Agenda

- What are Express Template Engines?
- Introduction to EJS
- How to create a `views` folder in EJS
- Conditional Rendering and Loops in EJS

Express Template Engines

A ***template engine*** enables you to use static template files in your application. At runtime, the template engine replaces variables in a template file wi actual values, and transforms the template into an HTML file sent to the client. This approach makes it easier to design an HTML page.

Some popular template engines that work with Express are Pug, Mustache, and EJS. The Express application generator uses Jade as its default, but also supports several others. Following are some of the most popular template engines :

- Pug
- Mustache
- EJS
- Jade
- dust
- handlebars
- templayed

In the above template engines, pug, EJS and mustache seems to be most popular choice. Pug is similar to Haml which uses whitespace. According the template-benchmark, pug is 2x slower than Handlebars, EJS.

Using Template Engines

Template engine makes you able to use static template files in your application. To render template files you have to set the following application settir properties:

- **Views:** It specifies a directory where the template files are located.

For example: `app.set('views', './views')`

If the `views` are not set explicitly, Express will look at the `./views` directory by default.

- **view engine:** It specifies the template engine that you use. For example, to use the Pug template engine: `app.set('view engine', 'ejs')` .

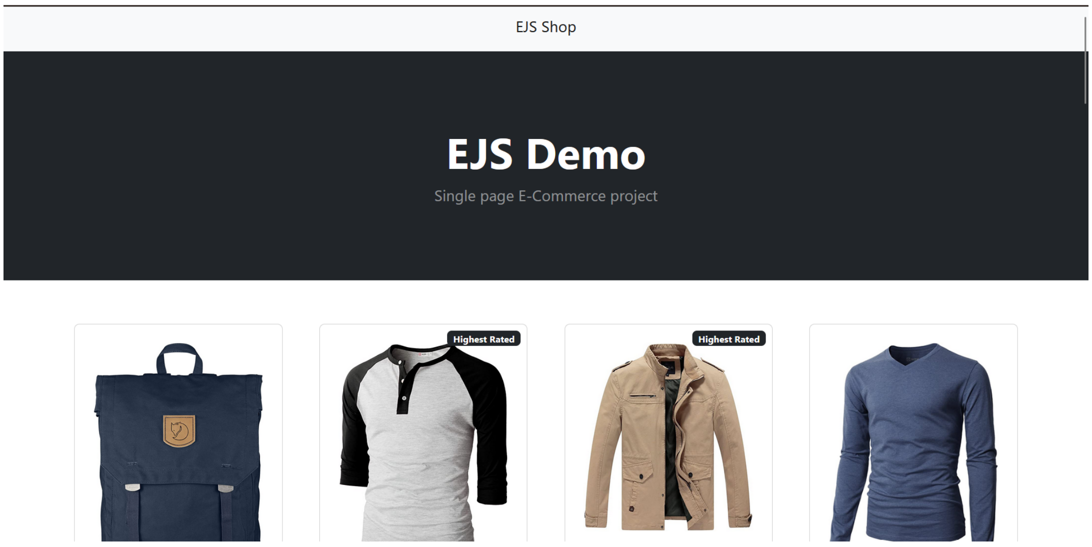
Introduction to EJS

When quickly creating Node applications, a fast way to template your application is sometimes necessary.

Jade comes as the default template engine for Express but Jade syntax can be overly complex for many use cases.

Embedded JavaScript templates (EJS) can be used as an alternative template engine.

In this session, you will learn how to apply EJS to an Express application, include repeatable parts of your site, and pass data to the views by creatir the following single page E-Commerce application:



EJS Demo

Single page E-Commerce project

Setting up Project Folder

First, open your terminal window and create a new project directory:

```
mkdir ejs-shop
```

Then, navigate to the newly created directory:

```
cd ejs-shop
```

At this point, you can initialize a new npm project:

```
npm init -y
```

Next, you will need to install the `express` package:

```
npm i express
```

Then install the `ejs` and `nodemon` packages:

```
npm i ejs
npm i --save-dev nodemon
```

Configuring with `server.js`

With all of the dependencies installed, let's configure the application to use EJS and set up the routes for the Index page and the About page.

