CSS FAQ

**Basics**

1. How to apply CSS to HTML

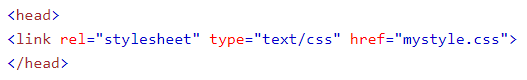
There are three different ways to apply CSS to an HTML document which are.

**External Stylesheet**

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

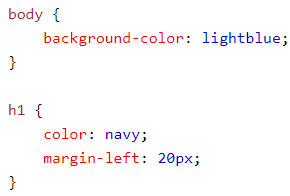
Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section:

**Example:**



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

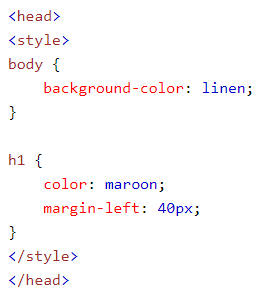
Here is how the "mystyle.css" looks:



**Internal Stylesheet**

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

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



**Inline Styles**

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

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

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

****

1. How to use whitespace in CSS

The CSS white-space property defines how to handle whitespace within an element.

**Syntax**

The syntax for the white-space CSS property is:



**Parameters or Arguments**

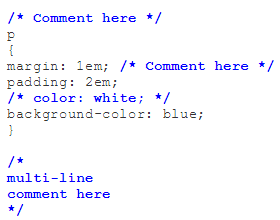
*value*

Determines how to handle whitespace inside of an element. It can be one of the following:

|  |  |
| --- | --- |
| Value | Description |
| normal | Whitespace is collapsed. Newline characters are treated as whitespace and collapsed. Text wrapping is allowed. p { white-space: normal; } |
| pre | Whitespace is preserved. Lines are broken by newline characters. Text wrapping is not allowed. p { white-space: pre; } |
| nowrap | Whitespace is collapsed. No line breaks. Text wrapping is not allowed. p { white-space: nowrap; } |
| pre-wrap | Whitespace is preserved. Lines are broken by newline characters. Text wrapping is allowed. p { white-space: pre-wrap; } |
| pre-line | Whitespace is collapsed. Lines are broken by newline characters and to fill line boxes. Text wrapping is allowed. p { white-space: pre-line; } |
| inherit | Element will inherit the white-space from its parent element p { white-space: inherit; } |

1. How to write comments in CSS

A CSS comment begins with "/\*", and ends with "\*/". Comments can appear before or within rule sets as well as across multiple lines. They can also be used to comment out entire rules or individual declarations. For example:



1. How to select elements via element name, class or ID

**Simple selectors:** Match one or more elements based on element type, class, or id.

**ID selectors**

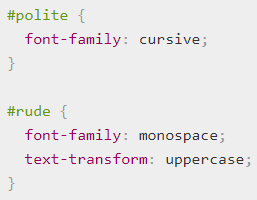
The ID selector consists of a hash/pound symbol (#), followed by the ID name of a given element. Any element can have a unique ID name set with the id attribute. It is up to you what name you choose for the ID. It's the most efficient way to select a single element.

**Example**

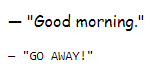
**HTML**

****

A simple style sheet:

****

**Output:**

****

1. How to select elements via attribute name and content

**Attribute selectors:** Match one or more elements based on their attributes/attribute values.

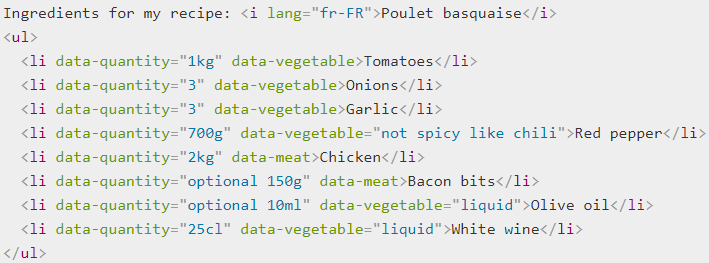
These attribute selectors try to match an exact attribute value:

**[attr] :** This selector will select all elements with the attribute attr, whatever its value.

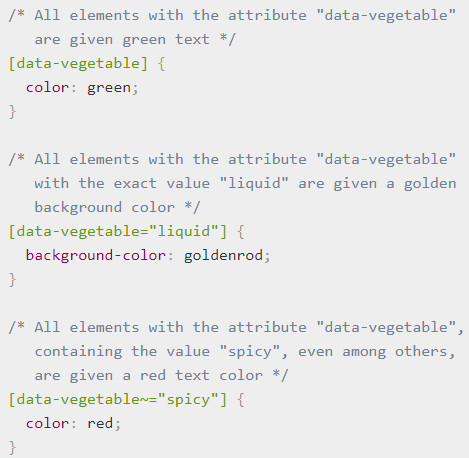
**[attr=val] :** This selector will select all elements with the attribute attr, but only if its value is val.

**[attr~=val]:** This selector will select all elements with the attribute attr, but only if the value val is one of a space-separated list of values contained in attr's value, for example a single class in a space-separated list of classes.

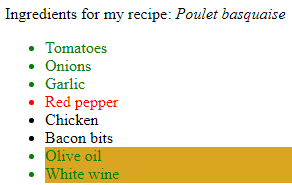
Let's look at an example featuring the following HTML snippet:



And a simple style sheet:



**Output:**

****

1. How to use pseudo-classes

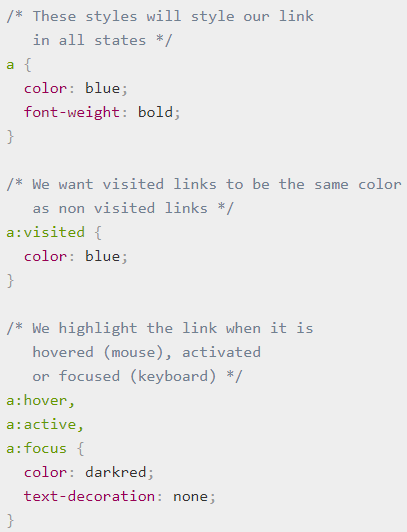
**Pseudo-classes:** Match one or more elements that exist in a certain state, such as an element that is being hovered over by the mouse pointer, or a checkbox that is currently disabled or checked, or an element that is the first child of its parent in the DOM tree.

**Example**

**HTML snippet:**

****

**CSS rules:**

****

**Output:**

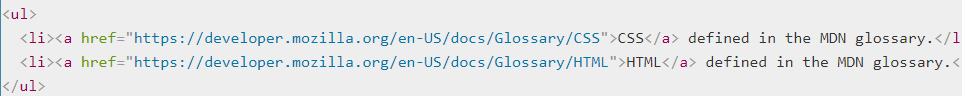
****

1. How to use pseudo-elements

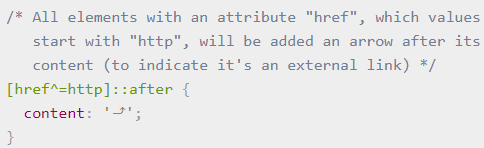
**Pseudo-elements:** Match one or more parts of content that are in a certain position in relation to an element, for example the first word of each paragraph, or generated content appearing just before an element.

