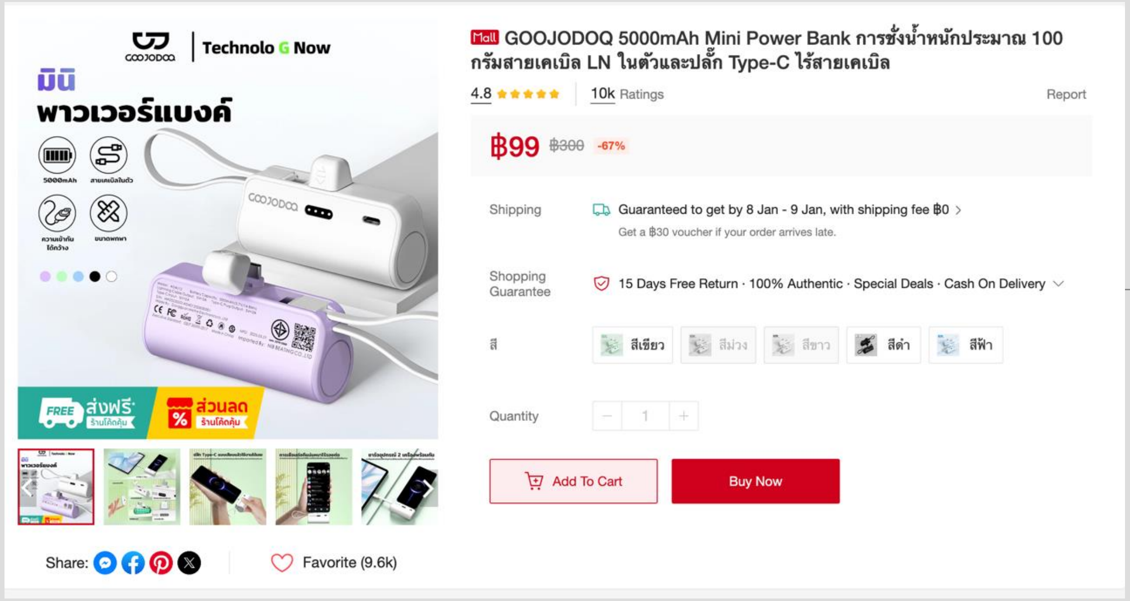
Chapter 06: Dynamic Templating with EJS

TEMPLATE + DATA → HTML (SERVER-SIDE RENDERING)

Outline

- **Dynamic template idea:** Template vs Data
- **Where rendering happens:** CSR vs SSR
- **EJS basics:** variables, if, loop, include
- **Mini demo:** 1 template → many product pages

Product Page 1



Mall GOOJODOQ 5000mAh Mini Power Bank การชั่งน้ำหนักประมาณ 100 กรัมสายเคเบิล LN ในตัวและปลั๊ก Type-C ไร้สายเคเบิล

4.8 ★★★★★ | 10k Ratings — Report

฿99 ~~฿300~~ -67%

Shipping: Guaranteed to get by 8 Jan - 9 Jan, with shipping fee ฿0 >
Get a ฿30 voucher if your order arrives late.

Shopping Guarantee: 15 Days Free Return · 100% Authentic · Special Deals · Cash On Delivery

