



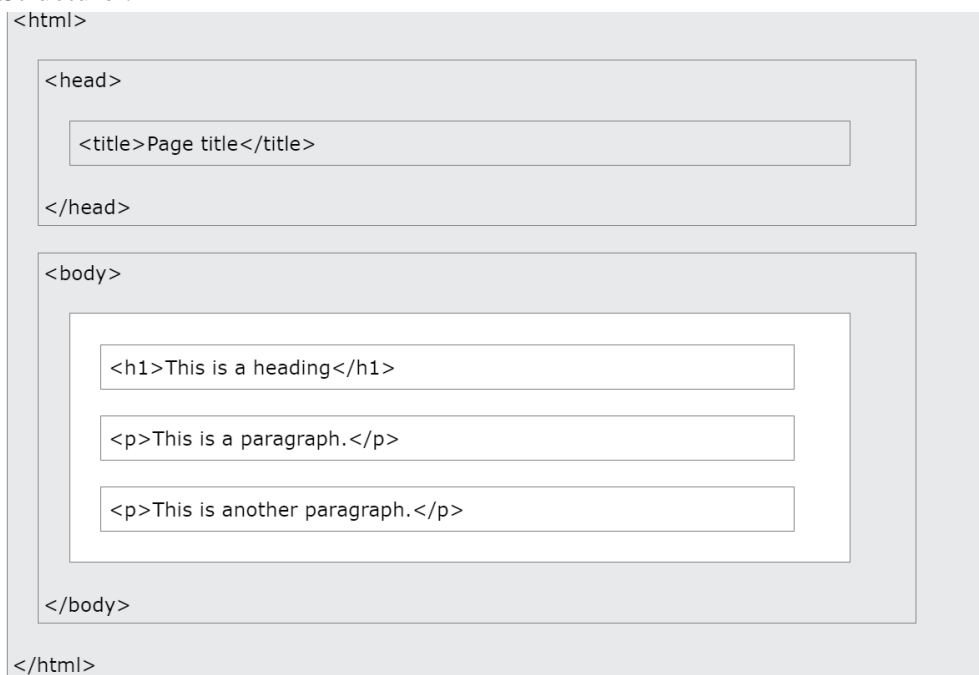
Exercise - 1

1.a) HTML-5 Module Name: Case-insensitivity, Platform-independency, DOCTYPE Declaration, Types of Elements, Html elements – Attributes, Metadata Elements

HTML5:

HTML5 is the next major revision of the HTML standard superseding HTML 4.01, XHTML 1.0, and XHTML 1.1. HTML5 is a standard for structuring and presenting content on the World Wide Web.

Html Page Structure :



HTML tags are not case sensitive: <P> means the same as <p>.

The HTML standard does not require lowercase tags, but W3C recommends lowercase in HTML, and demands lowercase for stricter document types like XHTML.

The DOCTYPE declaration is an instruction to the web browser about what version of HTML the page is written in. This ensures that the web page is parsed the same way by different web browsers.

Case-insensitivity: HTML is case-insensitive, meaning that tags, attributes, and attribute values can be written in either uppercase or lowercase letters. For example, <P> and <p> both represent a paragraph element in HTML.

DOCTYPE Declaration: The DOCTYPE declaration is a required element in HTML that specifies the type of HTML document. It is used by web browsers to determine which version of HTML to use when rendering a page.

Types of Elements: HTML elements can be divided into two types: block-level elements and inline elements. Block-level elements start on a new line and take up the full width of their parent container, while inline elements are placed within the flow of the text and only take up as much width as necessary.

HTML5 Elements: HTML5 elements are marked up using start tags and end tags. Tags are delimited using angle brackets with the tag name in between.



The difference between start tags and end tags is that the latter includes a slash before the tag name.

Following is the example of an HTML5 element –

`<p>...</p>`

HTML5 tag names are case insensitive and may be written in all uppercase or mixed case, although the most common convention is to stick with lowercase.

Most of the elements contain some content like `<p>...</p>` contains a paragraph. Some elements, however, are forbidden from containing any content at all and these are known as void elements. For example, **br**, **hr**, **link**, **meta**, etc.

Type of elements:

- Audio Element
- Canvas Element
- Input Types
- Output Element
- Progress Element
- SVG Element(Scalable Vector Graphics (SVG))
- Video Element

Source : <https://docs.aws.amazon.com/silk/latest/developerguide/html5-elements.html#canvas-element>

HTML5 ATTRIBUTES

Elements may contain attributes that are used to set various properties of an element.

Some attributes are defined globally and can be used on any element, while others are defined for specific elements only. All attributes have a name and a value and look like as shown below in the example.

Following is the example of an HTML5 attribute which illustrates how to mark up a div element with an attribute named class using a value of "example" –

`<div class = "example">...</div>`

Attributes may only be specified within start tags and must never be used in end tags.

HTML5 attributes are case insensitive and may be written in all uppercase or mixed case, although the most common convention is to stick with lowercase.

The basic Structure of html5 document

The following tags have been introduced for better structure –

- **section** – This tag represents a generic document or application section. It can be used together with h1-h6 to indicate the document structure.



- **article** – This tag represents an independent piece of content of a document, such as a blog entry or newspaper article.
- **aside** – This tag represents a piece of content that is only slightly related to the rest of the page.
- **header** – This tag represents the header of a section.
- **footer** – This tag represents a footer for a section and can contain information about the author, copyright information, et cetera.
- **nav** – This tag represents a section of the document intended for navigation.
- **dialog** – This tag can be used to mark up a conversation.
- **figure** – This tag can be used to associate a caption together with some embedded content, such as a graphic or video.

Metadata Element: Metadata is data that provides information about other data. In HTML, metadata elements provide information about the document and are placed in the <head> section of the HTML code. Examples of metadata elements include the <meta> tag, which provides information such as keywords and a description of the page, and the <title> tag, which provides the title of the page that is displayed in the browser's title bar.

Example Program for above basic tags:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset = "utf-8">
    <title>...</title>
  </head>
  <body>
    <header role = "banner">
      <h1>HTML5 Document Structure Example</h1>
      <p>This page should be tried in safari, chrome or Mozilla.</p>
    </header>
    <nav>
      <ul>
        <li><a href = "http://www.acet.ac.in/" target="_blank" >ACET</a></li>
        <li><a href = "http://www.acet.ac.in/?p=CSE" target="_blank" >CSE</a></li>
        <li><a href = "http://www.acet.ac.in/?p=IT" target="_blank">
          IT</a></li>
      </ul>
    </nav>
    <article>
      <section>
        <p>Once article can have multiple sections</p>
      </section>
    </article>
    <aside>
      <p>This is aside part of the web page</p>
    </aside>
```



```
<footer>
  <p>Created by <a href = "">veerendra</a></p>
</footer>
</body>
</html>
```

Tags in Exercises-1 a)

<!DOCTYPE>=> The HTML document type declaration, also known as DOCTYPE , is **the first line of code required in every HTML or XHTML document**. The DOCTYPE declaration is an instruction to the web browser about what version of HTML the page is written in

Metatag=> the meta tag defines the meta data about html document

Metatag always in side the <head> element and meta tag used to specify the character set page description , keywords ,author of the document and viewport settings

We have two important attributes for meta tag

1. name
2. content
3. http-equiv

1. name

we have 4 important values for name attributes those are description, keywords , author ,viewport
description => description of web page (Information about webpage)

keywords => keyword for define the search engine

author => define the author's name

viewport => to make webpage device compatibility

2. Content

Specifies the value associated with the http-equiv or name attribute

3. http-equiv

Provides an HTTP header for the information/value of the content attribute

Example of viewport

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

This gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

Initial-scale range 1.0 to 10

1b) Sectioning elements

Sectioning elements in HTML refer to HTML elements that define the structure and organization of a web page. These elements are used to divide the content into sections and allow accessibility features such as navigation and headings.

The main sectioning elements in HTML are:

<header>: used to define a header for a document or section

<nav>: used to define navigation links

<main>: used to define the main content of a document



`<article>`: used to define an independent, self-contained article
`<section>`: used to define a standalone section of a document
`<aside>`: used to define content that is tangentially related to the main content
`<footer>`: used to define a footer for a document or section.

Using these sectioning elements correctly can improve the accessibility and structure of your web pages.

1C) Module Name: Paragraph Element, Division and Span Elements, List Element

Paragraph Element:

The `<p>` tag **defines a paragraph**. Browsers automatically add a single blank line before and after each `<p>` element.

Example : `<p> This is Element P</p>`

Division:

The `div` tag is known as Division tag. The `div` tag is used in HTML to make divisions of content in the web page like (text, images, header, footer, navigation bar, etc). `Div` tag has both open(`<div>`) and closing (`</div>`) tag and it is mandatory to close the tag. The `Div` is the most usable tag in web development because it helps us to separate out data in the web page and we can create a particular section for particular data or function in the web pages.

`<div>`

`<h1>Aditya College of Engineering and Technology</h1>`

`<h2>Aditya College of Engineering </h2>`

`<h3>Aditya Engineering College </h3>`

`</div>`

Span:

The `` tag is **an inline container used to mark up a part of a text, or a part of a document**.

The `` tag is easily styled by CSS or manipulated with JavaScript using the class or id attribute. The `` tag is much like the `<div>` element, but `<div>` is a block-level element and `` is an inline element.

` sample text `

List Element:

HTML Lists are used to specify lists of information. All lists may contain one or more list elements.

There are three different types of HTML lists:

1. Ordered List or Numbered List (`ol`)
2. Unordered List or Bulleted List (`ul`)

HTML ORDERED LIST OR NUMBERED LIST

In the ordered HTML lists, all the list items are marked with numbers by default. It is known as numbered list also. The ordered list starts with `` tag and the list items start with `` tag.

``

`ACET`

`AEC`

`ACOE`



Output:

1. ACET
2. AEC
3. ACOE

HTML UNORDERED LIST OR BULLETED LIST

In HTML Unordered list, all the list items are marked with bullets. It is also known as bulleted list also. The Unordered list starts with tag and list items start with the tag.

ACET

AEC

ACOE

Output:

- ACET
- AEC
- ACOE

CSS (cascading style sheets)

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media

Types of CSS

Cascading Style Sheet(CSS) is used to set the style in web pages that contain HTML elements. It sets the background color, font-size, font-family, color, ... etc property of elements on a web page.

There are three types of CSS which are given below:

- Inline CSS
- Internal or Embedded CSS
- External CSS

Inline CSS: Inline CSS contains the CSS property in the body section attached with element is known as inline CSS. This kind of style is specified within an HTML tag using the style attribute.

```
<p style = "color:#009900; font-size:50px;  
font-style:italic; text-align:center;">  
Aditya  
</p>
```



Internal or Embedded CSS: This can be used when a single HTML document must be styled uniquely. The CSS rule set should be within the HTML file in the head section i.e the CSS is embedded within the HTML file.

```
<head>
  <title>Internal CSS</title>
  <style>
    .ptag {
      font-style:bold;
      font-size:20px;
    }
  </style>
</head>

<body>
  <div class ="ptag">
    A computer science Engineering
  </div>
</body>
```

External CSS: External CSS contains separate CSS file which contains only style property with the help of tag attributes (For example class, id, heading, ... etc). CSS property written in a separate file with .css extension and should be linked to the HTML document using **link** tag. This means that for each element, style can be set only once and that will be applied across web pages.

Example: The file given below contains CSS property. This file save with .css extension. For Ex: **style.css**

```
body {
  background-color:powderblue;
}
#ptag {
  font-style:bold;
  font-size:20px;
}
```

Below is the HTML file that is making use of the created external style sheet

- **link** tag is used to link the external style sheet with the html webpage.
- **href** attribute is used to specify the location of the external style sheet file.

