**Selectors in CSS**

A CSS selector is a pattern used to select the HTML elements you want to style. CSS selectors can range from simple to complex, allowing you to target

elements in various ways.

Selectors

Attribute selectors

Combinator

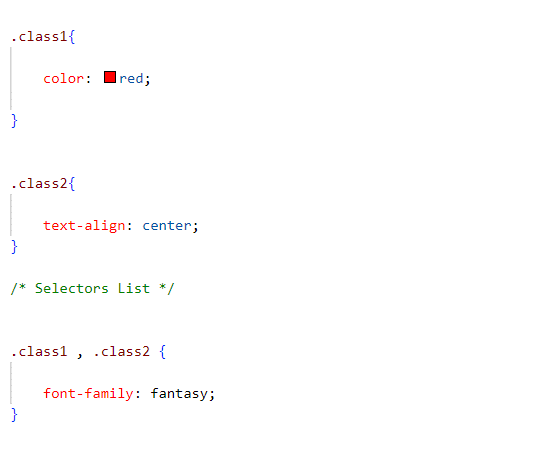
Pseudo class Selector

Pseudo Elements

Simple Selectors

1. Descendant selector
2. Child selector
3. Adjacent sibling selector
4. General sibling selector
5. Descendant selector
6. Child selector
7. Adjacent sibling selector
8. General sibling selector
9. Universal Selectors
10. Element Selectors
11. Class Selectors
12. ID Selectors
13. Selectors list
14. **Simple Selectors**

Simple selectors in CSS are used to select elements based on their name, id, class, or attribute.

1. **Universal Selectors (\*)**:
   * Selects all elements on a page.
   * Example: \* { margin: 0; } applies zero margin to all elements.
2. **Element Selectors**:
   * Selects all elements of a given type.
   * Example: p { font-size: 16px; } selects all <p> elements and sets their font size to 16px.
3. **Class Selectors (.)**:
   * Selects all elements with a specific class attribute.
   * Example: .highlight { background-color: yellow; } selects all elements with the class highlight and sets their background color to yellow.
4. **ID Selectors (#)**:
   * Selects a single element with a specific ID attribute.
   * Example: #header { color: blue; } selects the element with the ID header and sets its text color to blue.
5. **Selectors List**:
   * Combines multiple selectors separated by commas, applying the same styles to all selected elements.
   * Example: h1, h2, h3 { margin-bottom: 10px; } selects all <h1>, <h2>, and <h3> elements and sets their bottom margin to 10px.
6. **Combinator**

Combinators in CSS are tools used to select elements based on their relationships with other elements.

1. **Descendant Combinator ( )**:

* Selects elements that are inside another element.
* Example: div p { color: red; } changes the text color of all <p> tags inside <div> tags to red.

Valid HTML Structure:

* Do not nest block-level elements inside <p> tags.
* Ensure CSS selectors reflect the valid HTML structure.

**Rules**:

<p> (paragraph) tags cannot contain block-level elements (e.g., <h1>, <div>, <ul>, etc.).

<p>

<h1>Invalid HTML</h1>

</p>

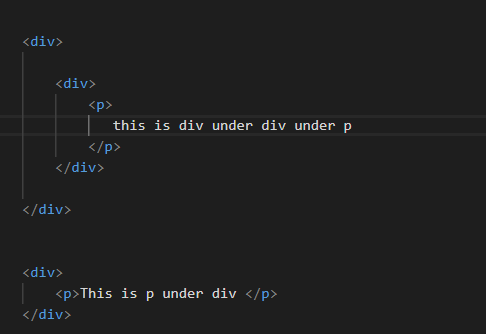
**Correct HTML Structure**

<div>

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

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

</div>



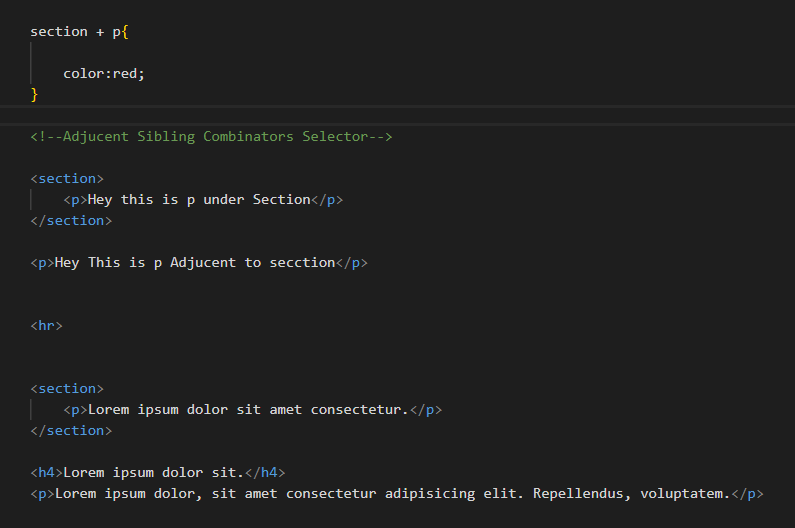
या मध्ये Syntax: ancestor descendant हे pair सगळे कडे सेम राहणार like **div p हे** जिथे असेल तिथ त्यची property लागते

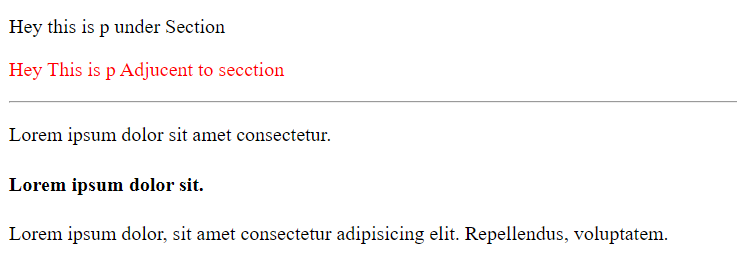
1. **Child Combinator (>)**:

* Selects elements that are directly inside another element.
* Example: ul > li { margin-bottom: 5px; } changes the bottom margin of <li> tags that are directly inside <ul> tags to 5px.

