# **CSS**

Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages.

CSS is easy to learn and understood but it provides powerful control over the presentation of an HTML document.

**WHY CSS?**

* **CSS saves time :**You can write CSS once and reuse same sheet in multiple HTML pages.
* **Easy Maintainance :**To make a global change simply change the style, and all elements in all the WebPages will be updated automatically.
* **Search Engines :**CSS is considered as clean coding technique, which means search engines won’t have to struggle to “read” its content.
* **Superior styles to HTML :**CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
* **Offline Browsing :**CSS can store web applications locally with the help of offline cache. Using of this we can view offline websites.

**CSS Syntax**

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document.

A style rule set consists of a selector and declaration block.

## Inline CSS

An inline CSS is used to apply a unique style to a single HTML tag/element.

An inline CSS uses the style attribute of an HTML element.

The following example sets the text color of the <h1> element to blue, and the text color of the <p> element to red:

<h1 style="color:blue;">A Blue Heading</h1>  
  
<p style="color:red;">A red paragraph.</p>

## Internal/Ebedded CSS

An internal CSS is used to define a style for a single HTML page.

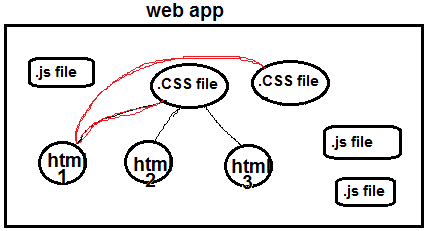
An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

The following example sets the text color of ALL the <h1> elements (on that page) to blue, and the text color of ALL the <p> elements to red. In addition, the page will be displayed with a "powderblue" background color:

<!DOCTYPE html>  
<html>  
<head>  
<style>  
body {background-color: powderblue;}  
h1   {color: blue;}  
p    {color: red;}  
</style>  
</head>  
<body>  
  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>  
  
</body>  
</html>

**External Style Sheets**

* The external CSS are defining in separate “.css” file and html tags are defined in the “.html” file.
* An external style sheet is used to define the style for many HTML pages.
* We can link (connect) the css file & html file by using <link> tag.
* It is reusable



Syn: <link href=”filename.css” rel=”stylesheet”>

**Note:**

**We can write comments in css (internal & external) using /\* \*/**

**CSS selectors**

* “Selector” is a syntax to select, it is used to select the desired elements in the webpage.
* When we use a selector, the browser searches the entire webpage for the matching elements and returns the matching elements; and we apply styles only for those matching elements.
* First we have to select the element/elements, and then only we can apply some styles to it.
* Selectors we can in internal & external css only.

**list of selectors:**

1. **Universal selector**
2. **Tag selector**
3. **ID selector**
4. **Class selector**
5. **Compound selector**
6. **Grouping selector**
7. **Child selector**
8. **Direct Child selector**
9. **Attribute selector**
10. **Hover selector**
11. **Focus selector**

**Universal selector**

* It selects all the tags in the webpages, include html, body, head etc…
* Used to define common properties for all tags (global styles).

**Syn: \*{**

**property:value;**

**…**

**}**

**Tag selector**

* It selects all the instances of the specified tag.

**Syn: tag-name{**

**property:value;**

**…**

**}**

**ID selector**

* It selects all the instances of the specified tag, means it used to specify common attributes of multiple tags.
* Id is “identification name”
* Id should be unique in the web page.
* # is symbol of ID selector.

**Syn: #ID{**

**property:value;**

**…**

**}**

**Class selector**

* It selects one or more elements, based on the class name, means it used to specify common attributes of multiple tags.
* We use same class for similar elements/tags.
* “.” is symbol of Class selector.

**Syn: .class{**

**property:value;**

**…**

**}**

**Compound selector**

* It selects the instances of specific tag, which have specified class name.
* Its combination of “tag” selector and “class” selector.

**Syn: tagname.class{**

**property:value;**

**…**

**}**

**Grouping selector**

* It selects the specifiedgroup of tags/elements, means to set common properties for different tags.
* “,” is the symbol of grouping selector.

**Syn: tag1, tag2, tag3, …{**

**property:value;**

**…**

**}**

**Child selector**

* It selects all the child tags/elements (including grandchild) of the specified parent tag,
* “space” is the symbol of child selector.

**Syn: p-tag ch-tag {**

**property:value;**

**…**

**}**

**Direct Child selector**

* It selects only the direct child tags/elements (excluding the grandchild) of the specified parent tag,
* “**>**” is the symbol of direct child selector.

**Syn: p-tag>ch-tag {**

**property:value;**

**…**

**}**

**Attribute selector**

* It selects all the tags/elementsthat are having specified attribute,
* “[ ]” is the symbol of attribute selector.

