CMR
INSTITUTE OF
TECHNOLOGY

Internal Assessment Test 3 Answer Key – Sep. 2022									
Sub:	Web Technologies					Sub Code:	20MCA23	Branch:	MCA
Date:	30/8/2022	Duration:	90 min's	Max Marks:	50	Sem	II		OBE

Q1) What is jquery? Why jquery ? Explain document ready handler with syntax and example

What is jQuery

• jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax.
• jQuery is a lightweight, "write less, do more", JavaScript library.
• Using raw JavaScript can result in dozens of lines of code.
• The creators of jQuery specifically created the library to make common tasks trivial.
• The real power in this jQuery statement comes from the selector, an expression for identifying target elements on a page that allows us to easily identify and grab the elements we need.

Why jQuery?

 Using raw JavaScript can result in dozens of lines of code for each of these tasks.
 The creators of jQuery specifically created the library to make common tas ks trivial. For example, designers will use JavaScript to "zebra-stripe" tables— highlighting every other row in a table with a contrasting color—taking up to 10 lines of code or more. Here's how we accomplish it using jQuery:

```
$("table tr:nth-child(even)").addClass("striped");
```

 jQuery statements to make your pages come alive.
 The real power in this jQuery statement comes from the selector, an expression for identifying target elements on a page that allows us to easily identify and grab the elements we need

jQuery provides a simple means to trigger the execution of code once the DOM tree, but not external image resources, has loaded. The formal syntax to define such code is as follows:

Syntax

```
$(document).ready(function() {
//jquery statements
});
```

Example

```
$(document).ready(function() {
$("table tr:nth-child(even)").addClass("even");
});
```

First, we wrap the document instance with the jQuery() function, and then we apply the ready() method, passing a function to be executed when the document is ready to be manipulated.
We called that the formal syntax for a reason; a shorthand form used much more frequently is as follows:

```
$(function() {
$("table tr:nth-child(even)").addClass("even");
});
```

By passing a function to $(), we instruct the browser to wait until the DOM has fully loaded (but only the DOM) before executing the code. Even better, we can use this technique multiple times within the same HTML document, and the browser will execute all of the functions we specify in the order that they are declared within the page.

Q2) List all the features and advantages of angular JS

Angular has the following key features which makes it one of the powerful frameworks in the market.

1. MVC – The framework is built on the famous concept of MVC (Model-ViewController). This is a design pattern used in all modern day web applications. This pattern is based on splitting the business logic layer, the data layer, and presentation layer into separate sections. The division into different sections is done so that each one could be managed more easily.
2. Data Model Binding – You don't need to write special code to bind data to the HTML controls. This can be done by Angular by just adding a few snippets of code.
3. Writing less code – When carrying out DOM manipulation a lot of JavaScript was required to be written to design any application. But with Angular, you will be amazed with the lesser amount of code you need to write for DOM manipulation.
4. Unit Testing ready – The designers at Google not only developed Angular but also developed a testing framework called "Karma" which helps in designing unit tests for AngularJS applications.

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

Q3) Explain scale animation and custom drop animation in jquery

1) A custom scale animation

 Consider a simple scale animation in which we want to adjust the size of the elements to twice their original dimensions. We write such an animation as

```
$('.animateMe').each(function(){
$(this).animate( {
width: $(this).width() * 2,
height: $(this).height() * 2
},
2000
);
});
```

 To implement this animation, we iterate over all the elements in the wrapped set
 via each() to apply the animation individually to each matched element.
 This is important because the property values that we need to specify for each element are based upon the individual dimensions for that element.
 If we always knew that we'd be animating a single element (such as if we were using an id selector) or applying the exact same set of values to each element, we could dispense with each() and animate the wrapped set directly.
 Within the iterator function, the animate() command is applied to the element (identified via this) with style property values for width and height set to double the element's original dimensions. The result is that over the course of two seconds (as specified by the duration parameter of 2000), the wrapped elements (or element) will grow from their original size to twice that original size.

