What is CSS?

- CSS (Cascading Style Sheets) is a fundamental technology used for designing and formatting web documents, particularly HTML and XML documents.
- Purpose: CSS is used to style and format the visual aspects of web pages, such as colors, fonts, spacing, layout, and more.
- CSS is a widely used language on the web.
- HTML, CSS and JavaScript are used for web designing.
- It helps the web designers to apply style on HTML tags.

History of CSS:

- **Early Web (Late 1980s - Early 1990s):** The web was text-based with minimal styling options, relying on HTML for both content and presentation.
- **Emergence of CSS (1996):** CSS was introduced by the W3C to separate content and presentation, with the release of CSS Level 1 (CSS1).
- **CSS2 (1998):** CSS2 expanded styling capabilities, including positioning and typography control.
- **Browser Support Issues (Late 1990s - Early 2000s):** Inconsistent browser support for CSS led to compatibility challenges.
- **CSS Zen Garden (2003):** Demonstrated the power of CSS in separating content from presentation.
- **CSS3 (Early 2000s):** Introduced modular specifications, adding features like gradients, animations, and responsive design with media queries.
- **Browser Standardization (2000s - Present):** Improved browser support reduced cross-browser compatibility issues.
- **CSS Frameworks and Preprocessors (Mid-2000s - Present):** The rise of CSS frameworks (e.g., Bootstrap) and preprocessors (e.g., SASS) streamlined web development.
- **Continual Evolution (Ongoing):** CSS continues to evolve, with new features and specifications being added to enhance web design and presentation.

❖ Father of CSS:

- Håkon Wium Lie is often referred to as the "father of CSS" or one of its co-creators. He worked on the development of CSS (Cascading Style Sheets) while he was with CERN (the European Organization for Nuclear Research) in the early 1990s. Together with Bert Bos, he proposed concept of CSS, and their work led to the first official CSS specification, CSS Level 1 (CSS1), in 1996

Advantages of CSS:

- **Separation of Concerns:** CSS separates design and layout from HTML content, making it easier to maintain and update websites.
- **Consistency:** It ensures consistent styling across web pages, promoting a unified brand identity.
- **Efficiency:** CSS files can be cached, reducing page load times and saving bandwidth.
- **Flexibility:** Allows for responsive design, adapting web layouts to various screen sizes and devices.
- **Modularity:** CSS is modular, enabling the reuse of styles across multiple pages.
- **Accessibility:** CSS supports accessibility features, making websites more inclusive.
- **Search Engine Optimization (SEO):** Properly structured CSS can improve a site's SEO
- **Faster Loading:** Reduces the amount of code in HTML, resulting in faster page rendering.
- **Ease of Maintenance:** Easier to update styles globally by modifying a single CSS file.
- Print-Friendly: Allows for creating print stylesheets, optimizing content for printouts.

❖ syntax:

```
selector {
  property: value;
  /* Additional properties and values */
}
```

❖ Explanation:

- **Selector:** Specifies which HTML element(s) the rule applies to. It can be an element name (e.g., p for paragraphs), a class (e.g., .my-class), an ID (e.g., #my-id), or more complex selectors.
- **Property**: Specifies the aspect of the element you want to style (e.g., color for text color, font-size for text size).
- **Value:** Defines the styling value for the property (e.g., blue for text color, 16px for font size).
- **Declaration Block:** Contains one or more property-value pairs enclosed in curly braces. Multiple properties are separated by semicolons.

Note: Selector{Property1: value1; Property2: value2; .....;}

Example:

```
h3 {
  /* Declaration Block */
  color: blue;      /* Property: Value */
  font-size: 30px;  /* Property: Value */
  font-weight: bold; /* Property: Value */
}
```

- To add the sytle in the HTML document:

- You can add CSS styles to HTML documents in several ways.

Example:

1. Inline CSS:

- Description: Inline CSS is added directly to individual HTML elements using the `style` attribute. It applies styles only to that specific element.

Example:

- This is a blue text.

2. Internal CSS:

- Description: Internal CSS is placed within the HTML document's `<style>` element in the `<head>`. It applies styles to all elements on that page.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p {
      color: green;
      font-size: 18px;
    }
  </style>
</head>
<body>
  <p>This is a green text.</p>
</body> </html>
```

3. External CSS:

- Description: External CSS is stored in a separate `.css` file and linked to the HTML document using the `<link>` element. It can be applied consistently across multiple HTML pages.

Example:

- Create an external CSS file (raj.css):

```
/* raj.css */
 p {
 color: red;
font-size: 20px;
 }
```

- Link it to your HTML document:

Example:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="./raj.css">
</head>
<body>
  <p>This is a red text.</p>
</body>
</html>
```

Note: To add an external CSS file located in a different folder to your HTML document, you can use the <link> element with a href attribute specifying the path to the CSS file.

- **Suppose you have the following directory structure:**

  - my_website/
    - index.html
    - css/
      - raj.css

- In your HTML file (index.html), you can add the external CSS file (raj.css) like this:

Example:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="css/raj.css">
</head>
<body>
  <h1>Welcome to My Website</h1>
  <p>This is some content.</p>
</body>
</html>
```

Note: If your CSS file is located on a different drive (e.g., a different disk partition) than your HTML file, you can specify an absolute file path using the file:/// protocol in the href attribute of the <link> element. Here's how to do it:

- **Suppose you have the following setup:**

  HTML file location: C:\my_website\index.html
  CSS file location: D:\css\raj.css
  In your HTML file (index.html), you can add the external CSS file (raj.css) like this:

Example:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="file:///D:/css/raj.css">
</head>
<body>
  <h1>Welcome to My Website</h1>
  <p>This is some content.</p>
</body>
</html>
```

Explanation:

We use the file:/// protocol to specify an absolute file path.
D: represents the drive where your CSS file (raj.css) is located.
D:/css/raj.css is the absolute path to the CSS file.

4. Embedded CSS:

- Description: Embedded CSS is placed within a `<style>` element directly in the HTML document's `<head>`. It allows for styling specific elements on that page.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p {
      color: purple;
      font-size: 22px;
    }
  </style>
</head>
<body>
  <p>This is a purple text.</p>
</body></html>
```

5. CSS Frameworks:

- Description: CSS frameworks like Bootstrap and Foundation provide pre-designed styles and components. You include their CSS files and utilize predefined classes to style elements.

Example:

- **Link to a CSS framework (Bootstrap):**

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body>
  <p class="text-primary">This is a blue text from Bootstrap.</p>
</body></html>
```

Note: Choose the method that best suits your project's needs.

Priority of adding CSS styles in HTML documents:

- The priority of adding CSS styles in HTML documents is determined by several factors.

Example:

1. Inline CSS:

- Priority**: Highest
- Explanation: Inline styles are added directly to individual HTML elements using the `style` attribute. They override all other styles.

Example:

- This text is red.

2.Internal/Embedded CSS:

- Priority: Second highest
- Explanation: Internal CSS is placed within a `<style>` element in the HTML document's `<head>`. It applies styles to the entire page.

Example:

```
<style>
  p {
    color: blue;
  }
</style>
```

3. External CSS:

- Priority: Third highest
- Explanation: External CSS is stored in a separate `.css` file linked to the HTML document using the `<link>` element. It applies styles to the entire page.

Example:

Create an external CSS file (styles.css):

```
p {
  color: green;
}`
```

- **Link it to your HTML document:**
  - <link rel="stylesheet" type="text/css" href="styles.css">

4. Specificity and Selector Specificity:

- Priority: Determined by selector specificity
- Explanation: When multiple CSS rules target the same element, the one with higher specificity takes precedence. Specificity is based on the selectors used.

Example:

```
/* More specific selector */
p.my-class {
  color: purple;
}
/* Less specific selector */
p {
  color: yellow;
}
```

- The more specific selector (`.my-class`) takes priority over the less specific one (`p`) for elements with the class "my-class."

5. Order of Styles:

- Priority: The last style rule encountered takes precedence
- Explanation: If multiple rules with the same specificity target the same element, the one defined last in the document or in an external CSS file takes priority.

Example:

```
/* This rule takes precedence */
p {
  color: orange;
}
/* This rule is overridden */
p {
  color: pink;
}
```

6. !important:

- Priority: Overrides all other styles
- Explanation: Adding `!important` to a CSS rule makes it the highest priority, even overriding inline styles and specificity.

Example:

```
p {
  color: brown !important;
}
```

CSS selector:

1. Universal Selector: `*`

- Explanation: Selects all elements on the page.

Example:

```
* {
  margin: 0;
  padding: 0;
}
```

2. Type Selector (Element Selector): `elementname`

- Explanation: Selects elements of a specific type.

Example:

```
p {
  color: blue;
}
```

- This selects all `<p>` (paragraph) elements and sets their text color to blue.

3. Class Selector: `.classname`

- Explanation: Selects elements with a specific class attribute value.

Example:

```
.highlight {
  background-color: yellow;
}
```

- This selects all elements with `class="highlight"` and gives them a yellow background color.

4. ID Selector: `#idname`

- Explanation: Selects a single element with a specific `id` attribute value.

Example:

```
#header {
  font-size: 24px;
}
```

- This selects the element with `id="header"` and sets its font size to 24 pixels.

5. Descendant Selector: `ancestor descendant`

- Explanation: Selects all descendants of the specified ancestor.

Example:

```
nav ul {
  list-style-type: square;
}
```

- This selects all `<ul>` elements that are descendants of a `<nav>` element and sets their list style to squares.

6. Child Selector: `parent > child`

- Explanation: Selects direct children of a parent element.

Example:

```
.menu > li {
  font-weight: bold;
}
```

- This selects all `<li>` elements that are direct children of elements with `class="menu"` and sets their font weight to bold.

7. Adjacent Sibling Selector: `element1 + element2`

- Explanation: Selects an element immediately preceded by a specified sibling element.

Example:

```
h2 + p {
  margin-top: 10px;
}
```

- This selects all `<p>` elements that come right after an `<h2>` element and adds a top margin of 10 pixels

8. General Sibling Selector: element1 ~ element2

- Explanation: Selects all sibling elements that follow a specified element.

Example:

```
h3 ~ p {
  color: gray;
}
```

- This selects all `<p>` elements that are siblings of `<h3>` elements and sets their text color to gray.

9. Attribute Selector: [attribute=value]

- Explanation: Selects elements with a specific attribute and value.

Example:

```
[data-type="button"] {
  background-color: green;
}
```

- This selects all elements with `data-type="button"` and gives them a green background color.

10. Pseudo-classes: :pseudo-class

- Explanation: Selects elements based on their state or position.

Example:

```
a:hover {
  text-decoration: underline;
}
```

- This selects all `<a>` elements when the mouse pointer hovers over them and adds an underline to the text.

11. Pseudo-elements: ::pseudo-element

- Explanation: Selects specific parts of an element.

Example:

```
p::before {
  content: "Before text: ";
}
```

- This inserts content before all `<p>` elements.

12. Grouping Selector: selector1, selector2, ...

- Explanation: Groups multiple selectors together to apply the same styles.

Example:

```
h1, h2, h3 {
  font-family: Arial, sans-serif;
}
```

- This selects all `<h1>`, `<h2>`, and `<h3>` elements and sets their font family to Arial or a generic sans-serif font.

13. Not Selector: :not(selector)

- Explanation: Selects elements that do not match a specified selector.

Example:

```
p:not(.special) {
  font-style: italic;
}
```

- This selects all `<p>` elements except those with `class="special"` and sets their font style to italic.

14. Attribute Value Prefix Selector: [attribute^="value"]

- Explanation: Selects elements with attribute values that start with a specified string.

Example:

```
[href^="https://"] {
  color: blue;
}
```

- This selects all elements with `href` attributes starting with "https://" and sets their text color to blue.

15. Attribute Value Substring Selector: [attribute*="value"]

- Explanation: Selects elements with attribute values that contain a specified substring.

Example:

```
[class*="button"] {
  background-color: gray;
}
```

- This selects all elements with `class` attributes containing "button" and gives them a gray background color.

16. Attribute Value Suffix Selector: [attribute$="value"]

- Explanation: Selects elements with attribute values that end with a specified string.

