MEAN STACK TECHNOLOGIES

MODULE I- HTML 5, JAVSCRIPT, NODE.JS, EXPRESS.JS AND TYPESCRIPT

(Skill Oriented Course)

Software configuration and installation

1. HTML &Javascript

x Simple editors such as Notepad or go for IDEs like Visual Studio Code(recommended), Eclipse

etc. which makes coding easier.

x And, to execute application, you can use any commonly used browser such as Google

Chrome(recommended), Mozilla Firefox etc

x Setup details: Environmental Setup for HTML5 - Viewer Page | Infosys Springboard

(onwingspan.com)

x Environment Setup: Internal - Viewer Page | Infosys Springboard (onwingspan.com)

2. Node JS

Download Node.js from the official site

Setup details :How to use Node.js - Viewer Page | Infosys Springboard (onwingspan.com)

3. Typescript

Installing TypeScript - Internal - Viewer Page | Infosys Springboard

(onwingspan.com)

Web Links:

1. https://infyspringboard.onwingspan.com/en/app/toc/lex_17739732834840810000_shared/overview

(HTML5)

2. https://infyspringboard.onwingspan.com/en/app/toc/lex_18109698366332810000_shared/overview

(Javascript)

3. https://infyspringboard.onwingspan.com/en/app/toc/lex_32407835671946760000_shared/overview

(Node.js & Express.js)

4. https://infyspringboard.onwingspan.com/en/app/toc/lex_9436233116512678000_shared/overview

(Typescript)

1.a Course Name: HTML5 - The Language

Module Name: Case-insensitivity, Platform-independency, DOCTYPE Declaration, Types of Elements, HTML Elements - Attributes, Metadata Element

Include the Metadata element in Homepage.html for providing description as "IEKart's is an online shopping website that sells goods in retail. This company deals with various categories like Electronics, Clothing, Accessories etc.

Source code:-

Types of Elements

```
<!DOCTYPE html>
<html>
   <body>
      <p> I am first paragraph.</p>
      <p> I am second paragraph.</p>
      <p>Paragraph is block element.</p>
      <span>I am first span.</span>
      <span>I am second span.</span>
      <span>Span is inline element.</span>
   </body>
</html>
```

HTML Elements

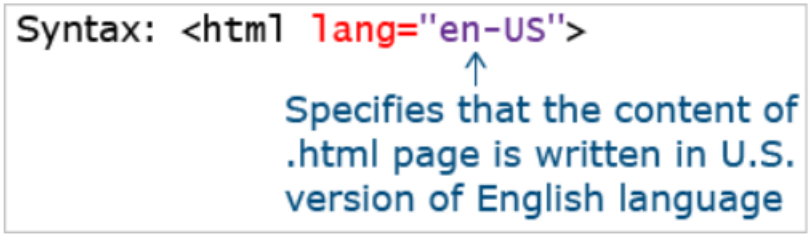
HTML elements can contain attributes that can be considered as an additional feature to set various properties and they are optional.

Some of the attributes can be used with any of the HTML elements and there can be referred to as 'global attributes'. Also, some attributes can be used only with particular elements. Following are some features of attributes:

- All the attributes can contain properties like name and value which can be used by a developer to assign respective details for that HTML elements.
- Attributes are to be set only in the start tag of a container HTML element.
- Attributes are case-insensitive, but it is recommended to use lowercase as a best practice.
- The best practice is always to quote attribute value even though we will not get any execution errors if they are not provided in quotes.

Example:

The lang attribute specifies the language of the content of the HTML page.



Metadata Element

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <title>Meta tag example</title>
  <!-- Specifies the character encoding for the document. -->
  <!-- ISO-8859-1 -->
  <meta charset="utf-8">
  <meta name="author" content="Susan">
  <meta name="description" content="This is a sample demo to understand HTML.">
  <!-- A web app wants to allow content from a current domain and truste domain xyz
    and all its subdomains -->
  <meta http-equiv="content-security-policy" content="default-src 'self' http://xyz.com" >
  <!-- Specify the preferred style sheet to use.-->
  <meta http-equiv="default-style" content="/style.css">
```

```
<!-- page content display to be adjusted to the device width being used to view the content with initial zoom level as 100% -->
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<p> Sample demo to explain meta tag with attributes </p>
</body>
</html>
```

<article>:

The <article> element is used to include self-contained composition on a web page.

```
<article>
   <h1>MEAN stack</h1>
   <p>MEAN stack training includes discussion on MongoDB, Node,
 Express and Angular with the corresponding certifications</p>

</arcicle>
```

<aside>:

The <aside> element is used to include content related to the main content of the web page.

```
<article>
   <h1>MEAN stack</h1>
   <p>MEAN stack training includes discussion on MongoDB, Node, Express and Angular with the corresponding certifications</p>
   <aside>
      <p>Visit our official website to attempt our certifications</p>
   </aside>

</arcicle>
```

<address>:

The <address> element helps to semantically organize address details in HTML content.

```
<address>
    John
    #231A
    Palace Lane
    Bangalore
</address>>

```

1.b Course Name: HTML5 - The Language

Module Name: Sectioning Elements

Enhance the Homepage.html of IEKart's Shopping Application by adding appropriate sectioning elements.

Source code:-

Sectioning Elements

```
<!DOCTYPE html>
<html>
  <body>
    <header>
      <nav> Login | SignUp </nav>
      <h1> Brand Name </h1>
    </header>
    <article>
      <section>
        <p>Paragraph 1</p>
        <p>Paragraph 2 is thematically related to Paragraph 1</p>
      </section>
```

```
    </article>

    <aside>

       Side info

    </aside>

    <footer>

       <nav> About Us | Contact Us </nav>

    </footer>

  </body>

</html>
```

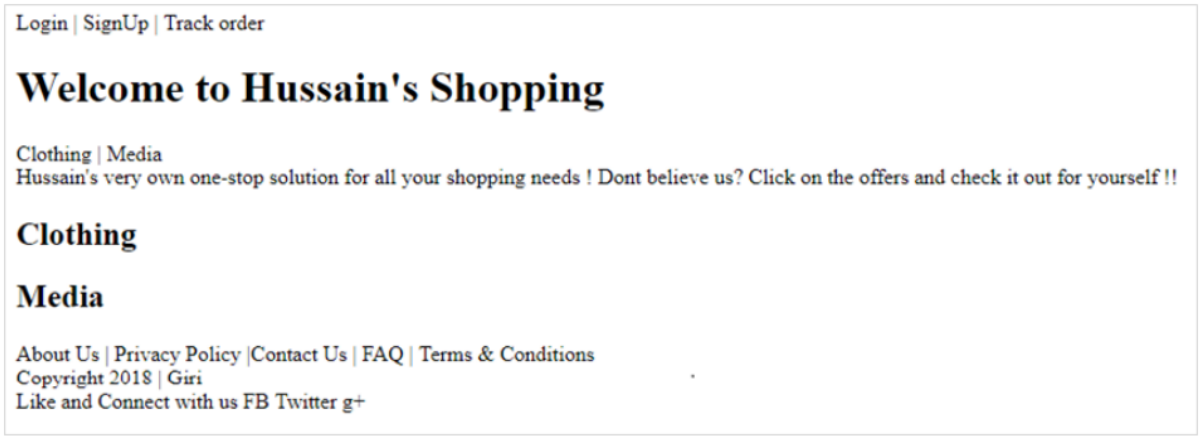
Problem Statement:

Consider the below code for Homepage.html. Enhance the semantics of content by adding appropriate sectioning elements to achieve the below-mentioned expected output.

Homepage.html:

```
<!DOCTYPE html>
<html>

    <head>
        <title>Hussian's Shopping</title>
        <meta name="keywords" content="Online, Shopping" />
    </head>

    <body>
        Login | SignUp | Track order Welcome to Hussian's Shopping Clothing | Media
    Hussain's very own one-stop solution for all your shopping needs !
            Dont believe us? Click on the offers and check it out for yourself !! Clothing
    Media About
        Us | Privacy Policy |Contact Us | FAQ | Terms & Conditions Copyright 2018 | Giri
    Like and Connect with us
        FB Twitter g+
    </body>

</html>


```

Expected Output:



1.c Course Name: HTML5 - The Language

Module Name: Paragraph Element, Division and Span Elements, List Element

Make use of appropriate grouping elements such as list items to "About Us" page of IEKart's Shopping Application

The paragraph element is generally used for denoting a paragraph. Any textual content can be mentioned inside this element.

It is defined using … tag.

Let us understand this with an example.

Copy the below code into your Visual Studio Code workspace.

```
<!DOCTYPE html>
<html>
   <body>
      <p>This is a Paragraph</p>
   </body>
</html>
```

To execute this code:

- Right-click on the file demo.html
- Select the option "Open with Live Server"
- Right-click on the browser, and select the inspect option.

Paragraph Element

SOURCE CODE:-

```
<!DOCTYPE html>
<html>
   <body>
      <p>Paragraph 1</p>
      <p>Paragraph 2</p>
      <p>Paragraph 3</p>
   </body>
</html>
```

Division and Span Elements

Similar to the division element, the span element is also used to group various other HTML tags to apply some common styles.

It is defined by using ... tag.

The span element is by default inline in nature, and hence no new line is added after the span ends. This tag is preferred only when we cannot use any other semantic tags.

Copy the below code into your Visual Studio Code workspace.

demo.html

```
<!DOCTYPE html>
<html>
    <body>
        <div>
            <span>first section of paragraph</span>
            <span>second section of paragraph</span>
        </div>
    </body>
```

```
</html>
```

To execute this code:

- Right-click on the file demo.html
- Select the option "Open with Live Server"
- Right-click on the browser, and select the inspect option.

Division and Span Elements

```
<!DOCTYPE html>

<html>

   <body>

      <div> <!-- Groups all 3 paragraphs -->

         <p>Paragraph 1</p>

         <p>Paragraph 2</p>

         <p>Paragraph 3</p>

         <span>first portion </span><span> second portion </span>

      </div>

   </body>

</html>
```

Unordered List

```
<!DOCTYPE html>

<html>

   <body>

      <ul>
```

```
        <li> One </li>
        <li> Two </li>
        <li> Three </li>
    </ul>
  </body>
</html>
```

Ordered List

```
<!DOCTYPE html>
<html>
  <body>
    <ol>
      <li>One</li>
      <li>Two</li>
      <li>Three</li>
    </ol>
    <ol start="4">
      <li>Four</li>
      <li>Five</li>
      <li>Six</li>
    </ol>
  </body>
</html>
```

Description List

```
<!DOCTYPE html>

<html>

   <body>

      <dl>

         <dt>The God of Small Things</dt>

         <dd>The story is authored by Arunthati Ry and is set in Kerala and revolves around the lives of two children Rahel and Esthepa and how they weave and imagine their childhood experiences.</dd>

         <dt>Shadow Lines</dt>

         <dd>Shadow Lines is an invigorating story by Amitav Ghosh about the borders that mark and limit our imaginations and memories. </dd>

         <dt>The Lord of The Rings</dt>

         <dd>An epic high fantasy book by the English author and scholar J.R.R.Tolkien</dd>

      </dl>

   </body>

</html>
```

OUTPUT:-

The God of Small Things
: The story is authored by Arunthati Ry and is set in Kerala and revolves around the lives of two children Rahel and Esthepa and how they weave and imagine their childhood experiences.

Shadow Lines
: Shadow Lines is an invigorating story by Amitav Ghosh about the borders that mark and limit our imaginations and memories.

The Lord of The Rings
: An epic high fantasy book by the English author and scholar J.R.R.Tolkien

Grouping Elements

```
<!DOCTYPE html>
<html>

  <head>
    <title>Hussian's Shopping</title>
    <meta name="keywords" content="Online, Shopping" />
  </head>
  <body>
    <header>
      <nav> Login | SignUp | Track order </nav>
      <h1> Welcome to Hussian's Shopping </h1>
      <nav> Electronics | Books | Sports | Media | Clothing | Offers Zone </nav>
    </header>
    <article>
      <h2>About Us</h2> Hussain's Shopping is one among the world wide on-line products dealer.
      <section> !--Include list items here--!</section>
    </article>
    <footer>
      <nav> About Us | Privacy Policy | Contact Us | FAQ | Terms & Conditions </nav> Copyright 2018 | Giri
    </footer>
    <aside> Like and Connect with us FB Twitter g+ </aside>
```

```
</body>

</html>
```

Expected Output:

1.d Course Name: HTML5 - The Language

Module Name: Link Element

Link "Login", "SignUp" and "Track order" to "Login.html", "SignUp.html" and "Track.html" page respectively. Bookmark each category to its details of IEKart's Shopping application.

Problem Statement:

Make below observation(s) in the given code-snippet:

1. Defining hyperlinks using the Link element.
2. The first link opens in a new tab.
3. The second link opens in the same tab.

Activities:

Change the value set for the href attribute and observe the output.

Note: You can execute this tryout in your Visual Studio code IDE or any other editors.

```html
<html>
    <body>
        <a href="https://owasp.org/" target="_blank">
            Opens OWASP website in new tab
        </a>
        <br/>
        <a href="https://owasp.org/" target="_top">
            Opens OWASP website in same tab
        </a>
    </body>
</html>
```

SOURCE CODE:-

```
<!DOCTYPE html>

<html>

  <head>

    <meta name="keywords" content="Online, Shopping" />

    <title>Hussain's Shopping</title>

  </head>

  <body>

    <header>

      <nav> Login | SignUp | Track order </nav>

      <h1>Welcome to Hussain's Shopping</h1>

      <nav> Clothing | Electronics </nav>

    </header>

    <article>

      Hussain's one-stop solution for all your shopping needs ! Dont believe us? Click on the offers and

      check it out for yourself !! <h2 id="Clothing"> Clothing </h2>

      <h2 id="Electronics"> Electronics</h2>

    </article>

    <footer>

      <nav> About Us | Privacy Policy | Contact Us | FAQ | Terms & Conditions </nav>
Copyright 2016 | ETA UI and MarkUp
```

```
    </footer>
    <aside> Like and Connect with us FB Twitter g+ </aside>
  </body>
</html>
```

Expected Output:

1.e Course Name: HTML5 - The Language

Module Name: Character Entities

Add the © symbol in the Home page footer of IEKart's Shopping application.

Problem Statement:

Consider below the Home page of Hussain's Shopping application. Modify the below code and add the © symbol in the footer.

Homepage.html

