## What is CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

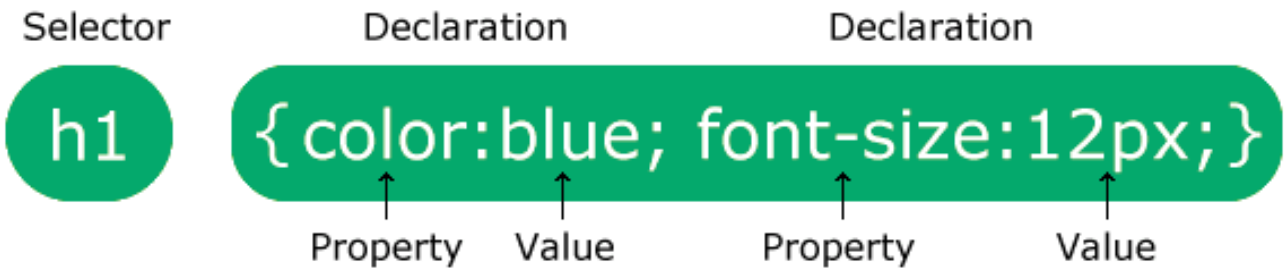
## Why Use CSS?

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
</style>
</head>
<body>
```

```
<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

CSS Syntax




## The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

```
<!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>

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

<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>

</body>
</html>

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

```
}

p.large {
  font-size: 300%;
}
</style>
</head>
<body>

<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>
<p class="center large">This paragraph will be red, center-aligned, and in a large font-size.</p>

</body>
</html>
```

## The CSS Universal Selector

The universal selector (*) selects all HTML elements on the page.

```
<!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>
```

## The CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
  color: red;
}

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

## Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS

- Inline CSS

## External CSS

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

“mystyle.css"

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

h1 {
  color: navy;
  margin-left: 20px;
}
```

**Internal CSS**

The internal style is defined inside the <style> element, inside the head section

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

```
<head>
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

## Inline CSS

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.

```
<!DOCTYPE html>
<html>
<body>


<h1 style="color:blue;text-align:center;">This is a heading</
h1>
<p style="color:red;">This is a paragraph.</p>


</body>
</html>
```

## CSS Comments

A CSS comment is placed inside the `<style>` element, and starts with `/*` and ends with `*/`

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: red;  /* Set text color to red */
}
</style>
</head>
<body>

<p>Hello World!</p>
<p>This paragraph is styled with CSS.</p>
<p>CSS comments are not shown in the output.</p>

</body>
</html>
```

**CSS Color Names**

```
<!DOCTYPE html>
<html>
<body>
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">
```

```
Lorem ipsum dolor sit amet, consectetuer
adipiscing elit, sed diam nonummy nibh euismod
tincidunt ut laoreet dolore magna aliquam erat
volutpat.
Ut wisi enim ad minim veniam, quis nostrud exerci
tation ullamcorper suscipit lobortis nisl ut
aliquip ex ea commodo consequat.
</p>

</body>
</html>
```

**CSS Text Color**

```
<!DOCTYPE html>
<html>
<body>

<h3 style="color:Tomato;">Hello World</h3>

<p style="color:DodgerBlue;">Lorem ipsum dolor sit
amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore
magna aliquam erat volutpat.</p>

<p style="color:MediumSeaGreen;">Ut wisi enim ad
minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut aliquip ex
ea commodo consequat.</p>

</body>
</html>
```

**CSS Border Color**

```
<!DOCTYPE html>
<html>
<body>

<h1 style="border: 2px solid Tomato;">Hello World</h1>

<h1 style="border: 2px solid DodgerBlue;">Hello World</h1>

<h1 style="border: 2px solid Violet;">Hello World</h1>

</body>
</html>
```

**CSS RGB Colors**

RGB Value

In CSS, a color can be specified as an RGB value, using this formula:

***rgb(red, green, blue)***

```
<!DOCTYPE html>
<html>
<body>

<h1>Specify colors using RGB values</h1>
```

```
<h2 style="background-color:rgb(255, 0, 0);">rgb(255, 0, 0)</h2>
<h2 style="background-color:rgb(0, 0, 255);">rgb(0, 0, 255)</h2>
<h2 style="background-color:rgb(60, 179, 113);">rgb(60, 179, 113)</h2>
<h2 style="background-color:rgb(238, 130, 238);">rgb(238, 130, 238)</h2>
<h2 style="background-color:rgb(255, 165, 0);">rgb(255, 165, 0)</h2>
<h2 style="background-color:rgb(106, 90, 205);">rgb(106, 90, 205)</h2>

</body>
</html>
```

## HEX Value

In CSS, a color can be specified using a hexadecimal value in the form:

## ***#rrggbb***

```
<!DOCTYPE html>
<html>
<body>

<h1>Specify colors using HEX values</h1>

<h2 style="background-color:#ff0000;">#ff0000</h2>
<h2 style="background-color:#0000ff;">#0000ff</h2>
<h2 style="background-color:#3cb371;">#3cb371</h2>
```

```
<h2 style="background-color:#ee82ee;">#ee82ee</h2>
<h2 style="background-color:#ffa500;">#ffa500</h2>
<h2 style="background-color:#6a5acd;">#6a5acd</h2>

</body>
</html>
```

## HSL Value

In CSS, a color can be specified using hue, saturation, and lightness (HSL) in the form:

**hsl(*hue, saturation, lightness*)**

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value. 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage. 0% is black, 50% is neither light or dark, 100% is white

```
<!DOCTYPE html>
<html>
<body>

<h1>Specify colors using HSL values</h1>

<h2 style="background-color:hsl(0, 100%, 50%);">hsl(0, 100%, 50%)</h2>
<h2 style="background-color:hsl(240, 100%, 50%);">hsl(240, 100%, 50%)</h2>
<h2 style="background-color:hsl(147, 50%, 47%);">hsl(147, 50%, 47%)</h2>
```

```
<h2 style="background-color:hsl(300, 76%, 72%);">hsl(300, 76%, 72%)</h2>
<h2 style="background-color:hsl(39, 100%, 50%);">hsl(39, 100%, 50%)</h2>
<h2 style="background-color:hsl(248, 53%, 58%);">hsl(248, 53%, 58%)</h2>
</body>
</html>
```

## CSS background-color

The `background-color` property specifies the background color of an element.

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

<h1>Hello World!</h1>

<p>This page has a light blue background color!</p>

</body>
</html>
```

## CSS background-image

The `background-image` property specifies an image to use as the background of an element.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("paper.gif");
}
</style>
</head>
<body>

<h1>Hello World!</h1>

<p>This page has an image as the background!</p>

</body>
</html>
```

## CSS background-repeat

By default, the `background-image` property repeats an image both horizontally and vertically.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
```

```
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<p>Here, a background image is repeated only horizontally!</
p>

</body>
</html>
```

**CSS background-position**

```
<!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>Hello World!</h1>
<p>Here, the background image is only shown once. In
addition it is positioned away from the text.</p>
```

```
<p>In this example we have also added a margin on the right side, so that the background image will not disturb the text.</p>


</body>
</html>
```

## CSS Border Width

The `border-width` property specifies the width of the four borders.

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  border-style: solid;
  border-width: 5px;
}

p.two {
  border-style: solid;
  border-width: medium;
}

p.three {
  border-style: dotted;
  border-width: 2px;
}

p.four {
  border-style: dotted;
```

```
  border-width: thick;
}

p.five {
  border-style: double;
  border-width: 15px;
}

p.six {
  border-style: double;
  border-width: thick;
}
</style>
</head>
<body>

<h2>The border-width Property</h2>
<p>This property specifies the width of the four borders:</p>

<p class="one">Some text.</p>
<p class="two">Some text.</p>
<p class="three">Some text.</p>
<p class="four">Some text.</p>
<p class="five">Some text.</p>
<p class="six">Some text.</p>

<p><b>Note:</b> The "border-width" property does not work if it is used alone.
Always specify the "border-style" property to set the borders first.</p>

</body>
```

</html>

## CSS Border Color

The `border-color` property is used to set the color of the four borders.

The color can be set by:

- name - specify a color name, like "red"
- HEX - specify a HEX value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- transparent

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  border-style: solid;
  border-color: red;
}

p.two {
  border-style: solid;
  border-color: green;
}

p.three {
  border-style: dotted;
  border-color: blue;
}
</style>
</head>
<body>

<h2>The border-color Property</h2>
<p>This property specifies the color of the four borders:</p>
```

```
<p class="one">A solid red border</p>
<p class="two">A solid green border</p>
<p class="three">A dotted blue border</p>

<p><b>Note:</b> The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.</p>

</body>
</html>
```

## Specific Side Colors

The `border-color` property can have from one to four values (for the top border, right border, bottom border, and the left border).

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  border-style: solid;
  border-color: red green blue yellow; /* red top, green right, blue bottom and yellow left */
}
</style>
</head>
<body>


<h2>The border-color Property</h2>
<p>The border-color property can have from one to four values (for the top border, right border, bottom border, and the left border):</p>


<p class="one">A solid multicolor border</p>
```

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

## HEX Values

The color of the border can also be specified using a hexadecimal value (HEX):

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  border-style: solid;
  border-color: #ff0000; /* red */
}

p.two {
  border-style: solid;
  border-color: #0000ff; /* blue */
}

p.three {
  border-style: solid;
  border-color: #bbbbbb; /* grey */
}
</style>
</head>
<body>

<h2>The border-color Property</h2>
```

```
<p>The color of the border can also be specified using a hexadecimal value (HEX):</p>

<p class="one">A solid red border</p>
<p class="two">A solid blue border</p>
<p class="three">A solid grey border</p>

