WEEK 6: NODE.JS

HUGH KENNEDY

NODESOURCE™

STACK.GL

DAY 1: NODE 101

INSTALLING NODE

Mac/Win	http://nodejs.org/download
"Other"	http://git.io/wUzU4g

WHAT IS NODE?

Node.js® is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

WAIT, WHAT?

open source
initiative

Paul Irish
@paul_irish

Following

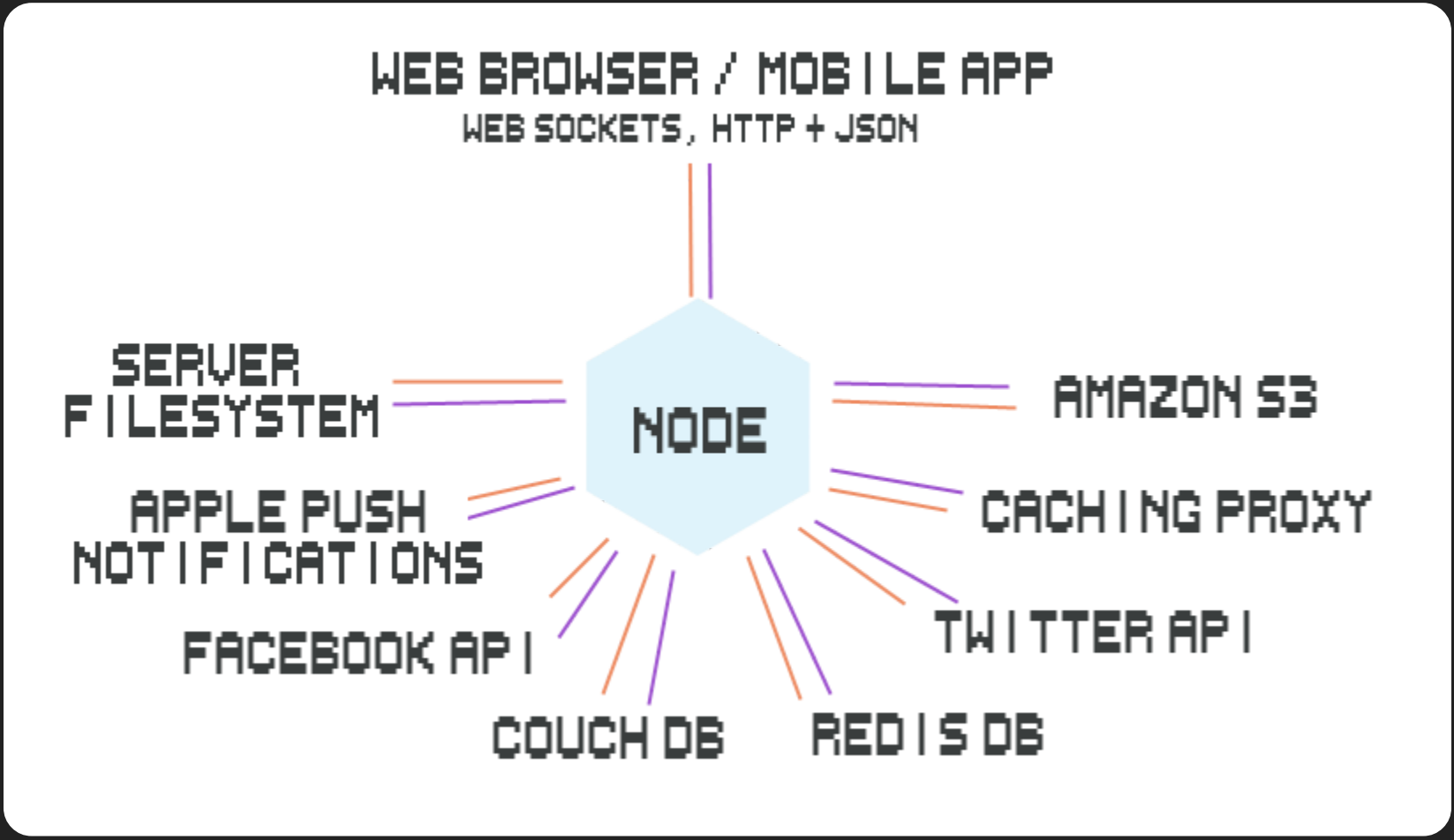
SimCity's UI is completely written in JavaScript running on V8 & WebKit: twitter.com/MaxisScott/sta… news.ycombinator.com/item?id=5359143

RETWEETS 708 FAVORITES 314

3:25 PM - 12 Mar 2013

NODE IS ESSENTIALLY V8, WITH SOME ADDITIONS SO THAT YOU CAN USE IT TO WRITE SERVERS.

