**The following pages will be dedicated to the understanding and persistent pursuit of becoming a fluent server side programmer.**

**This will aid me towards achieving a MERN stack developer role.**

**Created by Jonathan Mohabir**

[**https://github.com/EthanRBrown/web-development-with-node-and-express-2e**](https://github.com/EthanRBrown/web-development-with-node-and-express-2e)

**companion repo for code understanding.**

**Chapter 01 introducing express:**

* The age of JavaScript is truly among us.
* Thanks to node, knowing this language allows a front end developer to incorporate some server-side capabilities of his application he will be creating.

* This is truly an exciting time to be apart of internet technology.
* JavaScript started in 1996, but a lot of its features that were pushing for greater use among programmers came during the year of 2005.
* From there… in 2009 we then see the release of NODE JS. Node js is a JavaScript runtime environment. Node package manager contains packages of things that can help us develop applications for the web.
* The internet today is estimated to be worth 2.1 trillion dollars….. you can get in on a piece of that pie… if you know how.

**LOOKING AT EXPRESS**

* Express is simply a web application framework.

* Express allows us to actually write server side code .
* Express simply **accepts HTTP requests from the client, and then returns an HTTP response.**
* Single page applications are applications used by users which can allow for functionality of rendering data from a server, and contain a front end for displaying HTML , CSS, media assets, and JavaScript DOM manipulation.

**Chapter: 02 – Getting started with Node.**

* Node is best started when its actually being coded… so in that case refer to Jonathan\_coding\_prac repo.

* Inside chapter 02 of the orielly\_book directory contains this chapters code.
* Comments also included within the code.
* To start a simple server with node, the http module is required to be imported. As we are writing http requests and responses.
* Require is a keyword within node which allows us to work with different modules.
* When starting a server: declare an http constant which allow for methods of this module to be used.
* Set a Port for the app to be started onto.
* Express does better handling for this so just understanding the basics from this node file.
* A switch case was included into the create server method to demonstrate routing capabilities of the node system.

END OF CHAPTER 02, CHAPTER 03 moves into express.

**SAVING TIME WITH EXPRESS:**

* **SAMPLE PROJECT DIRECTORY CREATED, this directory will contain the code for following the example for the meadowlark travel site the author provides.**

* The server has been created within the first few lines of the meadowlark.js file.
* **Express is imported through the use of the require keyword. The app is set the methods of express, and then the port is set to listen into specified port number. This allows us to access from the browser using localhost.**
* Because app is set to express methods… we can use all of those methods.
* App.METHOD… method is a placeholder for what we can do.
* **App.Get => are methods which create routes which will render different parts of a request made by a user. That specific page will load when specifying the .send method in the response.**
* **App.Use => methods which express adds middleware.**
* **VIEW – content which is returned to the user, for website development usually through HTML. Frameworks like pug allow for the writing of HTML but removes the need for marking up.**
* **HANLDEBARS – A less abstract way to create views.**
* **The Project directory is assigned: npm I express-handlebars.**
* To work with handle bars they must first be configured using the engine method of your constant ‘APP’
* **app.engine(ext, callback)**
* **the extension is the module you are working with.**
* **The callback is the file that will be worked with.**
* **Follow code provided under app.get method which configures the handlebars template engine.**

**Revisiting chapter 03**

* No longer are we working with the simple http module of the node system, but now a faster process using express.

* App.METHOD – provides methods of the express framework.
* App.use – adds middleware to the server.
* HandleBars is the markup language that can help us write html with special tags to inject content.
* The handlebars is a node module which must be installed through the terminal.
* Any templating engine requires a sub-directory called views to be created.

Configuration for templating engine.

app.engine('handlebars', expressHandlebars({

  defaultLayout:'main',

}))

app.set('view-engine', 'handlebars')

// configuration for handle bars.

* The first two methods for configuring a templating engine that needs to be used is – **engine, and set.**

* Within engine, the name of the templating engine is passed, and then the constant value which was assigned the module of the engine, and then a method, with curly braces for other configurations.
* The **set** method is then called which is passed the “view-engine” and name of templating engine.

app.get('/about', (req,res) => {

  res.render('about')

})

* After configuring the handlebars templated html files, the routes are now updated in order to render the created handlebars pages.

Understanding Quality Assurance.

* The importance of writing quality code cannot be understated.

* Unit testing and integration testing is of few important matters that must be understood.
* A good understanding in development and quality assurance is highly valuable.
* In web development the four main areas of designing a piece of software: the developer must focus on, the **reach, usability, functionality and aesthetics**  of whatever it is that they are creating.

Testing paradigms commonly used:

* **Unit testing:** requires the developer to test the functionality of a single component of the application to see if it’s logic is running to plan.
* **Integration testing:** requires the developer to test the functionality of all components, or the whole system.
* **Puppeteer and Jest will be visited when developing these applications**

**Working with testing frameworks:**

* JEST is a framework that will be installed through the node package manager.

* A script is now added to the package.json file, in order to allow for JEST to do its thing. However, an error occurred because no tests were found. **Unit tests must be created themselves.**

**Going into unit testing:**

* Unit testing calls for functions/components whatever it is to be isolated. One method of doing this is called mocking.