CBCS SCHEME

USN | | | | | | | | | | |

MMC105

First Semester MCA Degree Examination, Dec.2025/Jan.2026
Web Technologies

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

		Module – 1	M	L	C
Q.1	a.	Explain the following terms (i) Image tag (ii) HTTP (iii) Audio & Video tags (iv) Progress tag.	10	L2	CO1
	b.	Define server. Explain any 2 types of servers.	06	L1	CO1
	c.	Explain MIME & its features in detail.	04	L2	CO1
		OR			
Q.2	a.	Discuss the basic structure of HTML5page with example.	05	L2	CO1
	b.	Define lists in HTML5. Describe ordered, unordered lists with suitable example.	10	L1	CO1
	c.	Explain in brief span and div tags with example for each.	05	L2	CO1
		Module – 2			
Q.3	a.	Define CSS. Describe the different levels of style sheet and their precedence.	08	L1	CO2
	b.	Explain any 2 simple selectors with example code.	06	L2	CO2
	c.	Describe any two different ways of array declaration in javascript with example.	06	L2	CO2
		OR			
Q.4	a.	Explain the concept of Box Model in CSS.	08	L2	CO2
	b.	Discuss any four Font proper tiesused in CSS with suitable example.	08	L2	CO2
	c.	Write a Javascript code that accepts 'n' as input from prompt and table of numbers from 1 to n and their squares should be displayed using alert.	04	L2	CO2
		Module – 3			
Q.5	a.	Explain the mechanism of event handling in JavaScript. Discuss any 2 mouse events with its event handlers.	10	L2	CO3
	b.	Discuss the application of Document Object Model (DOM) in JavaScript, and how the elements can be accessed in JavaScript illustrate with code snippet.	10	L2	CO3

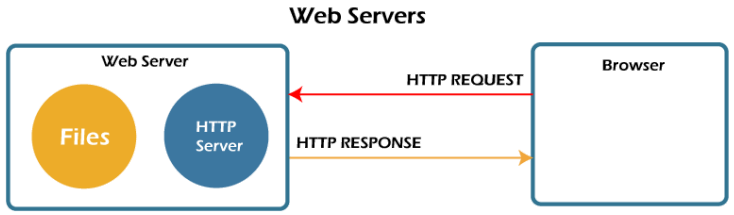
		OR			
Q.6	**a.**	Explain any 5 string methods in Javascript with suitable example.	**10**	**L2**	**CO3**
	b.	Discuss any two different form events with suitable example.	**10**	**L2**	**CO3**
		Module – 4			
Q.7	**a.**	Define directives in AngularJS. Explain the following AngularJS directives:(i) ng-app (ii) ng-init (iii) ng-bind (iv) ng-model.	**10**	**L2**	**CO4**
	b.	Explain in detail expressions in AngularJS with suitable code.	**06**	**L2**	**CO4**
	c.	Discuss in brief the usage of $scope with suitable example.	**04**	**L2**	**CO4**
		OR			
Q.8	**a.**	Discuss the Modules and Controllers with suitable AngularJS code.	**10**	**L2**	**CO4**
	b.	Discuss any 5 AngularJS filters with example.	**10**	**L2**	**CO4**
		Module – 5			
Q.9	**a.**	Explain how the services are handled in AngularJS. Discuss any 3 services of AngularJS with code.	**10**	**L2**	**CO4**
	b.	Mention any 3 validation directives. Write an AngularJS program to demonstrate client-side form validation.	**10**	**L2**	**CO4**
		OR			
Q.10	**a.**	Explain how the data in tables can be made iterative in AngularJS with example code.	**10**	**L2**	**CO4**
	b.	Discuss any 2 key events handled in AngularJS with a code snippet.	**10**	**L2**	**CO4**

* * * * *

CMR INSTITUTE OF TECHNOLOGY	USN	

VTU Theory Examination March 2026				
Sub:	Web Technologies		Code:	MMC105
	Answer Key	Marks	OBE	
			CO	RBT
	Module-1			
Q1(a)	Explain the following terms(i) image (ii) HTTP (iii) Audio & video tags (iv) Progress tag. **i) Image** An **image** in web development refers to a visual representation (such as photos, graphics, or icons) displayed on a webpage. In HTML, images are added using the <img> tag. It uses attributes like: • src (source of the image) • alt (alternative text) **Example:** <img src="photo.jpg" alt="Sample Image"> --- **(ii) HTTP (HyperText Transfer Protocol)** **HTTP** stands for **HyperText Transfer Protocol**. It is the communication protocol used to transfer data between a web browser (client) and a web server. • It works on a **request-response model** • Used to load web pages, images, videos, etc. • Common methods: GET, POST --- **(iii) Audio & Video Tags** HTML provides <audio> and <video> tags to embed media directly into webpages. • **<audio> tag**: Used to play sound files • **<video> tag**: Used to play video files They support controls like play, pause, volume, etc. **Example:**	10	CO1	L2

	<audio controls>
	<source src="audio.mp3" type="audio/mpeg">
	</audio>
	
	<video width="320" height="240" controls>
	<source src="movie.mp4" type="video/mp4">
	</video>
	
	(iv) Progress Tag
	
	The <progress> tag is used to represent the progress of a task (like loading or downloading).
	
