

Server-Side!

The Coding Bootcamp

A Moment of Caution

Focus!

Node + Express Servers and Routing are two of the

MOST IMPORTANT CONCEPTS
in the ENTIRE program.



Seriously... Try to learn this now.

Forewarning: This is the HARD Stuff

- These next units are some of the hardest to grasp.
- **But are also some of the most important.**
- This is where you go from humble HTML, CSS hackers to full-stack, employable engineers.
- Bring your A-Game!!!



Don't let this be you!

How to Succeed Through the Full-Stack Apocalypse

1. Form a Study Group Now

Get in the habit of explaining in-class exercises to one another. Work together on homework assignments.

2. Take notes!

Jot down concepts or key ideas that come out of activities. This class isn't a lecture, but it may help you keep things straight!

3. Ask “Conceptual” Questions

Don't let big picture ideas gloss over you. Be courageous and ask questions in class. (Don't worry -- if your question isn't relevant, we'll let you know. But if it is, you owe it to yourself to understand!)

4. Come to Office Hours

As things get trickier, we'll be available to review any concepts during office hours. Come to these. Ask questions! We want to help.

How to Succeed Through the Full-Stack Apocalypse

Lastly, be confident !!!

There is zero reason to get despondent at this point. If you've made it this far, you've proven that you have what it takes to succeed. **Keep going!**



Remember Our Mantra...

When it comes to web development...

Remember Our Mantra...

I know nothing.

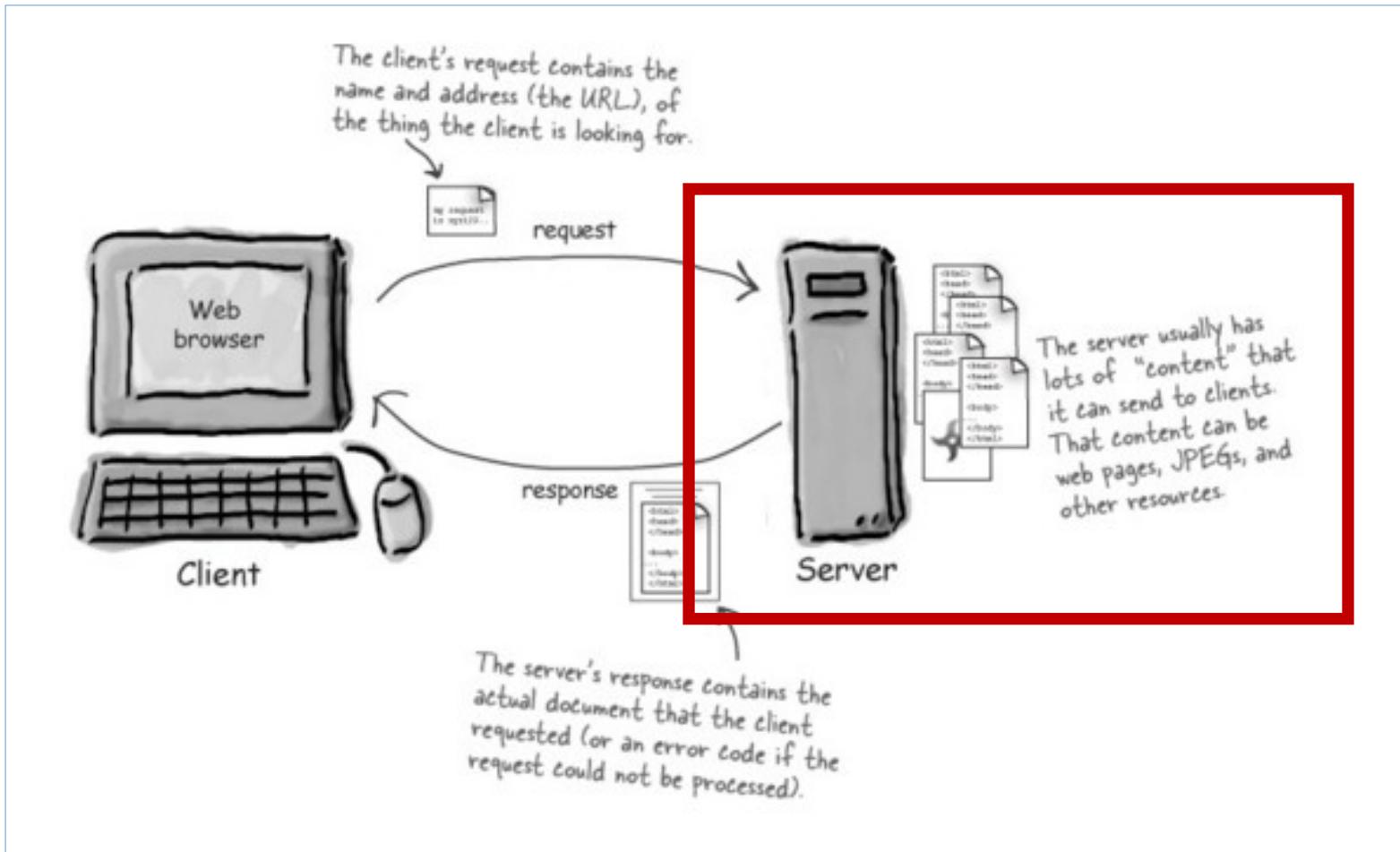
So Let's Begin...

Remind me again...

What is a **server**?

- In modern **web applications** there is a constant back-and-forth communication between the visuals displayed on the user's browser (**frontend**) and the data and logic stored on the server (**backend**).

Server Definition



Server:

The Machine and Code that handles requests and respond to them.

Remind me again...

What are examples of **server-side** functions?

- In modern **web applications** there is a constant back-and-forth communication between the visuals displayed on the user's browser (**frontend**) and the data and logic stored on the server (**backend**).

Server-Side Code in

Action!

Visiting a URL and then being given an HTML page.

- Visiting an API end-point that parse URL parameters to provide selective JSONs.
- Clicking an invoice that provides a PDF report.
- Image processing software that takes an image applies a filter, then saves the new version.
- Google providing “results” relevant to your searches on other sites.

Server-Side Code in

Action!

Visiting a URL and then being given an HTML page.