1. **Adjacent Sibling Combinator (+)**:

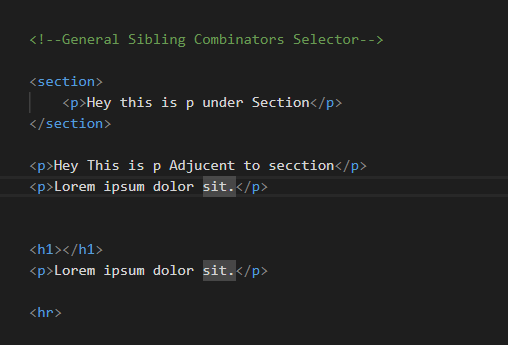
* Selects an element that comes immediately after another element.
* Example: h1 + p { margin-top: 0; } removes the top margin of a <p> tag that comes right after an <h1> tag.

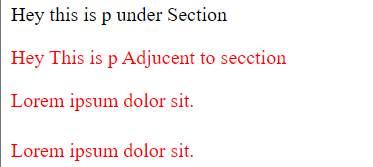




1. **General Sibling Combinator (~)**:

* Selects all elements that come after another element, even if there are other elements in between.
* Example: h1 ~ p { color: blue; } changes the text color of all <p> tags that come after an <h1> tag to blue, no matter how many tags are in between.





**3.Attribute selectors**

**Attribute Selectors**: Used to select elements based on the presence or value of their attributes.

1. **[attr]**:

* Selects elements with the specified attribute.
* Example: [type] { color: red; } selects all elements with a type attribute.

2. **[attr="value"]**:

* Selects elements with the specified attribute and value.
* Example: [type="text"] { background-color: yellow; } selects all elements with type="text".

3. **[attr~="value"]**:

* Selects elements with the specified attribute whose value is a space-separated list containing at least one specified value.
* Example: [class~="btn"] { font-weight: bold; } selects elements with a class attribute containing the word btn.

4. **[attr|="value"]**:

* Selects elements with the specified attribute whose value is either exactly "value" or starts with "value-" (used for language subcode).
* Example: [lang|="en"] { color: blue; } selects elements with a lang attribute value of "en" or starting with "en-".

5. **[attr^="value"]**:

* Selects elements with the specified attribute whose value begins with the specified value.
* Example: [href^="https"] { color: green; } selects elements with href attributes starting with "https".

6. **[attr$="value"]**:

* Selects elements with the specified attribute whose value ends with the specified value.
* Example: [src$=".jpg"] { border: 1px solid black; } selects elements with src attributes ending in ".jpg".

7. \**[attr*="value"]\*\*:

* Selects elements with the specified attribute whose value contains the specified value.
* Example: [title\*="flower"] { font-style: italic; } selects elements with title attributes containing the word "flower".

