New to Node

Web Development Boot Camp

Lesson 10.1

Project Week

Instructor Feedback

Seriously? Mind. Blown.

A Little Perspective on How Far You’ve Come:

Last time I said this, you barely knew what a `<div>` was.

Instructor Feedback

Things I’ve noticed people doing incredibly well:

- Stunning front-ends
- Amazing execution of challenging concepts (no one took the easy road)
- Fantastic explanations of your code and technology
- Basically everything!

Share your hard work with the world—find ways to showcase your code on social media and the web.

Close Out Your Projects Well

Take your finished project to the next level

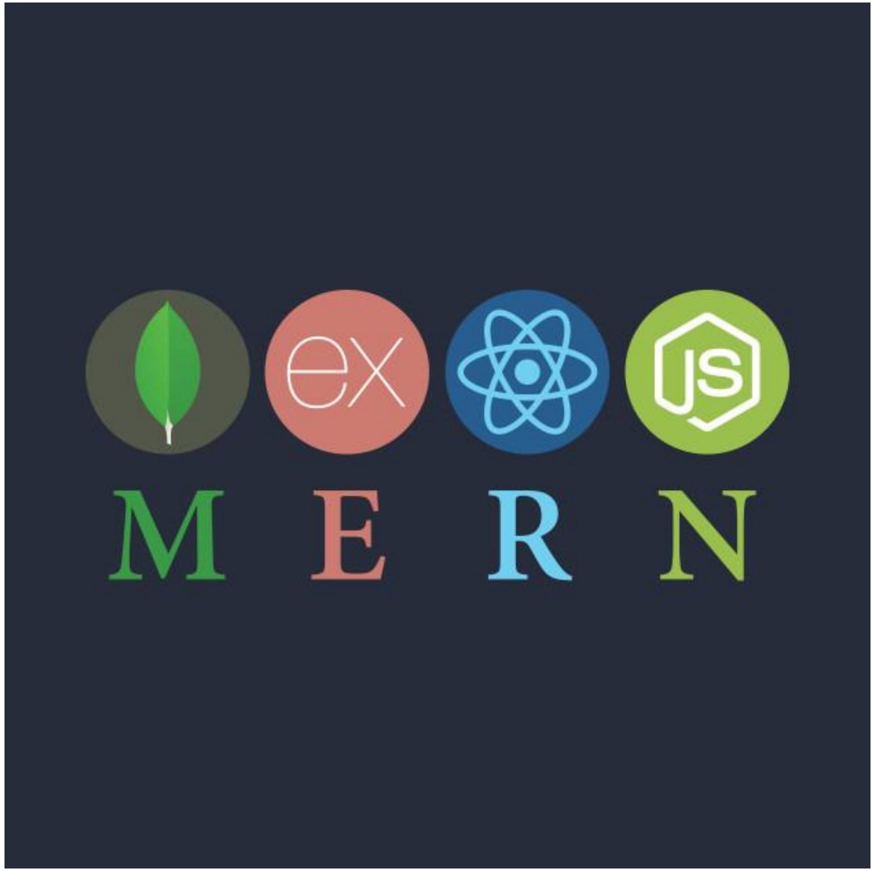
- Create READMEs for your code on GitHub.
- Use a custom domain URL.
- Create default, working “test cases.”
- Consider writing a blog article or making a video that builds from scratch.
- Fork/star other people’s code.

