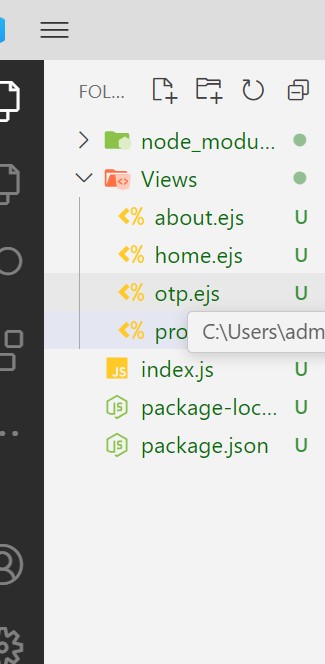
## Page:1

# EJS Project



## Index.js

const express = require("express");

const app = express();

app.set("view engine", "ejs");

app.get("/home", (req, res) => {

  //   res.send("<h1>Home Page</h1>"); //server side rendering

  res.render("home");

});

## Page:2

app.get("/about", (req, res) => {

  res.render("about");

});

app.get("/otp", (req, res) => {

  const randomotp = Math.round(Math.random() \* 10000);

  console.log(randomotp);

  res.render("otp", { otp: randomotp }); //key and value

});

app.get("/product", (req, res) => {

  let productdata = [

    {

      id: 1,

      title: "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",

      price: 109.95,

      description:

        "Your perfect pack for everyday use and walks in the forest. Stash your laptop (up to 15 inches) in the padded sleeve, your everyday",

      category: "men's clothing",

      image: "https://fakestoreapi.com/img/81fPKd-2AYL.\_AC\_SL1500\_.jpg",

      rating: {

        rate: 3.9,

        count: 120,

      },

    },

    {

      id: 2,

      title: "Mens Casual Premium Slim Fit T-Shirts",

## Page:3

  price: 22.3,

      description:

        "Slim-fitting style, contrast raglan long sleeve, three-button henley placket, light weight & soft fabric for breathable and comfortable wearing.",

      category: "men's clothing",

      image:

        "https://fakestoreapi.com/img/71-3HjGNDUL.\_AC\_SY879.\_SX.\_UX.\_SY.\_UY\_.jpg",

      rating: {

        rate: 4.1,

        count: 259,

      },

    },

    {

      id: 3,

      title: "Mens Cotton Jacket",

      price: 55.99,

      description:

        "Great outerwear jackets for Spring/Autumn/Winter, suitable for many occasions like working, hiking, camping, or traveling.",

      category: "men's clothing",

      image: "https://fakestoreapi.com/img/71li-ujtlUL.\_AC\_UX679\_.jpg",

      rating: {

        rate: 4.7,

        count: 500,

      },

    },

    {

      id: 4,

      title: "Mens Casual Slim Fit",

      price: 15.99,

## Page:4

  description:

        "The color could be slightly different between on the screen and in practice. Please review size information before purchasing.",

      category: "men's clothing",

      image: "https://fakestoreapi.com/img/71YXzeOuslL.\_AC\_UY879\_.jpg",

      rating: {

        rate: 2.1,

        count: 430,

      },

    },

    {

      id: 5,

      title:

        "John Hardy Women's Legends Naga Gold & Silver Dragon Station Chain Bracelet",

      price: 695,

      description:

        "From our Legends Collection, the Naga was inspired by the mythical water dragon that protects the ocean's pearl. Wear inward for love and abundance, outward for protection.",

      category: "jewelery",

      image: "https://fakestoreapi.com/img/71pWzhdJNwL.\_AC\_UL640\_QL65\_ML3\_.jpg",

      rating: {

        rate: 4.6,

        count: 400,

      },

    },

  ];

  res.render("product", { productdata });

});

app.listen(8080, () => {

## Page:5

  console.log("server is running on port 8080");

});

