CSS Notes

Introduction

CSS (**Cascading Style Sheets**) is a stylesheet language used to describe the presentation of a web page written in HTML. It controls the layout, colors, fonts, spacing, and overall design of a website, making it visually appealing and user-friendly.

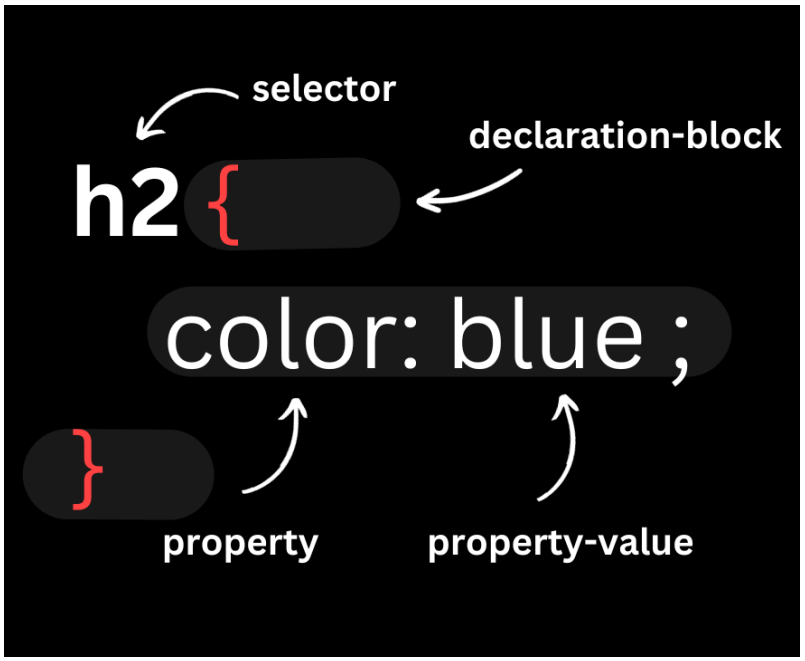
What is CSS?

- CSS stands for **Cascading Stylesheet**.
- CSS is a stylesheet language used to describe the presentation of an HTML document.
- CSS describes how elements should be rendered on screen.

Why Use CSS?

CSS defines styles of your web pages including design, layout and variations in display across devices and screen sizes.
CSS was **first proposed in 1994**, by Norwegian technologist **Håkon Wium Lie**.

CSS Syntax: -



A **CSS rule set / rule** is composed of a selector and a declaration block.

CSS Selectors

Element Selector : It is the simplest way to target all elements of a given type.

```
h1{
    color: chartreuse;
}
```

Class Selectors : Multiple elements can have the same class values.CSs class selectors are very useful when targeting different elements.The class selector consists of a dot (.), A class name is defined using the class attribute.

```
.my-Class{
    color: chartreuse;
}
```

An HTML element can have multiple class names separated by a white space.

```
<div class="navbar font-color"></div>
```

ID Selectors : The ID selector is very useful when targeting a single element. An ID selector consists of a has (#) followed by an ID Name.An ID name is defined using the id attribute.

```
#btn{
    color: chartreuse;
}
```

Grouping Selector

The group selector in CSS allows you to apply the same styles to multiple elements at once by separating them with a comma (,).

```
h1,p{
    color: brown;
    background-color: bisque;
}
```

Comments

Comments can be used to explain CSS codes and make it more readable. They are not rendered by browsers.

```
/* I am applying css on h1 and p element */
h1,p{
    color: brown;
    /* background-color: purple; */
}
/* The purple background color will not be
applied because its commented */
```

CSS Inserting

There are three simple ways to insert CSS into an HTML Document : -

- **Inline Styling**
- **Internal Style Sheet**
- **External Style Sheet**

Inline Styling : is useful for styling a single element. It is done by using the style attribute. Its value contains CSS declarations.

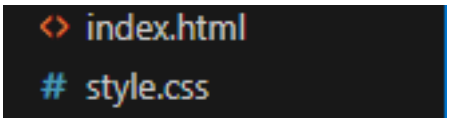
```
<h1 style="color: brown;font-family: cursive;">I am Learning CSS</h1>
```

Internal Style Sheet : An internal style sheet is useful for styling a single HTML document. CSS rules should be placed inside the `<style>` element, which must be within the `<head>` section.

```
<head>
    <style>
        h1{
            color: lime;
            background-color: burlywood;
        }
    </style>
</head>
```

External Style Sheet : An external style sheet is useful for applying the same styles to multiple HTML pages. It is saved with a `.css` extension (e.g., `filename.css`) and included in HTML using the `<link>` element. A CSS file should not contain a `<style>` element.

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



Color property : The `color` property in CSS is used to set the text color of an element.

```
h1{
    color: lime;
}
```

CSS colors are commonly specified using the following :-

- **Color Name**
- **Hexadecimal Color Value**
- **RGB Color Value**

Background property : The `background` property in CSS is used to set the background of an element. It allows you to define colors, images, positions, and more.

Background properties : -

- **background-color**
- **background-image**
- **background-repeat**
- **background-attachment**
- **background-position**
- **background-clip**

Background color property : we can specify a background color for an element using the background-color property.

```
.navbar{
    background-color: teal;
}
```

Background image property : The background-image property sets one or more images as the background of an element. The format of its value should be: url("image.jpg"). Single quotes and no quotes also work e.g. url('image.jpg') and url(image.jpg) .The text contained in the quotes are file paths.

```
.marvel-movie{
    background-image: url('Avenger.png');
}
```

Background repeat property : CSS automatically resets background images horizontally and vertically. To only repeat the background image horizontally or vertically we can use the background-repeat property.

```
.marvel-movie{
    background-repeat: no-repeat;
    background-repeat: repeat-x;
    background-repeat: repeat-y;
    background-repeat: round;
    background-repeat: space;
}
```

Background position property : we can set the initial position of a background image.

Valid Values :

- **Top, bottom, left, right, center**
- **Pixels**
- **Percentage**
- **And more.**

```
.marvel-movie{
    background-position: top right;
    background-position: 200px 400px;
    background-position: 20% 50%;
}
```

Background attachment property : this property sets whether a background image position is fixed within the viewport, or scroll with its containing block.

Valid values : -

- **scroll : Default value. The background image scrolls with the page.**
- **fixed : The background image stays fixed while the page scrolls.**
-

```
.marvel-movie{
    background-attachment: scroll;
    background-attachment: fixed;
}
```

Background shorthand property : we can specify all CSS background properties in one single property using its shorthand property.

The background property is a shorthand for the following CSS properties

- **background-color**
- **background-image**
- **background-repeat**
- **background-attachment**
- **background-position**

```
.marvel-movie{
    background:purple url('avengers') no-repeat fixed top center;
}
```

Width property : The `width` property in CSS is used to define the width of an element.

```
.navbar {
    width: 100px;
}
```

Height property : The `height` property in CSS is used to define the height of an element.

```
.navbar {
    height: 100px;
}
```

max-width & max-height : The `max-width` and `max-height` properties in CSS define the maximum width and height an element can have. They prevent an element from growing beyond a certain size while still allowing flexibility.

```
.navbar {
    max-width: 300px;
    max-height: 300px;
}
```

min-width & min-height : In CSS, `min-width` and `min-height` are used to specify the minimum width and height that an element can have.

CSS Border

Border Style : the border-style property sets the linestyle for all four sides of an element's border.

