Introduction to Node.js

By Dhanesh Purohit

What is Node.js?

Node.js is a platform built on Chrome's JavaScript runtime (v8) for easily building fast, scalable network applications.

Why use Node.js?

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.

Features

- A JavaScript runtime environment running Google Chrome’s V8 engine
- A server-side solution for JS
- Compiles JS, making it really fast
- Runs over the command line
- Designed for high concurrency
  - Without threads or new processes
- Never blocks, not even for I/O
- Uses the CommonJS framework
  - Making it a little closer to a real OO language

The customary Hello World program

```
console.log("Hello World!");
```

The asynchronous nature of JS

```
setTimeout(function(){ console.log(" World!"); }, 1000);

console.log("Hello");
```

But how do you write complex applications?

What about libraries?

The Node.js Standard Library

- Testing
- Interop
- OS
- File System
- Clustering
- Console
- Cryptography
- Debugging
- DNS
- Domain
- Events
- HTTP / HTTPS
- Path Manipulation
- Process Manipulation
- Buffers / Streams
- Timers
- URL Manipulation
- SSL
- Virtualization
- Utilities for Data format manipulation
- Compression
- Networking
- REPL

How do I use libraries? (using?)

```
var obj = require('module_name');
```

Node has several modules compiled into the binary. Core modules are always preferentially loaded if their identifier (read name) is passed to require().

For instance, require('http') will always return the built in HTTP module, even if there is a file by that name. And if the module is an external module then node will search for it in the below paths

- ./node_modules/http.js
- ./node_modules/http/index.js
- ./node_modules/http/package.json

What if I have multiple code files?

- A module prefixed with '/' is an absolute path to the file. For example, require('/home/marco/foo.js') will load the file at /home/marco/foo.js.
- A module prefixed with './' 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.
- If the given path does not exist, require() will throw an Error with its code property set to 'MODULE_NOT_FOUND'.

Now that we've learned how to use modules

Let's see some Node.js code in action

Reading Files (Async)

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

console.log("Starting to read file");

fs.readFile("./test.txt", function(error, data){ console.log
("File Contents: " + data); });

console.log("End of reading file");
```

Reading Files (Sync)

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

console.log("Starting to read file");

var data = fs.readFileSync("./test.txt");

console.log("File Contents: " + data);

console.log("End of reading file");
```

TCP Server in Node.js

```
var net = require("net");
var clients = [];
var server = net.createServer(function (socket) {
  socket.write("Echo server\r\n");
  clients.push(socket);
  clients.forEach(function(client){
    client.write(socket.remoteAddress + " joined\n");
  });
});
server.listen(8000, "127.0.0.1");
console.log("TCP Server started at 127.0.0.1:8000");
```

But I’m a web developer

How do I use Node.js to create web applications?

We've got you covered

```
var http = require("http");
var server = http.createServer(function (req, res) {
  res.writeHead(200, {"Content-Type": "text/plain"});
  res.write("Hello World\n");
  res.end();
});
server.listen(8000, "127.0.0.1");
console.log("Server running at http://127.0.0.1:8000/");
```

Can I serve HTML files?

```
var http = require("http");
var fs = require("fs");
var server = http.createServer(function (req, res) {
  console.log("Received request: " + req.url);
  fs.readFile("./public" + req.url, function(err, data){
    if(err){
      res.writeHead(404, {"Content-Type": "text/plain"});
      res.end("Sorry. The requested resource was not found");
    }
    else{
      res.writeHead(200, {"Content-Type": "text/html"});
      res.end(data);
    }
  });
});
server.listen(8000, "127.0.0.1");
console.log("Server running at http://127.0.0.1:8000/");
```

What about Real-time and Event-driven?

Introducing Socket.IO - Socket.IO is a JavaScript library for realtime web applications. It has two parts: a client-side library that runs in the browser, and a server-side library for node.js. Both components have a nearly identical API. Like node.js, it is event-driven.

What about dynamic content?

Template Engines like:

- Embedded JS - ejs
- Jade

Web Frameworks like:

- Express - express.js
- restify - building REST APIs for nodejs

Database Engines like:

- mssql - For Microsoft SQL Server
- mongoose - For MongoDB
- oracle - For Oracle
- mysql - For MySQL

How do I install those packages?

NPM is the abbreviation of the Node Package Manager

The NPM registry is a repository of node packages that anyone can publish to or access to install node packages for your projects to install a package use:

```
npm install oracle
```

To install something globally accessible on the machine, use the -g flag:

```
npm install -g express-generator
```

Who’s using Node.js?

- MySpace
- Uber
- LinkedIn
- Microsoft
- Yahoo!
- Klout
- DuckDuckGo
- Voxer
- eBay
- PayPal
- WalMart
- Dow Jones
- The New York Times
- Cloud9 IDE
- Pearson

But I wanna know more

For our next session

Details on:

- One of the template engine
- Coupled with one web framework
- Dynamic Data using one database engine

Thank you