SREE RAMA ENGINEERING COLLEGE::TIRUPATI

SREE RAMA

EDUCATIONAL INSTITUTIONS

II B.Tech_I sem(R20)

WEB APPLICATION DEVELOPMENT

LAB MANUAL

THE DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

2021-2022

COMPUTER SCIENCE AND ENGINEERING

Program Outcomes	
PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.
Program Specific Outcomes	
PSO1	**Construct web sites with valid HTML, CSS, JavaScript** **Design and develop web applications using Content Management Systems like Word Press.**
PSO2	**Create responsive Web designs that work on phones, tablets, or traditional laptops and widescreen monitors.**
PSO3	**Develop websites using jQuery to provide interactivity and engaging user experiences**
PSO4	**Embed Google chart tools in a website for better visualization of data. Design and develop web applications using Content Management Systems like Word Press.**

DATA STRUCTURES LAB SYLLABUS

Recommended Systems/Software Requirements:
Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100MB free disk space. HTML, CSS, JS, BOOTSTRAP-CSS, Browser, Google-charts, JQuery

S. No.	List of Experiments
1	HTML: What is a browser?, What is HTML?, Elements and Tags, Basic HTML5 structure, Metadata, <title>, Adding favicon, Comments, headings Task: Create a Basic HTML document
2	HTML (continued): Block-Level Elements & Inline Elements, Links (Understand Absolute vs Relative paths), Lists, Images, iframe (embed youtube video) Task: Create your Profile Page
3	HTML (continued): Tables: <table>, <tr>, <th>, <td>, Attributes for each Table element Task: Create a Class Timetable (to merge rows/columns, use rowspan/colspan)
4	HTML (continued): Form Elements: <input>, <select>, <textarea>, <button>, Attributes for each Form element. Task: Create a Student Hostel Application Form
5	Cascading Style Sheets (CSS): CSS Properties, Types of CSS, Selectors, box model, Pseudo-elements,z-index Task: Make the Hostel Application Form designed in Module -4 beautiful using CSS (add colors,backgrounds, change font properties, borders, etc.)
6	Bootstrap - CSS Framework: Layouts (Containers, Grid system), Forms, Other Components Task: Style the Hostel Application Form designed in Module-5still more beautiful using Bootstrap CSS (Re-size browser and check how the webpage displays in mobile resolution)
7	HTTP & Browser Developer Tools: Understand HTTP Headers (Request & Response Headers), URL & its Anatomy, Developer Tools: Elements/Inspector, Console, Network, Sources, performance, Application Storage. Task: Analyze various HTTP requests (initiators, timing diagrams, responses) and identify problems if any
8	JavaScript: Variables, Data Types, Operators, Statements, Objects, Functions, Events & Event Listeners, DOM. Task: Design a simple calculator using JavaScript to perform sum, product, difference, and quotient operations:
9	Dynamic HTML with JavaScript: Manipulate DOM, Error Handling, Promises, async/await, Modules. Task: Design& develop a Shopping Cart Application with features including Add Products, Update Quantity, Display Price(Sub-Total & Total), Remove items/products from the cart.
10	JQuery - A Javascript Library: Interactions, Widgets, Effects, Utilities, Ajax using JQuery. Task: Validate all Fields and Submit the Hostel Application Form designed in Module-6 using JQuery
11	Google Charts: Understand the Usage of Pie chart, Bar Chart, Histogram, Area & Line Charts, Gantt Charts. Task: Develop an HTML document to illustrate each chart with real-time examples.
12	Open Source CMS (Content Management System): What is a CMS?, Install CMS, Themes, Plugins. Task: Develop an E-learning website using any CMS(for example WordPress)

ATTAINMENT OF PROGRAM OUTCOMES & PROGRAM SPECIFIC OUTCOMES

Exp. No.	Experiment	Program Outcomes Attained	Program Specific Outcomes Attained
1	HTML: What is a browser?, What is HTML?, Elements and Tags, Basic HTML5 structure, Metadata, <title>, Adding favicon, Comments, headings Task: Create a Basic HTML document	PO1, PO2	PSO1
2	HTML (continued): Block-Level Elements & Inline Elements, Links (Understand Absolute vs Relative paths), Lists, Images, iframe (embed youtube video) Task: Create your Profile Page	PO1, PO2	PSO1
3	HTML (continued): Tables: <table>, <tr>, <th>, <td>, Attributes for each Table element Task: Create a Class Timetable (to merge rows/columns, use rowspan/colspan)	PO1, PO2	PSO1
4	HTML (continued): Form Elements: <input>, <select>, <textarea>, <button>, Attributes for each Form element. Task: Create a Student Hostel Application Form	PO1, PO2	PSO1
5	Cascading Style Sheets (CSS): CSS Properties, Types of CSS, Selectors, box model, Pseudo-elements, z-index Task: Make the Hostel Application Form designed in Module -4 beautiful using CSS (add colors, backgrounds, change font properties, borders, etc.)	PO1, PO2	PSO1, PSO2
6	Bootstrap - CSS Framework: Layouts (Containers, Grid system), Forms, Other Components Task: Style the Hostel Application Form designed in Module-5still more beautiful using Bootstrap CSS (Re-size browser and check how the webpage displays in mobile resolution)	PO1, PO3	PSO1,PSO 3
7	HTTP & Browser Developer Tools: Understand HTTP Headers (Request & Response Headers), URL & its Anatomy, Developer Tools: Elements/Inspector, Console, Network, Sources,performance, Application Storage. Task: Analyze various HTTP requests (initiators, timing diagrams, responses) and identify problems if any	PO1, PO2, PO3	PSO1, PSO3
8	JavaScript: Variables, Data Types, Operators, Statements, Objects, Functions, Events & Event Listeners, DOM. Task: Design a simple calculator using JavaScript to perform sum, product, difference, and quotient operations:	PO1, PO2, PO3	PSO1,PSO 3
9	Dynamic HTML with JavaScript: Manipulate DOM, Error Handling, Promises, async/await, Modules. Task: Design& develop a Shopping Cart Application with features including Add Products, Update Quantity, Display Price(Sub-Total & Total), Remove items/products from the cart.	PO1, PO2	PSO1,PSO 3
10	JQuery - A JavaScript Library: Interactions, Widgets, Effects, Utilities, Ajax using JQuery. Task: Validate all Fields and Submit the Hostel Application Form designed in Module-6 using JQuery	PO1, PO2	PSO1,PSO 3,PSO4
11	Google Charts: Understand the Usage of Pie chart, Bar Chart, Histogram, Area & Line Charts, Gantt Charts. Task: Develop an HTML document to illustrate each chart with real-time examples.	PO1, PO2, PO3	PSO1,PSO 4
12	Open Source CMS (Content Management System): What is a CMS?, Install CMS, Themes, Plugins. Task: Develop an E-learning website using any CMS(for example WordPress)	PO1, PO2, PO3	PSO3, PSO4

Exp No:01

AIM:……………………..

Source Code :

Input & Output:

Exp No: 01

What is browser?

- A browser is software that accesses and displays pages and files on the web.
- Browsers require a connection to the Internet (e.g., through a cable modem, a direct Ethernet connection, or Wi-Fi).
- Popular web browsers include Firefox, Internet Explorer, and Safari.

What is HTML?

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

Element

- The HTML element is everything from the start tag to the end tag:
- <tagname>Content goes here...</tagname>

Examples of some HTML elements:

```
<h1>My First Heading</h1>
<p>My first paragraph.</p>
```

Basic HTML 5 Structure

Below is a visualization of an HTML page structure:

```
<html>
<head>
<title>Page title</title>
</head>
<body>
<h1>This is a heading</h1>
```

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```

Meta Data:
The <meta> tag defines metadata about an HTML document. Metadata is data (information) about data.
<meta> tags always go inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings.
Metadata will not be displayed on the page, but is machine parsable.
Metadata is used by browsers (how to display content or reload page), search engines (keywords), and other web services.
There is a method to let web designers take control over the viewport (the user's visible area of a web page), through the <meta> tag (See "Setting The Viewport" example below).
Favicon:
A favicon is a small image displayed next to the page title in the browser tab.
How to Add a Favicon in HTML
You can use any image you like as your favicon. You can also create your own favicon on sites like https://favicon.cc.
A favicon image is displayed to the left of the page title in the browser tab, like this:
To add a favicon to your website, either save your favicon image to the root directory of your webserver, or create a folder in the root directory called images, and save your favicon image in this folder. A common name for a favicon image is "favicon.ico".
Headings:
HTML headings are defined with the <h1> to <h6> tags.
<h1> defines the most important heading. <h6> defines the least important heading.
Example

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

AIM: Create a Basic HTML document

Source Code:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Page Title</title>
  <link rel="icon" type="image/x-icon" href="/images/favicon.ico">
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Exp No: 02

HTML Block and Inline Elements

Every HTML element has a default display value, depending on what type of element it is. There are two display values: block and inline.

Inline Elements

- An inline element does not start on a new line.
- An inline element only takes up as much width as necessary.
- This is a <span> element inside a paragraph.

Example

<span>Hello World</span>

Block-level Elements

- A block-level element always starts on a new line.
- A block-level element always takes up the full width available (stretches out to the left and right as far as it can).
- A block level element has a top and a bottom margin, whereas an inline element does not.
- The <div> element is a block-level element.

Example

<div>Hello World</div>

Here are the block-level elements in HTML:

<address>
<article>
<aside>
<blockquote>
<canvas>
<dd>
<div>
<dl>
<dt>
<fieldset>
<figcaption>
<figure>
<footer>
<form>
<h1>-<h6>
<header>
<hr>
<li>
<main>
<nav>
<noscript>
<ol>
<p>
<pre>
<section>
<table>
<tfoot>
<ul>
<video>

HTML Links

- Links are found in nearly all web pages. Links allow users to click their way from page to page.
- HTML links are hyperlinks.
- You can click on a link and jump to another document.
- When you move the mouse over a link, the mouse arrow will turn into a little hand.

- Absolute URLs vs. Relative URLs
- Both examples above are using an absolute URL (a full web address) in the href attribute.
- A local link (a link to a page within the same website) is specified with a relative URL (without the "https://www" part):

Example

```
<h2>Absolute URLs</h2>
<p><a href="https://www.w3.org/">W3C</a></p>
<p><a href="https://www.google.com/">Google</a></p>

<h2>Relative URLs</h2>
<p><a href="html_images.asp">HTML Images</a></p>
<p><a href="/css/default.asp">CSS Tutorial</a></p>
```

HTML Images

Images can improve the design and the appearance of a web page.

Example

```
<img src="pic_trulli.jpg" alt="Italian Trulli">
```

HTML Images Syntax

The HTML <img> tag is used to embed an image in a web page.

Images are not technically inserted into a web page; images are linked to web pages. The <img> tag creates a holding space for the referenced image.

