

### **Experiment No. 03**

**Title:** Create flexible content layout using CSS3.

**Batch:A2      Roll No.:16010421063**

**Experiment No.: 3**

**Aim:** To apply basic CSS and flexible content layout using CSS3 to your website..

---

**Resources needed:** need to fill by students

---

### Theory:

**Cascading Style Sheets (CSS)** is a style sheet language used for describing the presentation of a document written in a markup language. HTML was NEVER intended to contain tags for formatting a web page. HTML was created to **describe the content** of a web page. CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

CSS has various levels and profiles like CSS1, CSS2, CSS3, CSS4. There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

### External Style Sheet

Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section of html page. An external style sheet is ideal when the style is applied to many pages.

### Internal Style Sheet

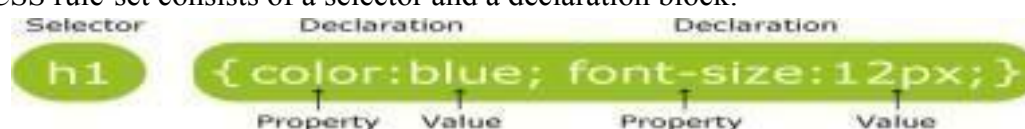
An internal style sheet may be used if one single page has a unique style. Internal styles are defined within the <style> element, inside the <head> section of an HTML page

### Inline Styles

An inline style may be used to apply a unique style for a single element. To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

### CSS Rule-Set

A CSS rule-set consists of a selector and a declaration block:



Eg:-

```
p {  
  text-align:  
  center; color:  
  red;  
}
```

Here p stands for paragraph element.

### **CSS Colors and Backgrounds**

Colors can be applied by using colors and background-color. Colors are displayed combining RED, GREEN, and BLUE light. The CSS background properties are used to define the background effects for elements.

### **CSS Margin and Padding Properties.**

The CSS margin properties are used to generate space around elements. Margin is the space outside something whereas padding is the space inside something. The margin properties set the size of the white space OUTSIDE the border. The CSS padding properties are used to generate space around content. The padding properties set the size of the white space between the element content and the element border.

### **CSS Dimension Properties**

The CSS dimension properties allow you to control the height and width of an element.






### **Navigation Bars**

Having easy-to-use navigation is important for any web site. With CSS you can transform boring HTML menus into good-looking navigation bars.

### **CSS Flexbox Layout**

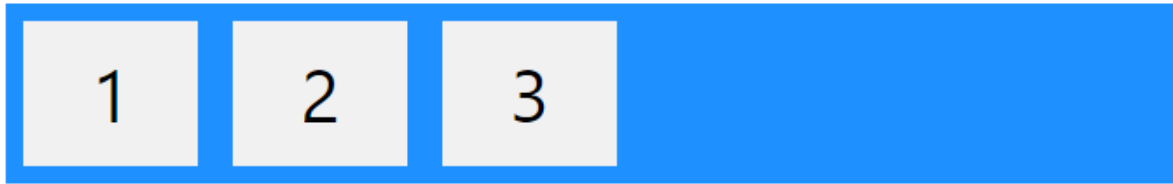
The Flexible Box Layout Module makes it easier to design flexible responsive layout structure without using float or positioning.

The flexbox properties are supported in all modern browsers.

				
29.0	11.0	22.0	10	48

# Flexbox Elements

To start using the Flexbox model, you need to first define a flex container.



The element above represents a flex container (the blue area) with three flex items.

Example

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

```
.flex-container > div {
  background-color:
  #f1f1f1; margin: 10px;
  padding: 20px;
  font-size:
  30px;
}
</style>
</head>
<body>
```

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

<p>A Flexible Layout must have a parent element with the <em>display</em> property set to <em>flex</em>.</p>

<p>Direct child elements(s) of the flexible container automatically becomes flexible items.</p>

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

Output



A Flexible Layout must have a parent element with the *display* property set to *flex*.

Direct child element(s) of the flexible container automatically becomes flexible items.

