CSS

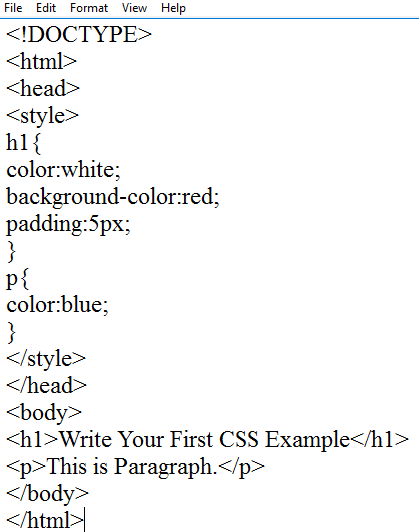
## Introduction to CSS:

**CSS** or CSS provides basic and advanced concepts of CSS technology. Our CSS is developed for beginners and professionals. The major points of CSS are given below:

* CSS stands for Cascading Style Sheet.
* CSS is used to design HTML tags.
* CSS is a widely used language on the web.

HTML, CSS and JavaScript are used for web designing. It helps the web designers to apply style on HTML tags.

**Example:**



Output



## What is CSS?

CSS 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.

## What does CSS do?

* You can add new looks to your old HTML documents.
* You can completely change the look of your website with only a few changes in CSS code.

## Why use CSS?

These are the three major benefits of CSS:

#### Solves a big problem

Before CSS, tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. For example: If you are developing a large website where fonts and color information are added on every single page, it will be become a long and expensive process. CSS was created to solve this problem. It was a W3C recommendation.

#### Saves a lot of time

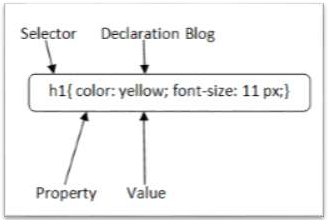
CSS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file.

#### Provide more attributes

CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

#### Syntax:

A CSS rule set contains a selector and a declaration block.



**Selector:** Selector indicates the HTML element you want to style. It could be any tag like <h1>,

<title> etc.

**Declaration Block:** The declaration block can contain one or more declarations separated by a semicolon. For the above example, there are two declarations:

* 1. color: yellow;
  2. font-size: 11 px;

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

**Property:** A Property is a type of attribute of HTML element. It could be color, border etc.

**Value:** Values are assigned to CSS properties. In the above example, value "yellow" is assigned to color property.

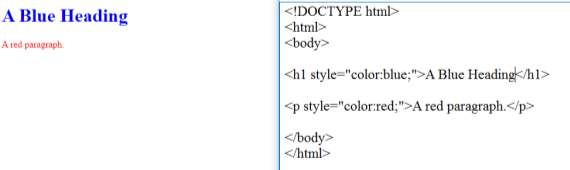
Selector {Property1: value1; Property2: value2; ;}

## CSS How to Use

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

* **Inline styles** — using the style attribute in the HTML start tag.
* **Embedded styles** — using the <style> element in the head section of a document.
* **External style sheets** — using the <link> element, pointing to an external CSS file.

We will cover all these three methods for inserting CSS one by one.



## How to Link CSS to HTML

To link your CSS to your HTML, you have to use the link tag with some relevant attributes. The link tag is a self-closing tag you should put at the head section of your HTML.

To link CSS to HTML with it, this is how you do it:

#### Syntax

<link rel="stylesheet" type="text/css" href="styles.css" />

**Attributes of the Link Tag The rel Attribute**

rel is the relationship between the external file and the current file. For CSS, you use stylesheet. For example, rel="stylesheet".

#### The type Attribute

type is the type of the document you are linking to the HTML. For CSS, it is text/css. For example type="text/css".

#### The href Attribute

href stands for “hypertext reference”. You use it to specify the location of the CSS file and the file name. It is a clickable link, so you can also hold CTRL and click it to view the CSS file.

For example, href="styles.css" if the CSS file is located in the same folder as the HTML file. Or href="folder/styles.css" if the CSS file is located on another folder.

## CSS Color

The color property in CSS is used to set the color of HTML elements. Typically, this property is used to set the background color or the font color of an element.

In CSS, we use color values for specifying the color. We can also use this property for the border- color and other decorative effects.

We can define the color of an element by using the following ways:

* RGB format.
* RGBA format.
* Hexadecimal notation.
* HSL.
* HSLA.
* Built-in color.

Let's understand the syntax and description of the above ways in detail.

#### RGB Format

RGB format is the short form of '**RED GREEN** and **BLUE**' that is used for defining the color of an HTMLelement simply by specifying the values of R, G, B that are in the range of 0 to 255.The color values in this format are specified by using the **rgb()** property. This property allows three values that can either be in percentage or integer (range from 0 to 255).

This property is not supported in all browsers; that's why it is not recommended to use it.

#### Syntax

color: rgb(R, G, B);

**RGBA Format**

It is almost similar to RGB format except that **RGBA** contains **A (Alpha)** that specifies the element's transparency. The value of alpha is in the range **0.0 to 1.0**, in which **0.0** is for fully transparent, and **1.0** is for not transparent.

#### Syntax

color:rgba(R, G, B, A);

**Hexadecimal notation**

Hexadecimal can be defined as a six-digit color representation. This notation starts with the **# symbol** followed by six characters ranges from **0 to F**. In hexadecimal notation, the first two digits represent the **red (RR)** color value, the next two digits represent the **green (GG)** color value, and the last two digits represent the **blue (BB)** color value.

The black color notation in hexadecimal is #000000, and the white color notation in hexadecimal is #FFFFFF. Some of the codes in hexadecimal notation are #FF0000, #00FF00, #0000FF, #FFFF00, and many more.

#### Syntax

color:#(0-F)(0-F)(0-F)(0-F)(0-F)(0-F);

**Short Hex codes**

It is a short form of hexadecimal notation in which every digit is recreated to arrive at an equivalent hexadecimal value.

For example, #7B6 becomes #77BB66 in hexadecimal.

The black color notation in short hex is #000, and the white color notation in short hex is #FFF. Some of the codes in short hex are #F00, #0F0, #0FF, #FF0, and many more.

#### HSL

It is a short form of **Hue, Saturation,** and **Lightness**. Let's understand them individually.

**Hue:** It can be defined as the degree on the color wheel from 0 to 360. 0 represents red, 120 represents green, and 240 represents blue.

**Saturation:** It takes value in percentage in which 100% represents fully saturated, i.e., no shades of gray, 50% represent 50% gray, but the color is still visible, and 0% represents fully unsaturated, i.e., completely gray, and the color is invisible.

**Lightness:** The lightness of the color can be defined as the light that we want to provide the color in which 0% represents black (there is no light), 50% represents neither dark nor light, and 100% represents white (full lightness).

Let's see the syntax of HSL in color property.

#### Syntax

color:hsl(H, S, L);

**HSLA**

It is entirely similar to HSL property, except that it contains **A (alpha)** that specifies the element's transparency. The value of alpha is in the range **0.0 to 1.0**, in which **0.0** indicates fully transparent, and **1.0** indicates not transparent.

#### Syntax

color:hsla(H, S, L, A);

Built-in Color

As its name implies, built-in color means the collection of previously defined colors that are used by using a name such as red, blue, green, etc.

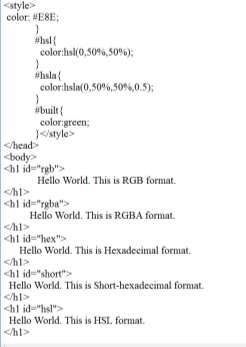
**Syntax**

color: color-name;

Let's see the list of built-in colors along with their decimal and hexadecimal values.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no.** | **Color name** | **Hexadecimal Value** | **Decimal Value or rgb() value** |
| **1.** | Red | #FF0000 | rgb(255,0,0) |
| **2.** | Orange | #FFA500 | rgb(255,165,0) |
| **3.** | Yellow | #FFFF00 | rgb(255,255,0) |
| **4.** | Pink | #FFC0CB | rgb(255,192,203) |
| **5.** | Green | #008000 | rgb(0,128,0) |
| **6.** | Violet | #EE82EE | rgb(238,130,238) |
| **7.** | Blue | #0000FF | rgb(0,0,255) |
| **8.** | Aqua | #00FFFF | rgb(0,255,255) |
| **9.** | Brown | #A52A2A | rgb(165,42,42) |
| **10.** | White | #FFFFFF | rgb(255,255,255) |
| **11.** | Gray | #808080 | rgb(128,128,128) |
| **12.** | Black | #000000 | rgb(0,0,0) |

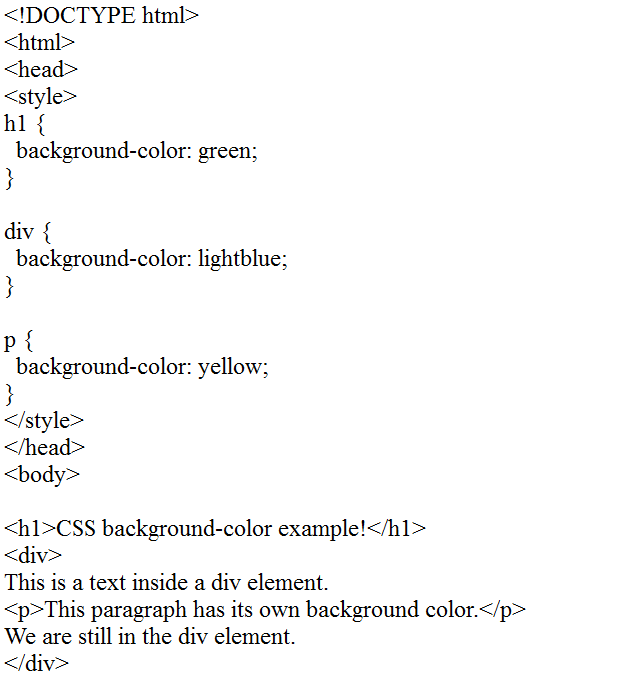
The illustration of CSS



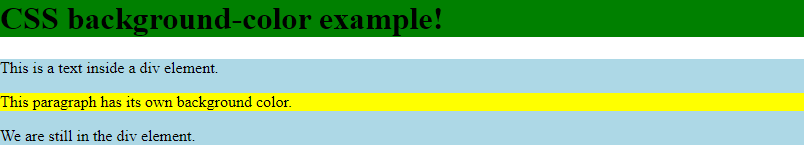
**Output:**



## CSS Backgrounds



**Output:**



Defines the color of the element's background.

* + Default background-color: transparent;
  + background-color: red;
  + background-color: #05ffb0;
  + background-color: rgb(50, 115, 220);
  + background-color: rgba(50, 115, 220, 0.3);
  + background-color: hsl(14, 100%, 53%);
  + background-color: hsla(14, 100%, 53%, 0.6);

**Default background-color: transparent;**

By default, the background color is transparent, basically meaning that there is no background color.

Hello World

**background-color: red;**

You can use one of the **140+ color names**. Hello World

**background-color: #05ffb0;**

You can use **hexadecimal** color codes. Hello World

**background-color: rgb(50, 115, 220);**

You can use **rgb()** color codes:

* + the first value is for red
  + the second value is for green
  + the third value is for blue

Each of them can have a value between **0** and **255**. Hello World

**background-color: rgba(50, 115, 220, 0.3);**

You can use **rgba()** color codes:

* + the first 3 values are for rgb
  + the 4th value is for the alpha channel and defines the opacity of the color The alpha value can go from zero **0** (transparent) to one **1** (opaque).

Hello World

**background-color: hsl(14, 100%, 53%);**

You can use **hsl()** color codes:

* + the first value is for hue and can go from **0** to **359**
  + the second value is for saturation and go from **0%** to **100%**
  + the third value is for luminosity and go from **0%** to **100%**

Hello World

**background-color: hsla(14, 100%, 53%, 0.6);**

You can use **hsl()a** color codes:

* + the first 3 values are for hsl
  + the 4th value is for the alpha channel and defines the opacity of the color The alpha value can go from zero **0** (transparent) to one **1** (opaque).

|  |  |
| --- | --- |
| **Value** | **Description** |
| none | It doesn't define any border. |
| dotted | It is used to define a dotted border. |
| dashed | It is used to define a dashed border. |
| solid | It is used to define a solid border. |
| double | It defines two borders wIth the same border-width value. |
| groove | It defines a 3d grooved border. effect is generated according to border-color value. |
| ridge | It defines a 3d ridged border. effect is generated according to border-color value. |
| inset | It defines a 3d inset border. effect is generated according to border-color value. |
| outset | It defines a 3d outset border. effect is generated according to border-color value. |

## CSS Borders

The CSS border is a shorthand property used to set the border on an element.

The CSS border properties are used to specify the style, color and size of the border of an element.

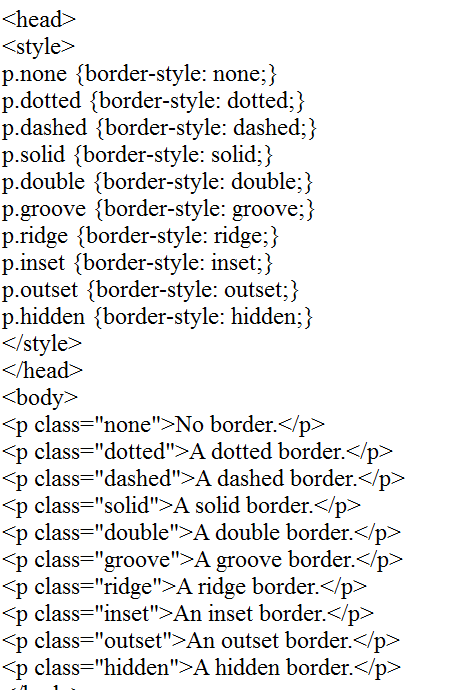
The CSS border properties are given below

* border-style
* border-color
* border-width
* border-radius

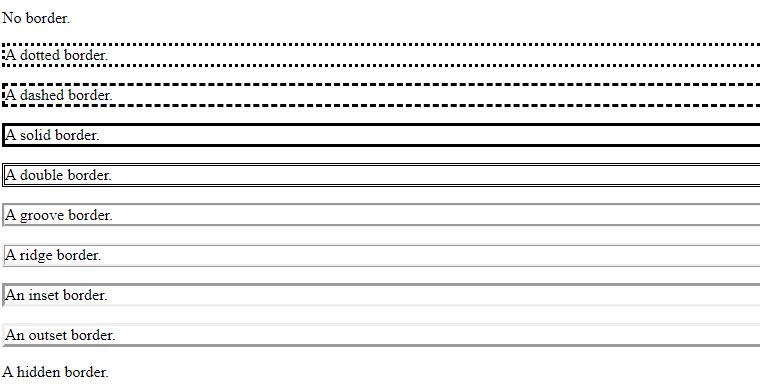
#### CSS border-style

The Border style property is used to specify the border type which you want to display on the web page.

There are some border style values which are used with border-style property to define a border.

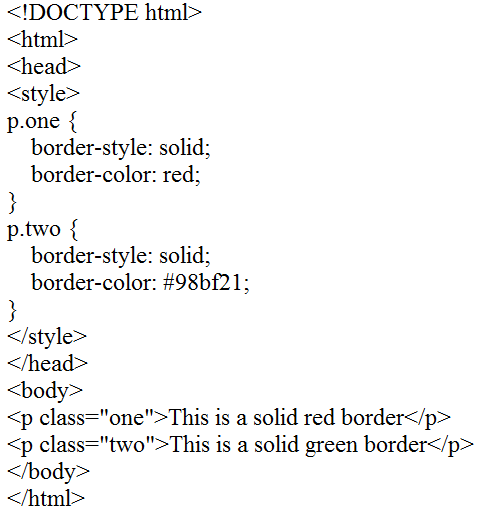


**Output:**



1. **CSS border-width**

The border-width property is used to set the border's width. It is set in pixels. You can also use the one of the three pre-defined values, thin, medium or thick to set the width of the border.



### Output:



* + Name: It specifies the color name. For example: "red".
  + RGB: It specifies the RGB value of the color. For example: "rgb(255,0,0)".
  + Hex: It specifies the hex value of the color. For example: "#ff0000".

## CSS Padding

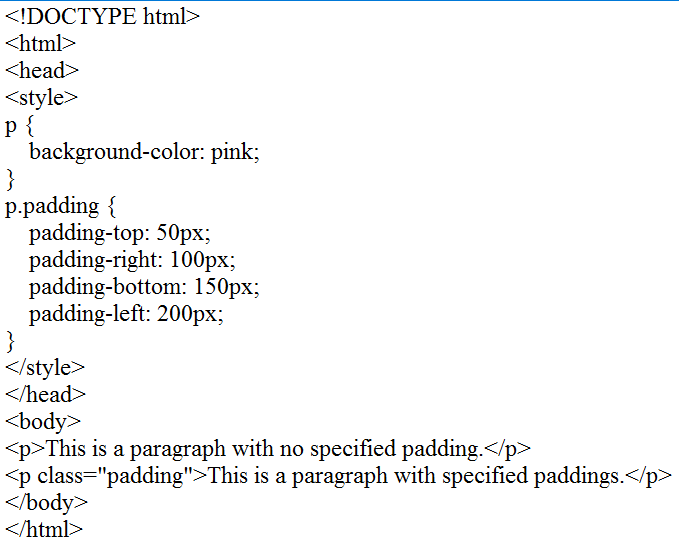
**CSS Padding property** is used *to define the space between the element content and the element border*.

It is different from CSS margin in the way that CSS margin defines the space around elements. CSS padding is affected by the background colors. It clears an area around the content.

Top, bottom, left and right padding can be changed independently using separate properties. You can also change all properties at once by using shorthand padding property.

|  |  |
| --- | --- |
| **Property** | **Description** |
| padding | It is used to set all the padding properties in one declaration. |
| padding-left | It is used to set left padding of an element. |
| padding-right | It is used to set right padding of an element. |
| padding-top | It is used to set top padding of an element. |
| padding-bottom | It is used to set bottom padding of an element. |

Example:



Output:



## CSS Height/Width

The CSS height and width property are used to specify the height and width of an element respectively. We can also set the maximum and minimum values for these properties using the max-height, max-width, min-height, and min-width properties.

