**1 a) Aim: Write a html program that makes use of basic tags like <html>, <head>, <title>, <body>,<p>,<hr>,<br><h1> to <h6>,<!-- --> tags and their attributes.**

**Description:**

**Block Formatting Tags**

We can use following tags to format blocks of text within HTML document. Some of the most frequently used Block-formatting tags are :

**Body tag (<body>)**

* Body tag contains some attributes such as bgcolor, background etc.
* *bgcolor* is used for setting the background color of a webpage which takes background color name or hexadecimal number such as #000000 to #FFFFFF
* background attribute used for setting mage as a background for webpage and it will take the path of the image which you can place as the background image in the browser.
* **Syntax:**
* <body bgcolor=”name/#rrggbb” background=”image name”> . . . </body>

**Paragraph tag (<p>)**

* HTML documents are divided into paragraphs.
* Paragraphs are defined with the <p> tag.
* Most text is part of a paragraph of information. Each paragraph is aligned to the left, right or center of the page by using an attribute called as align.
* **Syntax:**
* <p [align=”left” | “right” | “center”] > . . . </p>

**Heading tag(<Hn>)**

* HTML is having six levels of heading that are commonly used.
* The largest heading tag is <h1>.
* The different levels of heading tags are <h1> ,<h2>, <h3>, <h4>, <h5> <h6>.
* Each heading tag has an attribute called as align which can be set to left, center, or right. By default all headings align left.
* ***Syntax:***
* <h1 [align=”left” | “right” | “center”]> . . . </h1>
* <h2 [align=”left” | “right” | “center”]> . . . </h2> ……….
* <h6 [align=”left” | “right” | “center”]> . . . </h6>

**<hr> tag**

* This tag places a horizontal line across the screen.
* These lines are used to break up the page.
* This tag does not require an end tag.
* This tag also contains attributes which determines how the rule will be displayed.
* It can be aligned but by default is centered on the screen.
* The *size* attribute specifies the thickness of the rule in pixels.
* *noshade* draws the rule as a single thick line rather than giving it‟s default 3D appearance.
* ***Syntax:***
* <hr align=”left” | “right” | “center” size=”n” [noshade]/>.

**Comments**

* Comments can be inserted into the HTML code to make it more readable and understandable.
* Comments are ignored by the browser and are not displayed.
* HTML comment begins with “<!--“ and ends with “-->”.
* There should not be a space between angular bracket and exclamation mark. Each comment can contain as many lines of text as you like. If comment is having more lines, then each line must start and end with -- and must not contain -- within its body.
* <! -- this is a single line comment line - ->
* <! -- this is a multiline comment --
* -- spawned over --
* -- three line -->

**Line break tag**

* This tag is used to the break the line and start from the next line.
* It is an empty tag.

**<br/>**

**Program:**

<!DOCTYPE html>

<html>

<head>

<title>Working of Basic Tags</title>

</head>

<body bgcolor="skyblue">

<!--Basic Tags  -->

<center><p><h1>ABOUT ADITYA ENGINEERING COLLEGE<h1></p>

<hr><hr>

<br/>

<h1>Engineering College was established in the academic year 2001-02</h1>

<h2>The College is situated in an eco-friendly area of 180 acres with thick greenery at Surampalem, Gandepalli Mandal,East Godavari District, Andhra Pradesh.</h2>

<h3>The College is 15 KM away from Samalkot Railway Station on Howrah-Chennai Railway line in South Central Railway.</h3>

<h4>The College has four academic Buildings with a total carpet area of 35,425 Mts2.</h4>

<h5> Apart from two boys hostels and one girls hostel buildings.</h5>

<h6>The particulars of academic buildings and the departments / offices accommodated are as follows.

</h6>

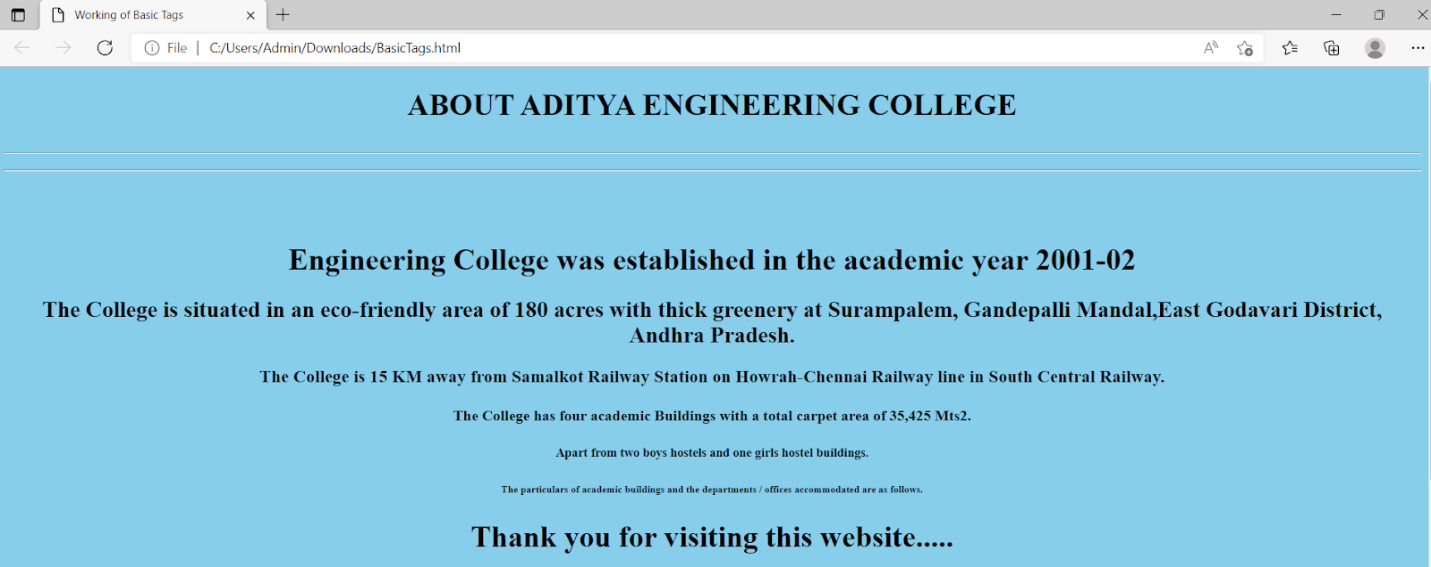
<h1>Thank you for visiting this website.....</h1>

</center>

</body>

</html>

**Output:**



**1 b) Aim: Write a HTML program, that makes use of text formatting tags like** **<b>,<i>,<u>,<strong>,<sub>,<sup>,<tt>,<pre>,<font>**

**Description:**

**Text Formatting Tags**

We can use character/text formatting tags to format a text block that is as small as a single character or as large as an entire document. Some of the most frequently used Character-formatting tags are :

**1. Boldface tag**

* This tag is used for implement bold effect on the text
* <b> ……. </b>

**2. Italic tag**

* This implements italic effects on the text.
* <i>…….</i>

**3. Underline tag**

* This is used to specify that the selected text be displayed with underline.
* <u>. . . </u>

**4. strong tag**

* This tag is used to always emphasized the text
* <strong>. . . </strong>

**5. sub and sup tag**

* These tags are used for subscript and superscript effects on the text.
* <sub>. . .</sub>
* <sup>. . .</sup>

**6.tt tag**

* This tag is used to give typewriting effect on the text
* <tt>. . .</tt>

**7.Pre-formatted text tag**

* It Considers spaces, new lines etc. and as it is prints the information.

<PRE>. . .</PRE>

**8. font tag**

* This sets font type, size, color and relative values for a particular text.
* Absolute font sizes are can be set from 1 to 7.
* Relative font sizes are set by using +/- 1 to 7.
* The color of the text is set by *color* attribute. This takes hex value which represents the amounts of red, green and blue in a chosen color.
* ***Syntax:***
* < font face=”font name ” size=”[+|-]n” color=”#rrggbb”>
* **face:** The style of the letter can be specified like arial, times new roman etc…
* **size:** The size can be specified .
* **color:** The color of the text to be displayed is specified through the color attribute.

**Program:**

<!DOCTYPE html>

<html>

<head>

<title>Formatting Tags</title>

</head>

<body bgcolor="silver">

<!-- Main Heading -->

<center><h4><u><b>Aditya</b><sub>Enlightens the Nescience</sub></u></h4></center>

<hr><br>

<center><h3><u>ABOUT ADITYA ENGINEERING COLLEGE</u></h3></center>

<p><b><u>Aditya</u></b> Engineering College was established in the academic year <strong>2001-02</strong> under the aegis of Aditya Academy, Kakinada with the approval of AICTE and Affiliated to JNTU with an intake of <i>180 in three UG Courses</i> in Engineering & Technology.</p>

<p>The College is situated in an <font face="tahoma" color="red">eco-friendly area of 180 acres with thick greenery at <u>Surampalem, Gandepalli Mandal</u></font>, East Godavari District, Andhra Pradesh. The College is <small>15 KM</small> away from Samalkot Railway Station on <big>Howrah-Chennai Railway</big> line in South Central Railway. The College is 35 Km away from Kakinada and Rajahmundry on ADB Road.</p>

<p>The College has four academic Buildings with a total carpet area of 35,425 Mts<sup>2</sup>. apart from two boys hostels and one girls hostel buildings. The particulars of academic buildings and the departments / offices accommodated are as follows.</p>

<p>The total<tt> student strength is 5052 with faculty strength of 355</tt> thus giving rise to healthy faculty student ratio.</p>

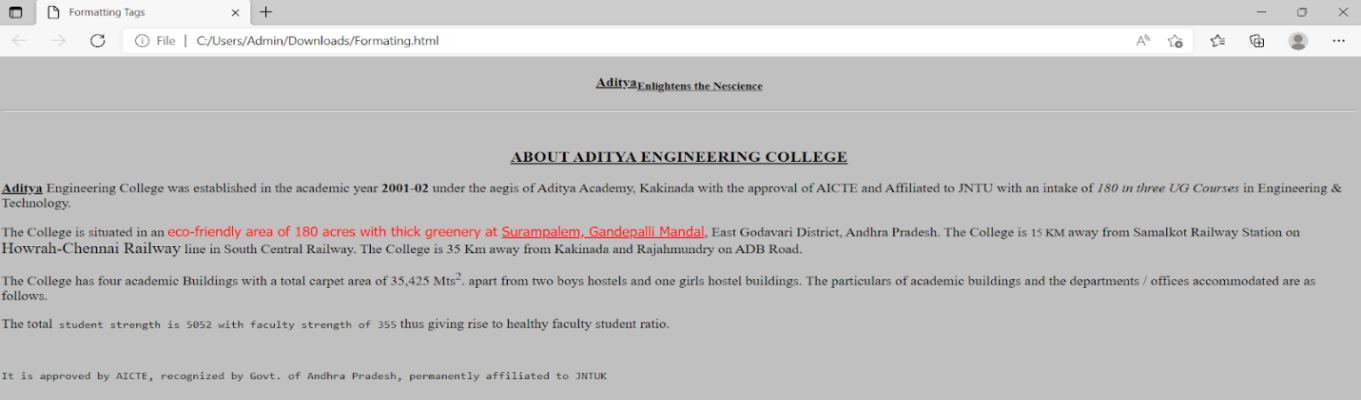
<br>

<pre>It is approved by AICTE, recognized by Govt. of Andhra Pradesh, permanently affiliated to JNTUK </pre>

</body>

</html>

**Output:**



**1 c)Aim: Write a HTML program, to explain the working of lists. Note: It should have an ordered list, unordered list, nested lists and ordered list in an unordered list and definition lists.**

## Description:

List is a collection of items**.**

* Types of lists are
* Ordered lists
* Unordered lists
* Nested lists
* Directory lists
* Definition lists

**Ordered Lists:**

* Ordered lists are also called as Numbered lists or Sequenced lists.
* In Ordered list, the list of items has an order that is signified by numbers, hence called as a numbered list.
* An ordered list should start with <OL> element, which is immediately followed by a <LI> and end of ordered list is specified by </OL> element.
* Different ordered list types like numeral list; alphabet list etc can be specified with TYPE tag.
* Optional parameter with <OL> tag is START attribute, which indicates the starting number or alphabet.

**Syntax:**

* <OL [type={“1” or “I”or “A” or “a” or “i”}] START=n>
* <LI>item1 </LI>
* <LI>item2 </LI>
* :
* :
* <LI>item n </LI>
* </OL>

**Different Ordered list types**

* Type=”1” (default) e.g.1,2,3,4…..
* Type=”A” Capital letters e.g.A,B,C…
* Type=”a” Small letters e.g. a,b,c……
* Type=”I” Large roman letters e.g. I, II, III,…

**Unordered Lists:**

* Unordered lists are also called as UnNumbered lists .
* The unordered list items are used to present a list of items, which are typically separated by white space/ or marked by bullets.
* An unordered list should start with <UL> element, which is immediately followed by a <LI> and end of ordered list is specified by </UL> element.
* TYPE attribute can also be added to <UL> tag that it indicates that the displayed bullet along with the list of items circle, square or disk.
* By default it is disc.

**Syntax:**

* <UL [type={square or disc or circle}] >
* <LI>item1 </LI>
* <LI>item2 </LI>
* :
* :
* <LI>item n </LI>
* </UL>

**Nested Lists:**

* Both ordered list and unordered list are used.
* We can put an ordered list in an unordered list and viceversa.

**Directory Lists:**

* A Directory list element is used to present a list of items containing up to 20 characters each.
* A directory list must start with <DIR> element, which s immediately followed by <LI> element.

**Definition Lists:**

* A definition list is a list of Definition terms.
* To create a definition list it must start with <DL> and immediately followed by <DT> tag
* <DD> tag is used to give a description of the definition.

**Program:**

<!DOCTYPE html>

<html>

<head>

<title>Working of Lists</title>

</head>

<body bgcolor="lightgray">

<!-- order list -->

<h3>Ordered List</h3>

<ol type="1" start=3>

 <li>CSE</li>

  <li>EEE</li>

   <li>ECE</li>

  <li>CIVIL</li>

 <li>MECH</li>

</ol>

<!-- unorder list -->

<h3>Unordered List</h3>

<ul>

 <li>CSE</li>

  <li>EEE</li>

   <li>ECE</li>

 <li>CIVIL</li>

 <li>MECH</li>

</ul>

<!-- Nested list -->

<h3>Nested List</h3>

The contents of a project are:

<ul type="circle">

<li>Introduction</li>

    <ol type="i">

        <li>Area Chosen</li>

        <li>Existing System - Disadvantages</li>

        <li>Proposed System - Advantages</li>

    </ol>

<li>SRS</li>

    <ol type="i">

        <li>Purpose</li>

        <li>Scope</li>

        <li>Functional Requirements</li>

        <li>Non-Functional Requirements</li>

        <li>Integration Requirements</li>

        <li>Security Requirements</li>

        <li>Usecases - Working</li>

    </ol>

<li>Literature Survey</li>

    <ol type="i">

        <li>Selecting Papers</li>

        <li>Work in each paper - Disadvantages</li>

        <li>Conclusion</li>

    </ol>

<li>Design</li>

    <ol type="i">

        <li>Database Schema</li>

        <li>UML Diagrams</li>

    </ol>

<li>Technologies Used</li>

<li>Coding</li>

<li>Test Cases</li>

    <ol type="i">

        <li>Integration tests - Results</li>

        <li>Unit tests - Results</li>

        <li>Module tests - Results</li>

    </ol>

<li>Conclusion</li>

<li>Future Scope</li>

<li>Bibliography</li>

</ul>

<br>

<!-- Order list in a unorder list-->

<h3>Ordered list in a Unorder List</h3>

<ol>

<li>CSE</li>

<li>EEE</li>

<ul>

<li>CSE</li>

<li>EEE</li>

</ul>

</ol>

<!-- Definition list-->

<h3>Definition list</h3>

<dl>

<dt>HTML</dt>

<dd>HyperText Markup Language is used for generating the webpages.</dd>

<dt>Javascript</dt>

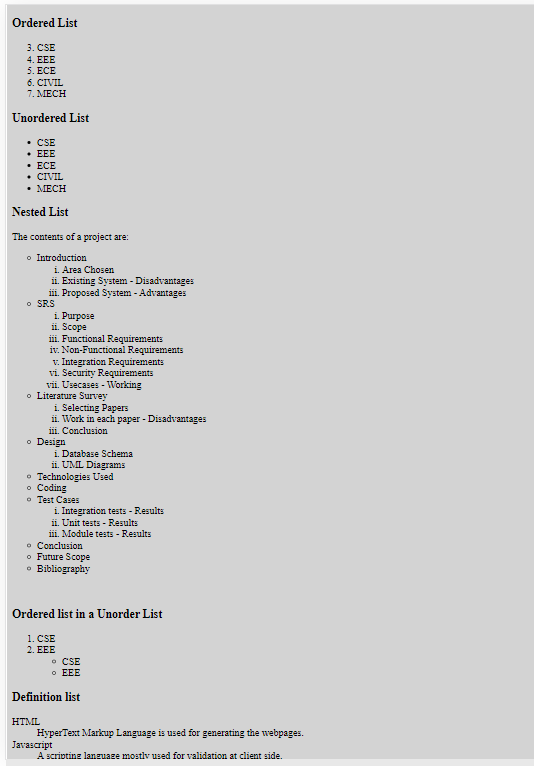
<dd>A scripting language mostly used for validation at client side.</dd>

</dl>

</body>

</html>

**Output:**



**1 d) Aim: Write a HTML program, to explain the working of hyperlinks using tag and href, target Attributes.**

**Note: Use text to link →https://www.aec.edu.in/**

**Use image to link →https://www.aec.edu.in/?p=Gallery**

**Description:**

**Hyperlinks**

* It is a block formatting tag
* The real power and flexibility of HTML is in Hyperlinks.
* Hyperlinks are created with anchor tag**(<a>)**
* We can use hyperlink to
* 1. create links with HTML pages
* 2. link different HTML pages
* 3. access services at other Internet sites

**Anchor tag:**

* The anchor tag is created by <a> . . .</a> tags.
* The tag has three sections: the address of the referenced document, a piece of text displayed as link, and the closing tag.

***Syntax:***

* <a href=”address”> Text </a>
* *href* means Hypertext references that can be used for giving the path of a file which you want to link.
* The “Text” between the <a> . . . </a> tags acted as a hyperlink. This text is called *“hypertext”.* When we click on this text, the linked page or file will be displayed.

