

```

1  /*
2      ===== Node.js Basics =====
3
4      1) How Does The Web Work (Refresher)?
5      2) Creating a Node.js Server
6      3) Using Node Core Modules
7      4) Working with Requests & Responses (Basics)
8      5) Asynchronous Code & the Event Loop
9
10 */
11
12 /*
13     ===== How the web Works =====
14
15     1) USER / Client (Browser)
16     2) enter domain Name http://mypage.com
17     3) Requests
18     4) To server <= your code (Node.js, PHP, Asp.NET...);
19     5) server connect to database
20     6) After server work done
21     7) server send response (HTML, Json...)to browser(client/ User)
22
23     work done by HTTP, HTTPS
24
25     => Hyper Text Transfer Protocol (HTTP)
26     A protocol for Transferring Data which is understood by Browser and server
27
28     => Hyper text Transfer Protocol Secure
29     HTTP + Data Encryption (during Transmission)
30
31 */
32
33 /*
34     ===== Creating a Node.js Server =====
35
36 */
37
38
39
40 // require() take file path to import file
41
42 const http = require('http');
43
44 const server = http.createServer((req, res)=>{
45     console.log(req);
46 });
47
48
49 server.listen(3000);
50
51
52 /*
53     ===== Node.js Program LifeCycle =====
54
55     1) RUN node app.js
56     2) start read the start Script
57     3) parse Code, Register Variables & Functions
58     4) Event Loop (The Node Application) +> Keeps on running as long as there are event listeners Registered
59
60 */
61
62
63 // ***** Get Request to server WHO and see url, method, headers info*****
64
65 const http = require('http');
66
67 const server = http.createServer((req, res)=>{
68
69     // see some who requested to this node server...
70     console.log(req.url, req.method, req.headers);
71
72 });
73
74 server.listen(3000);
75
76

```

```

77 // ***** Depend on the Request Server send the response to the client or user *****
78
79 const http = require('http');
80
81 const server = http.createServer((req, res)=>{
82
83     console.log(req.url, req.method, req.headers);
84
85     // setHeader("context-Type is default header", 'document type is')
86     res.setHeader('Content-Type', 'text/html');
87     res.write('<html>');
88     res.write('<head> <title> My First Page</title></head>');
89     res.write('<body> <h1>Hello from my Node.js Server! </h1> </body>');
90     res.write('</html>');
91     res.end();
92     // After end we can write it will show error
93
94 });
95
96 server.listen(3000);
97
98
99 // ===== Routing Requests =====
100
101 const http = require('http');
102
103 const fs = require('fs');
104
105
106 const server = http.createServer((req, res) => {
107
108     const url = req.url;
109     const method = req.method;
110     if(url === '/'){
111         res.write('<html>');
112         res.write('<head> <title> Eneter Message </title> </head>');
113         res.write('<body><form action="/message" method="POST"><input type="text" name="message"/><button>click</button></form></body>');
114         res.write('</html>');
115         return res.end();
116     }
117     if(url === '/message' && method === 'POST') {
118
119         fs.writeFileSync('message.txt', 'DUMMY');
120         res.statusCode = 302;
121         res.setHeader('Location', '/');
122         return res.end();
123     }
124 }
125
126 });
127
128 server.listen(3000);
129
130 /*
131     ===== Parsing Request Streams & Buffers =====
132
133     Example: incoming Request
134
135     Stream => Req_body_part_1 => Req_body_part_2 => Fully Parsed
136
137     // ***** Something we have to Focus *****
138
139 */
140
141 // For understanding code
142 const server = http.createServer((req, res)=>{
143
144     if(url === '/message' && method === 'POST'){
145
146         // when we get some data from post by user fill in the form
147         // That time we have to receive the data and convert and use to do something
148         // async way applying on that
149
150         const body = [];
151         req.on('data', (chunk)=>{
152             console.log(chunk);
153             body.push(chunk);
154         });
155         req.on('end', ()=>{
156             const parsedBody = body.join('');
157             console.log(parsedBody);
158             // Here we can do something with the data
159             // For example we can save it to a file or database
160             // or we can send it back to the client
161             res.writeHead(200, {'Content-Type': 'text/plain'});
162             res.end(parsedBody);
163         });
164     }
165 }
166
167 server.listen(3000);