## Product.ejs

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>product page</h1>

    <!--single card-->

    <% productdata.map((el)=>{%>

        <img src=<%=el.image %> alt="" height="200px" width="200px" />

        <h3>

            <%= el.title %>

        </h3>

        <% }) %>

</body>

</html>

## Page:6

## Otp.ejs

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>otp is <%= otp %>

    </h1>

</body>

</html>

# Explanation

This code is a basic Node.js Express server that uses **EJS (Embedded JavaScript)** as the template engine to render dynamic web pages. Let’s break down the core concepts and what’s happening:

**1. Express and EJS Setup**

javascript

Copy code

const express = require("express"); // Import the express framework

const app = express(); // Create an express app instance

app.set("view engine", "ejs"); // Set EJS as the templating engine

* express: A web framework for building web applications in Node.js.
* ejs: A template engine that helps render HTML with dynamic data.

## Page:7

* app.set("view engine", "ejs"): Tells Express to use EJS to render the HTML pages.

**2. Route Definitions**

In Express, routes define the endpoints (URLs) your server will respond to. Each route specifies a URL path, and a callback function to handle the request and response.

**Home Route (/home)**

javascript

Copy code

app.get("/home", (req, res) => {

res.render("home");

});

* app.get("/home", ...): Defines a route that responds to GET requests at the /home URL.
* res.render("home"): This renders the home.ejs file (from the views folder). The server will render this page to the client-side, dynamically replacing any EJS tags inside it.

**About Route (/about)**

javascript

Copy code

app.get("/about", (req, res) => {

res.render("about");

});

* Similar to the /home route, this renders an about.ejs file.

**OTP Route (/otp)**

javascript

Copy code

app.get("/otp", (req, res) => {

const randomotp = Math.round(Math.random() \* 10000);

console.log(randomotp);

res.render("otp", { otp: randomotp });

});

* This generates a random 4-digit number (randomotp) and renders it into the otp.ejs view. The otp value is passed as a key-value pair to the view, where { otp: randomotp } makes otp available inside the otp.ejs file to display.

## Page:8

* The console.log(randomotp) prints the generated OTP on the server side (useful for debugging).

**Product Route (/product)**

javascript

Copy code

app.get("/product", (req, res) => {

let productdata = [/\* Array of Product Objects \*/];

res.render("product", { productdata });

});

* This route handles the rendering of the product.ejs view and passes an array of product objects to the template (productdata).
* Inside the product.ejs file, you can loop through this array and display product information dynamically.

**3. The Server**

javascript

Copy code

app.listen(8080, () => {

console.log("server is running on port 8080");

});

* This starts the server on port 8080 and logs that the server is running.
* The server will now listen for requests on this port, respond to the routes defined, and render the corresponding EJS views.

**Summary of Key Concepts**

* **Express**: A web server framework that allows you to handle HTTP requests.
* **EJS**: A template engine that helps you render HTML dynamically, passing data from your server to the client-side.
* **Routes**: Define the paths (/home, /about, /otp, etc.) where your server will respond and what to do when those paths are requested.
* **res.render()**: This renders a template (e.g., home.ejs) and injects any data you pass into it.

You can create .ejs files in a views folder and use the passed data (like otp and productdata) to dynamically display content on your web pages.

## Page:9

# Product.ejs

This is an **EJS (Embedded JavaScript)** template, which combines HTML and JavaScript to dynamically render content on a webpage. Here's a breakdown of the code:

**1. HTML Structure**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

* **DOCTYPE**: Tells the browser that this is an HTML5 document.
* **<html lang="en">**: Sets the language of the document to English.
* **<meta charset="UTF-8">**: Specifies the character encoding (UTF-8 supports most characters in all languages).
* **<meta name="viewport" content="width=device-width, initial-scale=1.0">**: Ensures the webpage is responsive, making it scale properly on different devices (especially important for mobile view).
* **<title>Document</title>**: Sets the title of the web page (displayed in the browser tab).

