RESEARCH INNOVATION DISCOVERY

RID संस्था समस्या का समाधान

RUN BY TWKSAA WELFARE FOUNDATION

Reg.No: 048884
Foundation Day
30-09-2023

CSS (Cascading Style Sheets)

E-Book

Er. Rajesh Prasad(B.E, M.E)
Founder: TWF & RID Org.

- **RID ORGANIZATION** यानि **Research, Innovation and Discovery** संस्था जिसका मुख्य उदेश्य हैं आने वाले समय में सबसे पहले **NEW (RID, PMS & TLR)** की खोज, प्रकाशन एवं उपयोग भारत की इस पावन धरती से भारतीय संस्कृति, सभ्यता एवं भाषा में ही हो |
- देश, समाज, एवं लोगों की समस्याओं का समाधान **NEW (RID, PMS & TLR)** के माध्यम से किया जाये इसके लिए ही मैं राजेश प्रसाद **इस RID संस्था** की स्थपना किया हूँ |
- Research, Innovation & Discovery में रूचि रखने वाले आप सभी विधार्थियों, शिक्षकों एवं बुधीजिवियो से मैं आवाहनं करता हूँ की आप सभी **इस RID संस्था** से जुड़ें एवं अपने बुध्दि, विवेक एवं प्रतिभा से दुनियां को कुछ नई **(RID, PMS & TLR)** की खोजकर, बनाकर एवं अपनाकर लोगों की समस्याओं का समाधान करें |

त्वक्सा CSS के इस ई-पुस्तक में आप CSS से जुड़ी सभी बुनियादी अवधारणाएँ सीखेंगे। मुझे आशा है कि इस ई-पुस्तक को पढ़ने के बाद आपके ज्ञान में वृद्धि होगी और आपको कंप्यूटर विज्ञान के बारे में और अधिक जानने में रुचि होगी

"In this E-Book of TWKSAA CSS you will learn all the basic concepts related to CSS. I hope after reading this E- Book your knowledge will be improved and you will get more interest to know more thing about computer Science".

RID हमें क्यों करना चाहिए

(Research)

अनुसंधान हमें क्यों करना चाहिए ?

Why should we do research?

1. नई ज्ञान की प्राप्ति (Acquisition of new knowledge)
2. समस्याओं का समाधान(To Solving problems)
3. सामाजिक प्रगति (To Social progress)
4. विकास को बढ़ावा देने(To promote development)
5. तकनीकी और व्यापार में उन्नति(To advances in technology & business)
6. देश विज्ञान और प्रौद्योगिकी के विकास(To develop the country's science & technology)

(Innovation)

नवीनीकरण हमें क्यों करना चाहिए ?

Why should we do Innovation?

1. प्रगति के लिए(To progress)
2. परिवर्तन के लिए(For change)
3. उत्पादन में सुधार(To Improvement in production)
4. समाज को लाभ(To Benefit to society)
5. प्रतिस्पर्धा में अग्रणी (To be ahead of competition)
6. देश विज्ञान और प्रौद्योगिकी के विकास। (To develop the country's science & technology)

(Discovery)

खोज हमें क्यों करना चाहिए?

Why should we do Discovery?

1. नए ज्ञान की प्राप्ति(Acquisition of new knowledge)
2. अविष्कारों की खोज(To Discovery of inventions)
3. समस्याओं का समाधान(To Solving problems)
4. ज्ञान के विकास में योगदान(Contribution to development of knowledge)
5. समाज के उन्नति के लिए (for progress of society)
6. देश विज्ञान और तकनीक के विकास (To develop the country's science & technology)

New Technology पर **Research** करने के लिए सम्पर्क करें: **ridorg.in@gmail.com** **Mob.No: 9202707903**

❖ Research(अनुसंधान):

- अनुसंधान एक प्रणालीकरण कार्य होता है जिसमें विशेष विषय या विषय की नई ज्ञान एवं समझ को प्राप्त करने के लिए सिद्धांतिक जांच और अध्ययन किया जाता है। इसकी प्रक्रिया में डेटा का संग्रह और विश्लेषण, निष्कर्ष निकालना और विशेष क्षेत्र में मौजूदा ज्ञान में योगदान किया जाता है। अनुसंधान के माध्यम से विज्ञान, प्रोधोगिकी, चिकित्सा, सामाजिक विज्ञान, मानविकी, और अन्य क्षेत्रों में विकास किया जाता है। अनुसंधान की प्रक्रिया में अनुसंधान प्रश्न या कल्पनाएँ तैयार की जाती हैं, एक अनुसंधान योजना डिज़ाइन की जाती है, डेटा का संग्रह किया जाता है, विश्लेषण किया जाता है, निष्कर्ष निकाला जाता है और परिणामों को उचित दर्शाने के लिए समाप्ति तक पहुंचाया जाता है।

❖ Innovation(नवीनीकरण): -

- Innovation एक विशेषता या नई विचारधारा की उत्पत्ति या नवीनीकरण है। यह नए और आधुनिक विचारों, तकनीकों, उत्पादों, प्रक्रियाओं, सेवाओं या संगठनात्मक ढंगों का सृजन करने की प्रक्रिया है जिससे समस्याओं का समाधान, प्रतिस्पर्धा में अग्रणी होने, और उपयोगकर्ताओं के अनुकूलता में सुधार किया जा सकता है।

❖ Discovery (आविष्कार):

- Discovery का अर्थ होता है "खोज" या "आविष्कार"। यह एक विशेषता है जो किसी नए ज्ञान, अविष्कार, या तत्व की खोज करने की प्रक्रिया को संदर्भित करता है। खोज विज्ञान, इतिहास, भूगोल, तकनीक, या किसी अन्य क्षेत्र में हो सकती है। इस प्रक्रिया में, व्यक्ति या समूह नए और अज्ञात ज्ञान को खोजकर समझने का प्रयास करते हैं और इससे मानव सभ्यता और विज्ञान-तकनीकी के विकास में योगदान देते हैं।

नोट : अनुसंधान विशेषता या विषय पर नई ज्ञान के प्राप्ति के लिए सिस्टमैटिक अध्ययन है, जबकि आविष्कार नए और अज्ञात ज्ञान की खोज है।

सुविचार:	
1.	**समस्याओं का समाधान करने का उतम मार्ग हैं |** → शिक्षा ,RID, प्रतिभा, सहयोग, एकता एवं समाजिक-कार्य
2.	**एक इंसान के लिए जरूरी हैं |** → रोटी, कपड़ा, मकान, शिक्षा, रोजगार, इज्जत और सम्मान
3.	**एक देश के लिए जरूरी हैं |** - → संस्कृति-सभ्यता, भाषा, एकता, आजादी, संविधान एवं अखंडता
4.	**सफलता पाने के लिए होना चाहिए |** → लक्ष्य, त्याग, इच्छा-शक्ति, प्रतिबद्धता, प्रतिभा, एवं सतता
5.	**मरने के बाद इंसान छोडकर जाता हैं |** → शरीर, अन-धन, घर-परिवार, नाम, कर्म एवं विचार
6.	**मरने के बाद इंसान को इस धरती पर याद किया जाता हैं उनके**
➔ नाम, काम, दान, विचार, सेवा-समपर्ण एवं कर्मो से...	

आशीर्वाद (बड़े भैया जी)

Mr. RAMASHANKAR KUMAR

मार्गदर्शन एवं सहयोग

Mr. GAUTAM KUMAR

.सोच है जिनकी नई कुछ कर दिखाने की, खोज है रीड संस्था को उन सभी इंसानों की.

"अगर आप भी **Research, Innovation and Discovery** के क्षेत्र में रूचि रखतें हैं एवं अपनी प्रतिभा से दुनियां को कुछ नया देना चाहतें एवं अपनी समस्या का समाधान **RID** के माध्यम से करना चाहतें हैं तो **RID ORGANIZATION (रीड संस्था)** से जरुर जुड़ें" || **धन्यवाद** || **Er. Rajesh Prasad (B.E, M.E)**

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.

HOW TO APPLY CSS IN HTML

❖ 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 {
   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. 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.

CSS COMMENTS

- Comments are used to explain the code, and may help when you edit the source code at a later date.
- Comments are ignored by browsers.
- A CSS comment is placed inside the <style> element, and starts with /* and ends with */:

Example:

```
/* This is a single-line comment */
p {
  color: red;
}
```

You can add comments wherever you want in the code:

```
p {
  color: red;  /* Set text color to red */
}
p {
  color: /*red*/blue;
}
```

Priority of adding CSS styles in HTML

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

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:

```
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:

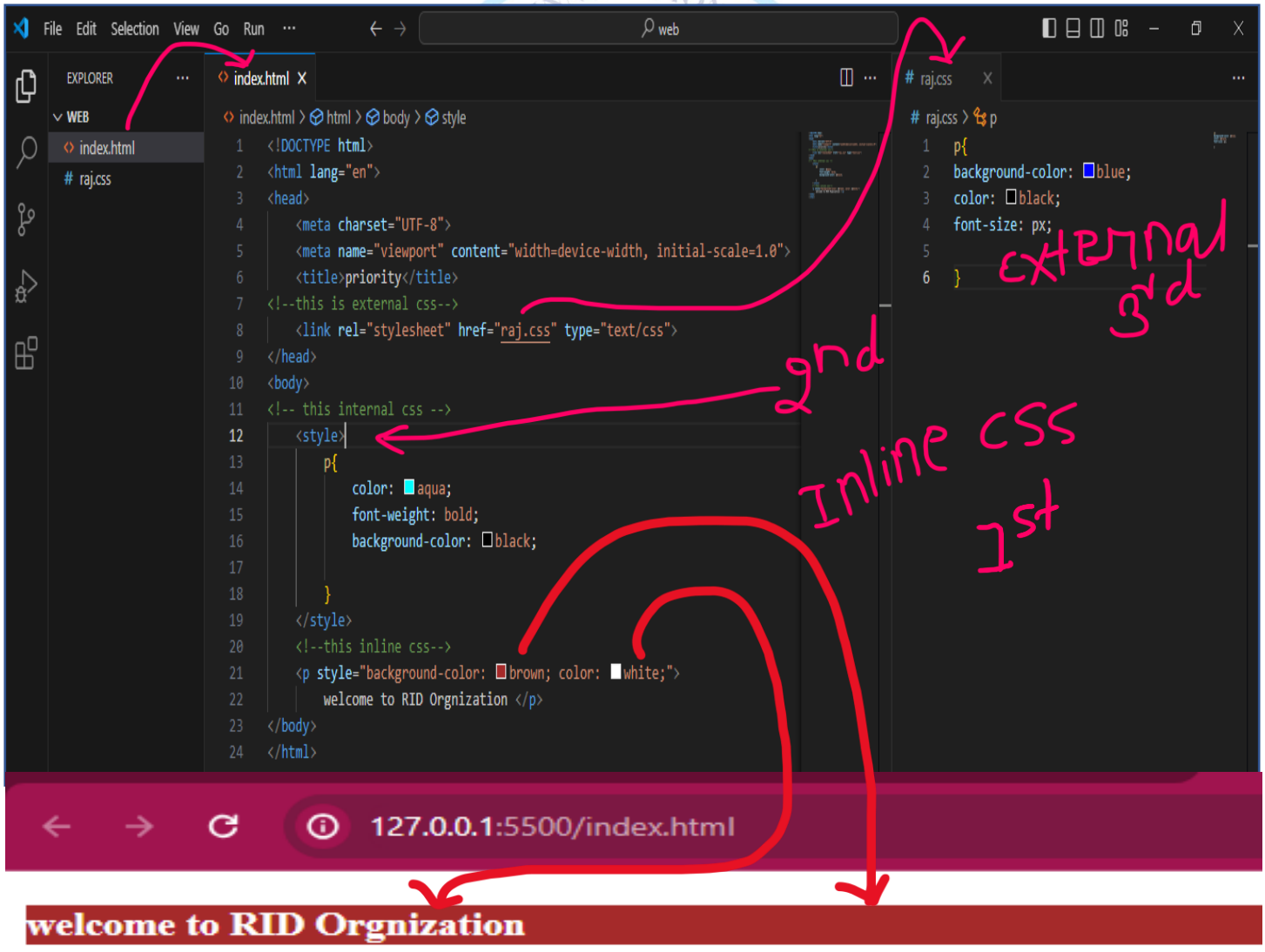
```
p {
   color: orange;
}
p {
   color: pink;
}
```

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;
}
```



SELECTOR IN CSS

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;
}
```

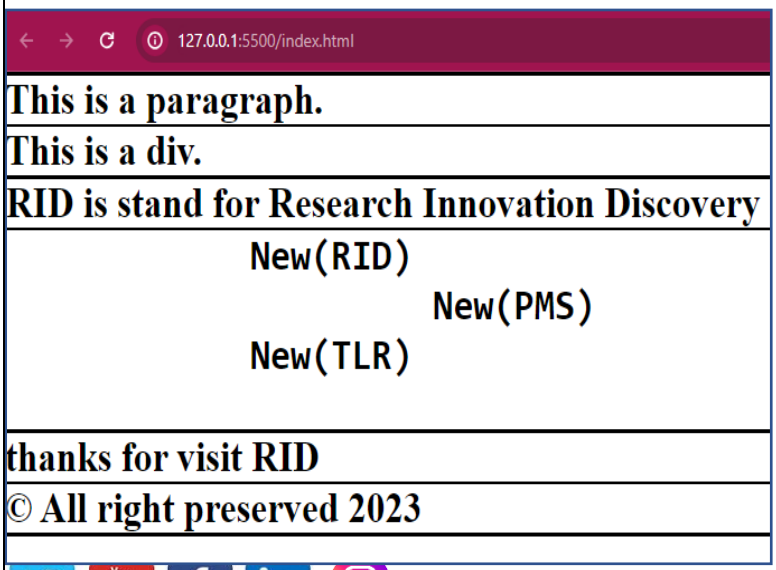
- This selects all elements with

CSS SELECTOR EXAMPLE WITH HTML

1. Universal Selector: `*` : Selects all elements on the page.

Example: index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Universal selector </title>
  <style>
    * {
      margin: 0;
      padding: 0;
      border: 1px solid black;
      font-size: 30px;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <p>This is a paragraph.</p>
  <div>This is a div.</div>
  <div>
    <h1>RID is stand for Research Innovation Discovery</h1>
    <pre>
      New(RID)
          New(PMS)
      New(TLR)
      </pre>
  </div>
  <p>thanks for visit RID</p>
  <footer>&copy; All right preserved 2023</footer>
</body></html>
```

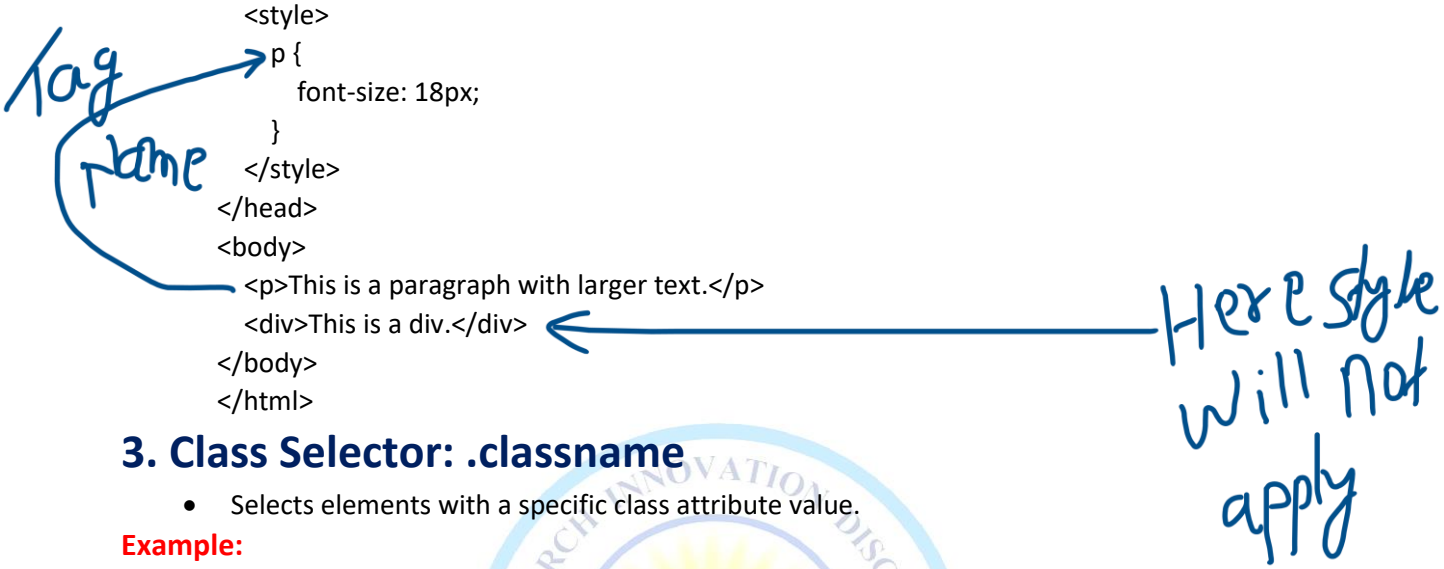


2. Type Selector (Element Selector): element name

- Selects elements of a specific type. Select with tag name

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>
```

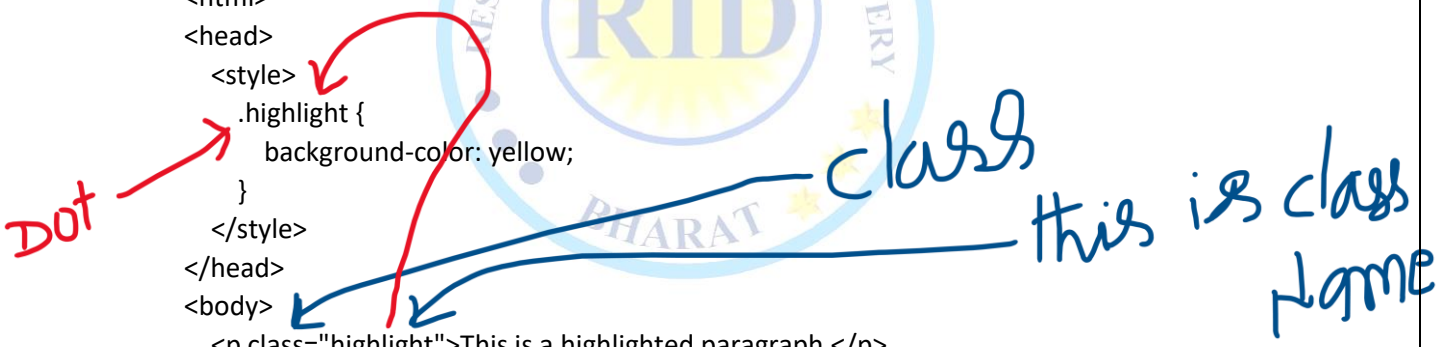


3. Class Selector: .classname

- 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>
```

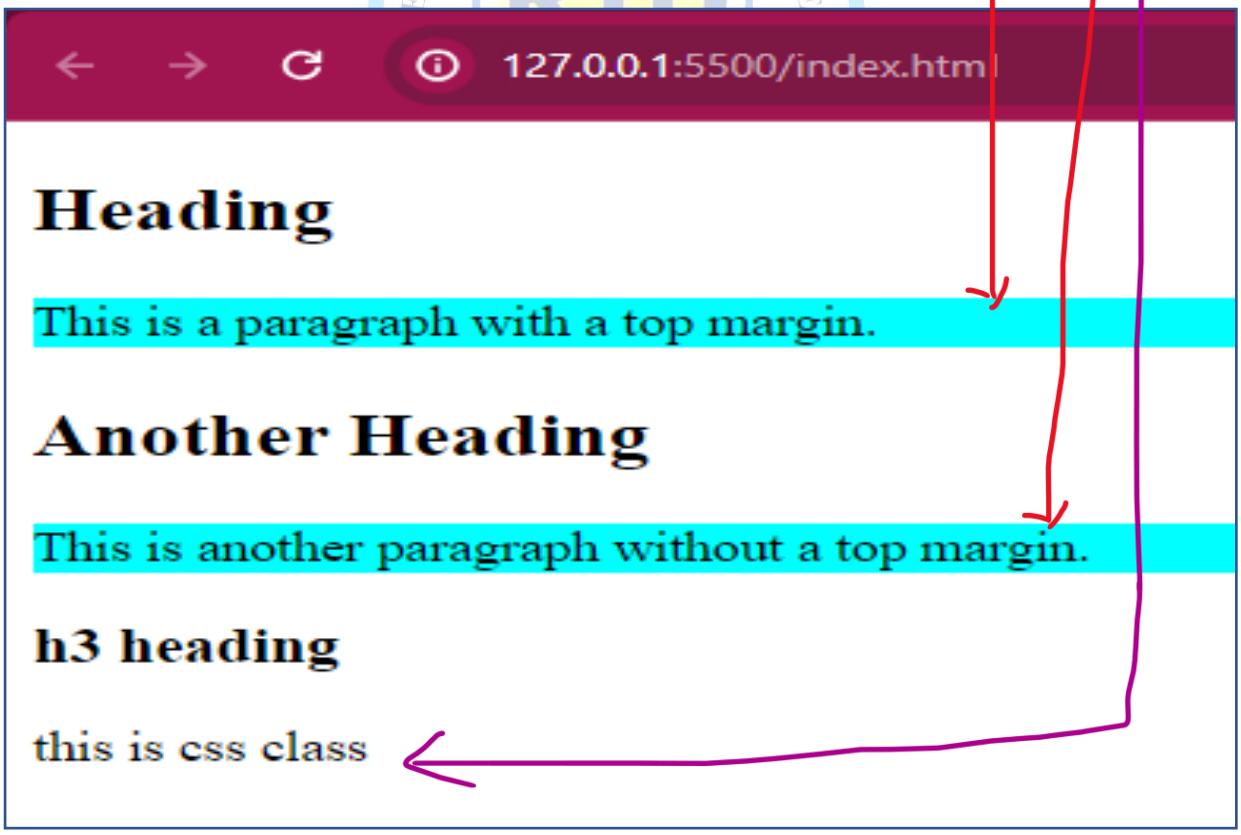
7. Adjacent Sibling Selector: element1 + element2

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

Example:

```
<!DOCTYPE html>
<html>
<head>
    <style>
        h2 + p {
            margin-top: 10px;
            background-color: aqua;
        }
    </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>
    <!--style will not apply on h3 heading <p> -->
    <h3>h3 heading </h3>
    <p>this is css class </p>
</body>
</html>
```

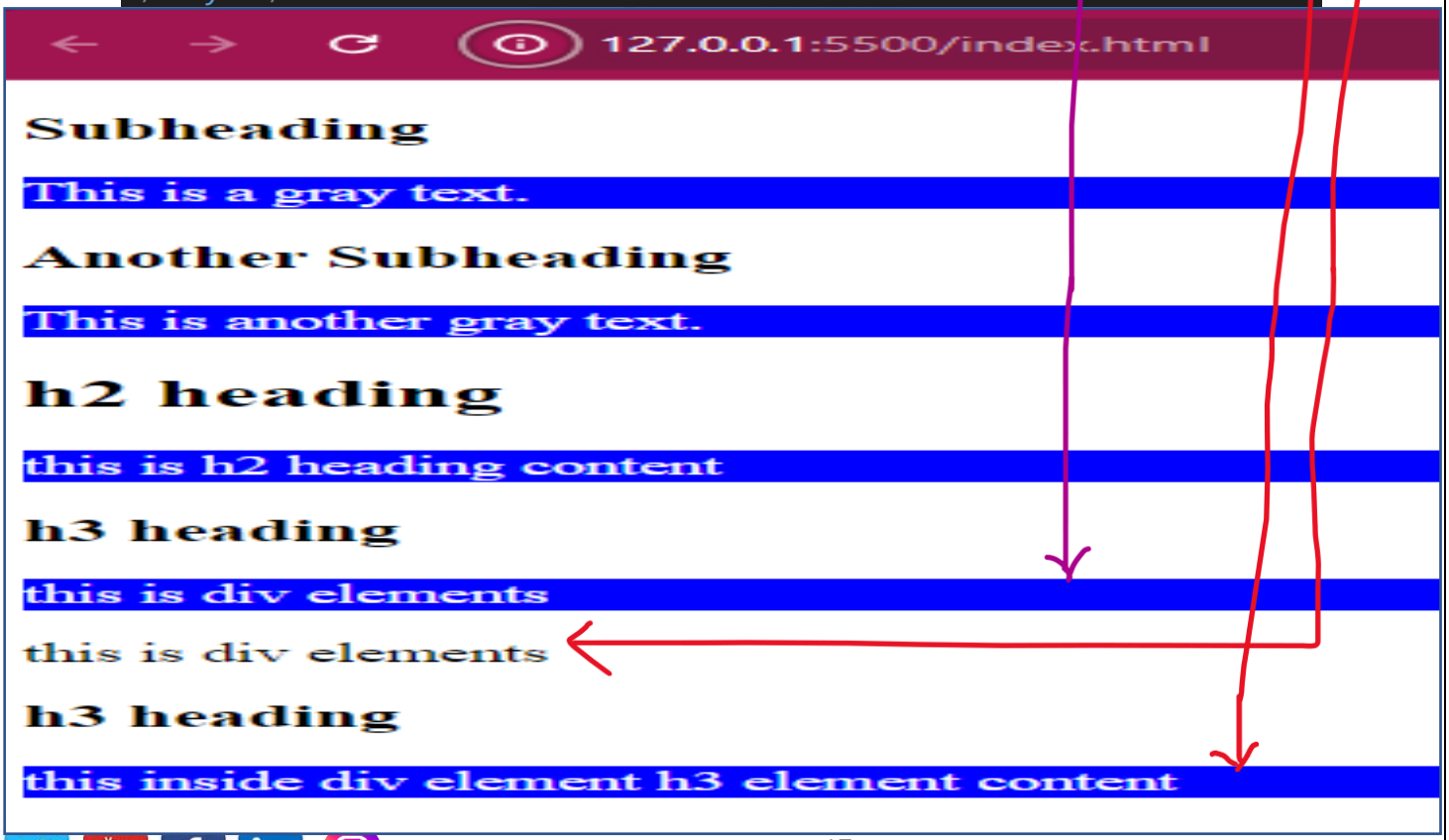
h2 + p



8. General Sibling Selector: element1 ~ element2

- Selects all sibling elements that follow a specified element.

```
<!DOCTYPE html>
<html>
<head>
    <style>
        h3 ~ p {
            color: white;
            background-color: blue;
        }
    </style>
</head> <body>
    <h3>Subheading</h3>
    <p>This is a gray text.</p>
    <h3>Another Subheading</h3>
    <p>This is another gray text.</p>
    <h2>h2 heading</h2>
    <p>this is h2 heading content</p>
    <div>
        <h3>h3 heading </h3>
    <p>this is div elements </p>
    </div>
    <div>
    <p>this is div elements </p>
    <h3>h3 heading</h3>
    <p>this inside div element h3 element content</p>
    </div>
</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 text Properties

1. text-align:

1) **center:** Aligns text to the center of the container.
2) **end:** Aligns text to the end of the container (right-aligned for left-to-right languages, left-aligned for right-to-left languages).
3) **left:** Aligns text to the left of the container.
4) **right:** Aligns text to the right of the container.
5) **start:** Aligns text to the start of the container (left-aligned for left-to-right languages, right-aligned for right-to-left languages).
6) **justify:** Stretches the lines of text so that each line has equal width, except for the last line.
7) **unset:** Resets the text-align property to its inherited value if it has one, otherwise behaves like initial.
8) **initial:** Sets the text-align property to its default value, which is typically start (left-aligned for left-to-right languages).

2. text-decoration:

1) **none:** Removes any text decoration (such as underline, overline, line-through).
2) **underline:** Adds an underline to the text.
3) **overline:** Adds a line above the text.
4) **line-through:** Adds a line through the middle of the text.
5) **underline overline:** Adds both underline and overline to the text.
6) **underline line-through:** Adds underline and line-through to the text.
7) **overline line-through:** Adds overline and line-through to the text.
8) **underline overline line-through:** Adds underline, overline, and line-through to the text.

3. text-transform:

1) **none:** No capitalization change.
2) **capitalize:** Transforms the first character of each word to uppercase.
3) **uppercase:** Transforms all characters to uppercase.
4) **lowercase:** Transforms all characters to lowercase.
5) **full-width:** Converts the characters to their full-width variant (useful for certain Asian languages).

4. text-overflow:

1) **clip:** Clips the text if it overflows its container.
2) **ellipsis:** Displays an ellipsis (...) to indicate that the text has overflowed its container.

5. text-shadow:

- This property accepts values for the horizontal offset, vertical offset, blur radius, and color of the shadow, such as text-shadow: 1px 1px 2px #000;.

6. text-rendering:

1) **auto:** Browser chooses the rendering mode, typically optimized for readability and speed.
2) **optimizeSpeed:** Prioritizes speed over quality.
3) **optimizeLegibility:** Prioritizes readability over speed.

4) **geometricPrecision:** Optimizes text for geometric precision, useful for small text sizes or when zoomed.

7. text-indent:

- This property specifies the indentation of the first line of text in a block container. For example, text-indent: 20px; indents the first line by 20 pixels.

8. text-emphasis:

1) **none:** No emphasis.
2) **accent:** Adds emphasis marks above each character.
3) **dot:** Adds dots below each character.
4) **circle:** Adds circles below each character.
5) **double-circle:** Adds double circles below each character.
6) **triangle:** Adds triangles below each character.
7) **sesame:** Adds sesame seed shapes below each character.

9. text-decoration-line:

1) **none:** No text decoration.
2) **underline:** Underlines the text.
3) **overline:** Adds a line above the text.
4) **line-through:** Adds a line through the middle of the text.

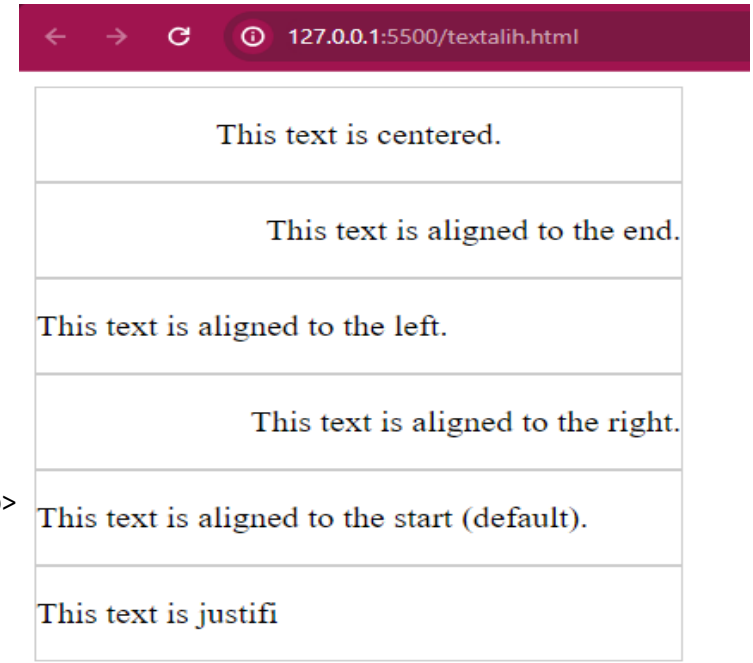
10. text-wrap:

1) **normal:** Allows long words to be broken and wrapped onto the next line if needed.
2) **nowrap:** Prevents wrapping of text. Text will overflow its container if it's too long.
3) **pre:** Preserves whitespace and line breaks in the text, without wrapping.
4) **pre-wrap:** Preserves whitespace and line breaks, but allows text wrapping.
5) **pre-line:** Preserves line breaks but collapses whitespace and allows wrapping.

Example-1 Text-align:

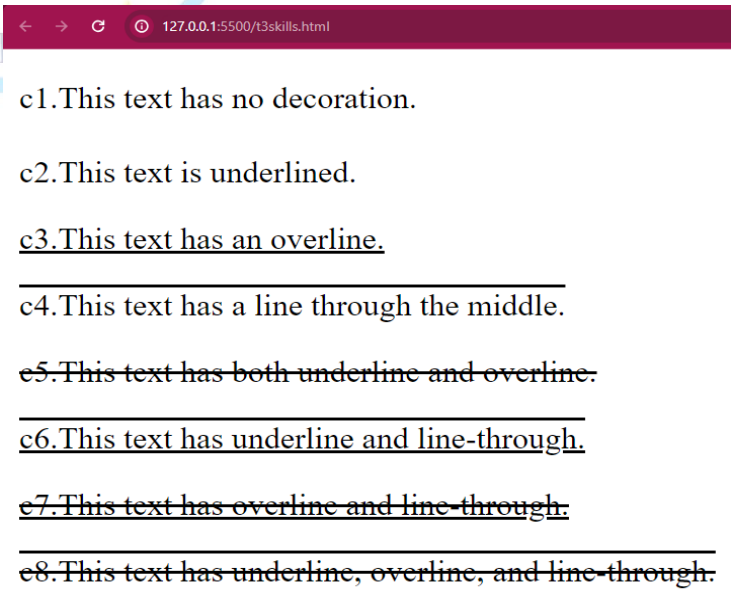
```
<!DOCTYPE html>
<html lang="en"><head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Text Alignment Examples</title><style>
  .container {
    width: 300px;
    border: 1px solid #ccc;
  }
  .center { text-align: center;}
  .end { text-align: end;}
  .left { text-align: left;}
  .right { text-align: right;}
  .start { text-align: start;}
  .justify { text-align: justify;}
  .unset { text-align: unset;}
  .initial { text-align: initial;}
</style></head><body>
<div class="container center">
  <p>This text is centered.</p>
```

```
</div>
<div class="container end">
  <p>This text is aligned to the end.</p>
</div>
<div class="container left">
  <p>This text is aligned to the left.</p>
</div>
<div class="container right">
  <p>This text is aligned to the right.</p>
</div>
<div class="container start">
  <p>This text is aligned to the start (default).</p>
</div>
<div class="container justify">
  <p>This text is justifying</p>
  </div> </body> </html>
```



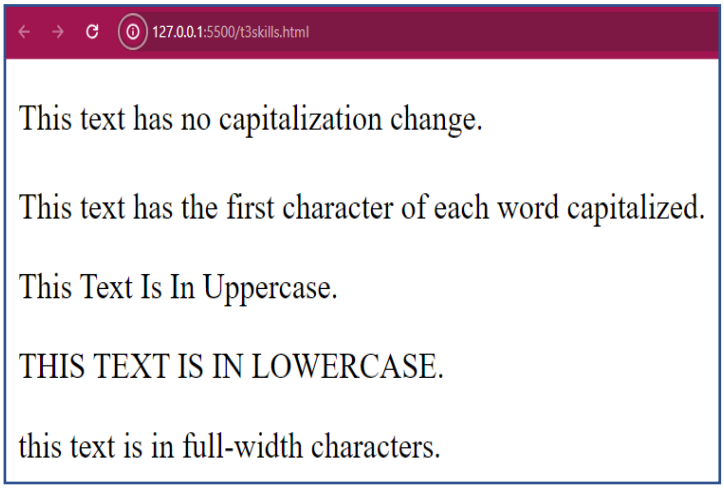
Example-2: text-decoration

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Text Decoration Examples</title>
<style>
  .c1 {
    margin-bottom: 20px;
  }
  .c2 { text-decoration: none; }
  .c3 { text-decoration: underline; }
  .c4 { text-decoration: overline; }
  .c5 { text-decoration: line-through; }
  .c6 { text-decoration: underline overline; }
  .c7 { text-decoration: underline line-through; }
  .c8 { text-decoration: overline line-through; }
  .c9 { text-decoration: underline overline line-through; }
</style>
</head>
<body>
<p class="c1">c1.This text has no decoration.</p>
<p class="c2">c2.This text is underlined.</p>
<p class="c3">c3.This text has an overline.</p>
<p class="c4">c4.This text has a line through the middle.</p>
<p class="c5">c5.This text has both underline and overline.</p>
<p class="c6">c6.This text has underline and line-through.</p>
<p class="c7">c7.This text has overline and line-through.</p>
<p class="c8">c8.This text has underline, overline, and line-through.</p>
</body> </html>
```



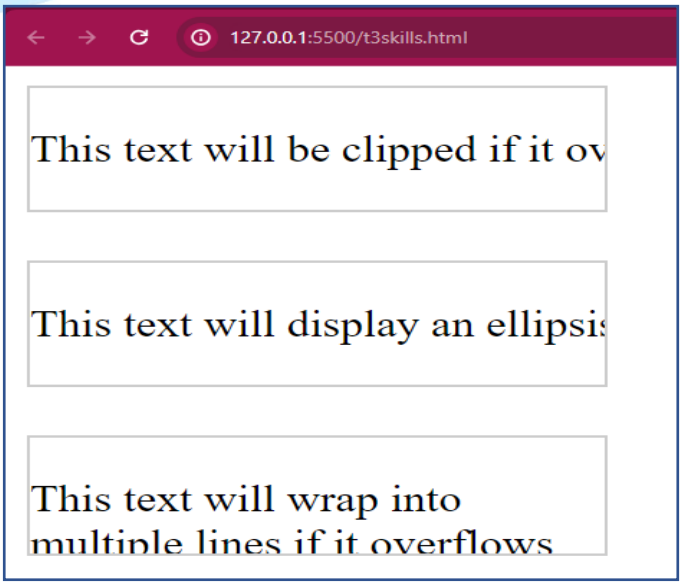
Example-3: text-decoration

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Text Transform Examples</title>
<style>
 .c1 { margin-bottom: 20px; }
 .c2 { text-transform: none; }
 .c3 { text-transform: capitalize; }
 .c4 { text-transform: uppercase; }
 .c5 { text-transform: lowercase; }
 .c6 { text-transform: full-width; }
</style> </head>
<body>
<p class="c1">This text has no capitalization change.</p>
<p class="c2">This text has the first character of each word capitalized.</p>
<p class="c3">This text is in uppercase.</p>
<p class="c4">This text is in lowercase.</p>
<p class="c5">This text is in full-width characters.</p>
</body></html>
```



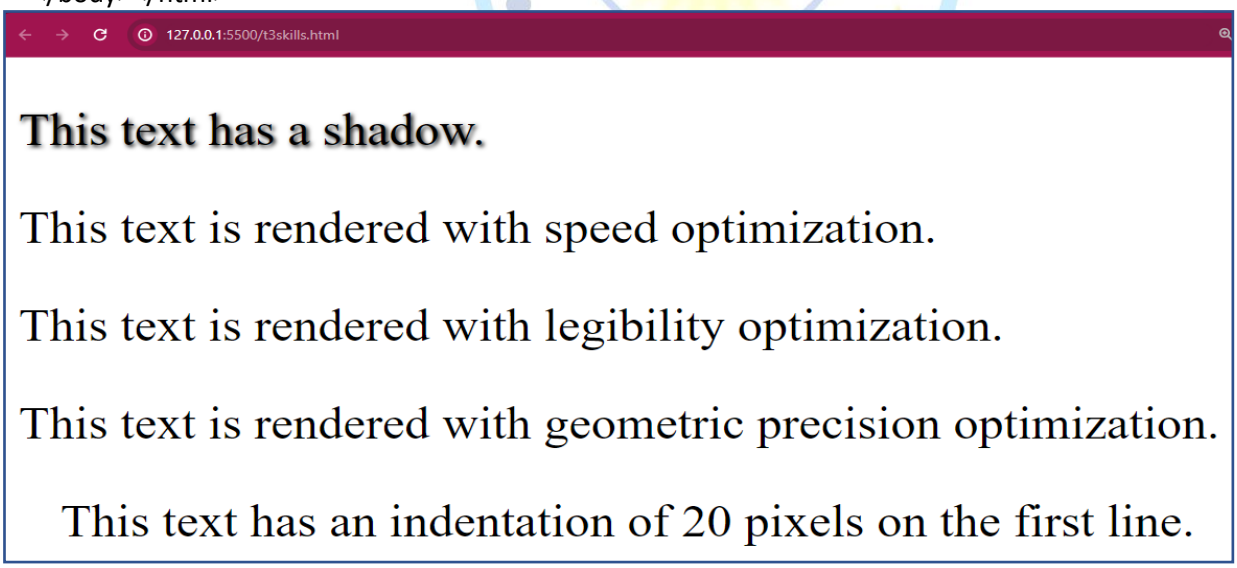
Text overflow Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>More Text Overflow Examples</title>
<style>
 .container {
  width: 200px;
  border: 1px solid #ccc;
  margin-bottom: 20px;
  overflow: hidden;
  white-space: nowrap;
 }
 .clip { text-overflow: clip; }
 .ellipsis { text-overflow: ellipsis; }
 .multiline {
  white-space: normal;
  overflow: hidden;
  height: 3em; }
</style> </head> <body>
<div class="container clip">
 <p>This text will be clipped if it overflows its container.</p>
</div>
<div class="container ellipsis">
  <p>This text will display an ellipsis (...) if it overflows its container.</p> </div>
<div class="container multiline">
 <p>This text will wrap into multiple lines if it overflows its container, but will be limited to 3 lines.</p>
</div></body></html
```