We can also replace this text with image. In that case, the image will act as a hyperlink.

* The target attribute specifies where to open the linked element.

**Syntax**

* <a target="\_blank|\_self|\_parent|\_top|*framename*">
* Attribute Values

|  |  |
| --- | --- |
| Value | Description |
| \_blank | Opens the linked document in a new window or tab |
| \_self | Opens the linked document in the same frame as it was clicked  (this is default) |
| \_parent | Opens the linked document in the parent frame |
| \_top | Opens the linked document in the full body of the window |
| *Framename* | Opens the linked document in the named iframe |

**Images**

* Images are second aspect of pleasant Web experience.
* We can add images to an HTML page to either improve its appearance or present important information.
* To add images to an HTML page, we can use several image formats. These formats include „*gif‟, „jpg‟ and „png‟.*
* Images can be added in two different ways:
* 1. By using *„background‟* attribute of <body> tag
* 2. By using <img> tag

**The Image tag**

* We can use image tag in an HTML page to add images along with text.
* An image added using the image tag occupies space within the HTML page.
* To add image to HTML page, use *<img>* tag.
* ***Syntax:***
* <img src=”URL” height=”n” width=”n” align = “left” | “right” | “top”|”middle” alt = ” string ” border=n >

**Attributes:**

* ***src :***
* The important parameter is SRC, because it specifies the location of the source file
* If the file is included in the current directory, no need to specify the path otherwise given the entire path.
* We can also use uniform resource locator(URL).
* b) ***align :*** used to specify the vertical alignment of an image
* c) ***height*** : used to specify the vertical area that an image will occupy in HTML page
* d) ***Width*** : used to specify the horizontal area that an image will occupy in HTML page
* e) ***alt :*** used to specify the text when browser unable to display the image or image not available.
* f) ***border:*** Specifies the border width along the image.

**Program:**

<!DOCTYPE html>

<html>

<head>

<title>Working of hyperlinks using tag and href,target Attributes</title>

</head>

<body bgcolor="#00FFFF">

<p>To visit Aditya Engineering College Website,<a href="https://www.aec.edu.in">Click here</a></p>

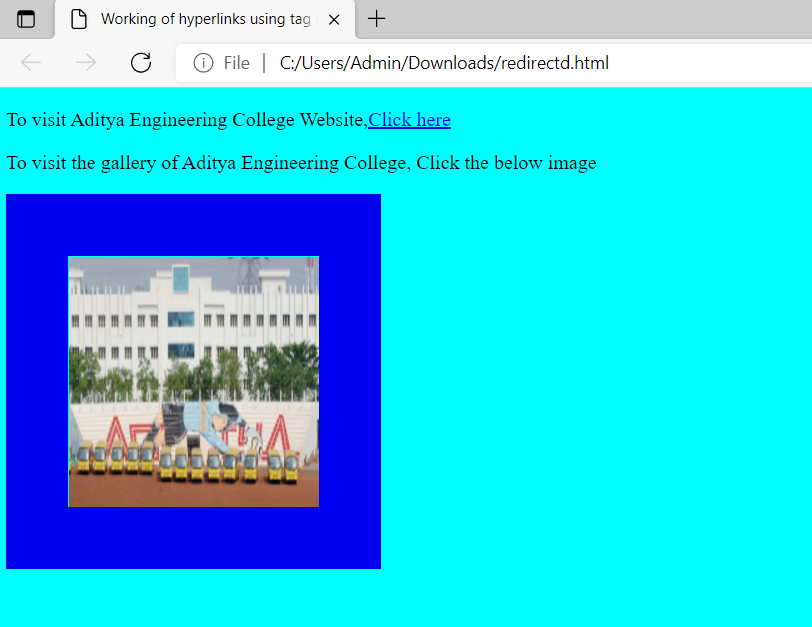
<p>To visit the gallery of Aditya Engineering College, Click the below image</p>

<a href="https://www.aec.edu.in/?p=Gallery" target="\_blank"><img src="aecglR.jpg" alt="Gallery" width="200" height="200" border="50"></a>

</body>

</html>

**Output:**



**2 a) Aim: Write a HTML program, to explain the working of tables by preparing a timetable. (Note: Use <caption> tag to set the caption to the table & also use cell spacing, cell padding, border,rowspan, colspan etc.).**

**Description:**

**Tables**

* Tables are defined with the **<table>** tag.
* A table is divided into rows with the **<tr>** tag
* Each row is divided into data cells with the **<td>** tag.
* td stands for "table data," and holds the content of a data cell.
* A <td> tag can contain text, links, images, lists, forms, other tables, etc.
* Header information in a table are defined with the **<th>** tag.
* All major browsers display the text in the <th> element as bold and centered

We can use various elements to specify the details of a table. Many table elements also take attributes, which allows you to further specify the look of the table.

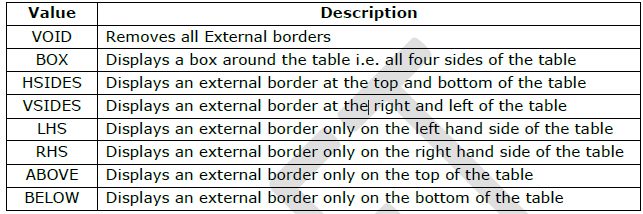
* TABLE
* Table row
* Table data
* Table Heading

**The TABLE Element**

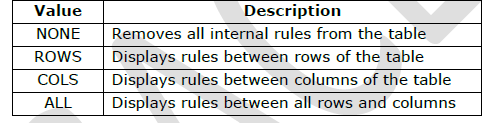
* The TABLE element is the container element for table and uses the <TABLE>...</TABLE> tags to enclose all the other table tags.
* If the <TABLE> tag is omitted or not closed, the browser ignores all the other tags that you specify for the table.
* Everything that we write between these two tags will be within a table.
* The attributes of the table will control in formatting of the table.

***The <TABLE> tag has the following attributes:***

* 1. ***align***: used to specify the alignment of a table in a HTML page.
* ***2.border*:** used to specify the thickness of the table border in pixels.
* ***3.bgcolor***: used to specify the background color for the table.
* 4. ***frame***: used to specify the which side of the outer border is visible in the browser.
* We must specify the border attribute before you specify the FRAME attribute.



***rules***: used to specify the borders between cells. You must specify the border attribute before you specify the FRAME attribute.



**Cell Attributes**

* ***cellspacing***: used to specify the spacing between cells in pixels.
* ***cellpadding***: used to specify the spacing between cell content and cell wall in pixels.
* ***height***: used to specify the height of table in pixels or %.
* ***width***: used to specify the width of table in pixels or %.
* ***Syntax:***
* ***<****TABLE align=”left | right | center” border=”n” bgcolor=”#rrggbb” cellspacing=”n”* *cellpadding=”n” frame=”value” rules=”value” height=”n | %” width=”n | %”>*
* *</TABLE>*

**Table Row:**

* Table row element is used to create rows in a table.
* The Table row element uses the <TR> tag to create a row.
* The <TR> tag has the following attributes:
* ***align***: used to specify the horizontal alignment of the contents for cells of a row.
* ***valign:*** used to specify the vertical alignment of the cell content for all cells of the row.
* ***bgcolor:*** used to specify the background color of the row.
* ***Syntax***:
* <TR *align=”left | right | center”*
* *valign=”top | bottom |middle”*
* *bgcolor=”#rrggbb”* > ….. </TR>

**Table Data Element:**

* The <TD> tag has the following attributes:
* ***colspan:*** used to specify the number of columns the cell can span.
* ***rowspan*:** used to specify the number of rows the cell can span
* **align:** used to specify the horizontal alignment of the data within a cell.
* ***valign***: used to specify the vertical alignment of data within the cell.
* ***bgcolor***: used to specify the background color of the cell.
* **Syntax:**
* <TD *align=”left | right | center”*
* *valign=”top | bottom |middle”*
* *bgcolor=”#rrggbb”*
* *colspan=”n”*
* *rowspan=”n”*> ….. </TD>

**Program:**

<html>

<head>

<title>Table</title>

</head>

<body bgcolor="#03d3fc">

<center>

<h1><u>TIMETABLE CSE-B</u></h1>

<table border="1" cellspacing="3" cellpadding="10">

<caption> VIth sem Period table </caption>

<tr>

<th><b>DAYS</b></th>

<th><b>9:30-10:20</b></th>

<th><b>10:20-11:10</b></th>

<th><b>11:10-12:00</b></th>

<th><b>12:00-12:50</b></th>

<th><b>12:50-1:30</b></th>

<th><b>1:30-4:00</b></th>

</tr>

<tr>

<td align="center"> MONDAY </td>

<td> ML </td>

<td> OOAD </td>

<td> CC </td>

<td> DWDM </td>

<td rowspan="6" style="bg-color:gray" align="center"> <b><i>L</br><br> U </br><br> N </br><br> C </br><br> H </i></b>

</td>

<td align="center"colspan="3" style="background-color:lightgray"> TRAINING </td>

</tr>

<tr>

<td align="center"> TUESDAY </td>

<td> ML </td>

<td> OOAD </td>

<td> DWDM </td>

<td> RPA </td>

<td align="center"colspan="3" style="background-color:lightgray"> TRAINING </td>

</tr>

<tr>

<td align="center"> WEDNESDAY </td>

<td> OOAD </td>

<td> WT </td>

<td> ML </td>

<td> CC</td>

<td align="center"colspan="3" style="background-color:lightgray"> TRAINING </td>

</tr>

<tr>

<td align="center"> THURSDAY </td>

<td> RPA </td>

<td> DWDM </td>

<td> RPA </td>

<td> WT </td>

<td align="center"colspan="3" style="background-color:lightgray"> TRAINING </td>

</tr>

<tr>

<td align="center"> FRIDAY </td>

<td> CC </td>

<td colspan="3" align="center" style="background-color:LightGray"> WT LAB(LAB-4) </td>

<td align="center"colspan="3" style="background-color:lightgray"> TRAINING </td>

</tr>

<tr>

<td align="center"> SATURDAY </td>

<td colspan="3" align="center" style="background-color:LightGray"> DW & OOADLAB(LAB-4) </td>

<td> WT/PROCTORING</td>

<td align="center"colspan="3" style="background-color:lightgray"> TRAINING </td>

</tr>

</table>

</center>

</body>

</html>

**Output:**

|  |
| --- |
|  |

**2 b) Aim: Write a HTML program, to explain the working of frames, such that page is to be divided into 3 parts on either direction. (Note: first frame → image, second frame → paragraph, third frame → hyperlink. And also make sure of using “no frame” attribute such that frames to be fixed).**

**Description:**

**Frames:**

* Frames are not supported in HTML5.
* A frame provides a facility to display one page at a time.
* Collection of these frames into a set provides a facility to display more than one page at a time on browser and is referred as a **frameset.**
* **A Frameset** is a collection of frames.
* **A web page** containing frame elements is called a framed page.
* **A framed page** begins with **<frameset>** & ends with **</frameset>** tag.
* Each individual frame is identified through **<frame>** tag.

**Syntax:**

* <frameset rows/cols="45%,55%">
* <frame name="framename" src=“filename" noresize scrolling="yes">
* <frame name=" framename " src=" filename " noresize>
* </frameset>

**Program:**

**Mainframe.html**

<html>

<head>

<title> FrameDemo</title>

<frameset rows="30%,40%,30%">

<frame name="f1" src="image.html" scrolling="Yes">

<frameset cols="30%,70%">

<frame name="f2" src="link.html">

<frame name="f3">

<marquee scrollamount="10">!!!!!!!</marquee>

</frameset>

<frame name="f4" src="textlink.html">

</frameset>

</head>

</html>

**Image.html**

<html>

<head>

</head>

<body bgcolor="#d8dde6">

<br><br>

<center><img src="aec.png"></center>

</body>

</html>

**Link.html**

<html>

<head>

</head>

<body bgcolor="#e87b4d">

<center><a href="https://www.aec.edu.in/?p=About-AEC" target="f3">About Us</a></center><br><br>

<center><a href="https://www.aec.edu.in/?p=Vision-Mission" target="f3">Vision & Mission</a></center><br><br>

<center><a href="https://www.aec.edu.in/?p=Chairmans-Message#chairmans\_message" target="f3">Chairman’s Message </a></center><br><br>

<center><a href="https://www.aec.edu.in/?p=Vice-Chairmans-Message#vice\_chairmans\_message" target="f3">Contact Us</a></center>

</body>

</html>

**Textlink.html**

<html>

<head>

</head>

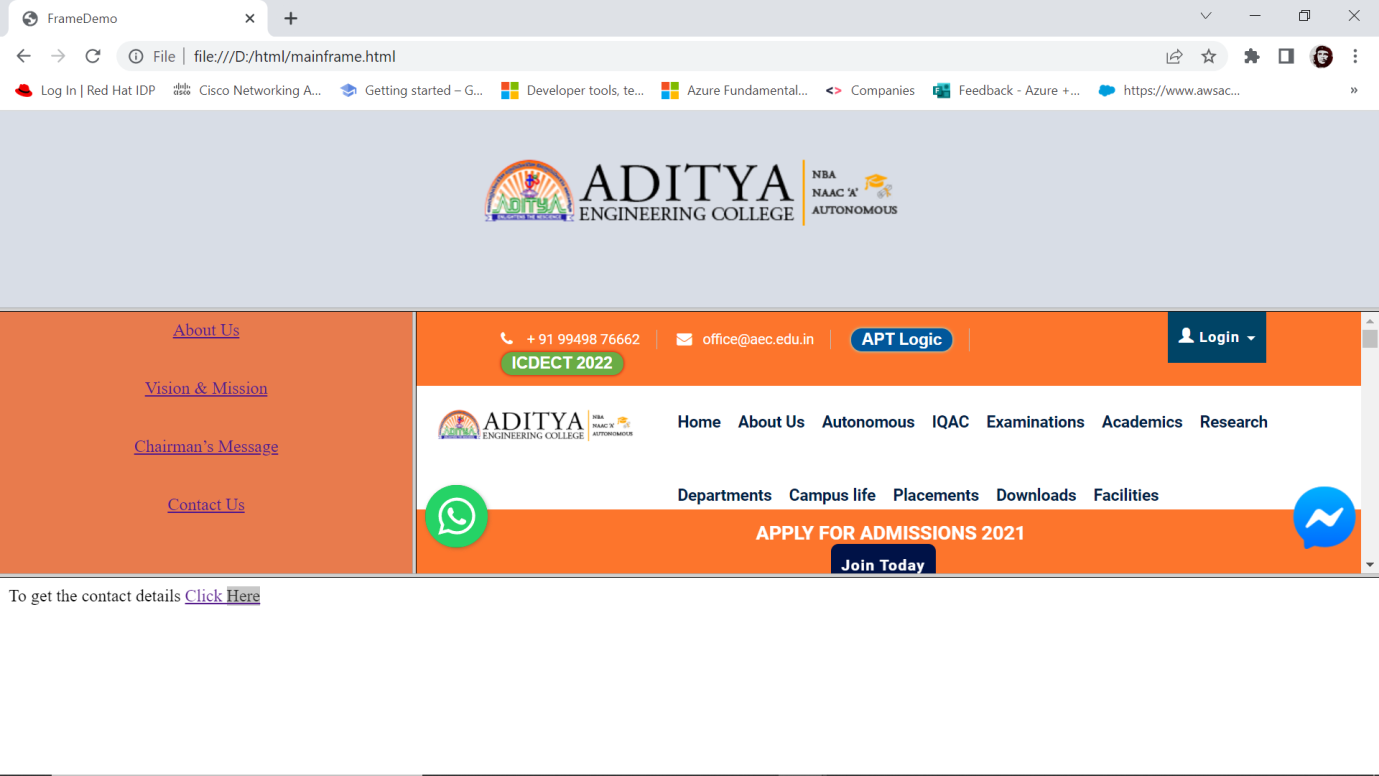
<body>

To get the contact details <a href="https://www.aec.edu.in/?p=contactus" target="f3">Click Here</a>

</body>

</html>

**Output:**



**2 c) Aim: Write a HTML program, to explain the working of forms by designing Registration form. (Note: Include text field, password field, number field, date of birth field, checkboxes, radio buttons, list boxes using and two buttons ie: submit and reset. Use tables to provide a better view).**

**Description:**

**Forms**

* HTML provides several user interactive elements such as **label, text fields, buttons, radio button, check box, combo box, list box, text area and password fields**.
* The **<form>** tag is used to create forms.
* The important attributes of **<form>** tag are name, **METHOD** and **ACTION.**
* **Method:** This parameter specifies how the data entered in the form can be sent to destination.
* There are 2 methods of sending information GET & POST

**Labels**

* HTML is not provides any separate tag for label.
* The text which is preceded by the interfacing element is taken as a label.

**Text field**

* Text is typically required to place one line string.
* Text fields can be created by using <input>tag.
* Syntax: <input type=”text” size=n name=”text name”>

**Buttons**

There are three types of buttons

* + **Button**
* Syntax: <input type=button value=”click me”>
  + **Submit**
* Syntax: <input type=submit value=”submit”>
  + **Reset**
* Syntax: <input type=reset value=”clear”>

**Radio buttons**

* + To display set of option to user, where he can select one among them, radio buttons are used.
  + ***Syntax***: <input type=”radio” name=”text name” value=”text” checked>
  + All the name attributes of group of radio buttons must be same and all the value attributes are different.
  + Value indicates the value that is stored while selecting particular button, which returns to name of group.
  + Checked parameter indicates the initial selection that we give as default.

