Unit VI

Unit III

3 Front End Technologies

3.1 Introduction to Web Framework

Q.1 What is a web framework ? Give the reasons for using web framework. ☞ [SPPU : Dec.-22, Marks 6]

Ans. :

- Web framework is a software framework which is designed to support the development of web applications including web services, web resources and Web APIs.
- Web framework is basically a software library that enables developers to write software that runs on the web.
- Web frameworks may be written in different languages and using different methodologies. The term 'stack' is applied to refer to the collection of different languages, software, and frameworks in use behind a specific service.

Why Web Framework ?

Following are some important reason which indicate why do we use web frameworks -

1. **Saves time :** The most striking feature of web framework is that it saves time and energy in developing any app because the developer doesn't need to worry about session handling, error handling and authentication logic. These functions are taken care of by web frameworks.
2. **Well organized app :** The web framework itself takes care of managing directories and files. This makes the complete application well organized.

3. **Flexibility and customizable :** Add-ons, themes, plugins, widgets enable rapid customization on the web application. This brings out a lot of flexibility in code development.
4. **Code reusability :** Framework also promotes the reuse of code.
5. **Security :** Framework makes developer sure that the application uses good security measurements because framework itself takes care of it.

Q.2 List and explain the features of any three popular web frameworks. ☞ **[SPPU : Dec.-22, Marks 6]**

Ans. :

1) Bootstrap

- Very well known and built by Twitter
- Easy to learn and looks professional
- Can be difficult to customize components

2) Angular

- Built by Google
- Well supported
- Encourages reusability
- Improves application scalability

3) React

- Built by Facebook
- Bundles frontend code into components
- Organizes code and data to make code more reusable
- Large learning curve

4) Express

- Uses JavaScript
- Very customizable
- Very lightweight
- Less built-in features
- Node is very fast

3.2 MVC

Q.3 What is MVC ? Explain MVC architecture in detail.

☞ [SPPU : June-22, Dec.-22, Marks 6]

Ans. : MVC :

- The MVC stands for model, view and controller. It is a pattern for the architectural framework.

MVC architecture :

It consists of three parts -

1. **Model :** This part of the architecture is responsible for managing the application data. This module responds to the request made from view. The model gives instructions to the controller to update when the response is made.
2. **View :** This part takes care of the presentation of data. The data is presented in desired format with the help of view. This is a script based system using JSP, ASP, PHP and so on.
3. **Controller :** The controller receives input, validates it and then performs business operations that modify the state of the data model. The controller basically responds to user requests and performs interaction with the model. Refer Fig. Q.3.1.

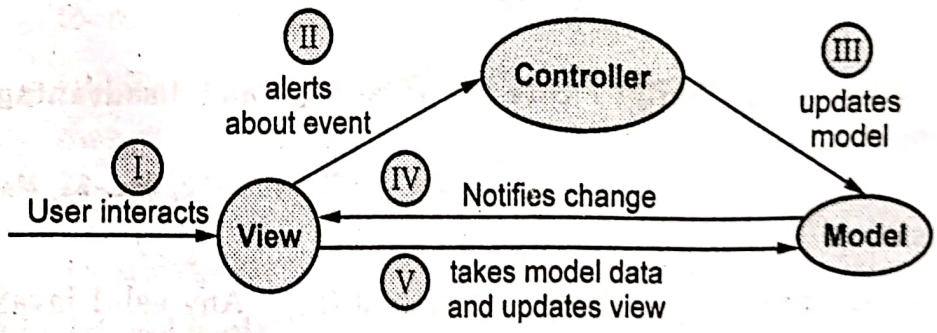


Fig. Q.3.1 MVC architecture

- Model represents the data.
- View is the user interface.
- Controller is the request handler.

Q.4 Enlist the features of MVC framework.

Ans. :

1. **Separation of logic :** There is a separation of application tasks such as input logic, business logic and UI logic. This makes testing and debugging easy. Modification in one component does not affect the other.
2. **Ability to provide multiple views :** In MVC model, we can create multiple views.
3. **Faster development process :** MVC supports rapid and parallel development. If an MVC model is used to develop any particular web application then it is possible that one programmer can work on the view while the other can work on the controller to create the business logic of the web application.
4. **Returns data without formatting :** MVC pattern returns data without applying any formatting. Hence, the same components can be used and called for use with any interface.
5. **Customization :** It is an extensible and pluggable framework. MVC framework are designed so that the components can be easily replaced or customized.

3.3 TypeScript

Q.5 What is TypeScript ? Give the advantages and disadvantages of using it.

☞ **[SPPU : June-22, Dec.-22, Marks 6]**

Ans. :

- TypeScript is a typed superset of JavaScript. Any valid JavaScript code is basically a valid code for typescript as well.
- TypeScript extends JavaScript by adding data types, classes and other object-oriented features with type-checking.
- The typescript compiles to plain javascript code.

- TypeScript is a programming language developed and maintained by Microsoft.
- TypeScript may be used to develop JavaScript applications for both client side and server side execution.

Advantages

1. It is an open source language with continuous development.
2. It runs on any web browser.
3. The TypeScript code can be called from existing JavaScript code. It can also work with JavaScript frameworks and libraries without any issue.
4. TypeScript has a support for the latest JavaScript feature from ECMAScript2015. The ECMAScript specification is a standardized specification of a scripting language. This standard is meant to ensure the interoperability of web pages across different web browsers. The 6th edition, ECMAScript 6 (ES6) and later renamed to ECMAScript 2015. Some of major enhancements of this standard include modules, class declarations and so on.

Disadvantages

1. Web browsers do not understand the TypeScript code. If we want to run TypeScript code in the web browser then we need to compile it. This compilation process converts it to JavaScript and then we can see the TypeScript code running in the web browser.
2. It does not support abstract classes.
3. TypeScript takes a time to compile the code.

Q.6 List and explain the features of TypeScript.

Ans. :

1. **Portability :** TypeScript is a portable because it can be executed on any browser or operating system. It can run on any environment where JavaScript runs.

2. **JavaScript :** The code written in JavaScript with valid .js extension can be converted to TypeScript by simply changing the extension from .js to .ts.
3. **Static Type checking :** This feature allows type checking at compile.
4. **Support for Library :** For developing the TypeScript coding we can use all JavaScript frameworks, tools and libraries easily.
5. **Support for OOP :** The TypeScript can make use of various features of object-oriented programming languages such as classes, interfaces, inheritance and so on.
6. **Client and Server side Development :** TypeScript allows the development of both server-side and client-side programming.

3.4 Modules in TypeScript

Q.7 Write a sample application in TypeScript to demonstrate the use of modules.

Ans. :

- A module is used to set code written in TypeScript. A module can be defined in a separate **.ts** file which can contain functions, classes, interfaces.
- We use the prefix **export** with all the definitions which we want to include in a module and want to access from other modules.

Demo Example

Step 1 : We will create **calculator.js** is a module which contains a class named **Addition**. This class is prefixed with keyword **export**.

calculator.js

```
export class Addition {
  a: number;
  b: number;
```

```
    constructor(a: number, b: number) {
        this.a = a;
        this.b = b;
    }
    display() {
        console.log ("Addition: " + (this.a + this.b) );
    }
}
```

The above class contains two variables **a** and **b**. It also contains one function **display()**.

Step 2 : Now we will create one JavaScript file for importing the class created in above step.

Definition Import : A module can be used in another module using an import statement.

Syntax :

```
import { export name } from "file path without extension"
```

Following file is used for import file.

app.ts

```
import {Addition} from './calculator';

let obj = new Addition(100, 200);
obj.display();
```

3.5 Angular Version 10+

Q.8 What is Angular ? List and explain its features.

☞ [SPPU : Dec.-22, Marks 6]

Ans. :

- Angular 10+ is a JavaScript framework which is used to create single page applications(SPAs). The single page application is a kind of application which dynamically rewrites the current page rather than loading entire new pages from a server. The SPAs provide the user a very fluid, reactive and fast experience.

- Angular is completely based on components. Due to which the applications created in Angular give the reactive speed.
- The Angular applications are created with the help of HTML and TypeScript.

Features of Angular

1. **Support for multiple platforms :** Angular is a cross platform language. It allows the developers to create desktop applications and progressive web applications
2. **High speed web applications :** Angular applications are fast and load quickly with new component router. Angular is specially used as a front end web development for the programming languages like Node.js, .Net, PHP, Struts, Spring and other servers for instant rendering by just using HTML and CSS.
3. **Productivity :** Angular uses template syntax and command-line tools for adding and testing the components. This enhances the productivity of Angular.
4. **Localization :** One of the best features of Angular 10+ is that the latest angular version supports for merging of multiple translation documents which could only load one file in the previous versions.
5. **Dynamic development :** Angular team releases two versions in a year. This helps the developer to have handy and updated functionalities each time along with each new release.
6. **Full stack development :** The full stack development is a buzzword that start-up companies use in this modern of software development. Angular is a complete framework of JavaScript. It provides testing, animations and accessibility.

3.6 Angular CLI

Q.9 Explain the use of Angular CLI.

Ans. :

- The Angular CLI is a command-line interface tool that you use to initialize, develop, scaffold and maintain Angular applications directly from a command prompt or terminal.
- It provides some commands and schematics which help us with faster code generation. Most of the frameworks and applications normally come with CLI.
- We can install Angular with the help of Angular CLI.

3.7 Angular Architecture

Q.10 Explain the Angular architecture with suitable block diagram.

Ans. : The main building blocks of Angular architecture are - (Refer Fig. Q.10.1)

1. Modules
2. Template
3. Components
4. Data Binding
5. Directives
6. Metadata
7. Injections and Services.

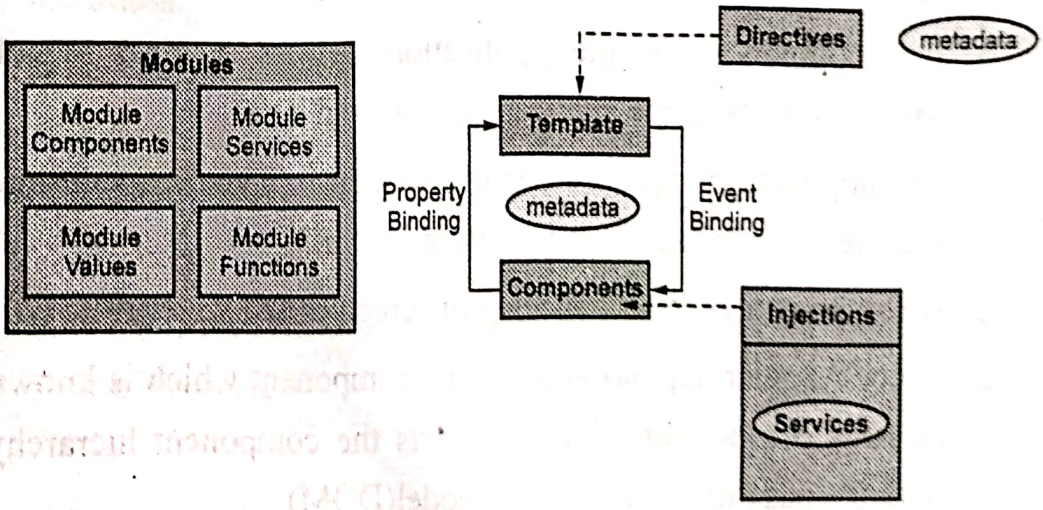


Fig. Q.10.1 Angular architecture

Let us discuss them one by one

1. **Modules :**
   - Every Angular App has a root module which is called as AppModule. It provides bootstrap mechanism that launches the application.
   - Organizing your code into distinct functional modules helps in managing the development of complex applications.
   - If we want to use another custom Angular module, then we need to register that module inside the **app.module.ts** file.
   - An app typically contains many functional modules.
2. **Template**
   - Template is basically used to combine HTML with Angular Markup and modify HTML element before displaying them.
   - Template directives provide program logic and binding markup which connects application data and DOM.
   - Typically in out Angular App the template looks like this -

```
<div style="text-align:center">
 <h1>
 //code for data binding
 </h1>
</div>
```

3. **Components**
   - The component in the application defines a class which basically contains the application logic and data.
   - The application logic is normally written in TypeScript format and view of the page is in HTML template.
   - A component controls a display of screen called a view.
   - Every Angular app has at least one component which is known as **root component** which connects the component hierarchy with the page document object model(DOM).

4. Data Binding

There are two types of data bindings -

1. **Property Binding :** This type of binding allows to pass the interpolated values from application data to HTML. The interpolated values are specified in {{ and }} bracket pair. For instance - the student.name is interpolating value.

```
<p>Name : {{ student.name }}</p>
```

2. **Event Binding :** Event binding is used to capture events on the user's end on the app and respond to it in the target environment by updating the application data.

5. Directives

- The directives extend the HTML with the help of new syntax.
- The directives are written using the prefix **ng.**
- These directives attach certain behaviour to the elements. There are two commonly used directives **ng-model** and **ng-bind.**
- **ng-model :** The ng-model binds the value of the HTML control with the specified AngularJS expression value.
- **ng-bind :** This directive replaces the HTML control value with a specified AngularJS expression value.

6. Metadata

- Metadata means data about data.
- Metadata tells Angular how to process a class. It is used to decorate the class so that it can configure the expected behavior of a class.
- The **metadata for a component** class has a template that defines a view.

- The **metadata for a service class** consists of information Angular needs to make it available to components through dependency injection (DI).

7. Injections and Services

- In Angular, it is a practice for data or logic that it should not be associated with a specific view. It is created as a **service class**. Due to this, the data or logic is shared across components.
- By **dependency injection**, we mean to allow access to the service by subscribing it. It acts as a delegate to the service.
- The **@Injectable** decorator immediately precedes the service class definition. The decorator provides the metadata that allows our service to be injected into client components as a dependency.

3.8 Angular Project Structure

Q.11 What is the use of src folder and package.json file in angular.

Ans. : scr folder

- This is the source folder of our angular application. That means this is the place where we need to put all our **application source code**. So, every component, service class, modules, everything we need to put in this folder only. Whenever we create an angular project, by default the angular framework creates lots of files folder within the **src** folder.

package.json

- This file is mandatory for every **npm** project. It contains basic information regarding the project such as name, description, license, commands which can be used. It also specifies dependencies. The dependencies are the packages required by the application to work correctly.

3.9 Angular Lifecycle

Q.12 Explain Angular lifecyle with the help of example.

Ans. :

- In Angular, every component has a life-cycle, a number of different stages it goes through from initializing to destroying. The view of the main component is rendered in **index.html** file using <app-root>. The **<app-root>** is specified using **selector**. In the **module.ts** file using the annotation **@NgModule** the component is specified in the **declarations** section.

Step 1 : Angular browser displays the **index.html** file on execution of Angular app. We normally say that "Angular renders your app"

In the **index.html** file, there is only one element **<app-root></app-root>**.

Step 2 : The <app-root> is defined as **selector**. In **src->app** folder. Just open the **app.component.ts**. This file contains the **selector** as follows

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

Step 3 : Inside the **app.module.ts** file the specification for AppComponent is mentioned.

```
import { AppRoutingModule } from './app-
routing.module';
import { AppComponent } from './app.component';
import { WelcomeComponent } from
'./welcome/welcome.component';

@NgModule({
  declarations: [
    AppComponent,
    WelcomeComponent
  ],
```

Step 4 : This is the flow of execution Angular app. This app can be displayed in the browser window when we issue the command **ng serve -o** on command prompt or terminal window.

3.10 Angular Modules

Q.13 What is Angular module ? Explain.

Ans. :

- Angular modules consolidate components, directives and pipes into cohesive blocks of functionality.
- For defining the module we use the **ngModule.**
- Every Angular application has at least one NgModule class, the root module, which is conventionally named **AppModule** and resides in a file named **app.module.ts.**
- An NgModule is defined by a class decorated with **@NgModule().** The @NgModule() decorator is a function that takes a single metadata object, whose properties describe the module.

3.11 Angular Components

Q.14 Give the sample layout of the Angular application with multiple components. Explain how to create and use the components in angular.

Ans. :

- Components are basic building block element of Angular. The components are associated with the template and they define different aspects of the user interface.
- Each component consists of -
  - An HTML template that declares what renders on the page.
  - A Typescript class that defines behavior.
  - A CSS selector that defines how the component is used in a template.
- Typical organized page in Angular App contains multiple components. It can be viewed as follows -

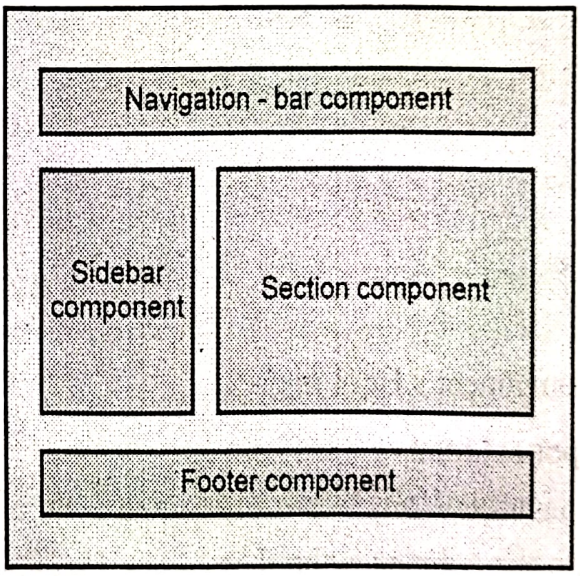


Fig. Q.14.1 Multiple components on angular App page

Demo Example

- We can create a component named 'welcome' as follows -

```
ng generate component welcome
```

Step 2 : We will keep our **app.component.html** file very simple. Delete everything from it which comes by default from the Angular framework and simply add following code to it.

app.component.html

```
<h1>Hello</h1>
```

Now the **welcome.component.ts** file is as follows -

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-welcome',
  templateUrl: './welcome.component.html',
  styleUrls: ['./welcome.component.css']
})
export class WelcomeComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }
  currentUser: String = "Anuradha"
}
```

This selector name will be used in app.component.html file (Refer step 5)

The code for component's Html file is as -

welcome.component.html

```
<p>welcome {{currentUser}}</p>
```

Finally we write app.component.html as -

app.component.html

```
<h1>Hello</h1>
<app-welcome></app-welcome>
```

3.12 Angular Data Binding

Q.15 What is Angular data binding ? Enlist the types of data binding.

Ans. :

- Data binding is a technique where data is in synchronous with components and its views.
- Data binding defines the communication between components and its views. It requires dynamic HTML and does not require any complex programming.

Types of data binding

- There are basically two approaches used for data bindings - One way Binding and two way Binding. The one way binding is done using interpolation binding, property binding and event binding.

Q.16 Explain interpolation binding in angular with example.

Ans. :

- Using interpolation we can bind the values to the user interface elements. This is one-way data binding. Because data moves from component to view element or HTML element. This is one-directional flow of data.

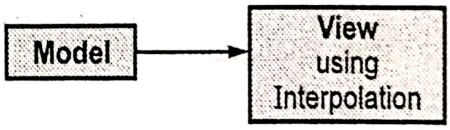


Fig. Q.16.1 Interpolation binding

Demo Example

app.component.ts

```
import { Component } from '@angular/core';

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
```

```
})
export class AppComponent {
 public roll_no:number = 101
  public name:String = 'Anuradha';
}
```

app.component.html

```
<h1 style="text-align: center;">WELCOME</h1>
<h2 style="text-align: center;">Roll_no: {{roll_no}}</h2>
<h2 style="text-align: center;">Name: {{name}}</h2>
```

Q.17 Explain event binding and property binding in angular with example.

☞ [SPPU : June-22, Marks 6]

Ans. : Property binding

- The property binding is one-way binding in which we can set the properties of the element to the user interface page. In this binding [ ] is used for data binding. Following example illustrates it.

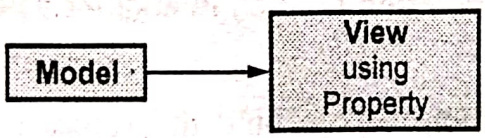


Fig. Q.17.1 Property binding

Demo Example

app.component.ts

```
import { Component } from '@angular/core';

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 public roll_no:number = 101;
 public name:String = 'Anuradha';
 public image = "/assets/technical.png"
}
```

app.component.html

```
<h1>WELCOME</h1>
<h2>Roll_no: {{roll_no}}</h2>
<h2>Name: {{name}}</h2>
<img [src] = "image" alt="Logo" style="height:80px;width:150px">
```

Event binding

- The event binding is the flow of data from view to component. In this case, the event is triggered on the view or HTML page. The view page sends data to the component.

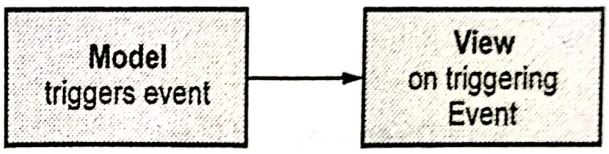


Fig. Q.17.2 Event binding

- In the following application, on clicking the button on the view page (i.e. app.component.html page), the data is send to the component

Demo Example

app.component.html

```
<h1>WELCOME</h1>
<h2>Roll_no: {{roll_no}}</h2>
<h2>Name: {{name}}</h2>
<button (click)="display()">Enroll</button>
```

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  public roll_no:number = 101;
  public name:String = 'Anuradha';
```

```
display()
{
 alert("You are now Enrolled!!! ");
}
}
```

Script Explanation :

- By above two steps we have created **Enroll** button on the HTML page and on clicking the button, "You are Enrolled" message will be displayed.

