ICON UK 2015
node.js for Domino developers

Presenter: Matt White
Company: LDC Via

Agenda

- What is node.js?
- Why am I interested?
- Getting started
- NPM
- Express
- Domino Integration
- Deployment

A note about fonts

- In this session I'll be talking about writing code and also issuing terminal commands.
- To differentiate...

`Courier is code`

Lucida is a console command

Who am I?

- Domino web developer since 1996
  - IdeaJam
- XPages developer since 2008
  - XPages101.net
- node.js developer since 2013
  - Run a platform built on node.js to migrate or extend your Domino data to MongoDB

LDC VIA

What is node.js?

- Very simply it is an open source, server-side JavaScript engine for doing "stuff"

- Most commonly it's used to run scalable network applications (e.g. web apps)

- But it's equally happy acting as a command line utility

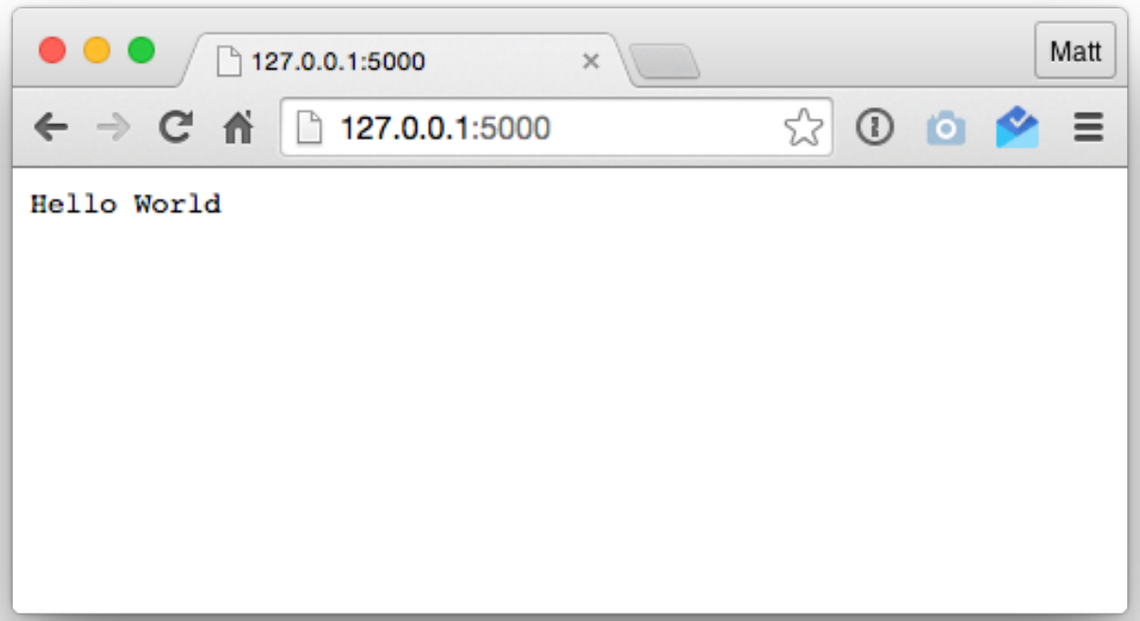
What is node.js?

- The JavaScript engine behind node.js is called ‘V8’ and it’s the same as used by the Chrome browser
  - So node.js is Server Side JavaScript, where have we heard that before?
- Runs as a program on your server or on various cloud services
- It’s open source with all that brings like the io.js vs node.js split
- It is still officially beta software, but it’s in use heavily around the world
  - Currently version 4.0.0
- There is a vast amount of material for you to leverage

What is node.js?

- At its simplest a web server can be created with a single JavaScript file like this:

```
var http = require('http');
http.createServer(function(req, res){
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World');
}).listen(5000, '127.0.0.1');
```



Why am I interested?

