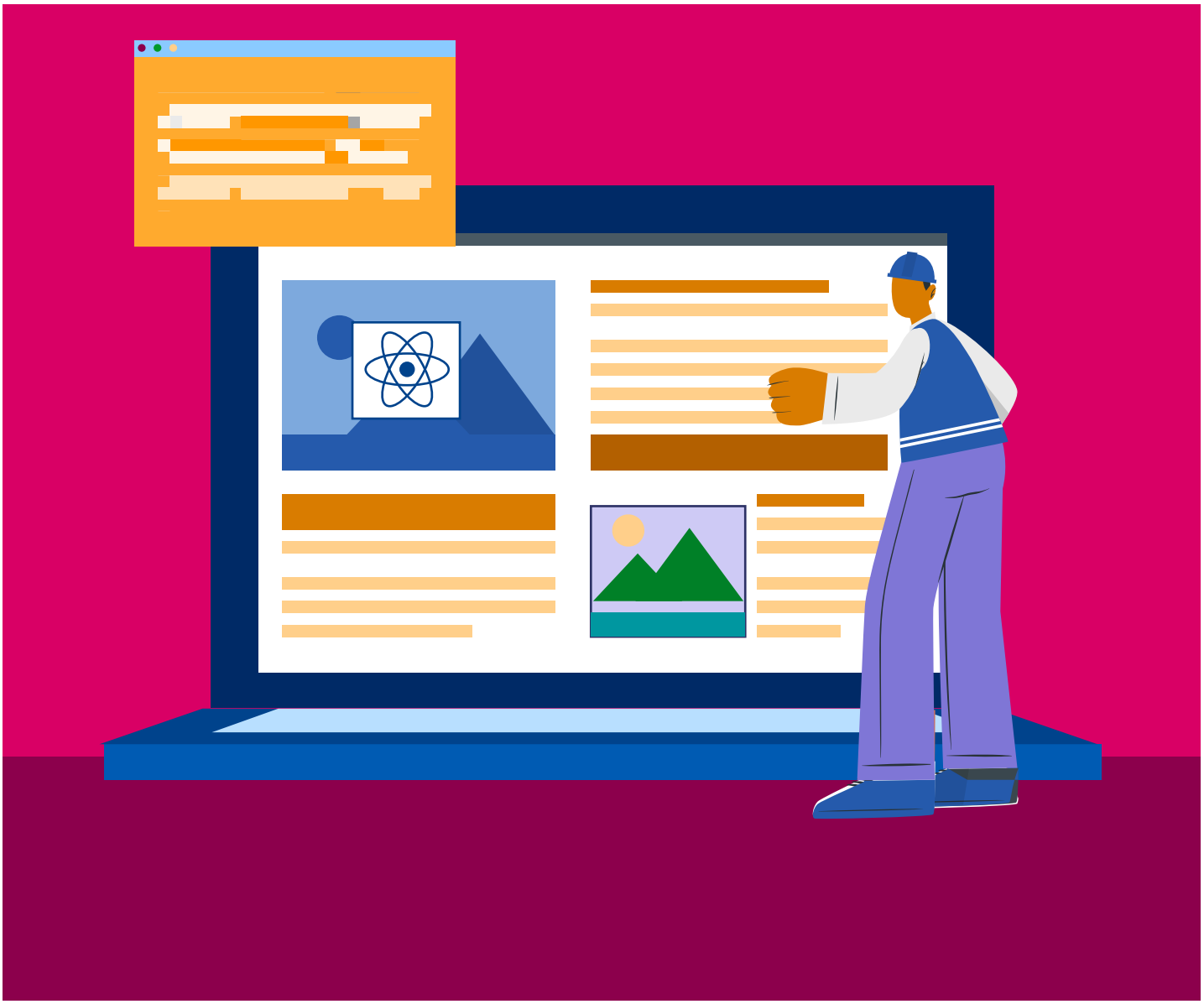
SCALER Topics

Cheatsheet

# Cascading Style Sheets



# India's best tech learning company

Learn industry-relevant skills with top tech veterans

**126%** Avg. CTC Hike | **Top 1%** Industry Instructors | **900+** Placement Partners

## Programs We Offer

**Software Development Course**

**Data Science & Machine Learning**

## The Scaler Recipe to Transform Your Career

- A structured & flexible program, that cares for you
- Be Mentored 1:1 by Experienced Professionals
- Become part of a thriving community for life

