Group A-(WAD)
Assignment 1 **a.** Create a responsive web page which shows the ecommerce/college/exam admin dashboard with sidebar and statistics in cards using HTML, CSS and Bootstrap. **b.** Write a JavaScript Program to get the user registration data and push to array/local storage with AJAX POST method and data list in new page.

Problem Statement:

Create a user registration form using HTML and JavaScript. When the user submits the form:

1. The data should be sent using the **AJAX POST method**.
2. The data should be stored in either an **array** or in **localStorage**.
3. A new page should display the list of all registered users from localStorage.

"JK ek simple website bana raha hai jahan wo apna naam, mobile number, aur email bhar ke 'Submit' karta hai. Computer uska data save kar leta hai. Fir jab JK 'Display' karta hai, toh wohi data dobara dikhai deta hai form mein."

🧠 AJAX Kya Hai?

AJAX ka full form hai **Asynchronous JavaScript and XML**.
Ye ek technique hai jo web pages ko bina reload kiye server se data exchange karne ki suvidha deti hai.

AJAX allow karta hai browser ko background me server se baat karne ke liye, bina pura page reload kiye.

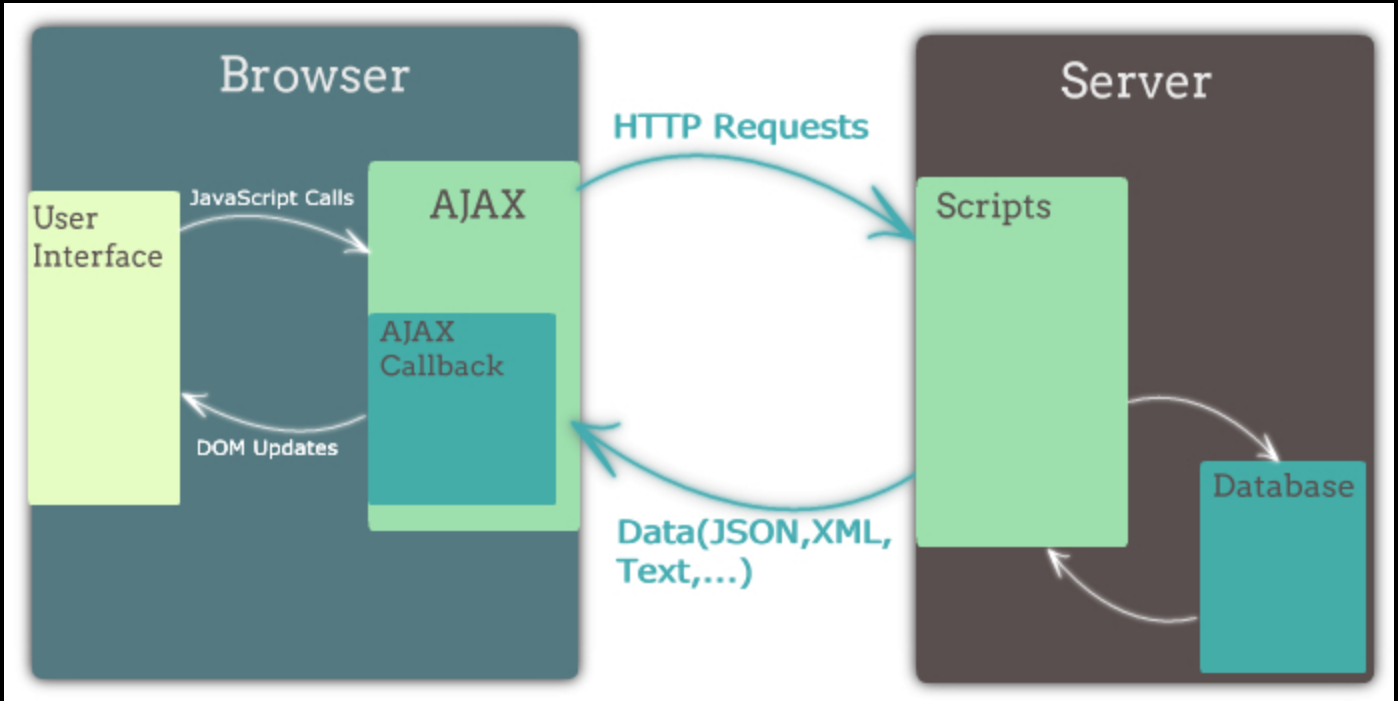
✅ AJAX Se Kya Hota Hai?

- Background me server se data aata hai.
- Page ke sirf ek part ko update kar sakte ho.
- Fast & Smooth experience milta hai — bilkul app jaise.

❌ Agar AJAX Na Hota To Kya Hota?

- Jab bhi user kuch data chahta (jaise ek user profile ya search result), **pura webpage reload hota**.
- Har action pe browser server ko request bhejta, aur **naya HTML page load hota**.
- Ye slow experience hota, aur user ko baar-baar wait karna padta.

Feature	Without AJAX	With AJAX
Page reload	✓ Yes	❌ No
Server interaction	Full HTML page	Only data file (message.txt)
Speed	Slower	Faster
User experience	Breaks flow	Smooth
Code complexity	Simpler form-based	JavaScript needed


Browser
User Interface
JavaScript Calls
AJAX
AJAX Callback
DOM Updates
HTTP Requests
Data(JSON,XML, Text,...)
Server
Scripts
Database

1. User Interface (UI):

•Jab user button click karta hai ya koi form submit karta hai,

•Tab JavaScript se ek function call hota hai.

2. AJAX Sends Request:

•JavaScript AJAX ka use karke server ko **HTTP Request** bhejta hai.

•Ye request asynchronous hoti hai – **page reload nahi hota**.

3. Server Receives Request:

•Server par koi **script (jaise PHP, Python,node etc.)** is request ko process karti hai.

•Script database ke saath interact karke required data fetch ya update karti hai.

4. Database Interaction:

•Server script database se data read ya write karti hai.

5. Server Sends Response:

•Server processed data ko **JSON, XML, ya plain text** format me wapas bhejta hai.

6. AJAX Callback:

•AJAX response ko receive karta hai aur ek callback function run hota hai.

7. DOM Update:

•JavaScript se web page ke content (DOM) ko update kiya jata hai bina page reload kiye.

•User ko turant naya data dikhai deta hai.

Maan lo tum ek form fill kar rahe ho aur ek button dabate ho "Get Data".

- Agar ye **Synchronous** hota:
  - Poora browser **ruk jata** jab tak server se data wapas nahi aata.
  - UI freeze ho jata.
- Agar ye **Asynchronous** hota (AJAX ki tarah):
  - Browser background me server se data la raha hota.
  - Tab tak tum kuch aur kar sakte ho – page ka UI chalta rehta hai.
  - Jab data aa jata hai, to JavaScript usse page pe dikha deti hai.

Callback Function kya hota hai?

Callback function ek **function hota hai jo kisi doosre function ke andar pass kiya jaata hai**, aur **baad me call (execute) kiya jaata hai** jab koi kaam complete ho jaye.

📌 **Simple Definition:**
"Callback function wo hota hai jo tab chalega jab koi kaam complete ho jaaye."
Ye aksar asynchronous kaam me use hota hai — jaise AJAX call ke baad server se response milne par.

Maan lo tumne dosto ko bola:

"Main khana bana raha hoon, khatam hone par main tumhe bulaunga.“

Yahan **"tumhe bulaunga"** ek **callback** hai — jo tab chalega jab pehla kaam (khana banana) complete ho jaye.

