

Introduction to ExpressJS

Agenda

- Introduction to ExpressJS
- Install ExpressJS
- ExpressJS – First Program
- Building Application Stack
- Express Routing
- Express Middleware

Introduction to ExpressJS

- Express is a web application framework for Node.
- It is a server-side or back-end framework not comparable to client-side frameworks like React, and Angular but can be used in combination to build full-stack applications.
- Express is a fast, unopinionated, and minimalist web framework for Node.js.
- Building web applications with Node.js is MUCH easier.
- Used for both server-rendered apps as well as APIs.
- Full control of request and response.
- Most popular framework for Node.
- Great to use with client-side frameworks as it's all JavaScript.

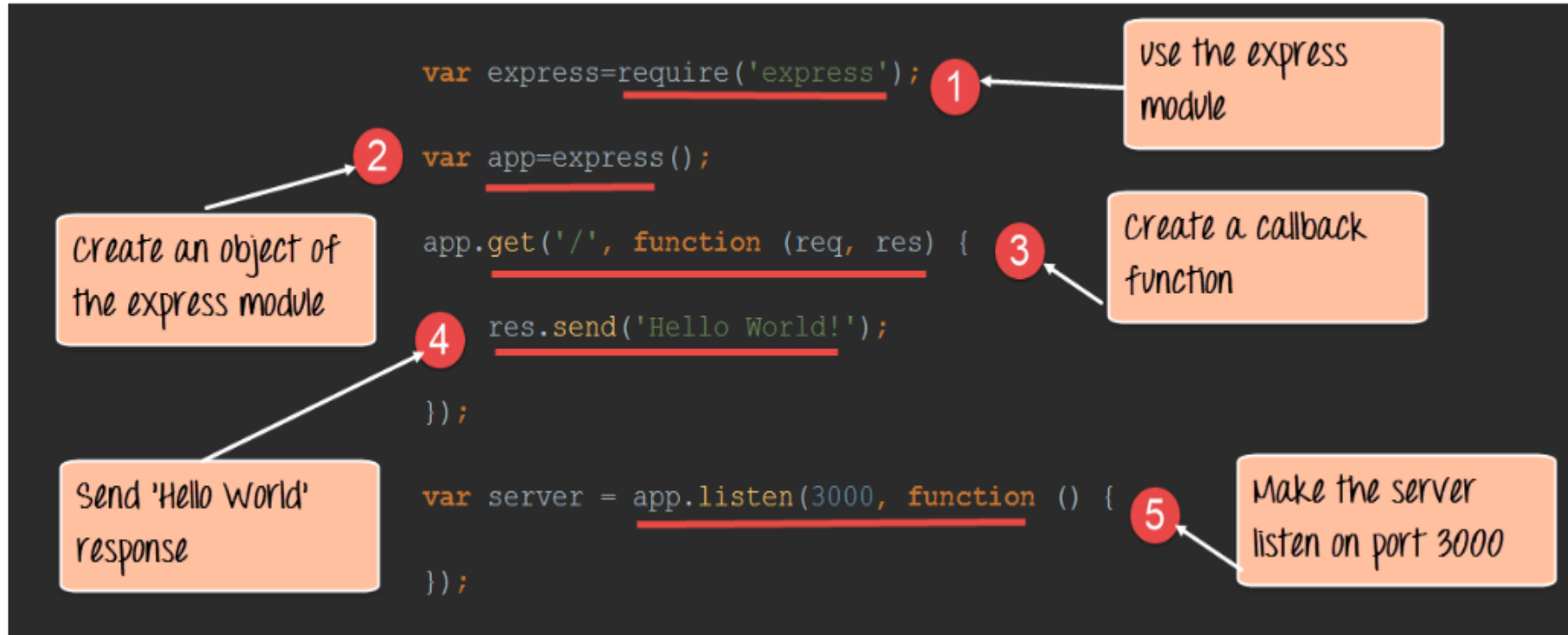
Most Popular Node.js Frameworks



Prerequisites

- JS fundamentals (Objects, Arrays, Conditions etc)
- Basic Node.js and NPM
- HTTP status codes
- JSON

