

# Unit -1

## CSS

Prepared by:  
Dr. Susheela

# Cascading Style Sheets

---

- CSS is the language we use to style an HTML document.
- CSS provides a set of style rules for defining the layout of HTML documents.
- CSS describes how HTML elements should be displayed.
- HTML largely determines textual content, CSS determines visual structure, layout, and aesthetics. HTML is a markup language, and CSS is a style sheet language.
- CSS makes it easier to enhance the look of the different elements on a web page.

# Need for CSS

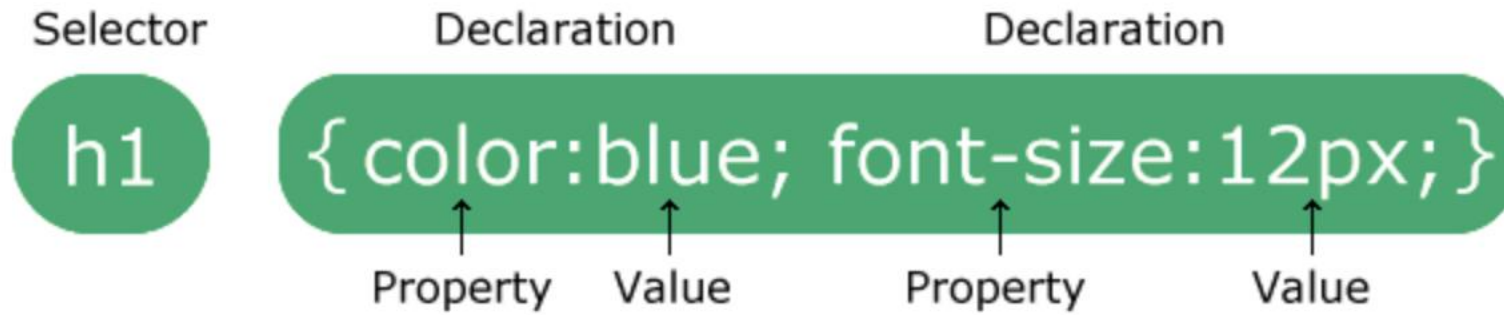
---

1. **Faster Page Speed / Reduce redundancy and increase usability**
  - CSS enables you to use less code. By using CSS, you can use one CSS rule and apply it to all occurrences of a certain tag within an HTML document.
2. **Better User Experience**
  - CSS not only makes web pages easy on the eye, it also allows for user-friendly formatting. When buttons and text are in logical places and well organized, user experience improves.
3. **Quicker Development Time**
  - With CSS, you can apply specific formatting rules and styles to multiple pages with one string of code. One cascading style sheet can be replicated across several website pages.
4. **Easy Formatting Changes /Easier to Maintain the site**
  - If you need to change the format of a specific set of pages, there's no need to change for every individual page. Just edit the corresponding CSS stylesheet and the changes will be applied to all the pages that are using that style sheet. site
5. **Compatibility Across Devices**
  - Whether mobile or tablet, desktop, or even smart TV, CSS combines with HTML to make responsive design possible
6. **Supported by all browsers**
7. **Presentation style of document separated from content**

# Basic Syntax and Structure

---

- A CSS rule consists of a selector and a declaration block.



- 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

# Types of CSS

---

- **Inline CSS –**

- by using the style attribute inside HTML elements

- **Internal CSS –**

- by using a `<style>` element in the `<head>` section

- **External CSS –**

- by using a `<link>` element to link to an external CSS file
- An external style sheet is used to define the style for many HTML pages.
- To use an external style sheet, add a link to it in the `<head>` section of each HTML page:

## Inline CSS

```
<h1 style="color:blue;">A Blue Heading</h1>
<p style="color:red;">A red paragraph.</p>
```

## Internal CSS

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>

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

</body>
</html>
```

## External CSS

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

**styles.css**

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

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

</body>
</html>
```

# Types of Elements

- Block Level Elements:
  - A block-level element always starts on a new line, and the browsers automatically add some space (a margin) before and after the element.
  - A block-level element always takes up the full width available (stretches out to the left and right as far as it can).
  - Two commonly used block elements are: `<p>` and `<div>`.
- Inline Elements
  - An inline element does not start on a new line.
  - An inline element only takes up as much width as necessary.
  - This is a `<span>` element inside a paragraph.
  - Empty HTML Elements are Inline elements
    - HTML elements with no content are called empty elements.
    - `<br>` `<hr>`
- **An inline element cannot contain a block-level element.**



# Types of Elements

- Block Level Elements

<address>	<article>	<aside>	<blockquote>	<canvas>	<dd>	<div>	<dl>	<dt>
<fieldset>	<figcaption>	<figure>	<footer>	<form>	<h1>–<h6>	<header>	<hr>	<li>
<main>	<nav>	<noscript>	<ol>	<p>	<pre>	<section>	<table>	<tfoot>
<ul>	<video>							

- Inline Elements

<a>	<abbr>	<acronym>	<b>	<bdo>	<big>	 	<button>	<cite>
<code>	<dfn>	<em>	<i>	<img>	<input>	<kbd>	<label>	<map>
<object>	<output>	<q>	<samp>	<script>	<select>	<small>	<span>	<strong>
<sub>	<sup>	<textarea>	<time>	<tt>	<var>			

# CSS Selectors

---

There are 6 types of selectors

1. Simple Selector
2. Class Selector
3. Id Selector
4. Nested Selector
5. Universal Selector
6. Group Selector
7. Pseudo Classes Selectors

# Simple Selector – Select by Tag

---

- Selector form is a single element to which the property and value is applied
- For example - Select all <span> elements

span

```
{  
background: DodgerBlue;  
color: #ffffff;  
}
```

<span>Here's a span with some text.</span>

<p>Here's a p with some text.</p>

<span>Here's a span with more text.</span>

Here's a span with some text.