	• It shows a progress bar
	• Uses attributes like:
	  ○ value (current progress)
	  ○ max (total value)
	
	Example:
	
	<progress value="50" max="100"></progress>			

Q1(b)	Define server. Explain any 2 types of server. What is a Web Server? A **server** is a computer system or software that provides services, resources, or data to other computers (called clients) over a network. It receives requests from clients and responds with the required information or service.  **Types of Servers (Any Two)** **1. Web Server** A **web server** is a server that stores, processes, and delivers web pages to users through web browsers using HTTP/HTTPS. **Functions:** • Hosts websites and web applications • Handles client requests and sends web pages • Supports protocols like HTTP and HTTPS **Examples:** • Apache HTTP Server • Nginx **2. File Server**	06	CO1	L1

	A **file server** is used to store and manage files so that multiple users can access them over a network. **Functions:** ● Centralized file storage ● Allows file sharing among users ● Provides security and access control **Examples:** ● Network Attached Storage (NAS) ● Windows File Server			
Q1(c)	Explain MIME & its features in detail. What is MIME? MIME stands for Multipurpose Internet Mail Extensions, a standard that extends the format of email to support text in different character sets, attachments such as images, audio, video, and application files, and other multimedia formats. Although originally developed for email, MIME types are now widely used in the context of the Web, where they describe the nature and format of a file or data. Key Features of MIME 1. Content Description: Specifies the type of data being sent. Examples: Text, image, video, audio, etc. 2. Encodings: Allows non-text data to be encoded in text-based formats for transmission (e.g., Base64). 3. Multipart Messages: Supports messages with multiple parts (e.g., an email with both text and an attachment). 4. Cross-Application Usage: Used by web browsers, servers, and email clients to handle and interpret file formats correctly. 7 MIME in HTTP In the context of the web, MIME types are used to indicate the type of content being transferred over HTTP. They are sent via the Content-Type header in HTTP responses. Examples of MIME Types: How MIME Works in HTTP 1. Server Response:	04	CO1	L2

	When a client requests a resource, the server includes the MIME type in the Content-Type header of its HTTP response. Example: HTTP/1.1 200 OK Content-Type: text/html; charset=UTF-8 2. Browser Behavior: Based on the MIME type, the browser decides how to process or display the content. For example: text/html: Render as a web page. application/pdf: Open a PDF viewer. image/jpeg: Display the image.			
	OR			
Q2(a)	Discuss the basic structure of HTML5 pages with examples. **Basic Structure of an HTML5 Page** An **HTML5 page** follows a standard structure that helps browsers understand and display content correctly. It consists of elements that define the document type, head section, and body section. --- **1. Document Type Declaration** The document starts with a declaration that tells the browser the version of HTML used. <!DOCTYPE html> --- **2. <html> Element** This is the root element of the HTML document. All other elements are placed inside it. <html lang="en"> </html> ● lang attribute specifies the language of the document. --- **3. <head> Section** The <head> contains metadata (information about the document), not visible on the webpage. **Includes:** ● <title>: Title of the page (shown in browser tab) ● <meta>: Character set, viewport settings ● <link>: External CSS	05	CO1	L2

- <script>: JavaScript files

Example:

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

---

4. <body> Section

The <body> contains all the visible content of the webpage such as text, images, links, etc.

```
<body>
 <h1>Welcome to HTML5</h1>
 <p>This is a sample page.</p>
</body>
```

---

5. Semantic Elements in HTML5

HTML5 introduced semantic tags that give meaning to the content structure:

- <header> – top section (logo, title)
- <nav> – navigation links
- <section> – grouped content
- <article> – independent content
- <aside> – sidebar content
- <footer> – bottom section

Complete Example of HTML5 Page

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>HTML5 Structure</title>
</head>
<body>