Example:

```
[src$=".jpg"] {
  border: 1px solid black;
}
```

- This selects all elements with

CSS Selector Example with Html:

1. Universal Selector: `*`

- Selects all elements on the page.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    * {
      margin: 0;
      padding: 0;
      border: 1px solid black;
    }
  </style>
</head>
<body>
  <p>This is a paragraph.</p>
  <div>This is a div.</div>
</body>
</html>
```

2. Type Selector (Element Selector): elementname

- Selects elements of a specific type.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p {
      font-size: 18px;
    }
  </style>
</head>
<body>
  <p>This is a paragraph with larger text.</p>
  <div>This is a div.</div>
</body>
</html>
```

3. Class Selector: .classname

- Selects elements with a specific class attribute value.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .highlight {
```

```
        background-color: yellow;
      }
    </style>
  </head>
  <body>
    <p class="highlight">This is a highlighted paragraph.</p>
    <div>This is a div.</div>
  </body>
  </html>
```

4. ID Selector: #idname

- Selects a single element with a specific `id` attribute value.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    #my-header {
      font-size: 24px;
    }
  </style>
</head>
<body>
  <h1 id="my-header">This is a header with larger text.</h1>
  <div>This is a div.</div>
</body>
</html>
```

5. Descendant Selector: ancestor descendant

- Selects all descendants of the specified ancestor.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    ul li {
      list-style-type: square;
    }
  </style>
</head>
<body>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
</body>
</html>
```

6. Child Selector: parent > child

- Selects direct children of a parent element.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    ul > li {
      font-weight: bold;
    }
  </style>
</head>
<body>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
</body>
</html>
```

7. Adjacent Sibling Selector: element1 + element2

- Selects an element immediately preceded by a specified sibling element.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h2 + p {
      margin-top: 10px;
    }
  </style>
</head>
<body>
  <h2>Heading</h2>
  <p>This is a paragraph with a top margin.</p>
  <h2>Another Heading</h2>
  <p>This is another paragraph without a top margin.</p>
</body>
</html>
```

8. General Sibling Selector: element1 ~ element2

- Selects all sibling elements that follow a specified element.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h3 ~ p {
      color: gray;
```

```
    }
  </style>
</head>
<body>
  <h3>Subheading</h3>
  <p>This is a gray text.</p>
  <h3>Another Subheading</h3>
  <p>This is another gray text.</p>
</body>
</html>
```

9. Attribute Selector: [attribute=value]

- Selects elements with a specific attribute and value.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    [data-type="button"] {
      background-color: green;
    }
  </style>
</head>
<body>
  <button data-type="button">Click me</button>
  <button>Don't click me</button>
</body>
</html>
```

10. Pseudo-classes: :pseudo-class

- Selects elements based on their state or position.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    a:hover {
      text-decoration: underline;
    }
  </style>
</head>
<body>
  <a href="#">Hover over me</a>
</body>
</html>
```

11. Pseudo-elements: ::pseudo-element

- Selects specific parts of an element.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p::before {
      content: "Before text: ";
    }
  </style>
</head>
<body>
  <p>This is regular text.</p>
</body>
</html>
```

12. Grouping Selector: `selector1, selector2, ...

- Groups multiple selectors together to apply the same styles.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h1, h2, h3 {
      font-family: Arial, sans-serif;
    }
  </style>
</head>
<body>
  <h1>Heading 1</h1>
  <h2>Heading 2</h2>
  <h3>Heading 3</h3>
</body>
</html>
```

13. Not Selector: :not(selector)

- Selects elements that do not match a specified selector.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p:not(.exclude) {
      font-style: italic;
    }
  </style>
</head>
<body>
  <p>This is a normal paragraph.</p>
  <p class="exclude">This paragraph is excluded.</p>
  <p>This is another normal paragraph.</p>
```

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

14. Attribute Value Prefix Selector: [attribute^="value"]

- Selects elements with attribute values that start with a specified string.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    [href^="https://"] {
      color: blue;
    }
  </style>
</head>
<body>
  <a href="https://example.com">Visit Example</a>
  <a href="http://example2.com">Visit Example 2</a>
</body>
</html>
```

15. Attribute Value Substring Selector: [attribute*="value"]

- Selects elements with attribute values that contain a specified substring.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    [class*="button"] {
      background-color: gray;
    }
  </style>
</head>
<body>
  <button class="submit-button">Submit</button>
  <button class="cancel-button">Cancel</button>
</body>
</html>
```

16. Attribute Value Suffix Selector: [attribute$="value"]

- Selects elements with attribute values that end with a specified string.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    [src$=".jpg"] {
      border: 1px solid black;
    }
  </style>
```

```
</head>
<body>
  <img src="image1.jpg" alt="Image 1">
  <img src="image2.png" alt="Image 2">
</body>
</html>
```

17. First-of-type Selector: :first-of-type

- Selects the first occurrence of an element type within its parent.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p:first-of-type {
      font-weight: bold;
    }
  </style>
</head>
<body>
  <p>This is the first paragraph.</p>
  <p>This is the second paragraph.</p>
  <div>This is a div.</div>
</body>
</html>
```

18. Last-of-type Selector: :last-of-type

- Selects the last occurrence of an element type within its parent.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p:last-of-type {
      font-style: italic;
    }
  </style>
</head>
<body>
  <p>This is the first paragraph.</p>
  <p>This is the second paragraph.</p>
  <div>This is a div.</div>
</body></html>
```

19. Nth-of-type Selector: :nth-of-type(n)

- Selects elements based on their position within their parent.

Example:

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

```
    <style>
      p:nth-of-type(2n) {
        color: red;
      }
    </style>
  </head>
  <body>
    <p>This is a normal paragraph.</p>
    <p>This paragraph is red.</p>
    <p>This paragraph is normal again.</p>
  </body></html>
```

20. First-child Selector: :first-child

- Selects elements that are the first child of their parent.

Example:

```
  <!DOCTYPE html>
  <html>
  <head>
    <style>
      p:first-child {
        font-weight: bold;
      }
    </style>
  </head>
  <body>
    <div>
      <p>This is the first child paragraph.</p>
      <p>This is another paragraph.</p>
    </div>
  </body></html>
```

21. Last-child Selector: :last-child

- Selects elements that are the last child of their parent.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p:last-child {
      font-style: italic;
    }
  </style>
</head>
<body>
  <div>
    <p>This is the first child paragraph.</p>
    <p>This paragraph is italic.</p>
  </div>
</body></html>
```

CSS Properties

❖ CSS Background:

❖CSS background properties:

- background-color
- background-image
- background-repeat
- background-position
- background-size
- background-attachment
- background-origin
- background-clip
- background-blend-mode

❖ syntax:

1. background-color:

- Syntax: `background-color: color;`

Example:

```
.container {
  background-color: #3498db;
}
```

2. background-image:

- Syntax: `background-image: url('image-path');`

Example:

```
.container {
  background-image: url('background-image.jpg');
}
```

3. background-repeat:

- Syntax: `background-repeat: repeat|repeat-x|repeat-y|no-repeat;`

Example:

```
.container {
  background-repeat: no-repeat;
}
```

4. background-position:

- Syntax: `background-position: x-position y-position;`

Example:

```
.container {
  background-position: center top;
}
```

5. background-size:

- Syntax: `background-size: auto|cover|contain|width height;`

Example:

```
.container {
```

```
  background-size: cover;
 }
```

6.background-attachment:

- Syntax: `background-attachment: scroll|fixed|local;`

Example:

```
.container {
  background-attachment: fixed;
 }
```

7. background-origin:

Syntax: `background-origin: padding-box|border-box|content-box;`

Example:

```
.container {
  background-origin: padding-box;
 }
```

8. background-clip:

- Syntax: `background-clip: padding-box|border-box|content-box;`

Example:

```
.container {
  background-clip: content-box;
 }
```

9. background-blend-mode:

- Syntax: `background-blend-mode: normal|multiply|screen|overlay|...;`

Example:

```
.container {
  background-blend-mode: multiply;
 }
```

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>CSS Background Example</title>
 <style>
  .container {
   width: 300px;
   height: 200px;
   background-color: #3498db;
   background-image: url('background-image.jpg');
   background-repeat: no-repeat;
   background-position: center top;
   background-size: cover;
   background-attachment: fixed;
   background-origin: padding-box;
   background-clip: content-box;
   background-blend-mode: multiply;
  }
 </style>
```

```
</head>
<body>
 <div class="container">
  <!-- Content goes here -->
 </div>
</body>
</html>
```

1.Example of background-color` property in HTML and CSS:

- raj.html

```
<!DOCTYPE html>
<html>
<head>
 <title>Background Color Example</title>
 <style>
  /* Define a CSS class with a background color */
  .colored-box {
   width: 300px;
   height: 100px;
   background-color: #3498db; /* Set the background color to a shade of blue */
   color: #fff; /* Set the text color to white for better visibility */
   text-align: center; /* Center align the text */
   line-height: 100px; /* Vertical alignment for the text */
  }
 </style>
</head>
<body>
 <!-- Create an HTML element with the CSS class applied -->
 <div class="colored-box">
  This is a colored box with a blue background.
 </div>
</body>
</html>
```

2.Background Image Example

- **Raj.html**

```
<!DOCTYPE html>
<html>
<head>
 <title>Background Image Example</title>
 <style>
  /* Define a CSS class with a background image */
  .image-box {
   width: 300px;
   height: 200px;
   background-image: url('background-image.jpg'); /* Set the background image */
   background-size: cover; /* Adjust the size to cover the container */
   background-repeat: no-repeat; /* Prevent image from repeating */
   background-position: center center; /* Center the image horizontally and vertically */
```

```
    color: #fff; /* Set the text color to white for better visibility */
    text-align: center; /* Center align the text */
    line-height: 200px; /* Vertical alignment for the text */
   }
  </style>
 </head>
 <body>
  <!-- Create an HTML element with the CSS class applied -->
  <div class="image-box">
   This is a div with a background image.
  </div>
 </body>
 </html>
```

3. background-repeat:

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>Background Repeat Example</title>
 <style>
  /* Define a CSS class with a background image and repeat property */
  .repeated-box {
   width: 300px;
   height: 200px;
   background-image: url('background-image.jpg'); /* Set the background image */
   background-repeat: repeat; /* Repeat the image both horizontally and vertically */
   color: #333; /* Set the text color to a darker shade */
   text-align: center; /* Center align the text */
   line-height: 200px; /* Vertical alignment for the text */
  }

  /* Define a CSS class with a background image and no-repeat property */
  .non-repeated-box {
   width: 300px;
   height: 200px;
   background-image: url('background-image.jpg'); /* Set the background image */
   background-repeat: no-repeat; /* Prevent image from repeating */
   color: #fff; /* Set the text color to white for better visibility */
   text-align: center; /* Center align the text */
   line-height: 200px; /* Vertical alignment for the text */
  }
 </style>
</head>
<body>
 <!-- Create an HTML element with the repeated background -->
 <div class="repeated-box">
  This is a div with a repeated background image.
```

```
</div>
<!-- Create an HTML element with the non-repeated background -->
<div class="non-repeated-box">
  This is a div with a non-repeated background image.
</div>
</body>
</html>
```

4.background-position:

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Background Position Example</title>
  <style>
    /* Define a CSS class with a background image and custom position */
    .positioned-box {
      width: 300px;
      height: 200px;
      background-image: url('background-image.jpg'); /* Set the background image */
      background-repeat: no-repeat; /* Prevent image from repeating */
      background-position: right top; /* Position the image at the top-right corner */
      color: #fff; /* Set the text color to white for better visibility */
      text-align: center; /* Center align the text */
      line-height: 200px; /* Vertical alignment for the text */
    }
  </style>
</head>
<body>
  <!-- Create an HTML element with the custom background position -->
  <div class="positioned-box">
    This is a div with a custom background position.
  </div>
</body>
</html>
```

5.background-position:

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Background Position Example</title>
  <style>
    /* Define a CSS class with a background image and custom position */
    .positioned-box {
      width: 300px;
      height: 200px;
      background-image: url('background-image.jpg'); /* Set the background image */
      background-repeat: no-repeat; /* Prevent image from repeating */
```

```
    background-position: right top; /* Position the image at the top-right corner */
    color: #fff; /* Set the text color to white for better visibility */
    text-align: center; /* Center align the text */
    line-height: 200px; /* Vertical alignment for the text */
   }
  </style>
 </head>
 <body>
  <!-- Create an HTML element with the custom background position -->
  <div class="positioned-box">
   This is a div with a custom background position.
  </div>
 </body>
 </html>
```

6.background-size:

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>Background Size Example</title>
 <style>
  /* Define a CSS class with a background image and custom size */
  .sized-box {
   width: 300px;
   height: 200px;
   background-image: url('background-image.jpg'); /* Set the background image */
   background-repeat: no-repeat; /* Prevent image from repeating */
   background-size: 100% 100%; /* Set the background size to cover the container */
   color: #fff; /* Set the text color to white for better visibility */
   text-align: center; /* Center align the text */
   line-height: 200px; /* Vertical alignment for the text */
  }
 </style>
</head>
<body>
 <!-- Create an HTML element with the custom background size -->
 <div class="sized-box">
  This is a div with a custom background size.
 </div>
</body>
</html>
```

7.background-attachment:

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>Background Attachment Example</title>
```

```
 <style>
  /* Define a CSS class with a background image and custom attachment behavior */
  .attached-box {
    width: 300px;
    height: 200px;
    background-image: url('background-image.jpg'); /* Set the background image */
    background-repeat: no-repeat; /* Prevent image from repeating */
    background-size: cover; /* Set the background size to cover the container */
    background-attachment: fixed; /* Fixed background attachment */
    color: #fff; /* Set the text color to white for better visibility */
    text-align: center; /* Center align the text */
    line-height: 200px; /* Vertical alignment for the text */
   }
 </style>
</head>
<body>
 <!-- Create an HTML element with the fixed background attachment -->
 <div class="attached-box">
  This is a div with a fixed background attachment.
 </div>
</body>
</html>
```

8.background-origin:

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>Background Origin Example</title>
 <style>
  /* Define a CSS class with a background image and custom origin */
  .origin-box {
    width: 300px;
    height: 200px;
    background-image: url('background-image.jpg'); /* Set the background image */
    background-repeat: no-repeat; /* Prevent image from repeating */
    background-size: cover; /* Set the background size to cover the container */
    background-origin: content-box; /* Set the background origin to content-box */
    background-clip: content-box; /* Set the background clip to content-box */
    color: #fff; /* Set the text color to white for better visibility */
    text-align: center; /* Center align the text */
    line-height: 200px; /* Vertical alignment for the text */
   }
 </style>
</head>
<body>
 <!-- Create an HTML element with the custom background origin -->
 <div class="origin-box">
```

```
    This is a div with a custom background origin.
  </div>
</body>
</html>
```

9. background-clip:

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Background Clip Example</title>
  <style>
    /* Define a CSS class with a background image and custom clip behavior */
    .clipped-box {
      width: 300px;
      height: 200px;
      background-image: url('background-image.jpg'); /* Set the background image */
      background-repeat: no-repeat; /* Prevent image from repeating */
      background-size: cover; /* Set the background size to cover the container */
      background-clip: content-box; /* Set the background clip to content-box */
      color: #fff; /* Set the text color to white for better visibility */
      text-align: center; /* Center align the text */
      line-height: 200px; /* Vertical alignment for the text */
    }
  </style>
</head>
<body>
  <!-- Create an HTML element with the custom background clip -->
  <div class="clipped-box">
    This is a div with a custom background clip.
  </div>
</body>
</html>
```

10.background-blend-mode:

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Background Blend Mode Example</title>
  <style>
    /* Define a CSS class with a background image and blend mode */
    .blend-box {
      width: 300px;
      height: 200px;
      background-image: url('background-image.jpg'); /* Set the background image */
      background-repeat: no-repeat; /* Prevent image from repeating */
      background-size: cover; /* Set the background size to cover the container */
      background-blend-mode: multiply; /* Apply the 'multiply' blend mode */
```

```
    color: #fff; /* Set the text color to white for better visibility */
    text-align: center; /* Center align the text */
    line-height: 200px; /* Vertical alignment for the text */
   }
  </style>
</head>
<body>
  <!-- Create an HTML element with the background blend mode applied -->
  <div class="blend-box">
    This is a div with a background blend mode.
  </div>
</body>
</html>
```

CSS Color:

- CSS (Cascading Style Sheets) provides a variety of ways to specify colors for styling web content.
- Colors are an essential part of web design and can greatly affect the visual appeal and user experience of a website.

