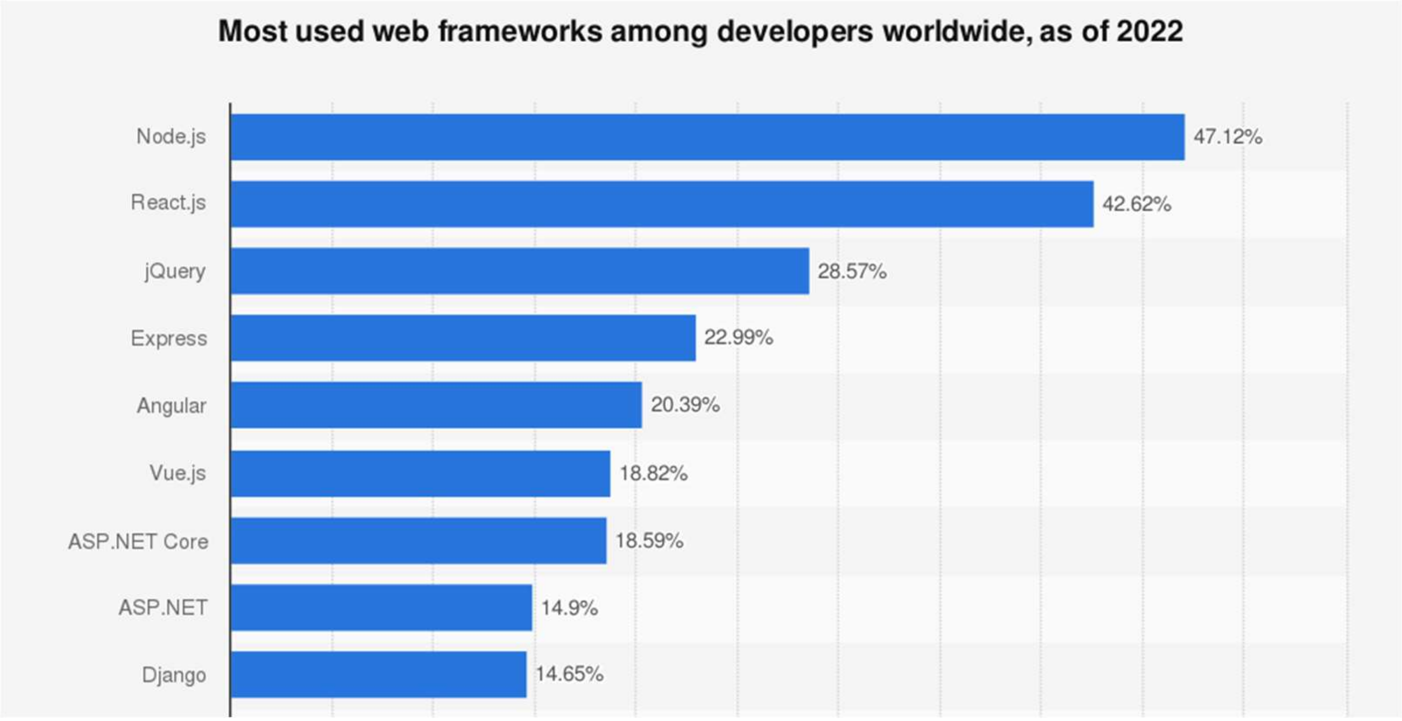
Introduction to NodeJS

What is NodeJS?

- NodeJS is a runtime environment for executing JavaScript code on a server
  - Outside of a browser
- Used for Backend development

NodeJS popularity



Browsers' JavaScript Engines

- Firefox
  - SpiderMonkey
- Edge
  - Chakra
- Opera
  - Carakan, Blink/V8
- Chrome
  - V8

JavaScript execution

- Before 2009, only in Browsers
- Since then, Engine V8 makes it possible to run JavaScript outside of a browser
  - NodeJS