 <header>
   <h1>My Website</h1>
 </header>

 <nav>
   <a href="#">Home</a>
   <a href="#">About</a>
 </nav>

 <section>
   <article>
     <h2>Introduction</h2>
     <p>This is an example of HTML5 page structure.</p>
   </article>
```

	</section> <footer> © 2026 My Website </footer> </body> </html>			
Q2(b)	Define lists in HTML5. Describe ordered, unordered list with suitable example. In HTML5, lists are used to group related items. There are three main types of lists, each with different attributes and use cases: 1. Unordered List ( ) 2. Ordered List ( ) Each type of list can have various attributes, although the list element itself has only a few that directly impact its appearance or behavior. Let's explore these lists and attributes with examples. --- **1. Unordered List ( )** An unordered list is used when the order of items doesn't matter. By default, items in an unordered list are displayed with bullets. Attributes: ● type: The type attribute specifies the bullet style. It's applicable only to unordered lists. ○ type="disc" (default, filled circle) ○ type="circle" (hollow circle) ○ type="square" (square bullet) Example: <ul type="square"> - Item 1 - Item 2 - Item 3 ● Result: A list with square bullets. --- **2. Ordered List ( )** An ordered list is used when the sequence of items is important. It is automatically numbered by default, but the numbering style can be modified. Attributes:	10	CO1	L1

	1. type: Specifies the numbering style. a. type="1" (default, decimal numbering) b. type="A" (uppercase letters) c. type="a" (lowercase letters) d. type="I" (uppercase Roman numerals) e. type="i" (lowercase Roman numerals) 2. start: Specifies the starting value for the list. By default, it starts at 1. 3. reversed: If present, the list will display in reverse order. Example: <ol type="A" start="5" reversed> - Item 5 - Item 6 - Item 7 1. Result: An ordered list starting at 5, using uppercase letters, and in reverse order (7, 6, 5).			
Q2(c)	Explain in brief span and div tags with example for each. ● **span** The element in HTML5 is an inline element used to group or style parts of text or other inline elements. It does not have any semantic meaning by itself but is used as a container to apply styles or manipulations to a section of content without affecting the layout or structure of the document. Unlike block-level elements such as , the element does not create line breaks or affect the flow of the document. It is purely for styling or scripting purposes. Syntax of : Content goes here Common Use Cases for ● Styling Specific Text: You can apply CSS styles to a specific section of text without affecting the entire paragraph. ● JavaScript Interactions: You can use to target specific parts of a text for dynamic changes or interactions using JavaScript ● **div tags** The HTML tag defines sections in HTML documents, serving as containers styled with CSS or controlled with JavaScript. It's easily customized using class or id attributes and can contain various content.	05	CO1	L2

	Note: Browsers add line breaks before and after elements by default.
	Div tag Usage
	1. The div tag is the block-level tag.
	1. It is used for applying styles and layout structure and allows us to manipulate and position content through CSS.
	● It also divides content into logical sections, aiding in the readability and maintainability of the code.
	1. As we know, a Div tag is a block-level tag containing the entire width. Hence, every div tag will start from a new line and not the same line.
	html
	<!DOCTYPE html>
	<html>
	
	<head>
	<style>
	div {
	color: white;
	background-color: 009900;
	margin: 2px;
	font-size: 25px;
	}
	</style>
	
	</head>
	
	<body>
	 div tag
	 div tag
	 div tag
	 div tag
	
	</body>
	
	</html>			

	Module-2			
Q3(a)	Define CSS. Describe the different levels of style sheet and their precedence. Cascading Style Sheets (CSS) is used to format the layout of a webpage. With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more! **Levels of CSS**	08	CO2	L1

CSS can be added to HTML documents in 3 ways:

- ❖ Inline - by using the style attribute inside HTML elements
- ❖ Internal - by using a <style> element in the <head> section
- ❖ External - by using a <link> element to link to an external CSS file

Inline CSS

An inline CSS is used to apply a unique style to a single HTML element.

An inline CSS uses the style attribute of an HTML element.

The following example sets the text color of the <h1> element to blue, and the text color of the <p> element to red:

Example:-

```
<h1 style="color:blue;">A Blue Heading</h1>
<p style="color:red;">A red paragraph.</p>
```

Internal CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

The following example sets the text color of ALL the <h1> elements (on that page) to blue, and the text color of ALL the <p> elements to red. In addition, the page will be displayed with a "blue" background color:

Example:-

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: blue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
```

	This is a paragraph.
	
	</body>
	</html>
	External CSS
	
	An external style sheet is used to define the style for many HTML pages.
	
	To use an external style sheet, add a link to it in the <head> section of each HTML page:
	
	Example:-
	
	<!DOCTYPE html>
	<html>
	<head>
	<link rel="stylesheet" href="styles.css">
	</head>
	<body>
	
	 This is a heading
	 This is a paragraph.
	
	</body>
	</html>
	
	The external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.
	
	Here is what the "styles.css" file looks like:
	
	styles.css
	
	body {
	background-color: powder blue;
	}
	h1 {
	color: blue;
	}
	p {
	color: red;
	}			

Q3(b)	Explain any 2 simple selectors with example code. **1) CSS Element Selector**	06	CO2	L2

The element selector selects the HTML element by name.

Example:-

```
<!DOCTYPE html>
<html>
<head>
<style>
p{
   text-align: center;
   color: blue;
}
</style>
</head>
<body>
<p>This style will be applied on every paragraph.</p>
<p>Me too!</p>
<p>And me!</p>
<h1>hello</p>
</body>
</html>
```

Output:-

This style will be applied on every paragraph.

Me too!

And me!

2) CSS Id Selector

The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.

It is written with the hash character (#), followed by the id of the element.

Let?s take an example with the id "para1".