```
1.  <!DOCTYPE html>
2.  <html>
```

```
3.
4.      <head>
5.          <meta name="keywords" content="Online, Shopping" />
6.          <title>Hussain's Shopping</title>
7.      </head>
8.
9.      <body>
10.         <header>
11.             <nav>
12.                 <a href="Login.html"> Login </a> | <a href="Signup.html"> SignUp </a> |<a
    href="TrackOrder.html"> Track order </a>
13.             </nav>
14.             <h1>Welcome to Hussain's Shopping</h1>
15.             <nav>
16.                 <a href="#Clothing"> Clothing </a> | <a href="#Media"> Media </a>
17.             </nav>
18.         </header>
19.         <article> Hussain & Co very own one-stop solution for all your shopping needs !
20.             <br />
21.             Dont believe us? Click on the offers
22.             and check it out for yourself !!
23.             <h2 id="Clothing"> Clothing </h2>
24.             <h2 id="Media"> Media </h2>
25.         </article>
26.         <footer>
27.             <nav> About Us | Privacy Policy | Contact Us | FAQ | Terms & Conditions </nav>
    Copyright
28.             <!-- Add character entity for Copyright symbol -->2018 | Giri
29.         </footer>
30.         <aside> Like and Connect with us FB Twitter g+ </aside>
31.     </body>
32.
33. </html>
```

Expected Output:-

1.f Course Name: HTML5 - The Language

Module Name: HTML5 Global Attributes

Add the global attributes such as contenteditable, spellcheck, id etc. to enhance the Signup Page functionality of IEKart's Shopping application

Problem Statement:

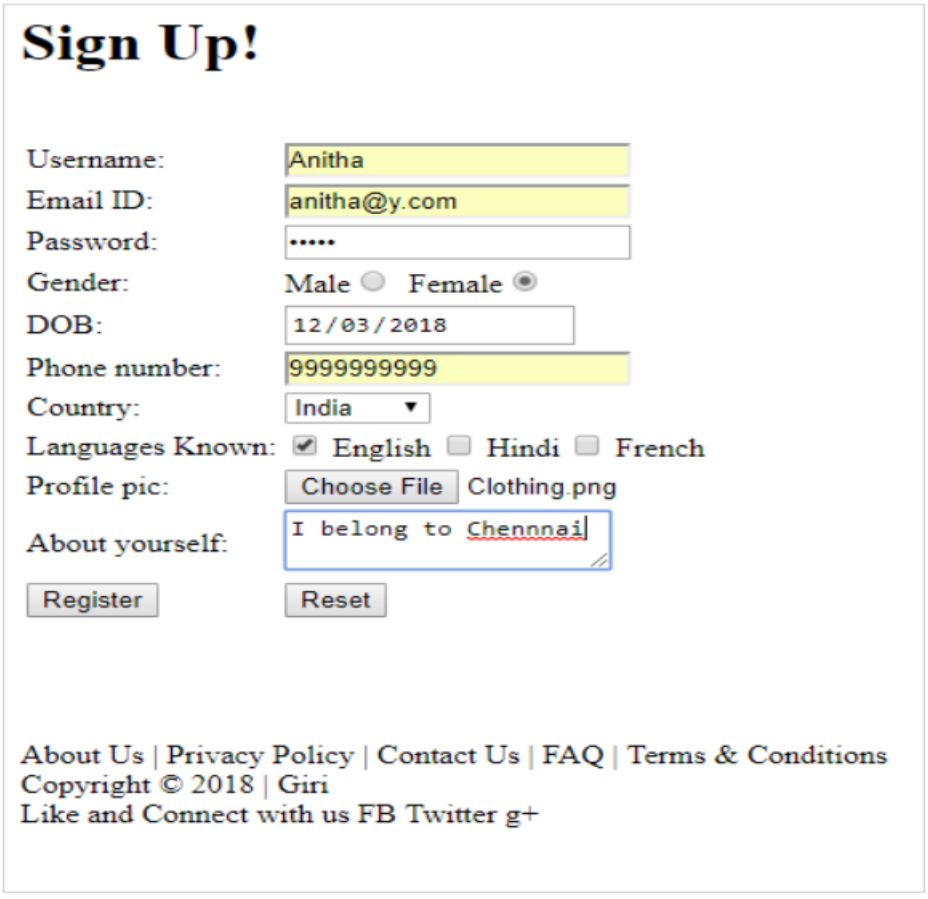
Consider the Signup Page of Hussain's Shopping application. Enhance its functionality by adding global attributes such as contenteditable and spellcheck to populate the below-given output.

Signup.html

```
<!DOCTYPE html>
<html>

    <head>
        <meta name="keywords" content="Online, Shopping" />
        <title>Hussain's Shopping</title>
    </head>

    <body>
        <header>
            <h1>Sign Up!</h1>
        </header>
        <article>
            <form action="success.html" method="get">
                <table>
                    <tr>
                        <td><br><label for="username">Username:</label></td>
                        <td><br><input type="text" id="username" placeholder="Enter your
username" required /></td>
                    </tr>
                    <tr>
                        <td><label for="email_id">Email ID:</label></td>
                        <td><input type="email" id="email_id" placeholder="Enter your email
ID" required /></td>
                    </tr>
                    <tr>
                        <td><label for="password">Password:</label></td>
                        <td><input type="password" id="password" placeholder="Enter your
password" required /></td>
                    </tr>
                    <tr>
                        <td><label for="gender">Gender:</label></td>
                        <td><label for="male">Male<input type="radio" id="male"
name="gender" value="M" />  <label for="female">Female<input
                                    type="radio" id="female" name="gender" value="F" /></td>
                    </tr>
                    <tr>
                        <td><label for="dob">DOB:</label></td>
                        <td><input type="date" id="dob" required /></td>
                    </tr>
                    <tr>
                        <td><label for="phone_no">Phone number:</label></td>
                        <td><input type="text" id="phone_no" placeholder="Enter your contact
number" pattern="+ [0-9] {12}" required /></td>
                    </tr>
                    <tr>
                        <td><label for="country">Country:</label></td>
                        <td><select id="country" placeholder="Select your country">
                                <option value="India" />India
                                <option value="India" />USA
                                <option value="India" />UK
                                <option value="India" />Canada
                                <option value="India" />Belgium
                                <option value="India" />France </select></td>
                    </tr>
                    <tr>
                        <td><label id="language">Languages Known:</label></td>
                        <td><input type="checkbox" name="language" id="english"
value="English" checked="checked" /><label for="english">
                                English </label> <input type="checkbox" name="language"
id="hindi" value="Hindi" /><label for="hindi"> Hindi
                            </label> <input type="checkbox" name="language" id="french"
value="French" /><label for="french"> French </label></td>
                    </tr>
                    <tr>
```

```
                        <td><label for="pic">Profile pic:</label></td>
                        <td><input type="file" id="pic" required /></td>
                    </tr>
                    <tr>
                        <td><label for="yourself" dir="ltr">About yourself:</label></td>
                        <td><textarea></textarea></td> <!-- Add contenteditable and
spellcheck attribute -->
                    </tr>
                    <tr>
                        <td><input type="submit" value="Register" />
                        <td><input type="reset" value="Reset" />
                    </tr>
                </table>
            </form> <br /> <br /> <br /> <br />
        </article>
        <footer>
            <nav> About Us | Privacy Policy | Contact Us | FAQ | Terms & Conditions </nav>
Copyright &copy; 2018 | Giri
        </footer>
        <aside> Like and Connect with us FB Twitter g+ </aside>
    </body>

</html>
```

Expected Output:



2.a Course Name: HTML5 - The Language

Module Name: Creating Table Elements, Table Elements : Colspan/Rowspan Attributes, border, cellspacing, cellpadding attributes

Enhance the details page of IEKart's Shopping application by adding a table element to display the available mobile/any inventories.

SOURCE CODE:-

```
<!DOCTYPE html>
<html>
   <body>
      <table>
         <caption>Team Details</caption>
         <thead>
         <tr>
            <th>Employee Name</th>
            <th>Emp ID</th>
            <th>Location</th>
         </tr>
         </thead>
         <tbody>
         <tr>
            <td>John</td>
            <td>102342</td>
            <td>Banglore</td>
         </tr>
         <tr>
            <td>Joseph</td>
            <td>890234</td>
```

```
        <td>Pune</td>
      </tr>
       <tr>
        <td>Sam</td>
        <td>102342</td>
        <td>Mysore</td>
      </tr>
      </tbody>
    </table>
  </body>
</html>
```

Expected output:

Team Details

Employee Name	Emp ID	Location
John	102342	Banglore
Joseph	890234	Pune
Sam	102342	Mysore

2.b Course Name: HTML5 - The Language

Module Name: Creating Form Elements, Color and Date Pickers, Select and Datalist Elements

Using the form elements create Signup page for IEKart's Shopping application.

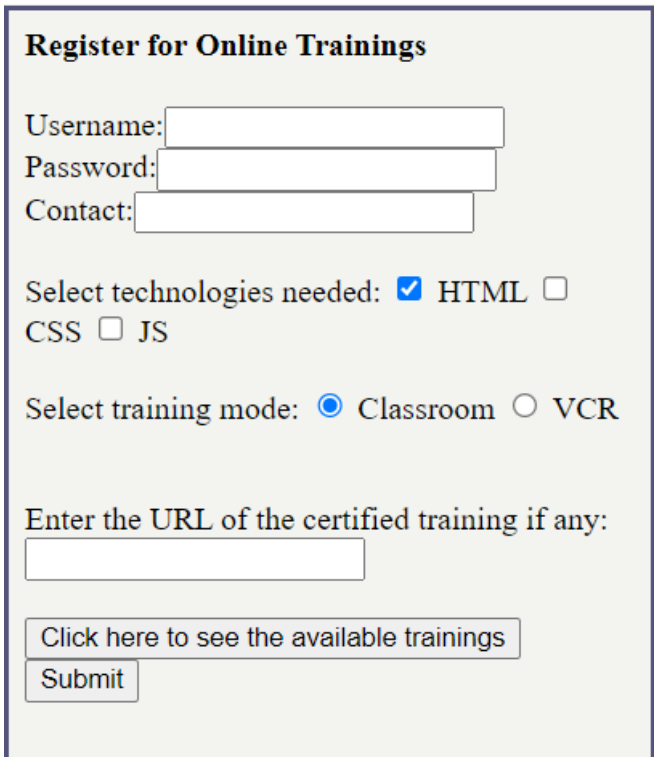
SOURCE CODE:-

```
<!DOCTYPE html>
```

```
<html>
  <body>
    <form>
      <h4>Register for Online Trainings</h4>
      <!-- input fields types text/password/email/number  -->
      Username:<input type="text"/> <br/>
      Password:<input type="password"/> <br/>
      Contact:<input type="number"/> <br/><br/>

      <!-- input fields type checkbox -->
      Select technologies needed:
      <input type="checkbox" value="HTML" checked /> HTML
      <input type="checkbox" value="CSS"/> CSS
      <input type="checkbox" value="JS"/> JS<br/><br/>

      <!-- input field type radio button -->
      Select training mode:
      <input type="radio" name="training" value="classroom" checked /> Classroom
      <input type="radio" name="training" value="VCR" /> VCR <br/>

      <!-- input field type hidden -->
      <input type="hidden" name="language" value="english" /> <br/><br/>

      Enter the URL of the certified training if any: <input type="url" /> <br/> <br/>

      <!-- input field type button -->
      <input type="button" value="Click here to see the available trainings"/>
  
      <!-- input field type submit -->
```

```
<input type="submit"/>   

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

OUTPUT:-



2.c Course Name: HTML5 - The Language

Module Name: Input Elements - Attributes

Enhance Signup page functionality of IEKart's Shopping application by adding attributes to input elements.

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

```
<head>
    <meta name="keywords" content="Online, Shopping" />
    <title>Hussian's Shopping</title>
</head>


<body>
    <header>
        <h1>Sign Up!</h1>
    </header>
    <article>
        <form action="success.html" method="get">
            <table>
                <tr>
                    <td><br>Username:</td>
                    <td><br><input type="text" name="UserName" /></td> <!-- Add placeholder and required attributes -->
                </tr>
                <tr>
                    <td>Password:</td>
                    <td><input type="password" name="name" /></td> <!-- Add placeholder and required attributes -->
                </tr>
                <tr>
                    <td>Gender:</td>
                    <td>Male<input type="radio" name="user_gender" value="M" />  Female<input type="radio" name="user_gender"
                        value="F"></td>
                </tr>
                <tr>
```

```
            <td>DOB:</td>
            <td><input type="date" name="user_date" /></td> <!-- Add required attribute
-->
        </tr>
        <tr>
            <td>Phone number:</td>
            <td><input type="text" name="phone" /><br><br></td> <!-- Add
placeholder, pattern and required attributes -->
        </tr>
        <tr>
            <td>Email ID:</td>
            <td><input type="email" name="user_email" /></td> <!-- Add placeholder
and required attributes -->
        </tr>
        <tr>
            <td><br>Languages Known:</td>
            <td><br><input type="checkbox" name="Langauges" value="English"
checked="checked" />English <input type="checkbox"
                name="Langauges" value="Hindi" /> Hindi <input type="checkbox"
name="Langauges" value="French" /> French </td>
        </tr>
        <tr>
            <td><br>Profile pic:</td>
            <td><br><input type="file" /></td> <!-- Add required attribute -->
        </tr>
        <tr>
            <td><br /><br /><input type="submit" value="Register" />
            <td><br /><br /><input type="reset" value="Reset" />
        </tr>
    </table>
```

```
        </form>
        <br /> <br /> <br /> <br />
    </article>
    <footer>
        <nav> About Us | Privacy Policy | Contact Us | FAQ | Terms & Conditions </nav>
Copyright &copy; 2018 | Giri
    </footer>
    <aside> Like and Connect with us FB Twitter g+ </aside>
  </body>

</html>
```

OUTPUT:-

2.d Course Name: HTML5 - The Language

Module Name: Media, Iframe

Add media content in a frame using audio, video, iframe elements to the Home page of IEKart's Shopping application.

SOURCE CODE:-

```
<!DOCTYPE html>
        <html>
          <head>
            <meta name="keywords" content="Online, Shopping" />
            <title>Hussain's Shopping</title>
          </head>

          <body>
            <header>
              <nav>
                <a href="Login.html"> Login </a> |
                <a href="Signup.html"> SignUp </a> |
                <a href="TrackOrder.html"> Track order </a>
              </nav>
              <h1>Welcome to Hussain's Shopping</h1>
              <nav>
                <a href="#Clothing"> Clothing </a> |
                <a href="#Media"> Media </a>
              </nav>
            </header>

            <article>
              Hussain & Co very own one-stop solution for all your shopping needs ! <br/>
```

```
        Dont believe us? Click on the offers and check it out for yourself !!


        <h2 id="Clothing"> Clothing </h2>
        <!-- Add image element here -->


        <h2 id="Media"> Media </h2>
        <!-- Add audio element here -->


        <br/>


        <!-- Add video element here -->


    </article>


    <footer>
        <nav>
            About Us | Privacy Policy | Contact Us | FAQ | Terms & Conditions
        </nav>
        Copyright &copy; 2018 | Giri
    </footer>


    <aside>
        Like and Connect with us FB Twitter g+
    </aside>
  </body>
</html>
```

COURSE COMPLETION CERTIFICATE:-

Infosys
Navigate your next

COURSE COMPLETION CERTIFICATE

The certificate is awarded to

Teja Lodagala

for successfully completing the course

HTML5 - The Language

on Sunday, February 12th 2023

Infosys | Springboard

Congratulations! You make us proud!

Thirumala Arohi
Senior Vice President and Head
Education, Training and Assessment (ETA)
Infosys Limited

Issued on: Sunday, February 12th 2023
This certificate can be verified by scanning the QR code at https://verify.onwingspan.com

3.a Course Name: Javascript

Module Name: Type of Identifiers

Write a JavaScript program to find the area of a circle using radius (var and let - reassign and observe the difference with var and let) and PI (const)

SOURCE CODE:-

```
<!DOCTYPE html>
<html>

<head>
    <title>Identifier Demo</title>
    <style>
        body {
            padding-top: 10px;
        }
    </style>
</head>

<body class="container-fluid">
    <div class="panel panel-primary">
        <div class="panel-heading">
            <h3>Body Temperature</h3>
        </div>

        <div class="panel-body">
            <script>
                var tempFahrenheit = 99;
                let TempFahrenheit = 99;
                // let TempFahrenheit=99; Redeclaration is not allowed
```

```
        const TEMP_CELSIUS = 38;
        // document.write() is used to write content onto the HTML page
        document.write("Default temperature (var) is: " + tempFahrenheit + "</span>");
        document.write("<br/>");
        document.write("Default temperature (let) is: " + TempFahrenheit + "</span>");
        document.write("<br/>");
        document.write("Normal body temperature in Celsius is: " + TEMP_CELSIUS +
"</span>");
      </script>
    </div>
  </div>
</body>

</html>
```

OUTPUT :-

Body Temperature
Default temperature (var) is: 99
Default temperature (let) is: 99
Normal body temperature in Celsius is: 38

3.b Course Name: Javascript

Module Name: Primitive and Non Primitive Data Types

Write JavaScript code to display the movie details such as movie name, starring, language, and ratings. Initialize the variables with values of appropriate types. Use template literals wherever necessary.

