DAY 6

We can use CSS in three-way external internal and inline; we can use it in different scenarios; we should know where to use them

Inline : is used for one element

Internal : is used when you target single web page

External : That target website for all pages

CSS Selectors

When we write element of css ex : h1{} it mean it will apply to all header 1 all of them in website.

We can use class and id attribute to select specific element in css. Id # and class . when we select in css.

Class selector can be used by many element while id is can be used by one element .

We can select attributes also in css by use p[draggable]{} this way [] in here we can select our css attributes.

Example:

<p draggable = false>The king of the wrold</p>

P[draggable=”false”]{

Color:red;

}

\* selector is unvirsel selector it select all the element in css

This is example we have done

ol {

  margin-left: -40px;

  margin-top: -20px;

  list-style-position: inside;

}

/\* Write your CSS below, don't change the rules above. \*/

p{

  color: red;

}

.note{

  font-size: 20px;

}

#id-selector-demo{

  color: green;

}

li[value = "4"]{

  color: blue;

}

\*{

  text-align: center;

}

Text-align : The text-align property in CSS is used to specify the horizontal alignment of text within an element. It determines how the text content is aligned relative to its containing element (typically a block-level element like a <div>, <p>, or <section>).

1: left: Aligns the text to the left of the container (default behavior for left-to-right text).

p {

text-align: left;

}

2: Right: Aligns the text to the right of the container.

3: Justifies the text, meaning the text will spread out to fill the entire width of the container, creating an even alignment on both sides.

4: center

    <li class="note" value="3">Class selectors target elements based on the value of the class attribute.</li>

I learn here value where it can change the number in list

width: 200px;

  height: 200px;

This will have us help to choice the width and height of container

Note: One main thing I learn in these today lesson is always choize The class attribute same but id different and every element in your html they should have both of them so we can access both of them.

Day 7

Font properties

font-size: it shows the size of the letter 20px, which means 1 px means pixel 1/96th inch. Another way you represent. they are  to represent it 1 point,  1 point, 1/72nd inch

1px  1/96th inch

1 pt. 1/72nd inch

1em 100% of parent

1rem 100% of the root relative to the root will be affected only if we change the root element. and rem is more consistent in use

1em = 20px; relative to the body

Day 8

Each element is a  box in HTML

Width: is horizontally expanding in CSS

Height: vertical

border : 10px solid color; it can take three values

we have border-top,border-left, and more

we can use border-width: left right  bottom top

Padding: put space but element and Broder

margin: that is outside of the border

use the control key to see where your border is

In CSS, the margin property is used to create space around elements, outside of their borders. When you use four values for the margin property, they specify the margins for the element in the following order (clockwise):

1. **Top margin**: The first value sets the margin at the top of the element.
2. **Right margin**: The second value sets the margin on the right side.
3. **Bottom margin**: The third value sets the margin at the bottom.
4. **Left margin**: The fourth value sets the margin on the left side.

margin: 10px 15px 20px 25px;

### Explanation:

* 10px → Top margin
* 15px → Right margin
* 20px → Bottom margin
* 25px → Left margin

### Quick Tip:

* If all four values are the same, you can use just one value: margin: 10px; applies 10px to all sides.
* If only two values are provided, the first value is applied to the top and bottom, and the second value to the left and right: margin: 10px 20px;.
* If three values are provided, the first is for the top, the second for left and right, and the third for the bottom: margin: 10px 20px 30px;.

Text-transform choice uppercase and lowercase

Day 8 :

**Cascading in CSS** refers to the method by which CSS rules are applied and resolved when multiple rules target the same element. The term "cascading" describes how styles are applied in a specific order, based on a set of rules and priorities. The cascade determines which CSS rule is ultimately applied to an element when there are conflicts.

### ****1. Inline Styles**** (Highest Priority)

* Styles applied directly to an HTML element using the style attribute.

<p style="color: red;">This text will be red.</p>

### ****3. Specificity****

When multiple rules are applied without !important, the specificity of the selector determines the precedence:

1. Inline styles (e.g., style="color: red;"): Highest specificity.
2. ID selectors (e.g., #id): More specific than class, pseudo-class, or attribute selectors.
3. Class, attribute, and pseudo-class selectors (e.g., .class, [attr=value], :hover): Moderate specificity.
4. Type selectors (e.g., div, p) and pseudo-elements (e.g., ::before): Lowest specificity.