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
   text-align: center;
   color: blue;
}
</style>
</head>
<body>
```

	Hello Javatpoint.com This paragraph will not be affected. cmrit </body> </html> **Output:-** Hello Javatpoint.com			

Q3(c)	Describe any two different ways of array declaration in javascript with example.
	
	Introduction
	
	In JavaScript, an **array** is a collection of elements (numbers, strings, objects, etc.) stored under a single variable.
	Arrays can be declared in multiple ways, and two common methods are:
	
	--- 	
	
	1. Using Array Literal (Recommended Way)
	
	● This is the simplest and most common way to declare an array.
	● Syntax:
	
	let arrayName = [element1, element2, element3, ...];
	
	Example:
	
	let fruits = ["Apple", "Banana", "Mango", "Orange"];
	console.log(fruits[0]); // Output: Apple
	console.log(fruits.length); // Output: 4
	
	Explanation:
	
	● Elements are listed inside square brackets [ ].
	● fruits[0] accesses the first element.
	● .length gives the number of elements in the array.
	
	--- 	
	
	2. Using Array Constructor
	
	● JavaScript also allows arrays to be declared using the Array() constructor.
	● Syntax:
	
	let arrayName = new Array(element1, element2, element3, ...);
	
	Example:
	
	let numbers = new Array(10, 20, 30, 40);
	console.log(numbers[2]); // Output: 30
	console.log(numbers.length); // Output: 4
	
	Explanation:
		06	CO2	L2

	• new Array() creates a new array object. • This method is less commonly used but useful when creating arrays dynamically.			
	OR			
Q4(a)	Explain the concept of Box Model in CSS. **BOX Model** The **CSS Box Model** is a fundamental concept in web design that describes how elements are structured and spaced on a webpage. Every element in CSS is treated as a rectangular box, and the box model determines its size, spacing, and positioning. --- **Structure of the CSS Box Model** The box model consists of the following components (from inside out): • **Content** The actual content of the box, such as text, images, or other elements. **Size controlled by**: width, height. div { width: 200px; height: 100px; } • **Padding** The space between the content and the border. **Size controlled by**: padding property. Padding increases the overall size of the box. Property: padding-top padding-right padding-bottom padding-left padding: 10px; Example:- div {	08	CO2	L2

```
    padding: 10px; /* Adds 10px padding on all sides */
    padding-left: 20px; /* Adds 20px padding to the left only */
}
```

- **Border**

  A line surrounding the padding and content.

  Size controlled by: border-width, border-style, border-color.

  Property:-

  border-width
  border-style (e.g., solid, dashed, dotted, none)
  border-color

  Example:-

```
border: 2px solid black;
div
 {
    border: 2px solid black; /* 2px solid border */
    border-radius: 10px; /* Rounded corners */
}
```

- **Margin**

  The space between the element and its neighboring elements.

  Size controlled by: margin property.

  Property:-
  margin-top

  margin-right

  margin-bottom

  margin-left
  margin: 20px;

  Example:

```
div {

margin: 15px; /* Adds 15px margin on all sides */
```

	margin-top: 20px; /* Adds 20px margin to the top only */
	
	}
	
	Box Sizing
	
	● The box-sizing property defines whether the width and height include the padding and border or not.
	
	div {
	
	box-sizing: border-box; /* Includes padding and border in width/height */
	
	}
	
	Example:-
	
	div {
	
	width: 200px;
	
	height: 100px;
	
	padding: 10px;
	
	border: 5px solid black;
	
	margin: 20px;
	
	box-sizing: border-box;
	
	}			

Q4(b)	Discuss any four Font proper tiesused in CSS with suitable example. CSS (Cascading Style Sheets) is used to style HTML elements. **Font properties in CSS** are used to control the appearance of text on a webpage. --- **Any Four Font Properties in CSS** **1. font-family** This property specifies the type of font used for text. **Example:** p {	08	CO2	L2

font-family: Arial, sans-serif;

}

---

2. font-size

This property defines the size of the text.

Example:

h1 {

font-size: 24px;

}

---

3. font-weight

This property controls the thickness (boldness) of text.

Values: normal, bold, bolder, lighter, or numeric values (100–900)

Example:

p {

font-weight: bold;

}

---

4. font-style

This property is used to make text italic or normal.

Values: normal, italic, oblique

Example:

p {

font-style: italic;

}

Combined Example

<!DOCTYPE html>

<html>

<head>

<style>

	p {
	font-family: Verdana, sans-serif;
	font-size: 18px;
	font-weight: bold;
	font-style: italic;
	}
	</style>
	</head>
	<body>
	 This is styled text using CSS font properties.
	</body>
	</html>			

Q4(c)	Write a javascript code that accepts 'n' as input from prompt and table of numbers from 1 to n and their squared should be displayed using alert.
	<script>
	// Accept input from user
	let n = prompt("Enter a number:");
	
	// Convert input to number
	n = parseInt(n);
	
	let result = "";
	
	// Generate numbers and their squares
	for (let i = 1; i <= n; i++) {
	result += i + " squared = " + (i * i) + "\n";
	}
	
	// Display result using alert
	alert(result);
	</script>
		04	CO2	L2

	Module-3			
Q5(a)	Explain the mechanism of event handling in javascript. Discuss any 2 mouse events with its event handlers. **1. Mechanism of Event Handling in JavaScript** In JavaScript, **events** are actions or occurrences that happen in the browser, such as user interactions (click, mouseover, keypress) or browser-generated events	10	L2	CO3

(load, resize).

Steps in event handling:

1. **Event Generation:**
   - The browser detects an event, like clicking a button or moving the mouse.
2. **Event Registration:**
   - You attach an **event handler** (a function) to the element that should respond to the event.
3. **Event Propagation:**
   - Events travel through the DOM tree in two phases:
     - **Capturing phase:** From the root to the target element
     - **Bubbling phase:** From the target element back up to the root
4. **Event Handling:**
   - The registered **event handler function executes** when the event occurs.

Methods to attach event handlers:

- **Inline event handler:**