```

```

153     body.push(chunk);
154   });
155   // After incoming data is received then 'end' run
156   req.on('end', ()=>{
157     const parserBody = Buffer.concat(body).toString();
158     console.log(parserBody);
159     const message = parserBody.split('=')[1];
160     fs.writeFileSync('message.txt', message);
161   });
162
163
164   }
165 });
166
167 // <Buffer 6d 65 73 73 61 67 65 3d 67 6f 6f 64 2b 62 6f 79>
168 // message=good+boy
169
170 // Full Code of store data from user input.....
171
172
173 const http = require('http');
174
175 const fs = require('fs');
176
177
178 const server = http.createServer((req, res) => {
179
180   const url = req.url;
181   const method = req.method;
182   if(url === '/'){
183     res.write('<html>');
184     res.write('<head> <title> Eneter Message </title> </head>');
185     res.write('<body><form action="/message" method="POST"><input type="text" name="message"/><button>click</button></form></body>');
186     res.write('</html>');
187     return res.end();
188   }
189   if(url === '/message' && method === 'POST') {
190
191     const body = [];
192
193     req.on('data', (chunk) => {
194       console.log(chunk);
195       body.push(chunk);
196     });
197
198
199     req.on('end', ()=>{
200       const parserBody = Buffer.concat(body).toString();
201       console.log(parserBody);
202       const message = parserBody.split('=')[1];
203       fs.writeFileSync('message.txt', message);
204     });
205
206     res.statusCode = 302;
207     res.setHeader('Location', '/');
208     return res.end();
209   }
210 }
211
212 });
213
214 server.listen(3000);
215 /* ***** OUTPUT in console log *****
216 <Buffer 6d 65 73 73 61 67 65 3d 67 6f 6f 64 2b 62 6f 79>
217 message=good+boy
218
219 */
220
221
222
223 // ===== using route.js and app.js =====
224
225 // ***** app.js *****
226
227
228 const http = require('http');
229

```

```

230 const route = require('./route');
231
232 const server = http.createServer(route);
233
234 server.listen(3000);
235
236 // ***** END *****
237
238 // ***** route.js *****
239
240
241 const fs = require('fs');
242
243 const routeHandler = (req, res) => {
244
245     const url = req.url;
246     const method = req.method;
247
248     if (url === '/') {
249         res.write('<html>');
250         res.write('<head> <title> Eneter Message </title> </head>');
251         res.write('<body><form action="/message" method="POST"><input type="text" name="message"/><button>click</button></form></body>');
252         res.write('</html>');
253         return res.end();
254     }
255     if (url === '/message' && method === 'POST') {
256
257         const body = [];
258
259         req.on('data', (chunk) => {
260             console.log(chunk);
261             body.push(chunk);
262
263         });
264
265         req.on('end', () => {
266             const parsedBody = Buffer.concat(body).toString();
267             console.log(parsedBody);
268             const message = parsedBody.split('=')[1];
269             fs.writeFileSync('message.txt', message);
270         });
271
272         res.statusCode = 302;
273         res.setHeader('Location', '/');
274         return res.end();
275     }
276 }
277
278 }
279
280 // only function export
281
282 module.exports = routeHandler;
283
284 // object expor from route file
285
286 module.exports = {
287     route: routeHandler,
288     someText: 'some hard text',
289 };
290
291 // another way of pass objec by dot
292 module.exports.rotue = routeHandler;
293 module.exports.someText = 'some hard text';
294
295 // only export systent also allow
296 exports.route = routeHandler;
297
298
299 // ***** END *****

```