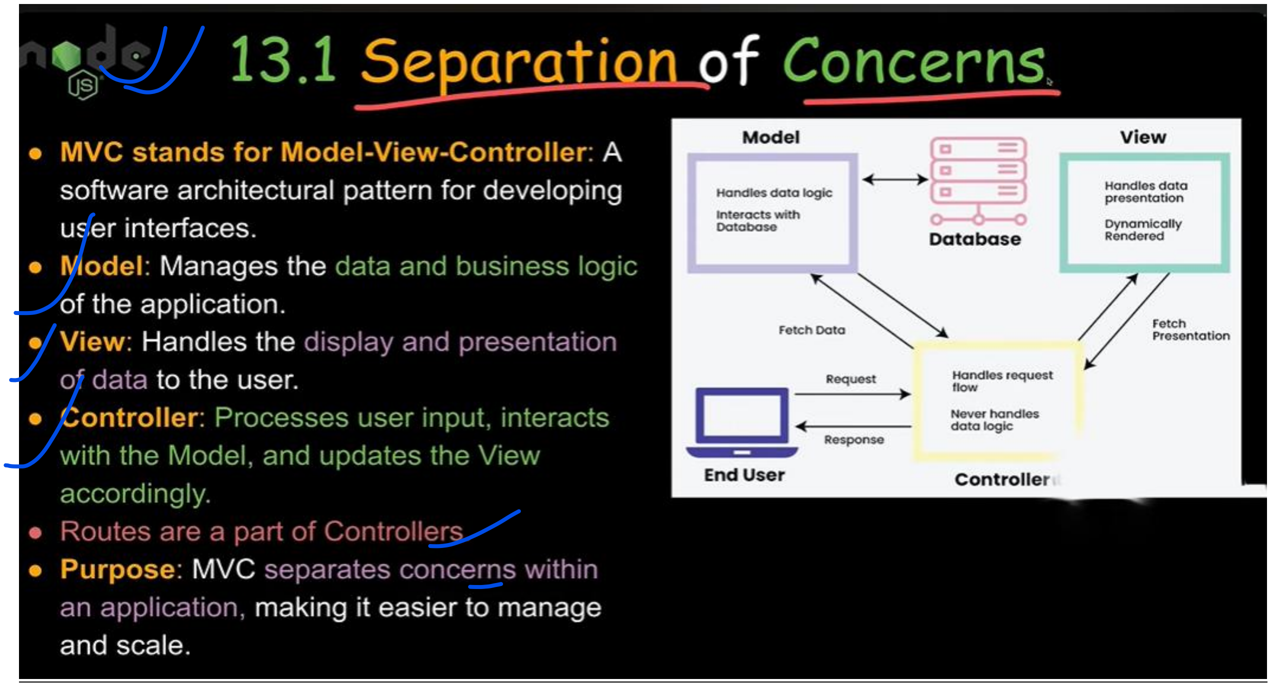
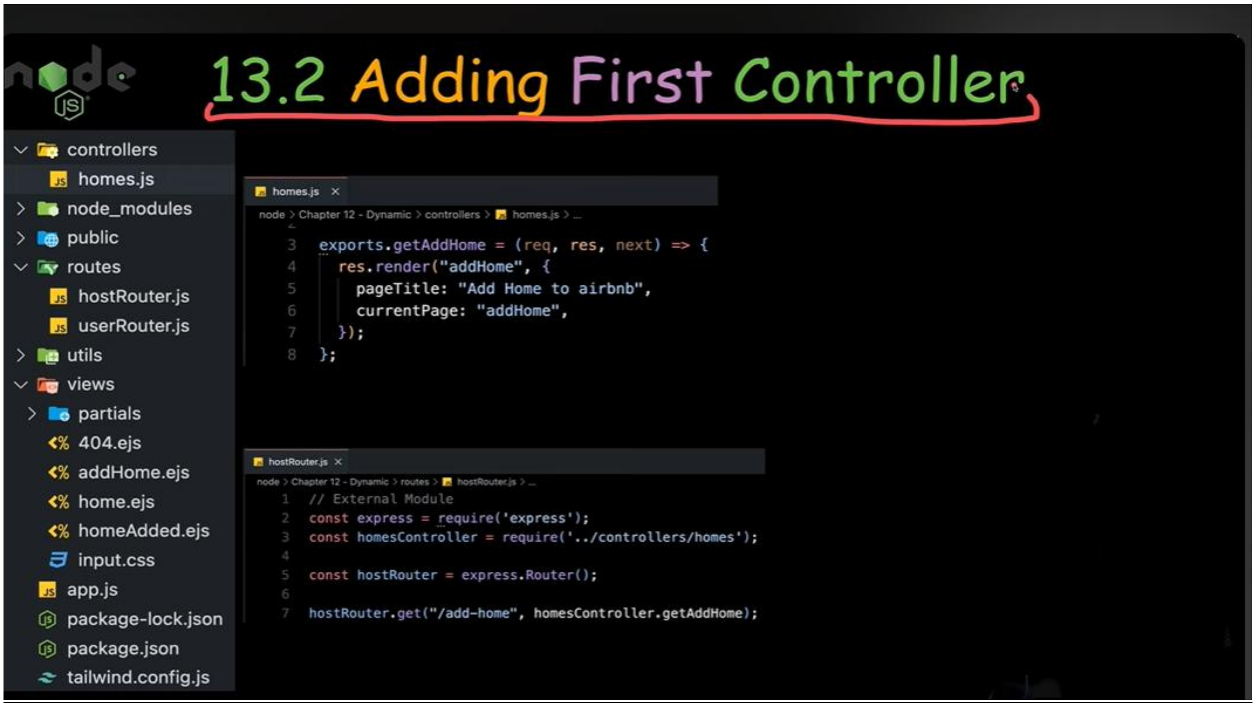
MVC(Model View Controller)