**Syn: tag [attribute=”value”]{**

**property:value;**

**…**

**}**

**Hover selector**

* It applies the style only when the user places the mouse pointer on the element, at run time.
* It automatically removes the style, if mouse pointer is comes out of element (now id displaying with original settings).
* It’s also called as “pseudo class”.
* “**:**” is the symbol of hover selector.

**Syn: tag:hover{**

**property:value;**

**…**

**}**

**focusselector**

* It applies the style only when the focus (cursor) is comes on to the element.
* It automatically removes the style, if mouse pointer is comes out of element (now id displaying with original settings).
* It’s also called as “pseudo class”.
* “**:**” is the symbol of hover selector.

**Syn: tag:focus{**

**property:value;**

**…**

**}**

**Note: it is applicable only for which control/element allows cursor.**

**CSS Precedence**

* **Css styles are applied in the following order (lower priority to higher priority).**
* **The higher priority style overrides the same property’s value of the lower priority.**

1. **Browser default style**
2. **Tag selector**
3. **Direct child selector**
4. **Child selector**
5. **Class selector**
6. **Attribute selector**
7. **ID selector**

**Note: “!important” is used to override the original style precedence.**

**Syn: selector{**

**Property:value !important;**

**…**

**}**

Selector => h1

Declaration => {

color:blue;font size:12px;

}

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

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

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

For Example:

->color is property and blue is value.

->font size is property and 12px is value.

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

Example :

In the following example all p elements will be center-aligned, with a blue text color:

p {

color: blue;

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.

Selector Description

\* Universal selector (all elements)

divTag selector (all <div> elements)

.blue Class selector (all elements with class blue)

.blue.redAll elements with class blue and red (a type of Compound selector)

#headline ID selector (the element with "id" attribute set to headline)

:lang(en) Element that matches :lang declaration, for example <span lang="en">

div> p child selector

**Note:** The value of an ID must be unique in a web page. It is a violation of the HTML standard to use the value of an ID more than once in the same document tree.

**Tag Selector:**

These are popularly known as type selectors. It matches every instance of the element type in the document free.

syntax:

div{

styles

styles

}

ex:

<head>

<style type='text/css'>

h1{

color:#FF0000;

background-color:#FFFF00;

font-size:20px;

text-decoration:underline;

font-family:comic sans ms;

}

</style>

</head>

<body>

<h1>welcome to type or tag selector</h1>

</body>

**THE UNIVERSAL SELECTORS** : Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type

\* {

color: #000000;

}

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

**THE ELEMENT SELECTOR :** The element selector selects elements based on the element name. You can select all p elements on a page like this (in this case, all p elements will be center-aligned, with a red text color) :

p {

text-align: center;

color: red;

}

**THE DESCENDANT SELECTOR** : Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to em element only when it lies inside ul tag.

Ul.em {

color: #000000;

}

**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.

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

#para1 {

text-align: center;

color: red;

}

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

#black h2 {

color: #000000;

}

EX:

<head>

<style type="text/css">

#h1{

color:#FF0000;

font-size:20px;

font-family:comic sans ms;

}

</style>

</head>

<body>

<p id='h1'>welcome to id selector</p>

</body>

Ex with prefix:

<head>

<style type="text/css">

b #h1{

color:FF00000;

font-size:20px;

font-decoration:underline;

font-family:comic san cs;

}

</style>

</head>

<body>

<b id="h1">welcome to id selector</b>

</body>

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

NOTE: An id name cannot start with a number.

**THE CLASS SELECTORS :**

The class selector selects elements with a specific class attribute.

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

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

.center {

text-align: center;

color: red;

}

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

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

**EX:**

<head>

<style type='text/css'>

.div{

clor:blue;

font-size:20px;

font-family:tahoma;

}

</style>

</head>

<body>

<div class="div">welcome to class selector</div>

<div class="div">

</body>

**GROUPING SELECTORS**

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

h1 {

text-align: center;

color: blue;

}

h2 {

text-align: center;

color: blue;

}

p {

text-align: center;

color: blue;

}

It will be better to group the selectors, to minimize the code. To group selectors, separate each selector with a comma. In the example below we have grouped the selectors from the code above:

h1, h2, p {

text-align: center;

color: red;

}

Before CSS

<!DOCTYPE html>

<html>

<head>

<title>Example</title>

</head>

<body>

<main>

<h1>HTML Page</h1>

<p>This is a basic web page.</p>

</main>

</body>

</html>

After CSS

In this example we add some CSS.

<!DOCTYPE html>

<html>

<head>

<title>Example</title>

<style>

main {

width: 200px;

height: 200px;

padding: 10px;

background: beige;

}

h1 {

font-family: fantasy, cursive, serif;

color: olivedrab;

border-bottom: 1px dotted darkgreen;

}

p {

font-family: sans-serif;

color: orange;

}

</style>

</head>

<body>

<main>

<h1>HTML Page</h1>

<p>This is a basic web page.</p>

</main>

</body>

</html>

All we did was add the following code to the example:

<style>

main {

width: 200px;

height: 200px;

padding: 10px;

background: beige;

}

h1 {

font-family: cursive;

color: olivedrab;

border-bottom: 1px dotted darkgreen;

}

p {

font-family: sans-serif;

color: orange;

}

</style>

CSS Versions

1. css 1.0 [1996]
2. css 2.0 [1998]
3. css 3.0 [2008]
4. css 4.0 [2014]

Version 4 comes with:-

CSS-Pro

CSS-Mobile

# External Stylesheet

An external CSS stylesheet can be applied to any number of HTML documents by placing a <link> element in each HTML document.