3.13 Directives and Pipes

Q.18 What is ngfor directive in angular ? Explain with example.

Ans. :

- This is a type of structural directive that shows the template for each item in the collection or list. The ngFor is prefixed by *. The syntax is

```
<li *ngFor="let item of items;">...</li>
```

Demo Example

- Following example illustrates how to use *ngFor directive for displaying the list of students.

app.component.ts

```
import { Component } from '@angular/core';
@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 students = [
  {roll_no:2,name:'Archana'},
  {roll_no:4,name:'Chinmaya'},
  {roll_no:6,name:'Soumya'},
```

```
  {roll_no:8,name:'Swati'}
 ];
}
```

app.component.html

```
<h1> List of Students</h1>
<table border='5'>
 <th>Roll_No</th>
 <th>Name</th>
 <tr *ngFor="let student of students;">
  <td>{{student.roll_no}}</td>
  <td>{{student.name}}</td>
 </tr>
</table>
```

Q.19 What is pipe ? Explain with example.

☞ [SPPU : June-22, Marks 6]

Ans. :

- Pipes are used to transform the data. That means pipes take data as input and are getting transformed into another format. The pipes are written using pipe operator which is denoted by |. The pipe can be applied to any view or template and to any data input.

Built-in Pipes

1. Lowercase
2. Uppercase
3. Currency
4. Date
5. JSON

Demo Example

app.component.ts

```
import { Component } from '@angular/core';
@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 student = {
```

```
  roll:101,
  name:'Anuradha',
  city:'Pune',
  fees:10000,
  DOB:'09/01/1980'
 };
}
```

app.component.html

```
<h1> Welcome</h1>
<div style="font-weight:bold;">
 Name in lower-case: {{student.name | lowercase }}
</div>
<div style="font-weight:bold;">
 City in upper-case: {{student.city | uppercase }}
</div>
<div style="font-weight:bold;">
Date of Birth: {{student.DOB | date }}
</div>
<div style="font-weight:bold;">
Fees: {{student.fees | currency:'INR'}}
</div>
```

3.14 Angular Services and Dependency Injections (DI)

Q.20 Write a simple angular application that demonstrates the services and dependency injection.

Ans. :

- A service is typically a class with a well-defined purpose. Angular component may depend upon one or more services.
- Dependency injection is a technique in which an object receives other objects that it depends on.
- Services are wired together using a Dependency Injection (DI) mechanism. We just need to have an injectable service class to be able to share these service methods to any consuming component

Also, DI in our Angular components/services can be implemented using either constructor or injector.

- The code that passes the service to the client is called the injector. Instead of the client specifying which service it will use, the injector tells the client what service to use.
- An Angular service is plain Typescript class having one or more methods (functionality) along with @Injectable decorator.
- The simple typescript used to create Angular service is

```
import { Injectable } from '@angular/core'; @Injectable()
export class myServicename {
  constructor() { }
}
```

Here, **@Injectable** decorator converts a plain Typescript class into Angular service.

Demo Example

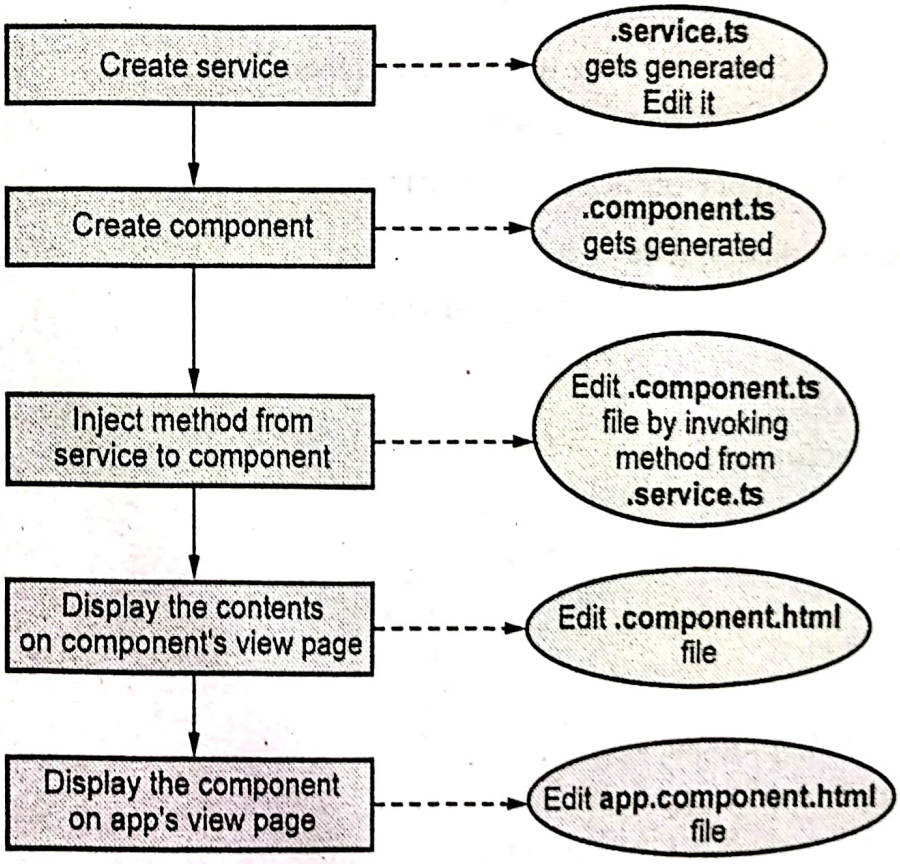


Fig. Q.20.1 Steps for implementation of service

- In the following application we create a service in which the list of students is defined with their roll numbers and names using **getStudents()** method. This service is then injected into the **component**. The component is then rendered on the web page.

app-routing-module.ts

```
import { NgModule } from '@angular/core';
import {StudentComponent} from './student/student.component';
import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [
 {path:'student', component:StudentComponent}
];

@NgModule({
 imports: [RouterModule.forRoot(routes)],
 exports: [RouterModule]
})
export class AppRoutingModule { }
```

Generate a service by using following command

```
ng generate service service_stud
```

Note that the name of the service is **service_stud**

Service-stud.service.ts

```
import { Injectable } from '@angular/core';

@Injectable({
 providedIn: 'root'
})
export class ServiceStudService {

 constructor() { }
 getStudents() {
  const studentList = [
   {roll_no:101,name:'Archana'},
   {roll_no:102,name:'Supriya'},
   {roll_no:103,name:'Sharda'},
   {roll_no:104,name:'Swati'},
```

```
    {roll_no:105,name:'Aishwarya'}
  ];
  return studentList;
 }
}
```

- The component's .ts file that is student.component.ts file and invoke the method **getStudents()** created in the above step. The code is as follows

student.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ServiceStudService } from '../service-stud.service'
@Component({
 selector: 'app-student',
 templateUrl: './student.component.html',
 styleUrls: ['./student.component.css']
})
export class StudentComponent implements OnInit {

 constructor(private serviceStudService:ServiceStudService) { }
 studentList:any;
 ngOnInit(): void {
  this.studentList= this.serviceStudService.getStudent();
 }
}
```

- The view file of the component i.e. **student.component.html**, in this file we are displaying the contents of the component.

student.component.html

```
<table border='1'>
  <th>RollNo</th>
  <th>Name</th>
  <tr *ngFor = "let s of studentList">
    <td>{{s.roll_no}}</td>
    <td> {{s.name}}</td>
  </table>
```

- The app's view file and that is app.component.html and invoke the component

app.component.html

```
<h1>Student List</h1>
<app-student></app-student>
```

3.15 Angular Routers

Q.21 Explain the concept of angular routers with the help of example.

Ans. :

- Routing means navigation from one page to another. To handle the navigation from one view to another we use **Router**. The Router enables navigation by interpreting a browser URL as an instruction to change the view.

Demo Example

- Following is a simple application in which we have created two components - one for **employee** and another for **department**.
- When we enter the URL **localhost:4200/employees** the **employee page** will be displayed. Similarly when we enter the URL **localhost:4200/departments** the **department page** will be displayed.
- This is called **routing**.

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { EmployeesComponent }
from './employees/employees.component';
import{DepartmentsComponent}
from './departments/departments.component'
const routes: Routes = [
 {
 path:'employees',
```

```
 component:EmployeesComponent
 },
 {
  path:'departments',
  component:DepartmentsComponent
 }

];
@NgModule({
 imports: [RouterModule.forRoot(routes)],
 exports: [RouterModule]
})
export class AppRoutingModule { }
```

- **Script Explanation :** In above script, we have first imported two components **EmployeesComponent and DepartmentsComponent.**
- In the Routes[] array we have defined the URL paths and name of the components.

app.component.html page is as follows -

```
<h1> Routing Demo</h1>
<router-outlet></router-outlet>
```

- **Script Explanation :** Router outlet is a dynamic component or a routing component, which is used to display the views based on corresponding navigation.
- When we use the tag <router-oulet></router-outlet> then at that location Angular will insert the views of corresponding components. These routes are specified in **app-routing.module.ts** file.

employees.component.html

<h2>Empoyees Page</h2>

departments.component.html

```
<h2>Departments Page</h2>
```

3.16 Angular Forms

Q.22 Explain the template driven form in angular.

Ans. :

- Template-driven forms rely on directives defined in the **FormsModule.**
- The **NgModel** directive binds the value changes in the attached form element with changes in the data model. Thus we can use ngModel to create two-way data bindings for reading and writing input-control values.
- The **NgForm** directive creates a top-level **FormGroup** instance and binds it to a <form> element to track aggregated form value and validation status. As soon as we import **FormsModule,** this directive becomes active by default on all <form> tags. There is no need to add a special selector.

app.module.ts

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

import { FormsModule } from '@angular/forms';
@NgModule({
 declarations: [
  AppComponent
 ],
 imports: [
  BrowserModule,
  FormsModule,
  AppRoutingModule
 ],
 providers: [],
```

```
 bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.html

```
<form #myForm="ngForm" (ngSubmit)="submit(myForm.value)">
   <div class="form-group">
    <label for="firstName">First Name </label>
    <input type ="text" name="firstName" [(ngModel)]="firstName" >
  </div>
  <br>
  <div class="form-group">
    <label for="lastName">Last Name </label>
    <input type ="text" name="lastName" [(ngModel)]="lastName" >
  </div>
  <br>
  <div class="form-group">
    <label for="mobileNum">Mobile Number </label>
    <input type ="text" name="mobileNum" [(ngModel)]="mobileNum"
>
  </div>
  <br><br>
  <button type="submit">Submit</button>
  </form>
```

Q.23 How to use reactive form in angular ?

Ans. :

- Reactive forms provide a model-driven approach to handling form inputs whose values change over the time. In Reactive forms, we need to import "ReactiveFormsModule" from angular forms library.
- Reactive forms are forms where we define the structure of the form in the component class. i.e. we create the form model with Form Groups and Form Controls. We also define the validation rules in the component class. Then, we bind it to the HTML form in the template.

- **FormControl** encapsulates the state of a single form element in our form. It stores the value and state of the form element and helps us to interact with them using properties and methods.
- **FormGroup** represents a collection of form Controls. It can also contain form groups and form arrays. In fact, an angular form is a **FormGroup.**

app.module.ts

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

import { FormsModule, ReactiveFormsModule }
from '@angular/forms';
@NgModule({
 declarations: [
  AppComponent
 ],
 imports: [
  BrowserModule,
  FormsModule,
  ReactiveFormsModule,
  AppRoutingModule
 ],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule { }
```

Script Explanation :

- We have imported two modules and those are - **FormsModule** and **ReactiveFormsModule**

app.component.html

```
<h1>Student Form</h1>

<form [formGroup]="form" (ngSubmit)="submit()">

  <div class="form-group">
    <label for="name">Name</label>
    <input
      formControlName="name"
      id="name"
      type="text"
      class="form-control">
  </div>
  <br>
  <div class="form-group">
    <label for="email">Email</label>
    <input
      formControlName="email"
      id="email"
      type="text"
      class="form-control">
  </div>
  <br><br>

  <button class="btn btn-primary" type="submit">Submit</button>
</form>
```

Script Explanation :

- Here, we have created a form with two textboxes and one submit button control. The FormGroup is used while defining the form elements.

In ts file. we will write **submit()** and get all input fields values.

app.component.ts

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

@Component({
```

```
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  form = new FormGroup({
   name:new FormControl(''),
   email:new FormControl('')
  });
  submit()
  {
   console.log(this.form.value);
  }
}
```

3.17 Introduction to ReactJS

Q.24 What is ReactJS ? Enlist the features of it.

Ans. :

- ReactJS is an open source, **component-based** front end **JavaScript library** maintained by Facebook.
- This library is responsible only for the **view layer** of the application. That means this JavaScript is for building user interfaces.

Features of ReactJS

- **Virtual DOM :** DOM stands for Document Object Model. It also provides a way to update the content, structure and style of the document. Virtual DOM is a representation of original DOM. When user updates something on the web application, DOM gets updated. Updating the DOM is very slow, most of the JavaScript frameworks update the whole DOM which makes it slower. But actually there is no need to update the whole DOM, instead, these frameworks should update only the part of DOM that is required to update. This is what the virtual DOM does. This is why ReactJS's

DOM manipulation is much faster than other frameworks. Due to this property, whenever any change is made in the web application, then those changes are reflected on the web page within no time.

- **Components :** This feature allows the web developer to create custom elements which can be reused in HTML.
- **JSX :** JSX is an extension of JavaScript syntax. JSX can be seen as a combination of javascript and XML. The syntax of JSX is very simple that makes writing components very easy.
- **One way Data Binding :** The ReactJS is designed in such a way that it follows, unidirectional or one way data binding. That means, data is allowed to flow in one direction at a time. This helps in achieving greater control over the application. This makes helps in increasing the efficiency of an application.

Q.25 Write a simple ReactJS application that displays Welcome user message.

Ans. : index.js

```
import React from 'react';
import ReactDOM from 'react-dom';

ReactDOM.render(
  <h1> Welcome, user!!!</h1>,
  document.getElementById('root')
);
```

3.18 React Components

Q.26 Explain the functional component in angular.

Ans. :

- Functional components are simple JavaScript functions. The functional components return the JSX(JavaScript XML)code to render the DOM tree. Following code fragment shows the functional component

```
function Welcome(props) {
 return <h1>Welcome {props.name}</h1>;
}
```

Alternatively the functional component can be created using arrow function as follows -

```
const Welcome = (props) => {
return <h1>Welcome {props.name}</h1>;
}
```

Note that the function name always start with capital letter.

Demo Example

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
function DisplayStudent(student){
 return <div>
<h2>Student Information....</h2>
  <p>
   <label>Roll_no:<b>{student.Roll}
   </b></label>
  </p>
  <p>
   <label>Name:<b>{student.Name}
   </b></label>
  </p>
  <p>
   <label>phone:<b>{student.Phone}
   </b></label>
  </p>
 </div>;
}

const element = <DisplayStudent Roll="111" Name = "Chitra" Phone
 = "1234567890"></DisplayStudent>
//render this component as Id='root'
ReactDOM.render(element,document.getElementById('root'));
```

Note that the function name's first letter must

Here the functional component is defined.

Passing the values to the functional component

Q.27 Explain the class component in angular.

Ans. :

- To define the class component we have to first create a class and **extend React.Component** class.
- Class components are ES6 classes that return JSX. Below, is a class component which is similar to Welcome function(as given in Functional component) :

```
class Welcome extends React.Component {
 render() {
  return <h1>Welcome, {this.props.name}</h1>;
 }
}
```

Demo Example

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
class Student extends React.Component{
 render(){
  return <div>
    <h2>Student Information....</h2>
    <p>
    <label>Roll_no:<b>{this.props.Roll}</b></label>
   </p>
   <p>
    <label>Name:<b>{this.props.Name}</b></label>
   </p>
   <p>
    <label>phone:<b>{this.props.Phone}</b></label>
   </p>

   </div>
 }
}
const element =<Student Roll="111" Name = "Chitra" Phone =
"1234567890"></Student>
ReactDOM.render(element,document.getElementById('root'));
```

Q.28 Given the difference between functional and class components.

Ans. : Difference between Functional component and Class component

Sr. No.	Functional component	Class component
1.	Functional components are short and simple, easy to understand.	Class components are complex.
2.	These are plain JavaScript functions that accept props as an argument and return React element.	These are the classes that gets extended from React.Component.
3.	There is no need to use **render()** method.	It must have **render()** method returning HTML.
4.	Constructors are not used.	Constructors are used as it needs to store the state.
5.	Functional components generally are just used for display purposes. These components are used to handle user interactions or state updates.	Class components are used as container components to handle state management and wrap child components.

3.19 Inter Components Communication

Q.29 Write a simple angular app to demonstrate the communication among the components.

