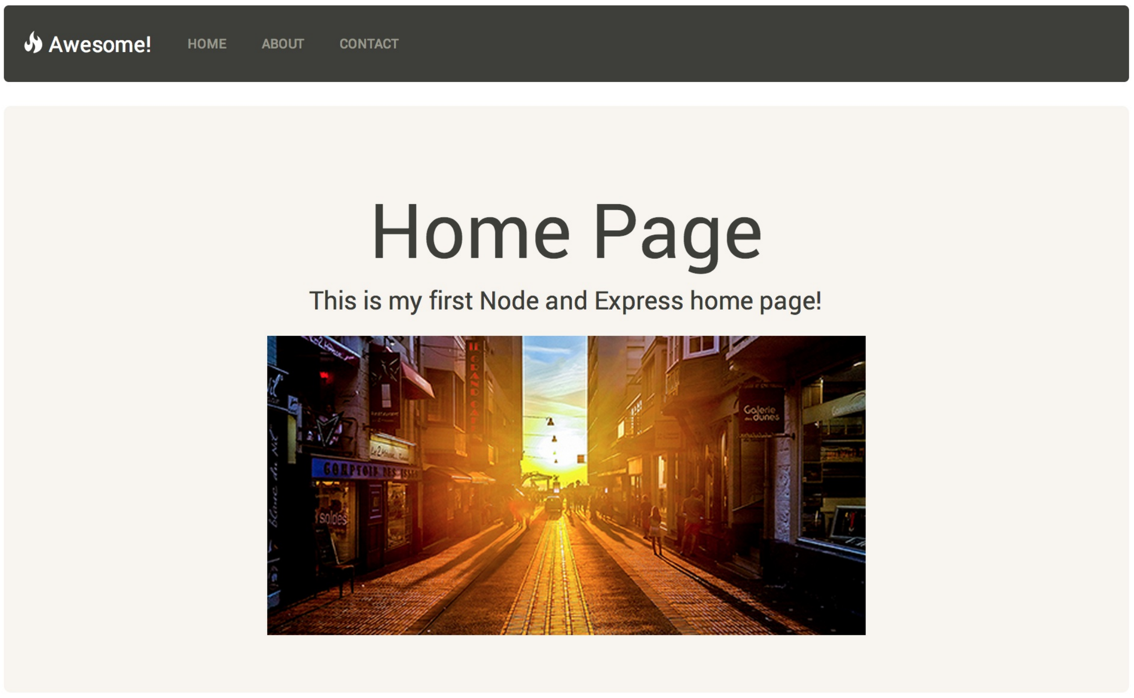
Building a Site with Node and Express

Node.js (http://nodejs.org/) has grown increasingly in popularity over the past couple years. With its adoption by large companies like Microsoft, Yahoo, PayPal, eBay, and many more, now is a great time to jump into Node development.

What We'll Be Building

In this booklet, we'll be looking at how to create a simple 3-page website using Node and its most popular framework, ExpressJS (http://expressjs.com/).

Here's a picture of the site we'll be creating. Nothing fancy from a design perspective. Our main focus will be on the Node and Express side of things and we'll just use Twitter Bootstrap (http://getbootstrap.com/) for quick styling.



Home Page

This is my first Node and Express home page!

By building a sample site using Node and Express, we will learn many things including:

- Node concepts, best practices, and getting started
- Express concepts, best practices, and getting started
- Routing applications with Express
- How to use EJS (http://embeddedjs.com/) (a JavaScript templating engine) to template views
- How to pass data and variables from server to HTML

We'll have 3 pages with 2 different layout types:

- Full Width Page (**Home** and **Contact**)
- Page with Sidebar (**About**)

By using a full page and a sidebar layout, we'll be able to see how we can template our views. This will benefit us because we don't have to rewrite our view files over and over. DRY (http://en.wikipedia.org/wiki/Don't_repeat_yourself) is the way to go!

Now that we know what we are building, lets get started with the fun stuff, the actual programming!

Starting our Application

Let's start out by looking at the file structure for our application. This is a good way to get a top-down view and now what files we will need. Here are the files we have:

- public (folder that will hold css/js/images)
- views (will have our view files)
    - partials (the repeatable things for our site (head, header, footer))
    - pages (the main pages for our site (home, about, contact))
- package.json (where we start our Node/Express application)
- server.js (where we configure Express and define site routes)

When starting a Node application, we will always start with the `package.json` file. This is where we define the main parts of our application like its name, version, author, license, and dependencies.

Let's create our `package.json` file with the minimal attributes needed to start our application.

```
{
    "name": "node-express-site",
    "main": "server.js",
    "dependencies": {
        "express": "~4.8.5",
        "ejs": "~1.0.0"
    }
}
```

Shortcut for Creating a package.json File: If you want an easy way to create the `package.json` file, npm comes with a great starting command: `npm init`. Just type that and watch the magic as your package.json file is generated for you.

Shortcut for Adding Dependencies: When adding dependencies, you won't always know the version number of the packages that you want. npm comes with another shortcut for adding dependencies to your project. Just type `npm install <package name> --save`. npm will automatically add your package to the dependencies section with the latest version!

