CSS Tutorial

CSS is a language that describes the style of an HTML document.

CSS describes how HTML elements should be displayed.

.

CSS Example

body {  
  background-color: lightblue;  
}  
  
h1 {  
  color: white;  
  text-align: center;  
}  
  
p {  
  font-family: verdana;  
  font-size: 20px;  
}

What is CSS?

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

Why Use CSS?

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

CSS Solved a Big Problem

HTML was NEVER intended to contain tags for formatting a web page!

HTML was created to **describe the content** of a web page, like:

<h1>This is a heading</h1>

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

When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS removed the style formatting from the HTML page!

CSS Saves a Lot of Work!

The style definitions are normally saved in external .css files.

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

CSS Syntax and Selectors

CSS Syntax

A CSS rule-set consists of a selector and a declaration block:

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

CSS Syntax

A CSS rule-set consists of a selector and a declaration block:



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

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

Each declaration includes a C

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

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

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

Example

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

p {  
  color: red;  
  text-align: center;  
}

CSS Selectors

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

1)The element Selector

The element selector selects elements based on the element name.

Example

You can select all <p> elements on a page like this (here, all <p> elements will be center-aligned, with a red text color):

p {  
  text-align: center;  
  color: red;  
}

The id Selector

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

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

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

Example

The style rule below will be applied to the HTML element with id="para1":

#para1 {  
  text-align: center;  
  color: red;  
}

**Note:** An id name cannot start with a number!

<html>

<head>

<style>

#para1 {

text-align: center;

color: red;

}

</style>

</head>

<body>

<p id="para1">On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

</p>

<p>

ormatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore tre look of your document to the original contained in your current template.

</p>

<h1>ssssssssssssssssssss</h1>

</body>

<h2 id="para1">aaaaaaaaaaaaaaaa</h2>

</html>

The class Selector

The class selector selects elements with a specific class attribute.

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

Example

In this example all HTML elements with class="center" will be red and center-aligned:

.center {  
  text-align: center;  
  color: red;  
}

You can also specify that only specific HTML elements should be affected by a class.

Example

In this example only <p> elements with class="center" will be center-aligned:

p.center {  
  text-align: center;  
  color: red;  
}

HTML elements can also refer to more than one class.

Example

In this example the <p> element will be styled according to class="center" and to class="large":

<p class="center large">This paragraph refers to two classes.</p>

**Note:** A class name cannot start with a number!

<html>

<head>

<style>

h1.bola {

text-align: center;

color: red;

}

</style>

</head>

<body>bola

<p class="bola">On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

</p>

<p class="bola">

ormatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

</p>

<h1 class="bola">ssssssssssssssssssss</h1>

<h2>aaaaaaaaaaaaaaaa</h2>

</body>

</html>

Grouping Selectors

If you have elements with the same style definitions, like this:

h1 {  
  text-align: center;  
  color: red;  
}  
  
h2 {  
  text-align: center;  
  color: red;  
}  
  
p {  
  text-align: center;  
  color: red;  
}

It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

Example

In this example we have grouped the selectors from the code above:

h1, h2, p  {  
  text-align: center;  
  color: red;  
}

<html>

<head>

<style>

h1,h2,p {

text-align: center;

color: red;

}

</style>

</head>

<body>bola

<p class="bola">On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

</p>

<p class="bola">

ormatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

</p>

<h1 class="bola">ssssssssssssssssssss</h1>

<h2>aaaaaaaaaaaaaaaa</h2>

</body>

</html>

CSS Comments

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

Comments are ignored by browsers.

Example

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

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

3ways to use css in html pages

* External style sheet
* Internal style sheet
* Inline style

1. External style sheet

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

Each page must include a reference to the external style sheet file inside the <link> element.