**Example**

CSS example that selects the space just after all absolute links and adds an arrow in that space:

****

Let's add this CSS rule:



**Output:**

****

1. How to apply multiple selectors to the same rule

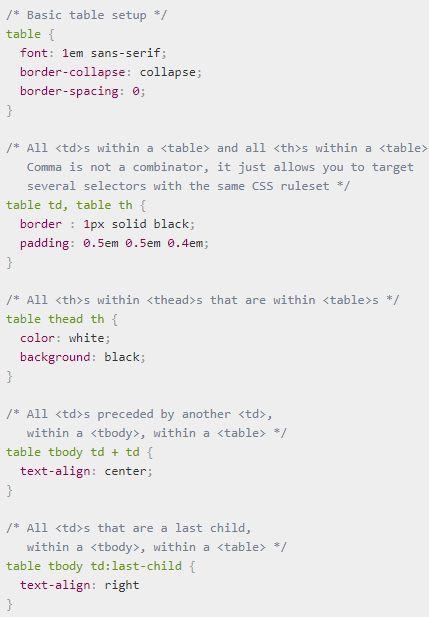
**Combinators:** These are not exactly selectors themselves, but ways of combining two or more selectors in useful ways for very specific selections. So for example, you could select only paragraphs that are direct descendants of divs, or paragraphs that come directly after headings.

|  |  |
| --- | --- |
| Combinators | Select |
| A, B | Any element matching A and/or B (see also [Multiple selectors on one rule](https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Combinators_and_multiple_selectors#Multiple_selectors_on_one_rule), below). |
| A B | Any element matching B that is a descendant of an element matching A (that is: a child, or a child of a child, etc.) |
| A > B | Any element matching B that is a direct child of an element matching A. |
| A + B | Any element matching B that is the next sibling of an element matching A (that is: the next child of the same parent.) |
| A ~ B | Any element matching B that is one of the next siblings of an element matching A (that is: one of the next children of the same parent.) |

**Combinators example**

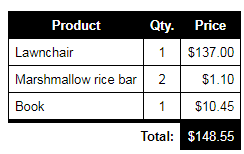


Then let's use the following style sheet:



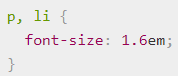


**Output:**

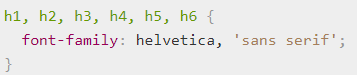
****

**Multiple selectors:** Again, these are not separate selectors; the idea is that you can put multiple selectors on the same CSS rule, separated by commas, to apply a single set of declarations to all the elements selected by those selectors.

**Example**

****

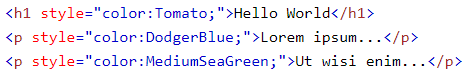
or this



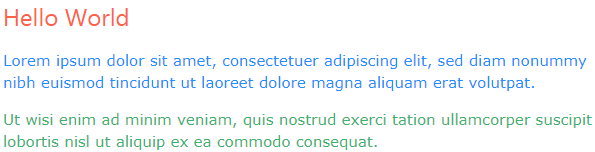
1. How to specify colors in CSS

**Text Color**

You can set the color of text:

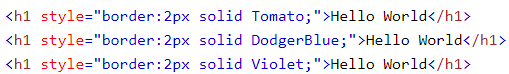


**Output:**

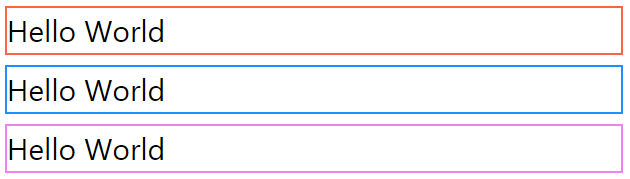


**Border Color**

You can set the color of borders:



**Output:**

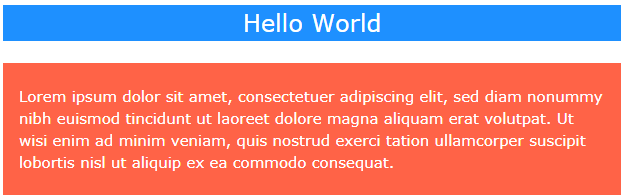
****

**Background Color**

You can set the background color for HTML elements



**Output:**

****

1. How to debug CSS in the browser

There are two that are available in all browsers: The DOM Inspector and the CSS Editor, which are available in Firefox in the page inspector tool.

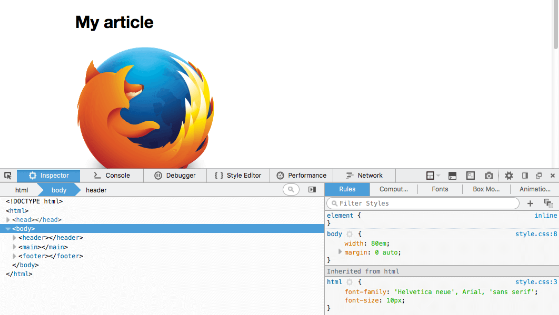
It is meant to be a simple, clear one column web page containing a simple article:



At the moment however it is a bit of a mess:



Let's start investigating the page with the page inspector's features. In Firefox you can open the page inspector using Cmd/Ctrl + I (or by choosing Tools > Web Developer > Inspector from the menu), which will give you a set of tools open in a window on the bottom of the browser like so:



Click on an element inside the DOM Inspector on the left, the CSS editor on the right will update to show all the CSS rules applied to that element. This is really useful, especially as any invalid properties appear with a line through them and a little warning symbol next to them.



**CSS and Text**

1. How to style text

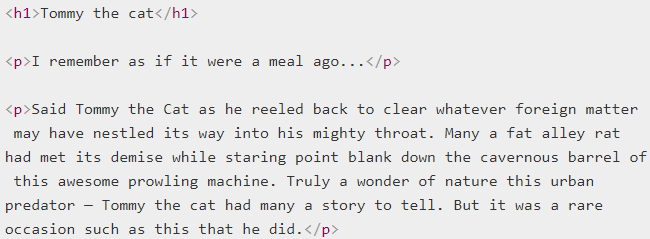
The CSS properties used to style text generally fall into two categories.

**Font styles:** Properties that affect the font that is applied to the text, affecting what font is applied, how big it is, whether it is bold, italic, etc.

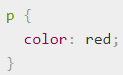
**Text layout styles:** Properties that affect the spacing and other layout features of the text, allowing manipulation of, for example, the space between lines and letters, and how the text is aligned within the content box.

