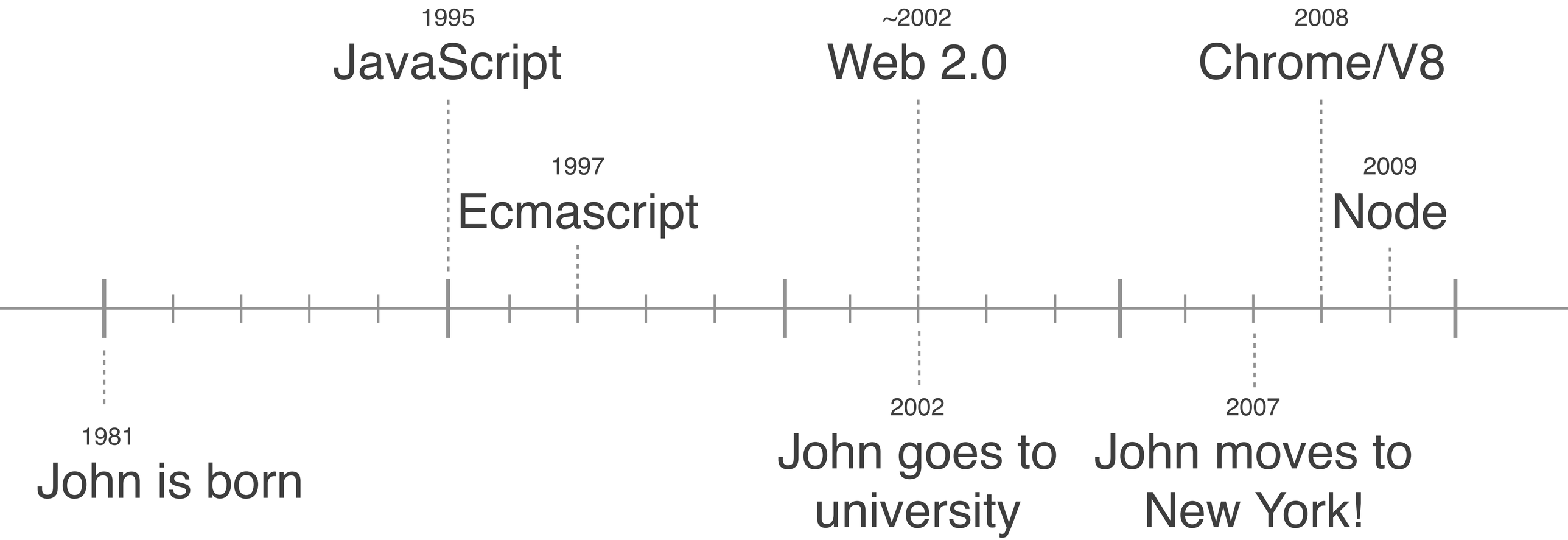
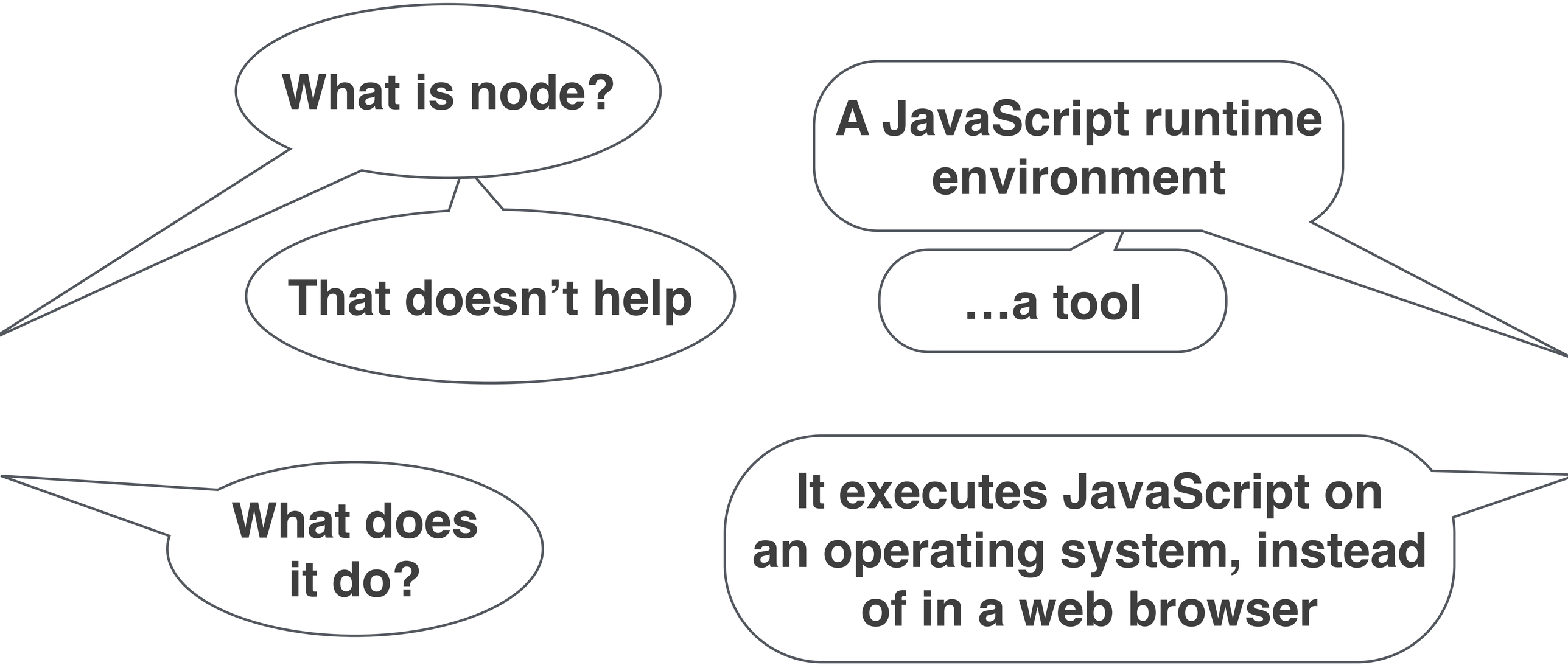
NODE.INTRO(

```
function (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
1981
John is born
2002
John goes to university
2007
John moves to New York!


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?

If you want to create a server and know JavaScript

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

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

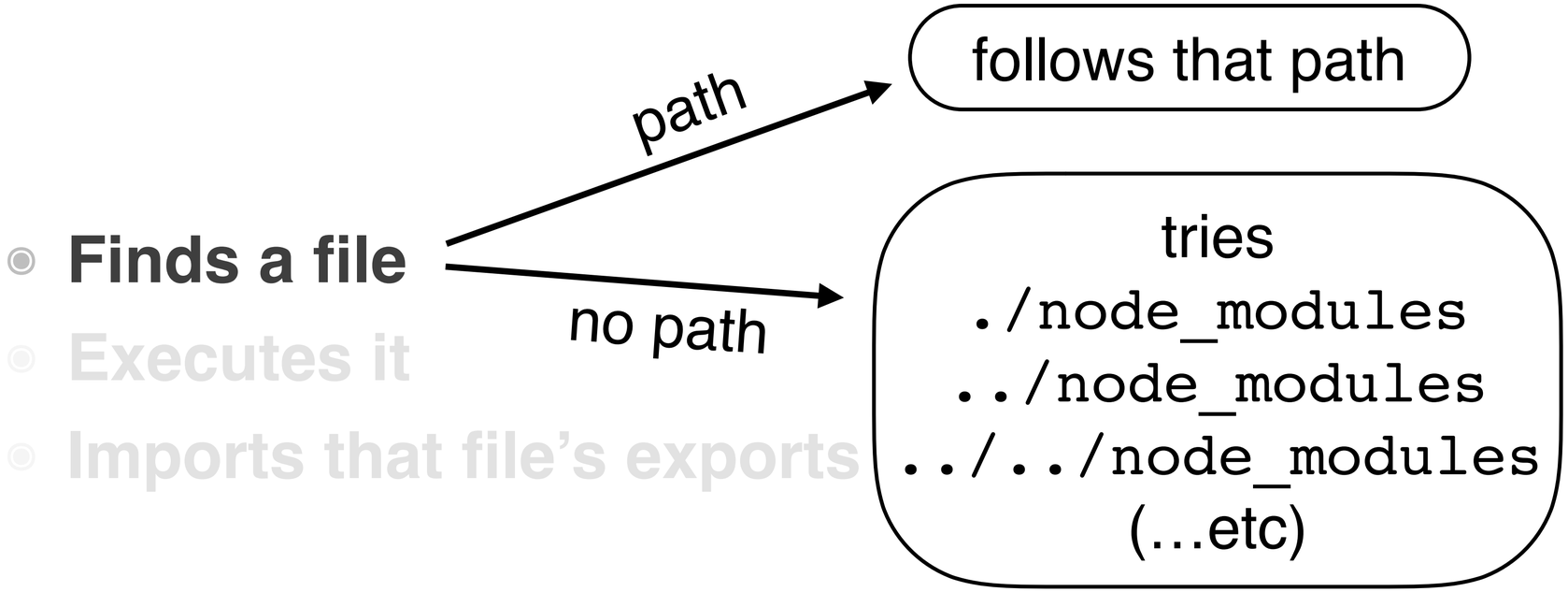
