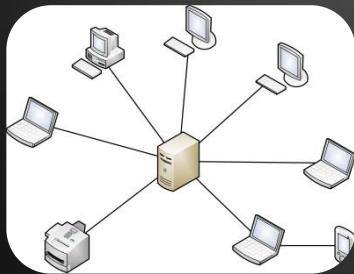




Introduction to NodeJS



What is the fuzz all about?

Telerik Software Academy
Learning & Development Team
<http://academy.telerik.com>



1. Overview of NodeJS

1. Building and installing NodeJS
2. Developing IDEs
3. What is the Event Loop?
4. Writing code with callbacks

2. Modules

1. Using modules
2. Installing modules



Overview of NodeJS



◆ Background

The collage consists of three overlapping screenshots:

- Top-left screenshot:** A web browser window showing a presentation slide titled "Ryan Dahl: Node.js, Evented I/O for V8 Javascript". The slide includes a photo of Ryan Dahl and a block of text explaining the benefits of event loops over synchronous I/O.
- Middle-left screenshot:** A GitHub interface showing the "Popular Starred Repositories" page. The "node" repository is listed as the top starred project, described as "evented I/O for v8 javascript". Other repositories like "bootstrap", "jquery", "html5-boilerplate", and "rails" are also visible.
- Bottom-right screenshot:** A video player showing a video titled "Ryan Dahl: Original Node.js presentation". The video frame shows Ryan Dahl speaking at a podium. Below the video player are controls for volume, playback progress (00:26 / 48:32), and other video settings.

- ◆ Node is written in JavaScript
 - ◆ One language on the server and the client
- ◆ Full control of the server
- ◆ Asynchronous and fast (callback oriented)

Number of iterations	Node.js	PHP
100	2.00	0.14
10'000	3.00	10.53
1'000'000	15.00	1119.24
10'000'000	143.00	10621.46
1'000'000'000	11118.00	1036272.19

Building Blocks & Installation

- ◆ NodeJS

- ◆ libuv – high-performance event I/O library
- ◆ V8 – Google Chrome's JavaScript engine
- ◆ JavaScript -> C++

