Server-Side!

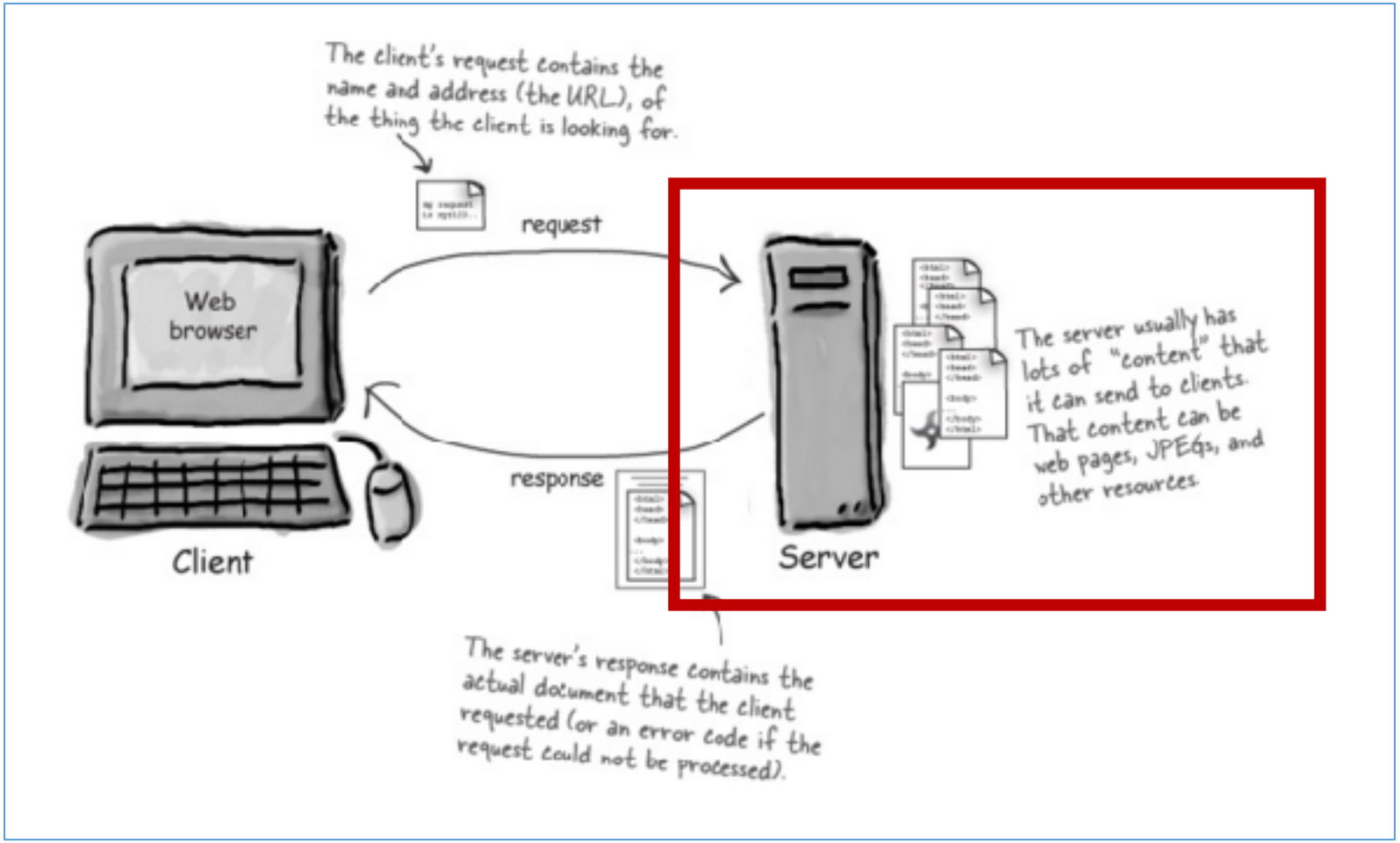
The Coding Bootcamp

So Let's Begin…

Remind me again…

What is a **server**?

Server Definition



Server:

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

Remind me again…

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

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.

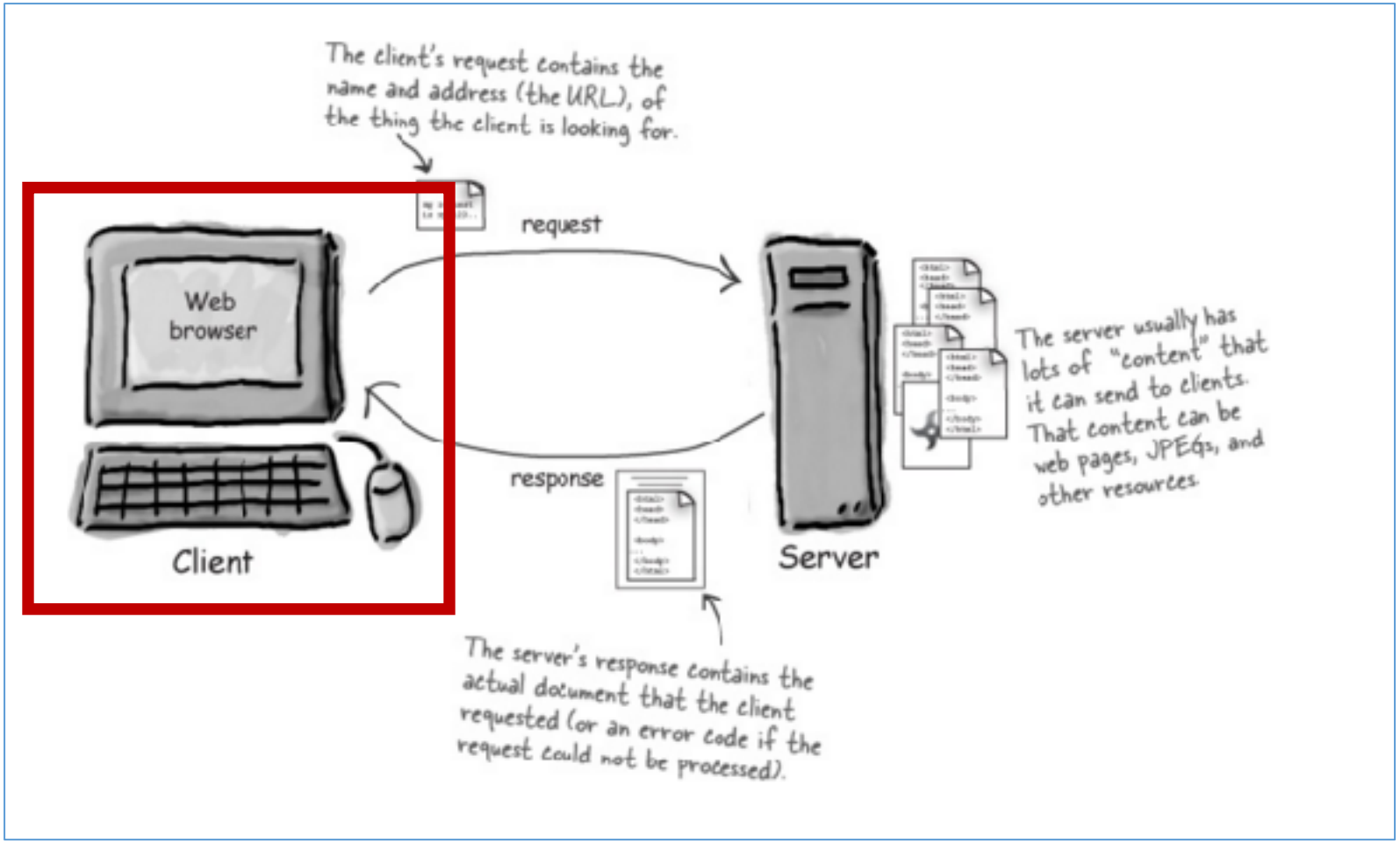
Server-Side Code in Action!