Here's a p with some text.

Here's a span with more text.

# Class Selector

---

- Class Selectors are used when you have a bunch of elements that should receive same styling.
- Using Class Selector, we can assign different styles to the same element.
- Example - `<span>` with `class="sky"` and Elements with `class="code"`

`span.sky`

```
{  
    background: DodgerBlue;  
}
```

`.code`

```
{  
    font-family: Consolas;  
}
```

Here's a span with some text.  
Another `<span>`.

`<span class="sky">Here's a span with some text.</span>`

`<span>`

Another `<span class="code"><span> &lt; span &gt; </span>`

`</span>`

# Id Selector

---

- To define a special style for special element we can use "id Selector".
- Id Selectors are used when you have exactly one element that should receive a certain kind of styling
- Example - Select `<span>` with `id="top"`

`<span id="top">Here's a span with some text.</span>`

`<span>Here's another.</span>`

`span#top`

`{`

`background: DodgerBlue;`

`}`

Here's a span with some text.  
Here's another.

# Universal Selector

---

- \* selector is commonly known as the universal selector
- This selector selects and styles ALL elements
- Example

```
*  
{  
    font-family: Calibri;  
}
```

# Nested Selector: Descendant

---

- <a> inside <div class="items">

div.items a

```
{  
    color: green;  
    font-weight: bold;  
}
```

```
<div class="items">  
  <a href="#">Item1</a>  
  <a href="#">Item2</a>  
  <a href="#">Item3</a>  
  <ul>  
    <li><a href="#">Item4</a></li>  
    <li><a href="#">Item5</a></li>  
    <li><a href="#">Item6</a></li>  
  </ul>  
</div>
```

Item1 Item2 Item3

- Item4
- Item5
- Item6

# Nested Selector: Adjacent Sibling

---

- <p> coming after <p>

p + p

```
{  
    font-style: italic;  
    font-weight: bold;  
}  
<div>  
    <p>I'm a paragraph</p>  
    <p>I am selected!</p>  
</div>  
<p>I am selected!</p>  
<h2>Monkey hair</h2>  
<p>I am NOT selected</p>
```

I'm a paragraph

*I am selected!*

I'm a paragraph

**Monkey hair**

I am NOT selected



# Grouping Selector

---

- You can apply a style to many selectors if you like.
- Just separate the selectors with a comma, as given in the following example:

h1, h2, h3

```
{  
    color: #36C;  
    font-weight: normal;  
    font-size: 20px;  
}
```

# Pseudo Classes Selectors

---

- A way of accessing HTML items that are not a part of the document tree.
- One of its main uses is to control the appearance of unvisited and visited links on a Webpage.
- Types of pseudo-class selectors
  - a:link – an unvisited link
  - a:visited – a visited link
  - a:hover – a link you have your mouse over
- CSS example:

a: hover

{

text-decoration: none;

}

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
a:link {  
  color: green;  
  background-color: transparent;  
}
```

```
a:visited {  
  color: pink;  
  background-color: transparent;  
}
```

```
a:hover {  
  color: red;  
  background-color: transparent;  
}
```

```
a:active {  
  color: yellow;  
  background-color: transparent;  
}  
</style>
```