2) A custom drop animation

 Let's say that we want to conspicuously animate the removal of an element from the display, perhaps because it's vitally important to convey to users that the item being removed is gone and that they should make no mistake about it.
 The animation we'll use to accomplish this will make it appear as if the element drops off the page, disappearing from the display as it does so.

- If we think about it for a moment, we can figure out that, by adjusting the top position of the element, we can make it move down the page to simulate the drop; adjusting the opacity will make it seem to vanish as it does so. And finally, when all that's done, we want to remove the element from the display (similar to the animated hide()).
- We accomplish this drop effect with the following code:

```
$('.animateMe').each(function(){
$(this).css('position','relative') .animate(
{
opacity: 0,
top: $(window).height() - $(this).height() -
$(this).position().top
},
'slow',
function(){ $(this).hide(); });
});
```

- We once again iterate over the element set, this time adjusting the position and opacity of the elements. But to adjust the top value of an element relative to its original position, we first need to change its CSS position style property value to relative.
- Then for the animation, we specify a target opacity of 0 and a computed top value.
- We don't want to move an element so far down the page that it moves below the window's bottom; this could cause scroll bars to be displayed where none may have been before, possibly distracting users. We don't want to draw their attention away from the animation—grabbing their attention is why we're animating in the first place! So we use the height and vertical position of the element, as well as the height of the window, to compute how far down the page the element should drop.

Q4) What are directives in angular JS? Explain following directive with examples?
i)ng-app ii) ng-model and ng-bind iii) ng-init iv) ng-repeat

App Directive

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.

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>
```

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.

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>
<div ng-app=""    ng-init="Username= 'Andy Smith' "> <p>User Name:
<input type="text" ng-model = "Username"></p> <p ng-
bind="Username"></p> </div>
</body>
</html>
```

Repeat Directive

ng-repeat = “variable in array”

The repeat directive works like a loop. The **ng-repeat** directive repeats to get the 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-8"
/> </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>
```

Q5) Explain following
i)setting attribute ii) removing attribute iii) fetching attribute iv)manipulating element property.

I. Manipulating element properties

jQuery doesn’t possess a specific command to obtain or modify the properties of elements. Rather, we use the native JavaScript notation to access the properties and their values. The trick is in getting to the element references in the first place. The

easiest way to inspect or modify the component elements of a matched set is with the each() command. The syntax of this command is as follows:

Command syntax: each

`each(iterator)`

Traverses all elements in the matched set invoking the passed iterator function for each.

Parameters

`iterator` (Function) A function called for each element in the matched set. The parameter passed to this function is set to the zero-based index of the element within the set, and the element itself is available as the `this` property of the function.

Returns

The wrapped set.

Example :