Create a new `server.js` file and open it with your code editor and add the following lines of code:

```
var express = require('express');
var app = express();

// set the view engine to ejs
app.set('view engine', 'ejs');

// use res.render to load up an ejs view file

// index page
app.get('/', function (req, res) {
  res.render('pages/index');
});
```

```
app.listen(8080);
console.log('Server is listening on port 8080');
```

This code defines the application and listens on port `8080`.

This code also sets EJS as the view engine for the Express application using:

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

Notice how the code sends a view to the user by using `res.render()`. It is important to note that `res.render()` will look in a `views` folder for th view. So you only have to define `pages/index` since the full path is `views/pages/index`.

Next, let's create the views using EJS.

Creating EJS Views and Partials

Like a lot of the applications you build, there will be a lot of code that is reused. These are considered *partials*. In this example, there will be five partia that will be used on the Home page and might subsequently be used on other pages as the project grov : `head.ejs`, `hero.ejs`, `navbar.ejs`, `productCard.ejs` and `footer.ejs`. Let's make those files now.

Create a new `views` directory:

```
mkdir views
```

Then, create a new `partials` subdirectory:

```
mkdir views/partials
```

In this directory, create a new `head.ejs` file and open it with your code editor. Add the following lines of code:

```
<meta charset="utf-8" />
<meta
  name="viewport"
  content="width=device-width, initial-scale=1, shrink-to-fit=no"
/>
<meta name="description" content="" />
<meta name="author" content="" />
<title>EJS Demo</title>
<!-- Bootstrap icons-->
<link
  href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-icons.css"
  rel="stylesheet"
/>
<!-- Core theme CSS (includes Bootstrap)-->
<link
  href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css"
  rel="stylesheet"
  integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1Bx"
  crossorigin="anonymous"
/>
```

This code contains metadata for the `head` for an HTML document. It also includes Bootstrap styles.

Next, create a new `navbar.ejs` file and open it with your code editor. Add the following lines of code:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container px-4 px-lg-5">
    <a class="navbar-brand mx-auto h1" href="#!">EJS Shop</a>
  </div>
</nav>
```

This code contains navigation for an HTML document and uses several classes from Bootstrap for styling.

Now, create `hero.ejs` file and add the following code :

```
<header class="bg-dark py-5">
  <div class="container px-4 px-lg-5 my-5">
    <div class="text-center text-white">
      <h1 class="display-4 fw-bolder">EJS Demo</h1>
      <p class="lead fw-normal text-white-50 mb-0">
        Single page E-Commerce project
      </p>
    </div>
  </div>
</header>
```

Next, create a new `footer.ejs` file and open it with your code editor. Add the following lines of code:

```
<div class="container">
  <p class="m-0 text-center text-white">
    Copyright &copy; AlmaBetter 2022
  </p>
</div>
```

Adding the EJS Partials to Views

You have four partials defined. Now you can `include` them in your views.

Use `<%- include('RELATIVE/PATH/TO/FILE') %>` to embed an EJS partial in another file.

- The hyphen `<%-` instead of just `<%` to tell EJS to render raw HTML.
- The path to the partial is relative to the current file.

Then, create a new `pages` subdirectory:

```
mkdir views/pages
```

In this directory, create a new `index.ejs` file and open it with your code editor. Add the following lines of code:

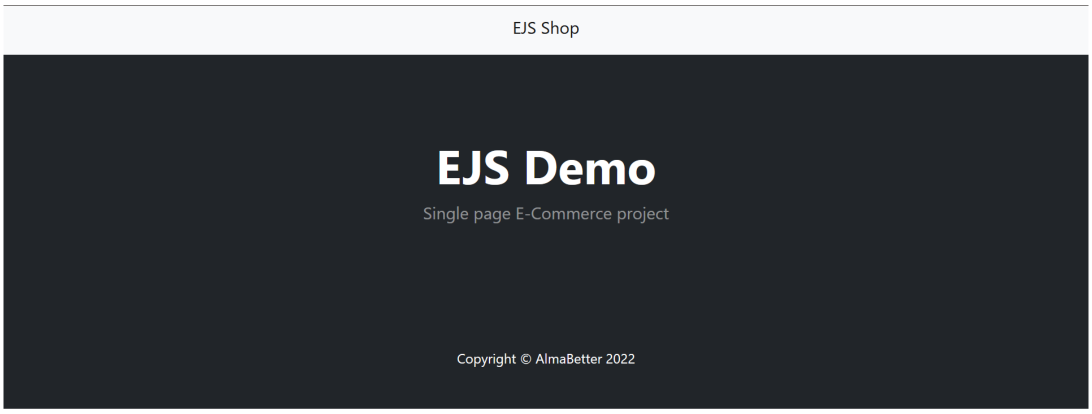
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <%- include('../partials/head'); %>
  </head>
  <body>
    <!-- Navigation-->
    <%- include('../partials/navbar'); %>

    <!-- Hero Section -->
    <%- include('../partials/hero'); %>

    <!-- Products -->

    <footer class="py-5 bg-dark">
      <%- include('../partials/footer'); %>
    </footer>
  </body>
</html>
```

