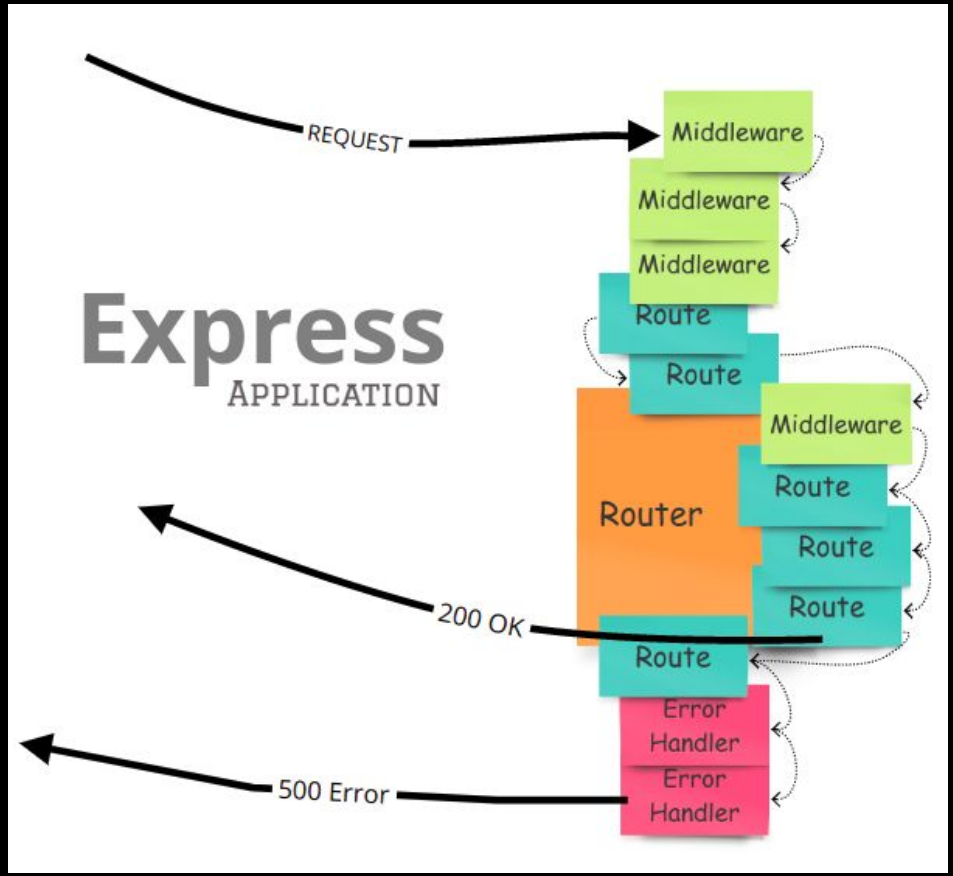
9. Starting with Express.js

1. What is Express.js
2. Need of Express.js
3. Installing Express.js
4. Adding Middleware
5. Sending Response
6. Express DeepDive
7. Handling Routes