📊 AJAX Flowchart Diagram

[1] User Fills Form (Name, Email, Mobile)

↓

[2] User Clicks "Submit" Button (btn-2)

↓

[3] jQuery Function Captures Form Data

↓

[4] Data is Converted into JSON and Stored in LocalStorage

↓

[5] Using jQuery .load(), jk2.html Page is Loaded into

↓

[6] On jk2.html, User Clicks "Display" Button (btn-1)

↓

[7] JavaScript Fetches Data from LocalStorage and Parses JSON

↓

[8] Form Fields Auto-Fill with Stored Name, Email, and Mobile

```html
<!-- Yeh batata hai ki page HTML5 mein likha gaya hai -->
<!DOCTYPE html>
<html lang="en">

<head>
    <!-- Yeh batata hai ki page ka character encoding kya hai (har language support kare) -->
    <meta charset="UTF-8">

    <!-- Yeh mobile ya small screen mein page ka size set karta hai -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- Page ka title tab mein dikhega -->
    <title>Ajax1</title>

    <!-- jQuery CDN (internet se jQuery library use kar rahe) -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
```

🧠 Pehle Samjho — CDN Kya Hota Hai?

📦 **CDN (Content Delivery Network)** ek internet ka shortcut hota hai — jisse tu kisi library ko **download kiye bina** direct use kar sakta hai apne HTML page mein.

✅ Steps to Get a CDN Link (Example: jQuery)

🔹 **Step 1: Google Pe Search Kar**

Type kare:

jquery cdn

Ya direct jao:

👉 https://cdnjs.com/libraries/jquery

- **Step 2: Latest Version Choose Karo**

Wahan tumhe version dikhenge. Latest version hamesha upar hota hai.
Example: 3.7.1

- **Step 3: Copy the Link**

Wahan tumhe ye link milega:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
```

Isko copy karo aur <head> section mein paste kar do.

jQuery kya hai?

jQuery ek fast, small, aur feature-rich JavaScript library hai. Ye HTML document traversal, event handling, animation, aur Ajax interactions ko bahut hi aasan bana deta hai. jQuery ka main goal hai JavaScript code ko simple aur browser-compatible banana.

Feature	Explanation	jQuery Used
DOM Ready	Page fully load hone ke baad code run karna	$(document).ready(...)
Event Handling	Button click pe action lena	$("#btn").click(...)
Get Input Value	Text field se value nikaalna	$("#id").val()
Set Input Value	Field me value set karna	$("#id").val(value)
Load External HTML	Dusra HTML part dynamically load karna	$("#div").load("file.html")

```html
<!-- JavaScript part start -->
<script>
    // Yeh function tab chalta hai jab poora page load ho chuka hota hai
    $(document).ready(function () {

        // Jab JK 'Submit' button dabata hai
        $("#btn-2").click(function () {

            // Yeh 3 input fields se value le raha hai (name, email, mobile)
            const obj = {
                name: $("#name").val(),      // JK ka naam
                email: $("#email").val(),    // JK ka email id
                mobile: $("#mobile").val()   // JK ka mobile number
            };

            // Object ko JSON format mein convert kar rahe (store karne ke liye)
            const myjson = JSON.stringify(obj);

            // JSON data ko browser ke localStorage mein save karte hain
            localStorage.setItem("data", myjson);

            // "jk2.html" file ko current page ke div1 mein load kar rahe
            $("#div1").load("http://127.0.0.1:5500/jk2.html");
        });
    });
</script>
</head>
```

```html
<body>
    <!-- Page heading -->
    <h1>JK - Student Details Form</h1>

    <!-- Yeh form ka container hai -->
    <div id="div1">

        <!-- Name field -->
        <label for="name">Enter name:</label><br>
        <input type="text" id="name" name="name"><br><br>

        <!-- Mobile number field -->
        <label for="mobile">Enter mobile number:</label><br>
        <input type="text" id="mobile" name="mobile"><br><br>

        <!-- Email field -->
        <label for="email">Enter email id:</label><br>
        <input type="text" id="email" name="email"><br><br>

        <!-- Button dabane pe data save hoga aur doosra page load hoga -->
        <button id="btn-2">Submit</button>

    </div>/#div1
</body>
```

JK - Student Details Form

Enter name:

Enter mobile number:

Enter email id:

Submit

```html
<!-- Yeh HTML page hai jo saved data dikhata hai -->
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ajax2</title>

    <!-- jQuery CDN load kar rahe -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
```

```
    <script>
        // Page load hone ke baad
        $(document).ready(function () {

            // Jab JK 'Display' button dabata hai
            $("#btn-1").click(function () {

                // Pehle se saved data uthao localStorage se
                const var1 = localStorage.getItem("data");

                // JSON ko normal JavaScript object mein convert karo
                const myobj = JSON.parse(var1);

                // Ab input fields mein JK ka saved data bhar do
                $("#name").val(myobj.name);
                $("#email").val(myobj.email);
                $("#mobile").val(myobj.mobile);
            });
        });
    </script>
</head>
```

```html
<body>
    <!-- Heading -->
    <h1>JK - Display Your Details</h1>

    <!-- Form jisme data dikhaya jayega -->
    <div id="div1">

        <!-- Name field -->
        <label for="name">Enter name:</label><br>
        <input type="text" id="name" name="name"><br><br>

        <!-- Mobile field -->
        <label for="mobile">Enter mobile number:</label><br>
        <input type="text" id="mobile" name="mobile"><br><br>

        <!-- Email field -->
        <label for="email">Enter email id:</label><br>
        <input type="text" id="email" name="email"><br><br>

        <!-- Button jo JK ka saved data dikhata hai -->
        <button id="btn-1">Display</button>

    </div>/#div1
</body>
```

JK - Display Your Details

Enter name:

Jayesh Gorakh Kande

Enter mobile number:

08766833516

Enter email id:

Abcde@gmail.com

Display

Assignment 2

a. Create version control account on GitHub and using Git commands to create repository and push your code to GitHub.

b. Create Docker Container Environment (NVIDEIA Docker or any other).

c. Create an Angular application which will do following actions: Register User, Login User, Show User Data on Profile Component

🔰 1. Angular Kya Hai?

Angular ek **frontend** **framework** hai jo **TypeScript** pe bana hai. Yeh Google ne banaya hai, aur single-page applications (SPA) banane ke liye use hota hai.

Term	Explanation
Library	Code ka collection hota hai jo specific kaam karta hai. Ex: Lodash, Axios
Framework	Full structure provide karta hai, jisme aap uske rules follow karte ho. Ex: Angular, Django
Module	Code ka chhota part jo export/import hota hai. Ex: Angular modules

Term	Sochne Ka Tarika	Kaam
Library	Tools ka set (screwdriver)	Specific task (sorting, formatting, etc.)
Framework	Ghar ka blueprint (strict rules)	App ka pura structure provide karta hai
Module	Code ka dabba (chhota unit)	Code ko organize aur reuse karne ke liye banate hain

🧠 TypeScript Kya Hai?

➡️ Definition:

TypeScript ek **JavaScript ka upgraded version** hai jisme *type safety* hoti hai (jaise number, string, boolean declare karna). Yeh JavaScript ka superset hai — iska matlab: **har JavaScript code valid TypeScript hota hai, but har TypeScript code valid JS nahi hota**.

🔧 Kyun Use Karte Hain?

- Error ko code likhte waqt pakadne ke liye
- Code ko zyada readable aur manageable banane ke liye
- Angular jaise framework mein strictly required hai

✨ TypeScript ke Basics (Angular ke Context mein)

```
// Variable Declaration with Types
let name: string = 'Amit';          // "name" sirf string hi le sakta hai    Cann
let age: number = 25;               // "age" sirf number hi le sakta hai
let isLoggedIn: boolean = true;     // "isLoggedIn" sirf true/false le sakta hai
```

```
// Variable Declaration
let name = 'Amit';                  // Type mention nahi, kuch bhi assign kar sakte ho
let age = 25;
let isLoggedIn = true;
```

```
// Function with parameter and return type
function add(a: number, b: number): number {    Duplicate function implementation.
  return a + b; // Dono input number hone chahiye, return bhi number hoga
}
```

```
// Function without type check
function add(a, b) {
  return a + b; // Agar a="2" aur b=3, to "23" ho jayega (galti ho sakti hai)
}
```

```
// Object with structure
let user: { name: string; age: number } = {    Cannot rede
  name: 'Amit',
  age: 25
}; // Sirf "name" (string) aur "age" (number) allowed hai
```

