Web Servers 101

LIGHTHOUSE LABS

AGENDA

What is Node.js

What is a web server

Creating a web server with vanilla Node.js

Create a web server with Express.js

Why was the JavaScript developer sad?

Because he didn't Node how to Express himself

What is Node.js

slido

What is Node.js? Select all that are true.

ⓘ Start presenting to display the poll results on this slide.

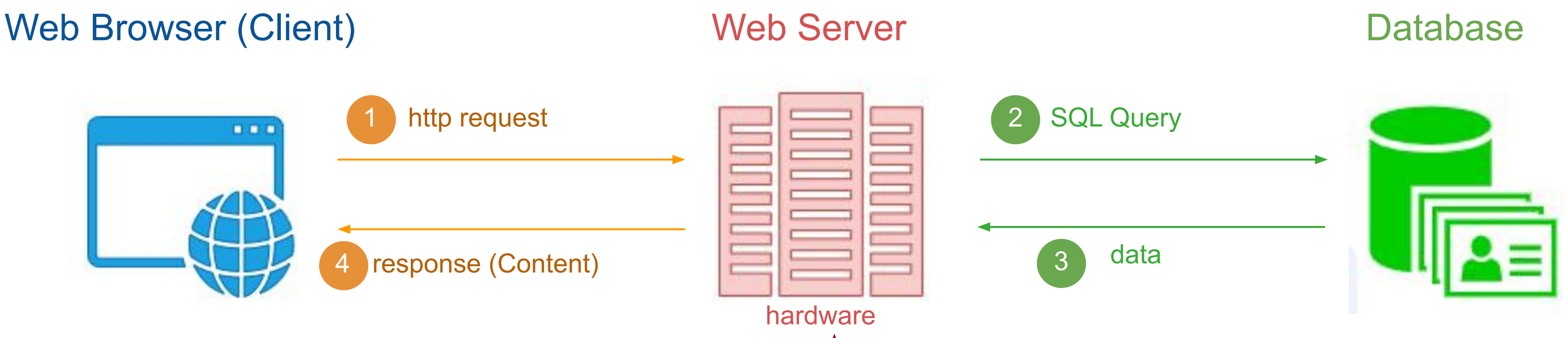
What is a web server?

LIGHTHOUSE LABS

What is a web server? Select all that are true.

ⓘ Start presenting to display the poll results on this slide.

Web 3-Tier Architecture



Web Browser (Client)

Web Server

Database

1 http request

4 response (Content)

2 SQL Query

3 data

hardware

Primary function is to render HTML

Accepts http requests from clients and sends back the resource

Keeps the data persistent. Allows CRUD operations with SQL

software

NodeJS / Express

ExpressJS allows to build Web app with built-in capabilities.

- View templates
- Routing system
- Middleware

Requests

- A web browser sends requests to a web server using **http**

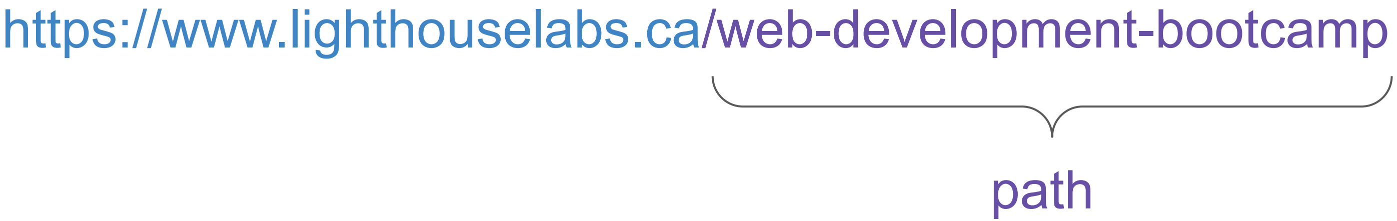
request = http verb + path (resource)

- 4 http verbs:

1. GET (Read)
2. POST (Create)
3. PUT (Update)
4. DELETE (Delete)

Request

- The path indicates the resource



https://www.lighthouselabs.ca/web-development-bootcamp

path

Response

- The web server sends back a response

resource (HTML document) + status code

Requests and Responses

- Requests and Responses contain 2 parts:

Headers + body

- **Headers**:
  - request or response meta-information
- **Body**:
  - Information submitted (Form) in the case of a request
  - Response sent by the server (HTML document)

Demo

LIGHTHOUSE LABS

Creating The Server

```
const http = require('http');
const port = 3000;

const server = http.createServer((request, response) => {



});

server.listen(port, () => console.log(`Server is listening on port ${port}`));
```

1 We need http core library

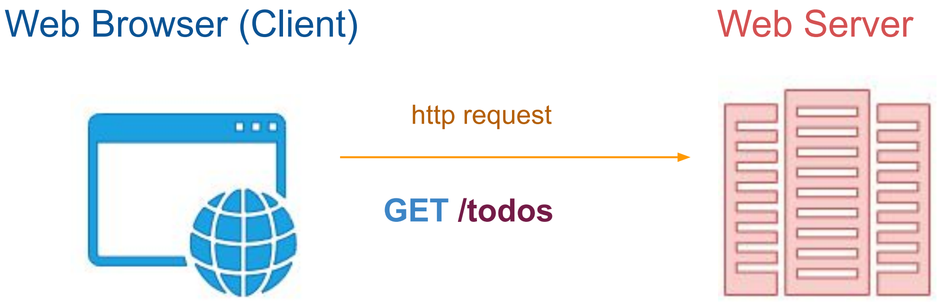
2 Setting a port for the server

3 Create a http server

4 Have the server listening for incoming http requests

5 request listener function handles the **request** and the **response**

Web Server Routing



Web Browser (Client)

Web Server

http request

GET /todos

The http request received is stored on the server in a **request object {...}**

2 key properties:

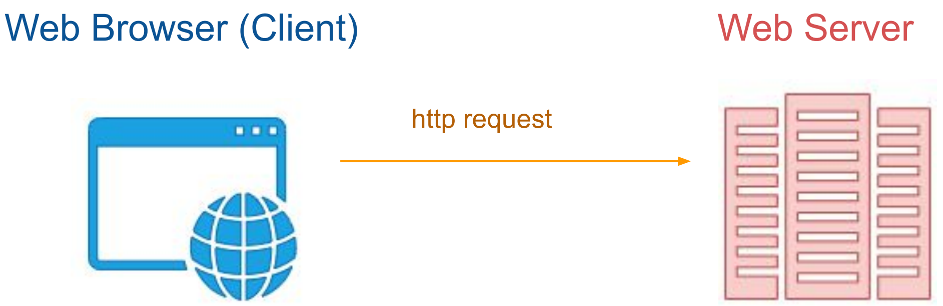
1. **method**
2. **url**

In this example:

method = GET
url = ‘/todo’

Route = Method + Path (url)

