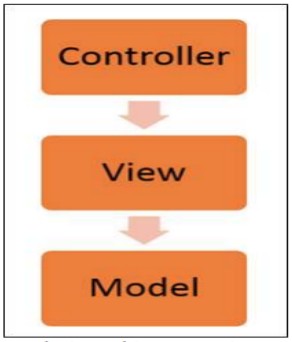
CMR INSTITUTE OF TECHNOLOGY	USN									

Internal Assessment Test –II March 2025				
Sub:	Web Technologies	Code:	MMC105	
Answer Key		Marks	OBE	
			CO	RBT
1	**What is AngularJs? Explain AngularJs Architecture with a Neat Diagram.** **What is AngularJS?** ● AngularJS is an open source Model-View-Controller framework which is similar to the JavaScript framework. ● AngularJS framework is used for developing mostly Single Page applications. ● This framework has been developed by a group of developers from Google itself. Because of the sheer support of Google and ideas from a wide community forum, the framework is always kept up to date. Also, it always incorporates the latest development trends in the market. AngularJS Architecture • The Controller represents the layer that has the business logic. User events trigger the functions which are stored inside your controller. The user events are part of the controller. • Views are used to represent the presentation layer which is provided to the end users • Models are used to represent your data. The data in your model can be as simple as just having primitive declarations. For example, if you are maintaining a student application, your data model could just have a student id and a name. Or it can also be complex by having a structured data model. If you are maintaining a car ownership application, you can have structures to define the vehicle itself in terms of its engine capacity, seating capacity, etc.	10	CO4	L3



Angularjs Architecture Diagram

AngularJS Advantages

- Since it's an open source framework, you can expect the number of errors or issues to be minimal.
- Two-way binding – Angular.js keeps the data and presentation layer in sync. Now you don't need to write additional JavaScript code to keep the data in your HTML code and your data later in sync. Angular.js will automatically do this for you. You just need to specify which control is bound to which part of your model
- Routing – Angular can take care of routing which means moving from one view to another. This is the key fundamental of single page applications; wherein you can move to different functionalities in your web application based on user interaction but still stay on the same page.
- Angular supports testing, both Unit Testing, and Integration Testing.
- It extends HTML by providing its own elements called directives. At a high level, directives are markers on a DOM element (such as an attribute, element name, and comment or CSS class) that tell AngularJS's HTML compiler to attach a specified behavior to that DOM element. These directives help in extending the functionality of existing HTML elements to give more power to your web application.

AngularJS MVC Architecture

MVC stands for Model View Controller. It is a software design pattern for developing web applications. It is very popular because it isolates the application logic from the user interface layer and supports separation of concerns.

The MVC pattern is made up of the following three parts:

1. Model: It is responsible for managing application data. It responds to the requests from view and to the instructions from the controller to update itself.
2. View: It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.
3. Controller: It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects.

	The controller receives input, validates it, and then performs business operations that modify the state of the data model.			
2	**What is a Service in AngularJS? with a code Snippet, explain $http service, and time out service.** <u>Services</u> 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. **1. $http (For AJAX requests)** Used to communicate with a server. Example:	10	CO3	L3

```
<!DOCTYPE html>

<html>

<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<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>

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

<p>This header will change after two seconds:</p>

<h1>{{myHeader}}</h1>

</div>

<p>The $timeout service runs a function after a specified number of milliseconds.</p>

<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>
```

3	**Write a java script 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">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Reverse Number</title>
</head>
<body>
   <h2>Reverse a Number</h2>
   <label for="number">Enter a number:</label>
   <input type="number" id="number">
   <button onclick="reverseInputNumber()">Reverse</button>
   <p id="result"></p>
```