1. Color Representation:

- In CSS, colors can be represented in several ways, including:
- Named Colors: CSS defines a set of 147 named colors that you can use directly in your styles. For example, color: red;.
- Hexadecimal Notation: Colors can be specified using a 6-character hexadecimal code. For example, color: #FF5733; represents a shade of orange.
- RGB Notation: Colors can be expressed using the RGB (Red, Green, Blue) model, with values ranging from 0 to 255 for each color channel. For example, color: rgb(255, 87, 51);.
- RGBA Notation: Similar to RGB, but with an additional alpha channel for transparency. For example, color: rgba(255, 87, 51, 0.5); specifies a semi-transparent color.
- HSL Notation: Colors can also be defined using the HSL (Hue, Saturation, Lightness) model, where hue is specified in degrees, saturation and lightness as percentages. For example, color: hsl(15, 100%, 50%); represents a shade of orange.
- HSLA Notation: Like HSL, but with an alpha channel for transparency. For example, color: hsla(15, 100%, 50%, 0.5);.

2. Color Usage:

- Colors are commonly used in CSS for various properties, including:
- color: Specifies the text color.
- background-color: Sets the background color of an element.
- border-color: Defines the color of an element's border.
- box-shadow: Specifies the color of a shadow effect.
- text-shadow: Sets the color of a text shadow.
- outline-color: Defines the color of an element's outline. And more.

3. Opacity and Transparency:

- CSS allows you to control the opacity and transparency of colors using the opacity property or by using RGBA or HSLA notations. An alpha (A) channel value of 0 means fully transparent, and 1 means fully opaque.

4. Color Names:

- CSS defines a set of named colors, making it easy to use common colors without specifying specific RGB values. Some examples include red, blue, green, black, white, and many more.

5. Gradients:

- CSS3 introduces gradient backgrounds, where you can create smooth transitions between two or more colors. Linear gradients and radial gradients are commonly used for backgrounds and can be specified using CSS properties like background-image and background.

6. Color Functions:

- CSS also provides color functions that allow you to manipulate colors dynamically. Examples include rgb(), rgba(), hsl(), hsla(), and more.

❖ Examples of different color representations in CSS:

1. Named Color:

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>Named Color Example</title>
 <style>
  .colored-text {
   color: red; /* Named color */
  }
 </style>
</head>
<body>
 <p class="colored-text">This is red text using a named color.</p>
</body>
</html>
```

2. Hexadecimal Notation:

```
<!DOCTYPE html>
<html>
<head>
 <title>Hexadecimal Color Example</title>
 <style>
  .colored-background {
   background-color: #3498db; /* Hexadecimal color */
  }
 </style>
</head>
<body>
 <div class="colored-background">
  This div has a background color using hexadecimal notation.
 </div>
</body>
</html>
```

3. RGB Notation:

```
<!DOCTYPE html>
<html>
<head>
 <title>RGB Color Example</title>
 <style>
  .colored-border {
   border: 3px solid rgb(255, 0, 0); /* RGB color */
  }
 </style>
</head>
<body>
```

```
  <div class="colored-border">
    This div has a red border using RGB notation.
  </div>
</body>
</html>
```

4. RGBA Notation:

```
<!DOCTYPE html>
<html>
<head>
  <title>RGBA Color Example</title>
  <style>
    .transparent-background {
      background-color: rgba(0, 128, 0, 0.5); /* RGBA color with transparency */
    }
  </style>
</head>
<body>
  <div class="transparent-background">
    This div has a semi-transparent green background using RGBA notation.
  </div>
</body>
</html>
```

5. HSL Notation:

```
<!DOCTYPE html>
<html>
<head>
  <title>HSL Color Example</title>
  <style>
    .colored-text {
      color: hsl(210, 100%, 50%); /* HSL color */
    }
  </style>
</head>
<body>
  <p class="colored-text">This is text with an HSL-based color.</p>
</body>
</html>
```

6. HSLA Notation:

```
<!DOCTYPE html>
<html>
<head>
  <title>HSLA Color Example</title>
  <style>
    .transparent-border {
      border: 3px solid hsla(120, 100%, 50%, 0.7); /* HSLA color with transparency */
    }
  </style>
```

```
</head>
<body>
  <div class="transparent-border">
    This div has a semi-transparent border using HSLA notation.
  </div>
</body>
</html>
```

❖Examples of different color usages in CSS:

1. Text Color:

```
<!DOCTYPE html>
<html>
<head>
  <title>Text Color Example</title>
  <style>
    .colored-text {
      color: blue; /* Text color */
    }
  </style>
</head>
<body>
  <p class="colored-text">This text has a blue color.</p>
</body>
</html>
```

2. Background Color:

```
<!DOCTYPE html>
<html>
<head>
  <title>Background Color Example</title>
  <style>
    .colored-background {
      background-color: #FF5733; /* Background color */
      color: white; /* Text color */
    }
  </style>
</head>
<body>
  <div class="colored-background">
    This div has an orange background and white text.
  </div>
</body>
</html>
```

3. Border Color:

```
<!DOCTYPE html>
<html>
<head>
  <title>Border Color Example</title>
```

```
  <style>
    .bordered-element {
      border: 2px solid green; /* Border color */
    }
  </style>
</head>
<body>
  <div class="bordered-element">
    This div has a green border.
  </div>
</body>
</html>
```

4. Box Shadow Color:

```
<!DOCTYPE html>
<html>
<head>
  <title>Box Shadow Color Example</title>
  <style>
    .shadowed-element {
      width: 200px;
      height: 100px;
      background-color: #3498db;
      box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.3); /* Box shadow color */
    }
  </style>
</head>
<body>
  <div class="shadowed-element">
    This div has a shadow with a custom color.
  </div>
</body>
</html>
```

5. Outline Color:

```
<!DOCTYPE html>
<html>
<head>
  <title>Outline Color Example</title>
  <style>
    .outlined-element {
      outline: 2px dotted red; /* Outline color */
    }
  </style>
</head>
<body>
  <div class="outlined-element">
    This div has a red dotted outline.
  </div>
```

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

6. Text Shadow Color:

```
<!DOCTYPE html>
<html>
<head>
  <title>Text Shadow Color Example</title>
  <style>
    .text-shadowed {
      text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5); /* Text shadow color */
    }
  </style>
</head>
<body>
  <h2 class="text-shadowed">This heading has a shadow with a custom color.</h2>
</body>
</html>
```

❖ Example of css color Opacity and Transparency:

- how to use opacity and transparency:

1. Opacity Property:

- The `opacity` property is used to control the overall opacity of an element and its content.

```
<!DOCTYPE html>
<html>
<head>
  <title>Opacity Example</title>
  <style>
    .transparent-box {
      background-color: blue;
      color: white;
      opacity: 0.5; /* 50% opacity */
    }
  </style>
</head>
<body>
  <div class="transparent-box">
    This is a div with reduced opacity.
  </div>
</body>
</html>
```

- In this example, the `opacity` property is set to `0.5`, making the blue box and its content 50% transparent.

2. RGBA Notation:

- You can also achieve transparency by using the `rgba()` notation for colors.

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

```
  <title>RGBA Example</title>
  <style>
    .transparent-box {
      background-color: rgba(255, 0, 0, 0.5); /* Red color with 50% opacity */
      color: white;
    }
  </style>
</head>
<body>
  <div class="transparent-box">
    This is a div with a semi-transparent red background.
  </div>
</body>
</html>
```

- In this example, `rgba(255, 0, 0, 0.5)` represents a red color with 50% opacity.

3. Background and Text Transparency:

- You can apply transparency independently to the background and text.

```
<!DOCTYPE html>
<html>
<head>
  <title>Background and Text Transparency Example</title>
  <style>
    .transparent-box {
      background-color: rgba(0, 128, 0, 0.7); /* Green background with 70% opacity */
      color: rgba(0, 0, 0, 0.5); /* Black text with 50% opacity */
    }
  </style>
</head>
<body>
  <div class="transparent-box">
    This is a div with a semi-transparent background and text.
  </div>
</body>
</html>
```

4.Transparency in Images:

- You can also apply transparency to images using the `opacity` property.

```
<!DOCTYPE html>
<html>
<head>
  <title>Image Transparency Example</title>
  <style>
    .transparent-image {
      opacity: 0.5; /* 50% opacity for the image */
    }
  </style>
</head>
<body>
```

```
  <img class="transparent-image" src="image.png" alt="Transparent Image" />
 </body>
 </html>
```

❖Example of css color name:

1. Text Color Using Named Color:

- You can set the text color using a named color.

```
<!DOCTYPE html>
<html>
<head>
  <title>Text Color with Named Color</title>
  <style>
    .colored-text {
      color: crimson; /* Named color */
    }
  </style>
</head>
<body>
  <p class="colored-text">This text is in crimson color.</p>
</body>
</html>
```

2. Background Color Using Named Color:

- You can set the background color of an element using a named color.

```
<!DOCTYPE html>
<html>
<head>
  <title>Background Color with Named Color</title>
  <style>
    .colored-background {
      background-color: gold; /* Named color */
      color: black; /* Text color */
    }
  </style>
</head>
<body>
  <div class="colored-background">
    This div has a gold background color.
  </div>
</body>
</html>
```

3. Border Color Using Named Color:

- You can set the border color of an element using a named color.

```
<!DOCTYPE html>
<html>
<head>
  <title>Border Color with Named Color</title>
  <style>
```

```
    .bordered-element {
      border: 2px solid seagreen; /* Named color */
    }
  </style>
</head>
<body>
  <div class="bordered-element">
    This div has a seagreen border.
  </div>
</body>
</html>
```

4. Box Shadow Color Using Named Color:

- You can set the color of a box shadow using a named color.

```
<!DOCTYPE html>
<html>
<head>
  <title>Box Shadow Color with Named Color</title>
  <style>
    .shadowed-element {
      width: 200px;
      height: 100px;
      background-color: lightblue;
      box-shadow: 5px 5px 10px darkgray; /* Named color for shadow */
    }
  </style>
</head>
<body>
  <div class="shadowed-element">
    This div has a box shadow with a darkgray color.
  </div>
</body>
</html>
```

5. Outline Color Using Named Color:

- You can set the outline color of an element using a named color.

```
<!DOCTYPE html>
<html>
<head>
  <title>Outline Color with Named Color</title>
  <style>
    .outlined-element {
      outline: 2px dashed royalblue; /* Named color for outline */
    }
  </style>
</head>
<body>
  <div class="outlined-element">
    This div has a royalblue dashed outline.
```

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

❖ Example of css color Gradients:

1. Linear Gradient Background:

- You can create a linear gradient background using the `linear-gradient()` function.

```
<!DOCTYPE html>
<html>
<head>
  <title>Linear Gradient Example</title>
  <style>
    .gradient-box {
      width: 300px;
      height: 200px;
      background: linear-gradient(to bottom, #3498db, #e74c3c); /* Linear gradient */
      color: white;
      text-align: center;
      line-height: 200px;
    }
  </style>
</head>
<body>
  <div class="gradient-box">
    This is a div with a linear gradient background.
  </div>
</body>
</html>
```

- In this example, a linear gradient background starts from `#3498db` (light blue) at the top and transitions to `#e74c3c` (red) at the bottom.

2. Radial Gradient Background:

- You can create a radial gradient background using the `radial-gradient()` function.

```
<!DOCTYPE html>
<html>
<head>
  <title>Radial Gradient Example</title>
  <style>
    .gradient-circle {
      width: 200px;
      height: 200px;
      background: radial-gradient(circle, #2ecc71, #e74c3c); /* Radial gradient */
      color: white;
      text-align: center;
      line-height: 200px;
    }
  </style>
</head>
<body>
```

```
 <div class="gradient-circle">
  This div has a radial gradient background.
 </div>
</body>
</html>
```

- In this example, a radial gradient creates a circle-shaped background starting from `#2ecc71` (green) at the center and transitioning to `#e74c3c` (red) at the edges.

3. Multiple Color Stops in Gradient:

- You can specify multiple color stops in a gradient to create more complex effects.