```
$('img').each(function(n){
        this.alt='This is image['+n+'] with an id of '+this.id;
});
```

II. Fetching attribute values

The attr() command can be used to either fetch the value of an attribute from the first element in the matched set or set attribute values onto all matched elements.
The syntax for the fetch variant of the attr() command is as follows:

Command syntax: attr

`attr(name)`

Obtains the values assigned to the specified attribute for the first element in the matched set.

Parameters

`name` (String) The name of the attribute whose value is to be fetched.

Returns

The value of the attribute for the first matched element. The value `undefined` is returned if the matched set is empty or the attribute doesn't exist on the first element.

The jQuery attr() command is much more than a wrapper around the JavaScript getAttribute() and setAttribute() methods. In addition to allowing access to the set of element attributes, jQuery provides access to some commonly used properties that, traditionally, have been a thorn in the side of page authors everywhere due to their browser dependency.

Normalized name	Source name
`class`	`className`
`cssFloat`	`styleFloat` for IE, `cssFloat` for others (when used with `.css`)
`float`	`styleFloat` for IE, `cssFloat` for others (when used with `.css`)
`for`	`htmlFor`

Normalized name	Source name
`maxlength`	`maxLength`
`readonly`	`readOnly`
`styleFloat`	`styleFloat` for IE, `cssFloat` for others (when used with `.css`)

III. Setting attribute values

There are two ways to set attributes onto elements in the wrapped set with jQuery.

Command syntax: attr

`attr(name,value)`

Sets the named attribute onto all elements in the wrapped set using the passed value.

Parameters

`name` (String) The name of the attribute to be set.

`value` (String|Object|Function) Specifies the value of the attribute. This can be any JavaScript expression that results in a value, or it can be a function. See the following discussion for how this parameter is handled.

Returns

The wrapped set.

Command syntax: attr

`attr(attributes)`

Sets the attributes and values specified by the passed object onto all elements of the matched set

Parameters

`attributes` (Object) An object whose properties are copied as attributes to all elements in the wrapped set

Returns

The wrapped set

IV. Removing attributes

In order to remove an attribute from DOM elements, jQuery provides the removeAttr() command. Its syntax is as follows:

Command syntax: removeAttr
`removeAttr(name)`
Removes the specified attribute from every matched element
Parameters
`name` (String) The name of the attribute to be removed
Returns
The wrapped set

Q6) what is filter, explain following with example

i) uppercase ii)lowercase iii)orderby iv) currency.

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>
```

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>
```

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>
```

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>
```

Q7) Explain how to use child, container and attribute selector in jquery.

2. Using child, container, and attribute selectors

Child selector

Consider the following HTML fragment:

```
<ul class="myList">
        <li><a href="http://jquery.com">jQuery supports</a>
                <ul>
                        <li><a href="css1">CSS1</a></li>
                        <li><a href="css2">CSS2</a></li>
                        <li><a href="css3">CSS3</a></li>
                        <li>Basic XPath</li>
                </ul>
```

```
        </li>
        <li>jQuery also supports
            <ul>
                <li>Custom selectors</li>
                <li>Form selectors</li>
            </ul>
        </li>
</ul>
```

Suppose we want to select the link to the remote jQuery site, but not the links to various local pages describing the different CSS specifications. Using basic CSS selectors, we might try something like ul.myList li a. Unfortunately, that selector would grab all links because they all descend from a list element.

A more advanced approach is to use child selectors, in which a parent and its direct child are separated by the right angle bracket character (>), as in

```
p > a
```

This selector matches only links that are direct children of a <p> element. If a link were further embedded, say within a within the <p>, that link would not be selected. Going back to our example, consider a selector such as

```
ul.myList > li > a
```

This selector selects only links that are direct children of list elements, which are in turn direct children of <ul> elements that have the class myList. The links contained in the sublists are excluded because the <ul> elements serving as the parent of the sublists <li> elements don't have the class myList,

Attribute selectors are also extremely powerful. Say we want to attach a special behavior only to links that point to locations outside our sites.

```
<li><a href="http://jquery.com">jQuery supports</a>
<ul>
<li><a href="css1">CSS1</a></li>
<li><a href="css2">CSS2</a></li>
<li><a href="css3">CSS3</a></li>
<li>Basic XPath</li>
</ul>
</li>
```

What makes the link pointing to an external site unique is the presence of the string http:// at the beginning of the value of the link's href attribute. We could select links with an href value starting with http:// with the following selector:

a[href^=http://]

This matches all links with a href value beginning with exactly http://. The caret character (^) is used to specify that the match is to occur at the beginning of a value. This is the same character used by most regular expression processors to signify matching at the beginning of a candidate string; it should be easy to remember.

There are other ways to use attribute selectors. To match an element that possesses a specific attribute, regardless of its value, we can use

form[method]

This matches any <form> element that has an explicit method attribute. To match a specific attribute value, we use something like

input[type=text]

This selector matches all input elements with a type of text.

div[title^=my]

This selects all elements with title attributes whose value begins with my.

What about an "attribute ends with" selector?

a[href$=.pdf]

This is a useful selector for locating all links that reference PDF files.

And there's a selector for locating elements whose attributes contain arbitrary strings anywhere in the attribute value:

a[href*=jquery.com]

As we would expect, this selector matches all <a> elements that reference the jQuery site.

container selector

li:has(a)

This selector matches all <li> elements that contain an <a> element. Note that this is not the same as a selector of li a, which matches all <a> elements contained within <li> elements.

Table shows the CSS selectors that we can use with jQuery.

Only a single level of nesting is supported. Although it's possible to nest one level, such as

foo:not(bar:has(baz))

additional levels of nesting, such as

foo:not(bar:has(baz:eq(2)))

aren't supported.

Q8) Explain with example any 5 events in angular JS

Click event

ng-click = “expression”

ng-click = “expression” defines a click event. When a button is clicked, an event occurs, and evaluates the expression. The click event normally works on button.

Example 5.1