```
<script>
    // Function to reverse a number
    function reverseNumber(num) {
        let reversed = 0;
        while (num > 0) {
            let digit = num % 10;
            reversed = reversed * 10 + digit;
            num = Math.floor(num / 10);
        }
        return reversed;
    }

    function reverseInputNumber() {
        let num = parseInt(document.getElementById("number").value, 10);
        if (!isNaN(num)) {
            let reversedNum = reverseNumber(num);
                document.getElementById("result").innerText = "Reversed Number: " +
reversedNum;
        } else {
            document.getElementById("result").innerText = "Invalid input. Please enter a
valid number.";
        }
    }
</script>
</body>
</html>
```

4	**Explain the following directives with an example.** **1) ng-app 2) ng-model 3) ng-init 4) ng-repeat 5) ng-bind** 1)ng-app ● It initializes an AngularJS application. ● It is usually placed in the <html> or <body> tag. <!DOCTYPE html> <html lang="en" ng-app="myApp">… <body> AngularJS Application </body> </html> 2)ng-model ● It binds the value of an HTML input element to a variable in the AngularJS scope.	10	CO4	L3

- It is commonly used in forms and input fields.

```
<div ng-app="">
   <input type="text" ng-model="name">
   <p>Hello, {{ name }}</p>
</div>
```

3)ng-init

- It initializes variables in the AngularJS scope.

```
<div ng-app="" ng-init="count=10">
   <p>Count: {{ count }}</p>
</div>
```

4)ng-repeat

- It repeats HTML elements based on an array or collection.

```
<div ng-app="" ng-init="students=['John', 'Alice', 'Bob']">
   <ul>
      <li ng-repeat="student in students">{{ student }}</li>
   </ul>
</div>
```

5) ng-bind

- It binds an expression to HTML without using curly braces ({{ }}).
- Useful when avoiding the flicker effect ({{ expression }} being displayed before AngularJS loads).

```
<div ng-app="" ng-init="message='Welcome to AngularJS'">
   <p ng-bind="message"></p>
</div>
```

5	**What is Filter? Explain Uppercase, lowercase, orderBy and Currency with an Example**. <u>Filters</u> **What is filter?** Filter is used to format the value of data. The pipe sign (|) indicates that filter is used. The proper syntax of filter looks like this: `Value | filter` Let`s try to understand the filers one by one.	10	CO4	L3

Uppercase filter

Value | uppercase

The uppercase filter changes the text to upper case. Suppose a user writes a text in lower case (e.g. ray) or title case (e.g. Ray) or in mixed case (e.g. rAy or RaY or rAY etc.), and you want the upper case result, then you will have to use upper case filter.

Example 3.1

```
<!DOCTYPE html>
<html >
<head>
   <title>AngularJSfor beginners</title> <script src="js\angular.min.js">
</script> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
/> </head>
<body>
<h3>Using Upper Case Filter</h3> <div ng-app=""  ng-init="Username=
'ray' "> <p>User Name: <input type="text" ng-model = "Username"></p>
<p style="color:red" ng-bind="Username | uppercase"></p> </div>
</body>
</html>
```

Output:

Using Upper Case Filter

User Name: ray

RAY

Explanation:

"Username | uppercase" changes the value of "Username" to uppercase.

In the above example, I set the default value (**ray**) in lower case, but the result becomes upper case (RAY).

Lowercase filter

Value | lowercase

The lowercase filter changes the text to lower case. Suppose a user writes a text in upper case (e.g. RAY YAO) or title case (e.g. Ray Yao) or in mixed case (e.g. rAy or RaY or rAY etc.), and you want the lower case result, then you will have to use lower case filter.

Example 3.2

```
<html >
<head>
   <title>AngularJSfor beginners</title> <script src="js\angular.min.js">
</script> </head>
<body>
<h3>Using Lower Case Filter</h3> <div ng-app=""  ng-init="Username=
'Ray YAO' "> <p>User Name: <input type="text" ng-model="Username">
</p> <p style="color:red" ng-bind="Username | lowercase"></p> </div>
</body>
</html>
```

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

Output:

Using Lower Case Filter

User Name: Ray YAO

ray yao

Explanation:

"Username | lowercase": changes the value of "Username" to lowercase.

In the above example, when I enter text (**Ray YAO**) in upper case, but the result become lower case (ray yao).

OrderBy filter

OrderBy filer is used to display values in ascending order or descending order. The syntax of “orderBy” looks like this:

```
Value | orderBy: ’value’    //for ascending order
Value | orderBy: ’-value’   //for descending order
```

Let`s take an example for better understanding.

Example 3.3

