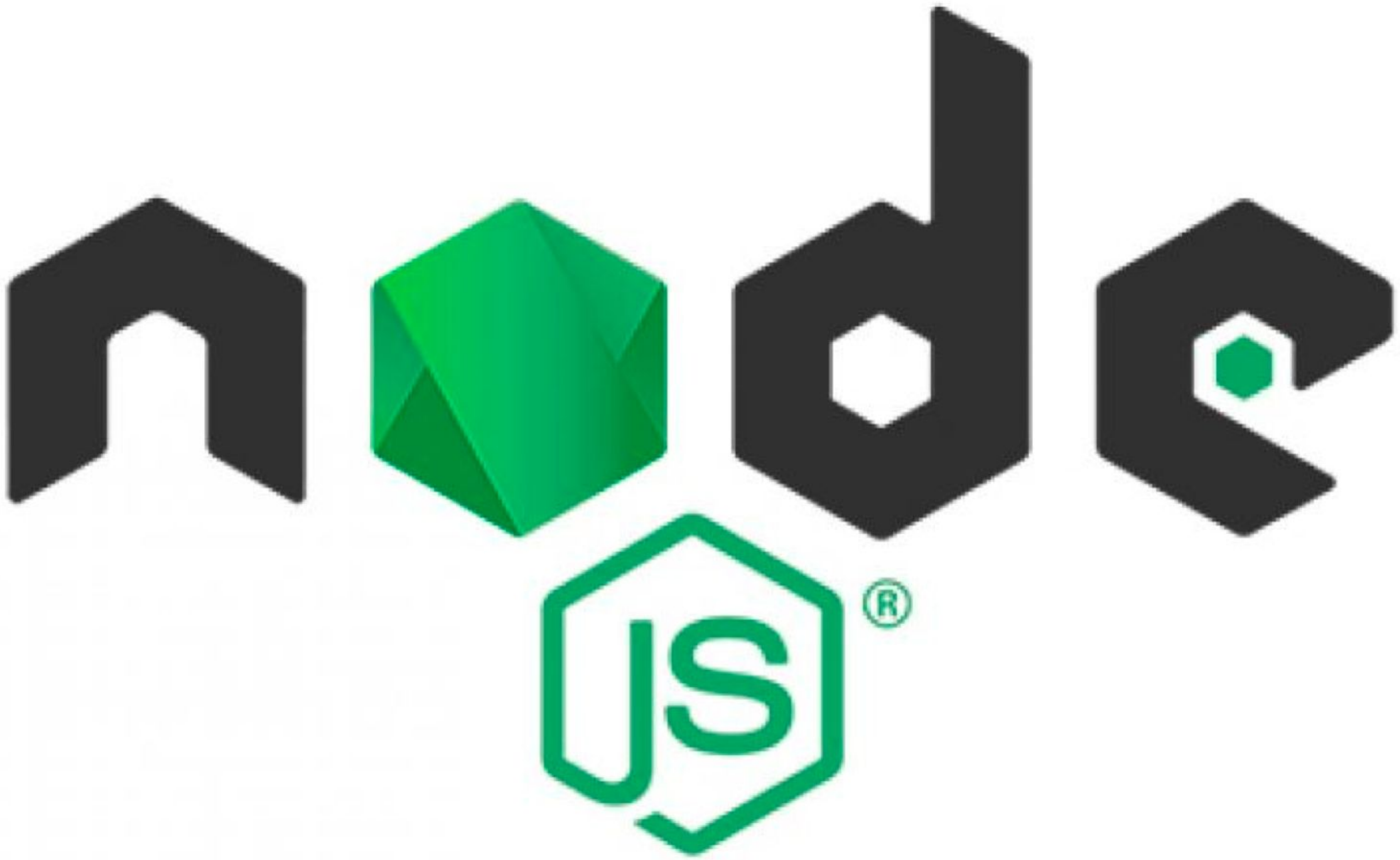
FULLSTACK
ACADEMY of CODE

Node & Node Modules

Guess who’s back(end)?

