

CSS

CSS stands for Cascading Style Sheets. CSS is a standard style sheet language used for describing the presentation (i.e. the layout and formatting) of the web pages.

Prior to CSS, nearly all of the presentational attributes of HTML documents were contained within the HTML markup (specifically inside the HTML tags); all the font colors, background styles, element alignments, borders and sizes had to be explicitly described within the HTML.

As a result, development of the large websites became a long and expensive process, since the style information were repeatedly added to every single page of the website.

To solve this problem CSS was introduced in 1996 by the World Wide Web Consortium (W3C), which also maintains its standard. CSS was designed to enable the separation of presentation and content. Now web designers can move the formatting information of the web pages to a separate style sheet which results in considerably simpler HTML markup, and better maintainability.

CSS3 is the latest version of the CSS specification. CSS3 adds several new styling features and improvements to enhance the web presentation capabilities.

Advantages of Using CSS

The biggest advantage of CSS is that it allows the separation of style and layout from the content of the document. Here are some more advantages, why one should start using CSS?

CSS Save Lots of Time — CSS gives lots of flexibility to set the style properties of an element. You can write CSS once; and then the same code can be applied to the groups of HTML elements, and can also be reused in multiple HTML pages.

Easy Maintenance — CSS provides an easy means to update the formatting of the documents, and to maintain the consistency across multiple documents. Because the content of the entire set of web pages can be easily controlled using one or more style sheets.

Pages Load Faster — CSS enables multiple pages to share the formatting information, which reduces complexity and repetition in the structural contents of the documents. It significantly reduces the file transfer size, which results in a faster page loading.

Superior Styles to HTML — CSS has much wider presentation capabilities than HTML and provide much better control over the layout of your web pages. So you can give far better look to your web pages in comparison to the HTML presentational elements and attributes.

Multiple Device Compatibility — CSS also allows web pages to be optimized for more than one type of device or media. Using CSS the same HTML document can be presented in different viewing styles for different rendering devices such as desktop, cell phones, etc.

CSS can either be attached as a separate document or embedded in the HTML document itself. There are three methods of including CSS in an HTML document:

Inline styles — Using the style attribute in the HTML start tag.

Embedded styles — Using the <style> element in the head section of a document.

External style sheets — Using the <link> element, pointing to an external CSS file.

Inline Styles

Inline styles are used to apply the unique style rules to an element by putting the CSS rules directly into the start tag. It can be attached to an element using the style attribute.

The style attribute includes a series of CSS property and value pairs. Each "property: value" pair is separated by a semicolon (;), just as you would write into an embedded or external style sheets. But it needs to be all in one line i.e. no line break after the semicolon,

Example

```
<h1 style="color:red; font-size:30px;">This is a heading</h1>
<p style="color:green; font-size:22px;">This is a paragraph.</p>
<div style="color:blue; font-size:14px;">This is some text content.</div>
```

Embedded Style Sheets

Embedded or internal style sheets only affect the document they are embedded in.

Embedded style sheets are defined in the <head> section of an HTML document using the <style> element. You can define any number of <style> elements in an HTML document but they must appear between the <head> and </head> tags.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>My HTML Document</title>
  <style>
    body { background-color: YellowGreen; }
    p { color: #fff; }
  </style>
</head>
<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph of text.</p>
</body>
</html>
```

External Style Sheets

An external style sheet is ideal when the style is applied to many pages of the website.

An external style sheet holds all the style rules in a separate document that you can link from any HTML file on your site. External style sheets are the most flexible because with an external style sheet, you can change the look of an entire website by changing just one file.

You can attach external style sheets in two ways — linking and importing.

Example

Style.css

```
body { background: lightyellow; font: 18px Arial, sans-serif; }
h1 { color: orange; }
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>My HTML Document</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph of text.</p>
</body>
</html>
```

Importing External Style Sheets

The @import rule is another way of loading an external style sheet. The @import statement instructs the browser to load an external style sheet and use its styles.

You can use it in two ways. The simplest is within the header of your document. Note that, other CSS rules may still be included in the <style> element.

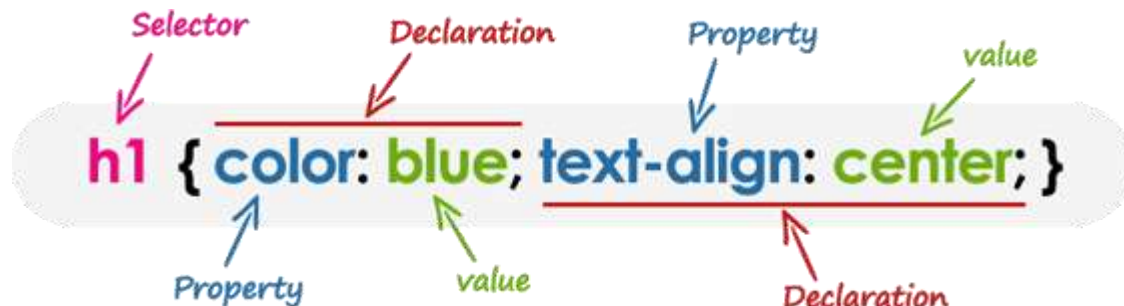
Example

```
<style>
  @import url("style.css");
  p {
    color: blue;
    font-size: 16px;
  }
</style>
```

Understanding CSS Syntax

A CSS stylesheet consists of a set of rules that are interpreted by the web browser and then applied to the corresponding elements such as paragraphs, headings, etc.

A CSS rule have two main parts, a selector and one or more declarations:



The selector specifies which element or elements in the HTML page the CSS rule applies to.

Whereas, the declarations within the block determines how the elements are formatted on a webpage. Each declaration consists of a property and a value separated by a colon (:) and ending with a semicolon (;), and the declaration groups are surrounded by curly braces {}.

The property is the style attribute you want to change; they could be font, color, background, etc. Each property has a value, for example color property can have value either blue or #0000FF etc

Writing Comments in CSS

Comments are usually added with the purpose of making the source code easier to understand. It may help other developer (or you in the future when you edit the source code) to understand what you were trying to do with the CSS. Comments are significant to programmers but ignored by browsers.

A CSS comment begins with `/*`, and ends with `*/`

Example