The Mystery of Back-End

FOCUS

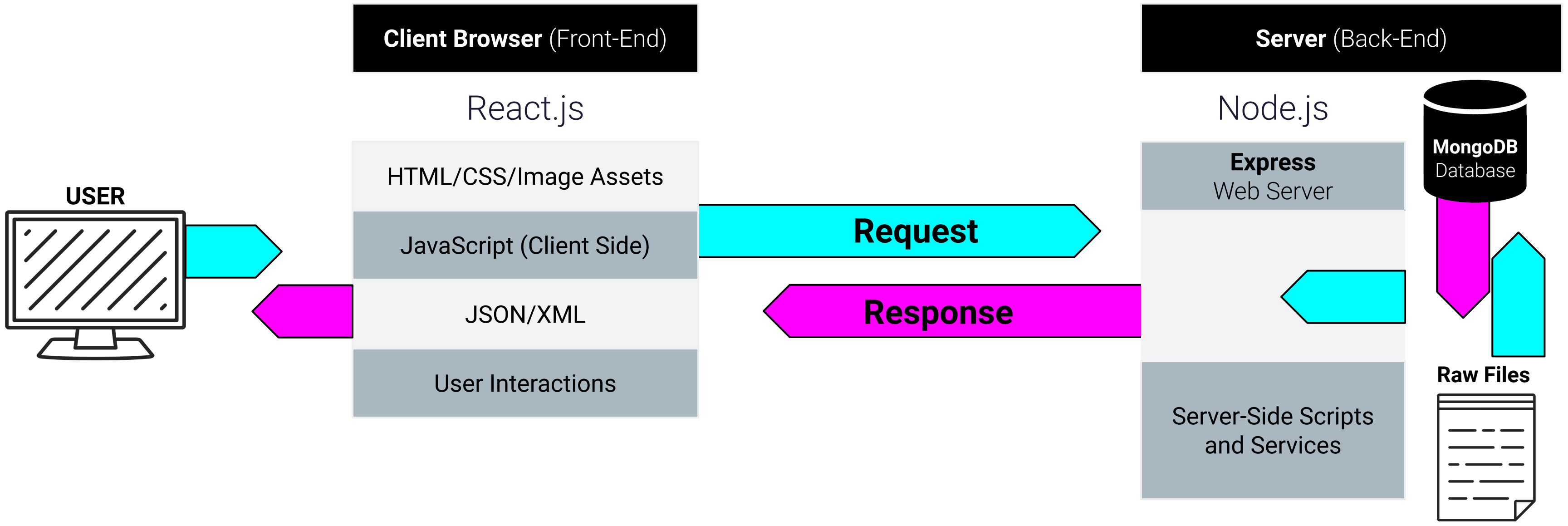
This next stuff is important!

Full-Stack Development?



Full-Stack Development

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



The “**Magic**” of **YouTube**

Key Question:
Examples of Server-Side Code?

Server-Side Code in Action!

Examples of server-side code:

- APIs that parse URL parameters to provide selective JSONs
- Dynamically render HTML using a template engine
- 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 that are relevant to your searches on other sites

Critical Question:
What Is a Server?

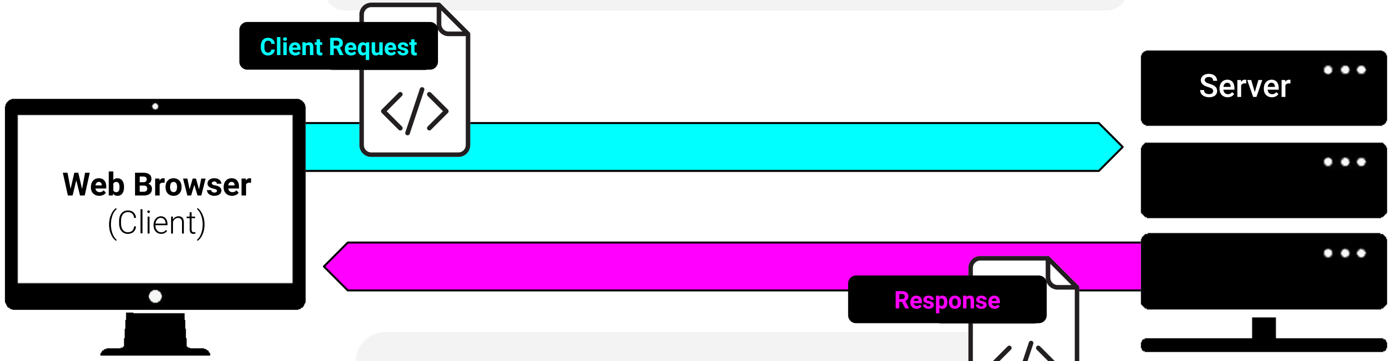
Definition of Server

A **web server** takes a client request and gives something back.

