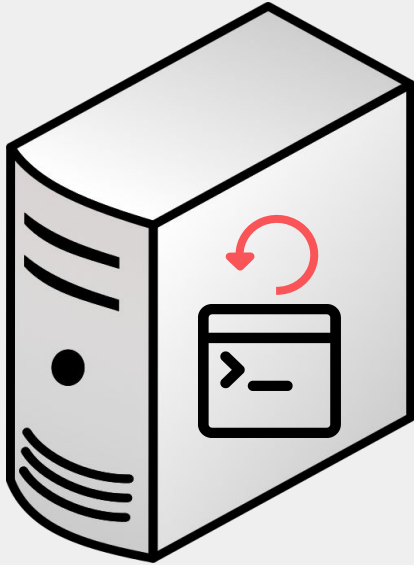


Set up your own Server

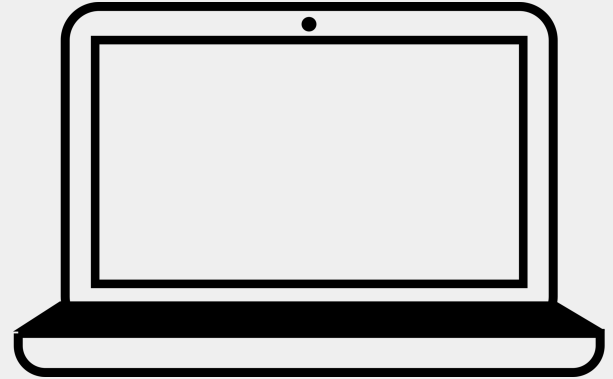
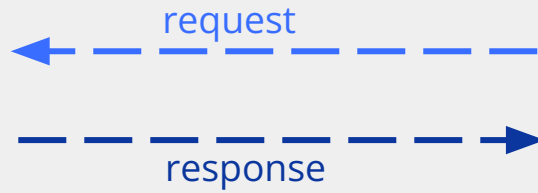
Mufaro Makiwa

What is a server?

What is a server?