**2. Body Content**

html

Copy code

<body>

<h1>Product Page</h1>

<!--single card-->

<% productdata.map((el) => { %>

## Page:10

<img src=<%= el.image %> alt="" height="200px" width="200px" />

<h3>

<%= el.title %>

</h3>

<% }) %>

</body>

</html>

* **<h1>Product Page</h1>**: Displays a header on the page saying "Product Page".
* **<!-- single card -->**: This comment is just a note for developers, explaining that the following section is a "single card" (used to show a product).

**3. EJS Tags for Dynamic Content**

The EJS-specific parts of the template allow JavaScript logic to be embedded into HTML.

**<% ... %> vs <%= ... %>**

* **<% %>**: Used for control flow statements (like loops, conditions, etc.). It executes JavaScript code but doesn’t print anything to the HTML.
* **<%= %>**: Outputs the value of a JavaScript expression or variable into the HTML.

**Mapping over productdata**

html

Copy code

<% productdata.map((el) => { %>

* **productdata.map((el) => { })**: This is a loop that iterates over the productdata array, where each product object is represented by el.
* For each product (el), the loop outputs dynamic content for the product’s image and title.

**Displaying the Product Image**

html

Copy code

<img src=<%= el.image %> alt="" height="200px" width="200px" />

* **<%= el.image %>**: This dynamically inserts the image URL from the el object into the src attribute of the <img> tag. It will display the image for each product.

**Displaying the Product Title**

html

## Page:11

Copy code

<h3><%= el.title %></h3>

* **<%= el.title %>**: This dynamically inserts the title of the product from the el object into an <h3> header tag for each product.

**Summary of Key Concepts:**

* **HTML and EJS Mix**: This template uses HTML for structure and design, while EJS inserts dynamic content.
* **<%= ... %> for Output**: EJS uses this syntax to print dynamic data (like el.title or el.image) from JavaScript into the HTML.
* **Looping with .map()**: The loop dynamically generates HTML for each product in the productdata array, so if you have 5 products, it will generate 5 sets of image and title blocks.

In this case, the page will show a product list with images and titles based on the productdata array passed from your server.

# Opt.ejs

This HTML document is a basic web page template that includes some dynamic content. Let’s break it down step by step:

**Structure of the Document**

1. **DOCTYPE Declaration**:

html

Copy code

<!DOCTYPE html>

* + This line tells the web browser that the document is an HTML5 document. It helps the browser render the page correctly.

1. **HTML Element**:

html

Copy code

<html lang="en">

* + This tag defines the beginning of the HTML document. The lang="en" attribute specifies that the language of the document is English.

1. **Head Section**:

html

Copy code

## Page:12

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

* + The <head> section contains metadata about the document:
    - <meta charset="UTF-8"> specifies the character encoding for the document, allowing it to display a wide range of characters.
    - <meta name="viewport" content="width=device-width, initial-scale=1.0"> helps with responsive design, ensuring that the page scales correctly on different devices (like mobiles and tablets).
    - <title>Document</title> sets the title of the page, which appears in the browser tab.

1. **Body Section**:

html

Copy code

<body>

<h1>otp is <%= otp %></h1>

</body>

* + The <body> section contains the content of the web page that will be displayed to users.
  + <h1> is a header tag that defines a large heading.
  + Inside the <h1> tag, you see <%= otp %>. This is a placeholder for a dynamic value, usually generated by a server-side language (like Node.js, Ruby on Rails, or PHP).
    - The <%= %> syntax is often used in templating engines (like EJS or JSP) to insert a variable value into the HTML. Here, otp would be replaced with the actual one-time password (OTP) value when the page is rendered.

**Summary**

In simple terms, this document is a template for a web page that will display a one-time password (OTP) within a heading. The OTP value will be provided by a server when the page is requested. The HTML structure ensures that the page is properly formatted and can be displayed across different devices.