```
<!DOCTYPE html>
<html>
<head>
 <title>Multiple Color Stops Example</title>
 <style>
  .gradient-box {
   width: 300px;
   height: 200px;
   background: linear-gradient(to right, #3498db, #e74c3c, #9b59b6); /* Multiple color stops */
   color: white;
   text-align: center;
   line-height: 200px;
  }
 </style>
</head>
<body>
 <div class="gradient-box">
  This div has a gradient with multiple color stops.
 </div>
</body>
</html>
```

- In this example, a linear gradient moves from `#3498db` (light blue) to `#e74c3c` (red) and then to `#9b59b6` (purple) from left to right.

4. Gradient as Text Color:

- Gradients can also be used as text colors.

```
<!DOCTYPE html>
<html>
<head>
 <title>Gradient Text Color Example</title>
 <style>
  .gradient-text {
   background: linear-gradient(to right, #3498db, #e74c3c); /* Gradient background */
   -webkit-background-clip: text; /* Clip text to gradient background */
   color: transparent; /* Make text color transparent */
  }
 </style>
</head>
<body>
```

```
  <h1 class="gradient-text">Gradient Text</h1>
 </body>
 </html>
```

- In this example, a linear gradient background is applied to the text, creating a gradient text effect.

❖ Example of css Color Functions:

1. rgb()` Function:

- The `rgb()` function allows you to specify colors using the Red, Green, and Blue color channels.

```
<!DOCTYPE html>
<html>
<head>
 <title>RGB Color Function Example</title>
 <style>
  .colored-box {
   background-color: rgb(255, 0, 0); /* Red color using rgb() function */
   color: white;
   text-align: center;
   padding: 20px;
  }
 </style>
</head>
<body>
 <div class="colored-box">
  This div has a red background color using the rgb() function.
 </div>
</body>
</html>
```

2. rgba()` Function:

- The `rgba()` function allows you to specify colors with an additional alpha channel for transparency.

```
<!DOCTYPE html>
<html>
<head>
 <title>RGBA Color Function Example</title>
 <style>
  .transparent-box {
   background-color: rgba(0, 128, 0, 0.5); /* Semi-transparent green using rgba() function */
   color: white;
   text-align: center;
   padding: 20px;
  }
 </style>
</head>
<body>
 <div class="transparent-box">
  This div has a semi-transparent green background color using the rgba() function.
 </div>
```

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

3. hsl()` Function:

- The `hsl()` function allows you to specify colors using the Hue, Saturation, and Lightness color model.

```
<!DOCTYPE html>
<html>
<head>
 <title>HSL Color Function Example</title>
 <style>
  .colored-text {
   color: hsl(210, 100%, 50%); /* Blue color using hsl() function */
   text-align: center;
   padding: 20px;
  }
 </style>
</head>
<body>
 <p class="colored-text">This text is in blue using the hsl() function.</p>
</body>
</html>
```

4. hsla()` Function:

- The `hsla()` function allows you to specify colors with an additional alpha channel for transparency.

```
<!DOCTYPE html>
<html>
<head>
 <title>HSLA Color Function Example</title>
 <style>
  .transparent-text {
   color: hsla(45, 100%, 50%, 0.6); /* Semi-transparent orange using hsla() function */
   text-align: center;
   padding: 20px;
  }
 </style>
</head>
<body>
 <p class="transparent-text">This text is semi-transparent orange using the hsla() function.</p>
</body>
</html>
```

CSS BOX MODEL:

- The CSS Box Model is a fundamental concept in web design and layout. It describes how elements on a web page are structured in terms of content, padding, borders, and margins.

image of box model:

❖ Content Area:

- The content area is the innermost part of an HTML element and contains the actual content, such as text, images, or other HTML elements.
- It is defined by the element's width and height properties.
- सामग्री क्षेत्र HTML तत्व का सबसे आंतरिक भाग है और इसमें वास्तविक सामग्री, जैसे पाठ, चित्र या अन्य HTML तत्व शामिल होते हैं।
- इसे तत्व की चौड़ाई और ऊंचाई गुणों द्वारा परिभाषित किया गया है।

❖ Padding:

- Padding is the space between the content area and the element's border.
- It can be adjusted using the padding property.
- Padding is used to create space between the content and the element's border, improving readability and aesthetics.
- पैडिंग सामग्री क्षेत्र और तत्व की सीमा के बीच का स्थान है।
- इसे पैडिंग प्रॉपर्टी का उपयोग करके समायोजित किया जा सकता है।
- पैडिंग का उपयोग सामग्री और तत्व की सीमा के बीच जगह बनाने, पठनीयता और सौंदर्यशास्त्र में सुधार करने के लिए किया जाता है।

❖ Padding Properties:

- padding-top, padding-right, padding-bottom, padding-left:
- These properties allow you to set padding for each side of an element individually.
- You can specify values in different units such as pixels (px), ems (em), percentages (%), etc.

Example:

```
.box {
  padding-top: 10px;     /* Padding on the top */
  padding-right: 20px;    /* Padding on the right */
  padding-bottom: 15px;   /* Padding on the bottom */
  padding-left: 30px;     /* Padding on the left */
}
```

Example:

```
.box {
 padding: 10px;        /* Equal padding on all sides */
 /* Or */
 padding: 10px 20px;    /* 10px top/bottom, 20px left/right */
 /* Or */
 padding: 10px 20px 15px 30px; /* Individual padding for all sides */
}
```

❖ Example with html:

- Equal Padding on All Sides:

```
<!DOCTYPE html>
<html>
<head>
 <title>Equal Padding Example</title>
 <style>
  .box {
   width: 200px;
   height: 100px;
   background-color: lightblue;
   padding: 10px; /* Equal padding on all sides */
  }
 </style>
</head>
<body>
 <div class="box">
  <p>This is a box with equal padding on all sides.</p>
 </div>
</body>
</html>
```

- In this example, the `padding` property is set to `10px`, which adds equal padding to all four sides of the `<div>`.
- Top/Bottom Padding and Left/Right Padding:

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>Top/Bottom and Left/Right Padding Example</title>
 <style>
  .box {
   width: 200px;
   height: 100px;
   background-color: lightblue;
   padding: 10px 20px; /* 10px top/bottom, 20px left/right */
  }
 </style>
</head>
<body>
```

```
 <div class="box">
  <p>This is a box with top/bottom and left/right padding.</p>
 </div>
</body>
</html>
```

- In this example, the `padding` property is set to `10px 20px`, which adds `10px` of padding to the top and bottom and `20px` of padding to the left and right of the `<div>`.

❖ Individual Padding for All Sides:

```
<!DOCTYPE html>
<html>
<head>
 <title>Individual Padding Example</title>
 <style>
  .box {
   width: 200px;
   height: 100px;
   background-color: lightblue;
   padding: 10px 20px 15px 30px; /* Individual padding for all sides */
  }
 </style>
</head>
<body>
 <div class="box">
  <p>This is a box with individual padding for all sides.</p>
 </div>
</body>
</html>
```

- In this example, the `padding` property is set to `10px 20px 15px 30px`, specifying individual padding values for the top, right, bottom, and left sides of the `<div>`.

Border:

- The border is a line that surrounds the padding and content areas.
- It can be customized using the border property.
- Borders are often used to visually separate elements or create boundaries around content.
- सीमा एक रेखा है जो पैडिंग और सामग्री क्षेत्रों को घेरती है।
- इसे बॉर्डर प्रॉपर्टी का उपयोग करके अनुकूलित किया जा सकता है।
- सीमाओं का उपयोग अक्सर तत्वों को दृष्टिगत रूप से अलग करने या सामग्री के चारों ओर सीमाएँ बनाने के लिए किया जाता है।

❖ border-width:

- The border-width property sets the width of the border.
- You can specify the width using values like px, em, rem, or keywords like thin, medium, or thick.
- border-style:
- The border-style property sets the style of the border, such as solid, dashed, dotted, etc.
- Common values include solid, dashed, dotted, double, groove, ridge, inset, and outset.

❖ border-color:

- The border-color property sets the color of the border.
- You can specify the color using color names, hexadecimal values, RGB values, or other valid color representations.

❖ border (shorthand):

- The border property is a shorthand property that combines border-width, border-style, and border-color in one declaration.
- You can set all three properties in one line, and they will be applied in the order of width, style, and color.

❖ Individual Border Widths:

- border-top-width: Sets the width of the top border.
- border-right-width: Sets the width of the right border.
- border-bottom-width: Sets the width of the bottom border.
- border-left-width: Sets the width of the left border.

Example:

```
.box {
  border-top-width: 2px;
  border-right-width: 4px;
  border-bottom-width: 3px;
  border-left-width: 1px;
}
```

❖ Shorthand Border Width:

```
.box {
  border-width: 2px;          /* Equal width on all sides */
  /* Or */
  border-width: 2px 4px;      /* Vertical (top/bottom) and horizontal (left/right) width */
  /* Or */
  border-width: 2px 4px 3px 1px; /* Individual width for top, right, bottom, left */
}
```

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Border Width Example</title>
  <style>
    .box {
      width: 200px;
      height: 100px;
      background-color: lightblue;
      /* Border properties */
      border-top-width: 2px;
      border-right-width: 4px;
      border-bottom-width: 3px;
      border-left-width: 1px;
      border-style: solid;
      border-color: red;
```

```
    }
  </style>
</head>
<body>
  <div class="box">
    <p>Border Width Example</p>
  </div>
</body>
</html>
```

❖ Explanation of the border-style property:

- CSS `border-style` Property:
- The `border-style` property sets the style of the border, specifying whether it should be solid, dashed, dotted, double, etc.

❖ Common Values for `border-style`:

1. none:
   - No border is displayed. The element has no visible border.
2. hidden:
   - Similar to `none`, but may leave a gap where the border would have been.
3. dotted:
   - The border is a series of dots.
4. dashed:
   - The border is a series of short dashes.
5. solid:
   - The border is a solid line.
6. double:
   - The border consists of two parallel lines.
7. groove:
   - The border appears to be carved into the page.
8. ridge:
   - The border appears to be raised from the page.
9. inset:
   - The border appears to be pressed into the page.
10. outset:
    - The border appears to be coming out of the page.
11. initial:
    - Sets the border style to its default value.
12. inherit:
    - Inherits the border style from its parent element.

❖ Multiple Values:

- You can specify different styles for each side of the element using the individual properties:
- order-top-style
- border-right-style
- border-bottom-style
- border-left-style

Example:

```
.box {
  border-top-style: dashed;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: double;
}
```

❖ Shorthand Property:

- The `border-style` property can also be used as a shorthand property to set the border style for all sides at once.
- You can specify one value for all sides or up to four values to set styles individually.

Example:

```
.box {
  border-style: dashed;      /* Equal style on all sides */
  /* Or */
  border-style: dashed solid dotted double; /* Individual style for top, right, bottom, left */
}
```

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Border Style Example</title>
  <style>
    .box {
      width: 200px;
      height: 100px;
      background-color: lightblue;
      /* Border properties */
      border-width: 2px;
      border-style: dotted; /* Set the border style */
      border-color: red;
    }
  </style>
</head>
<body>
  <div class="box">
    <p>Border Style Example</p>
  </div>
</body>
</html>
```

❖ CSS border-color Property:

- Individual Border Colors:
- To set individual border colors for each side of an element, you can use the following properties:
- border-top-color: Sets the color of the top border.
- border-right-color: Sets the color of the right border.
- border-bottom-color: Sets the color of the bottom border.
- border-left-color: Sets the color of the left border.

Example:

```
.box {
  border-top-color: red;
  border-right-color: green;
  border-bottom-color: blue;
  border-left-color: yellow;
}
Multiple Values:
.box {
  border-color: red;      /* Equal color on all sides */
  /* Or */
  border-color: red green blue yellow; /* Individual color for top, right, bottom, left */
}
```

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Border Color Example</title>
  <style>
    .box {
      width: 200px;
      height: 100px;
      background-color: lightblue;
      /* Border properties */
      border-width: 2px;
      border-style: solid;
      border-color: red; /* Set the border color */
    }
  </style>
</head>
<body>
  <div class="box">
    <p>Border Color Example</p>
  </div>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>Individual Border Properties Example</title>
 <style>
  .box {
   width: 200px;
   height: 100px;
   background-color: lightblue;
   /* Individual border properties for each side */
   border-top-width: 2px;
   border-top-style: solid;
   border-top-color: red;
   border-right-width: 4px;
   border-right-style: dashed;
   border-right-color: green;
   border-bottom-width: 3px;
   border-bottom-style: dotted;
   border-bottom-color: blue;
   border-left-width: 1px;
   border-left-style: double;
   border-left-color: yellow;
  }
 </style>
</head>
<body>
 <div class="box">
  <p>Individual Border Properties Example</p>
 </div>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Document</title>
 <style>
  div{
   width: 100px;
   height: 100px;
   border: 3px dotted green;
   background-color: blue;
  }
  p{
```

```
    width: 100px;
    height: 100px 100px;
    border-top: 30px double red;
    border-bottom: 5px dashed yellow;
    border-left: 4px groove orange;
    border-right: 4px solid yellow;
    background-color: aqua; }
  </style>
</head>
<body>
  <div>Bharat</div>
  <p>Namste!</p>
</body>
</html>
```

Margin:

- The margin is the space outside the border of an element.
- It can be controlled with the margin property.
- Margins provide separation between elements, defining the space between adjacent elements on a web page.
- मार्जिन किसी तत्व की सीमा के बाहर का स्थान है।
- इसे मार्जिन प्रॉपर्टी से नियंत्रित किया जा सकता है।
- मार्जिन तत्वों के बीच अलगाव प्रदान करते हैं, एक वेब पेज पर आसन्न तत्वों के बीच की जगह को परिभाषित करते हैं।

❖ CSS margin Property:

- The margin property sets the margin space on all four sides (top, right, bottom, left) of an HTML element.
- Values for margin:
- You can specify margin values in various units, including pixels (px), ems (em), rems (rem), percentages (%), and keywords.

Single Value:

➢ If you provide a single value, it sets the margin for all four sides equally.