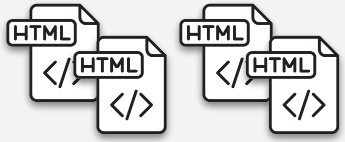
> *A web browser lets a user request a resource. The web server gets the request, finds the resource, and returns something to the user. Sometimes the resource is an HTML page. Sometimes it's a picture. Or a sound file. Or even a PDF document. Doesn't matter—the client asks for the thing (resource) [or action] and the server sends it back.*
>
> *...When we say "server," we mean either the physical (hardware) or the web server application (software) [that actually runs the server commands].*
>
> **— Kathy Sierra, *Head First Servlets and JSP (O'Reilly, 2004)***

Server Definition

The client's request contains the name and address (the URL) of what the client is looking for.



The server's response contains the actual document that the client requested (or an error code if the request could not be processed).



The server typically has lots of content that it can send to clients. That content can be web pages, JPEGs, and other resources.

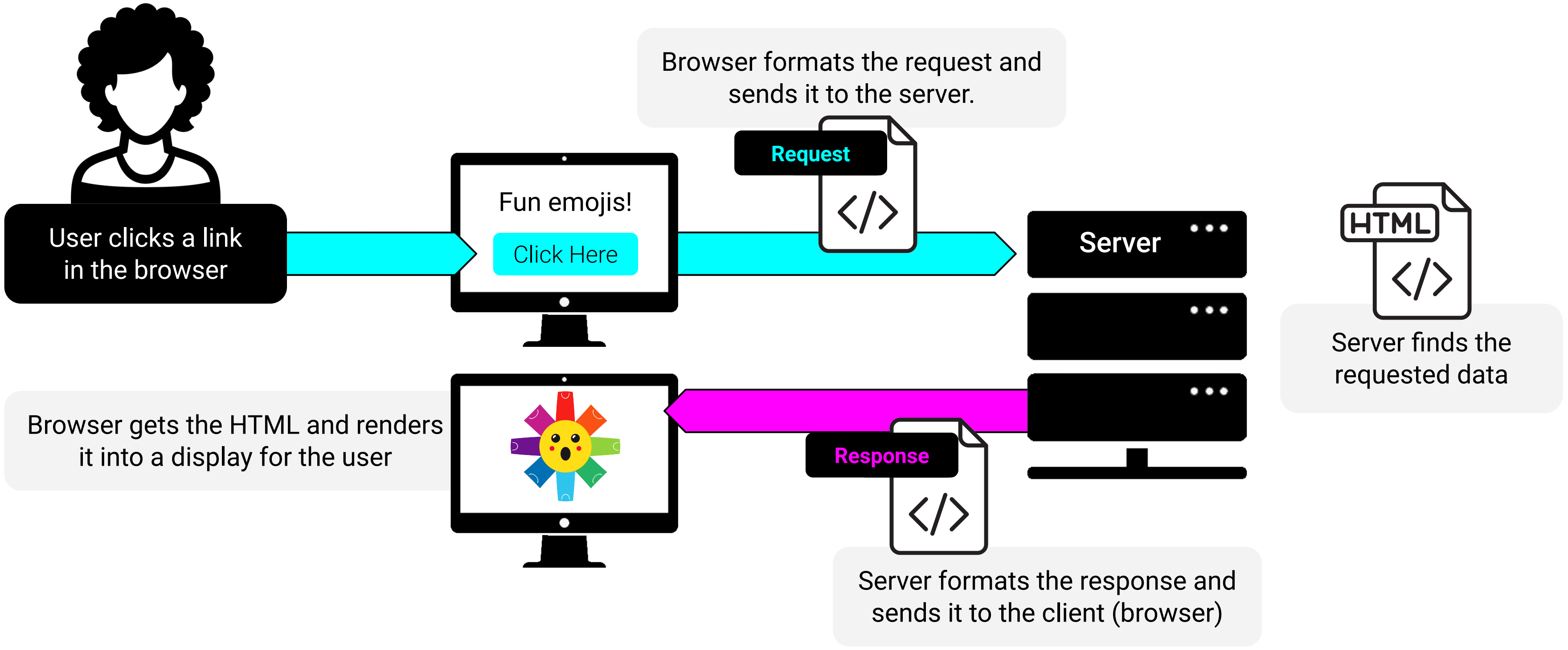
Definition of Web Client

A **web client** lets the user request something on the server and then shows the result (response) from the server.