**Checkboxes**

* These are similar to radio buttons as we can select more than one option among the options.
* Syntax: <input type=”checkbox” name=”text name” value=”text” checked>

**Combo box**

* Combo box displays one at a time and by clicking on combo arrow displays other options available.
* Syntax: <select name=“selectname” size=n multiple>
* <option value=“text” label=“labelname” selected>
* <option>
* </select>

**TextArea**

* Textarea is typically required to place multi line string.
* Syntax: <textarea name=”text name” cols=n rows=n> </teaxtarea>

**Password field**

* When user typed the information in a text field it is displayed as it is.
* If we want to display it in encoded form, use the password field.
* Syntax: <input type=”password” name=”pwd” size=n >

**Program:**

<html>

<head>

<title>Formdemo</title>

</head>

<body>

<table align="center">

<caption="Formdemo">

<form name="f1" method="get">

<tr>

<td>Username: </td>

<td><input type="text" name="uname" size="10" maxlength="15"></td>

</tr>

<tr>

<td>Password:</td>

<td><input type="password" name="pwd" size="10" maxsize="15"></td>

</tr>

<tr>

<td>Address:</td>

<td><textarea name="ta" rows="5" cols="10"></textarea></td>

</tr>

<tr>

<td>Languages known:</td>

<td><input type="checkbox" name="c1" value="Te" checked>Telugu</br>

<input type="checkbox" name="c2" value="Ta">Tamil</br>

<input type="checkbox" name="c3" value="Hi">Hindi</br>

<input type="checkbox" name="c4" value="En">English</br>

</td>

</tr>

<tr>

<td>Programming Proficiency:</td>

<td>

<input type="radio" name="r1" value="C">C</br>

<input type="radio" name="r1" value="Java">Java</br>

<input type="radio" name="r1" value="Python" checked>Python</br>

<input type="radio" name="r1" value="PHP">PHP</br>

</td>

</tr>

<tr>

<td>Willing to work at:</td>

<td>

<select name="s1" multiple>

<option value="Mu">Mumbai</option>

<option value="ch" label="Chennai">Chennai</option>

<option value="Hy">Hyderbad</option>

<option value="viz" selected>Vizag</option>

</select>

</td>

</tr>

<tr>

<td colspan="2"><input type="submit" value="submit" >

<input type="reset" value="reset">

</td>

</tr>

</form>

</table>

</body>

</html>

**Output:**



**2 d) Aim:**  **Write a HTML program, that makes use of <article> ,<aside> ,<figure> , <figcaption> ,<footer> ,<header> ,<main> ,<nav> ,<section> ,<div> ,<span> tags**

**Description:**

**Article Tag:**

* The <article> tag specifies independent, self-contained content.
* An article should make sense on its own and it should be possible to distribute it independently from the rest of the site.
* Potential sources for the <article> element:

1. Forum post

2. Blog post

3. News story

4. Comment

* Syntax:
* <article>….</article>

**Aside Tag:**

* The <aside> tag defines some content aside from the content it is placed in.
* The aside content should be related to the surrounding content.
* Syntax:
* <aside>…</aside>

**Figure and Figcaption Tags:**

* The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
* While the content of the <figure> element is related to the main flow, its position is independent of the main flow, and if removed it should not affect the flow of the document.
* The <figcaption> tag defines a caption for a <figure> element.
* The <figcaption> element can be placed as the first or last child of the <figure> element.
* Syntax:
* <figure>
* <img src=“” alt=“”>
* <figcaption>….</figcaption>
* </figure>

**Footer Tag:**

* The <footer> tag defines a footer for a document or section.
* A <footer> element should contain information about its containing element.
* A <footer> element typically contains:
* authorship information
* copyright information
* contact information
* sitemap
* back to top links
* related documents
* You can have several <footer> elements in one document.
* Contact information inside a <footer> element should go inside an <address> tag.

**Syntax:**

* <footer>
* <address>….</address>
* </footer>

**Header Tag:**

* The <header> element represents a container for introductory content or a set of navigational links.
* A <header> element typically contains:
* one or more heading elements (<h1> - <h6>)
* logo or icon
* authorship information
* You can have several <header> elements in one document.
* A <header> tag cannot be placed within a <footer>, <address>, <article> or another <header> element.
* Syntax:
* <header>….</header>

**Main Tag:**

* The <main> tag specifies the main content of a document.
* The content inside the <main> element should be unique to the document. It should not contain any content that is repeated across documents such as sidebars, navigation links, copyright information, site logos, and search forms.
* There must not be more than one <main> element in a document. The <main> element must NOT be a descendant of an <article>, <aside>, <footer>, <header>, or <nav> element.

**Syntax:**

* <main>….</main>

**Nav Tag:**

* The <nav> tag defines a set of navigation links.
* Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major block of **navigation links**.

**Syntax**:

* <nav>
* <a href””>…</a>
* <a href””>…</a>
* <a href””>…</a>
* </nav>

**Section Tag:**

* The <section> tag defines sections in a document, such as chapters, headers, footers, or any other sections of the document.

**Syntax:**

* <section>…</section>

**Div Tag:**

* The <div> tag defines a division or a section in an HTML document.
* The <div> tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript.
* The <div> tag is easily styled by using the class or id attribute.
* Any sort of content can be put inside the <div> tag!

**Syntax:**

* <div>…. </div>

**Span Tag:**

* The <span> tag is an inline container used to mark up a part of a text, or a part of a document.
* The <span> tag is easily styled by CSS or manipulated with JavaScript using the class or id attribute.
* The <span> tag is much like the [<div>](https://www.w3schools.com/tags/tag_div.asp) element, but <div> is a block-level element and <span> is an inline element.

**Syntax:**

* <span>….</span>

**Program:**

<!DOCTYPE html>

<html>

<head>

<title>New Tags In HTML</title>

</head>

<main>

<header>

<img src="https://aec.edu.in/adityanew/images/logo.png">

</header>

<nav>

<a href="https://www.aec.edu.in/?p=About-AEC">About AEC</a><br>

<a href="https://www.aec.edu.in/?p=Vision-Mission">Vision & Mission</a><br>

<a href="https://www.aec.edu.in/?p=Chairmans-Message#chairmans\_message">Chairman's message</a><br>

<a href="https://www.aec.edu.in/?p=Vice-Chairmans-Message#vice\_chairmans\_message">Vice Chairman's messsage</a><br><br>

</nav>

<aside>

<marquee direction="up">

II B.TECH -I Semester supplementary exam fee notification (R16,R13 & R10) JNTUK <a href="">Click Here</a><br>

B.TECH I Semester (AR19) Supplementary Exams Time Table (For Students January-2022)<br>

B.TECH I Semester (AR20) Supplementary Exams Time Table January-2022<br>

MBA -III Semester (AR19) I sessional Exam Time Table Dec 2021<br>

Tima Table for B.Tech - IV Semester End Examinations Supplementary (AR17) - FEB 2022 <a href="">Click Here</a><br>

</marquee>

</aside>

<article>

<br>The Department of Computer Science and Engineering (CSE) is established with the inception of college in the year 2001. It administers bachelor's programs in Computer Science and Engineering with an intake of 180 students, as well as master's programs in Computer Science and Engineering with intake of 12 students.

</article>

<figure>

<img src="https://www.aec.edu.in/logos/departments/cse.jpg" height="250px" width="1345px">

<figcaption>

Department of CSE

</figcaption>

</figure>

<div>

VISION OF CSE:To produce competent professionals to become part of the industry and research organizations at the national and international levels through excellence in Computer Science& Engineering education and research.

</div>

<section>

Mision of CSE

M1:

Designing curriculum to meet the future challenges in Computer Science& Engineering and society by anticipating relevant trends.<br>

M2:

Inculcating the problem solving skills, leadership qualities in students and enable them to work in teams to become successful in their careers.<br>

M3:

Nurturing with Scientific Research in the field of Information Technology, enable students to involve in technological innovations.<br>

M4:

Transforming the Computer Science and Engineering department as a leader in imparting Computer Science and Engineering education and research by a committed faculty.<br>

</section>

The predicted placements for the Academic year 2021-22 is<span style="color: red;font: arial 12;">2321</span>

</main>

</html>

**Output:**



**3 a) Aim: Write a program to apply different types (or levels of styles or style specification formats) - inline, internal, external styles to HTML elements. (identify selector, property and value).**

**Description:**

**Introduction to CSS:**

* **CSS** stands for **C**ascading **S**tyle **S**heets
* Styles define **how to display** HTML elements
* Styles were added to HTML 4.0 **to solve a problem**
* **External Style Sheets** can save a lot of work
* External Style Sheets are stored in **CSS files**

**Types of CSS:**

* There are three ways of inserting a style sheet:

1. Inline styles
2. Internal style sheet
3. External style sheet

**Inline CSS:**

* **Inline sheets**  can be used to format **only one tag** at a time
* The **inline** cascading style sheet is a kind of style sheet which the styles can be applied to **html tags** only.
* Using inline sheets, we can apply uniform style on tags for the whole document.
* **Disadvantage:** Inline sheet is not much suitable for web page designing because the actual contents of web page are mixed with the presentation.

***Syntax:***

* <Tag style="property : value " >

**Internal CSS:**

* **Advantage** of Internal style sheet comparing with inline sheets, at a time **several** tags can be formatted with **internal** sheets, where as in **inline sheets only one tag** at a time can be formatted.
* **Disadvantage** : when we want to apply style to more than one document at a time then internal sheet of no use.

**Syntax:**

* <head>
* <style type=“text/css”>
* Tagname{
* Tagproperties;
* }
* </style>
* </head>

**External CSS:**

* When we want to apply style to more than one document at a time then external sheets are used.
* Total style elements are defined in a separate document and this document is added to required web page.
* By using this, we can use this style sheets in different web pages. So we can achieve **reusability** by using external sheets.
* The document where all the style formats are placed , should have extension .**css**
* This page can be called in the web page by using **LINK** tag.

***Syntax:***

* <link rel=”stylesheet” type=”text/css’’ href=”sample.css”>
* **rel:** Specifies relationship between documents.
* **type:** indicates which type we are including.
* **href**: indicates style sheet document address.

**Program:**

**Inline CSS:**

<!DOCTYPE html>

<html>

<body>

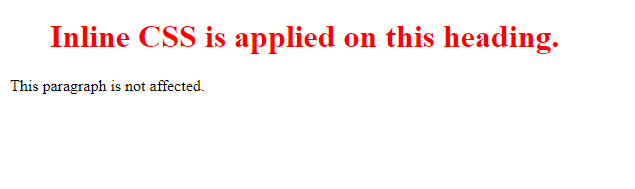
<h1 style="color:red;margin-left:40px;">Inline CSS is applied on this heading.</h1>

<p>This paragraph is not affected.</p>

</body>

</html>

**Output:**



**Internal CSS:**

<html>

<head>

<style>

.nav{

background-color:yellow;

list-syle-type:none;

text-align:center;

margin:0;

padding:0;

}

.nav li{

display:inline-block;

font-size:20px;

padding:20px;

}

</style>

</head>

<body>

<ul class="nav">

<li><a href="#home">Home</a></li>

<li><a href="#about us">About Us</a></li>

<li><a href="#achievements">Achievements</a></li>

<li><a href="#clients">Clients</a></li>

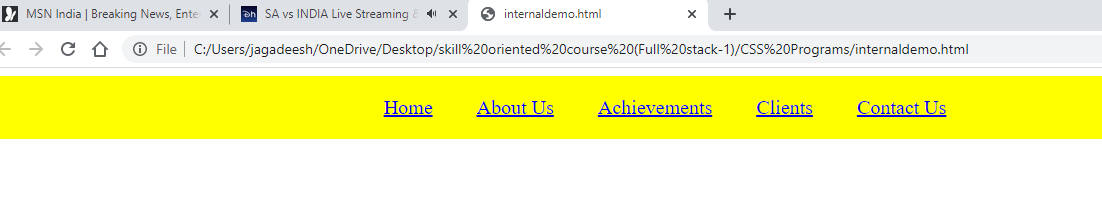
<li><a href="#contact us">Contact Us</a></li>

</ul>

</body>

</html>

**Output:**



**External CSS:**

**Style.css**

.nav{

background-color:yellow;

list-syle-type:none;

text-align:center;

margin:0;

padding:0;

}

.nav li{

display:inline-block;

font-size:20px;

padding:20px;

}

**External.html**

<html>

<head>

<link rel=”stylesheet” type=”text/css” href=”style.css”>

</head>

<body>

<ul class="nav">

<li><a href="#home">Home</a></li>

<li><a href="#about us">About Us</a></li>

<li><a href="#achievements">Achievements</a></li>

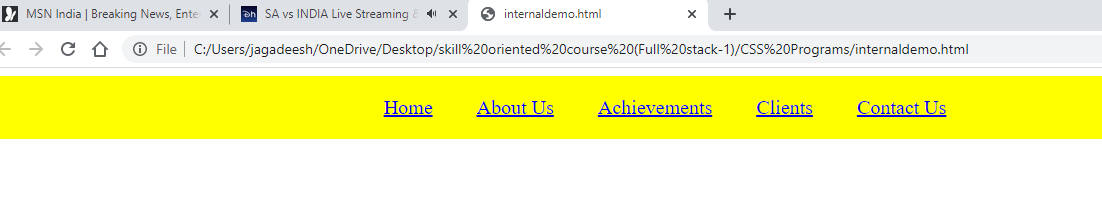
<li><a href="#clients">Clients</a></li>

<li><a href="#contact us">Contact Us</a></li>

</ul>

</body>

</html>

**Output:**

**3 b) Aim: Write a program to apply different types of selector forms**

1. **Simple selector (element, id, class, group, universal)**
2. **Combinator selector (descendant, child, adjacentsibling, general sibling)**
3. **Pseudo-class selector**
4. **Pseudo-element selector**
5. **Attribute selector**

**Description:**

Selectors are used to apply special effects.

Types of selectors are:

1. Simple selector

2. Class selector

3. Generic Selector

4. Universal selector

5. Id Selector

**Simple Selector:**

* The simple selector form is a single element to which the property and value is applied.
* Syntax:
* <head>
* <title>Simle selectors</title>
* <style type="text/css">
* tagname{
* properrties;
* }
* </style>
* </head>

**Class Selector:**

* Using class selector we can apply different styles to same element.
* Syntax: <head>
* <style type=“text/css”>
* Tagname.classname{
* Properties;
* }
* </style>
* </head>
* <tagname class=“classname”>….</tagname>

**Generic Selector:**

* The class can be defined in the generalized form.
* So that the particular class can be applied to any tag.
* Syntax: <head>
* <style type=“text/css”>
* .classname{
* Properties;
* }
* </style>
* </head>
* <tagname class=“classname”>….</tagname>

**Universal Selector:**

* This selector can be applied to all the elements in the document.
* This selector is denoted by \* symbol.
* Syntax:<head>
* <title>Universal selectors</title>
* <style type="text/css">
* \*
* {
* properties;
* }

**Id Slector:**

* The id selector is used to specify a style for a single, unique element.
* The id selector uses the id attribute of the HTML element, and is defined with a "#".
* Do **not** start an ID name with a number
* **Syntax:**
* #para1  
   {  
   text-align:center;  
   color:red;  
   }
* <tagname id=“idname”> ……</tagname>

**Attribute Slector:**

* The [attribute] selector is used to select elements with a specified attribute.
* The [attribute="value"] selector is used to select elements with a specified attribute and value.
* The [attribute~="value"] selector is used to select elements with an attribute value containing a specified word.
* The [attribute|="value"] selector is used to select elements with the specified attribute starting with the specified value.
* The [attribute^="value"] selector is used to select elements whose attribute value begins with a specified value.
* The [attribute$="value"] selector is used to select elements whose attribute value ends with a specified value.
* The [attribute\*="value"] selector is used to select elements whose attribute value contains a specified value.

**Program:**

**Simple Slector Demo:**

<html>

<head>

<style type="text/css">

p{

font-style:Tahoma;

font-size:40px;

border:5px double #ccc;

}

</style>

<body>

<p>This is a paragraph</p<br>

<p>All the paragraph's are displayed with the mentioned styles</p><br>

</body>

</html>

**Output:**



**Class Selctor Demo:**

<html>

<head>

<style type="text/css">

p{

font-style:Tahoma;

font-size:40px;

border:5px double #ccc;

}

p.redpara{

color:none;

background-color:red;

border:5px solid green;

}

</style>

<body>

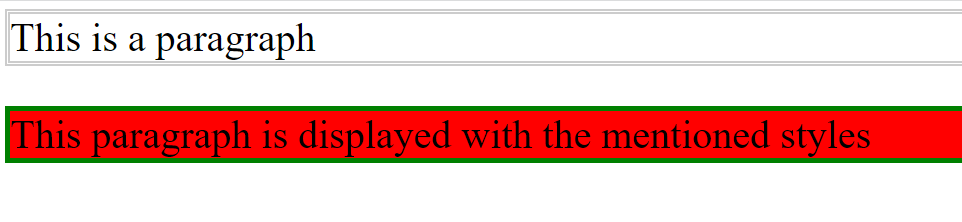
<p>This is a paragraph</p<br>

<p class="redpara">This paragraph is displayed with the mentioned styles</p><br>

</body>

</html>

**Output:**



**Id Selector Demo:**

<html>

<head>

<style type="text/css">

#red{

color:none;

background-color:red;

border:5px solid green;

}

</style>

<body>

<p id="red">This paragraph is displayed with the mentioned styles</p><br>

<div id="red">This division is displayed wit mentioned styles</div><br>

<section id="red">This section is dispalyed with the mentioned styles</section><br>

</body>

</html>

**Output:**



**Generic Slector Demo:**

<html>

<head>

<style type="text/css">

.red{

color:none;

background-color:red;

border:5px solid green;

}

</style>

<body>

<p class="red">This paragraph is displayed with the mentioned styles</p><br>

<div class="red">This division is displayed wit mentioned styles</div><br>

<section class="red">This section is dispalyed with the mentioned styles</section><br>

</body>

</html>

**Output:**



**Universal Slector Demo:**

<html>

<head>

<style type="text/css">

\*{

color:none;

background-color:red;

border:5px solid green;

}

</style>

<body>

<p >This paragraph is displayed with the mentioned styles</p><br>

<div >This division is displayed wit mentioned styles</div><br>

<section >This section is dispalyed with the mentioned styles</section><br>

</body>

</html>

**Output:**



**Attribute Selector Demo:**

<!DOCTYPE html>

<html>

<head>

<style>

input[type=text] {

width: 150px;

display: block;

margin-bottom: 10px;

background-color: yellow;

}

input[type=button] {

width: 120px;

margin-left: 35px;

display: block;

}

</style>

</head>

<body>

<form name="input" action="" method="get">

Firstname:<input type="text" name="Name" value="Peter" size="20">

Lastname:<input type="text" name="Name" value="Griffin" size="20">

<input type="button" value="Example Button">

</form>

</body>

</html>

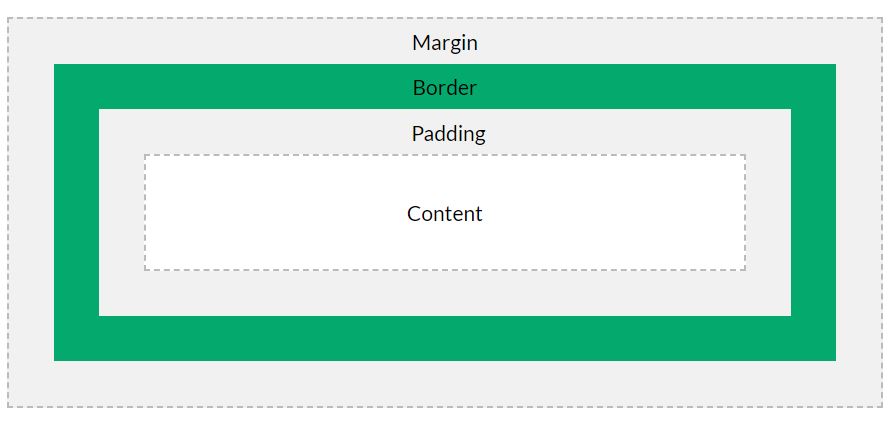
**Output:**



**3c. Write a program to apply Box model.**

**Description:**

* In CSS, the term "box model" is used when talking about design and layout.
* The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



* Explanation of the different parts:
* **Content** - The content of the box, where text and images appear
* **Padding** - Clears an area around the content. The padding is transparent
* **Border** - A border that goes around the padding and content
* **Margin** - Clears an area outside the border. The margin is transparent
* The box model allows us to add a border around elements, and to define space between elements.