</body>
</html>
```

**RGB Values**

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  border-style: solid;
  border-color: rgb(255, 0, 0); /* red */
}

p.two {
  border-style: solid;
  border-color: rgb(0, 0, 255); /* blue */
}

p.three {
  border-style: solid;
  border-color: rgb(187, 187, 187); /* grey */
}
</style>
</head>
```

```
<body>

<h2>The border-color Property</h2>
<p>The color of the border can also be specified using RGB values:</p>

<p class="one">A solid red border</p>
<p class="two">A solid blue border</p>
<p class="three">A solid grey border</p>

</body>
</html>
```

CSS Border

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
}
</style>
</head>
<body>

<h2>Individual Border Sides</h2>
<p>2 different border styles.</p>
```

</body>
</html>

If the `border-style` property has four values:

- **border-style: dotted solid double dashed;**
  - top border is dotted
  - right border is solid
  - bottom border is double
  - left border is dashed

If the `border-style` property has three values:

- **border-style: dotted solid double;**
  - top border is dotted
  - right and left borders are solid
  - bottom border is double

If the `border-style` property has two values:

- **border-style: dotted solid;**
  - top and bottom borders are dotted
  - right and left borders are solid

If the `border-style` property has one value:

- **border-style: dotted;**
  - all four borders are dotted

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  text-align: center;
}
/* Four values */
p.four {
  border-style: dotted solid double dashed;
}

/* Three values */
p.three {
```

```
  border-style: dotted solid double;
}

/* Two values */
p.two {
  border-style: dotted solid;
}

/* One value */
p.one {
  border-style: dotted;
}
</style>
</head>
<body>

<h2>Individual Border Sides</h2>
<p class="four">4 different border styles.</p>
<p class="three">3 different border styles.</p>
<p class="two">2 different border styles.</p>
<p class="one">1 border style.</p>

</body>
</html>
```

CSS Rounded Borders

```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  border: 2px solid red;
  padding: 5px;
}

p.round1 {
  border: 2px solid red;
  border-radius: 5px;
  padding: 5px;
}
```

```
p.round2 {
  border: 2px solid red;
  border-radius: 8px;
  padding: 5px;
}

p.round3 {
  border: 2px solid red;
  border-radius: 12px;
  padding: 5px;
}
</style>
</head>
<body>

<h2>The border-radius Property</h2>
<p>This property is used to add rounded borders to an element:</p>

<p class="normal">Normal border</p>
<p class="round1">Round border</p>
<p class="round2">Rounder border</p>
<p class="round3">Roundest border</p>

</body>
</html>
```

# CSS Margins

The CSS `margin` properties are used to create space around elements, outside of any defined borders.

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

```
  border: 1px solid black;
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
  background-color: lightblue;
}
</style>
</head>
<body>

<h2>Using individual margin properties</h2>

<div>This div element has a top margin of 100px, a right margin of 150px, a bottom margin of 100px, and a left margin of 80px.</div>

</body>
</html>

<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  margin: 25px 50px 75px 100px;
  background-color: lightblue;
}
</style>
</head>
<body>
```

```
<h2>The margin shorthand property - 4 values</h2>

<div>This div element has a top margin of 25px, a right margin of 50px, a bottom margin of 75px, and a left margin of 100px.</div>

<hr>
</body>
</html>

<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  margin: 25px 50px 75px;
  background-color: lightblue;
}
</style>
</head>
<body>

<h2>The margin shorthand property - 3 values</h2>
<div>This div element has a top margin of 25px, a right and left margin of 50px, and a bottom margin of 75px.</div>

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

# CSS Padding

The CSS `padding` properties are used to generate space around an element's content, inside of any defined borders.

CSS has properties for specifying the padding for each side of an element:

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  background-color: lightblue;
  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>
```

If the `padding` property has four values:

- **padding: 25px 50px 75px 100px;**
  - top padding is 25px
  - right padding is 50px
  - bottom padding is 75px
  - left padding is 100px

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  padding: 25px 50px 75px 100px;
  background-color: lightblue;
}
</style>
</head>
<body>

<h2>The padding shorthand property - 4 values</h2>

<div>This div element has a top padding of 25px, a right padding of 50px, a bottom padding of 75px, and a left padding of 100px.</div>

</body>
</html>
```

If the `padding` property has three values:

- **padding: 25px 50px 75px;**
  - top padding is 25px
  - right and left paddings are 50px
  - bottom padding is 75px

```
div {
  padding: 25px 50px 75px;
}
```

If the `padding` property has two values:

- **padding: 25px 50px;**
  - top and bottom paddings are 25px
  - right and left paddings are 50px

```
div {
  padding: 25px 50px;
}
```

If the `padding` property has one value:

- **padding: 25px;**
  - all four paddings are 25px

```
div {
  padding: 25px;
}
```

# CSS height and width Values

The `height` and `width` properties may have the following values:

- `auto` - This is default. The browser calculates the height and width
- `length` - Defines the height/width in px, cm, etc.
- `%` - Defines the height/width in percent of the containing block
- `initial` - Sets the height/width to its default value
- `inherit` - The height/width will be inherited from its parent value

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

```
  height: 200px;
  width: 50%;
  background-color: powderblue;
}
</style>
</head>
<body>

<h2>Set the height and width of an element</h2>

<div>This div element has a height of 200px and a width of 50%.</div>

</body>
</html>
```

# Setting max-width

The `max-width` property is used to set the maximum width of an element.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  max-width: 500px;
  height: 100px;
  background-color: powderblue;
}
</style>
</head>
<body>

<h2>Set the max-width of an element</h2>
```

This div element has a height of 100px and a max-width of 500px.

Resize the browser window to see the effect.

</body>
</html>

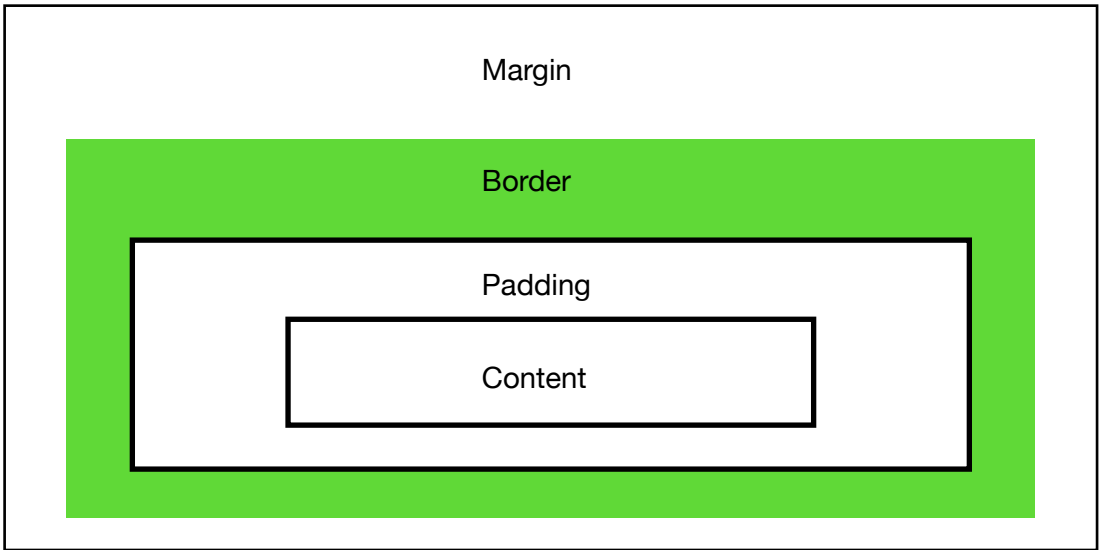
# The CSS Box Model

In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: content, padding, borders and margins. The image below illustrates the box model:

Explanation of the different parts:

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content



- **Margin** - Clears an area outside the border. The margin is transparent

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: lightgrey;
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
</style>
</head>
<body>

<h2>Demonstrating the Box Model</h2>

<p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.</p>

<div>This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</div>

</body>
</html>
```

# Text Color

The `color` property is used to set the color of the text. The color is specified by:

- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like “rgb(255,0,0)"

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  color: blue;
}

h1 {
  color: green;
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<p>This is an ordinary paragraph. Notice that this text is blue. The default text color for a page is defined in the body selector.</p>
<p>Another paragraph.</p>

</body>
</html>
```

**Text Color and Background Color**

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightgrey;
  color: blue;
```

```
}

h1 {
  background-color: black;
  color: white;
}

div {
  background-color: blue;
  color: white;
}
</style>
</head>
<body>

<h1>This is a Heading</h1>
<p>This page has a grey background color and a blue text.</
p>
<div>This is a div.</div>

</body>
</html>
```

# Text Alignment

The `text-align` property is used to set the horizontal alignment of a text.

A text can be left or right aligned, centered, or justified.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-align: center;
}
```

```
h2 {
  text-align: left;
}

h3 {
  text-align: right;
}
</style>
</head>
<body>

<h1>Heading 1 (center)</h1>
<h2>Heading 2 (left)</h2>
<h3>Heading 3 (right)</h3>

<p>The three headings above are aligned center, left and right.</p>

</body>
</html>
```

# Add a Decoration Line to Text

The `text-decoration-line` property is used to add a decoration line to text.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
```

```
  text-decoration: overline;
}

h2 {
  text-decoration: line-through;
}

h3 {
  text-decoration: underline;
}

p.ex {
  text-decoration: overline underline;
}
</style>
</head>
<body>

<h1>Overline text decoration</h1>
<h2>Line-through text decoration</h2>
<h3>Underline text decoration</h3>
<p class="ex">Overline and underline text decoration.</p>

<p><strong>Note:</strong> It is not recommended to
underline text that is not a link, as this often confuses
the reader.</p>

</body>
</html>
```

# Text Transformation

The `text-transform` property is used to specify uppercase and lowercase letters in a text.