When we talk about client, though, we usually mean both (or either) the human user and the browser application. The browser is the piece of the software that knows how to communicate with the server. The browser's other big job is interpreting the HTML code (sent by the server) and rendering the web page to the user.

— Kathy Sierra, *Head First Servlets and JSP (O'Reilly, 2004)*

Web Client Definition


Browser formats the request and sends it to the server.
Request
User clicks a link in the browser
Fun emojis!
Click Here
Server
HTML
Server finds the requested data
Browser gets the HTML and renders it into a display for the user
Response
Server formats the response and sends it to the client (browser)

Yay!

You made it through the ultra-important stuff!

Node.JS

Key Question:

So What Is node js?

Definition of NodeJS

Node.js is an open source, cross-platform JavaScript runtime environment designed to be run outside of browsers.

It is a general utility that can be used for a variety of purposes including asset compilation, scripting, monitoring, and **most notably as the basis for web servers.**

Activity:

Take a few moments to research five companies that actively use NodeJS in production.

Suggested Time:

Why Use NodeJS as a Server?

NodeJS reuses Javascript, which means a front-end Javascript developer can build an entire server themselves.

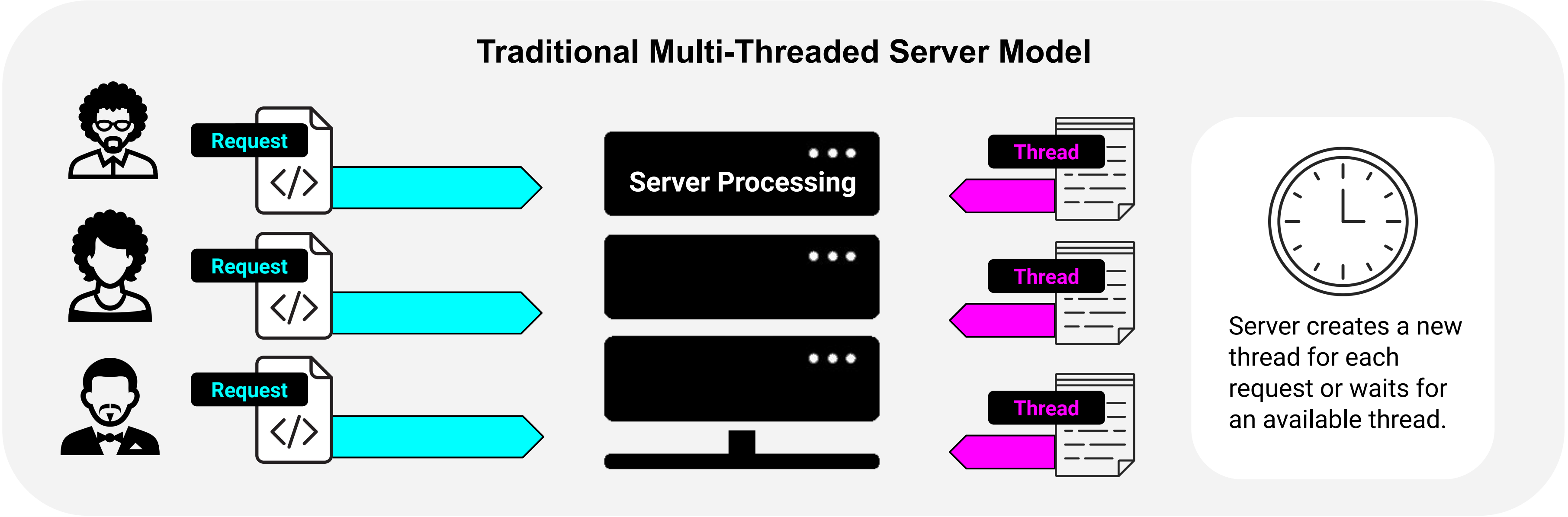
It's easily extendable. Numerous plugins exist to expand the capabilities of Node.

