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

1)

```
<!DOCTYPE html>
 <html ng-app="helloApp">
<head>
<meta charset="utf-8">
<title>Hello World AngularJS</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="HelloController">
 <h1>{{ message }}</h1>
 <script>
        var app = angular.module('helloApp', []);
 app.controller('HelloController', function($scope) {
        $scope.message = "Hello World";
});
</script>
</body>
</html>
```

---

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

```
<!DOCTYPE html>
<html ng-app="myApp">  <head>
        <meta charset="UTF-8">
   <title>AngularJS Directives Example</title>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>            <body >
        <div ng-controller="MainController">
        <h2>ng-model Example</h2>
        <label for="name">Enter your name: </label>
        <input type="text" id="name" ng-model="name">
        <p>Hello, {{ name }}!</p>

        <!-- ng-repeat Example (Loop through items) -->
    <h2>ng-repeat Example</h2>
        <ul>
        <li ng-repeat="item in items">{{ item }}</li>
        </ul>
        <!-- ng-if Example (Conditional Rendering) -->
        <h2>ng-if Example</h2>
        <div ng-if="isVisible">
        <p>This paragraph is conditionally rendered using ng-if!</p>
        </div>
        <button ng-click="toggleVisibility()">Toggle Paragraph</button> </hr>
        <!-- ng-show / ng-hide Example (Show/Hide an Element) -->
        <h2>ng-show / ng-hide Example</h2>
        <div ng-show="isShown">This text is shown when ng-show condition is true.</div>
        <div ng-hide="isShown">This text is hidden when ng-hide condition is false.</div>
        <button ng-click="toggleShow()">Toggle Show Text</button>
        <button ng-click="toggleShow()">Toggle Hide Text</button>   </div>
        <script>
```

```
    var app = angular.module('myApp', []);
app.controller('MainController', function($scope) {
        $scope.name = '';
        $scope.items = ['Apple', 'Banana', 'Cherry', 'Grapes'];
        $scope.isVisible = true;
    $scope.isShown = true;
    $scope.toggleVisibility = function() {
  $scope.isVisible = !$scope.isVisible;
};
 $scope.toggleShow = function() {
  $scope.isShown = !$scope.isShown;
};  });  </script>  </body>  </html>
```

---

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

```
<!DOCTYPE html>  <html ng-app="myApp">  <head>
    <meta charset="UTF-8">
  <title>AngularJS Module and Controller</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="MyController">
    <h1>{{ greeting }}</h1>
    <p>{{ message }}</p>
    <script>
        var app = angular.module('myApp', []);
    app.controller('MyController', function($scope) {
    $scope.greeting = "Hello from AngularJS!";
```

```
$scope.message = "This is another message from the controller.";
});
</script>  </body>  </html>
```

---

4. Demonstrate simple form using angular js script.

```
<!DOCTYPE html>
<html ng-app="formApp">
<head>
    <meta charset="UTF-8">
  <title>AngularJS Simple Form</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body>
    <div ng-controller="FormController">
    <form name="userForm">
        <label>Name:</label><br>
    <input type="text" name="username" ng-model="user.name" required><br>
        <label>Email:</label><br>
    <input type="email" name="useremail" ng-model="user.email" required><br>
    <button type="submit" ng-disabled="userForm.$invalid">Submit</button>
    </form>
    <p>Name: {{ user.name }}</p>
    <p>Email: {{ user.email }}</p>    </div>
    <script>
    var app = angular.module('formApp', []);
  app.controller('FormController', function($scope) {
```

```
$scope.user = { };
});
</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>
   var jsonData = `{
  "name": "devendra",
  "age": 25,
  "email": "dev@example.com",
  "skills": ["HTML", "CSS", "JavaScript"]
 }`;
  var user = JSON.parse(jsonData);
  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>";
  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:

o Choose Complete setup

o 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:

o View existing databases

o Create new ones

o 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:

o Database Name: studentDB (or any name you like)

o 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"

Devendra Pawar

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:

o 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.:

o 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'