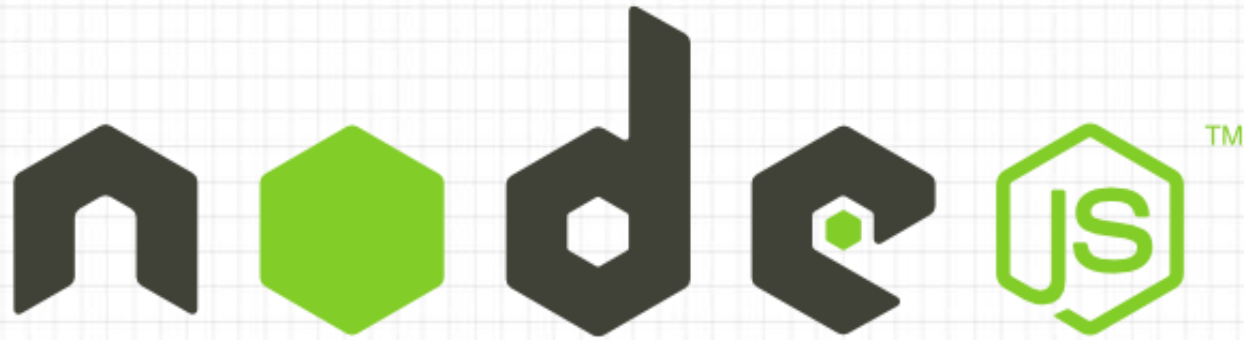


Node.JS Overview



Eyal Vardi
Microsoft MVP ASP.NET
blog: eyalvardi.wordpress.com





*"Everything that can be written in JavaScript
will eventually be written in JavaScript".*

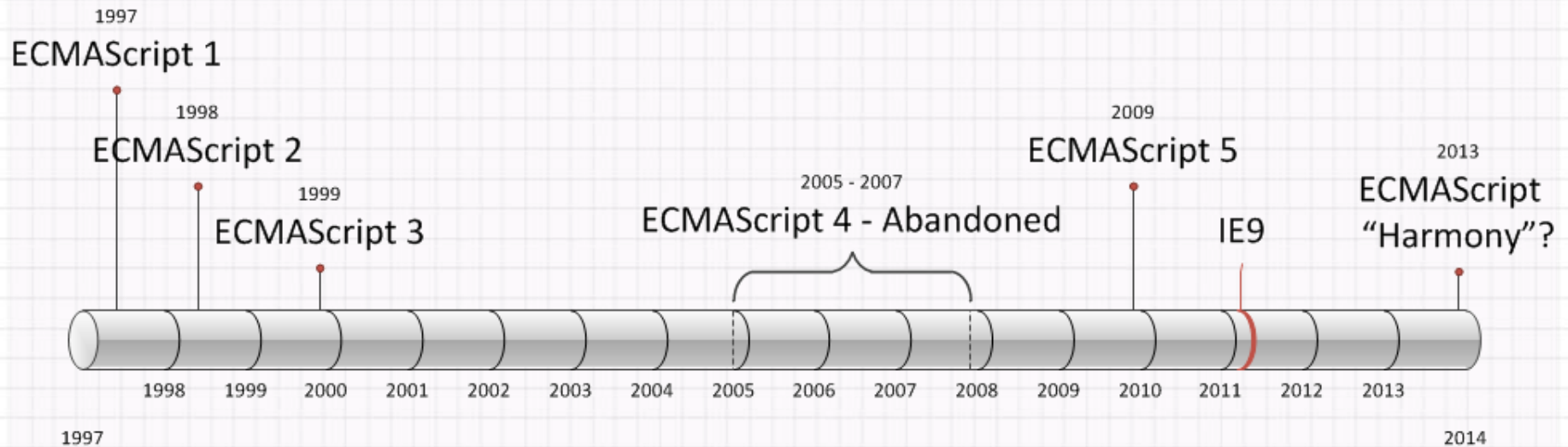
([Atwood's law](#))

Agenda

- JavaScript History
- Node Has Arrived
- Node Global Objects
- Modules System

JavaScript History

ECMAScript Versions



Node Has Arrived

NODE IS MORE POPULAR THAN EVER

Five years after its debut, Node.js is the third most popular project on GitHub.

