**a. Create a Git repository, add and commit a simple web application (5 pages)**

1. **Initialize a Git repository locally:**

bash

CopyEdit

mkdir simple-web-app

cd simple-web-app

git init

1. **Create 5 simple HTML pages (example):**

bash

CopyEdit

touch index.html about.html contact.html services.html portfolio.html

1. **Add content to the files (example for index.html):**

html

CopyEdit

<!-- index.html -->

<html>

<head><title>Home</title></head>

<body><h1>Welcome to My Site</h1></body>

</html>

1. **Add and commit files to Git:**

bash

CopyEdit

git add .

git commit -m "Initial commit: Added 5-page web application"

**b. Create a remote repo and push local repo to GitHub**

1. **Go to** [**GitHub**](https://github.com) **and create a new repository (e.g., simple-web-app)**
2. **Link the local repo to remote:**

bash

CopyEdit

git remote add origin https://github.com/your-username/simple-web-app.git

git branch -M main

git push -u origin main

1. **Explore push, pull, fetch:**
   * **Push:** Send local changes to GitHub:

bash

CopyEdit

git push

* + **Pull:** Pull latest changes from GitHub:

bash

CopyEdit

git pull

* + **Fetch:** Fetch changes without merging:

bash

CopyEdit

git fetch

git log origin/main # View commits from remote

**c. Clone a repository, modify and push back**

1. **Clone repository to new directory:**

bash

CopyEdit

git clone https://github.com/your-username/simple-web-app.git cloned-web-app

cd cloned-web-app

1. **Make changes (e.g., update about.html):**

bash

CopyEdit

echo "<p>Updated About Page</p>" >> about.html

git add about.html

git commit -m "Updated About Page"

git push

**d. Create branches, distribute work, and merge**

1. **Create and switch to new branches:**

bash

CopyEdit

git checkout -b feature-header

# Modify header in index.html

git add index.html

git commit -m "Added new header section"

git push -u origin feature-header

1. **Repeat for other features if needed. Then merge:**

bash

CopyEdit

git checkout main

git pull

git merge feature-header

git push

**e. Publish using GitHub Pages**

1. **Go to repository settings on GitHub.**
2. Scroll to **Pages** section.
3. Set source as main branch and root directory.
4. Save it — GitHub will provide a link like:  
   https://your-username.github.io/simple-web-app/

////////////////////////////////////////////////////////////////////////////////////////////////////

**a. Working with Prototypal Inheritance and Classes**

**Prototypal Inheritance Example:**

javascript

CopyEdit

const animal = {

speak() {

console.log(`${this.name} makes a noise.`);

}

};

const dog = Object.create(animal);

dog.name = 'Buddy';

dog.speak(); // Buddy makes a noise.

**Using Classes (ES6):**

javascript

CopyEdit

class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(`${this.name} makes a noise.`);

}

}

class Dog extends Animal {

speak() {

console.log(`${this.name} barks.`);

}

}

const dog = new Dog('Rex');

dog.speak(); // Rex barks.

**b. Working with Object and Array Destructuring**

**Object Destructuring:**

javascript

CopyEdit

const user = { name: 'Alice', age: 25, city: 'Paris' };

const { name, age } = user;

console.log(name, age); // Alice 25

**Array Destructuring:**

javascript

CopyEdit

const colors = ['red', 'green', 'blue'];

const [first, , third] = colors;

console.log(first, third); // red blue

**c. Working with Modules**

**Exporting from a module (math.js):**

javascript

CopyEdit

export function add(a, b) {

return a + b;

}

export const PI = 3.14;

**Importing in another file:**

javascript

CopyEdit

import { add, PI } from './math.js';

console.log(add(5, 2)); // 7

console.log(PI); // 3.14

Use type="module" in your HTML script tag when using modules.

