Introduction to CSS

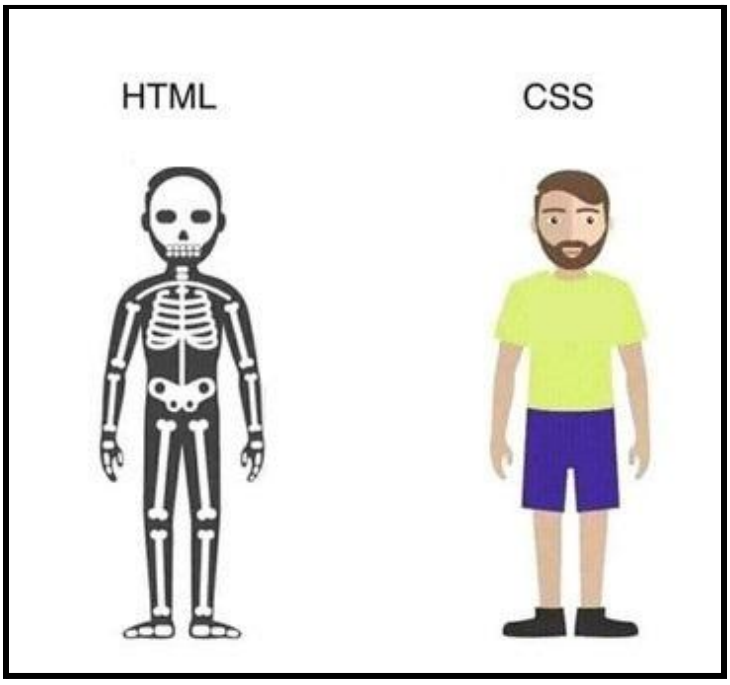
Topics Covered:

- What is CSS?
- Purpose of CSS.
- Syntax of CSS.
- Comments in CSS.
- What are the different ways to include CSS?
    - Inline CSS.
    - Internal CSS.
    - External CSS.

Topics in Detail:

CSS:

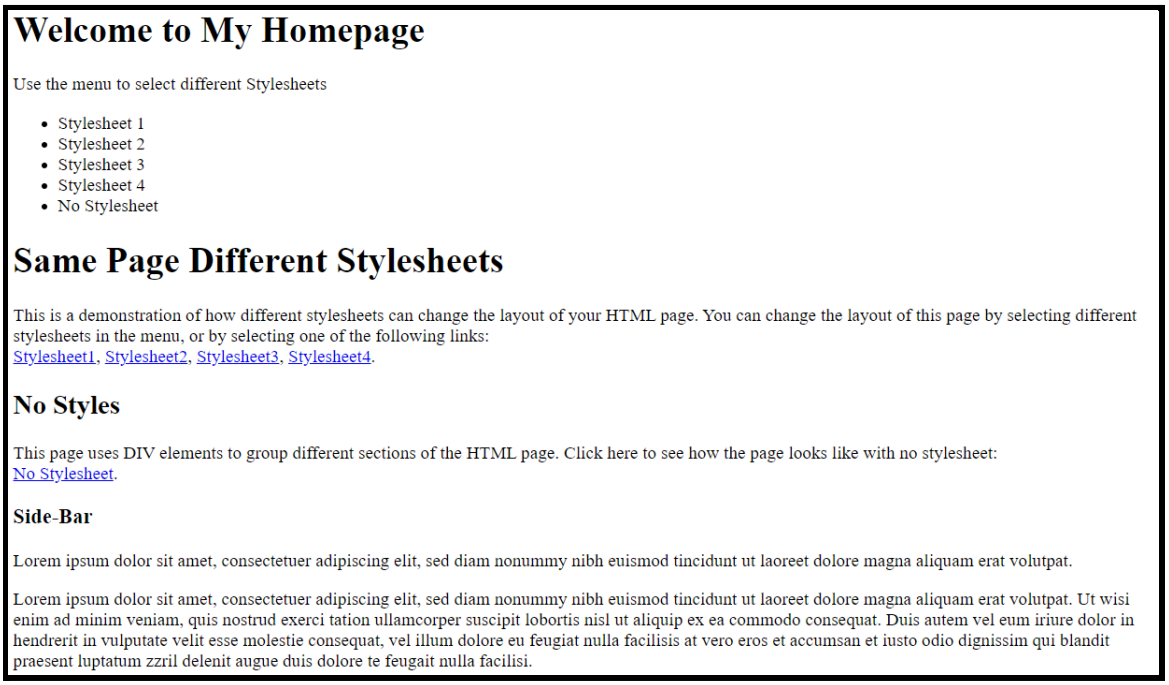
- CSS stands for **C**ascading **S**tyle **S**heet.
- CSS is used to give styles to HTML elements.
- CSS is used to control the style of the web document in a simple and easy way.
- CSS describes how the HTML elements have to be displayed on the browser.
- When CSS is applied externally the same styles can be applied to multiple elements and pages. So it saves work and effort.



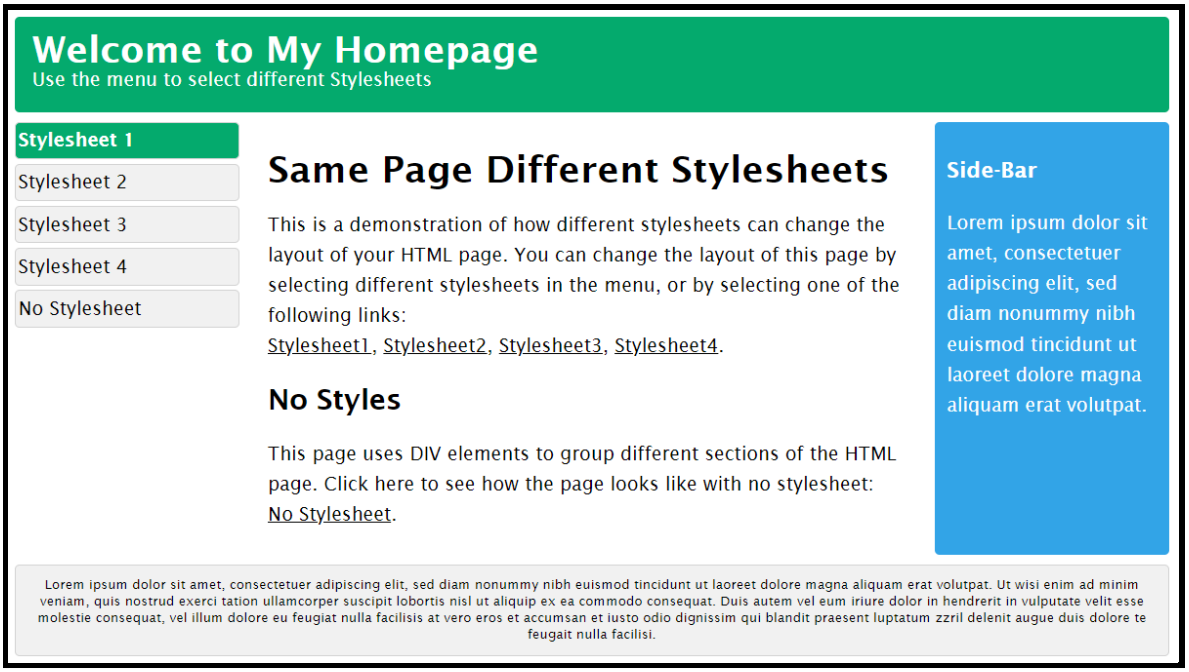
HTML just gives structure to your web pages, where CSS styles the elements, defines layout and makes the web page look attractive.

Purpose of CSS:

- The purpose of CSS is to make web pages presentable.
- Example: Web page without any styles applied.



- Same web page when styles are applied.

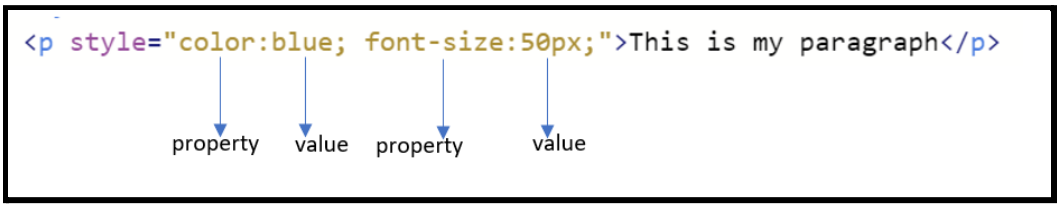


Applications of CSS:

- Creating stunning web pages and websites.
- Saves time by reusing the styles.
- Easy to maintain.

Basic CSS Syntax:

Inline style:



Internal CSS:

```
<head>
  <title>CSS</title>
  <style>
    p {
      color: red;
      text-align: center;
    }
  </style>
</head>
<body>
  <p>This is my paragraph</p>
</body>
```

- Each declaration has CSS properties and the value, separated by the semi colon.
- In the first declaration **color** is the property and **blue** is the value.
- In the second declaration **font-size** is the property and **50px** is the value.

Comments in CSS:

- Comments will not be displayed in the browser, but they help to understand the code better.
- Comments cannot be added in the inline styles of CSS. But they can be added in internal CSS and external CSS.
- Internal CSS:
  - Comments have to be added inside the <style> </style> tag.
  - CSS comment starts with **/*** and ends with ***/**
  - **Example:**

```
<style>
/* style applied for all paragraph */
  p {
    color: red; /* text color */
    text-align: center; /* text alignment */
  }
</style>
```

- Comments can be added in the same way in the external style sheet.

CSS Inclusion:

There are three ways to include styles to your HTML elements. They are:

- Inline CSS or Inline styles.
- Internal CSS or Embedded CSS.
- External CSS or External style sheet.

Inline CSS:

- The inline styling is done with help of the **style** attribute.
- The **style** attribute can be added to the HTML element to which the style has to be applied. And styles can be specified as the value of the **style** attribute.
- Example:

```
<p style="color:blue; font-size:50px;">This is my paragraph</p>
```

property value property value

- The value of style attribute is the combination of style declarations separated by semicolons.
- Output of the above code:

This is my paragraph

Internal CSS:

- In order to reuse the same styles within the same page or to apply same styles for multiple elements styles can be given inside the <style> </style> tags inside the <head> </head> part of the document.
- Example:

```
<head>
  <title>CSS</title>
  <style>
    p {
      color: red;
      text-align: center;
    }
  </style>
</head>
<body>
  <p>This is my paragraph</p>
</body>
```

- Output:

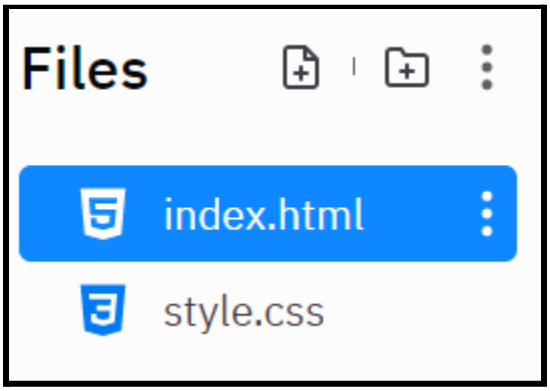
This is my paragraph

External CSS:

- The external style sheet is generally used when the same style has to be applied in multiple pages.
- The syntax will be similar to the internal CSS/ Embedded CSS, but the styles will be included from the external sheet which will be connected to the HTML file by linking the external style sheet.
- The external style sheet can be linked with the help of **<link>** tag.
- The external style sheet included should have the extension as **.css**.
- Example:

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

- Folder files:



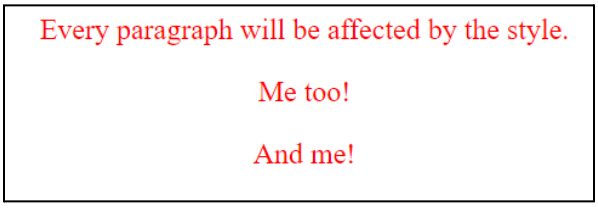
- Attributes and their purpose:

Attribute	Value	Description
type	text/css	Specifies the style language.
href	URL	Specifies the URL/ path link of the external style sheet.
rel	stylesheet/ icon	Specifies the relationship between the current document and linked document. Optional attribute.

CSS Syntax and Selectors - Practice Code

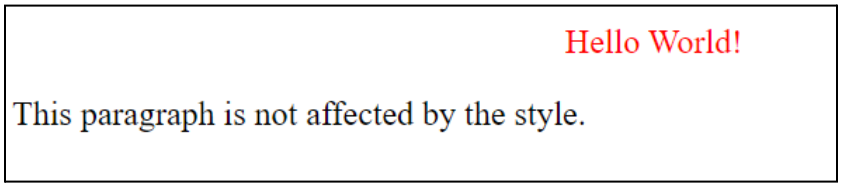
Problem statement 1: Element selector

Create a simple page with <p> tag, using the element selector add text color and text alignment to the <p> tag by understanding the purpose of the element selector.



Problem statement II: ID selector

Create a simple page with two <p> tags, one with id and one without id, add style to the <p> tag with id using id selector.



Problem statement III: Class selector

Create a simple web page with <h1> and <p> tags, add the same class name to both tags. Using the class selector add common styles to both tags.

Problem statement IV: Universal Selector

Create a web page with multiple HTML elements, add common styles to all the elements by using the universal selector.

Solution - Problem Statement I : Element Selector

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p>Every paragraph will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

Solution - Problem Statement II : ID Selector

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>

</body>
</html>
```

Solution - Problem Statement III : Class Selector

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  color: red;
}
</style>
</head>
```

```
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>

</body>
</html>
```

Solution - Problem Statement IV : Universal Selector

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>

<h1>Hello world!</h1>

<p>Every element on the page will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

CSS Background

Topics covered:

- What is CSS Background?
- What is Background-color?
- What is Background-image?
- What is Background-repeat?
- What is Background-attachment?
- What is Background-position?
- What is Background shorthand property?
- What is Background-size?

Topics in Detail:

CSS Background:

To add background effects to HTML elements background property is used. The background effect may be:

1. background-color
2. background-image
3. background-repeat
4. background-attachment
5. background-position

Background-color:

- The property background-color is used to set the background color of an HTML element.
- Syntax: **background-color**: color name;
- Example: `background-color: lightblue;`
- The value can be any valid color names, color codes (Hex values/rgb values)
- The transparency of the background color can be adjusted using the **opacity** property.
- The value of opacity can vary from 0.0 to 1.0.
- The lower value will have more transparency.
- Code and Style:

```html
<div class="bgcolor">
  <h1>CSS Background Color</h1>
  <p>The property background-color is used to set the background color of an
  HTML element.</p>
</div>
```

```css
.bgcolor{
  background-color: pink;
  opacity: 0.5;
}
```

- Output:

Background-Image:

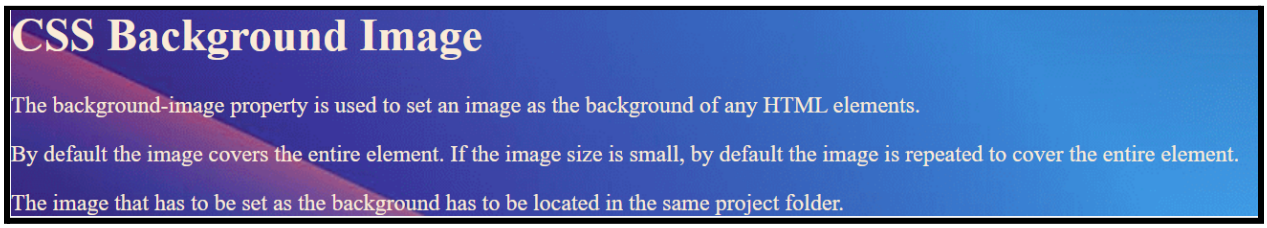
- The background-image property is used to set an image as the background of any HTML elements.
- By default the image covers the entire element. If the image size is small, by default the image is repeated to cover the entire element.
- The image that has to be set as the background has to be located in the same project folder.
- Syntax: background-image: url("image path");
- Code:

```
<div class="bgimage">
  <h1>CSS Background Image</h1>
  <p>The background-image property is used to set an image as the background of any HTML elements.</p>
  <p>By default the image covers the entire element. If the image size is small, by default the image is
  repeated to cover the entire element.</p>
  <p>The image that has to be set as the background has to be located in the same project folder.</p>
</div>
```

- Style:

```
.bgimage{
  background-image: url("image.png");
  color: antiquewhite
}
```

- Output:



- The background image chosen should match the text color.

Background-repeat:

- The background image property repeats the images horizontally and vertically if the image is smaller than the HTML element size. Sometimes it may look odd.
- Example:

- By using background-repeat property we can specify whether the background image should repeat horizontally/ repeat vertically/ should not repeat.
- Repeat horizontally: `background-repeat: repeat-x;`
- Repeat vertically: `background-repeat: repeat-y;`
- Should not repeat: `background-repeat: no-repeat;`

- The position of the image can be specified with the help of background-position property.
- Example: `background-position: right top;`
- Code:

```
<div class="bgrepeat">
  <h1>Background repeat</h1>
  <p>By using background-repeat property we can specify whether the background image should repeat
  horizontally/ repeat vertically/ should not repeat.</p>
</div>
```

- Style:

```
.bgrepeat{
  background-image: url("image2.jpg");
  background-repeat: no-repeat;
  background-position: right top;
  color: burlywood;
}
```

- Output:

Background-attachment:

- The background-attachment property specifies whether the background image attached should be scrolled along with the text or it should be in fixed size.
- Example:
  - `background-attachment: fixed;`
  - `background-attachment: scroll;`

Background-shorthand property:

- To shorten the code, all the background properties can be included in one single property background.
- The background is a shorthand property.
- Expanded code and Shorthand property:

```
body{
  background-color: pink;
  background-image: url("image.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

```
body{
  background: pink url("image.png") no-repeat right top;
}
```

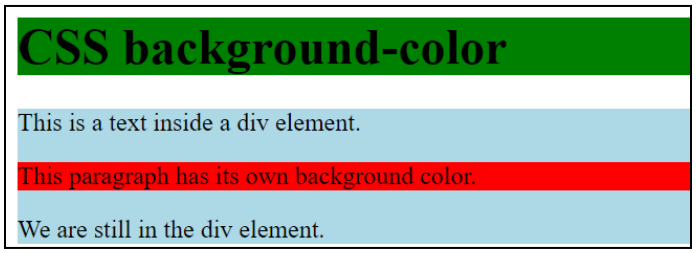
Background-size:

- The background-size property is used to set the size of the background image.
- Example: `background-size: 100px;`

CSS Background - Practice code

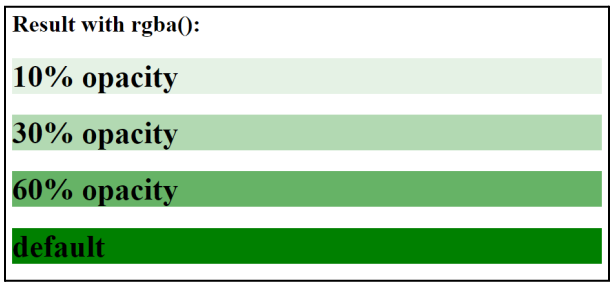
Problem Statement 1: Background Color

Create a simple HTML page with heading and paragraphs, wrap the paragraphs with a div element. Apply different background colors to the heading and div element using the element selector and apply background color to any one paragraph inside the div element using the class or id selector.



Problem Statement 2: Transparency using RGBA

Set the opacity for background color using RGBA, where A specifies the opacity for color. The alpha parameter is between 0.0 (fully transparent) and 1.0 (fully opaque). Apply styles by combining the element and class selector.(Example: ***div.class_name*** → you will learn about this detailly in the upcoming classes)

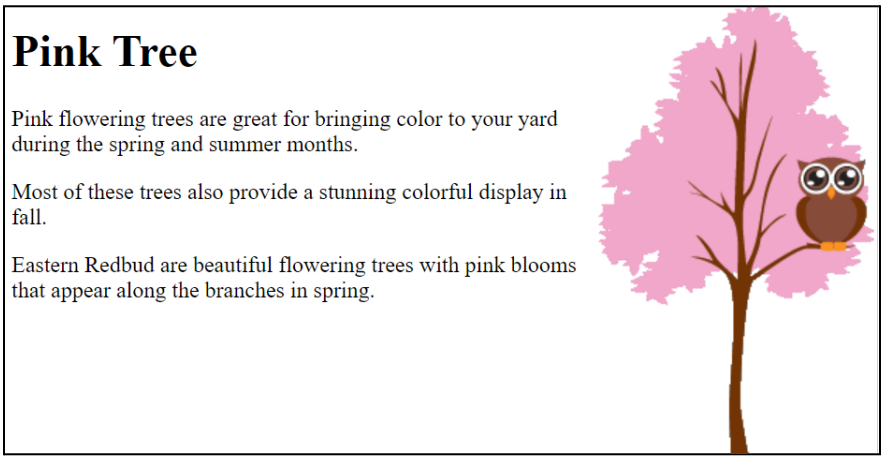


Problem Statement 3: Background Image

Create a simple HTML page and apply the background image to the whole <body> of the HTML page. Modify the text color depending on background image color.

Problem Statement 4: Background Image Position

Create a simple HTML page with text and image, and specify the position to the background image and make sure the image is displayed only one(no repeat).



Solution:

Problem statement 1:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  background-color: green;
}

div {
  background-color: lightblue;
}

.para1 {
  background-color: red;
}
</style>
</head>
<body>

<h1>CSS background-color</h1>
<div>
<p>This is a text inside a div element.</p>
<p class='para1'>This paragraph has its own background color.</p>
<p>We are still in the div element.</p>
</div>

</body>
</html>
```

Problem statement 2:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background: rgb(0, 128, 0);
}

div.first {
  background: rgba(0, 128, 0, 0.1);
}

div.second {
  background: rgba(0, 128, 0, 0.3);
}

div.third {
  background: rgba(0, 128, 0, 0.6);
}
</style>
</head>
<body>
<h2>Result with rgba():</h2>

<div class="first">
  <h1>10% opacity</h1>
</div>

<div class="second">
  <h1>30% opacity</h1>
</div>

<div class="third">
  <h1>60% opacity</h1>
</div>

<div>
  <h1>default</h1>
</div>

</body>
</html>
```

Problem statement 3:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("bgdesert.jpg");
  color: white;
}

</style>
</head>
<body>

<h1>Hello World!</h1>
<p>Paragraph</p>

</body>
</html>
```

Problem statement 4:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  margin-right: 200px;
}
</style>
</head>
<body>

