Chapter – 4

Style Sheet

Introduction to CSS:

CSS was invited by Håkon Wium Lie on October 10, 1994 and maintained through a group of people within the W3C called the CSS Working Group. **C**ascading **S**tyle **S**heets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, we 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, and 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.

Advantages:

1. **CSS Saves Time:**

We can write CSS once and then reuse same sheet in multiple HTML pages. We can define a style for each HTML element and apply it to as many Web pages as we want.

2. **Pages Load Faster:**

If we are using CSS, we 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.

3. **Easy Maintenance:**

To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.

4. **Superior Styles To HTML**:

CSS has a much wider array of attributes than HTML, so we can give a far better look to our HTML page in comparison to HTML attributes.

5. **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.

6. **Global web standards**:

Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

7. **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.

8. **Platform Independence**:

The Script offer consistent platform independence and can support latest browsers as well.

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.

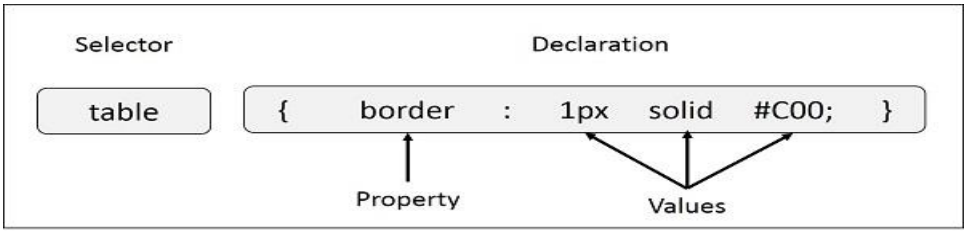
SYNTAX:

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.

We can put CSS Style Rule Syntax as follows:

selector { property: value }



Example: We 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. We can define selectors in various simple ways based on our comfort.

CSS COMMENTS:

Many times, we may need to put additional comments in our style sheet blocks. So, it is very easy to comment any part in style sheet. We can simply put our comments inside /*.....this is a comment in style sheet.....*/.

We can use /* ....*/ to comment multi-line blocks in similar way we do in C and C++ programming languages.

Example

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p {
        color: red;
        /* This is a single-line comment */
        text-align: center;
      }
      /* This is a multi-line comment */
    </style>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

TYPES OF SELECTORS:

To select html elements in an efficient way and provide more control over selection of html elements, CSS provides various types of selectors. Selectors allow us to locate or select html elements on the page as much as generically and as much as specifically and apply styles on them.

1. The Type/Tag Selectors:

Type/tag selectors are the name of the html tags as shown in example below:

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

2. The ID Selectors:

We can define style rules based on the *id* attribute of the elements. All the elements having that *id* will be formatted according to the defined rule.

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

This rule renders the content in black for every element with *id* attribute set to *black* in our document. We 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*.

3. The Class Selectors:

We can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

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

This rule renders the content in black for every element with class attribute set to *black* in our document. We 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*. We 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>
```

4. Contextual Selector:

Contextual selectors define styles that are only applied when certain tags are nested within other tags.

Contextual selectors are merely strings of two or more simple selectors separated by white space. These selectors can be assigned normal properties and, due to the rules of cascading order, they will take precedence over simple selectors. For example, the contextual selector in

```
p em {
    background: yellow;
}
```

a. The Descendant Selectors:

Suppose we 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 <ul> tag.

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

b. The Child Selectors:

We 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 or <td> would not have any effect of this rule.

c. General Sibling Selectors:

A selector that uses a general sibling selector matches elements based on sibling relationships. That is to say, the selected elements are beside each other in the HTML.

```
h2 ~ p {
  color:#009900;
}
```

This type of selector is declared using the tilde character (~). In this example, all paragraph elements (<p>) will be styled with the specified rules, but only if they are siblings of <h2> elements. There could be other elements in between the <h2> and <p>, and the styles would still apply.

Example:

```
<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<div class="box">
  <p>Paragraph example.</p>
</div>
```

In this example, the styles will apply only to the first three paragraph elements. The last paragraph element is not a sibling of the <h2>element because it sits inside the element.

d. <u>Adjacent Sibling Selector:</u>

A selector that uses the adjacent sibling uses the plus symbol (+), and is almost the same as the general sibling selector. The difference is that the targeted element must be an immediate sibling, not just a general sibling.

```
p + p {
  color:red;
}
```

This example will apply the specified styles only to paragraph elements that immediately follow other paragraph elements. This means the first paragraph element on a page would not receive these styles. Also, if another element appeared between two paragraphs, the second paragraph of the two wouldn't have the styles applied.

So, if we apply this selector to the following HTML:

```
<h2>Title</h2>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<p>Paragraph example.</p>
<div class="box">
    <p>Paragraph example.</p>
    <p>Paragraph example.</p>
</div>
```

The styles will apply only to the second, third, and fifth paragraphs in this section of HTML.

5. The Attribute Selectors:

We 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: #009900;
}
```

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-".

6. Multiple Style Rules:

We may need to define multiple style rules for a single element. We 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 (;)**. We can keep them in a single line or multiple lines. For better readability we keep them into separate lines.

7. Grouping Selectors:

We can apply a style to many selectors if we 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.

We can combine the various id selectors together as shown below

```
#content, #footer, #supplement {
  position: absolute;
  left: 510px;
  width: 200px;
}
```

8. 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.

Pseudo Class:

CSS pseudo-classes are used to add special effects to some selectors. We do not need to use JavaScript or any other script to use those effects. A simple syntax of pseudo-classes is as follows:

```
selector:pseudo-class {property: value}
```

CSS classes can also be used with pseudo-classes:

```
selector.class:pseudo-class {property: value}
```

The most commonly used pseudo-classes are as follows:

Value	Description
:link	Use this class to add special style to an unvisited link.
:visited	Use this class to add special style to a visited link.
:hover	Use this class to add special style to an element when you mouse over it.
:active	Use this class to add special style to an active element.
:focus	Use this class to add special style to an element while the element has focus.
:first-child	Use this class to add special style to an element that is the first child of some other element.
:lang	Use this class to specify a language to use in a specified element.

While defining pseudo-classes in a <style>...</style> block, following points should be noted:

- a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.
- a:active MUST come after a:hover in the CSS definition in order to be effective.
- Pseudo-class names are not case-sensitive.
- Pseudo-class are different from CSS classes but they can be combined.

Example:

```
<!DOCTYPE HTML>
<html>
  <head><title>Use of Pseudo-class</title>
    <style type="text/css" lang = "en">
            a:link {color:yellow;}
            a:visited {color:lime;}
            a:hover {color:blue; background-color:wheat; font-size:25px;}
```

```
            a:active {color:green;}
            a:focus {color:cyan;}
    </style>
  </head>
  <body>
    <a href=".\FirstProgram.html" target = "_blank">Click The Link</a>
  </body>
</html>
```

☑ **The :first-child pseudo-class:**

The *:first-child* pseudo-class matches a specified element that is the first child of another element and adds special style to that element that is the first child of some other element. To make :first-child work in IE <!DOCTYPE> must be declared at the top of document.