**d. Working with Function Generators and Symbols**

**Function Generator:**

javascript

CopyEdit

function\* counter() {

let i = 0;

while (true) {

yield i++;

}

}

const gen = counter();

console.log(gen.next().value); // 0

console.log(gen.next().value); // 1

**Symbols:**

javascript

CopyEdit

// Create unique symbols

const ID = Symbol('id');

const ROLE = Symbol('role');

// Create a user object with symbol properties

const user = {

name: 'Alice',

age: 28,

[ID]: 101, // Symbol property

[ROLE]: 'admin' // Another symbol property

};

// Accessing symbol properties

console.log(user.name); // Alice

console.log(user[ID]); // 101

console.log(user[ROLE]); // admin

// Loop through properties (symbols are hidden)

for (let key in user) {

console.log(key); // name, age (not symbols)

}

// Get all symbols on the object

const symbols = Object.getOwnPropertySymbols(user);

console.log(symbols); // [ Symbol(id), Symbol(role) ]

// Access symbol values dynamically

symbols.forEach(sym => {

console.log(`${sym.toString()} = ${user[sym]}`);

});

**e. Working with Closure**

**Closure Example:**

javascript

CopyEdit

function outer() {

let count = 0;

return function inner() {

count++;

console.log(`Count is ${count}`);

};

}

const counter = outer();

counter(); // Count is 1

counter(); // Count is 2

////////////////////////////////////////////////////////////////////////////////////////////////////////////

**a. Working with Higher Order Function in JavaScript**

javascript

CopyEdit

// Higher-order function: accepts a function as argument

function greet(name) {

return `Hello, ${name}!`;

}

function processUserInput(callback) {

const name = "Alice";

console.log(callback(name)); // Hello, Alice!

}

processUserInput(greet);

**b. Using Callback and Creating a Callback Hell Situation**

javascript

CopyEdit

// Simulated async steps using nested callbacks (callback hell)

function step1(callback) {

setTimeout(() => {

console.log("Step 1 complete");

callback();

}, 1000);

}

function step2(callback) {

setTimeout(() => {

console.log("Step 2 complete");

callback();

}, 1000);

}

function step3(callback) {

setTimeout(() => {

console.log("Step 3 complete");

callback();

}, 1000);

}

// Callback hell

step1(() => {

step2(() => {

step3(() => {

console.log("All steps completed (callback hell)");

});

});

});

**c. Working with XHR: Response**

javascript

CopyEdit

// Make an HTTP GET request using XMLHttpRequest

function fetchDataWithXHR() {

const xhr = new XMLHttpRequest();

xhr.open("GET", "https://jsonplaceholder.typicode.com/posts/1", true);

xhr.onload = function () {

if (xhr.status === 200) {

const data = JSON.parse(xhr.responseText);

console.log("XHR Response:", data);

} else {

console.error("XHR Error:", xhr.status);

}

};

xhr.onerror = function () {

console.error("Request failed.");

};

xhr.send();

}

fetchDataWithXHR();

**d. Dealing with Callback Hell Using Promises**

javascript

CopyEdit

// Rewriting the earlier steps using Promises

function step1() {

return new Promise(resolve => {

setTimeout(() => {

console.log("Step 1 complete");

resolve();

}, 1000);

});

}

function step2() {

return new Promise(resolve => {

setTimeout(() => {

console.log("Step 2 complete");

resolve();

}, 1000);

});

}

function step3() {

return new Promise(resolve => {

setTimeout(() => {

console.log("Step 3 complete");

resolve();

}, 1000);

});

}

step1()

.then(step2)

.then(step3)

.then(() => console.log("All steps complete (Promise)"));

**e. Dealing with Promise Chaining and Async / Await**

javascript

CopyEdit

// Same example using async/await

async function runSteps() {

await step1();

await step2();

await step3();

console.log("All steps complete (Async/Await)");

}

runSteps();

// Note: step1, step2, and step3 from the previous section are reused here.

////////////////////////////////////////////////////////////////////////////////////////////////////////

**a. Fetch API data and display in a table**

html

CopyEdit

<!-- index.html -->

<!DOCTYPE html>

<html>

<head>

<title>Fetch and Display Table</title>

</head>

<body>

<h2>User Data</h2>

<table border="1" id="userTable">

<tr>

<th>Name</th>

<th>Email</th>

<th>Username</th>

</tr>

</table>

<script>

fetch('https://jsonplaceholder.typicode.com/users')

.then(res => res.json())

.then(data => {

const table = document.getElementById("userTable");

data.forEach(user => {

const row = table.insertRow();

row.innerHTML = `<td>${user.name}</td><td>${user.email}</td><td>${user.username}</td>`;

});

});

</script>

</body>

</html>

**✅ b. Weather API – display city, min-temp, max-temp, humidity**

html

CopyEdit

<!-- weather.html -->

<!DOCTYPE html>

<html>

<head>

<title>Weather Info</title>

</head>

<body>

<h2>Weather Information</h2>

<input type="text" id="city" placeholder="Enter city" />

<button onclick="getWeather()">Get Weather</button>

<div id="weatherResult"></div>

<script>

async function getWeather() {

const city = document.getElementById("city").value;

