**The Basics Of The Express Framework -** [Ryan Schleck](https://medium.com/@ryan-m-schleck?source=post_page-----830e2c667ba4--------------------------------) Jun 18, 2022

**Introduction**

If you are into software development you have no doubt heard of the [Javascript](https://developer.mozilla.org/en-US/docs/Web/JavaScript" \t "_blank) runtime environment known as [NodeJS](https://nodejs.org/en/about/). I even tried to provide a glimpse of how this technology can be utilized to interact with local files and also serve html as a server in a [previous article](https://ryan-m-schleck.medium.com/an-introduction-to-nodejs-aeddbc7c6a91). In that article we used a method provided to us by NodeJS called [http](https://nodejs.dev/learn/the-nodejs-http-module) to write a lot of the html by hand. In this article we will delve in to a cleaner and easier way to do this by introducing a popular [web framework](https://en.wikipedia.org/wiki/Web_framework) that builds on NodeJS called [Express](https://expressjs.com/).

**What Is Express?**

According to Wikipedia Express was founded by TJ Holowaychuk in 2010 as a means to build web applications and [APIs](https://www.freecodecamp.org/news/what-is-an-api-in-english-please-b880a3214a82/) using Javascript in the NodeJS environment. You can start using express within any NodeJS project. If you need directions on how to install **npm** and **NodeJS** please refer to my [previous article](https://ryan-m-schleck.medium.com/an-introduction-to-nodejs-aeddbc7c6a91). I’m also assuming you have already run **npm init** to create a [package.json](https://docs.npmjs.com/cli/v8/configuring-npm/package-json) file. To install Express make sure you are in your projects root directory within terminal and enter:

npm install --save express

Once this finishes you should see an entry added to your dependencies within your [package.json](https://docs.npmjs.com/cli/v8/configuring-npm/package-json) file with the version of express you installed. You will then go to the top of your application (in this example **app.js**is the top of our application) and remove any routes you have along with what you used within **[createServer](https://nodejs.dev/learn/the-nodejs-http-module" \l "httpcreateserver" \t "_blank).**In order to begin using express you will need to require it near the beginning of the top of your application and set it to a variable. You will then set another variable to the return of the call for your first variable. This latest variable is a valid request handler and can be used to handle and respond to http requests. If this is as confusing as it sounded while I typed it out…here is a picture of the top of a small NodeJS application which uses Express:

A screenshot of a computer

Description automatically generated with medium confidence

Because this resides in a file called **app.js** we would start this project by entering **node app.js** within a terminal within the root of our project and opening a browser to the url of **http://localhost:3000/**

The additional utility this affords you is that you can now ‘use’ simple [http request methods](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods) when you define your routes. We will go over how this is done in greater detail within the next section. Now that we have some of the basics covered lets put it all together with a very simple example!

**Using Express**

One of the advantages Express provides is the ability to use multiple [middleware](https://expressjs.com/en/guide/using-middleware.html) functions that handle a [request object](https://expressjs.com/en/4x/api.html#req), a [response object](https://expressjs.com/en/4x/api.html#res), and the next middleware function. These middleware functions are used to handle multiple routes. In order to better visualize this lets go back to our single page application using Express that displays ‘Hello Express World!’. If we added another response for the same route we can do so using the **next** function like so:

Text

Description automatically generated

If we were to add multiple routes to our app the order in which we add them matters in that it is ‘first come first serve’. The [documentation on routing](https://expressjs.com/en/guide/routing.html) is a good place to start for understanding this process. It is best practice to place your routes in a **routes** folder and export them like so:

Text

Description automatically generated

You would then import these routes and set them to a variable at the top of your app. After doing this you can pass these routes as a middleware function like so:

Text

Description automatically generated

It is also a good idea to set a default page not found route at the end like our example above

Just for a bit of clarity here is a screen grab of the new /users middleware route that is just a very simple form:

Text

Description automatically generated

**Conclusion**

At this point we have a multi route page! Hopefully this has provided a great first glance at what the Express framework is and how to use it. There is of course a lot more that this framework can do. Feel free to checkout [the documentation](https://expressjs.com/) if you would like to look at some of these features yourself. I will focus on additional features Express provides us in future blog posts.