Valid values : -

- **none :** displays no border
- **hidden :** displays no border
- **dotted :** displays a series of rounded dots
- **dashed :** displays a series of short square-ended dashes or line segments
- **solid :** displays a single, straight, solid line
- **double :** displays two straight lines • groove : displays a border with a carved appearance • ridge: displays a border with an extruded appearance
- **inset :** displays a border that makes the element appear embedded
- **outset :** displays a border that makes the element appear embossed

```
.my-box{
    border-style: dashed;
    border-style: dotted;
    border-style: double;
    border-style: groove;
    border-style: hidden;
    border-style: inset;
    border-style: outset;
    border-style: ridge;
    border-style: solid;
    border-style: none;
}
```

Border Width : we can specify the width of an element's borders using the border-width CSS property.

Valid values :-

- **Thick**
- **Medium**
- **Thin**

```
.my-box {
    border-width: 10px;
}
```

Border Color : the border-color CSS property defines the color of a border.

```
.my-box {
    border-color: brown;
}
```

Border Shorthand Property : The border CSS property sets an element's border. It's a shorthand for the following CSS Properties.

```
.my-box {
    border: 2px solid black;
}
```

Border Radius : The `border-radius` property is used to create rounded corners for an element.

```
.my-box {
    border-radius: 10px;
}
```

Margin property : Margin property creates space around or outside an element.

Margin Individual Sides: -

- **margin-top :** sets the margin area on top of an element.
- **margin-right :** sets the margin area on the right of an element.
- **margin-bottom :** sets the margin area on the bottom of an element.
- **margin-left :** sets the margin area on the left of an element.

Margin Shorthand Property

- When one value is specified, it applies the same margin to all four sides.
- When two values are specified, the first value applies to the top and bottom , the second to the left and right.
- When three values are specified, the first value applies to the top, the second to the left and right, the third to the bottom.
- When four values are specified, the margins apply to the top, right, bottom, and left in that order (clockwise) respectively.

```css
/* one-value */
#div1 {
  margin: 20px;
}
/* two values */
#div2 {
  margin: 40px 20px;
}
/* three values */
#div3 {
  margin: 20px 40px 20px;
}
/* four values */
#div4 {
  margin: 40px 20px 40px 20px;
}
```

Horizontal Center an Element

Padding property : CSS padding creates space within an element.
It clears an area around the inside of an element.

Padding Individual Sides: -

- **padding-top :** sets the padding area on top of an element.
- **padding-right :** sets the padding area on the right of an element.
- **padding-bottom :** sets the padding area on the bottom of an element.
- **padding-left :** sets the padding area on the left of an element.

Padding Shorthand Property

- When one value is specified, it applies the same padding to all four sides.
- When two values are specified, the first value applies to the top and bottom , the second to the left and right.
- When three values are specified, the first value applies to the top, the second to the left and right, the third to the bottom.

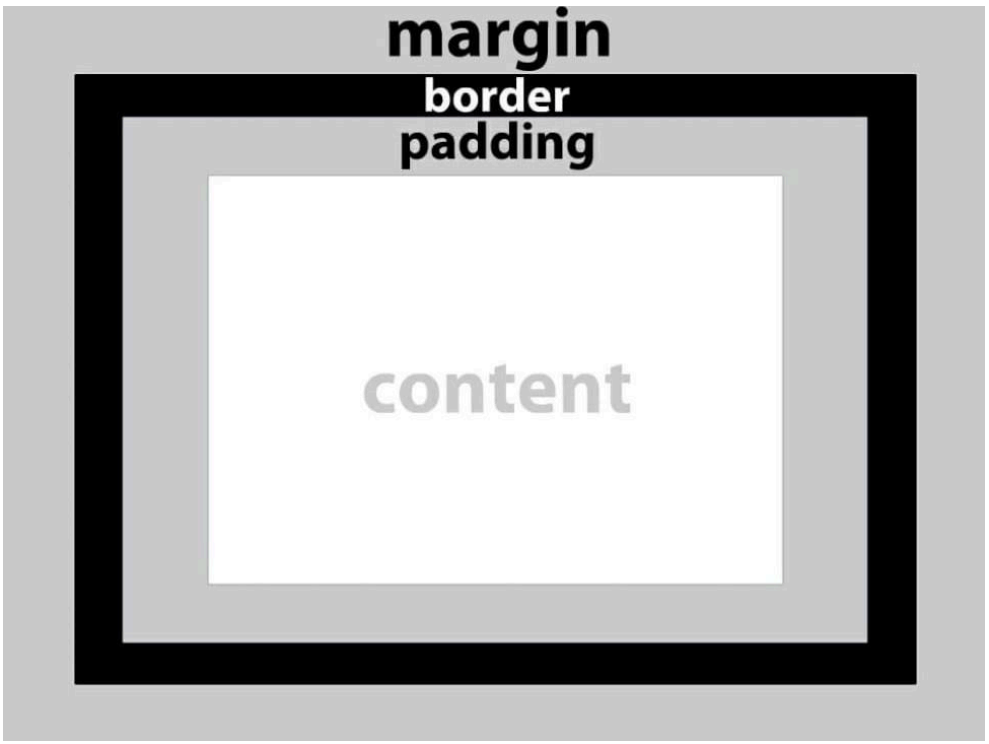
- When four values are specified, the padding applies to the top, right, bottom, and left in that order (clockwise) respectively.

```css
/* one value */
#element1 {
  padding: 20px;
}
/* two values */
#element2 {
  padding: 10px 20px;
}
/* three values */
#element3 {
  padding: 20px 10px 20px;
}
/* four values */
#element4 {
  padding: 10px 20px 30px 40px;
}
```

Note : By default, when you apply `padding` to an element, it increases its total size beyond the defined `width` and `height`, unless the `box-sizing` property is set to `border-box`.

Box Model : The **CSS Box Model** is the fundamental concept that defines how elements are structured and displayed on a webpage. Every element in CSS is treated as a rectangular box, consisting of **four parts**:

- **Content:** also called content box/area; it is the area where the content of the box is displayed.
- **Padding**: the space or area between the outer edge of the content box and the inner edge of the border; it is transparent.
- **Border:** the area between margin and the padding; its width, style and color can be changed.
- **Margin:** the area or space outside the border or outside the CSS box; it is transparent .



To get the total width or height of a CSS Box we have to use the following formulas.

CSS Box's width = left border's width + left padding's width + content box's width + right padding's width + right border's width.

CSS Box's height = top border's height + top padding's height + content box's height + bottom padding's height + bottom border's height.

CSS Text

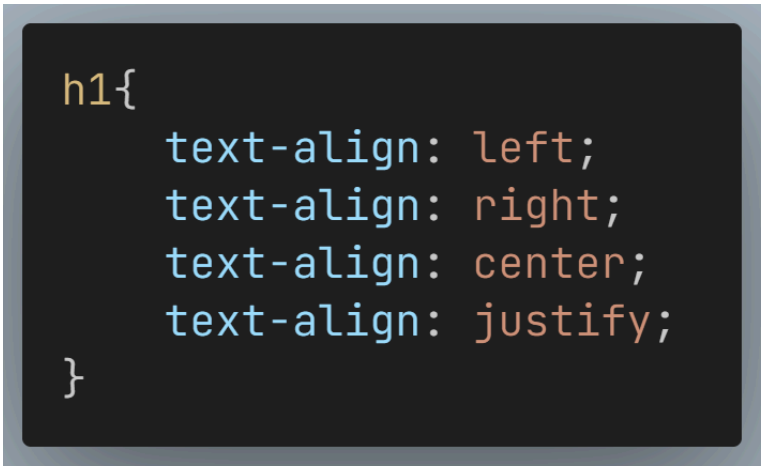
Color property : the color CSS property sets the color of a text.

```
h1{
    color: steelblue;
}
```

Text Align property : the text-align CSS property sets the horizontal alignment of a text.

