Express

07 October 2023 02:57 PM

Express is all JS.
Express is an NodeJS Framework.

To use Nodejs in Server side, we use Express

Let us first install node.
Now do npm i express.

Express is a server side framework

We do npm init.
Fill all the details.
We can also do npm init -y

NodeJS is used to execute JS in outside browser.

We do **npm intall express --save**

We can generate node modules again by **npm i**

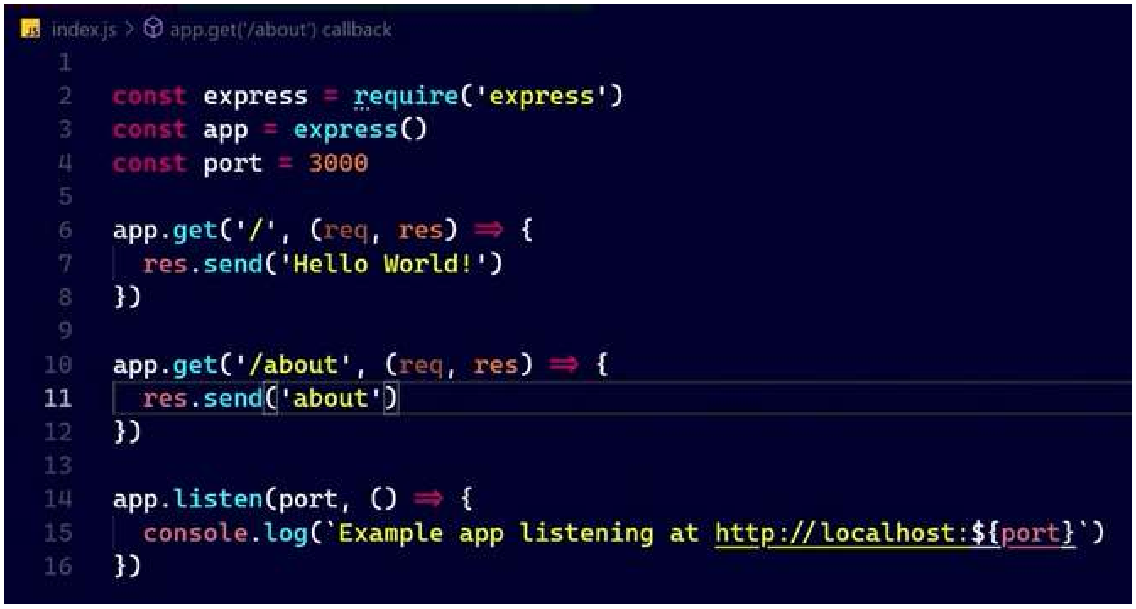
We can also do **npm uninstall express** to uninstall express

Let us make a index.js

Let us make a server using express.

If we want to show something in "/about" or "/contact" we have to do if-else in Http server made using NodeJS.
We need to send statusCOde ourself, Header, content-type everything is to be given by us in NOdeJS.

In Express we do not need to do this.

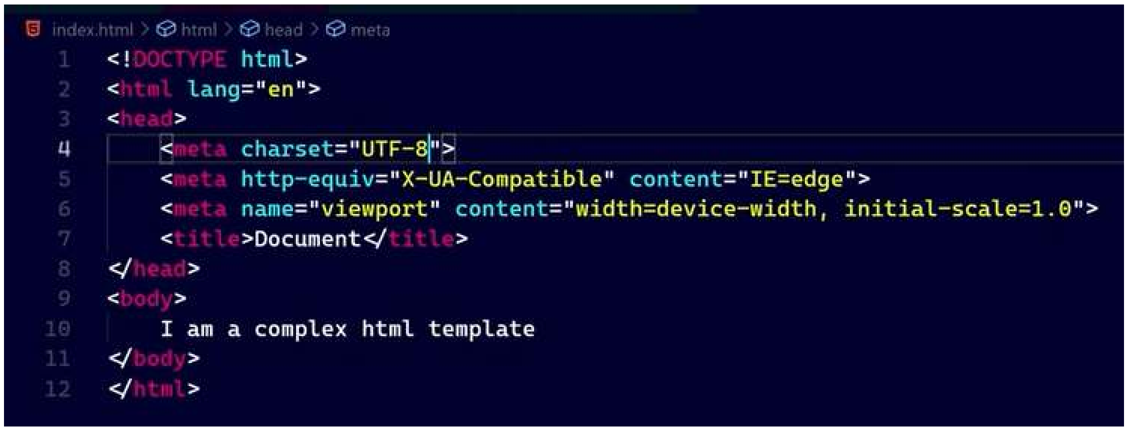


index.js > app.get('/about') callback

```javascript

const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.get('/about', (req, res) => {
  res.send('about')
})

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

Let us install Thunderclient now.
We can also use Postman.

Let us say we have index.html where we have some template

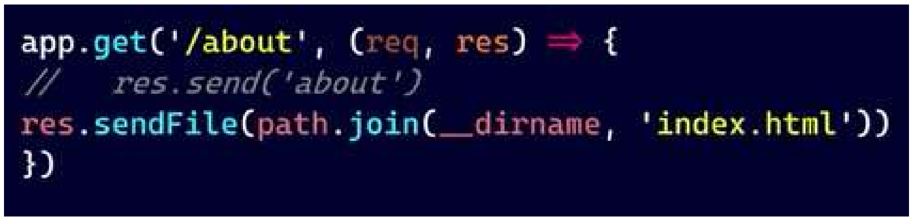


index.html > html > head > meta

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    I am a complex html template
</body>
</html>
```

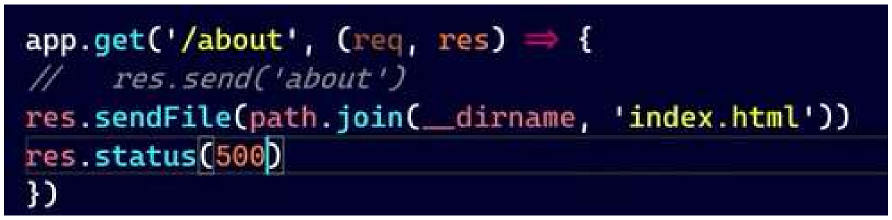
To show this, we use res.sendFile

We can use path module to give path pf index.html
Let us use nodemon, npm i -g nodemon

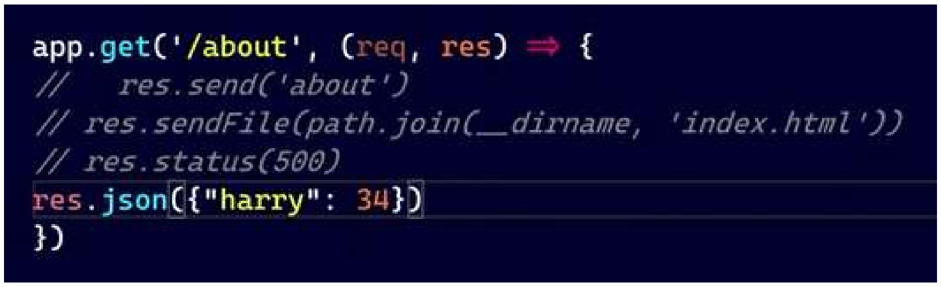


```javascript
app.get('/about', (req, res) => {
//   res.send('about')
res.sendFile(path.join(__dirname, 'index.html'))
})
```

We can also put Bootstrap in index.html and use it.
We can also send status code.
We can also make custom 404 not found page and chain it using res.status(500) or also send JSON like res.json({name: "lokesh"}

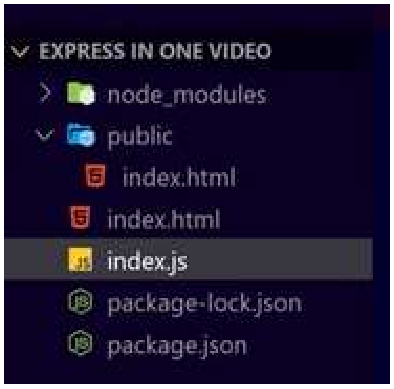


```
app.get('/about', (req, res) => {
//   res.send('about')
res.sendFile(path.join(__dirname, 'index.html'))
res.status(500)
})
```



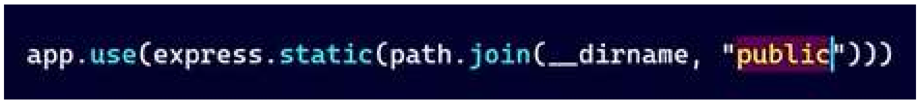
```
app.get('/about', (req, res) => {
//   res.send('about')
// res.sendFile(path.join(__dirname, 'index.html'))
// res.status(500)
res.json({"harry": 34})
})
```

