Intro to Node Servers & Express

Objectives

- To gain a conceptual understanding of server-side code.
- To learn the fundamentals of building a server using plain Node.js to listen and respond to client-side requests.
- To gain a preliminary understanding of the basic elements of an Express server.
- To gain an initial understanding of Express routing.

Darkness...

There is a light(saber) at the end of the tunnel...

How to Survive the Full Stack Apocalypse

- Form a study group
- Take notes
- Ask questions!
- Office hours
- RTFM

What is a web client?

What is a server?


XPS

What is localhost?

localhost === *this computer*

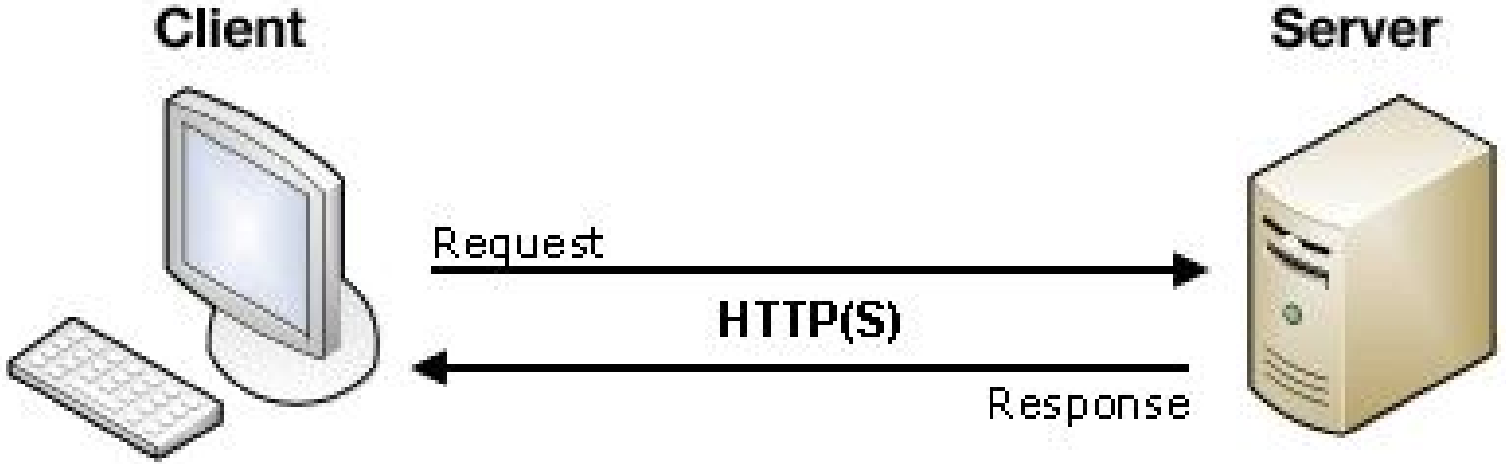

XPS

How do a server and web client communicate?

Hyper Text Transfer Protocol

Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

RTFM https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol


Client
Server
Request
HTTP(S)
Response

What is full stack web development?

Activity: Examples of Server Side Code (5 min)

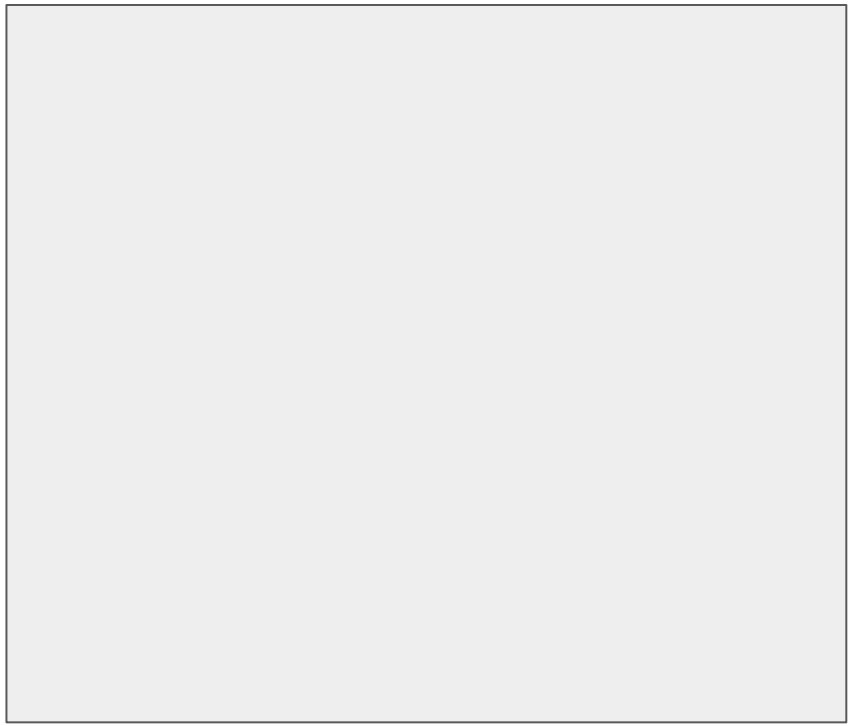
With your neighbor, find and discuss examples of server side code