- 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**?

Client Definition



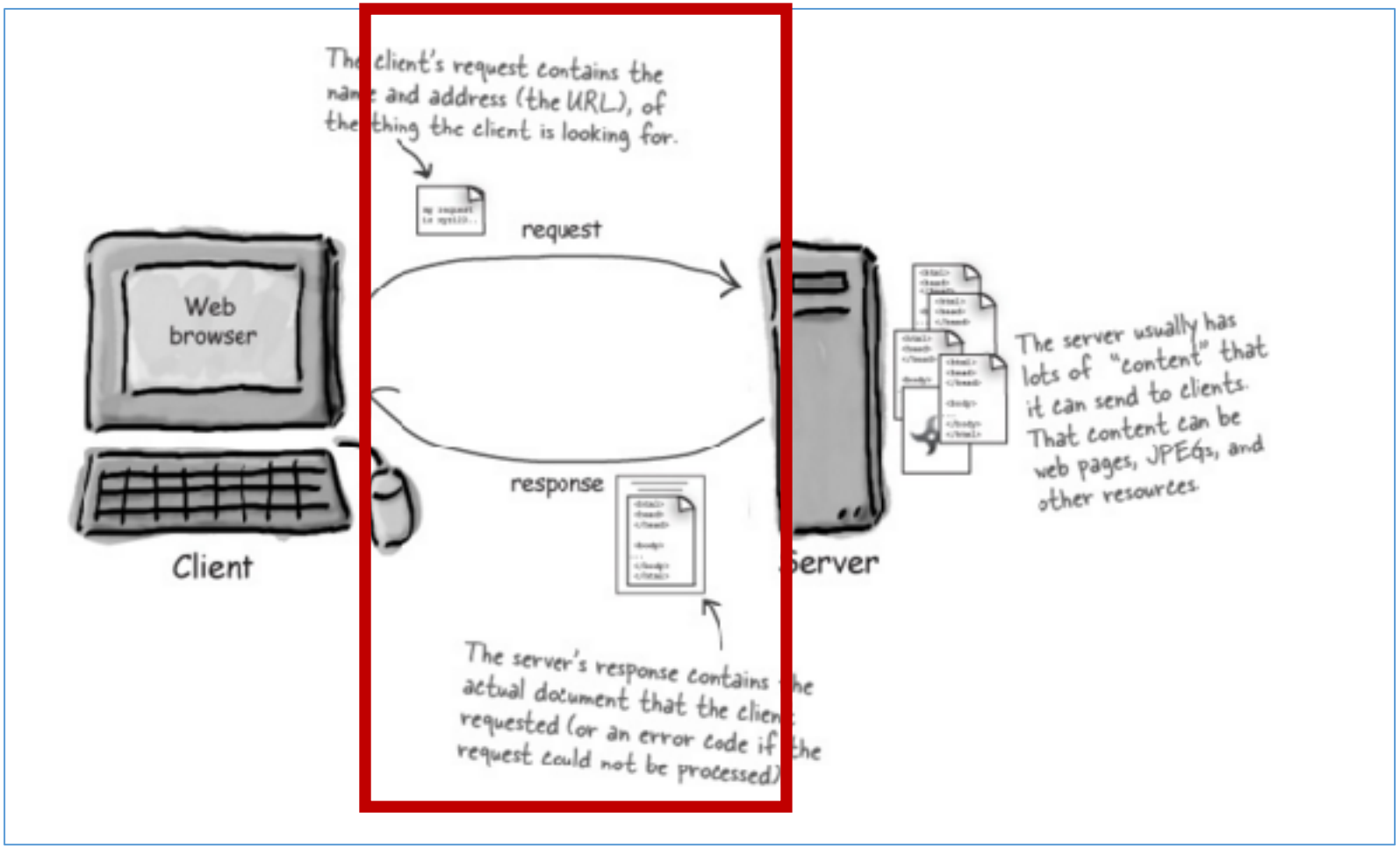
Client:

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

Bonus Question:

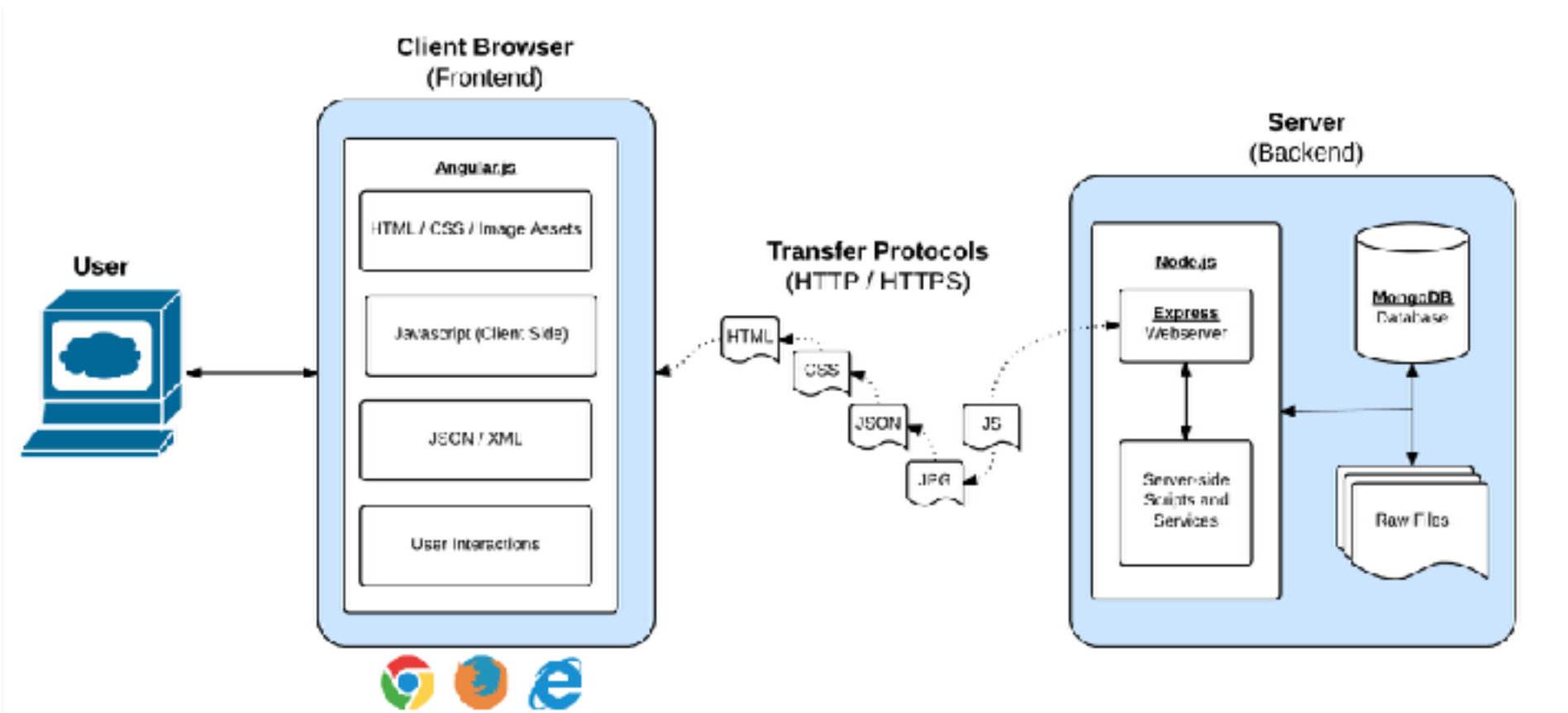
How do the client and server **communicate** with one another?

