Introduction to Node.js

Web Development Boot Camp

Lesson 9.1

Kanyed...

Most Awe-Inspiring

It causes awe; it is spectacular and magnificent.

FIRST PLACE: Team Frasier

RUNNER-UP: Team Seinfeld

Lesson 9.1

Most Useful

It serves a practical and useful purpose.

FIRST PLACE: Team Full House

RUNNER-UP: Team Frasier

Lesson 9.1

Most Creative

It is imaginative and contains original ideas.

FIRST PLACE: Team Frasier

RUNNER-UP: Team Everybody Loves Raymond

Lesson 9.1

Most High-Level of Tech

It has 1 or more really technically complex components.

FIRST PLACE: Team Frasier

RUNNER-UP: Team Seinfeld

Lesson 9.1

Best UI/UX

It has great looking features and a easy to understand user experience.

FIRST PLACE: Team Frasier

RUNNER-UP: Team Seinfeld

Lesson 9.1

Funniest

It has or shows a sense of humor.

FIRST PLACE: Team Friends

RUNNER-UP: Team Frasier

Lesson 9.1

Most Disruptively Innovative

It has innovative or groundbreaking features.

FIRST PLACE: Team Everybody Loves Raymond

RUNNER-UP: Team Frasier

Lesson 9.1

Most Socially Conscious

It has innovative or groundbreaking features.

FIRST PLACE: Team Full House

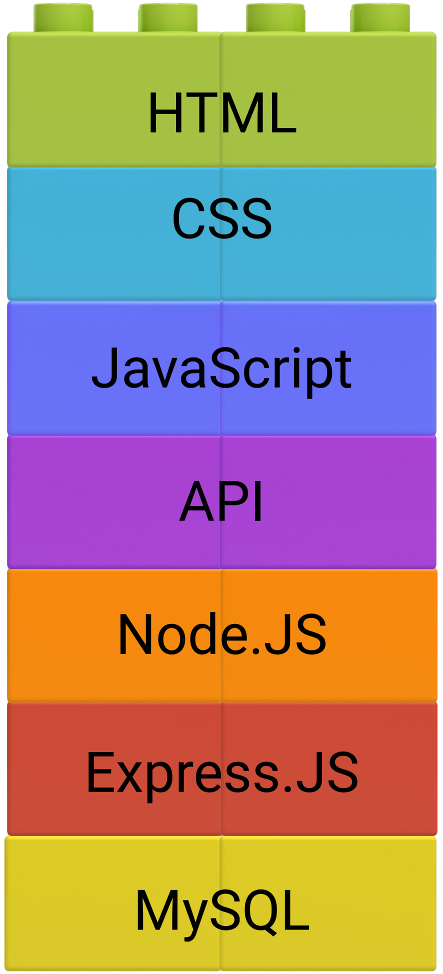
RUNNER-UP: Team Frasier

Lesson 9.1

What Is Full-Stack Web Development?

Full-Stack Web Development

Full-stack web development encompasses the suite of tools required to build both the front and back ends of a web application.

