

ASYNCHRONICITY

Do a skit



CONCURRENCY

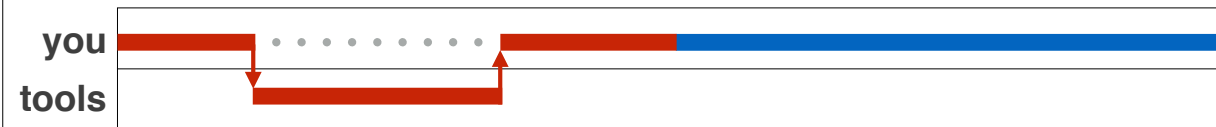
“Let’s bake a cake”

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...

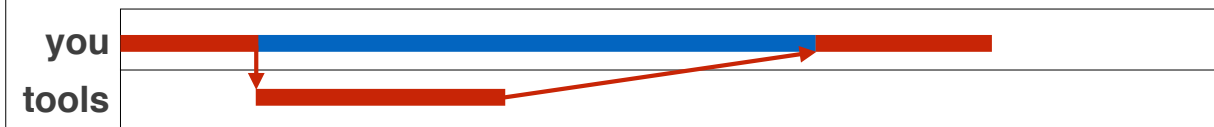


1. You only make the icing after the cake comes out of the oven



CONCURRENCY

Non-blocking...



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



CONCURRENCY

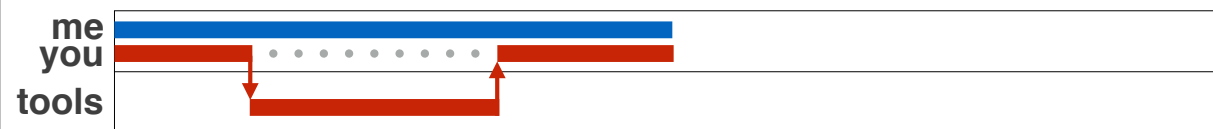
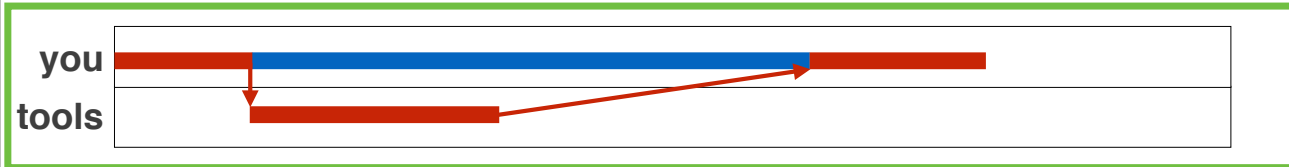
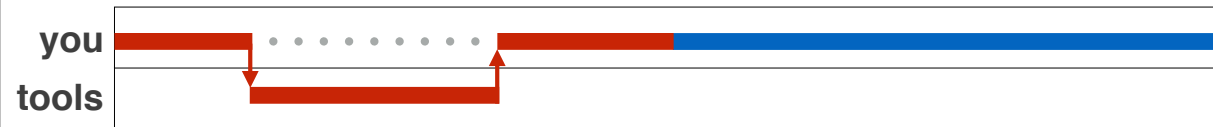
Parallel...



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



WHICH DESCRIBES JAVASCRIPT?




We say the middle one, but it's sort of not true

Er, not exactly

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

– SOME PEOPLE ON THE INTERNET

 FULLSTACK

7

INTRO TO NODE

Show demo in activity monitor — node is using multiple threads

“JavaScript is single-threaded” ...arguably yes

– OTHER PEOPLE ON THE INTERNET

JavaScript is a language, so it doesn't really have a thread.
Your JavaScript's execution is single-threaded.

Show examples of how APIs "leave" JS thread and have to "come back" at some point



ASYNC

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

Q: What does it mean for code to be asynchronous?



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

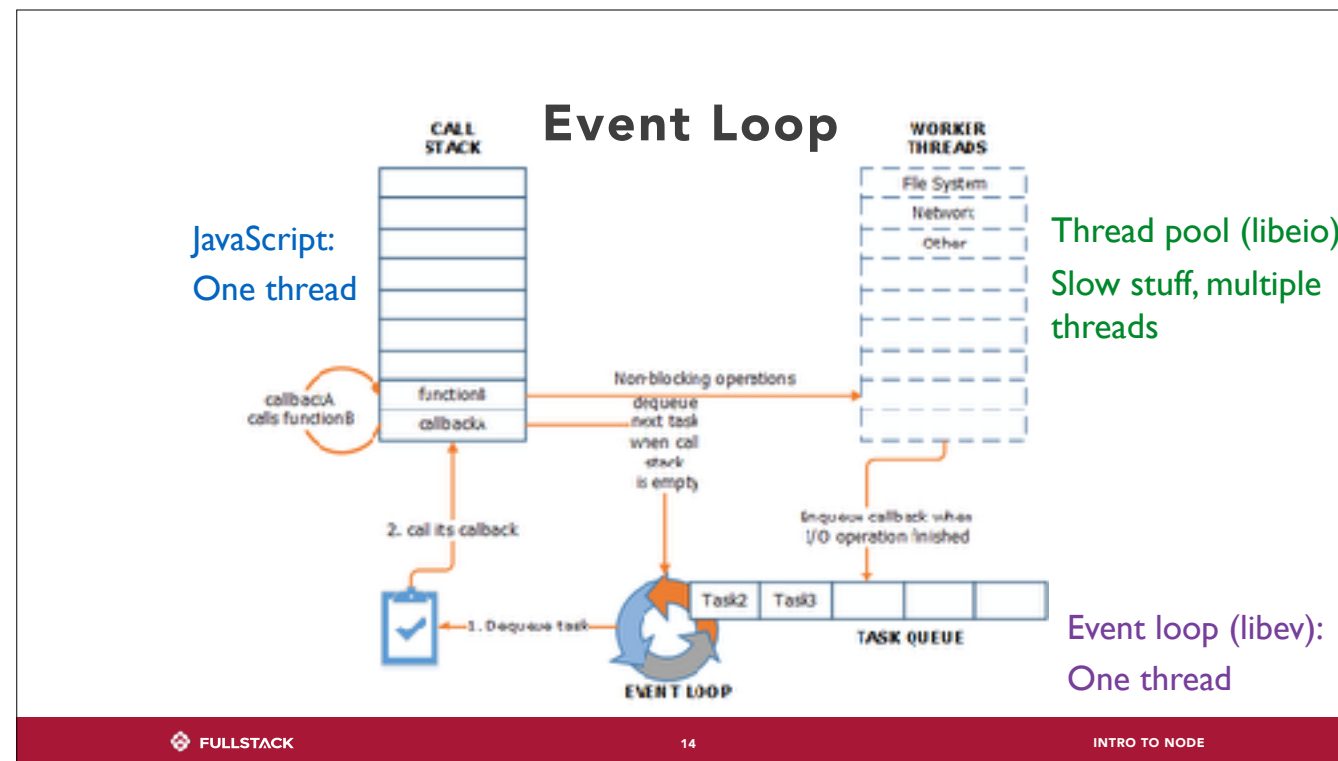
Pop quiz!



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
```



Recommended: watch event loop video. Demo on whiteboard with some simple code.

How do I know if a function is asynchronous?

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

That doesn't help

...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**

Demo:
Readfile of some large files