/\* Low specificity \*/ p { color: blue; }

/\* Higher specificity (class selector) \*/

.highlight {

color: green;

} /\*

Even higher specificity (ID selector) \*/ #special {

color: red;

}

### ****Source Order****

* If two rules have the same specificity and none are marked with !important, the rule that appears later in the stylesheet takes precedence.

p {

color: blue;

}

p {

color: green;

}

The paragraph will be **green** because the second rule appears later.

### ****Summary of Cascading Importance Levels****

1. !important overrides all other rules.
2. Inline styles take precedence over internal and external stylesheets.
3. Specificity resolves conflicts between selectors.
4. Source order breaks ties when specificity is the same.
5. Author styles override user styles, which override browser default styles.

By understanding this hierarchy, you can effectively control which CSS rules are applied and ensure consistent styling.

Cascading : happen in four : 1: position lower is stronger in higher levels 2: specifity id 3: type external , internal inline 4: importance

**Combining CSS Selectors**

Combining CSS selectors allows you to target elements more precisely by combining multiple selectors in various ways. This enables you to write efficient and specific CSS rules. Here's how you can combine selectors:

### ****1. Descendant Selector (**** ****)****

* Targets elements that are nested inside another element (any level of nesting).
* **Syntax**: parent child

div p {

color: blue;

}

### ****2. Child Selector (****>****)****

* Targets only the direct children of an element.
* **Syntax**: parent > child

### ****3. Adjacent Sibling Selector (****+****)****

* Targets an element that immediately follows another element.
* **Syntax**: element1 + element2

h1 + p {

font-size: 14px;

}

### ****4. General Sibling Selector (****~****)****

* Targets all siblings of an element that share the same parent and come after the specified element.
* **Syntax**: element1 ~ element2

h1 ~ p {

color: green;

}

This will select all <p> elements that come after an <h1> within the same parent.

### ****. Grouping Selector (****,****)****

* Targets multiple elements and applies the same styles to them.
* **Syntax**: selector1, selector2

h1, h2, h3 {

font-weight: bold;

}

* This will make all <h1>, <h2>, and <h3> elements bold.

### ****6. Combining Class, ID, and Type Selectors****

* You can combine different types of selectors for more specificity.

div#container .highlight {

background-color: yellow;

}

This will select any element with the class highlight inside a <div> with the ID container.

### ****7. Pseudo-classes and Pseudo-elements****

* Combine selectors with pseudo-classes (:hover, :nth-child) or pseudo-elements (::before, ::after) for specific behaviors or effects.

a:hover {

color: red;

} p:first-child {

font-weight: bold;

}

### ****. Attribute Selectors****

* Combine selectors with attributes for even more specific targeting.

input[type="text"][disabled] {

background-color: gray;

}

This will select all disabled text input fields.

### ****9. Universal Selector (****\*****)****

* Targets all elements, often combined with other selectors for specific purposes.

div \* {

margin: 0;

padding: 0;

}

This will reset the margin and padding for all elements inside <div>.

### ****Example: Combining Techniques****

div#main > ul.menu li:first-child a {

color: blue;

}

This rule targets the first <li> inside a <ul> with the class menu, which is a direct child of a <div> with the ID main, and styles the <a> tag inside it.

**combines selected in order to select specific element**

There are different rules to combine in CSS the first rule is the group rule

1 : group :

Selector, selector {

} IT applies the same style to the entire group anything you put in gourp it does not matter what is inside it can be class or id attributes.