  ```
  <button onclick="myFunction()">Click Me</button>
  ```

- **Using DOM property:**

  ```
  element.onclick = myFunction;
  ```

- **Using addEventListener (preferred):**

  ```
  element.addEventListener("click", myFunction);
  ```

---

2. Two Mouse Events in JavaScript

(i) click Event

- Triggered when the user **clicks** on an element.
- Commonly used for buttons, links, or images.

Example:

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

<button onclick="showMessage()">Click Me</button>

<script>
function showMessage() {
  alert("Button clicked!");
}
</script>

</body>
</html>
```

	Explanation:
	● onclick attaches the event handler to the button.
	● showMessage() executes when the button is clicked.
	
	--- 	
	
	(ii) mouseover Event
	● Triggered when the mouse **pointer moves over an element**.
	Example:
	`<!DOCTYPE html>`
	`<html>`
	`<body>`
	`<p onmouseover="hoverText(this)">Hover over me!</p>`
	`<script>`
	`function hoverText(element) {`
	`element.style.color = "red";`
	`element.innerHTML = "Mouse is over the text!";`
	`}`
	`</script>`
	`</body>`
	`</html>`
	Explanation:
	● onmouseover attaches the event handler to the paragraph.
	● When the mouse hovers over the paragraph, the text color and content change dynamically.			

Q5(b)	Discuss the application of document Object Model(DOM) in javascript, and how the elements can be accessed in javascript illustrated with code snippet. **Introduction to DOM** The **Document Object Model (DOM)** is a programming interface for HTML and XML documents. It represents the webpage as a **tree structure** where each node is an object representing a part of the document. JavaScript uses the DOM to **access, modify, add, or delete elements** dynamically. --- **Applications of DOM in JavaScript** The DOM allows developers to: 1. **Access HTML elements** 2. **Modify content and attributes**	10	L2	CO3

3. **Change styles (CSS)**
4. **Handle events (click, submit, etc.)**
5. **Create and delete elements dynamically**

Accessing Elements in JavaScript

JavaScript provides several methods to access elements:

1. By ID

```
<p id="demo">Hello</p>


<script>
 let element = document.getElementById("demo");
 element.innerHTML = "Hello World!";
</script>
```

---

2. By Class Name

```
<p class="text">Item 1</p>
<p class="text">Item 2</p>


<script>
 let elements = document.getElementsByClassName("text");
 elements[0].innerHTML = "Updated Item";
</script>
```

---

3. By Tag Name

```
<p>Paragraph 1</p>


<script>
 let para = document.getElementsByTagName("p");
 para[0].style.color = "blue";
</script>
```

---

	4. Using querySelector
	 Sample Text
	
	<script>
	let el = document.querySelector(".info");
	el.style.fontSize = "20px";
	</script>
	--- 	
	Example: DOM Manipulation
	<!DOCTYPE html>
	<html>
	<body>
	
	<h2 id="title">Original Title</h2>
	<button onclick="changeText()">Click Me</button>
	
	<script>
	function changeText() {
	document.getElementById("title").innerHTML = "Updated Title";
	}
	</script>
	
	</body>
	</html>			

	OR			
Q6(a)	Explain any 5 string methods in javascript with suitable example. **Introduction** In JavaScript, **string methods** are used to manipulate and work with text. These methods help in performing operations like searching, replacing, and modifying strings.	10	L2	CO3

	Any Five String Methods in JavaScript
	
	1. toUpperCase()
	
	Converts a string to uppercase letters.
	
	Example:
	
	`let text = "hello";`
	`let result = text.toUpperCase();`
	`console.log(result);  // HELLO`
	
	2. toLowerCase()
	
	Converts a string to lowercase letters.
	
	Example:
	
	`let text = "HELLO";`
	`let result = text.toLowerCase();`
	`console.log(result);  // hello`
	
	3. charAt()
	
	Returns the character at a specified index.
	
	Example:
	
	`let text = "JavaScript";`
	`let ch = text.charAt(4);`
	`console.log(ch);  // S`
	
	4. indexOf()
	
	Returns the position of the first occurrence of a specified value.
	
	Example:
	
	`let text = "Hello World";`
	`let pos = text.indexOf("World");`
	`console.log(pos);  // 6`
	
	5. replace()
	
	Replaces a specified value with another value in a string.
	