SOURCE CODE:-

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

```
<head>
   <title>Datatypes Demo</title>
   <style>
      body {
         padding-top: 10px;
      }
   </style>
</head>

<body class="container-fluid">
   <div class="panel panel-primary">
      <div class="panel-heading">
         <h3>Boolean Values</h3>
      </div>

      <div class="panel-body">
         <script>
            let empName = "Philip Jackson",
            empAge = 26,
            isConfirmed = 0,
            bonous = undefined,
            designation = "System Engineer",
            promotions = null;
            document.write("<b> Primitives and Non-Primitive Data types </b> <br /><br>");
            document.write("Employee Details: <br/>");
            document.write("<br/>Employee Name:"+empName);
         </script>
```

```
    </div>
  </div>
</body>
</html>
```

OUTPUT:-

Boolean Values
Primitives and Non-Primitive Data types
Employee Details:
Employee Name:Philip Jackson

3.c Course Name: Javascript

Module Name: Operators and Types of Operators

Write JavaScript code to book movie tickets online and calculate the total price, considering the number of tickets and price per ticket as Rs. 150. Also, apply a festive season discount of 10% and calculate the discounted amount.

SOURCE CODE:-

```
<!DOCTYPE html>
<html>

<head>
  <title>Operators Demo</title>
  <style>
    body {
      padding-top: 10px;
    }
  </style>
</head>
```

```
<body class="container-fluid">
  <div class="panel panel-primary">
    <div class="panel-heading">
      <h3>Body Temperature</h3>
    </div>

    <div class="panel-body">
      <p>Your body temperature is higher than the normal:
        <span class="blinking">
          <script>
            let tempFahren = 99;
            const NORMAL_FAHREN = 98.6;
            let tempCelsius;
            const THIRTYTWO = 32;
            const TEMP = 1.8;
            tempCelsius = (tempFahren - THIRTYTWO) / TEMP;
            let fever = tempFahren > NORMAL_FAHREN;
            document.write(fever);
            document.write("<br/>");
            document.write("Your temperature is Celsius  : " + tempCelsius);
            document.write("<br/>");
            document.write(typeof (fever));
            document.write("<br/>");
            document.write(typeof (tempFahren));
          </script>
        </span>
      </p>
    </div>
```

```
    </div>
</body>
</html>
```

OUTPUT:-

Body Temperature

Your body temperature is higher than the normal: true
Your temperature is Celsius : 37.22222222222222
boolean
number

3.d Course Name: Javascript

Module Name: Types of Statements, Non - Conditional Statements, Types of Conditional Statements, if Statements, switch Statements

Write a JavaScript code to book movie tickets online and calculate the total price based on the 3 conditions: (a) If seats to be booked are not more than 2, the cost per ticket remains Rs. 150. (b) If seats are 6 or more, booking is not allowed. (c) If se

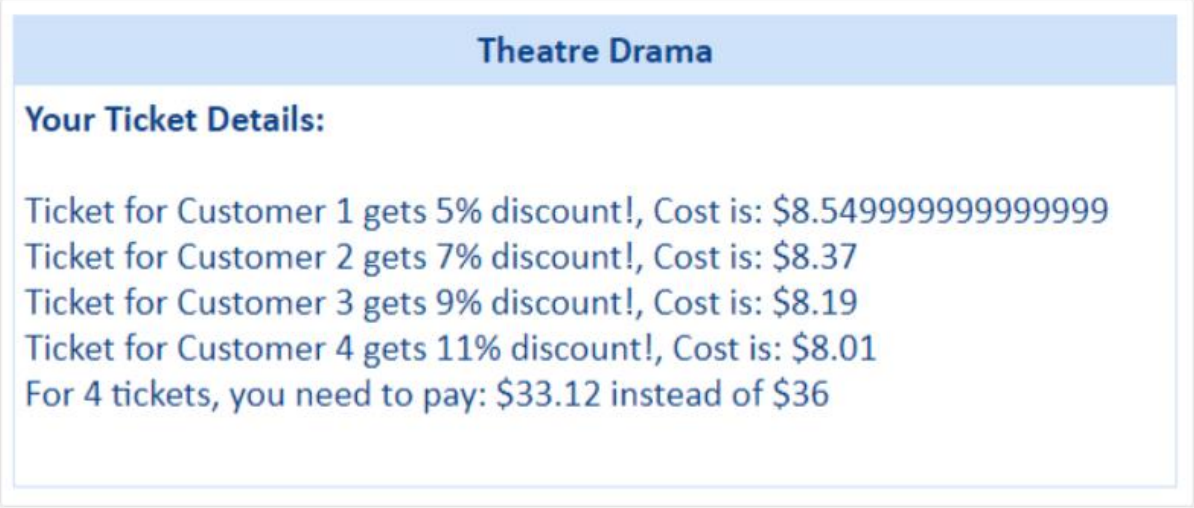
SOURCE CODE:-

```
<!DOCTYPE html>
<html>
<head>
    <title>Booking Details</title>
    <style>
        div#maincontent {
            height: 200px;
            width: 600px;
            border: 1px solid #CEE2FA;
            text-align: left;
```

```
            color: #08438E;
            font-family: calibri;
            font-size: 20;
            padding: 5px;
        }
        div#heading {
            text-decoration: bold;
            text-align: center;
            margin-top: 80px;
            width: 600px;
            border: 1px solid #CEE2FA;
            text-align: center;
            color: #08438E;
            background-color: #CEE2FA;
            font-family: calibri;
            font-size: 20;
            padding: 5px;
        }
        h4 {
            padding: 0;
            margin: 0;
        }
    </style>
</head>
<body>
    <center>
        <div id="heading">
            <b>Theatre Drama</b>
```

```
    </div>
    <div id="maincontent">
      <h4>Your Ticket Details:</h4>
      <br>
      <script>
        // Write the code to display the total price and discounted price of tickets
      </script>
    </div>
  </center>
</body>
</html>
```

OUTPUT:-



3.e Course Name: Javascript

Module Name: Types of Loops

Write a JavaScript code to book movie tickets online and calculate the total price based on the 3 conditions: (a) If seats to be booked are not more than 2, the cost per ticket remains Rs. 150. (b) If seats are 6 or more, booking is not allowed. (c) If

SOURCE CODE:-

```
<!DOCTYPE html>
<html>
<head>
   <title>Booking Details</title>
   <style>
      div#maincontent {
         height: 200px;
         width: 600px;
         border: 1px solid #CEE2FA;
         text-align: left;
         color: #08438E;
         font-family: calibri;
         font-size: 20;
         padding: 5px;
      }
      div#heading {
         text-decoration: bold;
         text-align: center;
         margin-top: 80px;
         width: 600px;
         border: 1px solid #CEE2FA;
         text-align: center;
```

```
            color: #08438E;
            background-color: #CEE2FA;
            font-family: calibri;
            font-size: 20;
            padding: 5px;
        }
        h4 {
            padding: 0;
            margin: 0;
        }
    </style>
</head>
<body>
    <center>
        <div id="heading">
            <b>Theatre Drama</b>
        </div>
        <div id="maincontent">
            <h4>Your Ticket Details:</h4>
            <br>
            <script>
                //Write the code to display the ticket details
            </script>
        </div>
    </center>
</body>
</html>
```

OUTPUT:-

```
Theatre Drama
Your Ticket Details:

Ticket for Customer 1 gets 5% discount!, Cost is: $8.549999999999999
Ticket for Customer 2 gets 7% discount!, Cost is: $8.37
Ticket for Customer 3 gets 9% discount!, Cost is: $8.19
Ticket for Customer 4 gets 11% discount!, Cost is: $8.01
For 4 tickets, you need to pay: $33.12 instead of $36
```

4.a Course Name: Javascript

Module Name: Types of Functions, Declaring and Invoking Function, Arrow Function, Function Parameters, Nested Function, Built-in Functions, Variable Scope in Functions

Write a JavaScript code to book movie tickets online and calculate the total price based on the 3 conditions: (a) If seats to be booked are not more than 2, the cost per ticket remains Rs. 150. (b) If seats are 6 or more, booking is not allowed. (c) If

SOURCE CODE:-

```
<!DOCTYPE html>
<html>
<head>
   <title>Booking Summary</title>
   <style>
      div#maincontent {
         height: 250px;
         width: 500px;
         border: 1px solid #CEE2FA;
         text-align: left;
         color: #08438E;
         font-family: calibri;
         font-size: 20;
         padding: 5px;
      }
      div#heading {
         text-decoration: bold;
         text-align: center;
         margin-top: 80px;
```

```
            width: 500px;
            border: 1px solid #CEE2FA;
            text-align: center;
            color: #08438E;
            background-color: #CEE2FA;
            font-family: calibri;
            font-size: 20;
            padding: 5px;
        }
        h2 {
            padding: 0;
            margin: 0;
        }
    </style>
</head>
<body>
    <center>
        <div id="heading">
            <h2>Booking Summary</h2>
        </div>
        <div id="maincontent">
            <script>
                //Write the code to display the booking summary
            </script>
        </div>
    </center>
</body>
</html>
```

OUTPUT:-

4.b Course Name: Javascript

Module Name: Working With Classes, Creating and Inheriting Classes

Create an Employee class extending from a base class Person. Hints: (i) Create a class Person with name and age as attributes. (ii) Add a constructor to initialize the values (iii) Create a class Employee extending Person with additional attributes role

SOURCE CODE:-

```
<!DOCTYPE html>
<html>

<head>
    <title>Classes Demo</title>
    <script>
        class Vehicle {
            constructor(make, model) {
                this.make = make;
```

```
        this.model = model;
      }
    }
    class Car extends Vehicle {
      constructor(make, model, regNo, fuelType) {
        super(make, model);
        this.regNo = regNo;
        this.fuelType = fuelType;
      }
      getDetails() {
        document.write(`${this.make},${this.model},${this.regNo},${this.fuelType}`);
      }
    }

    let cObj1 = new Car("Hundai", "i10", "KA01-6447", "Petrol");
    cObj1.getDetails();
  </script>
</head>

</html>
```

OUTPUT:-

Hundai,i10,KA01-6447,Petrol

4.c Course Name: Javascript

Module Name: In-built Events and Handlers

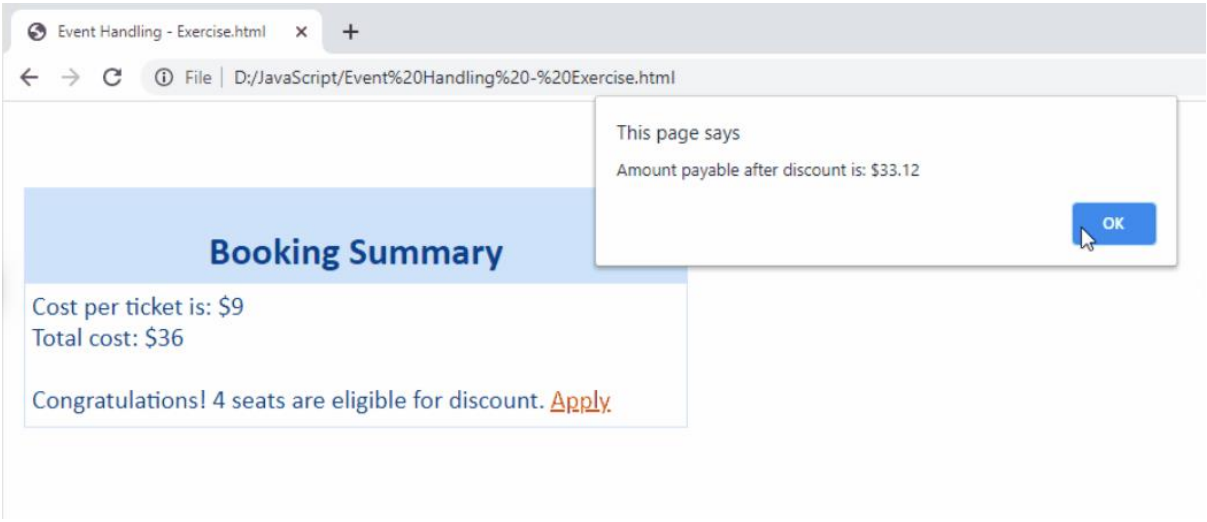
Write a JavaScript code to book movie tickets online and calculate the total price based on the 3 conditions: (a) If seats to be booked are not more than 2, the cost per ticket remains Rs. 150. (b) If seats are 6 or more, booking is not allowed. (c) If se

SOURCE CODE:-

```
<!DOCTYPE html>
<html>
<head>
   <title>Booking Details</title>
   <style>
      div#maincontent {
         height: 100px;
         width: 500px;
         border: 1px solid #CEE2FA;
         text-align: left;
         color: #08438E;
         font-family: calibri;
         font-size: 20;
         padding: 5px;
         margin-left: 10px;
      }
      div#heading {
         text-decoration: bold;
         text-align: center;
         margin-top: 40px;
         margin-left: 10px;
         width: 500px;
         border: 1px solid #CEE2FA;
         text-align: center;
         color: #08438E;
         background-color: #CEE2FA;
         font-family: calibri;
```

```
            font-size: 20;
            padding: 5px;
        }
        h2 {
            padding: 0;
            margin: 0;
        }
    </style>
</head>

<body>
    <div id="heading">
        <h2>Booking Summary</h2>
    </div>
    <div id="maincontent">
        <script>
            //Write the code to display the discounted price on click of the text
        </script>
    </div>
</body>

</html>
```

OUTPUT:-



4.d Course Name: Javascript

Module Name: Working with Objects, Types of Objects, Creating Objects, Combining and cloning Objects using Spread operator, Destructuring Objects, Browser Object Model, Document Object Model

If a user clicks on the given link, they should see an empty cone, a different heading, and a different message and a different background color. If user clicks again, they should see a re-filled cone, a different heading, a different message, and a diffe

SOURCE CODE:-

```
<html>

<head>
    <title>Objects Demo</title>
    <script>
        document.write("<b> Object Creation using Object Literal method </b>");
```

```
var Pdt1 =
{
   PdtId: "P001",
   PdtName: "Furniture",
   Price: 850,
   Qty: 0,
   TotalAmount: 0,
   ChangeQty: function (newQty) {
      this.TotalAmount = newQty * this.Price;
      return this.TotalAmount;
   },
};

document.write("<br/>");
document.write(Pdt1.PdtId);
document.write("<br/>");
document.write(Pdt1["PdtName"]);
document.write("<br/>");
document.write(Pdt1.TotalAmount);
Pdt1.ChangeQty(10);
document.write("<br/>");
document.write(Pdt1.TotalAmount);
document.write("<br/>");
document.write("-----------");
document.write("<br/>");
document.write(Pdt1.PdtId);
document.write("<br/>");
document.write(Pdt1["PdtName"]);
```

```
        document.write("<br/>");
        document.write(Pdt1.TotalAmount);
        Pdt1.ChangeQty(25);
        document.write("<br/>");
        document.write(Pdt1.TotalAmount);
        document.write("<br/>");
    </script>
</head>

<body>
</body>

</html>
```

OUTPUT:-

Object Creation using Object Literal method

```
P001
Furniture
0
8500
----------
P001
Furniture
8500
21250
Scripting output
```

5.a Course Name: Javascript

Module Name: Creating Arrays, Destructuring Arrays, Accessing Arrays, Array Methods

Create an array of objects having movie details. The object should include the movie name, starring, language, and ratings. Render the details of movies on the page using the array.