Save the changes to this file and then run the application. If you visit `http://localhost:8080/` in a web browser, you can observe the webpage:


EJS Shop

EJS Demo

Single page E-Commerce project

Copyright © AlmaBetter 2022

Creating Product Cards

Now let's create Cards to display products on the webpage. We'll start by creating `productCards.js` and enter the following code :

```
<div class="col mb-5">
  <div class="card h-100">
    <!-- Badge-->

    <!-- Product image-->
    <img
      class="card-img-top p-4 img-responsive"
      style="width: auto; height: 300px object-fit:cover;"
      src="<%= product.image %>"
      alt="..."
    />
    <!-- Product details-->
    <div class="card-body p-4">
      <div class="text-center">
        <!-- Product name-->
        <h5 class="fw-bolder"><%= product.title %></h5>
        <!-- Product price-->
        $<%= product.price %>
      </div>
    </div>
    <!-- Product actions-->
    <div class="card-footer p-4 pt-0 border-top-0 bg-transparent">
      <div class="text-center">
        <a class="btn btn-outline-dark mt-auto" href="#">
          Add to Cart
        </a>
      </div>
    </div>
  </div>
</div>
```

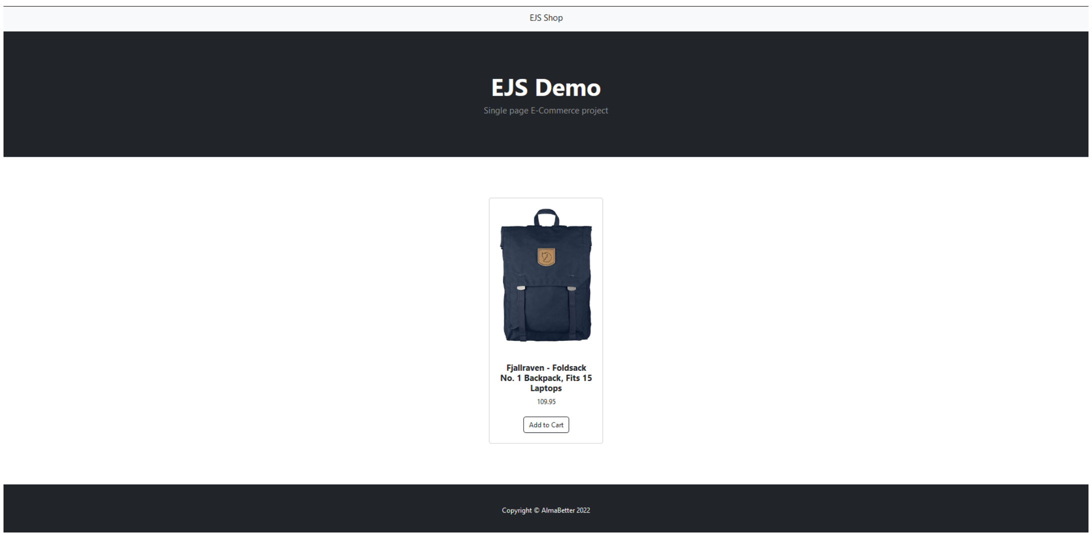
Now, update `index.ejs` to include this partial in it. It will look like :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <%- include('../partials/head'); %>
  </head>
  <body>
    <!-- Navigation-->
    <%- include('../partials/navbar'); %>
    <!-- Hero Section -->
    <%- include('../partials/hero'); %>

    <!-- Products -->
    <section class="py-5">
      <div class="container px-4 px-lg-5 mt-5">
        <div
          class="row gx-4 gx-lg-5 row-cols-2 row-cols-md-3 row-cols-xl-4 justify-content-center"
        >
          <%- include('../partials/productCard'); %>
        </div>
      </div>
    </section>

