I - 12501

Register No. | | | | | | | | |

BTech Degree Examination May 2022

Fourth Semester

Information Technology

20ITT44 – WEB TECHNOLOGY

(Regulation 2020)

Time: Three hours Maximum: 100 marks

Answer all Questions

Part – A (10 × 2 = 20 marks)

1. Depict the page structure of a HTML document. [CO1,K2]
2. Distinguish between Bootstrap and CSS. [CO1,K2]
3. List the advantages of using Javascript in web application development. [CO2,K1]
4. Write the syntax for declaring and initializing an array in Javascript. [CO2,K1]
5. Specify the usage of following MongoDB shell commands [CO3,K2]
   a. Show dbs
   b. Show collections
6. Comment on the use of Node Js for web application development. [CO3,K2]
7. State the main benefits of TypeScript over JavaScript. [CO4,K1]
8. Give the steps for invoking services in AngularJS. [CO4,K1]
9. Suggest the basic modules to be included while creating the Angular Reactive Form applications. [CO5,K2]
10. Comment on the advantages of using Dependency Injection. [CO5,K2]

Part – B (5 × 16 = 80 marks)

11. a. i) Create a basic web page with navigation link for following pages. (8) [CO1,K3]

Home
About US
Contact US
Products

ii) Create a simple Form using HTML tags for Authentication based access. (8) [CO1,K3]

(OR)

b. i) We Know that Containers are used to pad the content inside them, Create a code for including the two types of containers supported by Bootstrap. (8) [CO1,K3]

ii) Create a HTML page with 3X3 table with Images inside it. Image name and type are user choice. (8) [CO1,K3]

12. a. Design a JavaScript code to handle the following mouse events (16) [CO2,K3]
   1) Mouse Over
   2) Mouse Out
   3) Mouse Move

(OR)

b. Create a JavaScript based validation check for a Username in the form, email and password field with 8 characters Length, One capital letter, Numeric value and one symbol. (16) [CO2,K3]

13. a. i) Create a webserver to handle http request and response with the following url http://localhost:3000/admin and http://localhost:3000/student (8) [CO3,K3]

ii) Create a login page for college symposium and process the data using GET event handling. (8) [CO3,K3]

(OR)

b. Create a web application for library management that connects MongoDB with NodeJS. Write the queries to perform following: (16) [CO3,K3]

i. Insert book name, author name, edition and publication year.
ii. Retrieve the book that costs above Rs.500 and published in the year 2022.

14. a. Illustrate the various kinds of functions in typescript and its different types of parameters with example of user choice. (16) [CO4,K2]

(OR)

b. What are Angular directives? Explain the structural directive in Angular Application with example. (16) [CO4,K2]

15. a. i) Design a web page for hostel management with different components in AngularJS. Perform routing to navigate from one component to other component. (10) [CO5,K3]

ii) Debug the following app.compoent.html file. (6) [CO5,K3]

```
<h1>Custom Pipe</h1>
<b>Square root of 25 is: {{25 | sqrt}</b><br/>
<b>Square root of 729 is: {{729 | | sqrt}}</b>
<br />
<br />
<br />
<a routerLink = "new-cmp">New component<br />
<br/>
<router-outlet><router-outlet>
```

(OR)

b. i) Create Reactive Form for simple course registration with course id, name, description, branch and duration. (10) [CO5,K3]

ii) Comment on the pros and cons of Reactive Form over Template Driven Approach. (6) [CO5,K3]

Bloom's Taxonomy Level	Remembering (K1)	Understanding (K2)	Applying (K3)	Analysing (K4)	Evaluating (K5)	Creating (K6)
Percentage	5	25	70	-	-	-

BTech Degree Exam May 2022

Fourth Semester

Information Technology

20ITT44 WEB TECHNOLOGY

(Regulation 2020)

Answer all Question
Part – A (10 x 2=20 marks)

1. **Page Structure Element** (2)
   - The HTML document itself begins with <html> and ends with </html>.
   - The visible part of the HTML document is between <body> and </body>.