The <img> tag is empty, it contains attributes only, and does not have a closing tag.The <img> tag has two required attributes:

- src - Specifies the path to the image
- alt - Specifies an alternate text for the image

Syntax

```
<img src="url" alt="alternatetext">
```

Aim: To create a Profile page.

Source Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  </head>
<body>
<div class="ScriptTop">
   <div class="rt-container">
      <div class="col-rt-4" id="float-right">
       </div>
        </div>
</div>
<header>
   <div class="rt-container">
        <div class="col-rt-12">
        <div class="rt-heading">
                <h1>Student Profile Page </h1>
            </div>
      </div>
   </div>
</header>
```

```
<section>
  <div class="rt-container">
      <div class="col-rt-12">
        <div class="Scriptcontent">
    <!-- Student Profile -->
<div class="student-profile py-4">
 <div class="container">
  <div class="row">
   <div class="col-lg-4">
    <div class="card shadow-sm">
     <div class="card-header bg-transparent text-center">
      <img class="profile_img" src="https://source.unsplash.com/600x300/?student"
alt="student dp">
      <h3>Ishmam Ahasan Samin</h3>
     </div>
     <div class="card-body">
      <p class="mb-0"><strong class="pr-1">Student ID:</strong>321000001</p>
      <p class="mb-0"><strong class="pr-1">Class:</strong>4</p>
      <p class="mb-0"><strong class="pr-1">Section:</strong>A</p>
     </div>
    </div>
   </div>
   <div class="col-lg-8">
    <div class="card shadow-sm">
     <div class="card-header bg-transparent border-0">
      <h3 class="mb-0"><i class="far fa-clone pr-1"></i>General Information</h3>
     </div>
     <div class="card-body pt-0">
      <table class="table table-bordered">
       <tr>
        <th width="30%">Roll</th>
        <td width="2%">:</td>
        <td>125</td>
       </tr>
       <tr>
        <th width="30%">Academic Year	</th>
        <td width="2%">:</td>
        <td>2020</td>
       </tr>
       <tr>
        <th width="30%">Gender</th>
        <td width="2%">:</td>
        <td>Male</td>
       </tr>
       <tr>
        <th width="30%">Religion</th>
        <td width="2%">:</td>
        <td>Group</td>
       </tr>
       <tr>
        <th width="30%">blood</th>
        <td width="2%">:</td>
        <td>B+</td>
       </tr>
      </table>
     </div>
```

```
        </div>
          <div style="height: 26px"></div>
        <div class="card shadow-sm">
          <div class="card-header bg-transparent border-0">
            <h3 class="mb-0"><i class="far fa-clone pr-1"></i>Other Information</h3>
          </div>
          <div class="card-body pt-0">
              <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
                      </div>
                      </div>
    </div>
</section>
          </body>
</html>
```

Exp No:03

Table:

- HTML tables allow web developers to arrange data into rows and columns.
- A table in HTML consists of table cells inside rows and columns
- Each table cell is defined by a <td> and a </td> tag.(td=Table data)
- Everything between <td> and </td> are the content of the table cell.
- Each table row starts with a <tr> and end with a </tr> tag.
- use the <th> tag instead of the <td> tag to keep header.

Aim: Class time table

Source Code:

```
<!DOCTYPE html>
<html>
 <body>    <h1>TIME TABLE</h1>
    <table border="5" cellspacing="0" align="center">
        <!--<caption>Timetable</caption>-->
        <tr>
            <td align="center" height="50"
                width="100"><br>
                <b>Day/Period</b></br>
            </td>
            <td align="center" height="50"
                width="100">
                <b>I<br>9:30-10:20</b>
            </td>
            <td align="center" height="50"
                width="100">
                <b>II<br>10:20-11:10</b>
            </td>
            <td align="center" height="50"
                width="100">
                <b>III<br>11:10-12:00</b>
            </td>
            <td align="center" height="50"
                width="100">
                <b>12:00-12:40</b>
            </td>
            <td align="center" height="50"
                width="100">
                <b>IV<br>12:40-1:30</b>
            </td>
            <td align="center" height="50"
                width="100">
                <b>V<br>1:30-2:20</b>
            </td>
            <td align="center" height="50"
                width="100">
                <b>VI<br>2:20-3:10</b>
            </td>
            <td align="center" height="50"
                width="100">
                <b>VII<br>3:10-4:00</b>
            </td>
        </tr>
        <tr>
            <td align="center" height="50">
                <b>Monday</b></td>
            <td align="center" height="50">Eng</td>
            <td align="center" height="50">Mat</td>
            <td align="center" height="50">Che</td>
            <td rowspan="6" align="center" height="50">
                <h2>L<br>U<br>N<br>C<br>H</h2>
            </td>
            <td colspan="3" align="center"
                height="50">LAB</td>
            <td align="center" height="50">Phy</td>
        </tr>
        <tr>
            <td align="center" height="50">
                <b>Tuesday</b>
            </td>
            <td colspan="3" align="center"
                height="50">LAB
            </td>
            <td align="center" height="50">Eng</td>
            <td align="center" height="50">Che</td>
            <td align="center" height="50">Mat</td>
            <td align="center" height="50">SPORTS</td>
        </tr>
        <tr>
            <td align="center" height="50">
                <b>Wednesday</b>
```

```
          </td>
          <td align="center" height="50">Mat</td>
          <td align="center" height="50">phy</td>
          <td align="center" height="50">Eng</td>
          <td align="center" height="50">Che</td>
          <td colspan="3" align="center"
             height="50">LIBRARY
          </td>
       </tr>
       <tr>
          <td align="center" height="50">
             <b>Thursday</b>
          </td>
          <td align="center" height="50">Phy</td>
          <td align="center" height="50">Eng</td>
          <td align="center" height="50">Che</td>
          <td colspan="3" align="center"
             height="50">LAB
          </td>
          <td align="center" height="50">Mat</td>
       </tr>
       <tr>
          <td align="center" height="50">
             <b>Friday</b>
          </td>
          <td colspan="3" align="center"
             height="50">LAB
          </td>
          <td align="center" height="50">Mat</td>
          <td align="center" height="50">Che</td>
          <td align="center" height="50">Eng</td>
          <td align="center" height="50">Phy</td>
       </tr>
       <tr>
          <td align="center" height="50">
             <b>Saturday</b>
          </td>
          <td align="center" height="50">Eng</td>
          <td align="center" height="50">Che</td>
          <td align="center" height="50">Mat</td>
          <td colspan="3" align="center"
             height="50">SEMINAR
          </td>
          <td align="center" height="50">SPORTS</td>
       </tr>
    </table>
</body>

</html>
```

Output:

Exp No: 04

Form Elements:

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

- <input>
- <label>
- <select>
- <textarea>
- <button>
- <fieldset>
- <legend>
- <datalist>
- <output>
- <option>
- <optgroup>

Attributes:

accept-charset	:	Specifies the character encodings used for form submission
action	:	Specifies where to send the form-data when a form is submitted
autocomplete	:	Specifies whether a form should have autocomplete on or off
enctype	:	Specifies how the form-data should be encoded when submitting it to the server (only for method="post")
method	:	Specifies the HTTP method to use when sending form-data
name	:	Specifies the name of the form
novalidate	:	Specifies that the form should not be validated when submitted
rel	:	Specifies the relationship between a linked resource and the current document
target	:	Specifies where to display the response that is received after submitting the form

Aim: Student hostel application Form:

Source Code:

```
<html>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
<body>
<div class="container-fluid">
<h1 align="center">Hostel Registration Form</h1>
<!--<div id="s0">
   <img src="http://www.kgcas.ac.in/images/banner2.jpg" height="50" width="80">
   <img src="http://www.spvp.edu.in/SBPP/assets/img/portfolio/facilities-big-2.jpg"
height="50" width="80" id="img1">
   <img src="https://campushunt.in/photogallery/kg-college-of-arts-and-science-cateen-
344.jpg" height="50" width="80" id="img2">
</div>-->
<div id="s1">
        <form method="post">

        <table class="table table-striped table-condensed">
        <tr>
        <th align="left" class="txt1">Name of student:</th>
        <th align="left"><input type="text"></th></tr>
   <tr><th align="left">Class & Branch:</th> <th align="left"><input type="text" ></th></tr>
   <tr><th align="left">Adm. Fee Receipt no:</th> <th align="left"><input type="text"
></th></tr>
   <tr><th align="left">Gender:</th><th align="left"><input type="radio" name="gender"
value="m" checked> Male&nbsp;&nbsp;&nbsp;
   <input type="radio" name="gender" value="f"> Female</th></tr>
   <tr><th align="left">Date of Birth:</th> <th align="left"><select id="sl1">
   <option value="1">1</option>
   <option value="2">2</option>
   <option value="3">3</option>
   <option value="4">4</option>
   <option value="5">5</option>
    </select>
   <select id="sl2">
   <option value="Jan">January</option>
   <option value="Feb">February</option>
   <option value="Mar">March</option>
   <option value="Apr" selected>April</option>
   <option value="May">May</option>
    </select>
   <select id="sl3">
   <option value="2010">2010</option>
   <option value="2">2011</option>
   <option value="3">2012</option>
   <option value="4">2013</option>
   <option value="5">2014</option>
   <option value="6">2015</option>
   <option value="7">2016</option>
   <option value="8">2017</option>
   </select></th></tr>
   <tr><th align="left">Category:</th> <th align="left"><input type="text" ></th></tr>
   <tr><th align="left">Nationality:</th><th align="left"> <input type="text" ></th></tr>
   <tr><th align="left">Fathers name:</th><th align="left"><input type="text" ></th></tr>
   <tr><th align="left">Permanent Address:</th><th align="left"> <textarea rows="3"
cols="18"></textarea></th></tr>
   <tr><th align="left">Mobile no:</th><th align="left"><input type="text" ></th></tr>
   <tr><th colspan="2"><input type="submit" id="btn1" value="Submit">
   <input type="reset" value="Reset" id="btn2"> <input type="submit" value="Print"
id="btn3"></th></tr>
        </table>
        </form>
</div>
</div>
</body>
</html>
```

Exp No:05

What is CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once

External stylesheets are stored in CSS files

CSS properties:

• align-content	:	Specifies the alignment between the lines inside a flexible container when the items do not use all available space
• align-items	:	Specifies the alignment for items inside a flexible container
• align-self	:	Specifies the alignment for selected items inside a flexible container
• all	:	Resets all properties (except unicode-bidi and direction)
• animation	:	A shorthand property for all the animation-* properties

CSS Selectors:

In CSS, selectors are patterns used to select the element(s) you want to style.

• class	.intro	Selects all elements with class="intro"
• class1.class2	.name1.name2	Selects all elements with both name1 and name2 set within its class attribute
• class1 .class2	.name1 .name2	Selects all elements with name2 that is a descendant of an element with name1
• #id	#firstname	Selects the element with id="firstname"
• *	*	Selects all elements
• element	p	Selects all <p> elements
• element.class	p.intro	Selects all <p> elements with c

CSS Box Model:

- 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.

What are Pseudo-Elements?

- A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to: Style the first letter, or line, of an element ,Insert content before, or after, the content of an element

Syntax

- The syntax of pseudo-elements:

```
Selector:: pseudo-element {
  Property: value;
}
```

The z-index Property

- When elements are positioned, they can overlap other elements.
- The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).
- An element can have a positive or negative stack order

Aim: Student hostel application Form.

Source Code:

```
<html>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

```
<body>
<div class="container-fluid">
<h1 align="center">Hostel Registration Form</h1>
<!--<div id="s0">
   <img src="http://www.kgcas.ac.in/images/banner2.jpg" height="50" width="80">
   <img src="http://www.spvp.edu.in/SBPP/assets/img/portfolio/facilities-big-2.jpg"
height="50" width="80" id="img1">
   <img src="https://campushunt.in/photogallery/kg-college-of-arts-and-science-cateen-
344.jpg" height="50" width="80" id="img2">
</div>-->
<div id="s1">
        <form method="post">
        <table class="table table-striped table-condensed">
        <tr>
        <th align="left" class="txt1">Name of student:</th>
        <th align="left"><input type="text"></th></tr>
   <tr><th align="left">Class & Branch:</th> <th align="left"><input type="text" ></th></tr>
   <tr><th align="left">Adm. Fee Receipt no:</th> <th align="left"><input type="text"
></th></tr>
   <tr><th align="left">Gender:</th><th align="left"><input type="radio" name="gender"
value="m" checked> Male&nbsp;&nbsp;&nbsp;
   <input type="radio" name="gender" value="f"> Female</th></tr>
   <tr><th align="left">Date of Birth:</th> <th align="left"><select id="sl1">
   <option value="1">1</option>
   <option value="2">2</option>
   <option value="3">3</option>
   <option value="4">4</option>
   <option value="5">5</option>
    </select>
   <select id="sl2">
   <option value="Jan">January</option>
   <option value="Feb">February</option>
   <option value="Mar">March</option>
   <option value="Apr" selected>April</option>
   <option value="May">May</option>
    </select>
   <select id="sl3">
   <option value="2010">2010</option>
   <option value="2">2011</option>
   <option value="3">2012</option>
   <option value="4">2013</option>
   <option value="5">2014</option>
   <option value="6">2015</option>
   <option value="7">2016</option>
   <option value="8">2017</option>
   </select></th></tr>
   <tr><th align="left">Category:</th> <th align="left"><input type="text" ></th></tr>
   <tr><th align="left">Nationality:</th><th align="left"> <input type="text" ></th></tr>
   <tr><th align="left">Fathers name:</th><th align="left"><input type="text" ></th></tr>
   <tr><th align="left">Permanent Address:</th><th align="left"> <textarea rows="3"
cols="18"></textarea></th></tr>
   <tr><th align="left">Mobile no:</th><th align="left"><input type="text" ></th></tr>
   <tr><th colspan="2"><input type="submit" id="btn1" value="Submit">
   <input type="reset" value="Reset" id="btn2"> <input type="submit" value="Print"
id="btn3"></th></tr>
        </table>
```

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

CSS Code:
div.container-fluid h1
{
  color:blue;
  background-color:#acd487;
  font-family:Gill Sans MT;
  font-size:20px;
  }
div.container-fluid div#s1
{
  position:absolute;
  top:50px;
  left:15px;
}
div.container-fluid div#s0 img
{
  border:2px dashed blue;
}
div.container-fluid div#s0 img#img1
{
  position:absolute;
  top:115px;
  left:15px;
}
div.container-fluid div#s0 img#img2
{
  position:absolute;
  top:180px;
  left:15px;
}
div.container-fluid div#s1 input#btn1
{
padding:5px;
width:65px;
}
div.container-fluid div#s1 input#btn2
{
padding:5px;
width:65px;
}
div.container-fluid div#s1 input#btn3
{
padding:5px;
width:65px;
}
div.container-fluid div#s1 input
{
  border:2px solid #aedfac;
  color:red;
}
```

```
div.container-fluid div#s1 textarea
{
  border:2px solid #aedfac;
}
div.container-fluid div#s1 select
{
  border:2px solid #aedfac;
  }
```

Exp No: 06

Bootstraps-CSS Frame works: Grid Container:

- To make an HTML element behave as a grid container, you have to set the display property to grid or inline-grid.
- Grid containers consist of grid items, placed inside columns and rows.
- The grid-template-columns Property
- The grid-template-columns property defines the number of columns in your grid layout, and it can define the width of each column.
- The value is a space-separated-list, where each value defines the width of the respective column.
- If you want your grid layout to contain 4 columns, specify the width of the 4 columns, or "auto" if all columns should have the same width.

Form controls

- Textual form controls—like <input>s, <select>s, and <textarea>s—are styled with the .form-control class.
- Included are styles for general appearance, focus state, sizing, and more.
- Be sure to explore our custom forms to further style <select>s.

Aim: Student hostel application Form:

Source Code:

```
<html>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
<body>
<div class="container-fluid">
<h1 align="center">Hostel Registration Form</h1>
<!--<div id="s0">
    <img src="http://www.kgcas.ac.in/images/banner2.jpg" height="50" width="80">
    <img src="http://www.spvp.edu.in/SBPP/assets/img/portfolio/facilities-big-2.jpg"
height="50" width="80" id="img1">
    <img src="https://campushunt.in/photogallery/kg-college-of-arts-and-science-cateen-
344.jpg" height="50" width="80" id="img2">
</div>-->
<div id="s1">
          <form method="post">
          <table class="table table-striped table-condensed">
          <tr>
          <th align="left" class="txt1">Name of student:</th>
          <th align="left"><input type="text"></th></tr>
    <tr><th align="left">Class & Branch:</th> <th align="left"><input type="text" ></th></tr>
    <tr><th align="left">Adm. Fee Receipt no:</th> <th align="left">
   <input type="text"></th></tr>
```

```
 <tr><th align="left">Gender:</th><th align="left"><input type="radio" name="gender"
value="m" checked> Male&nbsp;&nbsp;&nbsp;
    <input type="radio" name="gender" value="f"> Female</th></tr>
    <tr><th align="left">Date of Birth:</th> <th align="left"><select id="sl1">
    <option value="1">1</option>
    <option value="2">2</option>
    <option value="3">3</option>
    <option value="4">4</option>
    <option value="5">5</option>
     </select>
    <select id="sl2">
    <option value="Jan">January</option>
    <option value="Feb">February</option>
    <option value="Mar">March</option>
    <option value="Apr" selected>April</option>
    <option value="May">May</option>
     </select>
    <select id="sl3">
    <option value="2010">2010</option>
    <option value="2">2011</option>
    <option value="3">2012</option>
    <option value="4">2013</option>
    <option value="5">2014</option>
    <option value="6">2015</option>
    <option value="7">2016</option>
    <option value="8">2017</option>
    </select></th></tr>
    <tr><th align="left">Category:</th> <th align="left"><input type="text" ></th></tr>
    <tr><th align="left">Nationality:</th><th align="left"> <input type="text" ></th></tr>
    <tr><th align="left">Fathers name:</th><th align="left"><input type="text" ></th></tr>
    <tr><th align="left">Permanent Address:</th><th align="left"> <textarea rows="3"
cols="18"></textarea></th></tr>
    <tr><th align="left">Mobile no:</th><th align="left"><input type="text" ></th></tr>
    <tr><th colspan="2"><input type="submit" id="btn1" value="Submit">
    <input type="reset" value="Reset" id="btn2"> <input type="submit" value="Print"
id="btn3"></th></tr>
          </table>
          </form>
</div>
</div>
</body>
</html>
```

CSS Code:

```
div.container-fluid h1
{
  color:blue;
  background-color:#acd487;
  font-family:Gill Sans MT;
  font-size:20px;
  }
div.container-fluid div#s1
{
  position:absolute;
  top:50px;
  left:15px;
}
div.container-fluid div#s0 img
{
  border:2px dashed blue;
}
div.container-fluid div#s0 img#img1
{
  position:absolute;
  top:115px;
  left:15px;
}
div.container-fluid div#s0 img#img2
{
  position:absolute;
  top:180px;
  left:15px;
}
div.container-fluid div#s1 input#btn1
{
padding:5px;
width:65px;
}
div.container-fluid div#s1 input#btn2
{
padding:5px;
width:65px;
}
div.container-fluid div#s1 input#btn3
{
padding:5px;
width:65px;
}
div.container-fluid div#s1 input
{
  border:2px solid #aedfac;
  color:red;
}
div.container-fluid div#s1 textarea
{
  border:2px solid #aedfac;
}
div.container-fluid div#s1 select
{
  border:2px solid #aedfac;
  }
```

Exp No:07

Aim: HTTP & Browser Developer Tools: Understand HTTP Headers (Request & Response Headers), URL & its Anatomy, Developer Tools: Elements/Inspector, Console, Network, Sources, performance, Application Storage.

- HTTP is a transfer protocol used by the World Wide Web distributed hypermedia system to retrieve distributed objects.
- HTTP uses TCP as a transport layer. Certain design features of HTTP interact badly with TCP, causing problems with performance and with server scalability.

- Latency problems are caused by opening a single connection per request, through connection setup and slow-start costs.
- Further avoidable latency is incurred due to the protocol only returning a single object per request.
- Scalability problems are caused by TCP requring a server to maintain state for all recently closed connections.

Requests

- The request format for HTTP is quite simple. The first line specifies an object, together with the name of an object to apply the method to.
- The most commonly used method is "GET", which ask the server to send a copy of the object to the client.
- The client can also send a series of optional headers; these headers are in RFC-822 format.
- The most common headers are "Accept", which tells the server which object types the client can handle, and "User-Agent", which gives the implementation name of the client.

Request syntax

```
 <METHOD> <URI> "HTTP/1.0" <crlf>
{<Header>: <Value> <crlf>}*
<crlf>
```

Example

```
GET  /index.html HTTP/1.0
Accept: text/plain
Accept: text/html
Accept: */*
User-Agent:  Yet Another User Agent
```

Responses

- The response format is also quite simple. Responses start with a status line indicating which version of HTTP the server is running, together with a result code and an optional message.
- This is followed by a series of optional object headers; the most important of these are "Content-Type", which describes the type of the object being returned, and "Content-Length", which indicates the length. The headers are terminated by an empty line.

The server now sends any requested data. After the data have been sent, the server drops the connection.

Response syntax

```
 "HTTP/1.0" <result-code> [<message>]  <crlf>
{<Header>: <Value> <crlf>}*
<crlf>
```

Example

```
HTTP/1.0 200 OK
Server: MDMA/0.1
MIME-version: 1.0
Content-type: text/html
Last-Modified: Thu Jul  7 00:25:33 1994
Content-Length: 2003

<title>MDMA - Multithreaded Daemon for Multimedia Access</title>
<hr>
....
<hr>
<h2> MDMA - The speed daemon </h2>
<hr>
[Connection closed by foreign host]
```

Source Code: Analyze various HTTP requests (initiators, timing diagrams, responses) and identify problems if any

- The problems with HTTP can best be understood by looking at the network traffic generated by a typical HTTP transaction. This example was generated by using Van Jacobsen's tcpdump program to monitor a client at UNC fetching a copy of the NCSA Home page. This page is 1668 bytes long, including response headers. The client at UNC was a Sun Sparcstation 20/512, running Solaris 2.3. The server at NCSA was a Hewlett Packard 9000/735 running HP-UX.

- The headers used in the request shown were captured from an xmosaic request. The headers consisted of 42 lines totaling around 1130 bytes; of these lines, 41 were "Accept".

Stage 1: Time = 0
The trace begins with the client sending a connection request to the http port on the server

```
.00000 unc.32840 > ncsa.80: S 2785173504:2785173504(0) win 8760 <mss 1460> (DF)
```

Stage 2: Time = 0.077
The server responds to the connect request with a connect response. The client acknowledges the connect response, and send the first 536 bytes of the request.

