NodeJS coding basics

Rajeev gupta
Trainer & consultant

The simplest – a hello world server app

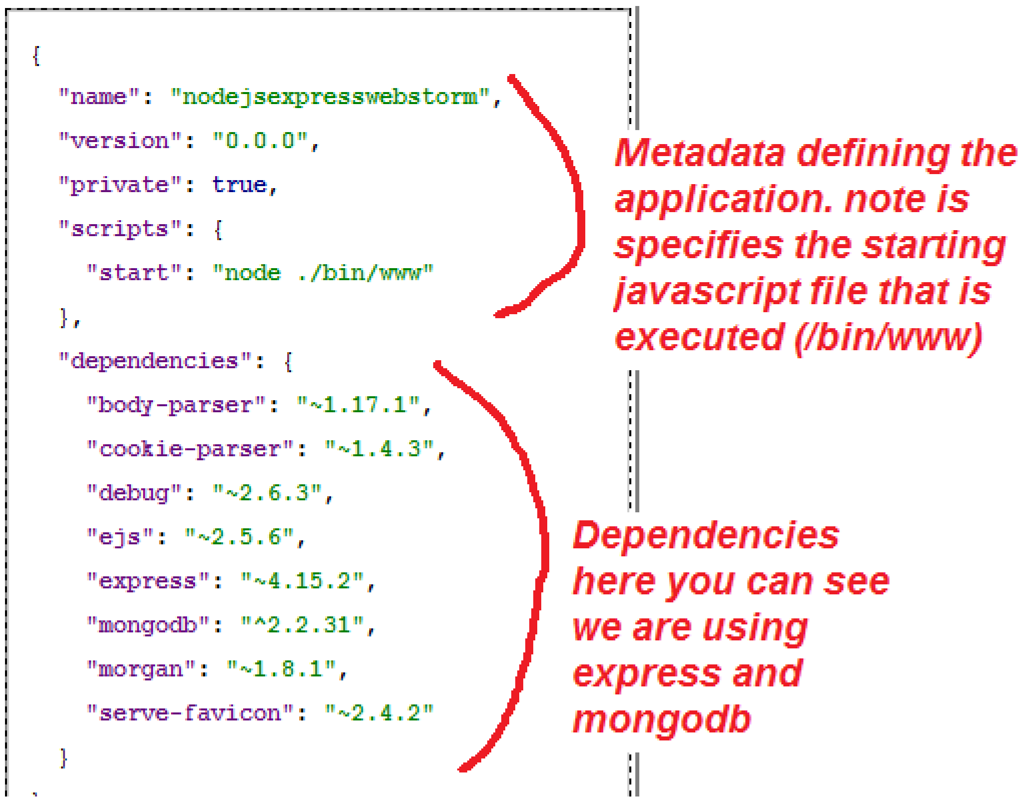
```
const http = require('http');   //requires the http module
const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
     res.statusCode = 200;          //building out the response
     res.setHeader('Content-Type', 'text/plain');
     res.end('Hello World\n');
       });

server.listen(port, hostname, () => {    //activate the server on port 3000
            console.log(`Server running at http://${hostname}:${port}/`);
       });
```

package.json

- Here is a simple file
- Like a ship’s manifest file



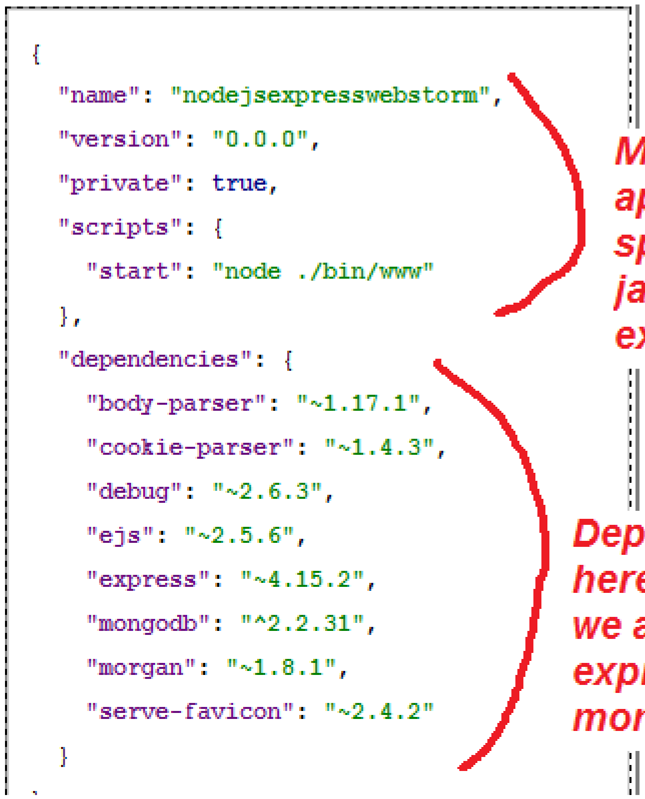
```
{
  "name": "nodejsexpresswebstorm",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.17.1",
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.3",
    "ejs": "~2.5.6",
    "express": "~4.15.2",
    "mongodb": "^2.2.31",
    "morgan": "~1.8.1",
    "serve-favicon": "~2.4.2"
  }
```