13.1 Separation of Concerns

- **MVC stands for Model-View-Controller**: A software architectural pattern for developing user interfaces.
- **Model**: Manages the data and business logic of the application.
- **View**: Handles the display and presentation of data to the user.
- **Controller**: Processes user input, interacts with the Model, and updates the View accordingly.
- Routes are a part of Controllers
- **Purpose**: MVC separates concerns within an application, making it easier to manage and scale.



13.2 Adding First Controller

```
controllers
  homes.js
node_modules
public
routes
  hostRouter.js
  userRouter.js
utils
views
  partials
  404.ejs
  addHome.ejs
  home.ejs
  homeAdded.ejs
  input.css
app.js
package-lock.json
package.json
tailwind.config.js
```

homes.js

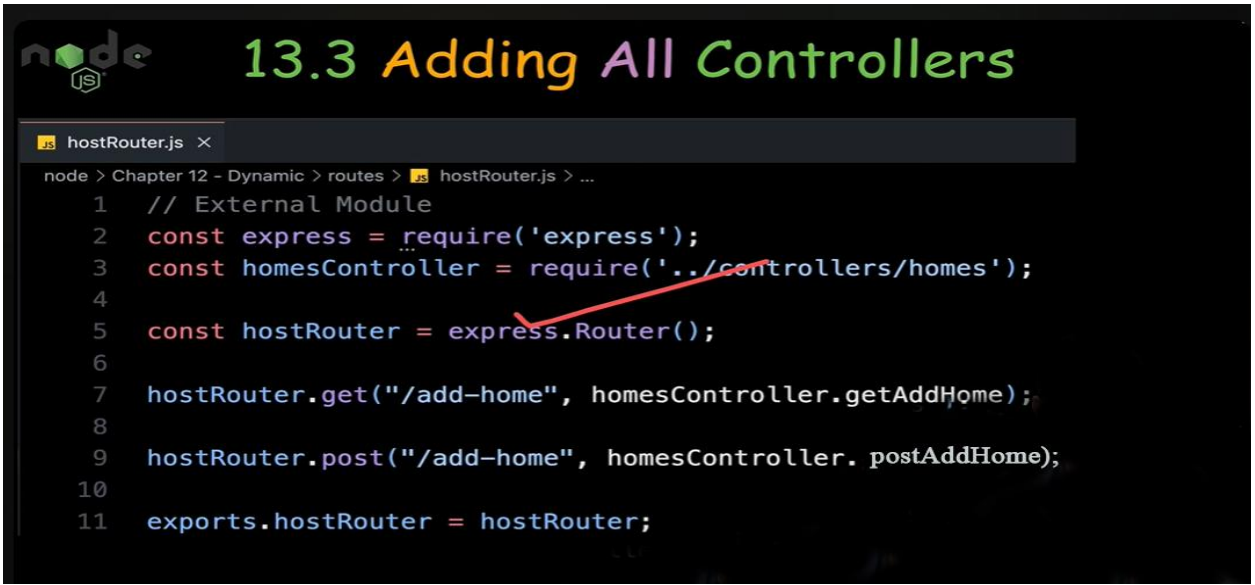
node > Chapter 12 - Dynamic > controllers > homes.js > ...

```javascript
exports.getAddHome = (req, res, next) => {
  res.render("addHome", {
    pageTitle: "Add Home to airbnb",
    currentPage: "addHome",
  });
};
```