2: Child

child

parent > child {

color: firebrick;

} Apply to a direct child  one level inside this one it should be one level not two

ex: div --- parent

         div --- child

it will change the child's  style

div > p{

    color: firebrick;

}

.flag > div > div{

}

3 : Descendant:  Apply to a descendent of left-side class

we want to go deep and select element  change the styles to work like this

.box li {

color: blue;

}

rightmost will be selected

4: Chaining selected: Apply where all selectors are true

there is no space between selectors when we want to be specific

h1#title. Big. heading{

} no space also there should be. For classes  it could be # for  id  also the element should be first otherwise it could be confessing  Always start with the element

5: combining combiners :

ul   p.done{

    font-size: 10px;

}

Position in css

The position property in CSS specifies how an element is positioned in the document and determines how its top, right, bottom, and left properties affect its placement. There are several values for the position property:

### 1. ****Static (Default)****

* **Definition**: The default value. The element is positioned according to the normal flow of the document.
* **Behavior**: Top, right, bottom, and left properties have no effect.

### 2. ****Relative****

* **Definition**: The element is positioned relative to its normal position.
* **Behavior**: The top, right, bottom, and left properties move the element relative to where it would have been in the normal flow

### 3. ****Absolute****

* **Definition**: The element is positioned relative to its nearest positioned ancestor (an ancestor with position: relative, absolute, or fixed). If no such ancestor exists, it is positioned relative to the <html> (root element).
* **Behavior**: Removed from the normal document flow.

### 4. ****Fixed****

* **Definition**: The element is positioned relative to the viewport, which means it stays in the same place even when the page is scrolled.
* **Behavior**: Removed from the normal document flow.

### 5. ****Sticky****

* **Definition**: The element toggles between relative and fixed, depending on the user's scroll position.
* **Behavior**: The element is treated as relative until it reaches a specified offset, then it sticks in place (like fixed) until the containing block's boundary is crossed.

position: sticky; top: 10px; /\* Sticks the element 10px below the top of the viewport \*/

### Key Notes:

* **Z-index**: Use the z-index property to control the stack order of positioned elements.
* **Overflow Issues**: Positioned elements can sometimes overflow their containers, so ensure you account for scrollbars or overflow settings.

position in CSS;

Static positioning is default positioning  and it is HTML default flow in the left and top does not do anything

position relative to the default position  it moves it to a new position where we can use left and right

Absolute positioning: position relative to the nearest  positioned ancestor or top left corner of the webpage it should be the nearest ancestor or it opts left corner webpage and eight will work top also in also we will use z-index: -1 to make box  back  pust it behind

position fixed:  scroll bar will not work

Note: if I want to make position absloute the parent should be relative

When you set an element's position to absolute, it is positioned relative to the nearest **positioned ancestor** (an ancestor with position set to relative, absolute, or fixed). If no such ancestor exists, the element is positioned relative to the initial containing block, which is typically the <html> or <body>.

By setting the parent element's position to relative, you establish it as the reference point for the child with position: absolute. This is a common technique to control the positioning of absolutely positioned elements within a specific container.

Also we can use inspect to make it center that video you watch

Go to inspect select copy selector to select the specfic element new tool to use

CSS DISPLAY

In HTML, the display property is a **CSS property** that defines how an element is displayed on the web page. It controls the layout behavior of an element, such as whether it appears inline, as a block, or is hidden. This property is essential for controlling the structure and design of a webpage.

### Common Values of display:

block:

* 1. The element starts on a new line and takes up the full width available.
  2. Examples: <div>, <p>, <h1> by default.

inline:

* The element does not start on a new line and only takes up as much width as necessary.
* Examples: <span>, <a>, <strong> by default.

inline-block:

* Combines characteristics of block and inline. The element flows inline but allows block-level properties (like setting width and height).

none:

* The element is not displayed and does not occupy any space on the page.

Flex:

* Turns an element into a flexible container to layout children using the flexbox model

grid:

* Turns an element into a grid container for arranging children in rows and columns.

table:

* Makes the element behave like a table.

### Importance of the display Property:

* It determines the rendering and layout of elements.
* It plays a critical role in building responsive and user-friendly interfaces.
* It is essential for using modern layout techniques like **flexbox** and **grid**.

Display:

Inline: it will put the element in line   we can not change the height and width

block: it will take the entire width and another element will go below it not in the same line  we can change the height and width

none: it will help us when we want the element to disappear

inline: block

n CSS, the float property is used to position elements horizontally, allowing text or other content to flow around it. It is commonly used for creating layouts, although newer techniques like **Flexbox** and **CSS Grid** have largely replaced it in modern web design.