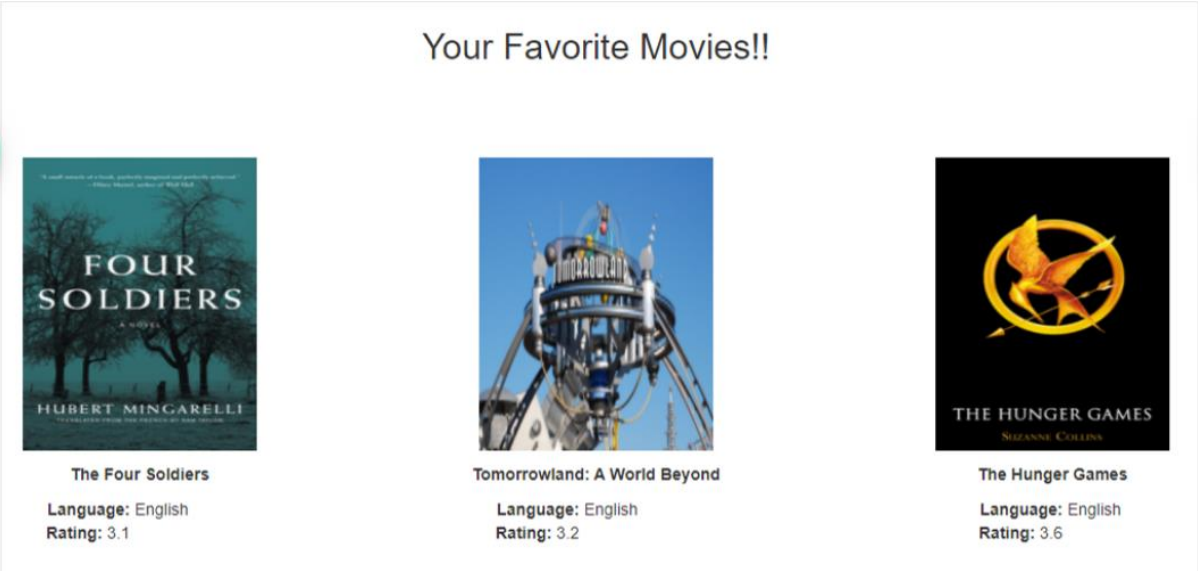
SOURCE CODE:-

```
<!DOCTYPE html>
<html>
<head>
    <title>Movies</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>
    <div style="margin-top:7%">
        <center>
            <h2>Your Favorite Movies!!</h2>
        </center>
    </div>
    <div class="container" style="padding-top:5%">
        <div class="row">
            <div class="col-md-4">
```

```
<div style="text-align: center;">
<img src="./images/FourSoldiers.jpg" width="200px" height="250px">
</div>
<div style="padding-top:10px;">
<div style="text-align: center; ">
<b><span id="mName0"></span></b></div>
<div style="padding-top:10px;padding-left: 101px;"><b>Language: </b><span id="lang0"></span>
</div>
<div style="padding-left: 100px;"><b>Rating: </b><span id="rating0"></span></div>
</div>
</div>
<div class="col-md-4">
<div style="text-align: center;">
<img src="./images/Tomorrowland.jpg" width="200px" height="250px">
</div>
<div style="padding-top:10px;">
<div style="text-align: center; "><b><span id="mName1"></span></b></div>
<div style="padding-top:10px;padding-left: 95px;"><b>Language: </b><span id="lang1"></span>
</div>
<div style="padding-left: 94px;"><b>Rating:</b><span id="rating1"></span></div>
</div>
</div>
<div class="col-md-4">
<div style="text-align: center;">
<img src="./images/HungerGames.jpg" width="200px" height="250px">
</div>
```

```
        <div style="padding-top:10px;">
            <div style="text-align: center; "><b><span id="mName2"></span></b></div>
            <div style="padding-top:10px;padding-left: 118px;"><b>Language: </b><span id="lang2"></span>
            </div>
            <div style="padding-left: 117px;"><b>Rating:</b><span id="rating2"></span></div>
        </div>
      </div>
    </div>
  </div>
</body>
<script>
  //Write the code here
</script>
</html>
```

OUTPUT:-



5.b Course Name: Javascript

Module Name: Introduction to Asynchronous Programming, Callbacks, Promises, Async and Await, Executing Network Requests using Fetch API

Simulate a periodic stock price change and display on the console. Hints: (i) Create a method which returns a random number - use Math.random, floor and other methods to return a rounded value. (ii) Invoke the method for every three seconds and stop

SOURCE CODE:-

Implement the following requirements to understand the concept of Asynchronous Programming.

1. Create a folder "Async_Await_Demo" and then follow the steps.

2. Create a file "demo.js" inside the folder and place the below-given code.

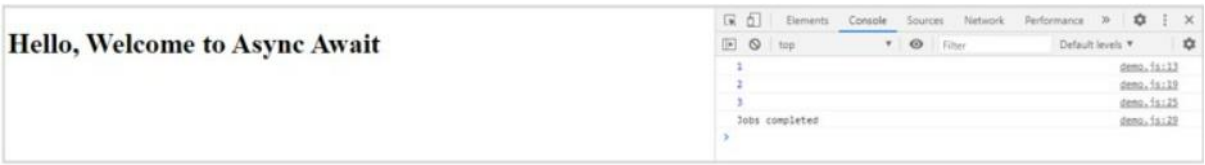
```
// returns a resolved promise value after asynchronous task completed
function asyncJob(x) {
        return new Promise(resolve => resolve(x + 1));
}
// we are linking series of three asynchronous tasks
async function execAsyncJobs() {
        /*await stops the execution of 'execAsyncJobs', until the promise returned by'asyncJob(0)', is resolved.
res1 assigned with resolved promise’s result value */
        const res1 = await asyncJob(0);
        console.log(res1);

        /* here,'asyncJob(res1)'is called, await stops execution of'execAsyncJobs', until the promise returned by'asyncJob(res1)' is resolved.
res2 assigned with resolved promise’s result value  */
        const res2 = await asyncJob(res1);
```

```
    console.log(res2);
    /* here, asyncJob(res2) is called, await stops execution of execAsyncJobs until the promise returned by asyncJob(res2) is resolved.
res3 assigned with resolved promise's result value */
    const res3 = await asyncJob(res2);
    console.log(res3);
    return "Jobs completed";
}
execAsyncJobs().then(res => console.log(res));
```

3. Create a file "index.html".

```
<!DOCTYPE html>
<html>
<head>
  <title> Async Await Example</title>
</head>
<body>
  <script type="module" src="./demo.js" ></script>
  <h1>Hello, Welcome to Async Await</h1>
</body>
</html>
```

OUTPUT:-



5.c Course Name: Javascript

Module Name: Creating Modules, Consuming Modules

Validate the user by creating a login module. Hints: (i) Create a file login.js with a User class. (ii) Create a validate method with username and password as arguments. (iii) If the username and password are equal it will return "Login Successful" else

SOURCE CODE:-

Implement the following requirements to understand the concept of Modular Programming

1. Create an application folder called "Default_Export_Import2" and then follow these steps.

2. Create a file "course1.js":

```
export let courseName = "Angular";
let course = {getCourseName:function(){ return courseName;},
  setCourseName:function(newCourseName){ courseName =newCourseName; }
};


export default course;
```

3. Create a file "course2.js". This file utilizes the entities defined in the course1.js file.

```
import crs, {courseName} from "./course1.js";
console.log(courseName);
console.log(crs.getCourseName());
crs.setCourseName("ES6");


console.log(crs.getCourseName());
```

Here "crs" is the default export.

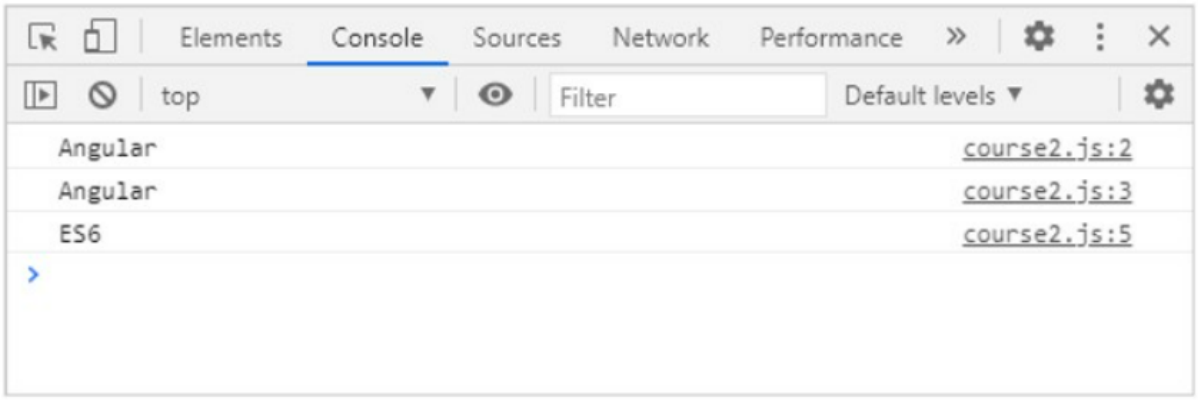
4. Create a file “index.html”:

```
<!DOCTYPE html>
<html>

<head>
    <title>Example</title>
</head>

<body>
    <script type="module" src="./course2.js"></script>
</body>

```

```
12. </html>
```

Here <script type="module"> at LineNo 9 indicates that the browser must consider a script as a module. In the above mentioned index.html file we have loaded course2.js module.

Output:-



COURSE COMPLETION CERTIFICATE:-

Infosys
Navigate your next

COURSE COMPLETION CERTIFICATE

The certificate is awarded to

Teja Lodagala

for successfully completing the course

JavaScript

on Tuesday, February 28th 2023

Infosys | Springboard

Congratulations! You make us proud!

Thirumala Arohi
Senior Vice President and Head
Education, Training and Assessment (ETA)
Infosys Limited

Issued on: Tuesday, February 28th 2023
This certificate can be verified by scanning the QR code at https://verify.onwingspan.com

6.a Course Name: Node.js

Module Name: How to use Node.js

Verify how to execute different functions successfully in the Node.js platform.

- Creating a JavaScript file
- Executing the JavaScript file using Node.js

Download the demo from this link web-server-demo

Demo steps:

Now that we know how to install the Node.js platform in our machine, let us create our first Node.js program.

Step 1: Create a folder NodeJS in D drive and create a new JavaScript file, **first.js** inside the folder. Type the below code inside the JavaScript file.

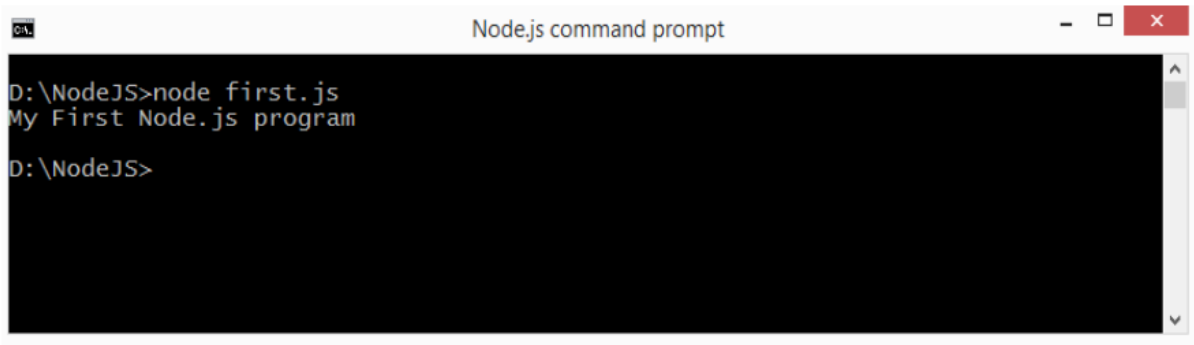
```
1.  console.log("My First Node.js program");
```

Step 2: Navigate to the created NodeJS folder in the NodeJS command prompt and execute the JavaScript file, first.js using the **node** command.

```
1.  node first.js
```

Step 3: After the successful interpretation of the code, we can see the output in the Node.js command prompt as shown below.

Node.js command prompt:



6.b Course Name: Node.js

Module Name: Create a web server in Node.js

Write a program to show the workflow of JavaScript code executable by creating web server in Node.js.

- Using require() and createServer() method
- Running a web server in Node.js

Download the demo from this link web-server-demo

Demosteps:

Step 1: Create a new JavaScript file **httpserver.js** and include the HTTP module.

```
1.  const http = require("http");
2.
```

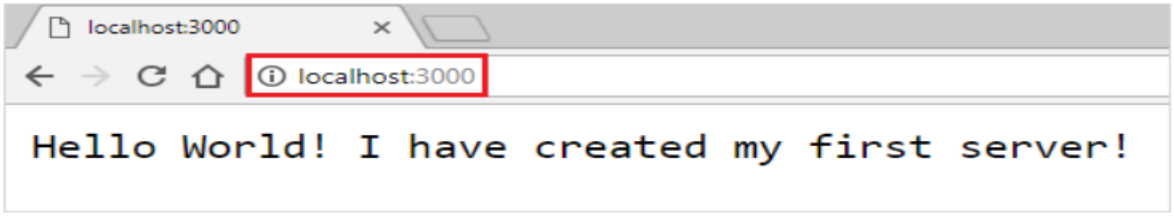
Step 2: Use the createServer() method of the HTTP module to create a web server.

```
1.  var server = http.createServer((req, res) => {
2.    res.write("Hello World! I have created my first server!");
3.    res.end();
4.  });
5.  server.listen(3000);
6.  console.log("Server started... Running on localhost:3000");
```

Step 3: Save the file and start the server using the **node** command. When the file executes successfully, we can observe the following output in the console.

```
D:\NodeJS>node httpserver
Server started... Running on localhost:3000
```

Step 4: We will observe the following in the browser.



6.c Course Name: Node.js
Module Name: Modular programming in Node.js
Write a Node.js module to show the workflow of Modularization of Node application.

- Creating a module
- Loading the module

Demosteps:

Let us try out how to create and load a module in a Node application.

Step 1: Create a file DBModule.js within the NodeJS folder created earlier.

```
exports.authenticateUser = (username, password) => {
  if (username === "admin" && password === "admin") {
    return "Valid User";
  } else return "Invalid User";
};


```

Step 2: Modify the file httpserver.js file created earlier as below.

app.js

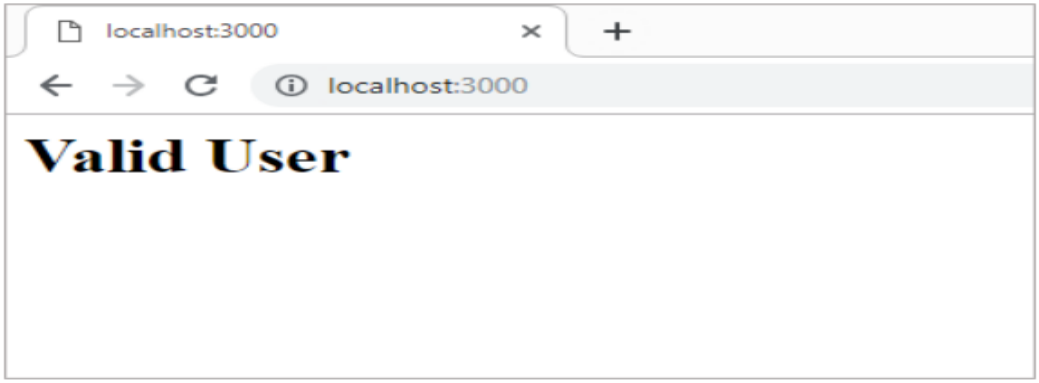
```
const http = require("http");
var dbmodule = require("./DBModule");
var server = http.createServer((request, response) => {
  result = dbmodule.authenticateUser("admin", "admin");
  response.writeHead(200, { "Content-Type": "text/html" });
  response.end("<html><body><h1>" + result + "</h1></body></html>");
  console.log("Request received");
});
server.listen(3000);
```

```
10. console.log("Server is running at port 3000");

11.
```

In the httpserver.js file, we are loading the module "DBModule" and then invoking the function named "authenticateUser()".

```
D:\NodeJS>node httpserver
Server is running at port 3000
```

Step 3: Run the httpserver.js using the **node** command.