Web Server Routing



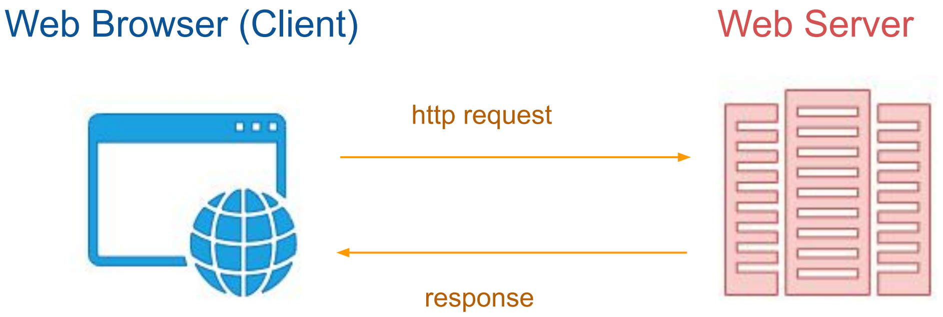
Routes

GET / => Display Homepage

GET /todos => Display todos

Not any of these => Display 404

Sending a Response



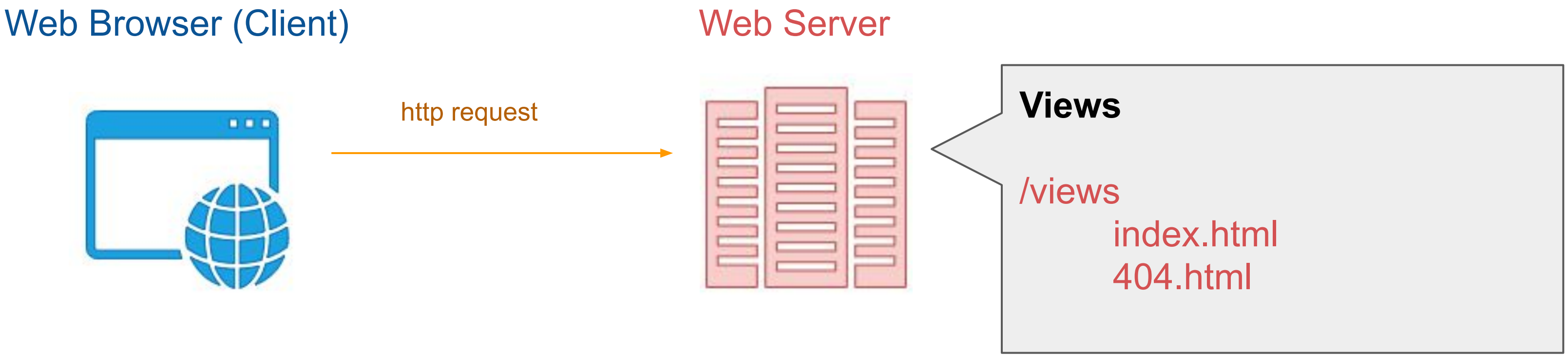
Response

response.writeHead => set elements in the headers of the response

response.write => Send back content to the client

response.end => Terminate the response

Views



Web Browser (Client)

http request

Web Server

Views

/views
index.html
404.html

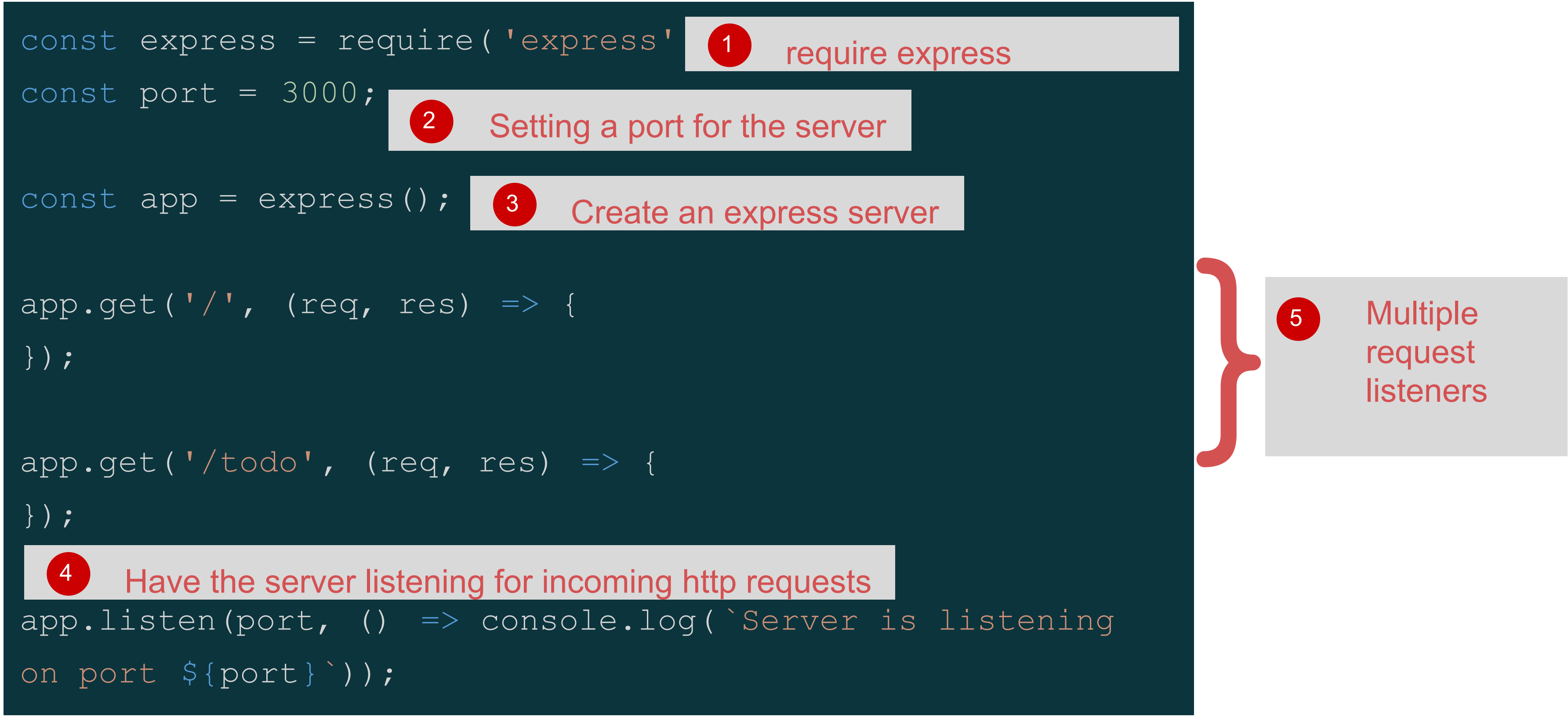
- We call views the different HTML pages that the server can send back to the browser
- The files are located in the views folder on the server