```
.07769 ncsa.80 > unc.32840: S 530752000:530752000(0) ack 2785173505 win 8192
.07784 unc.32840 > ncsa.80: . ack 1 win 9112 (DF)
.07989 unc.32840 > ncsa.80: P 1:537(536) ack 1 win 9112 (DF)
```

Stage 3: Time = 0.35
The server acknowledges the first part of the request. The client then sends the second part, and without waiting for a response, follows up with the third and final part of the request.

```
.35079 ncsa.80 > unc.32840: . ack 537 win 8192
.35092 unc.32840 > ncsa.80: . 537:1073(536) ack 1 win 9112 (DF)
.35104 unc.32840 > ncsa.80: P 1073:1147(74) ack 1 win 9112 (DF)
```

Stage 4: Time = 0.45
The server sends a packet acknowledging the second and third parts of the request, and containing the first 512 bytes of the response. It follows this with another packet containing the second 512 bytes. The client then sends a message acknowledging the first two response packets.

```
.45116 ncsa.80 > unc.32840: . 1:513(512) ack 1147 win 8190
.45473 ncsa.80 > unc.32840: . 513:1025(512) ack 1147 win 8190
.45492 unc.32840 > ncsa.80: . ack 1025 win 9112 (DF)
```

Stage 5: Time = 0.53
The server sends the third and fourth response packet. The fourth packet also contains a flag indicating that the connection is being closed. The client acknowledges the data, then sends a message announcing that it too is closing the connection. From the point of view of the client program, the transaction is now over.

```
.52521 ncsa.80 > unc.32840: . 1025:1537(512) ack 1147 win 8190
.52746 ncsa.80 > unc.32840: FP 1537:1853(316) ack 1147 win 8190
.52755 unc.32840 > ncsa.80: . ack 1854 win 9112 (DF)
.52876 unc.32840 > ncsa.80: F 1147:1147(0) ack 1854 win 9112 (DF)
```

Stage 6: Time = 0.60
The server acknowledges the close.

```
.59904 ncsa.80 > unc.32840: . ack 1148 win 8189
```

Exp No:04

JavaScript:
Variables:

- Variables are containers for storing data (storing data values).
- In this example, x, y, and z, are variables, declared with the var keyword.
- var x = 5;
- var y = 6;
- var z = x + y;
- 4 Ways to Declare a JavaScript Variable:
- Using var
- Using let
- Using const
- Using nothing

Data Types

Number:	let length = 16;
String:	let lastName = "Johnson";
Object:	let x = {firstName:"John", lastName:"Doe"};

Aim: Create a simple Program to build the Calculator in JavaScript using with HTML and CSS web languages.

Source Code:

```
<!DOCTYPE html>
<html lang = "en">
<head>
<title> JavaScript Calculator </title>

<style>
h1 {
        text-align: center;
        padding: 23px;
        background-color: skyblue;
        color: white;
        }

#clear{
width: 270px;
border: 3px solid gray;
        border-radius: 3px;
        padding: 20px;
        background-color: red;
}

.formstyle
{
width: 300px;
height: 530px;
margin: auto;
border: 3px solid skyblue;
border-radius: 5px;
padding: 20px;
}
```

```
input
{
width: 20px;
background-color: green;
color: white;
border: 3px solid gray;
          border-radius: 5px;
          padding: 26px;
          margin: 5px;
          font-size: 15px;
}

#calc{
width: 250px;
border: 5px solid black;
          border-radius: 3px;
          padding: 20px;
          margin: auto;
}

</style>

</head>
<body>
<h1> Calculator Program in JavaScript </h1>
<div class= "formstyle">
<form name = "form1">

          <!-- This input box shows the button pressed by the user in calculator. -->
  <input id = "calc" type ="text" name = "answer"> <br> <br>
  <!-- Display the calculator button on the screen. -->
  <!-- onclick() function display the number prsses by the user. -->
  <input type = "button" value = "1" onclick = "form1.answer.value += '1' ">
  <input type = "button" value = "2" onclick = "form1.answer.value += '2' ">
  <input type = "button" value = "3" onclick = "form1.answer.value += '3' ">
   <input type = "button" value = "+" onclick = "form1.answer.value += '+' ">
  <br> <br>

  <input type = "button" value = "4" onclick = "form1.answer.value += '4' ">
  <input type = "button" value = "5" onclick = "form1.answer.value += '5' ">
  <input type = "button" value = "6" onclick = "form1.answer.value += '6' ">
  <input type = "button" value = "-" onclick = "form1.answer.value += '-' ">
  <br> <br>

  <input type = "button" value = "7" onclick = "form1.answer.value += '7' ">
  <input type = "button" value = "8" onclick = "form1.answer.value += '8' ">
  <input type = "button" value = "9" onclick = "form1.answer.value += '9' ">
  <input type = "button" value = "*" onclick = "form1.answer.value += '*' ">
  <br> <br>

  <input type = "button" value = "/" onclick = "form1.answer.value += '/' ">
  <input type = "button" value = "0" onclick = "form1.answer.value += '0' ">
    <input type = "button" value = "." onclick = "form1.answer.value += '.' ">
          <!-- When we click on the '=' button, the onclick() shows the sum results on the
calculator screen. -->
  <input type = "button" value = "=" onclick = "form1.answer.value = eval(form1.answer.value)
```

```
">
  <br>
  <!-- Display the Cancel button and erase all data entered by the user. -->
  <input type = "button" value = "Clear All" onclick = "form1.answer.value = ' ' " id= "clear" >
  <br>

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

My HTML Web Page Project: A Report

1. **Introduction:**

This report details my experience in designing a few web pages using basic HTML tags. The goal of this project was to gain a foundational understanding of web development and learn how to structure content using HTML. I aimed to create simple, functional web pages that demonstrated my understanding of key HTML elements.

2. **Project Objectives:**

- Learn and apply basic HTML tags, including headings, paragraphs, lists, images, and links.

Let's delve into the specific objective: "Learn and apply basic HTML tags, including headings, paragraphs, lists, images, and links," and understand its significance in the context of the student's HTML project.

Significance of These Specific HTML Tags:

These tags represent the fundamental building blocks of most web pages. Mastering them is essential for any beginner in HTML.

- **Headings (<h1> to <h6>):**
  - Purpose: Structure content hierarchically, indicating the importance of different sections.
  - Significance: Essential for creating clear and organized web pages, improving readability and accessibility.
  - Application: Used to create titles and subtitles for each web page.
- **Paragraphs (<p>):**
  - Purpose: Define blocks of text.
  - Significance: The most basic way to display textual content, crucial for conveying information.
  - Application: Used to write descriptions, introductions, and other textual content.
- **Lists (<ul>, <ol>, <li>):**
  - Purpose: Organize information into lists, either unordered (bullet points) or ordered (numbered).
  - Significance: Enhance readability and make information easier to digest.
  - Application: Used to list hobbies, features, or any other set of related items.
- **Images (<img>):**
  - Purpose: Embed images into web pages.
  - Significance: Add visual appeal and context to web content.
  - Application: Used to display profile pictures, travel photos, or any other relevant images.
- **Links (<a>):**

    - Purpose: Create hyperlinks to other web pages or resources.
    - Significance: Connect web pages and enable navigation, a core feature of the web.
    - Application: Used to link to external websites, or other html pages that the student created.

Why These Tags Are "Basic":

- They are widely used in virtually every web page.
- They are relatively simple to understand and implement.
- They form the foundation for more complex HTML structures.

Learning and Application:

The student's objective implies a process of:

- **Learning:** Understanding the syntax and purpose of each tag.
- **Application:** Using these tags in their HTML code to create functional web pages.
- **Demonstration:** Showing that they can use these tags correctly.

- Understand the structure of an HTML document.

Understanding the structure of an HTML document is fundamental to creating any web page. The objective "Understand the structure of an HTML document" in the report signifies that the student aimed to grasp the core organizational principles of HTML. Let's break down what this entails:

Key Components of HTML Document Structure:

- **<!DOCTYPE html> Declaration:**
    - Purpose: This declaration informs the web browser about the HTML version being used.
    - Significance: Ensures that the browser renders the page correctly.
    - Placement: It's the very first line of code in an HTML document.
- **<html> Tag:**
    - Purpose: The root element of an HTML page. It encapsulates all other HTML elements.
    - Significance: Defines the start and end of the HTML document.
    - Placement: It contains the <head> and <body> tags.
- **<head> Tag:**
    - Purpose: Contains metadata about the HTML document. Metadata is data about data.
    - Significance: Provides information that is not directly displayed on the web page, such as the title, character set, and links to stylesheets.
    - Key elements within <head>:
        - **<title>:** Defines the title of the web page, displayed in the browser's title bar or tab.
        - **<meta>:** Provides various metadata, such as character encoding (charset), viewport settings, and keywords.
        - **<link>:** Links to external resources, such as stylesheets (CSS).
- **<body> Tag:**
    - Purpose: Contains the visible content of the HTML document.
    - Significance: Represents the main content that users see in the browser window.
    - Content: Includes all the elements that make up the web page's content, such as headings, paragraphs, images, links, and lists.

Why Understanding Structure Is Important:

- **Organization:** It ensures that the HTML code is well-organized and easy to understand.
- **Browser Compatibility:** It helps browsers interpret and render the web page correctly.
- **Maintainability:** It makes it easier to modify and update the web page.
- **Accessibility:** Proper structure is crucial for accessibility, allowing screen readers and other assistive technologies to interpret the content.
- **SEO (Search Engine Optimization):** Search engines rely on the structure of HTML to understand the content of a web page.

What the Student's Objective Implies:

- The student learned the purpose and placement of each of these core tags.
- The student understood the hierarchical relationship between these tags.
- The student was able to create valid HTML documents with the correct structure.

- Create multiple web pages with different content.

