Date / /

Express JS

⇒ It is a free and open-source web application framework for Node.js.
⇒ It is used for designing and building web applications quickly and easily
It is an unopinionated framework
↳ Unopinionated framework is flexible and allows you to do things in whatever way you wish.

Installing

⇒ Download and install VS code and Node.js. Create package.json by npm init.
⇒ Open the terminal and write npm install express --save.
⇒ Express will be installed.

CREATING A HTTP SERVER IN EXPRESS

i) Create a file Index.js
ii) Open the documentation of express and copy the code for Hello World.

⇒ The app responds with "Hello world" for request to the root URL (/).
⇒ For every other path it will respond with 404 not found.
↳ If you would do it in node, it will take a lots of if-else statement.

Thunderclient
→ It allows us to stay in VS code and do testing of APIs.
→ It is a HTTP client extension.

Creating two pages, and rendering.
i) Create index.html and write some content.
ii) In Index.js

```
Const express = require('express')
Const path = require('path')
Const app = express('path')
Const port = 3000

App.get('/', (req, res) => {
        res.send('HelloWorld')})
```

Date / /

```
app.get('/about', (req, res) => {
    res.sendFile(Path.join)(__dirname,
                            'index.html'))
```

↳ App.get lets you define a route

```
App.listen(port, () => {
console.log('App listening at http://
                localhost: ${port}')
    })
```

Result: The above code create two pages one with root url ('/') and other with ('/about').
⇒ One page render 'Hello world' and other will render content of HTML.

Changing Status code
To change status code of /about.

```
app.get('/about', (req, res) => {
 res.sendFile(path.join(__dirname
                        'index.html'))
    res.status(500)})
```

↑ Change to 500

DOMS | Page No.
Date / /

Sending JSON file

- To send JSON file in /about

```
app.get('/about', (req, res) => {
    res.json("Manny" = 34)})
```

* Express automatically sets the content type.

Tip: Install JSON formatter. It make easy to parse large JSON file.

Serving a Folder

- Create a public folder and inside it create index.html

Import path module

```
Const path = require('path')

App.use(express.static(path.json(
        __dirname, "Public")))
```

↳ It is used to mount specific middleware function(s) at the path, which is being specified.

Date / /

Middleware: functions that have access to the request object (req), the response object (res).

Writing our own middleware

```
Const Pikamiddleware = (req, res, next)
    => {Console.log(req) next()}
App.use(Pikamiddleware)
```

Next(): function in Express router which, when invoked, executes the middleware succeeding the current middleware.

Parameters

They are named URL segments that are used to capture the values specified at their position in the URL.

```
Eg. app.get('/Hello/:name', (req, res)
      => {res.send('Hello world' +
                    req.params.name)
    })
```

* Creating Blogexpress (Project)

i) Create a route folder and Blog.js inside it.

ii) Create a template folder and create index2.html → Inside it we are writing. 'This is site page')

iii) Inside static folder, create index2.js.

In index2.js

⇒ Copy the content of old index.js and write.

```
app.use(Express.static(path.join(__dirname, "static")))
                                    ↳ Created folder

app.use('/', require(path.join(__dirname, 'routes/Blog.js')))
                                    ↳ Blog.js in route folder.

});
```

In Blog.js, write

```
Const express = require('express')
Const path = require('path')
Const router = express.router()
res.sendfile(path.join(__dirname, '../templates/index2.html'))
```

↳ As the file is present in outer folder (Not inside routes folder)

Date / /

```
})
module.export = router
```

Result: The content of Index2.html will be rendered in our route url('/'). In our case 'This is Bike page' will be printed.

Now, serving two pages, we want.

i) '/' → Index2.html is rendered

ii) /Blog → Bloghome.html gets rendered

* Create a Bloghome.html and, we are writing 'This is Blog page'.