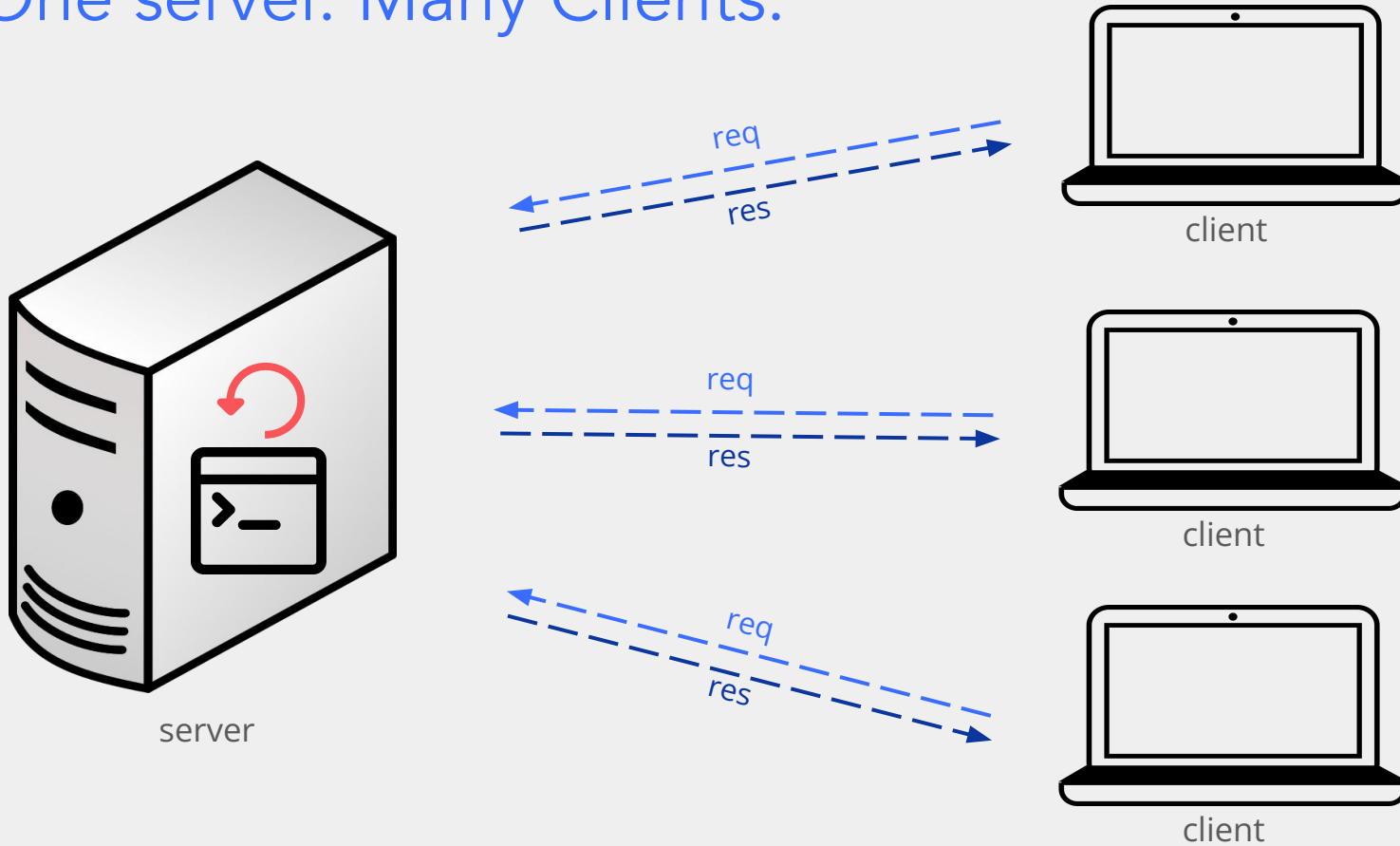


server



client

One server. Many Clients.



What is the need for a server?

- File access
- Centralization
- Security

Processes & Ports

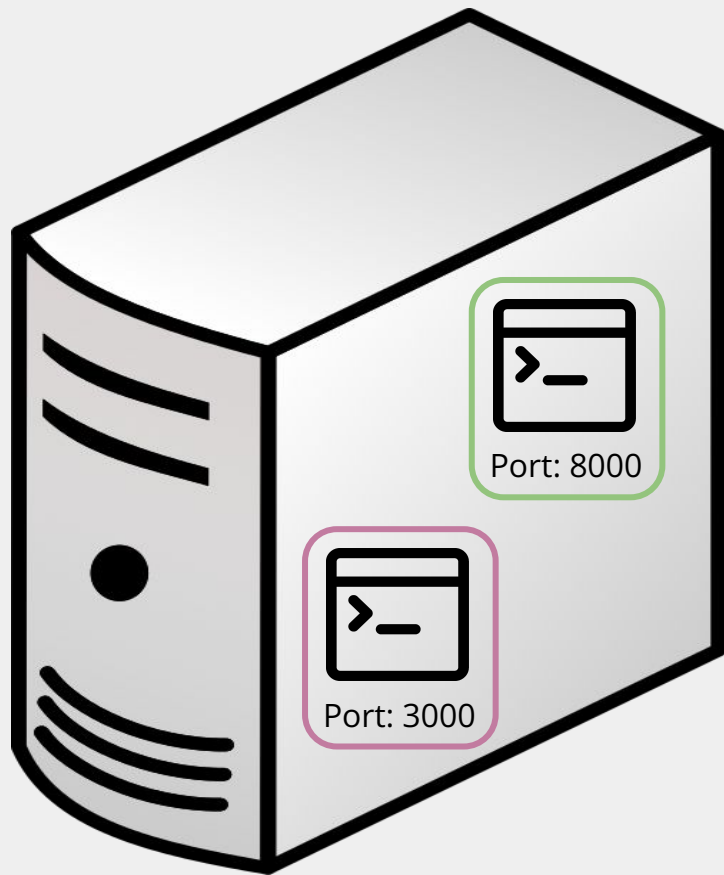
A server **binds** to a **port** on a computer.

`http://example.com:8000`

`http://example.com:3000`

`protocol://domain:port`

Why do we not need to specify ports on most websites?



Two servers on one computer

Using your own machine as a server

- Every computer can run server code!
- Your own computer has a special domain: **localhost**
 - `http://localhost:3000` → connects to a server on port 3000.

What is a server written in?

What is a server written in?

Many options!