ExpressJS

Fast, unopinionated, minimalist web framework for Node.js. It provides:

- Routing system
- View templates (ejs)
- Middleware

Creating The Server



```
const express = require('express'
const port = 3000;

const app = express();

app.get('/', (req, res) => {
});

app.get('/todo', (req, res) => {
});
app.listen(port, () => console.log(`Server is listening
on port ${port}`));
```

1 require express

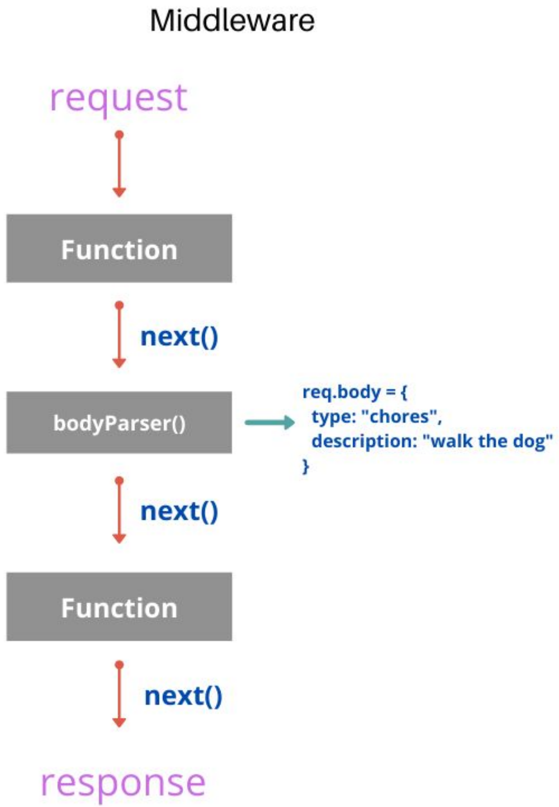
2 Setting a port for the server

3 Create an express server

4 Have the server listening for incoming http requests

5 Multiple request listeners

Middleware



```
req.body = {
  type: "chores",
  description: "walk the dog"
}
```

- *Middleware* is code (in the form of functions) that runs between the incoming request and the outgoing response

- ExpressJS on its own has very little functionality; it is through the use of middleware that the real power of Express comes out

Middleware

There are many popular middleware packages available to us via NPM. For example:

- body-parser: Parses the body of the incoming request, converting it to a JS object and attaching it to the request object (accessible with req.body)

- cookie-parser: Parses the cookie header, converting it to an object and attaching it to the request object (accessible with req.cookies)

- morgan: A logger that logs all requests/responses to the web servers console

What are the benefits of Express?

ⓘ Start presenting to display the poll results on this slide.

Questions?

LIGHTHOUSE LABS