HTTP Definition



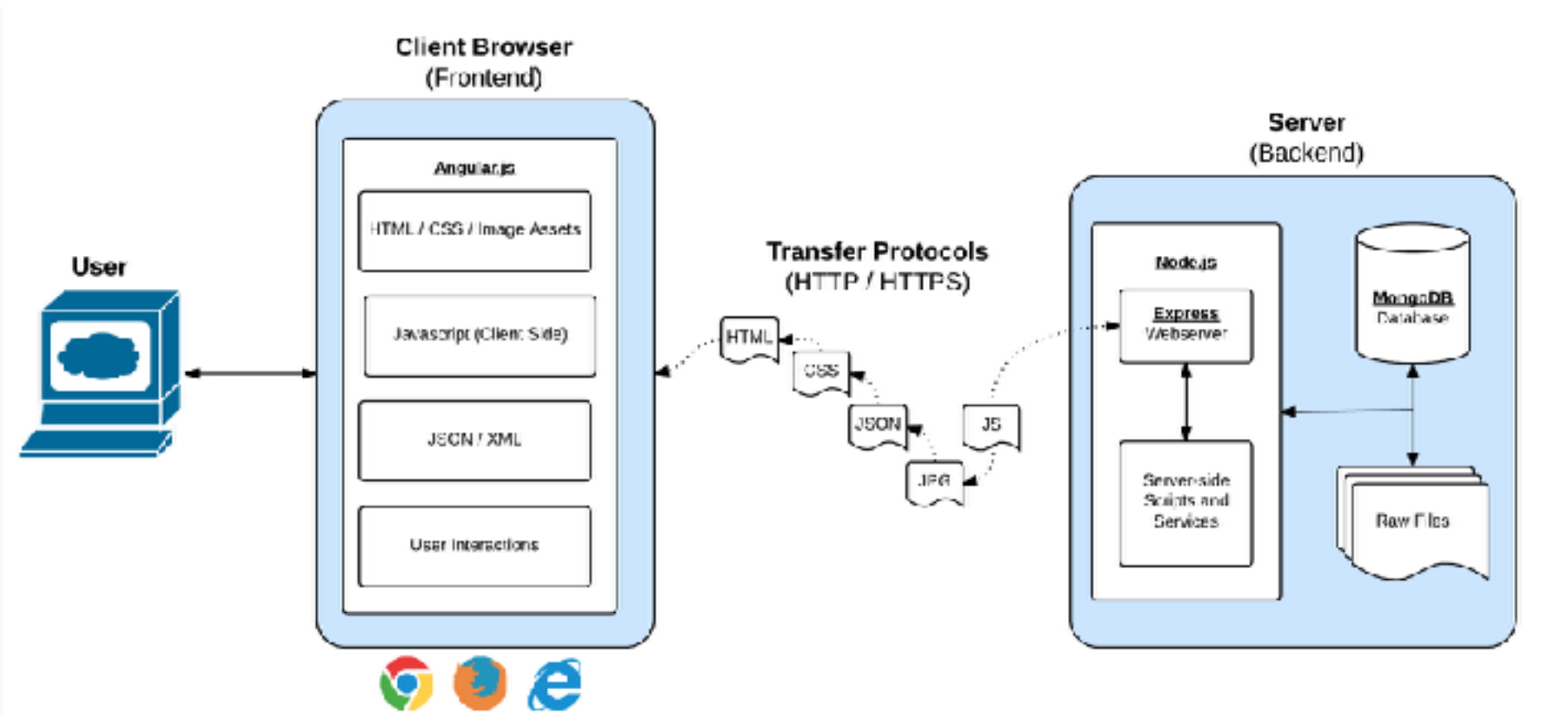
Clients and Servers communicate back and forth using a series of understood communications defined by **HTTP / HTTPs**.