The attribute rel of the <link> tag has to be set to "stylesheet", and the href attribute to the relative or absolute path to the stylesheet. While using relative URL paths is generally considered good practice, absolute paths can be used, too. In HTML5 the type attribute can be omitted.

It is recommended that the <link> tag be placed in the HTML ﬁle's <head> tag so that the styles are loaded before the elements they style. Otherwise, users will see a ﬂash of unstyled content.

Example

hello-world.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

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

</head>

<body>

<h1>Hello world!</h1>

<p>I ♥ CSS</p></body></html>

style.css

h1 {

color: green;

text-decoration: underline;

}

p {

font-size: 25px;

font-family: 'Trebuchet MS', sans-serif;

}

Make sure you include the correct path to your CSS ﬁle in the href. If the CSS ﬁle is in the same folder as your HTML ﬁle then no path is required (like the example above) but if it's saved in a folder, then specify it like this href="foldername/style.css".

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

External stylesheets are considered the best way to handle your CSS. There's a very simple reason for this: when you're managing a site of, say, 100 pages, all controlled by a single stylesheet, and you want to change your linkcolors from blue to green, it's a lot easier to make the change in your CSS ﬁle and let the changes "cascade" throughout all 100 pages than it is to go into 100 separate pages and make the same change 100 times. Again, if you want to completely change the look of your website, you only need to update this one ﬁle.

You can load as many CSS ﬁles in your HTML page as needed.

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

CSS rules are applied with some basic rules, and order does matter. For example, if you have a main.css ﬁle with some code in it:

p.green{ color: #00FF00; }

All your paragraphs with the 'green' class will be written in light green, but you can override this with another .css ﬁle just by including it after main.css. You can have override.css with the following code follow main.css, for example:

p.green{ color: #006600; }

Now all your paragraphs with the 'green' class will be written in darker green rather than light green.

Other principles apply, such as the '!important' rule, speciﬁcity, and inheritance.

When someone ﬁrst visits your website, their browser downloads the HTML of the current page plus the linked CSS ﬁle. Then when they navigate to another page, their browser only needs to download the HTML of that page; the CSS ﬁle is cached, so it does not need to be downloaded again. Since browsers cache the external stylesheet, your pages load faster.

**Internal Styles**

CSS enclosed in <style></style> tags within an HTML document functions like an external stylesheet, except that it lives in the HTML document it styles instead of in a separate ﬁle, and therefore can only be applied to the document in which it lives. Note that this element must be inside the <head> element for HTML validation (though it will work in all current browsers if placed in body).

<head>

<style>

h1 {

color: green;

text-decoration: underline;

}

p {

font-size: 25px;

font-family: 'Trebuchet MS', sans-serif;

}

</style>

</head>

<body>

<h1>Hello world!</h1>

<p>I ♥ CSS</p>

</body>

**CSS @import rule (one of CSS at-rule)**

The @import CSS at-rule is used to import style rules from other style sheets. These rules must precede all other types of rules, except @charset rules; as it is not a nested statement, @import cannot be used inside conditional group at-rules. @import.

How to use @import

You can use @import rule in following ways:

A. With internal style tag

<style> @import url('/css/styles.css'); </style>

B. With external stylesheet

The following line imports a CSS ﬁle named additional-styles.css in the root directory into the CSS ﬁle in which it appears:

@import '/additional-styles.css';

Importing external CSS is also possible. A common use case are font ﬁles.

@import 'https://fonts.googleapis.com/css?family=Lato';

An optional second argument to @import rule is a list of media queries:

@import '/print-styles.css' print; @import url('landscape.css') screen and (orientation:landscape); Section 1.4: Inline Styles

Use inline styles to apply styling to a speciﬁc element. Note that this is not optimal. Placing style rules in a <style> tag or external CSS ﬁle is encouraged in order to maintain a distinction between content and presentation.

Inline styles override any CSS in a <style> tag or external style sheet. While this can be useful in some circumstances, this fact more often than not reduces a project's maintainability.

The styles in the following example apply directly to the elements to which they are attached.

<h1 style="color: green; text-decoration: underline;">Hello world!</h1>

<p style="font-size: 25px; font-family: 'Trebuchet MS';">I♥ CSS</p>

Inline styles are generally the safest way to ensure rendering compatibility across various email clients, programs and devices, but can be time-consuming to write and a bit challenging to manage.