## Discover & connect with Alumni

**Sudhanshu Gera**
Software Engineer III

| Pre Scaler | Post Scaler |
|---|---|
| Wipro Limited | Walmart |

200% Hike

**Ankit Pangasa**
Senior Software Engineer

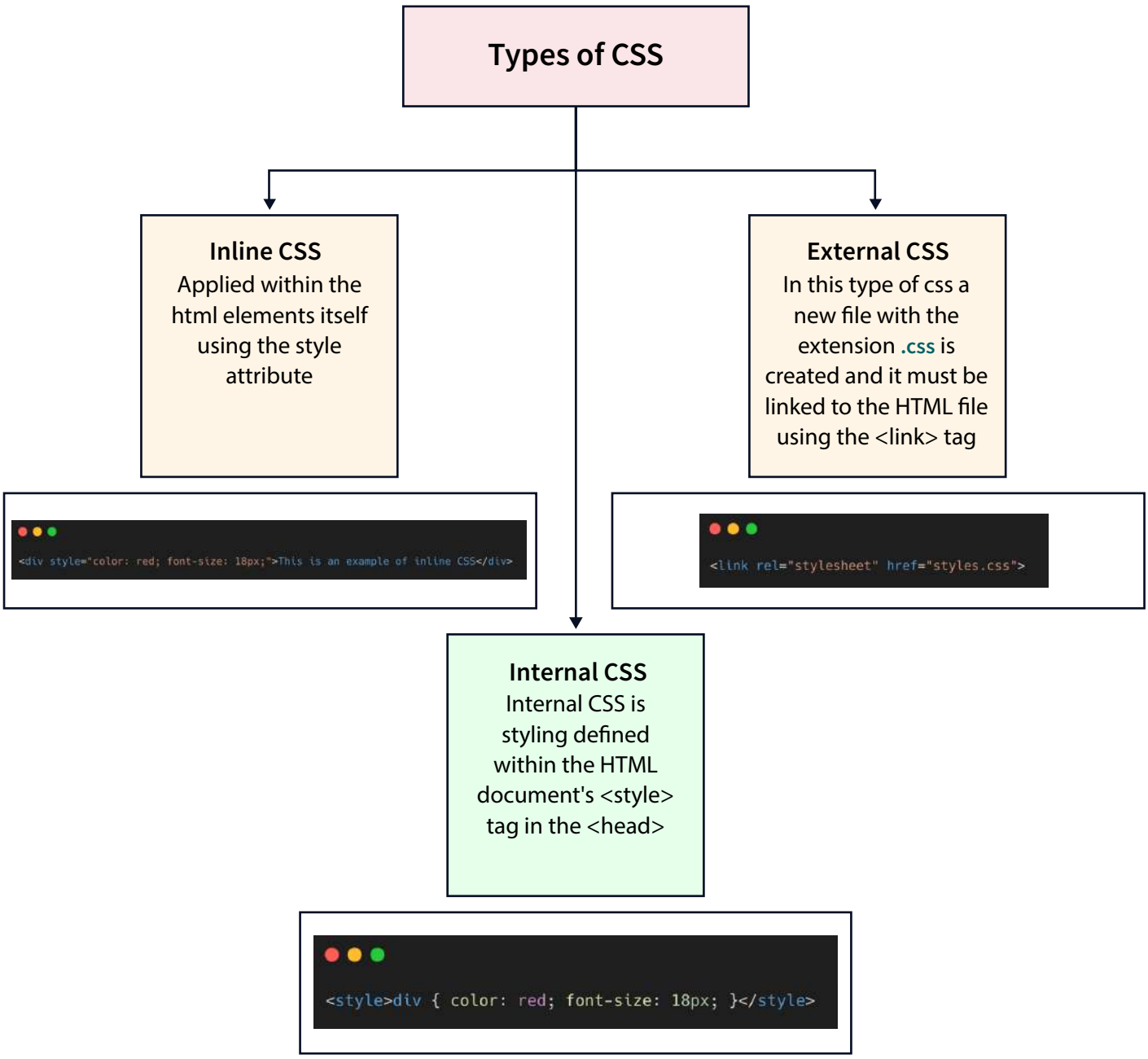
| Pre Scaler | Post Scaler |
|---|---|
| Adobe | Google |

200% Hike

Connect with Alumni

# 01 CSS (Cascading Style Sheet)

CSS (Cascading Style Sheets) is a language for styling web pages, controlling layout, and enhancing visual presentation.



# 02 Selectors

## Common Selectors

**HTML Code**

```
<body>
<div class="container">This is an example of class
selector</div>
<span>This is an example of element selector</span>
<p>This is an example of pseudo selector</p>
</body>
```

### Class Selector

| Syntax | `.classname{...}` |
|---|---|

**Usecase**

This is an example of class selector

Before applying

This is an example of class selector

After applying

**Code**

```
.container {
  color: blue;
  font-size: 20px;
  /* Add any other styles you want */
}
```

## Element Selector

| Syntax | `elementname{...}` |
|---|---|

**Usecase**

This is an example of element selector

Before applying

This is an example of element selector

After applying

**Code**

```
span {
        color: green;
        font-size: 18px;
        /* Add any other styles you want */
    }
```

## Pseudo Selector

| Syntax | `selector::pseudo-class{...}` |
|---|---|

**Usecase**

This is an example of pseudo selector

Before applying

This is an example of pseudo selector

After applying

**Code**

```
p::first-letter {
        font-size: 24px;
        font-weight: bold;
        color: red;
        /* Add any other styles you want */
      }
```

# Advanced Selectors

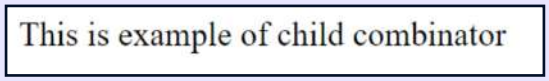
**HTML Code**

```
<body>
<div>This is example of child combinator</div>
  <div>This is example of Adjacent Sibling Combinator (+)</div>
<p>This paragraph immediately follows the div and will have bold red text</p>
  <div>This is example of General Sibling Combinator (~)</div>
</body>
```

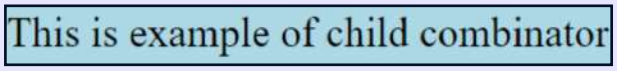
## Child Combinator(>)

| Syntax | `parent>child {...}` |
|---|---|

**Usecase**




Before applying




After applying

**Code**

```
body > div {
      background-color: lightblue;
      padding: 10px;
    }
```

## Adjacent Sibling Combinator (+)

| Syntax | `element1+element2{...}` |
|---|---|

**Usecase**

This is example of Adjacent Sibling Combinator (+)

This paragraph immediately follows the div and will have bold red text.

**Before applying**

This is example of Adjacent Sibling Combinator (+)

**This paragraph immediately follows the div and will have bold red text.**

**After applying**

**Code**

```
div + p {
        font-weight: bold;
        color: red;
    }
```

## General Sibling Combinator (~)

| Syntax | `element1 ~ element2 {..}` |
|---|---|

**Usecase**

This is example of General Sibling Combinator (~)

This paragraph is a sibling of the div and will have italic green text.