```
/* This is a CSS comment */
h1 {
  color: blue;
  text-align: center;
}
/* This is a multi-line CSS comment
that spans across more than one line */
p {
  font-size: 18px;
  text-transform: uppercase;
}
```

CSS Selectors

A CSS selector is a pattern to match the elements on a web page. The style rules associated with that selector will be applied to the elements that match the selector pattern.

Selectors are one of the most important aspects of CSS as they allow you to target specific elements on your web page in various ways so that they can be styled.

Several types of selectors are available in CSS

Universal Selector

The universal selector, denoted by an asterisk (*), matches every single element on the page.

The universal selector may be omitted if other conditions exist on the element. This selector is often used to remove the default margins and paddings from the elements for quick testing purpose

Example

```
* {  
  margin: 0;  
  padding: 0;  
}
```

Element Type Selectors

An element type selector matches all instance of the element in the document with the corresponding element type name.

Example

```
p {  
  color: blue;  
}
```

Id Selectors

The id selector is used to define style rules for a single or unique element.

The id selector is defined with a hash sign (#) immediately followed by the id value.

Example

```
#error {  
  color: red;  
}
```

Class Selectors

The class selectors can be used to select any HTML element that has a class attribute. All the elements having that class will be formatted according to the defined rule.

The class selector is defined with a period sign (.) immediately followed by the class value.

Example

```
.blue {  
  color: blue;  
}
```

```
p.blue {  
  color: blue;  
}
```

Child Selectors

A child selector is used to select only those elements that are the direct children of some element.

A child selector is made up of two or more selectors separated by a greater than symbol (>). You can use this selector, for instance, to select the first level of list elements inside a nested list that has more than one level.

Example

```
ul > li {  
  list-style: square;  
}  
ul > li ol {  
  list-style: none;  
}
```

Grouping Selectors

Often several selectors in a style sheet share the same style rules declarations. You can group them into a comma-separated list to minimize the code in your style sheet. It also prevents you from repeating the same style rules over and over again.

Example

```
h1, h2, h3 {  
  font-weight: normal;  
}  
h1 {  
  font-size: 36px;  
}  
h2 {  
  font-size: 28px;  
}  
h3 {  
  font-size: 22px;  
}
```

CSS Color

The color property defines the text color (foreground color in general) of an element. For instance, the color property specified in the body selector defines the default text color for the whole page

Example

```
body {  
    color: #ff5722;  
}
```

Colors in CSS most often specified in the following formats:

a color name - like "red"

a HEX value - like "#ff0000"

an RGB value - like "rgb(255, 0, 0)"

Example : Color Name

```
h1 {  
    color: red;  
}  
p {  
    color: purple;  
}
```

Example : HEX color

```
h1 {  
    color: #ffa500;  
}  
p {  
    color: #00ff00;  
}
```

Example : RGB Color

```
h1 {  
    color: rgb(255, 165, 0);  
}  
p {  
    color: rgb(0, 255, 0);  
}
```

Effect of Color Property on Borders and Outlines

The color property is not just for text content, but for anything in the foreground that takes a color value. For instance, if border-color or outline-color value hasn't been defined explicitly for the element, the color value will be used instead.

```
p.one {  
    color: #0000ff;  
    border: 2px solid;  
}  
p.two {  
    color: #00ff00;  
    outline: 2px solid;  
}
```

CSS Background

Background plays an important role in the visual presentation of a web page.

CSS provide several properties for styling the background of an element, including coloring the background, placing images in the background and managing their positioning, etc.

The background properties are background-color, background-image, background-repeat, background-attachment and background-position.

Background Color

The background-color property is used to set the background color of an element.

```
body {  
    background-color: #f0e68c;  
}
```

Background Image

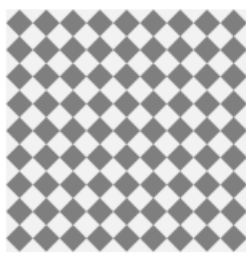
The background-image property set an image as a background of an HTML element.

```
body {  
    background-image: url("images/tile.png");  
}
```

Background Repeat

The background-repeat property allows you to control how a background image is repeated or tiled in the background of an element. You can set a background image to repeat vertically (y-axis), horizontally (x-axis), in both directions, or in neither direction.

