

COMP4021
Internet Computing

Building a Simple Server

Gibson Lam

Node.js and Express

- In this presentation, we will look at using the Express module to build a simple web server
- We assume that you have successfully installed Node.js and the Express module in your system
- We will build:
 - a server that allows access of static files
 - a server that returns JSON responses

Using Express

- To start using Express, you import the module and then create an Express app:

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

- You do things using this app, for example, start a web server like this:

```
app.listen(8000);
```

Starting a Web Server

- A web server has been started using this code:

```
app.listen(8000);
```



*A port number that the
web server is listening to*

- Remember the typical port number (the ‘door number’) for a web server is port 80
- But you can use any number that has not been used by the computer such as 8000

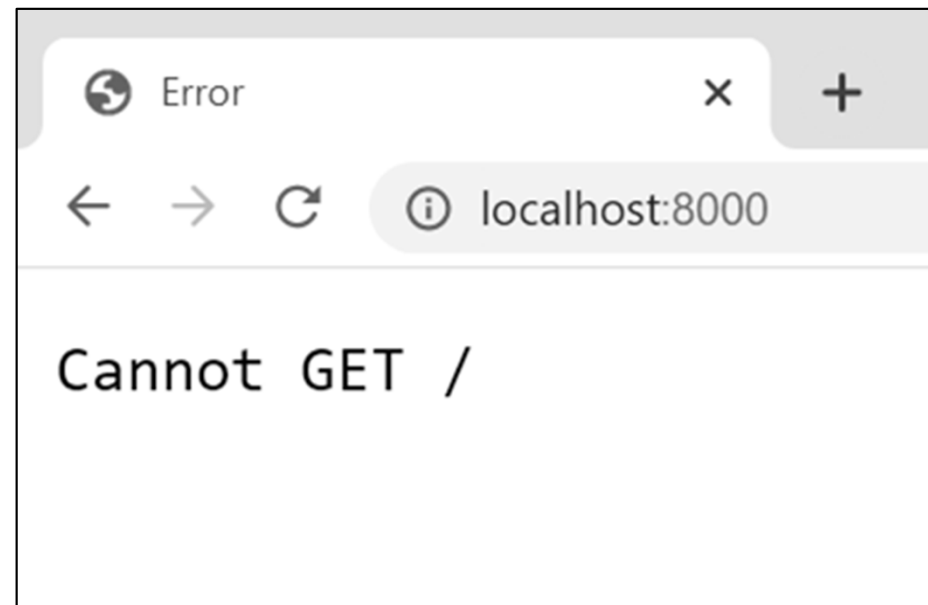
Accessing the Server

- You can use a browser to connect to the server using this URL:

`http://localhost:8000`

*Going through
port 8000 of the
local computer*

- This is what you get if you do that
- It gives you an error because we have not set it up correctly

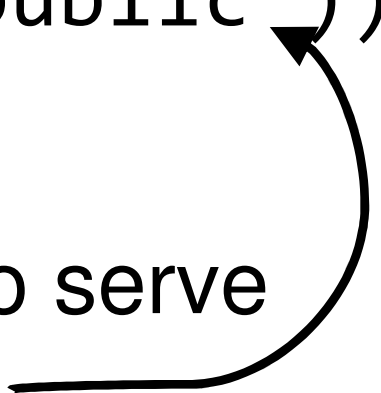


Serving Static Files

- Let's make a minimal web server
- The server only sends back any requested static files, i.e. files that do not change
- You only need to add one line of code, before running `app.listen()`, i.e.:

...

```
app.use(express.static('public'));  
app.listen(8000);
```

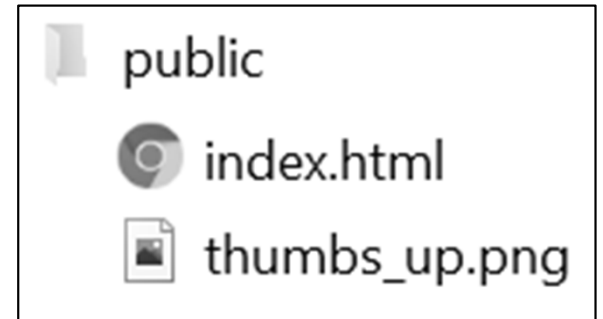
- This above code asks Express to serve the files under the folder 'public' 