```
<!DOCTYPE html>
<html>
<head>
<style>
p.uppercase {
  text-transform: uppercase;
}

p.lowercase {
  text-transform: lowercase;
}

p.capitalize {
  text-transform: capitalize;
}
</style>
</head>
<body>

<h1>Using the text-transform property</h1>

<p class="uppercase">This text is transformed to uppercase.</p>
<p class="lowercase">This text is transformed to lowercase.</p>
<p class="capitalize">This text is capitalized.</p>

</body>
</html>
```

# Text Indentation

The `text-indent` property is used to specify the indentation of the first line of a text:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-indent: 50px;
}
</style>
</head>
<body>

<h1>Using text-indent</h1>

<p>In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since. 'Whenever you feel like criticizing anyone,' he told me, 'just remember that all the people in this world haven't had the advantages that you've had.'</p>

</body>
</html>
```

# Letter Spacing

The `letter-spacing` property is used to specify the space between the characters in a text.

```
<!DOCTYPE html>
<html>
<head>
<style>
h2 {
  letter-spacing: 5px;
}

h3 {
  letter-spacing: -2px;
}
</style>
</head>
<body>

<h1>Using letter-spacing</h1>

<h2>This is heading 1</h2>
<h3>This is heading 2</h3>
</body>
</html>
```

# Word Spacing

The `word-spacing` property is used to specify the space between the words in a text.

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  word-spacing: 10px;
```

```
}

p.two {
  word-spacing: -2px;
}
</style>
</head>
<body>

<h1>Using word-spacing</h1>

<p>This is a paragraph with normal word spacing.</p>

<p class="one">This is a paragraph with larger word spacing.</p>

<p class="two">This is a paragraph with smaller word spacing.</p>

</body>
</html>
```

# Text Shadow

The `text-shadow` property adds shadow to text.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-shadow: 2px 2px;
```

```
}
</style>
</head>
<body>

<h1>Text-shadow effect!</h1>

</body>
</html>
```

Next, add a color (red) to the shadow:

# Text shadow effect!

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-shadow: 2px 2px red;
}
</style>
</head>
<body>

<h1>Text-shadow effect!</h1>

</body>
</html>
```

Then, add a blur effect (5px) to the shadow:

# Text shadow effect!

```
h1 {
  text-shadow: 2px 2px 5px red;
```

```
}
```

# Font Selection is Important

Choosing the right font has a huge impact on how the readers experience a website.

The right font can create a strong identity for your brand.

```
<!DOCTYPE html>
<html>
<head>
<style>
.p1 {
  font-family: "Times New Roman", Times, serif;
}

.p2 {
  font-family: Arial, Helvetica, sans-serif;
}

.p3 {
  font-family: "Lucida Console", "Courier New", monospace;
}
</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>
```

# Font Style

The `font-style` property is mostly used to specify italic text.

This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics

```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  font-style: normal;
}

p.italic {
  font-style: italic;
}

p.oblique {
  font-style: oblique;
}
</style>
</head>
<body>

<h1>The font-style property</h1>

<p class="normal">This is a paragraph in normal style.</p>
<p class="italic">This is a paragraph in italic style.</p>
```

```
<p class="oblique">This is a paragraph in oblique style.</p>

</body>
</html>
```

# Font Weight

The `font-weight` property specifies the weight of a font:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  font-weight: normal;
}

p.light {
  font-weight: lighter;
}

p.thick {
  font-weight: bold;
}

p.thicker {
  font-weight: 900;
}
</style>
</head>
<body>

<h1>The font-weight property</h1>
```

```
<p class="normal">This is a paragraph.</p>
<p class="light">This is a paragraph.</p>
<p class="thick">This is a paragraph.</p>
<p class="thicker">This is a paragraph.</p>

</body>
</html>
```

# Font Size

The `font-size` property sets the size of the text.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  font-size: 40px;
}

h2 {
  font-size: 30px;
}

p {
  font-size: 14px;
}
</style>
</head>
<body>
```

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

# Set Font Size With Em

To allow users to resize the text (in the browser menu), many developers use em instead of pixels.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  font-size: 2.5em; /* 40px/16=2.5em */
}

h2 {
  font-size: 1.875em; /* 30px/16=1.875em */
 }

p {
  font-size: 0.875em; /* 14px/16=0.875em */
}
</style>
</head>
```

```
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>Specifying the font-size in em allows all major browsers to resize the text.
Unfortunately, there is still a problem with older versions of IE. When resizing the text, it becomes larger/smaller than it should.</p>

</body>
</html>
```

**How To Use Google Fonts**

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
<style>
body {
  font-family: "Sofia", sans-serif;
}
</style>
</head>
<body>

<h1>Sofia Font</h1>
<p>Lorem ipsum dolor sit amet.</p>
<p>123456790</p>
```

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

**Use Multiple Google Fonts**

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Audiowide|Sofia|Trirong">
<style>
h1.a {
  font-family: "Audiowide", sans-serif;
}

h1.b {
  font-family: "Sofia", sans-serif;
}

h1.c {
  font-family: "Trirong", serif;
}
</style>
</head>
<body>

<h1 class="a">Audiowide Font</h1>
<h1 class="b">Sofia Font</h1>
<h1 class="c">Trirong Font</h1>

</body>
```

</html>

**Styling Links**

The four links states are:

- `a:link` - a normal, unvisited link
- `a:visited` - a link the user has visited
- `a:hover` - a link when the user mouses over it
- `a:active` - a link the moment it is clicked

```
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
  color: red;
}

/* visited link */
a:visited {
  color: green;
}

/* mouse over link */
a:hover {
  color: hotpink;
}

/* selected link */
a:active {
  color: blue;
}
</style>
</head>
<body>

<h2>Styling a link depending on state</h2>
```

```
<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order to be effective.</p>

</body>
</html>
```

# Text Decoration

The `text-decoration` property is mostly used to remove underlines from links:

```
<!DOCTYPE html>
<html>
<head>
<style>
a:link {
  text-decoration: none;
}

a:visited {
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}

a:active {
  text-decoration: underline;
```

```
}
</style>
</head>
<body>

<h2>Styling a link with text-decoration property</h2>

<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order to be effective.</p>

</body>
</html>
```

# Unordered Lists:

- Coffee
- Tea
- Coca Cola
  - Coffee
  - Tea
  - Coca Cola

# Ordered Lists:

1. Coffee
2. Tea
3. Coca Cola

- Coffee
- Tea

- Coca Cola

```
<!DOCTYPE html>
<html>
<head>
<style>
ul.a {
  list-style-type: circle;
}

ul.b {
  list-style-type: square;
}

ol.c {
  list-style-type: upper-roman;
}

ol.d {
  list-style-type: lower-alpha;
}
</style>
</head>
<body>

<h2>The list-style-type Property</h2>

<p>Example of unordered lists:</p>
<ul class="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

<ul class="b">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
```

```
<p>Example of ordered lists:</p>
<ol class="c">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="d">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

</body>
</html>
```

## CSS Tables

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid;
}
</style>
</head>
<body>

<h2>Add a border to a table:</h2>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
  </tr>
  <tr>
```

```
    <td>Lois</td>
    <td>Griffin</td>
  </tr>
</table>

</body>
</html>
```

**Full-Width Table**

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid;
}

table {
  width: 100%;
}
</style>
</head>
<body>

<h2>Full-width Table</h2>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
  </tr>
</table>
```

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

# Collapse Table Borders

The `border-collapse` property sets whether the table borders should be collapsed into a single border:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, td, th {
  border: 1px solid;
}

table {
  width: 100%;
  border-collapse: collapse;
}
</style>
</head>
<body>

<h2>Let the table borders collapse</h2>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  </tr>
  <tr>
    <td>Peter</td>
```

```
    <td>Griffin</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
  </tr>
</table>

</body>
</html>
```

# Horizontal Alignment

The `text-align` property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>.

```
<!DOCTYPE html>
<html>
<head>
<style>
table, td, th {
  border: 1px solid black;
}

table {
  border-collapse: collapse;
  width: 100%;
}

td {
  text-align: center;
}
```

```
</style>
</head>
<body>

<h2>The text-align Property</h2>

<p>This property sets the horizontal alignment (like left, right, or center) of the content in th or td.</p>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
    <td>$150</td>
  </tr>
  <tr>
    <td>Joe</td>
    <td>Swanson</td>
    <td>$300</td>
  </tr>
  <tr>
    <td>Cleveland</td>
```

```
    <td>Brown</td>
    <td>$250</td>
  </tr>
</table>

</body>
</html>
```

To left-align the content, force the alignment of <th> elements to be left-aligned, with the `text-align: left` property:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, td, th {
  border: 1px solid black;
}

table {
  border-collapse: collapse;
  width: 100%;
}

th {
  text-align: left;
}
</style>
</head>
<body>

<h2>The text-align Property</h2>
```

This property sets the horizontal alignment (like left, right, or center) of the content in th or td.

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
    <td>$150</td>
  </tr>
  <tr>
    <td>Joe</td>
    <td>Swanson</td>
    <td>$300</td>
  </tr>
  <tr>
    <td>Cleveland</td>
    <td>Brown</td>
    <td>$250</td>
  </tr>
</table>
```

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

# Vertical Alignment

The `vertical-align` property sets the vertical alignment (like top, bottom, or middle) of the content in <th> or <td>.

