node.js for Domino developers

Matt White | LDC Via

www.ldcvia.com

About Me

- Domino web developer since the 1990s
  - Lotus Award winner with IdeaJam
- Java developer
- XPages developer since before it was released!
- Co-founder of LDC Via
  - http://ldcvia.com
  - node.js
  - MongoDB

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

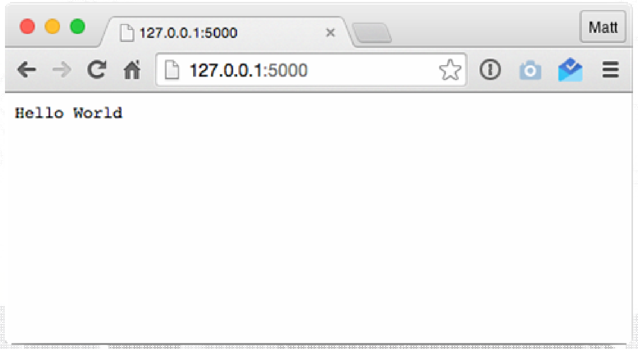
What is node.js?

- 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
- Currently version 4.4.2 (as at Apr 11th 2016)
- 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 line of JavaScript 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');
```



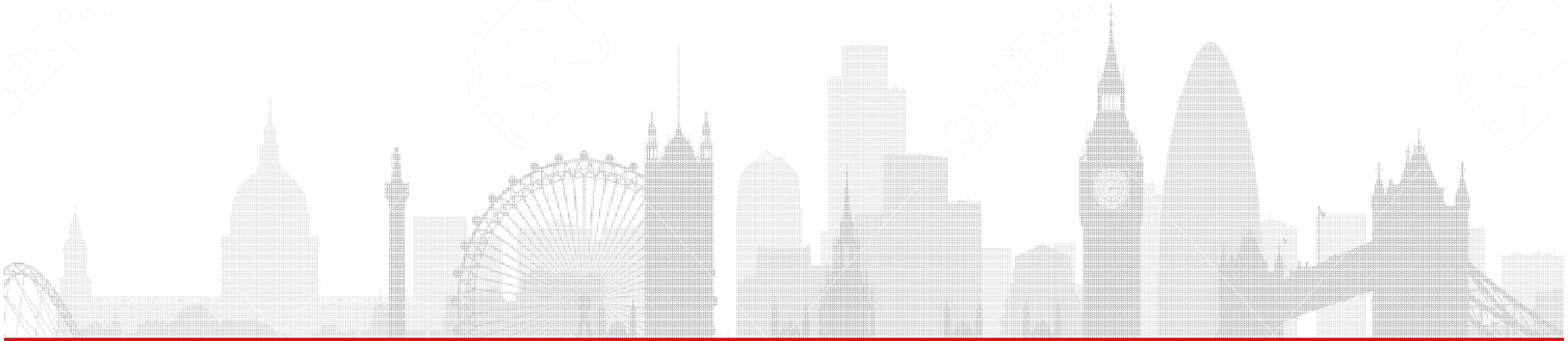
```
Hello World
```

- Then we launch the "server"