Valid values : -

- **left :** aligns text to the left edge.
- **center :** aligns text to the center.
- **right :** aligns text to the right edge.
- **justify :** aligns text its left and right edges to the left and right edges of the line box except for the last line.



```
h1{
    text-align: left;
    text-align: right;
    text-align: center;
    text-align: justify;
}
```

Text Transform property : The text-transform CSS property sets the capitalization of the text.

Valid Values :-

- **none :** prevents the case of the text from being changed
- **capitalize :** converts the first letter of each word to uppercase; other characters are unchanged
- **uppercase :** converts a text to uppercase
- **lowercase :** converts a text to lowercase

```css
h1{
    text-transform: capitalize;
    text-transform: lowercase;
    text-transform: uppercase;
    text-transform: none;
}
```

Text Indent property : The text-indent CSS property sets the length of the indentation of a text.

```css
p{
    text-indent: 30px;
}
```

Text Decoration property : The text-decoration CSS property sets the text decoration line, color and style. The text-decoration CSS property is actually a shorthand for text-decoration-line, text-decoration-style and text-decoration-color properties.

Text-decoration-line valid values : -

- **none :** provides no decorative line
- **underline :** provides a decorative line beneath a text
- **overline :** provides a decorative line above a text
- **line-through :** provides a decorative line going through the middle of a text

Text-decoration-style valid values : -

- **solid :** draws a solid line
- **dotted :** draws a dotted line
- **dashed :** draws a dashed line
- **double :** draws a double line
- **wavy :** draws a wavy Line

Text-decoration-color valid values : -

- **<colorName>**

```
p{
    text-decoration-line: underline;
    text-decoration-style: solid;
    text-decoration-color: cadetblue;
}
```

Text Decoration shorthand property
text-decoration: <text-decoration-line> <text-decoration-style> <text-decoration-color> <text-decoration-thickness>;

Very Helpful : if we want to remove underline from the anchor tag, we can use text-decoration:none; to remove the decoration.

Vertical align property : The `vertical-align` CSS property controls the vertical positioning of inline elements, table cells, or inline-block elements relative to their surrounding content.

Usage

- **Works on inline elements, inline-block elements, and table cells.**
- **Does not affect block-level elements.**

```
.vertical {
    vertical-align: top;
    vertical-align: middle;
    vertical-align: bottom;
}
```

Text Shadow property : The text-shadow CSS property adds shadows to text. Each shadow is described by some combination of X and Y offsets from the element, blur radius, and color.

Valid Values :

- **<color> :** optional; sets the color of the shadow; if this is not defined the value is left up to the user-agent
- **offset-x offset-y :** required value; it specifies the shadow's distance from the text; offset-x is the horizontal distance while offset-y is the vertical distance
- **blur-radius :** optional; defaults to 0; this is a <length> value, it defines the size of the blur/shadow

```
p{
    text-shadow: 2px 2px 3px grey;
}
```

Letter Spacing property : The letter-spacing CSS property sets the length and behaviour of the space between letters.

Valid Values :-

- **Normal :** sets the normal letter spacing for the current font.
- **<length>**

```
p{
    letter-spacing: 10px;
}
```

Word Spacing property : The word-spacing CSS property sets the length of the space between words.

Valid Values : -

- **normal :** sets the normal word spacing as defined by the browser
- **<length>**
- **<percentage>**

```
p{
    word-spacing: 10px;
}
```

Line Height property : The line-height property specifies the amount of space used in lines such as in text.

Valid Values :-

- **normal :** depends on the user agent
- **<number> :** the value is multiplied by the element's current font size; this is the preferred may to set line-height
- **<length>**
- **<percentage>**

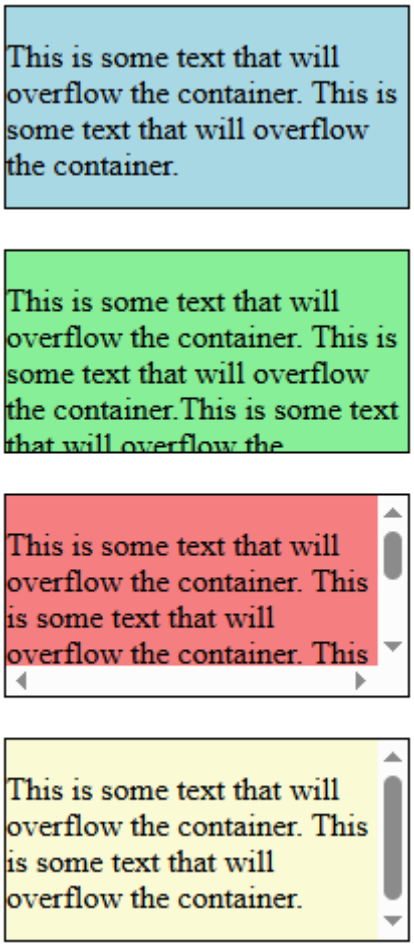
```
p{
    line-height: 10px;
}
```

Overflow property : The `overflow` property in CSS controls how content is displayed when it overflows its container. It is mainly used for handling scrollbars and content clipping.

Valid Values : -

- visible (default)
- hidden
- scroll
- auto
- clip

```
.box-1 {
    overflow: visible; /* Default value */
    overflow-x: hidden; /* Hide horizontal overflow */
    overflow-y: scroll; /* Always show vertical scrollbar */
    overflow: auto; /* Scrollbars appear when needed */
    overflow: hidden; /* Hide overflow */
    overflow: scroll; /* Always show scrollbars */
}
```



White Space property : The white-space CSS property sets how the white space is handled inside and element .

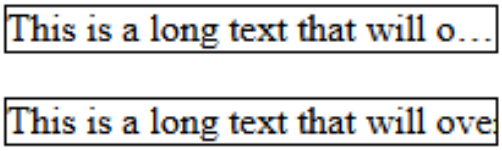
Valid Values : -

- **normal :** Sequences of white space are collapsed. Newline characters in the source are handled the same as other white spaces. Lines are broken as necessary to fill line boxes.
- **nowrap :** Collapses white space as for normal, but suppresses line breaks (text wrapping) within the source.
- **pre :** Sequences of white space are preserved. Lines are only broken at newline characters in the source and at
 elements.
- **pre-wrap :** Sequences of white space are preserved. Lines are broken at newline characters, at
, and as necessary to fill line boxes.

- **pre-line :** Sequences of white space are collapsed. Lines are broken at newline characters, at
 , and as necessary to fill line boxes.

Text-overflow property : The `text-overflow` CSS property controls how text is displayed when it overflows its container. It is commonly used with `white-space: nowrap` and `overflow: hidden` to create text truncation effects.

```
.text-overflow-clip {
    width: 200px;
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
    text-overflow: clip;
    border: 1px solid #000;
}
```



Text Direction property : The direction CSS property sets the direction of a text (and more).