INPUT/OUTPUT



node

```
[01:04] ~ > node
> 2 + 2
4
>
```

SID LEE DASHBOARD

WHO USES NODE?

- PayPal
- NASA
- Mapbox
- Yahoo
- Uber
- Walmart
- Facebook

DO NOT TOUCH

EELS 3D projection mapping multiplayer game

from **B-Reel** PRO

02:08

HD

WHY NODE?

NODE IS GOOD AT ASYNCHRONOUS PROGRAMMING

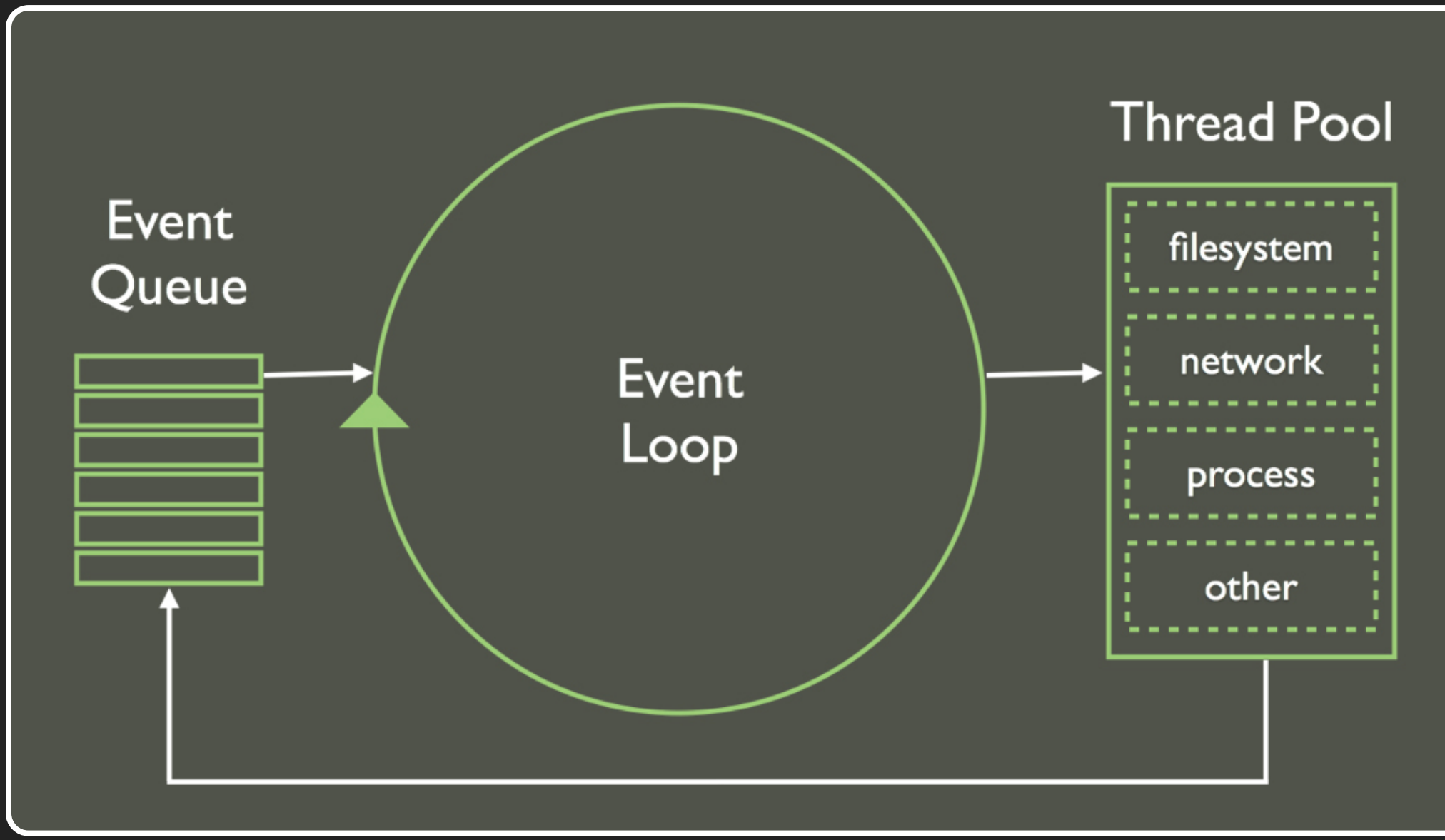
BLOCKING I/O

```
var data = fs.readFileSync('index.js', 'utf8')
```