COHORT: 2011-GHP-RM-WEB-FT

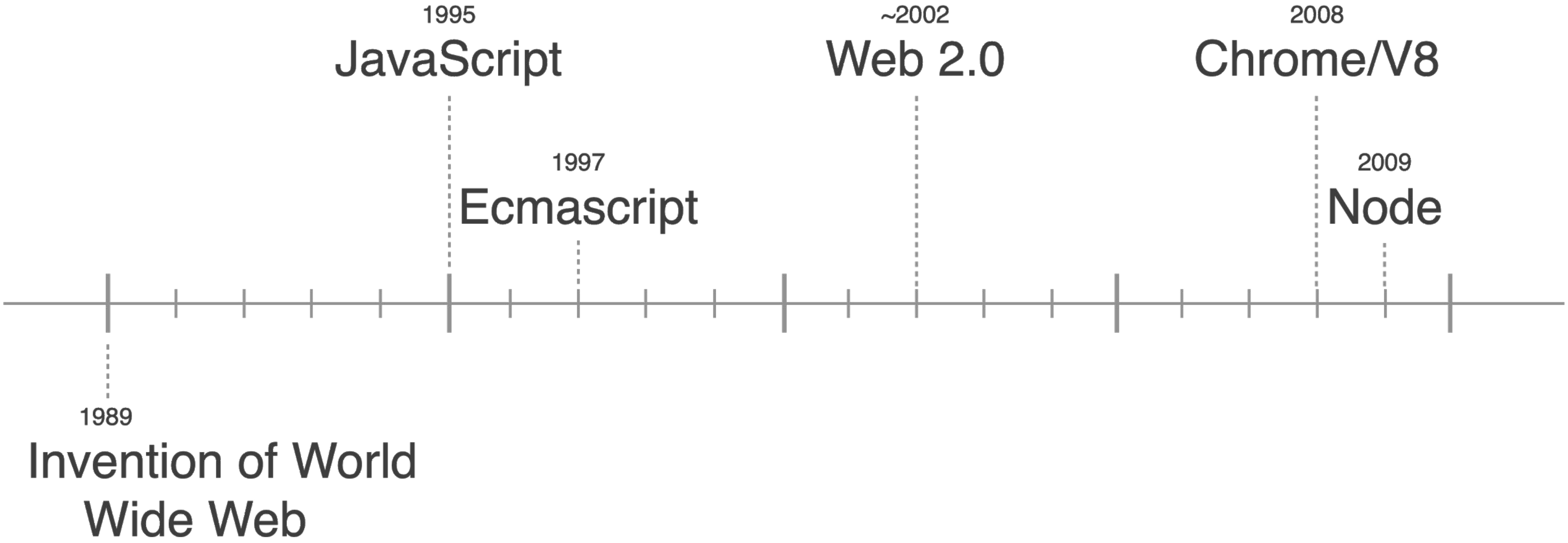
DATE: 11-13-2020


node
JS
®

But first, how do we run any programming language?