Full-Stack Development



- 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).**

Full-Stack Development



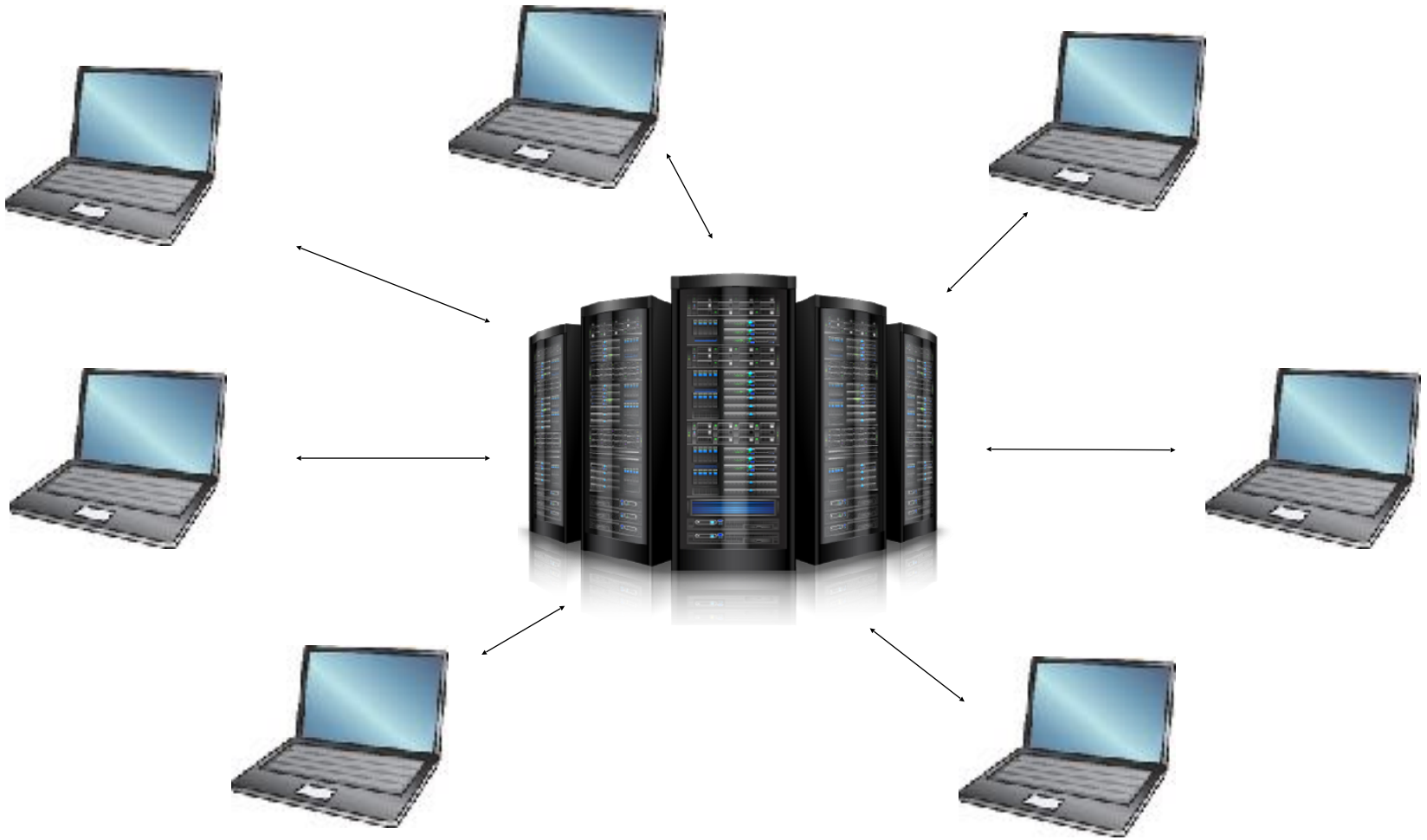
- In a way think of this as being **two distinct machines.**
- A server is one machine and the client is a second machine.

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**?

Where Do Servers Live?

- Servers live in dedicated hardware intended to handle ALL the requests and responses of many clients.

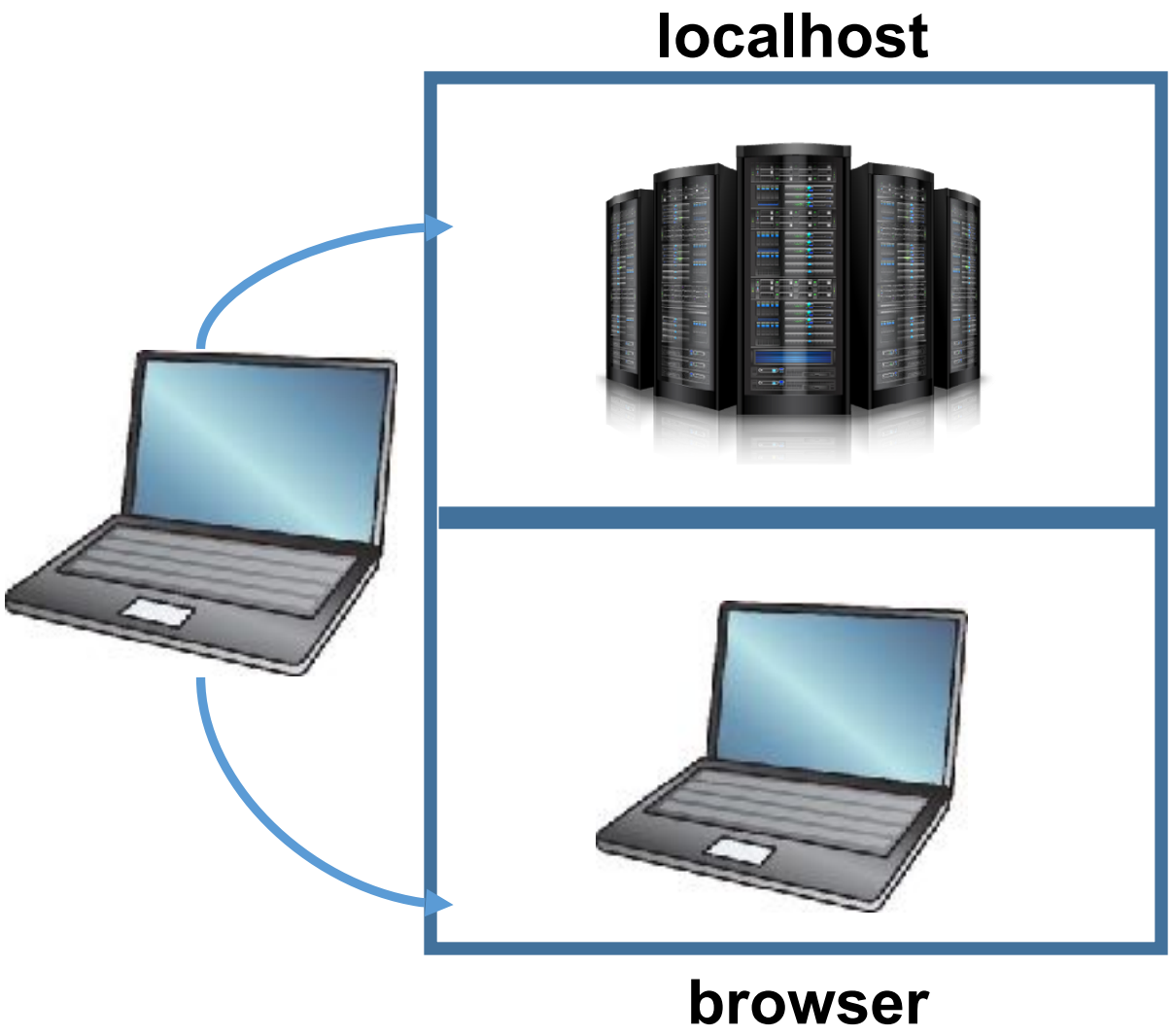
- Servers can live most often live on cloud platforms like AWS, Heroku, Google Cloud, etc.

Google Cloud Platform

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***.