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

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

const res = await fetch(url);

const data = await res.json();

const html = `

<p><strong>City:</strong> ${data.name}</p>

<p><strong>Min Temp:</strong> ${data.main.temp\_min} °C</p>

<p><strong>Max Temp:</strong> ${data.main.temp\_max} °C</p>

<p><strong>Humidity:</strong> ${data.main.humidity}%</p>

`;

document.getElementById("weatherResult").innerHTML = html;

}

</script>

</body>

</html>

**✅ c. Weather Forecast – date and temperature table**

html

CopyEdit

<!-- forecast.html -->

<!DOCTYPE html>

<html>

<head>

<title>Weather Forecast</title>

</head>

<body>

<h2>5-Day Forecast</h2>

<input type="text" id="city" placeholder="Enter city" />

<button onclick="getForecast()">Get Forecast</button>

<table border="1" id="forecastTable">

<tr><th>Date</th><th>Temperature (°C)</th></tr>

</table>

<script>

async function getForecast() {

const city = document.getElementById("city").value;

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

const url = `https://api.openweathermap.org/data/2.5/forecast?q=${city}&units=metric&appid=${apiKey}`;

const res = await fetch(url);

const data = await res.json();

const table = document.getElementById("forecastTable");

table.innerHTML = "<tr><th>Date</th><th>Temperature (°C)</th></tr>"; // Reset table

// Show forecast every 8 steps (~once per day)

data.list.filter((\_, i) => i % 8 === 0).forEach(item => {

const date = new Date(item.dt\_txt).toLocaleDateString();

const temp = item.main.temp;

const row = table.insertRow();

row.innerHTML = `<td>${date}</td><td>${temp}</td>`;

});

}

</script>

</body>

</html>

**✅ d. Plot Bar Chart with Chart.js (Date vs Temperature)**

html

CopyEdit

<!-- chart.html -->

<!DOCTYPE html>

<html>

<head>

<title>Weather Chart</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<h2>Weather Bar Chart</h2>

<input type="text" id="city" placeholder="Enter city" />

<button onclick="drawChart()">Draw Chart</button>

<canvas id="weatherChart" width="400" height="200"></canvas>

<script>

async function drawChart() {

const city = document.getElementById("city").value;

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

const url = `https://api.openweathermap.org/data/2.5/forecast?q=${city}&units=metric&appid=${apiKey}`;

const res = await fetch(url);

const data = await res.json();

const dates = [];

const temps = [];

data.list.filter((\_, i) => i % 8 === 0).forEach(item => {

dates.push(new Date(item.dt\_txt).toLocaleDateString());

temps.push(item.main.temp);

});

const ctx = document.getElementById('weatherChart').getContext('2d');

new Chart(ctx, {

type: 'bar',

data: {

labels: dates,

datasets: [{

label: 'Temp (°C)',

data: temps,

backgroundColor: 'rgba(54, 162, 235, 0.6)'

}]

},

options: {

responsive: true,

scales: {

y: {

beginAtZero: true

}

}

}

});

}

</script>

</body>

</html>

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**a. Create Custom / Local Modules Using Various Patterns**

**1. CommonJS (Node-style modules)**

**📄 math.js**

javascript

CopyEdit

// math.js

function add(a, b) {

return a + b;

}

function subtract(a, b) {

return a - b;

}

module.exports = { add, subtract };

**📄 app.js**

javascript

CopyEdit

const math = require('./math');

console.log(math.add(5, 3)); // 8

console.log(math.subtract(5, 3)); // 2

**2. ES6 Modules (requires "type": "module" in package.json)**

**📄 math.mjs**

javascript

CopyEdit

export function add(a, b) {

return a + b;

}

export function multiply(a, b) {

return a \* b;

}

**📄 app.mjs**

javascript

CopyEdit

import { add, multiply } from './math.mjs';

console.log(add(2, 3));

console.log(multiply(4, 5));

**✅ b. Explore os, path, util, and events modules**

// OS Module

const os = require('os');

console.log('Platform:', os.platform());

console.log('OS Type:', os.type());

console.log('CPU Architecture:', os.arch());

console.log('CPU Info:', os.cpus());

console.log('Total Memory:', os.totalmem());

console.log('Free Memory:', os.freemem());

console.log('Hostname:', os.hostname());

console.log('Uptime (seconds):', os.uptime());