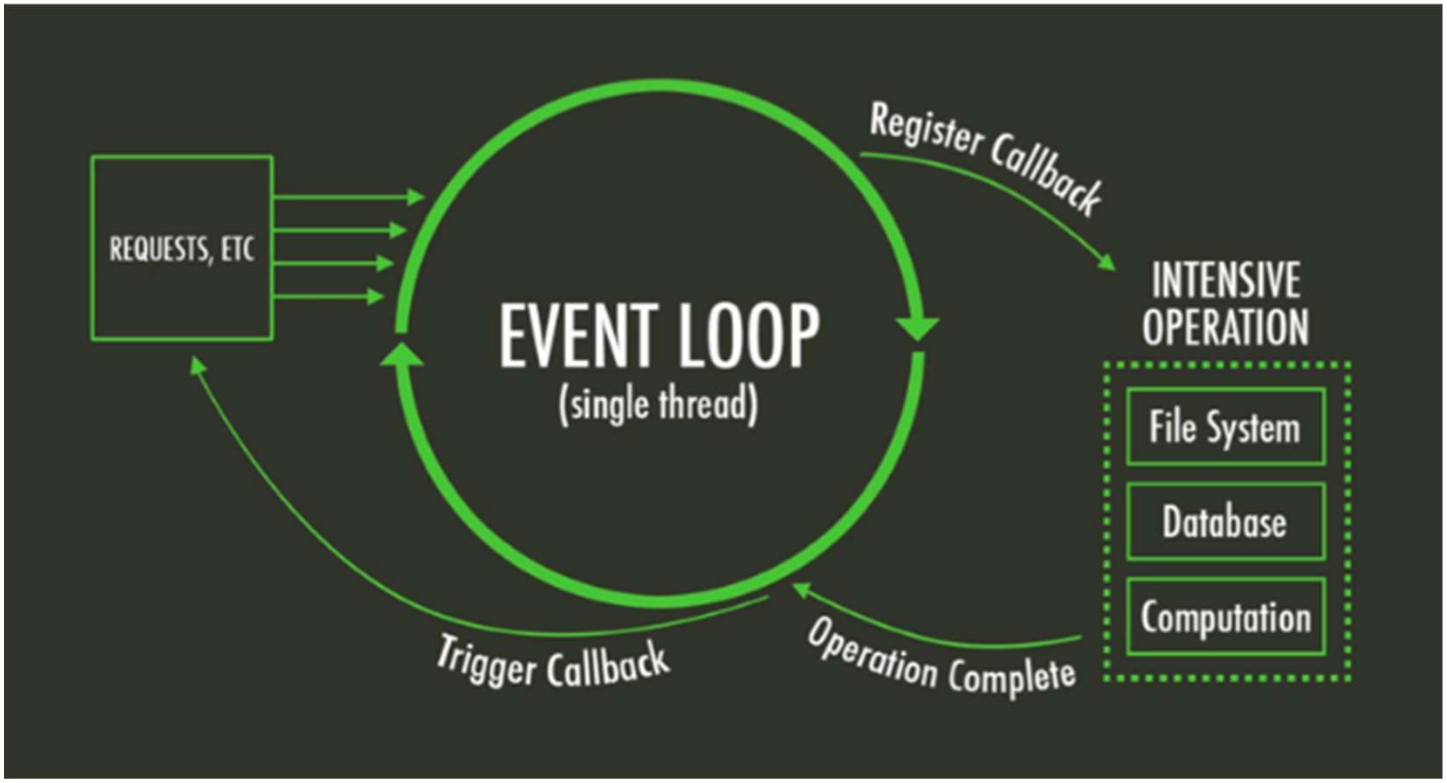
JavaScript outside Browser

- It does not have access to Browser’s specific elements
  - E.g., window, document, document.getElementById()…
- It has things that are not available in Browsers
  - module, global…
  - Access to file system
  - Access to Operating System
- The programming language is still JavaScript!