```
// Object without fixed structure
let user = {
  name: 'Amit',
  age: 25
}; // Koi bhi extra ya galat key daal sakte ho
```

```
// Array of strings
let skills: string[] = ['HTML', 'CSS', 'Angular']; // Sirf string values allowed
```

```
// Array with any type values
let skills = ['HTML', 'CSS', 'Angular']; // Aap number ya object bhi daal sakte ho
```

```typescript
// Interface: Ek rule banata hai object ke structure ke liye
interface User {
  name: string;
  email: string;
}

let u1: User = {
  name: 'Amit',
  email: 'amit@example.com'
}; // User interface ke rules follow karega
```

```
// Interface ka concept nahi hota
JavaScript me
```

```typescript
// Class with Constructor and Method
class Person1 {
  constructor(public name: string, public age: number) {
    // public likhne se automatic class member ban jata hai
  }

  greet(): string {
    return `Hi, I'm ${this.name}`; // "this.name" se class ka data access
  }
}
```

```typescript
const p11 = new Person1('Amit', 30);  // New object banaya
console.log(p1.greet());              // Output: Hi, I'm Amit
```

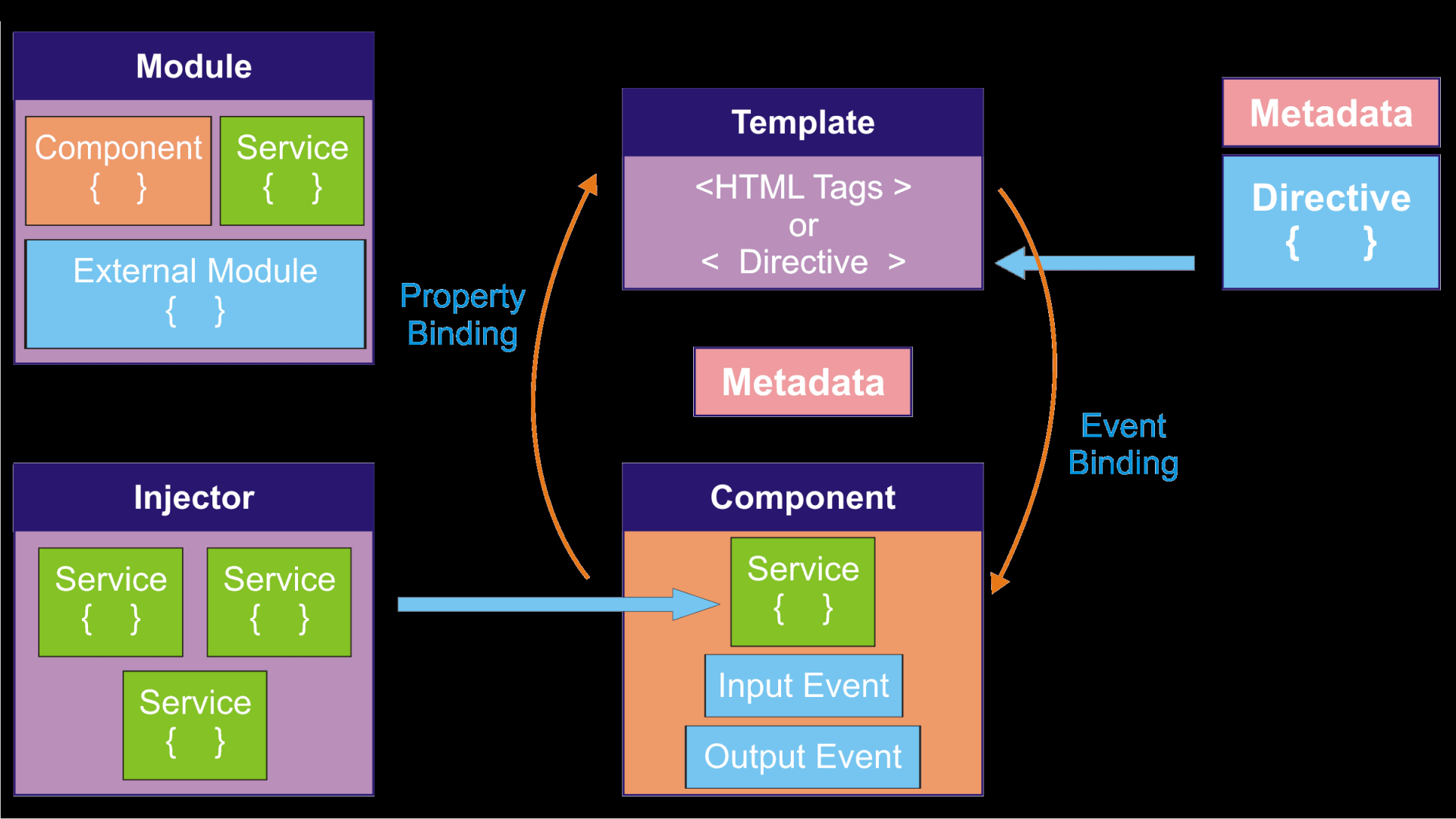
```
// Class without type
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }

  greet() {
    return `Hi, I'm ${this.name}`;
  }
}

const p1 = new Person('Amit', 30);
console.log(p1.greet());
```

```
// Type Inference
let message1 = 'Hello';                // TypeScript samajhta hai yeh string hai
// message = 123;  Error aayega, kyunki message ko pehle string define kiya
```

```
// No Type Inference
let message = 'Hello';
message = 123; // Allowed but galti ka chance zyada
```


Module
Component
{ }
Service
{ }
External Module
{ }
Injector
Service
{ }
Service
{ }
Service
{ }
Property
Binding
Template
<HTML Tags >
or
< Directive >
Metadata
Component
Service
{ }
Input Event
Output Event
Event
Binding
Metadata
Directive
{ }

1. Module

•Socho **Module** ek box jaisa hai jisme hum related chizien rakhte hain — jaise:

- •**Component** (UI banata hai)
- •**Service** (data ya logic handle karta hai)
- •**Value** (jaise number 3.14)
- •**Function** (jaise koi kaam karne wala code)

2. Template < >

•Ye HTML part hota hai jise user screen pe dekhta hai.

•Isme hum likhte hain kya show karna hai.

3. Metadata

•Ye ek tarah ka setting hota hai jo Angular ko batata hai:

- Ye component hai ya service ya directive
- Iska template kya hai
- Kya-kya properties use kar raha hai

4. Directive { }

•Ye HTML element ko special behavior deta hai.

•Jaise agar hum chahte hain ki ek box mouse ke upar aaye to color change ho — ye directive se hota hai.

5. Component { }

•Ye pura ek part hota hai app ka (jaise ek card, form, ya button).

•Isme HTML (template), CSS (design), aur TypeScript (logic) hota hai.

6. Injector / Service

•**Service** ek helper jaisa hota hai — jo data fetch karta hai, ya kuch logic handle karta hai.

•**Injector** ensure karta hai ki component ko sahi service mil jaye

7. Binding

•**Property Binding**: Component se Template tak data bhejna
(Example: user name show karna)

•**Event Binding**: Template se Component tak signal bhejna
(Example: Button click hone par kaam karna)

Easy Example:
Socho ek "Login Form" banaya:
•**Component**: Login form ka logic
•**Template**: Form ka HTML design
•**Service**: Server se login check karne ka kaam
•**Binding**: Username input lena (property), Login button dabana (event)

Feature	**Property Binding**	**Event Binding**
🔹 **Syntax**	[property]="expression"	(event)="handlerFunction()"
🔁 **Data Direction**	Component → DOM (Template)	DOM (Template) → Component
🎯 **Purpose**	DOM element ki property set karna	User ke event pe response dena
🧠 **Example Meaning**	"Component se value lo aur template me dikhayo"	"Template me koi kaam ho to component ko batao"
📦 **Example Use Case**	Image URL set karna, disabled button, input value	Button click, input change, mouse events
🔧 **Code Example**	<img [src]="imageUrl">	<button (click)="showAlert()">Click</button>
🧪 **Component Code**	imageUrl = 'https://abc.png';	showAlert() { alert('Clicked!'); }
🔄 **Binding Direction Type**	One-Way (from component to HTML)	One-Way (from HTML to component)
🎛️ **Affect On**	Element property or attribute	Event listener setup
🧩 **Used With**	Any DOM property (like src, value, disabled)	Any DOM event (like click, input, keyup)

Symbol	Binding Type	Meaning
[]	Property Binding	Set property from component
()	Event Binding	Call method on user action
[()]	Two-way Binding	Data sync both ways (ngModel)

Jab Angular ek component create karta hai, update karta hai ya destroy karta hai, to kuch predefined **functions (hooks)** call karta hai — inhe **Lifecycle Hooks** kehte

Lifecycle Hook	Kab Call Hota Hai?	Use Case Example
ngOnChanges()	@Input property change hone par	Data update from parent
ngOnInit()	Component initialize hone par	API calls, variable set
ngDoCheck()	Har change detection cycle mein	Custom checks
ngAfterContentInit()	Content projection ke baad	ke andar ka kaam
ngAfterContentChecked()	Projected content check hone ke baad	Validate projected data
ngAfterViewInit()	Component view fully load hone ke baad	DOM element access
ngAfterViewChecked()	View check hone ke baad	UI tweaks
ngOnDestroy()	Component destroy hone se pehle	Unsubscribe observables, cleanup timers

Hook Name	Kab Chalta Hai?
ngOnChanges	Parent data change hone pe
ngOnInit	Component pehli baar load hone pe
ngDoCheck	Har custom/manual change detect hone pe
ngAfterViewInit	Component ka HTML render hone ke baad
ngAfterViewChecked	Har view check ke baad
ngAfterContentInit	Jab parent dynamic content bheje (<ng-content>)
ngOnDestroy	Jab component screen se hataya ja raha ho

🧩 **Install:**

1.Node.js → https://nodejs.org/ (LTS version)

2.Angular CLI:

```
npm install -g @angular/cli
```

Create a new Angular project:

```
ng new project-name --no-standalone
```

Add flags during setup:

- Enable routing → ✓
- Include CSS (default styling) → ✓

app.component.html M

project-name > src > app > app.component.html > a

```html
<h1>"Welcome to JK Coding Pathshala – Jahaan Coding Seekhna Hai Asaan, Mazedaar aur Career-Building!"</h1>

<p>First time on the website ? do registration</p>

<a routerLink="register">Register</a>
<router-outlet></router-outlet>
```

Serve (start) the Angular application:

cd project-name ng serve

Angular CLI Commands to Generate Components

ng generate component home --no-standalone
ng generate component registration --no-standalone
ng generate component login --no-standalone

---

ng g c home --no-standalone
ng g c registration --no-standalone
ng g c login --no-standalone
Breakdown:
ng g c → generate component ka short form

--no-standalone → traditional NgModule-based component banane ke liye

home.component.html U ×

project-name > src > app > home > home.component.html > div.home-container > div.playlist-section > div.playlist > div.playlist-item

```html
<div class="home-container">
  <div class="intro-section">
    <h1>Welcome to JK Coding Patshala</h1>
    <p>Your one-stop destination to learn <strong>C</strong>, <strong>DSA</strong>, and <strong>Java Full Stack</strong>.</p>
    <p>Here, I teach programming concepts in a simple and easy-to-understand manner. Whether you're a beginner or preparing for placements, you will find everything you need to succeed!</p>
  </div>/.intro-section

  <div class="playlist-section">
    <h2>Featured Playlists</h2>
    <div class="playlist">
      <div class="playlist-item">
        <h3>c programming</h3>
        <p>Get a deep dive into c programming, covering everything from basics to advanced concepts.</p>
        <a href="https://www.youtube.com/playlist?list=PLE4oKxtdRjry1v7po46DGD6KnLD4gL06b" target="_blank">View Playlist</a>
      </div>/.playlist-item
      <div class="playlist-item">
        <h3>100 Days DSA Question Series</h3>
        <p>Practice DSA through 100 days of problem-solving, with solutions and
```

TS home.component.ts U ×

project-name > src > app > home > TS home.component.ts > ...

```
1   import { Component } from '@angular/core';
2
3   @Component({
4     selector: 'app-home',
5     standalone: false,
6     templateUrl: './home.component.html',
7     styleUrl: './home.component.css'
8   })
9   export class HomeComponent {
10
11  }
12
```

login.component.html U ×

project-name > src > app > login > login.component.html > ...

```html


<p>Plzz Enter your details to login</p>
<form>
  <div>
    <label for="name">name</label>
    <input type="text" id="username" name="username" required />
  </div>
  <div>
    <label for="password">Password</label>
    <input type="password" id="password" name="password" required />
  </div>

</form>

<button type="button" (click)="onLogin()">Login</button>
<p>Already have an account? <a routerLink="home">Login</a>
<router-outlet></router-outlet></p>

```

login.component.ts U

project-name > src > app > login > login.component.ts > LoginComponent > onLogin

```
1  import { Component } from '@angular/core';
2  import { Router } from '@angular/router';
3  @Component({
4    selector: 'app-login',
5    standalone: false,
6    templateUrl: './login.component.html',
7    styleUrl: './login.component.css'
8  })
9  export class LoginComponent {
10
11   constructor(private router: Router) {}
12   onLogin() {
13     // Handle login logic (for now it's just navigation)
14     console.log('User logged in');
15     this.router.navigate(['/home']); // Navigate to home page
16   }
17 }
18
```

registration.component.html U ×

project-name > src > app > registration > registration.component.html > form > div > input#password

```html

<h2>Register</h2>
<form>
  <div>
    <label for="name">name</label>
    <input type="text" id="username" name="username" required />
  </div>
  <div>
    <label for="password">Password</label>
    <input type="password" id="password" name="password" required />
  </div>

</form>

<button type="button" (click)="onRegister()">Register</button>
<p>Already have an account? <a routerLink="login">Login</a>
<router-outlet></router-outlet></p>
```

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
@Component({
  selector: 'app-registration',
  standalone: false,
  templateUrl: './registration.component.html',
  styleUrl: './registration.component.css'
})
export class RegistrationComponent {

  constructor(private router: Router) {}

  onRegister() {
    // Handle registration logic (for now it's just navigation)
    console.log('User registered');
    this.router.navigate(['login']); // Navigate to login page
  }

}
```

app-routing.module.ts M

project-name > src > app > app-routing.module.ts > ...

```
1  import { NgModule } from '@angular/core';
2  import { RouterModule, Routes } from '@angular/router';
3
4  // Component imports – zaruri hain routing ke liye
5  import { RegistrationComponent } from './registration/registration.component';
6  import { LoginComponent } from './login/login.component';
7  import { HomeComponent } from './home/home.component';
8
9  const routes: Routes = [
10   // { path: '', redirectTo: '/register', pathMatch: 'full' },  // Redirect to '/
     register' instead of '/registration'
11   { path: 'register', component: RegistrationComponent },  // Keep this lowercase for
     consistency
12   { path: 'login', component: LoginComponent },            // Path should be lowercase
     'login'
13   { path: 'home', component: HomeComponent }
14 ];
15
16 @NgModule({
17   imports: [RouterModule.forRoot(routes)],
18   exports: [RouterModule]
19 })
20 export class AppRoutingModule { }
21
```

Backend Series Ep 1

Assignment 3

a. Create a Node.JS Application which serves a static website.
b. Create four API using Node.JS, ExpressJS and MongoDB for CURD Operations on assignment 2.C.

🔹 Static Website kya hoti hai?

Static website wo hoti hai jisme HTML, CSS aur JS files fix hoti hain. Inka content server par predefined hota hai — user input ke according change nahi hota.

📄 Example:

- Portfolio site
- Resume page
- Simple landing page

- Node.js + Express ka use kyu karte hain?