**Program:**

<!DOCTYPE html>

<html>

<head>

<style>

div {

background-color: lightgrey;

width: 300px;

border: 15px solid green;

padding: 50px;

margin: 20px;

}

</style>

</head>

<body>

<h2>Demonstrating the Box Model</h2>

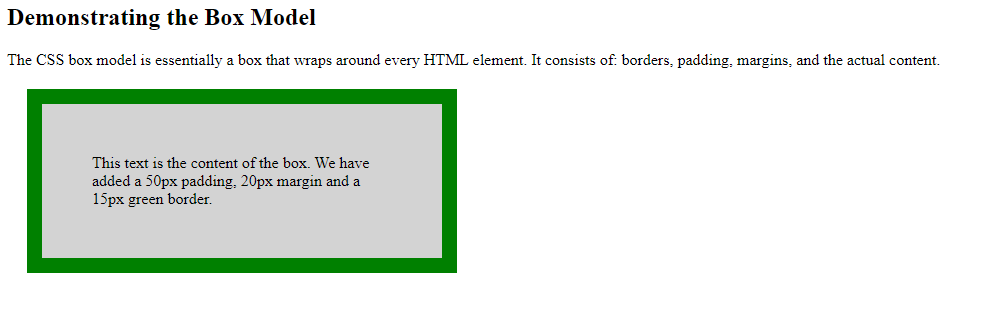
<p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.</p>

<div>This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. </div>

</body>

</html>

**Output:**



**3d. Write a CSS rule that places a background image halfway down the page, tilting it horizontally. The image should remain in place when the user scrolls up or down.**

**Program:**

<!DOCTYPE html>

<html>

<head>

<style>

img {

-webkit-box-reflect: right 20px;

}

</style>

</head>

<body>

<h1>CSS Image Reflection</h1>

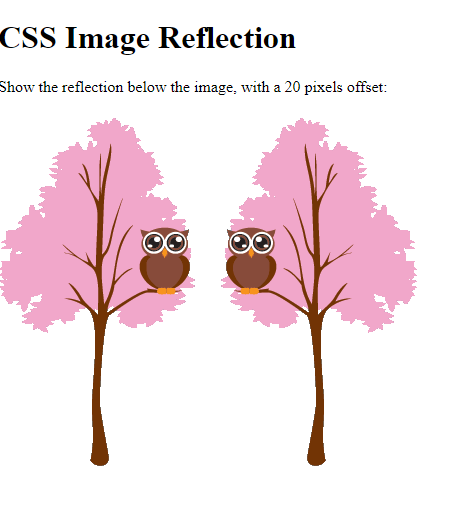
<p>Show the reflection below the image, with a 20 pixels offset:</p>

<img src="./images/img\_tree.gif">

</body>

</html>

**Output:**



**3e. Write a program using the following terms related to CSS:**

**i. font-size ii. font-weight iii. font-strech**

**iv. text-decoration v. text-transformation vi. text-alignment**

**Description:**

**Font Properties in CSS:**

* Using a font that is easy to read is important.
* The font adds value to your text. It is also important to choose the correct color and text size for the font.
* In CSS there are five generic font families:
* **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
* **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
* **Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
* **Cursive** fonts imitate human handwriting.
* **Fantasy** fonts are decorative/playful fonts.
* All the different font names belong to one of the generic font families
* In CSS, we use the font-family property to specify the font of a text.
* The font-family property should hold several font names as a "fallback" system, to ensure maximum compatibility between browsers/operating systems.
* Start with the font you want, and end with a generic family (to let the browser pick a similar font in the generic family, if no other fonts are available).
* The font names should be separated with comma.
* font-family: *family-name*|*generic-family*|initial|inherit;
* Font-family:Georgia|Palatino Linotype|Book Antiqua|Times New Roman|Arial|Helvetica|Arial Black|Impact|Lucida Sans Unicode|Tahoma|Verdana|Courier New|Lucida Console|initial
* In a small-caps font, all lowercase letters are converted to uppercase letters.
* However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.
* The font-variant property specifies whether or not a text should be displayed in a small-caps font.
* font-variant: normal|small-caps|initial|inherit;
* The font-variant-caps property controls the usage of alternate glyphs for capital letters.
* font-variant-caps: normal|small-caps|all-small-caps|petite-caps|all-petite-caps|unicase|titling-caps|initial|inherit|unset;
* The font-size property sets the size of a font.
* font-size:medium|xx-small|x-small|small|large|x-large|xx-large|smaller|larger|*length*|%|initial|inherit;
* The font-style property specifies the font style for a text.
* font-style: normal|italic|oblique|initial|inherit;
* The font-weight property sets how thick or thin characters in text should be displayed.
* font-weight: normal|bold|bolder|lighter|*number*|initial|inherit;

**Text Properties in CSS:**

* CSS has a lot of properties for formatting text
* The color property is used to set the color of the text. The color is specified by:
* a color name - like "red"
* a HEX value - like "#ff0000"
* an RGB value - like "rgb(255,0,0)"
* Look at CSS Color Values for a complete list of possible color values.
* The default text color for a page is defined in the body selector.
* Text-color/color:colorname/rgb/hex/hsl|initial|inherit;
* The text-align property is used to set the horizontal alignment of a text.
* A text can be left or right aligned, centered, or justified.
* When the text-align property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers)
* Text-align: left|right|center|justify|initial|inherit;
* The text-decoration property is used to set or remove decorations from text.
* The value text-decoration: none; is often used to remove underlines from links
* The other text-decoration values are used to decorate text
* Text-decoration: *text-decoration-line* *text-decoration-color* *text-decoration-style*|initial|inherit;
* The text-decoration-line property sets the kind of text decoration to use (like underline, overline, line-through).
* Text-decoration-line:none|underline|overline|line-through|initial|inherit;
* The text-decoration-style property sets the style of the text decoration (like solid, wavy, dotted, dashed, double).

text-decoration-style: solid|double|dotted|dashed|wavy|initial|inherit;

* The text-transform property is used to specify uppercase and lowercase letters in a text.
* It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word
* Text-transform:none|capitalize|uppercase|lowercase|initial|inherit;
* The text-indent property is used to specify the indentation of the first line of a text
* Text-indent:px|pts|cm|em|initial|inherit;
* The direction property specifies the text direction/writing direction within a block-level element.
* direction: ltr|rtl|initial|inherit;
* The text-overflow property specifies how overflowed content that is not displayed should be signaled to the user. It can be clipped, display an ellipsis (...), or display a custom string.
* Both of the following properties are required for text-overflow:
* text-overflow: clip|ellipsis|*string*|initial|inherit;
* The letter-spacing property is used to specify the space between the characters in a text.
* Letter spacing can be given either a positive or –ve value.
* Letter-spacing:+-normal|px|pts|cm|em|initial|inherit;
* The line-height property is used to specify the space between lines as floating point value
* Line-height:normal|value|length|initial|inherit;
* The word-spacing property is used to specify the space between the words in a text
* word-spacing: normal|px|pts|cm|em|inherit;
* The white-space property specifies how white-space inside an element is handled.
* White-space: normal|nowrap|pre|pre-line|pre-wrap|initial|inherit;
* The unicode-bidi property is used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document.
* unicode-bidi: normal|embed|bidi-override|initial|inherit;
* The vertical-align property sets the vertical alignment of an element.
* vertical-align: baseline|*length*|sub|super|top|text-top|middle|bottom|text-bottom|initial|inherit;
* The text-shadow property adds shadow to text.
* In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px)
* text-shadow: *h-shadow v-shadow blur-radius color*|none|initial|inherit;

**Program:**

<html>

<head>

<style>

.font{

font-family:Times;

font-style:oblique;

font-weight:900;

font-size:65%;

font-stretch:expanded;

border:2px solid;

}

.text{

text-align:center;

text-decoration:overline;

text-transform:capitalize;

border:2px solid;

}

</style>

</head>

<body>

<section class="font">

This text is displayed with applied font properties.

</section>

<div class="text">

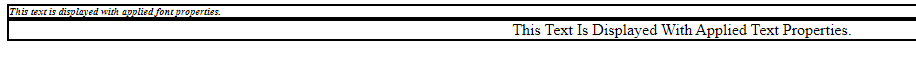
This text is displayed with applied text properties.

</div>

</body>

</html>

**Output:**



**4a Write a JavaScript program, that makes use of different objects (predefined – Array, String, Math, Date, RegExp and User-defined). Note: Use atleast 5 properties and methods from each object.**

**Description:**

* **JavaScript array** is an object that represents a collection of similar type of elements.
* There are 3 ways to construct array in JavaScript
* By array literal
* By creating instance of Array directly (using new keyword)
* By using an Array constructor (using new keyword)
* JavaScript array literal
* The syntax of creating array using array literal is given below:
* var arrayname=[value1,value2.....valueN];
* As you can see, values are contained inside [ ] and separated by , (comma).
* JavaScript Array directly (new keyword)
* The syntax of creating array directly is given below:
* var arrayname=new Array();
* Here, **new keyword** is used to create instance of array.
* JavaScript array constructor (new keyword)
* Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.
* The length Property
* The length property of an array returns the length of an array (the number of array elements).
* Syntax: arrayname.length
* Accessing the First and Last Array Elements
* Syntax: arrayname[0]//for first element
* Syntax:arrayname[arrsyname.length-1];//for last element
* We can use the Array.forEach() function to traverse array elements
* Methods in Math object are used for manipulation of numbers and to perform any common mathematical calculations.
* It contains many rounding methods like floor value, ceil value, round value and many trigonometric functions like sin, cos and tan functions and other functions like max, min etc…
* The web content is to be displayed on the web page in string form.
* Java script provides many string functions to process these string objects.
* A string is a collection of objects;these may include any kind of special characters, digits, normal characters.
* The String object is used to manipulate a stored piece of text.
* String manipulation can be done & generate HTML markup methods
* There are 2 ways to create string in JavaScript
* By string literal
* By string object (using new keyword)
* 1) By string literal
* The string literal is created using double quotes. The syntax of creating string using string literal is given below:
* var stringname="string value";
* By string object (using new keyword)
* The syntax of creating string object using new keyword is given below:
* var stringname=new String("string literal");
* Here new keyword is used to create instance of a string.
* The **JavaScript date** object can be used to get year, month and day.
* You can display a timer on the webpage by the help of JavaScript date object.
* You can use different Date constructors to create date object.
* It provides methods to get and set day, month, year, hour, minute and seconds.
* You can use 4 variant of Date constructor to create date object.
* Date()
* Date(milliseconds)
* Date(dateString)
* Date(year, month, day, hours, minutes, seconds, milliseconds)
* A regular expression is a sequence of characters that forms a **search pattern**.
* When you search for data in a text, you can use this search pattern to describe what you are searching for.
* A regular expression can be a single character, or a more complicated pattern.
* Regular expressions can be used to perform all types of **text search** and **text replace** operations.
* Syntax
* /*pattern*/*modifiers*;
* In JavaScript, regular expressions are often used with the two **string methods**: search() and replace().
* The search() method uses an expression to search for a match, and returns the position of the match.
* The replace() method returns a modified string where the pattern is replaced.
* The test() method is a RegExp expression method.
* It searches a string for a pattern, and returns true or false, depending on the result.
* The exec() method is a RegExp expression method.
* It searches a string for a specified pattern, and returns the found text as an object.
* If no match is found, it returns an empty *(null)* object.
* toString() returns the string representation of regular expression.

**Programs:**

**Arraydemo.html**

<html>

<head>

<script type="text/javascript">

var arr = [];

var size = window.prompt("Enter size of the array:");

for(var a=0; a<size; a++)

{

arr[a] = parseInt(prompt('Enter array Element ' + (a+1)));

}

document.writeln("Array generated is:"+arr);

arr.push(10);

document.writeln("Array after pushing is:"+arr);<br>

arr.pop();

document.writeln("Array after popping is:"+arr);

arr.shift();

document.writeln("Array after shifting is:"+arr);

delete arr[2];

document.writeln("Array after deleting is:"+arr);

arr.splice(3,0,3,4,5);

document.writeln("Array after splicing is:"+arr);

var a1=arr.slice(1,3);

document.writeln("Array created by splicing is:"+a1);

document.writeln("Length of the array is:"+arr.length);

Array.prototype.myprot = function() {

this.sort();

};

arr.myprot();

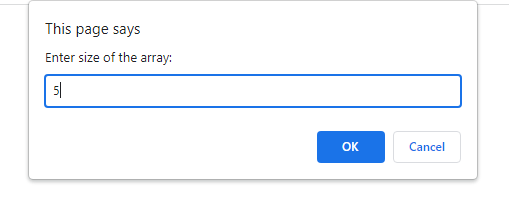
document.writeln("Array after sorting is:"+arr);

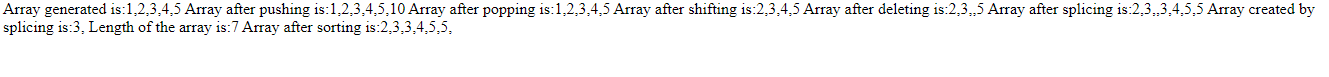
</script>

</head>

</html>

**Output:**





**Stringdemo.html:**

<html>

<head>

<script type="text/javascript">

var s="";

var s = window.prompt("Enter a string:");

document.writeln("String generated is:"+s);

document.writeln("Character at position 1 is:"+s.charAt(1));

document.writeln("Index of a is:"+s.indexOf('a'));

document.writeln("Pattern matched at:"+s.match("Java"));

var s1=s.slice(1,3);

document.writeln("String created by splicing is:"+s1);

var s2=s.substr(1,3);

document.writeln("String created by slicing is:"+s2);

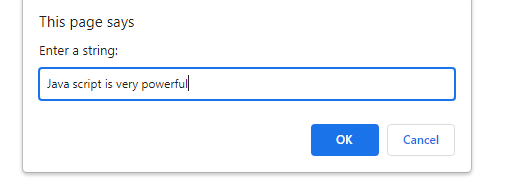
document.writeln("Length of the string is:"+s.length);

</script>

</head>

</html>

**Output:**





**Mathdemo.html:**

<html>

<head>

<script type="text/javascript">

var n = parseInt(window.prompt("Enter a number:"));

document.writeln("Ceil Value is:"+Math.ceil(n));

document.writeln("Floor value is:"+Math.floor(n));

var l=Math.random();

document.writeln("random value is:"+l);

document.writeln("sine value is:"+Math.sin(n));

document.writeln("Tan value is:"+Math.tan(n));

document.writeln(Log N base 2 value is"+Math.LN2);

document.writeln(Log N base 10 value is"+Math.LN10);

document.writeln(base 2 logarithm of E value is"+Math.LOG2E);

document.writeln(base 10 logarithm of E value is"+Math.LOG10E);

</script>

</head>

</html>

**Output:**



**Datedemo.html:**

<html>

<head>

<script type="text/javascript">

var dt = new Date( "April 11, 2022 08:30:00");

document.writeln(dt);

dt.setDate(10);

document.writeln("Changed date is:"+dt);

document.writeln("Hours are:"+dt.getHours());

document.writeln("Year is:"+dt.getFullYear());

document.writeln("Day is:"+dt.getDay());

</script>

</head>

</html>

**Output:**



**Regularexpressiondemo.html:**

<html>

<head>

<title>JavaScript RegExp exec Method</title>

</head>

<body>

<script type = "text/javascript">

var str = "Javascript is an interesting scripting language";

var re = new RegExp( "script", "g" );

var result1 = re.exec(str);

document.write("Test 1 - returned value : " + result1);

var re1 = new RegExp( "script", "g" );

var result2 = re1.test(str);

document.write("Test 2 - returned value : " + result2);

var re2 = new RegExp( "script", "g" );

var patt = new RegExp("Hello World", "g");

var res = patt.toString();

document.write("Test 3 - returned value : " + res);

</script>

</body>

</html>

**Output:**



**4b Write a javascript to display the denomination of the amount deposited in the bank in terms of 100’s, 50’s, 20’s, 10’s, 5’s, 2’s & 1’s. (Eg: If deposited amount is Rs.163, the output should be 1-100’s, 1-50’s, 1- 10’s, 1-2’s & 1- 1’s)**

**Program:**

<html>

<head><title>Display the Denomination</title></head>

<body><script>

a=prompt("Enter number"," ");

n=parseInt(a);

h=Math.floor(n/100);

n=n%100;

f=Math.floor(n/50);

n=n%50;

tw=Math.floor(n/20);

n=n%20;

t=Math.floor(n/10);

n=n%10;

fi=Math.floor(n/5);

n=n%5;

two=Math.floor(n/2);

n=n%2;

one=Math.floor(n/1);

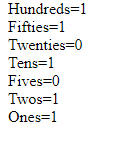
document.write("Hundreds="+h+"<br>Fifties="+f+"<br>Twenties="+tw+"<br>Tens="+t+"<br>Fives="+fi+"<br>Twos="+two+"<br>Ones="+one);

</script>

</body></html>

**Output:**





**4c Write a JavaScript that takes a number from one text field in the range of 0-999 and display it in other text field in words. If the number is out of range, it should show “out of range” and if it is not a number, it should show “not a number” message in the result box.**

**Program:**

<html>

<head>

<title>Number in words</title>

<script language="javascript">

function convert()

{

var num=document.forms["frm1"].num.value;

document.forms["frm1"].words.value="";

if(isNaN(num))

{

alert("Not a Number");

}

else if (num<0 || num>999)

{

alert("Out of Range");