Blocking vs. Non-blocking

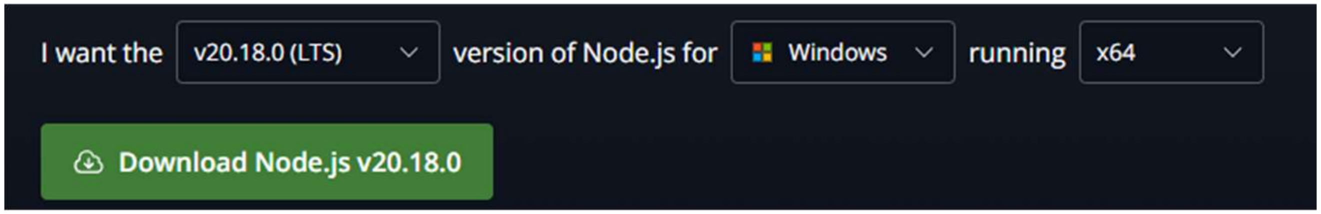
- Synchronous vs. Asynchronous
- Out-of-the-box, NodeJS is Asynchronous (i.e., non-blocking)
  - When waiting for an I/O to complete, do something else
  - When I/O is done, notify the process
  - The process then does something
  - Events queue
- NodeJS is good for I/O-bound apps
  - Not CPU-bound apps

Asynchronous in Node.js



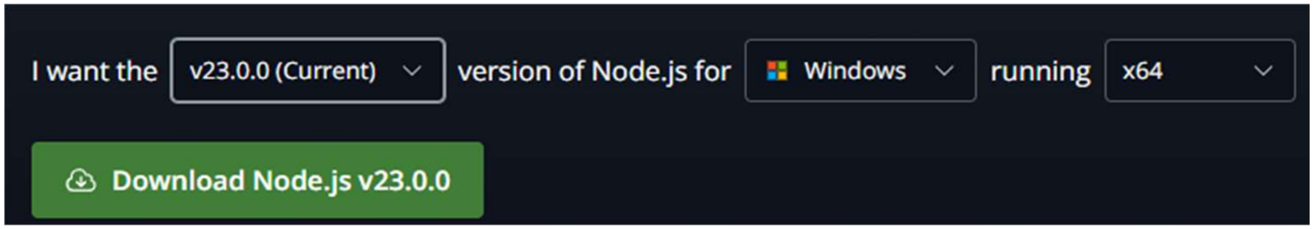
Setup the Environment

- Installing Node.js (nodejs.org)
  and npm (Node Package Manager)