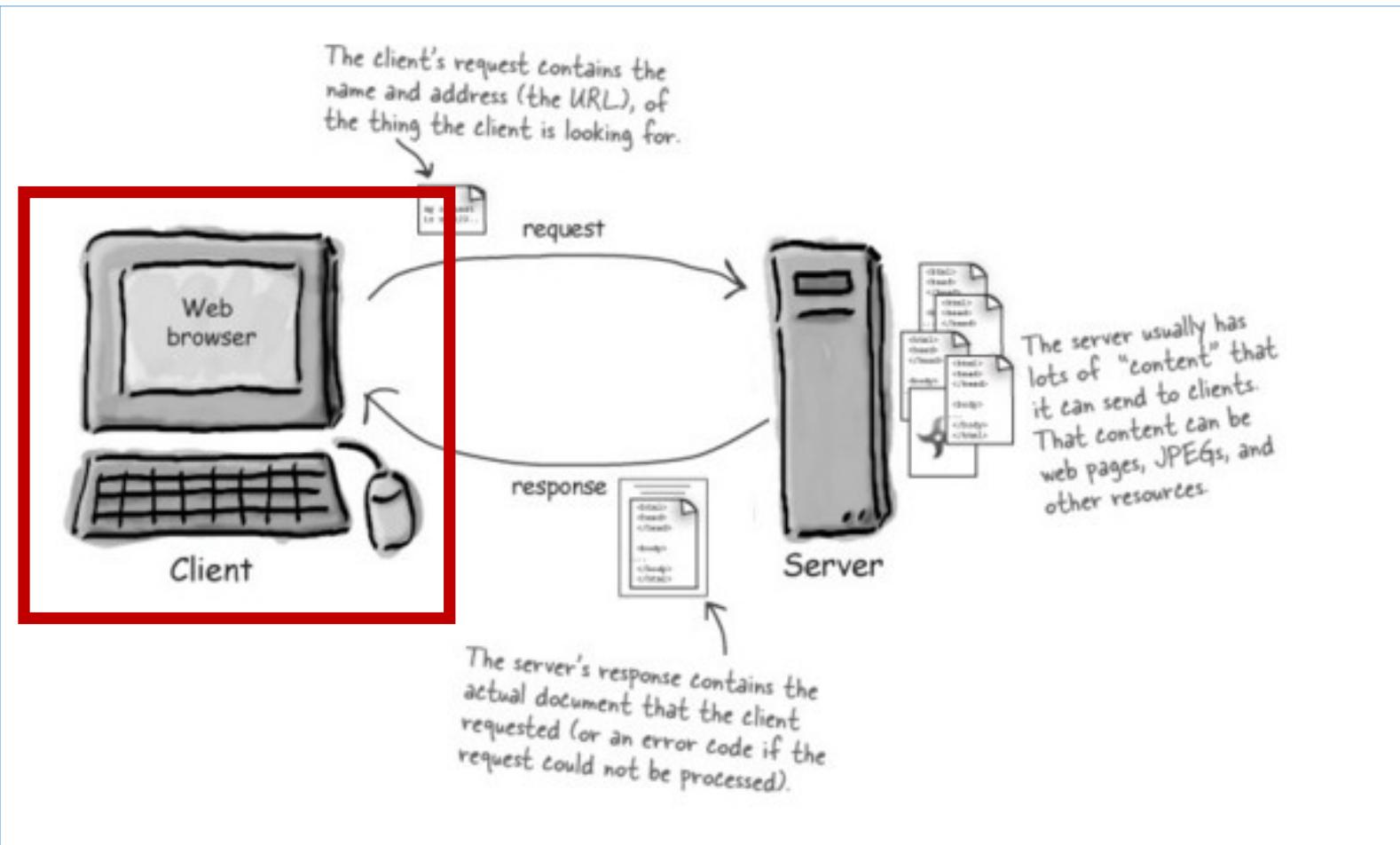
- Visiting an API end-point that parse URL parameters to provide selective JSONs.
- Clicking an invoice that provides a PDF report.
- Image processing software that takes an image applies a filter, then saves the new version.
- Google providing “results” relevant to your searches on other sites.

Remind me again...

What is a **client**?

- In modern **web applications** there is a constant back-and-forth communication between the visuals displayed on the user's browser (**frontend**) and the data and logic stored on the server (**backend**).

Client Definition



Client:

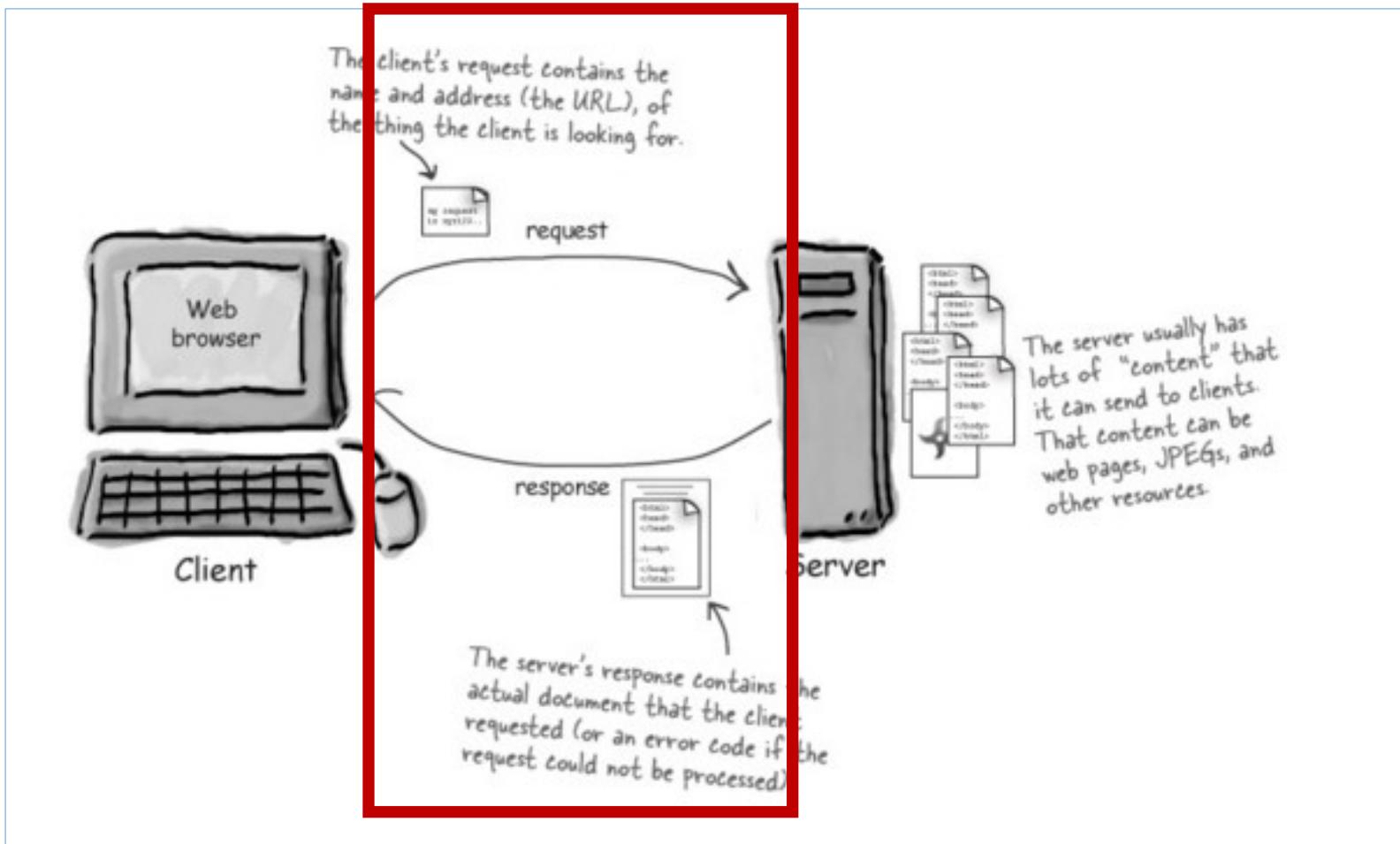
The users' personal machines that make "requests" of the server.

Bonus Question:

How do the client and server
communicate with one another?