```
<html>
  <head>
    <link rel="stylesheet" href="geeks.css"/>
  </head>

  <body>
```



```
<div id="ptag">
  A computer science Engineering
</div>
</body>
</html>
```

All CSS Simple Selectors

Selector	Example	Example description
<u>#id</u>	#firstname	Selects the element with id="firstname"
<u>.class</u>	.intro	Selects all elements with class="intro"
<u>element.class</u>	p.intro	Selects only <p> elements with class="intro"
<u>*</u>	*	Selects all elements
<u>element</u>	p	Selects all <p> elements
<u>element,element,..</u>	div, p	Selects all <div> elements and all <p> elements

Exercises-1(Program)

Style.css

```
.centerdiv
{
  position: fixed;
  left:50%;
  top:50%;
  padding: 10px;
  transform: translate(-50%,-50%);
  border: 2px solid black;
  background-color: whitesmoke;
  border-radius: 25px;
  box-shadow: 10px 10px 10px cornflowerblue;
  /*The CSS transform property "translate(-50%,-50%)"
  moves an element along the x-axis and y-axis by -50%
  of its own width and height respectively,
  relative to its current position.
  */
}
body{
  margin: 0;
  font-family: Arial, Helvetica, sans-serif;
}
.header{
```




```

overflow: hidden;
background-color: #f1f1f1;
}
.header a {
float: left;
color: black;
text-align: center;
padding: 12px;
text-decoration: none;
font-size: 18px;
line-height: 25px;
border-radius: 4px;
}
.header-right {
float: right;
}
.header a.logo {
font-size: 25px;
font-weight: bold;
}

#left{
float: left;
border-radius: 25px;
background-color: whitesmoke;
margin: 50px;
font-family: Arial, Helvetica, sans-serif;
font-size:medium;
}

Home.html
<!DOCTYPE html>
<html>
<head>
<meta name ="description" content="MeanStack">
<meta name="keywords" content="Online Shopping Site for Mobiles, Electronics, Furniture,
Grocery, Lifestyle, Books & More. Best Offers!">
<meta name="author" content="third cse">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
<meta http-equiv="refresh" content="2">
<link rel="stylesheet" href="styles.css">
<style>

</style>

```



```
</head>
<body>
<header class="header">
  <nav id="logo">
    <a href="homepage.html" class="logo" >
      E-kart
    </a>
  </nav>
  <nav class="header-right">
    <a href="#">Login</a>
    <a href="#">Signup</a>
    <a href="#">Track Order</a>
    <a href="About.html">Aboutus</a>
  </nav>
</header>
  <br/>
<div class="centerdiv">
  <p style="font-size: medium ; font-family: 'Times New Roman', Times, serif;">
    IEKart's is an online shopping website that sells goods in retail. This company deals
    with various categories like Electronics, Clothing, Accessories etc </p>
  <h1>Today Offers</h1>
  <section>
    <h2>Offers on samsung Mobile </h2>
    <p>offers on s21,s23 ,A50 <a href="#">please check </a></p>
  </section>

  <section>
    <h2>Offers on Shoes </h2>
    <p>Offers on Nike , Addidas, Sketchers <a href="#">please check </a>...</p>
  </section>
</div>
</body>
</html>
```

Aboutus.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="styles.css">
<title> About Page </title>
</head>
<body>
  <header class="header">
    <nav id="logo">
```



```

<a href="homepage.html" class="logo" >
  E-kart
</a>
</nav>
<nav class="header-right">
  <a href="#">Login</a>
  <a href="#">Singnup</a>
  <a href="#">Track Order</a>
  <a href="About.html">Aboutus</a>
</nav>
</header>
<div id="left">
<h1> About Lab </h1>
<ul type="Square">
  <li> Develop professional webpages of an application using html elements </li>
  <li>Utilize javascript for developing interactive html web pages and validat form data.</li>
  <li>Build a basic web serve using node.js and also working with node package manager </li>
  <li>Build a web serve using express.js</li>
  <li>Make use of type script to optimize java script code by using the concept of strict type
checking. </li>
</ul>
</div>
<div id="left">
<h1>About Website</h1>
<ul type="circle">
<li>
it is demo type of application for E-kart
</li>
<li>
  like flipkart,amazon,myntra,Ajio,...etc
</li>
<li> For front end .we need to use few technologies like html5,css,javascript </li>
<li> For Back-end we need to use Nodejs, expressjs,Typescript</li>
<li> Through this lab we will get good idea on E-commerce applications </li>
</ul>
</div>
</body>
</html>

```

Output :



IEKart's is an online shopping website that sells goods in retail. This company deals with various categories like Electronics, Clothing, Accessories etc

Today Offers

Offers on samsung Mobile

offers on s21,s23 ,A50 [please check](#)

Offers on Shoes

Offers on Nike , Addidas, Sketchers [please check...](#)

Above page is basic home page . when we click aboutus link it must be redirect to aboutus page like below output :



About Lab

- Develop professional webpages of an application using html elements
- Utilize javascript for developing interactive html web pages and validat form data.
- Build a basic web serve using node.js and also working with node package manager
- Build a web serve using express.js
- Make use of type script to optimize java script code by using the concept of strict type checking.

About Website

- it is demo type of application for E-kart
- like flipkart,amazon,myntra,Ajio,...etc
- For front end ,we need to use few technologies like html5,css,javascript
- For Back-end we need to use Nodejs, expressjs,Typescript
- Through this lab we will get good idea on E-commerce applications



Exercises -2

2a) Creating Table Elements, Table Elements : Colspan/Rowspan Attributes, border, cellspacing, cellpadding attributes

What is table tag:

<table> :

A table is a structured set of data made up of rows and columns (**tabular data**). A table allows you to quickly and easily look up values that indicate some kind of connection between different types of data, for example a person and their age, or a day of the week, or the timetable for a class .

Person	Age
Chris	38
Dennis	45
Sarah	29
Karen	47

Table tags :

Tag	Description
<table>	It defines a table.
<tr>	It defines a row in a table.
<th>	It defines a header cell in a table.
<td>	It defines a cell in a table.

Important attributes in table:

Colspan and Rowspan: The rowspan and colspan are the attributes of <td> tag. These are used to specify the number of rows or columns a cell should merge. The rowspan attribute is for merging rows and the colspan attribute is for merging columns of the table in HTML.

Border: The HTML <table> border Attribute is used to *specify the border of a table*. It sets the border around the table cells.

Syntax: <table border="1|0">

Cell spacing: Cell spacing is the space between each cell

Cell padding: Cell padding is the space between the cell edges and the cell content.

With Padding

hello	hello	hello
hello	hello	hello
hello	hello	hello

With Spacing

hello	hello	hello
hello	hello	hello
hello	hello	hello

```
<table border="1"
  cellpadding="4"
  cellspacing="5">
  <thead>
  <td><span>Name</span></td>
  <td><span>Age</span></td>
  </thead>
  <tbody>
```



```
<tr>
  <td>Rani</td>
  <td>30</td>
</tr>
<tr>
  <td>Rajan</td>
  <td>35</td>
</tr>
<tr>
  <td>Akshaya</td>
  <td>17</td>
</tr>
<tr>
  <td>Ashick</td>
  <td>13</td>
</tr>
</tbody>
</table>
```

Output:

Name	Age
Rani	30
Rajan	35
Akshaya	17
Ashick	13

Timetable using table tags

Day	I 10:30 to 11:30	II 11:30 to 12:30	III 12:30 to 1:30	IV 1:30 to 2:30	V 2:30 to 3:30
Monday	CD	MPMC	L U N C H	ML	MC
Tuesday	CD	MPMC		ML	MC

```
<table border="5" align="center">
  <tr>
    <td align="center" height="50">
      <b> Day </b>
    </td>
```



<td align="center" height="50">

I </br> 10:30 to 11:30

</td>

<td align="center" height="50">

 II </br> 11:30 to 12:30

</td>

<td align="center" height="50">

 III </br> 12:30 to 1:30

</td>

<td align="center" height="50">

 IV </br> 1:30 to 2:30

</td>

<td align="center" height="50">

 V </br> 2:30 to 3:30

</td>

</tr>

<tr>

<td align="center" height="50">

 Monday

</td>

<td align="center" height="50">

CD

</td>

<td align="center" height="50">

MPMC

</td>

<td rowspan="6" align="center" height="50">

<h2>L
U
N
C
H</h2>

</td>

<td align="center" height="50">

ML

</td>

<td align="center" height="50">

MC

</td>

</tr>

<tr>

<td align="center" height="50">

 Tuesday

</td>

<td align="center" height="50">

CD

</td>

<td align="center" height="50">

MPMC



```

</td>
<td align="center" height="50">
    ML
</td>
<td align="center" height="50">
    MC </td>
</tr>
</table>

```

The HTML <form> element can contain one or more of the following form elements:

1. <input>
2. <label>
3. <select>
4. <textarea>
5. <button>
6. <fieldset>
7. <legend>
8. <datalist>

The <datalist> tag specifies a list of pre-defined options for an <input> element.

The <datalist> tag is used to provide an "autocomplete" feature for <input> elements. Users will see a drop-down list of pre-defined options as they input data.

9. <output>
10. <option>
11. <optgroup>

The <optgroup> tag is used to group related options in a <select> element (drop-down list).

Form Attributes:

The following form attributes play the important role:

Attribute	Value	Description
<u>action</u>	URL	Specifies where to send the form-data when a form is submitted
<u>method</u>	get post	Specifies the HTTP method to use when sending form-data
<u>enctype</u>	application/x-www-form-urlencoded multipart/form-data text/plain	Specifies how the form-data should be encoded when submitting it to the server (only for method="post")

Html input Elements:

Here are the different input types you can use in HTML

- <input type="button">
- <input type="checkbox">
- <input type="date">
- <input type="email">



- <input type="file">
- <input type="hidden">
- <input type="number">
- <input type="password">
- <input type="radio">
- <input type="range">
- <input type="reset">
- <input type="search">
- <input type="submit">
- <input type="text">

```
<!DOCTYPE html>
```

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

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Document</title>
```

```
</head>
```

```
<body>
```

```
  <form action="/test/menu.html" method="get">
```

```
    <table align="center">
```

```
      <tr>
```

```
        <td> Username </td>
```

```
        <td> <input type="text" id="username" required> </td>
```

```
      </tr>
```

```
      <tr>
```

```
        <td>Password </td>
```

```
        <td> <input type="password" id="pwd" required> </td>
```

```
      </tr>
```

```
      <tr>
```

```
        <td> Enter your Branch </td>
```

```
        <td> <fieldset>
```

```
          <legend> Select your Branch </legend>
```

```
          <input type="radio" name="branch">cse </br>
```

```
          <input type="radio" name="branch">i.t </br>
```

```
          <input type="radio" name="branch">ece
```

```
        </fieldset> </td>
```

```
      </tr>
```

```
      <tr>
```

```
        <td>Skill set </td>
```

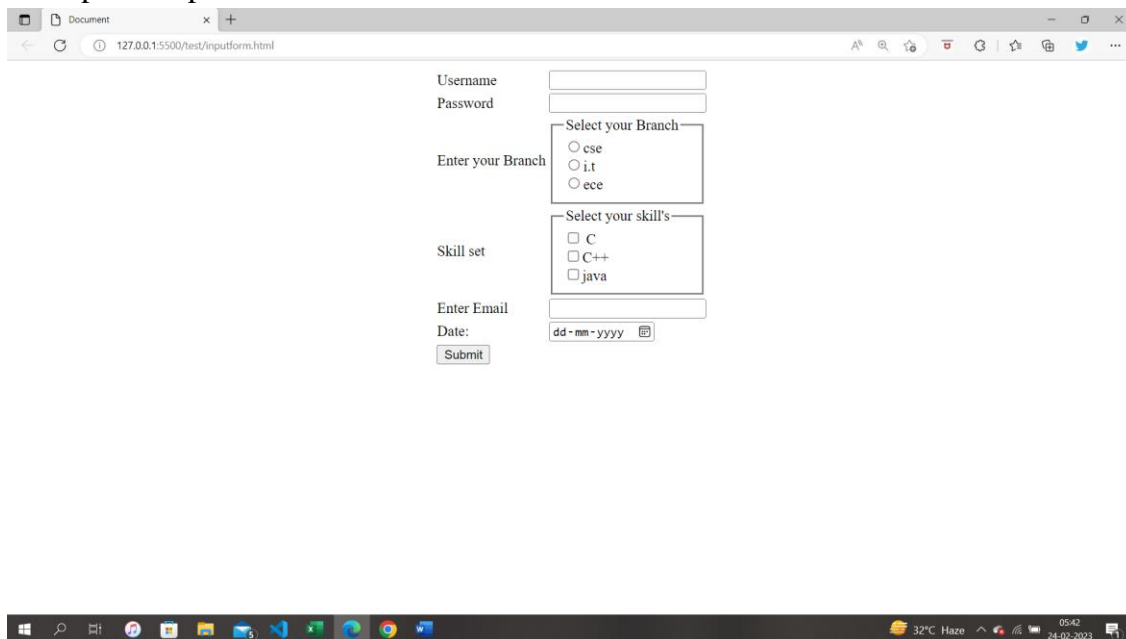
```
        <td> <fieldset>
```

```
          <legend>Select your skill's </legend>
```

```
          <input type="checkbox" name="C"> C </br>
```

```
<input type="checkbox" name="C++">C++ <br/>
<input type="checkbox" name="java">java
</fieldset> </td>
</tr>
<tr>
<td>Enter Email </td>
<td><input type="email" id="idem"> </td>
</tr>
<tr>
<td>Date:</td>
<td><input type="date"/> </td>
</tr>
<tr>
<td colspan="2"><input type="submit" id="idsub"> </td> </tr>
</table>
</form>
</body>
</html>
```

Example of input elements



2b) Module Name: Creating Form Elements, Color and Date Pickers, Select and Datalist Elements

Creating Form Elements, Color and Date Pickers, Select and Datalist Elements

Color pickers : The `<input type="date">` defines a date picker. The resulting value includes the year, month, and day.

Date pickers: Use the color picker by clicking and dragging your cursor inside the picker area to highlight a color on the right

Datalist: The `<datalist>` tag specifies a list of pre-defined options for an `<input>` element.

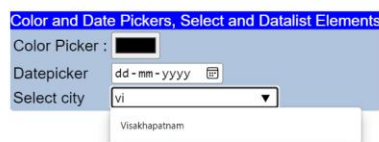


The <datalist> tag is used to provide an "autocomplete" feature for <input> elements. Users will see a drop-down list of pre-defined options as they input data.

The <datalist> element's id attribute must be equal to the <input> element's list attribute (this binds them together).