Example

External styles are defined within the <link> element, inside the <head> section of an HTML page:

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

mystyle.css

h1{

background-color:red;

color:blue;

}

h2{

background-color:red;

color:blue;

}

p{

background-color:red;

c<,olor:blue;

}

#p1

{

background-color:#111ddd;

color:blue;

}

h3.center

{

background-color:rgb(40,12,50);

color:blue;

}

Ex1.html

<html>

<head>

<link rel="stylesheet" type="text/css" href="Ex1.css">

</head>

<body>

<h1>aaaaaaaaaaaaaaaa</h1>

<h2>aaaaaaaaaaa</h2>

<p id="#p1">

aaaaaaaaaaaaaaaaaaaaaaaaaaa</p>

</body>

</html>

## **2)Internal Style Sheet**

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

### **Example**

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

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

## **Inline Styles**

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

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

### **Example**

Inline styles are defined within the "style" attribute of the relevant element:

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

CSS Properties: Text Properties: • color: blue; • text-align: center/left/rigth/justify; • text-decoration: overline/line-through/underline; • text-transform: uppercase/lowercase/capitalize; • text-indent: 10px/-10px; • letter-spacing: 5px/-5px; • line-height: 0.8; • direction: rtl; • word-spacing: 10px; • text-shadow: 3px 2px red; Example: html>

# CSS Properties:

**Text Properties:**

### color: blue;

* text-align: center/left/rigth/justify;
* text-decoration: overline/line-through/underline;
* text-transform: uppercase/lowercase/capitalize;
* text-indent: 10px/-10px;
* letter-spacing: 5px/-5px;
* line-height: 0.8;
* direction: rtl;
* word-spacing: 10px;
* text-shadow: 3px 2px red;

**Example:**

html>

<head>

<style type="text/css">

p.color {

color: blue;

}

p.align {

text-align: right;

}

p.decoration {

text-decoration: line-through;

}

p.indent {

text-indent: 50px;

}

p.lspace {

letter-spacing: 5px;

}

p.shadow {

text-shadow: 3px 2px red;

}

p.height {

line-height: 5.5;

}

p.uppercase {

text-transform: uppercase;

}

p.lowercase {

text-transform: lowercase;

}

p.capitalize {

text-transform: capitalize;

}

</style>

</head>

<body>

<p class="color">This is some text in blue color.</p>

<p class="align">This is some text aligned right.</p>

<p class="decoration">This is some text line through</p>

<p class="indent">This is some text with 50px indent</p>

<p class="lspace">This is some text with line spacing</p>

<p class="shadow">This is some shadow text</p>

<p class="height">This is some text with line height</p>

<p class="uppercase">This is some uppercase text.</p>

<p class="lowercase">This is some lowercase text.</p>

<p class="capitalize">This is some capitalized text.</p>

</body>

</html>

**Font Properties:**

### font-family: arial;

1. font-style: normal/italic/oblique;
2. font-size: 30px;
3. font-weight: normal/bold;
4. font-variant: normal/small-caps;

**Example:**

<!DOCTYPE html>

<html>

<head>

<style> p.fmly {

font-family: arial;

}

p.fmly1 {

font-family: symbol;

}

* 1. ormal {

font-style: normal;

}

p.italic {

font-style: italic;

}

* 1. blique {

font-style: oblique;

}

h1 {

font-size: 40px;

}

h2 {

font-size: 30px;

}

p {

font-size: 14px;

}

p.normal {

font-weight: normal;

}

p.light {

font-weight: lighter;

}

p.thick {

font-weight: bold;

}

p.thicker {

font-weight: 900;

}

p.normal {

font-variant: normal;

}

* 1. mall {

font-variant: small-caps;

}

</style>

</head>

<body>

<h1>CSS font-family</h1>

<p class="fmly">This is a paragraph</p>

<p class="fmly1">This is a paragraph</p>

<p class="normal">This is a paragraph in normal style.</p>

<p class="italic">This is a paragraph in italic style.</p>

<p class="oblique">This is a paragraph in oblique style.</p>

<h1>This is heading 1</h1>

<h2>This is heading 2</h2>

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

<p>This is another paragraph.</p>

<p class="normal">This is a paragraph.</p>

<p class="light">This is a paragraph.</p>

<p class="thick">This is a paragraph.</p>

<p class="thicker">This is a paragraph.</p>

<p class="normal">this is normal text</p>

<p class="small">Capitalised Small text.</p>

</body>

</html>

**Border Properties:**

1. border-style: dotted/dashed/solid/double/groove/ridge/inset/outset/none/hidden;
2. border-width: 5px;

border-width: 2px 10px 4px 20px;

1. border-color: red;

Border-color: red green blue yellow;

**Note:** four values indicates - first for top border, second for right border, third for bottom border, fourth for left border.

