**ABOUT THE COMPANY:**

Mevi Technology Company stands as a beacon of technological innovation in today's dynamic business environment. This comprehensive report provides an in-depth analysis and insight into the core facets of Mevi Technology, shedding light on its origins, key technological advancements, corporate culture, and future prospects.

Mevi Technology Company, established in the year of 2022 in Tumakuru has emerged as a trailblazer in the realm of IT sector particularly for students who are struggling to emerge to the Technology . Headquartered in Tumkur Karnataka, the company has rapidly grown from its humble beginnings to become a formidable player in the global technology landscape.

An analysis of Mevi Technology's market presence and global impact is crucial for understanding the company's position in the competitive landscape. This section provides an overview of Mevi's market share, strategic partnerships, and the geographical regions where it has made significant inroads.



**Chapter 1:**

**Introduction to HTML**

**HTML (Hypertext Markup Language)**

HTML serves as the foundation of web development, providing the structure and content of web pages. As a front-end developer, proficiency in HTML is essential for creating well-structured and accessible websites. Throughout my internship, I utilized HTML extensively to develop various web pages and user interfaces.

**Role and Importance:**

HTML, as a markup language, allows developers to organize and structure content on web pages using tags and elements. It provides the basic building blocks for creating web documents and enables seamless integration with other technologies like CSS and JavaScript. Some key aspects of HTML include:

**Semantic Markup:** HTML offers a wide range of semantic elements that convey the meaning and purpose of content within a web page. By using semantic tags such as <header>, <nav>, <main>, <article>, <section>, and <footer>, developers can create more meaningful and accessible web pages for users and search engines alike.

**Content Structure:** HTML facilitates the organization of content hierarchically, allowing developers to create clear and logical structures for web pages. This not only improves the readability of the code but also enhances the user experience by making the content easier to navigate and understand.

**Accessibility:** Properly structured HTML ensures that web content is accessible to users with disabilities, including those using assistive technologies such as screen readers. By adhering to accessibility standards and guidelines, developers can ensure that their websites are inclusive and usable by all individuals, regardless of their abilities.

Examples:

During my internship, I utilized HTML to create the structural framework for various web pages and applications. Here are some examples of HTML code snippets that I implemented:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Sample Web Page</title>

</head>

<body>

<header>

<h1>Welcome to My Website</h1>

<nav>

<ul>

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

<li><a href="#services">Services</a></li>

<li><a href="#portfolio">Portfolio</a></li>

<li><a href="#contact">Contact</a></li></ul>

</nav>

</header>

<main>

<section id="about">

<h2>About Us</h2>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit....</p>

</section>

<section id="services">

<h2>Our Services</h2>

<ul> <li>Web Design</li>

<li>Front-end Development</li>

<li>UI/UX Design</li>

</ul>

</section>

<!-- Additional sections and content -->

</main>

<footer>

<p>&copy; 2024 My Company. All rights reserved.</p>

</footer>

</body>

</html>

**HTML tags:**

HTML (Hypertext Markup Language) tags are the building blocks of web pages. They are used to structure and format the content of a webpage, allowing web browsers to interpret and display the content correctly. HTML tags are enclosed in angle brackets (< >) and typically come in pairs, with an opening tag and a closing tag, although some tags are self-closing.

Here are some commonly used HTML tags:

<!DOCTYPE>: Specifies the document type and version of HTML being used.

<html>: Defines the root element of an HTML document.

<head>: Contains meta-information about the document, such as the title, character set, and links to external resources.

<title>: Sets the title of the HTML document, displayed in the browser's title bar or tab.

<body>: Contains the main content of the HTML document that will be displayed in the browser window. <h1> to <h6>: Defines headings of different levels, with <h1> being the highest level and <h6> being the lowest.

<p>: Defines a paragraph of text.

<a>: Creates a hyperlink to another web page or resource.

<img>: Embeds an image into the HTML document.

<div>: Defines a division or section of the document, often used for layout and styling purposes.

<span>: Defines an inline section of the document, often used for applying styles or scripting to a specific portion of text.

<ul> and <ol>: Define unordered and ordered lists, respectively.

<li>: Defines a list item within an ordered or unordered list.

<table>: Defines a table for organizing data into rows and columns.

<tr>: Defines a row within a table.

<td>: Defines a cell within a table row.

<form>: Defines an HTML form for collecting user input.

<input>: Defines an input control within a form, such as a text field, checkbox, or button.

<textarea>: Defines a multi-line text input control within a form.

<button>: Defines a clickable button within a form.

These are just a few examples of HTML tags used to structure and format content on web pages. There are many more tags available, each serving a specific purpose in defining the layout, content, and functionality of a webpage. Understanding how to use HTML tags effectively is essential for creating well-structured and visually appealing web pages.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Example HTML Page</title>

</head>

<body>

<h1>Welcome to My Website</h1>

<p>This is a paragraph of text.</p>

<h2>Lists</h2>

<ul><li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li> </ul>

<h2>Links</h2>

<a href="https://www.example.com">Visit Example.com</a>

<h2>Image</h2>

<img src="https://via.placeholder.com/150" alt="Placeholder Image">

<h2>Table</h2>

<table border="1">

<tr> <td>Row 1, Cell 1</td>

<td>Row 1, Cell 2</td> </tr><tr>

<td>Row 2, Cell 1</td> <td>Row 2, Cell 2</td> </tr> </table> <h2>Form</h2>

<form action="#" method="post">

<label for="name">Name:</label>

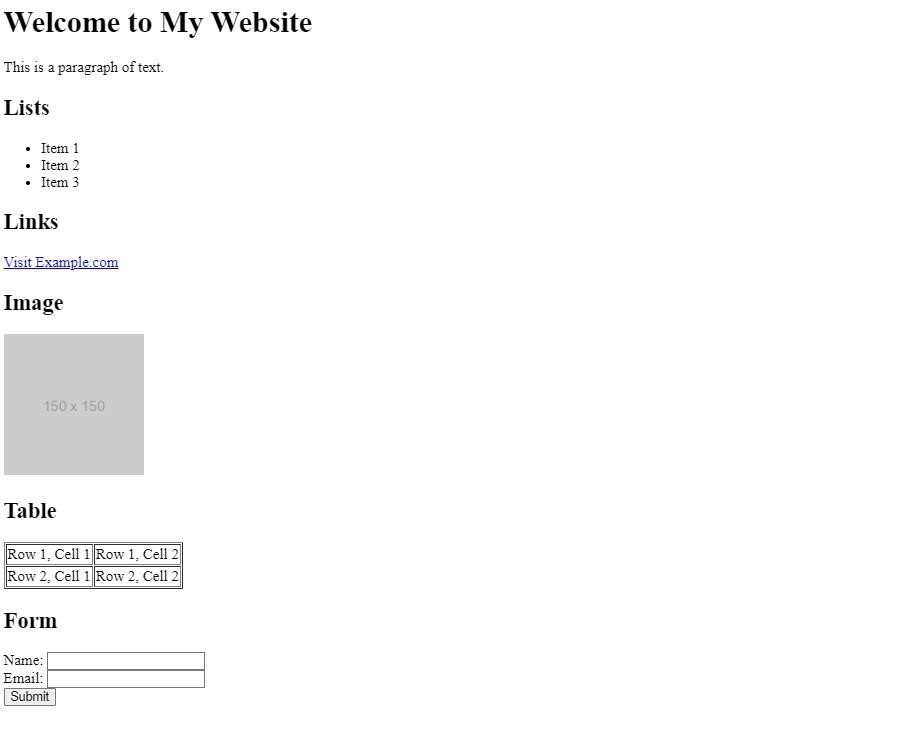
<input type="text" id="name" name="name"> <br>

<label for="email">Email:</label>

<input type="email" id="email" name="email"> <br>

<button type="submit">Submit</button> </form> </body>

</html>



**Chapter 2:**

**Introduction to CSS:**

CSS (Cascading Style Sheets) is a stylesheet language used to control the presentation and layout of HTML documents. It allows web developers to style and customize the appearance of web pages, including elements such as text, colors, fonts, spacing, and positioning. CSS works by selecting HTML elements and applying styling rules to them.

Here's an overview of CSS concepts and syntax:

Selectors: CSS selectors are used to target HTML elements that you want to style. Selectors can target elements based on their tag name, class, ID, attributes, or relationship with other elements.

Example:

/\* Targeting elements by tag name \*/ p

{ color: blue;}

/\* Targeting elements by class \*/

.highlight { background-color: yellow;}

/\* Targeting elements by ID \*/

#header { font-size: 24px; }

Properties and Values: CSS properties define the visual characteristics of elements, such as color, font size, width, height, etc. Each property is assigned a value that determines its specific appearance.

Example:

/\* Setting text color \*/ color: red;

/\* Changing font size \*/ font-size: 16px; /\* Adding padding \*/ padding: 10px; Comments: CSS supports comments to add notes or explanations within the stylesheet. Comments are ignored by the browser and are only visible to developers.

Example:

/\* This is a CSS comment \*/

Selectors and Declarations: CSS rules consist of selectors and declarations. Selectors target specific HTML elements, while declarations specify the styling properties and their values.

Example:

/\* Selector \*/ h1 { /\* Declaration \*/ color: blue; fontsize: 24px;

}

**Cascade and Specificity:** CSS follows a set of rules to determine which styles should be applied when there are conflicting styles targeting the same element. The cascade refers to the order of importance of styles, while specificity determines which rule takes precedence based on the specificity of the selector.

**Inheritance:** CSS properties can be inherited from parent elements to their children. This means that if a property is applied to a parent element, its children will inherit that property by default unless overridden.

**Units:** CSS supports various units for specifying lengths, such as pixels (px), percentages (%), ems (em), rems (rem), and others. These units determine the size and spacing of elements relative to the viewport or parent elements.

CSS can be included in HTML documents using the <style> element within the <head> section, or it can be linked externally using the <link> element. By combining HTML structure with CSS styling, developers can create visually appealing and responsive web pages.

**CSS properties along with a brief description of each:**

**color:** Sets the text color of an element. color: red; **font-family:** Defines the font family to be used for text.

font-family: Arial, sans-serif; **font-size:**

Sets the size of the font.

font-size: 16px; **font-weight:** Specifies the boldness of the font.

font-weight: bold; **text-align:** Aligns the text within its containing element.

text-align: center; **text-decoration:** Adds visual decoration to text, such as underlining or striking through. text-decoration: underline; **background-color:** Sets the background color of an element. background-color: #f0f0f0; **padding:** Adds space between the content of an element and its border. padding: 10px; **margin:** Sets the space outside an element's border. margin: 20px; **border:** Defines the border properties of an element. border: 1px solid black; **width:** Sets the width of an element. width: 300px;

**height:** Sets the height of an element. height: 200px; **display:** Determines how an element is displayed, such as block, inline, or inline-block.

display: block; **float:** Specifies whether an element should float to the left or right within its container. float: left; **position:** Sets the positioning method of an element, such as static, relative, absolute, or fixed. position: relative;

These are just a few examples of CSS properties that can be used to style and format HTML elements.

CSS provides a wide range of properties and values to customize the appearance and layout of web pages. Experimenting with different properties and values will help you achieve the desired design for your website.

Following is an Example:



**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Example HTML Page with CSS</title>

<style>

/\* CSS styles \*/ body { font-family: Arial, sans-serif; background-color: #f0f0f0; margin: 0; padding: 0;

}

.container { width: 80%; margin: 20px auto; padding: 20px; background-color: #fff; border-radius:

10px; box-shadow: 0 2px 5px rgba(0, 0,

0, 0.1);

}

h1 { color: #333; text-align: center;

margin-bottom: 20px;

} p { color:

#666; line-height:

1.6; margin-bottom:

15px;

}

.button { display: inline-block; padding: 10px 20px; backgroundcolor: #007bff;

color: #fff; text-decoration: none; border-radius: 5px; transition: background-color 0.3s ease;

}

.button:hover {

background-color: #0056b3;

}

.box { backgroundcolor: #f9f9f9; padding: 15px; border-radius: 5px; border: 1px solid #ccc; margin-bottom: 20px;

}

.alert { backgroundcolor: #f44336;

color: #fff; padding: 10px; border-radius: 5px; margin-bottom: 20px;

}

.alert.success {

background-color: #4CAF50;

}

.alert.warning {

background-color: #ff9800;

}

</style>

</head>

<body>

<div class="container">

<h1>Welcome to Example Website</h1>

<div class="box">

<p>This is a simple example HTML page with accompanying CSS styling.</p>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed accumsan, libero sit amet congue maximus, turpis ex fermentum nisi, non congue arcu metus id lacus.</p> <a href="#" class="button">Learn More</a>

</div>

<div class="alert">This is an error message.</div>

<div class="alert success">This is a success message.</div>

<div class="alert warning">This is a warning message.</div>

</div>

</body>

</html>

**Chapter 3:**

**Introduction to Java Script:**

JavaScript (JS) is a versatile programming language primarily used for web development. It enables interactive and dynamic features on web pages, such as user interactions, animations, form validation, and dynamic content updates. JavaScript code is executed by web browsers, making it a client-side scripting language.

Here's an overview of JavaScript and its key features:

**Syntax and Structure:**

JavaScript syntax is similar to other programming languages like Java and C. It uses variables, operators, functions, loops, and conditional statements to manipulate data and control program flow.

Example:

let message = "Hello, world!"; alert(message);

**Variables and Data Types:**

JavaScript variables are containers for storing data values. It supports various data types such as numbers, strings, booleans, arrays, objects, functions, and more.

Example:

let num = 10; let name = "John"; let isTrue = true; let fruits = ["apple", "banana", "orange"]; let person = { firstName: "John", lastName: "Doe" };

**Functions:**

Functions in JavaScript are reusable blocks of code that perform a specific task. They can be declared using the function keyword and can accept parameters and return values.

function greet(name) { return

"Hello, " + name + "!";

}

**DOM Manipulation:**

JavaScript interacts with the Document Object Model (DOM), which represents the structure of HTML documents. It allows developers to access, modify, and manipulate HTML elements and their attributes dynamically.