}

else

{

var len=num.length;

var words="";

for(var i=0;i<len;i++)

{

var n=num.substr(i,1);

switch(n)

{

case '0':words+="Zero ";break;

case '1':words+="One ";break;

case '2':words+="Two ";break;

case '3':words+="Three ";break;

case '4':words+="Four ";break;

case '5':words+="Five ";break;

case '6':words+="Six ";break;

case '7':words+="Seven ";break;

case '8':words+="Eight ";break;

default:words+="Nine ";

}

}

document.forms["frm1"].words.value=words;

}

}

</script>

</head>

<body>

<form name="frm1">

<center><h3>NUMBER IN WORDS</h3></center>

<br/>

<center>Enter a Number :<input type="text" name="num"</input><br/></center>

<br/>

<center><input type="button" name="inwords" value="In Words" onclick="convert()"></input></center>

<br/><br/><center>Number in Words :<input type="text" name="words"</input></center>

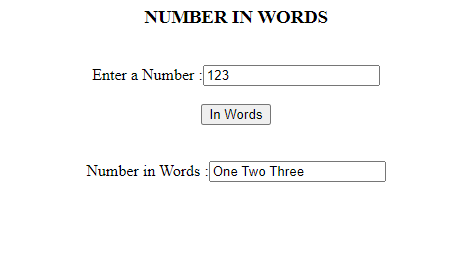
<br/>

</form>

</body>

</html>

**Output:**



**4d Write a JavaScript program, that makes use of class, object, method, constructor for student information system.**

**Program:**

<html>

<meta charset="utf-8"/>

<body> Demonstrating creation of objects with the help of class</br>

<script type="text/javascript">

class Student{

name;

getStudentName(){

document.write(this.name);

}

}

const student1 = new Student();

student1.name="Arjun";

const student2 = new Student();

student2.name="Jagadeesh";

const student3 = new Student();

student3.name="Lakshay";

student1.getStudentName();

student2.getStudentName();

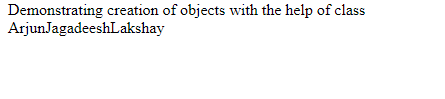
student3.getStudentName();

</script>

</body>

</html>

**Output:**



**5a Design a HTML having a text box and four buttons named Factorial, Fibonacci, Prime, and Palindrome. When a button is pressed an appropriate javascript function should be called to display**

1. **Factorial of that number**
2. **ii. Fibonacci series up to that number**
3. **iii. Prime numbers up to that number**
4. **iv. Is it palindrome or not**

**Program:**

<html>

<head>

<title>Javascript Demo</title>

<script type="text/javascript">

function fact(){

var n=parseInt(document.getElementById("n").value);

var fact=1;

if(n==0){

fact=1;

}

else{

for(i=1;i<=n;i++){

fact\*=i;

}

}

document.write("Factorial value is "+fact);

}

function fib(){

var n=parseInt(document.getElementById("n").value);

var n1 = 0, n2 = 1, next\_num, i;

document.write( "Fibonacci Series: ");

for ( i = 1; i <= n; i++)

{ document.write (" <br> " + n1);

next\_num = n1 + n2;

n1 = n2;

n2 = next\_num;

}

}

function prime(){

var n=parseInt(document.getElementById("n").value);

var store = [], i, j, primes = [];

for (i = 2; i <= n; ++i)

{

if (!store [i])

{

primes.push(i);

for (j = i << 1; j <= n; j += i)

{

store[j] = true;

}

}

}

document.write(primes);

}

function pal(){

var n=parseInt(document.getElementById("n").value);

var rem, temp, final = 0;

temp = n;

while(n>0)

{

rem = n%10;

n = parseInt(n/10);

final = final\*10+rem;

}

if(final==temp)

{

document.write("The inputed number is Palindrome");

}

else

{

document.write("The inputted number is not palindrome");

}

}

</script>

</head>

<body>

<table align="center">

<caption="Formdemo">

<form name="f1" method="post">

<tr>

<td>Enter a number: </td>

<td><input type="text" id="n" name="num" size="10" maxlength="15" required></td>

</tr>

<tr>

<td colspan=4><input type="submit" value="factorial" onclick="fact()">

<input type="submit" value="fib series" onclick="fib()">

<input type="submit" value="prime series" onclick="prime()">

<input type="submit" value="palindrome" onclick="pal()"></td>

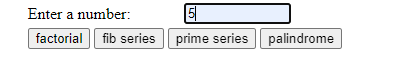
</tr>

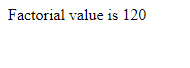
<tr>

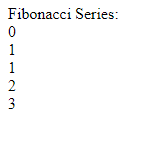
</body>

</html>

**Output:**













**5b Write JavaScript programs on Event Handling**

1. **Open a Window from the current window**
2. **Change color of background at each click of button or refresh of a page**
3. **Display calendar for the month and year selected from combo box**
4. **On Mouse over event**

**Description:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Events** | **Attribute** | **Meaning** | **Associated Tags** |
| blur | Onblur | Losing focus | <button> , <input>, <textarea>,<a> |
| change | Onchange | On occurrence of  Change | <input> , <textarea>,<select> |
| click | Onclick | When user clicks mouse button | <input> .<a> |
| dbclick | Ondbclick | When user double clicks mouse button | <input> .<a>,  <button> |
| focus | Onfocus | When user acquires the  input focus | <a>, <input> , <textarea>,<select> |
| keyup | Onkeyup | When user releases the  key from keyboard | Form elements |
| keydown | Onkeydown | When user presses key  Down | Form elements |
| keypress | Onkeypress | When user presses key | Form elements |
| mousedown | Onmousedown | When user clicks left  mouse button | Form elements |
| mouseup | Onmouseup | When user releases left  mouse button | Form elements |
| mousemove | Onmousemove | When user moves mouse | Form elements |
| mouseout | Onmouseout | User moves mouse away  from some element | Form elements |
| mouseover | Onmouseover | User moves mouse away over some element | Form elements |
| load | Onload | After getting document  Loaded | <body> |
| reset | Onreset | When reset button clicked | <form> |
| submit | Onsubmit | Submit button clicked | <form> |
| select | Onselect | On selection | <input>, <textarea> |
| unload | Onunload | User exits the document | <body> |

**Programs:**

**Openwindow.html:**

<html>

<body>

<script>

function openWindow() {

window.open('https://www.aec.edu.in');

}

</script>

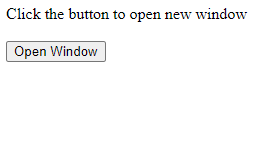
Click the button to open new window <br><br>

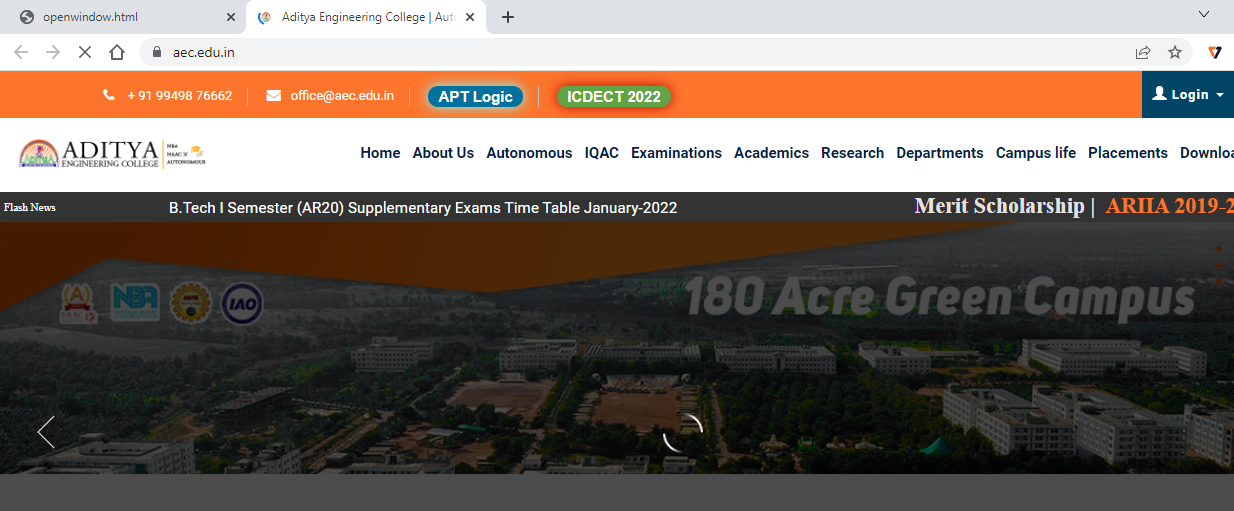
<button onclick="openWindow()"> Open Window </button>

</body>

</html>

**Output:**





**Onclickbutton.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<title>JavaScript change background color</title>

<script>

function changeBackgroundRed() {

document.body.style.background = "red";

}

</script>

</head>

<body>

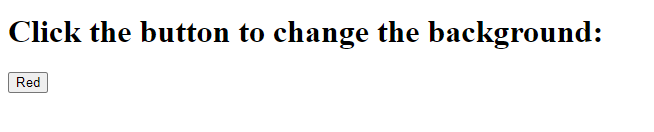
<h1>Click the button to change the background:</h1>

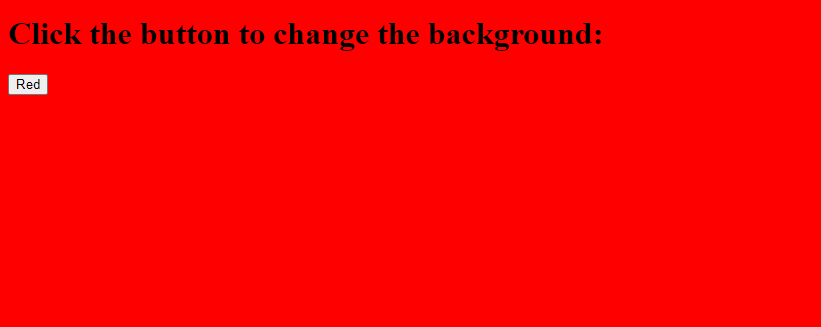
<button onclick="changeBackgroundRed();">Red</button>

</body>

</html>

**Output:**





**Onrefreshdemo.html:**

<html>

<head>

<script type="text/javascript">

var color = new Array();

color[0] = "#CC99FF";

color[1] = "#FF99CC";

color[2] = "#FF9999";

color[3] = "#FFCC99";

color[4] = "#FFFF99";

color[5] = "#CCFF99";

color[6] = "#99FF99";

color[7] = "#99FFCC";

color[8] = "#66FFFF";

color[9] = "#66CCFF";

function changeColor()

{

var randomColor = Math.floor(Math.random() \* color.length);

document.body.style.backgroundColor = color[randomColor];

}

</script>

</head>

<body onload="changeColor()">

</body>

</html>

**Output:**



**Calendardemo.html:**

<!DOCTYPE HTML>

<html>

<head>

  <style>

    table {

      border-collapse: collapse;

    }

    td,

    th {

      border: 1px solid black;

      padding: 3px;

      text-align: center;

    }

    th {

      font-weight: bold;

      background-color: #E6E6E6;

    }

  </style>

</head>

<body>

  <div id="calendar"></div>

  <script>

    function createCalendar(elem, year, month) {

      let mon = month - 1; // months in JS are 0..11, not 1..12

      let d = new Date(year, mon);

      let table = '<table><tr><th>MO</th><th>TU</th><th>WE</th><th>TH</th><th>FR</th><th>SA</th><th>SU</th></tr><tr>';

      for (let i = 0; i < getDay(d); i++) {

        table += '<td></td>';

      }

      while (d.getMonth() == mon) {

        table += '<td>' + d.getDate() + '</td>';

        if (getDay(d) % 7 == 6) {

          table += '</tr><tr>';

        }

        d.setDate(d.getDate() + 1);

      }

      if (getDay(d) != 0) {

        for (let i = getDay(d); i < 7; i++) {

          table += '<td></td>';

        }

      }

      table += '</tr></table>';

      elem.innerHTML = table;

    }

    function getDay(date) {

      let day = date.getDay();

      if (day == 0) day = 7;

      return day - 1;

    }

var year=parseInt(prompt("Enter Year:"));

var month=parseInt(prompt("Enter Month:"));

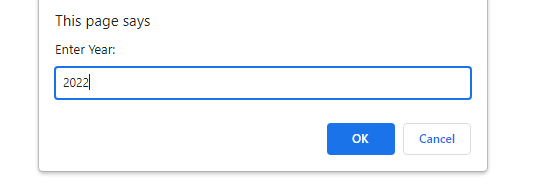
createCalendar(calendar, year, month);

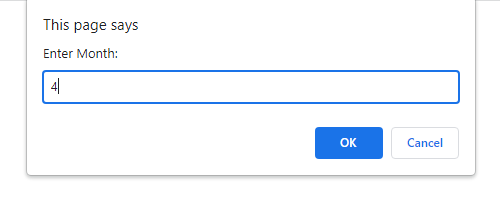
</script>

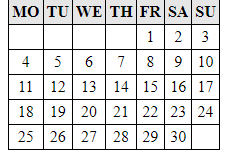
</body>

</html>

**Output:**







**Onmouseoverdemo.html:**

**Program:**

<!DOCTYPE html>

<html>

<body>

<h1 id="demo">Mouse over me</h1>

<script>

document.getElementById("demo").addEventListener("mouseover", mouseOver);

document.getElementById("demo").addEventListener("mouseout", mouseOut);

function mouseOver() {

document.getElementById("demo").style.color = "red";

}

function mouseOut() {

document.getElementById("demo").style.color = "black";

}

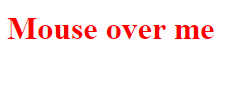
</script>

</body>

</html>

**Output:**





**5c Write a JavaScript to validate the following fields in a registration page created in Experiment 2**

1. **Name (start with alphabet and followed by alphanumeric and the length should not be less than 6 characters)**
2. **Password (it allows alphanumeric, special symbols and should not be less than 6 characters)**
3. **E-mail (should not contain invalid email addresses)**

**Program:**

<html>

<head>

<title>Formdemo</title>

<script type="text/javascript">

function validate(){

var name=document.getElementById("un").value;

var pass=document.getElementById("pwd").value;

var email=document.getElementById("em").value;

var re = /^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$/;

if(name.test(/^[a-zA-Z0-9]/)==false){

window.alert("Please enter a valid name");

}

else if(name.length<6){

window.alert("Username must be atleast 6 characters long");

}

else if(pass.test(/^(?=.\*\d)(?=.\*[a-z])(?.=\*[A-Z])(?.=\*[a-zA-Z0-9])(?.\*\s).{6,20}$/)==false){

window.alert("Password should have 1 lowercase, 1 uppercase, 1 digit, 1 special character and it should be greater than 6 characters");

}

else if (re.test(email)==false) {

alert("Please enter a valid email");

}

}

</script>

</head>

<body>

<table align="center">

<caption="Formdemo">

<form name="f1" method="post">

<tr>

<td>Username: </td>

<td><input type="text" id="un" name="uname" size="10" maxlength="15" required></td>

</tr>

<tr>

<td>Password:</td>

<td><input type="password" id="pwd" name="pwd" size="10" maxsize="15" required></td>

</tr>

<tr>

<td>Email </td>

<td><input type="text" id="em" name="uname" required></td>

</tr>

<tr>

<td>Address:</td>

<td><textarea name="ta" rows="5" cols="10"></textarea></td>

</tr>

<tr>

<td>Languages known:</td>

<td><input type="checkbox" name="c1" value="Te" checked>Telugu</br>

<input type="checkbox" name="c2" value="Ta">Tamil</br>

<input type="checkbox" name="c3" value="Hi">Hindi</br>

<input type="checkbox" name="c4" value="En">English</br>

</td>

</tr>

<tr>

<td>Programming Proficiency:</td>

<td>

<input type="radio" name="r1" value="C">C</br>

<input type="radio" name="r1" value="Java">Java</br>

<input type="radio" name="r1" value="Python" checked>Python</br>

<input type="radio" name="r1" value="PHP">PHP</br>

</td>

</tr>

<tr>

<td>Willing to work at:</td>

<td>

<select name="s1" multiple>

<option value="Mu">Mumbai</option>

<option value="ch" label="Chennai">Chennai</option>

<option value="Hy">Hyderbad</option>

<option value="viz" selected>Vizag</option>

</select>

</td>

</tr>

<tr>

<td colspan="2"><input type="submit" value="submit" onclick="validate()">

<input type="reset" value="reset">

</td>

</tr>

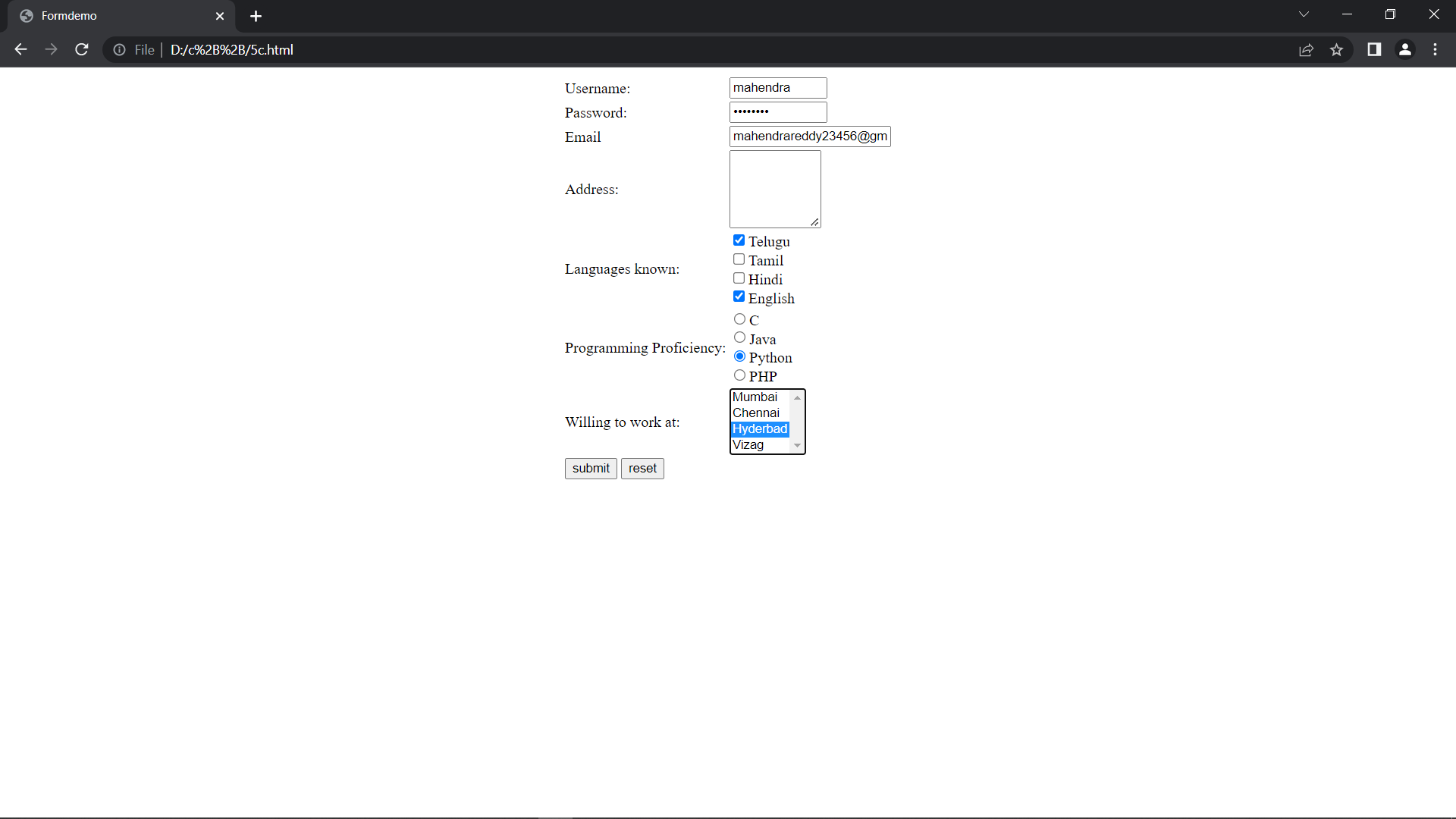
</form>

</table>

</body>

</html>

**Output:**



**Week 6:**

**Write an XML file which will display the Book information which includes the following: Title of the book, Author Name, ISBN number, Publisher name, Edition, Price.**