```
node hello-world.js
```

Hello World Demo

- Demo



www.ldcvia.com

Why am I interested?

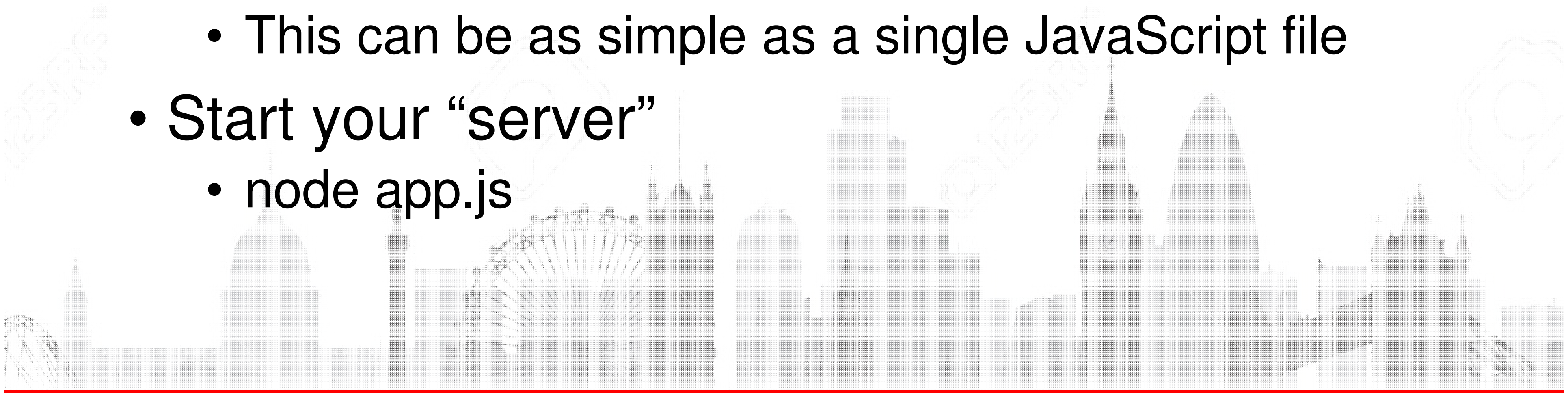
- My clients started to ask about node.js
  - Domino applications weren’t scaling well
  - Difficult to find people with Domino development skills
- It’s very quick to pick up
  - Simple JavaScript coding
- There are huge amounts of resources available for you
  - You spend less time tracking down obscure problems
  - You spend more time working on the actual applications
- Very easy to deploy and manage
- Scales for large applications very easily

Why should I be wary?

- A lot of choices are down to you
  - Someone has probably already solved your problem, but who?
  - Packages can become unmaintained
  - Package dependency is the new DLL hell
- There is no enforced structure for your code
  - Files and code can quickly become unwieldy to maintain if you don't plan properly
  - There are frameworks and patterns to help you

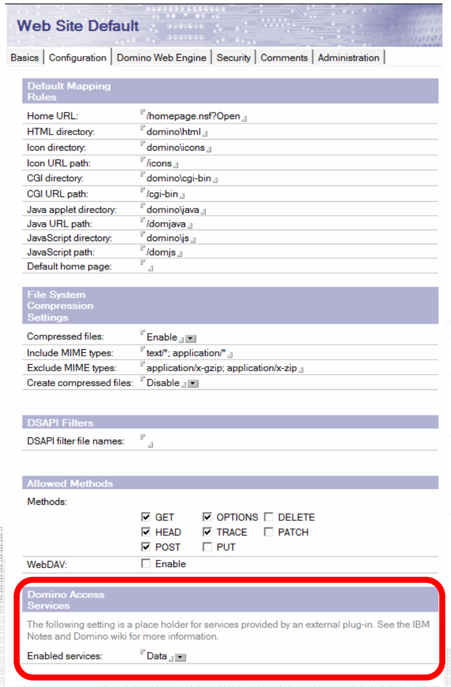
Getting started

- Configure Domino Data Service
- Download and install from https://nodejs.org
  - Runs on Windows, OS X and Linux
- Create a project
  - This can be as simple as a single JavaScript file
- Start your “server”
  - node app.js



Domino Data Service

- Also known as Domino Access Services
- Enables REST services for Domino data without having to code anything
- Has to be enabled for Server, database and views
- First we have to enable Domino Access Services on the Domino server
  - https://www-10.lotus.com/ldd/ddwiki.nsf/xpAPIViewer.xsp?lookupName=IBM+Domino+Access+Services+9.0.1#action=openDocument&res_title=IBM_Domino_Access_Services_overview_das901&content=apicontent



Web Site Default

Basics | Configuration | Domino Web Engine | Security | Comments | Administration

Default Mapping Rules

Home URL:	/homepage.nsf?Open
HTML directory:	domino\html
Icon directory:	domino\icons
Icon URL path:	/icons
CGI directory:	domino\cgi-bin
CGI URL path:	/cgi-bin
Java applet directory:	domino\java
Java URL path:	/domjava
JavaScript directory:	domino\js
JavaScript path:	/domjs
Default home page:	

File System Compression Settings

Compressed files:	Enable
Include MIME types:	text/*; application/*
