**CSS**

**CSS Colors:**

**1. RGB :**

rgb(*red,* *green*, *blue*)

Example : rgb(60, 60, 60)

**2. RGBA :** The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

rgba(*red,* *green*, *blue, alpha*)

Example : rgba(255, 99, 71, 0.5)

**3. HEX :**

#RRGGBB

Example : #3cb371

**4. HSL :** specified using hue, saturation, and lightness

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

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

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

Example : hsl(0, 100%, 50%)

**5. HSLA :**

hsla(*hue,* *saturation*, *lightness, alpha*)

Example : hsla(9, 100%, 64%, 0)

**CSS background-attachment**

The background-attachment property specifies whether the background image should scroll or be fixed

background-attachment: fixed;

background-attachment: scroll;

**CSS Border Style**

* groove - Defines a 3D grooved border. The effect depends on the border-color value
* ridge - Defines a 3D ridged border. The effect depends on the border-color value
* inset - Defines a 3D inset border. The effect depends on the border-color value
* outset - Defines a 3D outset border. The effect depends on the border-color value
* double

**Outline vs Border :**

Outline : on the border

* dotted - Defines a dotted outline
* dashed - Defines a dashed outline
* solid - Defines a solid outline
* double - Defines a double outline
* groove - Defines a 3D grooved outline
* ridge - Defines a 3D ridged outline
* inset - Defines a 3D inset outline
* outset - Defines a 3D outset outline
* none - Defines no outline
* hidden - Defines a hidden outline

**CSS Table :**

**Horizontal :**

text-align: center;

text-align: center;