    <footer class="py-5 bg-dark">
      <%- include('../partials/footer'); %>
    </footer>
  </body>
</html>
```

At this point, the webpage will look like this:


EJS Shop

EJS Demo

Single page E-Commerce project

Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops

109.95

Copyright © AlmaBetter 2022

Passing Data to Views and Partials

Let's define some basic variables and a list to pass to the Index page. Create `data.js` file in your root directory and add the following code:

```
const data = [
  {
    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 sle
    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 ',
    price: 22.3,
    description:
      'Slim-fitting style, contrast raglan long sleeve, three-button henley placket, light weight & soft fabric for brea
    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, such as working, hiking, camping,
    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,
    description:
      'The color could be slightly different between on the screen and in practice. / Please note that body builds vary
    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.
    category: 'jewelery',
    image:
      'https://fakestoreapi.com/img/71pWzhdJNwL._AC_UL640_QL65_ML3_.jpg',
    rating: { rate: 4.6, count: 400 },
  },
  {
    id: 6,
    title: 'Solid Gold Petite Micropave ',
    price: 168,
    description:
      'Satisfaction Guaranteed. Return or exchange any order within 30 days.Designed and sold by Hafeez Center in the Un
    category: 'jewelery',
    image:
      'https://fakestoreapi.com/img/61sbMiUnoGL._AC_UL640_QL65_ML3_.jpg',
    rating: { rate: 3.9, count: 70 },
  },
  {
    id: 7,
    title: 'White Gold Plated Princess',
    price: 9.99,
    description:
      "Classic Created Wedding Engagement Solitaire Diamond Promise Ring for Her. Gifts to spoil your love more for Enga
    category: 'jewelery',
    image:
      'https://fakestoreapi.com/img/71YAIFU48IL._AC_UL640_QL65_ML3_.jpg',
    rating: { rate: 3, count: 400 },
  },
  {
    id: 8,
    title: 'Pierced Owl Rose Gold Plated Stainless Steel Double',
    price: 10.99,
    description:
      'Rose Gold Plated Double Flared Tunnel Plug Earrings. Made of 316L Stainless Steel',
    category: 'jewelery',
    image:
      'https://fakestoreapi.com/img/51UDEzMJVpL._AC_UL640_QL65_ML3_.jpg',
    rating: { rate: 1.9, count: 100 },
  },
  {
    id: 9,
    title: 'WD 2TB Elements Portable External Hard Drive - USB 3.0 ',
    price: 64,
    description:
      'USB 3.0 and USB 2.0 Compatibility Fast data transfers Improve PC Performance High Capacity; Compatibility Formatt
    category: 'electronics',
    image: 'https://fakestoreapi.com/img/61IBBVJvSDL._AC_SY879_.jpg',
    rating: { rate: 3.3, count: 203 },
  },
  {
    id: 10,
    title: 'SanDisk SSD PLUS 1TB Internal SSD - SATA III 6 Gb/s',
    price: 109,
    description:
      'Easy upgrade for faster boot up, shutdown, application load and response (As compared to 5400 RPM SATA 2.5” hard
    category: 'electronics',
    image: 'https://fakestoreapi.com/img/61U7T1koQqL._AC_SX679_.jpg',
    rating: { rate: 2.9, count: 470 },
  },
  {
    id: 11,
    title:
      'Silicon Power 256GB SSD 3D NAND A55 SLC Cache Performance Boost SATA III 2.5',
    price: 109,
    description:
      '3D NAND flash are applied to deliver high transfer speeds Remarkable transfer speeds that enable faster bootup an
    category: 'electronics',
    image: 'https://fakestoreapi.com/img/71kWymZ+c+L._AC_SX679_.jpg',
    rating: { rate: 4.8, count: 319 },
  },
  {