Creating a Server

Creating a Server

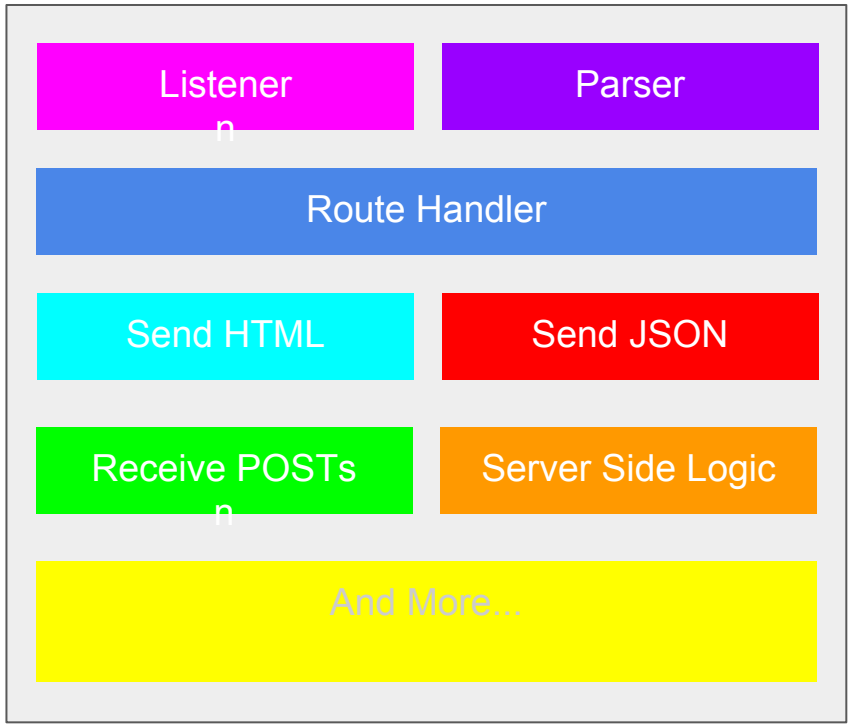
- For our purposes, “creating a server” equates to writing the code that handles what the server will do
    a. It’s important to note that even though you pay for server-side hardware, you still need to create the code that goes inside.
- This code you create handles things like:
    - Connections to the database
    - Handling client-side URL requests
    - Authenticating and logging user requests

Big Box



- Your server is an empty box
- We will fill it with code and modules to make it do stuff in response to requests

Big Box



- Listen for requests
- Parse URLs
- Based on URL keywords, route logic
- Send HTML or JSON files
- Receive data (POST requests)
- Do cool shit

My First Server

Demo: My First Server

Follow along...

```
var http = require("http");
```

The http package ("small box") allows our server ("big box") to have the capability of handling requests and responses.

Where did we get ‘http’?

Node standard library

```
var PORT = 8080;
```

What is a port IRL?