```
.box {
  margin: 20px; /* Equal margin on all sides */
}
```

Two Values:

- If you provide two values, the first value represents the margin for the top and bottom, and the second value represents the margin for the left and right.

```
.box {
  margin: 10px 20px; /* 10px top/bottom, 20px left/right */
}
```

Three Values:

- When you provide three values, they represent the margin for the top, left/right, and bottom, respectively.

```
.box {
  margin: 5px 10px 15px; /* 5px top, 10px left/right, 15px bottom */
}
```

Four Values:

- You can provide all four values individually to specify the margin for each side.

```
.box {
  margin: 5px 10px 15px 20px; /* top, right, bottom, left */
}
```

❖ Negative Margins:

- Negative margin values are allowed and can be used to overlap elements or pull elements closer to each other.

➢ Auto Value:

- The auto value can be used to horizontally center an element within its container.
- The margin property is not inherited, meaning that it doesn't inherit margin values from parent elements.

❖ Collapsing Margins:

- Margin values of adjacent elements can collapse, resulting in the larger of the two margins being used. This can affect spacing in certain situations.

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>Margin Example</title>
 <style>
  .box {
   width: 200px;
   height: 100px;
   background-color: lightblue;
   /* Margin properties */
   margin: 20px; /* Equal margin on all sides */
  }
  .box2 {
   width: 200px;
   height: 100px;
   background-color: lightcoral;
   /* Margin properties with different values */
   margin: 10px 20px; /* 10px top/bottom, 20px left/right */
  }
  .box3 {
   width: 200px;
   height: 100px;
   background-color: lightgreen;
   /* Margin properties with individual values */
   margin-top: 5px;
   margin-right: 15px;
   margin-bottom: 10px;
   margin-left: 25px;
  }
 </style>
```

```
</head>
<body>
 <div class="box">
  <p>Box 1 with equal margin on all sides.</p>
 </div>
 <div class="box2">
  <p>Box 2 with different margin values for top/bottom and left/right.</p>
 </div>
 <div class="box3">
  <p>Box 3 with individual margin values for each side.</p>
 </div>
</body>
</html>
```

❖ How the CSS Box Model is calculated:

- Total Width: The total width of an element is calculated as follows:
- Total Width = Width + (Left Padding) + (Right Padding) + (Left Border) + (Right Border) + (Left Margin) + (Right Margin)
- Total Height: The total height of an element is calculated as follows:
- Total Height = Height + (Top Padding) + (Bottom Padding) + (Top Border) + (Bottom Border) + (Top Margin) + (Bottom Margin)
- Key points to note about the CSS Box Model:
- The width and height properties determine the size of the content area.
- Padding adds space inside the element, but it doesn't change the total size of the element.
- Borders are drawn around the padding area.
- Margins are the space outside the border and define the spacing between elements.
- The box model is applied to all HTML elements, including block-level and inline elements.
- When working with the box model, it's essential to keep in mind that certain CSS properties, such as box-sizing, can affect how the box model behaves. The box-sizing property can be set to content-box (default) or border-box. When set to border-box, the total width and height of an element include the padding and border, making it easier to control the size of elements.

Explanation:

- Total Width = Width + (Left Padding) + (Right Padding) + (Left Border) + (Right Border) + (Left Margin) + (Right Margin)

Width:

- The Width represents the specified or computed width of the content area of the HTML element.
- It defines how wide the actual content, such as text or images, should be within the element.

Left Padding and Right Padding:

- The Left Padding and Right Padding represent the space between the content area and the element's inner border on the left and right sides, respectively.
- Padding is used to create space around the content, separating it from the border.

Left Border and Right Border:

- The Left Border and Right Border represent the width of the element's border on the left and right sides, respectively.
- Borders can be styled using CSS properties like border-width, border-style, and border-color.

Left Margin and Right Margin:

- The Left Margin and Right Margin represent the space between the element's outer border and adjacent elements (or the edge of the containing block) on the left and right sides, respectively.
- Margins control the spacing between elements on a web page and help define the layout.

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>Box Model Example</title>
 <style>
  .box {
   width: 200px;          /* Width */
   padding-left: 20px;     /* Left Padding */
   padding-right: 30px;    /* Right Padding */
   border-left: 5px solid red;   /* Left Border */
   border-right: 5px solid blue; /* Right Border */
   margin-left: 10px;      /* Left Margin */
   margin-right: 15px;     /* Right Margin */
   background-color: lightgray;
  }
 </style>
</head>
<body>
 <div class="box">This is a box.</div>
</body>
</html>
Total Width = 200px + 20px + 30px + 5px + 5px + 10px + 15px
Total Width = 275px
```

Explanation:

1. Height:

- The `Height` represents the specified or computed height of the content area of the HTML element.
- It defines how tall the actual content, such as text or images, should be within the element.

2. Top Padding and Bottom Padding:

- The `Top Padding` and `Bottom Padding` represent the space between the content area and the element's inner border on the top and bottom sides, respectively.
- Padding is used to create space around the content, separating it from the border.

3. Top Border and Bottom Border:

- The `Top Border` and `Bottom Border` represent the width of the element's border on the top and bottom sides, respectively.
- Borders can be styled using CSS properties like `border-width`, `border-style`, and `border-color`.

4. Top Margin and Bottom Margin:

- The `Top Margin` and `Bottom Margin` represent the space between the element's outer border and adjacent elements (or the edge of the containing block) on the top and bottom sides, respectively.
- Margins control the spacing between elements on a web page and help define the layout.

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>Box Model Height Example</title>
 <style>
  .box {
   height: 100px;           /* Height */
   padding-top: 20px;       /* Top Padding */
   padding-bottom: 30px;    /* Bottom Padding */
   border-top: 5px solid red;   /* Top Border */
   border-bottom: 5px solid blue; /* Bottom Border */
   margin-top: 10px;        /* Top Margin */
   margin-bottom: 15px;     /* Bottom Margin */
   background-color: lightgray;
  }
 </style>
</head>
<body>
 <div class="box">This is a box.</div>
</body>
</html>
```

In this example:

- The `height` property is set to `100px`, defining the height of the content area.
- `padding-top` and `padding-bottom` add space above and below the content.
- `border-top` and `border-bottom` create borders on the top and bottom.
- `margin-top` and `margin-bottom` specify margins above and below the element.

➢ **When you calculate the total height using the formula:**

- Total Height = Height + (Top Padding) + (Bottom Padding) + (Top Border) + (Bottom Border) + (Top Margin) + (Bottom Margin)

➢ **Substituting the values:**

- Total Height = 100px + 20px + 30px + 5px + 5px + 10px + 15px

Total Height = 185px

Box-sizing:

- box-sizing property is used to control how the total width and height of an element are calculated, including padding and borders.
- It determines whether the specified width and height values include or exclude the padding and border of the element.

❖ box-sizing` Property:

- The box-sizing property defines how the total width and height of an element are calculated.

Values for box-sizing

1. content-box` (Default):

- This is the default value. The width and height of the element's content area do not include padding or borders. Padding and borders are added to the specified width and height.

```
.box {
  box-sizing: content-box;
}
```

2. border-box:

- In this mode, the specified width and height include both the content area and any padding or border. Padding and borders are included in the total width and height.

```
.box {
  box-sizing: border-box;
}
```

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Box Sizing Example</title>
  <style>
    .box {
      width: 200px;
      height: 100px;
      background-color: lightblue;
      /* Box sizing property */
      box-sizing: border-box;
      padding: 20px;
      border: 5px solid red;
    }
  </style>
</head>
<body>
  <div class="box">
    <p>Box Sizing Example</p>
  </div>
</body>
</html>
```

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>Box Sizing Example</title>
 <style>
  /* Global box-sizing setting */
  html {
   box-sizing: border-box;
  }
  *, *::before, *::after {
   box-sizing: inherit;
  }
  /* Container div with padding */
  .container {
   width: 300px;
   padding: 20px;
   background-color: lightblue;
  }
  /* Default content-box behavior */
  .content-box {
   width: 100%;
   background-color: lightcoral;
   border: 5px solid red;
  }
  /* Border-box behavior */
  .border-box {
   width: 100%;
   background-color: lightgreen;
   border: 5px solid green;
  }
 </style>
</head>
<body>
 <h2>Content-Box</h2>
 <div class="container">
  <div class="content-box">
   <p>This is a content-box div.</p>
  </div>
 </div>
 <h2>Border-Box</h2>
 <div class="container">
  <div class="border-box">
   <p>This is a border-box div.</p>
  </div>
 </div>
</body></html>
```

CSS border-radius property

Property	Description
border-top-left-radius	It is used to set the border-radius for the top-left corner
border-top-right-radius	It is used to set the border-radius for the top-right corner
border-bottom-right-radius	It is used to set the border-radius for the bottom-right corner
border-bottom-left-radius	It is used to set the border-radius for the bottom-left corner

syntax:

- border-radius: value1 value2 value3 value4 / value5 value6 value7 value8;
  - value1: Defines the radius for the top-left corner.
  - value2: Defines the radius for the top-right corner.
  - value3: Defines the radius for the bottom-right corner.
  - value4: Defines the radius for the bottom-left corner.
- border-radius: 10px 20px 30px 40px;

➢ You can also use the / (slash) notation to specify different horizontal and vertical radii for each corner.

Example:

- border-radius: 10px 20px 30px 40px / 5px 15px 25px 35px;

Example:

```
/* Apply rounded corners to a div */
div {
  width: 100px;
  height: 100px;
  background-color: lightblue;
  border: 2px solid blue;
  border-radius: 20px; /* Uniformly rounded corners */
}
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<title> CSS border-radius </title>
<style>
div {
padding: 50px;
margin: 20px;
border: 6px ridge red;
width: 300px;
float: left;
height: 150px;
}
p{
font-size: 30px;
}
#one {
border-radius: 100px;
```

```
background: rgb(3, 170, 3);
}
#two {
border-radius: 25% 10%;
background: rgb(88, 3, 236);
}
#three {
border-radius: 35px 10em 10%;
background: yellow;
}
#four {
border-radius: 50px 50% 50cm 50em;
background: aqua;
}
</style>
</head>
<body>
<div id = "one">
<h2> Welcome to T3 skills center </h2>
<p> border-radius: 90px; </p>
</div>
<div id = "two">
<h2> Welcome to T3 skills center </h2>
<p> border-radius: 25% 10%; </p>
</div>
<div id = "three">
<h2> Welcome to T3 skills center </h2>
<p> border-radius: 35px 10em 10%; </p>
</div>
<div id = "four">
<h2>Welcome to T3 skills center</h2>
<p>border-radius: 50px 50% 50cm 50em;</p>
</div>
</body>
</html>
```

Output:

Text formatting:

1. Font Properties:

- font-family: Specifies the font family for text.
  - Syntax: `font-family: font-name, fallback-font;`
- font-size`: Sets the font size.
  - Syntax: font-size: value; (e.g., `font-size: 16px;`)
- font-weight: Defines the font weight (e.g., normal, bold).
  - Syntax: `font-weight: value;` (e.g., `font-weight: bold;`)
- font-style: Specifies the font style (e.g., normal, italic).
  - Syntax: `font-style: value;` (e.g., `font-style: italic;`)
- font-variant: Controls the use of small capitals in the font.
  - Syntax: `font-variant: value;` (e.g., `font-variant: small-caps;`)

2. Text Color:

- color: Sets the text color.
  - Syntax: `color: color-value;` (e.g., `color: #FF0000;`)

3. Text Alignment:

- text-align`: Aligns text horizontally (e.g., left, center, right).
  - Syntax: `text-align: value;` (e.g., `text-align: center;`)

4. Text Decoration:

- text-decoration: Adds or removes decorations like underlines and overlines.
  - Syntax: `text-decoration: value;` (e.g., `text-decoration: underline;`)
- text-decoration-line: Specifies type of text decoration (e.g., underline, overline, line-through).
  - Syntax: `text-decoration-line: value;` (e.g., `text-decoration-line: underline;`)
- text-decoration-color: Sets the color of the text decoration.
  - Syntax: `text-decoration-color: color-value;` (e.g., `text-decoration-color: blue;`)
- text-decoration-style: Defines the style of the text decoration (e.g., solid, dotted).
  - Syntax: `text-decoration-style: value;` (e.g., `text-decoration-style: dashed;`)

5. Text Transform:

- text-transform: Changes the capitalization of text (e.g., uppercase, lowercase, capitalize).
  - Syntax: `text-transform: value;` (e.g., `text-transform: uppercase;`)

6. Line Height:

- line-height: Sets the height of a line of text.
  - Syntax: `line-height: value;` (e.g., `line-height: 1.5;`)

7. Letter Spacing:

- letter-spacing`: Adjusts the space between characters.
  - Syntax: `letter-spacing: value;` (e.g., `letter-spacing: 2px;`)

8. Word Spacing:

- word-spacing: Adjusts the space between words.
  - Syntax: `word-spacing: value;` (e.g., `word-spacing: 4px;`)

9. Text Indentation:

- text-indent: Sets the indentation of the first line of text.
  - Syntax: `text-indent: value;` (e.g., `text-indent: 20px;`)

10. Text Shadow:

- text-shadow`: Adds a shadow effect to text.

