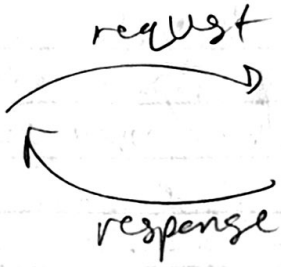
Data Exchange on Web

http messages
requests



node
express

Server side
(computers)

- servers constantly listen to serve requests and make resources available
- cloud is connected servers

Request Message Get, url, header, body (user)
Response Status, headers, response data (server)
JSON

HTTP Methods Get Post Put Delete

Setting up server w/ http module

http.CreateServer(() => request, response
Call back hit after every user request

port → communication endpoint of Server

response.end(data, encoding, callback)
MUST be called on each response
↳ gives a response to serve back to request

Request Header to give more info about response type
writeHead(Status, {Headers})
write() , then end()

Send back response Status codes to inform browser of result
(inherent in express)

Request Object and method of that object
Send back html based on path and Send Status

to send html, send contents of file not the file itself, serve manually

You can send any file type, but need to indicate the proper content-type

Express Basics

Express.js is a minimal and flexible node.js web app framework designed to make web dev much faster

app = express ( ); import express (create server)

app has get / post / put / delete / all / use / listen
look for http verbs

```
.get ("/path to what user is trying to access",
    callback () => {
    envoked every time user does get request

.all ("/*path" callback
        res.status(404).send("Oh no")
```

all http verbs

SET UP Server

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

app.listen(5000, () => {
  log('server is listening');
});
```

middleware

need absolute path when doing send file
need to send all files for index!
☆ use all files that are static in public folder

Static | API vs Server Side Rendering SSR

folder that is use and pulls all static code for a site to host and serve that content
- server does not have to change this content (js/css/etc)
Static relative to the server (not to the browser)

Can just use the entire root w/ folder and other files

API → json, sends data, res.json()
SSR → template, sends templates, res.render()

☆ API http interface to interact w/ our data ☆
that data is sent in json and will return w/ res.json()
responsible for sending back the data, not using data

JSON Basics

eventually, we will use a database to store this data
res.json() gets return data
this serves the data and you can get request

Route Param | Query String Param

Send html to server page
set up get request for resarce of directory path
iterate over products to send back specific items

Provide/return specific info of one product
route parameters
↳ :productID (placeholder)
use product ID string to get product
/1 fetches product w/ ID 1
Set up if statement to catch errors

Query String Params (url params)
Send small bits of info to server via url
?query = ___ + ___ +
search for matching data via url query in fetch url

/.../.../query is the query route (req.query)
to actualy query use ? then add qsp ——
unfound query sent w/ 200 b/c resarce exists but the query failed

Middleware

in Express → functions that execute during request to the server. Each has access to req, res obj
IS everywhere in express!

request → middleware → response
(do stuff)

get(path, middleware, callback)
when you work w/ middleware, you MUST pass it onto next middleware unless you are ending w/ res
next() or send()

---

1) would be nice to include middleware function from other file
2) what if you have a bunch of routes
   ☆ method, add middleware to any route
   module.exports
   put at the top
app.use(logger) → use logger as middleware on all routes
☆ app.use(/path, logger) → apply to any route under API
when base matches use

Multiple middleware functions ☆ order matters
pass in an array app.use([logger, auth]);

if auth works, then allow middleware to pass to next()
query string: if string matches, the pass to next
else, send status error

pass 2 middlewares to one route, add array
express middleware and third party

Express Method verbs

:id route param

- get read | post insert | Put update | Delete
Use these methods to work w/ database
require (include) assumes .js
adding data is easy
browser does get as default to read data

- Post/insert to dB/Browser use() adds middleware to all
in html, we see /login action
post() data (req.body) login handled on server
parse from data from url → if name valid → login
Javascript parse → See javascript.html
201 for succesful post
we know how to post/get but need to parse json
express.json() gets response as json
post req from app and provide content type

- Postmate Can group request in collections
test routes immediately instead of needing front end
to put/send data → go to html body and add json data

- Put → editing the data → w/ route param
get value in params (req) and send data on body
req.params → param sent | req.body has value

- Delete → user hits route, we remove from json list
delete w/ id or check if value pair exists

Get /api/orders (get all orders) | /api/orders/:id (get id order)
post /api/orders (send data via query or params)
Put /api/orders/:id Update by id
Delete /api/orders/:id delete by id

Express Router

Make app.js file less busy w/ routes
group routes together and have separate controllers

all routes are api/people → lets consolidate them
set up router from express

change app. → router.

middleware

app.use("api/people", require("./routes/people"));
use the api/people route as a base for the people.js file

Clean it up more w/ controllers
create functions w/ logic for routes
then you can just do
router.route("/").get(getPeople).post(createPerson)
Chain