

What is Node.js Express?

Node.js Express is a framework that can be used to create websites with Node.js. It needs to be installed first. It is not available as a core module with Node.js installation and it needs to be installed first.

We assume that you have Node.js installed on your computer.

Following are the steps to install Express in directory c:\myexpressapp1

Create a directory as myexpressapp1 in c:\

```
mkdir myexpressapp1
```

```
cd myexpressapp1
```

Then run npm init command inside folder c:\myexpressapp1

```
npm init
```

The “npm init” command will initialize a project and create the package.json file.

Above command will show following output

```
PS C:\> cd myexpressapp1
PS C:\myexpressapp1> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (myexpressapp1) |
```

Now keep on pressing Enter till you see Is this OK Message

Now write following command to make myexpressapp1 as Express application

```
npm install express
```

Above command will download and install Express module from npm repository and will make myexpressapp1 as Node.js Express Application.

You will see following output upon successful completion of above command.

```
PS C:\myexpressapp1> npm install express

added 58 packages, and audited 59 packages in 2s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\myexpressapp1>
```

Now we will write first program in express that will create a server that will listen on port 3000.

Give filename as express1.js and write the below given code in it.

If request to server will be a / it will print Hello World! and if the request to server will be /home server will send response Home Page.

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.get('/home', (req, res) => {
  res.send('Home Page')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

Output

Console Output

```
Example app listening on port 3000
```



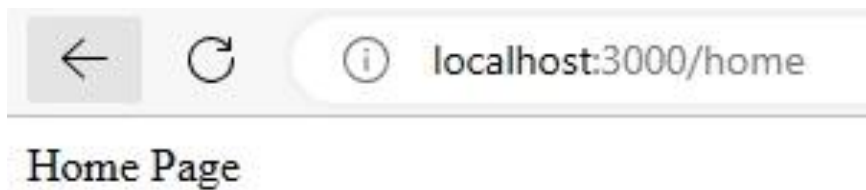
Output in browser when request is a /



localhost:3000

Hello World!

Output in browser when request is /home



Explanation of the above code

```
const express = require('express')
```

Above line will import module express into our nodejs application

```
const app = express()
```

Above line will create a constant app through we will receive requests and send response.

```
const port = 3000
```

Above code will create a constant as port and will assign it a value 3000 and our server will listen on port 3000 using this constant.

```
app.get('/', (req, res) => {  
  res.send('Hello World!')  
})
```

Above code will listen for a HTTP GET request with a / and will send a response to browser as Hello World!

```
app.get('/home', (req, res) => {  
  res.send('Home Page')  
})
```

Above code will listen for HTTP GET request with /home and will send a response to browser as Home Page.

```
app.listen(port, () => {  
  console.log(`Example app listening on port ${port}`)  
})
```

Above code will start a server which will listen on port 3000 and code will also print a message on Console Output that Example app listening on port 3000.

Following code will demonstrate next() function in app.use function

```
const express = require('express');  
  
const app = express();  
  
app.use((req, res, next) => {  
  console.log('In the middleware!');  
});
```

```
    next(); // Allows the request to continue to the next middleware in line
  });

app.use((req, res, next) => {
  console.log('In another middleware!');
  res.send('<h1>Hello from Express!</h1>');
});

app.listen(3000);
```

Following code will listen to a request from user and will print In the middleware on the Console and will transfer the control to function below it that is

```
app.use((req, res, next) => {
  console.log('In another middleware!');
  res.send('<h1>Hello from Express!</h1>');
});
```

Above code will print message In another middleware in Console and will also send a response in browser

Hello from Express.

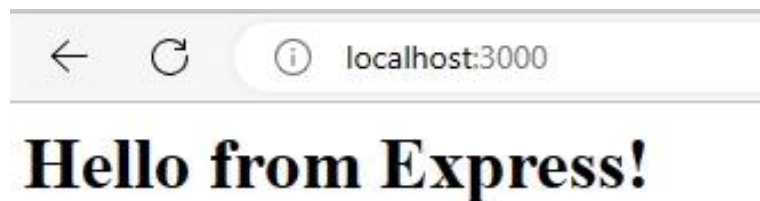
You can see following output in console in browser when

<http://localhost:3000> is called in browser

Output in console

```
In the middleware!
In another middleware!
```

Output in browser



Example of Express Code 2

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.urlencoded({extended: false}));

app.use('/add-product', (req, res, next) => {
  res.send('<form action="/product" method="POST"><input type="text"
name="title"><button type="submit">Add Product</button></form>');
```

```
});

app.post('/product', (req, res, next) => {
  console.log(req.body.title);

  res.redirect('/');
});

app.use('/', (req, res, next) => {
  res.send('<h1>Hello from Express!</h1>');
});

app.listen(3000);
```

```
const bodyParser = require('body-parser');
```

Above code will import module body-parser which is needed to get the body of the request sent by the browser.