```
<h2>Link Colors</h2>
```

```
<p>You can change the default colors of links</p>
```

```
<a href="html_images.asp" target="_blank">HTML Images</a>
```

```
</body>
```

```
</html>
```

# HTML Link Colors Using CSS

---

## Link Colors

You can change the default colors of links

HTML Images

## Link Colors

You can change the default colors of links

HTML Images

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

# CSS Length Units

---

- There are two types of length units: absolute and relative.
- Absolute Length Units
  - The absolute length units are fixed, and a length expressed in any of these will appear as exactly that size.
  - Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.
- Relative Length Units
  - Relative length units specify a length relative to another length property.
  - Relative length units scale better between different rendering mediums.

# Absolute Length Units

---

Unit	Description
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px *	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)

# Relative Length Units

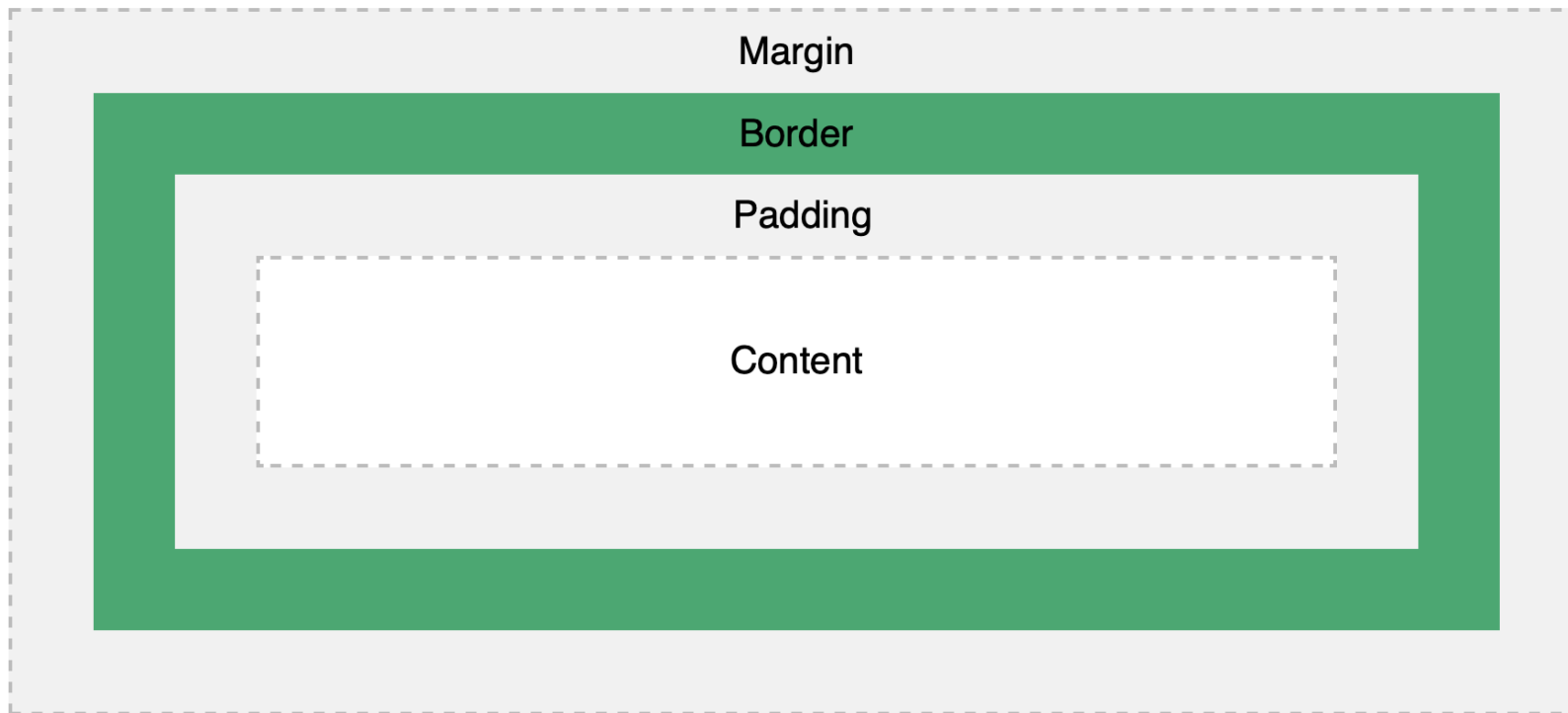
---

Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element

# CSS Box Model

---

- In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around every HTML element.
- It consists of margins, borders, padding, and the actual content. The image below illustrates the box model:
- Explanation of the different parts:
- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent



```
div {  
  width: 300px;  
  border: 15px solid green;  
  padding: 50px;  
  margin: 20px;  
}
```

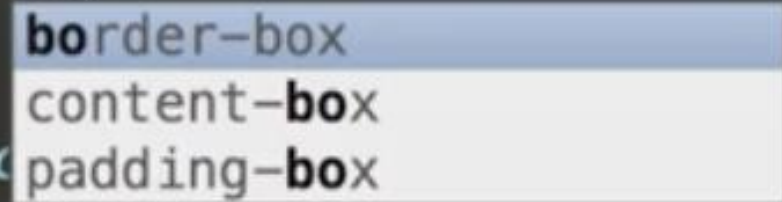


# CSS Box Model

---

- box-sizing property is used to define the size of the Box using width property.
- By default, width property is used to define the width of the content box.
- It consists of margins, borders, padding, and the actual content. The image below illustrates the box model:

```
#box {  
  background-color: blue;  
  padding: 10px 30px 10px 30px;  
  border: 20px solid black;  
  margin: 40px;  
  width: 300px;  
  box-sizing: bo  
}  
#content {  
  background-c
```




# CSS Box Model

---

- **overflow property**  
is used in-case when the content of the box takes more space than the specified size of the box (height & width of the box).
- Possible values: auto, hidden, inherit, scroll, visible
- Default value is visible.

```
#box {  
  background-color: blue;  
  padding: 10px;  
  border: 5px solid black;  
  width: 300px;  
  height: 50px;  
  margin-top: 50px;  
  overflow: |  
}  
#content {  
  background-color: yellow;  
}  
h1 {
```



# CSS - background property

---

- The background property is a shorthand property for:
  - background-color
  - background-image
  - background-position
  - background-size
  - background-repeat
  - background-origin
  - background-clip
  - background-attachment

# background-color

- With CSS, a color is most often specified by:
  - a valid color name - like "red"
  - a HEX value - like "#ff0000"
  - an RGB value - like "rgb(255,0,0)"
- Opacity / Transparency
  - The opacity property specifies the opacity/transparency of an element. It can take a value from 0.0 - 1.0.
  - The lower value, the more transparency

```
<!DOCTYPE html>
<html>
<head> <style>
div {
    background-color: green;
}
div.first {
    opacity: 0.3;
}
div.second {
    opacity: 0.6;
}
</style> </head>
<body>

<h1>Transparent Boxes</h1>

<div class="first"> <h1>opacity 0.3</h1> </div>

<div class="second"> <h1>opacity 0.6</h1> </div>

<div> <h1>opacity 1 (default)</h1> </div>

</body>
</html>
```

## Transparency using RGBA

```
<!DOCTYPE html>
<html>
<head> <style>
div {
  background-color: green;
}
div.first {
  opacity: 0.3;
}
div.second {
  opacity: 0.6;
}
</style> </head>
<body>