Open a browser and navigate to URL "http://localhost:3000" and observe the output.

6.d Course Name: Node.js

Module Name: Restarting Node Application

Write a program to show the workflow of restarting a Node application.

Whenever we are working on a Node.js application and we do any change in code after the application is started, we will be required to restart the Node process for changes to reflect. In order to restart the server and to watch for any code changes automatically, we can use the Nodemon tool.

Nodemon

Nodemon is a command-line utility that can be executed from the terminal. It provides a different way to start a Node.js application. It watches the application and whenever any change is detected, it restarts the application.

It is very easy to get started with this tool. To install it in the application, run the below command.

```
```

Once the 'nodemon' is installed in the machine, the Node.js server code can be executed by replacing the command "node" with "nodemon".

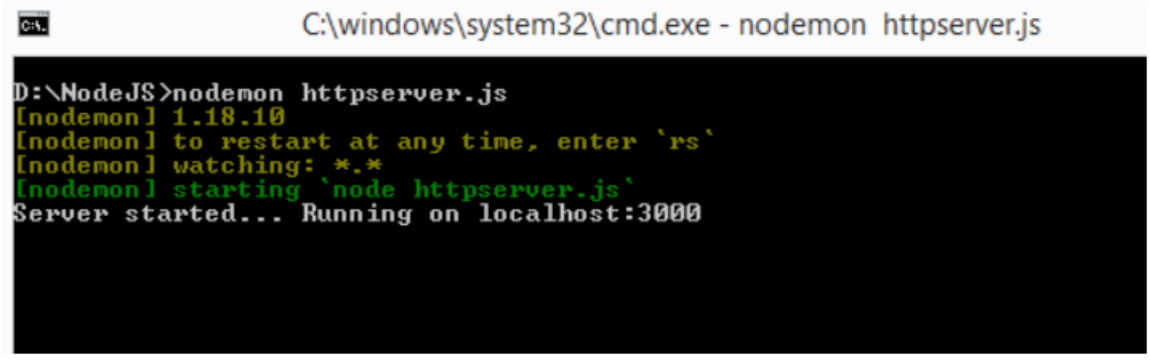
```
1. nodemon app.js
```

Thus the 'nodemon' starts the application in watch mode and restart the application when any change is detected.

Consider a simple Node.js code for creating a web server.

```
1. const http = require("http");
2.
3. var server = http.createServer((req, res) => {
4.   res.write("Hello World! I have created my first server!");
5.   res.end();
6. });
7. server.listen(3000);


8. console.log("Server started... Running on localhost:3000");
```

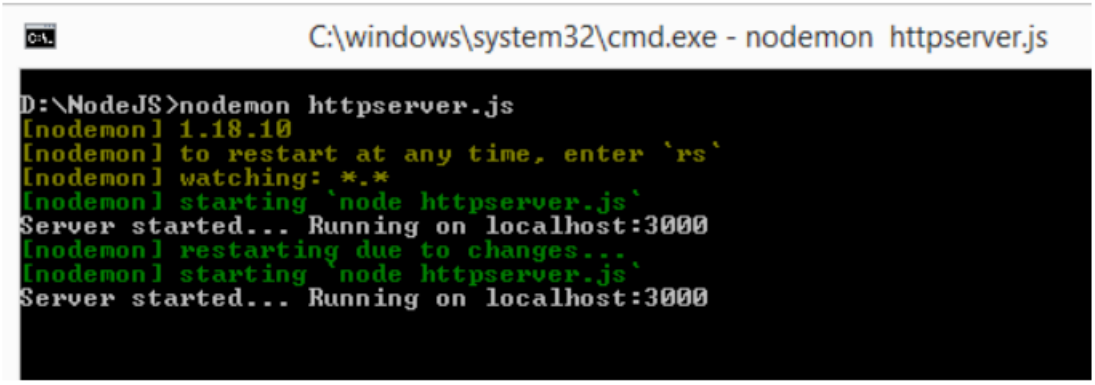
Observe the console on starting the nodemon in a command prompt window.



Now open the application code and do changes in the code as below.

```
const http = require("http");
var server = http.createServer((req, res) => {
  console.log("Request URL is " + req.url);
  res.write("Hello World! I have created my first server!");
  res.end();
});
server.listen(3000);


console.log("Server started... Running on localhost:3000");
```

Observe the console message in the command prompt. Nodemon automatically restarted the server on observing changes in the code.



6.e Course Name: Node.js

Module Name: File Operations

Create a text file src.txt and add the following data to it. Mongo, Express, Angular,

Node.

- Usage of fs.readFile() method to read the content of file.

Download the demo using this link Read-File-Demo

Demosteps:

Step 1: Create a JavaScript file 'fileSystem.js' and add the below code:

```
const fs = require('fs');
const { promisify } = require('util');

```

```
const readFile = promisify(fs.readFile);
(async () => {
    try {
        const fileData = await readFile('demo.txt', 'utf8');
        console.log(fileData);
    } catch (err) {
        console.log(err);
    }
})();
```

Step 2: Create a file demo.txt and add the following text.

```
Node.js is an open-source JavaScript run time environment which helps the developers to
build fast, scalable network applications.
Express is a very popular framework for Node.js, which simplifies the Node application

development.
```

Step 3: Run the file 'fileSystem.js' and observe the output in the console.

```
D:\Node>node fileSystem.js
Node.js is an open-source JavaScript run time environment which helps the developers to build fast, scalable network app
lications.
Express is a very popular framework for Node.js, which simplifies the Node application development.
```

Let us now see how to execute a JavaScript file that contains code for browser interaction through Node.js.

7.a Course Name: Express.js

Module Name: Defining a route, Handling Routes, Route Parameters, Query

Parameters

Implement routing for the AdventureTrails application by embedding the necessary

code in the routes/route.js file.

- Routing
- Request handling based on URL types

Demosteps:

1. Modify the **routing.js** file in router.js as shown below.

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

const routing = express.Router();
const notesController = require('../Controller/myNotes');

routing.get('/notes', notesController.getNotes);

routing.post('/notes', notesController.newNotes);

routing.all('*', notesController.invalid);

module.exports = routing;


```

2. Make sure App.js is having the below code.

```
const express = require('express');
const route = require('./Routes/routing');

const app = express();
app.use('/', route);

const port = process.env.PORT || 3000;
app.listen(port, () => {
  console.log(`App running on port ${port}...`);
});


```

3. Create the handlers on the myNotes.js inside the Controller folder.

```
exports.getNotes = async (req, res) => {
  try {
    res.status(200).json({
```

```
        message: 'You can now get the requested notes for your request ',
      });
    } catch (err) {
      res.status(404).json({
        status: 'fail',
        message: err,
      });
    }
};

exports.newNotes = async (req, res) => {
  try {
    res.status(201).json({
      data: 'New notes added for the POST request',
    });
  } catch (err) {
    res.status(404).json({
      status: 'fail',
      message: err,
    });
  }
};

exports.invalid = async (req, res) => {
  res.status(404).json({
    status: 'fail',
    message: 'Invalid path',
  });
};


```

4. Run the application and observe the different responses based on the request URL.

7.b Course Name: Express.js

Module Name: How Middleware works, Chaining of Middlewares, Types of Middlewares

In myNotes application: (i) we want to handle POST submissions. (ii) display customized error messages. (iii) perform logging.

In myNotes application:

1. we want to handle POST submissions
2. display customized error messages
3. perform logging

Using Morgan Middleware for logging

Morgan is a middleware that generates request logs in the server. The process of logging requests in an Express application is completely simplified with the usage of this middleware.

To use this middleware in the Express application, we need to install it.

```
npm install morgan
```

Consider the below code where the middleware is imported into the application and then configured to the Express application object.

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

var morgan = require('morgan');

var app = express();

app.use(morgan('combined'));


app.get('/login',myController.myMethod);
```

Morgan middleware also supports writing logs to a file.

```
var LogStream = fs.createWriteStream(path.join(__dirname, 'access.log'), { flags: 'a' });


app.use(morgan('combined', { stream: LogStream }));
```

The above code creates a write stream using the Node.js filesystem module and then the morgan middleware is attached to it. A new log file named "access.log" will be generated which contains the log details.

The above configuration of morgan middleware logs all the requests in the server console.

Now, configure the following middlewares in myNotes application:

Middleware	Purpose
body-parser	to fetch the booking details submitted by user.
Error handling	to display user friendly error messages.
Morgan	to log the requests in the server.

7.c Course Name: Express.js

Module Name: Connecting to MongoDB with Mongoose, Validation Types and

Defaults

Write a Mongoose schema to connect with MongoDB.

Organizations use various databases to store data and to perform various operations on the data based on the requirements.

Some of the most popular databases are:

- Cassandra
- MySQL
- MongoDB
- Oracle
- Redis
- SQL Server

A user can interact with a database in the following ways.

- Using query language of the respective databases' - eg: SQL
- Using an ODM(Object Data Model) or ORM(Object-Relational Model)

In this module, we will be dealing with the usage of the Object Data Model (ODM) to interact with the database.

A schema defines document properties through an object, where the key name corresponds to the property name in the collection. The schema allows you to define the fields stored in each document along with their validation requirements and default values.

To create a schema, we need to use the following lines of code:

```
1.  const mongoose = require('mongoose');
2.
3.  const schema = new mongoose.Schema({ property_1: Number, property_2: String });


4.
```

In a schema, we will define the data types of the properties in the document.

The most commonly used types in the schema are:

- String

To store string values. For e.g: employee name

- Number

To store numerical values. For e.g: employee Id

- Date

To store dates. The date will be stored in the ISO date format. For e.g: 2018-11-15T15:22:00.

- Boolean

It will take the values true or false and is generally used for performing validations. We will discuss validations in the next resource.

- ObjectId

Usually, ObjectId is assigned by MongoDB itself. Every document inserted in MongoDB will have a unique id which is created automatically by MongoDB and this is of type ObjectId.

- Array

To store an array of data, or even a sub-document array. For e.g: ["Cricket","Football"]

7.d Course Name: Express.js

Module Name: Models

Write a program to wrap the Schema into a Model object.

To use our schema definition, we need to wrap the **Schema** into a **Model** object we can work with.

The model provides an object which provides access to query documents in a named collection. Schemas are compiled into models using the **model()** method.

```
1. const Model = mongoose.model(name , schema)
```

The first argument is the singular name of the collection for which you are creating a Model.

The model() function makes a copy of the schema. Make sure that you have added everything you want to schema before calling the model().

Let us now create a model for our **myNotes** collection using the below code.

```
1. const NotesModel = mongoose.model("mynotes", myNotesSchema);
```

Here **NotesModel** is a collection object that will be used to perform CRUD operations.

Now we have created the model for our database collection, let us insert some data into the collection.

8.a Course Name: Express.js

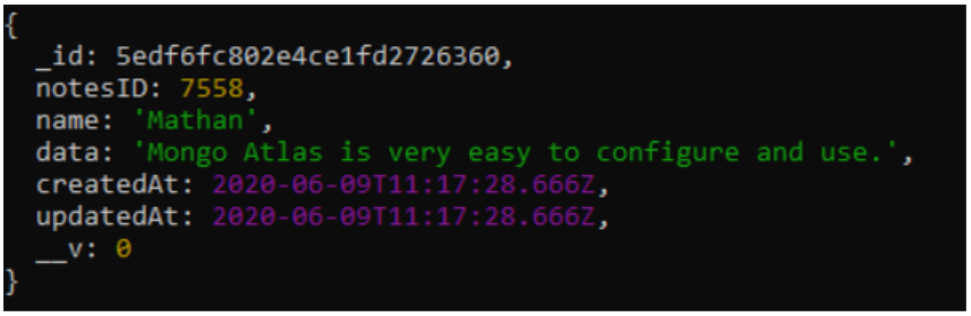
Module Name: CRUD Operations

Write a program to perform various CRUD (Create-Read-Update-Delete) operations using Mongoose library functions.

Operations - Create

```
exports.newNotes = async (req, res) => {
  try {
    const noteObj = {
      notesID: 7558,
      name: 'Mathan',
      data: 'Mongo Atlas is very easy to configure and use.',
    };
    const newNotes = await NotesModel.create(noteObj);
    console.log(newNotes);
  } catch (err) {
    console.log(err.errmsg);
  }
};
```

Output:-



Operations – Read

```
exports.getNotes = async (req, res) => {
  try {
    const notes = await NotesModel.find({}, { _id: 0, __v: 0 });
    if (notes.length > 0) {
      console.log(notes);
    }
  } catch (err) {
    console.log(err.errmsg);
  }
};
```

```
[
  {
    notesID: 7558,
    name: 'Mathan',
    data: 'Mongo Atlas is very easy to configure and use.',
    createdAt: 2020-06-09T11:17:28.666Z,
    updatedAt: 2020-06-09T11:17:28.666Z
  },
  {
    notesID: 7555,
    name: 'Mathan',
    data: 'This includes macOS package notarization and a change in configuration... ',
    createdAt: 2020-06-09T11:20:06.871Z,
    updatedAt: 2020-06-09T11:20:06.871Z
  },
  {
    notesID: 7556,
    name: 'Sam',
    data: 'AMD processor support in android studio.',
    createdAt: 2020-06-09T11:20:06.872Z,
    updatedAt: 2020-06-09T11:20:06.872Z
  },
  {
    notesID: 7557,
    name: 'Mathan',
    data: 'MERN',
    createdAt: 2020-06-09T11:20:06.872Z,
    updatedAt: 2020-06-09T11:20:06.872Z
  }
]
```


Operations – Update

```
exports.updateNotes = async (req, res) => {
  try {
    const noteObj = {
      name: 'Mathan',
      data: 'Updated notes',
    };
    const notes = await NotesModel.findOneAndUpdate(
      { notesID: 7555 },
      noteObj,
      {
        new: true, //to return new doc back
        runValidators: true, //to run the validators which specified in the model
```

```
    }
  );
  if (notes != null) {
    console.log(notes);
  }
 } catch (err) {
  console.log(err.errmsg);
 }
};
```

Output:-

```
{
  _id: 5edf706627f169388e6cc8ae,
  notesID: 7555,
  name: 'Mathan',
  data: 'Updated notes',
  createdAt: 2020-06-09T11:20:06.871Z,
  updatedAt: 2020-06-09T11:26:34.289Z,
  __v: 0
}
```

Operations – Delete

```
exports.deleteNotes = async (req, res, err) => {
 const delDet = await NotesModel.deleteOne({ notesID: 7555 });
 console.log(delDet);
};
```

Output:-

```
{ n: 1, ok: 1, deletedCount: 1 }
```

8.b Course Name: Express.js

Module Name: API Development

In the myNotes application, include APIs based on the requirements provided. (i) API

should fetch the details of the notes based on a notesID which is provided in the URL.

Test URL - http://localhost:3000/notes/7555 (ii) API should update the details bas

An application should be protected by HTTPS, even if it's not handling sensitive information for communications.

- The integrity of the application will be protected by HTTPS.
- The privacy and security of the data will be protected by HTTPS.