Feature	Fayda (Benefit)
Node.js	JavaScript-based runtime hai jo fast aur scalable hota hai.
Express.js	Node ka framework hai — easy routing aur file handling deta hai.

🔹 express.static() kya karta hai?

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

📁 Iska matlab hai:

Aapke project ke public folder se saare static files (HTML, CSS, JS, images) **direct browser ko send kiye jaayenge**, bina kisi backend processing ke.

User Request:

- User types URL in browser → http://localhost:3000

⬇

Express Server:

- Express receives GET request at "/"

⬇

Static Middleware:

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

⬇

File Lookup:

- Express searches public/index.html (default file)

⬇

File Found:

- index.html found → sent to browser

⬇

Browser Render:

- Browser loads HTML
- Loads linked CSS (style.css)
- Loads JS (script.js)

🎯 **Real-World Use Case**

- Simple Portfolio ya Resume banake serve karna
- College Project ka frontend test karna
- Static documentation ya tutorials

🔑 **Fayde (Advantages)**

✅ Super fast response (kyunki file directly bheji ja rahi hai)
✅ Backend config simple hota hai
✅ Extend karna easy hai — aage chalke APIs bhi add kar sakte ho
✅ Node.js asynchronous hai — zyada users ko handle kar sakta hai

1. **Initialize Your Project**

```
mkdir static-website
cd static-website
npm init -y
```

2. Install Express

```
npm install express
```

3. **Create Project Structure**

```
static-website/
│
├── public/
│   ├── index.html
│   ├── style.css
│   └── script.js
│
└── server.js
```

server.js | index.html

public > index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Static Website</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Welcome to My Static Website</h1>
  <p>This is served using Node.js and Express.</p>
  <script src="script.js"></script>
</body>
</html>
```

```css
body {
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
  padding: 20px;
  text-align: center;
}
```

server.js JS script.js ×

ublic > JS script.js

```
1  console.log('Website loaded successfully!');
2
```

server.js

server.js > ...

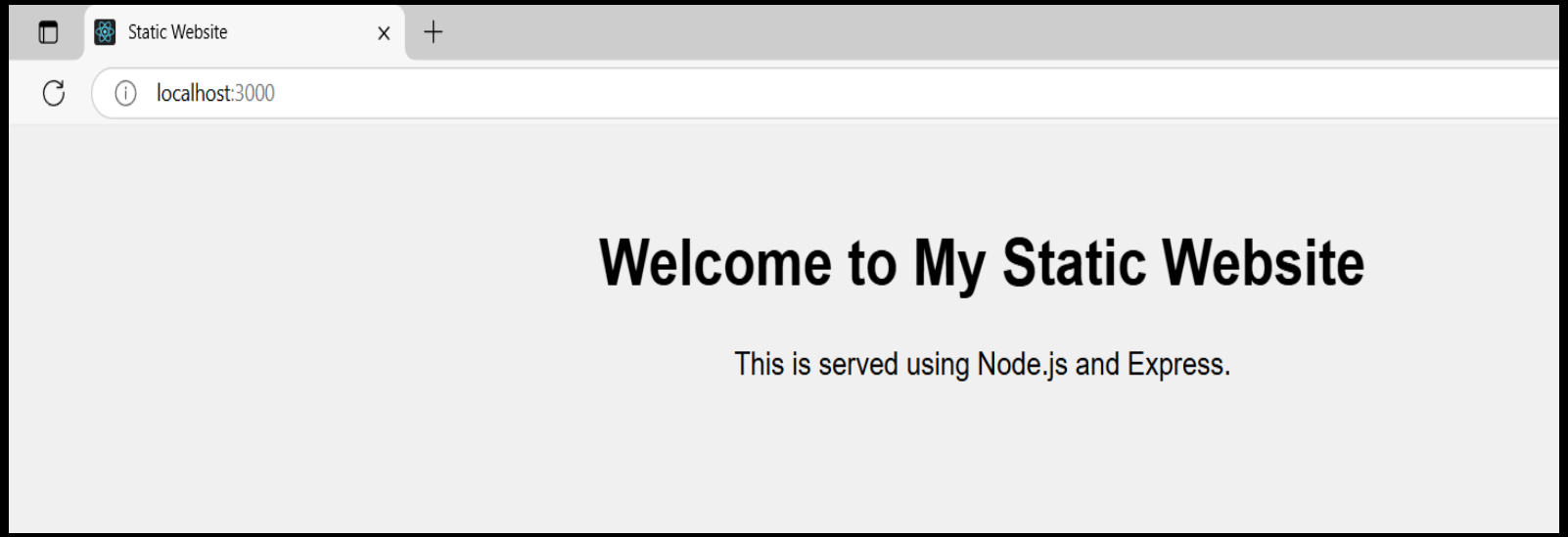
```javascript
const express = require('express');
const app = express();
const PORT = 3000;

// Serve static files from "public" directory
app.use(express.static('public'));

app.listen(PORT, () => {
  console.log(`Server is running at http://localhost:${PORT}`);
});

```

```
PS C:\Users\Admin\static-website> node server.js
Server is running at http://localhost:3000
```


Static Website
localhost:3000
Welcome to My Static Website
This is served using Node.js and Express.

Node.js + Express + MongoDB Based CRUD API Project for Student Data Management

◆ What is a CRUD API?

CRUD ka full form hota hai:
•**C** – Create (Naya data add karna)
•**R** – Read (Data fetch karna)
•**U** – Update (Existing data update karna)
•**D** – Delete (Data delete karna)

API (Application Programming Interface) ke through hum frontend ya tools (Postman) se backend ke saath interact karte hain.

- Tools and Technologies Used:

Technology	Role
Node.js	JavaScript runtime environment (server side JavaScript)
Express.js	Lightweight backend web framework
MongoDB	NoSQL database jisme hum data store karte hain
Mongoose	MongoDB ke liye ODM (Object Data Modeling) library
Postman	API testing tool

- **Project Flow:**

1.User API ko call karta hai (via Postman ya browser).

2.Request server (Express) tak pahuchti hai.

3.Server request ko handle karta hai (GET, POST, PUT, DELETE).

4.MongoDB me data add, read, update ya delete hota hai.

5.Server response return karta hai (success ya data).

Method	Use (काम क्या करता है)	One Line Explanation
GET	Data **read/fetch** karne ke liye	"Server se data lene ke liye use hota hai."
POST	Naya data **add/create** karne ke liye	"Server me naya data bhejne ke liye use hota hai."
PUT	Existing data ko **update** karne ke liye	"Pehle se maujood data me changes karne ke liye."
DELETE	Data ko **remove/delete** karne ke liye	"Server se kisi data ko delete karne ke liye hota hai."

🛠 STEP 1: Install Node.js

Agar Node.js install nahi hai:
👉 Download from: https://nodejs.org

Check version:

node -v npm -v

📁 STEP 2: Create Project Folder

```
mkdir student-api
cd student-api
```

📦 STEP 3: Initialize Node Project

```
npm init -y
```

This creates a package.json file.

📚 STEP 4: Install Required Packages

```
npm install express mongoose body-parser dotenv
```

- express → Backend framework
- mongoose → MongoDB se connect karne aur query karne ke liye
- body-parser → Request body ko JSON me convert karta hai
- dotenv → Secret variables manage karne ke liye

✍️ STEP 5: Create Files and Folders

```
student-api/
├── server.js        ◆ Main backend file
├── .env             ◆ Secret config file
├── models/
│   └── student.js   ◆ MongoDB schema
```

MongoDB

Log in to your account

Don't have an account? Sign Up

You have successfully logged out.

Google

GitHub

Or with email and password

Email Address

Next

MongoDB 8.0 is here

Up to 32% higher throughput, improved horizontal scaling, expanded queryable encryption capabilities, and more.

See everything that's new →

Clusters

Create cluster

Cluster0

Connect

Edit configuration

Monitoring for Cluster0 is paused.

Monitoring will automatically resume when you connect to your cluster.

Visit the documentation for more info

+ Add Tag

Connect to Cluster0

Set up connection security — 2 Choose a connection method — 3 Connect

Connect to your application

Drivers
Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)

Access your data through tools

Compass
Explore, modify, and visualize your data with MongoDB's GUI

Shell
Quickly add & update data using MongoDB's Javascript command-line interface

MongoDB for VS Code
Work with your data in MongoDB directly from your VS Code environment

Atlas SQL
Easily connect SQL tools to Atlas for data analysis and visualization

Go Back

Close

Connect to Cluster0

Set up connection security — Choose a connection method — 3 Connect

Connecting with MongoDB for VS Code

1. Install MongoDB for VS Code.

In VS Code, open "Extensions" in the left navigation and search for "MongoDB for VS Code." Select the extension and click install.

2. In VS Code, open the Command Palette.

Click on "View" and open "Command Palette."
Search "MongoDB: Connect" on the Command Palette and click on "Connect with Connection String."

3. Connect to your MongoDB deployment.

Paste your connection string into the Command Palette.