Valid Values : -

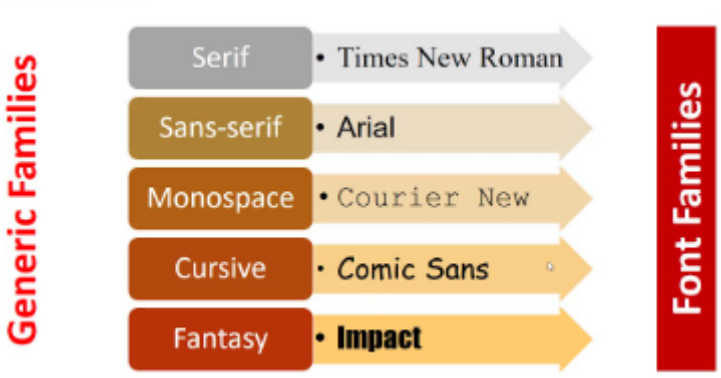
- **ltr :** sets text and other elements go from left to right.
- **rtl :** sets text and other elements go from right to left.

```
p{
    direction: ltl;
    direction: rtl;
}
```

CSS Fonts
CSS fonts define the font family, size, style, weight or boldness and variant of a text.

Font Family property : There are two types of font family names in CSS

- <generic-name> : a group of font families which have similar looks eg. San-serif, Monospace.
- <family-name> : a specific font family name e.g. Lucida Console, Verdana.




```
p{
    font-family: monospace;
}
```

Let’s learn how to add Google fonts

1. Go to https://fonts.google.com/ Website.
2. Choose a font.
3. Click Embedded and copy import or <link> and Paste in Head or external CSS.
4. Copy the font family name and paste it in CSS where you want to apply it.

Font size property : The `font-size` property in CSS is used to specify the size of the text. It helps control how large or small the text appears on a webpage

Note : Font size is em highly recommended

Font style property : The `font-style` property in CSS is used to specify the style of the text, such as normal, italic, or oblique.

```css
.para{
    font-style: italic;
    font-style: oblique;
    font-style: normal;
}
```

Font weight property : The `font-weight` property in CSS is used to control the thickness (boldness) of the text.

```css
.para{
    font-weight: 100;
    font-weight: 900;
}
```

Font variant property : The `font-variant` property in CSS is used to control the display of small-caps text and other typographic variations.

```css
.para{
    font-variant: normal;
    font-variant: small-caps;
}
```

Icons : we can add icons to our HTML page by using icons libraries such as font awesome, Bootstrap icons and Material icons by google.

Link : in CSS we can style links by changing their properties (color, background-color, text-decoration etc.)

Link Defaults :

- Links are underlined
- Unvisited links are blue
- Visited links are purple
- Hovering a link makes the mouse a little hand icon
- Focused links have an outline around them
- Activate links are red

Link States (pseudo classes)

1. **Link (unvisited)**
2. **Visited**
3. **Hover**
4. **Focus**
5. **Activate**

List-style-type : property sets the type or marker to use.

List-style-position : sets the position (whether its outside or inside) of the bullets.

Valid Values:

- **Outside**
- **Inside**

List-style-image : sets an image as bullets.

Table border collapsing
The border-collapse:collapse; css declaration collapses table borders into one.

Outline : an outline is a line drawn an element,outside the border(**If there is a border).**

Outline-style : property sets the style of an element's outline.
Outline-width : property sets the width of an element's outline.
Outline-color : property sets the color of an element's outline.

Opacity : The opacity property in CSS is used to control the transparency of an element.with values ranging from 0 (completely transparent) to 1 (fully visible).

Buttons : Create buttons of different types.

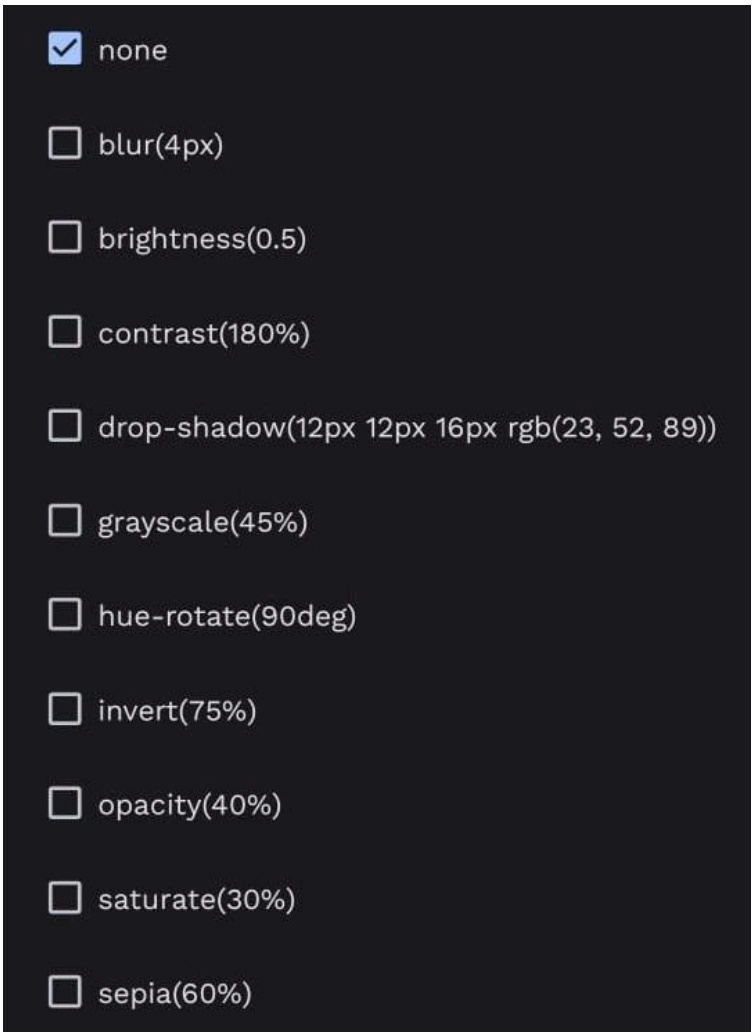
Images

Responsive Image

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

Create a beautiful image card.

Image filters : property applies graphical effects like blur or color shift to an element.



CSS Layout

Display : property specifies how an element should be displayed.

Valid values :-

- Inline
- Block
- Inline-block
- None
-

Hiding an Element

1) Set the display CSS property to none.

2) Set visibility CSS property to hidden.

Note : Understand the different between display none and visibility hidden.

Width and max-width properties : to prevent from horizontal scroll bar

```
div {
  width: 600px;
  background: gold;
}
```

```
div {
  max-width: 600px;
  background: gold;
}
```

Horizontally centering with margin auto

```
.div1 {
  margin: 0 auto; /* same as margin-top: 0;
  width: 270px;
  height: 300px;
  background: pink;
}
.div2 {
  margin: 0 auto; /* same as margin-top: 0;
  max-width: 600px;
  height: 300px;
  background: green;
}
```

Positioning : property sets how an element is positioned.

Valid values :

- **Static**
- **Relative**
- **Fixed**
- **Absolute**
- **sticky**

The top, right, bottom , left properties set the location of positioned elements.