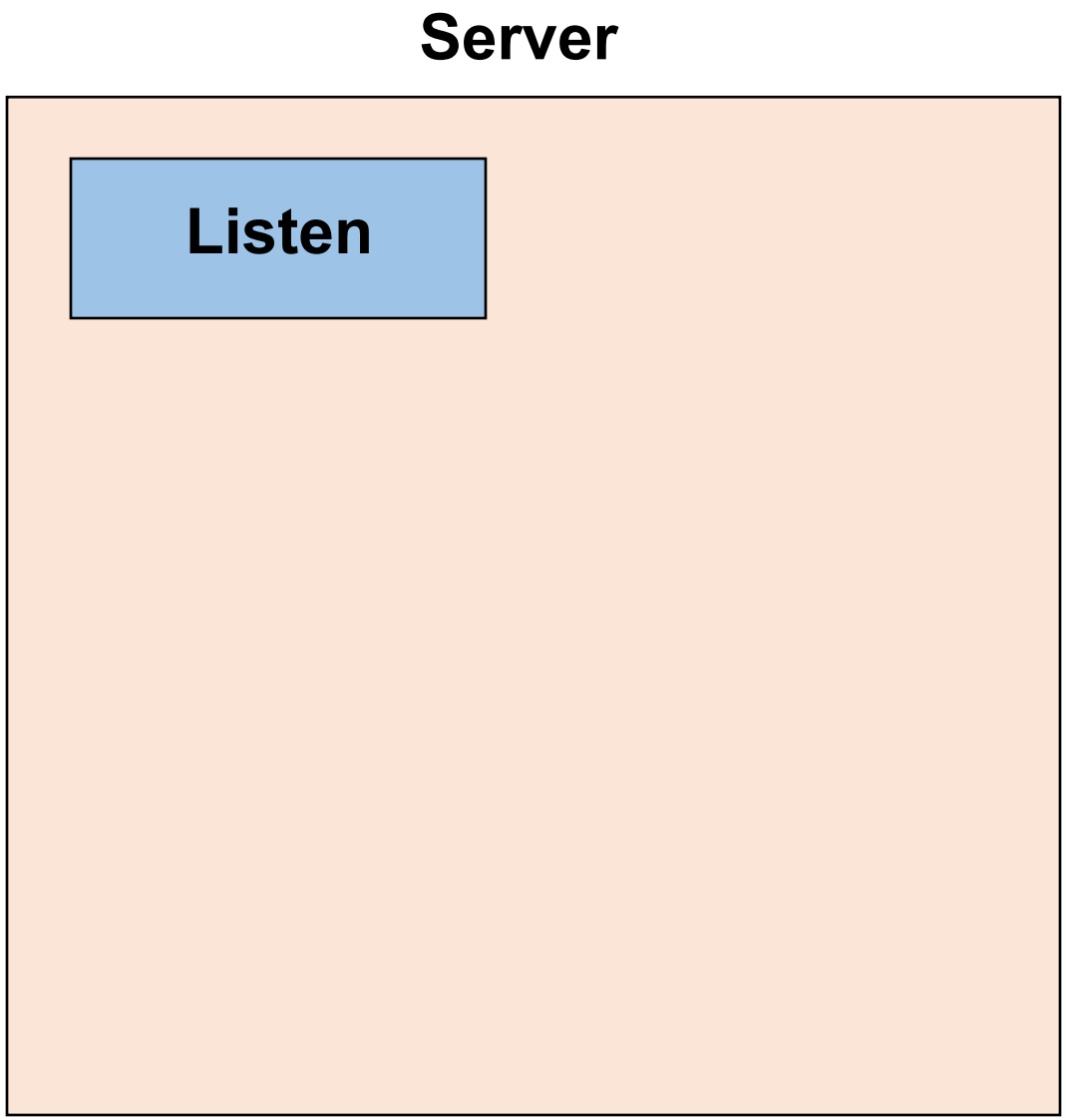
- It’s important to note that even though you pay for server-side hardware, you still need to create the code that goes inside.

- **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

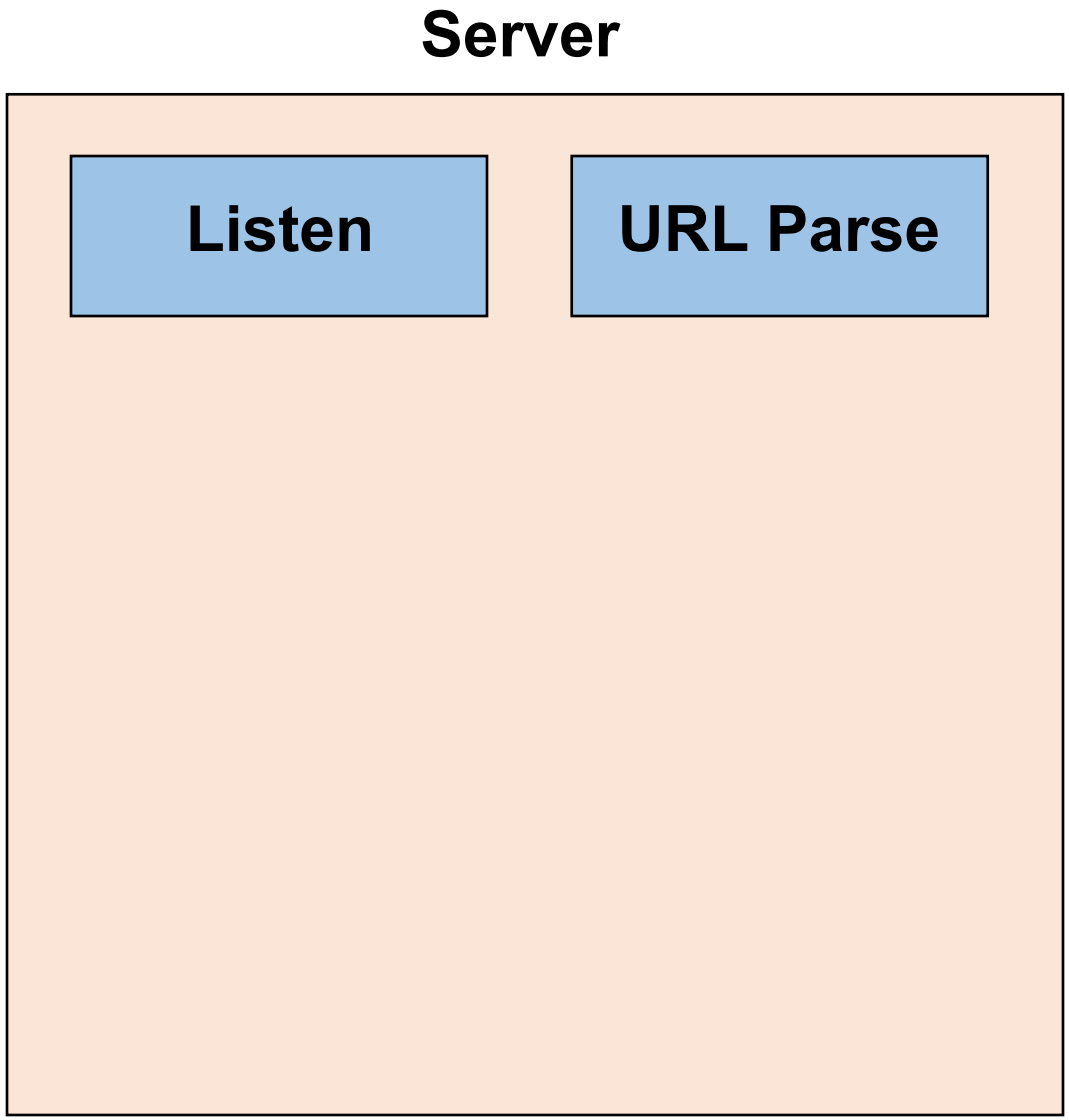
- Throughout this week… imagine your server to be a big, empty box.
- We will be adding code snippets and modules to give our big, empty box the powers to ***do*** stuff in response to all the requests that come in.