**Example**

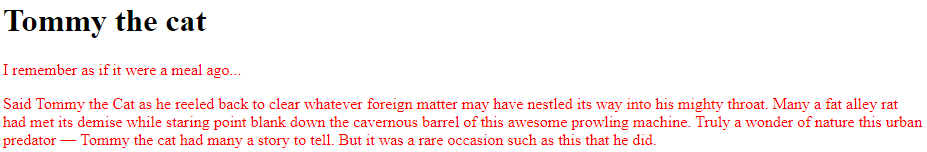
**Font**

****

**Color**

****

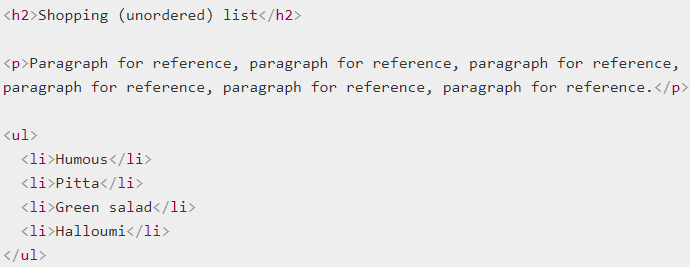
**Output:**

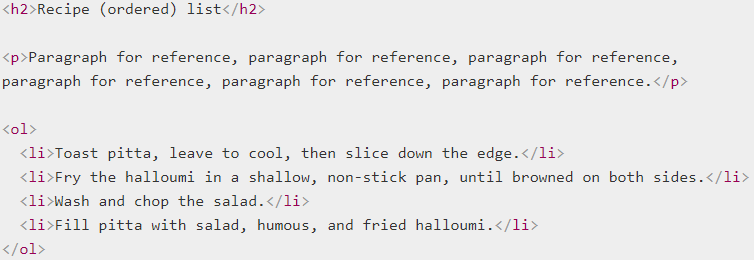
****

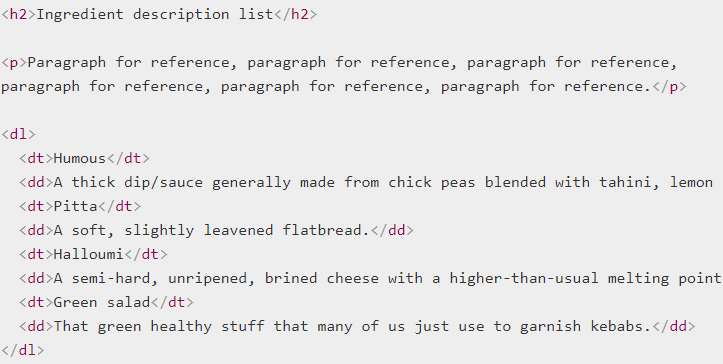
1. How to customize a list of elements

There are three way to customize a list of elements which are unordered, ordered and description lists.

The HTML for list example looks like:







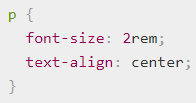
1. How to style links

There are different states that links can exist in, which can be styled using different pseudo-classes:

* **Link (unvisited):** The default state that a link resides in, when it isn't in any other state. This can be specifically styled using the :link pseudo class.
* **Visited:** A link when it has already been visited (exists in the browser's history), styled using the :visited pseudo class.
* **Hover:** A link when it is being hovered over by a user's mouse pointer, styled using the :hover pseudo class.
* **Focus:** A link when it has been focused (for example moved to by a keyboard user using the Tab key or similar, or programmatically focused using HTMLElement.focus()) — this is styled using the :focus pseudo class.
* **Active:** A link when it is being activated (e.g. clicked on), styled using the :active pseudo class.

The following example illustrates what a link will behave like by default.







You'll notice a few things as you explore the default styles:

* Links are underlined.
* Unvisited links are blue.
* Visited links are purple.
* Hovering a link makes the mouse pointer change to a little hand icon.
* Focused links have an outline around them — you should be able to focus on the links on this page with the keyboard by pressing the tab key (On Mac, you may need enable the Full Keyboard Access: All controls option by pressing Ctrl + F7 before this will work.)
* Active links are red (Try holding down the mouse button on the link as you click it.)

The default styles can be turned off/changed using the following CSS properties:

* **color** for the text color.
* **cursor** for the mouse pointer style — you shouldn't turn this off unless you've got a very good reason.
* **outline** for the text outline (an outline is similar to a border, the only difference being that border takes up space in the box and an outline doesn't; it just sits over the top of the background).

1. How to add shadows to text

You can apply drop shadows to your text using the text-shadow property. This takes up to four values, as shown in the example below:

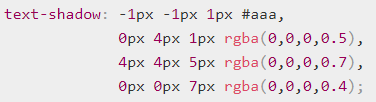


The four properties are as follows:

* The horizontal offset of the shadow from the original text — this can take most available CSS length and size units, but most comonly use px. This value has to be included.
* The vertical offset of the shadow from the original text; behaves basically just like the horizontal offset, except that it moves the shadow up/down, not left/right. This value has to be included.
* The blur radius — a higher value means the shadow is dispersed more widely. If this value is not included, it defaults to 0, which means no blur. This can take most available CSS length and size units.
* The base color of the shadow, which can take any CSS color unit. If not included, it defaults to black.

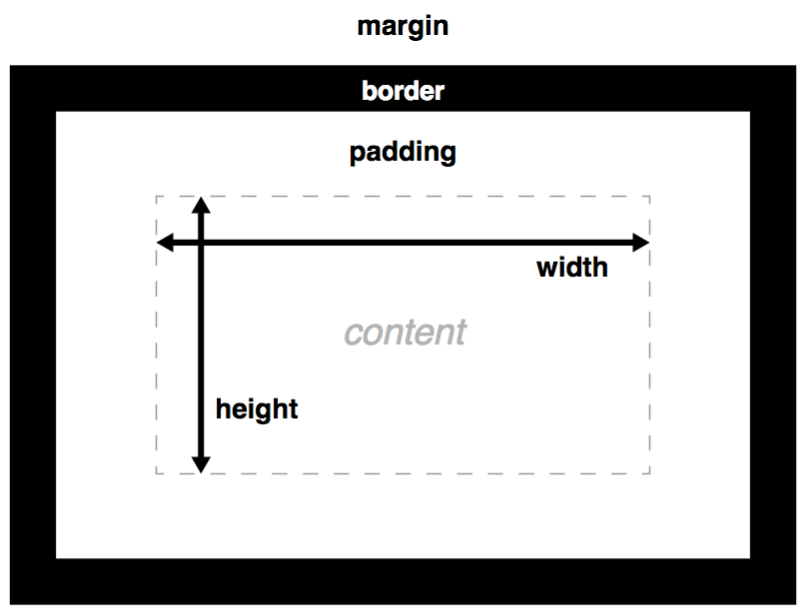
**Multiple Shadows**

You can apply multiple shadows to the same text by including multiple shadow values separated by commas, for example:



**Boxes and Layouts**

1. How to size CSS boxes



**width and height**

The width and height properties set the width and height of the content box, which is the area in which the content of the box is displayed — this content includes both text content sat inside the box, and other boxes representing nested child elements.

**padding**

Padding refers to the inner margin of a CSS box — between the outer edge of the content box and the inner edge of the border. The size of this layer can be set on all four sides at once with the padding shorthand property, or one side at a time with the padding-top, padding-right, padding-bottom and padding-left properties.

**border**

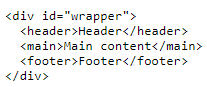
The border of a CSS box sits between the outer edge of the padding and the inner edge of the margin. By default the border has a size of 0 — making it invisible — but you can set the thickness, style and color of the border to make it appear. The border shorthand property allows you to set all of these on all four sides at once, for example border: 1px solid black. This can be broken down into numerous different longhand properties for more specific styling needs:

* border-top, border-right, border-bottom, border-left: Set the thickness, style and color of one side of the border.
* border-width, border-style, border-color: Set only the thickness, style, or color individually, but for all four sides of the border.
* You can also set one of the three properties of a single side of the border individually, using border-top-width, border-top-style, border-top-color, etc.

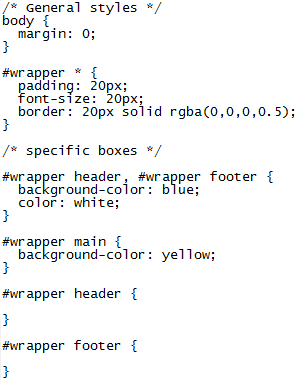
**margin**

The margin surrounds a CSS box, and pushes up against other CSS boxes in the layout. It behaves rather like padding; the shorthand property is margin and the individual properties are margin-top, margin-right, margin-bottom, and margin-left.