Metadata defining the application. note is specifies the starting javascript file that is executed (/bin/www)

Dependencies here you can see we are using express and mongodb

Modules in NodeJS

- Like packages in Java or libraries in C++
- Note the package.json refers to some standard NodeJS but, there are others



```
{
  "name": "nodejsexpresswebstorm",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.17.1",
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.3",
    "ejs": "~2.5.6",
    "express": "~4.15.2",
    "mongodb": "^2.2.31",
    "morgan": "~1.8.1",
    "serve-favicon": "~2.4.2"
  }
```

Metadata defining the application. note is specifies the starting javascript file that is executed (/bin/www)

Dependencies here you can see we are using express and mongodb

Express
EJS
MongoDB
Body-Parser
Cookie-Parser
Morgan
……..

installed in your project and referenced in your package.json file

- HOW TO INSTALL

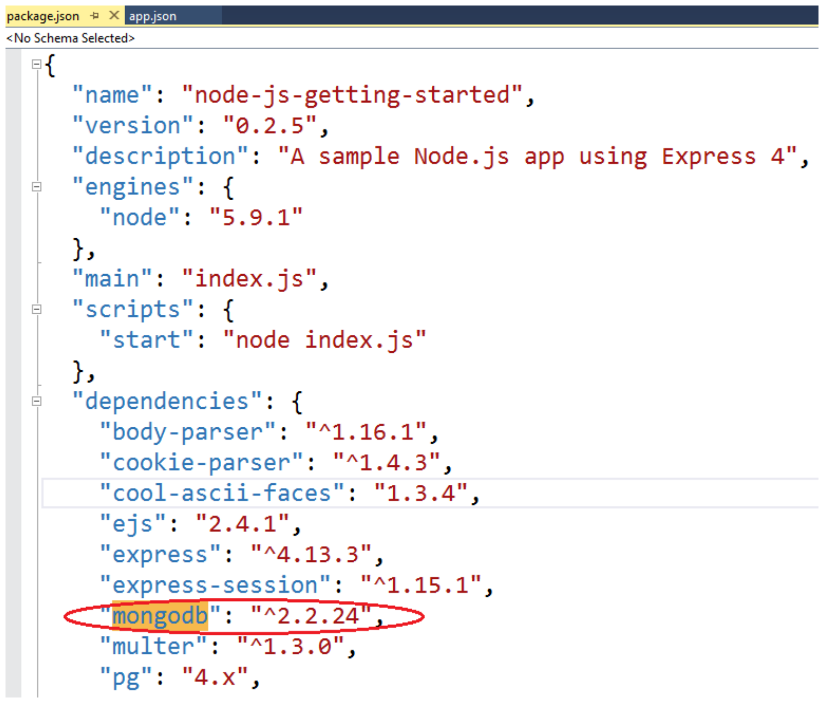
in terminal window in project type the following:

npm install <module name>

Example installing mongodb module

- npm install mongodb –save

Results in ALSO adding the dependency to the package.json



```
{
  "name": "node-js-getting-started",
  "version": "0.2.5",
  "description": "A sample Node.js app using Express 4",
  "engines": {
    "node": "5.9.1"
  },
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "dependencies": {
    "body-parser": "^1.16.1",
    "cookie-parser": "^1.4.3",
    "cool-ascii-faces": "1.3.4",
    "ejs": "2.4.1",
    "express": "^4.13.3",
    "express-session": "^1.15.1",
    "mongodb": "^2.2.24",
    "multer": "^1.3.0",
    "pg": "4.x",
```

module inside a NodeJS file □ use the require method

```
// main.js
 var greetings = require("./greetings.js");

// "Hello“
greetings.sayHelloInEnglish();

// "Hola“
greetings.sayHelloInSpanish();
```

Basically, you create an “object” variable
in this case called greetings that represents
the code in the loaded JS module (greetings.js)

Then to access a method you simply invoke
variableName.methodName(****)

and build responses in NodeJS

- You can use the basic http module we saw in our hello world repeated below

```
const http = require('http');   //requires the http
module
const hostname = '127.0.0.1';
const port = 3000;


const server = http.createServer((req, res) => {
        res.statusCode = 200;          //building out the
response
        res.setHeader('Content-Type', 'text/plain');
        res.end('Hello World\n');
         });


server.listen(port, hostname, () => {     //activate the
server on port 3000
               console.log(`Server running at http://$
{hostname}:${port}/`);
         });
```

-