**Changing CSS with JavaScript**

Pure JavaScript

It's possible to add, remove or change CSS property values with JavaScript through an element's style property.

var el = document.getElementById("element"); el.style.opacity = 0.5; el.style.fontFamily = 'sans-serif';

Note that style properties are named in lower camel case style. In the example you see that the css property fontfamily becomes fontFamily in javascript.

As an alternative to working directly on elements, you can create a <style> or <link> element in JavaScript and append it to the <body> or <head> of the HTML document.

**Styling Lists with CSS**

There are three diﬀerent properties for styling list-items: list-style-type, list-style-image, and list-styleposition, which should be declared in that order. The default values are disc, outside, and none, respectively. Each property can be declared separately, or using the list-style shorthand property.

list-style-type deﬁnes the shape or type of bullet point used for each list-item.

Some of the acceptable values for list-style-type:

disc circle square decimal lower-roman upper-roman none

To use square bullet points for each list-item, for example, you would use the following property-value pair:

li { list-style-type: square; }

The list-style-image property determines whether the list-item icon is set with an image, and accepts a value of none or a URL that points to an image.

li { list-style-image: url(images/bullet.png); }

The list-style-position property deﬁnes where to position the list-item marker, and it accepts one of two values: "inside" or "outside".

li { list-style-position: inside; }

Comments

**Single Line**

/\* This is a CSS comment \*/

div {

color: red; /\* This is a CSS comment \*/

}

# **Background**

With CSS you can set colors, gradients, and images as the background of an element.

It is possible to specify various combinations of images, colors, and gradients, and adjust the size, positioning, and repetition (among others) of these.

**Background Color**

The background-color property sets the background color of an element using a color value or through keywords, such as transparent, inherit or initial.

transparent, speciﬁes that the background color should be transparent. This is default. inherit, inherits this property from its parent element. initial, sets this property to its default value.

This can be applied to all elements, and ::first-letter/::first-line pseudo-elements.

Colors in CSS can be speciﬁed by diﬀerent methods.

Color names

**CSS**

div { background-color: red; /\* red \*/ }

**HTML**

<div>This will have a red background</div>

The example used above is one of several ways that CSS has to represent a single color.

**Hex color codes**

Hex code is used to denote RGB components of a color in base-16 hexadecimal notation. #ﬀ0000, for example, is bright red, where the red component of the color is 256 bits (ﬀ) and the corresponding green and blue portions of the color is 0 (00).

If both values in each of the three RGB pairings (R, G, and B) are the same, then the color code can be shortened into three characters (the ﬁrst digit of each pairing). #ff0000 can be shortened to #f00, and #ffffff can be shortened to #fff.

Hex notation is case-insensitive.

body { background-color: #de1205; /\* red \*/ }

.main {

background-color: #00f; /\* blue \*/ }

**RGB / RGBa**

Another way to declare a color is to use RGB or RGBa.

RGB stands for Red, Green and Blue, and requires of three separate values between 0 and 255, put between brackets, that correspond with the decimal color values for respectively red, green and blue.

RGBa allows you to add an additional alpha parameter between 0.0 and 1.0 to deﬁne opacity.

header { background-color: rgb(0, 0, 0); /\* black \*/ }

footer { background-color: rgba(0, 0, 0, 0.5); /\* black with 50% opacity \*/ }

**HSL / HSLa**

Another way to declare a color is to use HSL or HSLa and is similar to RGB and RGBa.

HSL stands for hue, saturation, and lightness, and is also often called HLS:

Hue is a degree on the color wheel (from 0 to 360). Saturation is a percentage between 0% and 100%. Lightness is also a percentage between 0% and 100%.

HSLa allows you to add an additional alpha parameter between 0.0 and 1.0 to deﬁne opacity.

li a { background-color: hsl(120, 100%, 50%); /\* green \*/ }

#p1 { background-color: hsla(120, 100%, 50%, .3); /\* green with 30% opacity \*/ }

Interaction with background-image

The following statements are all equivalent:

body { background: red; background-image: url(partiallytransparentimage.png); }

body { background-color: red; background-image: url(partiallytransparentimage.png); }

body { background-image: url(partiallytransparentimage.png); background-color: red;}

body { background: red url(partiallytransparentimage.png); }

They will all lead to the red color being shown underneath the image, where the parts of the image are transparent, or the image is not showing (perhaps as a result of background-repeat).

Note that the following is not equivalent:

body { background-image: url(partiallytransparentimage.png); background: red; }

Here, the value of background overrides your background-image.

EX:

ex:

<head>

<style type='text/css'>

div{

background-color:#FF00FF;

}

</style>

</head>

<body>

<div> working with css background...</div>

</body>

**Background Gradients**

Gradients are new image types, added in CSS3. As an image, gradients are set with the background-image property, or the background shorthand.