Basic Server Syntax




Basic Route Handling

- Handling requests/route is simple.
- `app.get()`, `app.post()`, `app.put()`, `app.delete()` etc.
- Access to params, query string, url parts, etc
- Express has routers so we can store routes in separate files and export.
- We can parse incoming data with Body Parser.

```
app.get('/', function(req, res) {  
  // Fetch from database  
  // Load pages  
  // Return JSON  
  // Full access to request & response  
});
```

Install Express JS

- Prerequisite: Node.js installed on the system.
- Visit the website for installation of node.js <http://nodejs.org/en>
- Assuming you've already installed [Node.js](#).
- Create a directory to hold your application, and make that as your working directory.

 Command Prompt

```
Microsoft Windows [Version 10.0.18362.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Aisha Begum>cd express

C:\Users\Aisha Begum\express>mkdir myexpress

C:\Users\Aisha Begum\express>cd myexpress

C:\Users\Aisha Begum\express\myexpress>_
```

Install Express JS

- Use the **npm init** command to create a package.json file for your application.
- This command prompts you for a number of things, such as the name and version of your application. Hit RETURN to accept the default settings.
- For more information on how package.json works, see specifics of [npm's package.json handling](#).

Install Express JS

 npm

```
C:\Users\Aisha Begum\express\myexpress>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.

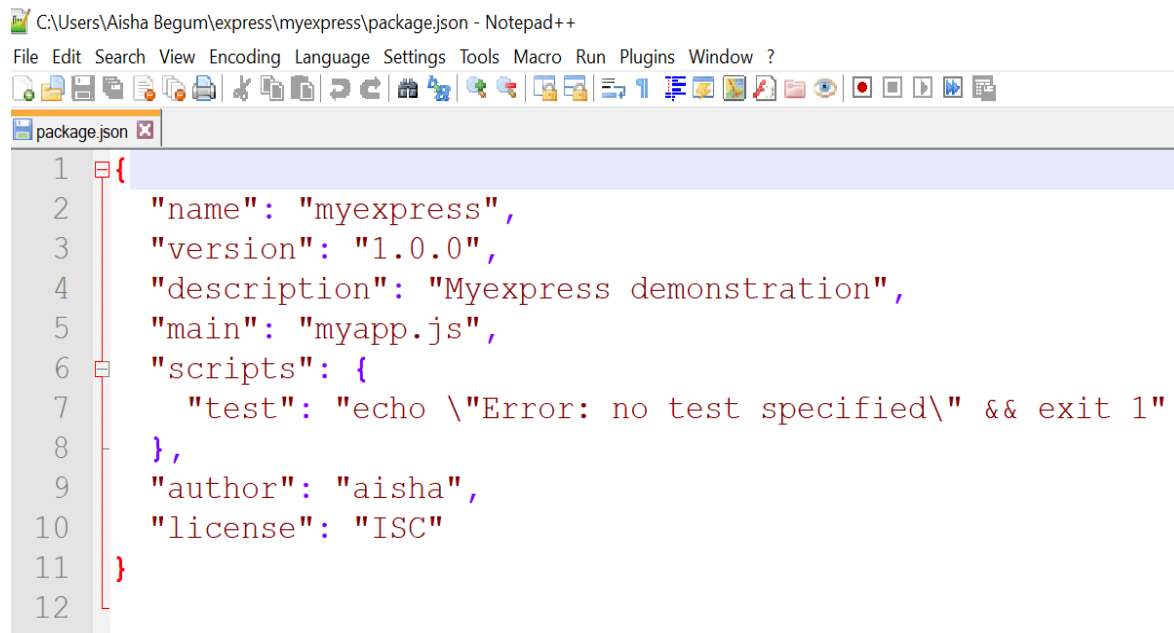
Press ^C at any time to quit.
package name: (myexpress)
version: (1.0.0)
description: Myexpress demonstration
entry point: (index.js) myapp.js
test command:
git repository:
keywords:
author: aisha
license: (ISC)
About to write to C:\Users\Aisha Begum\express\myexpress\package.json:

{
  "name": "myexpress",
  "version": "1.0.0",
  "description": "Myexpress demonstration",
  "main": "myapp.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "aisha",
  "license": "ISC"
}

Is this OK? (yes) _
```