// Change the text content of an element with id="demo" document.getElementById("demo").innerHTML = "Hello, world!";

**Event Handling:**

JavaScript enables event-driven programming, allowing developers to respond to user actions such as clicks, mouse movements, keypresses, etc. Events can be attached to HTML elements using event listeners.

// Add a click event listener to a button element document.getElementById("myButton").addEventListener("click", function() { alert("Button clicked!");

});

**Asynchronous Programming:**

JavaScript supports asynchronous programming through callbacks, promises, and async/await syntax. Asynchronous operations allow tasks to execute concurrently without blocking the main thread, enhancing responsiveness and performance. // Example of fetching data asynchronously fetch('https://api.example.com/data')

.then(response => response.json())

.then(data => console.log(data))

.catch(error => console.error('Error:', error));

**Browser APIs:**

JavaScript provides access to various browser APIs (Application Programming Interfaces) that allow developers to perform tasks such as manipulating the browser history, accessing geolocation data, handling drag-and-drop interactions, etc.

JavaScript is a powerful language that is widely used not only for web development but also for serverside development (Node.js), mobile app development (React Native), game development, and more.

Mastering JavaScript is essential for building modern, interactive web applications.

**Document Object Model (DOM):**

The Document Object Model (DOM) is a programming interface provided by web browsers that represents the structure of an HTML or XML document as a hierarchical tree-like structure. It allows developers to interact with the content, structure, and style of a web page dynamically using JavaScript. The DOM provides methods and properties to access, manipulate, and update the elements and attributes of an HTML document.

Here's an overview of the DOM and its key concepts:

**Hierarchical Structure:**

The DOM represents an HTML document as a tree structure, with each element, attribute, and text node as a node in the tree. The root of the tree is represented by the document object, which contains all other nodes.

**Node Types:**

Nodes in the DOM tree can be of different types, including element nodes, text nodes, attribute nodes, comment nodes, etc. Element nodes represent HTML elements such as <div>, <p>, <a>, etc., while text nodes contain text content within elements.

**Accessing Elements:**

JavaScript provides various methods to access elements in the DOM tree, such as getElementById, getElementsByClassName, getElementsByTagName, and querySelector. These methods allow developers

to select elements based on their IDs, class names, tag names, or CSS selectors.

// Access an element by its ID let element = document.getElementById('myElement'); **Manipulating Elements:**

Once an element is selected, developers can manipulate its properties, attributes, and content using DOM methods and properties. Common manipulation tasks include changing text content, modifying CSS styles, adding or removing classes, and appending or removing child elements.

// Change the text content of an element element.textContent = 'New text content'; // Add a CSS class to an element element.classList.add('highlight'); // Append a new child element let newElement = document.createElement('div'); element.appendChild(newElement);

**Event Handling:**

The DOM allows developers to attach event listeners to elements to respond to user interactions such as clicks, mouse movements, keypresses, etc. Event listeners can be added using the addEventListener method and removed using the removeEventListener method.

// Add a click event listener to an element element.addEventListener('click', function(event) { console.log('Element clicked');

});

**Traversal and Navigation:**

Developers can traverse the DOM tree to navigate between elements, access parent, child, and sibling nodes, and search for specific elements within the document.

// Access the parent element let parentElement = element.parentNode; // Access the first child element let firstChild = element.firstChild; **// Access the next sibling element** let nextSibling = element.nextSibling;

The DOM is a powerful and essential part of web development, enabling dynamic and interactive web pages. Understanding how to manipulate the DOM using JavaScript is crucial for building modern web applications.

**Example Program:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>DOM Manipulation Example</title>

<style>

.highlight {

background-color: yellow;

}

</style>

</head>

<body>

<h1>DOM Manipulation Example</h1>

<p id="demo">Click the button to change the text color.</p>

<button id="changeColorButton">Change Color</button>

<script>

// Access the button element by its ID let button = document.getElementById('changeColorButton');

// Add a click event listener to the button button.addEventListener('click', function() { // Access the paragraph element by its ID let paragraph = document.getElementById('demo');

// Change the text color and add a CSS class paragraph.style.color = 'blue'; paragraph.classList.add('highlight');

// Update the text content of the button button.textContent = 'Color Changed';

});

</script>

</body>

</html>

In this example:

We have an HTML document containing a heading (<h1>), a paragraph (<p>), and a button (<button>).

The paragraph has an ID of "demo", and the button has an ID of "changeColorButton".

We use JavaScript to select the button element by its ID and add a click event listener to it.

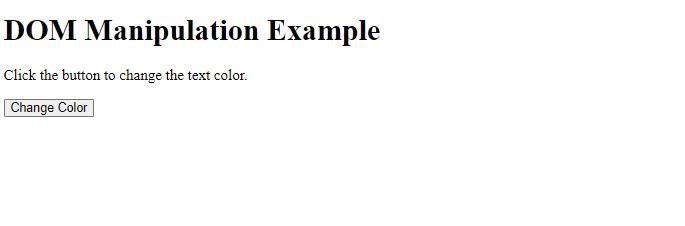
When the button is clicked, the event listener function is executed, which accesses the paragraph element by its ID.

We then change the text color of the paragraph to blue using the style.color property and add a CSS class "highlight" to it using the classList.add() method.

Finally, we update the text content of the button to indicate that the color has been changed.

You can save this code in an HTML file and open it in a web browser. When you click the "Change Color" button, the text color of the paragraph will change to blue, and the background color will turn yellow.

Additionally, the text content of the button will be updated to "Color Changed".





<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>DOM Manipulation Example</title>

<style>

.highlight { background-color: rgb(27, 117, 202);

}

</style>

</head>

<body>

<h1>DOM Manipulation Example</h1>

<p id="demo">Click the button to change the text color.</p>

<button id="changeColorButton">Change Color</button>

<script>

// Access the button element by its ID let button = document.getElementById('changeColorButton');

// Add a click event listener to the button button.addEventListener('click', function() { // Access the paragraph element by its ID let paragraph = document.getElementById('demo'); // Change the text color and add a CSS class paragraph.style.color = 'blue'; paragraph.classList.add('highlight'); // Update the text content of the button button.textContent =

'Color Changed';

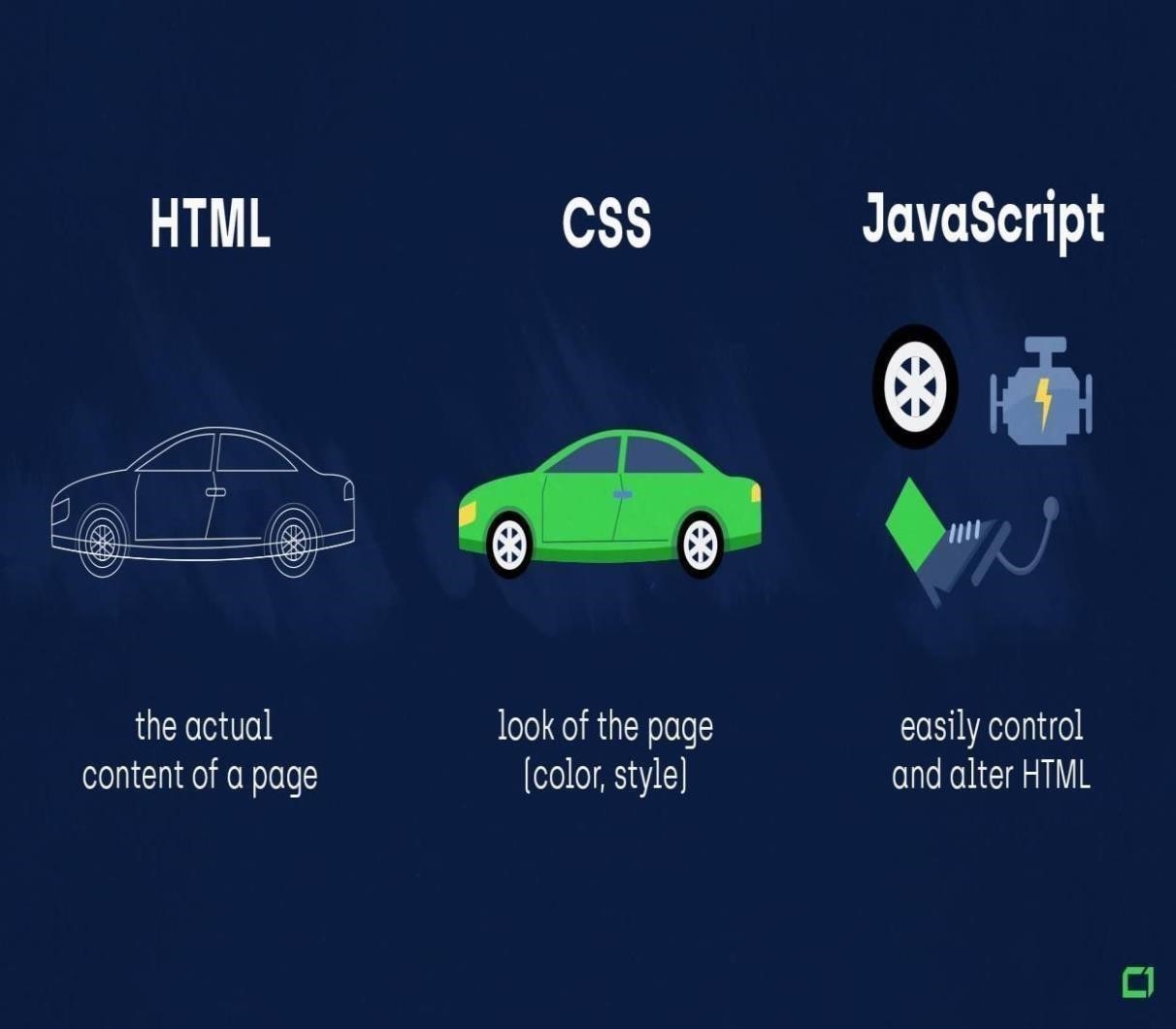
});