OVER
2 MILLION
DOWNLOADS PER MONTH

OVER
20 MILLION
DOWNLOADS OF v0.10x

OVER
475
WORLDWIDE MEETUPS

Node Has Arrived

NODE IS GROWING FASTER THAN EVER

Companies are hiring Node developers and everyone wants to learn more

Job Trends (Report from Indeed.com)



Search Trends (Interest over time on Google)



Node Has Arrived

NODE IS DEPLOYED BY BIG BRANDS

Big brands are using Node to power their business

Manufacturing



SIEMENS

Financial



eCommerce



Media



Technology



Node Has Arrived

COMPANIES SUPPORT NODE

Companies are committing resources to making Node.js more robust.

Top 5 Companies

As Node.js v0.10 becomes more critical to business, companies retain developers to contribute code to the project.

StrongLoop



Joyent



Voxer



Microsoft



Mozilla

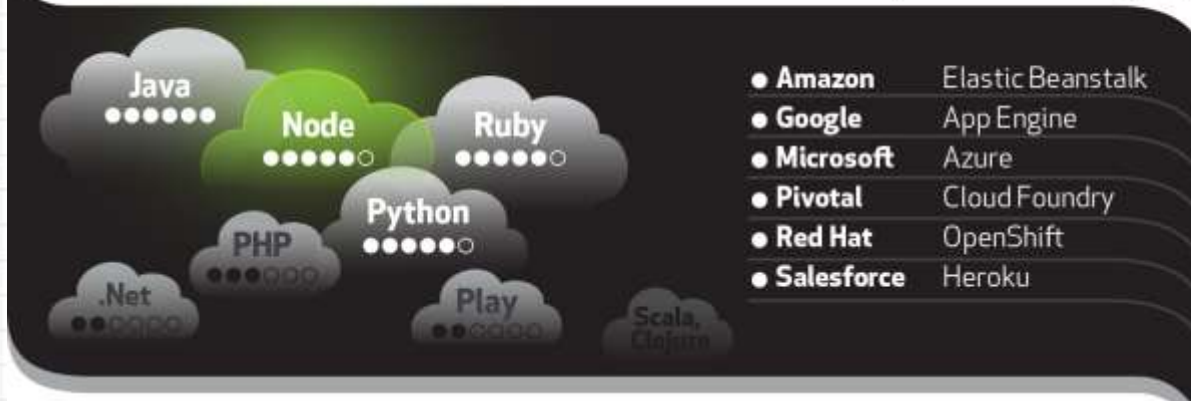


	StrongLoop		Joyent		Voxer		Microsoft		Mozilla	
	Node.js	Libuv	Node.js	Libuv	Node.js	Libuv	Node.js	Libuv	Node.js	Libuv
Count Of Patches	558	619	791	25	95	20	62	0	52	0
Lines Of Code	1,093,784	194,622	658,376	2,231	168,973	3,819	3,349	0	214,829	0
Lines Added/Changed	499,795	26,097	319,739	610	76,074	1,785	934	0	76,586	0

Node Has Arrived

NODE IS TOP 4 IN THE CLOUD

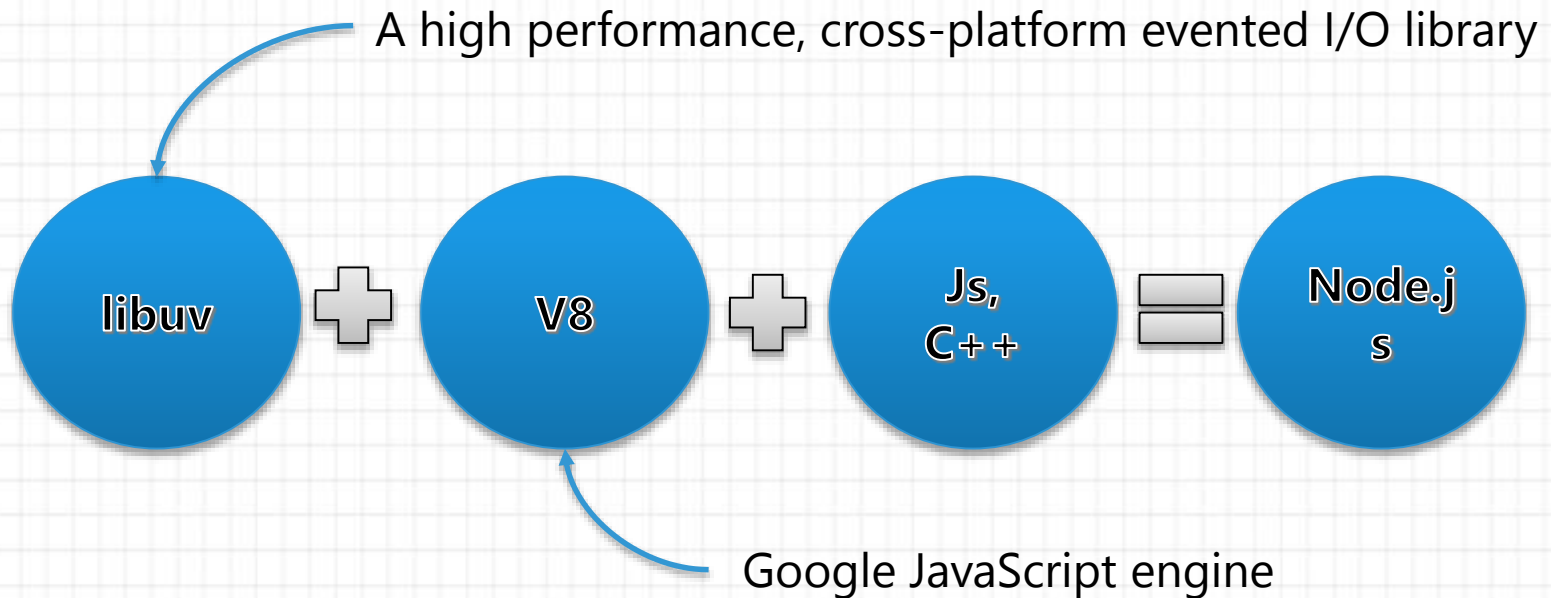
Node.js is one of the top four languages, supported by 5 of the 6 major platform-as-a-service providers.



