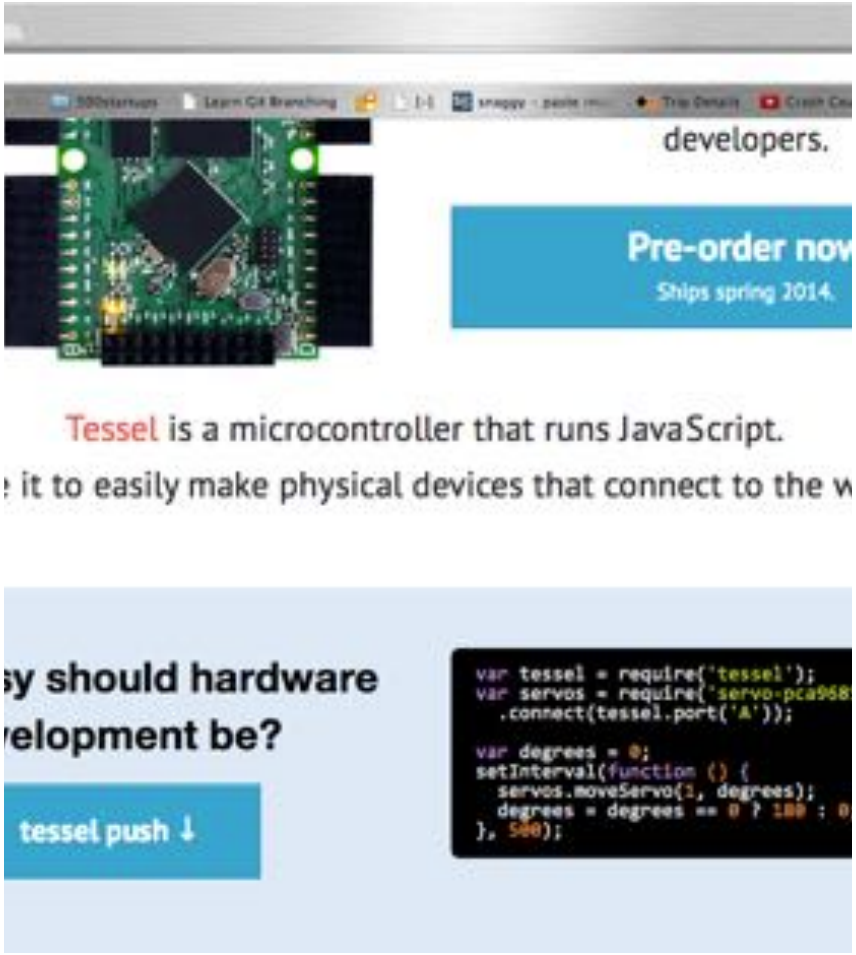
Introduction to Node.js

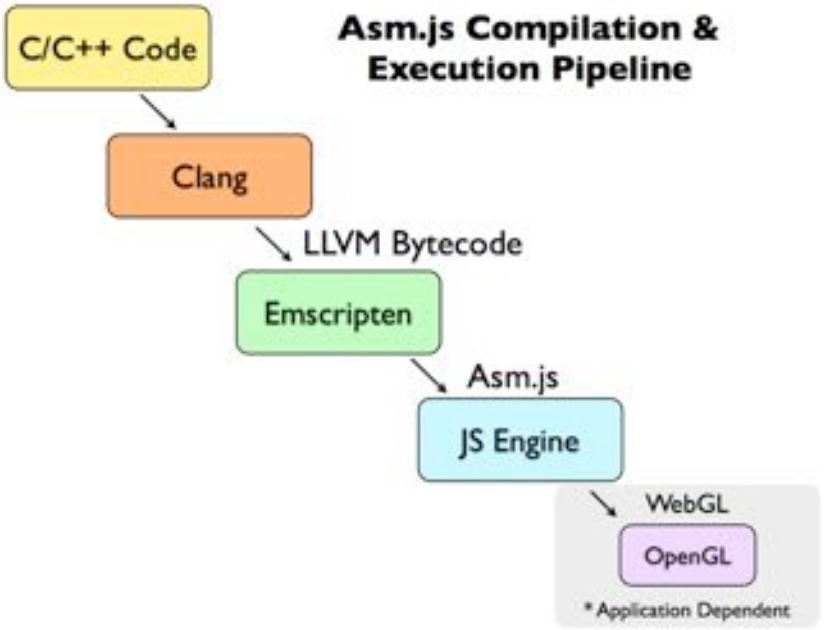
Atwood's Law:

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

Source: http://www.codinghorror.com/blog/2007/07/the-principle-of-least-power.html


developers.
Pre-order nov
Ships spring 2014.
Tessel is a microcontroller that runs JavaScript.
it to easily make physical devices that connect to the w
sy should hardware
velopment be?
tessel push ↓
var tessel = require('tessel');
var servos = require('servo-pca968
.connect(tessel.port('A'));
var degrees = 0;
setInterval(function () {
servos.moveServo(1, degrees);
degrees = degrees == 0 ? 180 : 0
}, 500);

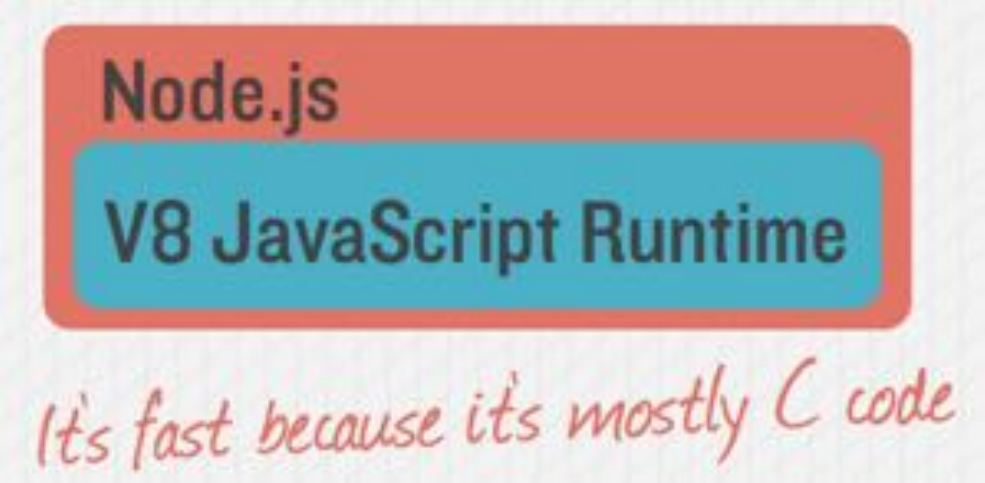

JAVASCRIPT
ALL THE THINGS


Asm.js Compilation &
Execution Pipeline
C/C++ Code
Clang
LLVM Bytecode
Emscripten
Asm.js
JS Engine
WebGL
OpenGL
* Application Dependent


Play
mozilla

What is Node.js?

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



- **asynchronous** i/o framework core in c++ on top of v8
- rest of it in javascript
- swiss army knife for network related stuffs
- can handle **thousands of concurrent** connections with minimal overhead (cpu/memory) on a single process

Async vs Sync

Blocking Code

BlockingCode.txt

```
1 Read file from Filesystem, set equal to “contents”
2 Print contents
3 Do something else
```

Non Blocking Code

NonBlockingCode.txt

```
1 Read file from Filesystem
2   whenever you’re complete, print the contents
3 Do Something else
```

Async vs Sync

Blocking Code

BlockingCode2.js

```
1 var contents = fs.readFileSync('/etc/hosts');
2 console.log(contents);
3 console.log('Doing something else');
```

Non Blocking Code

NonBlockingCode2.js

```
1 fs.readFile('/etc/hosts', function(err, contents) {
2   console.log(contents);
3 });
4
5 console.log('Doing something else');
```