Install Express JS

- The package.json file for your application is as shown below.
- Used notepad++ text editor



The screenshot shows a Notepad++ window with the file path `C:\Users\Aisha Begum\express\myexpress\package.json`. The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and ?. The toolbar contains various icons for file operations. The active tab is `package.json`. The code content is as follows:

```
1 {  
2   "name": "myexpress",  
3   "version": "1.0.0",  
4   "description": "Myexpress demonstration",  
5   "main": "myapp.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "author": "aisha",  
10  "license": "ISC"  
11 }  
12
```

Install Express JS

- Use the **npm i express** command to install express

```
package.json
1 {
2   "name": "myexpress",
3   "version": "1.0.0",
4   "description": "Myexpress demonstration",
5   "main": "myapp.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "author": "aisha",
10  "license": "ISC",
11  "dependencies": {
12    "express": "^4.17.1"
13  }
14 }
15
```

```
C:\Users\Aisha Begum\express\myexpress>npm i express
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN myexpress@1.0.0 No repository field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 12.804s
found 0 vulnerabilities

C:\Users\Aisha Begum\express\myexpress>
```

Install Express JS

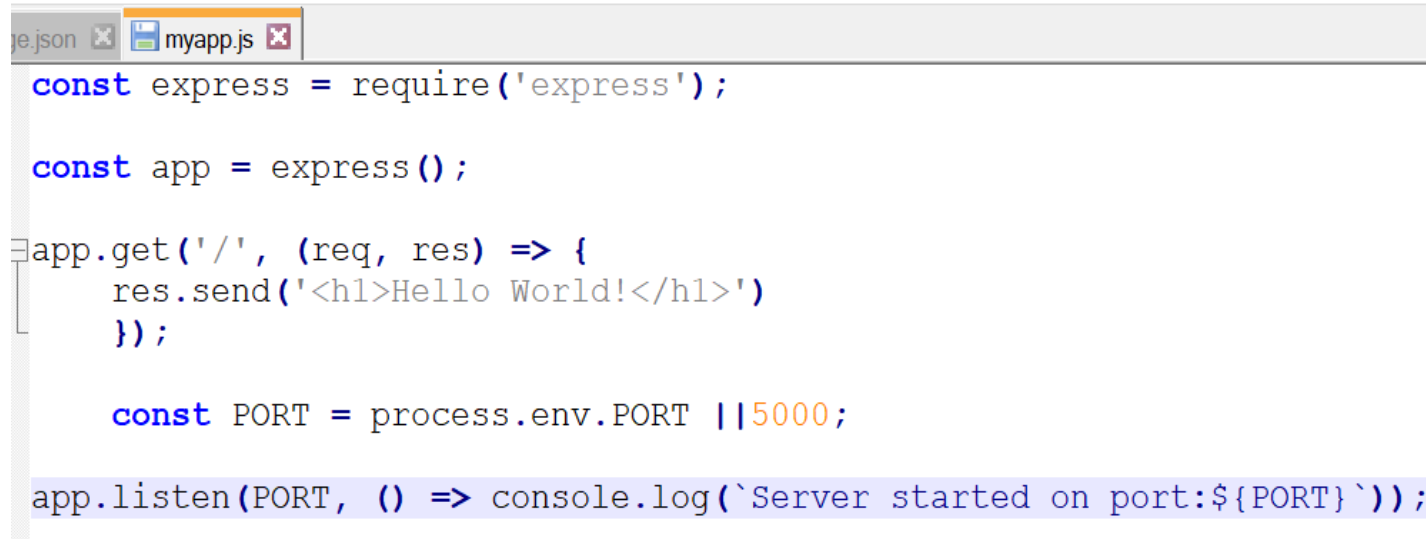
- Create a myapp.js file in the root folder of express.