And It's Here To Stay

For source material and more information, visit www.strongloop.com/infographic

Node.js Building Blocks



Node.JS Advantages

Architecture

- Single Thread
- App == Server
- Middleware

Deployment

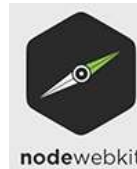
- XCopy
- Run Everywhere

Community

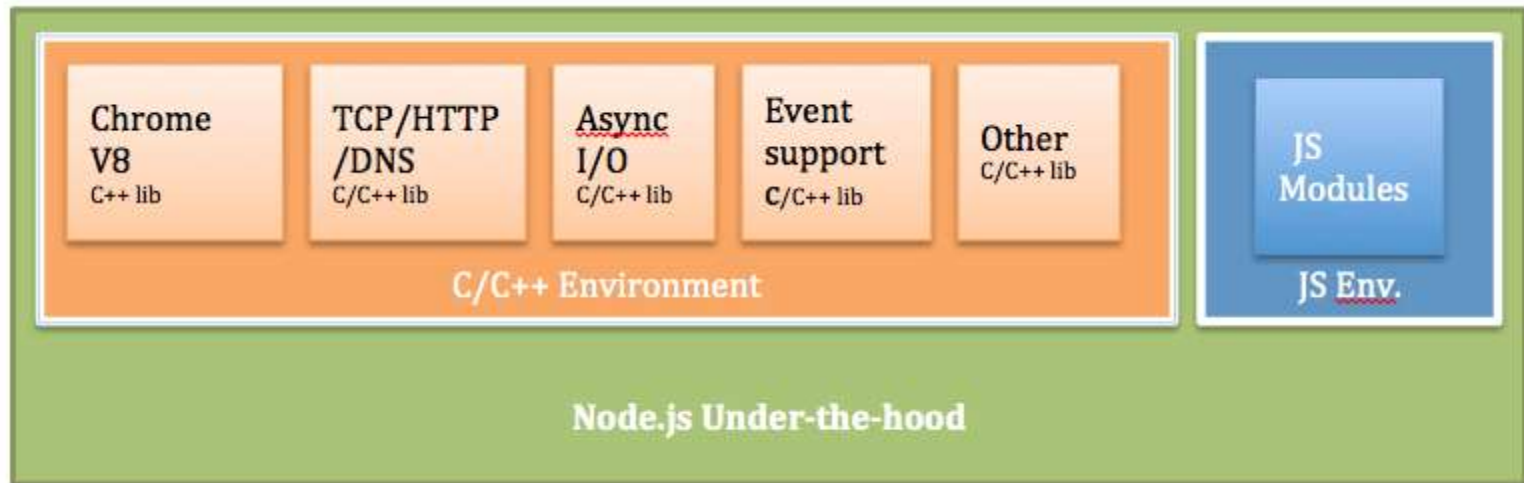
- 53,228 Packages
- 2.5M Download in day.