The Static Files

- Let's say you have the following files under the 'public' folder:

```
<!DOCTYPE html>
<html>
<head>
  <title>Node.js Server</title>
</head>
<body>
  <h1>Congratulations!
    You have set up the server!</h1>
  
</body>
</html>
```

index.html



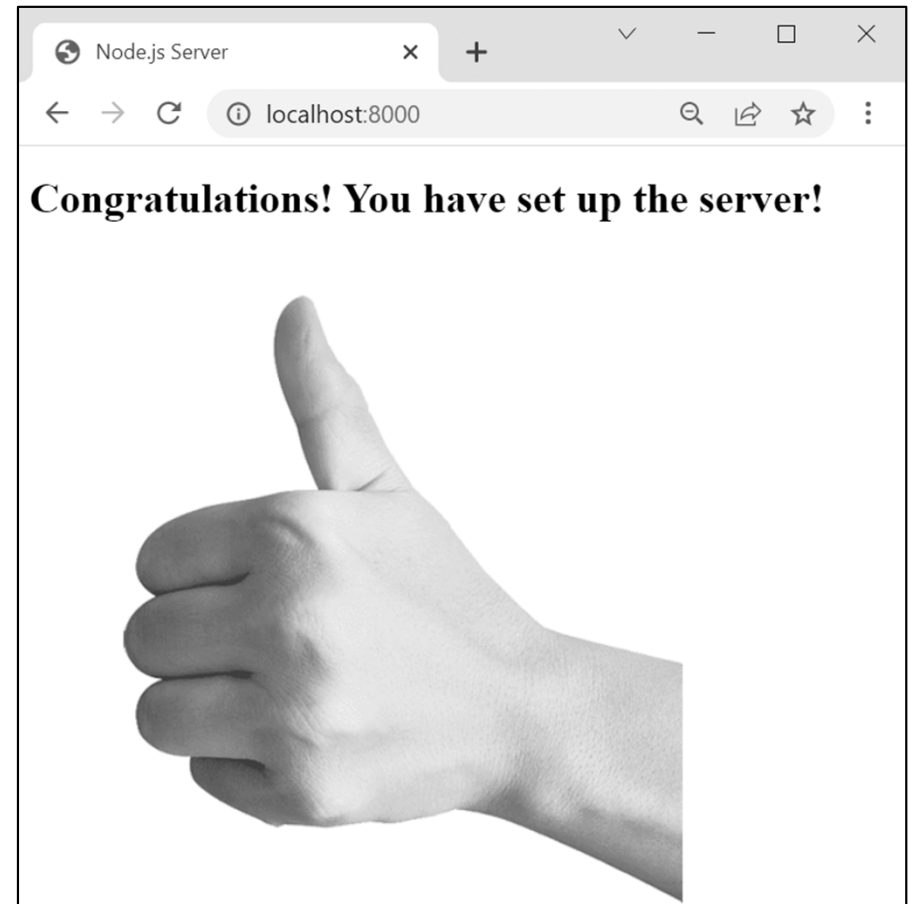
thumbs_up.png

Using the Server

- If you use the URL `http://localhost:8000`

in your browser,
you will get the
page on the right:

- So a few lines of code give you a completely working web server!