There are two types of gradient functions, linear and radial. Each type has a non-repeating variant and a repeating variant:

linear-gradient() repeating-linear-gradient() radial-gradient() repeating-radial-gradient()

**linear-gradient()**

A linear-gradient has the following syntax

background: linear-gradient( <direction>?, <color-stop-1>, <color-stop-2>, ...);

Value Meaning

<direction>Could be an argument like to top, to bottom, to right or to left; or an angle as 0deg, 90deg... . The angle starts from to top and rotates clockwise. Can be speciﬁed in deg, grad, rad, or turn. If omitted, the gradient ﬂows from top to bottom

<color-stop-list>List of colors, optionally followed each one by a percentage or length to display it at. For example, yellow 10%, rgba(0,0,0,.5) 40px, #fff 100%...

For example, this creates a linear gradient that starts from the right and transitions from red to blue

**.linear-gradient**{ background: linear-gradient(to left, red, blue); /\* you can also use 270deg \*/ }

You can create a diagonal gradient by declaring both a horizontal and vertical starting position.

.diagonal-linear-gradient { background: linear-gradient(to left top, red, yellow 10%);

}

It is possible to specify any number of color stops in a gradient by separating them with commas. The following examples will create a gradient with 8 color stops

.linear-gradient-rainbow { background: linear-gradient(to left, red, orange, yellow, green, blue, indigo, violet) }

radial-gradient() .radial-gradient-simple { background: radial-gradient(red, blue); }

.radial-gradient { background: radial-gradient(circle farthest-corner at top left, red, blue); }

Value Meaning

circleShape of gradient. Values are circle or ellipse, default is ellipse.

farthest-cornerKeywords describing how big the ending shape must be. Values are closest-side, farthestside, closest-corner, farthest-corner

top left Sets the position of the gradient center, in the same way as background-position.

**Repeating gradients**

Repeating gradient functions take the same arguments as the above examples, but tile the gradient across the background of the element.

.bullseye {

background: repeating-radial-gradient(red, red 10%, white 10%, white 20%);

} .

warning {

background: repeating-linear-gradient(-45deg, yellow, yellow 10%, black 10%, black 20% );

}

Value Meaning

-45deg Angle unit. The angle starts from to top and rotates clockwise. Can be speciﬁed in deg, grad, rad, or turn.

to left Direction of gradient, default is to bottom. Syntax: to [y-axis(top OR bottom)] [x-axis(left OR right)] ie to top right

yellow10% Color, optionally followed by a percentage or length to display it at.Repeated two or more times.

Note that HEX, RGB, RGBa, HSL, and HSLa color codes may be used instead of color names. Color names were used for the sake of illustration. Also note that the radial-gradient syntax is much more complex than linear-gradient, and a simpliﬁed version is shown here.

**Background Image**

The background-image property is used to specify a background image to be applied to all matched elements. By default, this image is tiled to cover the entire element, excluding margin.

.myClass {

background-image: url('/path/to/image.jpg');

}

To use multiple images as background-image, deﬁne comma separated url()

.myClass {

background-image: url('/path/to/image.jpg'),

url('/path/to/image2.jpg');

}

The images will stack according to their order with the ﬁrst declared image on top of the others and so on.

**Value Result**

url('/path/to/image.jpg') Specify background image's path(s) or an image resource speciﬁed with data URI schema (apostrophes can be omitted), separate multiples by comma

noneNo background image

initialDefault value

Inherit inheritparent's value

More CSS for Background Image

This following attributes are very useful and almost essential too.

background-size: xpxypx | x% y%;

background-repeat: no-repeat | repeat | repeat-x | repeat-y;

background-position: left offset (px/%) right offset (px/%) | center center | left top | right bottom;

**Background Shorthand**

The background property can be used to set one or more background related properties:

Value Description CSS Ver.

background-imageBackground image to use 1+

background-colorBackground color to apply 1+

background-positionBackground image's position1+

background-sizeBackground image's size 3+

background-repeatHow to repeat background image 1+

background-origin How the background is positioned (ignored when background-attachment is fixed) 3+ background-clip How the background is painted relative to the content-box, border-box, or the padding-box 3+

background-attachment How the background image behaves, whether it scrolls along with its containing block or has a ﬁxed position within the viewport 1+

initialSets the property to value to default 3+

inheritInherits property value from parent 2+

The order of the values does not matter and every value is optional

Syntax

The syntax of the background shorthand declaration is:

**background**: [<background-image>] [<background-color>] [<background-position>]/[<background-size>] [<background-repeat>] [<background-origin>] [<background-clip>] [<background-attachment>] [<initial|inherit>];

Examples

**background**: red;

Simply setting a background-color with the redvalue.

**background**: border-box red;

Setting a background-clip to border-box and a background-color to red.

**background**: no-repeat center url("somepng.jpg");

Sets a background-repeat to no-repeat, background-origin to center and a background-image to an image.