### alues of float:

none **(default)**:

* 1. The element does not float and stays in the normal document flow.

left:

* The element floats to the left of its container, and other content wraps around it on the right.

right:

* The element floats to the right of its container, and other content wraps around it on the left.

inherit:

* The float value is inherited from the parent element.

### Clearing Floats:

When using float, the element is removed from the normal flow, which can affect the layout. To fix this, you often need to "clear" floats.

**Using** clear **Property**:

* 1. Prevents elements from wrapping around floated elements.
  2. Values: left, right, both, none.

### Limitations of float:

* **Outdated for layouts**: It was widely used for page layouts in the past, but modern techniques like **Flexbox** and **Grid** are more versatile and easier to use.
* **Clearing required**: Floated elements need additional techniques like clearing to avoid breaking layouts.

float: it  will wrap the text around its parent

we can use clear properties also to clear the float

USE float only to wrap the text

Creating a responsive website ensures that your content looks great and functions properly on all devices, from desktops to tablets and mobile phones. Here's a step-by-step guide:

## ****1. Use a Responsive Meta Tag****

Include the following <meta> tag in the <head> section of your HTML to set the viewport for responsive behavior:

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

## ****2. Adopt a Fluid Layout****

Use percentage-based widths or flexible grid systems to make your layout adapt to different screen sizes.

### Example:

html

CopyEdit

<div style="width: 80%; margin: auto; background: lightblue; padding: 10px;">

This container adjusts to 80% of the screen width.</div>

## ****3. Use Media Queries****

CSS media queries apply styles based on the device's screen size.

### Example:

css

CopyEdit

/\* Default styles for large screens \*/body {

font-size: 18px;

background-color: lightgray;

}

/\* For tablets (width 768px or less) \*/@media (max-width: 768px) {

body {

font-size: 16px;

background-color: lightblue;

}

}

/\* For mobile phones (width 480px or less) \*/@media (max-width: 480px) {

body {

font-size: 14px;

background-color: lightgreen;

}

}

## ****4. Use Flexible Images and Videos****

Ensure media elements resize appropriately within their containers.

### Example:

css

CopyEdit

img, video {

max-width: 100%;

height: auto;

}

## ****5. Utilize Responsive Frameworks****

Frameworks like **Bootstrap**, **Foundation**, or **Tailwind CSS** provide pre-built classes for responsive design.

### Example with Bootstrap:

html

CopyEdit

<div class="container">

<div class="row">

<div class="col-md-6 col-sm-12">Column 1</div>

<div class="col-md-6 col-sm-12">Column 2</div>

</div></div>

## ****6. Use Flexbox or Grid Layout****

Modern CSS layout techniques like Flexbox and Grid make responsiveness easier.

### Flexbox Example:

css

CopyEdit

.container {

display: flex;

flex-wrap: wrap;

gap: 10px;

}.item {

flex: 1 1 calc(33.33% - 10px); /\* 3 columns \*/

background: lightcoral;

}@media (max-width: 768px) {

.item {

flex: 1 1 calc(50% - 10px); /\* 2 columns \*/

}

}@media (max-width: 480px) {

.item {

flex: 1 1 100%; /\* 1 column \*/

}

}

### Grid Example:

css

CopyEdit

.container {

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 10px;

}.item {

background: lightseagreen;

}@media (max-width: 768px) {

.container {

grid-template-columns: repeat(2, 1fr); /\* 2 columns \*/

}

}@media (max-width: 480px) {

.container {

grid-template-columns: 1fr; /\* 1 column \*/

}

}

## ****7. Test Responsiveness****

Test your website across different devices and screen sizes:

* Use browser developer tools (e.g., Chrome DevTools) to simulate various devices.
* Test on actual devices whenever possible.

## ****8. Optimize Typography and Buttons****

* Use relative units like em, rem, or % for font sizes.
* Ensure buttons and links are large enough to be easily tapped on touchscreens.

### Example:

css

CopyEdit

button {

padding: 10px 20px;

font-size: 1.2rem; /\* Relative unit \*/

}

## ****9. Avoid Fixed Dimensions****

Avoid using fixed widths and heights for containers, as they can break layouts on smaller screens.

