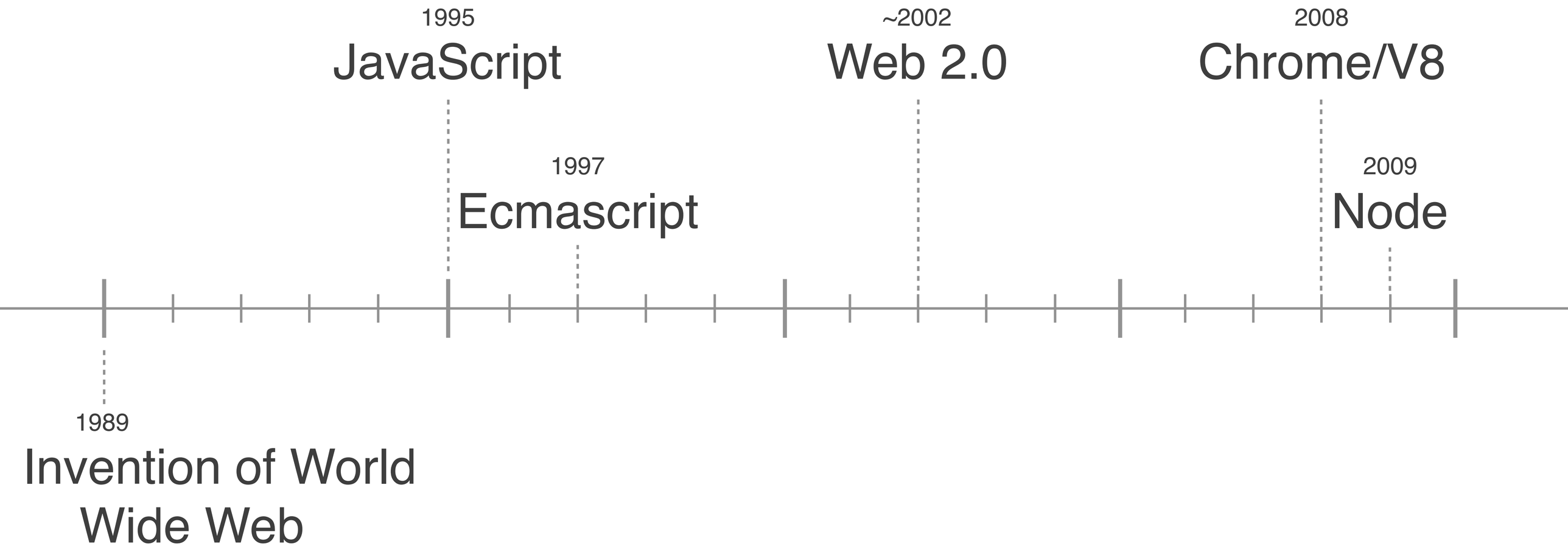
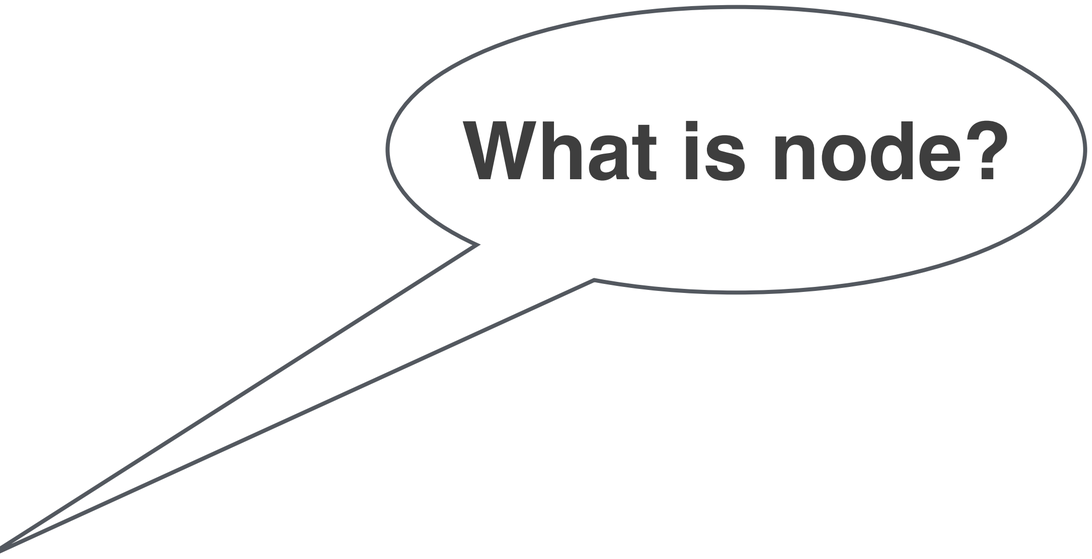
```
NODE.INTRO((err, ideas) => {
  if (err) throw new Question(err)
  else understand(ideas)
})
```