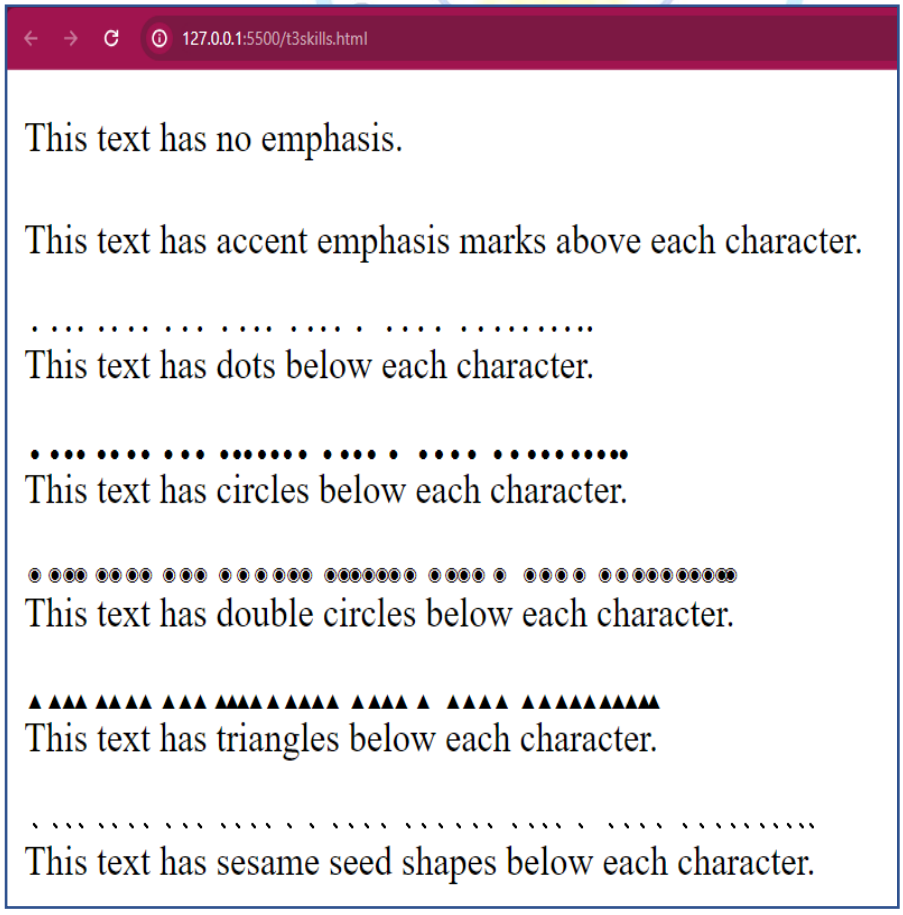
Example-5,6,7

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Text Styling Examples</title>
<style>
  .text {
    margin-bottom: 20px;
    font-size: 24px;
  }
  .shadow {  text-shadow: 1px 1px 2px #000; }
  .optimize-speed {   text-rendering: optimizeSpeed; }
  .optimize-legibility { text-rendering: optimizeLegibility; }
  .geometric-precision { text-rendering: geometricPrecision; }
  .indent {   text-indent: 20px; }
</style></head><body>
<p class="text shadow">This text has a shadow.</p>
<p class="text optimize-speed">This text is rendered with speed optimization.</p>
<p class="text optimize-legibility">This text is rendered with legibility optimization.</p>
<p class="text geometric-precision">This text is rendered with geometric precision optimization.</p>
<p class="text indent">This text has an indentation of 20 pixels on the first line.</p>
</body></html>
```



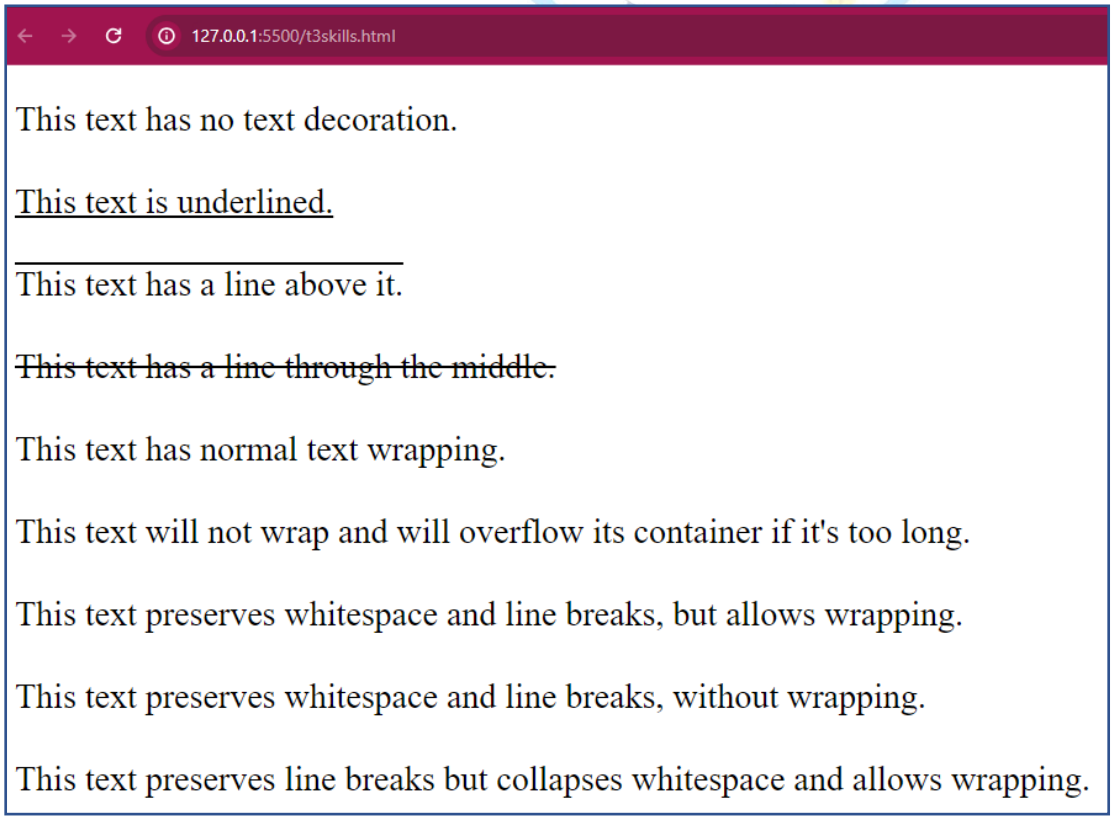
Example-8:text-emphasis

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Text Emphasis Examples</title>
<style>
  .text { margin-bottom: 20px; }
  .none {  text-emphasis: none; }
  .accent { text-emphasis: accent; }
  .dot { text-emphasis: dot; }
  .circle { text-emphasis: circle; }
  .double-circle { text-emphasis: double-circle; }
  .triangle { text-emphasis: triangle; }
  .sesame { text-emphasis: sesame; }
</style>
</head>
<body>
<p class="text none">This text has no emphasis.</p>
<p class="text accent">This text has accent emphasis marks above each character.</p>
<p class="text dot">This text has dots below each character.</p>
<p class="text circle">This text has circles below each character.</p>
<p class="text double-circle">This text has double circles below each character.</p>
<p class="text triangle">This text has triangles below each character.</p>
<p class="text sesame">This text has sesame seed shapes below each character.</p>
</body>
</html>
```



Example-9,10: text-decoration-line

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Text Decoration and Text Wrap Examples</title>
<style>
 .t { margin-bottom: 20px; }
 .none { text-decoration-line: none; }
 .underline { text-decoration-line: underline; }
 .overline { text-decoration-line: overline; }
 .line-through { text-decoration-line: line-through; }
 .nw { /* Text Wrap */ white-space: normal; }
 .nowrap { white-space: nowrap; }
 .pre-wrap { white-space: pre-wrap; }
 .pre { white-space: pre; }
 .pre-line { white-space: pre-line; }
</style></head>
<body>
<p class="t none">This text has no text decoration.</p>
<p class="t underline">This text is underlined.</p>
<p class="t overline">This text has a line above it.</p>
<p class="t line-through">This text has a line through the middle.</p>
<p class="t nw">This text has normal text wrapping.</p>
<p class="t nowrap">This text will not wrap and will overflow its container if it's too long.</p>
<p class="t pre-wrap">This text preserves whitespace and line breaks, but allows wrapping.</p>
<p class="t pre">This text preserves whitespace and line breaks, without wrapping.</p>
<p class="t pre-line">This text preserves line breaks but collapses whitespace and allows wrapping.</p>
</body> </html>
```



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.

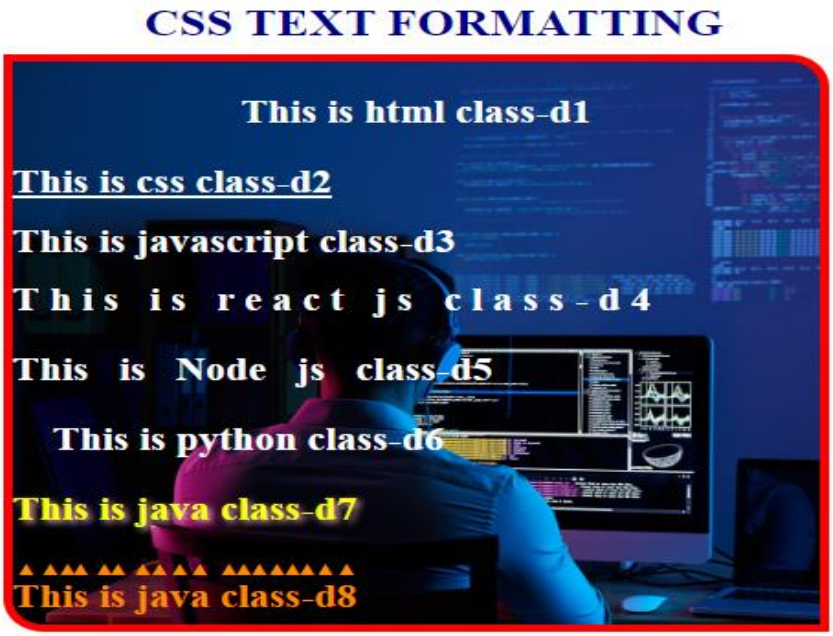
- **Syntax:** text-indent: value; (e.g., text-indent: 20px;)

10. Text Shadow:

- text-shadow: Adds a shadow effect to text.
- Syntax: text-shadow: h-shadow v-shadow blur-radius color;

Example:

```
<!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>
      .c1{
         width: 400px;
         height: 57vh;
         border: 5px solid red;
         border-radius: 0PX 20PX;
         background-image: url('img.jpg')
         background-size: cover;
         color: white;
         font-weight: bold;
         font-size: 20px;}
      #d1{ text-align: center;}
      #d2{ text-decoration: underline;}
      #d3{ line-height: 10px;}
      #d4{ letter-spacing: 5px;}
      #d5{ word-spacing: 10px;}
      #d6{ text-indent: 20px;}
      #d7{ text-shadow:  2px 2px 4px pink; color: yellow;}
      #d8{ text-emphasis: triangle; color: darkorange;}
      h2{ margin-left: 70px;
         line-height: 0px;
         color: darkblue;}
   </style></head>
<body>
   <h2>CSS TEXT FORMATTING</h2>
   <div class="c1">
      <p id="d1">This is html class-d1 </p>
      <p id="d2">This is css class-d2 </p>
      <p id="d3">This is JavaScript class-d3 </p>
      <p id="d4">This is react js class-d4 </p>
      <p id="d5">This is Node js class-d5 </p>
      <p id="d6">This is python class-d6 </p>
      <p id="d7">This is java class-d7 </p>
      <p id="d8">This is java class-d8 </p>
   </div>
</body>  </html></body></html>
```



CSS Background

❖CSS background properties

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

Color Name	Hex Code #RRGGBB	Decimal Code (R,G,B)
1. Lightcyan	#E0FFFF	rgb(224,255,255)
2. cyan	#00FFFF	rgb(0,255,255)
3. aqua	#00FFFF	rgb(0,255,255)
4. aquamarine	#7FFFD4	rgb(127,255,212)

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;
- **It is apply on background image only**

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

8. background-clip:

- **Syntax:** background-clip: padding-box|border-box|content-box;
- **It is apply on background color only**

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

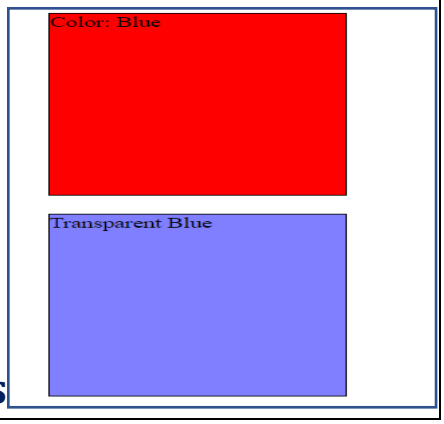
9. background-blend-mode:

- **Syntax:** background-blend-mode: normal|multiply|screen|overlay|...;
- **It is used for add multiple image and color in background.**

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

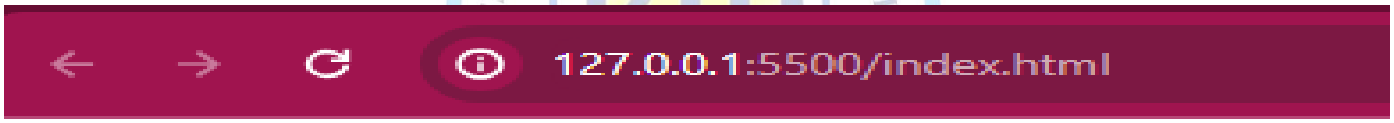
1.Example of background-color` property in HTML and CSS:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Background Colors Example</title>
  <style>
    .background-example {
      width: 200px;
      height: 200px;
      margin: 20px;
      border: 1px solid black;
    }
    .color-red { background-color: red; }
    .color-green { background-color: #00ff00; }
    .color-blue { background-color: rgb(255, 0, 0); }
    .color-transparent-blue { background-color: rgba(0, 0, 255, 0.5); }
    .color-hsl { background-color: hsl(120, 100%, 50%); }
    .color-hsla { background-color: hsla(120, 100%, 50%, 0.5); }
    .color-transparent { background-color: transparent; }
  </style>
</head>
<body>
  <div class="background-example color-red">Color: Red</div>
  <div class="background-example color-green">Color: Green</div>
  <div class="background-example color-blue">Color: Blue</div>
  <div class="background-example color-transparent-blue">Transparent Blue</div>
  <div class="background-example color-hsl">HSL Color</div>
  <div class="background-example color-hsla">HSLA Color</div>
  <div class="background-example color-transparent">Transparent</div>
</body> </html>
```



- **raj.html**

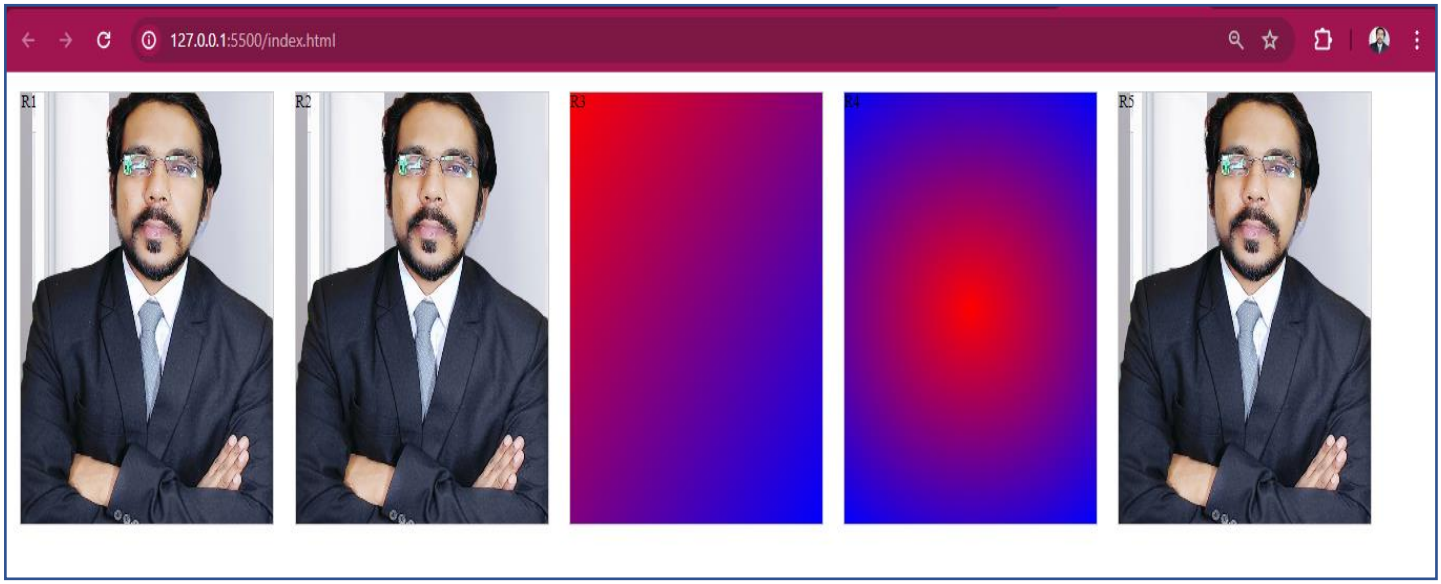
```
<!DOCTYPE html>
<html>
<head>
 <title>Background Color Example</title>
 <style>
  /* Define a CSS class with a background color */
  .colored-box {
   width: 300px;
   height: 100px;
   background-color: #3498db; /* Set the background color to a shade of blue */
   color: #fff; /* Set the text color 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:

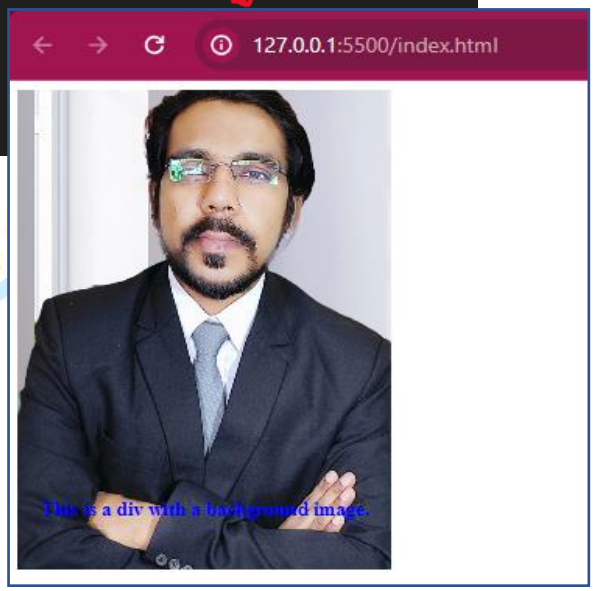
1) **/* Using a single image */**
   - background-image: url('example.jpg');
2) **/* Using multiple images (stacked on top of each other) */**
   - background-image: url('pattern1.jpg'), url('pattern2.jpg');
3) **/* Using linear gradient */**
   - background-image: linear-gradient(to bottom right, red, blue);
4) **/* Using radial gradient */**
   - background-image: radial-gradient(circle, red, blue);
5) **/* Using a combination of image and gradient */**
   - background-image: url('image.jpg'), linear-gradient(to bottom, rgba(255,255,255,0), rgba(255,255,255,1));

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Background Image Examples</title>
  <style>
    .background-example {
      width: 300px;
      height: 400px;
      margin: 10px;
      border: 1px solid #ccc;
      display: inline-block;
      background-size: cover;
      background-position: center;
    }
    .example-1 { background-image: url('myimg.jpeg'); }
    .example-2 { background-image: url('myimg.jpeg'), url('Picture1.png'); }
    .example-3 { background-image: linear-gradient(to bottom right, red, blue); }
    .example-4 { background-image: radial-gradient(circle, red, blue); }
    .example-5 { background-image: url(myimg.jpeg), linear-gradient(to bottom,
rgba(255,255,255,0), rgba(255,255,255,1)); }
  </style>
</head>
<body>
  <div class="background-example example-1">R1</div>
  <div class="background-example example-2">R2</div>
  <div class="background-example example-3">R3</div>
  <div class="background-example example-4">R4</div>
  <div class="background-example example-5">R5</div>
</body> </html>
```



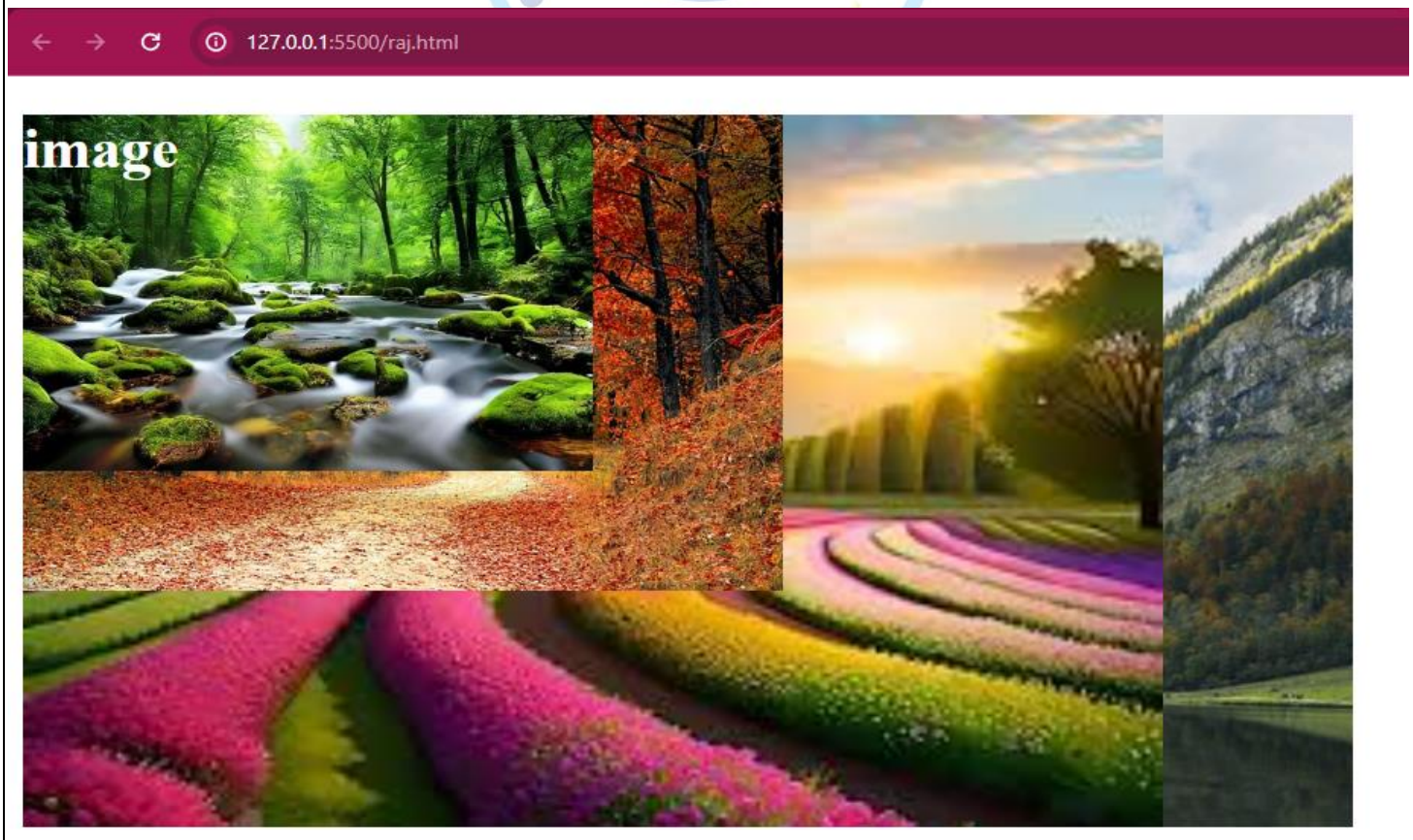
➢ **Raj.html**

```
<!DOCTYPE html>
<html>
<head>
  <title>Background Image Example</title>
  <style>
    .image-box {
      width: 300px;
      height: 400px;
      color: blue;
      font-weight: bold;
      background-image: url('myimg.jpeg');
      background-size: cover;
      background-repeat: no-repeat;
      background-position: center center;
      text-align: center;
      line-height: 700px;
    }
  </style>
</head>
<body>

  <div class="image-box">
    This is a div with a background image.
  </div>
</body></html>
```



How to add Multiple images in Background

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Background Repeat Example - repeat</title>
<style>
   .container {
      width: 100%;
      height: 400px;
      background-image: url(img2.jpg), url(img3.jpg),url(img4.jfif), url(img5.jpg);
      background-size: 300px, 400px,600px,700px,800px;
      background-repeat: no-repeat,no-repeat,no-repeat,no-repeat;
      background-position: auto;
      color: white;
   }
</style>
</head>
<body>
<div class="container">
   <h1>image</h1>
</div>
</body>
</html>
```

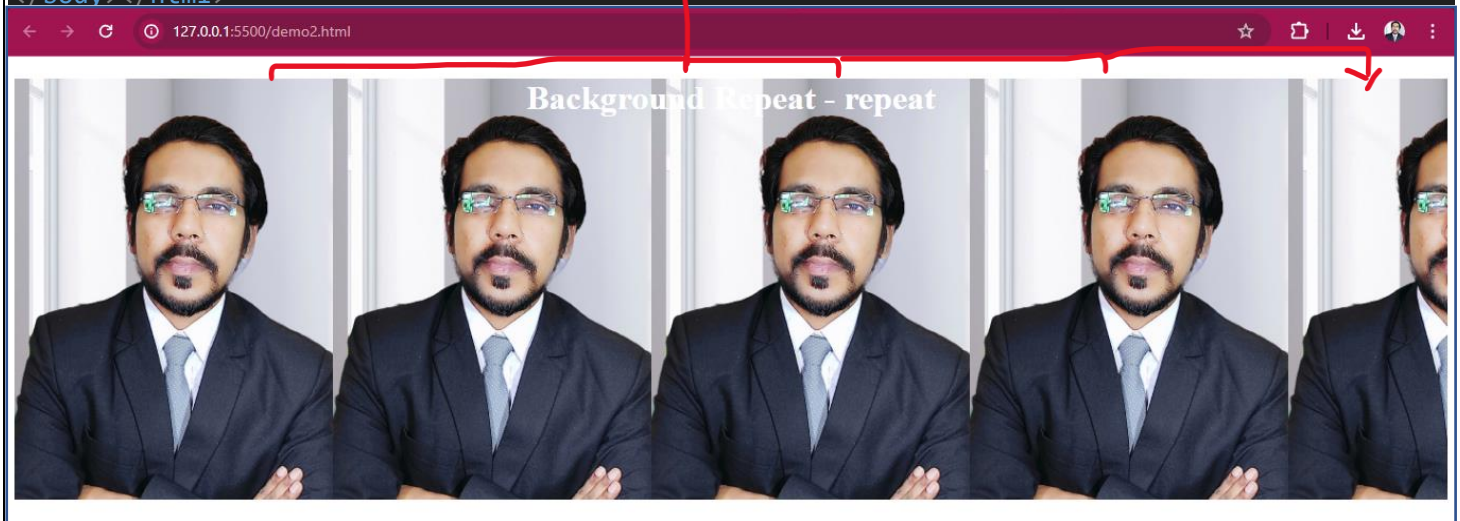



3. background-repeat:

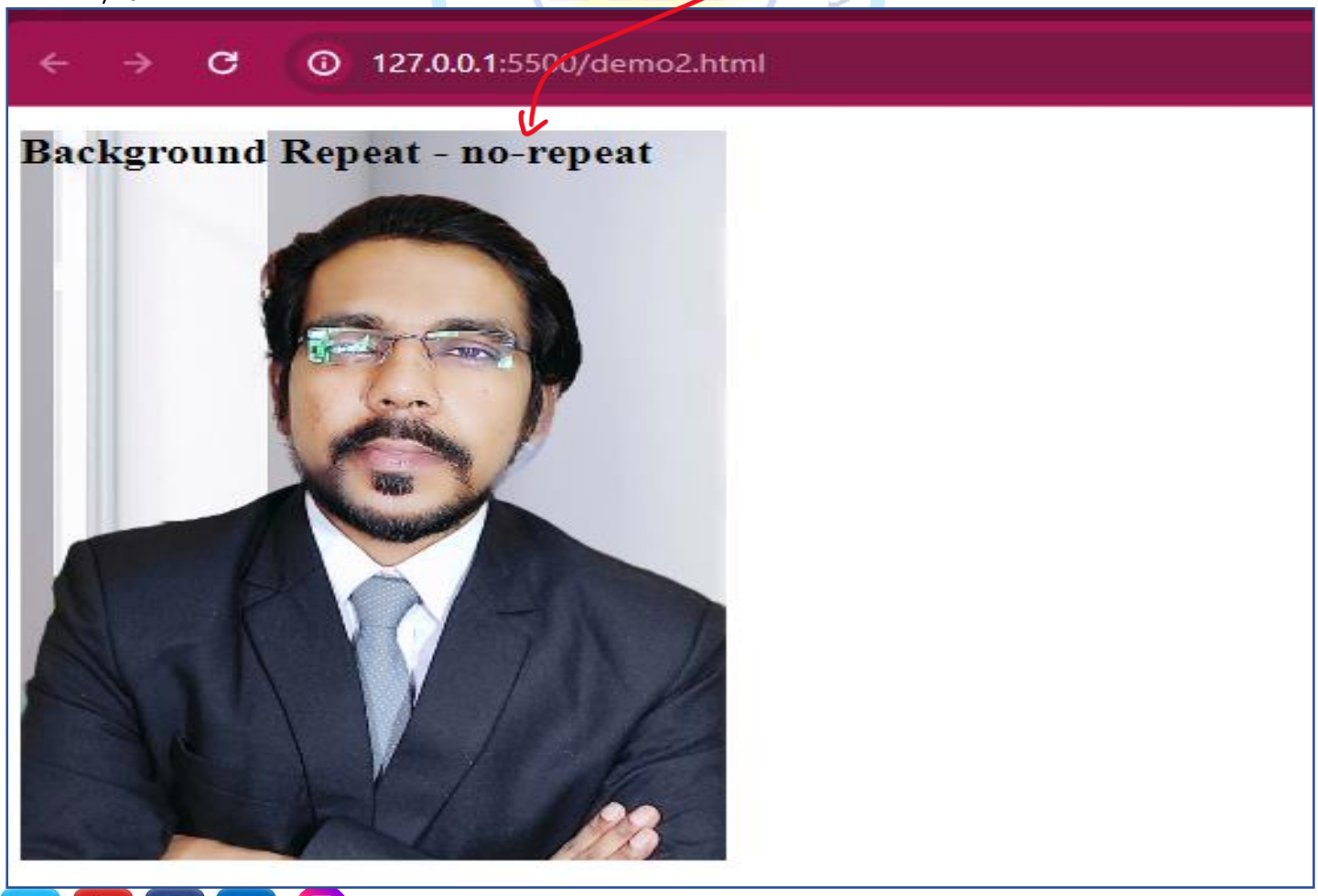
1) **repeat:** background image is repeated both horizontally and vertically to fill the container.
2) **repeat-x:** The background image is repeated only horizontally.
3) **repeat-y:** The background image is repeated only vertically.
4) **no-repeat:** The background image is displayed only once, and it's not repeated.
5) **space:** The background image is repeated both horizontally and vertically, but the spacing between each image is adjusted so that they fill the container without clipping at the edges.
6) **round:** Similar to space, the background image is repeated both horizontally and vertically, but the images are stretched or squeezed as necessary to avoid clipping at the edges.

Example:

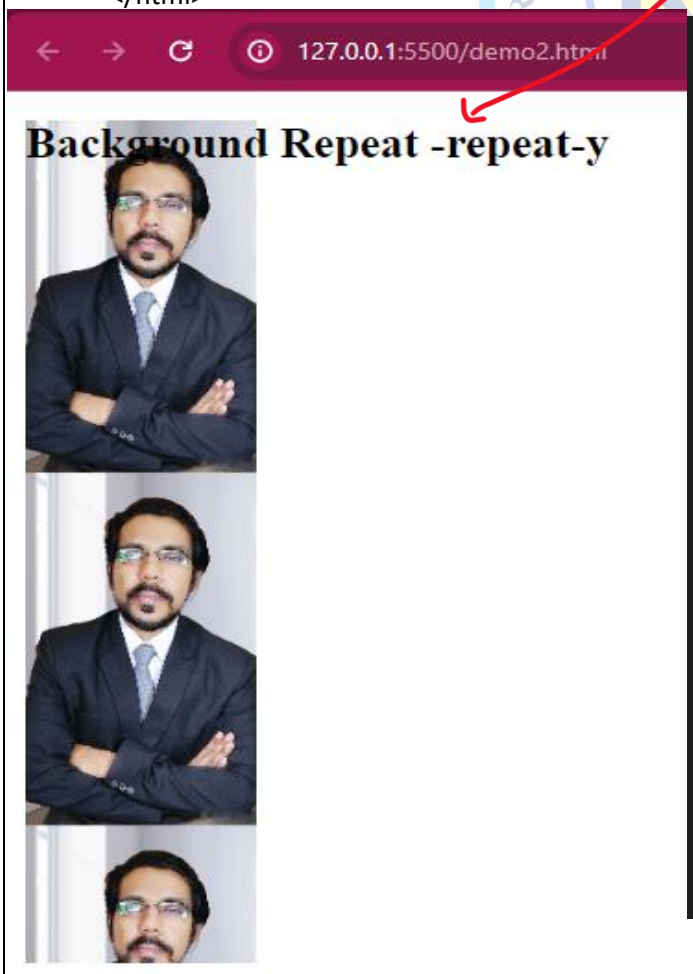
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Background Repeat Example - repeat</title>
<style>
    .container {
        width: 100%;
        height: 400px;
        background-image: url(img2.jpg);
        background-size: 300px;
        background-repeat: repeat;  /*background-repeat: repeat-x; */ /*background-repeat:
space; */,                          /* background-repeat: round; */

        color: white;
        text-align: center;
    }
</style>
</head>
<body>
<div class="container">
    <h1>Background Repeat - repeat</h1>
</div>
</body></html>
```



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Background Repeat Example - repeat</title>
<style>
   .container {
      width: 100%;
      height: 400px;
      background-image: url(myimg.jpeg);
      background-size: 300px;
      background-repeat: no-repeat;
      color: black;
      font-weight: bold;
      font-size: 10px;
   }
</style>
</head>
<body>
<div class="container">
   <h1>Background Repeat - no-repeat</h1>
</div>
</body>
</html>
```



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Background Repeat Example - repeat</title>
<style>
   .container {
      width: 100%;
      height: 400px;
      background-image: url(myimg.jpeg);
      background-size: 100px;
      background-repeat: repeat-y;
      color: black;
      font-weight: bold;
      font-size: 10px;
   }
</style>
</head>
<body>
<div class="container">
   <h1>Background Repeat -repeat-y</h1>
</div>
</body>
</html>
```




```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Background Repeat Example - round</title>
<style>
    .container {
        width: 100%;
        height: 400px;
        background-image: url(myimg.jpeg);
        background-size: 100px;
        background-repeat: round;
        color: black;
        font-weight: bold;
        font-size: 10px;
    }
</style></head>
<body>
<div class="container">
    <h1>Background Repeat round</h1>
</div>
</body>
</html>
```

4.BACKGROUND-POSITION

- **background-position** property in CSS allows you to specify the starting position of a background image within its container. It takes two values: **horizontal position and vertical position**. You can use keywords, **percentage values**, or **length values** to define these positions.

1. Keywords:

- Keywords such as **left, right, center, top, and bottom** can be used to specify the position of the background image relative to the container.

Example:
```
.container {
    background-image: url('background-image.jpg');
    background-repeat: no-repeat; }
.left-top { background-position: left top; }
.left-center { background-position: left center; }
.left-bottom { background-position: left bottom; }
```

1) **left top:** The background image is positioned at the top-left corner of the container.
2) **left center:** The background image is positioned at the left center of the container.
3) **left bottom:** The background image is positioned at the bottom-left corner of the container.
4) **right top:** The background image is positioned at the top-right corner of the container.
5) **right center:** The background image is positioned at the right center of the container.
6) **right bottom:** background image is positioned at the bottom-right corner of the container.
7) **center top:** background image is positioned at the center of the top edge of the container.
8) **center center:** The background image is positioned at the center of the container.
9) **center bottom:** background image is positioned center of the bottom edge of the container.

2. Percentage Values:

- it define position of the background image relative to the container's size. (0% (X-axis), 0% (Y-axis) represents top-left corner, while (100%, 100%) represents the bottom-right corner.

Example:
```
.container {
    background-image: url('background-image.jpg');
    background-repeat: no-repeat;
}
.percentage { background-position: 75% 25%; }
```

3. Length Values:

- Length values, such as pixels or ems, define the position of the background image from the top-left corner of the container.

Example:
```
.container {
    background-image: url('background-image.jpg');
    background-repeat: no-repeat;
}
.length { background-position: 200px 50px; }.
```

4. Combined Positioning:

- You can combine horizontal and vertical positions to fine-tune the placement of the background image.