Exclude MIME types:	application/x-gzip; application/x-zip
Create compressed files:	Disable

DSAPI Filters

DSAPI filter file names:	

Allowed Methods

Methods: ✓ GET ✓ OPTIONS ☐ DELETE ✓ HEAD ✓ TRACE ☐ PATCH ✓ POST ☐ PUT

WebDAV: ☐ Enable

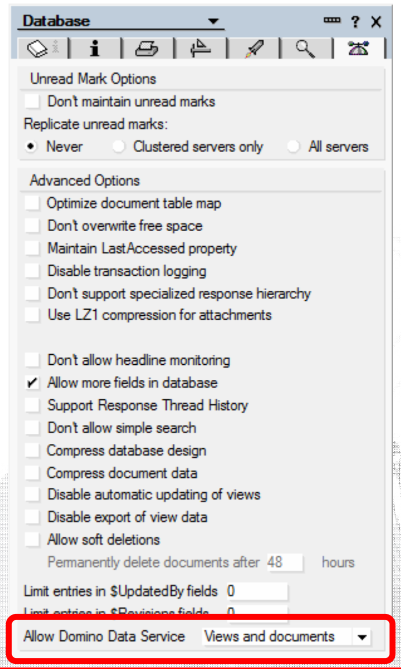
Domino Access Services

The following setting is a place holder for services provided by an external plug-in. See the IBM Notes and Domino wiki for more information.

Enabled services: Data

Domino Data Service

- Now for each database and view we want to access we need to enable DDS



Unread Mark Options

- [ ] Don't maintain unread marks

Replicate unread marks:
[✓ Never] Clustered servers only All servers

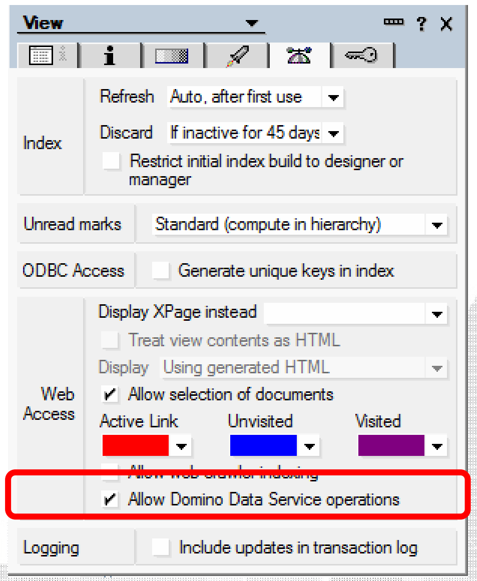
Advanced Options

- [ ] Optimize document table map
- [ ] Don't overwrite free space
- [ ] Maintain LastAccessed property
- [ ] Disable transaction logging
- [ ] Don't support specialized response hierarchy
- [ ] Use LZ1 compression for attachments
- [ ] Don't allow headline monitoring
- [x] Allow more fields in database
- [ ] Support Response Thread History
- [ ] Don't allow simple search
- [ ] Compress database design
- [ ] Compress document data
- [ ] Disable automatic updating of views
- [ ] Disable export of view data
- [ ] Allow soft deletions

Permanently delete documents after 48 hours

Limit entries in $UpdatedBy fields 0
Limit entries in $Revisions fields 0

Allow Domino Data Service: Views and documents



Index	Refresh: Auto, after first use Discard: If inactive for 45 days [ ] Restrict initial index build to designer or manager
Unread marks	Standard (compute in hierarchy)
ODBC Access	[ ] Generate unique keys in index
Web Access	Display XPage instead [ ] Treat view contents as HTML Display: Using generated HTML [x] Allow selection of documents Active Link 🟥 Unvisited 🟦 Visited 🟪 [ ] Allow web crawler indexing [x] Allow Domino Data Service operations
Logging	[ ] Include updates in transaction log

Node Package Manager

- 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
- Most importantly there are entire application frameworks to give you a jumpstart

Node Package Manager

- Find the package you want
  - http://npmjs.com
- 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');
```