- It's relatively simple to transition into from Domino
  - We already know JavaScript
  - One language covers server side and client side
  - Communicating with REST services is a common approach
- It's easy to develop and deploy applications
  - Everything runs in a single asynchronous thread
- Performance and scalability
  - Can handle thousands of concurrent connections with low overhead
  - Great for horizontal scaling as an application grows

Why should I be wary?

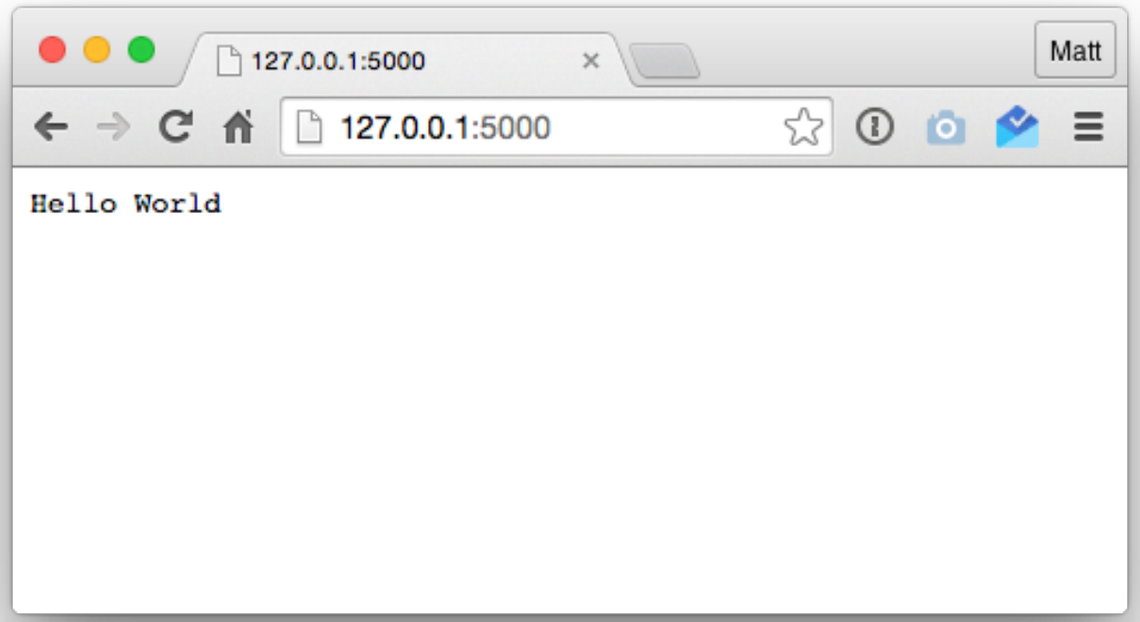
- It’s still beta software and things are changing
  - dependency hell is the new DLL hell
- A lot of choices are down to you
  - Someone has probably already solved your problem, but who?
  - Packages can become unmaintained
- All code is written using callbacks.
  - These can quickly become unwieldy to maintain if you don’t plan your code properly
  - It's worth learning about promises to mitigate this

Getting started

- Download and install from https://nodejs.org
  - Runs on Windows, OS X and Linux
- Get a good text editor
  - Sublime Text
  - Atom
- Create a project
  - As simple as a single JavaScript file
- Start “server”
  - `node app.js`

Getting started

- Demo



Getting started - NPM

- Although you can write everything yourself, you don't need to
- Node Package Manager (NPM) allows you to add modules to your application
  - It is installed as part of node.js
- NPM packages do pretty much everything. Including (but not limited to):
  - database connections
  - PDF generation
  - authentication
  - credit card handling
- Most importantly there are entire application frameworks to give you a jumpstart

Getting started - NPM

- You need to create a file called package.json
  - https://docs.npmjs.com/files/package.json

```
{   "name": "get-data",
   "version": "0.0.1",
   "private": true,
   "dependencies": {
     "some-npm-package": "1.0.0"
   }
}
```

Getting started - NPM

- Libraries or packages are added to your application by typing in the command line:

```
npm install some-npm-package -save
```