**BOX Model & Background**

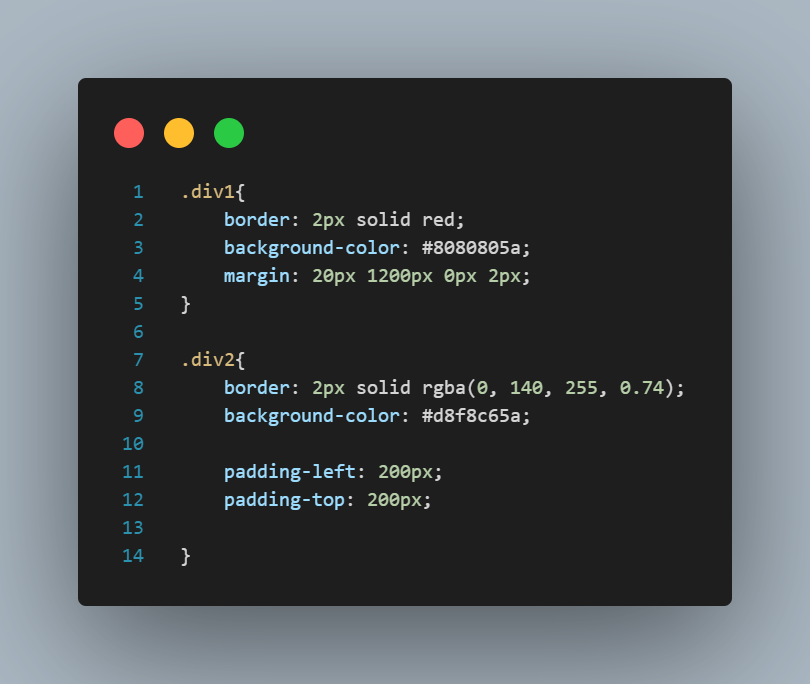
**Margine-Padding**

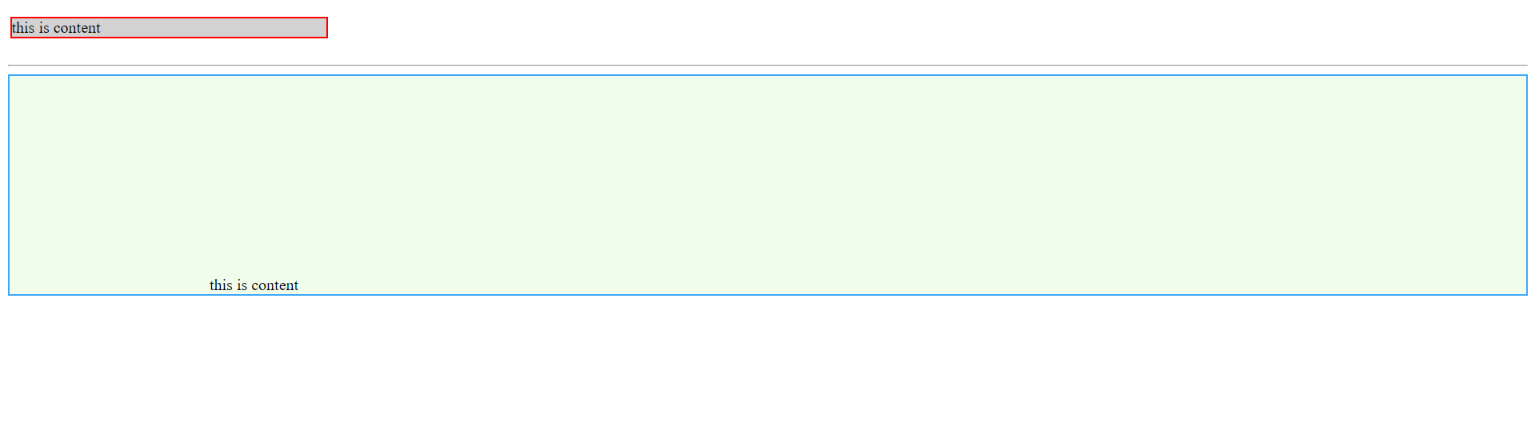
**Margin**

* **Margin**: Space outside the border of an element.
* **Syntax**:
  + **Single value**: margin: 10px; (applies to all sides)
  + **Two values**: margin: 10px 20px; (first value for top & bottom, second for left & right)
  + **Three values**: margin: 10px 20px 30px; (top, horizontal, bottom)
  + **Four values**: margin: 10px 20px 30px 40px; (top, right, bottom, left)
* **Individual sides**:
  + margin-top: 10px;
  + margin-right: 20px;
  + margin-bottom: 30px;
  + margin-left: 40px;

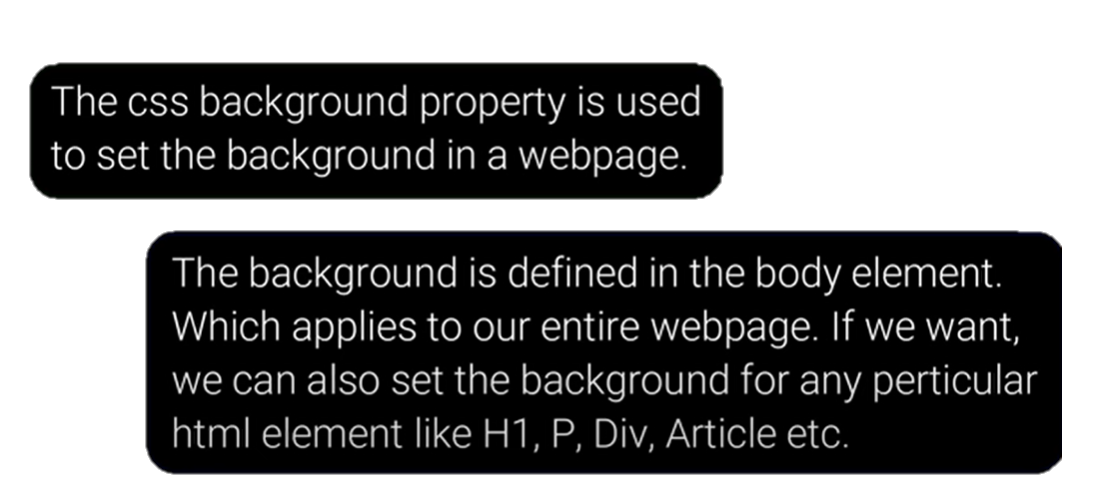
**Padding**

* **Padding**: Space inside the border of an element.
* **Syntax**:
  + **Single value**: padding: 10px; (applies to all sides)
  + **Two values**: padding: 10px 20px; (first value for top & bottom, second for left & right)
  + **Three values**: padding: 10px 20px 30px; (top, horizontal, bottom)
  + **Four values**: padding: 10px 20px 30px 40px; (top, right, bottom, left)
* **Individual sides**:
  + padding-top: 10px;
  + padding-right: 20px;
  + padding-bottom: 30px;
  + padding-left: 40px;





**Background Properties**





**CSS Background Properties: Sorted and Important Notes**

**1. background-color**

* **Sets the background color** of an element.
* **Syntax**: background-color: color;
* **Example**: background-color: #ff0000;

**2. background-image**

* **Sets the background image** of an element.
* **Syntax**: background-image: url('image.jpg');
* **Example**: background-image: url('pattern.png');