Generate the SSH keys in the server where the application will be running.

Configure the key and the certification and create the HTTPS server as shown in the below code:

```
const express = require('express');
const fs = require('fs');
const https = require('https');

const app = express();

app.get('/', function (req, res) {
  res.send('Welcome to HTTPS');
});

https
  .createServer(
    {
      key: fs.readFile('sshServer.key'),
      cert: fs.readFile('sshServer.cert'),
    },
    app
  )
  .listen(3000, function () {
    console.log('Server running on HTTPS');
  });


```

The application will be accessible using the URL, https://localhost:3000.

(ii) API should update the details bas

In the recently created myNotes application, include these below APIs based on the requirements provided.

1. API should fetch the details of the notes based on a notesID which is provided in the URL.

Test URL - http://localhost:3000/notes/7555

2. API should update the details based on the name which is provided in the URL and the data in the request body.

Test URL - http://localhost:3000/notes/Mathan

Note: Only one document in the collection needs to be updated.

3. API should delete the details based on the name which is provided in the URL.

Test URL - http://localhost:3000/notes/Mathan

Note: Only one document in the collection needs to be deleted

8.c Course Name: Express.js

Module Name: Why Session management, Cookies

Write a program to explain session management using cookies.

- Session management
- Use of Cookies

Download the source code for the demo here.

Demo steps:

1. Modify the app.js file in TestApp as shown below:

```
const express = require('express');
const cookieParser = require('cookie-parser');
const router = require('./Routes/routing');

const app = express();
app.use(cookieParser());
app.use('/', router);
app.listen(3000);
console.log('Server listening in port 3000');
```

```
10.
```

2. Modify the routing.js file in TestApp as shown below:

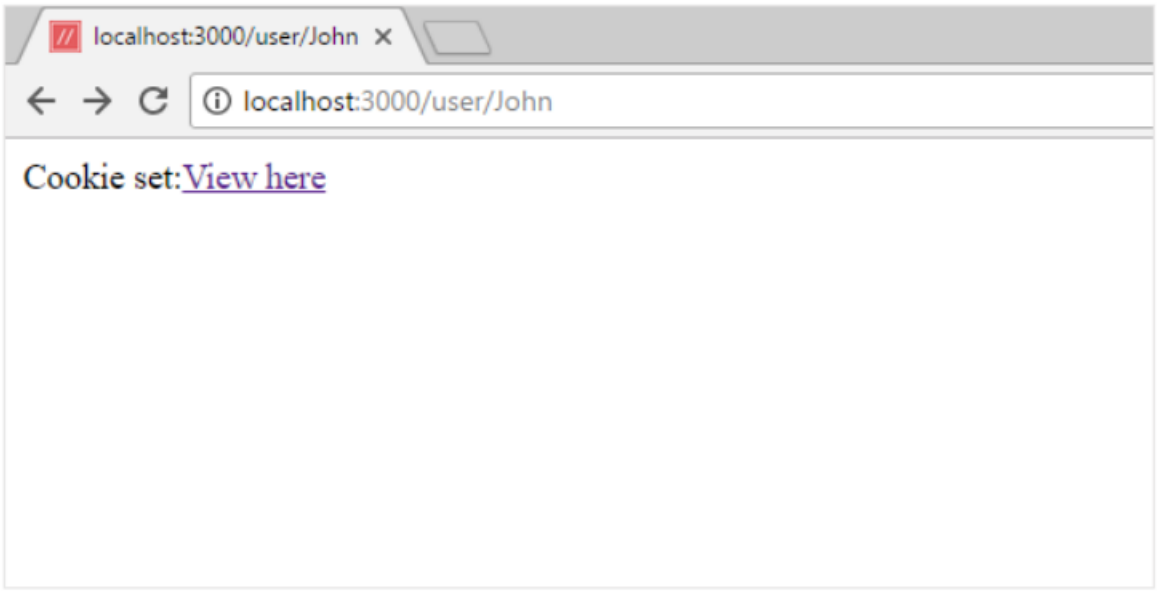
```
1.  const express = require('express');
2.  const myController = require('../Controller/myController');
3.
4.  const router = express.Router();
5.
6.  router.get('/user/:name', myController.user1);
7.  router.get('/user', myController.user2);
8.
9.  module.exports = router;
```

3. In Controller, add the below code:

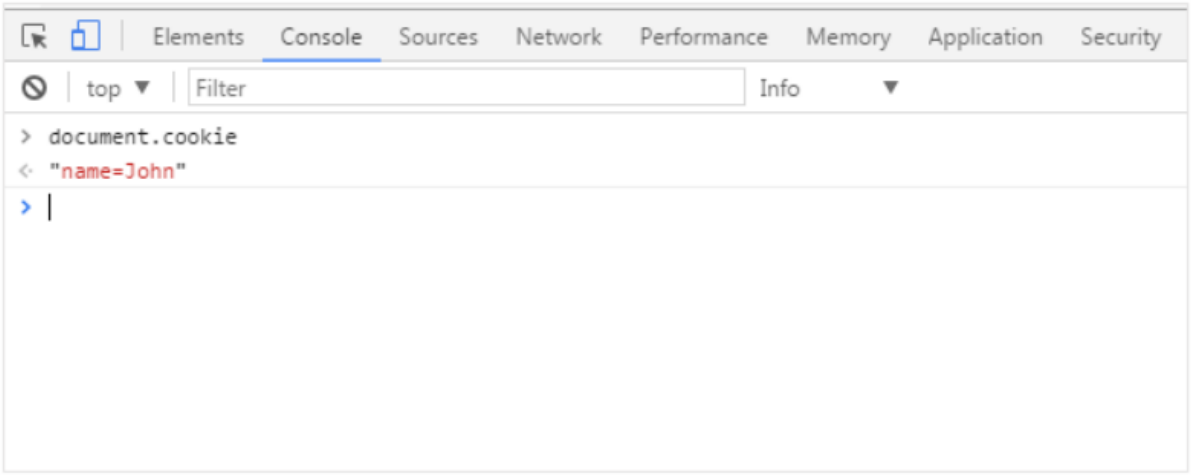
```
1.  exports.user1 = async (req, res) => {
2.    res.cookie('name', req.params.name);
3.    res.send('<p>Cookie set:<a href="/user"> View here </a>');
4.  };
5.
6.  exports.user2 = async (req, res) => {
7.    res.send(req.cookies.name);
8.  };


9.
```

4. Save the files. Start the server.

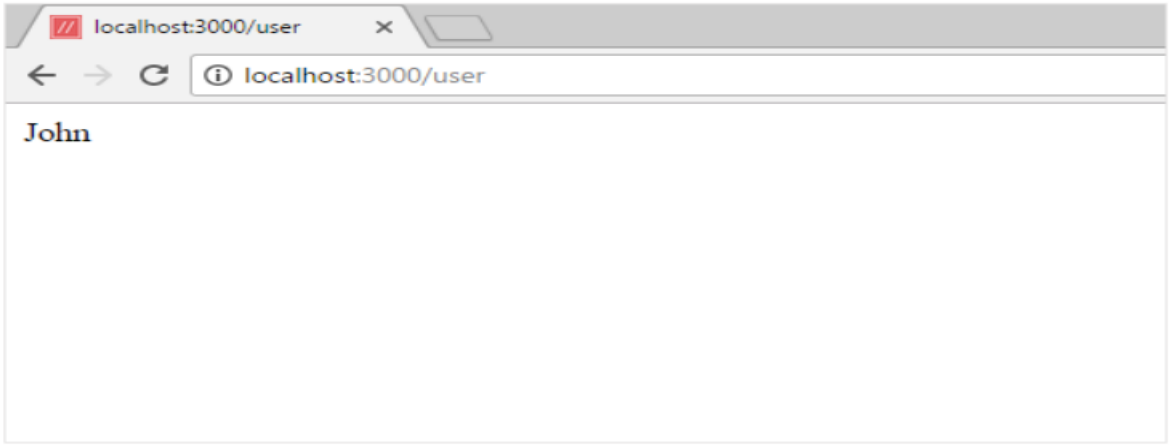
4. Access the URL "http://localhost:3000/user/<username>" and observe the output.



5. Open the developer tools in the browser window and click on the Console tab and type **document.cookie** and verify the value displayed.



6. On click of the "View here" link, the cookie is retrieved and displayed as below:



7. Open the developer tools in the browser window and click on the Console tab and type **document.cookie** and verify the value displayed.

8.d Course Name: Express.js

Module Name: Sessions

Write a program to explain session management using sessions.

- Session management
- Usage of the Session store

Demo steps:

1. Open the TestApp folder in the command prompt and run the command "npm install" for installing the dependencies mentioned in the package.json file.

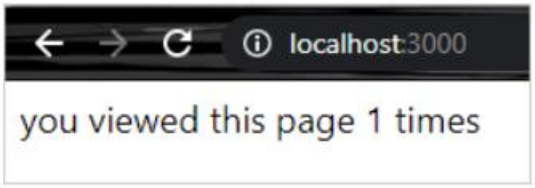
2. In the file server-session.js, use the below code:

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

const app = express();
const session = require('express-session');

app.use(
  session({
    name: 'my-session',
    cookie: {
      //Current Server time + maxAge = expire datetime
      maxAge: 1000 * 60,

```

```
      //Allows for the cookie to be read only by a server (no JS)
      httpOnly: true,

      //Set for the root of the domain.
      path: '/',
    },
    secret: 'mypasswordinfy',
  })
);

app.use((req, res, next) => {
  let { visit } = req.session;

  if (!visit) {
    visit = req.session.visit = {
      count: 1,
    };
  } else {
    visit.count++;
  }
  next();
});

app.get('/', (req, res) => {
  res.send(`you viewed this page ${req.session.visit.count} times`);
});

app.listen(3000);
console.log('Server started...running with port 3000');
```

2. Start the Server.

3. Observe the below output:



8.e Course Name: Express.js

Module Name: Why and What Security, Helmet Middleware

Implement security features in myNotes application

- Understand security vulnerabilities
- Use of Helmet middleware

Demo steps:

1. Unzip the demo folder, navigate to the folder in Node command prompt and issue npm install command to install the dependencies.

2. The app.js file before adding helmet middleware.

```
const express = require('express');
const routing = require('./route');

const app = express();
app.use('/', routing);
app.listen(3000);
console.log('Server listening in port 3000');


```

2. The route.js file content.

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

const router = express.Router();
router.get('/', function (req, res) {
  res.send('<h1>Express</h1>');
});
router.get('/about', function (req, res) {
  res.send('About Us Page');
});
module.exports = router;


```

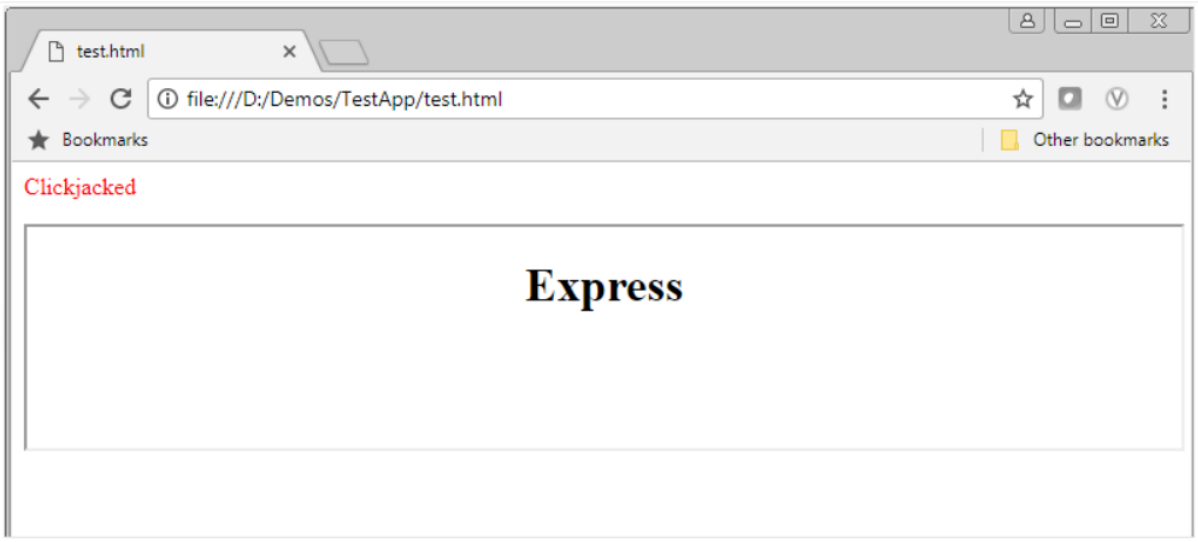
3. Create a file test.html in the TestApp folder to demonstrate a clickjacking attack.

Clickjacking is a malicious technique of tricking the user thereby rendering a page inside an iframe.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <style>
    p {
      color: red;
    }
    iframe {
      width: 100%;
      height: 90%
    }
  </style>
</head>
<body>
  <p>Clickjacked</p>
  <iframe src="http://localhost:3000"></iframe>
</body>


</html>
```

4. Open a command prompt and start the server.

5. Open the file test.html in the browser.



We can see that the page got loaded in the iframe.

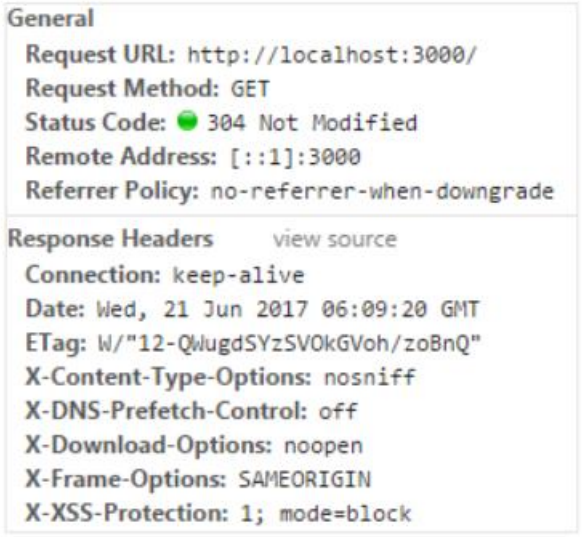
6.Modify the app.js file in TestApp by adding the helmet configuration as shown below.