Static position : by default, HTML elements are positioned static. The top, right, bottom , left properties don't affect static positioned elements.

```
div {
  position: static;
  width: 200px;
  height: 200px;
  background: #903C56;
  color: white;
}
```

Relative position : with position relative the element is positioned according to the normal flow of the document.The top, right, bottom , left properties specify the offset relative to itself.it will adjust from its normal position.Unlike the margin properties, it does not push against and affect other elements.

```
div.relative {
  position: relative;
  width: 100px;
  height: 100px;
  left: 25px;
  top: 25px;
  background: #903C56;
  color: white;
}
```

Fixed position : with position fixed the element is removed from the normal document flow, and no space is created for the element in the page layout.
It makes an element be positioned relative to the viewport, the element stays in its location even when scrolling.
The top, right, bottom , left properties specify the distance between top, right, bottom left edge of the element

```
div {
  position: fixed;
  width: 50px;
  height: 50px;
  top: 10px;
  right: 10px;
  background: #173459;
}
```

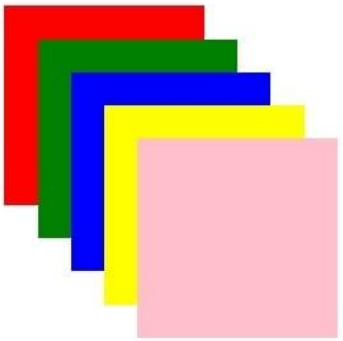
Absolute position : With position: absolute , the element is removed from the normal document flow, and no space is created for the element in the page layout. It makes an element be positioned relative to its container (nearest parent/ancestor element).
The container should be a positioned element. It is recommended to use the relative value as it is positioned according to the normal flow of the page. Note! an element with position: static is not a "positioned element".
The top , right , bottom and left CSS properties specify the distance between the top , right, bottom and left edges of the element and the top, right, bottom and left edge of its container accordingly.

```
div.container {
  position: relative;
  width: 300px;
  height: 300px;
  background: #78969D;
}
div.absolute {
  position: absolute;
  bottom: 10px;
  right: 10px;
  width: 50px;
  height: 50px;
  background: #f8f9f9;
}
```

Position sticky: With position: sticky, the element is positioned according to the normal flow of the document. It offsets the element to nearest ancestor scrolling element. The top , right , bottom and left CSS properties specify the offset to the top, right, bottom and left edge of the nearest ancestor scrolling element.

```
div {
  position: sticky;
  position: -webkit-sticky; /* for Safari *
  top: 0px;
  left: 0px;
  width: 100%;
  color: white;
  padding: 5px;
  font-size: 18px;
}
```

Stacking Context : The stacking context is a three-dimensional conceptualization of HTML elements along an imaginary z-axis relative to the user.
The fixed and absolute position always creates a new stacking context.
The z-index CSS property specifies the position of an element along the z-axis. It accepts <integer> values.
The higher the value, the closer the element to the user.



```
.div1 {
  z-index: 1;
  top: 10px;
  left: 10px;
  background: red;
}
```

CSS units : CSS has several different units for expressing a length.

Absolute Lengths
The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.

Unit	Description
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px *	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)

Relative Lengths

Relative length units specify a length relative to another length property. Relative length units scale better between different rendering medium.

Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to the width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element

Overflow property : Content usually overflows when it is bigger than its container (parent element).
The overflow-x CSS property sets what to do when content overflows a block-level element's left and right edges.
The overflow-y CSS property sets what to do when content overflows a block-level element's top and bottom edges.
Valid Values:

- visible : default; content is not clipped, overflowing content is rendered outside the box's edges
- hidden : content is clipped if necessary and invisible
- scroll: content is clipped if necessary to fit the padding box; browsers will display a scroll bar
- auto : if content fits inside the padding box it works the same as visible ; if content overflows the the padding box it works like scroll

```
.div1 {
  overflow-x: visible;
}
.div2 {
  overflow-x: hidden;
}
.div3 {
  overflow-x: scroll;
}
.div4 {
  overflow-x: auto;
}
```

```
.div1 {
  overflow-y: visible;
}
.div2 {
  overflow-y: hidden;
}
.div3 {
  overflow-y: scroll;
}
.div4 {
  overflow-y: auto;
}
```

The overflow CSS property is a shorthand for the overflow-x and overflow-y CSS properties.
It can have one or two values:

- When one value is specified, the value applies to both overflow-x and overflow-y properties
- When two values are specified, the first value applies to the overflow-x property while the second value applies to the overflow-y property

Float property : The float CSS property places an element on the left or right side of its container (containing block).
It allows inline elements to wrap around it.

Valid Values:

- none: default; the element does not float
- left: the element floats on the left side of its container
- right: the element floats on the left side of its container

Clear property : The clear CSS property sets whether an element can have floating elements around it.It can be applied to floating and non-floating elements.
Valid Values:

- none: default; elements are allowed to float on both sides
- left: elements are not allowed to float on the left side
- right: elements are not allowed to float on the right side
- both: elements are not allowed to float on both sides

Inline-block : unlike the inline value the inline-block value allows us to set an element's width and height.

```
span.inline-block-element {
  display: inline-block;
  width: 100px;
  height: 100px;
  background: lightgrey;
  border: 1px solid red;
}
```

Horizontally Centering Text

```
.center {
  text-align: center;
  width: 100%;
  border: 3px solid #173459;
}
```

Block level elements centering

```
.center {
  margin: 0 auto; /* same as margin-top: 0;
  width: 60%;
  padding: 20px;
  border: 3px solid #173459;
}
```

Horizontally centering an image

```
img {
  display: block;
  margin: 0 auto; /* same as margin-top: 0;
  width: 60%;
  height: auto;
}
```

Left and right align text

```
.left {
  text-align: left;
}
.right {
  text-align: right;
}
p {
  background: lightgrey;
}
```

Left and right align element

```
.right {
  position: absolute;
  right: 0; /* use left property to align to
  width: 60%;
  padding: 10px;
  border: 3px solid #173459;
}
```

Vertically centering using padding

```
div {
  padding-top: 40px;
  padding-bottom: 40px;
  border: 3px solid #173459;
}
```

Top and bottom align elements

```
.bottom {
  position: absolute;
  bottom: 0; /* use bottom property to alig
  width: 60%;
  padding: 10px;
  border: 3px solid #173459;
}
```

Both horizontally and vertically centering a text

```
div {
  width: 250px;
  line-height: 250px;
  text-align: center;
  font-size: 20px;
  background: LightBlue;
}
```

Both horizontally and vertically centering using position

```
.center {
  position: relative;
  height: 500px;
  border: 3px solid #173459;
}
.center > p {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
```

Combinators : A combinator is something that explains the relationship between the selectors.

CSS combinators :

- Descendent combinators (वंशज) : The descendant combinator matches all elements that are descendants of a specified element.

```
div p {
    background-color: yellow;
}
```

- Child Combinators : The child combinator selects all elements that are the children of a specified element.

```
div > p {
    background-color: yellow;
}
```

- Adjacent sibling combinators : The adjacent sibling combinator (+) is used to select an element that comes immediately after another specific element in the same parent.

```
div + p {
    background-color: yellow;
}
```

- General sibling combinator : The general sibling combinator (~) is used to select all sibling elements that come after a specified element, even if they are not immediately next to it.

```
div ~ p {
    background-color: yellow;
}
```

Pseudo class : a pseudo class is a keyword added to a selector that specifies a special state for the targeted elements.

```
selector:pseudo-class {
  property: value;
}
```

[Pseudo class cheat sheet]

Pseudo elements : a pseudo element is a keyword added to a selector that lets you style a specific part of the selected elements.

```
selector::pseudo-element {
  property: value;
}
```

[Pseudo element cheat sheet]

Attribute selector : selector target elements based on the presence or value of a given attribute.

Attribute selectors cheat sheet

CSS 3 : is the latest and current standard of CSS.

Rounded Corners : the border-radius property rounds the corners of an element's outer border edge.

Note : border-radius CSS property does not need the border properties to work.

Individual rounded corners

- border-top-left-radius
- border-top-right-radius
- border-bottom-right-radius
- border-bottom-left-radius

Border image : the border-image property is a shorthand for the following

- border-image-source : sets the source image used to create an element's border image. Valid value: <url>
- border-image-slice: divides the image specified by border-image-source into regions which form the components of an element's border image.
- border-image-width : sets the width of an element's border image.
- border-image-outset : sets the distance by which an element's border image is set out from its border box.
- border-image-repeat : defines how the edge regions of a source image are adjusted to fit the dimensions of an element's border image.