**Before applying**

This is example of General Sibling Combinator (~)

*This paragraph is a sibling of the div and will have italic green text.*

**After applying**

**Code**

```
div ~ p {
        font-style: italic;
        color: green;
    }
```

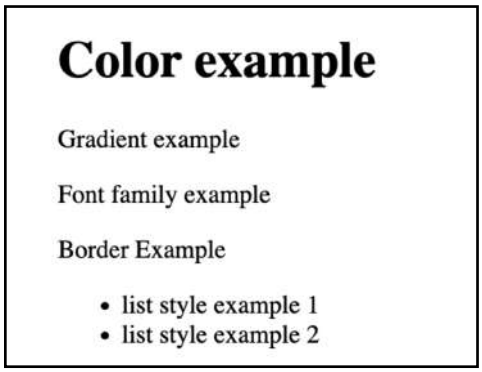
## Basic Selectors

| Property Name | Description |
|---|---|
| Element Selector | Selects HTML elements based on their tag names. **Syntax: elementName { … }** |
| Class Selector | Selects HTML elements based on their class attribute values. **Syntax: .className { … }** |
| Pseudo Selectors | Selects elements based on their position or state. **Syntax: selector:pseudo-class { … }** |
| Pseudo-elements | Allows styling specific parts of an element, such as the first line or first letter. **Syntax: selector::pseudo-element { … }** |

## Advanced Selectors

| Property Name | Description |
|---|---|
| ::after | Inserts content after the content of selected elements. **Syntax: selector::after { content: ""; … }** |
| ::first-line | Styles the first line of text within selected elements. **Syntax: selector::first-line { … }** |
| ::first-letter | Styles the first letter of text within selected elements. **Syntax: selector::first-letter { … }** |
| Child Combinator (>) | Selects direct child elements of a parent element. **Syntax: parent > child { … }** |
| Adjacent Sibling Combinator (+) | Selects an element immediately following another element. **Syntax: element1 + element2 { … }** |

# 03 Common Properties

| HTML Code | Output before applying CSS |
| --- | --- |
|  |  |





## Color Property

| Syntax | `color: ...` |
| --- | --- |

| Usecase | |
| --- | --- |
| Color Example<br>Before applying | Code |
| Color Example<br>After applying |  |

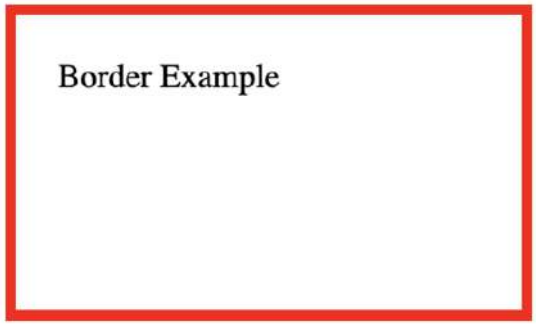
## Border Property

| Syntax | `border: ...` |
| --- | --- |

**Usecase**

Border example

Before applying




After applying

Code

```
div {
 width: 200px;
 height: 100px;
 border: 2px solid red;
 padding: 20px;
 }
```

## Gradient Property

| Syntax | `background: linear-gradient(to right, color1, color2, ...);` |
|---|---|

**Usecase**

Gradient example

Before applying


After applying

**Code**

```
div {
        width: 200px;
        height: 200px;
        background: linear-gradient(to right, red, yellow);
        color: white;
        text-align: center;
        line-height: 200px;
      }
```

## Font Family Property

| Syntax | `font-family:...;` |
|---|---|

**Usecase**

Font family example

Before applying

**Font family example**

After applying

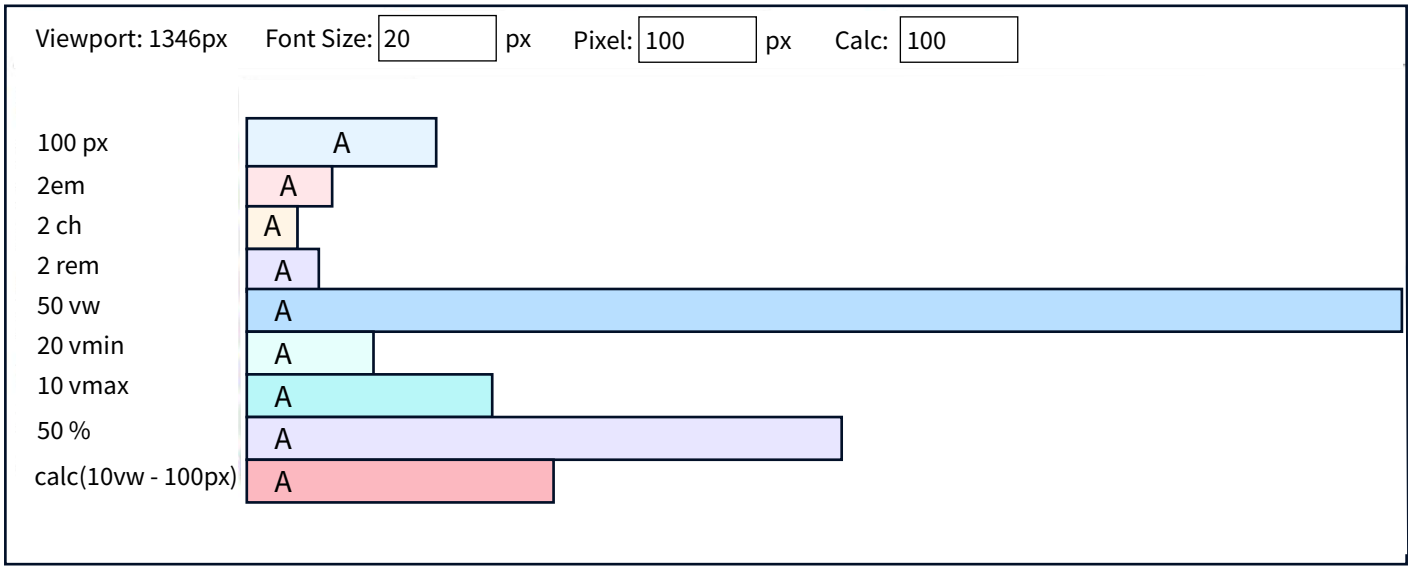
**Code**

```
p {
        font-family: "Franklin Gothic Medium";
      }
```

# 04 Units In CSS

| Unit | Description | Example |
|---|---|---|
| px | Pixel Units | width: 200px; |
| % | Percentage relative to the parent element | width: 50%; |
| em | Relative to the font-size of the element | font-size: 1.5em; |
| rem | Relative to the font-size of the root element | font-size: 1.2rem; |
| vh | Percentage of the viewport height | height: 50vh; |
| vw | Percentage of the viewport width | width: 25vw; |