**Example:**

<!DOCTYPE html>

<html>

<head>

<style>

p.one {

border-style: solid; border-color: red;

margin: 10px 100px 10px 10px;

}

p.two {

border-style: solid; border-color: green;

}

* 1. hree {

border-style: solid;

border-color: red green blue yellow;

}

</style>

</head>

<body>

<h2>The border-color Property</h2>

<p>This property specifies the color of the four borders:</p>

<p class="one">A solid red border</p>

<p class="two">A solid green border</p>

<p class="three">A solid multicolor border</p>

</body>

</html>

**Background Properties:**

* + background-color: red;
  + background-image: url("image\_path");
  + background-repeat: repeat-x/repeat-y/round/no-repeat;
  + background-attachment: fixed/scroll/local;
  + background-size: 20px 30px;
  + background-position: right top/right bottom/left top/left bottom;

or

30px 20px;

**Margin Properties:**

* + margin-top: 100px;
  + margin-bottom: 100px;
  + margin-right: 150px;
  + margin-left: 80px;

**Link Properties:**

* + a:link - unvisited link
  + a:hover – when mouse is on it
  + a:active – when mouse click on it
  + a:visited – after mouse clicked

**Example:**

<!DOCTYPE html>

<html>

<head>

<style>

/\* unvisited link \*/ a:link {

color: red;

}

/\* mouse over link \*/ a:hover {

color: green;

text-decoration:none;

}

/\* selected link \*/ a:active {

color: blue;

}

/\* visited link \*/ a:visited {

color: brown;

}

</style>

</head>

<body>

<a href="text\_css.html" target="\_blank">This is a link</a>

</body>

</html>

# CSS Box Model

## **The CSS Box Model**

In CSS, the term "box model" is used when talking about design and layout.

Explanation of the different parts:



The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

Explanation of the different parts:

* **Content** - The content of the box, where text and images appear
* **Padding** - Clears an area around the content. The padding is transparent
* **Border** - A border that goes around the padding and content
* **Margin** - Clears an area outside the border. The margin is transparent

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

### **Example**

Demonstration of the box model:

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

## **Width and Height of an Element**

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

**Important:** When you set the width and height properties of an element with CSS, you just set the width and height of the **content area**. To calculate the full size of an element, you must also add padding, borders and margins.

CSS Padding

Padding is used to create space around an element's content, inside of any defined borders.

This element has a padding of 70px.

## **CSS Padding**

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

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

## **Padding - Individual Sides**

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

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

All the padding properties can have the following values:

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

**Note:** Negative values are not allowed.

### **Example**

Set different padding for all four sides of a <div> element:

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

## **Padding - Shorthand Property**

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

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

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

So, here is how it works:

If the padding property has four values:

* **padding: 25px 50px 75px 100px;**
  + top padding is 25px
  + right padding is 50px
  + bottom padding is 75px
  + left padding is 100px

### **Example**

Use the padding shorthand property with four values:

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

If the padding property has three values:

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

### **Example**

Use the padding shorthand property with three values:

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

If the padding property has two values:

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

### **Example**

Use the padding shorthand property with two values:

div {  
  padding: 25px 50px;  
}

If the padding property has one value:

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

### **Example**

Use the padding shorthand property with one value:

div {  
  padding: 25px;  
}

## **Padding and Element Width**

The CSS width property specifies the width of the element's content area. The content area is the portion inside the padding, border, and margin of an element ([the box model](https://www.w3schools.com/css/css_boxmodel.asp)).