Ans. : index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
class Student extends React.Component{
 constructor(props){
  super(props);
 }
 render(){
```

```
  return <div>
    <h2>Student Information....</h2>
    <p>
    <label>Roll_no:<b>{this.props.Roll}</b></label>
   </p>
   <p>
    <label>Name:<b>{this.props.Name}</b></label>
   </p>
   <p>
    <label>phone:<b>{this.props.Phone}</b></label>
   </p>
   <p>
   <label>Total Marks:<b>{this.props.Marks}</b></label>
   </p>
<Marks English={this.props.English} Maths={this.props.Maths}
Science={this.props.Science}></Marks>
   </div>
 }
}

class Marks extends React.Component{
 constructor(props){
   super(props);
 }
 render(){
   return <div>
     <h3>Marks of Each Subject are...</h3>
     <p>
     <label>English:<b>{this.props.English}</b></label>
    </p>
    <p>
     <label>Maths:<b>{this.props.Maths}</b></label>
    </p>
    <p>
     <label>Science:<b>{this.props.Science}</b></label>
    </p>
   </div>
 }
}
```

```
const element =<Student Roll="111" Name = "Chitra" Phone =
"1234567890" Marks ="95" English="40" Maths="35"
Science="20"></Student>
ReactDOM.render(element,document.getElementById('root'));
```

3.20 Components Styling

Q.30 Can we use CSS stylesheet to apply the styles to React Components ? Illustrate it with the help of some example.

Ans. :

We can create a separate CSS stylesheet file and import it in the demo app.

Mystyle.css

```
h1 {
  color:red;
  background-color:rgb(157, 255, 0);
  padding:15px;
 }
```

Script Explanation : The above one is a normal .css file in which we have used the styles with color, background-color and padding attributes. Note that the color can be specified either with color name or with rgb values.

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './Mystyle.css';
const Mystylefunction = () => {
 return (
  <>
   <h1>Component with style</h1>
   <p>This is a normal text without any style</p>
  </>
 );
}
const element = <Mystylefunction></Mystylefunction>
ReactDOM.render(element,document.getElementById('root'));
```

Note that we have imported **.css** file which is already created in step 2

3.21 Routing

Q.31 What is react-routing technique ? Explain.

Ans. :

- Routing is a technique by which the user is directed to different pages based on their action.
- ReactJS router is a single page application. The single page application is a web application that interacts with the user by dynamically rewriting the current page, rather than loading entire new pages from the server. When a user types a specific URL into the browser and if this URL path matches any 'route' inside the router file, the user will be redirected to that particular route.
- Most of the popular sites such as Youtube, facebook and Instagram make use of ReactJs for rendering multiple views via routing technique.
- There are two important packages that are used while using ReactJs routing.
  1. **react-router :** It provides core routing components and functions for the React Router application.
  2. **React-router-dom :** This package enables the user to implement dynamic routing in a web app.
- React Router is the core package for the router. React Router DOM contains DOM bindings and gives you access to React Router by default.
- The React Router API is based on three components :
  - **<Router> :** The router that keeps the UI in sync with the URL.
  - **<Link> :** Renders a navigation link.
  - **<Route> :** Renders a UI component depending on the URL.
  - **<BrowserRouter> :** It is used for handling the dynamic URL.

Demo Example

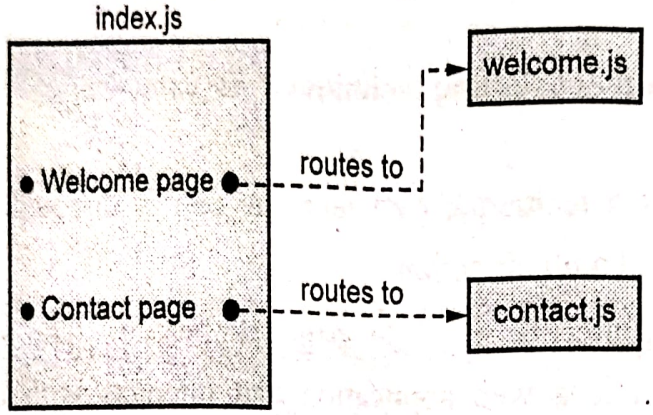


Fig. Q.31.1

welcome.js

```
import React from 'react';
function Welcome(){
  return(
    <div>
      <h1> Welcome User!!!!</h1>
    </div>
  );
}
export default Welcome
```

contact.js

```
import React from 'react';
function Contact(){
  return(
    <div>
      <p> <b>Phone:</b>1234567890 </p>
      <p><b>Email:</b>abc.xyz@gmail.com</p>
    </div>
  );
}
export default Contact
```

3.22 Redux - Architecture

Q.32 Explain redux - architecture in detail.

☞ [SPPU : June-22, Marks 6]

Ans. :

- Redux is a JavaScript library which can be used with ReactJS applications for state management. Redux manages the application state.
- For creating user interfaces React uses Redux. React Redux is the official React building for Redux.
- Redux works on following core concepts :

1. **Action :** Actions in Redux represent work being done. The actions can be fetching user data, logging the user in and so on.
2. **Reducer :** The reducers determine how state should change,
3. **Store :** The store holds a centralized copy of state and
4. **Middleware :** The middleware allows you to inject custom behavior into the process.

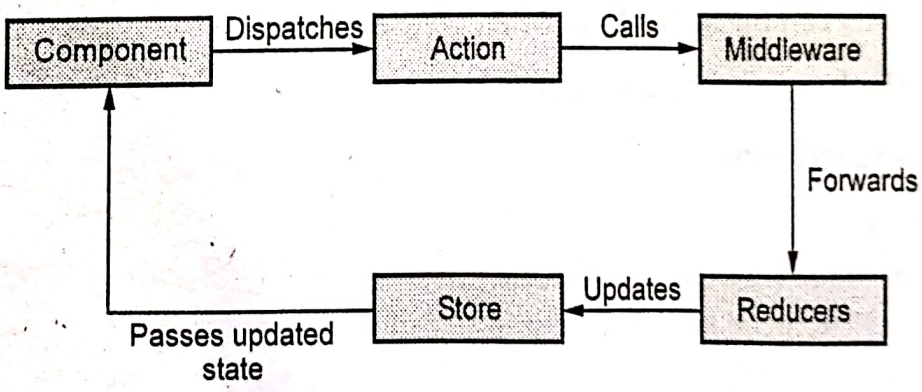


Fig. Q.32.1 Redux architecture

Action : It is a simple JavaScript object which is used to send (dispatch) information from react component to reducer.

Reducer : It is a pure synchronous function which takes two things - the previous state and action object. This function returns the **next state.**

Store : It is a centralized place. The store contains the state of the application. Developers can access and update the store with the help of various methods.

Middleware : It is used to execute asynchronous code between action and reducers.

3.23 Hooks

Q.33 Explain different types of Hooks in ReactJS.

☞ [SPPU : June-22, Marks 6]

Ans. :

- Hooks were added to React in version 16.8.
- Hooks allow function components to have access to state and other React features. Hence there is no need for class components.
- Hooks allow us to "hook" into React features such as state and lifecycle methods.

Rules of Hook

1. Only call the hooks at the top level.
2. Only call the hooks from React Functions.
3. Hooks can not be conditional.

1. useState() hook

index.js

```
import React,{useState} from 'react';
import ReactDOM from 'react-dom';
function Counter() {
const[count,setCount]=useState(0)
return(
 <div>
  <h3>Counter</h3>
  <button
onClick={()=>setCount(count+1)}>Count={count}</button>
 </div>
)
```

```
}
const element=<Counter></Counter>
ReactDOM.render(element,document.getElementById("root"));
```

2. useEffect() hook

- The **useEffect** Hook allows you to perform side effects. These side effects can be fetching data, updating it,using timers and so on.
- The syntax of **useEffect** is

```
useEffect(<function>, <dependency>)
the second parameter is optional.
```

- **useEffect** runs on every render(display). That means that when the count changes, a render happens, which then triggers another effect. There are three ways by which useEffect can be used

1. When there is no dependency

```
useEffect(() => {
  //Runs on every render
});
```

2. To run on first render

```
useEffect(() => {
  //Runs only on the first render
}, []);
```

3. render with the state value

```
useEffect(() => {
  //Runs on the first render
  //And any time any dependency value changes
}, [state]);
```

3. useContext() hook

- Context is a technique which helps in passing data or state through the component tree without having to pass props down manually through each nested component.
- The purpose of using context is to share data which is considered as a global data for the tree of react components.

- For using the context we need to create it first. To create context, we have to first Import createContext and initialize it. The code required to do so is as follows

```
import { useState, createContext } from "react";
import ReactDOM from "react-dom";

const UserContext = createContext()
```

- Next step is that we will use the Context Provider to wrap the tree of components that need the state Context. Just wrap the child components in the **Context Provider** and supply the state value. Because of this, all the child components can access the state value.
- In order to use the Context in a child component, we need to access it using the useContext Hook. So we will import it using the statement

```
import { useState, createContext, useContext } from "react";
```

- Then using **useContext()** method we can access the state value in any component. The Syntax is -

```
const context_name = useContext(initialValue);
```

END ...

Unit IV

4 Back End Technologies

Part I : Node.JS

4.1 : Introduction to Node.JS

Q.1 What is Node.js ? Enlist the features of Node.js.

Ans. : NodeJS is an open source technology for server.

- Using Node.js we can run JavaScript on server.
- It runs on various platform such as Windows, Linux, Unix, and MacOS.

Features

- Following are some remarkable features of node.js -

1) **Non blocking thread execution :** Non blocking means while we are waiting for a response for something during our execution of tasks, we can continue executing the next tasks in the stack.

2) **Efficient :** The node.js is built on V8 JavaScript engine and it is very fast in code execution.

3) **Open source packages :** The Node community is enormous and the number of permissive open source projects available to help in developing your application in much faster manner.

4) **No buffering :** The Node.js applications never buffer the data.

5) **Single threaded and highly scalable :** Node.js uses a single thread model with event looping. Similarly the non blocking

response of Node.js makes it highly scalable to sever large number of requests.

4.2 : Handling Data I/O in Node.JS

Q.2 Explain any four methods of console object in node.js with suitable examples. ☞[SPPU : Dec.-22, Marks 6]

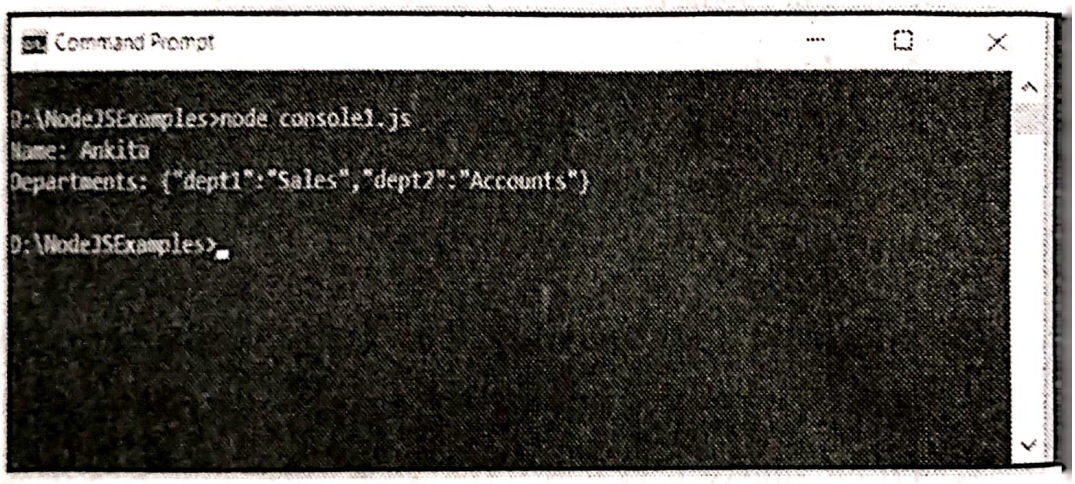
Ans. :

1) **console.log([data][, ...]) :** It is used for printing to stdout with newline. This function can take multiple arguments similar to a printf statement in C.

console1.js

```
var emp_name = 'Ankita',
        emp_depts = {
        dept1: 'Sales', dept2: 'Accounts'
        };
console.log('Name:%s'+'\n'+ 'Departments: %j',emp_name, emp_depts);
```

Output



2) **console.info([data][, ...]) :** This method is similar to console.log.

console2.js

```
var emp_name = 'Ankita',
  emp_depts= {
    dept1: 'Sales',
     dept2: 'Accounts'};
```

```
console.info('Name: %s'+'\n'+ 'Departments: %j',emp_name,
emp_depts);
```

Output

```
Name: Ankita
Departments: {"dept1":"Sales","dept2":"Accounts"}
```

3) console.error ([data][, ...]) : This method prints to stderr with newline

console3.js

```
var code = 401
console.error('Error!!!',code)
```

Output

```
Error!!! 401
```

4) console.warn([data][, ...]) : This method is similar to console.error() method.

- Consider following example – in which using the **fs** module we open a text file. If the file does not open successfully then console.warn() method displays the warning message.

console4.js

```
var fs = require('fs');
fs.open("input.dat","r",function(err,fs){
    if(err)
        console.warn(err);
    else
        console.log("File opening successfull!!!");
})
```

4.3 : Node.JS Events

Q.3 Explain NodeJS events with example.

☞ [SPPU : June-22, Marks 6]

Ans. : • Event-driven programming is a programming paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs/threads.

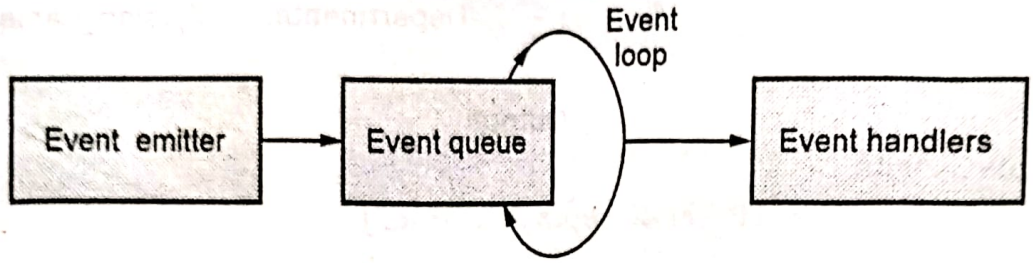


Fig. Q.3.1 Event driven architecture in node.js

What is Event ?

- Every time when user interacts with the webpage, event occurs when user clicks mouse button or presses some key on the keyboard.
- When events are triggered, some functions associated with these events get executed.
- Event driven programming makes use of the following concepts –

i) When some events occur, an event handler function is called. This is basically a call back function.

ii) The main program listens every event getting triggered and calls the associated event handler for that event.

Concept of Event Emitter :

- The **Eventmitter** is a module that facilitates communication between objects.
- The emitter objects performs following tasks -

1) It emits named events.

2) Registers and unregisters listener functions.

- Basically **EventEmitter** is a class which can be used to raise and handle custom events.

Steps for using event handling in Node.js

Step 1 : import the 'events' module and create an instance of it using following code –

```
var events = require('events');
```

Step 2 : Then create an instance of **EventEmitter()** class. This instance is created because we have to call on() and emit() functions using this instance.

Step 3 : Then define a callback function which should be called on occurrence of event. This function is also called as event listener funciton.

Step 4 : Then register the event using **on()** function. To this function name of the event and callback function is passed as arguments.

Step 5 : Finally raise the event using **emit()** function.

Example Code

eventDemo1.js

```
// get the reference of EventEmitter class of events module
var events = require('events');

//create an object of EventEmitter class by using above reference
var em = new events.EventEmitter();

var myfunction = function() {
    console.log("When event occurs, My Function is called")
}

//Bind FirstEvent with myfunction
em.on('FirstEvent', myfunction);

// Raising FirstEvent
em.emit('FirstEvent');
```

Output

```
D:\NodeJSExamples>node eventDemo1.js
```

When event occurs, My Function is called.

Q.4 Explain different phases of event loop in NodeJS.

Ans. : • Various phases of event loop are -

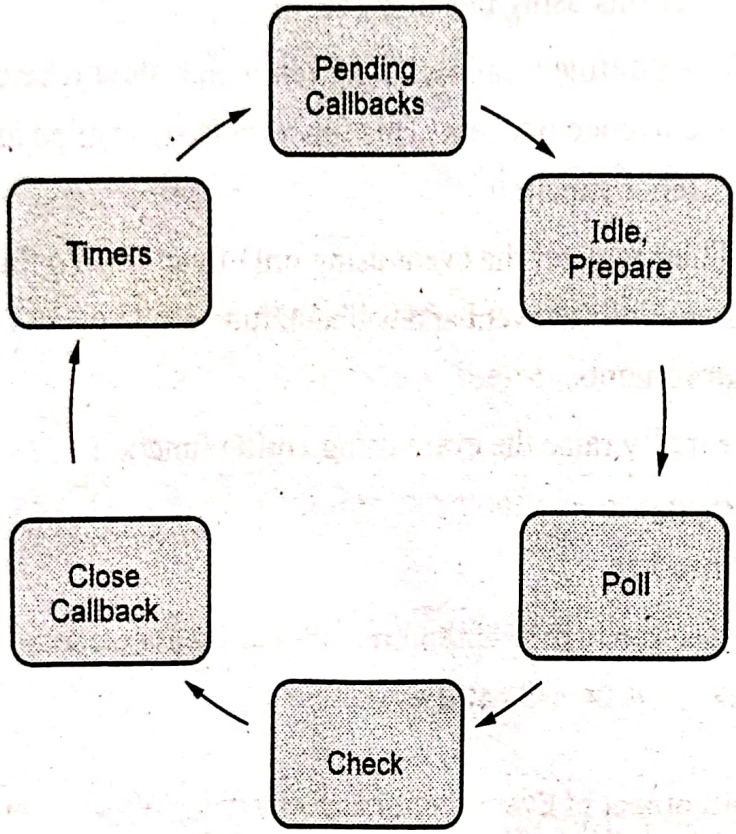


Fig. Q.4.1

1) **Timers :** This is the first phase in the event loop. It finds expired timers in every iteration (also known as Tick) and executes the timers callbacks created by setTimout and setInterval.
2) **Pending callbacks :** It handles I/O callbacks deferred to the next iteration, such as handling TCP socket connection error.
3) **Idle, prepare :** It is used internally.
4) **Poll :** The Poll phase calculates the blocking time in every iteration to handle I/O callbacks.
5) **Check :** This phase handles the callbacks scheduled by setImmediate(), and the callbacks will be executed once the Poll phase becomes idle.
6) **Close callback :** This phase handles callbacks if a socket or handle is closed suddenly and the 'close' event will be emitted.

4.4 : Node.JS Functions

Q.5 Explain callbacks in NodeJS with suitable example.

☞ [SPPU : June-22, Dec.-22, Marks 6]

Ans. : • Callbacks is a function which is usually passed as an argument to another function and it is usually invoked after some kind of event.

- The callback function is passed as an argument to another function. When particular event occurs then only it is invoked.
- By the time callback function is executing another task gets executed.
- Hence execution of callbacks is asynchronous execution.

Example Code

Step 1 : Create a simple text file named "myfile.txt" as follows –

myfile.txt

```
How are you?
Jack and Jill went up the hill,
to fetch a glass of water
```

Step 2 : Create a JavaScript file(.js) that contains the callback function

callbackDemo.js

```
var fs = require("fs");
console.log("Serving User1")

fs.readFile('myfile.txt', function (err, data) {
   if (err) return console.error(err);
   console.log("*** Contents of the File are...***");
   console.log(data.toString());
});
console.log("Serving User2");
console.log("Serving User3");
console.log("Good Bye!!!");
```

Output

```
D:\NodeJSExamples>node callbackDemo.js
Serving User1
Serving User2
Serving User3
Good Bye!!!
*** Contents of the File are...***
How are you?
Jack and Jill went up the hill,
to fetch a glass of water
```

Program Explanation : In above program,

1) We have to read a file named **myfile.txt**, but while reading a file it should not block the execution of other statements, hence a call back function is called in which the file is read.
2) By the time file gets read, the remaining two console functions indicating "Serving user 1" and "Serving user 2" are not blocked, rather they are executed, once the contents are read in the buffer, then those contents of the file are displayed on the console.