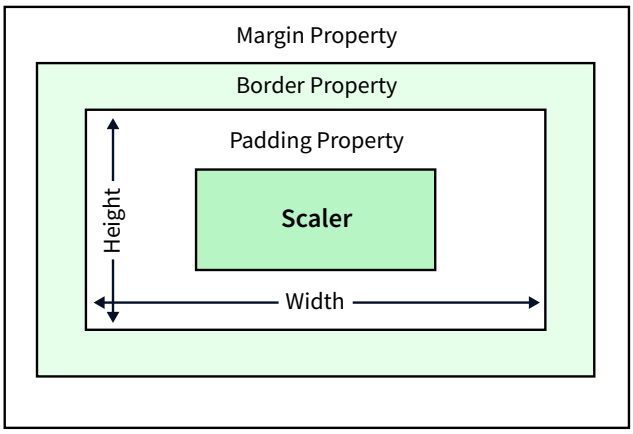
## Example



# 05 CSS Box Model

| Property | Description |
|---|---|
| Content | The actual content of the box, where text and images appear. |
| Padding | Transparent area around the content, inside the border. |
| Border | A border surrounding in the padding (optional) |
| Margin | Transparent area outside the border, separating boxes |
| Width | Total width of the box, including content, padding, and border |
| Height | Total height of the box, including content, padding, and border |
| Box-Sizing | Defines how the width and height of an element are calculated. |

## CSS Box-Model Property

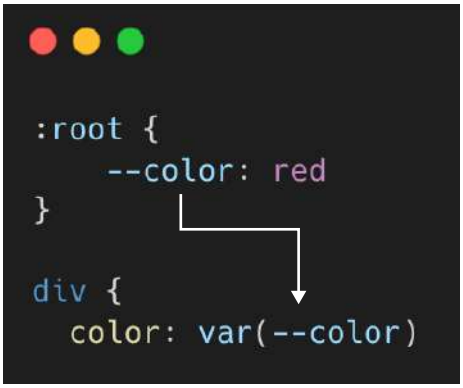



### Importance in Layout

The CSS Box Model is essential for web layout:

- Width and height properties control element size.
- Padding creates space between content and border.
- Border visually separates elements.
- Margin provides spacing outside elements, affecting layout.

# 06 CSS Variables

| Global Scope | Variable Scope |
| --- | --- |
| CSS variables declared at the root level or within the :root selector have a global scope, accessible throughout the entire stylesheet. | Locally scoped CSS variables, defined within a specific selector or rule set, are accessible only within that scope and its descendant elements, providing more granular control over styling. |
|  |  |





# 07 Filters in CSS

| Filter | Description |
| --- | --- |
| blur() | Applies a blur effect to an element. |
| brightness() | Adjusts the brightness of an element. |
| contrast() | Adjusts the contrast of an element. |
| drop-shadow() | Applies a drop shadow effect to an element. |

| | |
|---|---|
| grayscale() | Converts an element to grayscale. |
| hue-rotate() | Rotates the hue of an element's color. |
| invert() | Inverts the colors of an element. |
| opacity() | Adjusts the opacity of an element. |
| saturate() | Increases or decreases the saturation of an element. |
| sepia() | Applies a sepia tone effect to an element. |