#### Syntax

The syntax of CSS height and CSS width property is as follows −

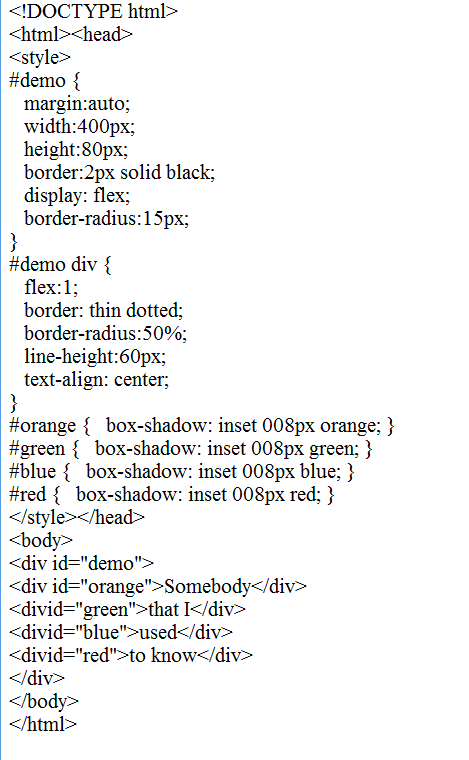
Selector {

height: /\*value\*/ width: /\*value\*/

}

|  |  |
| --- | --- |
| **Box Size** | **Calculation** |
| Total Width | width + padding-left + padding-right + border-left + border-right + margin-left + margin-right |
| Total Height | height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom |

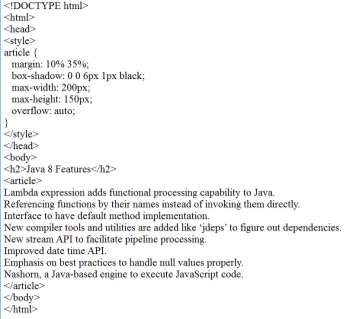
Example:



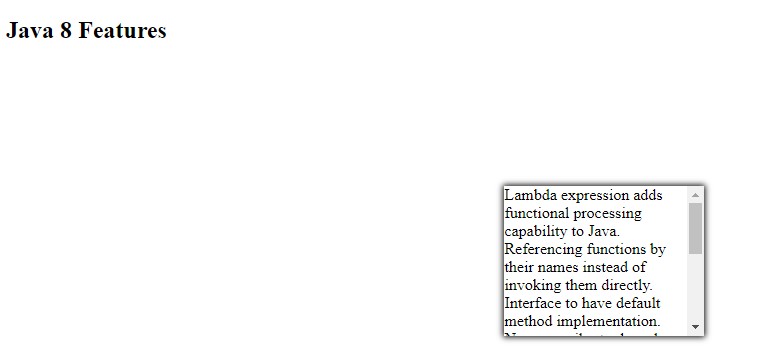
Output



Example:



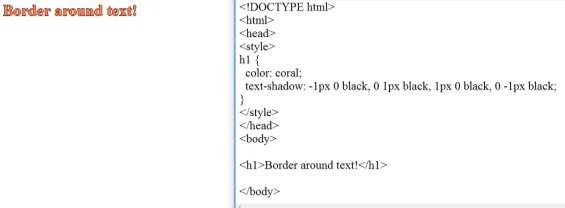
Output:



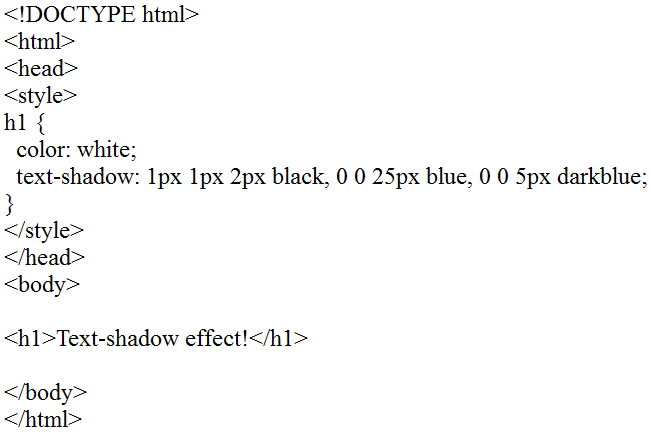
#### CSS SHADOWS

with css you can add shadow to text and to elements.

* Text-shadow
* Box-shadow Example:



Example:

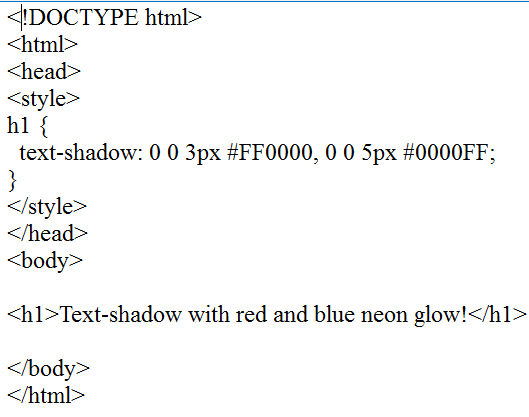


Output:



Example:

.



Output:



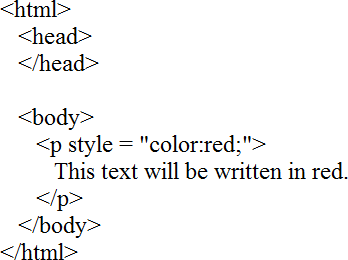
#### CSS Text

* The **color** property is used to set the color of a text.
* The **direction** property is used to set the text direction.
* The **letter-spacing** property is used to add or subtract space between the letters that make up a word.
* The **word-spacing** property is used to add or subtract space between the words of a sentence.
* The **text-indent** property is used to indent the text of a paragraph.
* The **text-align** property is used to align the text of a document.
* The **text-decoration** property is used to underline, overline, and strikethrough text.
* The **text-transform** property is used to capitalize text or convert text to uppercase or lowercase letters.
* The **white-space** property is used to control the flow and formatting of text.
* The **text-shadow** property is used to set the text shadow around a text.

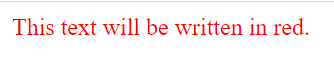
#### Set the Text Color

The following example demonstrates how to set the text color. Possible value could be any color name in any valid format.

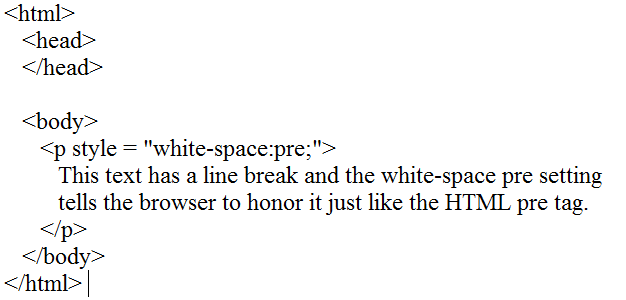
Example:



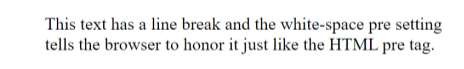
Output:



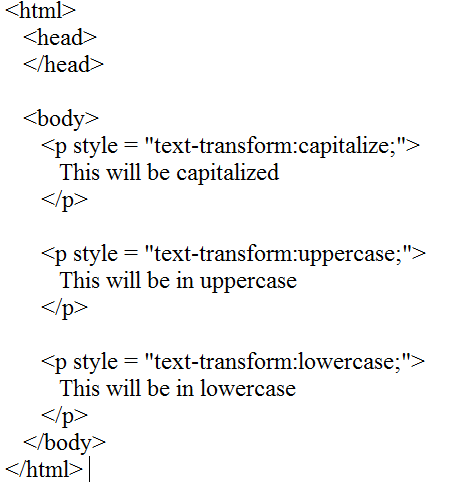
Example:



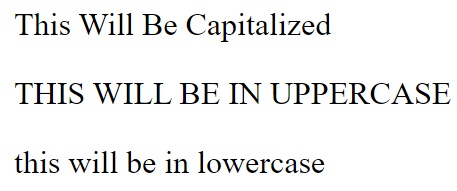
Output:



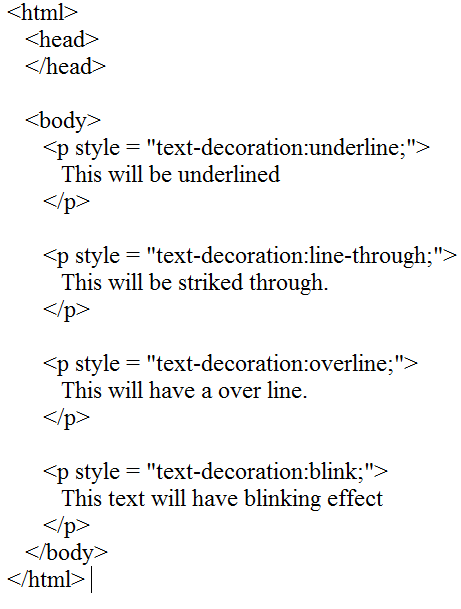
Example:



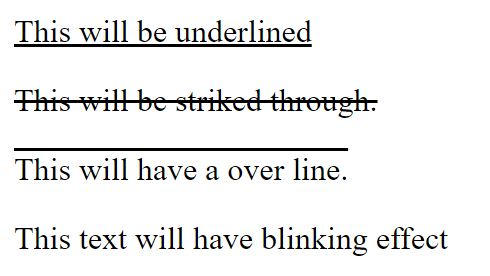
Output:



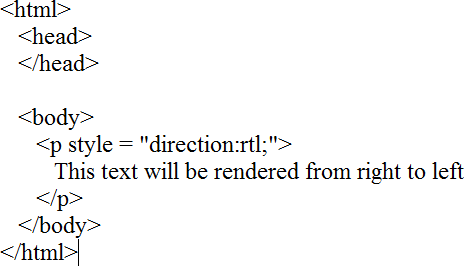
Example:



Output:



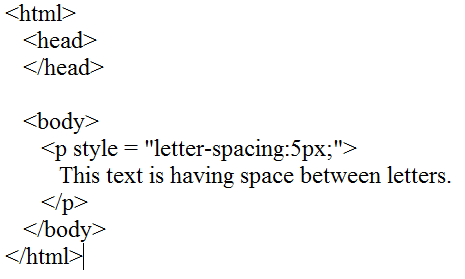
Example:



Output:



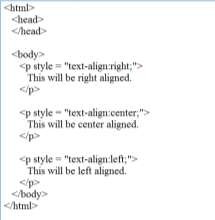
Example:



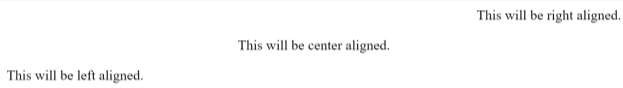
Output:



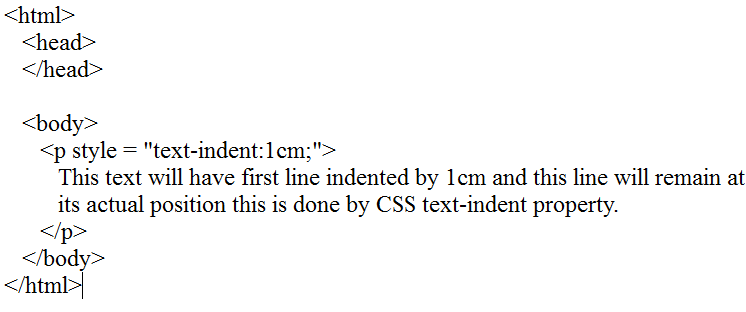
Example



Output:



Example:



Output:



CSS Font property is used to control the look of texts. By the use of CSS font property you can change the text size, color, style and more. You have already studied how to make text bold or underlined. Here, you will also know how to resize your font using percentage.

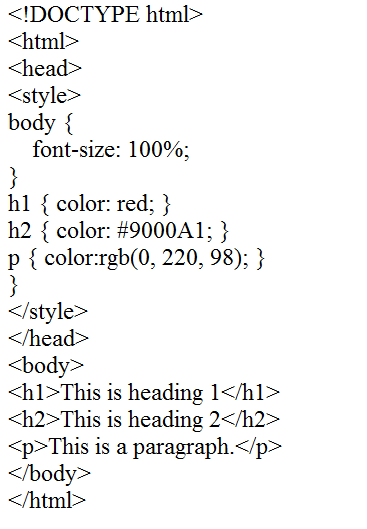
These are some important font attributes:

1. **CSS Font color**: This property is used to change the color of the text. (standalone attribute)
2. **CSS Font family**: This property is used to change the face of the font.
3. **CSS Font size**: This property is used to increase or decrease the size of the font.
4. **CSS Font style**: This property is used to make the font bold, italic or oblique.
5. **CSS Font variant**: This property creates a small-caps effect.
6. **CSS Font weight**: This property is used to increase or decrease the boldness and lightness of the font.

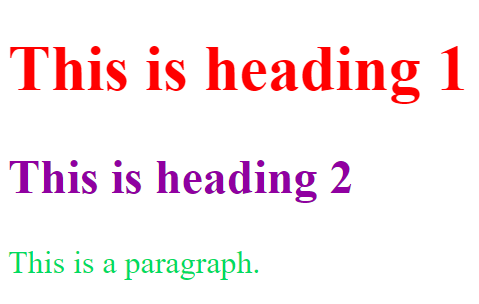
There are three different formats to define a color:

* + By a color name
  + By hexadecimal value
  + By RGB

CSS font color is a standalone attribute in CSS although it seems that it is a part of CSS fonts. It is used to change the color of the text.



Output:



#### CSS Font Family

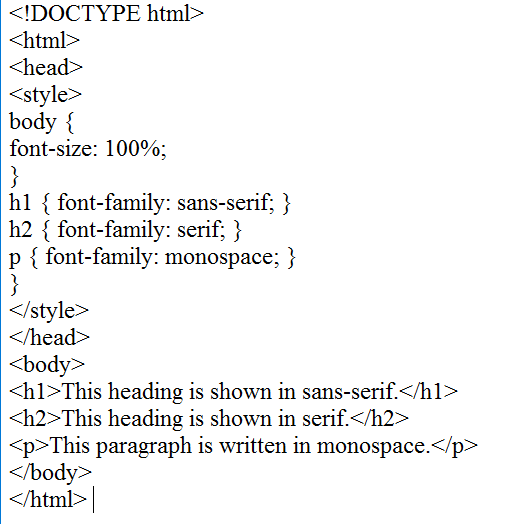
CSS font family can be divided in two types:

* + Generic family: It includes Serif, Sans-serif, and Monospace.
  + Font family: It specifies the font family name like Arial, New Times Roman etc.

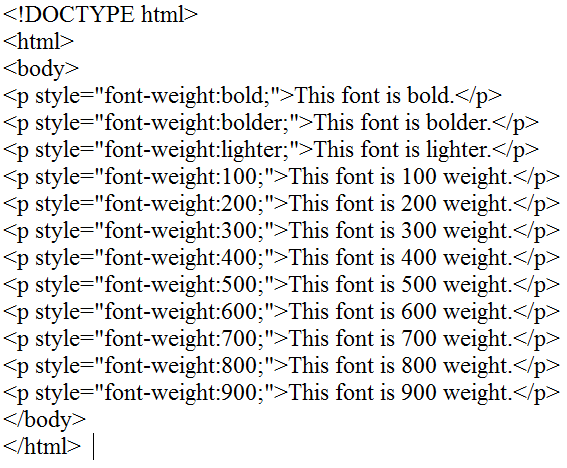
**Serif**: Serif fonts include small lines at the end of characters. Example of serif: Times new roman, Georgia etc.

**Sans-serif**: A sans-serif font doesn't include the small lines at the end of characters. Example of Sans-serif: Arial, Verdana etc.

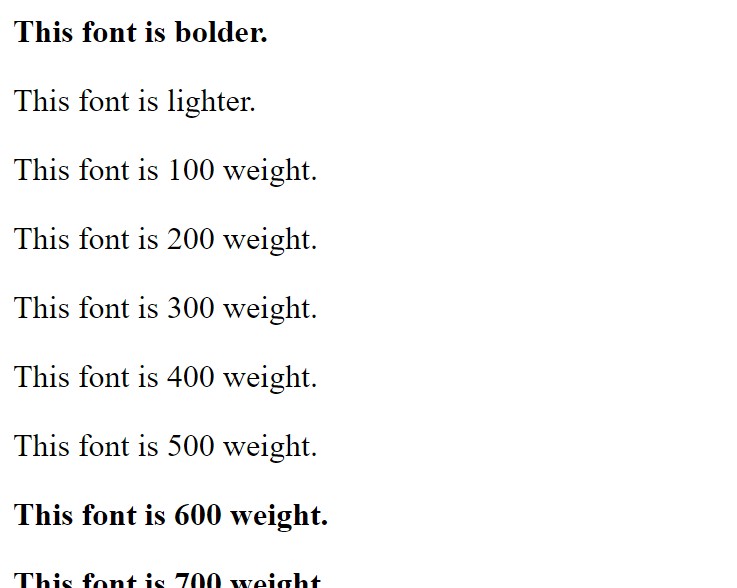
Example:



Output:



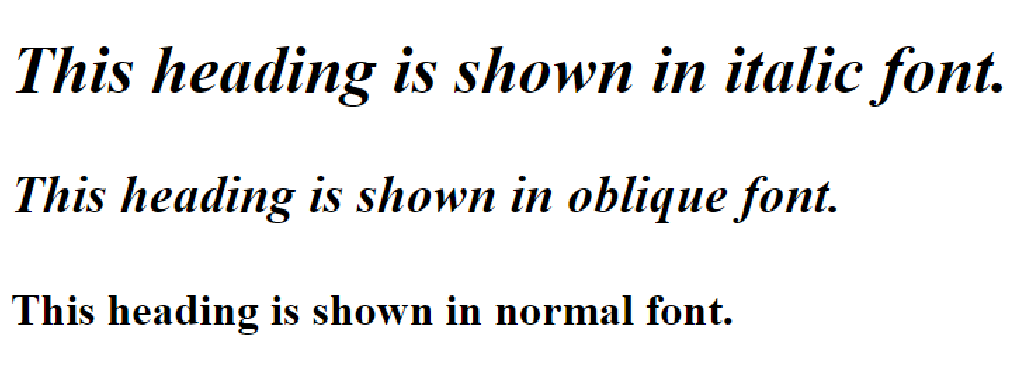
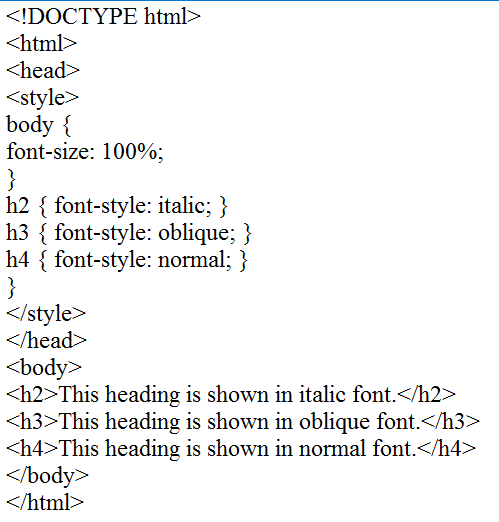
Output:



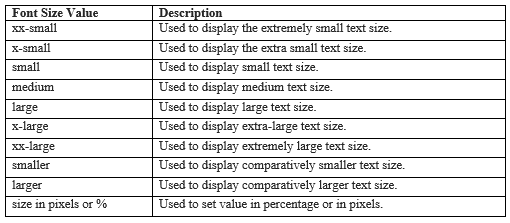
## CSS font

CSS Font style property defines what type of font you want to display. It may be italic, oblique, or normal.