```
app.use(bodyParser.urlencoded({extended: false}));
```

Above code is needed to get body of request in req.body and print it on console.

```
app.use('/add-product', (req, res, next) => {
  res.send('<form action="/product" method="POST"><input type="text"
name="title"><button type="submit">Add Product</button></form>');
});
```

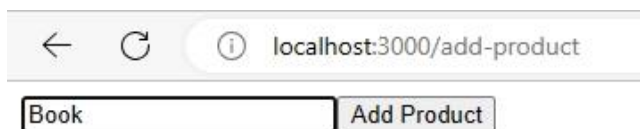
Above code will send a form which will send a post request to /product.

```
app.post('/product', (req, res, next) => {
  console.log(req.body.title);
  res.redirect('/');
});
```

Above code will handle the post request /product and will print the form data as req.body on console and then will redirect the browser to / and upon receiving a / request response will be sent as Hello From Express!.

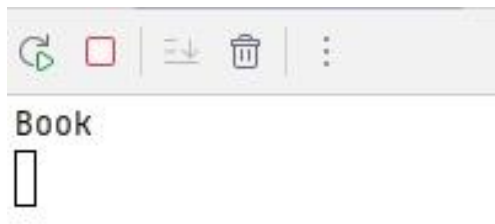
```
app.use('/', (req, res, next) => {
  res.send('<h1>Hello from Express!</h1>');
});
```

Output in browser after receiving request /add-product



After Clicking on Add Product Button

Output in Console



Output in browser



Example of Express Code or Application to send html files to browser

```
const express = require('express');
const path = require('path');
const app = express();
const PORT = 3000;

app.get('/', function (req, res, next) {
  res.sendFile(path.join(__dirname, 'views', 'index.html'));
});

app.get('/products', function (req, res, next) {
  res.sendFile(path.join(__dirname, 'views', 'products.html'));
});

app.listen(3000);
```

Above code will send file index.html in views folder under the current directory upon receiving request / and when it will receive request /products it will send file products.html in views folder in current directory.

Now create a new directory named as views under the current working directory or where the express application is present in this case it is c:\myexpressapp1

Inside views folder create two html files index.html and products.html

code for index.html is

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
```

```
    This is index page
  </body>
</html>
```

Code for products.html file is

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  This is Products Page...
</body>
</html>
```

Below is the explanation of the nodejs express code

```
const path = require('path');
```

Above code will import a core module in express that is path, this path core module is used to the files in express code.

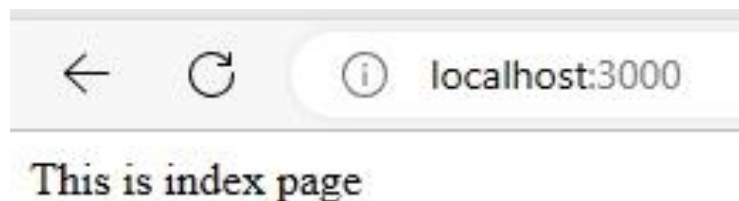
Notice the following code

```
res.sendFile(path.join(__dirname, 'views', 'index.html'));
```

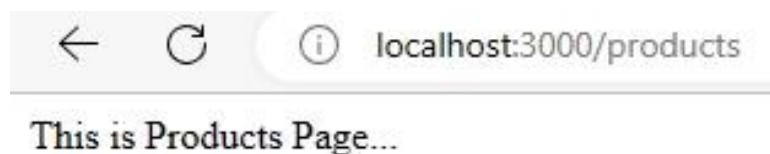
path.join(__dirname, 'views', 'index.html') will give path to index.html in views folder in the current working directory.

__dirname is used to give the path of current working directory.

Output in browser when a / request is made in browser



Output in browser when a /products request is made in browser



Following code will give response to status code 404 which means page is not found on server.

```
const express = require('express');
const path = require('path');
```

```

const app = express();
const PORT = 3000;

// With middleware
app.get('/', function (req, res, next) {
  res.sendFile(path.join(__dirname, '', 'index.html'));
});

app.get('/products', function (req, res, next) {
  res.sendFile(path.join(__dirname, '', 'products.html'));
});

app.use((req, res, next) => {
  res.status(404).sendFile(path.join(__dirname, '', '404.html'));
});

app.listen(3000);

```

Output in browser when /page1 is request to sent to server through browser



Page Not Found

How to send css files, image files and other static files in express to server.

In express you simply can't embed css files and image files like jpg or gif files in html pages.

you need to use following line of code in your express code to send static files

```
app.use(express.static('public'));
```

the above code will tell express code that static content inside html page is present in public folder inside the directory where express application is present.

Now create a public folder inside c:\myexpressapp1 and place two files in it

main.css

```

.h1class1{
  color: red;
}

```

and

laptop.jpg



Now go to directory of c:\myexpressapp1

and create a file html9.html with following code

```
<html>
<head>
  <title>Title</title>
  <link rel="stylesheet" href="/main.css" type="text/css">
</head>