```
<!DOCTYPE html>
<html>
<head>
<style>
table, td, th {
  border: 1px solid black;
}

table {
  border-collapse: collapse;
  width: 100%;
}

td {
  height: 50px;
  vertical-align: bottom;
}
</style>
</head>
<body>

<h2>The vertical-align Property</h2>
```

```
<p>This property sets the vertical alignment (like top, bottom, or middle) of the content in th or td.</p>


<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
    <td>$150</td>
  </tr>
  <tr>
    <td>Joe</td>
    <td>Swanson</td>
    <td>$300</td>
  </tr>
  <tr>
    <td>Cleveland</td>
    <td>Brown</td>
    <td>$250</td>
  </tr>
</table>

</body>
```

</html>

# Table Padding

To control the space between the border and the content in a table, use the `padding` property on <td> and <th> elements:

| First Name | Last Name | Savings |
|---|---|---|
| Peter | Griffin | $ 100 |
| Lois | Griffin | $ 150 |
| Joe | Swanson | $ 300 |

```
<!DOCTYPE html>
<html>
<head>
<style>
table, td, th {
  border: 1px solid #ddd;
  text-align: left;
}

table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  padding: 15px;
}
</style>
</head>
<body>
```

```
<h2>The padding Property</h2>

<p>This property adds space between the border and the content in a table.</p>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
    <td>$150</td>
  </tr>
  <tr>
    <td>Joe</td>
    <td>Swanson</td>
    <td>$300</td>
  </tr>
  <tr>
    <td>Cleveland</td>
    <td>Brown</td>
    <td>$250</td>
  </tr>
</table>

</body>
</html>
```

# Horizontal Dividers

| Peter | Griffin | $ 100 |
|---|---|---|
| Lois | Griffin | $ 150 |
| Joe | Swanson | $ 300 |

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  padding: 8px;
  text-align: left;
  border-bottom: 1px solid #ddd;
}
</style>
</head>
<body>

<h2>Bordered Table Dividers</h2>
<p>Add the border-bottom property to th and td for horizontal dividers:</p>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  <th>Savings</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
    <td>$100</td>
  </tr>
```

```
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
    <td>$150</td>
  </tr>
  <tr>
    <td>Joe</td>
    <td>Swanson</td>
    <td>$300</td>
  </tr>
  <tr>
    <td>Cleveland</td>
    <td>Brown</td>
    <td>$250</td>
  </tr>
</table>

</body>
</html>
```

# Hoverable Table

Use the `:hover` selector on <tr> to highlight table rows on mouse over:

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  padding: 8px;
```

```
  text-align: left;
  border-bottom: 1px solid #ddd;
}

tr:hover {background-color: coral;}
</style>
</head>
<body>

<h2>Hoverable Table</h2>

<p>Move the mouse over the table rows to see the effect.</
p>

<table>
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Points</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
    <td>$150</td>
  </tr>
  <tr>
    <td>Joe</td>
```

```
    <td>Swanson</td>
    <td>$300</td>
  </tr>
  <tr>
    <td>Cleveland</td>
    <td>Brown</td>
    <td>$250</td>
  </tr>
</table>

</body>
</html>
```

# Responsive Table

A responsive table will display a horizontal scroll bar if the screen is too small to display the full content:

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  text-align: left;
  padding: 8px;
}
```

```
tr:nth-child(even) {background-color: #f2f2f2;}
</style>
</head>
<body>

<h2>Responsive Table</h2>
<p>A responsive table will display a horizontal scroll bar if the screen is too
small to display the full content. Resize the browser window to see the effect:</p>
<p>To create a responsive table, add a container element (like div) with <strong>overflow-x:auto</strong> around the table element:</p>

<div style="overflow-x: auto;">
  <table>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
    </tr>
    <tr>
      <td>Jill</td>
```

```
  <td>Smith</td>
  <td>50</td>
  <td>50</td>
  <td>50</td>
  <td>50</td>
  <td>50</td>
  <td>50</td>
  <td>50</td>
  <td>50</td>
  <td>50</td>
  <td>50</td>
 </tr>
 <tr>
  <td>Eve</td>
  <td>Jackson</td>
  <td>94</td>
  <td>94</td>
  <td>94</td>
  <td>94</td>
  <td>94</td>
  <td>94</td>
  <td>94</td>
  <td>94</td>
  <td>94</td>
  <td>94</td>
 </tr>
 <tr>
  <td>Adam</td>
  <td>Johnson</td>
  <td>67</td>
  <td>67</td>
```

```
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
    </tr>
  </table>
</div>

</body>
</html>
```

# Block-level Elements

A block-level element ALWAYS starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The  element is a block-level element.
Examples of block-level elements:

- 
- <h1> - <h6>
- <p>
- <form>
- <header>
- <footer>
- <section>

# Inline Elements

An inline element DOES NOT start on a new line and only takes up as much width as necessary.

This is an inline  element inside a paragraph.

Examples of inline elements:

- 
- <a>
- <img>

# The position Property

The `position` property specifies the type of positioning method used for an element.

There are five different position values:

- `static`
- `relative`
- `fixed`
- `absolute`
- `sticky`

# position: static;

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.static {
  position: static;
```

```
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: static;</h2>

<p>An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:</p>

<div class="static">
This div element has position: static;
</div>

</body>
</html>
```

# position: relative;

An element with `position: relative;` is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  left: 30px;
```

```
  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>
```

# position: fixed;

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. The top, right, bottom, and left properties are used to position the element.

```
<!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>
```

# position: absolute;

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  width: 400px;
```

```
  height: 200px;
  border: 3px solid #73AD21;
}

div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: absolute;</h2>

<p>An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):</p>

<div class="relative">This div element has position: relative;
  <div class="absolute">This div element has position: absolute;</div>
</div>

</body>
</html>
```

# position: sticky;

An element with `position: sticky;` is positioned based on the user's scroll position.

A sticky element toggles between `relative` and `fixed`, depending on the scroll position.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.sticky {
  position: -webkit-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>

<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>Scroll back up to remove the stickyness.</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.

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.

</body>
</html>

# Positioning Text In an Image

How to position text over an image:

```
<!DOCTYPE html>
<html>
<head>
<style>
.container {
  position: relative;
}

.topleft {
  position: absolute;
  top: 8px;
  left: 16px;
  font-size: 18px;
}
```

```
.topright {
  position: absolute;
  top: 8px;
  right: 16px;
  font-size: 18px;
}

.bottomleft {
  position: absolute;
  bottom: 8px;
  left: 16px;
  font-size: 18px;
}

.bottomright {
  position: absolute;
  bottom: 8px;
  right: 16px;
  font-size: 18px;
}

.center {
  position: absolute;
  top: 50%;
  width: 100%;
  text-align: center;
  font-size: 18px;
}

img {
  width: 100%;
```

```
  height: auto;
  opacity: 0.3;
}
</style>
</head>
<body>

<h2>Image Text</h2>
<p>Add some text to an image in the top left corner:</p>

<div class="container">
  <img src="img_5terre_wide.jpg" alt="Cinque Terre" width="1000" height="300">
  <div class="topleft">Top Left</div>
   <div class="topright">Top Right</div>
 <div class="bottomleft">Bottom Left</div>
<div class="bottomright">Bottom Right</div>
 <div class="center">Centered</div>
</div>
</div>
</body>
</html>
```

# The float Property

The `float` property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The `float` property can have one of the following values:

- `left` - The element floats to the left of its container
- `right` - The element floats to the right of its container

- none - The element does not float (will be displayed just where it occurs in the text). This is default

float-left

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  float: right;
}
</style>
</head>
<body>

<h2>Float Right</h2>

<p>In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.</p>

<p><img src="pineapple.jpg" alt="Pineapple" style="width:170px;height:170px;margin-left:15px;">
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.</p>

</body>
</html>
```

float:right;

```
img {
  float: right;
}
```

float:none

```
img {
  float: none;
}
```

Float Next To Each Other

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  float: left;
  padding: 15px;
}

.div1 {
  background: red;
}

.div2 {
  background: yellow;
}

.div3 {
  background: green;
}
</style>
</head>
<body>

<h2>Float Next To Each Other</h2>
```

```
<p>In this example, the three divs will float next to
each other.</p>

<div class="div1">Div 1</div>
<div class="div2">Div 2</div>
<div class="div3">Div 3</div>

</body>
</html>
```

# The display: inline-block Value

Compared to `display: inline`, the major difference is that `display: inline-block` allows to set a width and height on the element.

Also, with `display: inline-block`, the top and bottom margins/paddings are respected, but with `display: inline` they are not.

Compared to `display: block`, the major difference is that `display: inline-block` does not add a line-break after the element, so the element can sit next to other elements.

```
<!DOCTYPE html>
<html>
<head>
<style>
span.a {
  display: inline; /* the default for span */
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}

span.b {
```

```
  display: inline-block;
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}

span.c {
  display: block;
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow;
}
</style>
</head>
<body>

<h1>The display Property</h1>

<h2>display: inline</h2>
<div>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Vestibulum consequat scelerisque
elit sit amet consequat. Aliquam erat volutpat. <span
class="a">Aliquam</span> <span class="a">venenatis</
span> gravida nisl sit amet facilisis. Nullam cursus
fermentum velit sed laoreet. </div>

<h2>display: inline-block</h2>
<div>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Vestibulum consequat scelerisque
elit sit amet consequat. Aliquam erat volutpat. <span
class="b">Aliquam</span> <span class="b">venenatis</
span> gravida nisl sit amet facilisis. Nullam cursus
fermentum velit sed laoreet. </div>

<h2>display: block</h2>
```

```
<div>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Vestibulum consequat scelerisque
elit sit amet consequat. Aliquam erat volutpat. <span
class="c">Aliquam</span> <span class="c">venenatis</
span> gravida nisl sit amet facilisis. Nullam cursus
fermentum velit sed laoreet. </div>

</body>
</html>
```

**CSS Opacity / Transparency**

The `opacity` property specifies the opacity/transparency of an element.

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  opacity: 0.5;
}
</style>
</head>
<body>


<h1>Image Transparency</h1>
<p>The opacity property specifies the transparency of an
element. The lower the value, the more transparent:</p>


<p>Image with 50% opacity:</p>
<img src="img_forest.jpg" alt="Forest" width="170"
height="100">
```

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

# Transparent Hover Effect

The `opacity` property is often used together with the `:hover` selector to change the opacity on mouse-over:

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  opacity: 0.5;
}

img:hover {
  opacity: 1.0;
}
</style>
</head>
<body>

<h1>Image Transparency</h1>
<p>The opacity property is often used together with
the :hover selector to change the opacity on mouse-over:</p>
<img src="img_forest.jpg" alt="Forest" width="170"
height="100">
<img src="img_mountains.jpg" alt="Mountains" width="170"
height="100">
<img src="img_5terre.jpg" alt="Italy" width="170"
height="100">
```

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

# 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.