```
<!DOCTYPE html>
<html>
<head>
<meta name="description" content="MeanStack">
<meta name="keywords" content="Online Shopping Site for Mobiles, Electronics, Furniture, Grocery, Lifestyle, Books & More. Best Offers!">
<meta name="author" content="third cse">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
<link rel="stylesheet" href="/Exercise-1/styles.css">
<style>
  .divcontent{
    /*margin-left: 50px;
    margin-right: 50px;
    margin-top: 50px;*/
    position:fixed;
    left: 50%;
    top:45%;
    max-width: 500px;
    background-color: lightsteelblue;
    transform: translate(-50%,-50%);
    border-radius: 4px;
  }
</style>
</head>
<body>
<div class="divcontent ">
  <div style="background-color: blue;color: white;max-width:500px; text-align: center; "> Color
and Date Pickers, Select and Datalist Elements </div>
    <table>
    <tr>
      <td>
        Color Picker :
      </td>
      <td> <input type="color" > </td>
    </tr>
    <tr>
      <td>
        Datepicker </td>
      <td>
        <input type="date" />
      </td>
    </tr>
  </div>
</body>
</html>
```

```
</td>
</tr>
<tr>
<td >
    <label> Select city </label>
</td>
<td><input list="citys" >
    <datalist id="citys">
        <option value="Hyderabad">
        <option value="Bangalore">
        <option value="Visakhapatnam"></option>
    </datalist></td>
</tr>
</div>
</body>
</body>
</html>
```



2C) Module Name: Input Elements – Attributes

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

Signup Page(Registration Page)

```
<!DOCTYPE html>
<html>
<head>
<meta name ="description" content="MeanStack">
```



```
<meta name="keywords" content="Online Shopping Site for Mobiles, Electronics, Furniture, Grocery, Lifestyle, Books & More. Best Offers!">
<meta name="author" content="third cse">
<meta name="viewport" content="width=device-width,initial-scale=1.0">
<link rel="stylesheet" href="/Exercise-1/styles.css">

<style>

input[type=text], select {
    width: 100%;
    padding: 12px 20px;

    display: inline-block;
    border: 1px solid #ccc;
    border-radius: 2px;
    box-sizing: border-box;
}
input[type=password] {
    width: 100%;
    padding: 12px 20px;

    display: inline-block;
    border: 1px solid #ccc;
    border-radius: 2px;
    box-sizing: border-box;
}
input[type=submit] {
    background-color: #04AA6D;
    color: white;
    border: none;
    height: 100%;
    padding: 12px 20px;
    border-radius: 4px;
    cursor: pointer;
    float: left;
    width:100%;
}
.divcontent{
    /*margin-left: 50px;
    margin-right: 50px;
    margin-top: 50px;*/
    position:fixed;
    left: 50%;
    top:45%;
    max-width: 500px;
```



```
background-color: lightsteelblue;

transform: translate(-50%,-50%);
border-radius: 4px;

}
h3{

color:white; max-width: 500px; background-color: blue; text-align: center;
}

</style>
</head>
<body>
    <header class="header">
        <nav id="logo">
            <a href="homepage.html" class="logo" >
                E-kart
            </a>
        </nav>
        <nav class="header-right">
            <a href="#">Login</a>
            <a href="#">Singup</a>
            <a href="#">Track Order</a>
            <a href="#"> </a>
            <a href="About.html">Aboutus</a>
        </nav>
    </header>
    <body>

    <div class="divcontent ">
        <div style="background-color: blue;color: white;max-width:500px; text-align: center; " >
Registration </div>

        <table>
        <tr>
            <td>
                <input type="text" id="txtinp" placeholder="Username" required />
            </td>

            <td> <input type="password" id="txtpwd" placeholder="password" required> </td>
        </tr>
        <tr>
            <td>
```



```

<input type="text" id="txtpincode" placeholder="Pincode"/></td>
<td>
  <input type="text" id="txtlocation" placeholder="Location"/>
</td>
</tr>
<tr>
<td colspan="2">
  <textarea cols="55" rows="6" id="address" placeholder="Address(Area and
Street)" ></textarea>
</td>
</tr>
<tr>
<td>
  <input type="text" id="txtcity" placeholder="City/Town"/>
</td><td>
  <select id="states">
    <option>---Select---</option>
    <option> Andhra Pradesh </option>
    <option> Telangana </option>
    <option>Bihar</option>
    <option>Jharkhand</option>
  </select>
</td>
</tr>
<tr>
<td>
  <input type="text" id="txtpincode" placeholder="Landmark"/></td>
<td>
  <input type="text" id="txtlocation" placeholder="Phone"/>
</td>
</tr>
<tr>
<td colspan="2">
  Address: </td>
</tr>
<tr>
<td><input type="radio" name="address">Home address </td>
<td>
  <input type="radio" name="address">Work
</td>
</tr>
<tr>
<td colspan="2">

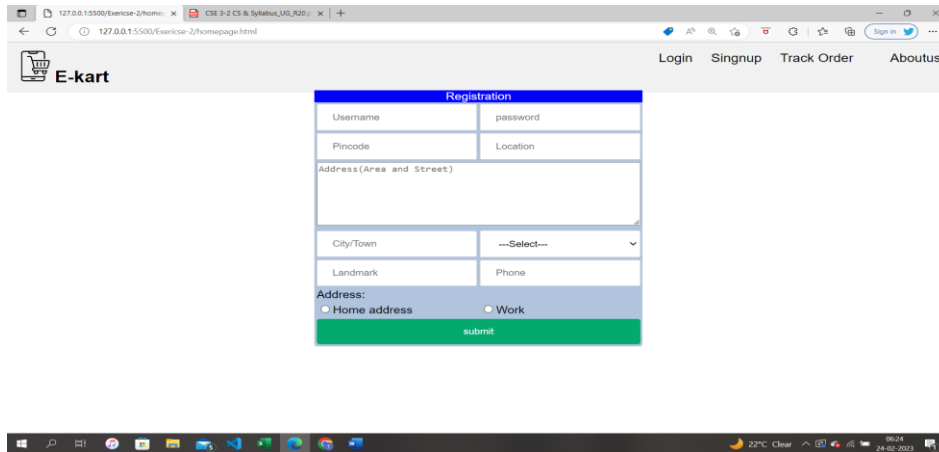
```

```

<input type="submit" value="submit"/>
</td>
</tr>
</table>
</div>
</body>
</body>
</html>

```

Output :



2d). Add media content in a frame using audio, video, iframe elements to the Home page of IEKart's Shopping application.

For this section we add YouTube video on our homepage using iframe tag

The <iframe> tag specifies an inline frame.

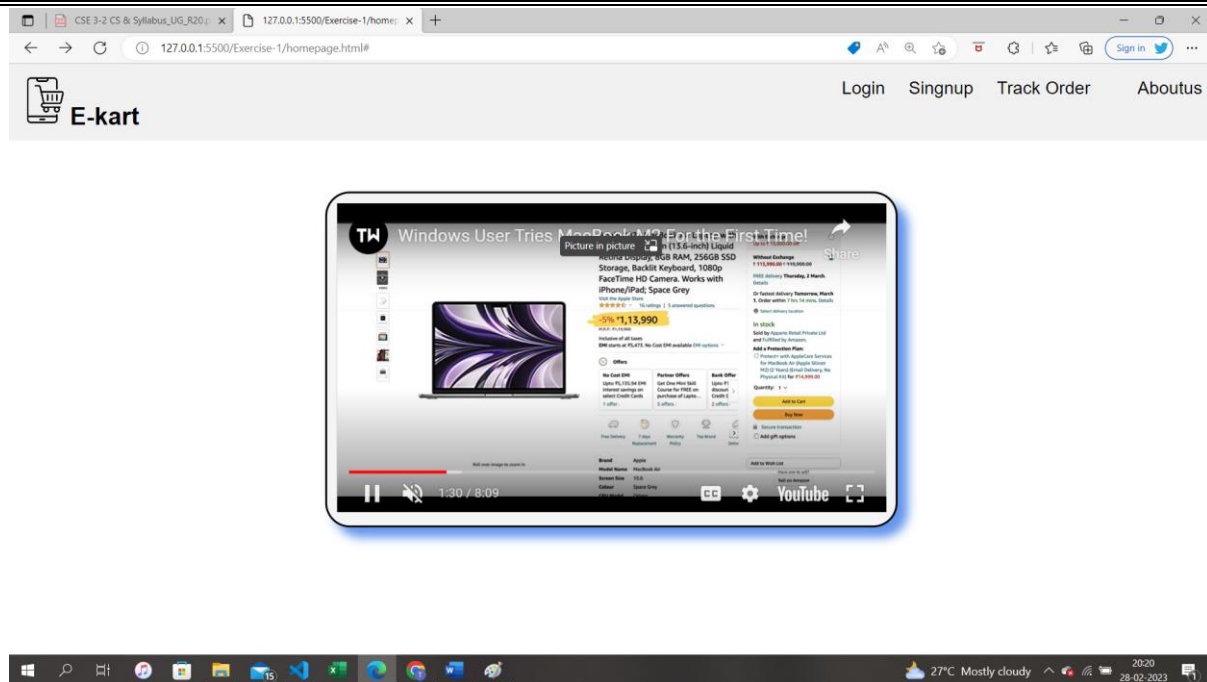
An inline frame is used to embed another document within the current HTML document.

Add below script in Exercises -1 homepage.html in the place centre div

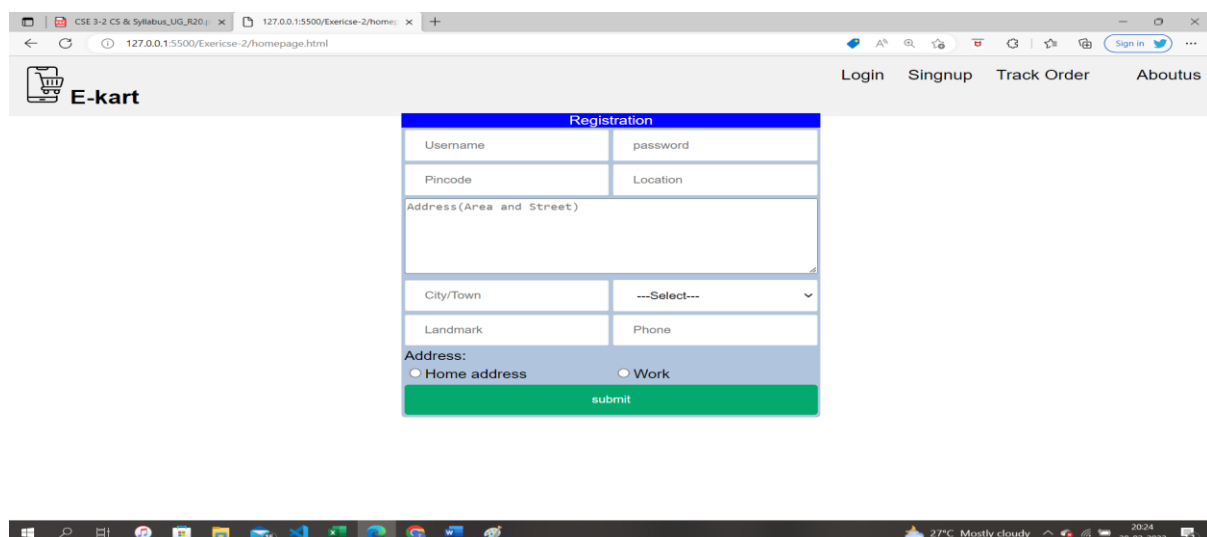
```

<div class="centerdiv">
  <iframe width="560" height="315" src="https://www.youtube.com/embed/a425hb1Zsjc"
  title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write;
  encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen>
  </iframe>
</div>

```

When we add above script in our homepage.html in Exercises -1 we got output like above.
Now click signup link then navigate to registration page





Exercises-3:

3A) 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)

Difference between let and var and const :

In JavaScript, users can declare a variable using 3 keywords that are var, let, and const

var	let	const
The scope of a <i>var</i> variable is functional scope.	The scope of a <i>let</i> variable is block scope.	The scope of a <i>const</i> variable is block scope.
It can be updated and re-declared into the scope.	It can be updated but cannot be re-declared into the scope.	It cannot be updated or re-declared into the scope.
It can be declared without initialization.	It can be declared without initialization.	It cannot be declared without initialization.
It can be accessed without initialization as its default value is "undefined".	It cannot be accessed without initialization otherwise it will give 'referenceError'.	It cannot be accessed without initialization, as it cannot be declared without initialization.

Program:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <script>
    const pi=3.14;
    var r=9;
    function FindArea(r)
    {
      return pi*r*r;
    }
    let Area= FindArea(r);
    document.writeln(Area);
  </script>
</head>
<body>
</br>
</br>
</body>
</html>
```



Data Types: Every Variable has a data type that tells what kind of data is being stored in a variable. There are two types of data types in JavaScript namely Primitive data types and Non-primitive data types.

Primitive data types: The predefined data types provided by JavaScript language are known as primitive data types.

Non-primitive data types: The data types that are derived from primitive data types of the JavaScript language are known as non-primitive data types. It is also known as derived data types or reference data types.

There are five types of primitive data types in Javascript.

1. **Number:** Number data type in javascript can be used to hold decimal values as well as values without decimals.
2. **String:** The string data type in javascript represents a sequence of characters that are surrounded by single or double quotes.
3. **Undefined:** The meaning of undefined is 'value is not assigned'.
4. **Boolean:** The boolean data type can accept only two values i.e. true and false.
5. **Null:** This data type can hold only one possible value that is null.

Non-Primitive data types in Javascript :

1. Object: Object in Javascript is an entity having properties and methods. Everything is an object in javascript.

How to create an object in javascript:

```
let student = {firstName:"ravi", lastName:"kumar", age:25, city:"vizag"};
```

or

```
const student = new Object();
```

```
person.firstName = "ravi";
```

```
person.lastName = "Kumar";
```

```
person.age = 25;
```

```
person.city = "vizag";
```

2.Array: With the help of an array, we can store more than one element under a single name.

Creating array :

```
A=[item1,item2,.....];
```

```
CONST BIKES = ["TRIUMPH", "BMW", "KAWASAKI NINJA "];
```

Template Literals :

Template Literals use back-ticks (`) rather than the quotes (") to define a string:

Interpolation:

Template literals provide an easy way to interpolate variables and expressions into strings.

The method is called string interpolation.