Node Package Manager

- So, for example, if we wanted to access remote REST services such as the Domino Access Services we'd look for a package
  - https://www.npmjs.com/package/restler
- Once found, we can install it in our app

```
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);
    }
);
```

Node Package Manager

- Demo

package.json

- Describes your node project and what dependencies it has
- Key information:
  - name
  - version
  - license
  - dependencies
- https://docs.npmjs.com/files/package.json

ExpressJS

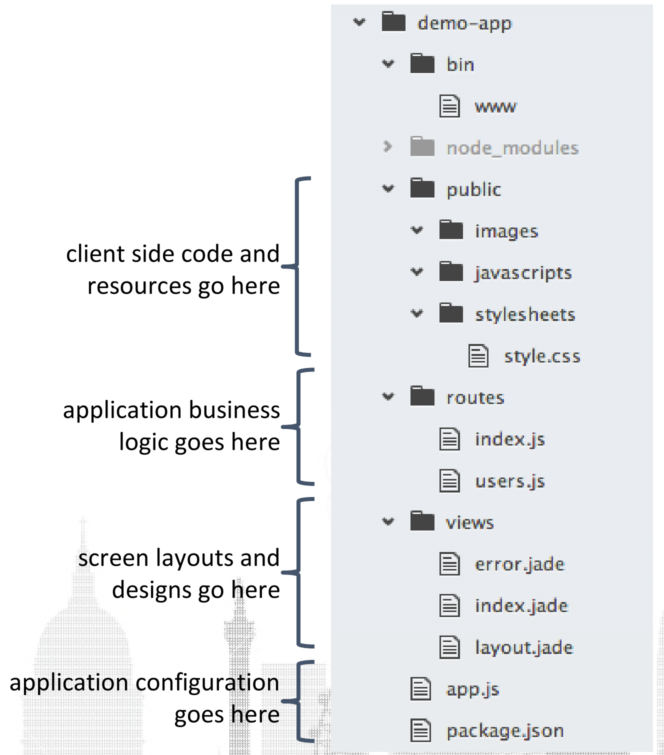
- One of the most commonly used packages is Express
  - http://expressjs.com
- This is an entire application framework that
  - handles page routing
  - organizes your code
  - generates HTML
- Used with Restler we can read JSON data from a Domino server and return it to the browser, formatted as HTML

ExpressJS

- To create a new project…

```
npm install express --save
npm install express-generator --g
express demo-app
cd demo-app
npm install
npm start
```