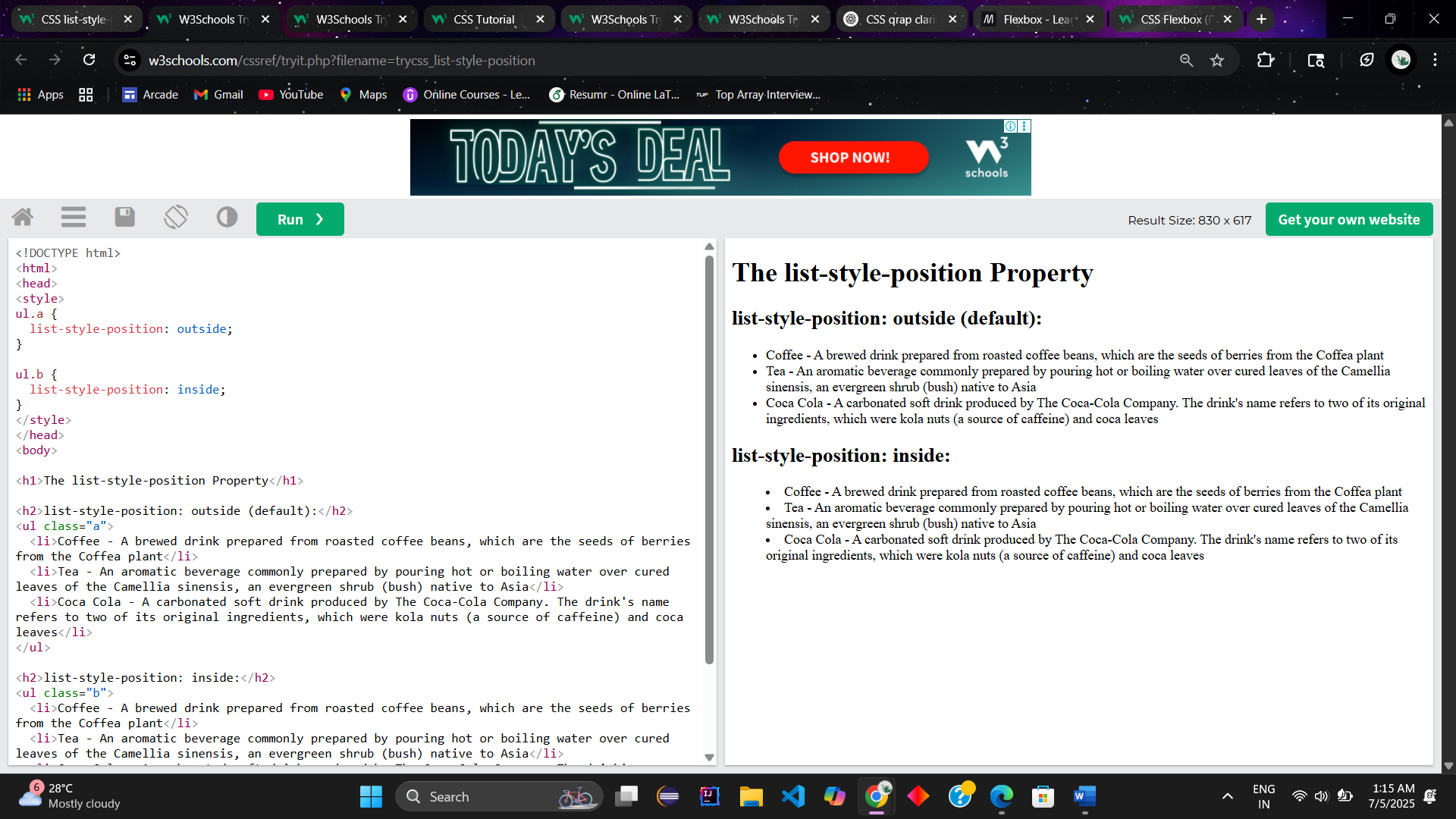
**Vertical :**

Vertical-align: bottom;

**CSS List**

list-style-position: outside;

list-style-position: inside;



**CSS Overflow**

* **visible -** Default. The overflow is not clipped. The content renders outside the element's box
* **hidden -** The overflow is clipped, and the rest of the content will be invisible
* **scroll -** The overflow is clipped, and a scrollbar is added to see the rest of the content
* **auto** - Similar to scroll, but it adds scrollbars only when necessary

**CSS Combinators**

**1. Descendant combinator (space) :** matches all elements that are descendants of a specified element.

Example :

div p {  
  background-color: yellow;  
}

**2. Child Combinator (>)** : The child combinator selects all elements that are the children of a specified element.

Example :

div > p {  
  background-color: yellow;  
}

**3. Next Sibling Combinator (+)** : next sibling combinator is used to select an element that is directly after another specific element.

Example :

div + p {  
  background-color: yellow;  
}

**CSS Psuedo Classes & Elements**

**✅ Pseudo-Classes**

**Definition:** A pseudo-class is used to define a special state of an element (like hover, focus, checked, etc.)

**🔹 Common Pseudo-Class Examples:**

| **Pseudo-Class** | **Description** | **Example** |
| --- | --- | --- |
| :hover | When mouse hovers over an element | button:hover { color: red; } |
| :focus | When an element (like input) is focused | input:focus { border: 2px solid blue; } |
| :first-child | Selects the first child of a parent | ul li:first-child { color: red; } |
| :last-child | Selects the last child of a parent | ul li:last-child { color: blue; } |
| :nth-child(n) | Selects the nth child | li:nth-child(2) { font-weight: bold; } |
| :not(selector) | Selects everything **except** the specified selector | button:not(.primary) { background: gray; } |
| :checked | Selects a checked radio/checkbox input | input:checked + label { color: green; } |
| :disabled / :enabled | Targets disabled or enabled inputs | input:disabled { background: #ccc; } |
| :valid / :invalid | Selects valid or invalid form inputs | input:valid { border: green; } |

**✅ Pseudo-Elements**

**Definition:** A pseudo-element lets you style specific parts of an element.

**🔹 Common Pseudo-Element Examples:**

| **Pseudo-Element** | **Description** | **Example** |
| --- | --- | --- |
| ::first-letter | Styles the **first letter** of a block of text | p::first-letter { font-size: 200%; } |
| ::first-line | Styles the **first line** of a block of text | p::first-line { font-weight: bold; } |
| ::before | Inserts content **before** an element | p::before { content: "→ "; color: gray; } |
| ::after | Inserts content **after** an element | p::after { content: " ✔"; color: green; } |
| ::selection | Styles the text selected by the user | ::selection { background: yellow; } |

**Flex Items Properties:**

* order

Example :

<div class="flex-container">  
  <div style="order: 3">1</div>  
  <div style="order: 2">2</div>  
  <div style="order: 4">3</div>  
  <div style="order: 1">4</div>  
</div>

* flex-grow : adjust self size
* flex-shrink
* flex-basis : sets initial length of the item

**✅ 1. Static (Default Position)**

* This is the default for all elements.
* The element stays in the normal document flow, and top, left, bottom, right have no effect.

**Example:**

<div style="position: static; border: 2px solid blue;">

This is static (default).