Example:
```
.container {
    background-image: url('background-image.jpg');
    background-repeat: no-repeat; }
.right-bottom { background-position: right bottom; }
.center-center { background-position: center center; }
```

Example-1 for using keyword:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Background Image Positioning</title>
<style>
   .container {
      width: 50%; height: 50%;
      background-image: url('earth.jpg');
      background-size: cover;
      background-repeat: no-repeat;
   }
   .left-top {
      background-position: left top;
   }
   .left-center {
      background-position: left center;
   }
   .left-bottom {
      background-position: left bottom;
   }
   .right-top {
      background-position: right top;
   }
   .right-center {
      background-position: right center;
   } .right-bottom {
      background-position: right bottom;
   }.center-top {
      background-position: center top;
   }
   .center-center {
      background-position: center center;
   }
   .center-bottom {
      background-position: center bottom;
   }
</style></head> <body>
   <div class="container left-top">Container-1</div>
   <div class="container left-center"> Container-2</div>
   <div class="container left-bottom"> Container -3</div>
   <div class="container right-top"> Container -4</div>
   <div class="container right-center"> Container -5</div>
   <div class="container right-bottom"> Container -6</div>
   <div class="container center-top"> Container -7</div>
   <div class="container center-center"> Container -8</div>
   <div class="container center-bottom"> Container -9</div>
</body> </html>
```

Example-2 for Percentage Values, Length Values, Combined Positioning:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Background Image Positioning</title>
<style>
   body {
      margin: 0;
      padding: 0;
      height: 100vh;
      background-color: #f0f0f0;
   }
   .container {
      width: 50%;
      height: 50%;
      margin: auto;
      border: 2px solid black;
      background-repeat: no-repeat;
   }

   .percentage {
      background-image: url('earth.jpg');
      background-position: 75% 25%;
   }

   .length {
      background-image: url('logo.PNG');
      background-position: 200px 50px;
   }

   .combined {
      background-image: url('myimg.jpeg');
      background-position: right bottom;
   }
</style>
</head>
<body>
   <div class="container percentage">container-1</div>
   <div class="container length">container-2</div>
   <div class="container combined">container-3</div>
</body>
</html>
```

5.background-size:

- background-size property allows you to specify the size of a background image. It can take various values to control how the background image should be sized and scaled within its container.

1. **Length Values:** You can specify the width and height of the background image using length values such as pixels (px) or percentages (%).

Example:

```
.container {
  background-image: url('background-image.jpg');
  background-size: 200px 100px; /* width height */
}
```

- This will set the width of the background image to 200 pixels and the height to 100 pixels.

2. **Cover:** The background image is scaled to be as large as possible while ensuring both its width and height cover the entire container. It might result in clipping the image if the aspect ratio of the container and the background image differ.

Example:

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

3. **Contain:** The background image is scaled to fit within the container while preserving its aspect ratio. It ensures that both the width and height of the image are fully visible within the container, but it may result in empty space within the container if the aspect ratio of the container and the background image differ.

Example:

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

4. **Auto:** The background image is displayed at its original size. It will not be stretched or scaled.

Example:

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

5. **Percentage Values:** You can specify the width and height of the background image as a percentage of the container's size.

Example:

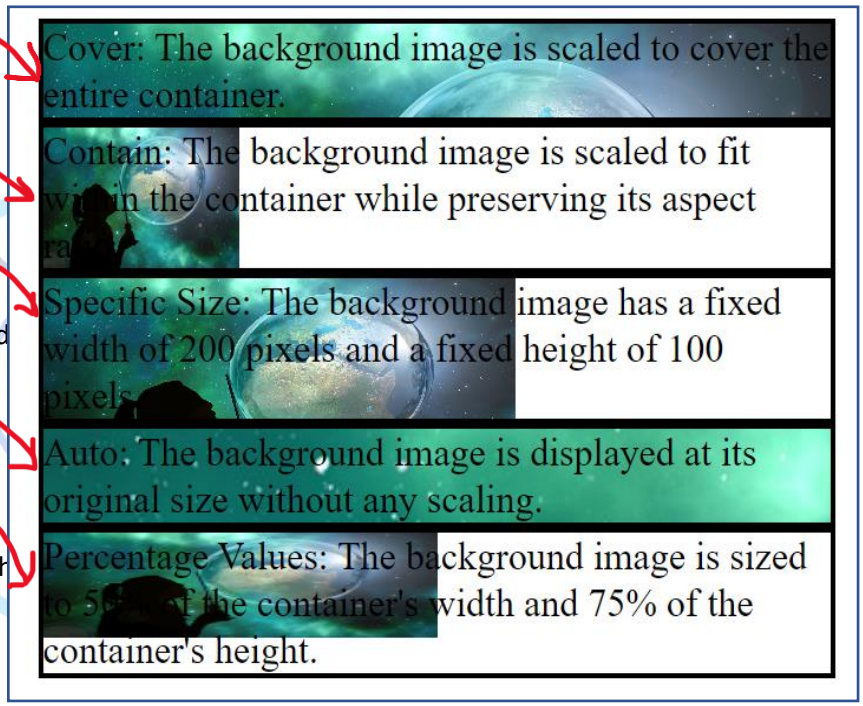
```
.container {
  background-image: url('background-image.jpg');
  background-size: 50% 75%; /* width height */
}
```

- This will set the width of the background image to 50% of the container's width and the height to 75% of the container's height.
- By using these different values for background-size, you can control how the background image is displayed and scaled within its container to achieve the desired visual effect.

Note: cover and contain are the most used background-size attribute values because they provide versatile solutions for achieving different visual effects while maintaining the integrity of the background image.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Background Image with Background-Size</title>
<style>
  .container {
    width: 50%;
    height: 50%;
    margin: auto;
    border: 2px solid black;
    background-image: url('earth.jpg');
    background-repeat: no-repeat;
  }
  .cover {
    background-size: cover;
  }
  .contain {
    background-size: contain;
  }
  .specific-size {
    background-size: 200px 100px; /* wid
  }
  .auto {
    background-size: auto;
  }
  .percentage {
    background-size: 50% 75%; /* width h
  }
</style>
</head>
<body>
  <div class="container cover">
    Cover: The background image is scaled to cover the entire container.
  </div>
  <div class="container contain">
    Contain: The background image is scaled to fit within the container while preserving its aspect ratio.
  </div>
  <div class="container specific-size">
    Specific Size: The background image has a fixed width of 200 pixels and a fixed height of 100 pixels.
  </div>
  <div class="container auto">
    Auto: The background image is displayed at its original size without any scaling.
  </div>
  <div class="container percentage">
    Percentage Values: The background image is sized to 50% of the container's width and 75% of the container's height.
  </div>
```



Example-2:

```
<!DOCTYPE html>
<html>
<head>
 <title>Background Size Example</title>
 <style>
  .sized-box {
   width: 300px;
   height: 200px;
   background-image: url(earth.jpg);
   background-repeat: no-repeat;
   background-size: 100% 100%;
   color: #fff;
   text-align: center;
   line-height: 200px;
  }
 </style>
</head>
<body>
 <div class="sized-box">
  This is a div with a custom background size.
 </div>
</body>
</html>
```

6.background-attachment:

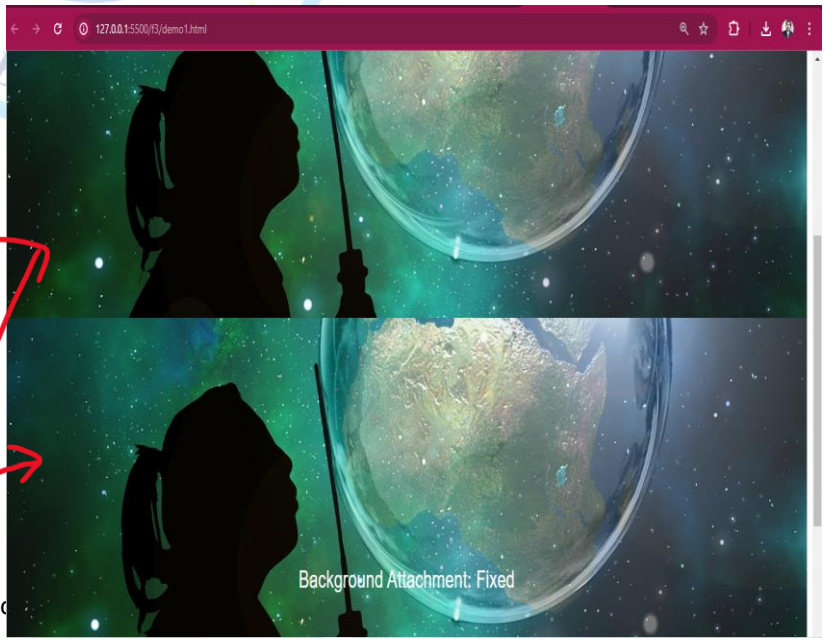
- It is specifies whether the background image should scroll with the content or remain fixed as the content is scrolled. It can take the following values:

1. **scroll:** The background image will scroll along with the content when the user scrolls through the webpage.
2. **fixed:** The background image will remain fixed relative to the viewport. It will not move when the user scrolls the content.
3. **Local:** it used box in side box image will scroll with content

Example-1:
```
.container {
    width: 100%;
    height: 100vh;
    background-image: url('background-image.jpg');
    background-repeat: no-repeat;
    background-size: cover;}
.scroll {
    background-attachment: scroll; }
.fixed {
    background-attachment: fixed; }
```

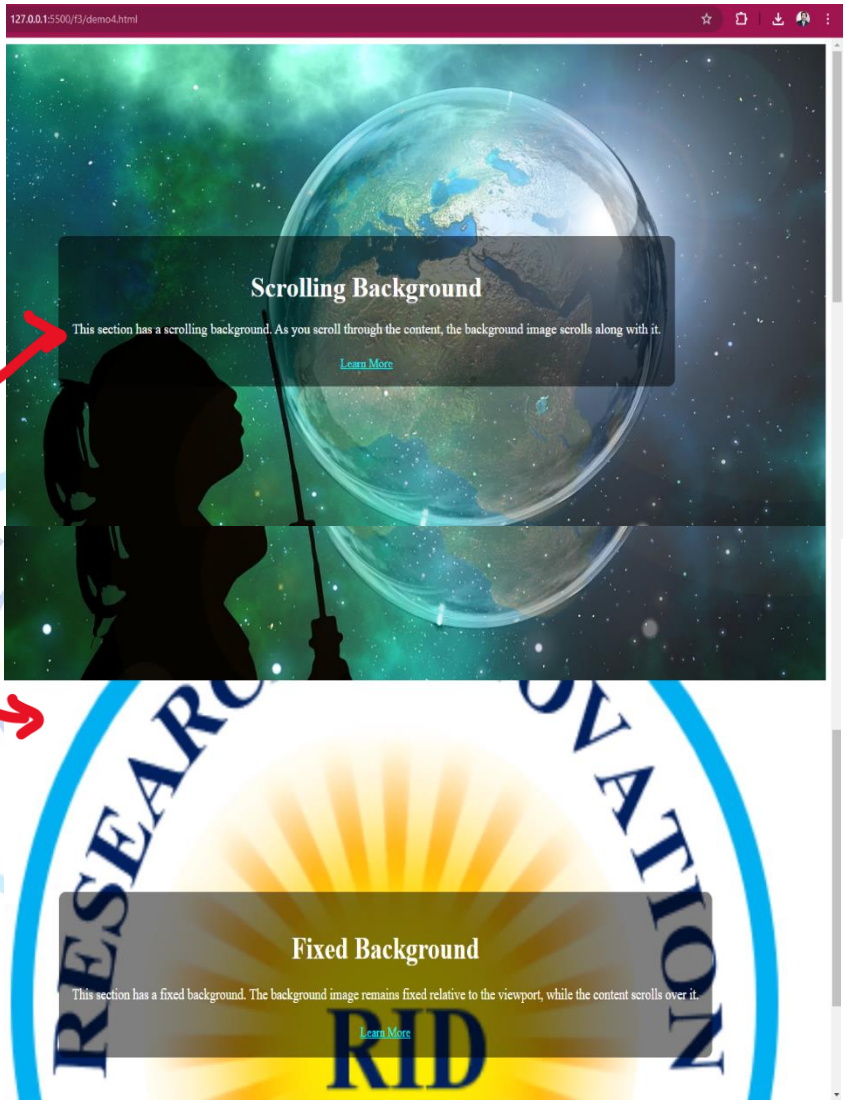
Example-2:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Background Attachment Example</title>
<style>
  .container {
    width: 100%;
    height: 100vh;
    background-image: url('earth.jpg');
    background-repeat: no-repeat;
    background-size: cover;
  }
  .scroll {
    background-attachment: scroll;
  }
  .fixed {
    background-attachment: fixed;
  }
</style>
</head> <body>
  <div class="container scroll">
    <div class="text">Background Attachment: Sc
  </div>
  <div class="container fixed">
    <div class="text">Background Attachment: Fixed</div>
  </div>
</body> </html>
```



Example-3:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Background Attachment Example</title>
<style>
 .container {
      width: 100%;
      height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
      color: white;
      text-align: center;}
   .scroll {
background-image: url('earth.jpg');
background-repeat: no-repeat;
background-size: cover;
background-attachment: scroll;}
   .fixed {
background-image: url('logo.PNG');
background-repeat: no-repeat;
background-size: cover;
background-attachment: fixed;}
   .text-container {
 background-color: rgba(0, 0, 0, 0.5);
 padding: 20px;
 border-radius: 10px;}
   h1 {
      font-size: 36px;
      margin-bottom: 10px; }
   p {
      font-size: 18px;
      line-height: 1.6;
      margin-bottom: 20px; }
</style></head> <body>
 <div class="container scroll">
 <div class="text-container">
    <h1>Scrolling Background</h1>
    <p>This section has a scrolling background. </p>
    <a style="color: aqua;" href="#" class="btn">Learn More</a>
      </div></div>
   <div class="container fixed">
      <div class="text-container">
         <h1>Fixed Background</h1>
         <p>This section has a fixed background.p>
         <a style="color: aqua;" href="#" class="btn">Learn More</a>
      </div></div>
```

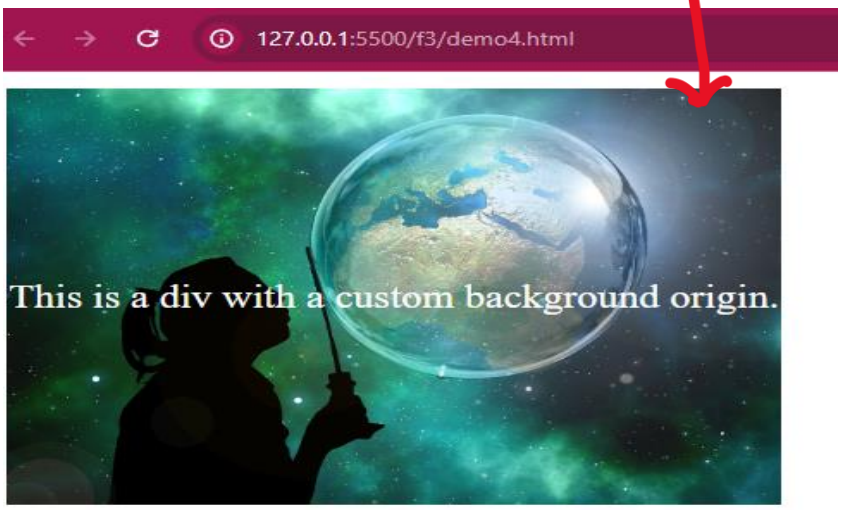


7.background-origin:

- background-origin property in CSS specifies the origin of a background image or gradient. It determines where the background positioning area originates from within an element's padding, border, or content box. The property accepts the following values:
- **padding-box:** The background image or gradient starts from the inner edge of the element's padding box.
- **border-box:** The background image or gradient starts from the inner edge of the element's border box.
- **content-box:** The background image or gradient starts from the inner edge of the element's content box.

Example-1:

```
<!DOCTYPE html>
<html>
<head>
 <title>Background Origin Example</title>
 <style>
  .origin-box {
   width: 300px;
   height: 200px;
   background-image: url('earth.jpg');
   background-repeat: no-repeat;
   background-size: cover;
   background-origin: content-box;
   color: #fff;
   text-align: center;
   line-height: 200px;
  }
 </style>
</head>
<body>
 <div class="origin-box">
  This is a div with a custom background origin.
 </div>
</body>
</html>
```



Example-2:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Background Origin Example</title>
<style>
  .container {
    width: 300px;
    height: 200px;
    border: 2px solid #000;
    padding: 20px;
    margin: 20px;}
  .padding-box {
    background-image: url('earth.jpg');
    background-repeat: no-repeat;
    background-size: cover;
    background-origin: padding-box;}
  .border-box {
    background-image: url('logo.PNG');
    background-repeat: no-repeat;
    background-size: cover;
    background-origin: border-box;}
  .content-box {
    background-image: url('myimg.jpeg');
    background-repeat: no-repeat;
    background-size: cover;
    background-origin: content-box;}
  .text {
    font-family: Arial, sans-serif;
    font-size: 18px;
    text-align: center;
    margin-top: 10px;
 font-weight: bold;
        background-color: yellow;
    color: red;
}
</style>
</head>
<body>
  <div class="container padding-box">
    <div class="text">Background Origin: padding-box</div>
  </div>
  <div class="container border-box">
    <div class="text">Background Origin: border-box</div>
  </div>
  <div class="container content-box">
    <div class="text">Background Origin: content-box</div>
  </div>
</body>
</html>
```

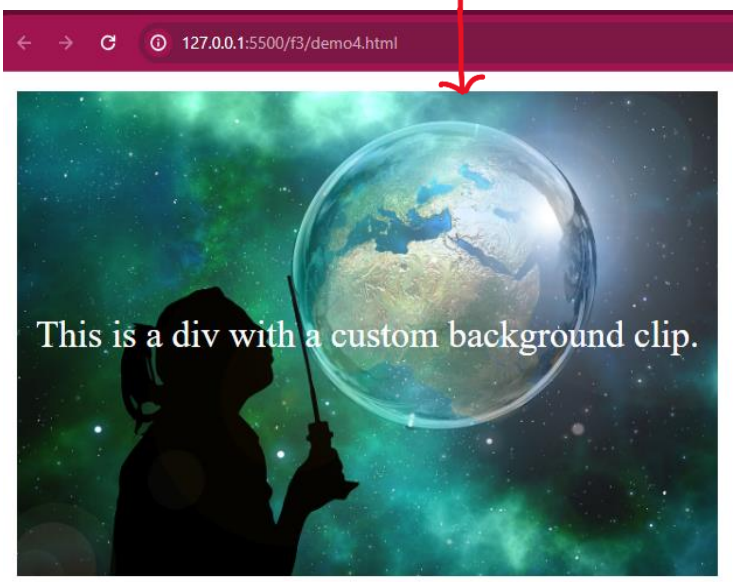
8. background-clip:

- background-clip property in CSS specifies how far the background color extends within an element. It determines whether the background is clipped to the content box, padding box, border box, or text.
- The property accepts the following values:

- **border-box:** The background extends to the outer edge of the border.
- **padding-box:** The background extends to the outer edge of the padding.
- **content-box:** The background extends to the outer edge of the content.
- **text:** The background extends only to the text (and not to the border, padding, or margin).

Example-1:

```
<!DOCTYPE html>
<html>
<head>
 <title>Background Clip Example</title>
 <style>
  .clipped-box {
   width: 300px;
   height: 200px;
   background-image: url('earth.jpg');
   background-repeat: no-repeat;
   background-size: cover;
   background-clip: content-box;
   color: #fff;
   text-align: center;
   line-height: 200px;
  }
 </style>
</head>
<body>
 <div class="clipped-box">
  This is a div with a custom background clip.
 </div>
</body>
```



Example-2:

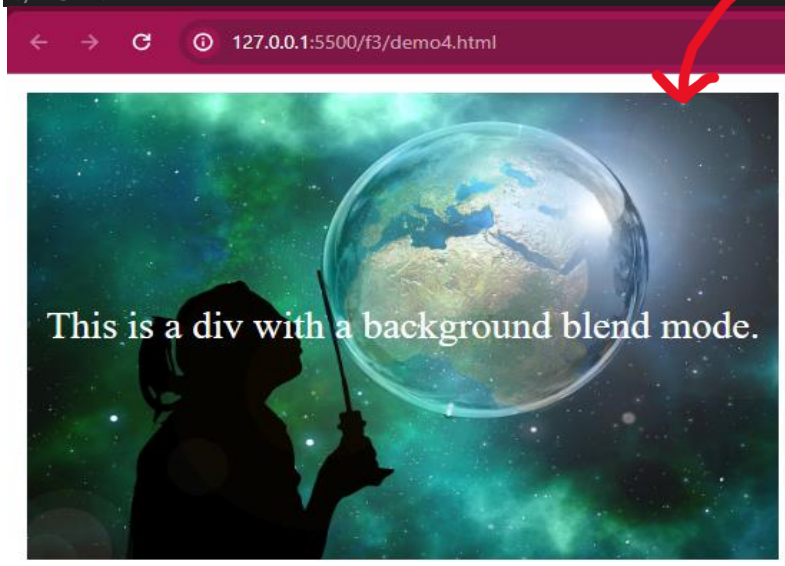
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Background Clip Example</title>
<style>
   .container {
      width: 300px;
      height: 200px;
      border: 5px solid black;
      padding: 20px;
      margin: 20px;
      font-size: 24px;
      text-align: center;
   }
.border-box {
background: linear-gradient(to bottom, #ff5733, #ffc300);
color: white;
background-clip: border-box;
   }
.padding-box {
background: linear-gradient(to bottom, #33ff57, #33f5ff);
color: black;
background-clip: padding-box;
   }
.content-box {
background: linear-gradient(to bottom, #5733ff, #ff33d6);
color: white;
background-clip: content-box;
   }
.text {
background: linear-gradient(to bottom, #ff3333, #ff9933);
-webkit-background-clip: text;
 color: black;}
</style>
</head> <body>
   <div class="container border-box">
      <div class="text">Background Clip: border-box</div>
   </div>
   <div class="container padding-box">
      <div class="text">Background Clip: padding-box</div>
   </div>
   <div class="container content-box">
      <div class="text">Background Clip: content-box</div>
   </div>
   <div class="container">
      <div class="text">Background Clip: text</div>
   </div>
</body></html>
```

9.background-blend-mode:

- background-blend-mode property in CSS allows you to specify how the background image or color of an element should blend with the content or other background images. It applies a blending mode to the background layers of an element.
- The property accepts various blending mode values, each determining how the colors of the background layers should blend. common blending modes include normal, multiply, screen, overlay, darken, lighten, color-dodge, color-burn, difference, exclusion, and more.

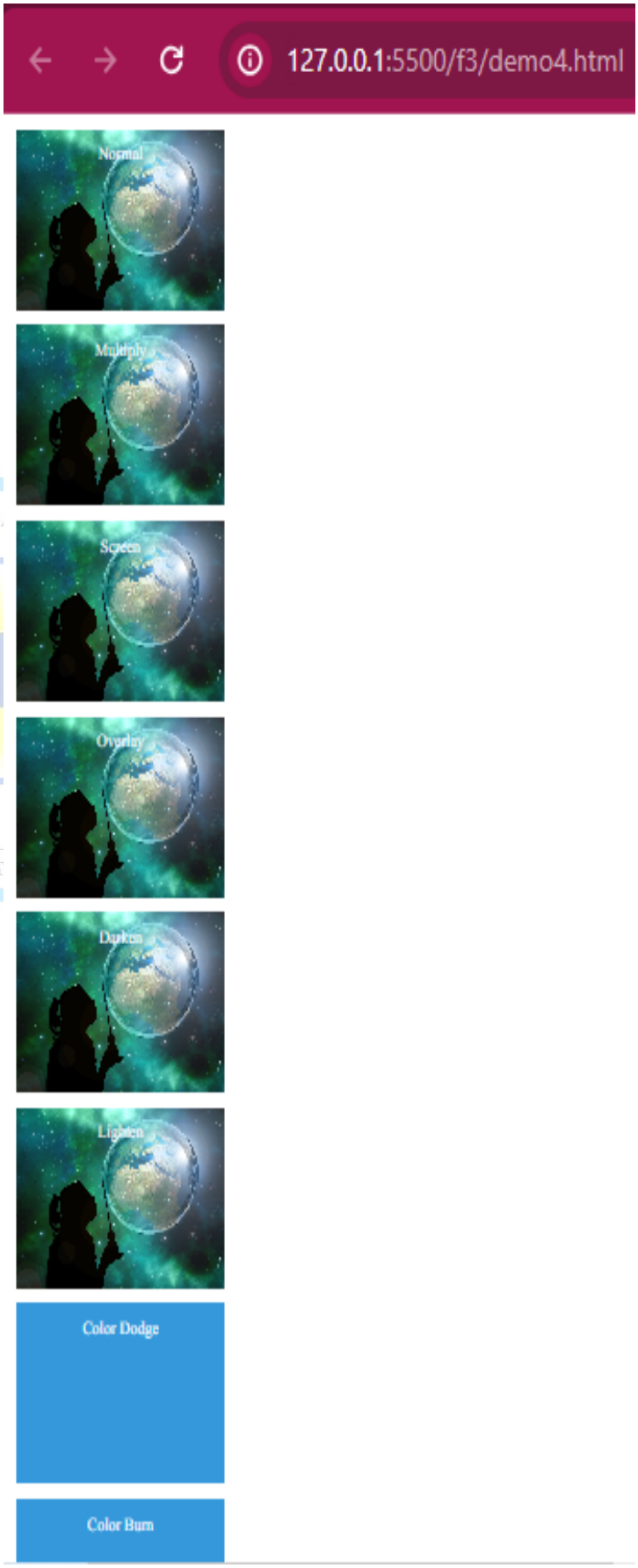
Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Background Blend Mode Example</title>
  <style>
    .blend-box {
      width: 300px;
      height: 200px;
      background-image: url('earth.jpg');
      background-repeat: no-repeat;
      background-size: cover;
      background-blend-mode: multiply;
      color: #fff;
      text-align: center;
      line-height: 200px;
    }
  </style>
</head>
<body>
  <div class="blend-box">
    This is a div with a background blend mode.
  </div>
</body>
</html>
```



Example-2:

```
<!DOCTYPE html>
<html lang="en"> <head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Blend Mode Example</title>
<style>
   .c { width: 300px;
      height: 200px;
      margin: 20px;
      padding: 20px;
      font-size: 24px;
      text-align: center;
   }
   .bi { background-image: url('earth.jpg');
        background-size: cover;
         color: white;}
   .bc { background-color: #3498db;
        color: white;}
   .n {
      background-blend-mode: normal;}
   .m {
      background-blend-mode: multiply;}
   .s { background-blend-mode: screen;}
   .o {
      background-blend-mode: overlay;}
   .d {
      background-blend-mode: darken;}
   .l { background-blend-mode: lighten;}
   .cd {
      background-blend-mode: color-dodge;}
   .cb {
      background-blend-mode: color-burn;}
   .diff {
      background-blend-mode: difference;}
   .ex {
      background-blend-mode: exclusion;}
</style>
</head<body>
   <div class="c bi n">Normal</div>
   <div class="c bi m">Multiply</div>
   <div class="c bi s">Screen</div>
   <div class="c bi o">Overlay</div>
   <div class="c bi d">Darken</div>
   <div class="c bi l">Lighten</div>
   <div class="c bc cd">Color Dodge</div>
   <div class="c bc cb">Color Burn</div>
   <div class="c bc diff">Difference</div>
   <div class="c bc ex">Exclusion</div>
</body> </html>
```



10.-webkit-background-clip:

- webkit-background-clip: text is a CSS property that allows you to control the clipping of background images or colors to the text of an element. When applied, it makes the background of the text transparent, allowing the background of the parent element to show through the text itself.

❖ **Why Use -webkit-background-clip: text?**

- **Visual Effects**, **Typography Enhancement** and **Flexibility**:

❖ **How to Use -webkit-background-clip: text**

1. **Basic Syntax**: this is typically used alongside background-image, background-color, or background-gradient and requires the text color to be transparent for the effect to be visible.

Example:

```
.text {
  color: transparent;  /* Hide the text color */
  background-image: linear-gradient(to right, #FF5733, #33FF57); /* Apply a gradient */
  -webkit-background-clip: text; /* Clip the background to the text */
  background-clip: text;  /* Standard property (not widely supported) */
}
```

Example: Apply background-image or background-color in each letter in a Word like RID

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Letters with Background Images</title>
  <style>
    h1, span { display: inline;
      background-image: url(t6.jpg);
      background-size: cover;
      -webkit-background-clip: text;
      color: transparent;
      font-size: 70px;
      font-family: 'Times New Roman', Times, serif;
      font-weight: bold;
      margin: 0; /* Remove default margin */
    }
    .spacer {  margin-left: 30px; /* Adjust the space between RID and BHARAT */}
  </style></head> <body>
  <h1>R</h1>
  <h1>I</h1>
  <h1>D</h1>
  <span class="spacer"></span> <!-- Spacer between RID and BHARAT -->
  <span>B</span>
  <span>H</span>
  <span>A</span>
  <span>R</span>
  <span>A</span>
  <span>T</span>
</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:

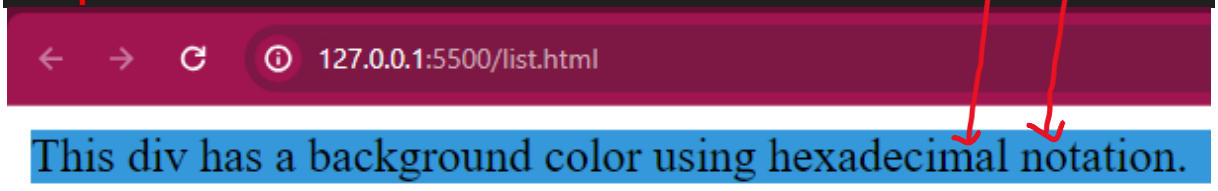
```
<!DOCTYPE html>
<html>
<head>
  <title>Named Color Example</title>
  <style>
    .c1 {
      color: red; /* Named color */
    }
  </style>
</head>
<body>
  <p class="c1">This is red text using a named color.</p>
</body>
</html>
```

2. Hexadecimal Notation:

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Hexadecimal Color Example</title>
  <style>
    .c1 {
      background-color: #3498db; /* Hexadecimal color */
      color: black;
    }
  </style>
</head>
<body>
  <div class="c1">
    This div has a background color using hexadecimal notation.
  </div>
</body>
</html>
```

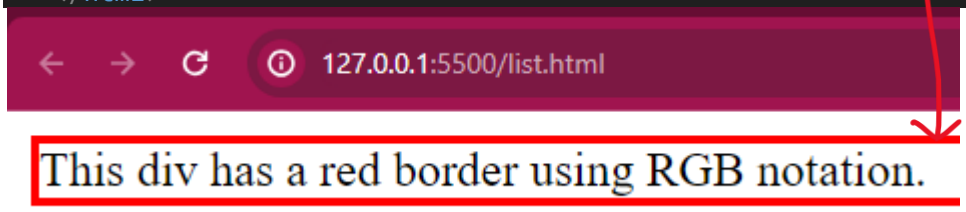
Output:



3. RGB Notation:

Example:

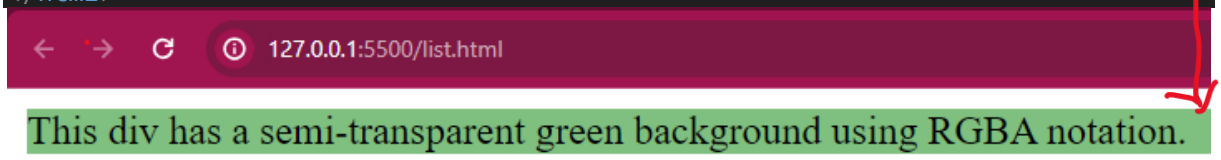
```
<!DOCTYPE html>
  <html>
  <head>
    <title>RGB Color Example</title>
    <style>
      .c1 {
        border: 3px solid rgb(255, 0, 0); /* RGB color */
      }
    </style>
  </head>
  <body>
    <div class="c1">
      This div has a red border using RGB notation.
    </div>
  </body>
  </html>
```



4. RGBA Notation:

Example:

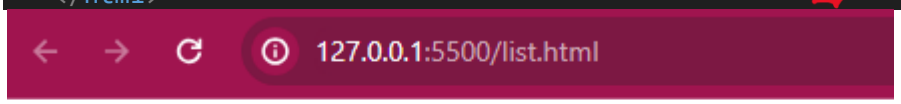
```
<!DOCTYPE html>
<html>
<head>
  <title>RGBA Color Example</title>
  <style>
    .c1 {
      background-color: rgba(0, 128, 0, 0.5); /* RGBA color with transparency
    }
  </style>
</head>
<body>
  <div class="c1">
    This div has a semi-transparent green background using RGBA notation.
  </div>
</body>
</html>
```



5. HSL Notation:

Example:

```
<!DOCTYPE html>
  <html>
  <head>
    <title>HSL Color Example</title>
    <style>
      .c1 {
        color: hsl(210, 100%, 50%); /* HSL color */
      }
    </style>
  </head>
  <body>
    <p class="c1">This is text with an HSL-based color.</p>
  </body>
  </html>
```

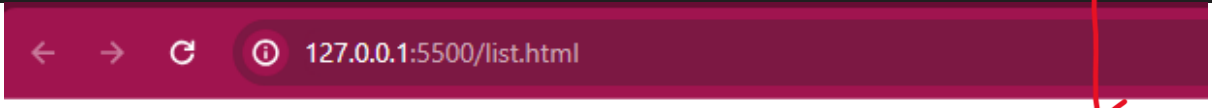


This is text with an HSL-based color.

6. HSLA Notation:

Example:

```
<!DOCTYPE html>
  <html>
  <head>
    <title>HSLA Color Example</title>
    <style>
      .c1 {
        border: 3px solid hsla(120, 100%, 50%, 0.7);
      }
    </style>
  </head>
  <body>
    <div class="c1">
      This div has a semi-transparent border using HSLA notation.
    </div>
  </body>
  </html>
```



This div has a semi-transparent border using HSLA notation.

❖Examples of different color usages in CSS:

1. Text Color:

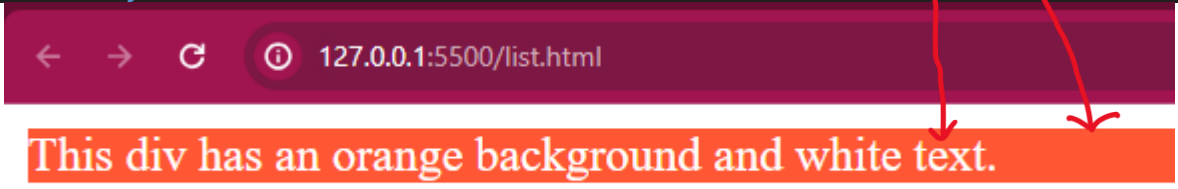
Example:

```
<!DOCTYPE html>
  <html>
  <head>
    <title>Text Color Example</title>
    <style>
      .c1 {
        color: blue; /* Text color */
      }
    </style>
  </head>
  <body>
    <p class="c1">This text has a blue color.</p>
  </body></html>
```

2. Background Color:

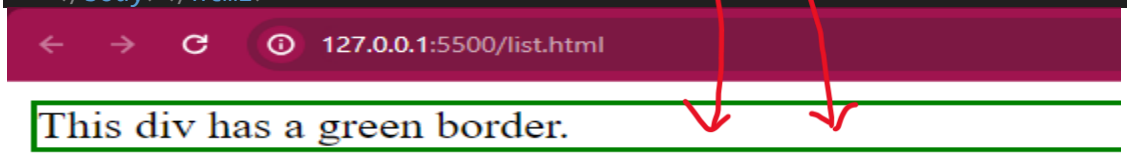
Example:

```
<!DOCTYPE html>
  <html>
  <head>
    <title>Background Color Example</title>
    <style>
      .c1 {
        background-color: #FF5733;
        color: white;
      }
    </style>
  </head>
  <body>
    <div class="c1">
      This div has an orange background and white text.
    </div>
  </body> </html>
```



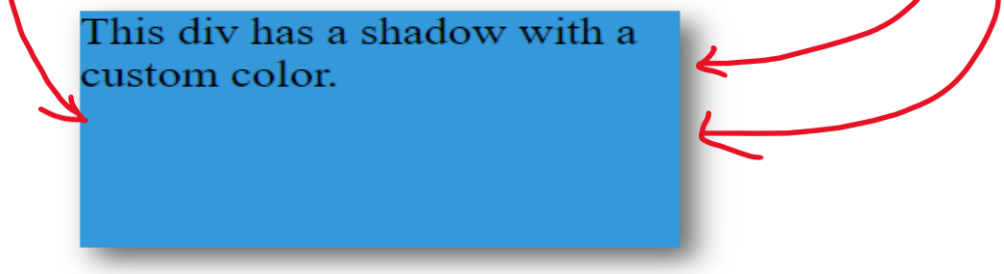
3. Border Color:

```
<!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:

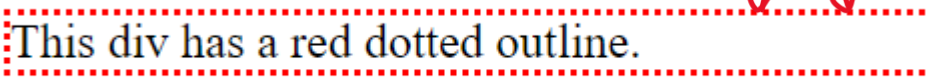
```
<!DOCTYPE html>
  <html>
  <head>
    <title>Box Shadow Color Example</title>
    <style>
      .shadowed-element {
        width: 200px;
        height: 100px;
        background-color: #3498db;
        box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.6); /* Box shadow color */
      }
    </style>
  </head>
  <body>
    <div class="shadowed-element">
      This div has a shadow with a custom color.
    </div>
  </body> </html>
```



5. Outline Color:

Example:

```
<!DOCTYPE html>
  <html>
  <head>
    <title>Outline Color Example</title>
    <style>
      .c1 {
        outline: 2px dotted red; /* Outline color */
      }
    </style>
  </head>
  <body>
    <div class="c1">
      <p>This div has a red dotted outline.</p>
    </div>
  </body>
  </html>
```



6. Text Shadow Color:

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Text Shadow Color Example</title>
  <style>
    #c1 {
      text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5); /* Text shadow color */
    }
  </style>
</head>
<body>
  <h2 id="c1">This heading has a shadow with a custom color.</h2>
</body>
</html>
```

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>
    .C1 {
      background-color: blue;
      color: white;
      opacity: 0.5; /* 50% opacity */
    }
  </style>
</head>
<body>
  <div class="C1">
    This is a div with reduced opacity.
  </div>
</body></html>
```

This is a div with reduced opacity.

➢ `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>
    .c1 {
      background-color: rgba(255, 0, 0, 0.5); /* Red color with 50% opacity */
      color: white;
    }
  </style>
</head>
<body>
  <div class="c1">
    This is a div with a semi-transparent red background.
  </div>
</body></html>
```

This is a div with a semi-transparent red background.

- 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.

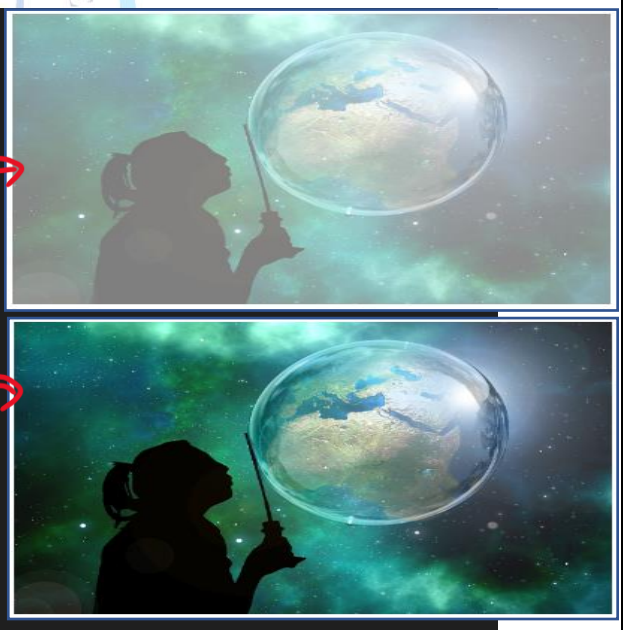
```
<!DOCTYPE html>
<html>
<head>
  <title>Background and Text Transparency Example</title>
  <style>
    .transparent-box {
      background-color: rgba(0, 128, 0, 0.7);
      color: rgba(0, 0, 0, 0.5); /* Black text with 50% opacity */
    }
  </style>
</head>
<body>
  <div class="transparent-box">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

This is a div with a semi-transparent background and text.

