

Express JS

Express.js is a web framework for Node.js. It is a fast, robust and asynchronous in nature.

It provides various features that make web application development fast and easy which otherwise takes more time using only Node.js.

Express.js is based on the Node.js middleware module called **connect** which in turn uses **http** module. So, any middleware which is based on connect will also work with Express.js.

Advantages of Express.js

1. Makes Node.js web application development fast and easy.
2. Easy to configure and customize.
3. Allows you to define routes of your application based on HTTP methods and URLs.
4. Includes various middleware modules which you can use to perform additional tasks on request and response.
5. Easy to integrate with different template engines like Jade, Vash, EJS etc.
6. Allows you to define an error handling middleware.
7. Easy to serve static files and resources of your application.
8. Allows you to create REST API server.
9. Easy to connect with databases such as MongoDB, Redis, MySQL

Install Express.js

You can install express.js using npm. The following command will install latest version of express.js globally on your machine so that every Node.js application on your machine can use it.

```
npm install -g express
```

The following command will install latest version of express.js local to your project folder.

```
C:\MyNodeJSApp> npm install express -save
```

Example 1

```
var express = require('express');
var app = express();

app.get('/', function(req, res){
  res.send("Hello world!");
});

app.listen(3000);
```

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)

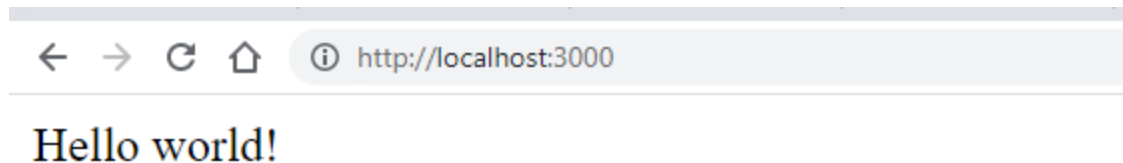
This function tells what to do when a **get** request at the given route is called. The callback function has 2 parameters, **request(req)** and **response(res)**. The request **object(req)** represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers,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



Express JS Routing

Web frameworks provide resources such as HTML pages, scripts, images, etc. at different routes.

The following function is used to define routes in an Express application –

app.method(path, handler)

This METHOD can be applied to any one of the HTTP verbs – get, set, put, delete. An alternate method also exists, which executes independent of the request type.

Path is the route at which the request will run.

Handler is a callback function that executes when a matching request type is found on the relevant route. For example,

Expget.js

```
var express = require('express');
var app = express();

app.get('/example', function(req, res){
  res.send("Hello World!");
});

app.listen(3000);
```

If we run our application and go to **localhost:3000/example**, the server receives a get request at route **/example**, our Express app executes the **callback** function attached to this route and sends **"Hello World!"** as the response.

← → ↻ 🏠 ⓘ http://localhost:3000/example

Hello World!

Exppost.js

We can also have multiple different methods at the same route. For example,

```
var express = require('express');
var app = express();

app.get('/example', function(req, res){
  res.send("Hello World!");
});

app.post('/example', function(req, res){
  res.send("You just called the post method at '/example'!\n");
});

app.listen(3000);
```

To test this request, open up your terminal and use cURL to execute the following request –

```
curl -X POST http://localhost:3000/hello
```

A special method, **all**, is provided by Express to handle all types of http methods at a particular route using the same function. To use this method, try the following.

```
app.all('/test', function(req, res){
  res.send("HTTP method doesn't have any effect on this route!");
});
```

This method is generally used for defining middleware

Routers

Defining routes like above is very tedious to maintain. To separate the routes from our main **index.js** file, we will use **Express.Router**. Create a new file called **things.js** and type the following in it.

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

```
var router = express.Router();

router.get('/', function(req, res){
  res.send('GET route on things.');
```



```
});
router.post('/', function(req, res){
  res.send('POST route on things.');
```



```
});

//export this router to use in our index.js
module.exports = router;
```

Now to use this router in our **index.js**, type in the following before the **app.listen** function call.

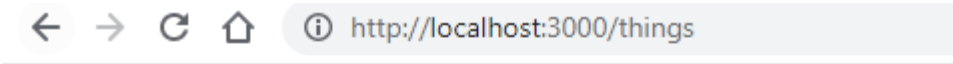
```
var express = require('Express');
var app = express();

var t = require('./things.js');

//both index.js and things.js should be in same directory
app.use('/things', t);

app.listen(3000);
```

The **app.use** function call on route **'/things'** attaches the **things** router with this route. Now whatever requests our app gets at the **'/things'**, will be handled by our things.js router. The **'/'** route in things.js is actually a subroute of **'/things'**. Visit **localhost:3000/things/** and you will see the following output.



← → ↻ ⬆ ⓘ http://localhost:3000/things

GET route on things.

Routers are very helpful in separating concerns and keep relevant portions of our code together. They help in building maintainable code. You should define your routes relating to an entity in a single file and include it using the above method in your **index.js** file.