   (or)
   <html>
   <head> ……..</head>
   <body> ………… </body>
   </html>

2. **Bootstrap vs CSS (Any 2 Points)** (2)

Cascading Style Sheet(CSS)	Bootstrap
CSS represent the style and the appearance of content like font, color, margin, padding, etc.	Bootstrap is a free and open-source CSS Framework that is used for developing responsive website.
CSS does not have a grid system.	Bootstrap is based on-grid system.
It currently working on CSS3 which is the latest version of CSS.	Bootstrap currently working on Bootstrap 5 which is the latest version of Bootstrap.
CSS does not provide responsive pages or website.	In Bootstrap we can design a responsive website or webpages.
CSS is more complex than Bootstrap because there is no pre-defined class and design.	Bootstrap is easy to understand and it has much pre-design class.
In CSS, we have to write code from scratch.	In Bootstrap, we can add pre-defined class into the code without writing code.

3. **Use of Javascript (any 4 points)** (2)

- JavaScript to program the behavior of web pages
- Show or hide more information with the click of a button
- Mouse / Button Events
- Form Validation
- Change the color of a button when the mouse hovers over it
- Slide through a carousel of images on the homepage
- Zooming in or zooming out on an image
- Displaying a timer or count-down on a website
- Playing audio and video in a web page
- Displaying animations
- Using a drop-down hamburger menu
- Game development

4. **Array declaration and initialization (Consider any one Method)** (2)

Metho – 1:
```
const cars = ["Saab", "Volvo", "BMW"];
```

Method -2:
```
const cars = [];
cars[0]= "Saab";
cars[1]= "Volvo";
cars[2]= "BMW";
```

Method -3:
```
const cars = new Array("Saab", "Volvo", "BMW");
```

5. (2)
- show dbs - list the existing databases
- show collections; / db.getCollectionNames(); - list all collections in Mongo

6. **Node.js use: (any 4 Points)** (2)

- Node.js is an open source server environment.
- it allows you to run JavaScript on the server.
- It uses asynchronous programming
- It can generate dynamic page content
- It can create, open, read, write, delete, and close files on the server
- It can collect form data
- It can add, delete, modify data in your database

7. **Use of Typescript over Javascript (any 4 Points)** (2)
- It is a strongly typed, object oriented, compiled language.
- It is both a language and a set of tools.
- It is a typed superset of JavaScript compiled to JavaScript.
- It is JavaScript plus some additional features.
- It is portable
- It supports other JS libraries

8. **Access Service:** (2)

An Angular service is a logic that are used to perform some specific task and can be used across multiple components in your application
(OR)

To access the service

```
mydept:string;
  constructor(private myservice:MyserviceService)
  {
    this.mydept=myservice.getdepartment();

  }
```

9. **Module for Reactive Form** (2)

Need to **import ReactiveFormsModule** from the @angular/forms package and add it to your NgModule's imports array to use reactive forms.

10. **Dependency injection, or DI** (2)

It is a design pattern in which a class requests dependencies from external sources rather than creating them.

Part – B

(5 x 16 = 80)

11.a.i **Basic webpage with Navigation [HTML Tags – 4 Mark Navigation – 4 Marks]** (8)

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>Company Name</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
  <script src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.slim.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"></script>
</head>

<body>

<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
  <!-- Brand/logo -->
  <a class="navbar-brand" href="#">Company Logo</a>

  <!-- Links -->
  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link" href="#">Home</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">About US</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Contact US</a>
    </li>
```

```
      <li class="nav-item">
      <a class="nav-link" href="#">Products</a>
    </li>
  </ul>
</nav>

<div class="container-fluid">
  <h3>Brand Name </h3>
  <p> Page Content</p>
</div>
</body>
</html>
```

11.a.ii) **Authentication Form** (8)

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
<h2>Login Form</h2>
<form action="/action_page.php" method="post">
   <div class="container">
    <label for="uname"><b>Username</b></label>
    <input type="text" placeholder="Enter Username" name="uname" required>

    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password" name="psw" required>

    <button type="submit">Login</button>
    </div>
  </form>
</body>
</html>
```

11.b.i **Bootstrap Containers:**
Two types of container classes are: (2)

- .container class provides a responsive fixed width container
- .container-fluid class provides a full width container, spanning the entire width of the viewport