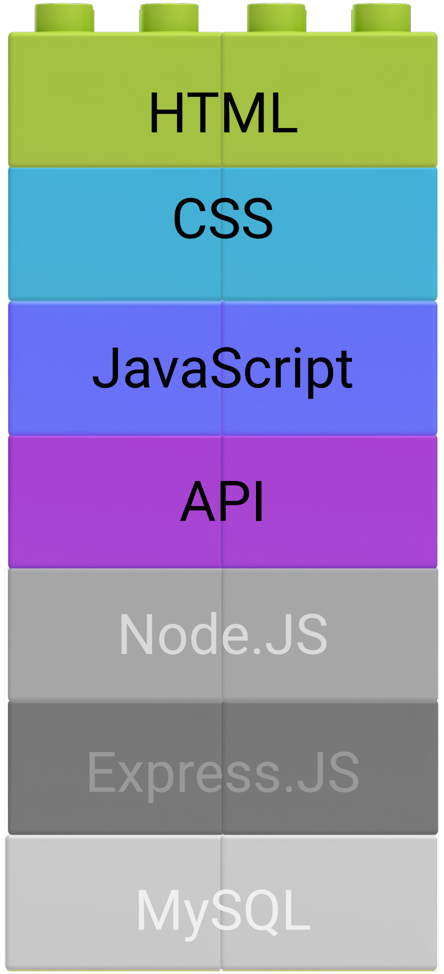


How Much of the Stack Do We Know?

Client-Side

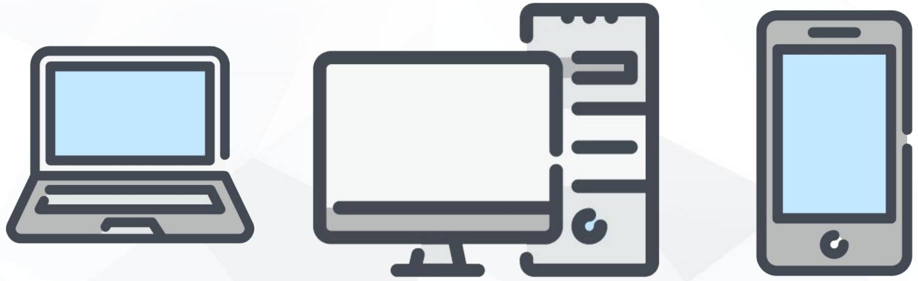
- HTML
- CSS
- JavaScript
- APIs

Further reading:
wikipedia.org/wiki/Client-side



What Is a Client?

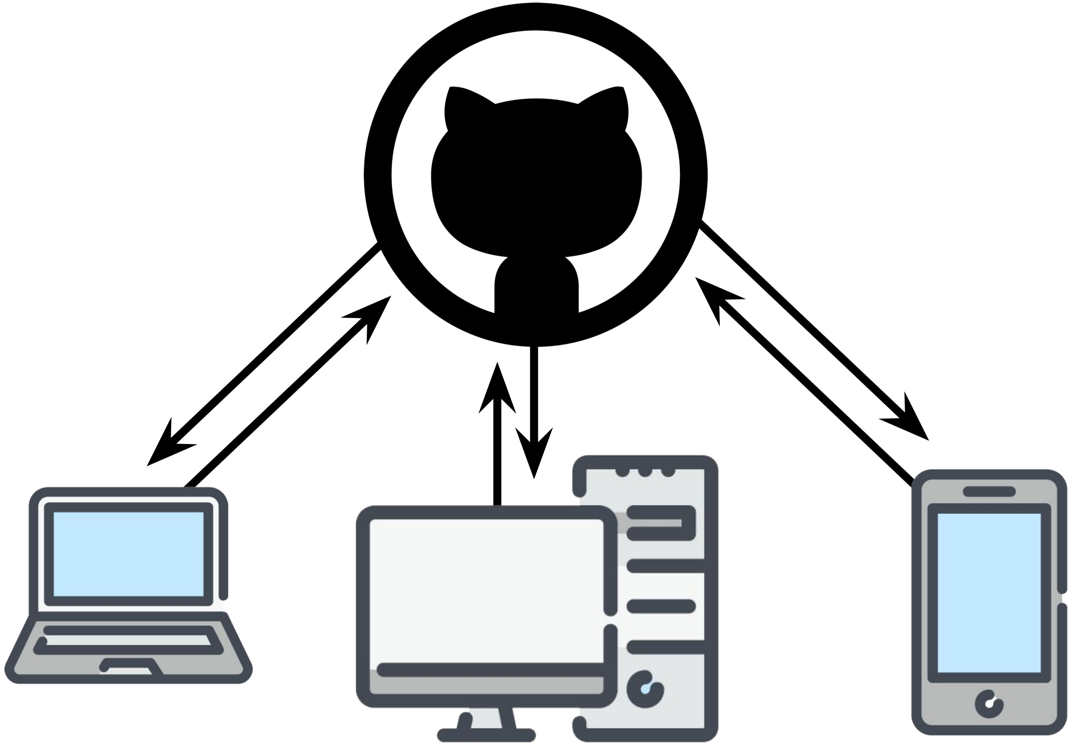
A **client** is a piece of computer hardware or software that accesses a service made available by a server. It can be a desktop computer, laptop, mobile device, and beyond! Further reading.