## Example

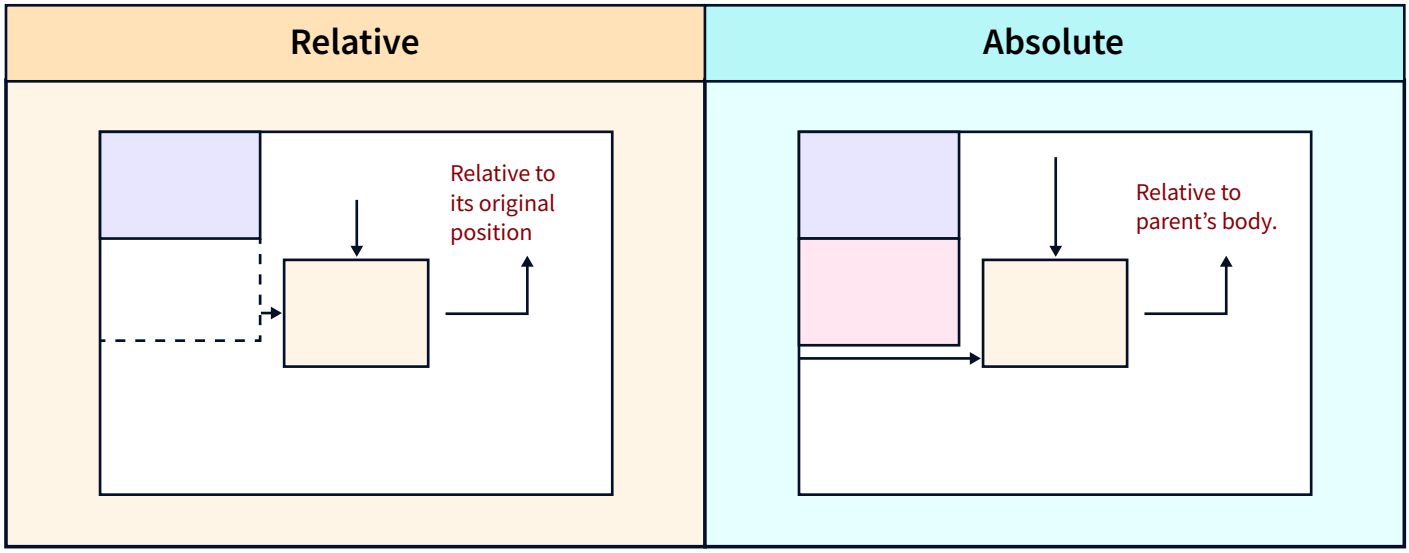

### How to use

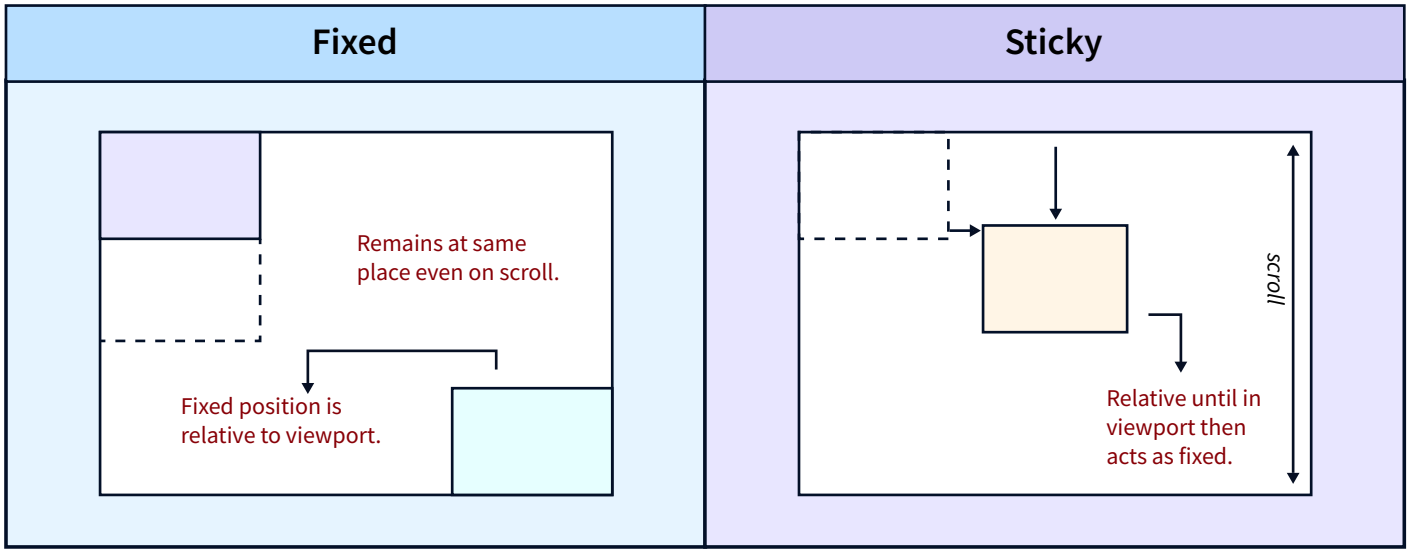
```
.element {
    blur: 5px; /* Applies a blur effect with a radius of 5 pixels */
    brightness: 150%; /* Adjusts the brightness to 150% */
    contrast: 120%; /* Adjusts the contrast to 120% */
    drop-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5); /* Applies a drop shadow effect */
    filter: grayscale(100%); /* Converts the element to grayscale */
}
```

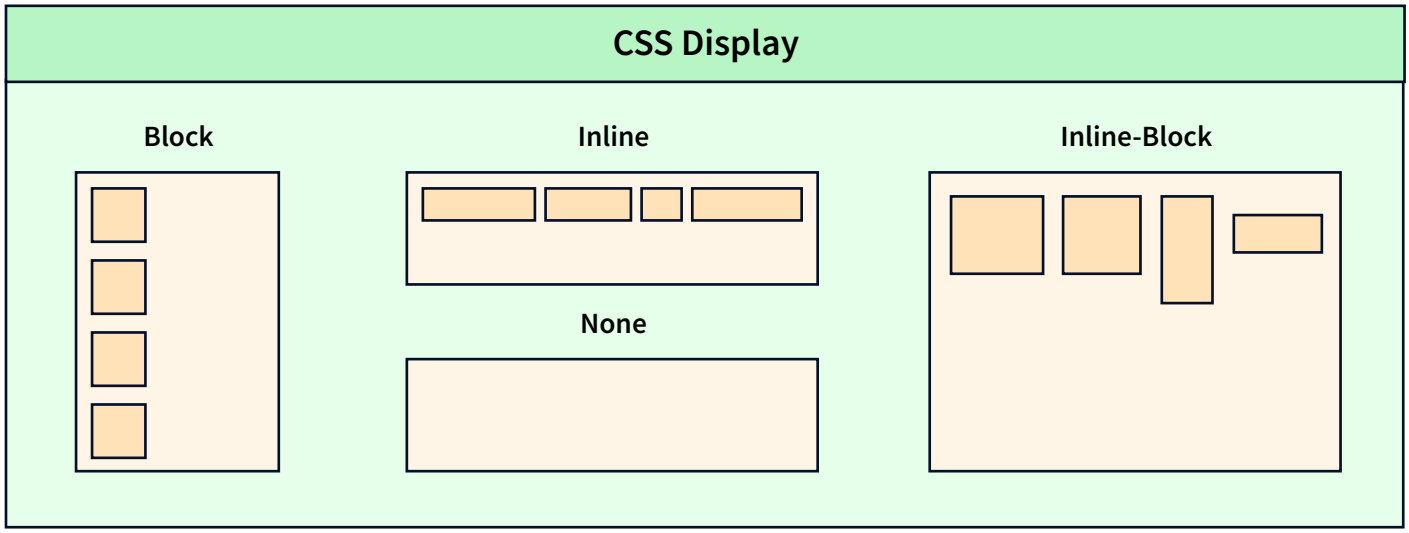
# 08 CSS Layout And Positioning

| Property | Specifiers | Description |
|---|---|---|
| display | | Specifies how an element is displayed. |
| | block: | Renders as a block-level element. |
| | inline: | Renders as an inline element. |
| | inline-block: | Combines features of inline and block. |
| | flex: | Enables a flexible layout model for its direct children. |
| | grid: | Enables a grid layout for its direct children. |
| position | | Specifies the positioning method used for an element. |
| | static: | Default positioning. |
| | relative: | Positions an element relative to its normal position. |
| | absolute: | Positions an element relative to its closest positioned ancestor. |
| | fixed: | Positions an element relative to the viewport. |
| | sticky: | Positions an element based on its position in the viewport until a specified scroll threshold is reached. |