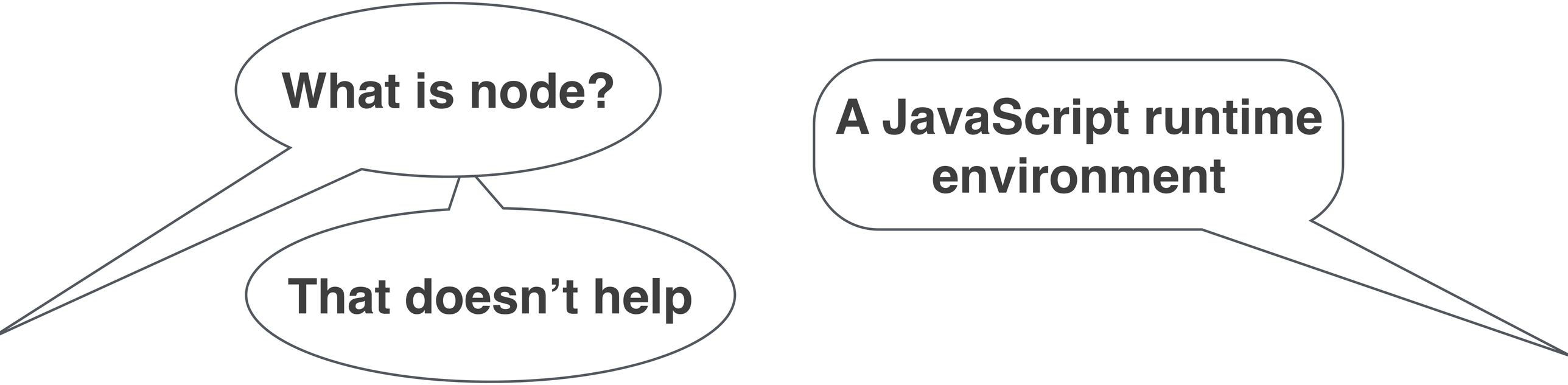
BACKGROUND

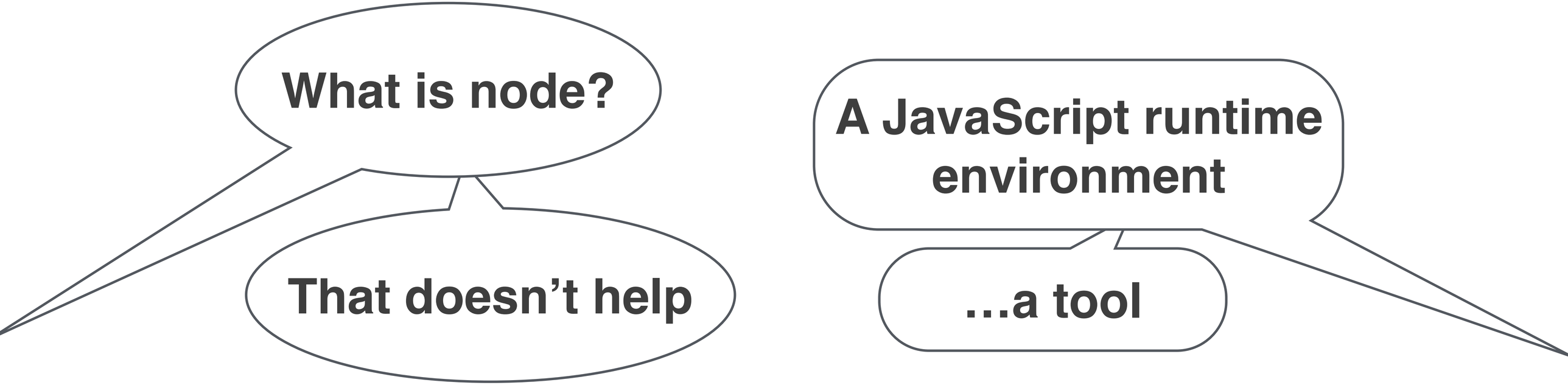
TIMELINE