4.5 : Node.JS Built- in Modules

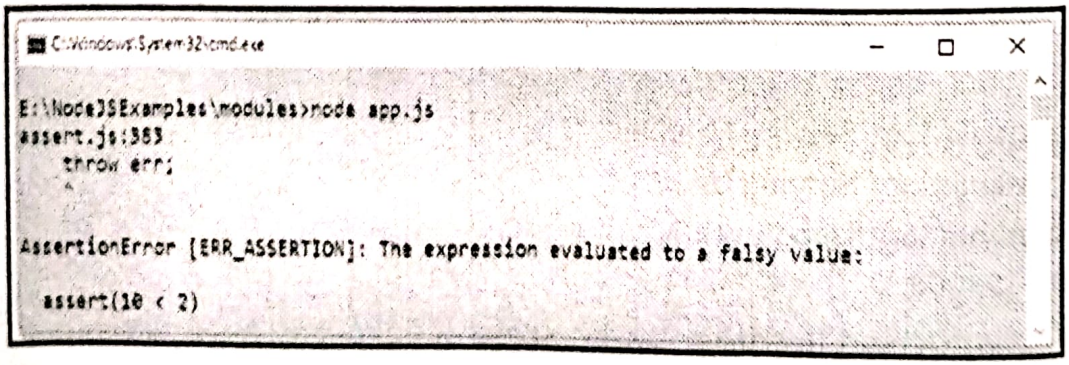
Q.6 State and explain any three built in modules in node.js.

Ans. : (1) assert : This module is used during the testing of expressions. If an expression evaluates to 0 or false, an error is thrown and program gets terminated.

app.js

```
var assert = require('assert');
assert(10 < 2);
```

The output is as follows -



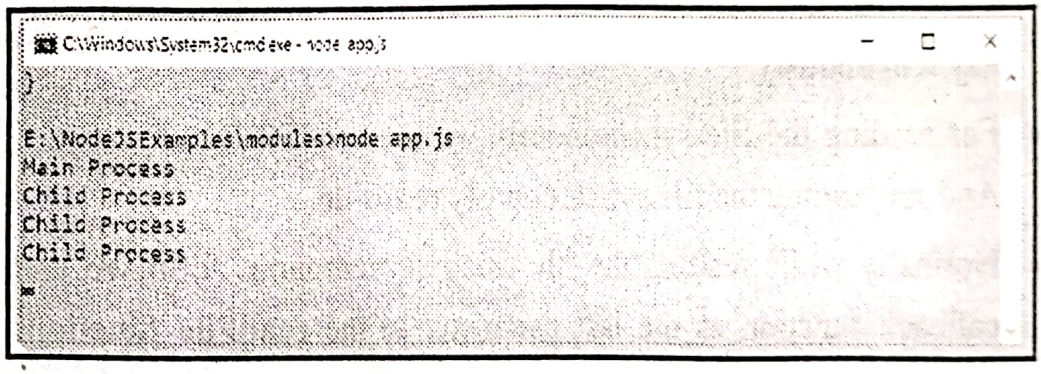
- Note that as the 10 is not less than 2, the above code will raise the error named **AssertionError**

(2) cluster : The cluster module helps in creating a child process. Both the main and child processes can run simultaneously and share the same server port. For example -

app.js

```
var cluster = require('cluster');
if (cluster.isWorker) {
  console.log('Child Process');
} else {
      console.log('Main Process');
      cluster.fork();
      cluster.fork();
      cluster.fork();
}
```

The output is as follows -



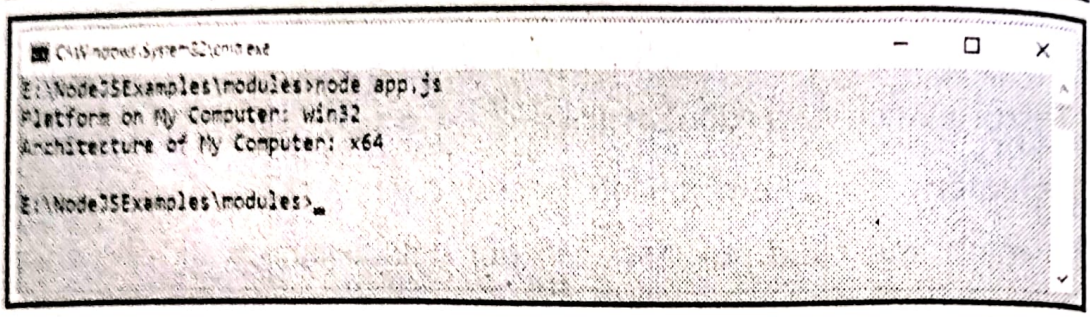
(3) os : The os module provides the information about operating system of your computer.

app.js

```
var os = require('os');
console.log("Platform on My Computer: " + os.platform());
console.log("Architecture of My Computer: " + os.arch());
```

The output is as follows -

```
node app.js
```



4.6 : File System

Q.7 With necessary source code, explain how to use file I/O operations in node.js.

Ans. : File system fs is implemented in the node.js using require function.

```
var fs = require('fs');
```

- The common operations used with the file system module are -

1. Read Operation

- The read file operation can be performed **synchronously** or **asynchronously.**
- For reading the file synchronously we use **readFileSync** method. And for reading the file synchronously **readFile.**
- Normally while reading the file using asynchronous form we use callback function as the last parameter to the **readFile** function. The form of this callback function is,

```
function(err, data)
{
   //function body
}
```

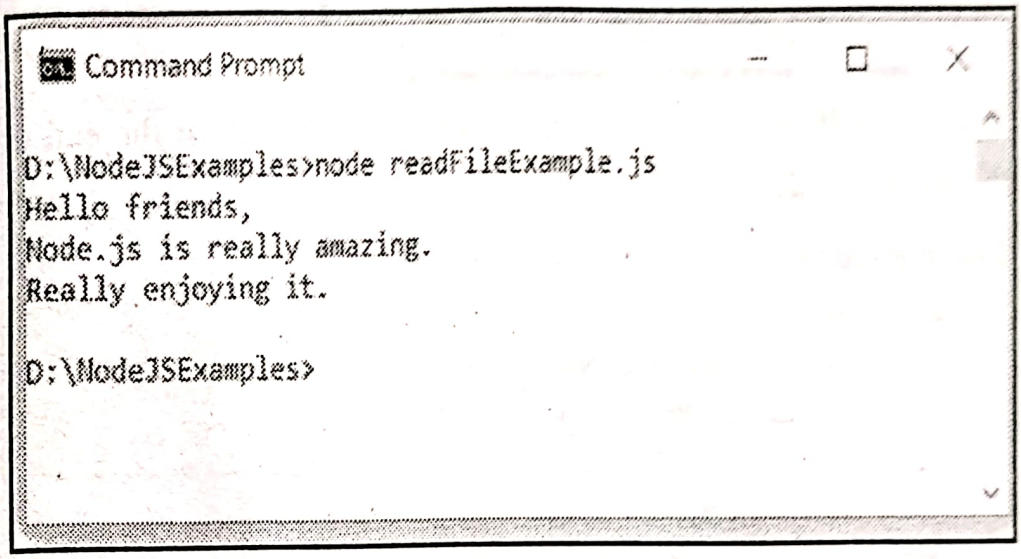
Example code

```
var fs = require("fs");
fs.readFile('input.txt',function(err,data){
    if(err)
        console.log(err);
    console.log(data.toString());
});
```

input.txt

```
Hello friends,
Node.js is really amazing.
Really enjoying it.
```

Output



2. Create File Operation

We can create an empty file using the command **open()**

Syntax

```
open(path,flags[,mode],callback)
```

where

path : It is the path of the filename.

flag : It indicates, the behaviour on opening a file. That means "r" is open file for reading, "w" is for writing, "rs" is for synchronous mode, "r+" or "w+" means open a file for both read and write purpose and so on.

mode : It indicates readable or writable mode.

Callback : It is basically a function which has two arguments(err,data)

Example Code

```
var fs = require("fs");
fs.open('myfile.txt', 'w',function(err,file){
    if(err)
```

```
        console.log(err)
    console.log('File created!!!');
});
```

3. Update / Write File Operation

For writing to the file two operations are used most commonly -

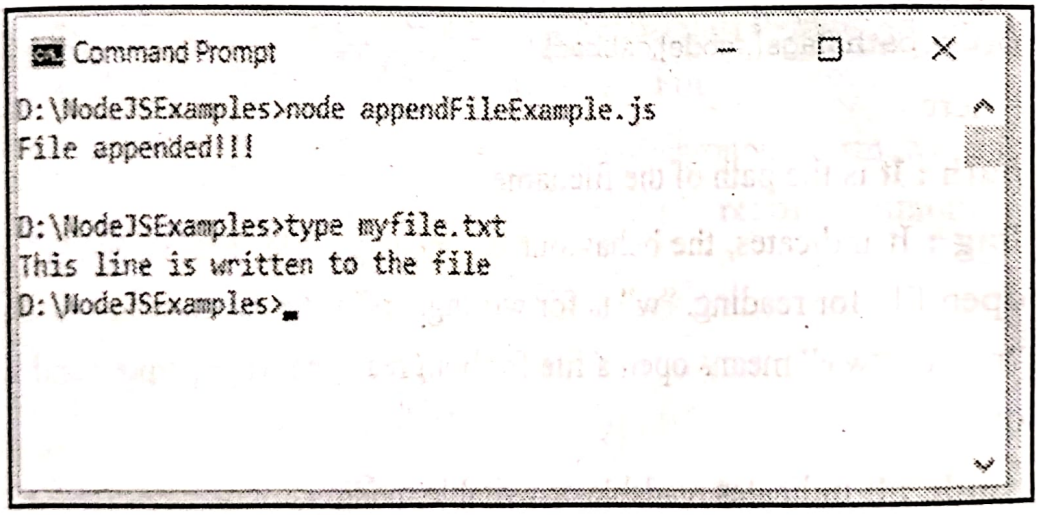
1. appendFile() and 2. writeFile()

The **appendFile()** method is used to write the contents at the end of the specified file.

Example Code

```
var fs = require("fs");
str = "This line is written to the file";
fs.appendFile('myfile.txt', str,function(err){
    if(err)
        console.log(err)
    console.log('File appended!!!');
});
```

Output



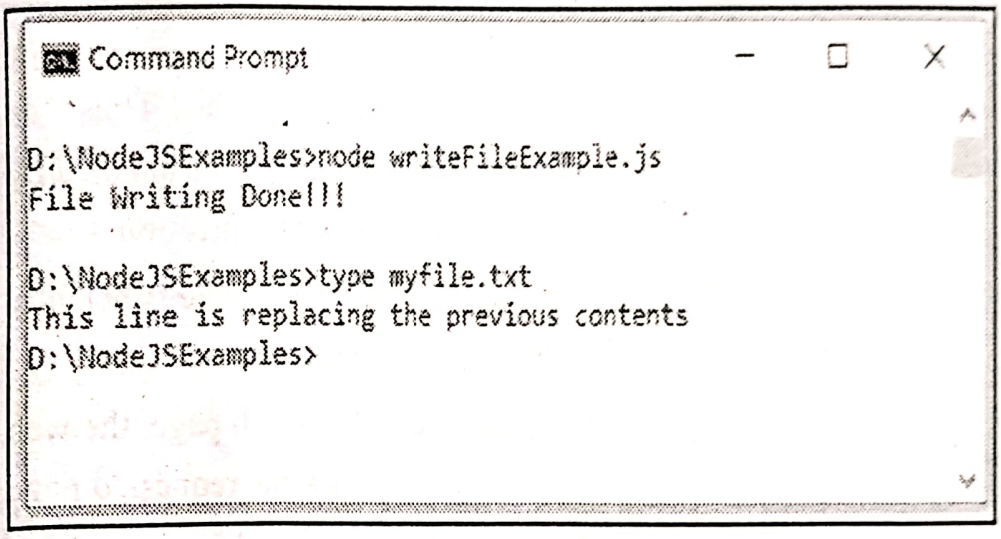
The **writeFile()** method is used to write the contents to the file

Example Code

```
var fs = require("fs");
str = "This line is replacing the previous contents";
fs.writeFile('myfile.txt', str,function(err){
    if(err)
```

```
        console.log(err)
    console.log('File Writing Done!!!');
});
```

Output



4.7 : NPM

Q.8 Explain the concept of NPM.

Ans. : • A package in Node.js contains all the files you need for a module.

- Modules are JavaScript libraries you can include in your project.
- The NPM stands for node package manager.
- NPM consists of two main parts :

1) a **CLI (command-line interface)** tool for publishing and downloading packages,
2) an **online repository** that hosts JavaScript packages.

- NPM gets installed along with the installation of node.js.

4.8 : Create HTTP Server

Q.9 Explain how a web module is controlled as a web server using Http in Node.js.

Ans. :

- The web module is mainly controlled by a web server that handles the requests and provides the responses. These requests and responses are handled by HTTP(Hyper Text Transfer Protocol) protocol.
- The user submits the request using web browser. Thus the communication between web server and web browser is handled by HTTP protocol. Following steps explain this communication -
  - When user submits a request for a web page, he/she is actually demanding for a page present on the web server.
  - When web browser submits the request for a web page, the web server responds this request by sending back the requested page to the web browser of the client's machine.

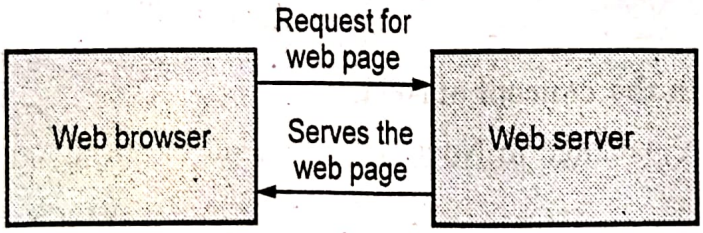


Fig. Q.9.1 Client-Server communication

Step 1 : Web client requests for the desired web page by providing the IP address of the website.

Step 2 : The web server locates the desired web page on the website and responds by sending back the requested page. If the page doesn't exist, it will send back the appropriate error page.

Step 3 : The web browser receives the page and renders it as required.

Example Code

- Node.js has a built in module named **http** which allows us to transfer data over the Hyper Text Transfer Protocol(HTTP).

- A web server using **http** module, is as follows -

WebModuleExample.js

```
var http = require("http");
var server = http.createServer(function(request,response) {
    response.writeHead(200,{"Content-Type": "text/plain"});
    response.end("Welcome to this Web Module Application!!!");
});
server.listen(8082);
console.log("Server is running on port 8082 port...");
```

4.9 : Create Socket Server

Q.10 What is socket ? Write a client server communication in node.js using socket programming.

Ans. : • **Socket :** A socket is basically an **endpoint** of a two-way communication link between two programs running on the network. Typically these programs are Server program and Client program.

- Thus socket is OS-controlled interface into which the applications can send or receive messages to and fro from another application.

Client Server Communication in node.js : • **Server** is a device which has resources and from which the services can be obtained. For example : there are various types of servers such as web server which is for storing the web pages, there are print servers for managing the printer services or there are database servers which store the databases.

- **Client** is a device which wants to get service from particular server.
- First of all server starts and gets ready to receive the client connections, The server-client communications occurs in following steps

Sr. No.	Client	Server
1.	Creates TCP socket.	Creates TCP socket.
2.	Client initiates the communication. i) Communicate.	Creates another socket for listening client. i) Accept message from client and repeatedly communicate.
3.	Close the connection.	Close the connection.

- We have two build to node.js programs - **One for server** and other for **client.**

TCP Server Program

- To build the TCP server program we have to follow the steps as given below -

Step 1 : We use "net" module by adding following line at the beginning of both the programs -

```
var net = require("net");
```

Step 2 : Now we use the **createServer()** method for creating the stream based TCP server. For that purpose we have to add following code in the program.

```
var server = net.createServer();
```

Step 3 : We use the event handler **on** for the server when any client gets connected to it. In this event handling code we use the **socket** class. The attributes **remoteAddress** and **remotePort** will return the remote address and port number of the client which is connected currently to the server.

Step 4 : Then using socket instance we can handle three events such as **data,close** and **error.**

- For **data** event, the data received from the client is displayed on the console using **cosole.log** method.
- For **close** event, the connection closing message is displayed to the client.

- If any error occurs during establishment of connection, the **error** event is fired, and error message is displayed on the console.

Step 5 : During this entire, client server communication Server must be in listening mode. This can be done using **listen** method. To this method, the port number is passed. The **same port number** must be used in the client program, so that both the server and client can communicate on the same port number.

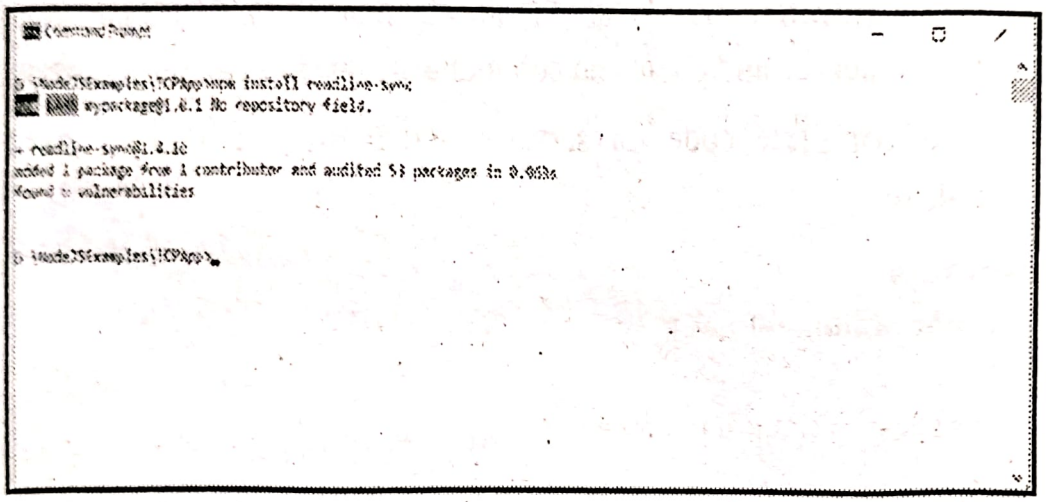
- The complete code for server program using TCP is as given below -

server.js

```
var net = require("net");

var server = net.createServer();

server.on("connection",function(socket){
    var clientAddress = socket.remoteAddress+":"+socket.remotePort;
    console.log("Connection with the client %s is
    established!!!",clientAddress);

    socket.on("data",function(mydata){
        console.log("\n Data received from client is:
        %s",mydata.toString());
        //sever sending data to the client
        socket.write("I am fine");
    });
socket.once("close",function(){
        console.log("Connection with %s is closed!!!",clientAddress);
});
socket.on("error",function(err){
cosole.log("Connection error %s",err.message);
});
});
server.listen(8082, function(){
    console.log("Server is listening to %j",server.address());
});
```

TCP Client Program

- Following is a client program, in which we are getting some input from the user through command prompt and sending that data to the server. For reading the input from keyboard we need to install **readline-sync** module. For installation of this module, open the command prompt. The installation of this module is as follows -



- To build the TCP client program we have to follow the steps as given below -

Step 1 : We use "net" module by adding following line at the beginning of the client program.

```
var net = require("net");
```

Step 2 : As our server is running on port number 8082 and the **localhost** as the host, the client must run on the same for establishing the communication between server and the client. The instance of socket class is created and it is named as var **client.**

```
const client = new net.Socket();
```

Step 3 : Now it is possible to establish a connection with the server using **connect** method. For establishing the connection we should pass the same hostname and port number as parameter.

Step 4 : Client can send the data to the server using **write** method. Here we are getting the data when user enters some data through keyboard. For getting the data from the console, we use the **readline-sync** module's **question** method. The required code for this operation is -

```
var read_line = require("readline-sync");

var mydata = read_line.question("Enter some data for sending it to server: ")
    client.write(mydata);
```

Step 5 : Two more events can be handled by the client program those are – **data** and **end**.