# CSS Positioning



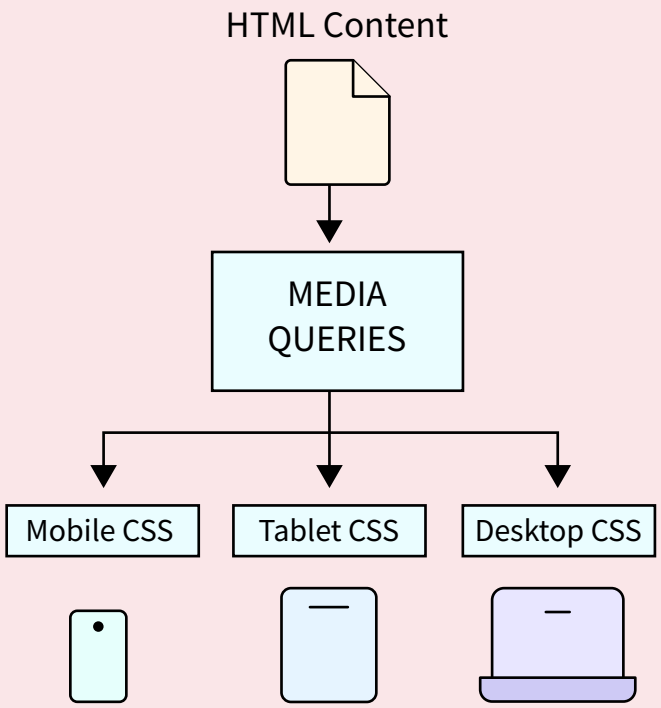



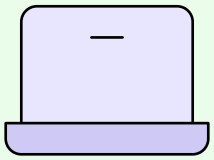
# 09 Responsive Design in CSS

Responsive design in CSS utilizes media queries to apply specific styles based on the characteristics of the device, such as screen width, height and orientation.

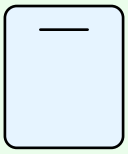
```
/*Media Queries */
@media screen and (max-width:600px)
{
  body {
      font-size: 14 px;
  }
```

## Examples

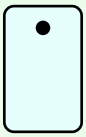


@media screen and (min-width 1024 px) {...}



@media screen and (min-width 768 px) and (max-width: 1023px) {...}



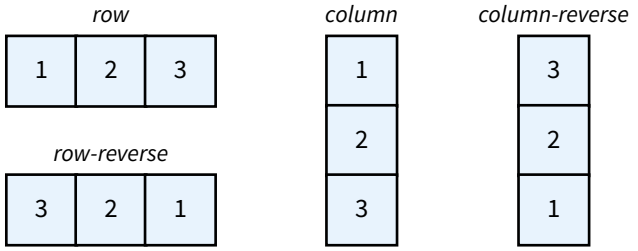
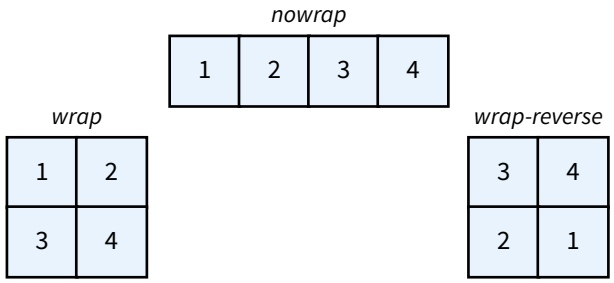
@media screen and (min-width 767 px) {...}

# 10 Flexbox in CSS

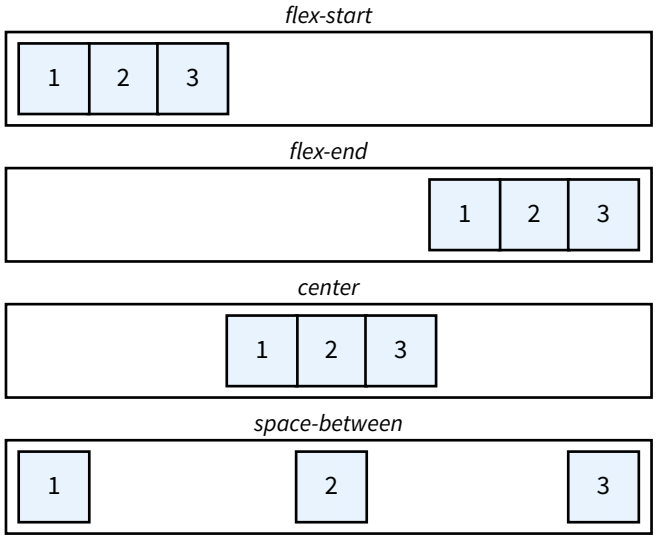
## Flex Container

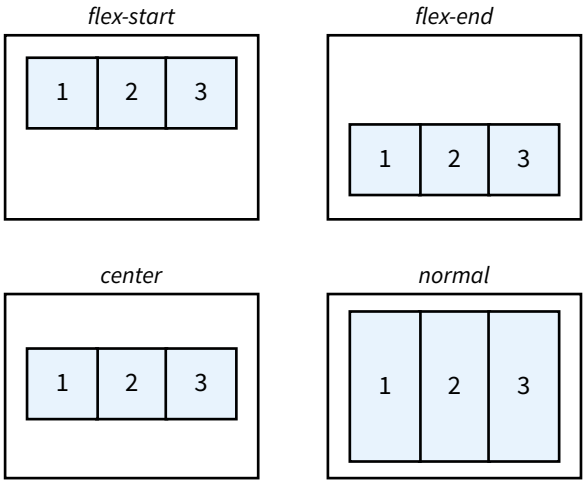
| Definition | The parent element that contains flex items. |
|---|---|

| Property | Usage |
|---|---|
| `display: flex;` | Establishes a flex container, enabling the use of flex properties on its children. |

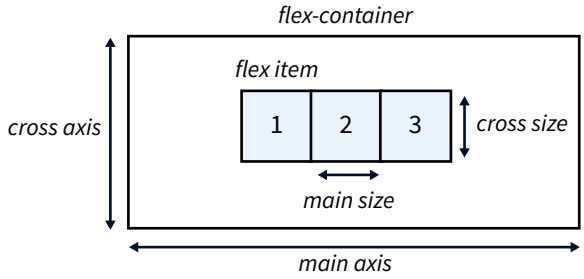
| Flex-Direction | Flex-Wrap |
|---|---|
| Sets how the flex items are placed inside the container. | Tells the container whether to wrap the items or not |
|  |  |

| Justify-Content | Align-Content |
| --- | --- |
| Tells the container how to align items horizontally. | Tells the container how to align items vertically. |




## Terminology




## Flex Items

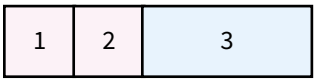
| Definition | The child elements within a flex container. |
| --- | --- |

| Properties | Usage |
| --- | --- |
| `flex-grow:` | Defines the ability for a flex item to grow. |
| `flex-shrink:` | Defines the ability for a flex item to shrink. |
| `flex-basis:` | Specifies the initial size of a flex item. |

## Flex-Basis

Sets the main size of a flex item.