## The CSS Flexbox Container Properties

The following table lists all the CSS Flexbox Container properties:

Property	Description
<a href="#"><u>align-content</u></a>	Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
<a href="#"><u>align-items</u></a>	Vertically aligns the flex items when the items do not use all available space on the cross-axis
<a href="#"><u>display</u></a>	Specifies the type of box used for an HTML element
<a href="#"><u>flex-direction</u></a>	Specifies the direction of the flexible items inside a flex container
<a href="#"><u>flex-flow</u></a>	A shorthand property for flex-direction and flex-wrap
<a href="#"><u>flex-wrap</u></a>	Specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line
<a href="#"><u>justify-content</u></a>	Horizontally aligns the flex items when the items do not use all available space on the main-axis

## The CSS Flexbox Items Properties

The following table lists all the CSS Flexbox Items properties:

Property	Description
<a href="#"><u>align-self</u></a>	Specifies the alignment for a flex item (overrides the flex container's align-items property)
<a href="#"><u>flex</u></a>	A shorthand property for the flex-grow, flex-shrink, and the flex-basis properties
<a href="#"><u>flex-basis</u></a>	Specifies the initial length of a flex item
<a href="#"><u>flex-grow</u></a>	Specifies how much a flex item will grow relative to the rest of the flex items inside the same container
<a href="#"><u>flex-shrink</u></a>	Specifies how much a flex item will shrink relative to the rest of the flex items inside the same container
<a href="#"><u>order</u></a>	Specifies the order of the flex items inside the same container

### Responsive Flexbox

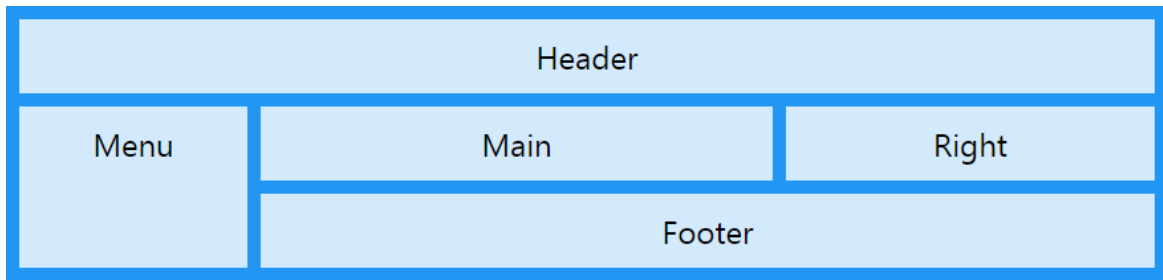
Laptop and Desktops:			Mobile phones and Tablets:		
1	2	3	1	2	3

For example, if you want to create a two-column layout for most screen sizes, and a one-column layout for small screen sizes (such as phones and tablets), you can change the flex-direction from row to column at a specific breakpoint (800px in the example below).