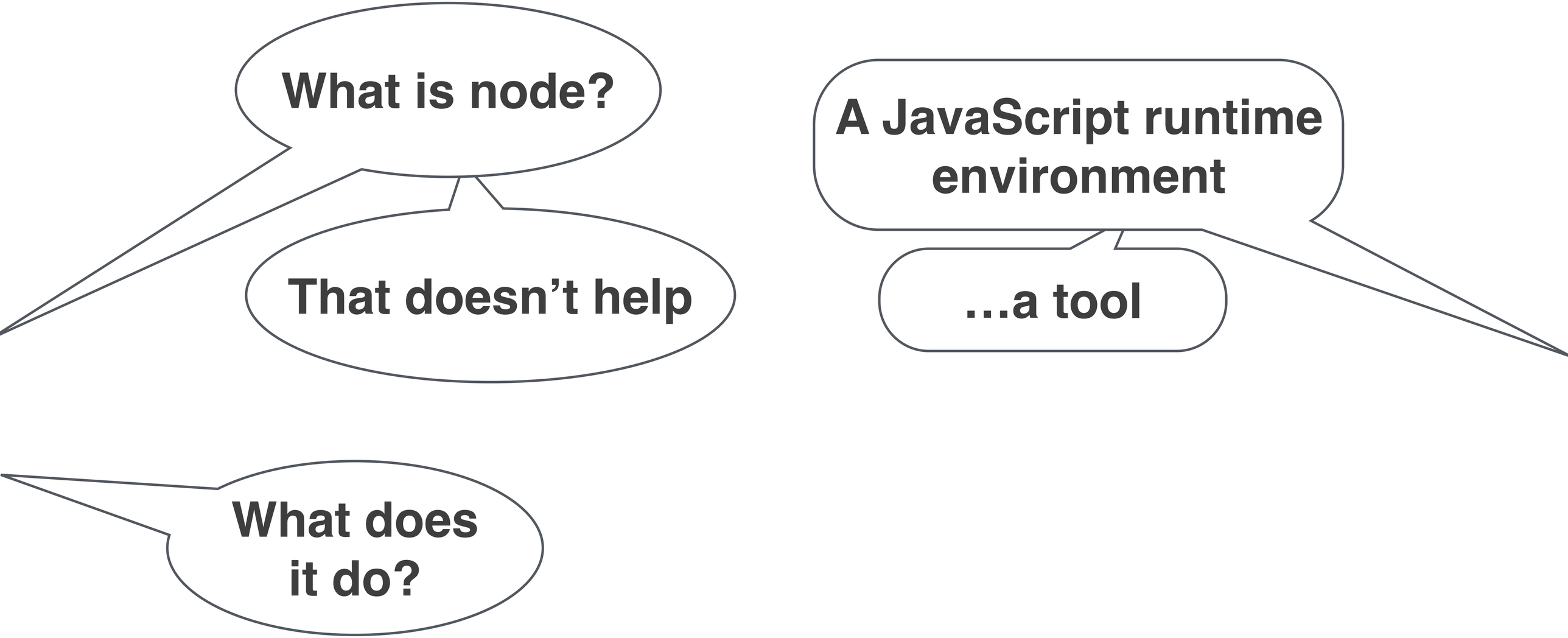

1995
JavaScript
~2002
Web 2.0
2008
Chrome/V8
1997
Ecmascript