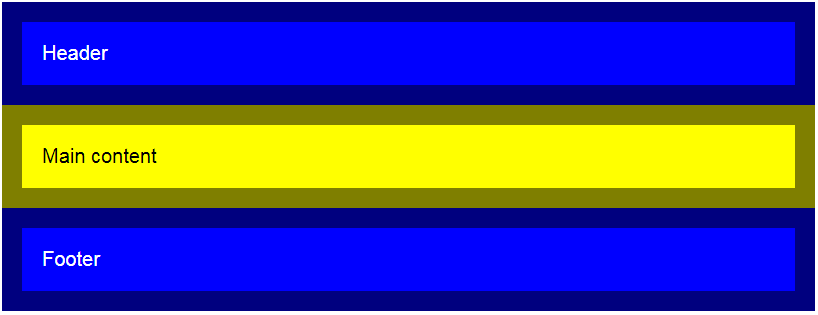
**HTML Input:**



**CSS Input:**



**Output:**

****

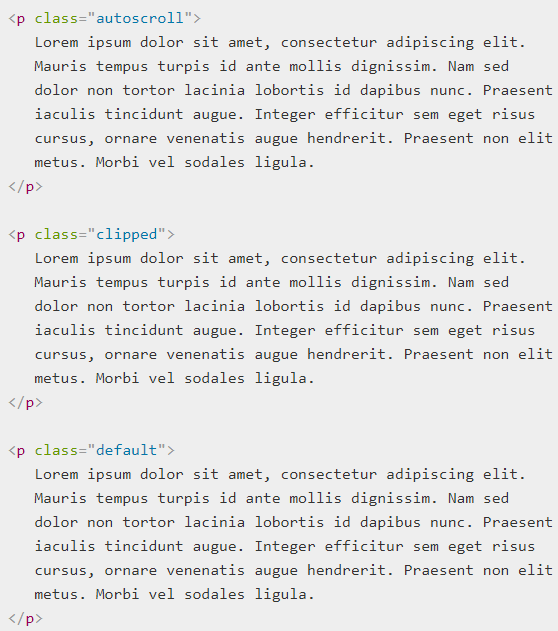
1. [How to control overflowing content](https://developer.mozilla.org/en-US/Learn/CSS/Introduction_to_CSS/Box_model#Overflow)

Overflow property takes several possible values, which are

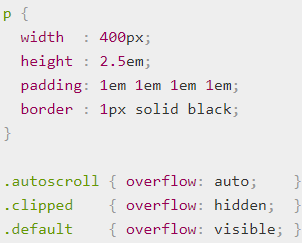
* **auto:** If there is too much content, the overflowing content is hidden and scroll bars are shown to let the user scroll to see all the content.
* **hidden:** If there is too much content, the overflowing content is hidden.
* **visible:** If there is too much content, the overflowing content is shown outside of the box (this is usually the default behavior.)

Here is a simple example to show how these settings work:

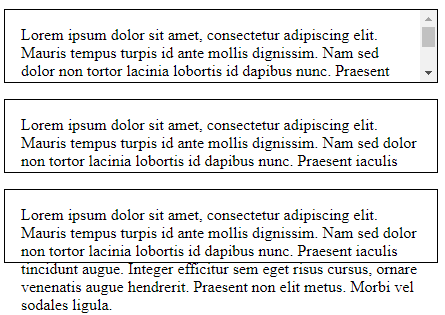
**HTML**

****

**CSS to apply to HTML:**

****

**Output:**



1. [How to control the part of a CSS box that the background is drawn under](https://developer.mozilla.org/en-US/Learn/CSS/Introduction_to_CSS/Box_model#Background_clip)

Box backgrounds are made up of colors and images, stacked on top of each other (background-color, background-image.) They are applied to a box and drawn under that box. By default, backgrounds extend to the outer edge of the border.

1. [How do I define inline, block, and inline-block?](https://developer.mozilla.org/en-US/Learn/CSS/Introduction_to_CSS/Box_model#Types_of_CSS_boxes)

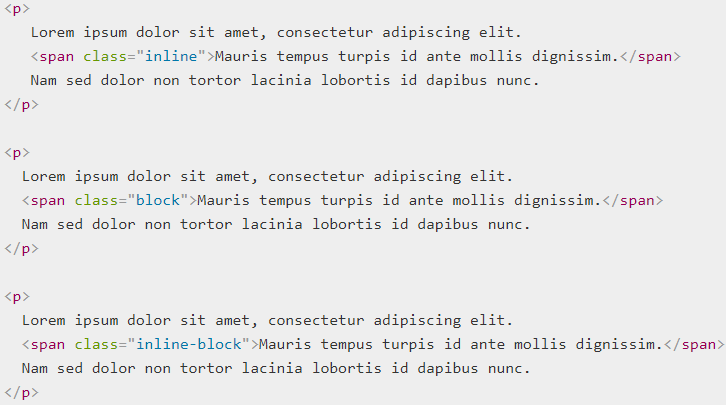
A **block box** is defined as a box that's stacked upon other boxes (i.e. content before and after the box appears on a separate line), and can have width and height set on it. The whole box model as described above applies to block boxes.

An **inline box** is the opposite of a block box: it flows with the document's text (i.e. it will appear on the same line as surrounding text and other inline elements, and its content will break with the flow of the text, like lines of text in a paragraph.) Width and height settings have no effect on inline boxes; any padding, margin and border set on inline boxes will update the position of surrounding text, but will not affect the position of surrounding block boxes.

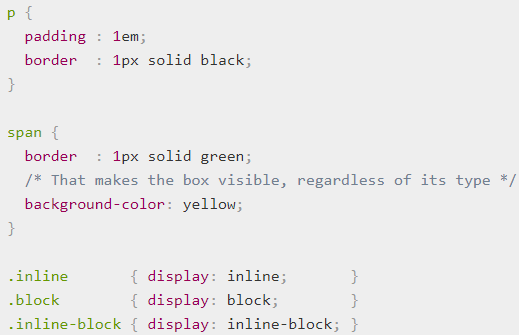
An **inline-block box** is something in between the first two: It flows with surrounding text without creating line breaks before and after it unlike a block box, but it can be sized using width and height and maintains its block integrity like a block box. It won't be broken across paragraph lines like an inline box. In the below example the inline-box goes onto the 2nd line of text while keeping the shape of a box as there is not enough space for it on the first line, whereas inline box does break on multiple lines if there is not enough space — it loses the shape of a box.

Let's take an simple example:

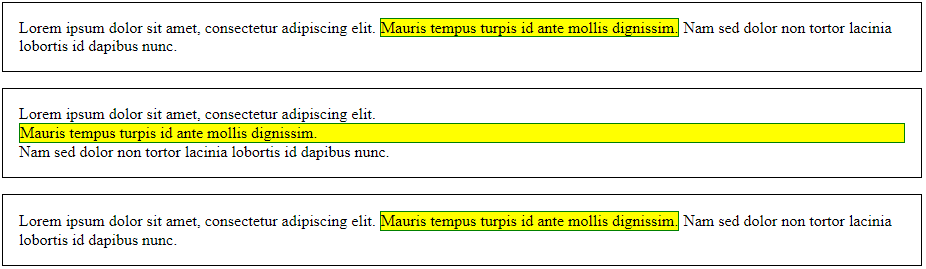
**HTML**

****

**let's add some CSS:**

****

**Output:**

****

1. [How to create fancy boxes](https://developer.mozilla.org/en-US/docs/Learn/CSS/Howto/create_fancy_boxes) (also see the [Styling boxes](https://developer.mozilla.org/en-US/docs/Learn/CSS/Styling_boxes) module, generally).

Creating fancy boxes are all about mastering CSS border and background properties and how to apply them to a given box.