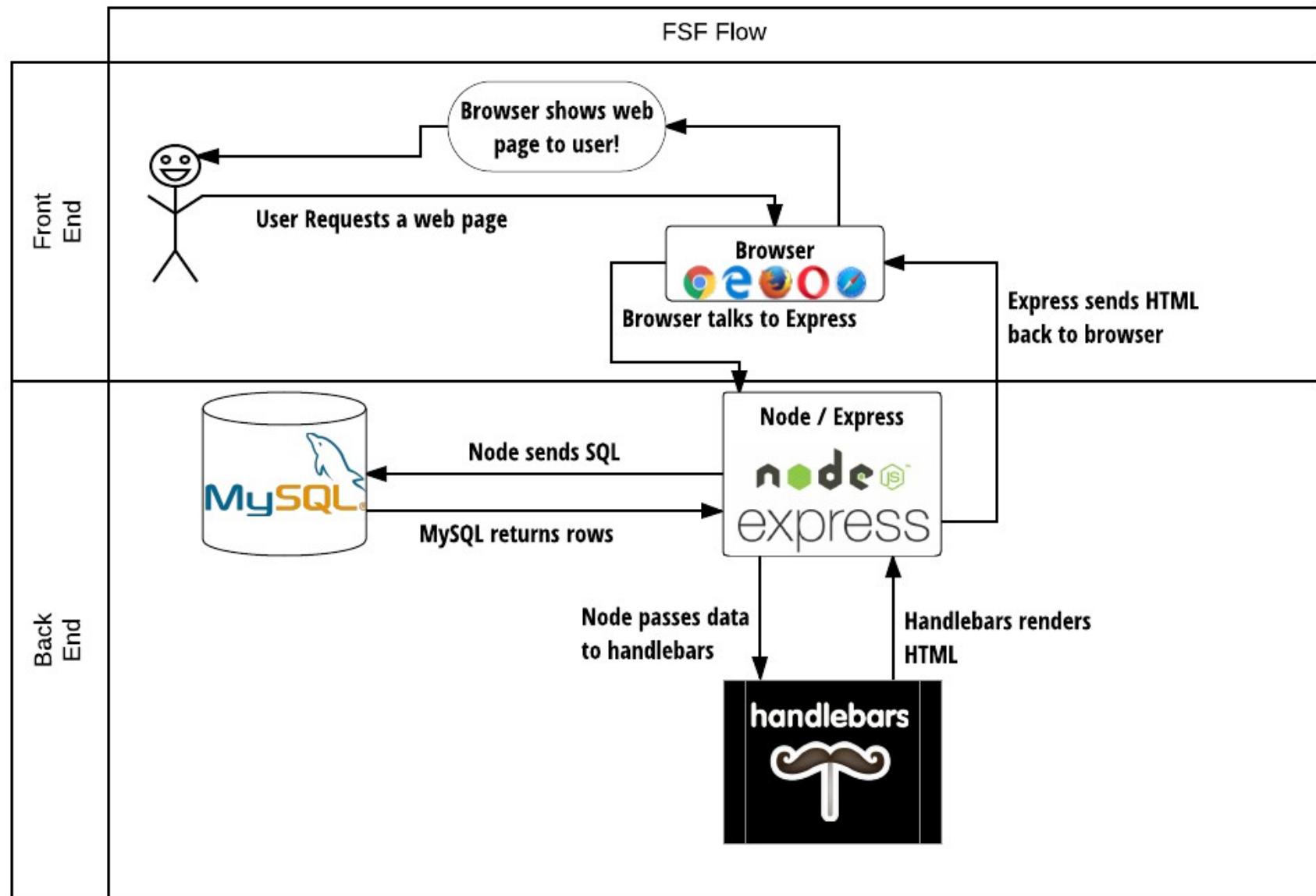
- In modern **web applications** there is a constant back-and-forth communication between the visuals displayed on the user's browser (**frontend**) and the data and logic stored on the server (**backend**).

HTTP Definition



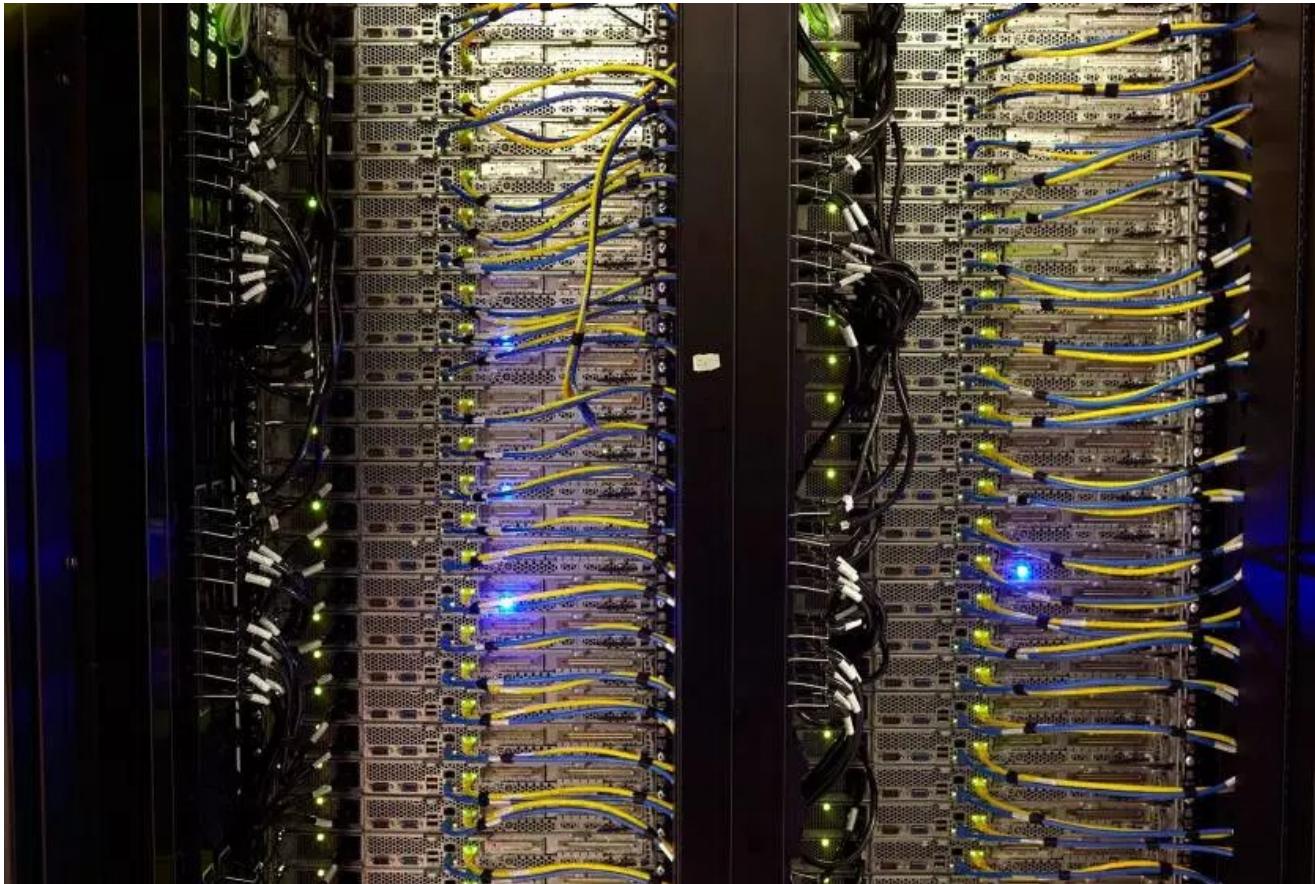
Clients and Servers communicate back and forth using a series of understood communications defined by **HTTP / HTTPS**.