- ◆ Installation

- ◆ <http://nodejs.org/>
- ◆ Run Command Prompt (cmd)
- ◆ Type "node" and run it

Developing IDEs

- ◆ IDEs

- ◆ JetBrains WebStorm



- ◆ Sublime Text 2/3

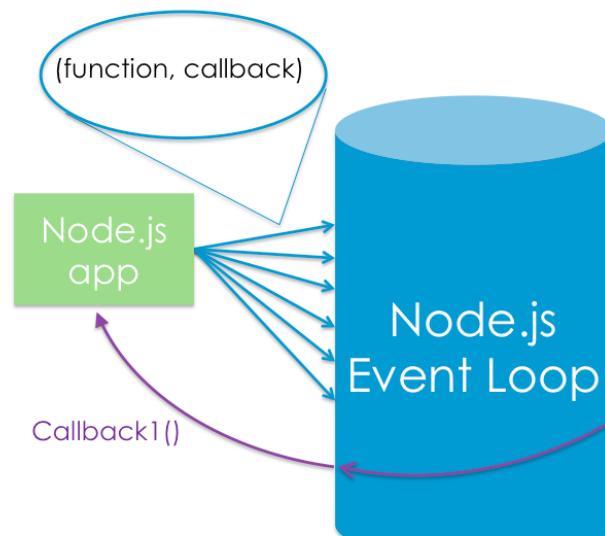


- ◆ Cloud9

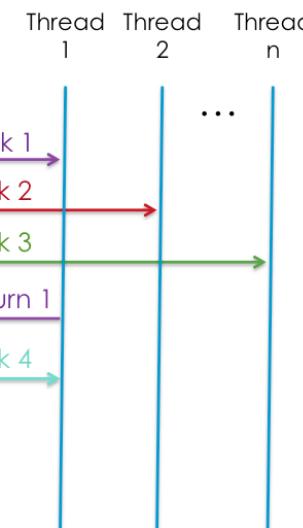


The Event Loop

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

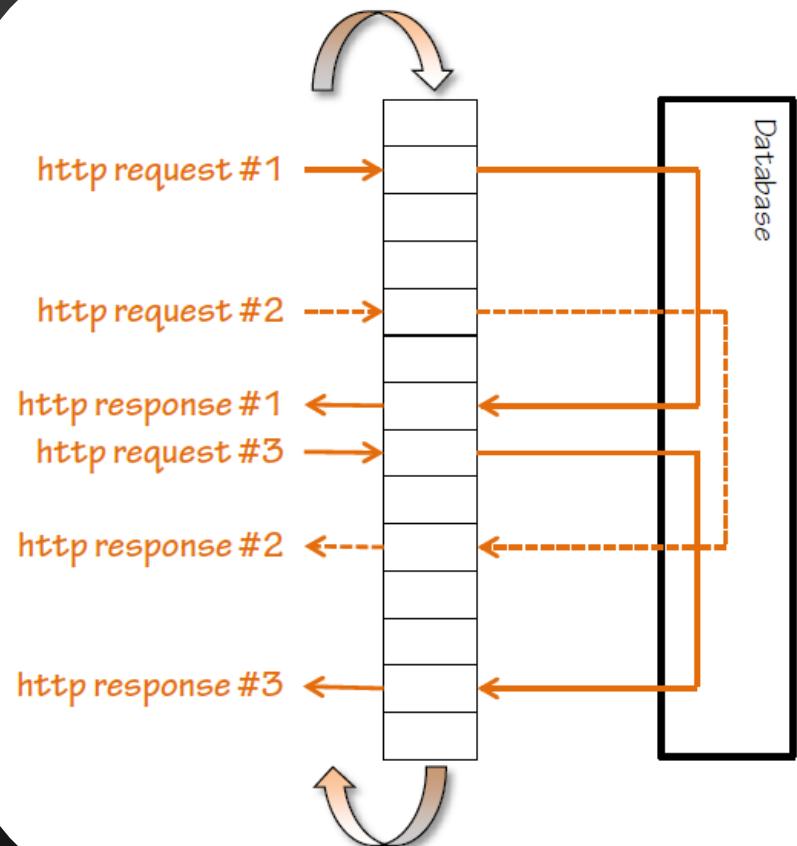
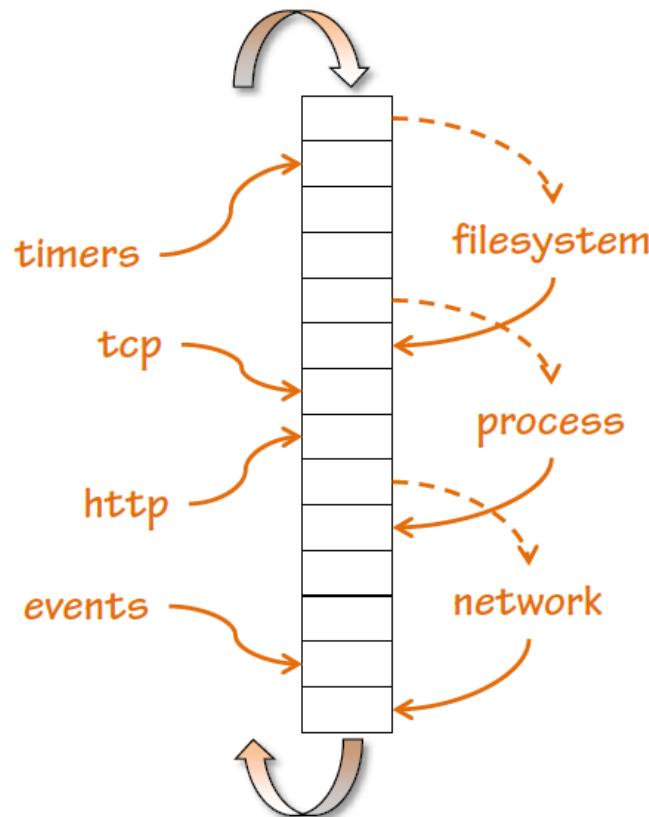


- 2 The event loop efficiently manages a thread pool and executes tasks efficiently...