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>
    .c1 {
        width: 200px;
        height: 150px;
      opacity: 0.5;
    }
    .c2{
        width: 200px;
        height: 150px;
    }
  </style>
</head>
<body>
  <img class="c1" src="earth.jpg" alt="Transparent Image" />
  <img class="c2" src="earth.jpg" 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>
```

This text is in crimson color.

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>
```

This div has a gold background color.

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>
    .c1 {
      border: 2px solid seagreen; /* Named color */
    }
  </style>
</head>
<body>
  <div class="c1">
    This div has a seagreen border.
  </div>
</body></html>
```

This div has a seagreen border.

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>
```

This div has a box shadow with a darkgray color.

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>
```

This div has a royalblue dashed outline.

❖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
```

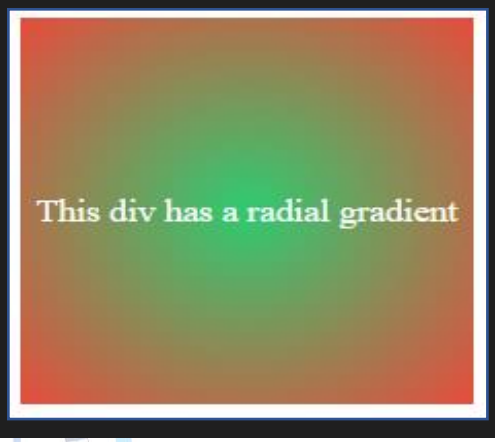
This is a div with a linear gradient

➢ 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>.
```



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>
```

❖ 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>
```

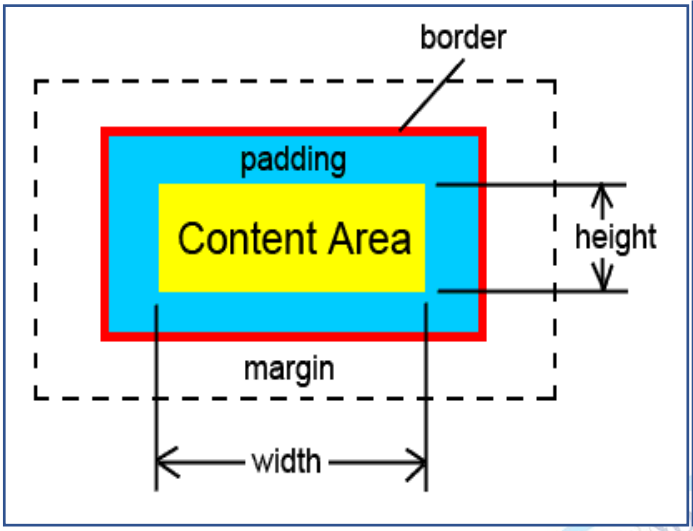
4. hsla() Function:

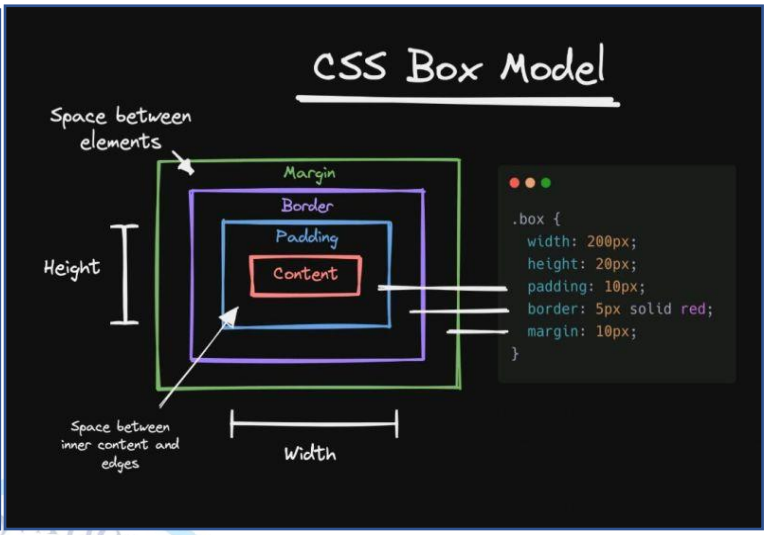
- The `hsla()` function allows you to specify colors with an additional alpha channel for transparency.

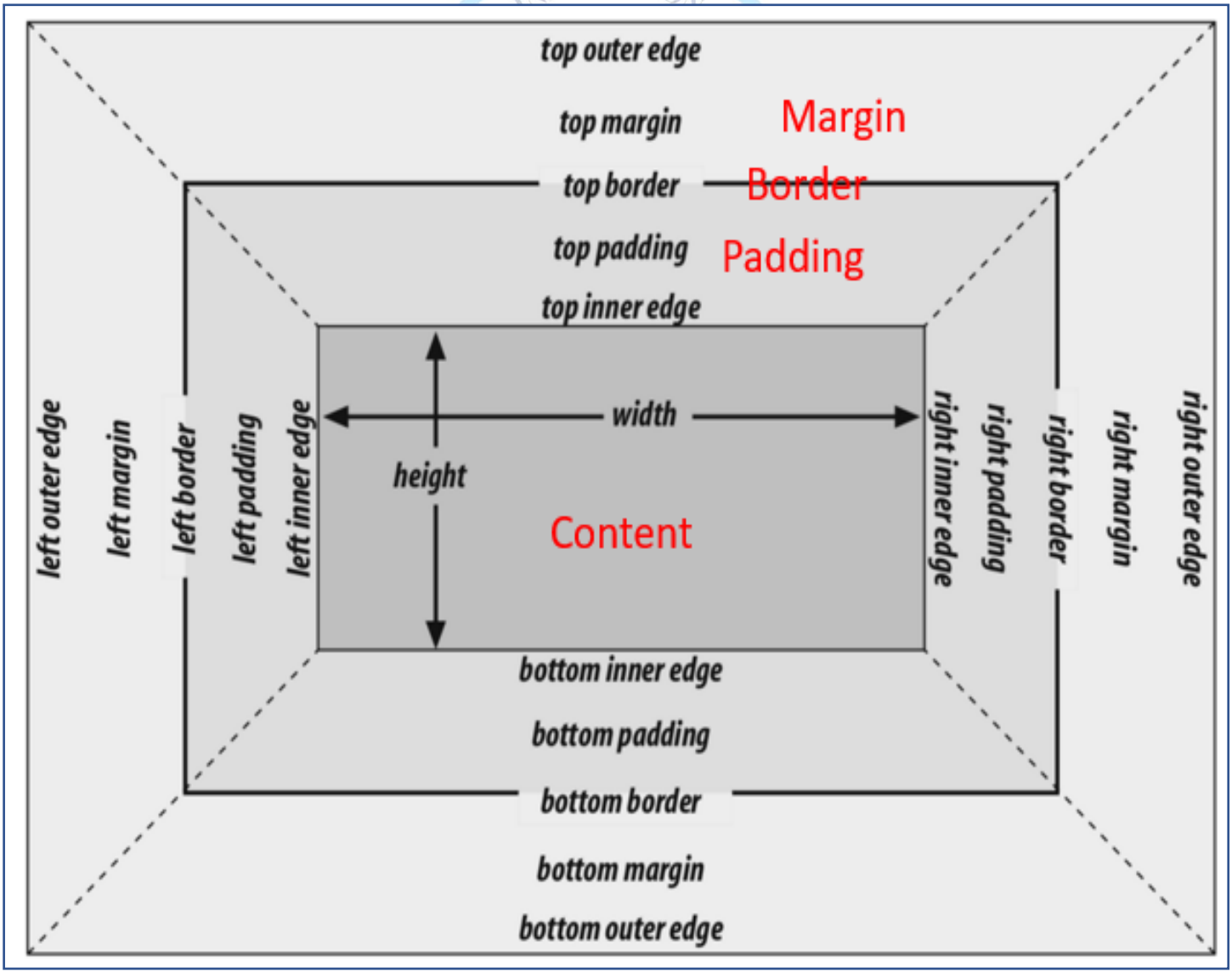
```
<!DOCTYPE html>
<html>
<head>
 <title>HSLA Color Function Example</title>
 <style>
  .transparent-text {
   color: hsla(45, 100%, 50%, 0.6); /* Semi-transparent orange using hsla() function */
   text-align: center;
   padding: 20px;
  }
 </style>
</head>
<body>
 <p class="transparent-text">This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







❖ 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.

❖ Padding:

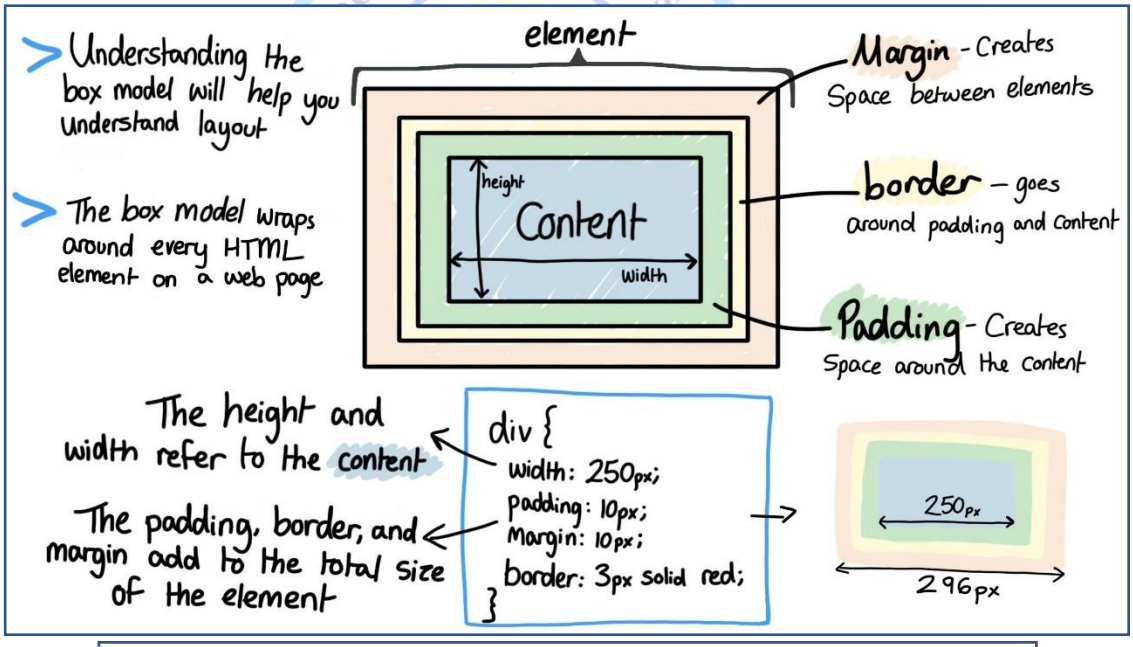
- Padding is the space between the content area and the element's border.
- It can be adjusted using the padding property.
- Padding is used to create space between the content and the element's border, improving readability and aesthetics.

❖ 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.

❖ 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.



PADDING PROPERTIES

- Padding is the space between the content area and the element's border.
- It can be adjusted using the padding property.
- **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.

❖ How to apply padding?

- Padding is short hand properties.
- **Vh, %, fr, em:** Relative units, as they are relative to some other value.
- **Px:** Absolute unit, as it represents a fixed measure (not relative to other values).

Example:

- ✓ **Padding 10px;**
- ✓ **Padding 10px, 20px;**
- ✓ **Padding 10px, 20px, 30px;**
- ✓ **Padding 10px, 20px, 30px, 40px;**

Explanation:

1. **Padding 10px;→**: Sets equal padding of 10 pixels on all four sides.
2. **Padding 10px, 20px;→**: Sets 10 pixels of padding on the top and bottom, and 20 pixels on the right and left sides.
3. **Padding 10px, 20px, 30px;→**: Sets 10 pixels of padding on the top, 20 pixels on the right and left sides, and 30 pixels on the bottom.
4. **Padding 10px, 20px, 30px, 40px;→**: Sets 10 pixels of padding on the top, 20 pixels on the right, 30 pixels on the bottom, and 40 pixels on the left side.

Syntax:

- **padding-top:** value;
- **padding-right:** value;
- **padding-bottom:** value;
- **padding-left:** value;

1. **padding-top:** Sets the padding for the top side of the element.
2. **padding-right:** Sets the padding for the right side of the element.
3. **padding-bottom:** Sets the padding for the bottom side of the element.
4. **padding-left:** Sets the padding for the left side of the element.

Example:

padding-top: 10px;
padding-right: 20px;
padding-bottom: 15px;
padding-left: 30px;

1. If one value is provided, it sets the padding for all four sides equally:
   - **padding: value;**
2. If two values are provided, the first value sets the padding for the top and bottom, and the second value sets the padding for the right and left sides:
   - **padding: value1 value2;**
3. If three values are provided, the first value sets the padding for the top, the second value sets the padding for the right and left sides, and the third value sets the padding for the bottom:
   - **padding: value1 value2 value3;**
4. If four values are provided, they represent the padding for the top, right, bottom, and left sides respectively:
   - **padding: value1 value2 value3 value4;**

➢ In each case, value can be expressed using different units like pixels (px), percentages (%), ems (em), or other supported units. You can also use negative values if needed. For example:

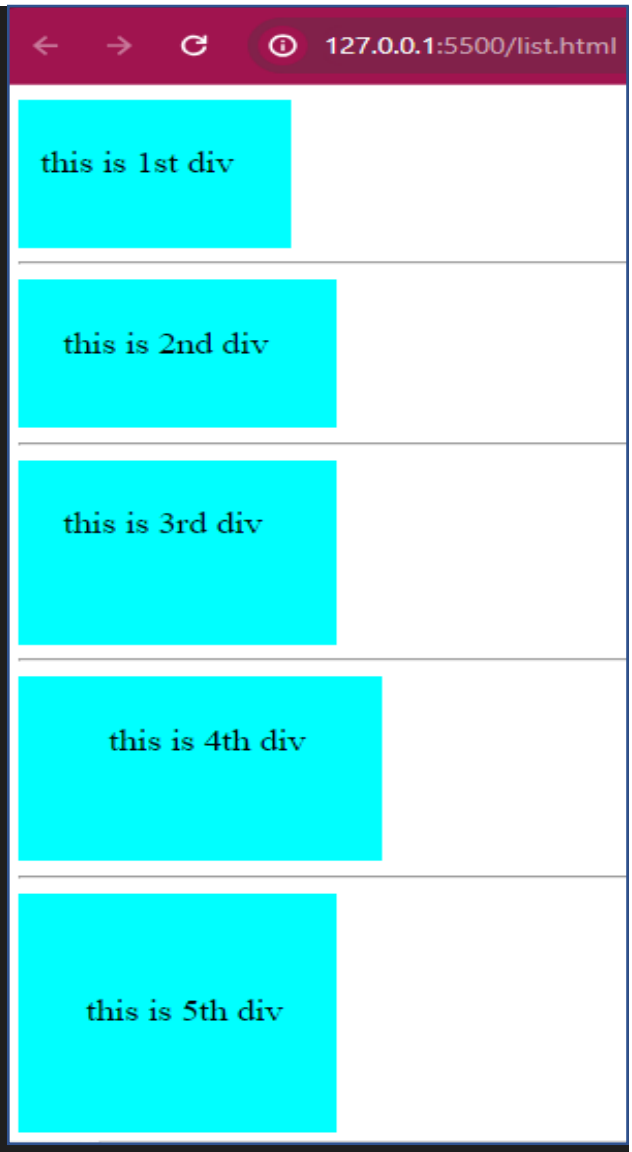
Example:

- **padding: 10px;** /* Equal padding on all sides */
- **padding: 10px 20px;** /* Vertical padding 10px, horizontal padding 20px */
- **padding: 10px 20px 30px;** /* Top padding 10px, horizontal padding 20px, bottom padding 30px */
- **padding: 10px 20px 30px 40px;** /* Top padding 10px, right padding 20px, bottom padd

Example:

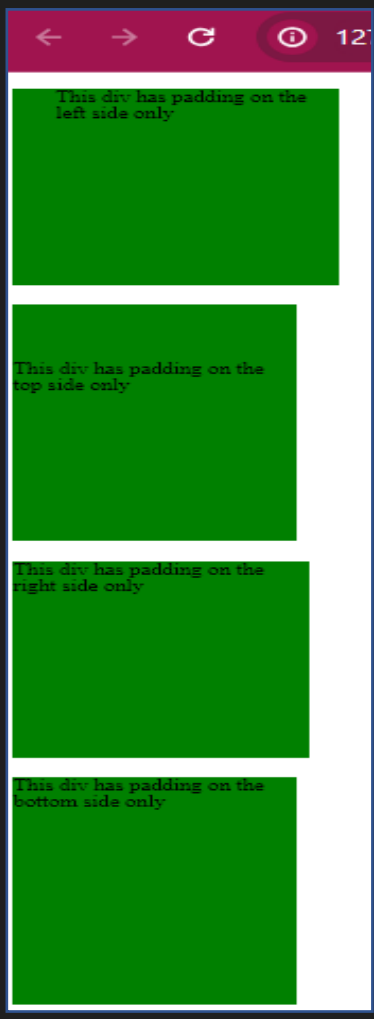
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    div{
      width: 100px;
      height: 20vh;
      background-color: aqua;
```

```
    }
    .c1{
      padding: 50px;
    }
    .c2{
      padding: 10px 20px;
    }
    .c3{
      padding: 10px 20px 30px;
    }
    .c4{
      padding: 10px 20px 30px 40px;
    }
    .c5{
      padding-left: 30px;
      padding-top: 40px;
      padding-right: 10px;
      padding-bottom: 30px;
    }
  </style>
</head>
<body>
  <div class="c1">
    <p>this is 1st div </p>
  </div>
  <hr>
  <div class="c2">
    <p>this is 2nd div </p>
  </div>
  <hr>
  <div class="c3">
    <p>this is 3rd div </p>
  </div>
  <hr>
  <div class="c4">
    <p>this is 4th div </p>
  </div>
  <hr>
  <div class="c5">
    <p>this is 5th div </p>
  </div>
</body>
</html>
```



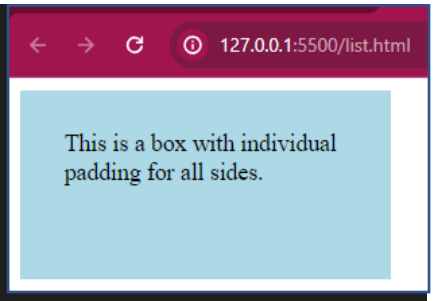
Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Padding Example</title>
  <style>
    .box {
      width: 200px;
      height: 200px;
      background-color: green;
      margin-bottom: 20px;
    }
    .left {
      padding-left: 30px;
    }
    .top {
      padding-top: 40px;
    }
    .right {
      padding-right: 10px;
    }
    .bottom {
      padding-bottom: 30px;
    }
  </style>
</head>
<body>
  <div class="box left">
    <p>This div has padding on the left side only</p>
  </div>
  <div class="box top">
    <p>This div has padding on the top side only</p>
  </div>
  <div class="box right">
    <p>This div has padding on the right side only</p>
  </div>
  <div class="box bottom">
    <p>This div has padding on the bottom side only</p>
  </div>
</body>
</html>
```



❖ 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.
- Borders are often used to visually separate elements or create boundaries around content.
- 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.

❖ Border-syntax:

```
selector {
    border: [border-width] [border-style] [border-color];
} or
border: width style color
```

❖ Example:

```
border: 2px solid black;
```

- **[border-width]:** Specifies the width of the border. It can be defined in pixels, ems, rems.
- **[border-style]:** Specifies the style of the border. It can be solid, dotted, dashed, double, groove, ridge, inset, outset, or none.
- **[border-color]:** Specifies the color of the border. It can be any valid CSS color value.

❖ possible value for style is:

- **none:** No border is displayed.
- **hidden:** Same as none, but for table elements.
- **dotted:** The border is a series of dots.
- **dashed:** The border is a series of short dashes.
- **solid:** The border is a solid line.
- **double:** The border is two solid lines.
- **groove:** The border looks as though it were carved into the page.
- **ridge:** The border looks as though it were coming out of the page.
- **inset:** The border makes the box look as though it were embedded in the page.
- **outset:** border makes the box look as though it were coming out of the page.

❖ If you want to apply separately:

- **border-left:** width style color ;
- **border-top:** width style color ;
- **border-right:** width style color ;
- **border-bottom:** width style color ;

Example:

```
.element {
  border-left: 2px solid red;
  border-top: 1px dashed blue;
  border-right: 3px dotted green;
  border-bottom: 2px double orange; }
```

❖border-width:

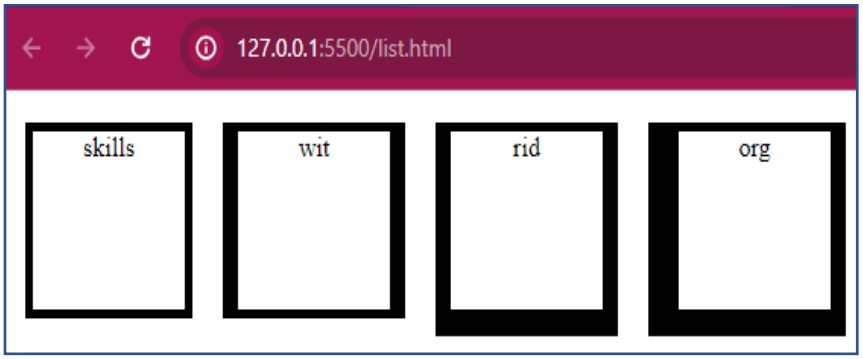
- The border-width property sets the width of the border.
- You can specify width using values like px, em, rem, or keywords like thin, medium, or thick.
- border-style:
- The border-style property sets the style of the border, such as solid, dashed, dotted, etc.
- Common values include solid, dashed,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

Example:

- border width: 10px;
- border width: 10px 20px;
- border width: 10px 20px 30px ;
- border width: 10px 20px 30px 40px ;

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Border Example</title>
<style>
  .box {
    width: 100px;
    height: 100px;
    margin: 10px;
    border: solid black;
    float: left;
  }
  .box1 {
    border-width: 5px; /* Border width: 10px (All sides) */
  }
  .box2 {
    border-width: 5px 10px; /* Border width: 10px top and bottom, 20px left and right */
  }
  .box3 {
    border-width: 5px 10px 15px; /* Border width: 10px top, 20px right, 30px bottom, 40px left */
  }
  .box4 {
    border-width: 5px 10px 15px 20px; /* Border width: 10px top, 20px right, 30px bottom, 40px left */
  }
</style>
</head>
<body>
<div class="box box1">skills</div>
<div class="box box2">wit</div>
<div class="box box3">rid</div>
<div class="box box4">org</div>
</body>
</html>
```

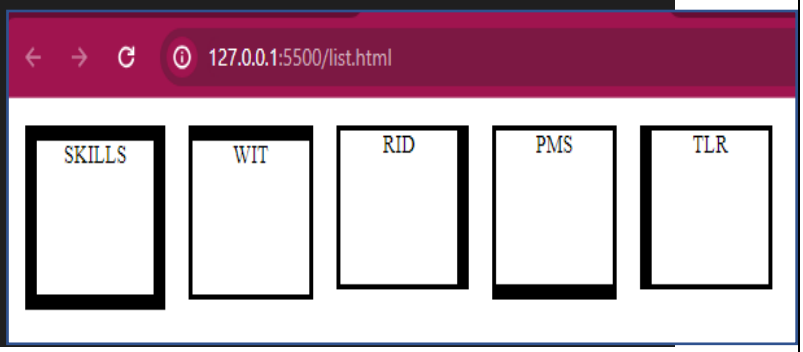


❖ Individual Border Widths:

- **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:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Border Width Example</title>
<style>
    .box {
        width: 100px;
        height: 100px;
        margin: 10px;
        float: left;
        border-style: solid;
        border-color: black;
        text-align: center;
    }
    .box1 {
        border-width: 10px; /* Border width: 10px on all sides */
    }
    .box2 {
        border-top-width: 10px; /* Border width: 10px on top */
    }
    .box3 {
        border-right-width: 10px; /* Border width: 10px on right */
    }
    .box4 {
        border-bottom-width: 10px; /* Border width: 10px on bottom */
    }
    .box5 {
        border-left-width: 10px; /* Border width: 10px on left */
    }
</style>
</head>
<body>
<div class="box box1">SKILLS</div>
<div class="box box2">WIT</div>
<div class="box box3">RID</div>
<div class="box box4">PMS</div>
<div class="box box5">TLR</div>
</body>
</html>
```



❖ Border-style:

➢ possible value for style is:

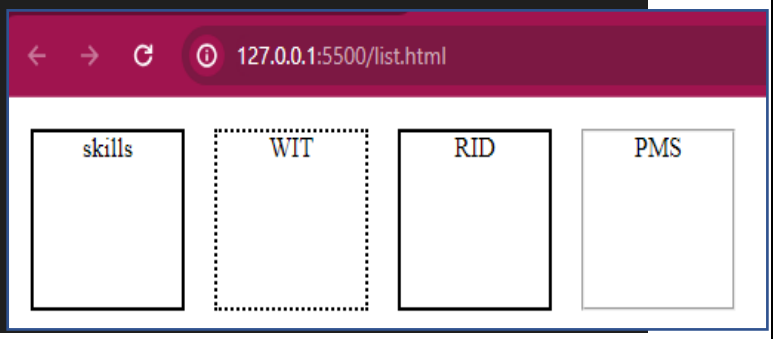
- **none:** No border is displayed.
- **hidden:** Same as none, but for table elements.
- **dotted:** The border is a series of dots.
- **dashed:** The border is a series of short dashes.
- **solid:** The border is a solid line.
- **double:** The border is two solid lines.
- **groove:** The border looks as though it were carved into the page.
- **ridge:** The border looks as though it were coming out of the page.
- **inset:** The border makes the box look as though it were embedded in the page.
- **outset:** border makes the box look as though it were coming out of the page.

Example:

- Border-style: solid;
- Border-style: solid dotted;
- Border-style: solid dotted double ;
- Border-style: solid dotted double groove;

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Border Style Example</title>
<style>
    .box {
        width: 100px;
        height: 100px;
        margin: 10px;
        float: left;
        text-align: center;
    }
       .solid {
        border-style: solid;
        border-width: 2px;
    }
    .dotted {
        border-style: dotted;
        border-width: 2px;
    }
    .double {
        border-style: double;
        border-width: 2px;
    }
    .groove {
        border-style: groove;
        border-width: 2px;
```

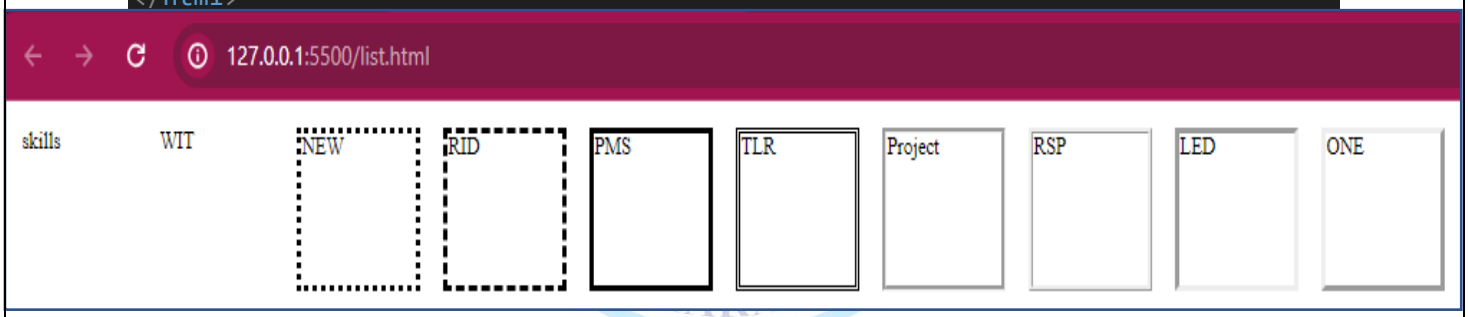
```
    }
</style>
</head>
<body>
<div class="box solid">skills</div>
<div class="box dotted"> WIT </div>
<div class="box double">RID</div>
<div class="box groove">PMS</div>
</body>
</html>
```



Example-2:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Border Style Example</title>
<style>
    .box {
        width: 100px;
        height: 100px;
        margin: 10px;
        float: left;
    }
    .none {
        border-style: none;
    }
    .hidden {
        border-style: hidden;
    }
    .dotted {
        border-style: dotted;
    }
    .dashed {
        border-style: dashed;
    }
    .solid {
        border-style: solid;
    }
    .double {
        border-style: double;
    }
    .groove {
        border-style: groove;
    }
    .ridge {
        border-style: ridge;
```

```
    }
    .inset {
        border-style: inset;
    }
    .outset {
        border-style: outset;
    }
</style>
</head>
<body>
<div class="box none">skills</div>
<div class="box hidden">WIT</div>
<div class="box dotted">NEW</div>
<div class="box dashed">RID</div>
<div class="box solid">PMS</div>
<div class="box double">TLR</div>
<div class="box groove">Project</div>
<div class="box ridge">RSP</div>
<div class="box inset">LED</div>
<div class="box outset">ONE</div>
</body>
</html>
```



❖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-color Property:

➢ Individual Border Colors:

- To set individual border colors for each side of an element, you can use the following properties:

1. **border-top-color:** Sets the color of the top border.
2. **border-right-color:** Sets the color of the right border.
3. **border-bottom-color:** Sets the color of the bottom border.
4. **border-left-color:** Sets the color of the left border.

Example-1:

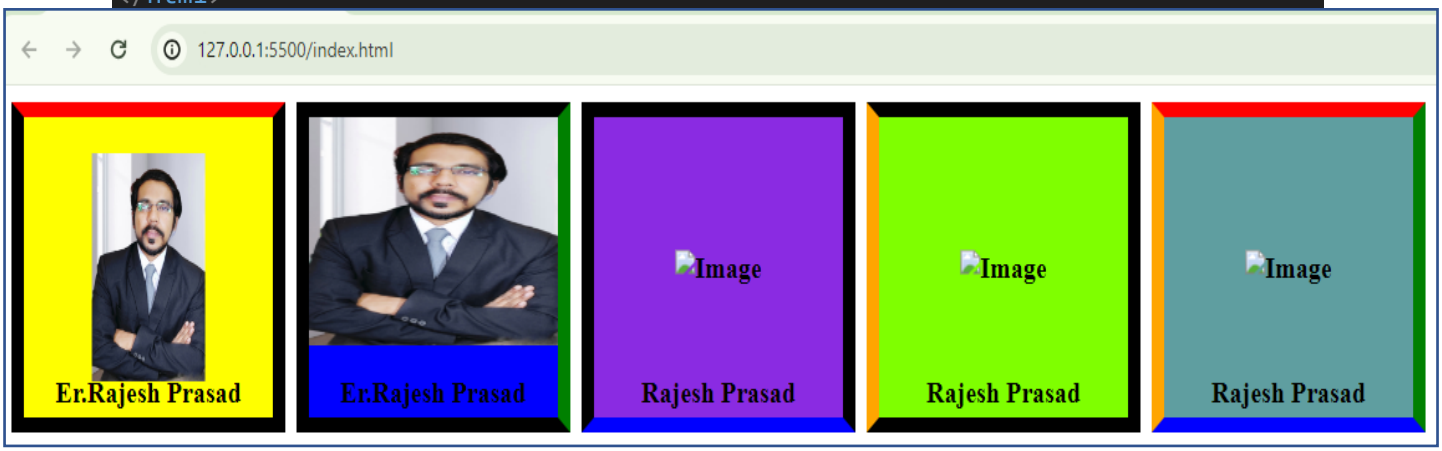
```
.box {
  border-top-color: red;
  border-right-color: green;
  border-bottom-color: blue;
  border-left-color: yellow;
}
Multiple Values:
.box {
  border-color: red;     /* Equal color on all sides */
  /* Or */
  border-color: red green blue yellow; /* Individual color for top, right, bottom, left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

Example-2:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Border Color Example</title>
<style>
    .box {
        width: 200px;
        height: 200px; /* Increased height to accommodate image and text */
        border-style: solid;
        border-width: 10px;
        text-align: center;
        font-size: 20px;
        font-weight: bold;
        display: inline-block;
        margin-right: 5px;
        margin-top: 3px;
        position: relative; /* Added for positioning the image */
    }
```

```
    .box img {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        max-width: 100%;
        max-height: 76%;
    }
    .box #d1 {
        position: absolute;
        top: 38%;
        left: 50%;
        width: 100%;
        height: 100%;
        background-size: cover;
    }
    .box span {
        display: block;
        position: absolute;
        bottom: 5px;
        left: 0;
        right: 0;
    }
    .top {
        border-top-color: red;
        background-color: yellow;
    }
    .right {
        border-right-color: green;
        background-color: blue;
    }
    .bottom {
        border-bottom-color: blue;
        background-color: blueviolet;
    }
    .left {
        border-left-color: orange;
        background-color: chartreuse;
    }
    .all {
        border-top-color: red;
        border-right-color: green;
        border-bottom-color: blue;
        border-left-color: orange;
        background-color: cadetblue;
    }
</style>
</head>
```

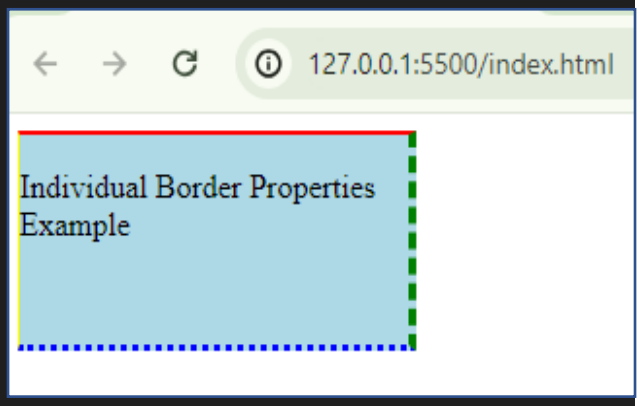
```
<body>
<div class="box top">
    <img src="myimg.jpeg" alt="Image">
    <span>Er.Rajesh Prasad</span>
</div>
<div class="box right">
    <img id="d1" src="myimg.jpeg" alt="Image">
    <span>Er.Rajesh Prasad</span>
</div>
<div class="box bottom">
    <img src="" alt="Image">
    <span>Rajesh Prasad</span>
</div>
<div class="box left">
    <img src="" alt="Image">
    <span>Rajesh Prasad</span>
</div>
<div class="box all">
    <img src="" alt="Image">
    <span>Rajesh Prasad</span>
</div>
</body>
</html>
```



Example-3:

```
<!DOCTYPE html>
<html>
<head>
  <title>Individual Border Properties Example</title>
  <style>
    .box {
      width: 200px;
      height: 100px;
      background-color: lightblue;
      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-4:

```
<!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.

❖ 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.

1. 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 */
}
```

2. 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 */
}
```

3. 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 */
}
```

4. 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:

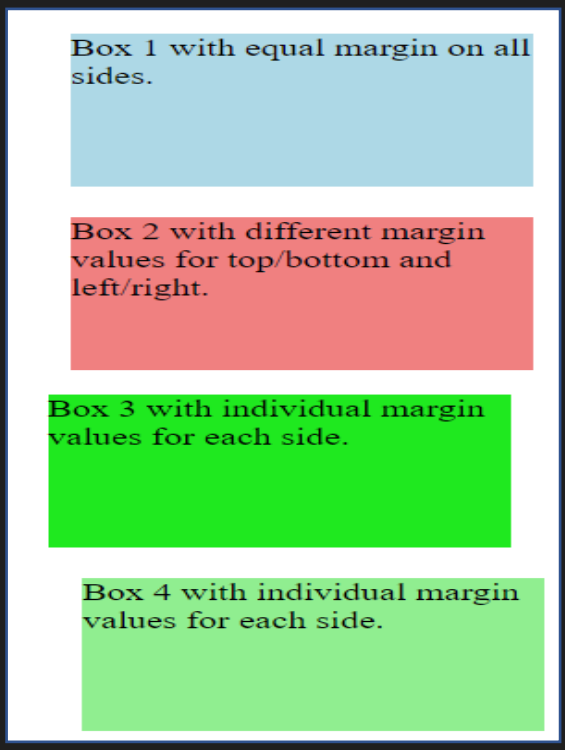
- The auto value can be used to horizontally center an element within its container.
- The margin property is not inherited, meaning that it doesn't inherit margin values from parent elements.

❖ 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: 20px;
    }
    .box2 {
      width: 200px;
      height: 100px;
      background-color: lightcoral;
      margin: 10px 20px;
    }
    .box3 {
      width: 200px;
      height: 100px;
      background-color: rgb(31, 233, 31);
      margin: 5px 10px 20px;
    }
    .box4 {
      width: 200px;
      height: 100px;
      background-color: lightgreen;
      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>
  <div class="box4">
    <p>Box 4 with individual margin values for each side.</p>
  </div>
</body> </html>
```



❖ How the CSS Box Model is calculated:

1. 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)

2. 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)

3. 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.

Explanation:

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

1. Width:

- 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.

2. Left Padding and Right Padding:

- 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.

3. Left Border and Right Border:

- 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.

4. Left Margin and Right Margin:

- 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.

