Intro to Web Server

Module 3 Week 6

Notes Repo: https://github.com/C-Shi/lhl-flex-lecture

Learning Objectives

Intro to Web Server

Express framework

Middleware

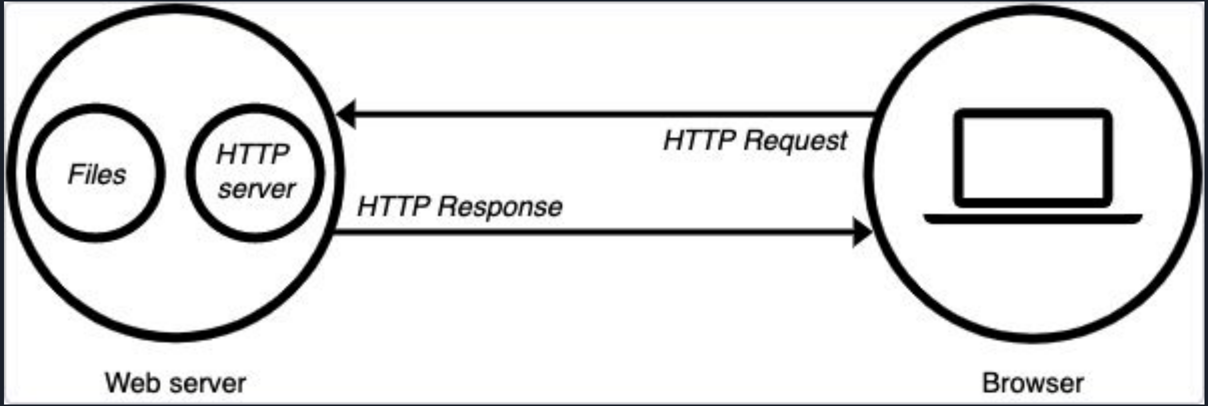
Template Engine

Web Server

Control how web users access hosted file

Software that understands URLs and HTTP

Act as a gateman between client and the resources



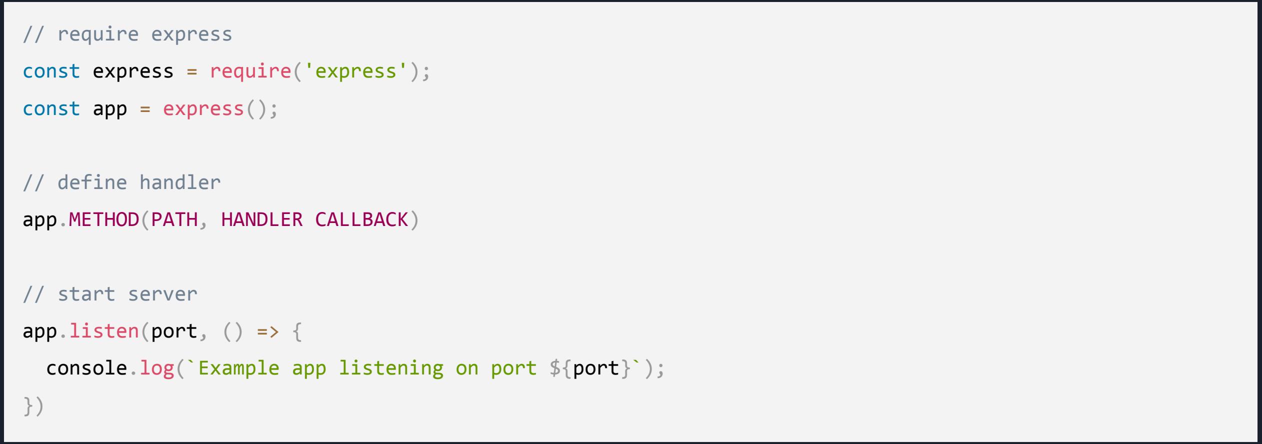
Express.js

A *lightweight*, node.js web framework

The main use of *Express* is to simplify the creation of route handlers

Easy Configuration:

1. Create an express instance
2. Define route and handler
3. Listen to a port



```
// require express
const express = require('express');
const app = express();

// define handler
app.METHOD(PATH, HANDLER CALLBACK)

// start server
app.listen(port, () => {
  console.log(`Example app listening on port ${port}`);
})
```

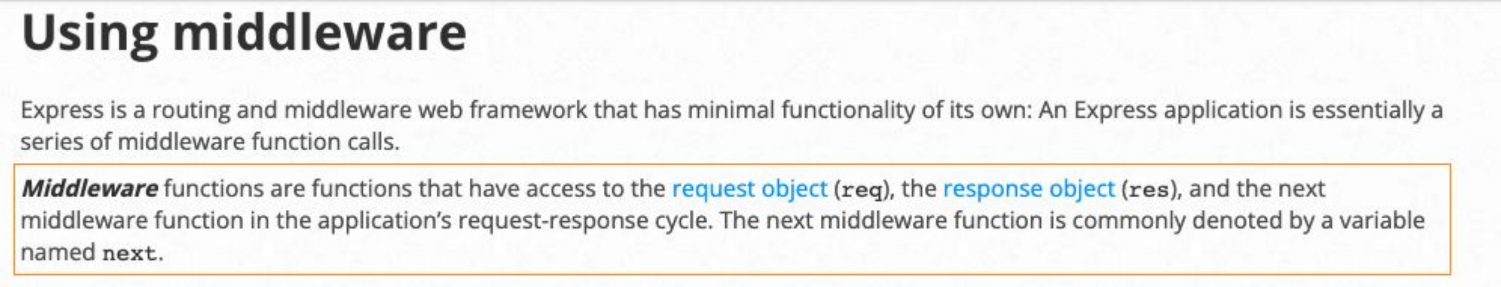
Middleware

Middleware is functions that run between Request and Response

Express itself provide very minimum amount of functionality, middleware make it powerful

Global middleware and *route specific* middleware

Middleware package and custom middleware



Using middleware

Express is a routing and middleware web framework that has minimal functionality of its own: An Express application is essentially a series of middleware function calls.

Middleware functions are functions that have access to the request object (`req`), the response object (`res`), and the next middleware function in the application's request-response cycle. The next middleware function is commonly denoted by a variable named `next`.

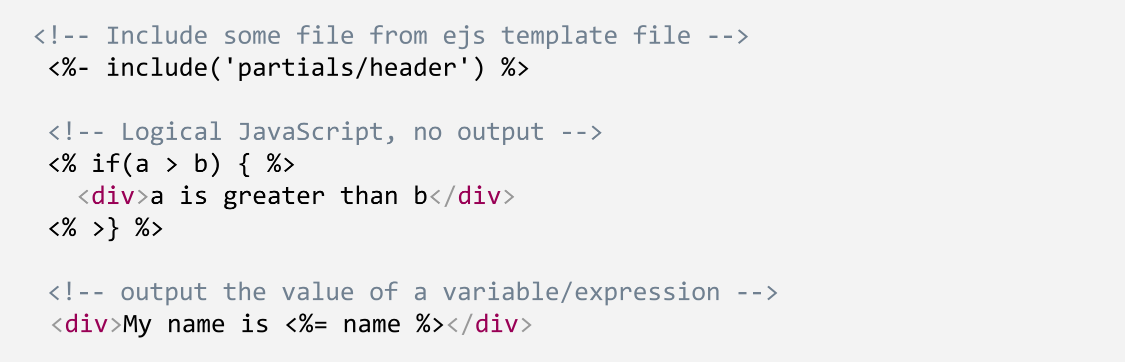
Template Engine

A processor to combine view template and data model

Dynamically generate HTML

Allow partials for reusable HTML

EJS (Embedded JavaScript)



```
<!-- Include some file from ejs template file -->
 <%- include('partials/header') %>

 <!-- Logical JavaScript, no output -->
 <% if(a > b) { %>
   <div>a is greater than b</div>
 <% >} %>

 <!-- output the value of a variable/expression -->
 <div>My name is <%= name %></div>
```