**background**: url('pattern.png') green;

In this example, the background-color of the element would be set to green with pattern.png, if it is available, overlayed on the colour, repeating as often as necessary to ﬁll the element. If pattern.png includes any transparency then the green colour will be visible behind it.

background: #000000 url("picture.png") top left / 600px auto no-repeat;

In this example we have a black background with an image 'picture.png' on top, the image does not repeat in either axis and is positioned in the top left corner. The / after the position is to be able to include the size of the background image which in this case is set as 600px width and auto for the height. This example could work well with a feature image that can fade into a solid colour.

NOTE: Use of the shorthand background property resets all previously set background property values, even if a value is not given. If you wish only to modify a background property value previously set, use a longhand property instead.

**Background Size**

General overview

The background-size property enables one to control the scaling of the background-image. It takes up to two values, which determine the scale/size of the resulting image in vertical and and horizontal direction. If the property is missing, its deemed auto in both width and height.

auto will keep the image's aspect ratio, if it can be determined. The height is optional and can be considered auto. Therefore, on a 256 px × 256 px image, all the following background-size settings would yield an image with height and width of 50 px:

background-size: 50px; background-size: 50px auto; /\* same as above \*/ background-size: auto 50px; background-size: 50px 50px;

So if we started with the following picture (which has the mentioned size of 256 px × 256 px),

we'll end up with a 50 px × 50 px on the user's screen, contained in the background of our element:

One can also use percentage values to scale the image with respect of the element. The following example would yield a 200 px × 133 px drawn image:

#withbackground {

background-image: url(to/some/background.png);

background-size: 100% 66%; width: 200px; height: 200px;

padding: 0; margin: 0;

}

The behaviour depends on the background-origin.

**Keeping the aspect ratio**

The last example in the previos section lost its original aspect ratio. The circle got into an ellipse, the square into a rectangle, the triangle into another triangle.

The length or percentage approach isn't ﬂexible enough to keep the aspect ratio at all times. auto doesn't help, since you might not know which dimension of your element will be larger. However, to cover certain areas with an

image (and correct aspect ratio) completely or to contain an image with correct aspect ratio completely in a background area, the values, contain and cover provide the additional functionality.

**Eggsplanation for contain and cover**

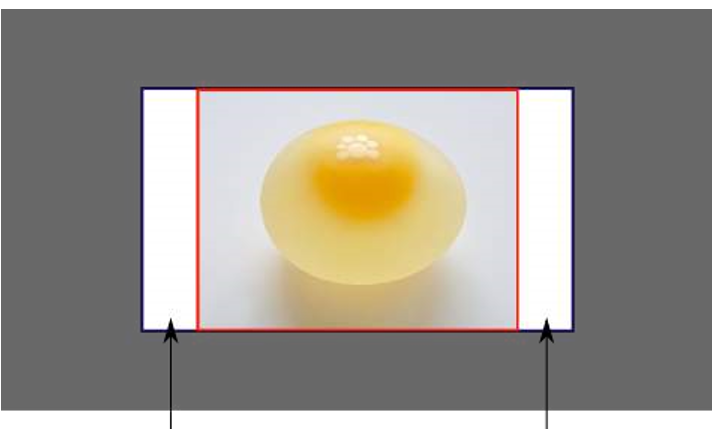
Sorry for the bad pun, but we're going to use a picture of the day by BiswarupGanguly for demonstration. Lets say that this is your screen, and the gray area is outside of your visible screen. For demonstration, We're going to assume a 16 × 9 ratio.

We want to use the aforementioned picture of the day as a background. However, we cropped the image to 4x3 for some reason. We could set the background-size property to some ﬁxed length, but we will focus on contain and cover. Note that I also assume that we didn't mangle the width and/or height of body.

**contain**

Scale the image, while preserving its intrinsic aspect ratio (if any), to the largest size such that both its width and its height can ﬁt inside the background positioning area.

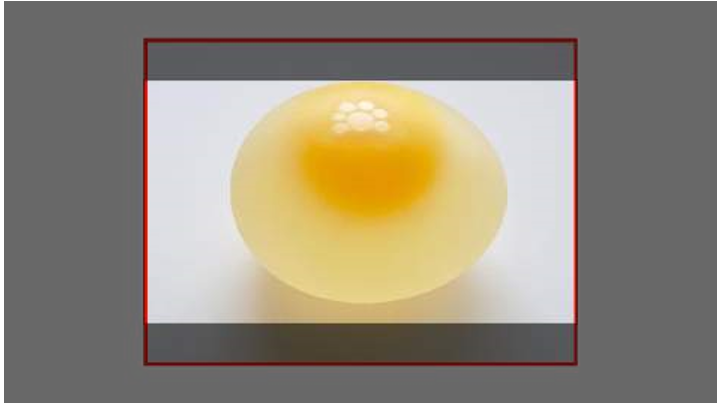
This makes sure that the background image is always completely contained in the background positioning area, however, there could be some empty space ﬁlled with your background-color in this case:



**cover**

Scale the image, while preserving its intrinsic aspect ratio (if any), to the smallest size such that both its width and its height can completely cover the background positioning area.

