

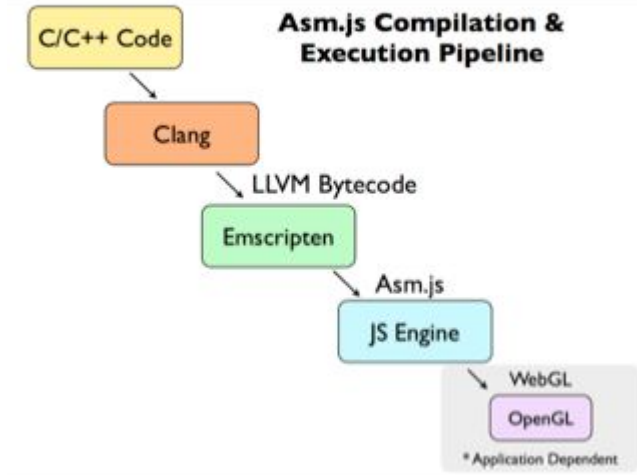
Introduction to Node.js

 Red Babel.

Atwood's Law

“any application that can be written in JavaScript, will eventually be written in JavaScript.”

-- Jeff Atwood, Stack Overflow Co-founder. Source: <http://blog.codinghorror.com/the-principle-of-least-power/>



<https://blog.mozilla.org/blog/2014/03/12/mozilla-and-epic-preview-unreal-engine-4-running-in-firefox/>

```
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered (default)
Real Time Clock Driver v1.12ac
JS clipboard: I/O at 0x03c0
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing disabled
serial8250: ttyS0 at I/O 0x3f8 (irq = 4) is a XScale
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
loop: loaded (max 8 devices)
ne.c:v1.10 9/23/94 Donald Becker (becker@scyld.com)
Last modified Nov 1, 2000 by Paul Gortmaker
NE*000 ethercard probe at 0x300: aa aa aa aa aa aa
eth0: NE2000 found at 0x300, using IRQ 9.
Uniform Multi-Platform E-IDE driver Revision: 7.00alpha2
ide: Assuming 50MHz system bus speed for PIO modes; override with idebus=xx
hda: JSLinux HARDDISK, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
hda: max request size: 128KiB
hda: 116736 sectors (59 MB) w/256KiB Cache, CHS=115/16/63
hda: unknown partition table
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
Using IPI Shortcut mode
Time: pit clocksource has been installed.
VFS: Mounted root (ext2 filesystem) readonly.
Freeing unused kernel memory: 128k freed
Booted in 6.488 s
Welcome to JS/Linux
/var/root #
```

Clear clipboard

ECMAScript 6

June 2015: <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>

- **Fat arrows function**
- **Classes**
- **Destructuring**
- **Promises**
- **Modules**
- **Generators**

What is Node.js

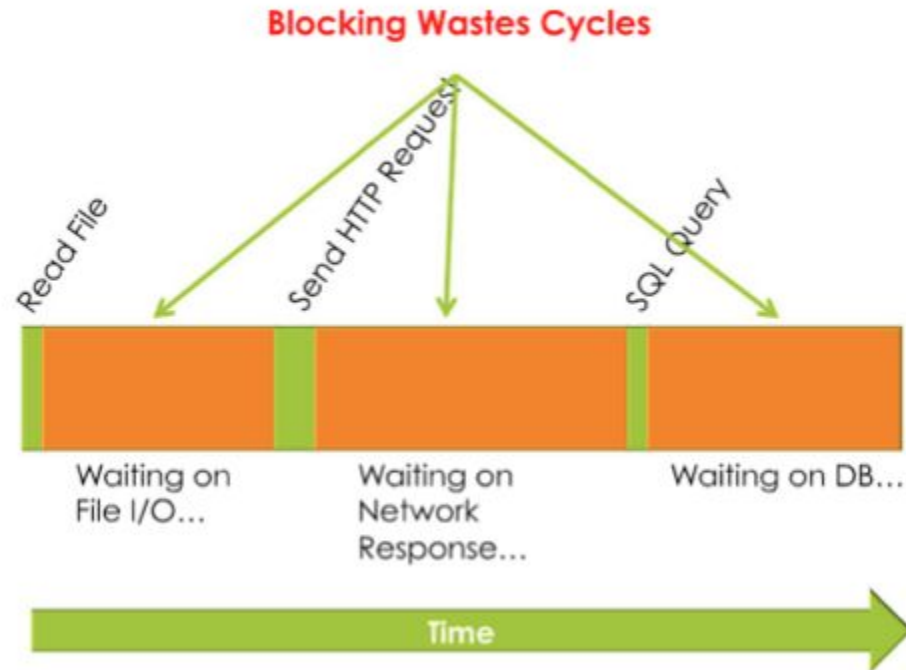
“Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Node.js uses an **event-driven, non-blocking I/O** model that makes it lightweight and efficient.”