    id: 12,
    title:
      'WD 4TB Gaming Drive Works with Playstation 4 Portable External Hard Drive',
    price: 114,
    description:
      "Expand your PS4 gaming experience, Play anywhere Fast and easy, setup Sleek design with high capacity, 3-year man
    category: 'electronics',
    image: 'https://fakestoreapi.com/img/61mtL65D4cL._AC_SX679_.jpg',
    rating: { rate: 4.8, count: 400 },
  },
  {
    id: 13,
    title:
      'Acer SB220Q bi 21.5 inches Full HD (1920 x 1080) IPS Ultra-Thin',
    price: 599,
    description:
      '21. 5 inches Full HD (1920 x 1080) widescreen IPS display And Radeon free Sync technology. No compatibility for V
    category: 'electronics',
    image: 'https://fakestoreapi.com/img/81QpkIctqPL._AC_SX679_.jpg',
    rating: { rate: 2.9, count: 250 },
  },
  {
    id: 14,
    title:
      'Samsung 49-Inch CHG90 144Hz Curved Gaming Monitor (LC49HG90DMNXZA) – Super Ultrawide Screen QLED ',
    price: 999.99,
    description:
      '49 INCH SUPER ULTRAWIDE 32:9 CURVED GAMING MONITOR with dual 27 inch screen side by side QUANTUM DOT (QLED) TECHN
    category: 'electronics',
    image: 'https://fakestoreapi.com/img/81Zt42ioCgL._AC_SX679_.jpg',
    rating: { rate: 2.2, count: 140 },
  },
  {
    id: 15,
    title: "BIYLACLESEN Women's 3-in-1 Snowboard Jacket Winter Coats",
    price: 56.99,
    description:
      'Note:The Jackets is US standard size, Please choose size as your usual wear Material: 100% Polyester; Detachable
    category: "women's clothing",
    image: 'https://fakestoreapi.com/img/51Y5NI-I5jL._AC_UX679_.jpg',
    rating: { rate: 2.6, count: 235 },
  },
  {
    id: 16,
    title:
      "Lock and Love Women's Removable Hooded Faux Leather Moto Biker Jacket",
    price: 29.95,
    description:
      '100% POLYURETHANE(shell) 100% POLYESTER(lining) 75% POLYESTER 25% COTTON (SWEATER), Faux leather material for sty
    category: "women's clothing",
    image: 'https://fakestoreapi.com/img/81XH0e8fefL._AC_UY879_.jpg',
    rating: { rate: 2.9, count: 340 },
  },
  {
    id: 17,
    title: 'Rain Jacket Women Windbreaker Striped Climbing Raincoats',
    price: 39.99,
    description:
      "Lightweight perfet for trip or casual wear---Long sleeve with hooded, adjustable drawstring waist design. Button
    category: "women's clothing",
    image:
      'https://fakestoreapi.com/img/71HblAHs5xL._AC_UY879_-2.jpg',
    rating: { rate: 3.8, count: 679 },
  },
  {
    id: 18,
    title: "MBJ Women's Solid Short Sleeve Boat Neck V ",
    price: 9.85,
    description:
      '95% RAYON 5% SPANDEX, Made in USA or Imported, Do Not Bleach, Lightweight fabric with great stretch for comfort,
    category: "women's clothing",
    image: 'https://fakestoreapi.com/img/71z3kpMAYsL._AC_UY879_.jpg',
    rating: { rate: 4.7, count: 130 },
  },
  {
    id: 19,
    title: "Opna Women's Short Sleeve Moisture",
    price: 7.95,
    description:
      '100% Polyester, Machine wash, 100% cationic polyester interlock, Machine Wash & Pre Shrunk for a Great Fit, Light
    category: "women's clothing",
    image: 'https://fakestoreapi.com/img/51eg55uWmdL._AC_UX679_.jpg',
    rating: { rate: 4.5, count: 146 },
  },
  {
    id: 20,
    title: 'DANVOUY Womens T Shirt Casual Cotton Short',
    price: 12.99,
    description:
      '95%Cotton,5%Spandex, Features: Casual, Short Sleeve, Letter Print,V-Neck,Fashion Tees, The fabric is soft and has
    category: "women's clothing",
    image: 'https://fakestoreapi.com/img/61pHAEJ4NML._AC_UX679_.jpg',
    rating: { rate: 3.6, count: 145 },
  },
];
```