We will test our websites in ThunderClient now.
Whatever static data we want to show to the public, we put it inside public folder



```
EXPRESS IN ONE VIDEO
  > node_modules
  v public
      index.html
    index.html
    index.js
    package-lock.json
    package.json
```

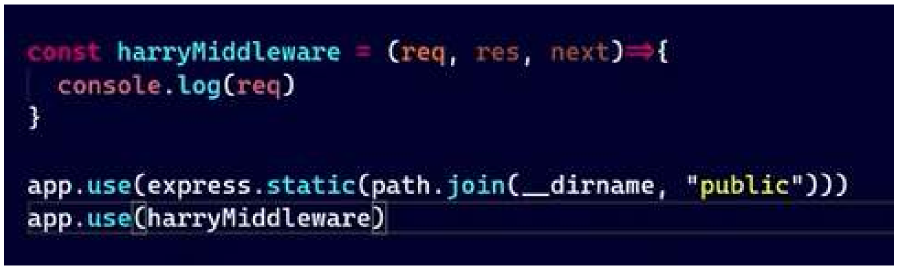
To serve static files we use express middleware
app.use()



```
app.use(express.static(path.join(__dirname, "public")))
```

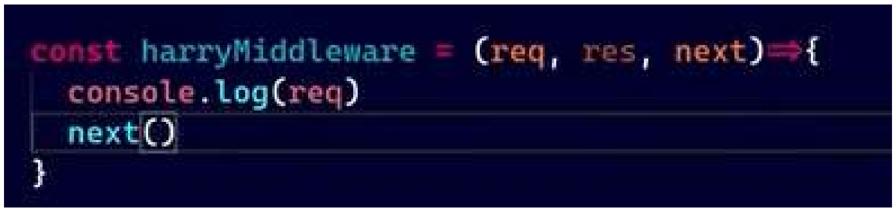
Now our "/" send this index.html inside public folder.

Let us write our own middleware
Middleware has control of request and response both
To use middleware we make use of (app.use())



```
const harryMiddleware = (req, res, next)=>{
  console.log(req)
}

app.use(express.static(path.join(__dirname, "public")))
app.use(harryMiddleware)
```

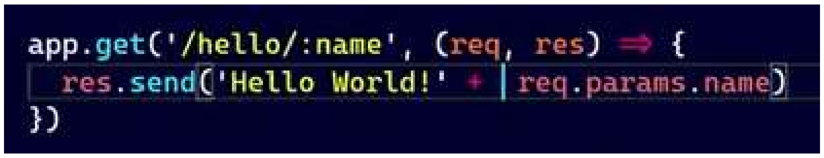
If we have more than one middleware and we want our next middleware to get called on finish of current middleware. We use next()



```
const harryMiddleware = (req, res, next)=>{
  console.log(req)
  next()
}
```

Its very rare to make our own middleware, generally we use in-built middlewares.

Let say we give some parameters in URL, how to use it?



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

If we want to fetch some data using these params, now we can do it easily.

Let say our app gets some data from a store where data is in JSON.

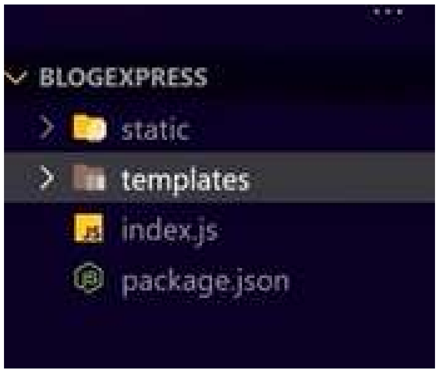
Let us build something like that. Let us build a blog website using it.

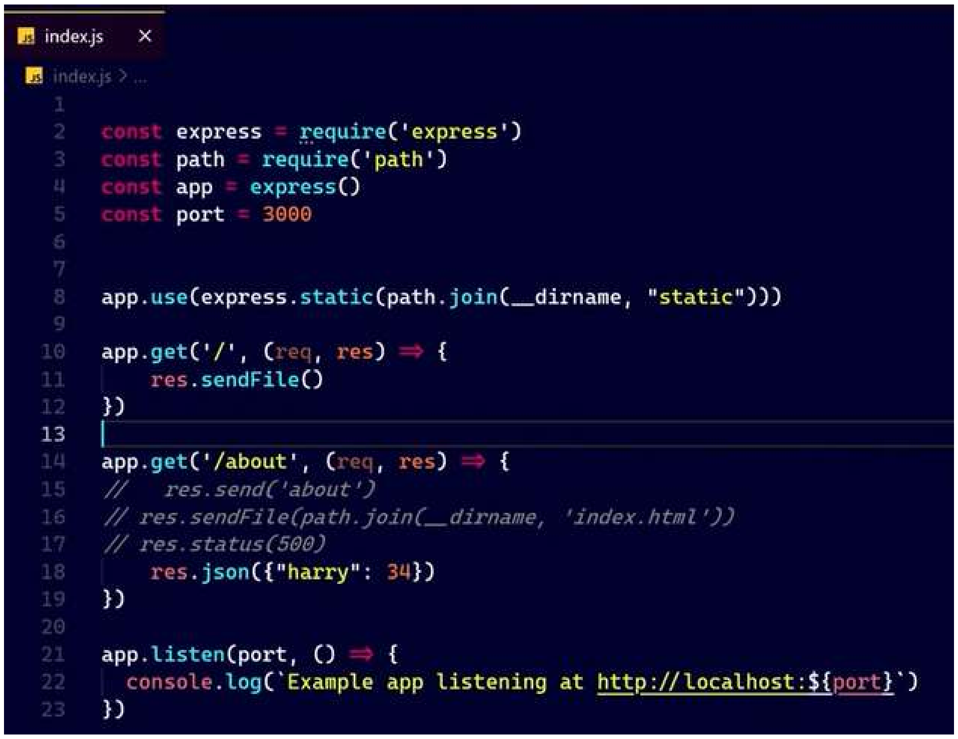
We make a static folder and keep all JS, CSS inside it.

We make some endpoints also.

Where we send some file.

We make a folder templates.



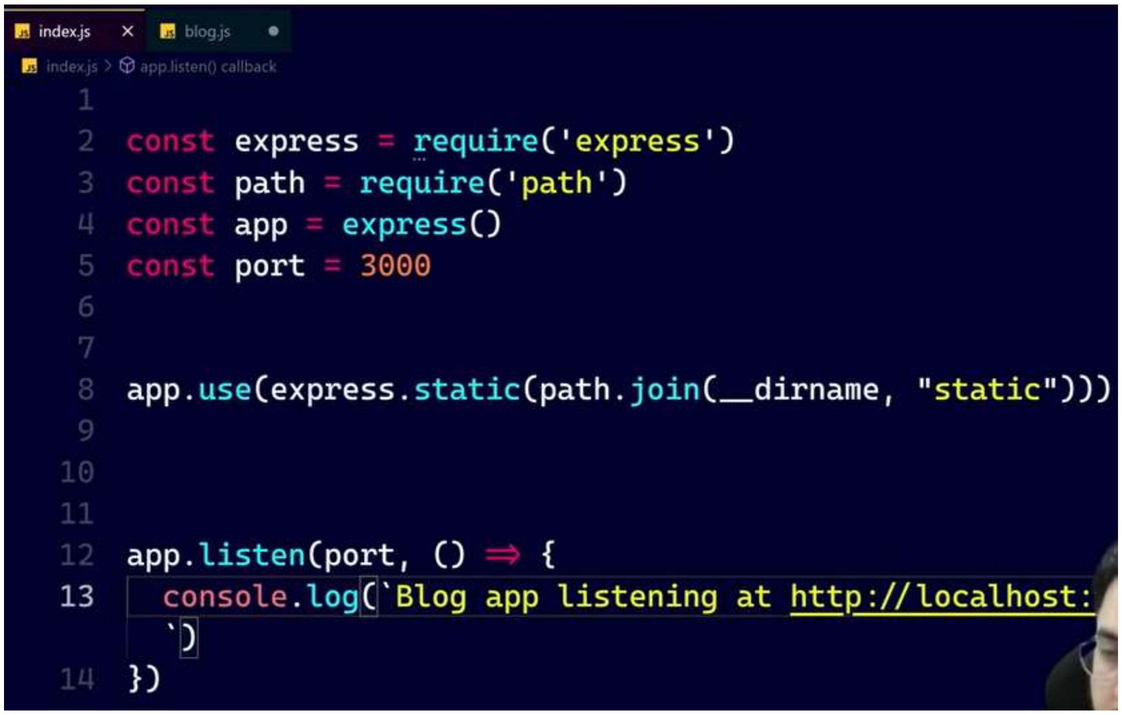

```
const express = require('express')
const path = require('path')
const app = express()
const port = 3000


app.use(express.static(path.join(__dirname, "static")))

app.get('/', (req, res) => {
    res.sendFile()
})

app.get('/about', (req, res) => {
//    res.send('about')
// res.sendFile(path.join(__dirname, 'index.html'))
// res.status(500)
    res.json({"harry": 34})
})

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

Let us seperate our routers in a separate folder routes.

Inside routes -> we make blog.js where we put all our routes related to our blogs

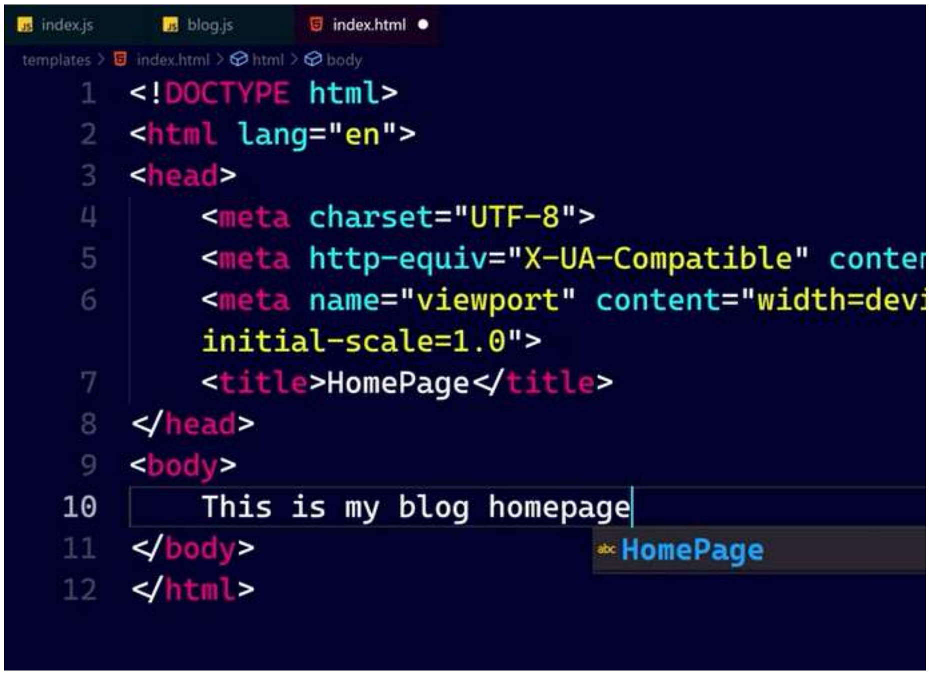
We use router from express.Router

Now our index.js do not contain any routes


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


app.use(express.static(path.join(__dirname, "static")))



app.listen(port, () => {
  console.log(`Blog app listening at http://localhost:
  `)
})
```