I want the v20.18.0 (LTS) version of Node.js for Windows running x64

Download Node.js v20.18.0

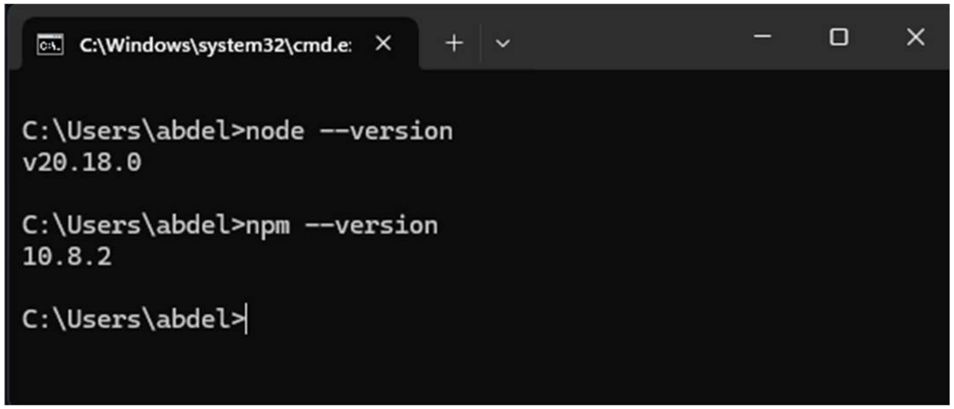


I want the v23.0.0 (Current) version of Node.js for Windows running x64

Download Node.js v23.0.0

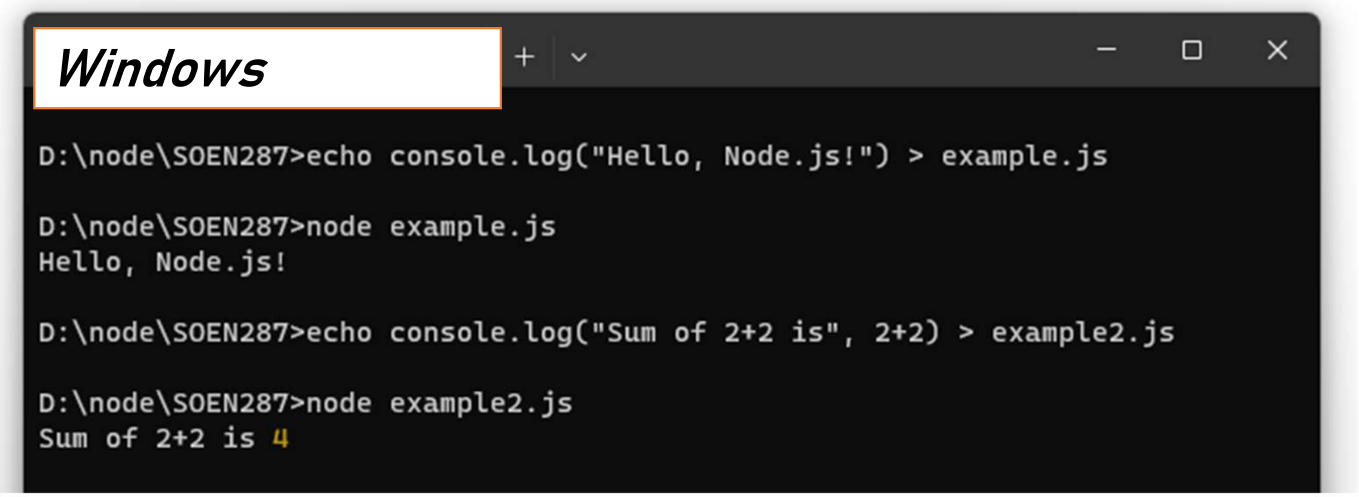
Setup the Environment

- Check node and npm version
  - Open a terminal and run following commands
    - node --version
    - npm --version

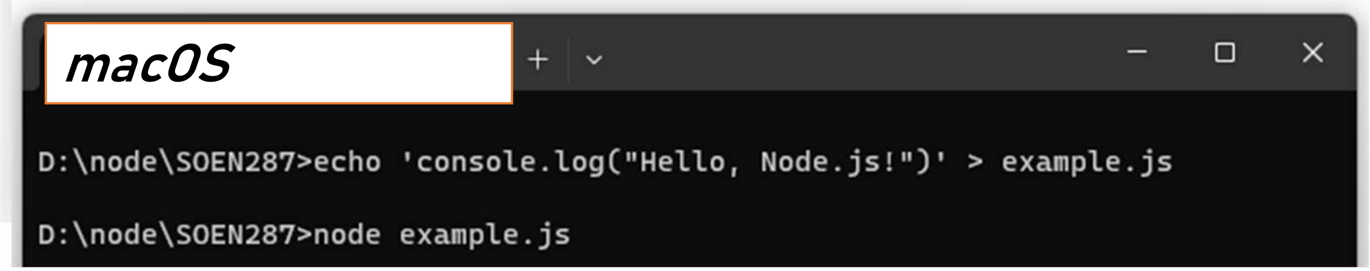


```
C:\Users\abdel>node --version
v20.18.0

C:\Users\abdel>npm --version
10.8.2

C:\Users\abdel>
```

Hello World Example

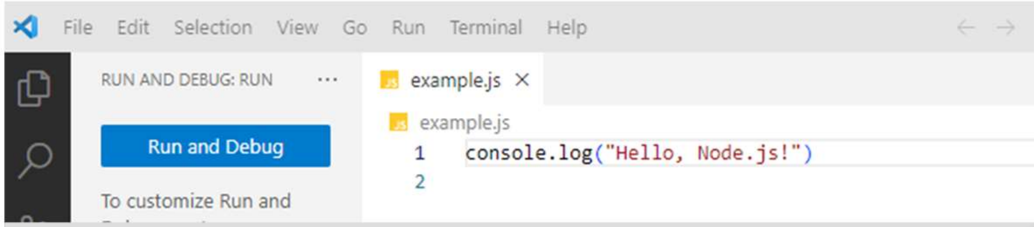


```
D:\node\SOEN287>echo console.log("Hello, Node.js!") > example.js

D:\node\SOEN287>node example.js
Hello, Node.js!

D:\node\SOEN287>echo console.log("Sum of 2+2 is", 2+2) > example2.js

D:\node\SOEN287>node example2.js
Sum of 2+2 is 4
```