(2)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
  <script src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.slim.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"></script>
</head>
<body>
  <div class="container">
   <p>This part is inside a .container class.</p>
 </div>
<div class="container-fluid">
   <p>This part is inside a .container-fluid class.</p>                    (2)
 </div>
</body>
</html>
```

(2)

11.b.ii <u>3 x 3 Table with Image</u> (8)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Untitled Document</title>
</head>

<body>
<table width="200" border="1">
  <tr>
    <td><img src="profile.jpg" width="200" height="200"></td>
    <td><img src="profile.jpg" width="200" height="200"></td>
    <td><img src="profile.jpg" width="200" height="200"></td>
  </tr>
  <tr>
    <td><img src="profile.jpg" width="200" height="200"></td>
    <td><img src="profile.jpg" width="200" height="200"></td>
    <td><img src="profile.jpg" width="200" height="200"></td>
  </tr>
  <tr>
    <td><img src="profile.jpg" width="200" height="200"></td>
    <td><img src="profile.jpg" width="200" height="200"></td>
    <td><img src="profile.jpg" width="200" height="200"></td>
  </tr>
</table>
</body>
</html>
```

12.a <u>Mouse Events : Over, Out and Move</u>

```
<html>
```

```
<head>
<style>
div {
  width: 100px;
  height: 100px;
  border: 1px solid black;
  margin: 10px;
  float: left;
  padding: 30px;
  text-align: center;
  background-color: lightgray;
}
p {
  background-color: white;
}
</style>
</head>
<body>
<div onmousemove="myMoveFunction()">
  <p>onmousemove: <br> <span id="demo">Mouse over !</span></p>                    (7)
</div>
<div onmouseout="myOutFunction()">
  <p>onmouseout: <br> <span id="demo2">Mouse Out !</span></p>
</div>
<div onmouseover="myOverFunction()">
  <p>onmouseover: <br> <span id="demo3">Mouse over !</span></p>
</div>
<script>
var x = 0;
var y = 0;
var z = 0;
function myMoveFunction() {                                                      (3)
  document.getElementById("demo").innerHTML = z+=1;
}

function myOutFunction() {
  document.getElementById("demo2").innerHTML = x+=1;                             (3)
}
function myOverFunction() {
  document.getElementById("demo3").innerHTML = y+=1;                             (3)
}
</script>
</body>
</html>
```

12.b Form Validation using Javascript:

```
<!DOCTYPE html>
<html>

<head>
        <title>creating mailing system</title>
        <style>
                legend {
                        display: block;
                        padding-left: 2px;
                        padding-right: 2px;
                        border: none;
                }
        </style>
        <script type="text/javascript">
                function validate() {
                                                                                                                        (4)
                        var user = document.getElementById("e").value;
                        var re = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
                        if (re.test(user)) {
                                alert("done");
                                return true;
                        }
                        else {
                                return false;
                        }

        var user = document.getElementById("uname").value;
               var re = /^[a-zA-Z0-9]{8}$/;
        if (re.test(user)) {
          alert("done");                                                                                              (4)
          return true;
        }
        else {

                    return false;
        }

var user = document.getElementById("password").value;
        var re = /^[a-z][A-Z]{8}$/;
        if (re.test(user)) {
          alert("done");                                                                                              (4)
          return true;
        }
        else {

          return false;
        }

                }
        </script>
```

```
</head>

<body bgcolor="cyan">
        <center>

                <form>
                        <fieldset style="width:300px">
                                <legend>Registation Form</legend>
                                <table>
                                        <tr>
                                                <input type="text" id="uname">
                                        </tr>

                                        <tr>
                                                <input type="email" id="e">
                                        </tr>
                                        <br><br>
                                        <tr>
                                                <input type="password" id="password">
                                        </tr>

                                        <tr><input type="submit"
                                                onclick="validate()" value="create">
                                        </tr>
                                </table>
                        </fieldset>
                </form>
        </center>
</body>

</html>
```

(4)

13.a.i <u>Node JS URL Handling</u> (8)

```
var http=require('http');
http.createServer(function(req,res){
 if(req.url=="/admin")
 {
  res.writeHead(200,{'Content-type':'text/plain'});
  res.end('Admin Page');
 }
 else if (req.url=="/student")
  {
   res.writeHead(200,{'Content-type':'text/html'});
        res.write('<html><body>');
        res.write('<h1>Student Page</h1>');
        res.write('</body></html>');
   res.end();
  }
 else {
  res.end('This is wrong url');
 }

}).listen(3000);
```