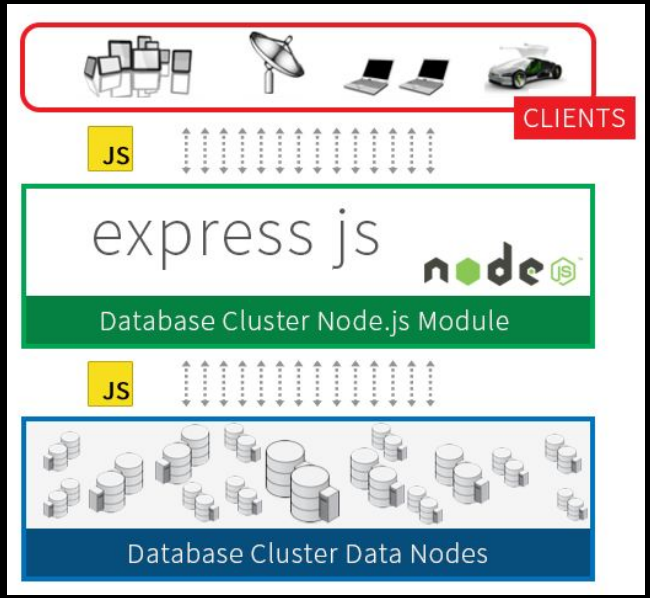
9.1 What is Express.js

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

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
```

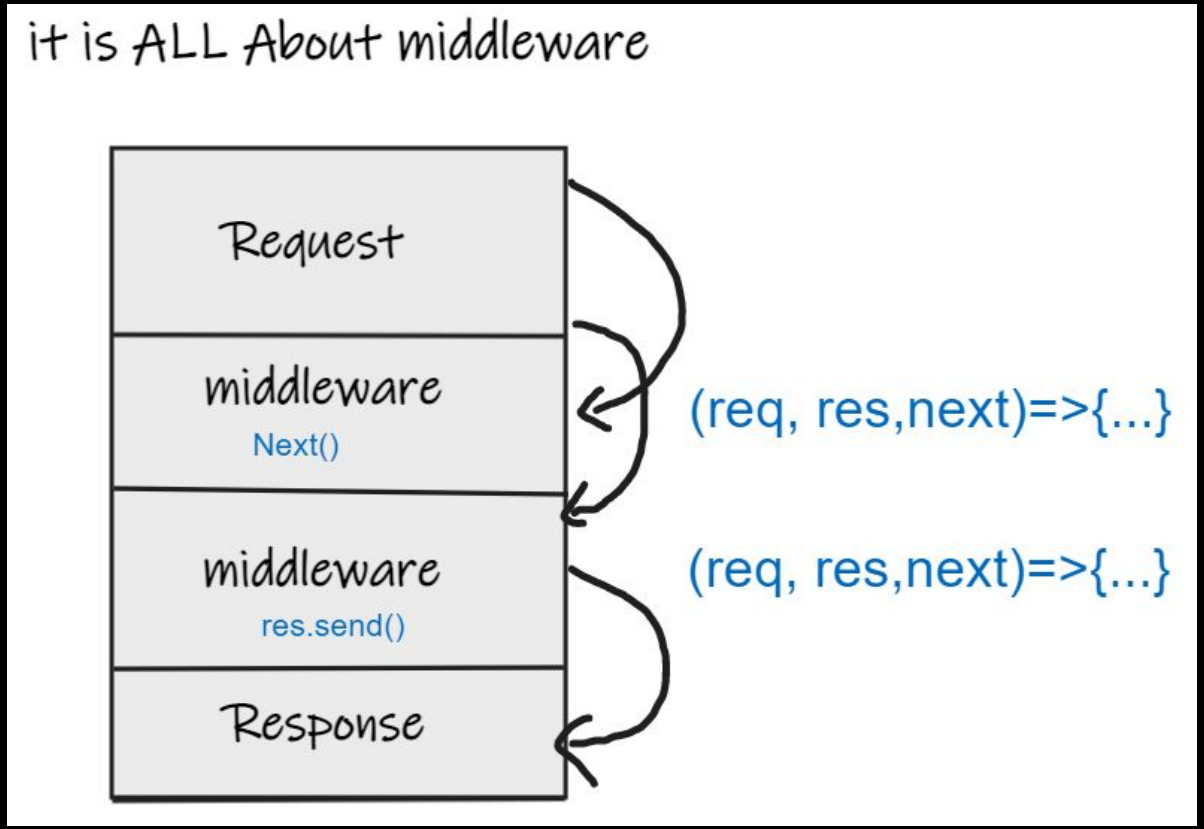
> By default with version npm 5.0+, `npm install` adds the module to the `dependencies` list in the `package.json` file; with earlier versions of npm, you must specify the `--save` option explicitly. Then, afterwards, running `npm install` in the app directory will automatically install modules in the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



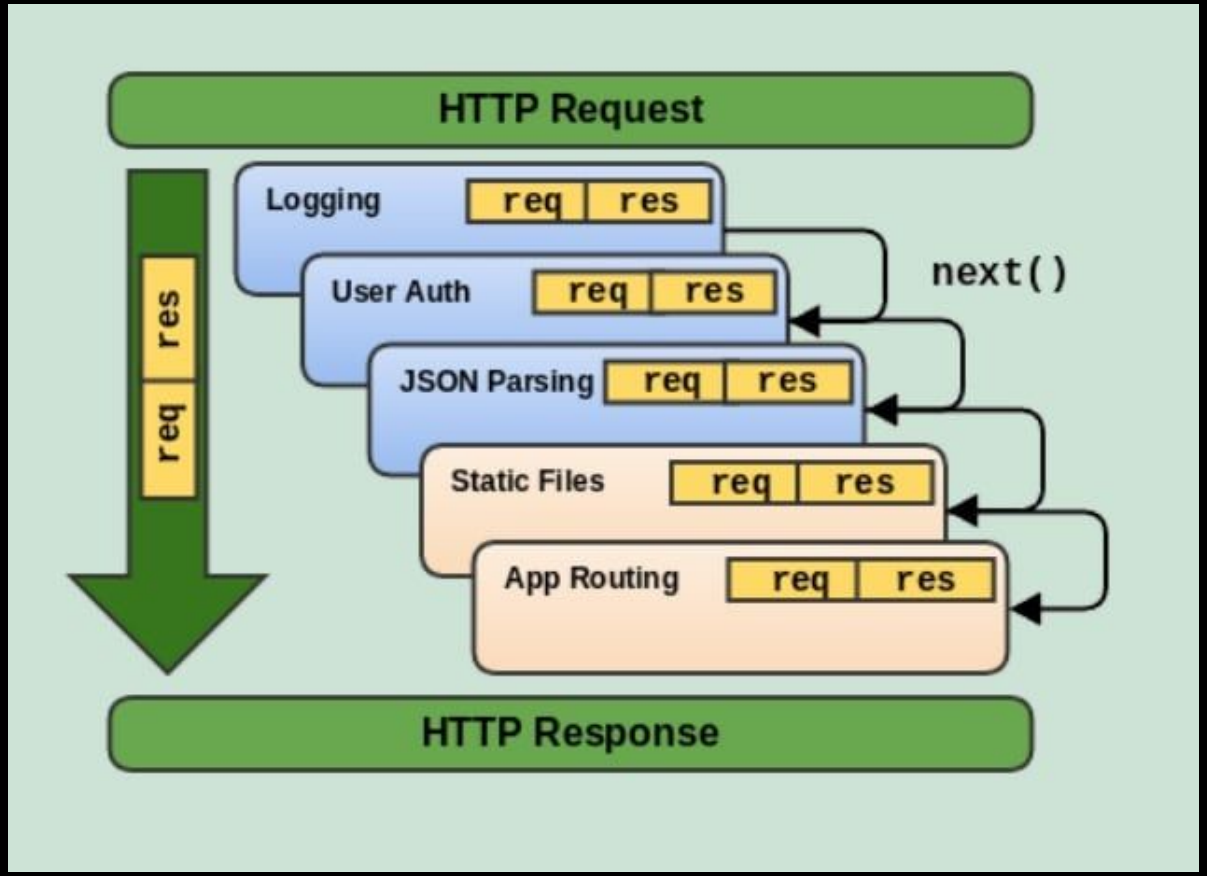
9.4 Adding Middleware

EXPRESS Js