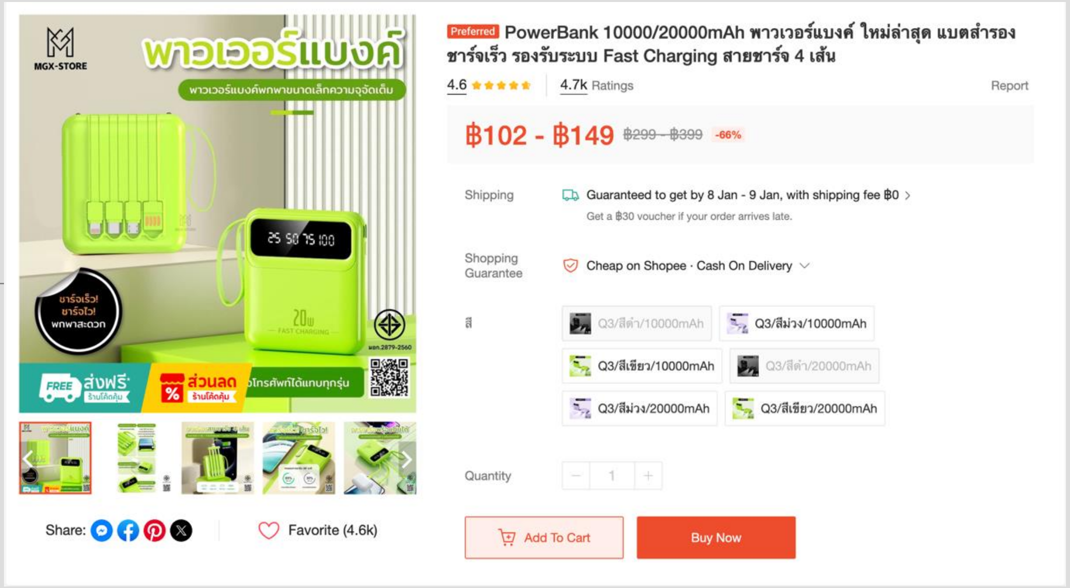
สี: สีเขียว | สีม่วง | สีขาว | สีดำ | สีฟ้า

Quantity: − 1 +

Add To Cart | Buy Now

Share: | Favorite (9.6k)

Product Page 2



Preferred PowerBank 10000/20000mAh พาวเวอร์แบงค์ ใหม่ล่าสุด แบตสำรอง ชาร์จเร็ว รองรับระบบ Fast Charging สายชาร์จ 4 เส้น

4.6 ★★★★★ | 4.7k Ratings — Report

฿102 - ฿149 ~~฿299 - ฿399~~ -66%

Shipping: Guaranteed to get by 8 Jan - 9 Jan, with shipping fee ฿0 >
Get a ฿30 voucher if your order arrives late.

Shopping Guarantee: Cheap on Shopee · Cash On Delivery

สี: Q3/สีดำ/10000mAh | Q3/สีม่วง/10000mAh | Q3/สีเขียว/10000mAh | Q3/สีดำ/20000mAh | Q3/สีม่วง/20000mAh | Q3/สีเขียว/20000mAh

Quantity: − 1 +

Add To Cart | Buy Now

Share: | Favorite (4.6k)

What parts are the same layout/structure?

Which parts change per product?

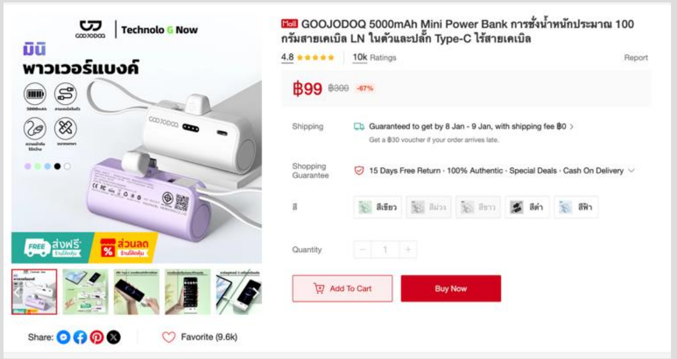
SSR idea (EJS in Express):

Server prepares **data object**
EJS combines **template + data**
Server sends **final HTML** to browser

product.ejs

app.ts

```
res.render("product", { product })
```


Mall GOOJODOQ 5000mAh Mini Power Bank การชั่งน้ำหนักประมาณ 100 กรัมสายเคเบิล LN ในตัวและปลั๊ก Type-C ไร้สายเคเบิล

4.8 ★★★★★ | 10k Ratings | Report

฿99 ~~฿300~~ -67%

Shipping: Guaranteed to get by 8 Jan - 9 Jan, with shipping fee ฿0
Get a ฿30 voucher if your order arrives late.