13.a.ii <u>Login System Using GET method:</u>

<u>Login Form:</u>

(4)

```
<html>
        <head>
                <title></title>
        </head>
        <body>
                <form action="http://localhost:7777/login" method="get">
                Enter your name<input type="text" name="username" value=""/><br/>
                Enter the Password<input type="text" name="password" value=""/><br/>
                        <br/>
                        <input type="submit" name="login" value="Login"/>
                </form>
        </body>
</html>
```

<u>Node JS Code:</u>

<u>Note: students can also make use of querystring() function to retrieve data</u>

(4)

```
http=require("http");
url = require("url");

function onRequest(request, response) {
var add = url.parse(request.url,true);
console.log("Request for " + add + " received.");

var query=add.query;
var name=query.username;
var password=query.password;
response.write("Hello "+name+"  Login success");
response.end();
}
http.createServer(onRequest).listen(7777);
console.log("Server has started...");
```

13.b. <u>Library Management System</u>

<u>HTML Form Code:</u>

(4)

```
<html>
<body>
<form method="post">
        Book name<input type="text" name="book_name" value=""/><br/>
        Author name<input type="text" name="author_name" value=""/><br/>

        Edition<input type="text" name="edition" /><br/>
        Publication Year<input type="text" name="pub_year" /><br/>

         Price<input type="text" name="cost" value=""/><br/>

        <input type="submit" name="show" value="Show"
                                formaction="http://localhost:3000/show"/>
        <input type="submit" name="login" value="Save"
                                formaction="http://localhost:3000/save"/>

</form>
</body>
</html>
```

Mongo DB Processing:

```
var MongoClient=require('mongodb').MongoClient;
var url='mongodb://127.0.0.1:27017/';
console.log("MongoDB");

exports.saveData= function (book_name, author_name,edition,pub_year,cost, response) {

        MongoClient.connect(url,function(err,db){ //Connection to server
        if(err) throw err;                                                                    (4)
        var dbcon=db.db('WTdemo');  //opening the db
        var msg="";
        var myobj = {"book_name":bookname , "author_nameemail":author,
”edition”:edition,”pub_year”:pub_year,”cost”:cost};
        dbcon.collection("books").insertOne(myobj,function(err,res){
                if (err)
                {
                        console.log(err);
                        msg="Book Data Not inserted";
                }
                else
                {
                        msg="Book Name:"+bookname +"  ***Inserted***";
                        console.log("Document inserted");
                }

                response.write(msg);
                response.end();
                db.close();
        });

        });

  };

  exports.showData= function ( response) {
```

```
            MongoClient.connect(url,function(err,db){ //Connection to server
            if(err) throw err;
            var dbcon=db.db('WTdemo');  //opening the db
            var msg="";
            var query = {  $and: [
          {“cost”: { $gt: 500} },                                                                 (4)
          { "pub_year":2022}};

            dbcon.collection("books").find(query).toArray(function(err, result) {
                    if (err)
                    {
                            console.log(err);
                            msg="Error!!!";
                    }
                    else
                    {

            var Length = result.length;
            msg="<table><tr><td>S.No</td><td>Book Name</td><td>Author Name</td></tr>";
            for(var i=0; i<Length; i++)
            {
            msg+="<tr><td>"+(i+1)+"</td><td>"+result[i].book_name+"</td><td>"+result[i].autho
    r_name+"</td></tr>";

            }
            msg+="</table>";
                    }
                    response.write(msg);
                    response.end();
                    db.close();

              });


            });

      };
```