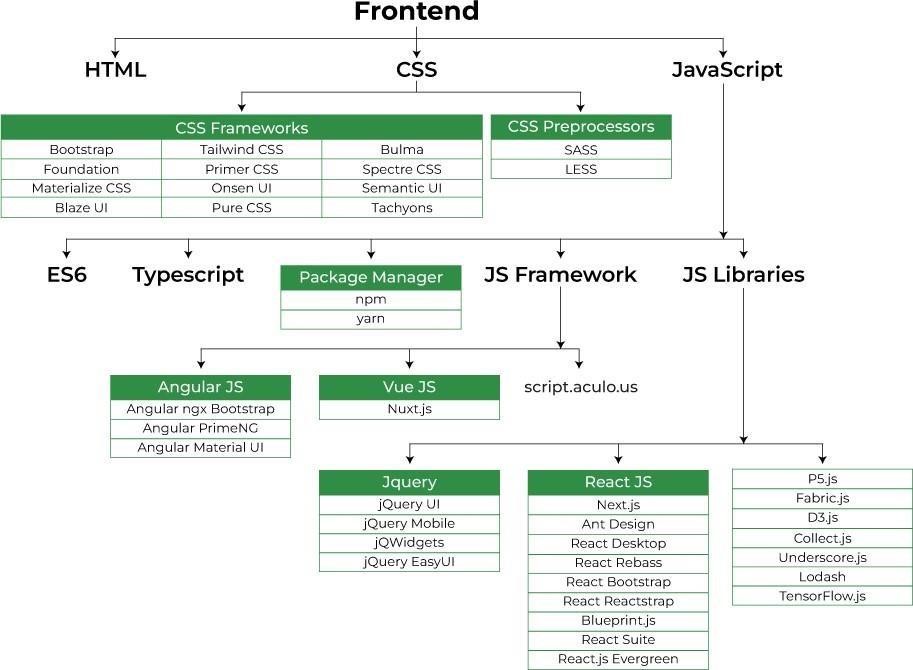
</script>

</body>

</html>

**The below picture shows difference between HTML, CSS, and JavaScript :**

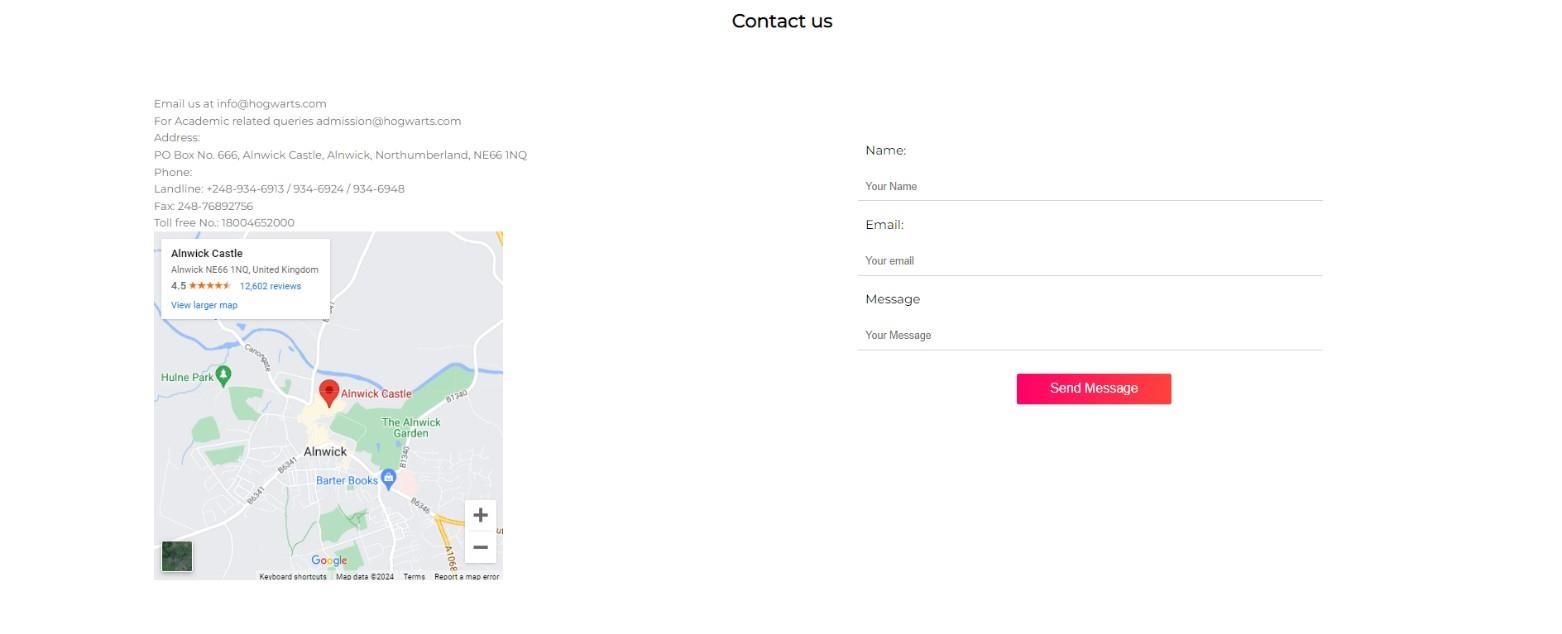
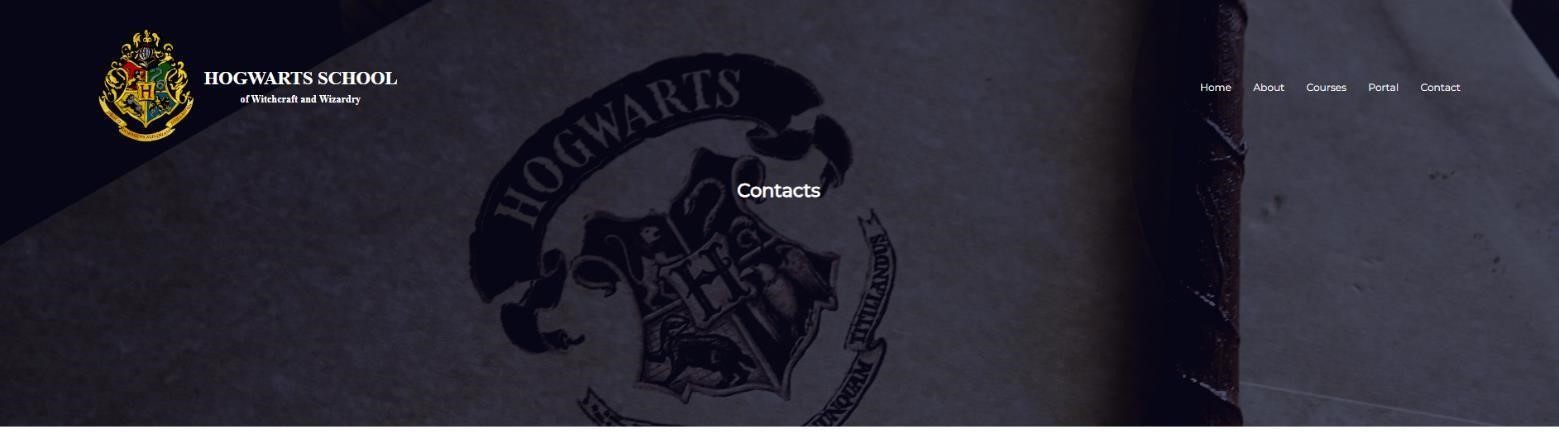
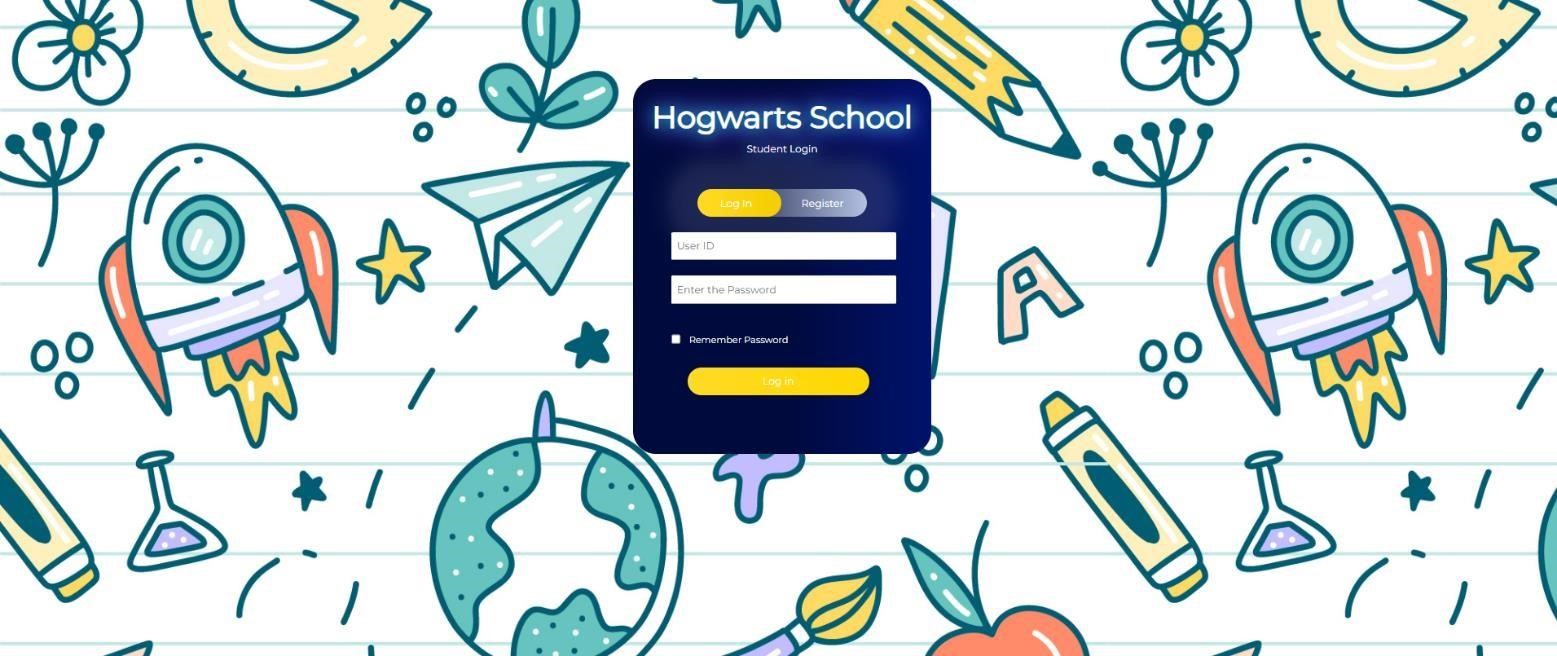