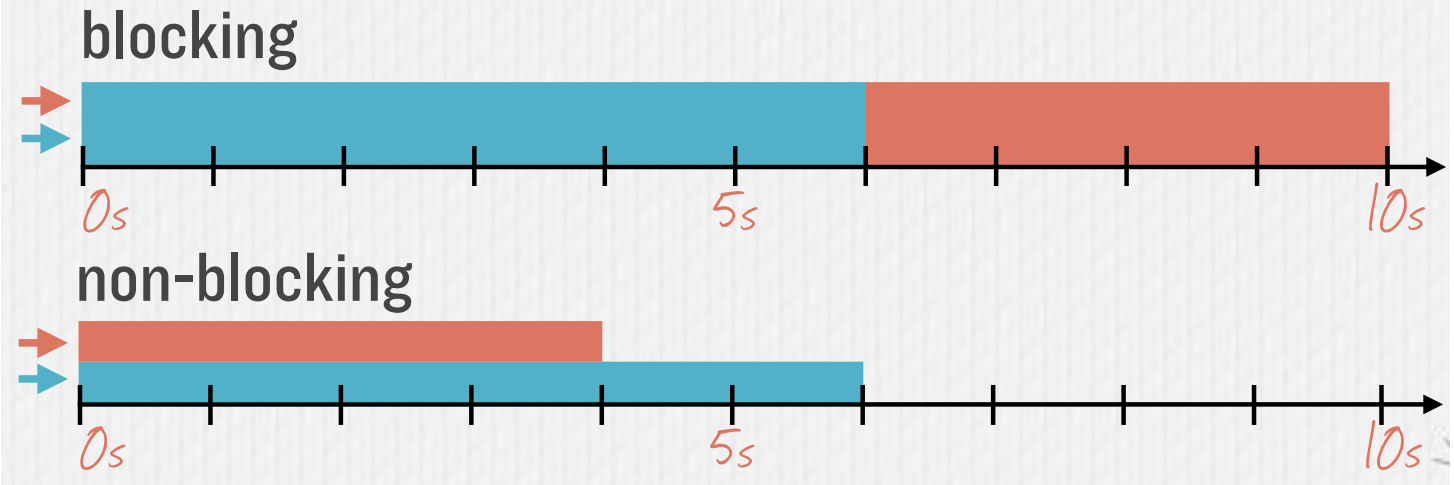
Callbacks

```
ch.js
1 fs.readFile('/etc/hosts', function(err, contents) {
2   console.log(contents);
3 });
4
5
6 var callback = function(err, contents) {
7   console.log(contents);
8 }
9 fs.readFile('/etc/hosts', callback);
```