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

### **Example**

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

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

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

### **Example**

Use the box-sizing property to keep the width at 300px, no matter the amount of padding:

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

## **CSS height and width Examples**

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

### **Example**

Set the height and width of a <div> element:

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

## **CSS Margins**

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

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

## **Margin - Individual Sides**

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

* margin-top
* margin-right
* margin-bottom
* margin-left

div {

border: 1px solid black;

margin-top: 100px;

margin-bottom: 100px;

margin-right: 150px;

margin-left: 80px;

background-color: lightblue;

}

### **Example**

This <div> element will have a total width of 350px:

div {  
  width: 320px;  
  padding: 10px;  
  border: 5px solid gray;  
  margin: 0;  
}

Here is the calculation:

320px (width)  
+ 20px (left + right padding)  
+ 10px (left + right border)  
+ 0px (left + right margin)  
**= 350px**

The total width of an element should be calculated like this:

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

The total height of an element should be calculated like this:

Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

# CSS Layout - display: inline-block

## **The display: inline-block Value**

Compared to display: inline, the major difference is that display: inline-block allows to set a width and height on the element.

Also, with display: inline-block, the top and bottom margins/paddings are respected, but with display: inline they are not.

Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements.

The following example shows the different behavior of display: inline, display: inline-block and display: block:

### **Example**

## <!DOCTYPE html>

## <html>

## <head>

## <style>

## span.a {

## display: inline; /\* the default for span \*/

## width: 100px;

## height: 100px;

## padding: 5px;

## border: 1px solid blue;

## background-color: yellow;

## }

## span.b {

## display: inline-block;

## width: 100px;

## height: 100px;

## padding: 5px;

## border: 1px solid blue;

## background-color: yellow;

## }

## span.c {

## display: block;

## width: 100px;

## height: 100px;

## padding: 5px;

## border: 1px solid blue;

## background-color: yellow;

## }

## </style>

## </head>

## <body>

## <h1>The display Property</h1>

## <h2>display: inline</h2>

## <div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span class="a">Aliquam</span> <span class="a">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>

## <h2>display: inline-block</h2>

## <div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span class="b">Aliquam</span> <span class="b">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>

## <h2>display: block</h2>

## <div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. <span class="c">Aliquam</span> <span class="c">venenatis</span> gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>

## </body>

## </html>

## **Using inline-block to Create Navigation Links**

One common use for display: inline-block is to display list items horizontally instead of vertically. The following example creates horizontal navigation links:

### **Example**

<!DOCTYPE html>

<html>

<head>

<style>

.nav {

background-color: yellow;

list-style-type: none;

text-align: center;

margin: 0;

padding: 0;

}

.nav li {

display: inline-block;

font-size: 20px;

padding: 20px;

}

</style>

</head>

<body>

<h1>Horizontal Navigation Links</h1>

<p>By default, list items are displayed vertically. In this example we use display: inline-block to display them horizontally (side by side).</p>

<p>Note: If you resize the browser window, the links will automatically break when it becomes too crowded.</p>

<ul class="nav">

<li><a href="#home">Home</a></li>

<li><a href="#about">About Us</a></li>

<li><a href="#clients">Our Clients</a></li>

<li><a href="#contact">Contact Us</a></li>

</ul>

</body>

</html>

CSS ADVANCE

## 1) **CSS Rounded Corners**

With the CSS border-radius property, you can give any element "rounded corners".

## **CSS border-radius Property**

The CSS border-radius property defines the radius of an element's corners.

**Tip:** This property allows you to add rounded corners to elements!

Here are three examples:

1. Rounded corners for an element with a specified background color:

Rounded corners!

2. Rounded corners for an element with a border:

Rounded corners!

3. Rounded corners for an element with a background image:

Rounded corners!

Here is the code:

<!DOCTYPE html>

<html>

<head>

<style>

#rcorners1 {

border-radius: 25px;

background: #73AD21;

padding: 20px;

width: 400px;

height: 150px;

}

#rcorners2 {

border-radius: 25px;

border: 2px solid #73AD21;

padding: 20px;

width: 200px;

height: 150px;

}

#rcorners3 {

border-radius: 25px;

background: url(paper.gif);

background-position: left top;

background-repeat: repeat;

padding: 20px;

width: 200px;

height: 150px;

}

</style>

</head>

<body>

<h1>The border-radius Property</h1>

<p>Rounded corners for an element with a specified background color:</p>

<p id="rcorners1">Rounded corners!</p>

<p>Rounded corners for an element with a border:</p>

<p id="rcorners2">Rounded corners!</p>

<p>Rounded corners for an element with a background image:</p>

<p id="rcorners3">Rounded corners!</p>

</body>

</html>

## **CSS border-radius - Specify Each Corner**

The border-radius property can have from one to four values. Here are the rules:

**Four values - border-radius: 15px 50px 30px 5px;** (first value applies to top-left corner, second value applies to top-right corner, third value applies to bottom-right corner, and fourth value applies to bottom-left corner):

**Three values - border-radius: 15px 50px 30px;** (first value applies to top-left corner, second value applies to top-right and bottom-left corners, and third value applies to bottom-right corner):

**Two values - border-radius: 15px 50px;** (first value applies to top-left and bottom-right corners, and the second value applies to top-right and bottom-left corners):

**One value - border-radius: 15px;** (the value applies to all four corners, which are rounded equally:

You could also create elliptical corners:

border-radius: 50px / 15px;

border-radius: 15px / 50px;

border-radius: 50%;

# 2) CSS Border Images

## **CSS Border Images**

With the CSS border-image property, you can set an image to be used as the border around an element.

## **CSS border-image Property**

The CSS border-image property allows you to specify an image to be used instead of the normal border around an element.

The property has three parts:

1. The image to use as the border
2. Where to slice the image
3. Define whether the middle sections should be repeated or stretched

We will use the following image (called "border.png"):



The border-image property takes the image and slices it into nine sections, like a tic-tac-toe board. It then places the corners at the corners, and the middle sections are repeated or stretched as you specify.

**Note:** For border-image to work, the element also needs the border property set!

Here, the middle sections of the image are repeated to create the border:

An image as a border!

Here is the code:

<!DOCTYPE html>

<html>

<head>

<style>

#borderimg {

border: 10px solid transparent;

padding: 15px;

border-image: url(border.png) 30 round;

}

</style>

</head>

<body>

<h1>The border-image Property</h1>

<p>Here, the middle sections of the image are repeated to create the border:</p>

<p id="borderimg">border-image: url(border.png) 30 round;</p>

<p>Here is the original image:</p><img src="border.png">

<p><strong>Note:</strong> Internet Explorer 10, and earlier versions, do not support the border-image property.</p>

</body>

</html>

# 3) CSS Colors

## **GBA Colors**

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

An RGBA color value is specified with: rgba(red, green, blue, alpha). The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

rgba(255, 0, 0, 0.2);

rgba(255, 0, 0, 0.4);

rgba(255, 0, 0, 0.6);

rgba(255, 0, 0, 0.8);

The following example defines different RGBA colors:

<!DOCTYPE html>

<html>

<head>

<style>

#p1 {background-color:rgba(255,0,0,0.3);}

#p2 {background-color:rgba(0,255,0,0.3);}

#p3 {background-color:rgba(0,0,255,0.3);}

#p4 {background-color:rgba(192,192,192,0.3);}

#p5 {background-color:rgba(255,255,0,0.3);}

#p6 {background-color:rgba(255,0,255,0.3);}

</style>

</head>

<body>

<h1>Define Colors With RGBA Values</h1>

<p id="p1">Red</p>

<p id="p2">Green</p>

<p id="p3">Blue</p>

<p id="p4">Grey</p>

<p id="p5">Yellow</p>

<p id="p6">Cerise</p>

</body>

</html>

# 4) CSS Gradients

SS gradients let you display smooth transitions between two or more specified colors.

CSS defines three types of gradients:

* **Linear Gradients (goes down/up/left/right/diagonally)**
* **Radial Gradients (defined by their center)**
* **Conic Gradients (rotated around a center point)**

## **CSS Linear Gradients**

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

### **Syntax**

background-image: linear-gradient(direction, color-stop1, color-stop2, ...);

**Direction - Top to Bottom (this is default)**

The following example shows a linear gradient that starts at the top. It starts red, transitioning to yellow:

top to bottom (default)

### **Example**

#grad {  
  background-image: linear-gradient(red, yellow);  
}

Example:

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 200px;

background-color: red; /\* For browsers that do not support gradients \*/

background-image: linear-gradient(red, yellow);

}

</style>

</head>

<body>

<h1>Linear Gradient - Top to Bottom</h1>

<p>This linear gradient starts red at the top, transitioning to yellow at the bottom:</p>