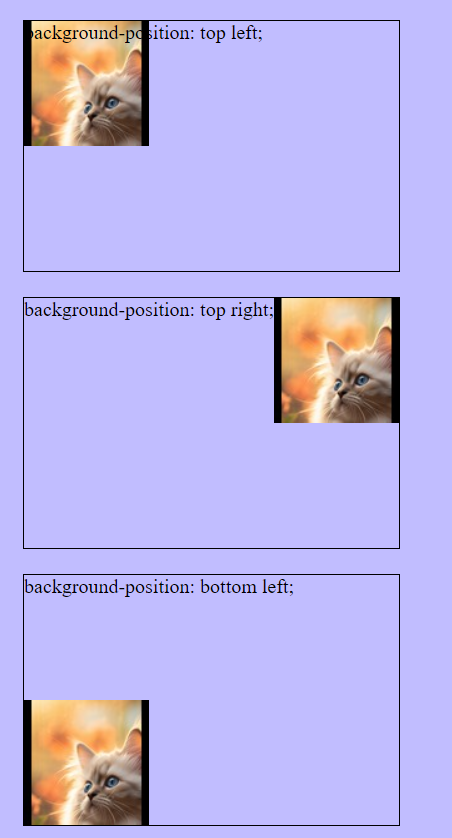
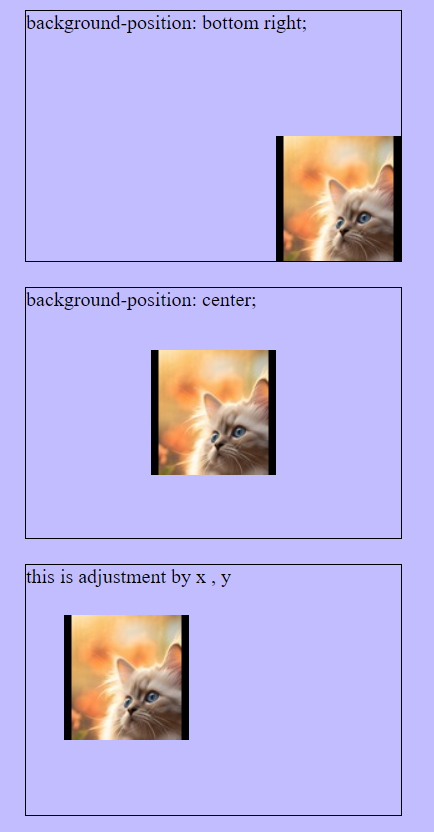
**3. background-repeat**

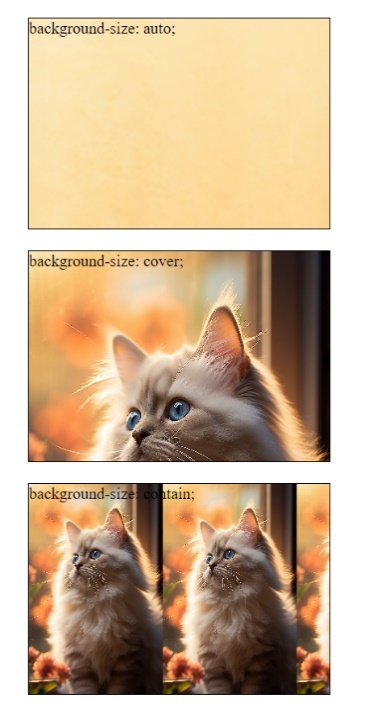
* **Defines how the background image is repeated**.
  + repeat: Repeats both horizontally and vertically (default).
  + repeat-x: Repeats horizontally.
  + repeat-y: Repeats vertically.
  + no-repeat: No repetition.
* **Syntax**: background-repeat: value;
* **Example**: background-repeat: no-repeat;



**4. background-position**

* **Sets the initial position** of the background image.
* **Syntax**: background-position: x-pos y-pos;
* **Example**: background-position: center center;



**5. background-size**

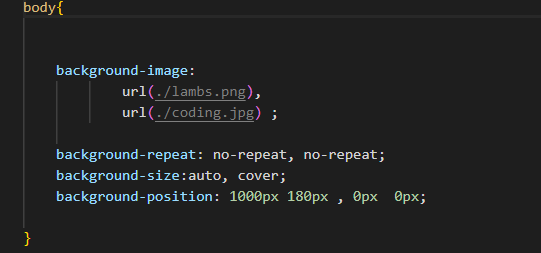
* **Specifies the size** of the background image.
  + cover: Scales the image to cover the entire element.
  + contain: Scales the image to be fully visible within the element.
* **Syntax**: background-size: value;
* **Example**: background-size: cover;