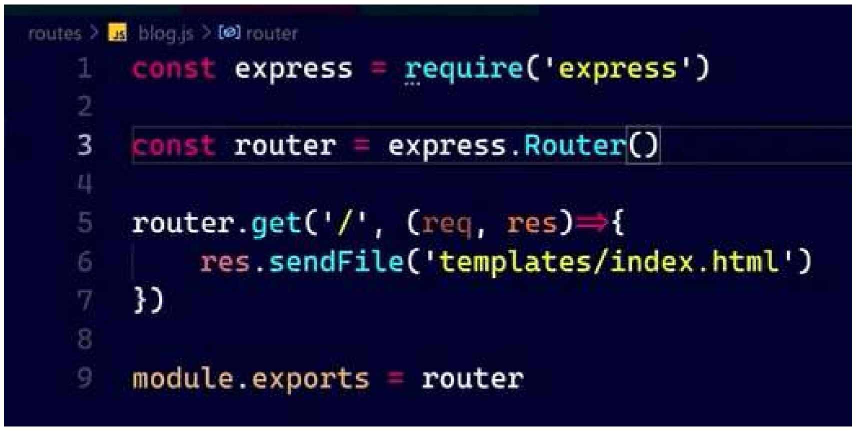
We make a index.html inside templates and make a basic HTML.

Inside index.html



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" conter
    <meta name="viewport" content="width=devi
    initial-scale=1.0">
    <title>HomePage</title>
</head>
<body>
    This is my blog homepage
</body>
</html>
```

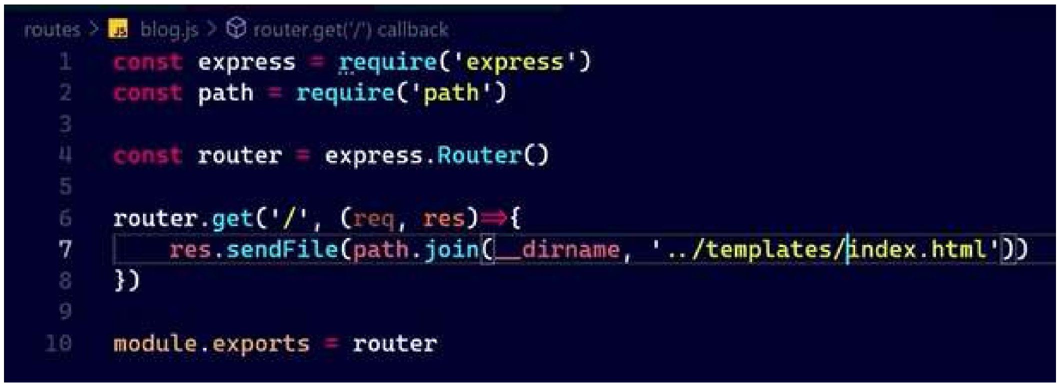
Inside blog.js we sendFile index.html in '/' endpoint
We install nodemon as devDependency.



```js
const express = require('express')

const router = express.Router()

router.get('/', (req, res)=>{
    res.sendFile('templates/index.html')
})

module.exports = router
```

Inside blog.js

We use __dirname to serve our index.html which is inside templates but __dirname gives us path of routes folder, to come out of it we do
Path.join(__dirname, '../templates/index.html')

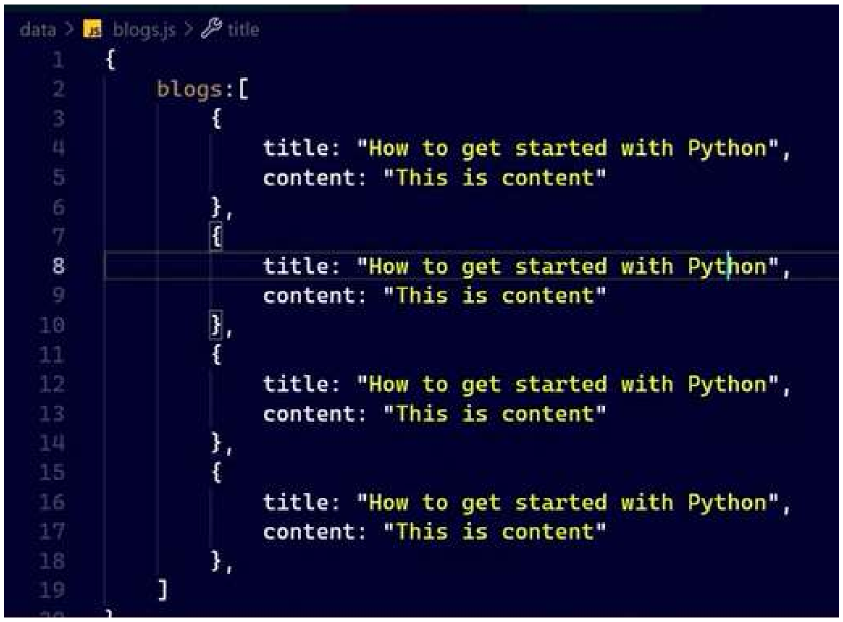