```
<!DOCTYPE html>
<html>
<body>


<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

<p>Note: We use href="#" for test links. In a real web site this would be URLs.</p>


</body>
</html>
```

**Vertical Navigation Bar**

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

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

li a {
  display: block;
  width: 60px;
  background-color: #dddddd;
}
</style>
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

<p>A background color is added to the links to show the link area.</p>
<p>Notice that the whole link area is clickable, not just the text.</p>

</body>
</html>
```

**Active/Current Navigation Link**

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 200px;
  background-color: #f1f1f1;
}

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;
}
</style>
</head>
<body>
```

```
<h2>Vertical Navigation Bar</h2>
<p>In this example, we create an "active" class with a green background color and a white text. The class is added to the "Home" link.</p>

<ul>
  <li><a class="active" href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

</body>
</html>
```

## CSS Flex

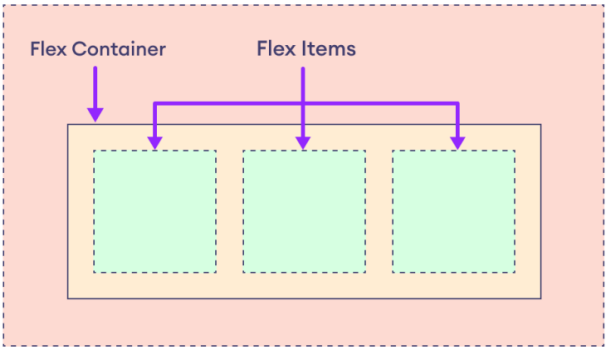
The flex layout provides an efficient and flexible way to layout, align, and distribute the space among elements within a container.

Before flexbox layouts, designers and developers had to use table, float, and position properties to create complex layouts.

The flexbox layout provides an easy way to create responsive and dynamic designs that adapt to various screen sizes and devices.

### Flex Container and Flex Items

In order to create a flex layout, we first need to set up a flex container. Any element can be set as a flex container using display: flex declaration.




```
<!-- A div element (flex container) having three paragraphs (flex items) → -->
<div>
    <p>First Item</p>
    <p>Second Item</p>
    <p>Third Item</p>
</div>

/* CSS */
div {
    display: flex;
}
```

The distribution and alignment of space within the flex container are adjusted with the help of the following properties.

- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content
- gap, row-gap, column-gap

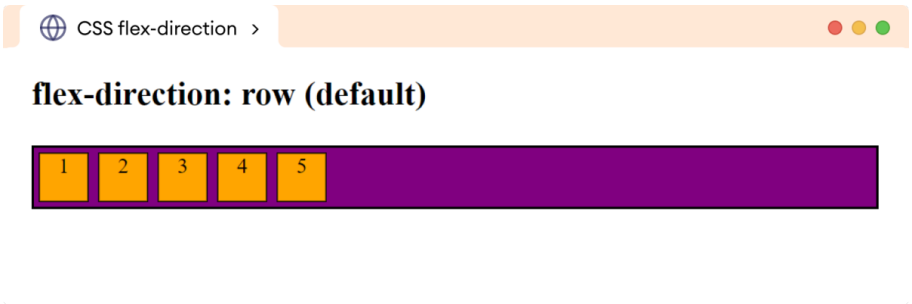
## Flex Direction

The flex-direction property specifies the direction of flex items within the flex container.

The possible value of the flex-direction property is row (default value), row-reverse, column, and column-reverse.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>CSS flex-direction</title>
    </head>
    <body>
        <h2>flex-direction: row (default)</h2>
        <div class="container">
            <div class="box">1</div>
            <div class="box">2</div>
            <div class="box">3</div>
            <div class="box">4</div>
            <div class="box">5</div>
        </div>
    </body>
</html>


/* CSS */

div.container {
```

```
    display: flex;
    flex-direction: row;
    border: 2px solid black;
    background-color: purple;
}
div.box {
    width: 40px;
    height: 40px;
    text-align: center;
    border: 1px solid black;
    background-color: orange;
    margin: 4px;
}
```




## flex-wrap: nowrap

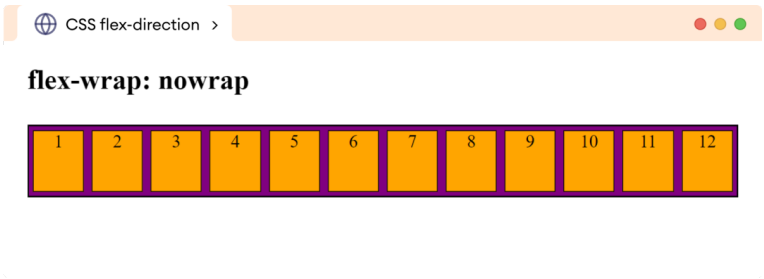
The nowrap value of the flex-wrap property keeps flex items squeezed in the direction set by flex-direction

Here, all the flex items are squeezed in the horizontal direction, and their original width is ignored.

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

```
        <meta name="viewport" content="width=device-width, initial-scale=1.0" /
>
        <link rel="stylesheet" href="style.css" />
        <title>CSS flex-wrap</title>
    </head>

    <body>
        <h2>flex-wrap: nowrap</h2>
        <div class="container">
            <div class="box">1</div>
            <div class="box">2</div>
            <div class="box">3</div>
            <div class="box">4</div>
            <div class="box">5</div>
            <div class="box">6</div>
            <div class="box">7</div>
            <div class="box">8</div>
            <div class="box">9</div>
            <div class="box">10</div>
            <div class="box">11</div>
            <div class="box">12</div>
        </div>
    </body>
</html>

/* CSS */

div.container {
    display: flex;
    flex-direction: row;
    /* prevents wrapping of flex items; default value */
    flex-wrap: nowrap;
    border: 2px solid black;
```

```
    background-color: purple;
}

div.box {
    width: 80px;
    height: 60px;
    text-align: center;
    border: 1px solid black;
    background-color: orange;
    margin: 4px;
}
```



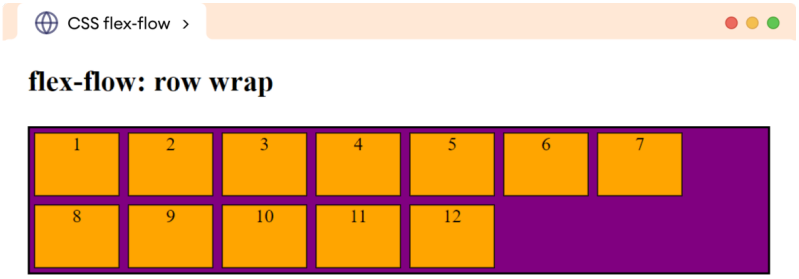

```
/* wraps the flex items in multiple line */
    flex-wrap: wrap;
/* wraps the flex items in multiple lines */
    flex-wrap: wrap-reverse;
```

## Flex Flow

The flex-flow is a shorthand property to specify the flex-direction and flex-wrap property value

```
<!DOCTYPE html>
<html lang="en">


    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
        <link rel="stylesheet" href="style.css" />
        <title>CSS flex-flow</title>
    </head>

    <body>
        <h2>flex-flow: row wrap</h2>
        <div class="container">
            <div class="box">1</div>
            <div class="box">2</div>
            <div class="box">3</div>
            <div class="box">4</div>
            <div class="box">5</div>
            <div class="box">6</div>
            <div class="box">7</div>
            <div class="box">8</div>
            <div class="box">9</div>
<div class="box">10</div>
            <div class="box">11</div>
            <div class="box">12</div>
        </div>

    </body>
</html>
```



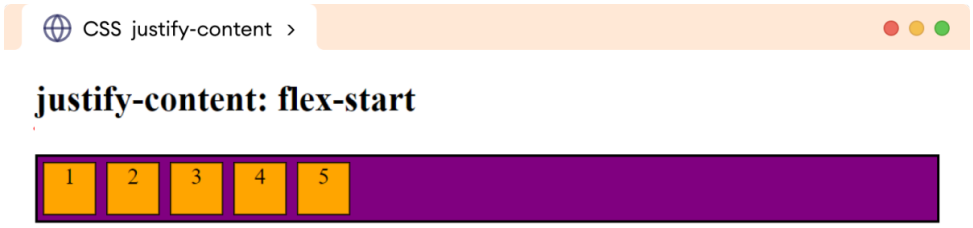
```
/* CSS */
div.container {
    display: flex;

    /* sets the flex-direction to row and flex-wrap to wrap value */
    flex-flow: row wrap;
    border: 2px solid black;
    background-color: purple;
}

div.box {
    width: 80px;
    height: 60px;
    text-align: center;
    border: 1px solid black;
    background-color: orange;
    margin: 4px;
}
```

## justify-content: flex-start

The flex-start value aligns the flex items at the start of the main axis (left for the flex-direction: row/row-reverse and top for the flex-direction: column/column-reverse.)