<h1>Transparent Boxes</h1>

<div class="first"> <h1>opacity 0.3</h1> </div>

<div class="second"> <h1>opacity 0.6</h1> </div>

<div> <h1>opacity 1 (default)</h1> </div>

</body>
</html>
```

```
<style>
div {
  background: rgb(0, 128, 0);
}

div.first {
  background: rgba(0, 128, 0, 0.3);
}

div.second {
  background: rgba(0, 128, 0, 0.6);
}
```

### Transparent Boxes

opacity 0.3

opacity 0.6

opacity 1 (default)

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("bgdesert.jpg");
}

p {
  background-image: url("paper.gif");
}
</style>
</head>
<body>

<h1>Hello World!</h1>

<p>This page has an image as the background!</p>

</body>
</html>
```

# Hello World!

This page has an image as the background!

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("img_tree.png");
    background-repeat: no-repeat;
    background-position: right top;
    margin-right: 200px;
    background-attachment: fixed;
}
</style>
</head>
<body>
```

# background-image

---

- The background-image property sets one or more background images for an element.
- By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally.
- Possible property values:
  - **url** - The URL to the image. To specify more than one image, separate the URLs with a comma
  - **none** - No background image will be displayed. This is default
  - **initial** - Sets this property to its default value.
  - **inherit** - Inherits this property from its parent element.



# background-position

---

- The background-position property sets the starting position of a background image.
- By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally.
- The top left corner is 0 0 or 0% 0%. The right bottom corner is 100 100 Or 100% 100%.
- Possible values:
  - **left top, left center, left bottom, right top, right center, right bottom, center top, center center, center bottom** - If you only specify one keyword, the other value will be "center"
  - **x% y%** - x is the horizontal position and y is the vertical position.
  - **xpos ypos** – xpos is the horizontal position and ypos is the vertical position.
  - **initial** - Sets this property to its default value.
  - **inherit** - Inherits this property from its parent element.

**background-position:** 50% 50%;

**background-position:** bottom right;

**background-position:** 50px 150px;

# background-size

---

- The background-size property specifies the size of the background images.
- Possible property values:
  - **auto** – Default value (original size)
  - **length** - Sets the width and height of the background image. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto".
  - **percentage** – Sets the width and height of the background image in percent of the parent element. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto"
  - **cover** - Resize the background image to cover the entire container, even if it has to stretch the image or cut a little bit off one of the edges
  - **contain** – Resize the background image to make sure the image is fully visible
  - **initial** – Sets this property to its default value
  - **inherit** – Inherits this property from its parent element.
- **background-size: auto;**
- **background-size: 75% 50%;**
- **background-size: cover;**

# background-repeat

---

- The background-repeat property sets if/how a background image will be repeated.
- By default, a background-image is repeated both vertically and horizontally.

Value	Description
repeat	The background image is repeated both vertically and horizontally. The last image will be clipped if it does not fit. This is default
repeat-x	The background image is repeated only horizontally
repeat-y	The background image is repeated only vertically
no-repeat	The background-image is not repeated. The image will only be shown once
space	The background-image is repeated as much as possible without clipping. The first and last image is pinned to either side of the element, and whitespace is distributed evenly between the images
round	The background-image is repeated and squished or stretched to fill the space (no gaps)
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

# background-attachment

---

- The background-attachment property sets whether a background image scrolls with the rest of the page, or is fixed.

Value	Description
scroll	The background image will scroll with the page. This is default
fixed	The background image will not scroll with the page
local	The background image will scroll with the element's contents
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

# background-origin

---

- The background-origin property specifies the origin position (the background positioning area) of a background image.

Value	Description
padding-box	Default value. The background image starts from the upper left corner of the padding edge
border-box	The background image starts from the upper left corner of the border
content-box	The background image starts from the upper left corner of the content
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Note: This property has no effect if background-attachment is "fixed".

# background-clip

---

- The background-clip property defines how far the background (color or image) should extend within an element.

Value	Description
border-box	Default value. The background extends behind the border
padding-box	The background extends to the inside edge of the border
content-box	The background extends to the edge of the content box
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

# Manipulating Texts

---

- Text Color

- The color property is used to set the color of the text. The color is specified by:
- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
body {
```

```
  color: blue;
```

```
}
```

```
h1 {
```

```
  color: green;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1>This is heading 1</h1>
```

```
<p>This is an ordinary paragraph. Notice that this text is blue. The  
default text color for a page is defined in the body selector.</p>
```

```
<p>Another paragraph.</p>
```

```
</body>
```

```
</html>
```

## This is heading 1

This is an ordinary paragraph. Notice that this text is blue. The default text color for a page is defined in the body selector.

Another paragraph.

# Text Alignment and Text Direction

---

- `text-align`:
  - used to set the horizontal alignment of a text
  - center, left, right, justify, initial, inherit
- `text-align-last`:
  - specifies how to align the last line of a text
  - auto, left, right, center, justify, start, end, initial, inherit
- `vertical-align`:
  - sets the vertical alignment of an element.
  - Baseline, text-top, text-bottom, sub, super



# CSS Text

---

- color
- background-color
- text-align
- text-align-last
- direction
- unicode-bidi
- vertical-align

# CSS Fonts

---

1. Font Family
2. Font Style
3. Font Weight
4. Font Variant
5. Font Size
6. Font Color
7. Font Shorthand

# CSS Font Families

---

In CSS there are five generic font families:

- 1.Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
- 2.Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
- 3.Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
- 4.Cursive** fonts imitate human handwriting.
- 5.Fantasy** fonts are decorative/playful fonts.