➢ Syntax: `text-shadow: h-shadow v-shadow blur-radius color;`

- Example: `text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);`

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    /* Font properties */
    p {
      font-family: Arial, sans-serif;
      font-size: 18px;
      font-weight: bold;
      font-style: italic;
      font-variant: small-caps;
    }
    /* Text color */
    .red-text {
      color: red;
    }
    /* Text alignment */
    .center-align {
      text-align: center;
    }
    /* Text decoration */
    .underline-text {
      text-decoration: underline;
    }
    /* Line height */
    .custom-line-height {
      line-height: 1.5;
    }
    /* Letter spacing */
    .letter-spacing-example {
      letter-spacing: 2px;
    }
    /* Word spacing */
    .word-spacing-example {
      word-spacing: 4px;
    }
    /* Text indentation */
    .text-indent-example {
      text-indent: 20px;
    }
    /* Text shadow */
    .text-shadow-example {
```

```
        text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
      }
    </style>
</head>
<body>
    <p>This is a paragraph demonstrating various text formatting properties.</p>
    <p class="red-text center-align underline-text custom-line-height">
      This paragraph has red color, center alignment, underline, and custom line height.
    </p>
    <p class="letter-spacing-example word-spacing-example text-indent-example text-shadow-example">
      This paragraph has customized letter spacing, word spacing, text indentation, and text shadow.
    </p>
</body>
</html>
```

Position Properties:

- CSS positioning properties are used to control the placement and positioning of HTML elements within a web page.

1. position` Property:

- The `position` property specifies the positioning method for an element.
- Values: `static`, `relative`, `absolute`, `fixed`, `sticky`.
- Default value: `static` (elements are positioned according to the normal flow of the document).

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .relative-box {
      position: relative;
      left: 50px;
      top: 20px;
    }
    .absolute-box {
      position: absolute;
      left: 100px;
      top: 50px;
    }
  </style>
</head>
<body>
  <div class="relative-box">
    <p>This is a relative positioned box.</p>
  </div>
  <div class="absolute-box">
    <p>This is an absolute positioned box.</p>
  </div>
</body>
</html>
```

- In this example, we have two boxes. The first one is `relative-box`, which is positioned relative to its normal position, and the second one is `absolute-box`, which is positioned absolutely with respect to the nearest positioned ancestor.

2. top`, `right`, `bottom`, and `left` Properties:

- These properties specify the offset of an absolutely or relatively positioned element from its normal position.
- Values: Length units (e.g., `px`, `em`), percentages.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
```

```
    .relative-box {
      position: relative;
      left: 50px;
      top: 20px;
    }
    .absolute-box {
      position: absolute;
      right: 20%;
      bottom: 30px;
    }
  </style>
</head>
<body>
  <div class="relative-box">
    <p>This is a relative positioned box.</p>
  </div>
  <div class="absolute-box">
    <p>This is an absolute positioned box.</p>
  </div>
</body>
</html>
```

- In this example, the `left`, `top`, `right`, and `bottom` properties are used to position elements relative to their normal positions.

3. z-index` Property:

- The `z-index` property controls the stacking order of positioned elements.
- Elements with a higher `z-index` value will appear on top of elements with a lower value.
- Values: Integer values.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .box1 {
      position: absolute;
      left: 50px;
      top: 20px;
      z-index: 2;
      background-color: lightblue;
    }
    .box2 {
      position: absolute;
      left: 70px;
      top: 40px;
      z-index: 1;
      background-color: lightgreen;
    }
  </style>
```

```
</head>
<body>
  <div class="box1">
    <p>Box 1</p>
  </div>
  <div class="box2">
    <p>Box 2</p>
  </div>
</body></html>
```

- In this example, `box1` has a higher `z-index` value than `box2`, so it appears on top of `box2`.

4. float` Property:

- The `float` property is used to align elements horizontally within their containing elements.
- Values: `left`, `right`, `none`.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .float-left {
      float: left;
      width: 50%;
    }
    .float-right {
      float: right;
      width: 50%;
    }
  </style>
</head>
<body>
  <div class="float-left">
    <p>This is a left-floated element.</p>
  </div>
  <div class="float-right">
    <p>This is a right-floated element.</p>
  </div>
</body>
</html>
```

- In this example, the `float` property is used to float elements to the left and right within their containing element.
- There are five possible values for the `position` property: `static`, `relative`, `absolute`, `fixed`, and `sticky`.

1. static:

- The default value.
- Elements with `position: static;` are positioned according to the normal flow of the document.
- `top`, `right`, `bottom`, and `left` properties have no effect on elements with `position: static;`.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .static-box {
      position: static;
      background-color: lightblue;
    }
  </style>
</head>
<body>
  <div class="static-box">
    <p>This is a static positioned box.</p>
  </div>
</body>
</html>
```

- In this example, the `static-box` is positioned according to the normal flow of the document.

2. relative:

- Elements with `position: relative;` are positioned relative to their normal position in the document flow.
- You can use `top`, `right`, `bottom`, and `left` properties to offset the element from its normal position.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .relative-box {
      position: relative;
      left: 50px;
      top: 20px;
      background-color: lightgreen;
    }
  </style>
</head>
<body>
  <div class="relative-box">
    <p>This is a relative positioned box.</p>
  </div>
</body>
</html>
```

- In this example, the `relative-box` is positioned 50 pixels to the right and 20 pixels down from its normal position.

3. absolute`:

- Elements with `position: absolute;` are positioned relative to their nearest positioned ancestor or the initial containing block if none exists.
- You can use `top`, `right`, `bottom`, and `left` properties to position the element relative to its containing block.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .container {
      position: relative;
    }
    .absolute-box {
      position: absolute;
      left: 100px;
      top: 50px;
      background-color: lightcoral;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="absolute-box">
      <p>This is an absolute positioned box.</p>
    </div>
  </div>
</body>
</html>
```

- In this example, the `absolute-box` is positioned 100 pixels to the right and 50 pixels down from its containing block, which is the `.container` element.

4. fixed:

- Elements with `position: fixed;` are positioned relative to the viewport (browser window).
- They do not move when the page is scrolled.
- Use `top`, `right`, `bottom`, and `left` properties to specify the exact position.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .fixed-box {
      position: fixed;
      right: 20px;
      bottom: 20px;
      background-color: lightskyblue;
    }
  </style>
```

```
</head>
<body>
  <div class="fixed-box">
    <p>This is a fixed positioned box.</p>
  </div>
  <p>Lorem ipsum dolor sit amet...</p>
</body>
</html>
```

- In this example, the `fixed-box` is positioned 20 pixels from the right and 20 pixels from the bottom of the viewport and remains fixed even when you scroll the page.

5. sticky:

- Elements with `position: sticky;` are positioned based on the user's scroll position.
- They act like `relative` positioning until they reach a specified offset, then they become `fixed`.
- Use `top`, `right`, `bottom`, and `left` properties to define the offset values.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .sticky-box {
      position: sticky;
      top: 10px;
      background-color: lightyellow;
    }
  </style>
</head>
<body>
  <div style="height: 200px;"></div>
  <div class="sticky-box">
    <p>This is a sticky positioned box.</p>
  </div>
  <div style="height: 1000px;"></div>
</body>
</html>
```

- In this example, the `sticky-box` is initially positioned 10 pixels from the top of the viewport. As you scroll down, it sticks to the top of the viewport and remains there until you

Display properties:

- display property is used to control how an HTML element is rendered in terms of its layout and visibility.
- It determines the box model used for an element and how it interacts with other elements in the document flow.

❖ display properties name:

- block
- inline
- inline-block
- none
- table
- table-row
- table-cell
- flex
- grid
- inline-flex
- inline-grid
- list-item
- list-item-group
- initial
- inherit

➢ In CSS, the `display` property is used to control how an HTML element is rendered in terms of its layout and visibility. There are various values for the `display` property, each with its own behavior. Here are the common `display` property values:

1. block:

- Elements with `display: block;` generate a block-level box, meaning they start on a new line and take up the full available width by default.
- Common block-level elements include `<div>`, `<p>`, `<h1>`, and `<ul>`.

Syntax: display: block;

2. inline:

- Elements with `display: inline;` generate an inline-level box, meaning they do not start on a new line and only take up as much width as necessary.
- Common inline-level elements include `<span>`, `<a>`, and `<strong>`.

Syntax: display: inline;

3. inline-block:

- Elements with `display: inline-block;` combine the characteristics of both `block` and `inline` elements. They do not start on a new line and take up only as much width as necessary, but you can apply block-level styling to them.
- Useful for creating inline elements with block-level styling.

Syntax: display: inline-block;

4. none:

- Elements with `display: none;` are completely hidden and do not occupy space on the page.
- Useful for hiding elements dynamically using JavaScript or for accessibility purposes.

Syntax: display: none;

5. table:

- Elements with `display: table;` generate a table-level box.
- Useful for creating table layouts.

Syntax: display: table;

6. table-row:

- Elements with `display: table-row;` generate a table row-level box within a table.
- Used to create table rows within a table layout.

Syntax: display: table-row;

7. table-cell:

- Elements with `display: table-cell;` generate a table cell-level box within a table row.
- Used to create table cells within a table layout.

Syntax: display: table-cell;

8. flex:

- Elements with `display: flex;` create a flex container.
- Child elements of the flex container become flex items, and they can be laid out in a flexible row or column format using flexbox properties.

Syntax: display: flex;

9. grid:

- Elements with `display: grid;` create a grid container.
- Child elements of the grid container become grid items, and they can be positioned in rows and columns using grid properties.

Syntax: display: grid;

10. inline-flex and inline-grid:

- These values are similar to `flex` and `grid`, respectively, but they behave like inline-level elements.

Syntax: display: inline-flex;

Syntax: display: inline-grid;

11. list-item:

- Elements with `display: list-item;` generate a list item-level box.
- Used for creating list items within a list.

Syntax: display: list-item;

Syntax: display: list-item-group;

12. initial` and `inherit:

- These values allow elements to inherit the `display` property from their parent or reset it to its default value.

Syntax: display: initial;

Syntax: display: inherit;

Example:1. inline:

- Elements with `display: inline;` do not start on a new line and only take up as much width as necessary.

```
<!DOCTYPE html>
<html>
<head>
  <style>
```

```
      .inline-box {
        display: inline;
        background-color: lightblue;
      }
    </style>
  </head>
  <body>
    <p>This is some <span class="inline-box">inline</span> text.</p>
  </body>
  </html>
```

2. inline-block:

- Elements with `display: inline-block;` do not start on a new line but can take up width as necessary while behaving like a block-level element.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .inline-block-box {
      display: inline-block;
      width: 100px;
      height: 100px;
      background-color: lightgreen;
    }
  </style>
</head>
<body>
  <p>This is some text with an <div class="inline-block-box"></div> inline-block element.</p>
</body>
</html>
```

3. none:

- Elements with `display: none;` are completely hidden and do not occupy space on the page.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .hidden-box {
      display: none;
    }
  </style>
</head>
<body>
  <p>This is some visible text.</p>
  <p class="hidden-box">This text is hidden.</p>
</body>
</html>
```

4. table:

- Elements with `display: table;` generate a table-level box.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .table-container {
      display: table;
      width: 300px;
      background-color: lightcoral;
    }
    .table-cell {
      display: table-cell;
      vertical-align: middle;
      text-align: center;
      height: 150px;
      background-color: lightyellow;
    }
  </style>
</head>
<body>
  <div class="table-container">
    <div class="table-cell">Table Cell Content</div>
  </div>
</body>
</html>
```

5. table-row:

- Elements with `display: table-row;` generate a table row-level box within a table.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .table {
      display: table;
      width: 300px;
      background-color: lightblue;
    }

    .table-row {
      display: table-row;
      background-color: lightgreen;
    }
  </style>
</head>
<body>
```

```
    <div class="table">
      <div class="table-row">Row 1</div>
      <div class="table-row">Row 2</div>
      <div class="table-row">Row 3</div>
    </div>
  </body>
  </html>
```

6. flex:

- Elements with `display: flex;` create a flex container. Child elements of the flex container become flex items, and they can be laid out in a flexible row or column format using flexbox properties.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .flex-container {
      display: flex;
      justify-content: space-between;
      background-color: lightblue;
    }
    .flex-item {
      width: 100px;
      height: 100px;
      background-color: lightgreen;
    }
  </style>
</head>
<body>
  <div class="flex-container">
    <div class="flex-item">1</div>
    <div class="flex-item">2</div>
    <div class="flex-item">3</div>
  </div>
</body>
</html>
```

7. grid:

- Elements with `display: grid;` create a grid container. Child elements of the grid container become grid items, and they can be positioned in rows and columns using grid properties.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .grid-container {
      display: grid;
      grid-template-columns: repeat(3, 1fr);
```

```
      gap: 10px;
      background-color: lightcoral;
    }
    .grid-item {
      width: 100px;
      height: 100px;
      background-color: lightyellow;
    }
  </style>
</head>
<body>
  <div class="grid-container">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
  </div>
</body>
</html>
```

8. inline-flex:

- Elements with `display: inline-flex;` behave like `flex` containers but are treated as inline-level elements.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .inline-flex-container {
      display: inline-flex;
      background-color: lightblue;
    }

    .inline-flex-item {
      width: 100px;
      height: 100px;
      background-color: lightgreen;
    }
  </style>
</head>
<body>
  <p>This is some text with an <div class="inline-flex-container"><div class="inline-flex-item">inline-flex element</div></div> inside.</p>
</body>
</html>
```

9. inline-grid:

- Elements with `display: inline-grid;` behave like `grid` containers but are treated as inline-level elements.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .inline-grid-container {
      display: inline-grid;
      grid-template-columns: repeat(3, 1fr);
      gap: 10px;
      background-color: lightcoral;
    }

    .inline-grid-item {
      width: 100px;
      height: 100px;
      background-color: lightyellow;
    }
  </style>
</head>
<body>
  <p>This is some text with an <div class="inline-grid-container"><div class="inline-grid-item">inline-grid element</div></div> inside.</p>
</body>
</html>
```

10. list-item:

- Elements with `display: list-item;` generate a list item-level box. This is typically used for creating list items within a list.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    ul {
      display: list-item-group; /* Specifies the list container */
    }

    li {
      display: list-item; /* Specifies list items */
      list-style-type: square; /* Custom bullet style */
    }
  </style>
</head>
<body>
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
```

```
    </ul>
  </body>
  </html>
```

11. list-item-group:

- Elements with `display: list-item-group;` create a list item group-level box. This is used for grouping and styling related list items.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .list-group {
      display: list-item-group;
    }

    .list-item {
      display: list-item;
      list-style-type: square;
    }
  </style>
</head>
<body>
  <div class="list-group">
    <div class="list-item">Item 1</div>
    <div class="list-item">Item 2</div>
    <div class="list-item">Item 3</div>
  </div>
</body>
</html>
```

12. initial:

- Elements with `display: initial;` reset the `display` property to its default value, which depends on the element type. This allows the element to inherit the default behavior for its type.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .initial-box {
      display: initial;
    }
  </style>
</head>
<body>
  <p>This is some visible text.</p>
  <div class="initial-box">This element resets its display property to default.</div>
</body>
</html>
```

4. inherit:

- Elements with `display: inherit;` inherit the `display` property value from their parent element. This allows an element to have the same display behavior as its parent.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .container {
      display: flex;
    }
    .child {
      display: inherit;
      width: 100px;
      height: 100px;
      background-color: lightblue;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="child">Inherited Display</div>
  </div>
</body>
</html>
```

To change the style of hyperlinks:

- you can use CSS to define different styles for four different states of links:

1. **Link (Unvisited):** This is the default style for a hyperlink when it has not been clicked or visited. You can change properties like text color, text decoration, and background color.
2. **Visited:** These styles are applied to links that have been clicked or visited. It's essential to differentiate visited links from unvisited ones.
3. **Hover:** These styles are applied when the mouse pointer is over the link. Hover styles can provide visual feedback to users.
4. **Active:** These styles are applied when the link is clicked or activated. Active styles give immediate feedback to users that they've interacted with the link.

1.Link (Unvisited):

```
/* Normal state - Unvisited link */
a {
  text-decoration: none; /* Remove the underline */
  color: #333; /* Link color */
}
```

2.Visited:

```
/* Visited link */
a:visited {
  color: #666; /* Visited link color */
}
```

3.Hover:

```
/* Hover state */
a:hover {
  text-decoration: underline; /* Add underline on hover */
  color: #FF5733; /* Link color on hover */
}
```

4.Active:

```
/* Active state - Link when clicked */
a:active {
  color: #FF0000; /* Link color when active (clicked) */
}
```

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    /* Normal state - Unvisited link */
    a {
      text-decoration: none; /* Remove the underline */
      color: #333; /* Link color */
    }
    /* Visited link */
    a:visited {
      color: #666; /* Visited link color */
```

```
      }
      /* Hover state */
      a:hover {
        text-decoration: underline; /* Add underline on hover */
        color: #FF5733; /* Link color on hover */
      }
      /* Active state - Link when clicked */
      a:active {
        color: #FF0000; /* Link color when active (clicked) */
      }
    </style>
</head>
<body>
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">About</a></li>
      <li><a href="#">Services</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
</body>
</html>
```

Apply the style for image:

- To apply styles to images using CSS, you can use CSS selectors to target the `<img>` element and set various properties like width, height, border, margins, and more

1. Create an HTML Structure:

- Start by creating an HTML structure with one or more `<img>` elements. Here's a basic

example:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="raj.css">
</head>
<body>
  <h1>Image Styling Example</h1>
  <img src="example.jpg" alt="Example Image">
</body>
</html>
```

- In this example, we have an `<img>` element displaying an image named `example.jpg`.

2. Create a CSS File:

- Create a separate CSS file (e.g., `styles.css`) where you'll define the styles for the `<img>` element. Link this CSS file to your HTML document using the `<link>` tag in the `<head>` section of your HTML file.

3. Define CSS Styles:

- Open your CSS file (`styles.css`) and define the styles for the `<img>` element. Here's an example of how to apply styles to images:

```
styles.css
Apply a maximum width to the image
img {
  max-width: 100%; /* Image won't exceed its parent's width */
  height: auto; /* Maintain aspect ratio */
  border: 2px solid #333; /* Add a border */
  margin: 10px; /* Add some margin around the image */
  padding: 5px; /* Add padding inside the border */
  display: block; /* Remove extra space under inline images */
}
```

- max-width` ensures that the image won't exceed its parent container's width while maintaining its aspect ratio.
  - `height: auto;` maintains the aspect ratio of the image.
  - `border` adds a border around the image.
  - `margin` and `padding` create spacing around the image.
  - `display: block;` removes any extra space below inline images.

4. Link CSS to HTML:

Make sure your HTML file (`index.html`) links to the CSS file you created in the `<head>` section:

```
<link rel="stylesheet" type="text/css" href="raj.css">
```

Apply style for table:

- To apply styles to an HTML table using CSS, you can use CSS selectors to target the table,

1. Create an HTML Table:

- Start by creating an HTML table structure. Here's a basic example:

Example:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <h1>Styled Table Example</h1>
  <table>
    <thead>
      <tr>
        <th>Name</th>
        <th>Age</th>
        <th>Country</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>John</td>
        <td>25</td>
        <td>USA</td>
      </tr>
      <tr>
        <td>Mary</td>
        <td>30</td>
        <td>Canada</td>
      </tr>
      <tr>
        <td>Lisa</td>
        <td>22</td>
        <td>UK</td>
      </tr>
    </tbody>
  </table>
</body>
</html>
```

- In this example, we have an HTML table with header cells (`<th>`) and data cells (`<td>`).

2. Create a CSS File:

- Create a separate CSS file (e.g., `styles.css`) where you'll define the styles for the table. Link this CSS file to your HTML document using the `<link>` tag in the `<head>` section of your HTML file.

3. Define CSS Styles:

- Open your CSS file (`styles.css`) and define the styles for the table and its elements.

Example of how to style a table:

styles.css

```
table {
  width: 100%;
  border-collapse: collapse; /* Merge table borders */
}
/* Style table headers */
th {
  background-color: #333;
  color: #fff;
  padding: 10px;
}
/* Style table data cells */
td {
  border: 1px solid #ccc;
  padding: 8px;
}
/* Style alternate rows for better readability */
tr:nth-child(even) {
  background-color: #f2f2f2;
}
```

- In this CSS code, we're applying the following styles:
  - `width` ensures the table spans the entire available width.
  - `border-collapse: collapse;` merges borders between table cells.
  - Styles for `<th>` and `<td>` elements control background colors, text colors, and padding.
  - `tr:nth-child(even)` applies a background color to alternate rows for better readability.

4. Link CSS to HTML:

- Make sure your HTML file (`index.html`) links to the CSS file you created in the `<head>` section:
- <link rel="stylesheet" type="text/css" href="styles.css">

How to use image in text field background

- You can use an image as a background for a text field (or input field) in HTML by applying CSS styles.

Example:

1. Create an HTML Form:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <h1>Text Field with Background Image</h1>
  <form>
    <label for="username">Username:</label>
    <input type="text" id="username" name="username">
  </form>
</body>
</html>
```

2. Create a CSS File:

- Create a separate CSS file (e.g., `styles.css`) where you'll define the styles for the text field and its background image. Link this CSS file to your HTML document using the `<link>` tag in the `<head>` section of your HTML file.

3. Define CSS Styles:

- Open your CSS file (`styles.css`) and define the styles for the text field, including the background image. Here's an example of how to set a background image for the text field:

➢ **styles.css**

```
input[type="text"] {
  padding: 10px; /* Add padding for text input */
  border: 1px solid #ccc; /* Add a border for the text input */
  background-image: url('background-image.jpg'); /* Set the background image */
  background-size: cover; /* Cover the entire input */
  background-repeat: no-repeat; /* Prevent image repetition */
}
```

In this CSS code, we're applying the following styles:

- `padding` adds some padding around the text input for spacing.
- `border` adds a border around the text input.
- `background-image` specifies the URL of the background image.
- `background-size` ensures that the image covers the entire input.
- `background-repeat` prevents the image from repeating.

4. Link CSS to HTML:

Make sure your HTML file (`index.html`) links to the CSS file you created in the `<head>` section:

```
<link rel="stylesheet" type="text/css" href="styles.css">
```

Animation

- CSS animations allow you to create dynamic and visually engaging effects on your web pages.
- You can animate various CSS properties like size, position, color, and more.

Example:

- CSS Animation Basics:
- CSS animations are created using the `@keyframes` rule and the `animation` property. Keyframes define how the animation progresses over time, and the `animation` property specifies the animation's duration, timing function, and other parameters.

❖ **basic structure:**

```
@keyframes animationName {
  0% {
    /* CSS properties at the start of the animation */
  }
  100% {
    /* CSS properties at the end of the animation */
  }
}
/* Apply the animation to an element */
.element {
  animation: animationName duration timing-function delay iteration-count direction;
}
```

- `animationName`: A name for your animation.
- `duration`: The time it takes for one iteration of the animation (e.g., `2s` for 2 seconds).
- `timing-function`: Specifies the pacing of the animation (e.g., `linear`, `ease`, `ease-in`, `ease-out`, `ease-in-out`, etc.).
- `delay`: The time before the animation starts (e.g., `1s` for 1 second).
- `iteration-count`: The number of times the animation should repeat (e.g., `infinite` for infinite looping).
- `direction`: The direction of the animation (`normal`, `reverse`, `alternate`, `alternate-reverse.

Example - Bouncing Ball Animation:

➢ Let's create a simple example of a bouncing ball animation:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <div class="ball"></div>
</body>
</html>
/* styles.css */
.ball {
  width: 50px;
  height: 50px;
  background-color: #3498db;
  border-radius: 50%;
```

```
    position: absolute;
    animation: bounce 2s ease-in-out infinite;
  }
  @keyframes bounce {
    0%, 100% {
      transform: translateY(0);
    }
    50% {
      transform: translateY(-100px);
    }
  }
```

- In this example:
  - We create a `<div>` element with the class `.ball`, which will represent our bouncing ball.
  - We use CSS to define the initial size, color, and shape of the ball.
  - We set the `position` property to `absolute` to control its position.
  - We apply the `bounce` animation to the ball, specifying that it should bounce for 2 seconds with an ease-in-out timing function and repeat infinitely.

How to add map to your website:

- To use a map on your website, you can embed an interactive map from a mapping service like Google Maps or use a JavaScript library like Leaflet to create custom maps. how to do both:

Option 1: Embedding Google Maps

1. Go to Google Maps:

- Visit the [Google Maps website](https://www.google.com/maps) and search for the location or area you want to display on your website.

2. Create or Select a Location:

- You can search for a location or click on the map to select a specific place.

3. Open the Menu:

- In the left sidebar, click on the menu icon (three horizontal lines) to open the menu.

4. Choose "Share or Embed Map":

- From the menu, select "Share or embed map."

5. Embed Map Options:

- You'll see options to embed the map. You can choose the size of the map, whether to include a marker, and more.

6. Copy the HTML Code:

- Copy the HTML code provided in the "Embed a map" section.

7. Paste the Code in Your Website:

- In your website's HTML file, paste the code where you want the map to appear.

example:

```
<!DOCTYPE html>
<html>
<head>
  <!-- Your other head content -->
</head>
<body>
  <!-- Your other webpage content -->
  <!-- Paste the Google Maps embed code here -->
  <iframe src="https://www.google.com/maps/embed?params"></iframe>
</body>
</html>
```

Option 2: Using the Leaflet JavaScript Library**

1. Include Leaflet Library:

- Download the Leaflet library or include it directly from a content delivery network (CDN) in your HTML file. Add this in the `<head>` section of your HTML file:
- <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />
- <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>

2. Create a Container Element:

In your HTML, create an element (e.g., a `<div>`) to serve as the container for your map:

```
<div id="map" style="height: 400px;"></div>
```

3. Initialize the Map with JavaScript:

- In a `<script>` tag at the bottom of your HTML file, initialize the map using JavaScript:

```
  <script>
    var map = L.map('map').setView([51.505, -0.09], 13);
    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
      attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
    }).addTo(map);
    L.marker([51.5, -0.09]).addTo(map)
      .bindPopup('A marker here!')
      .openPopup();
  </script>
```

- In this script, we create a map, set its center and zoom level, add a tile layer (map tiles), and add a marker with a popup.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Interactive Map Example</title>
  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />
  <style>
    /* Style the map container */
    #map {
      height: 400px;
    }
  </style>
</head>
<body>
  <h1>Interactive Map Example</h1>
  <!-- Create a container for the map -->
  <div id="map"></div>
  <!-- Include Leaflet library -->
  <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
  <script>
    // Initialize the map
    var map = L.map('map').setView([51.505, -0.09], 13);
    // Add a tile layer (in this case, using OpenStreetMap tiles)
    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
      attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
    }).addTo(map);
    // Add a marker with a popup
    L.marker([51.5, -0.09]).addTo(map)
      .bindPopup('A marker here!')
      .openPopup();
  </script>
</body>
</html>
```

How to apply video in background:

- To apply a video as a background to a webpage using HTML and CSS, follow these steps:

1. Prepare Your Video:

- First, you need to have a video file in a compatible format (e.g., MP4, WebM) that you want to use as the background. Ensure the video content is relevant and suits the design of your website.

