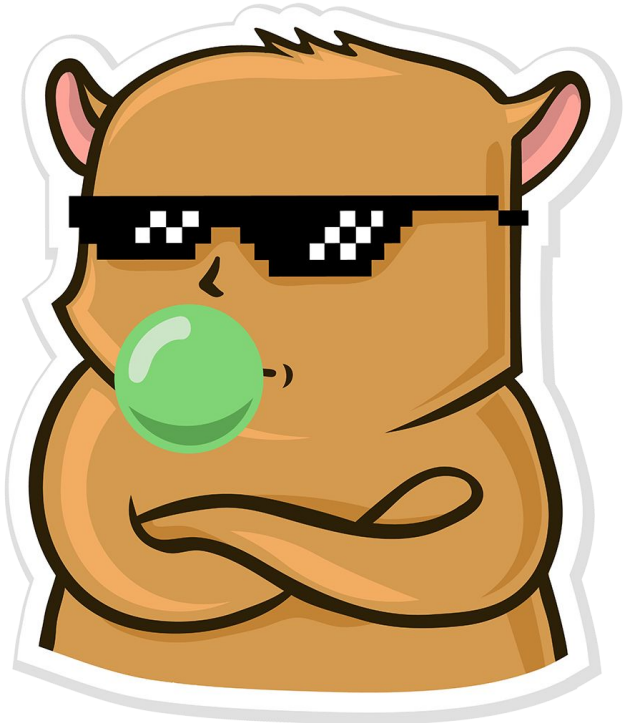
What is a **route**?



A **route** is a path to and an action to to perform *on* a resource.

# Sounds great.

What is a **resource**?



# What is a resource?

## A **resource** is anything your server provides

- A static file (e.g., html file, a pdf, an image, a CSS file, a Javascript file)
- Server-side rendered HTML documents
- API endpoint for data (e.g., a NYT article search returns JSON)
- API endpoint for a service (e.g., upload a PPTX for validation)

A **URL** is a:

**U**niversal

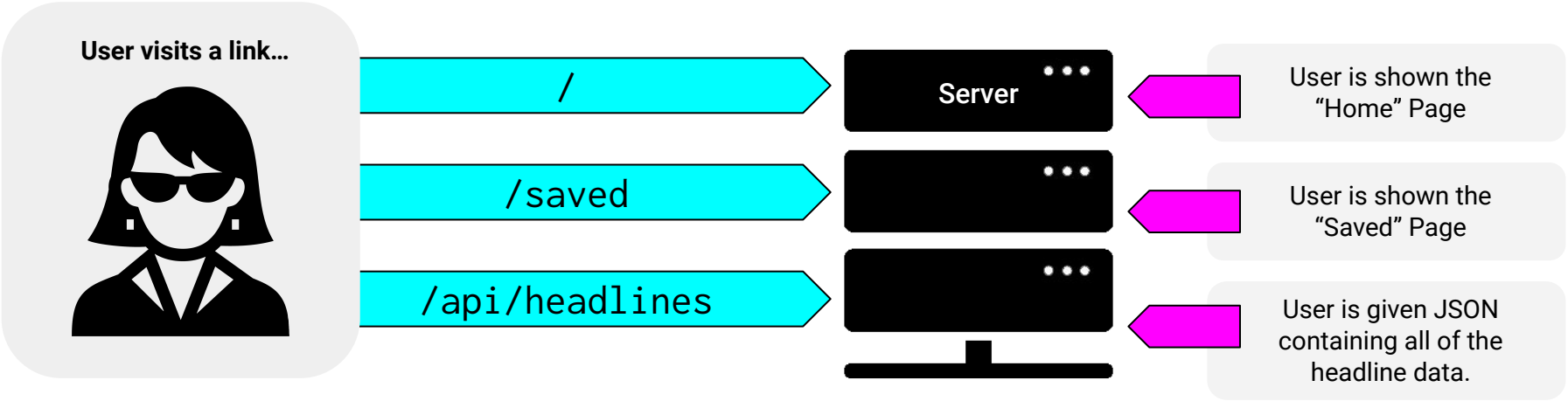
**R**esource

**L**ocator

# Quick Example

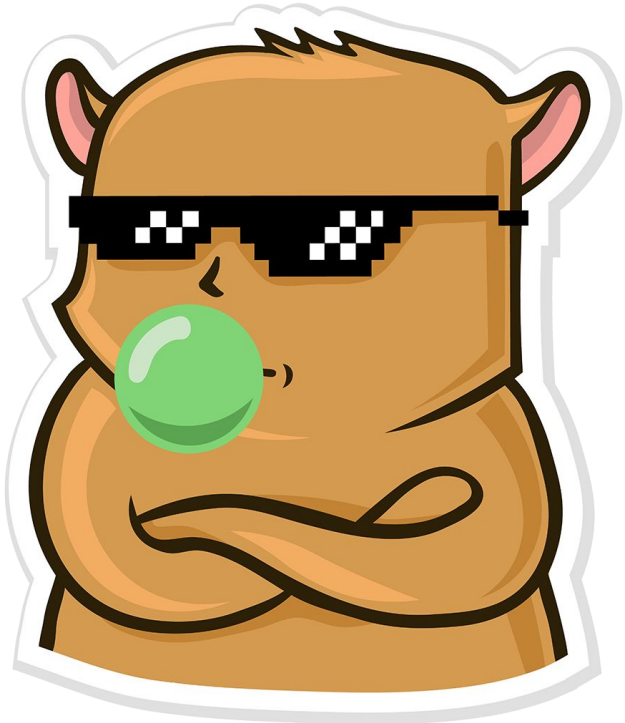
nyt-mongo-scraper.herokuapp.com

# Route Paths: resource examples



# Sounds great.

What did you mean by an **action**?



# HTTP Methods

# HTTP Methods

Methods describe an **action** to perform on a resource.

**GET**: retrieve a resource. Modify the request with **parameters**.

**POST:** create a new resource. Submit data through the request **body**.

**PUT**: modify an existing resource. Submit **parameters** and **body**

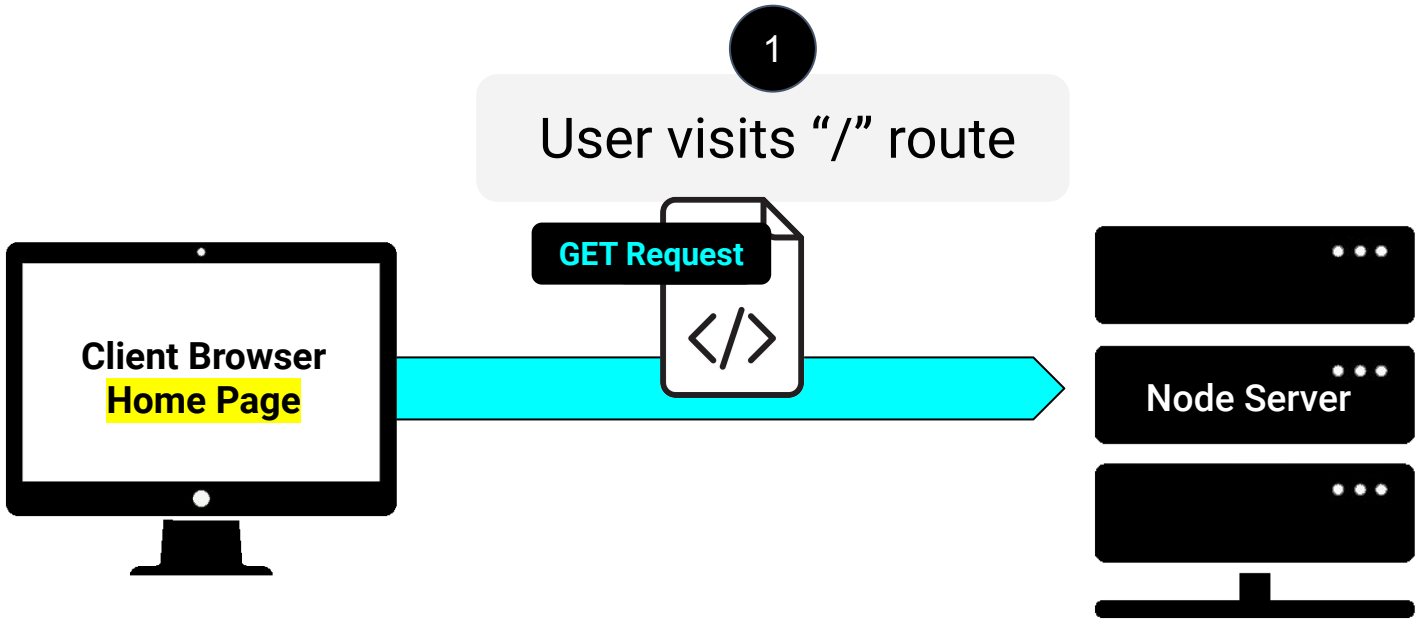
DELETE: remove an existing resource. Submit **parameters.**