HTTP methods

The HTTP method is supplied in the request and specifies the operation that the client has requested. The following table lists the most used HTTP methods –

S.No.	Method & Description
1	GET The GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect.
2	POST The POST method requests that the server accept the data enclosed in the request as a new object/entity of the resource identified by the URI.
3	PUT The PUT method requests that the server accept the data enclosed in the request as a modification to existing object identified by the URI. If it does not exist then the PUT method should create one.
4	DELETE The DELETE method requests that the server delete the specified resource.

Exprouteall.js

Route various request to the specified target

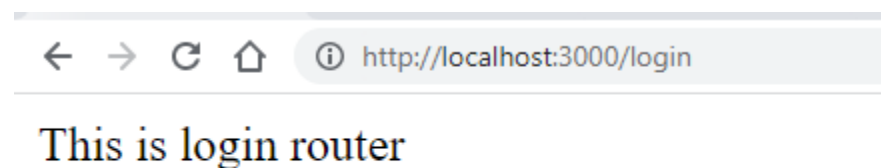
```
const express = require('express');
const app = express();
const router = express.Router();

router.get('/home', (req,res) => {
  res.send('Hello World, This is home router');
});

router.get('/profile', (req,res) => {
  res.send('
  Hello World, This is profile router
```

```
    ');  
  });  
  
  router.get('/login', (req,res) => {  
    res.send(  
      'Hello World, This is login router'  
    );  
  });  
  
  router.get('/logout', (req,res) => {  
    res.send(  
      'Hello World, This is logout router'  
    );  
  });  
  
  app.use('/', router);  
  
  app.listen(process.env.port || 3000);  
  
  console.log('Web Server is listening at port '+ (process.env.port || 3000));
```

```
C:\Users\Admin>node exprouteall.js  
Web Server is listening at port 3000
```



← → ↻ 🏠 ⓘ http://localhost:3000/logout

This is logout router

Handle GET Request

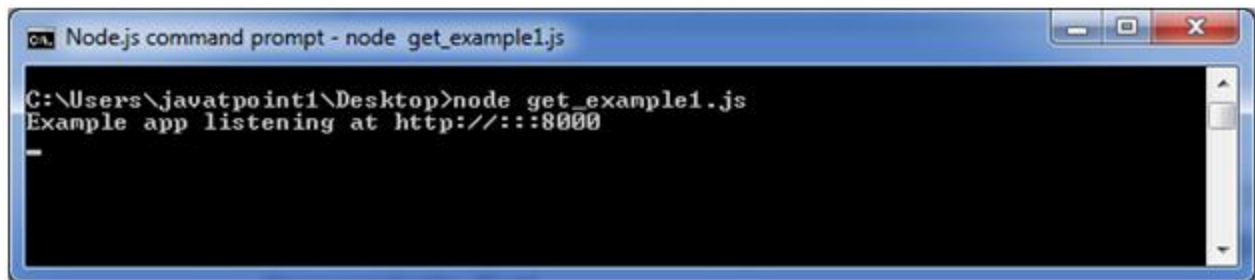
example.html

1. `<html>`
2. `<body>`
3. `<form action="http://127.0.0.1:8081/process_get" method="GET">`
4. First Name: `<input type="text" name="first_name">` `
`
5. Last Name: `<input type="text" name="last_name">`
6. `<input type="submit" value="Submit">`
7. `</form>`
8. `</body>`
9. `</html>`

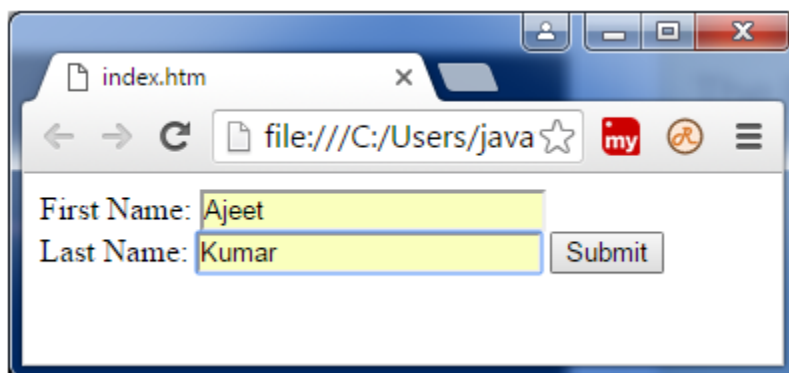
getexample1.js

1. `var express = require('express');`
2. `var app = express();`
3. `app.use(express.static('public'));`
- 4.
5. `app.get('/index.html', function (req, res) {`
6. `res.sendFile(__dirname + "/" + "index.html");`
7. `})`
8. `app.get('/process_get', function (req, res) {`
9. `response = {`
10. `first_name:req.query.first_name,`


```
11.     last_name:req.query.last_name
12. };
13. console.log(response);
14. res.end(JSON.stringify(response));
15. })
16. var server = app.listen(8000, function () {
17.
18.     var host = server.address().address
19.     var port = server.address().port
20.     console.log("Example app listening at http://%s:%s", host, port)
21.
22. })
```



Open the page index.html and fill the entries:



Handle POST Request

Lets see how to handle HTTP POST request and get data from the submitted form.