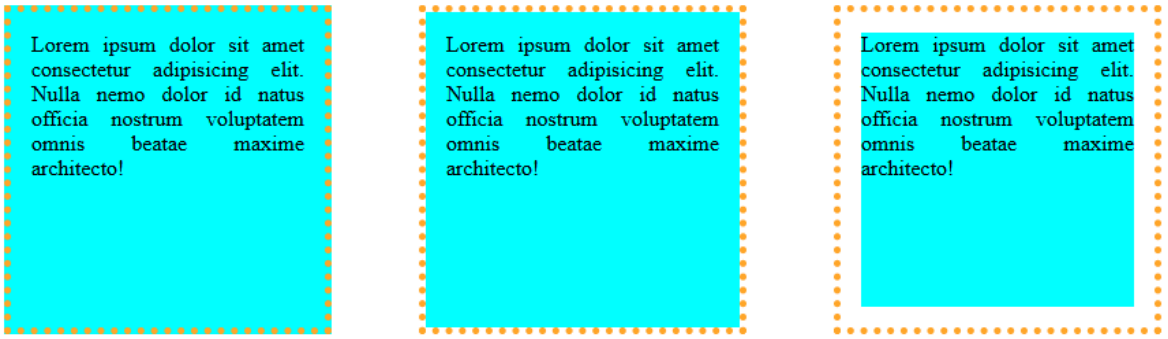
Background

Background clip property : The background-clip property defines how far the background (color or image) should extend within an element.

Valid values : -

- border-box (default)
- padding-box
- content-box

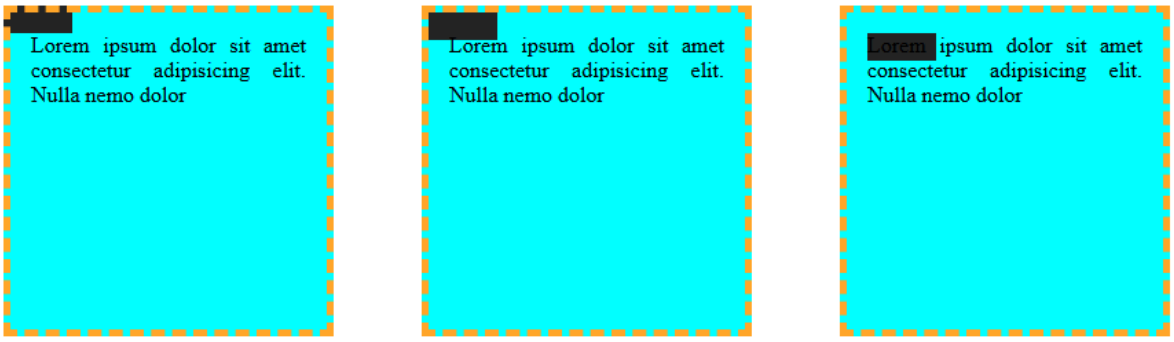
```
div{
    background-clip: border-box;
    background-clip: padding-box;
    background-clip: content-box;
}
```



Background origin property : The background-origin property specifies the origin position (the background positioning area) of a background image.

Valid values : -

- **border-box**
- **padding-box**
- **content-box**




```
div{
    background-origin: border-box;
    background-origin: padding-box;
    background-origin: content-box;
}
```

Background size property : allows us to change the size of a background using the background-size property.

Valid values : -

- cover
- contain
- <length>
- <percentage>

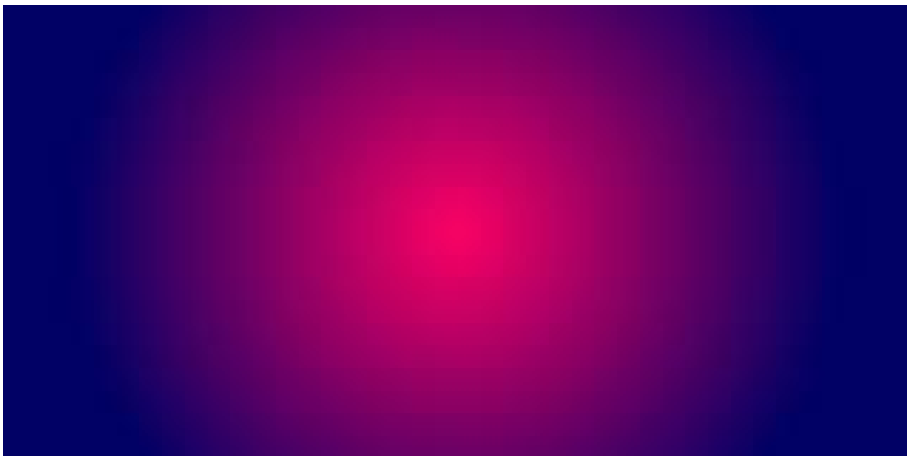
Gradients : CSS gradients are made of a progressive transition between two or more colors.
The transition between colors are usually smooth.

There are two types of gradients :-

- Linear gradients
- Radial gradients

Linear gradients : the linear-gradients() CSS function creates an image consisting of a progressive transition between two or more along a straight line.
Linear gradient - top to bottom (default)

Radial gradient : the radial-gradient() CSS function creates an image consisting of a progressive transition between two or more colors that radiate from an origin.
By default, shape is ellipse, size is farthest corner, and position is center.



2D transforms : the transform CSS property allows us to translate, rotate, scale and skew elements.
A CSS transformation is an effect that changes an element's size, shape and position.

2D Transform methods

- translate()
- translateX()
- translateY()
- rotate()
- scale()
- scaleX()
- scaleY()
- skew()
- skewX()
- skeY()
- matrix()

Translate : translate function reposition or move an element from its original position in the horizontal and / or vertical directions.

It is specified with two values, the first value represents the horizontal direction while the second value represents the vertical directions.

```
div {
  width: 200px;
  height: 200px;
  background: lightgrey;
}
.moved {
  transform: translate(30px, 40px);
  background: #173459;
}
```

translateX() : translateX() function repositions or moves an element from its original position in the horizontal directions.

```
div {
  width: 200px;
  height: 200px;
  background: lightgrey;
}
.moved {
  transform: translateX(30px);
  background: #173459;
}
```

translateY() : translateY() function repositions or moves an element from its original position in the vertical directions.

```
div {
  width: 200px;
  height: 200px;
  background: lightgrey;
}
.moved {
  transform: translateY(30px);
  background: #173459;
}
```

rotate() : rotate() function defines a transformation that rotates an element without deforming it.
The value should be an <angle>, if the value is positive the movement is clockwise and if the value is negative the movement is counter-clockwise.

```
.clockwise {
  transform: rotate(45deg);
  background: #173459;
}
.counter-clockwise {
  transform: rotate(-90deg);
  background: #903C56;
}
```

scale() : scale() function defines a transformation that resizes an element on the 2D plane.
The amount of scaling is defined by a vector, it can resize the horizontal and vertical dimensions at different scales.
The scale() function is specified with either one or two <number> values