Inside the Box: Connections



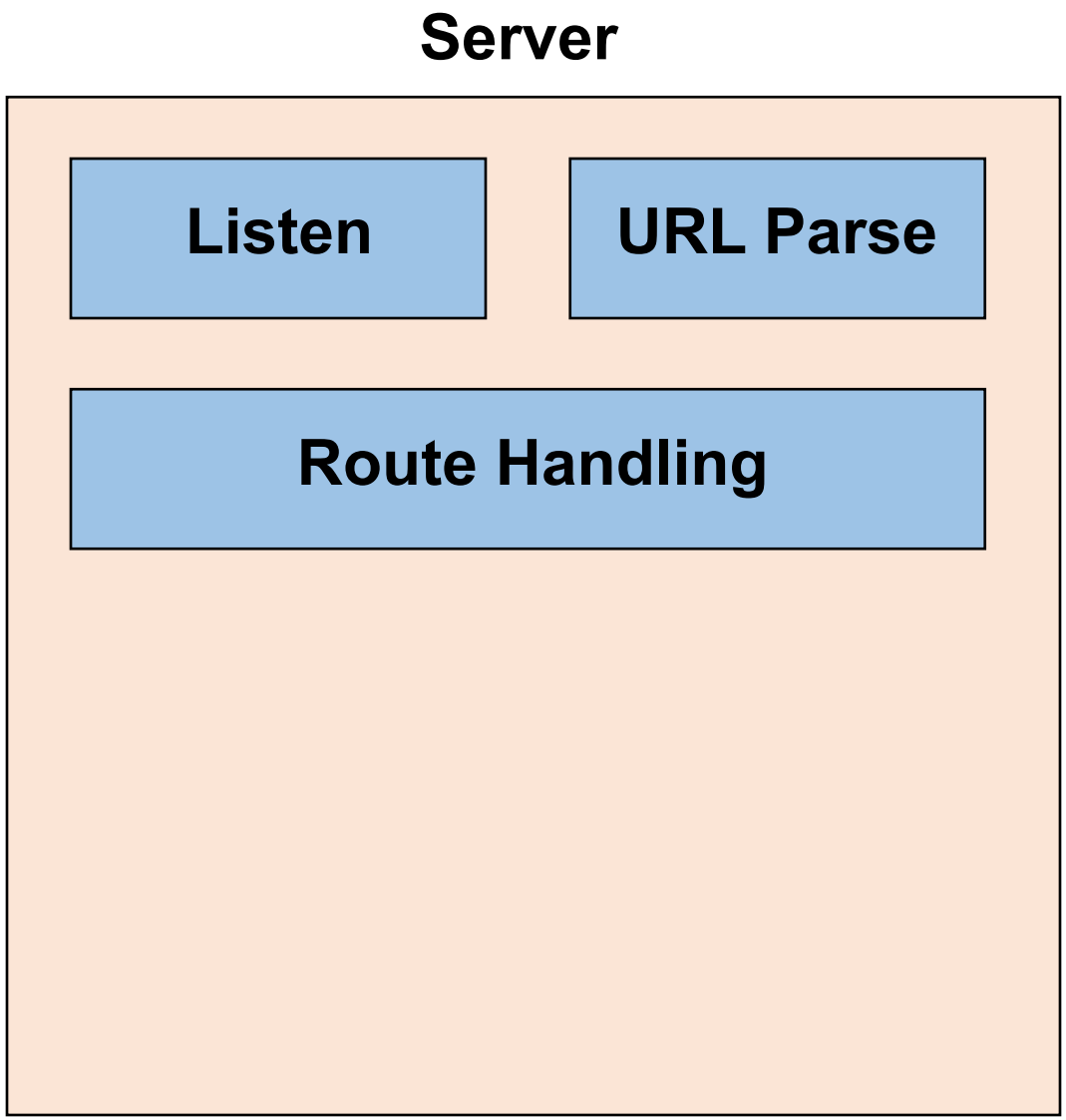
- We'll add listener such that the server can "begin" listening for requests.

Inside the Box: Parsing



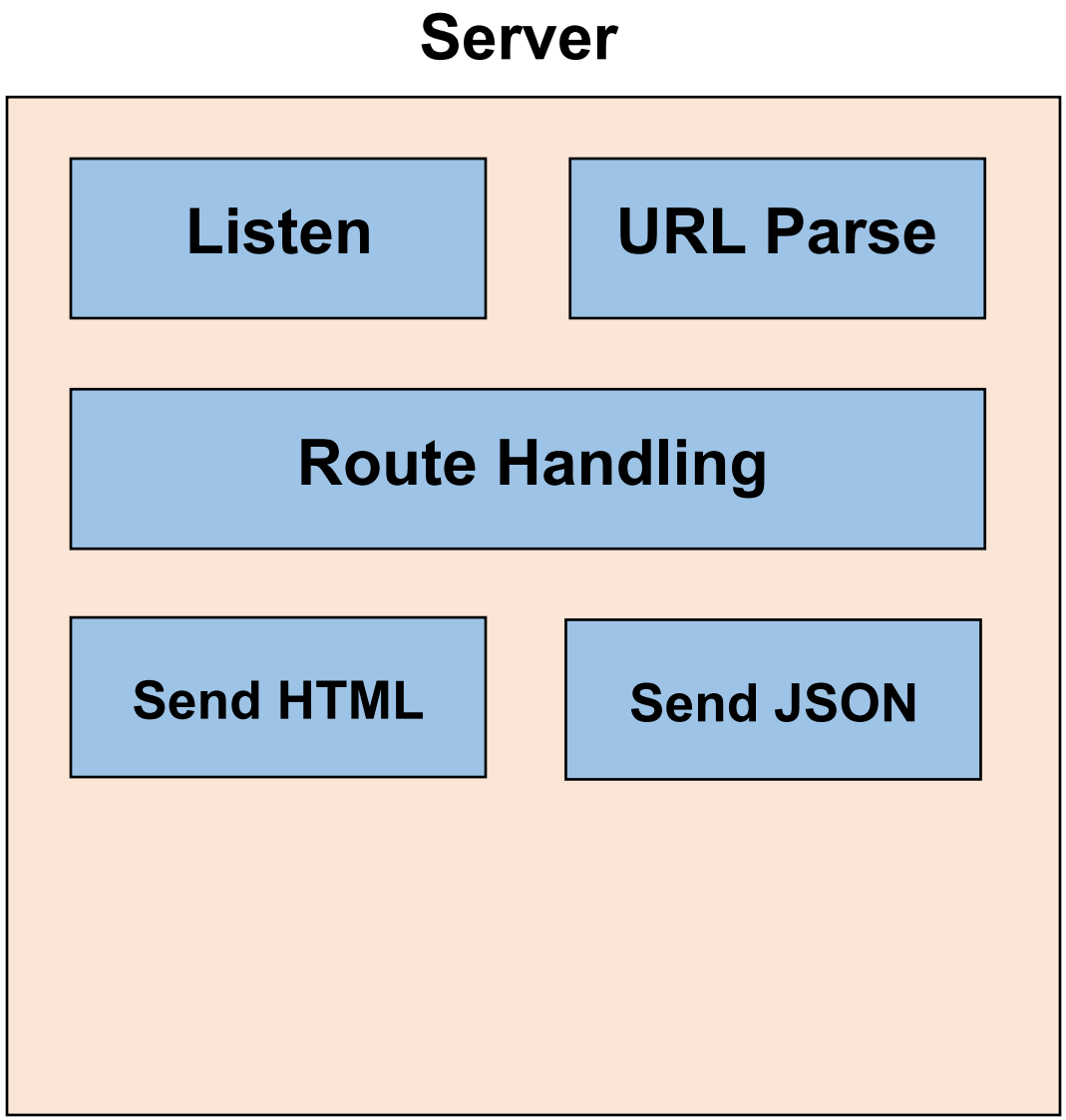
- We'll give our server the ability to "parse" URLs that the user requests.