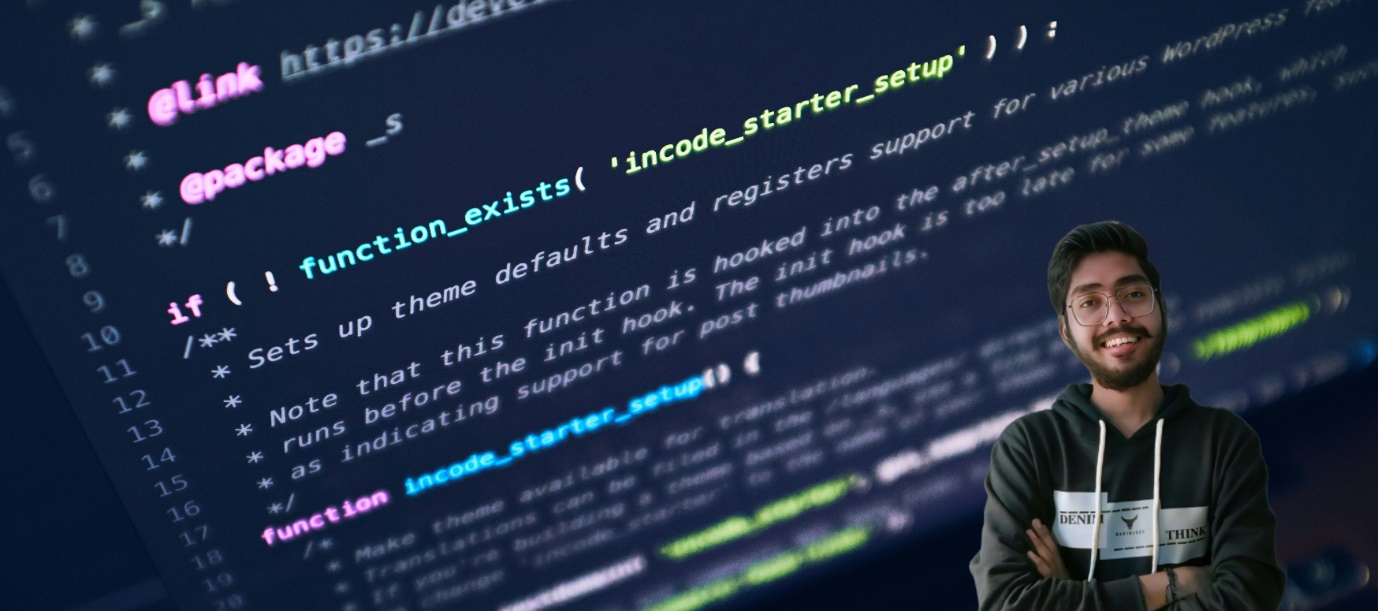
 **auto**: The background image retains its original dimensions.

 **cover**: The background image covers the entire element, maintaining aspect ratio but possibly cropping the image.

 **contain**: The background image is fully visible within the element, maintaining aspect ratio but possibly leaving empty space.

**Multiple Background:**





**6. background-attachments**

* **Sets whether a background image is fixed** or scrolls with the rest of the page.
  + **scroll**: Background scrolls with the element (default).
  + **fixed**: Background is fixed with regard to the viewport.
  + **local**: Background scrolls with the element's contents.
* **Syntax**: background-attachment: value;
* **Example**: background-attachment: fixed;

About the local:

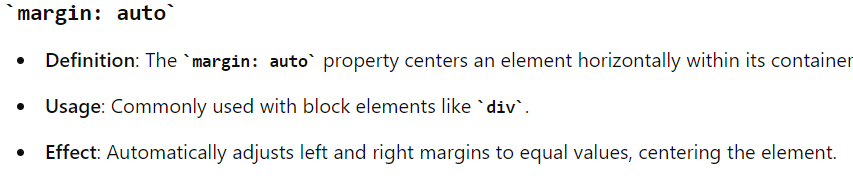
When you scroll the content inside the element, the background image moves along with it.

या मध्ये जेवा आपण block वर block ठवतो तेचा जर आपल्याला scroll property use करता येत नाहीत सो we use local

Module 19

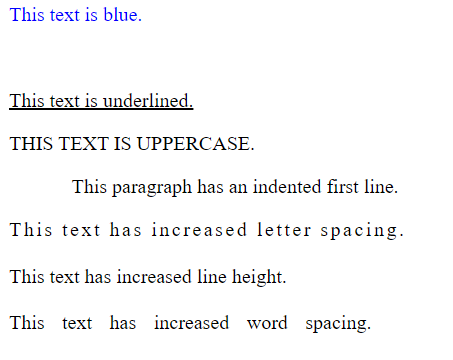
* **Margin And Padding**
* Text In CSS
* Font In CSS
* Display
* Height And Width In CSS

**Margin-Padding**

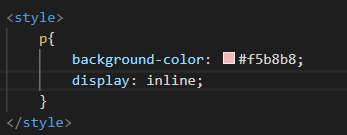


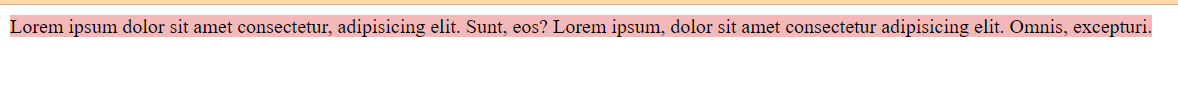
**Text properties And Font Property**





* **Display Properties**
  1. Display-inline
* The element is displayed as an inline element (takes up only as much width as necessary and does not start on a new line).
* This makes block element in inline



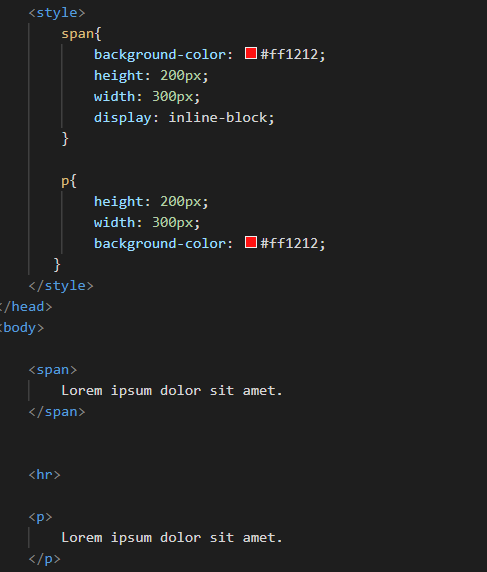


* 1. Display-Block

**Description**: The element is displayed as a block element (takes up the full width available and starts on a new line).