Server Program:

```
var module = require('./db_module');
var url = require('url');
var querystring = require('querystring');                                                        (4)
var http = require('http');

http.createServer(function(request, response) {
var data1 = '';
request.on('data', function(chunk) {
          data1 += chunk;
       });

request.on('end', function() {
var book_name = querystring.parse(data1)["book_name"];
var author_name = querystring.parse(data1)["author_name"];
var edition = querystring.parse(data1)["edition"];
var cost = querystring.parse(data1)["cost"];
var pub_year = querystring.parse(data1)["pub_year"];

if (request.url === '/show') {
module.showData( response);
          }
 else if (request.url === '/save') {
module.saveData(book_name, author_name,edition,pub_year,cost, response);
          }
     });

}).listen(3000);
console.log("Server started");
```

  or

Inserting Book Details Code

Html Code:
Server.js File :

```
const express=require('express')
const cors=require('cors')
const {connectDB}=require('./connection');
const{users}= require('./usermodel')

const app=express()
app.use(express.json())
app.use(cors())

app.post('/insert',async(req,res)=>{

   const data=await users.create(req.body);
   console.log(data);
   res.send(data);
})
app.get('/get',async(req,res)=>{
   const data =  await users.find({p_year:'2022',cost:{$gt:500}})
   res.send(data)
})
```

```
app.listen(7000,()=>
{
console.log('server listening at port 7000');
connectDB()
.then(    (e)=>
{
console.log("mongodb connected");}
)
.catch( (e)=>
{
console.log("mongodb connection error");
})

})
```

insert.html file :

```
<html>

  <body>
    <form method="post" name="f1" action="/">
      <label>Enter Book Name</label>
      <input type="text"  id="name" />
      <br /><br />
      <label>Enter author name</label>
      <input type="text"  id="auth" />
      <br /><br />
      <label>Enter Edition</label>
      <input type="text"  id="edition" />
      <br><br>
      <label>Enter Publication Year</label>

      <input type="text"  id="p_year">
      <br><br>
      <label >Enter cost of the book</label>
      <input type="number"  id="cost">
      <br><br>

      <input type="button" id="b1" value="Submit">
    </form>


    <div id="result" ></div>

  </body>
  <script>
```

```
const insertuser = document.querySelector("#b1");
const result = document.getElementById("result");


insertuser.addEventListener('click',()=>{
    const name = document.querySelector("#name").value;
    const auth = document.querySelector("#auth").value;
    const edition = document.querySelector("#edition").value;
    const year = document.querySelector("#p_year").value;
    const cost = document.querySelector("#cost").value;


    fetch("http://localhost:7000/insert",{
                mode:"cors",
                method:"POST",
                headers:{
                    'Content-type':'application/json'
                },
                body:JSON.stringify({
                    b_name:name,
                    a_name:auth,
                    edition:edition,
                    p_year:year,
                    cost:cost,
                })

            })
            .then(  (e)=>
            {

                document.getElementById("result").innerHTML="Inserted sucessfuly";
            })
            .catch((e) =>
                {

                document.getElementById("result").innerHTML="ERROR";
            }
            )


})


  </script>
</html>
```

connection.js file :

```
const mongoose = require('mongoose');
exports.connectDB=async()=>await mongoose.connect('mongodb://localhost:27017/db')
```

usermodel.js file :

```
const mongoose = require('mongoose');

const usermodel = new mongoose.Schema({
   b_name:String,
   a_name:String,
   edition:String,
   p_year:String,
   cost:Number
})
exports.users = mongoose.model("lib",usermodel,"lib");
```

Database Name : db
Collection Name : lib

Retrieve Books that costs above Rs.500 and published in the year 2022

http://localhost:7000/get

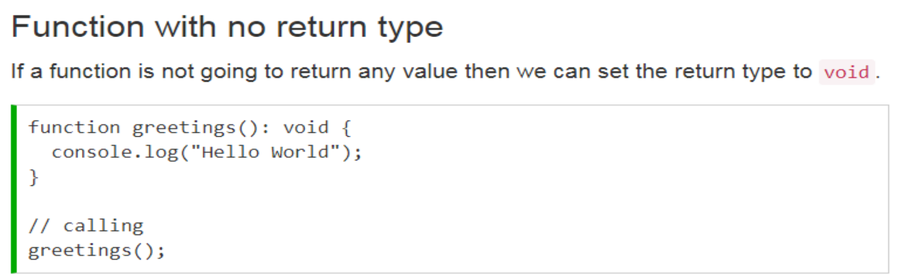
14.a <u>Types of Functions supported by Typescript:</u> **(Any 4 type with example 4x 4= 16)** (16)

<u>Function with Return type</u>

```
function Greet(greeting: string, name: string ) : string
{
   return greeting + ' ' + name + '!';
}
```