```
// 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
```

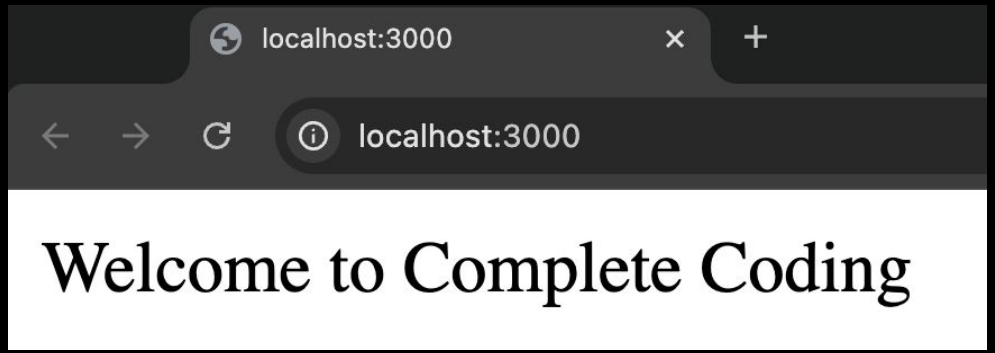
9.4 Adding Middleware

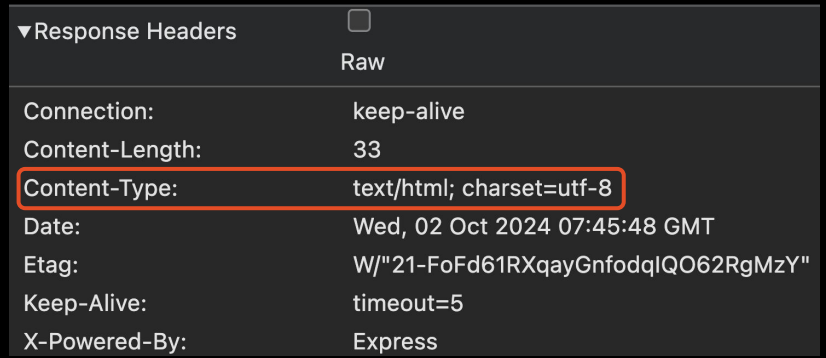


9.5 Sending Response

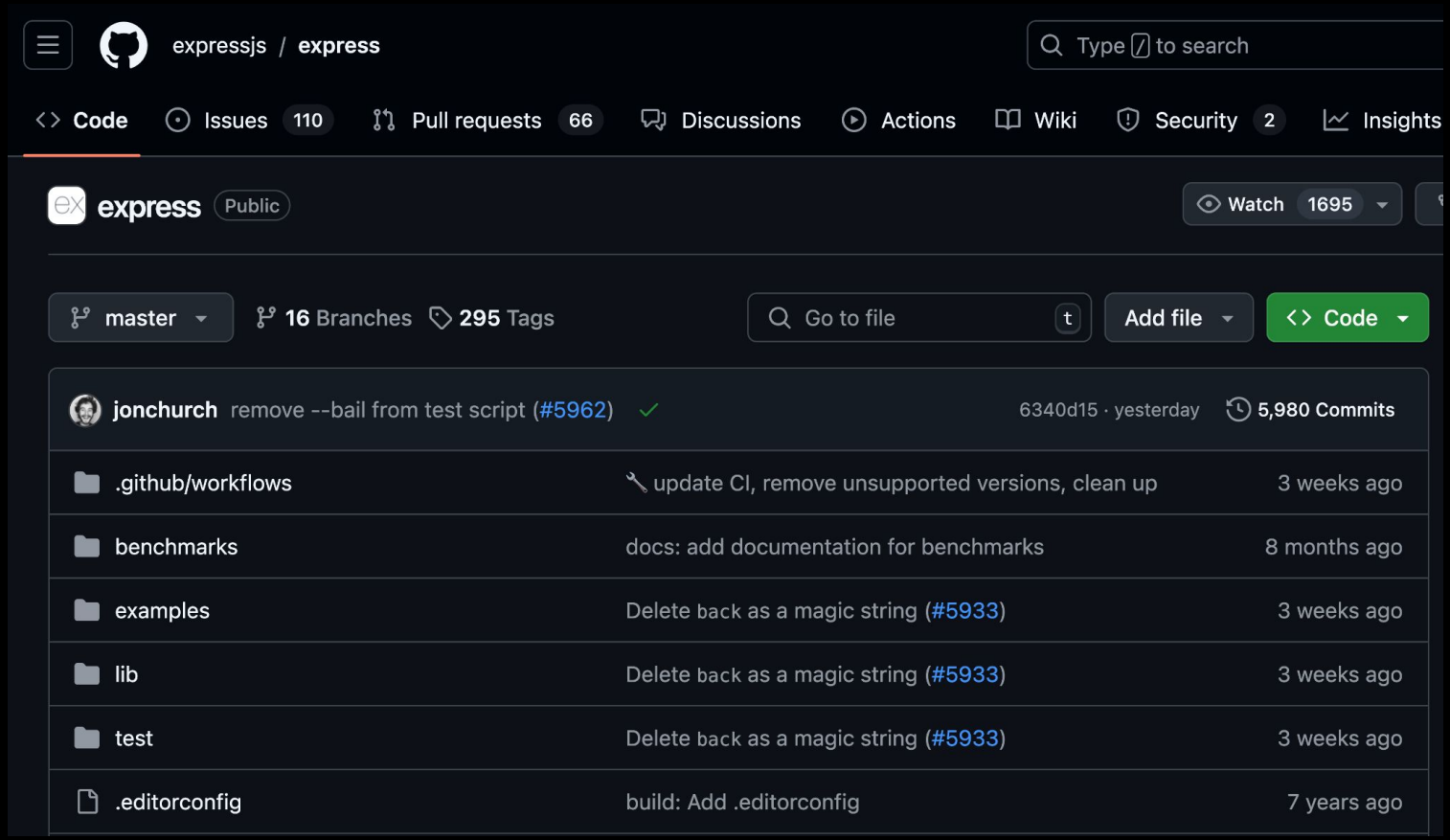
EXPRESS Js

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





9.6 Express DeepDive



9.6 Express DeepDive

```
101   /**
102    * Send a response.
103    *
104    * Examples:
105    *
106    *     res.send(Buffer.from('wahoo'));
107    *     res.send({ some: 'json' });
108    *     res.send('<p>some html</p>');
109    *
110    * @param {string|number|boolean|object|Buffer} body
111    * @public
112    */
113
114   res.send = function send(body) {
115     var chunk = body;
116     var encoding;
117     var req = this.req;
118     var type;
119
120     // settings
121     var app = this.app;
122
```

```
// string defaulting to html
case 'string':
  if (!this.get('Content-Type')) {
    this.type('html');
  }
  break;
case 'boolean':
```

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

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
`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. For examples, see Path examples.	'/' (root path)

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()