	Example:
	
	`let text = "I like Java";`
	`let newText = text.replace("Java", "JavaScript");`
	`console.log(newText);  // I like JavaScript`			

Q6(b)	Discuss any two different form events with suitable example.	10	L2	CO3

Introduction

In JavaScript, **form events** are actions that occur when a user interacts with a form (like entering data, submitting, or leaving a field). These events help in validating input and improving user interaction.

Any Two Form Events

1. onsubmit Event

The **onsubmit** event occurs when a form is submitted. It is commonly used for **form validation** before sending data to the server.

Example:

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

<form onsubmit="return validateForm()">
 Name: <input type="text" id="name">
 <input type="submit" value="Submit">
</form>

<script>
function validateForm() {
 let x = document.getElementById("name").value;
 if (x == "") {
   alert("Name must be filled out");
   return false; // prevents submission
 }
 return true;
}
</script>

</body>
</html>
```

---

2. onfocus Event

The **onfocus** event occurs when an input field gets focus (when the user clicks or tabs into it).

Example:

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

Enter your name: <input type="text" onfocus="changeColor(this)">

<script>
function changeColor(element) {
 element.style.background = "yellow";
}
</script>
```

	</body> </html>			
	Module-4			
Q7(a)	Define directives in AngularJS. Explain the following AngularJS directives(i) ng-app (ii) ng-init (iii) ng-bind (iv) ng-model. **Definition of Directives in AngularJS** **Directives in AngularJS** are special HTML attributes (or elements) that extend the functionality of HTML. They are used to bind application data to HTML elements and control the behavior of the DOM. Directives usually start with the prefix **ng-**. --- **Explanation of AngularJS Directives** **(i) ng-app** ● The **ng-app** directive initializes an AngularJS application. ● It defines the root element of the application. **Example:** My first AngularJS app --- **(ii) ng-init** ● The **ng-init** directive is used to initialize variables in an AngularJS application. **Example:** Name: {{name}} --- **(iii) ng-bind** ● The **ng-bind** directive binds application data to HTML elements. ● It updates the content automatically when data changes. **Example:** --- **(iv) ng-model** ● The **ng-model** directive binds input elements to application data	10	L2	CO4

	(two-way data binding). ● It keeps the model and view synchronized. **Example:** Enter name: <input type="text" ng-model="name"> Your name is: {{name}} 			
Q7(b)	Explain in detail expressions in AngularJS with suitable code. **introduction to Expressions in AngularJS** **Expressions in AngularJS** are used to **bind data from the model to the view**. They are evaluated by AngularJS and the result is displayed in HTML. Expressions are similar to JavaScript expressions but are **safer** and **simpler** to use in templates. **Syntax:** {{ expression }} --- **Key Features of AngularJS Expressions** 1. Evaluated **within double curly braces {{ }}**. 2. Can contain variables, operators, and function calls. 3. Automatically update when the model changes (**two-way binding**). 4. Do not use document.write() or alert() like normal JavaScript. --- **Examples of AngularJS Expressions** **1. Displaying a Variable** Hello, {{name}}! **Output:** Hello, John! --- **2. Performing Calculations** Sum: {{a + b}} **Output:** Sum: 15 --- **3. Using Functions in Expressions** Double of a: {{a * 2}} Square of b: {{b * b}}	06	L2	CO4

	  Output: Double of a: 10 Square of b: 100 --- **4. Two-Way Binding with Input** Enter your name: <input type="text" ng-model="name"> Hello, {{name}}! ● When the user types, {{name}} updates automatically. ● This demonstrates **real-time two-way binding**.			
Q7(c)	Discuss in brief the usage of $scope with suitable example. **Introduction to $scope** In AngularJS, **$scope** is an object that **connects the controller and the view**. ● It acts as a **context for model data**. ● Variables and functions defined on $scope can be accessed in the HTML view. ● Changes in $scope are automatically reflected in the view (**two-way data binding**). --- **Usage of $scope** 1. **Defining variables in the controller** 2. **Defining functions to perform actions** 3. **Sharing data between view and controller** --- **Example** <!DOCTYPE html> <html ng-app="myApp"> <head> <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script> </head> <body> Welcome, {{name}}! <input type="text" ng-model="name"> <button ng-click="greet()">Greet</button> <script> // Define AngularJS application	04	L2	CO4

	var app = angular.module("myApp", []);
	
	// Define controller
	app.controller("myCtrl", function($scope) {
	$scope.name = "John"; // variable in $scope
	
	$scope.greet = function() { // function in $scope
	alert("Hello, " + $scope.name + "!");
	}
	});
	</script>
	
	</body>
	</html>
	
	Explanation:
	
	● $scope.name is displayed in <p> and input box.
	● Changing the input updates the variable automatically (**two-way binding**).
	● $scope.greet() function can be triggered with the button click.			

	OR			

Q8(a)	Discuss the Modules and Controllers with suitable AngularJS code.
	