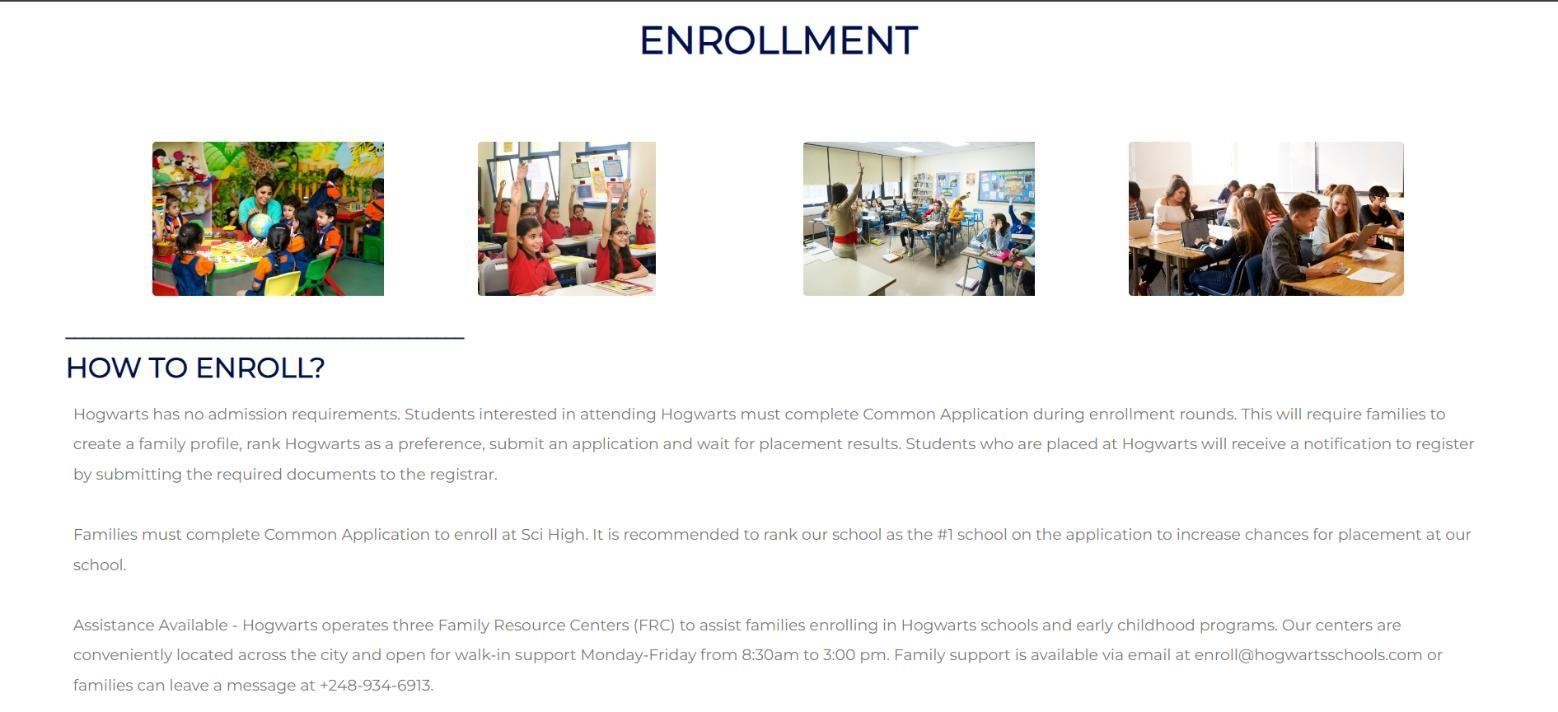
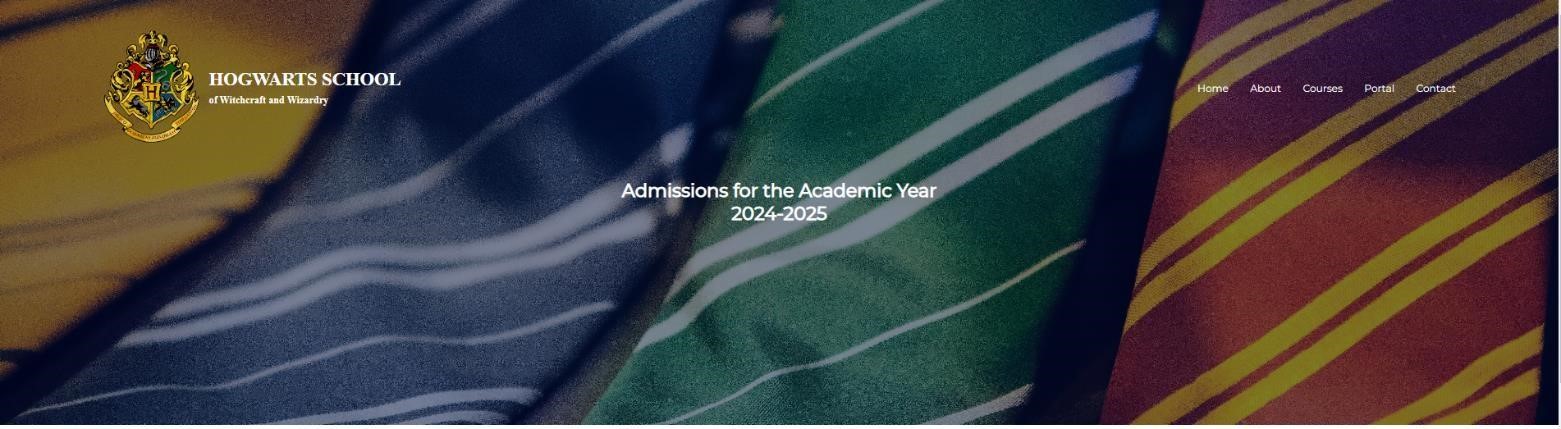
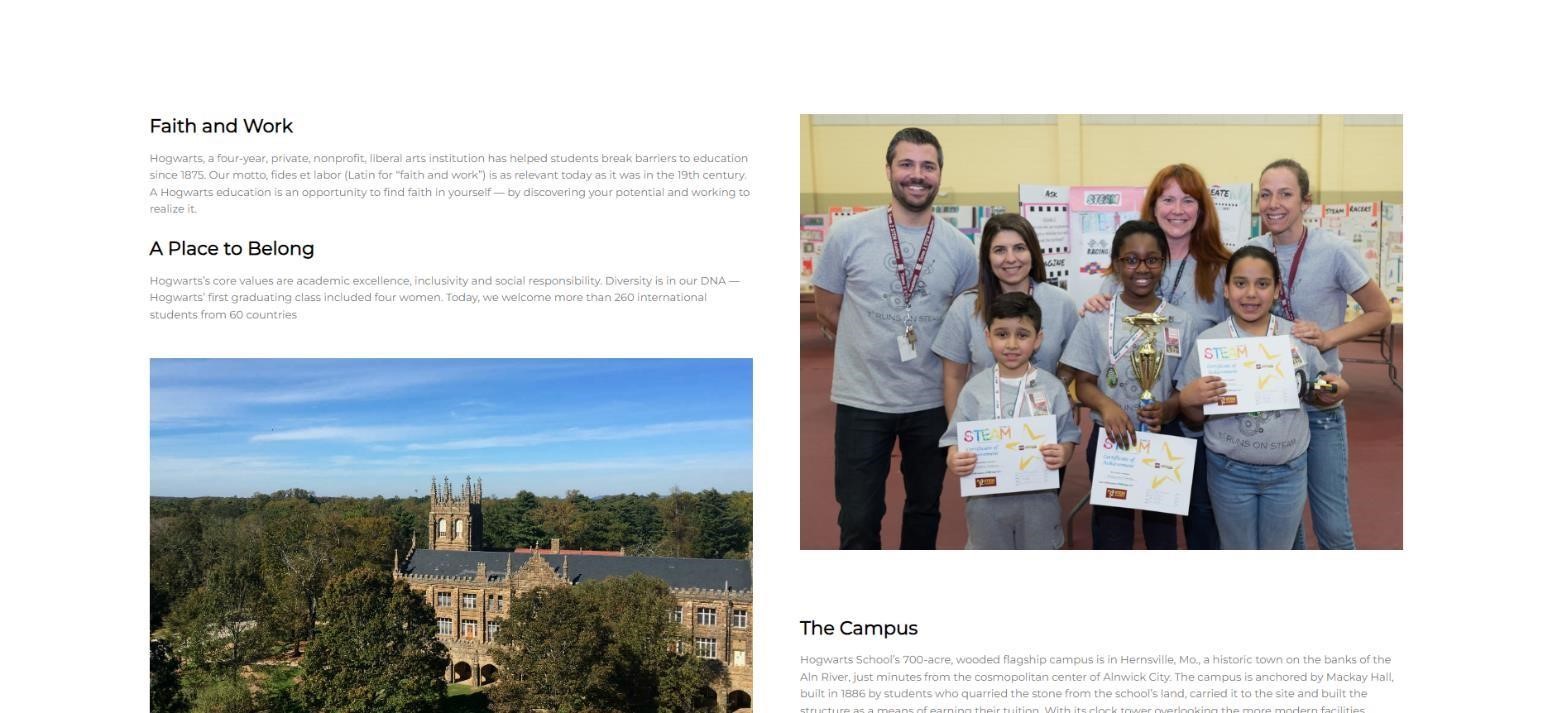
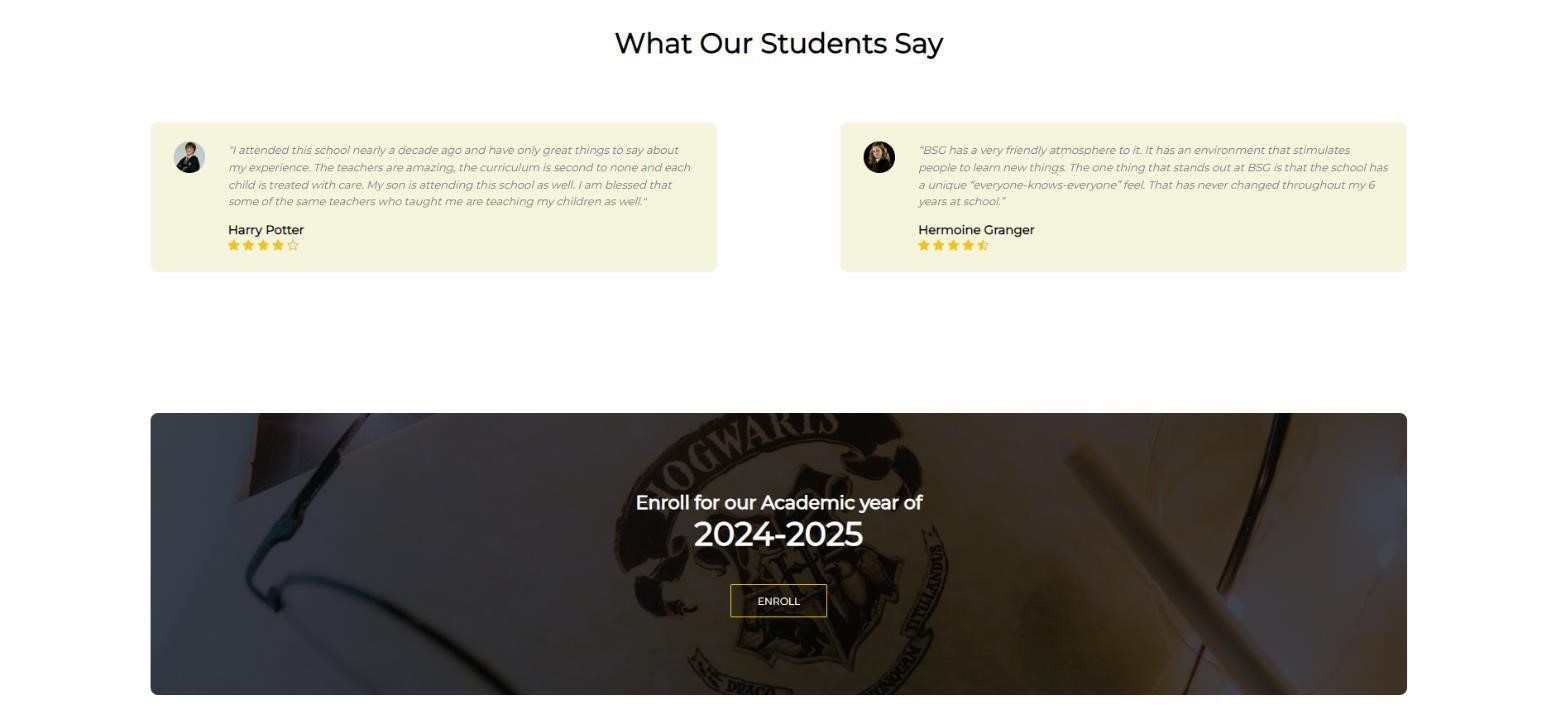
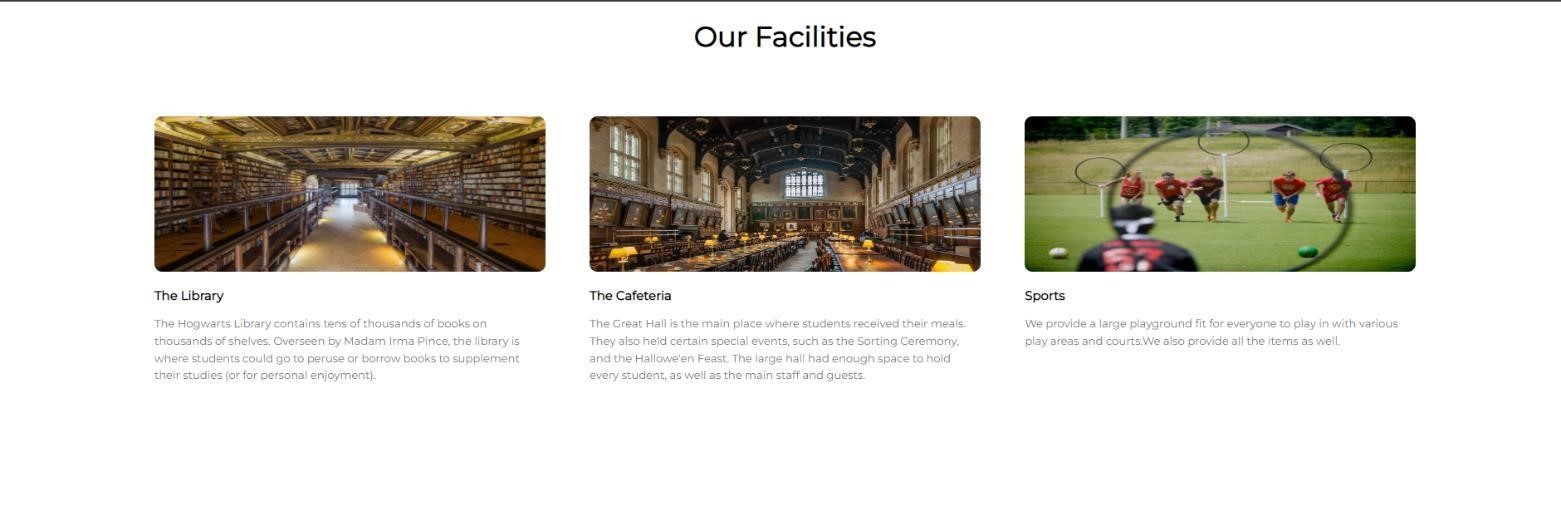
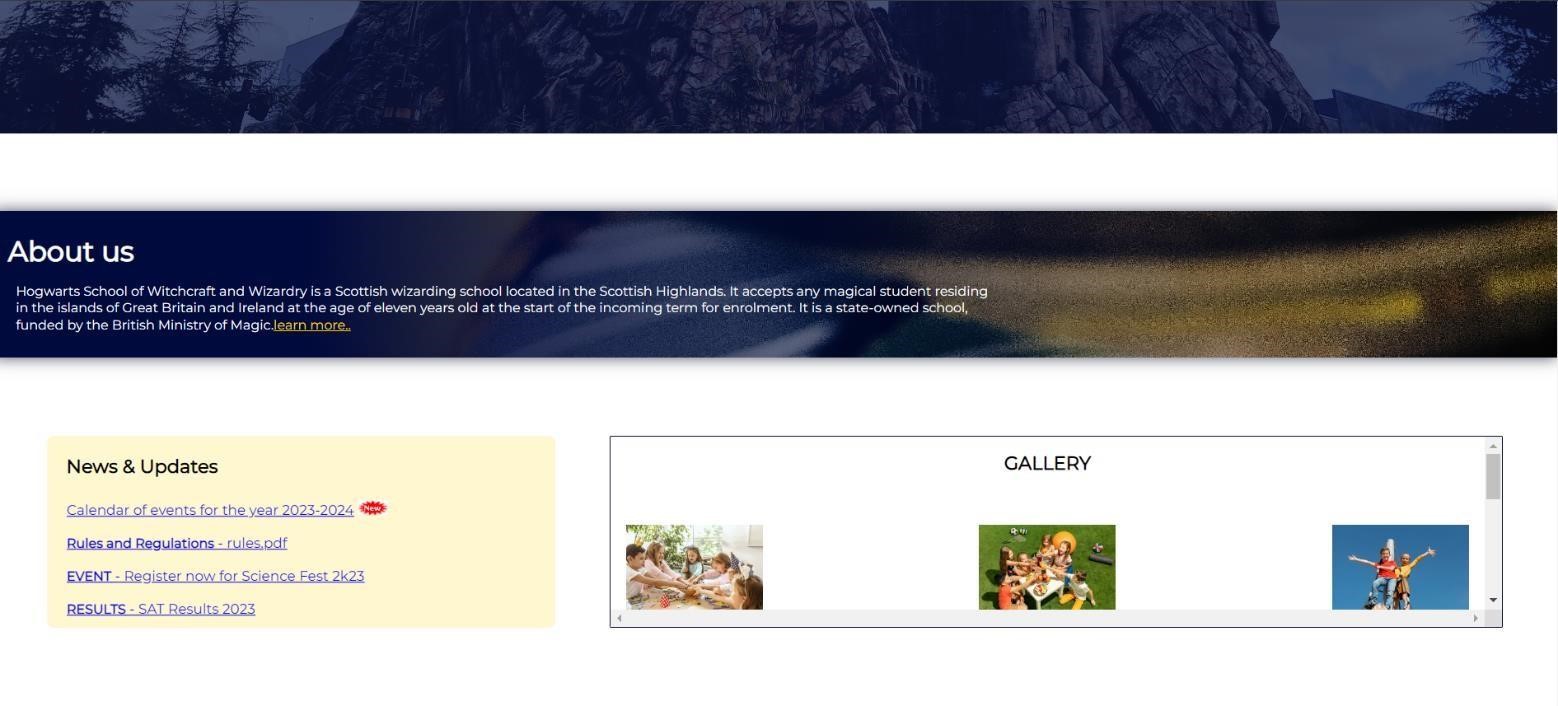