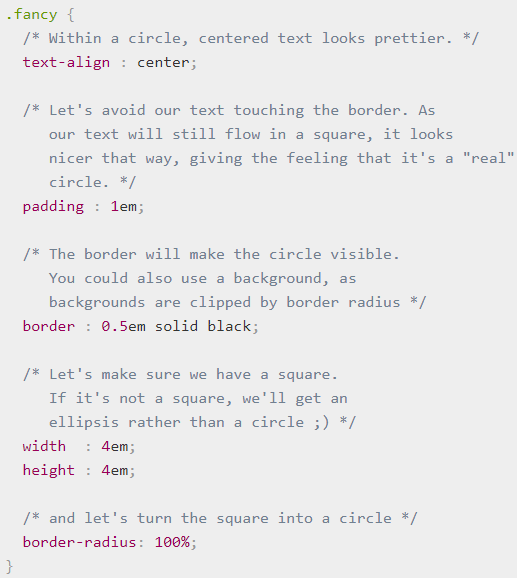
**Example:**

**HTML**

****

Ok, that's a very small bit of HTML, what can we tweak on that element? All of the following:

* Its box model properties: width, height, padding, border, etc.
* Its background properties: background, background-color, background-image, background-position, background-size, etc.
* Its pseudo-element: ::before and ::after
* and some aside properties like: box-shadow, transform, outline, etc.

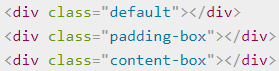


**Output:**

****

1. [How to use background-clip to control how much of the box your background image covers](https://developer.mozilla.org/en-US/Learn/CSS/Introduction_to_CSS/Box_model#Background_clip).

Let's have a look at an example, to see how this works. First, our HTML:



**CSS**



The above code produces the following result:



1. [How to change the box model completely using box-sizing](https://developer.mozilla.org/en-US/Learn/CSS/Styling_boxes/Box_model_recap#Changing_the_box_model_completely)

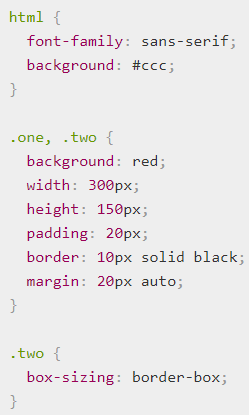
The total width of a box is the sum of its width, padding-right, padding-left, border-right, and border-left properties.

Let's look at an example. We will set up two identical <div> elements, giving each one the same width, border and padding. We will also use some JavaScript to print out the computed value (the final on-screen value in pixels) of the width for each box. The only difference is that we've given the bottom box box-sizing: border-box, but we've left the top box with its default behavior.

First**, the HTML:**



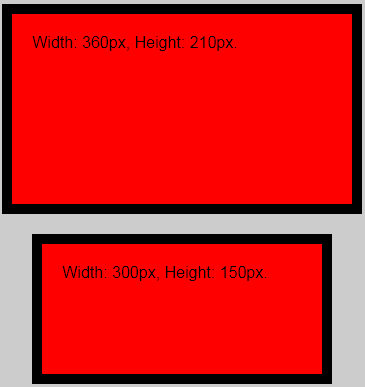
**CSS:**



JavaScript to measure the overall computed widths:



**Output:**



The width and height of the first box are equal to content + padding + border. The second box however has its width and height equal to the width and height set on the content via CSS. The padding and border haven't added onto the total width and height; instead, they've taken up some of the content's space, making the content smaller and keeping the total dimensions the same.

1. [How to control backgrounds](https://developer.mozilla.org/en-US/Learn/CSS/Styling_boxes/Backgrounds)

**Background:** The background of an element is the area that sits underneath an element's content, padding, and border. By default this is the case anyway — in newer browsers you can alter the area the background takes up, using the background-clip property

There are many different properties can use to manipulate the element's background.

* **background-color:** Sets a solid color for the background.
* **background-image:** Specifies a background image to appear in the background of the element. This can be a static file, or a generated gradient.
* **background-position:** Specifies the position that the background should appear inside the element background.
* **background-repeat:** Specifies whether the background should be repeated (tiled) or not.
* **background-attachment:** Specifies the behaviour of an element's background when its content scrolls, e.g. does it scroll with the content, or is it fixed?
* **background:** Shorthand for specifying the above five properties on one line.
* **background-size:** Allows a background image to be resized dynamically

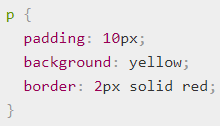
1. [How to control borders](https://developer.mozilla.org/en-US/Learn/CSS/Styling_boxes/Borders)

Border that sits comfortably between the element's padding and margin. By default the border has a size of 0 — making it invisible — but you can set the thickness, style and color of the border to make it appear.

**Border shorthand** - The border shorthand property allows you to set all of these on all four sides at once.

**For example**

****

****

**Output:**

****

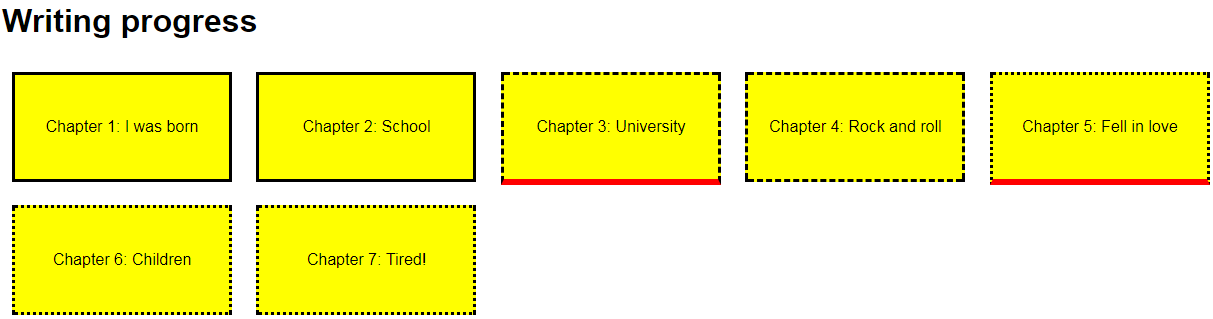
**Longhand options** - The border can be broken down into numerous different longhand properties for more specific styling needs:

* border-top, border-right, border-bottom, border-left: Set the thickness, style and color of one side of the border.
* border-width, border-style, border-color: Set only the thickness, style, or color individually, but for all four sides of the border.
* You can also set one of the three properties of a single side of the border individually, using border-top-width, border-top-style, border-top-color, etc.

Let's look at some CSS we could use to implement this:



**Output:**



**Border radius**

The border-radius property is used to add rounded corners to an element.

The border-radius property is a shorthand property for setting the four border-\*-radius properties.

If you specify only one value for the border-radius property, this radius will be applied to all 4 corners.

However, you can specify each corner separately if you wish. Here are the rules:

* **Four values**: first value applies to top-left, second value applies to top-right, third value applies to bottom-right, and fourth value applies to bottom-left corner
* **Three values:** first value applies to top-left, second value applies to top-right and bottom-left, and third value applies to bottom-right
* **Two values:** first value applies to top-left and bottom-right corner, and the second value applies to top-right and bottom-left corner
* **One value:** all four corners are rounded equally **Border Image -** [**border-image**](https://developer.mozilla.org/en-US/docs/Web/CSS/border-image) images makes it a lot easier to achieve complex patterned borders, albeit in modern browsers (Internet Explorer 11+ supports it, as well as other modern browsers.)