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

/* Responsive layout - makes a one column layout instead of a two-column layout */
@media (max-width: 800px) {
  .flex-container {
    flex-direction: column;
  }
}
```

### CSS Grid Layout Module



The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

## All CSS Grid Properties

Property	Description
<a href="#"><u>column-gap</u></a>	Specifies the gap between the columns
<a href="#"><u>gap</u></a>	A shorthand property for the <i>row-gap</i> and the <i>column-gap</i> properties
<a href="#"><u>grid</u></a>	A shorthand property for the <i>grid-template-rows</i> , <i>grid-template-columns</i> , <i>grid-template-areas</i> , <i>grid-auto-rows</i> , <i>grid-auto-columns</i> , and the <i>grid-auto-flow</i> properties
<a href="#"><u>grid-area</u></a>	Either specifies a name for the grid item, or this property is a shorthand property for the <i>grid-row-start</i> , <i>grid-column-start</i> , <i>grid-row-end</i> , and <i>grid-column-end</i> properties
<a href="#"><u>grid-auto-columns</u></a>	Specifies a default column size
<a href="#"><u>grid-auto-flow</u></a>	Specifies how auto-placed items are inserted in the grid
<a href="#"><u>grid-auto-rows</u></a>	Specifies a default row size
<a href="#"><u>grid-column</u></a>	A shorthand property for the <i>grid-column-start</i> and the <i>grid-column-end</i> properties

<a href="#"><u>grid-column-gap</u></a>	Specifies the size of the gap between columns
<a href="#"><u>grid-column-start</u></a>	Specifies where to start the grid item
<a href="#"><u>grid-gap</u></a>	A shorthand property for the <i>grid-row-gap</i> and <i>grid-column-gap</i> properties
<a href="#"><u>grid-row</u></a>	A shorthand property for the <i>grid-row-start</i> and the <i>grid-row-end</i> properties
<a href="#"><u>grid-row-end</u></a>	Specifies where to end the grid item
<a href="#"><u>grid-row-gap</u></a>	Specifies the size of the gap between rows
<a href="#"><u>grid-row-start</u></a>	Specifies where to start the grid item
<a href="#"><u>grid-template</u></a>	A shorthand property for the <i>grid-template-rows</i> , <i>grid-template-columns</i> and <i>grid-areas</i> properties
<a href="#"><u>grid-template-areas</u></a>	Specifies how to display columns and rows, using named grid items
<a href="#"><u>grid-template-columns</u></a>	Specifies the size of the columns, and how many columns in a grid layout
<a href="#"><u>grid-template-rows</u></a>	Specifies the size of the rows in a grid layout
<a href="#"><u>row-gap</u></a>	Specifies the gap between the grid rows

## Grid Container

To make an HTML element behave as a grid container, you have to set the display property to grid or inline-grid.

Grid containers consist of grid items, placed inside columns and rows.

### A grid container contains grid items.

By default, a container has one grid item for each column, in each row, but you can style the grid items so that they will span multiple columns and/or rows.

---

### Activity:

Design a website pages using CSS properties like Colors and Background-color,Lengths and percentages,Margin and padding,Borders,Navigation bars, flexbox layot, flexbox container and item properties, responsive flexbox ,CSS Grid ,Grid Container ,A grid container contains grid items.

---

**Results: Display the designed web pages along with the**

**code**

**style.css**

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



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

body {
    font-family: Arial, sans-serif;
    font-size: 16px;
    line-height: 1.5;
    color: #333;
    background-color: #f8f8f8;
    background-image: linear-gradient(
        45deg,
        rgba(255, 255, 255, 0.5) 25%,
        transparent 25%,
        transparent 50%,
        rgba(255, 255, 255, 0.5) 50%,
        rgba(255, 255, 255, 0.5) 75%,
        transparent 75%,
        transparent
    );
    background-size: 40px 40px;
    backdrop-filter: blur(5px);
}

a {
    color: #007bff;
    text-decoration: none;
}

ul {
    list-style: none;
}

header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    background-color: rgba(255, 255, 255, 0.3);
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
}

nav ul {
    display: flex;
}

nav li {
```

```
margin-left: 20px;
}

nav a {
  transition: all 0.3s ease;
}

nav a:hover {
  color: #555;
}

h1 {
  font-size: 36px;
}

main {
  display: flex;
  flex-wrap: wrap;
  padding: 40px;
  justify-content: space-between;
}

section {
  width: 70%;
}

.post-list {
  display: flex;
  flex-wrap: wrap;
  margin-top: 20px;
}

article {
  flex-basis: calc(33.33% - 20px);
  background-color: rgba(255, 255, 255, 0.3);
  padding: 20px;
  margin-bottom: 20px;
  border-radius: 10px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
}

article h3 {
  font-size: 24px;
  margin-bottom: 10px;
}

article p {
```

```
margin-bottom: 20px;
}

.read-more {
  display: inline-block;
  background-color: rgba(0, 123, 255, 0.8);
  color: #fff;
  padding: 10px 20px;
  border-radius: 5px;
  transition: all 0.3s ease;
}

.read-more:hover {
  background-color: rgba(0, 123, 255, 1);
}

aside {
  width: 25%;
  background-color: rgba(255, 255, 255, 0.3);
  padding: 20px;
  margin-left: 20px;
  border-radius: 10px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
}

aside h2 {
  font-size: 24px;
  margin-bottom: 10px;
}

aside p {
  margin-bottom: 20px;
}

footer {
  background-color: rgba(255, 255, 255, 0.3);
  color: #fff;
  padding: 20px;
  text-align: center;
  margin-top: 40px;
  border-radius: 10px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
}

.glass {
  background-color: rgba(255, 255, 255, 0.1);
  backdrop-filter: blur(5px);
}
```

```
    box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37);
    backdrop-filter: blur(5.5px);
    border-radius: 10px;
    border: 1px solid rgba(255, 255, 255, 0.18);
}

.glass::before {
    content: "";
    position: absolute;
    z-index: -1;
    top: -5px;
    left: -5px;
    right: -5px;
    bottom: -5px;
    background-color: rgba(255, 255, 255, 0.1);
    border-radius: 15px;
    backdrop-filter: blur(15px);
    box-shadow: 0 0 20px rgba(255, 255, 255, 0.5);
}

.glass::after {
    content: "";
    position: absolute;
    z-index: -1;
    top: -5px;
    left: -5px;
    right: -5px;
    bottom: -5px;
    background-color: rgba(255, 255, 255, 0.1);
    border-radius: 15px;
    backdrop-filter: blur(15px);
    box-shadow: 0 0 20px rgba(255, 255, 255, 0.5);
}

@media only screen and (max-width: 1024px) {
    main {
        flex-direction: column;
    }

    section {
        width: 100%;
    }

    aside {
        width: 100%;
        margin-left: 0;
        margin-top: 20px;
    }
}
```

```
}

article {
    flex-basis: 100%;
}
}

@media only screen and (max-width: 768px) {
    header {
        flex-direction: column;
        text-align: center;
    }

    nav {
        margin-top: 20px;
    }

    nav li {
        margin: 0 10px;
    }

    aside {
        padding: 10px;
    }
}

@media only screen and (max-width: 480px) {
    h1 {
        font-size: 28px;
    }

    article h3 {
        font-size: 20px;
    }

    aside h2 {
        font-size: 20px;
    }
}
```