ExpressJS



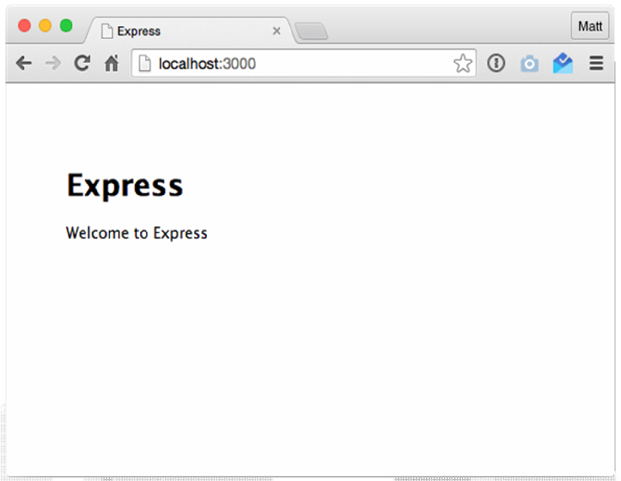
client side code and resources go here

application business logic goes here

screen layouts and designs go here

application configuration goes here

```
demo-app
  bin
    www
  node_modules
  public
    images
    javascripts
    stylesheets
      style.css
  routes
    index.js
    users.js
  views
    error.jade
    index.jade
    layout.jade
  app.js
  package.json
```



Express

Welcome to Express

ExpressJS

- Demo

Jade HTML

- The way that ExpressJS renders HTML is to use the Jade mark-up framework
  - http://jade-lang.com/
- Similar to XPages mark-up
  - You can conditionally show content
  - You can loop (or repeat) content
- Not tied to any front end framework
- Can be a little painful to get started with, this is a good article
  - http://webapplog.com/jade/

Jade HTML

- In our example app we can display the data from Domino like this

```
extends layout

block content
  h1= title
  div
    a(href="/") Home

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

Jade HTML

- Demo

Bower

- For further client side enhancement you can use Bower
  - http://bower.io

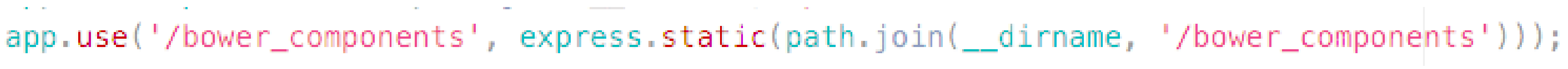
```
npm install --g bower
npm install bower --save
```

- Effectively the browser version of NPM
- We can, for example, add Bootstrap and jQuery

```
bower install jquery --save
bower install bootstrap --save
```

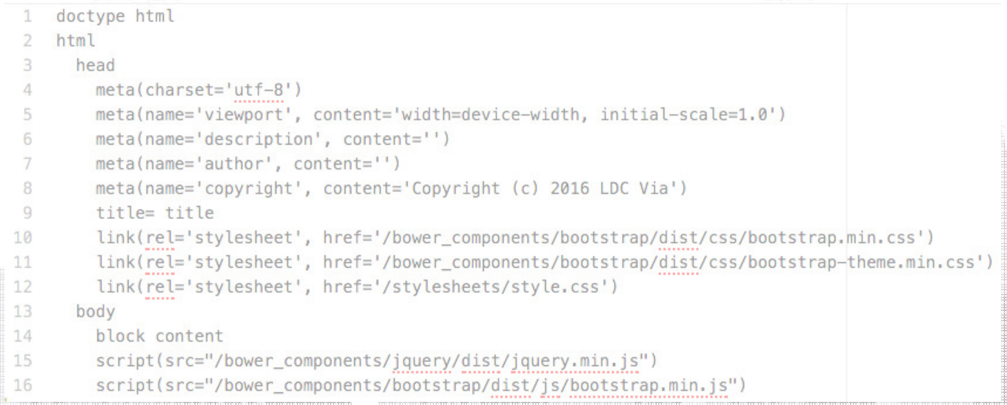
Bower

- Now we need to make those files publicy available by adding a route in app.js



```
app.use('/bower_components', express.static(path.join(__dirname, '/bower_components')));
```

- Then include the required files in our Jade template