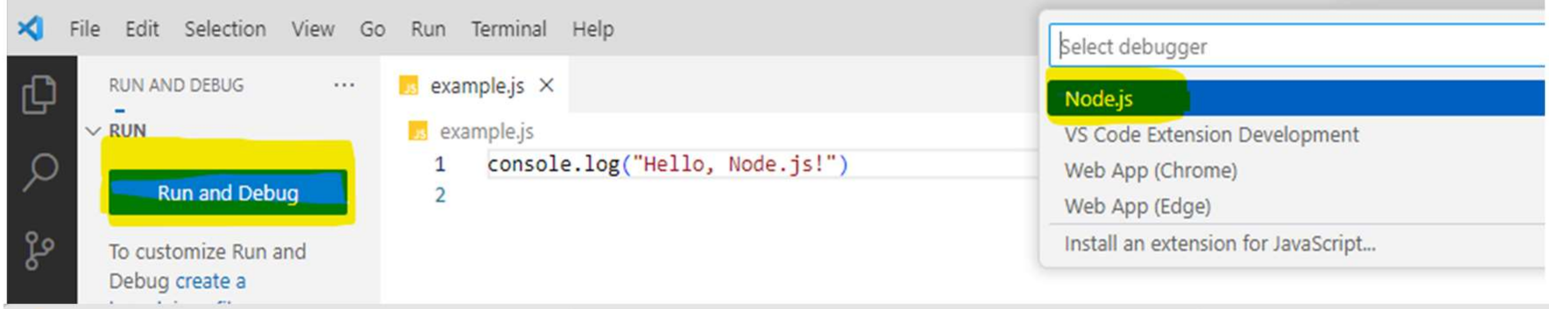
```
D:\node\SOEN287>echo 'console.log("Hello, Node.js!")' > example.js

D:\node\SOEN287>node example.js
```