hostRouter.js

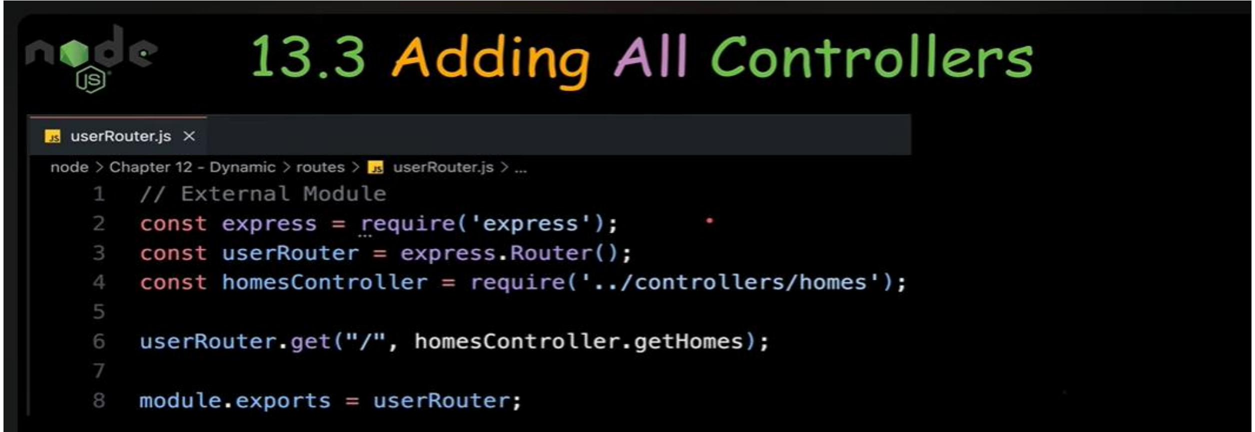
node > Chapter 12 - Dynamic > routes > hostRouter.js > ...

```javascript
// External Module
const express = require('express');
const homesController = require('../controllers/homes');

const hostRouter = express.Router();

hostRouter.get("/add-home", homesController.getAddHome);
```



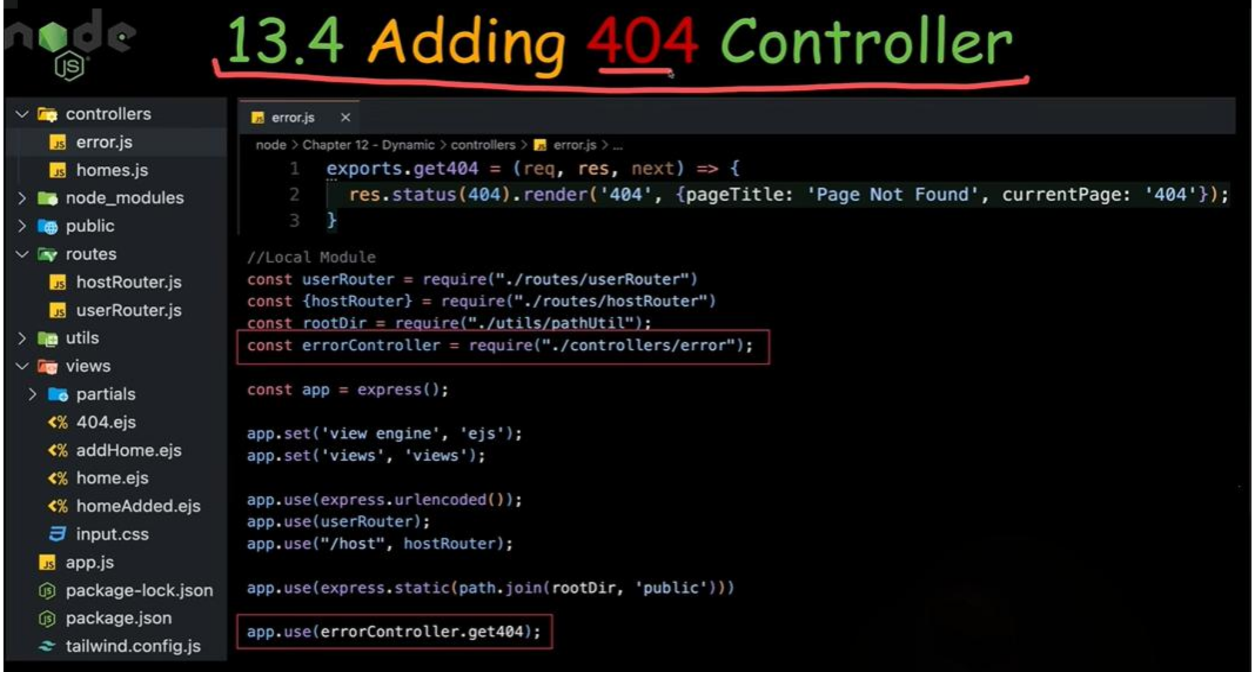
13.3 Adding All Controllers

```
// External Module
const express = require('express');
const homesController = require('../controllers/homes');

const hostRouter = express.Router();

hostRouter.get("/add-home", homesController.getAddHome);

hostRouter.post("/add-home", homesController. postAddHome);

exports.hostRouter = hostRouter;
```



