**What is Express?**

**Express provides a minimal interface to build our applications. It provides us the tools that are required to build our app. It is flexible as there are numerous modules available on npm, which can be directly plugged into Express.**

**Why Express?**

**Unlike its competitors like Rails and Django, which have an opinionated way of building applications, Express has no "best way" to do something. It is very flexible and pluggable.**

**Installing**

**Assuming you’ve already installed**[**Node.js**](https://nodejs.org/)**, create a directory to hold your application, and make that your working directory.**

**$ mkdir myapp**

**$ cd myapp**

**Use the npm init command to create a package.json file for your application. For more information on how package.json works, see**[**Specifics of npm’s package.json handling**](https://docs.npmjs.com/files/package.json)**.**

**$ npm init**

**This command prompts you for a number of things, such as the name and version of your application. For now, you can simply hit RETURN to accept the defaults for most of them, with the following exception:**

**$ npm install express**

**To install Express temporarily and not add it to the dependencies list:**

**$ npm install express --no-save**

**Index.js**

**var express = require('express');**

**var app = express();**

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

**res.send("Hello world!");**

**});**

**app.listen(3000);**

**How the App Works?**

**The first line imports Express in our file, we have access to it through the variable Express. We use it to create an application and assign it to var app.**

**app.get(route, callback)**

* **This function tells what to do when a get request at the given route is called.**
* **The callback function has 2 parameters, request(req) and response(res).**
* **The request object(req) represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, etc.**
* **Similarly, the response object represents the HTTP response that the Express app sends when it receives an HTTP request.**

**res.send()**

* **This function takes an object as input and it sends this to the requesting client. Here we are sending the string "Hello World!".**

**app.listen(port, [host], [backlog], [callback]])**

* **This function binds and listens for connections on the specified host and port. Port is the only required parameter here.**

**Routers**

**In Express.js, routers are used to organize routes and route handlers into separate modules. This helps in keeping the codebase clean, modular, and easy to maintain, especially for large applications. Routers allow you to break down your application's functionality into smaller, manageable parts.**

* Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).
* Each route can have one or more handler functions, which are executed when the route is matched.

**Route definition takes the following structure:**

* **app.METHOD(PATH, HANDLER)**

**Where:**

* + **app is an instance of express.**
  + **METHOD is an**[**HTTP request method**](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods)**, in lowercase.**
  + **PATH is a path on the server.**
  + **HANDLER is the function executed when the route is matched.**

Here's how you can use routers in Express.js:

1. **Creating a Router:** You can create a router using the express.Router() method. This method returns a router object which you can use to define routes.

// Importing the Express module

const express = require('express');

// Creating a router instance

const router = express.Router();

// Defining a route handler for the root path '/'

router.get('/', function(req, res) {

    // Sending the 'index.html' file as a response

    res.sendFile(\_\_dirname + "/index.html");

});

router.get('/youtube', function(req, res) {

    // Sending the 'index.html' file as a response

    res.redirect("https://www.youtube.com/watch?v=qt\_abdPXoU4");

});

router.get('/blog', function(req, res) {

    // Sending the 'index.html' file as a response

    res.redirect("https://www.youtube.com/watch?v=SccSCuHhOw0");

});

router.get('/instagram', function(req, res) {

    // Sending the 'index.html' file as a response

    res.redirect("https://www.youtube.com/watch?v=I92FyknWFJg");

});

// Exporting the router instance to be used in other parts of the application

module.exports = router;

**Mounting a Router: After creating a router, you need to mount it in your main application file (usually app.js or index.js). You can do this using the app.use() method.**

var express = require('express');

var app = express();

var routing=require('./router.js');

app.use('/',routing);

app.listen(8080);

**https://www.freecodecamp.org/news/express-explained-with-examples-installation-routing-middleware-and-more/**