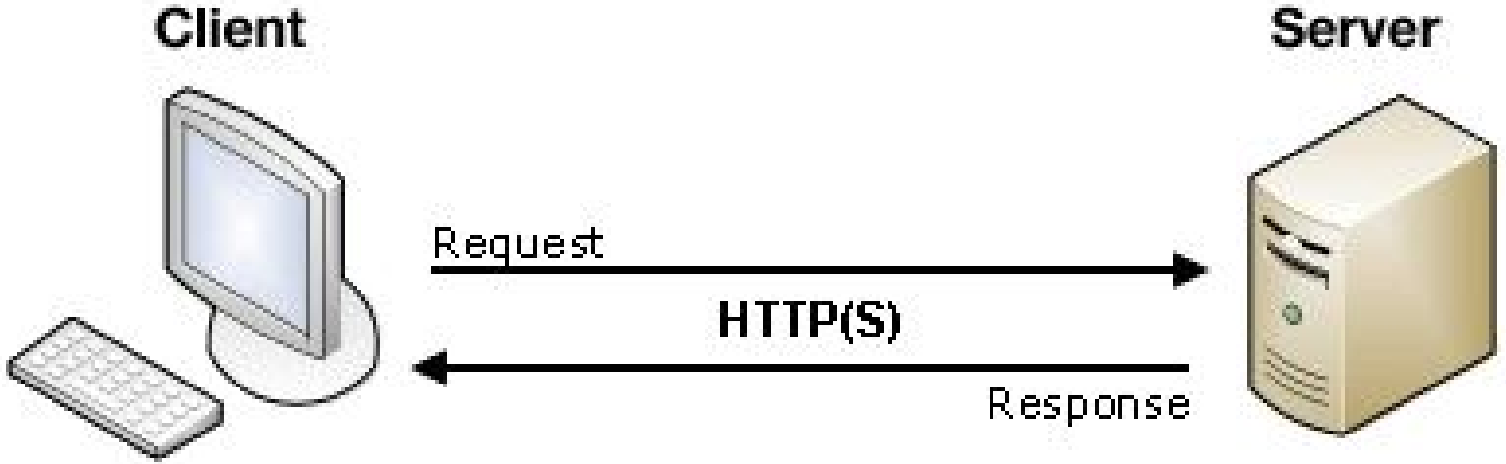
So what is a port in CS?

An endpoint of communication in an operating system

RTFM https://en.wikipedia.org/wiki/Port_(computer_networking)

```
function handleRequest(request, response) {

  response.end("It Works!! Path Hit: " + request.url);
}
```


Client
Server
Request
HTTP(S)
Response

```
var server = http.createServer(handleRequest);
```

The function that's passed in to createServer is called once for every HTTP request that's made against that server, so it's called the request handler

```
server.listen(PORT, function() {
  console.log("Server listening on: http://localhost:%s", PORT);
});
```

In order to actually serve requests, the listen method needs to be called on the server object.