```
body {  
    background-image: url("images/gradient.png");  
    background-repeat: repeat-x;  
}
```



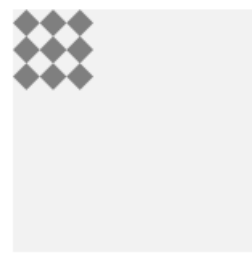
repeat



repeat-x



repeat-y



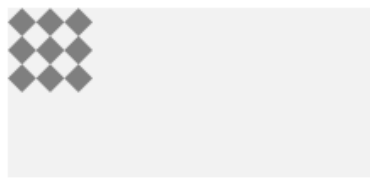
no-repeat

Background Position

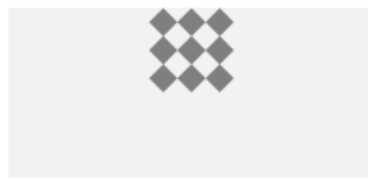
The background-position property is used to control the position of the background image.

If no background position has been specified, the background image is placed at the default top-left position of the element i.e. at (0,0)

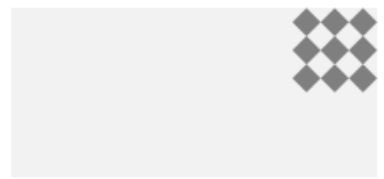
```
body {  
    background-image: url("images/robot.png");  
    background-repeat: no-repeat;  
    background-position: right top;  
}
```

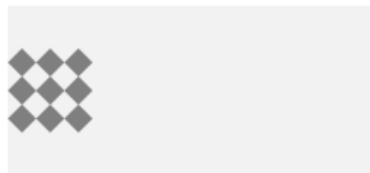
`background-position: left top;`
`background-position: 0 0;`



`background-position: top;`
`background-position: 50% 0;`



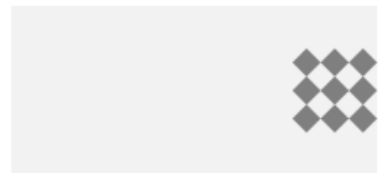
`background-position: right top;`
`background-position: 100% 0;`



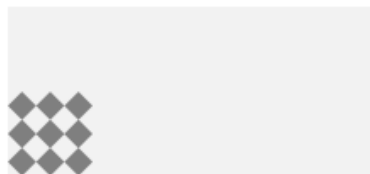
`background-position: left;`
`background-position: 0 50%;`



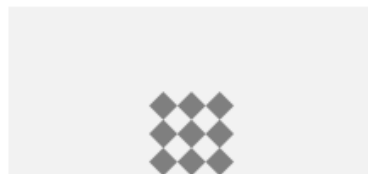
`background-position: center;`
`background-position: 50% 50%;`



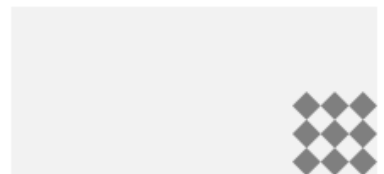
`background-position: right;`
`background-position: 100% 50%;`



`background-position: left bottom;`
`background-position: 0 100%;`



`background-position: bottom;`
`background-position: 50% 100%;`



`background-position: right bottom;`
`background-position: 100% 100%;`

Background Attachment

The background-attachment property determines whether the background image is fixed with regard to the viewport or scrolls along with the containing block.

```
body {  
  background-image: url("images/bell.png");  
  background-repeat: no-repeat;  
  background-attachment: fixed;  
}
```

The Background Shorthand Property

As you can see in the examples above, there are many properties to consider when dealing with the backgrounds. However, it is also possible to specify all these properties in one single property to shorten the code or avoid extra typing. This is called a shorthand property.

The background property is a shorthand property for setting all the individual background properties, i.e., background-color, background-image, background-repeat, background-attachment and the background-position

Syntax: `background: color image repeat attachment position;`

Example:

```
body {  
  background: #f0e68c url("images/smiley.png") no-repeat fixed 250px 25px;  
}
```

CSS Fonts

Font Properties

CSS provide several properties for styling the font of the text, including changing their face, controlling their size and boldness, managing variant, and so on.

The font properties are: font-family, font-style, font-weight, font-size, and font-variant.

Font Family

The font-family property is used to specify the font to be used to render the text.

This property can hold several comma-separated font names as a fallback system, so that if the first font is not available on the user's system, browser tries to use the second one, and so on.

generic font family which are five — serif, sans-serif, monospace, cursive and fantasy.

Example

```
body {  
  font-family: Arial, Helvetica, sans-serif;  
}
```

Note: If the name of a font family contains more than one word, it must be placed inside quotation marks, like "Times New Roman", "Courier New", "Segoe UI", etc.

Difference Between Serif and Sans-serif Fonts



Font Style

The font-style property is used to set the font face style for the text content of an element.

The font style can be normal, italic or oblique. The default value is normal.

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

Font Size

The font-size property is used to set the size of font for the text content of an element.

There are several ways to specify the font size values e.g. with keywords, percentage, pixels, ems, etc

Setting Font Size with Pixels

Setting the font size in pixel values (e.g. 14px, 16px, etc.) is a good choice when you need the pixel accuracy. Pixel is an absolute unit of measurement which specifies a fixed length.

Example

```
h1 {  
    font-size: 24px;  
}  
p {  
    font-size: 14px;  
}
```

Setting Font Size with EM

The em unit refers to the font size of the parent element. When defining the font-size property, 1em is equal to the size of the font that applies to the parent of the element.

So, if you set a font-size of 20px on the body element, then 1em = 20px and 2em = 40px.

However, if you haven't set the font size anywhere on the page, then it is the browser default, which is normally 16px. Therefore, by default 1em = 16px, and 2em = 32px.

Example

```
h1 {  
    font-size: 2em; /* 32px/16px=2em */  
}  
p {  
    font-size: 0.875em; /* 14px/16px=0.875em */  
}
```

Setting Font Size with Keywords

CSS provide several keywords that you can use to define font sizes.

An absolute font size can be specified using one of the following keywords: xx-small, x-small, small, medium, large, x-large, xx-large.

Example

```
body {  
    font-size: large;  
}  
h1 {  
    font-size: larger;  
}  
p {  
    font-size: smaller;  
}
```

Setting Font Size with Viewport Units

The font sizes can be specified using viewport units such as vw or vh.

Viewport units refer to a percentage of the browser's viewport dimensions, where 1vw = 1% of viewport width, and 1vh = 1% of viewport height. Hence, if the viewport is 1600px wide, 1vw is 16px.

Example

```
body {  
    font-size: 1vw;  
}  
h1 {  
    font-size: 3vw;  
}
```

however, there is a problem with the viewport units. On small screens fonts become so small that they are hardly readable. To prevent this you can utilize CSS calc() function

```
html {  
    font-size: calc(1em + 1vw);  
}  
h1 {  
    font-size: 3rem;  
}
```

Font Weight

The font-weight property specifies the weight or boldness of the font.

This property can take one of the following values: normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900 and inherit.

- The numeric values 100-900 specify the font weights, where each number represents a weight greater than its predecessor. 400 is same as normal & 700 is same as bold.
- The bolder and lighter values are relative to the inherited font weight, while the other values such as normal and bold are absolute font weights.

Example

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

Font Variant

The font-variant property allows the text to be displayed in a special small-caps variation.

Small-caps or small capital letters are slightly different to normal capital letters, in which lowercase letters appear as smaller versions of the corresponding uppercase letters.

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

CSS Text

Text Color

Text color is defined by the CSS color property

Example