- Then you can add the package to your code by “requiring” it

```
var somepackage = require('./some-npm-package');
```

Getting started - NPM

- So if we wanted to access remote REST services such as the Domino Access Services

```
npm install restler --save
```

- Then in our app we add the JavaScript to our application

```
var rest = require('restler');
rest.get('http://myurl.com/something')
.on('complete', function(data, response){
  console.log(data);});
```

Getting started - NPM

- And that's it, this code will go and attempt to load JSON data from the URL supplied and then we can do something with it

- You can absolutely write your own code to do this if you want

Getting started - NPM

- Demo

get-data — bash — 80×22

```
Matts-MacBook-Pro:get-data mattwhite$ npm start

> get-data@0.0.0 start /Users/mattwhite/Documents/Github/node-for-domino-develop
ers/get-data
> node app.js

[ { '@title': 'All Documents',
    '@folder': false,
    '@private': false,
    '@modified': '2014-07-18T07:26:44Z',
    '@unid': '8178B1C14B1E9B6B8525624F0062FE9F',
    '@href': 'http://dev.londc.com:80/demos/discussion.nsf/api/data/collections/
unid/8178B1C14B1E9B6B8525624F0062FE9F' },
  { '@title': 'Author Profiles',
    '@folder': false,
    '@private': false,
    '@modified': '2014-07-18T06:51:57Z',
    '@unid': 'D079D13757CA23338525659800456CFB',
    '@href': 'http://dev.londc.com:80/demos/discussion.nsf/api/data/collections/
unid/D079D13757CA23338525659800456CFB' },
  { '@title': 'By Alternate Name',
    '@folder': false,
```

Getting started - Express

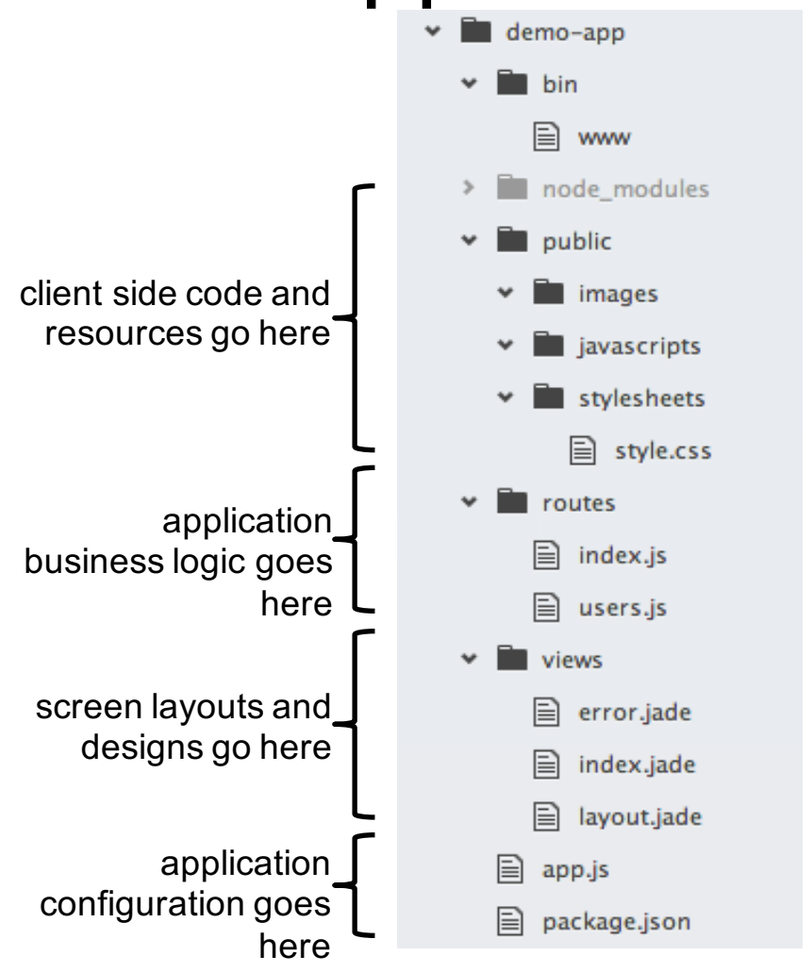
- One of the most commonly used packages is Express
  - http://expressjs.com
- This is an entire application framework that
  - handles page routing
  - organizes your code
  - generates HTML