## ****10. Use Scalable Navigation****

Implement responsive navigation, like a collapsible menu for smaller screens.

### Example with JavaScript:

html

CopyEdit

<nav>

<button id="toggleMenu">☰ Menu</button>

<ul id="menu" style="display: none;">

<li><a href="#">Home</a></li>

<li><a href="#">About</a></li>

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

</ul></nav><script>

document.getElementById('toggleMenu').addEventListener('click', () => {

const menu = document.getElementById('menu');

menu.style.display = menu.style.display === 'none' ? 'block' : 'none';

});</script>

**BootStrap**

we should use Bootstrap we want our product mobile to build fast when we should not use it when we want  complete control of the design of our  custom website

There are four ways to have a responsive website :

Media queries css grid css flexbox External frameworks e.g Bootstrap

### ****What is a Media Query in CSS?****

A media query is a CSS technique used to apply styles based on the characteristics of the user's device, such as screen size, resolution, orientation, or aspect ratio. It helps in creating **responsive designs** that adapt to different devices like desktops, tablets, and mobile phones.

@media media-type and (condition) {

/\* CSS rules \*/

}

### ****Common Media Query Features****

1. min-width: Styles apply when the viewport is **at least** a certain width.
2. max-width: Styles apply when the viewport is **at most** a certain width.
3. min-height **/** max-height: Based on the viewport height.
4. orientation: Detects whether the device is in **portrait** or **landscape** mode.
5. aspect-ratio: Matches based on the width-to-height ratio of the viewport.

/\* For devices with width 768px or less (e.g., tablets and phones) \*/

@media (max-width: 768px) {

body {

background-color: lightblue;

}

}

/\* For devices with width 1024px or more (e.g., desktops) \*/

@media (min-width: 1024px) {

body {

background-color: lightgreen;

}

}

/\* For portrait orientation \*/

@media (orientation: portrait) {

body {

font-size: 14px;

}

}

/\* For landscape orientation \*/

@media (orientation: landscape) {

body {

font-size: 18px;

}

}

4. Using Multiple Conditions

/\* Apply styles for screens between 600px and 1200px \*/

@media (min-width: 600px) and (max-width: 1200px) {

body {

background-color: lightcoral;

}

}

5. Aspect Ratio

/\* For devices with an aspect ratio of 16:9 or wider \*/

@media (min-aspect-ratio: 16/9) {

body {

background-color: lightgray;

}

}

@ media query :

Max-width: means anything that is smaller or equal to the size it should have a similar font or something else

@media (min-width: 600px) and (max-width: 900px){

}

we use flex for layout  and it will be 100% width

Display: flex;

gap:10px;

 .container {

      padding: 10px;

      background-color: gold;

      display: flex;

      gap: 10px;

    }

Flex direction:

By default: flex-direction: row; until there is no space

flex-direction: column;

The inline-flex property in CSS is a value of the display property that allows an element to behave like an inline element while maintaining the flex container properties. It combines the characteristics of inline and flex.

### ****Key Features of**** inline-flex****:****

1. **Inline Behavior**: The element remains inline, meaning it does not start on a new line and only takes up as much space as necessary.
2. **Flex Container**: It enables the child elements to be flex items, meaning they can be arranged using flexbox properties (justify-content, align-items, etc.).
3. **Does Not Stretch by Default**: Unlike flex, which takes up the full width of the parent container, inline-flex only takes up as much space as its content requires.

### ****When to Use**** inline-flex****?****

* When you want a flexbox container that behaves like an inline element (e.g., inside a paragraph).
* When you need to align elements within buttons, labels, or small UI components.
* When you need inline-level elements but also want to control their layout using flexbox.

flex-box layout

we have an order: where we can order the  layout

flex-wrap:nowrap;

flex-wrap: wrap; it is better if it fills the minimum width

justify-content:center: in CSS and it is very easy to horizontal center

align:item: this is the position of the item

align-content: it only works when you flex-wrap to wrap

flex-basis: it is like width

flex-growth: max-width

flex-shrink: min-width

we have a shorthand

flex:1 1 0; first is growth the second is the shrink and the 3 value is the basis

Height:100vh;

n CSS, height: 100vh; means that the element's height will be **100% of the viewport height**.