Flask

python



java

Express JS

javascript

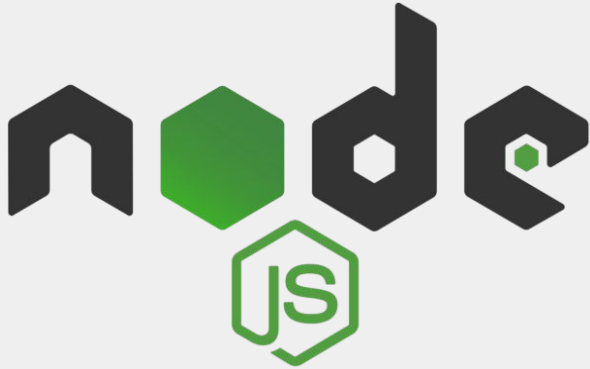


python

... and more!

What do we teach in this class?

We use **express.js**, but we also need to use a tool to **run** javascript code.



Node **runs** Javascript on your machine



Express allows you to **structure** your Javascript

Node Package Manager (npm)

“npm install”

“npm run hotloader”

Importing External Code

Every Javascript (Node) project has some necessary information.

Importing External Code

Every Javascript (Node) project has some necessary information.

```
{
  "name": "class-example",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

`package.json` holds project **metadata**.

Importing External Code

Every Javascript (Node) project has some necessary information.

```
{
  "name": "class-example",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

`package.json` holds project **metadata**.

Importing External Code

Every Javascript (Node) project has some necessary information.

```
{
  "name": "class-example",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

`npm install`



package.json



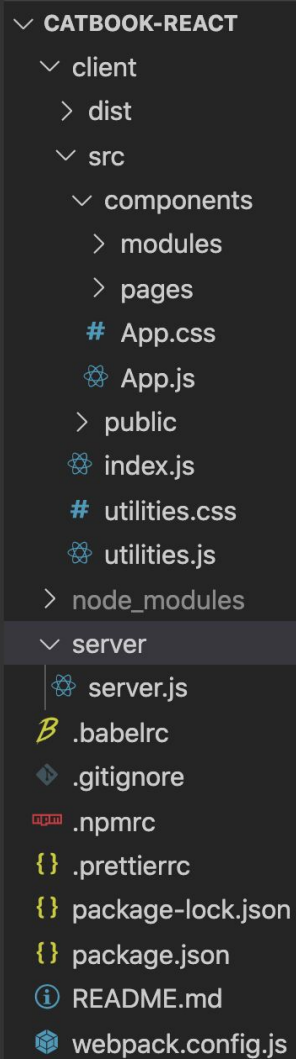
package-lock.json



node_modules

package.json holds project **metadata**.

node_modules contains **imported code**



Understanding our codebase

/client

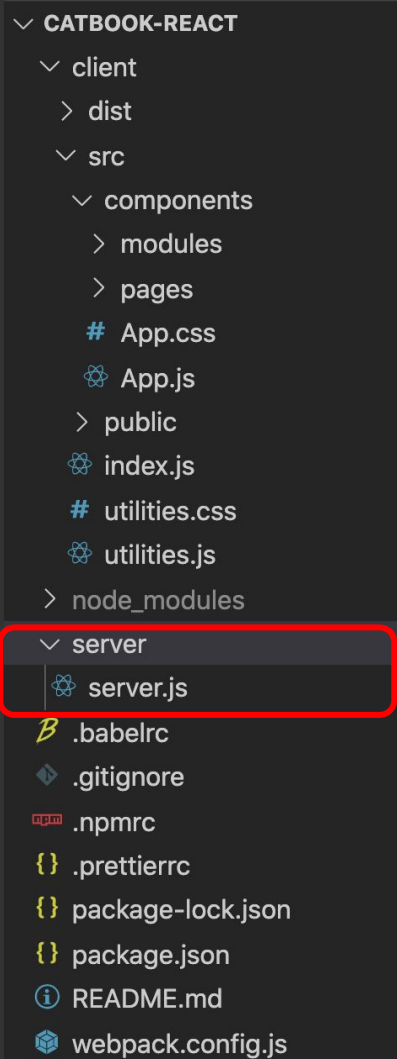
- Contains all of our React code, components, pages, utilities, etc. (Front end)

/server

- Contain all of our backend code

Other files

- Set up by staff to configure React app. Feel free to ask staff about any of these files



Understanding our codebase

/client

- Contains all of our React code, components, pages, utilities, etc. (Front end)

/server

- Contain all of our backend code

Other files

- Set up by staff to configure React app. Feel free to ask staff about any of these files

Creating our first API endpoint

// server.js

```
18 // create a new express server
19 const app = express();
20
21 app.get("/api/test", (req, res) => {
22   res.send({ message: "Wow I made my first API!" });
23 });
24
```

Creating our first API endpoint

// server.js