```
<!DOCTYPE html>
<html >
<head>
   <title>AngularJSfor beginners</title> <script src="js\angular.min.js">
</script> </head>
<body>
<h1>Using OrderBy filter</h1> <div ng-app="" ng-init="StudentsResult=
[{name: 'Tienq', marks:81},            {name: 'Svbrf', marks:70},
{name: 'Yaito', marks:90},             {name: 'Pewfn', marks:63}, {name:
'Riet', marks:98}]"> <table border="1" > <tr>
<th>Student Name</th> <th>Mathematics’ Result</th> </tr>
<tr ng-repeat="x in StudentsResult | orderBy:'-marks' "> <td ng-
bind="x.name "></td> <td ng-bind="x.marks "></td> </tr>
</table>
</div>
</body>
</html>
```

Output:

Using OrderBy filter

Student Name	Mathematics' Result
Riet	98
Yaito	90
Tienq	81
Svbrf	70
Pewfn	63

Explanation:

StudentsResult | orderBy:'-marks' displays the values of StudentsResult in descending order.

You can see that the highest mark is on top and the lowest mark is on bottom by using (value | orderBy:**'-marks'**).

If you want reverser the order, you can remove the **"-"**sign".

Currency filter

Value | currency

The currency filter is used to display the result in currency format.

Example 3.5

```
<!DOCTYPE html>
<html >
<head>
   <title>AngularJSfor beginners</title> <script src="js\angular.min.js">
</script> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8
/> </head>
<body>
<h1>Using Currency filter</h1> <div ng-app="" ng-init =
"Employees_Monthly_Salary=[{name: 'Jay', salary:8100}, {name: 'Sdwt',
salary:7000},            {name: 'Hao', salary:9000}, {name: 'Luoe',
salary:6300}, {name: 'Fin', salary:9800}]"> <table border="1" > <tr>
<th>Employee Name</th> <th>Employee Salary</th> </tr>
<tr ng-repeat="x in Employees_Monthly_Salary  "> <td ng-bind="x.name
"></td> <td ng-bind="x.salary | currency "></td> </tr>
</table>
</div>
</body>
</html>
```

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

Output:

Using Currency filter

Employee Name	Employee Salary
Jay	$8,100.00
Sdwt	$7,000.00
Hao	$9,000.00
Luoe	$6,300.00
Fin	$9,800.00

Explanation:

"x.salary | currency " converts the salary to currency format.

In the above example, there are two columns in the table, the first column is Employee Name and the second is Employee Salary. The salary column displays the salary in currency format.

Array filter

```
Array | filter:input
```

“Array | filter:input” can filter the array elements based on the user input.

Example 3.6

```
<!DOCTYPE html>
<html ng-app="">
<head>
<script src="js\angular.min.js"></script> <meta charset="utf-8"> </head>
<body>
<div ng-init="students =        // define an array “students”
[{name:'Andy', age:'19'},
{name:'Rose', age:'18'},
{name:'Jony', age:'17'},
{name:'Judy', age:'16'},
{name:'Tomy', age:'15'},
{name:'Lily', age:'14'}]"> </div>


<table>
  <tr><th>Name</th><th>Age</th></tr> <tr ng-repeat="person in students |
filter:myList" > // filter the array “students” according to the input value
    <td>{{person.name}}</td> <td>{{person.age}}</td> </tr>
</table>
<br><br>
<label>Please input one of the above name or age <br><br>
<input ng-model="myList"> </label>    // user input </body>
</html>
```

6	**Explain one-way and Two-Way data Binding in AngularJS with an Example.**
	Two-Way Binding
	
	The binding goes both ways. If the user changes the value inside the input field, the AngularJS property will also change its value:
	
	`<!DOCTYPE html>`
	`<html>`
	`<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>`
	`<body>`
	
	`<div ng-app="myApp" ng-controller="myCtrl">`
	`Name: <input ng-model="name">`
	`<h1>You entered: {{name}}</h1>`
	`</div>`
	
	`<script>`
		10	CO3	L3

```
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
   $scope.name = "John Doe";
});
</script>

<p>Change the name inside the input field, and you will see the name in the header changes accordingly.</p>