Significance of Creating Multiple Web Pages:

- **Reinforces Learning:**
  - Creating multiple pages requires the student to apply HTML concepts repeatedly, solidifying their understanding.
  - It allows them to experiment with different combinations of tags and content.
- **Demonstrates Versatility:**
  - It shows that the student can adapt their skills to create different types of web pages, rather than just a single, static page.
  - It promotes creativity in content development.
- **Simulates Real-World Web Development:**
  - Most websites consist of multiple pages, so this objective provides a realistic introduction to web development practices.
  - It introduces the concept of site navigation and linking between pages (even if just linking to external sites in this case).
- **Enhances Problem-Solving Skills:**
  - Each page presents unique challenges, requiring the student to troubleshoot and find solutions.
  - It helps them develop their ability to plan and organize content.

"Different Content" Implies:

- **Varied Topics:**
  - The student chose different subjects for each page (personal profile, hobbies, travel destinations), demonstrating their ability to work with diverse content.
- **Different Tag Usage:**
  - Each page likely used a slightly different combination of HTML tags, depending on the content. For example, the hobbies page heavily used lists, while the travel page focused on images and links.
- **Content Planning:**
  - It shows that the student had to plan the content of each page, and how to display that content.

What This Objective Demonstrates:

- **Practical Application:**
  - The student can apply their theoretical knowledge of HTML to create tangible web pages.
- **Organization and Planning:**
  - The student can structure and organize content for different purposes.
- **Independent Work:**
  - The student can work independently to create multiple web pages from start to finish.
- **Basic Web Site Structure:**
  - Even without internal linking, the student began to understand how a website is comprised of multiple pages.

- Gain hands-on experience in writing and testing HTML code.

Importance of Hands-On Experience:

- **Reinforces Theoretical Knowledge:**
  - Reading about HTML tags is different from actually using them. Hands-on experience solidifies understanding and reveals nuances that may not be apparent in theory.
- **Develops Practical Skills:**
  - Writing and testing code builds essential skills like:
    - Syntax accuracy.
    - Debugging.
    - Problem-solving.
    - Attention to detail.
- **Encourages Active Learning:**
  - Hands-on experience promotes active engagement with the material, leading to deeper learning and retention.
- **Builds Confidence:**
  - Successfully writing and testing code builds confidence in one's ability to create web pages.
- **Real-World Application:**
  - This is how web pages are actually created. There is no substitute for actually creating the code.

"Writing HTML Code" Implies:

- **Text Editor Usage:**
  - The student had to use a text editor to write the HTML code, which is a fundamental skill for any web developer.
- **Syntax Application:**
  - The student had to correctly apply HTML syntax, including tag names, attributes, and nesting.
- **Code Organization:**
  - The student had to organize the code in a logical and readable manner.

"Testing HTML Code" Implies:

- **Browser Usage:**
  - The student had to use a web browser to view and test the HTML code.
- **Debugging:**

    - The student had to identify and fix errors in the code, such as syntax errors or incorrect tag usage.
- **Visual Verification:**
    - The student had to visually verify that the web page was displayed as intended.
- **Iterative Process:**
    - Testing is not a one time activity. The student likely wrote code, tested, found errors, rewrote code, and retested many times.
- **Developer Tools:**
    - It is very likely that the student used the developer tools built into the browser. These tools are essential for debugging and inspecting HTML.

Why This Objective Is Crucial:

- **Practical Application:**
    - It transforms theoretical knowledge into practical skills.
- **Problem-Solving:**
    - It fosters the ability to identify and solve problems, a crucial skill for any programmer.
- **Self-Reliance:**
    - It empowers students to create and test their own web pages.

3. **Methodology:**

- **Learning HTML Basics**

Importance of Learning HTML Basics:

- **Foundation for Understanding:**
    - HTML is the language that structures web content. Without a solid understanding of its core concepts, building web pages would be impossible.
    - Learning the basics provides the necessary foundation for understanding more advanced HTML features and related technologies like CSS and JavaScript.
- **Syntax and Structure:**
    - It involves learning the correct syntax for HTML tags, attributes, and document structure. This is essential for writing valid and functional code.
    - Understanding the hierarchical structure of HTML documents (e.g., the relationship between <html>, <head>, and <body>) is vital for creating well-organized and maintainable web pages.
- **Essential Tag Familiarity:**
    - It entails becoming familiar with the most commonly used HTML tags and their purposes. This includes tags for headings, paragraphs, lists, images, links, and more.
    - Knowing how and when to use these tags is fundamental to creating basic web page layouts and content.
- **Building a Mental Model:**
    - Learning the basics helps students build a mental model of how web pages are constructed. This mental model is crucial for problem-solving and debugging.

What "Learning HTML Basics" Entails:

- **Resource Utilization:**
    - The student mentioned using online tutorials, documentation (like MDN Web Docs), and educational videos. This indicates a proactive approach to learning.

    - Using a variety of resources is beneficial for gaining a comprehensive understanding of HTML.
- **Conceptual Understanding:**
    - It's not just about memorizing tags; it's about understanding their purpose and how they work together.
    - Understanding the semantic meaning of tags is essential for creating accessible and well-structured web pages.
- **Syntax Mastery:**
    - Learning the correct syntax for HTML tags and attributes is crucial for writing valid code.
    - Paying attention to details like tag nesting and closing tags is essential for avoiding errors.
- **Tag Purpose:**
    - Understanding what each tag is for, and when to use it. For instance, knowing the difference between a header tag, and a paragraph tag.

Why This Step Is Essential:

- **Prevents Errors:**
    - A solid understanding of HTML basics helps prevent common errors that can arise from incorrect syntax or tag usage.
- **Facilitates Further Learning:**
    - A strong foundation in HTML makes it easier to learn more advanced web development concepts.
- **Increases Efficiency:**
    - Knowing the basics allows students to write code more efficiently and effectively.

- **Planning and Design**

Importance of Planning and Design:

- **Structured Approach:**
    - Planning ensures a structured and organized approach to web development, preventing haphazard coding and potential errors.
    - It helps to define the scope of the project and ensure that all objectives are met.
- **Content Organization:**
    - Planning involves organizing the content of each web page, determining the information to be included and how it will be presented.
    - It helps to create a logical flow of information and enhance the user experience.
- **Layout and Structure:**
    - Design involves visualizing the layout and structure of the web pages, including the placement of headings, paragraphs, images, and links.
    - It helps to create visually appealing and user-friendly web pages.
- **Purposeful Development:**
    - Planning gives each page a purpose. Instead of just creating random html, it gives the student a goal for each page.
- **Efficiency:**
    - Planning saves time. By thinking through the structure beforehand, the student will spend less time rewriting code.

What "Planning and Design" Entails:

- **Content Definition:**
  - The student decided on the content for each web page: a personal profile, hobbies, and travel destinations.
  - This demonstrates an ability to define the purpose and scope of each page.
- **Layout Visualization:**
  - The student visualized how the content would be arranged on each page, considering the placement of different HTML elements.
  - Even if it was a mental visualization, this is a very important step.
- **Tag Selection:**
  - The student implicitly selected the appropriate HTML tags for each type of content, such as headings for titles, paragraphs for text, and images for visual elements.
  - This shows an ability to apply the knowledge learned in the "Learning HTML Basics" section.
- **Site Structure (Even if Basic):**
  - By creating multiple pages, the student began to plan a small site structure. Even if the pages were not internally linked, the student was thinking about multiple pages.

Why This Step Is Essential:

- **Clarity and Focus:**
  - Planning provides clarity and focus, ensuring that the development process is aligned with the project's objectives.
- **Reduced Errors:**
  - Careful planning can help to identify potential problems early on, reducing the likelihood of errors during coding.
- **Improved User Experience:**
  - Thoughtful design can enhance the user experience by creating visually appealing and easy-to-navigate web pages.
- **Professionalism:**
  - Planning is a core part of professional web development.

- **Coding**

Importance of the Coding Stage:

- **Implementation of Design:**
  - This stage is where the student brings their planned design to life by writing the actual HTML code.
  - It's the transformation of a conceptual layout into a functional web page.
- **Application of Knowledge:**
  - The student applies their knowledge of HTML tags, attributes, and document structure to create the desired web page content and layout.
  - It's a practical demonstration of their understanding of HTML syntax and semantics.
- **Development of Technical Skills:**
  - The coding stage allows the student to develop essential technical skills, such as:
    - Accurate typing and syntax application.
    - Code organization and formatting.
    - Problem-solving and debugging.
- **Creation of a Tangible Product:**

    - The coding stage results in the creation of actual HTML files that can be viewed in a web browser.
    - It provides a sense of accomplishment and tangible evidence of the student's work.

What "Coding" Entails:

- **Text Editor Usage:**
    - The student used a text editor to write the HTML code, which is a fundamental tool for web development.
    - This demonstrates proficiency in using a basic code-writing environment.
- **Tag Implementation:**
    - The student correctly implemented the HTML tags that they had planned for, such as headings, paragraphs, lists, images, and links.
    - This shows their ability to translate design requirements into HTML code.
- **Attribute Usage:**
    - The student used appropriate attributes for each tag, such as the src attribute for images and the href attribute for links.
    - The student also used the alt attribute, and width attribute, showing good understanding of image tags.
- **Document Structure:**
    - The student adhered to the correct HTML document structure, including the <!DOCTYPE html> declaration, <html>, <head>, and <body> tags.
    - This ensures that the web pages are valid and can be rendered correctly by web browsers.
- **Code Organization and Comments:**
    - The report mentioned the use of comments. This is a very good habit. It shows the student is thinking about readability, and maintainability.
    - Proper indentation is also a key component of code organization.

Why This Stage Is Essential:

- **Realization of Design:**
    - It's the stage where the planned design is transformed into a functional web page.
- **Skill Development:**
    - It provides hands-on experience in writing HTML code, which is essential for web development.
- **Problem-Solving:**
    - It allows the student to identify and resolve coding errors, which is a crucial skill for any programmer.
- **Building Confidence:**
    - Successfully writing functioning code builds confidence in the students abilities.

- **Testing and Debugging**

Importance of Testing and Debugging:

- **Ensuring Functionality:**
    - Testing ensures that the web pages function as intended, displaying the correct content and layout.
    - It verifies that all HTML elements are rendered properly by the browser.
- **Identifying and Correcting Errors:**
    - Debugging involves identifying and fixing errors in the HTML code, such as syntax errors, incorrect tag usage, and layout issues.

  - It helps to improve the quality and reliability of the web pages.
- **Improving User Experience:**
  - Testing helps to identify any usability issues that may affect the user experience, such as broken links or distorted images.
  - It ensures that the web pages are user-friendly and accessible.
- **Developing Problem-Solving Skills:**
  - Debugging requires analytical and problem-solving skills, which are essential for any web developer.
  - It teaches students how to systematically identify and resolve errors.
- **Browser compatibility:**
  - Testing across multiple browsers helps to insure that the web pages will display correctly for the largest audience possible.