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
```

```
        <title>CSS justify-content</title>
    </head>
    <body>
        <h2>justify-content: flex-start</h2>
        <div class="container">
            <div class="box">1</div>
            <div class="box">2</div>
            <div class="box">3</div>
            <div class="box">4</div>
            <div class="box">5</div>
        </div>
    </body>
</html>
/*  CSS */
div.container {
    display: flex;
    /* default value */
    justify-content: flex-start;

    border: 2px solid black;
    background-color: purple;
}
div.box {
    width: 40px;
    height: 40px;
    text-align: center;
    border: 1px solid black;
    background-color: orange;
    margin: 4px;
}
```




## justify-content: center

```
div.container {
    display: flex;
    justify-content: center;
    border: 2px solid black;
    background-color: purple;
}
```

justify-content: flex-end

```
div.container {
    display: flex;
    justify-content: flex-end;
    border: 2px solid black;
    background-color: purple;
}
```

justify-content: space-between

```
div.container {
    display: flex;
    justify-content: space-between;
    border: 2px solid black;
    background-color: purple;
}
```

justify-content: space-around

```
div.container {
    display: flex;
    justify-content: space-around;
    border: 2px solid black;
```

```
    background-color: purple;
}

justify-content: space-evenly

div.container {
    display: flex;
    justify-content: space-evenly;
    border: 2px solid black;
    background-color: purple;
}
```

## Align Items

The align-items property distributes the available space between the flex items along the cross axis.

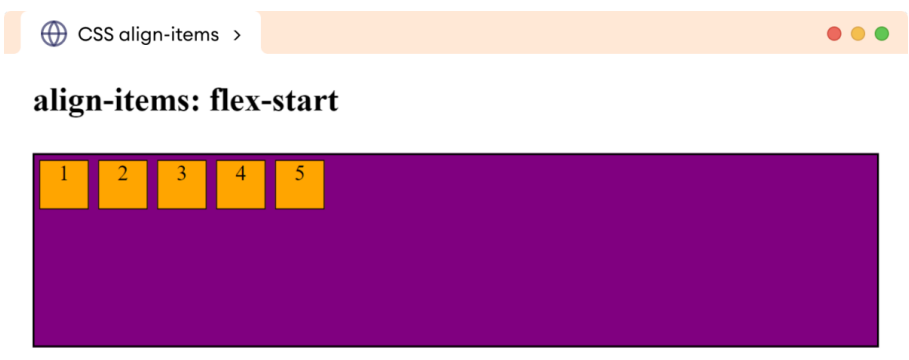
The space is distributed vertically for flex-direction: row and horizontally for flex-direction: column.

### align-items: flex-start

The flex-start value aligns the flex items at the start of the cross-axis (top for the flex-direction: row/row-reverse and left for the flex-direction: column/column-reverse).

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>CSS align-items</title>
    </head>
    <body>
        <h2>align-items: flex-start</h2>
```

```
        <div class="container">
            <div class="box">1</div>
            <div class="box">2</div>
            <div class="box">3</div>
            <div class="box">4</div>
            <div class="box">5</div>
        </div>
    </body>
</html>

div.container {
    height: 160px;
    display: flex;
    align-items: flex-start;
    border: 2px solid black;
    background-color: purple;
}

div.box {
    width: 40px;
    height: 40px;
    text-align: center;
    border: 1px solid black;
    background-color: orange;
    margin: 4px;
}
```



```
align-items: center;
align-items: flex-end;

align-items: baseline
div.container {
    height: 160px;
    display: flex;

    /* arranges the flex items with respective to baseline of flex items */
    align-items: baseline;

    border: 2px solid black;
    background-color: purple;
}

div.box {
    width: 40px;
    height: 40px;
    text-align: center;
    border: 1px solid black;
    background-color: orange;
    margin: 4px;
}

div.box3 {
    width: 90px;
    height: 90px;

    /* sets the line-height */
    line-height: 70px;
    background-color: skyblue;
```

```
}

div.box4 {
    width: 100px;
    height: 100px;

    /* sets the line-height */
    line-height: 140px;
    background-color: greenyellow;
}

align-content: flex-start;
align-content: flex-end;
align-content: center;
align-content: space-between;
align-content: space-around;
align-content: space-evenly;
```

## Gap Between Flex Items

The row-gap and column-gap properties control the spacing between the flex items within the flex container.

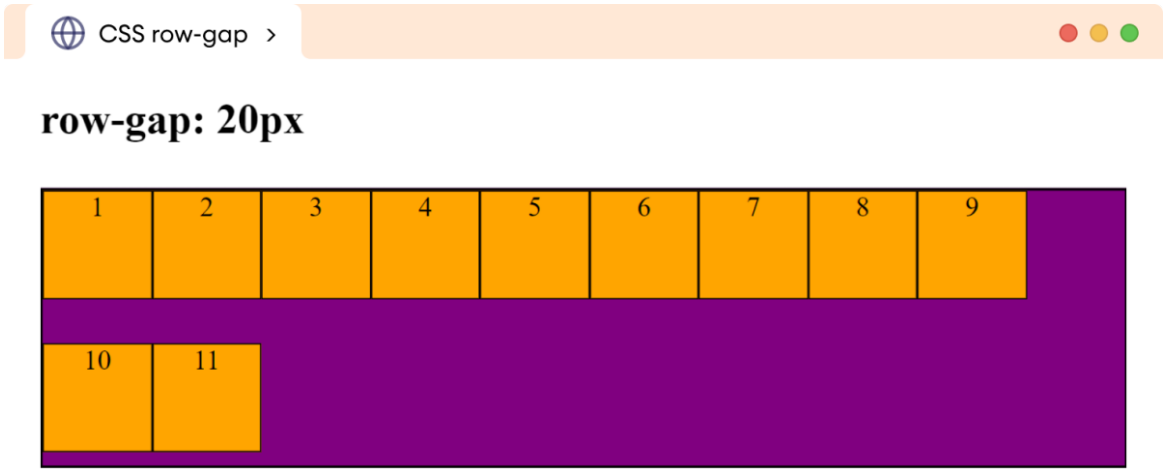
## CSS row-gap Property

The row-gap property sets the gap between rows of flex items across the cross-axis

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

```
        <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>CSS row-gap</title>
    </head>

    <body>
        <h2>row-gap: 20px</h2>
        <div class="container">
            <div class="box box1">1</div>
            <div class="box box2">2</div>
            <div class="box box3">3</div>
            <div class="box box4">4</div>
            <div class="box box5">5</div>
            <div class="box box6">6</div>
            <div class="box box7">7</div>
            <div class="box box8">8</div>
            <div class="box box9">9</div>
            <div class="box box10">10</div>
            <div class="box box11">11</div>
        </div>
    </body>
</html>

/* CSS */
div.container {
    height: 180px;
    display: flex;
    flex-wrap: wrap;
    /* creates a gap of 20px */
    row-gap: 20px;
    border: 2px solid black;
```

```
    background-color: purple;
}
div.box {
    width: 70px;
    height: 70px;
    text-align: center;
    border: 1px solid black;
    background-color: orange;
}
```




CSS column-gap Property

```
/* creates a gap of 20px */
    column-gap: 20px;
```

CSS gap Property

```
/* creates 40px gap between rows and columns */
    gap: 40px;
```

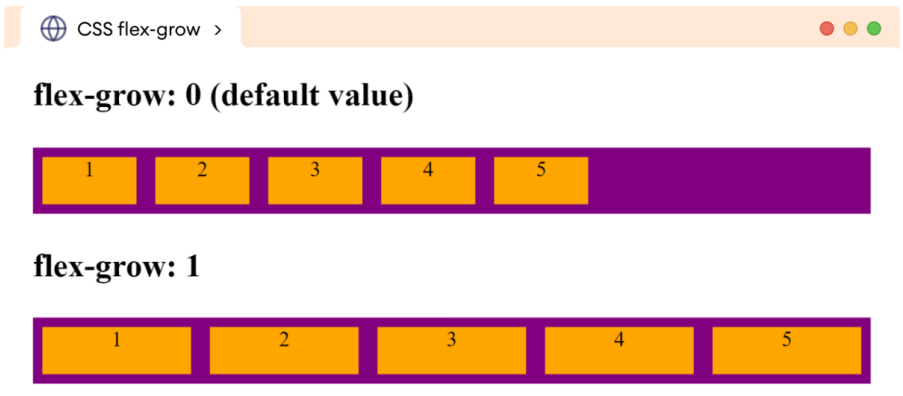
## CSS Flex-Grow Property

The flex-grow property determines how much flex items expand relative to the other items in the flex container.