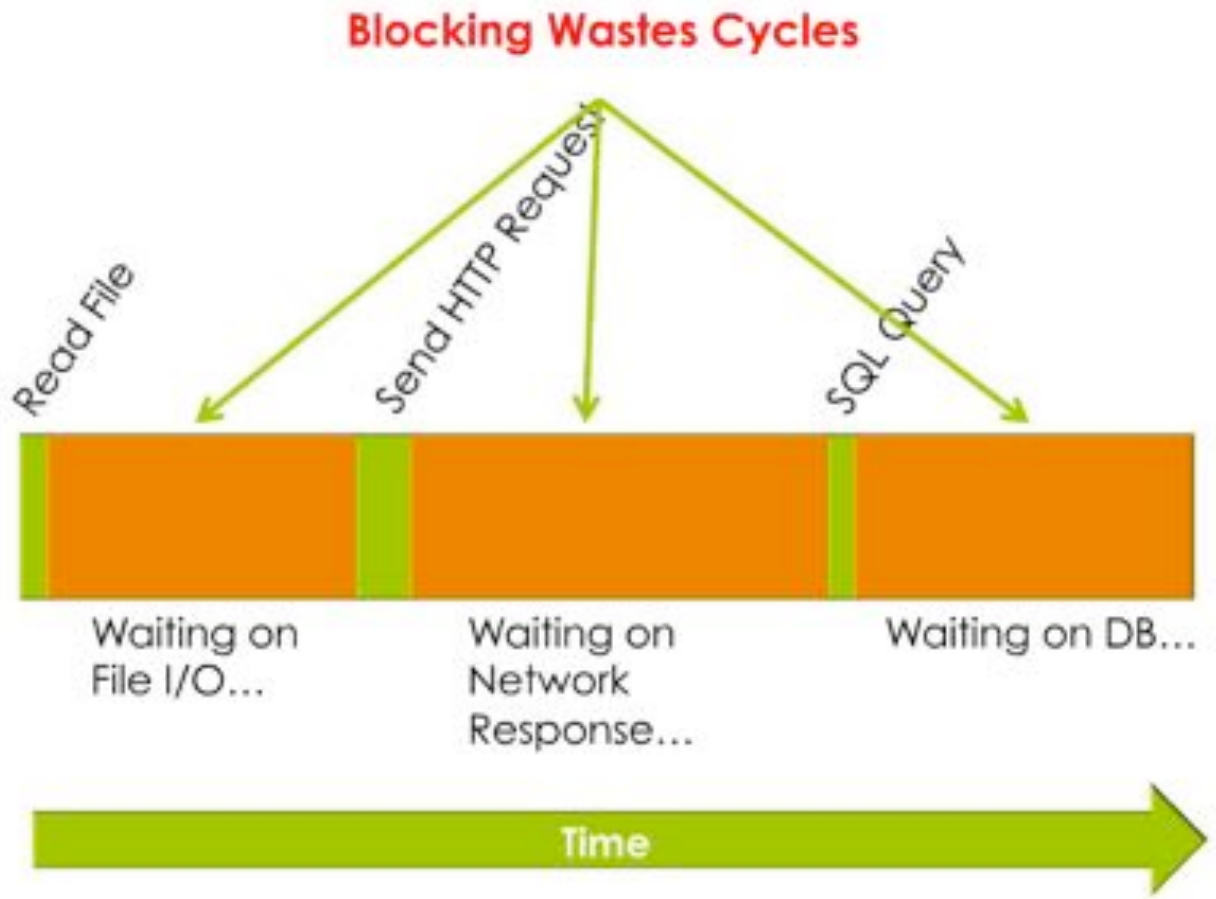
Async vs Sync

BlockVsNonBlock.js

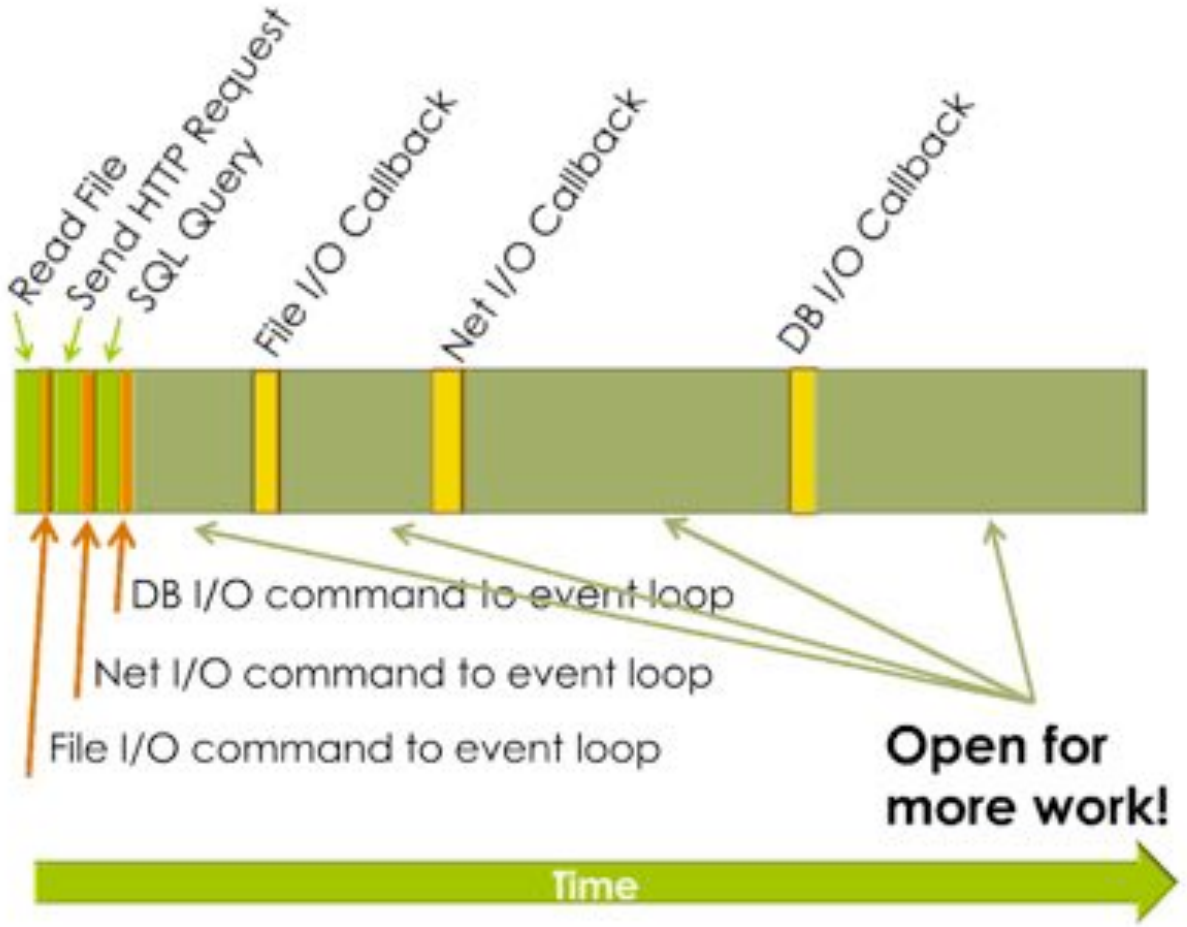
```
1 var callback = function(err, contents) {
2   console.log(contents);
3 }
4
5 fs.readFile('/etc/hosts', callback);
6 fs.readFile('/etc/inetcfg', callback);
```



Blocking


Blocking Wastes Cycles
Read File
Send HTTP Request
SQL Query
Waiting on File I/O...
Waiting on Network Response...
Waiting on DB...
Time

