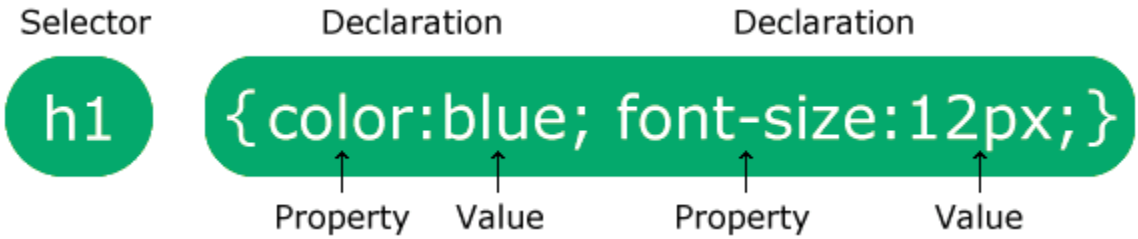
CSS Syntax



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

```
p {
 color: red;
 text-align: center;
}
```

p is a selector in CSS (it points to the HTML element you want to style: <p>).
color is a property, and red is the property value
text-align is a property, and center is the property value

```
/* There 3 ways to implement css
one way is using importing a css file in html head
section
<link rel="stylesheet" href="./abc.css">
Second way is to use <style> tag
   <style>
     h1 {
       color: yellow;
     }
     </style>
By using style attribute
Style attribute is supported with all html element
<div style="background-color: red;">Hi this is a
div</div>
*/
```

CSS Selectors

1. Simple selectors (select elements based on name, id, class)
2. Combinator selectors (select elements based on a specific relationship between them)
3. Pseudo-class selectors (select elements based on a certain state)
4. Pseudo-elements selectors (select and style a part of an element)

5. Attribute selectors (select elements based on an attribute or attribute value)

The CSS id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

```
#para1 {
 text-align: center;
 color: red;
}
```

The CSS class Selector

The class selector selects HTML elements with a specific class attribute.
To select elements with a specific class, write a period (.) character, followed by the class name.

```
.center {
 text-align: center;
 color: red;
}
```

The CSS Universal Selector
The universal selector (*) selects all HTML elements on the page.

```
* {
 text-align: center;
 color: blue;
}
```

The CSS Grouping Selector
The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
h1, h2, .text-center, #id-1 {
```

```
 text-align: center;
 color: red;
}
```

Ways to add CSS

There are three ways of inserting a style sheet:

External CSS: <link rel="stylesheet" href="mystyle.css">

Internal CSS:

```
<style>
body {
 background-color: lightblue;
}
</style>
```

Inline CSS: `<p style="color:red;">This is a paragraph.</p>`

CSS Backgrounds

1. background-color: Specifies the background color of an element.

   ```
   body {
    background-color: lightblue;
   }
   ```

2. background-image: Adds image to background
```
body {
 background-image: url("paper.gif");
}
```
3. background-repeat: Control repeat of images in background image both horizontally and vertically. (Not in Use)
4. Background-attachment: specifies whether the background image should scroll or be fixed.(Not in Use)
5. Background: It can be used as shorthand for all the above properties.

CSS Borders

border-style: The border-style property specifies what kind of border to display.

The following values are allowed:

dotted - Defines a dotted border
dashed - Defines a dashed border
solid - Defines a solid border
And many more….

border-width:
```
p.one {
 border-style: solid;
 border-width: 5px;
}

p.two {
 border-style: solid;
 border-width: medium;
}
```

border-color:
```
p.one {
 border-style: solid;
 border-color: red;
}
```

IMP: to apply specific property to one of the sides of the border you can specify it like this.