```
h1 {
  color: #ff0000;
}
p {
  color: green;
}
```

Text Alignment

The text-align property is used to set the horizontal alignment of a text. Possible values for this property are: left, right, center, justify, and inherit.

Example

```
h1 {
  text-align: center;
}
p {
  text-align: justify;
}
```

Text Decoration

The text-decoration property is used to set or remove decorations from text. Possible values of this property are: none, underline, overline, line-through, blink and inherit.

Example

```
h1 {
  text-decoration: overline;
}
h2 {
  text-decoration: line-through;
}
h3 {
  text-decoration: underline;
}
```

Removing the Links Default Underlines

The text-decoration property is commonly used to remove the default underlines from the hyperlinks.

Example

```
a {
  text-decoration: none;
  border-bottom: 1px dotted;
}
```

Text Transformation

The text-transform property is used to set the cases for a text. It can be used to set the element's text content into uppercase or lowercase letters, or capitalize the first letter of each word. Possible values for the text-transform property are: none, capitalize, uppercase, lowercase and inherit.

Example

```
p.up {
  text-transform: uppercase;
}
p.cap {
  text-transform: capitalize;
}
p.low {
  text-transform: lowercase;
}
```

Text Indentation

The text-indent property is used to set the indentation of the first line of text of an element. Possible values for the text-indent property are: percentage (%), length (specifying indent space) or inherit.

Example

```
p {
  text-indent: 100px;
}
```

Word Spacing

The word-spacing property used to sets the spacing between the words.

```
p.one {
  word-spacing: 20px;
}
p.two {
  word-spacing: 20px;
  text-align: justify;
}
p.three {
  word-spacing: 20px;
  white-space: pre;
}
```

Letter Spacing

The letter-spacing property is used to set extra spacing between the characters of text. Possible values for this property are: length (specifying the extra space to be inserted between characters in addition to the default inter-character space), normal and inherit.

```
h1 {
  letter-spacing: -3px;
}
p {
  letter-spacing: 10px;
}
```

```
}
```

Line Height

The line-height property defines height (also called leading) of a line of text. Possible values for this property are: percentage (%), length, number, normal and inherit.

```
p {  
  line-height: 1.2;  
}
```

CSS Links

A link has four different states — link, visited, active and hover. These four states of a link or hyperlink can be styled differently through CSS properties using the pseudo-classes of anchor element, depending on what state they are in.

- `a:link` — Set styles for a normal or unvisited links that has no mouse pointer over it.
- `a:visited` — Set styles for a link the user has visited but has no mouse pointer on it.
- `a:hover` — Set styles for a link when the user place the mouse pointer over it.
- `a:active` — Set styles for a link that is in the process of being clicked.

Example

```
a:link { /* unvisited link */
  color: #FF0000;
  text-decoration: none;
}
a:visited { /* visited link */
  color: #00FF00;
}
a:hover { /* mouse over link */
  color: #FF00FF;
  text-decoration: underline;
}
a:active { /* active link */
  color: #0000FF;
}
```


CSS Lists

Types of HTML Lists

There are three different types of list in HTML:

- Unordered lists — A list of items, where every list items are marked with bullets.
- Ordered lists — A list of items, where each list items are marked with numbers.
- Definition list — A list of items, with a description of each item.

Styling Lists with CSS

CSS provides the several properties for styling the most commonly used unordered and ordered lists.

These CSS list properties typically allow you to:

- Control the shape or appearance of the marker.
- Specify an image for the marker rather than a bullet point or number.
- Set the distance between a marker and the text in the list.
- Specify whether the marker or bullet would appear inside or outside of the box containing the unordered or ordered list items.

List Style Type

By default, items in an ordered list are numbered with Arabic numerals (1, 2, 3, etc.), whereas in an unordered list, items are marked with round bullets. But, you can change this default list marker type to any other type such as circle, square, roman numerals, latin letters, and so on using the list-style-type property.

Example

```
ul {  
    list-style-type: square;  
}  
ol {  
    list-style-type: upper-roman;  
}
```

Changing the Position of List Markers

The list markers are positioned outside of the list item's boxes, by default. However, you can use the list-style-position property to specify whether the markers or bullet points appear inside or outside of the list item's principal block boxes.

This property takes the value inside or outside, with outside being the default. If the value inside is used, the lines will wrap under the marker instead of being indented.

Example

```
ul.in li {  
    list-style-position: inside;  
}  
ul.out li {  
    list-style-position: outside;  
}
```

Using Images as List Markers

You can also set an image as a list marker using the `list-style-image` property.

```
ul li {  
    list-style-image: url("arrow.png");  
}
```

Cross Browser Solution for Image Marker

When using an image as bullet using the `list-style-image` property, the bullets does not line up to the text accurately across the browsers. As a work-around, you can properly align bullet images via the `background-image` CSS property.

Example

```
ul {  
    list-style-type: none;  
}  
ul li {  
    background-image: url("arrow.png");  
    background-position: 0px 5px; /* X-pos Y-pos (from top-left) */  
    background-repeat: no-repeat;  
    padding-left: 20px;  
}
```

CSS Tables

Styling Tables with CSS

When you build an HTML table without any styles or attributes, browsers display them without any border. Styling a table with CSS can greatly improve its appearance.

Adding Borders to Tables

The CSS border property is the best way to define the borders for the tables.

Example

```
table, th, td {  
    border: 1px solid black;  
}
```

Collapsing Table Borders

If you've seen the output of the previous example, you've noticed every table cell has separate borders as well as there is some space between the borders of adjacent table cells. It happens because table cell borders are separated by default. But, you can collapse the separate table borders into one by using the border-collapse property on the <table> element:

Example

```
table {  
    border-collapse: collapse;  
}  
table, th, td {  
    border: 1px solid black;  
}
```

Adjusting Space inside Tables

Example

```
th, td {  
    padding: 15px;  
}
```

Example

```
table {  
    border-spacing: 10px;  
}
```

Handling Empty Cells

```
table {  
    border-collapse: separate;  
    empty-cells: hide;  
}
```

Controlling the Position of Table Caption

The caption-side CSS property sets the vertical position of a table caption box.

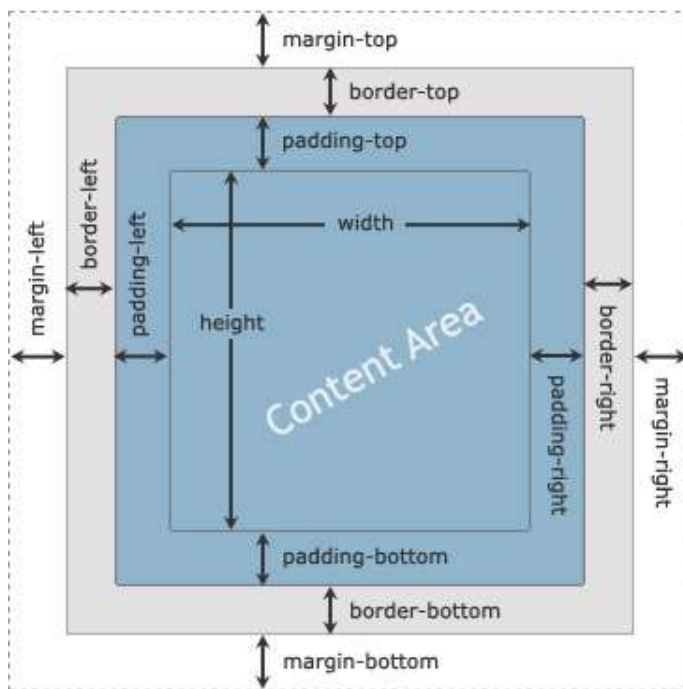
```
caption {  
    caption-side: bottom;  
}
```

CSS Box Model

What is Box Model

Every element that can be displayed is comprised of one or more rectangular boxes. CSS box model typically describes how these rectangular boxes are laid out on a web page. These boxes can have different properties and can interact with each other in different ways, but every box has a content area and optional surrounding padding, border, and margin.

The following diagram demonstrates how the padding, border, and margin CSS properties determines how much space an element can take on a web page.



Width and Height of Elements

Usually when you set the width and height of an element using the CSS width and height properties, in reality you are only setting the width and height of the content area of an element. The actual width and height of the element's box depend on several factors.

The actual space that an element's box might take is calculated like this:

Box Size	CSS Properties
Total Width	width + padding-left + padding-right + border-left + border-right + margin-left + margin-right
Total Height	height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom

CSS Margin

CSS Margin Properties

The CSS margin properties allow you to set the margins around the sides of an element's box. The margins does not have a background-color, it is completely transparent.

Setting Margins for Individual Sides

You can easily specify the different margins for the different sides of an element such as top, right, bottom or left side using the CSS individual margin property.

Example

```
h1 {  
    margin-bottom: 20px;  
}  
p {  
    margin-left: 10px;  
    margin-right: 30px;  
}
```

The margin Shorthand Property

The margin property is a shorthand property to avoid setting margin of each side separately: margin-top, margin-right, margin-bottom and margin-left.

Example

```
h1 {  
    margin: 0 10px;  
}  
p {  
    margin: 25px 50px 75px 100px;  
}
```

CSS Padding

CSS Padding Properties

The CSS padding properties allow you to set the padding area for an element that separates its border from its content. The padding is affected by the background-color of the box.

Define Paddings for Individual Sides

You can easily specify the different paddings for the different sides of an element such as top, right, bottom or left side using the CSS individual padding property.

Example

```
h1 {  
  padding-bottom: 10px;  
}  
p {  
  padding-top: 20px;  
  padding-left: 50px;  
}
```

The padding Shorthand Property

The padding property is a shorthand property to avoid setting padding for each side of an element separately i.e. padding-top, padding-right, padding-bottom, padding-left.

Example

```
h1 {  
  padding: 10px 20px;  
}  
p {  
  padding: 10px 15px 20px 25px;  
}
```

CSS Border

CSS Border Properties

The CSS border properties allow you to define the border area of a box. The border can either be a predefined style like, solid line, double line, dotted line, etc. or it can be an image. The following section will describe you how to set the various properties defining the style (border-style), color (border-color), and thickness (border-width) of the border.

The border-width Property

The border-width property specifies the width of the border area. It is a shorthand property for setting the thickness of all the four sides of an element's border at the same time.

Example

```
p {  
    border-width: medium 10px thick 15px;  
}
```

The border-style Property

The border-style property sets the style of a box's border such as: solid, dotted, etc. It is a shorthand property for setting the line style for all four sides of the elements border.

The border-style property may take one of the following values: none, hidden, dashed, dotted, double, groove, inset, outset, ridge and solid.

Example

```
p {  
    border-style: dotted;  
}
```

The border-color Property

The border-color property specify the color of a box's border. This is also a shorthand property for setting the color of all the four sides of an element's border.

Example

```
p {  
    border-style: solid;  
    border-color: #ff0000;  
}
```

The border Shorthand Property

The border CSS property is a shorthand property for setting one or more of the individual border properties border-style, border-width and border-color in a single rule.

Example

```
p {  
    border: 5px solid #ff4500;  
}
```

CSS Outline

The outlines are generally used to highlight elements. An outline at a glance looks very similar to the border, but it differs from border in the following ways:

- Outlines do not take up space, because they always placed on top of the box of the element which may cause them to overlap other elements on the page.
- Unlike borders, outlines won't allow us to set each edge to a different width, or set different colors and styles for each edge. An outline is the same on all sides.
- Outlines don't have any impact on surrounding elements apart from overlapping.
- Unlike borders, outlines don't change the size or position of the element.
- Outlines may be non-rectangular.

The outline-width Property

```
p {  
  outline-width: thick;  
}
```

The outline-style Property

This property may take one of the following values: none, hidden, dashed, dotted, double, groove, inset, outset, ridge and solid. Just like border-style property.

```
p {  
  outline-style: double;  
}
```

The outline-color Property










```
p {  
  outline-style: solid;  
  outline-color: #0000ff;  
}
```


CSS Cursors

Changing the Look of Cursor

The browsers typically display the mouse pointer over any blank part of a web page, the gloved hand over any linked or clickable item and the edit cursor over any text or text field. With CSS you can redefine those properties to display a variety of different cursors.

```
h1 {  
    cursor: move;  
}  
p {  
    cursor: text;  
}
```

Look	Values	Example
	default	a:hover{cursor:default;}
	pointer	a:hover{cursor:pointer;}
	text	a:hover{cursor:text;}
	wait	a:hover{cursor:wait;}
	help	a:hover{cursor:help;}
	progress	a:hover{cursor:progress;}
	crosshair	a:hover{cursor:crosshair;}
	move	a:hover{cursor:move;}
	url()	a:hover{cursor:url("custom.cur"), default;}

Creating a Customized Cursor

```
a {  
    cursor: url("custom.gif"), url("custom.cur"), default;  
}
```

CSS Overflow

Handling Overflowing Content

There may be a situation when the content of an element might be larger than the dimensions of the box itself. For example given width and height properties did not allow enough room to accommodate the content of the element.

CSS overflow property allowing you to specify whether to clip content, render scroll bars or display overflow content of a block-level element.

```
div {  
  width: 250px;  
  height: 150px;  
  overflow: scroll;  
}
```

Value	Description
visible	The default value. Content is not clipped; it will be rendered outside the element's box, and may thus overlap other content.
hidden	Content that overflows the element's box is clipped and the rest of the content will be invisible.
scroll	The overflowing content is clipped, just like hidden, but provides a scrolling mechanism to access the overflowed content.
auto	If content overflows the element's box it will automatically provides the scrollbars to see the rest of the content, otherwise scrollbar will not appear.

CSS Dimension

The width and height Properties

```
div {  
  width: 300px;  
  height: 200px;  
}
```

The max-height Property

```
div {  
  height: 200px;  
  max-height: 100px;  
}
```

The min-height Property

```
div {  
  height: 200px;  
  min-height: 300px;  
}
```

The max-width Property

```
div {  
  width: 300px;  
  max-width: 200px;  
}
```

The min-width Property

```
div {  
  width: 300px;  
  min-width: 400px;  
}
```

CSS Display

Display Block

The block value of the display property forces an element to behave like block-level element, like a `<div>` or `<p>` element. The style rules in the following example displays the `` and `<a>` elements as block-level elements:

```
span {  
    display: block;  
}  
a {  
    display: block;  
}
```

Display Inline

The inline value of the display property causes an element to behave as though it were an inline-level element, like a `` or an `<a>` element. The style rules in the following example displays the `<p>` and `` elements as inline-level elements:

```
p {  
    display: inline;  
}  
ul li {  
    display: inline;  
}
```

Display Inline-Block

The inline-block value of the display property causes an element to generate a block box that will be flowed with surrounding content i.e. in the same line as adjacent content. The following style rules displays the `<div>` and `` elements as inline-block:

```
div {  
    display: inline-block;  
}  
span {  
    display: inline-block;  
}
```

Display None

The value none simply causes an element to generate no boxes at all. Child elements do not generate any boxes either, even if their display property is set to something other than none. The document is rendered as though the element did not exist in the document tree.

```
h1 {  
    display: none;  
}  
p {  
    display: none;  
}
```

CSS Visibility

Controlling the Visibility of Elements

You can use the visibility property to control whether an element is visible or not. This property can take one of the following values listed in the table below:

Value	Description
visible	Default value. The box and its contents are visible.
hidden	The box and its content are invisible, but still affect the layout of the page.
collapse	This value causes the entire row or column to be removed from the display. This value is used for row, row group, column, and column group elements.
inherit	Specifies that the value of the visibility property should be inherited from the parent element i.e. takes the same visibility value as specified for its parent.

CSS Visibility vs Display

The display and visibility CSS properties appear to be the same thing, but they are in fact quite different and often confuse those new to web development.

- visibility: hidden; hides the element, but it still takes up space in the layout. Child element of a hidden box will be visible if their visibility is set to visible.
- display: none; turns off the display and removes the element completely from the document. It does not take up any space, even though the HTML for it is still in the source code. All child elements also have their display turned off, even if their display property is set to something other than none.

CSS Position

CSS Positioning Methods

Positioning elements appropriately on the web pages is a necessity for a good layout design. There are several methods in CSS that you can use for positioning elements.

Static Positioning

A static positioned element is always positioned according to the normal flow of the page. HTML elements are positioned static by default. Static positioned elements are not affected by the top, bottom, left, right, and z-index properties.

```
.box {  
  padding: 20px;  
  background: #7dc765;  
  position: static;  
}
```

Relative Positioning

A relative positioned element is positioned relative to its normal position.

In the relative positioning scheme the element's box position is calculated according to the normal flow. Then the box is shifted from this normal position according to the properties — top or bottom and/or left or right.

```
.box {  
  position: relative;  
  left: 100px;  
}
```

Absolute Positioning

An absolutely positioned element is positioned relative to the first parent element that has a position other than static. If no such element is found, it will be positioned on a page relative to the 'top-left' corner of the browser window. The box's offsets further can be specified using one or more of the properties top, right, bottom, and left.

```
.box {  
  position: absolute;  
  top: 200px;  
  left: 100px;  
}
```

Fixed Positioning

Fixed positioning is a subcategory of absolute positioning.

The only difference is, a fixed positioned element is fixed with respect to the browser's viewport and does not move when scrolled.