2. Create an HTML Structure:

- Create the HTML structure for your webpage. Here's a basic example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Video Background Example</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="video-container">
    <video autoplay muted loop id="video-bg">
      <source src="your-video.mp4" type="video/mp4">
      Your browser does not support the video tag.
    </video>
    <div class="content">
      <!-- Your website content goes here -->
      <h1>Welcome to My Website</h1>
      <p>This is a video background example.</p>
    </div>
  </div>
</body>
</html>
```

In this example:

- We create an HTML structure with a `<div>` container (`video-container`) that holds the video and website content.
- Inside the container, we embed a `<video>` element with the video file (`your-video.mp4`) and fallback content for unsupported browsers.
- The video is set to autoplay, muted, and loop for a continuous background effect.
- The `<div class="content">` is where you can place your website content.

3. Create CSS Styles:

- Create a CSS file (e.g., `styles.css`) to apply styles and position the video and content. Here's an example of CSS:

```
 styles.css
body, html {
  margin: 0;
  padding: 0;
  height: 100%;
}
.video-container {
```

```
    position: relative;
    height: 100vh; /* 100% viewport height */
    overflow: hidden;
}
#video-bg {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    object-fit: cover; /* Maintain aspect ratio and cover the entire container */
}
.content {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    text-align: center;
    color: #fff;
}
```

- In this CSS code:
  - We set the `body` and `html` to have no margin or padding and to take up 100% of the viewport height.
    - The `.video-container` is given a height of 100vh to fill the entire viewport.
    - The video (`#video-bg`) is set to cover the container using `object-fit: cover`.
    - The `.content` is centered both vertically and horizontally on the video.

4. Add Fallback Content:

- Inside the `<video>` element, provide fallback content (e.g., "Your browser does not support the video tag.") that will be displayed if the browser does not support video playback.

How to add google font family:

- To add Google Fonts to your website,
- follow these step-by-step instructions using HTML and CSS:

1. Visit the Google Fonts Website:

- Go to the [Google Fonts website](https://fonts.google.com/) to browse and select the fonts you want to use.

2.Select Fonts:

- Browse the fonts and click the "Select this font" button for the fonts you want to use. You can select one or multiple fonts.

3.Review Selection:

- Click the "Review" button at the bottom to review your font selection.

4. Get the Code:

- In the "Embed Font" section, you will see two tabs: "Embed" and "Standard." Choose the "Standard" tab. You will see a code snippet to include in your HTML file. It looks something like this:
- <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=FontName">
- Replace "FontName" with the name of the font or fonts you selected.

5. Add the HTML Code:

- In your HTML file's `<head>` section, paste the code snippet you obtained from Google Fonts.

For example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Google Fonts Example</title>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=FontName">
</head>
<body>
  <!-- Your website content goes here -->
  <h1>Welcome to My Website</h1>
  <p>This is an example of Google Fonts.</p>
</body></html>
```

6. Apply the Font in CSS:

- In your CSS file (or within a `<style>` block in your HTML file), you can specify the font family you want to use for specific elements. For example:

Example:

```
body {
  font-family: 'FontName', sans-serif;}
h1 {
  font-family: 'FontName', sans-serif;
}
p {
  font-family: 'FontName', sans-serif;
```

} Replace `'FontName'` with the name of the Google Font you selected.

Transformation and Transition:

❖ CSS Transformations:

- **Definition:** Transformations in CSS refer to the ability to change the shape, size, position, and orientation of HTML elements in 2D or 3D space.
- Common Transformations: CSS provides several transformation functions, including translate(), rotate(), scale(), skew(), and matrix(), which can be used to modify elements in various ways.
- Usage: Transformations are typically applied using the transform property. For example, you can rotate an element by transform: rotate(45deg); or scale it with transform: scale(1.2);.
- **Example:** You can use transformations to create effects like rotating an image when hovered or scaling up a button on click.

❖ CSS Transitions:

- Definition: Transitions in CSS allow you to smoothly animate changes in element properties (e.g., color, size, position) over a specified duration and with a specified timing function.
- Common Transition Properties: CSS transitions are applied to specific properties using the transition property. You can specify which properties to transition, the duration of the transition, the timing function (easing), and a delay before the transition starts.
- Usage: Transitions are often used for interactive effects. For example, you can create a button that changes color gradually when hovered over, giving the user visual feedback.
- Example: To apply a transition to the background-color property with a 0.3-second duration and an ease-in-out timing function, you can use transition: background-color 0.3s ease-in-out;.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Transformations Example</title>
  <style>
    /* Define a common style for transformation boxes */
    .transform-box {
      width: 100px;
      height: 100px;
      background-color: #3498db;
      margin: 20px;
      display: inline-block;
    }
    /* Apply a translation */
    .translate-box {
      transform: translate(50px, 20px);
    }
    /* Apply a rotation */
    .rotate-box {
      transform: rotate(45deg);
    }
```

```
      /* Apply a scaling */
      .scale-box {
        transform: scale(1.5);
      }
      /* Apply a skew */
      .skew-box {
        transform: skew(30deg, 20deg);
      }
      /* Apply a custom matrix transformation */
      .matrix-box {
        transform: matrix(1, 0.5, -0.5, 1, 0, 0);
      }
    </style>
</head>
<body>
    <h1>CSS Transformations Example</h1>
    <!-- Translate -->
    <div class="transform-box translate-box">Translate</div>
    <!-- Rotate -->
    <div class="transform-box rotate-box">Rotate</div>
    <!-- Scale -->
    <div class="transform-box scale-box">Scale</div>
    <!-- Skew -->
    <div class="transform-box skew-box">Skew</div>
    <!-- Matrix -->
    <div class="transform-box matrix-box">Matrix</div>
</body>
</html>
```

❖ Example of CSS transitions:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Transitions Example</title>
    <style>
      /* Define a common style for transition boxes */
      .transition-box {
        width: 100px;
        height: 100px;
        background-color: #3498db;
        margin: 20px;
        display: inline-block;
        text-align: center;
        line-height: 100px;
        font-size: 18px;
        color: #fff;
```

```
            cursor: pointer;
            transition: background-color 0.3s ease; /* Apply transition to background-color property */
        }
        /* Define hover styles for transition effect */
        .transition-box:hover {
            background-color: #e74c3c;
        }
    </style>
</head>
<body>
    <h1>CSS Transitions Example</h1>
    <!-- Transition Example 1 -->
    <div class="transition-box">Hover Me</div>
    <!-- Transition Example 2 -->
    <div class="transition-box">Hover Me</div>
    <!-- Transition Example 3 -->
    <div class="transition-box">Hover Me</div>
</body>
</html>
```

image transformations:

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Image Transformation Example</title>
   <style>
      /* Define styles for the container */
      .image-container {
         width: 300px;
         margin: 20px auto;
      }
      /* Define styles for the image */
      .image {
         width: 100%;
         max-width: 100%;
         height: auto;
         transition: transform 0.3s ease; /* Apply transition to the transform property */
      }
      /* Add hover effect */
      .image:hover {
         transform: scale(1.2); /* Scale the image on hover */
      }
   </style>
</head>
<body>
   <h1>Image Transformation Example</h1>
   <div class="image-container">
      <img class="image" src="your-image.jpg" alt="Sample Image">
   </div>
</body>
</html>
```

min width

- The `min-width` property in CSS allows you to set the minimum width of an element.
- This ensures that the element won't become narrower than the specified minimum width, even if its content is smaller.

example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>min-width Example</title>
  <style>
    /* Apply a min-width to a container */
    .container {
      min-width: 300px; /* Set the minimum width to 300 pixels */
      background-color: #f2f2f2;
      padding: 20px;
    }
  </style>
</head>
<body>
  <h1>min-width Example</h1>
  <!-- Create a container div with a minimum width -->
  <div class="container">
    <p>This container has a minimum width of 300 pixels.</p>
  </div>
</body>
</html>
```

In this example:
- We have a container `<div>` with a class of `.container`.
- The `.container` has a `min-width` set to `300px`, ensuring that it will always be at least 300 pixels wide.
- The background color and padding are applied for better visualization.

max width

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>max-width Example</title>
  <style>
    /* Apply a max-width to a container */
    .container {
      max-width: 600px; /* Set the maximum width to 600 pixels */
```

```
        background-color: #f2f2f2;
        padding: 20px;
        margin: 0 auto; /* Center the container horizontally */
      }
    </style>
</head>
<body>
    <h1>max-width Example</h1>
    <!-- Create a container div with a maximum width -->
    <div class="container">
      <p>This container has a maximum width of 600 pixels.</p>
      <img src="example-image.jpg" alt="Example Image" style="max-width: 100%; height: auto;">
    </div>
</body>
</html>
```

media query:

- Media queries in CSS allow you to apply different styles to elements based on various conditions, such as the width of the viewport or the device type. Here's a step-by-step guide to understanding and using media queries in CSS:

Step 1: Understanding the Basics

- What is a Media Query?: A media query is a CSS technique that lets you apply different styles to elements based on the characteristics of the user's device, such as screen size, resolution, or device orientation.

Step 2: Media Query Syntax

- Syntax: A media query starts with the `@media` keyword followed by a condition enclosed in parentheses. For example:

```
@media (max-width: 600px) {
  /* CSS rules for screens with a maximum width of 600px */
}
```

Step 3: Media Query Conditions

- Conditions: Media queries use various conditions to target specific devices or screen properties. Common conditions include:
  - `min-width` and `max-width` for targeting specific screen widths.
  - `min-height` and `max-height` for targeting specific screen heights.
  - `orientation` for targeting portrait or landscape orientation.
  - `screen` for targeting screens (as opposed to print, speech, etc.).

Step 4: Applying Styles

- Adding Styles: Within the media query block, you can add CSS rules just like you do outside media queries. These rules will only apply when the specified condition is met.

Step 5: Creating Responsive Designs

- Responsive Design: Media queries are commonly used for responsive web design. For example, you can define different layouts for mobile and desktop screens:

```
@media (max-width: 600px) {
  /* Mobile styles */
}
@media (min-width: 601px) {
  /* Desktop styles */
}
```

Step 6: Combining Conditions

- Combining Conditions: You can combine conditions using logical operators like `and` and `or`. For instance, you can target devices with a specific width and height:

```
@media (min-width: 600px) and (max-height: 800px) {
  /* Styles for devices with width >= 600px and height <= 800px */
}
```

Step 7: Testing and Debugging

- Testing: To test your media queries, simply resize your browser window to see how the styles change based on the conditions.

Step 8: Real-World Example**

```
/* Mobile styles */
@media (max-width: 600px) {
```

```
    body {
        font-size: 16px;
    }
    /* Other mobile-specific styles */
}
/* Desktop styles */
@media (min-width: 601px) {
    body {
        font-size: 18px;
    }
    /* Other desktop-specific styles */
}
```

how to create two navigation bars for different devices (mobile and desktop)

- To create two navigation bars for different devices (mobile and desktop), you can use media queries to show/hide navigation bars based on the screen width. Here's an example:

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Responsive Navigation Example</title>
  <style>
    /* Common navigation styles */
    ul.nav {
      list-style: none;
      padding: 0;
    }
    ul.nav li {
      display: inline-block;
      margin-right: 20px;
    }
    /* Mobile navigation - initially hidden */
    ul.nav-mobile {
      display: none;
    }
    /* Desktop navigation - initially visible */
    @media (min-width: 601px) {
      ul.nav-mobile {
        display: none;
      }
      ul.nav-desktop {
        display: block;
      }
    }
    /* Media query for screens with a maximum width of 600px (mobile) */
    @media (max-width: 600px) {
      ul.nav-mobile {
        display: block;
      }
      ul.nav-desktop {
        display: none;
      }
    }
  </style>
</head>
<body>
  <header>
    <!-- Mobile Navigation -->
```

```
    <nav>
      <ul class="nav nav-mobile">
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Services</a></li>
      </ul>
    </nav>
    <!-- Desktop Navigation -->
    <nav>
      <ul class="nav nav-desktop">
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Services</a></li>
        <li><a href="#">Portfolio</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>
  </header>
  <!-- Rest of your webpage content -->
</body>
</html>
```

- **In this example:**
  - We have two navigation bars: one for mobile devices (`ul.nav-mobile`) and one for desktop (`ul.nav-desktop`).
  - Initially, the mobile navigation is hidden (`display: none`) and the desktop navigation is visible (`display: block`).
  - Using media queries, we hide the desktop navigation and show the mobile navigation when the screen width is 600px or less.
  - For screens wider than 600px, the mobile navigation is hidden, and the desktop navigation is displayed.