Example:



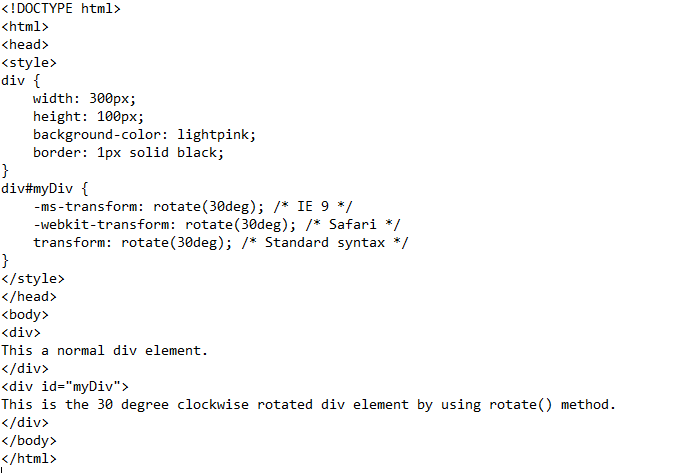
Output:



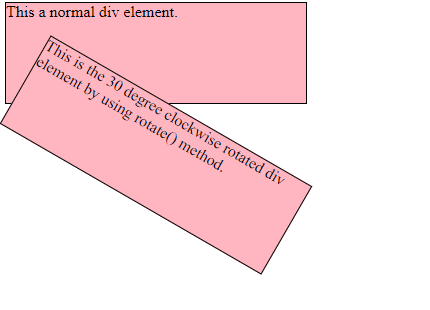
#### CSS 2D Transforms

|  |  |
| --- | --- |
| **Sr.No.** | **Value & Description** |
| 1 | **matrix(n,n,n,n,n,n)**  Used to defines matrix transforms with six values |
| 2 | **translate(x,y)**  Used to transforms the element along with x-axis and y-axis |
| 3 | **translateX(n)**  Used to transforms the element along with x-axis |
| 4 | **translateY(n)**  Used to transforms the element along with y-axis |
| 5 | **scale(x,y)**  Used to change the width and height of element |
| 6 | **scaleX(n)**  Used to change the width of element |
| 7 | **scaleY(n)**  Used to change the height of element |
| 8 | **rotate(angle)**  Used to rotate the element based on an angle |
| 9 | **skewX(angle)**  Used to defines skew transforms along with x axis |
| 10 | **skewY(angle)**  Used to defines skew transforms along with y axis |

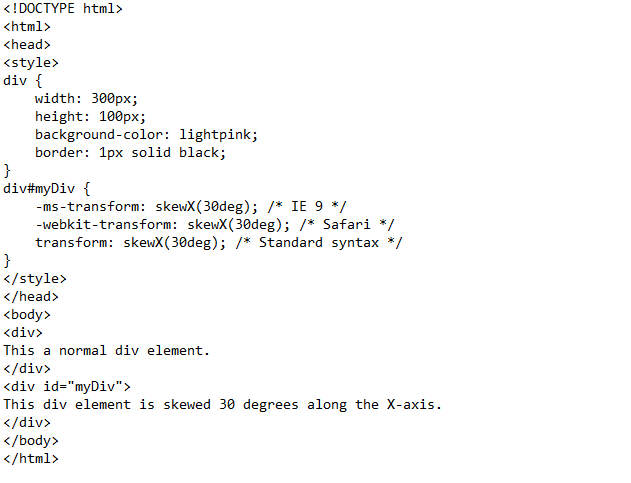
**Example**



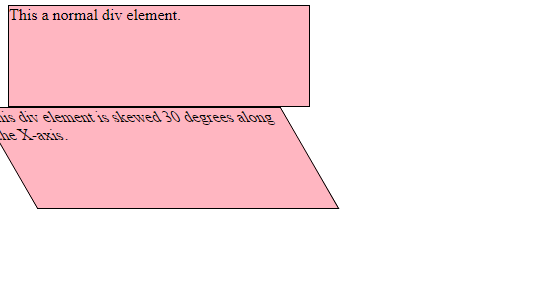
#### Output:

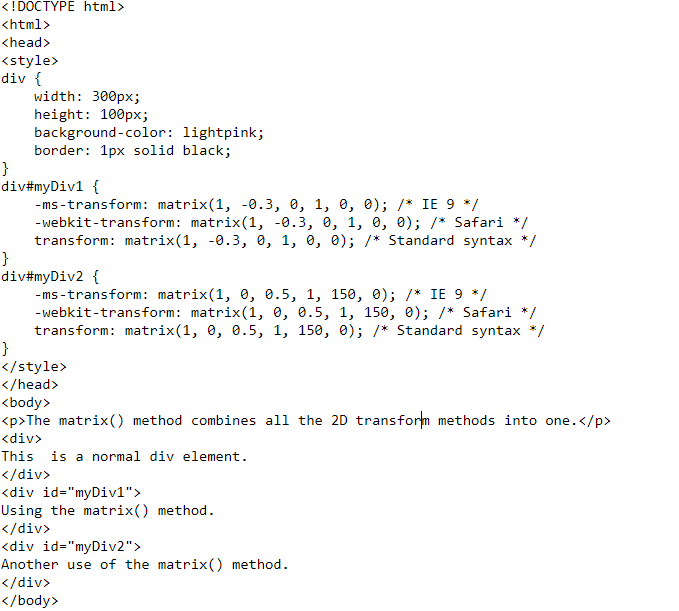


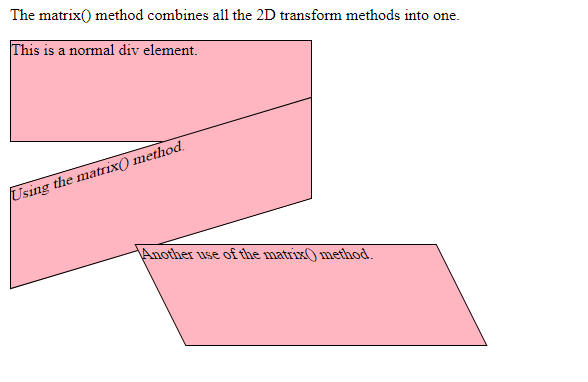
Example:



**Output:**



Exmaple:

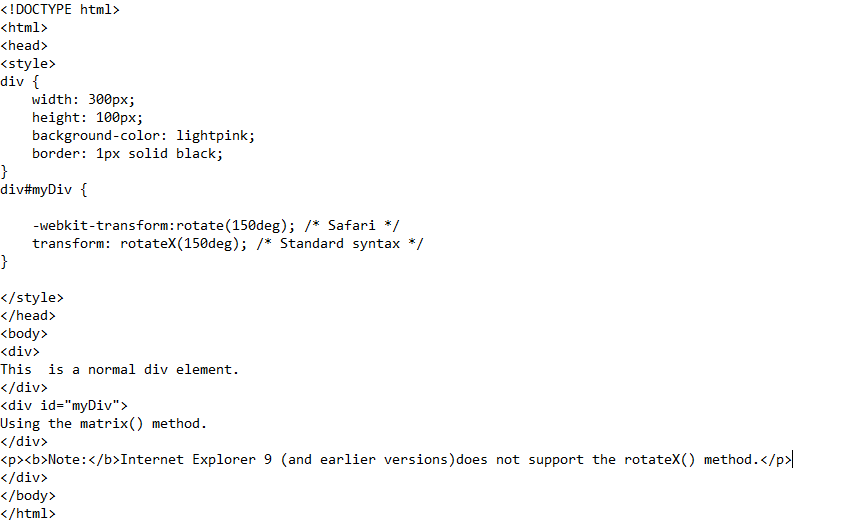
Output:

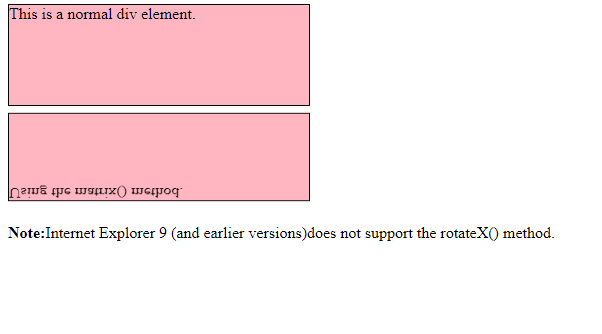
|  |  |
| --- | --- |
| **Function** | **Description** |
| matrix3D(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n) | It specifies a 3D transformation, using a 4x4  matrix of 16 values. |
| translate3D(x,y,z) | It specifies a 3D translation. |
| translateX(x) | It specifies 3D translation, using only the value  for the X-axis. |
| translateY(y) | It specifies 3D translation, using only the value  for the Y-axis. |
| translateZ(z) | It specifies 3D translation, using only the value  for the Z-axis. |
| scale3D(x,y,z) | It specifies 3D scale transformation |
| scaleX(x) | It specifies 3D scale transformation by giving a  value for the X-axis. |
| scaley(y) | It specifies 3D scale transformation by giving a  value for the Y-axis. |
| scaleZ(z) | It specifies 3D scale transformation by giving a  value for the Z-axis. |
| rotate3D(X,Y,Z,angle) | It specifies 3D rotation along with X-axis, Y-  axis and Z-axis. |
| rotateX(angle) | It specifies 3D rotation along with X-axis. |
| rotateY(angle) | It specifies 3D rotation along with Y-axis. |
| rotateZ(angle) | It specifies 3D rotation along with Z-axis. |
| perspective(n) | It specifies a perspective view for a 3D  transformed element. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Property** | **Chrome** | **IE** | **Firefox** | **Opera** | **Safari** |

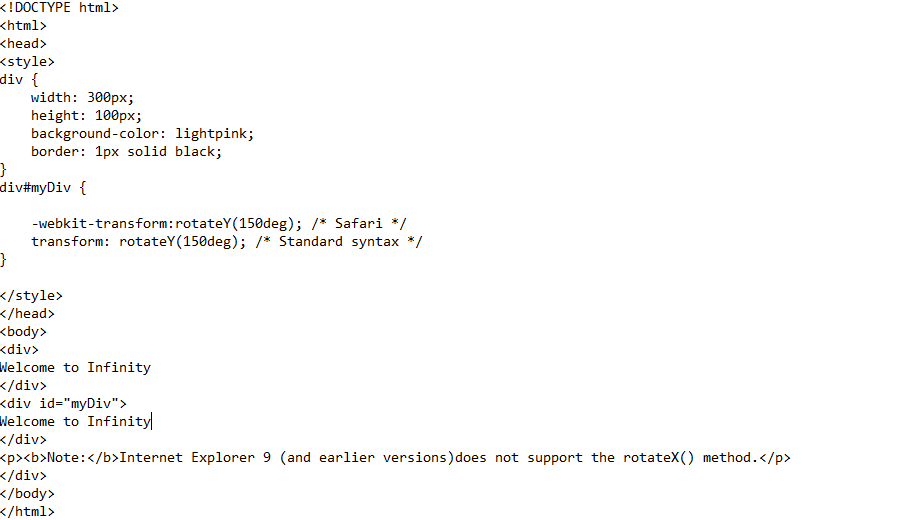
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| transform | 36.012.0 - webkit- | 10.0 | 16.0  10.0 -moz- | 23.015.0 - webkit- | 9.04.0 - webkit- |
| transform-origin (three-value syntax) | 36.012.0 - webkit- | 10.0 | 16.010.0 - moz- | 23.015.0 - webkit- | 9.04.0 - webkit- |
| transform-style | 36.012.0 - webkit- | 11.0 | 16.010.0 - moz- | 23.015.0 - webkit- | 9.04.0 - webkit- |
| perspective | 36.012.0 - webkit- | 10.0 | 16.010.0 - moz- | 23.015.0 - webkit- | 9.04.0 - webkit- |
| perspective- origin | 36.012.0 - webkit- | 10.0 | 16.010.0 - moz- | 23.015.0 - webkit- | 9.04.0 - webkit- |
| backface- visibility | 36.0  12.0 -webkit- | 10.0 | 16.010.0 - moz- | 23.015.0 - webkit- | 9.04.0 - webkit- |

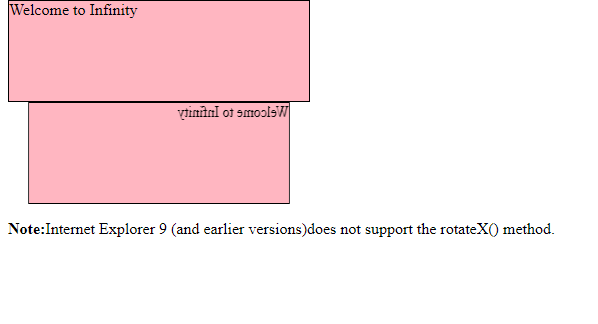
Example:

Output:

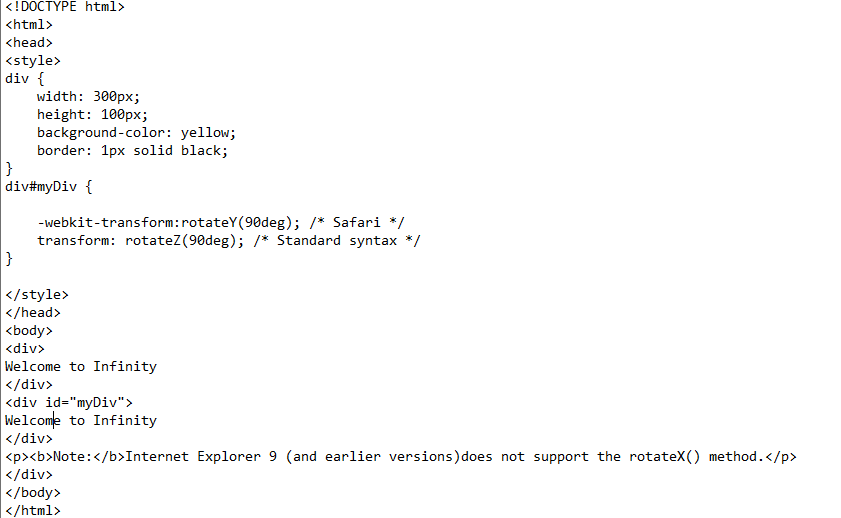


Example:

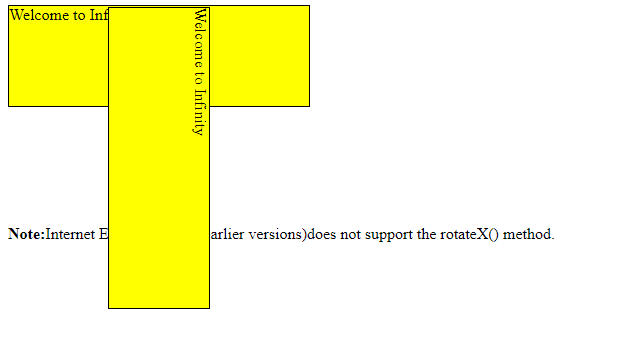




Example:



Output:

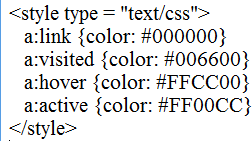


**CSS links**

We will revisit the same properties when we will discuss Pseudo-Classes of CSS.

* 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.

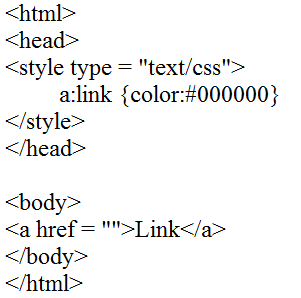
Remember a: **hover** MUST come after a: **link** and a: **visited** in the CSS definition in order to be effective. Also, a: **active** MUST come after a: **hover** in the CSS definition as follows −



Now, we will see how to use these properties to give different effects to hyperlinks.

#### Set the Color of Links

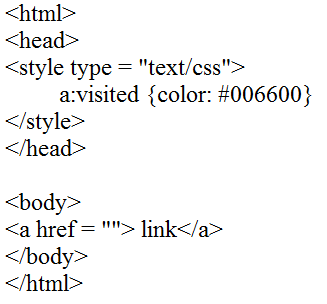
The following example demonstrates how to set the link color. Possible values could be any color name in any valid format.



It will produce the following black link −

#### Set the Color of Visited Links

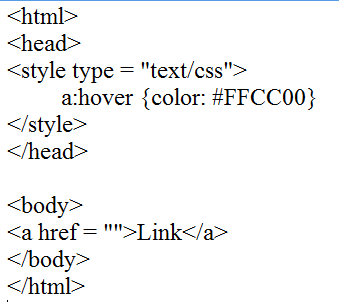
The following example demonstrates how to set the color of visited links. Possible values could be any color name in any valid format.



It will produce the following link. Once you will click this link, it will change its color to green.

#### Change the Color of Links when Mouse is over

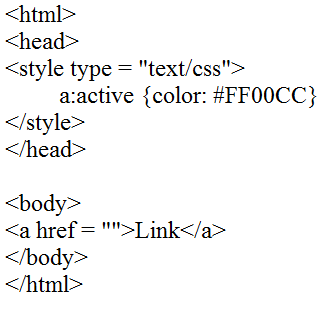
The following example demonstrates how to change the color of links when we bring a mouse pointer over that link. Possible values could be any color name in any valid format.