```
p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
}
```

Or use a combination of values like

```
p.two {
  border-style: solid;
  border-width: 20px 5px; /* 20px top and bottom, 5px on the sides */
}

p.three {
  border-style: solid;
  border-width: 25px 10px 4px 35px; /* 25px top, 10px right, 4px bottom and 35px left */
}
```

border-radius: rounds of border edges

```
p {
  border: 2px solid red;
  border-radius: 5px;
}
```

CSS Margins

Margins are used to create space around elements, outside of any defined borders.
CSS has properties for specifying the margin for each side of an element:

- margin-top
- margin-right
- margin-bottom
- margin-left

All the margin properties can have the following values:

- auto - the browser calculates the margin
- *length* - specifies a margin in px, pt, cm, etc.
- *%* - specifies a margin in % of the width of the containing element

- inherit - specifies that the margin should be inherited from the parent element

  Negative values allowed: A negative margin on an element allows it to eat up the space of its parent container

Shorthand:

```
p {
  margin: 25px 50px 75px 100px;
}
```

You can set the margin property to auto to horizontally center the element within its container.

The element will then take up the specified width, and the remaining space will be split equally between the left and right margins.

CSS Padding

The CSS padding properties are used to generate space around an element's content, inside of any defined border.

- length - specifies a padding in px, pt, cm, etc.
- % - specifies a padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

CSS Height Width

The height and width properties may have the following values:

- auto - This is default. The browser calculates the height and width
- length - Defines the height/width in px, cm etc.
- % - Defines the height/width in percent of the containing block
- initial - Sets the height/width to its default value
- inherit - The height/width will be inherited from its parent value

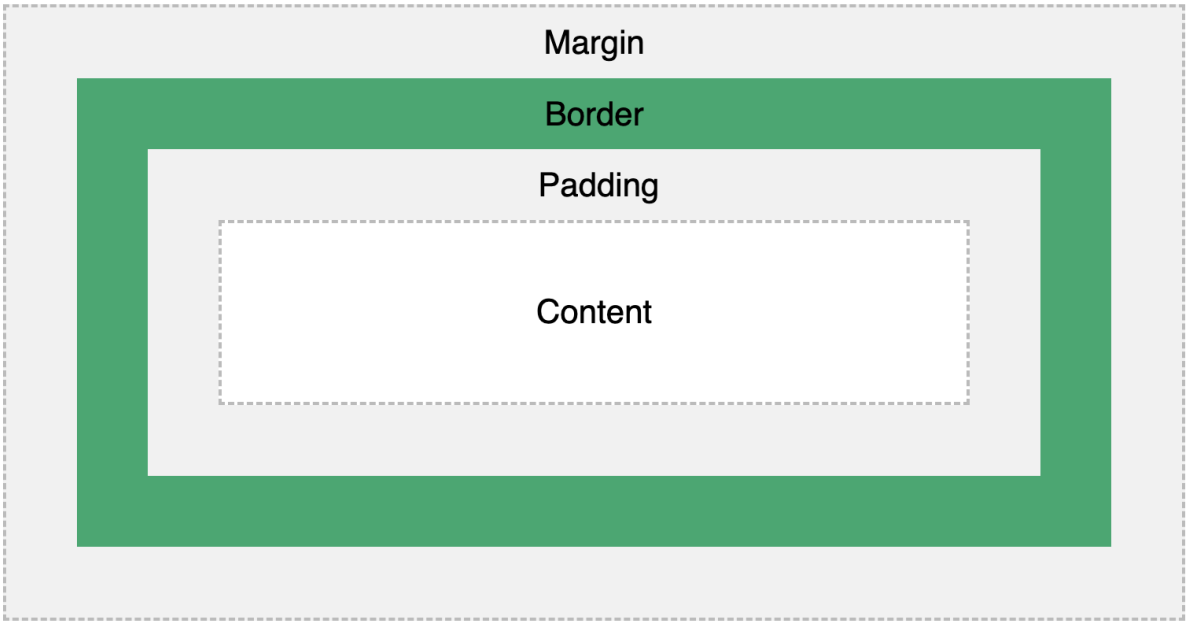
```
div {

 height: 100px;

 width: 500px;