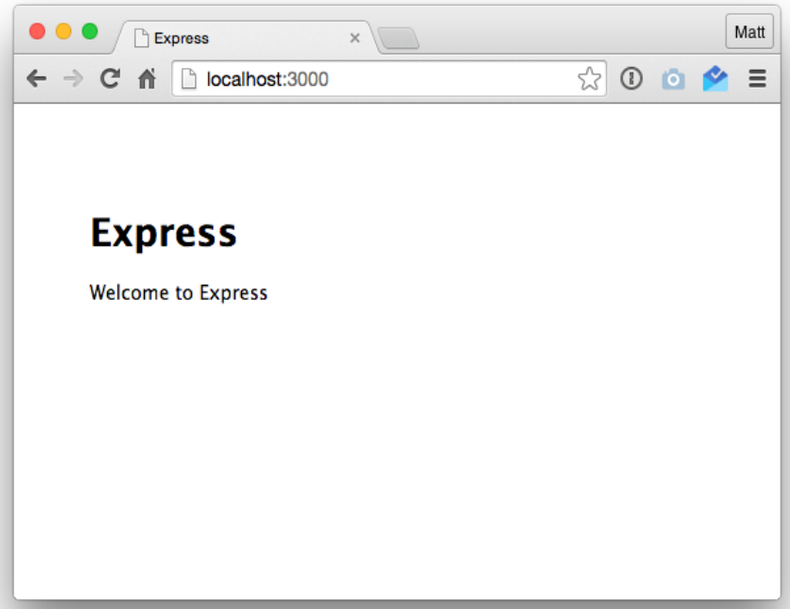
Getting started - Express

```
npm install express –save
npm install express-generator –g
express demo-app
cd demo-app
npm install
DEBUG=demo-app:* npm start
```

Getting started - Express

- Those commands will generate and launch a project for us
- We can build our application from this starting point





Domino Integration

- Using Restler we can easily read JSON data from a Domino server and return it to the browser, formatted as HTML

```
var express = require('express');
var router = express.Router();
var rest = require('restler');


/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

/* GET Domino View of Data */
router.get('/domino', function(req, res, next) {
rest.get('http://dev.londc.com/demos/discussion.nsf/api/data/collections/unid/81
78B1C14B1E9B6B8525624F0062FE9F')
  .on('complete', function(data, response){
    res.render('domino', {title: 'Domino Data', data: data});
  });
})

module.exports = router;
```

