# A Beginner's Intro to Back-End Programming - [Victoria Lo](https://hashnode.com/@victoria) [May 26, 2020](https://lo-victoria.com/a-beginners-intro-to-back-end-programming)

If you are new to programming, you've probably heard a few jargons here and there like "recursion" or "dynamic programming" and so on. One of them is probably **back-end** (no, not yours), aka "**server-side programming**".

## So, what's Back-End Programming?

To put it simply, whenever you visit a website, your browser makes a **HTTP request** to a web server to retrieve information. The server will then send a **HTTP response** back to the browser and the website will be displayed.

Think of your browser as a blank canvas, that has to ask someone (i.e. server) to instruct what things (info) to draw (display) on it. This is why if the browser fails to connect to the server or if the server fails to retrieve the information it needs for the browser to display, you might see an error like this:

Graphical user interface

Description automatically generated

So... Back-end programming is when you write code for the servers, aka the "instructions" to send to the browser on what to display. And, the browser will decide on the style and look on how to display the information in visually appealing, readable ways. Hence, coding for the browser, you guessed it, is front-end programming.

I hope that gives a clear enough definition of back-end programming. Let's move on to the juicy stuff.

## Back-End Languages

Now that you understand what is back-end programming, you might want to start deciding which language to learn for back-end.

So what are some back-end languages to learn? **TLDR: a lot.** Here's a snapshot on some common ones.

### ****SO WHICH LANGUAGE SHOULD I LEARN BEFORE I LOSE MY MIND?****

I hate to admit it but the answer is: it depends. Everyone probably started trying out a few popular ones first like Java, C++, C#, Python, PHP, etc. then decide which language they are most comfortable with.

Personally, because I am already using JavaScript for front-end programming, I go with using JavaScript for back-end too. For that reason, this tutorial will only cover NodeJS. (Sorry to Python, Java or other back-end language users).

## NodeJS and npm

NodeJS (or Node.js) is an open-source server environment which uses JavaScript to code on the server.

If you have ever heard of **npm install {something}**, the 'npm' is talking about the Node Package Manager, which hosts all Node.js packages or modules. Anytime you wish to install a particular package, you can just run **npm install {your-package}** on the command line! It's easy and convenient!

## An example

Let's say you want your web app to make a request to the server. As an example, let's assume you want to retrieve a message from the server.

### ****Step 1: Install axios****

First, we run **npm install axios**. Axios is a a useful package for making HTTP requests from the browser to the server.

### ****Step 2: Make a GET request****

Since we want our app to get information from the server, we should make a GET request like so:

const axios = require('axios'); //so that we can use axios

// Make a request for a message

axios.get('/message')

.then(function (response) {

// handle success

console.log(response);

})

.catch(function (error) {

// handle error

console.log(error);

})

.finally(function () {

// always executed

});

### ****Step 3: Install express****

Our browser has made a GET request, now it's time to have the server to receive its order and return some "instructions" back to the browser.

An npm called express can deal with just that. So run **npm install express** on your command line and create a new file called **server.js**, which will deal with the back-end.

### ****Step 4: Back-end Programming****

Now, we are finally doing some back-end programming, aka telling the server what to return. With our example, let's return the classic "Hello World!" message to the browser upon receiving the GET request.

const express = require('express') //don't forget this

const app = express()

app.get('/message', function (req, res) {

res.send('Hello World')

})

app.listen(3000) //port number

## ALL DONE!

And that's it! A simple example to show you what back-end programming is and how it works! If you'd like to know more about axios and express, check out their documentations ([axios](https://www.npmjs.com/package/axios" \t "_blank) and [express](https://www.npmjs.com/package/express)).

### The Importance of Back-End Programming

After the example, you might think: why does it seem so complicated? Do we really need to do this just to send some "Hello World!"?

It may not be needed in the example earlier but for more complex apps which needs to return dynamic information, back-end programming is essential.

Not only is back-end programming needed to display dynamic information, it also allows an **efficient way to store information**. Let's take a look at an e-commerce site. With all its products, how did it ever managed to load a page for every product? There's no way that the company create a static HTML page for each product they have. That's just impractical! With back-end programming, they can store all their product info in their database and if the user wants to take a look a particular product, the server just needs to retrieve that info and displays in on the site. Much more efficient, isn't it?

Back-end programming also helps to **create personalized user experiences**. Look at web apps like Netflix, which recommends movies and shows based on what each user has watched. Clearly, every user watched different movies and by having the back-end store the user's watch history, Netflix can easily have a personalized recommendation system for each user.

## Conclusion

There is still so much to talk about back-end programming. But this is just an intro so I don't want you to be overloaded with too much info. I commend you for reading this far and I encourage you to read more on your own to find out the beauty of back-end programming~

I hope this introduction has been helpful and motivates you to get started with back-end programming. Do like and share this intro if it helps you in any way, and please do not hesitate to ask questions in the comments below. Our community is all about sharing knowledge after all! Cheers!