The syntax is:

`${...}`

Variable Substitutions:

Template literals allow variables in strings:

```
let Subject = "MeanStack";
let Code = "R2011087";
let text = `SubjectName : ${Subject} , SubjectCode : ${Code} `;
document.writeln(text);
```

Expression Substitution:

```
let Mobileprice=7000;
let Headset = 100;
let totprice = `TotalPrice ${ (Mobileprice + Headset) } `;
document.writeln(totprice);
```

Another example of Template literals :

```
function add() {
    let header = "ACET";
    let tags = ["CSE","ECE","IT"];
    let html = `<h2> ${header} </h2>`;
    for(const x of tags) {
        html += `<li> ${x} </li>`;
    }
    html += `</ul>`;
    document.getElementById('demo').innerHTML=html;
}
```

Exercises-3: B) 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.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Movie Rating </title>
</style>
    .divcontent{
        /*margin-left: 50px;
        margin-right: 50px;
        margin-top: 50px;*/
        position:fixed;
        left: 50%;
        top:20%;
        max-width: 500px;
        background-color: white;
        transform: translate(-50%,-50%);
        border-radius: 4px;
```



```
}  
.column {  
  float: left;  
  width: 50%;  
}  
  
/* Clear floats after the columns */  
.row:after {  
  content: "";  
  display: table;  
  clear: both;  
}  
input[type=text], select {  
  width: 100%;  
  padding: 12px 20px;  
  display: inline-block;  
  border: 1px solid #ccc;  
  border-radius: 2px;  
  box-sizing: border-box;  
}  
input[type=submit] {  
  background-color:darkorange;  
  color: white;  
  border: none;  
  height: 100%;  
  padding: 12px 20px;  
  border-radius: 4px;  
  cursor: pointer;  
  float: left;  
  width:100%;  
}  
</style>  
</head>  
<body>  
  <div class="divcontent">  
    <div class="row">  
      <h2> Enter Movie details.... </h2>  
    </div>  
    <div class="row">  
      <div class="column" >  
        <input type ="text" id="mname" placeholder="Moviename"/>  
      </div>  
      <div class="column">  
        <input type="text" id="mstar" placeholder="staring"/>  
      </div>
```



```

</div>
<div class="row">
  <div class="column">
<input type="text" id="lang" placeholder="Language"/>
  </div>
  <div class="column">
<input type="text" id="rating" placeholder="ratings"/>
  </div>
</div>
<div class="row">
<input type="submit" onclick="movies()" value="Submit"/>
</div>
</br>
<div class="row" id="tbl" >
</div>
</div>
<script>
  function movies() {
    let M_name;
    let M_star;
    let lang;
    let rating;
    M_name= document.getElementById('mname').value;
    M_star=document.getElementById('mstar').value;
    lang = document.getElementById('lang').value;
    rating = document.getElementById('rating').value;
    starring = M_star.split(',');
    let htm = `<table border=1>
      <tr> <td> Move Name: </td> <td width:100%>${M_name} </td> </tr>
      <tr> <td> Starring: </td> <td width:100%>${starring} </td> </tr>
      <tr> <td> Language: </td> <td width:100%>${lang} </td> </tr>
      <tr> <td> Ratings: </td> <td width:100%>${rating} </td> </tr>
    </table>`
    document.getElementById('tbl').innerHTML=htm;
  }
</script>
</body>
</html>

```

Output:

Enter Movie details....

RRR	NTR, RAMCHARAN
TELUGU	5
Submit	

After enter the details of movie then click submit button out put display like below

Enter Movie details....

RRR	NTR, RAMCHARAN
TELUGU	5
Submit	

Move Name:	RRR
Starring:	NTR RAMCHARAN
Language:	TELUGU
Ratings:	5

3 C) 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.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Book Movie Tickets...</title>
  <style>
    .divcontent
    {
/*margin-left: 50px;
margin-right: 50px;
margin-top: 50px;*/
position:fixed;
left: 50%;
top:20%;
max-width: 500px;
background-color: white;
transform: translate(-50%,-50%);
border-radius: 4px;
}
.column {
float: left;
width: 50%;
}
```



```
/* Clear floats after the columns */
.row::after {
  display: table;
  clear: both;
}
input[type=text], select {
  width: 100%;
  padding: 12px 20px;
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 2px;
  box-sizing: border-box;
}
input[type=submit] {
  background-color:darkorange;
  color: white;
  border: none;
  height: 100%;
  padding: 12px 20px;
  border-radius: 4px;
  cursor: pointer;
  float: left;
  width:100%;
}
</style>
</head>
<body>
  <div class="divcontent">
    <div class="row">
      Book tickets ...
    </div>
    <div class="row">
      <div class="column">
        <select id="mvs" name="movies">
          <option> Select Movie </option>
          <option> RRR </option>
          <option> Avatar-2</option>

        </select>
      </div>
      <div class="column">
        <input type="text" id="not" placeholder="Number Of Tickets"/>
      </div>
    </div>
  </div>
```




```
<div class="row" >
  <input type="submit" value="ADD" onclick="show_price()" />
</div>

<div id="price"></div>
<div id="totaltickets"></div>
</div>
<script>
function show_price()
{
  const ticketcost = 150;
  let discountpercent=10;
  let not = document.getElementById('not').value;
  let mn = document.getElementById('mvs').value;
  let discountprice = ticketcost * discountpercent/100;
  let totalprice = ticketcost - discountprice;
  let finalprice = `For Movie ${mn}: Cost of ${not} tickets ${((totalprice*not))}`;
  document.getElementById('price').innerHTML= finalprice;
}
</script>
</body>
</html>
```

Output:

Book tickets ...

RRR	▼	2
ADD		

For Movie RRR: Cost of 2 tickets 270

Functions in JavaScript:

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

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

There are mainly two advantages of JavaScript functions.

1. **Code reusability:** We can call a function several times so it save coding.
2. **Less coding:** It makes our program compact. We don't need to write many lines of code each time to perform a common task.

Java script functions example:



```
<script>
function msg(){
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()" value="call function"/>
```

Java script function with arguments:

```
<script>
function getcube(number){
alert(number*number*number);
}
</script>
<form>
<input type="button" value="click" onclick="getcube(4)"/>
</form>
```

Function with return values:

```
<script>
function getInfo(){
return "hello javascript! How r u?";
}
a =getinfo();
document.getelementbyid(a);
</script>
```

Arrow Function:

Arrow function were introduced in ES6
Arrow function allow us to write shorter function syntax:
let myFunction = (a, b) => a * b;

Example of arrow function length of the elements of each array:

```
const materials = [ 'ACET','CSE','IT','MECH'];
console.log(materials.map(material => material.length));
```

O/p: 4,3,2,4



Exercise - 4

4 A) 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.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Book Movie Tickets...</title>
  <style>
    .divcontent
    {
/*margin-left: 50px;
margin-right: 50px;
margin-top: 50px;*/
position:fixed;
left: 50%;
top:20%;
max-width: 500px;
background-color: white;
transform: translate(-50%,-50%);
border-radius: 4px;
}
.column {
float: left;
width: 50%;
}

/* Clear floats after the columns */
.row::after {
display: table;
clear: both;
}
input[type=text], select {
width: 100%;
padding: 12px 20px;
display: inline-block;
border: 1px solid #ccc;
border-radius: 2px;
box-sizing: border-box;
}
input[type=submit] {
background-color:darkorange;
```



```
color: white;
border: none;
height: 100%;
padding: 12px 20px;
border-radius: 4px;
cursor: pointer;
float: left;
width:100%;
}
</style>
</head>
<body>
  <div class="divcontent">
    <div class="row">
      Book tickets ...
    </div>
    <div class="row">
      <div class="column">
        <select id="mvs" name="movies">
          <option> Select Movie </option>
          <option> RRR </option>
          <option> Avatar-2</option>

        </select>
      </div>
      <div class="column">
        <input type="text" id="not" placeholder="Number Of Tickets"/>
      </div>
    </div>
    <div class="row" >
      <input type="submit" value="ADD" onclick="calculate_price()" />
    </div>

    <div id="price"></div>
    <div id="totaltickets"></div>
  </div>
  <script>
    function calculate_price( )
    {
      const ticketcost = 150;
      let discountpercent=10;
      let not = document.getElementById('not').value;
      let mn = document.getElementById('mvs').value;
      if(not>2 && not<6)
      {
```



```
let sub= (discountpercent)=>{ return ticketcost * discountpercent/100 };
let discountprice=sub(discountpercent)- ticketcost;
let finalprice = `For Movie ${mn}: Cost of ${not} tickets $${(discountprice*not)}`;
document.getElementById('price').innerHTML= finalprice;
}
if(not<=2)
{
    let finalprice = `For Movie ${mn}: Cost of ${not} tickets $${(ticketcost*not)}`;
    document.getElementById('price').innerHTML= finalprice;
}
if(not>=6)
{
    document.getElementById('price').innerHTML="If seats are 6 or more than 6, booking is not
allowed";
}
}
</script>
</body>
</html>
```

Output:

Check with 3 tickets:

Book tickets ...

RRR	▼	3
-----	---	---

ADD

For Movie RRR: Cost of 3 tickets -405

According to the condition if user book more than 2 less than 6 10% discount on the tickets

So that we entered 3 tickets so that our discount price for each ticket 135 so that cost of 3 tickets - 405

Check with 6 tickets

Book tickets ...

RRR	▼	6
-----	---	---

ADD

If seats are 6 or more than 6, booking is not allowed



Now check with 2 tickets . In this case discount not apply on ticket price. Each ticket price 150

Book tickets ...

RRR	▼	2
ADD		

For Movie RRR: Cost of 2 tickets 300

What is class and constructor in Javascript :

In JavaScript, classes are the special type of functions. We can define the class just like function declarations and function expressions.

The JavaScript class contains various class members within a body including methods or constructor. The class is executed in strict mode. So, the code containing the silent error or mistake throws an error.

Example of Class:

Class Person

```
{ show(){ console.log(" Method from class person ") } // this is method in class }  
//let see how to create object for class in javascript  
let obj = new Person()  
obj.show()
```

Constructor in Javascript:

The constructor method is a special method:

- It has to have the exact name "constructor"
- It is executed automatically when a new object is created
- It is used to initialize object properties

If you do not define a constructor method, JavaScript will add an empty constructor method.

Example of Constructor :

```
class Person {  
  constructor(P_name,P_age) {  
    this.name = P_name;  
    this.age = P_age;  
  }  
  show() {  
    console.log(this.name,this.age);  
  }  
}  
let obj =new Person("abc",26);  
obj.show();
```



4b) 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.

```
class Person{
  constructor(P_name,P_age) {
    this.P_name = P_name;
    this.P_age = P_age;
  }
}
class employee extends Person{
  constructor(P_name,P_age,E_salary,E_id){
    super(P_name,P_age);
    this.salary = E_salary;
    this.id= E_id;
  }
  Show() {
    console.log('PersonName:'
+ this.P_name,'PersonAge:'+ this.P_age,'EmployeeSalary:'+ this.salary, 'Employee_Id:' + this.id);
  }
}
let obj= new employee("Raju",25,20000,4705);
obj.Show();
output:
PersonName:Raju PersonAge:25 EmployeeSalary:20000 Employee_Id:4705
```

What is Event in JavaScript?

HTML events are "**things**" that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can "**react**" on these events.

For Example :

- The user selects, clicks, or hovers the cursor over a certain element.
- The user chooses a key on the keyboard.
- The user resizes or closes the browser window.
- A web page finishes loading.
- A form is submitted.
- A video is played, paused, or ends.
- An error occurs.



Common Html Events:

Example: validate the input length using onkeyup event javascript :

onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>length Validation</title>
</head>
<body>
  <input type="text" id="txtinput" onkeyup="keypress()">
  <div id="msg" > </div>
  <script>
    keypress()=>
    { txtvalue = document.getElementById('txtinput').value;
      if(txtvalue.length <=4 )
      { document.getElementById('msg').style.color ="red";
        document.getElementById('msg').innerHTML="<h4> input must be between 4 and 10
characters </h4>"; }
      else if(txtvalue.length >10)
      { document.getElementById('msg').style.color ="red";
        document.getElementById('msg').innerHTML="<h4> input must be between 4 and 10
characters </h4>";}
      else
      { document.getElementById('msg').style.color="green";
        document.getElementById('msg').innerHTML="<h4> Valid input </h4>";
      }
    }
  </script>
</body>
</html>
```




4C) 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.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Book Movie Tickets...</title>
  <style>
    .divcontent
    {
/*margin-left: 50px;
margin-right: 50px;
margin-top: 50px;*/
position:fixed;
left: 50%;
top:20%;
max-width: 500px;
background-color: white;
transform: translate(-50%,-50%);
border-radius: 4px;
}
.column {
float: left;
width: 50%;
}

/* Clear floats after the columns */
.row::after {
display: table;
clear: both;
}
input[type=text], select {
width: 100%;
padding: 12px 20px;
display: inline-block;
border: 1px solid #ccc;
border-radius: 2px;
box-sizing: border-box;
}
input[type=submit] {
background-color:darkorange;
color: white;
```



```
border: none;
height: 100%;
padding: 12px 20px;
border-radius: 4px;
cursor: pointer;
float: left;
width:100%;
}
</style>
</head>
<body>
  <div class="divcontent">
    <div class="row">
      Book tickets ...
    </div>
    <div class="row">
      <div class="column">
        <select id="mvs" name="movies">
          <option> Select Movie </option>
          <option> RRR </option>
          <option> Avatar-2</option>

        </select>
      </div>
      <div class="column">
        <input type="text" id="not" placeholder="Number Of Tickets"
onkeyup="show_price()"/>
      </div>
    </div>
    <div class="row" >
      <input type="submit" value="ADD" onclick="show_price()" />
    </div>