Generic Font Family	Examples of Font Names
Serif	Times New Roman Georgia Garamond
Sans-serif	Arial Verdana Helvetica
Monospace	Courier New Lucida Console Monaco
Cursive	<i>Brush Script MT</i> <i>Lucida Handwriting</i>
Fantasy	<b>COPPERPLATE</b> Papyrus

```
<!DOCTYPE html>
<html>
<head>
<style>
.p1 {
  font-family: "Times New Roman", Times, serif;
}

.p2 {
  font-family: Arial, Helvetica, sans-serif;
}

.p3 {
  font-family: "Lucida Console", "Courier New", monospace;
}
</style>
</head>
<body>

<h1>CSS font-family</h1>
<p class="p1">This is a paragraph, shown in the Times New Roman font.</p>
<p class="p2">This is a paragraph, shown in the Arial font.</p>
<p class="p3">This is a paragraph, shown in the Lucida Console font.</p>

</body>
</html>
```

## CSS font-family

This is a paragraph, shown in the Times New Roman font.

This is a paragraph, shown in the Arial font.

This is a paragraph, shown in the Lucida Console font.

# CSS Font Style

---

- The font-style property is mostly used to specify italic text.
- This property has three values:
  - normal - The text is shown normally
  - italic - The text is shown in italics
  - oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

```
p.normal {  
    font-style: normal;  
}
```

```
p.italic {  
    font-style: italic;  
}
```

```
p.oblique {  
    font-style: oblique;  
}
```

# CSS Font Weight

---

- The font-weight property specifies the weight of a font: normal or bold

```
p.normal {  
    font-weight: normal;  
}
```

```
p.thick {  
    font-weight: bold;  
}
```

# CSS Font Variant

---

- The font-variant property specifies whether or not a text should be displayed in a small-caps font.
- In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

```
p.normal {  
    font-variant: normal;  
}
```

```
p.small {  
    font-variant: small-caps;  
}
```



```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  font-variant: normal;
}

p.small {
  font-variant: small-caps;
}
</style>
</head>
<body>
```

## The font-variant property

My name is Susheela.

MY NAME IS SUSHEELA.

```
<h1>The font-variant property</h1>
```

```
<p class="normal">My name is Susheela.</p>
```

```
<p class="small">My name is Susheela.</p>
```

```
</body>
```

```
</html>
```

# CSS Font Variant

---

- The font-size property sets the size of the text.
- Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.
- The font-size value can be an absolute, or relative size.

```
h1 {  
    font-size: 40px;  
}  
h2 {  
    font-size: 1.875em; /* 30px/16=1.875em */  
}  
body {  
    font-size: 100%;  
}  
p {  
    font-size: 10vw;  
}
```

# CSS Font with Shorthand Declaration

---

- The font property is a shorthand property for:
  - font-style
  - font-variant
  - font-weight
  - font-size/line-height
  - font-family
- The font-size and font-family values are required. If one of the other values is missing, their default value are used.

Note: The line-height property sets the space between lines.

- if you specify background-color specifically and then you go ahead and override it with the background, but you don't specify the color or you don't specify whatever the specific property is, whether it's background-image or background-repeat or background-position and so on.
- If you specify font without any dash subproperty, whatever properties that are inherited with a dash will be overridden unless you actually call them out directly, straight in the font property.

```
<!DOCTYPE html>
<html>
<head>
<style>
p.a {
  font: 15px Arial, sans-serif;
}

p.b {
  font: italic small-caps bold 12px/30px Georgia, serif;
}
</style>
</head>
<body>

<h1>The font shorthand Property</h1>

<p class="a">This is a paragraph. The font size is set to 15 pixels, and the font family is
Arial.</p>

<p class="b">This is a paragraph. The font is set to italic and bold, with small-caps (all
lowercase letters are converted to uppercase). The font size is set to 12 pixels, the line height
is set to 30 pixels, and the font family is Georgia.</p>

</body>
</html>
```

# The font shorthand Property

This is a paragraph. The font size is set to 15 pixels, and the font family is Arial.

*THIS IS A PARAGRAPH. THE FONT IS SET TO ITALIC AND BOLD, WITH SMALL-CAPS (ALL LOWERCASE LETTERS ARE CONVERTED TO UPPERCASE). THE FONT SIZE IS SET TO 12  
PIXELS, THE LINE HEIGHT IS SET TO 30 PIXELS, AND THE FONT FAMILY IS GEORGIA.*

# CSS Border

---

- The "border-style" property specifies what kind of border to display.
- **dotted** - Defines a dotted border
- **dashed** - Defines a dashed border
- **solid** - Defines a solid border
- **double** - Defines a double border
- **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
- **none** - Defines no border
- **hidden** - Defines a hidden border

```
<!DOCTYPE html>
<html>
<head>
<style>
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid}
</style>
</head>
```

```
<body>

<h2>The border-style Property</h2>
<p>This property specifies what kind of border to display:</p>

<p class="dotted">A dotted border.</p>
<p class="dashed">A dashed border.</p>
<p class="solid">A solid border.</p>
<p class="double">A double border.</p>
<p class="groove">A groove border.</p>
<p class="ridge">A ridge border.</p>
<p class="inset">An inset border.</p>
<p class="outset">An outset border.</p>
<p class="none">No border.</p>
<p class="hidden">A hidden border.</p>
<p class="mix">A mixed border.</p>

</body>
</html>
```

# The border-style Property

This property specifies what kind of border to display:

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border.