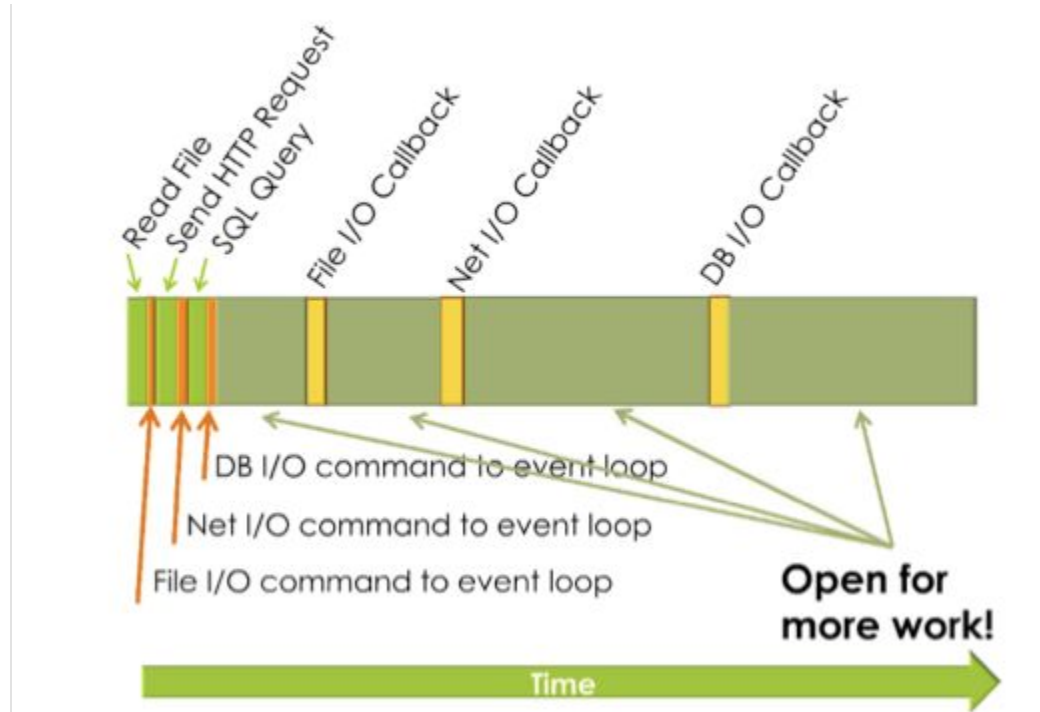
Allows you to build scalable network applications using JavaScript on the server-side.