express > myexpress

Name	Date modified	Type
node_modules	09-06-2020 11:04	File folder
myapp	09-06-2020 11:14	JavaScript File
package.json	09-06-2020 11:04	JSON File
package-lock.json	09-06-2020 11:04	JSON File

Express JS – First Program

- First program to print Hello World using express js.



The screenshot shows a code editor with two tabs: 'package.json' and 'myapp.js'. The 'myapp.js' tab is active and contains the following JavaScript code:

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

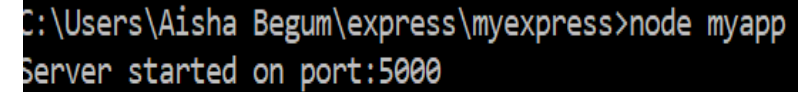
const app = express();

app.get('/', (req, res) => {
  res.send('<h1>Hello World!</h1>');
});

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => console.log(`Server started on port:${PORT}`));
```

Command Prompt - node myapp

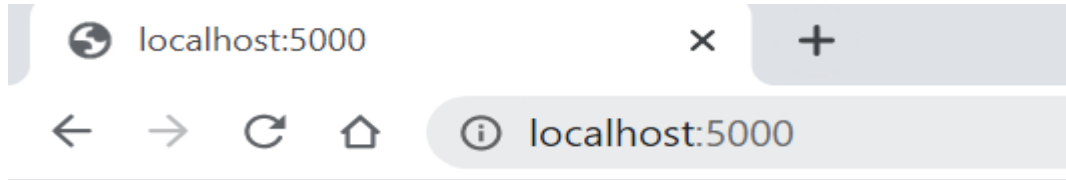


The screenshot shows a Command Prompt window with the following text:

```
C:\Users\Aisha Begum\express\myexpress>node myapp
Server started on port:5000
```

Express JS – First Program

- We can check the output on any browser by using the url: `http://localhost/5000`.



Hello World!

Building Application Stack

- What is MERN Stack?
- A stack is the mixture of technologies used to create Web applications.
- Any web application will be made utilizing various technologies like (frameworks, libraries, databases).
- The MERN stack is a JavaScript stack that is intended to make the Application Development process smoother.

Building Application Stack

- MERN Stack Components:
- MERN incorporates four open-source components:
 - MongoDB,
 - Express,
 - React, and
 - Node.js.
- These components give an end to end frameworks for developers to work with.

Building Application Stack

- Components Overview:
- MongoDB A document-oriented, No-SQL database used to store the application data.
- NodeJS: The JavaScript runtime environment. Used to run JavaScript on a machine rather than in a browser.
- ExpressJS: A framework layered on top of NodeJS, used to build the backend of a site using NodeJS functions and structures.
- ReactJS: A library created by Facebook. It is used to build UI components that create the user interface of the single-page web application.

Building Application Stack

- Benefits of MERN Stack
- Javascript is the programming language utilized both for client-side and server-side.
- For tech stack with different programming languages, developers need to find out how to interface them together. With the JavaScript stack, developers should be proficient in JavaScript and JSON.
- Using the MERN stack enables developers to build highly efficient web applications.

ExpressJS Routing

- Routing refers to the mechanism for serving the client the content it has asked for.
- It is the most important aspects of your website or web services.
- Routing in Express is simple, flexible, and robust.
- A route specification consists of
 - An HTTP method (GET, POST, etc.),
 - A path specification that matches the request URI,
 - And the route handler.
- The handler is passed in a request object and a response object.
- The request object can be inspected to get the various details of the request, and
- The response object's methods can be used to send the response to the client

Routing – Request Objects

- To access a parameter value we use req.params.
 - req.param(name [, defaultValue])
- req.query: This holds a parsed query string.
 - It's an object with keys: as the query string parameters and
 - Values as the query string values.
 - Multiple keys with the same name are converted to arrays, and
 - Keys with a square bracket notation result in nested objects
 - (e.g., order[status]=closed can be accessed as req.query.order.status).