- When one value is specified, the value applies to both horizontal and vertical dimensions.
- When two values are specified the first value applies to the horizontal dimension while the second to the vertical.

```
.scaled {
  transform: scale(.5);
}
.scaled-2 {
  transform: scale(2);
}
```

scaleX() : the scaleX() function defines a transformation that resizes an element on the 2D plane.
The amount of scaling is defined by a vector, it can resize the horizontal dimensions at different scales.

```
.scaled {
  transform: scaleX(.5);
}
```

scaleY() : the scaleY() function defines a transformation that resizes an element on the 2D plane.
The amount of scaling is defined by a vector, it can resize the vertical dimensions at different scales.

```
.scaled {
  transform: scaleY(.5);
}
```

skew() : skew() function defines a transformation that skews() an element on the 2D plane.
It is defined as either one or two values. The values should be <angle> , the first value skews the element along the X-axis the second value skews the element along the Y-axis.

```
.skewed {
  transform: skew(30deg, 30deg);
  color: #f8f9f9;
}
.skewed-1 {
  transform: skew(-30deg, -30deg);
  color: #f8f9f9;
}
```

skewX() : skewX() function defines a transformation that skews an element on the 2D plane along the x-axis.
It is defined with an <angle> value.

```
.skewed {
  transform: skewX(45deg);
  color: #f8f9f9;
}
.skewed-1 {
  transform: skewX(-45deg);
  color: #f8f9f9;
}
```

skewY() : skewY() function defines a transformation that skews an element on the 2D plane along the y-axis.
It is defined with an <angle> value.

```
.skewed {
  transform: skewY(45deg);
  color: #f8f9f9;
}
.skewed-1 {
  transform: skewY(-45deg);
  color: #f8f9f9;
}
```

matrix() : matrix() function defines a homogeneous 2D transformation matrix.
It is defined with six <number> values, each separated by a comma.
The first , second, third and fourth values describe the linear translation.
The fifth and six values describe the translation to apply.

```
.transformed {
  transform: matrix(1, 2, -1, 1, 80, 80);
}
```

Using 2D transforms with hover

```
div {
  width: 100px;
  height: 100px;
  background: #173459;
  color: #f8f9f9;
  border: 5px solid #903C56;
  transition-duration: 2s;
}
#translate:hover {
  transform: translate(30px, 40px);
}
#rotate:hover {
  transform: rotate(-90deg);
}
#scale:hover {
  transform: scale(.5);
}
#skew:hover {
  transform: skew(30deg, 30deg);
}
```

3D transform : a transformation is an effect that changes an element size, shape and position.

3D transform methods :

- rotateX()
- rotateY()
- rotateZ()

rotateX() : rotateX() function defines a transformation that rotates an element around the abscissa (x-axis or horizontal axis) without deforming it.
The value should be an <angle>

```
.rotated {
  transform: rotateX(140deg);
}
div {
  color: white;
  background: black;
  width: 150px;
  height: 150px;
}
```

rotateY() : rotateY() function defines a transformation that rotates an element around the abscissa (y-axis or vertical axis) without deforming it.
The value should be an <angle>

```
.rotated {
  transform: rotateY(140deg);
}
div {
  color: white;
  background: black;
  width: 150px;
  height: 150px;
}
```

rotateZ() : rotateZ() function defines a transformation that rotates an element around the z-axis without deforming it.

```
.rotated {
  transform: rotateZ(90deg);
}
div {
  color: white;
  background: black;
  width: 150px;
  height: 150px;
}
```

3D transforms with hover

```
div {
  width: 100px;
  height: 100px;
  background: #173459;
  color: #f8f9f9;
  border: 5px solid #903C56;
  transition-duration: 2s;
}
#rotateX:hover {
  transform: rotateX(140deg);
}
#rotateY:hover {
  transform: rotateY(140deg);
}
#rotateZ:hover {
  transform: rotateZ(90deg);
}
```

Transition : transition allows us to control animation speed when changing CSS properties.
CSS 3 transition can cause the change in a defined property to take place over a period of time, instead of just taking place immediately.

```
.without-transition {
  width: 100px;
  height: 100px;
  background: black;
}
.with-transition {
  width: 100px;
  height: 100px;
  background: black;
  transition-property: width;
  transition-duration: 2s;
  transition-delay: .5s;
}
div:hover {
  width: 200px;
}
```

Transition and duration property

The **transition-property** CSS property specifies the name of the property the transition effect is for.

The **transition-duration** CSS property sets how much time a transition should take to complete. By default the value is 0s, meaning no animation will occur.

```
div {
  width: 100px;
  height: 100px;
  background: #903C56;
  transition-property: height;
  transition-duration: 2s;
}
div:hover {
  height: 300px;
}
```

We can also have multiple transition properties by specifying multiple properties and separating each with a comma.

```
div {
  width: 100px;
  height: 100px;
  background: #903C56;
  transition-property: width, height,
  transition-duration: 2s;
}
div:hover {
  width: 200px;
  height: 200px;
  background: #173459;
}
```

Transition-timing-function : the transition-timing-function CSS property sets the timing calculation of the transition effect.

Valid Values : -

- Ease : default; the transition effect start slowly, then fast,then slowly ends.
- Linear : the transition effect has the same speed from start to end.
- Ease-in : the transition effect starts slowly.
- Ease-out : the transition effect ends slowly.
- Ease-in-out : the transition effect both starts and ends slowly.

```
div {
  transition-timing-function: ease;
  transition-property: width, background-
  transition-duration: 2s;
  width: 100px;
  height: 100px;
  background: blue;
}
div:hover {
  width: 200px;
  background: red;
}
```

Transition-delay : the transition-delay CSS property specifies a duration to wait before the transition effect starts.

```
div {
  width: 100px;
  height: 100px;
  background: #903C56;
  transition-property: height;
  transition-duration: 2s;
  transition-delay: 1s;
}
div:hover {
  height: 300px;
}
```

Transition shorthand

- **transition-property**
- **transition-duration**
- **transition-timing-function**
- **transition-delay**

```
.transition-4 {
  transition: margin-top 2s .5s ease-in-out;
}
```

Animation : the CSS animation HTML element transitions from one style to another.
With CSS animation we can animate elements without having to use (or know) javascript.

Animation have two components :-

- A style describing the CSS animation.
- A set of keyframes that indicate the start, intermediate waypoints and end states of the animation's style.

The @keyframes rule

To configure the actual appearance of the animation we need to use the @keyframes at-rule.
The @keyframes at-rule defines keyframes or waypoints along the animation sequence which controls the intermediate steps in a CSS animation.
Each @keyframes rule contains a style list of keyframe selectors, which specify percentages along the animation when the keyframe occurs, and a block containing the styles for that keyframe. 0% indicates the starting state of the animation sequence, while 100% indicates the final state of the animation. Because 0% and 100% are very important, they have special aliases: from and to.
Aside from from /0% and to/100%, we can optionally include additional keyframes (e.g. 25%, 33%, 50%, 66%, 75%) that describe intermediate steps between the start and end of the animation.
To use keyframes, create a @keyframes rule with a name that is then used by the animation-name property to match an animation to its keyframe declaration. We also need to specify the duration of the animation using the animation-duration CSS property.

```css
@keyframes myAnimation {
  from {
    background-color: gold;
    top: 0px;
  }
  to {
    background-color: green;
    top: 200px;
  }
}

div {
  animation-name: myAnimation;
  animation-duration: 2s;
  position: relative;
  width: 100px;
  height: 100px;
  background-color: gold;
}
```

```css
@keyframes myAnimation {
  1% {
    background-color: gold;
    top: 0px;
    left: 0px;
  }
  25% {
    background-color: green;
    top: 200px;
    left: 200px;
  }
  50% {
    background-color: gold;
    top: 0px;
    left: 200px;
  }
  75% {
    background-color: green;
    top: 200px;
    left: 0px;
  }
  100% {
    background-color: gold;
    top: 0px;
    left: 0px;
  }
}

diy {
}

div {
  animation-name: myAnimation;
  animation-duration: 2s;
  position: relative;
  width: 100px;
  height: 100px;
  background-color: gold;
}
```

Animation-timing-function : The animation-timing-function CSS property sets how an animation progresses through the duration of each cycle.