# 2 parts to a route

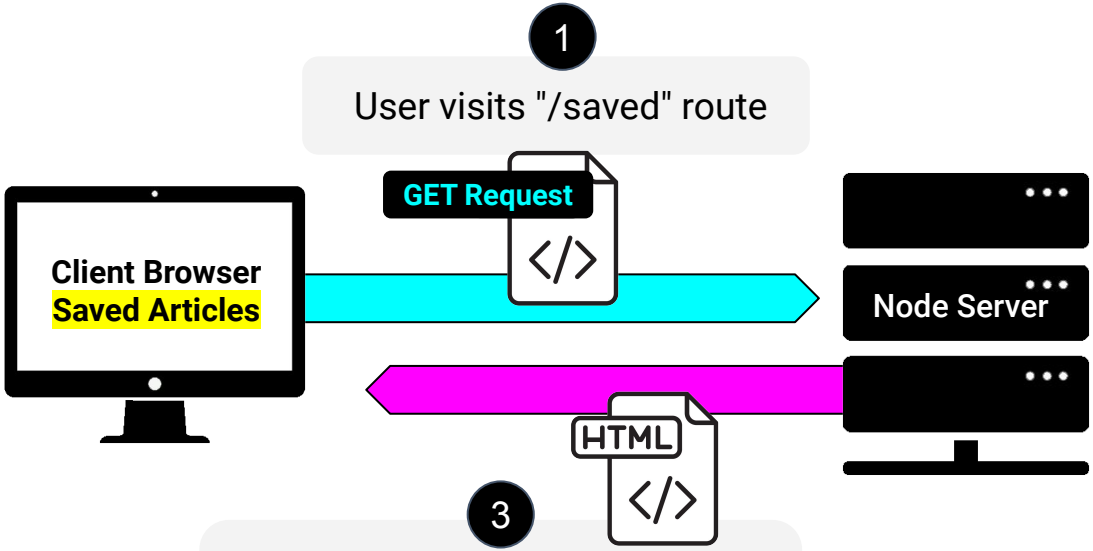
1. URL (e.g., `https://myapp.com/api/user`)

2. Method
   `GET`
   `POST`
   `PUT`
   `DELETE`

# Client-Server Communication (GET)



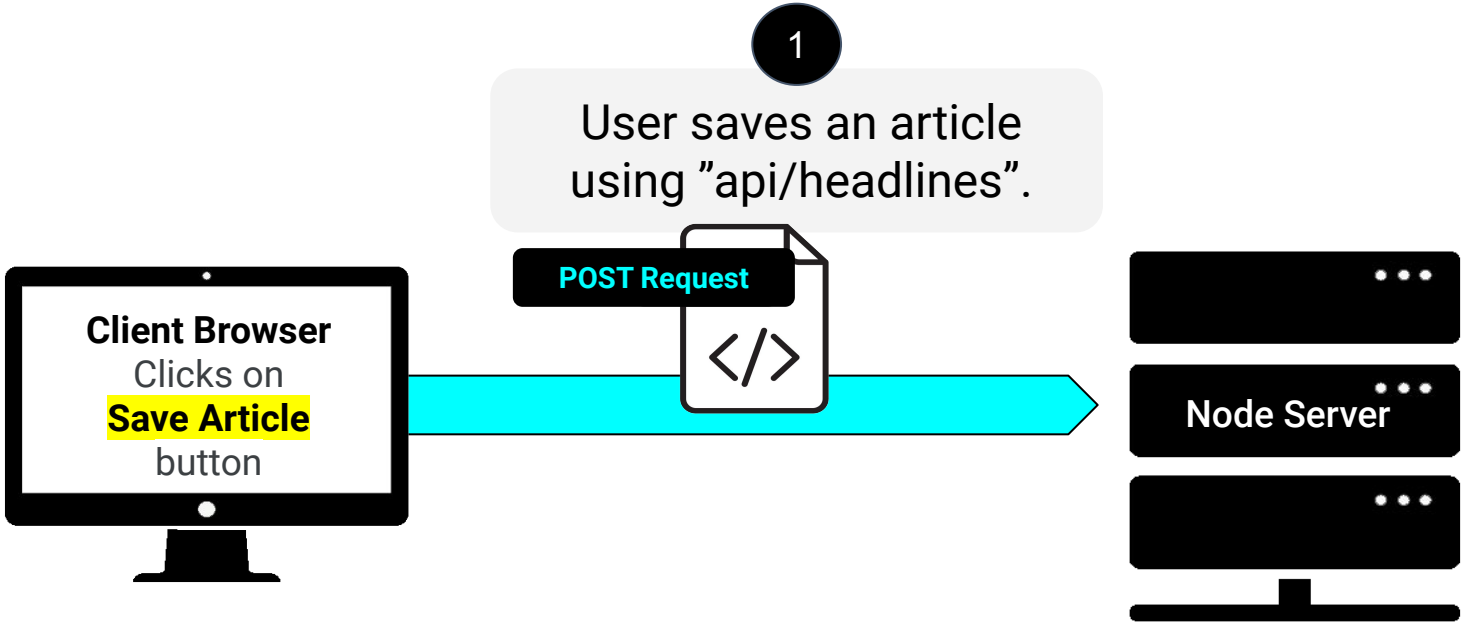
# Client-Server Communication (GET)