Full-Stack



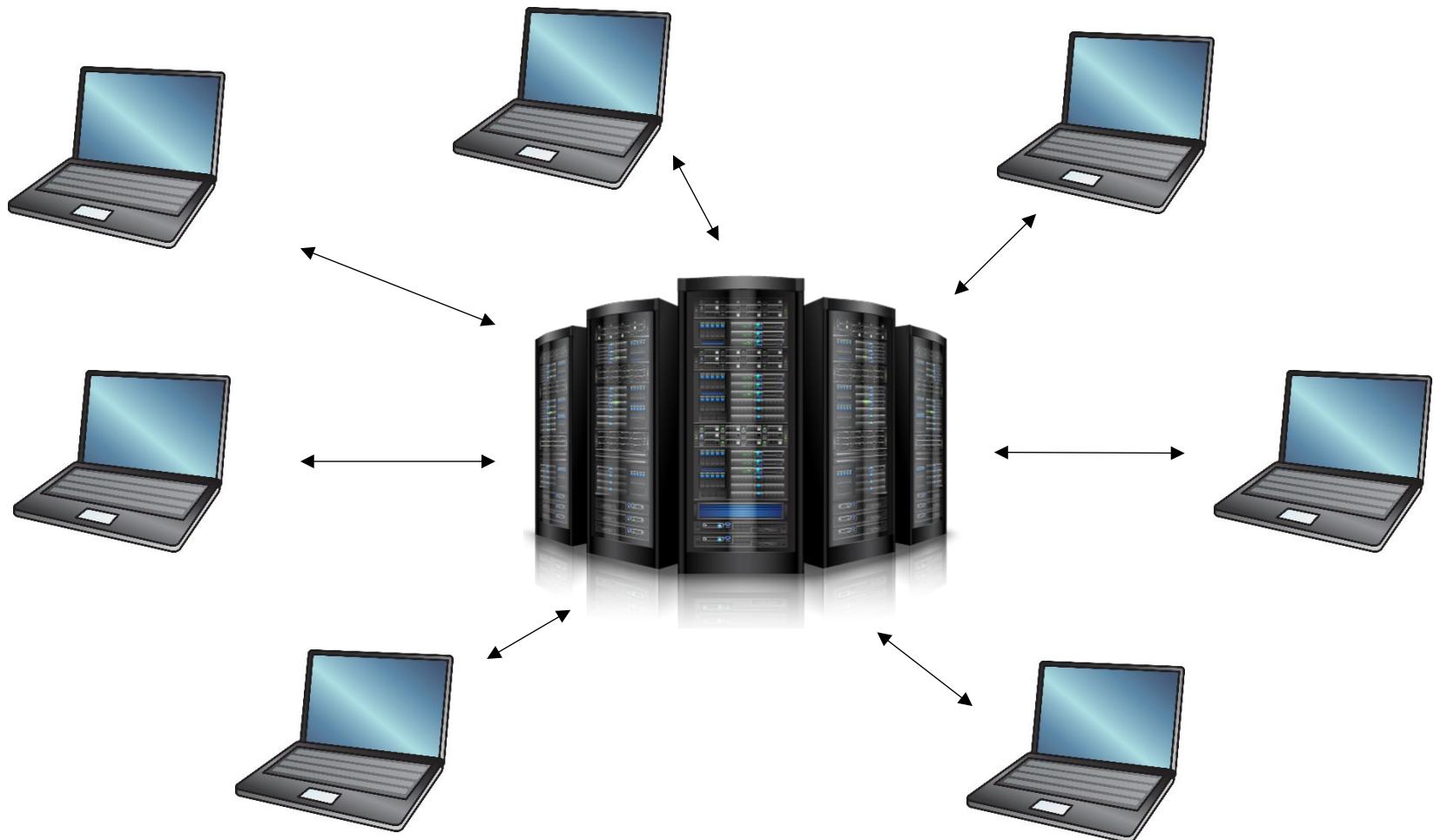
Digging Deep into Server

Visualizing Servers



Server Hardware look like large, ruggedized versions of desktop computers.

Visualizing Servers



These machines (and their respective code) handle all of the requests coming in from browsers accessing a website.

Visualizing Servers



Where does the server **live**?

- constant back-and-forth displayed on the user's browser (**frontend**) and the data and logic stored on the server (**backend**).

Where Do Servers Live?

- Servers live in dedicated hardware intended to handle ALL the requests and responses of many clients.
- Servers can live most often live on cloud platforms like AWS, Heroku, Google Cloud, etc.



Google Cloud Platform

Where Do Servers Live?



Where Do Servers Live?