2009
Node
1989
Invention of World Wide Web

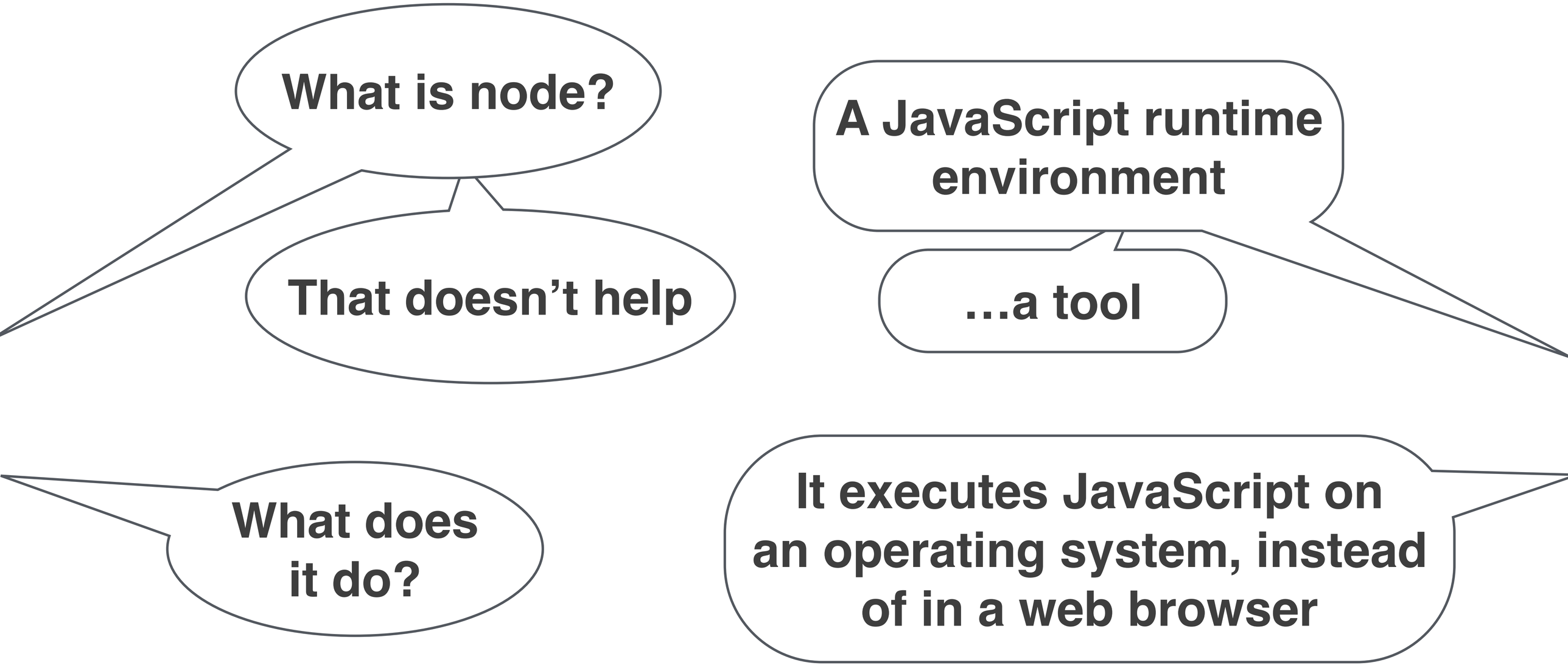

What is node?