The default value of the flex-grow is 0, meaning flex items do not expand by default.

```
<!DOCTYPE html>
<html lang="en">

    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" /
>
        <link rel="stylesheet" href="style.css" />
        <title>CSS flex-grow</title>
    </head>

    <body>
        <h2>flex-grow: 0 (default value)</h2>

        <!-- container without flex-grow -->
        <div class="container container1">
            <div class="box">1</div>
            <div class="box">2</div>
            <div class="box">3</div>
            <div class="box">4</div>
            <div class="box">5</div>
        </div>
      <h2>flex-grow: 1</h2>
        <!-- container with flex-grow -->
        <div class="container container2">
            <div class="box">1</div>
            <div class="box">2</div>
            <div class="box">3</div>
            <div class="box">4</div>
            <div class="box">5</div>
```

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

/* styles for both containers */
div.container {
    display: flex;
    flex-direction: row;
    background-color: purple;
}

/* styles for flex items of both containers */
div.box {
    width: 80px;
    height: 40px;
    text-align: center;
    background-color: orange;
    margin: 8px;
}
/* styles for second container  */
div.container2 div.box {
    flex-grow: 1;
}
```



## CSS Flex-Shrink Property

The flex-shrink property determines how much flex items shrink relative to the other items when there is not enough space in the flex container.

The default value of the flex-shrink is 1, meaning flex items shrink equally to fit within the available space.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>CSS flex-shrink</title>
    </head>
    <body>
        <h2>flex-shrink: 1 (default value) for all flex items</h2>
        <div class="container container1">
            <div class="box box1">1</div>
            <div class="box box2">2</div>
            <div class="box box3">3</div>
            <div class="box box4">4</div>
            <div class="box box5">5</div>
            <div class="box box6">6</div>
        </div>

        <h2>flex-shrink: 2 on the third item</h2>
        <div class="container container2">
            <div class="box box1">1</div>
            <div class="box box2">2</div>
```

```
            <div class="box box3">3</div>
            <div class="box box4">4</div>
            <div class="box box5">5</div>
            <div class="box box6">6</div>
        </div>
    </body>
</html>

/*  CSS */

/* styles for flex container */
div.container {
    display: flex;
    flex-direction: row;
    background-color: purple;
}

/* style for flex items */
div.box {
    width: 200px;
    height: 40px;
    text-align: center;
    background-color: orange;
    margin: 8px;
}

/* styles the third flex item of the second container */
div.container2 div.box3 {
    flex-shrink: 2;
}
```

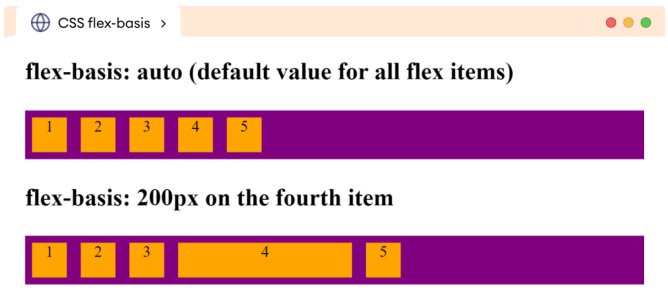
## CSS Flex-Basis Property

The flex-basis property defines the starting size of the flex item before any wrapping, growing, or shrinking occurs. It is an alternative to the width and height of the flex items.

By default, the flex-basis property is set to auto, which uses the specified width/height property value for the item size.

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8" />
      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
      <link rel="stylesheet" href="style.css" />
      <title>CSS flex-basis</title>
   </head>
   <body>
      <h2>flex-basis: auto (default value for all flex items)</h2>
      <div class="container container1">
         <div class="box box1">1</div>
         <div class="box box2">2</div>
         <div class="box box3">3</div>
         <div class="box box4">4</div>
         <div class="box box5">5</div>
      </div>
      <h2>flex-basis: 200px on the fourth item</h2>
      <div class="container container2">
         <div class="box box1">1</div>
         <div class="box box2">2</div>
         <div class="box box3">3</div>
         <div class="box box4">4</div>
```

```
            <div class="box box5">5</div>
        </div>
    </body>
</html>

/* CSS */
/* styles for flex container */
div.container {
    display: flex;
    flex-direction: row;
    background-color: purple;
}
/* style for flex items */
div.box {
    width: 40px;
    height: 40px;
    text-align: center;
    background-color: orange;
    margin: 8px;
}
/* styles the fourth flex item of the second container */
div.container2 div.box4 {
    flex-basis: 200px;
}
```



## Responsive Layout Using CSS Flex

CSS Flexbox provides an easy way to create responsive and dynamic designs that adapt to various screen sizes and devices.

The responsive layout allows the webpage to reshape and resize based on the size of the screen it's viewed on. This ensures a consistent user experience.

### Column Layouts Using CSS Flex

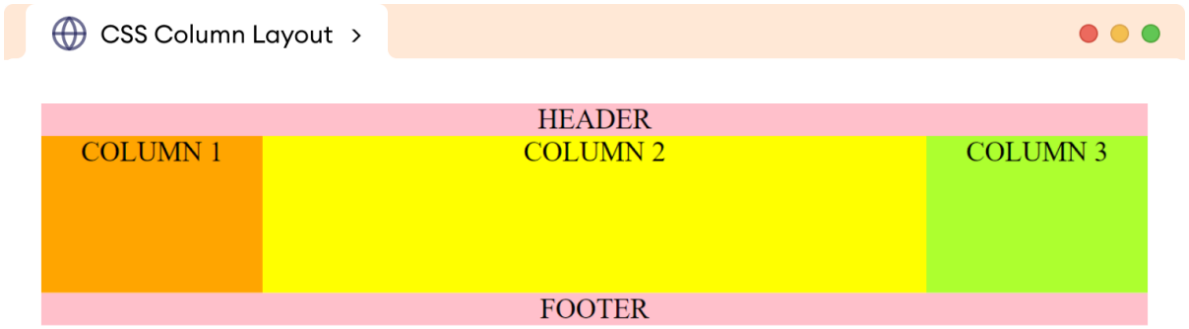
Column layout refers to the arrangement of content on a webpage in multiple columns. This allows us to organize and present information in a structured and visually appealing manner.

There are several common column layout configurations: 3-column, 2-column, and 1-column layouts.

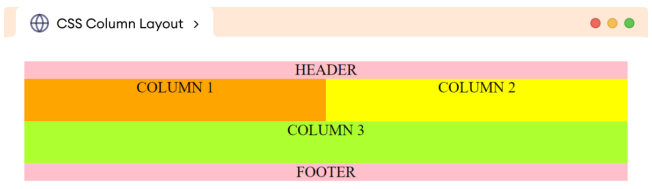
```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8" />
      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
      <link rel="stylesheet" href="style.css" />
      <title>CSS Column Layout</title>
   </head>
   <body>
      <header>HEADER</header>
      <main>
         <div class="column column1">COLUMN 1</div>
```

```
            <div class="column column2">COLUMN 2</div>
            <div class="column column3">COLUMN 3</div>
        </main>
        <footer>FOOTER</footer>
    </body>
</html>

/* CSS */
/* making a three column layout */
main {
    display: flex;
    flex-wrap: wrap;
    height: 100px;
    text-align: center;
}

.column1 {
    /* width takes 20% of the flex container */
    width: 20%;
    background-color: orange;
}

.column2 {
    /* width takes 60% of the flex container */
    width: 60%;
    background-color: yellow;
}

.column3 {
    /* width takes 20% of the flex container */
```

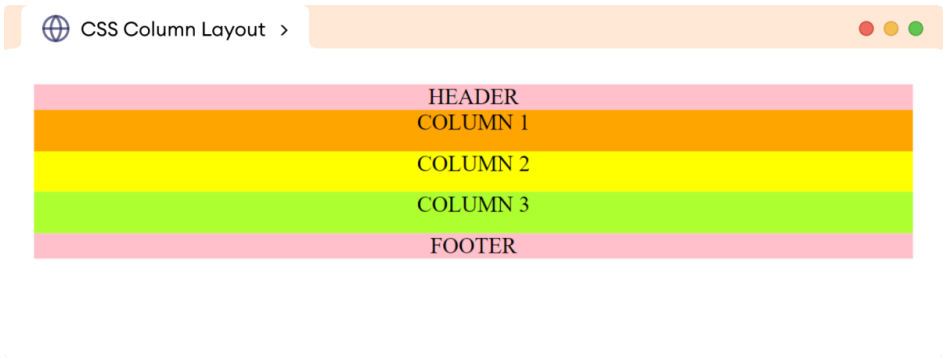
```
    width: 20%;
    background-color: greenyellow;
}

header,
footer {
    background-color: pink;
    text-align: center;
}
```




```
@media screen and (max-width: 768px) {
/* set column1 and column2 to 50% width each */
    .column1,
    .column2 {
        width: 50%;
    }

/* set column3 to 100% width */
    .column3 {
        width: 100%;
    }
}
```



```
@media screen and (max-width: 450px) {
    .column1,
    .column2,
    .column3 {
        width: 100%;
    }
}
```




## Example: Image Gallery Using CSS Flex

The flex property can be used to create a responsive image gallery that adapts to different screen sizes.

```
<!DOCTYPE html>
<html lang="en">


    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>CSS Flex Gallery</title>
    </head>


    <body>
        <div class="gallery">
```

```
        <img
            src="https://images.pexels.com/photos/3408744/pexels-
photo-3408744.jpeg"
            alt="Northern Winter Sky Image"
        />
        <img
            src="https://images.pexels.com/photos/1142950/pexels-
photo-1142950.jpeg"
            alt="Shining Stars Image"
        />
        <img
            src="https://images.pexels.com/photos/3933881/pexels-
photo-3933881.jpeg"
            alt="A River Flowing Image"
        />
        <img
            src="https://images.pexels.com/photos/5409751/pexels-
photo-5409751.jpeg"
            alt="A cloudy Mountain Image"
        />
        <img
            src="https://images.pexels.com/photos/4101555/pexels-
photo-4101555.jpeg"
            alt="A Winter Rainbow Image"
        />
        <img
            src="https://images.pexels.com/photos/443446/pexels-
photo-443446.jpeg"
            alt="A Clean Mountain Lake"
        />
      </div>
   </body>
</html>
```

```
/* CSS */
/* flex container */
div.gallery {
    display: flex;
    flex-wrap: wrap;
    Border: 1px solid black;
}

/* flex items */
img {
    width: 33%;
    height: 180px;
    margin: 2px;
}
```

For tablet devices, we can change the three-column layout above into a two-column layout.

```
@media screen and (max-width: 768px) {
    img {
        width: 49%;
        height: 250px;
            margin: 16px;
    }
}
```

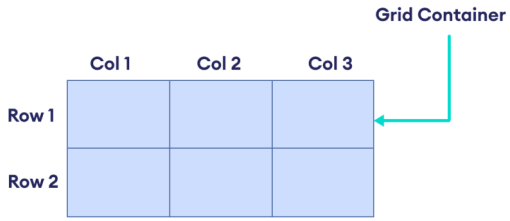
Finally, for mobile devices, we can adjust the flex item's width to span 100% of the container.