It will produce the following link. Now, you bring your mouse over this link and you will see that it changes its color to yellow.

#### Change the Color of Active Links

The following example demonstrates how to change the color of active links. Possible values could be any color name in any valid format.



#### CSS Lists

There are various CSS properties that can be used to control lists. Lists can be classified as ordered lists and unordered lists. In ordered lists, marking of the list items is with alphabet and numbers, whereas in unordered lists, the list items are marked using bullets.

We can style the lists using CSS. CSS list properties allow us to:

* + Set the distance between the text and the marker in the list.
  + Specify an image for the marker instead of using the number or bullet point.
  + Control the marker appearance and shape.
  + Place the marker outside or inside the box that contains the list items.
  + Set the background colors to list items and lists.

The CSS properties to style the lists are given as follows:

* + **List-style-type:** This property is responsible for controlling the appearance and shape of the marker.
  + **List-style-image:** It sets an image for the marker instead of the number or a bullet point.
  + **List-style-position:** It specifies the position of the marker.
  + **List-style:** It is the shorthand property of the above properties.
  + **Marker-offset:** It is used to specify the distance between the text and the marker. It is unsupported in IE6 or Netscape 7.

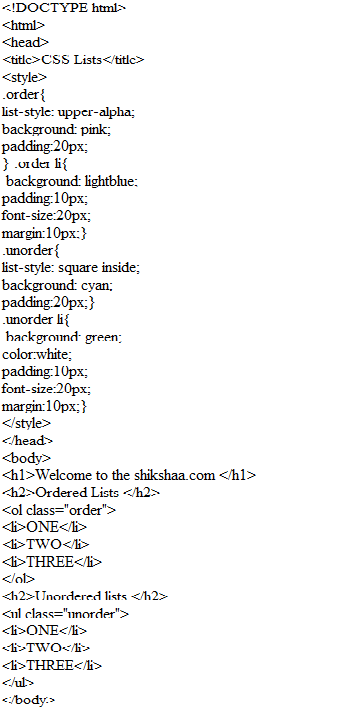
#### The list-style-type property

It allows us to change the default list type of marker to any other type such as square, circle, roman numerals, Latin letters, and many more. By default, the ordered list items are numbered with Arabic numerals (1, 2, 3, etc.), and the items in an unordered list are marked with round bullets (•).

If we set its value to **none,** it will remove the markers/bullets.

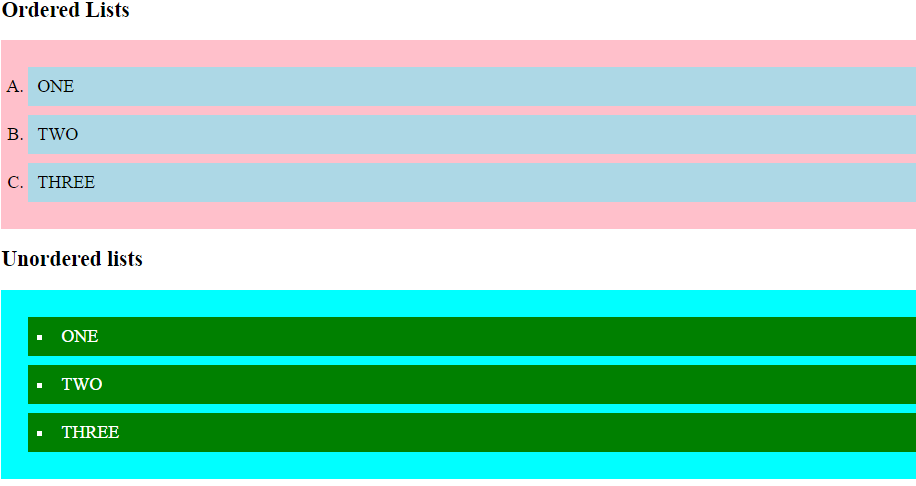
**Example:**

# CSS



CSS

Output:



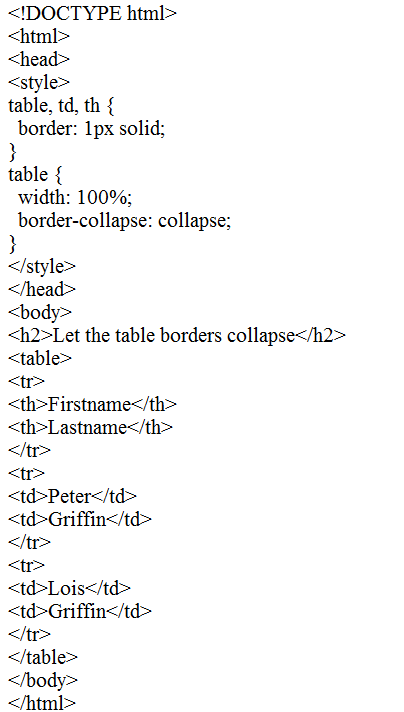
#### CSS TABLES

We can apply style on HTML tables for better look and feel. There are some CSS properties that are widely used in designing table using CSS:

* + border
  + border-collapse
  + padding
  + width
  + height
  + text-align
  + color
  + background-color

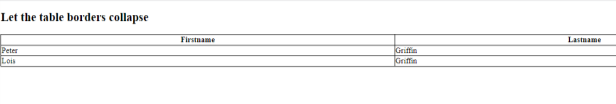
**Example**:

# CSS



CSS

#### Output:

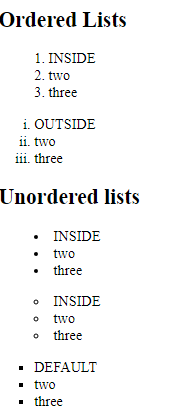


**Example:**

# CSS



Output:



#### CSS Box Model

The components that can be depicted on the web page consist of one or more than one rectangular box.

A CSS box model is a compartment that includes numerous assets, such as edge, border, padding and material. It is used to develop the design and structure of a web page. It can be used as a set of tools to personalize the layout of different components. According to the CSS box model, the web browser supplies each element as a square prism.

The following diagram illustrates how the CSS properties of width, height, padding, border and margin dictate that how much space an attribute will occupy on a web page.

The CSS box model contains the different properties in CSS. These are listed below.

#### Border

* + **Margin**

#### Padding

* + **Content**

Now, we are going to determine the properties one by one in detail.

#### Border Field

It is a region between the padding-box and the margin. Its proportions are determined by the width and height of the boundary.

#### Margin Field

This segment consists of the area between the boundary and the edge of the border.

The proportion of the margin region is equal to the margin-box width and height. It is better to separate the product from its neighbor nodes.

#### Padding Field

This field requires the padding of the component. In essence, this area is the space around the subject area and inside the border-box. The height and the width of the padding box decide its proportions.

#### Content Field

Material such as text, photographs, or other digital media is included in this area.

It is constrained by the information edge, and its proportions are dictated by the width and height of the content enclosure.

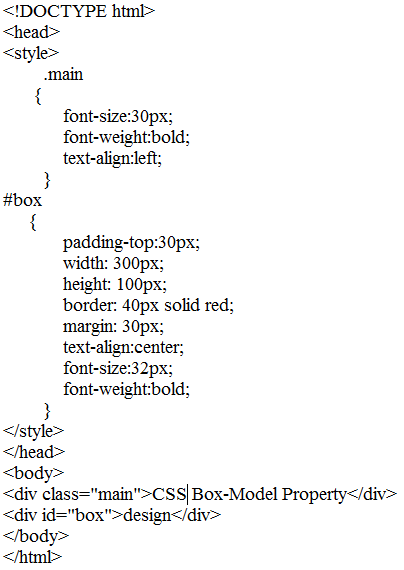
#### Elements of the width and height

Typically, when you assign the width and height of an attribute using the CSS width and height assets, it means you just positioned the height and width of the subject areas of that component. The additional height and width of the unit box is based on a range of influences.

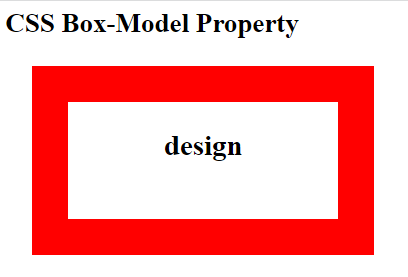
The specific area that an element box may occupy on a web page is measured as follows-

|  |  |
| --- | --- |
| **Size of the box** | **Properties of CSS** |
| Height | height + padding-top + padding-bottom + border-top + border-bottom + margin- top + margin-bottom |
| Width | width + padding-left + padding-right + border-left + border-right + margin-left  + margin-right |

Example:



Output:

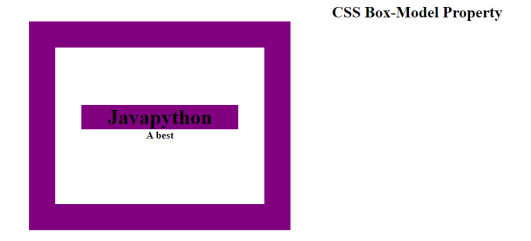


Example:





#### Output:

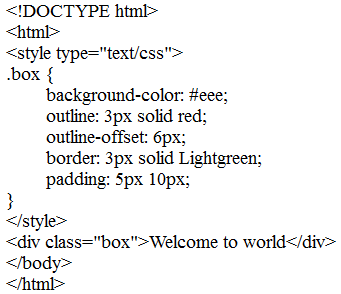


**CSS Outline**

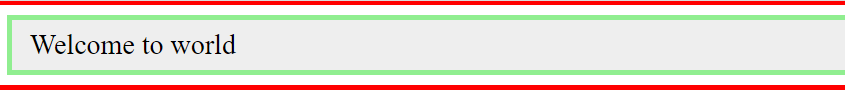
CSS outline is just like CSS border property. It facilitates you to draw an extra border around an element to get visual attention.

It is as easy as to apply as a border.

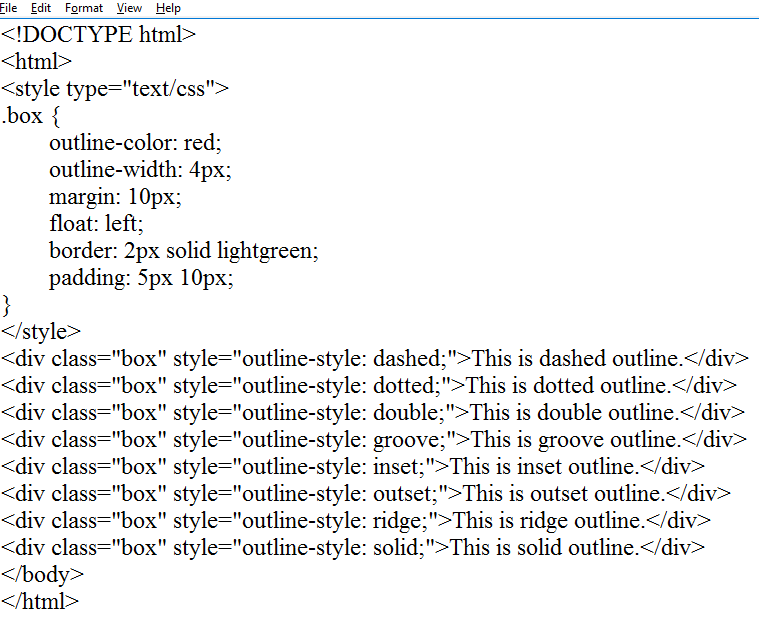
#### Example:



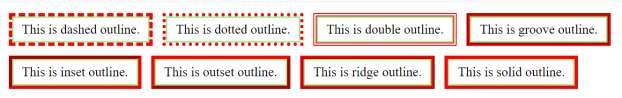
**Output**:



Example:



#### Output:



In the above example, you can see the three outline properties:

* + **Outline-width:** It is similar to margin and padding. It can be either an absolute value or a relative value or one of the predefined outline width values i.e. thin, medium or thick. It is preferred to use either an absolute value or a relative value because different browsers interpret differently while using predefined outline width values like thin, medium or thick.
  + **Outline-color:** It specifies the color that you use for the outline. It supports all the colors available in HTML and CSS.
  + **Outline-style:** In the above example, we have used only solid outline style while there are a lot of outline style i.e. hidden, dotted, dashed, solid, double, groove, ridge, inset and outset.

#### CSS Display

CSS display is the most important property of CSS which is used to control the layout of the element. It specifies how the element is displayed.

Every element has a default display value according to its nature. Every element on the webpage is a rectangular box and the [CSS](https://www.javatpoint.com/css-tutorial)

Property defines the behavior of that rectangular box.

|  |  |
| --- | --- |
| default value | inline |
| inherited | no |
| animation supporting | no |
| version | css1 |
| javascript syntax | object.style.display="none" |

#### Syntax:

display:value;

**CSS display values**

There are following CSS display values which are commonly used.

1. display: inline;
2. display: inline-block;
3. display: block;
4. display: run-in;
5. display: none;

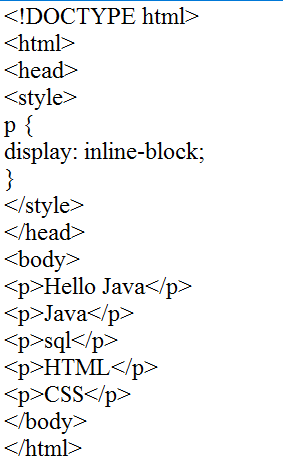
#### CSS display inline

The inline element takes the required width only. It doesn't force the line break so the flow of text doesn't break in inline example.

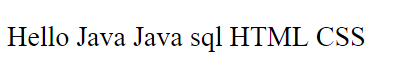
The inline elements are:

* <span>
* <a>
* <em>
* <b> etc.

Let's see an example of CSS display inline. Example:



Output:



#### CSS Position

The **CSS position property** is used *to set position for an element*. it is also used to place an element behind another and also useful for scripted animation effect.

You can position an element using the top, bottom, left and right properties. These properties can be used only after position property is set first. A position element's computed position property is relative, absolute, fixed or sticky.

Let's have a look at following CSS positioning:

* 1. CSS Static Positioning
  2. CSS Fixed Positioning
  3. CSS Relative Positioning
  4. CSS Absolute Positioning

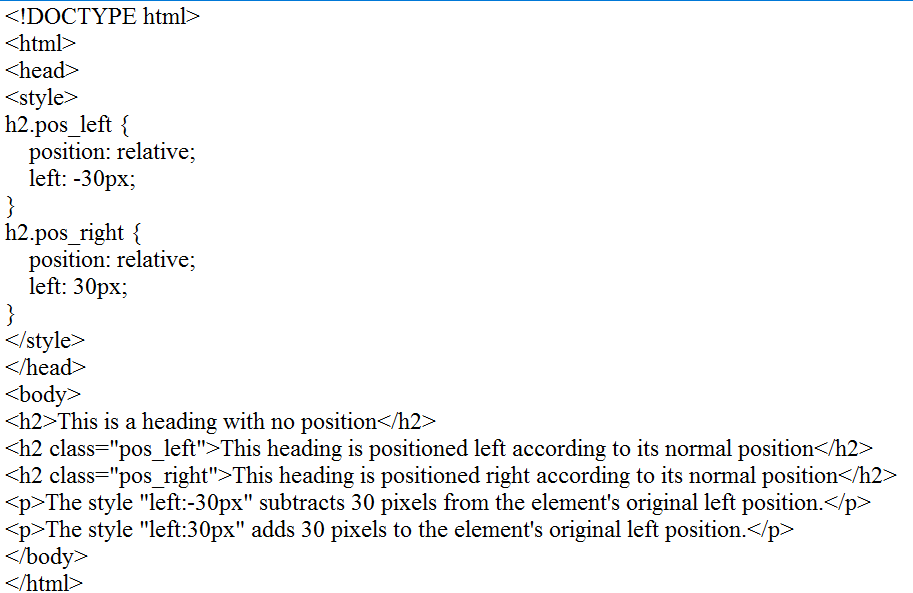
1. CSS Static Positioning

This is a by default position for HTML elements. It always positions an element according to the normal flow of the page. It is not affected by the top, bottom, left and right properties.

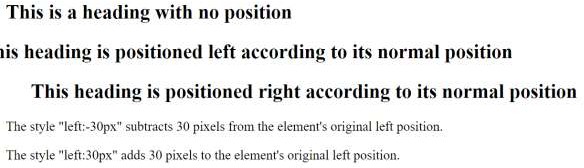
CSS Fixed Positioning

The fixed positioning property helps to put the text fixed on the browser. This fixed test is positioned relative to the browser window, and doesn't move even you scroll the window.

#### Example:



**Output**:

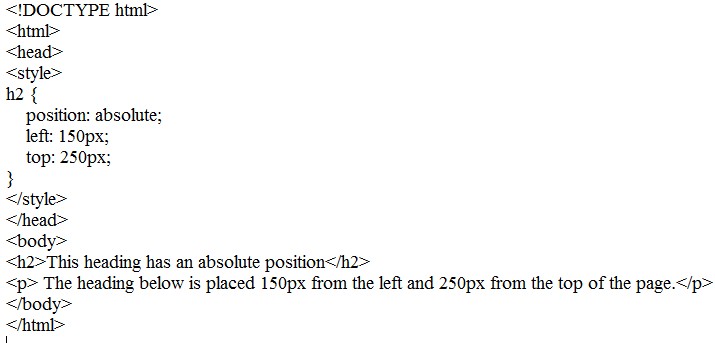


#### CSS Absolute Positioning

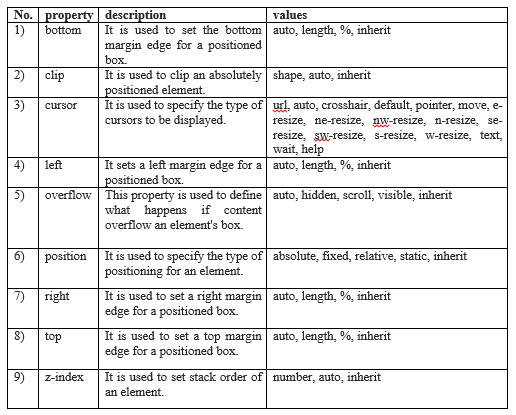
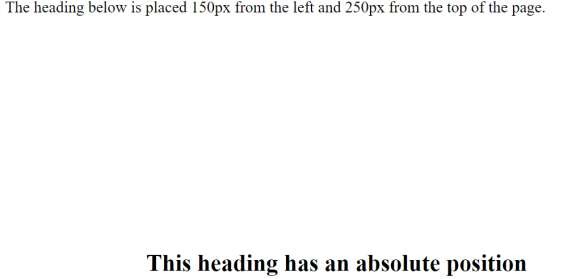
The absolute positioning is used to position an element relative to the first parent element that has a position other than static. If no such element is found, the containing block is HTML.

