


# Express Introduction :::



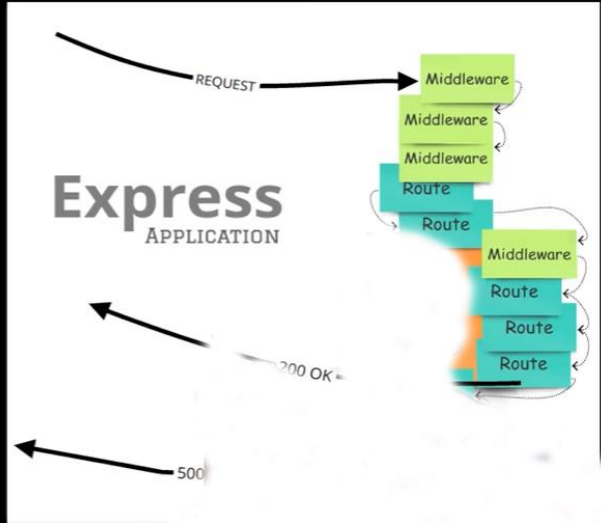
9.1 What is Express.js


EXPRESS Js

1. Express.js is a minimal and flexible web application framework for Node.js.

2. It provides a robust set of features for building single-page, multi-page, and hybrid web applications.

3. Express.js simplifies server-side coding by providing a layer of fundamental web application features.





9.2 Need of Express.js

EXPRESS Js

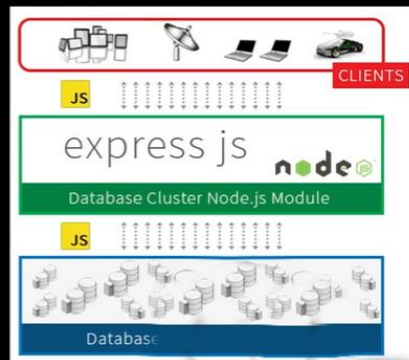
1. Express.js Simplifies Server Creation: Helps in quickly setting up and running a web server without the need for complex coding.

2. Routing Management: Provides a powerful routing mechanism to handle URLs and HTTP methods effectively.

3. Middleware Support: Allows the use of middleware to handle requests, responses, and any middle operations, making code modular and maintainable.

4. API Development: Facilitates easy and efficient creation of RESTful APIs.

5. Community and Plugins: Has a large ecosystem with numerous plugins and extensions, accelerating development time.





## 9.3 Installing Express.js

```
$ npm install express
```

To install Express temporarily and not add it to the dependencies list:

```
$ npm install express --no-save
```

By default with version npm 5.0+, `npm install` adds the module to the `dependencies` list in the `package.json` file; with earlier versions of npm, you must specify the `--save` option explicitly. Then, afterwards, running `npm install` in the app directory will automatically install modules in the dependencies list.



## 9.3 Installing Express.js

```
// Core Modules
const http = require('http');
// External Modules
const express = require('express');

const app = express();

const server = http.createServer(app);

const PORT = 3000;
server.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}/`);
});
```

```
prashantjain@Prashants-Mac-mini node % npm start
```

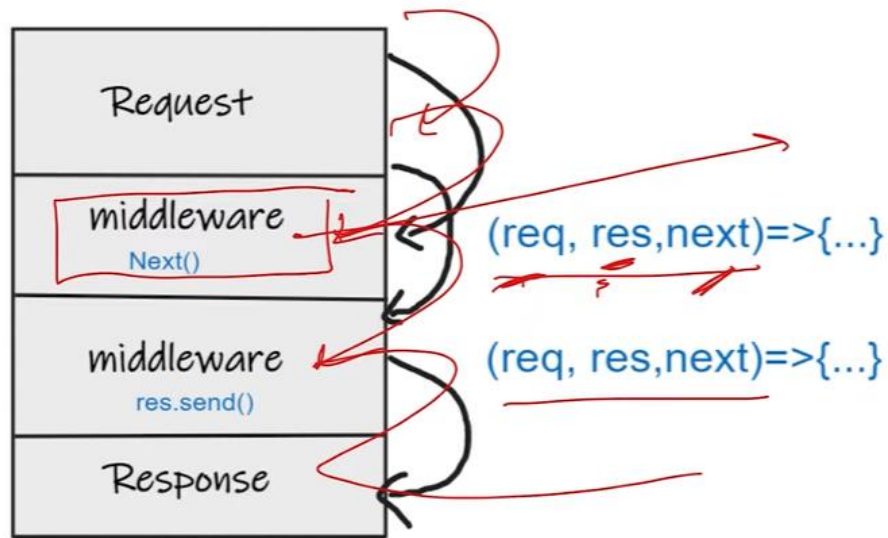
```
> node@1.0.0 start
> nodemon app.js
```

```
[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node app.js`
Server running at http://localhost:3000/
```