```
@media screen and (max-width: 480px) {
    img {
        width: 100%;
        height: 400px;
    }
}
```

## CSS Grid Introduction

The CSS Grid is a two-dimensional layout system that allows designers and developers to create complex and responsive layouts with ease.

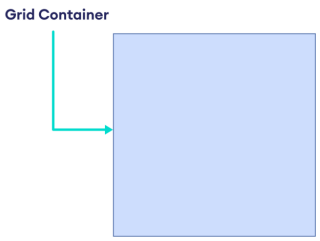
Grid layout creates a grid structure of rows and columns and positions elements within the grid's individual cells.




### 1. Set up the Grid Container

The first step for a grid-based layout is to define a grid container. This container will hold the grid items.

The display: grid declaration converts an element as a grid container.



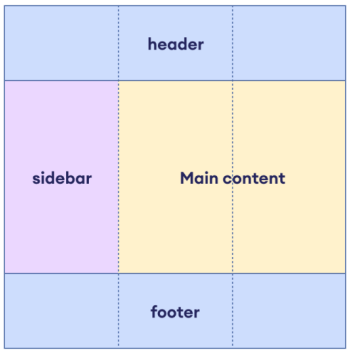
### 2. Create a Grid Structure

This second step involves specifying a structure for the grid by defining rows and columns.

The grid-template-rows and grid-template-column properties allow to specify rows and columns in a grid container.

### 3. Place Items in Grid Cells

The third and final step involves placing items into the individual cells of the grid. These items can be any HTML elements that we want to position within the grid layout.




## CSS Grid Container

CSS grid container is a HTML element that serves as a parent element of the grid items.

## CSS Grid Container Properties

The grid container has the following set of properties:

**Setting up the Grid:**

- display: grid - Turns an HTML element into a grid container.

**Specifying the Grid Rows and Columns:**

- grid-template-columns - Defines the size and number of columns.
- grid-template-rows - Defines the size and number of rows.
- grid-template-areas - Defines named grid areas where the grid items can be placed.
- grid-template - A shorthand for defining one property's rows, columns, and areas.

**Specifying the Grid Row and Column Gap:**

- grid-column-gap - Defines the gap between columns.
- grid-row-gap - Defines the gap between rows.
- grid-gap - A shorthand for defining both row and column gaps.

**Aligning the Grid Items:**

- justify-items - Aligns items horizontally within their grid cell.
- align-items - Aligns items vertically within their grid cell.
- place-items - A shorthand for aligning items both horizontally and vertically.

**Aligning the Grid:**

- justify-content - Aligns the grid along the horizontal axis.
- align-content - Aligns the grid along the vertical axis.

- place-content - A shorthand for aligning content both horizontally and vertically.

**Setting Automatic Grid Tracks:**

- grid-auto-columns - Defines the size of automatically generated columns.
- grid-auto-rows - Defines the size of automatically generated rows.
- grid-auto-flow - Defines the placement of auto-generated grid items.

**Shorthand Grid Property:**

- grid - A shorthand property for defining multiple grid properties in a single declaration.

## CSS display Property

The grid value of the display property sets the element as a grid container. For example,

```
<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="style.css" />
    <title>CSS Grid Container</title>
  </head>
```

```
    <body>
        <div class="container">
            <div class="item">1</div>
            <div class="item">2</div>
            <div class="item">3</div>
            <div class="item">4</div>
            <div class="item">5</div>
            <div class="item">6</div>
            <div class="item">7</div>
            <div class="item">8</div>
            <div class="item">9</div>
        </div>
    </body>

</html>

/* CSS */
div.container {

    /* sets the element as a grid container */
    display: grid;

    width: 400px;
    border: 2px solid black;
    background-color: orange;
    padding: 12px;
}

/* styles all grid items */
div.item {
    border: 1px solid black;
```

```
    background-color: greenyellow;
    text-align: center;
    padding: 5px;
    font-weight: bold;
}

/* creates a three columns of size 100px, 200px, 100px respectively */
    grid-template-columns: 100px 200px 100px;

or

grid-template-columns: 100px 25% 3em;

/* sets three rows of size 100px, 60px, and 80px respectively */
    grid-template-rows: 100px 60px 80px;
```

## CSS Fractional (Fr) Unit

The fractional unit (fr) divides the available space in the grid container into fractions. It distributes available space within the grid container among columns or rows in a flexible and dynamic way.

```
/* column-1, column-2, column-3 */
    grid-template-columns: 1fr 2fr 1fr;

    /* row-1, row-2, row-3 */
    grid-template-rows: 1fr 1fr 2fr;
```

## CSS grid-column-gap Property

The grid-column-gap property defines the spacing between columns in a grid container.

```
div.container {
    display: grid;

    /* defines 3 rows of 80px each */
    grid-template-rows: repeat(3, 80px);
```

```
    /* defines 3 columns of equal fraction */
    grid-template-columns: repeat(3, 1fr);

    /* creates a column gap of 20px */
    grid-column-gap: 20px;

    width: 400px;
    border: 2px solid black;
    background-color: orange;
}
```

## CSS grid-row-gap Property

The grid-row-gap property defines the spacing between the rows of the grid container.

```
div.container {
    display: grid;

    /* defines three rows of 80px each */
    grid-template-rows: repeat(3, 80px);

    /* defines three columns of equal fraction */
    grid-template-columns: repeat(3, 1fr);

    /* creates a  row gap of 20px */
    grid-row-gap: 20px;

    width: 400px;
    border: 2px solid black;
    background-color: orange;
}
```

## CSS grid-gap property

The grid-gap is a shorthand property to specify the grid-column-gap and grid-row-gap simultaneously

```
div.container {
    display: grid;
```

```
    /* defines three rows of 80px each */
    grid-template-rows: repeat(3, 80px);

    /* defines three columns of equal fraction */
    grid-template-columns: repeat(3, 1fr);

    /* creates a  row and column gap of 20px */
    grid-gap: 20px;

    width: 400px;
    border: 2px solid black;
    background-color: orange;
}
```

## CSS justify-content Property

The justify-content property controls the horizontal alignment of the grid within the grid container.

```
div.container {
    display: grid;

    /* defines rows and columns */
    grid-template-rows: repeat(2, 80px);
    grid-template-columns: repeat(3, 40px);

    /* aligns the entire grid to center in the grid container*/
    justify-content: center;

    width: 400px;
    gap: 12px;
    border: 2px solid black;
    background-color: orange;
}
```

## CSS align-content Property

The align-content property controls the alignment of the grid within the grid container along the horizontal direction.

```
div.container {
```

```
    display: grid;
    height: 180px;

    /* defines rows and columns */
    grid-template-rows: repeat(2, 30px);
    grid-template-columns: repeat(3, 80px);

    /* aligns the grid vertically to center of grid container */
    align-content: center;

    width: 400px;
    gap: 12px;
    border: 2px solid black;
    background-color: orange;
}
```

## Responsive Image Gallery Using CSS Grid

CSS grid properties can be used to create a responsive image gallery that adapts to different screen sizes.

```
<!DOCTYPE html>
<html lang="en">


    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-
scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>CSS Grid Gallery</title>
    </head>


    <body>
        <div class="gallery">


            <img class="four-grid-cells"
```

```
            src="https://images.pexels.com/photos/3408744/pexels-photo-3408744.jpeg"
            alt="Northern Winter Sky Image" />

        <img
            src="https://images.pexels.com/photos/1142950/pexels-photo-1142950.jpeg"
            alt="Shining Stars Image" />

        <img class="wide-image"
            src="https://images.pexels.com/photos/3933881/pexels-photo-3933881.jpeg"
            alt="A River Flowing Image" />

        <img
            src="https://images.pexels.com/photos/5409751/pexels-photo-5409751.jpeg"
            alt="A cloudy Mountain Image" />

        <img
            src="https://images.pexels.com/photos/4101555/pexels-photo-4101555.jpeg"
            alt="A Winter Rainbow Image" />

        <img class="wide-image"
            src="https://images.pexels.com/photos/443446/pexels-photo-443446.jpeg"
            alt="A Clean Mountain Lake" />

        <img
            src="https://images.pexels.com/photos/1142950/pexels-photo-1142950.jpeg"
            alt="Shining Stars Image" />
```

```
        </div>
    </body>

</html>
```

```
/* CSS */

.gallery {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    grid-auto-rows: auto;
    grid-auto-flow: dense;
    gap: 6px;
}

img {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

img.four-grid-cells {
    grid-row: span 2 / auto;
    grid-column: span 2 / auto;
}

img.wide-image {
    grid-column: span 2 / auto;
}
```

In the above example,

- grid-template-columns: repeat(4, 1fr) creates a four-column grid layout with equal-width columns
- grid-auto-rows: auto implicitly adds the rows of required size, until all grid items are placed
- grid-auto-flow: dense fits the grid content in all possible grid cells, without leaving any gaps between the grid items
- grid-row: span 2 / auto spans the specified element across two rows, adapting the row size as needed
- grid-column: span 2 / auto spans the specified element across two columns, adapting the column size as needed.

Now, let's add the following styles for tablet devices.

```
@media screen and (max-width: 768px) {
  .gallery {
      grid-template-columns: 1fr 1fr;
   }
}
```

Finally, the following style changes the 2-column grid layout to a single-column layout for mobile devices.

```
@media screen and (max-width: 480px) {

/* changes the grid layout to a single column */
   div.gallery {
      grid-template-columns: 1fr;
   }

/* resets the grid placement properties for
the images spanning four grid cells */
   img.four-grid-cells {
```

```
        grid-row: auto;
        grid-column: auto;
    }

/* resets the grid placement properties for
the images spanning two grid columns  */
    img.wide-image {
        grid-column: auto;
    }

}
```