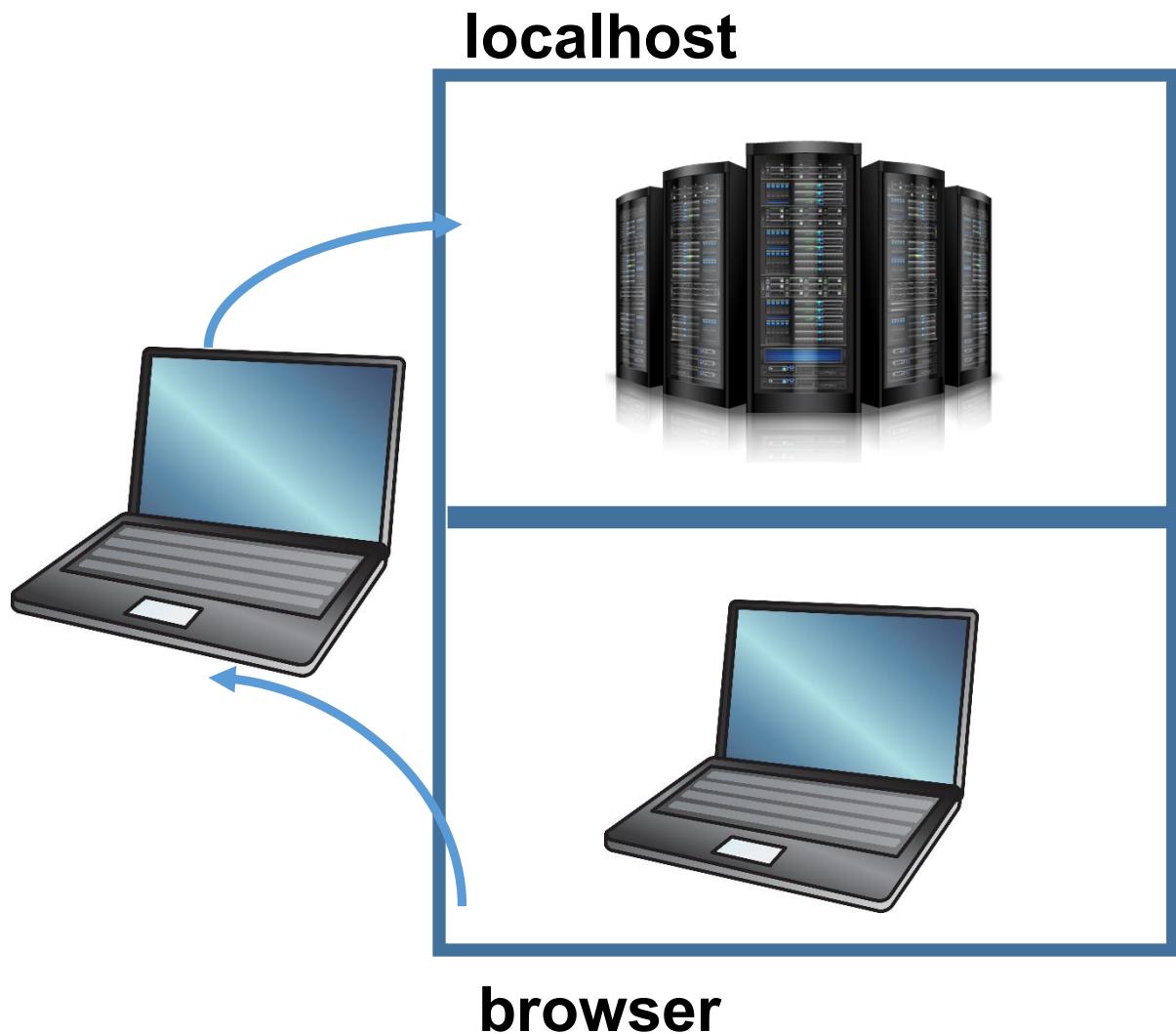
Visualizing Servers



Servers During Development

Important Note:

- During development our personal computers will be able to simulate both.
- We will create a “local server”
- And then use our browser to interact with it.



Building a “Server”

Creating a Server

- For our purposes, “creating a server” equates to writing the code that handles what the server will do.
- 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
 - Performing server-side processes
 - Authenticating user requests
 - Logging client requests

A Big Box

Server

- Throughout this week... imagine your server to be a big, empty box.
- We will be adding code snippets and modules to give our big, empty box the powers to **do** stuff in response to all the requests that come in.

Inside the Box: Connections

Server

Listen

- We'll add listener such that the server can "begin" listening for requests.

Inside the Box: Parsing

Server

Listen

URL Parse

- We'll give our server the ability to "parse" URLs that the user requests.

Inside the Box: Routing

Server

Listen

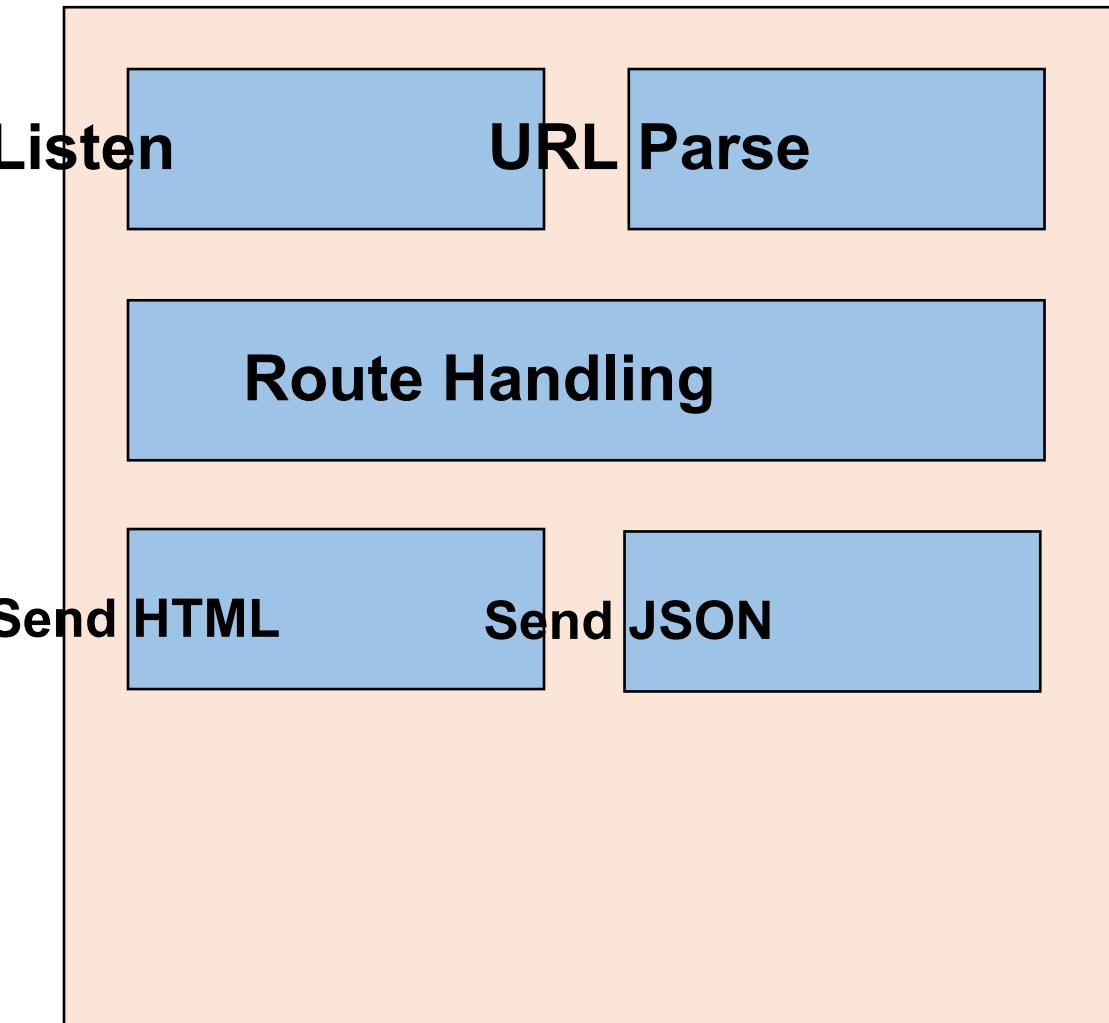
URL Parse

Route Handling

- Then based on the URL's keywords, our server will be able to **route** (or direct the flow of logic to initiate other processes)

Inside the Box: Sending Files

Server



- This subsequent process may be to send an HTML file to be rendered or to send a JSON file to be rendered...