### Explanation:

* vh stands for **viewport height**.
* 100vh means **100% of the height of the visible screen** (viewport).
* If the viewport height is **800 pixels**, height: 100vh; makes the element **800 pixels tall**.

### flex: 1; in CSS

The flex property is a shorthand for flex-grow, flex-shrink, and flex-basis

### ****Breakdown of**** flex: 1;

1. flex-grow: 1; → The element will grow and take up any available space equally with other flex items that also have flex: 1;.
2. flex-shrink: 1; → The element can shrink if needed when the container resizes.
3. flex-basis: 0%; → The element starts with zero width before distributing space.

#### ****How it works:****

* Both .box elements will take **equal space** inside the .container.
* If .container has **500px width**, each .box will take **250px**.
* If you change one box to flex: 2;, it will take **twice** as much space as the others.

### ****When to Use**** flex: 1;****?****

* To make elements **equally sized** in a flex container.
* To create **responsive layouts** without fixed widths.
* To distribute space dynamically.

flex box is good  for one dimension  it is is good for a horizontal layout

but gird is good for two dimensional layouts where we want to have vertical and horsetail layout

we have size of grid  minmax(400px,800px) we can use this it is really good and effective

Also we have grid-template-columns:300px;

we can use repeat function in grid grid-template-rows: repeat(2,200px)  it will save us alot of time

Grid-auto-rows;

Grid-auto-columns;

/\* Write your CSS code below to make the purple items size, grow and shrink like the green ones.\*/

.grid-container {

display: grid;

grid-template-columns: auto 400px minmax(200px,500px);

grid-template-rows: 1fr 1fr 2fr;

Grid-auto-rows: 50px;

}

Grid also allow you to overlay to another layout

The grid-area property in CSS defines the placement of an element within a CSS Grid layout. The shorthand grid-area: 2 / 2 / 4 / 4; is interpreted as:

grid-area: row-start / column-start / row-end / column-end;

For grid-area: 2 / 2 / 4 / 4;, it means:

* 2 → The element starts at **row 2**.
* 2 → The element starts at **column 2**.
* 4 → The element ends **before row 4** (i.e., it spans from row 2 to row 3).
* 4 → The element ends **before column 4** (i.e., it spans from column 2 to column 3).

1 2 3 4 1 ─── ─── ─── ─── 2 ─── [X] [X] ─── 3 ─── [X] [X] ─── 4 ─── ─── ─── ───

The [X] marks the area occupied by the element. It spans from **row 2 to row 4** and **column 2 to column 4**, covering a **2×2 area**.

Would you like a live example with code?

  #monsterStats{

        /\*  we use display none to hide some thing in html or makeing website  \*/

        display: none;

        background-color: red;

        margin-top: 20px;

        margin-bottom: 30px;

        border: 2px solid;

        height: 25px;

       }

We use display: none; to hide something in website

bootstrap: we can check for how much size of columns we want col-2  or we want col-12

it depends on us

col-6 is 50% col-2 is 1/6

also bootstrap has a point

we write it like this col-sm-2 and we don't need too much media queries

also, we can have multiple breakpoint col-sm-12 or col-md-8

<!-- Modify the HTML code for the purple boxes below so it behaves like the green ones. -->

<div class="row"> 6 out 12 is 50% percent in bootstrap

<div class="col-xl-6 col-sm-12">50% desktop, 100% mobile</div>

<div class="col-xl-6 col-sm-12">50% desktop, 100% mobile</div>

</div>

<div class="col-xl-6 col-sm-12" >Column 1</div>

<div class="col-xl-3 col-sm-6">Column 2</div>

<div class="col-xl-3 col-sm-6 ">Column 3</div>

</div>

The text-shadow property in CSS is used to apply shadow effects to text. It adds one or more shadows behind the text, creating a visual depth or glow effect.

text-shadow: horizontal-offset vertical-offset blur-radius color;

· **horizontal-offset**: The horizontal distance of the shadow from the text. Positive values move the shadow to the right, while negative values move it to the left.

· **vertical-offset**: The vertical distance of the shadow from the text. Positive values move the shadow down, while negative values move it up.