13.3 Adding All Controllers

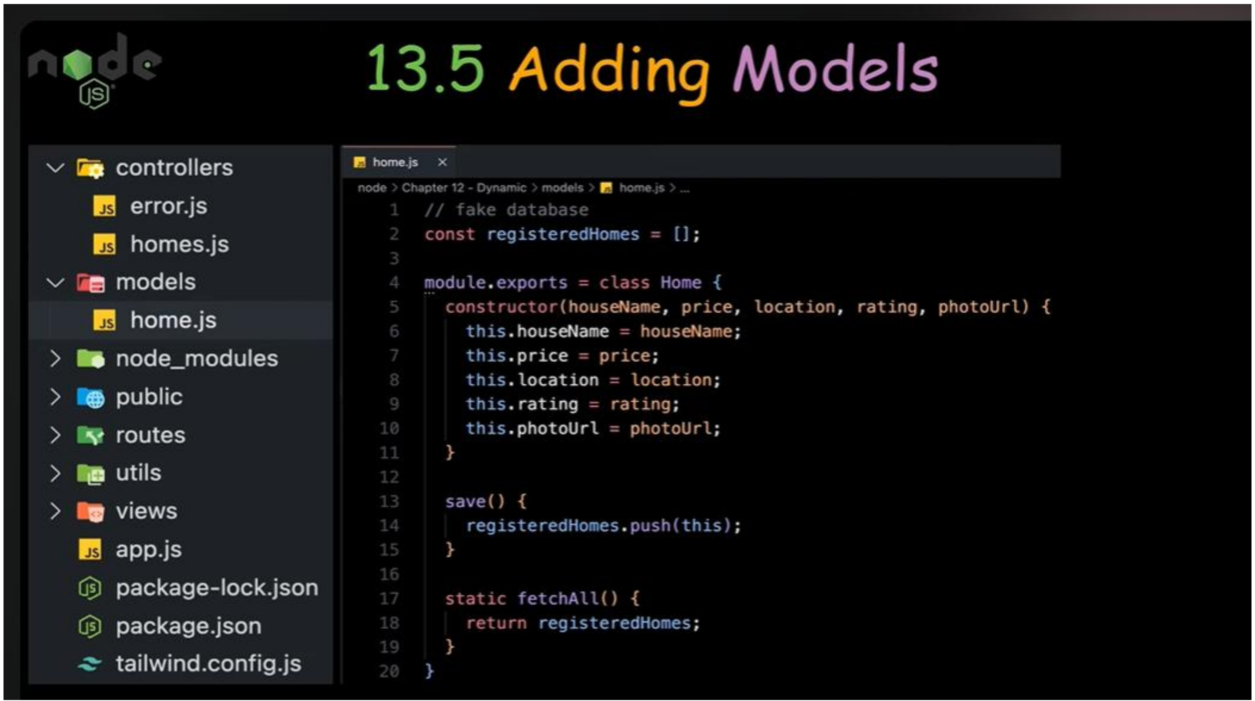
```
// External Module
const express = require('express');
const userRouter = express.Router();
const homesController = require('../controllers/homes');

userRouter.get("/", homesController.getHomes);

module.exports = userRouter;
```