Tools

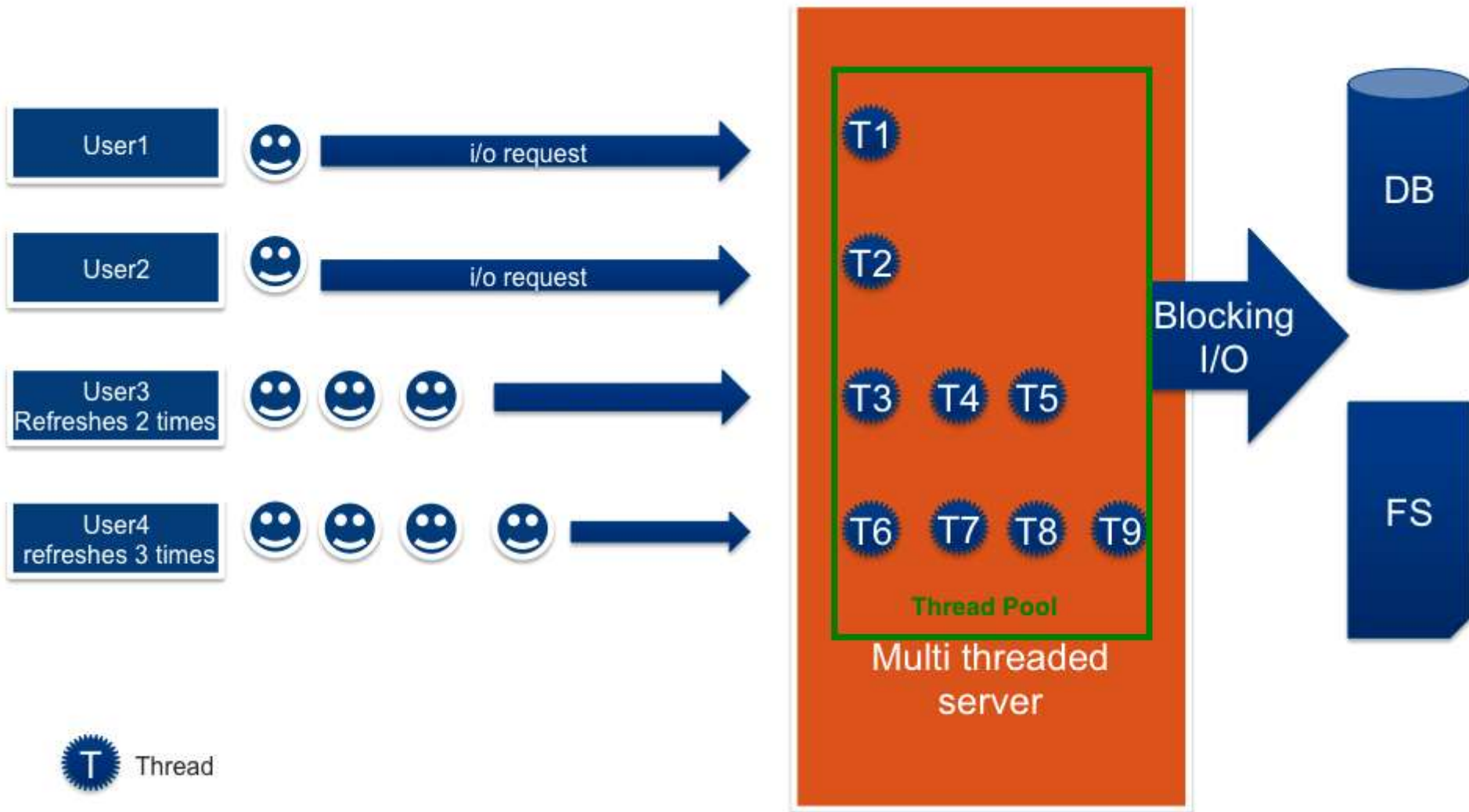


Node.js Under The Hood



(source: [Future-proofing Your Apps: Cloud Foundry and Node.js](#))

