

ASYNCHRONICITY

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
(3000ms elapse)
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



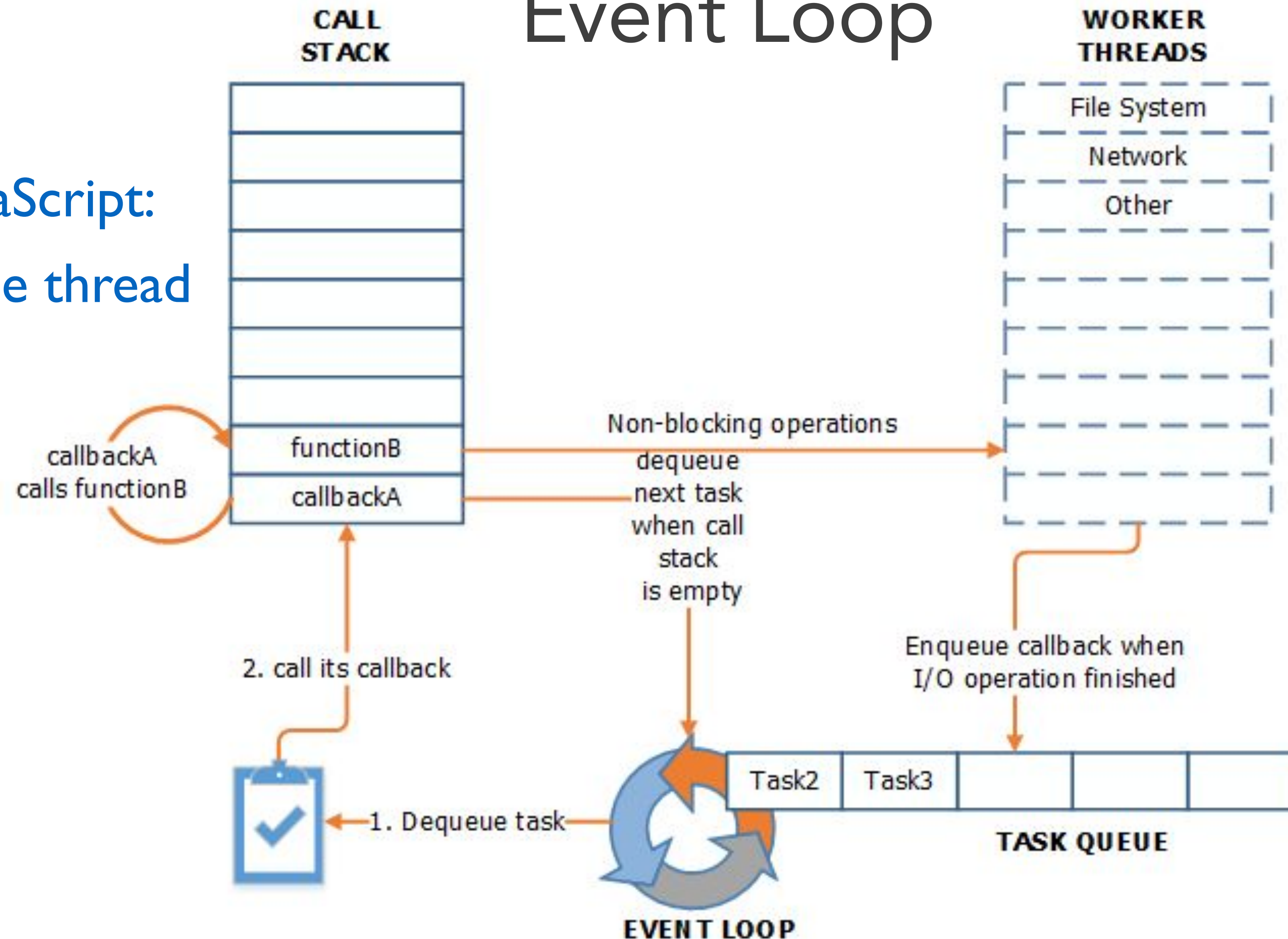
WHY?

```
var start = new Date;
setTimeout(function() { // starts a timer and holds onto
    var end = new Date; // the callback
    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 (libuv):
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

- ◉ **JavaScript is single-threaded but its runtime environment is not**
- ◉ **A callback executes when its async event finishes**
- ◉ **Anything you wish to do *after* the async event completes *must* happen in the callback**