13.4 Adding 404 Controller

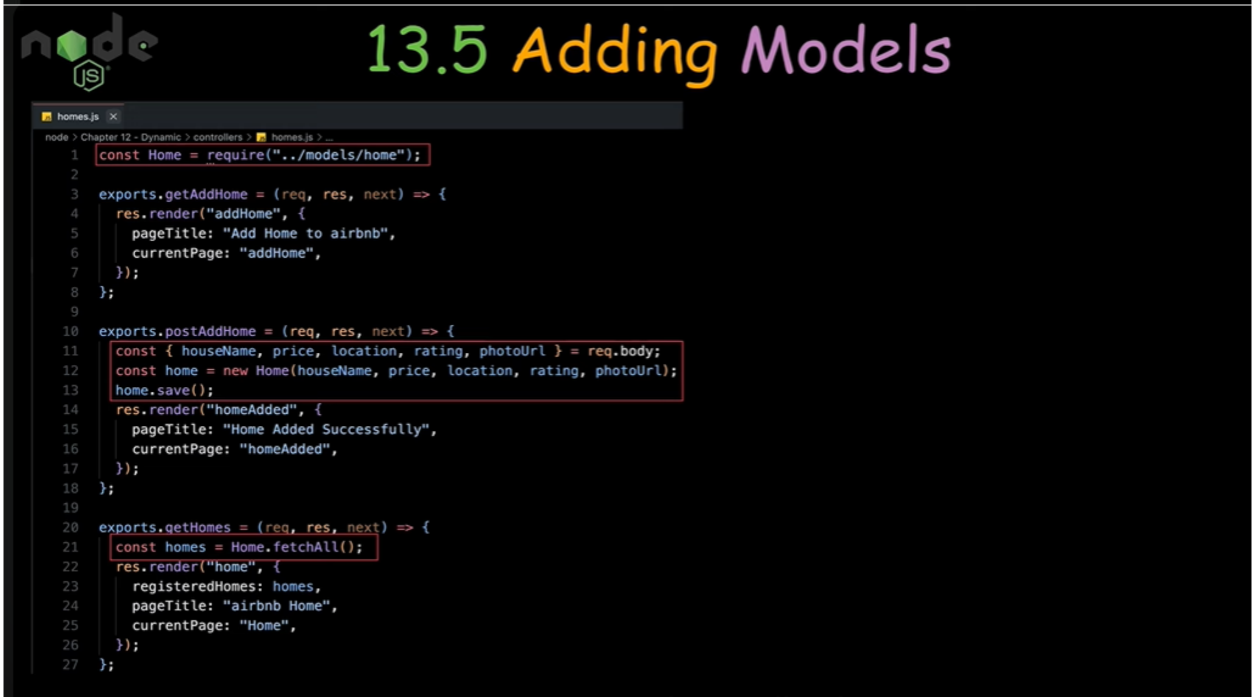
```
exports.get404 = (req, res, next) => {
  res.status(404).render('404', {pageTitle: 'Page Not Found', currentPage: '404'});
}
```

```
//Local Module
const userRouter = require("./routes/userRouter")
const {hostRouter} = require("./routes/hostRouter")
const rootDir = require("./utils/pathUtil");
const errorController = require("./controllers/error");

const app = express();

app.set('view engine', 'ejs');
app.set('views', 'views');

app.use(express.urlencoded());
app.use(userRouter);
app.use("/host", hostRouter);

app.use(express.static(path.join(rootDir, 'public')))

app.use(errorController.get404);
```