Blocking



Non blocking



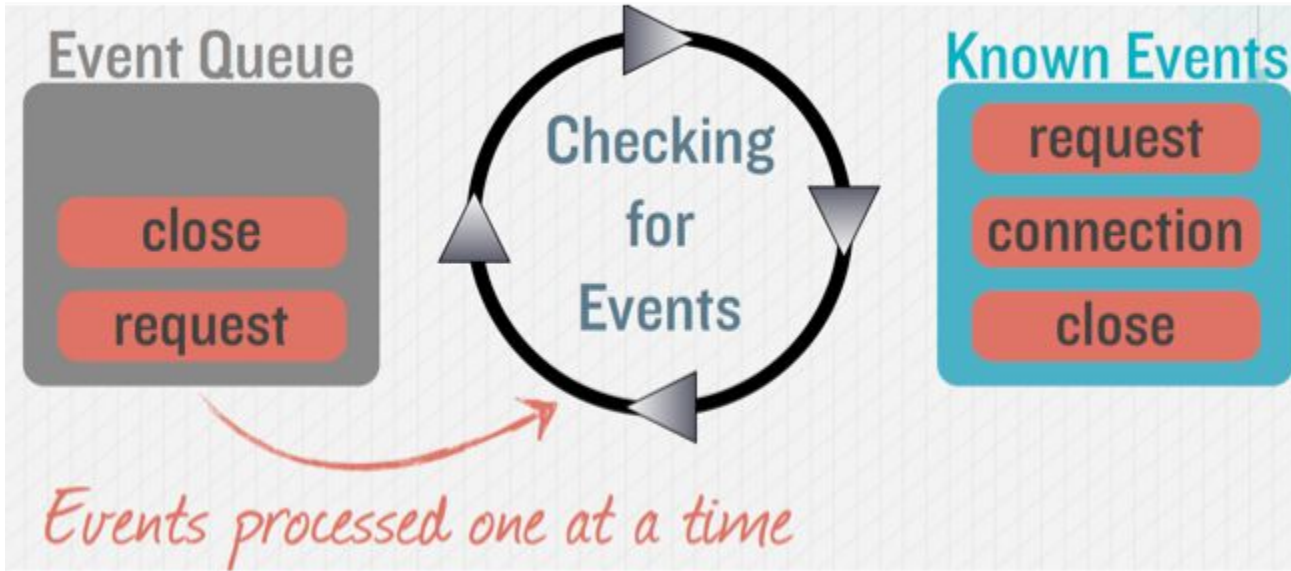
Node.js App

- Run entirely in a single thread
- Passes I/O requests to the event loop, along with callbacks

Your code then:

- Goes to sleep
- Uses no system resources
- Will be notified via callback when I/O is complete

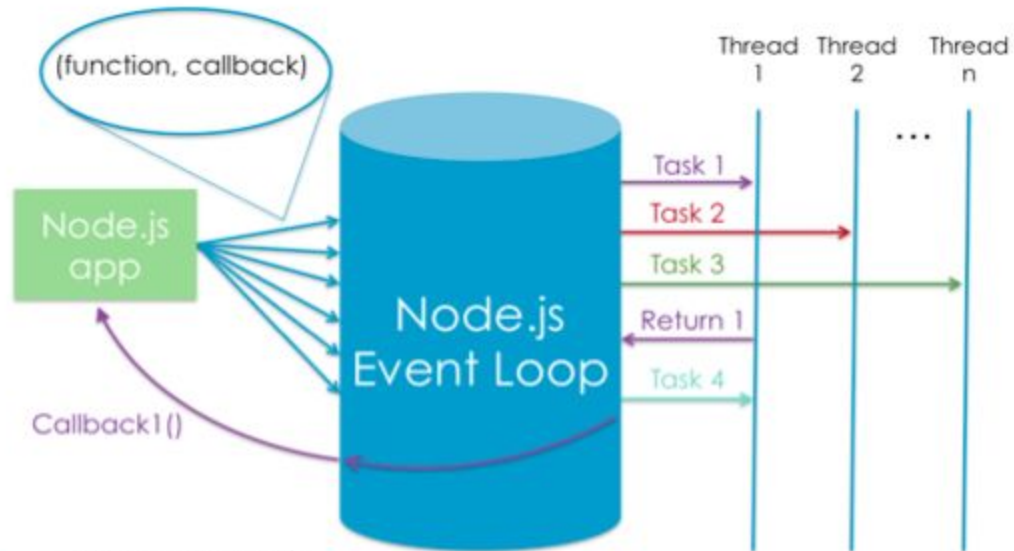
Event Loop



Event Loop

- 1 Node apps pass async tasks to the event loop, along with a callback

- 2 The event loop efficiently manages a thread pool and executes tasks efficiently...



- 3 ...and executes each callback as tasks complete

Examples