MODULES

`module.exports`

- **Assign it the data you want to expose**
- **A `require` of this file will return its `module.exports`**

`require`

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



NPM

- node package manager
- Command line tool
- Can find libraries of code online
- Downloads them locally (into `node_modules` directory)
- Keeps list of project dependencies in `package.json`

`package.json`

Describes your project, e.g. its dependencies…

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

ASYNCHRONICITY

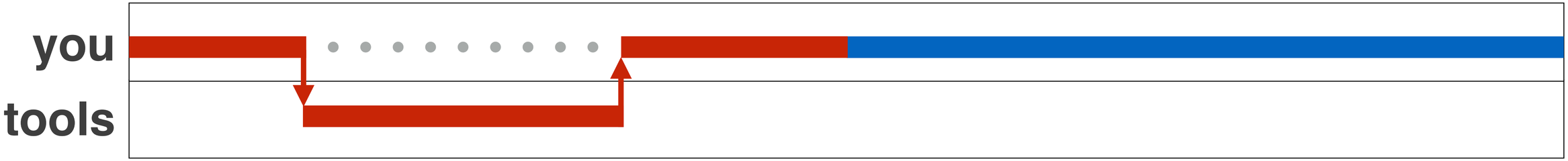
CONCURRENCY

"Let's bake a cake"

1. You only make the icing after the cake comes out of the oven
2. You make the icing while the cake is in the oven
3. I only make the icing and you only make the cake

CONCURRENCY

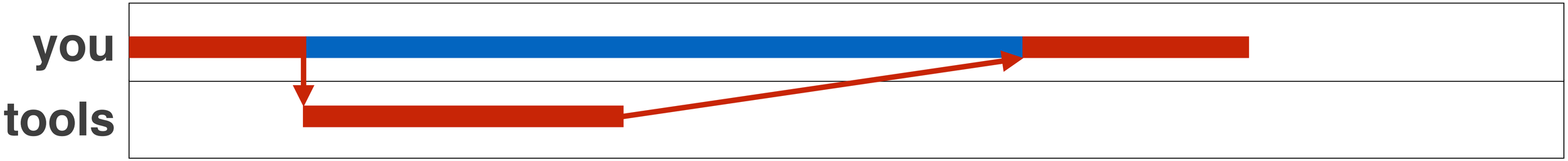
Blocking…



1. You only make the icing after the cake comes out of the oven

CONCURRENCY

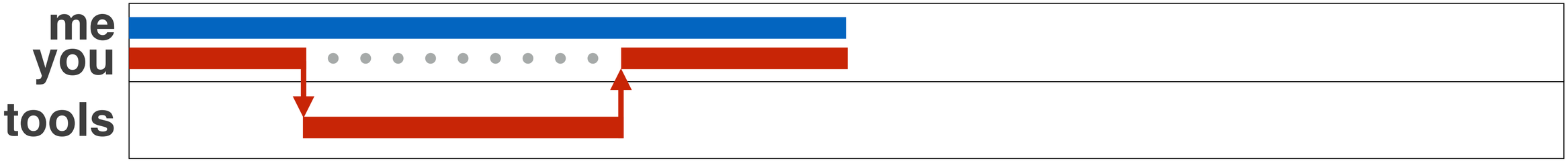
Non-blocking...