```
const express = require('express');
const helmet = require('helmet');
const routing = require('./route');

const app = express();
app.use(helmet());
app.use('/', routing);
app.listen(3000);
console.log('Server listening in port 3000');


```

7. Open a command prompt and start the server.

8.Access the URL "http://localhost:3000/" and observe the output.

Now access the developer tools in browser(F12) and click on the "Network" tab and observe the headers set as part of the response from the server.



We can see that helmet sets an 'X-Frame-Options' header to value "SAMEORIGIN', which instructs the browser to not allow framing from other domains.

9. Now open the page test.html and observe the output. Press Ctrl+ F5 to clear the cache.

COURSE COMPLETION CERTIFICATE :-

(Typescript)

9.a Course Name: Typescript

Module Name: Basics of TypeScript

On the page, display the price of the mobile-based in three different colors. Instead of using the number in our code, represent them by string values like GoldPlatinum, PinkGold, SilverTitanium.

SOURCE CODE:-

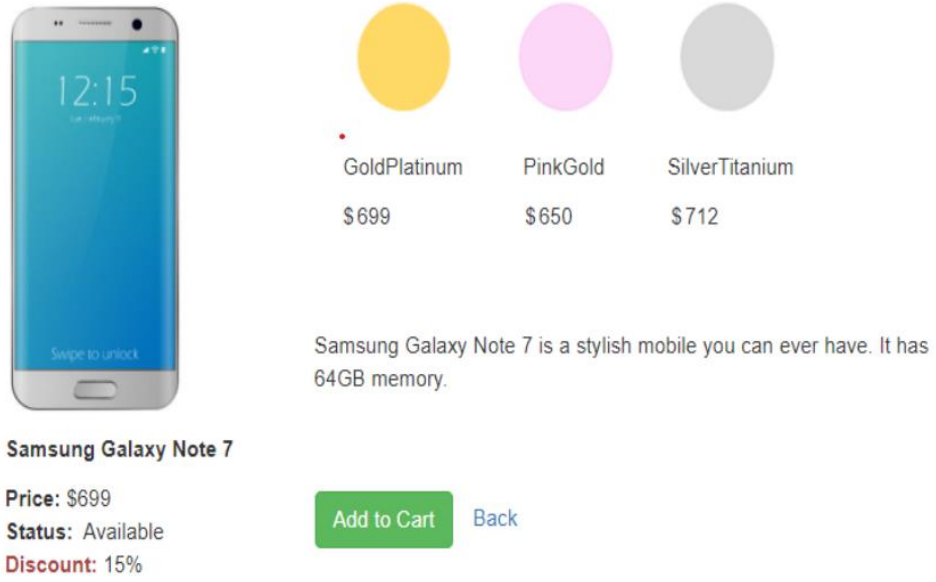
```
<!doctype html>
<html>
  <head>
      <title>Mobile Cart</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
      <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

   <style>
              .navbar-inverse {
   background-color:#005580;
  background-image: none;
  background-repeat: no-repeat;
             color:#ffffff;
 }
 .navbar-inverse .navbar-nav > .active > a {
```

```
    color: #ffffff;
    background-color:transparent;


}
.navbar-inverse .navbar-nav > li > a:hover{
            text-decoration: none;
    color: #ffffff;
}
            </style>
  </head>
  <body>
      <nav class="navbar navbar-inverse navbar-fixed-top">
       <div class="navbar-header">
         <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
           <span class="sr-only">Toggle navigation</span>
           <span class="icon-bar"></span>
           <span class="icon-bar"></span>
           <span class="icon-bar"></span>
         </button>
         <a class="navbar-brand" href="#">Mobile Cart</a>
       </div>
       <div id="navbar" class="collapse navbar-collapse">
         <ul class="nav navbar-nav">
           <li><a href="#">Home</a></li>


         </ul>
       <!--/.nav-collapse -->
       <ul class="nav navbar-nav navbar-middle" style="color:white; margin-right:30px;">
```

```
        <li><a href="Cart.html"><span class="glyphicon glyphicon-shopping-cart"
style="color:white"></span></a></li>

      </ul>

      </div>

    </nav>

    <div style="margin-top:7%">

      <center> <h2>Your Favorite Online Mobile Shop!</h2> </center>

    </div>

      <div class="container" style="padding-top:5%">

                  <div class="row">

                                                <div class="col-md-4">

                                                            <div style="text-align: center;">

                                                                        <img src="Images/Part
1/SamsungGalaxy_Gold.jpg" height="250px">

                                                            </div>

                  <div style="padding-top:10px;">

                                                            <div
style="cursor:pointer;color:Steelblue;text-align: center;"><b>

                        <span id="pName"></span></b></div>

                                                            <div style="padding-
top:10px;padding-left: 101px;"><b>Price:</b> &dollar;<span
id="pPrice"></span></div>

                        <div style="padding-left: 100px;"><b>Status:</b><span
id="pAvailable"></span></div>

                  </div>

                                          </div>


    </div>

    </div>

  </body>
```

```
<!--Adding the converted js file -->
  <script src="productlist.js"></script>
</html>
```

OUTPUT:-



9.b Course Name: Typescript

Module Name: Function

Define an arrow function inside the event handler to filter the product array with the selected product object using the productId received by the function. Pass the selected product object to the next screen.

Function in TypeScript Vs JavaScript:

- A function is a block of statements to perform a particular task.
- A sequence of statements written within function forms function body.
- Functions are executed when it is invoked by a function call. Values can be passed to a function as function parameters and the function returns a value.
- Functions in TypeScript are like functions in JavaScript with some additional features.

	TypeScript	JavaScript
Types:	Supports	Do not support
Required and optional parameters:	Supports	All parameters are optional
Function overloading:	Supports	Do not support
Arrow functions:	Supports	Supported with ES2015
Default parameters:	Supports	Supported with ES2015
Rest parameters:	Supports	Supported with ES2015

9.c Course Name: Typescript

Module Name: Parameter Types and Return Types

Consider that developer needs to declare a function - getMobileByVendor which accepts string as input parameter and returns the list of mobiles**.**

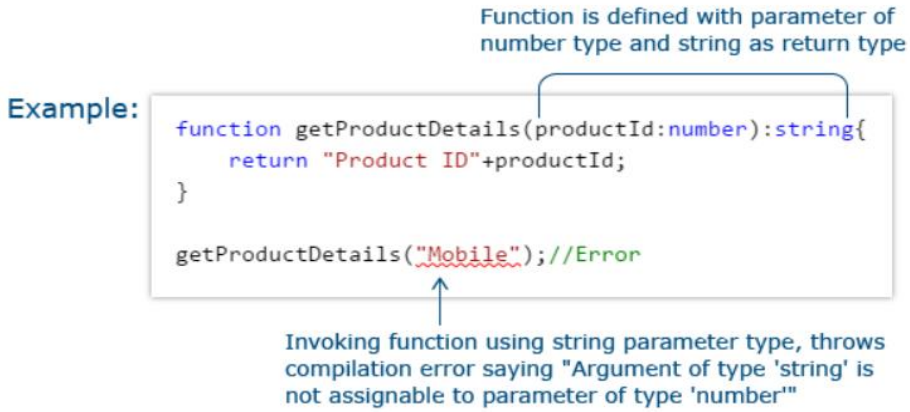
SOURCE CODE:-

```
// declaring a function which accepts string datatype as parameter and returns string array
function getMobileByManufacturer(manufacturer: string): string[] {

    let mobileList: string[];

    if (manufacturer === 'Samsung') {
        mobileList = ['Samsung Galaxy S6 Edge', 'Samsung Galaxy Note 7',
'Samsung Galaxy J7 SM-J700F'];
        return mobileList;
    } else if (manufacturer === 'Apple') {
    mobileList = ['Apple iPhone 5s', 'Apple iPhone 6s ', 'Apple iPhone 7'];
    return mobileList;
    } else {
    mobileList = ['Nokia 105', 'Nokia 230 Dual Sim'];
```

```
    return mobileList;
    }
}
```

// logic to populate the Samsung manufacturer details on console

console.log('The available mobile list: ' + getMobileByManufacturer('Samsung'));



9.d Course Name: Typescript

Module Name: Arrow Function

Consider that developer needs to declare a manufacturer's array holding 4 objects with id and price as a parameter and needs to implement an arrow function - myfunction to populate the id parameter of manufacturers array whose price is greater

SOURCE CODE:-

```
// declaring an Array with 3 objects
const manufacturers = [{ id: 'Samsung', price: 150 },
        { id: 'Microsoft', price: 200 },
        { id: 'Apple', price: 400  },
        { id: 'Micromax', price: 100  }
    ];
```

```
let test;

// Arrow function to populate the details of Array whose price is greater than 200
function myFunction() {
  test = manufacturers.filter((manufacturer) => manufacturer.price >= 200);
}

// self-invoking an arrow function
myFunction();

console.log('Details of Manufacturer array are : ');

// logic to populate the manufacturer array details based on id value
for (const item of test) {
    console.log(item.id);
}
```

9.e Course Name: Typescript

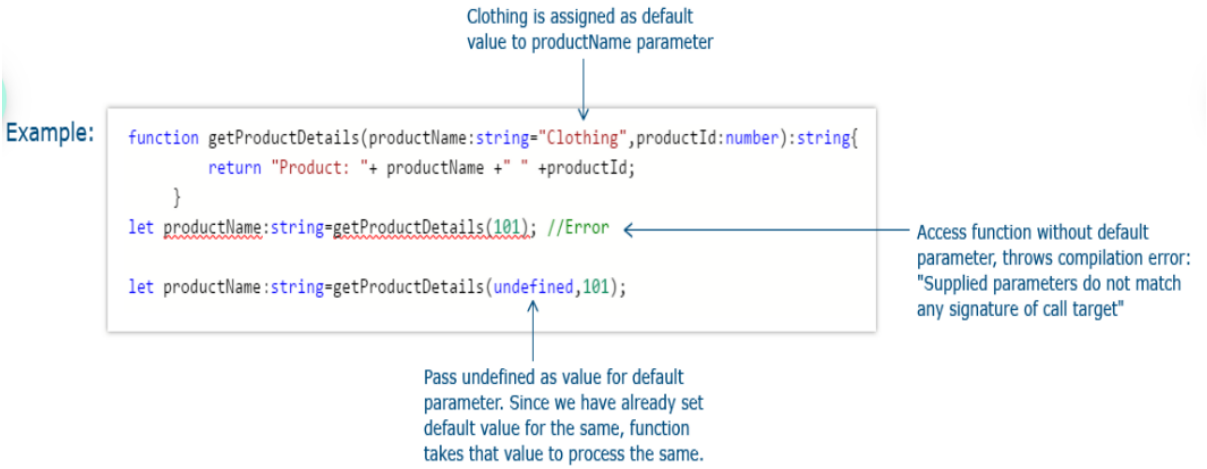
Module Name: Optional and Default Parameters

Declare a function - getMobileByManufacturer with two parameters namely manufacturer and id, where manufacturer value should passed as Samsung and id parameter should be optional while invoking the function, if id is passed as 101 then this function

```
// declaring a function with optional parameter
function getMobileByManufacturer(manufacturer: string = 'Samsung', id?: number): string[]
{

    let mobileList: string[];
```

```
  // logic to be evaluated if id parameter while invoking above declared function
   if (id) {
   if (id === 101) {
      mobileList = ['Moto G Play, 4th Gen', 'Moto Z Play with Style Mod'];
      return mobileList;
      }
   }

   // logic to return mobileList based on manufacturer category
   if (manufacturer === 'Samsung') {
        mobileList = [' Samsung Galaxy S6 Edge', ' Samsung Galaxy Note 7',
 ' Samsung Galaxy J7 SM-J700F'];
        return mobileList;
     } else if (manufacturer === 'Apple') {
          mobileList = [' Apple iPhone 5s', ' Apple iPhone 6s', ' Apple iPhone 7'];
          return mobileList;
     } else {
          mobileList = [' Nokia 105', ' Nokia 230 Dual Sim'];
          return mobileList;
     }

  }

  // statement to invoke function with optional parameter
console.log('The available mobile list : ' + getMobileByManufacturer('Apple'));

  // statement to invoke function with default parameter
```

console.log('The available mobile list : ' + getMobileByManufacturer(undefined, 101));



10.a Course Name: Typescript

Module Name: Rest Parameter

Implement business logic for adding multiple Product values into a cart variable which is type of string array.

```
// declaring a empty string array
const cart: string[] = [];

// arrow function logic to push values into cart array
const pushtoCart = (item: string) => { cart.push(item); };

// logic to add items into cart
function addtoCart(...productName: string[]): string[] {

    for (const item of productName) {
        pushtoCart(item);
    }
    return cart;
```

```
}

// to populate value on console
console.log('Cart Items are:' + addtoCart(' Moto G Play, 4th Gen', ' Apple iPhone 5s'));
```

10.b Course Name: Typescript

Module Name: Creating an Interface

Declare an interface named - Product with two properties like productId and productName with a number and string datatype and need to implement logic to populate the Product details.

```
// declaring an interface
interface Product {
   productId: number ;
   productName: string ;
}
```

```
// logic to display the Product details with interface object as parameter
function getProductDetails(productobj: Product): string {
    return 'The product name is : ' + productobj.productName;
}

// declaring a variable having interface properties
const prodObject = {productId: 1001, productName: 'Mobile'};

// declaring variable and invoking Product details function
const productDetails: string = getProductDetails(prodObject);

// line to populate the created product on console
console.log(productDetails);
```

Syntax:

```
interface Interfacename{
    properties;
    method declarations;
}
```

Example:

```
interface Product{
    productId:number;
    productName:string;
}

function getProductDetails(productobj:Product):string{
}

//Correct way of using the interface type
let productobj={productId:1001,productName:'Mobile'};

//InCorrect way of using the interface type
let productobj={productCategory:'Gadget',productName:'Mobile'};

getProductDetails(productobj);
```

Declaring interface with name Product. Function or object which is going to use this interface should contain properties declared in interface

Passing object with type Product interface as parameter to function. If the object passed does not have any of the properties declared in the Interface it throws compilation error

As this declaration does not have productId property of interface, it throws compilation error

10.c Course Name: Typescript

Module Name: Duck Typing

Declare an interface named - Product with two properties like productId and productName with the number and string datatype and need to implement logic to populate the Product details.

SOURCE CODE:-

```
// declaring an interface
interface Product {
    productId: number;
    productName: string;
}

// logic to display the Product details with interface object as parameter
// tslint:disable-next-line:adjacent-overload-signatures
function getProductDetails(productobj: Product): string {
    return 'The product name is : ' + productobj.productName;
}

// declaring a variable along with interface properties
const prodObject = {productId: 1001, productName: 'Mobile', productCategory: 'Gadget'};

// declaring variable and invoking Product details function
const productDetails: string = getProductDetails(prodObject);

// line to populate the created product variable on console
console.log(productDetails);
```

Example:

```
interface Product{
    productId:number;
    productName:string;
}

function getProductDetails(productobj:Product):string{


}
//InCorrect way of using the interface duck type
let prodObject={productName:'Mobile',productCategory:'Gadget'};

//Correct way of using the interface duck type
let prodObject={productId:1001,productName:'Mobile',productCategory:'Gadget'};

getProductDetails(prodObject);
```

Adding an additional property productCategory to demonstrate Duck Typing

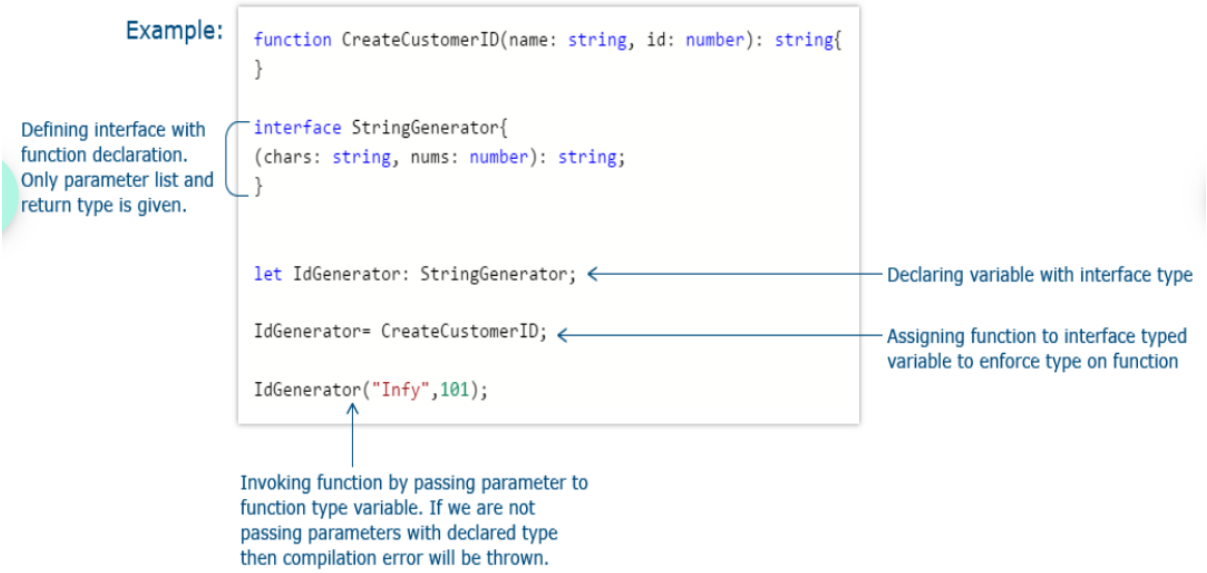
10.d Course Name: Typescript

Module Name: Function Types

Declare an interface with function type and access its value.