- On the **data** event, the data received from the server is displayed on the console window, using **console.log**

```
client.on("data",function(d){
    //body
});
```

- On the **end** event, the we can acknowledge the server than now client is going to end.

```
    client.on("end",function(){
        //body
    });
```

The complete code for client program using TCP is as given below -

client.js

```
var net = require('net');
var read_line = require("readline-sync");

const PORT = 8082;
const HOST = 'localhost';
// create a TCP client
const client = new net.Socket();

client.connect(PORT,HOST,function(){
    console.log("Connection has been established with server");
    //client sending data to server
```

```
    var mydata = read_line.question("Enter some data for sending it to
server: ")
    client.write(mydata);

});
client.on("data",function(d){
    console.log("Data received from server is: %s",d);
    client.end();
});
client.on("end",function(){
    console.log("Request is ended!!!");
});
```

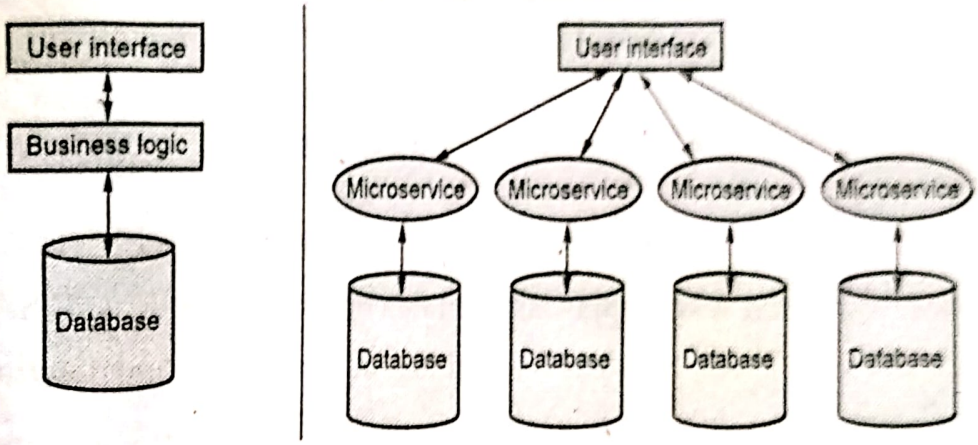
4.10 : Microservices - PM2

Q.11 Write a short note on PM2- microservices.

☞ **[SPPU : Dec.-22, Marks 5]**

Ans. :

- Microservices are architectural approaches which allows the developers to compartmentalize individual components of a larger application infrastructure. That means each component in the application can work independently.
- The developers can upgrade or modify the components without impacting the larger applications.
- The basic idea behind the creation of microservices is that - Instead of containing everything in a single unit, the microservices-based application is broken down into smaller, lightweight pieces based on a logical construct. The application consists of independent small (micro-) services, and when we deploy or scale the app, individual services get distributed within a set. Refer Fig. Q.11.1.
- PM2 stands for Process Manager 2. It is a versatile process manager written in Node.js.



Monolitic architecture Microservices architecture

Fig. Q.11.1 Concept of microservice architecture

- It is a free open source, efficient and cross platform process manager with **built in load balancer**.
- It is useful for real time app monitoring, efficient management of microservices and shutdown of applications.

Features of PM2

1) **Restarting after crashes :** PM2 allows to run the process running even after crashing.
2) **Restart persistence :** PM2 remembers all the running processes and restart them after booting.
3) **Remote monitoring and management :** With PM2 it is possible to keep track of remotely running processes and these processes can be easily managed.
4) **Performance :** Node.js event driven I/O model along with PM2 microservice strategy can handle an extreme amount of load with lesser response time.
5) **Built-in clustering :** PM2 handles all the logic internally so there is no need to make any changes in the code.
6) **Log management :** PM2 has built-in log management. It collects log data from your applications and writes it down into a single file.

Part II : ExpressJS

4.11 : Introduction to ExpressJS

Q.12 What is ExpressJS ? Give the features of it.

Ans. : • Express is web application framework used in node.js. It has rich set of features that help in building flexible web and mobile applications.

Features of Express

1) **Middleware :** Middleware is a part of program, that accesses to the database and respond to client requests.
2) **Routing :** Express JS provides a routing mechanism so that it is possible to reach to different web pages using the URLs.
3) **Faster Server Side Development :** It is very convenient to develop server side development using Express JS.
4) **Debugging :** ExpressJs makes debugging easier by providing the debugging mechanism.

4.12 : Configure Routes

Q.13 Write a code using ExpressJs to illustrate the concept of routes.

☞ **[SPPU : June-22, Marks 5]**

Ans. : • Routing is a manner by which an application responds to a client's request based on particular endpoint. The sample endpoints are,

localhost:8082/aboutus

localhost:8082/contact

localhost:8082/home

- There are various types of requests such as **GET, POST, PUT or DELETE.** While handling these requests in your express

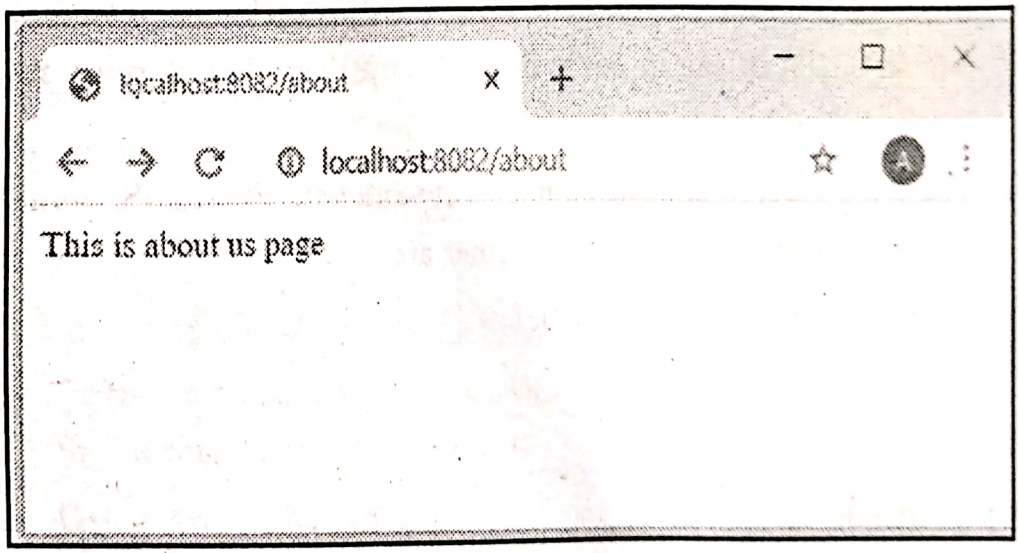
application, the request and response parameters are passed. These parameters can be used along with the **send** method.

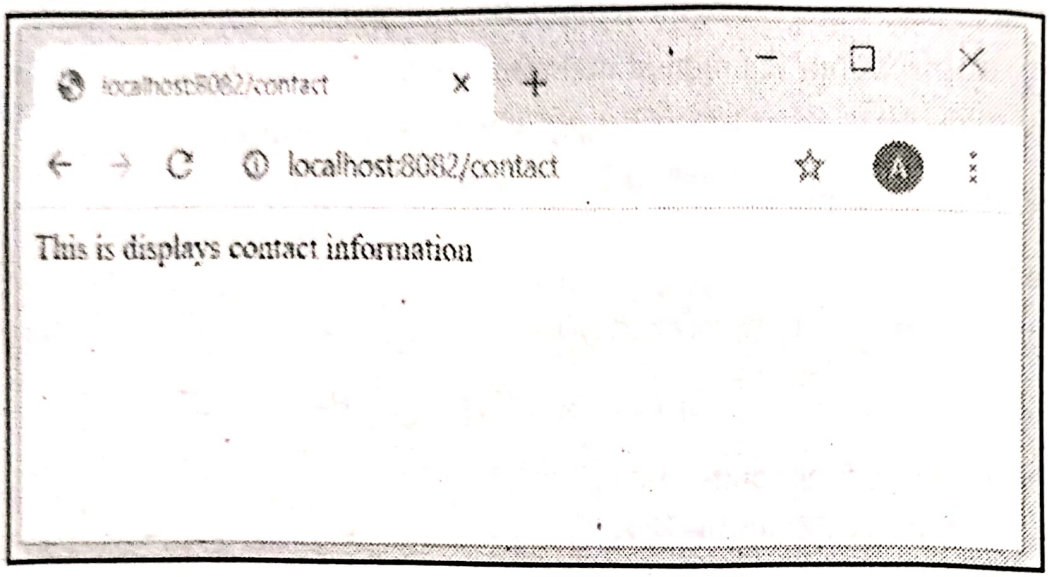
- For example -

app.js

```
var express = require('express');
var app = express();

app.get('/', function(req,res) {
    res.send("Welcome User!!!");
});
//routing to 'about us' page
app.get('/about', function(req,res) {
    res.send("This is about us page");
});
//routing to 'contact' page
app.get('/contact', function(req,res) {
    res.send("This is displays contact information");
});

var server = app.listen(8082, function(){
    console.log("Server started!");
});
```

Output





4.13 : Template Engines

Q.14 What is template engine ? How create and use it using expressJS ?

Ans. :

- Template engine is a package that renders the data or values in HTML pages.
- The most important feature of template engine is its **ability to inject data** from server into an HTML template and then send the final html page to the client side.
- Basically with the help of template engines a variable is created in our server(let us say in our expressJS script) and then at run time inserted in an html template.
- There are several template engines that work with ExpressJS. Some of the popularly used template engines are -
  - pug(Formerly known as jade)
  - mustache
  - dust
  - handlebars
  - ejs

- For demonstration of template engines, we use the template engine named **ejs**.

The code for **index.js** file is as follows –

index.js

```
var express = require('express')
var app = express()

app.set('view engine','ejs');

app.get('/', function (req, res) {
 res.send('<h1>Welcome User</h1>')
})

var customers = {
 1:"Supriya",
 2:"Siddharth",
 3:"Aniket",
 4:"Swati",
 5:"Devendra"
}
app.get('/customers/:id',function(req,res){
 res.render('customers',{custname:customers[req.params.id]})
})
app.listen(8082)
```

Calling the template engine named 'ejs'

This is a default home page which will display simply the welcome message

Create json object which contains customer's id and name

This will send the value in the variable **custname** to template engine rendering page. We will create this page in the next step

Script Explanation :

- Once we have installed EJS, we can call it into your Express app using following code,

```
app.set('view engine','ejs');
```

- Then we create a json object which will store the id and name of the customers.
- Using **res.render** function we will pass the names of the customers to the **customers.ejb** file

customers.ejb

```
<html>
  <head>
    <title>Template Engine Demo</title>
  </head>
  <body>
    <h2>Customer's Name is <%=custname %></h2>
  </body>
</html>
```

4.14 : ExpressJS as Middleware

Q.15 Explain ExpressJS as middleware with example.

☞ [SPPU : June-22, Marks 5]

Ans. : • Middleware comes in the middle of request and response cycles of Node.js execution.

- Middleware functions are functions that have access to the request object (req), the response object (res), and the next function in the application's request-response cycle.
- The **next** function is a function in the Express router which, when invoked, executes the middleware succeeding the current middleware.
- Middleware functions can perform the following tasks :
  - Execute any code.
  - Make changes to the request and the response objects.
  - End the request-response cycle.
  - Call the next middleware in the stack.
- To load the middleware function, call **app.use()**, specifying the middleware function.

index.js

```
var express = require('express')
ar app = express()
```

```
//creating middleware function
var myLogfunction = function (req, res, next) {
    let date_obj = new Date();
  console.log('Displaying contents on browser at
'+date_obj.getDate()+'-
+(date_obj.getMonth()+1)+'-'+date_obj.getFullYear())
  next()
}
//using middleware function
app.use(myLogfunction)

app.get('/', function (req, res) {
  res.send('<h1>Welcome User</h1>')
})

app.listen(8082)
```

4.15 : Serving Static Files

Q.16 Explain how to serve static file in Node.js.

Ans. : • Static files can be simple html, css or JavaScript files. It is possible to invoke those files using expressJs code.

- For serving the static file, expressJS makes use of **app.use()** method.
- The **app.use()** method mounts the middleware **express.static** for every request. The **express.static** middleware is responsible for serving the static assets of an Express.js application. Inside the **express.static** method the path for the desired static file must be specified.

Demo Example

- In this example, we will serve the static file .html. The elements in this static html file are styled using external .css file. We will write expressJS code which will serve the static html named **index.html**.

index.html

```
<html>
  <head>
  <title>Serving Static File</title>
  </head>
  <link rel ="stylesheet" href="css/style.css">
  <body>
    <h1>Welcome User</h1>
    <img src="flower.jpeg"/>
  </body>
  </html>
```

style.css

```
h1 {
  background-color: khaki;
  color: red;
  padding: 10px;
  border:solid;
  text-align: center;
  border-color: black;
}
img {
  display: block;
  margin: 0 auto;
}
```

This will display the image in a block and will be aligned at center

index.js

```
const path = require('path');
var express = require('express');
var app = express();
const filePath = path.join(__dirname,"../public");
app.use(express.static(filePath));

var server = app.listen(8082, function(){
    console.log("Server started!");
});
```

4.16 : REST HTTP Method APIs

Q.17 Write and explain a simple application using REST HTTP method APIs in node.js.

Ans. : • REST stands for Representational State Transfer. It is a set of rules that developers follow while creating their API.

- Each URL made by the client is a **request** and data sent back to the client is treated as response.

app.js

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

url = 'mongodb://localhost/EmployeeDB';

const app = express();
mongoose.connect(url, {useNewUrlParser:true})

const con = mongoose.connection  //getting the connection object

con.on('open',() => { //on opening the connection, connecting with
database
    console.log('Connected to Database')
})

app.use(express.json())
const employeeRouter = require('./routes/employees')  //initial
endpoint
app.use('/employees',employeeRouter)
app.listen(8082, () =>{
    console.log("Server Started!!!")
} )
```

URL for the database name

Connecting to MongoDB database using Mongoose package

- Create a file named **employees.js**. This file will handle the GET and POST requests of REST API. Using POST request we can create the API by inserting the data into the database. The GET request will retrieve and display the data from the database. Thus routing of GET and POST requests is done in this file.

employees.js

```
const express = require('express');
const router = express.Router();

const Employee = require('../models/employee');

router.get('/', async(req,res) =>{
    try {
        const employees = await Employee.find()
        res.json(employees)
    }
    catch(err) {
        res.send('Error is: '+err)
    }
})

router.post('/', async(req,res) => {
    const employee = new Employee({
        name: req.body.name,
        designation: req.body.designation
    })
```

This is a router program that routes on receiving the particular type of request

Handling GET command issued by the client, and using find(), displaying data present in the Employee database

Handling the POST request. The data is received here from client. Hence we use **req.body** followed by name of the data field

```
    try {
        const e1 = await employee.save();
        res.json(e1);
    }
    catch(err){
        res.send('Error is: '+err)
    }
})
module.exports = router
```

4.17 : Applying Basic HTTP Authentication

Q.18 Explain how to apply basic HTTP Authentication with suitable example.

Ans. : • Basic authentication works by prompting a Web site visitor for a username and password. This method is widely used because most browsers and Web servers support it.

- **HTTP basic authentication** is a simple **challenge and response mechanism**. In this method, a server can request authentication information such as user ID and password from a client.
- The client passes the authentication information to the server in an **Authorization header.**
- If a client makes a request for which the server expects authentication information, the server sends an HTTP response with a **401 status code**, indicating an **authentication error**, and a **WWW-Authenticate** header. A browser that receives a 401 error understands that it is required to supply correct user name and password.
- Most web clients handle this response by requesting a user ID and password from the end user.

- The HTTP WWW-Authenticate header is a response-type header. The WWW-Authenticate header field for basic authentication is constructed as following :

```
res.set('WWW-Authenticate', 'Basic');
```

Demo Example

node.js

```
var express = require("express");
var app = express();
var basicAuth = require('basic-auth');

var authFunction = function (req, res, next) {
 var user = basicAuth(req);
 if (!user || !user.name || !user.pass) {
  res.set('WWW-Authenticate', 'Basic');
  res.sendStatus(401);
  return;
 }
 if (user.name === 'admin' && user.pass === 'mypassword') {
  next();
 } else {
  res.set('WWW-Authenticate', 'Basic');
  res.sendStatus(401);
  return;
 }
}
 app.get("/login", authFunction, function (req, res) {
  res.send("<h3>You are an authentic User</h3>")
});

app.listen(8082);
console.log("Server running on 8082");
```

The HTTP WWW-Authenticate header is a response-type header which is set here.

For invalid user name and password the status code 401 is set.

4.18 : Implement Session Authentication

Q.19 How to create and use cookies using node.js ?

Ans. : • Cookie is a small piece of information used to store name-value pair. This cookie data is sent to the client with server request and stored at the client's machine. Every time when user loads the web site, this cookie is sent with the request. This helps in keeping track of the user's actions.

- Following example illustrates how to create cookie, display cookie data.

app.js

```
const express = require('express');
const cookieParser = require('cookie-parser');
const app = express();
app.use(cookieParser());


app.get('/',(req,res) => {
    res.send("Cookie Demo");
});

var student = {
    rollno: "101",
    name: "Parth"
};
 //Cookie is added
app.get('/setstudent',(req,res) => {
    res.cookie("studentData",student)
    res.send("Student data added to Cookie");
});

//Display Cookie
app.get('/getstudent', (req,res) =>{
    res.send(req.cookies);

});

//server listening
app.listen(8082, ()=> {
    console.log("Server is listening at port 8082");
})
```

Part III : MongoDB

4.19 : NoSQL and MongoDB Basics

Q.20 What is NoSQL ? What is the need for it ? Enlist various feature of NoSQL.

☞ [SPPU : June-22, Marks 6]

Ans. : • NoSQL stands for not only SQL.

- It is **nontabular database system** that store data differently than relational tables. There are various types of NoSQL databases such as document, key-value, wide column and graph.
- Using NoSQL we can maintain flexible schemas and these schemas can be scaled easily with large amount of data.

Need :

The NoSQL database technology is usually adopted for following reasons -

1) The NoSQL databases are often used for handling **big data** as a part of fundamental architecture.
2) The NoSQL databases are used for storing and modelling structured, semi-structured and unstructured data.
3) For the efficient execution of database with high availability, NoSQL is used.
4) The NoSQL database is non-relational, so it scales out better than relational databases and these can be designed with web applications.
5) For easy scalability, the NoSQL is used.

Features :

1) The NoSQL does not follow any relational model.
2) It is either schema free or have relaxed schema. That means it does not require specific definition of schem

3) Multiple NoSQL databases can be executed in distributed fashion.

4) It can process both unstructured and semi-structured data.

5) The NoSQL have higher scalability.

6) It is cost effective.

7) It supports the data in the form of key-value pair, wide columns and graphs.

Q.21 What is MongoDB ? Give its features.

☞ [SPPU : Dec.-22, Marks 6]

Ans. : • MongoDB is an **open source, document based database.**

- All the modern applications require Big data, faster development and flexible deployment. This need is satisfied by the document based database like MongoDB.

Features of MongoDB

1) It is a schema-less, document based database system.

2) It provides high performance data persistence.

3) It supports multiple storage engines.

4) It has a rich query language support.

5) MongoDB provides high availability and redundancy with the help of replication. That means it creates multiple copies of the data and sends these copies to a different server so that if one server fails, then the data is retrieved from another server.

6) MongoDB provides horizontal **scalability** with the help of **sharding.** Sharding means to distribute data on multiple servers.

7) In MongoDB, every field in the document is indexed as primary or secondary. Due to which data can be searched very efficiently from the database.

4.20 : MongoDB-Node.JS Communication

Q.22 List and explain various steps for MongoDB-node.js communication.

Ans. : • For connecting Node.js with MongoDB we need **MongoClient**. This can be created using following code.

Connect.js

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
//create database
MongoClient.connect(url, function(err, db) {
 if (err)
            throw err;
 var dbo = db.db("studentDB");
    console.log("Connected with Database");
});
```

Program Explanation : In above code,

1) First of all, we have to import the module 'mongodb' using **require.** Thus we are creating MongoDB client.
2) Then specify the URL for MongoDB by means of hostname(localhost) and port number on which MongoDB is running (27017). This is a path at which we are going to create a database.
3) Then we are using **connect** method by passing the above URL. Inside this function a database object is created by a database name "studentDB".