</body>
</html>
```

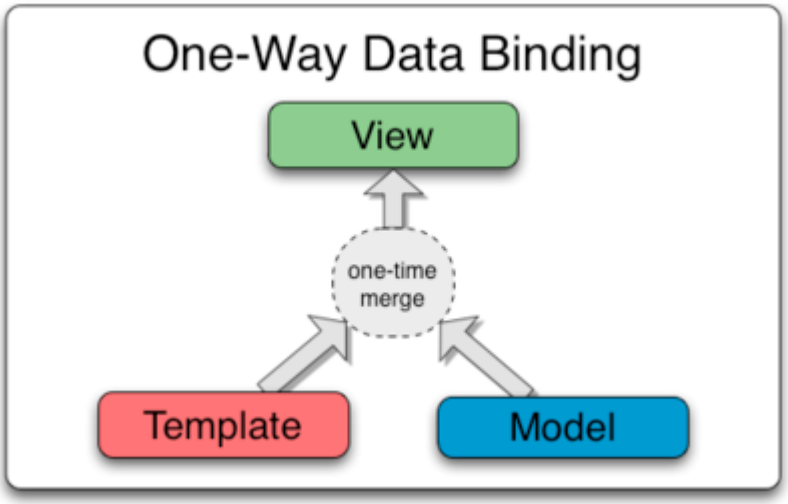
AngularJS Data Binding

Data binding is a very useful and powerful feature used in software development technologies. It acts as a bridge between the view and business logic of the application.

AngularJS follows Two-Way data binding model.

One-Way Data Binding

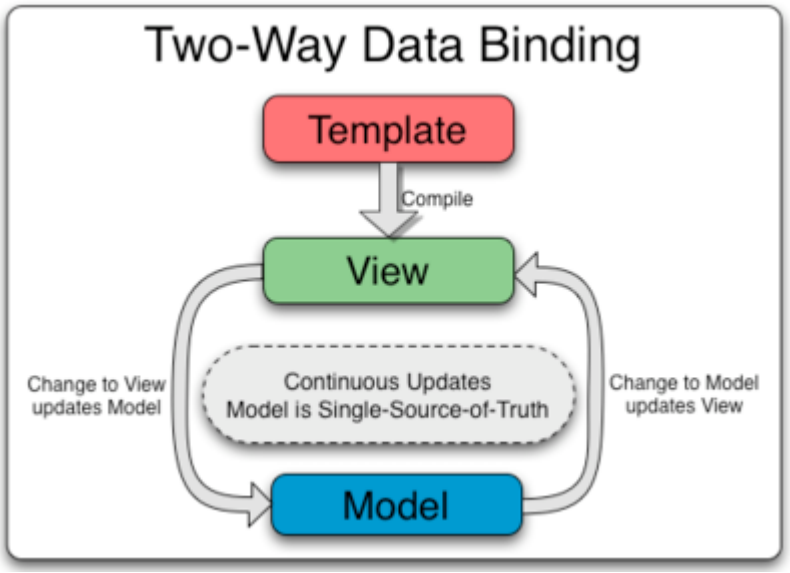
The one-way data binding is an approach where a value is taken from the data model and inserted into an HTML element. There is no way to update model from view. It is used in classical template systems. These systems bind data in only one direction.



Two-Way Data Binding

Data-binding in Angular apps is the automatic synchronization of data between the model and view components.

Data binding lets you treat the model as the single-source-of-truth in your application. The view is a projection of the model at all times. If the model is changed, the view reflects the change and vice versa.