}
```

CSS Box Model



CSS Text Properties:

Property	Role	Syntax	Values Accepted

color	set the color of the text	h1 {
		color: green;
		}
			1. color name
			2. HEX value
			3. RGB value

text-align	set the horizontal alignment of a text	{
		text-align: center;
		}
			1. Center
			2. Left
			3. Right
			4. Justify

Vertical-align (To define another elements property aligned with text)	property sets the vertical alignment of an element	img.a {
		vertical-align: baseline;
		}
			1. Baseline
			2. Text-top
			3. Text-bottom
			4. Sub
			5. Super

text-transform	specify uppercase and lowercase letters	p.uppercase {
		text-transform: uppercase;
		}
			1. Uppercase
			2. Lowercase
			3. Capitalize

letter-spacing	Specify the space between the characters	h1 {
		letter-spacing: 5px;
		}
			PX

line-height	specify the space between lines	p.small {
		line-height: 0.8;
		}
			ratio

CSS display:

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

A **block-level** element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The element is a block-level element.
Examples of block-level elements:

-
- <h1> - <h6>
- <p>
- <form>
- <header>
- <footer>
- <section>

An **inline** element does not start on a new line and only takes up as much width as necessary.

Examples of inline elements:

-
- <a>
- <img>

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them.

CSS Position

1. static
2. relative
3. fixed
4. absolute
5. sticky

static: every element has a static position by default, so the element will stick to the normal page flow. So if there is a left/right/top/bottom/z-index set then there will be no effect on that element.

relative: an element's original position remains in the flow of the document, just like the static value. But now left/right/top/bottom/z-index will work. The positional properties "nudge" the element from the original position in that direction.

absolute: the element is removed from the flow of the document and other elements will behave as if it's not even there whilst all the other positional properties will work on it.

fixed: the element is removed from the flow of the document like absolutely positioned elements. In fact they behave almost the same, only fixed positioned elements are always relative to the document, not any particular parent, and are unaffected by scrolling.

sticky: the element is treated like a relative value until the scroll location of the viewport reaches a specified threshold, at which point the element takes a fixed position where it is told to stick.

inherit: the position value doesn't cascade, so this can be used to specifically force it to, and inherit the positioning value from its parent.

The z-index Property

When elements are positioned, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

```
img {
 position: absolute;
 left: 0px;
 top: 0px;
 z-index: -1;
}
```

Important Concepts

[Pseudo-class]
[Pseudo-elements]
[CSS math functions]

Media Query:

Media Types:

all: Matches all devices
print: Matches documents that are viewed in a print preview or any media that breaks the content up into pages intended to print.
screen: Matches devices with a screen

Feature	Summary	Values	Added
`width`	Defines the widths of the viewport. This can be a specific number (e.g. `400px`) or a range (using `min-width` and `max-width`).	`<length>`	
`height`	Defines the height of the viewport. This can be a specific number (e.g. `400px`) or a range (using `min-height` and `max-height`).	`<length>`	
`aspect-ratio`	Defines the width-to-height aspect ratio of the viewport	`<ratio>`	

`orientation`	The way the screen is oriented, such as tall (`portrait`) or wide (`landscape`) based on how the device is rotated.	`portrait`
		
		`landscape`	

`overflow-block`	Checks how the device treats content that overflows the viewport in the block direction, which can be `scroll` (allows scrolling), `optional-paged` (allows scrolling and manual page breaks), `paged` (broken up into pages), and `none` (not displayed).	`scroll`
		
		`optional-paged`
		
		`paged`
			Media Queries Level 4

`overflow-inline`	Checks if content that overflows the viewport along the inline axis be scrolled, which is either `none` (no scrolling) or `scroll` (allows scrolling).	`scroll`
		
		`none`
			Media Queries Level 4

Media By Html:

```
<link rel="stylesheet" href="small.css" media="(min-width: 20em)" />

<link rel="stylesheet" href="medium.css" media="(min-width: 64em)" />

<link rel="stylesheet" href="large.css" media="(min-width: 90em)" />
```

CSS Grid
CSS flexbox
Combination Selectors
Attribute Selector