1. Four values - border-radius: 15px 50px 30px 5px:



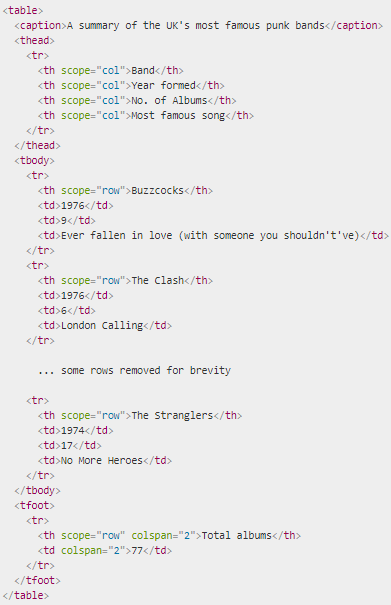
2. Three values - border-radius: 15px 50px 30px:



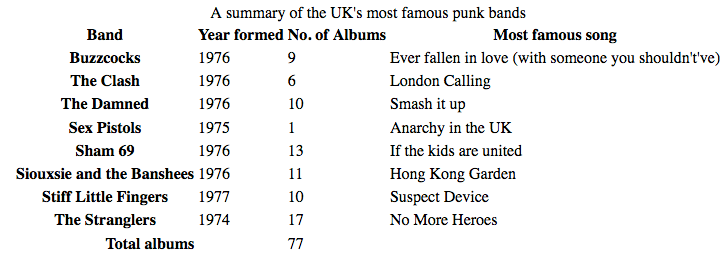
3. Two values - border-radius: 15px 50px:



1. [How to style an HTML table](https://developer.mozilla.org/en-US/Learn/CSS/Styling_boxes/Styling_tables)



The table is nicely marked up, easily styleable, and accessible, thanks to features such as scope, , <thead>, <tbody>, etc. Unfortunately, it doesn't look good when rendered on the screen.



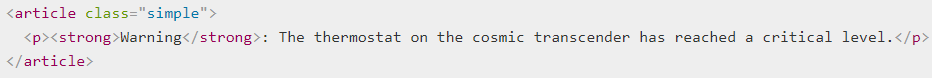
It just looks cramped, hard to read, and boring. We need to use some CSS to fix this up.

1. [How to add shadows to boxes](https://developer.mozilla.org/en-US/Learn/CSS/Styling_boxes/Advanced_box_effects#Box_shadows)

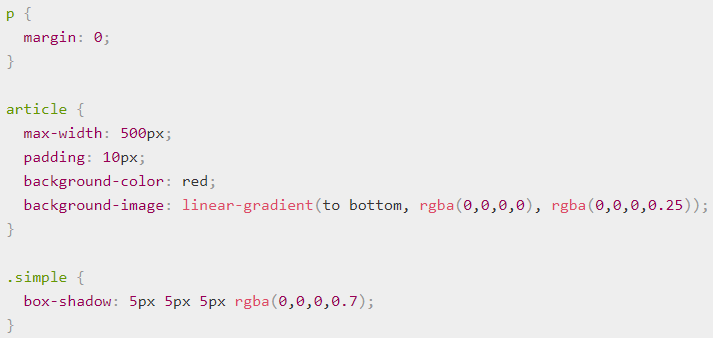
**box-shadow** allows you to apply one or more drop shadows to an actual element box. Like text shadows, box shadows are supported pretty well across browsers, but only in IE9+.

Let's look at an example

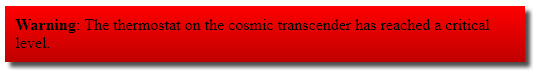
**HTML:**



**CSS**

****

**Output:**

****

There are four items in the box-shadow property value:

* The first length value is the horizontal offset — the distance to the right the shadow is offset from the original box (or left, if the value is negative).
* The second length value is the vertical offset — the distance downwards that the shadow is offset from the original box (or upwards, if the value is negative).
* The third length value is the blur radius — the amount of blurring applied to the shadow.
* The color value is the base color of the shadow.

**Uncommon or advanced techniques**

### General

1. [How to calculate specificity of a CSS selector](https://developer.mozilla.org/en-US/Learn/CSS/Introduction_to_CSS/Cascade_and_inheritance#Specificity)

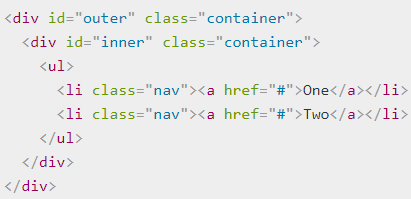
The amount of specificity a selector has is measured using four different values (or components), which can be thought of as thousands, hundreds, tens and ones — four single digits in four columns:

* **Thousands:** Score one in this column if the declaration is inside a style attribute (such declarations don't have selectors, so their specificity is always simply 1000.) Otherwise 0.
* **Hundreds:** Score one in this column for each ID selector contained inside the overall selector.
* **Tens:** Score one in this column for each class selector, attribute selector, or pseudo-class contained inside the overall selector.
* **Ones:** Score one in this column for each element selector or pseudo-element contained inside the overall selector.

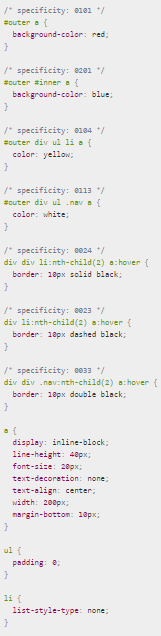
The following table shows a few isolated examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Selector | Thousands | Hundreds | Tens | Ones | Total specificity |
| h1 | 0 | 0 | 0 | 1 | 0001 |
| #important | 0 | 1 | 0 | 0 | 0100 |
| h1 + p::first-letter | 0 | 0 | 0 | 3 | 0003 |
| li > a[href=\*"en-US"] > .inline-warning | 0 | 0 | 2 | 2 | 0022 |
| #important div > div > a:hover, inside an element's [style](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes#attr-style)attribute | 1 | 1 | 1 | 3 | 1113 |

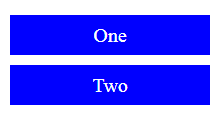
let's look at an example in action. Here is the HTML we are going to use:



And here is the CSS for the example:



**Output:**



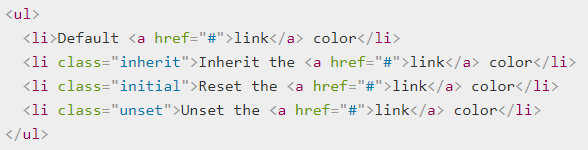
* The first two selectors are competing over the styling of the link's background color — the second one wins and makes the background color blue because it has an extra ID selector in the chain: its specificity is 201 versus 101.
* The third and fourth selectors are competing over the styling of the link's text color — the second one wins and makes the text white because although it has one less element selector, the missing selector is swapped out for a class selector, which is worth ten rather than one. So the winning specificity is 113 versus 104.
* Selectors 5–7 are competing over the styling of the link's border when hovered. Selector six clearly loses to five with a specificity of 23 versus 24 — it has one fewer element selectors in the chain. Selector seven, however, beats both five and six — it has the same number of sub-selectors in the chain as five, but an element has been swapped out for a class selector. So the winning specificity is 33 versus 23 and 24.

1. [How to control inheritance in CSS](https://developer.mozilla.org/en-US/Learn/CSS/Introduction_to_CSS/Cascade_and_inheritance#Controlling_inheritance)

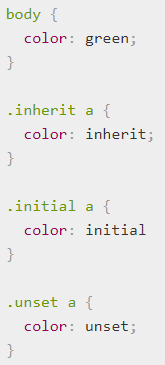
CSS provides three special universal property values for handling inheritance:

* **inherit :** This value sets the property value applied to a selected element to be the same as that of its parent element.
* **initial :** This value sets the property value applied to a selected element to be the same as the value set for that element in the browser's default style sheet. If no value is set by the browser's default style sheet and the property is naturally inherited, then the property value is set to inherit instead.
* **unset :** This value resets the property to its natural value, which means that if the property is naturally inherited it acts like inherit, otherwise it acts like initial.