VSCode Example



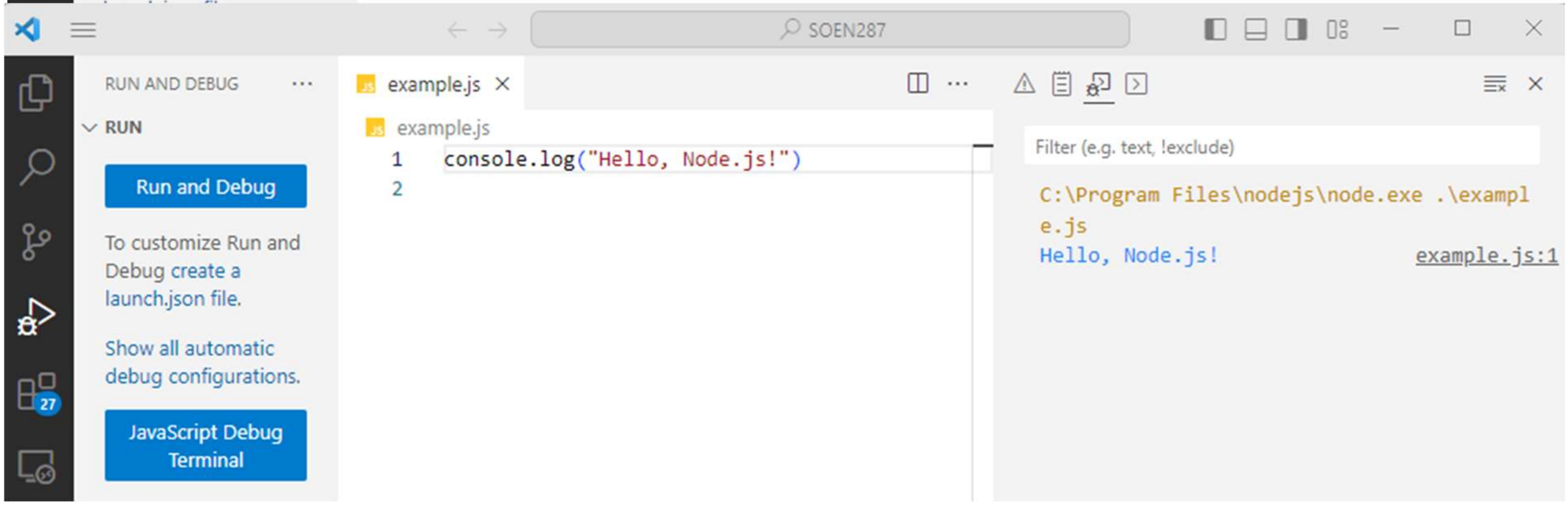
```
console.log("Hello, Node.js!")
```

Step 1



```
console.log("Hello, Node.js!")
```

Step 2



```
console.log("Hello, Node.js!")
```

```
C:\Program Files\nodejs\node.exe .\exampl
e.js
Hello, Node.js!                      example.js:1
```

Step 3

First example

- Make sure NodeJS is installed
- Create a file app.js
- Write the following code into the file:

```
function displayGreeting(){
     console.log("Hello World");
}
displayGreeting();
```

- In terminal:
  - Make sure you are in the same folder as file app.js
  - Run: node app.js

Modules

- Modularity is one of the sought-for software property
- Modularity: define the application in terms of independent yet (partially) interacting modules
  - e.g., many classes/packages in Java
- In NodeJS, you can define your own modules
- Also, NodeJS comes with a lot of modules
  - For a lot of things you might want to do
    - Access OS, access files…

Modules…

- Functions and variables defined in a JavaScript file/module are only available in that module
- Unless you:
  - export them from that file and
  - import them in the file/module where you want to use them

Modules…

- Consider a file file1.js, in the same folder, with the following code:

```
const pi = 3.14;

function computeCircleCircumference(radius){
     return 2 * pi * radius;
}
```

Modules…

- If in file app.js, we write and run:

```
console.log(computeCircleCircumference(3));
```

- You will get an error since the `computeCircleCircumference` function is not available in app.js context

How to export

```
const pi = 3.14;

function computeCircleCircumference(radius) {
  return 2 * pi * radius;
}

module.exports.computeCircleCircumference =
computeCircleCircumference;
```

How to import and use

```
const calculator = require("./file1");

console.log(calculator.computeCircleCircumference(3));
```

Check exports of a module

- From the module, execute:

```
console.log(module.exports);
```

File name and folder name

```
console.log(__filename); // two _
console.log(__dirname);
```

Sync vs. Async example

```
// Sync
const fs = require("fs");

const content1 = fs.readFileSync("./file1.js");
console.log(content1.toString());
```

Sync vs. Async example…

```
// Async
const fs = require("fs");

fs.readFile("./file1.js", function (e, c) {
  console.log(c.toString());
});
```

Async and Events

- An event can be raised, if required
- An event signals that something happened
  - e.g., a connection, a request
- A function can be attached to an event
  - When the event occurs, the function executes

Events example

```
const EventEmitter = require("events");
const eventEmitter = new EventEmitter();

// Listen for the event
eventEmitter.on("eventName", eventHandler);

// Emit the event
eventEmitter.emit("eventName");

function eventHandler() {
  console.log("Event received");
}
```

Important

- **The listener must be defined before the event is emitted!!**

Events example…

```
const EventEmitter = require("events");
const eventEmitter = new EventEmitter();

// Listen for the event
eventEmitter.on("eventName", eventHandler);

// Emit the event
eventEmitter.emit("eventName", 10);

function eventHandler(arg) {
  console.log("Event received " + arg);
}
```

The http module

- One of the most important module in NodeJS
- Provides functionalities to listen and serve HTTP requests