Class	Operation	Time cost
Memory	L1 cache reference:	1 ns
	L2 cache reference:	4 ns
	Main memory reference:	100 ns
I/O	SSD random-read:	16,000 ns
	Round-trip in same datacenter:	500,000 ns
	Physical disk seek:	4,000,000 ns
	Round-trip from US to EU:	150,000,000 ns

NON-BLOCKING I/O

```
fs.readFile('index.js', 'utf8', function(err, data) {
  if (err) throw err
  console.log(data)
})
```


Event
Queue
Event
Loop
Thread Pool
filesystem
network
process
other

RUNNING NODE

```
// index.js
console.log('hello world')
```

bash

```
[02:09] ~ > node index.js
hello world
[02:09] ~ >
```

node

```
[01:04] ~ > node
> 2 + 2
4
>
```

PROCESS.ARGV

```
// index.js
console.log(process.argv)
```

bash

```
[02:12] ~ > node index.js hello world
[ 'node', '/Users/hughsk/index.js', 'hello', 'world' ]
[02:12] ~ >
```


npm

NPM MAKES NODE GREAT

ANYONE CAN INSTALL

ANYONE CAN PUBLISH

>125,000 PACKAGES

>200 PACKAGES/DAY

- Server Frameworks
- Browser Libraries
- Build Tools
- Web APIs
- Protocols
- Computational Geometry
- Image/Audio Processing
- 3D Graphics
- Scientific Computing
- Databases
- Statistical Analysis

```
$ npm install stats-lite
```

```
$ ls -1 node_modules
stats-lite/
```

```
// index.js
var stats = require('stats-lite')

console.log(stats.stdev([1, 5, 6, 1, 2, 0]))
```

```
$ node index.js
2.217355782608345
```

NPM INSTALL GIPHY

PLAY AROUND WITH GIPHY'S API IN NODE

```
var token = 'dc6zaTOxFJmzC'
var giphy = require('giphy')(token)

giphy.translate({}, function(err, data) {
  if (err) throw err
})
```

ASSIGNMENT #1

GIPHY TRANSLATE TOOL

Use Giphy's "translate" API to create a script that outputs GIF based on user input. If you have extra time to spare, try jumping ahead and making a web server that serves these images to a browser based on the specified URL.

ASYNCHRONOUS PATTERNS

CALLBACKS

```
var doThing = function(callback) {
  setTimeout(function() {
    callback(null, { ok: true })
  }, 100)
}
```

```
doThing(function(err, result) {
  if (err) throw err

  console.log(result.ok)
})
```

EVENTEMITTERS

```
var code = new EventEmitter

setInterval(function() {
  code.emit('data', Math.random())
}, 1000)
```

```
code.on('data', function(result) {
  console.log(result)
})
```

STREAMS

We should have some ways of connecting programs like garden hose – screw in another segment when it becomes necessary to massage data in another way. This is the way of IO also.