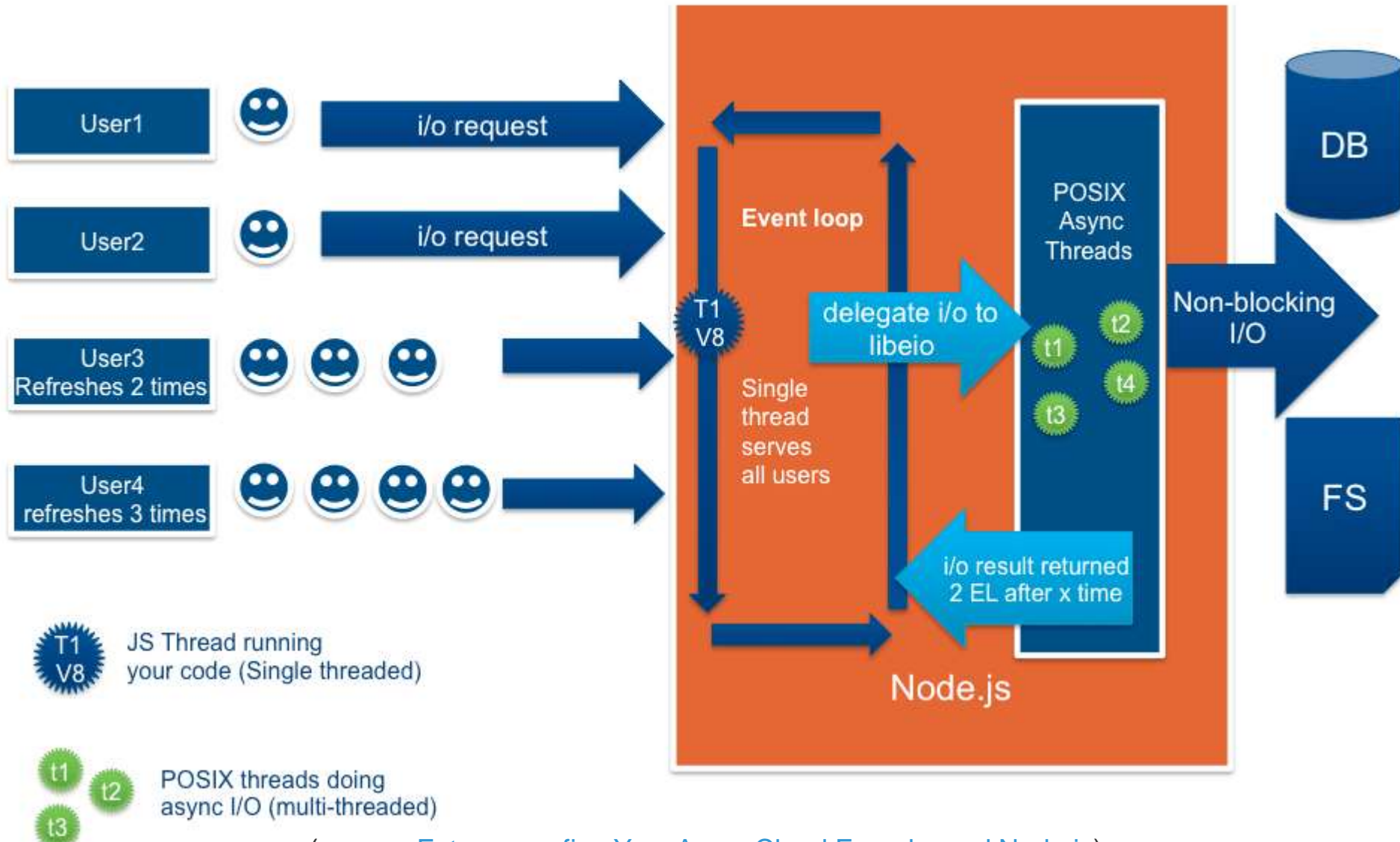
Multi-threaded HTTP Server Using Blocking I/O



Blocking I/O + direct interface to clients leads to thread-pools w/ larger number threads and more memory requirements

(source: [Future-proofing Your Apps: Cloud Foundry and Node.js](#))

Event-driven, Non-Blocking I/O



(source: [Future-proofing Your Apps: Cloud Foundry and Node.js](#))

demo

Node.js Hello World

Global Objects

- These objects are available in **all modules**. Some of these objects aren't actually in the **global scope but in the module scope**.