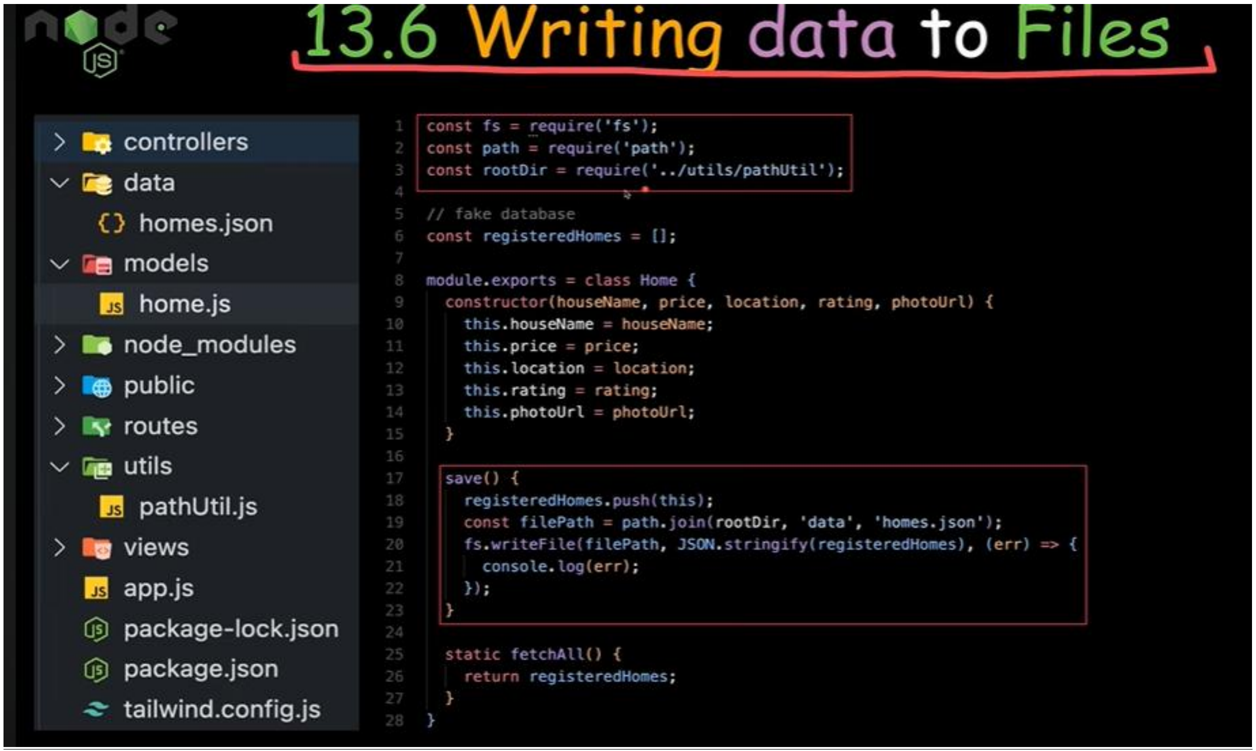
13.5 Adding Models

```
// fake database
const registeredHomes = [];

module.exports = class Home {
  constructor(houseName, price, location, rating, photoUrl) {
    this.houseName = houseName;
    this.price = price;
    this.location = location;
    this.rating = rating;
    this.photoUrl = photoUrl;
  }

  save() {
    registeredHomes.push(this);
  }

  static fetchAll() {
    return registeredHomes;
  }
}
```



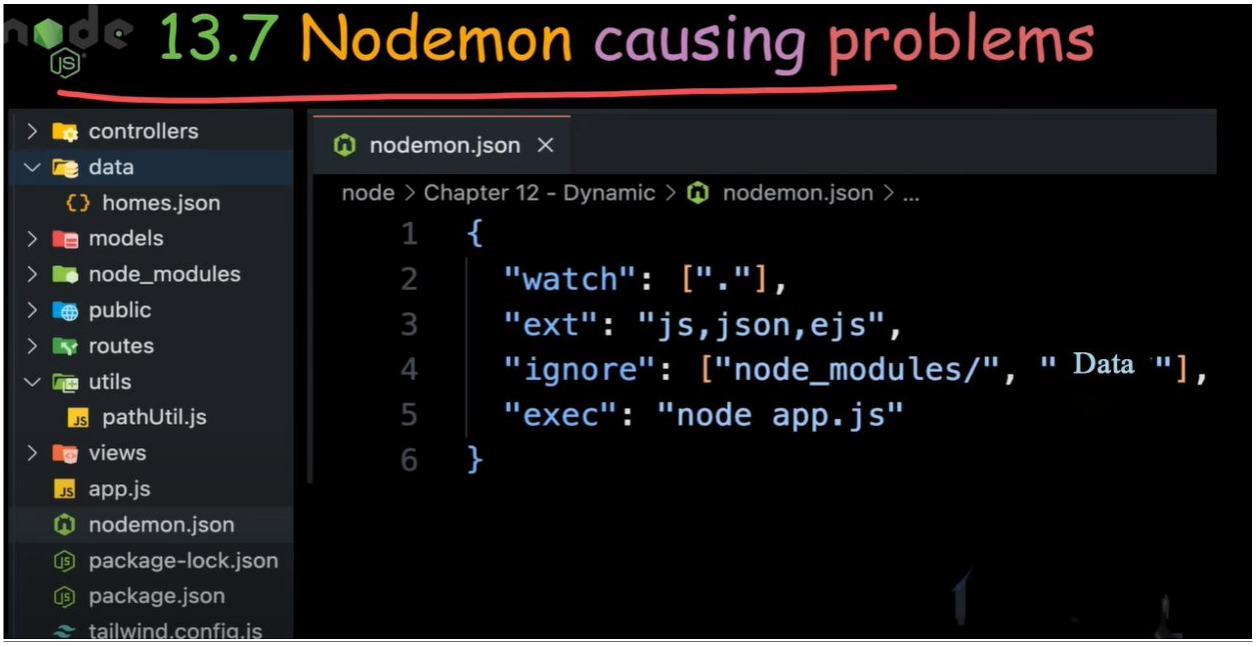
13.5 Adding Models

```
const Home = require("../models/home");

exports.getAddHome = (req, res, next) => {
  res.render("addHome", {
    pageTitle: "Add Home to airbnb",
    currentPage: "addHome",
  });
};

exports.postAddHome = (req, res, next) => {
  const { houseName, price, location, rating, photoUrl } = req.body;
  const home = new Home(houseName, price, location, rating, photoUrl);
  home.save();
  res.render("homeAdded", {
    pageTitle: "Home Added Successfully",
    currentPage: "homeAdded",
  });
};

exports.getHomes = (req, res, next) => {
  const homes = Home.fetchAll();
  res.render("home", {
    registeredHomes: homes,
    pageTitle: "airbnb Home",
    currentPage: "Home",
  });
};
```