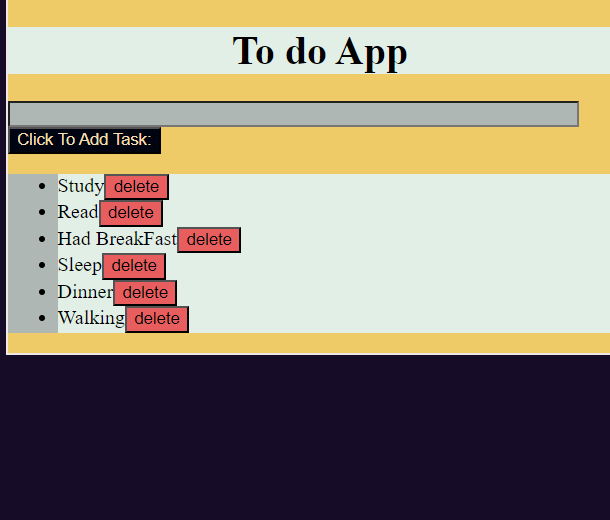
**Chapter 4:**

**Group Project :**

**School Web Site:**



**TO DO APP:**



**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<link rel="stylesheet" href="style.css">

<title>TO\_DO\_LIST</title>

</head>

<body>

<div>

<h1>To do App</h1>

<input type="text" placeholder="Enter Any Task To Add" id="input">

<button id="btn">Click To Add Task:</button>

<ul>

</ul>

</div>

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

</body>

</html>

**JS:**

let inp = document.querySelector("input");

let btn = document.querySelector("button");

let ul = document.querySelector("ul");

btn.addEventListener("click", function() {

let li = document.createElement("li");

li.innerText = inp.value;

let dltbtn = document.createElement("button");

dltbtn.innerText = "delete";

dltbtn.classList.add("delete");

li.appendChild(dltbtn);

ul.appendChild(li);

inp.value = " ";

