Creating a server in Node is simple.

There are two main ways to do it.

### #1 Using Node's core HTTP module

We can use Node's core HTTP module.

What do I mean by core?

I mean that it comes shipped with Node. This means you don't have to manually install this module in order to use it.

The following code listens to client requests on port 3000 and returns the text "hello world":

```
1. const http = require('http');
```

```
1. const port = 3000;
2.
3. const server = http.createServer( (req, res) => {
4.   res.writeHead(200, { 'Content-Type': 'text/plain' });
5.   res.write('hello world');
6.   res.end();
7. });
```

At this point, you have configured the server. But you have not yet started it.

To do this, you need to call the **listen()** method:

```
1. server.listen(port, () => {
```

```
1.   console.log(`Server is listening on port 3000`);
2. });
```

### #2 Using Express

For simple servers like the preceding one, you don't need to use Express.

But let's get real.

In the real world, pages are a lot more complex than the ones you have seen here. Usually, there are dozens of different routes, authentication requirements, and login pages. This is where Express comes into the picture.

*(Note: To use Express, you do need to install it on your machine globally or locally for your specific project.)*

Let's use Express to alter the preceding example:

```
1. // you have to install express on your machine in order to do this
```

```
1. const express = require('express');
2. // start your express app
3. const app = express();
4. // define your port
5. const port = 3000;
6.
7. // define your first homepage route
8. app.get('/', (req, res) => {
9.     res.send('hello world');
10.});
11.
12.// start your server
13.app.listen(port, () => {
14.    console.log(`Server listening on port ${port}`);
15.});
```

You can see that, under the hood, Express uses the methods we used directly from Node's HTTP module.

However, Express does more than just set up the initial server. It also gives us a ton of route handlers and settings to help us set up a powerful web app.

Routing just means how a web server will respond to certain URLs, such as **/index.html** and **/about.html**. Usually, to comply with RESTful rules, it is good practice to include verbs when sending AJAX requests to web servers (GET, POST, PUT, or DELETE). Express makes this very easy.

For example, instead of sending a GET request to add items to the user's cart, it is better to send a POST request to the **/cart** page, which will tell the server to add a new item to the user's cart.

**Bottom line: Setting up routes with Express makes our lives a lot easier.**