CMR INSTITUTE OF TECHNOLOGY	USN	

Internal Assessment Test –II, August 2025

Sub:	**Web Application Development**	**Code:**	**MMC205**	
	Answer Key	**Marks**	**OBE**	
			CO	RBT

1	What is Angular JS? Explain the following AngularJS directives:
	(i) ng_app (ii) ng_model (iii) ng_bind (iv) ng_init (v) ng_repeat
	
	1) ng-app= " "
	
	The app directive defines the area of AngularJS application. The syntax of app directive is ng-app = " "; In here the ng is the namespace of AngularJS and app is the application area of Angular JS.
	
	Example 2.1
	`<!DOCTYPE html>`
	`<html >`
	`<head>`
	`<title>AngularJS for beginners</title>`
	`<script src="js\angular.min.js"></script>`
	`</head>`
	`<body>`
	`<div ng-app=" ">`
	`The AngularJSapplication has been started.`
	`</div>`
	`</body>`
	`</html>`
	
	Output:
	The AngularJSapplication has been started.
		[10]	CO4	L4

Explanation:

In head section (**<script src = "js\angular.min.js"> </script>**), the framework of AngularJSis loaded, which means that we can use AngularJS framework now.
In body section (**),** AngularJS application can be written in here.
In last tag (** **), the AngularJSapplication is ended.

AngularJS extends HTML with ng-directives.

The ng-app directive defines an AngularJS application.

The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

The ng-bind directive binds application data to the HTML view.

2) Model Directive

ng-model = “data”

The model directive is used to bind the inputted value from HTML controls (input, checkbox and select etc.) to application data. The **ng-model = “data”** is the syntax of model directive. Let`s take an example for better understanding.

Example 2.2

```
<!DOCTYPE html>
<html >
<head>
   <title>AngularJSfor beginners</title> <script src="js\angular.min.js">
</script>
</head>
<body>
<div ng-app=""> <p>User Name: <br> <input type="text" ng-model =
"Username"></p> </div>
</body>
</html>
```

Output:

Username:

Explanation:

"ng-model = "Username"" binds the inputted value from HTML control to "Username".

Now you cannot see the result because this program is missing the code ** .**

3)Bind Directive

```
<p>ng-bind = "data"</p>
```

The bind directive is used to bind the data value to an html element ; the syntax of bind directive is ** ng-bind = "data" **. Let`s take an example for better understanding.

Example 2.3

```
<!DOCTYPE html>
<html >
<head>
   <title>AngularJSfor beginners</title> <script src="js\angular.min.js">
</script> </head>
<body>
<div ng-app=""> <p>User Name: <br> <input type="text" ng-model =
"Username"></p> <p ng-bind ="Username"></p> </div>
</body>
</html>
```

Open the notepad and paste the above mentioned code with .html extension, and type username "Ray Yao" in the input box.

Output:

User Name:
Ray Yao

Ray Yao

Explanation:

" ng-bind="Username" " binds the value of "Username" to tag, and "shows" its value .

In the above example, (**)** can update the value

of “Username” and writes it to tag.
ng-bind="data" can “output” the value of “data” in specified html element .

ng-model & ng-bind

ng-model = “data”: binds the inputted value from Html controls (textarea, checkbox, select, input and radio etc.) to “data”.

For example:

“ng-model = “Username”” binds the inputted value to “Username”.

(Just like assign the inputted value to “Username”.)

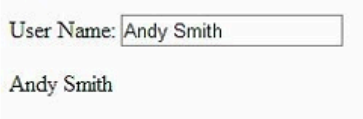
** ng-bind= “data” :** binds the “data” to an html element .

For example:

 binds the data “Username” to element and shows its value.

(Just like display the value of the “Username” in .)

4)Init Directive

	ng-init = "data = ‘value’” The init directive is used to initialize the data with a value. The syntax of init directive is **ng-init = "data = ‘value’”**. Let`s take an example for better understanding. **Example 2.4** <!DOCTYPE html> <html > <head> <title>AngularJSfor beginners</title> <script src="js\angular.min.js"> </script> </head> <body> ** User Name: <input type="text" ng-model = "Username"> ** </body> </html> Open the notepad and paste the above mentioned code with .html extension. **Output:**  **Explanation:** ng-init="Username= ‘Andy Smith’” initializes “Username” with a value “Andy Smith”. In the above example, (<div ng-app="" **ng-init = "Username= 'Andy Smith' "**>) initializes the value of User Name. When the page is loaded completely, the **5)Repeat Directive**			

ng-repeat = “variable in array”

The repeat directive works like a loop. The **ng-repeat** directive repeats to get th value of an array.