	1. Modules in AngularJS
	
	● A **module** is a container for different parts of an AngularJS application, such as controllers, services, filters, directives, etc.
	● Every AngularJS application must have **at least one module**.
	● Modules help in organizing the code and avoiding global variables.
	
	Syntax:
	
	var app = angular.module('moduleName', []);
	
	● 'moduleName' → name of the module
	● [] → array of dependencies (can be empty if no dependency)
	
	--- 	
	
	2. Controllers in AngularJS
	
	● A **controller** is a JavaScript function that **manages data and behavior** of a part of the application.
	● It is defined inside a module.
	● $scope is used to pass data and functions from the controller to the view.
	
	Syntax:
	
	app.controller('controllerName', function($scope) {
	// Define variables and functions on $scope
	});
	
	--- 	
	
	Example: Module and Controller
	<!DOCTYPE html>
		10	L2	CO4

```
<html ng-app="myApp">
<head>
                                                                    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></scri
pt>
</head>
<body>

<div ng-controller="myCtrl">
 <h2>{{greeting}}</h2>
 <p>Enter your name: <input type="text" ng-model="name"></p>
 <p>Hello, {{name}}!</p>
</div>

<script>
 // Define AngularJS module
 var app = angular.module('myApp', []);

 // Define controller within the module
 app.controller('myCtrl', function($scope) {
    $scope.greeting = "Welcome to AngularJS!";
    $scope.name = "John";  // Initial value
 });
</script>

</body>
</html>
```

Q8(b) Discuss any 5 AngularJS filters with example. | 10 | L2 | CO4

Introduction to AngularJS Filters

Filters in AngularJS are used to **format data before displaying it in the view**. They are applied in expressions using the pipe (|) symbol.
Filters can format text, numbers, dates, and more.

Syntax:

```
{{ expression | filterName }}
```

Five Common AngularJS Filters with Examples

1. currency

Formats a number as a currency value.

Example:

```
<div ng-app="" ng-init="price=2500.5">
 Price: {{price | currency}}
</div>
```

Output:

Price: $2,500.50

---

2. number

Formats a number with specified decimal places.

Example:

```
<div ng-app="" ng-init="num=1234.567">
 Number: {{num | number:2}}
</div>
```

Output:

Number: 1,234.57

---

3. uppercase

Converts text to uppercase letters.

Example:

```
<div ng-app="" ng-init="text='hello world'">
 Uppercase: {{text | uppercase}}
</div>
```

Output:

Uppercase: HELLO WORLD

---

4. lowercase

Converts text to lowercase letters.

Example:

```
<div ng-app="" ng-init="text='HELLO WORLD'">
 Lowercase: {{text | lowercase}}
</div>
```

Output:

Lowercase: hello world

---

5. date

Formats a date value according to a specified format.

Example:

```
<div ng-app="" ng-init="today = Date()">
 Today: {{today | date:'fullDate'}}
</div>
```

Output:

Today: Saturday, March 28, 2026

	Module-5			
Q9(a)	Explain how the services are handled in AngularJS. Discuss any 3 services of	10	L2	CO4

AngularJS with code.

Introduction to AngularJS Services

Services in AngularJS are **singleton objects** that provide **reusable functionality** across the application.

- Services are used to **share data, logic, or functions** between controllers and components.
- AngularJS provides built-in services and allows creating **custom services**.
- Services are typically **injected into controllers or other services** using **dependency injection**.

How Services are Handled

1. AngularJS creates a **singleton instance** of each service.
2. Services are **injected** into controllers or components where needed.
3. They **encapsulate logic**, so controllers remain lightweight.
4. Commonly used services include $http, $scope, $timeout, $interval, etc.

Syntax of injecting a service:

```
app.controller('myCtrl', function($scope, $http) {
  // $http service used here
});
```

Three Common AngularJS Services with Examples

1. $http Service

- $http is used to make **HTTP requests** (GET, POST) to fetch or send data to a server.

Example:

```
<!DOCTYPE html>
<html ng-app="myApp">
<head>
                                                    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></scri
pt>
</head>
<body>

<div ng-controller="myCtrl">
 <ul>
  <li ng-repeat="user in users">{{user.name}}</li>
 </ul>
</div>

<script>
var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $http) {
  $http.get("https://jsonplaceholder.typicode.com/users")
      .then(function(response){
```

	$scope.users = response.data;
	});
	});
	</script>
	
	</body>
	</html>
	--- 	
	2. $timeout Service
	● $timeout is used to **execute a function after a specified delay**, similar to setTimeout in JavaScript.
	Example:
	
	 {{message}}
	
	<script>
	var app = angular.module('myApp', []);
	
	app.controller('myCtrl', function($scope, $timeout) {
	$scope.message = "Wait for it...";
	$timeout(function() {
	$scope.message = "Hello after 3 seconds!";
	}, 3000);
	});
	</script>
	--- 	
	3. $interval Service
	● $interval is used to **execute a function repeatedly at specified intervals**, similar to setInterval in JavaScript.
	Example:
	
	 Counter: {{counter}}
	
	<script>
	var app = angular.module('myApp', []);
	
	app.controller('myCtrl', function($scope, $interval) {
	$scope.counter = 0;
	$interval(function() {
	$scope.counter++;
	}, 1000); // increments every 1 second
	});
	</script>			

Q9(b)	Mention any 3 validation directives. Write an AngularJS program to demonstrate client-side form validation. **1. Three Validation Directives in AngularJS**	10	L2	CO4

AngularJS provides built-in **validation directives** to validate form inputs:

1. **ng-required** – Makes an input field mandatory.
2. **ng-minlength** – Sets the minimum number of characters allowed.
3. **ng-pattern** – Validates input against a regular expression pattern.