*250px*




## Flex-Grow

Sets how much space should be taken up by the flex item.

*1*


*with a flex-grow set to 0 for the other items*

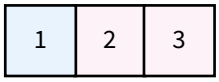
## Flex-Shrink

Sets how much should a flex item shrink relative to others

*0*


*with a flex-basis set to 250px*

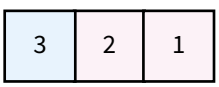
*1*



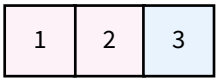

## Order

Sets the ordering of the complex items.

*-1*



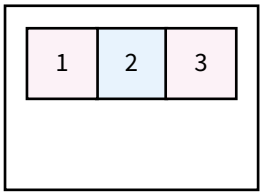

*0*



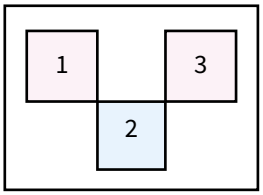

## Align-Self

Sets how individual items should be aligned inside the container

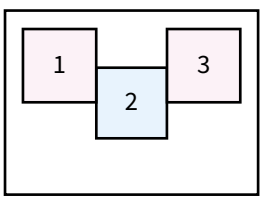
*flex-start*



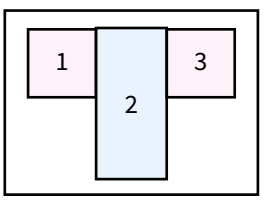

*flex-end*




*center*




*stretch*



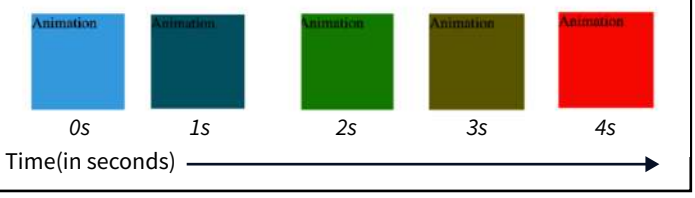
# 11 Transitions And Animations

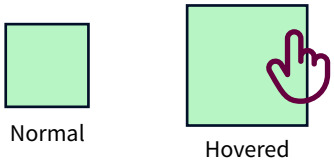
| Transitions | Animations |
| --- | --- |
| Smooth changes in property values. | Key-frame based animation effects. |
| `transition-property` | `@keyframes` |
| `transition-duration` | `animation-name` |
| `transition-timing-function` | `animation-duration` |
| `transition-delay` | `animation-timing-function` |
| | `animation-delay` |
| | `animation-iteration-count` |
| | `animation-direction` |
| | `animation-fill-mode` |
| | `animation-play-mode` |

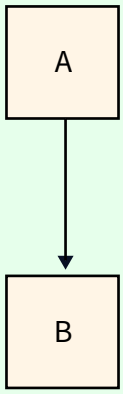
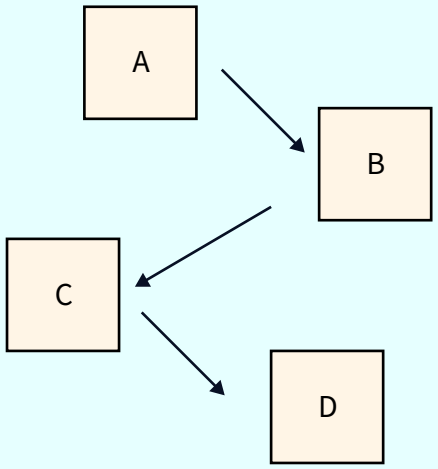
## Example

```css
.box{
      width: 100px;
      height: 100px;
      background-color: #3498db;
      margin: 20px;
      display: inline-block;
      animation-name: colorChange;
      animation-duration: 4s;
    }

    .box:hover {
      transform: scale(1.2);
    }

    @keyframes colorChange {
      0% { background-color: blue; }
      50% { background-color: green; }
      100% { background-color: red; }
    }
```

## Output Frame by Frame



| Transitions | Animations |
| --- | --- |
|   |   |

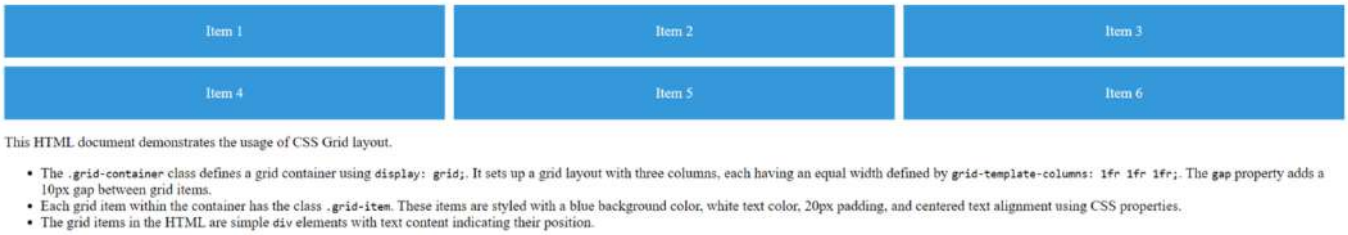
# 12 CSS Functions

| CSS Functions | Description | Syntax |
| --- | --- | --- |
| calc( ) | Allows mathematical calculations within CSS property values. | `width: calc(50% - 20px);` |
| var( ) | Defines and references custom CSS variables for reusable values. | `--main-color: #3498db;`<br>`color: var(--main-color);` |
| attr( ) | Retrieves HTML attribute values for styling based on element attributes. | `content: attr(data-text);` |
| rgb( ) | Specifies colors using the RGB color model. | `color: rgb(255, 0, 0); /* Red */` |