```
{ key: value }
```

```
?key1=value1&key2=value2&key3=value3
```

Routing – Request Objects

- `req.header`, `req.get(header)`:
 - The `get` method gives access to any header in the request.
 - The `header` property is an object with all headers stored as key-value pairs.
- `req.path`: The path for which the middleware function is invoked; can be any of:
 - A string representing a path.
 - A path pattern.
 - A regular expression pattern to match paths.
 - An array of combinations of any of the above.

Routing – Request Objects

- `req.url`, `req.originalURL`:
 - Contain the complete URL, including the query string.
 - If any middleware modifies the request URL, `originalURL` retains the original URL as it was received, before the modification.
- `req.body`:
 - Contains key-value pairs of data submitted in the request body.
 - By default, it is undefined and is populated when you use body-parsing middleware is installed to read and optionally interpret or parse the body.

Routing – Response Objects

- The res object represents the HTTP response that an Express app sends when it gets an HTTP request.
- res.send(body): Sends the HTTP response.
 - If the body is an object or an array, it is automatically converted to a JSON string with an appropriate content type.

```
res.send('<p>some html</p>')
```

- res.status(code): This sets the response status code.
 - If not set, it is defaulted to 200 OK.
 - One common way of sending an error is by combining the status() and send() methods in a single call like res.status(403).send("Access Denied").

```
res.status(400).send('Bad Request')
```

Routing – Response Objects

- `res.json(object)`: Sends a JSON response
 - This method sends a response (with the correct content-type) that is the parameter converted to a JSON string using `JSON.stringify()`.
 - The parameter can be any JSON type, including object, array, string, Boolean, number, or null.

```
res.json({ user: 'tobi' })
```

- `res.sendFile(path)`:
 - This responds with the contents of the file at path.
 - The content type of the response is guessed using the extension of the file.

```
res.sendFile('/uploads/' + uid + '/' + file)
```


Routing – Response Objects

- for a complete list, please refer to the Request documentation of Express at
- Request Objects: <http://expressjs.com/en/api.html#req>
- Response Objects: <http://expressjs.com/en/api.html#res>

ExpressJS Middleware

- An Express application is a series of middleware function calls.
- Router is nothing but a middleware function.
- Middleware functions have access to the request and response object (req,res), and the next middleware function in the application's request response cycle.
- The next middleware function is commonly denoted by a variable named next.
- next is the only built-in middleware (other than the router) available as part of Express.

ExpressJS Middleware

- Middleware can be at
 - The application level (i.e., applies to all requests) or
 - The router level (applies to specific request path patterns).
- The Middleware at the application level can be used like this: `app.use(middleware function);`
- The static middleware that knows the location of static files to serve.
- In order to use the same middleware in a route-specific way, you define it as:
 - `app.use('/public', express.static('static'));`
- This would have mounted the static files on the path `/public` and all static files would have to be accessed with the prefix `/public`,
- For e.g.,: `/public/index.html`.

Summary

- Express is a web application framework for Node. It is a server-side or back-end framework, fast, unopinionated, and minimalist web framework for Node.js.
- Use the **npm init** command to create a package.json file for your application.
- The command **npm i express** is used to install express.
- MERN has four open-source components MongoDB, Express, React, and Node.js. These components give an end to end frameworks for developers to work with.
- Using the MERN stack enables developers to build highly efficient web applications.
- Routing refers to the mechanism for serving the client the content it has asked for.
- There are various request and response objects used for interacting with the server.
- An Express application is a series of middleware function calls. Router is nothing but a middleware function.