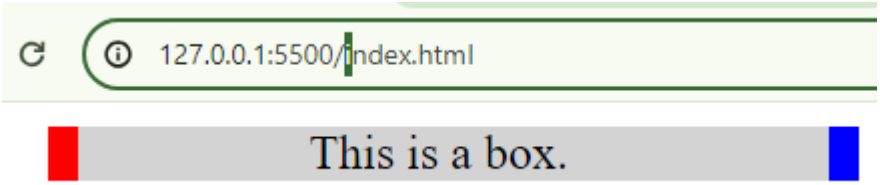
Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Box Model Example</title>
  <style>
    .box {
      width: 200px;
```

```
      padding-left: 20px;
      padding-right: 30px;
      border-left: 10px solid red;
      border-right: 10px solid blue;
      margin-left: 60px;
      margin-right: 15px;
      background-color: lightgray;
      text-align: center;
    }
  </style>
</head>
<body>
  <div class="box">This is a box.</div>
</body>
</html>
```

Total Width = 200px + 20px + 30px + 10px + 10px + 60px + 15px

Total Width = 345px



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:

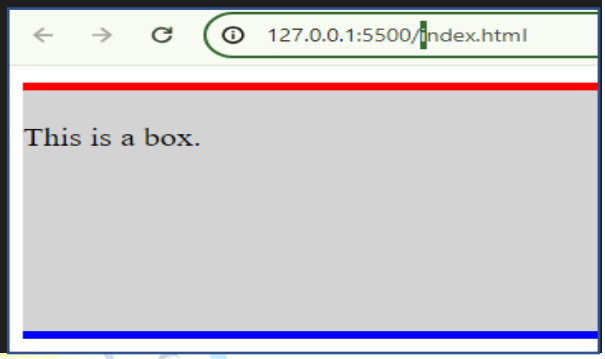
- The `Top Border` and `Bottom Border` represent the width of the element's border on the top and bottom sides, respectively.
- Borders can be styled using CSS properties like `border-width`, `border-style`, and `border-color`.

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;
      padding-top: 20px;
      padding-bottom: 30px;
      border-top: 5px solid red;
      border-bottom: 5px solid blue;
      margin-top: 10px;
      margin-bottom: 15px;
      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;
}
```

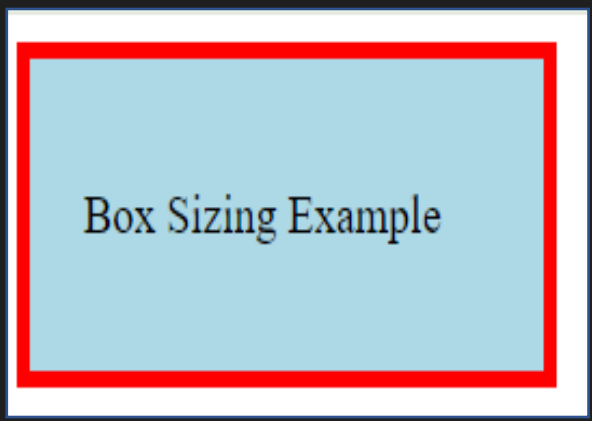
2. border-box:

- 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;
}
```

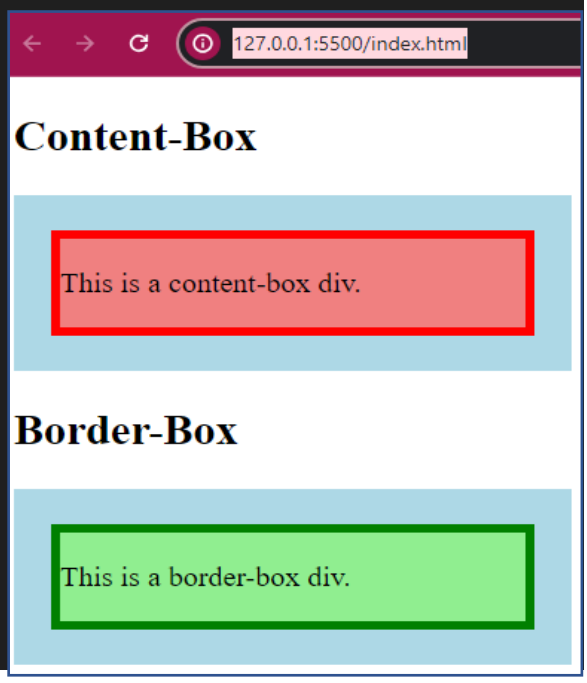
Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>Box Sizing Example</title>
  <style>
    .box {
      width: 200px;
      height: 100px;
      background-color: lightblue;
      box-sizing: 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>
    html {
      box-sizing: border-box;
    }
    *, *::before, *::after {
      box-sizing: inherit;
    }

    .container {
      width: 300px;
      padding: 20px;
      background-color: lightblue;
    }

    .content-box {
      width: 100%;
      background-color: lightcoral;
      border: 5px solid red;
    }

    .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
1. **border-top-left-radius**	: It is used to set the border-radius for the top-left corner
2. **border-top-right-radius**	: It is used to set the border-radius for the top-right corner
3. **border-bottom-right-radius**	: It is used to set the border-radius for bottom-right corner
4. **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 / (slash) notation to specify different horizontal and vertical radii for each corner.

Example:

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

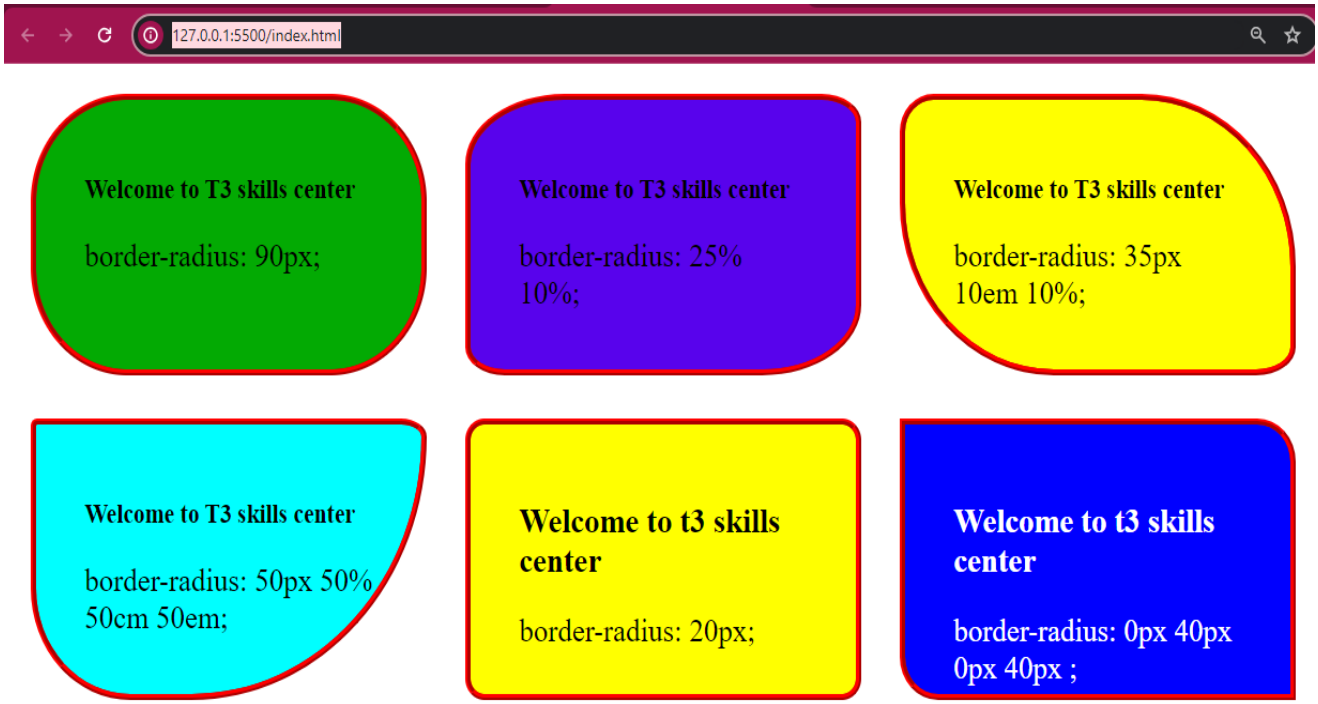
Example:

```
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;
            box-sizing: border-box;
        }
        #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;
        }
        .four{
            border-radius: 20px;
            background-color: yellow;
        }
        .five{
            border-radius: 0px 40px 0px 40px ;
            background-color: blue;
            color: white;
        }
    </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>
    <div class="four">
        <h1>Welcome to t3 skills center </h1>
        <p>border-radius: 20px;</p>
    </div>
    <div class="five">
        <h1>Welcome to t3 skills center </h1>
        <p>border-radius: 0px 40px 0px 40px ;</p>
    </div>
</body></html>
```



CSS LAYOUT - OVERFLOW

- The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.
- The overflow property has the following values:
  1. **visible -** Default. The overflow is not clipped. The content renders outside the element's box
  2. **hidden -** The overflow is clipped, and the rest of the content will be invisible
  3. **scroll -** The overflow is clipped, and a scrollbar is added to see the rest of the content
  4. **auto -** Similar to scroll, but it adds scrollbars only when necessary

Note: The overflow property only works for block elements with a specified height.

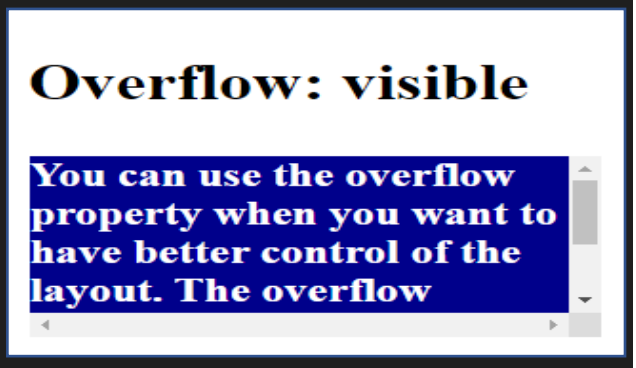
Note: In OS X Lion (on Mac), scrollbars are hidden by default and only shown when being used (even though "overflow:scroll" is set).

1. overflow: visible

- By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
background-color: darkblue;
width: 200px;
height: 65px;
border: 1px solid;
color: white;
font-weight: bold;
/* overflow: visible; */
/* overflow: hidden; */
overflow: scroll;
/* overflow: hidden; */
/* overflow: auto; */
/* overflow-y: scroll; */
/* overflow-x: scroll; */
/* overflow-wrap: break-word; */
}
</style>
</head>
<body>
<h2>Overflow: visible</h2>
<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>
</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).

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.

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.

4. float Property:

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

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;.

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.

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.

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.

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 lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }
        .box {
            width: 400px;
            height: 20vh;
            border: 5px solid red;
            margin: 3px;
            padding: 3px;
            color: white;
            font-weight: bold;
        }
        .box0 {
            width: 100%;
            height: 30vh;
            background-image: url(img2.PNG);
            background-size: cover;
            border: 5px solid green;
            border-radius: 0 20px;
            position: sticky;
            top: 0%;
            z-index: 1;
        }
        .box1 {
            background-color: aqua;
            position: relative;
            top: 0;
            color: black;
            background-image: url(img1.jpg);
            background-size: 50%;
            background-repeat: no-repeat;
        }
        .box2 {
            background-color: darkslateblue;
            position: relative;
            top: -30px;
            left: 93px;
            background-image: url(img1.jpg);
```

```
        background-size: 50%;
        background-repeat: no-repeat;
        /* position: absolute; */
        /* position: static; */
        /* position: sticky; */
        /* bottom: 30px; */
    }
    .box3 {
        background-color: green;
        position: relative;
        left: 160px;
        top: -60px;
        background-image: url(img1.jpg);
        background-size: 50%;
        background-repeat: no-repeat;
    }
    .logo4 {
        background-color: darkslateblue;
        position: absolute;
        top: 35vh;
        left: 840px;
        background-image: url(logo.PNG);
        background-size: cover;
        width: 310px;
        height: 50vh;
        border-radius: 20px;
        position: fixed;
    }
    .box5 {
        background-color: yellow;
        position: relative;
        /* position: relative; */
        /* position: static; */
        /* position: sticky; */
        /* position: fixed; */
        width: 810px;
        bottom: 0px;
        left: 5px;
    }
    .main {
        width: 95%;
        height: auto;
        border: 3px solid black;
        margin: 5px;
        padding: 5px;
        position: absolute;
        /* transform: translate(0); */
    }
```

```
        p{
            color: red;
        }
    </style>
</head>
<body>
    <div class="main">
        <div class="box box0"></div>
        <div class="box box1">1</div>
        <div class="box box2">2</div>
        <div class="box box3">3</div>
        <div class="box logo4">4</div>
        <div class="box box5">
            <p> <strong style=" font-weight: bold; color: darkblue; font-size:
20px;" >Research(अनुसंधान)</strong>:
                अनुसंधान एक प्रणालीकरण कार्य होता है जिसमें विशेष विषय या विषय की नई ज्ञान एवं समझ को प्राप्त करने के लिए सिद्धांतिक जांच
और अध्ययन किया जाता है। इसकी प्रक्रिया में डेटा का संग्रह और विश्लेषण, निष्कर्ष निकालना और विशेष क्षेत्र में मौजूदा ज्ञान में योगदान किया जाता है।
            <br>
            <b style=" color:  black; font-weight: bold;"> -Er.Rajesh Prasad
</b>
            </p>
        </div>
    </div>
</body>
</html>
```

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 property name:

1) none
2) inline
3) block
4) inline-block
5) flex
6) inline-flex
7) grid
8) inline-grid
9) list-item
10) list-item-group
11) initial
12) inherit
13) table
14) table-row
15) table-cell

- 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 ,<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 , <a>, and .

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.

Example for display: none, inline, block, inline-block

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>Display Examples</title>
<style>
  .none {
    display: none;
  }
  .inline {
    display: inline;
    background-color: lightblue;
    padding: 10px;
    line-height: 50px;
  }
  .c1{
    background-color: blue;
  }
  .c3{
    background-color: yellow;
  }
  .block {
    display: block;
    background-color: lightgreen;
    padding: 10px;
    margin-bottom: 10px;
  }
  .c11{
    background-color: blue;
  }
  .c33{
    background-color: yellow;
  }
  .inline-block {
    display: inline-block;
    background-color: lightcoral;
    padding: 10px;
    margin-right: 10px;
    margin-bottom: 10px;}
```

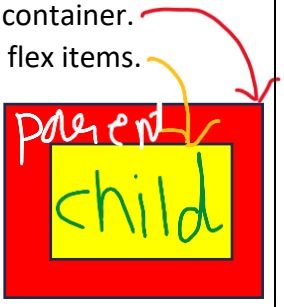
```
  .cc1{
    background-color: blue;
  }
  .cc2{
    background-color: red;
  }
  .cc3{
    background-color: blue;
  }
</style>
</head>
<body>
<div class="none">This is hidden with display: none</div>
<div class="inline">
<div class="inline c1">Inline Element 1</div>
<div class="inline c2">Inline Element 2</div>
<div class="inline c3">Inline Element 3</div>
</div>
<!-- Block -->
<div class="block">
<div class="block c11">Block Element 1</div>
<div class="block c22">Block Element 2</div>
<div class="block c33">Block Element 3</div>
</div>
<!-- Inline-block -->
<div class="inline-block">
<div class="inline-block cc1">Inline-block Element 1</div>
<div class="inline-block cc2">Inline-block Element 2</div>
<div class="inline-block cc3">Inline-block Element 3</div>
</div>
</body></html>
```

Display- Flex

Display: flex is a layout property that enables Flexbox layout model. where **child elements** (called flex items) are easily arranged in **rows or columns**. With Flexbox, you can control the alignment, spacing, and distribution of items within a container, even if their sizes are dynamic or vary in width/height.

❖ Key Concepts of Flexbox:

1. **Flex Container:** element to which display: flex is applied becomes the flex container.
2. **Flex Items:** The direct children of the flex container automatically become flex items.



❖ Flex Container Properties:

1. **display:** Defines the flex container (display: flex or display: inline-flex).
2. **flex-direction:** Specifies the direction of the flex items in the container.
   - row (default)
   - row-reverse
   - column
   - column-reverse
3. **justify-content**: Aligns flex items along main axis (horizontal if flex-direction: row).
   - flex-start (default)
   - flex-end
   - center
   - space-between
   - space-around
   - space-evenly
4. **align-items**: Aligns flex items along the cross-axis (vertical if flex-direction: row).
   - stretch (default)
   - flex-start
   - flex-end
   - center
   - baseline : (apply same size on item content)
5. **flex-wrap**: Controls whether the flex items should wrap onto multiple lines.
   - nowrap (default)
   - wrap
   - wrap-reverse
6. **flex-flow**: A shorthand for flex-direction and flex-wrap.
   - Example: flex-flow: row wrap;
7. **align-content**: Aligns flex lines along the cross-axis (applies only when there are multiple lines).
   - stretch (default)
   - flex-start
   - flex-end
   - center

- space-between
- space-around
- space-evenly

8. **gap**: Specifies the space between flex items (also works with grid layouts).
    - Example: gap: 10px; or gap:10px 20px; (gap: row column)
9. **row-gap**: Specifies the gap between flex rows.
    - Example: row-gap: 15px;
10. **column-gap**: Specifies the gap between flex columns.
    - Example: column-gap: 20px;

flex items

When using Flexbox (display: flex), various properties can be applied to the **flex items** (i.e., the child elements of the flex container) to control their behavior within the container. Below are the properties that apply specifically to **flex items**:

❖ Flex Item Properties:

1. **order**: Specifies the order of the flex items within the container. The default value is 0, and items with lower values appear first.
    - **Example:** order: 2;
2. **flex-grow**: Defines how much a flex item will grow relative to the rest of the items when there is extra space in the flex container. The default value is 0.
    - **Example:** flex-grow: 1; (item will grow to fill extra space)
3. **flex-shrink**: Defines how much a flex item will shrink relative to the rest of the items when there is not enough space in the flex container. The default value is 1.
    - **Example:** flex-shrink: 2; (item will shrink twice as much as other items)
4. **flex-basis**: Specifies the initial size of a flex item before any remaining space is distributed. It can be a fixed size (e.g., 200px) or a percentage. default value is auto.
    - **Example:** flex-basis: 150px;
5. **flex**: A shorthand property for setting flex-grow, flex-shrink, and flex-basis together. The default value is flex: 0 1 auto;.
    - **Example:** flex: 1 0 200px; (flex-grow: 1, flex-shrink: 0, flex-basis: 200px)
6. **align-self**: Overrides the align-items property for a specific flex item, allowing it to align differently along the cross-axis.
    - auto (default, follows align-items)
    - flex-start
    - flex-end
    - center
    - baseline
    - stretch
7. **margin** (with auto value): The margin property can also be used in combination with auto to align flex items.
    - Example: margin-left: auto; (pushes the item to the right side of the container)

Example: on display-flex

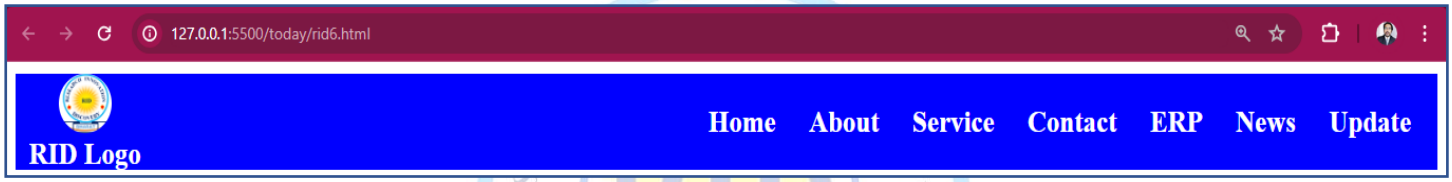
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>display-flex </title>
  <link rel="stylesheet" href="style.css">
</head> <body>
  <div class="parents">
    <div class="child child-1"> 1 </div>
    <div class="child child-2"> 2 </div>
    <div class="child child-3"> 3 </div>
    <div class="child child-4"> 4 </div>
    <div class="child child-5"> 5 </div>
   <div class="child child-6 "> 6 </div>
    <div class="child child-7 "> 7 </div>
    <div class="child child-8 "> 8 </div>
    <div class="child child-9 "> 9 </div>
    <div class="child child-10"> 10 </div>
  </div> </body> </html>
```

style.css

```
.parents{
   width: 90%;
  height: 60vh;
  border: 10px solid red;
  display: flex;
  display: inline-flex;
  flex-direction: row;
  /* flex-direction: column; */
  /* flex-direction: row-reverse;
  flex-direction: column-reverse; */
  /* justify-content: center; */
  /* justify-content: flex-end;
  justify-content: flex-start;
  justify-content: space-around;
  justify-content: space-between;
  justify-content: space-evenly; */
  /* align-items: center; */
  /* align-items: flex-end;
  align-items: flex-start;
  align-items: stretch; */
  /* align-items: baseline; */
/* flex-wrap: nowrap; */
  flex-wrap: wrap;
  /* flex-wrap: wrap-reverse; */
  /* flex-flow: row wrap; */
/*
  align-content: center; */
  /* align-content: flex-start;
  align-content: flex-end;
  align-content: space-around;
  align-content: space-between;
  align-content: space-evenly; */
/*
  gap: 10px 20px; */
  gap: 50px;
  /* row-gap: 20px; */
  /* column-gap: 40px; */
}
.child{
   width: 60px;
   height: 60px;
   border: 5px solid green;
}
.child-10{
   flex-grow: 3;
}
.child-1{
   /* flex-grow: 3; */
   /* margin-top: 20px; */
}
.child-2{
   font-size: 50px;
   /* flex-grow: 5; */
   /* align-self: flex-end; */
}
.child-3{
   /* flex-shrink: 3;
   align-self: flex-end; */
   /* align-self: flex-start; */
   align-self: flex-start;
}
.child-4{
  /* flex-grow: 3; */
}
.child-6{
   /* order: 2; */
   /* order: 3; */
}
.child-10{
```

```
Example-2
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Flex </title>
    <style>
        .navbar{
            width: 100%;
            height: 12vh;
            background-color: blue;
            align-items: center;
            justify-content: space-between;
            display: flex;
            font-size: 20px;
            font-weight: bold;
        }
        .logo{
        display: flex;
        flex-direction: column;
        align-items: center;
        margin-left: 10px;
        color: white;
        }
        .logo img{
            height: 40px;
            margin-bottom: 3px;
            border-radius: 50%;
            align-items: center;
        }
        .menu{
            margin-right: 10px;
            color: white;
        }
        .menu>div{
            display: inline-block;
            margin: 0 10px;
            text-align: end;
        }
    </style>
</head>
<body>
```

```
    <div class="navbar">
        <div class="logo">
            <img src="logo.PNG" alt="img">
            <span>RID Logo</span>
        </div>
        <div class="menu">
            <div class="c c1">Home</div>
            <div class="c c2">About</div>
            <div class="c c3">Service</div>
            <div class="c c4">Contact</div>
            <div class="c c5">ERP</div>
            <div class="c c6">News</div>
            <div class="c c7">Update</div>
        </div>
    </div>
</body></html>
```



Example-3: complete display flex concept

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>Document</title>
  <style>
    .m {
      width: 90%;
      height: 40vh;
      border: 5px solid black;
      border-radius: 0 10px;
      display: flex;
      color: black;
      font-weight: bold;
      align-items: center;
```

```
    /* align-items: flex-end; */
    /* align-items: flex-start; */
    /* flex-direction: column; */
    /* flex-direction: row-reverse; */
    /* flex-direction: column-reverse; */
    justify-content: center;
    /* justify-content: flex-start; */
    /* justify-content: flex-end; */
    /* justify-content: space-between; */
    /* justify-content: space-evenly; */
    /* justify-content: space-around; */
    /* flex-wrap: wrap; */
    /* flex-wrap: wrap-reverse; */
    /* flex-wrap: column wrap; */
  }
  .c {
    width: 200px;
    height: 28vh;
    border: 5px solid yellowgreen;
    margin: 5px;
    line-height: 5px;
  }
  .c1 {
    background-image: url(img1.jpg);
    background-size: cover;
    /* order: 3; */
  }
  .c2 {
    background-image: url(logo.PNG);
    background-size: cover;
  }
  .c3 {
    background-image: url('img1.jpg');
    background-size: cover;
    /* order: 2; */
    /* flex-grow: 5;
  height: 15vh; */
  }
  .c4 {
    background-image: url(logo.PNG);
    background-size: cover;
  }
```

```
    .c5 {
      background-image: url(pagali.jpg);
      background-size: cover;
    }
    .c6 {
      background-image: url(pagali2.jpg);
      background-size: cover;
    }
    p {
      margin-top: 29vh;
      color: black;
    }
  </style>
</head>
<body>
  <div class="m">
    <div class="c c1">1
      <p>Sangam</p>
    </div>
    <div class="c c2">2
      <p>Sushil</p>
    </div>
    <div class="c c3">3
      <p>Sujeet</p>
    </div>
    <div class="c c4">4
      <p>Satyam</p>
    </div>
    <div class="c c5">5
      <p>Priti</p>
    </div>
    <div class="c c6">6
      <p>Roshni</p>
    </div> </div> </body> </html>
```

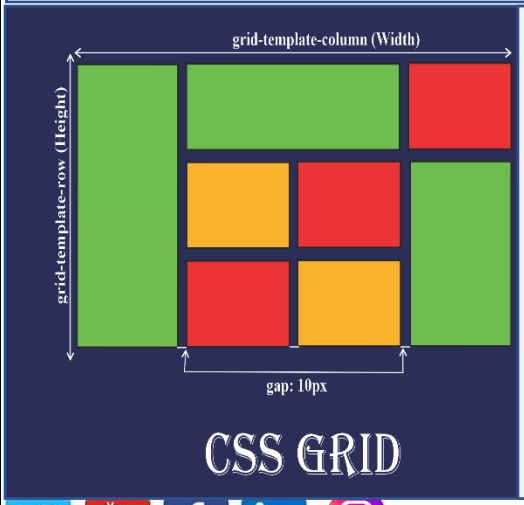
DISPLAY GRID

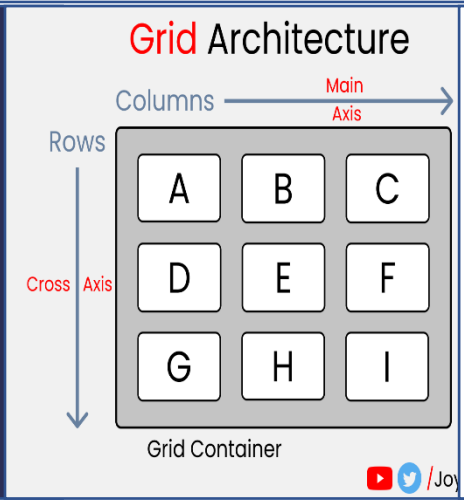
- CSS Grid, is a CSS module that provides a two-dimensional grid-based layout system, optimized for responsive design. It enables the creation of web layouts that are flexible, easy to manage, and visually consistent.

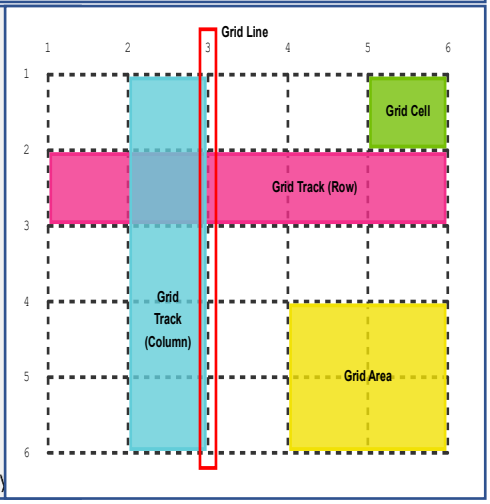
❖ Key Features:

- **Two-dimensional Layouts:** Unlike CSS Flexbox which is primarily one-dimensional, CSS Grid can handle both rows and columns simultaneously.
- **Grid Container and Grid Items:** A grid is made up of a grid container and its child elements, called grid items.
- **Tracks and Gaps:** Grid tracks (rows and columns) and gaps (gutters) can be defined to create the grid structure.
- **Flexible Sizing:** Supports flexible sizing with units such as fractions (fr), pixels (px), percentages (%), and more.
- **Placement and Alignment:** Grid items can be placed and aligned with precision using grid lines, spans, and areas.
- **Responsive Design:** It makes creating responsive layouts simpler through media queries and intrinsic sizing.

1) **grid:** It is shorthand for setting all the grid-related properties (grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns, grid-auto-flow, grid-gap, and grid).
2) **grid-template-rows:** Defines the size of the rows in a grid container.
3) **grid-template-columns:** Defines the size of the columns in a grid container.
4) **grid-template-areas:** Defines named grid areas.
5) **grid-template:** Shorthand for grid-template-rows, grid-template-columns, and grid-template-areas.
6) **grid-auto-rows:** Sets the size of automatically generated rows in a grid container.
7) **grid-auto-columns:** Sets the size of automatically generated columns in a grid container.
8) **grid-auto-flow:** Specifies how auto-placed items are positioned in the grid container.
9) **grid-row-gap:** Specifies the size of the gap between rows in a grid container.
10) **grid-column-gap:** Specifies the size of the gap between columns in a grid container.
11) **grid-gap:** Shorthand for grid-row-gap and grid-column-gap.
12) **justify-items:** Aligns grid items along the inline (row) axis.
13) **align-items:** Aligns grid items along the block (column) axis.
14) **justify-content:** Aligns grid items along the inline (row) axis of the grid container.
15) **align-content:** Aligns grid items along the block (column) axis of the grid container.
16) **grid-area:** Either specifies a name for the grid item, or it specifies the area within the grid to place the item.







Example-1

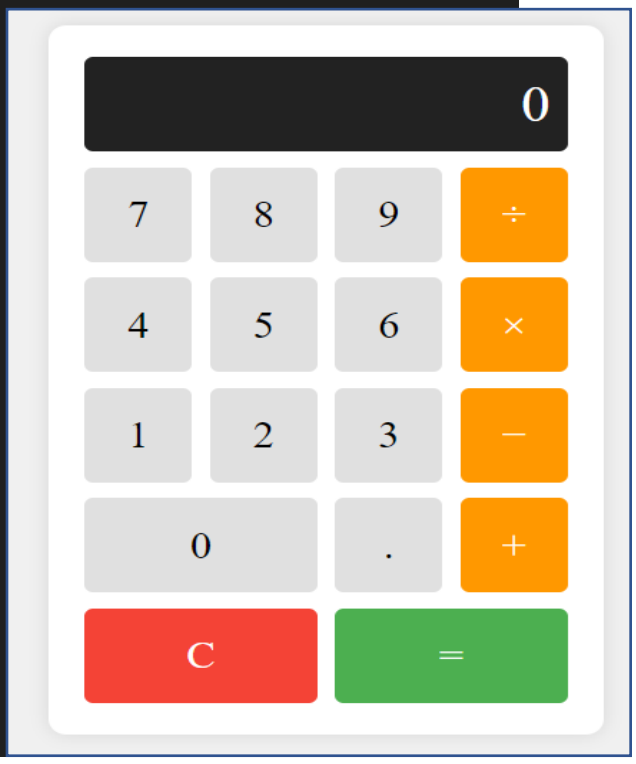
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>grid</title>
   <style>
      .container{
width: 800px;
height: 80vh;
background-color: blue;
display: grid;
gap: 10px;
grid-template-columns: 100px 100px 100px;
grid-template-rows: 100px 100px 100px; */
/* grid-template-columns: repeat(3, 1fr);
 grid-template-rows: repeat(3, 1fr); */
grid-template-columns: 1fr 1fr 1fr;
grid-template-rows: 1fr 1fr 1fr;
row-gap: 10px;column-gap: 30px;
      }
      .c{
         /* width: 100px;
         height: 100px; */
         background-color: brown;
      }
      .c1{
         grid-column-start: 1;
         grid-column-end: 3; }
      .c3{
         grid-row-start: 2;
         grid-row-end: 3;
         background-color: black;
         color: white;
         text-align: center; }
   </style>
</head>
<body>
   <div class="container">
      <div class="c c1">items1</div>
      <div class="c c2">items2</div>
      <div class="c c3">items3</div>
      <div class="c c4">items4</div>
      <div class="c c5">items5</div>
      <div class="c c6">items6</div>
      <div class="c c7">items7</div>
   </div>
</body>
</html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
   <style>
      .m{
         display: grid;
         grid-template-columns: 200px 200px 200px 200px;
         /* grid-template-columns: repeat(4, 300px);
         grid-template-columns: 1fr 1fr 1fr 1fr ; */
         /* grid-template-columns: repeat(auto-fill,
minmax(400px, 1fr)); */
         /* grid-template-rows: auto; */
         grid-template-rows: 200px 200px 200px 200px;
         /* grid-template-rows: repeat(5, 300px); */
         row-gap: 50px;
         column-gap: 30px;
         /* grid-auto-columns: 200px; */
         grid-auto-rows: minmax(100px, auto);
         grid-auto-columns: minmax(300px, auto);
         grid-template-areas:  "box1 box1 box1"
                               "box2 box2 box3";
         /* gap: 10px; */ }
      .c{  border: 5px solid red;
         border-radius: 10px; }
      .c1{ /* grid-column-start: 1;
         grid-column-end: 5;
         grid-auto-rows: 3; */
         grid-area: box1; }
      .c2{ /* grid-column-start: 2;
         grid-column-end: 4;
         grid-row-start: 1;
         grid-row-end: 4; */
         grid-area: box2; }
      .c3{ grid-area: box3;}
      .c4{ grid-area: box4; }
      .c5{grid-area: box5;}
      .c6{ grid-area: box6;} </style> </head> <body>
   <div class="m">
   <div class="c c1">box 1
      <img src="img1.jpg"> </div>
   <div class="c c2">box 2  lorem100 </div>
   <div class="c c3">box 3</div>
   <div class="c c4">box 4</div>
   <div class="c c5">box 5</div>
   <div class="c c6">box 6</div>
   </div>
</body> </html>
```

Example-2 Design Calculator and Design:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple CSS Grid Calculator</title>
    <style>   body {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color: #f0f0f0;
            margin: 0;
        }
        .calculator { display: grid;
            grid-template-areas:
                "display display display display"
                "seven eight nine divide"
                "four five six multiply"
                "one two three subtract"
                "zero zero dot add"
                "clear clear equals equals";
            grid-template-rows: repeat(6, 60px);
            grid-template-columns: repeat(4, 60px);
            gap: 10px;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
            background-color: #fff;
        }
        .display {
            grid-area: display;
            background-color: #222;
            color: #fff;
            font-size: 2rem;
            display: flex;
            justify-content: flex-end;
            align-items: center;
            padding: 0 10px;
            border-radius: 5px;
        }
        .button {
            background-color: #e0e0e0;
            font-size: 1.5rem;
            display: flex;
            justify-content: center;
```

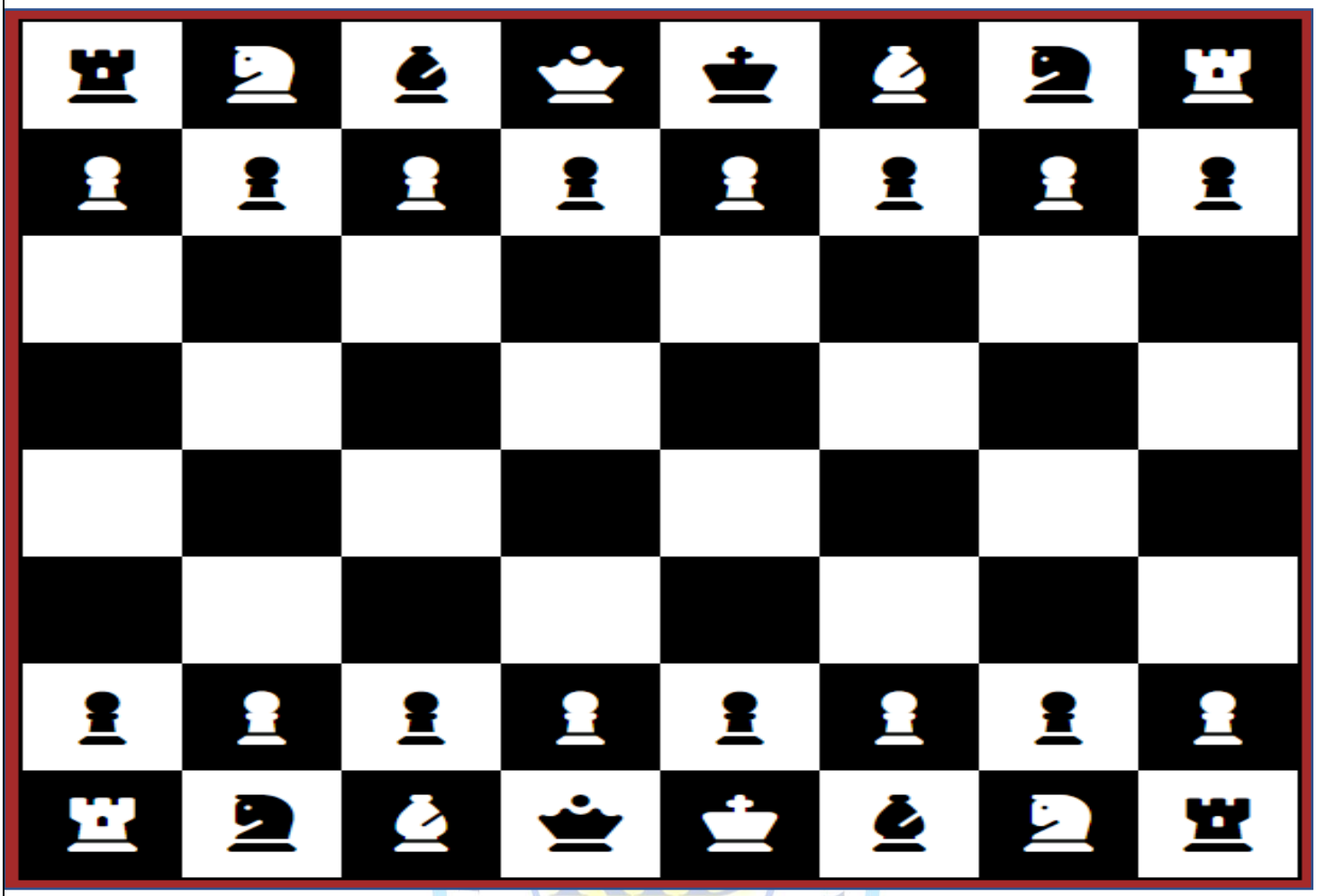
```
            align-items: center;
            border-radius: 5px;
            cursor: pointer;
            user-select: none;
        }
        .button:hover {
            background-color: #d0d0d0;
        }
        .button:active {
            background-color: #c0c0c0;}
    .clear { grid-area: clear; background-color: #f44336; color: #fff; }
    .divide { grid-area: divide; background-color: #ff9800; color: #fff; }
    .multiply { grid-area: multiply; background-color: #ff9800; color: #fff; }
    .subtract { grid-area: subtract; background-color: #ff9800; color: #fff; }
        .add { grid-area: add; background-color: #ff9800; color: #fff; }
        .equals { grid-area: equals; background-color: #4caf50; color: #fff; }
        .zero { grid-area: zero; }
        .one { grid-area: one; }
        .two { grid-area: two; }
        .three { grid-area: three; }
        .four { grid-area: four; }
        .five { grid-area: five; }
        .six { grid-area: six; }
        .seven { grid-area: seven; }
        .eight { grid-area: eight; }
        .nine { grid-area: nine; }
        .dot { grid-area: dot; }
    </style></head> <body>
    <div class="calculator">
        <div class="display">0</div>
        <div class="button clear">C</div>
        <div class="button divide">÷</div>
        <div class="button multiply">×</div>
        <div class="button subtract">−</div>
        <div class="button add">+</div>
        <div class="button equals">=</div>
        <div class="button zero">0</div>
        <div class="button one">1</div>
        <div class="button two">2</div>
        <div class="button three">3</div>
        <div class="button four">4</div>
        <div class="button five">5</div>
        <div class="button six">6</div>
        <div class="button seven">7</div>
        <div class="button eight">8</div>
        <div class="button nine">9</div>
        <div class="button dot">.</div>
    </div> </body> </html>
```



Example-3

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Chessboard with Font Awesome</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.2/css/all.min.css" integrity="sha512-
SnH5WK+bZxgPHs44uWIX+LLJAJ9/2PkPKZ5QiAj6Ta86w+fsb2TkcmfRyVX3pBnMFcV7oQPJkl9QevSCWr3W6A=="
crossorigin="anonymous" referrerpolicy="no-referrer" />
    <style>
        body {
            background-color: brown;
            height: 100vh;
            margin: 0;
            display: flex;
            align-items: center;
            justify-content: center;
        }
        .chessboard {
            display: grid;
            grid-template-columns: repeat(8, 60px);
            grid-template-rows: repeat(8, 60px);
            gap: 0;
            border: 2px solid black;
        }
        .square {
            width: 60px;
            height: 60px;
            display: flex;
            justify-content: center;
            align-items: center;
            font-size: 2rem;
        }
        .white {
            background-color: white;
        }
        .black {
            background-color: black;
            color: white;
        }
    </style>
</head>
<body>
    <div class="chessboard">
        <div class="square white"><i class="fas fa-chess-rook"></i></div>
        <div class="square black"><i class="fas fa-chess-knight"></i></div>
        <div class="square white"><i class="fas fa-chess-bishop"></i></div>
        <div class="square black"><i class="fas fa-chess-queen"></i></div>
        <div class="square white"><i class="fas fa-chess-king"></i></div>
        <div class="square black"><i class="fas fa-chess-bishop"></i></div>
        <div class="square white"><i class="fas fa-chess-knight"></i></div>
        <div class="square black"><i class="fas fa-chess-rook"></i></div>

        <div class="square black"><i class="fas fa-chess-pawn"></i></div>
        <div class="square white"><i class="fas fa-chess-pawn"></i></div>
        <div class="square black"><i class="fas fa-chess-pawn"></i></div>
        <div class="square white"><i class="fas fa-chess-pawn"></i></div>
        <div class="square black"><i class="fas fa-chess-pawn"></i></div>
        <div class="square white"><i class="fas fa-chess-pawn"></i></div>
```

```
        <div class="square black"><i class="fas fa-chess-pawn"></i></div>
        <div class="square white"><i class="fas fa-chess-pawn"></i></div>

        <div class="square white"></div>
        <div class="square black"></div>
        <div class="square white"></div>
        <div class="square black"></div>
        <div class="square white"></div>
        <div class="square black"></div>
        <div class="square white"></div>
        <div class="square black"></div>

        <div class="square black"></div>
        <div class="square white"></div>
        <div class="square black"></div>
        <div class="square white"></div>
        <div class="square black"></div>
        <div class="square white"></div>
        <div class="square black"></div>
        <div class="square white"></div>

        <div class="square white"></div>
        <div class="square black"></div>
        <div class="square white"></div>
        <div class="square black"></div>
        <div class="square white"></div>
        <div class="square black"></div>
        <div class="square white"></div>
        <div class="square black"></div>

        <div class="square black"></div>
        <div class="square white"></div>
        <div class="square black"></div>
        <div class="square white"></div>
        <div class="square black"></div>
        <div class="square white"></div>
        <div class="square black"></div>
        <div class="square white"></div>

        <div class="square white"><i class="fas fa-chess-pawn"></i></div>
        <div class="square black"><i class="fas fa-chess-pawn"></i></div>
        <div class="square white"><i class="fas fa-chess-pawn"></i></div>
        <div class="square black"><i class="fas fa-chess-pawn"></i></div>
        <div class="square white"><i class="fas fa-chess-pawn"></i></div>
        <div class="square black"><i class="fas fa-chess-pawn"></i></div>
        <div class="square white"><i class="fas fa-chess-pawn"></i></div>
        <div class="square black"><i class="fas fa-chess-pawn"></i></div>

        <div class="square black"><i class="fas fa-chess-rook"></i></div>
        <div class="square white"><i class="fas fa-chess-knight"></i></div>
        <div class="square black"><i class="fas fa-chess-bishop"></i></div>
        <div class="square white"><i class="fas fa-chess-queen"></i></div>
        <div class="square black"><i class="fas fa-chess-king"></i></div>
        <div class="square white"><i class="fas fa-chess-bishop"></i></div>
        <div class="square black"><i class="fas fa-chess-knight"></i></div>
        <div class="square white"><i class="fas fa-chess-rook"></i></div>
    </div>
</body>
</html>
```



CSS CURSOR PROPERTIES

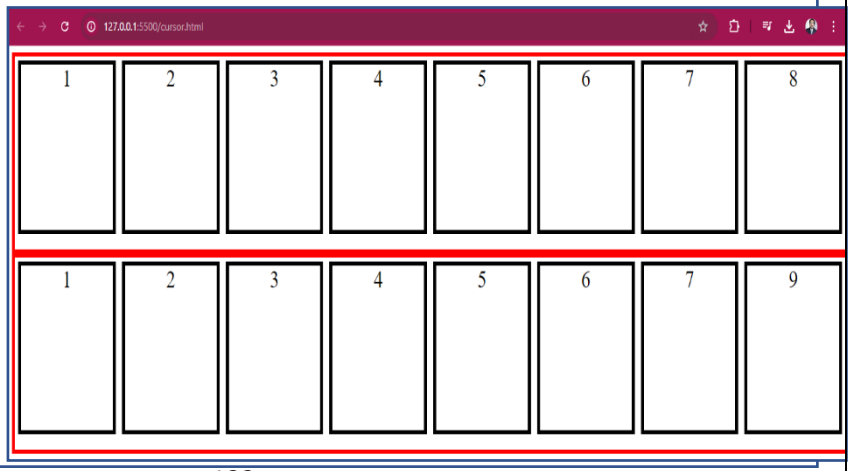
- These are some of the most common and important cursor properties you can use in CSS. Each property changes the cursor's appearance when it hovers over an element to indicate different kinds of actions or states.