What "Testing and Debugging" Entails:

- **Browser Usage:**
  - The student used a web browser to view and test the HTML files, which is a fundamental tool for web development.
  - This demonstrates proficiency in using a browser to inspect web pages.
- **Visual Inspection:**
  - The student visually inspected the web pages to ensure that the content and layout were displayed correctly.
  - This involves checking for any visual errors, such as misaligned elements or broken images.
- **Developer Tools Usage:**
  - The report mentioned the use of the browser's developer tools, which is a crucial aspect of debugging.
  - Developer tools provide valuable information about the HTML structure, CSS styles, and JavaScript code, allowing developers to identify and fix errors.
- **Error Identification:**
  - The student identified and corrected errors in the HTML code, such as incorrect file paths for images.
  - The student also mentioned learning about link targets.
- **Iterative Process:**
  - Testing and debugging is an iterative process, involving repeated cycles of testing, error identification, and correction.
  - This demonstrates the student's persistence and attention to detail.

Why This Stage Is Essential:

- **Quality Assurance:**
  - It ensures that the web pages are of high quality and meet the project's requirements.
- **Error Prevention:**
  - It helps to identify and prevent potential errors that could affect the user experience.
- **Skill Development:**
  - It develops essential debugging and problem-solving skills, which are crucial for any web developer.
- **Professionalism:**
  - Testing and debugging is a core part of professional web development.

- **Refinement:** After initial testing, I made adjustments to the code to improve the appearance and functionality of the web pages.

Importance of Refinement:

- **Iterative Improvement:**
  - Refinement acknowledges that the initial version of the web pages may not be perfect and that improvements can be made.
  - It highlights the iterative nature of web development, where continuous improvement is valued.
- **Attention to Detail:**
  - Refinement demonstrates attention to detail and a commitment to producing high-quality work.
  - It involves fine-tuning the code and layout to enhance the user experience.
- **Enhanced User Experience:**
  - Refinement can lead to improvements in the user experience, such as better layout, clearer content, and more intuitive navigation.
  - It ensures that the web pages are user-friendly and visually appealing.
- **Skill Development:**
  - Refinement allows the student to further develop their problem-solving and debugging skills.
  - It provides an opportunity to apply what they have learned and refine their coding techniques.
- **Professionalism:**
  - Showing that you are willing to refine your work is a very professional trait.

What "Refinement" Entails:

- **Code Optimization:**
  - Refinement may involve optimizing the HTML code for better performance or readability.
  - This could include removing unnecessary code, improving code formatting, or adding comments.
- **Layout Adjustments:**
  - Refinement may involve adjusting the layout of the web pages to improve their visual appeal or usability.
  - This could include changing the placement of elements, adjusting spacing, or modifying colors.
- **Content Enhancements:**
  - Refinement may involve enhancing the content of the web pages, such as adding more information, clarifying text, or improving image quality.
  - Making sure that all alt text is filled in, and that the site is as accessible as possible.
- **Bug Fixes:**
  - This is a part of refinement. After the initial testing, more bugs may be found, and then fixed.

Why This Stage Is Essential:

- **Quality Enhancement:**
  - It ensures that the web pages are of the highest possible quality.
- **User Satisfaction:**
  - It improves the user experience and increases user satisfaction.
- **Continuous Learning:**
  - It reinforces the importance of continuous learning and improvement in web development.
- **Professional Development:**

    - It helps the student to develop professional habits.

4. **Web Page Descriptions and Code Snippets:**

1. Personal Profile Page:

- This page includes my name, a brief introduction, and a profile picture.
- Key tags used: <h1>, <p>, <img>.
- Example code snippet:

```
HTML
<!DOCTYPE html>
<html>
<head>
  <title>My Profile</title>
</head>
<body>
  <h1>[Your Name]</h1>
  <img src="profile.jpg" alt="My Profile Picture" width="200">
  <p>Hello! I'm [Your Name], and I'm learning HTML.</p>
</body>
</html>
```

Purpose and Significance:

- **Introduction to Web Content:**
    - A personal profile page is a simple and effective way to introduce oneself online.
    - It serves as a starting point for creating more complex web content.
- **Application of Basic HTML:**
    - Creating a profile page allows the student to apply fundamental HTML tags, such as headings, paragraphs, and images.
    - It provides practical experience in structuring and presenting information on a web page.
- **Personalization and Creativity:**
    - A personal profile page allows the student to express their individuality and creativity.
    - It provides an opportunity to showcase their personality and interests.
- **Understanding of Image Embedding:**
    - The use of the <img> tag reinforces the student's understanding of how to embed images into a web page, and how to use the alt and width attributes.

Key Elements and Analysis:

- **Content:**
    - The page includes the student's name, a brief introduction, and a profile picture.
    - This demonstrates an ability to select and organize relevant information.
- **HTML Tags:**
    - The student used <h1> for the name, <p> for the introduction, and <img> for the profile picture.
    - This shows an understanding of how to use these tags to structure and present content.
- **Image Handling:**

    - The student included a profile picture, demonstrating an ability to embed images in a web page.
    - The student used the alt attribute, which is very important for accessibility.
- **Simplicity:**
    - The profile page is designed to be simple and straightforward, which is appropriate for a beginner's project.
    - This allows the student to focus on mastering the basics of HTML.

Example Code Analysis:

- **<!DOCTYPE html>:**
    - This declaration ensures that the browser renders the page correctly.
- **<html>, <head>, <body>:**
    - These tags provide the basic structure of the HTML document.
- **<title>:**
    - This tag sets the title of the web page, which is displayed in the browser's title bar or tab.
- **<h1>:**
    - This tag is used to create the main heading, which is the student's name.
- **<img>:**
    - This tag is used to display the profile picture, with the src attribute specifying the image file and the alt attribute providing alternative text.
- **<p>:**
    - This tag is used to create a paragraph for the student's introduction.

2. My Favorite Hobbies Page:

- This page lists my favorite hobbies using an unordered list.
- Key tags used: <h2>, <ul>, <li>.
- Example code snippet:

HTML
```
<!DOCTYPE html>
<html>
<head>
  <title>My Hobbies</title>
</head>
<body>
  <h2>My Favorite Hobbies</h2>
  <ul>
    <li>Reading</li>
    <li>Playing video games</li>
    <li>Drawing</li>
    <li>Listening to music</li>
  </ul>
</body>
</html>
```

Purpose and Significance:

- **Organizing Information:**

    - This page demonstrates the use of lists to organize information in a clear and readable format. Lists are fundamental for presenting structured data on web pages.
- **Applying Different List Types:**
    - The student specifically used an unordered list (<ul>), indicating a grasp of different list types in HTML. They understand that unordered lists are ideal for items where the order doesn't matter.
- **Content Planning:**
    - Choosing to list hobbies shows the student thought about the content they wanted to display and how to structure it effectively.
- **Building on Previous Skills:**
    - This page builds upon the skills used in the profile page, reinforcing the use of headings (<h2>) and demonstrating the combination of different HTML elements.

Key Elements and Analysis:

- **Content:**
    - The page lists the student's favorite hobbies, which is a simple yet effective way to provide personal information.
- **HTML Tags:**
    - The student correctly used <h2> for the heading, <ul> for the unordered list, and <li> for each list item. This shows an understanding of how to structure lists in HTML.
- **Structure and Readability:**
    - The use of a list makes the information easy to read and scan, demonstrating an understanding of how to present content effectively.

Example Code Analysis:

- **<h2>:**
    - This tag creates a subheading for the page, providing a clear title for the list of hobbies.
- **<ul>:**
    - This tag defines the unordered list, creating a bullet-point list of items.
- **<li>:**
    - This tag defines each list item within the unordered list.

3. Places I Would Like to Visit Page:

- This page includes images of places I wish to travel to, with links to more information about each place.
- Key tags used: <h2>, <img>, <a>.
- Example code snippet:

```
HTML
<!DOCTYPE html>
<html>
<head>
  <title>Travel Destinations</title>
</head>
<body>
  <h2>Places I Want to Visit</h2>
  <a href="https://en.wikipedia.org/wiki/Paris">
    <img src="paris.jpg" alt="Paris" width="300">
  </a>
  <a href="https://en.wikipedia.org/wiki/Tokyo">
    <img src="tokyo.jpg" alt="Tokyo" width="300">
  </a>
</body>
</html>
```

Purpose and Significance:

- **Hyperlinks and Navigation:**
    - This page introduces the <a> tag for creating hyperlinks, a fundamental concept for web navigation.
    - It shows the student's ability to link their page to external resources, simulating a key aspect of website functionality.
- **Combining Images and Links:**
    - The student effectively combines images with hyperlinks, creating clickable images that lead to external pages. This demonstrates a deeper understanding of how to integrate different HTML elements.
- **Content Relevance:**
    - Choosing travel destinations as the topic shows the student can relate HTML skills to personal interests, making the project more engaging.
- **Expanding on Previous Skills:**
    - This page builds on previous skills by using headings (<h2>), images (<img>), and introduces a new core concept: linking.

Key Elements and Analysis:

- **Content:**
    - The page features images of places the student wants to visit, each linked to a relevant web page (in this case, Wikipedia entries).
- **HTML Tags:**
    - The student correctly uses <h2> for the heading, <img> for the images, and <a> for the hyperlinks, demonstrating an understanding of how to create clickable links.
- **Image as Links:**
    - Making the images themselves the links (instead of separate text links) enhances user experience and demonstrates a more sophisticated approach.

Example Code Analysis:

- **<a href="...">:**
    - This tag defines the hyperlink, with the href attribute specifying the destination URL.
- **Nesting <img> within <a>:**
    - Placing the <img> tag inside the <a> tag makes the image clickable, taking the user to the specified link.

5. **Challenges and Solutions:**

- **Image paths:** I initially had trouble displaying images because of incorrect file paths. I learned to use relative paths and ensure that the image files were in the correct location.

Significance of "Image Paths":

- **Understanding File Structures:**
    - This challenge forces the student to understand how files are organized within a file system.
    - It highlights the importance of knowing where image files are located relative to the HTML file.
- **Relative vs. Absolute Paths:**
    - The student learned about relative paths, which are crucial for creating portable web pages.
    - Relative paths allow web pages to function correctly even when moved to different directories or servers.
    - Understanding the difference between relative and absolute file paths is a foundational concept.
- **Debugging Skills:**
    - Troubleshooting image path errors develops valuable debugging skills.
    - It requires the student to carefully examine file paths, identify errors, and test solutions.
- **Visual Feedback:**
    - The lack of an image provides immediate visual feedback, making the error easy to identify.
    - This direct feedback loop is essential for learning and reinforcing concepts.
- **Real-World Relevance:**
    - Image path issues are common in web development, so this challenge provides a realistic learning experience.

Why This Challenge Is Important for Beginners:

- **Foundation for Web Development:**
    - Understanding file paths is a fundamental concept in web development.
    - It's essential for working with images, stylesheets, and other external resources.
- **Problem-Solving Skills:**
    - This challenge requires the student to think critically and systematically to identify and resolve the error.
- **Attention to Detail:**
    - Accurate file paths require attention to detail, which is a valuable skill for any programmer.
- **Learning by Doing:**
    - The student learned by doing, which is the most effective way to learn programming.