1. **Write a Document Type Definition (DTD) to validate the above XML file.**
2. **Write a XML Schema Definition (XSD) to validate the above XML file.**
3. **Write a Extensible Stylesheet Language Transformations (XSL) to display data in table format**

**Description:**

**DTD:**

* The document type definition used to define the basic building block of any xml
* document.
* Using DTD we can specify the various elements types, attributes and their relationship with one another.
* Basically DTD is used to specify the set of rules for structuring data in any XML file.
* ***Various building blocks of XML are***
* 1. Elements
* 2. Attribute
* 3. CDATA
* 4. PCDATA
* CDATA also means character data.
* CDATA is text that will not be parsed by a parser.
* Tags inside the text will not be treated as markup and entities will not be expanded.
* PCDATA means parsed character data.
* Character data is e the text found between the start tag and the of an XML document.
* PCDATA is text that will be parse by a parser Tags inside the text will be treated as markup and entities will be expanded.
* <!ELEMENT student (name)\*>
* \* indicates zero or more occurrences.
* + indicates 1,2,…n
* ? indicate 0,1
* Default exactly once

**XSD:**

* The <schema> element is the root element of every XML schema.
* <schema>elementmay contains some attributes.
* <?xml version = "1.0"?>
* <xs:schema>
* --------------
* </xs:schema>
* XSD elements are two types.
* 1. Simple element
* 2. Complex element
* Various data types are String, Date, Numeric, Boolean
* xs:string It containing group of characters, lines, tabs or white spaces.
* xs:date Used to represent date. The format of this date is yyyy-mm-dd
* xs:time use to represent time. The format of this time is hh:mm:ss
* xs:decimal use to represent float values.
* xs:integer use to represent integer values.
* xs:boolean used to resent Boolean values either true or false.

**XSLT:**

* XSLT is a language capable of transforming as well as formatting given XML document into required format such as html and pdf files
* XSL means eXtensible Stylesheet Language
* Start with a raw XML document (create a xml file)
* Create an XSL style sheet (create a xsl file)
* Link the XSL style sheet to the XML document
* Use XSLT processors to transform your XML into XHTML

**Programs:**

**book.dtd:**

<!ELEMENTbookdetails (books+)>

<!ELEMENT books (title,author,isbn,publisher,edition,price)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT author (#PCDATA)>

<!ELEMENTisbn (#PCDATA)

<!ELEMENT publisher (#PCDATA)

<!ELEMENT edition (#PCDATA)>

<!ELEMENT price (#PCDATA)>

**book.xml:**

<?xml version="1.0"?>

<!DOCTYPEbookdetails SYSTEM "book.dtd">

<?xml:stylesheet type="text/xsl" href="book.xsl"?>

<bookdetails>

<books>

<title> C </title>

<author>Balaguruswamy</author>

<isbn> 123456</isbn>

<publisher>Oxford</publisher>

<edition> Edition-II </edition>

<price>$30.00</price>

</books>

<books>

<title> C++ </title>

<author>yaswanthkanethkar</author>

<isbn>654321</isbn>

<publisher>Tata McHill</publisher>

<edition> Edition-I </edition>

<price>$35.00</price>

</books>

<books>

<title> JAVA </title>

<author> Herbert Schildt</author>

<isbn>789456</isbn>

<publisher>Cenage</publisher>

<edition> Edition-IV </edition>

<price>$50.00</price>

</books></bookdetails>

**book.xsd:**

<?xml version="1.0"?>

<xs:schema>

xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="bookdetails">

<xs:complexType>

<xs:sequence>

<xs:element name="books” type="xs:string"/>

<xs:element name="title" type="xs:string"/>

<xs:element name="author" type="xs:string"/>

<xs:element name="isbn" type="xs:string"/>

<xs:element name="publisher" type="xs:string"/>

<xs:element name="edition" type="xs:string"/>

<xs:element name="price" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

**book.xml:**

<?xml version="1.0"?>

<!DOCTYPEbookdetails SYSTEM "book.dtd">

<?xml:stylesheet type="text/xsl" href="book.xsl"?>

<bookdetails>

<books>

<title> C </title>

<author>Balaguruswamy</author>

<isbn> 123456</isbn>

<publisher>Oxford</publisher>

<edition> Edition-II </edition>

<price>$30.00</price>

</books>

<books>

<title> C++ </title>

<author>yaswanthkanethkar</author>

<isbn>654321</isbn>

<publisher>Tata McHill</publisher>

<edition> Edition-I </edition>

<price>$35.00</price>

</books>

<books>

<title> JAVA </title>

<author> Herbert Schildt</author>

<isbn>789456</isbn>

<publisher>Cenage</publisher>

<edition> Edition-IV </edition>

<price>$50.00</price>

</books></bookdetails>

**book.xsl:**

<?xml version="1.0"?>

<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">

<html>

<head><center>Book Details</center></head>

<body>

<hr width="50%"/>

<table border="1" align="center">

<tr>

<th> TITLE </th>

<th> AUTHOR </th>

<th>ISBN</th>

<th>PUBLISHER</th>

<th> EDITON </th>

<th> PRICE </th>

</tr>

<xsl:for-each select="bookdetails/books">

<tr>

<td><xsl:value-of select="title"/></td>

<td><xsl:value-of select="author"/></td>

<td><xsl:value-of select="isbn"/></td>

<td><xsl:value-of select="publisher"/></td>

<td><xsl:value-of select="edition"/></td>

<td><xsl:value-of select="price"/></td>

</tr>

</xsl:for-each>

</table>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

**Output:**



**Week 7:**

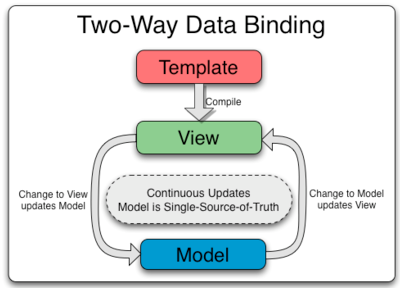
**Implement the following in Angular JS**

1. **data binding.**
2. **directives and Events.**
3. **Includes.**

**Description:**

**Data Binding:**

* Data binding is a very useful and powerful feature used in software development technologies.
* It acts as a bridge between the view and business logic of the application.
* AngularJS follows Two-Way data binding model.
* Data-binding in Angular apps is the automatic synchronization of data between the model and view components.
* Data binding lets you treat the model as the single-source-of-truth in your application.
* The view is a projection of the model at all times.
* If the model is changed, the view reflects the change and vice versa.



**Program:**

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

<div ng-app="myApp" ng-controller="myCtrl">

Name: <input ng-model="name">

<h1>You entered: {{name}}</h1>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.name = "Jagadeesh";

});

</script>

</body>

</html>

**Output:**



**Directives:**

* AngularJS directives are extended HTML attributes with the prefix ng-.
* The ng-app directive initializes an AngularJS application.
* The ng-init directive initializes application data.
* The ng-model directive binds the value of HTML controls (input, select,textarea) to application data.

**Program:**

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

<div data-ng-app="" data-ng-init="quantity=1;price=5">

<h2>Cost Calculator</h2>

Quantity: <input type="number" ng-model="quantity">

Price: <input type="number" ng-model="price">

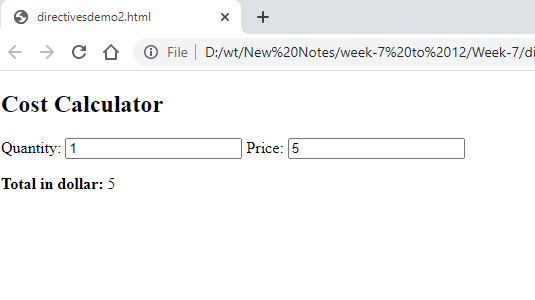
<p><b>Total in dollar:</b> {{quantity \* price}}</p>

</div>

</body>

</html>

**Output:**



**Events:**

You can add AngularJS event listeners to your HTML elements by using one or more of these directives:

* ng-blur
* ng-change
* ng-click
* ng-copy
* ng-cut
* ng-dblclick
* ng-focus
* ng-keydown
* ng-keypress
* ng-keyup
* ng-mousedown
* ng-mouseenter
* ng-mouseleave
* ng-mousemove
* ng-mouseover
* ng-mouseup
* ng-paste

The event directives allows us to run AngularJS functions at certain user events.

An AngularJS event will not overwrite an HTML event, both events will be executed.

## $event Object

You can pass the $event object as an argument when calling the function.

The $event object contains the browser's event object

**Program:**

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

<div ng-app="myApp" ng-controller="myCtrl">

<h1 ng-mousemove="myFunc($event)">Mouse Over Me!</h1>

<p>Coordinates: {{x + ', ' + y}}</p>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.myFunc = function(myE) {

$scope.x = myE.clientX;

$scope.y = myE.clientY;

}

});

</script>

</body>

</html>

**Output:**



**Include:**

With AngularJS, you can include HTML content using the **ng-include** directive

The HTML files you include with the ng-include directive, can also contain AngularJS code

**Program:**

**Main.html:**

<html>

<head></head>

<body>

<table border = "0">

<tr>

<td>Enter first name:</td>

<td><input type = "text" ng-model = "student.firstName"></td>

</tr>

<tr>

<td>Enter last name: </td>

<td><input type = "text" ng-model = "student.lastName"></td>

</tr>

<tr>

<td>Name: </td>

<td>{{student.fullName()}}</td>

</tr>

</table>

</body>

</html>

**Subjects.html:**

<html>

<head></head>

<body>

<p>Subjects:</p>

<table>

<tr>

<th>Name</th>

<th>Marks</th>

</tr>

<tr ng-repeat = "subject in student.subjects">

<td>{{ subject.name }}</td>

<td>{{ subject.marks }}</td>

</tr>

</table>

</body>

</html>

**Includesdemo.html:**

<html>

<head>

<title>Angular JS Includes</title>

<script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">

</script>

<style>

table, th , td {

border: 1px solid grey;

border-collapse: collapse;

padding: 5px;

}

tabletr:nth-child(odd) {

background-color: #f2f2f2;

}

tabletr:nth-child(even) {

background-color: #ffffff;

}

</style>

</head>

<body>

<h2>AngularJS Sample Application</h2>

<div ng-app = "mainApp" ng-controller = "studentController">

<div ng-include = "'/includedemo/main.html'"></div>

<div ng-include = "'/includedemo/subjects.html'"></div>

</div>

<script>

varmainApp = angular.module("mainApp", []);

mainApp.controller('studentController', function($scope) {

$scope.student = {

firstName: "Rakesh",

lastName: "Siddanathi",

fees:50000,

subjects:[

{name:'Physics',marks:70},

{name:'Chemistry',marks:80},

{name:'Math',marks:65},

{name:'English',marks:75},

{name:'Hindi',marks:67}

],

fullName: function() {

varstudentObject;

studentObject = $scope.student;

returnstudentObject.firstName + " " + studentObject.lastName;

}

};

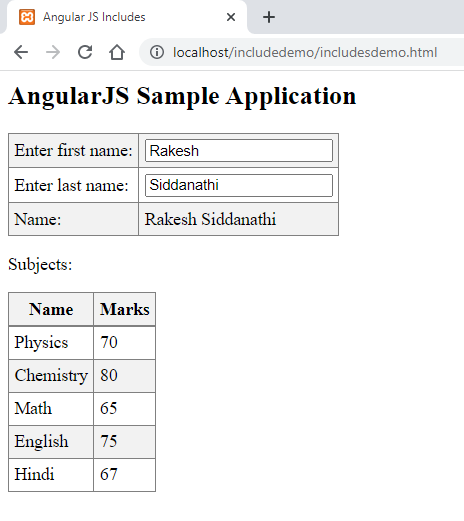
});

</script>

</body>

</html>

**Output:**



**Week-8:**

**Implement the following in Angular JS**

1. **routing.**
2. **form validation.**
3. **fetching data from MySQL.**

**Description:**

**Routing:**

If you want to navigate to different pages in your application, but you also want the application to be a SPA (Single Page Application), with no page reloading, you can use the ngRoute module.

The ngRoute module *routes* your application to different pages without reloading the entire application.

Use the $routeProvider to configure different routes in your application

We can also use the otherwise method, which is the default route when none of the others get a match.

**Program:**

**About.html:**

<html>

<head>

</head>

<body>

<p>Click on the below link o redirect to College info</p><br>

<a href="https://aec.edu.in/?p=About-AEC">About AEC</a>

</body>

</html>

**Aec.html:**

<html>

<head>

</head>

<body>

<p>Click on the below link to redirect to home page</p><br>

<ahref="https://www.aec.edu.in">AEC</a>

</body>

</html>

**Library.html:**

<html>

<head>

</head>

<body>

<p>Click on the below link to redirect to library info</p><br>

<ahref="https://aec.edu.in/?p=Library">AEC-Library</a>

</body>

</html>

**Routingdemo.html:**

<!DOCTYPE html>

<html>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular-route.js"></script>

<body ng-app="myApp">

<a href="#!about-aec">ABOUT AEC</a>

<a href="#!aec-lib">AEC LIBRARY</a>

<div ng-view></div>

<script>

var app = angular.module("myApp", ["ngRoute"]);

app.config(function($routeProvider) {

$routeProvider

.when("/about-aec", {

templateUrl : "about.html"

})

.when("/aec-lib", {

templateUrl : "library.html"

})

.otherwise( {

templateUrl : "aec.html"

});

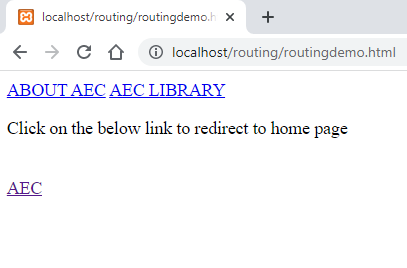
});

</script>

</body>

</html>

**Output:**



**Form Validation:**

AngularJS offers client-side form validation.

AngularJS monitors the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.

AngularJS also holds information about whether they have been touched, or modified, or not.

You can use standard HTML5 attributes to validate input, or you can make your own validation functions.

AngularJS is constantly updating the state of both the form and the input fields.

Input fields have the following states:

* $untouched The field has not been touched yet
* $touched The field has been touched
* $pristine The field has not been modified yet
* $dirty The field has been modified
* $invalid The field content is not valid
* $valid The field content is valid

They are all properties of the input field, and are either true or false.

Forms have the following states:

* $pristine No fields have been modified yet
* $dirty One or more have been modified
* $invalid The form content is not valid
* $valid The form content is valid
* $submitted The form is submitted

They are all properties of the form, and are either true or false.