Non blocking



Node.js App

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

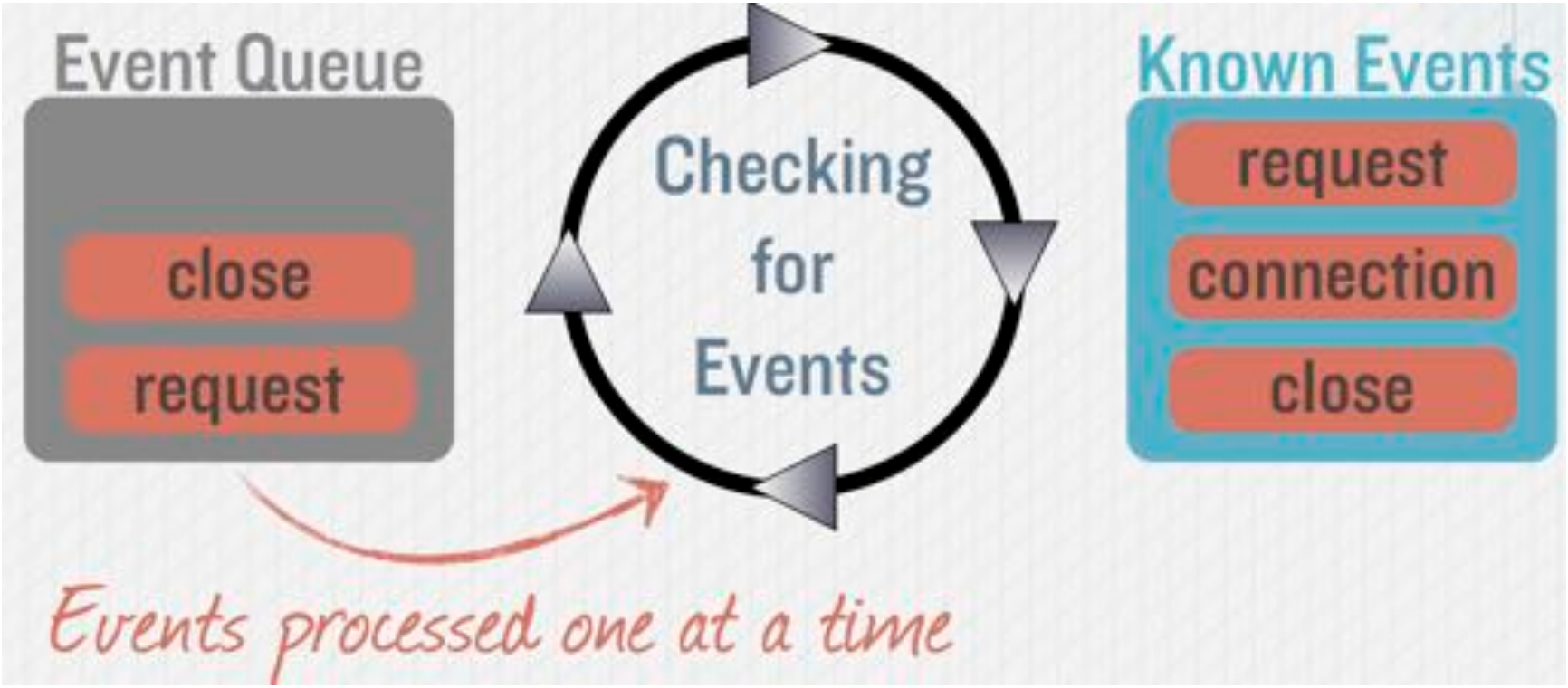
- Your code then:
  - Goes to sleep
  - Uses no system resources
  - Will be notified via callback when I/O is complete

Event (Loop)