4.21 : CRUD Operations using Node.JS

Q.23 Write code to create collection, insert data and delete data in MongoDB using NodeJS. ☞ **[SPPU : June-22, Marks 6]**

Ans. : • The CRUD operations are performed in collaboration with Node.js and MongoDB. The CRUD stands for Create, Read, Update and Delete operations.

create.js

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
//create database
MongoClient.connect(url, function(err, db) {
 if (err) throw err;
 var dbo = db.db("studentDB");
 dbo.createCollection("Student_Info",function(err,res){
    if(err)
        throw err;
    console.log("Collection Created")
    });
    db.close();
 });
```

Insertion of Data

We can insert one document or multiple document at a time.

insert.js

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
//Insert
MongoClient.connect(url, function(err, db) {
 if (err) throw err;
 var dbo = db.db("studentDB");
 var mydata = { name: "AAA", city: "Pune" };
 dbo.collection("Student_Info").insertOne(mydata,function(err,res){
 if(err)
    throw err;
        console.log("One document Inserted!!");
        db.close();
    });
});
```

Read Data

- We can read all the documents of the collection using **find** method. Following is a simple Node.js code that shows how to read the contents of the database

display.js

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
//Read
MongoClient.connect(url, function(err, db) {
 if (err) throw err;
 var dbo = db.db("studentDB");
 var cursor = dbo.collection("Student_Info").find({})
 cursor.each(function(err,doc){
    console.log(doc);
  db.close();
 });
});
```

Updating Data

We can change one or more fields of the document using **updateOne** method.

update.js

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
//Update
MongoClient.connect(url, function(err, db) {
 if (err) throw err;
 var dbo = db.db("studentDB");
 var mydata = { name: "DDD"};
 var newdata = {$set: {name:"TTT",city:"Jaypur"}}


dbo.collection("Student_Info").updateOne(mydata,newdata,function(err
,res){
 if(err)
    throw err;
        console.log("One document Updated!!");
```

```
        db.close();
    });
});
```

Deleting Data

This is the operation in which we are simply deleting the desired document. Here to delete a single record we have used **deleteOne** method. The code is as follows –

delete.js

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
//Delete
MongoClient.connect(url, function(err, db) {
 if (err) throw err;
 var dbo = db.db("studentDB");
 var mydata = { name: "TTT"};
 dbo.collection("Student_Info").deleteOne(mydata,function(err,res){
 if(err)
    throw err;
        console.log("One document Deleted!!");
        db.close();
    });
});
```

4.22 : Mongoose ODM for Middleware

Q.24 Write short note on - Mongoose ODM.

☞ [SPPU : Dec.-22, Marks 5]

Ans. : • ODM stands for Object Data Modelling.

- Mongoose is a MongoDB Object Data Modelling (ODM) tool or library designed to work in an asynchronous environment with node.js.
- It allows us to connect to the MongoDB database.
- Besides the data modelling in Node.JS Mongoose also provides a layer of CRUD features on top of MongoDB.

- It manages relationships between data, provides schema validation and is used to translate between objects in code and the representation of those objects in MongoDB.

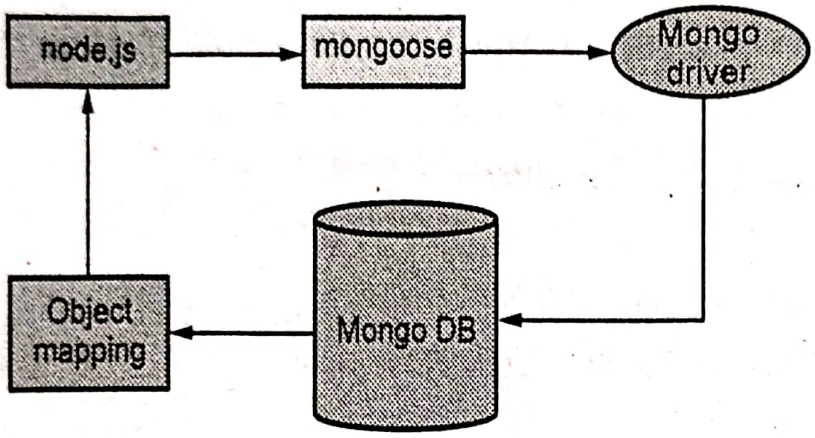


Fig. Q.24.1 Communication between node.js and MongoDB using mongoose

4.23 : Advanced MongoDB

Q.25 Give the MongoDB commands for creating, finding and deleting an index.

Ans. : • **Definition of index :** Index is a special data structure that store small part of collection's data in such a way that we can use it in querying.

- The index store the values of index fields outside the table or collection and keep track of their location in the disk.

1) Index creation

- Syntax for creating index is

```
db.<collection>.createIndex({KEY:1})
```

- The key determines the field on the basis of which the index is created. After the colon the direction of the key(1 or -1) is used to indicate ascending or descending order.
- The MongoDB will generate index names by concatenating the indexed keys with the direction of each key with underscore as

separator. For example if the index is on the field **name** and the order as **1** then the index will be created as **name_1.**

- We can also use name option to define custom index name while creating the index.
- For example -

```
>db.Student_details.createIndex({name:1},{name:"Student's Names"} )
```

2) Find Index

We can find all the available indexes in the MongoDB by using **getIndexes** method.

Syntax

```
db.<collection>.getIndexes()
```

For example –

```
>db.Student_details.getIndexes()
```

3) Drop Index

To delete an index we use **dropIndex** method.

Syntax

```
db.<collection>.dropIndex(Index Name)
```

For example

```
db.Student_details.dropIndex("Student's Names")
```

Q.26 What is replication ? Enlist the advantages of replication in database systems.

Ans. : • Replication is process of making data available across multiple data servers

- The replication is mainly used for security purpose. In case of sever failure and hardware failure the replication is used to restore the data.
- The replication in MongoDB is carried out with the help of **replica sets.**
- The replica sets are combination of various MongoDB instances having single primary node and multiple secondary nodes. The

secondary node automatically copies the changes made to primary in order to maintain the data across all servers. Refer Fig. Q.26.1.

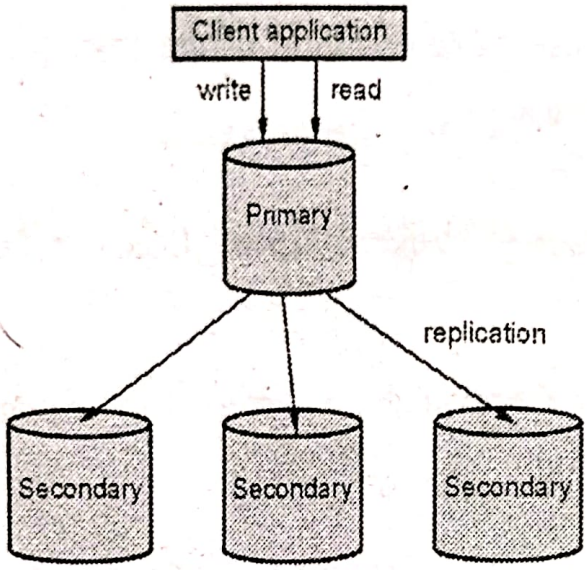


Fig. Q.26.1 Concept of replication

- If the primary node gets failed then the secondary node will take primary node's role to provide continuous availability of data. In this case the primary node selection is made by the process called **replica set elections.** In this process the most suitable secondary node will be selected as new primary node.

Benefits of Replication

1) Replication is used for data availability.
2) It is helpful for handling the situations such as hardware failure and server crash.
3) It enhances the performance as data is available across multiple machines and servers.

Q.27 What is the purpose of mapReduce ? Explain it with suitable example.

☞ **[SPPU : Dec.-22, Marks 6]**

Ans. : • **Map reduce** is a data processing programming model that helps in performing operations on large data sets and produce aggregate results.

- **Map reduce** is used for large volume of data. The syntax for map reduce is

```
>db.collection.mapReduce(
  function() {emit(key,value);}, ← map function
  function(key,values) {return reduceFunction}, { ← reduce function
    out: collection, ← the collection is created in which the result of
mapReduce can be stored
    query: document,
    sort: document,
    limit: number
  }
)
```

Where

1) **map Function :** It uses emit() function in which it takes two parameters key and value key. Here the key is on which we make groups(such as group by name, or age) and the second parameter is on which aggregation is performed like avg(), sum() is calculated on.
2) **reduce Function :** This is a function in which we perform aggregate functions like avg(), sum()
3) **out :** It will specify the collection name where the result will be stored.
4) **query :** We will pass the query to filter the resultset.
5) **sort :** It specifies the optional sort criteria.
6) **limit :** It specifies the optional maximum number of documents to be returned.

Advantages of mapReduce

1) MapReduce allows the developer to store complex result in separate collection.
2) MapReduce provides the tools to create incremental aggregation over large collections.
3) It is flexible.

Q.28 Explain the concept of sharding.

Ans. : • Sharding is a concept in MongoDB, which splits large data sets into small data sets across multiple MongoDB instances.

- It is not replication of data, but amassing different data from different machines.
- Sharding allows horizontal scaling of data stored in multiple shards or multiple collections. Logically all the shards work as one collection.
- Sharding works by creating a **cluster of MongoDB** instances which is made up of three components.
  - **Shard :** It is a a mongodb instance that contains the sharded data. The combination of multiple shards create a complete data set.
  - **Router :** This is a mongodb instance which is basically responsible for directing the commands sent by client to appropriate server.
  - **Config server :** This is mongodb instance which holds the information about various mongodb instances which hold the shard data.

END...

Unit V

5 Mobile Web Development

5.1 : Mobile-First

Q.1 What is mobile first ?

Ans. :

- Desktop, smartphone and tablet are having different screen size or layout to each other also the user experience to website or application too.
- If developer designs website or web application for desktop layout or screen size then it is very difficult to design for mobile devices such as smartphone, tablet etc.
- "Designing and developing user experience (user interface) for mobile before designing for desktop web or any other device is known as **Mobile-First** approach."

5.2 : What is Mobile Web ?

Q.2 Explain the concept of mobile web.

Ans. :

- The use of internet can be through various ways such as laptop, desktop, and smartphones (mobile).
- This internet access is done with wireless or wired connection through these devices.
- As most of the time users uses internet on their mobile devices which is easy to accessible with short period of time as well.

- Mobile device or smartphones are moveable so its flexible to access internet and web content on the go.
- "The use or access of internet and get web content thorough mobile devices is known as mobile web."

5.3 : Understanding Mobile Devices and Desktop

Q.3 What are the mobile devices ? ☞ **[SPPU : Dec.-22, Marks 3]**

Ans. :

- Mobile devices are also known as handheld computers or smartphones. There are various types of mobiles devices, some are listed below :
  - Smartphones
  - Tablets
  - e-Readers
  - Smart watches
  - Fitness bands, etc.

Q.4 What is desktop device ? ☞ **[SPPU : Dec.-22, Marks 3]**

Ans. :

- Desktop device can not be move from one place to another place easily.
- They are large and heavy.
- With this device the web content will be bigger as much as we have the large one desktop device.
- These devices are used by web designers and developers for better experience.

Q.5 What is Mobile-First and Mobile Web ? List different mobile devices. ☞ **[SPPU : June-22, Marks 6]**

Ans. : Mobile first : Refer Q. 1

Mobile web : Refer Q.2

Mobile devices : Refer Q.3

5.4 : Introduction to the jQuery Mobile Framework

Q.6 What is jQuery mobile ? ☞ [SPPU : June-22, Dec.-22, Marks 6]

Ans. :

- "**jQuery Mobile** is a unified user interface system across all popular mobile device platforms, built on the jQuery library and jQuery UI foundation."
- jQuery Mobile also known as **mobile framework** which is an open - source cross - platform framework.
- It provides touch friendly UI widgets that are designed for mobile devices.
- It has powerful theming framework to style your mobile application.
- You need to write your code once and it will work seamlessly on Android, iPhone, iPad, or any other mobile operating systems. The same code can be run on web browsers such as Google Chrome, Mozilla Firefox, Safari etc. on your desktop.

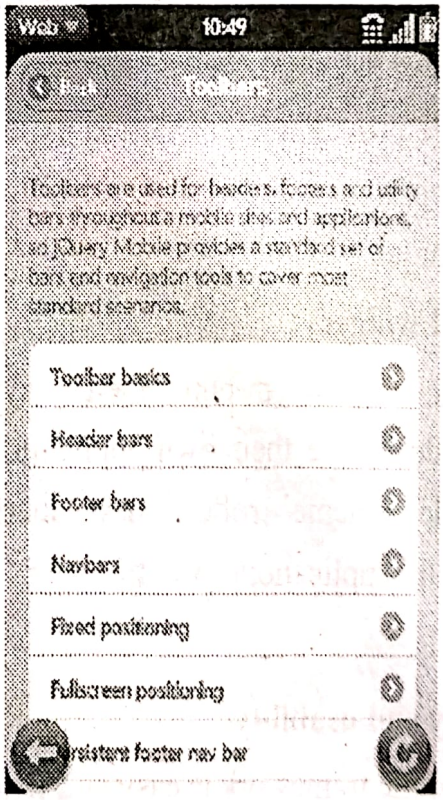


Fig. Q.6.1 A typical jQuery Mobile web app

Q.7 Is jQuery mobile framework ? If yes, explain it in brief.

Ans. : Refer Q.6.

Q.8 What are the features of jQuery mobile ?

Ans. :

- **Accessibility :**

  The jQuery Mobile used to develop for cross platform, cross device and cross browser which means from any device we can access website or web application easily.

- **Lightweight size :**

  Mobile jQuery's lightweight size (about 40 KB when minified)

- **Responsiveness :**

  The framework's full responsiveness enables the same underlying codebases to fit comfortably in different types of screens, from mobile devices to desktop-sized screens.

- **Progressive enhancement and graceful degradation :**

  Progressive enhancement is a simple but very powerful technique used in web design that defines layers of compatibility that allow any user to access the basic content, services and functionality of a website, while providing an enhanced experience for browsers with better support of standards. jQuery Mobile is totally built using this technique.

- **Theming and UI widgets :**

  jQuery Mobile has an in-built theme system that enables developers to determine their own application styling. With the jQuery Mobile Theme roller, developers can effectively customize their applications to fit their color, tastes, and preferences.

- **Great simplicity and usability :**

  The jQuery Mobile framework is easy and flexible.

Q.9 What does mean by progressive enhancement in jQuery mobile ?

Ans. : Refer Q.8.

Q.10 What is jQuery Mobile ? What are the advantages and disadvantages of jQuery Mobile ? ☞ [SPPU : June-22, Marks 6]

Ans. : jQuery mobile : Refer Q.6.

Advantages :

- Easy to learn if you are familiar to HTML, CSS, and JS.
- It has large library which enables you to perform complex functions as compared to JS.
- jQuery is cross platform framework that means you don't need to write code for different device resolutions, single codebase can work for various devices and platforms.
- It has open - source community where you can find several jQuery plugins to implement in your requirement.
- Make a website look more professional with use of effects or transitions.
- You can create custom theme using theme roller without writing the line of code.

Disadvantages :

- Limited functionality as compared to native mobile application.
- Limited to customize visual design.
- Limited options for CSS themes.
- Applications are slower on mobile device.

5.5 Set-up jQuery Mobile

Q.11 How to setup jQuery mobile ?

Ans. : To set-up jQuery mobile HTML5 document needs to include :

- The jQuery core JavaScript file
- The jQuery Mobile core JavaScript file

- The jQuery Mobile core CSS file
- The jQuery Mobile theme CSS file (optional)

There are two approaches to setup jQuery Mobile :

1. Download latest zip package and import in HTML head tag
2. Use CDN (Content Delivery Network) in HTML head tag

Q.12 What is difference between jQuery and jQuery mobile ?

☞ **[SPPU : June-22, Marks 6]**

Ans. : jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. On the other hand, jQuery Mobile is touch friendly Web Framework for Smartphones and Tablets.

Q.13 What is CDN and how it is powerful in web development ?

☞ **[SPPU : Dec.-22, Marks 6]**

Ans. : CDN stands for Content Delivery Network. Using CDN we can set up jQuery mobile.

- Using CDN is simple, you just need to copy and paste the URL to the JS of CSS external files.

```
<link rel="stylesheet" href="https://code.jquery.com/mobile /1.4.5/jquery.mobile-1.4.5.min.css" />
<script src="https://code.jquery.com/jquery-1.11.1.min.js"></script>
<script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js"></script>
```

5.6 : Pages

Q.14 What is page ? Write a code to create a page in jQuery mobile.

☞ **[SPPU : Dec.-22, Marks 6]**

Ans. : • The page is main unit of a jQuery Mobile, the page is divided into three parts : header, content, and footer.

- The main design part and mandatory section is a content and will be declared using div tag along with its role.

For example

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

```
<head>
 <meta charset="UTF-8" />
 <meta http-equiv="X-UA-Compatible" content="IE=edge" />
 <meta name="viewport" content="width=device-width,
  initial-scale=1.0" />
 <title>jQuery App</title>
 <link
  rel="stylesheet"
  href="https://code.jquery.com/mobile/1.4.5
        /jquery.mobile-1.4.5.min.css"
 />
 <script src="https://code.jquery.com/jquery-1.11.1.min.js">
  </script>
 <script src="https://code.jquery.com/mobile/1.4.5
  /jquery.mobile-1.4.5.min.js"></script>
</head>
<body>
 <div data-role="page">
  <div data-role="header">
   <h1>Page Header</h1>
  </div>
  <div data-role="content">
   <h4>Page Main Content</h4>
   <p>
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque a
atque
    placeat distinctio qui quia culpa laborum error deleniti molestiae
    quasi dolores alias accusamus dolorem esse sapiente, suscipit
    quibusdam quos.
   </p>
  </div>
  <div data-role="footer">
   <h4>Page footer</h4>
  </div>
 </div>
</body>
</html>
```

5.7 : Header and Footer

Q.15 Write a code to create a header and footer in jQuery mobile.

☞ [SPPU : June-22, Marks 6]

Ans. : Header and Footer for a page is used to design for special content of a web app such as in header we may have the title of a page or content title and navigation where in footer we can have copyright content, sitemap.

- The header text content will be in **h1** tag and footer text content will be in **h4** tag.
- The header divided into three subareas : left, title, and right where in left and right area we can place buttons and in title we can set the title of a page.
- The footer is optional.

Code : Refer Q.14.

5.8 : Content

Q.16 What is content ? Write a code to create content div in jQuery mobile.

Ans. :

- The data-role content is the main content of a page and it is design with HTML code.
- With this content div we can use framework styled controls such as buttons, lists and forms etc.

