Q1. What is Express.js?

﻿Express.Js is a popular open source web framework built on top of Node.Js that simplifies the development of net packages and APIs.  It’s often known as the de-facto popular framework for Node.Js development.

Here are a few basic stuff you want to recognize about Express.Js.

Efficient and Minimal : Express.Js is designed to be rapid and intuitive, offering fundamental capabilities that you can personalize and expand.

Routing: One of the key features of Express.Js is its strong routing scheme. This lets in you to outline specific methods on your application and handle specific HTTP verb statements (GET, POST, PUT, DELETE) for each method.

Middleware: Middleware implementation is a powerful idea in Express.Js.  These are a sequence of duties for each request, permitting you to perform not unusual duties consisting of logging, facts evaluation, or authentication.

API Development: Express.Js is a exquisite device for building RESTful APIs, which is a not unusual way to structure statistics for net packages.

Overall, Express.Js simplifies the development manner of Node.Js web packages and APIs with the aid of offering a structured framework and assisting functions.

Q2 .Create localhost server using express.

1. **Install Node.js and npm**: Ensure you have Node.js and npm (Node Package Manager) installed on your system. You can download them from [Node.js official website](https://nodejs.org/).
2. **Initialize a new Node.js project**: Create a new directory for your project and initialize it with npm init.
3. **Install Express.js**: Install Express.js using npm.
4. **Create the server file**: Write the code to create and run the Express.js server.

Here’s a step-by-step guide:

**Step 1: Install Node.js and npm**

Download and install Node.js from [here](https://nodejs.org/).

**Step 2: Initialize a New Node.js Project**

Create a new directory for your project and navigate into it. Then, initialize a new Node.js project:

bash

Copy code

mkdir my-express-app

cd my-express-app

npm init -y

**Step 3: Install Express.js**

Install Express.js using npm:

bash

Copy code

npm install express

**Step 4: Create the Server File**

Create a file named server.js in your project directory and add the following code:

javascript

Copy code

// Import Express

const express = require('express');

const app = express();

// Define a port

const PORT = 3000;

// Define a route

app.get('/', (req, res) => {

res.send('Hello, World!');

});

// Start the server

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

**Step 5: Run the Server**

Run your server using Node.js:

bash

Copy code

node server.js

You should see the message "Server is running on http://localhost:3000" in your terminal. You can now open your browser and navigate to http://localhost:3000 to see the "Hello, World!" message.

Q4.Create weather app.

i. Use Hbs.

ii. Get country, state, city using api

iii. Get weather data by city

This example builds a basic weather app using Express.js for the server, Handlebars for templating, and an external weather API to fetch weather data for a specified city.

**Note:** You'll need to replace YOUR\_API\_KEY with your own API key from a weather service provider like OpenWeatherMap (<https://openweathermap.org/api>).

**1. Project Setup:**

* Create a project directory: mkdir weather-app && cd weather-app
* Initialize project: npm init -y

**2. Install Dependencies:**

* Install Express and Handlebars: npm install express express-handlebars

**3. Folder Structure:**

weather-app/

- views/ (folder for Handlebars templates)

- index.hbs

- public/ (folder for static assets like CSS/JS)

- app.js (server code)

- package.json

**4. app.js (Server Code):**

JavaScript

const express = require('express');

const handlebars = require('express-handlebars');

const fetch = require('node-fetch'); // for API requests

const app = express();

const port = 3000;

// Configure Handlebars

app.engine('hbs', handlebars({ extname: '.hbs' }));

app.set('view engine', 'hbs');

// Serve static files from public directory

app.use(express.static('public'));

// Route to render the main page (index.hbs)

app.get('/', async (req, res) => {

// Default empty object for weather data

const weatherData = {};

// If city is provided in query string, fetch weather data

if (req.query.city) {

const city = req.query.city;

const apiKey = 'YOUR\_API\_KEY'; // Replace with your API key

const weatherUrl = `https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&APPID=${apiKey}`;

try {

const response = await fetch(weatherUrl);

const weatherData = await response.json();

} catch (error) {

console.error(error);