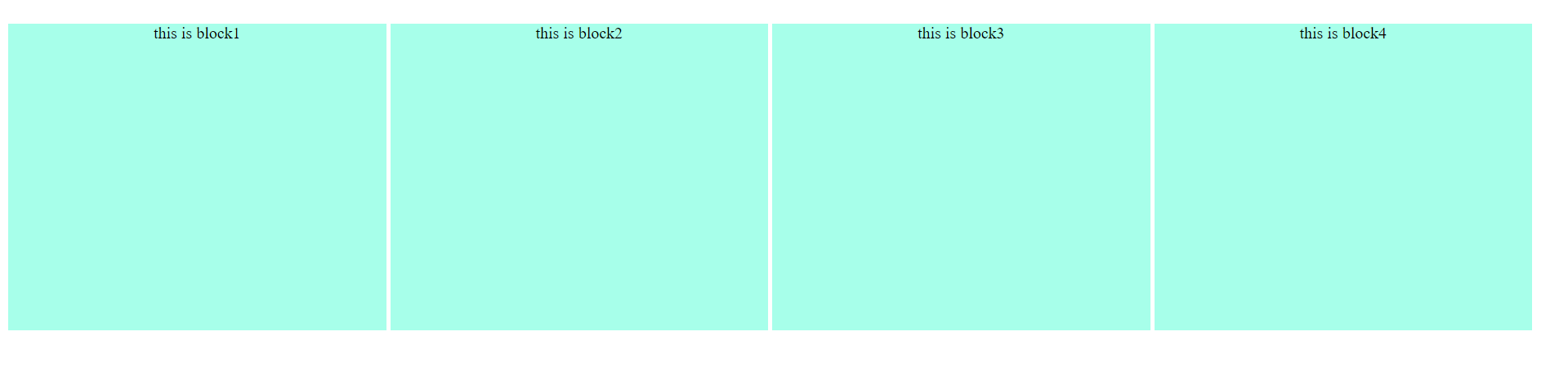
* 1. Inline-block

display: inline-block is a versatile CSS property that allows elements to behave like inline elements in terms of layout (sitting next to each other) while also having the block-like ability to accept width, height, padding, and margin settings. This makes it ideal for creating complex and responsive layouts that require precise control over element dimensions and positioning.



**key Points:**

* **Inline Layout with Block Capabilities**: Elements can sit next to each other while having dimensions set.



* **Alignment and Spacing**: Provides better control over alignment and spacing compared to pure inline or block elements.
* **Responsive Design**: Useful in responsive design for creating adaptable and flexible layouts.

END OF SECTION

**Height and Width**

**List of Contents:**

1. Width

2. Min-width

3. Max-width

4. Height

5. Min-height

6. Max-height

7. Box-sizing

Read the notebook notes

**1. Pseudo Class :**

Module 20

* **Pseudo Classes**
* Gradients
* Transitions
* Animation

**Definition**: Pseudo-class selectors in CSS are used to define the special states of elements. They allow you to apply styles to elements based on their state or position within the document structure, without the need to add classes or IDs.

**Key Points:**

* **State-Based Styling**: Apply styles based on the state of an element (e.g., :hover, :focus, :active).
* **Structural Styling**: Target elements based on their position in the document (e.g., :first-child, :nth-child()).
* **User Interaction**: Change the appearance of elements based on user actions (e.g.,: hover when a user hovers over an element).

Most frequently used CSS Pseudo class selectors.

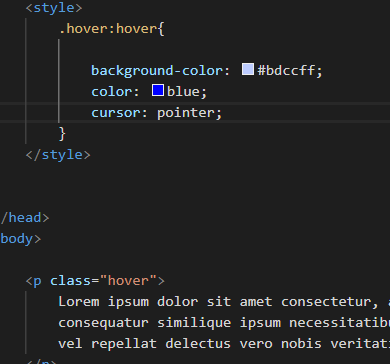
1. **: hover**: Interaction on hover.
2. **: focus**: Interaction when focused.
3. **:** **link**: Unvisited links.

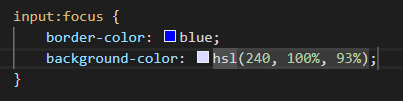
**: visited**: Visited links.

**: active**: Interaction on click.

1. **: first-child**: First child element.
2. **: lang**: Language-specific styling.
3. **: nth-child ()**: Styling elements based on position
4. Hover

***Selector/class/id:hover****{  
 }*

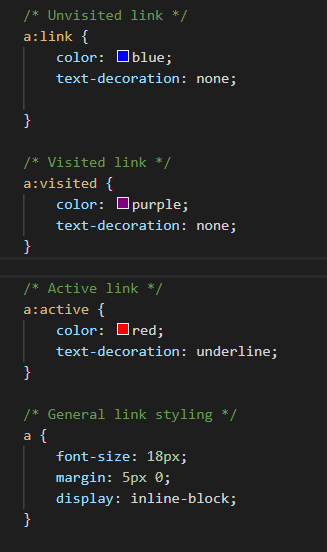


  
2.Focus**:**

  <input type="text">

3.Link:

* **link**: Unvisited links.
* **: visited**: Visited links.
* **: active**: Interaction on click.



* 1. first-child:
  2. nth child ():

<https://github.com/Lambodar2001/Web-Development-/blob/2cf7ddb48565dad1dd5a1d62d170e1356abfb7f5/2.CSS/CSS%209/index1.html>

 Descendant **Combinator**: Targets all levels of descendants.

 Child **Combinator**: Targets only direct children.

 First**-Child**: Targets the first child of a parent. – pseudo

* **Simplest Description**

**First Child (:first-child)**

* **What it does**: Selects the first element inside a parent.
* **Imagine**: You have a list of items. The :first-child selector will pick the very first item in that list.