This makes sure that the background image is covering everything. There will be no visible background-color, however depending on the screen's ratio a great part of your image could be cut oﬀ:



Demonstration with actual code div > div

{ background-image: url(http://i.stack.imgur.com/r5CAq.jpg);

background-repeat: no-repeat;

background-position: center center;

background-color: #ccc; border: 1px solid;

width: 20em; height: 10em;

}

div.contain {

background-size: contain;

}

div.cover {

background-size: cover;

}

**Additional styles for the explanation boxes**

div> div {

margin: 0 1ex 1ex 0; float: left;

}

div + div {

clear: both; border-top: 1px dashed silver; padding-top:1ex;

}

div> div::after {

background-color: #000; color: #fefefe;

margin: 1ex; padding: 1ex; opacity: 0.8; display: block; width: 10ex; font-size: 0.7em; content: attr(class);

}

<div><div class="contain"></div>

<p>

Note the grey background. The image does not cover the whole region, but it's fully <em>contained</em>.

</p>

</div>

<div><div class="cover"></div>

<p>Note the ducks/geese at the bottom of the image. Most of the water is cut, as well as a part of the sky. You don't see the complete image anymore, but neither do you see any background color; the image

<em>covers</em> all of the <code>&lt;div&gt;</code>.</p>

</div>

**Background Position**

The background-position property is used to specify the starting position for a background image or gradient

.myClass{ background-image: url('path/to/image.jpg'); background-position: 50% 50%; }

The position is set using an X and Y co-ordinate and be set using any of the units used within CSS.

Unit Description

value% value% A percentage for the horizontal oﬀset is relative to (width of background positioning area - width of background image). A percentage for the vertical oﬀset is relative to (height of background positioning area - height of background image) The size of the image is the size given by background-size.

valuepxvaluepxOﬀsets background image by a length given in pixels relative to the top left of the background positioning area

Units in CSS can be speciﬁed by diﬀerent methods .

**Longhand Background Position Properties**

In addition to the shorthand property above, one can also use the longhand background properties backgroundposition-x and background-position-y. These allow you to control the x or y positions separately.

NOTE: This is supported in all browsers except Firefox (versions 31-48) 2. Firefox 49, to be released September 2016, will support these properties. Until then, there is a Firefox hack within this Stack Overﬂow answer.

**The background-origin property**

The background-origin property speciﬁes where the background image is positioned.

Note: If the background-attachment property is set to fixed, this property has no eﬀect.

Default value: padding-box

Possible values:

padding-box - The position is relative to the padding box border-box - The position is relative to the border box content-box - The position is relative to the content box initial inherit

CSS

.example { width: 300px; border: 20px solid black; padding: 50px; background: url(https://static.pexels.com/photos/6440/magazines-desk-work-workspace-medium.jpg); background-repeat: no-repeat; }

.example1 {}

.example2 { background-origin: border-box; }

.example3 { background-origin: content-box; }

HTML

<p>No background-origin (padding-box is default):</p>

<div class="example example1">

<h2>LoremIpsum Dolor</h2>

<p>Loremipsum dolor sit amet, consectetueradipiscingelit, seddiamnonummynibheuismodtinciduntutlaoreetdolore magna aliquameratvolutpat.</p>

<p>Utwisienim ad minim veniam, quisnostrudexercitationullamcorpersuscipitlobortisnislutaliquip ex eacommodoconsequat.</p>

</div>

<p>background-origin: border-box:</p>

<div class="example example2">

<h2>LoremIpsum Dolor</h2>

<p>Loremipsum dolor sit amet, consectetueradipiscingelit, seddiamnonummynibheuismodtinciduntutlaoreetdolore magna aliquameratvolutpat.</p>

<p>Utwisienim ad minim veniam, quisnostrudexercitationullamcorpersuscipitlobortisnislutaliquip ex eacommodoconsequat.</p>

</div>

<p>background-origin: content-box:</p>

<div class="example example3">

<h2>LoremIpsum Dolor</h2>

<p>Loremipsum dolor sit amet, consectetueradipiscingelit, seddiamnonummynibheuismodtinciduntutlaoreetdolore magna aliquameratvolutpat.</p>

<p>Utwisienim ad minim veniam, quisnostrudexercitationullamcorpersuscipitlobortisnislutaliquip ex eacommodoconsequat.</p></div>

**Multiple Background Image**

In CSS3, we can stack multiple background in the same element.

#mydiv { background-image: url(img\_1.png), /\* top image \*/ url(img\_2.png), /\* middle image \*/ url(img\_3.png); /\* bottom image \*/ background-position: right bottom, left top, right top; background-repeat: no-repeat, repeat, no-repeat; }

Images will be stacked atop one another with the ﬁrst background on top and the last background in the back. img\_1 will be on top, the img\_2 and img\_3 is on bottom.