For example, to indent the first paragraph of all elements, you could use this definition:

```
<html>
  <head>
    <style type="text/css">
      div > p:first-child
      {
        text-indent: 25px;
      }
    </style>
  </head>
  <body>
    <div>
      <p>First paragraph in div. This paragraph will be indented</p>
      <p>Second paragraph in div. This paragraph will not be indented</p>
    </div>
    <p>But it will not match the paragraph in this HTML:</p>
    <div>
      <h3>Heading</h3>
      <p>The first paragraph inside the div. This paragraph will not be effected.</p>
    </div>
  </body>
</html>
```

THE :LANG PSEUDO-CLASS:

The language pseudo-class *:lang*, allows constructing selectors based on the language setting for specific tags.

This class is useful in documents that must appeal to multiple languages that have different conventions for certain language constructs. For example, the French language typically uses

angle brackets (< and >) for quoting purposes, while the English language uses quote marks (' and ').

In a document that needs to address this difference, we can use the :lang pseudo-class to change the quote marks appropriately. The following code changes the <blockquote> tag appropriately for the language being used –

```
<html>
  <head>
    <style type="text/css">
      /* Two levels of quotes for two languages*/
      :lang(en) { quotes: '"' '"'  "'"  "'"; }
      :lang(fr) { quotes: "<<" ">>" "<" ">"; }
    </style>
  </head>
  <body>
    <p>...<q lang="fr">A quote in a paragraph</q>...</p>
  </body>
</html>
```

The :lang selectors will apply to all the elements in the document. However, not all elements make use of the quotes property, so the effect will be transparent for most elements.

PSEUDO ELEMENTS:

CSS pseudo-elements are used to add special effects to some selectors. We do not need to use JavaScript or any other script to use those effects. A simple syntax of pseudo-element is as follows:

```
selector:pseudo-element {property: value}
```

CSS classes can also be used with pseudo-elements:

```
selector.class:pseudo-element {property: value}
```

The most commonly used pseudo-elements are as follows:

Value	Description
:first-line	Use this element to add special styles to the first line of the text in a selector.
:first-letter	Use this element to add special style to the first letter of the text in a selector.
:before	Use this element to insert some content before an element.
:after	Use this element to insert some content after an element.

1. The :First-Line Pseudo-Element:

The following example demonstrates how to use the *:first-line* element to add special effects to the first line of elements in the document.

```
<html>
```

```
<head>
  <style type="text/css">
    p:first-line { text-decoration: underline; }
    p.noline:first-line { text-decoration: none; }
  </style>
</head>
<body>
  <p class="noline"> This line would not have any underline because this belongs to  nline class.</p>
  <p>The first line of this paragraph will be underlined as defined in the CSS rule above. Rest of the lines in this paragraph will remain normal. This example shows how to use :first-line pseduo element to give effect to the first line of any HTML element.</p>
</body>
</html>
```

2. The :First-Letter Pseudo-Element:

The following example demonstrates how to use the *:first-letter* element to add special effects to the first letter of elements in the document.

```
<html>
<head>
  <style type="text/css">
    p:first-letter { font-size: 5em; }
    p.normal:first-letter { font-size: 10px; }
  </style>
</head>
<body>
  <p class="normal"> First character of this paragraph will be normal and will have font size 10 px;</p>
  <p>The first character of this paragraph will be 5em big as defined in the CSS rule above. Rest of the characters in this paragraph will remain normal. This example shows how to use :first-letter pseduo element to give effect to the first characters  of any HTML element.</p>
</body>
</html>
```

3. The :Before Pseudo-Element:

The following example demonstrates how to use the *:before* element to add some content before any element.

```
<html>
<head>
  <style type="text/css">
    p:before
    {
      content: url(/images/bullet.gif)
    }
```

```
    </style>
  </head>
  <body>
    <p> This line will be preceded by a bullet.</p>
    <p> This line will be preceded by a bullet.</p>
    <p> This line will be preceded by a bullet.</p>
  </body>
</html>
```

4. The :After Pseudo-Element:

The following example demonstrates how to use the *:after* element to add some content after any element.

```
<html>
  <head>
    <style type="text/css">
      p:after
      {
        content: url(/images/bullet.gif)
      }
    </style>
  </head>
  <body>
    <p> This line will be succeeded by a bullet.</p>
    <p> This line will be succeeded by a bullet.</p>
    <p> This line will be succeeded by a bullet.</p>
  </body>
</html>
```

MEASUREMENT UNITS:

CSS supports a number of measurements including absolute units such as inches, centimeters, points, and so on, as well as relative measures such as percentages and em units. We need these values while specifying various measurements in our Style rules example: **border = "1px solid red"**.

Unit	Description	Example
%	Defines a measurement as a percentage relative to another value, typically an enclosing element.	p { font-size: 16pt; line-height: 125%; }
cm	Defines a measurement in centimeters.	div { margin-bottom: 2cm; }
em	A relative measurement for the height of a font in em spaces. Because an em unit is equivalent to the size of a given font, if you assign a font to 12pt, each "em" unit would be 12pt; thus, 2em would be 24pt.	p { letter-spacing: 7em; }

ex	This value defines a measurement relative to a font's x-height. The x-height is determined by the height of the font's lowercase letter x.	p {font-size: 24pt; line-height: 3ex;}
in	Defines a measurement in inches.	p { word-spacing: .15in; }
mm	Defines a measurement in millimeters.	p { word-spacing: 15mm; }
pc	Defines a measurement in picas. A pica is equivalent to 12 points; thus, there are 6 picas per inch.	p { font-size: 20pc; }
pt	Defines a measurement in points. A point is defined as 1/72nd of an inch.	body { font-size: 18pt; }
px	Defines a measurement in screen pixels.	p { padding: 25px; }
vh	1% of viewport height.	h2 { font-size: 3.0vh; }
vw	1% of viewport width	h1 { font-size: 5.9vw; }
vmin	1vw or 1vh, whichever is smaller	p { font-size: 2vmin;}

APPLYING <DIV> TAG TO STYLE SHEET:

Div (short for division) divides the content into individual sections. Each section can then have its own formatting, as specified by the CSS. Div is a block-level container, meaning that there is a line feed after the tag.

For example, if we have the following CSS declaration:

```
.large {
  color: #00FF00;
  font-family:arial;
  font-size: 4pt;
}
```

The HTML code

```
<div class="large">
  This is a DIV sample.
</div>
```

APPLYING <SPAN> TAG TO STYLE SHEET:

Span is similar to div in that they both divide the content into individual sections. The difference is that span goes into a finer level, so we can span to format a single character if needed. There is no line feed after the tag.