index.html

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

```
<link rel="stylesheet" href="style.css" />
</head>
<body>
  <header>
    <nav>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Blog</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>
    <h1>My Blog</h1>
  </header>

  <main>
    <section>
      <h2>Recent Posts</h2>
      <div class="post-list">
        <article>
          <h3>Post Title</h3>
          <p>
            Lorem ipsum dolor sit amet, consectetur
            adipiscing elit. Nunc
            aliquam turpis eget eros malesuada, id
            consectetur quam eleifend.
            Mauris fermentum velit eu enim scelerisque, vel
            rhoncus ante
            bibendum.
          </p>
          <a href="#" class="read-more">Read More</a>
        </article>
        <article>
          <h3>Post Title</h3>
          <p>
            Lorem ipsum dolor sit amet, consectetur
            adipiscing elit. Nunc
            aliquam turpis eget eros malesuada, id
            consectetur quam eleifend.
            Mauris fermentum velit eu enim scelerisque, vel
            rhoncus ante
            bibendum.
          </p>
          <a href="#" class="read-more">Read More</a>
        </article>
        <article>
          <h3>Post Title</h3>
```

```
<p>
    Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Nunc
    aliquam turpis eget eros malesuada, id
consectetur quam eleifend.
    Mauris fermentum velit eu enim scelerisque, vel
rhoncus ante
    bibendum.
</p>
<a href="#" class="read-more">Read More</a>
</article>
</div>
</section>

<aside>
    <h2>About Me</h2>
    <p>
        Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Nunc aliquam
        turpis eget eros malesuada, id consectetur quam eleifend.
Mauris
        fermentum velit eu enim scelerisque, vel rhoncus ante
bibendum.
    </p>
</aside>
</main>

<footer>
    <p>&copy; 2023 My Blog. All Rights Reserved.</p>
</footer>
</body>
</html>
```

**Outcomes: CO 2 Create Web pages using HTML 5 and CSS.**

---

### Questions:

1. How many axis does a flex-box layout contain?

A flexbox layout contains two axis: the main axis and the cross axis. The main axis is the primary axis along which flex items are arranged, and it is defined by the flex-direction property. The cross axis is perpendicular to the main axis and is used for aligning items within the container. The direction of the cross axis is determined by the flex-direction property as well, but it is the opposite direction of the main axis.

2. What is the difference between auto-fill and auto-fit properties when defining a grid- template.

When defining a grid template, both auto-fill and auto-fit values can be used to define the number of columns or rows. auto-fill specifies that the grid container should create as many rows or columns as possible to fill the available space, but it does not increase the size of the grid items themselves. If there is not enough content to fill all the rows or columns, the empty cells will still be created. auto-fit is similar to auto-fill, but it also allows the grid items to be stretched to fill the available space. If there is not enough content to fill all the rows or columns, the unused cells will collapse and the grid items will be stretched to fill the remaining space. In summary, auto-fill creates as many grid tracks as possible without stretching the grid items, while auto-fit creates as many tracks as possible and stretches the grid items to fill the available space.

3. What is the meaning of the fr unit in the grid layout?

The fr unit in CSS grid layout stands for "fractional unit". It is used to define the size of grid tracks, which are the rows or columns that form a grid. When using fr, the available space is divided into fractions, where each fraction represents one unit of fr. For example, if a grid container has two columns and the first column is set to 1fr and the second column is set to 2fr, the second column will be twice as wide as the first column. fr units are useful because they allow the grid to be flexible and responsive. If the size of the container changes, the size of the grid tracks will adjust accordingly. They can also be used in combination with other units, such as pixels or percentages, to create more complex layouts. For example, the following code defines a grid with three columns, where the first and third columns have a fixed width of 100 pixels, and the second column takes up the remaining space using the fr unit:

---

### Conclusion: (Conclusion to be based on the outcomes achieved)

This experiment has demonstrated various CSS techniques and properties such as colors, layout, flexbox, CSS grid, and glassmorphism design, which can be used to create visually appealing and responsive web pages



Signature of faculty in-charge with date

---

**References:**

**Books/ Journals/ Websites:**

- "HTML5: Black Book", Dreamtech Publication.
- "Web Technologies: Black Book", Dreamtech Publication.
- <http://www.w3schools.com>