1.Demonstrate print "Hello Word" with Angular js. It specifies the Model, View, Controller part of an Angular js app.

```
<!DOCTYPE html>
<html ng-app="helloApp">
<head>
  <title>AngularJS Hello Word</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>
  <!-- View -->
  <div ng-controller="HelloController">
    <h1>{{ message }}</h1>
  </div>
  <!-- Controller Script -->
  <script>
    // Module (App)
    var app = angular.module('helloApp', []);
    // Controller
    app.controller('HelloController', function($scope) {
      // Model
      $scope.message = "Hello Word";
    });
  </script>
</body>
</html>
```

---

2. Demonstrate angular js script to implement Built-in Directives.

```
<!DOCTYPE html>
<html ng-app="directiveApp">
<head>
 <title>AngularJS Built-in Directives</title>
 <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>
 <div ng-controller="DirectiveController">
  <!-- ng-model -->
  <label>Enter your name:</label>
  <input type="text" ng-model="name">
  <!-- ng-bind -->
  <p>Hello, <span ng-bind="name"></span>!</p>
  <!-- ng-if -->
  <p ng-if="name">You typed your name!</p>
  <!-- ng-show / ng-hide -->
  <p ng-show="name.length < 5">Name is short!</p>
  <p ng-hide="name">Name field is empty.</p>
  <!-- ng-repeat -->
  <h3>Fruits List</h3>
  <ul>
   <li ng-repeat="fruit in fruits">{{ fruit }}</li>
  </ul>
  <!-- ng-click -->
```

```
    <button ng-click="addFruit()">Add Mango</button>
  </div>
  <script>
    var app = angular.module('directiveApp', []);
    app.controller('DirectiveController', function($scope) {
      $scope.name = "";
      $scope.fruits = ["Apple", "Banana", "Orange"];
      $scope.addFruit = function() {
        $scope.fruits.push("Mango");
      };
    });
  </script>
</body>
</html>
```

3. Demonstrate angular js script to add modules and controller.

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
  <title>AngularJS Module & Controller Example</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>
  <!-- View: Bind Controller to this section -->
  <div ng-controller="MyController">
    <h1>{{ greeting }}</h1>
  </div>
  <!-- AngularJS Script -->
  <script>
    // Step 1: Create the Module
    var app = angular.module('myApp', []);
    // Step 2: Create the Controller inside the Module
    app.controller('MyController', function($scope) {
      $scope.greeting = "Hello from AngularJS!";
    });
  </script>
</body>
</html>
```

4. Demonstrate simple form using angular js script.

```
<!DOCTYPE html>
<html ng-app="formApp">
<head>
  <title>Simple AngularJS Form</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
  <style>
    .error {
      color: red;
    }
  </style>
```

```
</head>
<body>
  <div ng-controller="FormController">
    <form name="userForm" ng-submit="submitForm()" novalidate>
      <!-- Name Field -->
      <label>Name:</label>
      <input type="text" name="name" ng-model="user.name" required />
      <span class="error" ng-show="userForm.name.$touched && userForm.name.$invalid">Name is required.</span>
      <br><br>
      <!-- Email Field -->
      <label>Email:</label>
      <input type="email" name="email" ng-model="user.email" required />
      <span class="error" ng-show="userForm.email.$touched && userForm.email.$invalid">Valid email is
required.</span>
      <br><br>
      <!-- Submit Button -->
      <button type="submit" ng-disabled="userForm.$invalid">Submit</button>
    </form>
    <!-- Output -->
    <div ng-if="submitted">
      <h3>Form Submitted!</h3>
      <p><strong>Name:</strong> {{ user.name }}</p>
      <p><strong>Email:</strong> {{ user.email }}</p>
    </div>
  </div>
  <script>
    // Create module
    var app = angular.module('formApp', []);
    // Create controller
    app.controller('FormController', function($scope) {
      $scope.user = {};
      $scope.submitted = false;
      $scope.submitForm = function() {
        if ($scope.userForm.$valid) {
          $scope.submitted = true;
        }
      };
    });
  </script>
</body>
</html>
```

5. Demonstrate the use of JSON in a webpage.

```
<!DOCTYPE html>
<html>
<head>
  <title>Using JSON in a Web Page</title>
</head>
<body>
  <h2>User Info (Loaded from JSON)</h2>
  <div id="userInfo"></div>
```

```
 <script>
  // Sample JSON data (usually this comes from a server)
  var jsonData = `{
    "name": "devendra",
    "age": 25,
    "email": "dev@example.com",
    "skills": ["HTML", "CSS", "JavaScript"]
  }`;

  // Parse the JSON string into a JavaScript object
  var user = JSON.parse(jsonData);

  // Display data in the web page
  var output = "<p><strong>Name:</strong> " + user.name + "</p>";
  output += "<p><strong>Age:</strong> " + user.age + "</p>";
  output += "<p><strong>Email:</strong> " + user.email + "</p>";
  output += "<p><strong>Skills:</strong> " + user.skills.join(", ") + "</p>";