```
doctype html
html
  head
    meta(charset='utf-8')
    meta(name='viewport', content='width=device-width, initial-scale=1.0')
    meta(name='description', content='')
    meta(name='author', content='')
    meta(name='copyright', content='Copyright (c) 2016 LDC Via')
    title= title
    link(rel='stylesheet', href='/bower_components/bootstrap/dist/css/bootstrap.min.css')
    link(rel='stylesheet', href='/bower_components/bootstrap/dist/css/bootstrap-theme.min.css')
    link(rel='stylesheet', href='/stylesheets/style.css')
  body
    block content
    script(src="/bower_components/jquery/dist/jquery.min.js")
    script(src="/bower_components/bootstrap/dist/js/bootstrap.min.js")
```

Authentication

- So far we've been operating against a publicly available database
  - Not very realistic

- We need to have the user authenticate with the Domino server
  - Create a login screen
  - Send the username and password to Domino
  - Return the session cookie to the browser
  - Include the session cookie with all future requests

Authentication

- On every page that requires authentication, add a check

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

- In the simplest case that there is a session cookie

```
exports.requiresLogin = function (req, res, next) {
  if (req.cookies['DomAuthSessId'] != null) return next()
  res.redirect('/login')
}
```

Authentication

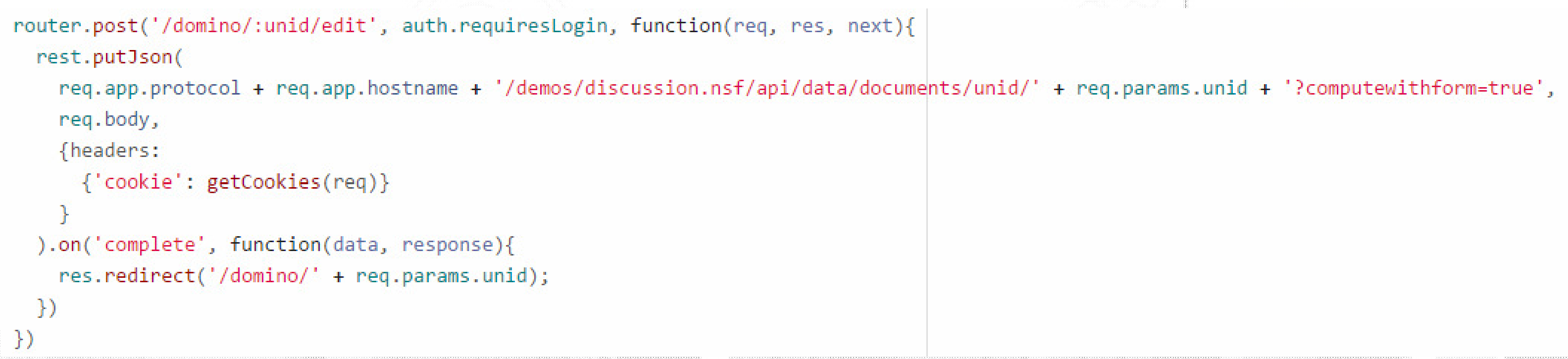
- Demo

Authentication Alternatives

- If you need to authenticate with an alternative source such as Google, Twitter, Facebook, Oauth, SAML…
- Passport is an NPM package that handles authentication
  - http://passportjs.org/
- 300 different strategies for different needs
- Simple configuration

Modifying Data

- Create forms using Jade to capture the data
- Add routes to the application to submit the data to
- For each route send JSON data to the Domino server



```
router.post('/domino/:unid/edit', auth.requiresLogin, function(req, res, next){
  rest.putJson(
    req.app.protocol + req.app.hostname + '/demos/discussion.nsf/api/data/documents/unid/' + req.params.unid + '?computewithform=true',
    req.body,
    {headers:
      {'cookie': getCookies(req)}
    }
  ).on('complete', function(data, response){
    res.redirect('/domino/' + req.params.unid);
  })
})
```

Modifying Data

- Demo

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

Deployment – Cloud

- Demo – Heroku & Bluemix

www.ldcvia.com

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
  - Equivalent of Agent Manager
  - 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
  - Can be integrated with grunt

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

Useful Links

- 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/LDCVia/node-for-domino-developers