For example, if we have the following CSS declaration:

```
.largefont {
  color: #0066FF;
  font-family:arial;
  font-size: 6px;
```

}

The HTML code

Span is not at the block level.

LINKING STYLE SHEET:

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

1. Embedded CSS - The <style> Element:

We can put our CSS rules into an HTML document using the <style> element. This tag is placed inside <head>...</head> tags. Rules defined using this syntax will be applied to all the elements available in the document.

```
<!DOCTYPE html>
<html>
  <head>
    <style type = "text/css" media = "all">
      body {
        background-color: linen;
      }
      h1 {
        color: maroon;
        margin-left: 40px;
      }
    </style>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

Attributes associated with <style> elements are:

Attribute	Value	Description
type	text/css	Specifies the style sheet language as a content-type (MIME type). This is required attribute.
media	screen tty tv projection handheld print braille aural	Specifies the device the document will be displayed on. Default value is *all*. This is an optional attribute.

	all	

2. Inline CSS - The style Attribute:

We can use *style* attribute of any HTML element to define style rules. These rules will be applied to that element only. Here is the generic syntax:

<element style = "...style rules....">

Attributes

Attribute	Value	Description
style	style rules	The value of *style* attribute is a combination of style declarations separated by semicolon (;).

Example

```
<html>
  <head>
  </head>
  <body>
    <h1 style = "color:#36C;"> This is inline CSS </h1>
  </body>
</html>
```

3. External CSS - The <link> Element:

The <link> element can be used to include an external stylesheet file in our HTML document.

An external style sheet is a separate text file with **.css** extension. We define all the Style rules within this text file and then we can include this file in any HTML document using <link> element.

Here is the generic syntax of including external CSS file

```
<head>
  <link type = "text/css" href = "..." media = "..." />
</head>
```

Attributes

Attribute	Value	Description
type	text/css	Specifies the style sheet language as a content-type (MIME type). This attribute is required.
href	URL	Specifies the style sheet file having Style rules. This attribute is a required.
media	screen tty tv projection handheld	Specifies the device the document will be displayed on. Default value is *all*. This is optional attribute.

	print braille aural all	

Example

Consider a simple style sheet file with a name *mystyle.css* having the following rules:

```
h1, h2, h3 {
  color: #36C;
  font-weight: normal;
  letter-spacing: .4em;
  margin-bottom: 1em;
  text-transform: lowercase;
}
```

Now we can include this file mystyle.css in any HTML document as follows:

```
<head>
  <link type = "text/css" href = "mystyle.css" media = " all" />
</head>
```

4. Imported CSS - @import Rule:

@import is used to import an external stylesheet in a manner similar to the <link> element. Here is the generic syntax of @import rule.

```
<head>
  <@import "URL";
</head>
```

Here URL is the URL of the style sheet file having style rules. We can use another syntax as well:

```
<head>
  <@import url("URL");
</head>
```

Example

```
<head>
  @import "mystyle.css";
</head>
```

CSS RULES OVERRIDING:

We have discussed four ways to include style sheet rules in a HTML document. Here is the rule to override any Style Sheet Rule.

- Any inline style sheet takes highest priority. So, it will override any rule defined in <style>...</style> tags or rules defined in any external style sheet file.
- Any rule defined in <style>...</style> tags will override rules defined in any external style sheet file.
- Any rule defined in external style sheet file takes lowest priority, and rules defined in this file will be applied only when above two rules are not applicable.

HANDLING OLD BROWSERS:

There are still many old browsers who do not support CSS. So, we should take care while writing our Embedded CSS in an HTML document. The following snippet shows how we can use comment tags to hide CSS from older browsers:

```
<style type="text/css">
  <!--
    body, td {
      color: blue;
    }
  -->
</style>
```

CREATING CSS FILE:

Open a text editing program. If we have Microsoft Windows PC then we have to open the program named Notepad (hold down the Windows Key on keyboard and press R, then type notepad and press enter). If we are using a Macintosh computer, launch the application named "TextEdit" (which can be found in Apps folder).

Step 1: Let's Write Our First Bit of CSS:

Let's imagine we have a simple web page with a heading, and we want the heading to be orange and center aligned. Add the following code into your new blank text document:

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

Step 2: Saving the CSS File:

Create a new folder on desktop (or another location) and name it **CSS-Test**. Now, back in to text editing program save the document as ***"style.css".***

Step 3: Creating HTML Page:

Our new CSS file is worthless if we don't apply it to a web page. Let's create a quick HTML page for this lesson. Create a new blank file in Notepad (or TextEdit) and add the following code:

```
<!DOCTYPE html>
```

```
<html>
    <head>
        <meta charset="utf-8">
        <title>CSS-Test</title>
    </head>
    <body>
        <h1>CSS-Test</h1>
        <div id="box-one">
        <p>This is box one.</p>
        </div>
        <div id="box-two">
        <p>This is box two.</p>
        </div>
    </body>
</html>
```

Step 4: Linking CSS File to HTML:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>CSS-Test</title>
        <link rel = "stylesheet" type = "text/css" href="link.css" media = "all">
    </head>
    <body>
        <h1>CSS-Test</h1>
        <div id="box-one">
        <p>This is box one.</p>
        </div>
        <div id="box-two">
        <p>This is box two.</p>
        </div>
    </body>
</html>
```

Step 5: Linking Multiple CSS Files to HTML:

```
<!DOCTYPE html>
    <html>
            <head>
                    <meta charset="utf-8">
                    <title>CSS-Test</title>
                    <!--External CSS-->
                    <link rel = "stylesheet" type = "text/css" href="link.css" media = "all">
                    <!--Internal CSS-->
                    <style type = "text/css">
                            #BoxOne{
```

```
                    background-color: gray;
                }

                #BoxTwo{
                    background-color: yellow;
                    padding: 10px;
                }
            </style>
        </head>
        <body>
            <h1>CSS-Test</h1>
            <div id = "BoxOne">
            <p>This is box one.</p>
            </div>
            <div id = "BoxTwo">
            <p>This is box two.</p>
            </div>
        </body>
    </html>
```

CSS – FONTS:

1. Set the Font Family:

Possible value could be any font family name.

```
<html>
  <head>
  </head>
  <body>
    <p style="font-family:georgia,garamond,serif;">
    This text is rendered in either georgia, garamond, or the default serif font
    depending on which font  you have at your system.
    </p>
  </body>
</html>
```

2. Set the Font Style:

Possible values are *normal, italic and oblique*.

```
<html>
  <head>
  </head>
  <body>
    <p style="font-style:italic;">
    This text will be rendered in italic style
    </p>
```

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

3. Set the Font Variant:

Possible values are normal and small-caps.

```
<html>
  <head>
  </head>
  <body>
    <p style="font-variant:small-caps;">
    This text will be rendered as small caps
    </p>
  </body>
</html>
```