    <div id="price"></div>
    <div id="totaltickets"></div>
  </div>
  <script>
    function show_price()
    {
      const ticketcost = 150;
      let discountpercent=10;
      let not = document.getElementById('not').value;
      let mn = document.getElementById('mvs').value;
      let discountprice = ticketcost * discountpercent/100;
      let totalprice = ticketcost - discountprice;
```



```
if( not >2 && not <6 ) {  
  let finalprice = `For Movie ${mn}: Cost of ${not} tickets ${((totalprice*not))}`;  
  document.getElementById('price').innerHTML= finalprice;  
}  
else if(not <=2 ) {  
  let finalprice = `For Movie ${mn}: Cost of ${not} tickets ${((ticketcost*not))}`;  
  document.getElementById('price').innerHTML= finalprice;  
}  
else {  
  let finalprice = " 6 or more than 6 tickets not allowed to book. ";  
  document.getElementById('price').innerHTML= finalprice;  
}  
}  
</script>  
</body>  
</html>
```

Spread Operator:

It takes group and try to spread multiple values

For example we have one function **sum()**

```
JS SpreadOperator.js > ...  
1  function sum(a,b)  
2  {  
3  
4      return a+b;  
5  
6  }  
7  mynum = [3,4];  
8  console.log(sum(mynum));  
9  
  
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  
PS F:\Aditya_College_Information\meanstacklab\Practice> node SpreadOperator.js  
3,4undefined  
PS F:\Aditya_College_Information\meanstacklab\Practice>
```

Now move with spread operator:

Spread operator syntax **...mynum**

```
8  console.log(sum(...mynum));  
9  
  
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  
PS F:\Aditya_College_Information\meanstacklab\Practice> node SpreadOperator.js  
3,4undefined  
PS F:\Aditya_College_Information\meanstacklab\Practice> node SpreadOperator.js  
7  
PS F:\Aditya_College_Information\meanstacklab\Practice>
```

Now let see another example on spread operator :



Based on above example you can add only two arguments / values through the function
Below example we can add 'N' number of arguments with same function using spread operator concept

```
function sum(...args)
{
    let sum=0;
    for(const arg of args)
    {
        sum += arg;
    }
    return sum;
}
console.log(sum(1,2,3,4));
```

Output: >node SpreadOperator.js

10

JavaScript array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

1. By array literal
2. By creating instance of Array directly (using new keyword)
3. By using an Array constructor (using new keyword)

//Declaring And Initializing Arrays

//1.array literal

var marks;

marks =[50,90,100];

console.log(marks[0]);

console.log(marks[2]);

//2. using array key word

var arr_names = new Array(4);

console.log(arr_names.length);

for(var i=0; i<arr_names.length; i++)

{

arr_names[i] = i*2

console.log(arr_names[i])

}

//3 . using array constructor

var names = new Array('acet','cse');

console.log(names);



Exercise - 5

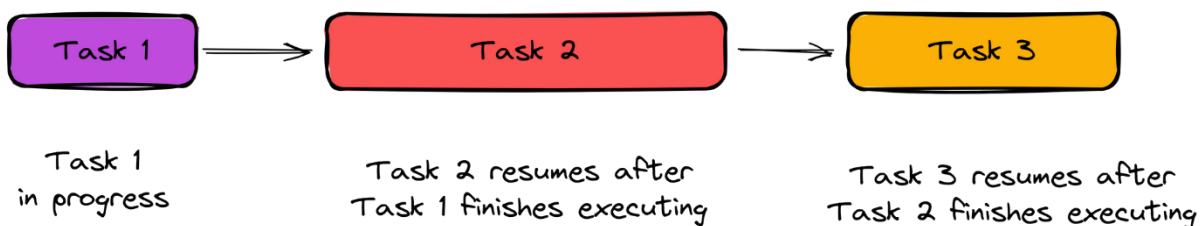
5a) Create an array of objects having movie details. The object should include the movie name, starring, language, and ratings. Render the details of movies on the page using the array

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h3> Movie details </h3>
  <div id="md"></div>
</body>
<script>
var Movie_details = [{Name:"RRR",Starring:"Ramcharn,ntr"},{
Name:"Bahubali",Starring:"Prabas" }];
let output = document.getElementById('md');
let result = ``;
for(let i=0; i<Movie_details.length; i++)
{
  result +=`MovieName: ${Movie_details[i].Name} , Starring: ${Movie_details[i].Starring}` + " ";
}
output.innerHTML=result;
</script>
</html>
```

What is Asynchronous Programming in javascript :

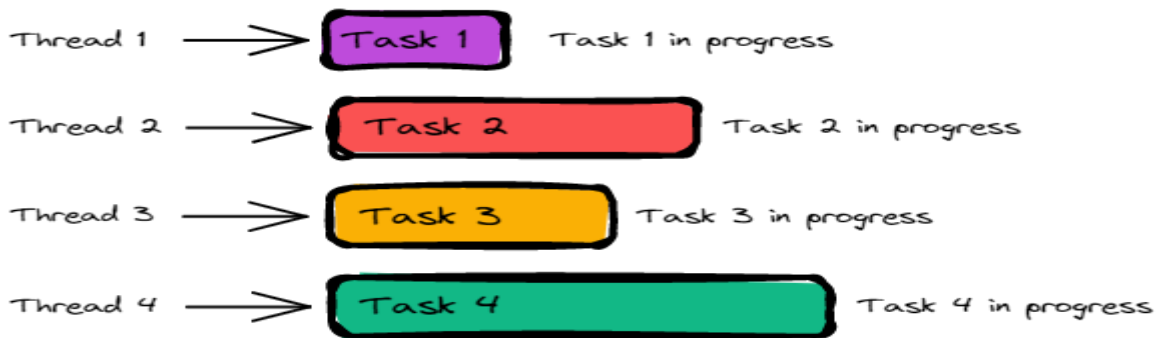
What is synchronous programming?

Synchronous programming is a way for computers to do things one step at a time, in the order they are given the instructions.



What is Asynchronous Programming:

Asynchronous programming is a way for a computer program to handle multiple tasks simultaneously rather than executing them one after the other



Example of Asynchronous Programming:

```
console.log("First call");
```

```
setTimeout(()=>{console.log("Second call")},2000);
```

```
console.log("Third call");
```

output:

```
First call
Third call
Second call
```

What is callback function in javascript:

A callback function is a function passed into another function as an argument

```
function myDisplayer(some) {
    console.log(some);
}

function myCalculator(num1, num2, myCallback) {
    let sum = num1 + num2;
    myCallback(sum);
}

myCalculator(5, 5, myDisplayer);
```

Callback function (with an arrow pointing to `myDisplayer` in the function call)

Out put: 10

Settimeout:

The setTimeout() method calls a function after a number of milliseconds.

1 second = 1000 milliseconds.

For example I have 3 statements sequence like below

Statement1

Statement2

Statement2

But I want to execute Statement2 after 2 seconds . this time we need to use set timeout



```
console.log("First call");

setTimeout(()=>{console.log("Second call")},2000);

console.log("Third call");
```

Output:

```
First call
Third call
Second call
```

SetInterval

The setInterval() function call a functions at specified intervals(in milliseconds)

The setInterval() method continues calling the function until clearInterval() is called or window closed

Below example for setInterval()

```
const test = setInterval(()=>{
  console.log("hello")
},2000)
```

```
setTimeout(() => {
  clearInterval(test)
  console.log("bye..");
}, 20000);
```

Output:

```
hello 6th :Second
hello 7th :Second
hello 8th :Second
hello 9th :Second
Setinterval Completed : setTimeout
```

5b) 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 when

About few functions which we used in above program

Math.random()=>Math.random() returns a random number between 0 (inclusive), and 1 (exclusive):

The Math.floor() method rounds a number DOWN to the nearest integer.

Solution for 5b):

```
function getRandomstockprice() {
```



```
// generate a random number between -1 and 1 (inclusive)
return Math.floor(Math.random() * 3) - 1;
}
function displayPrice(price) {
  console.log("Current price: $" + price.toFixed(2));
}
function PriceChanges() {
  let price = 100; // starting price
  const interval = setInterval(() => {
    // generate a random price change
    const priceChange = getRandomstockprice();
    // update the price
    price += priceChange;
    // display the price on the console
    displayPrice(price);
  }, 3000); // invoke the function every 3 seconds

  // stop the changes after 30 seconds (10 iterations)
  setTimeout(() => {
    clearInterval(interval);
    console.log("Changes stopped");
  }, 30000);
}

// start the changes of price
PriceChanges();
```

Output:

```
arrays> node 5b.js
Current price: $100.00
Current price: $101.00
Current price: $101.00
Current price: $100.00
Current price: $99.00
Current price: $99.00
Current price: $100.00
Current price: $99.00
Current price: $99.00
Changes stopped
```

5c) 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 "Wrong Login"

Login.html

```
<!DOCTYPE html>
```

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

```
<head>
```

```
<meta charset="UTF-8">
```




```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
<script src="login.js"></script>
</head>
<body>
  <b> Username: </b> <input type="text" id="un"/>
  <b> Password: </b> <input type="password" id="pwd"/>
  <input type="submit" onclick="validate()"/>
</body>
</html>
```

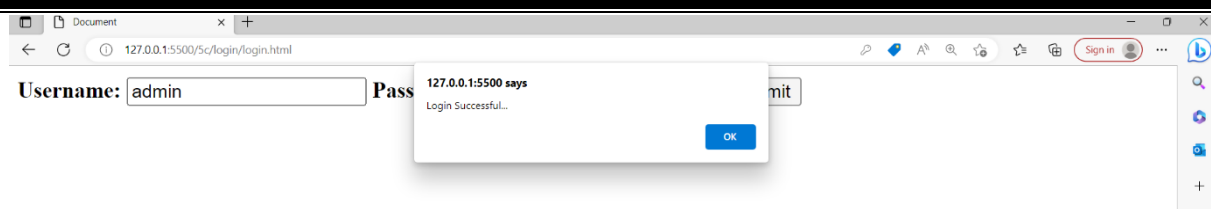
Login.js

```
class user
{
  constructor(un,pwd)
  {
    this.un = un;
    this.pwd = pwd;
  }
  validate()
  {
    if((this.un=="admin") && (this.pwd=="admin"))
    {
      alert("Login Successful...");
    }
    else
    {
      alert("Wrong Login...");
    }
  }
}

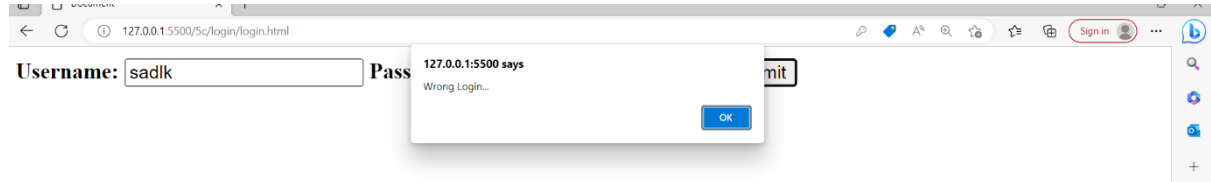
function validate()
{
  un = document.getElementById('un').value;
  pwd = document.getElementById('pwd').value;
  Obj = new user(un,pwd);
  Obj.validate();
}
```

Outputs:

Username and password with admin



Username and password without admin





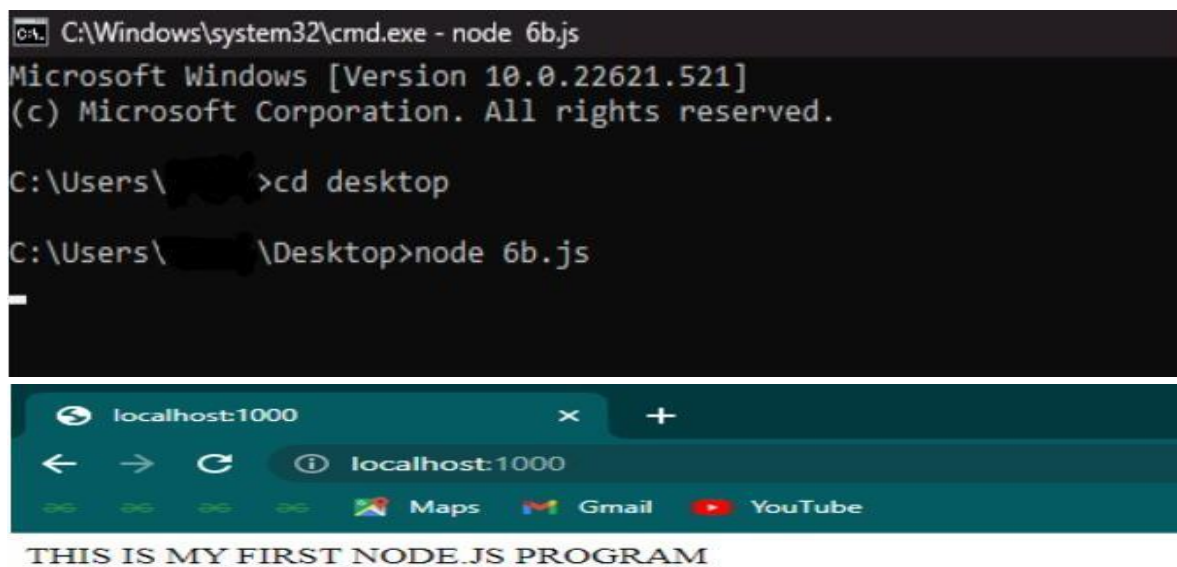
Exercises - 6

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.