With the absolute positioning, you can place an element anywhere on a page.

#### Example:



**Output:**



#### CSS Float

The **CSS float property** is *a positioning property*. It is used *to push an element to the left or right*, allowing other element to wrap around it. It is generally used with images and layouts.

To understand its purpose and origin, let's take a look to its print display. In the print display, image is set into the page such that text wraps around it as needed.

#### How it works

Elements are floated only horizontally. So it is possible only to float elements left or right, not up or down.

* 1. A floated element may be moved as far to the left or the right as possible. Simply, it means that a floated element can display at extreme left or extreme right.
  2. The elements after the floating element will flow around it.
  3. The elements before the floating element will not be affected.
  4. If the image floated to the right, the texts flow around it, to the left and if the image floated to the left, the text flows around it, to the right.
  5. CSS Float Properties

|  |  |  |
| --- | --- | --- |
| **Property** | **Description** | **Values** |
| clear | The clear property is used to avoid elements after the floating elements which flow around it. | left, right, both, none, inherit |
| float | It specifies whether the box should float or not. | left, right, none, inherit |

* 1. CSS Float Property Values

|  |  |
| --- | --- |
| **Value** | **Description** |
| none | It specifies that the element is not floated, and will be displayed just where it occurs in the text. this is a default value. |
| left | It is used to float the element to the left. |
| right | It is used to float the element to the right. |
| initial | It sets the property to its initial value. |
| inherit | It is used to inherit this property from its parent element. |

#### CSS Inline-block

The most common answer I heard was: *I always use it on buttons.*

Ultimately, I think that makes sense, but it contributes to an I see as a slight misunderstanding.

The idea is that you want elements that look like buttons (which might be crafted

with <a>, <button>, or perhaps <input>) to lay inline — like they do naturally — but be able to have margin and padding. That’s the slight misunderstanding part: display: inline; elements can *still have* margin and padding, and it probably behaves like you expect it to.

The tricky part is that:

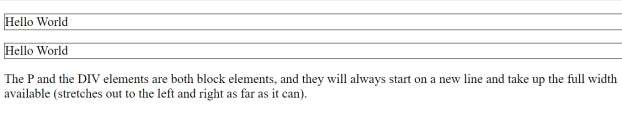
* The block-direction margin on inline elements is ignored entirely
* The padding on inline elements doesn’t affect the height of the line of text

So, while the buttons themselves are pretty much styled just fine, the parent element and surrounding text probably isn’t. Here’s a demo that:

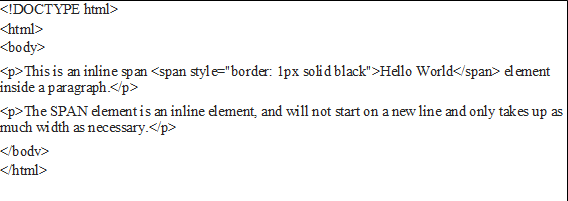
#### Example:



**Output:**



#### Example:



**Output:**



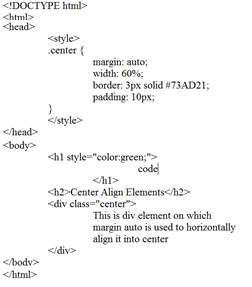
* [<a>](https://www.w3schools.com/tags/tag_a.asp)
* [<abbr>](https://www.w3schools.com/tags/tag_abbr.asp)
* [<acronym>](https://www.w3schools.com/tags/tag_acronym.asp)
* [<b>](https://www.w3schools.com/tags/tag_b.asp)
* [<bdo>](https://www.w3schools.com/tags/tag_bdo.asp)
* [<big>](https://www.w3schools.com/tags/tag_big.asp)
* [<br>](https://www.w3schools.com/tags/tag_br.asp)
* [<button>](https://www.w3schools.com/tags/tag_button.asp)
* [<cite>](https://www.w3schools.com/tags/tag_cite.asp)
* [<code>](https://www.w3schools.com/tags/tag_code.asp)
* [<dfn>](https://www.w3schools.com/tags/tag_dfn.asp)
* [<em>](https://www.w3schools.com/tags/tag_em.asp)
* [<i>](https://www.w3schools.com/tags/tag_i.asp)
* [<img>](https://www.w3schools.com/tags/tag_img.asp)
* [<input>](https://www.w3schools.com/tags/tag_input.asp)
* [<kbd>](https://www.w3schools.com/tags/tag_kbd.asp)
* [<label>](https://www.w3schools.com/tags/tag_label.asp)
* [<map>](https://www.w3schools.com/tags/tag_map.asp)
* [<object>](https://www.w3schools.com/tags/tag_object.asp)
* [<output>](https://www.w3schools.com/tags/tag_output.asp)
* [<q>](https://www.w3schools.com/tags/tag_q.asp)
* [<samp>](https://www.w3schools.com/tags/tag_samp.asp)
* [<script>](https://www.w3schools.com/tags/tag_script.asp)
* [<select>](https://www.w3schools.com/tags/tag_select.asp)
* [<small>](https://www.w3schools.com/tags/tag_small.asp)
* [<span>](https://www.w3schools.com/tags/tag_span.asp)
* [<strong>](https://www.w3schools.com/tags/tag_strong.asp)
* [<sub>](https://www.w3schools.com/tags/tag_sub.asp)
* [<sup>](https://www.w3schools.com/tags/tag_sup.asp)
* [<textarea>](https://www.w3schools.com/tags/tag_textarea.asp)
* [<time>](https://www.w3schools.com/tags/tag_time.asp)
* [<tt>](https://www.w3schools.com/tags/tag_tt.asp)
* [<var>](https://www.w3schools.com/tags/tag_var.asp)

#### CSS Align

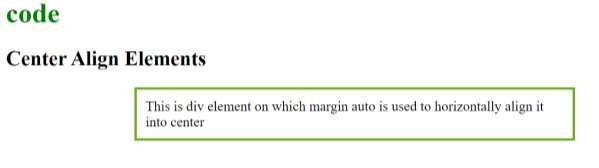
The **align** in CSS is used for positioning the items along with setting the distribution of space between and around content items. We can align the items either horizontally or vertically. The various methods and techniques are used to center them, by taking care of the left and the right margin, etc. The various method for aligning & its usage are discussed below:

**margin: auto:** This property is used to align a block element into the center.

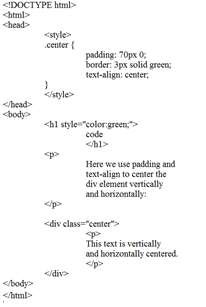
**Note:** Using *margin: auto* will not work in IE8 unless a !DOCTYPE is declared. **Example1:** This example describes the CSS align using the *margin: auto* property. Example:



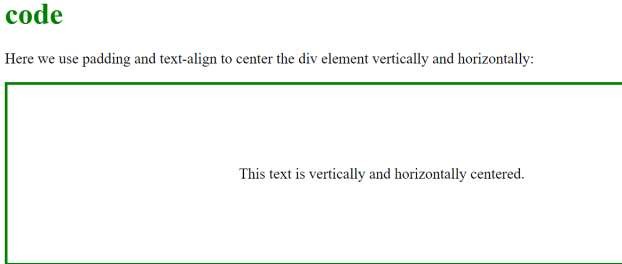
Output:



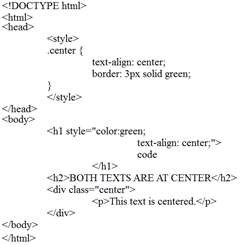
Example:



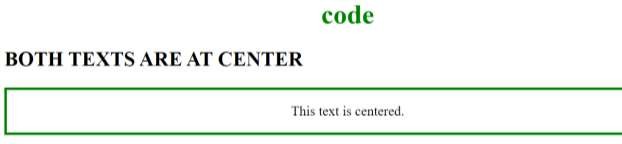
#### Output:



Example:



Output:



#### CSS Combinators

CSS Combinators clarifies the relationship between two selectors, whereas the selectors in CSS are used to select the content for styling.

There can be more than one simple selector in a [CSS selector,](https://www.javatpoint.com/css-selector) and between these selectors, we can include a combinator. Combinators combine the selectors to provide them a useful relationship and the position of content in the document.

There are four types of combinators in [CSS](https://javatpoint.com/css-tutorial) that are listed as follows:

* General sibling selector (~)
* Adjacent sibling selector (+)
* Child selector (>)
* Descendant selector (space)

#### General Sibling Selector (~)

It uses the **tlide (~)** sign as the separator between the elements. It selects the elements that follow the elements of first selector, and both of them are the children of the same parent. It can be used for selecting the group of elements that share the common parent element.

It is useful when we have to select the siblings of an element even if they are not adjacent directly.

#### Syntax

1. element ~ element {
2. /\*style properties\*/ 3. }

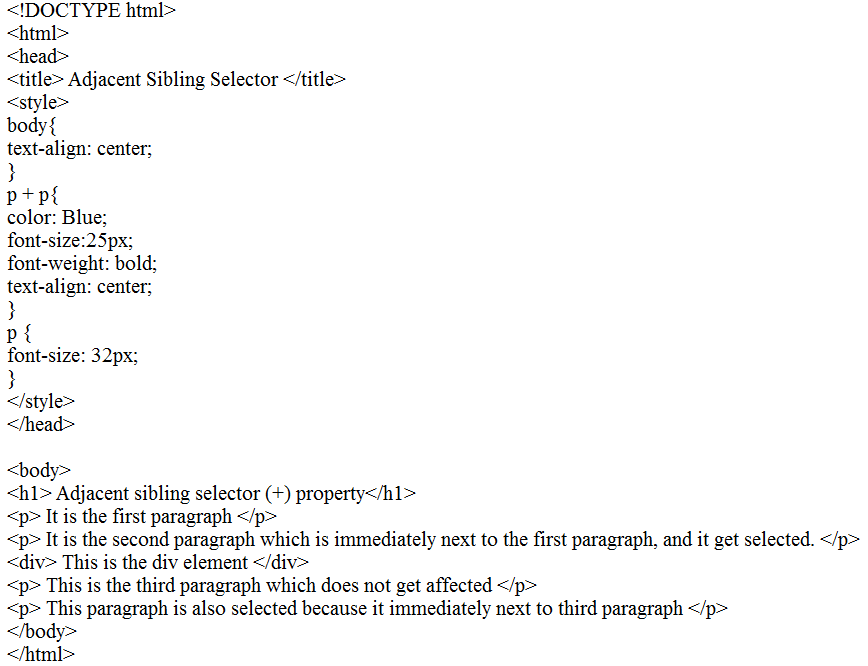
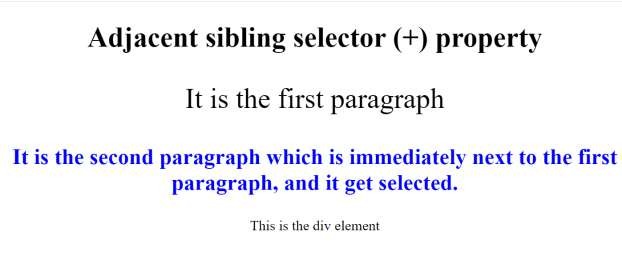
Suppose we have to select all **<p>** elements that are the siblings of **<div>** element, then we can write it as:

|  |  |  |
| --- | --- | --- |
| 1.  2.  3. | div ~ p {  /\*style properties\*/  } |  |

In this example we are selecting the **<p>** element that comes immediately after the **<p>** element. There is an **<div>** element that will not be selected, and the paragraph element after the div will also not be selected. But the **<p>** element that comes just next to the third paragraph will be affected.

#### Example:

**Output:**



#### Child Selector (>)

It uses the greater than **(>)** sign as the separator between the elements. It selects the direct descendant of the parent. This combinator only matches the elements that are the immediate child in the document tree. It is stricter as compared to the descendant selector because it selects the second selector only when the first selector is its parent.

The parent element must always be placed at the left of the **">"**. If we remove the greater than **(>)** symbol that designates this as a child combinator, then it will become the descendant selector.

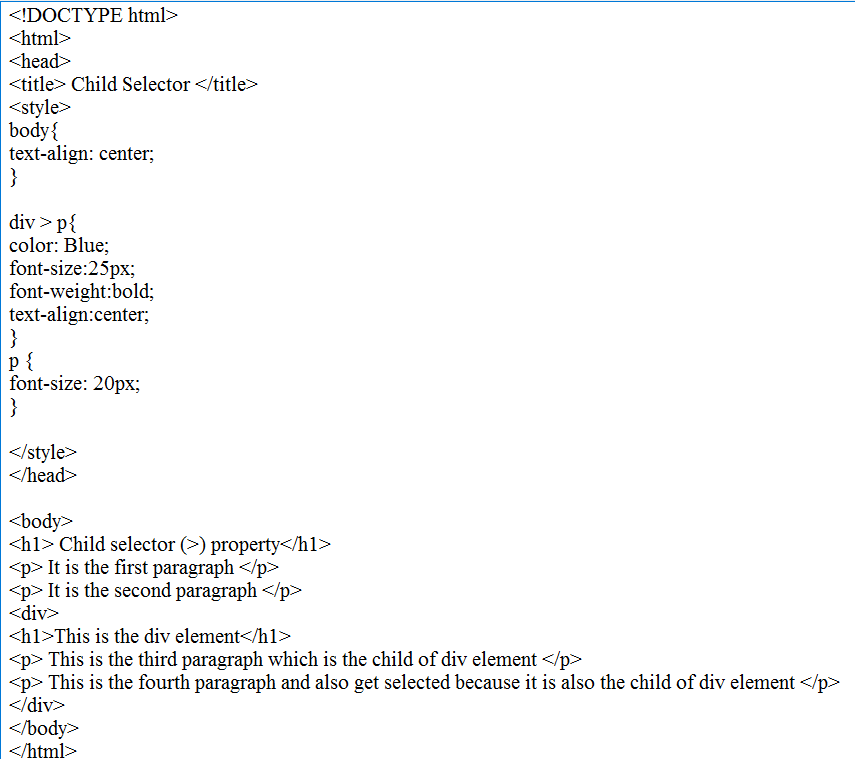
#### Syntax

1. element **>** element {
2. /\*style properties\*/ 3. }

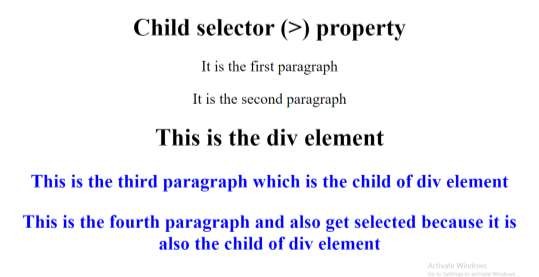
If we have to select the paragraph elements that are the children of **<div>** element then by using the child selector, it will be written as follows:

1. div **>** p {
2. /\*style properties\*/ 3. }

Example:



Output:



#### CSS Pseudo-class

A pseudo-class can be defined as a keyword which is combined to a selector that defines the special state of the selected elements. It is added to the selector for adding an effect to the existing elements based on their states. For example, The **":hover"** is used for adding special effects to an element when the user moves the cursor over the element.

|  |  |
| --- | --- |
| **pseudo- class** | **Description** |
| **:active** | IIt is used to add style to an active element. |
| **:hover** | IIt adds special effects to an element when the user moves the mouse pointer over  the element. |
| **:link** | IIt adds style to the unvisited link. |
| **:visited** | IIt adds style to a visited link. |
| **:lang** | IIt is used to define a language to use in a specified element. |
| **:focus** | IIt selects the element which is focused by the user currently. |
| **:first- child** | It adds special effects to an element, which is the first child of another element. |

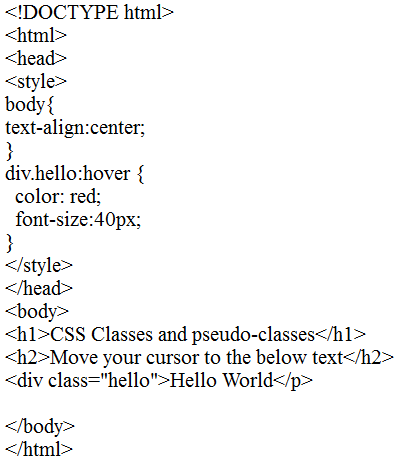
The names of the pseudo-class are not case-sensitive.

#### Syntax

A pseudo-class starts with a colon **(:).** Let's see its syntax.