```
18 // create a new express server
19 const app = express();
20
21 app.get("/api/test", (req, res) => {
22   res.send({ message: "Wow I made my first API!" });
23 });
24
```




HTTP Method

Creating our first API endpoint

// server.js

```
18 // create a new express server
19 const app = express();
20
21 app.get("/api/test", (req, res) => {
22   res.send({ message: "Wow I made my first API!" });
23 });
24
```




Express Route

Creating our first API endpoint

// server.js

```
18 // create a new express server
19 const app = express();
20
21 app.get("/api/test", (req, res) => {
22   res.send({ message: "Wow I made my first API!" });
23 });
24
```



Function handler

Creating our first API endpoint

// server.js

```
18  // create a new express server
19  const app = express();
20
21  app.get("/api/test", (req, res) => {
22    res.send({ message: "Wow I made my first API!" });
23  });
24
```

Server Response

Similar to a return statement

Creating our first API endpoint

// server.js

```
18 // create a new express server
19 const app = express();
20
21 app.get("/api/test", (req, res) => {
22   res.send({ message: "Wow I made my first API!" });
23 });
24
```

HTTP Method

Express Route

Function handler

Server Response

Similar to a
return
statement

Creating our first API endpoint

```
18 // create a new express server
19 const app = express();
20
21 app.get("/api/test", (req, res) => {
22   res.send({ message: "Wow I made my first API!" });
23 });
24
```

← → ↻ 🏠 ⓘ localhost:3000/api/test

📱 Apps 📁 webdev 📺 YouTube 📧 6.148 (5 unread) 🟠 Y

```
{"message":"wow I made my first API!"}
```


Adding Error Handling

```
25 // any server errors cause this function to run
26 app.use((err, req, res, next) => {
27
28
29
30
31
32
33
34
35
36
37
38
39 });
40
```

server.js

Adding Error Handling

```
25 // any server errors cause this function to run
26 app.use((err, req, res, next) => {
27   const status = err.status || 500;
28
29
30
31
32
33
34
35
36
37
38
39 });
40
```

server.js

Adding Error Handling

```
25 // any server errors cause this function to run
26 app.use((err, req, res, next) => {
27   const status = err.status || 500;
28   if( status === 500 ){
29     // 500 means Internal Server Error
30     console.log("The server errored when processing a request");
31     console.log(err);
32   }
33
34
35
36
37
38
39 });
40
```

server.js

Adding Error Handling

```
25 // any server errors cause this function to run
26 app.use((err, req, res, next) => {
27   const status = err.status || 500;
28   if( status === 500 ){
29     // 500 means Internal Server Error
30     console.log("The server errored when processing a request");
31     console.log(err);
32   }
33
34   res.status(status);
35
36
37
38
39 });
40
```

server.js

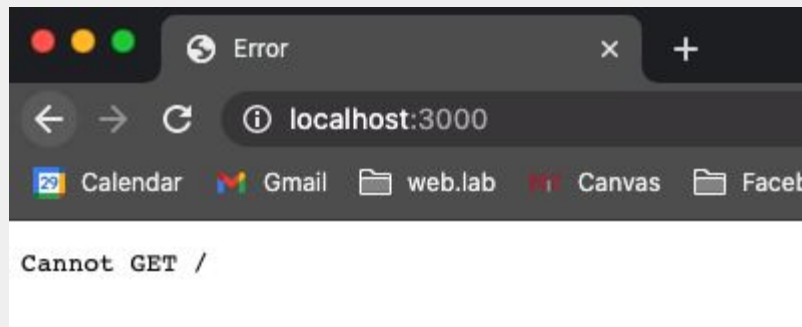
Adding Error Handling

```
25 // any server errors cause this function to run
26 app.use((err, req, res, next) => {
27   const status = err.status || 500;
28   if( status === 500 ){
29     // 500 means Internal Server Error
30     console.log("The server errored when processing a request");
31     console.log(err);
32   }
33
34   res.status(status);
35   res.send({
36     status: status,
37     message: err.message,
38   });
39 });
40
```

server.js

Serving React Files

Navigate to `http://localhost:3000` in your browser.



Introducing a "Catch All" Endpoint

All endpoints which are not concretely defined will hit the `*` endpoint.

We want to return the HTML and bundle files of our React app.

Introducing a "Catch All" Endpoint

All endpoints which are not concretely defined will hit the `*` endpoint.

We want to return the HTML and bundle files of our React app.