*index.html is the
root file of the
Express server*

Returning JSON Content

- It is very common that a web server sends 'pure' data to the client, such as JSON data
- Let's extend our minimal web server so that it can return JSON content
- To do that, you need to:
 - Configure the server to handle GET requests under a specific request path
 - Use the HTTP response to send JSON data back to the client

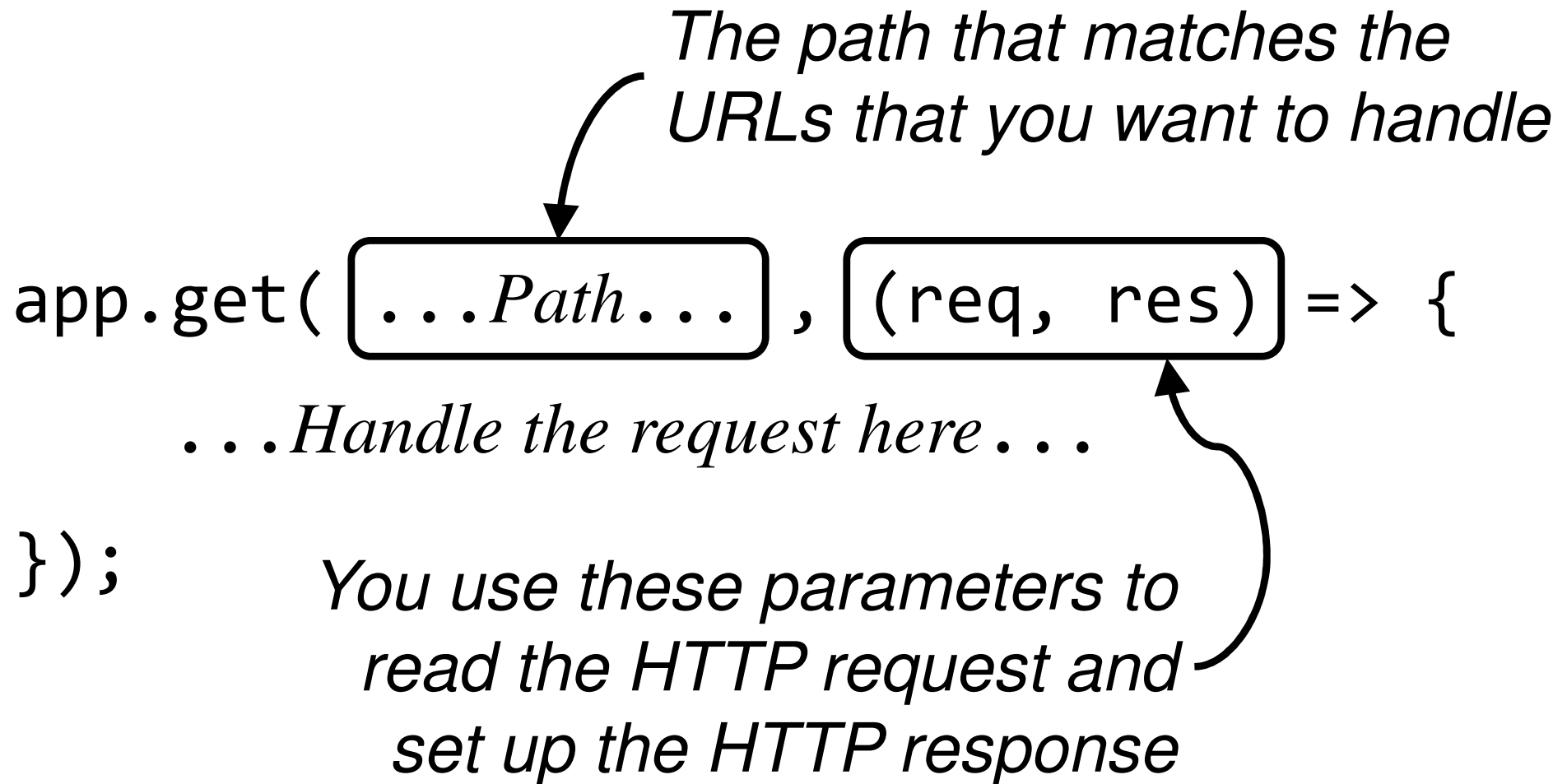
Handling GET Requests

- You can set up the server to handle the HTTP GET requests, like this:

The path that matches the URLs that you want to handle

```
app.get( ...Path... , (req, res) => {  
    ...Handle the request here...  
});
```


You use these parameters to read the HTTP request and set up the HTTP response



Using the Path

- The path parameter in `app.get()` tells Express the URLs you want to work with
- It tries to match the path of the URL, i.e.:

`http://me.com/files/images/face.png`

A horizontal curly brace is positioned below the path portion of the URL, specifically under the segment `/files/images/face.png`. The brace starts under the first slash of the path and ends under the final file extension, with a small vertical tick mark pointing down at its center.