**Program:**

<html>

<head>

<title>Angular JS Forms</title>

<script src = "https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>

<style>

table, th , td {

border: 1px solid grey;

border-collapse: collapse;

padding: 5px;

}

tabletr:nth-child(odd) {

background-color: #f2f2f2;

}

tabletr:nth-child(even) {

background-color: #ffffff;

}

</style>

</head>

<body>

<h2>AngularJS Sample Application</h2>

<div ng-app = "mainApp" ng-controller = "studentController">

<form name = "studentForm" novalidate>

<table border = "0">

<tr>

<td>Enter first name:</td>

<td><input name = "firstname" type = "text" ng-model = "firstName" required>

<span style = "color:red" ng-show = "studentForm.firstname.$dirty && studentForm.firstname.$invalid">

<span ng-show = "studentForm.firstname.$error.required">First Name is required.</span>

</span>

</td>

</tr>

<tr>

<td>Enter last name: </td>

<td><input name = "lastname" type = "text" ng-model = "lastName" required>

<span style = "color:red" ng-show = "studentForm.lastname.$dirty && studentForm.lastname.$invalid">

<span ng-show = "studentForm.lastname.$error.required">Last Name is required.</span>

</span>

</td>

</tr>

<tr>

<td>Email: </td><td><input name = "email" type = "email" ng-model = "email" length = "100" required>

<span style = "color:red" ng-show = "studentForm.email.$dirty&&studentForm.email.$invalid">

<span ng-show = "studentForm.email.$error.required">Email is required.</span>

<span ng-show = "studentForm.email.$error.email">Invalid email address.</span>

</span>

</td>

</tr>

<tr>

<td>

<button ng-click = "reset()">Reset</button>

</td>

<td>

<button ng-disabled = "studentForm.firstname.$dirty &&

studentForm.firstname.$invalid || studentForm.lastname.$dirty &&

studentForm.lastname.$invalid || studentForm.email.$dirty&&

studentForm.email.$invalid" ng-click="submit()">Submit</button>

</td>

</tr>

</table>

</form>

</div>

<script>

varmainApp =angular.module("mainApp", []);

mainApp.controller('studentController', function($scope) {

$scope.reset = function() {

$scope.firstName = "Jagadeesh";

$scope.lastName = "Siddanathi";

$scope.email = "jagadeesh.siddanathi@aec.edu.in";

}

$scope.reset();

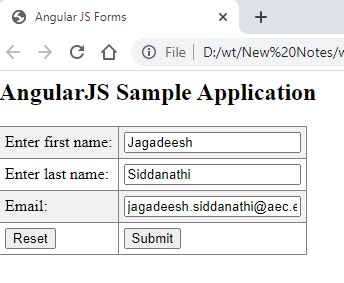
});

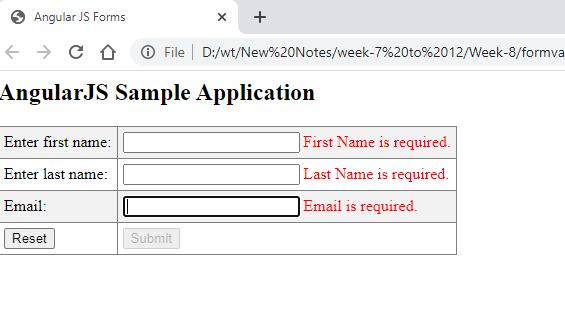
</script>

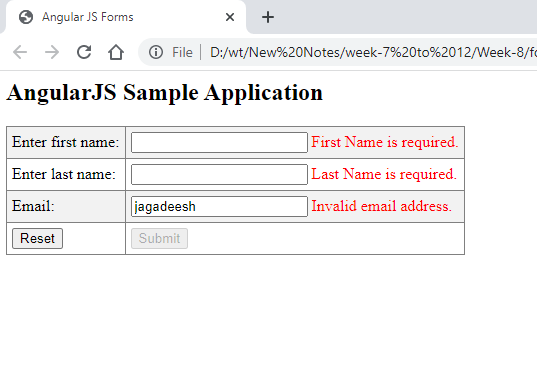
</body>

</html>

**Output:**







**Fetching Data from MySQL:**

**Program:**

**Data.html:**

<!DOCTYPE html>

<html>

<style>

table, th , td {

border: 1px solid grey;

border-collapse: collapse;

padding: 5px;

}

tabletr:nth-child(odd) {

background-color: #f1f1f1;

}

tabletr:nth-child(even) {

background-color: #ffffff;

}

</style>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

<body>

<div ng-app="myApp" ng-controller="customersCtrl">

<table>

<th>Name</th>

<th>Email</th>

<tr ng-repeat="x in users">

<td>{{ x.Name }}</td>

<td>{{ x.Email }}</td>

</tr>

</table>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('customersCtrl', ['$scope', '$http', function ($scope, $http) {

$http({

method: 'get',

url: 'connect.php'

}).then(function successCallback(response) {

// Store response data

$scope.users = response.data.records;

});

}]);

</script>

</body>

</html>

**Connect.php:**

<?php

header("Access-Control-Allow-Origin: \*");

header("Content-Type: application/json; charset=UTF-8");

$conn = new mysqli("localhost", "root", "", "aditya");

$result = $conn->query("SELECT uname,email FROM registration");

$outp = "";

while($rs = $result->fetch\_array(MYSQLI\_ASSOC)) {

if ($outp != "") {$outp .= ",";}

$outp .= '{"Name":"' . $rs["uname"] . '",';

$outp .= '"Email":"' . $rs["email"] . '"}';

}

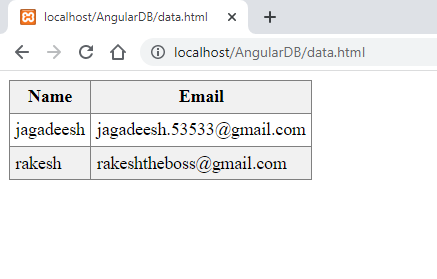
$outp ='{"records":['.$outp.']}';

$conn->close();

echo($outp);

?>

**Output:**



**Week 9:**

**Implement the following in React JS**

1. **rendering elements.**
2. **simple component.**
3. **stateful component.**

**Description:**

React element is the smallest renderable unit available in React. We can render such elements using the ReactDOM. React elements are different from DOM elements as React elements are simple javascript objects and are efficient to create. React elements are the building blocks of any React app and should not be confused with React components

**Rendering an Element in React:** In order to render any element into the Browser DOM, we need to have a container or root DOM element. It is almost a convention to have a div element with the id=”root” or id=”app” to be used as the root DOM element. Let’s suppose our index.html file has the following statement inside it.

<div id="root"></div>

**Program :**

App.js :

import React,{ Component } from 'react';

class App extends Component {

render() {

return (

<div>

<h1>Hello World!</h1>

</div>

);

}

}

export default App;

Index.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<meta name="theme-color" content="#000000" />

<meta

name="description"

content="Web site created using create-react-app"/>

<link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

<link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

<title>React App</title>

</head>

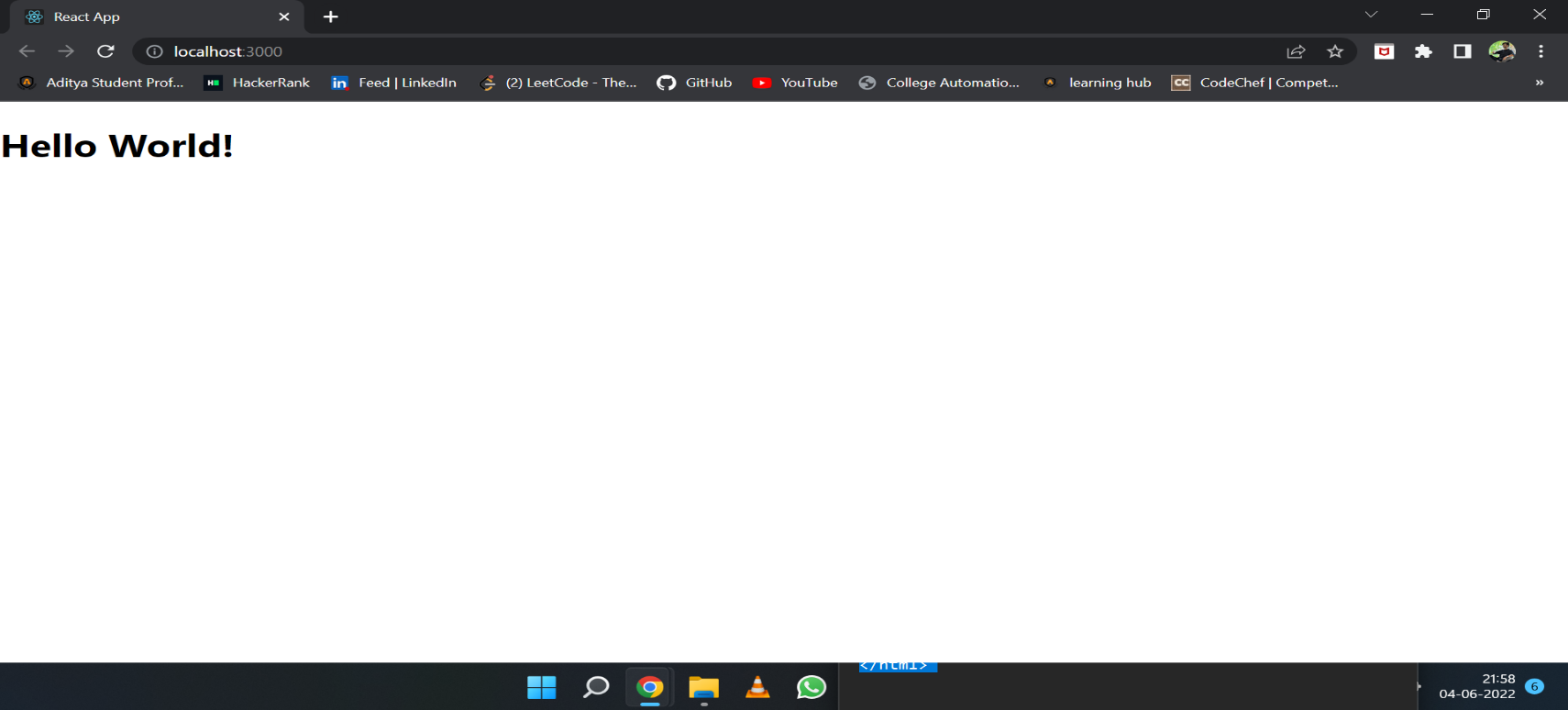
<body>

<div id="root"></div>

</body>

</html>

**Output:**



**Week 10:**

Implement the following in React JS

i. form validation.

ii. to-do list application.

iii. fetching data from MySQL

**Description:**

**Program:**

Index.html:

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="shortcut icon" href="%PUBLIC\_URL%/favicon.ico">

<title>React Form Validation Demo</title>

</head>

<body>

<div id="root"></div>

</body>

</html>

Form.js:

import React, { Component } from 'react';

import { FormErrors } from './FormErrors';

import './Form.css';

class Form extends Component {

constructor (props) {

super(props);

this.state = {

email: '',

password: '',

formErrors: {email: '', password: ''},

emailValid: false,

passwordValid: false,

formValid: false

}

}

handleUserInput = (e) => {

const name = e.target.name;

const value = e.target.value;

this.setState({[name]: value},

() => { this.validateField(name, value) });

}

validateField(fieldName, value) {

let fieldValidationErrors = this.state.formErrors;

let emailValid = this.state.emailValid;

let passwordValid = this.state.passwordValid;

switch(fieldName) {

case 'email':

emailValid = value.match(/^([\w.%+-]+)@([\w-]+\.)+([\w]{2,})$/i);

fieldValidationErrors.email = emailValid ? '' : ' is invalid';

break;

case 'password':

passwordValid = value.length >= 6;

fieldValidationErrors.password = passwordValid ? '': ' is too short';

break;

default:

break;

}

this.setState({formErrors: fieldValidationErrors,

emailValid: emailValid,

passwordValid: passwordValid

}, this.validateForm);

}

validateForm() {

this.setState({formValid: this.state.emailValid && this.state.passwordValid});

}

errorClass(error) {

return(error.length === 0 ? '' : 'has-error');

}

render () {

return (

<form className="demoForm">

<h2>Sign up</h2>

<div className="panel panel-default">

<FormErrors formErrors={this.state.formErrors} />

</div>

<div className={`form-group ${this.errorClass(this.state.formErrors.email)}`}>

<label htmlFor="email">Email address</label>

<input type="email" required className="form-control" name="email"

placeholder="Email"

value={this.state.email}

onChange={this.handleUserInput} />

</div>

<div className={`form-group ${this.errorClass(this.state.formErrors.password)}`}>

<label htmlFor="password">Password</label>

<input type="password" className="form-control" name="password"

placeholder="Password"

value={this.state.password}

onChange={this.handleUserInput} />

</div>

<button type="submit" className="btn btn-primary" disabled={!this.state.formValid}>Sign up</button>

</form>

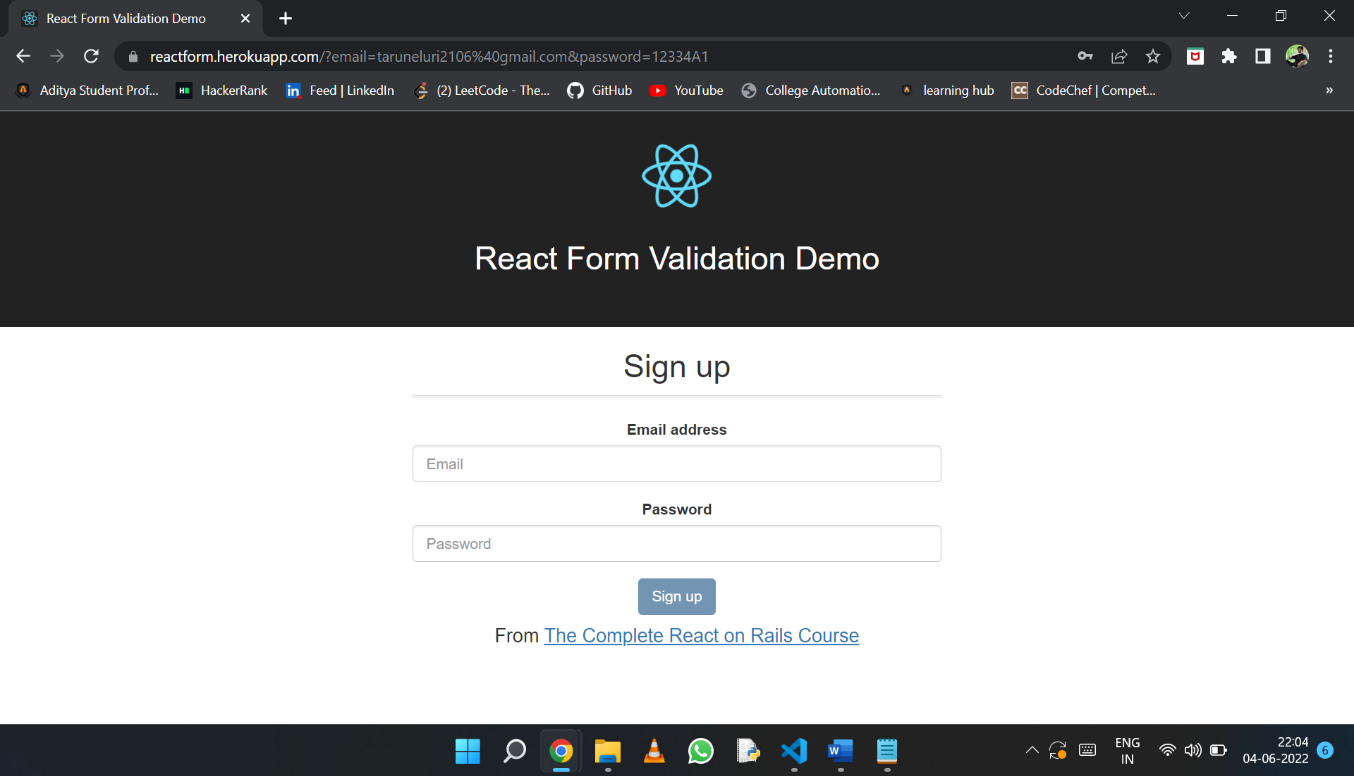
)

}

}

export default Form;

**output :**



**Week 11:**

Write a PHP Script to

i. find out the Sum of individual digits of a given number uptosingle digit.

ii. check whether the given number is Palindrome or not.

iii. work with indexed and associative arrays

iv. fetch data from HTML form

**Program:**

**i.**

<?php

function digSum( $n)

{

$sum = 0;

while($n > 0 || $sum > 9)

{

if($n == 0)

{

$n = $sum;

$sum = 0;

}

$sum += $n % 10;

$n = (int)$n / 10;

}

return $sum;

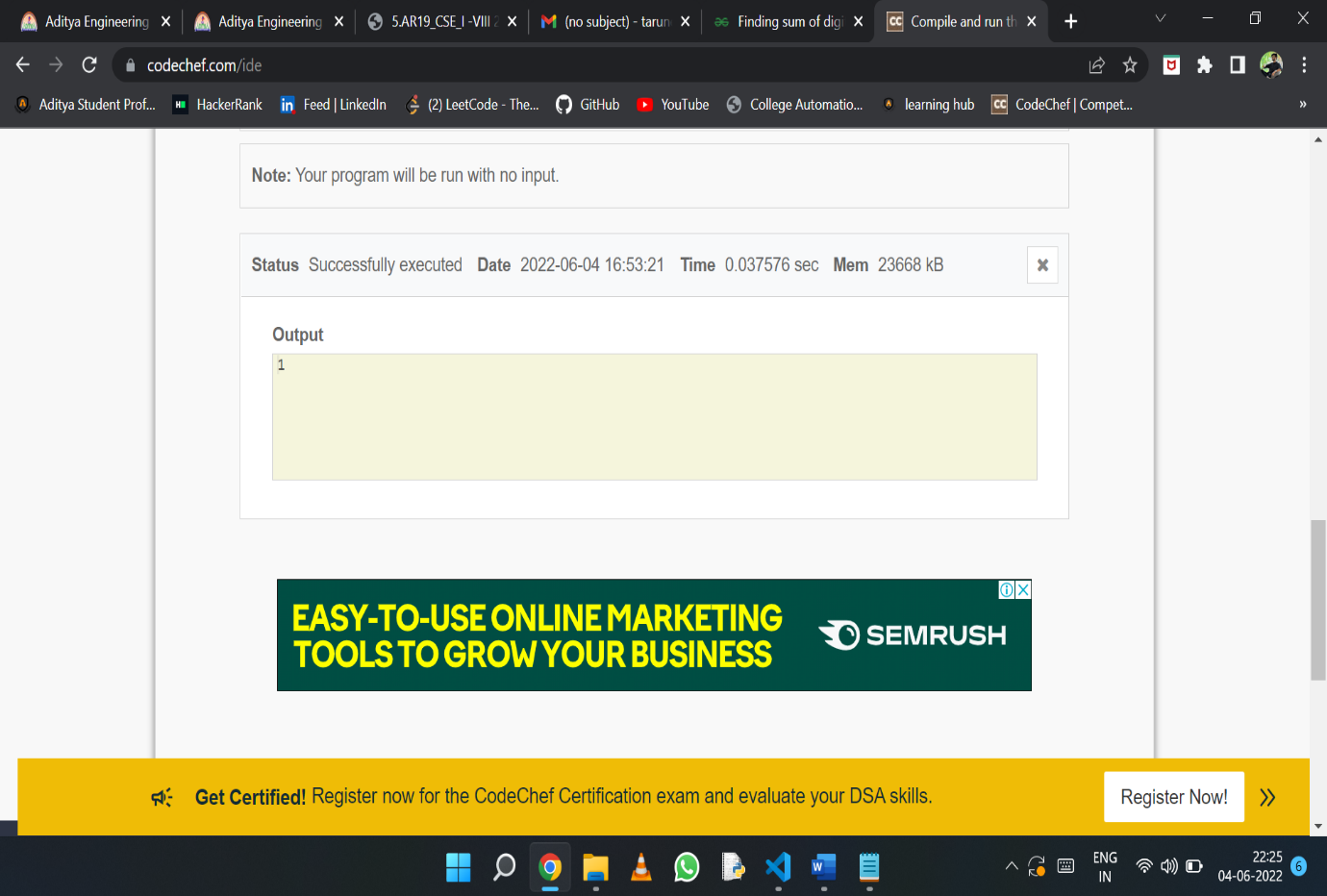
}

$n = 1234;

echo digSum($n);

?>

Output:



ii.