</div>

**Behavior:**  
The box stays exactly where it appears in the normal flow of the HTML.

**✅ 2. Relative**

* Moves **relative to its original position**.
* Still takes up its original space in the layout.

**Example:**

<div style="position: relative; top: 20px; left: 30px; border: 2px solid green;">

This is relative (moved down 20px and right 30px).

</div>

**Behavior:**  
The element shifts from its original position, but its original space remains occupied.

**✅ 3. Absolute**

* Positioned **relative to the nearest ancestor** that is not static.
* If no such ancestor, positions relative to the <html> element (viewport).

**Example:**

<div style="position: relative; border: 2px solid orange; height: 150px;">

Parent (Relative)

<div style="position: absolute; top: 10px; left: 10px; background: yellow;">

Child (Absolute)

</div>

</div>

**Behavior:**  
The yellow box is placed inside the orange box, 10px from the top and left **of the parent container** (because the parent has position: relative).

**✅ 4. Fixed**

* Always relative to the **viewport**.
* Does **not move when the page scrolls**.

**Example:**

<div style="position: fixed; bottom: 10px; right: 10px; background: red; color: white;">

Fixed Box

</div>

**Behavior:**  
The box stays at the bottom-right corner even if you scroll the page.

**✅ 5. Sticky**

* Starts as **relative**, then becomes **fixed** when you scroll past a certain point (defined by top, bottom, etc.).
* Sticks inside its **parent container**.

**Example:**

<div style="height: 500px; background: #f0f0f0;">

Scroll down

<div style="position: sticky; top: 20px; background: lightblue; padding: 10px;">

I am sticky! (stays 20px from top while scrolling)

</div>

</div>

**Behavior:**  
When you scroll, the blue box moves normally until it is 20px from the top of the viewport. Then it stays stuck while its parent container is still in view.

**✅ Quick Demo: Full HTML Code**

Here’s a combined example to see all positions in action:

<!DOCTYPE html>

<html>

<head>

<style>

body {

height: 2000px; /\* to enable scrolling \*/

}

.static {

position: static;

border: 2px solid blue;

}

.relative {

position: relative;

top: 20px;

left: 30px;

border: 2px solid green;

}

.parent {

position: relative;

height: 150px;

border: 2px solid orange;

margin-top: 20px;

}

.absolute {

position: absolute;

top: 10px;

left: 10px;

background: yellow;

}

.fixed {

position: fixed;

bottom: 10px;

right: 10px;

background: red;

color: white;

padding: 10px;

}

.sticky-container {

height: 500px;

margin-top: 20px;

background: #f0f0f0;

}

.sticky {

position: sticky;

top: 20px;

background: lightblue;

padding: 10px;

}

</style>

</head>

<body>

<div class="static">Static Position</div>

<div class="relative">Relative Position</div>

<div class="parent">

Parent (Relative)

<div class="absolute">Absolute Position</div>

</div>

<div class="sticky-container">

<div class="sticky">Sticky Position</div>

</div>

<div class="fixed">Fixed Position</div>

</body>

</html>

**✅ 1. Block Element**

* **Takes the full width** of its parent container (100% by default).
* Always starts on a **new line**.
* You can set **width, height, margin, padding**.
* **Examples:** <div>, <p>, <h1>.

**Example:**

<div style="display: block; background: lightgreen; width: 200px;">

I am a block element

</div>

**✅ 2. Inline Element**

* **Does NOT start on a new line**, stays in the same line as other inline elements.
* Takes only as much width as the content needs.
* **Cannot set width/height** (only padding left/right works, top/bottom doesn’t affect layout).
* **Examples:** <span>, <a>, <strong>.

**Example:**

<span style="display: inline; background: yellow;">

I am inline

</span>

**✅ 3. Inline-Block**

* Behaves like **inline** (doesn't break line) but **allows width, height, margin, and padding** like block elements.
* Great for creating **buttons** or **horizontal menus**.

**Example:**

<span style="display: inline-block; background: orange; width: 100px; height: 50px;">

Inline-block Box

</span>

**✅ 4. None**

* The element is **completely hidden** (not displayed and no space is taken).

**Example:**

<p style="display: none;">You can't see me!</p>

**✅ Visual Difference**

| **Type** | **Starts New Line** | **Width & Height** | **Example Use** |
| --- | --- | --- | --- |
| **Block** | Yes | Yes | Div, Sections |
| **Inline** | No | No | Links, Span |
| **Inline-Block** | No | Yes | Buttons, Nav |