This is the path of the URL

- Some examples are shown in the next slide

Working Path Examples

- Let's say the path parameter is `'/admin'`
- Any URLs that have their path starting with `'/admin'` are handled by `app.get()`, i.e.:

`http://localhost:8000/admin`

`http://localhost:8000/admin?user=root`

- But these URLs do not match:

`http://localhost:8000/adminpage`

`http://localhost:8000/admin.html`


`http://localhost:8000/pages/admin`

Making a JSON Response

- You can make your server to send out JSON response easily

- Here is an example:

Note the path used here



```
app.get("/serverinfo", (req, res) => {
```

```
  res.json({  
    name: "First Node.js Server"  
  });
```

```
});
```

Use the HTTP response to send some JSON data (the input is a JS object)

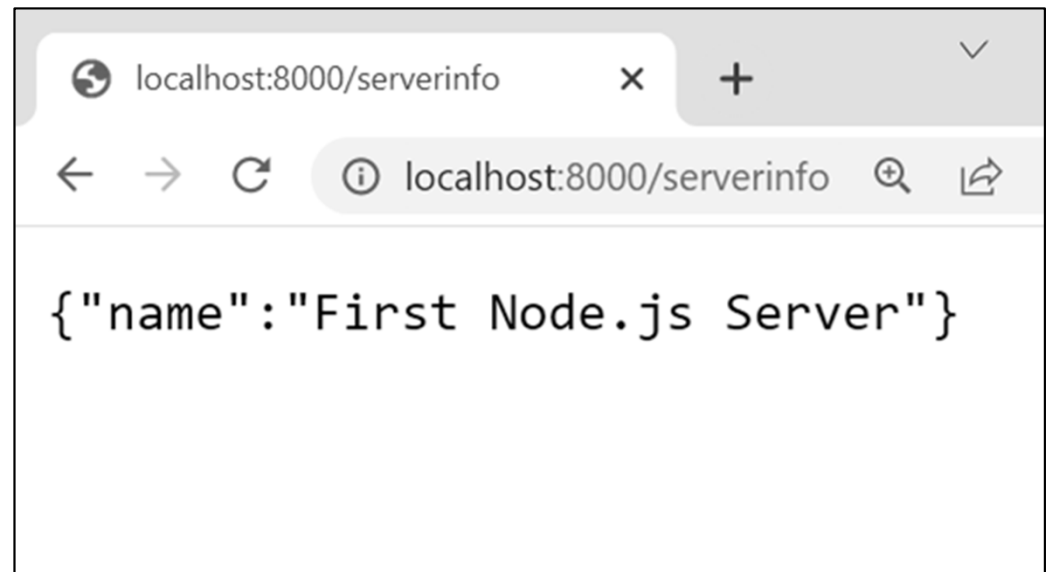


Accessing the Data

- To get to the example JSON, you can enter this URL in the browser:

`http://localhost:8000/serverinfo`

- You will then get the JSON data back in the browser, like this:



Examining the HTTP Response

- If you view the HTTP response headers from the browser, you can see the content type has been correctly returned to you:

HTTP/1.1 200 OK

X-Powered-By: Express

Content-Type: application/json; charset=utf-8



Content-Length: 31

ETag: W/"1f-TZE5rDAhbt43Ky0yzeWpovZgDc"

Date: Tue, 15 Mar 2022 17:38:25 GMT

Connection: keep-alive

Keep-Alive: timeout=5

Extending the Server

- We have built a simple web server that can serve static content and JSON data
- Later in the course, we will extend the web server to include more features, such as:
 - Reading JSON data from files
 - Using the query parameters
 - Updating JSON data
 - Handle POST requests