Inside the Box: Receiving Posts

Server

Listen

URL Parse

Route Handling

Send HTML

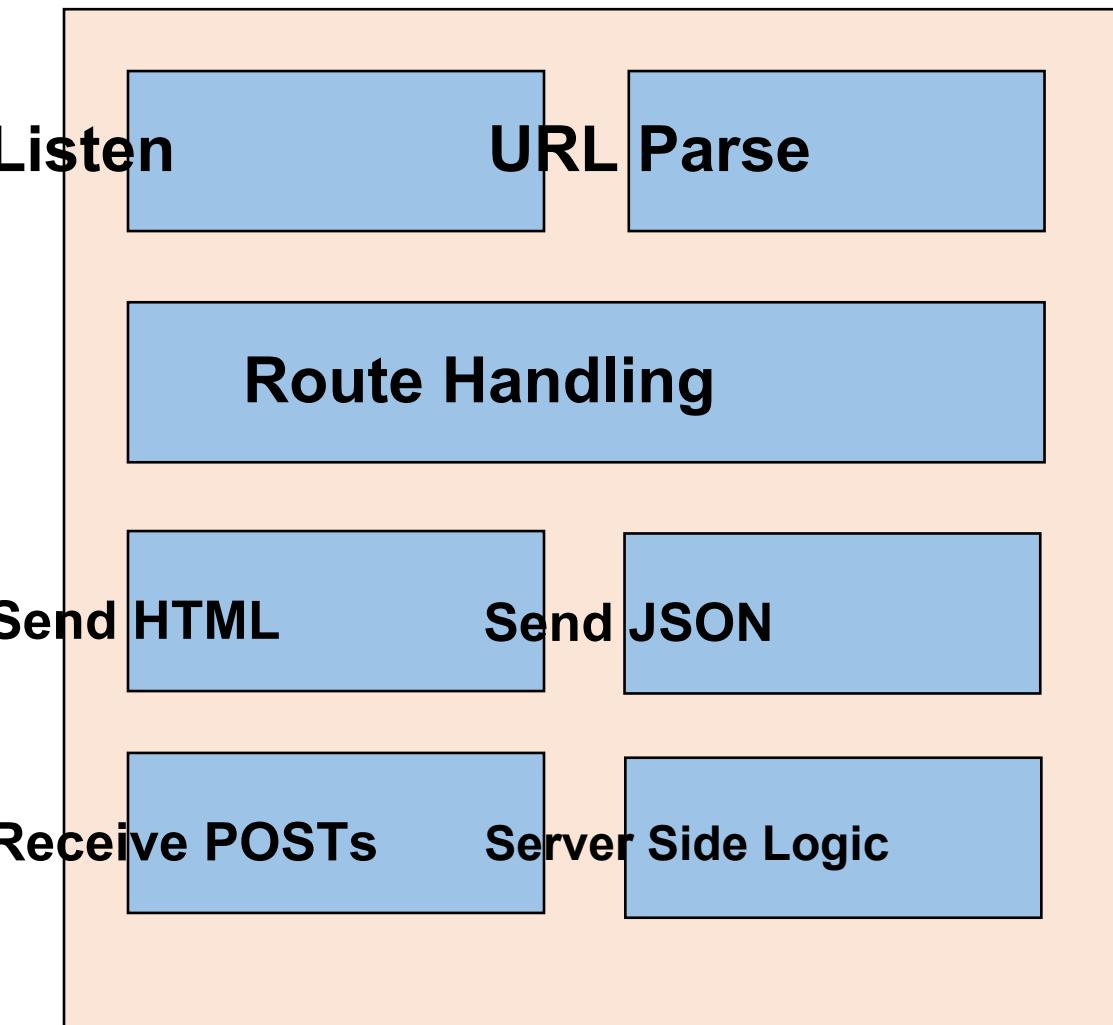
Send JSON

Receive POSTs

- We may also have a new module to handle receiving user's POST requests (i.e. the data they send the server)

Inside the Box: Performing Logic

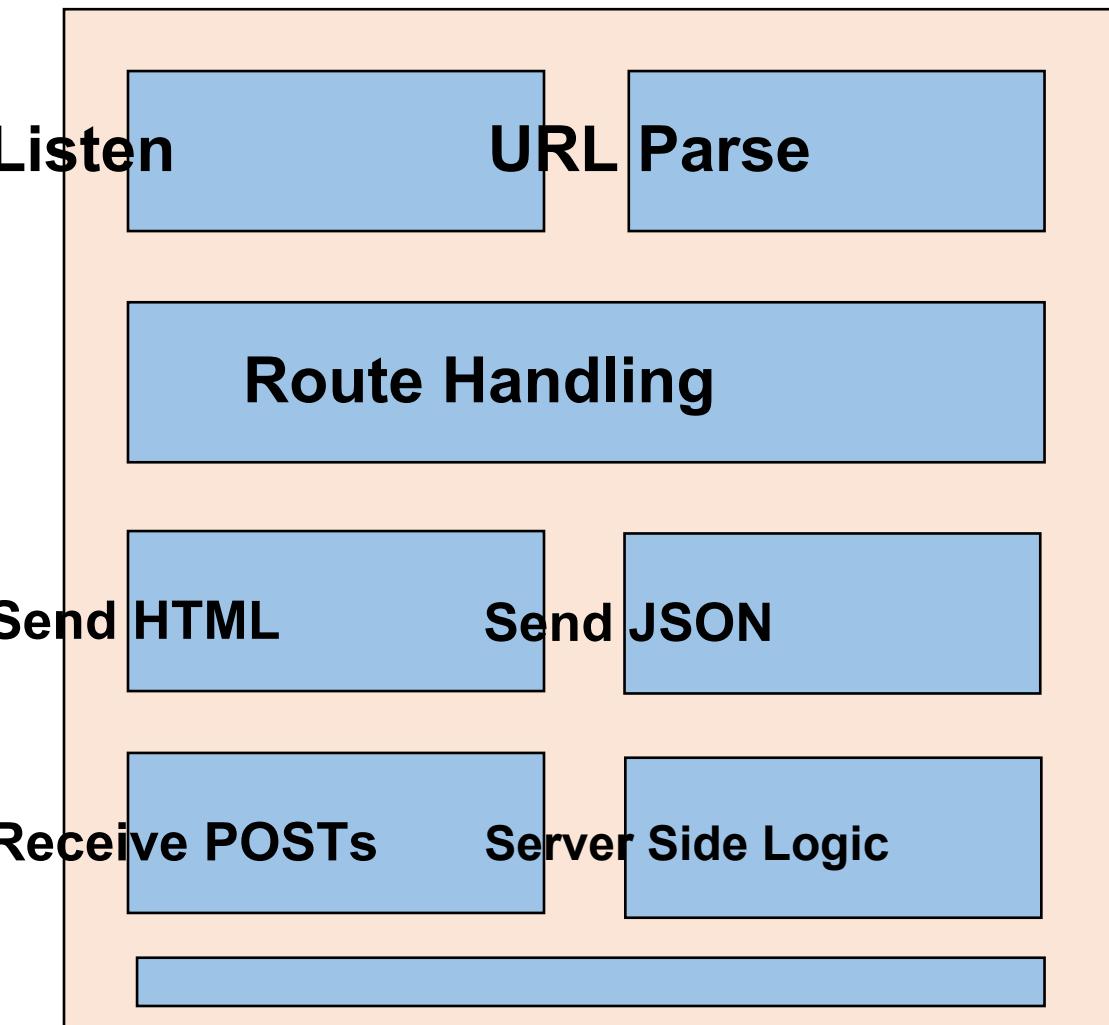
Server



- We may also have complex server-side logic that we want to initiate in response to user's visiting a route endpoint or sending us data.

Inside the Box: And More!

Server



- But it doesn't stop there!
- We may add in functionality for authentication, logging requests, connecting to databases, and so much more.
- But always remember... we're **coding out these functionalities into our box!**

Questions?

Let's Get Coding!

Express Yourself...

Early Concerns...

Comment #1 - Concern

“This is all so confusing...”

Comment #1 - Response

And it will be for your employer as well.



Comment #2 - Concern

“I feel like I’m just copying and pasting”

Comment #2 - Response



- Backend code libraries mean YOU have to code less.
- Often you are just copying “best-practices” over and over again.

Express

Remind me again...

What is **Express**?

- In modern **web applications** there is a constant back-and-forth communication between the visuals displayed on the user's browser (**frontend**) and the data and logic stored on the server (**backend**).