```
// declaring a function
function CreateCustomerID(name: string, id: number): string {
    return 'The customer id is ' + name + ' ' + id;
    }

    // declaring a interface with function type
interface StringGenerator {

    (chars: string, nums: number): string;
    }

    // creating reference variable of above declared interface
let idGenerator: StringGenerator;

    // assignment of function to interface reference variable
```

```
idGenerator = CreateCustomerID;

    // declaring a string parameter to hold return value of function type interface
const customerId: string = idGenerator('Mr.Tom', 101);

    // line to populate the Customer Details
console.log(customerId);
```



11.a Course Name: Typescript

Module Name: Extending Interfaces

Declare a productList interface which extends properties from two other declared interfaces like Category,Product as well as implementation to create a variable of this interface type.

```
// declaring a Category interface
interface Category {
    categoryName: string;
```

```
}

// declaring a Product interface
interface Product {
    productName: string;
    productId: number;
}

// declaring a ProductList interface which is extends from Category and Product interfaces
interface ProductList extends Category, Product {
    list: Array<string>;
}

// declaring a variable which is type of ProductList interface
const productDetails: ProductList = {
    categoryName: 'Gadget',
    productName: 'Mobile',
    productId: 1234,
    list: ['Samsung', 'Motorola', 'LG']
};

// assigning list value of productDetails variable into another variable
const listProduct = productDetails.list;

// assigning productName value of productDetails variable into another variable
const pname: string = productDetails.productName;

// line to populate Product name
```

console.log('Product Name is ' + pname);

// line to populate Product list

console.log('Product List is ' + listProduct);

Example:

```
interface Category{
    categoryName:string;
}

interface Product{
    productName:string;
    productId:number;
}
interface productList extends Category,Product{
    list:Array<string>;
}

let productDetails:productList={
    categoryName:'Gadget',
    productName:'Mobile',
    productId:1234,
    list:['Samsung','Motorola','LG']
}
```

Extending productList interface from two other interfaces Category and Product

Creating object literal productDetails with productList interface type. Throw compilation error, if we do not define properties declared in super interfaces.

11.b Course Name: Typescript

Module Name: Classes

Consider the Mobile Cart application, Create objects of the Product class and place them into the productlist array.

- Class is a template from which objects can be created.
- It provides behavior and state storage.
- It is meant for implementing reusable functionality.
- Use a class keyword to create a class.

Example:

Class name

```
class Product{
    productId:number;
    getProductDetails(productId :number):string{
        return "product id is"+productId;
    }
}
```

Member variable declared in class

Function defined in class

11.c Course Name: Typescript

Module Name: Constructor

Declare a class named - Product with the below-mentioned declarations: (i) productId as number property (ii) Constructor to initialize this value (iii) getProductId method to return the message "Product id is <<id value>>".

```
// declaring a Product class
class Product {

    static productPrice: string;
    productId: number;
// constructor declaration
    constructor(productId: number) {
        this.productId = productId;
    }
    getProductId(): string {
        return 'Product id is : ' + this.productId;
    }
}

// creation of Product class object
const product: Product = new Product(1234);

// line to populate the product id details
console.log(product.getProductId());
```

Example:

```
class Product{
    productId:number;
    constructor(productId:number){
        this.productId=productId;
    }
    getProductId():string{
    }
}
var product:Product=new Product(1234);
```

Parameterized constructor is used to initialize productId property

Creating instance of class with parameterized constructor

11.d Course Name: Typescript

Module Name: Access Modifiers

Create a Product class with 4 properties namely productId, productName, productPrice, productCategory with private, public, static, and protected access modifiers and accessing them through Gadget class and its methods.

```
// declaring a Product class with access modifiers
class Product {
    static productPrice = 150;
    private productId: number;
    public productName: string;
    protected productCategory: string;
    // declaration of constructor with 3 parameters
    constructor(productId: number, productName , productCategory) {
        this.productId = productId;
        this.productName = productName;
        this.productCategory = productCategory;
    }
    // method ot display product id details
    getProductId() {
```

```
        console.log('The Product id is : ' + this.productId);
    }
}
// declaring a Gadget class extending the properties from Product class
class Gadget extends Product {
    // method to display productCategory property
    getProduct(): void {
        console.log('Product category is : ' + this.productCategory);
    }
}
// Gadget Class object creation
const g: Gadget = new Gadget(1234, 'Mobile', 'SmartPhone');
// invoking getProduct method with the help of Gadget object
g.getProduct();

// invoking getProductId method with the help of Gadget object
g.getProductId();

// line to populate product name property with Gadget object
console.log('Product name is : ' + g.productName);
// line to populate static property product price directly with Product class name
console.log('Product price is : $' + Product.productPrice);
```

Example:

```
class Product{
    public productId:number;
    constructor(productId:number){
        this.productId=productId;
    }
}
var product:Product=new Product(1234);
console.log(product.productId);
```

Declaring productId property using public keyword

Accessing productId outside class since it is declared using public keyword

```
class Product{
    private productId:number;
    constructor(productId:number){
        this.productId=productId;
    }
}
var product:Product=new Product(1234);
console.log(product.productId); //Error
```

Declaring productId property using private keyword

Accessing productId outside class. Since it is declared using private keyword it is not accessible outside class. This line throws compilation error.

12.a Course Name: Typescript

Module Name: Properties and Methods

Create a Product class with 4 properties namely productId and methods to

setProductId() and getProductId().

Instead of declaring instance variables and then passing parameters to the constructor and initializing it, you can reduce the code by adding access specifiers to the parameters passed to the constructor.

Consider the below code:

Example:

```
class Product{
    private productId:number;
    constructor(productId:number){
        this.productId=productId;
    }
}
```

Passing parameter to constructor and setting private property with value passed to constructor

Instead of this declare the parameter itself with any of the access modifiers and reduce the lines of code used for the initialization.

Let us rewrite the above code:

Example:

```
class Product{
    constructor(private productId:number){
    }
}
```

Initial line of declaration and setting of value inside constructor is removed and parameter is declared with private keyword. The same is applicable for public or protected keywords as well.

12.b Course Name: Typescript

Module Name: Creating and using Namespaces

Create a namespace called ProductUtility and place the Product class definition in it.

Import the Product class inside productlist file and use it.

```
<!doctype html>
<html>
<head>
    <title>Mobile Cart</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
    <style>
```

```
        .navbar-inverse {
            background-color: #005580;
            background-image: none;
            background-repeat: no-repeat;
            color: #ffffff;
        }
        .navbar-inverse .navbar-nav>.active>a {
            color: #ffffff;
            background-color: transparent;
        }
        .navbar-inverse .navbar-nav>li>a:hover {
            text-decoration: none;
            color: #ffffff;
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar"
                aria-expanded="false" aria-controls="navbar">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
```

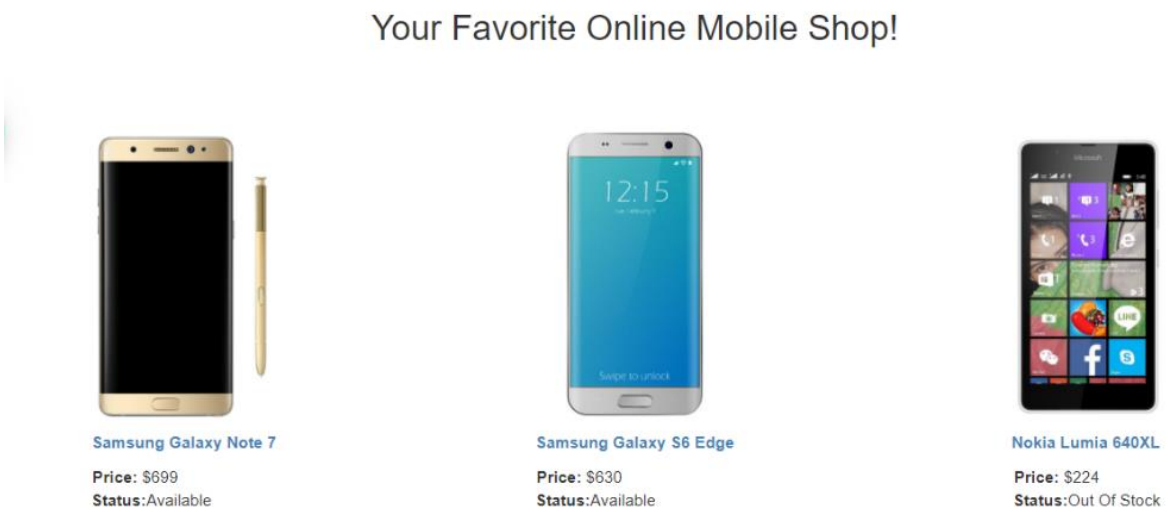
```
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="#">Mobile Cart</a>
        </div>
        <div id="navbar" class="collapse navbar-collapse">
          <ul class="nav navbar-nav">
            <li><a href="#">Home</a></li>
          </ul>
          <!--/.nav-collapse -->
          <ul class="nav navbar-nav navbar-middle" style="color:white; margin-right:30px;">
            <li><a href="Cart.html"><span class="glyphicon glyphicon-shopping-cart" style="color:white"></span></a>
            </li>
          </ul>
        </div>
      </nav>
      <div style="margin-top:7%">
        <center>
          <h2>Your Favorite Online Mobile Shop!</h2>
        </center>
      </div>
      <div class="container" style="padding-top:5%">
        <div class="row">
```

```
<div class="col-md-4">
    <div style="text-align: center;">
        <img src="images/SamsungGalaxy_Gold.jpg" height="250px">
    </div>
    <div style="padding-top:10px;">
        <div onclick="getMobileDetails();" style="cursor:pointer;color:Steelblue;text-align: center;">
            <b><span id="pName0">Samsung Galaxy Note 7</span></b>
        </div>
        <div style="padding-top:10px;padding-left: 101px;"><b>Price:</b> &dollar;<span
                id="pPrice0">699</span></div>
        <div style="padding-left: 100px;"><b>Status:</b><span id="pAvailable0">Available</span></div>
    </div>
</div>
<div class="col-md-4">
    <div style="text-align: center;">
        <img src="images/samsung_edge_silver.jpg" height="250px">
    </div>
    <div style="padding-top:10px;">
        <div onclick="getMobileDetails('samsungedge',231);"
            style="cursor:pointer;color:Steelblue;text-align: center;"><b><span id="pName1">Samsung Galaxy
                S6 Edge</span></b></div>
```

```
            <div style="padding-top:10px;padding-left: 95px;"><b>Price:</b> &dollar;<span
                id="pPrice1">630</span></div>
            <div style="padding-left: 94px;"><b>Status:</b><span id="pAvailable1">Available</span></div>
        </div>
      </div>
      <div class="col-md-4">
        <div style="text-align: center;">
          <img src="images/lumia_640xl.jpg" height="250px">
        </div>
        <div style="padding-top:10px;">
          <div onclick="getMobileDetails('lumia',875);"
            style="cursor:pointer;color:Steelblue;text-align: center; "><b><span id="pName2">Nokia Lumia
              640XL</span></b></div>
          <div style="padding-top:10px;padding-left: 118px;"><b>Price:</b> &dollar;<span
              id="pPrice2">224</span></div>
          <div style="padding-left: 117px;"><b>Status:</b><span id="pAvailable2">Out of Stock!</span></div>
        </div>
      </div>
    </div>
  </div>
</body>
```

```
<!-- Adding the converted js file -->

<script src="productlist.js"></script>

</html>
```

OUTPUT:-



12.c Course Name: Typescript

Module Name: Creating and using Modules

Consider the Mobile Cart application which is designed as part of the functions in a module to calculate the total price of the product using the quantity and price values and assign it to a totalPrice variable.

- Creating a module
- Importing a module

Demo steps:

Step 1: Create a file module_demo.ts

```
export function MaxDiscountAllowed(noOfProduct: number): number {
    if (noOfProduct > 5) {
        return 30;
    } else {
        return 10;
    }
}
class Utility {
    CalculateAmount(price: number, quantity: number): number {
        return price * quantity;
    }
}
interface Category {
    getCategory(productId: number): string;
}
export const productName = 'Mobile';
export {Utility, Category};
```

Step 2: Create another file 2.module_import.ts

```
import { Utility as mainUtility, Category, productName, MaxDiscountAllowed } from
"./module_demo";
const util = new mainUtility();
const price = util.CalculateAmount(1350, 4);
const discount = MaxDiscountAllowed(2);
console.log(`Maximum discount allowed is: ${discount}`);
console.log(`Amount to be paid: ${price}`);
console.log(`ProductName is: ${productName}`);
```

Step 3: Open a command prompt and navigate to the folder in which module files resides and run the below command from the command line

Step 4: Run **node 2.module_import.js** from the command line

```
D:\courseware\Typescript\ILP\demos\modules>node 2.module_import.js
Maximum discount allowed is: 10
Amount to be paid: 5200
ProductName is: Mobile
```

12.d Course Name: Typescript
Module Name: What is Generics, What are Type Parameters, Generic Functions, Generic Constraints
Create a generic array and function to sort numbers as well as string values.

Generics is a concept using which we can make the same code work for multiple types.

It accepts type parameters for each invocation of a function or a class or an interface so that the same code can be used for multiple types.

Consider the below code where you implement a similar function for two different types of data:

```
function printString(stringData:string):string{
return stringData;
}
```

```
function printNumber(numberData:number):number{
return numberData;
}
```

```
function printData<T>(data:T):T{
return data;
}
```

<T> represents type parameter

It generalizing type of parameter and function return type.

Same code works for number or string or any other type of parameter.

Type Parameters are used to specify the type, a generic will work over.

They are listed using an angle bracket<>.

The actual type will be provided while invoking function or instance creation.

Consider a generic function given below:

```
function printData<T>(data:T):T{
return data;
}
```

To access this function with different types you will use the below code:

```
let data = printData<string>('Hello Generics');
console.log(data); //string

let data = printData('Hello Generics');
console.log(data); //string

let data1 = printData(123);
console.log(data1); //number
```

Invoking function using type parameter by passing type inside the <> bracket. In this case, type of parameter and function return type will be dependent on type parameter

Invoking generic function without type parameter. Here type is decided depending on parameter type.

Invoking generic function with number type parameter

Generic Functions

```
// declaring Generic function named printData
function printData<T>( data: T): T {
    return data;
    }

// variable declaration to access Generic function
const stringData: string = printData<string>('Hello Generics');

// line to populate the result of Generic function on console.
console.log('String data ' + stringData);
```

Generic Constraints

```
// declaring AddLength interface
interface AddLength {
    length: number;
}

// declaring orderLength method with Generic constraints
function orderLength<T extends AddLength>(arg: T): T {
    const lengthValue = arg.length;
    console.log('Length is ' + lengthValue);
    return arg;
}

// declaring a class Product implementing AddLength interface
class Product implements AddLength {
    length = 10;
```

```
}

// variable of Product class
const product: Product = new Product();

// creation of variable which holds the return value of orderLength method
const product1 = orderLength(product);

// line to populate the length of Product class on console
console.log('Product Length ' + product1.length);
```

COURSE COMPLETION CERTIFICATE:-