2. You make the icing while the cake is in the oven

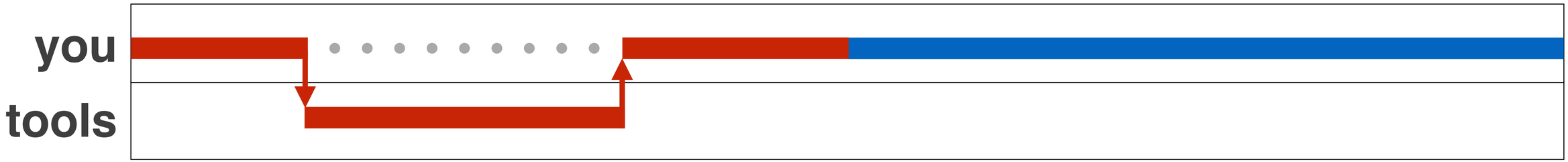
CONCURRENCY

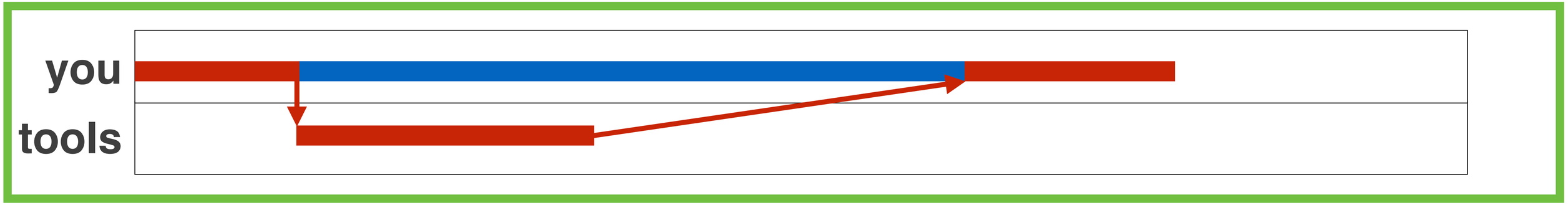
Parallel…



3. I only make the icing and you only make the cake

WHICH DESCRIBES JAVASCRIPT?





Er, not exactly

"Node.js is a ~~single-threaded~~, event-driven, non-blocking I/O platform"

– SOME PEOPLE ON THE INTERNET

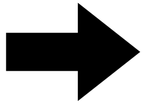
"JavaScript is single-threaded" …arguably yes

– OTHER PEOPLE ON THE INTERNET

ASYNC

(Code is asynchronous if) the execution order is not dependent upon the command order

WHAT HAPPENS?



```
console.log(‘Some callbacks');
setTimeout(function(){
  console.log('you');
}, 3000);
console.log('love');
```

```
Some callbacks
love
you
```

EVENT BASED

A function that executes asynchronously…

1. Kicks off some external process
2. Registers an event handler for when that process finishes (callback)

WHAT HAPPENS?

```
var start = new Date;
setTimeout(function(){
  var end = new Date;
  console.log('Time elapsed:', end - start, 'ms');
}, 500);

while (new Date - start < 1000) {};
```

```
=> Time elapsed: 1000 ms
```

WHY?

