

Server side rendering

CSR vs SSR

- **CSR (Client Side Rendering):** usa JavaScript per modificare dinamicamente il DOM nel browser.
 - Vantaggi: interfacce interattive (es. social).
 - Svantaggi: prestazioni scarse con grandi volumi di dati.
- **SSR (Server Side Rendering):** il server genera l'HTML completo prima di inviarlo.
 - Vantaggi: più semplice, ottimo per SEO.
 - Usato nei marketplace (Amazon, eBay...).

Cos'è il Templating?

- Il templating è una tecnica che permette di generare pagine web dinamiche combinando codice HTML con codice JavaScript.
- **Vantaggi:**
 - **Riduzione del codice duplicato:** evita la ripetizione di strutture comuni come header, footer e menu di navigazione.
 - **Facilità di manutenzione:** modifiche a componenti comuni si riflettono su tutte le pagine.
 - **Velocità di sviluppo:** consente di concentrarsi sul contenuto specifico di ogni pagina senza riscrivere il layout generale.

EJS Embedded JavaScript Templates

- **EJS** è un motore di template per Node.js che permette di **inserire codice JavaScript direttamente in HTML**.
- Estensione dei file: **.ejs**
- Vantaggi
 - Leggero e veloce da configurare.
 - Ottimo per **SSR semplice e immediato**.
 - Si integra facilmente con **Express.js**.
 - Nessuna configurazione complicata come nei framework frontend (es. React, Vue).

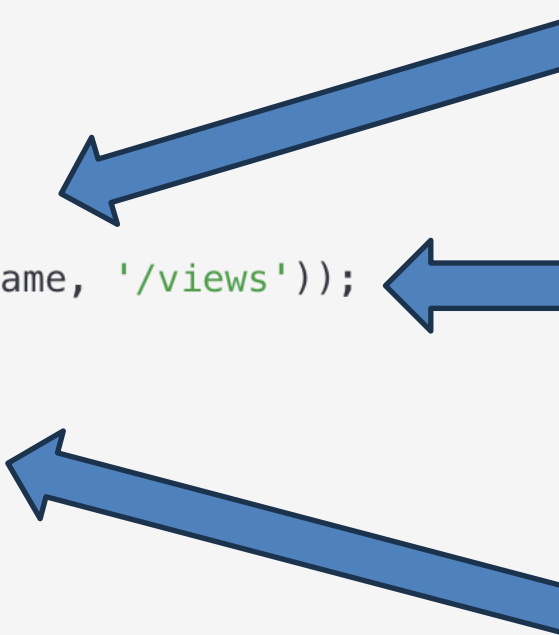
Express ed EJS

```
const express = require('express');
const app = express();
const port = 3000;
const path = require('path');

app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, '/views'));

app.get('/', (req, res) => {
  res.render('homepage.ejs');
});

app.listen(port, () => {
  console.log(`Server in esecuzione sulla porta ${port}`);
});
```



Imposta **EJS** come motore di template: i file .ejs saranno usati per generare HTML dinamico.

Specifica la directory `views/` come posizione in cui Express cercherà i file .ejs.

Quando l'utente visita `/`, il server **renderizza** la view `index.ejs`

EJS file

```
<!DOCTYPE html>
<html lang="it">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Templating in Node JS</title>
</head>
<body>
  <h1>Benvenuto su EJS in Node JS</h1>
</body>
</html>
```



Dove sta: views/homepage.ejs



Nome del file: homepage.ejs

EJS Tag

Tag	Descrizione
<code><% %></code>	Scriptlet: per controllo di flusso (es. if, for), nessun output
<code><%- %></code>	Output non HTML-escaped (per inserire HTML grezzo)
<code><%= %></code>	Output HTML-escaped (per sicurezza)
<code><%# %></code>	Commento: non viene eseguito né mostrato
<code><%%</code>	Stampa un simbolo % letterale
<code><%_ %></code>	Slurping iniziale: rimuove tutti gli spazi prima del tag
<code><%_</code>	Trim-mode (newline slurp): rimuove il newline successivo
<code>_ %></code>	Slurping finale: rimuove tutti gli spazi dopo il tag
<code>%></code>	Tag di chiusura normale

Esempio

contenuto di views/esempio.ejs:

```
<!DOCTYPE html>
<html lang="it">
<head>
  <meta charset="UTF-8">
  <title>Esempio EJS</title>
</head>
<body>

  <!-- <% %> Scriptlet: struttura di controllo -->
  <% if (notifiche > 0) { %>
    <p>Hai <%= notifiche %> nuove notifiche.</p>
  <% } else { %>
    <p>Nessuna nuova notifica.</p>
  <% } %>

  <!-- <%= %> Output HTML-escaped -->
  <p>Utente: <%= utente %></p>

  <!-- <%- %> Output non HTML-escaped -->
  <p><%- messaggio %></p>

</body>
</html>
```

```
res.render('esempio', {
  utente: 'Mario',
  messaggio: '<strong>Benvenuto!</strong>',
  notifiche: 3
});
```

- `res.render('esempio', { ... })`: renderizza la view `esempio.ejs` passando dei dati.
- `utente`: 'Mario': variabile utente con valore stringa.
- `messaggio`: 'Benvenuto!': messaggio HTML da stampare **non-escaped** con `<%- %>`.
- `notifiche`: 3: numero intero, può essere usato in condizioni (if) o per mostrare notifiche.



Loop EJS

```
<ul>
  <% products.forEach(product => { %>
    <li><%= product.name %>: ₹<%= product.price %></li>
  <% }) %>
</ul>
```

```
const products = [
  { name: "Laptop", price: 75000 },
  { name: "Smartphone", price: 40000 },
  { name: "Tablet", price: 25000 }
];
```

- Il ciclo `forEach` **scorre ogni oggetto** dentro l'array `products`.
- Per **ogni prodotto**, EJS stampa una riga come questa:

```
<li>Laptop: ₹75000</li>
<li>Smartphone: ₹40000</li>
<li>Tablet: ₹25000</li>
```

`<% ... %>` = Esegue codice JavaScript **senza stampare nulla**

`<%= ... %>` = Esegue codice **e stampa il risultato nel documento HTML**

EJS partials

```
views/
|
| └─ partials/
|     └─ header.ejs
|
| └─ products.ejs
| └─ contact.ejs
```

```
<header>
  <h1>My Shop</h1>
  <nav>
    <a href="/products">Products</a> |
    <a href="/contact">Contact</a>
  </nav>
  <hr>
</header>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Products</title>
</head>
<body>
  <% include partials/header %>

  <h2>Our Products</h2>
  <ul>
    <% products.forEach(product => { %>
      <li><%= product.name %>: ₹<%= product.price %></li>
    <% }) %>
  </ul>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Contact Us</title>
</head>
<body>
  <% include partials/header %>

  <h2>Contact Us</h2>
  <p>Email: info@myshop.com</p>
</body>
</html>
```