```js
const express = require('express')
const path = require('path')

const router = express.Router()

router.get('/', (req, res)=>{
    res.sendFile(path.join(__dirname, '../templates/index.html'))
})

module.exports = router
```

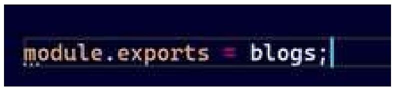
To use these router in index.js we do
App.use(__dirname , require(./routes/blog.js))

Now we make another routes for /blog inside blog.js to display all our blogs

To get all blogs data we make data folder and inside it we make blogs.js
We make a JSON of Array of blogs



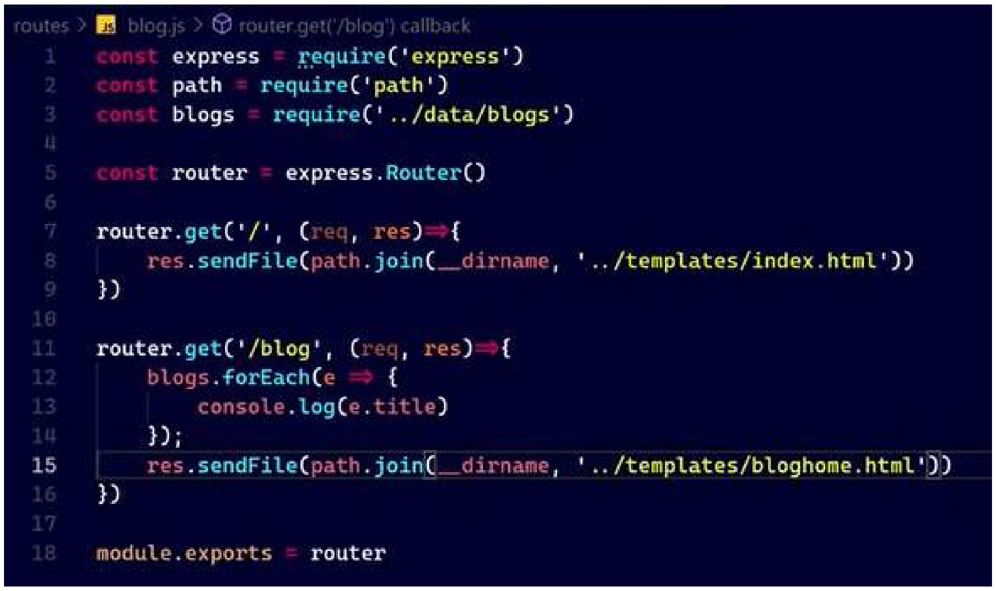
```
data > blogs.js > title
{
    blogs:[
        {
            title: "How to get started with Python",
            content: "This is content"
        },
        {
            title: "How to get started with Python",
            content: "This is content"
        },
        {
            title: "How to get started with Python",
            content: "This is content"
        },
        {
            title: "How to get started with Python",
            content: "This is content"
        },
    ]
```



```
module.exports = blogs;
```

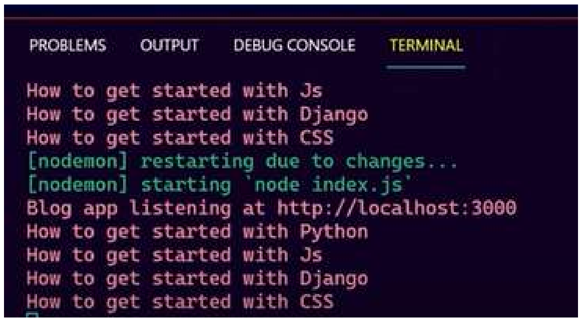
We can read this file using os module

We require this in our blog.js



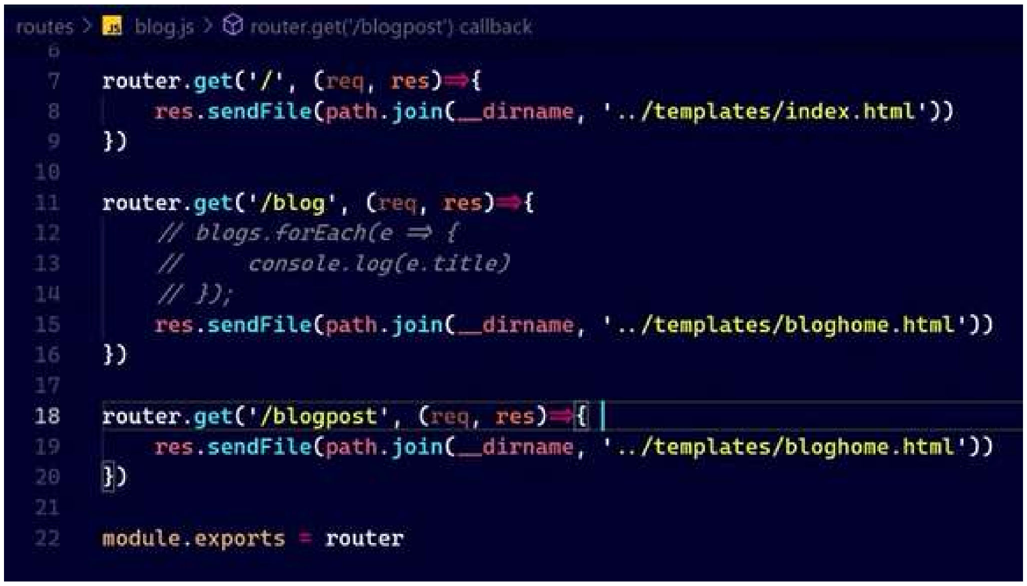
```
routes > blog.js > router.get('/blog') callback
const express = require('express')
const path = require('path')
const blogs = require('../data/blogs')

const router = express.Router()

router.get('/', (req, res)=>{
    res.sendFile(path.join(__dirname, '../templates/index.html'))
})

router.get('/blog', (req, res)=>{
    blogs.forEach(e => {
        console.log(e.title)
    });
    res.sendFile(path.join(__dirname, '../templates/bloghome.html'))
})

module.exports = router
```

Our console looks like



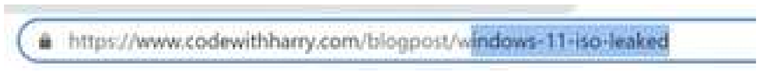
```
How to get started with Js
How to get started with Django
How to get started with CSS
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Blog app listening at http://localhost:3000
How to get started with Python
How to get started with Js
How to get started with Django
How to get started with CSS
```

Now we make blogHome.html to serve these notes.
We make another endpoint /blogpost to display blog posts

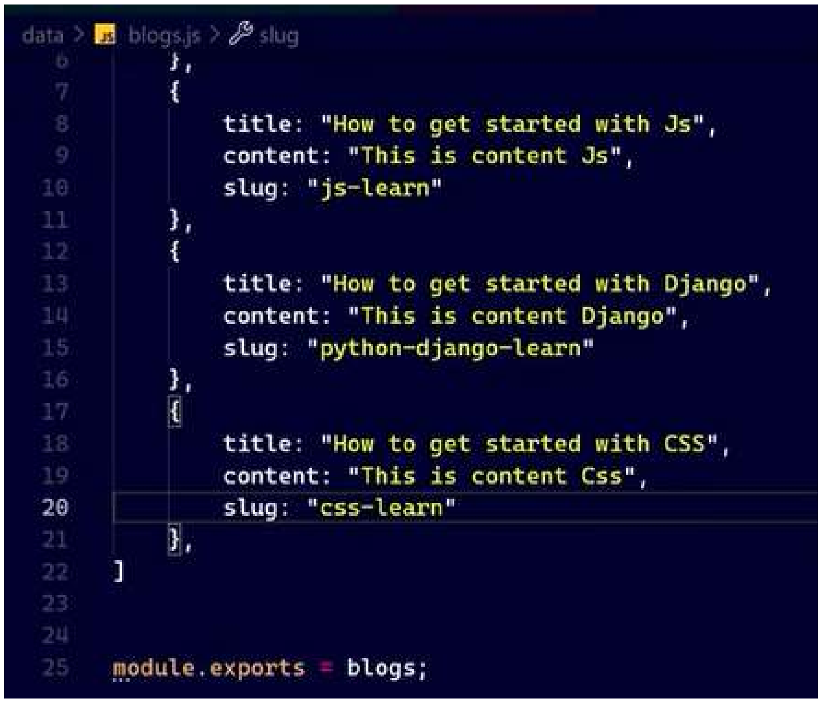


```
router.get('/', (req, res)=>{
    res.sendFile(path.join(__dirname, '../templates/index.html'))
})

router.get('/blog', (req, res)=>{
    // blogs.forEach(e => {
    //     console.log(e.title)
    // });
    res.sendFile(path.join(__dirname, '../templates/bloghome.html'))
})

router.get('/blogpost', (req, res)=>{
    res.sendFile(path.join(__dirname, '../templates/bloghome.html'))
})

module.exports = router
```

Let us add slug also in the data blogs.js
Slug is

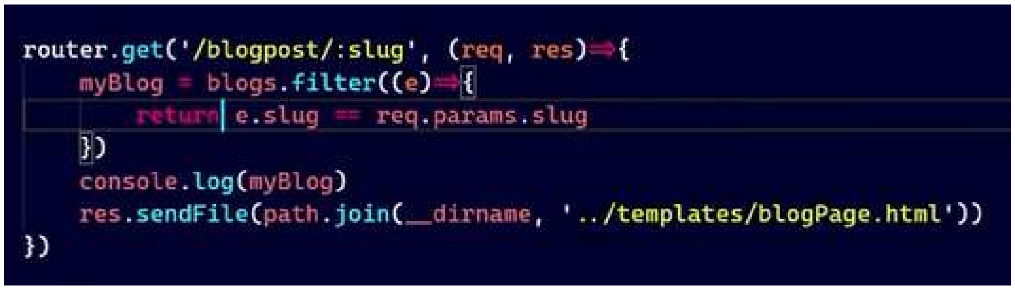