```
mongodb+srv://kbtug22146:<db_password>@cluster0.0iigr1v.mongodb.net/
```

Replace **<db_password>** with the password for the **kbtug22146** user. Ensure any options are URL encoded. You can edit your database user password in Database Access.

4. Click "Create New Playground" in MongoDB for VS Code to get started.

Learn more about Playgrounds

RESOURCES

- Connect to MongoDB through VSCode
- Access your Database Users
- Explore your data with playgrounds
- Troubleshoot Connections

Go Back

Done

JAYESH'S ORG - 2025-04-01 > FOODWEB

Clusters

Find a database deployment...

Cluster0 | Connect | View Monitoring | Browse Collections | ···

Monitoring for Cluster0 is Paused

Monitoring will automatically resume when you connect to

Visit the documentation for more info.

JAYESH'S ORG - 2025-04-01 > FOODWEB

Database Access

Database Users | Custom Roles

+ ADD NEW DATABASE USER

JAYESH'S ORG - 2025-04-01 > FOODWEB > DATABASES

Cluster0

Overview | Real Time | Metrics | **Collections** | Atlas Search | Query Insights | Performance Advisor

DATABASES: 2 COLLECTIONS: 9

+ Create Database

Search Namespaces

- food-del
  - foods
  - orders
  - users
- sample_mflix

food-del.foods

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 571B TOTAL DOCUMENTS: 4 INDEXES TOTAL SIZE: 36KB

Find | Indexes | Schema Anti-Patterns 0 | Aggregation | Search Indexes

Generate queries from natural language in Compass

Filter

```
Type a query: { field: 'value' }
```

QUERY RESULTS: 1-4 OF 4

Insert Document

To collection foods

VIEW

```
1 {"_id":{"$oid":"6818e32a3fedf2d59e01b1ea"},
2 }
```

STEP 6: .env File (MongoDB Connection)

Create .env file:

```
PORT=3000
MONGODB_URI=your_mongodb_connection_string
```

👉 Get connection string from MongoDB Atlas

🧠 STEP 7: Create models/student.js

```
const mongoose = require("mongoose");

const studentSchema = new mongoose.Schema({
  name: String,
  marks: Number
});

module.exports = mongoose.model("Student", studentSchema);
```

🚀 STEP 8: Create server.js

```
const express = require("express");
const mongoose = require("mongoose");
const bodyParser = require("body-parser");
require("dotenv").config(); // .env file ko load karta hai

const Student = require("./models/student");
const app = express();

app.use(bodyParser.json());
```

```
// MongoDB connect
mongoose.connect(process.env.MONGODB_URI)
.then(() => console.log("MongoDB Connected"))
.catch((err) => console.log("Mongo Error:", err));
```

```
// Get all students
app.get("/students", async (req, res) => {
  const students = await Student.find();
  res.send(students);
});

// Get student by name
app.get("/students/:name", async (req, res) => {
  const { name } = req.params;
  const students = await Student.find({ name });
  res.send(students);
});
```

```
// Add student
app.post("/add-student", async (req, res) => {
  const { name, marks } = req.body;
  const newStudent = new Student({ name, marks });
  await newStudent.save();
  res.send("Student added");
});

// Delete student
app.delete("/delete-student/:name", async (req, res) => {
  const { name } = req.params;
  await Student.findOneAndDelete({ name });
  res.send("Student deleted");
});
```

```
//  Update student
app.put("/update", async (req, res) => {
  const { name, marks } = req.body;
  const updated = await Student.findOneAndUpdate(
    { name },
    { $set: { marks } },
    { new: true }
  );
  res.send(updated);
});

app.listen(process.env.PORT, () => {
  console.log(`Server is running on port ${process.env.PORT}`);
});
```

🧪 STEP 9: Test API using Postman

Method	URL	Body/Param Example	Description
POST	/add-student	{ "name": "Amit", "marks": 90 }	Create new student
GET	/students	—	Get all students
GET	/students/Amit	—	Get student by name
DELETE	/delete-student/Amit	—	Delete student
PUT	/update	{ "name": "Amit", "marks": 95 }	Update student

POST http://localhost:3000/add-student Send

Status: 200 OK Size: 13 Bytes Time: 37 ms

Query Headers 2 Auth Body 1 Tests Pre Run

Response Headers 6 Cookies Results Docs

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1  { "name": "Amit", "marks": 90 }
```

```
1  Student added
```

GET http://localhost:3000/students Send

Query Headers 2 Auth Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1
```

Status: 200 OK Size: 195 Bytes Time: 23 ms

Response Headers 6 Cookies Results Docs


```
[
  {
    "_id": "6818de449a8d151dc346e89b",
    "name": "jk",
    "marks": 20
  },
  {
    "_id": "6818dff5c7ab255431890610",
    "name": "Amit",
    "marks": 95,
    "__v": 0
  },
  {
    "_id": "6818e070c7ab255431890613",
    "name": "Amit",
    "marks": 90,
    "__v": 0
  }
]
```

GET http://localhost:3000/students/Amit Send

Status: 200 OK Size: 137 Bytes Time: 145 ms

Query Headers 2 Auth Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1
```

Response Headers 6 Cookies Results Docs


```
1  [
2    {
3      "_id": "6818dff5c7ab255431890610",
4      "name": "Amit",
5      "marks": 95,
6      "__v": 0
7    },
8    {
9      "_id": "6818e070c7ab255431890613",
10     "name": "Amit",
11     "marks": 90,
12     "__v": 0
13   }
14 ]
```

DELETE http://localhost:3000/delete-student/Amit Send

Query Headers 2 Auth Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1
```

Status: 200 OK Size: 15 Bytes Time: 32 ms

Response Headers 6 Cookies Results Docs

```
1  Student deleted
```

PUT http://localhost:3000/update Send

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1  { "name": "Amit", "marks": 95 }
```

Status: 200 OK Size: 67 Bytes Time: 119 ms

Response Headers 6 Cookies Results Docs

```
1  {
2    "_id": "6818dff5c7ab255431890610",
3    "name": "Amit",
4    "marks": 95,
5    "__v": 0
6  }
```

"Kya aap ne itne easy way mein padha hai?"

1. Version Control System (VCS)

•**VCS** ek system hai jo aapke code ke different versions ko track karta hai.

•**Types**:

- **Local VCS**: Local machine pe changes track karta hai.
- **Centralized VCS**: Ek central server pe code rakhte hain.
- **Distributed VCS**: Har developer apne local machine pe full copy rakhta hai (e.g., Git).

Example: Git mein tumhare paas apne code ka pura history hota hai aur tum kabhi bhi purani version pe wapas ja sakte ho.

Example:
Maan lo tum ek **calculator app** bana rahe ho. Tumne pehle **addition** feature banaya aur uska code likha. Phir tumne usme **subtraction** add kiya. Agar tumhe kabhi purane version (jaise addition wala code) pe wapas jaana ho, toh **version control** tumhe yeh allow karta hai. Tumhare **code ke har change ka snapshot** liya jaata hai, taaki agar kuch galat ho jaaye, toh tum purana code wapas le sakte ho.