// Path Module

const path = require('path');

const filePath = '/users/admin/docs/file.txt';

console.log(path.basename(filePath)); // Output: file.txt

console.log(path.dirname(filePath)); // Output: /users/admin/docs

console.log(path.extname(filePath)); // Output: .txt

console.log(path.join('folder1', 'folder2', 'file.txt')); // Output: folder1/folder2/file.txt

// Util Module

const util = require('util');

const name = 'Alice';

const age = 25;

console.log(util.format('Name: %s, Age: %d', name, age));

**// Output: Name: Alice, Age: 25**

const fs = require('fs');

const util = require('util');

const readFile = util.promisify(fs.readFile);

async function read() {

try {

const data = await readFile('example.txt', 'utf8');

console.log(data);

} catch (err) {

console.error('Error reading file:', err);

}

}

**read();**

const util = require('util');

const obj = {

name: 'Node.js',

features: ['events', 'streams', 'http'],

nested: { level: 1, deeper: { level: 2 } }

};

console.log(util.inspect(obj, { depth: null }));

**// Events Module**

const EventEmitter = require('events');

// Create event emitter instance

const emitter = new EventEmitter();

// Define an event handler

emitter.on('greet', (name) => {

console.log(`Hello, ${name}!`);

});

// Emit the event

emitter.emit('greet', 'Alice');

**✅ c. Use fs module to Create Directories and Files**

javascript

CopyEdit

// filesystem.js

const fs = require('fs');

const path = require('path');

const dir = './myFolder';

// Create directory

if (!fs.existsSync(dir)) {

fs.mkdirSync(dir);

console.log("Directory created.");

}

// Create text file

fs.writeFileSync(path.join(dir, 'hello.txt'), 'Hello, world!');

console.log("Text file created.");

// Create JSON file

const data = { name: "Alice", age: 30 };

fs.writeFileSync(path.join(dir, 'data.json'), JSON.stringify(data, null, 2));

console.log("JSON file created.");

**✅ d. Read and Write Streaming Data**

const fs = require('fs');

// Create readable stream from input.txt

const readStream = fs.createReadStream('input.txt', 'utf8');

// Create writable stream to output.txt

const writeStream = fs.createWriteStream('output.txt');

// Pipe the read stream directly into the write stream

readStream.pipe(writeStream);

// Optional: handle events

readStream.on('data', (chunk) => {

console.log('Reading chunk:', chunk.length);

});

readStream.on('end', () => {

console.log('Finished reading');

});

writeStream.on('finish', () => {

console.log('Finished writing');

}); /////////////////////////////////////////////////////////////////////////////////////////////////////////

**a. HTTP Server (Core Node.js) — Serve HTML, TEXT, JSON**

javascript

CopyEdit

// httpServer.js

const http = require('http');

const fs = require('fs');

const path = require('path');

const server = http.createServer((req, res) => {

console.log(`Request: ${req.url}`);

if (req.url === '/' || req.url === '/home') {

res.writeHead(200, { 'Content-Type': 'text/html' });

res.end('<h1>Welcome to Home Page</h1>');

}

else if (req.url === '/about') {

res.writeHead(200, { 'Content-Type': 'text/plain' });

res.end('This is a plain text About Page.');

}

else if (req.url === '/api/user') {

res.writeHead(200, { 'Content-Type': 'application/json' });

const user = { name: "Alice", age: 25 };

res.end(JSON.stringify(user));

}

else {

res.writeHead(404, { 'Content-Type': 'text/html' });

res.end('<h1>404 - Page Not Found</h1>');

}

});

server.listen(3000, () => {

console.log('HTTP Server running at http://localhost:3000');

});

Run with:

bash

CopyEdit

node httpServer.js

**✅ b. Express Server with Endpoints on Port 3000**

javascript

CopyEdit

// expressServer.js

const express = require('express');

const app = express();

// Text route

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

res.send('<h1>Welcome to Express Home</h1>');

});

// Plain text

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

res.type('text/plain');

res.send('This is an Express server.');

});

// JSON route

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

res.json({ name: "Bob", city: "New York" });

});

// HTML file response (optional)

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

res.sendFile(\_\_dirname + '/index.html');

});

app.listen(3000, () => {

console.log('Express server running at http://localhost:3000');

});

/////////////////////////////////////////////////////////////////////////////////////////////////////