Icon	Values	Examples
	default	style="cursor: default;"
	hand	style="cursor: hand;"
	pointer	style="cursor: pointer;"
	crosshair	style="cursor: crosshair;"
	text	style="cursor: text;"
	wait	style="cursor: wait;"
	help	style="cursor: help;"
	move	style="cursor: move;"
	e-resize	style="cursor: e-resize;"
	ne-resize	style="cursor: ne-resize;"
	nw-resize	style="cursor: nw-resize;"
	n-resize	style="cursor: n-resize;"
	se-resize	style="cursor: se-resize;"
	sw-resize	style="cursor: sw-resize;"
	s-resize	style="cursor: s-resize;"
	w-resize	style="cursor: w-resize;"
	progress	style="cursor: progress;"
	all-scroll	style="cursor: all-scroll;"
	col-resize	style="cursor: col-resize;"
	no-drop	style="cursor: no-drop;"
	not-allowed	style="cursor: not-allowed;"
	row-resize	style="cursor: row-resize;"
	vertical-text	style="cursor: vertical-text;"

Name values

1. default: cursor: default;
2. hand: cursor: hand;
3. pointer: cursor: pointer;
4. crosshair: cursor: crosshair;
5. text: cursor: text;
6. wait: cursor: wait;
7. help: cursor: help;
8. move: cursor: move;
9. e-resize: cursor: e-resize;
10. ne-resize: cursor: ne-resize;
11. nw-resize: cursor: nw-resize;
12. n-resize: cursor: n-resize;
13. se-resize: cursor: se-resize;
14. sw-resize: cursor: sw-resize;
15. s-resize: cursor: s-resize;
16. w-resize: cursor: w-resize;
17. progress: cursor: progress;
18. all-scroll: cursor: all-scroll;
19. col-resize: cursor: col-resize;
20. no-drop: cursor: no-drop;
21. not-allowed: cursor: not-allowed;
22. row-resize: cursor: row-resize;
23. vertical-text: cursor: vertical-text;

```
<!DOCTYPE html>
<html lang="en"><head>
   <meta charset="UTF-8">
   <style>
      .main{ width: 100%;
         height: 35vh;
         border: 5px solid red;
         display: flex;
         text-align: center;
         font-size: 30px;}
      .c{  width: 25%;
         height: 30vh;
         border: 5px solid black;
         margin: 5px;}
      .c1{ cursor: default; }
      .c2{ cursor: hand; }
      .c3{ cursor: pointer; }
      .c4{ cursor: crosshair;}
      .c5{cursor: text;}
      .c6{cursor: wait;}
      .c7{cursor: help;}
      .c8{cursor: move;}
      .c11{ cursor: e-resize; }
      .c22{ cursor: ne-resize; }
      .c33{ cursor: n-resize; }
      .c44{ cursor: not-allowed;}
      .c55{cursor: no-drop;}
      .c66{cursor: url(img1.jpg), pointer;}
      .c77{cursor: all-scroll;}
      .c88{cursor: progress;}
   </style></head><body>
   <div class="main">
   <div class="c c1">1</div>
   <div class="c c2">2</div>
   <div class="c c3">3</div>
   <div class="c c4">4</div>
   <div class="c c5">5</div>
   <div class="c c6">6</div>
   <div class="c c7">7</div>
   <div class="c c8">8</div>  </div>
<div class="main">
   <div class="c c11">1</div>
   <div class="c c22">2</div>
   <div class="c c33">3</div>
   <div class="c c44">4</div>
   <div class="c c55">5</div>
   <div class="c c66">6</div>
   <div class="c c77">7</div>
   <div class="c c88">9</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):

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

2.Visited:

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

3.Hover:

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

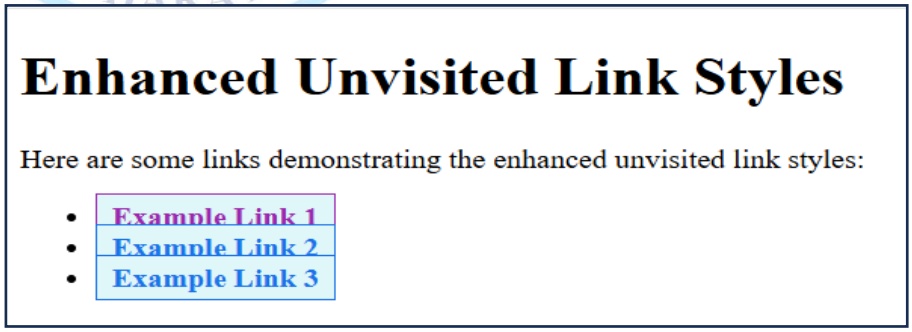
4.Active:

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

Note: unvisited link is a hyperlink that a user has not clicked or visited yet. In web development, the browser keeps track of links a user has visited. Links that have not yet been clicked by the user appear in their unvisited state, which can be styled differently from links that have already been visited. This behavior helps users differentiate between links they've already explored and those they haven't.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Enhanced Unvisited Link Style</title>
  <style>
    /* Unvisited link */
    a {
      color: #1A73E8;
      font-weight: bold;
      text-decoration: none;
      background-color: #e0f7fa;
      border: 1px solid #1A73E8;
      padding: 4px 8px;
      transition: background-color 0.3s, color 0.3s;
    }
    /* Visited link */
    a:visited {
      color: #9C27B0;
      background-color: transparent;
      border-color: #9C27B0;
    }
    /* Hover state */
    a:hover {
      background-color: #FF9800;
      color: #fff;
      text-decoration: underline;
    }
    /* Active state */
    a:active {
      color: red;
      background-color: black;
    }
  </style>
</head>
<body>
  <h1>Enhanced Unvisited Link Styles</h1>
  <p>Here are some links demonstrating the enhanced unvisited link styles:</p>
  <ul>
    <li><a href="https://www.example.com" target="_self">Example Link 1</a></li>
    <li><a href="https://www.sample.com" target="_self">Example Link 2</a></li>
    <li><a href="https://www.demo.com" target="_self">Example Link 3</a></li>
  </ul>
</body>
</html>
```



Drop Box

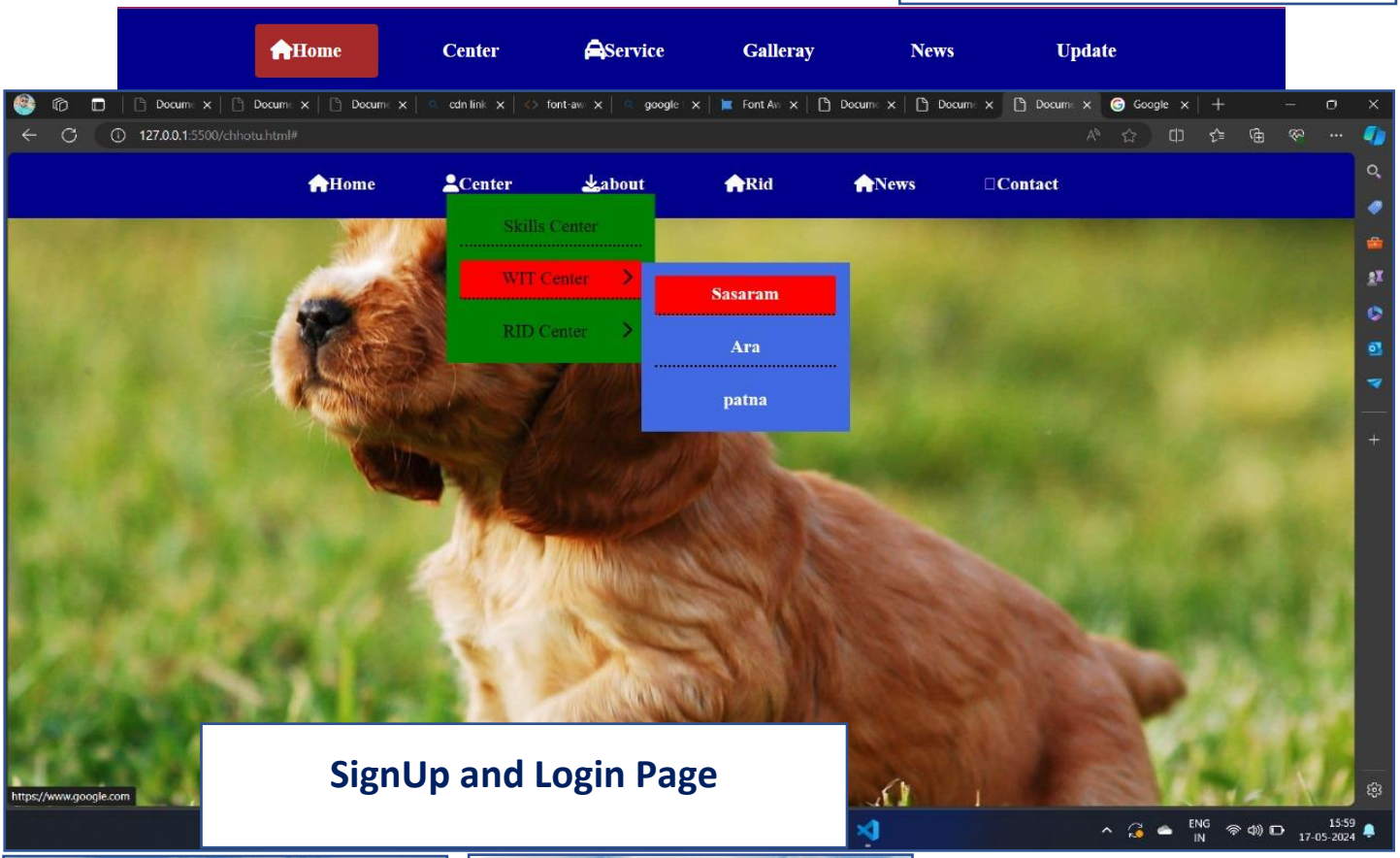
Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="dropbox.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ " />
</head>
<body>
  <div class="menu-bar">
    <ul>
<li id="act"><a href="#"><i class="fa-solid fa-house"></i>Home </a></li>
      <li><a href="#">Center</a>
        <div class="sub-menu-1">
          <ul>
            <li>Skills Center</li>
<li class="raj">WIT Center <i class="fa-solid fa-greater-than"></i>
            <div class="sub-menu-2">
              <ul>
                <li>Bhopal</li>
                <li>Patna</li>
                <li>Delhi</li>
              </ul>
            </div>
            </li>
<li class="raj" >RID Center <i class="fa-solid fa-greater-than"></i>
              <div class="sub-menu-2">
                <ul>
                  <li>Researcch Center</li>
                  <li>Innovation  Center</li>
                  <li>Discovery Center</li>
                </ul>
              </div></li></ul></div></li>
      <li><a href="#"><i class="fa-solid fa-taxi"></i>Service</a> </li>
      <li><a href="#">Galleray</a> </li>
      <li><a href="#">News </a>
        <div class="sub-menu-1">
          <ul>
            <li>Skills Center</li>
            <li>WIT Center <i class="fa-solid fa-greater-than"></i></li>
            <li>RID Center <i class="fa-solid fa-greater-than"></i></li>
          </ul>
        </div>
      </li>
      <li id="up"><a href="#">Update</a> </li>
    </ul> </div>
</body>
</html>
```

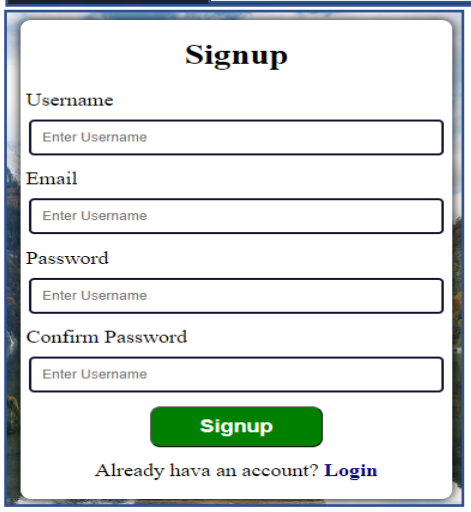
Dropboc.css

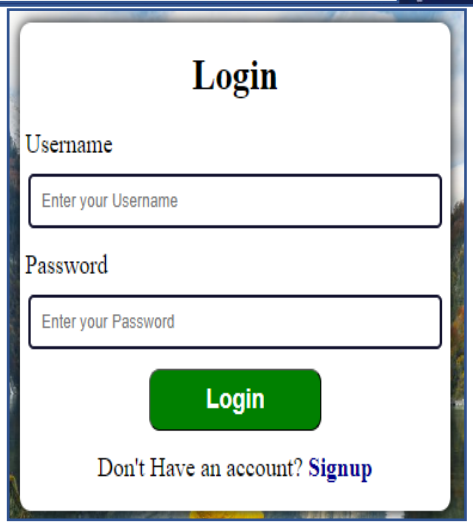
```
* { margin: 0;
   padding: 0;
   box-sizing: border-box }
body{
   background-image: url(nature1.jpg);
   background-size: cover;
   background-repeat: no-repeat; }
.menu-bar{
   background-color: darkblue;
   color: white;
   text-align: center; }
.menu-bar ul{
   display: inline-flex;
   list-style: none;}
.menu-bar ul li{
   padding: 20px;
   margin: 20px;
   font-size: 25px;
   position: relative;
   width: 160px; }
.menu-bar ul li a{
   text-decoration: none;
   color: white;
   font-weight: bold; }
#act, .menu-bar ul li:hover{
   background-color: brown;
   border-radius: 5px;
   text-align: left; }
.sub-menu-1{
   display: none; }
.menu-bar ul li:hover .sub-menu-1{
   display: block;
   margin-top: 20px;
   position: absolute;
  background-color: black;
   margin-left: -15PX; }
.menu-bar ul li:hover .sub-menu-1 ul{
   display: block; }
.sub-menu-1 ul li{
   font-size: 15px;
   border-bottom: solid red; }
.sub-menu-1 ul li:last-child{
   border-bottom: none; }
.fa-greater-than{
   float: right;
   color: white; }
.sub-menu-2{
   display: none; }
```

```
.raj:hover .sub-menu-2{
   display: block;
   margin-top: 0px;
   margin-left: 140px;
   position: absolute;
   background-color: black;
   /* background-image:
url(nature1.jpg); */
   background-size: cover;
   border: 5px solid red;
   border-radius: 5px;
   color: white;
   font-weight: bold;font-size: 30px;
}
```



SignUp and Login Page





Signup.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Signup</title>
  <link  rel="stylesheet" href="Style.css">
</head>
<body>
  <div class="container">
    <h1>Signup</h1>
    <form>
      <label for="username">Username</label>
      <input type="text" required placeholder="Enter Username">
      <label for="email">Email</label>
      <input type="email" required placeholder="Enter Username">
      <label for="password">Password</label>
      <input type="password" required placeholder="Enter Username">
      <label for="Re-Password">Confirm Password</label>
      <input type="password" required placeholder="Enter Username">
      <button type="submit">Signup</button>
    </form>
 <p>Already hava an account?  <a href="Login.html"> Login</a></p>
  </div>
</body> </html>
```

Login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Login</title>
  <link  rel="stylesheet" href="signup.css">
</head>
<body>
  <div class="container">
    <h1>Login</h1>
    <form>
      <label for="username">Username</label>
     <input type="text" required placeholder="Enter your Username">
      <label for="Password">Password</label>
      <input type="text" required placeholder="Enter your Password">
      <button type="submit">Login</button>
    </form>
    <p>Don't Have an account? <a href="Sigup.html">Signup</a></p>
  </div> </body> </html>
```

Style.css

```
body{
 /* background-color: aqua; */
   background-image: url(nature1.jpg);
   background-size: cover;
   margin: 0;
   padding: 0;
   display: flex;
   justify-content: center;
   align-items: center;
   height: 100vh; }
.container{
background-color: white;
   border-radius: 10px;
   box-shadow: 0 0 10px black;
   width: 400px;
   text-align: center; }
h1{margin-top: 20px; }
label{
display: block;
   text-align: left;
   margin-bottom: 10px;
   font-size: 20px;
   padding-left: 5px;}
input[type="text"],
input[type="email"],
input[type="password"]{
   width: 90%;
   padding: 10px;
   margin-bottom: 15px;
   border-radius: 5px;
   border: 3px solid rgb(18, 18, 46); }
button{ width: 40%;
   padding: 10px;
   background-color: green;
   color: white;
   font-weight: bold;
   border-radius: 10px;
   font-size: 20px;
   cursor: pointer; }
button:hover{ background-color: brown;}
p{ margin-top: 15px; font-size: 20px; }
a{
   text-decoration: none;
   /* color: green; */
   color: darkblue;
   font-weight: bold;
}
a:hover{
   text-decoration: underline;
}
```

APPLY STYLE FOR TABLE

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

❖ CSS Properties for Table Styling:

1. border

- **Usage:** Defines the border around elements.
- **Example:** border: 2px solid black; **Applies to:** table, th, td

2. border-collapse

- **Usage:** Specifies whether table borders should be collapsed into a single border or separated.
- **Ex:** border-collapse: collapse; , **border-collapse: separate**; (Default value) **Applies to:** table

3. border-spacing

- **Usage:** Specifies distance between borders of adjacent cells (only works with border-collapse: separate). **Example:** border-spacing: 10px 30px; **Applies to:** table

Note:

- **Single Value** (e.g., border-spacing: 10px;): Applies the same spacing horizontally and vertically between cells.
- **Two Values** (e.g., border-spacing: 10px 30px;): The first value sets horizontal spacing (distance between columns), and the second value sets vertical spacing (distance between rows).

Example:

```
<!DOCTYPE html>
<html lang="en"> <head>
  <meta charset="UTF-8">
  <title>Border Collapse Example</title>
  <style> /* Table with separated borders */
    .table-separate {
      border: 2px solid black; /* Table border */
   border-collapse: separate; /* Separate borders for cells */
      border-spacing: 10px 20px;
/* Space between cells (only works with 'separate') */
    } /* Table with collapsed borders */
    .table-collapse {
      border: 2px solid black; /* Table border */
      border-collapse: collapse;
/* Collapses borders into a single line */
    } /* Styling for cells */
    th, td {
      border: 2px solid black; /* Cell borders */
      padding: 10px;
      text-align: center; }
  </style> </head> <body>
  <h1>Table with Separate Borders</h1>
  <table class="table-separate">
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
    </tr>
    <tr>
      <td>Row 1, Cell 1</td>
      <td>Row 1, Cell 2</td>
    </tr>
    <tr>
      <td>Row 2, Cell 1</td>
      <td>Row 2, Cell 2</td>
    </tr>
  </table>

  <h1>Table with Collapsed Borders</h1>
  <table class="table-collapse">
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
    </tr>
    <tr>
      <td>Row 1, Cell 1</td>
      <td>Row 1, Cell 2</td>
    </tr>
    <tr>
      <td>Row 2, Cell 1</td>
      <td>Row 2, Cell 2</td>
    </tr>
  </table>
</body> </html>
```

Table with Separate Borders

Header 1	Header 2
Row 1, Cell 1	Row 1, Cell 2
Row 2, Cell 1	Row 2, Cell 2

Table with Collapsed

Header 1	Header 2
Row 1, Cell 1	Row 1, Cell 2
Row 2, Cell 1	Row 2, Cell 2

Table with Width and Height on Table, TH, and TD

Header 1	Header 2
Row 1, Cell 1	Row 1, Cell 2
Row 2, Cell 1	Row 2, Cell 2

4. Width and Height on <table>

- **width on <table>**: Defines the overall width of the table on the page. It can be set in percentages (e.g., width: 80%;) to make the table responsive or in fixed units (e.g., width: 500px;).
- **height on <table>:** Specifies the overall height of the table. If the content doesn't fill the specified height, it leaves empty space at the bottom. This property might not always be respected without table-layout: fixed;.

5. Width and Height on <th> and <td>

- **width on <th> or <td>:** Controls the width of individual columns or cells. When used on <th>, it typically affects the entire column, as headers span full columns. When used on <td>, it allows custom sizing of specific cells.
- **height on <th> or <td>:** Controls the height of each cell or row. If all cells in a row have the same height, the row will appear consistent in height.

6. table-layout: fixed;

- **Setting table-layout: fixed;** on the <table> element is useful when you want the width and height values to be strictly applied. It fixes the layout, ensuring the cells maintain specified dimensions without resizing to fit content. **table-layout: auto; (default value)**

Example:

```
<!DOCTYPE html>
<html lang="en">
<head> <title>Table Width and Height Example</title>
  <style>
    table {
      width: 80%;
      height: 300px;
      border-collapse: collapse;
      table-layout: fixed;
/* Fixes table layout for consistent sizing */
      border: 1px solid black; }
    th, td {
      width: 50%;
      height: 50px;
      border: 1px solid black;
      text-align: center;
      padding: 10px; } </style>
</head> <body>
  <h1>Table with Width and Height on Table, TH, and TD</h1>
  <table>
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
    </tr>
    <tr>
      <td>Row 1, Cell 1</td>
      <td>Row 1, Cell 2</td>
    </tr>
    <tr>
      <td>Row 2, Cell 1</td>
      <td>Row 2, Cell 2</td>
    </tr>
  </table></body> </html>
```

7. Padding in Table Cells

- **Usage**: The padding property defines the amount of space between the content of the cell (text, images, etc.) and the cell's border.
- **Example**: padding: 10px; adds 10 pixels of space inside each cell.
- **Applies to**: <th> and <td> elements.

8. text-align

- **Usage**: This property specifies how text (and other inline content) is aligned horizontally within the table cells.
- **Example**:
  - ✓ text-align: left; aligns the text to the left.
  - ✓ text-align: center; centers the text horizontally.
  - ✓ text-align: right; aligns the text to the right.
- **Applies to**: <td>, <th>, and sometimes the <table> (but typically for cell content).

9. vertical-align

- **Usage**: This property specifies the vertical alignment of content inside the table cells.
- **Example**:
  - ✓ vertical-align: top; aligns the content to the top of the cell.
  - ✓ vertical-align: middle; centers the content vertically within the cell.
  - ✓ vertical-align: bottom; aligns the content to the bottom of the cell.
- **Applies to**: <td>, <th>, and <table> (for table cell content).

10.caption-side

- **Usage**: The caption-side property specifies the position of a table's caption, which can be placed above or below the table.
- **Example**:
  - caption-side: top; places the caption above the table.
  - caption-side: bottom; places the caption below the table.
- **Applies to**: <caption> element.

11. :nth-child()

- **Usage**: The :nth-child() pseudo-class allows you to apply styles to specific child elements based on their position within a parent element. It can be used to style every nth element, such as every even or odd row in a table.
- **Example**:
  - tr:nth-child(even) { background-color: aqua; } styles every even row of the table with an aqua background.
- **Applies to**: <tr>, <td> elements.

12. :hover

- **Usage**: The :hover pseudo-class applies styles when the user hovers over an element, providing interactive feedback.
- **Example**: tr:hover { background-color: darkmagenta; } changes the background color of a row to dark magenta when hovered over.
- **Applies to**: <tr>, <td> elements.

13.Empty Cells in a Table

Empty cells are table cells (`<td>` or `<th>`) that do not contain any content. You can use CSS to style these cells, making it clear when data is missing. **Identifying Empty Cells** To target empty cells, you can use the `:empty` pseudo-class in CSS. This selects cells that have no content.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Table Styling Example</title>
  <style>
    table {
      width: 100%;
      border-collapse: collapse;
      margin-bottom: 20px;
    }
    caption {
      caption-side: top;
      font-weight: bold;
      margin-bottom: 10px;
    }
    th, td {
      border: 1px solid black;
      padding: 10px;
      text-align: center;
      vertical-align: middle;
    } /* Style for even rows */
    tr:nth-child(even) {
      background-color: aqua;
    }
    /* Hover effect for table rows */
    tr:hover {
      background-color: darkmagenta;
      color: white;
    } /* Style for empty cells */
    td:empty {
      background-color: #f9f9f9;
      color: #999;
      font-style: italic;
    } </style>
</head>
<body>
  <h1>Table Styling Example</h1>
  <table>

 <caption>Sample Table with Various Styles</caption>
    <tr>
      <th>Item</th>
      <th>Quantity</th>
      <th>Status</th>
    </tr>
    <tr>
      <td>Apples</td>
      <td>10</td>
      <td>Available</td>
    </tr>
    <tr>
      <td>Bananas</td>
      <td></td> <!-- Empty cell -->
      <td>Unavailable</td>
    </tr>
    <tr>
      <td>Cherries</td>
      <td>20</td>
      <td></td> <!-- Empty cell -->
    </tr>
    <tr>
      <td></td> <!-- Empty cell -->
      <td>5</td>
      <td>In Stock</td>
    </tr>
    <tr>
      <td>Grapes</td>
      <td>15</td>
      <td>Available</td>
```

Table Styling Example

Sample Table with Various Styles

Item	Quantity	Status
Apples	10	Available
Bananas		Unavailable
Cherries	20	
	5	In Stock
Grapes	15	Available

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
   <link rel="stylesheet" href="table.css">
</head>
<body>
   <table>
      <p>This is NRI Student list:</p>
 <caption>Table:Student Record</caption>
      <tr>
         <th>Name</th>
         <th>Roll_No</th>
         <th>Branch</th>
      </tr> <tr>
         <td>Deeraj</td>
         <td>53</td>
         <td>CSE</td>
      </tr> <tr>
         <td>Deepak</td>
         <td>52</td>
         <td>CSE</td>
      </tr> <tr>
         <td>Rahul</td>
         <td>51</td>
         <td>CSE</td>
      </tr> <tr>
         <td>Subhash</td>
         <td>50</td>
         <td>CSE</td>
      </tr>
      <tr>
         <td>Tushar</td>
         <td>49</td>
         <td>CSE</td>
      </tr> <tr>
         <td>Satya</td>
         <td>75</td>
         <td>CSE</td>
      </tr> <tr>
         <td id="d1">Jayoti</td>
         <td>49</td>
         <td>ME</td>
      </tr>
   </table>
   <h1>This is T3 Student list:</h1>
</body>
</html>
```

Table.css

```
table,th,td{
   border: 2px solid black;
   border-collapse: collapse;
   /* border-collapse: separate;
   border-spacing: 10px 10px; */
}
th,td{
   width: 45%;
   height: 30px; }
tr:nth-child(even){
   background-color: green;
}
tr:nth-child(odd){
   background-color: pink;
 }
tr:nth-child(8){
   background-color: chocolate; }
tr:nth-child(even):hover{
  background-color: darkturquoise; }
tr:nth-child(odd):hover{
   background-color: red;
   cursor: pointer; }
td:nth-child(even){
   background-color: darkblue;
   color: white;
   font-weight: bold;  }
th{
   background-color: aqua;
   text-align: center;
}
th,td{
   height: 50px; }
td{
   /* vertical-align: bottom;
   /* vertical-align: middle; */
   vertical-align: top;
}
caption{
   caption-side: bottom;
   caption-side: top;
}
 td{
   empty-cells: hide; }
 #d1{
   background-color: red;
   text-align: center;
   font-size: 30px;
 }
```

This is T3 Student list:

Table:Student Record

Name	Roll_No	Branch
Deeraj	53	CSE
Deepak	52	CSE
Rahul	51	CSE
Subhash	50	CSE
Tushar	49	CSE
Satya	75	CSE
Jayoti	49	ME

This is T3 Student list:

Task 1: Inventory Management Table

Description: Create a table to track inventory items in a store, showing item names, quantities, and their availability status.

Task 2: Student Grades Table

Description: Design a table to display student grades for a class, including student names, subjects, and their scores. Include highlighting for passing and failing grades.

Task 3: Employee Attendance Log

Description: Construct a table to log employee attendance, with columns for employee names, dates, and their attendance status (Present, Absent, or Leave). Use color coding to visually differentiate attendance statuses.

Task 4: Sales Report Table

Description: Create a table to summarize sales data for a business, including columns for product names, units sold, revenue generated, and sales representatives. Implement sorting functionality for each column.

Task 5: Event Schedule Table

Description: Design a table to display the schedule of an upcoming conference, featuring columns for session titles, speakers, timings, and locations. Use alternating row colors for better readability.

Task 6: Recipe Ingredients Table

Description: Construct a table for a recipe that lists ingredients, their quantities, and any notes (e.g., substitutions or preparation tips). Include styling to highlight ingredients that are optional

Task

Advanced Styled Table Example

Header 1	Header 2	Header 3	Header 4
Data 1	Data 2	Data 3	Data 4
Data 5	Data 6	Data 7	Data 8
Data 9	Data 10	Data 11	Data 12
Data 13	Data 14	Data 15	Data 16
Data 17	Data 18	Data 19	Data 20
Footer 1	**Footer 2**	**Footer 3**	**Footer 4**

Styled Table Example

Header 1	Header 2	Header 3	Header 4
Data 1	Data 2	Data 3	Data 4
Data 5	Data 6	Data 7	Data 8
Data 9	Data 10	Data 11	Data 12
Footer 1	Footer 2	Footer 3	Footer 4

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. Width and Height

- **Usage**: Control the size of the image.
- **Example**:

```
img {
   width: 100%; /* Full width of the container */
   height: auto; /* Maintains aspect ratio */
}
```

4. Padding

- **Usage**: Add space inside the border of the image (if wrapped in a container).
- **Example**:

```
.image-container {
   padding: 10px; /* Space inside the container */
}
```

2. Border

- **Usage**: Define the border around an image.
- **Example**:

```
img {
   border: 2px solid black; /* Solid black border */
}
```

3. Border Radius

- **Usage**: Create rounded corners for images.
- **Example**:

```
img {
   border-radius: 10px; /* Rounded corners */
}
```

5. Margin

- **Usage**: Create space outside the image.
- **Example**:

```
img {
   margin: 20px; /* Space around the image */
}
```

6. Display

- **Usage**: Control the display behavior of the image.
- **Example**:

```
img {
 display: block; /* Makes the image a block element */
margin: auto; /* Center the image */
}
```

9. Hover Effects

- **Usage**: Apply styles when the mouse hovers over the image.
- **Example**:

```
img:hover {
   transform: scale(1.1); /* Zoom in on hover */
   transition: transform 0.3s; /* Smooth transition */
}
```

10. Filter

- **Usage**: Apply visual effects like blur or brightness.
- **Example**:

```
img {
   filter: grayscale(100%); /* Convert to grayscale */
}
```

11. Box Shadow

- **Usage**: Add shadow effects around the image.
- **Example**:

```
img {
   box-shadow: 5px 5px 15px rgba(0, 0, 0, 0.5); /* Shadow effect */
}
```

12. Text Align

- **Usage**: Align text with respect to the image (when text is around the image).
- **Example**:

```
.image-container {
   text-align: center; /* Center align text with image */
}
```

7. Object Fit

- **Usage**: Control how an image fills its container.
- **Example**:

```
img {
   width: 100%; /* Fill the width of the container */
   height: 200px; /* Fixed height */
   object-fit: cover;
 /* Cover the entire container */
}
```

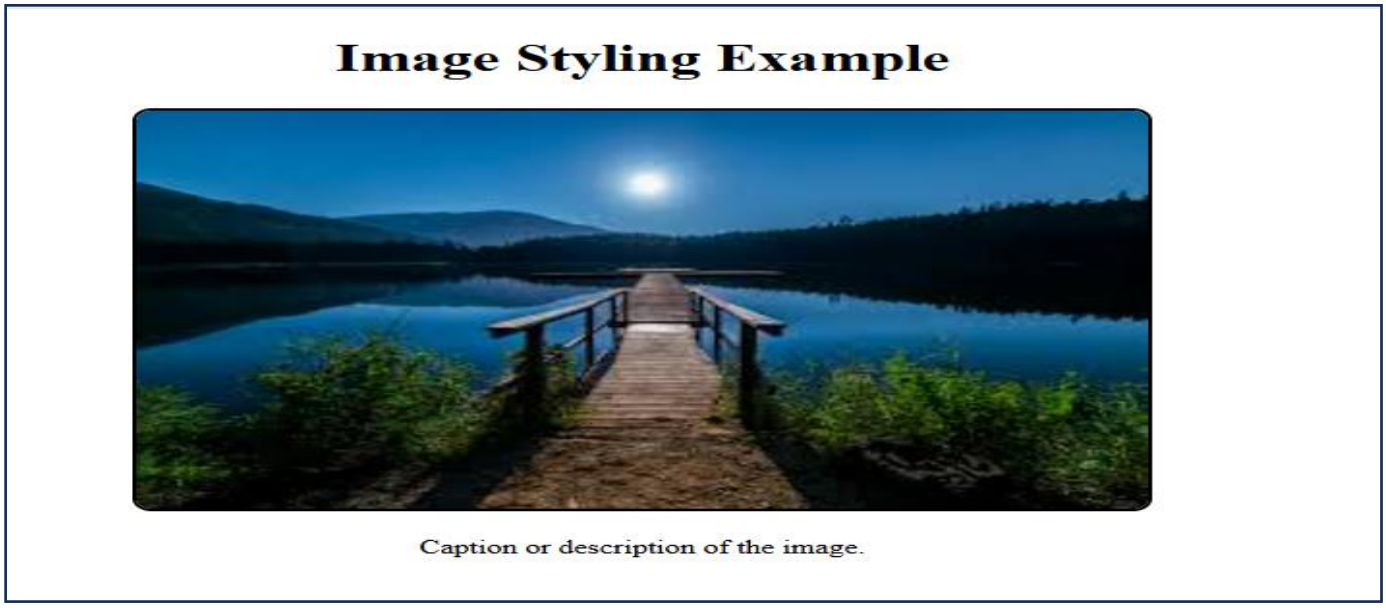
8. Opacity

- **Usage**: Control the transparency of the image.
- **Example**:

```
img {
   opacity: 0.8; /* Slightly transparent image */
}
```

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Image Styling Example</title>
  <style>
    img {
      width: 60%;
      height: auto;
      border: 2px solid black;
      border-radius: 10px;
      opacity: 0.9; /* Slightly transparent */
      transition: transform 0.3s; /* Transition effect for hover */
    }
    img:hover {
      transform: scale(1.1); /* Zoom effect on hover */
    }
    .image-container {
      margin: 20px;
      text-align: center;
    }
  </style>
</head>
<body>
  <h1>Image Styling Example</h1>
  <div class="image-container">
    <img src="img33.PNG" alt="Description of image">
    <p>Caption or description of the image.</p>
  </div>
</body> </html>
```