The http module…

```
const http = require("http");
const PORT = 5000;

const server = http.createServer();

// Low level code, dealing with sockets
server.on("connection", (socket) => {
  console.log("Connected!");
});

server.listen(PORT, () =>
      console.log("listeneing on port " + PORT));
```

The http module…

```
const http = require("http");
const PORT = 5000;

// Better, but still low level
const server = http.createServer((request, response) => {
  response.write("From the <b>server</b>");
  response.end(); // Finish the response
});

server.listen(PORT, () =>console.log("listening on port " + PORT));
```

nodemon

- If your server is listening and you make changes to the source file, the changes are not visible yet
  - You need to manually restart the server (stop then start)
  - Or you can use nodemon
- You might need to install it:

```
npm install -g nodemon
```

- On Windows, you might need to enable powershell scripting and developer mode
  - You might even need to restart your computer
- Start your app with `nodemon` rather than `node`

```
nodemon app.js
```

Set HTTP headers

- If needed
- e.g.:

```
response.writeHead(200, { 'Content-Type': 'text/plain' });
```

Or

```
response.writeHead(200, { 'Content-Type': 'text/html' });
```

The http module…

```
const http = require("http");
const port = 5000;

// Better, but still low level
const server = http.createServer((request, response) => {
  response.writeHead(200, { 'Content-Type': 'text/html' });
  response.write("From the <b>server</b>");
  response.end(); // Finish the response
});

server.listen(PORT, () => console.log("listening on port " + PORT));
```

Different “Routes”

```
const http = require("http");
const PORT = 5000;
// Better, but still low level
const server = http.createServer((request, response) => {
  if (request.url === "/") {
    response.write("From the root /");
  }
  if (request.url === "/abc") {
    response.write("From /abc");
  }
  response.end();
});
server.listen(PORT, ()=> console.log("listening on port " + PORT));
```

Issues

- As routes increase, so does the linear arrangement of the function
- We might need to make the difference between post/get…
  - On all routes!!!

Express

- A framework for building web apps
- Provides a robust set of server-side features
- Simplifies dealing with multiple routes and HTTP methods

Express…

- Install it first:

```
npm install express
```

Express…

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

app.get('/', (req, res) => {
  res.send("From root / with get");
});

app.post('/abc', (req, res) => {
 res.send("From /abc with post");
});

const PORT = 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

Express – Handling forms

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Document</title>
  </head>
  <body>
    <form method="get" action="/formHandler">
      <input type="text" name="inputName" />
      <input type="submit" />
    </form>
  </body>
</html>
```

** Put this code in a file called form.html*

Express – Handling forms…

```
const express = require("express");
const app = express();

app.use(express.urlencoded({ extended: false }));

const PORT = 5000;

app.get("/", (request, response) => {
  response.sendFile(__dirname + "/form.html");
});
```

Express – Handling forms…

```
app.get("/formHandler", (request, response) => {
  const input = request.query.inputName;
  // Do something with the input
  // Notify the user
});

app.post("/formHandler", (request, response) => {
  const input = request.body.inputName;
  // Do something with the input
  // Notify the user
});

app.listen(PORT, () => {
  console.log("Listening");
});
```

Upload a file to server

- Add the following to the form:

```
enctype="multipart/form-data" as a form property (after action="")
```

```
<input type="file" name="logoFile"/>
```

Server

- You need a middleware that handles “multi-part” bodies
- To extract the file from the multi-part form
- There are many
  - Here is an example with: multer

Using multer

- npm install multer

```
const multer = require('multer’);
```

Using multer…

```
const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, "uploads/logo/");
  },
  filename: function (req, file, cb) {
    cb(null, "Logo-" + file.originalname);
  },
});

const upload = multer({ storage: storage });
```

Using multer…

```
app.use(express.urlencoded({ extended: false }));

app.post("/formHandler", upload.single('logoFile'), (req, res) => {
  const input = req.body.inputName;

  // Optional, as part of input validation
  if (!req.file)
       res.send("You did not submit a file.");
  else
       res.send("File received. Thank you.");
});
```

Practice

- Apply what was covered above for one of the forms in your project