Program:

```
var http = require('http'); http.createServer(function(req,res){  
res.writeHead(200, {'Content-Type':'text/html'});  
res.write('THIS IS MY FIRST NODE.JS PROGRAM');  
res.end();  
}).listen(1000);
```

Output:



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.

Program:

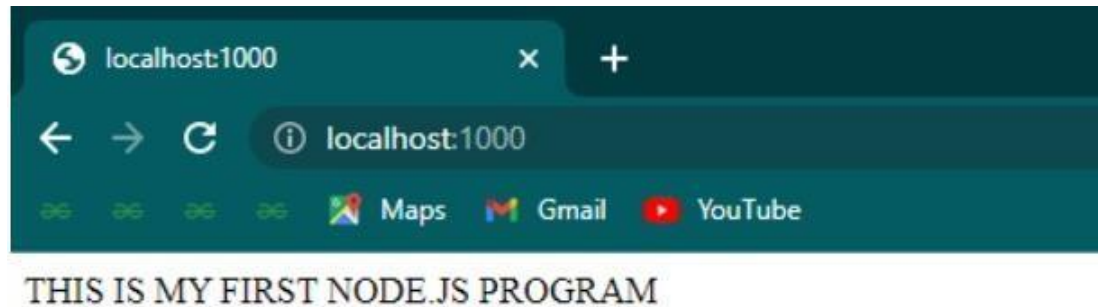
```
var http = require('http'); http.createServer(function(req,res){  
res.writeHead(200, {'Content-Type': 'text/html'});  
res.write('THIS IS MY FIRST NODE.JS PROGRAM');  
res.end();  
}).listen(1000);
```

Output:



```
C:\Windows\system32\cmd.exe - node 6b.js
Microsoft Windows [Version 10.0.22621.521]
(c) Microsoft Corporation. All rights reserved.

C:\Users\>cd desktop
C:\Users\ \Desktop>node 6b.js
```



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.

PROGRAM:

```
//myfile1.js
exports.data = function () {return
"FRIENDS!!!!";};

var http = require('http'); var d =
require('./myfile1');
http.createServer(function (req, res) { res.writeHead(200, {'Content-
Type': 'text/html'}); res.write("WELCOME " + d.data());
res.end();

}).listen(8080);
```

Output:

```
C:\Windows\system32\cmd.exe - node 6c.js
Microsoft Windows [Version 10.0.22621.521]
(c) Microsoft Corporation. All rights reserved.

C:\Users\>cd desktop
C:\Users\ \Desktop>node 6c.js
```

6d Course Name: Node.js Module Name: Restarting Node Application Write a program to show the workflow of restarting a Node application.

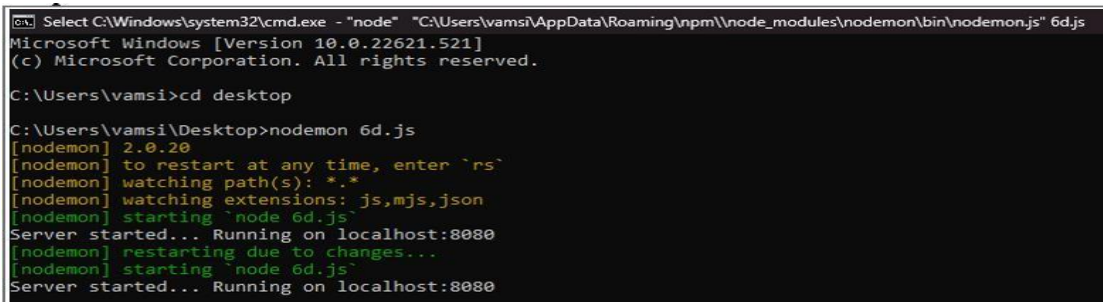
```
const http = require("http");

var server = http.createServer((req, res) => {
  res.write("I have modified the server!");
  res.end();
});

server.listen(8080);

console.log("Server started... Running on localhost:8080");
```

output:



```
Microsoft Windows [Version 10.0.22621.521]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vamsi>cd desktop
C:\Users\vamsi\Desktop>nodemon 6d.js
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node 6d.js`
Server started... Running on localhost:8080
[nodemon] restarting due to changes...
[nodemon] starting `node 6d.js`
Server started... Running on localhost:8080
```

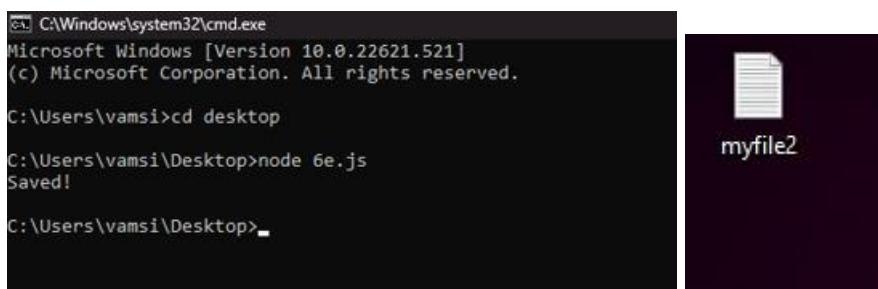
6.e Course Name: Node.js Module Name: File Operations Create a text file src.txt and add the following data to it. Mongo, Express, Angular, Node.

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

fs.appendFile('myfile2.txt', 'Mongo,Express,Angular,Node', function (err) {if
(err) throw err;

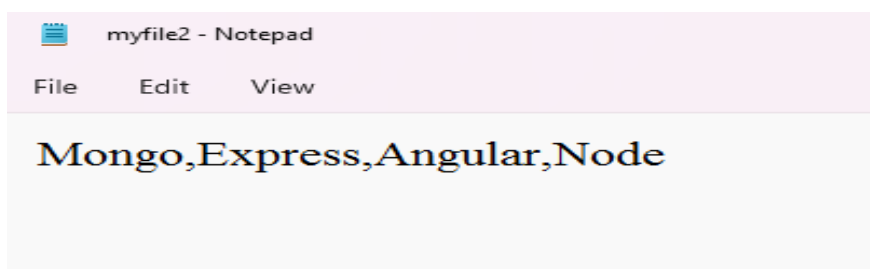
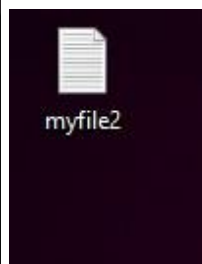
console.log('Saved!');

});
```



```
Microsoft Windows [Version 10.0.22621.521]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vamsi>cd desktop
C:\Users\vamsi\Desktop>node 6e.js
Saved!
C:\Users\vamsi\Desktop>_
```





Exercise - 7

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.

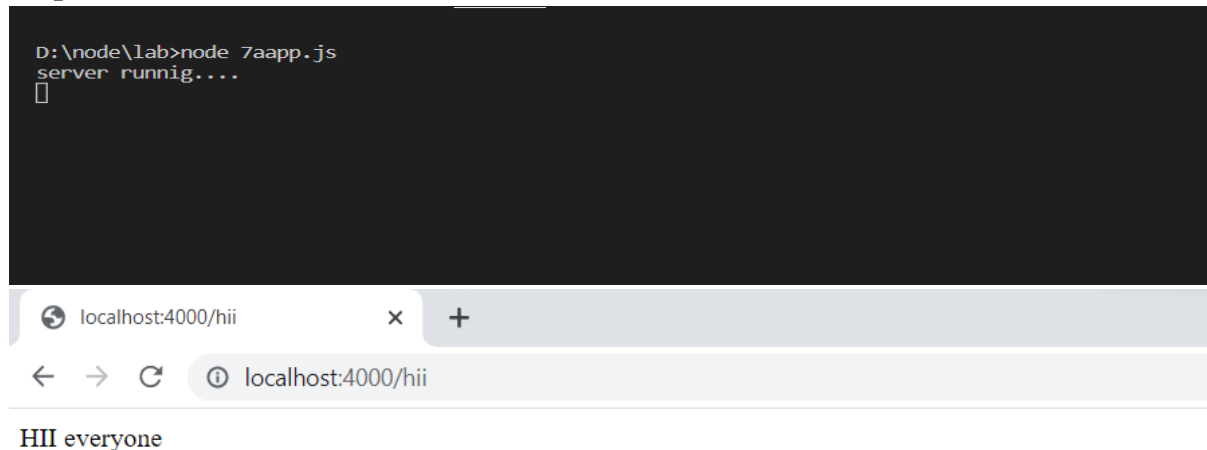
App.js

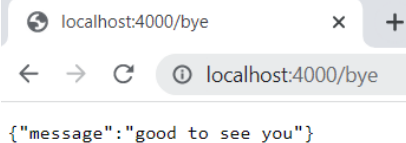
```
var express=require('express')
var route=require('./7aroute.js')
var app=express()
app.use('/',route)
app.listen(4000,function(){
  console.log('server runnig....')
```

Route.js

```
var express=require('express')
var router=express.Router()
router.get('/hii',function(req,res){
  res.status(200).send('HII everyone');
})
router.get('/bye',function(req,res){
  res.status(200).json({ message:"good to see you" });
})
router.all('*',function(req,res){
  res.status(200).json({ status:'fail',message:"Invalid" });
})
module.exports=router;
```

output:





localhost:4000/bye

localhost:4000/bye

```
{"message":"good to see you"}
```

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.

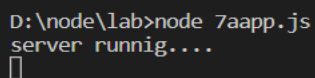
App.js

```
var express=require('express')
var route=require('./route.js')
var app=express()
app.use('/',route)
app.listen(4000,function(){
  console.log('server runnig....')
```

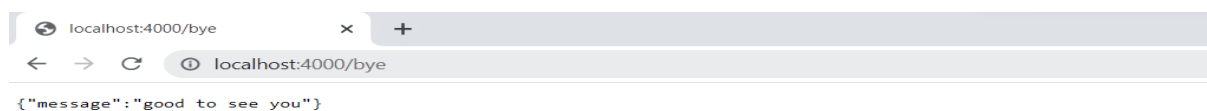
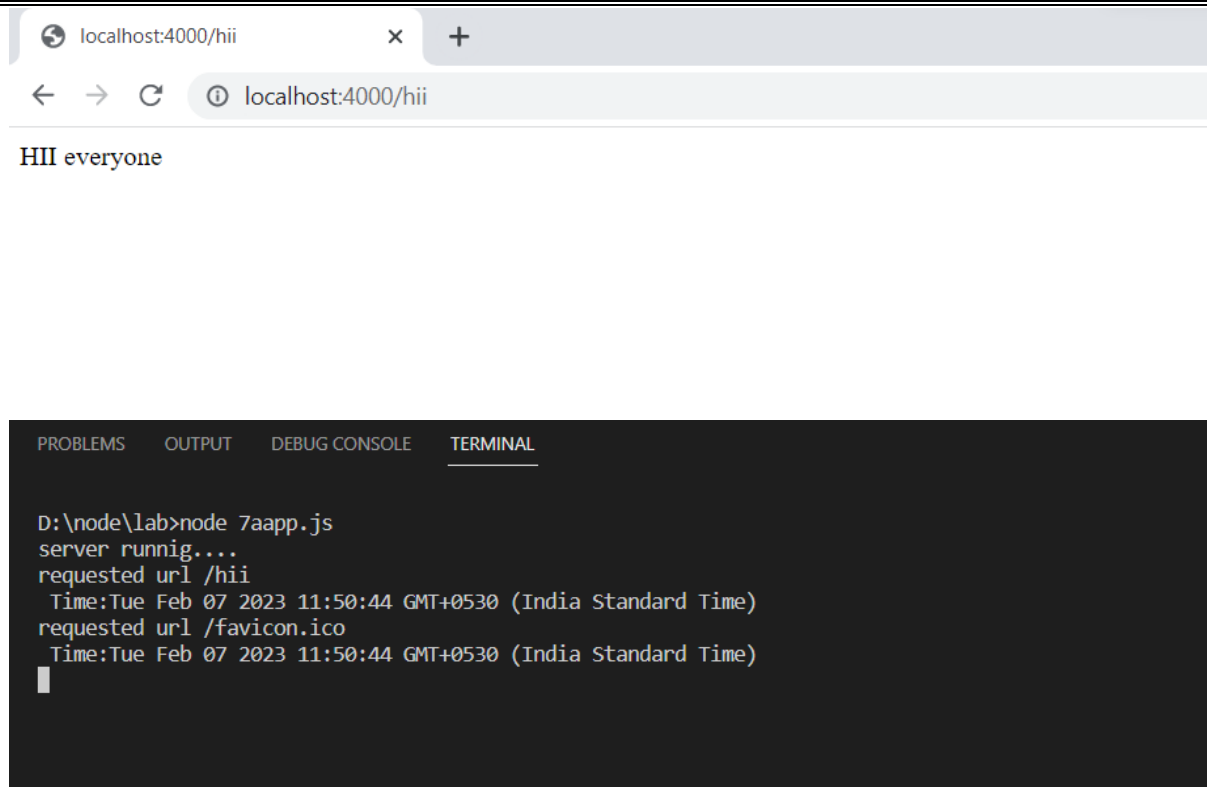
Route.js

```
var express=require('express')
var router=express.Router()
router.get('/hii',function(req,res){
  res.status(200).send('HII everyone');
})
router.get('/bye',function(req,res){
  res.status(200).json({ message:"good to see you" });
})
router.all('*',function(req,res){
  res.status(200).json({ status:'fail',message:"Invalid" });
})
module.exports=router;
```

OUTPUT:



```
D:\node\lab>node 7aapp.js
server runnig....
█
```



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. PROGRAM:

```
var express=require('express')
var mongoose=require('mongoose');
const url='mongodb://0.0.0.0:27017/Hell';
mongoose.set('strictQuery', true);
mongoose.connect(url).then(function(){
  console.log('connected....');
})
```

```
const myNotesSchema = new mongoose.Schema(
  {
    notesID: {
      type: Number,
    },
    name: {
```




```

    type: String,
  },
  data: {
    type: String,
  },
},
{
  timestamps: {
    createdAt: true,
    updatedAt: true,
  },
}
);

```

OUTPUT:

```

D:\node\lab>node 7c.js
connected....

```

7.d Course Name: Express.js Module Name: Models Write a program to wrap the Schema into a Model object.

PROGRAM:

```

//to create model simply means creating collection in database
//syntax:: variable name= mongoose.model('collectionname',schema);

//7D
var express=require('express')
var mongoose=require('mongoose');
const url='mongodb://0.0.0.0:27017/Hell';
mongoose.set('strictQuery', true);
mongoose.connect(url).then(function(){
  console.log('connected....');
})

const myNotesSchema = new mongoose.Schema(
  {
    notesID: {
      type: Number,
    },
    name: {

```



```
    type: String,
  },
  data: {
    type: String,
  },
},
{
  timestamps: {
    createdAt: true,
    updatedAt: true,
  },
}
);
var bo=mongoose.model("12345",myNotesSchema)
console.log('model created.....');
```

OUTPUT:

A screenshot of a terminal window with a dark background. At the top, there are four tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL', with 'TERMINAL' being the active tab. The terminal shows the command 'D:\node\lab>node 7c.js' followed by the output 'model created.....' and 'connected....' on two separate lines. A cursor is visible at the end of the second line.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

D:\node\lab>node 7c.js
model created.....
connected....
█
```



Exercise - 8

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.