<h1>Pink Tree</h1>
<p>Pink flowering trees are great for bringing color to your yard during the spring and summer months.</p>
<p>Most of these trees also provide a stunning colorful display in fall.</p>
<p>Eastern Redbud are beautiful flowering trees with pink blooms that appear along the branches in spring.</p>
</body>
</html>
```

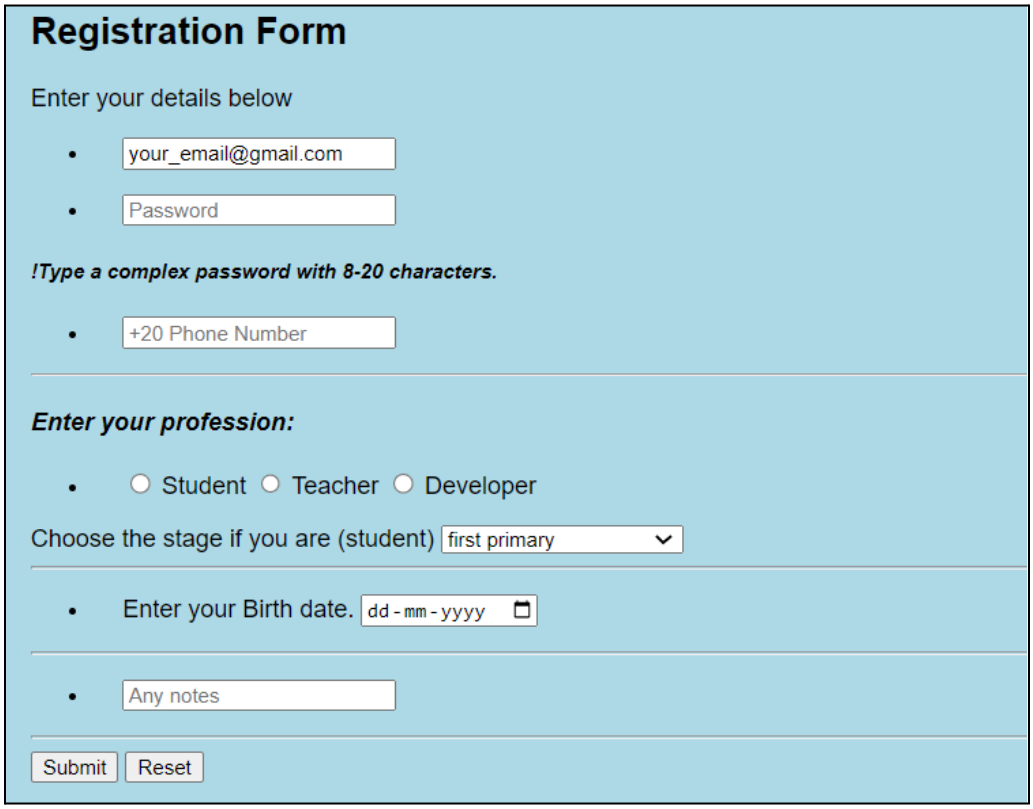
Registration Form

(This is a minor assignment and submission is not required. Will release solution next week so that you can self- evaluate)

Problem statement:

With your new gained knowledge in CSS selectors, background, and box model properties create a simple registration form using HTML form, input elements, dropdown, select one option and make the form attractive by using CSS properties.

Sample output:



Grading parameters:

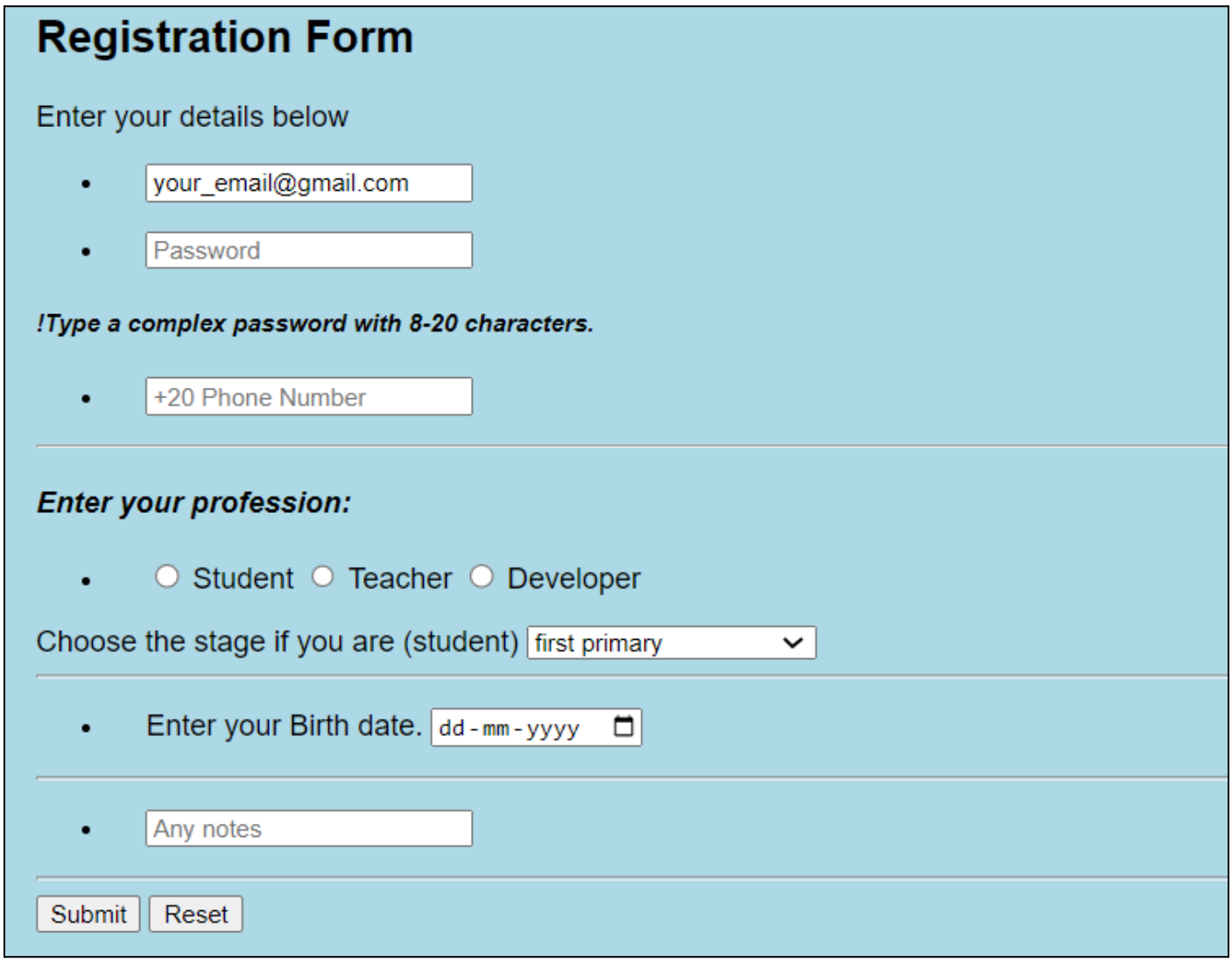
Parameters	Rubics
Form	10 - Used form with at least 5 form elements and 2 buttons 5 - Used form with only few form element (>5) and 2 buttons 0 - No form is used.
Box Model	10 - Usage of padding/margin/border properties to align the form 0 - No box model property is used.
Background	10 - Added background color/image in your project 0 - No background color/image is used
Selector	10 - Used class/id/element selectors for applying styles and some inline styles are used 5 - only inline styles are used 0 - no styles are applied

Registration Form

Problem statement:

With your new gained knowledge in CSS selectors, background, and box model properties create a simple registration form using HTML form, input elements, dropdown, select one option and make the form attractive by using CSS properties.

Sample output:



Sample Code:

HTML:

```
<!DOCTYPE html>
<html>

<head>
  <title>Registration Form</title>
  <link href="style.css" rel="stylesheet" type="text/css" />
</head>

<body style="background-color: lightblue">
<h2>
Registration Form
</h2>
<p>
Enter your details below
</p>
  <form method="post" target="_blank">
    <div>
      <ul class="ul1">
        <li class="li1">
          <input type="email" required placeholder="Email"
value="your_email@gmail.com" name="email" autofocus />
        </li>
      </ul>
    </div>
    <div>
      <ul>
        <li class="li2">
          <input type="password" required placeholder="Password"
name="Password" minlength="8" maxlength="20" />
        </li>
      </ul>
      <h5>
        <i>
          <p>!Type a complex password with 8-20 characters.</p>
        </i>
      </h5>
    </div>
    <div>
      <ul>
        <li class="li3">
          <input type="number" required placeholder="+20 Phone Number"
name="Phone Number" />
        </li>
      </ul>
    </div>
    <hr />
    <h4>
      <i>
        <p>Enter your profession:</p>
      </i>
    </h4>
```

```
    <ul>
      <li class="li4">
        <input id="Student" type="radio" , name="Profession" value="Student"
/>
        <label for="Student">Student</label>
        <input id="Teacher" type="radio" , name="Profession" value="Teacher"
/>
        <label for="Teacher">Teacher</label>
        <input id="Developer" type="radio" , name="Profession"
value="Developer" />
        <label for="Developer">Developer</label>
      </li>
    </ul>
    <label for="stage">Choose the stage if you are (student)</label>
    <select name="stage" id="stage">
      <optgroup label="Primary stage">
        <option value="first primary">first primary</option>
        <option value="second primary">second primary</option>
        <option value="third primary">third primary</option>
        <option value="Fourth primary">Fourth primary</option>
        <option value="Fifth primary">Fifth primary</option>
        <option value="sixth primary">sixth primary</option>
      </optgroup>
      <optgroup label="Preparatory stage">
        <option value="first preparatory">first preparatory</option>
        <option value="second preparatory">second preparatory</option>
        <option value="third preparatory">third preparatory</option>
      </optgroup>
      <optgroup label="Secondary stage">
        <option value="first secondary">first secondary</option>
        <option value="second secondary">second secondary</option>
        <option value="third secondary">third secondary</option>
      </optgroup>
    </select>
    <hr />
    <ul>
      <li class="li5">
        <label for="d">Enter your Birth date.</label>
        <input type="date" name="date" id="d" />
      </li>
    </ul>
    <hr />
    <div>
      <ul>
        <li class="li6">
          <input type="text" placeholder="Any notes" name="Nots" />
        </li>
      </ul>
    </div>
    <hr />
    <input type="submit" />
    <input type="reset" value="Reset" />
  </form>
</body>

</html>
```

CSS:

```
body {
  padding-left: 30px;
  padding-right: 30px;
}

.li1 {
  padding-left: 20px;
}
.li2 {
  padding-left: 20px;
}
.li3 {
  padding-left: 20px;
}
.li4 {
  padding-left: 20px;
}
.li5 {
  padding-left: 20px;
}
.li6 {
  padding-left: 20px;
}
```

CSS Box Model

Topics Covered:

- Box model.
- CSS width and height.
- CSS Borders.
- CSS Padding.
- CSS margin.
- Maximum and minimum width and height.
- CSS overflow.

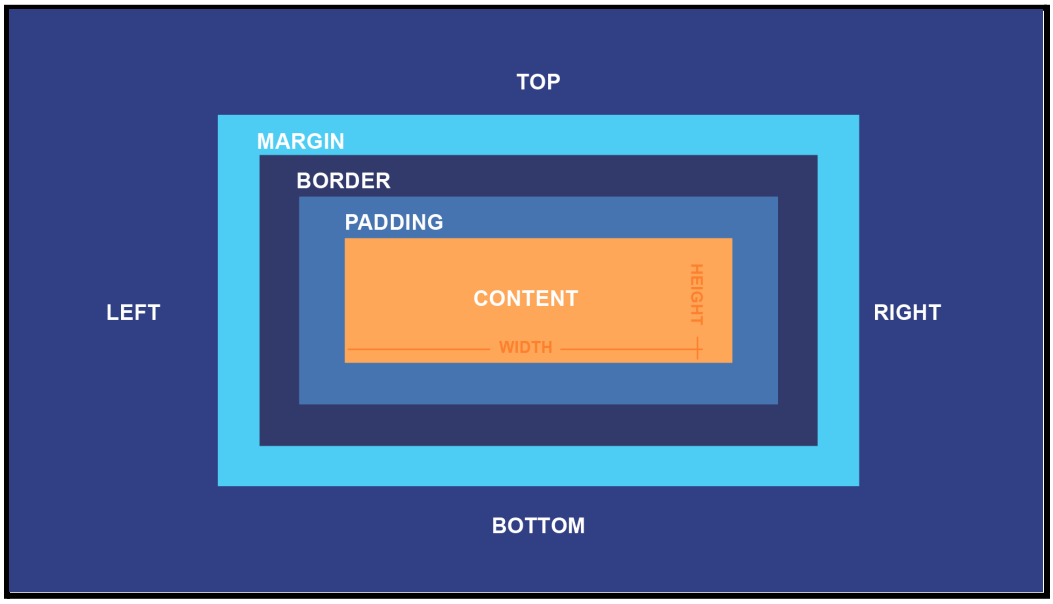
Topics in Detail:

CSS Box model:

- Box model helps you to understand how elements are positioned and displayed on a website.
- All elements on a web page are interpreted by the browser as “living” inside of a box. This is what is meant by the box model.
- For example:
    - If you have set the background color of an element, you may have noticed that the color was applied not only to the area directly behind the element but also to the area around the element.
    - when you change the background color of an element, you change the background color of its entire box.
- The box model includes:
    - The dimensions of an element’s box.
    - The borders of an element’s box. (border of element box is already learnt in the last topic)
    - The paddings of an element’s box.
    - The margins of an element’s box.

Box model properties:

The box model comprises a set of properties of elements that can take up space on a web page.



- Width and height: The width and height of content.
- Padding: The space between the content and the border.
- Border: The thickness, style of border surrounding the content and padding.
- Margin: The space between the border of the element and outside edge/ border of other elements.

Width and Height:

- The content of any HTML element has two dimensions width and height.
- The dimension of the element can be modified with the properties width and height.
- Example:

```
#main {
 width: 400px;
 height: 1000px;
}
```

CSS-Borders:

- A border is a line that surrounds an element, like a frame around a painting.
- Different styles, width, color, radius can be kept for HTML elements.
    - border-style.
    - border-width.
    - border-color.
    - border-radius.

CSS-Padding:

- The space between the content and the border of the box is known as the padding.
- For example, padding is a space between the picture and it's frame.
- The space can be modified by using the padding property in CSS.
- Example:

```
#main {
 padding: 40px;
}
```

- The padding property is often used to expand the background color and make the content look spacious.
- The padding property can be specifically used to all four sides:
    - padding-top: 40px;
    - padding-right: 30px;
    - padding-bottom: 20px;
    - padding-left: 10px;

Padding Shorthand property:

- Padding shorthand property let's you specify all the padding properties as values on a single line.
- 4 values- Example: padding: 6px 11px 4px 9px;
    - padding-top: 6px;
    - padding-right: 11px;
    - padding-bottom: 4px;
    - padding-left: 9px;

- 3 values - Example: padding: 5px 10px 20px;
    - padding-top: 5px;
    - padding-left: 10px; and padding-right: 10px;
    - padding-bottom: 20px;
- 2 values - Example: padding: 5px 10px;
    - padding-top: 5px; and padding-bottom: 5px;
    - padding-right: 10px; and padding-left: 10px;

CSS-Margin:

- Margin refers to the space directly outside the box, or margin can be referred to the space between the border of two elements.
- The margin property of the CSS is used to specify the size of the space between the borders of two elements.
- Example:

```
P{
margin:
20px;
}
```

- The margin property can be specifically used to all four sides:
    - margin-top: 40px;
    - margin-right: 30px;
    - margin-bottom: 20px;
    - margin-left: 10px;

Margin shorthand property:

- Margin shorthand property let's you specify all the padding properties as values on a single line.
- 4 values- Example: padding: 6px 11px 4px 9px; where,
    - margin-top: 6px;
    - margin-right: 11px;
    - margin-bottom: 4px;
    - margin-left: 9px;

- 3 values - Example: margin: 5px 10px 20px; where,
    - margin-top: 5px;
    - margin-right: 10px; and margin-left: 10px;
    - margin-bottom: 20px;
- 2 values - Example: margin: 5px 10px;
    - margin-top: 5px; and margin-bottom: 5px;
    - margin-left: 10px; and margin-right: 10px;

margin: auto;

- The margin property also allows you to center content, with help of the value “auto”.
- Example:

```
div.headline {
  width: 400px;
  margin: 0
auto;
}
```

    - In the example above, the width of the div is set to 400 pixels, which is less than the width of most screens.
    - This will cause the div to center within a containing element that is greater than 400 pixels wide.
    - It’s not possible to center an element that takes up the full width of the page, so the width is given for the div element.
    - The top and bottom margin of the div element will be set to 0.
    - The auto value instructs the browser to adjust the left and right margins until the element is centered within its containing element.

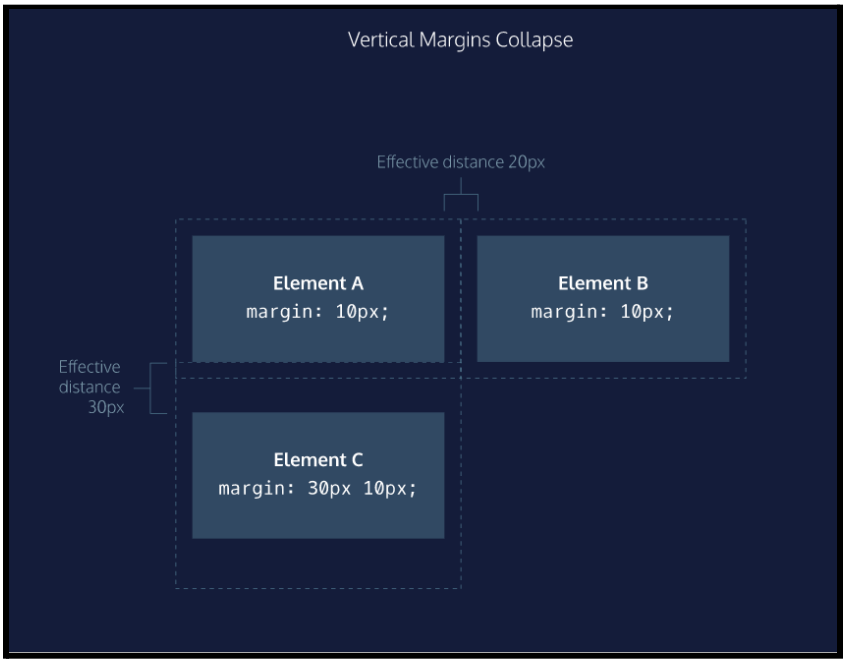
Margin collapse:

- padding is space added inside an element’s border, while margin is space added outside an element’s border.
- One additional difference is that top and bottom margins, also called vertical margins, **collapse**, while top and bottom padding does not.
- Horizontal margins (left and right), like padding, are always displayed and added together.

- Example:

```
#elementa {
  margin: 10px;
}

#elementb {
  margin: 10px;
}

#elementc {
  margin: 30px 10px;
}
```

- For elements A and B the all the margin spaces are 10px. For element C the top and bottom margin is 30px and left and right margin is 10px;
- The elements A and B are placed horizontally next to each other. So the space between A and B elements is (10 +10) 20px. Which will remain the same.
- The element A and C are placed vertically one below the other(element C is placed below the element A).
    - The bottom margin of element A is 10px;
    - The top margin of element C is 30px;
    - The total space is 40px but the vertical gap between element A and C will be 30px. This is known as the Margin collapse.



Margin collapse

Minimum and maximum Height and Width:

- The web page can be viewed through displays of differing screen size, the content on the web page can suffer from those changes in size.
- To avoid this problem, CSS offers two properties that can limit how narrow or how wide an element's box can be sized to:
    - min-width—this property ensures a minimum width of an element's box.
    - max-width—this property ensures a maximum width of an element's box.

```
p {
  min-width:
300px;
  max-width:
600px;
}
```

    - the width of all paragraphs will not shrink below 300 pixels, nor will the width exceed 600 pixels.
- Similar to width, You can also limit the minimum and maximum *height* of an element:
    - min-height — this property ensures a minimum height for an element's box.
    - max-height — this property ensures a maximum height of an element's box.
    - Example:

```
p {
  min-height: 150px;
  max-height: 300px;
}
```

    - the height of all paragraphs will not shrink below 150 pixels and the height will not exceed 300 pixels.

Overflow:

The overflow property controls what happens to content that spills, or overflows, outside its box. The most commonly used values are:

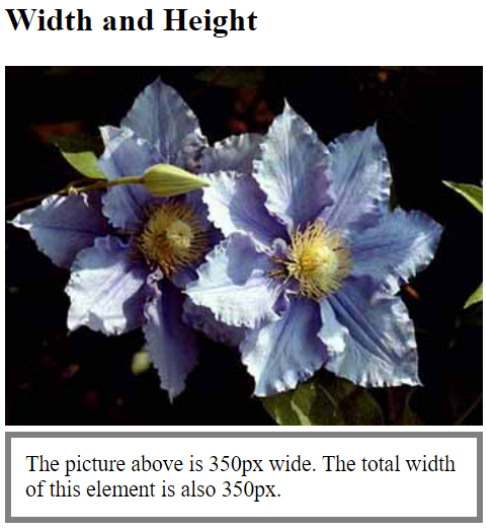
- hidden—when set to this value, any content that overflows will be hidden from view.
- scroll—when set to this value, a scrollbar will be added to the element's box so that the rest of the content can be viewed by scrolling.
- visible—when set to this value, the overflow content will be displayed outside of the containing element. Note, this is the default value.
- Example:

```
p {
  overflow:
scroll;
}
```

CSS Box Model - Practice Code

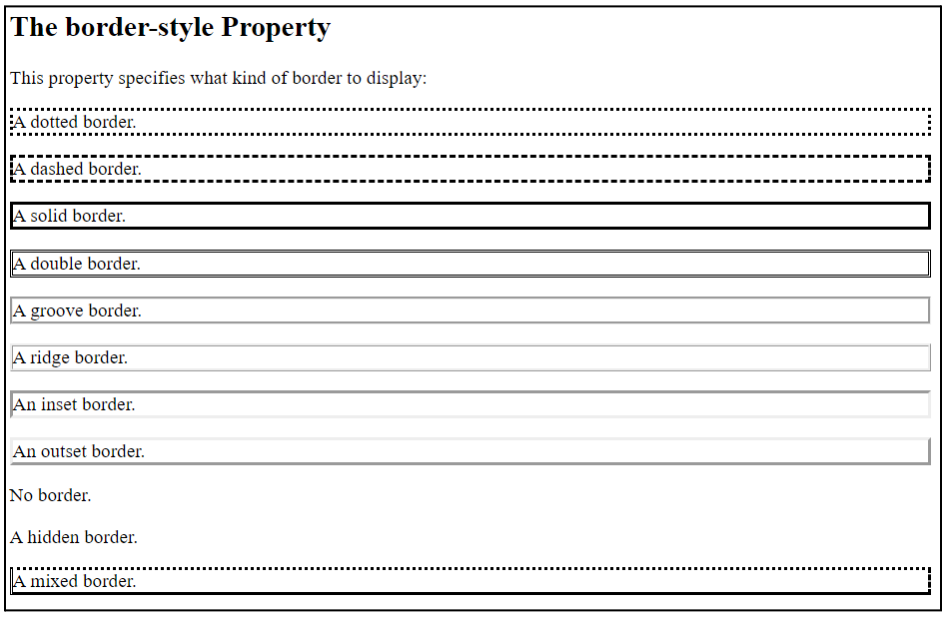
Problem Statement 1: Width and Height

Create a simple HTML page with an image and some content about the image. Depending on size of the image, align the content below the image using the width and height properties. Add a border to the content section.



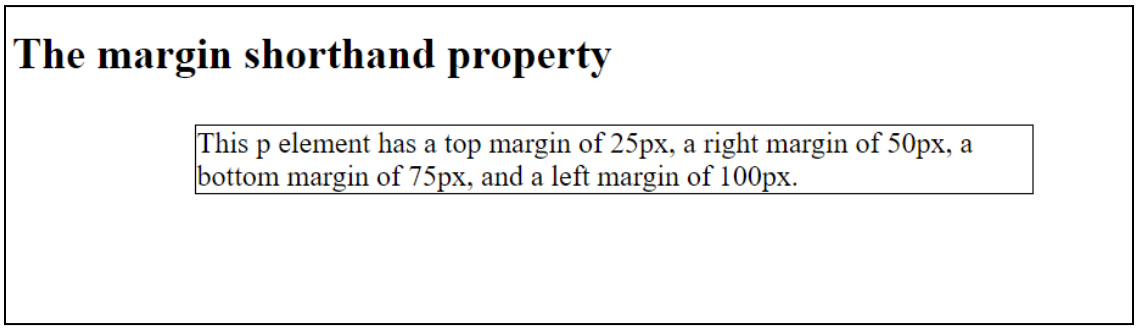
Problem Statement 2: Border Styles

Create a simple HTML page and try all types of borders (dotted, dashed…) available by creating that many paragraphs and applying border properties to them.



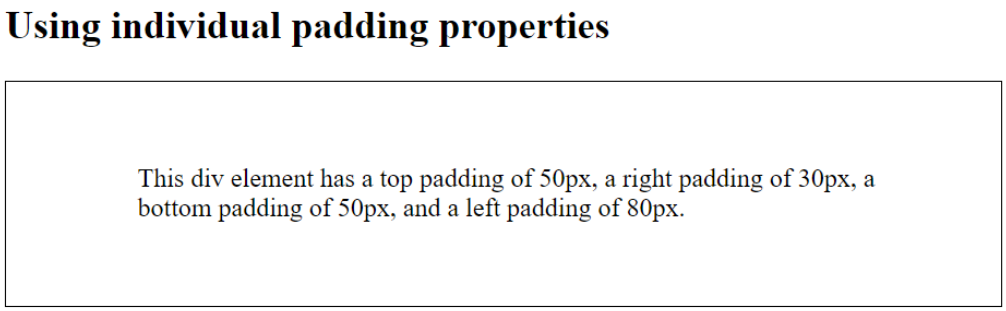
Problem Statement 3: Margin

Create a simple HTML page with heading and paragraph, add different margins for all sides of the paragraph using margin shorthand property. Make sure to add a border to the paragraph so that you can see the margin difference visibility.



Problem Statement 4: Padding

Create a simple HTML page with heading and a div element. Using individual padding properties add padding to the div element.Make sure to add a border to the div element so that you can see the padding difference visibility.



Solution:

Problem Statement 1:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 320px;
  padding: 10px;
  border: 5px solid gray;
  margin: 0;
}
</style>
</head>
```

```
<body>
<h2>Width and Height</h2>
<img src="klematis4_big.jpg" width="350" height="263" alt="Klematis">
<div>The picture above is 350px wide. The total width of this element is
also 350px.</div>
</body>
</html>
```

Problem Statement 2:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
</style>
</head>
<body>

<h2>The border-style Property</h2>
<p>This property specifies what kind of border to display:</p>