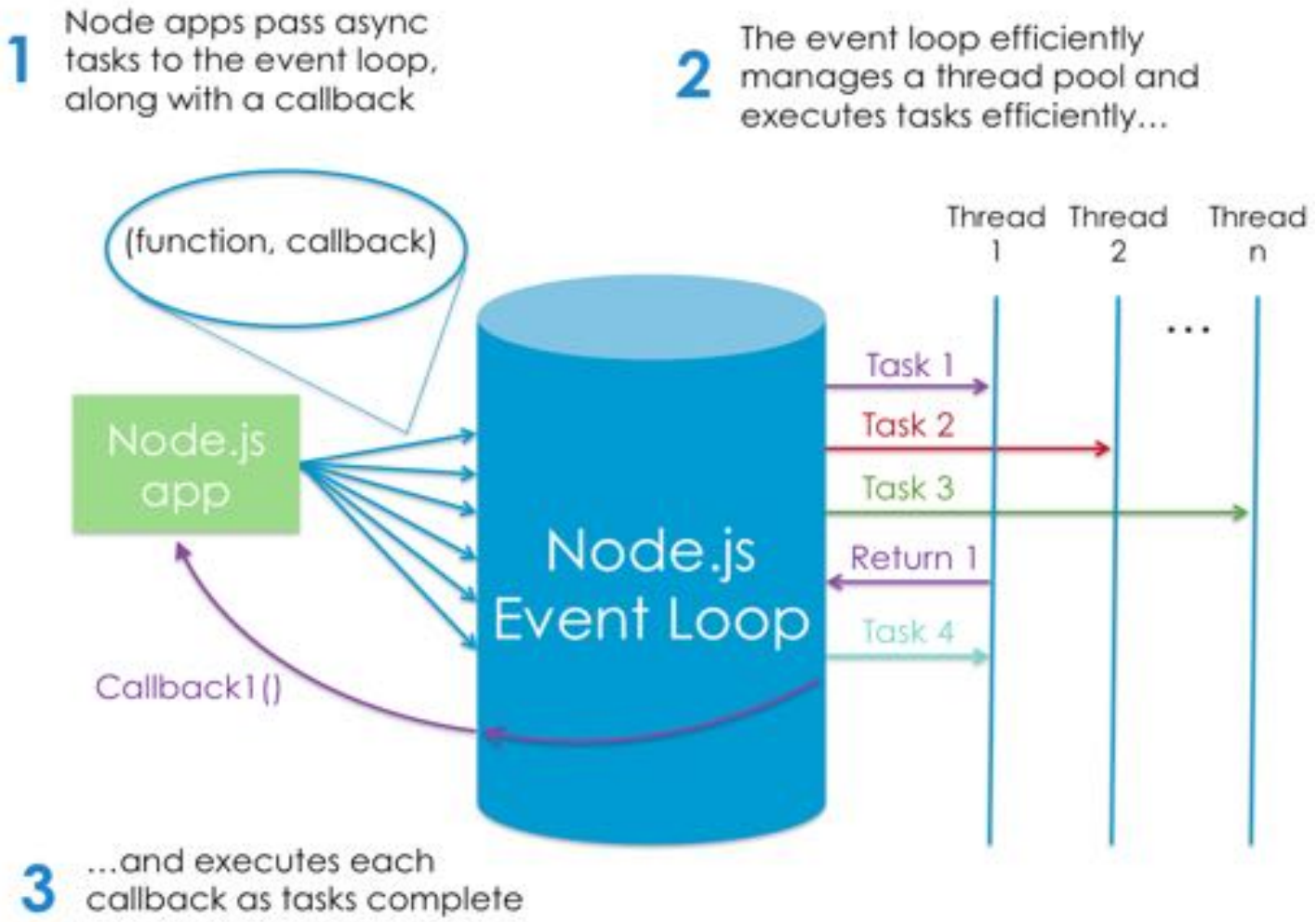
eventEmitter.js

```
1  var EventEmitter = require('events').EventEmitter;
2
3  var logger = new EventEmitter();
4
5  logger.on('error', function(message){
6    console.log('ERR: ' + message);
7  });
8
9  logger.emit('error', 'Spilled Milk');
10
11 logger.emit('error', 'Eggs Cracked');
```

Event Loop


Event Queue
close
request
Checking for Events
Known Events
request
connection
close
Events processed one at a time

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…
(function, callback)
Node.js app
Node.js Event Loop
Thread 1
Thread 2
Thread n
Task 1
Task 2
Task 3
Return 1
Task 4
Callback1()
3
…and executes each callback as tasks complete

Example

References

- https://www.udemy.com/lectures/understanding-the-nodejs-event-loop-91298
- http://www.slideshare.net/gabriele.lana/introduction-to-nodejs
- http://courseware.codeschool.com/node_slides.pdf
- https://speakerdeck.com/faisalabid/node-dot-js-and-you-at-codemotion-roma

More in depth

- https://blog.jcoglan.com/2013/03/30/callbacks-are-imperative-promises-are-functional-nodes-biggest-missed-opportunity/
- http://www.futurealoof.com/posts/nodemodules-in-git.html

Related

- https://npmjs.org/
- https://github.com/visionmedia/express
- https://github.com/remy/nodemon
- https://github.com/node-inspector/node-inspector
- https://github.com/nodejitsu/forever
- https://github.com/kriskowal/q
- https://github.com/visionmedia/debug
- https://bitbucket.org/alemhnan/nestub/