Git	GitHub
Git ek **version control** system hai.	GitHub ek **remote repository hosting platform** hai.
Git locally install hota hai, aur aap apne system pe code track karte ho.	GitHub ek **cloud-based** platform hai jahan code ko online store kiya jaata hai.
Git ka use code track karne aur versions ko manage karne ke liye hota hai.	GitHub ka use code share karne aur team collaboration ke liye hota hai.
Git ko aap apne computer pe use karte ho.	GitHub ka use aap **online** karte ho, jahan aap apne code ko share karte ho.

Command	Explanation	Example	Purpose
git init	Git repository initialize karne ke liye.	git init	Ek nayi repository banata hai jahan aap apne code ko track kar sakte ho.
git add .	All changes ko stage (prepare) karne ke liye.	git add .	Saare changes ko staging area mein daalna.
git add <filename>	Specific file ko stage karne ke liye.	git add index.html	Sirf index.html file ko stage karna.
git commit -m "message"	Code ko save (commit) karne ke liye, ek message ke saath.	git commit -m "Added login feature"	Code changes ko commit karte waqt unhe save karte ho aur message ke through explain karte ho.
git push origin main	Local code ko GitHub repository pe bhejne ke liye.	git push origin main	Local repository ke changes ko remote GitHub repository (origin) pe upload karte ho.
git pull origin main	Remote repository se changes ko apne local machine pe lana.	git pull origin main	GitHub se latest code changes apne local system pe download karna.
git branch <branch-name>	Nayi branch banane ke liye.	git branch feature/login	Nayi branch (jaise feature/login) create karna.
git checkout <branch-name>	Kisi branch pe switch karna.	git checkout feature/login	Tum feature/login branch pe switch kar jaoge.
git merge <branch-name>	Ek branch ko doosri branch ke saath merge karna.	git merge feature/login	feature/login branch ko main branch ke saath merge karna.

Summary:

- **Git** ka use apne code ko version track karne aur changes ko save karne ke liye hota hai.
- **GitHub** ka use code ko **remote server** pe store aur share karne ke liye hota hai.
- **Git commands** ka flow simple hai: init → add → commit → push → pull → branch → merge.

🔧 STEP 1: GitHub par account banao

1. Browser mein jao: https://github.com
2. Click karo **Sign up**.
3. Apna **email**, **username**, **password** daalo.
4. Verification complete karo aur account create kar lo.

Check installation:

```
git --version
```

💻 **STEP 2: Git install karo (Agar system mein nahi hai)**

Windows:

1. Visit karo: https://git-scm.com/downloads
2. Apne OS ke hisaab se download karo.
3. Next-next karke install kar lo (default settings chalne do).

📁 STEP 3: Local Project Folder banao

```
mkdir my-project
cd my-project
```

Ya agar project pehle se bana hai:

```
cd path/to/your-project
```

🔗 STEP 4: Git initialize karo apne project mein

```
git init
```

Ye command tumhare folder ko Git repository bana degi.

📄 STEP 5: File banao aur changes stage karo

```
echo "# My First GitHub Project" > README.md

git add .
```

STEP 6: First commit karo

```
git commit -m "First commit"
```

🌐 **STEP 7: GitHub pe nayi repository banao**

1. GitHub pe login karo.
2. Top-right corner mein **+** icon → click **"New repository"**.
3. Repository ka naam do (e.g., my-project)
4. Initialize with README **untick karo** (kyunki tumhare paas local mein file hai).
5. Create repository.

🔗 STEP 8: Remote URL add karo

GitHub tumhe ek URL dega, e.g.:

https://github.com/username/my-project.git

Use add karo as remote:

```
git remote add origin https://github.com/username/my-project.git
```

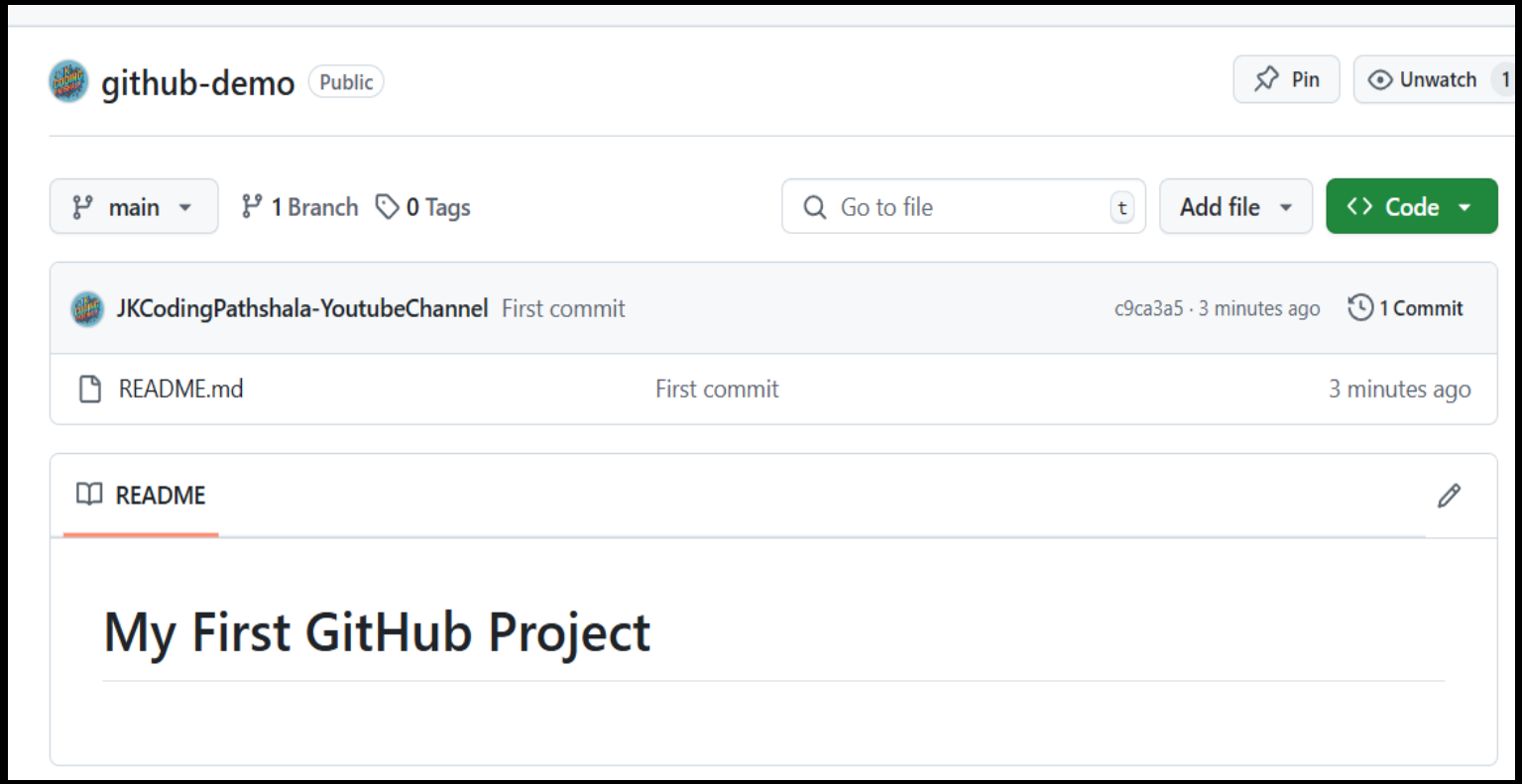
🚀 STEP 9: Code push karo GitHub pe

```
git branch -M main
git push -u origin main
```