```
<!DOCTYPE html>
<html >
<head>
   <title>AngularJSfor beginners</title> <script src="js\angular.min.js">
</script> </head>
<body>
<h3>Add Two Numbers Using Click Event</h3> <div ng-app=""  ng-
init="firstNumber=47; secondNumber=23"> <p>First Number :  <input
type="number" ng-model = "firstNumber"></p> <p>Second Number:
<input type="number" ng-model = "secondNumber"></p> <button ng-
click="Result=firstNumber + secondNumber"> Add Numbers </button>
<p>Result:<p style="font-weight:bold; color:blue" ng-bind = "Result">
</p></p> </div>
</body>
</html>
```

Double Click event

ng-dblclick = “expression”

ng-click = “expression” defines a double click event. When a button is double clicked, an event occurs, and evaluates the expression.

Double click event normally works on button.

Example 5.2

```
<!DOCTYPE html>
<html >
<head>
   <title>AngularJS for beginners</title> <script src="js/angular.min.js">
</script> </head>
<body>
<h4>Add Two Numbers Using Double Click Event</h4> <div ng-app=""  ng-
init="firstNumber=26;SecondNumber=89"> <p>First Number:
    <input type="number" ng-model="firstNumber">
</p> <p>Second Number: <input type="number" ng-
model="SecondNumber"></p> <button ng-dblclick="Result=firstNumber
+ SecondNumber">Double Click</button> <p>Result:<p style="font-
weight:bold;color:blue" ng-bind="Result"> </p> </div>
</body>
</html>
```

Mouse Move event

ng-mousemove = “expression”

ng-mousemove = “expression” defines a mouse move event. When the mouse moves, an event occurs, and evaluates the expression.

Mouse move event normally works on div, body and specific area or element.

Example 5.3

```
<!doctype html>
<html>
<head>
<script src="js\angular.min.js"> </script>
</head>
<body ng-app="">
<br><br>
<textarea ng-mousemove="count = count + 1"
ng-init="count=0">
```

Mouse Over event

ng-mouseover = “expression”

ng-mouseover = “expression” defines a mouse over event. When the mouse hovers over, an event occurs, and evaluates the expression.

Mouse over event normally works on div, body and specific area or element.

Example 5.4

```
<!doctype html>
<html>
<script src="js\angular.min.js"></script> <body ng-app="">
 <br><br>
<textarea ng-mouseover="count = count + 1"
ng-init="count=0">
Here is a textarea.
</textarea>
<br><br>
<h2>count: {{count}}</h2> </body>
</html>
```

(Assume you move the mouse over the textarea for 2 times.) **Output:**

Here is a textarea.

count: 2

Mouse Leave event

```
ng-mouseleave = “expression”;
```

ng-mouseleave = “expression” defines a mouse leave event. When the mouse leaves a specified element, an event occurs, and evaluates the expression.

Example 5.5