```
module.exports = { data };
```

Import this data into `server.js` and add the following code :

```
var express = require('express');
var app = express();
var { data } = require('./data');

// set the view engine to ejs
app.set('view engine', 'ejs');

// use res.render to load up an ejs view file

// index page
app.get('/', function (req, res) {
  res.render('pages/index', {
    products: data,
  });
});

app.listen(8080);
console.log('Server is listening on port 8080');
```

This code passes the `data` array to `index.ejs` as `products` .

To echo a single variable, you can use `<%= tagline %>` inside the ejs file. Here, we'll loop over this array and further pass the da
to `productCard.ejs` to display all the products.

Looping over Data in EJS

To loop over data, you can use `.forEach` .

Revisit `index.ejs` in your code editor and add the following lines of code:

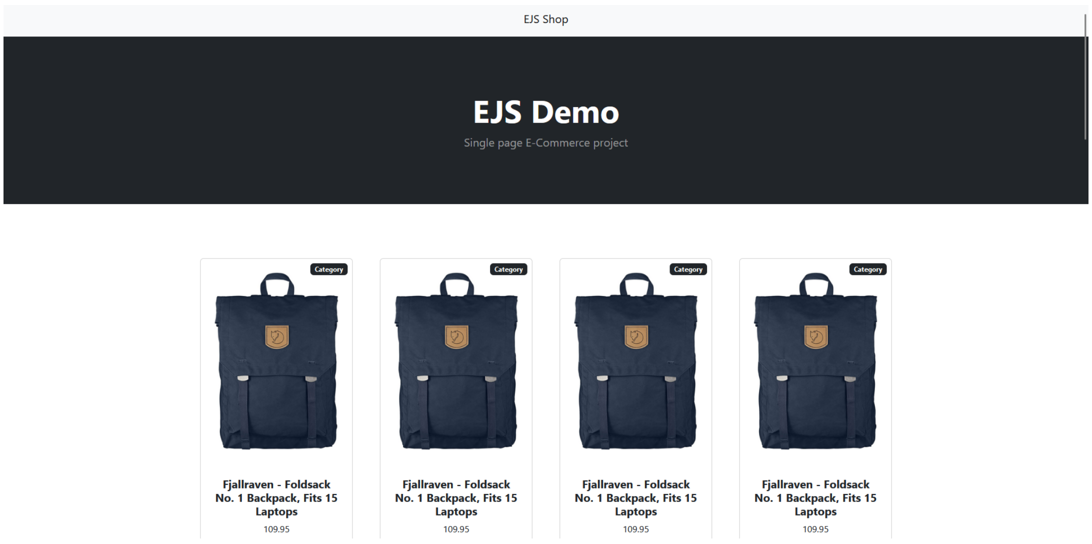
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <%- include('../partials/head'); %>
  </head>
  <body>
    <!-- Navigation-->
    <%- include('../partials/navbar'); %>

    <!-- Hero Section -->
    <%- include('../partials/hero'); %>

    <!-- Products -->
    <section class="py-5">
      <div class="container px-4 px-lg-5 mt-5">
        <div
          class="row gx-4 gx-lg-5 row-cols-2 row-cols-md-3 row-cols-xl-4 justify-content-center"
        >
          <% products.forEach(function (product){ %> <%-
          include('../partials/productCard', {product: 'product'}); %>
          <% }); %>
        </div>
      </div>
    </section>

    <footer class="py-5 bg-dark">
      <%- include('../partials/footer'); %>
    </footer>
  </body>
</html>
```

You'll notice we're displaying the same product 20 times like following:


EJS Shop
EJS Demo
Single page E-Commerce project
Category
Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops
109.95

This means we're looping over the entire array of data but displaying the same information each time since it is hard-coded in `productCard.ejs` We're already passing the product data to product cards using the following syntax:

```
<% products.forEach(function (product){ %>
    <%- include('../partials/productCard', {product: 'product'}); %>
<% }); %>
```

Let's use this variable in `productCard.ejs` to display the data dynamically. It's code will be:

```
<div class="col mb-5">
  <div class="card h-100">
    <!-- Badge-->
    <div
      class="badge bg-dark text-white position-absolute"
      style="top: 0.5rem; right: 0.5rem"
    >
      <%= product.category %>
    </div>
    <!-- Product image-->
    <img
      class="card-img-top p-4 img-responsive"
      style="width: auto; height: 300px object-fit:cover;"
      src="<%= product.image %>"
      alt="..."
    />
    <!-- Product details-->
    <div class="card-body p-4">
      <div class="text-center">
        <!-- Product name-->
        <h5 class="fw-bolder"><%= product.title %></h5>
        <!-- Product price-->
        $<%= product.price %>
      </div>
    </div>
    <!-- Product actions-->
    <div class="card-footer p-4 pt-0 border-top-0 bg-transparent">
      <div class="text-center">
        <a class="btn btn-outline-dark mt-auto" href="#">
          Add to Cart
        </a>
      </div>
    </div>
  </div>
</div>
```

Conditional Rendering in EJS

EJS lets you conditionally render HTML Elements using If-Else statements or ternary operators. Let's make use of this to give a `Highest Rated` badg
to all our products that are rated higher than `4.0` .

The syntax to use If-Else in EJS is as follows:

```
<% if (user) { %>
  <h2><%= user.name %></h2>
<% } %>
```

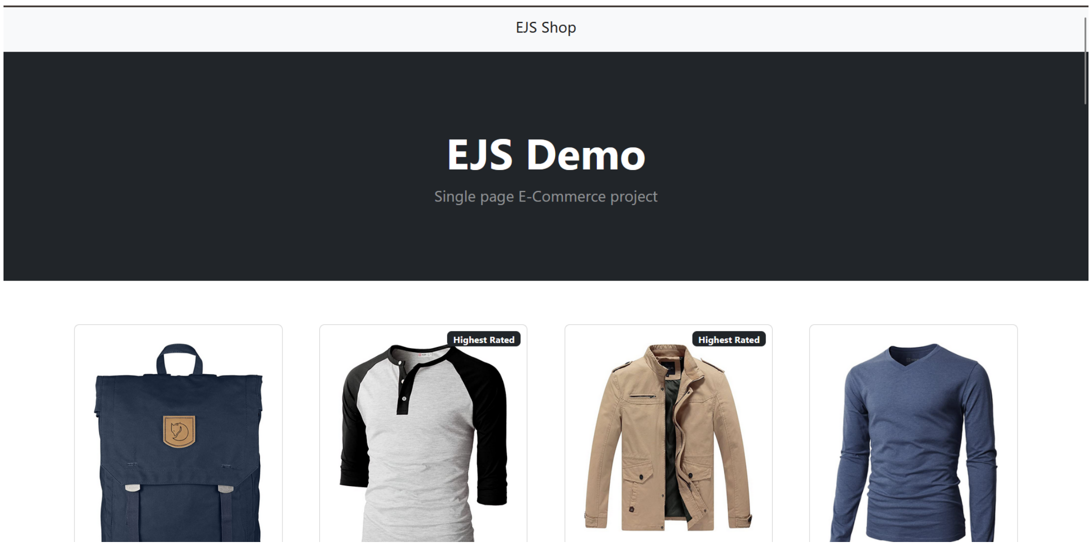
Following this syntax, update `productCard.ejs` to conditionally render the badge. It's code will look as follows:

```
<div class="col mb-5">
  <div class="card h-100">

    <!-- Badge-->
    <% if (typeof product.rating.rate != 'undefined' && product.rating.rate >= 4.0) { %>
      <div
        class="badge bg-dark text-white position-absolute"
        style="top: 0.5rem; right: 0.5rem"
      >
        Highest Rated
      </div>
    <% } %>

    <!-- Product image-->
    <img
      class="card-img-top p-4 img-responsive"
      style="width: auto; height: 300px object-fit:cover;"
      src="<%= product.image %>"
      alt="..."
    />
    <!-- Product details-->
    <div class="card-body p-4">
      <div class="text-center">
        <!-- Product name-->
        <h5 class="fw-bolder"><%= product.title %></h5>
        <!-- Product price-->
        $<%= product.price %>
      </div>
    </div>
    <!-- Product actions-->
    <div class="card-footer p-4 pt-0 border-top-0 bg-transparent">
      <div class="text-center">
        <a class="btn btn-outline-dark mt-auto" href="#">
          Add to Cart
        </a>
      </div>
    </div>
  </div>
</div>
```

Finally, this is what the webpage looks like:



Conclusion

In this session we learned about:

- Template Engines in Express
- EJS
- Creating `views` and `partials` using EJS
- Looping through data in EJS
- Conditional rendering in EJS

Interview Questions

> Name some of the Template Engines supported by Express.

Pug, Mustache, EJS and Handlebars are some of the most popular template engines used with Express.

> What is the default template engine used by Express?

Express uses Jade as its default template engine.

> Explain what a template engine is

A template engine allows you to use static template files in your application. At runtime, the template engine replaces variables in a template file wi actual values, and transforms the template into an HTML file to be sent to the client. This approach makes it easier to design HTML pages and workir with data.

Thank You !