```
https://www.codewithharry.com/blogpost/windows-11-iso-leaked
```

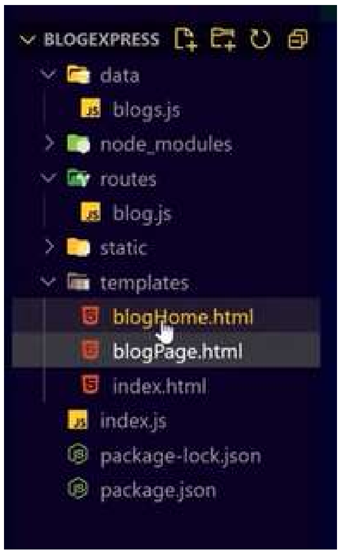

data > blogs.js > slug

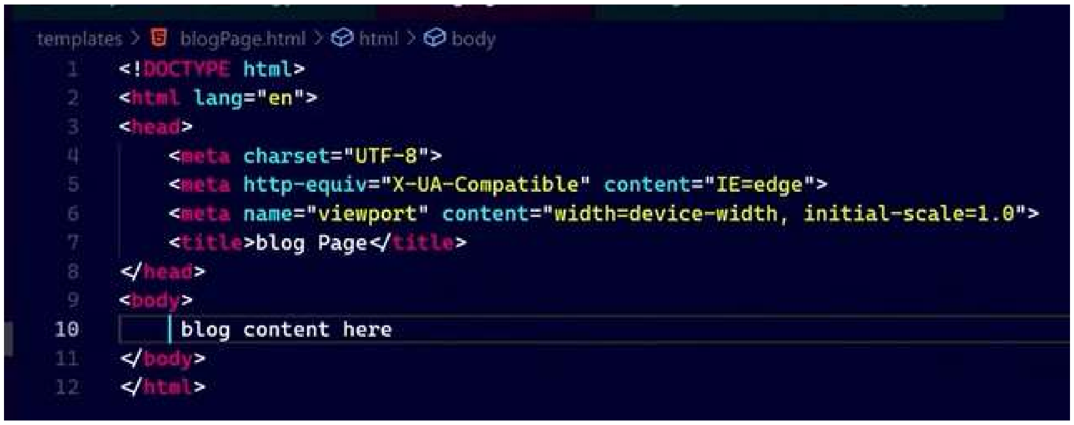
```
    },
    {
        title: "How to get started with Js",
        content: "This is content Js",
        slug: "js-learn"
    },
    {
        title: "How to get started with Django",
        content: "This is content Django",
        slug: "python-django-learn"
    },
    {
        title: "How to get started with CSS",
        content: "This is content Css",
        slug: "css-learn"
    },
]


module.exports = blogs;
```

We make blogpage.html for /blogpost end point



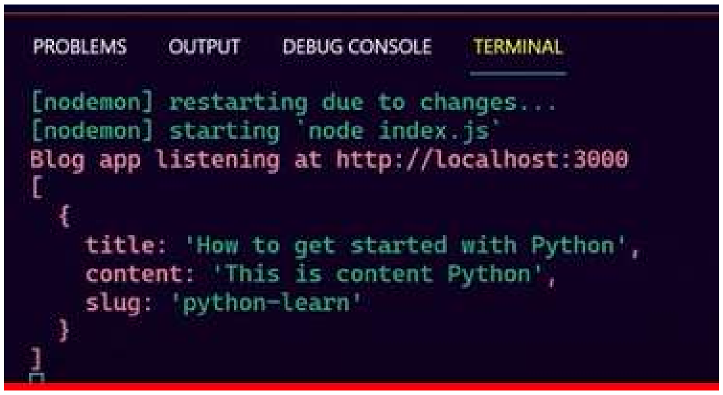
```
router.get('/blogpost/:slug', (req, res)=>{
    myBlog = blogs.filter((e)=>{
        return e.slug == req.params.slug
    })
    console.log(myBlog)
    res.sendFile(path.join(__dirname, '../templates/blogPage.html'))
})
```





templates > blogPage.html > html > body

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>blog Page</title>
</head>
<body>
    blog content here
</body>
</html>
```



```
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Blog app listening at http://localhost:3000
[
  {
    title: 'How to get started with Python',
    content: 'This is content Python',
    slug: 'python-learn'
  }
]
```

Now we have got our blog so we can use this blog.
How to use this blog now?? How to send this data in backend, how to show it in frontend?

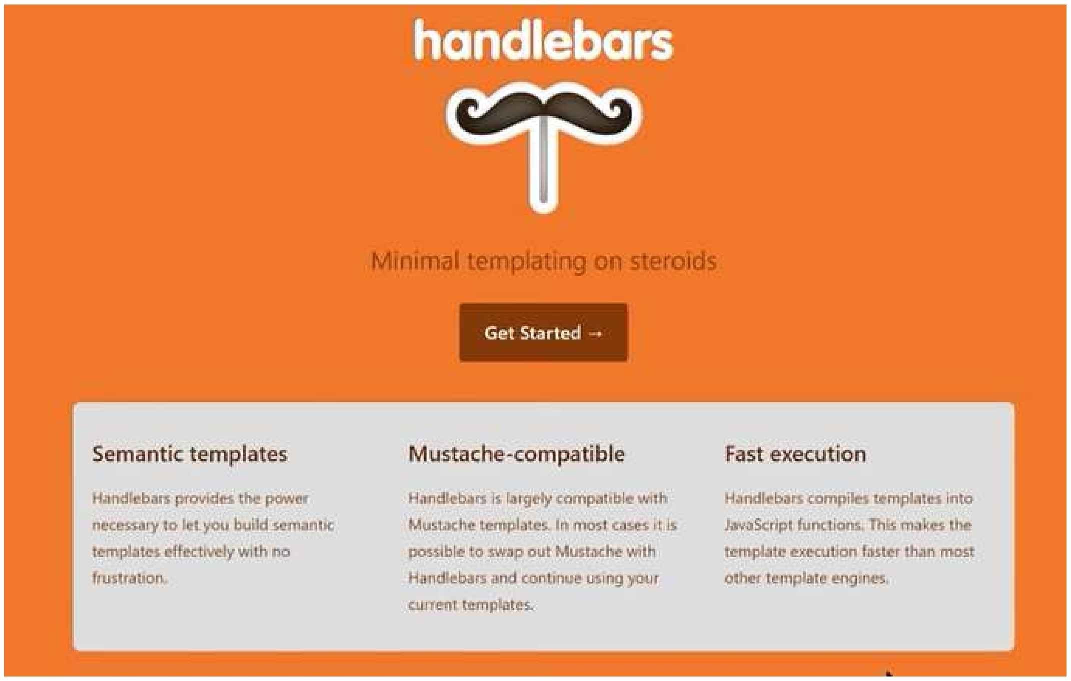
For database we will use mongoose.

We have many templates for express like pug, Handlebars, EJS

We will use express-handleBars

We need template engine to show our data in our frontend.

Express-handlebars is different than handler-bars



Semantic templates

Handlebars provides the power necessary to let you build semantic templates effectively with no frustration.

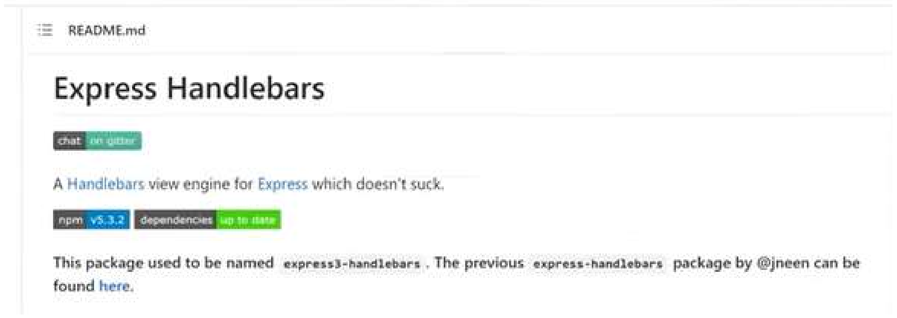
Mustache-compatible

Handlebars is largely compatible with Mustache templates. In most cases it is possible to swap out Mustache with Handlebars and continue using your current templates.

Fast execution

Handlebars compiles templates into JavaScript functions. This makes the template execution faster than most other template engines.

Express-handlebars is a npm package



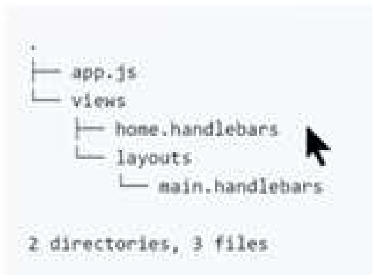
Express Handlebars

A Handlebars view engine for Express which doesn't suck.