```
15 // import libraries needed for the webserver to work!
16 const express = require("express"); // backend framework for our node server.
17 const path = require("path"); // provide utilities for working with file and directory paths
18
```


Introducing a "Catch All" Endpoint

All endpoints which are not concretely defined will hit the `*` endpoint.

We want to return the HTML and bundle files of our React app.

```
15 // import libraries needed for the webserver to work!
16 const express = require("express"); // backend framework for our node server.
17 const path = require("path"); // provide utilities for working with file and directory paths
18
```

Introducing a "Catch All" Endpoint

All endpoints which are not concretely defined will hit the `*` endpoint.

We want to return the HTML and bundle files of our React app.

```
26 // Load the compiled react files, which will serve /index.html and /bundle.js
27 const reactPath = path.resolve(__dirname, "..", "client", "dist");
28
29
30
31
32
33
34
```

Introducing a "Catch All" Endpoint

All endpoints which are not concretely defined will hit the `*` endpoint.

We want to return the HTML and bundle files of our React app.

```
26 // Load the compiled react files, which will serve /index.html and /bundle.js
27 const reactPath = path.resolve(__dirname, "..", "client", "dist");
28 app.use(express.static(reactPath));
29
30
31
32
33
34
```

Introducing a "Catch All" Endpoint

All endpoints which are not concretely defined will hit the `*` endpoint.

We want to return the HTML and bundle files of our React app.

```
26 // Load the compiled react files, which will serve /index.html and /bundle.js
27 const reactPath = path.resolve(__dirname, "..", "client", "dist");
28 app.use(express.static(reactPath));
29
30 // for all other routes, render index.html and let the react router handle it
31 ∨ app.get("*", (req, res) => {
32   // ...
33 });
34
```

Introducing a "Catch All" Endpoint

All endpoints which are not concretely defined will hit the `*` endpoint.

We want to return the HTML and bundle files of our React app.

```
26 // Load the compiled react files, which will serve /index.html and /bundle.js
27 const reactPath = path.resolve(__dirname, "..", "client", "dist");
28 app.use(express.static(reactPath));
29
30 // for all other routes, render index.html and let the react router handle it
31 ∨ app.get("*", (req, res) => {
32   |   res.sendFile(path.join(reactPath, "index.html"));
33   | });
34
```

```
15 // import libraries needed for the webserver to work!
16 const express = require("express"); // backend framework for our node server.
17 const path = require("path"); // provide utilities for working with file and directory paths
18
19 // create a new express server
20 const app = express();
21
22 ✓ app.get("/api/test", (req, res) => {
23   |   res.send({ message: "Wow I made my first API!" });
24   | });
25
26 // Load the compiled react files, which will serve /index.html and /bundle.js
27 const reactPath = path.resolve(__dirname, "..", "client", "dist");
28 app.use(express.static(reactPath));
29
30 // for all other routes, render index.html and let the react router handle it
31 ✓ app.get("*", (req, res) => {
32   |   res.sendFile(path.join(reactPath, "index.html"));
33   | });
```

Serving React Files

Order matters!

You must define your endpoints from **most specific** to **least specific**.

```
app.get("/api/test", (req, res) => {  
  res.send("Endpoint A");  
});
```

```
app.get("/api/*", (req, res) => {  
  res.send("Endpoint B");  
});
```

```
app.get("/api/*", (req, res) => {  
  res.send("Endpoint B");  
});
```

```
app.get("/api/test", (req, res) => {  
  res.send("Endpoint A");  
});
```

Serving React Files

Order matters!

You must define your endpoints from **most specific** to **least specific**.

```
app.get("/api/test", (req, res) => {  
  res.send("Endpoint A");  
});
```

```
app.get("/api/*", (req, res) => {  
  res.send("Endpoint B");  
});
```

good!

```
app.get("/api/*", (req, res) => {  
  res.send("Endpoint B");  
});
```

```
app.get("/api/test", (req, res) => {  
  res.send("Endpoint A");  
});
```

bad :(

localhost:3000 or localhost:5000?

- Both are servers running on our machine!
 - localhost:5000 → **hotloader, processes react code.**
 - localhost:3000 → **node server, our backend server.**
- To view your website → use localhost:5000
- To test your backend → use localhost:3000

Last step: Adding Middleware

Express allows you to add **middleware**.

- Run code in between receiving a request and running endpoint code.
- Register middleware by calling `app.use(...)`
- Express JSON Parser Middleware will be needed for the next workshop!

```
19 // create a new express server
20 const app = express();
21
22 // allow us to make post requests
23 app.use(express.json());
24
```