<body>
<h1 class="h1class1">Hello World Implementation of Static Files</h1>

</body>
</html>
```

Now write the following code in express file like express9.js

```
const express = require('express');
const path = require('path');
const app = express();
const PORT = 3000;

app.use(express.static('public'));
// With middleware
app.get('/index', function (req, res, next) {
  res.sendFile(path.join(__dirname, 'html9.html'));
});

app.listen(3000);
```

Run the above express9.js file and you will see the following output in browser

Hello World Implementation of Static Files



Now we will see a express code where we will print data entered by user in a form and print the data in console

Under views folder create a file as form1.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<form action="/printdata" method="POST">
  Name : <input type="text" name="nametxt">
  <br>
  <input type="submit" value="Print">
</form>
</body>
</html>
```

Now the above html code will display a html page which will display a form which will send a post request to server as /printdata

Now write express code in express10.js as

```
const express = require('express');
const path = require('path');
const app = express();
const PORT = 3000;

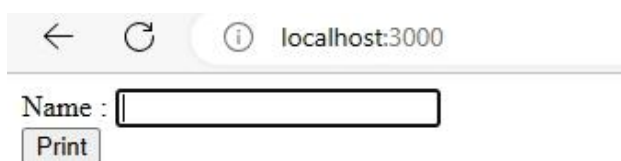
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.get('/', function (req, res, next) {
  res.sendFile(path.join(__dirname, 'views', 'form1.html'));
});
app.post('/printdata', function (req, res, next) {
  const data = req.body;
  console.log(data.nametxt);
  res.send();
});

app.listen(3000);
```

Now run the code and point your browser to <http://localhost:3000>

and you will see output in browser



The screenshot shows a web browser window. The address bar at the top displays 'localhost:3000'. Below the address bar, there is a form. The form starts with the text 'Name :', followed by a text input field. Below the input field is a button labeled 'Print'.

Type in the name textbox as Raman and Click on Print

You will see the following output in console



Notice the following code

```
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
```

The above is needed to get the form data in json form

```
app.post('/printdata',function (req,res,next) {
  const data = req.body;
  console.log(data.nametxt);
  res.send();
});
```

Above code will parse post request as /printdata and will first store body of the request which means data entered in the form in data and will then take out data in form field nametxt and will print it on console.

Now how to print data entered by user in a form in a html page

For this purpose we need to use a template engine and we will use template engine as ejs

create an ejs file with name as printformdata.ejs in views folder and write the following code in it.

```
<!DOCTYPE html>
<html>
<head>
  <title>Home Page</title>
</head>

<body>

This is our home page.<br />
Welcome <%=nametxt%>, to our home page.
</body>
</html>
```

Now create a html file in views folder with name as form2.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
<form action="/printdata" method="POST">
  Name : <input type="text" name="nametxt">
```

```
<br>
<input type="submit" value="Print">
</form>
</body>
</html>
```

Now create an express11.js file with the following code

```
const express = require('express');
const path = require('path');
const app = express();
const PORT = 3000;
app.set('view engine', 'ejs');
var nametxt;
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.get('/', function (req, res, next) {
  res.sendFile(path.join(__dirname, 'views', 'form2.html'));
});
app.post('/printdata', function (req, res, next) {
  const data = req.body;
  nametxt=data.nametxt;
  res.render('printformdata', { nametxt: nametxt });
});
app.listen(3000);
module.exports = nametxt;
```

Explanation of the code is as follows

```
app.set('view engine', 'ejs');
```

Above code will set the template engine as ejs.

There are many other template engines available in express and popular ones are pug and handlebars but we will use ejs in this express application.

```
var nametxt;
```

above code will create a variable nametxt in our express code

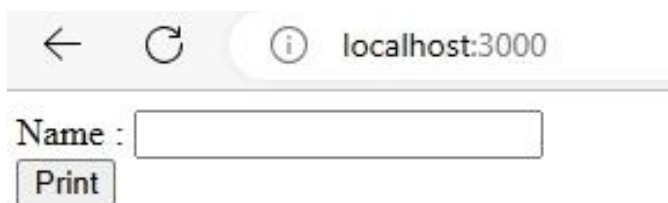
```
module.exports = nametxt;
```

Above code will send or we can say export nametxt variable to our ejs file printformdata.ejs file in views folder.

```
res.render('printformdata', { nametxt: nametxt });
```

Above code will make variable nametxt in printformdata.ejs file.

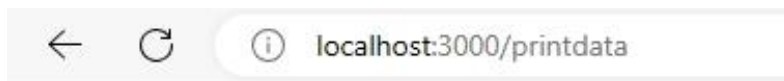
Output in browser will be for url <http://localhost:3000/>



The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. Below the address bar, there is a form with a text input field labeled 'Name :'. The input field is empty. Below the input field, there is a button labeled 'Print'.

Now enter Name as Raman and click on Print

You will see the following output



This is our home page.
Welcome Raman, to our home page.