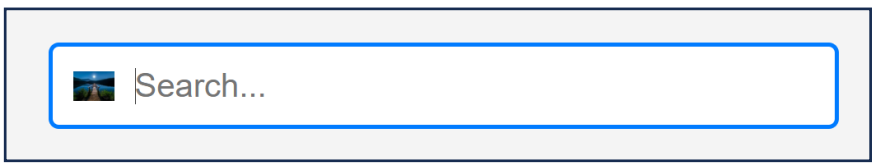


USE IMAGE IN TEXT FIELD BACKGROUND

- You can use an image as a background for a text field (or input field) in HTML by applying CSS styles. You can enhance the visual appearance of forms on a website.
  - ✓ **HTML Structure**: You need an <input> field where users can enter text.
  - ✓ **CSS Background Property**: Use the background-image property to set an image as the background of the input field. Other properties like background-size, background-repeat, and background-position help control how the image appears within the field.
  - ✓ **Padding and Color**: Adjust the padding to ensure the text doesn't overlap with the background image. You can also set the text color to ensure good visibility against the background.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Input Field Background Image</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      display: flex;  justify-content: center;
      align-items: center;
      height: 100vh;
    }
    .input-container {
      position: relative;
      width: 300px;
    }
    .input-container input {
      width: 100%;
      padding: 10px 40px;
      border: 2px solid #ccc;
      border-radius: 5px;
      background-image: url(t3.jpg);
      background-repeat: no-repeat;
      background-position: 10px center;
      background-size: 20px;
    }
    .input-container input:focus {
      border-color: #007bff; /* Change border color on focus */
      outline: none; /* Remove default outline */
    }
  </style>
</head>
<body>
  <div class="input-container">
    <input type="text" placeholder="Search...">
  </div>
</body></html>
```



ANIMATION

- CSS Animations allow you to animate the transition of elements in a web page, making your site more dynamic and engaging. With animations, you can change an element's properties, such as its position, size, color, and more, over a specified duration. CSS animations are created using a combination of keyframes and various animation properties.
- **animation:** name duration timing-function delay iteration-count direction fill-mode play-state;
- animation shorthand property allows you to set all animation properties in one declaration. The order of the values should be **animation-name**, **animation-duration**, **animation-timing-function, animation-delay**, **animation-iteration-count**, **animation-direction**, and **animation-fill-mode.**

1. @keyframes

- The @keyframes rule defines the changes in the CSS properties of an element at various stages of the animation. It specifies what styles the element will have at certain points during the animation.

Example:

➢ Using from and to

- This is the simplest form, using only the from and to keywords, representing the start and end of the animation.
- **Syntax:**

```
@keyframes example {
  from { background-color: red; }
  to { background-color: yellow; }
}
```

➢ Using Percentage Values

- Percentage values represent specific points animation timeline (0% is the start, 100% is end).
- **Syntax**:

```
@keyframes example {
  0% { background-color: red; }
  50% { background-color: blue; }
  100% { background-color: yellow; }
}
```

➢ Combining from, to, and Percentage Values

- You can combine from/to with percentages if you prefer.
- **Syntax**:

```
@keyframes example {
  from { background-color: red; }
  50% { background-color: blue; }
  to { background-color: yellow; }
}
```

➢ Multiple Property Changes at Keyframes

- Change multiple CSS properties at different points.
- **Syntax**:

```
@keyframes example {
  0% {
    background-color: red;
    transform: scale(1);
  }
```

```
    50% {
      background-color: blue;
      transform: scale(1.5);
    }
    100% {
      background-color: yellow;
      transform: scale(1); } }
```

➢ Specifying Keyframes for a Specific CSS Property

- Only changing one property throughout the keyframes (though other properties will remain unaffected).
- **Syntax**:

```
@keyframes scaleUp {
  0% { transform: scale(1); }
  100% { transform: scale(1.5); }
}
```

➢ Using @keyframes with Vendor Prefixes

- Some browsers may require vendor prefixes for @keyframes (e.g., -webkit- for older versions of Safari and Chrome).
- **Syntax**:

```
@-webkit-keyframes example {
  from { background-color: red; }
  to { background-color: yellow; }
}
@keyframes example {
  from { background-color: red; }
  to { background-color: yellow; }
}
```

- **Usage**:

```
.element {
  -webkit-animation: example 2s infinite;
  animation: example 2s infinite;
}
```

2. Using Multiple Animations with Different Keyframes

- You can apply multiple @keyframes animations to a single element by specifying them in the animation property.
- **Syntax**:

```
@keyframes fadeIn {
  0% { opacity: 0; }
  100% { opacity: 1; }
}
@keyframes moveRight {
  0% { transform: translateX(0); }
  100% { transform: translateX(100px); }
}
```

3. animation-name

- The animation-name property assigns a name to the @keyframes rule that should be applied to the element. It links the keyframes to the element.

Example:

```
.animated-element {
  animation-name: example; }
```

4. animation-duration

- The animation-duration property specifies the length of time an animation should take to complete one cycle. It can be defined in seconds (s) or milliseconds (ms).

Example:

```
.animated-element {
  animation-duration: 4s; }
```

5. animation-delay

- The animation-delay property defines the amount of time to wait before the animation starts. This can also be specified in seconds (s) or milliseconds (ms).

Example:

```
.animated-element {
  animation-delay: 2s; }
```

6. animation-iteration-count

- The animation-iteration-count property specifies the number of times an animation cycle should be played. It can be a number or the keyword infinite for continuous repetition.

Example:

```
.animated-element {
  animation-iteration-count: infinite;
}
```

7. animation-direction

- The animation-direction property determines whether the animation should play in reverse on alternate cycles. It can take values like **normal, reverse, alternate, or alternate-reverse**.

Example:

```
.animated-element {
  animation-direction: alternate;
}
```

8. animation-timing-function

- The animation-timing-function property specifies the speed curve of the animation. It defines how the intermediate states of the animation are calculated. Common values are linear, ease, ease-in, ease-out, and ease-in-out.

Example:

```
.animated-element {
  animation-timing-function: ease-in-out;
}
```

9. animation-fill-mode

- The animation-fill-mode property specifies how a CSS animation applies styles to its target before and after its execution. It can take values such as none, forwards, backwards, and both.

Example:

```
.animated-element {
  animation-fill-mode: forwards;
}
```

```
<!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>
   .c1{
      width: 200px;
      height: 200px;
      border: 2PX solid red;
      border-radius: 5px;
      /* background-color: antiquewhite; */
      animation-name: smy;
      animation-duration: 6s;
      animation-iteration-count: infinite;
      background-size: cover;
      box-sizing: border-box;
      position: relative;
      animation-delay: 1s;
      /* animation-fill-mode: both; */
      animation-fill-mode: both;
      /* animation-direction: reverse; */
      animation-direction: alternate;
      /* animation: mymove 5s; */
      color: black;
      font-weight: bold;
      text-align: center;
      font-size: 25px;
      text-shadow: 10px salmon;
   }
   @keyframes smy{
      0%{
         background-image: url(img3.jpg);
         left: 0;
         top: 0;
      }
      25%{
```

```
   @keyframes smy{
      0%{
         background-image: url(img3.jpg);
         left: 0;
         top: 0;
      }
      25%{
        background: url(img2.jpg);
        left: 400px;
        top: 0;
      }
      50%{
         background-image: url(img3.jpg);
         left: 400px;
         top: 400px;
         color: aqua;
      }
      75%{
         background-image: url(img1.jpg);
         left: 0;
         top: 400px;
         color: green;
      }
      100%{
         background-image: url(img3.jpg);
         left: 0;
         top: 0;
      }}
   .c2{
      width: 196px;
      height: 192px;
      /* background-color: red; */
      border-radius: 5px;
      margin-left: 200px;
      border: 3px solid green;
      background-size: cover;
   }
 </style>
</head>
<body>
<h1> Welcome all of you .....</h1>
<div class="c1">Name</div>
<div class="c2"><img src="t6.jpg"
width="190px" height="190px"></div>
</body>
</html>
```

Welcome all of you .....

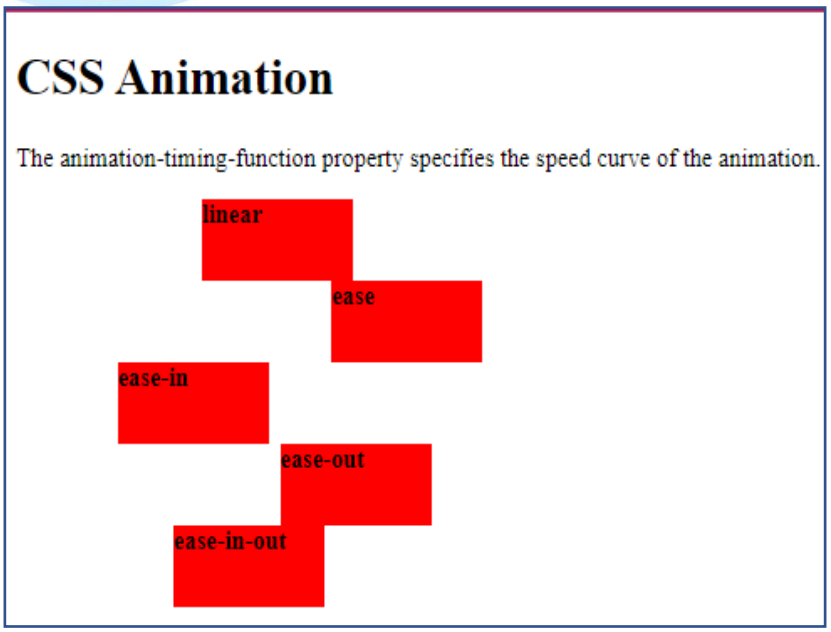
❖Specify the Speed Curve of the Animation:

- The animation-timing-function property specifies the speed curve of the animation.
- The animation-timing-function property can have the following values:

1. **ease -** Specifies an animation with a slow start, then fast, then end slowly (this is default)
2. **linear -** Specifies an animation with the same speed from start to end
3. **ease-in -** Specifies an animation with a slow start
4. **ease-out -** Specifies an animation with a slow end
5. **ease-in-out -** Specifies an animation with a slow start and end
6. **cubic-bezier(n,n,n,n) -** Lets you define your own values in a cubic-bezier function

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 50px;
  background-color: red;
  font-weight: bold;
  position: relative;
  animation: mymove 5s;
  animation-fill-mode: forwards;
}
#div1 {animation-timing-function: linear;}
#div2 {animation-timing-function: ease;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
#div5 {animation-timing-function: ease-in-out;}
@keyframes mymove {
  from {left: 0px;}
  to {left: 300px;}
}
</style>
</head>
<body>
<h1>CSS Animation</h1>
<div id="div1">linear</div>
<div id="div2">ease</div>
<div id="div3">ease-in</div>
<div id="div4">ease-out</div>
<div id="div5">ease-in-out</div>
</body>
</html>
```



❖ Specify the fill-mode For an Animation:

- CSS animations do not affect an element before first keyframe is played or after the last keyframe is played. The animation-fill-mode property can override this behavior.
- The animation-fill-mode property specifies a style for the target element when the animation is not playing (before it starts, after it ends, or both).

➢ The animation-fill-mode property can have the following values:

1. **none -** Default value. Animation will not apply any styles to the element before or after it is executing
2. **forwards -** The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count)
3. **backwards -** The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period
4. **both -** The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions

Example-1:

```
div {
  width: 100px;
  height: 100px;
  background: red;
  position: relative;
  animation-name: example;
  animation-duration: 3s;
  animation-fill-mode: forwards; }
```

Example-2:

```
div {
  width: 100px;
  height: 100px;
  background: red;
  position: relative;
  animation-name: example;
  animation-duration: 3s;
  animation-delay: 2s;
  animation-fill-mode: backwards; }
```

Example-3

```
div {
  width: 100px;
  height: 100px;
  background: red;
  position: relative;
  animation-name: example;
  animation-duration: 3s;
  animation-delay: 2s;
  animation-fill-mode: both; }
```

❖ Animation Shorthand Property

- The example below uses six of the animation properties:

Example

```
div {
  animation-name: example;
  animation-duration: 5s;
  animation-timing-function: linear;
  animation-delay: 2s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```

Property	Description
1) **@keyframes**	Specifies the animation code
2) **Animation**	A shorthand property for setting all the animation properties
3) **animation-delay**	Specifies a delay for the start of an animation
4) **animation-direction**	Specifies whether an animation should be played forwards, backwards or in alternate cycles
5) **animation-duration**	Specifies how long time an animation should take to complete one cycle
6) **animation-fill-mode**	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
7) **animation-iteration-count**	Specifies the number of times an animation should be played
8) **animation-name**	Specifies the name of the @keyframes animation
9) **animation-play-state**	Specifies whether the animation is running or paused
10) **animation-timing-function**	Specifies the speed curve of the animation

HOW TO ADD MAP IN 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:

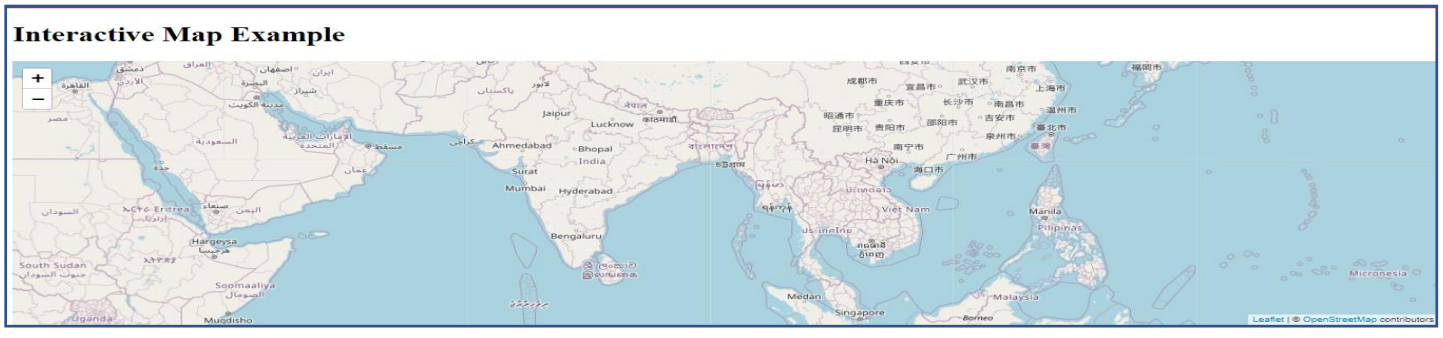
3. Initialize the Map with JavaScript:

- 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>
```

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>
    #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:

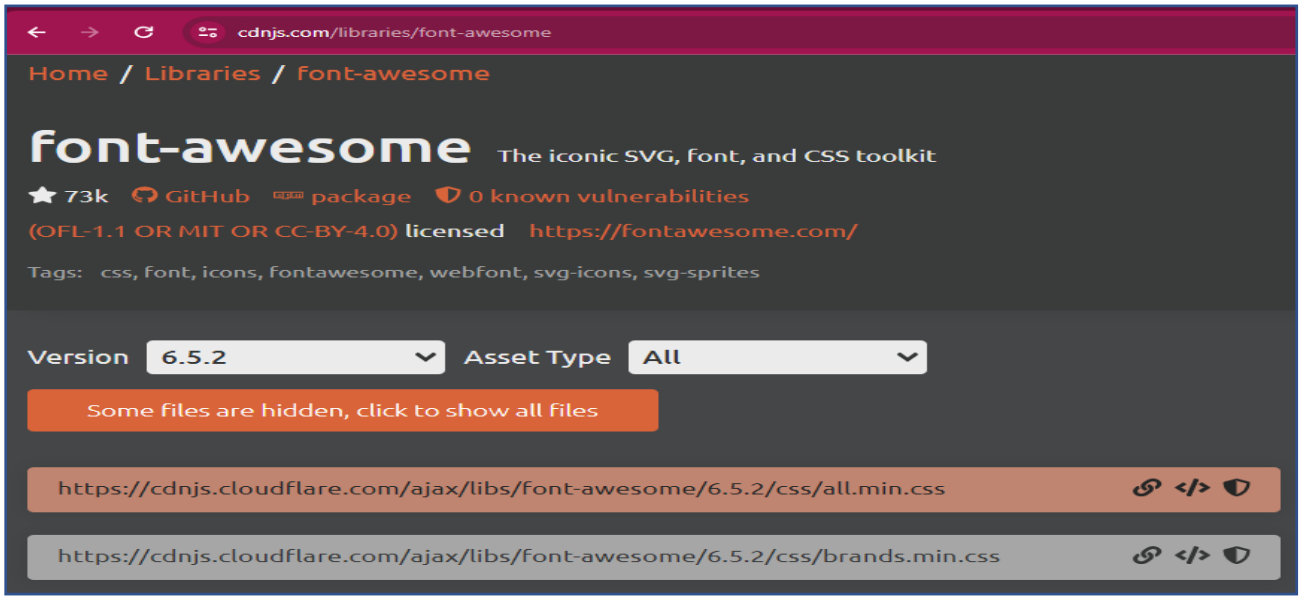
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Google Fonts Example</title>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=FontName">
</head>
<body>
  <!-- Your website content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

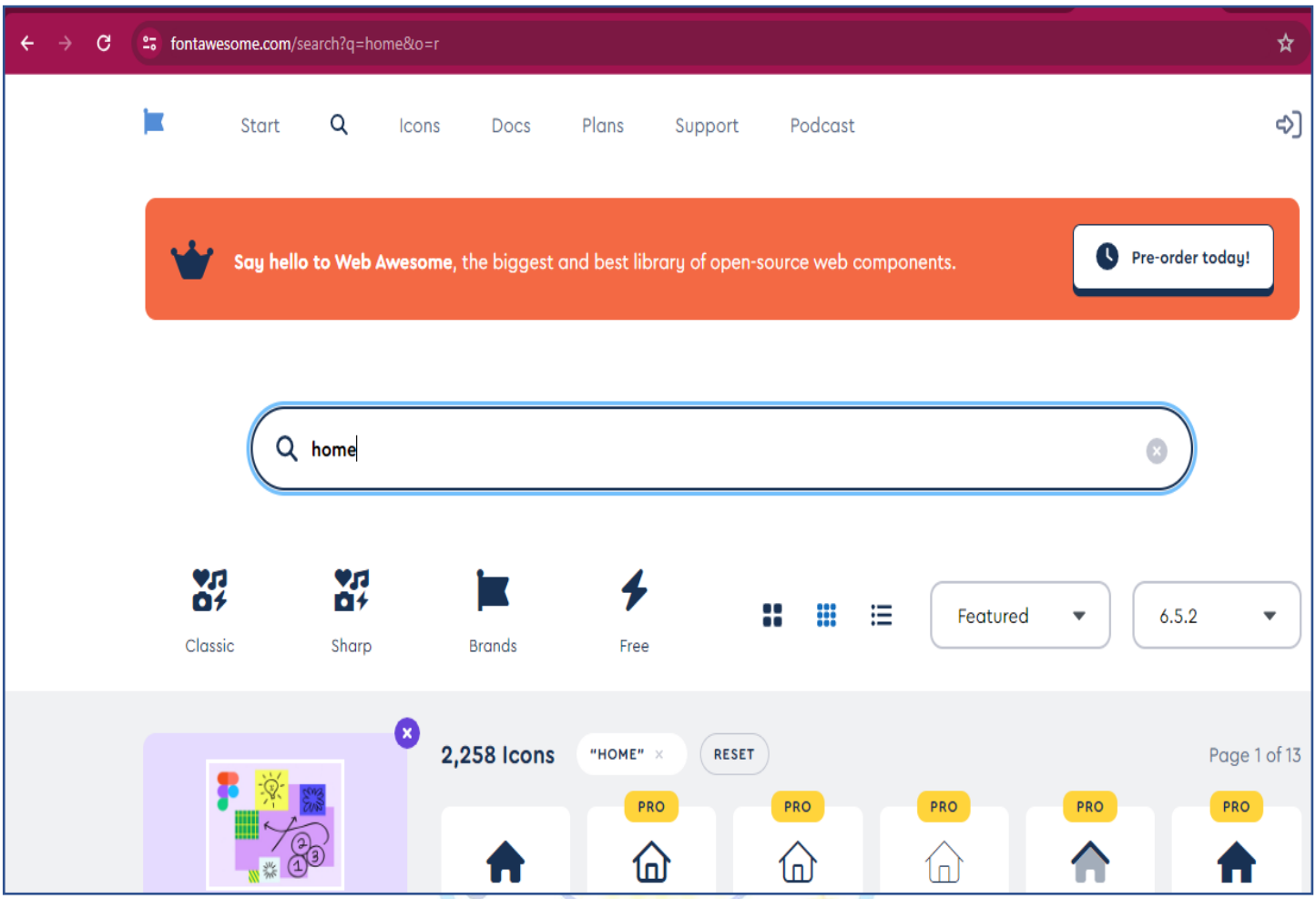
6. Apply the Font in CSS:

- 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.
```




fontawesome.com/search?q=home&o=r
Start
Icons
Docs
Plans
Support
Podcast
Say hello to Web Awesome, the biggest and best library of open-source web components.
Pre-order today!
home
Classic
Sharp
Brands
Free
Featured
6.5.2
2,258 Icons
"HOME"
RESET
Page 1 of 13
PRO
PRO
PRO
PRO
PRO

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.

❖ Transformation Types:

- **Translate:** Moves an element along the X, Y, or Z axis without affecting its original position in the document flow.
- **Scale**: Resizes an element proportionally along the X and/or Y axis, either enlarging or shrinking it.
- **Rotate**: Rotates an element around a specified point, either in 2D (around the Z-axis) or in 3D space (around the X, Y, or Z-axis).
- **Skew**: Tilts or shears an element along the X or Y axis, creating a slanted or "parallelogram" effect.

Explanation:

1. Translate

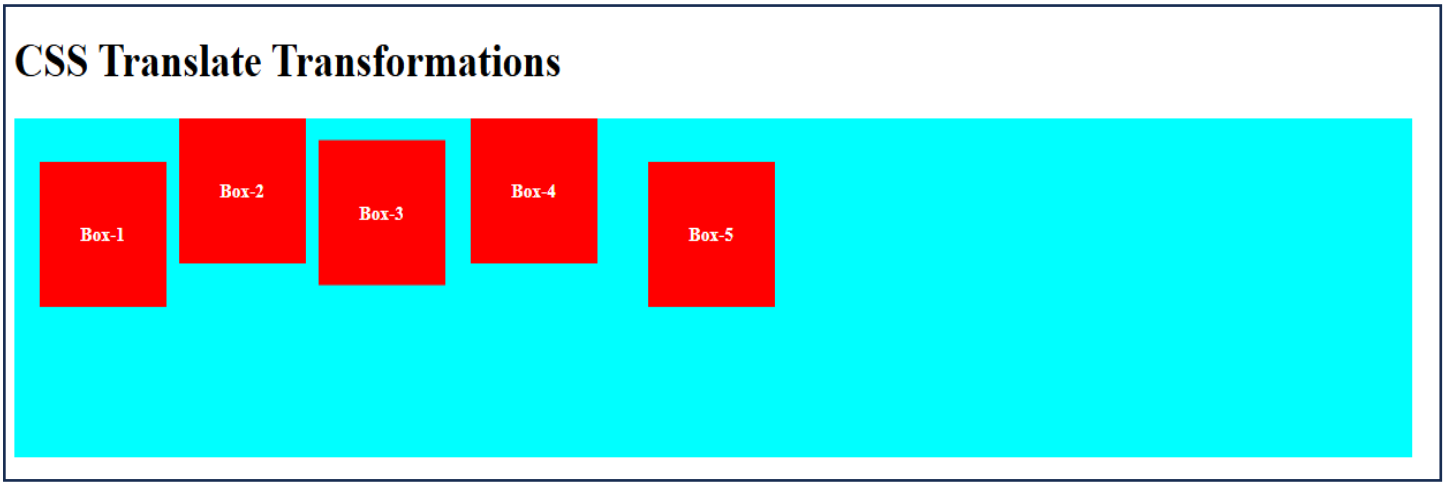
- Moves an element along the X, Y, or Z axis without affecting its original position in the document flow. Useful for shifting elements around without impacting other elements.

- **translate(x, y)**: Moves an element horizontally by x and vertically by y.
  - transform: translate(20px, 30px); /* Moves 20px right, 30px down */
- **translateX(x)**: Moves an element along the X-axis.
  - transform: translateX(10px); /* Moves 10px to the right */
- **translateY(y)**: Moves an element along the Y-axis.
  - transform: translateY(15px); /* Moves 15px down */
- **translateZ(z)**: Moves an element along the Z-axis, adding depth (used in 3D transformations).
  - transform: translateZ(50px); /* Moves 50px "closer" in 3D space */
- **translate3d(x, y, z)**: Moves an element in 3D space along the X, Y, and Z axes.
  - transform: translate3d(20px, 30px, 50px); /* Moves 20px right, 30px down, and 50px forward */

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>CSS Translate Transformations</title>
 <style>
  .container {
   display: flex;
   flex-wrap: wrap;
   gap: 20px;}
```

```
    .box {
      width: 100px;
      height: 100px;
      background-color: red;
      color: white;
      display: flex;
      align-items: center;
      justify-content: center;
      font-weight: bold;
      font-size: 14px;
    }
    .translate { transform: translate(20px, 30px);  }
    .translateX {transform: translateX(10px); }
    .translateY { transform: translateY(15px);  }
    .translateZ { transform: translateZ(50px);  }
    .translate3d { transform: translate3d(20px, 30px, 50px);  }
    .container {
      width: 90%;
      height: 40vh;
      background-color: aqua;
    }
  </style>
</head>
<body>
  <h1>CSS Translate Transformations</h1>
  <div class="container">
    <div class="box translate">Box-1</div>
    <div class="box translateX">Box-2</div>
    <div class="box translateY">Box-3</div>
    <div class="box translateZ">Box-4</div>
    <div class="box translate3d">Box-5</div>
  </div>
</body>
</html>
```



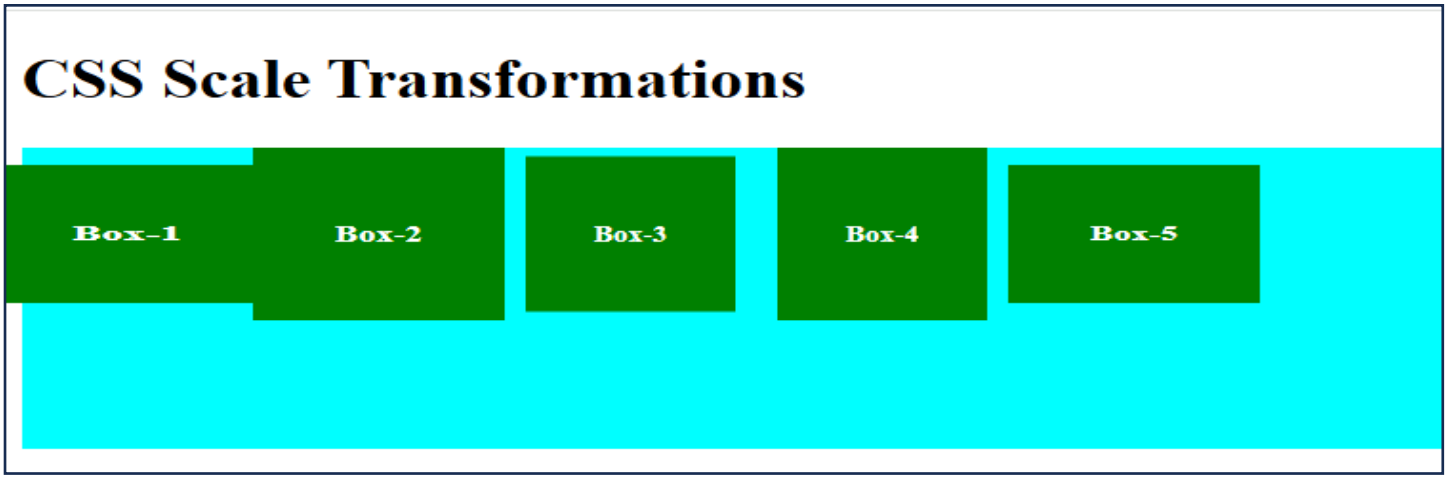
2. Scale

-Resizes an element along X, Y, & Z axes. Values greater than 1 enlarge, values between 0 and 1 shrink.

- **scale(x, y)**: Scales the width by x and height by y.
  - transform: scale(1.5, 0.8); /* 150% wider, 80% shorter */
- **scaleX(x)**: Scales the width by x.
  - transform: scaleX(1.2); /* 120% wider */
- **scaleY(y)**: Scales the height by y.
  - transform: scaleY(0.9); /* 90% height */
- **scaleZ(z)**: Scales the element along the Z-axis, used in 3D space.
  - transform: scaleZ(1.5); /* 150% depth scaling */
- **scale3d(x, y, z)**: Scales the element along the X, Y, and Z axes.
  - transform: scale3d(1.2, 0.8, 1.5); /* 120% width, 80% height, 150% depth */

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">

 <title>CSS Scale Transformations</title>
 <style>
  .container {
   display: flex;
   flex-wrap: wrap;
   gap: 20px; }
  .box {
   width: 100px;
   height: 100px;
   background-color:green;
   color: white;
   display: flex;
   align-items: center;
   justify-content: center;
   font-weight: bold;
   font-size: 14px;
}
  .scale { transform: scale(1.5, 0.8); }
  .scaleX {transform: scaleX(1.2);{
  .scaleY { transform: scaleY(0.9); }
  .scaleZ { transform: scaleZ(1.5);  }
  .scale3d { transform: scale3d(1.2, 0.8, 1.5); }
  .container {
   width: 90%;
   height: 30vh;
   background-color: aqua; }
 </style>
</head>
<body>
 <h1>CSS Scale Transformations</h1>
 <div class="container">
  <div class="box scale">Box-1</div>
  <div class="box scaleX">Box-2</div>
  <div class="box scaleY">Box-3</div>
  <div class="box scaleZ">Box-4</div>
  <div class="box scale3d">Box-5</div>
 </div>
</body> </html>
```



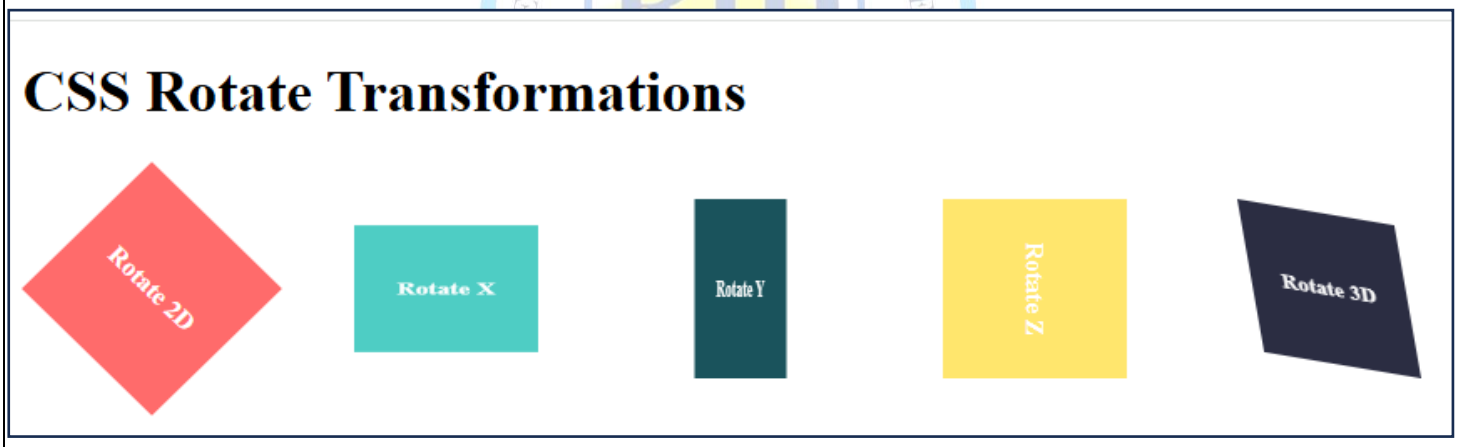
3. Rotate

- Rotates an element around a specified axis (or around the Z-axis by default in 2D).
- **rotate(angle):** Rotates an element around its center in the 2D plane.
- transform: rotate(45deg); /* Rotates 45 degrees clockwise */
- **rotateX(angle):** Rotates an element around the X-axis, creating a 3D "flip" effect.
- transform: rotateX(45deg); /* Rotates 45 degrees around the X-axis */
- **rotateY(angle)**: Rotates an element around the Y-axis, creating a 3D horizontal flip.
- transform: rotateY(60deg); /* Rotates 60 degrees around the Y-axis */
- **rotateZ(angle)**: Rotates an element around the Z-axis. This is similar to rotate(angle) in 2D.
- transform: rotateZ(90deg); /* Rotates 90 degrees around the Z-axis */
- **rotate3d(x, y, z, angle)**: Rotates an element in 3D space around an arbitrary axis defined by the vector (x, y, z).
- transform: rotate3d(1, 1, 0, 45deg); /* Rotates 45 degrees around the vector (1, 1, 0) */

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Rotate Transformations</title>
  <style>
    /* Container styling for better layout */
    .container {
      display: flex;
      gap: 20px;
      flex-wrap: wrap;
    } /* Base style for each box */
    .box {
      width: 100px;
      height: 100px;
      display: flex;
      align-items: center;
      justify-content: center;
      color: white;
      font-weight: bold;
      font-size: 14px;
      margin: 20px;
    } /* Rotate 2D */
    .rotate-2d {
      background-color: #ff6b6b;
      transform: rotate(45deg); /* 45 degrees clockwise */
    } /* Rotate around X-axis */
    .rotate-x {
      background-color: #4ecdc4;
      transform: rotateX(45deg); /* 45 degrees around X-axis */
    } /* Rotate around Y-axis */
    .rotate-y {
```

```
        background-color: #1a535c;
        transform: rotateY(60deg); /* 60 degrees around Y-axis */
      } /* Rotate around Z-axis */
      .rotate-z {
        background-color: #ffe66d;
        transform: rotateZ(90deg); /* 90 degrees around Z-axis */
      } /* Rotate 3D */
      .rotate-3d {
        background-color: #2b2d42;
        transform: rotate3d(1, 1, 0, 45deg); /* 45 degrees around vector (1, 1, 0) */
      }
    </style>
  </head>
  <body>
    <h1>CSS Rotate Transformations</h1>
    <div class="container">
      <div class="box rotate-2d">Rotate 2D</div>
      <div class="box rotate-x">Rotate X</div>
      <div class="box rotate-y">Rotate Y</div>
      <div class="box rotate-z">Rotate Z</div>
      <div class="box rotate-3d">Rotate 3D</div>
    </div>
  </body> </html>
```



4. Skew

Tilts an element along its X or Y axis, creating a slanted effect.