**nth-child (:nth-child(n))**

* **What it does**: Selects the nth element inside a parent, where n is a number you specify.
* **Imagine**: You have a list of items. The :nth-child(2) selector will pick the second item in that list, :nth-child(3) will pick the third, and so on.

**Examples**

* **:first-child**: Think of a group of boxes. If you apply a style with :first-child, only the first box gets styled.
* **:nth-child(3)**: Think of a group of boxes. If you apply a style with :nth-child(3), only the third box gets styled. You can replace 3 with any number to style the corresponding box.

* here's a list of frequently used nth-child formulas:

1. **Every Even Child (nth-child(even) or nth-child(2n))**
2. **Every Odd Child (nth-child(odd) or nth-child(2n+1))**
3. **Every 3rd Child (nth-child(3n))**
4. **Every 3rd Child Starting from 1 (nth-child(3n+1))**
5. **First Child (nth-child(1))**
6. **Last Child (nth-last-child(1))**
7. **First 5 Children (nth-child(-n+5))**
8. **Children Starting from the 3rd (nth-child(n+3))**
9. **Every 4th Child Starting from 2 (nth-child(4n+2))**
10. **Every 5th Child (nth-child(5n))**

* **Understanding n in the :nth-child Formula**
* **n**: This is a variable that starts at 0 and increments by 1. It can be used to create patterns for selecting elements.

**Common Patterns**

1. **nth-child(n)**:
   * Selects every element (acts like a wildcard, since every element fits the formula).
2. **nth-child(2n)**:
   * Selects every even-numbered element.
   * Pattern: 2, 4, 6, 8, ...
3. **nth-child(2n+1)**:
   * Selects every odd-numbered element.
   * Pattern: 1, 3, 5, 7, ...
4. **nth-child(3n)**:
   * Selects every third element.
   * Pattern: 3, 6, 9, 12, ...
5. **nth-child(3n+1)**:
   * Selects every third element starting from the first.
   * Pattern: 1, 4, 7, 10, ...
6. **nth-child(-n+3)**:
   * Selects the first three elements.
   * Pattern: 1, 2, 3
7. **nth-child(n+3)**:
   * Selects all elements starting from the third.
   * Pattern: 3, 4, 5, 6, ...

**Example Breakdown**

* **nth-child(2n)**:
  + n = 0: 2\*0 = 0 (not selected, as nth-child starts from 1)
  + n = 1: 2\*1 = 2 (selects 2nd child)
  + n = 2: 2\*2 = 4 (selects 4th child)
  + n = 3: 2\*3 = 6 (selects 6th child)
  + ...
* **nth-child(3n+1)**:
  + n = 0: 3\*0 + 1 = 1 (selects 1st child)
  + n = 1: 3\*1 + 1 = 4 (selects 4th child)
  + n = 2: 3\*2 + 1 = 7 (selects 7th child)
  + n = 3: 3\*3 + 1 = 10 (selects 10th child)
  + ...

Using n in the formula allows you to create flexible and powerful patterns for selecting elements based on their order in the document.

1. **Pseudo Elements ::**

Pseudo-elements are like virtual elements you can style in CSS to enhance how content looks or behaves without adding extra HTML. Here's what each does:

1. **::first-line**: Styles the first line of text in a block.
2. **::first-letter**: Styles the first letter of a block of text, like making it bigger or a different color.
3. **::after**: Adds content after an element's content, like inserting icons or text after a paragraph.
4. **::before**: Adds content before an element's content, useful for adding icons or labels before text.
5. **::marker**: Styles list markers, like bullets or numbers in lists.
6. **::selection**: Styles the text that's selected by the user on a webpage, changing its color or background.

<https://github.com/Lambodar2001/Web-Development-/blob/8dbe51f0db322f5cf9495f461abc3ed3a986c1ef/2.CSS/CSS%209/index2.html>

1. **Gradient**

List of Content

1. Gradient

2. Linear gradient

3. Radial gradient

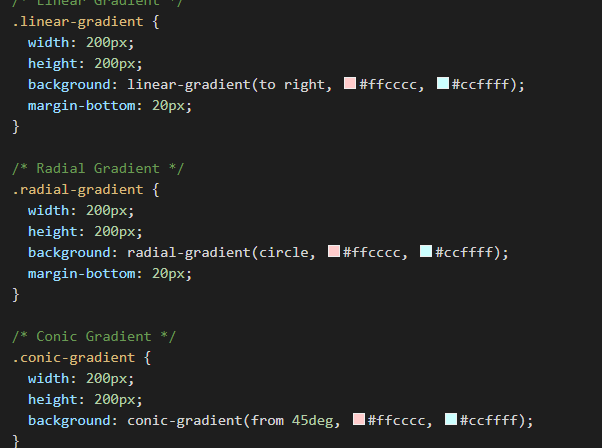
4. Conic gradient

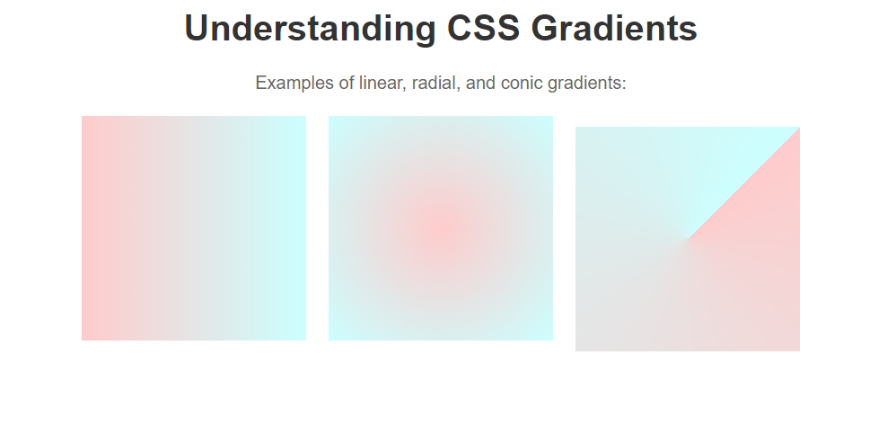
It lets you display smooth transitions between two or more specified colors.