1. selector: pseudo-class {
2. property: value; 3. }

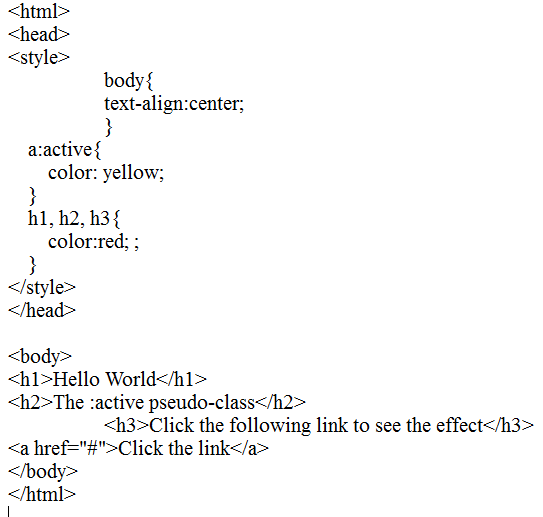
Example:



#### Output:



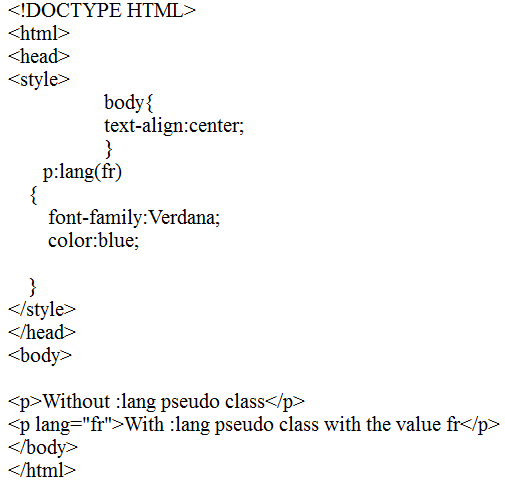
**Example:**



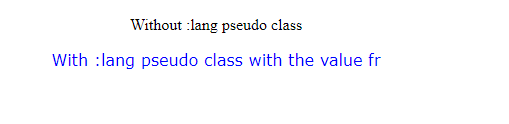
#### Output:



Example:



Output:



#### CSS Pseudo-element

CSS pseudo-elements are used to add special effects to some selectors. You 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 −

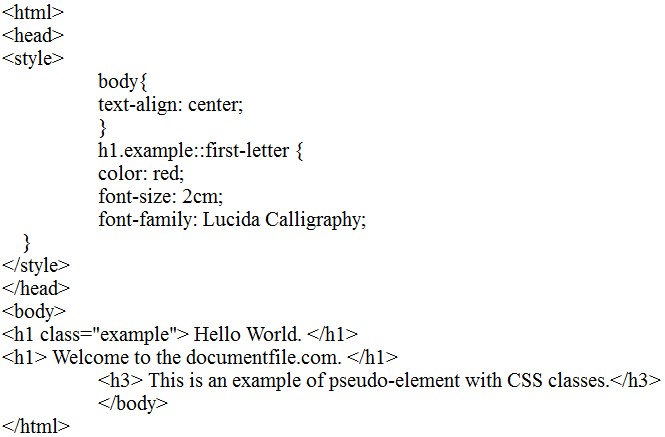
selector.class:pseudo-element {property: value}

The most commonly used pseudo-elements are as follows −

|  |  |
| --- | --- |
| **Sr.No.** | **Value & Description** |
| 1 | **:first-line** |

|  |  |
| --- | --- |
|  | Use this element to add special styles to the first line of the text in a selector. |
| 2 | **:first-letter**  Use this element to add special style to the first letter of the text in a selector. |
| 3 | **:before**  Use this element to insert some content before an element. |
| 4 | **:after**  Use this element to insert some content after an element. |

#### Example:



Output:



A pseudo-class can be defined as a keyword which is combined to a selector that defines the special state of the selected elements. Unlike the pseudo-classes, the pseudo-elements are used to style the specific part of an element, whereas the pseudo-classes are used to style the element.

As an example, a pseudo-element can be used to style the first letter or the first line of an element. The pseudo-elements can also be used to insert the content after or before an element.

#### Syntax

Pseudo-element has a simple syntax which is given as follows:

1. selector::pseudo-element {
2. property: value; 3. }

We have used the **double colon notation (::pseudo-element)** in the syntax. In CSS3, the double colon replaced the single colon notation for pseudo-elements. It was an attempt from W3C to differentiate between the pseudo-elements and pseudo-classes. So, it is recommended to use **double colon notation (::pseudo-element)** instead of using single-colon notation **(:)**.

In CSS1 and CSS2, there is the use of a single colon **(:)** syntax for both pseudo-elements and pseudo-classes. The single colon syntax is valid for pseudo-elements in CSS1 and CSS2 for backward compatibility.

Although there are several [CSS](https://www.javatpoint.com/css-tutorial) pseudo-elements, we are discussing some of the most commonly used. The widely used CSS pseudo-elements are tabulated as follows:

|  |  |
| --- | --- |
| **pseudo-element** | **Description** |
| **::first-letter (:first- letter)** | It selects the first letter of the text. |
| **::first-line (:first-line)** | It styles the first line of the text. |
| **::before (:before)** | It is used to add something before the element's content. |
| **::after (:after)** | It is used to add something after the element's content. |
| **::selection** | It is used to select the area of an element that is selected by the user. |

Let's discuss the above pseudo-elements, along with an example. The: first-letter pseudo-element

As its name implies, it affects the first letter of the text. It can be applied only to block-level

elements. Instead of supporting all CSS properties, it supports some of the CSS properties that are given below.

* Color properties **(such as color)**

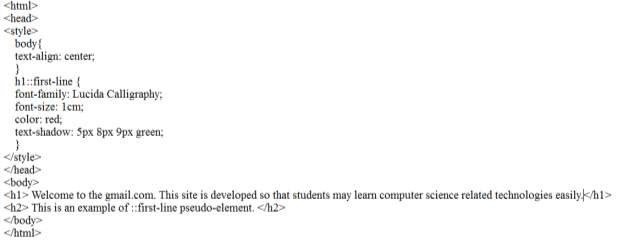
#### Font properties (such as font-style, font-family, font-size, font-color, and many more).

* Margin properties **(such as margin-top, margin-right, margin-bottom, and margin- left)**.

#### Border properties (like border-top, border-right, border-bottom, border-left, border- color, border-width, and many more).

* Padding properties **(such as padding-top, padding-right, padding-bottom, and padding-left)**.
* Background properties **(such as background-color, background-repeat, background- image, and background-position)**.
* Text related properties **(such as text-shadow, text-transform, text-decoration, etc.)**.
* Other properties are **vertical-align** (only when the **float** is '**none**') **word-spacing, line- height, line-spacing, etc.**

#### Example:



Output:



#### CSS Navigation Bar

A Navigation bar or navigation system comes under GUI that helps the visitors in accessing information. It is the UI element on a webpage that includes links for the other sections of the website.

A navigation bar is mostly displayed on the top of the page in the form of a horizontal list of links. It can be placed below the logo or the header, but it should always be placed before the main content of the webpage.

It is important for a website to have easy-to-use navigation. It plays an important role in the website as it allows the visitors to visit any section quickly.

Let's discuss the horizontal navigational bar and vertical navigational bar in detail.

#### Horizontal Navigation Bar

The horizontal navigation bar is the horizontal list of links, which is generally on the top of the page.

Let's see how to create a horizontal navigation bar by using an example.

#### Example

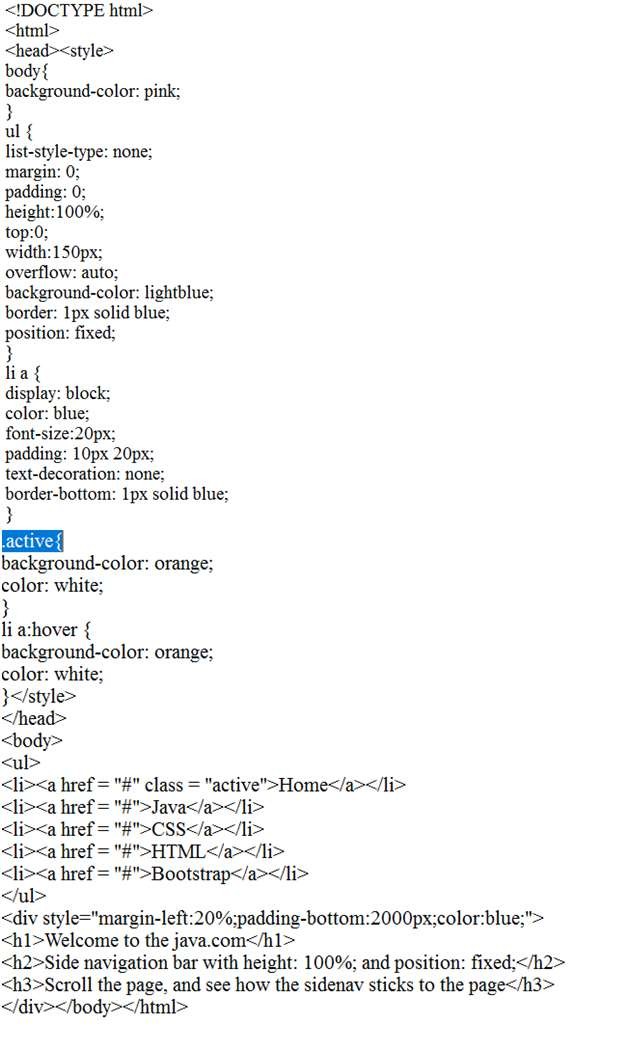
In this example, we are adding the **overflow: hidden** property that prevents the **li** elements from going outside of the list, **display: block** property displays the links as the block elements and makes the entire link area clickable.

We are also adding the **float: left** property, which uses float for getting the block elements to slide them next to each other.

If we want the full-width background color then we have to add the **background-color** property to **<ul>** rather than the **<a>** element.

**Example:**

# CSS



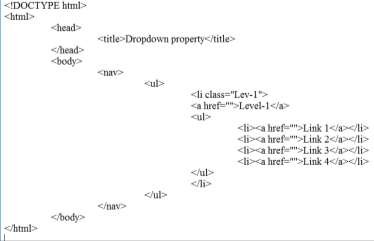
Output:



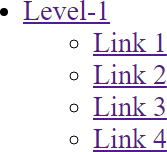
#### CSS Dropdowns

Dropdowns are one of the most important parts of an interactive website. CSS is used to design the drop-down menus. A drop-down is a bunch of lists under an unordered list i.e. <ul> as it is popularly known in HTML world. Nested list (<li>) tags under the <ul> tag used to create drop- down structure. To bring out the effects use CSS for the components present in the structure. The CSS is very straightforward property used to create the drop-down menu.

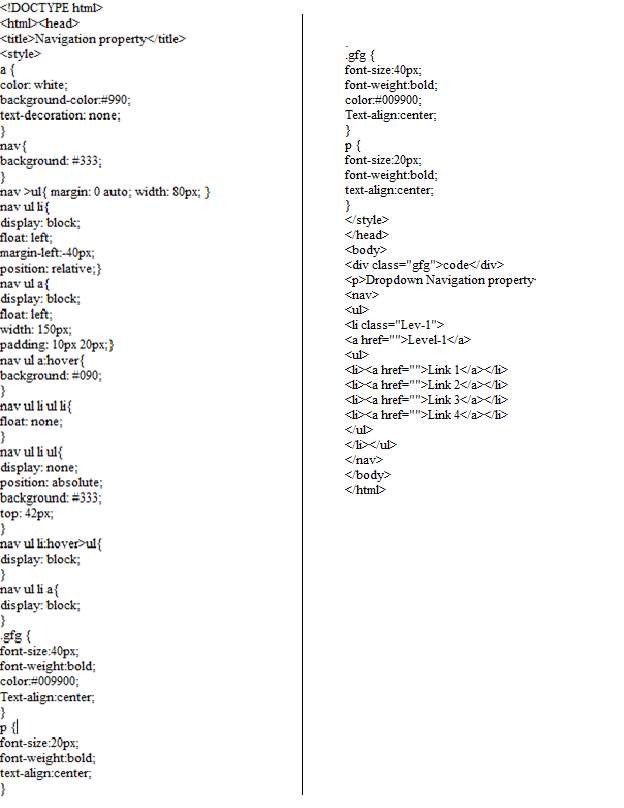
**Example:**



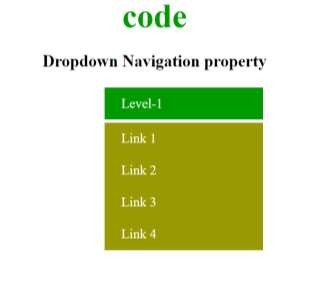
Output:



Example:



Output:



#### CSS Tooltips

CSS Tooltips are a great way to display extra information about something when the user moves the mouse cursor over an element

CSS Tooltips are a great way to display extra information about something when the user moves the mouse cursor over an element.

#### CSS Images

Images are an important part of any web application. Including a lot of images in a web application is generally not recommended, but it is important to use the images wherever they required. CSS helps us to control the display of images in web applications.The styling of an image in CSS is similar to the styling of an element by using the borders and margins.

There are multiple CSS properties such as **border** property, **height** property, **width** property, etc. that helps us to style an image.

Let's discuss the styling of images in CSS by using some illustrations. Thumbnail Image

Images play an important role in any webpage. Though it is not recommended to include a lot of images, but it is still important to use good images wherever required.

CSS plays a good role to control image display. You can set the following image properties using CSS.

* The **border** property is used to set the width of an image border.
* The **height** property is used to set the height of an image.
* The **width** property is used to set the width of an image.
* The **-moz-opacity** property is used to set the opacity of an image.

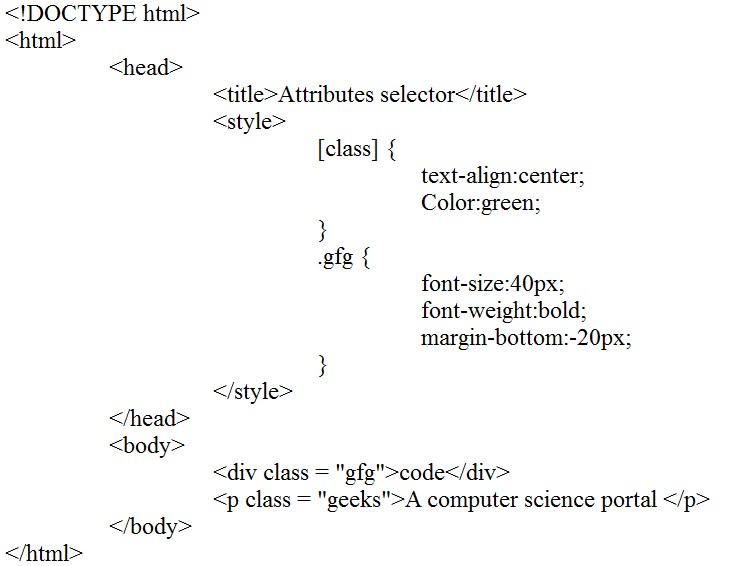
#### CSS Attr Selectors

The CSS Attribute Selector is used to select an element with some specific attribute or attribute value. It is an excellent way to style the HTML elements by grouping them based on some specific attributes and the attribute selector will select those elements with similar attributes.

There are several types of attribute selectors which are discussed below:

* + **[attribute] Selector:** This type of attribute selector is used to select all the elements that have the specified attribute and applies the CSS property to that attribute. For example the selector [class] will select all the elements with the style attribute.

#### Example



**Output:**



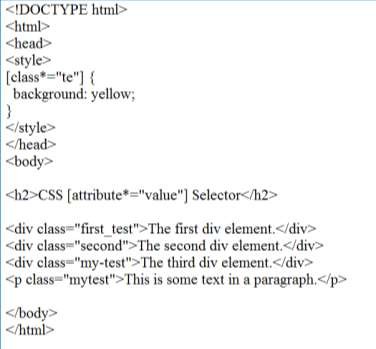
#### Example:



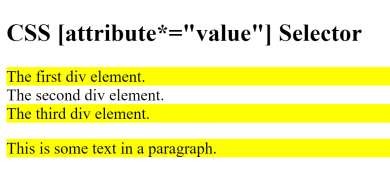
**Output:**



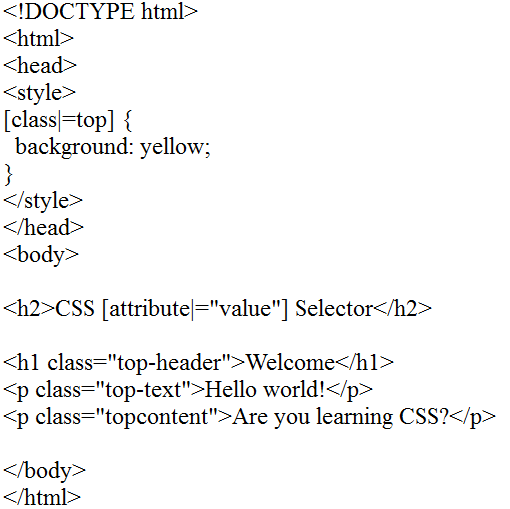
#### Example:



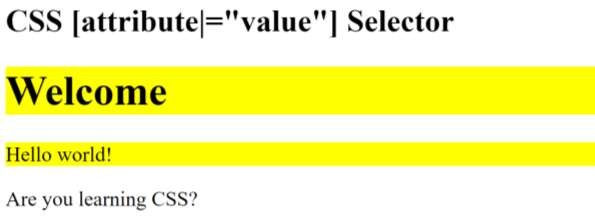
**Output:**



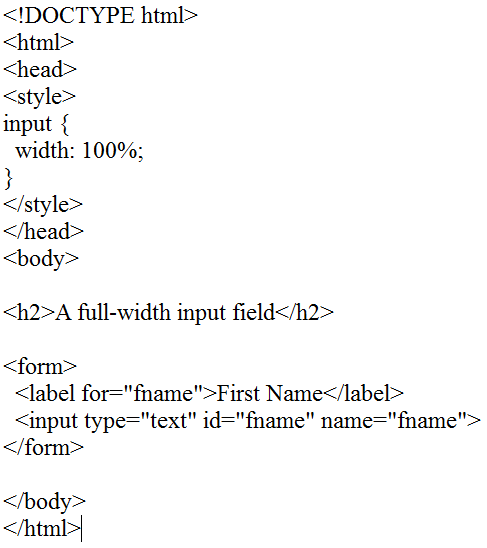
#### Example:



**Output:**



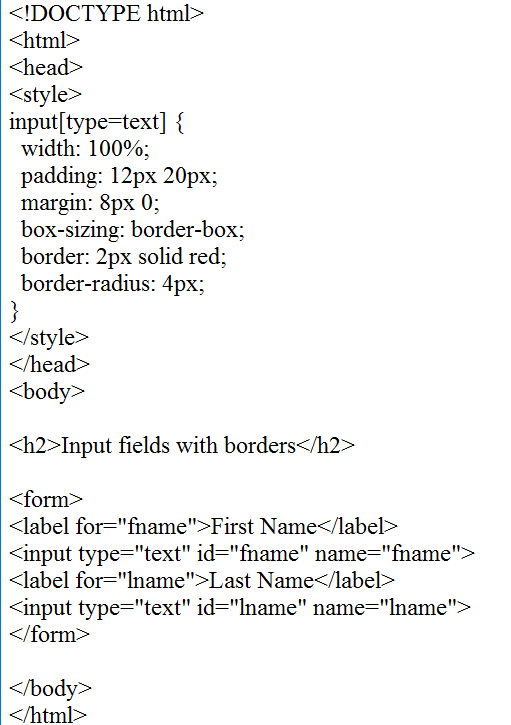
#### CSS Forms



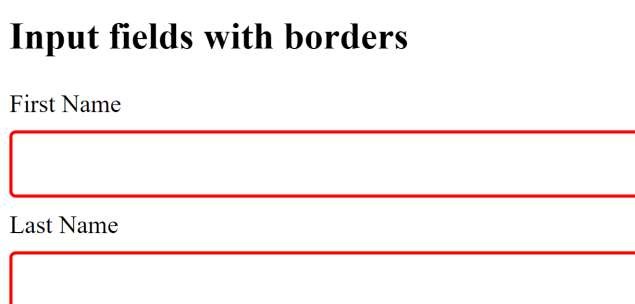
**Output:**



#### Example:



**Output**:



#### CSS Counters

CSS counters are similar to variables. These are maintained by CSS and those values can be incremented by CSS rules to track how many times they are used.