What is node?
A JavaScript runtime environment


What is node?
That doesn’t help
A JavaScript runtime environment


What is node?
That doesn’t help
A JavaScript runtime environment
…a tool


What is node?
That doesn’t help
What does it do?
A JavaScript runtime environment
…a tool


What is node?
That doesn't help
A JavaScript runtime environment
…a tool
What does it do?
It executes JavaScript on an operating system, instead of in a web browser

files (e.g. app.js)

fs | process | net

`<script></script>`s

window | history | document

WHY CARE?

WHY CARE?

If you want to create a server and know JavaScript

WHY CREATE A SERVER?

WHY CREATE A SERVER?

If you want to create a custom website or webapp

SERVER

- A program running on a computer connected to the internet
- Serves content requested by remote clients

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 — not doing anything**
- **Ready to be run as a process**

- **Process is execution**
  - memory allocated
  - CPU performing steps
- **"Live"**
- **Produces results**
- **Interactive**
- **Can be started/stopped**
- **Multiple processes from one program…**

Program vs. Process

"recipe"

- **Program is data**
  - machine code (pre-compiled)
  - bytecode (re-compiled by a VM)
  - text file (can be interpreted)
- **Inert — not doing anything**
- **Ready to be run as a process**

- **Process is execution**
  - memory allocated
  - CPU performing steps
- **"Live"**
- **Produces results**
- **Interactive**
- **Can be started/stopped**
- **Multiple processes from one program…**

Program vs. Process

"recipe"

- **Program is data**
  - machine code (pre-compiled)
  - bytecode (re-compiled by a VM)
  - text file (can be interpreted)
- **Inert — not doing anything**
- **Ready to be run as a process**

"cooking"

- **Process is execution**
  - memory allocated
  - CPU performing steps
- **"Live"**
- **Produces results**
- **Interactive**
- **Can be started/stopped**
- **Multiple processes from one program…**

COOKING METAPHOR

	(term)	(metaphor)
`log('hi');`		
JavaScript		
V8		
Node		
Sierra		

COOKING METAPHOR

	(term)	(metaphor)
`log('hi');`	program	
JavaScript		
V8		
Node		
Sierra		

COOKING METAPHOR

	(term)	(metaphor)
`log('hi');`	program	
JavaScript	programming language	
V8		
Node		
Sierra		

COOKING METAPHOR

	(term)	(metaphor)
`log('hi');`	program	
JavaScript	programming language	
V8	engine/VM/interpreter	
Node		
Sierra		

COOKING METAPHOR

	(term)	(metaphor)
`log('hi');`	program	
JavaScript	programming language	
V8	engine/VM/interpreter	
Node	runtime environment	
Sierra		

COOKING METAPHOR

	(term)	(metaphor)
`log('hi');`	program	
JavaScript	programming language	
V8	engine/VM/interpreter	
Node	runtime environment	
Sierra	operating system	

COOKING METAPHOR

	(term)	(metaphor)
`log('hi');`	program	recipe
JavaScript	programming language	
V8	engine/VM/interpreter	
Node	runtime environment	
Sierra	operating system	

COOKING METAPHOR

	(term)	(metaphor)
`log('hi');`	program	recipe
JavaScript	programming language	recipe language
V8	engine/VM/interpreter	
Node	runtime environment	
Sierra	operating system	