Express.JS

Express

[Home](#) [Getting started](#) [Guide](#) [API reference](#) [Advanced topics](#) [Resources](#)

Express

Fast, unopinionated,
minimalist web framework for
[Node.js](#)

```
$ npm install express --save
```

Web framework for node to make

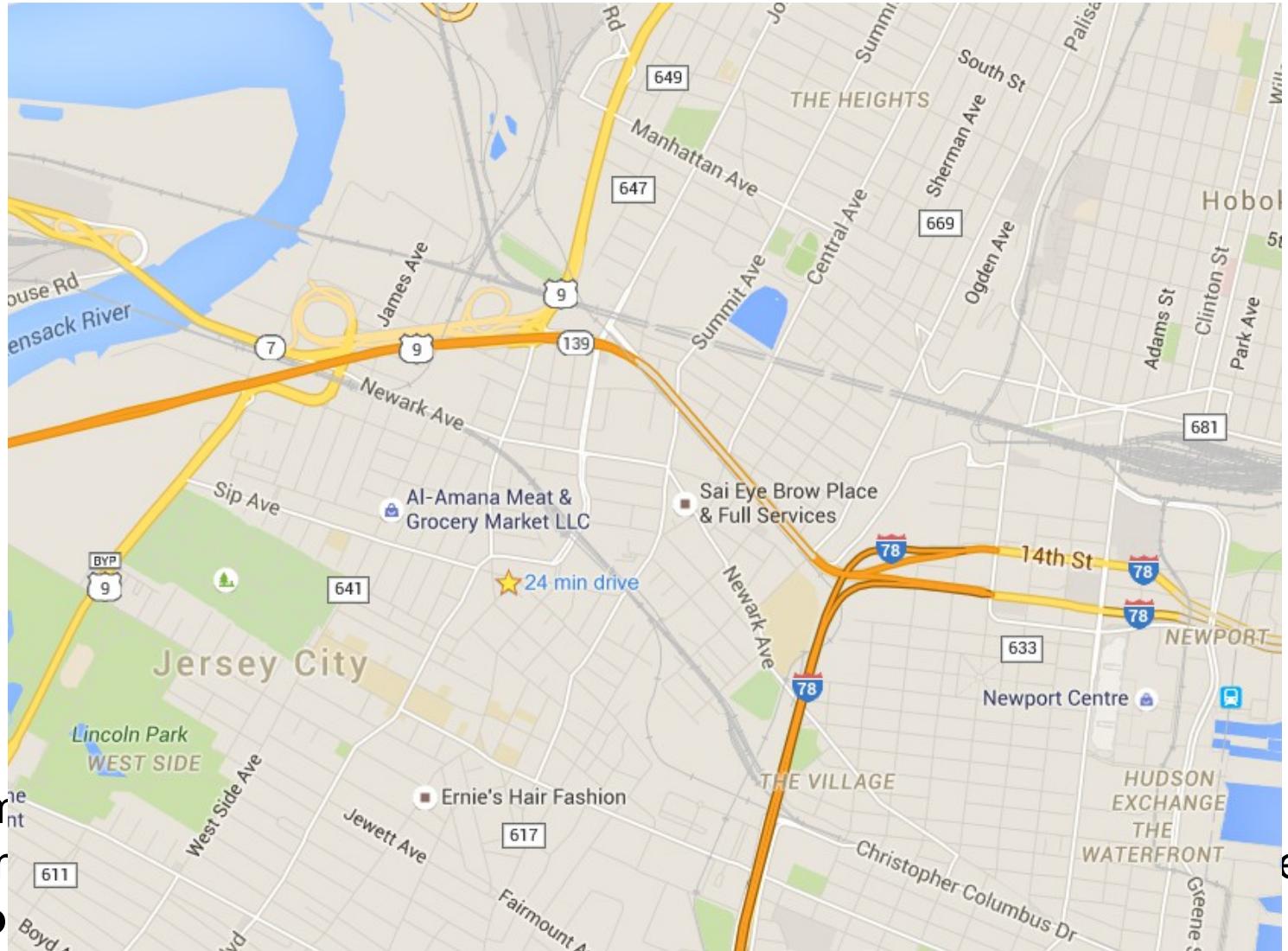
- In modern web applications there is a constant back-and-forth communication between the visuals displayed on the user's browser (**frontend**) and the data and logic stored on the server (**backend**).

Remind me again...

What is a **route**?

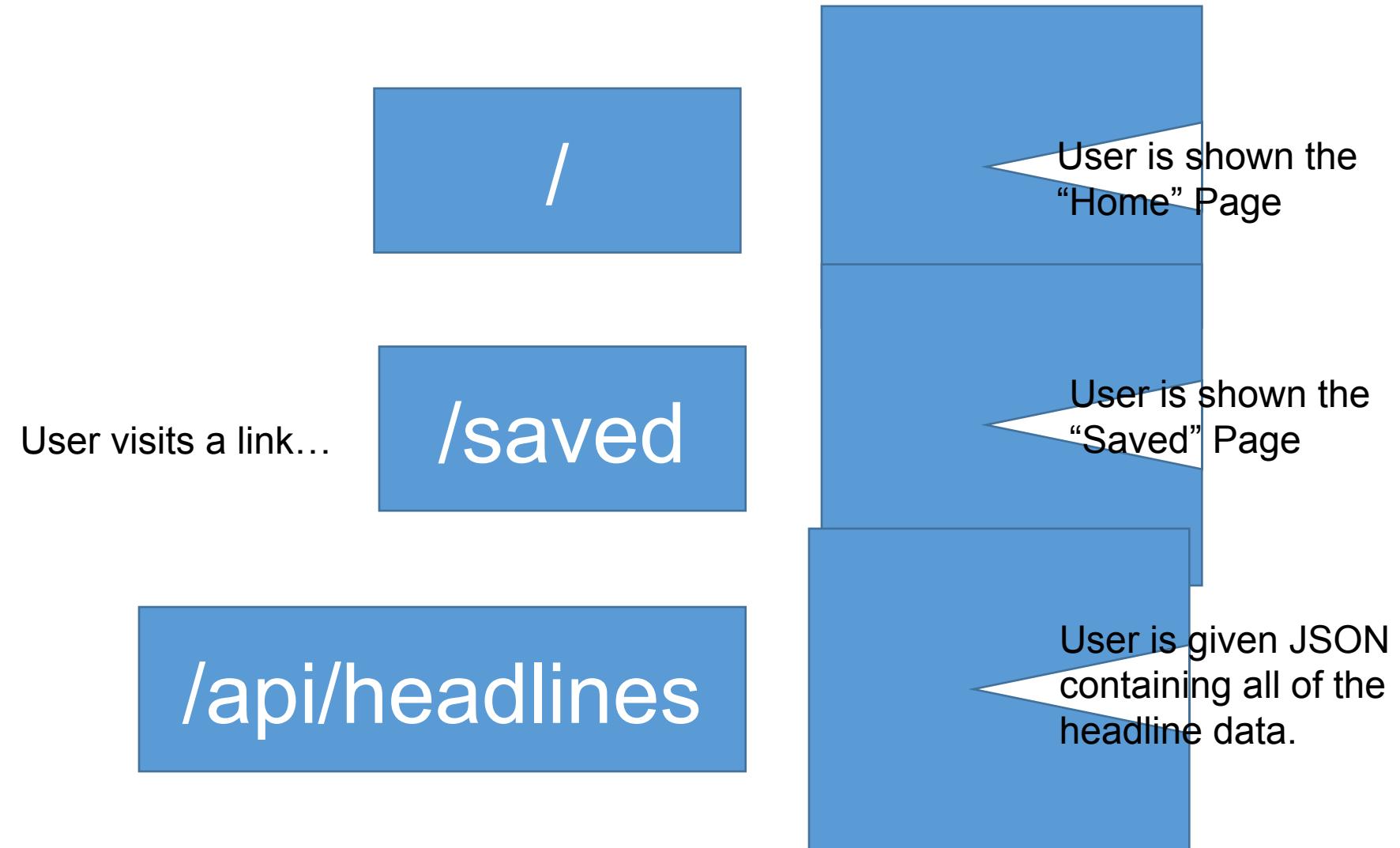
- In modern **web applications** there is a constant back-and-forth communication between the visuals displayed on the user's browser (**frontend**) and the data and logic stored on the server (**backend**).