});

ul.addEventListener("click", function(event) {

if (event.target.nodeName == "BUTTON") {

let item = event.target.parentElement;

item.remove();

}

})

**CSS:**

body {

background-color: #160C28;

}

div {

width: 500px;

height: auto;

background-color: #EFCB68;

border: 2px solid rgb(239, 232, 232);

margin: auto;

}

div h1 {

text-align: center;

background-color: #E1EFE6;

}

#input {

width: 450px;

display: grid;

color: #000411;

}

ul {

background-color: #AEB7B3;

font-family: 'Times New Roman', Times, serif;

}

li {

background-color: #E1EFE6;

}

#input {

background-color: #AEB7B3;

}

#btn {

background-color: #000411;

color: wheat;

}

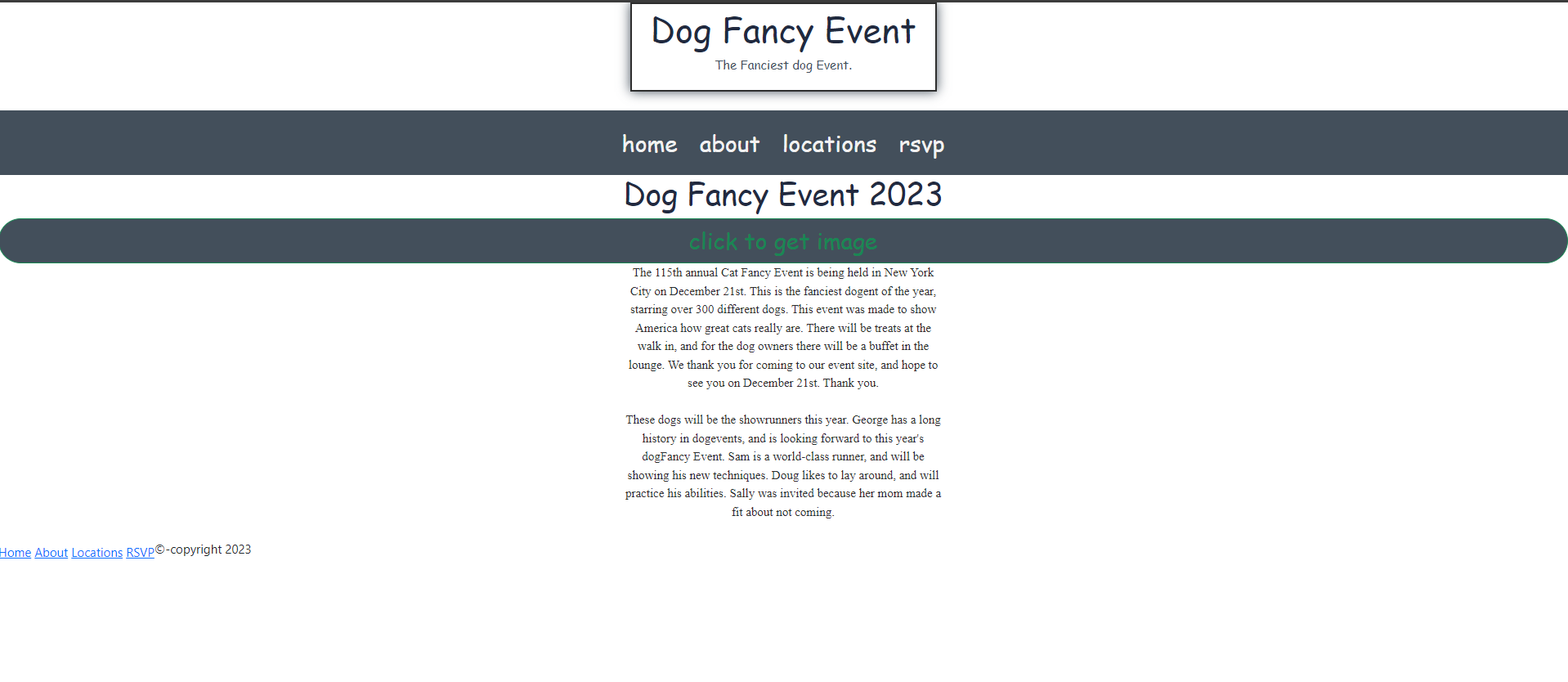
.delete {

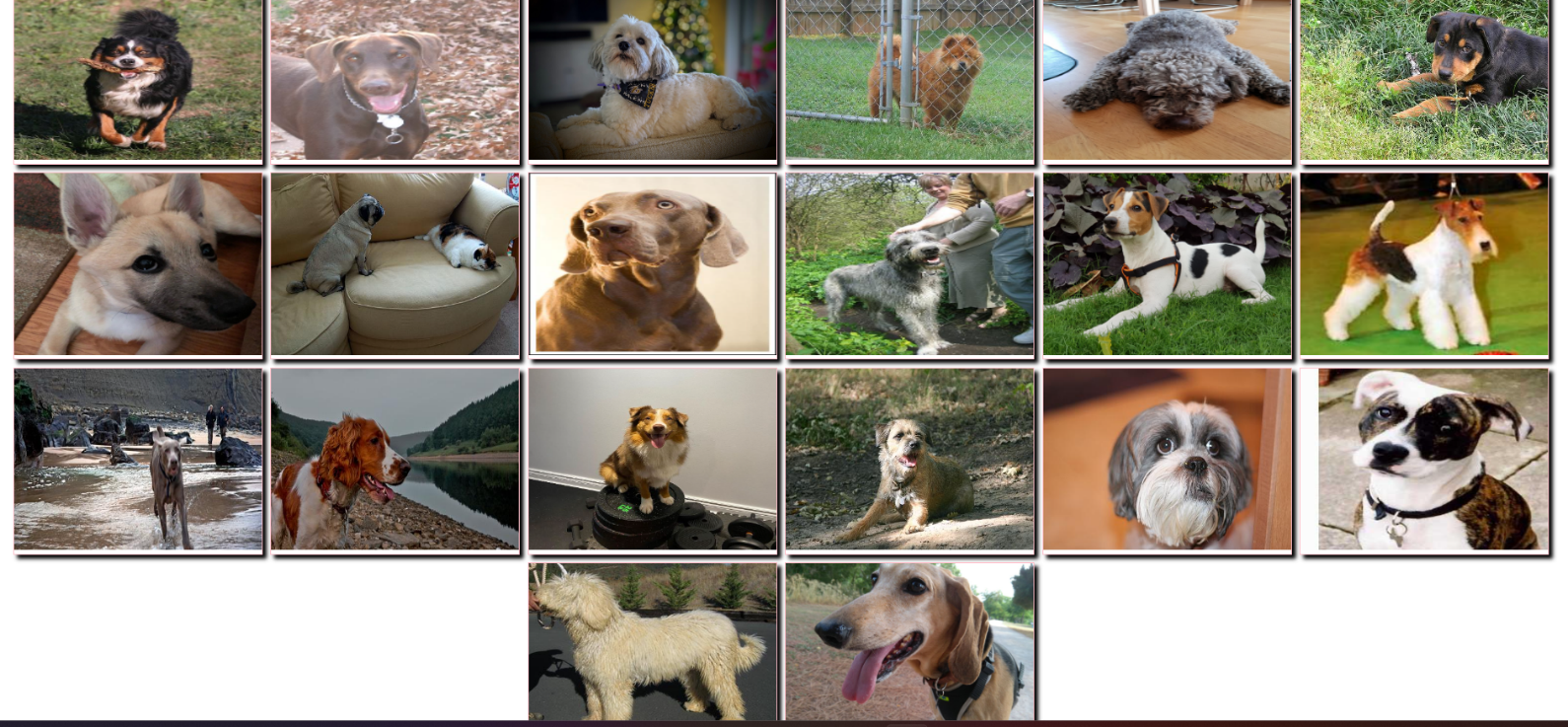
background-color: rgb(232, 94, 94);

text-align: center;

}

**The Fanciest dog Event:**

****

****

**CODE:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Dog Fancy Event</title>

<link rel="stylesheet" href="style.css">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="bootstrap-stylesheet">

</head>

<body>

<header>

<h1>Dog Fancy Event</h1>

<p>The Fanciest dog Event.</p>

</header>

<br>

<nav id="topnav">

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

<a href="#">About</a></li><li>

<a href="#">Locations</a>

</li> <li> <a href="#">RSVP</a> </li> </ul>

</nav>

<div id="content">

<h1>Dog Fancy Event 2023</h1>

<button type="button" class="btn btn-outline-success" id="get\_cat">click to get image</button>

<p id="event">The 115th annual Cat Fancy Event is being held in New York City on December 21st. This is the fanciest dogent of the year, starring over 300 different dogs. This event was made to show America how great cats really are. There will be treats at

the walk in, and for the dog owners there will be a buffet in the lounge. We thank you for coming to our event site, and hope to see you on December 21st. Thank you.</p>

<br> <ul id="list">

</ul>

<p id="info ">These dogs will be the showrunners this year. George has a long history in dogevents, and is looking forward to this year's dogFancy Event. Sam is a world-class runner, and will be showing his new techniques. Doug likes to lay around, and will

practice his abilities. Sally was invited because her mom made a fit about not coming.</p>

</div>

<br>

<footer>

<nav class="footernev">

<ul>

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

<li><a href="# ">About</a></li>

<li><a href="# ">Locations</a></li>

<li><a href="# ">RSVP</a></li>

</ul>

</nav>

<p>&copy;-copyright 2023</p>

</footer>

<script src="https://cdn.jsdelivr.net/npm/axios@1.1.2/dist/axios.min.js "></script>

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

</body>

</html>

let url2 = "https://dog.ceo/api/breeds/image/random";

let btn = document.querySelector("#get\_cat");

btn.addEventListener("click", async() => {

for (let i = 0; i < 10; i++) {

let link = await getimage();

show\_image(link);

console.log(link);

}

});

function show\_image(link) {

setTimeout(() => {

let list = document.querySelector("#list");

let img = document.createElement("img");

img.setAttribute("src", link);

list.appendChild(img);

}, 1000);

}

async function getimage() {

try {

let res = await axios.get(url2);

// console.log(res.data.message);

return res.data.message;

} catch (err) {

console.log("err", err);

}

}

img {

width: 320px;

height: 240px;

margin: 5px;

border: 2px solid pink;

box-shadow: 5px 5px 5px black;

padding-bottom: 5px;

}

header {

font-family: 'Dancing script', cursive;

text-align: center;

border: 3px solid #2E2E2E;

padding: 5px;

margin: 0px auto;

width: 400px;

box-shadow: 0px 0px 15px #434f5b;

}

header h1 {

color: #212a3f;

font-size: 45px;

}

header p {

color: #434f5b;

}

#topnav {

font-family: 'Dancing script', cursive;

background-color: #434f5b;

font-size: 30px;

text-transform: lowercase;

width: 415x;

margin: 0 auto;

}

#topnav ul {

padding: 20px 15px;

margin: 0px;

text-align: center;

}

#topnav ul li {

list-style-type: none;

display: inline;

}

#topnav ul li a {

color: #f2f2f2;

text-decoration: none;

margin: 0px 5px;

padding: 5px;

}

#topnav ul li a:hover {

border-bottom: 3px solid#f2f2f2;

transition: 50ms liner 0s;

-moz-transition: 50ms linear 0s;

-webkit-transition: 50ms linear 0s;

}

body {

background-color: #f2f2f2;

}

#content {

font-family: 'lora', serif;

}

#content h1,h2,h3,h4,h4,h6 {

color: #222a3f;

text-align: center;

font-family: 'Dancing script', cursive;

}

#content p {

color: #2e2e2e;

text-align: center;

width: 415px;

margin: 0px auto;

}

#content ul {

text-align: center;

margin: 0px;

padding: 0px;

}

.footernev {

padding-top: 5px;

padding-left: 0px;

float: left;

}

.footernev ul {

margin: 0px;

padding: 0px;

}

.footernev ul li {

list-style-type: none;

display: inline;

}

.footernav ul li a:link,

.footernav ul li a:visited {

color: #212A3F;

text-decoration: none;

text-transform: lowercase;

font-family: 'Dancing Script', cursive;

}

.footernav ul li a:hover {

color: #8F8F8F;

}

#get\_cat {

font-family: 'Dancing script', cursive;

background-color: #434f5b;

font-size: 30px;

text-transform: lowercase;

width: 100%;