```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js
"></script>
<body>
<div ng-app="" ng-init="firstName='Ajeet'">
<p>Input something in the input box:</p>
<p>Name: <input type="text" ng-model="firstName"></p>
<p>You wrote: {{ firstName }}</p>
</div>
</body>
</html>
```

In the above example, the {{ firstName }} expression is an AngularJS data binding expression. Data binding in AngularJS binds AngularJS expressions with AngularJS data.

{{ firstName }} is bound with ng-model="firstName".

Let's take another example where two text fields are bound together with two ng-model directives:

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js
"></script>
<body>
<div data-ng-app="" data-ng-init="quantity=1;price=20">
<h2>Cost Calculator</h2>
Quantity: <input type="number" ng-model="quantity">
Price: <input type="number" ng-model="price">
<p><b>Total in rupees:</b> {{quantity * price}}</p>
</div>
</body>
```

	</html>			
7	**Discuss different types of expression in AngularJs with an Example.** Expressions **{{ Expression }}** {{Expression}} is used to bind the value with html element and displays the value. It works same as **ng-bind** directive. {{Expression}} is written within two curly brackets. The {{expression}} is basically pure JavaScript expression. **String Expression** We know that string is collection of characters. In AngularJSthe string expression looks like this. <element> {{First String + Second String}} </element> **Example 6.1** <!DOCTYPE html> <html > <head> <title>AngularJS for beginners</title> <script src="js/angular.min.js"></script> </head> <body> Combine Two String Using String Expression ** First String : <input type="text" ng-model="firstString"/> Second String: <input ng-model="secondString"/> Resulting String: {{firstString +" "+secondString}} ** </body> </html> **Output:** **Combine Two String Using String Expression** First String : Ray Second String: Yao Resulting String: **Ray Yao**	10	CO3	L2

Explanation:

"{{firstString +" "+secondString}}" joins two strings together.

{{ expression}} displays the value of expression.

In the above example, the text **Ray** is written in the first text box and **Yao** in the second text box, but in the resulting string area, Ray string and Yao string are combined due to use of string expression {{ }}. Note: The plus + sign is used for string concatenation.

Number Expression

In AngularJSyou can perform different mathematic operation by using Number Expression.

```
<element> {{First Number + Second Number}}
</element>
```

Example 6.2

```
<!DOCTYPE html>
<html >
<head>
   <title>AngularJS for beginners</title> <script src="js/angular.min.js">
</script> </head>
<body>
<h4>Multiply Two Number Using Number Expression</h4> <div ng-app=""
ng-init="firstNumber=9;secondNumber=6"> First Number
   : <input type="number" ng-model="firstNumber"/>
<br><br> Second Number: <input type="number" ng-
model="secondNumber"/><br><br> Result:<p style="color:blue;font-
weight:bold;">{{firstNumber * secondNumber}}</p> </div>
</body>
</html>
```

Output:

Multiply Two Number Using Number Expression

First Number : 9

Second Number: 6

Result:

54

Explanation:

"{{firstNumber * secondNumber}}" multiplies the firstNumber and the secondNumber.

{{ expression}} displays the value of expression.

In the above example, the number **9** is written in the first text box and **6** in the second text box, and **54** is the result of multiplication of 9 and 6. You can perform any arithmetic operation by using Number Expression.

Object Expression

AngularJSobject works like a JavaScript object. The syntax looks like this:

```
object = {property: value}
```

Example 6.3

```
<html >
<script src= "js\angular.min.js"></script>
<body>
<h4>Object Expression</h4> <div ng-app="" ng-init="EmployeeObject =
{Emp_name: 'Jay Smith',  Emp_Month: 'June.15 2015', Emp_salary:
'$8000'}"> <p>Employee Name : {{EmployeeObject.Emp_name}}</p>
<p>Salary's Month: {{EmployeeObject.Emp_Month}}</p> <p>Employee
Salary: {{EmployeeObject.Emp_salary}}</p> </div>
</body>
</html>
```

Output:

```
Object Expression

Employee Name: Jay Smith

Salary's Month: June 15 2015

Employee Salary: $8000
```

"Emp_salary" is a property.

{{ object.property }} displays the value of the property.