Objects:

- `global`
- `process`
- `Console`
- `module`
- `exports`

functions:

- `Buffer`
- `require()`
- `setTimeout(cb,ms)` , `clearTimeout(t)`
- `setInterval(cb,ms)` , `clearInterval(t)`

Modules System

Modules

- Node has a simple module loading system.
 - Files and modules are in one-to-one correspondence.

foo.js

```
var circle = require('./circle.js');  
console.log('The area of radius 4: ' + circle.area(4));
```

circle.js

```
var PI = Math.PI;  
exports.area = function (r) {return PI * r * r;};  
exports.circumference = function (r) {return 2 * PI * r;};
```

The variable **PI** is
private to circle.js

exports === module.exports

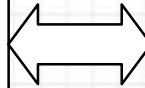
The module Object

- In each module, the **module** free variable is a reference to the object representing the current module.
 - For convenience, **module.exports** is also accessible via the **exports** module-global.
- Module object properties:
 - id
 - filename
 - loaded
 - parent
 - children

Module Cycles

```
console.log('a starting');
exports.done = false;
var b = require('./b.js');
console.log('in a,b.done= %j', b.done);
exports.done = true;
console.log('a done');
```

a.js



```
console.log('b starting');
exports.done = false;
var a = require('./a.js');
console.log('in b,a.done= %j', a.done);
exports.done = true;
console.log('b done');
```

b.js

```
console.log('main starting');
var a = require('./a.js');
var b = require('./b.js');
console.log('in main, a.done=%j, b.done=%j', a.done, b.done);
```

main.js

b.js tries to load **a.js**. In order to prevent an infinite loop an **unfinished copy** of the **a.js** exports object is returned to the **b.js** module.

```
main starting
a starting
b starting
in b, a.done = false
b done
in a, b.done = true
a done
in main, a.done=true, b.done=true
```

Output

Core Modules

- Node has several modules **compiled** into the **binary**.
- The core modules are defined in node's source in the **lib/** folder.
- Core modules name:
 - Assert
 - Buffer
 - child_process
 - Cluster
 - Crypto
 - Dgram
 - Dns
 - Events
 - Fs
 - http
 - https
 - Net
 - Os
 - Path
 - Punycode
 - Querystring
 - Readline
 - Repl
 - string_decoder
 - Tls
 - Tty
 - url
 - Util
 - Vm
 - zlib

File Modules

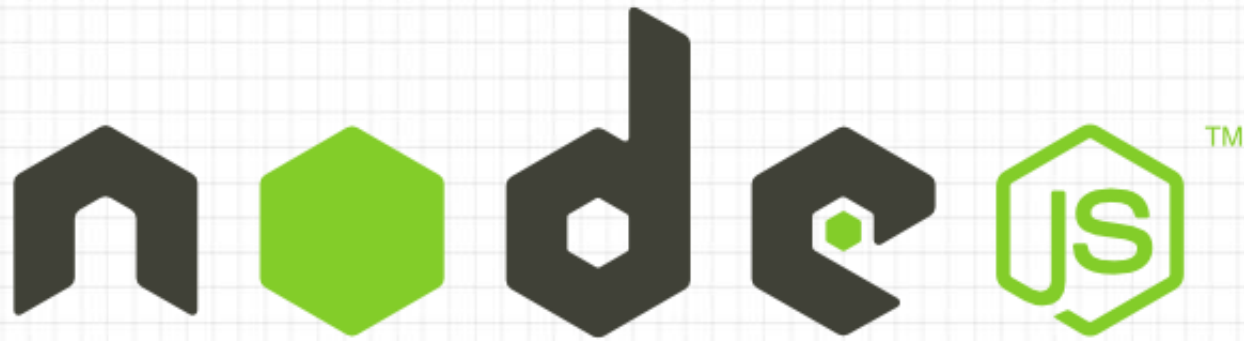
- If the exact filename is **not found**, then node will attempt to load the required filename with the added extension of **.js, .json, and then .node**.
- A module prefixed:
 - `'/'` is an absolute path to the file.
 - `'./'` is relative to the file calling `require()`.
 - Without a leading `'/'` or `'./'` to indicate a file, the module is either a "**core module**" or is loaded from a **node_modules** folder.

Folders as Modules

- There are two ways in which a folder may be passed to `require()` as an argument.
 - Create a **package.json** file in the root of the folder, which specifies a **main module**.
 - If there is no `package.json` file present in the directory, then node will attempt to load an **index.js** or **index.node** file out of that directory.

Module Caching

- Modules are cached after the first time they are loaded.
- Multiple calls to `require()` may **not** cause the module code to be **executed multiple times**.
- Modules are cached based on their resolved filename.
 - Since modules may resolve to a different filename based on the location of the calling module, it is not a guarantee that `require()` will always return the exact same object, if it would resolve to different files.



Thanks

eyalvardi.wordpress.com



Eyal Vardi
Microsoft MVP ASP.NET
blog: eyalvardi.wordpress.com