**CSS defines three types of gradients:**

**● Linear Gradients (goes down/up/left/right/diagonally)**

**● Radial Gradients (defined by their center)**

**● Conic Gradients (rotated around a center point)**



1. **Transition**

**1. Transitions**

**2. Tooltips**

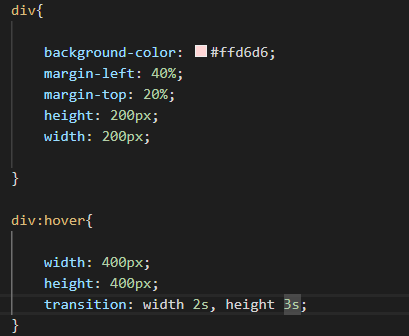
1. **Transition**:

References: [**https://github.com/Lambodar2001/Web-Development-/blob/179208d38401e9d37f7812a089e4b7e30efc7524/2.CSS/CSS%209/index4.html**](https://github.com/Lambodar2001/Web-Development-/blob/179208d38401e9d37f7812a089e4b7e30efc7524/2.CSS/CSS%209/index4.html)

Height and width

Color

Syntax:  
  
transition: propertyManipulating sec ;



**2. CSS Transition Timing Function:**

References: [**https://github.com/Lambodar2001/Web-Development-/blob/792bbc5c775f58db835d634368441bc6e504d9ff/2.CSS/CSS%209/index5.html**](https://github.com/Lambodar2001/Web-Development-/blob/792bbc5c775f58db835d634368441bc6e504d9ff/2.CSS/CSS%209/index5.html)

The transition-timing-function property in CSS controls the speed of an animation. It defines how the intermediate states of the transition are calculated.

**Values:**

1. **linear:** The animation has the same speed from start to finish.
2. **ease:** The default. The animation starts slow, then speeds up, and slows down at the end.
3. **ease-in:** The animation starts slowly and then speeds up.
4. **ease-out:** The animation starts quickly and then slows down.
5. **ease-in-out:** The animation starts slowly, speeds up in the middle, and slows down at the end.
6. **steps(n, direction):** The animation changes in steps. n is the number of steps, and direction can be start or end.
7. **cubic-bezier(x1, y1, x2, y2):** Allows you to define your own timing function. It uses cubic-bezier curves.

**ubic-bezier(x1, y1, x2, y2)**

**Definition:** The cubic-bezier(x1, y1, x2, y2) function in CSS allows you to create a custom timing function for transitions and animations. It defines the speed of the animation using a cubic Bezier curve.

**Parameters:**

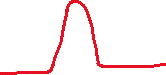
* **x1, y1, x2, y2:** These are four control points that define the shape of the cubic Bezier curve.
  + x1 and x2 must be in the range [0, 1] because they represent time (0% to 100% of the duration).
  + y1 and y2 can be any value, as they represent the progress of the animation.

**How It Works:**

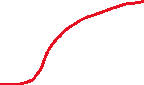
* The cubic Bezier curve is a mathematical formula used to create smooth curves.
* The curve starts at (0,0) and ends at (1,1).
* The points (x1, y1) and (x2, y2) control the shape of the curve.



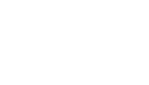
1. **Ease :** **The default. The animation starts slow, then speeds up, and slows down at the end.**



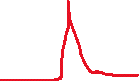
1. **Ease-in ease-in:** The animation starts slowly and then speeds up.



1. **ease-out:** The animation starts quickly and then slows down.



1. **ease-in-out:** The animation starts slowly, speeds up in the middle, and slows down at the end.



**cubic-bezier(x1, y1, x2, y2)**

**Definition:** The cubic-bezier(x1, y1, x2, y2) function in CSS allows you to create a custom timing function for transitions and animations. It defines the speed of the animation using a cubic Bezier curve.

**Parameters:**

* **x1, y1, x2, y2:** These are four control points that define the shape of the cubic Bezier curve.
  + x1 and x2 must be in the range [0, 1] because they represent time (0% to 100% of the duration).
  + y1 and y2 can be any value, as they represent the progress of the animation.

**How It Works:**

* The cubic Bezier curve is a mathematical formula used to create smooth curves.
* The curve starts at (0,0) and ends at (1,1).
* The points (x1, y1) and (x2, y2) control the shape of the curve.

**Example:**

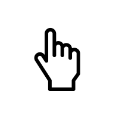
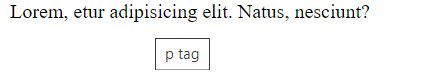
Css

***transition-timing-function: cubic-bezier(0.25, 0.1, 0.25, 1);***

**tooltip:**

A tooltip in CSS is a small pop-up box that appears when you hover over an element. It typically provides additional information about the element or context. You can create tooltips using CSS by *title*



****

1. **Transition**