LOLCODE

TIMELINE



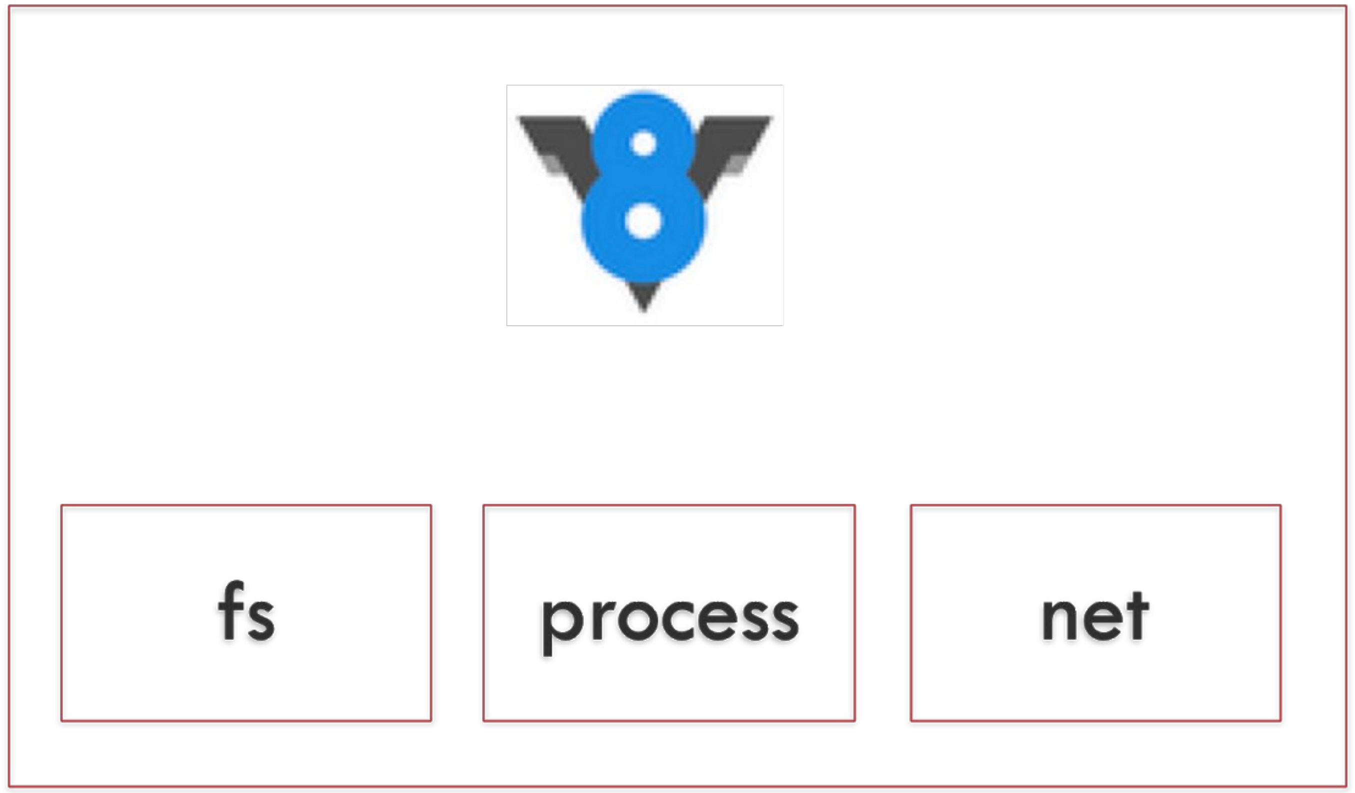
1995
JavaScript
1997
Ecmascript
~2002
Web 2.0
2008
Chrome/V8
2009
Node
1989
Invention of World Wide Web

V8 Engine

- **Runtime Environment:** Everything we need to execute a program
- V8 Engine provides the **Runtime Environment** in which JavaScript executes
- Because of V8, JavaScript can be run anywhere other than the browser

Different Environment; Different Strokes

files (e.g. app.js)