Installing Express and EJS

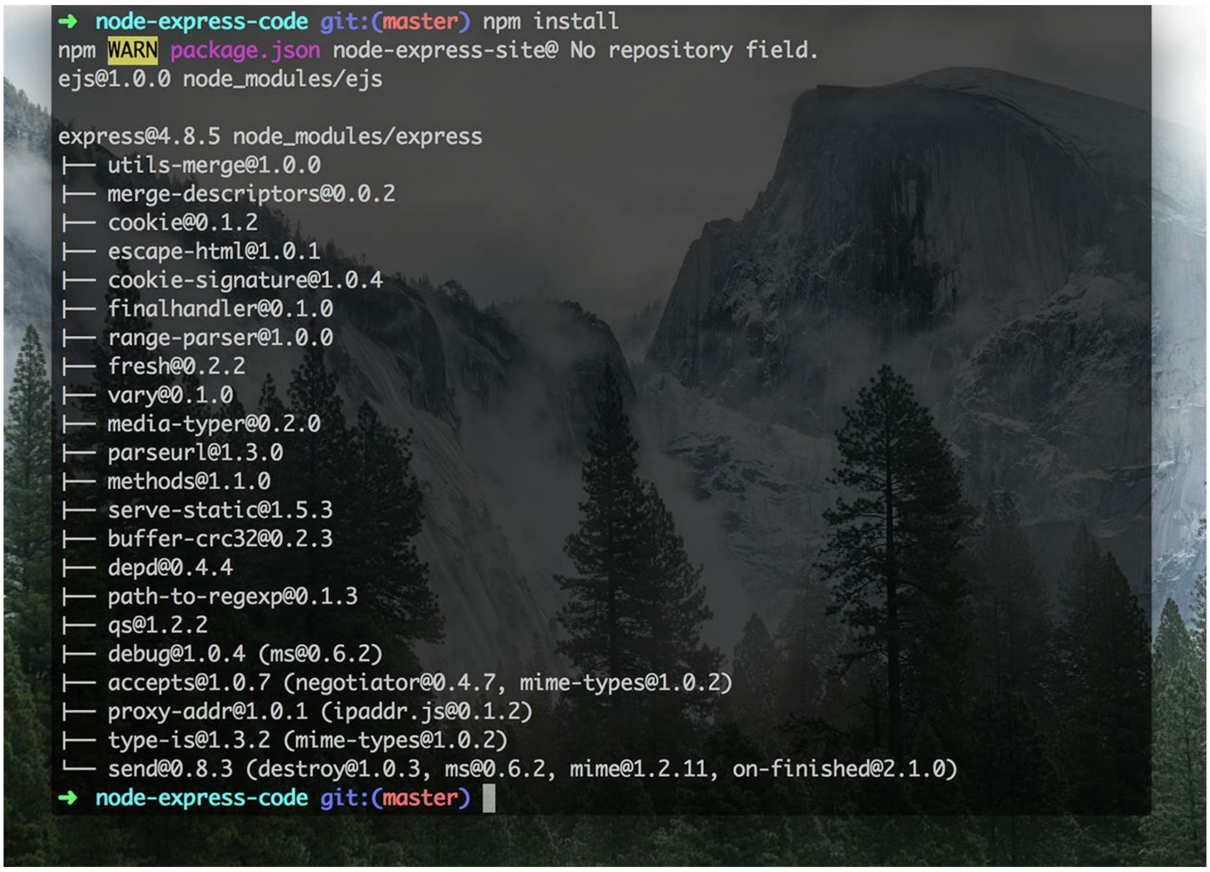
In the above `package.json` file, we have defined:

- **name**: The name of our application
- **main**: The file that we will use to start up our application.

- **dependencies**: The dependencies that we will need (Express and EJS).

By adding `express`, `ejs`, and the version we want to the dependencies, we can now bring in both of these packages by running:

`npm install`



```
➜ node-express-code git:(master) npm install
npm WARN package.json node-express-site@ No repository field.
ejs@1.0.0 node_modules/ejs

express@4.8.5 node_modules/express
├── utils-merge@1.0.0
├── merge-descriptors@0.0.2
├── cookie@0.1.2
├── escape-html@1.0.1
├── cookie-signature@1.0.4
├── finalhandler@0.1.0
├── range-parser@1.0.0
├── fresh@0.2.2
├── vary@0.1.0
├── media-typer@0.2.0
├── parseurl@1.3.0
├── methods@1.1.0
├── serve-static@1.5.3
├── buffer-crc32@0.2.3
├── depd@0.4.4
├── path-to-regexp@0.1.3
├── qs@1.2.2
├── debug@1.0.4 (ms@0.6.2)
├── accepts@1.0.7 (negotiator@0.4.7, mime-types@1.0.2)
├── proxy-addr@1.0.1 (ipaddr.js@0.1.2)
├── type-is@1.3.2 (mime-types@1.0.2)
└── send@0.8.3 (destroy@1.0.3, ms@0.6.2, mime@1.2.11, on-finished@2.1.0)
➜ node-express-code git:(master)
```

We can see npm bring in the express and the ejs packages and place them into the **node_modules** folder that gets created.

Now we have **defined our application** and **have the dependencies we need**. Let's start configuring our application using Node and Express in our `server.js` file.

Starting a Node and Express Server

We defined our main file earlier in our `package.json` file. We will be using a file called `server.js`. In this file, we will:

- Set up a Node server using Express
    - We will be able to visit our site in our browser at `http://localhost:8080`
- Configure our app to **use ejs** as the templating engine
- Set up our routes
- Start the server!

Let's start up our `server.js` file and break it down for each section. We'll start by calling Express.

```
// load the express and create our application
var express = require('express');
var app     = express();

// set the port based on environment
var port    = process.env.PORT || 8080;

// START THE SERVER ================
// ==================================
app.listen(port);
console.log(port + ' is the magic port!');
```

Using EJS as our View Engine

Setting Up Express Routes

Setting Up The Base Site and Styles

Templating Our View Files

Passing Data to Our Views

Conclusion and Further Reading