What Is GitHub Pages Doing?

Awesome Project Deployed to GitHub Pages

GitHub Pages is *serving* content to clients.

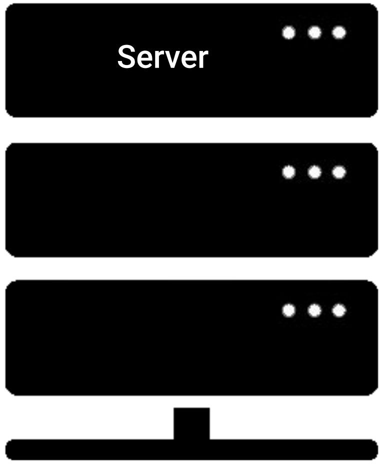


What Is a Server?

Server Definition

Depending on the context, a server is both the physical hardware *and* software that hears requests from users and returns something, such as an HTML or image file, or completes a process.

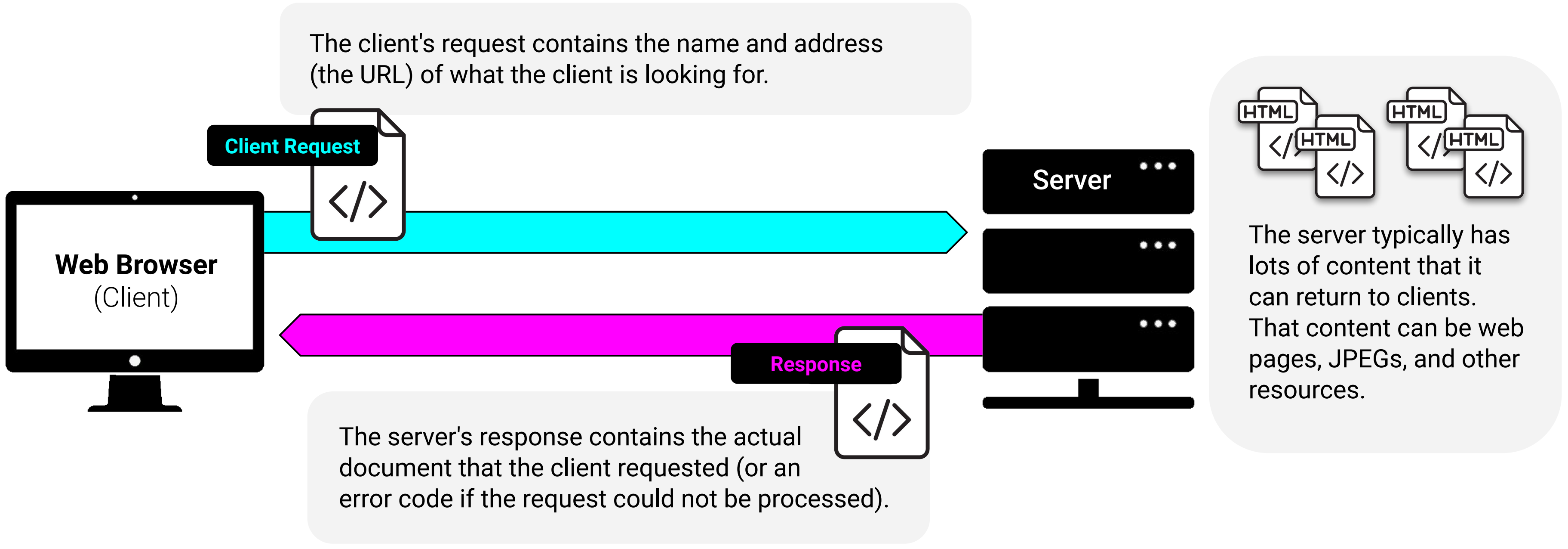
```
var http = require("http");
var PORT = 8080;

function handleRequest(request, response) {
  response.end("It Works!! Path Hit: " + request.url);
}

var server = http.createServer(handleRequest);
server.listen(PORT, function() {
  console.log("Server listening on: http://localhost:" + PORT);
});
```