4. Set the Font Weight:

The font-weight property provides the functionality to specify how bold a font is. Possible values could be *normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900*.

```
<html>
  <head>
  </head>
  <body>
    <p style="font-weight:bold;">This font is bold.</p>
    <p style="font-weight:bolder;">This font is bolder.</p>
    <p style="font-weight:500;">This font is 500 weight.</p>
  </body>
</html>
```

5. Set the Font Size:

The font-size property is used to control the size of fonts. Possible values could be *xx-small, x-*small, small, medium, large, x-large, xx-large, smaller, larger, size in pixels or in %.

```
<html>
  <head>
  </head>
  <body>
    <p style="font-size:20px;">This font size is 20 pixels</p>
    <p style="font-size:small;">This font size is small</p>
    <p style="font-size:large;">This font size is large</p>
  </body>
</html>
```

6. Set the Font Size Adjust:

This property enables you to adjust the x-height to make fonts more legible. Possible value could be any number.

```
<html>
  <head>
  </head>
  <body>
    <p style="font-size-adjust:0.61;">
      This text is using a font-size-adjust value.
    </p>
  </body>
</html>
```

7. Set the Font Stretch:

This property relies on the user's computer to have an expanded or condensed version of the font being used. Possible values could be *normal, wider, narrower, ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded, ultra-expanded*.

```
<html>
  <head>
  </head>
  <body>
    <p style="font-stretch:ultra-expanded;">
      If this doesn't appear to work, it is likely that your computer doesn't have a
      condensed or expanded version of the font being used.
    </p>
  </body>
</html>
```

8. Shorthand Property:

We can use the *font* property to set all the font properties at once. For example

```
<html>
  <head>
  </head>
  <body>
    <p style="font:italic small-caps bold 15px georgia;">
    Applying all the properties on the text at once.
    </p>
  </body>
</html>
```

CSS – TEXT:

1. Set the Text Color:

Possible value could be any color name in any valid format.

```
<html>
  <head>
  </head>
  <body>
    <p style="color:red;">
    This text will be written in red.
    </p>
  </body>
</html>
```

2. Set the Text Direction:

Possible values are *ltr or rtl*.

```
<html>
  <head>
  </head>
  <body>
    <p style="direction:rtl;">
    This text will be renedered from right to left
    </p>
  </body>
</html>
```

3. Set the Space between Characters:

Possible values are *normal or a number specifying space.*.

```
<html>
  <head>
  </head>
  <body>
    <p style="letter-spacing:5px;">
    This text is having space between letters.
    </p>
  </body>
</html>
```

4. Set the Space between Words:

Possible values are *normal or a number specifying space*.

```
<html>
  <head>
  </head>
```

```
  <body>
    <p style="word-spacing:5px;">
    This text is having space between words.
    </p>
  </body>
</html>
```

5. Set the Text Indent:

Possible values are % or a number specifying indent space.

```
<html>
  <head>
  </head>
  <body>
    <p style="text-indent:1cm;">
    This text will have first line indented by 1cm and this line will remain at
    its actual position this is done by CSS text-indent property.
    </p>
  </body>
</html>
```

6. Set the Text Alignment:

Possible values are left, right, center, justify.

```
<html>
  <head>
  </head>
  <body>
    <p style="text-align:right;">
    This will be right aligned.
    </p>
    <p style="text-align:center;">
    This will be center aligned.
    </p>
    <p style="text-align:left;">
    This will be left aligned.
    </p>
  </body>
</html>
```

7. Decorating the Text:

Possible values are *none, underline, overline, line-through, blink*.

```
<html>
  <head>
  </head>
```

```
  <body>
    <p style="text-decoration:underline;">
    This will be underlined
    </p>

    <p style="text-decoration:line-through;">
    This will be striked through.
    </p>

    <p style="text-decoration:overline;">
    This will have a over line.
    </p>

    <p style="text-decoration:blink;">
    This text will have blinking effect
    </p>
  </body>
</html>
```

8. Set the Text Cases:

Possible values are *none, capitalize, uppercase, lowercase*.

```
<html>
  <head>
  </head>
  <body>
    <p style="text-transform:capitalize;">
    This will be capitalized
    </p>

    <p style="text-transform:uppercase;">
    This will be in uppercase
    </p>

    <p style="text-transform:lowercase;">
    This will be in lowercase
    </p>
    </body>
</html>
```

9. Set the White Space between Text:

Possible values are *normal, pre, nowrap*.

```
<html>
  <head>
  </head>
```

```
  <body>
    <p style="white-space:pre;">
    This text has a line break and the white-space pre setting tells the browser to honor
    it just like the HTML pre tag.</p>
  </body>
</html>
```

10. Set the Text Shadow:

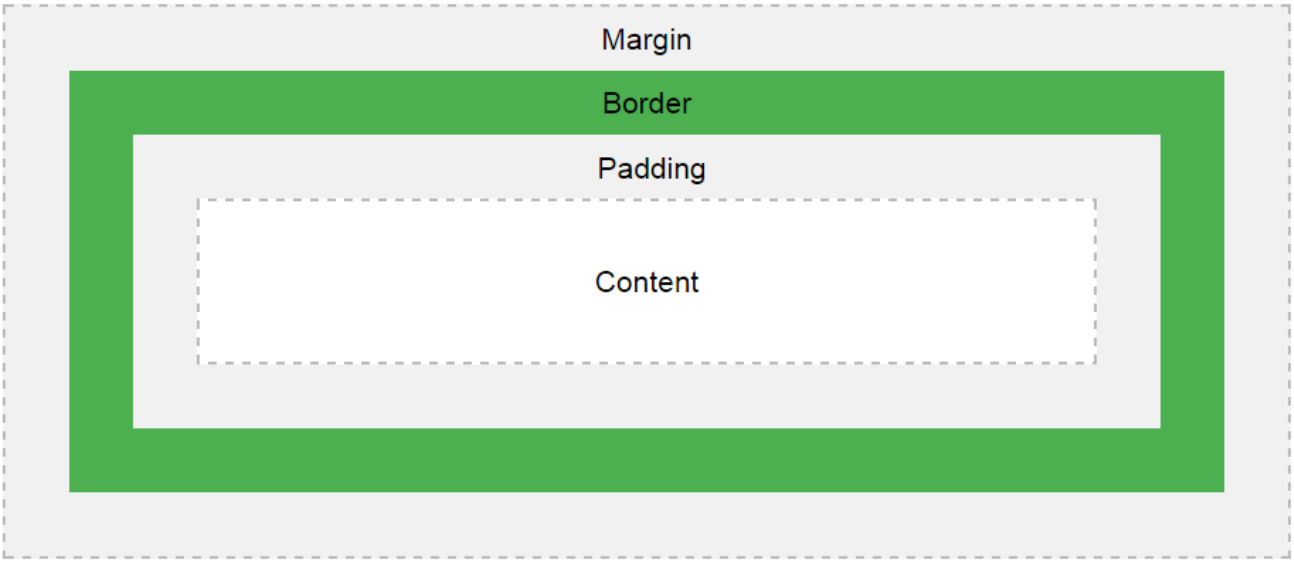
The following example demonstrates how to set the shadow around a text. This may not be supported by all the browsers.

```
<html>
  <head>
  </head>
  <body>
    <p style="text-shadow:4px 4px 8px blue;">
    If your browser supports the CSS text-shadow property, this text will have a blue shadow.
    </p>
  </body>
</html>
```

Introducing the Box Model:

In a document, each element is represented as a rectangular box. Determining the size, properties like its color, background, borders aspect and the position of these boxes is the goal of the rendering engine.