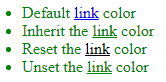
Let's take a look at an example. First HTML:



Now CSS for styling:



**Output:**

****

Let's explain what's going on here:

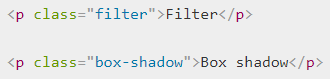
* We first set the color of the <body> to green.
* As the color property is naturally inherited, all child elements of body will have the same green color. It's worth noting that browsers set the color of links to blue by default instead of allowing the natural inheritance of the color property, so the first link in our list is blue.
* The second rule sets links within an element with the class inherit to inherit its color from its parent. In this case, it means that the link inherits its color from its <li> parent, which, by default inherits its color from its own <ul> parent, which ultimately inherits its color from the <body> element, which had its color set to green by the first rule.
* The third rule selects any links within an element with the class initial and sets their color to initial. Usually, the initial value set by browsers for the text color is black, so this link is set to black.
* The last rule selects all links within an element with the class unset and sets their color to unset — we unset the value. Because the color property is a naturally inherited property it acts exactly like setting the value to inherit. As a consequence, this link is set to the same color as the body — green.

### Advanced effects

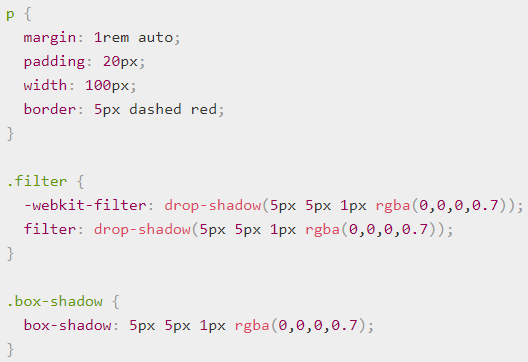
1. [How to use filters in CSS](https://developer.mozilla.org/en-US/Learn/CSS/Styling_boxes/Advanced_box_effects#Filters)

CSS Filters provide a way to apply filters to an element in the same way as you might apply a filter to a layer in a graphics package like Photoshop. There are a number of different options available, and you can read all about them in greater detail on the filter reference page.

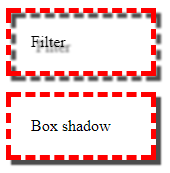
**HTML**



**And now some CSS to a apply a drop shadow to each:**



**Output:**



As you can see, the filter drop-shadow follows the exact shape of the text and border dashes. The box shadow just follows the square of the box.

Some other things to note:

* Filters are very new — they are supported in most modern browsers, including Microsoft Edge, but they are not supported in Internet Explorer at all. If you use filters in your designs, you should therefore make sure your content is usable without the filters.
* You'll see that we've included a version of the filter property with -webkit- prefixed. This is called a Vendor Prefix, and is used sometimes by a browser before it finalizes its implementation of a new feature, to support that feature and experiment with it while not clashing with the non-prefixed version. Vendor prefixes were never intended to be used by web developers, but they do get used sometimes on production pages if experimental features are really desired. In this case, the -webkit- version of the property is currently required for Chrome/Safari/Opera support, while Edge and Firefox use the final, non-prefixed version.

1. [How to use blend modes in CSS](https://developer.mozilla.org/en-US/Learn/CSS/Styling_boxes/Advanced_box_effects#Blend_modes)

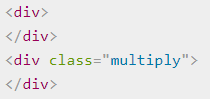
CSS blend modes allow us to add blend modes to elements that specify a blending effect when two elements overlap — the final color shown for each pixel will be the result of a combination of the original pixel color, and that of the pixel in the layer underneath it. Blend modes are again very familiar to users of graphics applications like Photoshop.

There are two properties that use blend modes in CSS:

* **background-blend-mode**, which blends together multiple background images and colors set on a single element.
* **mix-blend-mode**, which blends together the element it is set on with elements it is overlapping — both background and content.

**background-blend-mode**

**HTML**

****

**CSS — we are adding two background images to the <div>s: a linear gradient and a png:**

****

**Output:**

****

**mix-blend-mode**



**CSS**

****

**Output:**

****

### Layout

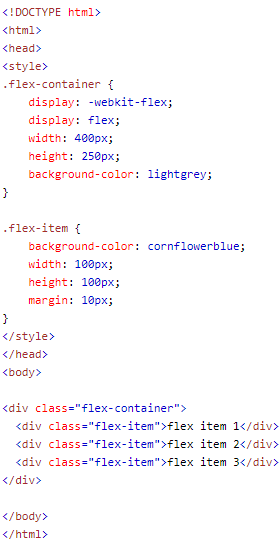
1. [Using CSS flexible boxes](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Flexible_boxes)

Flexible boxes, or flexbox, is a new layout mode in CSS3.

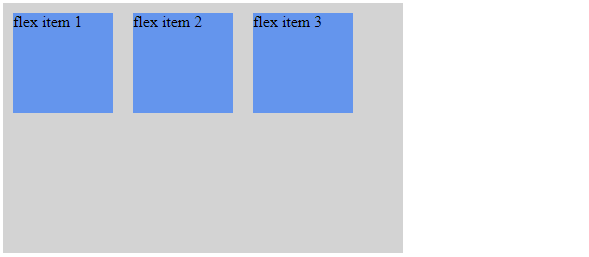
Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.

For many applications, the flexible box model provides an improvement over the block model in that it does not use floats, nor do the flex container's margins collapse with the margins of its contents.

The following example shows three flex items. They are positioned by default: along the horizontal flex line, from left to right:

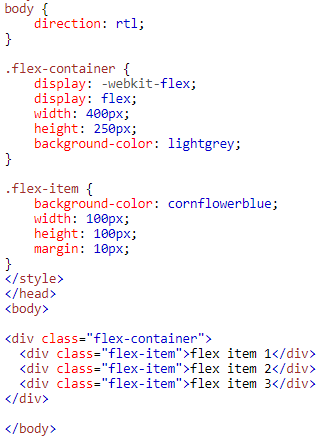


**Output:**

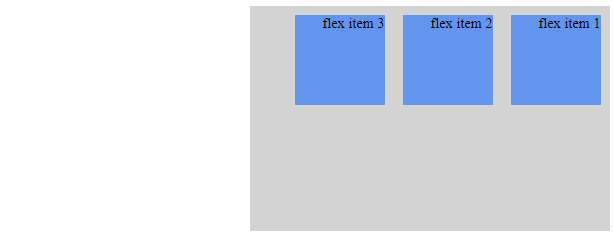
****

It is also possible to change the direction of the flex line.

If we set the direction property to rtl (right-to-left), the text is drawn right to left, and also the flex line changes direction, which will change the page layout:



**Output:**

****

1. [Using CSS multi-column layouts](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Using_multi-column_layouts)

CSS Multi-column Layout Module extends the block layout mode to allow the easy definition of multiple columns of text.

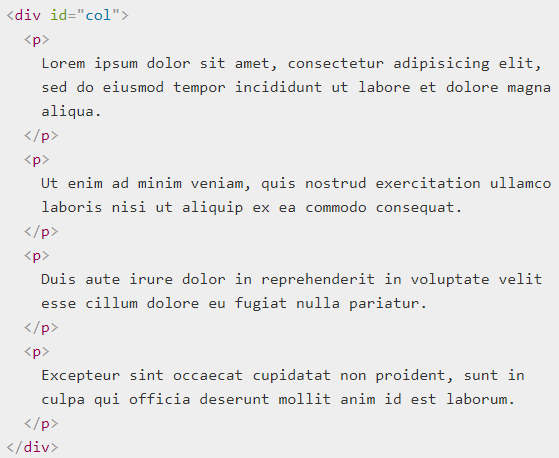
**Column count and width**

Two CSS properties control whether and how many columns will appear: column-count and column-width.