<u>Function with Optional Parameter</u>

```
function Greet(greeting: string, name?: string ) : string {
   return greeting + ' ' + name + '!';
}
```



Function with no return type

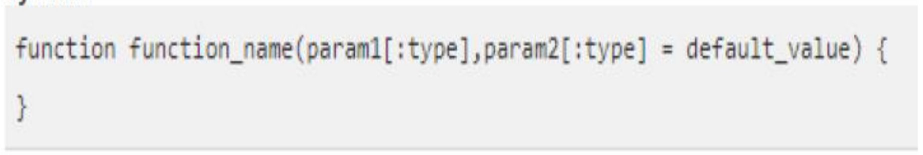
If a function is not going to return any value then we can set the return type to `void`.

```
function greetings(): void {
  console.log("Hello World");
}

// calling
greetings();
```

<u>Function with Default Value:</u>

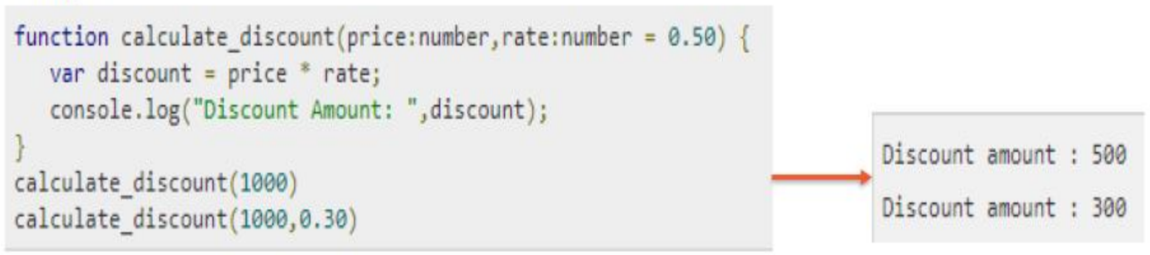
- **Syntax:**



```
function function_name(param1[:type],param2[:type] = default_value) {

}
```

- **Example:**



```
function calculate_discount(price:number,rate:number = 0.50) {
   var discount = price * rate;
   console.log("Discount Amount: ",discount);
}
calculate_discount(1000)
calculate_discount(1000,0.30)
```

```
Discount amount : 500
Discount amount : 300
```

<u>Anonymous Function:</u>

```
var result = function (a:number, b:number) {
    return a + b;
};
var c = result(12, 2);
console.log(c);
```

<u>Rest Parameters:</u>

```
function addNumbers(...nums:number[]) {
   var i;
   var sum:number = 0;

   for(i = 0;i<nums.length;i++) {
      sum = sum + nums[i];
   }
   console.log("sum of the numbers",sum)
}
```

<u>Arrow function / Lambda Function:</u>

```
var fun2 = (): number => {
  return Math.random();
};

// calling
```

```
console.log("Random number: " + fun2());
```