Example 2.5

```
<html >
<head>
   <title>AngularJSfor beginners</title> <script src="js\angular.min.js">
</script> <meta http-equiv="Content-Type" content="text/html; charset=UTF-
/> </head>
<body>
<div ng-app=""  ng-init = "ColorName = ['Pink', 'Red', 'Green', 'Blue',
'Black', 'White', 'Yellow', 'Gray']"> <p style="color:green; font-
weight:bold">Colours Name:</p> <ol>
   <li ng-repeat ="x in ColorName"> <p ng-bind="x"></p> </li>
</ol>
</div>
</body>
</html>
```

Open the notepad and paste the above code with .html extension.

2 Write a program to create custom forms in Bootstrap with an example. [10] CO3 L3

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <!-- Bootstrap 4 CSS -->
 <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.6.0/css/bootstrap.min.css">
</head>
<body>
 <div class="container my-4">
  <h3>Custom Form (Bootstrap 4)</h3>
  <form>
   <!-- Custom checkboxes -->
   <div class="custom-control custom-checkbox mb-2">
    <input type="checkbox" class="custom-control-input" id="cb1">
    <label class="custom-control-label" for="cb1">Custom checkbox</label>
   </div>
   <div class="custom-control custom-checkbox mb-3">
    <input type="checkbox" class="custom-control-input" id="cb2" checked>
    <label class="custom-control-label" for="cb2">Checked checkbox</label>
   </div>
```

	<!-- Custom radio buttons inline --> <input type="radio" id="radio1" name="radios" class="custom-control-input"> <label class="custom-control-label" for="radio1">Option 1</label> <input type="radio" id="radio2" name="radios" class="custom-control-input" checked> <label class="custom-control-label" for="radio2">Option 2</label> <!-- Custom select --> <label for="sel1">Choose an option</label> <select id="sel1" class="custom-select"> <option selected>Choose...</option> <option value="1">One</option> <option value="2">Two</option> <option value="3">Three</option> </select> <!-- Custom file input --> <div class="			

3	Write a jQuery program to show and hide elements.
	<!DOCTYPE html>
	<html lang="en">
	<head>
	<meta charset="UTF-8">
	<title>jQuery Show/Hide Demo</title>
	<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
	<style>
	#box { width: 200px; height: 100px; background: #cce; margin: 20px 0; padding: 10px; }
	</style>
	</head>
	<body>
	
	 jQuery Show / Hide / Toggle Demo
	
	<button id="hide-btn">Hide Box</button>
	<button id="show-btn">Show Box</button>
	<button id="toggle-btn">Toggle Box</button>
	
	 This is the box content.
	
	<script>
	$(document).ready(function() {
	$('#hide-btn').click(function() {
	$('#box').hide(); // hides immediately
	});
		[10]	CO2	L1

```
    $('#show-btn').click(function() {
      $('#box').show(); // shows immediately
    });

    $('#toggle-btn').click(function() {
      $('#box').toggle(500); // toggles visibility over 500ms
    });
  });
 </script>

</body>
</html>
```

4 What is Asynchronous in JavaScript? Explain the Types of Asynchronous with Example. [10] CO3 L3

What Is Asynchronous in JavaScript?

Asynchronous programming allows JavaScript to perform long-running operations—such as network requests, timers, or file reads—without blocking the main thread. Instead, JavaScript fires off these tasks and continues executing other code, handling results later when they're ready via callbacks/promises/etc. GeeksforGeeks+15InfoWorld+15Medium+15

🔁 Types of Asynchronous Patterns

1. Callbacks

A *callback* is a function passed to another function to be executed after an asynchronous task completes. Historically, this was JavaScript's only built-in approach for async logic—but it can lead to deeply nested callbacks ("callback hell") when chaining multiple operations. Reddit+3GeeksforGeeks+3blip.school+3

Callback Example:

```
function fetchData(callback) {
  setTimeout(() => {
    callback({ id: 1, name: 'Alice' });
  }, 1000);
}

fetchData(data => {
  console.log('Data:', data);
});
console.log('This logs before the data');
```

Output order:

```
This logs before the data
Data: {id: 1, name: 'Alice'}
```

---

2. Promises

Introduced in ES6, a **Promise** represents the eventual result (or failure) of an asynchronous operation. Promises provide structured chaining with .then() and .catch(), making code far more readable. This approach helps avoid callback nesting. GeeksforGeeks+3GeeksforGeeks+3blogs.shivangyadav.me+3

Promise Example:

```
function fetchData() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      const data = { id: 1, name: 'Alice' };
      resolve(data); // or reject('Error message');
    }, 1000);
  });
}

fetchData()
  .then(data => console.log('Data:', data))
  .catch(err => console.error('Error:', err));
```

---

3. Async / Await

In ES2017, **async / await** was introduced as syntactic sugar over promises—allowing asynchronous code to look and be written like synchronous code. This greatly improves readability and flow control. blip.school+15Medium+15Reddit+15

Async / Await Example:

```
function fetchData() {
  return new Promise((resolve) => {
    setTimeout(() => resolve({ id: 1, name: 'Alice' }), 1000);
  });
}

async function getData() {
  try {
    const data = await fetchData();
    console.log('Data:', data);
  } catch (error) {
    console.error('Error:', error);
  }
}

getData();
console.log('This logs first');
```

	Output order: This logs first Data: { id: 1, name: 'Alice' }			

5	What is $scope and explain how to use controllers with an example.
	
	What is Scope?
	Scope is a JavaScript object which contains model data.
	`function ($scope) { }`
	$scope is a parameter of JavaScript function which is called by a controller. Let`s take an example for understanding.
	Example 7.2
	<script>
	function **($scope)** {
	$scope.firstNumber = 23;
	$scope.secondNumber = 63;
	}
	</script> **Explanation:** In above example, the **$scope** is the parameter of the function ($scope) { }. $scope is an object in AngularJS.
	$scope.**firstNumber and **$scope.secondNumber are models used in HTML. Model data is accessed by the $scope object. We assign values to the model with following formula: "$scope.property = value".
	
	For example:
	$scope.firstNumber = 23;
	$scope.secondNumber = 63;
	
	Controllers
	
	What is Controller?
	The controller is basically a JavaScript object that controls the flow of data of an application. So the AngularJS application is controlled by Controller.
		[10]	CO4	L5

How to define Controller?

The **ng-controller** directive is used to define the Controller. We know that Controller is a JavaScript object which contains JavaScript function and properties. The syntax of Controller is as following:

```
<div ng-app=""  ng-controller="controllerName">
```

Example 7.1

```
<html>
<script src= "js\angular.min.js"></script> <body>
<div ng-app="Calculation" ng-controller="myController"> First Number:
<input type="number" ng-model="firstNumber"><br> Second Number:
<input type="number" ng-model="secondNumber"><br> <br>
Sum: {{firstNumber + secondNumber}}
</div>
<script>
var app = angular.module('Calculation', [ ]); app.controller('myController',
function($scope) {
   $scope.firstNumber = 4; $scope.secondNumber = 8; });
</script>
</body>
</html>
```

Output:

First Number: 4
Second Number: 8

Sum: 12

6	Explain with an example the bootstrap “Button classes” for different styles of Buttons in bootstrap. <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title>W3Schools-Style Bootstrap Buttons</title> <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet"> </head> <body> Button Variants	[10]	CO4	L4

	<button class="btn">Basic</button> <button class="btn btn-default">Default</button> <button class="btn btn-primary">Primary</button> <button class="btn btn-success">Success</button> <button class="btn btn-info">Info</button> <button class="btn btn-warning">Warning</button> <button class="btn btn-danger">Danger</button> <button class="btn btn-link">Link</button> <h2 class="mt-4">Outline Buttons</h2> <button class="btn btn-outline-primary">Primary</button> <button class="btn btn-outline-secondary">Secondary</button> <button class="btn btn-outline-success">Success</button> <button class="btn btn-outline-danger">Danger</button> <button class="btn btn-outline-warning">Warning</button> <button class="btn btn-outline-info">Info</button> <button class="btn btn-outline-dark">Dark</button> <button class="btn btn-outline-light text-dark">Light</button> <h2 class="mt-4">Button Sizes</h2> <button class="btn btn-primary btn-lg">Large</button> <button class="btn btn-primary">Normal</button> <button class="btn btn-primary btn-sm">Small</button> <button class="btn btn-primary btn-xs">XSmall</button> <h2 class="mt-4">Block-Level & Disabled / Active States</h2> <button class="btn btn-primary btn-block">Block-level Button</button> <button class="btn btn-primary active mt-2">Active Primary</button> <button class="btn btn-primary disabled mt-2">Disabled Primary</button> <h2 class="mt-4">Grouped Buttons</h2> <button class="btn btn-primary">A</button> <button class="btn btn-primary">B</button> <button class="btn btn-primary">C</button> <button class="btn btn-secondary">X</button> <button class="btn btn-secondary">Y</button> <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script> </body> </html>			
7	Write a Javascript program to accept a number and display the reverse of a given number.	[10]	CO2	L2

	<!DOCTYPE html>
	<html lang="en">
	<head>
	<meta charset="UTF-8">
	<title>Reverse a Number</title>
	</head>
	<body>
	<script>
	function reverseNumber(n) {
	let num = Math.abs(n);
	let reversed = 0;
	while (num > 0) {
	const digit = num % 10;
	reversed = reversed * 10 + digit;
	num = Math.floor(num / 10);
	}
	return reversed * Math.sign(n);
	}
	
	const input = prompt("Enter an integer:");
	const parsed = parseInt(input, 10);
	if (isNaN(parsed)) {
	alert("Please enter a valid integer.");
	} else {
	const rev = reverseNumber(parsed);
	alert(`Original number: ${parsed}\nReversed number: ${rev}`);
	console.log("Reversed number:", rev);
	}
	</script>
	</body>
	</html>			

8	What is Responsive Web Design? Explain the Bootstrap element: Tables. # Responsive Web Design (RWD) **Definition:** Responsive Web Design is an approach that makes web pages render well on a variety of devices and screen sizes by using flexible layouts, images, and media queries. ❖ **Responsive Design Principles** **1. Fluid Grid Layouts** 🔹 **Definition:**	[10]	CO3	L3

Use relative units (%, fr) instead of fixed units (px) to allow layout elements to scale proportionally.

- **Syntax:**

```
.container {
  width: 100%;
  max-width: 1200px;
  margin: 0 auto;
}
```

- **Example:**

```
<div class="container">
  <p>This layout will stretch based on screen size.</p>
</div>
```

2. Flexible Images and Media

- **Definition:**

Images and videos should resize themselves to fit within the container without overflowing.

- **Syntax:**

```
img {
  max-width: 100%;
  height: auto;
}
```

- **Example:**

```
<img src="example.jpg" alt="Responsive Image">
```

3. 🧩 Media Queries

- **Definition:**

CSS technique that applies styles based on device width, height, orientation, or resolution.

🔹 **Syntax:**

```
@media (max-width: 768px) {
  body {
    background-color: lightblue;
  }
}
```

🔹 **Example:**

```
@media (max-width: 600px) {
  .sidebar {
    display: none;
  }
}
```

4. 📱 Mobile-First Approach

🔹 **Definition:**

Start designing for smaller screens first, and then enhance the design for larger screens using min-width.

🔹 **Syntax:**

```
/* Default for mobile */
body {
        Font-size:12px;
    }
    @media only screen and (min-width: 500px) and (max-width: 800px) {
        body {
            font-size:30px;
        }
    }
```

- **Example:**

```
<div>Responsive text size</div>
```

Bootstrap Basic Table

A basic Bootstrap table has a light padding and only horizontal dividers.

The .table class adds basic styling to a table:

The .table-striped class adds zebra-stripes to a table:

The .table-bordered class adds borders on all sides of the table and cells:

The .table-hover class adds a hover effect (grey background color) on table rows:

The .table-condensed class makes a table more compact by cutting cell padding in half:

Contextual classes can be used to color table rows (<tr>) or table cells (<td>):

Class	Description
.active	**Applies the hover color to the table row or table cell**
.success	**Indicates a successful or positive action**
.info	**Indicates a neutral informative change or action**
.warning	**Indicates a warning that might need attention**
.danger	**Indicates a dangerous or potentially negative action**

The .table-responsive class creates a responsive table. The table will then scroll horizontally on small devices (under 768px). When viewing on anything larger than 768px wide, there is no difference:

```
<div class="table-responsive">
  <table class="table">
    <tr>
      <th>Firstname</th>
      <th>Lastname</th>
      <th>Email</th>
    </tr>
    <tr>
      <td>Default</td>
      <td>Defaultson</td>
      <td>def@somemail.com</td>
```

	</tr> <tr class="success"> <td>Success</td> <td>Doe</td> <td>john@example.com</td> </tr> </table> 			
9	What is a service in Angular JS? With a code snippet, explain $http service, and timeout service. **Services** Angular Service Service is a function or an object, which is used to provide with a specified action. In AngularJS, there are about 30 builtin services, such as $http, $location, $interval and $timeout. Types of Services in AngularJS AngularJS provides several built-in services and also allows you to create custom services. Here are some commonly used built-in services: 1. **$http**: For making AJAX requests. 2. **$location**: For handling URL manipulation. 3. **$timeout**: For delaying code execution. 4. **$interval**: For repeated execution at specified intervals. **Built-in AngularJS Services** **1.** $http **(For AJAX requests)** Used to communicate with a server. **Example:**	[10]	CO4	L4

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></sc
ript>
<body>

<div ng-app="myApp" ng-controller="myCtrl">

<p>Today's welcome message is:</p>

<h1>{{myWelcome}}</h1>

</div>

<p>The $http service requests a page on the server, and the response is set as
the value of the "myWelcome" variable.</p>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope, $http) {
  $http.get("welcome.htm").then(function (response) {
     $scope.myWelcome = response.data;
  });
});
</script>

</body>
</html>
```

2. $timeout (For Delayed Execution)

Executes a function after a delay.

Example:

```
<!DOCTYPE html>
<html>
```

	<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
	<body>
	
	 This header will change after two seconds:
	
	 {{myHeader}}
	
	 The $timeout service runs a function after a specified number of milliseconds.
	
	<script>
	var app = angular.module('myApp', []);
	app.controller('myCtrl', function($scope, $timeout) {
	  $scope.myHeader = "Hello World!";
	  $timeout(function () {
	  $scope.myHeader = "How are you today?";
	  }, 2000);
	});
	</script>
	
	</body>
	</html>			

10	Write an Angular JS Program to create a single page application. <!DOCTYPE html> <html lang="en" ng-app="myApp"> <head> <meta charset="UTF-8"> <title>AngularJS Single Page App</title> <!-- AngularJS and routing scripts --> <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script> <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular-route.min.js"></script> </head> <body> <!-- Navigation Links --> <nav> Home | About | Contact	[10]	CO4	L5

```
</nav>

<!-- View Container -->
<div ng-view></div>

<!-- AngularJS App & Routing -->
<script>
  angular.module('myApp', ['ngRoute'])
    .config(function($routeProvider) {
      $routeProvider
        .when('/', {
          templateUrl: 'home.html',
          controller: 'HomeController'
        })
        .when('/about', {
          templateUrl: 'about.html',
          controller: 'AboutController'
        })
        .when('/contact', {
          templateUrl: 'contact.html',
          controller: 'ContactController'
        })
        .otherwise({ redirectTo: '/' });
    })
    .controller('HomeController', function($scope) {
      $scope.message = 'Welcome to the Home Page!';
    })
    .controller('AboutController', function($scope) {
      $scope.message = 'Here is some info About us.';
    })
    .controller('ContactController', function($scope) {
      $scope.message = 'Get in touch via Contact page.';
    });
</script>

<!-- Inline Templates (so no separate files needed) -->
<script type="text/ng-template" id="home.html">
  <h2>Home</h2>
  <p>{{message}}</p>
</script>

<script type="text/ng-template" id="about.html">
  <h2>About</h2>
  <p>{{message}}</p>
</script>

<script type="text/ng-template" id="contact.html">
  <h2>Contact</h2>
  <p>{{message}}</p>
</script>
```

	</body> </html>			