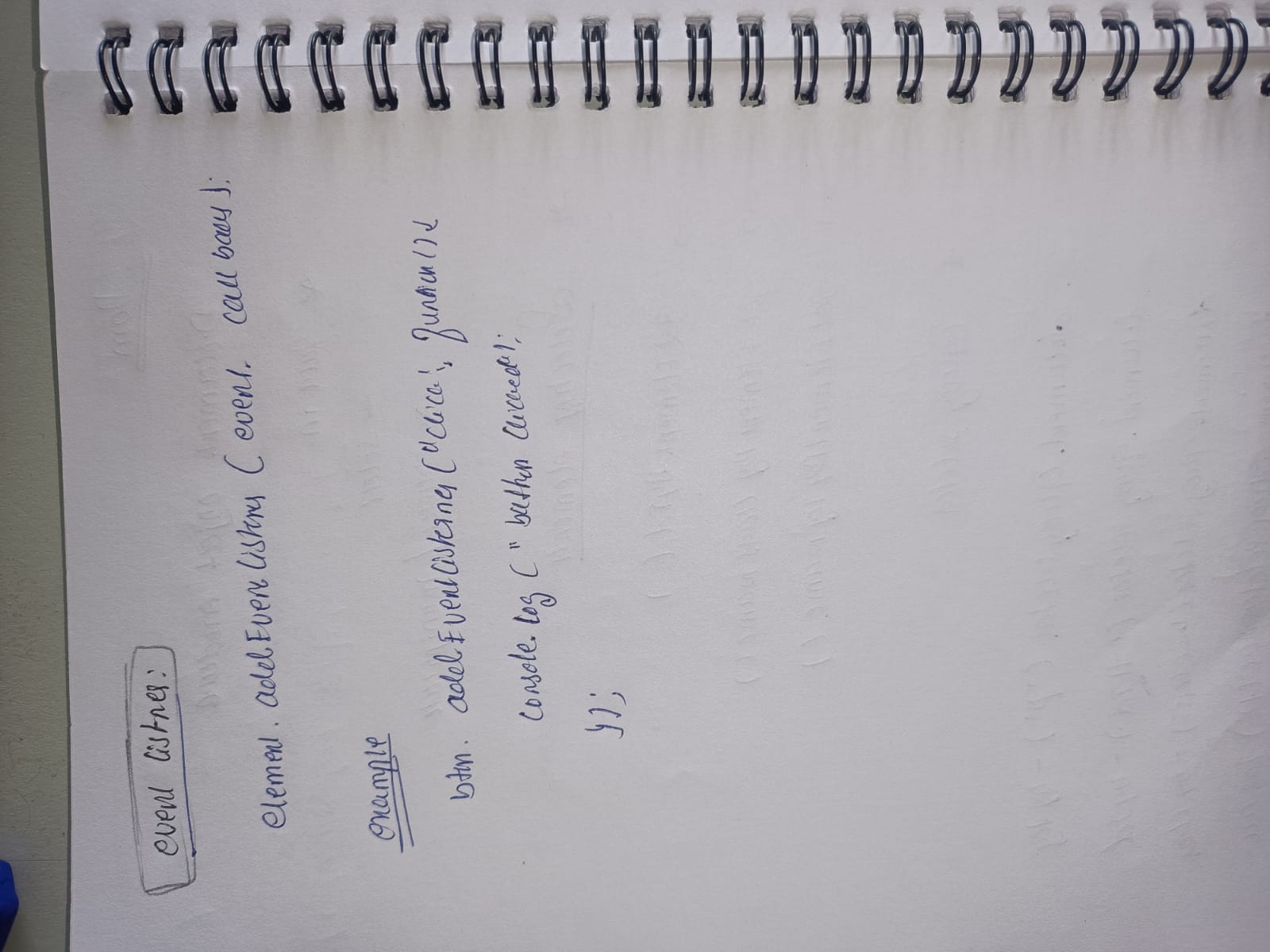
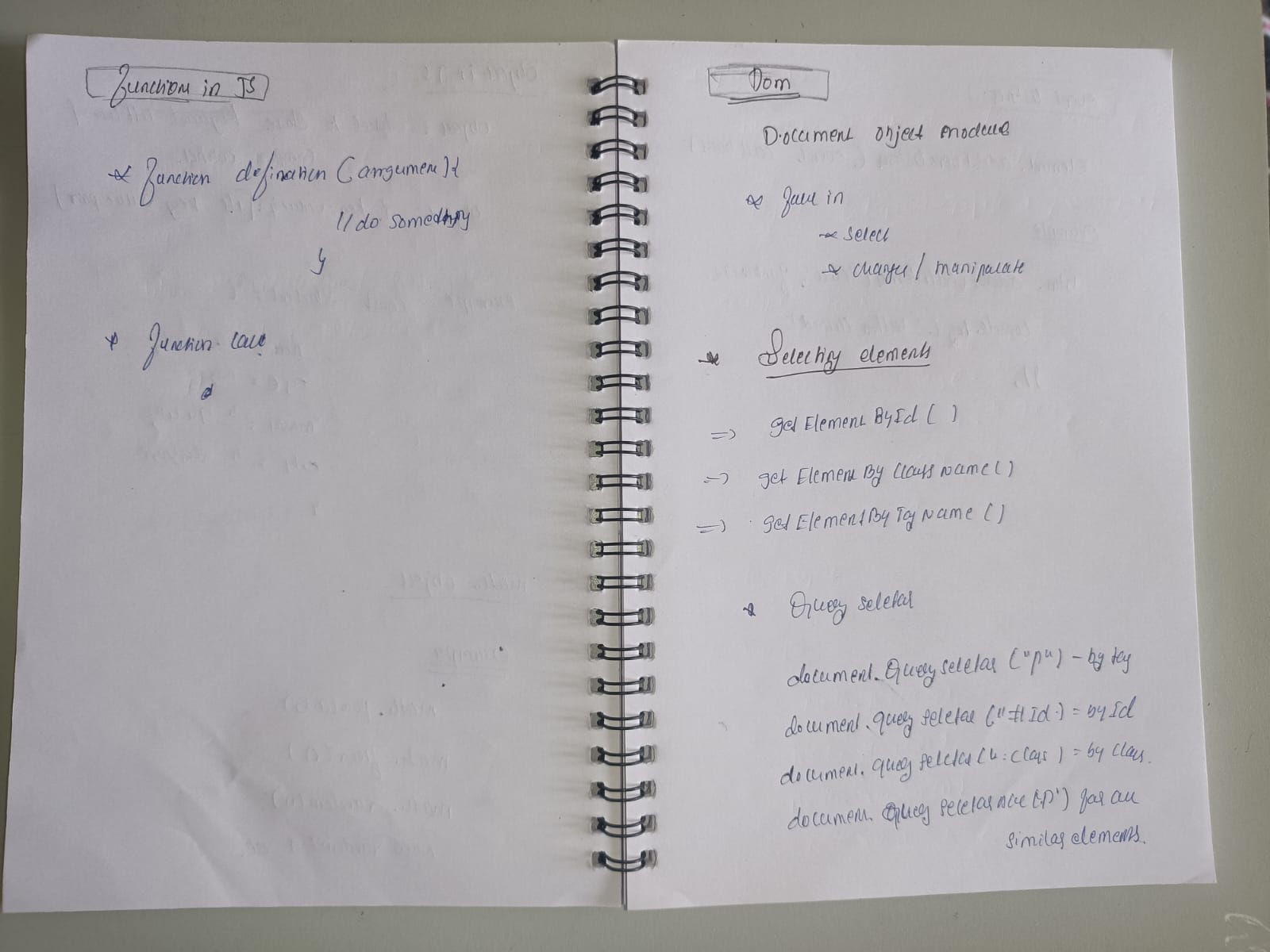
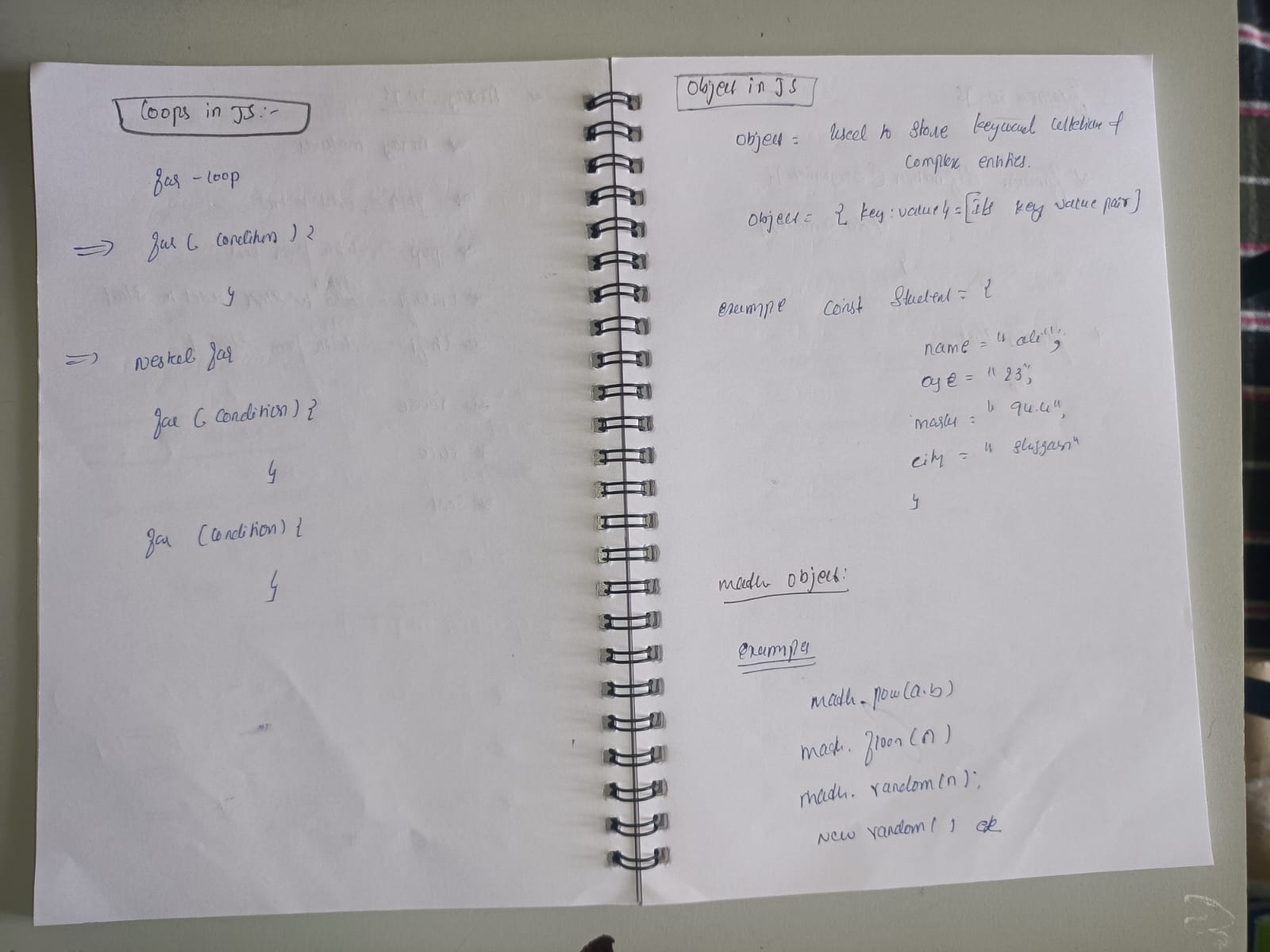
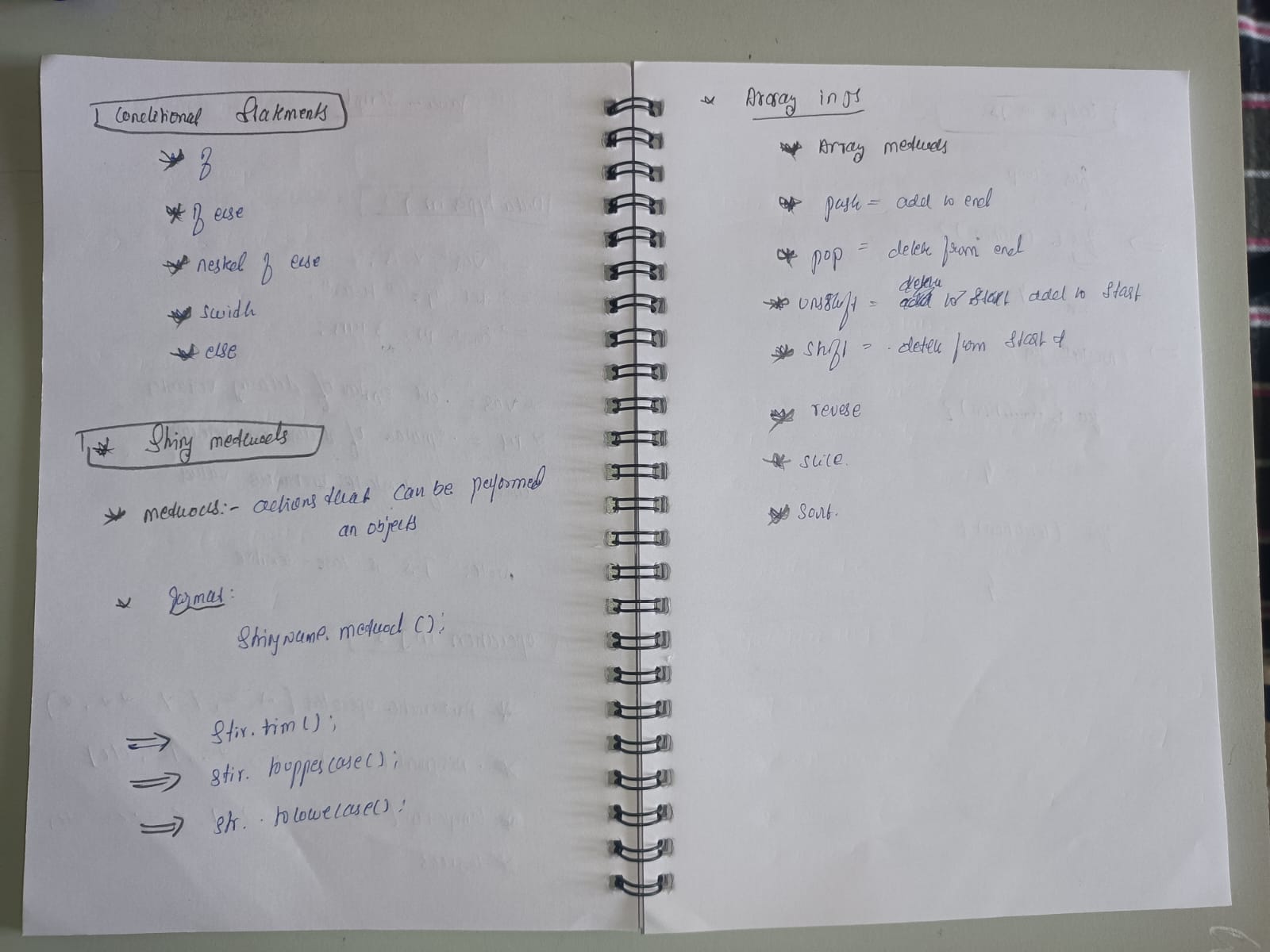
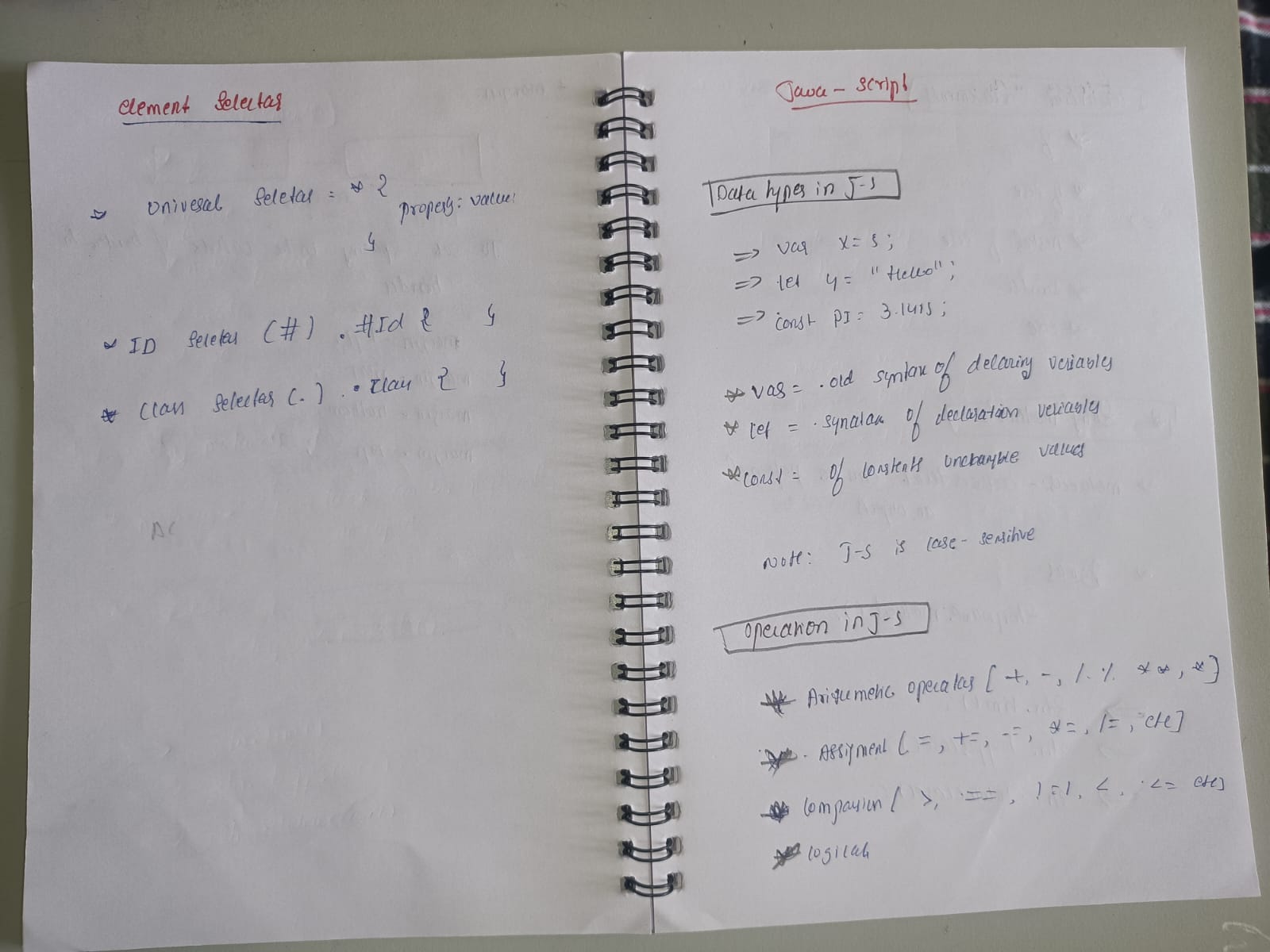
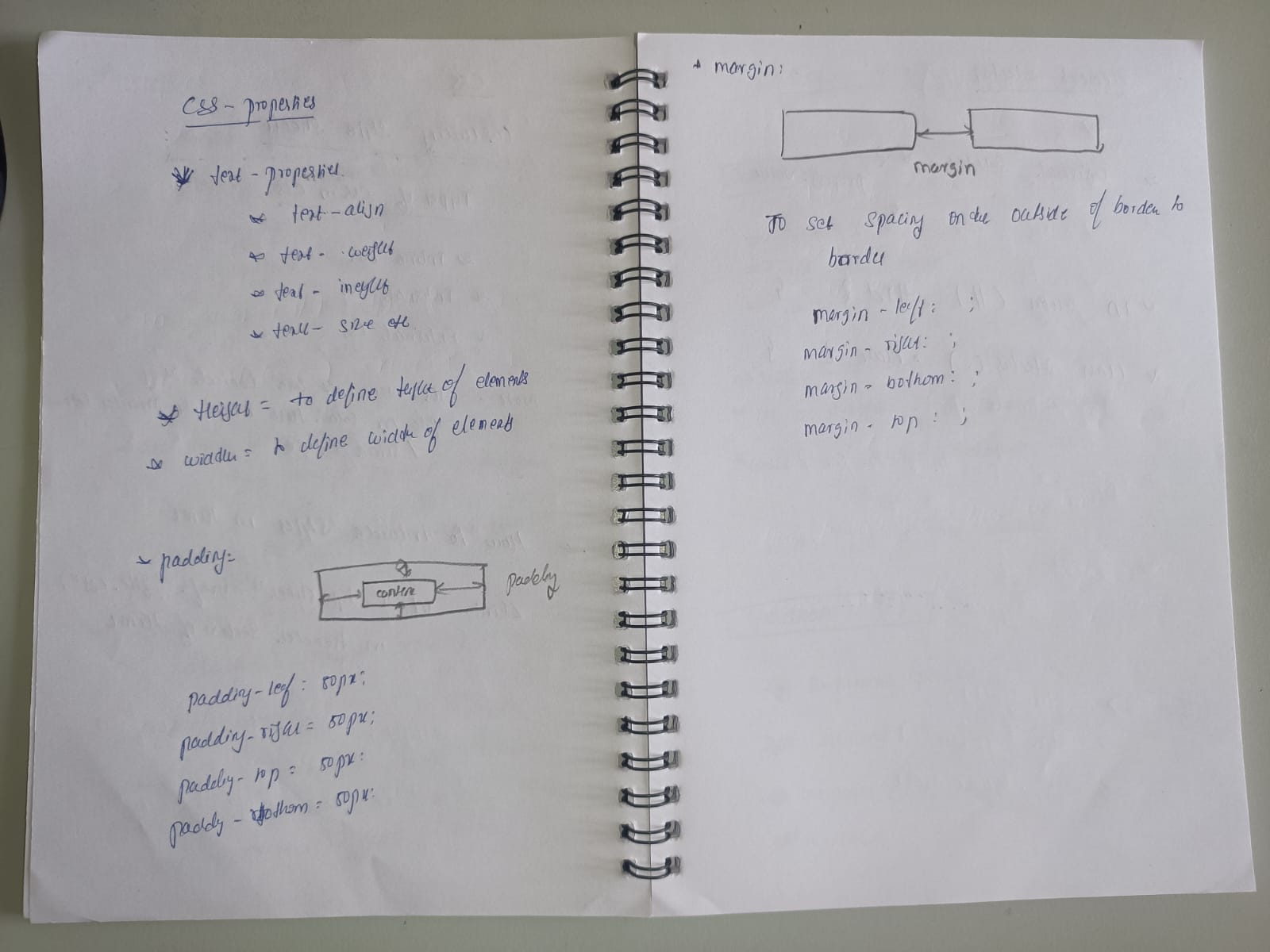
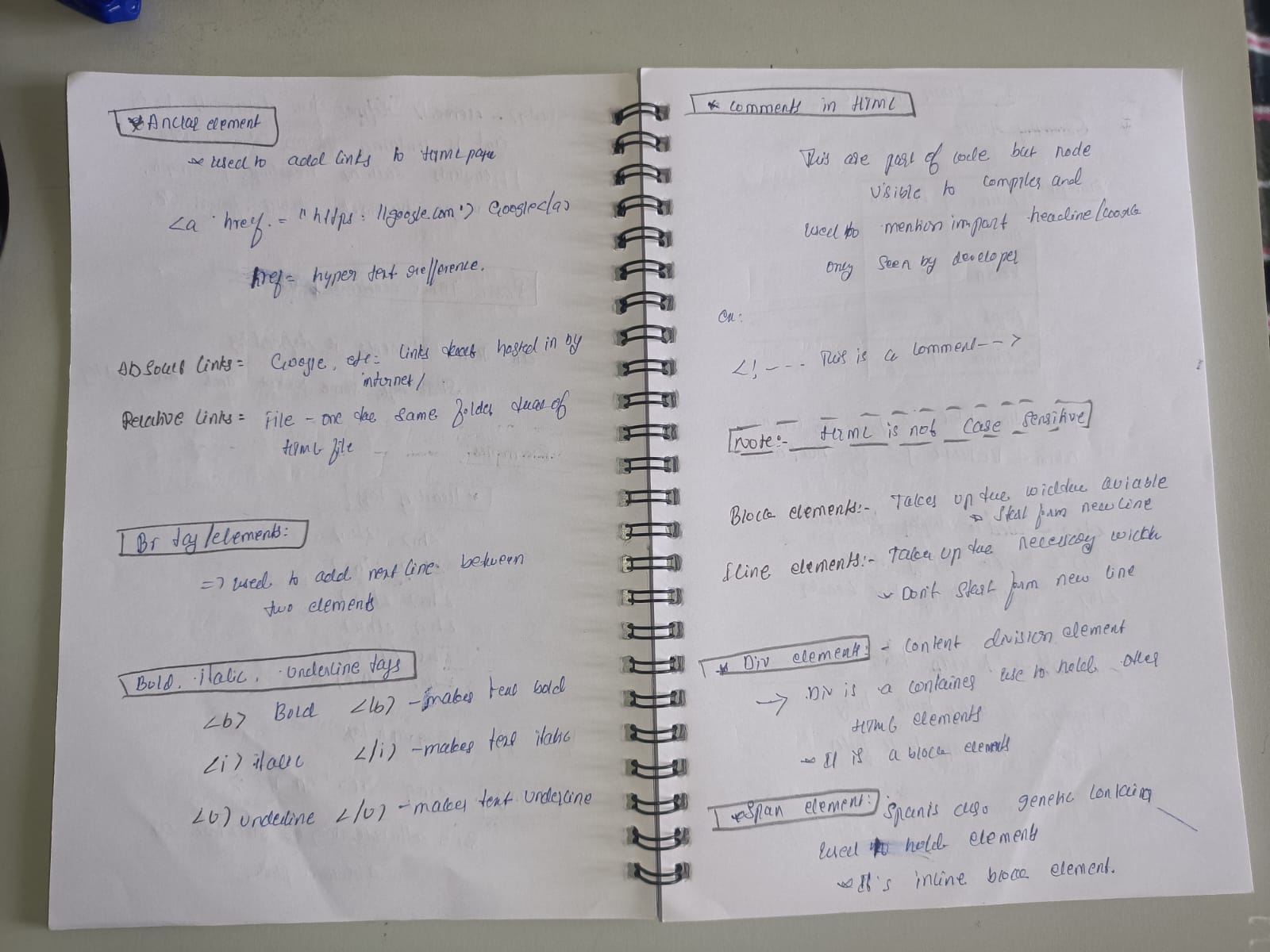