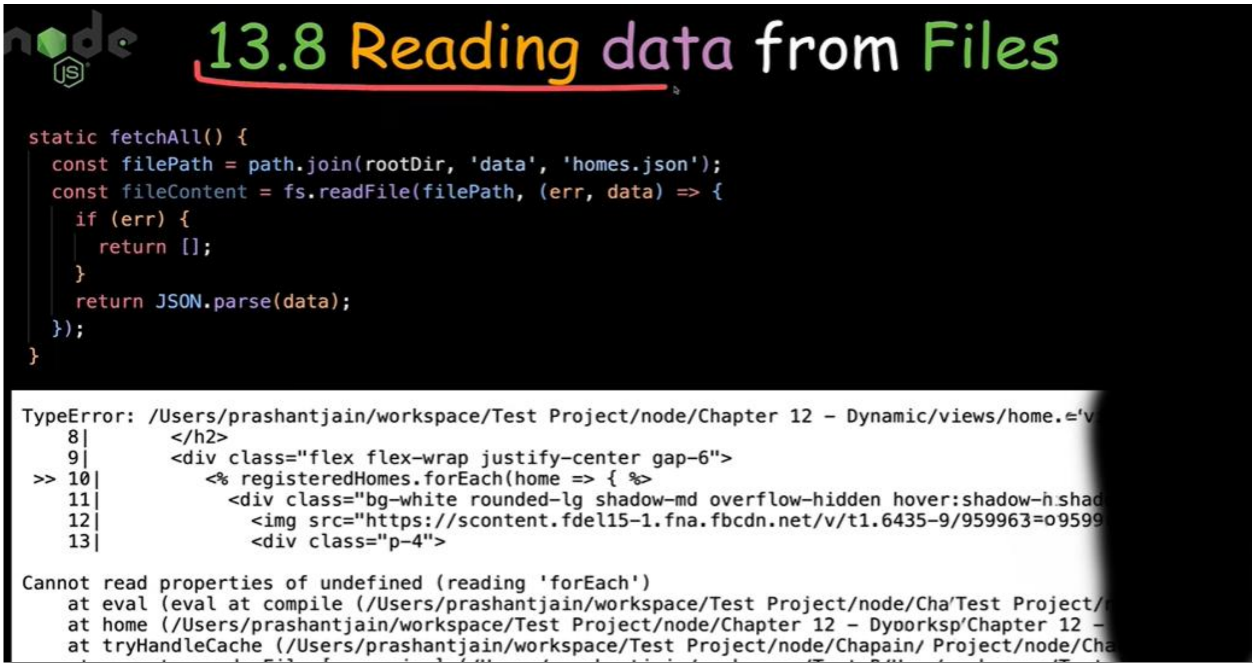
13.6 Writing data to Files

```
const fs = require('fs');
const path = require('path');
const rootDir = require('../utils/pathUtil');

// fake database
const registeredHomes = [];

module.exports = class Home {
  constructor(houseName, price, location, rating, photoUrl) {
    this.houseName = houseName;
    this.price = price;
    this.location = location;
    this.rating = rating;
    this.photoUrl = photoUrl;
  }

  save() {
    registeredHomes.push(this);
    const filePath = path.join(rootDir, 'data', 'homes.json');
    fs.writeFile(filePath, JSON.stringify(registeredHomes), (err) => {
      console.log(err);
    });
  }

  static fetchAll() {
    return registeredHomes;
  }
}
```



