

Express.js Tutorial

Express.js Tutorial

Welcome to Express.js Tutorial. In this series of Express.js Tutorial, we will learn how to get started with Express.js and different concepts of Express.js with well detailed examples.

Get Started with Express.js

Following two tutorials provide you an detailed introduction to Express.js web framework and installation.

- [What is Express.js?](#) – A brief introduction to Express.js.
- [Install Express.js](#) – Steps to install express.js using npm.

Express.js Example

Following is a simple example for Express.js application.

```
var express =  
require('express')  
  
var express = require('express')  
  
// create express application instance  
var app = express()  
// express route  
app.get('/', function (req, res) {  
  res.send('This is a basic Example for Express.js by TUTORIALKART')  
})  
// start server  
var server = app.listen(8000)
```

In the above code, we created an instance of express application, then defined a router to handle `GET` requests on URL path `/`. Then we started the server to listen on port `8000`.

A more detailed example to build a web application and make it run is provided at: [Express.js Tutorial – Express.js Example Application](#).

Express.js Routes

The express.js routes are those that handle a specific HTTP Request on a specified URL path. Following is an example Express route.

```
// express route  
app.get('/hello/')
```

```
// express route
app.get('/hello/', function (req, res) {
  res.send('This is a basic Example for Express.js by TUTORIALKART')
})
```

app is the express application instance. We can call HTTP methods like GET (like in the above code snippet), POST, HEAD, COPY, PATCH, MOVE, etc. The first argument is the URL path. The function (second argument to route) gets hooked to paths that match with the path specified. From the above example, the function (req, res) gets hooked to only those requests that have the path baseurl `/hello/`.

Detailed Express.js Tutorial on Routes – [Express.js Routes](#).

Express.js Middleware

Middlewares are functions that can be executed in an order for a request before sending a response to the client. Following is an example.

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

var express = require('express')
var app = express()
// define middleware function
function logger(req, res, next) {
  console.log(new Date(), req.url)
  next()
}
// calls logger:middleware for each request-response cycle
app.use(logger)
```

logger is a middleware function, where it can get request and response as arguments. Also next() function to continue with the other functions in the request-response cycle.

Complete Express.js Tutorial on Middleware – [Express Middleware](#).

Express.js Router

Express Router is used to Create independent Router objects.

Node.js

» [Node.js Tutorial](#)

Get Started With Node.js

» [Install Node.js Ubuntu Linux](#)

» [Install Node.js Windows](#)

» [Node.js - Basic Example](#)

» [Node.js - Command Line Arguments](#)

» [Node.js - Modules](#)

» [Node.js - Create a module](#)

» [Node.js - Add new functions to Module](#)

» [Node.js - Override functions of Module](#)

» [Node.js - Callback Function](#)

» [Node.js - forEach](#)

Express.js

» [Express.js Tutorial](#)

» [What is Express.js?](#)

» [Express.js Application Example](#)

» [Install Express.js](#)

» [Express.js Routes](#)

» [Express.js Middleware](#)

» [Express.js Router](#)

Node.js Buffers

» [Node.js Buffer - Create, Write, Read](#)

» [Node.js Buffer - Length](#)

» [Node.js - Convert JSON to Buffer](#)

» [Node.js - Array to Buffer](#)

Node.js HTTP

» [Node.js - Create HTTP Web Server](#)

» [Node.js - Redirect URL](#)

Node.js MySQL

» [Node.js MySQL](#)

» [Node.js MySQL - Connect to MySQL Database](#)

» [Node.js MySQL - SELECT FROM](#)

» [Node.js MySQL - SELECT WHERE](#)

» [Node.js MySQL - ORDER BY](#)

» [Node.js MySQL - INSERT INTO](#)

» [Node.js MySQL - UPDATE](#)

Node.js MySQL - DELETE

Node.js MySQL - Result Object

Node.js MongoDB

Node.js MongoDB

Node.js - Connect to MongoDB

Node.js - Create Database in MongoDB

Node.js - Drop Database in MongoDB

Node.js - Create Collection in MongoDB

Node.js - Delete Collection in MongoDB

Node.js - Insert Documents to MongoDB Collection

MongoError: failed to connect to server

Node.js Mongoose

Node.js Mongoose Tutorial

Node.js Mongoose - Installation

Node.js Mongoose - Connect to MongoDB

Node.js Mongoose - Define a Model

Node.js Mongoose - Insert Single Document to MongoDB

Node.js Mongoose - Insert Multiple Documents to MongoDB

Node.js URL

Node.js - Parse URL parameters

Node.js FS (File System)

Node FS

Node FS - Read a File

Node FS - Create a File

Node FS - Write to a File

Node FS - Append to a File

Node FS - Rename a File

Node FS - Delete a File

Node FS Extra - Copy a Folder

Node.js JSON

Node.js Parse JSON

Node.js Write JSON Object to File

Node.js Error Handling

Node.js Try Catch

» [Node.js Try Catch](#)

Node.js Examples

» [Node.js Examples](#)

» [Node.js - Handle Get Requests](#)

» [Node.js Example - Upload files to Node.js server](#)

Useful Resources

» [Node.js Interview Questions](#)

» [How to Learn Programming](#)