This package used to be named `express3-handlebars`. The previous `express-handlebars` package by @jneen can be found here.

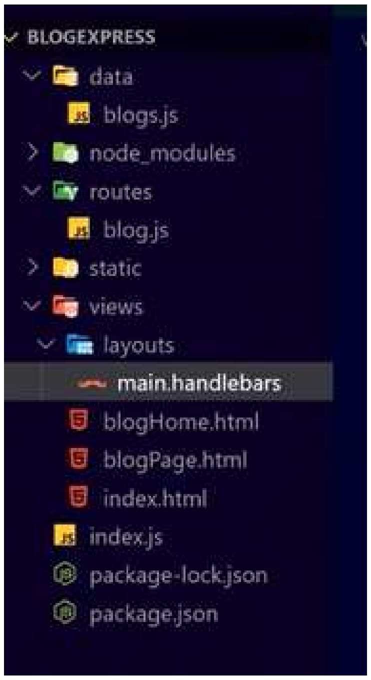
We do npm i express-handlebars

Directory structure of handlesbars is

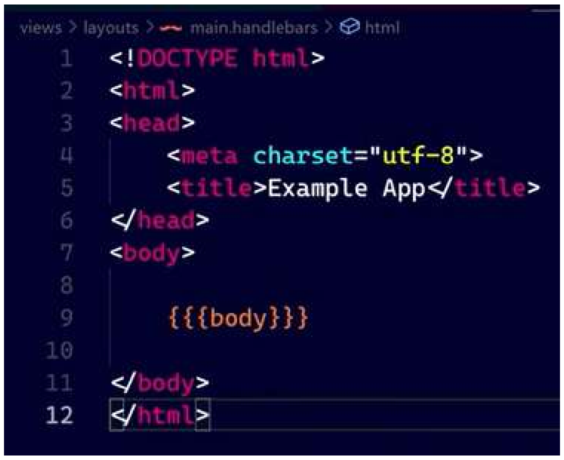


```
.
├── app.js
└── views
    ├── home.handlebars
    └── layouts
        └── main.handlebars

2 directories, 3 files
```

We need to make views folder and we make layout folder.
We make main.handlebars



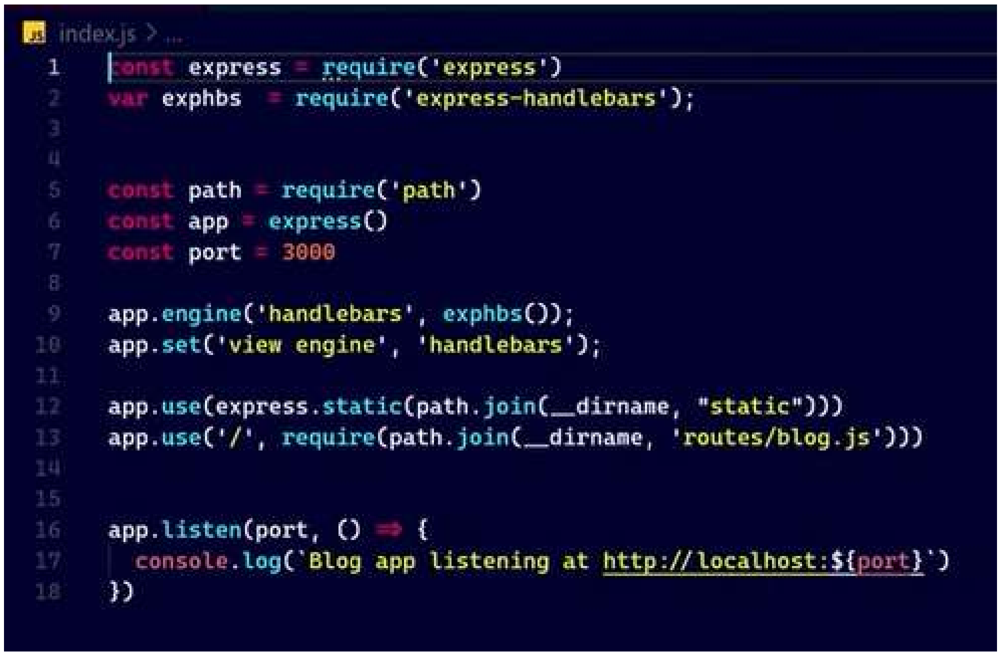
```
BLOGEXPRESS
  data
    blogs.js
  node_modules
  routes
    blog.js
  static
  views
    layouts
      main.handlebars
    blogHome.html
    blogPage.html
    index.html
  index.js
  package-lock.json
  package.json
```



views > layouts > main.handlebars > html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Example App</title>
</head>
<body>

    {{{body}}}

</body>
</html>
```

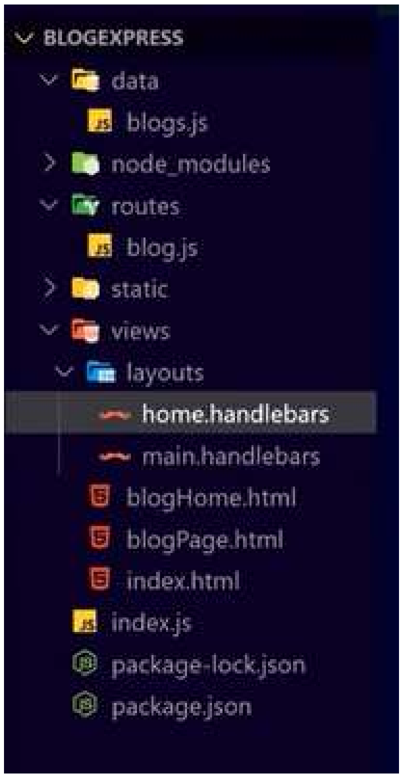
Inisde index.js we need to require it also.



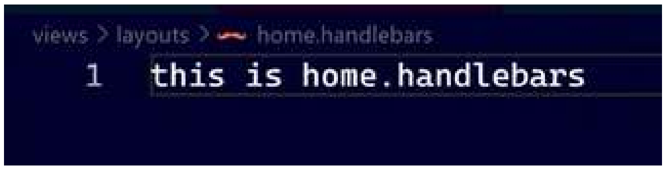
index.js > ...

```javascript
const express = require('express')
var exphbs  = require('express-handlebars');


const path = require('path')
const app = express()
const port = 3000

app.engine('handlebars', exphbs());
app.set('view engine', 'handlebars');

app.use(express.static(path.join(__dirname, "static")))
app.use('/', require(path.join(__dirname, 'routes/blog.js')))


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

To start using handlebars we need to make home.handlebars inside views folder.



```
BLOGEXPRESS
  data
    blogs.js
  node_modules
  routes
    blog.js
  static
  views
    layouts
      home.handlebars
      main.handlebars
    blogHome.html
    blogPage.html
    index.html
  index.js
  package-lock.json
  package.json
```

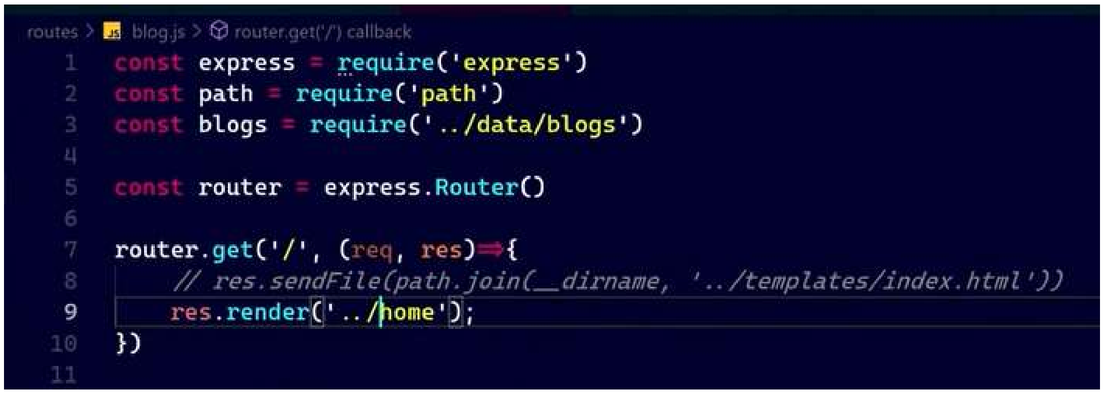


views > layouts > home.handlebars

```
this is home.handlebars
```

Now instead of res.sendFile we will use res.render

In our blog.js

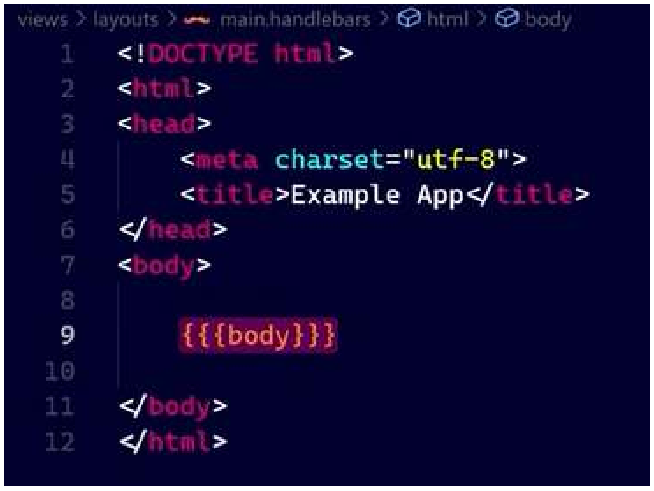


routes > blog.js > router.get('/') callback

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

const router = express.Router()