Shopping Guarantee: 15 Days Free Return · 100% Authentic · Special Deals · Cash On Delivery

สี: สีเขียว | สีม่วง | สีขาว | สีดำ | สีฟ้า

Quantity: 1

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title><%= product.title %></title>
</head>

<body>
  <header>
    <nav>
      <a href="/">Shop</a>
      <a href="/category/powerbank">Powerbanks</a>
    </nav>
  </header>

  <main>
    <article>
      <section aria-label="Product layout">
        <!-- Left: media -->
        <section aria-label="Product images">
          <figure>
            <img src="<%= product.images[0] %>" alt="<%= product.title %>" />
          </figure>

          <ul aria-label="Thumbnails">
            <% product.images.forEach((img) => { %>
              <li><img src="<%= img %>" alt="" /></li>
            <% }) %>
          </ul>
        </section>

        <!-- Right: info + purchase -->
        <aside aria-label="Product purchase info">
          <h1><%= product.title %></h1>

          <section aria-label="Rating">
            <span><%= product.rating %></span>
            <span>(<%= product.ratingCount %> ratings)</span>
          </section>

          <section aria-label="Price">
            <% if (product.priceMin !== product.priceMax) { %>
              <strong>฿<%= product.priceMin %> - ฿<%= product.priceMax %></strong>
            <% } else { %>
              <strong>฿<%= product.priceMin %></strong>
            <% } %>
          </section>
```

Data object

Loop

If-else condition

```
          <section aria-label="Shipping">
            <p><%= product.shippingText %></p>
          </section>

          <form action="/cart/add" method="post">
            <input type="hidden" name="productId" value="<%= product.id %>" />

            <section aria-label="Variants">
              <h2>Options</h2>
              <ul>
                <% product.variants.forEach((v) => { %>
                  <li>
                    <label>
                      <input type="radio" name="variantId" value="<%= v.id %>" />
                      <span><%= v.label %></span>
                    </label>
                  </li>
                <% }) %>
              </ul>
            </section>

            <section aria-label="Quantity">
              <label>
                Quantity
                <input type="number" name="qty" value="1" min="1" />
              </label>
            </section>

            <section aria-label="Actions">
              <button type="submit" name="action" value="add">Add to Cart</button>
              <button type="submit" name="action" value="buy">Buy Now</button>
            </section>
          </form>
        </aside>
      </section>

      <section aria-label="Product details">
        <h2>Description</h2>
        <p><%= product.description %></p>
      </section>
    </article>
  </main>