· **blur-radius** (optional): How much the shadow should be blurred. A higher value creates a more spread-out shadow. If omitted, the shadow will be sharp.

· **color** (optional): The color of the shadow. You can use color names, RGB, RGBA, HSL, or hexadecimal values. If omitted, the default color is black.

The transition property in CSS is used to create smooth animations when CSS properties change. It allows you to control the duration, timing, and other aspects of how the property change should be animated.

transition: property duration timing-function delay;

· **property**: Specifies the CSS property (or properties) that you want to animate. It could be a specific property like background-color, width, opacity, or all to apply the transition to all properties that change.

· **duration**: The time (in seconds or milliseconds) that the transition takes to complete. For example, 2s or 500ms.

· **timing-function** (optional): Specifies the speed curve of the transition (how the transition progresses over time). Common values are:

* ease (default): Starts slow, becomes faster, and ends slow.
* linear: Moves at a constant speed.
* ease-in: Starts slow, then speeds up.
* ease-out: Starts fast, then slows down.
* ease-in-out: Starts and ends slow, but speeds up in the middle.

· **delay** (optional): The delay before the transition starts. It’s defined in seconds (s) or milliseconds (ms).

button {

background-color: blue;

color: white;

transition: background-color 0.3s ease-in-out;

}

button:hover {

background-color: green;

}

· In this example, when you hover over the button, the background color transitions from blue to green in 0.3s.

· The ease-in-out timing function makes the change smooth, starting and ending slowly but speeding up in the middle.

In CSS, :hover is a **pseudo-class** used to define the style of an element when the user interacts with it by hovering their mouse over it. This interaction is often used for creating visual feedback, such as changing the appearance of buttons, links, or other elements to indicate that they are clickable or interactive.

button {

background-color: #4CAF50; /\* Green background \*/

color: white;

padding: 10px 20px;

border: none;

cursor: pointer;

}

button:hover {

background-color: #45a049; /\* Darker green on hover \*/

}

Learn

  box-shadow: 0 0 15px #ffcc00;

It is really cool

The transform property in CSS is used to apply transformations to an element, such as moving, rotating, scaling, or skewing it. This allows you to visually change the shape and position of an element without affecting its layout in the document flow.

The value can be one of several transformation functions, each altering the element in different ways. Here are the most commonly used ones:

### Common transform functions:

translate(x, y): Moves an element along the X (horizontal) and Y (vertical) axes.

* 1. translateX(x): Moves the element along the X-axis only.
  2. translateY(y): Moves the element along the Y-axis only.

transform: translate(50px, 100px); /\* Moves the element 50px right and 100px down \*/

div { transform: rotate(45deg); /\* Rotates the element 45 degrees clockwise \*/ }

scale(x, y): Scales (resizes) the element along the X and Y axes.

* scaleX(x): Scales the element along the X-axis.
* scaleY(y): Scales the element along the Y-axis.

div { transform: scale(1.5, 2);

/\* Scales the element 1.5 times on the X-axis and 2 times on the Y-

axis \*/ }

div { transform: matrix(1, 0, 0, 1, 50, 100); /\* Applies a combination of translation, scaling, rotation \*/ }

div {

transform: translate(50px, 50px) rotate(45deg) scale(1.2);

}

div {

width: 100px; height: 100px; background-color: red; transition: transform 0.5s ease;

}

div:hover {

transform: scale(1.5) rotate(45deg);

}

### Key Points:

* The transform property allows you to change how an element looks without affecting its position in the document.
* It's commonly used for interactive effects, animations, and visual enhancements.
* transform doesn't affect the flow of other elements around it, unlike properties like top, left, or position, which move the element in the document.

In summary, the transform property is a versatile tool that can be used to manipulate elements in various ways, making it a crucial part of CSS for creating dynamic, engaging web designs.

Understanding linear-gradient(135deg, #f5f7fa, #c3cfe2);

This is a CSS linear gradient that smoothly transitions between two colors.

135deg: This defines the direction of the gradient.

0deg → Left to right

90deg → Top to bottom

135deg → Diagonal (top-left to bottom-right)

#f5f7fa (Light Grayish Blue): This is the starting color.