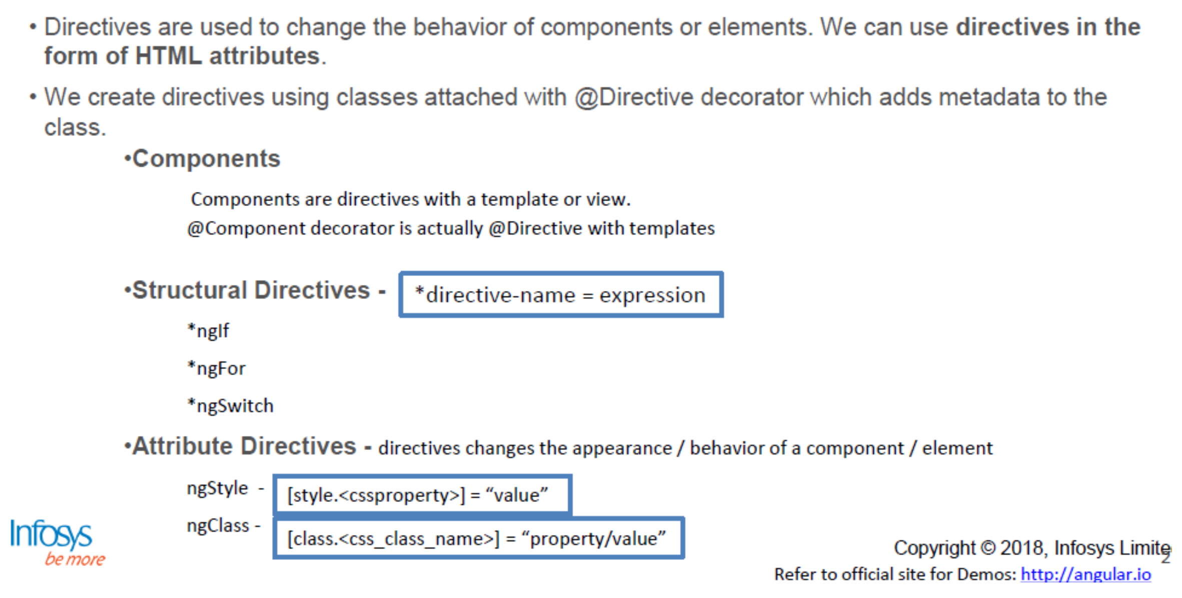
Function Overloading:

```
//overloaded functions
function sum(x: number, y: number): number;
function sum(x: number, y: number, z: number): number;
//the combined implementation

function sum(x: number, y: number, z?: number): number {
   if (typeof z == 'undefined') {
      return x + y;
   } else {
      return x + y + z;
   }
}

let y = sum(1, 2);//calling first overloaded function
console.log(y);
let m = sum(1, 2, 3);//calling second overloaded function
console.log(m);
```

14.b. Angular Directives:



- Directives are used to change the behavior of components or elements. We can use **directives in the form of HTML attributes**.
- We create directives using classes attached with @Directive decorator which adds metadata to the class.
  - **Components**
    Components are directives with a template or view.
    @Component decorator is actually @Directive with templates
  - **Structural Directives** - *directive-name = expression
    *ngIf
    *ngFor
    *ngSwitch
  - **Attribute Directives** - directives changes the appearance / behavior of a component / element
    ngStyle - [style.<cssproperty>] = "value"
    ngClass - [class.<css_class_name>] = "property/value"

Infosys be more

Copyright © 2018, Infosys Limited 2
Refer to official site for Demos: http://angular.io

(4)

Structural Directives:

Ngfor:

(4)

App.component.ts

```
months = ["January", "Feburary", "March", "April", "May",
  "June", "July", "August", "September",
  "October", "November", "December"];
```
App.component.html
```
 <div>
  <ul>
<li *ngFor = "let i of months">{{i}} </li>
</ul>
</div>
```

Ngif:

App.component.html
```
<div>
   <h4>NgIf</h4>
   <h6 *ngIf="a>2">{{ a }} </h6>
</div>
```
(4)

App.component.ts
```
a=10;
```

ngswitch:

App.component.ts

selectedValue=3;

App.component.html (4)

```
<div [ngSwitch]="selectedValue">
   <div *ngSwitchCase="'One'">One is Pressed</div>
   <div *ngSwitchCase="'Two'">Two is Selected</div>
   <div *ngSwitchCase="3">Three is Selected</div>
   <div *ngSwitchCase="4">{{selectedValue}}</div>
   <div *ngSwitchDefault>Default Option</div>
</div>
```

15.a.i Angular Router Tutorial (10)

Step 1:
Create a new Project
```
ng new Routerdemo
```

Step 2:
Create three components namely home, about and dashboard by using the following command.

```
ng g c home
ng g c about
ng g c dashboard
```

Step 3:

Now app.module.ts file will look like as below

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { DashboardComponent } from './dashboard/dashboard.component';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    AboutComponent,
    DashboardComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 4:
Add the following in app-routing.module.ts file

```
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { DashboardComponent } from './dashboard/dashboard.component';
const routes: Routes = [
  {
    path:'home',
    component:HomeComponent
  },
  {
    path:'aboutus',
    component: AboutComponent
  },
  {
    path:'dashboard',
    component: DashboardComponent
  }
];
```