Agar password maange toh GitHub ka personal access token use karo (not your password).

```
PS C:\Users\Admin\Desktop\github> cd my-project
PS C:\Users\Admin\Desktop\github\my-project> git init
Initialized empty Git repository in C:/Users/Admin/Desktop/github/my-project/.git/
PS C:\Users\Admin\Desktop\github\my-project> echo "# My First GitHub Project" > README.md
PS C:\Users\Admin\Desktop\github\my-project> git add .
PS C:\Users\Admin\Desktop\github\my-project> git commit -m "First commit"
[master (root-commit) c9ca3a5] First commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
PS C:\Users\Admin\Desktop\github\my-project> git remote add origin https://github.com/JKCodingPathshala-YoutubeChannel/github-demo.git
PS C:\Users\Admin\Desktop\github\my-project> git branch -M main
PS C:\Users\Admin\Desktop\github\my-project> git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 271 bytes | 67.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/JKCodingPathshala-YoutubeChannel/github-demo.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Ab Tumhara Code GitHub Pe Successfully Push Ho Gaya Hai!



🔐 Personal Access Token (PAT) kaise banao (first time only)

1. GitHub pe jao → top-right pe profile icon → **Settings**
2. Left side mein **Developer settings** → **Personal Access Tokens** → **Fine-grained tokens**
3. Token banao, repo access allow karo.
4. Generate token → copy karke use karo jab Git push karega.

Example:

Maan lijiye aap push kar rahe ho:

```
git push origin main
```

Uske baad, terminal mein yeh prompt dikhai dega:

```
Username for 'https://github.com': your-username
Password for 'https://github.com':
```

Yahan **Password** ke jagah, aapko **apna GitHub Personal Access Token (PAT)** paste karna hoga.

Kuch is tareeke se:

Password for 'https://github.com': <paste-your-token-here>

Important: Jab aap token paste karenge, terminal mein kuch bhi dikhai nahi dega (koi asterisks ***** ya koi characters nahi dikhenge).
Lekin jab aap paste karenge, token **safely** send ho jayega GitHub ko.
Aur agar aap **Git Credential Manager** ko setup kar lete ho (jo maine pehle bataya), toh aapko har baar token dalne ki zarurat nahi padegi.

Assignment 4

a. Create a simple Mobile Website using jQuery Mobile.
b. Deploy/Host Your web application on AWS VPC or AWS Elastic Beanstalk. Mini Project

jQuery Mobile ka Simple Theory Concept:

jQuery Mobile ek lightweight framework hai, jo specially **mobile-friendly websites** banane ke liye design kiya gaya hai. Iska main goal hai ki aapko quickly mobile apps jaise websites banane ka option mile, bina jyada coding ke.

Step 1: jQuery Mobile Theme Builder se Theme Select Karna

•**Go to jQuery Mobile Theme Builder**:

•Open jQuery Mobile Theme Builder.

•**Select a Theme**:

•Yahan pe aapko theme ka preview milega. Aap apne website ke liye color, font, button style, etc.
•
•select kar sakte hain.
•**Colors** select karne ke liye **"Color"** tab pe click karein aur apne color choices ko set karein.
•**Download the Theme**:

•Jab aapko apna theme pasand aa jaye, toh **"Download"** button pe click karein.

•Yeh aapko **ZIP file** download karne ke liye kahega. Download karne ke baad, ZIP file ko extract kar lein.

ThemeRoller
For jQuery Mobile

jQuery Mobile Version 1.4.5

undo | redo | Inspector off | Download theme zip file | Import or upgrade | Help center

Global | A | B | C | +

Swatch A — Delete — Duplicate

- Page
- Header/Footer Bar
- Body
- Link
- Button: Normal
  - TEXT COLOR
  - TEXT SHADOW 0 1px 0 #444444
  - BACKGROUND #ed1c24
  - BORDER #820f13
- Button: Hover
- Button: Pressed
- Active State

Drag a color onto an element below

LIGHTNESS — SATURATION

Recent Colors — colors...

jQuery mobile

Show alternative icons in preview — See icon demos for usage.

A	B	C
Body	Body	Body
Sample text and links.	Sample text and links.	Sample text and links.
List Header	List Header	List Header
Read-only list item	Read-only list item	Read-only list item
Linked list item	Linked list item	Linked list item
Radio	Radio	Radio
Checkbox	Checkbox	Checkbox
On / Off	On / Off	On / Off
Option 1	Option 1	Option 1

Add swatch...

Step 2: Theme Files ko Apne Project mein Add Karna

1.Extract the ZIP File:

- ZIP file ko extract karne ke baad, aapko **themes** folder milega.
- Is folder ko apne project folder mein add karna hai. Jaise ki index.html ke saath
- ek **themes** folder ho, jisme aapka CSS file rahega.

2.Include the Theme CSS Files in Your HTML:

- Ab aapko apne index.html file mein theme ka CSS link include karna hoga.
- Aap jo theme download kar chuke hain, uska path apne HTML mein include karein.

EXPLORER

OPEN EDITORS

index.html

JQUERY-MOBILE-THEME-184734-0

themes

images

A.css

A.min.css

jquery.mobile.icons.min.css

index.html

index.html

index.html > html > body > div#page2

```html
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>jQuery Mobile: Theme Download</title>
  <link rel="stylesheet" href="themes/A.min.css" />
  <link rel="stylesheet" href="themes/jquery.mobile.icons.min.css" />
  <link rel="stylesheet" href="http://code.jquery.com/mobile/1.4.5/jquery.mobile.structure-1.4.5.min.css" />
  <script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
  <script src="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js"></script>
</head>
<body>
```

```html
<!-- Page 1: Login -->
<div data-role="page" id="page1">
  <div data-role="header">
      <h1>Login</h1>
  </div>
  <div data-role="content">
      <label>Email:</label>
      <input type="email">
      <label>Password:</label>
      <input type="password">
      <button><a href="#page2">Login</a></button>
      <p>New user? <a href="#page2">Register here</a></p>
  </div>
/div>/#page1

<!-- Page 2: Register -->
<div data-role="page" id="page2">
  <div data-role="header">
      <h1>Register</h1>
  </div>
  <div data-role="content">
      <label>Name:</label>
      <input type="text">
      <label>Email:</label>
      <input type="email">
      <label>Password:</label>
      <input type="password">
      <button><a href="#page1">Register</a></button>
  </div>
```

Login

Email:

Password:

Login

New user? Register here

Register

Name:

Email:

Password:

Register

jayesh_kande_

Jayesh Kande

16	275	276
posts	followers	following

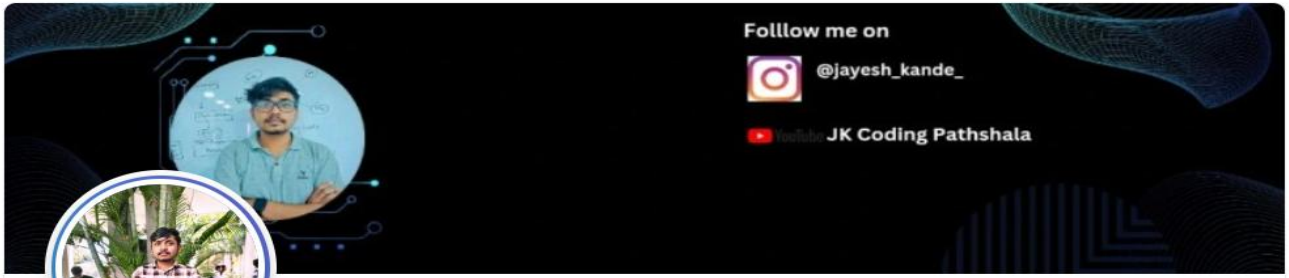
23

रास्ते बदलो, मंजिल नहीं

yt.openinapp.co/0y0qd

LinkedIn

Articles | People | Learning | Jobs | Games



Jayesh Kande

Third-Year IT Engineering Student | Aspiring Web Developer | Java Enthusiast | Data Structures & Algorithms Learner | Proficient in C, C++, Java, and MERN Stack | AI + Web Development Project Enthusiast

Nashik, Maharashtra, India · Contact Info

494 followers · 495 connections

See your mutual connections

Join to view profile

Message

Kbt engineering college nashik

✨ Thank You for Watching! ✨

📲 Follow us on Instagram:**@jayesh_kande_**
🔗 Connect with us on LinkedIn:[**Jayesh Kande]**