In CSS, each of these rectangular boxes is described using the standard *box model*. This model describes the space of the content taken by an element. Each box has four edges: the **margin edge**, **border edge**, **padding edge**, and **content edge**.



The **content area** is the area containing the real content of the element. It often has a background, a color or an image (in that order, an opaque image hiding the background color) and is located inside the *content edge*; its dimensions are the *content width*, or *content-box width*, and the *content height*, or *content-box height*.

If the CSS box-sizing property is set to default, the CSS properties width, min-width, max-width, height, min-height and max-height control the content size.

The **padding area** extends to the border surrounding the padding. When the content area has a background, color, or image set on it, this will extend into the padding, which is why you can think of the padding as extending the content. The padding is located inside the *padding edge*, and its dimensions are the *padding-box width* and the *padding-box height*.

The space between the padding and the content edge can be controlled using the padding-top, padding-right, padding-bottom, padding-left and the shorthand padding CSS properties.

The **border area** extends the padding area to the area containing the borders. It is the area inside the *border edge*, and its dimensions are the *border-box width* and the *border-box height*. This area depends on the size of the border that is defined by the border-width property or the shorthand border.

The **margin area** extends the border area with an empty area used to separate the element from its neighbors. It is the area inside the *margin edge*, and its dimensions are the *margin-box width* and the *margin-box height*.

The size of the margin area is controlled using the margin-top, margin-right, margin-bottom, margin-left and the shorthand margin CSS properties.

When margin collapsing happens, the margin area is not clearly defined since margins are shared between boxes.

LINKS:

- ☑ The **:link** signifies unvisited hyperlinks.
- ☑ The **:visited** signifies visited hyperlinks.
- ☑ The **:hover** signifies an element that currently has the user's mouse pointer hovering over it.
- ☑ The **:active** signifies an element on which the user is currently clicking.

Usually, all these properties are kept in the header part of the HTML document.

Remember a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective. Also, a:active MUST come after a:hover in the CSS definition as follows:

```
<style type="text/css">
  a:link {color: #000000}
  a:visited {color: #006600}
  a:hover {color: #FFCC00}
  a:active {color: #FF00CC}
</style>
```

BACKGROUNDS:

We can set the following background properties of an element:

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

```
<style type = “text/css”>
    body {
        background-color:yellow;
    }
</style>
```

- ☑ The **background-image** property is used to set the background image of an element.

```
<style>
    body  {
      background-image: url("css.jpg");
      background-color: #cccccc;
    }
</style>
```

- ☑ The **background-repeat** property is used to control the repetition of an image in the background.

```
<style>
    body {
      background-image: url("css.jpg");
      background-repeat: repeat;
    }
 </style>
```

- ☑ The **background-position** property is used to control the position of an image in the background.

```
<style>
    body {
      background-image: url("css.jpg");
      background-position:100px 200px;
    }
 </style>
```

- ☑ The **background-attachment** property is used to control the scrolling of an image in the background.

```
<style>
    body  {
      background-image: url('/css/images/css.jpg');
      background-repeat: no-repeat;
      background-attachment: fixed;
    }
</style>
```

☑ The **background** property is used as a shorthand to specify a number of other background properties.

```
<p style="background:url(/images/pattern1.gif) repeat fixed;">
  This parapgraph has fixed repeated background image.
</p>
```

LIST:

Lists are very helpful in conveying a set of either numbered or bullet points. We have the following five CSS properties, which can be used to control lists:

1. The list-style-type Property:

The *list-style-type* property allows us to control the shape or style of bullet point (also known as a marker) in the case of unordered lists and the style of numbering characters in ordered lists.

Here are the values which can be used for an unordered list:

Value	Description
none	NA
disc (default)	A filled-in circle
circle	An empty circle
square	A filled-in square

Here are the values, which can be used for an ordered list:

Value	Description	Example
decimal	Number	1,2,3,4,5
decimal-leading-zero	0 before the number	01, 02, 03, 04, 05
lower-alpha	Lowercase alphanumeric characters	a, b, c, d, e
upper-alpha	Uppercase alphanumeric characters	A, B, C, D, E
lower-roman	Lowercase Roman numerals	i, ii, iii, iv, v
upper-roman	Uppercase Roman numerals	I, II, III, IV, V
lower-greek	The marker is lower-greek	alpha, beta, gamma
lower-latin	The marker is lower-latin	a, b, c, d, e
upper-latin	The marker is upper-latin	A, B, C, D, E

Example:

```
<html>
  <head>
  </head>

  <body>
    <ul style="list-style-type:circle;">
      <li>Maths</li>
      <li>Social Science</li>
```

```
    <li>Physics</li>
  </ul>

  <ul style="list-style-type:square;">
    <li>Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
  </ul>

  <ol style="list-style-type:decimal;">
    <li>Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
  </ol>

  <ol style="list-style-type:lower-alpha;">
    <li>Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
  </ol>

  <ol style="list-style-type:lower-roman;">
    <li>Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
  </ol>
 </body>
</html>
```

2. The list-style-position Property:

The *list-style-position* property indicates whether the marker should appear inside or outside of the box containing the bullet points. It can have one the two values:

Value	Description
none	NA
inside	If the text goes onto a second line, the text will wrap underneath the marker. It will also appear indented to where the text would have started if the list had a value of outside.
outside	If the text goes onto a second line, the text will be aligned with the start of the first line (to the right of the bullet).

Example:

```
<html>
  <head>
  </head>
```

```
<body>
  <ul style="list-style-type:circle; list-style-position:outside;">
    <li>Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
  </ul>

  <ul style="list-style-type:square;list-style-position:inside;">
    <li>Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
  </ul>

  <ol style="list-style-type:decimal;list-style-position:outside;">
    <li>Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
  </ol>

  <ol style="list-style-type:lower-alpha;list-style-position:inside;">
    <li>Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
  </ol>
 </body>
</html>
```

3. The list-style-image Property:

The *list-style-image* allows us to specify an image so that we can use our own bullet style. The syntax is similar to the background-image property with the letters url starting the value of the property followed by the URL in brackets. If it does not find the given image then default bullets are used.

Example:

```
<html>
  <head>
  </head>

  <body>
    <ul>
      <li style="list-style-image: url(/images/bullet.gif);">Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ul>
```

```
    <ol>
      <li style="list-style-image: url(/images/bullet.gif);">Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ol>
  </body>
</html>
```

4. The list-style Property:

The *list-style* allows us to specify all the list properties into a single expression. These properties can appear in any order.

Example:

```
<html>
  <head>
  </head>

  <body>
    <ul style="list-style: inside square;">
      <li>Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ul>

    <ol style="list-style: outside upper-alpha;">
      <li>Maths</li>
      <li>Social Science</li>
      <li>Physics</li>
    </ol>
  </body>
</html>
```

5. The marker-offset Property:

The *marker-offset* property allows us to specify the distance between the marker and the text relating to that marker. Its value should be a length as shown in the following example:

Unfortunately, this property is not supported in IE 6 or Netscape 7.

Example:

```
<html>
  <head>
  </head>

  <body>
    <ul style="list-style: inside square; marker-offset:2em;">
      <li>Maths</li>
```

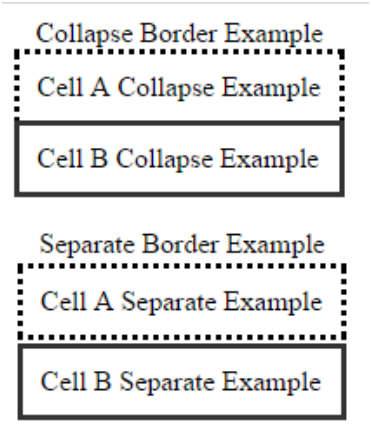
```
    <li>Social Science</li>
    <li>Physics</li>
  </ul>

  <ol style="list-style: outside upper-alpha; marker-offset:2cm;">
    <li>Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
  </ol>
 </body>
</html>
```

TABLES:

1. The border-collapse Property:

This property can have two values *collapse* and *separate*. The following example uses both the values:

```
<html>
 <head>
  <style type="text/css">
   table.one {border-collapse:collapse;}
   table.two {border-collapse:separate;}
   td.a {
    border-style:dotted;
    border-width:3px;
    border-color:#000000;
    padding: 10px;
   }
   td.b {
    border-style:solid;
    border-width:3px;
    border-color:#333333;
    padding:10px;
   }
  </style>

 </head>
 <body>

  <table class="one">
   <caption>Collapse Border Example</caption>
   <tr><td class="a"> Cell A Collapse Example</td></tr>
   <tr><td class="b"> Cell B Collapse Example</td></tr>
  </table>
  <br />
```



```
    <table class="two">
      <caption>Separate Border Example</caption>
      <tr><td class="a"> Cell A Separate Example</td></tr>
      <tr><td class="b"> Cell B Separate Example</td></tr>
    </table>
  </body>
</html>
```

2. The border-spacing Property:

The border-spacing property specifies the distance that separates adjacent cells'. borders. It can take either one or two values; these should be units of length.

If we provide one value, it will applies to both vertical and horizontal borders. Or we can specify two values, in which case, the first refers to the horizontal spacing and the second to the vertical spacing.

Example:

```
<html>
  <head>

    <style type="text/css">
      table.one {
        border-collapse:separate;
        width:400px;
        border-spacing:10px;
      }
      table.two {
        border-collapse:separate;
        width:400px;
        border-spacing:10px 50px;
      }
    </style>

  </head>
  <body>

    <table class="one" border="1">
      <caption>Separate Border Example with border-spacing</caption>
      <tr><td> Cell A Collapse Example</td></tr>
      <tr><td> Cell B Collapse Example</td></tr>
    </table>
    <br />

    <table class="two" border="1">
      <caption>Separate Border Example with border-spacing</caption>
```

Separate Border Example with border-spacing

Cell A Collapse Example
Cell B Collapse Example

Separate Border Example with border-spacing

Cell A Separate Example
Cell B Separate Example

```
    <tr><td> Cell A Separate Example</td></tr>
    <tr><td> Cell B Separate Example</td></tr>
  </table>

 </body>
</html>
```

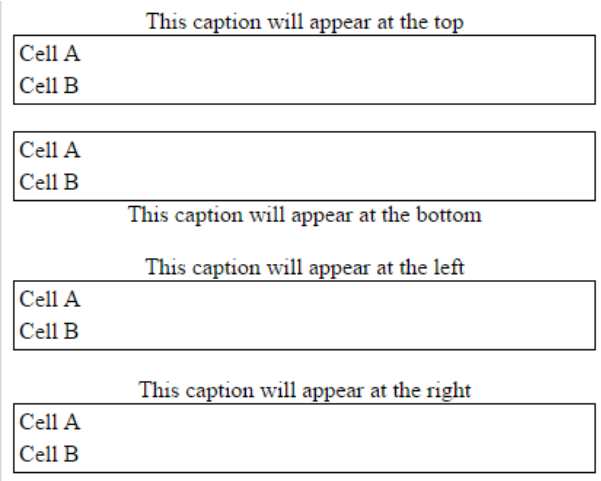
3. The caption-side Property:

The caption-side property allows us to specify where the content of a <caption> element should be placed in relationship to the table. The table that follows lists the possible values.

This property can have one of the four values *top, bottom, left* or *right*. The following example uses each value.

Example:

```
<html>
 <head>

  <style type="text/css">
   caption.top {caption-side:top}
   caption.bottom {caption-side:bottom}
   caption.left {caption-side:left}
   caption.right {caption-side:right}
  </style>

 </head>
 <body>

  <table style="width:400px; border:1px solid black;">
   <caption class="top">
   This caption will appear at the top
   </caption>
   <tr><td > Cell A</td></tr>
   <tr><td > Cell B</td></tr>
  </table>
  <br />

  <table style="width:400px; border:1px solid black;">
   <caption class="bottom">
   This caption will appear at the bottom
   </caption>
   <tr><td > Cell A</td></tr>
   <tr><td > Cell B</td></tr>
  </table>
  <br />
```



```
    <table style="width:400px; border:1px solid black;">
      <caption class="left">
      This caption will appear at the left
      </caption>
      <tr><td > Cell A</td></tr>
      <tr><td > Cell B</td></tr>
    </table>
    <br />

    <table style="width:400px; border:1px solid black;">
      <caption class="right">
      This caption will appear at the right
      </caption>
      <tr><td > Cell A</td></tr>
      <tr><td > Cell B</td></tr>
    </table>

  </body>
</html>
```

4. The empty-cells Property:

The empty-cells property indicates whether a cell without any content should have a border displayed. This property can have one of the three values - *show, hide* or *inherit*. Here is the empty-cells property used to hide borders of empty cells in the <table> element.

```
<html>
  <head>

    <style type="text/css">
      table.empty{
        width:350px;
        border-collapse:separate;
        empty-cells:hide;
      }
      td.empty{
        padding:5px;
        border-style:solid;
        border-width:1px;
        border-color:#999999;
      }
    </style>

  </head>
  <body>

    <table class="empty">
```

	Title one	Title two
Row Title	value	value
Row Title	value	

```
    <tr>
      <th></th>
      <th>Title one</th>
      <th>Title two</th>
    </tr>

    <tr>
      <th>Row Title</th>
      <td class="empty">value</td>
      <td class="empty">value</td>
    </tr>

    <tr>
      <th>Row Title</th>
      <td class="empty">value</td>
      <td class="empty"></td>
    </tr>
    </table>

  </body>
</html>
```

5. The table-layout Property:

The table-layout property is supposed to help us to control how a browser should render or lay out a table. This property can have one of the three values: *fixed, auto* or *inherit*. The following example shows the difference between these properties.

```
<html>
  <head>

    <style type="text/css">
      table.auto {
        table-layout: auto
      }
      table.fixed{
        table-layout: fixed
      }
    </style>

  </head>
  <body>

    <table class="auto" border="1" width="100%">
    <tr>
      <td width="20%">10000000000000000000000000000</td>
      <td width="40%">10000000</td>
```

10000000000000000	10000000	100

100000000000	10000000	100

```
      <td width="40%">100</td>
    </tr>
    </table>
    <br />

    <table class="fixed" border="1" width="100%">
    <tr>
      <td width="20%">10000000000000000000000000000</td>
      <td width="40%">10000000</td>
      <td width="40%">100</td>
    </tr>
    </table>

  </body>
</html>
```

OUTLINES:

Outlines are very similar to borders, but there are few major differences as well:

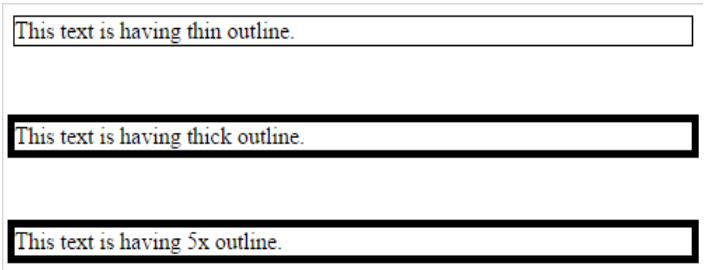
- ☑ An outline does not take up space.
- ☑ Outlines do not have to be rectangular.
- ☑ Outline is always the same on all sides; we cannot specify different values for different sides of an element.

We can set the following outline properties using CSS:

1. The outline-width Property:

The *outline-width* property specifies the width of the outline to be added to the box. Its value should be a length or one of the values *thin, medium, or thick,* just like the border-width attribute.

A width of zero pixels means no outline.



Example:

```
<html>
  <head>
  </head>

  <body>
    <p style="outline-width:thin; outline-style:solid;">
    This text is having thin outline.
    </p>
    <br />

    <p style="outline-width:thick; outline-style:solid;">
    This text is having thick outline.
    </p>
```

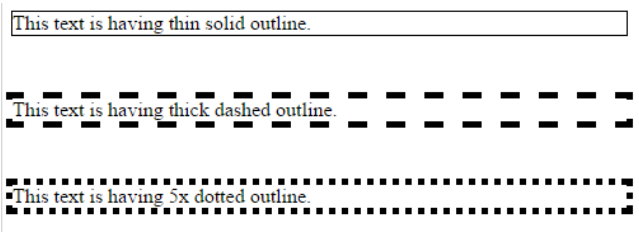
```
    <br />

    <p style="outline-width:5px; outline-style:solid;">
    This text is having 5x outline.
    </p>
  </body>
</html>
```

2. The outline-style Property:

The *outline-style* property specifies the style for the line (solid, dotted, or dashed) that goes around an element. It can take one of the following values:

- ☑ **none:** No border. (Equivalent of outline-width:0;)
- ☑ **solid:** Outline is a single solid line.
- ☑ **dotted:** Outline is a series of dots.
- ☑ **dashed:** Outline is a series of short lines.
- ☑ **double:** Outline is two solid lines.
- ☑ **groove:** Outline looks as though it is carved into the page.
- ☑ **ridge:** Outline looks the opposite of groove.
- ☑ **inset:** Outline makes the box look like it is embedded in the page.
- ☑ **outset:** Outline makes the box look like it is coming out of the canvas.
- ☑ **hidden:** Same as none.

Example:

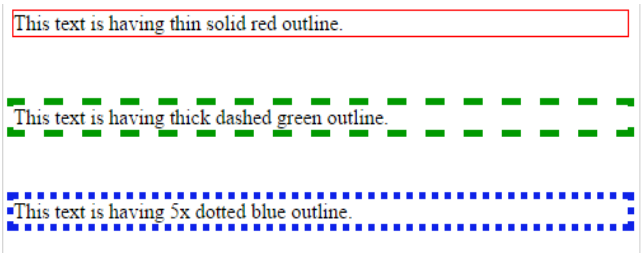



```
<html>
  <head>
  </head>

  <body>
    <p style="outline-width:thin; outline-style:solid;">
    This text is having thin solid  outline.
    </p>
    <br />

    <p style="outline-width:thick; outline-style:dashed;">
    This text is having thick dashed outline.
    </p>
    <br />

    <p style="outline-width:5px;outline-style:dotted;">
    This text is having 5x dotted outline.
    </p>
  </body>

</html>
```

3. The outline-color Property:

The *outline-color* property allows you to specify the color of the outline. Its value should either be a color name, a hex color, or an RGB value, as with the color and border-color properties.

Example:

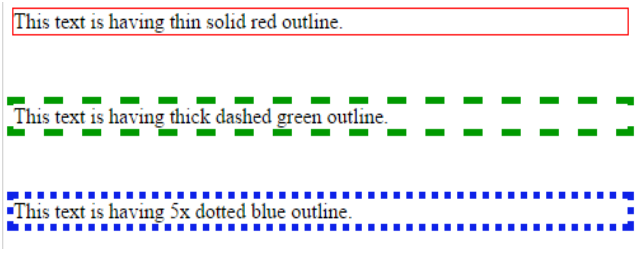



```
<html>
  <head>
  </head>

  <body>
    <p style="outline-width:thin; outline-style:solid;outline-color:red">
    This text is having thin solid red  outline.
    </p>
    <br />

    <p style="outline-width:thick; outline-style:dashed;outline-color:#009900">
    This text is having thick dashed green outline.
    </p>
    <br />

    <p style="outline-width:5px;outline-style:dotted;outline-color:rgb(13,33,232)">
    This text is having 5x dotted blue outline.
    </p>
  </body>

</html>
```

4. The outline Property:

The *outline* property is a shorthand property that allows us to specify values for any of the three properties discussed previously in any order but in a single statement.

Example:




```
<html>
  <head>
  </head>

  <body>
    <p style="outline:thin solid red;">
    This text is having thin solid red outline.
    </p>
    <br />

    <p style="outline:thick dashed #009900;">
    This text is having thick dashed green outline.
    </p>
```

```
    <br />

    <p style="outline:5px dotted rgb(13,33,232);">
    This text is having 5x dotted blue outline.
    </p>
  </body>

</html>
```

POSITIONING:

CSS helps us to position our HTML element. We can put any HTML element at whatever location we like. We can specify whether we want the element positioned relative to its natural position in the page or absolute based on its parent element.

Now, we will see all the CSS positioning related properties with examples:

1. Relative Positioning:

Relative positioning changes the position of the HTML element relative to where it normally appears. So "left:20" adds 20 pixels to the element's LEFT position.

We can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

- ☑ Move Left - Use a negative value for *left*.
- ☑ Move Right - Use a positive value for *left*.
- ☑ Move Up - Use a negative value for *top*.
- ☑ Move Down - Use a positive value for *top*.

Example:

```
<html>
  <head>
  </head>
  <body>
    <div style="position:relative;left:80px;top:2px;background-color:yellow;">
    This div has relative positioning.
    </div>
  </body>
</html>
```

2. Absolute Positioning:

An element with **position: absolute** is positioned at the specified coordinates relative to our screen top-left corner. We can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

- ☑ Move Left - Use a negative value for left.
- ☑ Move Right - Use a positive value for left.

- ☑ Move Up - Use a negative value for top.
- ☑ Move Down - Use a positive value for top.

Example:

```
<html>
  <head>
  </head>
  <body>
    <div style="position:absolute; left:80px; top:20px; background-color:yellow;">
    This div has absolute positioning.
    </div>
  </body>
</html>
```

3. Fixed Positioning:

Fixed positioning allows us to fix the position of an element to a particular spot on the page, regardless of scrolling. Specified coordinates will be relative to the browser window. We can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

- ☑ Move Left - Use a negative value for left.
- ☑ Move Right - Use a positive value for left.
- ☑ Move Up - Use a negative value for top.
- ☑ Move Down - Use a positive value for top.

Example:

```
<html>
  <head>
  </head>
  <body>
    <div style="position:fixed; left:80px; top:20px; background-color:yellow;">
    This div has fixed positioning.
    </div>
  </body>
</html>
```

Layout:

CSS provides plenty of controls for positioning elements in a document. Since CSS is *the wave of the future,* why not use CSS instead of tables for page layout purposes?

The following list collects a few pros and cons of both the use of table and CSS for layout:

- ☑ Most browsers support tables, while CSS support is being slowly adopted.

- ☑ Tables are more forgiving when the browser window size changes morphing their content and wrapping to accommodate the changes accordingly. CSS positioning tends to be exact and fairly inflexible.
- ☑ Tables are much easier to learn and manipulate than CSS rules.

But each of these arguments can be reversed:

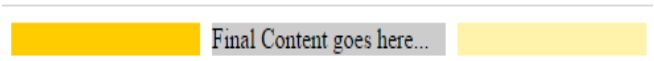
- ☑ CSS is pivotal to the future of Web documents and will be supported by most browsers.
- ☑ CSS is more exact than tables, allowing our document to be viewed as we intended, regardless of the browser window.
- ☑ Keeping track of nested tables can be a real pain. CSS rules tend to be well organized, easily read, and easily changed.

CSS also provides *table-layout* property to make our tables load much faster. Following is an example:

```
<table style="table-layout:fixed; width:600px;">
  <tr height="30">
    <td width="150">CSS table layout cell 1</td>
    <td width="200">CSS table layout cell 2</td>
    <td width="250">CSS table layout cell 3</td>
  </tr>
</table>
```

We notice the benefits more on large tables. With traditional HTML, the browser had to calculate every cell before finally rendering the table. When we set the table-layout algorithm to *fixed*, however, it only needs to look at the first row before rendering the whole table. It means our table will need to have fixed column widths and row heights.

Sample Column Layout:




```
<!DOCTYPE html>
    <html>
    <head>
  <style style="text/css">
  body {
    margin:9px 9px 0 9px;
    padding:0;
    background:#FFF;
  }

  #level0 {background:#FC0;}

  #level1 {
    margin-left:143px;
    padding-left:9px;
    background:#FFF;
  }

  #level2 {background:#FFF3AC;}
```

```
  #level3 {
    margin-right:143px;
    padding-right:9px;
    background:#FFF;
  }

  #main {background:#CCC;}
</style>
</head>
<body>
  <div id="level0">
    <div id="level1">
      <div id="level2">
        <div id="level3">
          <div id="main">
            Final Content goes here...
          </div>
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```

BROWSER SAFE COLORS:

Here is the list of 216 colors which are supposed to be most safe and computer independent colors. These colors vary from hexa code 000000 to FFFFFF. These colors are safe to use because they ensure that all computers would display the colors correctly when running a 256 color palette:

000000	000033	000066	0000z99	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF

333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF
660000	660033	660066	660099	6600CC	6600FF
663300	663333	663366	663399	6633CC	6633FF
666600	666633	666666	666699	6666CC	6666FF
669900	669933	669966	669999	6699CC	6699FF
66CC00	66CC33	66CC66	66CC99	66CCCC	66CCFF
66FF00	66FF33	66FF66	66FF99	66FFCC	66FFFF
990000	990033	990066	990099	9900CC	9900FF
993300	993333	993366	993399	9933CC	9933FF
996600	996633	996666	996699	9966CC	9966FF
999900	999933	999966	999999	9999CC	9999FF
99CC00	99CC33	99CC66	99CC99	99CCCC	99CCFF
99FF00	99FF33	99FF66	99FF99	99FFCC	99FFFF
CC0000	CC0033	CC0066	CC0099	CC00CC	CC00FF
CC3300	CC3333	CC3366	CC3399	CC33CC	CC33FF
CC6600	CC6633	CC6666	CC6699	CC66CC	CC66FF
CC9900	CC9933	CC9966	CC9999	CC99CC	CC99FF
CCCC00	CCCC33	CCCC66	CCCC99	CCCCCC	CCCCFF
CCFF00	CCFF33	CCFF66	CCFF99	CCFFCC	CCFFFF
FF0000	FF0033	FF0066	FF0099	FF00CC	FF00FF
FF3300	FF3333	FF3366	FF3399	FF33CC	FF33FF

FF6600	FF6633	FF6666	FF6699	FF66CC	FF66FF
FF9900	FF9933	FF9966	FF9999	FF99CC	FF99FF
FFCC00	FFCC33	FFCC66	FFCC99	FFCCCC	FFCCFF
FFFF00	FFFF33	FFFF66	FFFF99	FFFFCC	FFFFFF