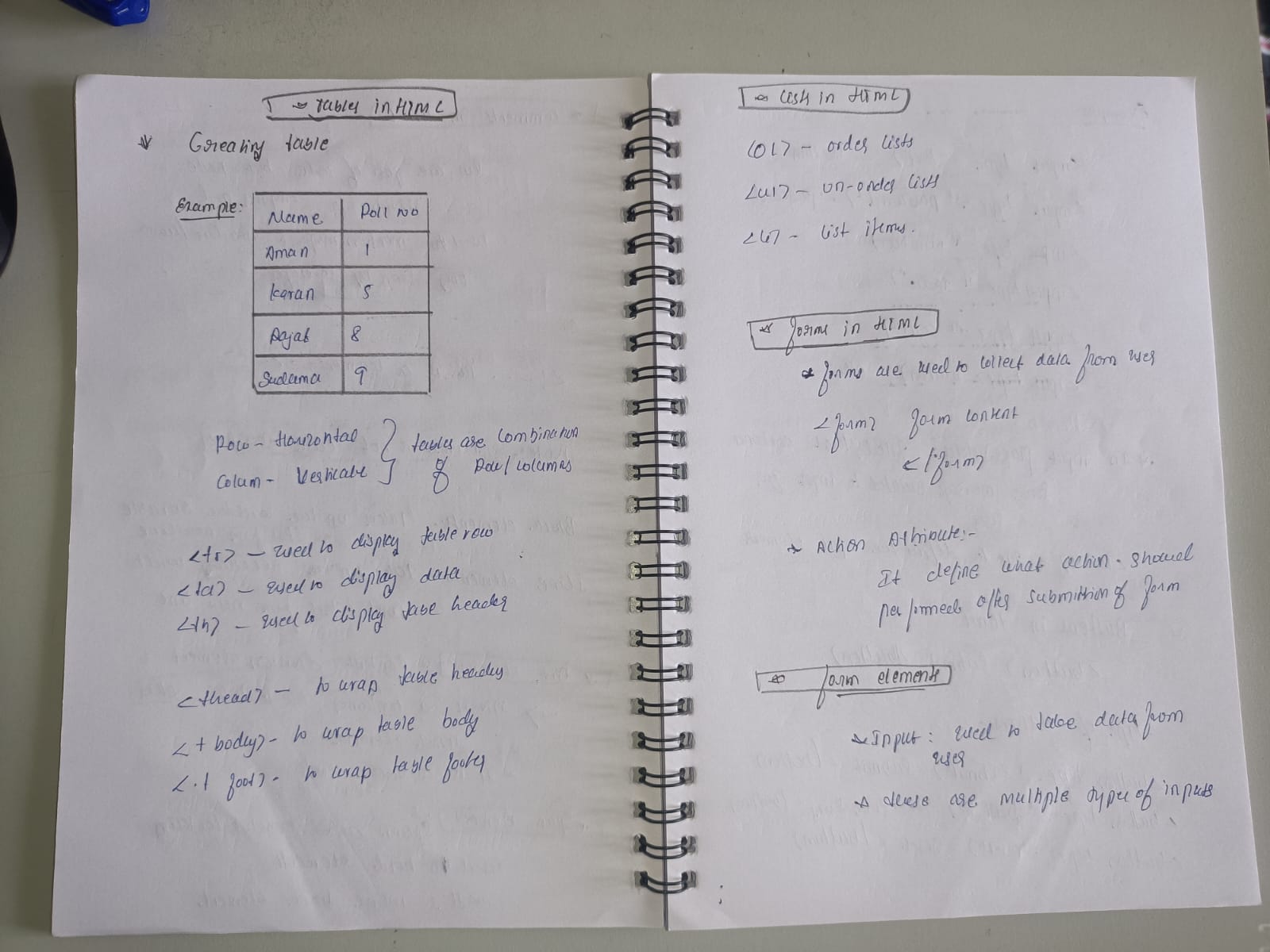
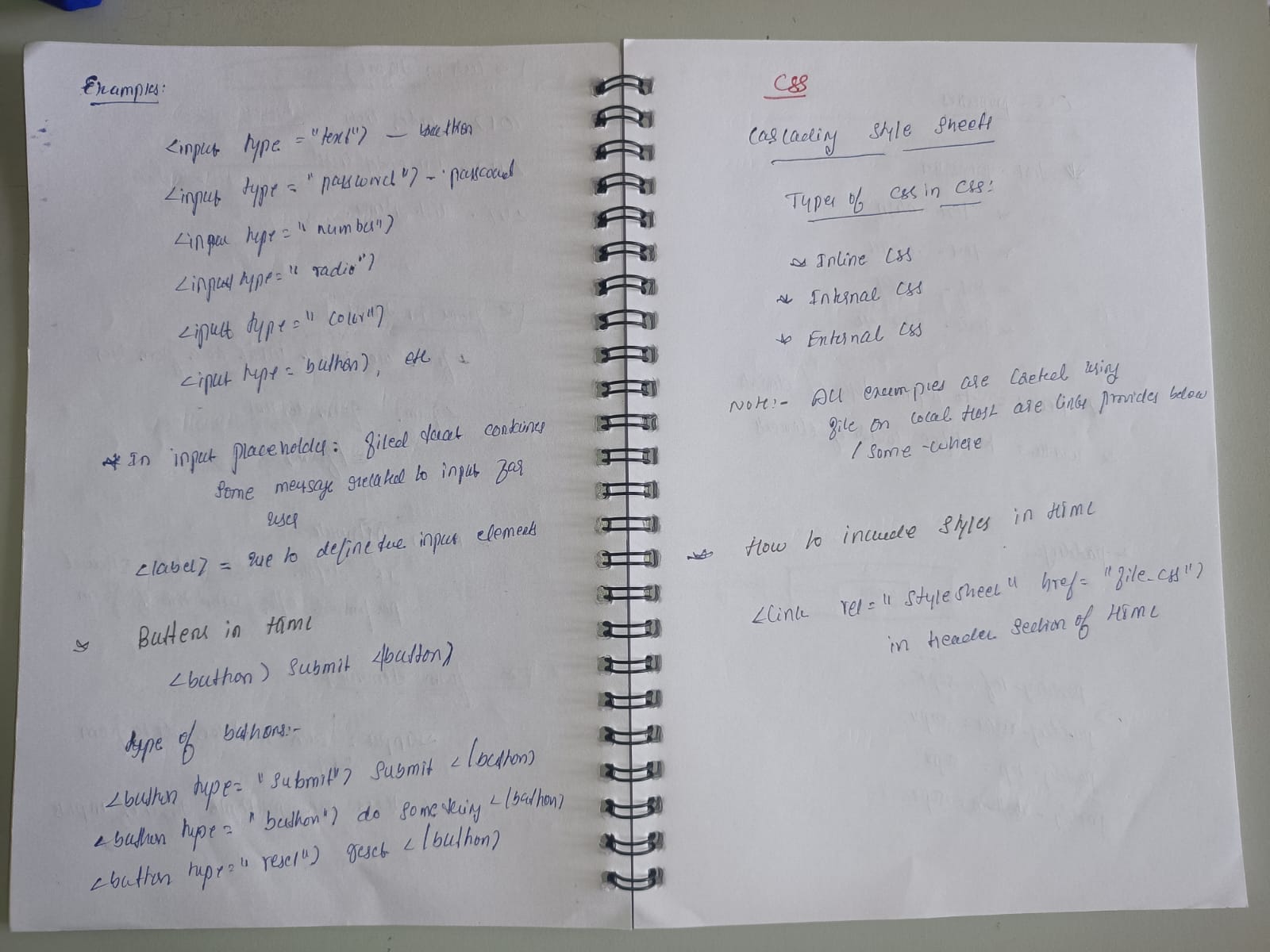
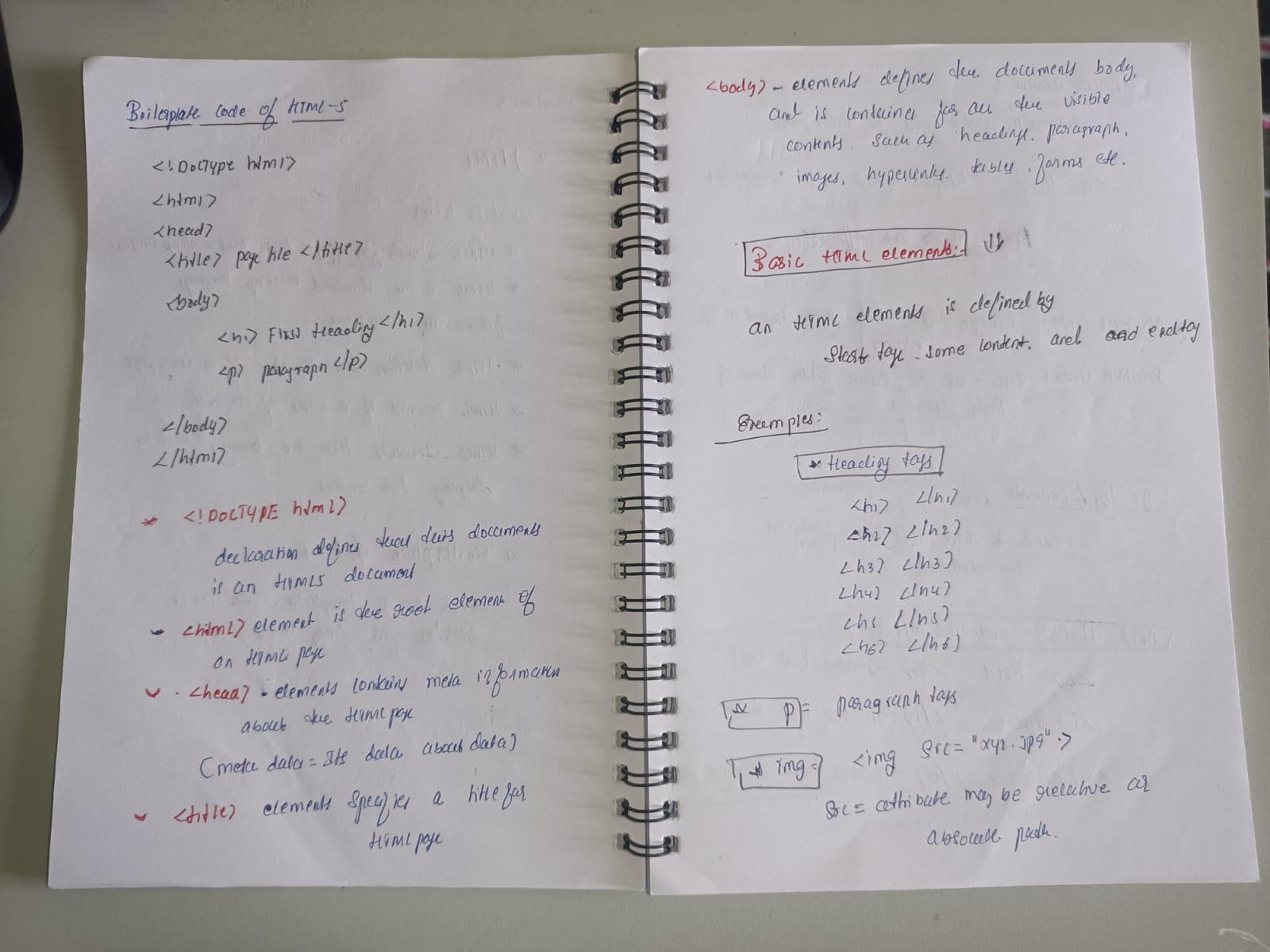
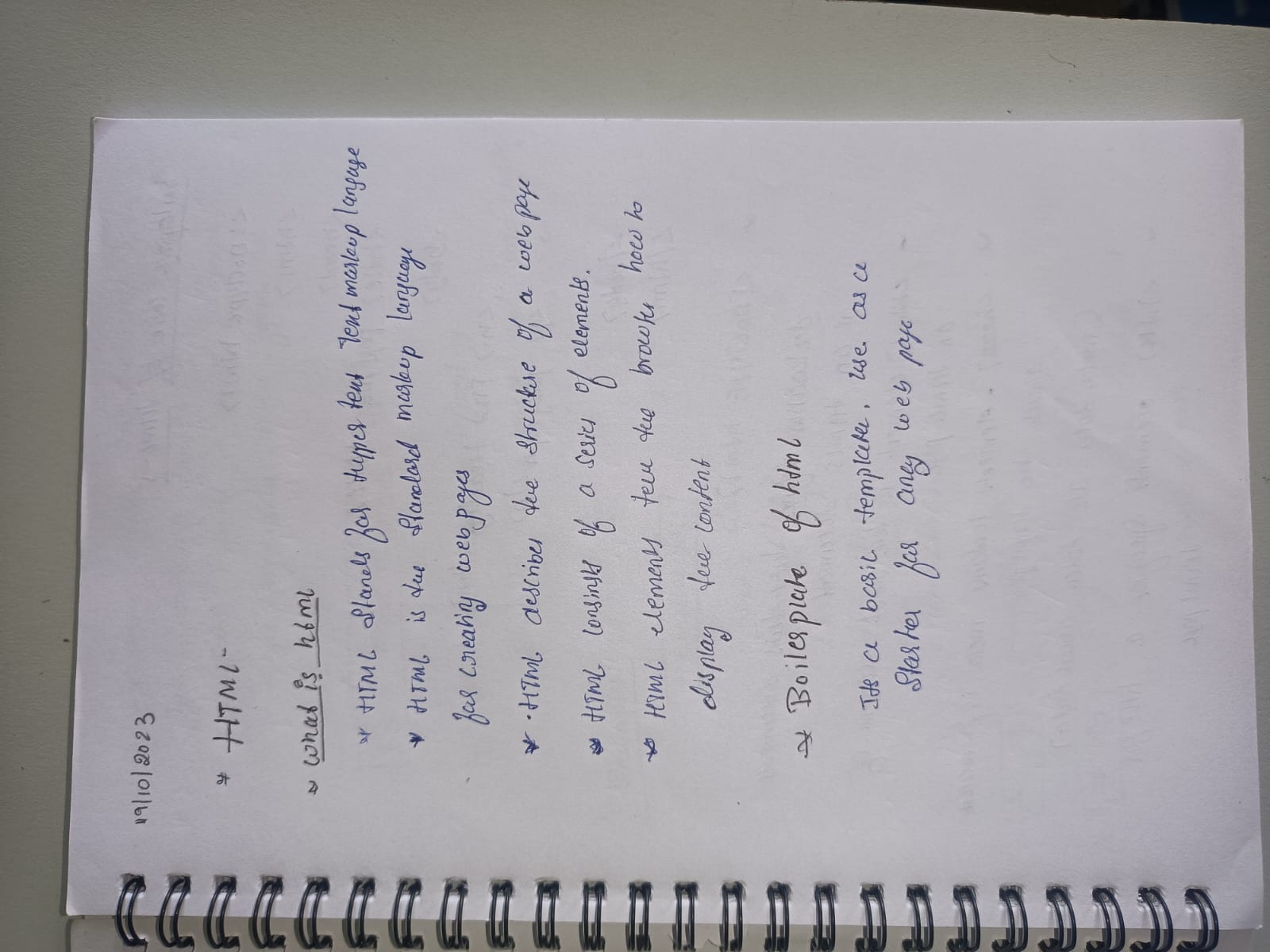
margin: 0 auto;

border-radius: 30px;

align-items: end;

}

**Chapter 5:**

**Diary :**