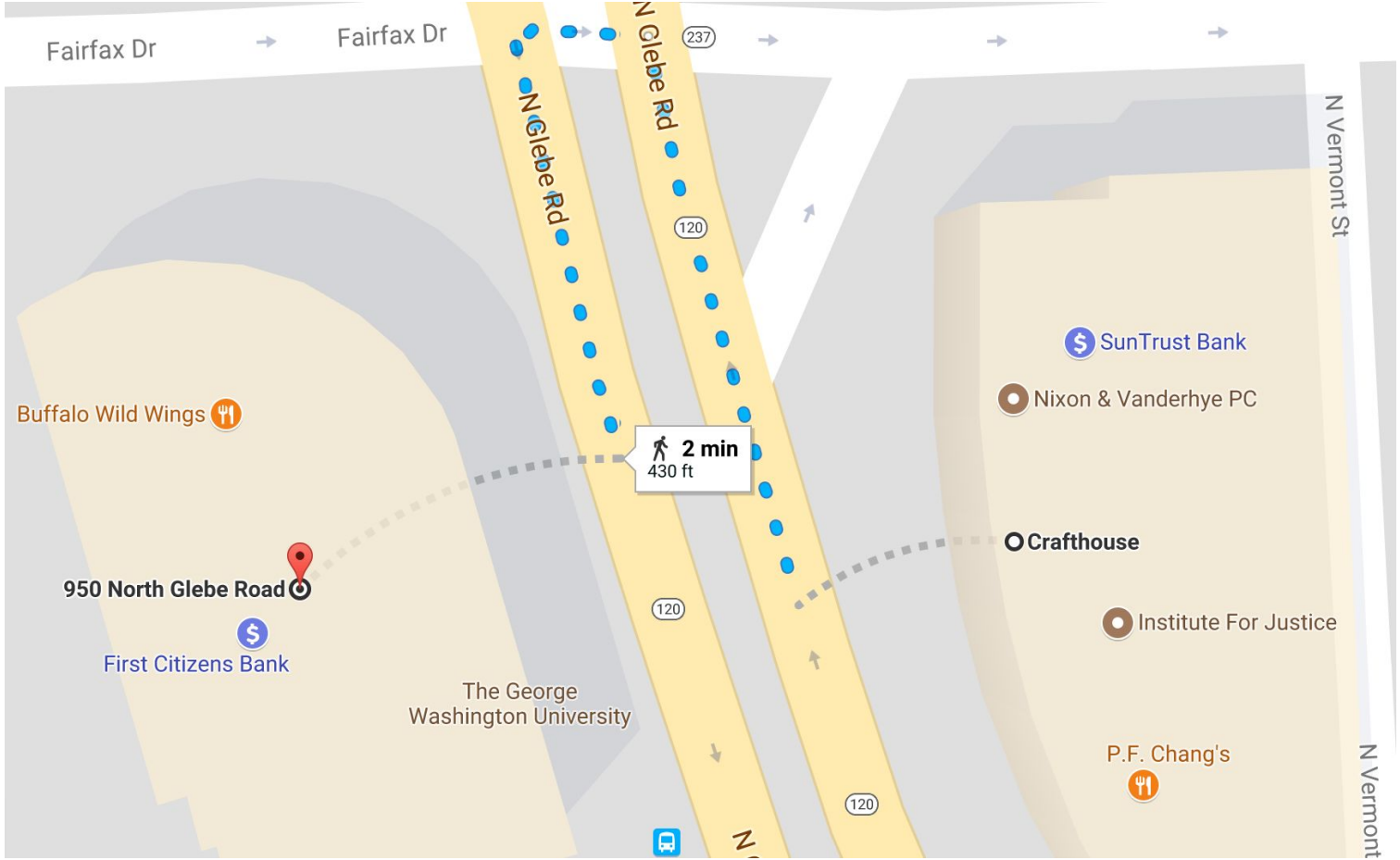
RTFM https://nodejs.org/api/http.html

Activity: Two Servers

See 02-Two-Servers for instructions

Review: Two Servers

Routing

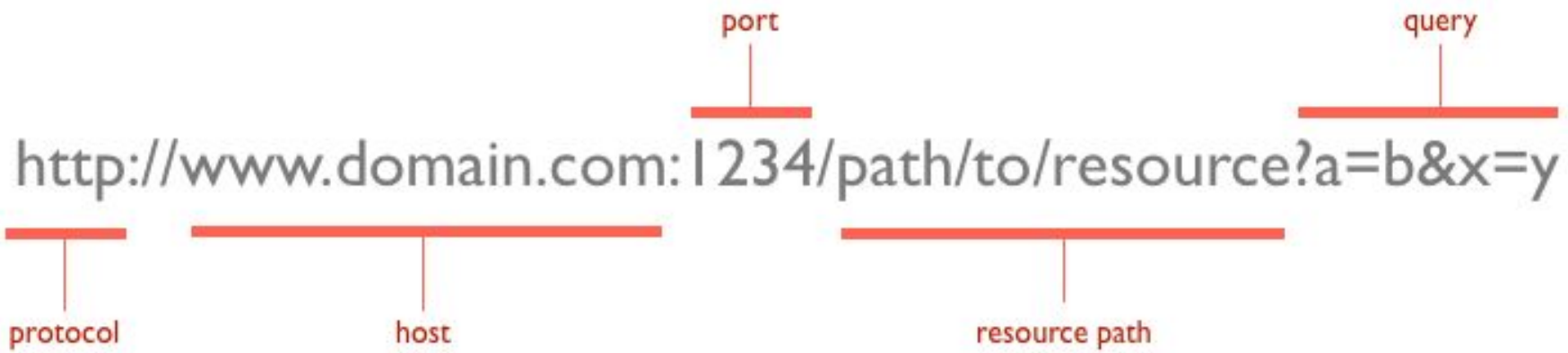

Fairfax Dr
Fairfax Dr
237
N Glebe Rd
N Glebe Rd
120
N Vermont St
SunTrust Bank
Nixon & Vanderhye PC
Buffalo Wild Wings
2 min
430 ft
Crafthouse
950 North Glebe Road
120
Institute For Justice
First Citizens Bank
The George
Washington University
P.F. Chang's
120
N Vermont

maps.google.com

How an application responds to a client request

What is a URL?