```
<!doctype html>
<html>
<head>
<script src="js\angular.min.js"> </script>
</head>
<body ng-app=""> <br><br>
<textarea ng-mouseleave="count = count + 1"
ng-init="count=0"> Here is a textarea
</textarea>
<br><br> <h2>count: {{count}}</h2> <body>
</html>
```

(Assume you move the mouse and leave the textarea for 10 times.).

Q9) Explain following methods with examples
i)wrapInner() ii) wrap() iii) append() iv)appendTo() v)html and text()

Command syntax: wrap

`wrap(wrapper)`

Wraps the elements of the matched set with the passed HTML tags or a clone of the passed element.

Parameters

`wrapper` (String|Element) The opening and closing tags of the element with which to wrap each element of the matched set, or an element to be cloned and server as the wrapper.

Returns

The wrapped set.

Example

To wrap each link with the class surprise in a with the class hello, we write

```
$("a.surprise").wrap("<div class='hello'></div>")
```

When multiple elements are collected in a matched set, the wrap() method operates on each one individually. If we’d rather wrap all the elements in the set as a unit, we can use the wrapAll() method instead:

sometimes we may not want to wrap the elements that are in a matched set, but rather their contents. For just such cases, the wrapInner() method is available:

Command syntax: wrapInner

`wrapInner(wrapper)`

Wraps the contents, to include text nodes , elements of the matched set with the passed HTML tags or a clone of the passed element.

Parameters

`wrapper` (String|Element) The opening and closing tags of the element with which to wrap each element of the matched set, or an element to be cloned and server as the wrapper.

Returns

The wrapped set

Moving and copying elements

To add content to the end of existing content, the append() command is available.

Command syntax: append

`append(content)`

Appends the passed HTML fragment or elements to the content in all matched elements.

Parameters

`content` (String|Element|Object) A string, element, or wrapped set to append to the elements of the wrapped set. See the following description for details.

Returns

The wrapped set.

This function accepts a string containing an HTML fragment, a reference to an existing or newly created DOM element, or a jQuery wrapped set of elements. Consider the following simple case:

```
$('p').append('<b>some text<b>');
```

This statement appends the HTML fragment created from the passed string to the end of the existing content of all <p> elements on the page.

If we want to move or copy an element from one place to another, a simpler approach uses the appendTo() command, which allows us to grab an element and move it somewhere else in the DOM.

Command syntax: appendTo

`appendTo(target)`

Moves all elements in the wrapped set to the end of the content of the specified target(s).

Parameters

`target` (String|Element) A string containing a jQuery selector or a DOM element. Each element of the wrapped set will be appended to that location. If more than one element matches a string selector, the element will be copied and appended to each element matching the selector.

Returns

The wrapped set.

Example

```
$('#flower').appendTo('#targets p')
```

Replacing HTML or text content

html() command, which allows us to retrieve the HTML contents of an element when used without parameters or, to set its contents when used with a parameter.

Command syntax: html

`html()`

Obtains the HTML content of the first element in the matched set.

Parameters
none

Returns
The HTML content of the first matched element. The returned value is identical to accessing the `innerHTML` property of that element.

Command syntax: html

`html(text)`

Sets the passed HTML fragment as the content of all matched elements

Parameters
`text` (String) The HTML fragment to be set as the element content

Returns
The wrapped set

We can also set or get only the text contents of elements. The text()command, when used without parameters, returns a string that's the concatenation of all text.

Command syntax: text

`text()`

Concatenates all text content of the wrapped elements and returns it as the result of the command

Parameters
none

Returns
The concatenated string

Q10) What is expression in angular JS explain with example following expressions
i)string expression ii)number expression iii)object expression iv) array 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>
<h4>Combine Two String Using String Expression</h4> <div ng-app="" >
First String    : <input type="text" ng-
model="firstString"/><br><br> Second String: <input ng-
model="secondString"/><br><br> Resulting String:<p
style="color:blue;font-weight:bold;">{{firstString +" "+secondString}}
</p> </div>
</body>
</html>
```

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>
```

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>
```

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>
```