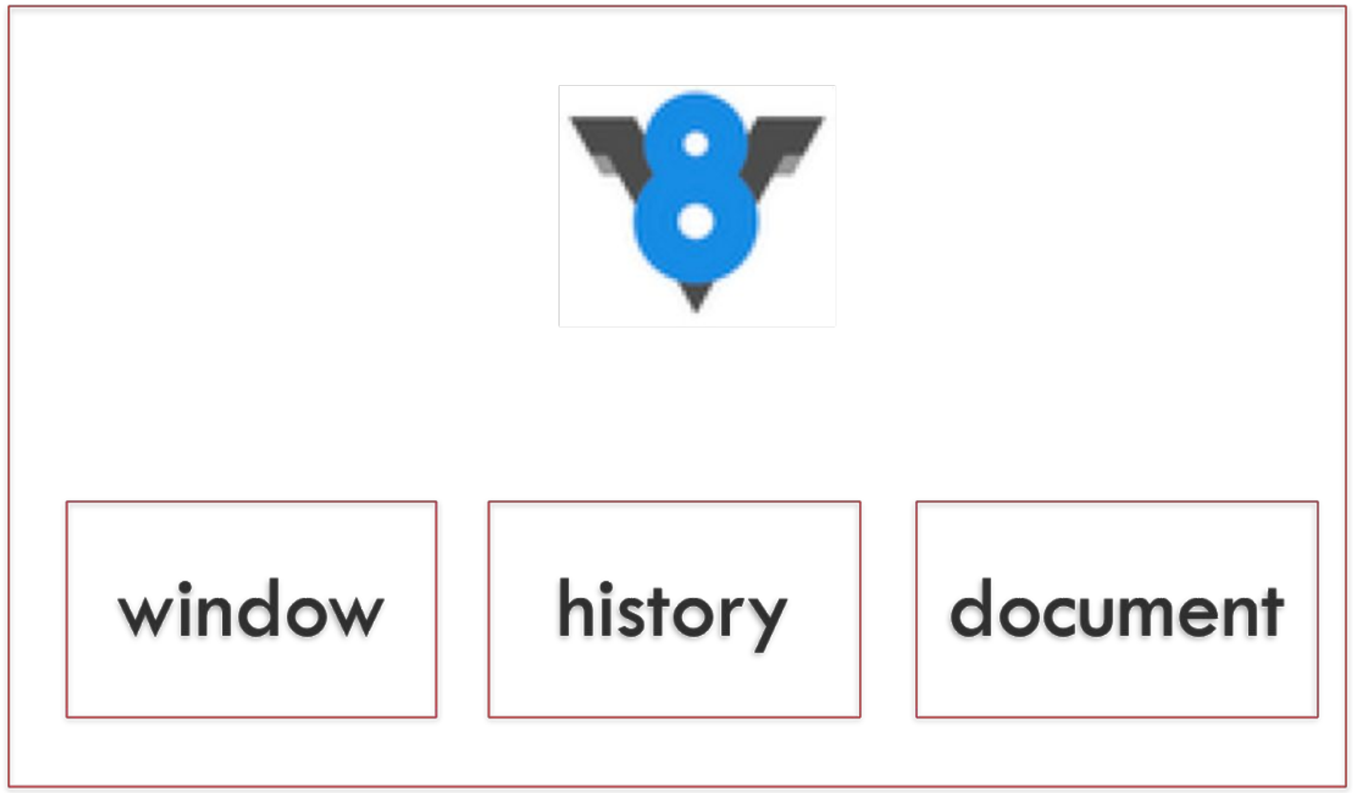


fs

process

net

`<script></script>`



window

history

document

Why do we care about JavaScript running somewhere other than a browser?

If we want to create a **server** and know JavaScript

Why do we care about creating a **server**?

If we want to create a custom, more complex website or webapp

Server

- An executing program that is running on a computer connected to the internet
- "Serves" content requested by remote clients
- Analogy: Waiter at a restaurant
  - You (the **client**) request food from the waiter. The waiter processes it by giving it to the chef. Once it's done, the waiter brings it back to you (the **client**)

Client

- A client is anything that is issuing a request
- A client can be:
  - A desktop app
  - Browser
  - Mobile app
  - IoT (Ex: Smart Refrigerator)

If programming were cooking...

Program vs. Process

- **Program is data**
  - Machine code (pre-compiled)
  - Bytecode (re-compiled by a VM)
  - Text File (can be interpreted)
- **Inert**
  - Does not do anything
- **Ready to be run as a process**

“Recipe”

- **Process is execution**
  - Memory allocation
  - CPU performing steps
- **“Live”**
- **Produces results**
- **Interactive**
- **Can be started/stopped**
- **Multiple processes from one program**

“Cooking”

Cooking Metaphor

	(term)	(metaphor)
`log('hi');`	program	recipe
JavaScript	programming language	recipe language
V8	engine/VM/interpreter	chef
Node	runtime environment	kitchen
Mojave	operating system	building (restaurant?)

Modules and the Node Environment

Module

- Simply put: A node **module** is a file that contains JavaScript code
  - You can consider them libraries
- This enables us to follow some Software Engineering patterns
  - Different files; different responsibilities (Single Responsibility Principle)
  - Separates our code (Separation of Concerns)
  - Structure visibility (We can determine where we are in the program more easily)
- Modules also make it easier to collaborate and test

Global Variables

Every module in Node has access to the same set of global variables

process
global
console
setTimeout/clearTimeout
setInterval/clearInterval

Module Variables

Every module in Node has its own set of “module” variables that are available in the default scope

__dirname
__filename
module
require

What we don’t have

We no longer have access to:

window
document
alert
history

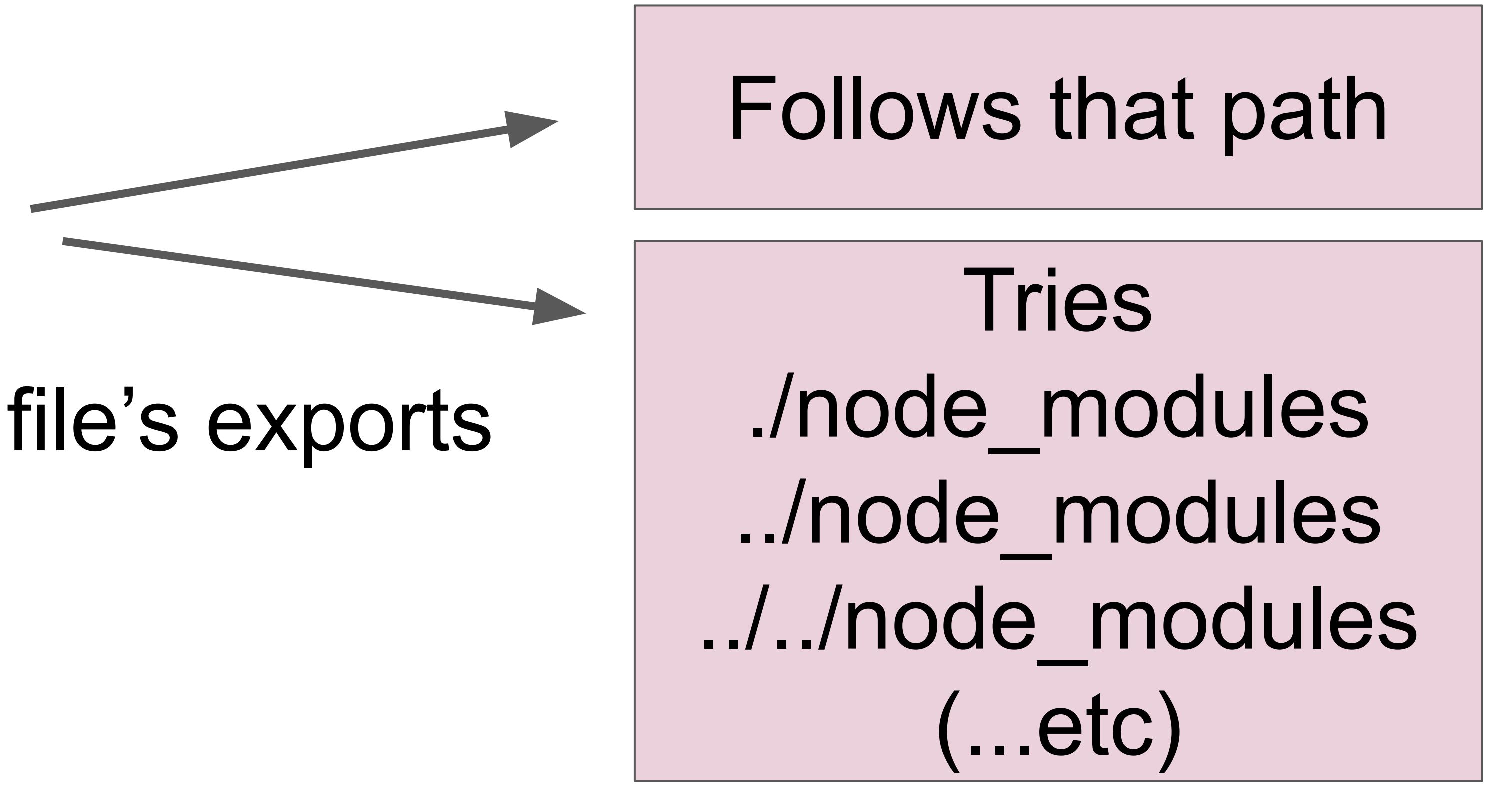
Remember: We are in a different environment now. Everything above is browser specific

Using Modules

- As mentioned, modules are useful because it separates concerns for us
- Node gives us a way to use modules across our program by way of **module.exports**
  - It is initially an empty object
  - We assign it the data we want to expose
  - We have to **require** the file (**module)** in other files to use its contents
    - A require of the file **will return its module.exports**

require

- Finds a file
- Executes it
- Imports that file’s exports



Follows that path

Tries
./node_modules
../node_modules
../../node_modules
(...etc)

Notes:

- We can leave off the .js part of a file path
- A path to a directory will look for index.js by default

Sources of Modules

- Ones we define ourselves
- Ones Node gives us by default
- Third party (NPM)

NPM

- Node Package Manager
- Command line tool
- Can find libraries of code online
- Downloads them locally or globally (into node_modules) directory
- Keeps list of project dependencies in package.json

package.json

- Describes your project
  - Special scripts to run/use in the project
  - Shows project dependencies needed to run the project
- Enables collaboration within your team
- Enables sharing within the Node community

Recap of Key Concepts

- **Runtime environment**: Gives us everything we need to run a program. V8 Engine provides this for us to run JavaScript outside the browser
- **Server**: An executing program on a computer connected to the internet that processes requests
- **Client:** Something that makes a request

Recap of Key Concepts

- **Modules:** Files that contain JavaScript code usually separated into different functionalities
- **module.exports**: Exposes the pieces of code/data we may want other files to use
- **require: pulls in** what **module.exports** exposes