Uniform Resource Locator


port
query
http://www.domain.com:1234/path/to/resource?a=b&x=y
protocol
host
resource path

RTFM https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_URL

Demo: Portfolio

- Open 03-Portfolio
- $ node server.js
  - The url library is needed to parse the requested URL.
  - We use the abbreviated terms req and res, which are short for request and response.
  - We use a switch-case statement which routes the code to a different function depending on the URL provided.
  - Finally, the way in which we created HTML dynamically and rendered it on the page in each function.

Activity: Portfolio (5 min)

Explain 03-Portfolio/server.js to your friends and neighbors

Demo: Serving HTML

- Open 04-Serving-HTML
- What’s going to happen?

Activity: Serving HTML (5 min)

Discuss 04-Serving-HTML/* with your friends and neighbors

Demo: Serve Favorites

$ node 05-Serve-Favorites/server.js

Activity: Serve Favorites (30 min)

See 05-Serve Favorites for instructions

Review: Serving Favorites

- We created the basic skeleton of a Node server (requiring: url && http; port)
- We set up a listener to initiate the server's handling of requests.
- We created a function handleRequest which takes in a request URL, parses it, then relays the user to the correct page.

Activity: Lunch

Download and install Postman https://www.getpostman.com/

HTTP Methods

GET vs. POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

- GET - Requests data from a specified resource
- POST - Submits data to be processed to a specified resource

Demo: Request Methods

This app is going to log the type of request it receives, along with any information that was sent with the request

- Open Postman
- $ node 06-Request-Methods/server.js
- Send requests

Activity: POST Requests (20 min)

You will create a server that will accept POST requests

See 07-POST-Method for instructions

Review: POST Requests

POST routes are effectively endpoints that client-side code can use to send data to the server.

Express

Express is a framework for Node to simplify server code

https://expressjs.com/

What is a route?

Demo: Routes

http://nyt-mongo-scraper.herokuapp.com/

1. User visits “/” route (GET)
2. User visits “/saved” (GET)
3. Request triggers the code in the server route
4. Server responds by returning HTML with saved articles
5. User saves an article using “/api/headlines” (POST)
6. Request triggers the code in the server route which sets article to saved in db
7. Saved articles are loaded in HTML

Demo: Routing

http://starwars-express.herokuapp.com/

Also http://starwars-express.herokuapp.com/api

The API accepts an additional parameter at the end of the URL.

For example http://starwars-express.herokuapp.com/api/yoda

$ node 08-StarWars-1/server1.js

http://localhost:3000/

http://localhost:3000/yoda

```
var app = express();
```

Creates an Express application

Activity: The Phantom Menace (5 min)

See 08-StarWars-1 for instructions

Review: 08-StarWars-1/server1.js

```
var obiwankenobi = {
  name: "Obi Wan Kenobi",
  role: "Jedi Knight",
  age: 42,
  forcePoints: 1350
}

app.get('/obiwankenobi', function(req, res){
  res.json(obiwankenobi);
})
```

Activity: Attack of the Clones (5 min)

See 09-StarWars-2 for instructions

Review: 09-StarWars-2/server2.js

- We created an array of character objects
- /:characters syntax is a way of saying we have a "variable" parameter in the URL route
- Pass the route to variable chosen using req.params.characters
- See console

Activity: Revenge of the Sith

See 10-StarWars-3 for instructions

Review: 10-StarWars-3/server3.js

- http://localhost:3000/api/yoda
- http://localhost:3000/api/hansolo

?

This parameter is optional

http://localhost:3000/api

req.params. ...

Creates a route parameter

for?

This for-loop "checks" which character is being sought after in the URL -- then finds that character's information and re-displays it back to the user in the form of a JSON

Where might we find routing like this?

https://www.washingtonpost.com/business/technology/andrew-ng-hates-paranoid-androids-and-other-fun-facts/2017/08/21/ce4cdb02-8697-11e7-96a7-d178cf3524eb_story.html?utm_term=.e714dbbc6455

Demo: Homework

$ node /Users/jarednielsen/FullStack-Lesson-Plans/01-Class-Content/13-express/02-Homework/Solutions/FriendFinder/server.js

http://localhost:8080/

You will build a dating application or compatibility test.

In essence, the application saves each user's survey responses in the database, then compares the responses against everyone in the database to identify the best match.

Further Reading

- https://nodejs.org/en/docs/guides/anatomy-of-an-http-transaction/
- https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
- https://www.quora.com/Why-is-80-a-special-port-in-localhost
- https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
- https://www.w3schools.com/tags/ref_httpmethods.asp