#c3cfe2 (Soft Blue): This is the ending color.

This creates a soft, modern, and clean gradient.

button {

background: linear-gradient(135deg, #f5f7fa, #c3cfe2);

border: none;

padding: 10px 20px;

border-radius: 8px;

font-size: 1rem;

cursor: pointer;

transition: 0.3s;

}

button:hover {

background: linear-gradient(135deg, #c3cfe2, #f5f7fa);

body { background: linear-gradient(135deg, #f5f7fa, #c3cfe2); }

A **linear gradient** is a way to create a gradual transition between two or more colors along a straight line. It is commonly used in CSS for background styling.

background: linear-gradient(direction, color1, color2, ...);

· direction: Defines the angle or side from where the gradient starts.

· color1, color2, ...: Defines the colors used in the gradient.

**Using Keywords** (to, left, right, top, bottom)

This means the gradient starts from **left (red)** and moves **right (blue)**

· **0deg** = Top to bottom.

· **90deg** = Left to right.

background: linear-gradient(to bottom, yellow, green);

background: linear-gradient(45deg, pink, purple);

background: linear-gradient(to right, rgba(255, 0, 0, 0.5), rgba(0, 0, 255, 0.5));

### ****Understanding**** box-shadow ****in CSS****

The box-shadow property is used to add shadow effects to elements, making them look **elevated** or giving a **depth effect**.

box-shadow: offsetX offsetY blurRadius spreadRadius color;

· offsetX: Moves shadow **horizontally** (left/right).

· offsetY: Moves shadow **vertically** (up/down).

· blurRadius: Softens the shadow edges.

· spreadRadius: Expands or shrinks the shadow.

· color: Defines the shadow color.

.card { box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.2); }

box-shadow: 0px 0px 15px rgba(0, 123, 255, 0.6);

### ****Understanding**** transition ****in CSS****

The transition property allows you to create **smooth animations** when CSS properties change, making your UI feel more interactive and polished.

transition: property duration timing-function delay;

· property → The CSS property to animate (e.g., background, color, transform).

· duration → How long the animation lasts (0.3s, 1s, etc.).

· timing-function → Defines how the animation progresses (ease, linear, ease-in, ease-out, etc.).

· delay (optional) → Time to wait before starting the transition

Smooth Hover Effect on a Button

transition: background 0.3s ease-in-out, transform 0.2s ease;

button:hover { background: #0056b3; transform: scale(1.1); }

transition: left 0.4s ease-in-out;

@keyframes slideIn {

    from {

        width: 0;

    }

    to {

        width: 100%;

    }

}

The CSS property background-size: 400% 400%; means that the background image is stretched to **four times** its original width and height relative to the element's background positioning area.

· The background-size property accepts **one or two values** per background layer.

· If you define multiple sizes, they should match the number of background images.

background-size: 200% 200%; /\* Width = 200%, Height = 200% \*/

background-image: url(image1.jpg), url(image2.jpg); background-size: 200% 200%, 100%;

The background-size property in CSS is used to define the size of the background image of an element. It controls how the background image is scaled and displayed within the element.

background-size: auto;

background-size: 100px 50px; /\* Width = 100px, Height = 50px \*/ background-size: 50% 50%; /\* 50% of the element’s width and height \*/

**One Value (Width only, Height auto-adjusts)**

* If only one value is given, the second is automatically set to auto (keeping aspect ratio)

cover

* The background image covers the entire element while maintaining its aspect ratio.
* Some parts of the image may be cropped.

background-size: cover;

contain

* The background image fits inside the element without cropping, maintaining its aspect ratio.

| **Property** | **Behavior** |
| --- | --- |

|  |  |
| --- | --- |
| cover | Image fills the entire area (cropped if needed). |

|  |  |
| --- | --- |
| contain | Image is fully visible (may leave empty space). |

main {

    background: rgba(255, 255, 255, 0.1);

    backdrop-filter: blur(10px);

    border-radius: 15px;

    padding: 2rem;

    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);

    border: 1px solid rgba(255, 255, 255, 0.18);

    width: 100%;

    max-width: 500px;

    animation: slideIn 1s ease-in-out;

  }

New thing I learn in css is:

resize: vertical;