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 is JavaScript Object Notation file, that has a {key: value } pairs. .
6. Dependency:project depends on those libraries, that used to work for a specific task ex: express , work as a http server handler ,
7. Npm start a script.
8. Npm install “dependency/module name” will install the module and it appear in node modules folder.
9. We need to Install express , mongoose(MongoDB), dotenv(.env) and nodemon(it run continuously and display changes on save),

Front-end = ClientSide- has GUI,

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

Node JS webserver:

1. Client:a computer that provides front-end / for people’s use .A websites GUI that we see is client side.
2. server. - backend, that run the website’s serversite connection
   1. In general, all of the machines on the Internet can be categorized as two types: servers and clients. Those machines that provide services (like Web servers or FTP servers) to other machines are **servers**. And the machines that are used to connect to those services are **clients**
3. Request(client sent server a req and server gives feedback as a response.), [for one req – there is 1 res]
4. Port is a accesspoint in webserver. [Request goes on a specific port]
5. NodeJs – A runtime Environment. [Express as a server module]
6. **Endpoints:** (where we perform a req )
   1. [a path to req] -> Endpoints = Server URL + ENDPOINTS path
   2. [www.sitename.com/user](http://www.sitename.com/user) is a endpoint. Here, [www.google.com](http://www.google.com/) = url , /user = endpoint path
7. After give a req to endpoint url, a res comes back as JSON data
8. API = Application programming Interface is a bunch of “public/private” endpoints
9. HTTP req packet:
   1. Request Line[urls],
   2. Header[user agent{browser name}, cookies, etc ],
   3. Body[…datas]
10. HTTP Req types:
    1. GET – get datas/pages
    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
11. 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… ]
12. 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, Authentication, Token.
3. Body: Data.
4. URL: Query String, Params(parameter).

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.

MongoDB:

1. mongodb is a noSQL datgabase(Non relational database)
   1. relational database is SQL, MySQL, postgraSQL, etc. those has database, and tables which contain informations. Tables are related each other by a primary key and forign keys.
   2. NoSQL databases are non relational database that has collection (read alternative table), that contains json object {key: value} pairs.
   3. Lets see an example in SQL and MongoDB:
      1. in SQL: let table name “Info”

Table: Info

|  |  |  |  |
| --- | --- | --- | --- |
| \_ID | Student name | Roll | Age |
| 1 | Abdullah | 39 | 20 |
| 2 | Umar | 89 | 20 |

2. in MongoDB: let table name “Info”

{

Info:

{

“\_ID”:”1”,

“Student name”:”Abdullah”,

“Roll”:”39”,

“Age”:”20”

},

{

“\_ID”:”2”,

“Student name”:”Umar”,

“Roll”:”89”,

“Age”:”20”

},

}

------------------------------------------------END-----------------------------------------------