```
<!DOCTYPE html>
<html lang="en">
 <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,
     initial-scale=1.0" />
    <title>jQuery App</title>
```

```
<link
 rel="stylesheet"
 href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
 1.4.5.min.css"
/>
<script src="https://code.jquery.com/jquery-1.11.1.min.js">
</script>
<script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
1.4.5.min.js"></script>
</head>
 <body>
   <div data-role="page" id="page1">
     <div data-role="header">
       <h1>First page</h1>
     </div>
     <div data-role="content">
       <a href="" data-role="button">This is a button</a>
       <h3>List of web technologies</h3>
       <ul data-role="listview" data-theme="g">
         <li><a href="#">HTML</a></li>
         <li><a href="#">CSS</a></li>
         <li><a href="#">JavaScript</a></li>
         <li><a href="#">jQuery</a></li>
       </ul>
     </div>
     <div data-role="footer">
       <h4>Technical Publications</h4>
     </div>
   </div>
 </body>
</html>
```

5.9 : Navigation

Q.17 What is navigation ? Write a code to navigate from one page to another page in jQuery mobile.

Ans. :

- We can create navigation with button click or a tag href property to new page.
- Navigation is used for navigation from one page to another page.

- The jQuery Mobile detects browsers back button to deliver backward navigation option in android device.
- If you want to visual back button for a page you should add the following code to div having data-role as page.

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

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width,
   initial-scale=1.0" />
  <title>jQuery App</title>
  <link rel="stylesheet" href="https://code.jquery.com/mobile
   /1.4.5/jquery.mobile-1.4.5.min.css" />
  <script src="https://code.jquery.com/jquery-
   1.11.1.min.js"></script>
  <script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
   1.4.5.min.js"></script>
</head>

<body>
  <body>
    <div data-role="page" id="page1">
      <div data-role="header">
        <h1>First page</h1>
      </div>
      <div data-role="content">
        <p>This is the main content of the page.</p>
        <p>You can go to the <a href="#page2">
         second page</a>.</p>
      </div>
      <div data-role="footer">
        <h4>Technical Publications</h4>
      </div>
    </div>

    <div data-role="page" id="page2" data-title="This is the second
```

```
        page"
style="overflow:hidden;">
        <div data-role="header">
          <h1>I'm a header</h1>
          <a href="#page1" data-icon="arrow-l"
           class="ui-btn-left">Back</a>
        </div>
        <div data-role="content">
          <p>This is the main content of the second page</p>
          <p>You can go back using the header's button or using
           your browser's back button.
        </div>
        <div data-role="footer">
          <h4>Technical Publications</h4>
        </div>
      </div>
    </body>

</html>
```

5.10 : Dialog

Q.18 What is dialog and when it is useful ? Write a code to open dialog for user confirmation for book delete.

Ans. :

- Dialog is a just a page which is another way to show the web content on popup.
- The dialog has a close button at the top left or right corner.
- To open a dialog, we need to use data-rel = "dialog" inside the tag from where you want to go.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

```
  <meta name="viewport" content="width=device-width,
   initial-scale=1.0" />
  <title>jQuery App</title>
  <link rel="stylesheet" href="https://code.jquery.com/mobile/
   1.4.5/jquery.mobile-1.4.5.min.css" />
  <script src="https://code.jquery.com/jquery-
   1.11.1.min.js"></script>
  <script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
   1.4.5.min.js"></script>
</head>
<body>
  <body>
    <div data-role="page" id="page1">
      <div data-role="header">
        <h1>First page</h1>
      </div>
      <div data-role="content">
        <h2>Introduction to jQuery</h2>
        <p>Author: Mr. Mahesh Bodhgire</p>
        <p>This is for jQuery beginners to learn jQuery
         from scratch.</p>
        <p>
          <a href="#alert" data-rel="dialog" data-transition="pop"
           data-role="button">Buy
          Now</a>
        </p>
      </div>
      <div data-role="footer">
        <h4>Technical Publications</h4>
      </div>
    </div>
    <div data-role="page" id="alert" data-dialog="true">
      <div data-role="header">
        <h1>Confirmation</h1>
      </div>
      <div data-role="content">
        <h2>Are you sure you want to buy this book?</h2>
        <!-- This will be a normal page loading -->
        <a href="#page1" data-role="button">Yes, Buy Now</a>
```

```
        <!-- This is just a close link -->
        <a href="#page1" data-role="button">No,
         I will get it later</a>
      </div>
    </div>
  </body>
</html>
```

5.11 : Icons

Q.19 How to use icon in jQuery mobile app ? Write a code to use icon in button.

Ans. :

- A set of built-in icons in jQuery Mobile can be applied to buttons, collapsible, listview buttons and more.
- There is an SVG and PNG image of each icon.
- By default, the SVG icons, that look great on both SD and HD screens, are used. On platforms that don't support SVG the framework falls back to PNG icons.
- To add an icon to link buttons and button elements, use the name prefixed with ui-icon- as class.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,
     initial-scale=1.0" />
    <title>jQuery App</title>
    <link
      rel="stylesheet"
      href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
     1.4.5.min.css"
    />
    <script src="https://code.jquery.com/jquery-1.11.1.min.js">
     </script>
```

```
<script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
 1.4.5.min.js"></script>
</head>
<body>
 <div data-role="page" id="page1">
  <div data-role="header">
    <h1>Icons</h1>
  </div>
    <div data-role="content">
     <div class="ui-nodisc-icon">
          <a
      href="#"
      class="ui-btn ui-shadow ui-corner-all ui-icon-delete
      ui-btn-icon-notext ui-btn-b
      ui-btn-inline" >
      Delete</a>
          <a
      href="#"
      class="ui-btn ui-shadow ui-corner-all ui-icon-plus
      ui-btn-icon-notext ui-btn-b
      ui-btn-inline">
      Plus</a>
        <a
      href="#"
      class="ui-btn ui-shadow ui-corner-all ui-icon-minus
     ui-btn-icon-notext ui-btn-b
     ui-btn-inline">
      Minus</a>
        <a
      href="#"

     class="ui-btn ui-shadow ui-corner-all ui-icon-check
      ui-btn-icon-notext ui-btn-b
      ui-btn-inline">
     Check</a>
    </div>
    <div>
```

```
            <button
                    class="ui-btn ui-shadow ui-corner-all
                    ui-btn-icon-left ui-icon-action">
              action
            </button>

            <button
                class="ui-btn ui-shadow ui-corner-all ui-btn-icon-left
                 ui-icon-bars">
              bars
            </button>
            <button
                class="ui-btn ui-shadow ui-corner-all ui-btn-icon-left
                ui-icon-calendar">
                 calendar
            </button>
            <button
                class="ui-btn ui-shadow ui-corner-all ui-btn-icon-left
                ui-icon-clock">
              clock
            </button>
            <button
                class="ui-btn ui-shadow ui-corner-all ui-btn-icon-left
                ui-icon-delete">
                delete
              </button>
            </div>
        </div>
        <div data-role="footer">
            <h4>Technical Publications</h4>
          </div>
        </div>
    </body>
</html>
```

5.12 : Transitions

Q.20 What is transition ?

Ans. :

- The jQuery Mobile framework includes a set of CSS-based transition effects that can be applied to any page link or form submission.
- Whenever user goes from one page to another page, jQuery mobile uses smooth animation transition.
- Default transition animation is right to left.
- We can change or define new transition for each page in web app. Use **data-transition** attribute to apply animation transition for pages.

Q.21 What are the transition types in jQuery mobile ?

Ans. : The transitions in jQuery Mobile are :

1. fade
2. flip
3. slide
4. slideup
5. slidedown
6. pop

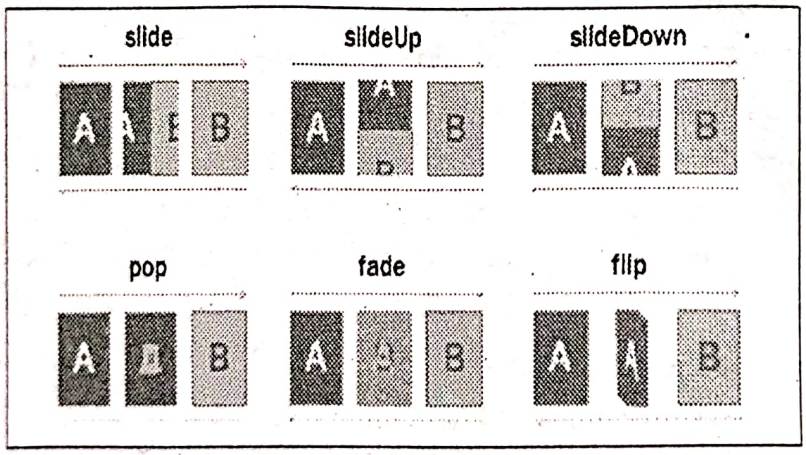


Fig. Q.21.1 jQuery tansition animation

5.13 : Theming

Q.22 How to theme web apps in jQuery mobile ?

Ans. :

- jQuery mobile has powerful theming process to show appearance of the UI.
- A theme is a group of definitions for layout, styles, and colors etc.
- Each theme has different color swatches for user interface.
- Following are some of the theme color swatches available in jQuery Mobile.

Letter	Description	Color in Default Theme
a	Highest level of visual priority	Black
b	Secondary level of visual priority	Blue
c	Baseline level	Silver
d	Alternate secondary level	Gray
e	Accent	Yellow

5.14 : Widgets

Q.23 What are widgets in jQuery mobile ?

Ans. :

- Widgets are controls in jQuery Mobile.
- They are used to design form, list, popups, tables and many more components in web apps.

Q.24 Create a button in jQuery mobile.

Ans. :

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="UTF-8" />
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width,
initial-scale=1.0" />
<title>jQuery App</title>
<link
    rel="stylesheet"
    href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
    1.4.5.min.css"
/>
<script src="https://code.jquery.com/jquery-1.11.1.min.js">
</script>
<script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
1.4.5.min.js"></script>
</head>
<body>
<div data-role="page" id="page1">
<div data-role="header">
    <h1>First page</h1>
</div>
<div data-role="content">
    <a href="#" class="ui-btn">Using Anchor</a>
    <button class="ui-btn">Using Button</button>
    <button class = "ui-btn ui-corner-all">Rounded Button
    Element</button>
    <a href = "#" class = "ui-btn ui-mini">Small Button</a>
</div>
<div data-role="footer">
    <h4>Technical Publications</h4>
</div>
</div>
</body>
</html>
```

Q.25 Create a navbar in jQuery mobile.

Ans. :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width,
initial-scale=1.0" />
<title>jQuery App</title>
<link
 rel="stylesheet"
 href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
 1.4.5.min.css"
/>
<script src="https://code.jquery.com/jquery-1.11.1.min.js">
</script>
<script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
 1.4.5.min.js"></script>
</head>
<body>
<div data-role="page" id="page1">
  <div data-role="header">
      <h1>First page</h1>
  </div>
  <div data-role="content">
     <div data-role="navbar">
      <ul>
       <li><a href="#" class="ui-btn-active">Home</a></li>
       <li><a href="#">About</a></li>
       <li><a href="#">Contact</a></li>
      </ul>
   </div>
  </div>
  <div data-role="footer">
   <h4>Technical Publications</h4>
  </div>
 </div>
</body>
</html>
```

5.15 : Layout

Q.26 How to apply grid system in jQuery mobile ? What is class name for three column grids ?

Ans. :

- The layout in jQuery Mobile is the process of making responsive UI for web apps.
- The most used grid system to create page payout through the series of rows and columns.

jQuery Mobile Grid :

- It includes **ui-grid-solo** class to create single grid in div with 100 % width.
- And if you want to use two column, three column, four column, five column grid system use class **ui-grid-a, ui-grid-b, ui-grid-c, ui-grid-d** respectively.
- Within these classes use child div classes as **ui-block-a, ui-block-b, ui-block-c, ui-block-d.**

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0" />
    <title>jQuery App</title>
   <link
      rel="stylesheet"
       href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
      1.4.5.min.css"/>
    <script src="https://code.jquery.com/jquery-
   1.11.1.min.js"></script>
    <script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
  1.4.5.min.js"></script>
```

```
</head>
<body>
  <div data-role="page" id="page1">
    <div data-role="header">
      <h1>First page</h1>
    </div>
    <div data-role="content">
      <h2>Single Column Grid</h2>
      <div class="ui-grid-solo">
       <div class="ui-block-a">
       <div class="ui-bar ui-bar-a">First Block</div>
       </div>
      </div>
      <h2>Two Column Grid</h2>
      <div class="ui-grid-a">
        <div class="ui-block-a">
        <div class="ui-bar ui-bar-a">First Block</div>
       </div>

       <div class="ui-block-b">
            <div class="ui-bar ui-bar-b">Second Block</div>
       </div>
    </div>
    <h2>Three Column Grid</h2>
    <div class="ui-grid-b">
      <div class="ui-block-a">
        <div class="ui-bar ui-bar-a">First Block</div>
       </div>

       <div class="ui-block-b">
        <div class="ui-bar ui-bar-b">Second Block</div>
       </div>

       <div class="ui-block-c">
          <div class="ui-bar ui-bar-a">Third Block</div>
        </div>
      </div>
      <h2>Four Column Grid</h2>
      <div class="ui-grid-c">
```

```
        <div class="ui-block-a">
          <div class="ui-bar ui-bar-a">First Block</div>
        </div>

        <div class="ui-block-b">
          <div class="ui-bar ui-bar-b">Second Block</div>
        </div>

        <div class="ui-block-c">
          <div class="ui-bar ui-bar-a">Third Block</div>
        </div>

        <div class="ui-block-d">
         <div class="ui-bar ui-bar-b">Fourth Block</div>
        </div>
      </div>
      <h2>Five Column Grid</h2>
      <div class="ui-grid-d">
        <div class="ui-block-a">
          <div class="ui-bar ui-bar-a">First Block</div>
        </div>

        <div class="ui-block-b">
          <div class="ui-bar ui-bar-b">Second Block</div>
        </div>

       <div class="ui-block-c">
         <div class="ui-bar ui-bar-a">Third Block</div>
       </div>
       <div class="ui-block-d">
          <div class="ui-bar ui-bar-b">Fourth Block</div>
       </div>

       <div class="ui-block-e">
         <div class="ui-bar ui-bar-a">Fourth Block</div>
       </div>
      </div>
    </div>
    <div data-role="footer">
```

```
    <h4>Technical Publications</h4>
   </div>
  </div>
 </body>
</html>
```

5.16 : Events

Q.27 Explain any 3 events in jQuery mobile with example.

☞ [SPPU : June-22, Marks 6]

Ans. :

- Events are used for interaction with user interface and trigger some action after some event.
- Events are mouse click, mouse double click, scroll page, page on load, keypress, keyup and keydown.

```
<!DOCTYPE html>
<html lang="en">
 <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,
     initial-scale=1.0" />
    <title>jQuery App</title>
     <link
       rel="stylesheet"
        href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
        1.4.5.min.css"
    />
    <script src="https://code.jquery.com/jquery-
    1.11.1.min.js"></script>
    <script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
    1.4.5.min.js"></script>
    <script>
        $(document).on("pagecreate","#page1",function() {
            $("button").on("click",function() {
                $(this).hide();
            });
```

```
        });
      </script>
</head>
<body>
  <div data-role="page" id="page1">
  <div data-role="header">
      <h1>First page</h1>
  </div>
  <div data-role="content">
     <button class = "ui-btn">Click Me to hide</button>
   </div>
   <div data-role="footer">
      <h4>Technical Publications</h4>
  </div>
 </div>
</body>
</html>
```

5.17 : Forms

Q.28 Why web apps require forms and what are the types of forms ?

Ans. :

- Forms are used to collect data from end user and send or process in background for any functionality.
- The forms are made up with multiple inputs and button widgets.
- We can design forms for login, register, contact, etc.

Q.29 Write a code to create login form.

Ans. :

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
   <meta name="viewport" content="width=device-width,
   initial-scale=1.0" />
    <title>jQuery App</title>
```

```
    <link rel="stylesheet"
href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
1.4.5.min.css" />
     <script src="https://code.jquery.com/
      jquery-1.11.1.min.js"></script>
     <script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
     1.4.5.min.js"></script>
</head>
<body>
     <div data-role="page" id="page1">
       <div data-role="header">
          <h1>Student Registration</h1>
        </div>
        <div data-role="content">
            <form>
                <label for="fname">Name</label>
                <input type="text" name="fname" id="fname"
                  placeholder="Full Name">
                <label for="email">Email</label>
                <input type="email" name="email" id="email"
                  placeholder="Email Address">
                   <label for="contact">Contact No</label>
                <input type="number" name="contact" id="contact"
                  placeholder="Contact Number">
                  <label for="date">Date of Birth</label>
               <input type="date" name="date" id="date">
               <label for="select">Select Branch</label>
               <select name="select" id="select">
                    <option value="1">IT</option>
                    <option value="2">CSE</option>
                    <option value="3">EnTC</option>
                    <option value="4">Mech</option>
                    <option value="5">Civil</option>
                   </select>
                   Gender
                   <label for="radio1">
                    <input type="radio" name="radio-choice-0"
                        id="radio1">Male</input>
                   </label>
                   <label for="radio2">
```

```
                <input type="radio" name="radio-choice-0"
                id="radio2">Female</input>
                </label>
                <label for="radio3">
                 <input type="radio" name="radio-choice-0"
                    id="radio3">Other</input>
                </label>
                <button>Register</button>
        </div>
        <div data-role="footer">
          <h4>Technical Publications</h4>
    </div>
  </div>
</body>
</html>
```

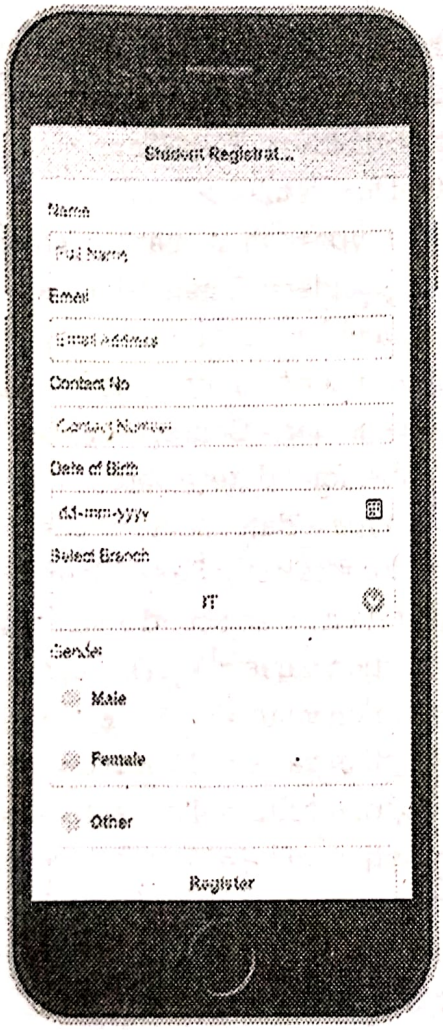


Fig. Q.29.1 Student registration form

5.18 : CSS Classes

Q.30 List any five CSS classes for jQuery mobile.

Ans. :

- jQuery Mobile framework uses different classes to design web app.
- These classes are applied to HTML elements / tags. Some of the classes are listed below :
  - ui-corner-all - To make elements in with rounded corners
  - ui-shadow - To display shadow of elements
  - ui-overlay-shadow – To display overlay shadow of elements
  - ui-mini - To make elements smaller
  - ui-btn - It shows element as button style
  - ui-icon-ICONNAME – It will show icon (ICONNAME will be different)
  - ui-bar-(a-z) - It shows the color bars such as header footer among a-z
  - ui-body-(a-z) - It shows the color body with a-z value
  - ui-grid-solo - It creates single column grid.

Q.31 What ui-overlay-shadow class will do ?

Ans. : Refer Q.30.

5.19 Data Attributes

Q.32 What is the use of data attributes ? Explain any two types of data attributes used in jQuery mobile framework.

Ans. :

- The jQuery Mobile framework uses HTML5 data - attributes to allow for markup-based initialization and configuration of widgets.
- These attributes are optional.

- To avoid naming conflicts with other plugins or frameworks that also use data- attributes

Following are some of the attributes for HTML5 elements in jQuery Mobile.

1. Button attributes

Attribute Name	Value
data-corners	true | false
data-icon	home | delete | plus | arrow-u | arrow-d | check | gear | grid | star | custom | arrow-r | arrow-l | minus | refresh | forward | back | alert | info | search
data-iconpos	left | right | top | bottom | notext
data-iconshadow	true | false
data-inline	true | false
data-shadow	true | false
data-theme	swatch letter (a-z)

2. Dialog attributes

Attribute Name	Value
data-close-btn-text	string (text for the close button, dialog only)
data-dom-cache	true | false
data-id	string (unique id for the page)
data-fullscreen	true | false (used in conjunction with fixed toolbars)
data-overlay-theme	swatch letter (a-z) - overlay theme when the page is opened in a dialog
data-theme	swatch letter (a-z)
data-title	string (title used when page is shown)

5.20 : Building a Simple Mobile Web Page

Q.33 Write a code for building a simple mobile web page of your own choice.

Ans. :

- With jQuery Mobile framework lets design a simple mobile webpage for IMDB-Movie app.
- In this webpage we are using Ajax for IMDB API call. We need to design 2 HTML page i.e. Homepage and Movie detail page. Also, we require JavaScript code to get movie details from api.