First, create Index.html file in the root folder of your application and write the following HTML code in it.

expindex.html

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta charset="utf-8" />
  <title></title>
</head>
<body>
  <form action="/submit-student-data" method="post">
    First Name: <input name="firstName" type="text" /> <br />
    Last Name: <input name="lastName" type="text" /> <br />
    <input type="submit" />
  </form>
</body>
</html>
```

Body Parser

To handle HTTP POST request in Express.js version 4 and above, you need to install middleware module called [body-parser](#).

This body-parser module parses the JSON, buffer, string and url encoded data submitted using HTTP POST request. Install body-parser using NPM as shown below.

```
npm install body-parser --save
```

Now, import body-parser and get the POST request data as shown below.

expapp.js

```
var express = require('express');
var app = express();

var bodyParser = require("body-parser");
```

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

app.get('/', function (req, res) {
  res.sendFile('expindex.html');
});

app.post('/submit-student-data', function (req, res) {
  var name = req.body.firstName + ' ' + req.body.lastName;

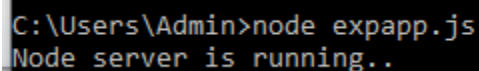
  res.send(name + ' Submitted Successfully!');
});

var server = app.listen(5000, function () {
  console.log('Node server is running..');
});
```

In the above example, POST data can be accessed using req.body. The req.body is an object that includes properties for each submitted form. Index.html contains firstName and lastName input types, so you can access it using req.body.firstName and req.body.lastName.

Run node server

run the above example using `node expapp.js` command, point your browser to *http://localhost:5000* and see the following result.



```
C:\Users\Admin>node expapp.js
Node server is running..
```

Fill the First Name and Last Name in the above example and click on **submit**. For example, enter "James" in First Name textbox and "Bond" in Last Name textbox and click the submit button. The following result is displayed.

← → ↻ 🏠 ⓘ http://localhost:5000

First Name:

Last Name:

First Name:

Last Name:

← → ↻ 🏠 ⓘ http://localhost:5000/submit-student-data

VIT University Submitted Successfully!

Insert into database

App.js

```
var express=require("express");
```

```
var bodyParser=require("body-parser");
```

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

```
mongoose.connect('mongodb://localhost:27017/gfg');
```

```
var db=mongoose.connection;

db.on('error', console.log.bind(console, "connection error"));

db.once('open', function(callback){

    console.log("connection succeeded");

})

var app=express()

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

app.use(express.static(__dirname+"/public"));

app.get("/",function(req,res){

    res.sendFile(__dirname+"/index1.html");

});

app.post('/signup', function(req,res){

    var name = req.body.name;

    var email =req.body.email;

    var pass = req.body.password;

    var phone =req.body.phone;
```

```
var data = {

    "name": name,

    "email":email,

    "password":pass,

    "phone":phone

}

db.collection('details').insertOne(data,function(err, collection){

    if (err) throw err;

    console.log("Record inserted Successfully");

});

return res.sendFile(__dirname+"/signup_success.html");

})

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

```
res.set({  
    'Access-control-Allow-Origin': '*'  
});  
  
return res.redirect('index1.html');  
  
}).listen(8081)
```

Index1.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title> Signup Form</title>
```

```
<link rel="stylesheet"
```

```
href=
```

```
"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
```

```
integrity=
```

```
"sha384-BVYiISiFeK1dGmJRAkyCuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
```

```
crossorigin="anonymous">
```

```
<link rel="stylesheet" type="text/css" href="style.css">
```

```
</head>
```

```
<body>
```

```
<br>
```

```
<br>
```

```
<br>
```

```
<div class="container" >
```

```
<div class="row">
```

```
<div class="col-md-3">
```

```
</div>
```

```
<div class="col-md-6 main">
```

```
<form action="/signup" method="post">
```

```
<h1> Signup form </h1>
```

```
<input class="box" type="text" name="name" id="name"
```

```
placeholder="Name" required /><br>
```



```
<input class="box" type="email" name="email" id="email"
placeholder="E-Mail " required /><br>
```

```
<input class="box" type="password" name="password"
id="password" placeholder="Password " required/><br>
```

```
<input class="box" type="text" name="phone" id="phone"
placeholder="Phone Number " required/><br>
```

```
<br>
```

```
<input type="submit" id="submitDetails"
name="submitDetails" value="Submit" /><br>
```

```
</form>
```

```
</div>
```

```
<div class="col-md-3">
```

```
</div>
```

```
</div>
```

</div>

</body>

</html>