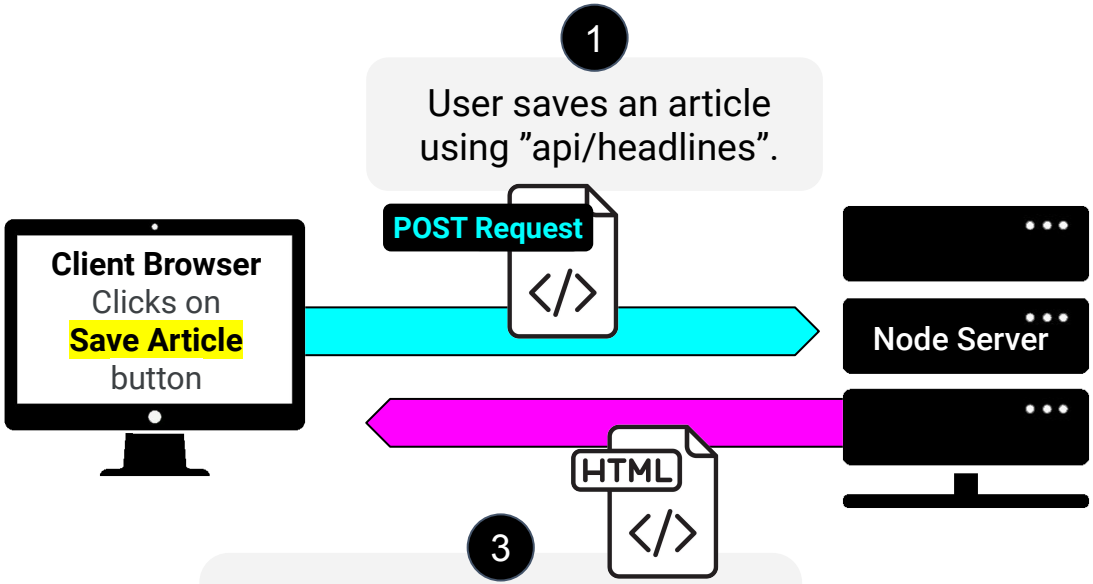

2

Request triggers the code in the server route. The server then finds the relevant HTML content and data.

# Client-Server Communication (POST)



# Client-Server Communication (POST)



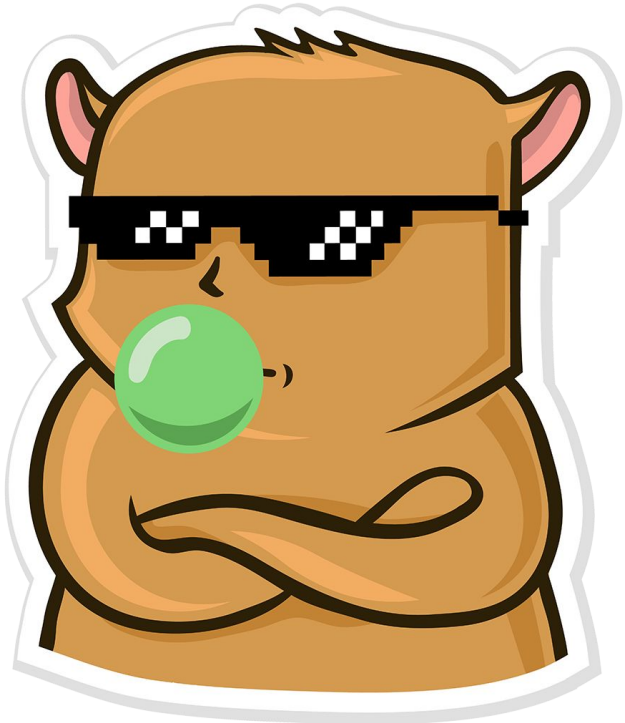

2

Request triggers the code in the server route. The server then sets the article as “saved” in the database.

# Express

# Remind Me Again

What is the `http` module?



# Node `http` module

is a standard Node library that can create a bare-bones web server.

# Review: Node.js http module

```
const http = require(“http”);
const PORT = 8080;

function handleRequest(request, response) {
  response.end("It works! Hi there!!!!!!: " + request.url);
}

const server = http.createServer(handleRequest);

server.listen(PORT, function () {
  console.log("listening on port " + PORT);
});
```

# What's wrong with just using the `http` module?

Nothing really. But...

You have to handle routes with a lot more code (two parts to a route)

Parsing the request body is more work

Sending files is more work

One tends to end up needing a framework to organize the code (i.e., DRY)

# Express.JS

Express is a web framework for Node that makes creating server-side code much simpler.

# <Time to Code>