<div id="grad1"></div>

</body>

</html>

#grad {  
  background-image: linear-gradient(to right, red , yellow);  
}

#grad {  
  background-image: linear-gradient(to right, red , yellow);  
}

#grad {  
  background-image: linear-gradient(to bottom right, red, yellow);  
}

#grad {  
  background-image: linear-gradient(180deg, red, yellow);  
}

**Radial Gradient**

#grad {  
  background-image: radial-gradient(red, yellow, green);  
}

#grad {  
  background-image: radial-gradient(red 5%, yellow 15%, green 60%);  
}

#grad {  
  background-image: radial-gradient(circle, red, yellow, green);  
}

## **CSS Conic Gradients**

A conic gradient is a gradient with color transitions rotated around a center point.

To create a conic gradient you must define at least two colors.

### **Syntax**

background-image: conic-gradient([from angle] [at position,] color [degree], color [degree], ...);

By default, angle is 0deg and position is center.

<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

height: 200px;

width: 200px;

background-color: red; /\* For browsers that do not support gradients \*/

background-image: conic-gradient(red, yellow, green);

}

</style>

</head>

<body>

<h1>Conic Gradient - Three Colors</h1>

<div id="grad1"></div>

</body>

</html>

A conic gradient with five colors:

#grad {  
  background-image: conic-gradient(red, yellow, green, blue, black);  
}

# 5) CSS Shadow Effects

## **CSS Shadow Effects**

With CSS you can add shadow to text and to elements.

In these chapters you will learn about the following properties:

* text-shadow
* box-shadow

## **CSS Text Shadow**

The CSS text-shadow property applies shadow to text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

## **Text shadow effect!**

<!DOCTYPE html>

<html>

<head>

<style>

h1 {

text-shadow: 2px 2px;

}

</style>

</head>

<body>

<h1>Text-shadow effect!</h1>

</body>

</html>

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

h1 {  
  text-shadow: 2px 2px 5px red;  
}

h1 {  
  color: white;  
  text-shadow: 2px 2px 4px #000000;  
}

h1 {  
  text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;  
}

# 6) CSS 2D Transforms

## **CSS 2D Transforms**

CSS transforms allow you to move, rotate, scale, and skew elements.

Mouse over the element below to see a 2D transformation:

**2D rotate**

In this chapter you will learn about the following CSS property:

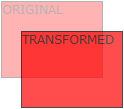
* transform

## **CSS 2D Transforms Methods**

With the CSS transform property you can use the following 2D transformation methods:

* translate()
* rotate()
* scaleX()
* scaleY()
* scale()
* skewX()
* skewY()
* skew()
* matrix()

## **The translate() Method**



The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

The following example moves the <div> element 50 pixels to the right, and 100 pixels down from its current position:

### **Example**

div {  
  transform: translate(50px, 100px);  
}

## **The rotate() Method**



The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

The following example rotates the <div> element clockwise with 20 degrees:

### **Example**

div {  
  transform: rotate(20deg);  
}

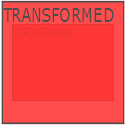
Using negative values will rotate the element counter-clockwise.

The following example rotates the <div> element counter-clockwise with 20 degrees:

### **Example**

div {  
  transform: rotate(-20deg);  
}

## **The scale() Method**



The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).

The following example increases the <div> element to be two times of its original width, and three times of its original height:

### **Example**

div {  
  transform: scale(2, 3);  
}

The following example decreases the <div> element to be half of its original width and height:

### **Example**

div {  
  transform: scale(0.5, 0.5);  
}

## **The scaleX() Method**

The scaleX() method increases or decreases the width of an element.

The following example increases the <div> element to be two times of its original width:

### **Example**

div {  
  transform: scaleX(2);  
}

The following example decreases the <div> element to be half of its original width:

### **Example**

div {  
  transform: scaleX(0.5);  
}

## **The scaleY() Method**

The scaleY() method increases or decreases the height of an element.

The following example increases the <div> element to be three times of its original height:

### **Example**

div {  
  transform: scaleY(3);  
}

The following example decreases the <div> element to be half of its original height:

### **Example**

div {  
  transform: scaleY(0.5);  
}

## **The skewX() Method**

The skewX() method skews an element along the X-axis by the given angle.