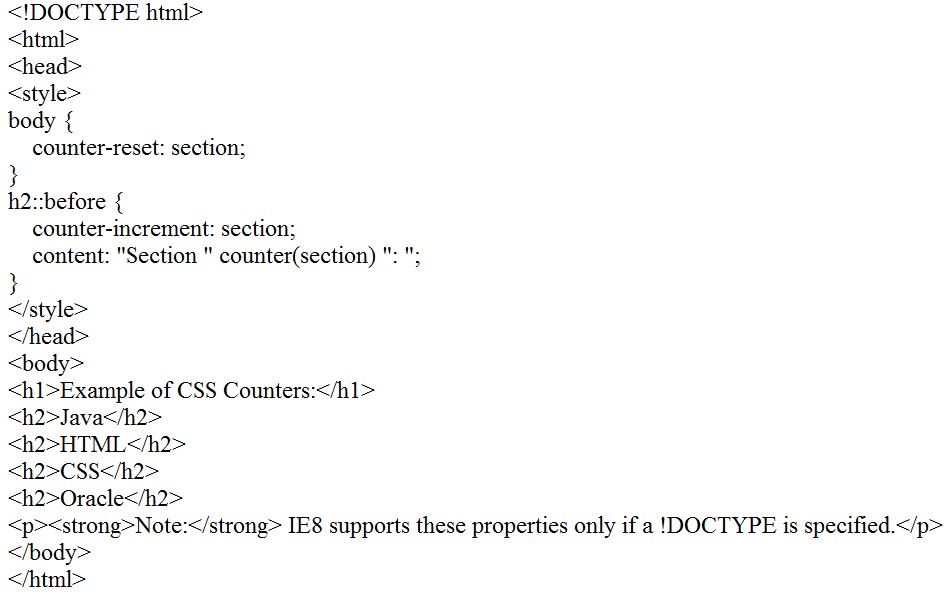
CSS counters facilitate simple CSS based incrementing and display of a number for generated content.

#### CSS Counter Properties

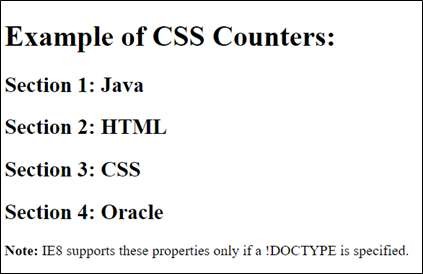
Following is a list of properties that are used with CSS counter:

* **Counter-reset:** It is used to create or reset a counter.
* **Counter-increment:** It is used to increment the counter value.
* **Content:** It is used to insert generated content.
* **Counter() or Counters() function:** It is used to add the value of a counter to an element.

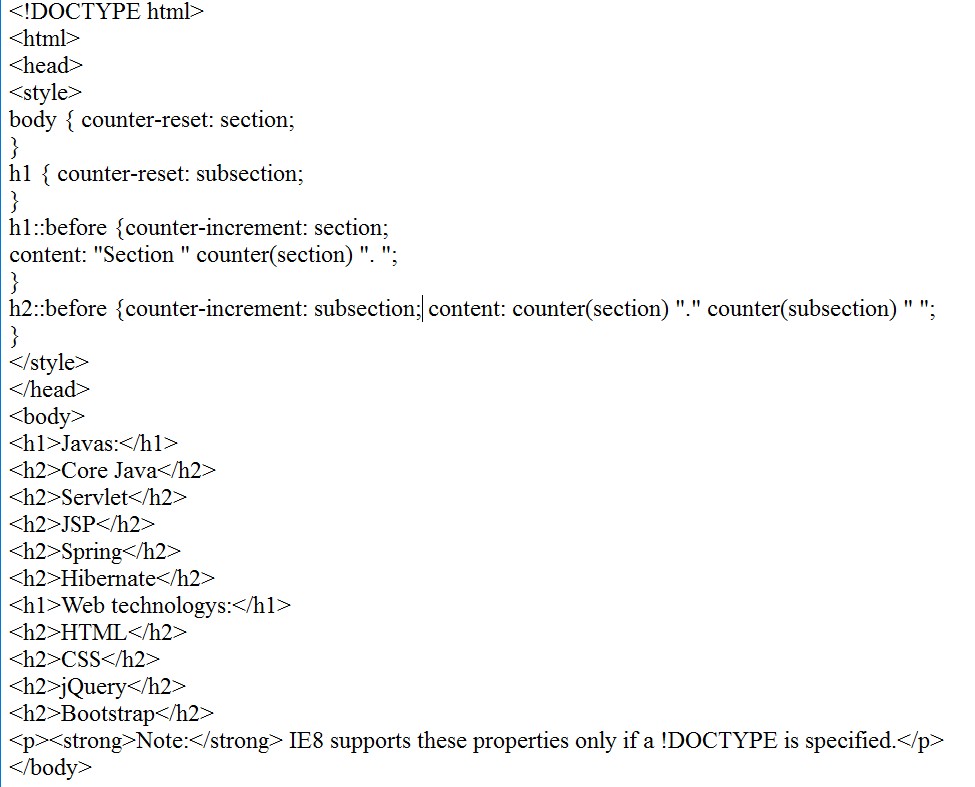
#### Example:



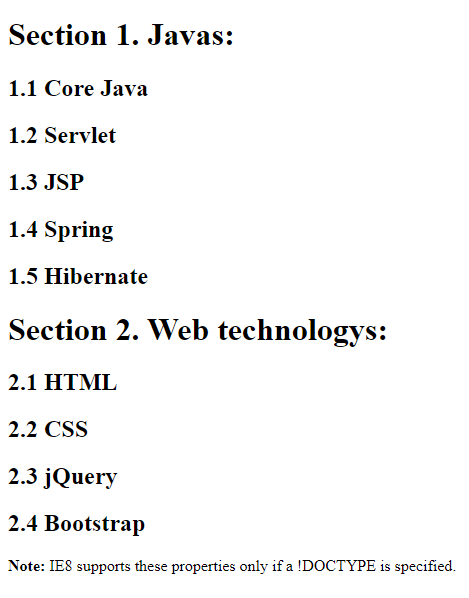
**Output**:



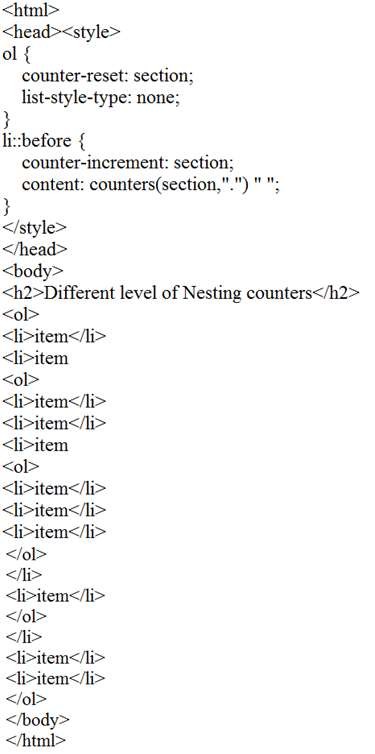
#### Example:



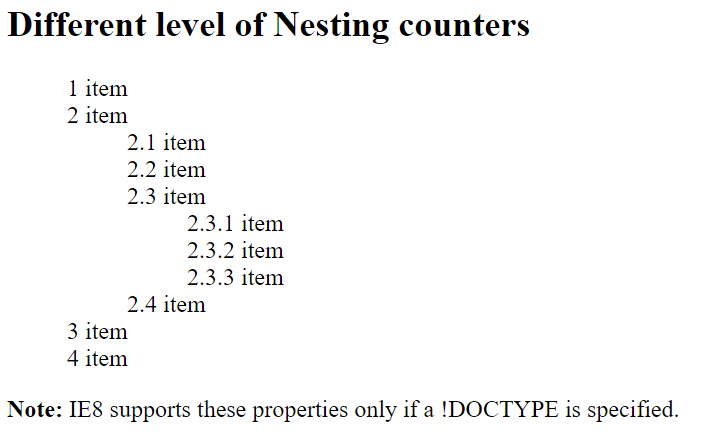
**Output**:



#### Example:



**Output:**



#### CSS Animations

**CSS animations** make it possible to animate transitions from one CSS style configuration to another. Animations consist of two components, a style describing the CSS animation and a set of keyframes that indicate the start and end states of the animation's style, as well as possible intermediate waypoints.

There are three key advantages to CSS animations over traditional script-driven animation techniques:

1. They're easy to use for simple animations; you can create them without even having to know JavaScript.
2. The animations run well, even under moderate system load. Simple animations can often perform poorly in JavaScript. The rendering engine can use frame-skipping and other techniques to keep the performance as smooth as possible.
3. Letting the browser control the animation sequence lets the browser optimize performance and efficiency by, for example, reducing the update frequency of animations running in tabs that aren't currently visible.

#### [Configuring an animation](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Animations/Using_CSS_animations#configuring_an_animation)

To create a CSS animation sequence, you style the element you want to animate with

the animation property or its sub-properties. This lets you configure the timing, duration, and other details of how the animation sequence should progress. This does **not** configure the actual

appearance of the animation, which is done using the [@keyframes](https://developer.mozilla.org/en-US/docs/Web/CSS/%40keyframes) at-rule as described in the Defining the animation sequence using keyframes section below.

The sub-properties of the animation property are:

#### animation-delay

* 1. Specifies the delay between an element loading and the start of an animation sequence and whether the animation should start immediately from its beginning or partway through the animation.

#### animation-direction

* 1. Specifies whether an animation's first iteration should be forward or backward and whether subsequent iterations should alternate direction on each run through the sequence or reset to the start point and repeat.

#### animation-duration

* 1. Specifies the length of time in which an animation completes one cycle.

#### animation-fill-mode

* 1. Specifies how an animation applies styles to its target before and after it runs.

#### animation-iteration-count

* 1. Specifies the number of times an animation should repeat.

#### animation-name

* 1. Specifies the name of the @keyframes at-rule describing an animation's keyframes.

#### animation-play-state

* 1. Specifies whether to pause or play an animation sequence.

#### animation-timing-function

* 1. Specifies how an animation transitions through keyframes by establishing acceleration curves.

#### [Defining animation sequence using keyframes](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Animations/Using_CSS_animations#defining_animation_sequence_using_keyframes)

After you've configured the animation's timing, you need to define the appearance of the animation. This is done by establishing one or more keyframes using the [@keyframes](https://developer.mozilla.org/en-US/docs/Web/CSS/%40keyframes) at-rule. Each keyframe describes how the animated element should render at a given time during the animation sequence.

Since the timing of the animation is defined in the CSS style that configures the animation, keyframes use a <percentage> to indicate the time during the animation sequence at which they take place. 0% indicates the first moment of the animation sequence, while 100% indicates the final state of the animation. Because these two times are so important, they have special

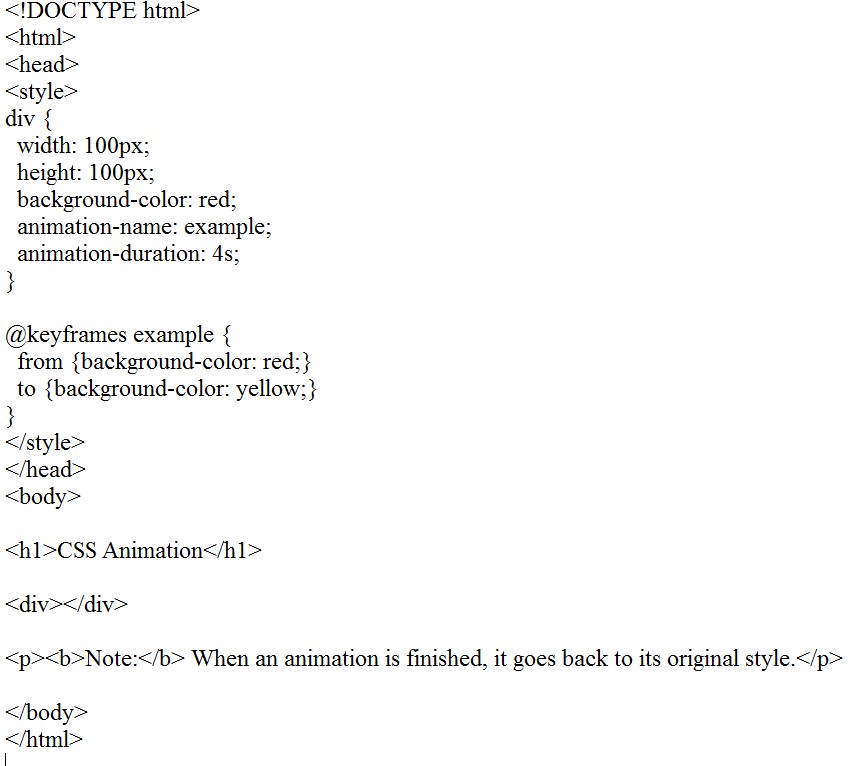
aliases: from and to. Both are optional. If from/0% or to/100% is not specified, the browser starts or finishes the animation using the computed values of all attributes.

You can optionally include additional keyframes that describe intermediate steps between the start and end of the animation.

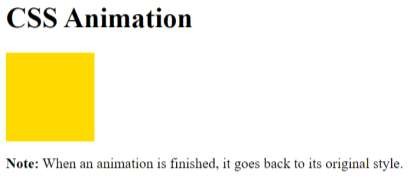
#### [Using the animation shorthand](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Animations/Using_CSS_animations#using_the_animation_shorthand)

The animation shorthand is useful for saving space. As an example, some of the rules we've been using through this article:

Example:



#### Output:



**CSS Buttons**

In HTML, we use the button tag to create a button, but by using CSS properties, we can style the buttons. Buttons help us to create user interaction and event processing. They are one of the widely used elements of web pages.

During the form submission, to view or to get some information, we generally use buttons. Let's see the basic styling in buttons.

#### Basic styling in Buttons

There are multiple properties available that are used to style the button element. Let's discuss them one by one.

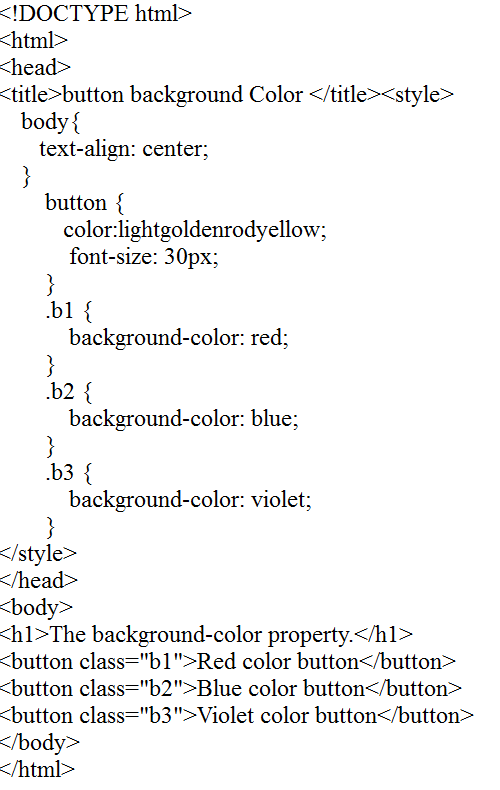
#### background-color

As we have discussed earlier, this property is used for setting the [background color](https://www.javatpoint.com/css-background-color) of the button element.

#### Syntax

1. element {
2. // background-color style 3. }

Example:



#### Output:

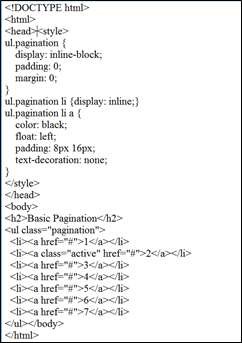


**Example:**

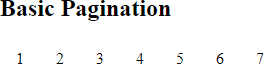
#### CSS Pagination

CSS pagination is a very useful technique for indexing different pages of a website on the homepage. If your website has lots of pages, you have to add some sort of pagination to each page.