<?php

function Palindrome($number){

$temp = $number;

$new = 0;

while (floor($temp)) {

$d = $temp % 10;

$new = $new \* 10 + $d;

$temp = $temp/10;

}

if ($new == $number){

return 1;

}

else{

return 0;

}

}

$original = 1441;

if (Palindrome($original)){

echo "Palindrome";

}

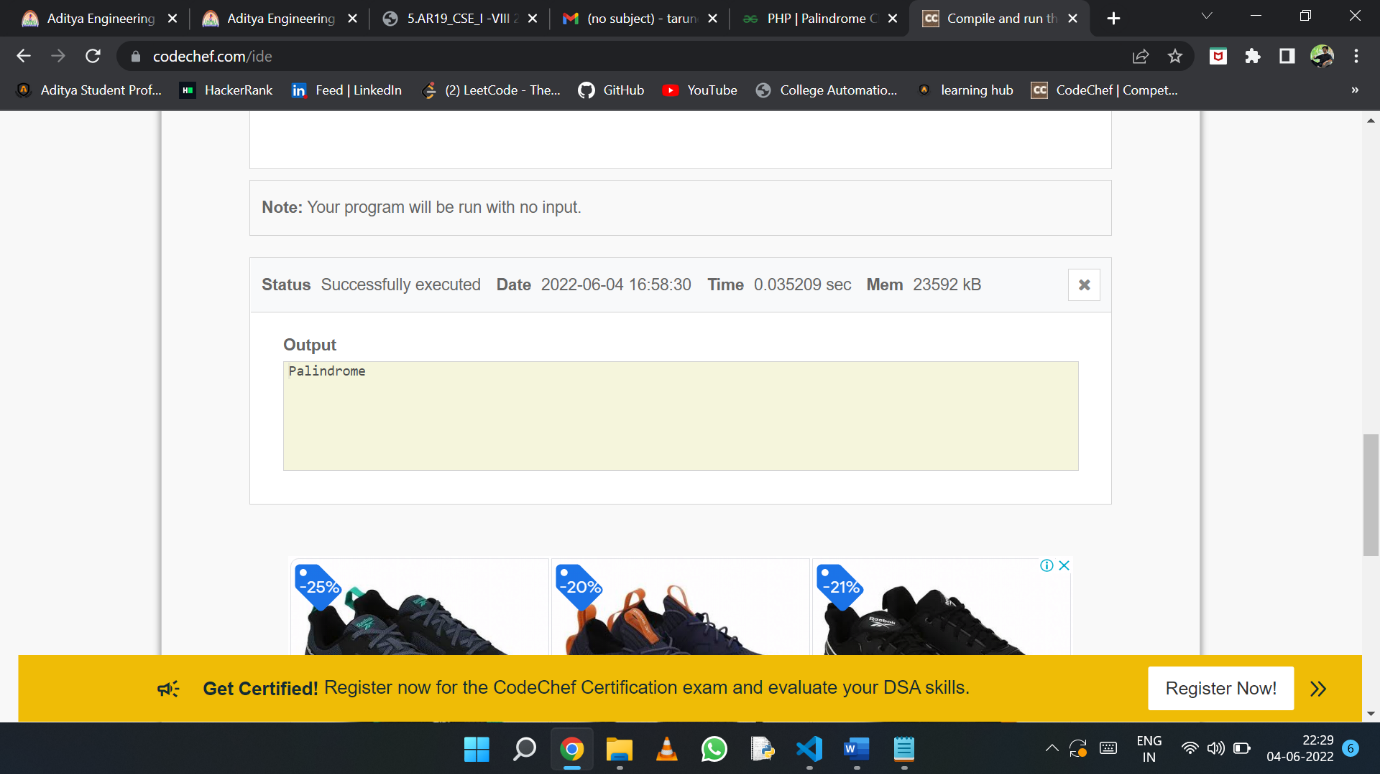
else {

echo "Not a Palindrome";

}

?>

Output:



iii.

<?php

/\* First method to create an associate array. \*/

$student\_one = array("Maths"=>95, "Physics"=>90,

"Chemistry"=>96, "English"=>93,

"Computer"=>98);

/\* Second method to create an associate array. \*/

$student\_two["Maths"] = 95;

$student\_two["Physics"] = 90;

$student\_two["Chemistry"] = 96;

$student\_two["English"] = 93;

$student\_two["Computer"] = 98;

/\* Accessing the elements directly \*/

echo "Marks for student one is:\n";

echo "Maths:" . $student\_two["Maths"], "\n";

echo "Physics:" . $student\_two["Physics"], "\n";

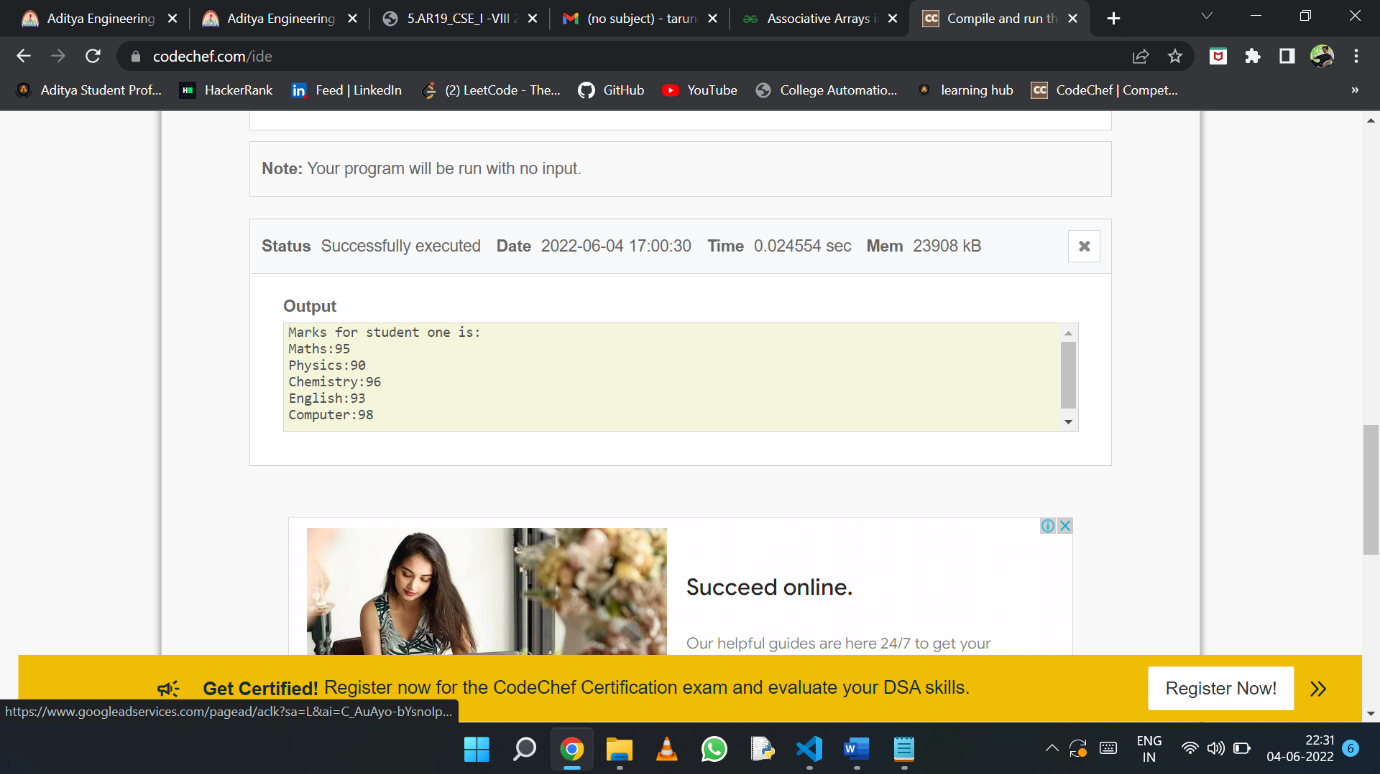
echo "Chemistry:" . $student\_two["Chemistry"], "\n";

echo "English:" . $student\_one["English"], "\n";

echo "Computer:" . $student\_one["Computer"], "\n";

?>

Output:



iv.

<html>

<body>

<form action="welcome.php" method="post">

Name: <input type="text" name="name"><br>

E-mail: <input type="text" name="email"><br>

<input type="submit">

</form>

</body>

</html>

Welcome.php:

<html>

<body>

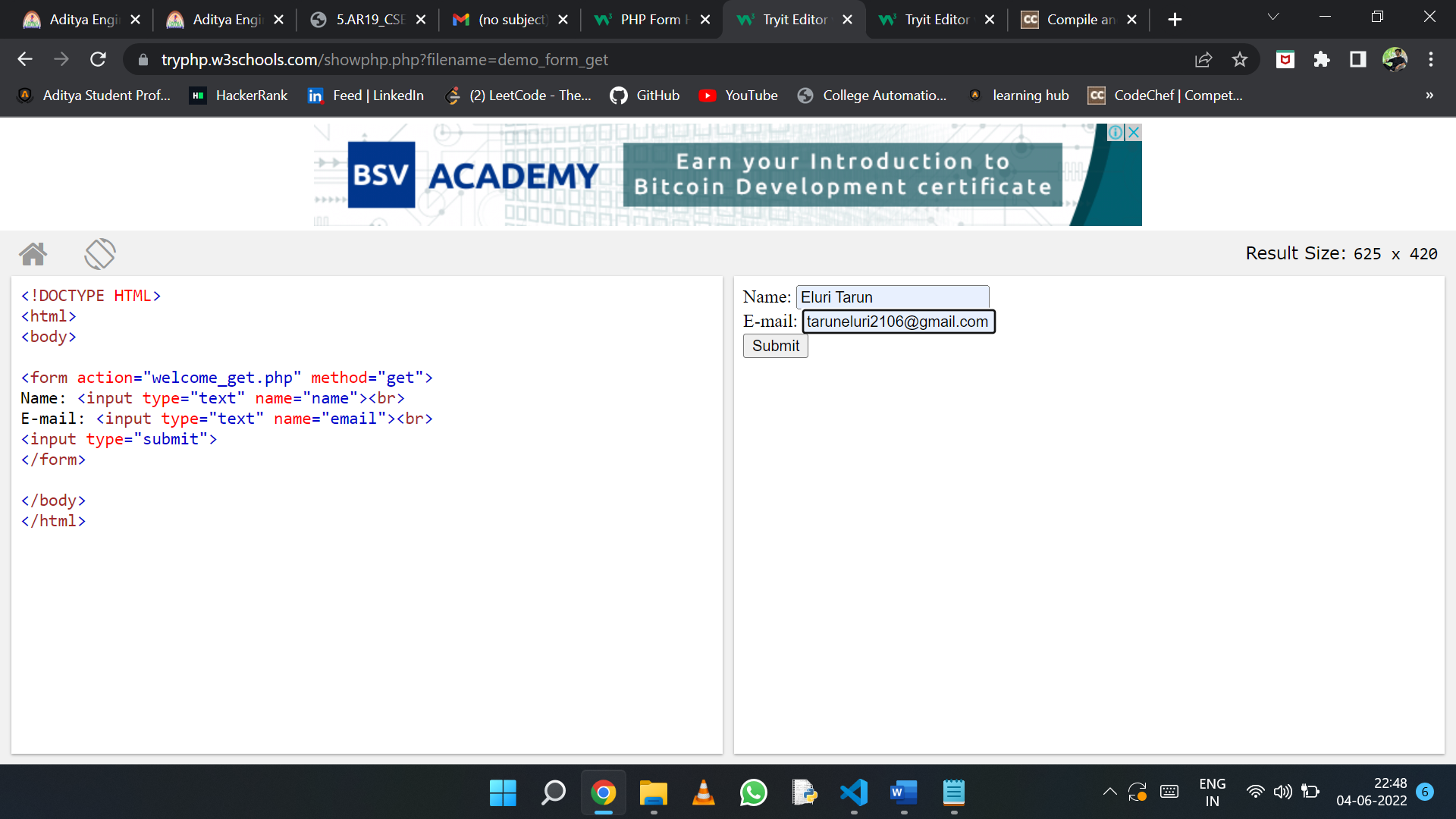
Welcome <?php echo $\_POST["name"]; ?><br>

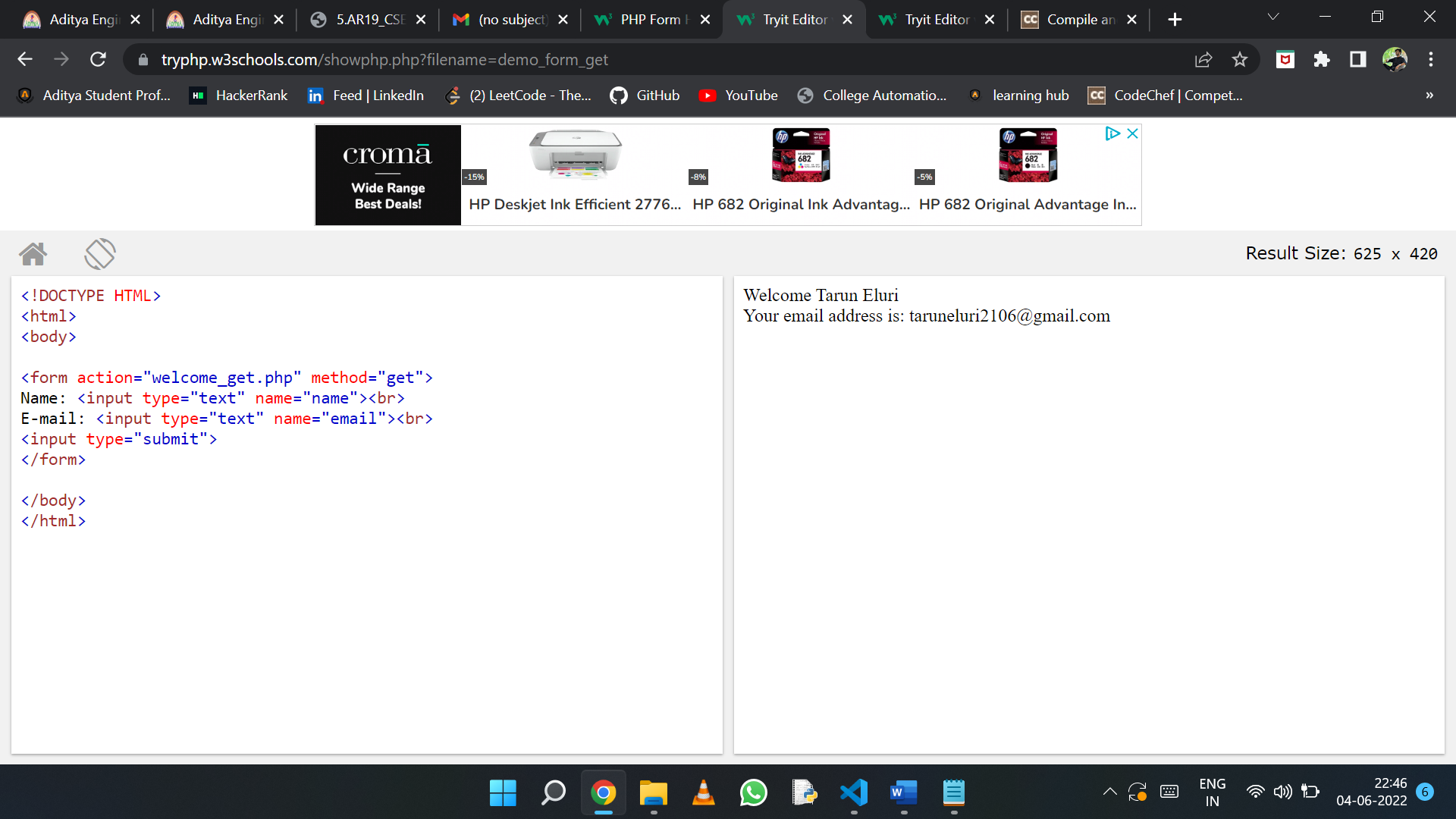
Your email address is: <?php echo $\_POST["email"]; ?>

</body>

</html>

Output:





**Week 12:**

Write a PHP Script to interact with MySQL and perform the following

i. insert data

ii. retrieve data

iii. update data

iv. authenticate user with form data.

Write a program to interact with MongoDB using

i. PHP

ii. Angular JS

iii. React JS

iv. authenticate user with form data.

**Program:**

**1.**

<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)

VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {

echo "New record created successfully";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

$conn->close();

?>

**Program m to interact with MongoDB using Angularjs :**

var http = require('http');

var fs = require('fs');

var path = require('path');

var url = require('url');

var qs = require('querystring');

var mimeTypes = {

"html": "text/html",

"jpeg": "image/jpeg",

"jpg": "image/jpeg",

"png": "image/png",

"js": "text/javascript",

"css": "text/css"};

var databaseUrl = "test";

var collections = ["testData"]

var db = require("mongojs").connect(databaseUrl, collections);

var server = http.createServer(function onRequest(request, response) {

var urlParts = url.parse(request.url);

var fullPath = urlParts.pathname;

var page = 'pages' + urlParts.pathname;

var jsonUserOject = '';

if (fullPath == "/post") {

var userName = '';

request.on('data', function(chunk) {

jsonUserObject = JSON.parse(chunk.toString());

userName = jsonUserObject.name;

userEmail = jsonUserObject.email;

db.testData.insert({name: userName, email: userEmail}, function(err, testData) {

if( err || !testData) console.log("Unable to add user");

});

});

}

var mimeType = mimeTypes[path.extname(page).split(".")[1]];

fs.exists(page, function fileExists(exists) {

if (exists) {

response.writeHead(200, mimeType);

fs.createReadStream(page).pipe(response);

} else {

response.write('Page Not Found');

response.end();

}

});

}).listen(3300);

<script>

angular.module('formExample', [])

.controller('ExampleController', ['$scope', '$http', function($scope, $http) {

$scope.master = {};

$scope.update = function(user) {

if ($scope.formx.$valid) {

$scope.master = angular.copy(user);

$http.post('http://localhost:3300/post', $scope.master

).success(function(data, status, headers, config) {

alert("Success!")

}).error(function(data, status, headers, config) {

alert("Error");

});

}

};

$scope.reset = function() {

$scope.user = angular.copy($scope.master);

};

$scope.reset();

}]);

</script>