}

}

res.render('index', { weatherData }); // Pass weather data to template

});

app.listen(port, () => {

console.log(`Server listening on port ${port}`);

});

Use code [with caution.](https://d.docs.live.net/faq#coding)

content\_copy

**5. views/index.hbs (Handlebars Template):**

HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Weather App</title>

</head>

<body>

<h1>Weather App</h1>

<form action="/" method="GET">

<label for="city">Enter city name:</label>

<input type="text" id="city" name="city" placeholder="City name">

<button type="submit">Get Weather</button>

</form>

{{#if weatherData}}

<h2>{{weatherData.name}}</h2>

<p>Temperature: {{weatherData.main.temp}} °C</p>

<p>Description: {{weatherData.weather[0].description}}</p>

{{else}}

<p>Please enter a city name to see weather information.</p>

{{/if}}

</body>

</html>

Use code [with caution.](https://d.docs.live.net/faq#coding)

content\_copy

**Explanation:**

* The app.js file sets up the Express server and configures Handlebars as the templating engine.
* The / route renders the index.hbs template and passes any available weather data as context.
* If a city parameter exists in the query string, the server fetches weather data from the API using the provided API key.
* The index.hbs template displays a form to enter a city name and conditionally shows weather information using Handlebars templating syntax.

**Running the App:**

1. Install an API key from a weather service provider.
2. Replace YOUR\_API\_KEY in app.js with your actual API key.
3. Run the server: node app.js
4. Open http://localhost:3000/ in your browser.
5. Enter a city name and submit the form to see the weather information.

Create world clock app.

Here's a world clock app built with Node.js and Express.js:

**1. Project Setup:**

* Create a project directory: mkdir world-clock-app && cd world-clock-app
* Initialize project: npm init -y

**2. Install Dependencies:**

* Install Express and Moment.js: npm install express moment

**3. Folder Structure:**

world-clock-app/

- views/ (folder for Handlebars templates)

- index.hbs

- public/ (folder for static assets like CSS/JS)

- app.js (server code)

- package.json

**4. app.js (Server Code):**

JavaScript

const express = require('express');

const handlebars = require('express-handlebars');

const moment = require('moment-timezone'); // for time zone handling

const app = express();

const port = 3000;

// Configure Handlebars

app.engine('hbs', handlebars({ extname: '.hbs' }));

app.set('view engine', 'hbs');

// Serve static files from public directory

app.use(express.static('public'));

// Route to render the world clock page

app.get('/', (req, res) => {

const cities = [

{ name: 'London', offset: 0 },

{ name: 'New York', offset: -5 },

{ name: 'Tokyo', offset: 9 },

// Add more city objects with names and offsets

];

const currentTime = moment().format('YYYY-MM-DD HH:mm:ss');

const cityTimes = cities.map(city => ({

name: city.name,

time: moment.utc().add(city.offset, 'hours').tz(city.name).format('h:mm:ss A')

}));

res.render('index', { currentTime, cityTimes });

});

app.listen(port, () => {

console.log(`Server listening on port ${port}`);

});

Use code [with caution.](https://d.docs.live.net/faq#coding)

content\_copy

**5. views/index.hbs (Handlebars Template):**

HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>World Clock App</title>

</head>

<body>

<h1>World Clock</h1>

<p>Current UTC Time: {{currentTime}}</p>

<ul>

{{#each cityTimes}}

<li>{{name}}: {{time}}</li>

{{/each}}

</ul>

</body>

</html>

Use code [with caution.](https://d.docs.live.net/faq#coding)

content\_copy

**Explanation:**

* This code defines an array of city objects in the server-side JavaScript (app.js).
* The / route gets the current UTC time and iterates through the cities array.
* For each city, it uses Moment.js with time zone support to calculate the local time based on the offset.
* The route renders the index.hbs template and passes the current UTC time and an array of city objects with their local times as context.
* The index.hbs template displays the current UTC time and iterates through the cityTimes data to display each city name and its local time.

**Running the App:**

1. Run the server: node app.js
2. Open http://localhost:3000/ in your browser.
3. You'll see the current UTC time and the local times for the defined cities.