* Create Blogs.js

```
blogs = [{
        title: "Python"
        Content: "Let python"
    }
    :
]]
module.export = Blogs
```

→ Array of object

In Blog.js

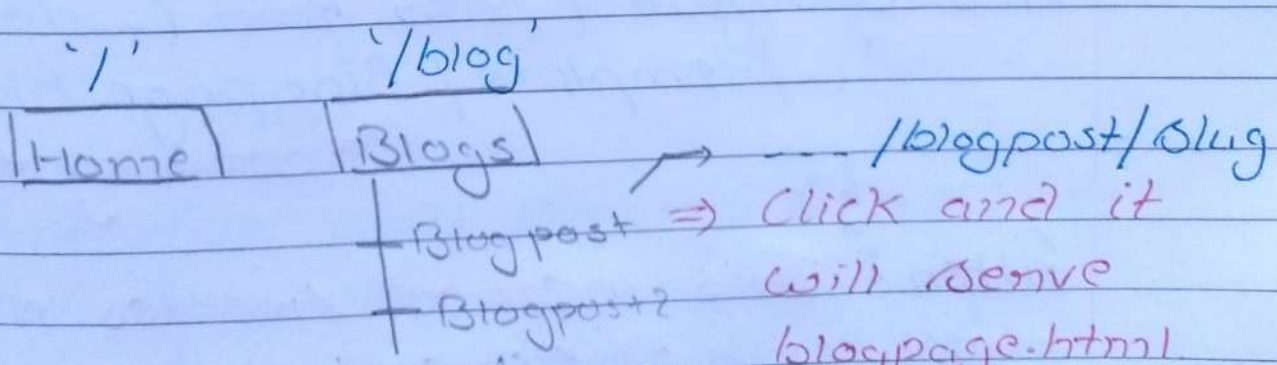
```
const blogs = require('../data/blogs')

router.get('/blog', (req, res) => {
res.sendfile(path.join(__dirname,
    '../templates/~~index2~~.html'))})
```

BlogHome

```
blogs.foreach(e => {
 console.log(e.title)});
```

Result: BlogHome.html gets served in '/blog' and content of Blogs.js gets printed in the terminal.

⇒ Now what if we want to show our blogpost in /blog instead of bloghome.html. Something like this



To do so, Open blogs.js

```
blog = [{
        title: "Python"
        Content: "Learn Py"
        Slug : " Learning with P.C "}
```

More objects ;
in this array can't be added.

* Create Blogpage.html and write something, we are writing 'content here'

In Blog.js

```
router.get('/blogpost/:slug', (req, res)
  => {myBlog = blogs.filter((e) => {

return e.slug = req.params.slug})
    (console.log (myBlog)
res.sendfile (path.join (__dirname,
      '../template/blogpage.html)
                    ↳ serving this file
})
```

filter: This method creates a new array with all elements that pass the test implemented by the provided function.

Express Handlebars: Organise data

Pug: Template engine for the node and the browser.

Handlebar: Helps in JS templating

But we are going to use express handlebars.

Express.handlebars: It is an npm package.

To install:

```
npm install express-handlebars
```

Structure:

- Index.js
- Views
  - Home.handlebars
  - Layout folders
    - Main.handlebars

In main.handlebar

⇒ This file will be rendered in the browser.

⇒ Copy the pre-define code of main. handlebar.

{{{body}}}

↳ Content of Home.handlebar will be served here. And main.handlebar will be served in browser

Create Home.handlebars.

⇒ Write Something, like This is H.hb

In index.js, copy

```
var exphbs = require('express-handlebars');
app.engine('handlebars', exphbs());
app.set('viewengine', 'handlebars');
```

In Blog.js

```
router.get('/', (req, res) => {
  res.render('../home'); })
```

⇒ Your content of Home.handlebars will be printed in browser

Now we want to do

Home → Blog
↳ render content of Blog/Home.handlebar

Page No.
Date / /

Get Bootstrap in main.handlebars.
Remember to use href = "/blog"
href = "/"

In Blog Home.handlebars (create in views)

* use Each helper, like, → serve blog

```
{{#each blogs}}
    {{this.title}}
{{/each}}
```

In blog.js

```
router.get('/blog', (req, res) => {
    res.render('blog Home', { blogs:
                              blogs});
```

⇒ Your content will be rendered.

* You can combine link with the rendered text, with help of <a href, and can extend the structure.

#All the best