  <footer>
    <small>© Shop</small>
  </footer>
</body>
</html>
```

Rendering: CSR vs SSR

Client-side rendering

When:

- UI is highly interactive (filters, live search, dashboards)
- Updates happen without full page reload

Technologies:

- Fetch API + DOM manipulation
- React / Vue / Angular / Svelte

Server-side rendering

When:

- Fast first page load (HTML ready immediately)
- SEO / share preview important
- Content-driven pages (catalog, product detail)

Technologies:

- EJS / Pug / Handlebars
- Express: res.render("view", data)

Most real apps are Hybrid: SSR for initial page + CSR for dynamic parts (Fetch + update DOM).

What is a Dynamic Template?

Template = HTML structure (layout)

Data = values that change (title, price, images, variants)

Rendering = combine Template + Data → HTML

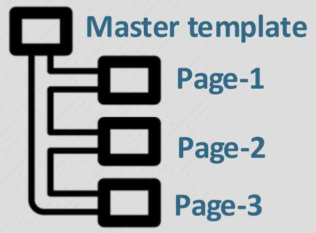
Same template, many pages

- Product A uses data A
- Product B uses data B
- Layout stays the same

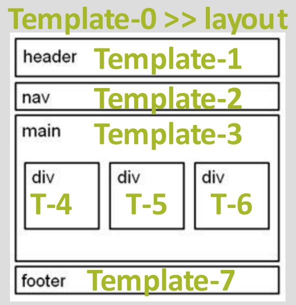
Uses of Dynamic HTML Template

Template can be applied to various levels:

1. **Dynamic by webpages**
   - To keep the same look of the whole website that has several pages. Template can be used as a mastered copy for pages in the website. One template can be applied to several pages.
   - (similar to the use of CSS file) *What's the difference between CSS file and Template file?*



2. **Dynamic by page partials**
   - One webpage can use several templates. One page can have several parts e.g. header/ menu bar / several block of contents/ footer. Each part can have its own template convenient for changes in the partial and preventing changes to other partials. Changing the region template will effect to every page that uses the template.



3. **Dynamic by the contents**
   - The web content/ information can be added/ edited/ removed by users or events.

Need of Dynamic HTML Template

To serve the concept of website design

- Consistent look throughout the website – *top consideration!*
  - to maintain a consistent look and feel for website

- Consistent navigation throughout the website
  - to easily navigate basic pages such as Homepage, Sitemap, Contact, Search and About
  - visitor should access these pages from any page on the site.

- Web maintenance in content/ style/ layout/ interaction
  - easily maintain consistency of web elements throughout the website
  - a change to template effects to every pages using the template

https://partyspacedesign.com/
https://www.plaimanas.com/ **web profile of professional web designer
https://pyvetstudio.com/

Embedded JavaScript Template

Embedded JavaScript Template (EJS) is a simple templating language that lets you generate HTML markup with plain JavaScript.

https://ejs.co/

Feature of EJS Module:

1. Use plain JavaScript.
2. Fast Development time.
3. Simple syntax.
4. Faster execution.
5. Easy Debugging.
6. Active Development.

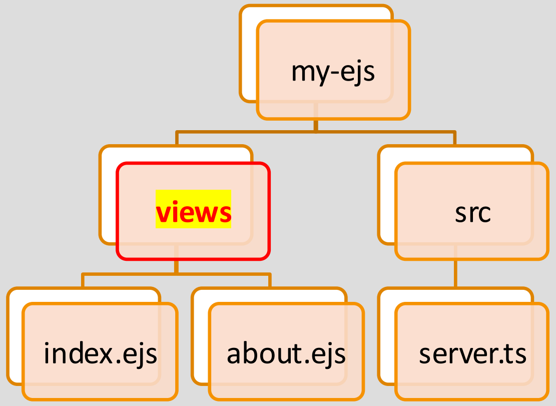
<%= EJS %>

Embedded JavaScript templating.

Implementation with EJS Template

Step 1 — Setting Up the Project

- Create a new folder: my-ejs
- Initiate the folder with npm
- Install dependencies (express and ejs)
- Install dev dependencies (TypeScript, types)
- Configure TypeScript
- Create files/folders



Step 2 — src/server.ts (Express + EJS)

```
import express from "express";
import path from "path";
const app = express();
const PORT = 8000;

// Tell Express to use EJS
app.set("view engine", "ejs");

// Important: make views folder work after compiling to /dist
app.set("views", path.join(__dirname, "..", "views"));

app.get("/", (req, res) => {
  res.render("index", { title: "Home" });
});

app.get("/about", (req, res) => {
  res.render("about", { title: "About" });
});

app.listen(PORT, () => {
  console.log(`Server is listening on port ${PORT}`);
});
```

Step 3 — views/index.ejs

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title><%= title %></title>

<!-- CSS (load bootstrap from a CDN) -->
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.5.2/css/bootstrap.min.css">
<style>
 body { padding-top:50px; }
</style>
</head>
<body class="container">
<main>
 <div class="jumbotron">
  <h1>Hello World</h1>
  <p>Here's the main template using EJS</p>
 </div>
</main>
</body>
</html>
```

`app.set(property, value)`: set Express app settings

`view engine`: choose template engine (**EJS**)
`res.render("view", data)`: render an EJS file from the **views folder** and send HTML to the client

EJS Basics

Output value

```
<h1><%= product.title %></h1>
```

Condition (if)

```
<% if (product.discountPercent) { %>
  <span>Sale!</span>
<% } %>
```

Loop (for each)

```
<% product.images.forEach(img => {
%>
  <img src="<%= img %>">
<% }) %>
```

Include partial

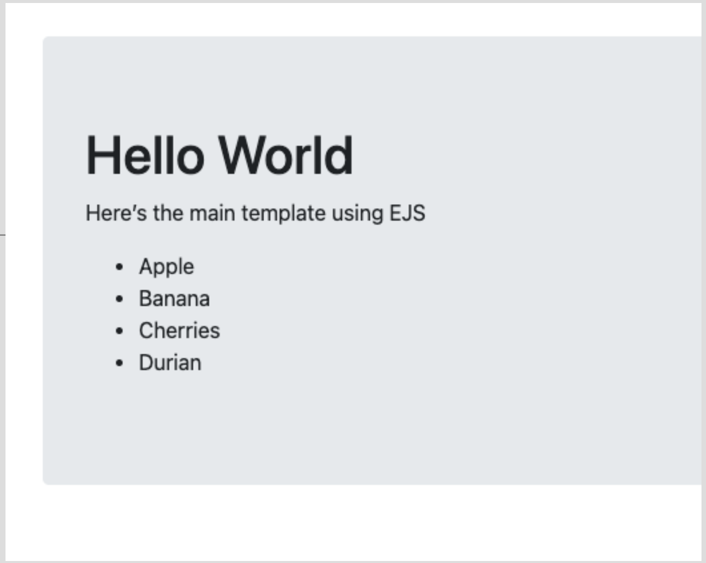
```
<%- include("partials/header") %>
```

<%- ... %> (unescaped output)

Tells EJS to output the included result as **raw HTML**

Quick exercise (5 min):

Render a list of courses or products from an array passed by route.



Hello World

Here's the main template using EJS

- Apple
- Banana
- Cherries
- Durian

Modify server.ts

```
app.get("/", function (req, res) {
const fruits = [
{ name: "Apple", color: "red" },
{ name: "Banana", color: "yellow" },
{ name: "Cherries", color: "red" },
{ name: "Durian", color: "yellow" },
];

res.render("index", {
      title: "Home",
      fruits: fruits
});
});
```

Modify index.ejs

```
<body class="container">
   <main>
     <div class="jumbotron">
       <h1>Hello World</h1>
       <p>Here’s the main template using EJS</p>
      <ul>
         <% fruits.forEach((fruit) => { %>
         <li><%= fruit.name %> is <%= fruit.color %></li>
         <% }); %>
      </ul>
    </div>
   </main>
</body>
```

Partials (Reusable Layout)

Problem: header/footer repeated in every page

Solution: partials

Files:

- views/partials/header.ejs
- views/partials/footer.ejs

Use: an EJS include that inserts another template file (a partial) into the current page.

```
<%- include("partials/header", { title: product.title }) %>
...
<%- include("partials/footer") %>
```

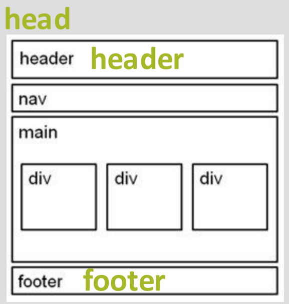
Basic layout pattern (manual):
header include
page content
footer include

Quick exercise (5 min):

Add head, header and footer partials to the current page.

Creating the EJS Template and its Partials

- Inside the **my-ejs/views** folder, modify **index.ejs** as shown below
- Inside the **my-ejs/views** folder, create a new folder **partials**
- Inside the **my-ejs/views/partials** folder, create **head.ejs, header.ejs , footer.ejs** as shown below



index.ejs

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

<header>
 <%- include('partials/header'); %>
</header>

<main>
 <div class="jumbotron">
  <h1>Hello World</h1>
  <p>Here’s the main template using EJS</p>
  <ul>
     <% fruits.forEach((fruit) => { %>
     <li><%= fruit.name %> is <%= fruit.color
%></li>
     <% }); %>
   </ul>
 </div>
</main>

<footer>
 <%- include('partials/footer'); %>
</footer>

</body>
</html>
```

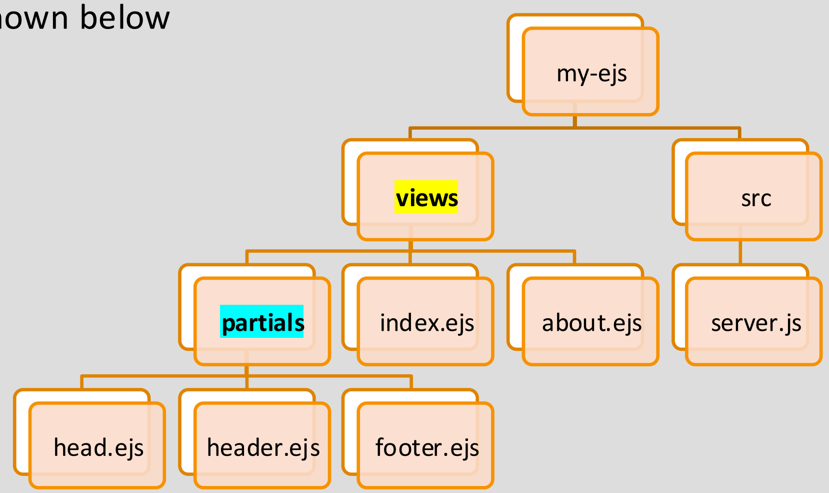
head.ejs

```
<meta charset="UTF-8">
<title>Hello EJS</title>

<!-- CSS (load bootstrap from a CDN) -->
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/tw
itter-bootstrap/4.5.2/css/bootstrap.min.css">
<style>
 body { padding-top:50px; }
</style>
```

header.ejs

```
<nav class="navbar navbar-expand-lg navbar-
light bg-light">
 <a class="navbar-brand" href="/">Hello
EJS</a>
 <ul class="navbar-nav mr-auto">
  <li class="nav-item">
   <a class="nav-link" href="/">Home</a>
  </li>
  <li class="nav-item">
   <a class="nav-link"
href="/about">About</a>
  </li>
 </ul>
</nav>
```



footer.ejs

```
<p class="text-center text-muted">&copy; Copyright 2020 </p>
```

One Route, One Template, Many Products

URL pattern:

- /product/:id

Flow:

- read id from URL
- find product data (DB / array)
- render with same template

```
app.get("/product/:id", (req, res) => {
  const product = findProduct(req.params.id);
  res.render("product", { product });
});
```

Mini feature build (hands-on)

Feature: “ Fruit List”

Route 1: /fruits renders list

Route 2: /fruits/:id renders detail

Use:

- loop in list page
- conditional for “not found” case
- link building in EJS

Deliverable:

Working list/detail navigation.

```
app.get("/fruits/:id", function (req, res) {
const fruitId = parseInt(req.params.id);
const fruits = [
{ id: 1, name: "Apple", color: "red", description: "An apple a day keeps the doctor
away." },
{ id: 2, name: "Banana", color: "yellow", description: "Bananas are high in
potassium." },
{ id: 3, name: "Cherries", color: "red", description: "Cherries are delicious and
rich in antioxidants." },
{ id: 4, name: "Durian", color: "yellow", description: "Durian is known as the king
of fruits." },
];


 if (fruitId >= 1 && fruitId < fruits.length) {
     res.render("fruitDetail", {
     fruit: fruits[fruitId],
   });
  } else {
     res.status(404).send("Fruit not found");
   }
});
```

Common pitfalls

- **View not found**
  - check views/ path + file name (res.render("index") → views/index.ejs)
- **res.send() vs res.render()**
  - send = send text/JSON, render = EJS template → HTML
- **Template variable undefined**
  - make sure you pass it in res.render("page", { ... })
- **<%- %> with user input**
  - can cause XSS (use <%= %> by default)
- **Too much logic in EJS**
  - keep logic in route/controller, EJS = display only

Exit questions

1. When choose EJS over SPA?
2. <%= vs <%-?
3. How to reuse header/footer?