<p class="dotted">A dotted border.</p>
<p class="dashed">A dashed border.</p>
<p class="solid">A solid border.</p>
<p class="double">A double border.</p>
<p class="groove">A groove border.</p>
<p class="ridge">A ridge border.</p>
<p class="inset">An inset border.</p>
<p class="outset">An outset border.</p>
<p class="none">No border.</p>
<p class="hidden">A hidden border.</p>
<p class="mix">A mixed border.</p>

</body>
</html>
```

Problem Statement 3:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  border: 1px solid black;
  margin: 25px 50px 75px 100px;
}
</style>
</head>
<body>
<h2>The margin shorthand property</h2>
<p>This p element has a top margin of 25px, a right margin of 50px, a bottom margin of 75px, and a left margin of 100px.</p>
</body>
</html>
```

Problem Statement 4:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
</style>
</head>
<body>

<h2>Using individual padding properties</h2>

<div>This div element has a top padding of 50px, a right padding of 30px, a bottom padding of 50px, and a left padding of 80px.</div>

</body>
</html>
```

CSS Enhancement Properties

Topics Covered:

- What is Text formatting?
- What are different text formatting properties?
- What is Font formatting?
- What are different font formatting properties?
- What is CSS ***float*** property and its possible values?
- What is CSS ***clear*** property and its possible values?
- What is CSS ***display*** property and its possible values?

Topics in Detail:

Text formatting:

- To manipulate the HTML text, CSS text formatting properties are used.
- Text formatting properties of CSS are:
    - color.
    - direction.
    - letter-spacing.
    - word-spacing.
    - text-indent.
    - text-align.
    - text-decoration.
    - text-transform.
    - white-space.
    - text-shadow.

Text Formatting Properties:

Text color:

- The text color of the HTML text can be modified by using **color** property.
- Example:

```
h1{
  color: red;
}
```

Text direction:

- The direction can be specified to the HTML elements by using direction property.
- The possible values of property direction are:
  - rtl - right to left.
  - ltr - left to right.
- Example:

```
.ritole{
  direction: rtl;
}
.letori{
  direction: ltr;
}
```

Right to left direction

Left to right direction

Letter spacing:

- To set the space between the characters of any text, letter-spacing property is used.
- Possible values for the letter-spacing are:
  - normal.
  - Number specifying values (5 px, 10px)
- Example:

```
.spacing{
  letter-spacing: normal;
}
```

```
.spacing{
  letter-spacing: 5px;
}
```

Word spacing:

- The space between two words can be manipulated by using the word-spacing property.
- Possible values for the word-spacing are:
  - normal.
  - Number specifying values (5 px, 10px)
- Example:

```
.spacing{
  word-spacing: 20px;
}
```

Text indent:

-
- Example:

```
.spacing{
  letter-spacing: normal;
  text-indent: 2cm;
}
```

Text alignment:

- To align the text, text-align property is used.
- The possible values are.
  - Left.
  - Right.
  - Center.
  - Justify.
- Example:

```
.spacing{
  letter-spacing: normal;
  width: 800px;
  text-align: right;
}
```

Text decoration:

- The text decorations like underline, strike through, overline, blick can be added to the text by using text-decoration property.
- The possible values are:
  - underline.
  - overline.
  - line-through.
  - blink.
  - none.
- Example:

```
.decoration{
  text-decoration: underline;
}
```

Text cases:

- To change the cases of the character and words, text-transform property is used.
- The possible values are:
  - capitalize - First character of all the words will be in uppercase.
  - uppercase - all the characters of the words will be in uppercase.
  - lowercase - all the characters of the words will be in lowercase.
- Example:

```
.decoration{
  text-decoration: underline;
  text-transform: capitalize;
}
```

White space between text:

- White space: Whitespace is **any string of text composed only of spaces, tabs or line breaks.**
- White space inside any html element can be handled by using white-space property.
- The possible values are:
  - **pre** - Whitespace is preserved by the browser. Text will only wrap on line breaks. Acts like the <pre> tag in HTML.
  - **normal** - Sequences of whitespace will collapse into a single whitespace. Text will wrap when necessary. This is the default value.
  - **nowrap** - Sequences of whitespace will collapse into a single whitespace. Text will never wrap to the next line. The text continues on the same line until a
 tag is encountered.
- Example:

```
.spacing{
  letter-spacing: normal;
  white-space:normal;
}
```

Text shadow:

- To set shadow around the text for decoration purpose, text-shadow property is used.
- This property may not be supported by some browsers.
- Syntax: text-shadow: h-shadow v-shadow blur-radius color;
  - h-shadow: horizontal shadow.
  - v-shadow: vertical shadow.

    - Blur-radius: optional, default value is 0.
- Example:

```
.decoration{
  text-decoration: underline;
  text-transform: capitalize;
  text-shadow: 4px 4px 8px pink;
}
```

Font Formatting:

- To manipulate the HTML fonts, font formatting can be used.
- Font formatting can be done using following properties:
    - font-family.
    - font-style.
    - font-variant.
    - font-weight.
    - font-size.

Font Formatting Properties:

Font family:

- To specify the type of font format for the text, font-family property can be used.
- The value of the property font-family should be the name of the font-family.
- Example:

```
.spacing{
  letter-spacing: normal;
  white-space:normal;
  font-family:georgia,garamond,serif;
}
```

Font style:

- To specify the style of the font, font-style property can be used.
- The values of the property font-style can be normal, italics, and oblique.
- Example:

```
.spacing{
  letter-spacing: normal;
  white-space:normal;
  font-family:georgia,garamond,serif;
  font-style:italic;
}
```

Font variant:

- To specify the difference between the first character of the sentence, even when all the characters are in uppercase, font-variant property is used.
- The possible values are:
  - small-caps
  - normal.
- Example:

```
.letori{
  direction: ltr;
  font-variant:small-caps;
}
```

Font size:

- To manipulate the size of the text, font-size property can be used.
- The possible values are:
  - xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger, size in pixels or in %.
- Example:

```
font-size: xx-small;
```

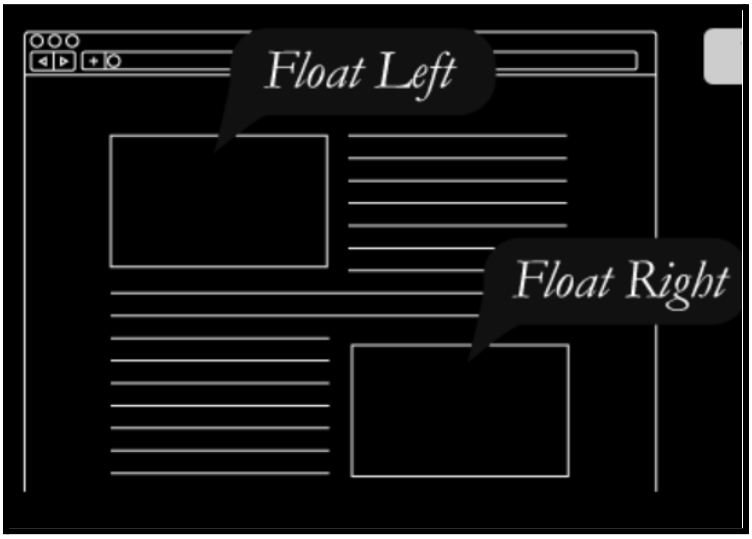
Font shorthand property:

- All the font formatting properties can be used with a single property font.
- Example:

```
h1{
  color: red;
  font:italic small-caps bold 15px georgia;
}
```

CSS float:

- CSS ***float*** is a positioning property.
- CSS ***float*** property specifies how the element should be positioned.
- CSS ***float*** property is used to push an element left or right. Elements can be pushed either left or right only horizontally, not vertically.



CSS ***float*** property values:

Property	Description	Values
float	Specifies the position of the element, whether it should float or not, and where it should float.	left, right, none, inherit

Values description:

Property	Value	Description
float	left	To make the element float to the left side of the container.
float	right	To make the element float to the right side of the container.
float	inherit	To inherit the float value of the parent element.
float	none	No float.

Example:

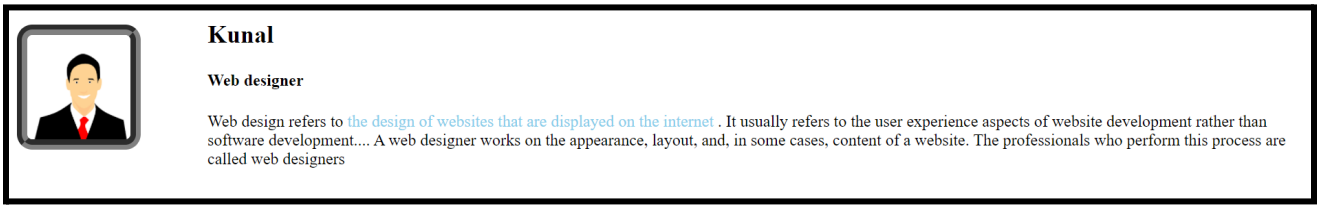
Code

```
  <body>
    <img src="image.png" alt="profile"
width="250px"></img>
    <h2 class="profile name">Kunal</h2>
    <h4 class="profile">Web designer</h4>
    <p class="profile">Web design refers to <span
style="color: skyblue"> the design of websites
that are displayed on the internet </span>. It
usually refers to the user experience aspects of
website development rather than software
development.... A web designer works on the
appearance, layout, and, in some cases, content
of a website. The professionals who perform this
process are called web designers</p>
  </body>
```

Style

```
img{
  width: 100px;
  height: 100px;
  border-radius: 15px;
  border-color: grey;
  border-style: groove;
  border-width: 10px;
  margin: 5px;
  float: left;
}

.profile{
  margin-left: 190px;
}
```

Output:



CSS *clear* Property:

- To clear the next element that flows around the floating element, a ***clear*** property is

used.

- The ***clear*** property is used to specify what should happen to the next element that is next to the floating element.

Property	Description	Values
clear	To clear the next element that flows around the floating element. If the next element is to be positioned below, not left or right of the floating element.	left, right, both, none, inherit

Example:

Code

```
  <body>
    <img src="image.png" alt="profile"
width="250px"></img>
    <h2 class="profile name">Kunal</h2>
    <h4 class="profile">Web designer</h4>
    <p class="profile">Web design refers to <span
style="color: skyblue"> the design of websites
that are displayed on the internet </span>. It
usually refers to the user experience aspects of
website development rather than software
development.... A web designer works on the
appearance, layout, and, in some cases, content
of a website. The professionals who perform this
process are called web designers</p>
  </body>
```

Style

```
img{
  width: 100px;
  height: 100px;
  border-radius: 15px;
  border-color: grey;
  border-style: groove;
  border-width: 10px;
  margin: 5px;
  float: left;
}

.profile{
  clear:left;
}
```

Output:

Kunal

Web designer

Web design refers to the design of websites that are displayed on the internet . It usually refers to the user experience aspects of website development rather than software development.... A web designer works on the appearance, layout, and, in some cases, content of a website. The professionals who perform this process are called web designers

CSS *display* Property:

- The ***display*** property is used to specify how the element should be displayed on the web page.
- The ***display*** property is used to control the layout of the display.
- Every HTML element has a default display property, depending on their type.
    - Block level element - starts on a new line.
    - Inline element - does not start in a new line, takes up necessary space.
- To override the default display nature of the elements, the ***display*** property is used.
- The ***display*** property specifies the display behavior (type of rendering box - all elements are considered to be surrounded by the rectangular box in HTML) of an element.

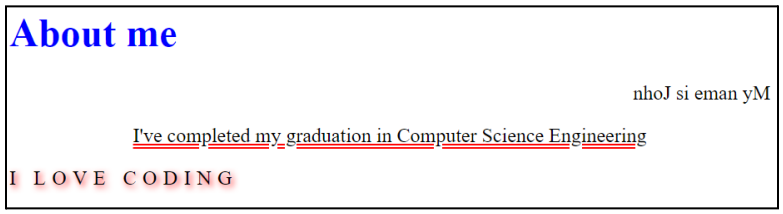
Values and Description of Display properties:

Value	Description
inline	Does not force the line break, makes the element take only the required space.
inline-block	Same as value inline, but addition to that height and width can be applied to the elements.
block	Make the element take as much horizontal space as they can.
run-in	Displays the element as block or inline depending on the content.
none	Totally removes the element from the page. Does not provide any space.
flex	Displays an element as a block level flex container.
grid	Displays an element as a block level grid container.
inline-flex	Displays an element as an inline level flex container.
inline-grid	Displays an element as an inline level grid container.
list-item	Make the element behave like element.
table	Make the element behave like <table> element.
table-cell	Make the element behave like <td> element.
table-row	Make the element behave like <tr> element.
table-caption	Make the element behave like <caption> element.

CSS Enhancement properties - Practice code

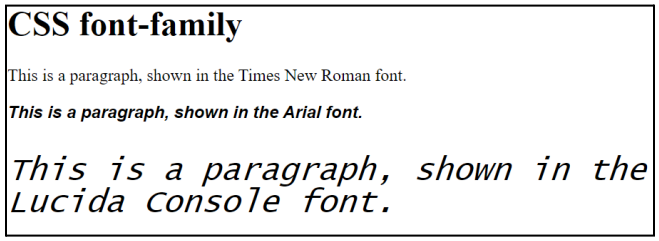
Problem Statement 1: Text

Create a simple web page with headings and paragraphs. Make the web page attractive using CSS text properties: text color, decoration, alignment, direction, shadow, and spacing.



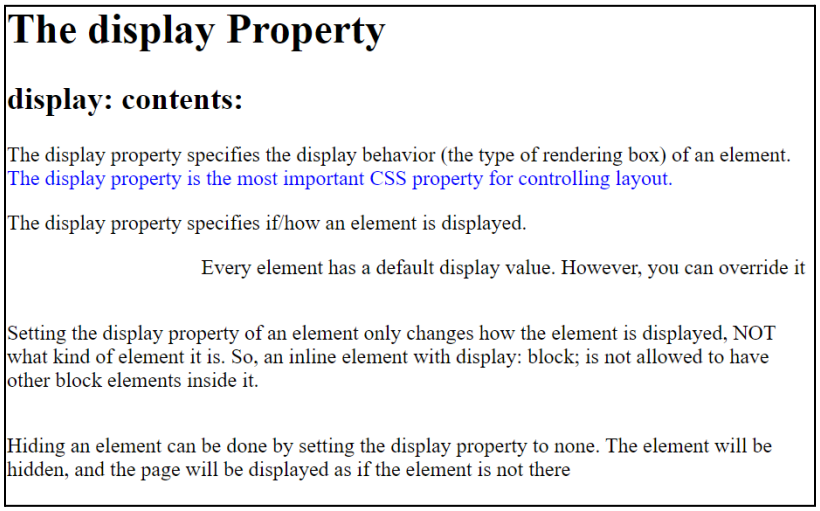
Problem Statement 2: Fonts

Create a simple web page with multiple paragraphs. Apply different font properties (font-family, font-style, font-weight, and font-size) to all paragraphs to make the web page attractive.



Problem Statement 3: Display

Create a simple web page with multiple paragraphs. Apply different display properties to all paragraphs by understanding the property values used.



Problem Statement 4: Float

Create a webpage with an image and paragraph that describes the image. Using the float property make the image appear left/right to the paragraph.

Pineapple

Pineapples are tropical fruits that are rich in vitamins, enzymes and antioxidants. They may help boost the immune system, build strong bones and aid indigestion. Plus, despite their sweetness, pineapples are low in calories. Pineapples are members of the bromeliad family, and are the only bromeliad that produces edible fruit, according to the Purdue University Center for New Crops and Plant Products(opens in new tab). The fruit is made of many individual berries that grow together around a central core. Each pineapple scale is an individual flower, or berry. The nutritional benefits of pineapples are as attractive as their unique anatomy. "Pineapples contain high amounts of vitamin C and manganese," said San Diego-based nutritionist Laura Flores. These tropical fruits are also a good way to get important dietary fiber and bromelain (an enzyme).

Solution

Problem Statement 1

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
 color: blue;
}
.para1 {
  direction: rtl;
  unicode-bidi: bidi-override;
}
.para2 {
      text-align: center;
    text-decoration-line: underline;
    text-decoration-style: double;
    text-decoration-color: red;
}
.para3 {
      text-transform: uppercase;
    letter-spacing: 5px;
    text-shadow: 2px 2px 5px red;

}
</style>
</head>
<body>
<h1>About me</h1>
<p class="para1">My name is John</p>
<p class="para2">I've completed my graduation in Computer Science
Engineering</p>
<p class="para3">I love coding</p>
</body>
</html>
```

Problem Statement 2

```
<!DOCTYPE html>
<html>
<head>
<style>
.p1 {
  font-family: "Times New Roman", Times, serif;
  font-style: normal;
}
.p2 {
  font-family: Arial, Helvetica, sans-serif;
  font-style: italic;
  font-weight: bold;
}
.p3 {
  font-family: "Lucida Console", "Courier New", monospace;
  font-style: oblique;
  font-size: 30px;
}
</style>
</head>
<body>
<h1>CSS font-family</h1>
<p class="p1">This is a paragraph, shown in the Times New Roman font.</p>
<p class="p2">This is a paragraph, shown in the Arial font.</p>
<p class="p3">This is a paragraph, shown in the Lucida Console font.</p>
</body>
</html>
```

Problem Statement 3

```
<!DOCTYPE html>
<html>
<head>
<style>
.a {
  display: contents;
  border: 1px solid red;
  background-color: lightgrey;
  padding: 10px;
  width: 200px;
}
.b {
  display: inline;
  color:blue;
}
.c{
      display: inherit;
}
.d{
      display: flex;
      flex-direction: row-reverse;
}
.e{
```

```
        display:none;
}
.f{
        display: inline-flex;
}
.g{
        display: run-in;
}
</style>
</head>
<body>
<h1>The display Property</h1>
<h2>display: contents:</h2>
<div class="a">
The display property specifies the display behavior (the type of rendering
box) of an element.
</div>
<p class="b">The display property is the most important CSS property for
controlling layout.
</p>
<p class="c">
The display property specifies if/how an element is displayed.
</p>
<p class="d">Every element has a default display value. However, you can
override it</p>
<p class="e">
Changing an inline element to a block element, or vice versa, can be useful
for making the page look a specific way, and still follow the web
standards.</p>
<p class="f">Setting the display property of an element only changes how the
element is displayed, NOT what kind of element it is. So, an inline element
with display: block; is not allowed to have other block elements inside
it.</p>
<p class="g">Hiding an element can be done by setting the display property
to none. The element will be hidden, and the page will be displayed as if
the element is not there</p>
</body>
</html>
```

Problem Statement 4

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  float: right;
}
</style>
</head>
<body>
<h2>Pineapple</h2>
<p><img src="pineapple.jpg" alt="Pineapple"
style="width:170px;height:170px;margin-left:15px;">
Pineapples are tropical fruits that are rich in vitamins, enzymes and
```

```
antioxidants. They may help boost the immune system, build strong bones and
aid indigestion. Plus, despite their sweetness, pineapples are low in
calories. Pineapples are members of the bromeliad family, and are the only
bromeliad that produces edible fruit, according to the Purdue University
Center for New Crops and Plant Products(opens in new tab). The fruit is made
of many individual berries that grow together around a central core. Each
pineapple scale is an individual flower, or berry.
The nutritional benefits of pineapples are as attractive as their unique
anatomy. "Pineapples contain high amounts of vitamin C and manganese," said
San Diego-based nutritionist Laura Flores. These tropical fruits are also a
good way to get important dietary fiber and bromelain (an enzyme).</p>
</body>
</html>
```

CSS Positioning

Topics Covered:

- Block level elements.
- CSS position property.
- Possible values of Position property.
- Z-index.

Topics in Detail:

Block level elements:

- Block level elements create a full width of their parent elements, and they prevent other elements from appearing in the same horizontal line.
- Block level elements take up their own line of space and do not overlap with each other.
- The default position of the block level elements is to appear on the left side of the browser.

- The default position of an element can be changed by using ***position*** property.

CSS Position Property:

- The CSS ***position*** property is used to set position for an element.
- The CSS ***position*** property is also used to place an element behind another and also useful for scripted animation effects.
- The CSS ***position*** property can take following possible values:
    - static.
    - relative.
    - absolute.
    - fixed.
    - sticky.

Possible values of Position property:

Position: Static;

- The default value of the CSS ***position*** property is ***static***.
- HTML elements are positioned ***static*** by default.
- An element with ***position: static;*** is not positioned in any special way.
- It is not affected by top, right, left, bottom properties.

Position: Relative;

- The relative ***position*** property is used to set the element relative to its normal position.
- Example:

```
.green-box {
  background-color: green;
  position: relative;
}
```

- The code in the above example instructs the browser to place the ***.green-box*** element in relative position.
- But it does not specify where the .green-box element should be positioned. This can be done by accompanying the position declaration with any one of the following offset properties.
    - top - moves the element down from the top.
    - bottom - moves the element up from the bottom.
    - left - moves the element away from the left to right side.
    - right - moves the element away from the right to left side.
- The values of the offset properties can be in pixels, ems, percentages,...
- Example:

```
.green-box {
  background-color: green;
  position: relative;
  top: 50px;
  left: 120px;
}
```

- Before and after applying relative position and offset values:

- Offsetting the relative property will not affect the positioning of other elements.

Position: Absolute;

- When an element's ***position*** is set to absolute, all other elements on the page will ignore the element and act like it is not present on the page.
- The element will be positioned relative to its closest positioned parent element, while offset properties can be used to determine the final position from there.
- Example:

```
header {
  background-color: #466995;
  border-bottom: 1px solid #466995;
  position: absolute;
  width: 100%;
}
```

Position: Fixed;

- When the element ***position*** is set to absolute, the element will scroll when the user scrolls the document.
- We can fix an element to a specific position on the page (regardless of user scrolling) by setting its position to fixed, and accompanying it with the familiar offset properties top, bottom, left, and right.
- Example:

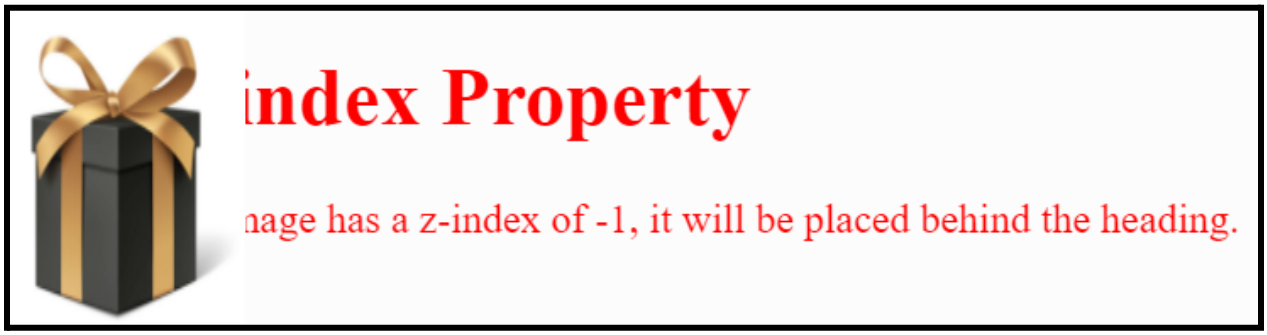
```
header {
  background-color: #466995;
  border-bottom: 1px solid #466995;
  position: fixed;
  width: 100%;
}
```

Position: Sticky;

- The sticky value is another ***position*** value that keeps an element in the document flow as the user scrolls, but ***sticks*** to a specified position as the page is scrolled further.
- This is done by using the sticky value along with the familiar offset properties, as well as one new one.
- A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

Z-index:

- When elements on a web page have combinations of different positions, their contents can overlap, making the content difficult to read.
- Example:

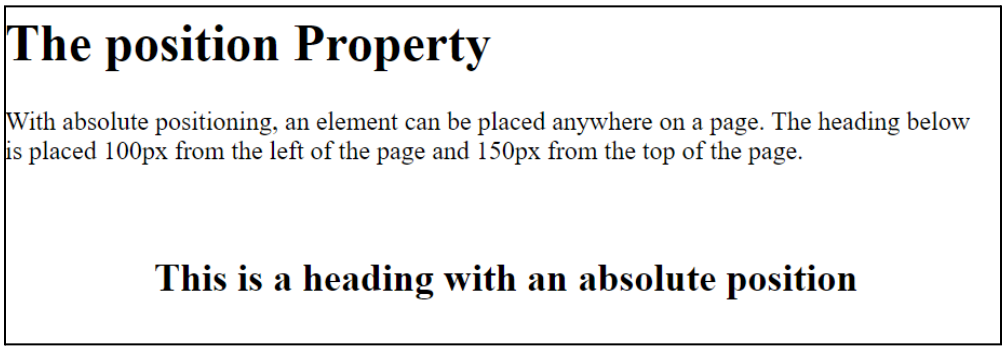


- The z-index property specifies the stack order of an element. z-index only works on positioned elements.
- The z-index property accepts integer values. Depending on their values, the integers instruct the browser on the order in which elements should be layered on the web page.

CSS Position - Practice code

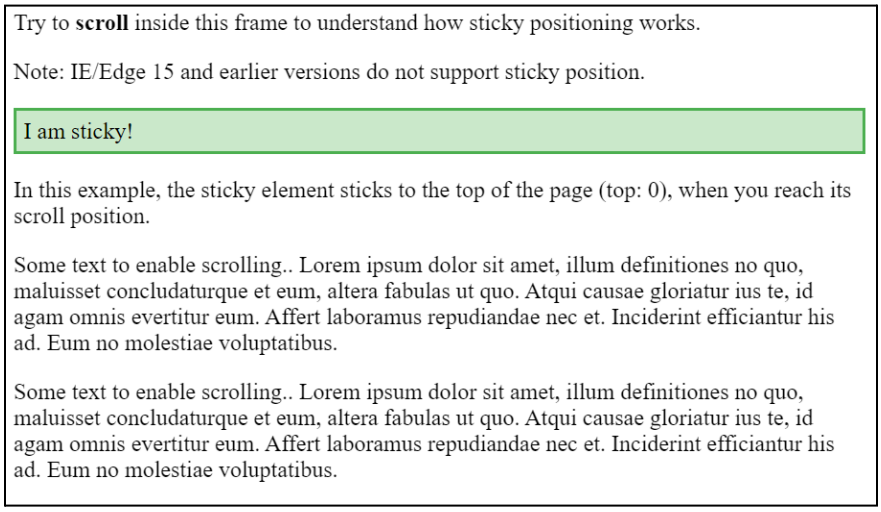
Problem Statement 1: Absolute

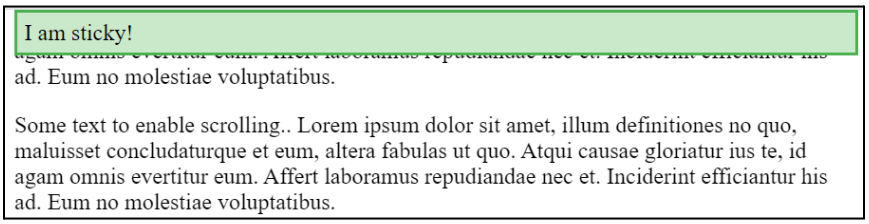
Create a simple web page with headings and paragraphs, apply position property and value as absolute to any of the HTML elements.



Problem Statement 2: sticky

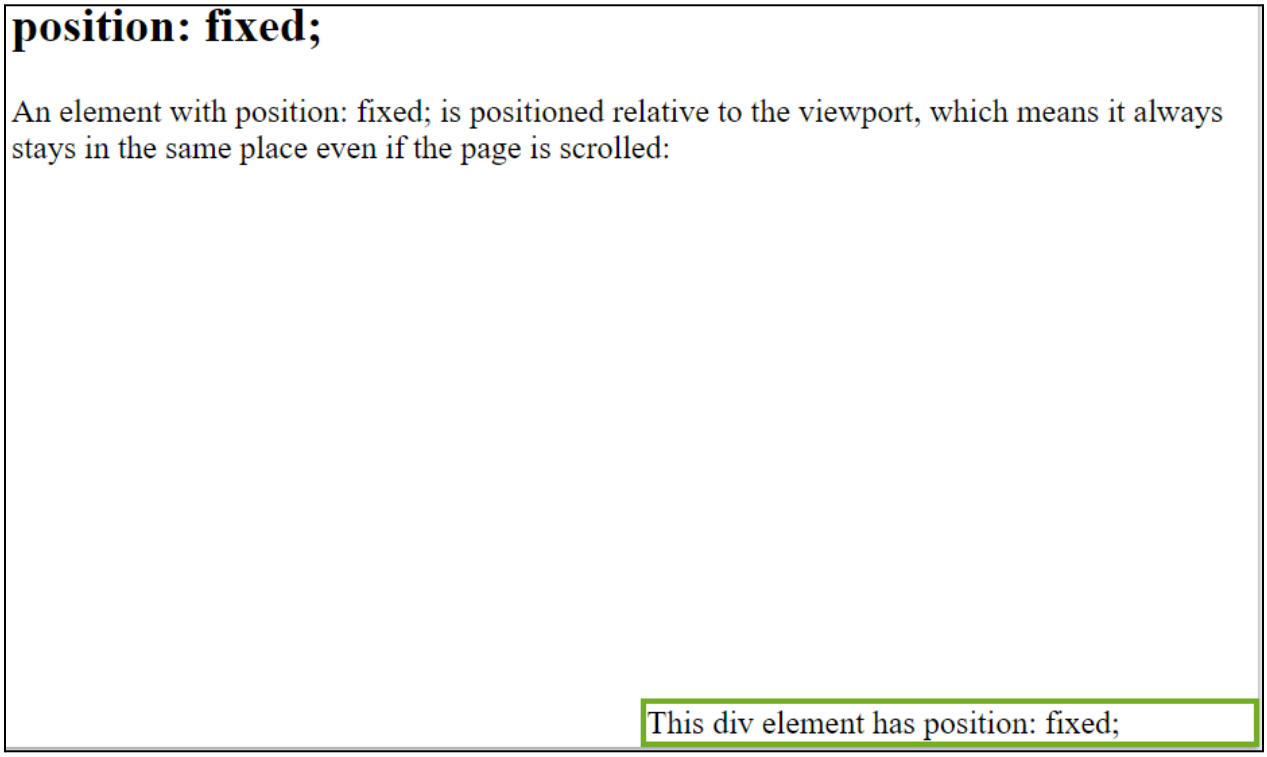
Create a simple web page with headings and paragraphs, apply position property and value as sticky to any of the HTML elements.





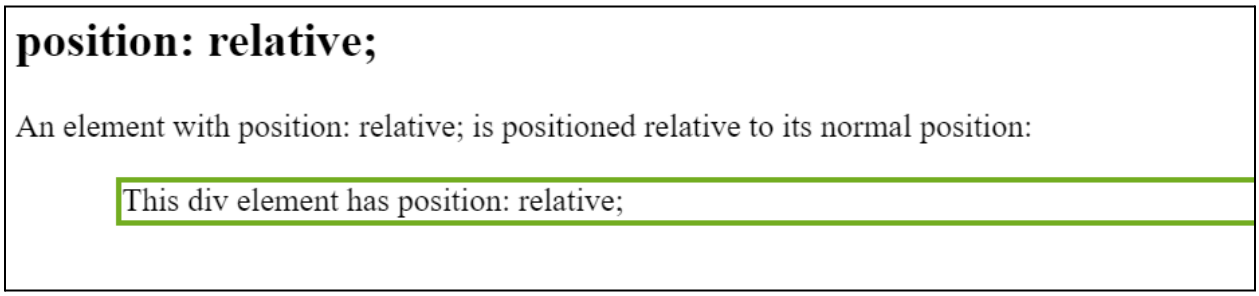
Problem Statement 3: Fixed

Create a simple web page with headings and paragraphs, apply position property and value as fixed to any of the HTML elements.



Problem Statement 4: Relative

Create a simple web page with headings and paragraphs, apply position property and value as relative to any of the HTML elements.



Solution

Problem Statement 1

```
<!DOCTYPE html>
<html>
<head>
<style>
h2 {
  position: absolute;
  left: 100px;
  top: 150px;
}
</style>
</head>
<body>
<h1>The position Property</h1>
<h2>This is a heading with an absolute position</h2>
<p>With absolute positioning, an element can be placed anywhere on a page.
The heading below is placed 100px from the left of the page and 150px from
the top of the page.</p>
</body>
</html>
```

Problem Statement 2

```
<!DOCTYPE html>
<html>
<head>
<style>
div.sticky {
  position: sticky;
  top: 0;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
</style>
</head>
<body>
<p>Try to <b>scroll</b> inside this frame to understand how sticky
positioning works.</p>
<p>Note: IE/Edge 15 and earlier versions do not support sticky position.</p>
<div class="sticky">I am sticky!</div>

<div style="padding-bottom:2000px">
  <p>In this example, the sticky element sticks to the top of the page (top:
0), when you reach its scroll position.</p>
  <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum
definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo.
Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus
repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae
voluptatibus.</p>
  <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum
```

```
definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo.
Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus
repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae
voluptatibus.</p>
</div>
</body>
</html>
```

Problem Statement 3

```
<!DOCTYPE html>
<html>
<head>
<style>
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>
<h2>position: fixed;</h2>
<p>An element with position: fixed; is positioned relative to the viewport,
which means it always stays in the same place even if the page is
scrolled:</p>
<div class="fixed">
This div element has position: fixed;
</div>
</body>
</html>
```

Problem Statement 4

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  left: 50px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>
<h2>position: relative;</h2>
<p>An element with position: relative; is positioned relative to its normal
position:</p>
<div class="relative">
This div element has position: relative;
</div>
```

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

Instagram Logo

(This is a minor assignment and submission is not required. Will release solution next week so that you can self- evaluate)

Problem statement:

With your new gained knowledge in CSS background, width, height, display, position, and flex properties design an instagram logo.

Sample Output:

Grading Parameters:

Parameters	Rubics
Background	10 - Use of radial-gradient background 0 - If no element is used.
Box model	10 - If border, padding, border-radius, and directional properties are used. 5 - If any one of the box model properties are used. 0 - No Box model properties are used.
Width and Height	10 - Use of width and height properties. 5 - Either width or height properties are used. 0 - No width and height properties are used.
Position and place-items	10 - Use of position and place-item properties. 5 - Use of either position and place-item property. 0 - neither position or place-item property is used.

CSS Flex

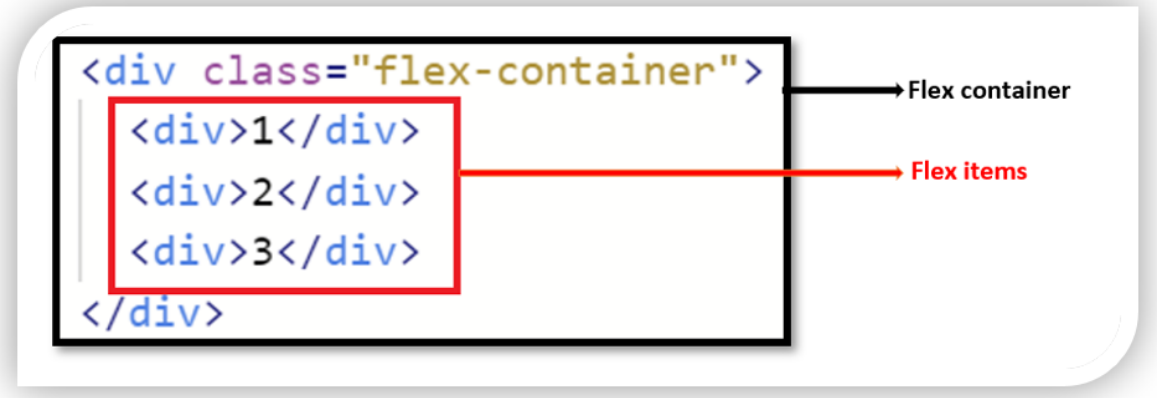
Topics Covered:

1. What is Flexbox?
2. Uses of Flexbox.
3. Important terminologies.
4. Flex properties.
    a. Flex container properties.
    b. Flex item properties.

Topics in Detail:

Flexbox:

- Flexbox is a one-dimensional layout model, where flex means flexible length on flexible items.
- The flexbox layout must have a **parent element(container)** with the **display** property "**flex**". So that the child elements of the container automatically become **flex items**.



- Flexbox provides an easy way to arrange items within a container.
- Flexbox **aligns and distributes space** among the items in the container even when the size of the items is unknown or dynamic.
- Flexbox expands/shrinks to fill available space/prevent overflow and distributes space among the items.
- Flexbox is used for creating small-scale layouts.
- Flexbox aligns the items vertically and horizontally using rows and columns.

Flexbox uses:

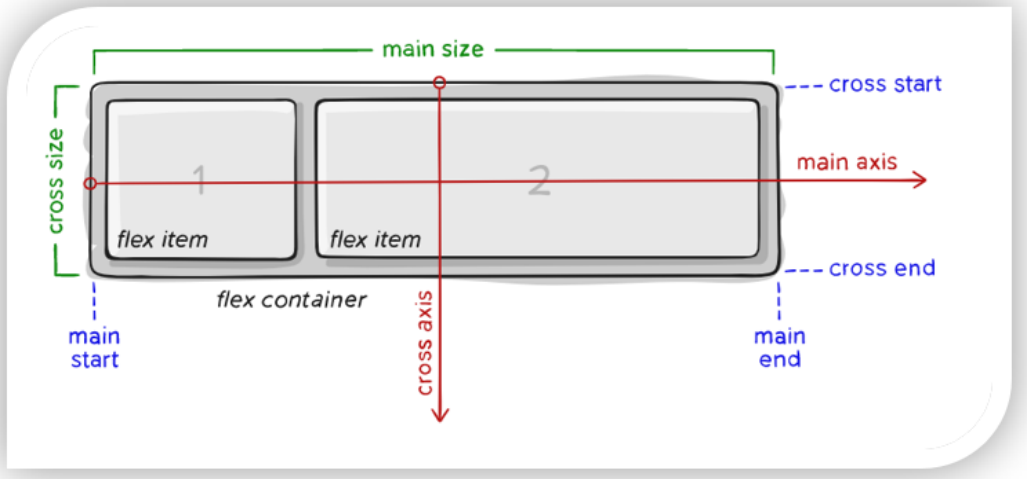
- The flexbox makes the website/web app behave in a similar manner even when they are viewed on different devices that have different screen sizes.
- To create responsive and mobile-friendly websites/web apps flexbox can be used.

Important terminologies:

- The two main components of flexbox are flex container and flex item.
    - **Flex container**: Property of parent element that is declared with display property flex.

```
.flex-container {
  display: flex;
}
```

    - **Flex item**: Properties of child elements. There may be one or many flex items.
- Flex layout is based on **flex-flow directions**, which are used in displaying the flex items. Flexbox has two axes.
    - **Main axis**: The primary axis along with the flex items are laid out.
    - **Cross axis**: The axis perpendicular to main axis is called cross axis.

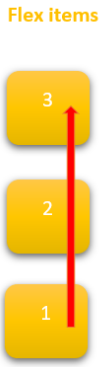
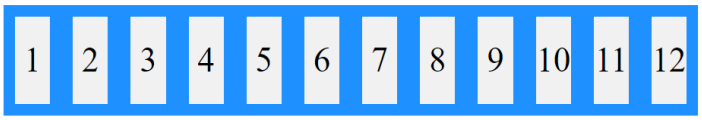
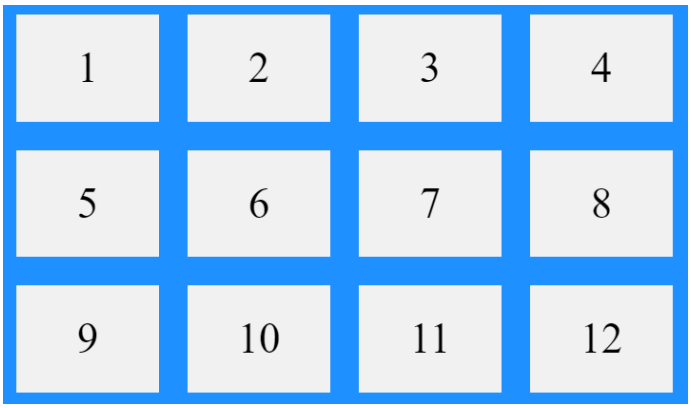


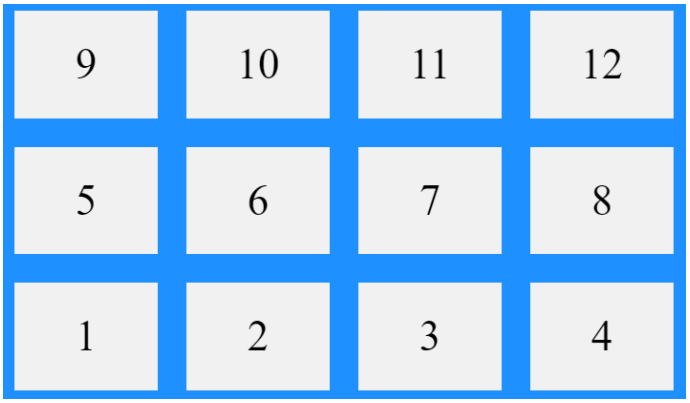
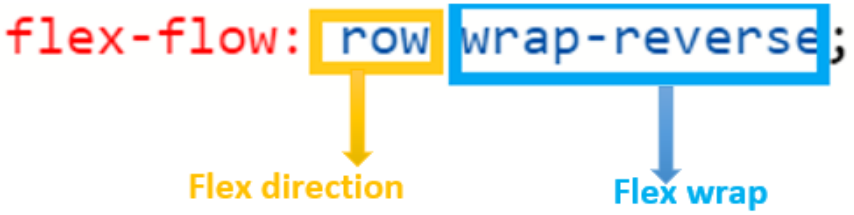
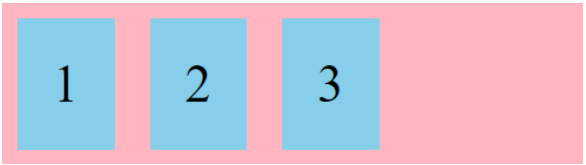
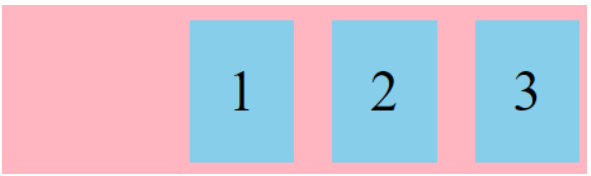
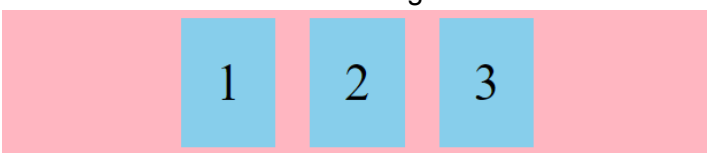
NOTE: It is not necessary that the main axis should always be horizontal, it depends on the **flex-direction** property. The main axis and cross axis are perpendicular to each other.

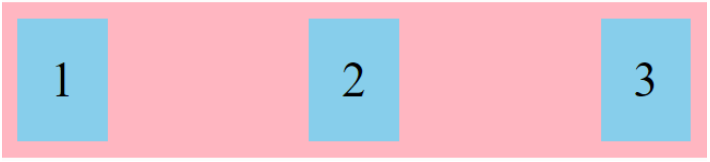
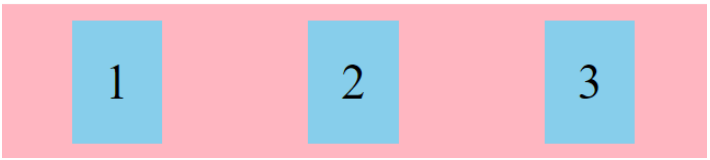
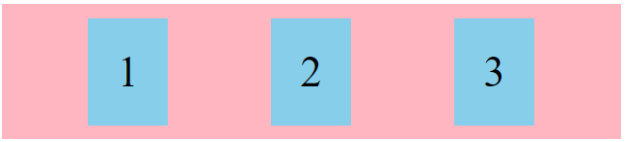
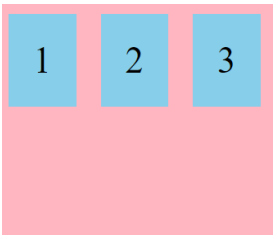
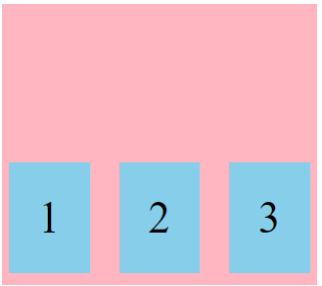
Flex properties:

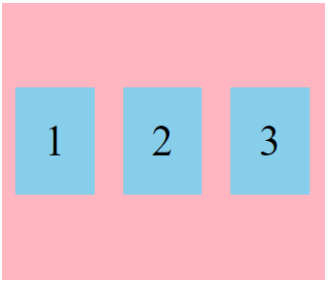
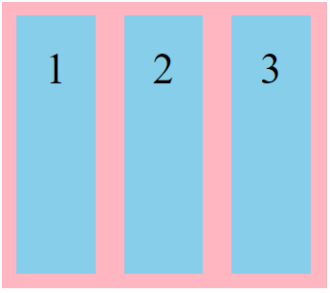
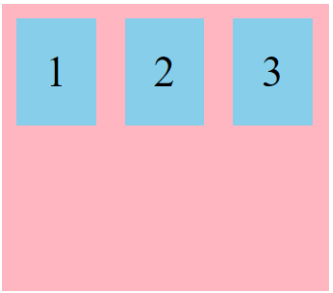
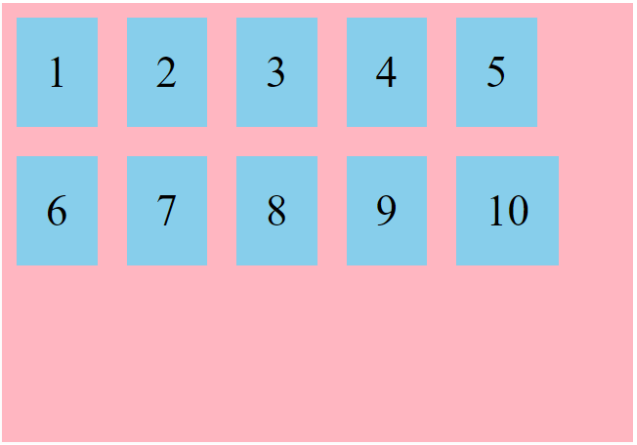
Flex container properties:

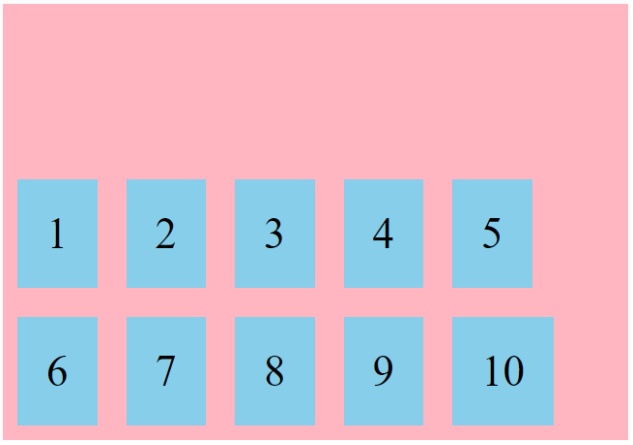
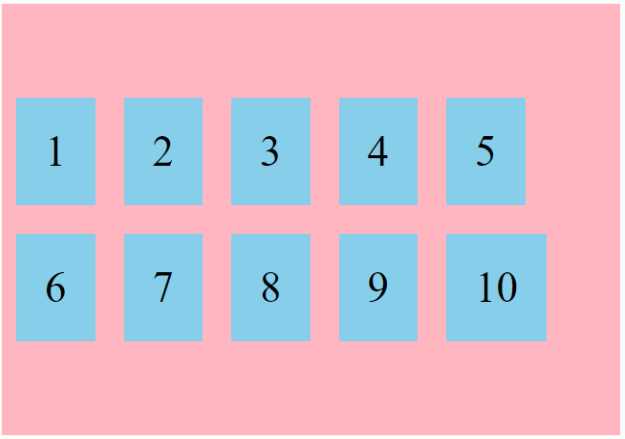
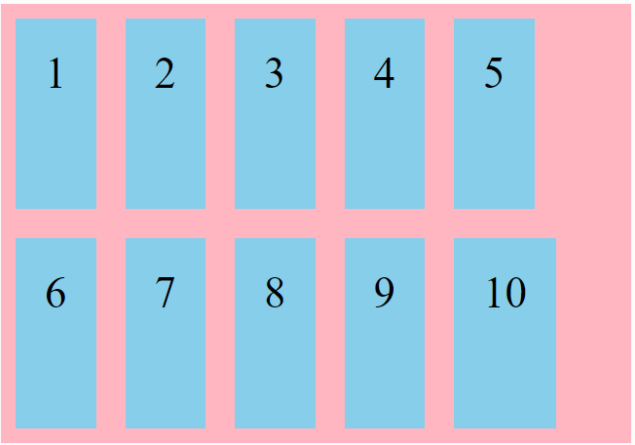
Property	Value and Description	
display	flex	Enables flex context to all direct children of the container. `display: flex;`
flex-direction	Specifies the direction of flex items in the main axis.	
flex-direction	row	Default value, left to right. 
flex-direction	row-reverse	Right to left. 
flex-direction	column	Top to bottom. 

