What EJS Actually Is

EJS (“Embedded JavaScript”) is just **HTML + short snippets of JavaScript** that Express renders *before* sending the page to the browser.

Think of it as a “smart HTML file” that can:

include reusable pieces like a header or footer

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

output variables from your controller

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

loop or conditionally render content

```
<ul>
  <% for (const item of items) { %>
    <li><%= item.name %></li>
  <% } %>
</ul>
```

If you remove the `<% … %>` tags, what’s left is valid HTML.

So for **Deliverable 1**, You’re basically writing HTML pages—but placed inside `.ejs` files so later (when controllers pass data) those same pages can show dynamic content.

How Deliverable 1 Fits the Architecture

From `ARCHITECTURE.md`:

- Views (EJS) live in `src/views/`
- Controllers (JS) live in `src/controllers/`
- Routes point a URL to a controller

For Week 7, You only touch the **views** layer.

Each page (Home, About, Feature, etc.) is its own `.ejs` file in `src/views/`.
The app will route users to those pages using Express, e.g.:

```
// src/routes/index.js
router.get('/', (req, res) => res.render('index', { title: 'Home'
}));
router.get('/about', (req, res) => res.render('about', { title:
'About' }));
router.get('/feature', (req, res) => res.render('feature', {
title: 'Feature' }));
```

That means:

- When someone visits `/`, Express renders `views/index.ejs`.
- When you click “About”, it renders `views/about.ejs`.
- No JavaScript logic or database calls yet.

What You Should Actually Be Doing

Create three `.ejs` files under `src/views/`
Example:

```
index.ejs (which is your landing page, should have some context)
feature.ejs
feature2.ejs (or this can be an about, a signup, a contact page, etc)
```

1. **Add standard HTML markup** in those files.
   You can ignore `<% %>` for now—just put the page structure there.
   (E.g., headings, paragraphs, images, nav bar, footer.)

2. **Style the pages** in `/public/css/style.css`.
3. **Set up basic navigation** so links in the header let you switch pages.
4. **Run the app locally** (`npm run dev`) and confirm you can navigate between pages.
5. **Use Git workflow:**
    - Create branch `feature/week7-layout` from main (1 branch per group per week/ submission)
    - If partners: start a liveshare session, share the link with a partner, pair program over a call or side by side.
    - Commit changes (if liveshare, only host can do this). Make sure the commit messages are meaningful
    - Open PR, fill out template → Request review from another team in Teams so I can see it (assignments in the table on Canvas)
    - Merge to `main`
    - Submit PR link on Canvas (I will grade what is in `main` only)

Deliverable 1 Summary (Simplified)

Goal: Create the static front-end foundation of your app using HTML and CSS inside EJS templates.

At the end of the week you should have:

- 3 working EJS pages served by Express routes
- Navigation between them
- Basic styling (hopefully modified from what you were given; I imagine not everyone wants the same styling)
- Proper PR + peer review merged into main

- Updated README describing your pages

Extras

https://www.youtube.com/watch?v=NGXbXI26Ejk

https://www.youtube.com/watch?v=SccSCuHhOw0&t=436s

https://www.youtube.com/watch?v=8zdyhDmqzQ0