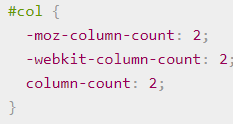
The column-count property sets the number of columns to a particular number. E.g.,

**Example 1**

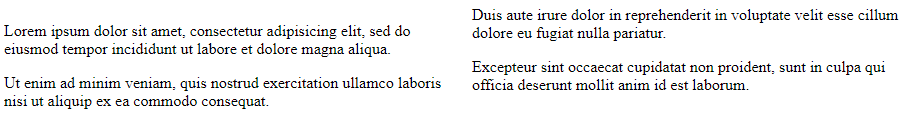
**HTML**

****

**CSS**

****

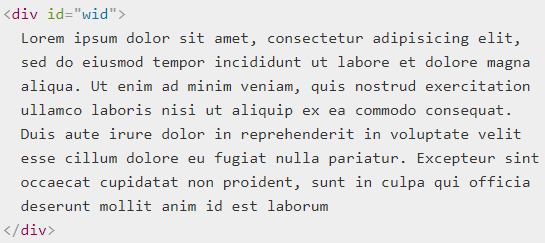
**Output:**

****

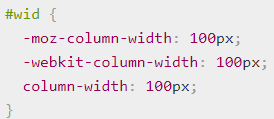
The column-width property sets the minimum desired column width. If column-count is not also set, then the browser will automatically make as many columns as fit in the available width.

**Example 2**

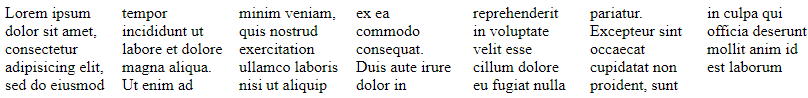
**HTML**

****

**CSS**

****

**Output:**

****

In a multi-column block, content is automatically flowed from one column into the next as needed. All HTML, CSS and DOM functionality is supported within columns, as are editing and printing.

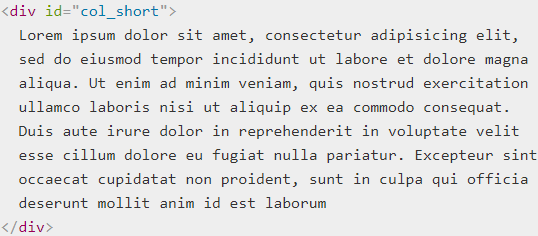
**The columns shorthand**

Web designer will use one of the two CSS properties: column-count or column-width. As values for these properties do not overlap, it is often convenient to use the shorthand columns. E.g.

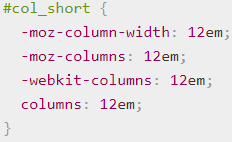
The CSS declaration column-width:12em can be replaced by:

**Example 3**

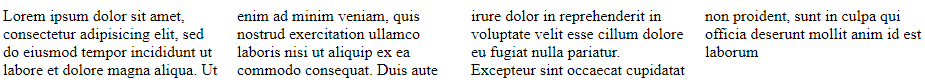
**HTML**



**CSS**

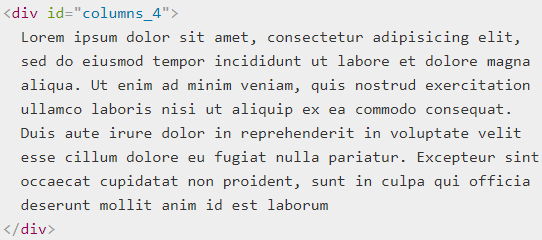
****

**Output:**

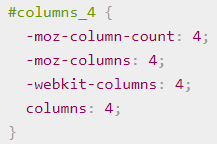
****

**Example 4**

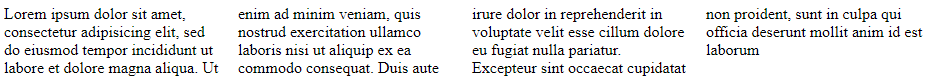
**HTML**

****

**CSS**

****

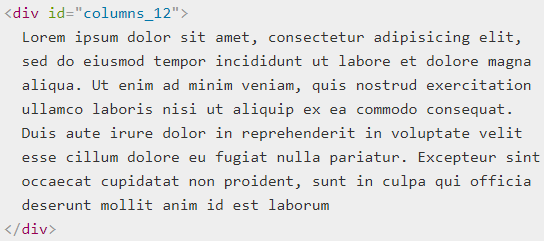
**Output:**

****

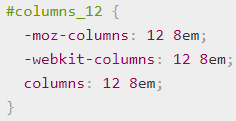
The two CSS declarations column-width:8em and column-count:12 can be replaced by:

**Example 5**

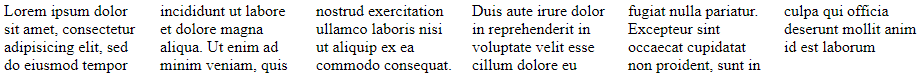
**HTML**

****

**CSS**

****

**Output:**

****

**Height Balancing**

The CSS3 Column specification requires that the column heights must be balanced: that is, the browser automatically sets the maximum column height so that the heights of the content in each column are approximately equal. Firefox does this.

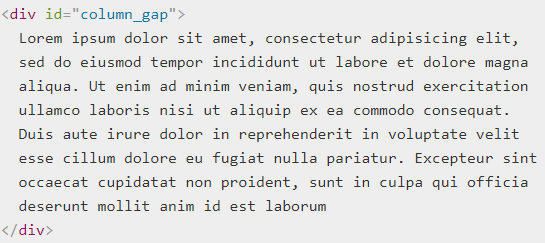
However, in some situations it is also useful to set the maximum height of the columns explicitly, and then lay out content starting at the first column and creating as many columns as necessary, possibly overflowing to the right. Therefore, if the height is constrained, by setting the CSS height or max-height properties on a multi-column block, each column is allowed to grow to that height and no further before adding new column. This mode is also much more efficient for layout.

**Column Gaps**

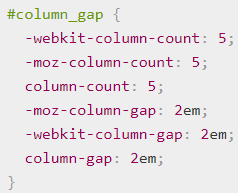
There is a gap between columns. The recommended default is 1em. This size can be changed by applying the column-gap property to the multi-column block:

**Example 6**

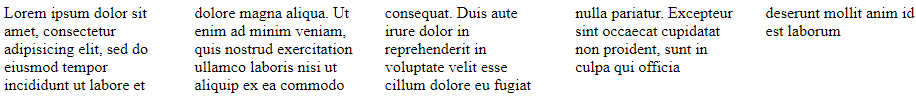
**HTML**

****

**CSS**

****

**Output:**

****

1. [Using CSS generated content](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started/Content)

Content specified in a stylesheet can consist of text or images.

**Text content**

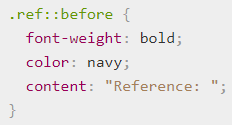
CSS can insert text content before or after an element. To specify this, make a rule and add ::before or ::after to the selector. In the declaration, specify the content property with the text content as its value.

**Example**

**HTML**

****

**CSS**

****

**Output:**

****

**Image content**

To add an image before or after an element, you can specify the URL of an image file in the value of the content property.

**Example**

This rule adds a space and an icon after every link that has the class glossary:

