

Introduction

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS is an acronym stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

What is CSS?

- **CSS** stands for **Cascading Style Sheets**
- CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**
- CSS **saves a lot of work**. It can control the layout of multiple web pages all at once
- External stylesheets are stored in **CSS files**

Why Use CSS?

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

Advantages of CSS

- **CSS saves time** – You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** – If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
- **Easy maintenance** – To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** – CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** – Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** – Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.
- **Offline Browsing** – CSS can store web applications locally with the help of an offline cache. Using of this, we can view offline websites. The cache also ensures faster loading and better overall performance of the website.
- **Platform Independence** – The Script offer consistent platform independence and can support latest browsers as well.

Who Creates and Maintains CSS?

CSS was invented by **Håkon Wium Lie** on October 10, 1994 and maintained through a group of people within the W3C called the CSS Working Group. The CSS Working Group creates documents called **specifications**. When a specification has been discussed and officially ratified by W3C members, it becomes a recommendation.

These ratified specifications are called recommendations because the W3C has no control over the actual implementation of the language. Independent companies and organizations create that software.

NOTE – The World Wide Web Consortium, or W3C is a group that makes recommendations about how the Internet works and how it should evolve.

CSS Versions

Cascading Style Sheets, level 1 (CSS1) was came out of W3C as a recommendation in December 1996. This version describes the CSS language as well as a simple visual formatting model for all the HTML tags.

CSS2 was became a W3C recommendation in May 1998 and builds on CSS1. This version adds support for media-specific style sheets e.g. printers and aural devices, downloadable fonts, element positioning and tables.

CSS3 was became a W3C recommendation in June 1999 and builds on older versions CSS. it has divided into documentations is called as Modules and here each module having new extension features defined in CSS2.

CSS3 Modules

CSS3 Modules are having old CSS specifications as well as extension features.

- Selectors
- Box Model
- Backgrounds and Borders
- Image Values and Replaced Content
- Text Effects
- 2D/3D Transformations
- Animations
- Multiple Column Layout
- User Interface

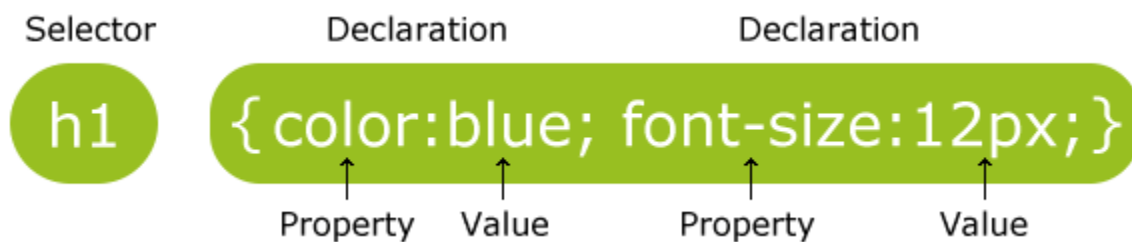
CSS Style Rules

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts –

- **Selector** – A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.

- **Property** - A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be *color*, *border* etc.
- **Value** - Values are assigned to properties. For example, *color* property can have value either *red* or *#F1F1F1* etc.

CSS Syntax



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

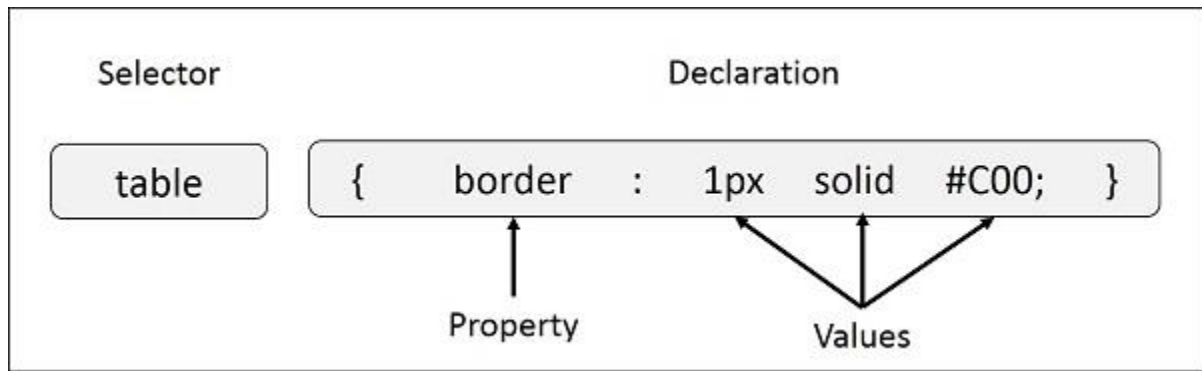
The declaration block contains one or more declarations separated by semicolons.

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

A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

In the following example all `<p>` elements will be center-aligned, with a red text color:

```
selector { property: value }
```



Example: You can define a table border as follows –

```
table { border : 1px solid #C00; }
```

Here table is a selector and border is a property and given value *1px solid #C00* is the value of that property.

You can define selectors in various simple ways based on your comfort. Let me put these selectors one by one.

The Type Selectors

This is the same selector we have seen above. Again, one more example to give a color to all level 1 headings:

```
h1 {  
  color: #36CFFF;  
}
```

The Universal Selectors

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type –

```
* {  
  color: #000000;  
}
```

This rule renders the content of every element in our document in black.

The Descendant Selectors

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to `` element only when it lies inside `` tag.

```
ul em {  
    color: #000000;  
}
```

The Class Selectors

The class selector selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class.

In the example below, all HTML elements with `class="center"` will be red and center-aligned:

```
.black {  
    color: #000000;  
}
```

This rule renders the content in black for every element with class attribute set to *black* in our document. You can make it a bit more particular. For example:

```
h1.black {  
    color: #000000;  
}
```

This rule renders the content in black for only `<h1>` elements with class attribute set to *black*.

You can apply more than one class selectors to given element. Consider the following example:

```
<p class="center bold">
```

This para will be styled by the classes *center* and *bold*.

</p>

The ID Selectors

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

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

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

```
#black {  
    color: #000000;  
}
```

This rule renders the content in black for every element with *id* attribute set to *black* in our document. You can make it a bit more particular. For example –

```
h1#black {  
    color: #000000;  
}
```

This rule renders the content in black for only <h1> elements with *id* attribute set to *black*.

The true power of *id* selectors is when they are used as the foundation for descendant selectors, For example:

```
#black h2 {  
    color: #000000;  
}
```

In this example all level 2 headings will be displayed in black color when those headings will lie with in tags having *id* attribute set to *black*.

The Child Selectors

You have seen the descendant selectors. There is one more type of selector, which is very similar to descendants but have different functionality. Consider the following example –

```
body > p {  
    color: #000000;  
}
```

This rule will render all the paragraphs in black if they are direct child of <body> element. Other paragraphs put inside other elements like <div> or <td> would not have any effect of this rule.

The Attribute Selectors

You can also apply styles to HTML elements with particular attributes. The style rule below will match all the input elements having a type attribute with a value of *text* –

```
input[type = "text"]{  
    color: #000000;  
}
```

The advantage to this method is that the <input type = "submit" /> element is unaffected, and the color applied only to the desired text fields.

There are following rules applied to attribute selector.

- **p[lang]** - Selects all paragraph elements with a *lang* attribute.
- **p[lang="fr"]** - Selects all paragraph elements whose *lang* attribute has a value of exactly "fr".
- **p[lang~="fr"]** - Selects all paragraph elements whose *lang* attribute contains the word "fr".

- **p[lang]="en"]** - Selects all paragraph elements whose *lang* attribute contains values that are exactly "en", or begin with "en-".

Multiple Style Rules

You may need to define multiple style rules for a single element. You can define these rules to combine multiple properties and corresponding values into a single block as defined in the following example –

```
h1 {  
  color: #36C;  
  font-weight: normal;  
  letter-spacing: .4em;  
  margin-bottom: 1em;  
  text-transform: lowercase;  
}
```

Here all the property and value pairs are separated by a **semi colon (;)**. You can keep them in a single line or multiple lines. For better readability we keep them into separate lines.

For a while, don't bother about the properties mentioned in the above block. These properties will be explained in the coming chapters and you can find complete detail about properties in CSS References.

Grouping Selectors

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;  
  letter-spacing: .4em;  
  margin-bottom: 1em;
```

```
text-transform: lowercase;

}
```

This define style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.

You can combine the various *id* selectors together as shown below –

```
#content, #footer, #supplement {

    position: absolute;

    left: 510px;

    width: 200px;

}
```

CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment starts with `/*` and ends with `*/`. Comments can also span multiple lines:

Example

```
p {
    color: red;
    /* This is a single-line comment */
    text-align: center;
}

/* This is
a multi-line
comment */
```

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

External Style Sheet

With an external style sheet, you can change the look of an entire website by changing just one file!

Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section:

Example

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

Here is how the "mystyle.css" looks:

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

h1 {
    color: navy;
    margin-left: 20px;
}
```

Note: Do not add a space between the property value and the unit (such as `margin-left: 20 px;`). The correct way is: `margin-left: 20px;`

Internal Style Sheet

An internal style sheet may be used if one single page has a unique style.

Internal styles are defined within the `<style>` element, inside the `<head>` section of an HTML page:

Example

```
<head>
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
```

Inline Styles

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the `style` attribute to the relevant element. The `style` attribute can contain any CSS property.

The example below shows how to change the color and the left margin of a `<h1>` element:

Example

```
<h1 style="color:blue;margin-left:30px;">This is a heading</h1>
```

Tip: An inline style loses many of the advantages of a style sheet (by mixing content with presentation). Use this method sparingly.

Multiple Style Sheets

If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

Example

Assume that an external style sheet has the following style for the <h1> element:

```
h1 {  
  color: navy;  
}
```

then, assume that an internal style sheet also has the following style for the <h1> element:

```
h1 {  
  color: orange;  
}
```

If the internal style is defined after the link to the external style sheet, the <h1> elements will be "orange":

Example

```
<head>  
<link rel="stylesheet" type="text/css" href="mystyle.css">  
<style>  
h1 {  
  color: orange;  
}  
</style>  
</head>
```

However, if the internal style is defined before the link to the external style sheet, the <h1> elements will be "navy":

Example

```
<head>  
<style>  
h1 {
```

```
    color: orange;
}
</style>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Cascading Order

What style will be used when there is more than one style specified for an HTML element?

Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style (inside a specific HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or a browser default value.

There are four ways to associate styles with your HTML document. Most commonly used methods are inline CSS and External CSS.

CSS Colors Properties

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

Color Names

In HTML, a color can be specified by using a color name:



LightGray

HTML supports [140 standard color names](#).

Background Color

You can set the background color for HTML elements:

Hello World

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>  
<p style="background-color:Tomato;">Lorem ipsum...</p>
```

Text Color

You can set the color of text:

Hello World

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example

```
<h1 style="color:Tomato;">Hello World</h1>  
<p style="color:DodgerBlue;">Lorem ipsum...</p>  
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

Border Color

You can set the color of borders:

Hello World

Hello World

Hello World

Example

```
<h1 style="border:2px solid Tomato;">Hello World</h1>  
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>  
<h1 style="border:2px solid Violet;">Hello World</h1>
```

Color Values

In HTML, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values:

Same as color name "Tomato":

rgb(255, 99, 71)

#ff6347

hsl(9, 100%, 64%)

Same as color name "Tomato", but 50% transparent:

Example

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>  
<h1 style="background-color:#ff6347;">...</h1>  
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>  
  
<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>  
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>
```

RGB Value

In HTML, a color can be specified as an RGB value, using this formula:

rgb(*red*, *green*, *blue*)

Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.

To display the color black, all color parameters must be set to 0, like this: rgb(0, 0, 0).

To display the color white, all color parameters must be set to 255, like this: rgb(255, 255, 255).

Experiment by mixing the RGB values below:



rgb(255, 99, 71)

RED

255

GREEN

99

BLUE

71

Example



rgb(255, 0, 0)



rgb(0, 0, 255)

rgb(60, 179, 113)


rgb(238, 130, 238)

rgb(255, 165, 0)

rgb(106, 90, 205)

Shades of gray are often defined using equal values for all the 3 light sources:

Example



rgb(0, 0, 0)

rgb(60, 60, 60)

rgb(120, 120, 120)

rgb(180, 180, 180)

rgb(240, 240, 240)

rgb(255, 255, 255)

HEX Value

In HTML, a color can be specified using a hexadecimal value in the form:

#rrggbb

Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

For example, #ff0000 is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).

Example

#ff0000

#0000ff



#3cb371

#ee82ee

#ffa500

#6a5acd

Shades of gray are often defined using equal values for all the 3 light sources:

Example



#000000

#3c3c3c

#787878

#b4b4b4

#f0f0f0

#ffffff

HSL Value

In HTML, a color can be specified using hue, saturation, and lightness (HSL) in the form:

hsl(*hue*, *saturation*, *lightness*)

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white

Example

hsl(0, 100%, 50%)

hsl(240, 100%, 50%)



`hsl(147, 50%, 47%)`

`hsl(300, 76%, 72%)`

`hsl(39, 100%, 50%)`

`hsl(248, 53%, 58%)`

Saturation

Saturation can be describe as the intensity of a color.

100% is pure color, no shades of gray

50% is 50% gray, but you can still see the color.

0% is completely gray, you can no longer see the color.

Example



`hsl(0, 100%, 50%)`

`hsl(0, 80%, 50%)`



hsl(0, 60%, 50%)

hsl(0, 40%, 50%)

hsl(0, 20%, 50%)

hsl(0, 0%, 50%)

Lightness

The lightness of a color can be described as how much light you want to give the color, where 0% means no light (black), 50% means 50% light (neither dark nor light) 100% means full lightness (white).

Example



hsl(0, 100%, 0%)

hsl(0, 100%, 25%)

hsl(0, 100%, 50%)

hsl(0, 100%, 75%)

hsl(0, 100%, 90%)

hsl(0, 100%, 100%)

Shades of gray are often defined by setting the hue and saturation to 0, and adjust the lightness from 0% to 100% to get darker/lighter shades:

Example

hsl(0, 0%, 0%)

hsl(0, 0%, 24%)

hsl(0, 0%, 47%)

hsl(0, 0%, 71%)

hsl(0, 0%, 94%)

hsl(0, 0%, 100%)

RGBA Value

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with:

rgba(*red, green, blue, alpha*)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Example

rgba(255, 99, 71, 0)

rgba(255, 99, 71, 0.2)



rgba(255, 99, 71, 1)

HSLA Value

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with:

hsla(*hue*, *saturation*, *lightness*, *alpha*)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Example

hsla(9, 100%, 64%, 0)

hsla(9, 100%, 64%, 0.2)



hsla(9, 100%, 64%, 1)

CSS Text Properties

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)"

Look at [CSS Color Values](#) for a complete list of possible color values.

The default text color for a page is defined in the body selector.

Example

```
body {  
  color: blue;  
}
```

```
h1 {  
  color: green;  
}
```

Note: For W3C compliant CSS: If you define the **color** property, you must also define the **background-color**.

Text Alignment

The **text-align** property is used to set the horizontal alignment of a text.

A text can be left or right aligned, centered, or justified.

The following example shows center aligned, and left and right aligned text (left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left):

Example

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

```
h2 {  
  text-align: left;  
}
```

```
h3 {  
  text-align: right;  
}
```

When the **text-align** property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers):

Example

```
div {  
  text-align: justify;  
}
```

Text Decoration

The **text-decoration** property is used to set or remove decorations from text.

The value **text-decoration: none;** is often used to remove underlines from links:

Example

```
a {  
  text-decoration: none;  
}
```

The other **text-decoration** values are used to decorate text:

Example

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

Note: It is not recommended to underline text that is not a link, as this often confuses the reader.

Text Transformation

The **text-transform** property is used to specify uppercase and lowercase letters in a text.

It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

Example

```
p.uppercase {  
  text-transform: uppercase;  
}  
  
p.lowercase {  
  text-transform: lowercase;  
}
```

```
p.capitalize {  
  text-transform: capitalize;  
}
```

Text Indentation

The **text-indent** property is used to specify the indentation of the first line of a text:

Example

```
p {  
  text-indent: 50px;  
}
```

Letter Spacing

The **letter-spacing** property is used to specify the space between the characters in a text.

The following example demonstrates how to increase or decrease the space between characters:

Example

```
h1 {  
  letter-spacing: 3px;  
}  
  
h2 {  
  letter-spacing: -3px;  
}
```

Line Height

The **line-height** property is used to specify the space between lines:

Example

```
p.small {  
  line-height: 0.8;  
}
```

```
p.big {  
  line-height: 1.8;  
}
```

Text Direction

The **direction** property is used to change the text direction of an element:

Example

```
p {  
  direction: rtl;  
}
```

Word Spacing

The **word-spacing** property is used to specify the space between the words in a text.

The following example demonstrates how to increase or decrease the space between words:

Example

```
h1 {  
  word-spacing: 10px;  
}
```

```
h2 {  
  word-spacing: -5px;  
}
```

Text Shadow

The **text-shadow** property adds shadow to text.

The following example specifies the position of the horizontal shadow (3px), the position of the vertical shadow (2px) and the color of the shadow (red):

Example

```
h1 {  
  text-shadow: 3px 2px red;  
}
```

All CSS Text Properties

Property	Description
color	Sets the color of text
direction	Specifies the text direction/writing direction
letter-spacing	Increases or decreases the space between characters in a text
line-height	Sets the line height
text-align	Specifies the horizontal alignment of text
text-decoration	Specifies the decoration added to text
text-indent	Specifies the indentation of the first line in a text-block

<u>text-shadow</u>	Specifies the shadow effect added to text
<u>text-transform</u>	Controls the capitalization of text
<u>text-overflow</u>	Specifies how overflowed content that is not displayed should be signaled to the user
<u>unicode-bidi</u>	Used together with the <u>direction</u> property to set or return whether the text should be overridden to support multiple languages in the same document
<u>vertical-align</u>	Sets the vertical alignment of an element
<u>white-space</u>	Specifies how white-space inside an element is handled
<u>word-spacing</u>	Increases or decreases the space between words in a text

CSS Fonts Properties

The CSS font properties define the font family, boldness, size, and the style of a text.

Difference Between Serif and Sans-serif Fonts



CSS Font Families

In CSS, there are two types of font family names:

- **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** - a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

Note: On computer screens, sans-serif fonts are considered easier to read than serif fonts.

Font Family

The font family of a text is set with the `font-family` property.

The `font-family` property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Note: If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".

More than one font family is specified in a comma-separated list:

Example

```
p {  
  font-family: "Times New Roman", Times, serif;  
}
```

For commonly used font combinations, look at our [Web Safe Font Combinations](#).

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)

Example

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

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

Font Size

The `font-size` property sets the size of the text.

Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

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.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Note: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

Set Font Size With Pixels

Setting the text size with pixels gives you full control over the text size:

Example

```
h1 {  
  font-size: 40px;  
}  
  
h2 {  
  font-size: 30px;  
}  
  
p {  
  font-size: 14px;  
}
```

Tip: If you use pixels, you can still use the zoom tool to resize the entire page.

Set Font Size With Em

To allow users to resize the text (in the browser menu), many developers use em instead of pixels.

The em size unit is recommended by the W3C.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: $pixels/16=em$

Example

```
h1 {  
  font-size: 2.5em; /* 40px/16=2.5em */  
}  
  
h2 {
```

```
font-size: 1.875em; /* 30px/16=1.875em */
}

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

In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.

Unfortunately, there is still a problem with older versions of IE. The text becomes larger than it should when made larger, and smaller than it should when made smaller.

Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default font-size in percent for the <body> element:

Example

```
body {
font-size: 100%;
}

h1 {
font-size: 2.5em;
}

h2 {
font-size: 1.875em;
}

p {
font-size: 0.875em;
}
```

Our code now works great! It shows the same text size in all browsers, and allows all browsers to zoom or resize the text!

Font Weight

The `font-weight` property specifies the weight of a font:

Example

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

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

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.

Example

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

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


All CSS Font Properties

Property	Description
font	Sets all the font properties in one declaration
font-family	Specifies the font family for text
font-size	Specifies the font size of text
font-style	Specifies the font style for text
font-variant	Specifies whether or not a text should be displayed in a small-caps font
font-weight	Specifies the weight of a font

CSS Backgrounds Properties

The CSS background properties are used to define the background effects for elements.

CSS background properties:

- background-color
- background-image
- background-repeat

- background-attachment
- background-position

Background Color

The **background-color** property specifies the background color of an element.

The background color of a page is set like this:

Example

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

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)"

Look at [CSS Color Values](#) for a complete list of possible color values.

In the example below, the <h1>, <p>, and <div> elements have different background colors:

Example

```
h1 {  
  background-color: green;  
}  
  
div {  
  background-color: lightblue;  
}  
  
p {  
  background-color: yellow;  
}
```

Background Image

The `background-image` property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

The background image for a page can be set like this:

Example

```
body {  
  background-image: url("paper.gif");  
}
```

Below is an example of a **bad** combination of text and background image. The text is hardly readable:

Example

```
body {  
  background-image: url("bgdesert.jpg");  
}
```

Note: When using a background image, use an image that does not disturb the text.

Background Image - Repeat Horizontally or Vertically

By default, the `background-image` property repeats an image both horizontally and vertically.

Some images should be repeated only horizontally or vertically, or they will look strange, like this:

Example

```
body {  
  background-image: url("gradient_bg.png");  
}
```

If the image above is repeated only horizontally (**background-repeat: repeat-x;**), the background will look better:

Example

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

Tip: To repeat an image vertically, set **background-repeat: repeat-y;**

Background Image - Set position and no-repeat

Showing the background image only once is also specified by the **background-repeat** property:

Example

```
body {  
  background-image: url("img_tree.png");  
  background-repeat: no-repeat;  
}
```

In the example above, the background image is shown in the same place as the text. We want to change the position of the image, so that it does not disturb the text too much.

The position of the image is specified by the **background-position** property:

Example

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

Background Image - Fixed position

To specify that the background image should be fixed (will not scroll with the rest of the page), use the `background-attachment` property:

Example

```
body {  
  background-image: url("img_tree.png");  
  background-repeat: no-repeat;  
  background-position: right top;  
  background-attachment: fixed;  
}
```

Background - Shorthand property

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.

The shorthand property for background is `background`:

Example

```
body {  
  background: #ffffff url("img_tree.png") no-repeat right top;  
}
```

When using the shorthand property the order of the property values is:

- `background-color`
- `background-image`
- `background-repeat`
- `background-attachment`
- `background-position`

It does not matter if one of the property values is missing, as long as the other ones are in this order.

All CSS Background Properties

Property	Description
<u>background</u>	Sets all the background properties in one declaration
<u>background-attachment</u>	Sets whether a background image is fixed or scrolls with the rest of the page
<u>background-color</u>	Sets the background color of an element
<u>background-image</u>	Sets the background image for an element
<u>background-position</u>	Sets the starting position of a background image
<u>background-repeat</u>	Sets how a background image will be repeated

CSS Border Properties

The CSS **border** properties allow you to specify the style, width, and color of an element's border.

Border Style

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

The following values are allowed:

- **dotted** - Defines a dotted border
- **dashed** - Defines a dashed border
- **solid** - Defines a solid border
- **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

The **border-style** property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example

```
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 double;}
```

Result:

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border. The effect depends on the border-color value.

A ridge border. The effect depends on the border-color value.

An inset border. The effect depends on the border-color value.

An outset border. The effect depends on the border-color value.

No border.

A hidden border.

A mixed border.

Note: None of the OTHER CSS border properties described below will have ANY effect unless the `border-style` property is set!

Border Width

The `border-width` property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.

The `border-width` property can have from one to four values (for the top border, right border, bottom border, and the left border).

5px border-width

Example

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



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

```
p.three {  
  border-style: solid;  
  border-width: 2px 10px 4px 20px;  
}
```

Border Color

The `border-color` property is used to set the color of the four borders.

The color can be set by:

- name - specify a color name, like "red"
- Hex - specify a hex value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- transparent

The `border-color` property can have from one to four values (for the top border, right border, bottom border, and the left border).

If `border-color` is not set, it inherits the color of the element.

Red border

Example

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

```
p.two {  
  border-style: solid;  
  border-color: green;
```

```
}
```

```
p.three {  
  border-style: solid;  
  border-color: red green blue yellow;  
}
```

Border - Individual Sides

From the examples above you have seen that it is possible to specify a different border for each side.

In CSS, there are also properties for specifying each of the borders (top, right, bottom, and left):

Different Border Styles

Example

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

The example above gives the same result as this:

Example

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

So, here is how it works:

If the `border-style` property has four values:

- **border-style: dotted solid double dashed;**
 - top border is dotted

- right border is solid
- bottom border is double
- left border is dashed

If the `border-style` property has three values:

- **`border-style: dotted solid double;`**
 - top border is dotted
 - right and left borders are solid
 - bottom border is double

If the `border-style` property has two values:

- **`border-style: dotted solid;`**
 - top and bottom borders are dotted
 - right and left borders are solid

If the `border-style` property has one value:

- **`border-style: dotted;`**
 - all four borders are dotted

The `border-style` property is used in the example above. However, it also works with `border-width` and `border-color`.

Border - Shorthand Property

As you can see from the examples above, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The `border` property is a shorthand property for the following individual border properties:

- `border-width`
- `border-style` (required)
- `border-color`

Example

```
p {  
  border: 5px solid red;  
}
```

Result:

Some text

You can also specify all the individual border properties for just one side:

Left Border

```
p {  
  border-left: 6px solid red;  
  background-color: lightgrey;  
}
```

Result:

Some text

Bottom Border

```
p {  
  border-bottom: 6px solid red;  
  background-color: lightgrey;  
}
```

Result:

Some text

Rounded Borders

The `border-radius` property is used to add rounded borders to an element:

Normal border

Round border

Rounder border

Roudest border

Example

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

Note: The `border-radius` property is not supported in IE8 and earlier versions.

All CSS Border Properties

Property	Description
border	Sets all the border properties in one declaration
border-bottom	Sets all the bottom border properties in one declaration
border-bottom-color	Sets the color of the bottom border

[border-bottom-style](#)

Sets the style of the bottom border

[border-bottom-width](#)

Sets the width of the bottom border

[border-color](#)

Sets the color of the four borders

[border-left](#)

Sets all the left border properties in one declaration

[border-left-color](#)

Sets the color of the left border

[border-left-style](#)

Sets the style of the left border

[border-left-width](#)

Sets the width of the left border

[border-radius](#)

Sets all the four border-*-radius properties for rounded corners

[border-right](#)

Sets all the right border properties in one declaration

[border-right-color](#)

Sets the color of the right border

<u>border-right-style</u>	Sets the style of the right border
<u>border-right-width</u>	Sets the width of the right border
<u>border-style</u>	Sets the style of the four borders
<u>border-top</u>	Sets all the top border properties in one declaration
<u>border-top-color</u>	Sets the color of the top border
<u>border-top-style</u>	Sets the style of the top border
<u>border-top-width</u>	Sets the width of the top border
<u>border-width</u>	Sets the width of the four borders

CSS Margins

The CSS **margin** properties are used to create space around elements, outside of any defined borders.

With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

Margin - Individual Sides

CSS has properties for specifying the margin for each side of an element:

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

All the margin properties can have the following values:

- `auto` - the browser calculates the margin
- *length* - specifies a margin in px, pt, cm, etc.
- `%` - specifies a margin in % of the width of the containing element
- `inherit` - specifies that the margin should be inherited from the parent element

Tip: Negative values are allowed.

The following example sets different margins for all four sides of a `<p>` element:

Example

```
p {  
  margin-top: 100px;  
  margin-bottom: 100px;  
  margin-right: 150px;  
  margin-left: 80px;  
}
```

Margin - Shorthand Property

To shorten the code, it is possible to specify all the margin properties in one property.

The `margin` property is a shorthand property for the following individual margin properties:

- `margin-top`
- `margin-right`

- `margin-bottom`
- `margin-left`

So, here is how it works:

If the `margin` property has four values:

- **`margin: 25px 50px 75px 100px;`**
 - top margin is 25px
 - right margin is 50px
 - bottom margin is 75px
 - left margin is 100px

Example

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

If the `margin` property has three values:

- **`margin: 25px 50px 75px;`**
 - top margin is 25px
 - right and left margins are 50px
 - bottom margin is 75px

Example

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

If the `margin` property has two values:

- **`margin: 25px 50px;`**
 - top and bottom margins are 25px
 - right and left margins are 50px

Example

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

If the `margin` property has one value:

- **margin: 25px;**
 - all four margins are 25px

Example

```
p {  
  margin: 25px;  
}
```

The auto Value

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

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

Example

```
div {  
  width: 300px;  
  margin: auto;  
  border: 1px solid red;  
}
```

The inherit Value

This example lets the left margin of the `<p class="ex1">` element be inherited from the parent element (`<div>`):

Example

```
div {  
  border: 1px solid red;  
  margin-left: 100px;  
}
```

```
p.ex1 {  
  margin-left: inherit;  
}
```

Margin Collapse

Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

This does not happen on left and right margins! Only top and bottom margins!

Look at the following example:

Example

```
h1 {  
  margin: 0 0 50px 0;  
}
```

```
h2 {  
  margin: 20px 0 0 0;  
}
```

In the example above, the `<h1>` element has a bottom margin of 50px and the `<h2>` element has a top margin set to 20px.

Common sense would seem to suggest that the vertical margin between the `<h1>` and the `<h2>` would be a total of 70px (50px + 20px). But due to margin collapse, the actual margin ends up being 50px.

All CSS Margin Properties

Property	Description
margin	A shorthand property for setting the margin properties in one declaration

<u>margin-bottom</u>	Sets the bottom margin of an element
--------------------------------------	--------------------------------------

<u>margin-left</u>	Sets the left margin of an element
------------------------------------	------------------------------------

<u>margin-right</u>	Sets the right margin of an element
-------------------------------------	-------------------------------------

<u>margin-top</u>	Sets the top margin of an element
-----------------------------------	-----------------------------------

CSS Padding

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

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

Padding - Individual Sides

CSS has properties for specifying the padding for each side of an element:

- **padding-top**
- **padding-right**
- **padding-bottom**
- **padding-left**

All the padding properties can have the following values:

- *length* - specifies a padding in px, pt, cm, etc.
- *%* - specifies a padding in % of the width of the containing element

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

Note: Negative values are not allowed.

The following example sets different padding for all four sides of a <div> element:

Example

```
div {  
  padding-top: 50px;  
  padding-right: 30px;  
  padding-bottom: 50px;  
  padding-left: 80px;  
}
```

Padding - Shorthand Property

To shorten the code, it is possible to specify all the padding properties in one property.

The **padding** property is a shorthand property for the following individual padding properties:

- padding-top
- padding-right
- padding-bottom
- padding-left

So, here is how it works:

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

Example

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

If the **padding** property has three values:

- **padding: 25px 50px 75px;**
 - top padding is 25px
 - right and left paddings are 50px
 - bottom padding is 75px

Example

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

If the **padding** property has two values:

- **padding: 25px 50px;**
 - top and bottom paddings are 25px
 - right and left paddings are 50px

Example

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

If the **padding** property has one value:

- **padding: 25px;**
 - all four paddings are 25px

Example

```
div {  
  padding: 25px;  
}
```

Padding and Element Width

The CSS `width` property specifies the width of the element's content area. The content area is the portion inside the padding, border, and margin of an element ([the box model](#)).

So, if an element has a specified width, the padding added to that element will be added to the total width of the element. This is often an undesirable result.

In the following example, the `<div>` element is given a width of 300px. However, the actual rendered width of the `<div>` element will be 350px (300px + 25px of left padding + 25px of right padding):

Example

```
div {  
  width: 300px;  
  padding: 25px;  
}
```

To keep the width at 300px, no matter the amount of padding, you can use the `box-sizing` property. This causes the element to maintain its width; if you increase the padding, the available content space will decrease. Here is an example:

Example

```
div {  
  width: 300px;  
  padding: 25px;  
  box-sizing: border-box;  
}
```

All CSS Padding Properties

Property	Description
padding	A shorthand property for setting all the padding properties

	in one declaration
padding-bottom	Sets the bottom padding of an element
padding-left	Sets the left padding of an element
padding-right	Sets the right padding of an element
padding-top	Sets the top padding of an element

CSS Height and Width

The **height** and **width** properties are used to set the height and width of an element.

The **height** and **width** can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block.

This element has a height of 200 pixels and a width of 50%

Example

```
div {
  height: 200px;
  width: 50%;
  background-color: powderblue;
}
```

This element has a height of 100 pixels and a width of 500 pixels.

Example


```
div {  
  height: 100px;  
  width: 500px;  
  background-color: powderblue;  
}
```

Note: The `height` and `width` properties do not include padding, borders, or margins; they set the height/width of the area inside the padding, border, and margin of the element!

Setting max-width

The `max-width` property is used to set the maximum width of an element.

The `max-width` can be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).

The problem with the `<div>` above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.

Using `max-width` instead, in this situation, will improve the browser's handling of small windows.

Tip: Drag the browser window to smaller than 500px wide, to see the difference between the two divs!

This element has a height of 100 pixels and a max-width of 500 pixels.

Note: The value of the `max-width` property overrides `width`.

The following example shows a `<div>` element with a height of 100 pixels and a max-width of 500 pixels:

Example

```
div {  
  max-width: 500px;  
  height: 100px;
```

```
background-color: powderblue;
}
```

All CSS Dimension Properties

Property	Description
height	Sets the height of an element
max-height	Sets the maximum height of an element
max-width	Sets the maximum width of an element
min-height	Sets the minimum height of an element
min-width	Sets the minimum width of an element
width	Sets the width of an element

The 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:

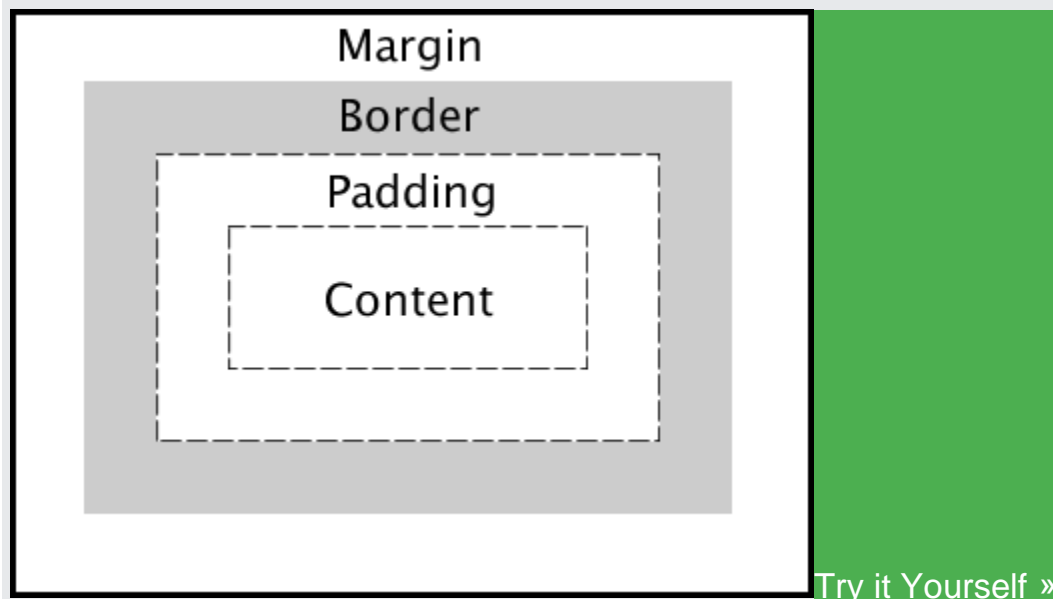
Explanation of the different parts:

- **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

The box model allows us to add a border around elements, and to define space between elements.

Example

Demonstration of the box model:



CSS Outline Properties

An outline is a line that is drawn around elements, **OUTSIDE** the borders, to make the element "stand out".

CSS has the following outline properties:

- **outline-style**
- **outline-color**

- `outline-width`
- `outline-offset`
- `outline`

Note: Outline differs from borders! Unlike border, the outline is drawn outside the element's border, and may overlap other content. Also, the outline is NOT a part of the element's dimensions; the element's total width and height is not affected by the width of the outline.

Outline Style

The `outline-style` property specifies the style of the outline, and can have one of the following values:

- `dotted` - Defines a dotted outline
- `dashed` - Defines a dashed outline
- `solid` - Defines a solid outline
- `double` - Defines a double outline
- `groove` - Defines a 3D grooved outline
- `ridge` - Defines a 3D ridged outline
- `inset` - Defines a 3D inset outline
- `outset` - Defines a 3D outset outline
- `none` - Defines no outline
- `hidden` - Defines a hidden outline

Example

```
p.dotted {outline-style: dotted;}
p.dashed {outline-style: dashed;}
p.solid {outline-style: solid;}
p.double {outline-style: double;}
p.groove {outline-style: groove;}
p.ridge {outline-style: ridge;}
p.inset {outline-style: inset;}
p.outset {outline-style: outset;}
```

Note: None of the other outline properties will have any effect, unless the `outline-style` property is set!

Outline Color

The `outline-color` property is used to set the color of the outline.

The color can be set by:

- name - specify a color name, like "red"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- Hex - specify a hex value, like "#ff0000"
- invert - performs a color inversion (which ensures that the outline is visible, regardless of color background)

The following example shows some different outlines with different colors. Also notice that these elements also have a thin black border inside the outline:

A solid red outline.

A double green outline.

An outset yellow outline.

Example

```
p.ex1 {  
  border: 1px solid black;  
  outline-style: solid;  
  outline-color: red;  
}
```

```
p.ex2 {  
  border: 1px solid black;  
  outline-style: double;  
  outline-color: green;  
}
```

```
p.ex3 {
```

```
border: 1px solid black;
outline-style: outset;
outline-color: yellow;
}
```

The following example uses `outline-color: invert`, which performs a color inversion. This ensures that the outline is visible, regardless of color background:

A solid invert outline.

Example

```
p.ex1 {
  border: 1px solid yellow;
  outline-style: solid;
  outline-color: invert;
}
```

Outline Width

The `outline-width` property specifies the width of the outline, and can have one of the following values:

- thin (typically 1px)
- medium (typically 3px)
- thick (typically 5px)
- A specific size (in px, pt, cm, em, etc)

The following example shows some outlines with different widths:

A thin outline.

A medium outline.

A thick outline.

A 4px thick outline.

Example

```
p.ex1 {
  border: 1px solid black;
```

```
outline-style: solid;
outline-color: red;
outline-width: thin;
}

p.ex2 {
border: 1px solid black;
outline-style: solid;
outline-color: red;
outline-width: medium;
}

p.ex3 {
border: 1px solid black;
outline-style: solid;
outline-color: red;
outline-width: thick;
}

p.ex4 {
border: 1px solid black;
outline-style: solid;
outline-color: red;
outline-width: 4px;
}
```

Outline - Shorthand property

The **outline** property is a shorthand property for setting the following individual outline properties:

- **outline-width**
- **outline-style** (required)
- **outline-color**

The **outline** property is specified as one, two, or three values from the list above. The order of the values does not matter.

.

Example

```
p.ex1 {outline: dashed;}  
p.ex2 {outline: dotted red;}  
p.ex3 {outline: 5px solid yellow;}  
p.ex4 {outline: thick ridge pink;}
```

Outline Offset

The `outline-offset` property adds space between an outline and the edge/border of an element. The space between an element and its outline is transparent.

The following example specifies an outline 15px outside the border edge:

This paragraph has an outline 15px outside the border edge.

Example

```
p {  
  margin: 30px;  
  border: 1px solid black;  
  outline: 1px solid red;  
  outline-offset: 15px;  
}
```

The following example shows that the space between an element and its outline is transparent:

This paragraph has an outline 15px outside the border edge.

Example

```
p {  
  margin: 30px;  
  background: yellow;  
  border: 1px solid black;  
  outline: 1px solid red;
```



```
outline-offset: 15px;  
}
```

All CSS Outline Properties

Property	Description
outline	A shorthand property for setting outline-width, outline-style, and outline-color in one declaration
outline-color	Sets the color of an outline
outline-offset	Specifies the space between an outline and the edge or border of an element
outline-style	Sets the style of an outline
outline-width	Sets the width of an outline

CSS Lists Properties

HTML Lists and CSS List Properties

In HTML, there are two main types of lists:

- unordered lists () - the list items are marked with bullets
- ordered lists () - the list items are marked with numbers or letters

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

Different List Item Markers

The `list-style-type` property specifies the type of list item marker.

The following example shows some of the available list item markers:

Example

```
ul.a {  
  list-style-type: circle;  
}  
  
ul.b {  
  list-style-type: square;  
}  
  
ol.c {  
  list-style-type: upper-roman;  
}  
  
ol.d {  
  list-style-type: lower-alpha;  
}
```

Note: Some of the values are for unordered lists, and some for ordered lists.

An Image as The List Item Marker

The `list-style-image` property specifies an image as the list item marker:

Example

```
ul {  
  list-style-image: url('sqpurple.gif');  
}
```

Position The List Item Markers

The `list-style-position` property specifies whether the list-item markers should appear inside or outside the content flow:

Example

```
ul {  
    list-style-position: inside;  
}
```

Remove Default Settings

The `list-style-type:none` property can also be used to remove the markers/bullets. Note that the list also has default margin and padding. To remove this, add `margin:0` and `padding:0` to `` or ``:

Example

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
}
```

List - Shorthand property

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

Example

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

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 are missing, the default value for the missing property will be inserted, if any.

Styling List With Colors

We can also style lists with colors, to make them look a little more interesting.

Anything added to the `` or `` tag, affects the entire list, while properties added to the `` tag will affect the individual list items:

Example

```
ol {  
  background: #ff9999;  
  padding: 20px;  
}
```

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

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

```
ul li {  
  background: #cce5ff;  
  margin: 5px;  
}
```

Result:

1. Coffee
2. Tea
3. Coca Cola

- Coffee

- Tea
- Coca Cola

All CSS List Properties

Property	Description
list-style	Sets all the properties for a list in one declaration
list-style-image	Specifies an image as the list-item marker
list-style-position	Specifies if the list-item markers should appear inside or outside the content flow
list-style-type	Specifies the type of list-item marker

CSS Tables roperities

The look of an HTML table can be greatly improved with CSS:

Company	Contact
Alfreds Futterkiste	Maria Anders
Berglunds snabbköp	Christina Berglund

Centro comercial Moctezuma	Francisco Chang
Ernst Handel	Roland Mendel
Island Trading	Helen Bennett
Königlich Essen	Philip Cramer
Laughing Bacchus Winecellars	Yoshi Tannamuri
Magazzini Alimentari Riuniti	Giovanni Rovelli

Table Borders

To specify table borders in CSS, use the `border` property.

The example below specifies a black border for `<table>`, `<th>`, and `<td>` elements:

Example

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

Notice that the table in the example above has double borders. This is because both the table and the `<th>` and `<td>` elements have separate borders.

Collapse Table Borders

The `border-collapse` property sets whether the table borders should be collapsed into a single border:

Example

```
table {  
  border-collapse: collapse;  
}
```

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

If you only want a border around the table, only specify the `border` property for `<table>`:

Example

```
table {  
  border: 1px solid black;  
}
```

Table Width and Height

Width and height of a table are defined by the `width` and `height` properties.

The example below sets the width of the table to 100%, and the height of the `<th>` elements to 50px:

Example

```
table {  
  width: 100%;  
}  
  
th {  
  height: 50px;  
}
```

Horizontal Alignment

The `text-align` property sets the horizontal alignment (like left, right, or center) of the content in `<th>` or `<td>`.

By default, the content of `<th>` elements are center-aligned and the content of `<td>` elements are left-aligned.

The following example left-aligns the text in `<th>` elements:

Example

```
th {  
    text-align: left;  
}
```

Vertical Alignment

The **vertical-align** property sets the vertical alignment (like top, bottom, or middle) of the content in `<th>` or `<td>`.

By default, the vertical alignment of the content in a table is middle (for both `<th>` and `<td>` elements).

The following example sets the vertical text alignment to bottom for `<td>` elements:

Example

```
td {  
    height: 50px;  
    vertical-align: bottom;  
}
```

Table Padding

To control the space between the border and the content in a table, use the **padding** property on `<td>` and `<th>` elements:

Example

```
th, td {  
    padding: 15px;  
    text-align: left;  
}
```

Horizontal Dividers

First Name	Last Name	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Add the `border-bottom` property to `<th>` and `<td>` for horizontal dividers:

Example

```
th, td {
  border-bottom: 1px solid #ddd;
}
```

Hoverable Table

Use the `:hover` selector on `<tr>` to highlight table rows on mouse over:

First Name	Last Name	Savings
Peter	Griffin	\$100

Lois	Griffin	\$150
Joe	Swanson	\$300

Example

`tr:hover { background-color: #f5f5f5; }`

Striped Tables

First Name	Last Name	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

For zebra-striped tables, use the `nth-child()` selector and add a `background-color` to all even (or odd) table rows:

Example

```
tr:nth-child(even) {background-color: #f2f2f2;}
```

Table Color

The example below specifies the background color and text color of <th> elements:

First Name	Last Name	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Example

```
th {
    background-color: #4CAF50;
    color: white;
}
```

Responsive Table

A responsive table will display a horizontal scroll bar if the screen is too small to display the full content:

First	Last	Points	Points	Points	Points	Points	Points	Points	Points	P
1	2	3	4	5	6	7	8	9	10	11

Name	Name									
Jill	Smith	50	50	50	50	50	50	50	50	50
Eve	Jackson	94	94	94	94	94	94	94	94	94
Adam	Johnson	67	67	67	67	67	67	67	67	67

Add a container element (like `<div>`) with `overflow-x:auto` around the `<table>` element to make it responsive:

Example

```
<div style="overflow-x:auto;">
```

```
<table>
```

```
... table content ...
```

```
</table>
```

```
</div>
```

Note: In OS X Lion (on Mac), scrollbars are hidden by default and only shown when being used (even though "overflow:scroll" is set).

All CSS Table Properties

Property	Description
border	Sets all the border properties in one declaration
border-	Specifies whether or not table borders should be collapsed

[collapse](#)

[border-spacing](#)

Specifies the distance between the borders of adjacent cells

[caption-side](#)

Specifies the placement of a table caption

[empty-cells](#)

Specifies whether or not to display borders and background on empty cells in a table

[table-layout](#)

Sets the layout algorithm to be used for a table

CSS Links Properties

With CSS, links can be styled in different ways.

[Text Link](#) [Text Link](#) [Link Button](#) [Link Button](#)

Styling Links

Links can be styled with any CSS property (e.g. `color`, `font-family`, `background`, etc.).

Example

```
a {  
  color: hotpink;  
}
```

In addition, links can be styled differently depending on what **state** they are in.

The four links states are:

- `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

Example

```
/* unvisited link */
```

```
a:link {  
  color: red;  
}
```

```
/* visited link */
```

```
a:visited {  
  color: green;  
}
```

```
/* mouse over link */
```

```
a:hover {  
  color: hotpink;  
}
```

```
/* selected link */
```

```
a:active {  
  color: blue;  
}
```

When setting the style for several link states, there are some order rules:

- `a:hover` MUST come after `a:link` and `a:visited`
- `a:active` MUST come after `a:hover`

Text Decoration

The `text-decoration` property is mostly used to remove underlines from links:

Example

```
a:link {
    text-decoration: none;
}

a:visited {
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

a:active {
    text-decoration: underline;
}
```

Background Color

The `background-color` property can be used to specify a background color for links:

Example

```
a:link {
    background-color: yellow;
}

a:visited {
    background-color: cyan;
}

a:hover {
    background-color: lightgreen;
}

a:active {
    background-color: hotpink;
}
```

Advanced - Link Buttons

This example demonstrates a more advanced example where we combine several CSS properties to display links as boxes/buttons:

Example

```
a:link, a:visited {  
    background-color: #f44336;  
    color: white;  
    padding: 14px 25px;  
    text-align: center;  
    text-decoration: none;  
    display: inline-block;  
}  
  
a:hover, a:active {  
    background-color: red;  
}
```

CSS Layout - The display Property

The **display** property is the most important CSS property for controlling layout.

The **display** property specifies if/how an element is displayed.

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

Click to show panel

Block-level Elements

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

The `<div>` element is a block-level element.

Examples of block-level elements:

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

Inline Elements

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

This is an inline `` element inside a paragraph.

Examples of inline elements:

- ``
- `<a>`
- ``

Display: none;

`display: none;` is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.

The `<script>` element uses `display: none;` as default.

Override The Default Display Value

As mentioned, every element has a default display value. However, you can override this.

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.

A common example is making inline `` elements for horizontal menus:

Example

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

Note: Setting the display property of an element only changes **how the element is displayed**, NOT what kind of element it is. So, an inline element with `display: block`; is not allowed to have other block elements inside it.

The following example displays `` elements as block elements:

Example

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

The following example displays `<a>` elements as block elements:

Example

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

Hide an Element - `display:none` or `visibility:hidden`?

`display:none`



Remove

visibility:hidden



Hide

Reset



Reset All

Hiding an element can be done by setting the `display` property to `none`. The element will be hidden, and the page will be displayed as if the element is not there:

Example

```
h1.hidden {  
  display: none;  
}
```

`visibility:hidden`; also hides an element.

However, the element will still take up the same space as before. The element will be hidden, but still affect the layout:

Example

```
h1.hidden {  
  visibility: hidden;  
}
```

CSS Display/Visibility Properties

Property	Description
display	Specifies how an element should be displayed
visibility	Specifies whether or not an element should be visible

CSS Layout - The position Property

The **position** property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

The 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**

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the **position** property is set first. They also work differently depending on the position value.

position: static;

HTML elements are positioned static by default.

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

An element with **position: static;** is not positioned in any special way; it is always positioned according to the normal flow of the page:

This <div> element has position: static;

Here is the CSS that is used:

Example

```
div.static {  
    position: static;  
    border: 3px solid #73AD21;  
}
```

position: relative;

An element with **position: relative;** 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.

This <div> element has position: relative;

Here is the CSS that is used:

Example

```
div.relative {  
    position: relative;  
    left: 30px;  
    border: 3px solid #73AD21;  
}
```


position: fixed;

An element with **position: fixed;** 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.

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

Example

```
div.fixed {  
  position: fixed;  
  bottom: 0;  
  right: 0;  
  width: 300px;  
  border: 3px solid #73AD21;  
}
```

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.

Note: A "positioned" element is one whose position is anything except **static**.

Here is a simple example:

This <div> element has position: relative;

This <div> element has position: absolute;

Here is the CSS that is used:

Example

```
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;  
}
```

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).

Note: Internet Explorer, Edge 15 and earlier versions do not support sticky positioning. Safari requires a -webkit- prefix (see example below). You must also specify at least one of **top**, **right**, **bottom** or **left** for sticky positioning to work.

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

Example

```
div.sticky {  
  position: -webkit-sticky; /* Safari */  
  position: sticky;  
  top: 0;  
  background-color: green;
```

```
border: 2px solid #4CAF50;
}
```

Overlapping Elements

When elements are positioned, they can overlap other elements.

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

An element can have a positive or negative stack order:

This is a heading



Because the image has a z-index of -1, it will be placed behind the text.

Example

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

An element with greater stack order is always in front of an element with a lower stack order.

Note: If two positioned elements overlap without a **z-index** specified, the element positioned last in the HTML code will be shown on top.

Positioning Text In an Image

How to position text over an image:

Example



Bottom Left

Top Left

Top Right

Bottom Right

Centered

All CSS Positioning Properties

Property	Description
bottom	Sets the bottom margin edge for a positioned box
clip	Clips an absolutely positioned element

left	Sets the left margin edge for a positioned box
position	Specifies the type of positioning for an element
right	Sets the right margin edge for a positioned box
top	Sets the top margin edge for a positioned box
z-index	Sets the stack order of an element

CSS Layout - Overflow

The CSS **overflow** property controls what happens to content that is too big to fit into an area.

This text is really long and the height of its container is only 100 pixels. Therefore, a scrollbar is added to help the reader to scroll the content. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum laritatem.

CSS Overflow

The `overflow` property specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area.

The `overflow` property has the following values:

- `visible` - Default. The overflow is not clipped. It renders outside the element's box
- `hidden` - The overflow is clipped, and the rest of the content will be invisible
- `scroll` - The overflow is clipped, but a scrollbar is added to see the rest of the content
- `auto` - If overflow is clipped, a scrollbar should be added to see the rest of the content

Note: The `overflow` property only works for block elements with a specified height.

Note: In OS X Lion (on Mac), scrollbars are hidden by default and only shown when being used (even though "overflow:scroll" is set).

overflow: visible

By default, the overflow is `visible`, meaning that it is not clipped and it renders outside the element's box:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

Example

```
div {  
  width: 200px;  
  height: 50px;  
  background-color: #eee;  
  overflow: visible;  
}
```

overflow: hidden

With the **hidden** value, the overflow is clipped, and the rest of the content is hidden:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

Example

```
div {  
  overflow: hidden;  
}
```

overflow: scroll

Setting the value to **scroll**, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

Example

```
div {  
  overflow: scroll;  
}
```

overflow: auto

The **auto** value is similar to **scroll**, only it add scrollbars when necessary:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

Example

```
div {  
  overflow: auto;  
}
```

overflow-x and overflow-y

The **overflow-x** and **overflow-y** properties specifies whether to change the overflow of content just horizontally or vertically (or both):

overflow-x specifies what to do with the left/right edges of the content.
overflow-y specifies what to do with the top/bottom edges of the content.

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

Example

```
div {  
  overflow-x: hidden; /* Hide horizontal scrollbar */  
  overflow-y: scroll; /* Add vertical scrollbar */  
}
```

All CSS Overflow Properties

Property	Description
overflow	Specifies what happens if content overflows an element's box
overflow-x	Specifies what to do with the left/right edges of the content if it overflows the element's content area
overflow-y	Specifies what to do with the top/bottom edges of the content if it overflows the element's content area

CSS Layout - float and clear

The CSS **float** property specifies how an element should float.

The CSS **clear** property specifies what elements can float beside the cleared element and on which side.

The float Property

The **float** property is used for positioning and layout on web pages.

The **float** property can have one of the following values:

- left - The element floats to the left of its container
- right- The element floats to the right of its container
- none - The element does not float (will be displayed just where it occurs in the text). This is default
- inherit - The element inherits the float value of its parent

In its simplest use, the **float** property can be used to wrap text around images.

Example - float: right;

The following example specifies that an image should float to the **right** in a text:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec

congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac...

Example

```
img {  
  float: right;  
}
```

Example - float: left;

The following example specifies that an image should float to the **left** in a text:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac...

Example

```
img {  
  float: left;  
}
```

Example - No float

In the following example the image will be displayed just where it occurs in the text (float: none;):



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac...

Example

```
img {  
  float: none;  
}
```

The clear Property

The **clear** property lets specifies what elements can float beside the cleared element and on which side.

The **clear** property can have one of the following values:

- none - Allows floating elements on both sides. This is default
- left - No floating elements allowed on the left side

- right- No floating elements allowed on the right side
- both - No floating elements allowed on either the left or the right side
- inherit - The element inherits the clear value of its parent

The most common way to use the `clear` property is after you have used a `float` property on an element.

When clearing floats, you should match the clear to the float. If an element is floated to the left, then you should clear to the left. Your floated element will continue to float, but the cleared element will appear below it on the web page.

The following example clears the float to the left. Means that no floating elements are allowed on the left side (of the div):

Example

```
div {  
  
clear: left;  
}
```

The clearfix Hack

If an element is taller than the element containing it, and it is floated, it will overflow outside of its container.

Without Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...



With Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...



Then we can add `overflow: auto;` to the containing element to fix this problem:

Example

```
.clearfix {  
    overflow: auto;  
}
```

The `overflow:auto` clearfix works well as long as you are able to keep control of your margins and padding (else you might see scrollbars). The **new, modern clearfix hack** however, is safer to use, and the following code is used for most webpages:

Example

```
.clearfix::after {  
    content: "";  
    clear: both;  
    display: table;  
}
```

You will learn more about the `::after` pseudo-element in a later chapter.

Web Layout Example

It is common to do entire web layouts using the `float` property:

Example

```
.header, .footer {  
    background-color: grey;  
}
```

```
color: white;
padding: 15px;
}
```

```
.column {
float: left;
padding: 15px;
}
```

```
.clearfix::after {
content: "";
clear: both;
display: table;
}
```

```
.menu {
width: 25%;
}
```

```
.content {
width: 75%;
}
```

All CSS Float Properties

Property	Description
clear	Specifies what elements can float beside the cleared element and on which side
float	Specifies how an element should float

[overflow](#) Specifies what happens if content overflows an element's box

[overflow-x](#) Specifies what to do with the left/right edges of the content if it overflows the element's content area

[overflow-y](#) Specifies what to do with the top/bottom edges of the content if it overflows the element's content area

CSS3 supported to add shadow to text or elements. Shadow property has divided as follows

- Text shadow
- Box Shadow

Text shadow

CSS3 supported to add shadow effects to text. Following is the example to add shadow effects to text

```
<html>

<head>

  <style>

    h1 {

      text-shadow: 2px 2px;

    }

    h2 {

      text-shadow: 2px 2px red;

    }
```

```
h3 {  
  text-shadow: 2px 2px 5px red;  
}  
h4 {  
  color: white;  
  text-shadow: 2px 2px 4px #000000;  
}  
h5 {  
  text-shadow: 0 0 3px #FF0000;  
}  
h6 {  
  text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;  
}  
p {  
  color: white;  
  text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1>Tutorialspoint.com</h1>
```

```
<h2>Tutorialspoint.com</h2>
```

```
<h3>Tutorialspoint.com</h3>
```

```
<h4>Tutorialspoint.com</h4>
```

```
<h5>Tutorialspoint.com</h5>
```

```
<h6>Tutorialspoint.com</h6>
```

```
<p>Tutorialspoint.com</p>
```

```
</body>
```

```
</html>
```

box shadow

Used to add shadow effects to elements,Following is the example to add shadow effects to element

```
<html>
```

```
<head>
```

```
<style>
```

```
div {
```

```
width: 300px;
```

```
height: 100px;
```

```
padding: 15px;
```

```
background-color: red;
```

```
box-shadow: 10px 10px;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```



```
<div>This is a div element with a box-shadow</div>

</body>

</html>
```

CSS Border image property is used to add image boarder to some elements.you don't need to use any HTML code to call boarder image.A sample syntax of boarder image is as follows –

```
#borderimg {
  border: 10px solid transparent;
  padding: 15px;
}
```

The most commonly used values are shown below –

Values	Description
border-image-source	Used to set the image path
border-image-slice	Used to slice the boarder image
border-image-width	Used to set the boarder image width
border-image-repeat	Used to set the boarder image as rounded, repeated and stretched

Example

Following is the example which demonstrates to set image as a border for elements

```
<html>

<head>
```

<style>

```
#borderimg1 {  
    border: 10px solid transparent;  
    padding: 15px;  
    border-image-source: url(/css/images/border.png);  
    border-image-repeat: round;  
    border-image-slice: 30;  
    border-image-width: 10px;  
}
```

```
#borderimg2 {  
    border: 10px solid transparent;  
    padding: 15px;  
    border-image-source: url(/css/images/border.png);  
    border-image-repeat: round;  
    border-image-slice: 30;  
    border-image-width: 20px;  
}
```

```
#borderimg3 {  
    border: 10px solid transparent;  
    padding: 15px;  
    border-image-source: url(/css/images/border.png);  
    border-image-repeat: round;  
    border-image-slice: 30;
```

```
        border-image-width: 30px;
    }
</style>

</head>

<body>

    <p id="borderimg1">This is image boarder example.</p>
    <p id="borderimg2">This is image boarder example.</p>
    <p id="borderimg3">This is image boarder example.</p>

</body>

</html>
```

CSS Attribute Selectors

Style HTML Elements With Specific Attributes

It is possible to style HTML elements that have specific attributes or attribute values.

CSS [attribute] Selector

The `[attribute]` selector is used to select elements with a specified attribute.

The following example selects all `<a>` elements with a target attribute:

Example

```
a[target] {
    background-color: yellow;
}
```

CSS [attribute="value"] Selector

The `[attribute="value"]` selector is used to select elements with a specified attribute and value.

The following example selects all <a> elements with a target="_blank" attribute:

Example

```
a[target="_blank"] {  
    background-color: yellow;  
}
```

CSS [attribute~="value"] Selector

The [attribute~="value"] selector is used to select elements with an attribute value containing a specified word.

The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

Example

```
[title~="flower"] {  
    border: 5px solid yellow;  
}
```

The example above will match elements with title="flower", title="summer flower", and title="flower new", but not title="my-flower" or title="flowers".

CSS [attribute|= "value"] Selector

The [attribute|= "value"] selector is used to select elements with the specified attribute starting with the specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value has to be a whole word, either alone, like class="top", or followed by a hyphen(-), like class="top-text"!

Example

```
[class|= "top"] {  
    background: yellow;  
}
```

CSS [attribute^="value"] Selector

The `[attribute^="value"]` selector is used to select elements whose attribute value begins with a specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value does not have to be a whole word!

Example

```
[class^="top"] {  
    background: yellow;  
}
```

CSS `[attribute$="value"]` Selector

The `[attribute$="value"]` selector is used to select elements whose attribute value ends with a specified value.

The following example selects all elements with a class attribute value that ends with "test":

Note: The value does not have to be a whole word!

Example

```
[class$="test"] {  
    background: yellow;  
}
```

CSS `[attribute*="value"]` Selector

The `[attribute*="value"]` selector is used to select elements whose attribute value contains a specified value.

The following example selects all elements with a class attribute value that contains "te":

Note: The value does not have to be a whole word!

Example

```
[class*="te"] {  
  background: yellow;  
}
```

The attribute selectors can be useful for styling forms without class or ID:

Example

```
input[type="text"] {  
  width: 150px;  
  display: block;  
  margin-bottom: 10px;  
  background-color: yellow;  
}
```

```
input[type="button"] {  
  width: 120px;  
  margin-left: 35px;  
  display: block;  
}
```