Setup:

1. Install node , check if node install by command ‘node -v’
2. Install vscode / webstrom
3. Npm – node package manager..
4. Npm init create a package.json script..
5. .json file.
6. Dependency,
7. Npm start script.
8. Npm install , node modules.
9. Install express , mongoose, dotenv,nodemon,

Front-end = ClientSide- has GUI,

Back-end= ServerSide- code, Database acces, API’s etc.

Node JS webserver:

1. Client , Webserver.
2. Request, Response [1 req – 1 res]
3. Port in webserver. [Request on a port]
4. NodeJs – A runtime Environment. [Express as a server]
5. Endpoints [a path to req] , Endpoints = Server URL + ENDPOINTS path
   1. [www.sitename.com/user](http://www.sitename.com/user) here, [www.google.com](http://www.google.com) = url , /user = endpoint path
6. After give a req to endpoint url, a res comes back as JSON data
7. API = Application programming Interface is a bunch of “public/private” endpoints
8. HTTP req packet:
   1. Request Line[urls],
   2. Header[user agent{browser name}, cookies, etc ],
   3. Body[…datas]
9. HTTP Req types:
   1. GET – get datas
   2. POST – submit/insert data
   3. PUT – replace/update data(changes full source)
   4. PATCH – replace/update data(changes a part of source)
   5. DELETE – Delete data
10. HTTP Response packet:
    1. Status line [200- OK, 404 – file not found etc.…]
    2. Headers [server/content info, {server: express, content-type:html} etc.]
    3. Body[requested data return here… ]
11. HTTP Response Status:
    1. 2XX = Success [200:OK, 201:Created, 202:Accepted]
    2. 3XX = Redirections [301:Moved Permanently, 302:Found]
    3. 4XX = Client Error [400:Bad req, 401:Unauthorized, 403:Forbidden, 404:Not Found,405:Method Not Allowed]
    4. 5XX = Server Error [500:Internal Server Error, 502:Bad Gateway ]

ExpressJS:

1. Express: main server object(HTTP server)
2. Middleware: methods to process a req object.
3. Request: Object – methods and propertied to get HTTP request info.
4. Response: Object – methods and properties to get HTTP response info.
5. Router: Manages path/endpoints in the server

Static hosting:

HTML/CSS/JS/Images directly from server to frontend without manipulation/programming.[Publicly available , anyone can access] Static hosting needs middleware[ex: func express.static(‘public’) , so public directory’s file will be available in server, and anyone can access it]. Static hosting/Files are not API it lodes static files for frontend only. {Index.html : is a file,static server look at first. }

Environment Veriable(.env):

1. Process.env: process level veriable which maintained by OS
   1. Protecting secrets.
   2. Assign dynamic system resource.
   3. Provide flexibility to migrate one OS to another, DB URL’s etc.

Request Object:

1. Methods: GET, POST, PUT, DELETE.
2. Headers: Cookie, Auth, Token.
3. Body: Data.
4. URL: Query String, Params.

Sending Data in Request Objects:

1. URL: Query String[ex: <https://localhost:8080/home?name=billi&age=50>; here, name = billi and age = 50 is query string][Don’t use sensitive data here. ]
2. URL: URL Params: [ex: <https://localhost:8080/home/billi/50>; here, billi and 50 are parameter][Don’t use sensitive data here. ]
3. BODY: data in json format.[hidden part, we can use sensitive data i.e password here.] In HTML Form : GET send data in URL, POST send data in body. Bodyparser require to get body from req.

Middlewares: (Modifies req and sent a resp)

1. After getting req, it sent modified req to another middleware or sent a resp.
2. Server.use() work as a middleware.
3. Usefull middlewares:
   1. Body-parser: req body data parsing.
   2. express.static: static file hosting.
   3. Morgan: logging server req/resp data.
   4. session: session management.

Server Session:

1. store a data for specific user,
2. session has an ID, that store on cookie

Passport JS:

1. Signup /Login flow

Hashing:

1. Hash Algo to secure pass
2. Salt added extra sec to hash