Inside the Box: Routing



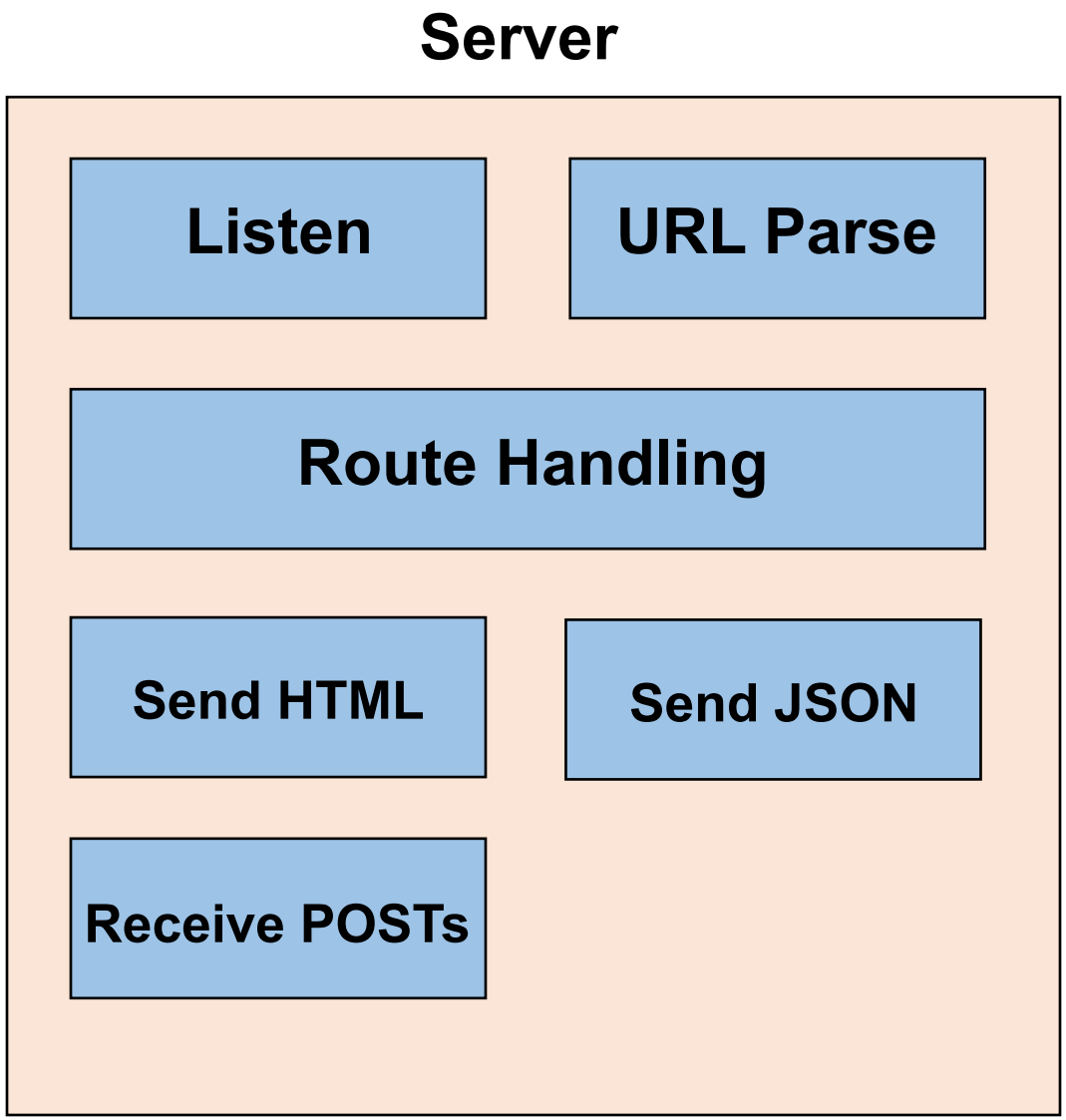
- Then based on the URL's keywords, our server will be able to **route** (or direct the flow of logic to initiate other processes)