The following example skews the <div> element 20 degrees along the X-axis:

### **Example**

div {  
  transform: skewX(20deg);  
}

## **The skewY() Method**

The skewY() method skews an element along the Y-axis by the given angle.

The following example skews the <div> element 20 degrees along the Y-axis:

### **Example**

div {  
  transform: skewY(20deg);  
}

## **The skew() Method**

The skew() method skews an element along the X and Y-axis by the given angles.

The following example skews the <div> element 20 degrees along the X-axis, and 10 degrees along the Y-axis:

### **Example**

div {  
  transform: skew(20deg, 10deg);  
}

If the second parameter is not specified, it has a zero value. So, the following example skews the <div> element 20 degrees along the X-axis:

### **Example**

div {  
  transform: skew(20deg);  
}

## **The matrix() Method**



The matrix() method combines all the 2D transform methods into one.

The matrix() method take six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements.

The parameters are as follow: matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

### **Example**

div {  
  transform: matrix(1, -0.3, 0, 1, 0, 0);  
}

# 7) CSS 3D Transforms

## **CSS 3D Transforms**

CSS also supports 3D transformations.

Mouse over the elements below to see the difference between a 2D and a 3D transformation:

## **CSS 3D Transforms Methods**

With the CSS transform property you can use the following 3D transformation methods:

* rotateX()
* rotateY()
* rotateZ()

## **The rotateX() Method**

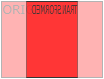


The rotateX() method rotates an element around its X-axis at a given degree:

### **Example**

#myDiv {  
  transform: rotateX(150deg);  
}

## **The rotateY() Method**



The rotateY() method rotates an element around its Y-axis at a given degree:

### **Example**

#myDiv {  
  transform: rotateY(150deg);  
}

## **The rotateZ() Method**

The rotateZ() method rotates an element around its Z-axis at a given degree:

### **Example**

#myDiv {  
  transform: rotateZ(90deg);  
}

## 8) **CSS Transitions**

CSS transitions allows you to change property values smoothly, over a given duration.

**Mouse over the element below to see a CSS transition effect:**

**CSS**

In this chapter you will learn about the following properties:

* transition
* transition-delay
* transition-duration
* transition-property
* transition-timing-function

## **How to Use CSS Transitions?**

To create a transition effect, you must specify two things:

* the CSS property you want to add an effect to
* the duration of the effect

**Note:** If the duration part is not specified, the transition will have no effect, because the default value is 0.

The following example shows a 100px \* 100px red <div> element. The <div> element has also specified a transition effect for the width property, with a duration of 2 seconds:

### **Example**

div {  
  width: 100px;  
  height: 100px;  
  background: red;  
  transition: width 2s;  
}

The transition effect will start when the specified CSS property (width) changes value.

Now, let us specify a new value for the width property when a user mouses over the <div> element:

### **Example**

div:hover {  
  width: 300px;  
}

div {  
  transition: width 2s, height 4s;  
}

div {  
  transition-delay: 1s;  
}

div {  
  transition: width 2s, height 2s, transform 2s;  
}

## **Specify the Speed Curve of the Transition**

The transition-timing-function property specifies the speed curve of the transition effect.

The transition-timing-function property can have the following values:

* ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
* linear - specifies a transition effect with the same speed from start to end
* ease-in - specifies a transition effect with a slow start
* ease-out - specifies a transition effect with a slow end
* ease-in-out - specifies a transition effect with a slow start and end
* cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

The following example shows some of the different speed curves that can be used:

### **Example**

#div1 {transition-timing-function: linear;}  
#div2 {transition-timing-function: ease;}  
#div3 {transition-timing-function: ease-in;}  
#div4 {transition-timing-function: ease-out;}  
#div5 {transition-timing-function: ease-in-out;}

## **More Transition Examples**

The CSS transition properties can be specified one by one, like this:

### **Example**

div {  
  transition-property: width;  
  transition-duration: 2s;  
  transition-timing-function: linear;  
  transition-delay: 1s;  
}