Following are the different types of pagination:



#### Output:



**Example:**



#### Output:

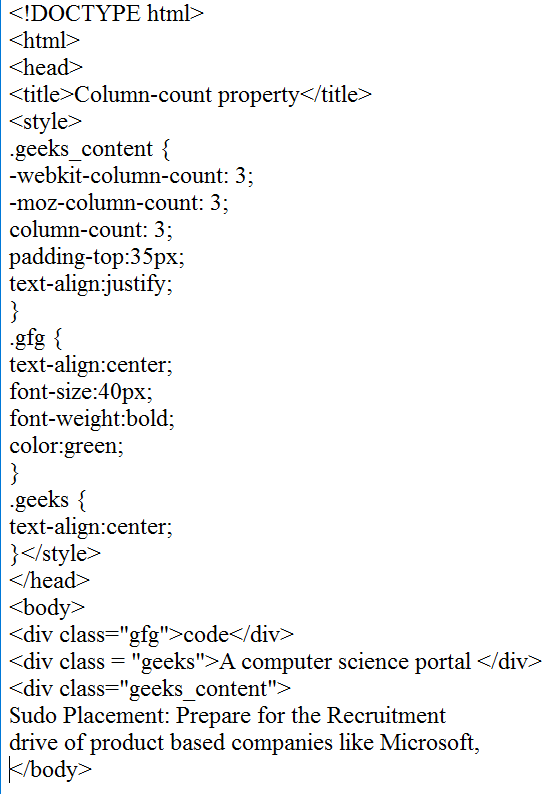


**CSS Multiple Columns**

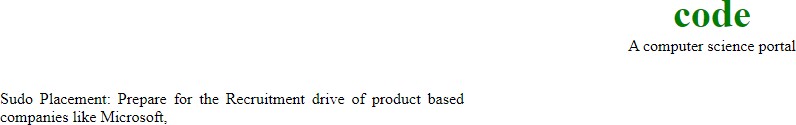
The multiple columns are used to create column layout in the web pages. There are many column property in CSS which are listed below:

* + column-count
  + column-gap
  + column-rule-style
  + column-rule-width
  + column-rule-color
  + column-rule
  + column-span
  + column-width

#### Example:



Output:



#### CSS User Interface

CSS provides many user interface features like resizing elements, outlines and box sizing. Following is a list of some common properties of CSS3 user interface:

|  |  |
| --- | --- |
| **Values** | **Description** |
| appearance | It facilitates users to make elements as user interface elements. |
| box-sizing | It facilitates users to fix elements on area in clear way. |
| icon | It is used to provide the icon on area. |
| resize | It is used to resize elements which are on area. |
| outline-offset | It is used to set space between an outline and the edge or border of an element. |
| nav-down | It is used to move down while pressing the down arrow button in keypad. |
| nav-left | It is used to move left while pressing the left arrow button in keypad. |
| nav-right | It is used to move right while pressing the right arrow button in keypad. |
| nav-up | It is used to move up while pressing the up arrow button in keypad. |

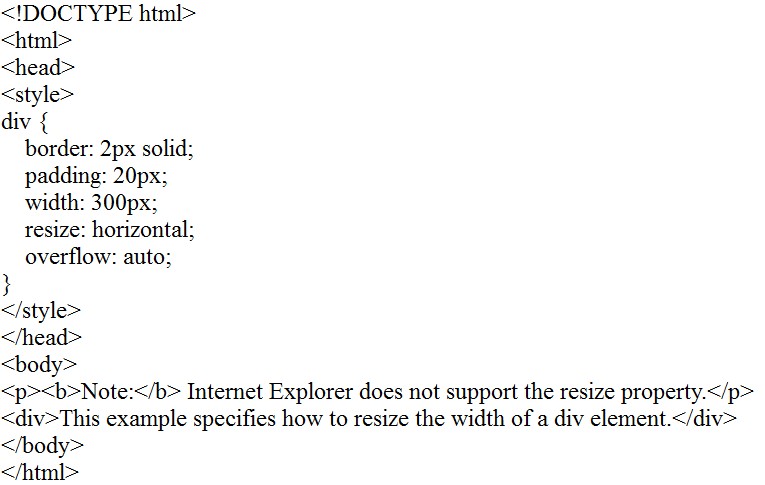
**Note:** resize and outline-offset are the most important properties of the CSS user interface. Resize property can have 3 common values:

* Horizontal resize
* Vertical resize
* Both (horizontal & vertical) resize.

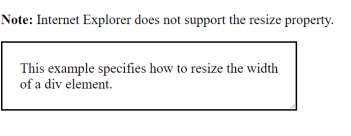
#### CSS3 Resize property

The CSS3 resize property specifies that whether an element should be resized by the user or not.

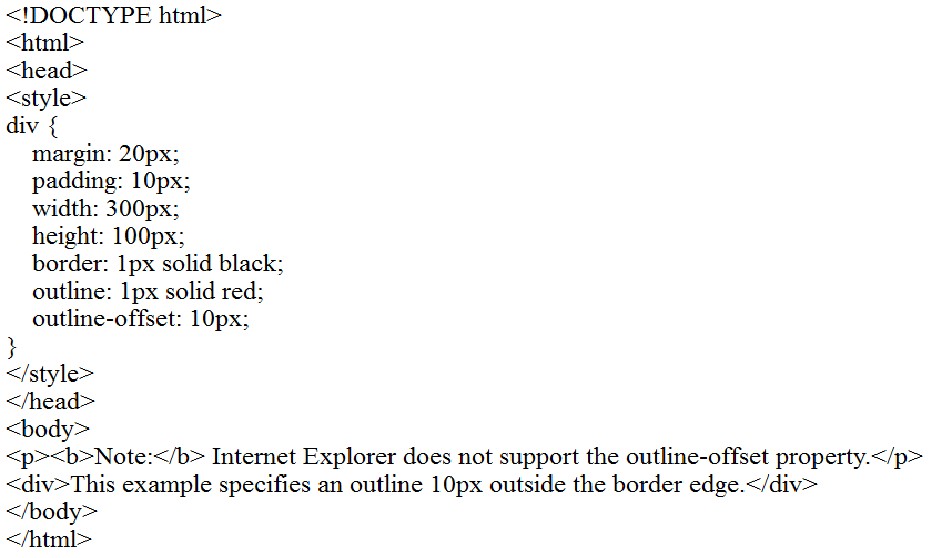
#### Example:



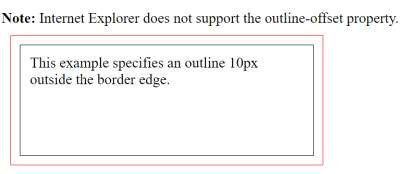
**Output:**



#### Example:



**Output:**



#### What is CSS Box sizing Property?

* The CSS box-sizing property is used to specify how to calculate the total height and width of an element.
* It controls the size of an element with a specified height and width.
* It allows you to include the padding and border within the total height and width of the element.

Before starting with CSS box-sizing, let's first understand what issue occurs if we don't use this property?

#### Without CSS box-sizing

If we don't include box-sizing property, then the box model works like below as default:

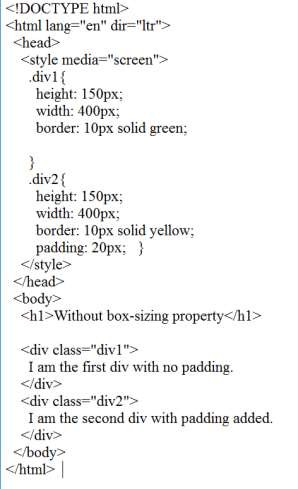
1. width + padding + border = actual visible width of an element's box
2. height + padding + border = actual visible height of an element's box

***It means if we create a box of some specific height and width and then add padding and border to it, it will look wider than the actual width.***

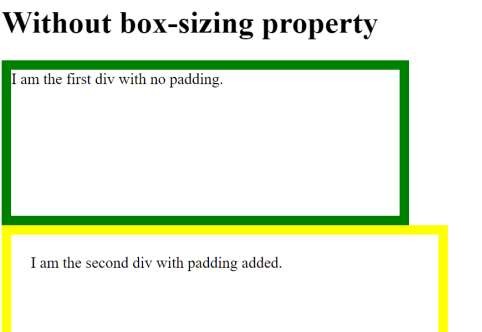
To deal with this issue, developers need to adjust the values for height and width to give space for border and padding. Let's understand it with an example:

**Example:** Let's create two div with same height and width, but with different border and padding.

#### Example:



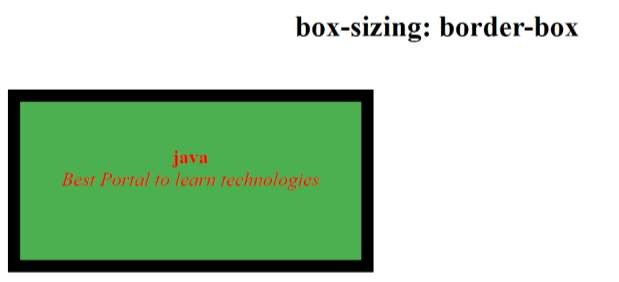
**Output:**



#### Example:



**Output:**



1. {width: 500px;
2. border: 10px solid black;}

The browser will display a box of size 500, in which 480px wide area for width will be included.

In this, the width and height are calculated by including content, border, and padding. **But it does not include the margin.**

The dimensions of the element are calculated as:

1. Width = border + padding + width of the content
2. Height = border + padding + height of the content.

#### CSS Filters Syntax

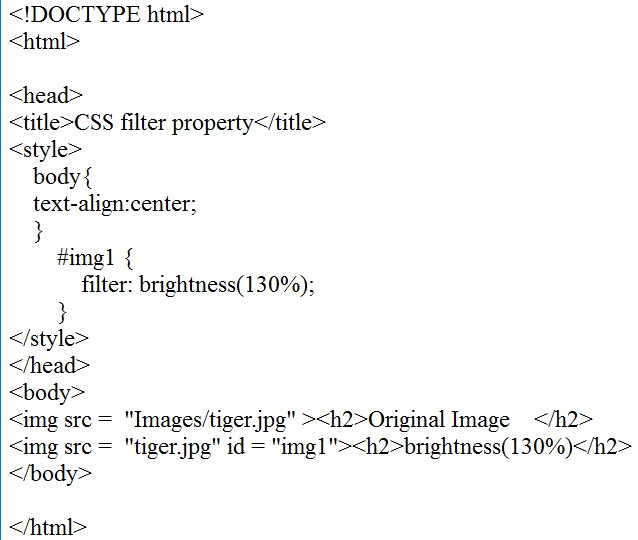
1. filter: none | invert() | drop-shadow() | brightness() | saturate() | blur() | hue- rotate() | contrast() | opacity() | grayscale() | sepia() | url();

**brightness()**

As its name implies, it is used to set the brightness of an element. If the brightness is 0%, then it represents completely black, whereas 100% brightness represents the original one. It can also accept values above 100% that provide brighter results.

We can understand it by using the following illustration.

#### Example:



**Output:**



#### invert()

It is used to invert the samples in the input image. Its 100% value represents completely inverted, and 0% values leave the unchanged input. Negative values are not allowed in it.

#### CSS Media Queries

The **Media query** in CSS is used to create a responsive web design. It means that the view of a web page differs from system to system based on screen or media types. The breakpoint specifies for what device-width size, the content is just starting to break or deform.

Media queries can be used to check many things:

* + width and height of the viewport
  + width and height of the device
  + Orientation
  + Resolution

A media query consist of a media type that can contain one or more expression which can be either true or false. The result of the query is true if the specified media matches the type of device the document is displayed on. If the media query is true then a style sheet is applied.

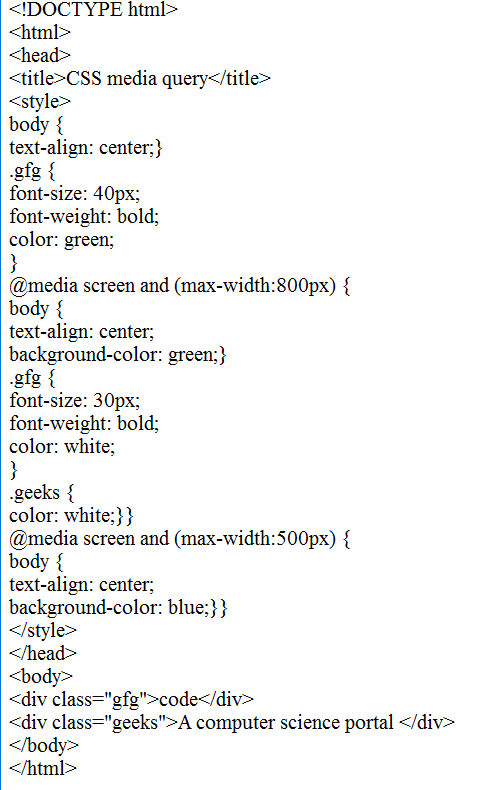
#### Syntax:

@media not | only mediatype and (expression) {

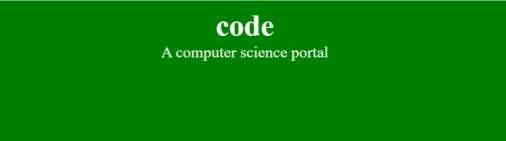
// Code content

}

**Example:**



#### Output:



**Media Types in CSS:** There are many types of media types which are listed below:

* + **all:** It is used for all media devices
  + **print:** It is used for printer.
  + **screen:** It is used for computer screens, smartphones, etc.
  + **speech:** It is used for screen readers that read the screen aloud.

**Features of Media query:** There are many features of media query which are listed below:

* + **color:** The number of bits per color component for the output device.
  + **grid:** Checks whether the device is grid or bitmap.
  + **height:** The viewport height.
  + **aspect ratio:** The ratio between width and height of the viewport.
  + **color-index:** The number of colors the device can display.
  + **max-resolution:** The maximum resolution of the device using dpi and dpcm.
  + **monochrome:** The number of bits per color on a monochrome device.
  + **scan:** The scanning of output devices.
  + **update:** How quickly can the output device modify.
  + **width:** The viewport width.

**Supported Browsers:** The browser supported by CSS *media query* are listed below:

* + Chrome 21.0 and above
  + Mozilla 3.5 and above
  + Microsoft Edge 12.0
  + Opera 9.0 and above
  + Internet Explorer 9.0 and above
  + Safari 4.0 and above

#### CSS Responsive

**CSS3 Responsive Web Design**

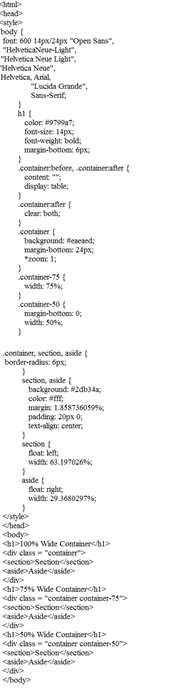
Responsive web design provides an optimal experience, easy reading and easy navigation with a minimum of resizing on different devices such as desktops, mobiles and tabs).

#### Responsive structure

Below image shows the responsive structure of web pages.

**Example**

# CSS



#### Output:



**What is The Viewport?**

The viewport is the user's visible area of a web page.

The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.

Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.

Then, when we started surfing the internet using tablets and mobile phones, fixed size web pages were too large to fit the viewport. To fix this, browsers on those devices scaled down the entire web page to fit the screen.

This was not perfect!! But a quick fix. Setting the Viewport

HTML5 introduced a method to let web designers take control over the viewport, through the

<meta> tag.

You should include the following <meta> viewport element in all your web pages:

<meta name="viewport" content="width=device-width, initial-scale=1.0">

This gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser. Here is an example of a web page without the viewport meta tag, and the same web page with the viewport meta tag:

Without the viewport meta tag

Tip: If you are browsing this page with a phone or a tablet, you can click on the two links above to see the difference.

Size Content to The Viewport

Users are used to scroll websites vertically on both desktop and mobile devices - but not horizontally!

So, if the user is forced to scroll horizontally, or zoom out, to see the whole web page it results in a poor user experience.

Some additional rules to follow:

1. Do NOT use large fixed width elements - For example, if an image is displayed at a width wider than the viewport it can cause the viewport to scroll horizontally. Remember to adjust this content to fit within the width of the viewport.
2. Do NOT let the content rely on a particular viewport width to render well - Since screen dimensions and width in CSS pixels vary widely between devices, content should not rely on a particular viewport width to render well.
3. Use CSS media queries to apply different styling for small and large screens - Setting large absolute CSS widths for page elements will cause the element to be too wide for the viewport on a smaller device. Instead, consider using relative width values, such as width: 100%. Also, be careful of using large absolute positioning values. It may cause the element to fall outside the viewport on small devices.

Building a Responsive Grid-View

Lets start building a responsive grid-view.

First ensure that all HTML elements have the box-sizing property set to border-box. This makes sure that the padding and border are included in the total width and height of the elements.

Add the following code in your CSS:

\* {

box-sizing: border-box;

}

Read more about the box-sizing property in our CSS Box Sizing chapter.

The following example shows a simple responsive web page, with two columns: 25%75%

#### Example

.menu { width: 25%; float: left;

}

.main { width: 75%; float: left;

}

The example above is fine if the web page only contains two columns.

However, we want to use a responsive grid-view with 12 columns, to have more control over the web page.

First we must calculate the percentage for one column: 100% / 12 columns = 8.33%.

Then we make one class for each of the 12 columns, class="col-" and a number defining how many columns the section should span:

#### CSS:

.col-1 {width: 8.33%;}

.col-2 {width: 16.66%;}

.col-3 {width: 25%;}

.col-4 {width: 33.33%;}

.col-5 {width: 41.66%;}

.col-6 {width: 50%;}

.col-7 {width: 58.33%;}

.col-8 {width: 66.66%;}

.col-9 {width: 75%;}

.col-10 {width: 83.33%;}

.col-11 {width: 91.66%;}

.col-12 {width: 100%;}

All these columns should be floating to the left, and have a padding of 15px:

#### CSS:

[class\*="col-"] { float: left; padding: 15px;

border: 1px solid red;

}

Each row should be wrapped in a <div>. The number of columns inside a row should always add

up to 12:

#### HTML:

<div class="row">

<div class="col-3">...</div><!-- 25% -->

<div class="col-9">...</div><!-- 75% -->

</div>

The columns inside a row are all floating to the left, and are therefore taken out of the flow of the

page, and other elements will be placed as if the columns do not exist. To prevent this, we will add a style that clears the flow:

#### CSS:

.row::after {

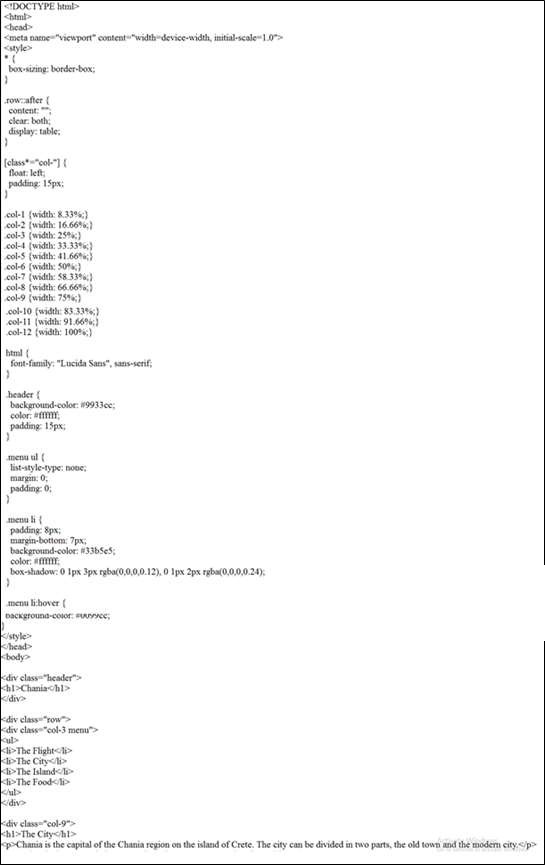
content: ""; clear: both; display: table;

}

We also want to add some styles and colors to make it look better:

**Example**

# CSS



Output:

CSS



\*\*\*\*\*\*\*\*\*\* THANK YOU…\*\*\*\*\*\*\*\*\*\*