Inside the Box: Sending Files



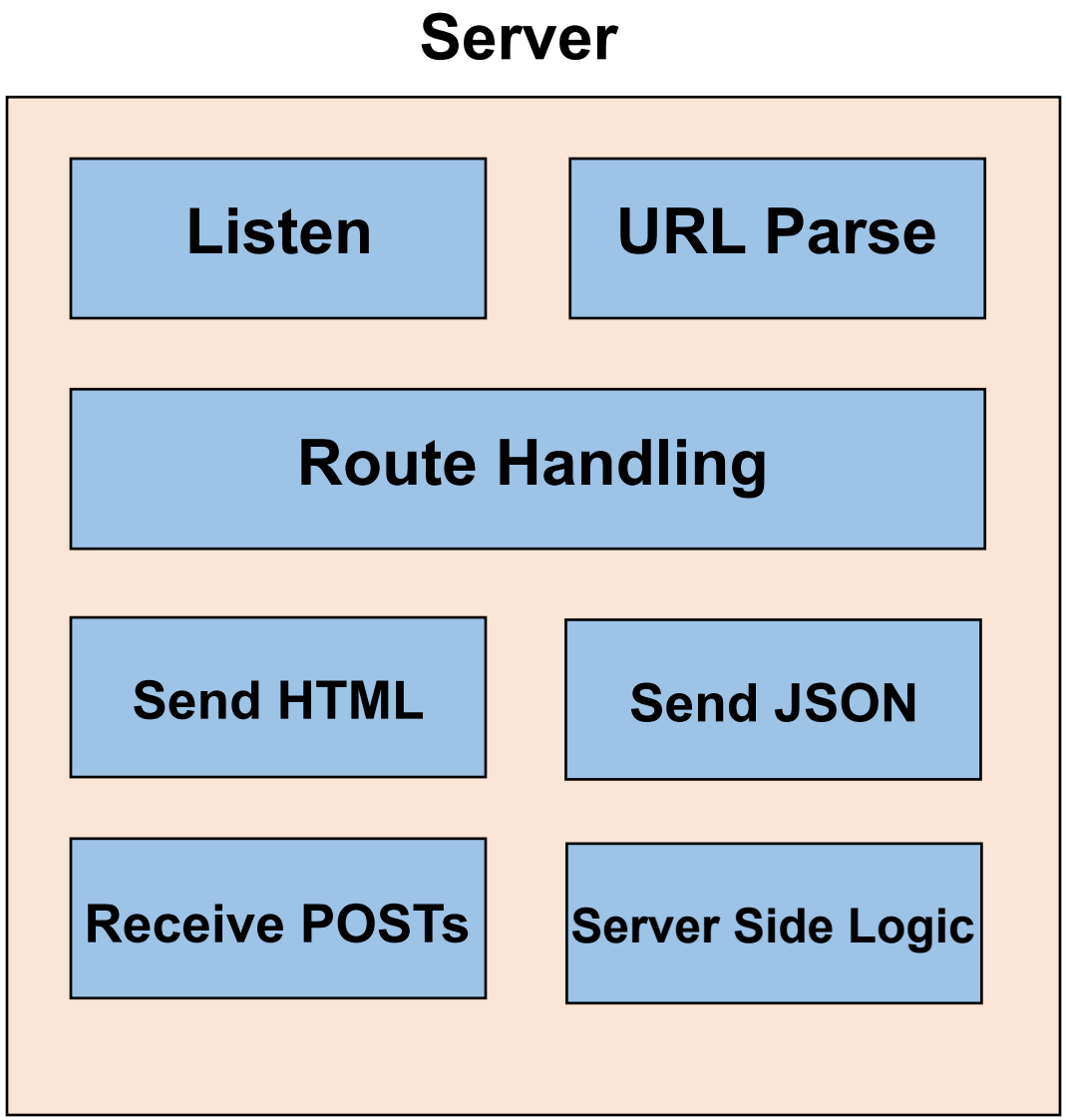
- This subsequent process may be to send an HTML file to be rendered or to send a JSON file to be rendered…

Inside the Box: Receiving Posts



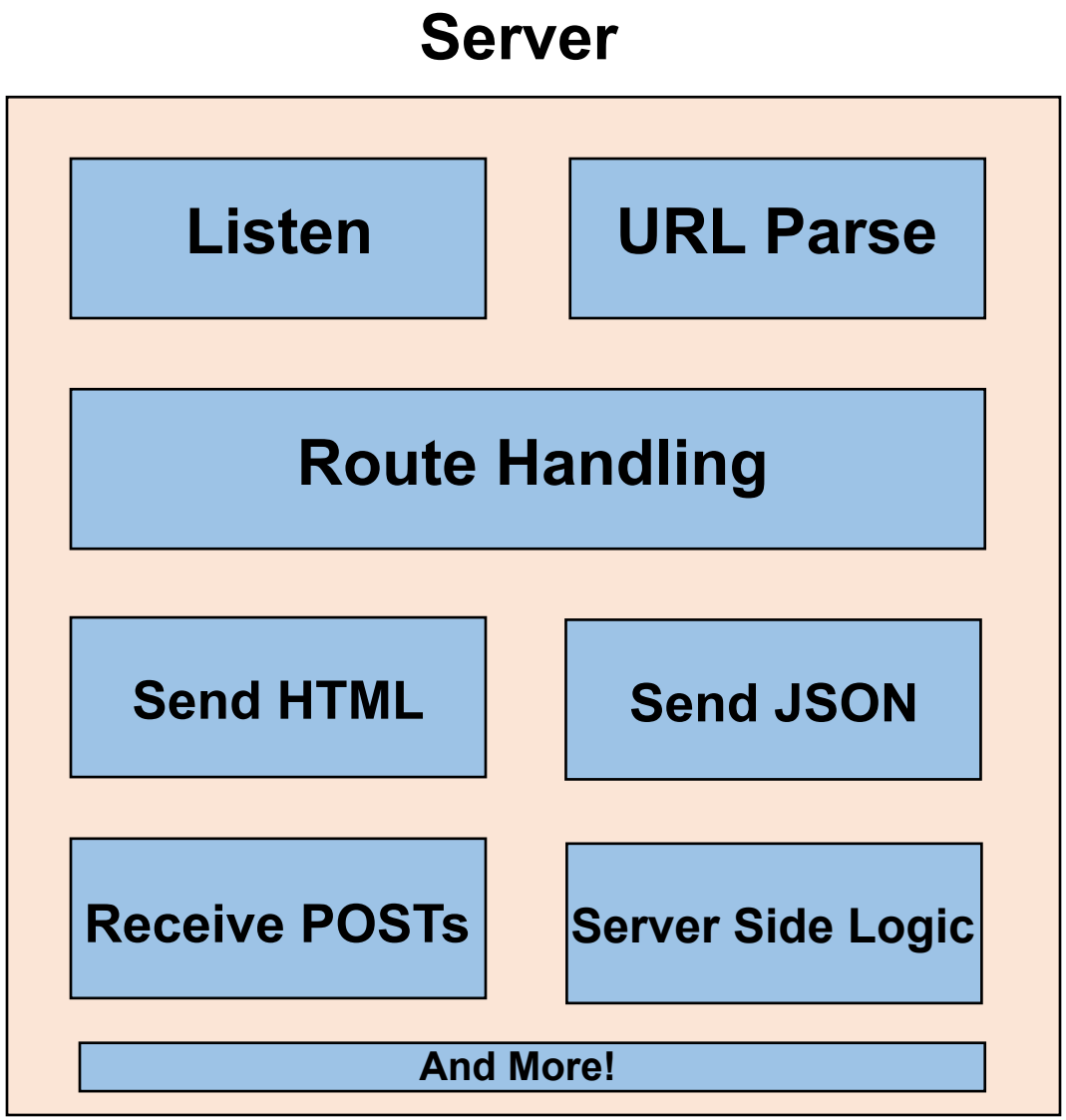
- We may also have a new module to handle receiving user's POST requests (i.e. the data they send the server)

Inside the Box: Performing Logic



- 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!



- 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!