A ridge border.

An inset border.

An outset border.

No border.

A hidden border.

A mixed border.



# Padding

---

The CSS padding properties are used to generate space around an element's content, inside of any defined borders. If the padding property has four values:

• **padding: 25px 50px 75px 100px;**

- top padding is 25px
- right padding is 50px
- bottom padding is 75px
- left padding is 100px

Property	Description
<u>padding</u>	A shorthand property for setting all the padding properties in one declaration
<u>padding-bottom</u>	Sets the bottom padding of an element
<u>padding-left</u>	Sets the left padding of an element
<u>padding-right</u>	Sets the right padding of an element
<u>padding-top</u>	Sets the top padding of an element



```
<!DOCTYPE html>
<html>
<head>
<style>
div.ex1 {
  width: 300px;
  background-color: yellow;
}
div.ex2 {
  width: 300px;
  padding: 25px;
  background-color: lightblue;
}
</style>
</head>
<body>
<h2>Padding and element width</h2>
<div class="ex1">This div is 300px wide.</div>
<br>
<div class="ex2">The width of this div is 350px, even though it is
defined as 300px in the CSS.</div>
</body>
</html>
```

## Padding and element width

This div is 300px wide.

The width of this div is 350px, even though it is defined as 300px in the CSS.

```
div {
  padding: 25px 50px 75px 100px;
}
```

# CSS Lists

---

- The CSS list properties allow you to:
  - Set different list item markers for ordered lists
  - Set different list item markers for unordered lists
  - Set an image as the list item marker
  - Add background colors to lists and list items
- CSS Lists Properties
  - **list-style-type** - specifies the type of list item marker ([circle](#), [square](#), [upper-roman](#), [lower-alpha](#)).
  - **list-style-image** - specifies an image as the list item marker.
  - **list-style-position** - specifies the position of the list-item markers (bullet points).
    - "**list-style-position: outside;**" means that the bullet points will be outside the list item.
    - "**list-style-position: inside;**" means that the bullet points will be inside the list item.
  - **list-style-type:none** - can also be used to remove the markers/bullets. Default margin and padding will remain in the list. To remove this, add `margin:0` and `padding:0` to `<ul>` or `<ol>`

# CSS List - Shorthand property

---

- The list-style property is a shorthand property. It is used to set all the list properties in one declaration:

```
ul {  
  list-style: square inside url("gehu.jpg");  
}
```

- When using the shorthand property, the order of the property values are:
  - list-style-type (if a list-style-image is specified, the value of this property will be displayed if the image for some reason cannot be displayed)
  - list-style-position (specifies whether the list-item markers should appear inside or outside the content flow)
  - list-style-image (specifies an image as the list item marker)
  - If one of the property values above is missing, the default value for the missing property will be inserted, if any.

```
<!DOCTYPE html>
<html>
<head>
<style>
ol {
  background: #ff9999;
  padding: 20px;
}

ul {
  background: #3399ff;
  padding: 20px;
}

ol li {
  background: #ffe5e5;
  color: darkred;
  padding: 5px;
  margin-left: 35px;
}

ul li {
  background: #cce5ff;
  color: darkblue;
  margin: 5px;
}
</style>
</head>
```

```
<body>

<h1>Styling Lists With Colors</h1>

<ol>
  <li>Coffee</li>
  <li>Tea</li>
</ol>

<ul>
  <li>Coffee</li>
  <li>Tea</li>
</ul>

</body>
</html>
```

## Styling Lists With Colors

1. Coffee
2. Tea

- Coffee
- Tea

# Position Property

---

- The position property specifies the type of positioning method used for an element.
- There are five different position values:
  - static
  - relative
  - fixed
  - absolute
  - sticky

Property	Description
<u>bottom</u>	Sets the bottom margin edge for a positioned box
<u>clip</u>	Clips an absolutely positioned element
<u>left</u>	Sets the left margin edge for a positioned box
<u>position</u>	Specifies the type of positioning for an element
<u>right</u>	Sets the right margin edge for a positioned box
<u>top</u>	Sets the top margin edge for a positioned box

## position: static

---

- HTML elements are positioned static by default
- Static positioned elements are not affected by the top, bottom, left, and right properties.

## position: relative

---

- HTML element is positioned relative to its normal position. Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.
- Other content will not be adjusted to fit into any gap left by the element

# position: fixed

---

- HTML element is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.
- A fixed element does not leave a gap in the page where it would normally have been located.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.static {
  position: static;
  border: 3px solid #73AD21;
}
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
</style>
</head>
```

```
<h2>position: static;</h2>
<div class="static">
This div element has position: static;
</div>
<h2>position: relative;</h2>
<div class="relative">
This div element has position: relative;
</div>
<div class="fixed">
This div element has position: fixed;
</div>
</body>
</html>
```

**position: static;**

This div element has position: static;

**position: relative;**

This div element has position: relative;

shot

This div element has position: fixed;



# position: absolute

---

- An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}
```

```
div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
```

```
</style>
</head>
<body>
```

```
<h2>position: absolute;</h2>
```

```
<div class="relative">This div element has position: relative;
  <div class="absolute">This div element has position: absolute;</div>
</div>
```

```
</body>
</html>
```

## position: absolute;

This div element has position: relative;

This div element has position: absolute;

# position: sticky

---

- An element with `position: sticky;` is positioned based on the user's scroll position.
- A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like `position: fixed`).

```
<head>
<style>
div.sticky {
  position: -webkit-sticky;
  position: sticky;
  top: 0;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<p>Try to <b>scroll</b> inside this frame to understand how sticky
positioning works.</p>
```

```
<div class="sticky">I am sticky!</div>
```

```
<div style="padding-bottom:2000px">
```

```
  <p>In this example, the sticky element sticks to the top of the
  page (top: 0), when you reach its scroll position.</p>
```

```
  <p>Scroll back up to remove the stickyness.</p>
```

```
</div>
```

```
</body>
```

```
</html>
```

Try to **scroll** inside this frame to understand how sticky positioning works.