PROGRAM:

```
//CRUD operations Create,Read,Update,Delete
var mongoose=require('mongoose');
const url='mongodb://0.0.0.0:27017/Hell';
mongoose.set('strictQuery', true);
mongoose.connect(url,{useNewUrlParser:true},{useUnifiedTopology:true}).then(function(){
  console.log('connected....');})
var schema=mongoose.Schema({name:String,roll:Number});
var bo=mongoose.model("12345",schema);


//Create
var b1=new bo({name:"Rajesh",roll:897});
b1.save(function(err,res){
  if(err)console.log(err)
  console.log('Saved')
})

//update
bo.updateOne({roll:897},{ $set:{name:"rambo"} },function(err,res){
  console.log(res);
})

//Delete
bo.deleteOne({name:"Rajesh"},function(er,res){
  console.log(res)
  console.log('deleted successfully');
})

//Read
bo.find(function(err,res){
  console.log(res);
})
```

Output:



```
D:\node\lab>node 8a.js
connected....
Saved
```



```
D:\node\lab>node 8a.js
connected....
[
  {
    _id: new ObjectId("63e1f1dd8c90300197e0061c"),
    name: 'Rajesh',
    roll: 897,
    __v: 0
  }
]
```

```
D:\node\lab>node 8a.js
connected....
{
  acknowledged: true,
  modifiedCount: 1,
  upsertedId: null,
  upsertedCount: 0,
  matchedCount: 1
}
```

```
D:\node\lab>node 8a.js
connected....
{ acknowledged: true, deletedCount: 1 }
deleted successfully
```

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 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. (iii) 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

Program:

Routing.js

```
const express = require('express');
const routing = express.Router();
const nodes = require('./myNotes');
routing.get('/notes', nodes.getNotes);
routing.get('/update/:id', nodes.updateNotes);
routing.get('/delete/:id', nodes.deleteNotes);
routing.get('/find/:id', nodes.findwithId);
routing.all('*', nodes.invalid);
module.exports = routing;
```

**Mynotes.js**

```
const NotesModel = require('./myNotesSchema');
var a;
exports.getNotes = async (req, res) => {
  try {
    NotesModel.find( function(err, result){
      if ( err )
        console.log(err);
      else
        a=result;
    });
    res.status(200).json({
      status: 'success',
      results: a.length,
      data: {
        a,
      },
    });
  }
  catch (err) {
    res.status(404).json({
      status: 'fail',
      message: 'loading fail',
    });
  }
};
exports.findwithId=async (req,res)=>{
  try {
    NotesModel.find({ notesID:req.params.id }, function(err, result){
      if ( err )
        console.log(err);
      else
        a=result;
    });
    res.status(200).json({
      status: 'success',
      results: a.length,
      data: {
        a,
      },
    });
  }
  catch (err) {
    res.status(404).json({
      status: 'fail',
```



```
    message: 'loading fail',
  });
}
}
exports.updateNotes = async (req, res) => {
  try {
    NotesModel.updateOne({ name: req.params.id }, { $set: { data: "Data is just modified now
successfully" } }, function(er, result) {
      if(er)
        console.log(er)
      else a=result;
    })
    res.status(200).json({
      status: 'success',
      data: {
        a,
      },
    });
  } catch (err) {
    res.status(404).json({
      status: 'updatefailed',
      message: err,
    });
  }
};
exports.deleteNotes = async (req, res) => {
  const delDet = NotesModel.deleteOne({ name: req.params.id
}, function(err, result) { if(err) console.log(err) ;else console.log(result) });
  if (delDet.deletedCount === 0) {
    res.status(404).json({
      status: 'fail',
      message: 'No notes available for this ID',
    });
  } else {
    res.status(200).json({
      status: 'success',
      message: `Notes with ${req.params.id} ID deleted`,
    });
  }
};
exports.invalid = async (req, res) => {
  console.log(params.req.id);
  res.status(404).json({
    status: 'fail',
    message: 'Invalid path',
```



```
});
```

```
};
```

Mynotesschema.js

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://0.0.0.0:27017/Hell', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => console.log('DB connection successful!'));
```

```
//Schema
```

```
const myNotesSchema = new mongoose.Schema(
{
  notesID: {
    type: Number,
    unique: true,
    required: [true, 'Required field'],
  },
  name: {
    type: String,
    required: [true, 'Required field'],
  },
  data: {
    type: String,
  },
},{
  timestamps: {
    createdAt: true,
    updatedAt: true,
  },});
```

```
//Model
```

```
const NotesModel = mongoose.model('mynotes', myNotesSchema);
module.exports = NotesModel;
```

App.js

```
const express = require('express');
const bodyparser = require('body-parser');
const route = require('./routing');
const app = express();
app.use(bodyparser.json());
app.use('/', route);
const port = process.env.PORT || 3000;
app.listen(port, () => {
  console.log(` App running on port .....`);
});
```

Output:



```
localhost:3000/notes
{"status":"success","results":4,"data":[{"_id":"6381d58033c297be48cec0c1","notesID":34,"name":"MANHATAN","data":"Data is just modified now successfully","updatedAt":"2022-11-26T13:26:23.979Z"}, {"_id":"6381d76e92f383699e0ac82b","notesID":12,"name":"ASDG","data":"hello","createdAt":"2022-11-26T09:07:58.328Z","__v":0}, {"_id":"6381d7a260652414e286f515","notesID":892,"name":"MANHATAN","data":"hello","createdAt":"2022-11-26T09:08:50.168Z","updatedAt":"2022-11-26T12:21:15.394Z","__v":0}, {"_id":"6381d7f45d223564d3f41610","notesID":8952,"name":"LKOSOG","data":"hello","createdAt":"2022-11-26T09:10:12.788Z","updatedAt":"2022-11-26T09:10:12.788Z","__v":0}]}
```

```
localhost:3000/find/892
{"status":"success","results":1,"data":{"a":[{"_id":"6381d7a260652414e286f515","notesID":892,"name":"MANHATAN","data":"hello","createdAt":"2022-11-26T09:08:50.168Z","updatedAt":"2022-11-26T12:21:15.394Z","__v":0}]}}
```

```
localhost:3000/delete/MANHATAN
{"status":"success","message":"Notes with MANHATAN ID deleted"}
```

```
localhost:3000/update/ASDG
{"status":"success","data":{"a":{"acknowledged":true,"modifiedCount":1,"upsertedId":null,"upsertedCount":0,"matchedCount":1}}}
```

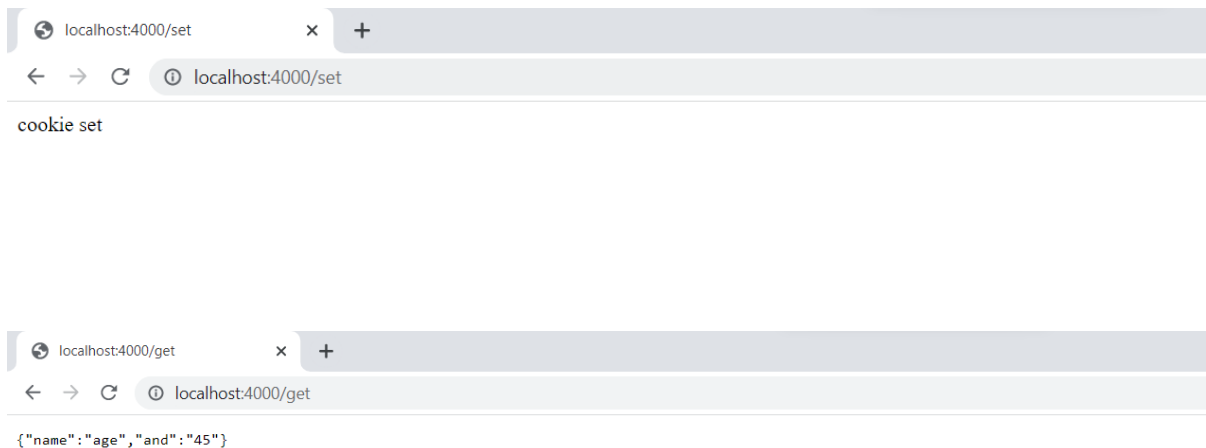
8.c Course Name: Express.js Module Name: Why Session management, Cookies Write a program to explain session management using cookies.

PROGRAM:

```
var express=require('express')
var cookie=require('cookie-parser')
var app=express()
app.use(cookie())
app.get('/set',function(req,res){
```



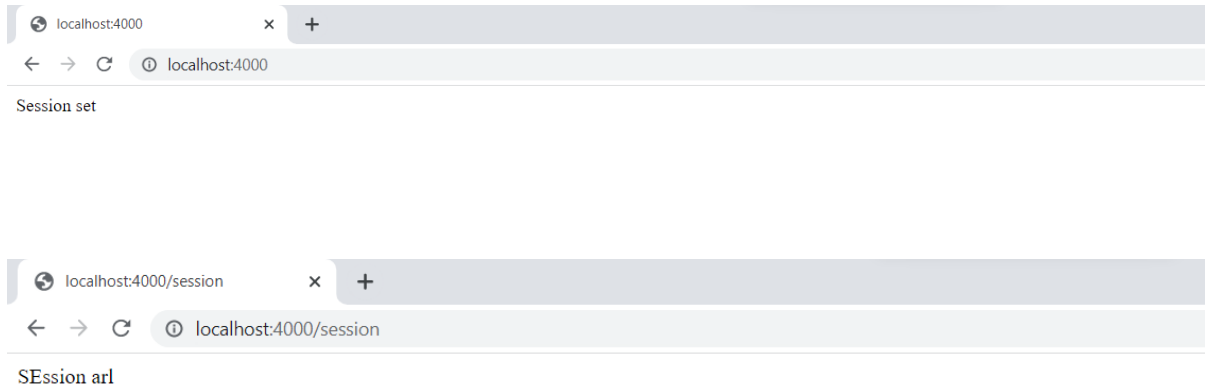

```
res.cookie('name','age');
res.cookie('and','45');
res.status(200).send('cookie set');
})
app.get('/get',function(req,res){
  res.status(200).send(req.cookies);
})
app.listen(4000,function(){
  console.log('runingg.....');
})
```

OUTPUT:**8.d Course Name: Express.js Module Name: Sessions Write a program to explain session management using sessions.****PROGRAM:**