Doug McIlroy. October 11, 1964

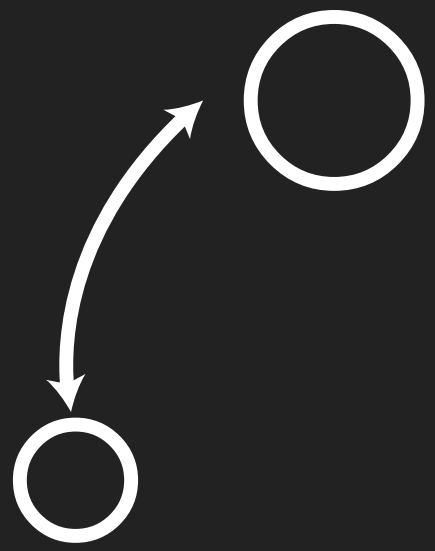
```
# Get unique values from column 1 of a .csv file
# Write output to index.txt
$ cat index.csv | cut -d, -f1 | uniq > index.txt
```

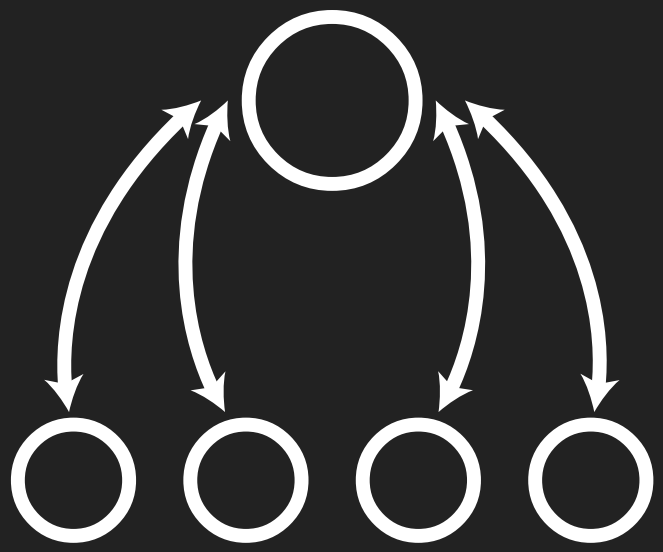
```
fs.createReadStream('index.csv')
  .pipe(split('\n')) // Split on newlines
  .pipe(csv2(','))   // Split each line into values (-d,)
  .pipe(pick(0))     // Pick the first value (-f1)
  .pipe(unique())    // Only output unique values (uniq)
  .pipe(join('\n'))  // Add the newlines back into your output
  .pipe(fs.createWriteStream('index.txt'))
```

STREAMS ARE HARD TO WRITE

BUT EASY TO USE

SERVERS AND CLIENTS





TCP

A "BASELINE" FOR RAW DATA

```
var net = require('net')

var server = net.createServer(function(connection) {
  connection.write('hello\n')
  connection.write('world\n')
  connection.end()
})

server.listen(9999, function(err) {
  if (err) throw err
  console.log('telnet localhost 9999')
})
```

bash

bash | node | bash | node | node | bash

```
[13:42] ~ > telnet localhost 9999
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
hello
world
Connection closed by foreign host.
[13:42] ~ >
```

```
var net = require('net')

var client = net.connect({
  host: 'localhost',
  port: 9999
})

client.pipe(process.stdout)
```

```
var net = require('net')

var client = net.connect({
  host: 'localhost',
  port: 9999
})

process.stdin
  .pipe(client)
  .pipe(process.stdout)
```

```
var net = require('net')

var server = net.createServer(function(connection) {
  console.log('connection established')
  connection.write('hello\n')
  connection.write('world\n')

  connection.pipe(process.stdout)
})

server.listen(9999, function(err) {
  if (err) throw err
  console.log('telnet localhost 9999')
})
```

bash

bash | node | bash | node | node | bash

```
[13:54] ~ > echo 'lorem ipsum' | node index.js
hello
world
[13:54] ~ > echo 'hello again' | node index.js
hello
world
[13:54] ~ >
```

node

bash | node | bash | node | node | bash

```
connection established
lorem ipsum
connection established
hello again
```

ASSIGNMENT #2

CREATE A TCP CHAT SERVER/CLIENT

- You should be able to start the server using *node server.js*, and have it listen for incoming connections in the background.
- The client may send messages using *node client.js <server-address> <message>*.
- Try adding some functionality to the server or client: colorized text, emoji support, ASCII art, are some fun examples.
- Try making the chat client persistent: that is to say, you can start it up once and continue typing messages without having to restart the process.