I am sticky!

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

Scroll back up to remove the stickyness.

Output After Scrolling

I am sticky!

Scroll back up to remove the stickyness.

# Bootstrap

---

- Bootstrap is an open-source and free CSS framework that helps in directing a responsive device-friendly mobile-first front-end webpage development tool.
- Bootstrap mainly includes CSS (Cascading Style Sheets) and an optional JavaScript-supported design template (plug-ins) that deals with typography, buttons, forms, and other user interface components. This Bootstrap framework helps rapid web development and supports developers in creating responsive web pages.
- Twitter Blueprint was the first name for Bootstrap and was developed on Twitter by Mr. Mark Otto and Jacob Thornton. It was released as an open-source product on GitHub in August 2011. The framework is primarily built to encourage design uniformity and reliability of web pages across applications. Before its existence, developers used various external libraries to perform responsive web development, leading to incompatibilities in web development and heavy maintenance burdens.
- <https://getbootstrap.com>



[Docs](#) [Examples](#) [Icons](#) [Themes](#) [Blog](#)

⌘ K



v5.3 ▾



New! Never-Ending Support for Bootstrap →



# Build fast, responsive sites with Bootstrap

Powerful, extensible, and feature-packed frontend toolkit. Build and customize with Sass, utilize prebuilt grid system and components, and bring projects to life with powerful JavaScript plugins.



## Getting started

[Introduction](#)[Download](#)[Contents](#)[Browsers & devices](#)[JavaScript](#)[Webpack](#)[Parcel](#)[Vite](#)[Accessibility](#)[RFS](#)[RTL](#)[Contribute](#)

## Customize

[Overview](#)[Sass](#)[Options](#)[Color](#)[Color modes](#)

# Download

[View on GitHub](#)

Download Bootstrap to get the compiled CSS and JavaScript, source code, or include it with your favorite package managers like npm, RubyGems, and more.



Get 10 free Adobe Stock photos. Start downloading amazing royalty-free stock photos today.

ads via Carbon

## On this page

### Compiled CSS and JS

[Source files](#)[Examples](#)[CDN via jsDelivr](#)[Alternative CDNs](#)

### Package managers

[npm](#)[yarn](#)[RubyGems](#)[Composer](#)[NuGet](#)

## Compiled CSS and JS

Download ready-to-use compiled code for **Bootstrap v5.3.3** to easily drop into your project, which includes:

- Compiled and minified CSS bundles (see [CSS files comparison](#))
- Compiled and minified JavaScript plugins (see [JS files comparison](#))

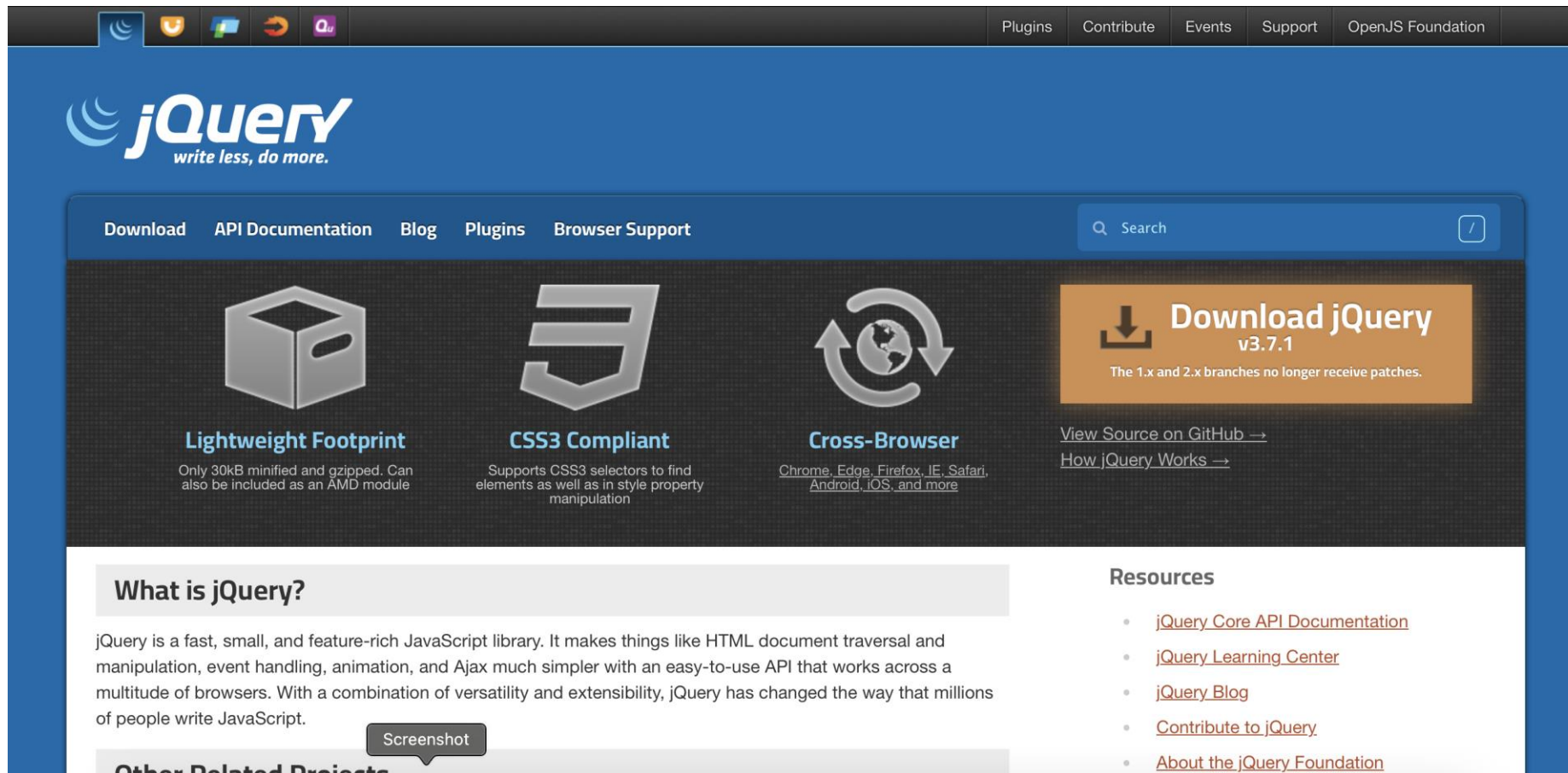
This doesn't include documentation, source files, or any optional JavaScript dependencies like Popper.