```
var express=require('express')
var session=require('express-session')
var app=express()
app.use(session({ secret:'YOUR_KEY',resave:true,saveUninitialized:true}))
app.get('/',function(req,res){
  req.session.name="SEssion arl"
  return res.send("Session set")
})
app.get('/session',function(req,res){
  return res.send(req.session.name)
})
app.listen(3000,function(){
  console.log("runnig....")
})
```



OUTPUT:



**8.e Course Name: Express.js Module Name: Why and What Security, Helmet Middleware
Implement security features in myNotes application**

PROGRAM:

.HTML

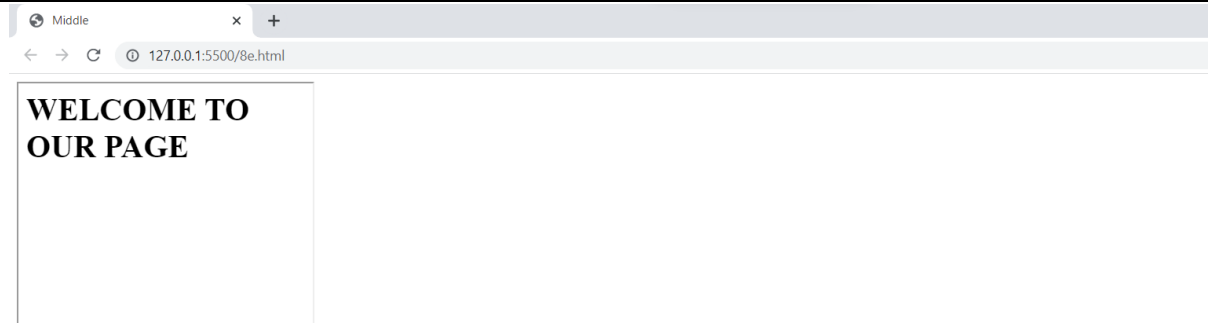
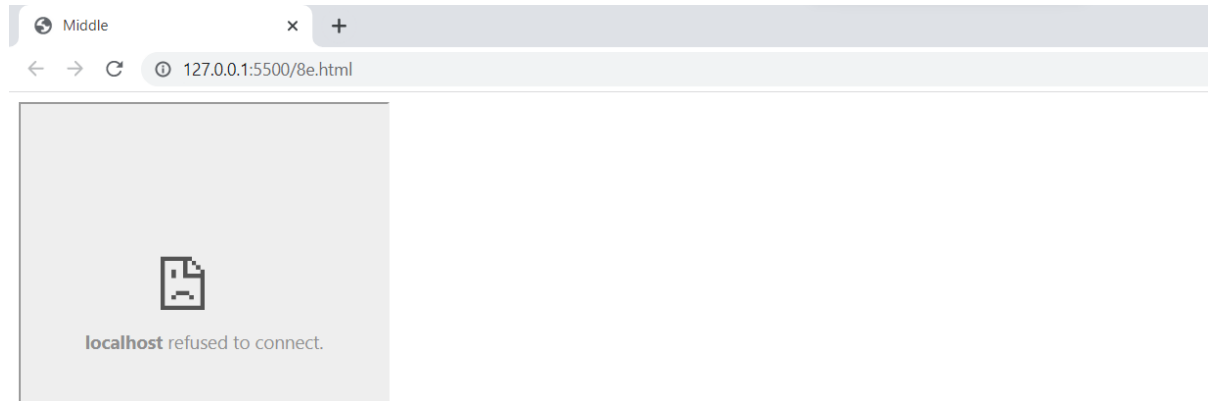
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Middle</title>
</head>
<body>
  <iframe src="http://localhost:4000/ab" height="300" width="300"></iframe>
</body>
</html>
```

.js

```
var ex=require('express')
var hel=require('helmet')
var app=ex()
app.use(hel())
app.get('/ab',(req,res)=>{
  res.send('<h1>WELCOME TO OUR PAGE</h1>');
})
app.listen(4000,()=>{ console.log('rer..')});
```

OUTPUT:

WITHOUT USING HELMET:

**WITH USING HELMET:**



Exercise – 9

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.

Program:

```
function price(a:string){  
  if(a=="GoldPlatinum")  
  {  
    return 10000;  
  }  
  else if(a=="PinkGold")  
  {  
    return 12000;  
  }  
  else if(a=="SilverTitanium")  
  {  
    return 13000;  
  }  
}  
console.log(price("SilverTitanium"));  
console.log(price("PinkGold"));
```

Output:

```
D:\mstd\typescrip>ts-node 9a.ts  
13000  
12000
```

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.

Program:

```
var manufacturers = [{ productId:121,id: 'Samsung', price: 150 },  
{ productId:122,id: 'Microsoft', price: 200 },  
{ productId:123,id: 'Apple', price: 400 },  
{ productId:124,id: 'Micromax', price: 100 }  
];  
var i:number=0;  
var getProductdetails=(product : number):string=>{  
  for(let i=0;i<manufacturers.length;i++)
```



```
{
  if(manufacturers[i].productId==product)
  {
    break;
  }
}
return "productID:"+manufacturers[i].productId+"\n ProductName:"+manufacturers[i].id+"\n Price:
"+manufacturers[i].price;
};
console.log(getproductdetails(1234));
```

Output:

```
productID:121
ProductName:Samsung
Price: 150
```

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.

Program:

```
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;
  }
}
console.log('The available Samsung mobile list: [' +
getMobileByManufacturer('Samsung')+']');
console.log('\nThe available Iphone mobile list: [' + getMobileByManufacturer('Apple')+']');
```



Output:

```
D:\mstd\typescrip>ts-node 9c.ts
The available Samsung mobile list: [Samsung Galaxy S6 Edge,Samsung Galaxy Note 7,Samsung Galaxy J7 SM-J700F]

The available Iphone mobile list: [Apple iPhone 5s,Apple iPhone 6s ,Apple iPhone 7]

D:\mstd\typescrip>
```

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 than or equal to 150 dollars then below mentioned code-snippet would fit into this requirement.

Program:

```
var manufacturers = [{ id: 'Samsung', price: 150 },
{ id: 'Microsoft', price: 200 },
{ id: 'Apple', price: 400 },
{ id: 'Micromax', price: 100 }
];
console.log('Details of Manufacturer array are : ');
function myFunction() {
var test = manufacturers.filter((m) => m.price >= 150);
for (var item of test) {
    console.log(item.id);
}
}
myFunction();
```

OUTPUT:

```
D:\node>ts-node 9c.ts
Details of Manufacturer array are :
Samsung
Microsoft
Apple

D:\node>
```



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 be passed as Samsung and id parameter should be optional while invoking the function, if id is passed as 101 then this function should return Moto mobile list and if manufacturer parameter is either Samsung/Apple then this function should return respective mobile list and similar to make Samsung as default Manufacturer. Below mentioned code-snippet would fit into this requirement.

Program:

```
function getMobileByManufacturer(manufacturer: string = 'Samsung', id?: number): string[] {  
    let mobileList: string[]; if  
(id) {  
        if (id === 101) {  
            mobileList = ['Moto G Play, 4th Gen', 'Moto Z Play with Style Mod'];  
            return mobileList;  
        }  
    }  
    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; } }  
console.log('The available mobile list : ' + getMobileByManufacturer('Apple'));  
console.log('The available mobile list : ' + getMobileByManufacturer(undefined, 101))
```

Output:

```
D:\node>ts-node 9e.ts  
The available mobile list :  Apple iPhone 5s, Apple iPhone 6s, App  
le iPhone 7  
The available mobile list : Moto G Play, 4th Gen,Moto Z Play with  
Style Mod  
D:\node>
```



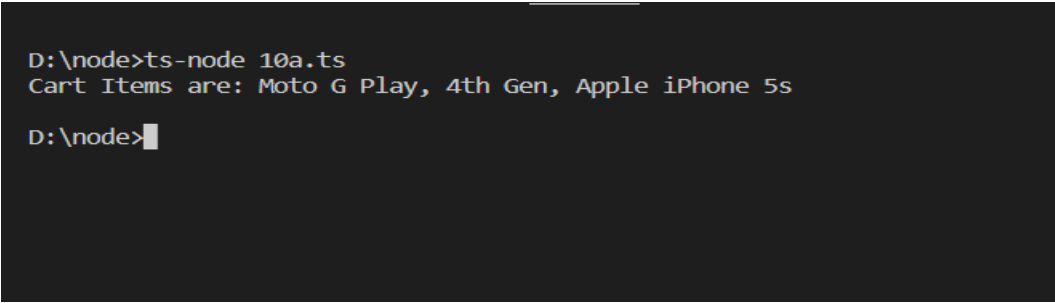
Exercise - 10

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.

Program:

```
const cart: string[] = [];  
const pushtoCart = (item: string) => { cart.push(item); };  
function addtoCart(...productName: string[]): string[] {  
    for (const item of productName) {  
        pushtoCart(item);  
    }  
    return cart;  
}  
console.log('Cart Items are:' + addtoCart(' Moto G Play, 4th Gen', ' Apple iPhone 5s'));
```

Output:



```
D:\node>ts-node 10a.ts  
Cart Items are: Moto G Play, 4th Gen, Apple iPhone 5s  
D:\node>
```

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.

```
interface Product {  
    productId: number ;  
    productName: string ;  
}  
function getProductDetails(productobj: Product): string { return  
    'The product name is : ' + productobj.productName;  
}  
  
const prodObject = {productId: 1001, productName: 'Mobile'};  
const productDetails: string = getProductDetails(prodObject);  
console.log(productDetails);
```




OUTPUT:

```
D:\node>ts-node 10b.ts
The product name is : Mobile

D:\node>
```

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.

Program:

```
interface Product { productId: number; productName: string;
}
function getProductDetails(productobj: Product): string { return "Product ID:
"+productobj.productId+"\nThe product name is : ' + productobj.productName;
}
const prodObject = {productId: 1001, productName: 'Mobile', productCategory: 'Gadget'};
const productDetails: string = getProductDetails(prodObject); console.log(productDetails);
```

OUTPUT:

```
D:\node>ts-node 10c.ts
Product ID: 1001
The product name is : Mobile

D:\node>
```

10.d Course Name: Typescript Module Name: Function Types Declare an interface with function type and access its value.

Program:

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



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

```
let idGenerator: StringGenerator; idGenerator =  
CreateCustomerID;  
const customerId: string = idGenerator('Mr.Tom', 101);  
console.log(customerId);
```

Output:

```
D:\node>ts-node 10d.ts  
The customer id is Mr.Tom 101  
  
D:\node>
```



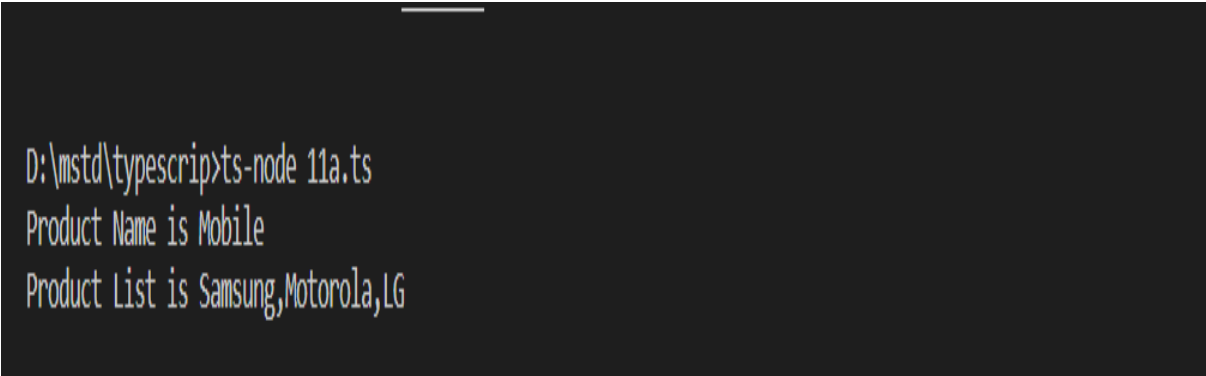
Exercise - 11

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.

PROGRAM:

```
interface Category {  
  categoryName: string;  
}  
interface Product {  
  productName: string;  
  productId: number;  
}  
interface ProductList extends Category, Product {  
  list: Array<string>;  
}  
const productDetails: ProductList = {  
  categoryName: 'Gadget',  
  productName: 'Mobile',  
  productId: 1234,  
  list: ['Samsung', 'Motorola', 'LG']};  
const listProduct = productDetails.list;  
const pname: string = productDetails.productName;  
console.log('Product Name is ' + pname);  
console.log('Product List is ' + listProduct);
```

Output:



```
D:\mstd\typescript>ts-node 11a.ts  
Product Name is Mobile  
Product List is Samsung,Motorola,LG
```

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.

Program:

```
class Product {  
  productId: number;  
  productName: string;  
  constructor(productId: number, productName: string) {  
    this.productId = productId;  
    this.productName = productName;  
  }  
}
```



```
getDetails(): string {  
    return 'Product id is : ' + this.productId+'\n Product name is :'+this.productName;  
}  
}  
const ar:Product[]=[];  
let i:number=1;  
for(i=1;i<=4;i++)  
{  
    ar.push(new Product(i,"str"+i));  
}  
console.log(ar);
```

Output:

```
C:\WINDOWS\system32\cmd.  ×  +  v  
D:\mstd\typescrip>ts-node 11b.ts  
[  
  Product { productId: 1, productName: 'str1' },  
  Product { productId: 2, productName: 'str2' },  
  Product { productId: 3, productName: 'str3' },  
  Product { productId: 4, productName: 'str4' }  
]
```



Exercise - 12

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().

Program:

```
class Product {  
    public productPrice = 150;  
    private productId: number=0;  
    public productName: string="Iphone";  
    public productCategory: string="Mobile";  
  
    set setProductId(productId:number){  
        this.productId=productId;  
    }  
    get getProductId():string{  
        return 'Product id :'+this.productId;  
    }  
}  
  
const ProductDetails:Product=new Product();  
ProductDetails.setProductId=199;  
console.log(ProductDetails.getProductId);
```

Output:

```
C:\WINDOWS\system32\cmd. X + v  
  
D:\mstd\typescrip>ts-node 12a.ts  
Product id :199
```

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.

Utility.ts

```
namespace ProductUtility{  
    export class Product1 {  
        static productPrice: string;  
        productId: number;  
        constructor(productId: number) {  
            this.productId = productId;  
        }  
    }  
}
```



```
}  
getProductId(): string {  
    return 'Product id is : ' + this.productId;  
}}}
```

12b.ts

```
///import prod=ProductUtility;  
let pe=new prod.Product1(8);  
interface ProductList {  
    list: Array<string>;  
}  
const productDetails: ProductList= {  
    list: ['Samsung', 'Motorola', 'LG']  
};  
const listProduct = productDetails.list;  
console.log('Product List is ' + listProduct);  
console.log(pe.getProductId())
```

Output:

```
C:\WINDOWS\system32\cmd. X + v  
  
D:\mstd\typescrip>tsc --outFile Final.js Utility.ts 12b.ts  
  
D:\mstd\typescrip>node Final.js  
Product List is Samsung,Motorola,LG  
Product id is : 8
```