router.get('/', (req, res)=>{
    // res.sendFile(path.join(__dirname, '../templates/index.html'))
    res.render('../home');
})
```

What we are doing is
While using res.render('../home')

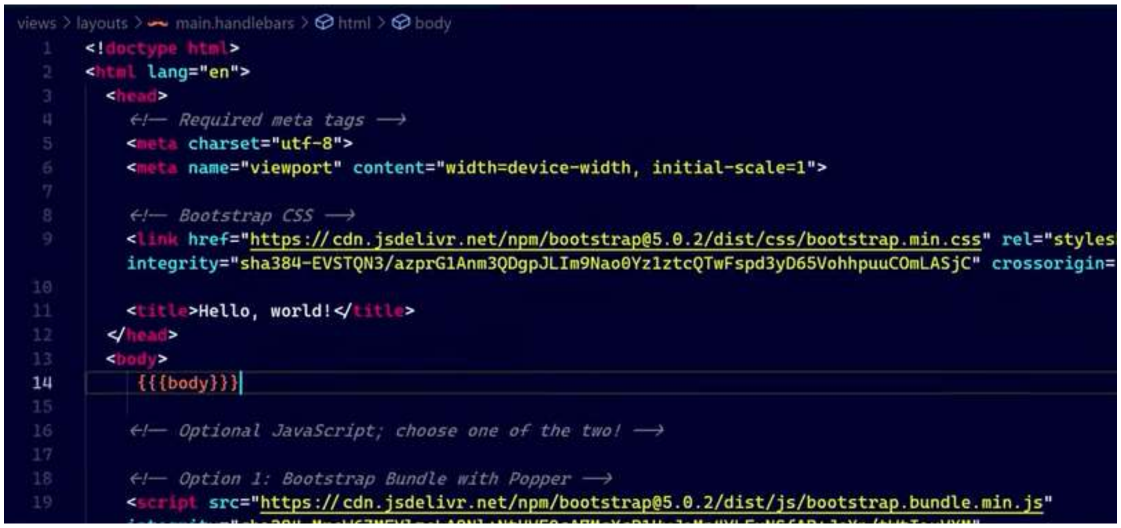
We are saying, jo bhi home.handlebars ke andar hai usko main.handlebars ki {{body}} ke andar daal do baaki same template of main.handlebars hi render kro when we hit '/' endpoint.
So ab hum kuch bhi render kra skte h {{body}} ke andar.



views > layouts > main.handlebars > html > body

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Example App</title>
</head>
<body>

    {{{body}}}

</body>
</html>
```

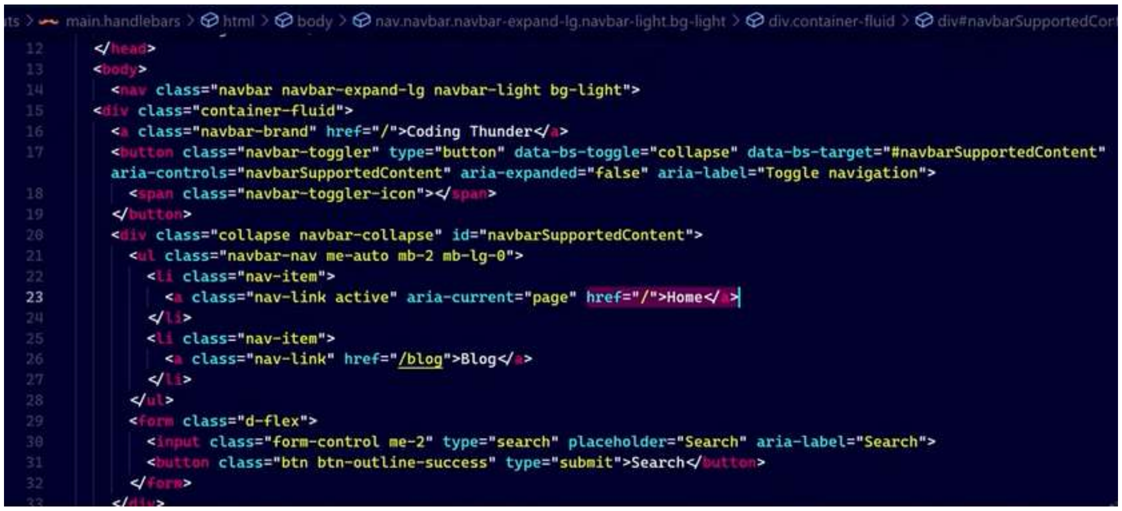
Now let us put some bootstrap in our main.handlebar and home.handlebars



views > layouts > main.handlebars > html > body

```html
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="styles
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin=

    <title>Hello, world!</title>
  </head>
  <body>
     {{{body}}}

    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
```

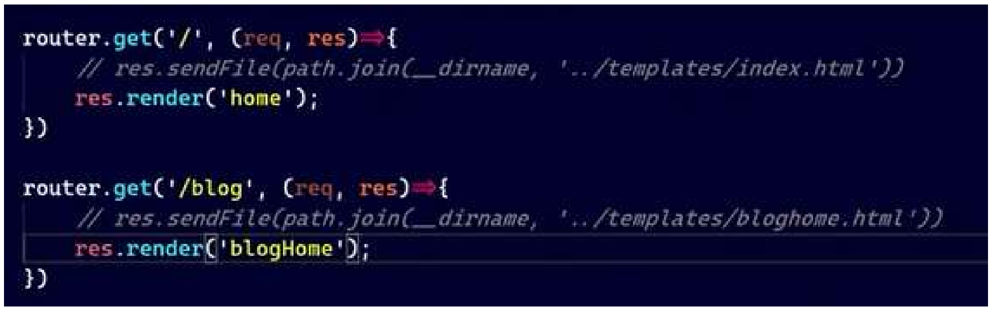
Let us put a navbar in main.handlebars before {{body}}
Now, let us put some bootstrap in our home.handlebars



ts > main.handlebars > html > body > nav.navbar.navbar-expand-lg.navbar-light.bg-light > div.container-fluid > div#navbarSupportedCon

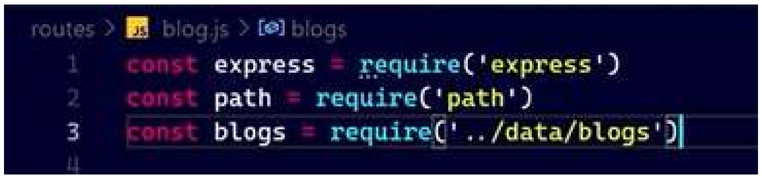
```html
  </head>
  <body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="/">Coding Thunder</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent"
    aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="/">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/blog">Blog</a>
        </li>
      </ul>
      <form class="d-flex">
        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
```

Now we make blogHome.handlebars


```
router.get('/', (req, res)=>{
    // res.sendFile(path.join(__dirname, '../templates/index.html'))
    res.render('home');
})

router.get('/blog', (req, res)=>{
    // res.sendFile(path.join(__dirname, '../templates/bloghome.html'))
    res.render('blogHome');
})
```

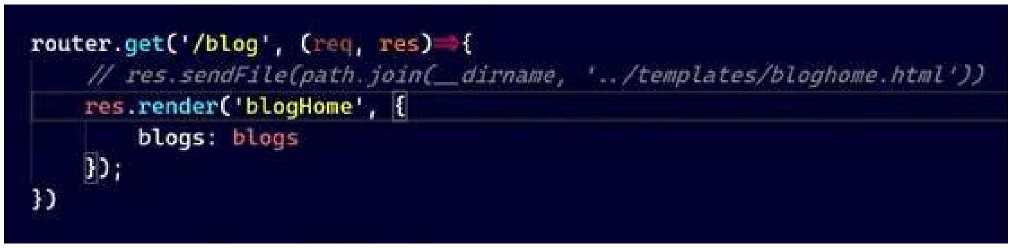
Inside blogHome.handlebars we will render our blogs
We can use Built-in helpers to write if-else or each etc helpers to make our page working.