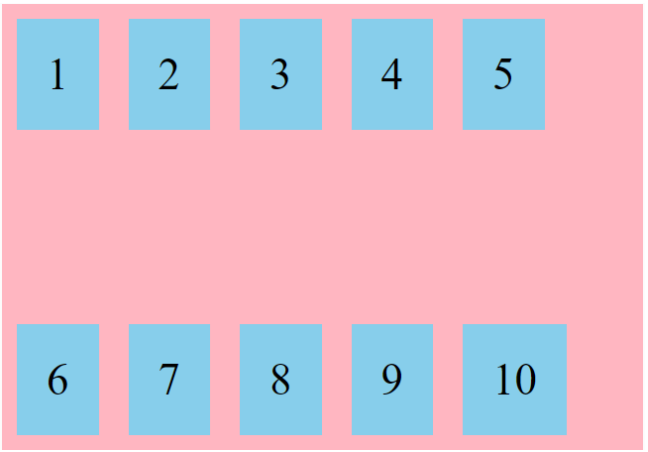
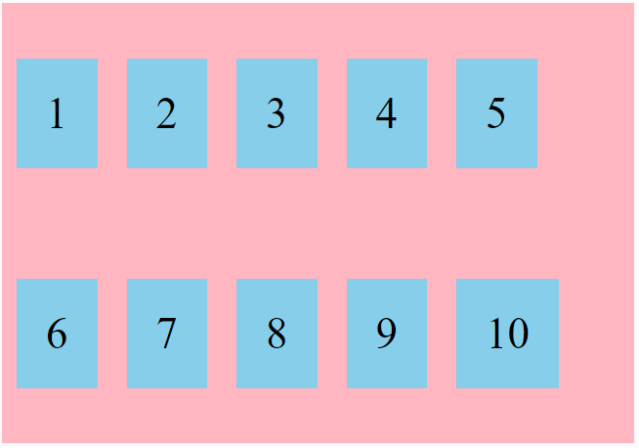
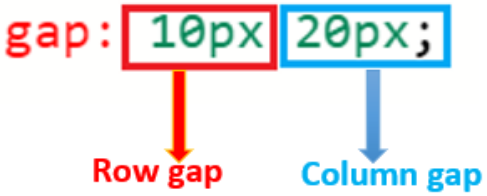
Property	**Value and Description**	
flex-direction	column-reverse	Bottom to top. 
flex-wrap	Specifies whether the flex items should wrap or not when there is no enough space in a line.	
flex-wrap	nowrap	Wrapping will not happen, all flex items will be in one line. 
flex-wrap	wrap	Flex items will be wrapped in multiple lines depending on the number of items from top to bottom. 
flex-wrap	wrap-reverse	Flex items will be wrapped in multiple lines depending on the number of items from bottom to top.

		
Property	**Value and Description**	
flex-flow	Shorthand property for flex-direction and flex-wrap. 	
justify-content	To align flex items on the main axis.	
justify-content	flex-start	Default value. Flex items will be packed towards the start of the flex-direction. 
justify-content	flex-end	Flex items will be packed towards the end of the flex-direction. 
justify-content	center	Flex items will be centered along the container line. 
justify-content	space-between	Flex items will be evenly distributed along the container line.

		
justify-content	space-around	Flex items will be evenly distributed along the container line with equal space around them. 
Property	**Value and Description**	
justify-content	space-evenly	Flex items will be evenly distributed along the container line. The space between all the items and items to edges are even. 
align-items	To align the flex items on the cross axis.	
align-items	flex-start	Flex items will be placed in the start of the cross axis. 
align-items	flex-end	Flex items will be placed at the end of the cross axis. 
align-items	center	Flex items will be placed in the center of the cross axis.

		
align-items	stretch	All flex items will be aligned in the same space in cross axis by stretching the container. 
Property	**Value and Description**	
align-items	baseline	Baseline of the flex items will be aligned. 
align-content	To align the flex lines. It will be effective only on multi-line flex containers where the flex-wrap property is either wrap or wrap-reverse. Similar to align-items but here the flex lines are aligned.	
align-content	flex-start	

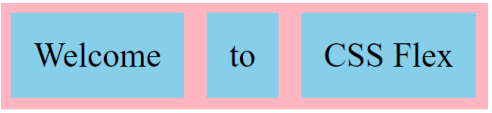
align-content	flex-end	
Property	**Value and Description**	
align-content	center	
align-content	stretch	

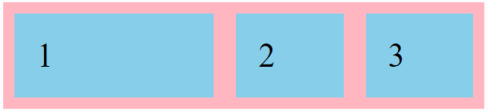
align-content	space-between	
Property	**Value and Description**	
align-content	space-around	
gap	To control space between flex items. 	

Flex item properties:

skill academy
by testbook

Property	Value	Description

order	Value must be a number. Default value is 0.
	`<div style="order: 2"> to</div>`
	`<div style="order: 3">CSS Flex</div>`
	`<div style="order: 1">Welcome</div>`
		Specifies the order of flex items.
		 

flex-grow	Value must be a number. Default value is 0.
	`<div style="flex-grow: 3">1</div>`
	`<div style="flex-grow: 1">2</div>`
	`<div style="flex-grow: 1">3</div>`
		To make the flex item grow with respect to given value.
		 

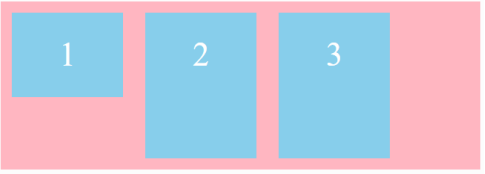
flex-shrink	Value must be a number. Default value is 1. If the value is given as 0, it won't let the flex item shrink.
	`<div style="flex-shrink: 8">1</div>`
	`<div style="flex-grow: 1">2</div>`
	`<div style="flex-grow: 1">3</div>`
		To make the flex item shrink with respect to given value.
		 

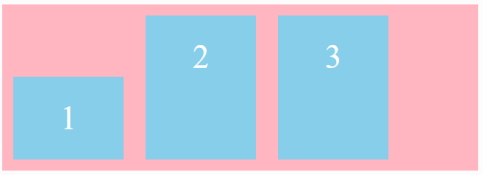
Property	**Value**	**Description**

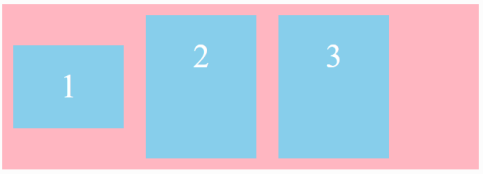
flex-basis	Length (px, %, rem, etc…)
	`<div style="flex-basis: 200px">1</div>`
	`<div>2</div>`
	`<div>3</div>`
		The value will be assigned as the default size of the flex item and the remaining space will be distributed.
		 

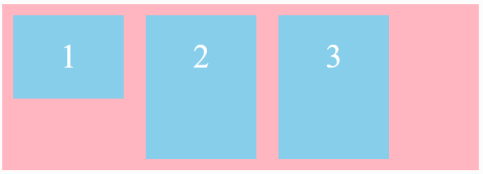
flex	Shorthand property for flex-grow, flex-shrink, flex-basis.
	`<div>1</div>`
	`<div style="flex: 0 8 0px">2</div>`
	`<div>3</div>`	

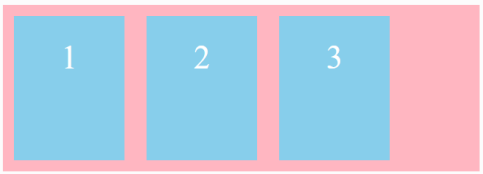
align-self	Specifies the alignment for selected flex items inside the container.	

align-self	flex-start
	`<div style="align-self: flex-start">1</div>`
	`<div>2</div>`
	`<div>3</div>`
		

align-self	flex-end
	`<div style="align-self: flex-end">1</div>`
	`<div>2</div>`
	`<div>3</div>`
		

align-self	center
	`<div style="align-self: center">1</div>`
	`<div>2</div>`
	`<div>3</div>`
		

align-self	baseline
	`<div style="align-self: baseline">1</div>`
	`<div>2</div>`
	`<div>3</div>`
		

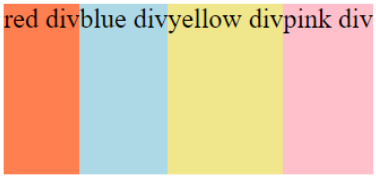
align-self	stretch
	`<div style="align-self: stretch">1</div>`
	`<div>2</div>`
	`<div>3</div>`
		

CSS Flexbox - Practice code

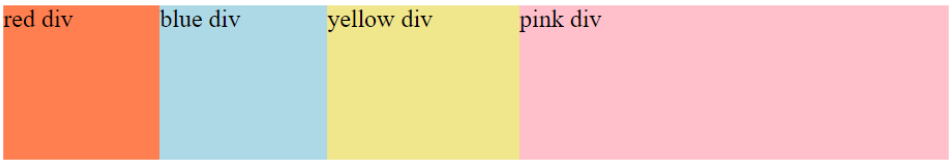
Problem Statement 1: Flex Grow

- Create a web page with a div section that has four div sections in it and apply different colors to them.

- Apply display: flex property to the main div section.

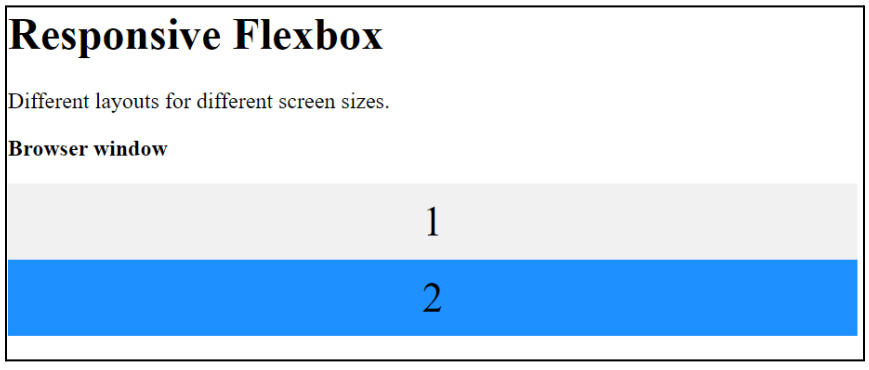


- Add flex-grow properties to each div section and make the web page look good.



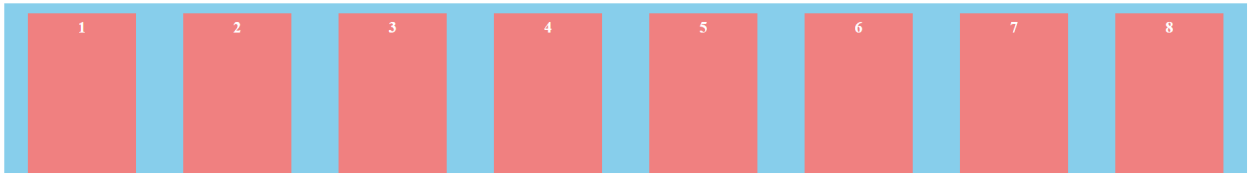
Problem Statement 2: Responsive Flexbox

Create a responsive flexbox design using display:flex, flex-wrap, text-align, flex, and box-sizing properties.



Problem Statement 3: Flex Container

Create a container similar to the following image, with help of flex properties.



Problem Statement 4: Flex Flow

With help of the display, flex-flow properties and div sections create the following output.

Solution:

Problem Statement 1:

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

#mymain {
  width:100%;
  height: 100px;
  display: flex;
}
#bluediv {
flex-grow: 1;
}
#reddiv{
flex-grow: 1;
}
#yellowdiv{
flex-grow: 1;
}
#pinkdiv{
flex-grow: 4;
}
</style>
</head>
<body>

<h1>The flex-grow property</h1>
```

```
<div id="myDIV">
<div id="mymain">
  <div id="reddiv" style="background-color:coral;">red div</div>
  <div id="bluediv" style="background-color:lightblue;">blue div</div>
  <div id="yellowdiv" style="background-color:khaki;">yellow div</div>
  <div id="pinkdiv" style="background-color:pink;">pink div</div>
</div>
</div>

</body>
</html>
```

Problem Statement 2:

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  box-sizing: border-box;
}

.flex-container {
  display: flex;
  flex-wrap: wrap;
  font-size: 30px;
  text-align: center;
}

.flex-item-left {
  background-color: #f1f1f1;
  padding: 10px;
  flex: 50%;
}

.flex-item-right {
  background-color: dodgerblue;
  padding: 10px;
  flex: 50%;
}

/* Responsive layout - makes a one column-layout instead of a two-column
layout */
@media (max-width: 700px) {
  .flex-item-right, .flex-item-left {
    flex: 100%;
  }
}
</style>
</head>
<body>
```

```
<h1>Responsive Flexbox</h1>

<p>Different layouts for different screen sizes.</p>
<p><b>Browser window</b></p>

<div class="flex-container">
  <div class="flex-item-left">1</div>
  <div class="flex-item-right">2</div>
</div>

</body>
</html>
```

Problem Statement 3:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Flex</title>
    <style>
    .flex-container {
      display: flex;
      flex-flow: row wrap;
      justify-content: space-around;
      background: skyblue;
    }

    .flex-container > div {
      background: lightcoral;
      padding: 5px;
      width: 100px;
      height: 150px;
      margin-top: 10px;
      color: white;
      font-weight: bold;
      text-align: center;
    }

    </style>
  </head>
  <body>
    <div class="flex-container">
      <div>1</div>
      <div>2</div>
      <div>3</div>
      <div>4</div>
      <div>5</div>
      <div>6</div>
      <div>7</div>
      <div>8</div>
    </div>
    <script src="script.js"></script>
  </body>
</html>
```

Problem Statement 4:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color:#E7E9EB;
}
#myDIV {
  height:300px;
  background-color:#FFFFFF;
}
#mymain {
  height: 100px;
  display: flex;
  flex-flow: column-reverse wrap-reverse;
}
#mymain div {
  width:60px;
  height:60px;
  padding:5px;
}
</style>
</head>
<body>

<h1>The flex-flow property</h1>

<div id="myDIV">
<div id="mymain">
  <div style="background-color:coral;">red div</div>
  <div style="background-color:lightblue;">blue div</div>
  <div style="background-color:khaki;">yellow div</div>
  <div style="background-color:pink;">pink div</div>
</div>
</div>

</body>
</html>
```

Instagram Logo

With your new gained knowledge in CSS background, width, height, display, position, and flex properties design an instagram logo.

Sample Output:

Sample code:

HTML Code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>replit</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <div class="insta">
      <div class="innerbox"></div>
    </div>
  </body>
</html>
```

CSS Code:

```
.insta{
  width: 150px;
  height: 150px;
  background: radial-gradient(circle at 30% 107%, #fdf497 0%, #fdf497 5%, #fd5949 45%, #d6249f 60%, #285aeb 90%);
  border-radius: 35px;
  display: grid;
  place-items: center;
}
```

```css
.innerbox{
  width: 120px;
  height: 120px;
  border: 10px solid #fff;
  border-radius: 32px;
  display: grid;
  place-items: center;
  position: relative;
}

.innerbox::before{
  content: '';
  width: 45px;
  height: 45px;
  border: 10px solid #fff;
  border-radius: 50%;
  position: absolute;
}

.innerbox::after{
  content: '';
  width: 10px;
  height: 10px;
  border: 2px solid #fff;
  border-radius: 50%;
  background: #fff;
  position: absolute;
  top: 8px;
  right:10px;
}
```

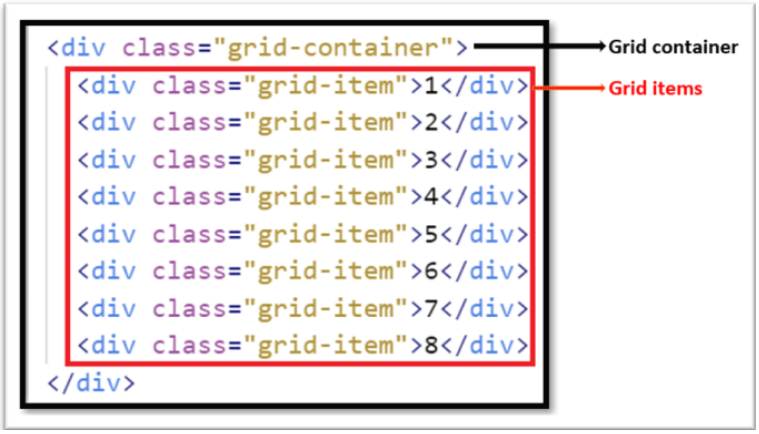
CSS Grid

Topics Covered:

1. What is CSS Grid?
2. Important terminologies.
3. Grid properties.
    a. Grid container properties.
    b. Grid item properties.
4. Difference between CSS Flex and CSS Grid.

CSS Grid:

- CSS grid is a two-dimensional layout model with rows and columns to design web pages.
- CSS grid avoids using gloat and positioning.
- The grid layout must have a **parent element(container)** with the **display** property "**grid**" or "**inline-grid**". So that the child elements of the container automatically become **grid items**.



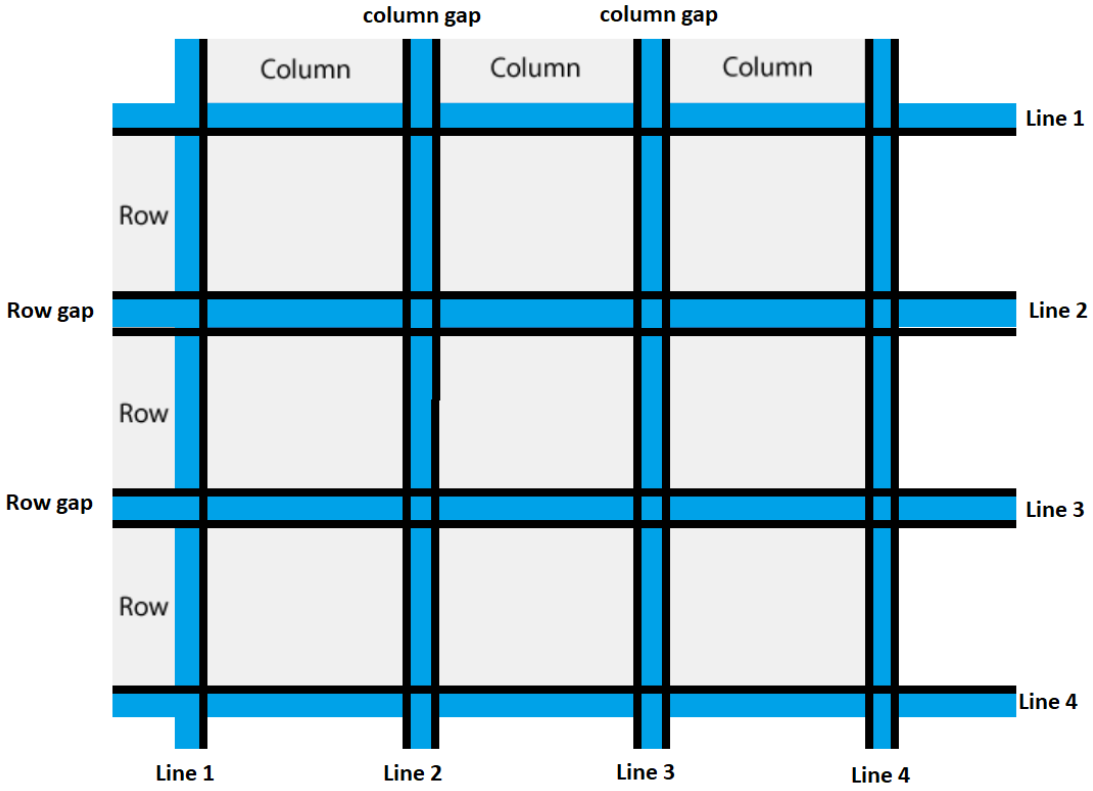
Grid terminologies:

Terminologies	Description
grid columns	The vertical lines of the grid items
grid rows	The horizontal lines of the grid items

grid gaps	The space between each row or column.
	`grid-gap: 50px 100px;`
	The space between each row is called the row gap.
	`grid-row-gap: 50px;`
	The space between each column is called a column gap.
	`grid-column-gap: 50px;`

Terminologies	Description
grid lines	The line between each row or column. The line between each line is called the row line. The line between each line is called the column line. The lines are used to place the items in the container by using the following properties: • grid-column-start • grid-column-end • grid-row-start • grid-row-end

Grid terminologies can be easily understood with help of the following image:

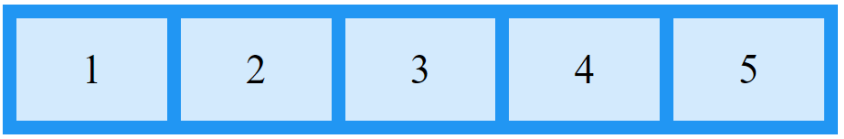


Grid properties:

Grid container properties:

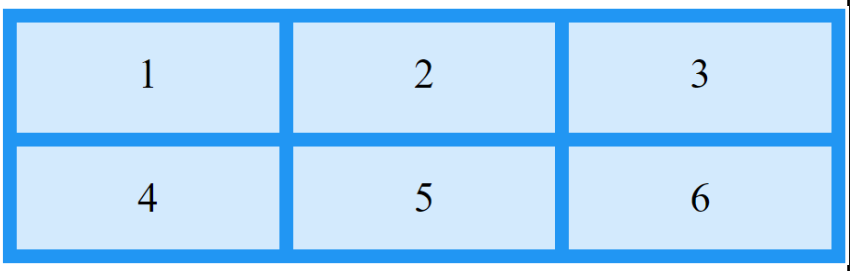
<table>
<tr><th>Property</th><th>Value</th><th>Description</th></tr>
<tr><td rowspan="2">grid-template-columns</td><td>Width of each column or auto.</td><td>Defines the number of columns and their width in the grid.</td></tr>
<tr><td colspan="2">Code:

```
grid-template-columns: auto auto auto auto auto;
```

Output:


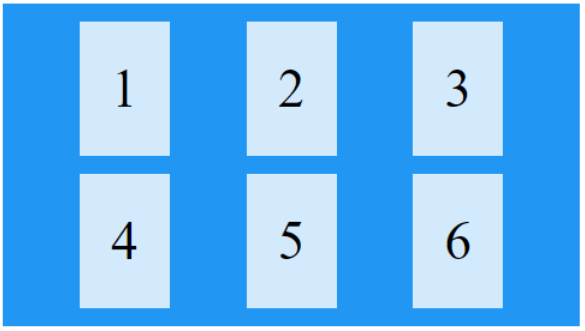
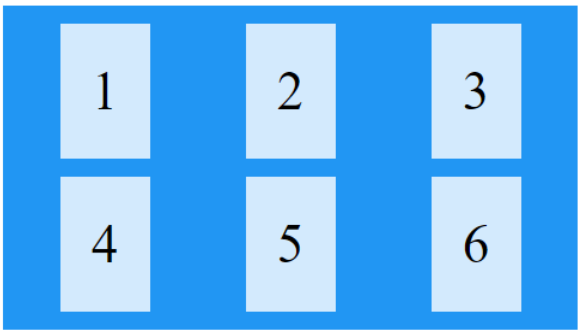
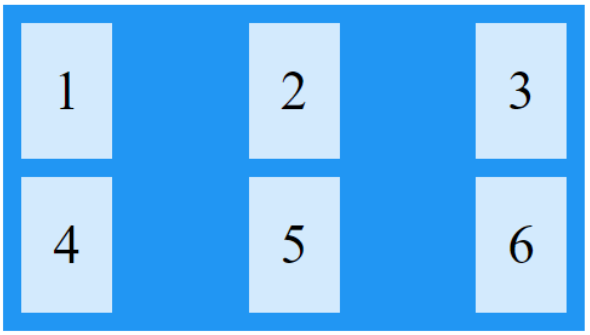
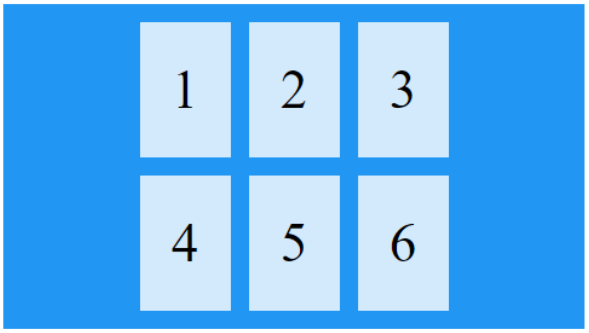
</td></tr>
<tr><td rowspan="2">grid-template-rows</td><td>Height of each row or auto.</td><td>Defines the number of rows and their heights in the grid</td></tr>
<tr><td colspan="2">Code:

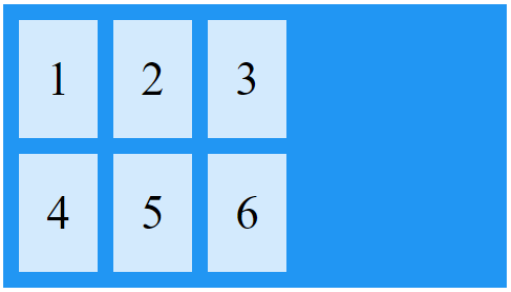
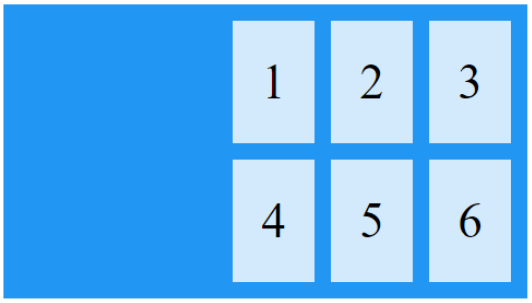
```
grid-template-columns: auto auto auto;
grid-template-rows: 80px auto;
```

Output:


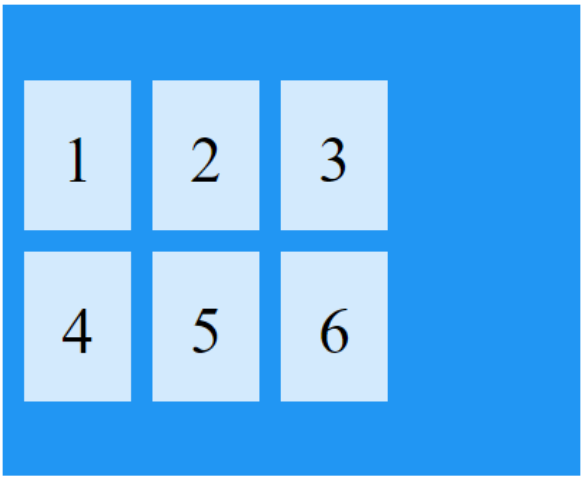
</td></tr>
<tr><td>justify-container</td><td colspan="2">Used to align the whole grid inside the container. The total width of the grid must be less than the container’s width to add this property.

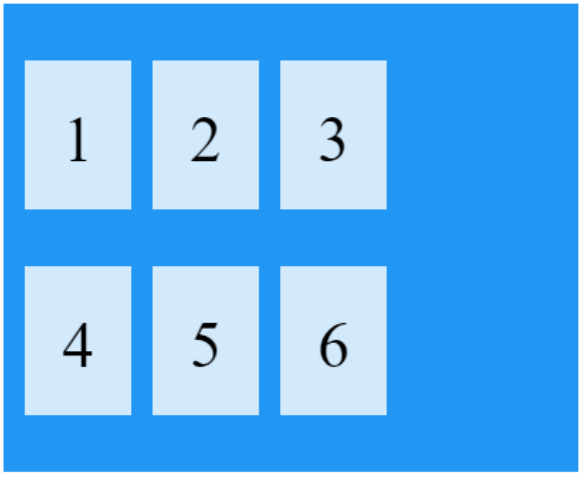
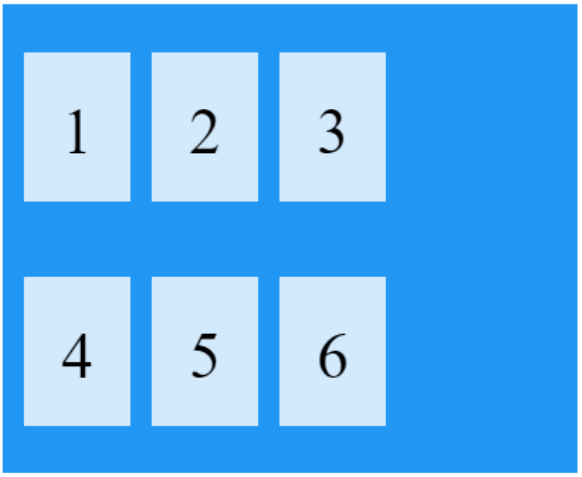
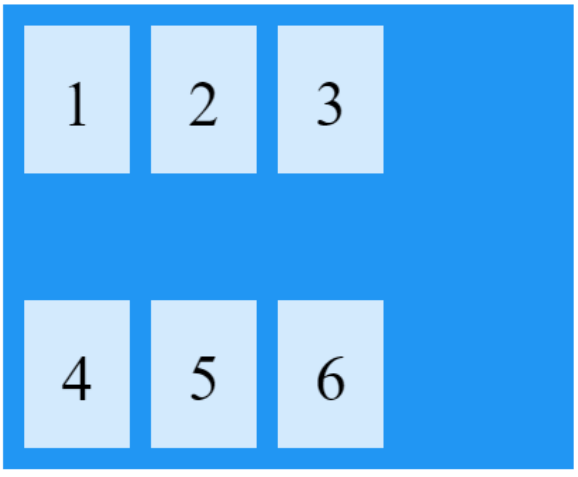
```
.grid-container {
  display: grid;
  width: 300px;
  /*Make the grid smaller than the container*/
  grid-template-columns: 50px 50px 50px;
}
```
</td></tr>
</table>

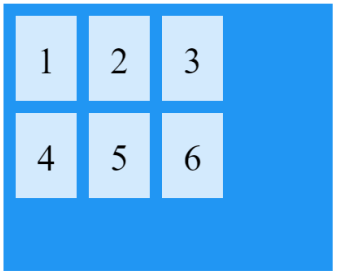
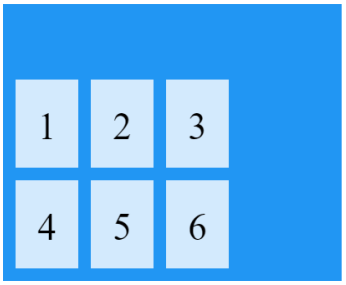
Property	Value	Description
justify-container	space-evenly	Gives an equal amount of space between and around the columns. 
justify-container	space-around	Gives an equal amount of space around the columns. 
justify-container	space-between	Gives an equal amount of space between the columns. 
justify-container	center	Aligns the grid in the middle of the container. 

Property	Value	Description
justify-container	start	Aligns the grid at the beginning of the container. 
justify-container	end	Aligns the grid at the end of the container. 

align-content	Used to align the grid items vertically inside the container. The total height of the grid items must be less than the container's height to add this property.
	`width: 250px;`
	`grid-template-columns: 50px 50px 50px;`
	`height: 200px;`
	`grid-template-rows: 70px 70px;`	

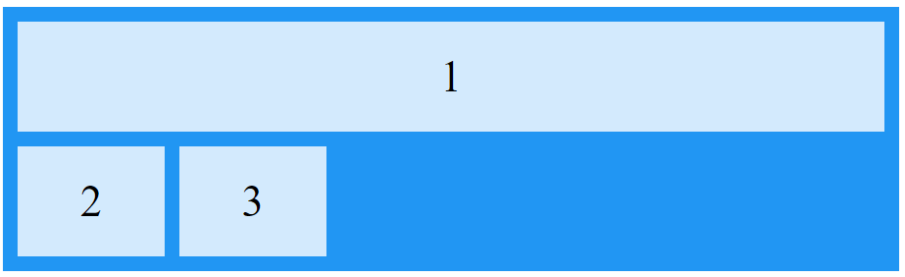
align-content	center	Aligns the rows in the center of the container. 

Property	Value	Description
align-content	space-evenly	Gives an equal amount of space between and around the rows. 
align-content	space-around	Gives an equal amount of space around the rows. 
align-content	space-between	Gives an equal amount of space between the rows. 

Property	Value	Description
align-content	start	Aligns the row at the beginning of the container. 
align-content	end	Aligns the row at the end of the container. 

Gird items property:

Property	Value	Description
grid-column	Column line numbers	Define where the item will start and end vertically. Shorthand property of grid-column-start and grid-column-end.

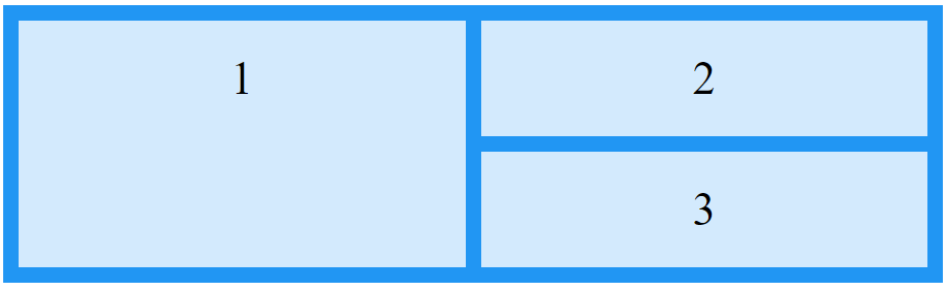
	Code:
	`grid-template-columns: auto auto auto auto auto auto;`
	`<div class="grid-container"> <div class="item1">1</div> <div class="item2">2</div> <div class="item3">3</div> </div>`
	`.item1 { grid-column: 1 / 7; }`
	• 6 columns, 3 items, 7 column lines, and 3 row lines.
	Output:
	 	

<table>
<tr><th>Property</th><th>Value</th><th>Description</th></tr>
<tr><td rowspan="3">grid-row</td><td>Row line numbers</td><td>Define where the item will start and end horizontally. Shorthand property of grid-row-start and grid-row-end.</td></tr>
<tr><td colspan="2">Code:

```
grid-template-columns: auto auto;
```

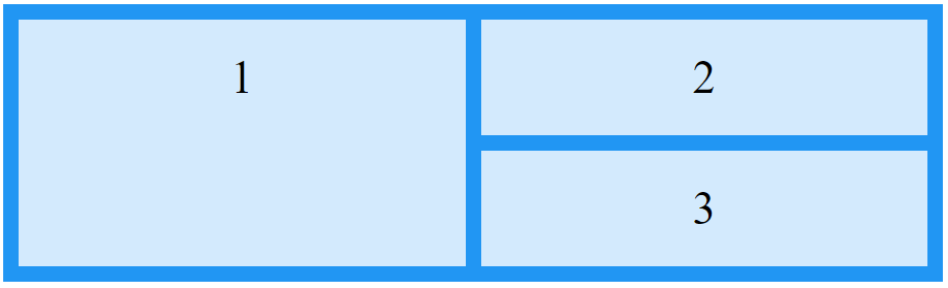
```
<div class="grid-container">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
</div>
```

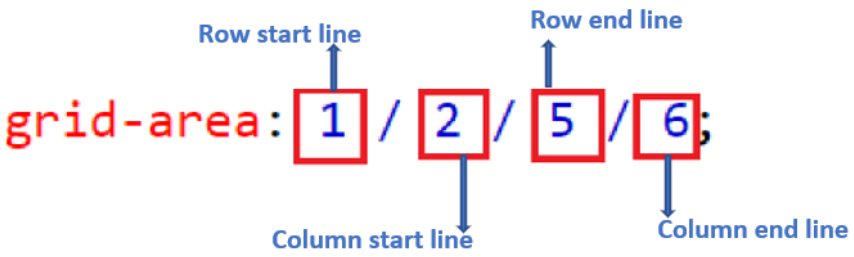
```
.item1 {
  grid-row: 1 / 3;
}
```

2 columns, 3 items, 3 column lines, and 3 row lines
Output:


</td></tr>
<tr><td colspan="2">Span can be used instead of line numbers.

```
.item1 {
  grid-row: 1 / span 2;
}
```



</td></tr>
<tr><td>grid-area</td><td colspan="2">Shorthand property of grid-column-start, grid-column-end, grid-row-start, and grid-row-end.


</td></tr>
</table>

Difference between CSS Flex and CSS Grid:

CSS Flex	CSS Grid
One-dimensional layout can work on either row or column at a time.	Two-dimensional layout can work on rows and columns at a time.
Best suitable for application components and small-scale layouts.	Designed for large-scale layouts that are non-linear in design.
Helps to align contents.	Helps to design the outer layout of the web page.
Content first.	Layout first.

CSS Grid - Practice code

Problem Statement 1: Grid Definition

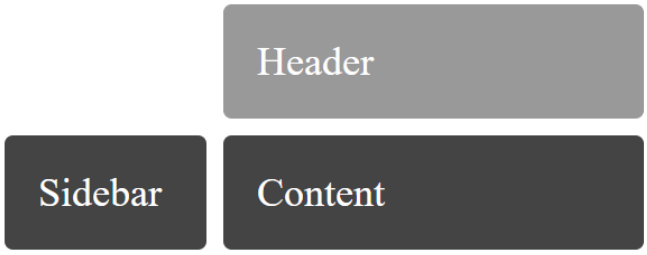
Design the following output, by using display:grid, grid-column-template, grid-gap, background-color, color, and box-model properties.

Problem Statement 2: Spanning Tracks

To create Grid Areas that are larger than a single grid track we specify an end line that is more than one track away, using grid-column and grid-row shorthand.

Problem Statement 3: Grid Areas

Design the following output by using grid-area, grid-gap, grid-template-columns, and grid-template-areas properties.



Problem Statement 4: Grid Lines

Design the following output using grid-area, grid-gap, grid-template-columns, grid-template-areas, grid-column and grid-row properties.

Solution

Problem Statement 1:

HTML:

```
<div class="wrapper">
  <div class="box a">A</div>
  <div class="box b">B</div>
  <div class="box c">C</div>
  <div class="box d">D</div>
  <div class="box e">E</div>
  <div class="box f">F</div>
</div>
```

CSS:

```
body {
  margin: 40px;
}

.wrapper {
  display: grid;
  grid-template-columns: 100px 100px 100px;
  grid-gap: 10px;
  background-color: #fff;
  color: #444;
}

.box {
  background-color: #444;
  color: #fff;
  border-radius: 5px;
  padding: 20px;
  font-size: 150%;
}
```

Problem Statement 2:

HTML:

```
<div class="wrapper">
  <div class="box a">A</div>
  <div class="box b">B</div>
  <div class="box c">C</div>
  <div class="box d">D</div>
</div>
```

CSS:

```
body {
  margin: 40px;
}

.wrapper {
    display: grid;
    grid-gap: 10px;
    grid-template-columns: 100px 100px 100px;
    background-color: #fff;
    color: #444;
  }

  .box {
    background-color: #444;
    color: #fff;
    border-radius: 5px;
    padding: 20px;
    font-size: 150%;

  }

  .a {
    grid-column: 1 / 3;
    grid-row: 1;
  }
  .b {
    grid-column: 3 ;
    grid-row: 1 / 3;
  }
  .c {
    grid-column: 1 ;
    grid-row: 2 ;
  }
  .d {
    grid-column: 2;
    grid-row: 2;
}
```

Problem Statement 3:

HTML:

```
<div class="wrapper">
  <div class="box header">Header</div>
  <div class="box sidebar">Sidebar</div>
  <div class="box content">Content</div>
</div>
```

CSS:

```
.header {
    grid-area: header;
  }


  .wrapper {
    display: grid;
    grid-gap: 10px;
    grid-template-columns: 120px  120px  120px;
    grid-template-areas:
                "....... header  header"
                "sidebar content content";
    background-color: #fff;
    color: #444;
  }
.box {
  background-color: #444;
  color: #fff;
  border-radius: 5px;
  padding: 20px;
  font-size: 150%;
}
.header {
  background-color: #999;
}
```

Problem Statement 4:

HTML:

```
<div class="wrapper">
  <div class="box header">Header</div>
  <div class="box sidebar">Sidebar</div>
  <div class="box content">Content</div>
  <div class="overlay">overlay</div>
</div>
```

CSS:

```
.sidebar {
    grid-area: sidebar;
  }

  .content {
    grid-area: content;
  }

  .header {
    grid-area: header;
  }


  .wrapper {
    display: grid;
    grid-gap: 10px;
    grid-template-columns: 120px 120px 120px;
    grid-template-areas:
    "header  header  header"
    "sidebar content content";
    background-color: #fff;
    color: #444;
  }

  .box {
    background-color: #444;
    color: #fff;
    border-radius: 5px;
    padding: 20px;
    font-size: 150%;

  }

  .header {
    background-color: #999;
  }

  .overlay {
    background-color: red;
    z-index: 10;
    grid-column: content-start / content-end;
    grid-row: header-start / content-end;
  }
```

CSS Combinators

Topics Covered:

- CSS Combinators.
- Types of Combinators.
    - Descendant selector.
    - Child selector.
    - Adjacent Sibling selector.
    - General Sibling selector.

Topics in Detail:

CSS Combinators:

- Combinator explains the relationship between the selectors(CSS selectors are the patterns used to select the elements for style purposes).
- There can be more than one simple selector or complex selector in a **CSS selector**, and between these selectors, we can include a combinator
- Combinators combine the selectors to provide them a useful relationship and the position of content in the document.
- A CSS Combinator explains the relationship between one or more simple selectors.

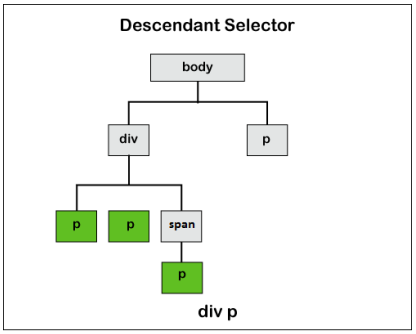
Types of Combinator

- Descendant Selector
- Child Selector
- Adjacent Sibling Selector
- General Sibling Selector

Selector	Symbol	Description
Descendant	Space	Selects all descendent elements of specific element
Child	>	Selects all child elements of specific element
Adjacent Sibling	+	Selects an element that is immediately following the specific element
General Sibling	~	Selects all next sibling elements following the specific element

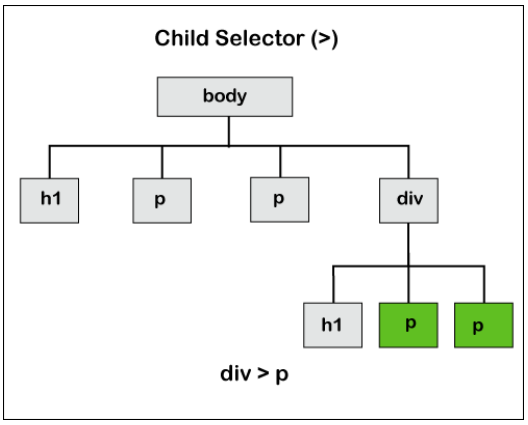
Descendant Selector (Space)

- Descendant Selector selects **all descendant** elements of a specific element.
- It can be a **direct** child or **deeper** than five levels, but it will still be referred to as a **descendant**.
- It combines two selectors in which the **first** selector represents an **ancestor** (parent, parent's parent, etc.), and the **second** selector represents **descendants**. The elements matched by the second selector are selected if they have an **ancestor** element that matches the first selector.



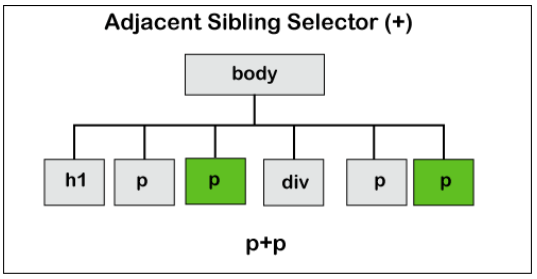
Child Selector (>)

- Child Selector selects **all children elements** of a specific element.
- It selects the **direct** descendant of the parent.
- This combinator only matches the elements that are the **immediate child** in the document tree.
- It is **stricter** as compared to the **descendant selector** because it selects the second selector only when the first selector is its parent.



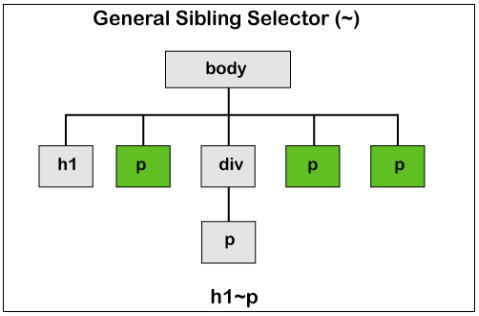
Adjacent Sibling Selector (+)

- Selects an element that is **immediately** following the specific element.
- It matches the second element only when the element immediately follows the first element, and both of them are the children of the **same** parent.



General Sibling Selector (~)

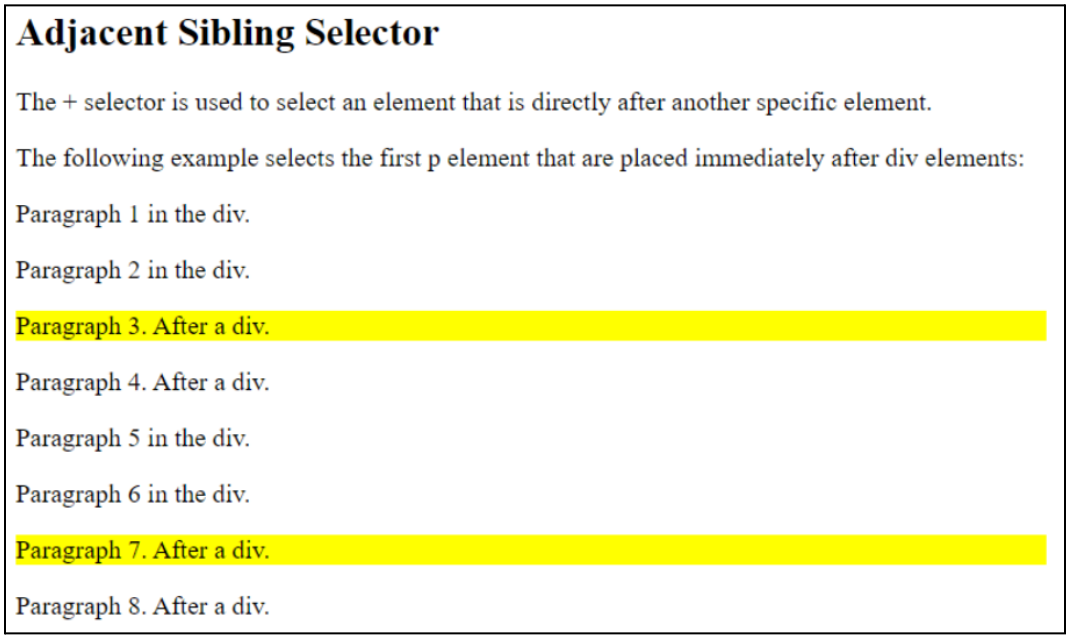
- Selects all the elements that follow the elements of the first selector, and both of them are children of the **same** parent.
- It can be used for selecting the **group** of elements that share the **common** parent element.



CSS Combinators - Practice Code

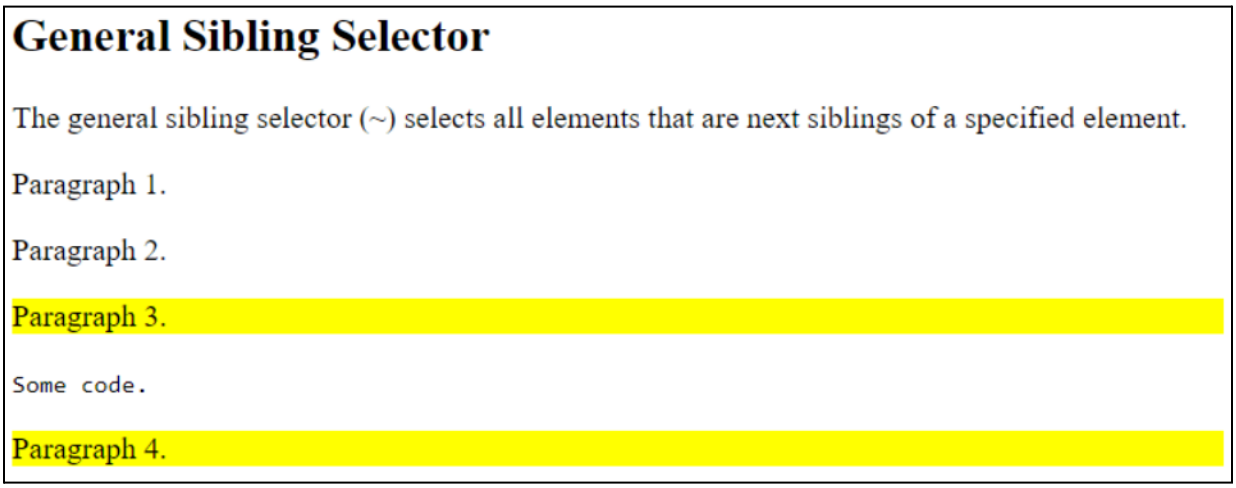
Problem Statement 1: Adjacent Sibling Selector

With help of the adjacent sibling selectors apply styles to paragraph elements that are directly followed by the div element in the HTML code.



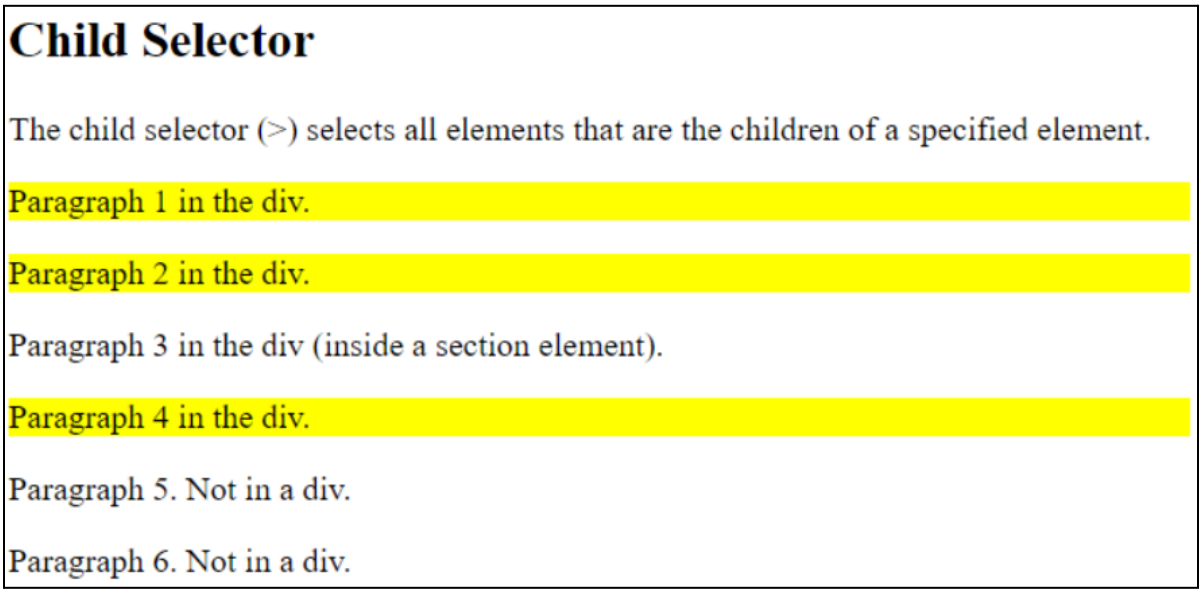
Problem Statement 2: General Sibling Selector

With help of the general sibling selector apply styles to all paragraph elements that are next siblings of a div element.



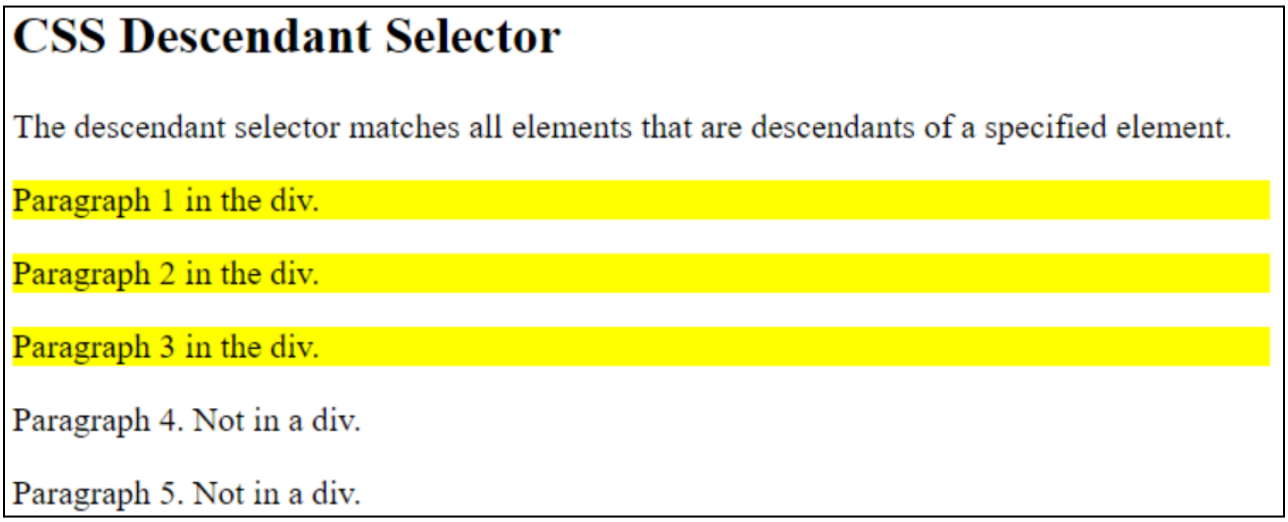
Problem Statement 3: Child Combinator Selector

With help of the child combinator selector apply styles to all paragraph elements that are the children of a div element.



Problem Statement 4: Descendant Combinator Selector

With help of descendant Combinator Selector apply styles to all paragraph elements that are descendants of a div element.



Solution:

Problem Statement 1:

```
<!DOCTYPE html>
<html>
<head>
<style>
div + p {
  background-color: yellow;
}
</style>
</head>
<body>

<h2>Adjacent Sibling Selector</h2>

<p>The + selector is used to select an element that is directly after
another specific element.</p>
<p>The following example selects the first p element that are placed
immediately after div elements:</p>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
</div>

<p>Paragraph 3. After a div.</p>
<p>Paragraph 4. After a div.</p>

<div>
  <p>Paragraph 5 in the div.</p>
  <p>Paragraph 6 in the div.</p>
</div>

<p>Paragraph 7. After a div.</p>
<p>Paragraph 8. After a div.</p>

</body>
</html>
```

Problem Statement 2:

```
<!DOCTYPE html>
<html>
<head>
<style>
div ~ p {
  background-color: yellow;
}
</style>
</head>
<body>
```

```
<h2>General Sibling Selector</h2>

<p>The general sibling selector (~) selects all elements that are next
siblings of a specified element.</p>

<p>Paragraph 1.</p>

<div>
  <p>Paragraph 2.</p>
</div>

<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>

</body>
</html>
```

Problem Statement 3:

```
<!DOCTYPE html>
<html>
<head>
<style>
div > p {
  background-color: yellow;
}
</style>
</head>
<body>

<h2>Child Selector</h2>

<p>The child selector (>) selects all elements that are the children of a
specified element.</p>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section>
    <!-- not Child but Descendant -->
    <p>Paragraph 3 in the div (inside a section element).</p>
  </section>
  <p>Paragraph 4 in the div.</p>
</div>

<p>Paragraph 5. Not in a div.</p>
<p>Paragraph 6. Not in a div.</p>

</body>
</html>
```

Problem Statement 4:

```
<!DOCTYPE html>
<html>
<head>
<style>
div p {
  background-color: yellow;
}
</style>
</head>
<body>

<h2>CSS Descendant Selector</h2>

<p>The descendant selector matches all elements that are descendants of a
specified element.</p>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```

CSS Animations

Topics Covered

- Keyframes
- Animation
    - animation-name
    - animation-duration
    - animation-delay
    - animation-direction

Topics in Detail:

CSS Animators

- CSS Animations is a technique used to **change** the **appearance** and **behavior** of various elements in web pages.
- It will control the elements by changing their motions or display.
- It has **replaced** the animation created by **Flash** and **JavaScript**.
- The animation is created using the **@keyframe** rule.
- It has two parts,
    - **CSS Properties** (describe the animation of the elements)
    - **keyframes (**specific **time intervals** at which the animations have to occur)
- When the animation is created in the **@keyframe** rule, it must have a **selector** otherwise, the animation will have **no effect**.

@keyframe

- Keyframes are the **foundation** of CSS Animations.
- It will control the **intermediate steps** in a CSS animation sequence.
- It defines the display of animation at the corresponding stages in the whole duration.

Animation Properties

- animation-name
- animation-duration
- animation-delay
- animation-direction
- animation-iteration-count
- animation-timing-function
- animation-fill-mode

animation-name:

- It is used to describe the name of the **@keyframe** that has the CSS animation sequence.
- Syntax: ***animation-name: animation _name;***

animation-duration:

- It specifies the **time duration** of the animation to complete **one** cycle.
- If animation-duration is **not mentioned**, **no animation** will occur because the **default** value is **0 seconds**.
- We can specify animation-duration by using the keywords "**from**" and "**to**" (which represents 0% (start) and 100% (complete)). Instead, we can also use **percent**.

animation-delay:

- It specifies the **delay** when the animation should start.
- It allows **Negative** values. If using negative values, the animation will be playing as if it has started **already** before *N* seconds.

animation-iteration-count:

- It specifies the **number of times** an animation should run.
- If we specify the animation-iteration-count value as **infinite**, the animation will repeat **indefinitely**.

animation-direction:

- It specifies the direction of the animation

Values	Description
normal (default)	The animation is played **forward**
reverse	The animation is played in the **reverse** direction i.e. backward
alternate	The animation is played forwards first, and then backward
alternate-reverse	The animation is played backward first, and then forwards

animation-timing-function:

- It is used to specify the **speed curve** of the animation.

Values	Description
ease (default)	The animation starts slowly, then fast, and then finally ends slowly
linear	The animation plays with the same speed from start to end
ease-in	The animation plays with a slow start
ease-out	The animation plays with a slow end
ease-in-out	The animation starts and ends slowly
cubic-bezier(n,n,n,n)	Lets you define your own values in a cubic-bezier function

animation-fill-mode:

- CSS animations **will not** affect an element before the first keyframe is played or after the last keyframe is played. This behavior can be **overridden** by the **animation-fill-mode** property.
- It is used to specify the **style** for the element when the animation is **not playing**.

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

animation-play-state:

- This allows you to **play/pause** the animation.

Animation Shorthand Property:

- A **shorthand property** for setting all the animation properties.
- The properties should be in the following order
- Syntax:
  animation: [animation-name] [animation-duration] [animation-timing-function] [animation delay] [animation-iteration-count] [animation-direction] [animation-fill-mode] [animation-play-state];

- Example: Without shorthand property:

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

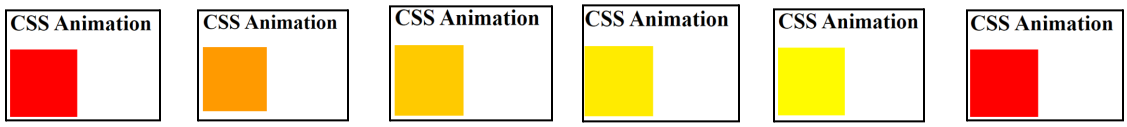
- Example: With shorthand property:

```
div {
    animation: example 5s linear 2s infinite alternate;
}
```

CSS Animation - Practice code

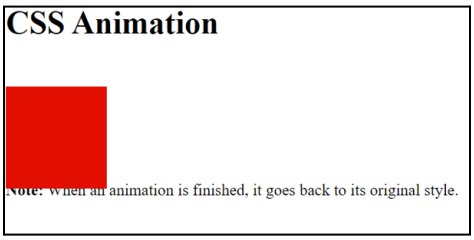
Problem Statement 1: Keyframes

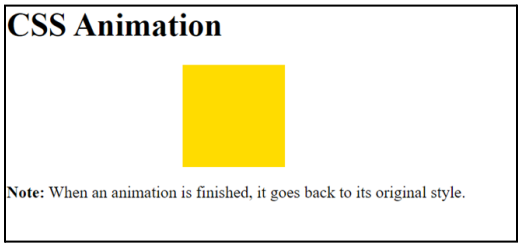
Create a box, whose background color changes slowly from color to other color and when an animation is finished, it goes back to its original original.

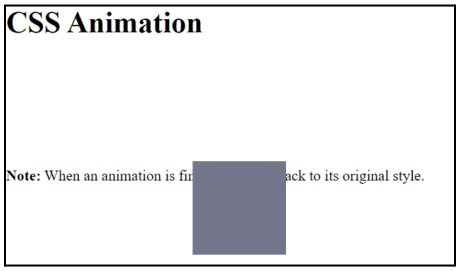


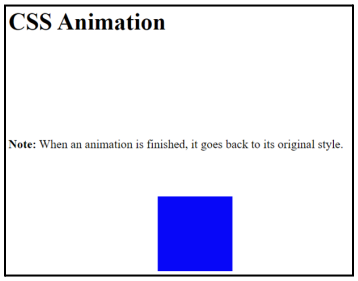
Problem Statement 2: Keyframes

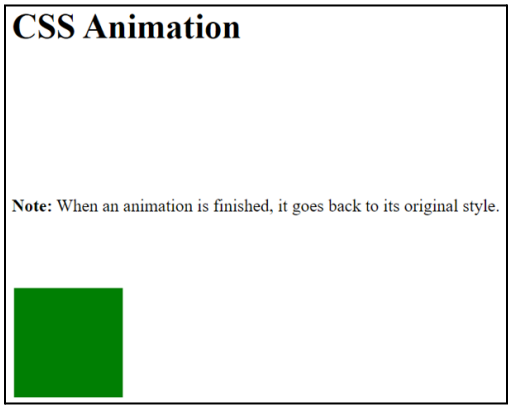
Create a box that changes color and moves with help of keyframe, animation-name, animation-duration, background-color, top and left properties.

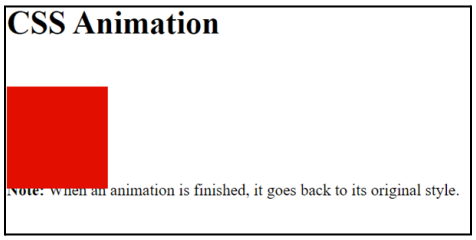






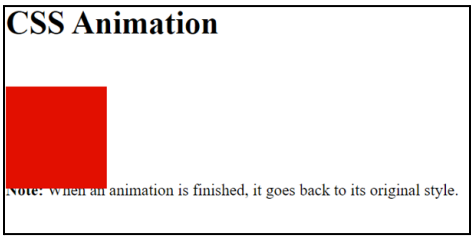


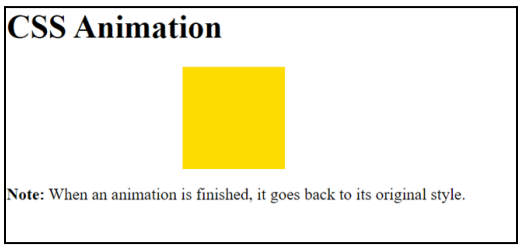


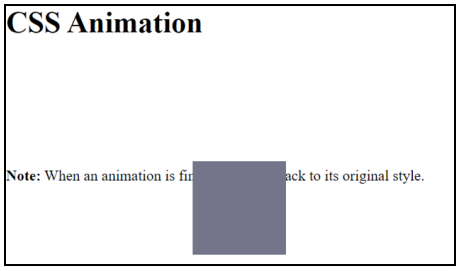


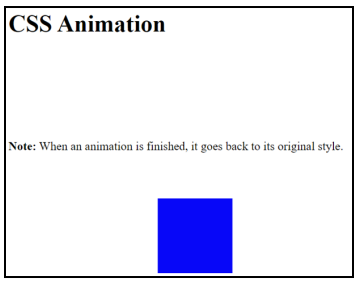
Problem Statement 3: Animation delay

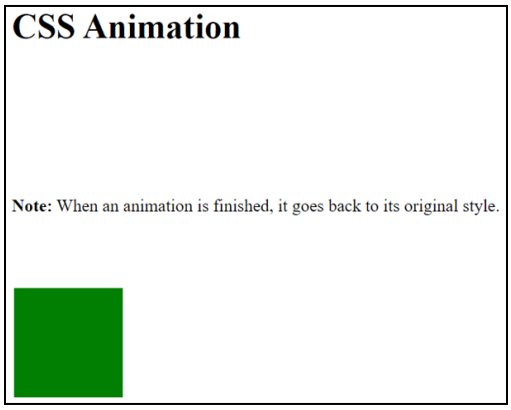
Create a box that changes color and moves with help of keyframe, animation-name, animation-duration, background-color, top and left properties. And delay the animation before starting using animation-delay property.

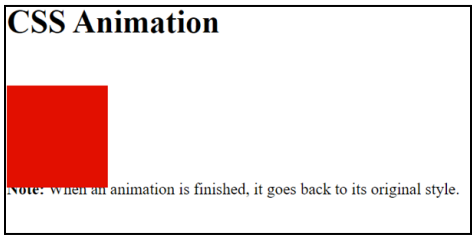






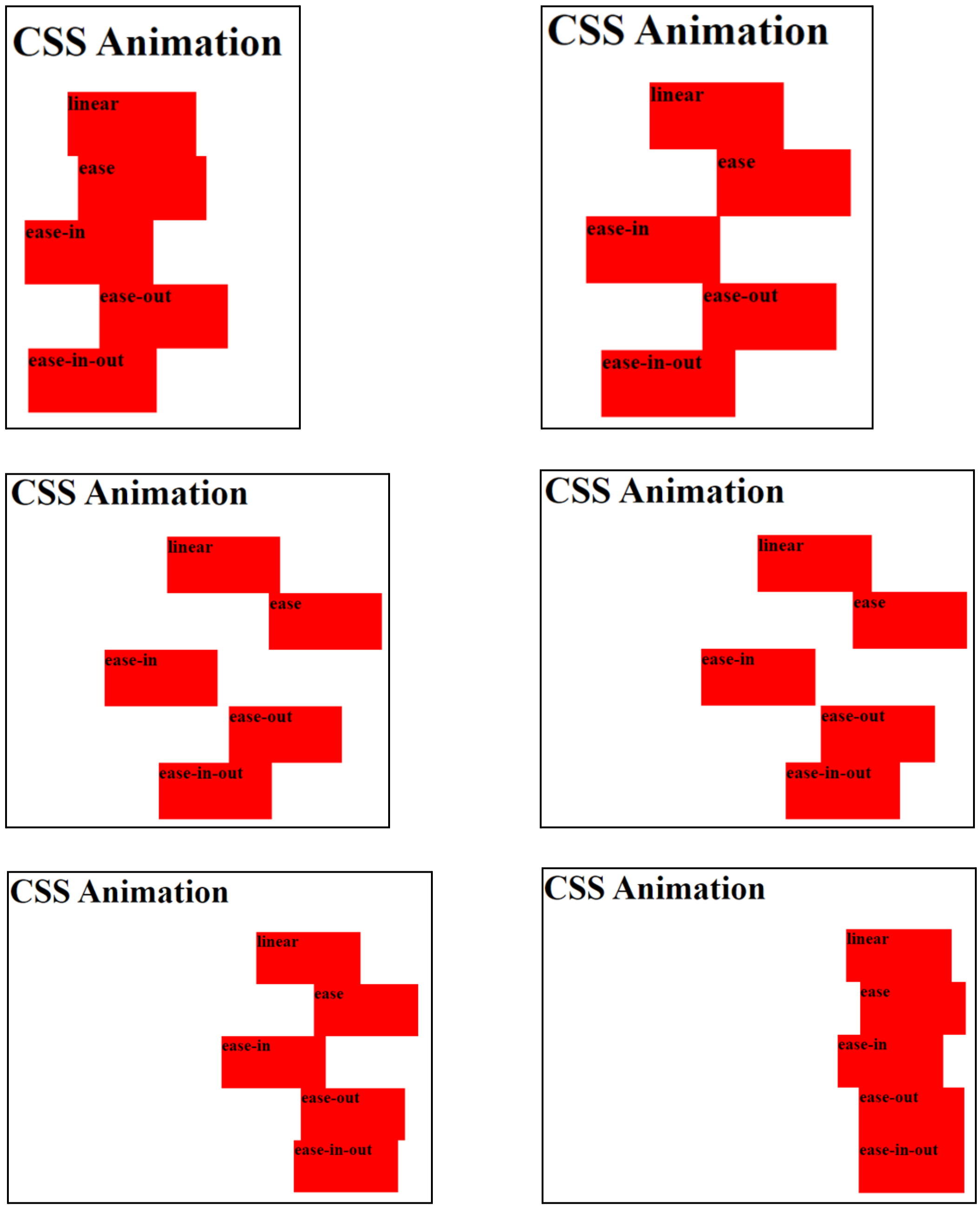






Problem Statement 4: Speed curve

Using animation-timing-function property, specify the speed curve of the animation and create the following animation.



Solution

Problem Statement 1:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}

@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
</style>
</head>
<body>

<h1>CSS Animation</h1>
<div></div>
</body>
</html>
```

Problem Statement 2:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
}

@keyframes example {
  0%   {background-color:red; left:0px; top:0px;}
  25%  {background-color:yellow; left:200px; top:0px;}
  50%  {background-color:blue; left:200px; top:200px;}
  75%  {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>
```

```
<h1>CSS Animation</h1>

<div></div>

<p><b>Note:</b> When an animation is finished, it goes back to its original
style.</p>

</body>
</html>
```

Problem Statement 3:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
  animation-delay: 2s;
}

@keyframes example {
  0%   {background-color:red; left:0px; top:0px;}
  25%  {background-color:yellow; left:200px; top:0px;}
  50%  {background-color:blue; left:200px; top:200px;}
  75%  {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>

<h1>CSS Animation</h1>

<div></div>

<p><b>Note:</b> When an animation is finished, it goes back to its original
style.</p>
</body>
</html>
```

Problem Statement 4:

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

```
  width: 100px;
  height: 50px;
  background-color: red;
  font-weight: bold;
  position: relative;
  animation: mymove 5s infinite;
}

#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>
```

CSS Navigation Bar

Topics Covered

- Navigation bar in CSS
    - Vertical Navigation Bar
    - Horizontal Navigation Bar
- Responsive Web Design
    - Viewport
    - Media Queries
    - Fluid Grids
    - Flexible Visuals

Topics in Detail

Navigation Bar in CSS

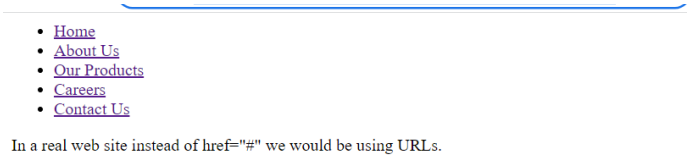
- Navigation bar comes under **Graphical User Interface.**
- It helps the webpage user to access information by **navigating** to other sections of the website using **links**.
- Usually, the navigation bar is on the **top** of the web page as a **horizontal list of links**.
- It can be placed **below the header or logo**, but it must always be placed **before** the main content.
- A website with **easy-to-use** navigation bar allows the user to visit any section easily and quickly.
- **Standard HTML** is the base for the Navigation Bar.
- Basically, **Navigation Bar** is a **list of links**.

Example

In this code let us try to list the menu by using <ul> and <li> tags.

```
<html>
<!--  <header>…
<body>
    <ul>
        <li><a href="#home">Home</a></li>
        <li><a href="#about">About Us</a></li>
        <li><a href="#products">Our Products</a></li>
        <li><a href="#careers">Careers</a></li>
        <li><a href="#conatct">Contact Us</a></li>
    </ul>
    <p>In a real web site instead of href="#" we would be using URLs.</p>
</body>
</html>
```

Output



- Navigation Bar should not have list markers so we set ***list-style-type: none;*** to remove the bullets.
- To remove browser default settings, we set ***margin: 0;*** and ***padding: 0;***

Types of Navigation Bar

- Vertical Navigation Bar
- Horizontal Navigation Bar

Vertical Navigation

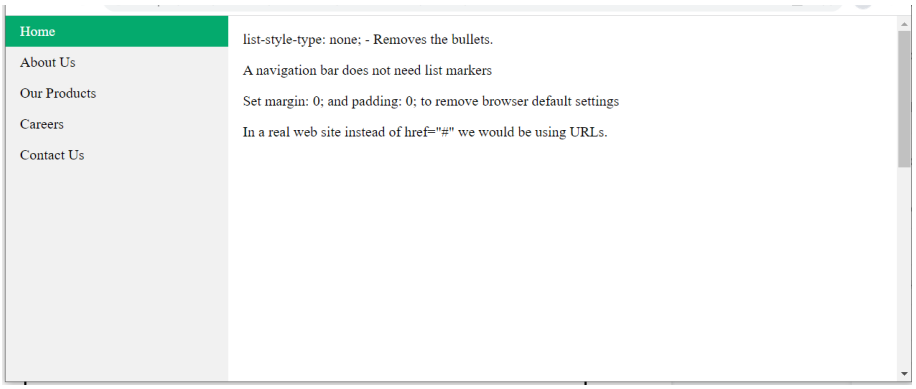
HTML

```
<body>
    <ul>
        <li><a class="active" href="#home">Home</a></li>
        <li><a href="#about">About Us</a></li>
        <li><a href="#products">Our Products</a></li>
        <li><a href="#careers">Careers</a></li>
        <li><a href="#conatct">Contact Us</a></li>
    </ul>
    <div style="margin-left:25%;padding:1px 16px;height:1000px;">
        <p>list-style-type: none; - Removes the bullets.</p>
        <p> A navigation bar does not need list markers</p>
        <p>Set margin: 0; and padding: 0; to remove browser default settings</p>
    <p>In a real web site instead of href="#" we would be using URLs.</p></div>
</body>
```

Style

```
body {margin: 0;}

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 25%;
  background-color: #f1f1f1;
  position: fixed;
  height: 100%;
  overflow: auto;
}

li a {
  display: block;
  color: #000;
  padding: 8px 16px;
  text-decoration: none;
}

li a.active {
  background-color: #04AA6D;
  color: white;
}

li a:hover:not(.active) {
  background-color: #555;
  color: white;
}
```

Output



- ***display: block;*** - We display the links as block elements which allows us to specify the width, padding, margin, height, etc. In the Vertical Navigation Bar, the menu should be a part of a web page.
- ***width: 25%;*** - By default block elements take up the full width available. So We have to specify the width. The width can be set in the <ul> tag as well.