Valid Values:

- ease: default; the animation effect starts slowly, then fast, then slowly ends
- linear: the animation effect has the same speed from start to end
- ease-in : the animation effect starts slowly
- ease-out: the animation effect ends slowly
- ease-in-out: the animation effect both starts and ends slowly

```css
div {
  animation-name: myAnimation;
  animation-duration: 2s;
  animation-timing-function: ease;
  position: relative;
  width: 100px;
  height: 100px;
  background-color: gold;
}
```

Animation delay : the animation-delay CSS property specifies a duration to wait before the transition effect starts.

```css
@keyframes myAnimation {
  from {
    background-color: gold;
    top: 0px;
    left: 0px;
  }
  to {
    background-color: green;
    top: 200px;
    left: 200px;
  }
}

div {
  animation-name: myAnimation;
  animation-duration: 2s;
  animation-delay: 1s;
  position: relative;
  width: 100px;
  height: 100px;
  background-color: gold;
}
```

Repeating animations : We can repeat a CSS animation using the animation-iteration-count CSS property.
It sets the number of times an animation cycle should be played. It can take a <number> value or the keyword infinite, meaning the animation will be repeated indefinitely.

```
div {
  animation-name: myAnimation;
  animation-duration: 2s;
  /* try to change the value below with a nu
  animation-iteration-count: infinite;
  position: relative;
  width: 100px;
  height: 100px;
  background-color: gold;
}
```

Animation direction : The animation-direction CSS property sets whether an animation should play forwards, backwards, or alternating back and forth.
Valid Values:

- normal: default; the animation plays forwards each cycle
- reverse: the animation plays backwards each cycle
- alternate: the animation reverses direction each cycle, with the first iteration being played forwards
- alternate-reverse: the animation reverses direction each cycle, with the first iteration being played backwards

```
div {
  animation-name: myAnimation;
  animation-duration: 2s;
  animation-iteration-count: infinite;
  animation-direction: normal;
  position: relative;
  width: 100px;
  height: 100px;
  background-color: gold;
}
```

Animation fill mode : The animation-fill-mode CSS property sets how a CSS animation applies styles to its target before and after its execution.

Valid Values:

- none: default; the animation will not apply styles to the element while it's not executing
- forwards: the element will retain the styles set by the last keyframe
- backwards: the element will retain the styles set by the last keyframe
- Both : the element will retain the styles set by the first and last keyframes

```
p {
  animation-name: myAnimation;
  animation-duration: 2s;
  animation-fill-mode: none;
}
```

Animation play state : The animation-play-state CSS property defines whether an animation is running or paused.

Valid Values:

- running: default; the animation is playing
- paused: the animation is paused

```
@keyframes running {
  from {
    top: 0px;
  }
  to {
    top: 500px;
  }
}

.running {
  animation-name: running;
  animation-duration: 5s;
  animation-play-state: running; /* default
  background: #173459;
}
.paused {
  animation-name: paused;
  animation-duration: 5s;
  animation-play-state: paused;
}
div {
  width: 150px;
  height: 100px;
  position: relative;
  background: black;
  color: white;
}
```

Animation shorthand

- Animation-name
- Animation-duration
- Animation-timing-function
- Animation-delay
- Animation-iteration-count
- Animation-direction
- Animation-fill-mode
- Animation-play-state

Recommendation : always use the individual sub properties.

Note : animation-name and animation-duration properties are required.

Resize property : the resize property sets whether an element is resizable.

Valid Values:

- none: the element does not offer any user-controllable method for resizing it
- horizontal: the element provides a control allowing it to be resized horizontally
- vertical: the element provides a control allowing it to be resized vertically
- both: the element provides a control allowing it to be resized both horizontally and vertically

By default, elements are not resizeable, but we can make it become resizeable by a user.

```
.horizontal {
  resize: horizontal;
}
.vertical {
  resize: vertical;
}
.both {
  resize: both;
}
div {
  padding: 50px;
  border: 1px solid black;
}
```

By default the textarea element is resizable but we can make it not resizable, but we can make it not resizable if we need to.

```
textarea {
  resize: none;
}
```

Box sizing : The box-sizing CSS property sets how the total width and height of an element is calculated.

Valid Values:

- content-box: gives the default box-sizing behavior where the widths of the padding and border are added to the size of an element
- border-box: the defined width and height include any border and padding, it causes the content-box to shrink

by default, the padding and border widths add up to the total render width and height of an element.

```
.content-box {
  box-sizing: content-box;
}
.border-box {
  box-sizing: border-box;
}
div {
  width: 200px;
  height: 200px;
  padding: 20px;
  border: 10px solid black;
  background: yellow;
}
```

Most developers prefer to apply box-sizing:border-box to universal selector.

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

Flex property

Media queries : CSS3 Media Queries are useful when you want to modify your web page depending on a device's general type or specific characteristics and parameters.

For instance, you can set distinct styles for device's with different:

- width and height of the viewport
- orientation (i.e. portrait or landscape in mobile devices)
- screen resolution

A media query is composed of an optional media type and any number of media feature expressions.
It can have multiple expressions combined using logical operators.
Media queries are case-insensitive.
A media query computes true when the media type matches the device. When that happens, all corresponding style sheets or rules will be applied.

CSS media type : CSS media types describe the category of a device.

They are as follows:

- all: suitable for all devices
- print: intended for paged material and documents viewed on a screen in print preview mode
- screen: intended primarily for screens
- speech: intended for speech synthesizers

CSS Media Features

CSS Media Features describe specific characteristics of the user agent, output device,or environment.

Media feature expressions test for their presence or value, and are entirely optional. Each media feature expression must be surrounded by parentheses.

The @media Rule : the @media at-rule is used to target media types and media features.

```
@media not|only mediatype and (expressions)
  css-rules;
}
```

The expressions resolve to either true or false.

If true, it means that the expressions match the media type of the device the web page is being displayed on.

When that is the case then all the corresponding style sheet (e.g. CSS declaration blocks) are applied to the document.

The media type is optional except when using the not or only operators. If they are not used, the all media type will be implied.

Example:

This example automatically changes the background color of the document depending on the media features of the device.

Specifically, when the screen is 500 pixels wide or or wider the background color turns green.

```
@media screen and (min-width: 500px) {
  body {
    background: green;
  }
}
```

Responsive web design

Responsive Web Design (RWD) is an approach to web design that ensures a website looks and functions well on a variety of devices and screen sizes, from desktops to tablets and smartphones. It uses flexible layouts, fluid grids, and media queries to automatically adjust the website's content and design based on the screen size and resolution.

Responsive web design viewport

First, what is a viewport?

Viewport is the area of the window in which web contents can be seen. Or in simplest terms, it is the visible area of a webpage that a user can see.

The viewport size depends on the device of the user. In mobile phones the viewport size is much smaller compared to laptops or desktops which are much wider.

Before, web pages were not mobile-optimized making them look bad in mobile phones or tablets since they are only designed for computer screens.

Additionally, web pages before are defined with fixed sizes, for instance, a web page can have wrappers or containers with a width of 1000px which causes web pages to look bad on narrow screen devices.

Viewport meta tag

The **viewport meta tag** is an HTML tag that helps control how a web page is displayed on mobile devices. It ensures that the page scales correctly to fit different screen sizes, improving usability and responsiveness.

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

Without viewport meta tag

With viewport meta tag