What the Student Learned:

- **Relative Paths:**
    - How to use relative paths to link to image files that are located in the same directory or in subdirectories.
- **File Organization:**
    - The importance of organizing files in a logical and consistent manner.
- **Debugging Techniques:**
- How to use visual inspection and other debugging techniques to identify and resolve file path errors.

- **Link target:** I learned how to open links in a new tab by using the target="_blank" attribute within the <a> tag.

Significance of "Link Target":

- **User Experience Control:**
  - The target attribute allows developers to control how links open, which directly impacts the user's browsing experience.
  - Knowing how to open links in new tabs or windows is crucial for maintaining the user's flow and preventing them from navigating away from the current page.
- **Understanding HTML Attributes:**
  - This challenge reinforces the importance of HTML attributes and how they modify the behavior of HTML elements.
  - It demonstrates that HTML tags are not just static elements but can be customized with attributes.
- **Web Navigation Concepts:**
  - Learning about link targets helps students understand fundamental web navigation concepts.
  - It teaches them how to create links that enhance the user's ability to explore related content without losing their place.
- **Browser Behavior:**
  - It gives the student a better understanding of how browsers handle links.
- **Professionalism:**
  - Using target="_blank" when linking to external sites is considered good practice.

What the Student Learned:

- **target="_blank" Attribute:**
  - The student learned how to use the target="_blank" attribute to open links in a new tab or window.
  - This is a common practice for linking to external websites to avoid disrupting the user's experience on the current site.
- **Other target Values:**
  - While the report specifically mentions _blank, it's likely the student also gained an awareness of other target values, such as _self, _parent, and _top, even if they didn't use them in this project.
- **User Interface Design:**
  - The student gained a small amount of understanding of user interface design. They learned that controlling the way a link opens is a part of creating a good user experience.

Why This Challenge Is Important:

- **Practical Application:**
  - The target attribute is a practical tool that is used frequently in web development.
- **User Experience:**
  - It helps students understand the importance of user experience and how to create links that are both functional and user-friendly.
- **Attribute usage:**
  - Reinforces how html attributes work.

- **Tag nesting:** ensuring all tags were correctly nested was challenging at first, but using proper indentation helped to solve this.

The "Tag nesting" challenge mentioned in the report is a fundamental concept in HTML and is crucial for creating well-formed and functional web pages. Let's explore its significance:

Significance of "Tag Nesting":

- **HTML Structure and Hierarchy:**
    - Tag nesting is essential for defining the hierarchical structure of an HTML document.
    - It ensures that elements are properly contained within their parent elements, creating a logical and organized structure.
- **Browser Rendering:**
    - Proper tag nesting is crucial for ensuring that web browsers can correctly interpret and render the HTML code.
    - Incorrect nesting can lead to unexpected behavior, layout issues, and even broken pages.
- **Code Readability and Maintainability:**
    - Correctly nested tags make the HTML code more readable and easier to maintain.
    - Proper indentation, as mentioned in the solution, is essential for visualizing the nesting structure and identifying errors.
- **Accessibility:**
    - Proper nesting is also important for accessibility, as screen readers and other assistive technologies rely on the HTML structure to interpret the content.
- **Avoiding Errors:**
    - Incorrect tag nesting is a very common source of errors. Learning to nest tags correctly early on prevents a lot of future debugging time.

What the Student Learned:

- **Hierarchical Relationships:**
    - The student learned how HTML elements are nested within each other, creating parent-child relationships.
- **Proper Indentation:**
    - The student learned how to use proper indentation to visualize the nesting structure and identify errors.
- **Syntax Accuracy:**
    - The student learned the importance of closing tags and ensuring that they are closed in the correct order.
- **Debugging:**
    - The student learned to debug by checking the tag nesting.

Why This Challenge Is Important:

- **Foundation for HTML Development:**
    - Tag nesting is a foundational concept that is essential for all HTML development.
- **Error Prevention:**
    - Mastering tag nesting helps to prevent common errors and improve the quality of HTML code.
- **Best Practices:**
    - Proper tag nesting is a best practice that is followed by professional web developers.
- **It is the base of the DOM:**
- The Document Object Model, which javascript uses, is based off of the html tag testing.

6. **Conclusion:**

This project provided a valuable introduction to HTML and web development. I successfully created three basic web pages and gained practical experience in using fundamental HTML tags. I learned the importance of planning, coding, testing, and debugging in the web development process.

Importance of the "Conclusion":

- **Summarizes Key Learnings:**
  - It provides a concise summary of the student's accomplishments and the knowledge they gained.
  - It reinforces the main objectives of the project and how they were achieved.
- **Reflects on the Learning Process:**
  - It allows the student to reflect on their learning journey and identify the most valuable aspects of the project.
  - It encourages self-assessment and highlights areas of growth.
- **Reinforces Key Concepts:**
  - It reiterates the importance of fundamental HTML concepts and their practical application.
  - It helps to solidify the student's understanding of the material.
- **Provides Closure:**
  - It brings the report to a satisfying conclusion, summarizing the project's outcomes and providing a sense of completion.
- **Sets the Stage for Future Learning:**
  - It often includes a section about future learning. This is very important.

7. **Future Improvements:**

- Learn and implement CSS to style the web pages and improve their appearance.

Significance of Learning and Implementing CSS:

- **Separation of Concerns:**
  - CSS (Cascading Style Sheets) allows for the separation of content (HTML) from presentation (CSS). This is a fundamental principle in web development, making code more organized and maintainable.
- **Enhanced Visual Appeal:**
  - CSS is the primary tool for styling web pages, enabling developers to control layout, colors, fonts, and other visual aspects.
  - It transforms basic HTML pages into visually appealing and engaging designs.
- **Improved User Experience:**
  - CSS can be used to create responsive designs that adapt to different screen sizes and devices, ensuring a consistent and enjoyable user experience.
  - It allows for refined control of spacing, alignment, and other visual elements that contribute to usability.
- **Increased Control over Presentation:**
  - CSS provides granular control over the appearance of HTML elements, allowing for precise styling and customization.
  - This level of control is not possible with HTML alone.
- **Efficiency and Reusability:**
  - CSS allows for the creation of reusable styles, reducing the amount of code needed and making it easier to maintain consistent styling across multiple pages.
  - One style sheet can control many html pages.
- **Modern Web Development:**

    - CSS is an integral part of modern web development, and proficiency in CSS is essential for creating professional-looking websites.

What Learning and Implementing CSS Entails:

- **CSS Syntax:**
    - Understanding CSS syntax, including selectors, properties, and values.
- **CSS Selectors:**
    - Learning how to target specific HTML elements using various CSS selectors (e.g., element selectors, class selectors, ID selectors).
- **CSS Properties:**
    - Becoming familiar with common CSS properties, such as color, font-size, background-color, margin, and padding.
- **CSS Box Model:**
    - Understanding the CSS box model, which defines the layout and spacing of HTML elements.
- **CSS Layout Techniques:**
    - Learning CSS layout techniques, such as flexbox and grid, for creating complex and responsive layouts.
- **External, Internal, and Inline Styles:**
    - Understanding the different ways that CSS can be applied to HTML.
- **Responsive Design:**
    - Learning how to use media queries to create web pages that adapt to different screen sizes.

Why This Is a Natural Next Step:

- **Builds Upon HTML Foundation:**
    - CSS complements HTML, allowing students to build upon their existing HTML knowledge and create more sophisticated web pages.
- **Enhances Project Outcomes:**
    - Implementing CSS would significantly enhance the visual appeal and user experience of the student's existing web pages.
- **Prepares for Advanced Web Development:**
    - Proficiency in CSS is essential for learning more advanced web development concepts, such as JavaScript frameworks and front-end development.

- Explore more advanced HTML tags and concepts

Significance of Exploring Advanced HTML:

- **Expanded Functionality:**
    - Advanced HTML tags and concepts unlock more sophisticated web page functionalities.
    - This allows for the creation of richer and more interactive web experiences.
- **Increased Versatility:**
    - A deeper understanding of HTML expands the range of web projects the student can undertake.
    - It enables them to create more complex and dynamic web pages.

- **Modern Web Development:**
    - While basic HTML is essential, modern web development often requires knowledge of more advanced features.
    - This includes semantic HTML, multimedia embedding, and form handling.
- **Improved Accessibility:**
    - Many advanced HTML concepts relate to semantic HTML, which is crucial for creating accessible web pages.
    - This ensures that web content is usable by a wider audience, including people with disabilities.
- **Better SEO:**
    - Semantic html helps with search engine optimization. Search engines use the tags to help determine the content of the page.

Examples of Advanced HTML Tags and Concepts:

- **Semantic HTML:**
    - Tags like <article>, <section>, <nav>, <aside>, <header>, <footer>, and <main> provide meaning to the structure of the web page.
    - This improves accessibility and SEO.
- **Multimedia Embedding:**
    - The <video> and <audio> tags allow for the embedding of multimedia content directly into web pages.
    - This provides more engaging and interactive user experiences.
- **Forms and Input Types:**
    - Advanced form elements and input types, such as <input type="date">, <input type="email">, and <textarea>, allow for more complex data collection.
    - This is crucial for creating interactive web applications.
- **Canvas and SVG:**
    - The <canvas> and <svg> tags allow for the creation of dynamic graphics and animations.
    - This opens up possibilities for interactive visualizations and games.
- **Web Storage:**
    - Learning about local storage and session storage.
- **Web Components:**
    - Learning about how to make reusable html components.

Why This Is a Logical Next Step:

- **Building on Foundational Knowledge:**
    - Exploring advanced HTML builds upon the student's existing knowledge of basic HTML tags and concepts.
- **Expanding Skill Set:**
    - It expands the student's skill set and prepares them for more complex web development projects.
- **Staying Current:**
    - It helps the student stay current with the latest web development trends and technologies.
- **Deeper Understanding:**
    - It helps the student gain a much deeper understanding of how the internet works.

- Learn how to use forms.

Significance of Learning Forms:

- **User Interaction:**
  - Forms enable user interaction, allowing users to provide input, submit data, and engage with web applications.
  - This is crucial for creating dynamic and interactive web experiences.
- **Data Collection:**
  - Forms are the primary method for collecting user data, such as contact information, survey responses, and feedback.
  - This data can be used for various purposes, such as marketing, research, and customer service.
- **Web Application Development:**
  - Forms are essential for building web applications, such as login pages, registration forms, and search interfaces.
  - They provide the foundation for creating interactive and data-driven web experiences.
- **E-commerce:**
  - Forms are critical for e-commerce websites, allowing users to enter shipping addresses, payment information, and product preferences.
  - They enable online transactions and shopping experiences.
- **Feedback and Communication:**
  - Forms facilitate feedback and communication between users and website owners.
  - They allow users to report issues, ask questions, and provide suggestions.