```
home.html
<!DOCTYPE html>
<html lang="en">
 <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
     <meta name="viewport" content="width=device-width,
    initial-scale=1.0" />
     <title>OMDB APP</title>
    <link
        rel="stylesheet"
         href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
         1.4.5.min.css"
    />
     <script src="https://code.jquery.com/jquery-
    1.11.1.min.js"></script>
     <script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
    1.4.5.min.js"></script>
</head>
<body>
  <div data-role="page" id="home">
    <div data-role="header" data-theme="b">
        <h1>Movie Listing</h1>
    </div>
    <div data-role="content">
     <form id="searchForm">
      <div class="ui-field-contain">
       <input
```

```
        type="text"
        name="search"
        id="search"
        placeholder="Search Movie.."
      />
     </div>
    </form>
    <br>
    <ul id="movieList" data-role="listview"></ul>
   </div>
   <div data-role="footer" style="text-align: center" data-theme="b">
    <p>OMDB App</p>
   </div>
  </div>
  <script src="./js/api.js"></script>
 </body>
</html>
```

movie.html

```
<!DOCTYPE html>
<html lang="en">
 <head>      <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0" />
    <title>OMDB APP</title>
    <link
      rel="stylesheet"
       href="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
       1.4.5.min.css"
  />
  <script src="https://code.jquery.com/jquery-
   1.11.1.min.js"></script>
  <script src="https://code.jquery.com/mobile/1.4.5/jquery.mobile-
   1.4.5.min.js"></script>
 </head>
 <body>
   <div data-role="page" id="movie">
   <div data-role="header" data-theme="b">
      <h1>Movie Details</h1>
       <a href="index.html" class="ui-btn">Back</a>
```

```
    </div>
    <div data-role="content">
      <div id="movieTop"><br /><br /></div>
      <ul id="movieDetail" data-role="listview"></ul>
    </div>
    <div data-role="footer" style="text-align: center" data-theme="b">
      <p>OMDB App</p>
    </div>
   </div>
   <script src="./js/api.js"></script>
  </body>
</html>
```

api.js

```
$(document).ready(function () {
  $("body").on("submit", "#searchForm", function (e) {
    let searchText = $("#search").val();
    getMovies(searchText);
    e.preventDefault();
  });
});

$(document).on("pagebeforeshow", "#movie", function () {
  let movieId = sessionStorage.getItem("movieId");
  getMovieDetails(movieId);
});
function movieClick(imdbID) {
  sessionStorage.setItem("movieId", imdbID);
  $.mobile.changePage("movie.html");
}
function getMovies(searchText) {
  $.ajax({
    method: "GET",
    url:
`http://www.omdbapi.com?apikey=7d19230d&s=${searchText}`,
  }).done(function (data) {
    console.log(data);
    let movies = data.Search;
    let output = "";
```

```
    $.each(movies, function (i, movie) {
     output += `
          <li>
            <a onclick="movieClick('${movie.imdbID}')" href="#">
                  <img src="${movie.Poster}">
                <h2>${movie.Title}</h2>
                  <p>Release Year: ${movie.Year}</p>
            </a>
          </li>
        `;
    });
    $("#movieList").html(output).listview("refresh");
  });
}

function getMovieDetails(movieId) {
  $.ajax({
    method: "GET",
    url: `http://www.omdbapi.com?apikey=7d19230d&i=${movieId}`,
  }).done(function (movie) {
    let movieTop = `
          <div style="text-align:center;">
            <h1>${movie.Title}</h1>
            <img src="${movie.Poster}">
          </div>
        `;
    $("#movieTop").html(movieTop);
    let movieDetails = `
          <li><strong>Rated:</strong>${movie.Rated}</li>
          <li><strong>Released:</strong>${movie.Released}</li>
          <li><strong>IMDB
Rating:</strong>${movie.imdbRating}</li>
          <li><strong>Actors:</strong>${movie.Actors}</li>
          <li><strong>Director:</strong>${movie.Director}</li>
        `;
    $("#movieDetail").html(movieDetails).listview("refresh");
  });
}
```

Unit VI

6 Web Application Deployment

6.1 : AWS Cloud

Q.1 What is cloud computing ? [SPPU : Dec.-22, Marks 3]

Ans. : • Cloud computing is one of the most usable services in IT services where we can use this service for various operations.

- We can deliver the IT infrastructure such as servers, storage, database, networking, software, analytics and intelligence over the internet which is the cloud.
- It will make faster accessibility of the IT resources for the development and deployment of any kind of project.
- It helps to lower your operating costs and run infrastructure more efficiently and scale as your business needs change.

Q.2 What are the benefits of cloud computing ?

☞ [SPPU : Dec.-22, Marks 3]

Ans. : • Cost

- Performance speed
- Productivity
- Reliability
- Security

Global scale

Q.3 What are the types of cloud computing ?

Ans. : • IaaS - Infrastructure as a service

- PaaS - Platform as a service

- SaaS - Software as a service
- Serverless computing

Q.4 What is AWS cloud ? List different services provided by it.

☞ [SPPU : June-22, Dec.-22, Marks 6]

Ans. : • AWS is Amazon Web Service which is a cloud platform offering over 200 fully-featured services from data centers globally.

- It is a leading cloud computing platform.
- With AWS cloud we can deliver the projects on cloud IT infrastructure.
- It has more functionality for storage, delivery, networking, database, software, etc.
- It is the most secure cloud platform as compared to other platform providers which are very flexible to end-users.
- AWS has the largest community of customers and developers for support.

AWS Services

- As AWS provides offers over 200 fully featured services which include storage, database, networking, software, etc. there are some important services that we should know about it.
  - AWS compute services
  - Storage
  - Database services
  - Security services
  - Analytics
  - Developer tools
  - Deployment and management
  - Mobile services
  - Internet of Things (IoT)
  - Migration

6.2 : AWS Elastic Compute

Q.5 What is EC2 service ?

Ans. : • EC2 is a virtual machine with an operating system and hardware component that you want to use.

- AWS allows us to run various virtual computers and manage them with single hardware.
- EC2 is one of the most used primary services from the AWS ecosystem.
- EC2 enables on-demand, scalable computing capacity in the AWS cloud.
- An AWS EC2 instance eliminates the up-front investment for hardware, so there is no need to maintain any rented hardware.
- EC2 enables you to build and run applications faster.
- You can use EC2 in AWS to launch as many virtual servers as you need.

Q.6 What are the EC2 types ?

Ans. : • General purpose

- Compute-optimized
- Memory-optimized
- Accelerated computing
- Storage optimized

Q.7 What is PuTTY ? **[SPPU : Dec.-22, Marks 2]**

Ans. : • PuTTY is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection. It can also connect to a serial port.

Q.8 How to connect EC2 instance with PuTTY ?

☞ **[SPPU : Dec.-22, Marks 3]**

Ans. : Step 1 : Download puttygen for creating a .ppk file as putty doesn't accept .pem file generated by AWS.

Step 2 : Convert your .pem file to .ppk file using PuttyGen. Load your .pem file generated by AWS. Then save the private key (.ppk) file.

Step 3 : Open Putty. Add you IP. Add user name.Add ppk file. Click Open. Give the ip address or the host name. Then in data section give the User name for the instance for linux its generally "ec2-user".

Step 4 : Click open.

Thus now EC2 instance is connected.

6.3 : AWS Elastic Load Balancer and its Types

Q.9 What is elastic load balancer and explain its working.

☞ **[SPPU : June-22, Marks 5]**

Ans. : • Every application needs good efficiency, performance and availability.

- The network traffic and load on a single server decrease the efficiency, performance, and availability of an application.
- Elastic load balancer or ELB is a service provided by AWS to distribute incoming traffic across multiple servers or clusters.
- The ELB increases the availability and fault tolerance of an application.
- AWS load balancer will distribute your workloads across multiple compute resources, such as a Virtual Machine or Virtual Server.
- With AWS management console we can create load balancers.

ELB Work

- ELB accepts all the traffic from the client and then routes this traffic to the target that the user wants.
- If the load balancer finds an unhealthy target, then it will stop redirecting its users there and will move with the other healthy targets until that target is declared healthy.

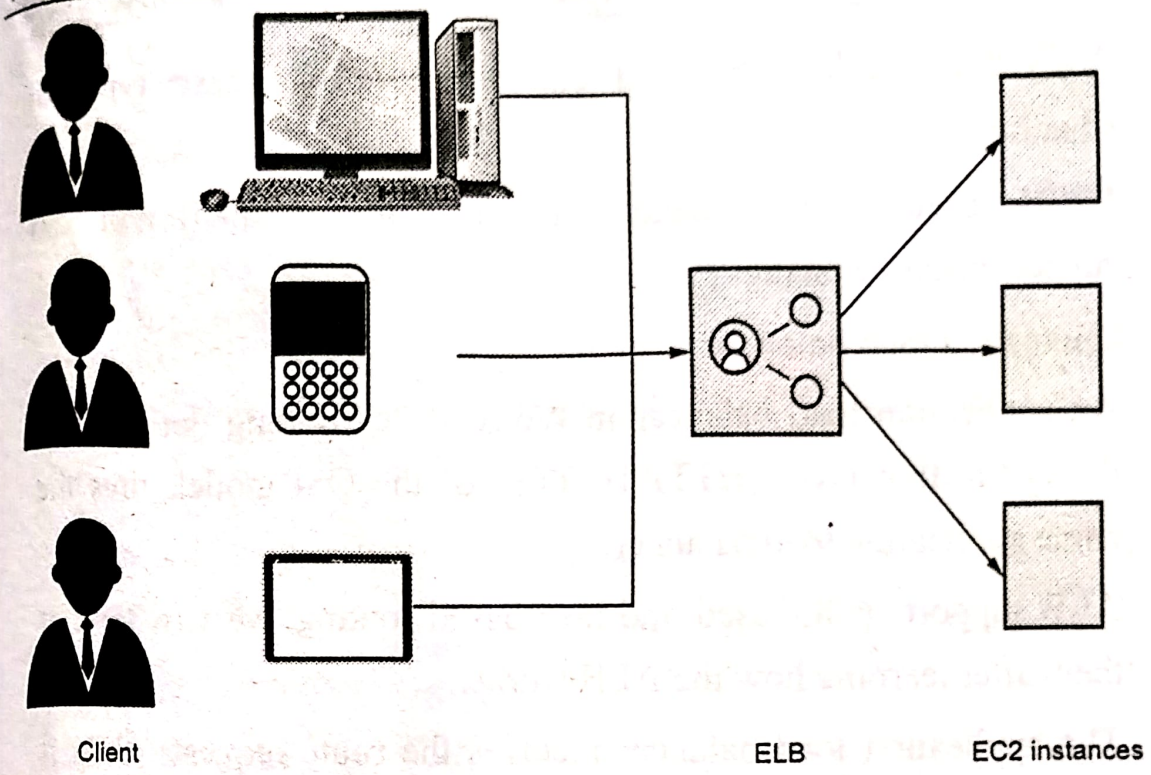


Fig. Q.9.1 ELB

Q.10 Explain in detail types of elastic load balancer.

☞ **[SPPU : June-22, Marks 6]**

Ans. : 1. Classic load balancer

- It is a traditional load balancer that is used initially.
- The classic Load balancer in AWS is used on EC2-classic instances.
- This is the previous generation's load balancer and also it doesn't allow host-based or path-based routing.
- It ends up reducing efficiency and performance in certain situations.
- It is operated on connection level as well as request level.

2. Network load balancer

- Network load balancer in AWS takes routing decisions in the transport layer (TCP/SSL) of the OSI model.
- It is mainly used for load balancing TCP traffic.
- It can handle millions of requests per second.

- AWS network load balancer can be trusted in these types of situations.
- Widely used to load balancing the TCP traffic and it will also support elastic or static IP.

3. Application load balancer

- An application load balancer in AWS makes routing decisions at the application layer (HTTP/HTTPS) of the OSI model, thus the name application load balancer.
- ALB supports path-based and host-based routing, we will look at them after learning how the ALB works.
- The application load balancer receives the route requests, then it inspects the received packets.
- Then it chooses the best target possible for the type of load and sends it to the target with the highest efficiency.

6.4 : AWS VPC and Component of VPC

Q.11 What is AWS VPC ? **☞[SPPU : Dec.-22, Marks 3]**

Ans. : • AWS provides security to the cloud and servers with its services.

- VPC or Virtual Private Cloud provides additional security levels on the AWS services that you are using.
- VPS gives you full control over routing traffic to and from your instances.
- AWS VPC is a private subsection of AWS in which you can place AWS resources such as EC2 instances and databases.
- It gives all the benefits of the traditional network that you have for your own data center.
- Resources and applications are accessed through IPv4 or IPv6 in your AWS VPC.

- There are two types of VPC :

  1. Default VPC 2. User defined VPC

Q.12 What are the advantages of VPC ?

Ans. : • EC2 Instance security group membership can be changed while it is running.

- Static IPv4 assigned to instances that persist across the start and stop.
- Access Control List (ACL) is an additional security layer to protect instances.
- Multiple IPv4 can be assigned to your instances.
- Control both inbound and outbound traffic of instances.

Q.13 What are the different components of VPC ?

☞ **[SPPU : June-22, Marks 5, Dec.-22, Marks 3]**

Ans. : • **Subnet :** A segment of a VPC's where you can place groups to isolated resources.

- **Internet gateway :** VPC side of a connection to utilize public Internet.
- **NAT gateway :** A highly available, managed Network Address Translation (NAT) service for your resources in a private subnet to access the Internet.
- **Virtual private gateway :** The Amazon VPC side of a VPN connection for secure transactions.
- **Peering connection :** To route traffic via private IP addresses between two peered VPCs.
- **VPC endpoints :** Enables private connectivity for your service in AWS without using an Internet Gateway, VPN, Network Address Translation (NAT) devices, or firewall proxies.
- **Egress-only internet gateway :** A stateful gateway that provides egress-only access for IPv6 traffic from the VPC to the Internet.

6.5 : AWS Storage

Q.14 Explain the concept of AWS storage

Ans. : • Cloud storage is a cloud computing model that stores data on the Internet through a cloud computing provider that manages and operates data storage as a service.

- It is delivered on-demand and manages your own data storage infrastructure.
- It is cost-effective for storage buying.
- AWS storage services have different provisions for highly confidential data, frequently accessed data and not so frequently accessed data.

Q.15 Explain any three AWS storage services.

☞ **[SPPU : June-22, Dec.-22, Marks 6]**

Ans. : 1. Simple Storage Service (S3)

- It is an object storage type and stores any type or size data.
- It is used for web applications, mobile applications, analytics, and backup services.
- It provides user management control for any specific requirement in the application to store the data.
- With S3 we can create, rename, delete folders with the help of a web-based file explorer.
- AWS provides 99.99999999% durability to deliver data to end-users.
- It provides 3 forms of encryption, including server-side encryption and client-side encryption.

2. Amazon S3 Glacier

- It is an object storage type.
- It is used for long-term data storage and especially for archival data.

- It also provides encryption on data for security.
- This service is used for a low retrieval rate of data in any application.
- It allows running queries and analytics on it directly.
- It provides 99 % durability.

3. Elastic Block Storage (EBS)

- It is a block storage type.
- It is like hard drive storage.
- This storage is attached to the EC2 instance and used as block storage, where we can install any operating system.
- It is available in SSD or HDD format.
- EBS volumes are network file systems.
- These volumes get automatically replicated within availability zones for better availability and durability.
- We can dynamically increase or decrease the capacity of storage.

4. Amazon EC2 Instance Storage

- It is a block storage type.
- It is used as temporary storage for EC2 instances.
- The temporary storage of data that changes frequently like buffers, queue cache, etc.
- It uses SSD for high I/O performance.
- Durability provides with replication of storage.

Q.16 What is S3 bucket ?

☞ **[SPPU : Dec.-22, Marks 6]**

Ans. : • S3 has two primary entities called buckets and objects.

- Objects are stored in buckets.
- Within this bucket, you can create multiple folders to differentiate the files.

- If you want to upload images you may create a folder named images and store it in the logical address of the file with the prefix 'images'
- You can create a maximum of 100 buckets per account.
- The bucket names should be unique.

6.6 : Deploy Website or Web Application on AWS

Q.17 List steps to deploy website on EC2.

Ans. :

Step 1 : Connect server with PuTTY

Step 2 : Update Ubuntu instance

Step 3 : Install Apache 2

Step 4 : Go back to your instances page and click on 'launch-wizard-1' under security groups

Step 5 : Go to Inbound and click on Edit inbound rules

Step 6 : Then, add an HTTP rule with Source as 'Anywhere-IPv4' and save the rule.

Step 7 : On your browser, use your public IP address, put it in the URL box and run

Step 8 : You can create your HTML page or upload your code with GitHub in the/var/www/html folder.

- With these steps, you can deploy the website on the EC2 instance.

Q.18 How to update the Ubuntu server ?

Ans. : To update the Ubuntu server following command is used

sudo apt-get update

Q.19 What is the default port no for HTTP protocol ?

Ans. : The default HTTP and HTTPS ports for the Web server are port 80 and 443, respectively.

6.7 : Launch an Application with AWS Elastic Beanstalk

Q.20 What is elastic beanstalk and enlist the advantages of using it ? [SPPU : June-22, Marks 2]

Ans. : • Elastic Beanstalk is a compute service for web applications.

- It is a pre-configured EC2 server where environment configuration is already done.
- EC2 is infrastructure-as-a-service whereas Elastic Beanstalk is platform-as-a-service.
- It makes it easier for developers to deploy and manage applications on AWS.
- In EC2 we have to create a deployment environment for application deployment like for PHP, Python, Node.JS, etc. but in Elastic Beanstalk, it provides configured server.
- It cost pay-per-use of resources so there is no additional cost for you.

AWS Elastic Beanstalk Advantages

- Developer productivity
- Cost-effective
- Easy to start
- Customization
- Management and updates

END...