Domino Integration

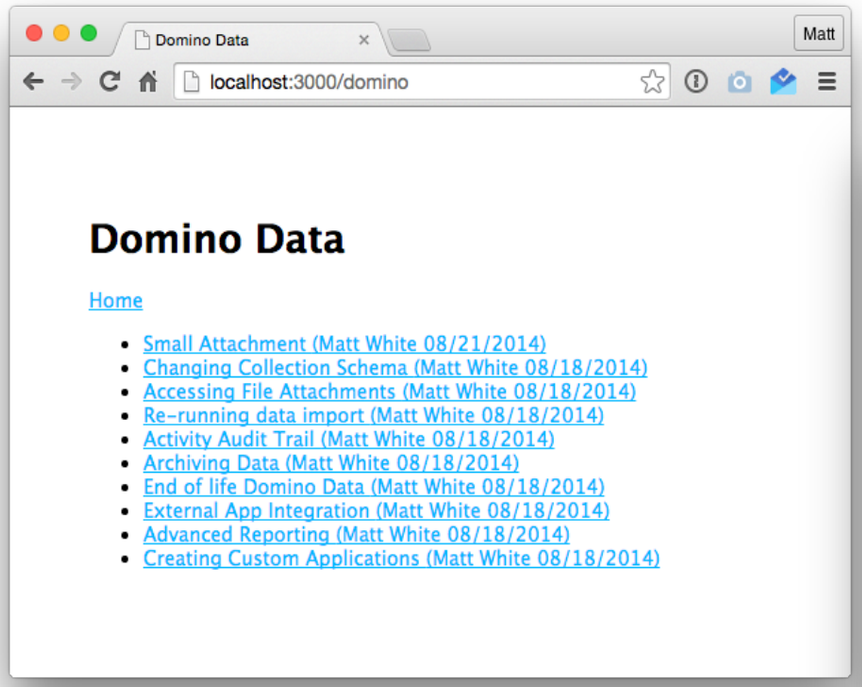
- To display the data we can use Jade HTML

```
extends layout

block content

  h1= title

  div

    a(href="/") Home

  ul

    each doc in data

      li

        a(href="/domino/" + doc['@unid']) #{doc['$117']}
```

Domino Integration

- Demo

Get a view of data



Get an individual document

Deployment

- It's worth investigating build or workflow tools
  - Grunt (is what I use primarily)
  - Gulp (an alternative)
    - They do things like compile JavaScript, CSS and can take many other boring tasks off your hands
- As with development, deployment is pretty simple
- There are two choices
  - Build your own server
  - Use a node.js cloud service

Deployment – On Premises

- We are used to deploying apps onto Domino servers
- We can take a similar approach with node.js
- Simply build a new server (Windows or Linux)
- Install node.js
- Copy across application files
- Depending on popularity of application you may want to investigate load balancing options

Deployment - Cloud

- Several options to investigate, including…
  - Heroku
  - Bluemix
  - AWS
- Easiest way to deploy is from a Git repository (e.g. Github)
- Usually there are free options for development environments so you can show others what you're doing

MEAN Stack

- We've talked so far about pure node.js and node.js with Express
- The MEAN stack is the generally accepted default way of developing apps
    - M - mongoDB
    - E - Express
    - A - AngularJS
    - N - node.js
- You can build a new MEAN app using the command line
- You do not have to use all elements
    - For example, I currently use M E N but only sometimes use A

Useful packages

- async
  - Helps you work with lists of data that you need to perform asynchronous operations on
  - Client and server side support
- cron
  - Allows you to set tasks to run on any schedule (e.g. every hour, day, week etc)
- mocha
  - A unit testing framework for node.js applications

Useful packages

- moment
  - Great utility for working with dates and times
  - Client and server side support
- mongodb / mongoose
  - If you are using MongoDB as a back-end database, these drivers help you connect to and manage your data
- nodemon
  - A way of launching your app so that when you change code it automatically relaunches

Useful packages

- passport
  - The de facto standard for handling application authentication
  - Works with Facebook, LinkedIn, Google, OAuth
  - More than 300 authentication strategies in total
- pdfkit
  - For generating PDF files

Other resources

- https://nodejs.org/
- http://mean.io/
- https://www.npmjs.com/
- http://expressjs.com/
- http://passportjs.org/
- https://github.com/caolan/async
- https://github.com/ncb000gt/node-cron
- http://momentjs.com/
- http://mongoosejs.com/
- http://nodemon.io/
- http://www-10.lotus.com/ldd/ddwiki.nsf/xpAPIViewer.xsp?lookupName=IBM+Domino+Access+Services+9.0.1#action=openDocument&res_title=Viewfolder_collection_GET_dds10&content=apicontent

Contact me:

- Contact me:
  - @mattwhite
  - matt@ldcvia.com
  - Download slides at: http://mattwhite.me/presentations
  - Sample code at: https://github.com/LonDC/node-for-domino-developers
  - For more about LDC Via come and see me at our stand
  - Discount code for LDC Via: "ICONUK2015"