COOKING METAPHOR

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

COOKING METAPHOR

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

COOKING METAPHOR

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

MODULES AND THE NODE ENVIRONMENT

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
```

`module`

- **Object**
- **Represents the module itself**
- **Most importantly, has a property called `exports`**

`module.exports`

- **Initially an empty object**
- **Assign it the data you want to expose**
- **A `require` of this file (“module”) will return its `module.exports`**

require

`require`

- Finds a file

`require`

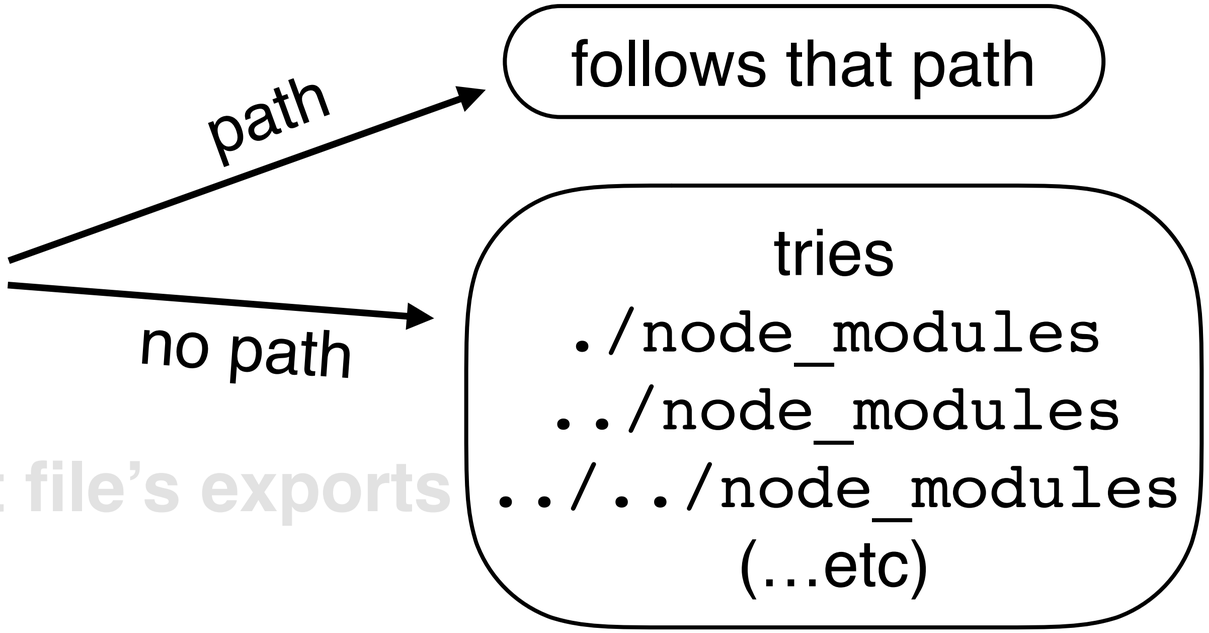
- Finds a file
- Executes it

`require`

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

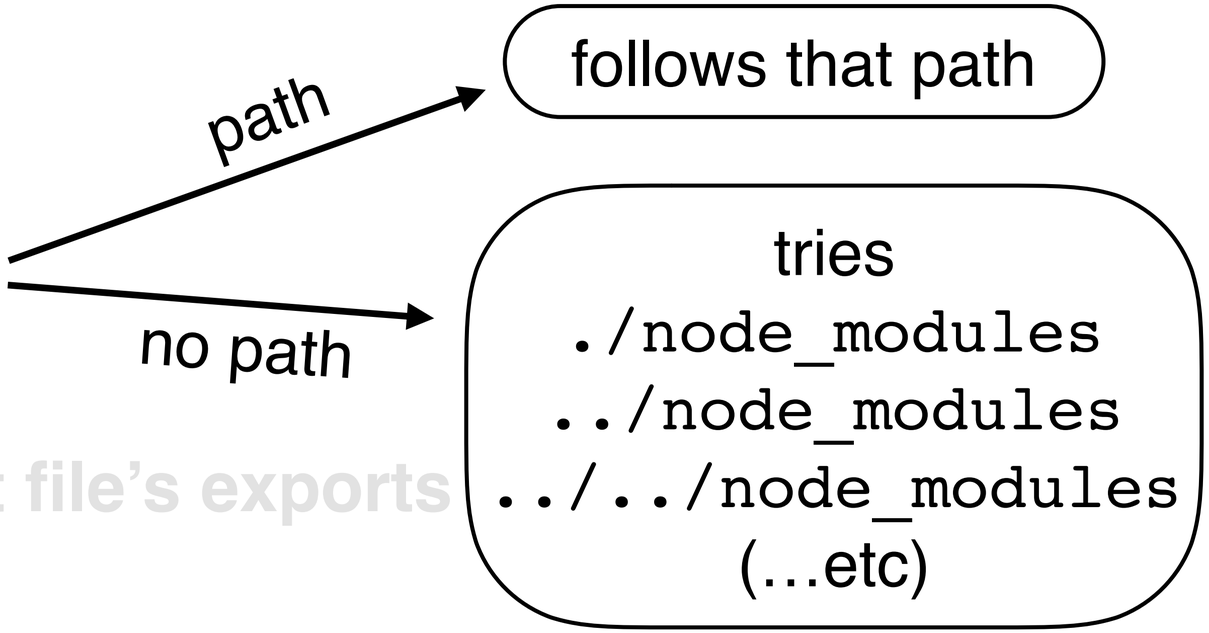
`require`