[Download](#)

# Benefits of Bootstrap

---

- bootstrap.js file depends on jQuery.
- In order to get this file, go to <https://jquery.com> and click on this Download link.





# Benefits of Bootstrap

---

- Browser supportive: Every browser supports this Bootstrap Framework.
- Mobile-first approach: The Bootstrap framework has a preexisting mobile-first style all through the library and not as separate files.
- Simple and easy to start: If you know HTML and CSS, you can quickly start working with Bootstrap, and its documentation is available on the official site.
- Responsive design and looks: Web pages designed using the Bootstrap framework have responsive CSS that can adjust to the screen size of large desktops, notebooks, tablets, and mobiles.
- Easy customization: It provides some built-in components and functionalities that are easy to customize.
- Clean interface or Developers: The bootstrap framework provides a new and consistent result for building user interfaces on web pages.
- It is an open-source framework with web-based customization.
- **Benefits of Bootstrap Framework**
- It produces fewer cross-browser bugs.
- It is a consistent framework supported by all web browsers and CSS-based compatibility improvements.
- It is a lightweight and hence widely used framework for creating responsive sites.
- It's easily customizable.
- It has a simple and effective grid system.
-

# Media Query

- Media Query is used to design a responsive device-friendly front-end webpage.

# Program (Media Query)

```
<!doctype HTML>
<html>
<head>
  <style>
    div
    {
      font-size: 30px;
      border: 5px double blue;
      margin: 10px;
    }
    div:nth-child(2)
    {
      font: 40px rgb(0, 33, 17);
      background: blue;
      background-color: powderblue;
    }
    div > h3:nth-child(2)
    {
      background: yellow;
      font-size: 20px;
    }
  </style>
</head>
<body>
```

```
  @media (min-width: 500px) and (max-width: 1000px)
  {
    div
    {
      font-size: 80%;
    }
    div > h3:nth-child(2)
    {
      background: yellow;
      font-size: 90%;
    }
  }
</style>
</head>
<body>

  <div>
    This is 1st div element

    <h3> This is First heading </h3>
    <h3> This is Second heading </h3>
    <h3> This is Third heading </h3>
    <h3> This is Forth heading </h3>
  </div>

  <div>
    This is 2nd div element
  </div>

</body>
</html>
```

This is 1st div element

**This is First heading**

**This is Second heading**

**This is Third heading**

**This is Forth heading**

This is 2nd div element



OUTPUT (When Page width is  
in between 500 – 1000px)

OUTPUT (When Page width  
either less than 500 or greater  
than 1000px)



This is 1st div element

**This is First heading**

**This is Second heading**

**This is Third heading**

**This is Forth heading**

This is 2nd div element

# Positioning Elements by Floating

- Most of the UIs today, web UIs today that are made, are made by floating the elements.
- width & float property should be used together to create a two-column flexible design where the columns will be flexible as we expand and contract the browser.
- float property is used to float elements.
- When you float elements, it takes them out of the regular document flow, which will disturb the appearance of the content in web page.
- In order to correct it, we need to tell the browser that when it comes to that element, the browser should resume the regular document flow by using the clear property.
- clear property is used to clear the side (left or right or both) of the element.
- clear: left means nothing should be allowed to be floating to the left of that element in which we had specified this property.
- Designing a 2-column flexible design
  - property: value for both column element (div or p)
    - width: 50%; float: left;
  - property: value for the main element
    - clear: left.

# References

---

- <https://www.w3schools.com/css/>
- <https://devmountain.com/blog/what-is-css-and-why-use-it/>
- [https://www.geekinterview.com/question\\_details/35100](https://www.geekinterview.com/question_details/35100)
- <https://m2.material.io/design/introduction>
- <https://www.code-boost.com/scss-setup/>