Array Expression

The array expression of AngularJS works like JavaScript array. The syntax looks like this:

```
Array=[val1, val2, val3,]
```

Example 6.4

```
<!DOCTYPE html>
<html >
<head>
   <title>AngularJS for beginners</title> <script src="js/angular.min.js">
</script> </head>
<body>
<h4>My Math Result Using Array Expression</h4> <div ng-app="" ng-
init="MyArray=[98,96,93,90,99]"> <p>My score in mathematics is:
{{MyArray[4]}}</p> </div>
</body>
</html>
```

Output:

My Math Result Using Array Expression

My score in mathematics is: 99

8	**What is $scope and explain how to use controls with an example.** What is $scope in AngularJS? ● $scope is an object that acts as a bridge between the controller and the view (HTML). ● It holds data and functions that can be used in an AngularJS application. ● When data in $scope changes, the view updates automatically (two-way data binding). Using $scope with Controllers ● Controllers are used to manage application logic. ● $scope allows passing data from the controller to the view. Example: Using $scope in a Controller <!DOCTYPE html> <html lang="en" ng-app="myApp">> <head>	10	CO4	L3

```
    <title>AngularJS $scope Example</title>
                                                                                <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="myController">

    <h2>AngularJS $scope Example</h2>

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

    <p>Click the button to update the message:</p>
    <button ng-click="updateMessage()">Change Message</button>

    <p>{{ message }}</p>

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

        app.controller("myController", function($scope) {
            $scope.name = "Daya";
            $scope.message = "Welcome to AngularJS!";

            $scope.updateMessage = function() {
                $scope.message = "Message updated successfully!";
            };
        });
    </script>

</body>
</html>
```

9	**With the help of an example for each explain any five array methods of javascript.** Five JavaScript Array Methods with Examples JavaScript provides various built-in array methods that help in manipulating and working with arrays efficiently. Below are five commonly used array methods, along with examples. 1. push() ● Adds one or more elements to the end of an array. ● Modifies the original array. ● Returns the new length of the array. Example let fruits = ["Apple", "Banana"];	10	CO2	L3

```
fruits.push("Mango", "Orange");
console.log(fruits);  // Output: ["Apple", "Banana", "Mango", "Orange"]
```

2. pop()

- Removes the last element from an array.
- Modifies the original array.
- Returns the removed element.

Example

```
let numbers = [10, 20, 30, 40];
let removedElement = numbers.pop();
console.log(numbers);        // Output: [10, 20, 30]
console.log(removedElement); // Output: 40
```

3. shift()

- Removes the first element from an array.
- Modifies the original array.
- Returns the removed element.

Example

```
let colors = ["Red", "Blue", "Green"];
let firstColor = colors.shift();
console.log(colors);      // Output: ["Blue", "Green"]
console.log(firstColor);  // Output: "Red"
```

4. unshift()

- Adds one or more elements to the beginning of an array.
- Modifies the original array.
- Returns the new length of the array.

```
let cities = ["Delhi", "Mumbai"];
cities.unshift("Bangalore", "Hyderabad");
console.log(cities);  // Output: ["Bangalore", "Hyderabad", "Delhi", "Mumbai"]
```

5. sort() Method in JavaScript

- The sort() method is used to sort the elements of an array.
- By default, it sorts elements as strings in ascending order.
- If sorting numbers, a compare function must be used.

```
let fruits = ["Banana", "Apple", "Mango", "Orange"];
fruits.sort();
console.log(fruits); // Output: ["Apple", "Banana", "Mango", "Orange"]
```

10	**Give Example for pattern matching methods of strings in Javascript.**
	
	Regular expressions are used with string methods like .match(), .replace(), .test(), etc.
	
	1. .test() → Checks if a match exists (returns true or false)
	
	let regex = /hello/i; // "i" makes it case insensitive
	
	console.log(regex.test("Hello World")); // true
	
	console.log(regex.test("Hi there")); // false
	
	2. .match() → Returns an array of matches
	
	let text = "I love JavaScript. JavaScript is powerful.";
	
	let regex = /JavaScript/g; // "g" flag for global search
	
	console.log(text.match(regex));
	
	// Output: ["JavaScript", "JavaScript"]
	
	3. .replace() → Replaces text based on a pattern
	
	let text = "I love JavaScript";
	
	let regex = /JavaScript/;
	
	let newText = text.replace(regex, "Python");
	
	console.log(newText);
	
	// Output: "I love Python"
	4. .search() → Finds the index of the first match
	let text = "Learn JavaScript, JavaScript is great!";
	
	let regex = /JavaScript/;
	
	console.log(text.search(regex));
	
	// Output: 6 (index of first occurrence)
		10	CO3	L3