- **Finds a file**
- **Executes it**
- **Imports that file's exports**



`require`

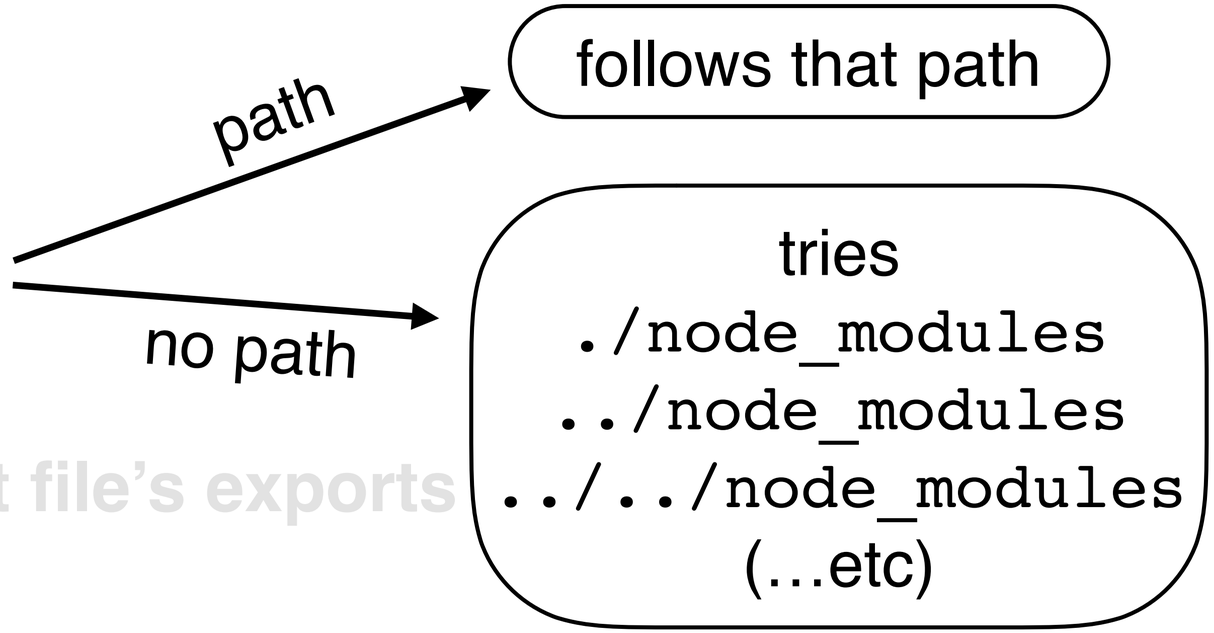
- **Finds a file**
- Executes it
- Imports that file's exports



- You can leave off the .js in a file path

`require`

- **Finds a file**
- Executes it
- Imports that file's exports



- You can leave off the .js in a file path
- A path to a directory will look for index.js

DEMO

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, e.g. its dependencies…

`package.json`

Describes your project, e.g. its dependencies…

- **Collaboration within your team**

`package.json`

Describes your project, e.g. its dependencies…

- **Collaboration within your team**
- **Sharing within the node community**

SUMMARY

- **Node allows for server-side JavaScript**
- `require` **pulls in what** `module.exports` **exposes**