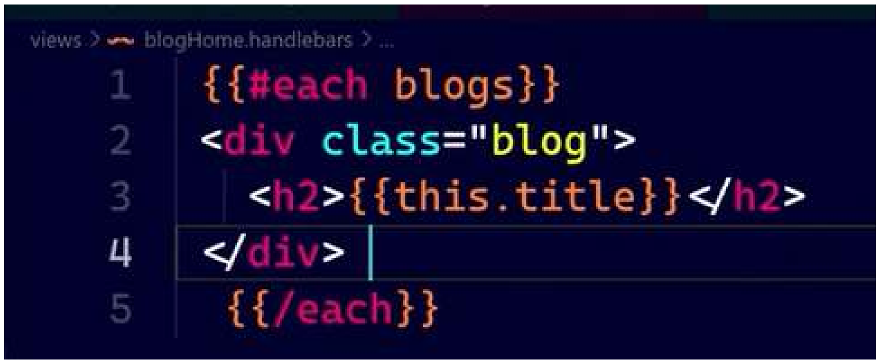
Now we have blogs in blog.js

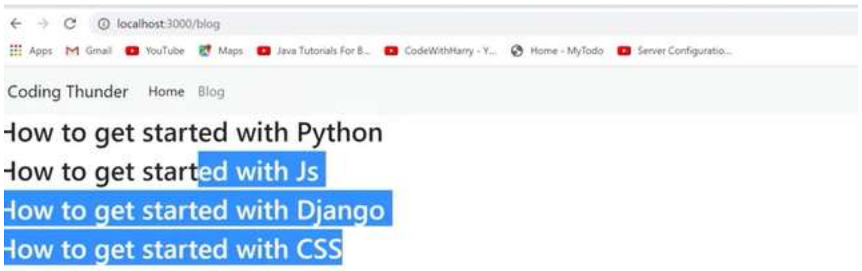

```
routes > blog.js > blogs
const express = require('express')
const path = require('path')
const blogs = require('../data/blogs')
```

What we want is, when we hit /blog endpoint. We need to render blogHome.handlebars but we need to pass blogs to it also for rendering them.

We see docs for it.


```
router.get('/blog', (req, res)=>{
    // res.sendFile(path.join(__dirname, '../templates/bloghome.html'))
    res.render('blogHome', {
        blogs: blogs
    });
})
```


```
views > blogHome.handlebars > ...
{{#each blogs}}
<div class="blog">
  <h2>{{this.title}}</h2>
</div>
 {{/each}}
```


localhost:3000/blog

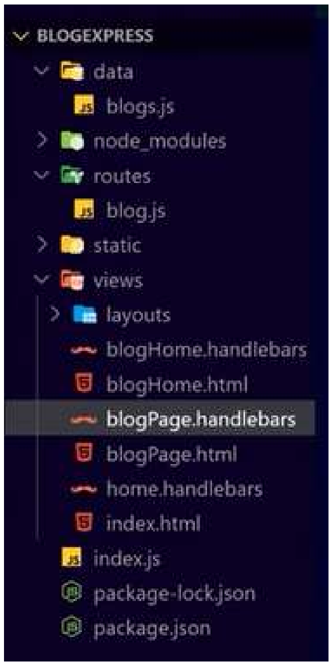
Coding Thunder Home Blog

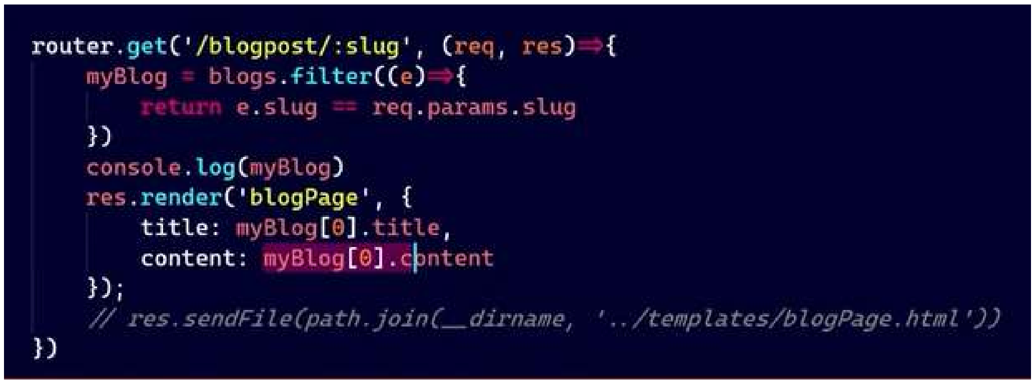
How to get started with Python
How to get started with Js
How to get started with Django
How to get started with CSS

We only put main.handlebars inside views/layout folder

Else all files like home.handlebars etc will be only inside views folder, not inside layout folder.

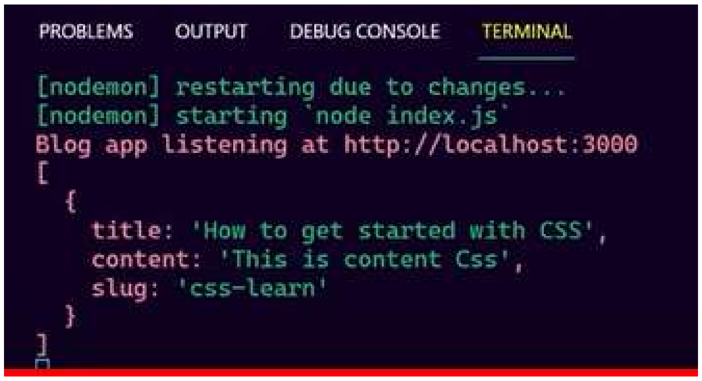
We make a blogPage.handlebars to show any specific blog based on it slug



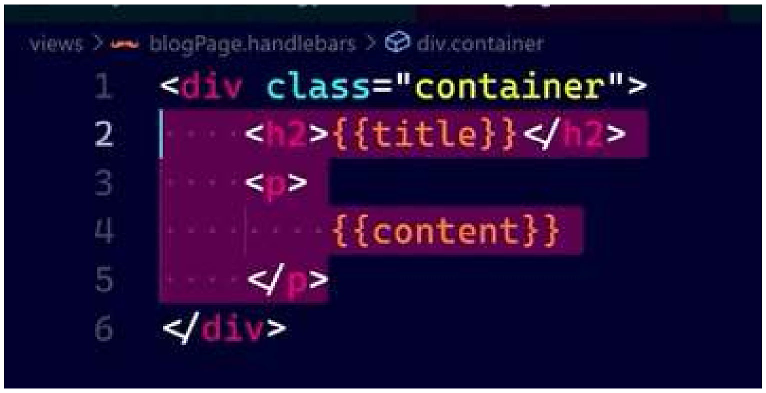


```
router.get('/blogpost/:slug', (req, res)=>{
    myBlog = blogs.filter((e)=>{
        return e.slug == req.params.slug
    })
    console.log(myBlog)
    res.render('blogPage', {
        title: myBlog[0].title,
        content: myBlog[0].content
    });
    // res.sendFile(path.join(__dirname, '../templates/blogPage.html'))
})
```

myBlog is an array so we use myBlog[0] to show the result of filter method



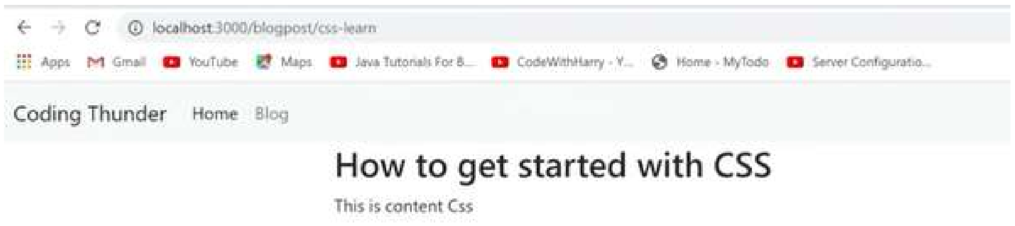
```
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Blog app listening at http://localhost:3000
[
  {
    title: 'How to get started with CSS',
    content: 'This is content Css',
    slug: 'css-learn'
  }
]
```



views > blogPage.handlebars > div.container

```
<div class="container">
    <h2>{{title}}</h2>
    <p>
        {{content}}
    </p>
</div>
```

Output



localhost:3000/blogpost/css-learn

Apps Gmail YouTube Maps Java Tutorials For B... CodeWithHarry - Y... Home - MyTodo Server Configuratio...

Coding Thunder Home Blog

How to get started with CSS

This is content Css