13.7 Nodemon causing problems

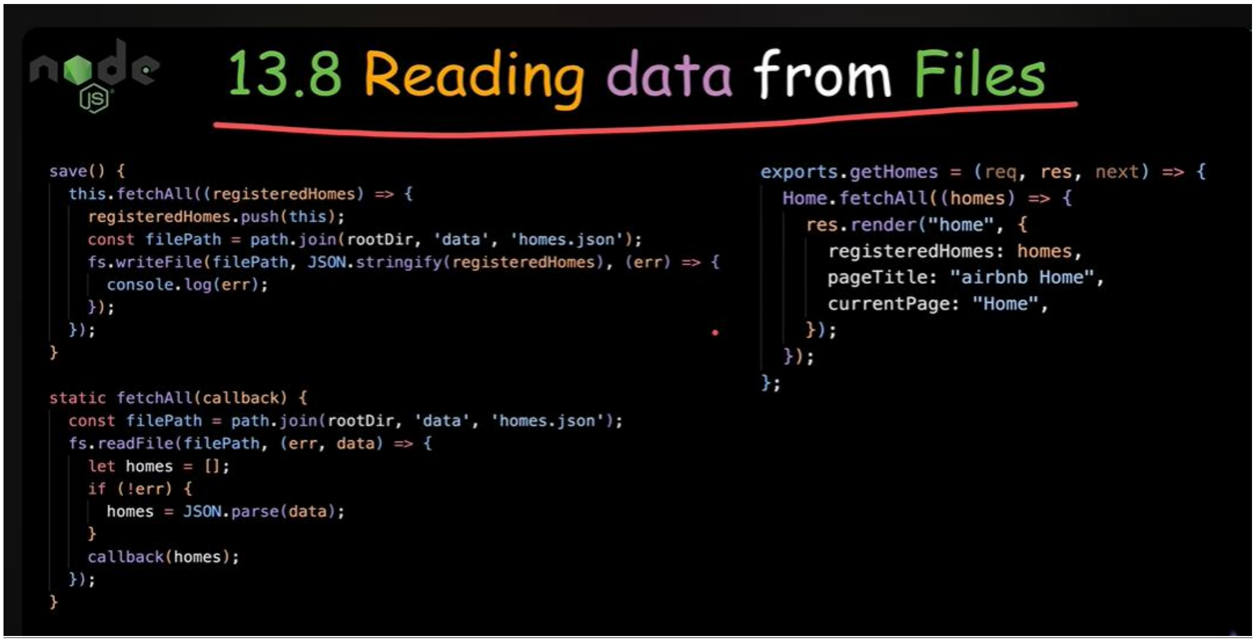
```
{
  "watch": ["."],
  "ext": "js,json,ejs",
  "ignore": ["node_modules/", " Data "],
  "exec": "node app.js"
}
```



13.8 Reading data from Files

```
static fetchAll() {
  const filePath = path.join(rootDir, 'data', 'homes.json');
  const fileContent = fs.readFile(filePath, (err, data) => {
    if (err) {
      return [];
    }
    return JSON.parse(data);
  });
}
```

```
TypeError: /Users/prashantjain/workspace/Test Project/node/Chapter 12 - Dynamic/views/home.e'v
    8|         </h2>
    9|         <div class="flex flex-wrap justify-center gap-6">
 >> 10|           <% registeredHomes.forEach(home => { %>
    11|             <div class="bg-white rounded-lg shadow-md overflow-hidden hover:shadow-h shad
    12|               <img src="https://scontent.fdel15-1.fna.fbcdn.net/v/t1.6435-9/959963=o9599
    13|               <div class="p-4">

Cannot read properties of undefined (reading 'forEach')
    at eval (eval at compile (/Users/prashantjain/workspace/Test Project/node/Cha'Test Project/
    at home (/Users/prashantjain/workspace/Test Project/node/Chapter 12 - Dyoorksp'Chapter 12 -
    at tryHandleCache (/Users/prashantjain/workspace/Test Project/node/Chapain/ Project/node/Cha
```



13.8 Reading data from Files

```
save() {
  this.fetchAll((registeredHomes) => {
    registeredHomes.push(this);
    const filePath = path.join(rootDir, 'data', 'homes.json');
    fs.writeFile(filePath, JSON.stringify(registeredHomes), (err) => {
      console.log(err);
    });
  });
}

static fetchAll(callback) {
  const filePath = path.join(rootDir, 'data', 'homes.json');
  fs.readFile(filePath, (err, data) => {
    let homes = [];
    if (!err) {
      homes = JSON.parse(data);
    }
    callback(homes);
  });
}
```

```
exports.getHomes = (req, res, next) => {
  Home.fetchAll((homes) => {
    res.render("home", {
      registeredHomes: homes,
      pageTitle: "airbnb Home",
      currentPage: "Home",
    });
  });
};
```