## 9.4 Adding Middleware

it is ALL About middleware



## 9.4 Adding Middleware

```
// Adding Middleware
app.use((req, res, next) => {
  console.log("First Middleware", req.url, req.method);
  next();
});

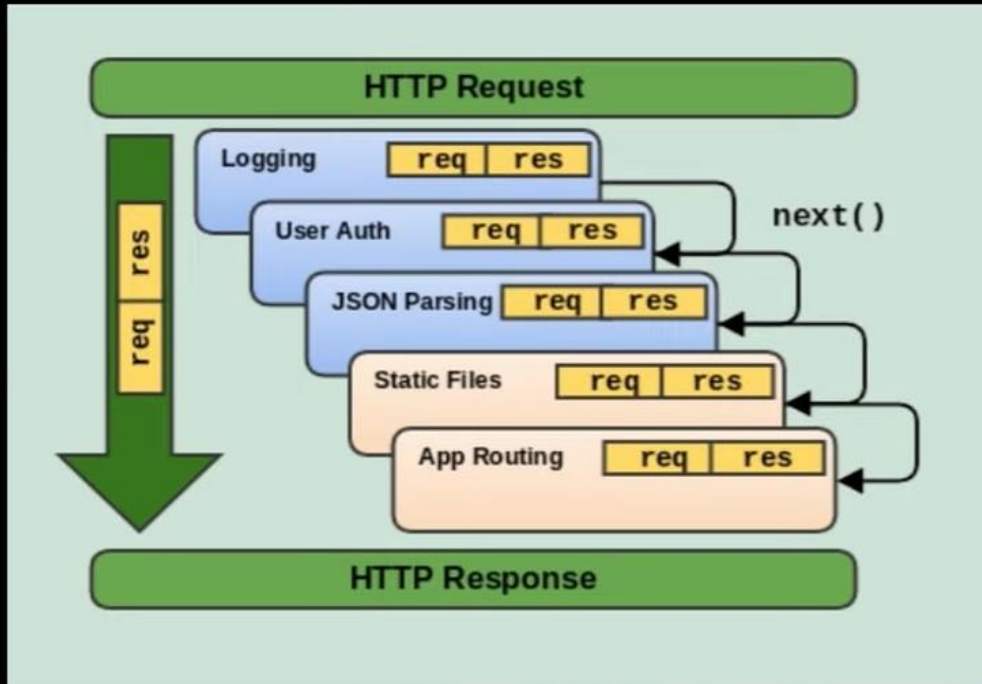
app.use((req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
});

const server = http.createServer(app);
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Server running at http://localhost:3000/
First Middleware / GET
Second Middleware / GET
```



## 9.4 Adding Middleware



## 9.5 Sending Response

EXPRESS Js

```
app.use((req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
  res.send('<p>Welcome to Complete Coding</p>');
});
```

localhost:3000

localhost:3000

Welcome to Complete Coding

Response Headers

Header	Value
Connection	keep-alive
Content-Length	33
Content-Type	text/html; charset=utf-8
Date	Wed, 02 Oct 2019 12:00:00 GMT
Etag	W/"21-FoFdU"
Keep-Alive	timeout=5
X-Powered-By	Express



## 9.6 Express DeepDive

EXPRESS Js

expressjs / express

Code Issues 110 Pull requests 66 Discussions Actions Wiki Security 2 Insights

express Public Watch 1695

master 16 Branches 295 Tags Go to file Add file Code

jonchurch remove --bail from test script (#5962) 6340d15 - yesterday 5,980 Commits

.github/workflows	update CI, remove unsupported versions, clean up	3 weeks ago
benchmarks	docs: add documentation for benchmarks	8 months ago
examples	Delete back as a magic string (#5933)	3 weeks ago
lib	Delete back as a magic string (#5933)	3 weeks ago
test	Delete back as a magic string (#5933)	3 weeks ago
.editorconfig	build: Add .editorconfig	3 weeks



## 9.6 Express DeepDive

```
// Core Modules
const http = require('http');
// External Modules
const express = require('express');

const app = express();

// Adding Middleware
app.use((req, res, next) => {
  console.log("First Middleware", req.url, req.method);
  next();
});

app.use((req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
  res.send('<p>Welcome to Complete Coding</p>');
});

const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}/`);
});
```





## 9.7 Handling Routes

EXPRESS Js

### `app.use([path,] callback [, callback...])`

Mounts the specified [middleware](#) function or functions at the specified path: the middleware function is executed when the base of the requested path matches `path`.

#### Arguments

Argument	Description	Default
<code>path</code>	The path for which the middleware function is invoked; can be any of: <ul style="list-style-type: none"><li>• A string representing a path.</li><li>• A path pattern.</li><li>• A regular expression pattern to match paths.</li><li>• An array of combinations of any of the above.</li></ul> For examples, see <a href="#">Path examples</a> .	<code>'/'</code> (root)



## 9.7 Handling Routes

EXPRESS Js

```
const app = express();
app.use('/', (req, res, next) => {
  console.log("Hello World");
  //res.send('<p>Welcome to Submit Details Page</p>');
  next();
});

app.use('/submit-details', (req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
  res.send('<p>Welcome to Submit Details Page</p>');
});

app.use('/', (req, res, next) => {
  console.log("Second Middleware", req.url, req.method);
  res.send('<p>Welcome to Complete Coding</p>');
});
```

1. Order matters
2. Can not call `next()` after `send()`
3. `"/"` matches everything
4. Calling `res.send` implicitly calls `res.end()`



## 9.7 Handling Routes

EXPRESS Js

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

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

```
app.listen(3000);
```

HTTP method for which the middleware function applies.

Path (route) for which the middleware function applies.

The middleware function.

Callback argument to the middleware function, called "next" by convention.

HTTP response argument to the middleware function.

HTTP request argument to the middleware function.



## 9.7 Handling Routes

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

```
app.post('/', (req, res, next) => {
```

```
  console.log("Hello World");
```

```
  //res.send('<p>Welcome to Submit Details Page</p>');
```

```
  next();
```

```
});
```

```
app.use('/submit-details', (req, res, next) => {
```

```
  console.log("Second Middleware", req.url, req.method);
```

```
  res.send('<p>Welcome to Submit Details Page</p>');
```

```
});
```

```
app.use('/', (req, res, next) => {
```

```
  console.log("Second Middleware", req.url, req.method);
```

```
  res.send('<p>Welcome to Complete Coding</p>');
```

```
});
```