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

The Event Loop



◆ Standard way

```
var conn = getDbConnection(connectionString);
var stmt = conn.createStatement();
var results = stmt.executeQuery(sqlQuery);
for (var i=0; i<results.length; i++) {
    // print results[i];
}
```

◆ Callback approach

```
getDbConnection(connectionString, function(err, conn) {
    conn.createStatement(function(err, stmt) {
        var results = stmt.executeQuery(sqlQuery);
        results.on('row', function(result) {
            // print result
        });
    });
});
```

◆ Convention

- **Callback is last parameter in the async call**
- **Error is first parameter in the callback**

```
var handleResults = function(error, results) {  
    // if error is undefined...  
    // do something with the results  
}  
  
getStuff(inputParam, handleResults);
```

- ◆ For simple uses – anonymous function

```
getStuff(inputParam, function(error, results) {  
    // if error is undefined...  
    // do something with the results  
});
```

- ◆ Closures are your friend

```
someOtherFunction(function(err, stuffToGet) {  
    var foo = 23;  
    getStuff(stuffToGet, function(error, results) {  
        // if error is undefined...  
        // do something with the results (and foo)  
    });  
});
```

- ◆ Do not overuse!

Asynchronous Code

Live Demo

Using Modules



- ◆ Modules are used with "require"

```
var first = require('first');
var Second = require('second');
var justPart = require('largeModule').justPart;

var propertyResult = 2 + first.property; // export variable
var functionResult = first.function() * 3; // export function

var second = new Second(); // export object

console.log(justPart()); // export part of object
```

◆ Built-in modules

- ◆ Come with Node
- ◆ Are "require"-ed with string identifier

```
var fs = require('fs');
```

◆ Commonly used modules

- ◆ fs, http, crypto, os
- ◆ More at <http://nodejs.org/api/>

Built-in Modules

Live Demo

- ◆ Each .js file is a different module
 - ◆ Are "require"-ed with file system semantics
 - ◆ ".js" is not needed in the string

```
var data = require('./data'); // in same directory
var a = require('./other/a'); // in child directory
var b = require('../lib/b'); // in parent directory's child
var justPart = require('~/data').part; // just part of module
```

- ◆ Variable are exported with module.exports

```
// first.js

var count = 2;
var doIt = function(i, callback) { ... }
module.exports.doIt = doIt;
module.exports.someVar = 'result';
```

```
// second.js

var one = require('./first');
one.doIt(23, function (err, result) {
  console.log(result);
});
console.log(one.someVar);
console.log(one.count); // invalid
```

Your Modules

Live Demo

Third-Party Modules

- ◆ Third-Party Modules

- ◆ Installed from Node Package Manager (NPM)
- ◆ Command: "npm install *mdl_name*"
- ◆ Are "require"-ed with string identifier

```
var request = require('request');
```

- ◆ Some modules have command line tools
- ◆ Command: "npm install -g *mdl_name*"
 - ◆ Example: Express, Mocha

Third-Party Modules

Live Demo

- ◆ <http://nodejs.org/> - NodeJS official web site
- ◆ <http://nodejs.org/api/> - API documentation
- ◆ <http://blog.nodejitsu.com/npm-cheatsheet> - NPM documentation
- ◆ <https://npmjs.org/> - NPM official web site
- ◆ <https://github.com/felixge/node-style-guide> - NodeJS style guide

Introduction to NodeJS

Questions?