  // Add to the HTML DOM
  document.getElementById("userInfo").innerHTML = output;
 </script>
</body>
</html>
```

---

6. Demonstrate Installation steps of MongoDB and Connect to the database

1. Installation Steps for MongoDB (Community Edition)

🔹 **Step 1: Download MongoDB**

- Go to: https://www.mongodb.com/try/download/community
- Choose your OS (Windows, macOS, Linux)
- Select the **MSI (for Windows)** or **TGZ/ZIP** for others
- Click **Download**

🔹 **Step 2: Install MongoDB**

- Run the installer (e.g., mongodb-windows-x86_64-x.x.x-signed.msi)
- Follow the setup wizard:
    - Choose **Complete** setup
    - Make sure to select the **Install MongoDB Compass** option if not already checked
- Click **Install**

🔹 **Step 3: Add MongoDB to PATH (Windows)**

- The installer usually does this automatically.
- To verify:
    1. Open **Command Prompt**
    2. Run: mongo --version
    3. You should see the installed version info.

2. Start MongoDB Server

🟢 **Option 1: Run as a Service (default in installation)**

MongoDB starts as a Windows service automatically.

🟢 **Option 2: Manually from Terminal (macOS/Linux)**

```
mongod --dbpath "C:\data\db"  # Make sure this folder exists
```

Default MongoDB port is **27017**

✅ **3. Connect to MongoDB Using Compass**

🔹 **Step 1: Open MongoDB Compass**

- Launch the app from your installed programs.

🔹 **Step 2: Use Default Connection URI**

In Compass:mongodb://localhost:27017

🔹 **Step 3: Click "Connect"**

- This connects to your **local MongoDB server**.
- You can now:
  - View existing databases
  - Create new ones
  - Add collections and documents

Optional: Create and View a Database

🔸 **Create New Database**

1. In Compass, click on **"Create Database"**
2. Name your database (e.g., studentDB)
3. Create a collection (e.g., students)

🔸 **Insert a Document**

```
{
 "name": "Devendra Pawar",
 "age": 25,
         "course": "MCA"
}
```

You'll see this in your Compass GUI!

---

7. Demonstrate Create a Table in MongoDB

Steps to Create a Table (Collection) in MongoDB using Compass

🔹 **Step 1: Open MongoDB Compass**

- Launch MongoDB Compass from your applications or start menu.

🔹 **Step 2: Connect to MongoDB**

- Use the default connection string (if MongoDB is running locally):mongodb://localhost:27017
- Click **"Connect"**

🔹 **Step 3: Create a New Database and Collection**

1. **Click "Create Database"** on the left panel.
2. A dialog box will appear:
   - **Database Name**: studentDB (or any name you like)
   - **Collection Name**: students (this is like the table name)
3. Click **"Create Database"**

Compass will now create:

- A new **database** called studentDB
- A new **collection** inside it called students

🔹 **Step 4: Insert Data (Rows)**

1. Select the studentDB database from the left panel.
2. Click the students collection.
3. Click **"Insert Document"**.
4. Enter a sample document (like a row in a table):

```
{
 "name": "Devendra",
 "age": 25,
 "department": "MCA"
}
```

5. Click **"Insert"**

Now you've successfully:

- Created a MongoDB "table" (collection)
- Inserted a "row" (document)

8.Demonstrate CRUD Operations on MongoDB tables

Database Used in Example

- **Database**: studentDB
- **Collection (Table)**: students

1. CREATE – Insert a Document

1. Open **Compass** and connect to mongodb://localhost:27017.
2. Select your database (studentDB), then the students collection.
3. Click **"Insert Document"**.
4. Add data:

```
{
  "name": "Devendra Pawar",
  "age": 25,
  "department": "MCA"
}
```

5. Click **"Insert"**.

This is equivalent to inserting a new row in a table.

2. READ – View/Retrieve Documents

1. In the students collection, Compass automatically shows all documents.
2. You can use the **Filter bar** to run queries:
   - Show all students in MCA:

```
{ "department": "MCA" }
```

Equivalent to SELECT * FROM students WHERE department = 'MCA'

✅ 3. UPDATE – Modify a Document

1. Find the document you want to edit.
2. Click the **"Edit"** icon (pencil) on the right of that document.
3. Modify any field, e.g.:
   - Change "age": 25 to "age": 22
4. Click **"Update"**

Equivalent to UPDATE students SET age = 22 WHERE name = 'Devendra Pawar'

✅ 4. DELETE – Remove a Document

1. Locate the document you want to remove.
2. Click the **"Delete" (trash can)** icon next to the document.
3. Confirm deletion.

Equivalent to DELETE FROM students WHERE name = 'Devendra Pawar'