# 13 CSS Grids

| Topics | Descriptions | Example Syntax |
|---|---|---|
| Grid Container | Defines a grid container to hold grid items. | `display: grid;` |
| Grid Items | Elements placed inside a grid container. | `grid-column, grid-row` |

**Example**




# 14 Transformations in CSS

**2D Transformations** Apply transformations in a 2D plane

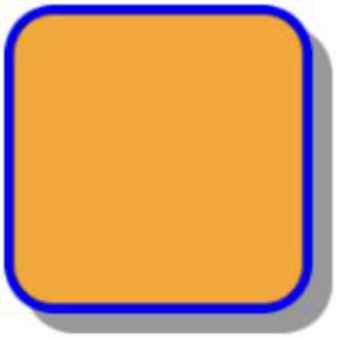
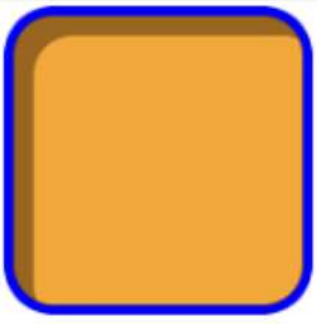
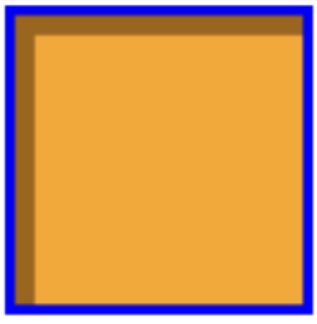
| Topics | Descriptions | Example Syntax |
|---|---|---|
| rotate | Rotates an element | `transform: rotate (45deg);` |
| scale | Scales an element | `transform: scale(1.5);` |

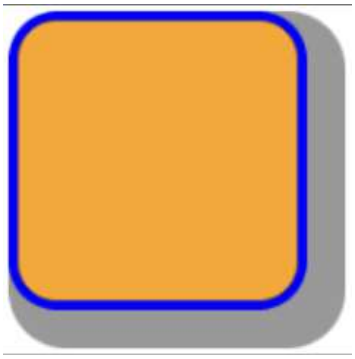
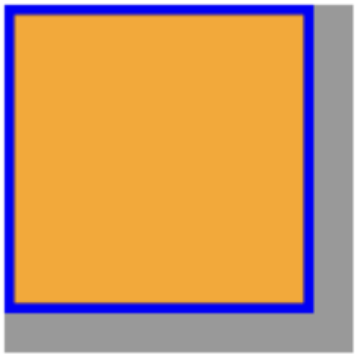
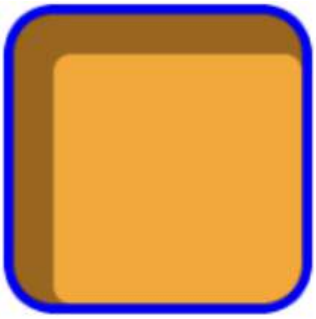
| | | |
|---|---|---|
| translate | Moves an element along the X and Y axes | `transform: translate (20px,30px);` |

**3D Transformations** Apply transformations in a 3D space

| | | |
|---|---|---|
| rotateX | Rotates an element around the X-axis | `transform: rotateX(45deg);` |
| rotateY | Rotates an element around the Y-axis | `transform: rotateY(45deg);` |
| translateZ | Moves an element along the Z-axis | `transform: translateZ (50px);` |

# 15 CSS 3 Features

| | | |
|---|---|---|
| `border 5px solid blue;`<br>`background-colour orange;`<br>`width: 144px;`<br>`height 144px;` | `border-radius: 20px;` | `border-radius: 0;` |
| `box-shadow:`<br>`rgba {0,0,0,0,4}`<br>`10px 10px;` |  |  |
| `box-shadow:`<br>`rgba {0,0,0,0,4}`<br>`10px 10px;`<br>`inset` |  |  |

| | | |
|---|---|---|
| `box-shadow:`<br>`  rgba {0,0,0,0,4}`<br>`  10px 10px 0`<br>`  10px /* spread */` |  |  |
| `box-shadow:`<br>`  rgba {0,0,0,0,4}`<br>`  10px 10px 0`<br>`  10px /* spread */`<br>`  inset` |  |  |

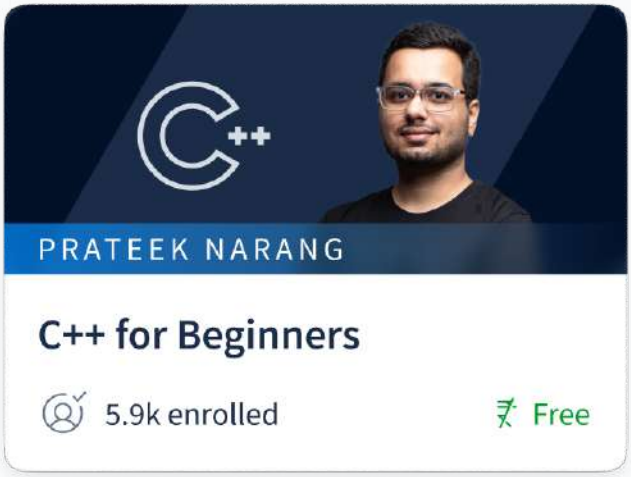
## Key Concepts in CSS Mastery

- **Selector Specificity:** Understand the hierarchy of CSS selectors to ensure proper application of styles.
- **Box Model:** Master the concept of content, padding, border, and margin to effectively control element layout.
- **Responsive Design:** Implement media queries to create designs that adapt seamlessly across different devices and screen sizes.
- **Cross-Browser Compatibility:** Test and ensure that CSS styles render consistently across various web browsers for a uniform user experience.
- **CSS Preprocessors:** Utilize tools like Sass or Less to streamline CSS development by enabling features like variables, mixins, and nested rules.

# SCALER TOPICS

Unlock your potential in software development with
**FREE COURSES** from **SCALER TOPICS!**

**Register now and take the first step towards your future Success!**





**That's not it. Explore 20+ Courses by clicking below**

Explore Other Courses

Practice **CHALLENGES**
and become 1% better everyday



Explore Other Challenges