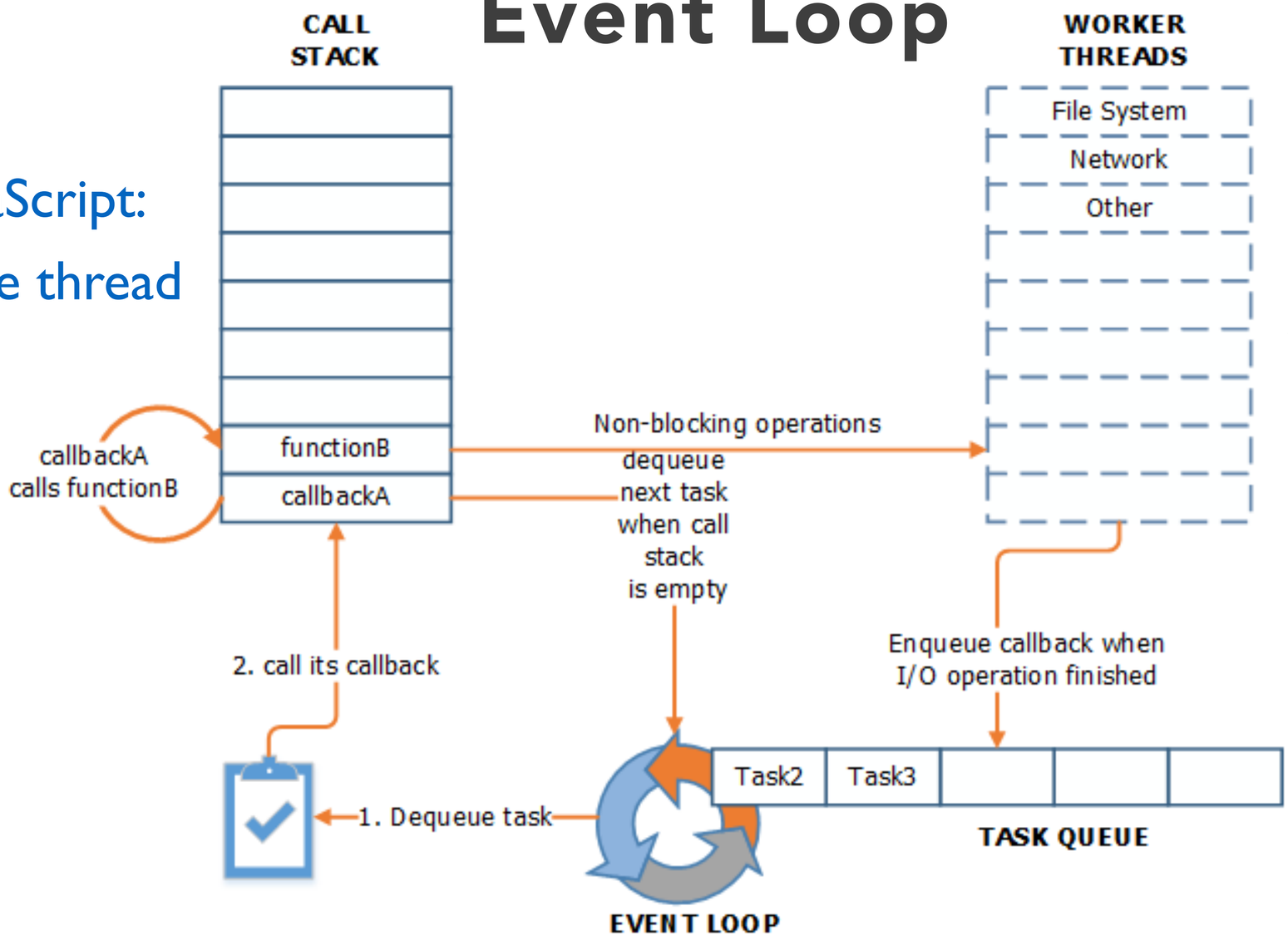
```
var start = new Date;
setTimeout(function(){ // starts up a timeout only
  var end = new Date;
  console.log('Time elapsed:', end - start, 'ms');
}, 500);

while (new Date - start < 1000) {}; // idles for 1000 ms
// meanwhile, halfway through, the timer finishes
// but while loops are blocking
// and js does not interrupt blocking commands
// after the while it has no other commands
// so it will execute the queued callback
```

Event Loop

JavaScript:

One thread



Thread pool (libeio):

Slow stuff, multiple threads

Event loop (libev):

One thread

How do I know if a function is asynchronous?

That doesn't help

If you want to be sure, you have to look it up

…Wait really?

Well, async operations often have the following callback pattern:

```
asyncThing(function(err,data){…})
```

SUMMARY

- Node allows for server-side JavaScript
- `require` pulls in what `module.exports` puts out
- JavaScript is single-threaded but its runtime environment is not
- A callback executes when its async event finishes