Step5 :Add the following line in app.component.html

```
<div class="toolbar" role="banner">
  <nav>
    <a routerLink="home">Home</a>
    <a routerLink="aboutus">About</a>
    <a routerLink="dashboard">Dashboard</a>
  </nav>
</div>
<router-outlet></router-outlet>
```

Step 6:
Run the Project using the following command

```
ng serve --open
```

15.a.ii Debugging: (6)

```
<h1> Custom Pipe </h1>
<b> Square root of 25 is:{{25 | sqrt}} </b><br/>
<b> Square root of 729 is:{{729 | sqrt}} </b>
<br/>
<br/>
<br/>
<a routerLink="new-cmp">New component</a><br/>
<br/>
<router-outlet></router-outlet>
```

15.b.i Reactive form: (10)

Step 1: Create a new Project

```
ng new ReactFormProject
```

Step 2 : To use reactive form controls, import ReactiveFormsModule from the @angular/forms package and add it to your NgModule's imports array.

```
import { ReactiveFormsModule } from '@angular/forms';

@NgModule({
  imports: [
    // other imports ...
    ReactiveFormsModule
  ],
})
```

Step3: Add new Component into the project

```
ng g c ReactForm
```

Step 4: In the react-form.component.ts import the FormGroup , FormControl and Validators classes from the @angular/forms package.

```
import { FormGroup, FormControl } from '@angular/forms';
import { Validators } from '@angular/forms';
```

With reactive forms, the logic is declared entirely in the component class.

Code:

```
import { FormControl, FormGroup } from '@angular/forms';
import { Validators } from '@angular/forms';
 …………
export class ReactFormComponent implements OnInit {
        constructor() { }
        ngOnInit(): void {
                }
              });

                onSubmit()
                {
                  // TODO: Use EventEmitter with form value
                  this.msg="First Name : "+this.profileForm.value.Name;
                }
}
```

Step 3: Adding a Form to the Component Template

Open react-form.component.html and add the following lines of code

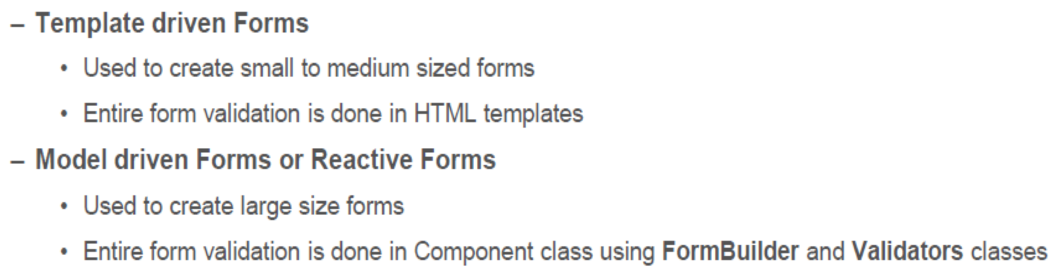
```
<form [formGroup]="profileForm" (ngSubmit)="onSubmit(profileForm)">
   <label for="course_id"> Course ID: </label>
   <input id="course_id" type="text" formControlName="course_id">
   <label for="first-name"> Name: </label>
   <input id="first-name" type="text" formControlName="Name">
   <label for="branch">Branch: </label>
   <input id="branch" type="text" formControlName="branch">
   <label for="description">Description: </label>
   <input id="description" type="description" formControlName="description">
   <label for="duration">Duration: </label>
   <input id="duration" type="duration" formControlName="duration">

      <button type="submit" >Submit</button>
 </form>
```

Step 4: Run the application

```
ng serve --open
```

15.b.ii (6)



- **Template driven Forms**
  - Used to create small to medium sized forms
  - Entire form validation is done in HTML templates
- **Model driven Forms or Reactive Forms**
  - Used to create large size forms
  - Entire form validation is done in Component class using **FormBuilder** and **Validators** classes

<u>(OR)</u>

Comparison Index	Reactive Forms	Template-driven Forms
Setup (form model)	Reactive forms are more explicit. They are created in component class.	Template-driven forms are less explicit. They are created by directives.
Data model	Structured	Unstructured
Predictability	Synchronous	Asynchronous

Form validation	Functions	Directives
Mutability	Immutable	Mutable
Scalability	Low-level API access	Abstraction on top of APIs