Key Concepts and Elements of HTML Forms:

- **<form> Tag:**
  - The <form> tag defines an HTML form for user input.
  - It specifies the action (where to send the form data) and the method (how to send the data).
- **Input Types:**
  - Various <input> types, such as text, password, email, checkbox, radio, date, and submit, allow for different types of user input.
  - Understanding the different input types is crucial for creating effective forms.
- **Labels:**
  - The <label> tag provides labels for form elements, improving accessibility and usability.
  - Labels help users understand the purpose of each input field.
- **Text Areas:**
  - The <textarea> tag allows for multi-line text input, such as comments or messages.
- **Select Menus:**
  - The <select> and <option> tags create drop-down menus for selecting options.
- **Buttons:**
  - The <button> tag creates clickable buttons for submitting forms or performing other actions.
- **Form Validation:**
  - HTML5 provides built-in form validation attributes, such as required, pattern, and type, to ensure data integrity.
  - Learning about these attributes is essential for creating robust forms.
- **Form Accessibility:**
  - Understanding how to create accessible forms, using ARIA attributes and proper labeling.
- **Form Submission:**
  - Understanding how form data is submitted to the server and processed.

Why Learning Forms Is Essential:

- **Interactivity:**
  - It enables the creation of interactive web pages that engage users.
- **Data Collection:**
  - It provides a means to collect user data for various purposes.
- **Web Application Development:**
  - It's a fundamental skill for building web applications.
- **Practical Application:**
  - Forms are used extensively in real-world web development.
- **Essential for Back-end Integration:**
  - Forms are how the front-end interacts with the back-end.

- Learn about responsive web design.

Significance of Responsive Web Design:

- **Multi-Device Compatibility:**
  - Responsive design ensures that web pages adapt seamlessly to different screen sizes and devices, from desktops to smartphones and tablets.
  - This is essential for providing a consistent and optimal user experience across all platforms.
- **Improved User Experience:**
  - Responsive design enhances the user experience by eliminating the need for users to zoom, scroll horizontally, or struggle with poorly formatted content.
  - It makes web pages more accessible and user-friendly.
- **Increased Mobile Traffic:**
  - With the increasing prevalence of mobile devices, responsive design is crucial for reaching a wider audience.
  - It ensures that mobile users have a positive experience on your website.
- **SEO Benefits:**
  - Google prioritizes mobile-friendly websites in its search rankings.
  - Responsive design helps improve SEO by providing a single URL for all devices, simplifying indexing and avoiding duplicate content issues.
- **Future-Proofing:**
  - Responsive design is a future-proof approach to web development, as it anticipates the emergence of new devices and screen sizes.
  - It allows websites to adapt to future technologies without requiring major redesigns.
- **Cost-Effectiveness:**
  - Maintaining a single responsive website is more cost-effective than developing and maintaining separate mobile and desktop versions.

Key Concepts and Techniques in Responsive Web Design:

- **Media Queries:**
  - Media queries allow developers to apply different CSS styles based on screen size, device orientation, and other factors.
  - They are the foundation of responsive design, enabling websites to adapt to different screen sizes.
- **Flexible Grids:**
  - Flexible grids, often using CSS Grid or Flexbox, allow for the creation of layouts that adapt to different screen sizes.
  - They ensure that content flows and resizes appropriately.

- **Flexible Images:**
  - Flexible images, using techniques like max-width: 100%; and the <picture> element, ensure that images resize appropriately on different devices.
  - They prevent images from overflowing their containers and maintain image quality.
- **Viewport Meta Tag:**
  - The viewport meta tag (<meta name="viewport" ...>) is essential for controlling the viewport on mobile devices.
  - It ensures that web pages are displayed correctly on mobile screens.
- **Mobile-First Design:**
  - Mobile-first design involves designing for mobile devices first and then progressively enhancing the design for larger screens.
  - This approach ensures that the mobile experience is prioritized.
- **Testing and Debugging:**
  - Testing responsive designs on different devices and screen sizes is crucial for ensuring a consistent user experience.
  - Browser developer tools and online testing tools can be used for this purpose.

Why Learning Responsive Web Design Is Essential:

- **Modern Web Development:**
  - It's a fundamental skill for modern web development.
- **User Experience:**
  - It's crucial for providing a positive user experience across all devices.
- **SEO:**
  - It helps improve SEO and reach a wider audience.
- **Future-Proofing:**
  - It prepares websites for future technologies and devices.

- Learn Javascript.

Significance of Learning JavaScript:

- **Interactivity and Dynamic Content:**
  - JavaScript enables the creation of interactive web pages that respond to user actions.
  - It allows for dynamic content updates, animations, and other interactive elements.

- **Front-End Development:**
  - JavaScript is the primary language for front-end web development, responsible for the user interface and user experience.
  - It allows developers to create rich and engaging web applications.
- **Back-End Development (Node.js):**
  - With Node.js, JavaScript can also be used for back-end development, creating server-side applications.
  - This allows for full-stack JavaScript development, using a single language for both front-end and back-end.
- **Web Application Frameworks:**
  - JavaScript frameworks like React, Angular, and Vue.js are widely used for building complex web applications.
  - Learning JavaScript is essential for working with these frameworks.
- **Browser Manipulation:**

    - Javascript allows developers to manipulate the DOM (Document Object Model). This is how websites are changed in response to user interaction.
- **Increased Functionality:**
    - Javascript allows developers to add many features to websites that are impossible with html and css alone.

Key Concepts and Areas of JavaScript Learning:

- **Basic Syntax and Concepts:**
    - Variables, data types, operators, control flow (loops, conditionals), functions.
- **DOM Manipulation:**
    - Selecting, modifying, and creating HTML elements using JavaScript.
    - Event handling (responding to user actions).
- **Asynchronous JavaScript:**
    - Understanding asynchronous operations, such as AJAX requests and promises.
    - Working with APIs and fetching data from servers.
- **Object-Oriented Programming (OOP):**
    - Understanding OOP concepts in JavaScript, such as objects, classes, and inheritance.
- **ES6+ Features:**
    - Learning about modern JavaScript features, such as arrow functions, destructuring, and modules.
- **Browser APIs:**
    - Accessing browser APIs, such as the Geolocation API, Canvas API, and Web Storage API.
- **Debugging and Testing:**
    - Using browser developer tools and testing frameworks to debug and test JavaScript code.

Why Learning JavaScript Is Essential:

- **Modern Web Development:**
    - It's a fundamental skill for modern web development.
- **Interactivity and User Experience:**
    - It's crucial for creating interactive and engaging web experiences.
- **Full-Stack Development:**
    - It enables full-stack development with Node.js.
- **Career Opportunities:**
    - JavaScript developers are in high demand, offering numerous career opportunities.
- **Large Community:**
    - There is a very large online community that supports javascript developers.

8. **Attachments:**

- HTML files for each web page.

Significance of the HTML Files:

- **Tangible Evidence of Learning:**
    - These files are the concrete output of the student's work, demonstrating their ability to write and structure HTML code.
    - They provide tangible evidence of the student's learning progress.
- **Practical Application of Concepts:**

    - The files showcase the student's ability to apply HTML tags, attributes, and document structure in a real-world context.
    - They demonstrate the student's understanding of how to create functional web pages.
- **Demonstration of Problem-Solving:**
    - The files reflect the student's ability to debug and refine their code, as mentioned in the "Challenges and Solutions" and "Refinement" sections.
    - They show the student's ability to overcome obstacles and produce working web pages.
- **Basis for Further Development:**
    - These files serve as a foundation for further web development learning, such as adding CSS styles or JavaScript functionality.
    - They provide a starting point for more complex web projects.
- **Assessment Tool:**
    - These files are the primary method that the instructor will use to assess the students understanding of the html language.

What the HTML Files Should Contain:

- **Valid HTML Structure:**
    - Each file should adhere to the basic HTML document structure, including the <!DOCTYPE html> declaration, <html>, <head>, and <body> tags.
    - They should have properly nested tags and correct syntax.
- **Appropriate HTML Tags:**
    - The files should contain the appropriate HTML tags for the content of each web page, such as headings, paragraphs, lists, images, and links.
    - They should demonstrate the student's ability to select and use relevant HTML elements.
- **Clear and Organized Code:**
    - The code should be well-organized and easy to read, with proper indentation and comments.
    - This demonstrates the student's attention to detail and code maintainability.
- **Functional Content:**
    - The content of each web page should be functional and relevant to the project's objectives.
    - They should demonstrate the student's ability to create meaningful web content.
- **Correct File Paths:**
    - If the HTML files include images or other external resources, the file paths should be correct.
    - This demonstrates the student's understanding of file management.

Why These Files Are Essential:

- **Proof of Work:**
    - They provide concrete proof of the student's work and effort.
- **Demonstration of Skills:**
    - They showcase the student's HTML skills and knowledge.
- **Foundation for Future Learning:**
    - They serve as a foundation for further web development learning.
- **Assessment:**
    - They are the primary tool used to assess the students abilities.

- Image files used in the project.

Significance of the Image Files:

- **Complementing HTML Content:**
  - Images are essential for adding visual appeal and context to web pages.
  - They enhance the user experience and make web pages more engaging.
- **Demonstrating Image Handling:**
  - Including the image files demonstrates the student's ability to handle images in HTML, including embedding them using the <img> tag.
  - It also shows if the student understood how to use the alt attribute, and the width and height attributes.
- **Testing and Debugging:**
  - The image files are crucial for testing and debugging the web pages, particularly regarding image paths and display.
  - They allow the student to verify that the images are displayed correctly in the browser.
- **Content Relevance:**
  - The images should be relevant to the content of the web pages, such as the profile picture in the personal profile page or the travel destination images.
  - This demonstrates the student's ability to integrate images into their web content effectively.
- **Completing the Project:**
  - Without the image files, the web pages would be incomplete and would not accurately reflect the student's work.
  - The image files are an integral part of the project deliverables.

What the Image Files Should Represent:

- **Appropriate Formats:**
  - The image files should be in web-friendly formats, such as JPEG, PNG, or GIF.
  - This ensures that the images are displayed correctly in web browsers.
- **Relevant Content:**
  - The images should be relevant to the content of the web pages, enhancing the overall message and visual appeal.
- **Proper Naming Conventions:**
  - The image files should have descriptive and consistent file names.
  - This makes it easier to manage and organize the files.
- **Correct Dimensions:**
  - The images should have appropriate dimensions and resolutions for web display.
  - This ensures that the images are displayed correctly without distortion or excessive file sizes.
- **Correct Location:**
  - The image files should be located in the correct directory relative to the HTML files.
  - This ensures that the images are displayed correctly in the browser.

Why These Files Are Essential:

- **Visual Representation:**
  - They provide a visual representation of the student's web pages.
- **Completeness:**
  - They complete the project deliverables and ensure that the web pages are fully functional.
- **Testing:**

    - They are necessary for testing and debugging the web pages.
- **Demonstration:**
    - They demonstrate the student's ability to work with images in HTML.
- **Real world simulation:**
    - Websites use images, so this is a simulation of real world website creation.