We can also use background shorthand property for this:

#mydiv { background: url(img\_1.png) right bottom no-repeat, url(img\_2.png) left top repeat, url(img\_3.png) right top no-repeat; }

We can also stack images and gradients:

#mydiv{ background: url(image.png) right bottom no-repeat, linear-gradient(to bottom, #fff 0%,#000 100%); }

**Background Attachment**

The background-attachment property sets whether a background image is ﬁxed or scrolls with the rest of the page.

body {

background-image: url('img.jpg');

background-attachment: fixed;

}

Value Description

scrollThe background scrolls along with the element. This is default.

ﬁxed The background is ﬁxed with regard to the viewport.

localThe background scrolls along with the element's contents.

initialSets this property to its default value.

inheritInherits this property from its parent element.

**Examples background-attachment: scroll**

The default behaviour, when the body is scrolled the background scrolls with it:

body { background-image: url('image.jpg'); background-attachment: scroll; }

**background-attachment: ﬁxed**

The background image will be ﬁxed and will not move when the body is scrolled:

body { background-image: url('image.jpg'); background-attachment: fixed; }

**background-attachment: local**

The background image of the div will scroll when the contents of the div is scrolled.

div {

background-image: url('image.jpg'); background-attachment: local;

}

**Background Clip**

Deﬁnition and Usage: The background-clip property speciﬁes the painting area of the background.

Default value: border-box

Values

* **border-box** is the default value. This allows the background to extend all the way to the outside edge of the element's border.
* **padding-box** clips the background at the outside edge of the element's padding and does not let it extend into the border;
* **content-box** clips the background at the edge of the content box. inherit applies the setting of the parent to the selected element.

CSS

.example { width: 300px; border: 20px solid black; padding: 50px; background: url(https://static.pexels.com/photos/6440/magazines-desk-work-workspace-medium.jpg);

background-repeat: no-repeat;

}

.example1 {}

.example2 { background-origin: border-box; }

.example3 { background-origin: content-box; }

**HTML**

<p>No background-origin (padding-box is default):</p>

<div class="example example1"><h2>LoremIpsum Dolor</h2><p>Loremipsum dolor sit amet, consectetueradipiscingelit, seddiamnonummynibheuismodtinciduntutlaoreetdolore magna aliquameratvolutpat.</p><p>Utwisienim ad minim veniam, quisnostrudexercitationullamcorpersuscipitlobortisnislutaliquip ex eacommodoconsequat.</p></div>

<p>background-origin: border-box:</p><div class="example example2">

<h2>LoremIpsum Dolor</h2>

<p>Loremipsum dolor sit amet, consectetueradipiscingelit, seddiamnonummynibheuismodtinciduntutlaoreetdolore magna aliquameratvolutpat.</p>

<p>Utwisienim ad minim veniam, quisnostrudexercitationullamcorpersuscipitlobortisnislutaliquip ex eacommodoconsequat.</p>

</div>

<p>background-origin: content-box:</p><div class="example example3">

<h2>LoremIpsum Dolor</h2>

<p>Loremipsum dolor sit amet, consectetueradipiscingelit, seddiamnonummynibheuismodtinciduntutlaoreetdolore magna aliquameratvolutpat.</p>

<p>Utwisienim ad minim veniam, quisnostrudexercitationullamcorpersuscipitlobortisnislutaliquip ex eacommodoconsequat.</p>

</div>

**Background Repeat**

The background-repeat property sets if/how a background image will be repeated.

By default, a background-image is repeated both vertically and horizontally.

div { background-image: url("img.jpg"); background-repeat: repeat-y; }

Here's how a background-repeat: repeat-y looks like:



**background-blend-mode Property**

.my-div { width: 300px; height: 200px; background-size: 100%; background-repeat: no-repeat; background-image: linear-gradient(to right, black 0%,white 100%), url(‘a.jpeg'); background-blend-mode:saturation; } <div class="my-div">Loremipsum</div>

CSS Syntax: background-blend-mode: normal | multiply | screen | overlay | darken | lighten | color-dodge | saturation | color | luminosity;

**Background Color with Opacity**

If you set opacity on an element it will aﬀect all its child elements. To set an opacity just on the background of an element you will have to use RGBA colors. Following example will have a black background with 0.6 opacity.

/\* Fallback for web browsers that don't support RGBa \*/ background-color: rgb(0, 0, 0);

/\* RGBa with 0.6 opacity \*/ background-color: rgba(0, 0, 0, 0.6);

/\* For IE 5.5 - 7\*/ filter: progid:DXImageTransform.Microsoft.gradient(startColorstr=#99000000, endColorstr=#99000000);

/\* For IE 8\*/ -ms-filter: "progid:DXImageTransform.Microsoft.gradient(startColorstr=#99000000, endColorstr=#99000000)";