CS454 Node.js & Angular.js

Cydney Auman
CSULA

Combining Node & Angular - Week 5

NPM Basics

```
npm init
```
// assist in creating the package.json.

```
npm install
```
// installs all modules in the package.json - installed in the node_modules directory.

```
npm install --save <module-name>
```
// installed the module by name and auto-magically adds it to package.json.

```
npm install -g <module-name>
```
// installs the module globally. allows you to use module outside of node project

Node HTTP Server

The built-in node module responsible for running HTTP servers and making HTTP requests is called "`http`".

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

var server = http.createServer(function(request, response) {
  response.writeHead(200, { "Content-Type": "text/plain" });
  response.end("Hello World");
});



server.listen(8000, "localhost");
console.log("Server running at http://localhost:8000/");
```

Request

Whenever we make a request to the server, the request handler function is called. request is a request that comes from the client. Sometime this shortened to req.

If you start you app by going to the command line and doing node server.js

Visit localhost:8000 and you'll see what URL you are requesting. It is a GET request, and that you've sent a some headers.

Response

The response is the next argument in the function. Just like the prior argument is often shortened to res.

With each response, you get the data ready to send, and then you call `response.end()`. Eventually, you ***must*** call this method. This method does the actual sending of data. If this method is not called, the server just hangs forever.

Node with Express

Express.js describes itself as a "a minimal and flexible Node.js web application framework, providing a robust set of features for building single and multi-page, and hybrid web applications."

In short, it's a framework for building web applications with Node.js.

Node with Express

```
var express        = require('express');
var bodyParser     = require('body-parser');
var morgan         = require('morgan');

var app            = express();

app.use(express.static(__dirname + '/public'));
app.use(morgan('dev')); // allow logging
app.use(bodyParser.json()) // parse application/json

app.get("/", function(request, response) {
  response.writeHead(200, { "Content-Type": "text/plain" });
  response.end("Hello World!");
});
app.listen(8000);
```

Middleware

Middleware is a function with access to the request object (req) and the response object (res) in express.

Application level middleware are bound to an instance of express typically using - app.use().

Handling Views with Express

```
var express = require("express");
var app = express();

// set the view directory to /views
app.set("views", __dirname + "/views");

// use the Jade templating language
app.set("view engine", "jade");
```

Setup where our view files are located. Then we set up which template language to use. In this case - Jade, which is a templating language.

Jade

Jade is a high performance template engine. It is a clean, ***whitespace sensitive*** syntax for writing html.

```
doctype html
html(lang="en")
  head
    title= pageTitle
    script(type='text/javascript', src='angular.js')
  body
    h1 Jade - node template engine
    #container.col
      if youAreUsingJade
        p You are amazing
      else
        p Get on it!
      p.
        Jade is a terse and simple templating language with a
        strong focus on performance and powerful features.
```

Angular Dependency Injection

Dependency Injection is a software design pattern in which an object is given its dependencies, rather than the object creating them itself.

It is about removing the hard-coded dependencies and making it possible to change them whenever needed.