```
.box {  
  position: fixed;  
  top: 200px;  
  left: 100px;  
}
```

CSS Layer (Z –index)

CSS Float

CSS Pseudo classes(:first-child, :last-child, nth-child(2n))

CSS Pseudo element (::after , ::before)

CSS Opacity (opacity:0.5) / filter: alpha(opacity=50)

CSS3 Gradient

CSS3 box-shadow

CSS3 box-shadow Property

The box-shadow property can be used to add shadow to the element's boxes. You can even apply more than one shadow effects using a comma-separated list of shadows. The basic syntax of creating a box shadow can be given with:

box-shadow: offset-x offset-y blur-radius color;

```
.box{  
  width: 200px;  
  height: 150px;  
  background: #ccc;  
  box-shadow: 5px 5px 10px #999;  
}
```

CSS3 text-shadow Property

You can use the text-shadow property to apply the shadow effects on text. You can also apply multiple shadows to text using the same notation as box-shadow.

```
h1 {  
  text-shadow: 5px 5px 10px #666;  
}  
h2 {  
  text-shadow: 5px 5px 10px red, 10px 10px 20px yellow;  
}
```

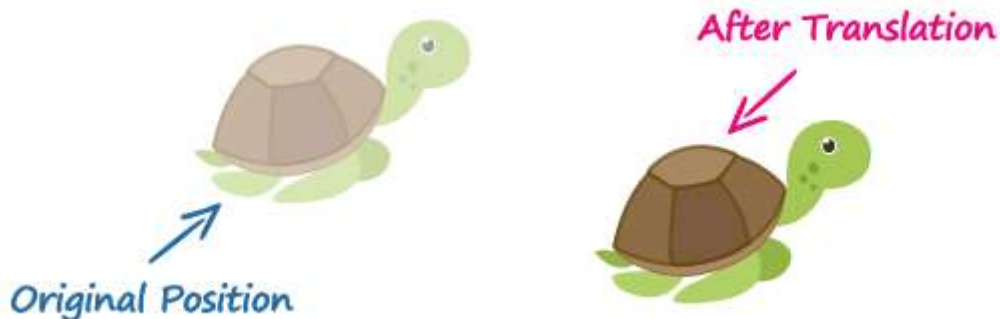
CSS3 2D Transforms

The translate() Function

Moves the element from its current position to a new position along the X and Y axes. This can be written as `translate(tx, ty)`. If `ty` isn't specified, its value is assumed to be zero.

```
img {  
  -webkit-transform: translate(200px, 50px); /* Chrome, Safari, Opera */  
  -moz-transform: translate(200px, 50px); /* Firefox */  
  -ms-transform: translate(200px, 50px); /* IE 9 */  
  transform: translate(200px, 50px); /* Standard syntax */  
}
```

The function `translate(200px, 50px)` moves the image 200 pixels horizontally along the positive x-axis, and 50 pixels vertically along the positive y-axis.



The rotate() Function

The `rotate()` function rotates the element around its origin (as specified by the `transform-origin` property) by the specified angle.

```
p {  
  -webkit-transform: rotate(30deg); /* Chrome, Safari, Opera */  
  -moz-transform: rotate(30deg); /* Firefox */  
  -ms-transform: rotate(30deg); /* IE 9 */  
  transform: rotate(30deg); /* Standard syntax */  
}
```

The scale() Function

The `scale()` function increases or decreases the size of the element. It can be written as `scale(sx, sy)`. If `sy` isn't specified, it is assumed to be equal to `sx`.

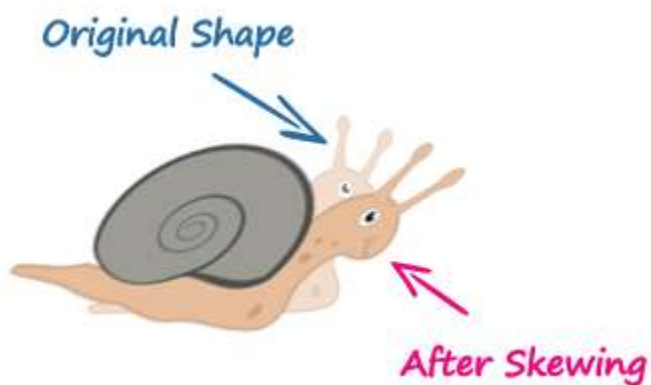
```
img {  
  -webkit-transform: scale(1.5); /* Chrome, Safari, Opera */  
  -moz-transform: scale(1.5); /* Firefox */  
  -ms-transform: scale(1.5); /* IE 9 */  
  transform: scale(1.5); /* Modern Browsers */  
}
```


The skew() Function

The skew() function skews the element along the X and Y axes by the specified angles. It can be written as skew(ax, ay). If ay isn't specified, its value is assumed to be zero.

```
img {  
  -webkit-transform: skew(-40deg, 15deg); /* Chrome, Safari, Opera */  
  -moz-transform: skew(-40deg, 15deg); /* Firefox */  
  -ms-transform: skew(-40deg, 15deg); /* IE 9 */  
  transform: skew(-40deg, 15deg); /* Modern Browsers */  
}
```

The function skew(-40deg, 15deg) skews the element -40 degree horizontally along the x-axis, and 15 degree vertically along the y-axis.



CSS3 Transitions

Understanding CSS3 Transitions

Normally when the value of a CSS property changes, the rendered result is instantly updated. A common example is changing the background color of a button on mouse hover. In a normal scenario the background color of the button is changes immediately from the old property value to the new property value when you place the cursor over the button.

```
button {  
    background: #fd7c2a;  
    /* For Safari 3.0+ */  
    -webkit-transition-property: background;  
    -webkit-transition-duration: 2s;  
    /* Standard syntax */  
    transition-property: background;  
    transition-duration: 2s;  
}  
button:hover {  
    background: #3cc16e;  
}
```

Performing Multiple Transitions

Each of the transition properties can take more than one value, separated by commas, which provides an easy way to define multiple transitions at once with different settings.