- **skew(xAngle, yAngle)**: Skews the element by xAngle along the X-axis and yAngle along the Y-axis.
  - transform: skew(30deg, 10deg); /* Skews 30 degrees on X, 10 degrees on Y */
- **skewX(angle)**: Skews the element along the X-axis only.
  - transform: skewX(20deg); /* Skews 20 degrees along the X-axis */
- **skewY(angle)**: Skews the element along the Y-axis only.
  - transform: skewY(15deg); /* Skews 15 degrees along the Y-axis */

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Skew Transformations</title>
  <style>
    .container {
      display: flex;
      gap: 20px;
      flex-wrap: wrap;
    }  .box { width: 120px;
                height: 100px;
                display: flex;
                align-items: center;
                justify-content: cente
                color: white;
                font-weight: bold;
                font-size: 14px;
                margin: 20px;
                border-radius: 8px; } .skew {
      background-color: #ff6b6b;
      transform: skew(30deg, 10deg);  }
    .skew-x {
      background-color: #4ecdc4;
      transform: skewX(20deg);  }
    .skew-y {
      background-color: #1a535c;
      transform: skewY(15deg); }
  </style>
</head>
<body>
  <h1>CSS Skew Transformations</h1>
  <div class="container">
    <div class="box skew">Skew (X, Y)</div>
    <div class="box skew-x">Skew X</div>
    <div class="box skew-y">Skew Y</div>
  </div> </body> </html>
```

5. Perspective

- Adds depth to 3D-transformed elements, giving a realistic view based on the viewer's position. The perspective property can be applied to a container element, affecting all its children with 3D transforms.
  - **perspective(value)**: Defines the depth of the 3D scene; smaller values give a more pronounced 3D effect.
- transform: perspective(500px) rotateY(45deg); /* Sets a 3D perspective at 500px distance */
- **Alternatively, perspective can be set as a property on a parent element**:

```
.container {
   perspective: 500px;
}
```

6. Transform Origin

- Defines the point around which transformations (rotate, scale, skew) are applied. By default, this point is the center of the element (50% 50%).
  - **transform-origin: x-axis y-axis z-axis;**: Sets the origin for transformations.
- transform-origin: top left; /* Sets the origin to the top-left corner */
- transform-origin: 100px 50px; /* Sets the origin to 100px right, 50px down */

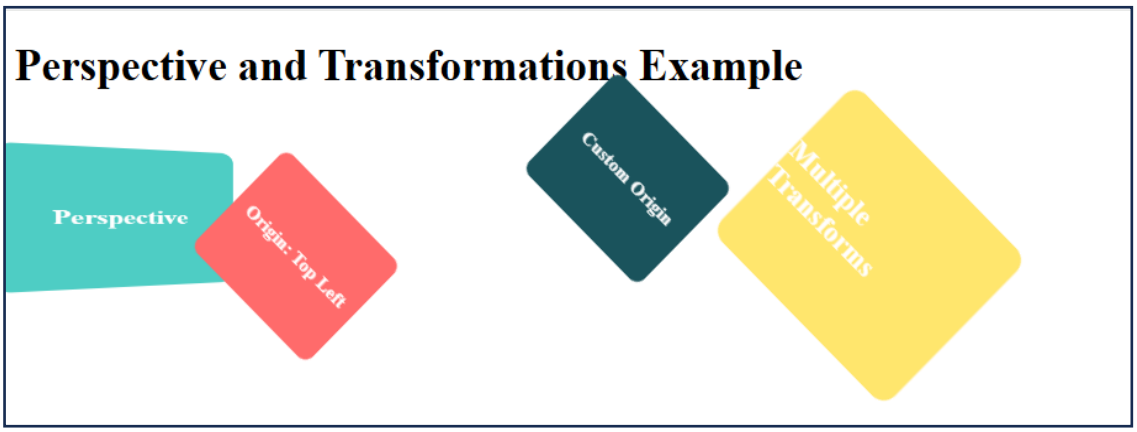
7. Combining Multiple Transformations

- You can apply multiple transformations at once by listing them with spaces between each transformation function. They are applied in order from left to right.
- transform: translate(10px, 20px) scale(1.5) rotate(45deg);

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Perspective and Transformations</title>
   <style>
      .container {
         display: flex;
         gap: 20px;
         flex-wrap: wrap;
         perspective: 500px;
      }
      .box {
         width: 120px;
         height: 100px;
         display: flex;
         align-items: center;
         justify-content: center;
         color: white;
         font-weight: bold;
         font-size: 14px;
         margin: 20px;
         border-radius: 8px;
         background-color: #4ecdc4;
```

```
    }
    .perspective {
      transform: rotateY(45deg);
    }
    .transform-origin {
      background-color: #ff6b6b;
      transform-origin: top left;
      transform: rotate(45deg);
    }
    .transform-origin-custom {
      background-color: #1a535c;
      transform-origin: 100px 50px;
      transform: rotate(45deg);
    }
    .multiple-transforms {
      background-color: #ffe66d;
      transform: translate(10px, 20px) scale(1.5) rotate(45deg);
    }
  </style>
</head>
<body>
  <h1>Perspective and Transformations Example</h1>
  <div class="container">
    <div class="box perspective">Perspective</div>
    <div class="box transform-origin">Origin: Top Left</div>
    <div class="box transform-origin-custom">Custom Origin</div>
    <div class="box multiple-transforms">Multiple Transforms</div>
  </div>
</body>
</html>
```



❖ Summary Table of Transform Functions:

❖ Transformation

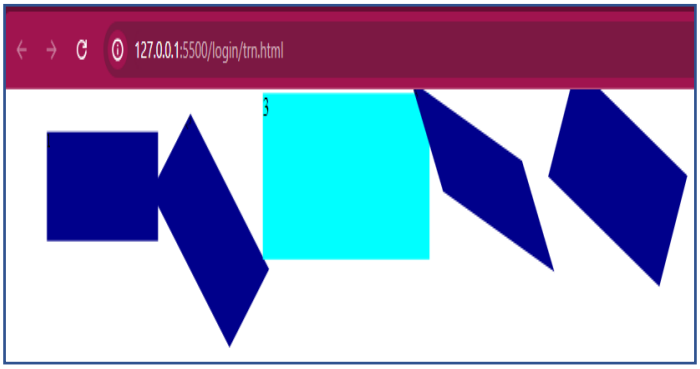
Function	Description	Example
translate(x, y)	Moves the element along X and Y axes	translate(50px, 20px)
translateX(x)	Moves the element along the X-axis only	translateX(50px)
translateY(y)	Moves the element along the Y-axis only	translateY(20px)
translateZ(z)	Moves the element along the Z-axis (3D)	translateZ(30px)
translate3d(x, y, z)	Moves the element in 3D space	translate3d(50px, 20px, 30px)
scale(x, y)	Scales the element in X and Y directions	scale(1.5, 0.8)
scaleX(x)	Scales the element along the X-axis only	scaleX(1.2)
scaleY(y)	Scales the element along the Y-axis only	scaleY(0.8)
scaleZ(z)	Scales the element along the Z-axis (3D)	scaleZ(1.2)
scale3d(x, y, z)	Scales the element in 3D space	scale3d(1.2, 0.8, 1.5)
rotate(angle)	Rotates the element in 2D space	rotate(45deg)
rotateX(angle)	Rotates the element around the X-axis (3D)	rotateX(45deg)
rotateY(angle)	Rotates the element around the Y-axis (3D)	rotateY(45deg)
rotateZ(angle)	Rotates the element around the Z-axis	rotateZ(45deg)
rotate3d(x, y, z, angle)	Rotates the element in 3D around an arbitrary axis	rotate3d(1, 1, 0, 45deg)
skew(xAngle, yAngle)	Skews the element along the X and Y axes	skew(20deg, 10deg)
skewX(angle)	Skews the element along the X-axis only	skewX(20deg)
skewY(angle)	Skews the element along the Y-axis only	skewY(15deg)
perspective(value)	Sets perspective for 3D transformations	perspective(500px)

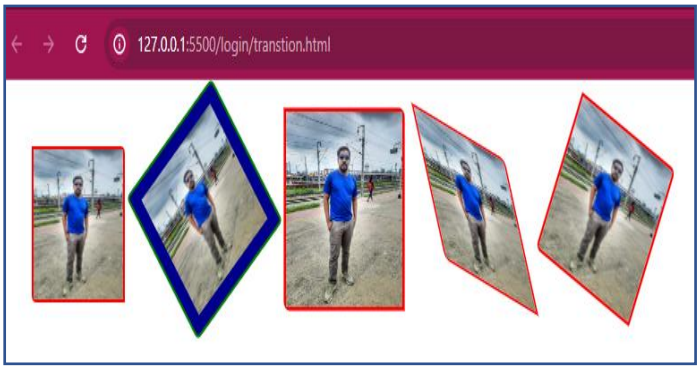
Example-1:

```
<!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>
.t{
   width: 200px;
   height: 100px;
   background-color: darkblue;
   margin: 20px;
   display: inline-block;
}
.b{ transform: translate(50px, 10px); }
.b1{
   transform: rotate(45deg);
   position: relative;
   top: 50px; }
.b2{ transform: scale(1.5);
   background: aqua; }
.b3{ transform: skew(30deg, 20deg);}
.b4{ transform: matrix(1,0.5, -0.5,1,0,0);}
   </style>
</head>
<body>
   <div class="t b">1</div>
   <div class="t b1">2</div>
   <div class="t b2">3</div>
   <div class="t b3">4</div>
   <div class="t b4">5</div>
</body>
</html>
```

Example-2:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
   <style>  .t{
         width: 200px;
         height: 200px;
         background-color: darkblue;
         margin: 40px;
         display: inline-block;
         color: white;
         border: 5px solid red;
         border-radius: 0 10px; }
   .t img{
      width: 100%;
      height: 100%; }
      .b1{
         transform: translate(30px,20px);
         border: 5px solid red; }
      .b2{
         transform: rotate(-40deg);
         border: 5px solid green;
         border-radius: 10px;
         padding: 20px; }
.b3{ transform: scale(1.3); }
.b4{ transform: skew(20deg,20deg); }
.b5{ transform: matrix(1,0.5,-0.5,1,0,0); }
   </style> </head>
<body>
   <div class="t b1"> <img src="r3.jpeg"> </div>
   <div class="t b2"> <img src="r3.jpeg"> </div>
   <div class="t b3"><img src="r3.jpeg"> </div>
   <div class="t b4"> <img src="r3.jpeg"> </div>
   <div class="t b5"> <img src="r3.jpeg"> </div>
</body> </html>
```





❖ 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;.

❖ Key Concepts of CSS Transitions:

1. Transition Properties

- CSS provides specific properties to control transitions. The four primary properties are:

➢ **transition-property:** Specifies the CSS property or properties to which the transition should be applied.

➢ **Example:** transition-property: background-color;
all can be used to apply the transition to all properties that change.
Example: transition-property: all;
transition-duration: Defines how long the transition should take to complete. It's specified in seconds (s) or milliseconds (ms).
Example: transition-duration: 0.5s; (half a second)

➢ **transition-timing-function:** Determines the pace of the transition over time, affecting the speed of change at different points.

❖ Common timing functions include:

- **ease:** Starts slow, speeds up, then slows down at the end (default).
- **linear:** Maintains a constant speed.
- **ease-in:** Starts slow and speeds up.
- **ease-out:** Starts fast and slows down.
- **ease-in-out:** Combines ease-in and ease-out for a gradual start and end.
- **cubic-bezier:** Allows for a custom pacing curve using four points.

➢ **Example:** transition-timing-function: ease-in;
transition-delay: Specifies a delay before the transition starts, allowing for staggered or sequential animations.

➢ **Example:** transition-delay: 0.2s; (waits for 0.2 seconds before starting)

2. Shorthand for Transitions

- Instead of using individual properties, you can use the shorthand transition property to define multiple transition settings in one line:
- transition: property duration timing-function delay;
- For example:
- transition: background-color 0.5s ease-in 0.2s;

Example of CSS transitions:

IMAGE TRANSFORMATIONS

```
<!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>
    .image-container {
      width: 300px;
      margin: 20px auto;
    }
    .image {
      width: 100%;
      max-width: 100%;
      height: auto;
      transition: transform 0.3s ease;
    }
    .image:hover {
      transform: scale(1.2);
    }
  </style>
</head>
<body>
  <h1>Image Transformation Example</h1>
  <div class="image-container">
    <img class="image" src="image/img4.jpg" alt="Sample Image">
    <p>image size will increase after hover</p>
  </div>
</body>
</html>
```

Image Transformation Example

image size will increase after hover

If you want to open a website served through VS Code's Live Server extension on a mobile device, follow these steps:

Prerequisites

- Your computer should have **Visual Studio Code** installed.
- You should have the **Live Server extension** installed in VS Code.
- Both your computer and mobile device must be on the **same Wi-Fi network**.

Step-by-Step Guide

1. Open Your Project in VS Code

- Launch Visual Studio Code on your computer.
- Open the folder or project you want to serve.

2. Start the Live Server

- In VS Code, right-click the index.html file (or the main HTML file of your project) and select **"Open with Live Server"**.
- Alternatively, click on **"Go Live"** at the bottom of the VS Code window if the Live Server extension is already set up.
- This will open a new tab in your default browser with a URL like http://127.0.0.1:5500 or http://localhost:5500.

3. Find Your Computer's Local IP Address

- To access the site from your mobile device, you need your computer's local IP address (since localhost only works on the same device).
- On **Windows**:
  - Open **Command Prompt** and type ipconfig.
  - Look for the line labeled **IPv4 Address** under your active network connection. It will look like 192.168.1.x or something similar.
- On **macOS**:
  - Open **System Preferences** > **Network**.
  - Select your Wi-Fi connection, and you'll see an IP address that looks similar to 192.168.1.x.

4. Modify the URL to Use the Local IP Address

- In your browser's address bar on your computer, replace localhost or 127.0.0.1 with your computer's local IP address (e.g., 192.168.1.x:5500).
- If the page loads, it confirms that the server is accessible over your local network.

5. Open the URL on Your Mobile Device

- On your mobile device, open the browser.
- Type the modified URL from Step 4 (http://192.168.1.x:5500) in the browser's address bar.
- Hit **Enter**, and the Live Server webpage should load on your mobile device.

6. Test Changes in Real Time

- Now, any changes you make to your code in VS Code will automatically refresh on both your computer and mobile device, thanks to Live Server's real-time reload feature.

Troubleshooting Tips

- **Firewall Settings**: Ensure that your firewall isn't blocking connections to the Live Server port (usually 5500).
- **Same Network**: Confirm that both your computer and mobile device are connected to the same Wi-Fi network.
- **Live Server Settings**: In VS Code, go to **Settings** > **Extensions** > **Live Server Config** and check Local IP or Host settings if there are connectivity issues

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.

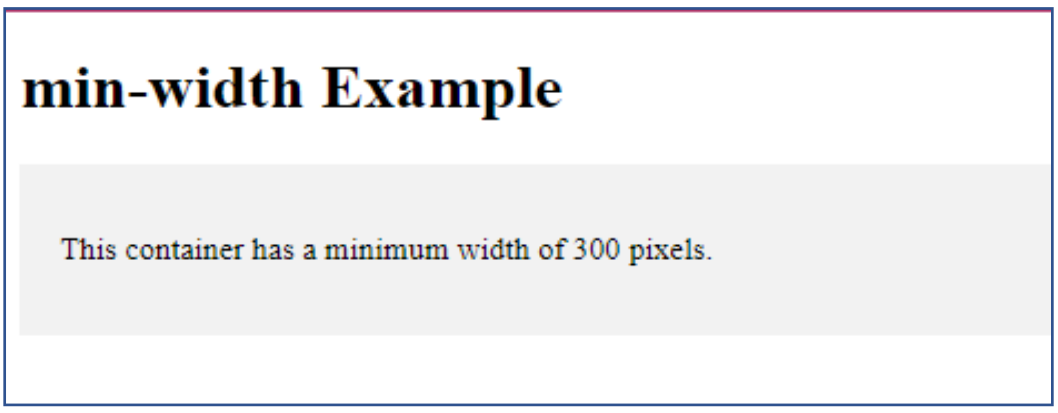
Why We Use min-width:

1. **Prevent Elements from Shrinking Too Much**: min-width ensures that important elements, like buttons or content boxes, don't become too narrow on smaller screens or in tighter spaces, which could make them hard to read or interact with.
2. **Maintain Layout Integrity**: It helps preserve a consistent design by preventing elements from collapsing, which can help avoid layout issues on different screen sizes.

Example:

- We have a container 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.

```
<!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>
    .container {
      min-width: 300px; /* Set the minimum width to 300 pixels */
      background-color: #f2f2f2;
      padding: 20px;
    }
  </style>
</head>
<body>
  <h1>min-width Example</h1>
  <div class="container">
    <p>This container has a minimum width of 300 pixels.</p>
  </div>
</body>
</html>
```



MAX WIDTH

- max-width property in CSS is used to set the maximum allowable width of an element. It restricts an element's width to the specified value, even if the content inside the element would typically require more space.

Syntax:

```
.container {
  max-width: 800px;
}
```

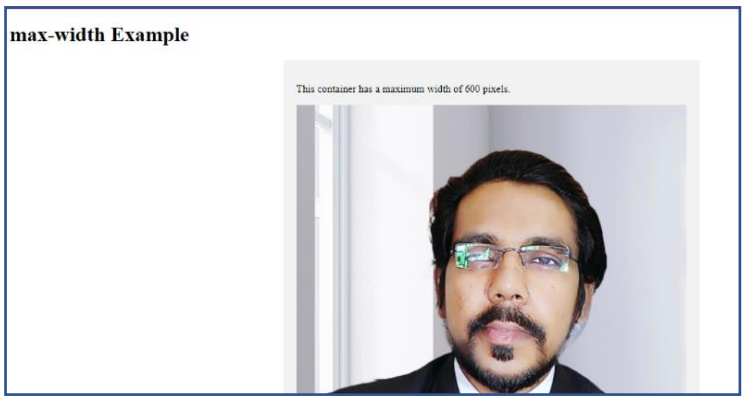
Why We Use max-width:

1. **Responsive Design**: It's particularly helpful for making websites look good on different screen sizes. By setting max-width, content doesn't stretch too much on larger screens, making it easier to read and visually balanced.
2. **Control Layout**: It helps maintain a comfortable reading width and aesthetic design by preventing elements from expanding excessively.

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>
    .container {
      max-width: 600px;
      background-color: #f2f2f2;
      padding: 20px;
      margin: 0 auto;
    }
  </style>
</head>
<body>
  <h1>max-width Example</h1>
  <div class="container">
    <p>This container has a maximum width of 600 pixels.</p>
    <img src="myimg.jpeg" alt="Example Image" style="max-width: 100%; height: auto;">
  </div>
</body>
</html>
```

```
.container {
    min-width: 200px;
    max-width: 800px;
    background-color: #f2f2f2;
    padding: 20px;
    margin: 0 auto;
}
```



MEDIA QUERY

CSS media queries allow you to apply styles based on conditions such as screen size, device type, orientation, and more. This makes them essential for building responsive websites that adapt to different screen sizes, from mobile phones to large desktop monitors.

➢ comprehensive guide to media queries, covering all key concepts:

1. Basic Syntax of Media Queries

- The basic syntax of a media query is as follows:

```
@media (condition) {
  /* CSS rules */
}
```

Example:

```
@media (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

This rule changes the background color to light blue when the screen width is 600 pixels or less.

2. Commonly Used Media Features

- Media features specify what aspects of the user's environment or device the media query should respond to. Here are some commonly used ones:

2.1 Width and Height

- **max-width**: Applies styles when the screen width is equal to or less than the specified value.
- **min-width**: Applies styles when the screen width is equal to or greater than the specified value.
- **max-height** and **min-height** work similarly for screen height.

Example:

```
@media (max-width: 768px) {
  /* Styles for tablets and smaller devices */
  .container {
    padding: 10px;
  }
}
```

2.2 Orientation

- **orientation: portrait**: Applies styles when the device is in portrait mode.
- **orientation: landscape**: Applies styles when the device is in landscape mode.

Example:

```
@media (orientation: landscape) {
  .sidebar {
    display: none;
  } }
```

2.3 Resolution

- **min-resolution** and **max-resolution**: Apply styles based on the pixel density of the device, often measured in DPI (dots per inch) or DPCM (dots per centimeter).
- This is useful for targeting high-resolution displays like Retina screens.

Example:

```
@media (min-resolution: 300dpi) {
  img {
    border: 2px solid #333;
  }
}
```

3. Logical Operators

Logical operators allow you to combine multiple media features:

- **and**: Combines multiple conditions; all conditions must be true.
- **, (comma)**: Acts as an "or" operator, where any condition being true will apply the styles.
- **not**: Negates the condition, so styles apply when the condition is false.
- **only**: Used to target specific media types (like screen or print) to exclude older browsers that don't support media queries.

Example of and:

```
@media (min-width: 600px) and (max-width: 1200px) {
  /* Applies only to screens between 600px and 1200px wide */
  .content {
    font-size: 18px;
  }
}
```

Example of comma (,):

```
@media (max-width: 600px), (orientation: portrait) {
  /* Applies when the screen width is 600px or less OR when in portrait mode */
  .navbar {
    font-size: 14px;
  }
}
```

4. Targeting Specific Devices

- CSS media queries allow you to target specific devices using media types like screen, print, or all:

- **screen**: Styles for screens like desktops, tablets, and phones.
- **print**: Styles for printed documents.
- **all**: Applies to all media types, combining screen and print.

Example:

```
@media print {
  body {
    background-color: white;
    color: black;
  } }
```

5. Range of Breakpoints

➢ Breakpoints are screen widths where the layout changes to better fit the screen. Common breakpoints are:

- Mobile: 320px - 600px
- Tablet: 600px - 900px
- Desktop: 900px - 1200px+

Using min-width and max-width helps ensure styles apply within these ranges.

Example:

Mobile devices

```
@media (max-width: 600px) {
  .container {
    padding: 5px;
  }
}
```

Tablets

```
@media (min-width: 601px) and (max-width: 900px) {
  .container {
    padding: 15px;
  }
}
```

Desktop

```
@media (min-width: 901px) {
  .container {
    padding: 30px;
  }
}
```

6. Using Media Queries with Flexbox and Grid

- CSS media queries work effectively with layout techniques like Flexbox and Grid, helping to adapt content layout at different screen sizes.

Example with Flexbox:

```
.container {
  display: flex;
  flex-direction: row;
}
@media (max-width: 768px) {
  .container {
    flex-direction: column; /* Switch to column layout on smaller screens */
  }
}
```

Example with Grid:

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
@media (max-width: 768px) {
  .grid-container {
    grid-template-columns: repeat(2, 1fr); /* 2 columns on smaller screens */
  }
}
@media (max-width: 480px) {
  .grid-container {
    grid-template-columns: 1fr; /* 1 column on very small screens */
  }}
```

7. Responsive Font Sizes with vw and Media Queries

- Using media queries with vw (viewport width) units can help set responsive font sizes, so the text scales smoothly across different screen sizes.

Example:

```
body {
  font-size: 2vw; /* Font size scales with viewport width */
}
@media (max-width: 768px) {
  body {
    font-size: 4vw; /* Larger font size for smaller screens */
  }
}
```

8. Advanced Example: Combining Multiple Concepts

- This example combines several media query concepts, including device orientation, screen width breakpoints, and layout adjustments.

Base styles

```
.container {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 20px;
}
```

Large desktops

```
@media (min-width: 1200px) {
  .container {
    grid-template-columns: repeat(4, 1fr);
  }
}
```

Tablets and small desktops

```
@media (min-width: 768px) and (max-width: 1199px) {
  .container {
    grid-template-columns: repeat(2, 1fr);
  }
}
```

Mobile phones

```
@media (max-width: 767px) {
  .container {
    grid-template-columns: 1fr;
  }
}
```

Portrait mode on phones

```
@media (max-width: 600px) and (orientation: portrait) {
  .container {
    background-color: lightgrey; /* Changes background color in portrait mode */
  }
}
```

Summary of Key Media Query Concepts:

1. **Media Types**: Control where the styles apply, e.g., screen, print.
2. **Media Features**: Specify the conditions, e.g., min-width, max-width, orientation.
3. **Logical Operators**: Combine multiple conditions with and, not, and comma ,.
4. **Breakpoints**: Screen size thresholds to adjust layouts.
5. **Responsive Design**: Combine media queries with layout techniques like Flexbox, Grid, and vw units.

Example: Apply styles for screens with a minimum resolution of 300dpi.

```
@media (min-resolution: 300dpi) {
  body {
    background-image: url('high-res-bg.jpg');
  }
}
```

❖ Device Type:

Example: Apply styles specifically for handheld devices.

```
@media (handheld) {
  body {
    font-size: 0.8em;
  }
}
```

❖ Multiple Conditions:

➢ Combine conditions using and, , (comma for OR), and not.

```
@media (min-width: 600px) and (max-width: 1024px) {
  body {
    background-color: yellow;
  }
}
@media (min-width: 768px), (orientation: landscape) {
  body {
    font-size: 1.5em;
  }
}
```

Example: Responsive Design with Media Queries

Base styles

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  color: #333;
}
header {
  background-color: #4CAF50;
  color: white;
  text-align: center;
  padding: 1em;
```

```
}
main {
  padding: 1em;
}
footer {
  background-color: #222;
  color: white;
  text-align: center;
  padding: 1em;
}
```

Media Query for tablets and smaller devices

```
@media (max-width: 768px) {
  body {
    font-size: 14px;
  }
  header {
    padding: 0.5em;
  }
  main {
    padding: 0.5em;
  }
}
```

Media Query for mobile devices

```
@media (max-width: 480px) {
  body {
    font-size: 12px;
  }
  header {
    font-size: 1.2em;
  }
  main {
    font-size: 0.9em;
  }}/* Media Query for large screens */
@media (min-width: 1024px) {
  body {
    font-size: 18px;
  }
  header {
    font-size: 2em;
  }
  main {
    padding: 2em;  }}
```

Tips for Using Media Queries

Mobile-First Approach: Start with styles for mobile devices and use min-width queries to add styles for larger screens.

Minimize CSS Complexity: Keep media queries organized and avoid overly complicated conditions.

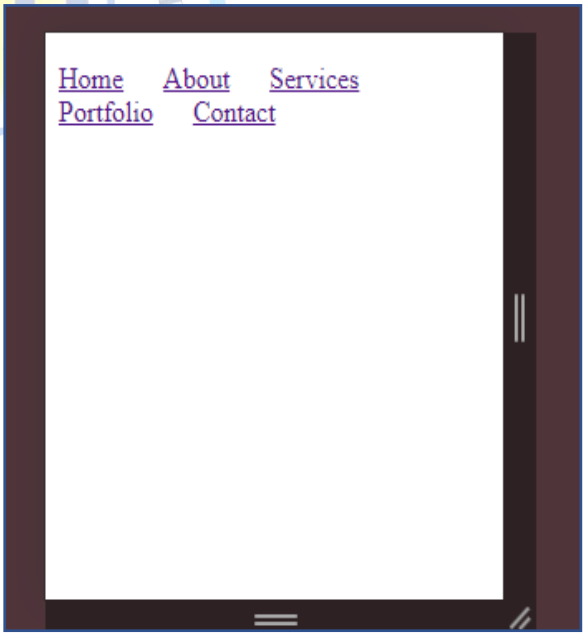
Test Across Devices: Always test your media queries on real devices or using browser tools to ensure they work as expected.

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>
       <li><a href="#">Portfolio</a></li>
      <li><a href="#">Contact</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>
```



Background Color Change with Media Query Using and Operator Using or Operator (comma in CSS)

index.html > html > head > style > {} @media (min-width: 601px)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Media Query Example</title>
    <style>
        .box {
            background-color: lightblue;
            color: black;
            text-align: center;
            padding: 40px;
            font-size: 20px;
            border-radius: 10px;
        }/* When screen width is 600px or less (mobile view) */
        @media (max-width: 600px) {
            .box {
                background-color: lightcoral;
                color: white;
                font-size: 18px;
                text-align: left;
            } }/* When screen width is 601px or more (desktop view) */
        @media (min-width: 601px) {
            .box {
                background-color: lightgreen;
                text-align: center;
                font-size: 22px;}}
</style></head><body>
    <h1>Responsive Box Example</h1>
    <div class="box">
        Resize the browser window to see the background and text style change!
    </div>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Media Query with OR Example</title>
    <style>
        .box {
            background-color: lightblue;
            color: black;
            text-align: center;
            padding: 40px;
            font-size: 20px;
            border-radius: 10px;
        }

        /* Apply when screen width is less than 500px OR more than 1000px */
        @media (max-width: 500px), (min-width: 1000px) {
            .box {
                background-color: violet;
                color: white;
                font-size: 22px;
            }
        }
    </style>
</head>
<body>
    <h1>Media Query with OR Operator</h1>
    <div class="box">
        This style applies when width is below 500px OR above 1000px.
    </div>
</body>
</html>
```

index.html > html > head >

Home About Services Portfolio Contact

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Responsive Navigation Bar</title>
  <style> /* Basic styling */
    body {margin: 0;font-family: Arial, sans-serif;}
    nav {
      background-color: #1e90ff; /* Blue header */
      padding: 10px 20px;
    }
    .nav-links {
      display: flex;
      justify-content: flex-end;
      align-items: center;
      list-style: none;
      margin: 0;padding: 0;}
    .nav-links li { margin-left: 20px;}
    .nav-links a {
      text-decoration: none;
      color: white;
      font-weight: bold;
      transition: color 0.3s ease;}
    .nav-links a:hover { color: #ffd700; /* Gold on hover */ }
    .logo {
      float: left;
      color: white;
      font-size: 22px;
      font-weight: bold;
      text-transform: uppercase;
    }
```

```
<body>
  <nav>
    <div class="nav-container">
      <div class="logo">RID Bharat</div>
      <ul class="nav-links">
        <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>
    </div>
  </nav>
  <section style="padding: 40px; text-align: center;">
    <h2>Welcome to RID Bharat</h2>
    <p>Resize the window to see the navigation bar adjust for desktop, tablet, and mobile screens.</p>
  </section>

</body>
</html>
```

index.html > html > body

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Media Query with AND Example</title>
    <style>
        .box {
            background-color: lightgray;
            color: black;
            text-align: center;
            padding: 40px;
            font-size: 20px;
            border-radius: 10px;
        }/* Apply when screen width is between 601px and 900px */
        @media (min-width: 601px) and (max-width: 900px) {
            .box {
                background-color: lightseagreen;
                color: white;
                font-size: 22px;
            }} /* For screens larger than 900px */
        @media (min-width: 901px) {
            .box {background-color: lightgreen;}
        } /* For screens smaller than or equal to 600px */
        @media (max-width: 600px) {
            .box { background-color: lightcoral;
                   color: white;}}
    </style>
</head>
<body>
    <h1>Media Query with AND Operator</h1>
    <div class="box">
        Resize the window – color changes only when width is between 601px and 900px.
    </div>
</body></html>
```

```
        .logo {
            font-size: 22px;
            font-weight: bold;
            text-transform: uppercase;
        }
        /* Clear float */
        .nav-container::after {
            content: "";
            display: block;
            clear: both;
        }
        /* ===== Tablet View (601px - 900px) ===== */
        @media (max-width: 900px) and (min-width: 601px) {
            .nav-links {justify-content: center;}
            .logo {
                display: block;
                text-align: center;
                float: none;
                margin-bottom: 8px;
            }}/* ===== Mobile View (≤600px) ===== */
        @media (max-width: 600px) {
            .nav-links {
                flex-direction: column;
                align-items: center;}
            .nav-links li { margin: 10px 0;}
            .logo {
                display: block;
                text-align: center;
                float: none;
                margin-bottom: 10px;}}
    </style>
</head>
```

How we you can Develop a responsive website

- Developing a responsive website with CSS involves making the layout adapt smoothly to different screen sizes and devices. This is typically achieved through several techniques, such as using flexible layouts, media queries, responsive units, and flexible images.

Key Techniques for Responsive Design in CSS:

1. Use of Flexible Layouts (Fluid Grids)

- Instead of fixed widths, use percentages or relative units (like em or rem) for widths of elements so they can adjust based on the screen size.
- **Example:**

```
.container {
  width: 80%; /* Adjusts width based on the screen size */
  padding: 1rem;
}
```

2. Media Queries

- Media queries allow you to apply specific styles depending on the screen's width, height, orientation, or resolution. They're essential for changing layouts on different devices, such as desktops, tablets, and mobiles.
- Example:

```
/* Styles for small screens */
@media (max-width: 600px) {
  .container {
    width: 100%; /* Full width on small screens */
  }
}
/* Styles for medium screens */
@media (min-width: 601px) and (max-width: 1024px) {
  .container {
    width: 80%;
  }
}
```

3. Responsive Units

- Use units like percentages, vw (viewport width), and vh (viewport height) to create flexible layouts.
- **Example:**

```
.header {
  height: 10vh; /* 10% of viewport height */
}
```

4. Responsive Typography

- Font sizes can be scaled using em, rem, or even viewport-based units (vw and vh). This ensures that text size scales appropriately across devices.
- **Example:**

```
body {
  font-size: 1.2vw; /* Text size adapts based on viewport width */
}
```

5. Flexible Images and Media

- Use the max-width property to make images scale within their containers, so they resize based on the screen size.
- **Example:**

```
img {
  max-width: 100%;
  height: auto; /* Maintains the aspect ratio */
}
```

6. CSS Flexbox and Grid Layouts

- Flexbox and Grid are CSS layout systems that are inherently flexible and are excellent for creating responsive designs. They allow elements to adjust dynamically based on available space.
- **Example (Flexbox):**

```
.container {
  display: flex;
  flex-wrap: wrap; /* Allows items to wrap onto the next line */
}
```

- **Example (Grid):**

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr
  gap: 20px;
}
```

CSS UNITS

- CSS offers a variety of units that are used to define sizes for elements, spacing, font sizes, and other properties. These units fall under two main categories: **absolute units** (fixed sizes) and **relative units** (sizes that adjust based on other factors, like screen size or parent elements). Here's a rundown of each with concepts and examples:

1. Absolute Units:

- These units are fixed and do not scale based on the parent element or screen size.

1. px (Pixels):

- A pixel is a single point on the screen, commonly used for fixed dimensions.

Example:

```
.box {
  width: 200px;
  height: 150px;
}
```

2. cm (Centimeters) / mm (Millimeters):

- Centimeters and millimeters are rarely used in web design because they're based on physical measurements, but they can be useful for print styles.

Example:

```
.print-section {
  width: 10cm;
}
```

3. in (Inches)

- Like centimeters, inches are rarely used on screens but are often applied for print designs.

Example:

```
.print-image {
  height: 2in;
}
```

4. pt (Points)

- Points are commonly used in typography, particularly for print. 1 pt = 1/72 inch.

Example:

```
.text {
  font-size: 12pt;
}
```

5. pc (Picas)

- One pica is equal to 12 points. It's primarily used in print but not common in web design.

Example:

```
.page {
  margin: 2pc;
}
```

2. Relative Units

- Relative units scale depending on their context, which makes them useful for responsive designs.

1. % (Percentage)

- Defines size as a percentage of the parent element, making it flexible.

Example:

```
.container {
  width: 80%; /* 80% of the parent container */
}
```

2. em

- Relative to the font size of the element's parent. For example, if the parent font size is 16px, 1em would be 16px, 2em would be 32px, and so on.

Example:

```
.text {
  font-size: 1.5em; /* 1.5 times the font size of the parent */
}
```

3. rem (Root em)

- Similar to em, but relative to the root (html) font size rather than the parent element. This provides consistency across the page.
- Example:

```
css
Copy code
.heading {
  font-size: 2rem; /* 2 times the root font size */
}
```

4. vw (Viewport Width)

- Relative to 1% of the viewport's width. 1vw is 1% of the viewport's total width.

Example:

```
.banner {
  width: 50vw; /* 50% of the viewport's width */
}
```

5. vh (Viewport Height)

- Relative to 1% of the viewport's height. 1vh is 1% of the viewport's total height.

Example:

```
.full-screen {
  height: 100vh; /* 100% of the viewport's height */
}
```

6. vmin and vmax

- vmin is relative to the smaller dimension (width or height) of the viewport, while vmax is relative to the larger dimension.

Example:

```
.square {
  width: 50vmin; /* 50% of the smaller viewport dimension */
  height: 50vmin;
}
```

7. **ch**
   - Represents the width of the "0" character in the element's font, useful for creating columns of text or fixed-width layouts based on character size.

Example:

```
.code-block {
  width: 60ch; /* Width for approximately 60 characters */
}
```

8. **ex**
   - Relative to the x-height (height of lowercase "x") of the element's font. Used less often but can help with typography alignment.

Example:

```
.small-text {
  line-height: 1.5ex;
}
```

9. **lh (Line Height)**
   - Relative to the line height of the element, commonly used for setting vertical spacing in typography.

Example:

```
.paragraph {
  margin-bottom: 2lh;
}
```

Practical Tips for Choosing Units

- **Pixels (px)** for precise and small elements.
- **Percentages (%)** for flexible layouts, especially widths.
- **em and rem** for font sizes and spacing to ensure scalability.
- **Viewport units (vw, vh)** for full-screen layouts or responsiveness.
- **ch** for fixed-width elements based on character count.

CSS Interview Question and Answer

1. What is CSS?

Answer: CSS (Cascading Style Sheets) is a stylesheet language used to describe the presentation of a document written in HTML or XML.

2. What is the box model in CSS?

Answer: The CSS box model describes the rectangular boxes generated for elements, consisting of margins, borders, padding, and the content area.

3. What is the difference between class and id selectors?

Answer: class selectors can be used multiple times within a document, while id selectors are unique to a single element.

4. What is the difference between absolute, relative, fixed, and static positioning?

Answer:

absolute: Positioned relative to the nearest positioned ancestor.
relative: Positioned relative to its normal position.
fixed: Positioned relative to the viewport.
static: Default position, element follows the normal document flow.

5. What are CSS pseudo-classes?

Answer: Pseudo-classes are used to define the special state of an element, e.g., :hover, :first-child.

6. What are CSS pseudo-elements?

Answer: Pseudo-elements are used to style specific parts of an element, e.g., ::before, ::after.

7. How can you include CSS in an HTML document?

Answer: Through inline styles, internal stylesheets within the <style> tag, and external stylesheets using the <link> tag.

8. What is a CSS preprocessor?

Answer: A CSS preprocessor is a scripting language that extends CSS and compiles it into regular CSS. Examples include Sass, LESS, and Stylus.

9. What is the use of the z-index property?

Answer: The z-index property specifies the stack order of elements; elements with a higher z-index appear in front of those with a lower z-index.

10. How can you center a block element horizontally in CSS?

Answer: By setting margin: auto; and a width for the element.

11. What is Flexbox?

Answer: Flexbox (Flexible Box Layout) is a layout model that allows items within a container to be automatically arranged in a predictable way to accommodate different screen sizes.

12. How do you create a CSS grid layout?

Answer: By using the display: grid; property on a container and defining rows and columns with grid-template-rows and grid-template-columns.

13. What is the difference between margin and padding?

Answer: Margin is the space outside an element's border, while padding is the space inside the element's border.

14. How can you make a website responsive?

Answer: By using media queries, flexible grid layouts, flexible images, and employing techniques like responsive units (%, em, rem, vw, vh).

15. What are media queries?

Answer: Media queries are a feature of CSS that allows content to adapt to different screen sizes and resolutions using the @media rule.

16. How do you apply styles for a specific element when it is being hovered over?
Answer: By using the :hover pseudo-class, e.g., element:hover { /* styles */ }.
17. What is a CSS transition?
Answer: CSS transitions allow changes in CSS property values to occur smoothly over a specified duration.
18. How can you use CSS to hide an element?
Answer: By using display: none; or visibility: hidden;.
19. What are CSS animations?
Answer: CSS animations enable the gradual change of CSS properties using keyframes, defined with the @keyframes rule.
20. What does the inherit value do in CSS?
Answer: The inherit value causes a property to take the same computed value as the property for the element's parent.
21. How can you make a list horizontal using CSS?
Answer: By setting the list items to display: inline; or using display: flex; on the parent container.
22. What is the purpose of the calc() function in CSS?
Answer: The calc() function allows you to perform calculations to determine CSS property values dynamically.
23. How do you apply a style to multiple classes?
Answer: By separating the classes with a comma in the selector, e.g., .class1, .class2 { /* styles */ }.
24. What is the difference between inline and block elements?
Answer: inline elements do not start on a new line and only take up as much width as necessary. block elements start on a new line and take up the full width available.
25. How can you create a dropdown menu using CSS?
Answer: By using position: relative; on the parent and position: absolute; on the child, along with :hover to toggle visibility.
26. What is the ::before pseudo-element used for?
Answer: The ::before pseudo-element is used to insert content before the content of an element.
27. How can you make text italic in CSS?
Answer: By using the font-style: italic; property.
28. What does box-sizing: border-box; do?
Answer: It makes the width and height properties include content, padding, and border, but not the margin.
29. What is the em unit in CSS?
Answer: The em unit is a relative unit that is based on the font size of the element. 1em is equal to the current font size.
30. How can you clear floated elements?
Answer: By using the clear property on a following element or using the clearfix hack.

What is RID Organization (RID संस्था क्या है)?

- **RID Organization** यानि **Research, Innovation and Discovery Organization एक** संस्था हैं जो TWF (TWKSAA WELFARE FOUNDATION) **NGO** द्वारा RUN किया जाता है | जिसका मुख्य उदेश्य हैं आने वाले समय में सबसे पहले **NEW (RID, PMS & TLR)** की खोज, प्रकाशन एवं उपयोग भारत की इस पावन धरती से भारतीय संस्कृति, सभ्यता एवं भाषा में ही हो |
- देश, समाज, एवं लोगों की समस्याओं का समाधान **NEW (RID, PMS & TLR)** के माध्यम से किया जाये इसके लिए ही **इस RID Organization** की स्थपना 30.09.2023 किया गया है | जो TWF द्वारा संचालित किया जाता है |
- TWF (TWKSAA WELFARE FOUNDATION) NGO की स्थपना 26-10-2020 में बिहार की पावन धरती सासाराम में Er. RAJESH PRASAD एवं Er. SUNIL KUMAR द्वारा किया गया था जो की भारत सरकार द्वारा मान्यता प्राप्त संस्था हैं |
- Research, Innovation & Discovery में रूचि रखने वाले आप सभी विधार्थियों, शिक्षकों एवं बुधीजिवियो से मैं आवाहनं करता हूँ की आप सभी **इस RID संस्था** से जुड़ें एवं अपने बुध्दि, विवेक एवं प्रतिभा से दुनियां को कुछ नई **(RID, PMS & TLR)** की खोजकर, बनाकर एवं अपनाकर लोगों की समस्याओं का समाधान करें |

MISSION, VISSION & MOTIVE OF "RID ORGANIZATION"	
मिशन	हर एक ONE भारत के संग
विजन	TALENT WORLD KA SHRESHTM AB AAYEGA भारत में और भारत का TALENT भारत में
मकसद	NEW (RID, PMS, TLR)

MOTIVE OF RID ORGANIZATION NEW (RID, PMS, TLR)		
NEW (RID)		
R	I	D
Research	Innovation	Discovery
NEW (TLR)		
T	L	R
Technology, Theory, Technique	Law	Rule
NEW (PMS)		
P	M	S
Product, Project, Production	Machine	Service

RID रीड संस्था की मिशन, विजन एवं मकसद को सार्थक हमें बनाना हैं |
भारत के वर्चस्व को हर कोने में फैलना हैं |
कर के नया कार्य एक बदलाव समाज में लाना हैं |
रीड संस्था की कार्य-सिध्दांतों से ही, हमें अपनी पहचान बनाना हैं |

Er. Rajesh Prasad (B.E, M.E)
Founder:
TWF & RID Organization

CSS के इस E-Book में अगर मिलती त्रुटी मिलती है तो कृपया हमें सूचित करें |

WhatsApp's No: 9892782728 or

Email Id: ridorg.in@gmail.com

|| धन्यवाद ||