Fast implementation, which allows you to create an entire working server with only a few lines of code.

Single-Threaded Asynchronous Model, which means it can handle multiple requests simultaneously and not get bottlenecked.

Synchronous Threading

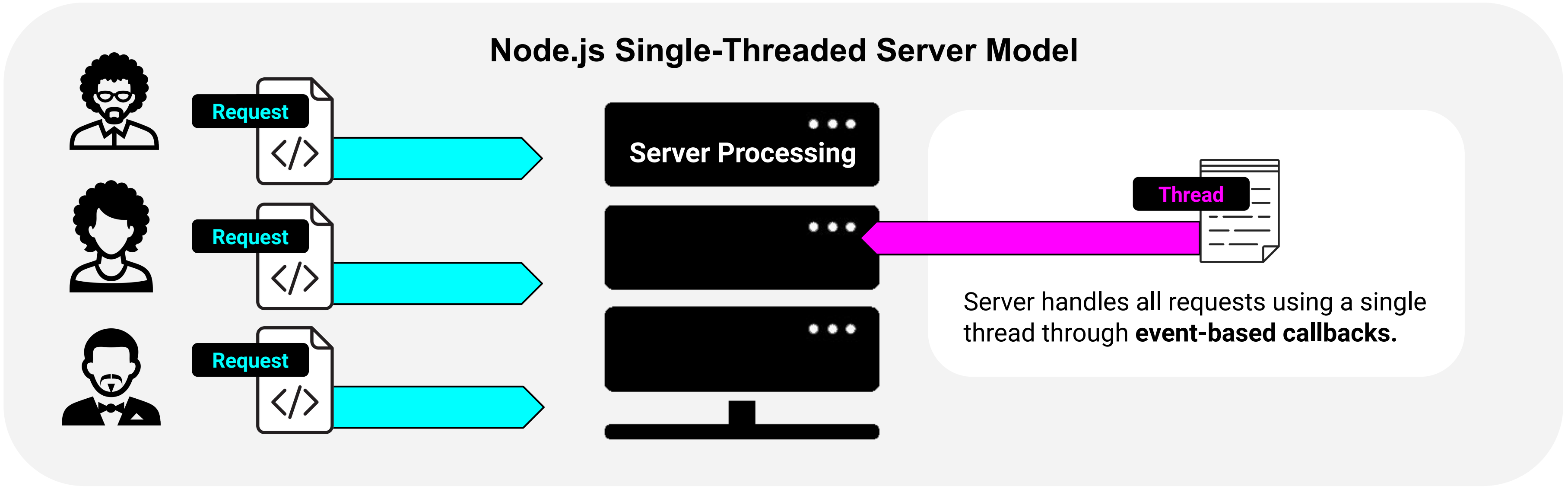
In synchronous threading, each request requires its own thread. No other request can pass through that thread until it completes. This can create bottlenecks.



Server creates a new thread for each request or waits for an available thread.

Asynchronous Threading (the Node Way)

In Node-based asynchronous threading, a single thread is used throughout. Each thread is “put to the side” using callbacks and responded to when ready. Because of this, there is no limit on the number of requests that can be responded to and there is no bottleneck.



Server handles all requests using a single thread through **event-based callbacks.**

<Time to Code>