```
button {  
    background: #fd7c2a;  
    border: 3px solid #dc5801;  
    /* For Safari 3.0+ */  
    -webkit-transition-property: background, border;  
    -webkit-transition-duration: 1s, 2s;  
    /* Standard syntax */  
    transition-property: background, border;  
    transition-duration: 1s, 2s;  
}  
button:hover {  
    background: #3cc16e;  
    border-color: #288049;  
}
```

Transition Shorthand Property

There are many properties to consider when applying the transitions. However, it is also possible to specify all the transition properties in one single property to shorten the code.

The transition property is a shorthand property for setting all the individual transition properties (i.e., transition-property, transition-duration, transition-timing-function, and transition-delay) at once in the listed order.

```
button {  
  background: #fd7c2a;  
  -webkit-transition: background 2s ease-in 0s; /* For Safari 3.0+ */  
  transition: background 2s ease-in 0s; /* Standard syntax */  
}  
button:hover {  
  background: #3cc16e;  
}
```

Property	Description
transition	A shorthand property for setting all the four individual transition properties in a single declaration.
transition-delay	Specifies when the transition will start.
transition-duration	Specifies the number of seconds or milliseconds a transition animation should take to complete.
transition-property	Specifies the names of the CSS properties to which a transition effect should be applied.
transition-timing-function	Specifies how the intermediate values of the CSS properties being affected by a transition will be calculated.

CSS3 Animations

The CSS3 animations take it a step further with keyframe-based animations that allow you to specify the changes in CSS properties over time as a set of keyframes, like flash animations. Creating CSS animations is a two step process, as shown in the example below:

- The first step of building a CSS animation is to defining individual keyframes and naming an animation with a keyframes declaration.
- The second step is referencing the keyframes by name using the animation-name property as well as adding animation-duration and other optional animation properties to control the animation's behavior.

```
.box {  
  width: 50px;  
  height: 50px;  
  background: red;  
  position: relative;  
  /* Chrome, Safari, Opera */  
  -webkit-animation-name: moveit;  
  -webkit-animation-duration: 2s;  
  /* Standard syntax */  
  animation-name: moveit;  
  animation-duration: 2s;  
}
```

```
/* Chrome, Safari, Opera */  
@-webkit-keyframes moveit {  
  from {left: 0;}  
  to {left: 50%;}  
}
```

```
/* Standard syntax */  
@keyframes moveit {  
  from {left: 0;}  
  to {left: 50%;}  
}
```

Property	Description
<code>animation</code>	A shorthand property for setting all the animation properties in single declaration.
<code>animation-name</code>	Specifies the name of <code>@keyframes</code> defined animations that should be applied to the selected element.
<code>animation-duration</code>	Specifies how many seconds or milliseconds that an animation takes to complete one cycle of the animation.
<code>animation-timing-function</code>	Specifies how the animation will progress over the duration of each cycle i.e. the easing functions.
<code>animation-delay</code>	Specifies when the animation will start.
<code>animation-iteration-count</code>	Specifies the number of times an animation cycle should be played before stopping.
<code>animation-direction</code>	Specifies whether or not the animation should play in reverse on alternate cycles.
<code>animation-fill-mode</code>	Specifies how a CSS animation should apply styles to its target before and after it is executing.
<code>animation-play-state</code>	Specifies whether the animation is running or paused.
<code>@keyframes</code>	Specifies the values for the animating properties at various points during the animation.

CSS Multi- Column Layout

The CSS3 has introduced the multi-column layout module for creating multiple column layouts in an easy and efficient way. Now you can create layouts like you see in magazines and newspapers without using the floating boxes

```
p {  
  -webkit-column-count: 3; /* Chrome, Safari, Opera */  
  -moz-column-count: 3; /* Firefox */  
  column-count: 3; /* Standard syntax */  
}
```

Property	Description
column-count	Specifies the number of columns inside a multi-column element.
column-fill	Specifies how content is spread across columns.
column-gap	Specifies the gap between the columns.
column-rule	Specifies a straight line or rule, to be drawn between each column.
column-rule-color	Specifies the color of the rule between columns.
column-rule-style	Specifies the style of the rule between columns.
column-rule-width	Specifies the width of the rule between columns.
column-span	Specifies how many columns an element spans across.
column-width	Specifies the optimal width of the columns.
columns	A shorthand property for setting both the <code>column-width</code> and the <code>column-count</code> properties at the same time.

CSS3 Media Queries

Media Queries and Responsive Web Design

Media queries allow you to customize the presentation of your web pages for a specific range of devices like mobile phones, tablets, desktops, etc. without any change in markups. A media query consists of a media type and zero or more expressions that match the type and conditions of a particular media features such as device width or screen resolution.

Since media query is a logical expression it can be resolve to either true or false. The result of the query will be true if the media type specified in the media query matches the type of device the document is being displayed on, as well as all expressions in the media query are satisfied. When a media query is true, the related style sheet or style rules are applied to the target device.

```
/* Smartphones (portrait and landscape) ----- */
```

```
@media screen and (min-width: 320px) and (max-width: 480px){
```

```
    /* styles */
```

```
}
```

```
/* Smartphones (portrait) ----- */
```

```
@media screen and (max-width: 320px){
```

```
    /* styles */
```

```
}
```

```
/* Smartphones (landscape) ----- */
```

```
@media screen and (min-width: 321px){
```

```
    /* styles */
```

```
}
```

```
/* Tablets, iPads (portrait and landscape) ----- */
```

```
@media screen and (min-width: 768px) and (max-width: 1024px){
```

```
    /* styles */
```

```
}
```

```
/* Tablets, iPads (portrait) ----- */
```

```
@media screen and (min-width: 768px){
```

```
    /* styles */
```

```
}

/* Tablets, iPads (landscape) ----- */

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

    /* styles */

}

/* Desktops and laptops ----- */

@media screen and (min-width: 1224px){

    /* styles */

}

/* Large screens ----- */

@media screen and (min-width: 1824px){

    /* styles */

}
```