---

2. AngularJS Program for Client-Side Form Validation

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

<h2>AngularJS Client-Side Form Validation</h2>

<form name="myForm" novalidate>

 <!-- Name field (required) -->
 Name:
       <input type="text" name="username" ng-model="username"
ng-required="true">
      <span class="error" ng-show="myForm.username.$touched &&
myForm.username.$error.required">
   Name is required.
 </span>
 <br><br>

 <!-- Email field (pattern validation) -->
 Email:
 <input type="email" name="email" ng-model="email" ng-required="true">
       <span class="error" ng-show="myForm.email.$touched &&
myForm.email.$error.required">
   Email is required.
 </span>
       <span class="error" ng-show="myForm.email.$touched &&
myForm.email.$error.email">
   Invalid email address.
 </span>
 <br><br>

 <!-- Password field (minlength) -->
 Password:
     <input type="password" name="password" ng-model="password"
ng-required="true" ng-minlength="6">
      <span class="error" ng-show="myForm.password.$touched &&
myForm.password.$error.required">
   Password is required.
 </span>
      <span class="error" ng-show="myForm.password.$touched &&
myForm.password.$error.minlength">
```

	Password must be at least 6 characters. <button type="submit" ng-disabled="myForm.$invalid">Submit</button> </form> </body> </html>			
	OR			
Q10(a)	Explain how the data in tables can be made iterative in AngularJS with example code. In AngularJS, tables can display **dynamic or iterative data** using the **ng-repeat directive**. ● ng-repeat loops through an array or collection and creates **table rows for each item**. ● This allows data to be displayed **dynamically without manually writing each row**. --- **Steps to Make Table Data Iterative** 1. Define a **module and controller**. 2. Store data in an **array of objects** inside $scope. 3. Use **ng-repeat** on <tr> to generate table rows dynamically. 4. Bind individual fields using {{ }} expressions. --- **Example Code** <!DOCTYPE html> <html ng-app="myApp"> <head> <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script> </head> <body> Student Details <table border="1" cellpadding="5"> <tr> <th>Roll No</th> <th>Name</th> <th>Grade</th> </tr> <tr ng-repeat="student in students"> <td>{{student.roll}}</td> <td>{{student.name}}</td> <td>{{student.grade}}</td> </tr>	10	L2	CO4

	</table>
	
	<script>
	var app = angular.module('myApp', []);
	app.controller('myCtrl', function($scope) {
	$scope.students = [
	{roll: 1, name: 'Alice', grade: 'A'},
	{roll: 2, name: 'Bob', grade: 'B'},
	{roll: 3, name: 'Charlie', grade: 'A'},
	{roll: 4, name: 'David', grade: 'C'}
	];
	});
	</script>
	
	</body>
	</html>			

Q10(b)	Discuss any 2 key events handled in AngularJS with a code snippet. **Introduction to Events in AngularJS** AngularJS allows handling **user interactions** such as clicks, mouse movements, and form submissions using **event directives**. ● These events are used to **trigger functions** defined in the controller or $scope. ● Event directives in AngularJS start with ng-, e.g., ng-click, ng-dblclick, ng-mouseover. --- **Two Key AngularJS Events** **1. ng-click** ● **Description:** Executes a function when an element is clicked. ● **Use:** Commonly used for buttons, links, or interactive elements. **Example:** <!DOCTYPE html> <html ng-app=""> <head> </head> <body> Button clicked {{count}} times. <button ng-click="count = count + 1">Click Me</button> </body> </html> **Explanation:** ● ng-click="count = count + 1" increments the count variable on each click.	10	L2	CO4

- {{count}} automatically updates the view.

---

2. ng-dblclick

- **Description:** Executes a function when an element is double-clicked.
- **Use:** Used for special actions triggered by double-click.

Example:

```
<!DOCTYPE html>
<html ng-app="">
<head>
</head>
<body>

<div ng-init="message='Double-click the box'">
 <p>{{message}}</p>
 <div ng-dblclick="message='You double-clicked!'"
                    style="width:150px;    height:50px;    background-color:lightblue;
text-align:center; line-height:50px; cursor:pointer;">
   Double-click me
 </div>
</div>

</body>
</html>
```

Explanation:

- ng-dblclick changes the $scope variable message when the div is double-clicked.
- The change is **immediately reflected in the view** using AngularJS binding.