```
li a {
  display: block;
  color: #000;
  width: 25%;
  padding: 8px 16px;
  text-decoration: none;
}
```

- Set the navigation bar **background color** to **gray** and when the user moves the cursor over the menu the background color of the link has to change to **dark gray**.

```
li a:hover {
  background-color: #555;
  color: white;
}
```

- **Active/Current Navigation Link**
  - We have added an "**active**" class to the current link so that the user will get to know the current web page he is viewing.

```
li a.active {
  background-color: #04AA6D
  color: white;
}
```

- Set ***overflow: auto;*** to enable scroll if the side nav has more content.

- We can make the Vertical Navigation Bar **stick** to one position even on a **scroll** by setting ***position: fixed;***

```css
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 25%;
  background-color: #f1f1f1;
  position: fixed;
  height: 100%;
  overflow: auto;
}
```

Horizontal Navigation Bar

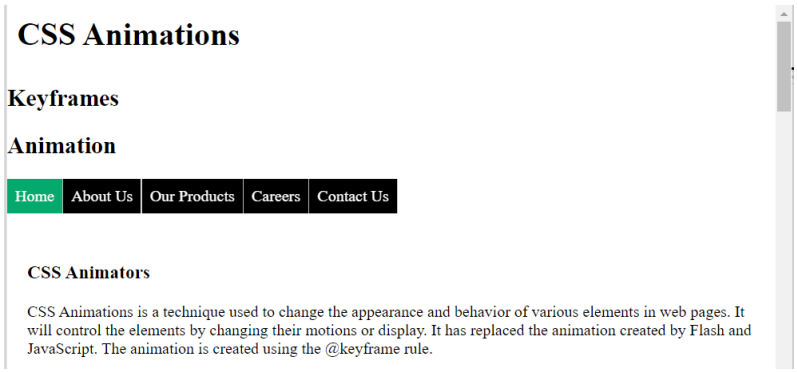
HTML

```html
<body>
  <h1 style="padding:10px">CSS Animations</h1>
  <h2>Keyframes</h2>
  <h2>Animation</h2>
    <ul>
        <li><a class="active" href="#home">Home</a></li>
        <li><a href="#about">About Us</a></li>
        <li><a href="#products">Our Products</a></li>
        <li><a href="#careers">Careers</a></li>
        <li><a href="#conatct">Contact Us</a></li>
    </ul>
   <div style="padding:20px;margin-top:10px;height:1500px;">
    <h3>CSS Animators</h3>
<p>CSS Animations is a technique used to change the appearance
and behavior of various elements in web pages.
It will control the elements by changing their motions or display.
It has replaced the animation created by Flash and JavaScript.
The animation is created using the @keyframe rule.</p>

<p>It has two parts,
CSS Properties (describe the animation of the elements)
keyframes (specific time intervals at which the animations have to occur)
When the animation is created in the @keyframe rule, it must have
a selector otherwise, the animation will have no effect.</p>

<p>@keyframe
Keyframes are the foundation of CSS Animations.
It will control the intermediate steps in a CSS animation sequence.
It defines the display of animation at the corresponding
stages in the whole duration.</p>
</div>
</body>
```

Style

```css
<style>
  body {margin:0;}
  ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    top: 0;
    position: sticky;
    width: 100%;
  }

  li{
    float: left;
    border-right:1px solid #bbb;
  }
  li a {
    display: block;
    color: white;
    text-align: center;
    text-decoration: none;
    padding: 8px;
    background-color: black;
  }
  li a.active {
    background-color: #04AA6D;
    color: white;
  }

  li a:hover:not(.active) {
    background-color: #555;
    color: white;
  }
  </style>
```

Output



- To create a horizontal navigation bar there are two ways.
- They are
    - Inline List Items
    - Float List Items

Inline List Items

- Horizontal Navigation Bar is created by specifying <li> element as **inline**.

```
li{
  display: inline;
}
```

- The line breaks before and after each item in the list is removed to display in one line.
- elements are displayed as **block** by default.

Float List Items

- Horizontal Navigation Bar is created by specifying <li> element as ***float: left;***.
- Setting ***float: left;*** will make the block elements float next to each other.
- ***display: block;*** - We display the links as block elements which allows us to specify the width, padding, margin, height, etc.

```
li{
  float: left;
}

li a {
  display: block;
  padding: 8px;
  background-color: black;
}
```

- Set the navigation bar **background color** to **black** and when the user moves the cursor over the menu the background color of the link has to change to **gray**.

```
li a {
  display: block;
  color: white;
  text-align: center;
  text-decoration: none;
  padding: 8px;
  background-color: black;
}
li a:hover {
  background-color: #555;
  color: white;
}
```

- **Active/Current Link Navigation**
  - We have added an "**active**" class to the current link so that the user will get to know the current web page he is viewing.
  - The current Link will appear in **dark green** color.

```
li a.active {
  background-color: #04AA6D;
  color: white;
}
```

- Adding ***border: right;*** property to divide the links

```
li{
  float: left;
  border-right:1px solid #bbb;
}
```

- We can make the Navigation Bar **stay** in one position either top or bottom even when **scrolled** by setting ***position: fixed;***

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  top: 0;
  position: fixed;
  width: 100%;
}
```

- Fixed position might **not work properly** on **mobile devices**.
- **Sticky Navbar**
  - Sticky elements **switch** between **relative** and **fixed** position depending on the position of the **scroll**.

- It behaves **relative** until it reaches a specific position and later behaves **fixed**.

```css
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  top: 0;
  position: -webkit-sticky; /* Safari */
  position: sticky;
  width: 100%;
}
```

- Sticky Position is **not supported by IE**. For **Safari,** it requires a **-webkit- prefix**.
- For Sticky position to work we have to specify **top, right, bottom or left**.

Responsive Web Design

- Web Pages can be viewed on **any device** like mobile, laptop, tablet, etc. Regardless of the device, our web page should **look good** and **easy to use** for this we go for **Responsive Web Design**.
- Responsive Web Design uses only **HTML** and **CSS** to create a web page that looks good on all devices.
- While viewing on **smaller devices**, the web page should not leave any information. Instead, the content should **fit** any device.
- To **resize, hide, shrink, enlarge or move** the content using **HTML and CSS** to look good on any screen is called **Responsive Web Design**.

Viewport

- The **Visible area** of a web page is called **Viewport**.
- The viewport differs from device to device. **Mobile phones** have **smaller** viewports when compared to computers.
- Initially, Web pages are designed **only** for **computers** by having **static design** and **fixed size**.
- **Fixed-size** web pages are **too large** for the viewport of **mobiles and tablets**.
- **<meta>** tag in **HTML5** helped web designers by taking control over the **viewport**.

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

  - **width=device-width** - Sets the web page width to screen-width of the device.
  - **initial-scale=1.0** - Sets the initial zoom level of the page when loaded by the browser for the first time.
- Elements with **large fixed widths** should not be used.
- The content **should not** rely on particular **viewport width** to render well.

- Page elements **should not** have **large absolute CSS width**. Instead, use **relative** width values such as **width: 100%. Large absolute values** cause the element to fall **outside** the viewport.

Ingredients of RWD

- Media Queries
- Fluid Grids
- Flexible Visuals

Media Queries

- Media Query is a technique in **CSS3**
- If a condition is true, the **@media** rule is used to include a block of CSS properties.
- **Media queries** are used to make images smaller on all mobiles but larger on other devices like ipads, desktops, etc.

```
body {
  background-color: lightgreen;
}

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

- If the browser window is within 600px the background color is light blue or else the background color will be light green.

Flexible Grid / Fluid Grid

- The **rearrangement of columns** themselves to fit the screen size is possible only by **Flexible Grid**. These Flexible Grids are created using CSS.

Flexible Visuals

- Images will scale up and down if **width: 100%;** and **height: auto;** is set.

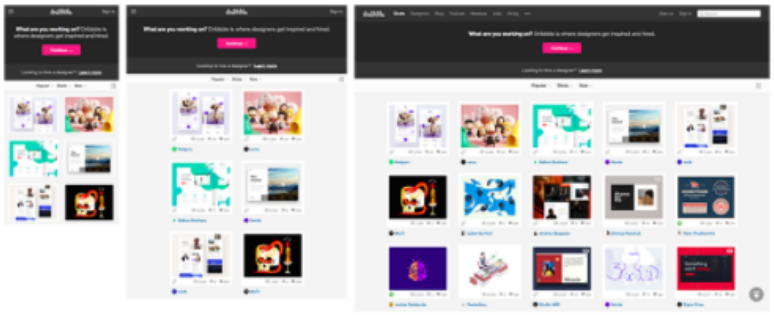
```
img {
  width: 100%;
  height: auto;
}
```

Real Time Examples of RWD

- **DropBox**

  - DropBox uses **fluid grid** and **flexible visuals** to achieve this responsive website.

- **Dribbble**



  - Dribbble website uses a **flexible grid** that condenses from 5 columns on computers to 2 columns on mobiles and tablets.

Nav Bar - Practice Code

Steps to see the output:

- Open VS code.
- Select New File in the opened folder…
- Save the file in the appropriate folder/ create the folder and save the file with extension.
- Copy and Add the below given html and css code in the file.
- View the file in the browser, by right click on the file name in the left pane.
- View of file in browser.

Problem Statement 1: Vertical Navigation Bar

Make a vertical navigation bar for a page which consists of at least 4 links.

Output:-

Problem Statement 2: Horizontal Navigation Bar

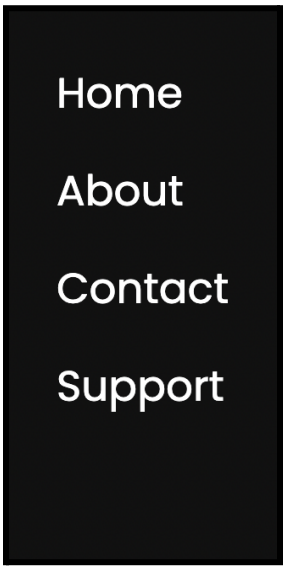
Make a horizontal navigation bar for a page which consists of at least 4 links and has a bottom border.

Output:-

Problem Statement 3: Fixed Side - Nav Bar

Make a vertical fixed navigation bar for a page which is responsive.

Output:



Solutions

Problem Statement 1:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>JS Navbar Practice Code </title>
</head>
<style>
   .verticalnavbar {
       width: 150px;
     }
     .verticalnavbar a {
       background-color: rgb(217, 213, 213);
       color: black;
       display: block;
       padding: 10px;
       text-decoration: none;
     }

     .verticalnavbar a:hover {
       background-color: rgb(48, 212, 210);
     }

     .verticalnavbar a.active {
       background-color: #0491aa;
       color: white;
     }
</style>
<body>
   <div class="verticalnavbar">
       <a
href="https://testbook.com/aeje-drdo-ssc-imd-dhamaka-pack-coaching"
class="active">Home</a>
       <a href="#">About</a>
       <a href="#">Contact</a>
       <a href="#">Support</a>
     </div>
</body>
</html>
```

Problem Statement 2:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>JS Nav Bar Practice Code </title>
</head>
<style>
   .horizontalnavbar {
       background-color: rgb(0, 0, 0);
       overflow: hidden;
   }
   .horizontalnavbar a {
       float: left;
       display: block;
       color: #f2f2f2;
       text-align: center;
       padding: 20px;
       text-decoration: none;
       font-size: 18px;
       font-family:'Poppins';
       border-bottom: 5px solid transparent;
   }

   .horizontalnavbar a:hover {
       border-bottom: 5px solid rgb(13, 209, 220);
   }

   .horizontalnavbar a.active {
       border-bottom: 5px solid rgb(13, 209, 220);
   }
</style>
<body>
   <div class="horizontalnavbar">
       <a
href="https://testbook.com/aeje-drdo-ssc-imd-dhamaka-pack-coaching"
class="active">Home</a>
       <a href="#">About</a>
       <a href="#">Contact</a>
       <a href="#">Support</a>
     </div>
</body>
</html>
```

Problem Statement 3:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>JS Nav Bar Practice Code </title>
</head>
<style>
    .fixednav {
        height: 100%;
        width: 120px;
        position: fixed;
        z-index: 1;
        top: 0;
        left: 0;
        background-color: #111;
        overflow-x: hidden;
        padding: 20px;
      }

      .fixednav a {
        padding: 10px;
        text-decoration: none;
        font-family:'Poppins';
        font-size: 25px;
        color: #ffffff;
        display: block;
      }

      .fixednav a:hover {
        color: #18d1b8;
      }

      .main {
        margin-left: 160px;
        padding: 0px 10px;
      }

      @media screen and (max-height: 450px) {
        .fixednav {padding-top: 15px;}
        .fixednav a {font-size: 18px;}
      }

</style>
<body>
    <div class="fixednav">
        <a
href="https://testbook.com/aeje-drdo-ssc-imd-dhamaka-pack-coaching"
class="active">Home</a>
        <a href="#">About</a>
        <a href="#">Contact</a>
        <a href="#">Support</a>
    </div>
</body>
</html>
```

The complete CSS Guide

Topics Covered:

- What is CSS?
- CSS Font properties.
- CSS Text properties.
- CSS Background properties.
- CSS Border properties.
- Box Model properties.
    - Margin properties.
    - Padding properties.
- CSS Table properties.
- CSS Positioning properties.
- CSS Transition properties.

Topics in Detail:

CSS:

- **CSS stands for** Cascading Style Sheets.
- The documents written in HTML are formatted and presented using CSS.
- There are many features in HTML and CSS is the most popular addition to it. Rather than changing the web page itself, with CSS, only the styles need to be changed, which means fewer chances to the overall code.
- Further, HTML contains a lot of repeated code for each web page, which can be put in a common file with CSS and used across web pages.
- **CSS contains** – selector and declaration block. A declaration block consists of property-value pairs.

Font

Property	Values	Example
font-style	normal/italic/inherit/oblique	font-style: normal
font-variant	normal/inherit/small-caps	font-variant: small-caps
font-weight	normal/bold/bolder/lighter/100-900/inherit	font-weight:bold

font-size	?px/?%/small/medium/large	font-size: large font-size :5
font-family	verdana/calibri.. etc…	font-family: verdana

Text

Property	**Values**	**Example**
text-align	left/right/center/justify	text-align: justify;
letter spacing	normal/length/?%	letter spacing : 3%;
Text-outline	None/length/color	Text-outline: red
word-wrap	normal/length	word-wrap: normal;
direction	ltr/rtl/inherit	Direction: ltr;
text-wrap	normal/unrestricted/none	text-wrap: normal
text-indent	?%/?px	text-indent: 2%
word-spacing	normal/?%/?px	word-spacing: normal
text-transform	none/uppercase/lowercase/capitalize	text-transform: lowercase
text-emphasis	none/dot/open/filled/circle/triangle	text-emphasis: filled
text-justify	auto/distribute/inter-word	text-justify:distribute

Backgrounds

Property	Values	Example
background-size	auto/cover/?px/?%	background-size: cover
background-image	url/none	background-image: none
background-repeat	no-repeat/repeat-x/repeat-y/repeat	background-repeat: repeat
background-attachment	fixed/scroll	background-attachment: fixed
background-color	color/transparent	background-color: white
background-position	can be any position from different combinations like top left, top right, top center. Same with the bottom. can be mentioned in terms of position x-% and y-%	background-position: top-left;
background-origin	the starting point of the background	background-origin: 0;
background-clip - lets you control how much of the background image should extend beyond the element's content or padding	content-box/padding-box/border-box/ no-clip/?%/?px	background-clip: no-clip

Borders

Property	Values	Example
border-width	thin/thick/medium/?px	border-width: medium border-width: 20px
border-style	none/dashed/dotted/inset/double/solid	border-style : dotted
border-color	name of the color	border-color: black
border-left: border-left-color border-left-width		border-left-color: black border-left-width : 10px
border-right: border-right-color border-right-width		border-right-color : black border-right-width : 15px
border-top: border-top-width border-top-color		border-top-width : 10px border-top-color : blue
border-bottom: border-bottom-color border-bottom-width		border-bottom-color : black border-bottom-width : 15px
border-decoration-break	maintain consistent design amongst fragments of broken element slice/clone	border-decoration-break: slice;

border-radius border-top-right-radius border-bottom-right-radius border-bottom-left-radius border-top-left-radius	?px	border-top-left-radius : 20px
border-image	?%/stretch/repeat/round/none	border-image : repeat border-image : 12px

Box Model

Property	**Values**	**Example**
float	left | right | none	float : right
height	auto length %	height : 10px
max-height	none length %	max-height : 10px
max-width	none length %	max-width : 120%
min-height	none length %	min-height : 50%
min-width	auto % length	min-width : 30px

Margin

Property	Values	Example
margin-bottom	auto/length %	margin-bottom : 20px
margin-left	auto/height %	margin-left : auto
margin-right	auto/height %	margin-right : 30%
margin-top	auto/length %	margin-top : 40mm

Padding

Property	Values	Example
padding-bottom	length %	padding-bottom : 20px
padding-top	length %	padding-top : 20px
padding-right	length %	padding-right : 20px
padding-left	length %	padding-left : 20px
display	none/inline/block/inline-block/list-item/run-in/compact/table/inline-table/table-row-group/table-headergroup/table-footer-group/table-row/table-column-group/table-column/table-cell/table-caption/ruby/ruby-base/ruby-text/ruby-base-group/ruby-text-group	display : inline

marquee-direction	forward/reverse	marquee-direction : forward
marquee-loop	infinite/integer	marquee-loop : 10
marquee-play-count	infinite/integer	marquee-play-count : 50
marquee-speed	slow/normal/fast	marquee-speed : slow
marquee-style	scroll/slide/alternate	marquee-style : scroll
overflow	visible/hidden/scroll/auto/no-display/no-content/overflow-x/overflow-y	overflow : visible
overflow-style	auto/marquee-line/marquee-block	overflow-style : auto
overflow-x	visible/hidden/scroll/auto/no-display/no-content	overflow-x : scroll
rotation	angle	rotation : 20deg
rotation-point	position (paired value off-set)	rotation-point : 50% 50%
visibility	visible/hidden/collapse	visibility: hidden
clear	left/right/both/none	clear: left

Table

Property	Possible values
border-collapse	collapse/separate
empty-cells	show/hide
border-spacing	?%/?px
table-layout	auto/fixed
caption-side	top/bottom/left/right

Positioning

Property	Values	Example
bottom/right/top/left	auto/%/length	bottom: 20% top : auto left : 40px right : 25px
z-index	auto/number	z-index : 2
clip	shape/auto	clip : auto
position	fixed/static/relative/absolute	position : static

Transitions

Property	Values	Example
transitions-delay	time (ms/s)	transitions-delay : 20ms
transitions-duration	time (ms/s)	transitions-duration : 2s
transitions-property	none/all	transitions-property: none
transition-timing-function	ease/linear/ease-in/ease-out/ease-in-out/cubicBezier(number, number, number, number)	transition-timing-function: ease-in-out;

skill academy
by testbook

Minor Assignment

Calendar and Text animation

Problem Statement 1: Calendar

With your newly gained knowledge in CSS Grid properties create a simple static calendar using HTML list and CSS: grid, display, background, and box model properties.

Sample Output:

Grading Parameters:

Parameters	Rubics
HTML Elements	10 - Use of headings, lists with div sections and class names to create a calendar. 5 - Use of headings, lists with div sections without class names 0 - No use of headings, lists with div sections in HTML.
Box Model	10 - Use of width, padding, and margin properties. 5 - If any two box model properties are used. 0 - If no box model properties are used.
Grid properties	10 - Use of Display: grid, grid-template-columns, and grid-column-start properties. 5 - Use of any two grid properties. 0 - If no grid properties are used.

Problem Statement 2: Text animation

With your newly gained knowledge in CSS combinators and animations create a simple text animation web page with help of text, fonts, box model, animation, combinators, and keyframes.

Sample Output:

The highlighted text keeps on changes

Grading Parameters:

Parameters	Rubics
HTML Elements	10 - Use of div elements with id's and required text. 0 - If no div elements are used.
Box Model	10 - Use of width, padding, and margin properties. 5 - If any two box model properties are used. 0 - If no box model properties are used.
Combinators	10 - Use of combinators to apply styles 0 - Not used combinators to apply styles
Animation	10 - Use of animation properties and keyframes to change the text as expected. 5 - Used animation properties and keyframes but did not work as expected. 0 - No use of animations.

Calendar and Text animation

Problem Statement 1: Calendar

With your newly gained knowledge in CSS Grid properties create a simple static calendar using HTML list and CSS: grid, display, background, and box model properties.

Sample Output:

Solution:

HTML:

```
<div class="calendar-wrapper">
  <h1>December 2020</h1>
  <ol class="calendar">

    <li class="day-name">Sun</li>
    <li class="day-name">Mon</li>
    <li class="day-name">Tue</li>
    <li class="day-name">Wed</li>
    <li class="day-name">Thu</li>
    <li class="day-name">Fri</li>
    <li class="day-name">Sat</li>

    <li class="first-day">1</li>

    <li>2</li>
    <li>3</li>
    <li>4</li>
    <li>5</li>
    <li>6</li>
```

```
    <li>7</li>
    <li>8</li>
    <li>9</li>
    <li>10</li>
    <li>11</li>
    <li>12</li>
    <li>13</li>
    <li>14</li>
    <li>15</li>
    <li>16</li>
    <li>17</li>
    <li>18</li>
    <li>19</li>
    <li>20</li>
    <li>21</li>
    <li>22</li>
    <li>23</li>
    <li>24</li>
    <li>25</li>
    <li>26</li>
    <li>27</li>
    <li>28</li>
    <li>29</li>
    <li>30</li>
    <li>31</li>
  </ol>
</div>
```

CSS:

```
.calendar-wrapper {
  max-width: 280px;
  font: 100% system-ui;
}
.calendar {
  display: grid;
  grid-template-columns: repeat(7, 1fr);
}

.first-day {
  grid-column-start: 3;
}


.day-name {
  background: #eee;
}

h1 {
  text-align: center;
}
ol {
  list-style: none;
  margin: 0;
  padding: 0;
```

```
  text-align: center;
}
li {
  padding: 2px;
}
```

Problem Statement 2: Text animation

With your newly gained knowledge in CSS combinators and animations create a simple text animation web page with help of text, fonts, box model, animation, combinators, and keyframes.

Sample Output:

The highlighted text keeps on changes

Solution:

HTML:

```
<div id=container>
  Make
  <div id=flip>
    <div><div>wOrK</div></div>
    <div><div>lifeStyle</div></div>
    <div><div>Everything</div></div>
  </div>
  AweSoMe!
</div>

<p>a css3 animation demo</p>
```

CSS:

```
body {
  margin:0px;
  text-align:center;
}

#container {
  color:#999;
  text-transform: uppercase;
  font-size:36px;
  font-weight:bold;
```

```
  padding-top:200px;
  position:fixed;
  width:100%;
  bottom:45%;
  display:block;
}

#flip {
  height:50px;
  overflow:hidden;
}

#flip > div > div {
  color:#fff;
  padding:4px 12px;
  height:45px;
  margin-bottom:45px;
  display:inline-block;
}

#flip div:first-child {
  animation: show 5s linear infinite;
}

#flip div div {
  background:#42c58a;
}
#flip div:first-child div {
  background:#4ec7f3;
}
#flip div:last-child div {
  background:#DC143C;
}

@keyframes show {
  0% {margin-top:-270px;}
  5% {margin-top:-180px;}
  33% {margin-top:-180px;}
  38% {margin-top:-90px;}
  66% {margin-top:-90px;}
  71% {margin-top:0px;}
  99.99% {margin-top:0px;}
  100% {margin-top:-270px;}
}

p {
  position:fixed;
  width:100%;
  bottom:30px;
  font-size:12px;
  color:#999;
  margin-top:200px;
}
```