What Is the Client-Server Model?

The Client-Server Model

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 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 return to clients. That content can be web pages, JPEGs, and other resources.

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 the browser.

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.**

Student Activity: Hello Node

01 In this activity, you will write and run your first Node.js application.

02 Create a file, `index.js`, in your working directory. Write JavaScript to log the string, "Hellooo, Node!".

03 Run the program using Node from the command line.

04 Confirm that it logged the text as you would expect.

Suggested Time: 10 minutes

Student Activity: Parameter Check Activity

01 In this activity, you will write a Node.js command line application that accepts two arguments and returns true if the two values are equal and false if they are not.

02 Create a file, `index.js`, in your working directory. * Write a script using `process.argv` to accept two command line arguments and compare their values.

03 * Start by simply logging the value of each argument to console.

04 *There's more than one way to solve this problem!

Suggested Time: 10 minutes

Student Activity: Appending Files

01 In this activity, you will create a "commit logger" that records commit messages (for poetic purposes) using `fs`.

02 In your working directory, create a Node application, `index.js`, that accepts a command line argument, and, each time it is run, will write the argument to file _without_ overwriting the existing text.

03 You may need to consult the Node.js `fs` documentation for the appropriate method and its usage.

04 *If `fs.readFile` reads a file and `fs.writeFile` writes, but _overwrites_, a file, what method do you think will allow you to _append_ text to a file?

Suggested Time: 10 minutes

Student Activity: Modularization

01 The JavaScript `Math` library contains many useful properties and methods.

02 But it's missing a few of the basics. In this activity, you will create your own `maths` module then import it into a Node application to access its properties and methods.

03 Create two files, `index.js` and `maths.js`, then import `maths.js` into `index.js`. Write four methods for the results of the four basic mathematical operations in `maths.js`: `sum`, `difference`, `product`, and `quotient`; and then export them.

04 In `index.js`, create variables to capture the values passed from the command line in `process.argv`: `operation`, `numOne` and `numTwo`. * Next, write a `switch` statement that accepts an `operation` parameter.

*Within each `case` of the `switch`, use the corresponding `maths` methods to perform the operation on the numbers taken from the command line using `process.argv`.

Suggested Time: 10 minutes

Student Activity: package.json

01 In this activity, you will install the `inquirer` package using `npm`.

02 Run `npm install`

03 What happens? How did `npm` know you wanted to install `inquirer`? Where did those additional packages come from?

04 How would we create our own `package.json` files?

Suggested Time: 10 minutes

Student Activity: npm init

01 In this activity, you will initialize your first Node project using `npm` and install (and save) the `inquirer` package.

02 From the command line, run `npm init`.

03 Follow the prompts and enter relevant information.

04 Once your `package.json` file is created, run `npm install inquirer --save` to install our dependency.

Suggested Time: 10 minutes

Student Activity: Inquirer User

01 In this activity, you will build a simple command line application that accepts user input and writes it to a `.json` file.

02 Initialize your repository and install the `inquirer` dependency.

03 Prompt your user with the following questions: * "What is your name?" * "What languages do you know?" * What is your preferred method of communication?"

04 Write the user response to a file.

Suggested Time: 10 minutes