Routes = Maps



- In m
com
(fro
er

Routes = Maps



Quick Example

The screenshot shows a web application with a blue header bar. On the left, it says "Mongo Scraper". In the center, there are three buttons: "Home", "Saved Articles", and a red button labeled "SCRAPE NEW ARTICLES!". Below the header, there is a large image of a vinyl record.

News Article Preview:

Headline: MONGO SCRAPER

Text Preview:

id's greenhouse gas emissions, dilution from fertilizers, habitat degradation. Under our current farmers use lights, ventilation, growing technologies to create an environment to grow cannabis, a lot of greenhouse emissions. At the precipice of a similar fate for agriculture in Canada, it's time to see if we can shift marijuana cultivation onto a more environmentally friendly path.

So far, growers have already begun to express concerns about the large-scale pot, envisioning an alternative model, similar to craft breweries, that consumers will appreciate as a healthy and friendly product for a higher price. It's clear that many British Columbians expect extra dollars for craft beer proves

Section Headings:

the federal government with a large-scale approach to Shoppers who have legalized

Body Text:

abandoned vessels. Transport Canada has abandoned or derelict vessels, undoubtedly an under-reported environmental hazard. Each is, at least, an eyesore and a source of environmental contamination.

The cost to remove an abandoned vessel ranges from small change to several hundred thousand dollars, depending on complexity. As many owners near their end, owners are tempted to dump their vessels in public waters. As litterbugs on land found at the dawn of anti-litter laws, penalties for abandoning vessels are needed to discourage the practice.

But many government agencies are involved: Transport Canada, the Coast Guard, federal and provincial environmental agencies, and local governments. Overlapping agencies dilute responsibility, cause inefficiency and foster inaction.

At least M-40 carries some political and moral weight in calling on the government to do something.

The bad news is that motions have no binding legal effect. Furthermore, the wording

Other Content Preview:

CHRISTINE SUTER SUPPORTS FOOD BANK

Christine Suter from C2Skymultisport and Dave Weston from Weston Vancouver worked together with Oppenheimer Whistler Half Marathon to support the Whistler Food Bank 5km fun run and walk in support of Whistler Community Services Society.

We raised \$1,000 and donated over 80lbs of food! Thanks to everyone for supporting our community!

Christine Suter
Whistler

KUDOS TO MUNI PARKS STAFF

I'm rehabbing from knee-replacement surgery and I walk Emerald Forest and Lost Lake Park trails daily. We sometimes fail to appreciate what a wonderful system of well-planned and beautifully constructed trails we have in our backyard.

Kudos to the trail building crews, parks planners, and Muni Council(s) that have

Article Summary:

How Whales Became the Planet's Biggest Animals

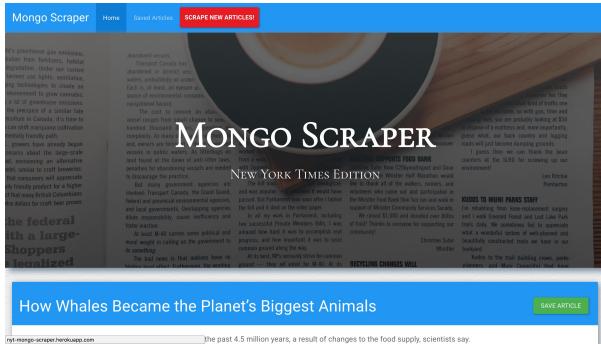
nyt-mongo-scraper.herokuapp.com the past 4.5 million years, a result of changes to the food supply, scientists say.

[SAVE ARTICLE](#)

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

Client-Server Communication (GET)

Client Browser



1) User visits “/”
route

(GET Request)

Node Server

```
[chriseckenrode:~/Sites/mongoscraper]$ node server
Listening on port:3000
mongoose connection is successful
```

Client-Server Communication (GET)

Client Browser



- 1) User visits /saved
- (GET Request)**
- 3) Server responds by providing HTML with saved articles

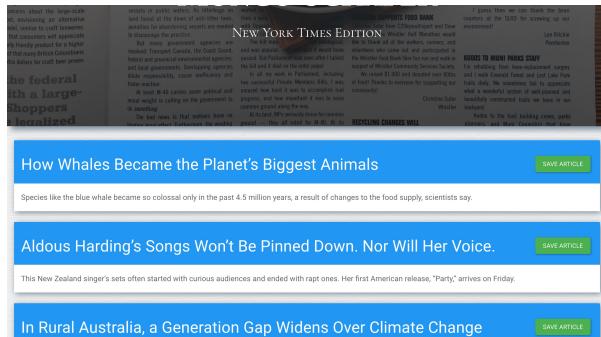
Node Server

A terminal window with a dark background and white text. It shows the command "[chriseckenrode:~/Sites/mongoscraper]\$ node server" and the output "[chriseckenrode:~/Sites/mongoscraper]\$ node server Listening on port:3000 mongoose connection is successful". There are also some small icons at the bottom of the terminal window.

- 2) Request triggers the code in the Server route. The server then finds the relevant HTML content and data

Client-Server Communication (POST)

Client Browser



1) User saves an article using "api/headlines"

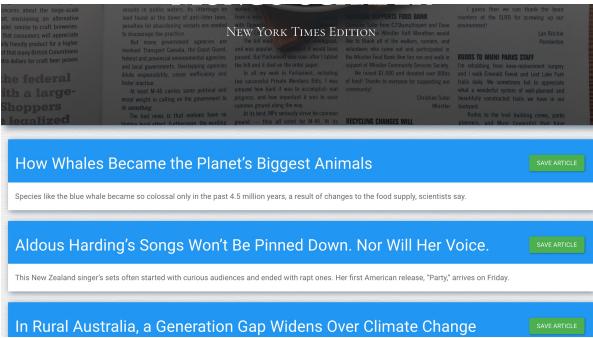
(POST Request)

Node Server

```
[chriseckenrode:~/Sites/mongoscraper]$ node server
Listening on port:3000
mongoose connection is successful
```

Client-Server Communication (POST)

Client Browser



1) User saves an article using "api/headlines"

(POST Request)

Node Server

The screenshot shows a web application titled "SAVED ARTICLES". It lists two saved articles from the "NY TIMES EDITION" section:

- How Whales Became the Planet's Biggest Animals**
Species like the blue whale became so colossal only in the past 4.5 million years, a result of changes to the food supply, scientists say.
[ARTICLE NOTES](#) [DELETE FROM SAVED](#)
- Janet Mock on Being Called a 'Trans Advocate'**
Species like the blue whale became so colossal only in the past 4.5 million years, a result of changes to the food supply.
[ARTICLE NOTES](#) [DELETE FROM SAVED](#)

3) “Saved” articles are loaded on the saved page.

2) Request triggers the code in the Server route. The server then sets the article as “saved” in the database.

Activity Time!
